Copyright

by

Josiah Paul Hanna

2019

The Dissertation Committee for Josiah Paul Hanna
certifies that this is the approved version of the following dissertation:

Data Efficient Reinforcement Learning with Off-policy and Simulated Data

Committee:

Peter Stone, Supervisor

Scott Niekum

Philipp Krähenbühl

Richard Sutton

Data Efficient Reinforcement Learning with Off-policy and Simulated Data

by

Josiah Paul Hanna

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 2019

To Aimee for her sacrifices, encouragement, and
endless support at every step along the way.

Acknowledgments

Completing this dissertation would have been impossible without the support of many people. First, I am very grateful to Peter Stone for his guidance, mentorship, and support. He has given me freedom to pursue my research interests and taught me how to think critically about the work we have done together. Peter is deeply committed to the success of all his students, and was always ready and willing to give advice for every facet of research work and professional life. I have learned so much from having the opportunity to have him as my adviser and for that I am very grateful.

Throughout my Ph.D., I've been very fortunate to work with Scott Niekum and to have him on my dissertation committee. Even though I was not in his lab, Scott has had an open door for advice and discussion throughout my time at UT Austin. I am grateful for his guidance both professionally and in research.

I am also very grateful for the other members of my dissertation committee whose feedback greatly improved the my dissertation. Rich Sutton provided detailed feedback, particularly showing me how to be very precise in placing my work in the wider reinforcement learning literature. Philipp Krähenbühl provided valuable feedback and also spent a lot of his time helping me prepare for academic job interviews.

Aside from my committee, I am grateful to have had advice and guidance from other senior researchers and faculty mentors who have helped me progress as a

young researcher. I would like to thank Craig Boutillier for hosting me during an internship at Google and I am very grateful for all that I learned from him there and since. I would like to thank Steve Boyles for his guidance throughout my Ph.D. and help preparing me for an academic career. I would like to thank Philip Thomas for his time spent providing feedback on my research and mathematical writing.

I have been fortunate to have had the support of my fellow computer science students, Learning Agents Research Group lab members, and members of the UT Austin Villa RoboCup team. I have made many friends while at UT Austin who have made the path to my Ph.D. much easier and much more fun. In particular, I would like to thank Michael Albert, Stefano Albrecht, Shani Alkoby, Daniel Brown, Donna Chen, Sid Desai, Ishan Durugkar, John Fang, Katie Genter, Wonjoon Goo, Harsh Goyal, Xiang Gu, Justin Hart, Matthew Hausknecht, Eddy Hudson, Joel Iventosch, Ajinkya Jain, Yuqian Jiang, Haresh Karnan, Josh Kelle, Piyush Khandewal, Matteo Leonetti, Michael Levin, Elad Liebman, Rudi Lioutikov, Shih-yun Lo, Patrick MacAlpine, Jake Menashe, Reuth Mirsky, Prabhat Nagarajan, Sanmit Narvekar, Aishwarya Padmakumar, Brahma Pavse, Tarun Rambha, Ashay Rane, Jivko Sinapov, Guni Sharon, Jesse Thomason, Faraz Torabi, Daniel Urieli, Xinyu Wang, Garrett Warnell, Harel Yeddison, Ruohan Zhang, and Shiqi Zhang.

Before beginning my Ph.D., I was very fortunate to have the mentorship of Judy Goldsmith at the University of Kentucky. I owe a great deal to her for her guidance and the opportunities she opened up for me. I am also very grateful for Patrice Perny and Paul Weng who hosted me for a summer at l'Université Pierre et Marie Curie and gave me much guidance as I began learning how to conduct research. I also would like to thank Ken Calvert, Philip Kraemer, Brent Seales, and Pat Whitlow for their mentorship while at the University of Kentucky and for their continued support since then.

I am very grateful to the staff at UT Austin who have supported me throughout

graduate school. Stacy Miller has done so much to make travel and graduate student life much easier. The UTCS IT staff have saved me countless hours of setting up and maintaining software for research. And Katie Dahm has spent much time answering my many questions about completing degree requirements.

I have been blessed with a supportive family, without whom I never would have been able to start and complete a Ph.D. I would like to thank my parents, David and Sarah Hanna, for always fostering a sense of curiosity and encouraging me in any pursuit, my brothers, Daniel and John Hanna, who have always challenged me to work hard and learn new things, and Kevin, Regi, and Michael Goffinet and Kendall and Taylor Foley for their belief in me and for always being ready to celebrate both big and small accomplishments.

Finally, I could never have completed my Ph.D. without the support of my wife, Aimee Hanna. I am very grateful for how she has helped me grow, the encouragement she provides, and sacrifices she has made. She has always been ready with advice and ready to help me think through problems. She has been a supportive, loving partner through every challenge and without her I would not be where I am today.

Josiah Paul Hanna

The University of Texas at Austin
December 2019

Data Efficient Reinforcement Learning with Off-policy and Simulated Data

Publication No. _____

Josiah Paul Hanna, Ph.D.

The University of Texas at Austin, 2019

Supervisor: Peter Stone

Learning from interaction with the environment – trying untested actions, observing successes and failures, and tying effects back to causes – is one of the first capabilities we think of when considering autonomous agents. Reinforcement learning (RL) is the area of artificial intelligence research that has the goal of allowing autonomous agents to learn in this way. Despite much recent success, many modern reinforcement learning algorithms are still limited by the requirement of large amounts of experience before useful skills are learned. Two possible approaches to improving data efficiency are to allow algorithms to make better use of past experience collected with past behaviors (known as *off-policy* data) and to allow algorithms to make better

use of simulated data sources. This dissertation investigates the use of such auxiliary data by answering the question, **"How can a reinforcement learning agent leverage off-policy and simulated data to evaluate and improve upon the expected performance of a policy?"**

This dissertation first considers how to directly use off-policy data in reinforcement learning through importance sampling. When used in reinforcement learning, importance sampling is limited by high variance that leads to inaccurate estimates. This dissertation addresses this limitation in two ways. First, this dissertation introduces the *behavior policy gradient algorithm* that adapts the data collection policy towards a policy that generates data that leads to low variance importance sampling evaluation of a fixed policy. Second, this dissertation introduces the family of *regression importance sampling estimators* which improve the weighting of already collected off-policy data so as to lower the variance of importance sampling evaluation of a fixed policy. In addition to evaluation of a fixed policy, we apply the behavior policy gradient algorithm and regression importance sampling to *batch policy gradient* policy improvement. In the case of regression importance sampling, this application leads to the introduction of the *sampling error corrected policy gradient estimator* that improves the data efficiency of batch policy gradient algorithms.

Towards the goal of learning from simulated experience, this dissertation introduces an algorithm – the *grounded action transformation algorithm* – that takes small amounts of real world data and modifies the simulator such that skills learned in simulation are more likely to carry over to the real world. Key to this approach is the idea of local simulator modification – the simulator is automatically altered to better model the real world for actions the data collection policy would take in states the data collection policy would visit. Local modification necessitates an iterative approach: the simulator is modified, the policy improved, and then more data is collected for further modification.

Finally, in addition to examining them each independently, this dissertation also considers the possibility of combining the use of simulated data with importance sampled off-policy data. We combine these sources of auxiliary data by control variate techniques that use simulated data to lower the variance of off-policy policy value estimation. Combining these sources of auxiliary data allows us to introduce two algorithms – *weighted doubly robust bootstrap* and *model-based bootstrap* – for the problem of lower-bounding the performance of an untested policy.

Contents

Acknowledgments v

Abstract viii

List of Tables xvii

List of Figures xviii

Chapter 1 Introduction 1

1.1 Importance Sampling 4

1.2 Leveraging Simulation 6

1.3 Simulation and Importance Sampling 8

1.4 Contributions 9

1.5 Dissertation Overview 10

Chapter 2 Background 13

2.1 Reinforcement Learning Notation 14

2.2 Policy Value Estimation 16

2.2.1 Objectives for Policy Value Estimation 17

2.2.2 Variance, Bias, and Consistency 18

2.2.3 Three Classes of Policy Value Estimators 19

2.3 Policy Improvement 25

2.4 Summary . . . 28

Chapter 3 Collecting Data for Off-policy Policy Value Estimation 29

3.1 Incremental Policy Value Estimation . . . 30
3.2 The Optimal Variance Behavior Policy . . . 31
3.3 The Behavior Policy Search Problem . . . 32
3.4 Behavior Policy Gradient Theorem . . . 34
3.4.1 Control Variate Extension . . . 36
3.4.2 Connection to REINFORCE . . . 37
3.5 Empirical Study . . . 38
3.5.1 Empirical Set-up . . . 39
3.5.2 Empirical Results . . . 40
3.5.3 Control Variate Extension Results . . . 43
3.5.4 Rareness of Event Study . . . 46
3.6 Summary . . . 47

Chapter 4 Collecting Data for Off-Policy Policy Improvement 48

4.1 Importance Sampled Batch Policy Gradient . . . 49
4.2 Batch Policy Gradient Estimation with an Improved Behavior Policy 50
4.3 Parallel Policy Search: Towards a Full Algorithm . . . 51
4.4 Empirical Study . . . 51
4.4.1 Empirical Set-up . . . 52
4.4.2 Empirical Results . . . 54
4.5 Challenges for an Efficient Algorithm . . . 56
4.5.1 Variance Reduction vs. Exploration . . . 56
4.5.2 Synchronizing Target and Behavior Policy Updates . . . 58
4.6 Summary . . . 58

Chapter 5 Weighting Data for Off-policy Policy Value Estimation 60
5.1 Limitations of Ordinary Importance Sampling 61
5.2 Regression Importance Sampling 63
5.2.1 Correcting Sampling Error in Discrete Action Spaces 65
5.2.2 Correcting Sampling Error in Continuous Action Spaces . . . 67
5.3 Theoretical Analysis 70
5.4 RIS with Function Approximation 71
5.5 Empirical Study 72
5.5.1 Empirical Set-up 72
5.5.2 Empirical Results 73
5.6 Summary 81

Chapter 6 Weighting Data for Policy Improvement 83
6.1 Sampling Error in Batch Policy Gradient Learning 84
6.2 The Sampling Error Corrected Policy Gradient Estimator 87
6.3 Theoretical Analysis 90
6.4 Empirical Study 92
6.4.1 Empirical Set-up 93
6.4.2 Empirical Results 96
6.5 Summary 100

Chapter 7 Learning with Simulated Data 102
7.1 Learning in a Simulator 103
7.2 Grounded Simulation Learning 104
7.3 The Grounded Action Transformation Algorithm 106
7.4 Empirical Study 110
7.4.1 Empirical Set-up 110
7.4.2 Empirical Results 117

7.5 Summary . . . 124

Chapter 8 Combining Off-policy Data with Simulated Data 125

8.1 Confidence Intervals for Off-policy Value Estimation . . . 126
8.2 Off-Policy Bootstrapped Lower Bounds . . . 127
8.2.1 Model-based Bootstrap . . . 129
8.2.2 Weighted Doubly Robust Bootstrap . . . 130
8.3 Theoretical Analysis: When is Model Error High? . . . 132
8.4 Empirical Analysis . . . 135
8.4.1 Experimental Set-up . . . 135
8.4.2 Experimental Results . . . 137
8.5 Summary . . . 140

Chapter 9 Related Work 142

9.1 Sampling Off-Policy Data . . . 142
9.1.1 Adaptive Importance Sampling in Reinforcement Learning . . . 143
9.1.2 Policy Improvement with Adaptive Importance Sampling . . . 144
9.2 Weighting Off-Policy Data . . . 144
9.2.1 Importance Sampling with an Estimated Behavior Policy . . . 144
9.2.2 Exact Expectation Methods . . . 148
9.3 Learning with Simulated Data . . . 150
9.3.1 Simulator Modification . . . 150
9.3.2 Robustness through Simulator Variance . . . 152
9.3.3 Simulator as Prior Knowledge . . . 153
9.3.4 Reality Gap in the Observation Space . . . 153
9.4 Combining Simulation and Importance Sampling . . . 154
9.4.1 High Confidence Off-Policy Policy Value Estimation . . . 154
9.4.2 Bootstrapping in Reinforcement Learning . . . 155

9.5 Policy Evaluation vs. Policy Value Estimation 156
9.6 Summary 158

Chapter 10 Conclusion and Future Work **159**
10.1 Contributions 161
10.2 Future Work 163
10.2.1 Sampling Off-Policy Data 163
10.2.2 Weighting Off-Policy Data 167
10.2.3 Learning with Simulated Data 171
10.2.4 Combining Off-Policy and Simulated Data 173
10.3 Concluding Remarks 175

Appendix A Notation Summary **176**

Appendix B Acronym Summary **179**

Appendix C Collecting Off-Policy Data: Derivations and Proofs **182**
C.1 Behavior Policy Gradient Theorem 182
C.1.1 MSE Gradient for an Unbiased Off-Policy Policy Evaluation Method 183
C.1.2 Behavior Policy Gradient Theorem 185
C.1.3 MSE Gradient for the Doubly Robust Estimator 186
C.1.4 MSE Gradient for the Per-Decision Importance Sampling Estimator 188
C.2 The Behavior Policy Gradient Algorithm and the Cross-Entropy Method 188
C.3 Behavior Policy Gradient and Unbiasedness 191

Appendix D Weighting Off-Policy Data: Derivations and Proofs **194**
D.1 Regression Importance Sampling is Consistent 194
D.1.1 Definitions and Assumptions 195

D.1.2 Consistency Proof . . . 197
D.2 Asymptotic Variance of Regression Importance Sampling . . . 201
D.3 Connection between RIS and REG . . . 204
D.4 Sampling Error Corrected Policy Gradient Estimator Variance . . . 206

Appendix E Model-based Error: Derivations and Proofs **210**
E.1 Error Bound in Terms of Trajectories . . . 210
E.1.1 On-Policy Model Error Bound . . . 211
E.1.2 Off-Policy Model Error Bound . . . 212
E.1.3 Bounding Theorem 8.1 in terms of a Supervised Loss Function 212
E.2 Finite-sample Error Bound . . . 214

Appendix F Additional Empirical Details **216**
F.1 Chapter 3: Behavior Policy Search . . . 216
F.1.1 Grid World . . . 216
F.1.2 Continuous Control . . . 217
F.1.3 Domain Independent Details . . . 218
F.2 Chapter 4: Parallel Policy Search . . . 218
F.3 Chapter 5: Regression Importance Sampling . . . 219
F.3.1 SinglePath . . . 220
F.3.2 Grid World . . . 220
F.3.3 Linear Dynamical System . . . 221
F.3.4 Continuous Control . . . 221
F.4 Chapter 8: Combining Simulated with Off-Policy Data . . . 222
F.4.1 Mountain Car . . . 222
F.4.2 Cliff World . . . 222

Bibliography **225**

List of Tables

4.1 Improvement in expected return after a single iteration for on-policy and off-policy batch policy gradient methods. 54

7.1 The initial parameters of the UNSW walk engine. 116

7.2 Maximum learned velocity and percent improvement for the grounded action transformation algorithm when optimizing the UNSW walk engine. 122

7.3 Comparison of the grounded action transformation to baselines for optimizing the UNSW walk engine for transfer from SimSpark to Gazebo. 123

List of Figures

1.1 Illustration of importance sampling 5
1.2 Chapter dependencies. 12

3.1 Behavior policy gradient reduction of mean squared error on the Grid World domain. 41
3.2 Behavior policy gradient reduction of variance on the Grid World domain. 42
3.3 Behavior policy gradient step-size sensitivity 43
3.4 Behavior policy gradient reduction of mean squared error on the Cart Pole Swing Up and Acrobot domains. 44
3.5 Doubly robust behavior policy gradient reduction of mean squared error on the Grid World domain and how the rareness of an extreme event affects reduction of mean squared error by behavior policy gradient. 45

4.1 Multi-step policy gradient learning with an optimized behavior policy. 55
4.2 Learning curves for parallel policy search and trust-region policy optimization on the Cart Pole problem. 57

5.1 Illustration of sampling error in a discrete environment with three possible trajectories. 62

5.2 Illustration of correcting sampling error with regression importance sampling in a continuous action domain. 69
5.3 The Single Path MDP 74
5.4 Comparison of regression importance sampling with alternatives in the Grid World domain. 76
5.5 Comparison of different n for RIS(n) in the Single Path MDP. . . . 77
5.6 Comparison of regression importance sampling with alternatives in the Linear Dynamical System domain. 78
5.7 Comparison of the mean squared error of RIS(0) to OIS with different neural network architectures in the Hopper and HalfCheetah domains. 79
5.8 Mean squared error of RIS throughout behavior policy estimation in the Hopper and HalfCheetah domains. 81

6.1 Illustration of sampling error in a Grid World domain. 87
6.2 The Grid World, Mountain Car, Linear Dynamical System and Cart Pole domains used to study the sampling error corrected policy gradient estimator. 93
6.3 Neural network policy representation that shares layers for empirical policy learning. 96
6.4 Learning curves for the sampling error corrected policy gradient estimator and batch Monte Carlo policy gradient estimator on the Linear Dynamical System and Cart Pole domains. 97
6.5 Learning curves for the sampling error corrected policy gradient estimator and batch Monte Carlo policy gradient estimator on the Mountain Car domain for different batch sizes. 98
6.6 Grid World ablation studies for the sampling error corrected policy gradient estimator. 100

7.1 Diagram of the grounded simulation learning framework. 107
7.2 The augmented simulator induced by the grounded action transformation algorithm. 108
7.3 The physical Softbank NAO and its simulated model in the Gazebo and SimSpark simulators. 111
7.4 Diagram of the degrees of freedom of the NAO robot. 111
7.5 Average performance of policies learned in simulation with and without simulator grounding. 118
7.6 Visualization of the robot's LeftShoulderPitch degree of freedom in Gazebo, SimSpark, and the grounded SimSpark simulator. 119
7.7 Visualization of the robot's right AnklePitch joint while walking in the real world, simulation, and the grounded simulation. 121

8.1 Illustration of the Cliff World domain. 135
8.2 Average empirical lower bound for different confidence interval methods on the Mountain Car and Cliff World domains. 138
8.3 Empirical error rate of different confidence interval methods on the Mountain Car and Cliff World domains. 140

10.1 RoboCup 2D and 3D Simulators 172

F.1 The SinglePath MDP used in Chapter 5. 220
F.2 The Cliff World domain used in Chapter 8. 223

Chapter 1

Introduction

Sequential decision-making tasks are among the most challenging tasks in the field of artificial intelligence. Examples of sequential decision-making tasks include a robot picking up and folding laundry, software choosing when to run vents to cool a data center, and a web marketing system choosing the ads to show a user in order to maximize the long-term likelihood that the user buys a product. In such tasks, the decision-making agent must repeatedly choose actions in order to maximize long-term expected utility. While expert engineers may be able to program robots and software agents to perform sequential decision-making tasks in constrained environments, the unstructured nature of the real world requires systems that can learn and generalize their experience to new situations.

Reinforcement learning (RL) algorithms provide a promising alternative to hand-coded skills, allowing sequential decision-making agents to acquire skills autonomously given only a reward function measuring task performance (Sutton and Barto, 1998). An agent using an RL algorithm attempts to maximize its expected reward obtained over time by learning an action-selection policy. To find the optimal policy, an RL algorithm must reason about a combinatorially large number of action sequences, explore the effects of untested action sequences, and assign credit or blame

for delayed effects to past actions.

Recently, RL has had many empirical successes, e.g., Levine et al. (2016); Mnih et al. (2015); Silver et al. (2016); MacAlpine et al. (2015). However, the majority of these successes have taken place within *simulated environments* where the simulation is based on idealized models of the real world. Unfortunately, a large gap exists between the amount of experience required by some of the most successful RL algorithms and the reality of collecting that experience on a physical system. For example, two leading RL methods – DDPG (Lillicrap et al., 2015) and PPO (Schulman et al., 2017) – require thousands of episodes of experience which is impractical for many applications. Aside from the time needed to collect this experience, the real world may be non-stationary so that the environment is changing while the agent is trying to learn. For instance, a robot's joints may wear down while in web-marketing the actions available to the agent may change as new ad campaigns begin and end. Furthermore, in the real world, unsafe actions may harm the agent, environment, or even humans.

One reason for the large gap between success in simulation and success in the real world is that many RL algorithms are limited to using experience collected with the most recently learned policy. This characteristic is known as being *on-policy*. In contrast, *off-policy* RL exploits data from a different data collection policy – off-policy data – to evaluate and improve upon the current policy. One of the most widely applied techniques for the direct use of off-policy data is a statistical technique called *importance sampling*. Unfortunately, importance sampling is known to suffer from high variance which may make it unreliable in practice (Thomas et al., 2015a).

In addition to challenges with using off-policy data, many RL algorithms are unable to incorporate simulated data. Many problem domains have existing simulators which could allow an RL agent to supplement real world experience with synthetic data collected in simulation. However, if an RL agent attempts to learn

directly from such simulated data, it may learn policies that are over-specialized to the simulated environment. Unfortunately, it is frequently observed that even small differences between simulated environments and the real world may cause behaviors learned in simulation to fail when applied in the real world (Abbeel et al., 2006; Cutler et al., 2014; Kober et al., 2013).

Creating techniques that allow RL algorithms to exploit *off-policy* and simulated data could reduce the sample complexity of real world reinforcement learning. This dissertation investigates how such auxiliary data can be used to increase the data-efficiency of learning and evaluating policies for sequential decision-making tasks. We study a specific instance of the RL setting – that of an episodic, fully observable Markov decision process. Within this setting, this dissertation answers the question:

How can a reinforcement learning agent leverage off-policy and simulated data to evaluate and improve upon the expected performance of a policy?

This dissertation answers this question in the following ways:

1. Showing how an RL agent should *collect* off-policy data for low variance importance-sampling-based policy value estimation and learning;

2. Showing how an RL agent should *weight* off-policy data for low variance importance-sampling-based policy value estimation and learning;

3. Showing how an RL agent can use simulated experience for policy learning; and

4. Showing how an RL agent can combine off-policy importance-sampled data and simulated experience for high confidence policy value estimation.

While *policy improvement* – updating the current policy to a better policy – is the primary goal of RL, much of the research in this dissertation focuses on

the sub-problem of *policy value estimation*. Policy value estimation is the RL sub-problem with the goal of determining the expected cumulative reward a certain policy will obtain for an agent. Effective policy value estimation is a necessary step before policy improvement for many RL algorithms. Thus, our contributions for utilizing off-policy and simulated data for policy value estimation is expected to also pay dividends for more data efficient policy improvement. Data efficient policy value estimation is also critical for real world problems in which human decision makers (e.g., a manager in industry or a policy maker in government) may require that the expected value of using a policy be known before allowing the policy to be deployed.

1.1 Importance Sampling

Importance sampling is a technique for re-weighting the observed rewards from off-policy data to reflect the relative likelihood of observing them under the policy to be evaluated (the *evaluation* policy) instead of the policy used to generate the data (the *behavior* policy). Figure 1.1 presents an example of how importance sampling re-weights rewards to estimate the value of an evaluation policy with experience from a different behavior policy.

While importance sampling is often the method of choice when on-policy data is unavailable, it has two main limitations. First, importance sampling may be inaccurate due to high variance when the behavior policy is not carefully selected. Second, the accuracy of importance sampling depends on actions being observed at their expected frequency under the behavior policy – something that likely only happens as the amount of data becomes infinite.

We address the first limitation by adapting the behavior policy towards a behavior policy that generates data that leads to more accurate policy value estimation. One problem in policy value estimation is that rare events sometimes greatly impact estimates of expected value. When the evaluation policy would rarely

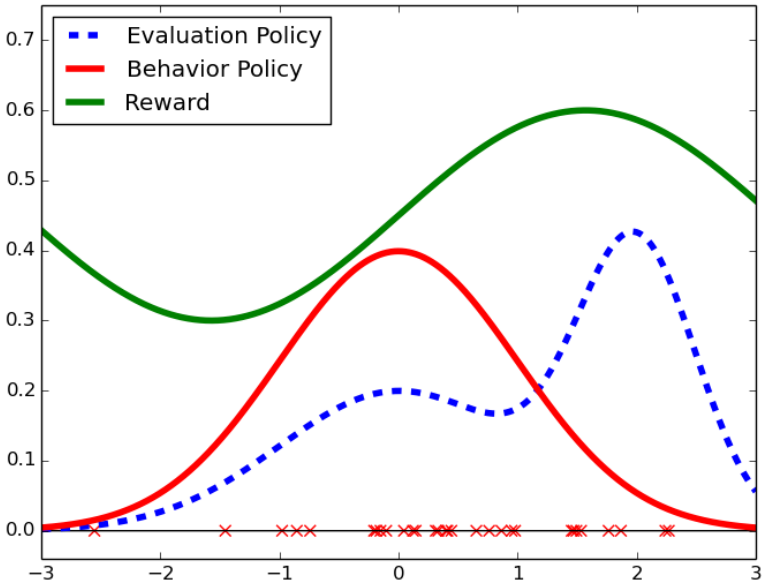


Figure 1.1: An example of importance sampling for policy evaluation in a single decision task. The agent's action is to choose a real-valued number between -3 and 3 and receive a reward according to the thick, green curve. We wish to evaluate an evaluation policy that selects actions with probability density given by the dashed, blue curve. However, we only have action samples (red Xs) collected from a behavior policy with action selection probability density given by the solid, red curve. Importance sampling re-weights the samples so that their weighting in a sample average approximates the weighting they would receive if sampled from the evaluation policy distribution. Intuitively, samples in areas where the blue curve is higher than the red curve are up-weighted and samples from other areas are down-weighted.

experience these events, even on-policy value estimation may have high variance unless the amount of available data is large. Instead, a lower variance estimate can be obtained by using a behavior policy that experiences such events more often but then down-weights the associated rewards with importance sampling. In Chapter 3, we introduce the *behavior policy search* (BPS) problem: searching for a behavior policy that leads to lower variance off-policy policy value estimation than on-policy policy

value estimation. In addition to introducing the BPS problem, this dissertation develops behavior policy search algorithms that adapt the behavior policy to collect off-policy data that lowers the variance of importance sampling estimates of policy performance.

This dissertation also address the second limitation of importance sampling: that the off-policy data should be observed at its expected frequency under the behavior policy. When the distribution of observed data (termed the *empirical policy*) fails to match the expected distribution of data (the behavior policy), importance sampling suffers from what we call sampling error. In Chapter 5, we introduce a family of estimators that correct sampling error by first estimating the empirical policy and then using it in place of the true behavior policy. This contribution also makes importance sampling applicable to settings when the behavior policy is unknown, as our estimators do *not* require knowledge of the behavior policy.

In addition to considering the problem of policy value estimation, we also consider how these new algorithms can be applied to policy improvement. Specifically, we consider the class of *batch policy gradient* policy improvement algorithms. We show that improving the behavior policy before data collection or estimating the empirical policy after data collection can lead to faster policy improvement when using this class of algorithms (Chapters 4 and 6 respectively).

1.2 Leveraging Simulation

Humans are able to leverage mental models of the physical world to learn new control tasks without extensive experience with the task. While these mental models are imperfect, they allow learning and planning to occur without any real world interaction. Having a model of the world is not unique to humans and animals; in many real world problems, simulated environments provide a form of prior knowledge that RL algorithms can exploit to improve data-efficiency. The second part of this

dissertation considers how real world data can be used to modify simulators such that skills learned in simulation are more likely to be effective in the real world.

In theory, transferring learning from simulation can make current RL algorithms immediately applicable to tasks such as robot learning in the physical world. Unfortunately, even small discrepancies between the physics of the real world and the physics of the simulator can often cause learning in simulation to find policies that fail in the real world. As an example, consider a robot learning to walk in a simulator where frictional forces are under-modeled. The robot may learn it can move its leg joints very quickly to achieve a fast walk. When the same controls are applied in the real, physical world, the walk may be jerky and the robot may fall over.

In order to leverage simulation, we use a small amount of real world data to modify the simulator such that the agent's actions affect the simulated world state in a way similar to how they would affect the world state in reality. Key to this approach is that the simulator does not need to be more realistic globally – it just needs to model the world dynamics well for actions the current policy would take in states the current policy would visit. This approach is an instance of grounded simulation learning (GSL) (Farchy et al., 2013) in which real world data is used to make simulation more realistic, policy improvement takes place within simulation, and then the improved policy is used to collect more data for further modification. Further modification is necessary since, as the policy changes, the agent is likely to visit new states where the real world is modeled poorly.

In Chapter 7 of this dissertation, we introduce a GSL algorithm that leverages real world data so that skills learned in simulation have an increased chance of transferring to the real world. We then evaluate the algorithm on learning tasks for the Softbank NAO robot. Our experiments show this algorithm allows reinforcement learning to take place entirely in simulation and also allows improving upon a

state-of-the-art walking controller for the NAO.

1.3 Simulation and Importance Sampling

So far we have discussed two complementary approaches to the problem of leveraging off-policy data to enhance the data-efficiency of reinforcement learning: improving imperfect environment simulators (Section 1.2) and importance sampling for direct off-policy data use (Section 1.1). We now consider how the approaches can be combined towards evaluation of policies and reinforcement learning algorithms. Recent work has demonstrated that model-based policy evaluation (i.e., evaluation within a simulated environment) can be combined with direct importance sampling methods to produce more accurate off-policy policy evaluation *without introducing statistical bias* (Jiang and Li, 2016; Thomas and Brunskill, 2016a). In Chapter 8 of this dissertation, we leverage this recent work to determine high confidence bounds for off-policy policy value estimation.

We apply simulation as a control variate for importance sampling methods to the problem of high confidence off-policy policy value estimation. The high confidence off-policy policy value estimation problem is to find a lower bound on the expected performance of an evaluation policy using off-policy data. The problem is more challenging than just constructing an accurate off-policy policy value estimator because a lower bound must take into account the variance of the estimation technique. Existing methods for this problem that only use importance sampling may provide loose lower bounds because importance sampling may have high variance no matter how different the behavior policy is from the evaluation policy. By combining the hybrid importance sampling and simulation methods introduced by Thomas and Brunskill (2016a), we obtain tighter lower bounds on the expected performance of an untested policy.

1.4 Contributions

In summary, this dissertation makes the following contributions to the reinforcement learning literature:

1. Formulation of the behavior policy search problem for collecting off-policy data for low variance importance sampling and an algorithm for this problem (Chapter 3).

2. A study of behavior policy search applied to batch policy gradient reinforcement learning (Chapter 4).

3. A family of off-policy policy value estimators that correct for the differences between the observed and true data distributions of off-policy data (Chapter 5).

4. Use of regression importance sampling (Contribution 3) to enhance the data efficiency of batch policy gradient reinforcement learning (Chapter 6).

5. A novel GSL algorithm called GAT for grounded action transformation that allows an RL agent to learn with simulated data (Chapter 7).

6. A simulation-based method for lower bounding the performance of untested policies with off-policy data (Chapter 8).

7. Hybrid importance sampling and simulation methods for lower bounding the performance of untested policies with off-policy data (Chapter 8).

Taken together, these contributions advance the capabilities of reinforcement learning algorithms, open up many new promising directions for research pertaining to off-policy learning and evaluation, and improve the usefulness of reinforcement learning in the physical world.

1.5 Dissertation Overview

This dissertation is laid out as follows. Though the dissertation is written as if it will be read from beginning to end, it is not strictly necessary to do so. Figure 1.2 specifies how the chapters and appendices of this dissertation depend on one another.

1. In Chapter 2, we give necessary background for this dissertation. We begin with introducing the specific reinforcement learning setting that we study. We then define the policy value estimation sub-problem and define foundational terminology and methods for this problem. Finally, we introduce the policy improvement sub-problem and introduce the class of batch policy gradient algorithms.

2. In Chapter 3, we formulate the *behavior policy search problem* and introduce the *behavior policy gradient algorithm* to address this problem. This problem and solution algorithm are Contribution 1 of this dissertation.

3. In Chapter 4, we apply behavior policy search to improve batch policy gradient reinforcement learning. The study of behavior policy search and policy improvement is Contribution 2 of this dissertation.

4. In Chapter 5, we introduce the family of *regression importance sampling estimators* that perform importance sampling using an estimated behavior policy. This family of estimators allows lower variance weighting of off-policy data for policy value estimation compared to using the true behavior policy and is Contribution 3 of this dissertation.

5. In Chapter 6, we introduce the *sampling error corrected policy gradient estimator* that provides lower variance weighting of data for batch policy gradient learning compared to using a common sample average approach. This estimator is Contribution 4 of this dissertation.

6. In Chapter 7, we introduce the *grounded action transformation algorithm* that allows a reinforcement learning agent to learn from simulated data. This algorithm is Contribution 5 of this dissertation.

7. In Chapter 8, we address the high confidence off-policy policy value estimation problem and introduce the final two contributions of the dissertation. First, we introduce the *model-based bootstrap algorithm* (Contribution 6) that estimates confidence intervals using multiple learned models. Second, we introduce the *weighted doubly robust bootstrap algorithm* (Contribution 7) that estimates confidence intervals using the weighted doubly robust estimator (to be introduced in Chapter 2).

8. In Chapter 9, we survey existing literature that pertains to the contributions of this dissertation.

9. Finally, in Chapter 10, we summarize the presented contributions and outline directions for future work.

10. In Appendix A we summarize notation used throughout this dissertation.

11. In Appendix B we define acronyms used throughout this dissertation.

12. In Appendix C we provide full derivations of theoretical results appearing in Chapter 3.

13. In Appendix D we provide full derivations of theoretical results appearing in Chapter 5 and 6.

14. In Appendix E we provide full derivations of theoretical results appearing in Chapter 8.

15. In Appendix F we provide experimental details to complement those included in Chapters 3, 4, 5, and 8. Full experimental details for Chapters 6 and 7 are

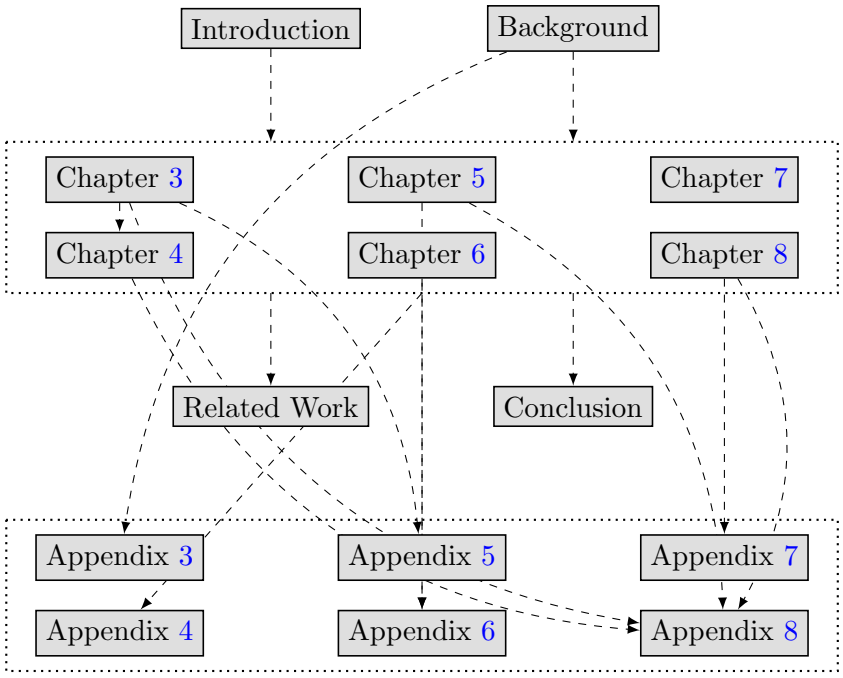


Figure 1.2: Chapter dependencies in this dissertation. Arrows denote that one chapter should be read before another. All chapters in a dotted rectangle can be read independently of one another unless marked otherwise.

included in those chapters.

Chapter 2

Background

Before presenting the contributions of this dissertation, we first provide necessary background. In this chapter we formalize the type of reinforcement learning problems that we study and introduce the reinforcement learning sub-problems of policy value estimation and policy improvement. We also introduce classes of existing algorithms that are used throughout this dissertation.

Throughout this dissertation we will follow the convention that sets are denoted with calligraphic capital letters (e.g., $\mathcal{S}$) and random variables are denoted with capital letters (e.g., S_t is the random variable representing the state observed at time t). Instantiations of random variables (e.g., $S_t = s$) and elements of sets ($s \in \mathcal{S}$) are denoted with lower case letters. Functions and scalar constants are also denoted with lower case letters. Vectors are denoted with bold lower case letters (e.g., $\boldsymbol{\theta}$). We will make and note exceptions when necessary to match conventional reinforcement learning notation.

2.1 Reinforcement Learning Notation

We consider the problem of an autonomous agent attempting to complete tasks in an unknown environment. This problem setting has been formulated in many different ways throughout the literature. In this dissertation, we formalize the studied setting as a fully observable, finite-horizon, episodic *Markov decision process* (MDP), where the agent fully knows its current state and interacts with the environment for a fixed number of time-steps before returning to an initial state and starting again (Puterman, 2014). Though some algorithms and results may transfer to the partially observable, infinite-horizon, and non-episodic settings, we limit the scope of our work in this dissertation to the fully observable, finite-horizon, and episodic setting that we formalize below.

An MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, P, r, L, \gamma, d_0)$ where:

- $\mathcal{S}$ is a set of possible world states.
- $\mathcal{A}$ is a set of actions available to the agent.
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function giving the probability of transitioning to a state s' after choosing action a in state s. P is also known as the dynamics of the environment. We use a capital P as is standard in the MDP and RL literature.
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a scalar reward function.
- L is the maximum length of one episode of interacting with the environment. Note that we use a capital L even though L is a constant.
- $\gamma \in [0, 1]$ is a discount factor that allows us to express a preference for immediate rewards compared to future rewards. Unless otherwise noted, we use $\gamma = 1$.
- d_0 is an initial state distribution.

- s_∞ is the terminal state.

The agent's behavior is determined by its *policy*. A policy is a probability mass function over actions, conditioned on the current state: $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. A policy is deterministic if $\pi(a|s) = 1$ for only one a in each s.[1] Otherwise it is a stochastic policy.

The agent interacts with the environment MDP as follows: The agent begins in initial state $S_0 \sim d_0$. At discrete time-step t the agents takes action $A_t \sim \pi(\cdot|S_t)$. The environment responds with $R_t \coloneqq r(S_t, A_t)$ and $S_{t+1} \sim P(\cdot|S_t, A_t)$ according to the reward function and state transition dynamics. After interacting with the environment for L steps the agent returns to a new initial state and the process repeats. If the agent enters the terminal state, s_∞, it remains there and receives zero reward until step L is reached. The agent does *not* know P, r, or d_0.

A trajectory, h, of length L is a state-action-reward history, $s_0, a_0, r_0, \ldots, s_{L-1}, a_{L-1}, r_{L-1}$. We define the return of a trajectory to be $g(h) = \sum_{t=0}^{L-1} \gamma^t r_t$.

Any policy, π, and transition dynamics, P, induce a distribution over trajectories. We write $\Pr(H = h|\pi, \mathcal{M})$ to denote the probability of observing trajectory h when following π in $\mathcal{M}$. When it is clear from the context what the MDP is, we will write $\Pr(H = h|\pi)$ and write $H \sim \pi$ to denote a trajectory sampled by executing π. The expected discounted return of policy π in MDP $\mathcal{M}$ is defined as $v(\pi) \coloneqq v(\pi, \mathcal{M}) \coloneqq \mathbf{E}[g(H)|H \sim \pi]$.

In addition to discussing entire trajectories and their return, we will also sometimes refer to trajectory segments and their returns. Given that h is a trajectory, we will use $h_{t:t'}$ to denote the partial trajectory, $s_t, a_t, r_t, ..., s_{t'}, a_{t'}, r_{t'}$. If $t < 0$, $h_{t:t'}$ denotes the beginning of the trajectory until step t'.

Let the action-value function, $q^\pi : \mathcal{S} \times \mathcal{A} \times \{0, ..., L-1\} \to \mathbb{R}$, be the expected

[1] We define notation for discrete MDPs, however, unless otherwise noted, all results and discussion hold for continuous $\mathcal{S}$ and $\mathcal{A}$ by replacing summations with integrals and probability mass functions with probability density functions.

return of following π after taking a particular action in a particular state and time-step. Formally, $q^\pi(s, a, t) = \mathbf{E}[\sum_{t'=t}^{L-1} R_{t'} | S_t = s, A_t = a, H_{t+1:L-1} \sim \pi]$ and let the state-value function, $v^\pi : \mathcal{S} \times \{0, ..., L-1\} \to \mathbb{R}$, be the expected value of q^π:

$$v^\pi(s, t) = \mathbf{E}[q^\pi(s, A, t) | A \sim \pi].$$

If the time-step when a state-action pair was encountered is ambiguous, we will write $q^\pi(s, a, \cdot)$ to denote the expected sum of discounted rewards remaining in the episode. Similarly, we will write $v^\pi(s, \cdot)$ to denote the same for state-values. From these definitions, it follows that $v(\pi) = \mathbf{E}[v^\pi(S_0, 0) | S_0 \sim d_0]$.

This section formalized the reinforcement learning setting of a learning agent taking actions in an unknown, episodic environment. The next two sections describe two sub-problems that the reinforcement learning community studies: policy value estimation and policy improvement. The first sub-problem is explained in detail in Section 2.2. The second sub-problem is introduced more briefly in Section 2.3.

2.2 Policy Value Estimation

Given a particular policy, π, and MDP, $\mathcal{M}$, we may want to know how much reward the agent can expect to receive if it follows π in $\mathcal{M}$. In the reinforcement learning setting, this question is asking, "what is $v(\pi, \mathcal{M})$?"

Specifically, we are given an *evaluation policy*, π_e, for which we would like to estimate $v(\pi_e, \mathcal{M})$ for some MDP $\mathcal{M}$; for the rest of this section we will suppress the dependency of v on $\mathcal{M}$. We assume a batch setting where we are given a dataset of trajectories, $\mathcal{D}$, or are able to run a policy in the environment to collect such a dataset. A policy value estimator, PE, uses trajectories in $\mathcal{D}$ to estimate $v(\pi_e)$. If π_e was used to collect the trajectories in $\mathcal{D}$ then PE is an *on-policy* policy value

estimator. Otherwise, it is an *off-policy* policy value estimator.[2]

We will assume for all observed trajectories that we know the *behavior* policy that was used to sample the trajectory and that this policy is the same for all trajectories. We formalize this assumption by defining $\mathcal{D} \coloneqq \{(H_i, \pi_b)\}_{i=1}^m$ where m is the number of trajectories in $\mathcal{D}$. We will use S_t^i, A_t^i, and R_t^i to denote the random variables representing the state, action, and reward at time-step t in the i^{th} trajectory.

2.2.1 Objectives for Policy Value Estimation

We now describe two possible objectives for policy value estimation: minimal mean squared error and high confidence policy value estimation.

2.2.1.1 Minimal Mean Squared Error

The first objective is minimal *mean squared error* (MSE). Before introducing this objective, we define $p_{\mathcal{D}}$ to be the probability distribution over all possible realizations of the data $\mathcal{D}$. This distribution will be determined by the MDP $\mathcal{M}$ and the behavior policy, π_b, however, we suppress this dependence below:

$$p_{\mathcal{D}}(\mathcal{D} = \{(h_i, \pi_b)\}) \coloneqq \prod_{i=1}^{m} \Pr(H_i = h_i | \pi_b, \mathcal{M}).$$

We can now define the mean squared error (MSE) of an estimator; minimizing this quantity is the first policy value estimation objective.

Definition 2.1.

$$\text{MSE}\Big[\text{PE}\Big] \coloneqq \mathbf{E}\Bigg[\Big(\texttt{PE}(\mathcal{D}) - v(\pi_e)\Big)^2 \;\Bigg|\; \mathcal{D} \sim p_{\mathcal{D}}\Bigg].$$

[2]The problem of policy value estimation has also been called *batch* policy evaluation (Liu et al., 2018) or just policy evaluation (Thomas and Brunskill, 2016a). We use *policy value estimation* in this dissertation to avoid confusion with the problem of learning the *value function* that is widely studied in the reinforcement learning literature.

This objective is the objective most commonly considered when the goal is to produce the most accurate estimator (e.g., Thomas and Brunskill (2016a); Precup et al. (2000)).

2.2.1.2 High Confidence Policy Value Estimation

An alternative to minimizing MSE, is to lower bound the value of $v(\pi_e)$ so that the true (unknown) value of $v(\pi_e)$ is above the lower bound with a given probability. We call this problem the *high-confidence policy value estimation* problem. Specifically, given a confidence parameter, $\delta \in [0, 1]$, high confidence policy value estimation methods determine a lower bound, $v_\delta(\pi_e)$, on $v(\pi_e)$ such that $v_\delta(\pi_e) \leq v(\pi_e)$ with probability at least $1 - \delta$. That is, for k different realizations of the observed data, $\mathcal{D}$, the expected number of times that $v_\delta(\pi_e)$ is greater than $v(\pi_e)$ is no more than δk. This objective is desirable in situations where safety is important – we want to estimate the value of the policy accurately with a bounded risk of over-estimating.

2.2.2 Variance, Bias, and Consistency

When discussing estimators for policy value estimation, we will primarily discuss three statistical properties: variance, bias, and consistency. The *variance* of an estimator describes how much its estimates differ from the expected value of its estimates.

Definition 2.2.

$$\text{Var}\Big[\text{PE}\Big] \coloneqq \mathbf{E}\Bigg[\Big(\text{PE}(\mathcal{D}) - \mathbf{E}[\text{PE}(\mathcal{D})]\Big)^2 \;\Bigg|\; \mathcal{D} \sim p_{\mathcal{D}}\Bigg]$$

The *bias* of an estimator is the difference between the expected value of the estimator and the true (unknown) value $v(\pi_e)$.

Definition 2.3.

$$\text{Bias}\Big[\text{PE}\Big] \coloneqq \mathbf{E}\Big[\text{PE}(\mathcal{D}) \;\Big|\; \mathcal{D} \sim p_{\mathcal{D}}\Big] - v(\pi_e).$$

Definition 2.4. *An estimator is an* ***unbiased*** *estimator of* $v(\pi_e)$ *if* $\text{Bias}[\text{PE}] = 0$.

Bias and variance have the following relationship with MSE:

$$\text{MSE }[\text{PE }] = \text{Var}[\text{PE}] + \text{Bias}[\text{PE}]^2.$$

Finally, consistency is concerned with the asymptotic error of an estimator. A consistent estimator has zero MSE with probability 1 as the number of trajectories goes to infinity.

Definition 2.5. *Let* $\mathcal{D}_m$ *be the random variable representing the trajectory set with* m *trajectories. An estimator is a* ***consistent*** *estimator of* $v(\pi_e)$ *if,*

$$\Pr\Big(\lim_{m\to\infty} \text{PE}(\mathcal{D}_m) = v(\pi_e)\Big) = 1.$$

Note that estimators can be biased and consistent or unbiased and inconsistent. The former case arises when the estimator's bias decreases asymptotically (with respect to the size of $\mathcal{D}$) to zero. The latter case arises when the estimator's variance does *not* decrease asymptotically to zero.[3]

2.2.3 Three Classes of Policy Value Estimators

In this subsection, we introduce three common classes of policy value estimators. This dissertation makes contributions to the understanding and practice of each class of estimators.

[3]One common example of an unbiased but inconsistent estimator is the following: let $X_1, X_2, ..., X_n$ be n samples from a normal distribution. The estimator that always returns X_1 is an unbiased estimate of the mean of the distribution but it is *not* a consistent estimator because X_1 always has positive variance.

2.2.3.1 Importance Sampling Estimators

One class of policy value estimators directly uses averages of the observed returns to estimate $v(\pi_e)$. The most straightforward of such methods is the *on-policy Monte Carlo* (MC) estimator. Given a data set, $\mathcal{D}$, of m trajectories *sampled from* π_e, the Monte Carlo estimate of $v(\pi_e)$ is the average return:

$$\mathrm{MC}(\mathcal{D}) \coloneqq \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{L-1}\gamma^t R_t^i = \frac{1}{m}\sum_{i=1}^{m} g(H_i).$$

This estimator is unbiased and consistent given mild assumptions.[4] However, this method can have high variance.

The Monte Carlo estimator can be generalized to the *off-policy* setting by re-weighting returns from any *behavior policy*, π_b, such that they are unbiased estimates of the expected return of the *evaluation* policy. The off-policy Monte Carlo estimator is known in the RL literature as the *Importance Sampling* (IS) estimator. The re-weighted IS return of a trajectory, H, sampled from behavior policy π_b is:

$$\mathrm{IS}(\pi_e, H, \pi_b) \coloneqq g(H)\prod_{t=0}^{L-1}\frac{\pi_e(A_t|S_t)}{\pi_b(A_t|S_t)}.$$

Intuitively, the IS return up-weights returns that were more likely under π_e than π_b and down-weights returns that were less likely under π_e compared to π_b. The IS estimator is then:

$$\mathrm{IS}(\pi_e, \mathcal{D}) \coloneqq \frac{1}{m}\sum_{i=1}^{m}\mathrm{IS}(\pi_e, H_i, \pi_b).$$

Note that when π_b and π_e are the same the IS estimator is identical to the Monte Carlo estimator.

In RL, importance sampling allows off-policy data to be used as if it were

[4] If $v(\pi_e)$ exists, the Monte Carlo estimator is consistent by the Khintchine Strong law of large numbers (Sen and Singer, 1993).

on-policy. Importance sampling is both unbiased and consistent, however, like the Monte Carlo estimator, it may suffer from high variance. The variance of IS may in fact be worse than that of on-policy Monte Carlo because the importance weights themselves may have high variance.

Many methods have been proposed to lower the variance of IS. We will discuss two of these throughout this dissertation: weighted importance sampling and per-decision importance sampling.

Weighted importance sampling normalizes the importance weights so that they are bounded in $[0, 1]$. Define the importance weight up to and including time step t for trajectory i as:

$$\rho_t^i = \prod_{j=0}^{t} \frac{\pi_e(A_j^i|S_j^i)}{\pi_b(A_j^i|S_j^i)}.$$

The weighted importance sampling estimate normalizes the importance weights by their total sum:

$$\text{WIS}(\pi_e, \mathcal{D}) = \sum_{i=1}^{m} \frac{\rho_{L-1}^i}{\sum_{j=1}^{m} \rho_{L-1}^j} g(H_i).$$

Unlike the possibly unbounded variance of IS weights, the variance of WIS weights is bounded since each weight must be between 0 and 1. The normalization introduces bias into the estimate, however the bias decreases asymptotically to zero and thus weighed importance sampling provides consistent estimates (Precup et al., 2000).

Per-decision importance sampling (PDIS) (Precup et al., 2000) makes use of the fact that rewards are independent of future actions by importance sampling the individual rewards instead of the full return:

$$\text{PDIS}(\pi_e, \mathcal{D}) = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{L-1} \rho_t^i \gamma^t R_t^i.$$

PDIS tends to have lower variance than the basic IS estimator yet remains free of bias. Like IS and WIS, it is consistent.

Weighted and per-decision importance sampling can also be combined to produce the *per-decision weighted importance sampling estimator* (PDWIS):

$$\text{PDWIS}(\pi_e, \mathcal{D}) = \sum_{i=1}^{m} \sum_{t=0}^{L-1} \frac{\rho_t^i}{\sum_{j=1}^{m} \rho_t^j} \gamma^t R_t^i.$$

PDWIS is biased but consistent (Thomas, 2015).

2.2.3.2 Model-based Policy Value Estimation

An alternative to importance sampling is model-based policy value estimation. The model-based (MB) policy value estimator estimates $v(\pi_e)$ by first using all observed trajectories to estimate the transition probabilities and reward function of the underlying MDP. Let $\mathcal{M}$ be the MDP under which we want to evaluate π_e. A model is defined as $\widehat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \widehat{P}, \hat{r}, \gamma, \hat{d}_0)$ where $\widehat{P}$, $\hat{d}_0$, and $\hat{r}$ are estimated from the (s, a, r, s') tuples that occur in trajectories in $\mathcal{D}$. Then the MB estimator returns $v(\pi_e)$ as the expected return of π_e when following π_e in $\widehat{\mathcal{M}}$.

If a model can capture the true MDP's dynamics or generalize well to unseen parts of the state-action space then model-based estimates can have much lower variance than importance sampling estimates. However, models reduce variance at the cost of adding bias to the estimate. Bias in model-based estimates of $v(\pi_e)$ may arise from two sources:[5]

1. When we lack data for a particular (s, a) pair, we must make assumptions about how to estimate $P(\cdot|s, a)$.

2. If we use function approximation, we must make assumptions about the model class to which P belongs.

[5]Model bias may also arise when the agent is acting in a *partially-observable* Markov decision process. However, since we restrict ourselves to MDPs in this dissertation, we will *not* discuss this form of bias in depth.

In Chapter 7 we will discuss learning in environment simulators. In this dissertation, we assume that having access to an environment simulator is formally equivalent to having access to a model $\widehat{\mathcal{M}}$ which approximates the true environment, $\mathcal{M}$, in its state transition probabilities.

2.2.3.3 Doubly Robust Value Estimation

Our final class of estimator uses possibly biased and inconsistent models to lower the high variance of importance sampling methods while remaining unbiased and consistent. Such methods are known as *doubly robust* (DR) estimators (Thomas and Brunskill, 2016a; Jiang and Li, 2016; Dudík et al., 2011). These methods combine importance sampling estimation with model-based estimation and are known as doubly robust because they can produce accurate estimates as long as *either* the importance sampling estimate or model-based estimate is accurate.

In the RL setting, the DR estimator replaces the re-weighted return with:

$$\text{DR}(\pi_e, H, \pi_b, \hat{q}^{\pi_e}, \hat{v}^{\pi_e}) = \hat{v}^{\pi_e}(S_0, 0) + \sum_{t=0}^{L-1} \rho_t^{(H)} \gamma^t (R_t - \hat{q}^{\pi_e}(S_t, A_t, t) + \gamma \hat{v}^{\pi_e}(S_{t+1}, t+1))$$

where $\hat{q}^{\pi_e} : \mathcal{S} \times \mathcal{A} \times \{0, ..., L-1\} \to \mathbb{R}$ is any estimate of q^{π_e}, $\hat{v}^{\pi_e} : \mathcal{S} \times \{0, ..., L-1\} \to \mathbb{R}$ is the expected value of $\hat{q}^{\pi_e}$: $\hat{v}^{\pi_e}(s, t) = \mathbf{E}[\hat{q}^{\pi_e}(s, A, t) | A \sim \pi_e]$ and $v(S_L, L) \coloneqq 0$. Intuitively, DR is replacing part of the randomness of a PDIS estimate with the known expected return under the approximate model. The batch DR estimator is then the mean of the DR return over all trajectories in $\mathcal{D}$:

$$\text{DR}(\pi_e, \mathcal{D}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e}) = \frac{1}{m} \sum_{i=1}^{m} \text{DR}(\pi_e, H_i, \pi_b, \hat{q}^{\pi_e}, \hat{v}^{\pi_e}).$$

To understand how DR can incorporate a model and remain unbiased we briefly describe control variates which are central to the derivation provided by Thomas and Brunskill (2016a). If we wish to estimate $\mathbf{E}[X]$ for a random variable

X we can obtain a lower variance estimator by estimating the random variable $Z = X - Y + \mathbf{E}[Y]$ instead; Y is a second random variable, with known expectation, termed a control variate. Since $\mathbf{E}[Z] = \mathbf{E}[X] - \mathbf{E}[Y] + \mathbf{E}[Y] = \mathbf{E}[X]$ this new estimator is unbiased. The variance of Z is $\mathrm{Var}[Z] = \mathrm{Var}[X] + \mathrm{Var}[Y] - 2\,\mathrm{Cov}[X, Y]$. Provided $2\,\mathrm{Cov}[X, Y] > \mathrm{Var}[Y]$ then the variance of Z is $\mathrm{Var}[X] + \mathrm{Var}[Y] - 2\,\mathrm{Cov}[X, Y] < \mathrm{Var}[X]$. DR is able to incorporate a model yet remain free of model bias because the model value function only serves as a control variate which changes the variance of the PDIS estimate.

The DR estimator is an off-policy estimator and can be used with data generated by any policy. When the method is used on-policy, we will refer to the DR estimator as the *advantage-sum* estimator (ASE) as it has appeared previously in the literature under this name (Zinkevich et al., 2006; White and Bowling, 2009; Veness et al., 2011):

$$\mathrm{ASE}(\pi_e, \mathcal{D}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e}) = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{v}^{\pi_e}(S_0^i, 0) + \sum_{t=0}^{L-1} \gamma^t \delta_t^i \right)$$

where $\delta_t^i \coloneqq R_t^i - \hat{q}^{\pi_e}(S_t^i, A_t^i, t) + \gamma \hat{v}^{\pi_e}(S_{t+1}^i, t+1)$. To the best of our knowledge, ASE was developed independently from the DR estimator and takes the name *advantage-sum* due to its connection to the *advantage function* in reinforcement learning, i.e., the difference $\hat{q}^{\pi_e}(s, a, t) - \hat{v}^{\pi_e}(s, t)$.

Finally, we can further reduce the variance of the DR estimator by replacing the weights $\frac{\rho_t^i}{m}$ with $\frac{\rho_t^i}{\sum_{j=1}^{m} \rho_t^j}$ as done by weighted importance sampling. This estimator is called the *weighted doubly robust* (WDR) estimator (Thomas and Brunskill, 2016a)

and is defined as:

$$\text{WDR}(\pi_e, \mathcal{D}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e}) = \sum_{i=1}^{m} \sum_{t=0}^{L-1} \frac{\rho_t^i}{\sum_{j=1}^{m} \rho_t^j} \gamma^t \left(R_t^i - \hat{q}^{\pi_e}(S_t^i, A_t^i, t) + \gamma \hat{v}^{\pi_e}(S_{t+1}^i, t+1) \right) + \frac{1}{m} \sum_{i=1}^{m} \hat{v}^{\pi_e}(S_0^i, 0).$$

WDR has lower variance than DR at the cost of bias from the normalized importance weights. Like DR, WDR is consistent (Thomas and Brunskill, 2016a).

2.3 Policy Improvement

The primary goal of reinforcement learning research is designing algorithms that address the problem of *policy improvement*. In this dissertation, we define policy improvement as the problem of finding the policy, π, that maximizes $v(\pi, \mathcal{M})$ for a target environment $\mathcal{M}$.

In this dissertation, we assume that π is parameterized by a vector $\boldsymbol{\theta}$ and denote the parameterized policy as $\pi_{\boldsymbol{\theta}}$. Given this representation and policy parameters, $\boldsymbol{\theta}$, the goal of a step of policy improvement is to find $\boldsymbol{\theta}'$, such that $v(\pi_{\boldsymbol{\theta}'}) > v(\pi_{\boldsymbol{\theta}})$. Policy improvement algorithms typically rely on evaluating the effects of taking different actions and then changing $\boldsymbol{\theta}$ so that $\pi_{\boldsymbol{\theta}}$ puts more probability mass on actions that lead to higher returns.

Many of the same techniques used for policy value estimation can be adapted for use within policy improvement algorithms. Thus, efficient and effective policy value estimation can lead to more efficient and effective policy improvement. For this reason, even though policy improvement is the ultimate goal, we focus on improving policy value estimation throughout much of this dissertation.

As in policy value estimation, policy updates can be made on-policy or off-policy depending on how data is collected to compute the update.

Batch Policy Gradient Reinforcement Learning

One common class of reinforcement learning algorithms is the class of *batch policy gradient* methods. Contributions 2 and 4 apply to batch policy gradient learning and thus we briefly introduce this class of methods here.

Policy gradient methods learn a (locally) optimal policy by updating the policy parameters along the gradient of v with respect to $\boldsymbol{\theta}$. This gradient can be analytically derived as:

$$\frac{\partial}{\partial \boldsymbol{\theta}} v(\pi_{\boldsymbol{\theta}}) = \mathbf{E}\left[\sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \sum_{i=t}^{L-1} \gamma^i R_i \middle| H \sim \pi_{\boldsymbol{\theta}}\right]. \tag{2.1}$$

When the discount factor, γ, equals 1, we can also make use of the following expression which is proportional to the true gradient:

$$\frac{\partial}{\partial \boldsymbol{\theta}} v(\pi_{\boldsymbol{\theta}}) \propto \mathbf{E}\left[q^{\pi_{\boldsymbol{\theta}}}(S, A, \cdot) \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S) \middle| S \sim d_{\pi_{\boldsymbol{\theta}}}, A \sim \pi_{\boldsymbol{\theta}}\right] \tag{2.2}$$

where $d_{\pi_{\boldsymbol{\theta}}}(s)$ is the distribution of states observed when running $\pi_{\boldsymbol{\theta}}$ in $\mathcal{M}$ and $q^{\pi_{\boldsymbol{\theta}}}(S, A, \cdot)$ is the expected sum of rewards until the end of the episode. Policy gradient methods multiply gradient estimates by constant step-size parameter and thus only the right gradient direction is needed to improve $\pi_{\boldsymbol{\theta}}$. Thus either (2.1) or (2.2) can be used in batch policy gradient learning.

Since the expectations in 2.1 and 2.2 depend on the unknown environment and return probabilities (via P, $d_{\pi_{\boldsymbol{\theta}}}$, or $q^{\pi_{\boldsymbol{\theta}}}$), the gradient is typically approximated with sampling. We detail how this estimation is done for (2.2).

Let $\mathcal{T} = \{(S_j, A_j)\}_{j=1}^m$ be a set of m state-action pairs observed while following $\pi_{\boldsymbol{\theta}}$ in the environment. The *Monte Carlo* batch policy gradient estimator is defined as:

$$\frac{\partial}{\partial \boldsymbol{\theta}} v(\pi_{\boldsymbol{\theta}}) \approx g_{\texttt{mc}}(\mathcal{T}) = \frac{1}{m} \sum_{j=1}^{m} \hat{q}^{\pi_{\boldsymbol{\theta}}}(S_j, A_j, \cdot) \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_j|S_j) \tag{2.3}$$

where $\hat{q}^{\pi_{\boldsymbol{\theta}}}(s, a, \cdot)$ is an estimate of the discounted sum of rewards remaining in the episode, $q^{\pi_{\boldsymbol{\theta}}}(s, a, \cdot)$.[6] For sufficiently large m, the Monte Carlo estimator approximately weights each $\hat{q}^{\pi_{\boldsymbol{\theta}}}(s, a, \cdot) \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)$ by the probability $d_{\pi_{\boldsymbol{\theta}}}(s)\pi_{\boldsymbol{\theta}}(a|s)$ and $g_{\texttt{mc}}(\mathcal{T})$ closely approximates $\frac{\partial}{\partial \boldsymbol{\theta}} v(\pi_{\boldsymbol{\theta}})$. While this estimator is known to have high variance, the policy gradient and its Monte Carlo approximation form the basis for many other methods that give strong performance. In particular, batch policy gradient methods include reinforcement learning algorithms that can obtain high performance on complex tasks (e.g., Schulman et al. (2017); Wang et al. (2016); Gu et al. (2017b)). Algorithm 1 shows pseudocode for a generic batch policy gradient method.

Algorithm 1 Generic Batch Policy Gradient Method

Input: Initial policy parameters, $\boldsymbol{\theta}_0$, batch size m, a step-size for each iteration, α_i, and number of iterations n.

Output: Optimized policy parameters $\boldsymbol{\theta}_n$.

1: **for all** $i = 0$ to n **do**
2: $\quad \mathcal{T}_i =$ Sample m steps: $(S, A) \sim \pi_{\boldsymbol{\theta}_i}$
3: $\quad g_{\texttt{mc}} \leftarrow \frac{1}{m} \sum_{j=1}^{m} \hat{q}^{\pi_{\boldsymbol{\theta}}}(S_j, A_j, \cdot) \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_i}(A_j|S_j)$
4: $\quad \boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha_i \cdot g_{\texttt{mc}}$
5: **end for**
6: **Return** $\boldsymbol{\theta}_n$

Batch policy gradient algorithms usually share the general iterative steps:

1. Collect m state-action pairs from the environment by running the current policy $\pi_{\boldsymbol{\theta}i}$. We will call the set of these state-action pairs $\mathcal{T}_i$.

2. Use $\mathcal{T}_i$ to compute $\hat{q}^{\pi_{\boldsymbol{\theta}_i}}(S, A, \cdot)$ for all S, A that occur in $\mathcal{T}_i$.

3. Approximate $\frac{\partial}{\partial \boldsymbol{\theta}} v(\pi_{\boldsymbol{\theta}i})$ with (2.3) using $\mathcal{T}_i$ and the $\hat{q}^{\pi_{\boldsymbol{\theta}i}}$ values.

4. Set $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha_i g_{\texttt{mc}}(\mathcal{T}_i)$ where α_i is a step-size that may vary across iterations.

[6] A simple way to obtain $\hat{q}^{\pi}(s, a, \cdot)$ is to use the observed sum of discounted rewards following the occurrence of a in s. This method provides an unbiased estimate of $q^{\pi}(s, a, \cdot)$ and is the method we use throughout this dissertation.

The exact implementation of any of these steps can vary from method to method. For example, Williams (1992) uses $\hat{q}^{\pi_{\boldsymbol{\theta}}}(s,a,t) = \sum_{t'=t}^{L-1} \gamma^{t'} r_{t'}$ to estimate $q^{\pi_{\boldsymbol{\theta}}}$ while Sutton et al. (2000a) fit a linear function approximator, $\hat{q}_{\mathbf{w}}$, and use it as the estimate of $q^{\pi_{\boldsymbol{\theta}}}$. It is also common to use the *advantage function*, $\hat{a}^{\pi}(s,a,t) = \hat{q}^{\pi}(s,a,t) - \hat{v}^{\pi}(s,t)$, in place of $\hat{q}^{\pi}(s,a,t)$ where $\hat{v}^{\pi}(s,t) = \mathbf{E}[\hat{q}^{\pi}(s,A,t)|A \sim \pi]$. Replacing $\hat{q}^{\pi}$ with $\hat{a}^{\pi}$ leaves the gradient unchanged but may reduce variance as $\hat{v}^{\pi}$ serves as a control variate for $\hat{q}^{\pi}$ (Greensmith et al., 2004; Williams, 1992).

2.4 Summary

In this chapter, we have introduced the fully observable, finite-horizon, and episodic reinforcement learning setting and the common notation that we will use throughout this dissertation. We have also introduced two reinforcement learning sub-problems: policy value estimation and policy improvement. In the following chapters we will introduce the contributions of this dissertation and describe their significance to addressing these two sub-problems.

Chapter 3

Collecting Data for Off-policy Policy Value Estimation

In this chapter, we consider how a reinforcement learning agent should collect data for off-policy value estimation of a fixed evaluation policy. Specifically, we consider the off-policy value estimation technique of importance sampling and consider how to collect data (i.e., choose the behavior policy) for low variance importance sampling estimates of a policy's value. We introduce a methodology for learning a behavior policy that collects data for such low variance estimates.[7]

Importance sampling (introduced in Section 2.2.3.1) re-weights returns observed while executing the behavior policy, π_b, such that they are unbiased estimates of the performance of the evaluation policy, π_e. Presently, importance sampling is usually used when off-policy data is already available or when executing the evaluation policy is impractical. In these situations – where π_b is dictated by circumstance – importance sampling estimates often have high variance (Thomas et al., 2015a). For this reason, an implicit assumption in the RL community has generally been that

[7]This chapter contains work that was done in collaboration with Philip Thomas and Scott Niekum and previously published at ICML 2017 (Hanna et al., 2017b).

on-policy policy value estimation is more accurate when it is feasible. Contribution 1 of this thesis is to show how appropriate selection of the behavior policy can lead to lower variance importance-sampling-based policy value estimates than on-policy estimates.

In Section 9.1 we will discuss related literature to choosing the behavior policy for importance sampling in reinforcement learning. Here, we note that the algorithm we introduce in this chapter is an *adaptive importance sampling* algorithm (Rubinstein, 1997; Arouna, 2004). Prior approaches to adaptive importance sampling in reinforcement learning have considered adapting the MDP transition dynamics while we consider adapting the behavior policy (Ciosek and Whiteson, 2017; Desai and Glynn, 2001; Frank et al., 2008).

3.1 Incremental Policy Value Estimation

This section poses the policy value estimation problem in an incremental, episodic setting. We are given an *evaluation policy*, π_e, for which we would like to estimate $v(\pi_e)$. We assume π_e is parameterized by $\boldsymbol{\theta}_e$ and we have access to $\boldsymbol{\theta}_e$. At iteration i, we sample a single trajectory H_i with a policy $\pi_{\boldsymbol{\theta}_i}$ and add $\{H_i, \pi_{\boldsymbol{\theta}_i}\}$ to a set $\mathcal{D}$. We use $\mathcal{D}_i$ to denote the set at iteration i. A method that always (i.e., $\forall i$) chooses $\boldsymbol{\theta}_i = \boldsymbol{\theta}_e$ is on-policy; otherwise, the method is off-policy. A policy value estimation method, PE, uses $\mathcal{D}$ to estimate $v(\pi_e)$, i.e., $\text{PE}(\pi_e, \mathcal{D})$ is a scalar-valued estimate of $v(\pi_e)$. Our goal is to design a policy value estimation algorithm that produces estimates with low MSE at the i^{th} iteration:

$$\text{MSE}[\text{PE}] = \mathbf{E}\left[\left(\text{PE}(\pi_e, \mathcal{D}) - v(\pi_e)\right)^2 \middle| H_0 \sim \pi_{\boldsymbol{\theta}_0}, ..., H_i \sim \pi_{\boldsymbol{\theta}_i}\right].$$

We focus on selecting the behavior policy for unbiased estimators of $v(\pi_e)$ and leave behavior policy selection for biased estimators to future work. For unbiased

estimators, minimizing variance is equivalent to minimizing MSE.

We use importance sampling for unbiased estimates of $v(\pi_e)$. We first describe the theoretical potential for variance reduction with an appropriately selected behavior policy. In general this policy will be unknown. Thus, we introduce a policy value estimation sub-problem – the behavior policy search problem – solutions to which will adapt the behavior policy to provide lower mean squared error policy performance estimates.

3.2 The Optimal Variance Behavior Policy

We first observe that, in MDPs with *deterministic* P and d_0, an appropriately selected behavior policy can lower the variance of importance sampling to zero. This observation motivates the idea that off-policy policy value estimation can have lower variance than on-policy policy value estimation. While this observation has been made for importance sampling *outside of* RL (Rubinstein and Kroese, 2016), we show here that a zero-variance policy is possible for MDPs with deterministic P and d_0 and any evaluation policy, under the assumption that all returns are either all positive or all negative. These assumptions are only made to illustrate the potential for variance reduction with an appropriately selected behavior policy. In the following section we describe how an initial policy can be adapted towards a minimal variance behavior policy even when the MDP is stochastic and the returns have mixed signs.

Let $w_\pi(H) \coloneqq \prod_{t=0}^{L-1} \pi(A_t|S_t)$, i.e., the probability of taking the sequence of actions observed in trajectory H conditioned on the observed states. Consider a behavior policy $\pi_b{}^\star$ such that for any trajectory, H:

$$v(\pi_e) = \mathrm{IS}(\pi_e, H, \pi_b{}^\star) = g(H)\frac{w_{\pi_e}(H)}{w_{\pi_b{}^\star}(H)}.$$

Rearranging the terms of this expressions yields:

$$w_{\pi_b{}^\star}(H) = g(H)\frac{w_{\pi_e}(H)}{v(\pi_e)}.$$

Thus, if we can select $\pi_b{}^\star$ such that the probability of observing any $H \sim \pi_b{}^\star$ is $\frac{g(H)}{v(\pi_e)}$ times the likelihood of observing $H \sim \pi_e$ then the IS estimate has zero variance with only a single sampled trajectory; regardless of the value of $g(H)$, the importance weight under $\pi_b{}^\star$ will scale $g(H)$ exactly to $v(\pi_e)$ and the importance-sampled return will equal $v(\pi_e)$.

Unfortunately, such a zero variance behavior policy depends on the unknown value $v(\pi_e)$ as well as the unknown reward function r (via $g(H)$). Thus, while there exists an optimal variance behavior policy for IS – which is not π_e – in practice we cannot analytically determine $\pi_b{}^\star$. Additionally, $\pi_b{}^\star$ may be unrepresentable by any $\boldsymbol{\theta}$ in our policy class.

3.3 The Behavior Policy Search Problem

Since the behavior policy with zero variance cannot be analytically determined (even when it exists) we instead introduce the *behavior policy search* (BPS) problem for finding π_b that lowers the MSE of estimates of $v(\pi_e)$. A BPS problem is defined by the inputs:

1. An evaluation policy π_e with policy parameters $\boldsymbol{\theta}_e$.
2. An off-policy policy value estimation algorithm, $\text{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})$, that takes a trajectory, $H \sim \pi_{\boldsymbol{\theta}}$, or, alternatively, a set of trajectories, and returns an estimate of $v(\pi_e)$.

A BPS solution is a policy, $\pi_{\boldsymbol{\theta}_b}$, that generates trajectories, H, such that $\text{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}_b})$ has lower MSE than $\text{OPE}(\pi_e, H, \pi_e)$. Algorithms for this prob-

lem are BPS algorithms.

Recall that we consider an incremental policy value estimation setting where at each iteration we can select a behavior policy to collect a trajectory and add this trajectory to a dataset, $\mathcal{D}$. At each iteration, we use all trajectories in $\mathcal{D}$ to estimate $v(\pi_e)$. At the i^{th} iteration, a BPS algorithm selects a behavior policy that will be used to generate a trajectory, H_i. The policy value estimation algorithm, OPE, then estimates $v(\pi_e)$ using all trajectories in $\mathcal{D}$. Naturally, the selection of the behavior policy depends on how a policy value estimation algorithm estimates $v(\pi_e)$.

In a BPS problem, the i^{th} iteration proceeds as follows. First, given all of the past behavior policies, $\{\pi_{\boldsymbol{\theta}_j}\}_{j=0}^{i-1}$, and the resulting trajectories, $\{H_j\}_{j=0}^{i-1}$, the BPS algorithm must select $\boldsymbol{\theta}_i$. The policy $\pi_{\boldsymbol{\theta}_i}$ is run for one episode to generate the trajectory H_i. Then the BPS algorithm uses OPE to estimate $v(\pi_e)$ given the available data, $\mathcal{D} \coloneqq \{(H_j, \pi_{\boldsymbol{\theta}_j})\}_{j=0}^{i}$.

One natural question is: if we are given a limit on the number of trajectories that can be sampled, is it better to "spend" some of our limited trajectories on BPS instead of using on-policy estimates? The answer to this question depends on how we define "better." In terms of sample efficiency, adapting the behavior policy will provide a more accurate estimate. Since each $\text{OPE}(\pi_e, H_i, \pi_{\boldsymbol{\theta}_i})$ is an unbiased estimator of $v(\pi_e)$, we can use all sampled trajectories to compute $\text{OPE}(\pi_e, \mathcal{D})$. Provided for all iterations, the variance of OPE with $\pi_{\boldsymbol{\theta}i}$ is less than that of OPE with π_e (i.e., on-policy policy value estimation) then a BPS algorithm will be guaranteed to achieve lower MSE than an on-policy policy value estimation, showing that it is, in fact, worthwhile to do so. This claim is supported by our empirical study.

On the other hand, adapting the behavior policy increases the computational complexity of estimating $v(\pi_e)$. The exact increase will depend on the behavior policy search algorithm used, however, it seems unlikely that a behavior policy search algorithm will match the simplicity and computational efficiency of simply running

the evaluation policy. Since this dissertation focuses on data efficiency, we focus on sample efficiency. However, practitioners must decide which is more important for a particular application.

3.4 Behavior Policy Gradient Theorem

We now introduce the main algorithmic component of Contribution 1: an analytic expression for the gradient of the mean squared error of the importance sampling estimator and a stochastic gradient descent algorithm that adapts $\pi_{\boldsymbol{\theta}}$ to minimize the MSE between the importance sampling estimate and $v(\pi_e)$. Our algorithm – *behavior policy gradient* (BPG) – begins with on-policy estimates and adapts the behavior policy with gradient descent on the MSE with respect to $\boldsymbol{\theta}$. The gradient of the MSE with respect to the policy parameters is given by the following theorem:

Theorem 3.1. *Behavior Policy Gradient Theorem*

$$\frac{\partial}{\partial \boldsymbol{\theta}} \text{MSE}\Big[\text{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}})\Big] = \mathbf{E}\left[-\text{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \middle| H \sim \pi_{\boldsymbol{\theta}}\right]$$

Proof. See Appendix C.1 for full proof.

The proof of Theorem 3.1 relies on the fact that the MSE of an estimator is the sum of its variance and the square of its bias. Since importance sampling is unbiased, its MSE is equal to its variance. Thus, the gradient of the MSE given by Theorem 3.1 is also the gradient of the variance which can be estimated without knowledge of $v(\pi_e)$. Full details of this derivation are included in Appendix C.1.

BPG uses stochastic gradient descent in place of exact gradient descent: replacing the expectation in Theorem 3.1 with an unbiased estimate of the true gradient. In theory, the single trajectory H_i is sufficient for an unbiased estimate of this gradient. In our experiments, we sample a batch, $\mathcal{B}_i$, of k trajectories with

$\pi_{\boldsymbol{\theta}_i}$ to lower the variance of the gradient estimate at iteration i. In the BPS setting, sampling a batch of trajectories is equivalent to holding $\boldsymbol{\theta}$ fixed for k iterations and then updating $\boldsymbol{\theta}$ with the k most recent trajectories used to compute the gradient estimate.

Full details of BPG are given in Algorithm 2. At iteration i, BPG samples a batch, $\mathcal{B}_i$, of k trajectories with $\pi_{\boldsymbol{\theta}_i}$ and adds $\{(H_{i+j}, \pi_{\boldsymbol{\theta}_i})_{j=1}^{k}\}$ to data set $\mathcal{D}_i$ (Lines 4-5). Then BPG updates $\boldsymbol{\theta}_i$ with an empirical estimate of Theorem 3.1 (Line 6). After n iterations, the BPG estimate of $v(\pi_e)$ is:

$$\text{IS}(\pi_e, \mathcal{D}_n) = \frac{1}{n \cdot k} \sum_{\{H, \pi_{\boldsymbol{\theta}}\} \in \mathcal{D}_n} \text{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}}).$$

Algorithm 2 Behavior Policy Gradient

Input: Evaluation policy parameters, $\boldsymbol{\theta}_e$, batch size k, a step-size for each iteration, α_i, and number of iterations n.

Output: Final behavior policy parameters $\boldsymbol{\theta}_n$ and the IS estimate of $v(\pi_e)$ using all sampled trajectories.

1: $\boldsymbol{\theta}_0 \leftarrow \boldsymbol{\theta}_e$
2: $\mathcal{D}_0 = \{\}$
3: **for all** $i \in 0...n$ **do**
4: $\quad \mathcal{B}_i =$ Sample k trajectories $H \sim \pi_{\boldsymbol{\theta}_i}$
5: $\quad \mathcal{D}_{i+1} = \mathcal{D}_i \cup \mathcal{B}_i$
6: $\quad \boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \frac{\alpha_i}{k} \sum_{j=1}^{k} \text{IS}(\pi_e, h_j, \pi_{\boldsymbol{\theta}})^2 \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_i}(a_t^j | s_t^j)$
7: **end for**
8: **Return** $\boldsymbol{\theta}_n$, $\text{IS}(\pi_e, \mathcal{D}_n)$

Given that the gradient descent step-size parameters are consistent with standard gradient descent convergence conditions, BPG will converge to a behavior policy that locally minimizes the variance.[8] At best, BPG converges to the globally optimal behavior policy *within the parameterization of* π_e. Since the parameterization of π_e

[8] Gradient descent converges when the step-size at iteration i, α_i, is chosen such that $\sum_{i=0}^{\infty} \alpha_i = \infty$ and $\sum_{i=0}^{\infty} \alpha_i^2 < \infty$ (Bertsekas and Tsitsiklis, 2000).

determines the class of representable distributions it is possible that the theoretically optimal variance behavior policy is unrepresentable under this parameterization. Nevertheless, a suboptimal behavior policy still yields lower variance estimates of $v(\pi_e)$, provided it decreases variance compared to on-policy returns.

Though IS is an unbiased estimator, the estimate returned by Algorithm 2 is *not* necessarily unbiased since trajectories are non-i.i.d. Nevertheless, we show in Appendix C.3 that the estimate returned by Algorithm 2 is unbiased.

3.4.1 Control Variate Extension

In cases where an approximate model of the environment is available, we can further lower variance by adapting the behavior policy of the doubly robust estimator (Jiang and Li, 2016; Thomas and Brunskill, 2016a). The Doubly Robust (DR) estimator computes the average difference between the observed importance-sampled rewards and the predicted expected reward under a model of the environment's transition and reward function. Provided the expected reward predictions are correlated with the true rewards, DR has lower variance than using the importance-sampled rewards alone (See Section 2.2.3.3 for full details of the DR estimator).

We show here that we can adapt the behavior policy to lower the mean squared error of DR estimates. We denote this new method DR-BPG for *doubly robust behavior policy gradient*.

Let $w_{\pi,t}(H) = \prod_{i=0}^{t} \pi(A_t|S_t)$ and $\hat{v}^{\pi_e}$ and $\hat{q}^{\pi_e}$ be the state and action value functions of π_e in the approximate model. Recall from Section 2.2.3.3 that the DR estimator for a single trajectory is:

$$\mathrm{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e}) \coloneqq \hat{v}(S_0, 0) + \sum_{t=0}^{L-1} \frac{w_{\pi_e,t}}{w_{\pi_{\boldsymbol{\theta}},t}} (R_t - \hat{q}^{\pi_e}(S_t, A_t, t) + \hat{v}^{\pi_e}(S_{t+1}, t+1)).$$

We can reduce the mean squared error of DR with gradient descent using

unbiased estimates of the gradient given by the following corollary to Theorem 3.1

Corollary 3.1.

$$\frac{\partial}{\partial \boldsymbol{\theta}} \text{MSE}\left[DR(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e})\right] = \mathbf{E}[(\text{DR}(\pi_e, H, \boldsymbol{\theta}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e})^2 \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \\ - 2\,\text{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e})(\sum_{t=0}^{L-1} \gamma^t \delta_t \frac{w_{\pi_e,t}}{w_{\boldsymbol{\theta},t}} \sum_{i=0}^{t} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_i|S_i))]$$

where $\delta_t = R_t - \hat{q}^{\pi_e}(S_t, A_t, t) + \hat{v}^{\pi_e}(S_{t+1}, t+1)$ *and the expectation is taken over* $H \sim \pi_{\boldsymbol{\theta}}$.

Proof. See Appendix C.1.3 for the full proof.

The first term of $\frac{\partial}{\partial \boldsymbol{\theta}} MSE$ is analogous to the gradient of the importance-sampling estimate with $\text{IS}(\pi_e, H, \boldsymbol{\theta})$ replaced by $\text{DR}(\pi_e, H, \boldsymbol{\theta}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e})$. The second term accounts for the covariance of the DR terms.

We can obtain the MSE gradient for PDIS as the special case of Corollary 3.1 where $\hat{q}^{\pi_e}$ and $\hat{v}^{\pi_e}$ are equal to zero for all (s, a) pairs. We provide the expression for this gradient in Appendix C.1.4.

In practice, DR has been noted to perform best when all trajectories are used to estimate the model and then also used to estimate $v(\pi_e)$ (Thomas and Brunskill, 2016a). However, for DR-BPG, updating the model as $\pi_{\boldsymbol{\theta}}$ is learned will change the the surface of the MSE objective we seek to minimize and thus DR-BPG will only converge once the model stops changing. Computing the model from the same data used in the DR estimate also violates assumptions made for the theoretical analysis of DR (Thomas and Brunskill, 2016a). In our experiments, we consider both a changing and a fixed model.

3.4.2 Connection to REINFORCE

BPG is closely related to existing work in batch policy gradient RL (c.f., Sutton et al.

(2000a)) and we draw connections between one such method and BPG to illustrate how BPG changes the distribution of trajectories. The REINFORCE algorithm (Williams, 1992) attempts to maximize $v(\pi_{\boldsymbol{\theta}})$ through gradient ascent on $v(\pi_{\boldsymbol{\theta}})$ using unbiased estimates of the gradient of $v(\pi_{\boldsymbol{\theta}})$:

$$\frac{\partial}{\partial \boldsymbol{\theta}} v(\pi_{\boldsymbol{\theta}}) = \mathbf{E}\left[g(H) \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \middle| H \sim \pi_{\boldsymbol{\theta}} \right].$$

Intuitively, REINFORCE increases the probability of all actions taken during H as a function of $g(H)$. This update increases the probability of actions that lead to high return trajectories. BPG can be interpreted as REINFORCE where the return of a trajectory is the square of its importance-sampled return. Thus BPG increases the probability of all actions taken along H as a function of $IS(\pi_e, H, \boldsymbol{\theta})^2$. The magnitude of $IS(\pi_e, H, \boldsymbol{\theta})^2$ depends on two qualities of H:

1. The magnitude of $g(H)^2$.
2. The relative likelihood of H under π_e compared to $\pi_{\boldsymbol{\theta}}$ (i.e., $\prod_{t=0}^{L-1} \frac{\pi_e(A_t|S_t)}{\pi_{\boldsymbol{\theta}}(A_t|S_t)}$).

These two qualities demonstrate a balance in how BPG changes trajectory probabilities. Increasing the probability of a trajectory under $\pi_{\boldsymbol{\theta}}$ will decrease $IS(\pi_e, H, \boldsymbol{\theta})^2$ and so BPG increases the probability of a trajectory when $g(H)^2$ is large enough to offset the decrease in $IS(\pi_e, H, \boldsymbol{\theta})^2$ caused by decreasing the importance weight.

3.5 Empirical Study

This section presents an empirical study of variance reduction through behavior policy search. Our experiments are principally designed to answer the question, "Can behavior policy search with BPG reduce policy value estimation MSE compared

to on-policy estimates in both tabular and continuous domains?" We address this question by evaluating BPG in three domains.

3.5.1 Empirical Set-up

We briefly describe our experimental domains here; full details are provided in Appendix F.1.

The first domain is a 4x4 Grid World with a terminal state with reward 10 at $(3, 3)$, a state with reward -10 at $(1, 1)$, a state with reward 1 at $(1, 3)$, and all other states having reward -1. Policies are represented with a state-dependent soft-max distribution over actions. In this domain we can study BPG without concern of whether or not an improved behavior policy is representable in our class of function approximator.

We obtain two evaluation policies by applying the REINFORCE algorithm (Williams, 1992) to the task, starting from a policy that selects actions uniformly at random. We then select one evaluation policy, π_1, from the early stages of learning – an improved policy but still far from having converged – and one after learning that has almost converged, π_2. We run our main experiments once with $\pi_e \coloneqq \pi_1$ and a second time with $\pi_e \coloneqq \pi_2$.

Our second and third tasks are the continuous control Cart Pole Swing Up and Acrobot tasks implemented within RLLAB (Duan et al., 2016). These domains require that BPG optimize the behavior policy within a given class of function approximator. In both domains, the evaluation policy is a neural network that maps the state to the mean of a Gaussian distribution. The specific evaluation policies are obtained by partially optimizing randomly initialized policies using the trust-region policy optimization algorithm (Schulman et al., 2015a).

In all domains we run multiple trials where each trial consists of a fixed number of iterations. At each iteration, each algorithm collects a batch of trajectories and

computes a new estimate of $v(\pi_e)$. All algorithms have access to the same number of trajectories at the same iteration across trials.

3.5.2 Empirical Results

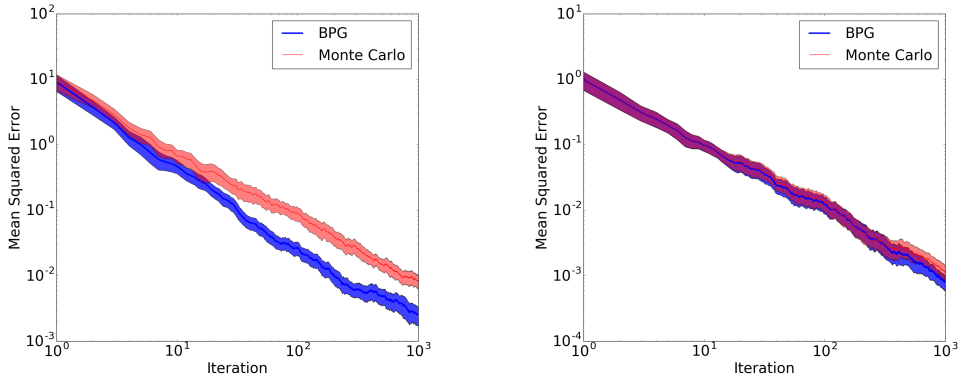
We now present our empirical results.

3.5.2.1 Grid World

Figure 3.1 compares BPG to the on-policy Monte Carlo estimator for both Grid World policies, π_1 and π_2. At each iteration, each method collects 100 additional trajectories. BPG uses a step-size, α, of 5×10^{-6}. Our main point of comparison is the mean squared error (MSE) of both estimates at iteration i over 100 trials. For π_1, BPG significantly reduces the MSE of on-policy estimates (Figure 3.1a). For π_2, BPG also reduces MSE, however, it is only a marginal improvement.

At the end of each trial we used the final behavior policy to collect 100 more trajectories and estimate $v(\pi_e)$. In comparison to a Monte Carlo estimate with 100 trajectories from π_1, MSE is 85.48% lower with this improved behavior policy. For π_2, the MSE is 31.02% lower. This result demonstrates that BPG can find behavior policies that substantially lower MSE.

To understand the disparity in performance when π_e changes, we plot the distribution of returns under each π_e (Figures 3.2b and 3.2c). These plots show the variance of the returns sampled by π_1 is much higher; π_1 sometimes samples returns with twice the magnitude of any sampled by π_2. To quantify the decrease in variance for BPG, we also measure and plot the variance of $\text{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}_i})$ for each of π_1 and π_2 (Figure 3.2a). The high initial variance when π_e is π_1 means there is much more room for BPG to improve the behavior policy.



(a) Results with $\pi_e \coloneqq \pi_1$

(b) Results with $\pi_e \coloneqq \pi_2$

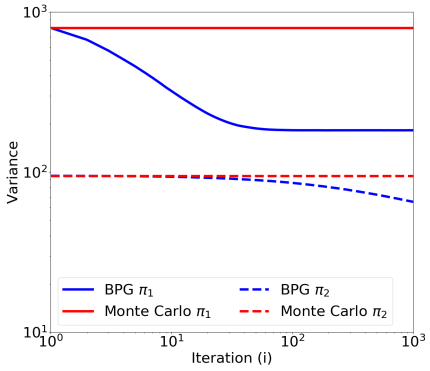
Figure 3.1: Grid World experiments when π_e is a partially optimized policy, π_1, (3.1a) and a converged policy, π_2, (3.1b). Results are averaged over 100 trials of 1000 iterations with a shaded region representing a 95% confidence interval. The y-axis shows the mean squared error and the x-axis shows the iteration number. Axes are log-scaled. In both instances, BPG lowers MSE more than on-policy Monte Carlo returns (statistically significant, $p < 0.05$).

3.5.2.2 Step-Size Sensitivity

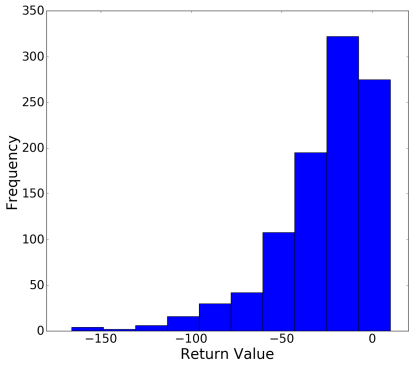
BPG requires setting a step-size parameter for the stochastic gradient descent update. We evaluated BPG under different settings of this parameter to determine robustness to how the step-size is set. Figure 3.3 shows variance reduction for the tested settings. This figure show that BPG can be robust to different settings of the step-size, however, for particularly large step-sizes, the algorithm may diverge.

3.5.2.3 Continuous Control

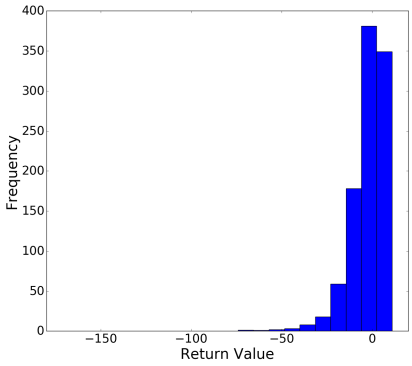
Figure 3.4 shows reduction of MSE on the Cart Pole Swing Up and Acrobot domains. Each method collects 500 trajectories at each iteration and BPG uses a step-size of 5×10^{-5}. Again we see that BPG reduces MSE faster than Monte Carlo value estimation. In contrast to the discrete Grid World experiment, this experiment demonstrates the applicability of BPG to the continuous control setting. While



(a) Variance Reduction



(b) Histogram of π_1 Returns



(c) Histogram of π_2 Returns

Figure 3.2: Comparison of variance reduction between π_1 and π_2 in Grid World domain. Figure 3.2a shows variance on the y-axis and iteration number on the x-axis. These axes are log-scaled. Results are plotted for Monte Carlo value estimation with π_1 and π_2 and for BPG evaluations of π_1 and π_2. Results are averaged over 100 trials of 1000 iterations. Figures 3.2b and 3.2c give the distribution of returns under the two different π_e.

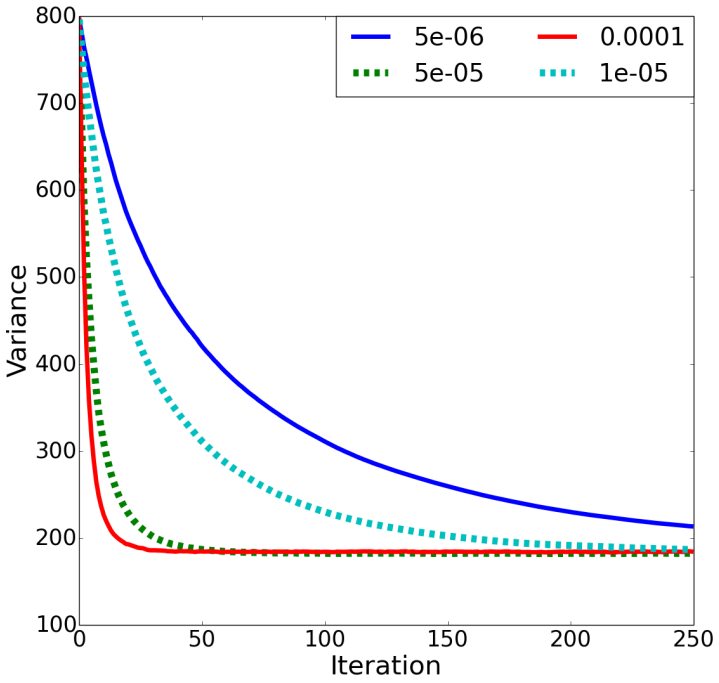
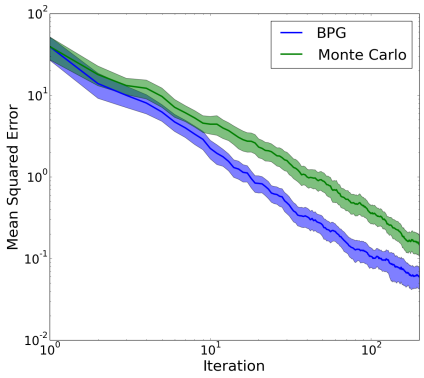


Figure 3.3: Step-size Sensitivity: this figure shows the effect of varying the step-size parameter for representative settings (BPG diverged for step-size values higher than the ones shown). The y-axis is the variance of the importance sampling estimator and the x-axis is the iteration number. Unlike other figures in this chapter, axes are *not* log-scaled. We ran BPG for 250 iterations per step-size setting and averaged results over 5 trials.

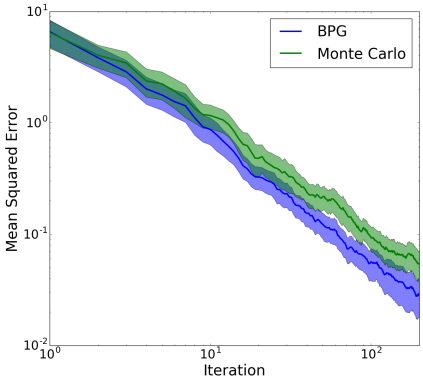
BPG significantly outperforms Monte Carlo value estimation in Cart Pole Swing Up, the gap is much smaller in Acrobot. This result also demonstrates that BPG (and behavior policy search) can lower the variance of policy value estimation when the policy must generalize across different states.

3.5.3 Control Variate Extension Results

In this section, we evaluate the combination of model-based control variates with the behavior policy gradient algorithm. Specifically, we compare doubly robust BPG



(a) Cart Pole Swing Up MSE



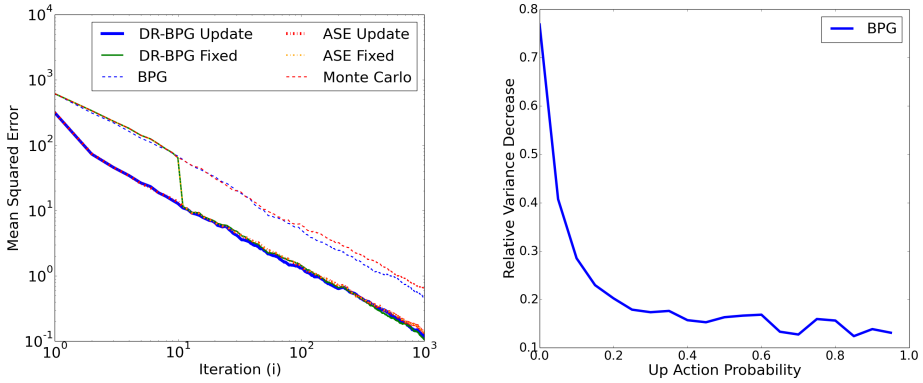
(b) Acrobot MSE

Figure 3.4: Mean squared error reduction on the Cart Pole Swing Up and Acrobot domains. The y-axis gives MSE and the x-axis is the iteration number. Axes are log-scaled. We adapt the behavior policy for 200 iterations and average results over 100 trials. Error bars are for 95% confidence intervals.

(DR-BPG) with the Advantage-Sum Estimator (introduced in Section 2.2.3.3 and referred to as ASE) that uses $\boldsymbol{\theta}_i = \boldsymbol{\theta}_e$ for all i.

In these experiments we use a 10x10 stochastic Grid World where the added stochasticity and increased size increase the difficulty of building an accurate model from data. The layout of this Grid World is identical to the deterministic Grid World except the terminal state is at $(9, 9)$ and the $+1$ reward state is at $(1, 9)$. When the agent moves, it moves in its intended direction with probability 0.9, otherwise it goes left or right with equal probability. Noise in the environment increases the difficulty of building an accurate model from trajectories.

Since these methods require a model we construct this model in one of two ways. The first method uses all trajectories in $\mathcal{D}$ to build the model and then uses the same set to estimate $v(\pi_e)$ with ASE or DR. The second method uses trajectories from the first 10 iterations to build the model and then fixes the model for the remaining iterations. For DR-BPG, behavior policy search starts at iteration 10 under this second condition. We call the first method "Update" and the second



(a) Control Variate MSE

(b) Rare Event Improvement

Figure 3.5: **3.5a:** Comparison of DR-BPG and ASE (on-policy DR) on a larger stochastic Grid World. For the fixed model methods, the significant drop in MSE at iteration 10 is due to the introduction of the model control variate. For clarity we do not show error bars. The mean difference between the final estimate of DR-BPG and ASE with the fixed model averaged over 300 trials is statistically significant ($p < 0.05$); the difference between the same methods with a constantly improving model is not. **3.5b:** Varying the probability of a high rewarding terminal action in the Grid World domain. Each point on the horizontal axis is the probability of taking this action. The vertical axis gives the mean relative decrease in variance after adapting $\boldsymbol{\theta}$ for 500 iterations. Denoting the initial variance as V_i and the final variance as V_f, the relative decrease is computed as $\frac{V_i - V_f}{V_i}$. Results are averaged over 100 trials. A 95% confidence interval region is shaded around the mean but is small.

method "Fixed." The update method invalidates the theoretical guarantees of these methods but learns a more accurate model. In both instances, we estimate $P(s, a, s')$ as the proportion of times that the agent transitions to state s' after taking action a in state s. Similarly, we estimate $r(s, a)$ as the mean reward received after taking action a in state s.

Figure 3.5 demonstrates that combining BPG with a model-based control variate (DR-BPG) can lead to further reduction of MSE compared to the control variate alone (ASE). Specifically, with the fixed model, DR-BPG outperformed all other methods. DR-BPG using the update method for building the model performed

competitively with ASE although its MSE was not statistically significantly lower. We also evaluate the final learned behavior policy of the fixed model variant of DR-BPG. For a batch size of 100 trajectories, the DR estimator with this behavior policy improves upon the ASE estimator with the same model by 56.9%.

For DR-BPG, estimating the model with all data still allowed steady progress towards lower variance. This result is interesting since a changing model changes the surface of our variance objective and thus gradient descent on the variance has no theoretical guarantees of convergence. Though we do not include an experimental study, we observed that setting the step-size, α, for DR-BPG was more challenging for either model type. Thus while we have shown BPG can be combined with control variates, more work is needed to produce a robust method.

3.5.4 Rareness of Event Study

Our final experiment aims to understand how the gap between on- and off-policy variance is affected by the probability of rare events. The intuition for why behavior policy search can lower the variance of on-policy estimates is that a well selected behavior policy can cause rare and high magnitude events to occur. We test this intuition by varying the probability of a rare, high magnitude event and observing how this change affects the performance gap between on- and off-policy value estimates. For this experiment, we use a variant of the deterministic Grid World where taking the UP action in the initial state (the upper left corner) causes a transition to the terminal state with a reward of +50. We use π_1 from our earlier Grid World experiments but we vary the probability, p, of choosing UP when in the initial state so that with probability p the agent will receive a large reward and end the trajectory. We use a constant step-size of 10^{-5} for all values of p and run BPG for 500 iterations. We plot the relative decrease of the variance as a function of p over 100 trials for each value of p. We use relative variance to normalize across problem instances. Note

that under this measure, even when p is close to 1, the relative variance is not equal to zero because as p approaches 1 the initial variance also goes to zero.

This experiment illustrates that as the initial variance increases, the amount of improvement BPG can achieve increases. As p becomes closer to 1, the initial variance becomes closer to zero and BPG barely improves over the variance of Monte Carlo (in terms of absolute variance there is no improvement). When π_e rarely takes the high rewarding UP action (p close to 0), BPG improves policy value estimation by increasing the probability of this action. This experiment supports our intuition for why off-policy data collection can be preferable to on-policy data collection.

3.6 Summary

This chapter has described Contribution 1 of this dissertation. In this chapter we introduced the behavior policy search problem in order to improve estimation of $v(\pi_e)$ for a fixed evaluation policy π_e. We presented an algorithm – behavior policy gradient – for this problem which adapts the behavior policy with stochastic gradient descent on the variance of the importance-sampling estimator. Finally, we presented empirical results which demonstrate that BPG lowers the mean squared error of estimates of $v(\pi_e)$ compared to on-policy estimates in three RL tasks.

We also extended BPG to the doubly robust estimator and showed that we can further improve the accuracy of policy value estimation by combining behavior policy search with control variate variance reduction. In the future, the ideas behind BPG could be combined with additional off-policy estimators to further lower the variance of policy value estimation. Finally, we presented an experiment showing that the rareness of high magnitude return trajectories plays a role in how much improvement BPG provides. In the next chapter, we show that BPG can also be used to lower the variance of policy gradient estimates for the problem of policy improvement in reinforcement learning.

Chapter 4

Collecting Data for Off-Policy Policy Improvement

In the previous chapter we showed that a properly selected behavior policy led to significant improvements in policy value estimation. In this chapter, we show that a properly selected behavior policy can lower the variance of batch policy gradient policy improvement.

This chapter contributes an initial study of combining behavior policy search with policy improvement. Specifically, we consider optimizing a behavior policy with a behavior policy search algorithm to lower the variance of batch policy gradient estimates. We then use this policy to collect trajectories for an importance-sampled policy gradient estimate. Empirical results show promise for increasing the effectiveness of batch policy gradient RL in this way. We also introduce an algorithm that simultaneously optimizes the behavior policy to minimize variance while optimizing a target policy to maximize the expected return. Empirical results show this method can lead to faster policy improvement though it does so less reliably than an on-policy learning method. Finally we discuss how adapting the behavior policy to provide lower variance batch policy gradient estimates relates to the traditional notion of

exploration in RL. This initial study constitutes Contribution 2 of this dissertation.[9]

The contribution in this chapter pertains to how an RL agent should collect data for low variance policy gradient estimates. We survey literature related to this question in Section 9.1. The question of how to collect data for low variance policy gradient estimates also relates to the question of how an RL agent should collect data to *explore* its environment and find an optimal policy. We do *not* address the exploration question. We discuss in Section 4.5.1 that the objective of variance reduction may be at odds with the objective of exploration.

4.1 Importance Sampled Batch Policy Gradient

In this section we introduce an off-policy version of the policy gradient expression from Section 2.3. This expression allows us to turn on-policy batch policy gradient methods into off-policy batch policy gradient methods.

In Section 2.3, we provided two expressions for the policy gradient. Here, we recall the gradient expressed in terms of an expectation over trajectories:

$$\frac{\partial}{\partial \boldsymbol{\theta}} v(\pi_{\boldsymbol{\theta}}) = \mathbf{E}\left[g(H) \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \middle| H \sim \pi_{\boldsymbol{\theta}} \right]. \tag{4.1}$$

The simplest method that uses estimates of (4.1) to update the policy is the REINFORCE algorithm Williams (1992). We refer to this algorithm as *on-policy* REINFORCE when the gradient estimate uses trajectories collected with the current policy $\pi_{\boldsymbol{\theta}}$ to estimate the gradient at $\boldsymbol{\theta}$.

As written in (4.1), the gradient must be estimated with trajectories sampled from $\pi_{\boldsymbol{\theta}}$. We can use importance sampling to generalize (4.1) to be estimated with

[9] This chapter contains work that was previously published at the 2018 AAAI Spring Symposium on Data Efficient Reinforcement Learning (Hanna and Stone, 2018). It also contains unpublished work done in collaboration with Xiang Gu.

trajectories from any policy.

$$\frac{\partial}{\partial \boldsymbol{\theta}} v(\pi_{\boldsymbol{\theta}}) = \mathbf{E}\left[\mathrm{IS}(\pi_{\boldsymbol{\theta}}, H, \pi_b) \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \middle| H \sim \pi_b\right]. \tag{4.2}$$

We refer to the batch policy gradient algorithm that uses estimates of (4.2) as *off-policy* REINFORCE.

The benefit of expressing the policy gradient in this form is that we can estimate (4.2) with trajectories from *any* policy. The downside is that off-policy estimates with importance sampling may have high variance if π_b is not chosen carefully. For this reason, most batch policy gradient methods in the literature are on-policy (Schulman et al., 2017; Gu et al., 2017a).

4.2 Batch Policy Gradient Estimation with an Improved Behavior Policy

The choice of π_b partially determines the variance of estimates of (4.2). As in the previous chapter, we expect that a properly selected behavior policy will decrease estimation variance. Behavior policy search gives us a method for properly selecting the behavior policy.

The simplest instantiation of this idea requires first learning a behavior policy that generates data for low variance importance sampling estimates and then using it to estimate the off-policy policy gradient which will be used to update a target policy. Specifically, we first run BPG as described in the previous chapter. We then use the resulting behavior policy to collect data to estimate the importance sampled policy gradient and update the target policy.

The main limitation of this approach is that we require data to learn the behavior policy before we collect data for policy improvement. Thus even if we lower variance, it is unclear there is a data reduction benefit to this exact approach.

Instead of producing a data efficient algorithm, we focus here on the question of whether or not an optimized behavior policy can be used for more accurate gradient estimation.

The other drawback is that the optimized behavior policy is just for the initial target policy. The optimal behavior policy depends on the target policy. Thus, as the target policy changes, the optimal behavior policy should change as well.

4.3 Parallel Policy Search: Towards a Full Algorithm

In the preceding section, we discussed how a properly chosen behavior policy could lower the variance of policy gradient estimation. However, as the target policy changes during learning, the behavior policy should also change to remain a behavior policy that gives low variance gradient estimates for the current target policy. In this section, we propose a method for incorporating behavior policy search into a full batch policy gradient algorithm.

The proposed algorithm alternates between updating the target policy to maximize expected return and updating the behavior policy to minimize the variance of evaluating the target policy. We call this algorithm *parallel policy search* (PPS) and provide pseudo-code in Algorithm 3. Parallel policy search initializes the behavior policy to be identical to the target policy (Line 1). It then repeatedly, samples trajectories with the behavior policy (Line 3), updates the target policy with an unbiased estimate of the off-policy policy gradient (Line 4), and then updates the behavior policy with an unbiased estimate of the behavior policy gradient (Line 5).

4.4 Empirical Study

In this section we conduct experiments to see whether a properly selected behavior policy for off-policy batch policy gradient updates leads to more efficient policy

Algorithm 3 Parallel Policy Search

input Initial policy parameters $\boldsymbol{\theta}$; Step-size α and β.
output A learned target policy parameter $\boldsymbol{\theta}$ such that $\pi_{\boldsymbol{\theta}} \approx \pi^{\star}_{\boldsymbol{\theta}}$

1: Initialize $\boldsymbol{\theta}_b \leftarrow \boldsymbol{\theta}$
2: **loop**
3: Sample m trajectories from behavior policy: $H \sim \pi_{\boldsymbol{\theta}_b}$
4: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha * \frac{1}{m} \sum_{j=1}^{m} \text{IS}(\pi_{\boldsymbol{\theta}}, H_j, \pi_{\boldsymbol{\theta}_b}) \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t^j | S_t^j)$
5: $\boldsymbol{\theta}_b \leftarrow \boldsymbol{\theta}_b + \beta * \frac{1}{m} \sum_{j=1}^{m} \text{IS}(\pi_{\boldsymbol{\theta}}, H_j, \pi_{\boldsymbol{\theta}_b})^2 \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_b}(A_t^j | S_t^j)$
6: **end loop**

improvement than on-policy batch policy gradient estimates. We design experiments to answer the questions:

1. Does a behavior policy selected with BPG lead to more accurate estimation of the policy gradient direction compared to on-policy estimates?

2. Can a behavior policy selected with BPG for the initial target policy improve learning speed across multiple policy gradient updates before it must be re-optimized for the current target policy?

3. Can off-policy PPS learn faster than an on-policy batch policy gradient method?

4.4.1 Empirical Set-up

We answer the above questions with experiments using the Cart Pole domain implented in the OpenAI gym (Brockman et al., 2016). Both the behavior policy and target policy are represented as a softmax distribution over actions where the logits come from a linear combination of state variables. The initial behavior policy is trained with BPG to minimize the variance of an importance sampling evaluation of the initial target policy.

4.4.1.1 Policy Improvement Step Quality

Our first experiment compares the quality of the update direction computed with an off-policy REINFORCE method to the quality of the update direction computed with on-policy REINFORCE. In order to make this comparison, we sample a batch of trajectories with the initial policy and another batch with π_b. We estimate the on-policy REINFORCE gradient, the off-policy REINFORCE gradient estimated with a behavior policy trained with BPG to evaluate the initial policy, and the off-policy REINFORCE gradient estimated with a randomly initialized behavior policy. We aim to identify which approach yields a gradient pointing in the direction of highest performance improvement.

Since performance improvement is a function of both gradient direction and gradient magnitude, we select the optimal step-size, α, for each gradient estimate. Specifically, after estimating the gradient $\mathbf{g}$, we perform a line-search over α to find the value that maximizes $v(\pi_{\boldsymbol{\theta}'})$ where $\boldsymbol{\theta}' = \boldsymbol{\theta} + \alpha\mathbf{g}$. We begin with an initial α value of 1×10^{-4}, evaluate $v(\pi_{\boldsymbol{\theta}'})$, then double α, and repeat the process until $v(\pi_{\boldsymbol{\theta}'})$ stops increasing. The goal of this procedure is to avoid conflating the ability to estimate gradient direction with the magnitude of the estimated gradient step.[10] Results are shown in Table 4.1.

4.4.1.2 Multi-step Policy Improvement

Our second experiment investigates if a behavior policy trained to evaluate the initial policy can be used to estimate the policy gradient at other policies. For this experiment, we collect a single set of 100 trajectories with the behavior policy and adapt the target policy with off-policy REINFORCE for 10 iterations. We compare to on-policy REINFORCE. Results are shown in Figure 4.1.

[10] An alternative procedure could be to normalize each gradient estimate and use the same step-size for the comparison.

4.4.1.3 Parallel Policy Search

Our final experiment compares the learning speed of parallel policy search (PPS) and an on-policy batch policy gradient algorithm. We implement PPS within the OpenAI Baselines (Dhariwal et al., 2017) implementation of trust-region policy optimization (TRPO) (Schulman et al., 2015a). TRPO is an on-policy batch policy gradient method that, at each iteration, adapts the step-size, α, to take as large a step as possible subject to a constraint on the KL-divergence between the action distribution of successive policies. In our implementation of parallel policy search, we use the TRPO step-size selection mechanism for both behavior and target policy updates. Full details of this approach are given in Appendix F.2.

We compare PPS to TRPO on a variant of the Cart Pole problem where we only allow episodes to last for up to 25 time-steps. This modification simplifies the problem by reducing the number of $\frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\boldsymbol{\theta}_b}(a|s)}$ factors in the full-trajectory importance weight. Results are shown in Figure 4.2.

4.4.2 Empirical Results

We now present the results of these experiments.

4.4.2.1 Policy Improvement Step Quality

Method	Average Return (std.)
Random π_b	54.92 (8.27)
On-policy	55.081 (1.31)
Optimized π_b	**68.656 (15.7)**

Table 4.1: Comparison of one-step improvement in average return when estimating the policy gradient with off-policy and on-policy REINFORCE. For each behavior policy we sample 200 trajectories and estimate the policy gradient direction with (4.2). We then perform a line search along the gradient to find the step-size that maximally increases the average return of the target policy. Results are averaged over 50 independent runs.

Table 4.1 shows that the average gradient direction computed with off-policy REINFORCE leads to a much larger increase in expected return. However, we also note that the variance of the performance improvement is also higher. While in most cases expected performance increases above the increase obtained by on-policy REINFORCE or random policy off-policy REINFORCE, the fact that the variance of the improvement has increased may suggest that lowering the variance of policy value estimates does *not* necessarily lead to a lower variance batch policy gradient estimate.

4.4.2.2 Multi-step Policy Improvement

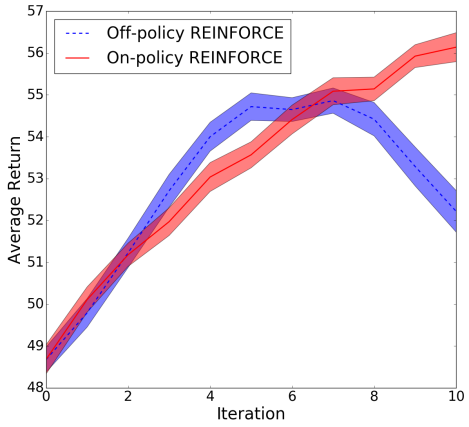


Figure 4.1: Comparison of multi-step improvement in average return when estimating the policy gradient with off-policy and on-policy REINFORCE.

Figure 4.1 demonstrates that an improved π_b for importance sampling evaluation can lead to faster learning compared to on-policy REINFORCE – even without re-sampling new trajectories. However, the improvement is concentrated in the first few iterations of policy improvement before the target policy has changed significantly. This observation motivates the need for behavior policy adaptation as the target policy changes.

4.4.2.3 Parallel Policy Search

Figure 4.2a shows PPS learning faster than TRPO in the early iterations. However, in later iterations the mean performance of PPS drops below that of TRPO. A closer look at the individual algorithm runs, reveals that PPS can be unstable; it sometimes learns quickly and other times fails to improve $v(\pi_e)$ at all. On the other hand, TRPO learns slower but more stably across trials.

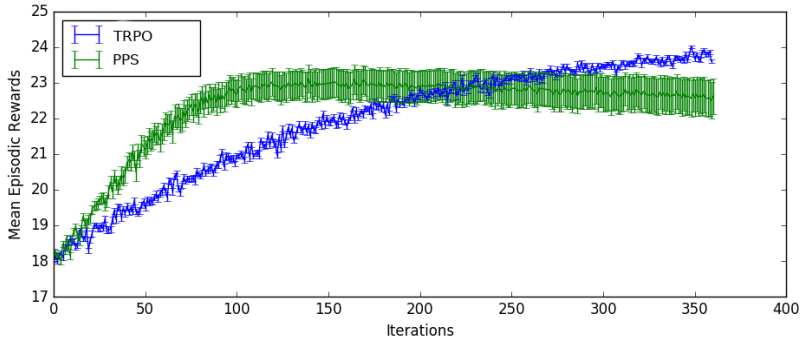
4.5 Challenges for an Efficient Algorithm

The empirical results in the preceding section demonstrate potential for parallel policy search but also demonstrates more work is needed to improve upon state-of-the-art on-policy algorithms. We now describe two challenges that must be overcome to scale up parallel policy search. We leave addressing these challenges for future work and discuss them further in Chapter 10.

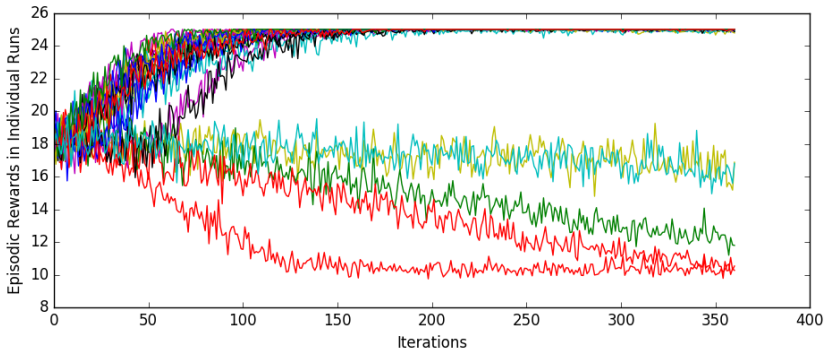
4.5.1 Variance Reduction vs. Exploration

One of the central challenges of reinforcement learning is the trade-off between exploration and exploitation. In standard policy gradient RL, we optimize a stochastic policy where sampling in the action space provides exploration (Sutton and Barto, 1998). Parallel policy search now replaces the exploration done by the target policy with exploration done by a policy optimized to have lower variance. Though the optimized stochastic behavior policy will still provide exploration, it is optimized to reduce variance – *not* try out new actions.

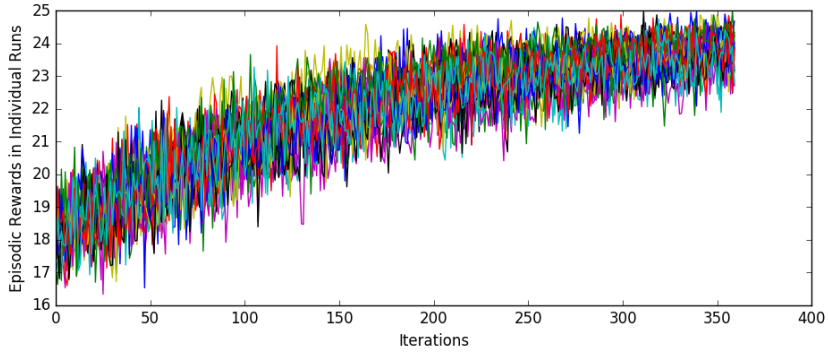
The main risk of simultaneous variance reduction and exploration is premature convergence to a sub-optimal policy. Consider that the initial policy likely puts very low probability on sequences of actions that lead to high return. Since these sequences will likely go unobserved in early iterations, the behavior policy search will



(a) Mean Performance



(b) PPS Individuals



(c) TRPO Individuals

Figure 4.2: Parallel policy search (PPS) compared to trust-region policy optimization (TRPO) on a modified version of the Cart Pole problem. Each method is run 25 times for 350 iterations each. Figure 4.2a plots the mean performance for each algorithm with a 95% confidence interval. Figure 4.2b plots each of the 25 PPS trials separately. Figure 4.2c plots each of the 25 TRPO trials separately.

be unaware of them. Adapting the behavior policy may result in taking probability mass from these unobserved high return sequences and giving it to lower return but observed sequences. The result is that the behavior policy may stop exploring some potentially good actions and the target policy will never receive data to learn about how to act in other parts of the state space. Premature convergence of the behavior and target policy may be part of the reason that some trials of PPS converge to poor solutions (Figure 4.2b).

4.5.2 Synchronizing Target and Behavior Policy Updates

The other main challenge is that parallel policy search involves optimizing two policies at the same time. It is known to be difficult to tune the step-size for policy gradient methods. Parallel policy search now requires tuning two step-size parameters. Further complicating the problem is that the steps must be synchronized so that the behavior policy tracks the current target policy. Too small or too large behavior policy updates may result in a behavior policy that fails to lower variance for the target policy at a specific iteration. The challenge of synchronizing policy updates may also be part of why PPS sometimes fails to learn the optimal policy in our experiments.

4.6 Summary

We have presented preliminary steps towards a batch policy gradient algorithm that uses off-policy data for more efficient updates. We have described how the behavior policy search algorithm introduced in Chapter 3 could be adapted to the policy improvement setting. We then presented experiments showing that a carefully selected behavior policy can improve the step direction of the REINFORCE method and that this same behavior policy can be used for multiple updates before it performs worse than an on-policy update. These results indicate that research into how to

adapt the behavior policy as the policy being learned changes has the potential to further improve the data efficiency of batch policy gradient reinforcement learning. We also introduced an algorithm – parallel policy search – that simultaneously adapts the behavior policy to lower variance while optimizing the target policy to increase expected return. Experiments showed PPS has promise for increasing the learning speed of batch policy gradient RL but also showed more work is needed for a robust algorithm. These experiments are an initial study of using a behavior policy search algorithm for more efficient policy improvement and constitute Contribution 2 of this dissertation.

Chapter 5

Weighting Data for Off-policy Policy Value Estimation

In this chapter, we consider how an RL agent should *weight* already-sampled data for off-policy policy value estimation with a fixed evaluation policy. Specifically, we first note that the off-policy policy value estimation technique of importance sampling may suffer from what we term *sampling error* – actions are observed in a proportion different than their true probability under the behavior policy. This sampling error degrades the data efficiency of off-policy policy value estimation. We then introduce a family of estimators that corrects for such sampling error to produce more accurate off-policy policy value estimation.[11] In contrast to Chapter 3 where we asked how to collect data for accurate policy value estimation, in this chapter, we ask how to weight already-sampled data for the most accurate policy value estimate. This new family of estimators is Contribution 3 of this dissertation.

The estimators we introduce in this chapter are importance sampling estimators that first estimate the policy that generated the data and then use this estimated

[11] This chapter contains work that was done in collaboration with Scott Niekum and previously published at ICML 2019 (Hanna et al., 2019).

policy as the behavior policy in an importance sampling estimate. Such methods are typically motivated by settings where the behavior policy, π_b is unknown and thus importance sampling could *not* be applied. In such settings it may be natural to assume that importance sampling will perform worse because it is using an estimate in place of the "correct" behavior policy probability. In this chapter we show that using the estimate *improves* the MSE of the importance sampling estimator. Even when π_b is known, our new methods provide more accurate policy value estimates than the importance sampling techniques described in Chapter 2. Furthermore, though this dissertation focuses on off-policy data, our new methods lead to more accurate policy value estimation in the on-policy setting as well.

Estimating the behavior policy has been studied before in contextual bandits (Narita et al., 2019; Xie et al., 2018) and contextless bandits (Li et al., 2015). We provide a more in depth survey of past work in Section 9.2. For now, we note that our work is the first introduction of such a method in the finite-horizon, episodic MDP setting.

Throughout this chapter we will refer to the importance sampling estimator (as introduced in Section 2.2.3.1) as the *ordinary importance sampling* (OIS) estimator and the ratio $\frac{\pi_e(a|s)}{\pi_b(a|s)}$ as the ordinary importance sampling weight. This change in terminology will allow us to distinguish between importance sampling using the true behavior policy and our new family of estimators – regression importance sampling estimators – that use importance sampling with an estimated behavior policy.

5.1 Limitations of Ordinary Importance Sampling

The ordinary importance sampling estimator may have high variance when the behavior policy is *not* chosen to minimize variance (as done in Chapter 3). A number of importance sampling variants have been proposed to address the high variance problem, however, all such variants use the OIS weight. The common reliance on

OIS weights suggests an implicit assumption by the RL community that OIS weights are the correct way to address the data distribution shift problem found when using off-policy data. While much research has gone into lowering the variance of importance sampling for off-policy reinforcement learning, one aspect of estimation that is rarely questioned is whether the ordinary importance weight is the correct way to re-weight off-policy data. Hence, when an application requires estimating an unknown π_b in order to compute importance weights, the application is implicitly assumed to only be approximating the desired weights.

However, OIS weights themselves are sub-optimal in at least one respect: the weight of each trajectory in the OIS estimate is inaccurate unless we happen to observe each trajectory according to its true probability. When the empirical

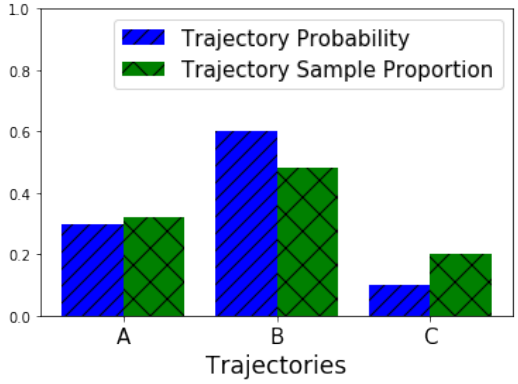


Figure 5.1: Sampling error in a discrete environment with three possible trajectories. Trajectories are sampled i.i.d. with the given probabilities and are observed in the given proportion. An OIS policy value estimate will place too much weight on Trajectory A and Trajectory C and too little weight on Trajectory B.

frequency of any trajectory is unequal to its expected frequency under π_b, the OIS estimator puts either too much or too little weight on the trajectory. We refer to error due to some trajectories being either over- or under-represented in $\mathcal{D}$ as *sampling error*. Figure 5.1 demonstrates the phenomenon of sampling error in a discrete MDP. Sampling error may be unavoidable when we desire an unbiased estimate of $v(\pi_e)$.

However, correcting for it by properly weighting trajectories will, in principle, give us a lower mean squared error estimate.

The problem of sampling error is related to a Bayesian objection to Monte Carlo integration techniques: OIS ignores information about how close trajectories in $\mathcal{D}$ are to one another (O'Hagan, 1987; Ghahramani and Rasmussen, 2003). This objection is easiest to understand in deterministic and discrete environments though it also holds for stochastic and continuous environments. In a deterministic environment, additional samples of any trajectory, h, provide no new information about $v(\pi_e)$ since only a single sample of h is required to know how to weight $g(h)$ in the estimate. However, the more times a particular trajectory appears, the more weight it receives in an OIS estimate even though the correct weighting of $g(h)$, $\Pr(h|\pi_e)$, is known since π_e is known. In stochastic environments, it is reasonable to give more weight to recurring trajectories since the recurrence provides additional information about the unknown state-transition probabilities. However, ordinary importance sampling also relies on sampling to approximate the known policy probabilities.

Finally, we note that the problem of sampling error applies to any variant of importance sampling using OIS weights, e.g., weighted importance sampling (Precup et al., 2000), per-decision importance sampling (Precup et al., 2000), the doubly robust estimator (Jiang and Li, 2016; Thomas and Brunskill, 2016a), and the MAGIC estimator (Thomas and Brunskill, 2016a). Sampling error is also a problem for on-policy Monte Carlo policy value estimation since the on-policy Monte Carlo estimator is a special case of the OIS estimator when the behavior policy is the same as the evaluation policy.

5.2 Regression Importance Sampling

In this section we introduce Contribution 3 of this dissertation: a family of estimators called *regression importance sampling* (RIS) estimators that correct for sampling

error in $\mathcal{D}$ by importance sampling with an estimated behavior policy. The motivation for this approach is that, though $\mathcal{D}$ was sampled with π_b, the trajectories in $\mathcal{D}$ may appear as if they had been generated by a different policy, $\pi_{\mathcal{D}}$. For example, if π_b would choose between two actions with equal probability in a particular state, the data might show that one action was selected more often than the other in that state. Thus instead of using OIS to correct from π_b to π_e, we introduce RIS estimators that corrects from $\pi_{\mathcal{D}}$ to π_e.

We assume that, in addition to $\mathcal{D}$, we are given a policy class – a set of policies – Π^n where each $\pi \in \Pi^n$ is a distribution over actions conditioned on the immediate preceding state and the last n states and actions preceding that state: $\pi : \mathcal{S}^{n+1} \times \mathcal{A}^n \to [0, 1]$. The RIS($n$) estimator first estimates the maximum likelihood behavior policy in Π^n given $\mathcal{D}$:

$$\pi_{\mathcal{D}}{}^{(n)} := \operatorname*{argmax}_{\pi \in \Pi^n} \sum_{i=1}^{m} \sum_{t=0}^{L-1} \log \pi(A_t^i | H_{t-n:t}^i). \tag{5.1}$$

We discuss the reasoning why estimating action probabilities conditioned on history is interesting in Section 5.2.1 and study the choice of the n parameter in Section 5.5.2.2. The RIS(n) estimate is then the importance sampling estimate with $\pi_{\mathcal{D}}{}^{(n)}$ replacing π_b.

$$\text{RIS}(n)(\pi_e, \mathcal{D}) := \frac{1}{m} \sum_{i=1}^{m} g(H_i) \prod_{t=0}^{L-1} \frac{\pi_e(A_t^i | S_t^i)}{\pi_{\mathcal{D}}{}^{(n)}(A_t^i | H_{t-n:t}^i)}$$

Analogously to OIS, we refer to $\frac{\pi_e(A_t|S_t)}{\pi_{\mathcal{D}}{}^{(n)}(S_t|H_{t-n:t})}$ as the RIS(n) weight for action A_t, state S_t, and trajectory segment $H_{t-n:t}$. Note that the RIS(n) weights are always well-defined since $\pi_{\mathcal{D}}{}^{(n)}$ never places zero probability mass on any action that occurred in $\mathcal{D}$.

We have introduced RIS as a family of estimators where different RIS methods

estimate the behavior policy conditioned on different history lengths. Among these estimators, our primary method of study is RIS(0). For larger n, RIS(n) may be less reliable for small sample sizes as the ${\pi_{\mathcal{D}}}^{(n)}$ estimate will be highly peaked (it will be 1 for most observed actions.) We verify this claim empirically below. However, as we discuss in Section 5.5.2.2, larger n may produce asymptotically better sampling error corrections and thus asymptotically better estimates.

5.2.1 Correcting Sampling Error in Discrete Action Spaces

We now present an example illustrating how RIS corrects for sampling error in sampled off-policy data. We make several limiting assumptions in this section with the intention to build intuition. These assumptions are *not* made for our theoretical and empirical analysis.

Consider an MDP with deterministic P and d_0 and finite $|\mathcal{S}|$ and $|\mathcal{A}|$. Let $\mathcal{H}$ be the (finite) set of possible trajectories under π_b and suppose that our observed data, $\mathcal{D}$, contains at least one of each $h \in \mathcal{H}$. With finite $\mathcal{S}$ and $\mathcal{A}$, the maximum likelihood behavior policy can be computed with count-based estimates. We define $c(h_{i:j})$ as the number of times that trajectory segment $h_{i:j}$ appears during any trajectory in $\mathcal{D}$. Similarly, we define $c(h_{i:j}, a)$ as the number of times that action a is observed following trajectory segment $h_{i:j}$ during any trajectory in $\mathcal{D}$. RIS(n) estimates the behavior policy as:

$$\pi_{\mathcal{D}}(a|h_{i-n:i}) := \frac{c(h_{i-n:i}, a)}{c(h_{i-n:i})}.$$

Observe that both OIS and all variants of RIS can be written in one of two forms:

$$\underbrace{\frac{1}{m}\sum_{i=1}^{m}\frac{w_{\pi_e}(H_i)}{w_{\pi}(H_i)}g(H_i)}_{(i)} = \underbrace{\sum_{h\in\mathcal{H}}\frac{c(h)}{m}\frac{w_{\pi_e}(h)}{w_{\pi}(h)}g(h)}_{(ii)}$$

where $w_\pi(h) = \prod_{t=0}^{L-1} \pi(a_t|s_t)$ and for OIS, $\pi := \pi_b$ and for RIS(n), $\pi := \pi_{\mathcal{D}}{}^{(n)}$ as defined in Equation (5.1).

If we had sampled trajectories using $\pi_{\mathcal{D}}{}^{(L-1)}$ instead of π_b, in our deterministic environment, the probability of each trajectory, h, would be $\Pr(H = h|H \sim \pi_{\mathcal{D}}{}^{(L-1)}) = \frac{c(h)}{m}$. Thus Form (ii) can be written as:

$$\mathbf{E}\left[\frac{w_{\pi_e}(H)}{w_\pi(H)} g(H) \middle| H \sim \pi_{\mathcal{D}}{}^{(L-1)}\right].$$

To emphasize what we have shown so far: OIS and RIS are both sample-average estimators whose estimates can be written as exact expectations. However, this exact expectation is under the distribution that trajectories were observed and *not* the distribution of trajectories under π_b.

Consider choosing $w_\pi := w_{\pi_{\mathcal{D}}}^{(L-1)}$ as RIS($L-1$) does. This choice results in (ii) being exactly equal to $v(\pi_e)$[12] On the other hand, choosing $w_\pi := w_{\pi_b}$ will *not* return $v(\pi_e)$ unless we happen to observe each trajectory at its expected frequency (i.e., $\pi_{\mathcal{D}}{}^{(L-1)} = \pi_b$).

Choosing w_π to be $w_{\pi_{\mathcal{D}}{}^{(n)}}$ for $n < L-1$ also does *not* result in $v(\pi_e)$ being returned in this example. This observation is surprising because even though we know that the true $\Pr(H = h|\pi_b) = \prod_{t=0}^{L-1} \pi_b(a_t|s_t)$, it does not follow that the estimated probability of a trajectory is equal to the product of the estimated Markovian action probabilities, i.e., that $\frac{c(h)}{m} = \prod_{t=0}^{L-1} \pi_{\mathcal{D}}{}^{(0)}(a_t|s_t)$. With a finite number of samples, the data may have higher likelihood under a non-Markovian behavior policy – possibly even a policy that conditions on all past states and actions. Thus, to fully correct for sampling error, we must importance sample with an estimated non-Markovian behavior policy. However, $w_{\pi_{\mathcal{D}}{}^{(n)}}$ with $n < L-1$ still provides a better sampling error correction than w_{π_b} since any $\pi_{\mathcal{D}}{}^{(n)}$ will reflect the statistics of $\mathcal{D}$ while π_b does

[12]This statement follows from the importance sampling identity: $\mathbf{E}[\frac{\Pr(H|\pi_e)}{\Pr(H|\pi)} g(h)|H \sim \pi] = \mathbf{E}[g(H)|H \sim \pi_e] = v(\pi_e)$ and the fact that we have assumed a deterministic environment.

not. This statement is supported by our empirical results comparing RIS(0) to OIS and a theoretical result we present in the following section that states that RIS(n) has lower asymptotic variance than OIS for all n.

Before concluding this section, we discuss two limitations of the presented example – these limitations are *not* present in our theoretical or empirical results. First, the example lacks stochasticity in the rewards and transitions. In stochastic environments, sampling error arises from sampling states, actions, and rewards while in deterministic environments, sampling error only arises from sampling actions. Neither RIS nor OIS can correct for state and reward sampling error since such a correction requires knowledge of what the true state and reward frequencies are and these quantities are typically unknown in the MDP policy value estimation setting.

Second, we assumed that $\mathcal{D}$ contains at least one of each trajectory possible under π_b. If a trajectory is absent from $\mathcal{D}$ then RIS($L-1$) has non-zero bias. Theoretical analysis of this bias for both RIS($L-1$) and other RIS variants is an open question for future analysis.

5.2.2 Correcting Sampling Error in Continuous Action Spaces

In the previous subsection, we presented an example showing how RIS corrects for sampling error in $\mathcal{D}$ in deterministic and finite MDPs. Most of this discussion assumed that the state and action spaces of the MDP were finite. Here, we discuss sampling error in continuous action spaces. The primary purpose of this discussion is intuition and we limit discussion to a setting that can be easily visualized. We consider a deterministic MDP with scalar, real-valued actions, reward $R : \mathcal{A} \to \mathbb{R}$, and $L = 1$.

We assume the support of π_b and π_e is bounded and for simplicity assume the

support to be $[0, 1]$. Policy value estimation is equivalent to estimating the integral:

$$v(\pi_e) = \int_0^1 r(a)\pi_e(a)da \tag{5.2}$$

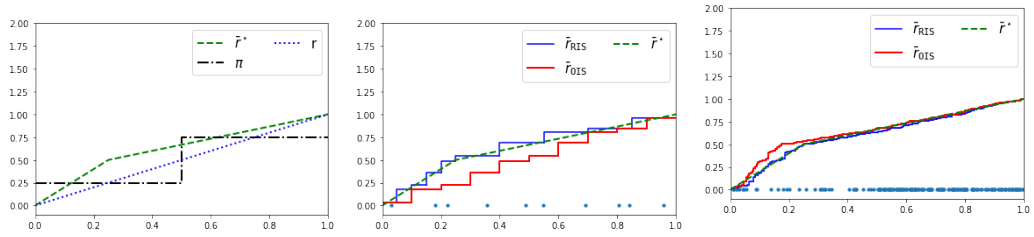
and the ordinary importance sampling estimate of this quantity with m samples from π_b is:

$$\text{OIS}(\pi_e, \mathcal{D}) = \frac{1}{m}\sum_{i=1}^{m}\frac{\pi_e(A_i)}{\pi_b(A_i)}r(A_i). \tag{5.3}$$

Even though the OIS estimate is a sum over a finite number of samples, we show it is exactly equal to an integral over a particular piece-wise function. We assume (w.l.o.g) that the A_i's are in non-decreasing order, ($A_0 <= A_i <= A_m$). Imagine that we place the $r(A_i)$ values uniformly across the interval $[0, 1]$ so that they divide the range $[0, 1]$ into m equal bins. In other words, we maintain the relative ordering of the action samples but ignore the spatial relationship between samples. We now define piece-wise constant function $\bar{r}_{\text{OIS}}$ where $\bar{r}_{\text{OIS}}(a) = r(A_i)$ if a is in the i^{th} bin. The ordinary importance sampling estimate is exactly equal to the integral $\int_0^1 \bar{r}_{\text{OIS}}(a)da$.

It would be reasonable to assume that $\bar{r}_{\text{OIS}}(a)$ is approximating $r(a)\pi_e(a)$ since the ordinary importance sampling estimate (5.3) is approximating (5.2), i.e., $\lim_{m\to\infty} \bar{r}_{\text{OIS}}(a) = r(a)\pi_e(a)$. In reality, $\bar{r}_{\text{OIS}}$ approaches a *stretched* version of r where areas with high density under π_e are stretched and areas with low density are contracted. We call this stretched version of r, $\bar{r}^\star$. The integral of $\int_0^1 \bar{r}^\star(a)da$ is $v(\pi_e)$.

Figure 5.2a gives a visualization of an example $\bar{r}^\star$ using on-policy Monte Carlo sampling from an example π_e and linear r. In contrast to the true $\bar{r}^\star$, the OIS approximation to $\bar{r}$, $\bar{r}_{\text{OIS}}$ stretches ranges of r according to the number of samples in that range: ranges with many samples are stretched and ranges without many samples are contracted. As the sample size grows, any range of r will be stretched



(a) Policy and Reward (b) 10 Sample Approximation (c) 200 Sample Approximation

Figure 5.2: Policy values estimation in a continuous armed bandit task. Figure 5.2a shows a reward function, r, and the pdf of a policy, π, with support on the range $[0, 1]$. With probability 0.25, π selects an action less than 0.5 with uniform probability; otherwise π selects an action greater than 0.5. The reward is equal to the action chosen. All figures show $\bar{r}^\star$: a version of r that is stretched according to the density of π; since the range $[0.5, 1]$ has probability 0.75, r on this interval is stretched over $[0.25, 1]$. Figure 5.2b and 5.2c show $\bar{r}^\star$ and the piece-wise $\bar{r}_{\text{OIS}}$ and $\bar{r}_{\text{RIS}}$ approximations to $\bar{r}^\star$ after 10 and 200 samples respectively.

in proportion to the probability of getting a sample in that range. For example, if the probability of drawing a sample from $[a, b]$ is 0.5 then $\bar{r}^\star$ stretches r on $[a, b]$ to cover half the range $[0, 1]$. Figure 5.2 visualizes $\bar{r}_{\text{OIS}}$ the OIS approximation to $\bar{r}^\star$ for sample sizes of 10 and 200.

In this analysis, sampling error corresponds to over-stretching or under-stretching r in any given range. The limitation of ordinary importance sampling can then be expressed as follows: given π_e, we know the correct amount of stretching for any range and yet OIS ignores this information and stretches based on the empirical proportion of samples in a particular range. On the other hand, RIS first divides by the empirical pdf (approximately undoing the stretching from sampling) and then multiplies by the true pdf to stretch r a more accurate amount. Figure 5.2 also visualizes the $\bar{r}_{\text{RIS}}$ approximation to $\bar{r}^\star$ for sample sizes of 10 and 200. In this figure, we can see that $\bar{r}_{\text{RIS}}$ is a closer approximation to $\bar{r}^\star$ than $\bar{r}_{\text{OIS}}$ for both sample sizes. In both instances, the mean squared error of the RIS estimate is less than that of the OIS estimate.

Since r may be unknown until sampled, we will still have non-zero MSE. However the basic OIS estimate has error due to *both* sampling error and unknown r values. RIS has error only due to the unknown r values for actions that remain unsampled.

5.3 Theoretical Analysis

In this section we summarize new theoretical results (full proofs appear in the appendices) as well as a connection to prior work from the multi-armed bandit literature:

Proposition 5.1. *$\forall n$, assuming the estimate of ${\pi_{\mathcal{D}}}^{(n)}$ is consistent,* $\text{RIS}(n)$ *is a consistent estimator of $v(\pi_e)$:* $\text{RIS}(n)(\pi_e, \mathcal{D}) \xrightarrow{a.s.} v(\pi_e)$.

Proof. See Appendix D.1 for a full proof. □

Corollary 5.1. *Let $\Pi^n_{\boldsymbol{\theta}}$ be a class of twice differentiable policies, $\pi_{\boldsymbol{\theta}}(\cdot|s_{t-n}, a_{t-n}, \ldots, s_t)$. If $\exists \tilde{\boldsymbol{\theta}}$ such that $\pi_{\tilde{\boldsymbol{\theta}}} \in \Pi^n_{\boldsymbol{\theta}}$ and $\pi_{\tilde{\boldsymbol{\theta}}} = \pi_b$ then*

$$\text{Var}_{\mathtt{A}}(\text{RIS}(n)(\pi_e, \mathcal{D})) \leq \text{Var}_{\mathtt{A}}(\text{OIS}(\pi_e, \mathcal{D}, \pi_b))$$

where $\text{Var}_{\mathtt{A}}$ *denotes the asymptotic variance.*

Proof. This result is a corollary to Theorem 1 of Henmi et al. (2007) for general Monte Carlo integration (see Appendix D.2 for a full proof). □

We highlight that the derivation of the result by Henmi et al. (2007) includes some $o(n)$ and $o_p(1)$ terms that may be large for small sample sizes; the lower variance is asymptotic and we leave analysis of the finite-sample variance of RIS to future work.

RIS is related to the REG estimator studied by Li et al. (2015). For finite MDPs, Li et al. (2015) introduced the *regression* (REG) estimator and show it

has asymptotic lower minimax MSE than OIS provided the estimator has full knowledge of the environment's transition probabilities. With this knowledge REG can correct for sampling error in both the actions and state transitions. RIS(L-1) is an approximation to REG that only corrects for sampling error in the actions. The derivation of the connection between REG and RIS($L-1$) is given in Appendix D.3.

Finally, we also note that prior theoretical analysis of importance sampling with an estimated behavior policy has made the assumption that $\pi_{\mathcal{D}}$ is estimated with different data than the data used for the policy value estimate (Dudík et al., 2011; Farajtabar et al., 2018). This assumption simplifies the theoretical analysis but makes it inapplicable to regression importance sampling as RIS estimates $\pi_{\mathcal{D}}$ with $\mathcal{D}$.

5.4 RIS with Function Approximation

The example in Section 5.2.1 presented RIS with count-based estimation of $\pi_{\mathcal{D}}$. In many practical settings, count-based estimation of $\pi_{\mathcal{D}}$ is intractable and we must rely on function approximation. For example, in our final experiments we learn $\pi_{\mathcal{D}}$ as a Gaussian distribution over actions with the mean given by a neural network. Two practical concerns arise when using function approximation for RIS: avoiding over-fitting and selecting the class of function approximator.

RIS uses all of the available data to both estimate $\pi_{\mathcal{D}}$ and compute the off-policy estimate of $v(\pi_e)$. Unfortunately, the RIS estimate may suffer from high variance if the function approximator is too expressive and $\pi_{\mathcal{D}}$ is over-fit to our data. Additionally, if the policy class of π_b is unknown, it may be unclear what is the right function approximation representation for $\pi_{\mathcal{D}}$. A practical solution is to use a validation set – distinct from $\mathcal{D}$ – to select an appropriate policy class and appropriate regularization criteria for RIS. This solution is a small departure from the previous definition of RIS as selecting $\pi_{\mathcal{D}}$ to maximize the log likelihood on $\mathcal{D}$.

Rather, we select $\pi_{\mathcal{D}}$ to maximize the log likelihood on $\mathcal{D}$ while avoiding over-fitting. This approach represents a trade-off between robust empirical performance and a potentially stronger sample correction by further maximizing log likelihood on the data used for computing the RIS estimate.

5.5 Empirical Study

We now present an empirical study of the RIS estimator across several policy value estimation tasks. Our experiments are designed to answer the following questions:

1. What is the empirical effect of replacing OIS weights with RIS weights in sequential decision making tasks?
2. How important is using $\mathcal{D}$ to both estimate the behavior policy and compute the importance sampling estimate?
3. How does the choice of n affect the MSE of RIS(n)?

With non-linear function approximation, our results suggest that the common supervised learning approach of model selection using hold-out validation loss may be sub-optimal for the regression importance sampling estimator. Thus, we also investigate the question:

4. Does minimizing hold-out validation loss set yield the minimal MSE regression importance sampling estimator when estimating $\pi_{\mathcal{D}}$ with gradient descent and neural network function approximation?

5.5.1 Empirical Set-up

We run policy value estimation experiments in several domains. We provide a short description of each domain here; a complete description and additional experimental details are given in Appendix F.3.

- **Grid World:** This domain is a 4×4 Grid World used in prior off-policy value estimation research (Thomas, 2015; Thomas and Brunskill, 2016a). RIS uses count-based estimation of π_b. This domain allows us to study RIS separately from questions of function approximation.

- **Single Path:** See Figure 5.3 for a description. This domain is small enough to make implementations of RIS($L-1$) and the REG method from Li et al. (2015) tractable. All RIS methods use count-based estimation of π_b.

- **Linear Dynamical System:** This domain is a point-mass agent moving towards a goal in a two dimensional world by setting x and y acceleration. Policies are linear in a second order polynomial transform of the state features. We estimate $\pi_{\mathcal{D}}$ with ordinary least squares.

- **Simulated Robotics:** We also use two continuous control tasks from the OpenAI gym: Hopper and HalfCheetah.[13] In each task, we use neural network policies with 2 layers of 64 tanh hidden units each for π_e and π_b.

In all domains we run repeated trials of each experiment. Except for the Simulated Robotics domains, a trial consists of evaluating the squared error of different estimators over an increasing data set. The average squared error over multiple trials is an unbiased estimate of the mean squared error of each method. In the Simulated Robotics domain, a trial consists of collecting a single batch of 400 trajectories and evaluating the squared error of different estimators on this batch.

5.5.2 Empirical Results

We now present our empirical results. Except where specified otherwise, RIS refers to RIS(0).

[13] For these tasks we use the Roboschool versions: https://github.com/openai/roboschool

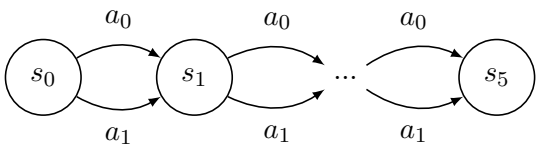


Figure 5.3: The Single Path MDP. This environment has 5 states, 2 actions, and $L = 5$. The agent begins in state 0 and both actions either take the agent from state n to state $n + 1$ or cause the agent to remain in state n. **Not shown:** If the agent takes action a_1 it remains in its current state with probability 0.5.

5.5.2.1 Finite MDP Policy Value Estimation

Our first experiment compares several importance sampling variants implemented with both RIS weights and OIS weights in the Grid World domain. Specifically, we use the basic IS estimator, the *weighted* IS estimator (Precup et al., 2000), *per-decision* IS, the *doubly robust* (Jiang and Li, 2016), and the *weighted doubly robust* estimator (Thomas and Brunskill, 2016a).

Figure 5.4a shows the MSE of the evaluated methods averaged over 100 trials. The results show that, for this domain, using RIS weights lowers MSE for all tested IS variants relative to OIS weights.

We also evaluate alternative data sources for estimating $\pi_{\mathcal{D}}$ in order to establish the importance of using $\mathcal{D}$ to both estimate $\pi_{\mathcal{D}}$ and compute the value estimate. Specifically, we consider:

1. **Independent Estimate**: In addition to $\mathcal{D}$, this method has access to an additional set, $\mathcal{D}_{\texttt{train}}$. The behavior policy is estimated with $\mathcal{D}_{\texttt{train}}$ and the policy value estimate is computed with $\mathcal{D}$. Since state-action pairs in $\mathcal{D}$ may be absent from $\mathcal{D}_{\texttt{train}}$ we use Laplace smoothing to ensure that the importance weights never have a zero in the denominator.

2. **Extra-data Estimate**: This baseline is the same as **Independent Estimate** except it uses both $\mathcal{D}_{\texttt{train}}$ and $\mathcal{D}$ to estimate π_b. Only $\mathcal{D}$ is used to compute the policy value estimate.

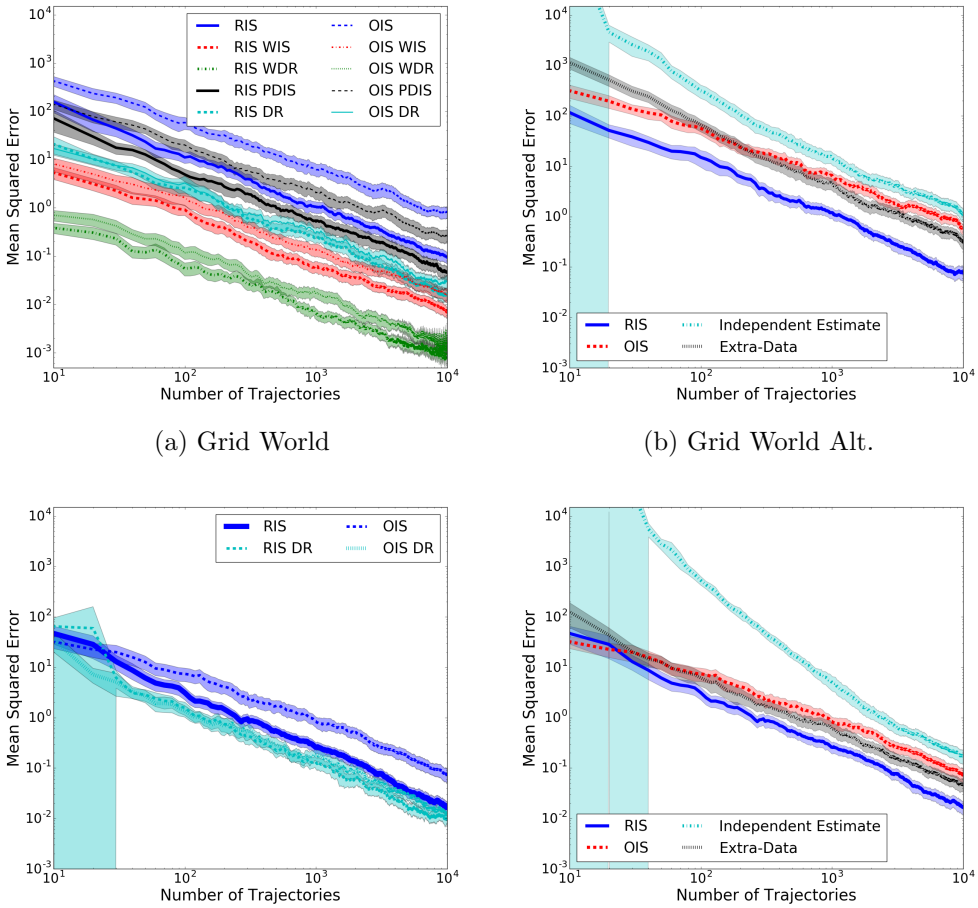
Figure 5.4b shows that these alternative data sources for estimating π_b decrease accuracy compared to RIS and OIS. **Independent Estimate** has high MSE when the sample size is small but its MSE approaches that of OIS as the sample size grows. We understand this result as showing that this baseline cannot correct for sampling error in the off-policy data since the behavior policy estimate is unrelated to the data used in computing the value estimate. **Extra-data Estimate** initially has high MSE but its MSE decreases faster than that of OIS. Since this baseline estimates π_b with data that includes $\mathcal{D}$, it can partially correct for sampling error – though the extra data harms its ability to do so. Only estimating $\pi_{\mathcal{D}}$ with $\mathcal{D}$ and $\mathcal{D}$ alone lowers MSE over OIS for all sample sizes.

We also repeat these experiments for the on-policy setting and present results in Figure 5.4c and Figure 5.4d. We observe similar trends as in the off-policy experiments suggesting that RIS can lower variance in Monte Carlo sampling methods even when OIS weights are otherwise unnecessary.

5.5.2.2 RIS(n)

In the Grid World domain it is difficult to observe the performance of RIS(n) for various n because of the long horizon: smaller n perform similarly and larger n scale poorly with L. To see the effects of different n more clearly, we use the Single Path domain. Figure 5.5 gives the mean squared error for OIS, RIS, and the REG estimator of Li et al. (2015) that has full access to the environment's transition probabilities. For RIS, we use $n = 0, 3, 4$ and each method is run for 200 trials.

Figure 5.5 shows that higher values of n and REG tend to give inaccurate estimates when the sample size is small. However, as data increases, these methods give increasingly accurate value estimates. In particular, REG and RIS(4) produce estimates with MSE more than 20 orders of magnitude below that of RIS(3) (Figure 5.5 is cut off at the bottom for clarity of the rest of the results). REG eventually



(a) Grid World

(b) Grid World Alt.

(c) Grid World On-Policy

(d) Grid World On-Policy Alt.

Figure 5.4: Grid World policy value estimation results. In all subfigures, the x-axis is the number of trajectories collected and the y-axis is mean squared error. Axes are log-scaled. The shaded region represents a 95% confidence interval. (a) Grid World Off-policy Value Estimation: The main point of comparison is the RIS variant of each method to the OIS variant of each method. (b) Grid World $\pi_{\mathcal{D}}$ Estimation Alternatives: This plot compares RIS and OIS to two methods that replace the true behavior policy with estimates from data sources other than $\mathcal{D}$. Subfigures (c) and (d) repeat experiments (a) and (b) with the behavior policy from (c) and (d) as the evaluation policy.

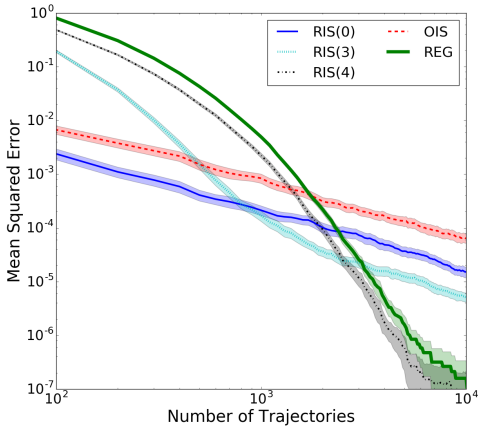


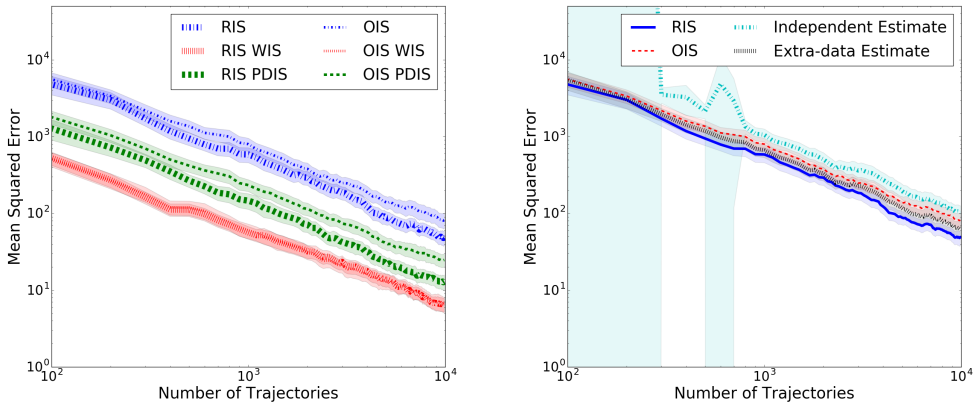
Figure 5.5: Off-policy value estimation in the Single Path MDP for various n. The x-axis is the number of trajectories in $\mathcal{D}$ and the y-axis is MSE. Both axes are log-scaled. The curves for REG and RIS(4) have been cut-off to more clearly show all methods. These methods converge to an MSE value of approximately 1×10^{-31}

passes the performance of RIS(4) since its knowledge of the transition probabilities allows it to eliminate sampling error in both the actions and the environment. In the low-to-medium data regime, only RIS(0) outperforms OIS. However, as data increases, the MSE of all RIS methods and REG decreases faster than that of OIS. The similar performance of $RIS(L-1)$ and REG supports the connection between these methods that we discussed in Section 5.2.

5.5.2.3 RIS with Linear Function Approximation

Our next set of experiments consider continuous state and action spaces in the Linear Dynamical System domain. RIS represents $\pi_{\mathcal{D}}$ as a Gaussian policy with mean given as a linear function of the state features. Similar to in Grid World, we compare three variants of IS, each implemented with RIS and OIS weights: the ordinary IS estimator, weighted IS (WIS), and per-decison IS (PDIS). Each method is averaged over 200 trials and results are shown in Figure 5.6a.

We see that RIS weights lower the MSE of both IS and PDIS, while both



(a) LDS

(b) LDS Alt. Weights

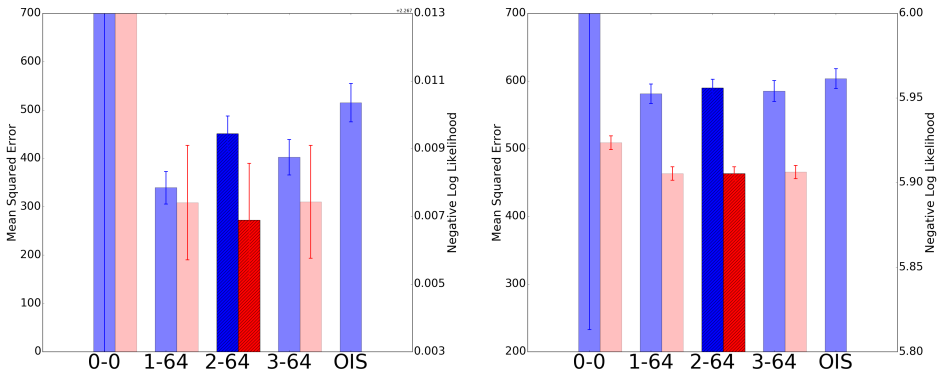
Figure 5.6: Linear dynamical system results. Figure 5.6a shows the mean squared error (MSE) for three IS variants with and without RIS weights. Figure 5.6b shows the MSE for different methods of estimating the behavior policy compared to RIS and OIS. Axes and scaling are the same as in Figure 5.4a.

WIS variants have similar MSE. This result suggests that the MSE improvement from using RIS weights depends, at least partially, on the variant of IS being used.

Similar to Grid World, we also consider estimating $\pi_{\mathcal{D}}$ with either an independent data-set or with extra data and see a similar ordering of methods. **Independent Estimate** gives high variance estimates for small sample sizes but then approaches OIS as the sample size grows. **Extra-Data Estimate** corrects for some sampling error and has lower MSE than OIS. RIS lowers MSE compared to all baselines.

5.5.2.4 RIS with Neural Networks

Our remaining experiments use the Hopper and HalfCheetah domains. RIS represents $\pi_{\mathcal{D}}$ as a neural network that maps the state to the mean of a Gaussian distribution over actions. The standard deviation of the Gaussian is given by state-independent parameters. In these experiments, we sample a single batch of 400 trajectories per trial and compare the MSE of RIS and OIS on this batch.



(a) Hopper (b) HalfCheetah

Figure 5.7: Figures 5.7a and 5.7b compare different neural network architectures (specified as #-layers-#-units) for regression importance sampling on the Hopper and HalfCheetah domain. The darker, blue bars give the MSE for each architecture and OIS. Lighter, red bars give the negative log likelihood of a hold-out data set. Our main point of comparison is the MSE of the architecture with the lowest hold-out negative log likelihood (given by the darker pair of bars) compared to the MSE of OIS.

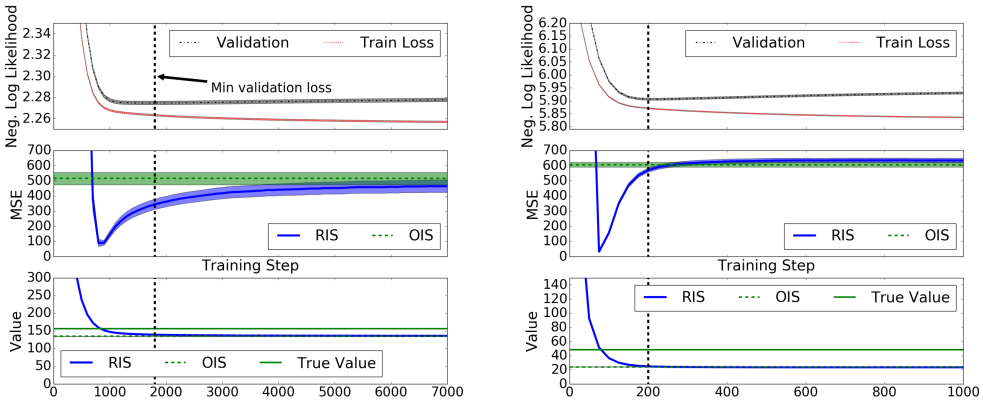
Figure 5.7 compares the MSE of RIS for different neural network architectures. Our main point of comparison is RIS using the architecture that achieves the lowest validation error during training (the darker bars in Figure 5.7). Under this comparison, the MSE of RIS with a two-hidden-layer network is lower than that of OIS in both Hopper and HalfCheetah, though, in HalfCheetah, the difference is statistically insignificant. We also observe that the policy class with the best validation error does *not* always give the lowest MSE (e.g., in Hopper, the two hidden layer network gives the lowest validation loss but the network with a single layer of hidden units has $\approx 25\%$ less MSE than the two hidden layer network). This last observation motivates our final experiment.

5.5.2.5 RIS Model Selection

Our final experiment aims to better understand how hold-out validation error relates to the MSE of the RIS estimator when using gradient descent to estimate neural network approximations of $\pi_{\mathcal{D}}$. This experiment duplicates our previous experiment, except every 25 steps of gradient descent we stop optimizing $\pi_{\mathcal{D}}$ and compute the RIS estimate with the current $\pi_{\mathcal{D}}$ and its MSE. We also compute the training and hold-out validation negative log-likelihood. Plotting these values gives a picture of how the MSE of RIS changes as our estimate of $\pi_{\mathcal{D}}$ changes. Figure 5.8 shows these plots for the Hopper and HalfCheetah domains.

We see that the policy with minimal MSE and the policy that minimizes validation loss are misaligned. If training is stopped when the validation loss is minimized, the MSE of RIS is lower than that of OIS (the intersection of the RIS curve and the vertical dashed line in Figure 5.8. However, the $\pi_{\mathcal{D}}$ that minimizes the validation loss curve is *not* identical to the $\pi_{\mathcal{D}}$ that minimizes MSE.

To understand this result, we also plot the mean RIS estimate throughout behavior policy learning (bottom of Figure 5.8). We can see that at the beginning of training, RIS tends to *over-estimate* $v(\pi_e)$ because the probabilities given by $\pi_{\mathcal{D}}$ to the observed data will be small (and thus the RIS weights are large). As the likelihood of $\mathcal{D}$ under $\pi_{\mathcal{D}}$ increases (negative log likelihood decreases), the RIS weights become smaller and the estimates tend to *under-estimate* $v(\pi_e)$. The implication of these observations, for RIS, is that during behavior policy estimation the RIS estimate will likely have zero MSE at some point. Thus, there may be an early stopping criterion – besides minimal validation loss – that would lead to lower MSE with RIS, however, to date we have not found one. Note that OIS also tends to under-estimate policy value in MDPs as has been previously analyzed by Doroudi et al. (2017).



(a) Hopper (b) HalfCheetah

Figure 5.8: Mean squared error and estimate of the importance sampling estimator during training of $\pi_{\mathcal{D}}$. The x-axis is the number of gradient descent steps. The top plot shows the training and validation loss curves. The y-axis of the top plot is the average negative log-likelihood. The y-axis of the middle plot is mean squared error (MSE). The y-axis of the bottom plot is the value of the estimate. MSE is minimized close to, but slightly before, the point where the validation and training loss curves indicate that overfitting is beginning. This point corresponds to where the RIS estimate transitions from over-estimating to under-estimating the policy value.

5.6 Summary

This chapter has described Contribution 3 of this dissertation: a family of importance sampling methods for policy value estimation, called regression importance sampling methods, that apply importance sampling after first estimating the behavior policy that generated the data. Notably, RIS estimates the behavior policy from the same set of data that is also used for the IS estimate. Computing the behavior policy estimate and IS estimate from the same set of data allows RIS to correct for the sampling error inherent to importance sampling with the true behavior policy. We showed that these methods have lower asymptotic variance than ordinary importance sampling and are consistent. We evaluated RIS across several policy value estimation

tasks and show that it lowers MSE compared to ordinary importance sampling – that uses the true behavior policy – in several off-policy policy value estimation tasks. Finally, we showed that, as the sample size grows, it can be beneficial to ignore knowledge that the true behavior policy is Markovian even if that knowledge is available.

This chapter introduced regression importance sampling for lower variance weighting when correcting for distribution shift in off-policy data for policy value estimation. In the next chapter we will show how the same technique can be used for more accurate policy improvement with batch policy gradient reinforcement learning.

Chapter 6

Weighting Data for Policy Improvement

The previous chapter showed how proper weighting of a reinforcement learning agent's experienced data can result in more accurate off-policy policy value estimation. In particular, by correcting for *sampling error*, we are able to lower the variance of importance sampling policy value estimation with less off-policy data. In this chapter, we translate our contribution to policy value estimation into the policy improvement setting by introducing an algorithm that reduces sampling error for batch policy gradient algorithms.[14]

We consider batch policy gradient reinforcement learning, as introduced in Section 2.3. Batch policy gradient algorithm implementations commonly rely on Monte Carlo sampling to approximate the policy gradient. Such methods may suffer from sampling error when only allowed access to a finite amount of environment experience. In this chapter, we introduce a method that corrects sampling error in batch policy gradient learning in the same way that regression importance sampling

[14]This chapter contains work that was previously published at AAMAS 2019 (Hanna and Stone, 2019).

(Chapter 5) corrects sampling error in off-policy policy value estimation. The so-called *sampling error corrected* policy gradient algorithm leads to reinforcement learning agents that can obtain higher expected reward with less data compared to agents learning with a batch Monte Carlo policy gradient algorithm. This new method is Contribution 4 of this dissertation.

Though corrections for sampling error have appeared in the policy gradient literature before (Sutton et al., 2000b; Asadi et al., 2017; Ciosek and Whiteson, 2018), these approaches require learning the action-value function, q^π. One of the main reasons that practitioners may prefer policy gradient algorithms to value-based RL methods such as Q-learning (Watkins, 1989) or actor-critic algorithms (Sutton, 1984) is that the policy may be a simpler function to approximate than the action-value function (Sutton and Barto, 1998). Contribution 4 of this dissertation provides a way to correct sampling error in batch policy gradient learning that does *not* necessarily require an explicit action-value function. In Section 9.2.2 we discuss related approaches to correcting or eliminating sampling error, however these approaches require an approximation of the action-value function in addition to the policy.

6.1 Sampling Error in Batch Policy Gradient Learning

The batch Monte Carlo policy gradient estimator introduced in Section 2.3, $g_{\texttt{mc}}$, is a common approach to estimating the gradient in policy gradient learning. In this section, we discuss approximation error in $g_{\texttt{mc}}$ and present the view that – for a fixed dataset – $g_{\texttt{mc}}$ is the gradient estimated under the wrong distribution of states and actions. This view echos the arguments put forward in Chapter 5 that ordinary importance sampling is flawed for finite amounts of data. Here, we present these arguments in the batch policy gradient reinforcement learning setting.

Recall from Section 2.3 the expression proportional to the policy gradient:

$$\frac{\partial}{\partial \boldsymbol{\theta}} v(\pi_{\boldsymbol{\theta}}) \propto \mathbf{E}\left[q^{\pi_{\boldsymbol{\theta}}}(S, A, \cdot) \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S) \middle| S \sim d_{\pi_{\boldsymbol{\theta}}}, A \sim \pi_{\boldsymbol{\theta}} \right] \tag{6.1}$$

where $d_{\pi_{\boldsymbol{\theta}}}$ is the distribution of states observed when running $\pi_{\boldsymbol{\theta}}$ in $\mathcal{M}$. Let $\mathcal{T} = \{(S_j, A_j)\}_{j=1}^m$ be a set of m observed state-action pairs that occurred while following the current policy, $\pi_{\boldsymbol{\theta}}$, in the MDP of interest. Let $\hat{q}^{\pi_{\boldsymbol{\theta}}}(s, a, \cdot)$ be an estimate of $q^{\pi_{\boldsymbol{\theta}}}(s, a, \cdot)$ that can be obtained from either function approximation or simply from the sum of rewards following the occurrence of (s, a).[15] Given an instance of $\mathcal{T}$, the batch Monte Carlo policy gradient estimator is defined as:

$$g_{\texttt{mc}}(\mathcal{T}) \coloneqq \frac{1}{m} \sum_{j=1}^{m} \hat{q}^{\pi_{\boldsymbol{\theta}}}(S_j, A_j) \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_j|S_j). \tag{6.2}$$

We first note that, for a finite number of sampled states and actions, $g_{\texttt{mc}}$ will likely have error unless $\mathcal{T}$ happens to contain each pair (s, a) at its long-run expected frequency, $d_{\pi_{\boldsymbol{\theta}}}(s)\pi(a|s)$, and $\widehat{q}^{\pi_{\boldsymbol{\theta}}}(s, a, \cdot) = q^{\pi_{\boldsymbol{\theta}}}(s, a, \cdot)$ for all s, a. In this section we discuss error in gradient estimation due to sampling in $d_{\pi_{\boldsymbol{\theta}}}$ and π and ignore error due to differences between $\widehat{q}^{\pi_{\boldsymbol{\theta}}}(s, a, \cdot)$ and $q^{\pi_{\boldsymbol{\theta}}}(s, a, \cdot)$.

Let $d_{\mathcal{T}}(s)$ be the proportion of times that s occurs in $\mathcal{T}$ and $\pi_{\mathcal{T}}(a|s)$ be the proportion of times that action a occurred in state s in $\mathcal{T}$. Formally, let $m(s)$ be the number of times that we observe state s in $\mathcal{T}_i$ and let $m(s, a)$ be the number of times that we observe action a in state s. We define $d_{\mathcal{T}}(s) = \frac{m(s)}{m}$ and $\pi_{\mathcal{T}}(a|s) = \frac{m(s,a)}{m(s)}$. Let $\tilde{q}^{\pi}(s, a)$ be the mean value of $\hat{q}^{\pi}(s, a, \cdot)$ in $\mathcal{T}$ Finally, we define the function $\bar{q}^{\pi} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as the mean value of $\hat{q}^{\pi}(s, a, \cdot)$ in $\mathcal{T}$. If (s, a) is missing from $\mathcal{T}$ then $\bar{q}^{\pi}(s, a) \coloneqq 0$. Given these definitions, the batch Monte Carlo policy gradient

[15] We use the latter in our empirical studies.

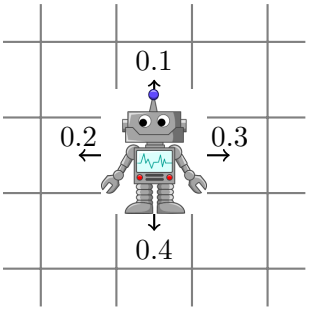
estimator can be re-written as:

$$
\begin{aligned}
g_{\texttt{mc}}(\mathcal{T}) &= \frac{1}{m}\sum_{j=1}^{m} \widehat{q}^{\pi_{\boldsymbol{\theta}}}(s_j, a_j, \cdot)\frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_j|s_j) \\
&= \frac{1}{m}\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} m(s,a)\bar{q}^{\pi_{\boldsymbol{\theta}}}(s,a)\frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) \\
&= \sum_{s\in\mathcal{S}} d_{\mathcal{T}}(s)\sum_{a\in\mathcal{A}} \pi_{\mathcal{T}}(a|s)\bar{q}^{\pi_{\boldsymbol{\theta}}}(s,a)\frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) \\
&= \mathbf{E}\left[\bar{q}^{\pi_{\boldsymbol{\theta}}}(S,A)\frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A|S)\middle| S \sim d_{\mathcal{T}}, A \sim \pi_{\mathcal{T}}\right].
\end{aligned}
$$

Notably, the sample average in (6.2) has been replaced with an exact expectation over actions as in (6.1). However, the expectation is taken over the action distribution $\pi_{\mathcal{T}}$ and *not* $\pi_{\boldsymbol{\theta}}$. This expression suggests that sampling error in the Monte Carlo approximation can be viewed as evaluating the gradient under the wrong distribution. Figure 6.1 expresses how sampling error relates to weighting each $a \in \mathcal{A}$ in a Monte Carlo gradient estimate for a fixed state.

This section has argued that *for a fixed set of data*, the batch Monte Carlo policy gradient estimator will be equal to an exact expectation taken over the wrong distribution of states and actions. The correct state distribution is unknown. However, we do know the correct action distribution ($\pi_{\boldsymbol{\theta}}$) and thus can correct the inaccurate weighting. In the next section we introduce an algorithm that uses importance sampling to apply this correction.

Before concluding this section, we note that $g_{\texttt{mc}}$ is an unbiased estimator of $\frac{\partial}{\partial \boldsymbol{\theta}} v(\pi_{\boldsymbol{\theta}})$. That is, if we were to repeatedly sample batches of data and estimate the gradient, the gradient estimates would be correct in expectation. However, once a single batch of data has been collected, we might ask, "can we correct for the sampling inaccuracy observed in this fixed sample?"



Action	$\pi_{\boldsymbol{\theta}}$	$\pi_{\mathcal{T}}$	$g_{\texttt{mc}}$ weight	$g_{\texttt{sec}}$ weight
Up	0.1	0.15	0.15	0.1
Right	0.3	0.35	0.35	0.3
Down	0.4	0.3	0.3	0.4
Left	0.2	0.2	0.2	0.2

Figure 6.1: Sampling error in a fixed state s of a Grid World environment. Each action a is sampled with probability $\pi_{\boldsymbol{\theta}}(a|s)$ and is observed in the proportion given by $\pi_{\mathcal{T}}(a|s)$. Monte Carlo weighting gives each return $q^{\pi_{\boldsymbol{\theta}}}(a|s)$ the weight $\pi_{\mathcal{T}}(a|s)$ while our new sampling error corrected weighting gives each return $q^{\pi_{\boldsymbol{\theta}}}(a|s)$ the weight $\pi_{\mathcal{T}}(a|s)\frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\mathcal{T}}(a|s)} = \pi_{\boldsymbol{\theta}}(a|s)$. Thus SEC weights each advantage by the correct amount while the Monte Carlo estimator will have error unless the empirical proportion of sampled actions, $\pi_{\mathcal{T}}$, is equal to the expected proportion, $\pi_{\boldsymbol{\theta}}$ for all actions.

6.2 The Sampling Error Corrected Policy Gradient Estimator

The previous section presented the view that sampling error in Monte Carlo approximations can be viewed as *covariate shift* – we are interested in an expectation under $d_{\pi_{\boldsymbol{\theta}}}$ and $\pi_{\boldsymbol{\theta}}$ but instead we have an expectation under $d_{\mathcal{T}}$ and $\pi_{\mathcal{T}}$. Viewing the sampling error as covariate shift suggests a simple solution: use importance sampling to correct for the distribution shift.

In this setting, we will treat the empirical distribution of actions conditioned on state, $\pi_{\mathcal{T}}$ as the behavior policy π_b and then use importance sampling to correct for the shift between the empirical and desired distribution. We call this approach the *sampling error corrected* (SEC) policy gradient estimator.

In practice, using the true $\pi_{\mathcal{T}}$ may introduce high bias into gradient estimates, particularly in continuous state and action spaces. This bias arises because using $\pi_{\mathcal{T}}$ can be shown to be equivalent to assuming that $\widehat{q}^{\pi_{\boldsymbol{\theta}}}(s,a)$ is zero for all unobserved actions.[16] Instead, let π_ϕ be a parametric estimate of the policy that generated our data.[17] The SEC estimator estimates ϕ so that π_ϕ is the maximum likelihood policy that generated our data:

$$\phi = \underset{\phi}{\operatorname{argmax}} \sum_{j=1}^{m} \log \pi_\phi(A_j|S_j). \tag{6.3}$$

Importantly, SEC estimates ϕ with the same m samples that will be used to estimate the policy gradient. If ϕ is estimated with a different set of samples then π_ϕ will contain no information for correcting sampling error – our experiments confirm this observation. For most RL benchmarks, (6.3) can be formulated as a supervised learning problem.

Given π_ϕ, SEC re-weights each $\widehat{q}^{\pi_{\boldsymbol{\theta}}}(s_i,a_i,\cdot)\frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_i|s_i)$ by the ratio of the true likelihood $\pi_{\boldsymbol{\theta}}$ to the estimated empirical likelihood π_ϕ:

$$g_{\texttt{sec}}(\mathcal{T}_i) = \frac{1}{m}\sum_{j=1}^{m} \frac{\pi_{\boldsymbol{\theta}}(A_j|S_j)}{\pi_\phi(A_j|S_j)} \widehat{q}^{\pi_{\boldsymbol{\theta}}}(S_j,A_j,\cdot)\frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_j|S_j). \tag{6.4}$$

Intuitively, when an action is sampled more often than its expected proportion, $g_{\texttt{sec}}$ down-weights the gradient estimate following that action. Similarly, when an action is sampled less often than its expected proportion, $g_{\texttt{sec}}$ up-weights the gradient estimate following that action. As we will discuss in the next section, if π_ϕ is close to $\pi_{\mathcal{T}}$ then this sampling correction can eliminate variance in the action selection. Full details of this approach are given in Algorithm 4.

Importance sampling in reinforcement learning is typically applied for *off-*

[16] This assumption can be seen in the definition of $\bar{q}$ in Section 6.1.

[17] We assume in this dissertation that we use a parametric policy estimate and leave non-parametric estimates to future work.

Algorithm 4 Sampling Error Corrected Policy Gradient

Input: Initial policy parameters, $\boldsymbol{\theta}_0$, batch size m, a step-size for each iteration, α_i, and number of iterations n.

Output: Optimized policy parameters $\boldsymbol{\theta}_n$.

1: **for all** $i = 0$ to n **do**
2: Sample m steps $(S, A) \sim \pi_{\boldsymbol{\theta}_i}$
3: $\phi_i \leftarrow \underset{\phi}{\text{argmax}} \sum_{j=1}^{m} \log \pi_\phi(a_j|s_j)$
4: $g_{\text{sec}} \leftarrow \frac{1}{m} \sum_{j=1}^{m} \frac{\pi_{\boldsymbol{\theta}}(a_j|s_j)}{\pi_\phi(a_j|s_j)} \widehat{q}^{\pi_{\boldsymbol{\theta}}}(s_j, a_j, \cdot) \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_i}(a_j|s_j)$
5: $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha_i \cdot g_{\text{sec}}$
6: **end for**
7: **Return** $\boldsymbol{\theta}_n$

policy learning, i.e., learning with data that has been generated by a policy that is different from the current policy. Despite this connection to off-policy learning, we remain in the *on-policy* setting: data is collected with the current policy, used to update the current policy, and then discarded. In the off-policy setting, importance sampling corrects from the distribution that actions were sampled from to the distribution of actions under the current policy. SEC uses importance sampling to correct from the distribution that actions were observed at to the distribution of actions under the current policy.

The SEC estimator is related to the use of importance sampling for off-policy reinforcement learning where the behavior policy must be estimated before it can be used to form the importance weights. In practice, behavior policy estimation can be challenging when the distribution class of the true behavior policy is unknown (Raghu et al., 2018). Fortunately, in the policy gradient setting, we have complete access to the behavior policy and can specify the model class of π_ϕ to be the same as $\pi_{\boldsymbol{\theta}}$. We can even simplify the π_ϕ model class by estimating a policy that conditions on intermediate representations of $\pi_{\boldsymbol{\theta}}$. For example if $\pi_{\boldsymbol{\theta}}$ is a convolutional neural network, we can use all but the last layer of $\pi_{\boldsymbol{\theta}}$ as a feature extractor and then model π_ϕ as a linear function of these features. We evaluate this technique in our

experiments.

6.3 Theoretical Analysis

In this section we analyze the variance of $g_{\texttt{sec}}$ compared to that of $g_{\texttt{mc}}$. We make a few assumptions that simplify the analysis:

1. The action space is discrete and if a state is observed then all actions have also been observed in that state.

2. The return estimate $\widehat{q}^{\pi_{\boldsymbol{\theta}}}$ is computed independently of $\mathcal{T}$. This assumption implies $\bar{q}^{\pi_{\boldsymbol{\theta}}}(s,a)$ (as defined in Section 6.1) is a constant with respect to a fixed (s,a) in $\mathcal{T}$.

3. For all observed states, our estimated policy π_ϕ is equal to $\pi_{\mathcal{T}}$, i.e., if action a occurs k times in state s and s occurs n times in $\mathcal{T}$ then $\pi_\phi(a|s) = \frac{k}{n}$.

We briefly discuss the implications of these assumptions at the end of this section and evaluate SEC without these assumptions in Section 6.4.

Let $\mathbb{S}$ be the random variable representing the states in $\mathcal{T}$ and $\mathbb{A}$ be the random variable representing the actions in $\mathcal{T}$. We will use $\mathcal{T} = \{\mathbb{S}, \mathbb{A}\}$ to make explicit that $\mathcal{T}$ depends on both the randomness in the set of sampled states and sampled actions. We can now give the central theoretical claim of this chapter.

Proposition 6.1. *Let* $\text{Var}_{\mathbb{S},\mathbb{A}}(g)$ *denote the variance of estimator* g *with respect to random variables* $\mathbb{S}$ *and* $\mathbb{A}$*. For the Monte Carlo estimator,* $g_{\texttt{mc}}$*, and the* SEC *estimator,* $g_{\texttt{sec}}$*:*

$$\text{Var}_{\mathbb{S},\mathbb{A}}\left(g_{\texttt{sec}}(\{\mathbb{S},\mathbb{A}\})\right) \leq \text{Var}_{\mathbb{S},\mathbb{A}}\left(g_{\texttt{mc}}(\{\mathbb{S},\mathbb{A}\})\right)$$

Proof. We provide a proof sketch in this section. The full proof is provided in Appendix D.

We first note that both $g_{\texttt{sec}}$ and $g_{\texttt{mc}}$ can be written as:

$$g(\{\mathbb{S},\mathbb{A}\}) = \sum_{s\in\mathcal{S}} d_{\mathcal{T}}(s) \sum_{a\in\mathcal{A}} \pi_{\mathcal{T}}(a|s) w(s,a) \bar{q}^{\pi_{\boldsymbol{\theta}}}(s,a) \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) \tag{6.5}$$

where $w(s,a) = \frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\mathcal{T}}(a|s)}$ for $g_{\texttt{sec}}$ and $w(s,a) = 1$ for $g_{\texttt{mc}}$.

Using the law of total variance, the variance of (6.5) can be decomposed as:

$$\mathrm{Var}_{\mathbb{S},\mathbb{A}}\left(g(\{\mathbb{S},\mathbb{A}\})\right) = \underbrace{\mathbf{E}_{\mathbb{S}}\left[\mathrm{Var}_{\mathbb{A}}\left(g(\{\mathbb{S},\mathbb{A}\}|\mathbb{S})\right)\right]}_{\Sigma_{\mathbb{A}}} + \underbrace{\mathrm{Var}_{\mathbb{S}}\left(\mathbf{E}_{\mathbb{A}}\left[g(\{\mathbb{S},\mathbb{A}\})|\mathbb{S}\right]\right)}_{\Sigma_{\mathbb{S}}}.$$

The first term, $\Sigma_{\mathbb{A}}$, is the expected variance due to stochasticity in the action selection.

Claim 6.1. $\mathrm{Var}_{\mathbb{A}}\left(g_{\texttt{sec}}(\{\mathbb{S},\mathbb{A}\}|\mathbb{S})\right) = 0.$

Proof. See Appendix D. Intuitively, this claim follows from the fact that using $w(s,a) = \frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\mathcal{T}}(a|s)}$ results in all randomness due to $\mathbb{A}$ canceling out. □

From Claim 6.1, $\Sigma_{\mathbb{A}}$ will be zero since $\mathbf{E}_{\mathbb{S}}[0] = 0$. However, this term will be positive for $g_{\texttt{mc}}$ since the Monte Carlo estimator does *not* have zero variance in general.[18]

The second term, $\Sigma_{\mathbb{S}}$, is the variance due to only visiting a limited number of states before estimating the gradient.

Claim 6.2. $\mathbf{E}_{\mathbb{A}}\left[g_{\texttt{sec}}(\{\mathbb{S},\mathbb{A}\})|\mathbb{S}\right] = \mathbf{E}_{\mathbb{A}}\left[g_{\texttt{mc}}(\{\mathbb{S},\mathbb{A}\})|\mathbb{S}\right].$

Proof. See Appendix D. Under Assumption (1) and (3) the expectation over $\pi_{\mathcal{T}}$ (i.e., $\sum_{a\in\mathcal{A}} \pi_{\mathcal{T}}(a|s)$) in (6.5) is converted to an exact expectation over $\pi_{\boldsymbol{\theta}}$ ($\sum_{a\in\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s)$) and $g_{\texttt{mc}}$ is an unbiased estimator of this exact expectation. □

[18] The Monte Carlo estimator has zero variance with respect to action sampling only when $\widehat{q}^{\pi_{\boldsymbol{\theta}}}(s,a)$ is equal for all actions in any state.

From Claim 6.2, it follows that $\Sigma_{\mathbb{S}}$ will be the same for both $g_{\texttt{mc}}$ and $g_{\texttt{sec}}$. Since $\Sigma_{\mathbb{S}}$ is identical for both terms and $\Sigma_{\mathbb{A}}$ is zero for $g_{\texttt{sec}}$, the variance of $g_{\texttt{sec}}$ can be no more than that of $g_{\texttt{mc}}$. □

Our claim that the variance of $g_{\texttt{sec}}$ is at most that of $g_{\texttt{mc}}$ has been shown under a limiting set of assumptions. The assumption that all actions have been observed in all sampled states and that we can estimate $\pi_{\mathcal{T}}$ exactly limits the analysis to discrete state and action domains. Analyzing the estimators' variances under relaxed assumptions is an interesting direction for future work.

Finally, we note that in typical policy gradient implementations the assumption that $\widehat{q}^{\pi_{\boldsymbol{\theta}}}$ is computed independently of $\mathcal{T}$ is typically violated. In this case, the variance decomposition will have a third term that is due to variance in the return estimates:

$$\Sigma_{\tau} = \mathbf{E}_{\mathbb{S},\mathbb{A}}\left[\text{Var}\left(g(\{\mathbb{S},\mathbb{A}\}|\mathbb{S},\mathbb{A})\right)\right].$$

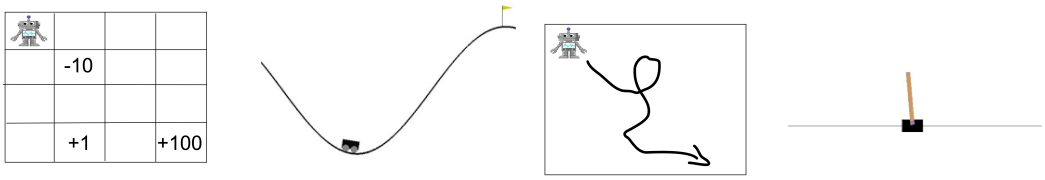
This term may not necessarily be less for either estimator and we leave its analysis to future work. We also discuss in Section 10.2 how the SEC estimator could be modified to lower the variance of the return estimates.

6.4 Empirical Study

In this section we present an empirical evaluation of the sampling error corrected policy gradient estimator. While the analysis in the previous section was based on limiting assumptions, we now evaluate whether $g_{\texttt{sec}}$ can lead to faster learning in practice, even when these assumptions are violated. Specifically, we study $g_{\texttt{sec}}$ in both discrete and continuous state spaces, in discrete and continuous action spaces, and when the return estimates are *not* independent of the gradient estimate. Our main empirical question is, "Does replacing $\widehat{q}^{\pi}(s,a,\cdot)$ with $\frac{\pi(a|s)}{\pi_{\phi}(a|s)}\widehat{q}^{\pi}(s,a,\cdot)$ lead to faster learning within a batch policy gradient method?"

6.4.1 Empirical Set-up

We first describe four reinforcement learning tasks and the motivation for evaluating SEC in these domains. Figure 6.2 displays images of these domains.



(a) Grid World (b) Mountain Car (c) LDS (d) Cart Pole

Figure 6.2: Illustrations of the domains used in our experiments. LDS is short for Linear Dynamical System.

6.4.1.1 Grid World

Our first domain is a 4×4 Grid World and we use REINFORCE (Williams, 1992) as the underlying batch policy gradient algorithm. The agent begins in grid cell $(0, 0)$ and trajectories terminate when it reaches $(3, 3)$. The agent receives a reward of 100 at termination, -10 at $(1, 1)$ and -1 otherwise. The agent's policy is a state-dependent softmax distribution over actions:

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{e^{\theta_{s,a}}}{\sum_{a' \in \mathcal{A}} e^{\theta_{s,a'}}}.$$

With this representation, the policy does *not* generalize across states or actions.

The SEC estimator estimates the policy by counting how many times each action is taken in each state. This domain closely matches the assumptions made in our theoretical analysis. Specifically, the state and action spaces are discrete and π_ϕ is exactly equal to $\pi_{\mathcal{T}}$. While we do not explicitly enforce the assumption that all actions are observed in all states, the small size of the state and action space ($|\mathcal{S}| = 16$ and $|\mathcal{A}| = 4$) makes it likely that this assumption holds.

In our implementation of REINFORCE we normalize the gradient estimates by dividing by their magnitudes and use a step-size of 1. The gradient for both methods is estimated with a batch size of 10 trajectories.

6.4.1.2 Tabular Mountain Car

Our second domain is a discretized version of the classic Mountain Car domain, where an agent attempts to move an under-powered car up a steep hill by accelerating to the left or right or sitting still. The original task has a state of the car's position (a continuous scalar in the range $[-1.2, 0.6]$) and velocity (a continuous scalar in the range $[-0.07, 0.07]$). Position is discretized into 6 bins and velocity into 8 bins for a total of 4292 discrete states. We use REINFORCE as the batch policy gradient algorithm. The agent's policy is a state-dependent softmax distribution over the three discrete actions as is used in the Grid World domain.

The SEC estimator estimates the policy by counting how many times each action is taken in each state. This domain has a large number of discrete states and it is unlikely that all actions are observed in all observed states. In this setting, $g_{\texttt{sec}}$ will have higher bias. This domain matches our theoretical setting in that states and actions are discrete and π_ϕ is exactly equal to $\pi_{\mathcal{T}}$.

As in Grid World we normalize the gradient estimates by dividing by their magnitudes and use a step-size of 1. We run each method with batch sizes of 100, 200, 600, and 800 trajectories.

6.4.1.3 Linear Dynamical System

Our third domain is a two-dimensional linear dynamical system with additive Gaussian noise. The reward is the agent's distance to the origin and trajectories last for 20 time-steps. In this domain the learning agent uses a linear Gaussian policy to select continuous valued accelerations in the x and y direction. We use

the OpenAI Baselines (Dhariwal et al., 2017) implementation of *trust-region policy optimization* (TRPO) as the underlying batch policy gradient algorithm (Schulman et al., 2015b). We set the generalized advantage estimation parameters (γ, λ) both to 1. Unless noted otherwise, we use the default OpenAI Baselines default values for all other hyperparameters. We estimate π_ϕ with ordinary least squares and estimate a state-independent variance parameter. In this domain, none of our theoretical assumptions hold. We include it to evaluate $g_{\texttt{sec}}$ with simple function approximation. We estimate the TRPO surrogate objective and constraint with 1000 steps per batch and set the KL-Divergence constraint, $\epsilon = 0.01$.

6.4.1.4 Cart Pole

Our final domain is the Cart Pole domain from OpenAI Gym (Brockman et al., 2016) and we again use TRPO. We estimate the TRPO surrogate objective and constraint with 200 steps per batch and set the KL-Divergence constraint, $\epsilon = 0.001$. The policy representation is a two layer neural network with 32 hidden units in each layer. The output of the network is the parameters of a softmax distribution over the two actions. We consider two parameterizations of π_ϕ:

1. π_ϕ is a neural network with the same architecture as $\pi_{\boldsymbol{\theta}}$. We estimate π_ϕ with batch gradient descent. This method is labeled SEC Neural Network.

2. π_ϕ is a linear policy that receives the activations of the last hidden layer of $\pi_{\boldsymbol{\theta}}$ as input. The dual π_ϕ and $\pi_{\boldsymbol{\theta}}$ architecture is shown in Figure 6.3. We estimate the weights of π_ϕ with gradient descent. This method is labeled SEC Linear.

Again, this domain violates all assumptions made in our theoretical analysis. We include this domain to study $g_{\texttt{sec}}$ with more complex function approximation. This setting allows us to study $g_{\texttt{sec}}$ with neural network policies but is simple enough to avoid extensive tuning of hyper-parameters.

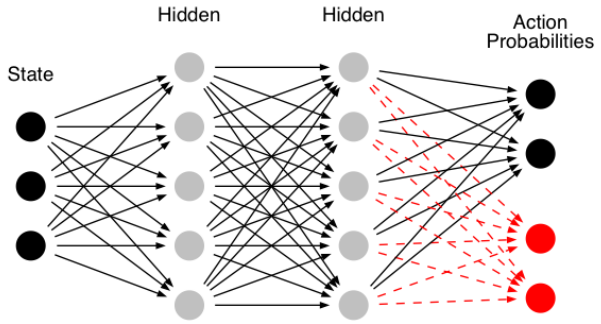


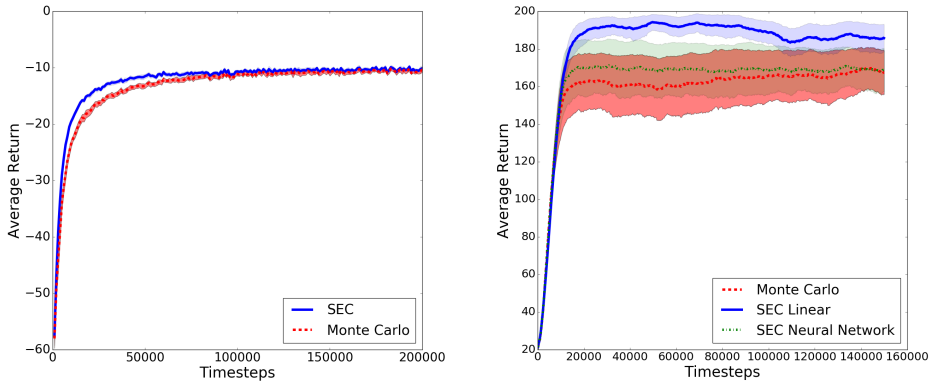
Figure 6.3: A simplified version of the neural network architecture used in Cart Pole. The true architecture has 32 hidden units in each layer. The current policy $\pi_{\boldsymbol{\theta}}$ is given by a neural network that outputs the action probabilities as a function of state (black nodes). The estimated policy, π_ϕ, is a linear policy that takes as input the activations of the final hidden layer of $\pi_{\boldsymbol{\theta}}$. Only the weights on the red, dashed connections are changed when estimating π_ϕ.

6.4.2 Empirical Results

We now present our empirical results.

6.4.2.1 Main Results

Results for the Linear Dynamical System (LDS), and Cart Pole environment are given in Figure 6.4. In all three domains, we see that the SEC methods lead to learning speed-up compared to the Monte Carlo based approaches. In the LDS and Mountain Car environments, SEC outperforms Monte Carlo in time to convergence to optimal. In Cart Pole, both variants of SEC learn faster initially, however, Monte Carlo catches up to the neural network version of SEC. This result demonstrates that we can leverage intermediate representations of $\pi_{\boldsymbol{\theta}}$ (in this case, the activations of the final hidden layer) to learn π_ϕ with a simpler model class. In fact, results suggest that fitting a simpler model improves performance. We hypothesize that simpler models require less hyper-parameter re-tuning throughout learning and so we get a more accurate estimate of $\pi_{\mathcal{T}}$ which leads to a more accurate sampling error



(a) LDS (b) Cart Pole

Figure 6.4: Learning results for the Linear Dynamical System (LDS) and Cart Pole domains. The x-axis is the number of timesteps and the y-axis is the average return of a policy. We run 25 trials of each method using different random seeds. The shaded region represents a 95% confidence interval. In both domains we see that all variants of sampling error corrected policy gradient outperforms the batch Monte Carlo policy gradient in either time to optimal convergence or final performance.

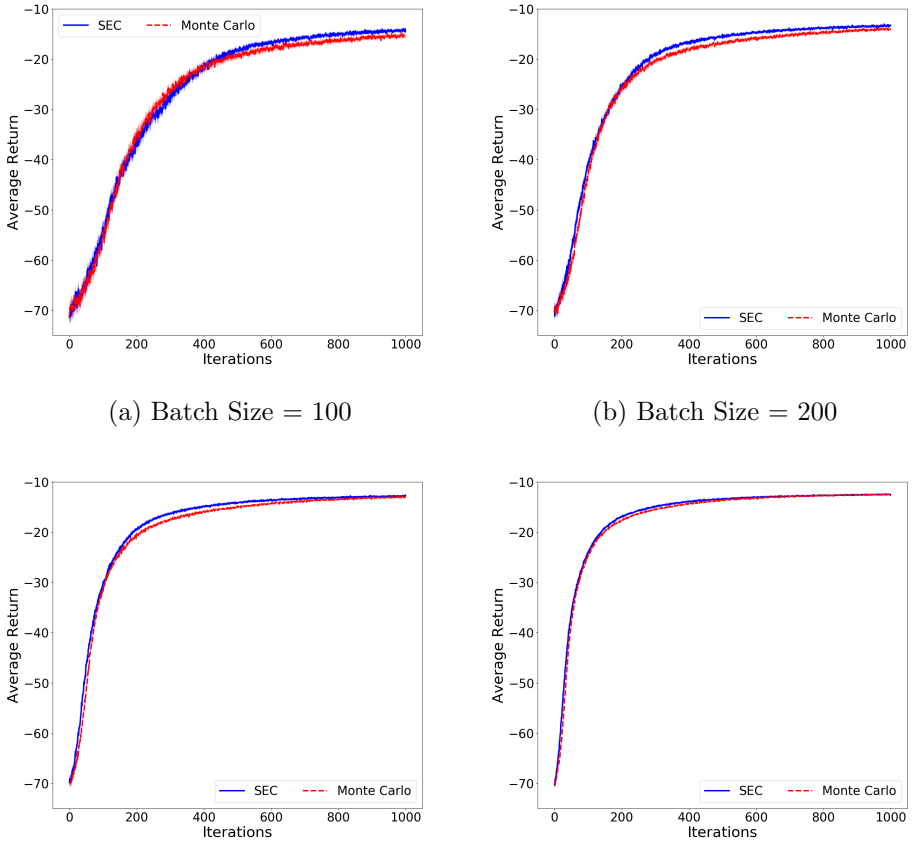
correction.

6.4.2.2 Mountain Car Batch Size

We also compare SEC to Monte Carlo in the Mountain Car domain. We run our experiments four times with a different batch size in each experiment. Each experiment consists of 25 trials for each algorithm.

Figure 6.5 shows results for each of the different tested batch sizes. For each batch size, we can see that SEC improves upon the Monte Carlo approach. The relative improvement does change across batches. With the largest batch size, improvement is marginal as π_ϕ will approach π and so SEC and Monte Carlo will return the same gradient. For the smallest batch size, improvement is again marginal – though the small batch size means Monte Carlo has higher variance, it also means that SEC may have higher bias as some actions will be unobserved in visited states.

Intermediate batch sizes have the widest gap between the two methods – the batch size is small enough that Monte Carlo has high variance but that SEC has less bias.



(a) Batch Size = 100

(b) Batch Size = 200

(c) Batch Size = 600

(d) Batch Size = 800

Figure 6.5: Learning results for the Mountain Car domain with different batch sizes. The x-axis is the number of iterations (i.e., the number of times the policy has been updated). The y-axis is average return. We run 25 trials of each method using different random seeds. The shaded region represents a 95% confidence interval. For all batch sizes we see that the sampling error corrected policy gradient outperforms the batch Monte Carlo policy gradient in either time to optimal convergence or final performance after 1000 iterations.

6.4.2.3 Grid World Ablations

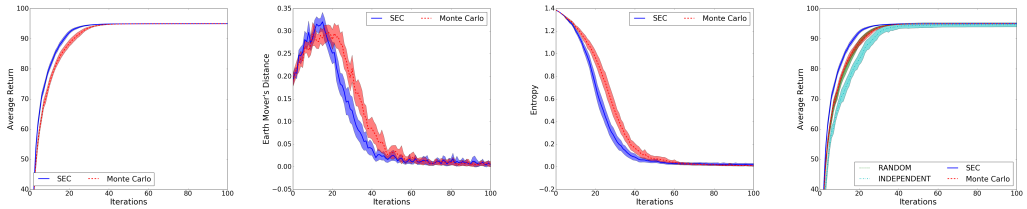
Figure 6.6 shows several results in the Grid World domain. First, Figure 6.6a shows that SEC leads to faster convergence compared to Monte Carlo. This domain most closely matches our theoretical assumptions where we showed SEC has lower variance than Monte Carlo gradient estimates. The lower variance translates into faster learning.

We also use the Grid World domain to perform a quantitative evaluation of sampling error. As a measure of sampling error we use the Earth Mover's distance between the current policy $\pi_{\boldsymbol{\theta}}$ and the empirical frequency of actions, $\pi_{\mathcal{T}}$. Intuitively, for any state, s, the Earth Mover's distance measures how much probability mass must be moved to transform $\pi_{\mathcal{T}}(\cdot|s)$ into $\pi_{\boldsymbol{\theta}}(\cdot|s)$.[19] Figure 6.6b shows that sampling error increases and then decreases during learning. Peak sampling error is aligned with where the learning curve gap between the two methods is greatest. Note that sampling error naturally decreases as learning converges because the policy becomes more deterministic. Figure 6.6c shows that the entropy of the current policy goes to zero, i.e., becomes more deterministic. A more deterministic policy will have less sampling error and so we expect to see less advantage from SEC as learning progresses.

Finally, we also verify the importance of using the same data to both estimate π_ϕ and estimate the policy gradient. Figure 6.6d introduces two alternatives to SEC:

- INDEPENDENT: Estimates π_ϕ with a separate set of m samples and then uses this estimate to estimate $\mathcal{T}_i$
- RANDOM: Instead of computing importance weights, we randomly sample weights from a normal distribution and use them in place of the learned SEC weights.

[19] We choose the Earth Mover's distance (also known as the Wasserstein distance) as opposed to the more commonly used KL-divergence since $\pi_{\mathcal{T}}$ and $\pi_{\boldsymbol{\theta}}$ may not share support. That is, there may be an action, a, where $\pi_{\mathcal{T}}(a|s)$ is 0 and $\pi_{\boldsymbol{\theta}}(a|s) > 0$.



(a) Avg. Return (b) Sampling Error (c) Entropy (d) Alternatives

Figure 6.6: Sampling error corrected policy gradient in the Grid World Domain. Figure 6.6a shows the average return for SEC and MC. Figure 6.6b shows earth mover's distance between the current policy and estimated policy at each iteration. Figure 6.6c shows policy entropy at each iteration. Figure 6.6d shows two alternative weight corrections. Results are averaged over 25 trials and confidence bars are for a 95% confidence interval.

Figure 6.6a shows that INDEPENDENT hurts performance compared to Monte Carlo. RANDOM performs marginally worse than Monte Carlo. This result demonstrates the need to use the same set of data to estimate π_ϕ and the gradient.

6.5 Summary

In this chapter we introduced Contribution 4 of this dissertation: the sampling error corrected policy gradient estimator (SEC). SEC corrects the empirical weighting of the observed samples to be closer to their true probability of occurring when executing the current policy $\pi_{\boldsymbol{\theta}_i}$. This weighting contrasts with the commonly-used batch Monte Carlo policy gradient estimator that weights each sample by its empirical frequency. Theoretical results show that under a limiting set of conditions SEC has lower variance than the Monte Carlo estimator. We also presented an empirical study of SEC and found that it can increase the learning speed of REINFORCE and trust-region policy optimization even when these theoretical conditions fail to hold.

This chapter concludes our contributions that improve how reinforcement learning agents sample and weight experience for policy value estimation and policy

improvement. In the following chapter, we turn to the use of simulated experience for reinforcement learning. We will then return to using off-policy data in the Chapter 8 where we will combine off-policy data with simulated experience for more efficient estimation of confidence intervals for policy value estimation.

Chapter 7

Learning with Simulated Data

So far in this dissertation, we have described ways to use off-policy data to improve the data efficiency of policy value estimation and policy improvement. In this chapter we turn our attention to simulated data and how it can be used in a reinforcement learning task. In Chapter 8 we will discuss a contribution towards using simulated and off-policy data together.

Simulation is a valuable tool for reinforcement learning for robotics research as execution of a robotic skill in simulation is comparatively easier than real world execution. However, the value of simulation learning may be limited by the inherent inaccuracy of simulators in modeling the dynamics of the physical world (Kober et al., 2013). As a result, learning that takes place in a simulator is unlikely to improve real world performance. In this chapter, we will focus on applying reinforcement learning in robotics though we are in fact interested in any setting where an inaccurate simulator of the target MDP is available a priori. In the literature, the problem of learning in simulation in a way that improves real world performance is known as the *sim-to-real* or *sim2real* problem. In Section 9.3 we discuss related work in this area and how our contribution fits into this literature.

In this chapter, we present Contribution 5 of this dissertation: an algorithm

that allows an autonomous agent to learn in a simulated environment and the resulting policy to improve performance in the real world.[20] This novel algorithm is demonstrated on three robot tasks on a simulated or physical bipedal humanoid robot.

7.1 Learning in a Simulator

In this chapter, we operate in the policy improvement setting introduced in Chapter 2. We are interested in learning a policy, π, for an MDP, $\mathcal{M}$, such that $v(\pi, \mathcal{M})$ is maximized. We wish to minimize the number of actions that must be taken in $\mathcal{M}$ before a good policy is learned.

Recall from Chapter 2 that a *simulator* or *model* for $\mathcal{M}$ is an MDP, $\mathcal{M}_{\texttt{sim}}$, that has the same state-space and action-space as $\mathcal{M}$ but a different state-transition function ($P_{\texttt{sim}}$ instead of P).[21] In this chapter we make the assumption that the reward function, r, is user-defined and thus is identical for $\mathcal{M}$ and $\mathcal{M}_{\texttt{sim}}$. However, the different dynamics distribution means that for any policy, π, $v(\pi, \mathcal{M}) \neq v(\pi, \mathcal{M}_{\texttt{sim}})$ since π induces a different trajectory distribution in $\mathcal{M}$ than in $\mathcal{M}_{\texttt{sim}}$. Thus, for any π' with $v(\pi', \mathcal{M}_{\texttt{sim}}) > v(\pi, \mathcal{M}_{\texttt{sim}})$, it does *not* follow that $v(\pi', \mathcal{M}) > v(\pi, \mathcal{M})$ – in fact $v(\pi', \mathcal{M})$ could be much worse than $v(\pi, \mathcal{M})$. In practice and in the literature, learning in simulation often fails to improve expected performance (Farchy et al., 2013; Christiano et al., 2016; Rusu et al., 2016b; Tobin et al., 2017).

[20] This chapter contains work that was previously published at AAAI 2017 (Hanna and Stone, 2017).

[21] A closely related body of work considers how learning can take place in simulation when the observations the agent receives are different from the real world (e.g., rendered images vs. natural images). We discuss this work in our related work section but consider this problem orthogonal to the problem of differing dynamics.

Connection to Previous Chapters

At first, discussing learning with simulated data may appear to be a significant departure from Chapters 3 to 6 that focused on using off-policy data. In fact, the off-policy learning problem and the sim-to-real learning problem are related through the problem of *distribution shift*. Consider that in on-policy learning, trajectories are generated from:

$$\Pr(H|\pi, \mathcal{M}) \coloneqq d_0(S_0)\pi(A_0|S_0)P(S_1|S_0, A_0)\cdots P(S_{L-1}|S_{L-2}, A_{L-2})\pi(A_{L-1}|S_{L-1}).$$

In off-policy learning, the action probabilities change to those of a different behavior policy and so the distribution of trajectories becomes:

$$\Pr(H|\pi_b, \mathcal{M}) \coloneqq d_0(S_0)\pi_b(A_0|S_0)P(S_1|S_0, A_0)\cdots P(S_{L-1}|S_{L-2}, A_{L-2})\pi_b(A_{L-1}|S_{L-1}).$$

In sim-to-real learning, the environment probabilities change and so the distribution of trajectories becomes:

$$\Pr(H|\pi, \mathcal{M}_{\texttt{sim}}) \coloneqq d_0(S_0)\pi(A_0|S_0)P_{\texttt{sim}}(S_1|S_0, A_0)\cdots P_{\texttt{sim}}(S_{L-1}|S_{L-2}, A_{L-2})\pi(A_{L-1}|S_{L-1}).$$

In both problems we must deal with a shifting data distribution – only the cause of the distribution shift changes.

Though we are still dealing with a distribution shift in our data, the importance sampling techniques introduced in Chapters 3 to 6 are inapplicable to the sim-to-real problem because P is typically unknown. Thus we cannot compute the numerator for a transition-probability-based importance weight $\frac{P(s'|s,a)}{P_{\texttt{sim}}(s'|s,a)}$.

7.2 Grounded Simulation Learning

In this section we introduce the *grounded simulation learning* (GSL) framework as presented by Farchy et al. (2013). Contribution 5 is an instantiation of this

framework. GSL allows reinforcement learning in simulation to succeed by using trajectories from $\mathcal{M}$ to first modify $\mathcal{M}_{\texttt{sim}}$ such that the modified $\mathcal{M}_{\texttt{sim}}$ is a higher fidelity model of $\mathcal{M}$. The process of making the simulator more like the real world is referred to as *grounding*.

The GSL framework assumes the following:

1. There is an imperfect simulator MDP, $\mathcal{M}_{\texttt{sim}}$, that models the MDP environment of interest, $\mathcal{M}$. Furthermore, $\mathcal{M}_{\texttt{sim}}$ must be *modifiable*. In this dissertation, we formalize modifiable as meaning that the simulator has parameterized transition probabilities $P_\phi(\cdot|s,a) \coloneqq P_{\texttt{sim}}(\cdot|s,a;\phi)$ where the vector ϕ can be changed to produce, in effect, a different simulator.

2. There is a policy improvement algorithm, `optimize`, that searches for π which increase $v(\pi, \mathcal{M}_{\texttt{sim}})$. The `optimize` routine returns a set of candidate policies, Π to evaluate in $\mathcal{M}$.

We formalize the notion of grounding as minimizing a similarity metric between the trajectory distribution of the real world and simulation. Let $d(p,q)$ be a measure of similarity between probabilities p and q. Given a dataset of trajectories, $\mathcal{D}_{\texttt{real}} \coloneqq \{H_i\}_{i=1}^m$, sampled from some policy in $\mathcal{M}$, simulator grounding of $\mathcal{M}_{\texttt{sim}}$ amounts to finding $\phi^\star$ such that:

$$\phi^\star = \operatorname*{argmin}_\phi \sum_{h \in \mathcal{D}_{\texttt{real}}} d\left(\Pr(h|\pi), \Pr_{\texttt{sim}}(h|\pi;\phi)\right) \tag{7.1}$$

where $\Pr(h|\pi)$ is the probability of observing trajectory h in the real world and $\Pr_{\texttt{sim}}(h|\pi;\phi)$ is the probability of h in the simulator with dynamics parameterized by ϕ. For instance, if $d(p(h), q(h)) \coloneqq -\log q(h)$ then $\phi^\star$ minimizes the negative log-likelihood or equivalently the Kullback-Leibler divergence between $\Pr(\cdot|\pi, \mathcal{M})$ and $\Pr_{\texttt{sim}}(\cdot|\pi, \phi^\star)$.

Assuming we can solve (7.1), the GSL framework is as follows:

1. Execute an initial policy, π_0, in the real world to collect a data set of trajectories, $\mathcal{D}_{\texttt{real}} = \{H_j\}_{j=1}^{m}$.

2. Solve (7.1) to find $\phi^\star$ that makes $\Pr(H|\pi_0, \mathcal{M}_{\texttt{sim}})$ closer to $\Pr(H|\pi_0, \mathcal{M})$ for all $H \in \mathcal{D}_{\texttt{real}}$.

3. Use `optimize` to find a set of candidate policies Π that improve $v(\cdot, \mathcal{M}_{\texttt{sim}})$ in the modified simulation.

4. Evaluate each proposed $\pi_c \in \Pi$ in $\mathcal{M}$ and return the policy:

$$\pi_1 \coloneqq \underset{\pi_c \in \Pi}{\operatorname{argmax}}\, v(\pi_c, \mathcal{M}).$$

GSL can be applied iteratively with π_1 being used to collect more trajectories to ground the simulator again before learning π_2. The re-grounding step is necessary since changes to π result in changes to the distribution of states that the agent observes. When the distribution changes, a simulator that has been modified with data from the state distribution of π_0 may be a poor model under the state distribution of π_1. The entire GSL framework is illustrated in Figure 7.1.

7.3 The Grounded Action Transformation Algorithm

We now introduce Contribution 5 of this dissertation – a novel GSL algorithm called the *grounded action transformation* (GAT) algorithm. GAT improves the grounding step (Step 2) of the GSL framework. The main idea behind GAT is to augment the simulator with a differentiable *action transformation function*, g, which transforms the agent's simulated action into an action which – when taken in simulation – produces the same transition that would have occurred in the physical system. The function, g, is represented with a parameterized function approximator whose parameters serve as ϕ for the augmented simulator.

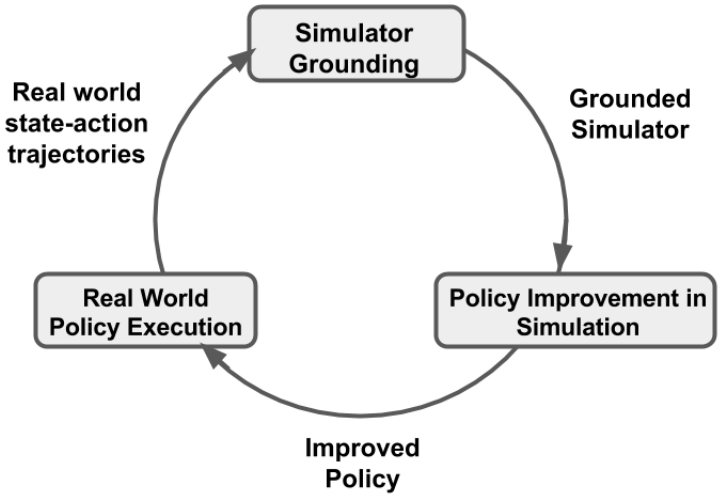


Figure 7.1: Diagram of the grounded simulation learning framework.

Before presenting this chapter's contribution, we first refine the state-space definition from Chapter 2 to facilitate the presentation. We will assume that the agent's state is a vector in $\mathbb{R}^k$ and action is a vector in $\mathbb{R}^l$ for some $k, l \in \mathbb{N}_+$. Let $\mathbf{x}$ be a subset of the components of state $\mathbf{s}$ and let $\mathcal{X}$ be the set of all possible values for $\mathbf{x}$. We refer to $\mathbf{x}$ as the state variables of interest.

Our instantiation of GAT learns two functions: f which predicts the effects of actions in $\mathcal{M}$ and $f_{\texttt{sim}}^{-1}$, which predicts the action needed in simulation to reproduce the desired effects. The function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{X}$ is a forward model that predicts the effect on the state variables of interest given an action chosen in a particular state in $\mathcal{M}$. The function $f_{\texttt{sim}}^{-1} : \mathcal{S} \times \mathcal{X} \to \mathcal{A}$ is an inverse model that predicts the action that causes a particular effect on the state variables of interest given the current state in simulation. The overall transformation function $g : \mathcal{S} \times \mathcal{A} \to \mathcal{A}$ is specified as $g(\mathbf{s}, \mathbf{a}) := f_{\texttt{sim}}^{-1}(\mathbf{s}, f(\mathbf{s}, \mathbf{a}))$. When the agent is in state $\mathbf{s}_t$ in the simulator and takes action $\mathbf{a}_t$, the augmented simulator replaces $\mathbf{a}_t$ with $g(\mathbf{s}_t, \mathbf{a}_t)$ and the simulator returns $\mathbf{s}_{t+1}$ where $\mathbf{x}_{t+1}$ is closer in value to what would be observed in $\mathcal{M}$ had $\mathbf{a}_t$

been taken there.

The advantage of GAT is that learning f and $f_{\texttt{sim}}^{-1}$ is a supervised learning problem which can be solved with a variety of techniques. Figure 7.2 illustrates the augmented simulator. Pseudocode for the full GAT algorithm is given in Algorithm 5.

The main motivation for modifying the agent's simulated action is that it allows us to treat the simulator as a blackbox. While physics-based simulators typically have a large number of parameters determining the physics of the simulated environment (e.g., friction coefficients, gravitational values) these parameters are not necessarily amenable to numerical optimization of (7.1). In other words, it may be computationally intensive to determine how to change a physical parameter to make the simulator produce trajectories closer to the ones we observe in the real world. Action modification allows us to transform simulator modification into a supervised learning problem.

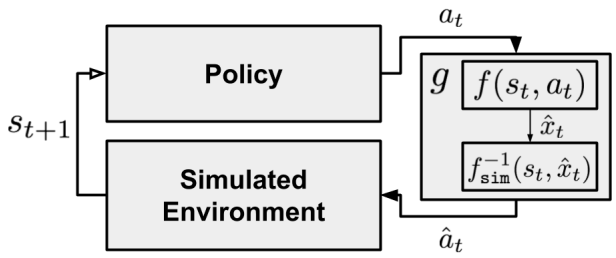


Figure 7.2: The augmented simulator which can be grounded to the real world with supervised learning.

The result of this form of simulator modification is not necessarily a globally more accurate simulator for the real world. Our only goal is that the simulator is more realistic for trajectories *sampled with the grounding policy*. If we can achieve this goal, then we can locally improve the policy without any additional real world data. A simulator that is more accurate globally may provide a better starting point for GAT, however, if we can focus on simulator modification local to the grounding

Algorithm 5 Grounded Action Transformation (GAT). **Input:** An initial policy, π_0, the environment, $\mathcal{M}$, a simulator, $\mathcal{M}_{\texttt{sim}}$, smoothing parameter α, and a policy improvement method, `optimize`. The function `rolloutN`$(\texttt{Env}, \pi, \texttt{n})$ executes n trajectories with π in the provided environment, `Env`, and returns the observed state transition data. The functions `trainForwardModel` and `trainInverseModel` estimate models of the forward and inverse dynamics respectively given a dataset of trajectories.

$i \leftarrow 0$
repeat
  $\mathcal{D}_{\texttt{real}} \leftarrow \texttt{RolloutN}(\mathcal{M}, \pi_i, n)$
  $\mathcal{D}_{\texttt{sim}} \leftarrow \texttt{RolloutN}(\mathcal{M}_{\texttt{sim}}, \pi_i, n)$
  $f \leftarrow \texttt{trainForwardModel}(\mathcal{D}_{\texttt{real}})$
  $f^{-1}_{\texttt{sim}} \leftarrow \texttt{trainInverseModel}(\mathcal{D}_{\texttt{sim}})$
  $g(s, a) \leftarrow f^{-1}_{\texttt{sim}}(s, f(s, a))$
  $\Pi \leftarrow \texttt{optimize}(\mathcal{M}_{\texttt{sim}}, \pi_i, g)$
  $i \leftarrow i + 1$
  $\pi_i \leftarrow \text{argmax}_{\pi \in \Pi}\, v(\pi)$
until $v(\pi_i) < v(\pi_{i-1})$

Return $\text{argmax}_i\, v(\pi_i)$

policy we can still obtain policy improvement in low fidelity simulators.

We also note that GAT minimizes the error between the immediate state transitions of $\mathcal{M}_{\texttt{sim}}$ and those of $\mathcal{M}$. Another possible objective would be to observe the difference between trajectories in $\mathcal{M}$ and $\mathcal{M}_{\texttt{sim}}$ and ground the simulator to minimize the total error over a trajectory. Such an objective could lead to an action modification function g that accepts short-term error if it reduces the error over the entire trajectory. Here we will accept minimizing one-step error as a good proxy for minimizing our ultimate objective which is that the current policy π produces similar trajectories in both $\mathcal{M}$ and $\mathcal{M}_{\texttt{sim}}$.

7.4 Empirical Study

We now present an empirical study of applying GAT for reinforcement learning with simulated data.

7.4.1 Empirical Set-up

We evaluate GAT on several different robot learning tasks on the NAO robot. We describe these tasks here.

7.4.1.1 General NAO Task Description

All empirical tasks in this chapter use either a simulated or physical Softbank NAO robot.[22] The NAO is a humanoid robot with 25 degrees of freedom (See Figure 7.3a). Though the NAO has 25 degrees of freedom, we restrict ourselves to observing and controlling 15 of them (we ignore joints that are less important for our experimental tasks – joints in the head, hands, and elbows). We will refer to the degrees of freedom as the joints of the robot. Figure 7.4 shows a diagram of the NAO and its different joints.

We define the state variables of interest to be the angles of the robot's joints. In addition to angular position, the robot's state consists of joint angular velocities and other task-dependent variables. The robot's actions are desired joint angular positions which are implemented at a lower software level using PID control.

Since the output of g may not be smooth from timestep to timestep, we use a smoothing parameter, α, to ensure stable motions. The action transformation function (Algorithm 5, line 7) is then defined as:

$$g(\mathbf{s}, \mathbf{a}) \coloneqq \alpha f^{-1}_{\texttt{sim}}(\mathbf{s}, f(s, \mathbf{a})) + (1 - \alpha)\mathbf{a}.$$

[22] https://www.ald.softbankrobotics.com/en

(a) A Softbank NAO Robot (b) NAO in Gazebo (c) NAO in SimSpark

Figure 7.3: The three robotic environments used in this chapter. The Softbank NAO is our target physical robot. The NAO is simulated in the Gazebo and SimSpark simulators. Gazebo is a higher fidelity simulator which we also use as a surrogate for the real world in an empirical comparison of grounded action transformation (GAT) to baseline methods.

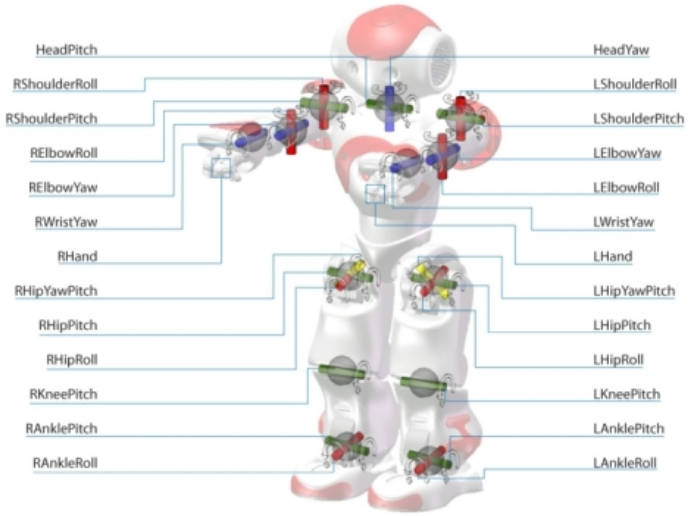


Figure 7.4: Diagram of the Softbank NAO robot with joints (degrees of freedom) labeled. Each joint has a sensor that reads the current angular position of the joint and can be controlled by providing a desired angular position for the joint. In this work, we ignore the HeadYaw, HeadPitch, left and right ElbowRoll, left and right ElbowYaw, left and right WristYaw, and left and right Hand joints. There is also no need to control the right HipYawPitch joint as, in reality, this degree of freedom is controlled by the movement of the left HipYawPitch Joint. This image was downloaded from: http://doc.aldebaran.com/2-8/family/nao_technical/lola/actuator_sensor_names.html

In our experiments, we set α as high as possible subject to the walk remaining stable.

In all tasks our implementation of GAT uses a history of the joint positions and desired joint positions as an estimate of the NAO's state to input into the forward and inverse models. Instead of directly predicting $\mathbf{x}_{t+1}$, the forward model, f, is trained to predict the change in $\mathbf{x}_t$ after taking $\mathbf{a}_t$. The inverse model $f^{-1}_{\texttt{sim}}$ takes the current $\mathbf{x}_t$ and a desired change in $\mathbf{x}_t$ and outputs the action needed to cause this change. Since both the state variables of interest and actions have angle units, we train both f and $^{-1}_{\texttt{sim}}$ to output the sine and cosine of each output angle. From these values we can recover the predicted output with the arctan function.

We consider two simulators in this work: the Simspark[23] Soccer Simulator used in the annual RoboCup 3D Simulated Soccer competition and the Gazebo simulator from the Open Source Robotics Foundation.[24] SimSpark enables fast simulation but is a lower fidelity model of the real world. Gazebo enables high fidelity simulation with an additional computational cost. The NAO model in both of these simulations is shown in Figure 7.3a.

Across all tasks we learn the policy using the covariance matrix adaptation evolutionary strategies (CMA-ES) algorithm (Hansen et al., 2003). CMA-ES is a stochastic search algorithm that iteratively updates a distribution over candidate policies. At each iteration, CMA-ES samples a set of policy parameter values from a Gaussian distribution. It then uses the evaluation of each candidate policy in simulation to update the sampling distribution for the next iteration. CMA-ES has been found to be very effective at optimizing robot skills in simulation (Urieli et al., 2011). In all experiments we sample 150 candidate policies at each iteration as we were only able to submit up to 150 parallel policy evaluations at a time on the University of Texas Computer Science distributed computing cluster.

[23] http://simspark.sourceforge.net
[24] http://gazebosim.org

7.4.1.2 Arm Control

Our first task requires the NAO to learn to raise its arms from its sides to a goal position, $\mathbf{p}^\star$ which is defined to be halfway to horizontal (lift 45 degrees). We call this task the "Arm Control" task.

In this task, the robot's policy only controls the two shoulder joints responsible for raising and lowering the arms. The angular position of these joints are the state variables of interest, $\mathbf{x}$. The policy is a linear mapping from $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ to the action $\mathbf{a}_t$:

$$\pi(\mathbf{x}_t, \mathbf{x}_{t-1}) = \mathbf{w} \cdot (\mathbf{x}_t, \mathbf{x}_{t-1}) + \mathbf{b}$$

where $\mathbf{w}$ and $\mathbf{b}$ are learnable parameters. The agent receives at time t a reward:

$$r(\mathbf{x}_t) = \frac{1}{|\mathbf{x}_t - \mathbf{p}^\star|_2^2}$$

and the episode terminates after 200 steps or when either of the robot's arms raise higher than 45 degrees. The optimal policy is to move as close as possible to 45 degrees without lifting higher.

We apply GAT for sim-to-sim transfer from Simspark ($\mathcal{M}_{\texttt{sim}}$) to Gazebo ($\mathcal{M}$ – effectively treating Gazebo as the real world). In Simspark, the shoulder joints are more responsive to commands and thus the robot learns it must take weaker actions to prevent overshooting the target. In Gazebo, the same policy fails to get the arms close to the target.

We represent f and $f_{\texttt{sim}}^{-1}$ with linear functions. To train f we collect 50 trajectories in $\mathcal{M}$ and train $f_{\texttt{sim}}^{-1}$ with 50 trajectories from $\mathcal{M}_{\texttt{sim}}$. For CMA-ES, we optimize the policy for 50 iterations.

7.4.1.3 Linear Walk Policy Optimization

Our second task is walking forward with a linear control policy. The state variables of interest are 10 joints in the robot's legs (ignoring the left HipYawPitch joint) and the 4 joints controlling its shoulders. The actions are desired angular positions for all 15 of these joints.

The policy inputs are the gyroscope that measures forward-backward angular velocity, y, and the gyroscope that measures side-to-side angular velocity, x. We also provide as input an open-loop sine wave. The sine wave encodes prior knowledge that a successful walking policy will repeat actions periodically. The final form of the policy is:

$$\pi(\langle x, y, \sin(c \cdot t)\rangle) = \mathbf{w} \cdot \langle x, y, \sin(c \cdot t)\rangle + \mathbf{b}$$

where c is a learnable scalar that controls the walking step frequency. The policy outputs only commands for the left side of the robot's body and the commands for the right side are obtained by reflecting these commands around a learned value. That is, for each joint, j, on the left side of the robot's body we learn a parameter ψ_j and obtain the action for the right side of the robot's body by reflecting the policy's output for j across ψ_j. This representation is equivalent to expressing the policy for the right side of the robot's body as:

$$\pi_r(\langle x, y, \sin(c \cdot t)\rangle) = \psi - (\mathbf{w} \cdot \langle x, y, \sin(c \cdot t)\rangle + \mathbf{b} - \psi).$$

In our experiments, instead of optimizing a separate ψ vector, we clamp ψ to be equal to $\mathbf{b}$.

We define the reward as a function of entire trajectories instead of s, a pairs. Let $\Delta(h)$ be the robot's forward change in position during trajectory h and let $\mathbb{I}(h)$

take value 1 if the robot falls during trajectory h and 0 otherwise. In simulation:

$$g(h) \coloneqq \Delta(h) - 25 \cdot \mathbb{I}(h)$$

where the penalty of -25 encourages policies that cause the robot to walk more stably. On the physical robot we only measure forward distance traveled; if the robot falls we count the distance traveled as zero:

$$g(h) \coloneqq \Delta(h) \cdot (1 - \mathbb{I}(h)).$$

We apply GAT for sim-to-real transfer from Simspark to the physical NAO. We learn f and $f^{-1}_{\texttt{sim}}$ with linear regression. To train f we collect 10 trajectories in $\mathcal{M}$ and train $f^{-1}_{\texttt{sim}}$ with 50 trajectories from $\mathcal{M}_{\texttt{sim}}$. We chose 10 trajectories for $\mathcal{M}$ because after 10 the robot's motors may begin to heat up which changes the dynamics of the joints.

7.4.1.4 Sim-to-Real Walk Engine Policy Optimization

Finally, we evaluate GAT on the task of bipedal robot walking with a state-of-the-art walk controller for the NAO robot. The initial policy is the open source University of New South Wales (UNSW) walk engine developed for RoboCup Standard Platform League (SPL) competitions (Ashar et al., 2015; Hall et al., 2016). This walk controller has been used by at least one team in the 2014, 2015, 2016, 2017, 2018, 2019 RoboCup Standard Platform League (SPL) championship games in which teams of five NAOs compete in soccer matches. To the best of our knowledge, it is the fastest open source walk available for the NAO. The UNSW walk engine has 15 parameters that determine features of the walk (see Table 7.1 for a full list of these parameters). The values of the parameters from the open source release constitute the parameterization of the initial policy π_0. Hengst (2014) describes the UNSW walk controller in more

Parameter Name	Parameter Value
Center of Mass Offset	0.01
Base Walk Period	0.23
Walk Hip Height	0.23
Max Forward	0.3
Max Left Step	0.2
Max Turn	0.87
Max Forward Change	0.15
Max Left Change	0.2
Max Turn Change	0.8
Base Leg Lift	0.012
Arm Swing	6.0
Pendulum Height	300.0
Forward Extra Foot Height	0.01
Left Extra Foot Height	0.02
Start Lift Divisor	3.5

Table 7.1: The initial parameter values found in the open source release of the UNSW walk engine. Some of these values were explicit parameters in the open source release; others were hard-coded constants that we chose to allow CMA-ES to modify during policy optimization.

detail.

For this task, $v(\pi, \mathcal{M})$ is the average forward walk velocity while executing π. On the physical robot, a trajectory terminates once the robot has walked four meters ($\approx$ 20.5s with the initial policy) or falls. In simulation a trajectory terminates after a fixed time interval (7.5 seconds in SimSpark and 10 seconds in Gazebo) or when the robot falls. For policy improvement in simulation, we apply CMA-ES for 10 iterations with 150 candidate policies evaluated in each iteration.

We implement GAT with two two-hidden-layer neural networks – one for f and one for $f_{\texttt{sim}}^{-1}$. Each function is a neural network with 200 hidden units in the first layer and 180 hidden units in the second. The network architectures were selected based on error measured on a held-out data set.

The data set $\mathcal{D}$ consists of 15 trajectories collected with π_0 on the physical NAO. To ensure the robot's motors stayed cool, we waited five minutes after

collecting every five trajectories. For each iteration of GAT, we run 10 iterations of the CMA-ES algorithm. For each iteration of CMA-ES we select argmax $v(\pi, \mathcal{M}_{\texttt{sim}})$ and evaluate it on the physical robot. If the robot falls in any trajectory the policy is considered unstable.

7.4.1.5 Sim-to-Sim Walk Engine Policy Optimization

We also present a sim-to-sim evaluation of GAT using Gazebo as a surrogate for the real world. Unless stated otherwise, all experimental details are the same as those used in the sim-to-real evaluation. As baselines, we evaluate the effectiveness of GAT compared to learning with no grounding and grounding $\mathcal{M}_{\texttt{sim}}$ by adding Gaussian noise to the robot's actions. Adding an "envelope" of noise has been used before to minimize simulation bias by preventing the policy improvement algorithm from overfitting to the simulator's dynamics (Jakobi et al., 1995). We refer to this baseline as NOISE-ENVELOPE. We hypothesize that GAT is modifying simulation in a more effective way then just forcing learning to be robust to perturbation and will thus obtain a higher level of performance.

For GAT we collect 50 trajectories of robot experience to train f and 50 trajectories of simulated experience to train $f_{\texttt{sim}}^{-1}$. For each method, we run 10 iterations of the CMA-ES algorithm. In each iteration, 150 candidate policies are sampled and each is evaluated in simulation with 20 trajectories. Overall, the CMA-ES optimization requires 30,000 simulated trajectories for each trial. This process is repeated 10 times for each method.

7.4.2 Empirical Results

We now present our main empirical results.

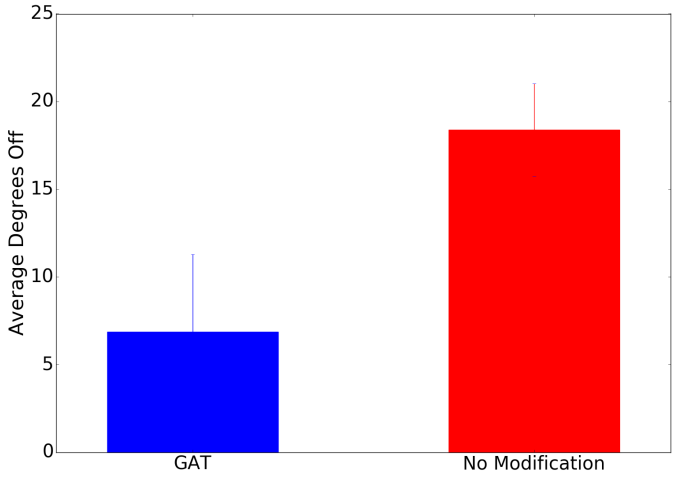


Figure 7.5: Mean performance of best policies found on the Arm Control task. We run 10 trials using GAT and 10 trials directly transferring from $\mathcal{M}_{\texttt{sim}}$ to $\mathcal{M}$ ("No Modification"). The y-axis gives the average distance to the target position during a trajectory (lower is better). Error bars are for a 95% confidence interval.

7.4.2.1 Arm Control Results

On the Arm Control task we evaluate whether GAT allows learning better policies in simulation than learning without simulator modification. We refer to the latter method as "No Modification." For each method, we run 10 trials. On each trial we run 50 iterations of CMA-ES. For each iteration we take the best performing candidate policy and evaluate it in $\mathcal{M}$. Our main point of comparison is which method finds a policy that allows the robot to move its arms closer to the target position (higher $v(\pi, \mathcal{M})$).

Figure 7.5 shows the mean performance of $v(\pi', \mathcal{M})$ for π' learned in simulation either with GAT or with "No Modification." Results show that GAT is able to overcome the reality gap and results in policies that reduce error in arm position.

We also visualize the effect of the action modification function, g, in the simulator. Figure 7.6 shows how the robot's LeftShoulderPitch joint moves in $\mathcal{M}$,

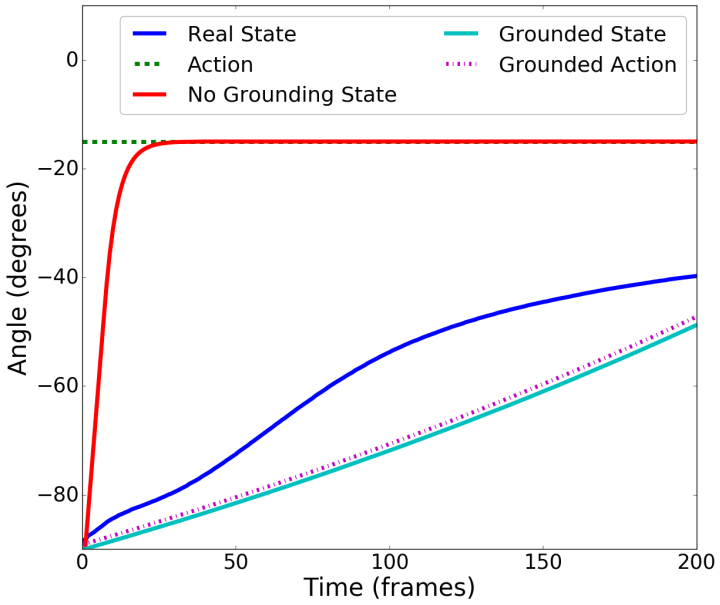


Figure 7.6: Visualization of the robot's LeftShoulderPitch joint position in $\mathcal{M}$, $\mathcal{M}_{\texttt{sim}}$, and $\mathcal{M}_{\texttt{sim}}$ after applying GAT. The x-axis is time in frames (50 frames per second). The y-axis has units of angles which is the unit for both the plotted actions and states. Trajectories were generated in each environment with a policy that sets a constant desired position of -15 degrees ("Action"). "Real State" shows the LeftShoulderPitch position in $\mathcal{M}$, "No Grounding State" shows position in $\mathcal{M}_{\texttt{sim}}$, and "Grounded State" shows position in the grounded $\mathcal{M}_{\texttt{sim}}$. "Grounded Action" shows the action that the GAT action modification function takes in place of "Action."

$\mathcal{M}_{\texttt{sim}}$, and the grounded $\mathcal{M}_{\texttt{sim}}$ when a constant joint command of -15 degrees is applied. In $\mathcal{M}_{\texttt{sim}}$ the position of the LeftShoulderPitch responds immediately to the command while in $\mathcal{M}$ the position changes much more slowly. After applying GAT, the position changes much slower in simulation as the action modification function reduces the magnitude of the desired change.

7.4.2.2 Linear Policy Walking Results

In the Linear Policy Walking task we measure performance based on how far forward the robot walks. The initial policy fails to move the robot forward at all – though it is executing a walking controller, its feet never break the friction of the carpet and so it remains at the starting position. We run five trials of learning with simulator modification and five trials without. On average learning in simulation with GAT resulted in the robot moving 4.95 cm forward while without simulator modification the robot only moved 1.3 cm on average.

Across the five trials *without* modification, two trials fail to find any improvement. The remaining three only find improvement in the first iteration of CMA-ES – before CMA-ES has been able to begin exploiting inaccuracies in the simulation. In contrast, all trials *with* simulator modification find improving policies and improvement comes in later learning iterations (on average iteration 3 is the best).

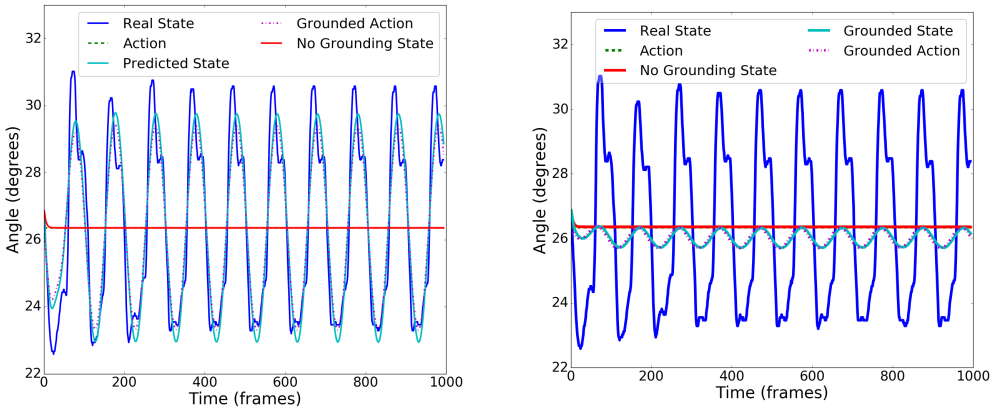
We also plot example trajectories to see how the modified and unmodified simulations compare to reality. Instead of plotting all state and action variables, we only plot the state variable representing the robot's right AnklePitch joint and the action that specifies a desired position for this joint. This joint was chosen because the main failure of policies learned without simulator modification is that the robot's feet never break the friction of the carpet. Learning to properly move the ankles may be important for a policy to succeed in the real world.

Figure 7.7a shows the prediction of joint position for the learned forward model, f, as well as the joint position in the real world and simulation. The "Predicted State" curve is generated by using f as a simulator of how the joint position changes in response to the actions.[25] Figure 7.7a shows that in the real world the right AnklePitch joint oscillates around the desired angular position as given by the robot's

[25]Note that f would *not* suffice for policy improvement as it only models how the joint positions change and *not* the effect of these changes on walk velocity.

action. The forward model f predicts this oscillation while the simulator models the joint position as static.

Figure 7.7b shows the actual real world and simulated trajectories, both for the modified and unmodified simulators. Though the modified simulator still fails to capture all of the real world oscillation, it does so more than no modification. Learning in a simulator that more accurately models this motion leads to policies that are able to lift the robot's legs enough to walk.



(a) Predicted trajectory

(b) Actual trajectory

Figure 7.7: Visualization of the robot's right AnklePitch joint during the Linear Policy Walking task. Both sub-figures show the position trajectory for $\mathcal{M}$ (denoted "Real State") and $\mathcal{M}_{\texttt{sim}}$ ("No Grounding State"). They also both show the action though it is covered by the "No Grounding State" curve. Figure 7.7a shows the GAT forward model's prediction of position given the same action sequence. Figure 7.7b shows the actual position when acting in the modified simulation.

7.4.2.3 Simulator to Physical NAO Results

Table 7.2 gives the physical world walk velocity of policies learned in simulation with GAT. The physical robot walks at a velocity of 19.52 cm/s with π_0. Two iterations of GAT with SimSpark increased the walk velocity of the NAO to 27.97

Method	Velocity (cm/s)	% Improve
π_0	19.52	0.0
GAT SimSpark π_1	26.27	34.58
GAT SimSpark π_2	27.97	43.27
GAT Gazebo π_1	26.89	37.76

Table 7.2: This table gives the maximum learned velocity and percent improvement for each method starting from π_0 (top row).

cm/s – an improvement of 43.27% compared to π_0.[26] GAT with SimSpark and GAT with Gazebo both improved walk velocity by over 30%.

Policy improvement with CMA-ES required 30,000 trajectories per iteration to find the 10 policies that were evaluated on the robot. In contrast the total number of trajectories executed on the physical robot is 65 (15 trajectories in $\mathcal{D}$ and 5 evaluations per $\pi_c \in \Pi$). This result demonstrates GAT can use sample-intensive simulation learning to optimize real world skills with a low number of trajectories on the physical robot.

Farchy et al. (2013) demonstrated the benefits of re-grounding and further optimizing π. We reground with 15 trajectories collected with the best policy found by GAT with SimSpark and optimize for a further 10 iterations of CMA-ES in simulation. The second iteration results in a walk, $\boldsymbol{\theta}_2$, which averages 27.97 cm/s for a total improvement of 43.27% over $\boldsymbol{\theta}_0$.

Overall, improving the UNSW walk by over 40% shows that GAT can learn walk policies that outperform the fastest known stable walk for the NAO robot.

7.4.2.4 SimSpark to Gazebo Results:

Table 7.3 gives the average improvement in stable walk policies for each method and the number of trials in which a method failed to produce a stable improvement. Results show that GAT maximizes policy improvement in v while minimizing

[26] A video of the learned walk policies is available at https://www.cs.utexas.edu/users/AustinVilla/?p=research/real_and_sim_walk_learning.

Method	% Improve	Failures	Best Iteration
No Ground	11.094	7	1.33
NOISE-ENVELOPE	18.93	5	6.6
GAT	**22.48**	**1**	2.67

Table 7.3: This table compares the grounded action transformation algorithm (GAT) with baseline approaches for transferring learning between SimSpark and Gazebo. The first column displays the average maximum improvement found by each method after the first policy update made by CMA-ES. The second column is the number of times a method failed to find a stable walk. The third column gives the average iteration of CMA-ES when the best policy was found. No Ground refers to learning done in the unmodified simulator.

failure to transfer when transferring from a low-fidelity to high-fidelity simulator. NOISE-ENVELOPE improves upon no grounding in both improvement and number of iterations without improvement. Adding noise to the simulator encourages CMA-ES to propose robust policies which are more likely to be stable. However, GAT further improves over NOISE-ENVELOPE – demonstrating that action transformations are grounding the simulator in a more effective way than injecting noise.

Table 7.3 also shows that on average GAT finds an improved policy within the first few policy updates after grounding. When learning with no grounding finds an improvement it is also usually in an early iteration of CMA-ES. The grounding done by GAT is inherently local to the trajectory distribution of $\pi_{\boldsymbol{\theta}_0}$. Thus as $\pi_{\boldsymbol{\theta}}$ changes, the action transformation function fails to produce a more realistic simulator. As policy improvement progresses, the best policies in each CMA-ES iteration begin to over-fit to the dynamics of $\mathcal{M}_{\texttt{sim}}$. Without grounding over-fitting happens almost immediately. Noise modification methods can mitigate over-fitting by emphasizing robust policies although it is also limited in finding as strong of an improvement as GAT.

7.5 Summary

In this chapter, we have introduced an algorithm which allows a robot to learn a policy in a simulated environment and the resulting policy transfer to the physical robot. This algorithm, called the grounded action transformation algorithm, makes a contribution towards allowing reinforcement learning agents to leverage simulated data that is typically unusable in reinforcement learning. Giving agents the ability to learn skills in simulation with only small amounts of real world experience, greatly enhances the data efficiency of RL agents. This algorithm constitutes Contribution 5 of this dissertation.

We empirically evaluated GAT on three robot learning tasks using the NAO robot. In all cases, GAT leads to higher task performance compared to no grounding. We also compared GAT to a simulator randomization baseline and found that using real world data to modify the simulation was more effective than simply adding noise to the simulation. Finally, we applied GAT to optimizing the parameters of an existing walk controller and learned the fastest stable walk that we know of for the NAO robot.

Reinforcement learning algorithms struggle when the distribution of trajectories differs from that under the current policy and the environment of interest. In this Chapter, we introduced an algorithm for the setting when distribution of trajectories changes because the environment of interest is replaced with a simulator. This work complements the approaches of earlier chapters that dealt with distribution shift when the current policy is replaced with a different behavior policy. In the next chapter, we introduce algorithms that combine using both simulated and off-policy data. These algorithms constitute the final contributions of this dissertation.

Chapter 8

Combining Off-policy Data with Simulated Data

So far in this dissertation our discussion of policy value estimation has focused on minimal MSE estimates. In this chapter we will turn to a second objective for policy value estimation: computing confidence intervals for policy value estimates. We refer to this problem as the high confidence off-policy policy value estimation problem. Existing methods for this problem are based on importance sampling methods and tend to require large amounts of data to produce tight confidence intervals. In this chapter, we show how a combination of simulated and off-policy data can tighten confidence intervals for policy value estimation.

This chapter makes two contributions to using simulated data for computing confidence intervals for policy value estimation. First, we introduce a method for computing confidence intervals from the model-based estimator. The model-based estimator (introduced in Chapter 2) is a straightforward way to use simulated data to compute policy value estimates. However, how best to obtain confidence intervals for these estimates is still an open question. To address this question, we introduce a method that combines the model-based estimator with a statistical technique known

as bootstrapping. This new method, that we call MB-BOOTSTRAP, allows us to produce confidence intervals with the model-based estimator. MB-BOOTSTRAP is Contribution 6 of this dissertation and the first contribution of this chapter.

Unfortunately, model-based evaluation may be both biased and inconsistent, resulting in confidence intervals that are never sufficiently tight or fail to include $v(\pi_e)$. To avoid this problem, we turn to the doubly robust and weighted doubly robust estimators described in Chapter 2. These estimators allow us to use simulated data alongside importance-sampled off-policy data while remaining unbiased (in the case of doubly robust) and consistent (for both doubly robust and weighted doubly robust). In this chapter we combine these methods with bootstrap confidence intervals to address the high confidence off-policy value estimation problem. The resulting method, which we call WDR-BOOTSTRAP, is Contribution 7 of this dissertation.[27]

The algorithmic contributions of this chapter build on a previous approach of combining off-policy value estimators with statistical bootstrapping (Thomas et al., 2015b). The model-based estimator and weighted doubly robust estimator also come from earlier work in the literature. It is the combination of these lower variance off-policy value estimators with statistical bootstrapping that forms the contributions of this chapter. We discuss other alternative approaches to producing confidence intervals for off-policy policy value estimation in Section 9.4.

8.1 Confidence Intervals for Off-policy Value Estimation

Before introducing the contributions of this chapter, we first recall the general high confidence policy value estimation problem and discuss why it is challenging in

[27]This chapter contains work that was done in collaboration with Scott Niekum and previously published at AAMAS 2017 (Hanna et al., 2017a).

the off-policy setting. Throughout this chapter, we will focus on the problem of computing a lower bound on a policy value estimate. However, the methods we introduce are equally applicable for computing upper bounds or two-sided confidence intervals.

Recall from Chapter 2, that in the high confidence policy value estimation problem, we are given a set of m trajectories, $\mathcal{D} = \{(H_i, \pi_b)\}_{i=1}^m$, where $H_i \sim \pi_b$. We are also given an evaluation policy, π_e and a confidence level, $\delta \in [0, 1]$. Our objective is to determine a confidence lower bound, $v_\delta(\pi_e)$, on $v(\pi_e)$ such that $v_\delta(\pi_e) \leq v(\pi_e)$ with probability at least $1 - \delta$. The probabilistic lower bound means that if the lower bound was computed m times with m different realizations of $\mathcal{D}$, the expected number of times that $v_\delta(\pi_e) > v(\pi_e)$ is $m\delta$. Ideally, v_δ is *tight* or at least as close to $v(\pi_e)$ as possible while not exceeding the allowable δ error rate.

If π_b is distinct from π_e, the problem becomes the high confidence *off-policy* value estimation problem. Existing methods for this problem are based on the importance sampling estimator. Unfortunately, the high variance of importance sampling leads to these methods providing loose lower bounds (Thomas et al., 2015a). While Contributions 1 and 3 could potentially improve these methods, we focus here on how a combination of simulated and off-policy data leads to novel methods to address this problem.

8.2 Off-Policy Bootstrapped Lower Bounds

If we had access to the cumulative distribution function (CDF) of our policy value estimates, it would be straightforward to determine a lower bound: simply return the δ-percentile value of the distribution. Since we lack access to the CDF, we will instead estimate the distribution of our policy value estimates and use the estimated distribution to compute a confidence interval. We accomplish this objective with bootstrapping. Bootstrapping is a technique for estimating the distribution of a

statistic of interest (Efron, 1987). In this setting, our statistic of interest is an off-policy value estimate of $v(\pi_e)$.

We give pseudocode for a bootstrap lower bound method in Algorithm 6. We define **Off-PolicyEstimate** to be any method that takes a data set, $\mathcal{D} \coloneqq \{(H_i, \pi_b)\}_{i=1}^m$ and a policy, π_e, and returns a policy value estimate, $\hat{v}(\pi_e)$, (i.e., an off-policy estimator). The output of **Off-PolicyEstimate** is a statistic of $\mathcal{D}$ and we aim to estimate the distribution of **Off-PolicyEstimate**.

Bootstrapping estimates the distribution of **Off-PolicyEstimate** by creating b new sets of trajectories by sampling *with replacement* from the dataset $\mathcal{D}$ (Algorithm 6, Lines 1-2). We term these new trajectory sets *bootstrap datasets*. For each bootstrap dataset, we can use **Off-PolicyEstimate** to determine an estimate, $\hat{v}$, of $v(\pi_e)$ (Line 3). Since each bootstrap will contain different proportions of the original trajectories, this procedure produces a distribution over the value of $\hat{v}$. From this distribution we can estimate a $1 - \delta$ lower bound by taking the δ-percentile estimate (Line 6) after the $\hat{v}$s are sorted (Line 5). As the dataset, $\mathcal{D}$, grows, the distribution of $\hat{v}_j$ becomes a closer approximation of the distribution of **Off-PolicyEstimate** and our confidence intervals become more accurate.

Algorithm 6 is a general algorithm for off-policy bootstrap confidence intervals. We are not the first to consider the use of bootstrap confidence intervals for off-policy policy value estimation: Thomas et al. (2015b) use a variant of bootstrapping with importance sampling as **Off-PolicyEstimate**. Our contribution is to use learned models of the environment's transition function to produce tighter confidence intervals than these methods.

While bootstrapping has strong guarantees as $m \to \infty$, bootstrap confidence intervals lack finite sample guarantees. Using bootstrapping requires the assumption that the bootstrap distribution is representative of the distribution of the statistic of interest which may be false for a finite sample. Therefore, we characterize bootstrap

methods for producing confidence intervals as "approximate high-confidence" due to this possibly false assumption.[28] In contrast to lower bounds from concentration inequalities, bootstrapped lower bounds can be thought of as approximating the allowable δ error rate instead of upper bounding it. However, bootstrapping is considered safe enough for high risk medical predictions and in practice has a well-established record of producing accurate confidence intervals (Chambless et al., 2003).

Algorithm 6 Bootstrap Confidence Interval

Input is an evaluation policy π_e, a data set of trajectories, $\mathcal{D}$, a confidence level, $\delta \in [0, 1]$, and the required number of bootstrap estimates, b.

input π_e, $\mathcal{D}$, π_b, δ, b

output $1 - \delta$ confidence lower bound on $v(\pi_e)$.

1: **for all** $i \in [1, b]$ **do**
2: $\quad \tilde{\mathcal{D}}_i \leftarrow \{(H_1^i, \pi_b), \ldots, (H_n^i, \pi_b)\}$ where $H_j^i \sim \mathcal{U}(\mathcal{D})$ // where $\mathcal{U}(\mathcal{D})$ is a uniform distribution over trajectory-policy pairs in $\mathcal{D}$.
3: $\quad \hat{v}_i \leftarrow$ **Off-PolicyEstimate**$(\pi_e, \tilde{\mathcal{D}}_i, \pi_b)$
4: **end for**
5: `sort`$(\{\hat{v}_i | i \in [1, b]\})$ // Sort ascending
6: $l \leftarrow \lfloor \delta b \rfloor$
7: **Return** $\hat{v}_l$

8.2.1 Model-based Bootstrap

Our first approach to using simulated data for policy value confidence intervals is to directly use the model-based estimator as **Off-PolicyEstimate** in Algorithm 6. That is, for each bootstrap data set, we build a model, $\widehat{\mathcal{M}}$, of the environment MDP, $\mathcal{M}$. We then estimate $v(\pi_e, \widehat{\mathcal{M}})$. If the size of $\mathcal{S}$ and $\mathcal{A}$ are small enough, $v(\pi_e, \widehat{\mathcal{M}})$ can be computed exactly with value iteration. Otherwise, $v(\pi_e, \widehat{\mathcal{M}})$ can be estimated by simulating trajectories with π_e in $\widehat{\mathcal{M}}$. We call this method MB-BOOTSTRAP for *model-based bootstrap* and provide pseudocode in Algorithm 7.

[28] Thomas et al. (2015a) refer to such confidence intervals as "semi-safe."

Algorithm 7 MB-BOOTSTRAP **Confidence Interval**
Input is an evaluation policy π_e, a data set of trajectories, $\mathcal{D}$, a confidence level, $\delta \in [0, 1]$, and the required number of bootstrap estimates, b. **BuildModel** is a function that uses trajectories to construct a model of the underlying MDP.

input π_e, $\mathcal{D}$, π_b, δ, b

output $1 - \delta$ confidence lower bound on $v(\pi_e)$.

for all $i \in [1, b]$ **do**
  $\tilde{\mathcal{D}}_i \leftarrow \{(H_1^i, \pi_b), \ldots, (H_n^i, \pi_b)\}$ where $H_j^i \sim \mathcal{U}(\mathcal{D})$ // $\mathcal{U}(\mathcal{D})$ is a uniform distribution over trajectory-policy pairs in $\mathcal{D}$.
  $\widehat{\mathcal{M}} \leftarrow$ **BuildModel**$(\tilde{\mathcal{D}})$
  $\hat{v}_i \leftarrow v(\pi_e, \widehat{\mathcal{M}})$
end for
`sort`$(\{\hat{v}_i | i \in [1, b]\})$ // Sort ascending
$l \leftarrow \lfloor \delta b \rfloor$
Return $\hat{v}_l$

The main drawback to using MB-BOOTSTRAP is that the model-based estimator may be both biased and inconsistent. Thus even as the amount of available data grows, confidence intervals from MB-BOOSTRAP may remain too loose or, worse, have a higher error-rate than the specified δ level. However, when accurate models can be built, the low variance of the model-based estimator may lead to tighter confidence intervals than could be produced with other methods. In Section 8.3 we derive an upper bound on the difference between $v(\pi_e, \mathcal{M})$ and $v(\pi_e, \widehat{\mathcal{M}})$ that provides insight into settings where model bias may be high (and bounds from MB-BOOTSTRAP less trustworthy). In the following subsection we introduce our second contribution, WDR-BOOTSTRAP, that allows us to use simulated data while remaining free of model bias.

8.2.2 Weighted Doubly Robust Bootstrap

To bypass the lack of consistency of the model-based estimator, we now turn to the WDR estimator. We introduce a second algorithm that uses WDR as **Off-PolicyEstimate** in Algorithm 6. We call this new algorithm WDR-BOOTSTRAP

for *weighted doubly robust bootstrap* and provide pseudocode for it in Algorithm 8. This algorithm is Contribution 7 of this dissertation.

Algorithm 8 WDR-BOOTSTRAP **Confidence Interval**

Input is an evaluation policy π_e, a data set of trajectories, $\mathcal{D}$, a confidence level, $\delta \in [0, 1]$, and the required number of bootstrap estimates, b. **BuildModel** is a function that uses trajectories to construct a model of the underlying MDP.

input π_e, $\mathcal{D}$, π_b, δ, b

output $1 - \delta$ confidence lower bound on $v(\pi_e)$.

$\widehat{\mathcal{M}} \leftarrow$ **BuildModel**$(\mathcal{D})$

$\hat{q}^{\pi_e}, \hat{v}^{\pi_e} \leftarrow$ value functions of π_e in $\widehat{\mathcal{M}}$.

for all $i \in [1, b]$ **do**

  $\tilde{\mathcal{D}}_i \leftarrow \{(H_1^i, \pi_b), \ldots, (H_n^i, \pi_b)\}$ where $H_j^i \sim \mathcal{U}(\mathcal{D})$ // where $\mathcal{U}(\mathcal{D})$ is a uniform distribution over trajectory-policy pairs in $\mathcal{D}$.

  $\hat{v}_i \leftarrow$WDR$(\pi_e, \tilde{\mathcal{D}}_i, \widehat{\mathcal{M}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e})$

end for

`sort`$(\{\hat{v}_i | i \in [1, b]\})$ // Sort ascending

$l \leftarrow \lfloor \delta b \rfloor$

Return $\hat{v}_l$

In most ways, Algorithm 8 is similar to Algorithm 7. Aside from changing **Off-PolicyEstimate**, the other notable difference is that WDR-BOOTSTRAP only constructs a single model of the environment. We estimate the value functions $\hat{q}^{\pi_e}$ and $\hat{v}^{\pi_e}$ with the single model and then evaluate WDR with these value functions on the bootstrap datasets.

We could consider using the unweighted DR estimator as an alternative to WDR for combining off-policy and simulated data. We use WDR instead of DR because WDR typically has lower variance than DR and so can produce tighter confidence intervals. Though WDR is biased, it remains consistent and can thus leverage an inaccurate model to still produce tight and reliable confidence intervals.

8.3 Theoretical Analysis: When is Model Error High?

In this chapter, we are introducing two novel methods for determining confidence intervals for off-policy policy value estimation. The strength of WDR-BOOTSTRAP compared to MB-BOOTSTRAP is that it can still produce consistent confidence intervals even with an inaccurate model. However, it is an open question as to how to identify settings where model error is likely to be high and WDR-BOOTSTRAP is a more appropriate choice than MB-BOOTSTRAP. Towards answering this question, we now present a theoretical upper bound on the difference between $v(\pi_e, \widehat{\mathcal{M}})$ for a fixed model and $v(\pi_e, \mathcal{M})$.

Theorem 8.1 bounds the error of $v(\pi_e, \widehat{\mathcal{M}})$ produced by a fixed model, $\widehat{\mathcal{M}}$, as a function of the training error achieved when building $\widehat{\mathcal{M}}$. This bound provides insight into the settings in which MB-BOOTSTRAP is likely to be unsuccessful.

To prove this bound, we require the additional assumption that the reward function for $\mathcal{M}$ is known and bounded in $[0, r_{\texttt{max}}]$. We also will make use of the notation $P_{\mathcal{M}}$ and $d_{0,\mathcal{M}}$ to refer to the transition probabilities and initial state probabilities for MDP $\mathcal{M}$.

Theorem 8.1. *For* MDP $\mathcal{M}$*, any policies* π_e *and* π_b*, and an approximate model,* $\widehat{\mathcal{M}}$*, estimated with i.i.d. trajectories,* $H \sim \Pr(\cdot|\mathcal{M}, \pi_b)$*, the error in the model-based estimate of* $v(\pi_e, \mathcal{M})$ *with* $\widehat{\mathcal{M}}$*,* $v(\pi_e, \widehat{\mathcal{M}})$*, is upper bounded by:*

$$\left|v(\pi_e, \widehat{\mathcal{M}}) - v(\pi_e, \mathcal{M})\right| \leq 2\sqrt{2}L \cdot r_{\texttt{max}} \sqrt{\mathbf{E}\left[\rho^{(H)}_{L-1} \log \frac{\Pr(H|\pi_e, \mathcal{M})}{\Pr(H|\pi_e, \widehat{\mathcal{M}})} \middle| H \sim \pi_b\right]}$$

where $\rho^{(H)}_{L-1}$ *is the importance weight of trajectory* H *at step* L*.*

Proof. See Appendix E.1 for the full proof. □

The expectation in Theorem 8.1 is an importance-sampled *Kullback-Leibler* (KL) divergence. This expectation is thus a measure of similarity between the

distribution of trajectories in the real MDP versus the model with more weight placed on trajectories that are more likely under π_e than the behavior policy.

This result tells us that the error in a model-based estimate depends on how different the distribution of trajectories under the model is from the distribution of trajectories seen when executing π in the true MDP. Since most model building techniques (e.g., supervised learning algorithms, tabular methods) build the model from (s_t, a_t, s_{t+1}) transitions even if the transitions come from sampled trajectories (i.e., non-i.i.d. transitions), we express Theorem 8.1 in terms of transitions:

Corollary 8.1. *For* MDP, $\mathcal{M}$, *any policies* π_e *and* π_b *and an approximate model,* $\widehat{\mathcal{M}}$, *with transition probabilities,* $P_{\widehat{\mathcal{M}}}$, *estimated with trajectories* $H \sim \pi_b$, *the bias of the approximate model's estimate of* $v(\pi_e, \mathcal{M})$, $v(\pi_e, \widehat{\mathcal{M}})$, *is upper bounded by:*

$$|v(\pi_e, \widehat{\mathcal{M}}) - v(\pi_e, \mathcal{M})| \leq 2\sqrt{2}L \cdot r_{\texttt{max}} \sqrt{\epsilon_0 + \sum_{t=1}^{L-1} \mathbf{E}\left[\rho_t^{(H)} \epsilon(S_t, A_t) \middle| S_t, A_t \sim d^t_{\pi_b, \mathcal{M}}\right]}$$

where $d^t_{\pi_b, \mathcal{M}}$ *is the distribution of states and actions observed at time* t *when executing* π_b *in the true* MDP, $\epsilon_0 := D_{\texttt{KL}}(d_{0,\mathcal{M}} || d_{0,\widehat{\mathcal{M}}})$, *and* $\epsilon(s,a) = D_{\texttt{KL}}(P_{\mathcal{M}}(\cdot|s,a) || P_{\widehat{\mathcal{M}}}(\cdot|s,a)))$.

Proof. See Appendix E.1.3 for the proof of Corollary 8.1. □

Since P is unknown it is impossible to compute the $D_{\texttt{KL}}$ terms in Corollary 8.1. However, $D_{\texttt{KL}}$ can be approximated with two common supervised learning loss functions: negative log likelihood and cross-entropy. We can express Corollary 8.1 in terms of either negative log-likelihood (a regression loss function for continuous MDPs) or cross-entropy (a classification loss function for discrete MDPs) and minimize the bound with observed (s_t, a_t, s_{t+1}) transitions. In the case of discrete state-spaces this approximation upper bounds $D_{\texttt{KL}}$. In continuous state-spaces the approximation is correct within the average differential entropy of P which is a problem-specific

constant. Theorem 8.1 can be extended to finite sample bounds using Hoeffding's inequality:

Corollary 8.2. *For* MDP $\mathcal{M}$*, any policies* π_e *and* π_b *and an approximate model,* $\widehat{\mathcal{M}}$*, with transition probabilities,* $P_{\widehat{\mathcal{M}}}$*, estimated with* (s, a) *transitions from trajectories* $H \sim \pi_b$*, and after observing* m *trajectories then with probability* α*, the error of the approximate model's estimate of* $v(\pi_e, \mathcal{M})$*,* $v(\pi_e, \widehat{\mathcal{M}})$*, is upper bounded by:*

$$\left| v(\pi_e, \widehat{\mathcal{M}}) - v(\pi_e, \mathcal{M}) \right| \leq 2L \cdot r_{\texttt{max}} \cdot$$
$$\sqrt{2\bar{\rho}_{L-1}\sqrt{\frac{\ln(\frac{1}{\alpha})}{2m}} - \frac{1}{m}\sum_{j=1}^{m} \rho_{L-1}^{j} \left(\log d_{0,\widehat{\mathcal{M}}}(s_1^j) + \sum_{t=1}^{L-1} \log P_{\widehat{\mathcal{M}}}(s_{t+1}^j | s_t^j, a_t^j) \right)}$$

where $\bar{\rho}_{L-1}$ *is an upper bound on the importance ratio, i.e., for all* h*,* $\rho_{L-1}^{(h)} < \bar{\rho}_{L-1}$*.*

Proof. See Appendix E.2 for the proof of Corollary 8.2. □

Corollary 8.1 allows us to estimate the upper bound proposed in Theorem 8.1 while Corollary 8.2 allows us to upper bound our estimate of the upper bound. Given that we can estimate Corollary 8.1, we could estimate the bound and subtract it from the model-based estimate. This adjustment would prevent the model-based estimate from overestimating $v(\pi_e)$ which would in turn make sure the lower bound of MB-BOOTSTRAP did not overshoot $v(\pi_e)$. However in practice the dependence on the maximum reward makes the bound too loose to subtract off from the lower bound found by MB-BOOTSTRAP. Instead, we observe it characterizes settings where the MB estimator may exhibit high bias. Specifically, a MB estimate of $v(\pi_e)$ will have low error when we build a model which obtains low training error under the negative log-likelihood or cross-entropy loss functions where the error due to each (s_t, a_t, s_{t+1}) is importance-sampled to correct for the difference in distribution. This result holds regardless of whether or not the true transition dynamics are

Figure 8.1: Cliff World domain in which an agent (A) must move between or around cliffs to reach a goal (G).

representable by the model class. It is interesting to note that the largest importance weight upper bounds the variance of IS when returns are bounded. Here it upper bounds the error of MB when returns are bounded.

8.4 Empirical Analysis

We now present empirical evaluation of MB-BOOTSTRAP, WDR-BOOTSTRAP, and other bootstrapping off-policy methods across two policy value estimation tasks.

8.4.1 Experimental Set-up

Before presenting our empirical results, we briefly introduce the experimental set-up for each domain. Remaining details are included in Appendix F.4.

The first experimental domain is the discretized version of the Mountain Car task that we also used in Chapter 6. States are discretized horizontal position and velocity (for a total of 4292 states) and the agent may choose to accelerate left, right, or neither. We build tabular models which cannot generalize from the (s, a) pairs we observe in $\mathcal{D}$. We compute the model action value function, $\hat{q}^{\pi_e}$, and state value function, $\hat{v}^{\pi_e}$ with value-iteration for WDR. We use Monte Carlo rollouts to estimate $v(\pi_e, \widehat{\mathcal{M}})$ with MB.

Our second domain is a continuous two-dimensional Cliff World (depicted in

Figure 8.1) where a point mass agent navigates a series of cliffs to reach a goal state, $\mathbf{g}$. Domain dynamics are linear with additive Gaussian noise.

We build models in two ways: linear regression (converges to the true transition probabilities as $m \to \infty$) and regression over nonlinear polynomial basis functions.[29] The first model class choice represents the ideal case and the second is the case when the true dynamics are outside the learnable model class. Our results refer to WDR-BOOTSTRAP$^{\text{LR}}$ and WDR-BOOTSTRAP$^{\text{PR}}$ as the WDR estimator using linear regression and polynomial regression models respectively. Similarly, we evaluate MB-BOOTSTRAP$^{\text{LR}}$ and MB-BOOTSTRAP$^{\text{PR}}$. These dynamics mean that the bootstrap models of WDR-BOOTSTRAP $^{\text{LR}}$ and MB-BOOTSTRAP$^{\text{LR}}$ will quickly converge to a correct model as the amount of data increases since they build models with linear regression. On the other hand, these dynamics mean that the models of WDR-BOOTSTRAP$^{\text{PR}}$ and MB-BOOTSTRAP$^{\text{PR}}$ will quickly converge to an incorrect model since they use regression over nonlinear polynomial basis functions.

In each domain, we estimate a 95% confidence lower bound ($\delta = 0.05$) with our proposed methods and the importance sampling bootstrap methods from Thomas et al. (2015b).[30] To the best of our knowledge, these IS methods are the current state-of-the-art for approximate high confidence off-policy policy value estimation. We use $b = 2000$ bootstrap estimates, $\hat{v}_i$ and compute the true value of $v(\pi_e)$ with 1,000,000 Monte Carlo roll-outs of π_e in each domain.

For each domain we computed the lower bound for m trajectories with m varying logarithmically. For each m we sample a set of m trajectories with the behavior policy and compute the lower bound with each method on that set of trajectories.

[29] For each state feature, x, we include features 1, x^2, x^3 but not x.

[30] The importance sampling methods from Thomas et al. use a variant of bootstrapping known as bias-corrected and accelerated bootstrapping (BCa) which adjusts the distribution of the bootstrap statistic. This method is more complex than the simpler bootstrap variant we use and is suitable when the data distribution may be heavy-tailed, as is the case for importance sampling estimators.

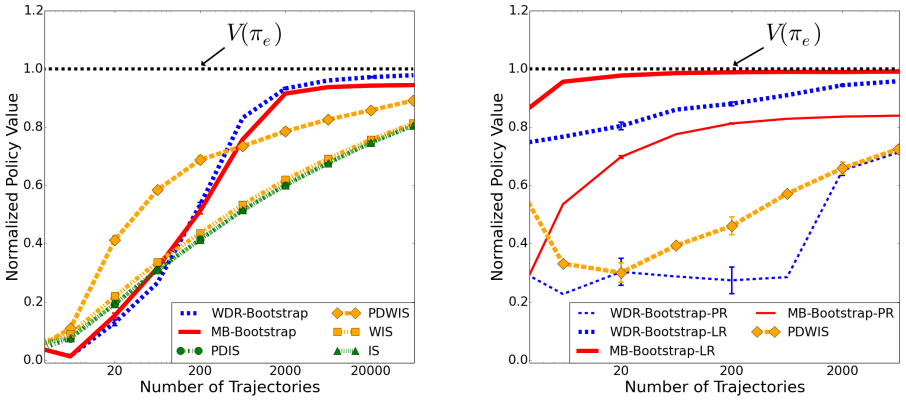
We run 400 trials for Mountain Car and 100 for Cliff World. The large number of trials is required for the empirical error rate calculations. When plotting the average lower bound across methods, we only average valid lower bounds (i.e., $\hat{v}_\delta(\pi_e) \leq v(\pi_e)$) because invalid lower bounds raise the average which can make a method appear to produce a tighter average lower bound when it fact it has a higher error rate.

8.4.2 Experimental Results

Figure 8.2 displays the average empirical 95% confidence lower bound found by each method in each domain. The ideal result is a lower bound, $v_\delta(\pi_e)$, that is as large as possible subject to $v_\delta(\pi_e) < v(\pi_e)$. Given that any statistically consistent method will achieve the ideal result as $m \to \infty$, our main point of comparison is which method gets closest the fastest. As a general trend we note that MB-BOOTSTRAP and WDR-BOOTSTRAP get closer to this ideal result with less data than all other methods. Figure 8.3 displays the empirical error rate for MB-BOOTSTRAP and WDR-BOOTSTRAP and shows that they approximate the allowable 5% error in each domain.

8.4.2.1 Mountain Car

In Mountain Car(Figure 8.2a), both WDR-BOOTSTRAP and MB-BOOTSTRAP outperform purely all IS methods (IS, WIS, PDIS, and PDWIS) in reaching the ideal result. We also note that both methods produce approximately the same average lower bound. The modeling assumption that lack of data for some (s, a) results in a transition to s is a form of negative model bias which lowers the performance of MB-BOOTSTRAP. Therefore, even though MB will eventually converge to $v(\pi_e)$ it does so no slower than WDR which can produce good estimates even when the model is inaccurate. This negative bias also leads to one importance sampling variant



(a) Mountain Car (b) Cliff World

Figure 8.2: The average empirical lower bound for the Mountain Car and Cliff World domains. Each plot displays the 95% lower bound on $v(\pi_e)$ computed by each method with varying amounts of trajectories. The ideal lower bound is just below the line labelled $v(\pi_e)$. Results demonstrate that the proposed model-based bootstrapping (MB-BOOTSTRAP) and weighted doubly robust bootstrapping (WDR-BOOTSTRAP) find a tighter lower bound with less data than previous importance sampling bootstrapping methods. For clarity, we omit IS, WIS and PDIS in Cliff World as they were outperformed by PDWIS. Error bars are for a 95% two-sided confidence interval.

(PDWIS) producing a tighter bound for small data sets although it is overtaken by MB-BOOTSTRAP and WDR-BOOTSTRAP as the amount of data increases.

Figure 8.3a shows that the MB-BOOTSTRAP and WDR-BOOTSTRAP error rate is much lower than the required error rate yet Figure 8.2a shows the lower bound is no looser. Since MB-BOOTSTRAP and WDR-BOOTSTRAP are low variance estimators, the average bound can be tight with a low error rate. It is also notable that since bootstrapping only approximates the 5% allowable error rate all methods can do worse than 5% when data is extremely sparse (only two trajectories).

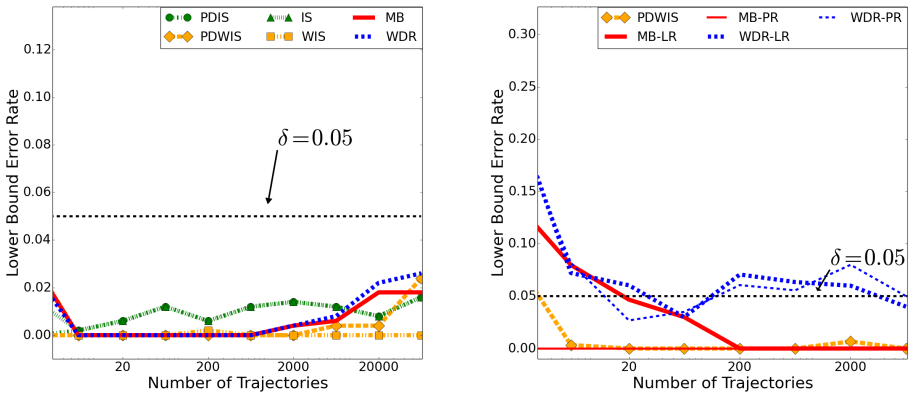
8.4.2.2 Cliff World

In Cliff World (Figure 8.2b), we first note that MB-BOOTSTRAP$^{\text{PR}}$ quickly converges to a suboptimal lower bound. In practice an incorrect model may lead to a bound that is too high (positive bias) or too loose (negative bias). Here, MB-BOOTSTRAP$^{\text{PR}}$ exhibits negative asymptotic bias and we converge to a bound that is too loose. MB-BOOTSTRAP$^{\text{LR}}$ with the correct model converges to a tight lower bound.

WDR-BOOTSTRAP is free of this asymptotic bias since it only uses the model as a control variate. Our theoretical results suggest MB error is high when evaluating π_e since the polynomial basis function models have high training error when errors are importance-sampled to correct for the off-policy model estimation. If we compute the bound in Section 8.3 and subtract the value off from the bound estimated by MB-BOOTSTRAP$^{\text{PR}}$ then the lower bound estimate will be unaffected by bias. Unfortunately, our theoretical bound (and other model-error bounds in earlier work) depends on the largest possible return, $L \cdot r_{\texttt{max}}$ and thus ensuring the model-based estimate lower-bounds $v(\pi_e)$ in this straightforward way reduces data-efficiency gains when bias may in fact be much lower.

The second notable trend is that WDR is also negatively impacted by the incorrect model. In Figure 8.2b we see that WDR-BOOTSTRAP$^{\text{LR}}$ (correct model) starts at a tight bound and increases from there. WDR-BOOTSTRAP$^{\text{PR}}$ with an incorrect model performs worse than PDWIS until larger m. Using an incorrect model with WDR decreases the variance of the PDWIS term less than the correct model would but we still expect less variance and a tighter lower bound than PDWIS by itself. One possibility is that error in the estimate of the model value functions coupled with the inaccurate model increases the variance of WDR.

We note WDR is particularly susceptible to error in continuous action settings. Recall that WDR requires a state value function, $\hat{v}^{\pi_e}$, such that $v^{\pi_e}(s,t) =$



(a) Mountain Car (b) Cliff World

Figure 8.3: Empirical error rate for the Mountain Car and Cliff World domains. The lower bound is computed k times for each method ($k = 400$ for Mountain Car, $k = 100$ for Cliff World) and we count how many times the lower bound is above the true $v(\pi_e)$. All methods correctly approximate the allowable 5% error rate for a 95% confidence lower bound.

$\sum_a \pi_e(a|s)\hat{q}^{\pi_e}(s,a,t)$. In continuous action settings the summation is replaced with an integral which may be analytically intractable unless $\hat{q}^{\pi_e}$ has a particular form.[31] In Cliff World, we did *not* have a $\hat{q}^{\pi_e}$ of one of these forms and we used Monte Carlo integration to approximate the integral. However, this approximation potentially introduced additional error into the estimates.

8.5 Summary

This chapter has introduced two novel bootstrapping method for approximate high confidence off-policy policy value estimation that combines both model-based simulation and direct use of off-policy data with importance sampling. We have shown on two empirical settings that these approaches lead to tighter confidence intervals for policy value estimation than existing methods that only use off-policy data with

[31] See Ciosek and Whiteson (2018) for a partial list of such forms.

importance sampling.

The first of these contributions, MB-BOOTSTRAP, uses the (likely) biased and statistically inconsistent model-based estimator. When model error is low, we can expect tight confidence intervals with this method. To better understand settings where model error may be high, we introduce a bound of the difference between $v(\pi_e, \mathcal{M})$ and $v(\pi_e, \widehat{\mathcal{M}})$ for a fixed model, $\widehat{\mathcal{M}}$. This bound characterizes settings where model error may be high and alternative confidence interval methods should be used. MB-BOOTSTRAP is Contribution 6 of this dissertation.

The second of this chapter's contributions, WDR-BOOTSTRAP, overcomes potential inconsistency that could arise from using a model. This method uses the WDR estimator which produces consistent estimates even when model error is high. Thus we can use simulated data without introducing model bias. The downside of WDR-BOOTSTRAP is the IS component of WDR increases variance which may result in looser confidence intervals even when model error is low. WDR-BOOTSTRAP is Contribution 7 of this dissertation.

These contributions complete the contributions of this dissertation.

Chapter 9

Related Work

The contributions of this dissertation build upon the contributions of many people. In this chapter we discuss these earlier works and how our contributions relate to them. Section 9.1 surveys work related to adaptive importance sampling for variance reduction in reinforcement learning. Section 9.2 surveys work related to regression importance sampling and reducing sampling error in reinforcement learning. Section 9.3 surveys work on leveraging simulation for reinforcement learning. Section 9.4 surveys work related to high confidence off-policy policy value estimation and (statistical) bootstrapping for RL. Finally, Section 9.5 discusses the problem of value-function learning that is related to policy value estimation.

9.1 Sampling Off-Policy Data

This section covers work related to behavior policy search and adaptive importance sampling for policy value estimation and policy improvement.

9.1.1 Adaptive Importance Sampling in Reinforcement Learning

Behavior policy search and BPG are closely related to existing work on adaptive importance-sampling. While adaptive importance-sampling has been studied in the Monte Carlo simulation literature, we focus here on adaptive importance-sampling for MDPs and Markov Reward Processes (i.e., Markov chains with rewards at each state). Existing work on adaptive IS in RL has considered changing the transition probabilities to lower the variance of policy evaluation (Desai and Glynn, 2001; Frank et al., 2008) or lower the variance of batch policy gradient estimates (Ciosek and Whiteson, 2017). Since the transition probabilities are typically uncontrollable in RL, adapting the behavior policy is a more general approach to adaptive IS in RL.

The cross-entropy method (CEM) is a general method for adaptive importance-sampling and could, in principle, be applied to reinforcement learning (Rubinstein, 1997). CEM attempts to minimize the Kullback-Leibler divergence between the current sampling distribution and the optimal sampling distribution. As discussed in Section 3.2, this optimal behavior policy only exists under a set of restrictive conditions. When the required conditions are met, a gradient-based version of CEM performs a similar behavior policy update as BPG (See Appendix C.2 for a proof). However, when the required conditions are *not* met, adapting the behavior policy by minimizing variance, as BPG does, is still applicable.

Aside from adaptive importance sampling, other methods exist for lowering the variance of on-policy estimates. Control variates (Zinkevich et al., 2006; White and Bowling, 2009), common random numbers, and antithetic variates (Veness et al., 2011) are other variance reduction techniques that have been applied to policy value estimation. These techniques require a model of the environment and do not appear to be applicable to the general RL policy value estimation problem. One place where these technique are applicable is Monte Carlo Tree Search (MCTS) where a simulator for the environment is typically available. We note that BPG could potentially be

applied to lower the variance of value estimates in MCTS.

9.1.2 Policy Improvement with Adaptive Importance Sampling

Chapter 4 examined how behavior policy search can be used to increase the data efficiency of batch policy gradient reinforcement learning. Previously, Bouchard et al. (2016) adapted the behavior policy to lower the variance of batch policy gradient estimates. Ciosek and Whiteson (2017) adapted the environment transition probabilities and correct with importance sampling in a way that lowers the variance of batch policy gradient estimates. Both these methods minimize the trace of the covariance matrix of the importance-sampled policy gradient. In contrast, we used the behavior policy gradient algorithm to lower the variance of the importance-sampled return.

9.2 Weighting Off-Policy Data

In this section, we survey work related to importance sampling with an estimated behavior policy. We also discuss work related to the problem of eliminating sampling error in reinforcement learning.

9.2.1 Importance Sampling with an Estimated Behavior Policy

The regression importance sampling estimator introduced in Chapter 5 replaces the true behavior policy with its empirical estimate. Earlier work has studied many different variants of this approach. We divide earlier work into methods that show estimating the behavior policy leads to more accurate estimates than using the true behavior policy and methods that seemingly show that estimating the behavior policy leads to less accurate estimates. For the former, we will discuss how the contribution of Chapter 5 is different than these earlier works. For the latter, we will discuss how research showing that estimating the behavior policy leads to less accurate estimates

does *not* contradict our findings that regression importance sampling has lower MSE than that of ordinary importance sampling.

9.2.1.1 Estimating the Behavior Policy is Better

A number of research works have shown that using the empirical behavior policy improves importance sampling relative to using the true behavior policy. To the best of our knowledge, all such work has been done in the multi-armed bandit, contextual bandit, or causal inference communities. One can directly extend these methods to MDPs by estimating the empirical distribution of trajectories instead of the empirical distribution of actions. Unfortunately, such a method is impractical as it requires knowing the probability of a trajectory under π_e which requires knowing the state transition probabilities. In Appendix D.3 we show that our RIS($L-1$) estimator can be viewed as an approximation of such a method. Under this view, the more straightforward RIS(0) estimator is a greater departure from methods that have appeared previously in the literature.

Our work took inspiration from Li et al. (2015) who prove, for contextless bandits, that using the empirical behavior policy has lower minimax mean squared error than using the true behavior policy. They corroborate these theoretical findings with experiments showing that the mean squared error of the so-called REG estimator decreases faster than that of OIS. The main distinction between this work and Chapter 5 is that we are interested in policy value estimation for full MDPs where actions affect both reward and the next state. Our theoretical results are only concerned with the asymptotic sample size while Li et al. (2015) provide results for finite sample sizes.

For contextual bandits, Narita et al. (2019) prove that importance sampling with the empirical policy minimizes asymptotic variance among all asymptotically normal estimators (including ordinary importance sampling). They also provide a

large-scale study of policy value estimation with the empirical behavior policy on an ad-placement task. Xie et al. (2018) provide similar results and prove a reduction in finite-sample mean squared error when using the empirical behavior policy. Again, our work differs from these two works in that we are concerned with full MDPs.

It has long been known in the causal inference literature that the empirical behavior policy outperforms using the true behavior policy. In this literature, the behavior policy action probabilities are known as *propensities* and importance sampling is known as *inverse propensity scoring* (Austin, 2011). Rosenbaum (1987) first introduced this approach using parametric propensity estimates. In later work, Hirano et al. (2003) studied this approach using non-parametric propensity score estimates. The causal inference problems studied can be viewed as a restricted class of contextual bandit problems. Under that view, our work differs from these earlier studies in that we are concerned with full MDPs.

Importance sampling is commonly defined as a way to use samples from a *proposal* distribution to estimate an expectation under a *target* distribution. Henmi et al. (2007) proved that importance sampling with a maximum likelihood parametric estimate of the proposal distribution had lower asymptotic variance than using the true proposal distribution. Our asymptotic variance analysis of regression importance sampling is a corollary to this theoretical result. Delyon and Portier (2016) proved the benefit of using a non-parametric estimate of the proposal distribution.

Other works have explored directly estimating the importance weights instead of first estimating the proposal distribution (i.e., behavior policy) to compute the importance weights (Oates et al., 2017; Liu and Lee, 2017). These "blackbox" importance sampling approaches show superior convergence rates compared to ordinary importance sampling. These methods have also, to the best of our knowledge, *not* been studied in Markov decision processes or for sequential data.

Finally, a few works have used the estimated behavior policy to smooth

importance weights when the data has been generated by a set of behavior policies with limited support. Gruslys et al. (2017) use an empirical estimate of the behavior policy for an off-policy actor-critic algorithm. Levine and Koltun (2013) combine a set of local policies learned with LQR into a global policy and then use this global policy as the behavior policy to importance sample data for a batch policy gradient algorithm. Strehl et al. (2010) estimate the behavior policy when the data was generated by a set of deterministic behavior policies. Our work differs from all of these approaches in that we use the empirical behavior policy in order to reduce sampling error while these works use the empirical behavior policy to smooth importance weights.

9.2.1.2 Estimating the Behavior Policy is Worse

In the literature, it is *not* always the case that using an estimated behavior policy improves importance sampling. Here, we discuss this work and why it does *not* contradict our findings that estimating a behavior policy can improve importance sampling.

In contextual bandit problems, Dudík et al. (2011) present theoretical results showing that an estimated behavior policy may increase the variance of importance sampling while also introducing bias. Farajtabar et al. (2018) prove similar results for full MDPs. However, in these works the behavior policy is estimated with a *separate* set of data than the set used for computing the off-policy value estimate. Because the behavior policy is estimated with a separate set of data it has no power to correct sampling error in the data used for the off-policy value estimate. In fact, these theoretical findings are in line with our experiments showing that it is important to use the same set of data both to estimate the behavior policy and to compute the regression importance sampling estimate (see Figures 5.4b, 5.4d, 5.6b in Chapter 5).

Raghu et al. (2018) report that larger differences between the true behavior

policy and estimated behavior policy lead to more error in the off-policy value estimate. However, they measure off-policy value estimation error with respect to the true behavior policy weighted importance sampling estimate and so it is *not* surprising that as the policies become more different the error increases.

Finally, we note that in settings where the behavior policy is unknown, or is a complex function of the state, more work is needed with regards to how to select the right policy class for regression importance sampling or what it means to avoid over-fitting when estimating the behavior policy. While these questions are open, it is possible that more research may show the empirical behavior policy is worse than the true behavior policy even when using regression importance sampling.

9.2.2 Exact Expectation Methods

In Chapter 5 and 6 we have used importance sampling with an estimated behavior policy to correct sampling error in reinforcement learning. Here, we discuss alternative approaches that avoid sampling error altogether.

The SARSA algorithm (Rummery and Niranjan, 1994) uses (S, A, R, S', A') tuples to learn an estimate of q^{π_e} with the update:

$$q^{\pi_e}(S, A, t) \leftarrow q^{\pi_e}(S, A, t) + \alpha(R + q^{\pi_e}(S', A', t+1) - q^{\pi_e}(S, A, t)).$$

This update requires two sampled actions. Sampling error due to double sampling in the action-space can be reduced with the *expected* SARSA update (Van Seijen et al., 2009):

$$q^{\pi_e}(S, A, t) \leftarrow q^{\pi_e}(S, A, t) + \alpha(R + \sum_{a \in \mathcal{A}} \pi_e(a|S') q^{\pi_e}(S', a, t+1) - q^{\pi_e}(S, A, t)).$$

Expected SARSA requires either a small discrete action-space or for π_e and q^{π_e} to have forms that allow analytic integration. Regression importance sampling is

applicable when these conditions fail to hold.

Expected SARSA can be extended to a *multi-step* algorithm with the tree-backup algorithm (Precup et al., 2000; Sutton and Barto, 1998). More recent work has shown that the amount of sampling as opposed to exact expectations can be done on a per-state basis using the $Q(\sigma)$ algorithm (Asis et al., 2018). Other tree-backup-like algorithms have been proposed and hold the promise to eliminate sampling error in off-policy data (Yang et al., 2018; Shi et al., 2019). Like expected SARSA, these algorithms require the ability to compute the sum of $\pi_e(a|s)q^{\pi_e}(s,a,t)$ over all $a \in \mathcal{A}$.

In policy gradient reinforcement learning, Sutton et al. (2000b) introduced the *all-actions policy gradient* algorithm that avoids sampling in the action-space by analytically computing the expectation of $q^{\pi_{\boldsymbol{\theta}}}(s,a)\frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)$. This approach has been further developed as the *expected policy gradient* algorithm (Ciosek and Whiteson, 2018; Fellows et al., 2018), the *mean actor-critic* algorithm (Asadi et al., 2017), and the *MC-256* algorithm (Petit et al., 2019). With a good approximation of q^π, these algorithms learn faster than a batch Monte Carlo policy gradient estimator. However, requiring a good approximation of q^π undercuts one of the primary reasons for using policy gradient RL: it may be easier to represent a good policy than to represent the correct action-value function (Sutton and Barto, 1998). The sampling error corrected policy gradient estimator provides an alternative method for reducing sampling error when q^π is difficult to learn. We also note that estimating π (as the sampling error corrected policy gradient estimator does) may be easier than estimating q^π since the right function approximator class for π is known while, in general, it is unknown for q^π.

9.3 Learning with Simulated Data

The challenge of transferring learned policies from simulation to reality has received much research attention of late. This section surveys this recent work as well as older research in simulation-transfer methods. We note that our work also relates to model-based reinforcement learning. However, much of model-based reinforcement learning focuses on learning a simulator for the target task MDP (often from scratch) while we focus on settings where an inaccurate simulator is available a priori.

We divide the sim-to-real literature into four categories: simulator modification, simulator randomization or simulator ensembles, simulators as prior knowledge, and sim-to-real perception learning.

9.3.1 Simulator Modification

We classify sim-to-real works that attempt to use real world experience to change the simulator as simulator modification approaches. This category of work is the category most similar to Chapter 7 and the grounded action transformation (GAT) algorithm.

Abbeel et al. (2006) use real-world experience to modify an inaccurate model of a deterministic MDP. The real-world experience is used to modify $P_{\texttt{sim}}$ so that the policy gradient in simulation is the same as the policy gradient in the real world. Cutler et al. (2014) use lower fidelity simulators to narrow the action search space for faster learning in higher fidelity simulators or the real world. This work also uses experience in higher fidelity simulators to make lower fidelity simulators more realistic. Both these methods assume *random access modification* – the ability to arbitrarily modify the simulated dynamics of any state-action pair. This assumption is restrictive in that it may be false for many simulators especially for real-valued states and actions.

Other work has used real world data to modify simulator parameters (e.g.,

coefficients of friction) (Zhu et al., 2018) or combined simulation with Gaussian processes to model where real world data has not been observed (Lee et al., 2017). Such approaches may extrapolate better to new parts of the state-space, however, they may fail if no setting of the physics parameters can capture the complexity of the real world. Golemo et al. (2018) train recurrent neural network to predict differences between simulation and reality. Then, following actions in simulation, the resulting next state is corrected to be closer to what it would be in the real world. This approach requires the ability to directly set the state of the simulator which is a requirement we avoid with GAT.

Manual parameter tuning is another form of simulator modification that can be done prior to applying reinforcement learning. Lowrey et al. (2018) carefully identify simulation parameters before applying policy gradient reinforcement learning to learn to push an object to target positions. Tan et al. (2018) perform similar system identification (including disassembling the robot and making measurements of each part) and adding action latency modeling before using deep reinforcement learning to learn quadrapedal walking. In contrast to these approaches, the GAT algorithm takes a data-driven approach to modifying the simulator without the need for expert system identification.

Finally, while most approaches to simulator modification involve correcting the simulator dynamics, other approaches attempt to directly correct $v(\pi, \mathcal{M}_{\texttt{sim}})$. Assuming $v(\pi, \mathcal{M}) = v(\pi, \mathcal{M}_{\texttt{sim}}) + \epsilon(\pi)$, Iocchi et al. (2007) attempt to learn $\epsilon(\pi)$ for any π. Then policy search can be done directly on $v(\pi, \mathcal{M}_{\texttt{sim}}) + \epsilon(\pi)$ without needing to evaluate $v(\pi, \mathcal{M})$. Rodriguez et al. (2019) introduce a similar approach except they take into account uncertainty in extrapolating the estimate of $\epsilon(\pi)$ and use Bayesian optimization for policy learning. Like our work in Chapter 7, both of these works apply their techniques to bipedal locomotion. Koos et al. (2010) use multi-objective optimization to find policies that trade off between optimizing

$v(\pi, \mathcal{M}_{\texttt{sim}})$ and a measure of how likely π is to transfer to the real world.

9.3.2 Robustness through Simulator Variance

Another class of sim-to-real approaches is methods that attempt to cross the reality gap by learning robust policies that can work in different variants of the simulated environment. The key idea is that if a learned policy can work in different simulations then it is more likely to be able to perform well in the real world. The simplest instantiation of this idea is to inject noise into the robot's actions or sensors (Jakobi et al., 1995; Miglino et al., 1996) or to randomize the simulator parameters (Peng et al., 2017; Molchanov et al., 2019; OpenAI et al., 2018). Unlike data driven approaches (such as GAT), such *domain randomization* approaches learn policies that are robust enough to cross the reality gap but may give up some ability to exploit the target real world environment.

A number of works have attempted to combine domain randomization and real world data to adapt the simulator. Chebotar et al. (2019) randomize simulation parameters and use real world data to update the distribution over simulation parameters while simulatenously learning robotic manipulation tasks. A similar approach is taken by Ramos et al. (2019). Muratore et al. (2018) attempt to use real world data to predict transferrability of policies learned in a randomized simulation. Mozifian et al. (2019) attempt to maintain a wide distribution over simulator parameters while ensuring the distribution is narrow enough to allow reinforcement learning to exploit instances that are most similar to the real world.

Domain randomization is used to learn policies that are robust enough to transfer to the real world. An alternative approach that does not involve randomness is to learn policies that perform well under an ensemble of different simulators (Boeing and Bräunl, 2012; Rajeswaran et al., 2017; Lowrey et al., 2018). Pinto et al. (2017b) simultaneously learn an adversary that can perturb the learning agent's

actions while it learns in simulation. The learner must learn a policy that is robust to disturbances and then will perform better when transferred to the real world.

9.3.3 Simulator as Prior Knowledge

Another approach to sim-to-real learning is to use experience in simulation to reduce learning time on the physical robot. Cully et al. (2015) use a simulator to estimate fitness values for low-dimensional robot behaviors which gives the robot prior knowledge of how to adapt its behavior if it becomes damaged during real world operation. Cutler and How (2015) use experience in simulation to estimate a prior for a Gaussian process model to be used with the PILCO (Deisenroth and Rasmussen, 2011) learning algorithm. Rusu et al. (2016a; 2016b) introduce progressive neural network policies which are initially trained in simulation before a final period of learning in the true environment. Christiano et al. (2016) turn simulation policies into real world policies by transforming policy actions so that they produce the same effect that they did in simulation. Marco et al. (2017) use simulation to reduce the number of policy evaluations needed for Bayesian optimization of task performance. In principle, GAT could be used with any of these approaches to correct the simulator dynamics which would lead to a more accurate prior.

9.3.4 Reality Gap in the Observation Space

Finally, while we focus on the reality gap due to differences in simulated and real world dynamics, much recent work has focused on transfer from simulation to reality when the policy maps images to actions. In this setting, even if P and $P_{\texttt{sim}}$ are identical, policies may fail when transferred to the real world due to the differences between real and rendered images. Domain randomization is a popular technique for handling this problem. Unlike the dynamics randomization techniques discussed above, in this setting domain randomization means randomizing features of the

simulator's rendered images (Sadeghi and Levine, 2017; Tobin et al., 2017, 2018; Pinto et al., 2017a). This approach is useful in that it forces deep reinforcement learning algorithms to learn representations that focus on higher level properties of a task and not low-level details of image appearance.

Computer vision domain adaptation methods can also be used to overcome the problem of differing observation spaces (Fang et al., 2018; Tzeng et al., 2016; Bousmalis et al., 2018; James et al., 2019). A final approach is to learn perception and control separately so that the real world perception system is only trained with real world images (Zhang et al., 2016; Devin et al., 2017).

9.4 Combining Simulation and Importance Sampling

Chapter 8 introduced two algorithms for approximate high confidence off-policy policy value estimation method that uses a control variate and statistical bootstrapping. This section surveys work related to high confidence off-policy policy value estimation and statistical bootstrapping.

9.4.1 High Confidence Off-Policy Policy Value Estimation

Concentration inequalities have been used with importance-sampled returns for lower bounds on off-policy estimates (Thomas et al., 2015a). The concentration inequality approach is notable in that it produces a true probabilistic bound on the policy performance. A similar method was proposed by Bottou et al. (2013) who clip importance weights to lower the variance of the importance sampling estimator. Unfortunately, these approaches require prohibitive amounts of data and were shown to be far less data-efficient than bootstrapping with importance sampling (Thomas et al., 2015b; Thomas, 2015).

Jiang and Li (2016) evaluated the doubly robust estimator for safe-policy improvement. They compute confidence intervals with a method similar to the

Student's t-test confidence interval shown to be less data-efficient than bootstrapping (Thomas et al., 2015b). Methods using t-test confidence intervals are, like statistical bootstrapping, only "semi-safe" because they make a possibly false assumption that off-policy corrected returns are normally distributed. Cohen et al. (2018) also use t-test confidence intervals for safe policy improvement.

Chow et al. (2015) and Ghavamzadeh et al. (2016) use ideas from robust optimization to derive model-based lower bounds on $v(\pi_e)$. These bound are computable only if the error in each transition can be bounded and is inapplicable for estimating bias in continuous state-spaces. Model-based PAC MDP methods can be used to synthesize policies which are approximately optimal with high probability (Fu and Topcu, 2014). These methods are only applicable to discrete MDPs and require large amounts of data. In contrast, MB-BOOTSTRAP and WDR-BOOTSTRAP are applicable to continuous or discrete MDPs.

Outside of policy value estimation, Brown and Niekum (2018) introduce a high confidence method for *inverse* reinforcement learning. In the inverse reinforcement learning setting, the agent attempts to infer the reward function with respect to which an expert demonstrator was acting optimally. Letting $\hat{\pi}^\star$ be the policy that is optimal with respect to the estimated reward $\hat{r}$ and π_d be the expert demonstrator's policy, Brown and Niekum (2018) introduce an algorithm that ensures $v(\hat{\pi}^\star)$ is within ϵ of $v(\pi_d)$.

9.4.2 Bootstrapping in Reinforcement Learning

Other previous work has used *statistical* bootstrapping to handle uncertainty in RL. One of the primary reasons to represent uncertainty with bootstrapping is for exploration. The TEXPLORE algorithm learns multiple decision tree models from subsets of experience to represent uncertainty in model predictions (Hester and Stone, 2010). Osband et al. (2016) use bootstrap datasets to train multiple

heads for a deep Q-Network Q-value estimator. Each head gives a different Q-value estimate which serves as a measure of uncertainty used to guide exploration. In this dissertation we use bootstrapping for high confidence off-policy value estimation instead of exploration.

White and White (2010) use time-series bootstrapping to place confidence intervals on value-function estimation during policy learning. Thomas and Brunskill (2016a) introduce an estimate of the model-based estimator's bias using a combination of WDR and bootstrapping. Chua et al. (2018) build different models from bootstrap data sets to represent uncertainty in the environment's transition probabilities. While these methods are related through the combination of bootstrapping and RL, they do *not* address the problem of confidence intervals for off-policy policy value estimation that we address in Chapter 8.

9.5 Policy Evaluation vs. Policy Value Estimation

Chapters 3 and 5 addressed the policy value estimation problem. The policy value estimation problem is closely related to the problem of policy evaluation and so we discuss the related policy evaluation literature here.

In the policy evaluation problem, we are given an evaluation policy π_e and tasked with estimating the value function $v^{\pi_e}(s,t) \coloneqq \mathbf{E}[\sum_{j=t}^{L} R_j | S_t = s, A_j \sim \pi_e]$ for all states s (Sutton and Barto, 1998). Denoting the estimated value function as $\hat{v}^{\pi_e}$, we seek to minimize the mean squared *value* error of this estimate:

$$\sum_{s \in \mathcal{S}} \mu(s)(\hat{v}^{\pi_e}(s,\cdot) - v^{\pi_e}(s,\cdot))^2,$$

where $\mu(s) \geq 0, \sum_{s \in \mathcal{S}} \mu(s) = 1$ is a measure of how much we care about the error in state s (Sutton and Barto, 1998). In contrast, in the policy value estimation problem,

we estimate a scalar estimate, $\hat{v}$, and desire low squared error for this estimate:

$$(\hat{v} - v(\pi_e))^2.$$

where the true value of the policy, $v(\pi)$, is equal to the expectation of the true value function $v^{\pi_e}(S, 0)$ under the initial state distribution d_0.

As in policy value estimation, we require data from some policy in order to estimate v^{π_e}. When the data comes from π_e, the problem is on-policy policy evaluation. When the data comes from a different policy, the problem is off-policy policy evaluation. Importance sampling is a widely used technique for correcting off-policy data in policy evaluation (Precup et al., 2000; Munos et al., 2016). In Chapter 10, we will discuss extending our policy value estimation work on importance sampling to the policy evaluation setting.

Policy evaluation introduces other challenges not present in the policy value estimation problem. First, the need to estimate $v^{\pi_e}(s, t)$ for a possibly infinite number of states necessitates function approximation to represent v^{π_e}. Second, value function learning admits the possibility of using intermediate estimates of $v^{\pi_e}(s, t)$ for updating the value of other states, a process known as bootstrapping in the RL community. Function approximation, bootstrapping, and off-policy learning are a *deadly triad* that, in combination, can produce unstable and inconsistent estimates of v^{π_e}(Baird, 1995; Sutton and Barto, 1998). Much work has been done on finding stable value function estimators that can leverage all three of these techniques (e.g., Baird (1995); Sutton et al. (2016); Mahmood et al. (2017)). To the best of our knowledge, neither improving the behavior policy or estimating the behavior policy has been studied for policy evaluation. In Chapter 10, we discuss these combinations as interesting directions for future work.

9.6 Summary

This chapter has surveyed work related to the topics of this dissertation with the intent of placing our contributions within the existing literature. While much more could be said about work that has been done in off-policy reinforcement learning and the use of simulation in robotics, we focused on work related to the algorithms introduced in this dissertation.

Chapter 10

Conclusion and Future Work

Solutions to many real world problems involve taking sequences of actions to achieve long-term goals. The reinforcement learning problem is an effective way to model such settings where an intelligent agent interacts with a task. Furthermore, reinforcement learning algorithms have led to stronger empirical performance than hand-designed controllers, policies from classical control, or policies learned with supervised learning. Unfortunately, many algorithms capable of learning strong policies require large amounts of domain experience. As long as this fact remains true, it will be difficult for such reinforcement learning algorithms to be widely applied.

Part of this data-inefficiency problem is that reinforcement learning algorithms are sensitive to the distribution of data they observe during learning. Experience is best when it comes from the current policy and is generated in the environment of interest. Experience from off-policy or simulated data is generally harder for an RL agent to use. Allowing RL agents to learn from off-policy or simulated data would make learning more efficient. Towards this goal, this dissertation makes several contributions towards answering the following question for finite-horizon, episodic reinforcement learning problems:

How can a reinforcement learning agent leverage off-policy and simulated data to evaluate and improve upon the expected performance of a policy?

Towards our big question, we first asked the smaller question of how should an RL agent *collect* off-policy data for low variance importance sampling? In Chapter 3, we focused on the problem of policy value improvement and asked how to collect off-policy data so that importance sampling policy value estimation has low variance. We introduced the behavior policy search problem and an algorithm that addresses this problem for the reinforcement learning sub-problem of policy value estimation. Furthermore, in Chapter 4, we demonstrated the utility of behavior policy search for policy improvement.

We next asked the question of how should an RL agent *weight* experience it has collected from another policy (i.e., off-policy data) to correct for distribution shift in the action selection? Again, we focused on the off-policy technique of importance sampling and showed that the common ordinary importance sampling weights are sub-optimal (Chapter 5). These weights require using the true data collecting behavior policy probabilities $\pi_b(a|s)$. We showed that replacing the true behavior policy probabilities with their empirical estimate results in more accurate off-policy policy value estimation. We also showed, in Chapter 6, that a similar approach leads to more accurate policy improvement.

In addition to off-policy learning, this dissertation also considered how an RL agent can use simulated data when a domain simulator is available a priori (Chapter 7). Though the underlying problem of distribution shift is the same, for simulated data, importance sampling cannot be used because the transition function is unknown. Instead, we introduced an algorithm that allows a small amount of real world data to be used to first correct the simulator before reinforcement learning is done in simulation. We demonstrated the utility of this algorithm on three robot learning tasks – including a walking experiment that leads to the fastest walking controller we know of on the NAO robot.

Finally, we combined the use of off-policy and simulated data to produce more data efficient algorithms for the problem of high confidence policy value estimation (Chapter 8). In this problem the goal is to produce a tight estimated confidence interval on the value of an untested policy. Importance-sampling-based techniques for this problem lead to confidence intervals that are too loose. We showed how learned simulators can be used for this problem and also how learned simulators can be combined with off-policy data to retain theoretical advantages of importance-sampling-based confidence interval methods. We introduced two algorithms for the approximate high confidence policy value estimation problem and demonstrated empirically that they provide tighter confidence bounds than existing techniques that only use off-policy data.

10.1 Contributions

This dissertation makes the following contributions to the reinforcement learning literature.

1. In Chapter 3, we introduced the behavior policy search problem for finding a behavior policy that produces data for low variance importance sampling policy value estimation. We also introduced the behavior policy gradient algorithm as one solution to this problem and provided an empirical study of behavior policy search with the behavior policy gradient algorithm.

2. In Chapter 4, we studied the application of behavior policy search to policy improvement. We demonstrated empirically the potential for using an improved behavior policy for more efficient off-policy learning. Finally, we identified directions for further study of simultaneous behavior and target policy learning.

3. In Chapter 5, we introduced a family of regression importance sampling estimators that replace the true behavior policy used in importance sampling

with the empirical estimate of the behavior policy. We proved that all members of this family have asymptotically lower or equal variance than using the true behavior policy and we demonstrated empirically that regression importance sampling improves off-policy policy value estimation compared to ordinary importance sampling.

4. In Chapter 6, we introduced the sampling error corrected (SEC) policy gradient estimator for batch policy gradient reinforcement learning. Under a set of limiting assumptions we proved that SEC has variance less than or equal to that of the batch Monte Carlo policy gradient estimator. We also conducted an empirical study of learning with SEC and showed faster convergence to the optimum policy when using SEC as opposed to the batch Monte Carlo policy gradient estimator.

5. In Chapter 7, we introduced the grounded action transformation (GAT) algorithm that uses small amounts of real world data to modify a simulated environment so that the simulated environment can be used for reinforcement learning. We applied GAT to three robot learning tasks and showed it allowed skills learned entirely in simulation to transfer to the physical robot. We also used GAT to optimize the parameters of a state-of-the-art robot walking controller which led to the fastest stable walking controller for the NAO robot.

6. In Chapter 8, we introduced the model-based bootstrap (MB-BOOTSTRAP) algorithm for approximate high confidence policy value estimation. We derived an upper bound on the error in model-based estimates of a policy's value. We demonstrated empirically that MB-BOOTSTRAP produces tighter confidence interval estimates than importance-sampling-based methods.

7. Also in Chapter 8, we introduced the weighted doubly robust bootstrap (WDR-BOOTSTRAP) algorithm for approximate high confidence policy value

estimation. We demonstrated empirically that WDR-BOOTSTRAP produces tighter confidence interval estimates than importance-sampling-based methods while retaining provable consistency.

10.2 Future Work

The work of this dissertation has answered many smaller questions towards answering the thesis question of how a reinforcement learning agent can leverage off-policy and simulated data for more efficient policy value estimation and policy improvement. It has also raised many new questions for future research. In this section, we discuss these questions.

As an overarching direction for future work, we note that the contributions made in this dissertation have been made in a particular setting of the reinforcement learning problem. In particular, new algorithms, theoretical results, and empirical studies made the assumption that tasks are modeled as episodic, finite-horizon, fully observable Markov decision processes. Extending algorithms and results to the continuing, infinite-horizon, and partially observable settings is an important direction for future work. We also note that our contributions pertaining to policy gradient RL (Contributions 2 and 4) studied *batch* policy gradient RL. Extending results to the non-batch setting is another important direction for future work.

10.2.1 Sampling Off-Policy Data

Chapters 3 and 4 discussed how a learning agent should collect data for more efficient policy value estimation and policy improvement. The focus of these chapters was on using importance sampling to lower the variance of off-policy policy value estimation. Here we discuss connections to regression importance sampling (Chapter 5), consider extensions to value function learning, further discuss using behavior policy search for policy improvement, and provide some remaining theoretical questions concerning

behavior policy search.

10.2.1.1 Exploration for Policy Evaluation

Chapter 3 introduced the behavior policy gradient algorithm that provides low variance importance sampling estimates. This algorithm assumes the true behavior policy action probabilities are used in the importance sampling estimate. Chapter 5 introduced regression importance sampling (RIS) and showed that an estimate of the behavior policy yields a stronger importance sampling method compared to using the true behavior policy. This observation raises the question of "how should we perform behavior policy search for a RIS estimate of $v(\pi_e)$?"

The bandit setting is illustrative for showing that a good behavior policy for ordinary importance sampling may be sub-optimal for regression importance sampling. Consider a k-armed bandit with deterministic rewards on each arm. After all k arms have been observed, the RIS estimate will have both zero bias and zero variance.[32] Thus the optimal behavior policy for RIS should increase the probability of unobserved actions; it is a non-stationary policy that depends on all of the past actions.

In contrast, as shown in Chapter 3, the optimal behavior policy for ordinary importance sampling is to take actions in proportion to $\pi_e(a)r(a)$. Thus behavior policy search may yield a behavior policy that is sub-optimal for the RIS estimator.

Bandit problems are a useful way to show that the best choice of behavior policy depends on whether we will use ordinary importance sampling or regression importance sampling. However, our ultimate goal is full MDPs and future work should consider how to extend behavior policy search to RIS in both bandits and MDPs.

[32] This statement follows from having deterministic rewards and the observation of Li et al. (2015) that importance sampling with an estimated behavior policy is equivalent to an analytic expectation over the estimated reward function.

10.2.1.2 Behavior Policy Search for Value Function Learning

This dissertation has focused on the goal of estimating $v(\pi_e)$ which is the expected return of a policy under the initial state distribution. This problem is closely related to the more general problem of learning the value function of a policy, $v^{\pi_e} : \mathcal{S} \to \mathbb{R}$, that gives, for any state, the expected sum of rewards from that state (Sutton and Barto, 1998). A natural question is whether or not ideas from behavior policy search lead to more efficient value function learning.

Recall that $v^{\pi_e}(s)$ is the expected cumulative reward obtained when following policy π_e from state s. Our goal is to estimate v^{π_e} with minimal MSE *over all states*. Specifically, given an estimated value-function, $\hat{v}^{\pi_e}$, and the true (unknown in practice) value-function, v^{π_e}, we wish to minimize:

$$\sum_{s \in \mathcal{S}} \mu(s) \left(\hat{v}^{\pi_e}(s) - v^{\pi_e}(s)\right)^2$$

where $\mu(s)$ is a state-dependent weighting that captures how much we care about getting the value estimate right in state s. Value function learning methods typically compute a target, U_t, and update $\hat{v}^{\pi_e}(S_t)$ towards this target. For convergence to v^{π_e}, the target should be an estimate of $v^{\pi_e}(S_t)$.

A straightforward approach to compute U_t is to use the Monte Carlo return following S_t:

$$U_t = \prod_{j=t}^{L} \frac{\pi_e(A_j|S_j)}{\pi_b(A_j|S_j)} (R_t + R_{t+1} + ... + R_L).$$

With the right choice of behavior policy, we can use importance sampling to lower the variance of this evaluation.

While straightforward, the Monte Carlo approach ignores the fact that estimates of the value of one state can be used when estimating the value of another state. Instead of full Monte Carlo returns, n-step returns allow us to make use of

the intermediate value function estimates:

$$U_t = R_t + R_{t+1} + ... + R_{n-1} + v^{\pi_e}(S_{t+n}).$$

The n-step return is a Monte Carlo return that is truncated after n steps with the remaining rewards replaced with the estimated value of the state at step n. As with the full return, we can use importance sampling to create an off-policy n-step return:

$$U_t = \prod_{j=t}^{t+n} \frac{\pi_e(A_j|S_j)}{\pi_b(A_j|S_j)} \left(R_t + R_{t+1} + ... + R_{n-1} + v^{\pi_e}(S_{t+n})\right).$$

The variance of this estimate is dependent on the choice of π_b.

In principle, the variance of the n-step return can be lowered with behavior policy search in the same way that we used behavior policy search to lower the variance of the Monte Carlo return. The main challenge is that the variance of the n-step return will also change as the value function improves. For instance, assume the value function is initialized to zero for all states. Then the value function term in the n-step return contributes nothing to the variance of the n-step return and behavior policy search would just lower the variance of the n reward terms. However, after updating the value function, the variance of the n-step return will also depend on the magnitude of the value function at the $n^{\texttt{th}}$ state along the partial trajectory. For this new value function, it may be that a different behavior policy is needed to achieve the most variance reduction. The changing value function means that behavior policy search must track a moving target – adapting the behavior policy to lower the variance of the n-step whose variance is also changing as the value function changes.

10.2.1.3 Behavior Policy Search for Policy Improvement

Chapter 4 discussed combining behavior policy search with policy improvement when using batch policy gradient reinforcement learning. Though an algorithm – parallel policy search – was proposed, it remains to be seen if it is an improvement over common on-policy batch policy gradient methods.

The most important question is how to resolve the tension between the need to explore and the need to reduce variance. The former involves trying new actions while the latter involves increasing the probability of already tried actions that led to high magnitude returns. Decreasing variance with behavior policy search should allow faster learning but there is a risk of sub-optimal convergence if the behavior policy stops trying optimal actions. Determining how to balance gains from variance reduction with exploration of optimal actions is an important step to realizing a robust parallel policy search algorithm.

Simultaneously learning the behavior and target policy requires synchronizing the policy update rates so that the behavior policy remains a lower variance behavior policy than the target policy. In general, tuning two dependent policy search procedures will be a difficult process and lead to a less robust method. Identifying ways to synchronize the policy learning rates or set trust-regions on the policy updates are two possible directions for improving this process and improving the robustness of parallel policy search.

10.2.2 Weighting Off-Policy Data

Chapters 5 and 6 showed how already collected data should be *weighted* by a learning agent for policy value estimation and policy improvement. The discussion focused on correcting sampling error by importance sampling with an estimated behavior policy. In this subsection, we discuss extensions to this work.

10.2.2.1 Correcting Sampling Error in Other Ways

Both regression importance sampling and the sampling error corrected policy gradient estimator use importance sampling to correct sampling error in the observed actions. In reinforcement learning, sampling error is also due to sampling in the initial state distribution and environment dynamics. We term such sampling error *environment sampling error*. Correcting this error would, in principle, lead to a more efficient use of data. However, the true probabilities of initial states and next states are unknown and thus it is unclear what the correct weighting should be.

A naive approach to correct environment sampling error would be to estimate these probabilities with a separate data set and then use the estimate as the true value for a sampling error correction. However, this approach would likely be unsuccessful as the error in estimating the transition probabilities might be as high as the sampling error.

A more promising approach might be to focus on correcting sampling error in the initial state distribution. In the episodic RL setting, the initial state distribution is typically fixed and we obtain a new sample from the distribution every time a new trajectory begins. In contrast, new samples from $P(\cdot|s,a)$ are only encountered when the agent reaches state s and takes action a.

Batch policy gradient algorithms are one class of RL methods where estimating the initial state distribution to correct sampling error could be useful. Batch policy gradient methods typically sample a set of new trajectories at each iteration, use it to estimate the gradient for updating the policy, and then discard the trajectories before the next iteration. Instead of discarding all data, we can use data from iterations 1 to i to estimate the initial state distribution, d_0. We could then use the estimated d_0 as the true initial state distribution to correct initial state sampling error in iteration $i+1$. This approach makes use of data that would otherwise be discarded. Empirical and theoretical study of this approach is an interesting direction for future work.

10.2.2.2 Regression Importance Sampling for High Confidence Policy Value Estimation

Chapter 5 showed that regression importance sampling leads to lower mean squared error policy value estimation. It remains to be seen if RIS also leads to tighter confidence intervals for high confidence policy value estimation. One way to tackle this problem would be to simply use RIS with the general bootstrap procedure we introduced in Chapter 8. Given that RIS has been empirically shown to have lower variance than OIS, we could expect such a method to produce tighter confidence intervals.

A more challenging direction for future work would be to obtain true confidence intervals with an estimated behavior policy. While the data efficiency of bootstrapping is desirable, it only provides approximate confidence bounds. In order to determine exact confidence intervals for RIS, we would need to develop *concentration inequalities* for RIS in the same way that one can use Hoeffding's inequality to establish confidence intervals for OIS. One possible direction is to explore use of the Dvoretzky-Kiefer-Wolfowitz inequality which bounds how far the empirical distribution of samples is from the true distribution (Dvoretzky et al., 1956). Regardless of the exact approach, exact confidence bounds for importance sampling with an estimated behavior policy would be of great value to providing provable guarantees of safety in real world settings.

10.2.2.3 Regression importance sampling for value function learning

Correcting sampling error with RIS could also lead to more efficient value function learning. The value function learning problem raises two new questions. First, we are now learning a value for all states and not just the expected value under the initial state distribution. Does RIS provide the same data efficiency benefits for value function learning as it does for policy value estimation given a fixed batch of

data? Empirical and theoretical work should validate if RIS provides lower variance learning targets for value function updates.

The second question pertains to learning a value function under *online* learning as opposed to the batch setting. In the online setting, the learning agent processes one (S, A, S', R) tuple at a time, using the information in this transition to compute a target for updating the value function estimate. If applying RIS in this setting, the main question is whether we estimate the behavior policy with only the single tuple or use all data observed up to the current time. The former is closer to our recommendation of using only the evaluation data to compute the estimate, but risks introducing too much bias. The latter is farther from this recommendation but may allow correcting sampling error in the online stochastic approximation of v^{π_e}. Studying these approaches empirically and theoretically is one direction for extending RIS to value function learning.

10.2.2.4 Open Theoretical Questions

In Chapter 5 we proved RIS has asymptocially at most the variance of OIS. Further theoretical analysis should examine the finite-sample bias, variance, and mean squared error of RIS compared to OIS. A starting point for this work could be the results of Li et al. (2015) who provide bounds on these finite-sample quantities in the bandit setting. Extending these results to MDPs would give us a deeper understanding of when RIS is lower MSE estimator compared to OIS. The empirical results in Chapter 5 provide strong evidence that RIS is always preferable to OIS. However, theoretical analysis would strengthen this claim.

The theoretical analysis in Chapter 5 did *not* distinguish different RIS methods according to how much history they conditioned on (the estimator parameter n). Theoretical analysis of the finite-sample bias-variance trade-off and asymptotic variance for different RIS methods would deepen our understanding of how to choose

n. Empirical results on the Singlepath domain (Chapter 5, Figure 5.5) suggest that small n have lower small-sample MSE while large n have asymptotically lower MSE. Verifying this finding formally is an interesting direction for future work.

10.2.3 Learning with Simulated Data

Chapter 7 introduced an algorithm, grounded action transformation (GAT), that allows a reinforcement learning agent to learn with simulated data.

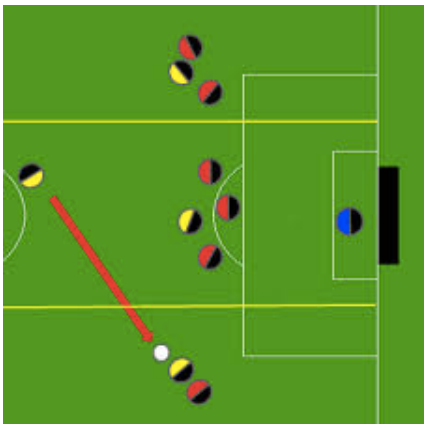
10.2.3.1 Sim-to-real in Non-Robotics Domains

In Chapter 7, we evaluated GAT on a physical NAO robot. GAT is not specific to the NAO and could be applied on other robotics tasks or even non-robotics tasks where a simulator is available a priori. The latter is of particular interest as the sim-to-real problem has been studied to a much lesser extent in non-robotics domains. GAT is most applicable in tasks where the dynamics have a basis in physics and actions have a direct effect on some state variables. In such settings, it is reasonable to assume that an effective action grounding function can be learned. It may be less applicable where the dynamics are derived from other factors such as human behavior. Identifying ways to automatically determine when GAT is applicable is an interesting direction for future work.

10.2.3.2 Sim-to-real in extremely low fidelity simulations

We have formulated sim-to-real as two MDPs that only differ in the transition probabilities. Another interesting problem is to consider transfer when the MDPs differ in the state and action spaces yet have some shared structure as to allow learning in one MDP to benefit learning in the other. Of particular interest are environment pairs where one MDP can be thought of as a low fidelity abstraction of the other. For example, consider the two robot soccer domains depicted in Figure

10.1. One domain is a simulation of soccer played between circles while the other domain fully simulates the dynamics of humanoid robots. In such domains, high level strategy policies may be able to transfer from low to high fidelity while motion control policies for the high fidelity simulator have no analog in the low fidelity one. Determining what can transfer directly, what can transfer with simulator grounding, and what simply must be learned in the high fidelity model is an interesting direction for future work.



(a) RoboCup 2D Simulator

(b) RoboCup 3D Simulator

Figure 10.1: Screenshots from the RoboCup 2D and RoboCup 3D simulators.[33] Both simulated robot soccer tasks involve high-level coordination of robot teammates against adversaries, however, they differ in the low-level state and action spaces.

It is reasonable to believe that some elements of a policy can be learned in the low-fidelity model and transferred without modification to the target task. For instance, team strategy for the RoboCup 3D simulation could be learned in the RoboCup 2D simulation as teammate and opponent positions are the most relevant information for team coordination and these aspects of the task are modeled in both simulations. However, even in the shared high-level state and action space, direct transfer may fail if the learned strategy requires low level skills that are unrealizable in the target task. For instance, in the RoboCup 2D simulation competition, team

strategies typically involve many short, accurate passes. However, in the RoboCup 3D simulaton competition, the inaccuracy of passing causes teams to use strategies that rely on passing less. In this case, the low level 3D skills prevent high level strategy from transferring. In such cases, grounding the low-fidelity simulator is a promising direction for allowing skills to transfer.

If the available simulator is too low-fidelity, some aspects of behavior will simply have to be learned in the target task. For example, the robots in the RoboCup 3D simulation cannot learn bipedal walk control policies in the RoboCup 2D simulation. Even in such cases, there remains a use for the simulator as a means to shape the learning process in the target task. For instance, learning in the target task could be done to both maximize reward and to match the high-level outcomes in the low-fidelity simulator. This joint task would have a more dense reward signal and might be easier to learn.

10.2.4 Combining Off-Policy and Simulated Data

In Chapter 8, we introduced the WDR-BOOTSTRAP method that leverages the doubly robust estimator to combine off-policy and simulated data. Doubly robust estimators have great promise for lowering the variance of importance-sampled off-policy data with simulated data. We discuss here opportunities to improve these estimators, in particular, with an eye towards robotics applications.

10.2.4.1 Continuous Actions and Doubly Robust Estimators

Doubly robust estimators (DR and WDR) require a state-value estimate, $\hat{v}^{\pi_e}(s)$, for all s that occur in the observed $\mathcal{D}$ and an action-value estimate, $\hat{q}^{\pi_e}(s,a)$ for all (s,a) that occur in $\mathcal{D}$. Furthermore, $\hat{v}^{\pi_e}(s)$ must equal $\mathbf{E}_{A\sim\pi_e}\left[\hat{q}^{\pi_e}(s,A)\right]$ to avoid introducing model bias into the estimate. This condition is easily met when $\mathcal{A}$ is

[33] https://ssim.robocup.org/

a finite set as $\hat{v}^{\pi_e}(s)$ can analytically be computed from $\hat{q}^{\pi_e}$. However, when $\mathcal{A}$ is infinite it may be intractable to exactly compute the expectation.

To the best of our knowledge, our Cliff World experiment in Chapter 8 is the only study that used a doubly robust estimator with continuous action spaces. In this experiment, we used a Monte Carlo approximation of $\hat{v}^{\pi_e}$. This approach only provides an approximation and was also computationally demanding. Results in this experiment suggested that the error introduced from this approximation may have impacted the quality of the WDR estimator. An interesting direction for future work is to determine how best to handle continuous actions or how to account for this error when using doubly robust estimators.

A promising possibility is to study forms of $\hat{q}^{\pi_e}$ that make an analytic evaluation of $\mathbf{E}_{A\sim\pi_e}\left[\hat{q}^{\pi_e}(s, A)\right]$ tractable. A number of special forms of $\hat{q}^{\pi_e}$ and π_e exist where analytic evaluation becomes tractable (Ciosek and Whiteson, 2018). Even if $\hat{q}^{\pi_e}$ is an arbitrary function, provided it is differentiable, it may be possible to use a Taylor expansion approximation to obtain a form that can be integrated over (Gu et al., 2017a). The drawback of this direction is that we give up some representation power in $\hat{q}^{\pi_e}$ in exchange for being able to tractably integrate over it.

Another interesting direction is to study how error in the estimate of $\hat{v}^{\pi_e}$ impacts the accuracy of doubly robust estimators. In some cases it may be acceptable to have some error in $\hat{v}^{\pi_e}$ if $\hat{q}^{\pi_e}$ is a more accurate estimate of the true action-value function, q^{π_e}. Both theoretical and empirical study of these cases is an interesting direction for future work.

10.2.4.2 Combining Existing Simulations with Doubly Robust Estimators

Our work with doubly robust estimators used simulators (i.e., models) learned directly from data. An alternative approach is to leverage existing simulations as

the model for DR or WDR. This approach has the benefit of incorporating domain knowledge encoded in the simulation. A natural extension to using a simulator for doubly robust estimators would be to first improve the simulation using the grounded action transformation algorithm.

10.3 Concluding Remarks

This dissertation has developed techniques that allow reinforcement learning agents to learn and evaluate policies using off-policy and simulated data. The ability to leverage these forms of auxiliary data is an important step towards more data efficient reinforcement learning and more data efficient evaluation of learned skills. Towards the goal of leveraging these data sources, this dissertation has enhanced both the theory and practice of reinforcement learning and opened up many new directions for research in off-policy and sim-to-real reinforcement learning.

Appendix A

Notation Summary

This appendix summarizes notation used in this dissertation. In general, we use the notation that capital letters denote random variables, lower case letters represent elements of sets, and calligraphic capital letters denote sets. Vectors are written in bold lowercase letters.

- $\mathcal{S}$ is a set of possible world states. Elements of $\mathcal{S}$ are typically denoted s.
- $\mathcal{A}$ is a set of actions available to the agent. Elements of $\mathcal{A}$ are typically denoted a.
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function giving the probability of transitioning to a state s' after choosing action a in state s. P is also known as the dynamics of the environment.
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function.
- L is the maximum length of one episode of interacting with the environment.
- $\gamma \in [0, 1]$ is a discount factor that allows us to express a preference for immediate rewards compared to future rewards.

- d_0 is an initial state distribution.
- s_∞ is the terminal state.
- A trajectory h is a sequence of states, actions, and rewards of length L: $s_0, a_0, r_0, \ldots s_{L-1}, a_{L-1}, r_{L-1}$.
- $h_{i:j}$ is the trajectory segment: $s_i, a_i, r_i, \ldots, s_j, a_j, r_j$.
- $g(h)$ is the return of h. The return of h is the discounted sum of rewards received during h: $\sum_{t=0}^{L-1} \gamma^t r_t$.
- $\Pr(\cdot|\pi)$ is the distribution of trajectories when taking actions according to π. We sometimes write $H \sim \pi$ in place of $H \sim \Pr(\cdot|\pi)$ to denote sampling a trajectory from the trajectory distribution induced by π.
- $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a policy.
- $v(\pi, \mathcal{M})$ is the expected return of policy π in MDP $\mathcal{M}$, $v(\pi, \mathcal{M}) \coloneqq \mathbf{E}\left[g(H)|H \sim \pi\right]$.
- $v^\pi(s, t)$ is the expected return from following policy π from state s at step t.
- $q^\pi(s, a, t)$ is the expected return from following policy π after taking action a in state s at step t.
- $\mathcal{D}$ is a set of m trajectories generated by the behavior policy π_b: $\mathcal{D} \coloneqq \{(H_i, \pi_b)\}_{i=1}^m$.
- $p_\mathcal{D}$ is the distribution over different realizations of $\mathcal{D}$.
- $v_\delta(\pi)$ is a $1 - \delta$ confidence interval lower bound on $v(\pi)$.
- $\pi_{\boldsymbol{\theta}}$ is a policy parameterized by vector $\boldsymbol{\theta}$.

- $\rho_t^{(h)} \coloneqq \prod_{i=0}^{t} \frac{\pi(a_i|s_i)}{\pi_b(a_i|s_i)}$ is the importance weight ratio up to and including time-step t for trajectory h.

- MSE$(\cdot)$ denotes the mean squared error of its argument with respect to $v(\pi_e)$. We assume that the random variable under which the MSE is computed is clear from the context.

- Var$(\cdot)$ denotes the variance of its argument. We assume that the random variable under which the variance is computed is clear from the context.

- Bias$(\cdot)$ denotes the bias of its argument. We assume that the random variable under which the bias is computed is clear from the context.

Appendix B

Acronym Summary

This section contains a list of acronyms used throughout this dissertation, listed by the chapter in which they are first introduced.

Chapter 2

- MDP: Markov decision process.
- MSE: Mean squared error.
- PE: Policy evaluate, a general policy value estimator.
- MC: The Monte Carlo estimator.
- IS: The importance sampling estimator.
- WIS: The weighted importance sampling estimator.
- PDIS: The per-decision importance sampling estimator.
- PDWIS: The per-decision weighted importance sampling estimator.
- MB: The model-based estimator.

- DR: The doubly robust estimator.
- ASE: The advantage-sum estimator.
- WDR: The weighted doubly robust estimator.

Chapter 3

- OPE: Off-policy evaluate, a general off-policy policy value estimator.
- BPG: The behavior policy gradient algorithm.
- DR-BPG: The doubly robust behavior policy gradient algorithm.
- CEM: The cross-entropy method.

Chapter 4

- PPS: The parallel policy search algorithm.
- TRPO: The trust-region policy optimization algorithm.

Chapter 5

- OIS: The ordinary importance sampling estimator.
- RIS: The regression importance sampling estimator.
- REG: The regression estimator.
- LDS: The linear dynamical system domain.

Chapter 6

- SEC: The sampling error corrected policy gradient estimator.

Chapter 7

- GSL: The grounded simulation learning framework.
- GAT: The grounded action transformation algorithm.
- CMA-ES: The covariance matrix adaptation evolutionary strategy algorithm.
- UNSW: The University of New South Wales.
- SPL: The Standard Platform League of RoboCup.

Chapter 8

- MB-BOOTSTRAP: The model-based bootstrap algorithm.
- WDR-BOOTSTRAP: The weighted doubly robust bootstrap algorithm.
- KL: Kullback-Leibler.

Chapter 9

- PAC: Probably approximately correct.
- MSVE: Mean squared value error.

Appendix C

Collecting Off-Policy Data: Derivations and Proofs

This appendix includes the derivations of all theoretical results referenced in Chapter 3. We also derive a connection between the behavior policy gradient algorithm and an existing adaptive importance sampling method known as the cross-entropy method (Rubinstein, 1999). Finally, we provide a proof that BPG provides unbiased estimates of $v(\pi_e)$.

C.1 Behavior Policy Gradient Theorem

In this section, we derive the gradient of the variance of importance sampling with respect to the behavior policy parameters. We first derive an analytic expression for the gradient of the variance of an arbitrary, unbiased off-policy policy evaluation estimator, $\text{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})$. From our general derivation we derive the gradient of the variance of the ordinary importance sampling estimator and then extend to the doubly robust and per-decision estimators.

C.1.1 MSE Gradient for an Unbiased Off-Policy Policy Evaluation Method

Lemma C.1 gives the gradient of the mean squared error (MSE) for any unbiased off-policy policy evaluation method.

Lemma C.1.

$$\frac{\partial}{\partial \boldsymbol{\theta}} \operatorname{MSE}\Big[\operatorname{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})\Big] = \mathbf{E}\Big[\operatorname{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 (\sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t)) + \frac{\partial}{\partial \boldsymbol{\theta}} \operatorname{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 \;\Big|\; H \sim \pi_{\boldsymbol{\theta}}\Big]$$

Proof. We begin by decomposing $\Pr(H = h|\pi)$ into two components – one that depends on π and the other that does not. Let

$$w_\pi(h) \coloneqq \prod_{t=0}^{L-1} \pi(a_t|s_t),$$

and

$$p(h) \coloneqq \Pr(h|\pi)/w_\pi(h),$$

for any π such that h is in the support of π (any such π will result in the same value of $p(h)$). These two definitions mean that $\Pr(H = h|\pi) = p(h)w_\pi(h)$.

The MSE of the OPE estimator is given by:

$$\begin{aligned}\operatorname{MSE}[\operatorname{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})] = &\operatorname{Var}[\operatorname{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})] + \\ &\underbrace{\left(\mathbf{E}[\operatorname{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})] - v(\pi_e)\right)^2}_{\text{bias}^2}. \end{aligned} \tag{C.1}$$

Since the OPE estimator is unbiased, i.e., $\mathbf{E}[\operatorname{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})] = v(\pi_e)$, the second

term is zero and so:

$$\begin{aligned}
\mathrm{MSE}(\mathrm{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})) =& \mathrm{Var}[\mathrm{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})] && \text{(C.2)}\\
=& \mathbf{E}\left[\mathrm{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 \middle| H \sim \pi_{\boldsymbol{\theta}}\right] - \\
& \mathbf{E}[\mathrm{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}}) | H \sim \pi_{\boldsymbol{\theta}}]^2 && \text{(C.3)}\\
=& \mathbf{E}\left[\mathrm{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 \middle| H \sim \pi_{\boldsymbol{\theta}}\right] - v(\pi_e)^2 && \text{(C.4)}
\end{aligned}$$

To obtain the MSE gradient, we differentiate $\mathrm{MSE}[\mathrm{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})]$ with respect to $\boldsymbol{\theta}$:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}} \mathrm{MSE}[\mathrm{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})] =& \frac{\partial}{\partial \boldsymbol{\theta}} \left(\mathbf{E}\left[\mathrm{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 \middle| H \sim \pi_{\boldsymbol{\theta}}\right] - v(\pi_e)^2\right) && \text{(C.5)}\\
=& \frac{\partial}{\partial \boldsymbol{\theta}} \mathbf{E}\left[\mathrm{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 \middle| H \sim \pi_{\boldsymbol{\theta}}\right] && \text{(C.6)}\\
=& \frac{\partial}{\partial \boldsymbol{\theta}} \sum_h \Pr(h|\pi_{\boldsymbol{\theta}}) \, \mathrm{OPE}(\pi_e, h, \pi_{\boldsymbol{\theta}})^2 && \text{(C.7)}\\
=& \sum_h \Pr(h|\boldsymbol{\theta}) \frac{\partial}{\partial \boldsymbol{\theta}} \mathrm{OPE}(\pi_e, h, \pi_{\boldsymbol{\theta}})^2 + \\
& \mathrm{OPE}(\pi_e, h, \pi_{\boldsymbol{\theta}})^2 \frac{\partial}{\partial \boldsymbol{\theta}} \Pr(h|\boldsymbol{\theta}) && \text{(C.8)}\\
=& \sum_h \Pr(h|\boldsymbol{\theta}) \frac{\partial}{\partial \boldsymbol{\theta}} \mathrm{OPE}(\pi_e, h, \pi_{\boldsymbol{\theta}})^2 + \\
& \mathrm{OPE}(\pi_e, h, \pi_{\boldsymbol{\theta}})^2 p(h) \frac{\partial}{\partial \boldsymbol{\theta}} w_{\pi_{\boldsymbol{\theta}}}(h) && \text{(C.9)}
\end{aligned}$$

Consider the last factor of the last term in more detail:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}} w_{\pi_{\boldsymbol{\theta}}}(h) =& \frac{\partial}{\partial \boldsymbol{\theta}} \prod_{t=0}^{L-1} \pi_{\boldsymbol{\theta}}(a_t|s_t) && \text{(C.10)}\\
\overset{\text{(a)}}{=}& \left(\prod_{t=0}^{L-1} \pi_{\boldsymbol{\theta}}(a_t|s_t)\right) \left(\sum_{t=0}^{L-1} \frac{\frac{\partial}{\partial \boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a_t|s_t)}{\pi_{\boldsymbol{\theta}}(a_t|s_t)}\right) && \text{(C.11)}\\
=& w_{\pi_{\boldsymbol{\theta}}}(h) \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log\left(\pi_{\boldsymbol{\theta}}(a_t|s_t)\right), && \text{(C.12)}
\end{aligned}$$

where **(a)** comes from the multi-factor product rule. Continuing from (C.9) we have that:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \text{MSE}(\text{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})) = \mathbf{E}\Bigg[\text{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log\left(\pi_{\boldsymbol{\theta}}(A_t|S_t)\right) + \frac{\partial}{\partial \boldsymbol{\theta}} \text{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 \;\Big|\; H \sim \pi_{\boldsymbol{\theta}} \Bigg].$$

□

C.1.2 Behavior Policy Gradient Theorem

We now use Lemma C.1 to prove the Behavior Policy Gradient Theorem which is the main theoretical contribution of Chapter 3.

Theorem 3.1. *Behavior Policy Gradient Theorem*

$$\frac{\partial}{\partial \boldsymbol{\theta}} \text{MSE}[\text{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}})] = \mathbf{E}\left[-\text{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \,\middle|\, H \sim \pi_{\boldsymbol{\theta}} \right]$$

Proof. We first derive $\frac{\partial}{\partial \boldsymbol{\theta}} \text{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2$. Theorem 3.1 then follows directly from using $\frac{\partial}{\partial \boldsymbol{\theta}} \text{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2$ as $\frac{\partial}{\partial \boldsymbol{\theta}} \text{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2$ in Lemma C.1.

$$\begin{aligned}
\text{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 &= \left(\frac{w_{\pi_e}}{w_{\pi_{\boldsymbol{\theta}}}} g(H)\right)^2 \\
\frac{\partial}{\partial \boldsymbol{\theta}} \text{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 &= \frac{\partial}{\partial \boldsymbol{\theta}} \left(\frac{w_{\pi_e}(H)}{w_{\pi_{\boldsymbol{\theta}}}(H)} g(H)\right)^2 \\
&= 2 \cdot g(H) \frac{w_{\pi_e}(H)}{w_{\pi_{\boldsymbol{\theta}}}(H)} \frac{\partial}{\partial \boldsymbol{\theta}} \left(g(H) \frac{w_{\pi_e}(H)}{w_{\pi_{\boldsymbol{\theta}}}(H)}\right) \\
&\overset{\text{(a)}}{=} -2 \cdot g(H) \frac{w_{\pi_e}(H)}{w_{\pi_{\boldsymbol{\theta}}}(H)} \left(g(H) \frac{w_{\pi_e}(H)}{w_{\pi_{\boldsymbol{\theta}}}(H)}\right) \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \\
&= -2\, \text{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t)
\end{aligned}$$

where **(a)** comes from the multi-factor product rule and the likelihood-ratio trick (i.e., $\frac{\frac{\partial}{\partial \boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(A|S)}{\pi_{\boldsymbol{\theta}}(A|S)} = \log(\pi_{\boldsymbol{\theta}}(A|S))$

Substituting this expression into Lemma C.1 completes the proof of Theorem 3.1:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \operatorname{MSE}[\operatorname{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}})] = \mathbf{E}\left[-\operatorname{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \middle| H \sim \pi_{\boldsymbol{\theta}} \right].$$

□

C.1.3 MSE Gradient for the Doubly Robust Estimator

We also present an extension of the IS MSE gradient to the doubly robust (DR) estimator. Recall that for a single trajectory, H, DR is given as:

$$\operatorname{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e}) \coloneqq \hat{v}^{\pi_e}(S_0, 0) + \sum_{t=0}^{L-1} \gamma^t \frac{w_{\pi_e,t}}{w_{\pi_{\boldsymbol{\theta}},t}} \delta_t$$

where $\hat{v}^{\pi_e}$ is an approximation of the state-value function of π_e, $\hat{q}^{\pi_e}$ is an approximation of the action-value function of π_e, $w_{\pi,t} \coloneqq \prod_{j=0}^{t} \pi(A_j|S_j)$, and $\delta_t \coloneqq R_t - \hat{q}^{\pi_e}(S_t, A_t, t) + \gamma^t \hat{v}^{\pi_e}(S_{t+1}, t+1)$

The gradient of the mean squared error of the DR estimator is given by the following corollary to the Behavior Policy Gradient Theorem:

Corollary 3.1.

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\theta}} \operatorname{MSE}\left[\operatorname{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e})\right] = \mathbf{E}[&(\operatorname{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e})^2 \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \\ &- 2 \operatorname{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e}) (\sum_{t=0}^{L-1} \gamma^t \delta_t \frac{w_{\pi_e,t}}{w_{\pi_{\boldsymbol{\theta}},t}} \sum_{i=0}^{t} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_i|S_i)) | H \sim \pi_{\boldsymbol{\theta}}]\end{aligned}$$

where $\delta_t = R_t - \hat{q}^{\pi_e}(S_t, A_t, t) + \gamma^t \hat{v}^{\pi_e}(S_{t+1}, t+1)$.

Proof. As with Theorem 3.1, we first derive $\frac{\partial}{\partial \boldsymbol{\theta}} \mathrm{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e})^2$. Corollary 3.1 then follows directly from using $\frac{\partial}{\partial \boldsymbol{\theta}} \mathrm{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e})^2$ as $\frac{\partial}{\partial \boldsymbol{\theta}} \mathrm{OPE}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2$ in Lemma C.1.

Let $\delta_t \coloneqq R_t - \hat{q}^{\pi_e}(S_t, A_t, t) + \hat{v}^{\pi_e}(S_{t+1}, t+1)$.

$$\mathrm{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e})^2 = \left(\hat{v}^{\pi_e}(S_0, 0) + \sum_{t=0}^{L-1} \gamma^t \frac{w_{\pi_e,t}}{w_{\pi_{\boldsymbol{\theta}},t}} \delta_t \right)^2$$

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}} \mathrm{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e})^2 =& \frac{\partial}{\partial \boldsymbol{\theta}} \left(\hat{v}^{\pi_e}(S_0, 0) + \sum_{t=0}^{L-1} \gamma^t \frac{w_{\pi_e,t}}{w_{\pi_{\boldsymbol{\theta}},t}} \delta_t \right)^2 \\
=& 2\, \mathrm{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e}) \frac{\partial}{\partial \boldsymbol{\theta}} \left(\hat{v}^{\pi_e}(S_0, 0) + \sum_{t=0}^{L-1} \gamma^t \frac{w_{\pi_e,t}}{w_{\pi_{\boldsymbol{\theta}},t}} \delta_t \right) \\
=& -2\, \mathrm{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e}) \left(\sum_{t=0}^{L-1} \gamma^t \frac{w_{\pi_e,t}}{w_{\pi_{\boldsymbol{\theta}},t}} \delta_t \sum_{i=0}^{t} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_i | S_i) \right)
\end{aligned}$$

Thus the $\mathrm{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e})$ gradient is:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}} \mathrm{MSE}\left[\mathrm{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e})\right] = \mathbf{E}[&\mathrm{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e})^2 \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t | S_t) \\
&- 2\, \mathrm{DR}(\pi_e, H, \pi_{\boldsymbol{\theta}}, \hat{q}^{\pi_e}, \hat{v}^{\pi_e}) \left(\sum_{t=0}^{L-1} \gamma^t \delta_t \frac{w_{\pi_e,t}}{w_{\pi_{\boldsymbol{\theta}},t}} \sum_{i=0}^{t} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_i | S_i) \right) | H \sim \pi_{\boldsymbol{\theta}}]
\end{aligned}$$

□

The expression for the DR behavior policy gradient is more complex than the expression for the IS behavior policy gradient. Lowering the variance of DR involves accounting for the covariance of the sum of terms. Intuitively, accounting for the covariance increases the complexity of the expression for the gradient.

C.1.4 MSE Gradient for the Per-Decision Importance Sampling Estimator

We also note that the gradient of the mean squared error of per-decision importance sampling can be obtained as the special case of Corollary 3.1 where $\hat{v}^{\pi_e}$ and $\hat{q}^{\pi_e}$ are zero for all states and actions:

Corollary C.1.

$$\frac{\partial}{\partial \boldsymbol{\theta}} \operatorname{MSE}\left[\operatorname{PDIS}(\pi_e, H, \pi_{\boldsymbol{\theta}})\right] = \mathbf{E}[\operatorname{PDIS}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t)$$
$$- 2 \operatorname{PDIS}(\pi_e, H, \pi_{\boldsymbol{\theta}})(\sum_{t=0}^{L-1} \gamma^t R_t \frac{w_{\pi_e,t}}{w_{\pi_{\boldsymbol{\theta}},t}} \sum_{i=0}^{t} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_i|S_i))|H \sim \pi_{\boldsymbol{\theta}}]$$

Proof. Let $\hat{v}^{\pi_e}(s,t) \coloneqq 0$ for all states and time-steps and let $\hat{q}^{\pi_e}(s,a,t) \coloneqq 0$ for all (s,a,t) and the proof follows from Corollary C.1.3. □

We have *not* implemented this gradient as a component of behavior policy search but include the derivation since PDIS is a popular unbiased off-policy estimator.

C.2 The Behavior Policy Gradient Algorithm and the Cross-Entropy Method

In this section we derive a connection between the behavior policy gradient algorithm and the cross-entropy method (Rubinstein, 1999). The cross-entropy method is an existing approach to adaptive importance sampling. We show under a set of limiting assumptions that the behavior policy update rule used by the cross-entropy method is similar to that of behavior policy gradient.

The cross-entropy method (CEM) is an estimation technique for adapting the sampling distribution (i.e., the behavior policy) to obtain lower variance importance

sampling estimates. CEM adapts $\boldsymbol{\theta}$ to minimize the Kullback-Leibler divergence ($D_{\mathtt{KL}}$) between the theoretically optimal sampling distribution, $\pi_b^\star$, and the current sampling distribution parameterized by $\boldsymbol{\theta}$. For some families of distributions this update can be computed analytically. Unfortunately in the sequential decision making setting an analytic update is impossible. Instead we derive a gradient based approach to minimizing $D_{\mathtt{KL}}$ which elicits a connection between CEM and BPG.

First, note that for ${\pi_b}^\star$ to exist, we require the condition that $\forall h,\ g(h)$ is positive and that P and r are deterministic. This statement does not imply importance sampling cannot be used to lower the variance of off-policy estimates; it just means that ${\pi_b}^\star$, as derived in Section 3.2, will not exist unless all returns are positive.

The gradient of the KL-divergence between the distribution of trajectories under the optimal behavior policy, $\Pr(\cdot|{\pi_b}^\star)$, and the distribution of trajectories under $\pi_{\boldsymbol{\theta}}$, $\Pr(\cdot|\pi_{\boldsymbol{\theta}})$ is:

$$\frac{\partial}{\partial \boldsymbol{\theta}} D_{\mathtt{KL}}(\Pr(\cdot|{\pi_b}^\star)||\Pr(\cdot|\pi_{\boldsymbol{\theta}})) \propto \mathbf{E}\left[-\operatorname{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}}) \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \middle| H \sim \pi_{\boldsymbol{\theta}}\right]$$

Proof. First consider the KL-divergence between the optimal behavior policy and

the current behavior policy $\pi_{\boldsymbol{\theta}}$.

$$
\begin{aligned}
&D_{\mathtt{KL}}(\Pr(\cdot|{\pi_b}^\star)||\Pr(\cdot|\pi_{\boldsymbol{\theta}})) \\
&= \mathbf{E}\left[\log \frac{Pr(H|\pi^\star)}{\Pr(H|\pi_{\boldsymbol{\theta}})}\middle| H \sim {\pi_b}^\star\right] \\
&= \mathbf{E}\left[\log \frac{w_{{\pi_b}^\star}(H)}{w_{\boldsymbol{\theta}}(H)}\middle| H \sim {\pi_b}^\star\right] \\
&= \mathbf{E}\Big[\log w_{\pi_e}(H) - \log w_{\pi_{\boldsymbol{\theta}}}(H) + \log g(H) - \log v(\pi_e) \;\Big|\; H \sim {\pi_b}^\star\Big] \\
&= \sum_h \Pr(h|\pi_e)\frac{g(h)}{v(\pi_e)}\left[\log w_{\pi_e}(H) - \log w_{\pi_{\boldsymbol{\theta}}}(H) + \log g(H) - \log v(\pi_e)\right] \\
&= \frac{1}{v(\pi_e)}\mathbf{E}\left[g(H)[\log w_{\pi_e}(H) - \log w_{\pi_{\boldsymbol{\theta}}}(H) - \log g(H) - \log v(\pi_e)]|H \sim \pi_e\right]
\end{aligned}
$$

Differentiating with respect to $\boldsymbol{\theta}$, we obtain:

$$
\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{\theta}} D_{\mathtt{KL}}(\Pr(\cdot|{\pi_b}^\star)||\Pr(\cdot|\pi_{\boldsymbol{\theta}})) \\
&\qquad \propto \mathbf{E}\left[-g(H)\frac{\partial}{\partial \boldsymbol{\theta}}\log w_{\pi_{\boldsymbol{\theta}}}(H)\middle| H \sim \pi_e\right] && \text{(C.13)} \\
&\qquad = \mathbf{E}\left[-g(H)\sum_{t=0}^{L}\frac{\partial}{\partial \boldsymbol{\theta}}\log \pi_{\boldsymbol{\theta}}(A_t|S_t)\middle| H \sim \pi_e\right] && \text{(C.14)} \\
&\qquad = \mathbf{E}\left[-\mathrm{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}})\sum_{t=0}^{L}\frac{\partial}{\partial \boldsymbol{\theta}}\log \pi_{\boldsymbol{\theta}}(A_t|S_t)\middle| H \sim \pi_{\boldsymbol{\theta}}\right] && \text{(C.15)}
\end{aligned}
$$

□

Equation (C.15) is almost the same as the behavior policy gradient except the importance-sampled return is *not* squared. Recall from Section 3.4.2 that the interpretation of the behavior policy gradient update is to increase the probability of large magnitude returns. We can interpret the update given by Equation (C.15) as telling us to increase the probability of large, positive returns. Since the existence

of ${\pi_b}^\star$ depends on all returns being positive, both Equation (C.15) and the behavior policy gradient have the same interpretation.

The difference between Equation (C.15) and the behavior policy gradient theorem is that the importance-sampled return is *not* squared in Equation (C.15). Thus, the gradient update with (C.15) does not increase the probability of high magnitude trajectories to the same extent that BPG does. A practical consequence of this derivation is that estimates of (C.15) may have lower variance than estimates of the behavior policy gradient. Thus, it may be easier to use (C.15) in practice, provided the assumptions needed for ${\pi_b}^\star$ to exist hold.

C.3 Behavior Policy Gradient and Unbiasedness

In this section, we prove that the estimate of BPG is an unbiased estimate of $v(\pi_e)$. If only trajectories from a single $\pi_{\boldsymbol{\theta}_i}$ were used then clearly $\text{IS}(\pi_{\boldsymbol{\theta}}, H, \pi_{\boldsymbol{\theta}_i})$ is an unbiased estimate of $v(\pi_e)$. The difficulty is that the BPG's estimate at iteration n depends on all $\pi_{\boldsymbol{\theta}_i}$ for $i = 1 \dots n$ and each $\pi_{\boldsymbol{\theta}_i}$ is *not* independent of the others. Nevertheless, we prove here that BPG produces an unbiased estimate of $v(\pi_e)$ at each iteration.

The BPG estimate at iteration n is $\frac{1}{n}\sum_{i=1}^{n} \text{IS}(\pi_e, H_i, \pi_{\boldsymbol{\theta}_i})$. To make the dependence of $\boldsymbol{\theta}_i$ on $\boldsymbol{\theta}_{i-1}$ explicit, we will write $\boldsymbol{\theta}_i \coloneqq f(H_{i-1}, \boldsymbol{\theta}_{i-1})$ where $H_{i-1} \sim \pi_{\boldsymbol{\theta}_{i-1}}$ and f is the BPG policy update.

Proposition C.1. BPG *is an unbiased estimator of* $v(\pi_e)$.

Proof. To show unbiasedness, we must show that:

$$\mathbf{E}\left[\frac{1}{n}\sum_{i=1}^{n} \text{IS}(\pi_e, H_i, \pi_{\boldsymbol{\theta}_i}) \middle| \boldsymbol{\theta}_0 = \boldsymbol{\theta}_e\right] = v(\pi_e)$$

where the expectation is taken over the trajectories H_i.

$$
\begin{aligned}
&\mathbf{E}\left[\frac{1}{n}\sum_{i=1}^{n}\mathrm{IS}(\pi_e, H_i, \pi_{\boldsymbol{\theta}_i}) \;\middle|\; \boldsymbol{\theta}_0 = \boldsymbol{\theta}_e\right] \\
&\qquad = \sum_{h_0}\Pr(h_0|\boldsymbol{\theta}_0)\sum_{h_1}\Pr(h_1|f(h_0,\boldsymbol{\theta}_0))\cdots \\
&\qquad\quad \cdots\sum_{h_n}\Pr(h_n|f(h_{n-1},\boldsymbol{\theta}_{n-1}))\frac{1}{n}\sum_{i=1}^{n}\mathrm{IS}(\pi_e, H_i, \pi_{\boldsymbol{\theta}_i}) \\
&\qquad = \frac{1}{n}\sum_{i=1}^{n}\sum_{h_0}\Pr(h_0|\boldsymbol{\theta}_0)\sum_{h_1}\Pr(h_1|f(h_0,\boldsymbol{\theta}_0))\cdots \\
&\qquad\quad \cdots\sum_{h_n}\Pr(h_n|f(h_{n-1},\boldsymbol{\theta}_{n-1}))\,\mathrm{IS}(\pi_e, H_i, \pi_{\boldsymbol{\theta}_i}) \\
&\qquad = \frac{1}{n}\sum_{i=1}^{n}\sum_{h_0}\Pr(h_0|\boldsymbol{\theta}_0)\sum_{h_1}\Pr(h_1|f(h_0,\boldsymbol{\theta}_0))\cdots \\
&\qquad\quad \cdots\sum_{h_i}\Pr(h_i|f(h_{i-1},\boldsymbol{\theta}_{i-1}))\,\mathrm{IS}(\pi_e, H_i, \pi_{\boldsymbol{\theta}_i})\cdots \\
&\qquad\quad \cdots\underbrace{\sum_{h_{i+1}}\Pr(h_{i+1}|f(h_i,\boldsymbol{\theta}_i))\sum_{h_n}\Pr(h_n|f(h_{n-1},\boldsymbol{\theta}_{n-1}))}_{1} \\
&\qquad = \frac{1}{n}\sum_{i=1}^{n}\sum_{h_0}\Pr(h_0|\boldsymbol{\theta}_0)\sum_{h_1}\Pr(h_1|f(h_0,\boldsymbol{\theta}_0))\cdots \\
&\qquad\quad \cdots\underbrace{\sum_{h_i}\Pr(h_i|f(h_{i-1},\boldsymbol{\theta}_{i-1}))\,\mathrm{IS}(\pi_e, H_i, \pi_{\boldsymbol{\theta}_i})}_{v(\pi_e)} \\
&\qquad = \frac{1}{n}v(\pi_e)\sum_{i=1}^{n}\underbrace{\sum_{h_0}\Pr(h_0|\boldsymbol{\theta}_0)\sum_{h_1}\Pr(h_1|f(h_0))\cdots}_{1} \\
&\qquad = \frac{1}{n}v(\pi_e)n \\
&\qquad = v(\pi_e)
\end{aligned}
$$

□

Notice that, even though BPG's off-policy estimates are unbiased, they are *not* statistically independent. This means that concentration inequalities, like Hoeffding's inequality, cannot be applied directly. We conjecture that the conditional independence properties of BPG (specifically that H_i is independent of H_{i-1} given $\boldsymbol{\theta}_i$), are sufficient for Hoeffding's inequality to be applicable.

Appendix D

Weighting Off-Policy Data: Derivations and Proofs

This appendix includes the derivations of all theoretical results referenced in Chapter 5 and Chapter 6. Specifically, we show that regression importance sampling (RIS) is a consistent estimator of $v(\pi_e)$, RIS has lower asymptotic variance than OIS, and the SEC policy gradient estimator has lower variance than the Monte Carlo policy gradient estimator. We also derive a connection between RIS and the REG estimator of Li et al. (2015).

D.1 Regression Importance Sampling is Consistent

In this section we show that the regression importance sampling (RIS) estimator is a consistent estimator of $v(\pi_e)$ under two assumptions. The main intuition for this proof is that RIS is performing policy search on an estimate of the log-likelihood, $\widehat{\mathcal{L}}(\pi|\mathcal{D})$, as a surrogate objective for the true log-likelihood, $\mathcal{L}(\pi)$. Since π_b has generated our data, π_b is the optimal solution to this policy search. As long as, for all π, $\widehat{\mathcal{L}}(\pi|\mathcal{D})$ is a consistent estimator of $\mathcal{L}(\pi)$ then selecting $\pi_{\mathcal{D}} = \underset{\pi}{\operatorname{argmax}}\, \widehat{\mathcal{L}}(\pi|\mathcal{D})$

will converge probabilistically to π_b and the RIS estimator will be the same as the OIS estimator which is a consistent estimator of $v(\pi_e)$. If the set of policies we search over, Π, is countable then this argument is almost enough to show RIS to be consistent. The difficulty (as we explain below) arises when Π is *not* countable.

Our proof takes inspiration from Thomas and Brunskill who show that their magical policy search algorithm converges to the optimal policy by maximizing a surrogate estimate of policy value (2016b). They show that performing policy search on a policy value estimate, $\hat{v}(\pi)$, will almost surely return the policy that maximizes $v(\pi)$ if $\hat{v}(\pi)$ is a consistent estimator of $v(\pi)$. The proof is almost identical; the notable difference is substituting the log-likelihood, $\mathcal{L}(\pi)$, and a consistent estimator of the log-likelihood, $\widehat{\mathcal{L}}(\pi|\mathcal{D})$, in place of $v(\pi)$ and $\hat{v}(\pi)$.

D.1.1 Definitions and Assumptions

Let $\mathcal{H}_n$ be the set of all possible state-action trajectory segments with n states and $n-1$ actions:

$$\mathcal{H}_n = \mathcal{S}^n \times \mathcal{A}^{n-1}.$$

We will denote elements of $\mathcal{H}_n$ as h_n and random variables that take values from $\mathcal{H}_n$ as H_n. Let $d_{\pi_b,\mathcal{H}_n} : \mathcal{H}_n \to [0,1]$ be the distribution over elements of $\mathcal{H}_n$ induced by running π_b. In Chapter 2, we defined π_b to be a function mapping state-action pairs to probabilities. Here, we define $\pi_b : \mathcal{H}_n \times \mathcal{A} \to [0,1]$, i.e., a policy that conditions the distribution over actions on the preceding length n trajectory segment. These definitions are equivalent provided for any $h_{n,i} = (s_i, a_i, ...s_{i+n-1})$ and $h_{n,j} = (s_j, a_j, ...s_{j+n-1})$, if $s_{i+n-1} = s_{j+n-1}$ then $\forall a\ \pi_b(a|h_{n,i}) = \pi_b(a|h_{n,j})$.

Let $(\Omega, \mathcal{F}, \mu)$ be a probability space and $D_m : \Omega \to \mathcal{D}$ be a random variable. $D_m(\omega)$ is a sample of m trajectories with $\omega \in \Omega$. Let d_{π_b} be the distribution of states

under π_b. Define the expected log-likelihood:

$$\mathcal{L}(\pi) = \mathbf{E}[\log \pi(A|h_n)|h_n \sim d_{\pi_b,\mathcal{H}_n}, A \sim \pi_b]$$

and its sample estimate from samples in $D_m(\omega)$:

$$\widehat{\mathcal{L}}(\pi|D_m(\omega)) = \frac{1}{mL} \sum_{h \in D_m(\omega)} \sum_{t=0}^{L-1} \log \pi(a_t^h|h_{t-n,t}).$$

Note that:

$$\pi_b = \operatorname*{argmax}_{\pi \in \Pi} \mathcal{L}(\pi)$$

$$\pi_{\mathcal{D}}{}^{(n)} = \operatorname*{argmax}_{\pi \in \Pi} \widehat{\mathcal{L}}(\pi|D_m(\omega)).$$

Define the KL-divergence ($D_{\mathtt{KL}}$) between π_b and $\pi_{\mathcal{D}}$ after segment h_n as:

$$\delta_{\mathtt{KL}}(h_n) = D_{\mathtt{KL}}(\pi_b(\cdot|h_n), \pi_{\mathcal{D}}(\cdot|h_n)).$$

Assuming for all h_n and a the variance of $\log \pi(a|h_n)$ is bounded, $\widehat{\mathcal{L}}(\pi|D_m(\omega))$ is a consistent estimator of $\mathcal{L}(\pi)$. We make this assumption explicit:

Assumption D.1. *(Consistent Estimation of Log likelihood). For all* $\pi \in \Pi$, $\widehat{\mathcal{L}}(\pi|D_m(\omega)) \xrightarrow{a.s.} \mathcal{L}(\pi)$.

This assumption will hold when the support of π_b is a subset of the support of π for all $\pi \in \Pi$, i.e., no $\pi \in \Pi$ places zero probability measure on an action that π_b might take. We can ensure this assumption is satisfied by only considering $\pi \in \Pi$ that place non-zero probability on any action that π_b has taken.

We also make an additional assumption about the piece-wise continuity of the log-likelihood, $\mathcal{L}$, and the estimate of the log-likelihood, $\widehat{\mathcal{L}}$. First we present two necessary definitions as given by Thomas and Brunskill (2016b):

Definition D.1. *(Piecewise Lipschitz continuity). We say that a function $f : M \to \mathbb{R}$ on a metric space (M, d) is piecewise Lipschitz continuous with respect to Lipschitz constant K and with respect to a countable partition, $\{M_1, M_2, ...\}$ if f is Lipschitz continuous with Lipschitz constant K on all metric spaces in $\{(M_i, d_i)\}_{i=1}^{\infty}$.*

Definition D.2. *(δ-covering). If (M, d) is a metric space, a set $X \subset M$ is a δ-covering of (M, d) if and only if $\max_{y \in M} \min_{x \in X} d(x, y) \leq \delta$.*

Assumption D.2. *(Piecewise Lipschitz objectives). Our policy class, Π, is equipped with a metric, d_Π, such that for all $D_m(\omega)$ there exist countable partition of Π, $\Pi^{\mathcal{L}} \coloneqq \{\Pi_1^{\mathcal{L}}, \Pi_2^{\mathcal{L}}, ...\}$ and $\Pi^{\widehat{\mathcal{L}}} \coloneqq \{\Pi_1^{\widehat{\mathcal{L}}}, \Pi_2^{\widehat{\mathcal{L}}}, ...\}$, where $\mathcal{L}$ and $\widehat{\mathcal{L}}(\cdot|D_m(\omega))$ are piecewise Lipschitz continuous with respect to $\Pi^{\mathcal{L}}$ and $\Pi^{\widehat{\mathcal{L}}}$ with Lipschitz constants K and $\widehat{K}$ respectively. Furthermore, for all $i \in \mathbb{N}_{>0}$ and all $\delta > 0$ there exist countable δ-covers of $\Pi_i^{\mathcal{L}}$ and $\Pi_i^{\widehat{\mathcal{L}}}$.*

As pointed out by Thomas and Brunskill, this assumption holds for the most commonly considered policy classes but is also general enough to hold for other settings (see Thomas and Brunskill (2016b) for further discussion of Assumptions D.1 and D.2 and the related definitions).

D.1.2 Consistency Proof

Before proving Proposition 5.1 from Chapter 5, we prove a Lemma necessary for the final proof.

Lemma D.1. *If Assumptions D.1 and D.2 hold then* $\mathbf{E}[\delta_{\mathtt{KL}}(h_n)|h_n \sim d_{\pi_b, \mathcal{H}_n}] \xrightarrow{a.s.} 0$.

Proof. Define $\Delta(\pi, \omega) = |\widehat{\mathcal{L}}(\pi|D_m(\omega)) - \mathcal{L}(\pi)|$. From Assumption D.1 and one definition of almost sure convergence, for all $\pi \in \Pi$ and for all $\epsilon > 0$:

$$\Pr\left(\liminf_{m \to \infty} \{\omega \in \Omega : \Delta(\pi, \omega) < \epsilon\}\right) = 1. \tag{D.1}$$

Thomas and Brunskill point out that because Π may not be countable, (D.1) may not hold at the same time for all $\pi \in \Pi$. More precisely, it does *not* immediately follow that for all $\epsilon > 0$:

$$\Pr\left(\liminf_{m\to\infty}\{\omega \in \Omega : \forall \pi \in \Pi, \Delta(\pi, \omega) < \epsilon\}\right) = 1. \tag{D.2}$$

Let $C(\delta)$ denote the union of all of the policies in the δ-covers of the countable partitions of Π assumed to exist by Assumption 2. Since the partitions are countable and the δ-covers for each region are assumed to be countable, we have that $C(\delta)$ is countable for all δ. Thus, for all $\pi \in C(\delta)$, (D.1) holds simulatenously. More precisely, for all $\delta > 0$ and for all $\epsilon > 0$:

$$\Pr\left(\liminf_{m\to\infty}\{\omega \in \Omega : \forall \pi \in C(\delta), \Delta(\pi, \omega) < \epsilon\}\right) = 1. \tag{D.3}$$

Consider a $\pi \notin C(\delta)$. By the definition of a δ-cover and Assumption D.2, we have that $\exists \pi' \in \Pi_i^{\mathcal{L}}, d(\pi, \pi') \leq \delta$. Since Assumption D.2 requires $\mathcal{L}$ to be Lipschitz continuous on $\Pi_i^{\mathcal{L}}$, we have that $|\mathcal{L}(\pi) - \mathcal{L}(\pi')| \leq K\delta$. Similarly $|\widehat{\mathcal{L}}(\pi|D_m(\omega)) - \widehat{\mathcal{L}}(\pi'|D_m(\omega))| \leq \widehat{K}\delta$. So, $|\widehat{\mathcal{L}}(\pi|D_m(\omega)) - \mathcal{L}(\pi)| \leq |\widehat{\mathcal{L}}(\pi|D_m(\omega)) - \mathcal{L}(\pi')| + K\delta \leq |\widehat{\mathcal{L}}(\pi'|D_m(\omega)) - \mathcal{L}(\pi')| + (\widehat{K} + K)\delta$. Then it follows that for all $\delta > 0$:

$$(\forall \pi \in C(\delta), \Delta(\pi, \omega) \leq \epsilon) \to \left(\forall \pi \in \Pi, \Delta(\pi, \omega) < \epsilon + (K + \widehat{K})\delta\right).$$

Substituting this into (D.3) we have that for all $\delta > 0$ and for all $\epsilon > 0$:

$$\Pr\left(\liminf_{m\to\infty}\{\omega \in \Omega : \forall \pi \in \Pi, \Delta(\pi, \omega) < \epsilon + (K + \widehat{K})\delta\}\right) = 1.$$

The next part of the proof massages (D.3) into a statement of the same form

as (D.2). Consider the choice of $\delta \coloneqq \epsilon/(K+\widehat{K})$. Define $\epsilon' = 2\epsilon$. Then for all $\epsilon' > 0$:

$$\Pr\left(\liminf_{m\to\infty}\{\omega \in \Omega : \forall \pi \in \Pi, \Delta(\pi,\omega) < \epsilon'\}\right) = 1. \tag{D.4}$$

Since $\forall \pi \in \Pi, \Delta(\pi,\omega) < \epsilon'$, we obtain:

$$\Delta(\pi_b,\omega) < \epsilon' \tag{D.5}$$

$$\Delta(\pi_{\mathcal{D}},\omega) < \epsilon' \tag{D.6}$$

and then applying the definition of Δ:

$$\begin{aligned}
\mathcal{L}(\pi_{\mathcal{D}}) &\overset{(a)}{\leq} \mathcal{L}(\pi_b) && \text{(D.7)}\\
&\overset{(b)}{<} \widehat{\mathcal{L}}(\pi_b|D_m(\omega)) + \epsilon' && \text{(D.8)}\\
&\overset{(c)}{\leq} \widehat{\mathcal{L}}(\pi_{\mathcal{D}}|D_m(\omega)) + \epsilon' && \text{(D.9)}\\
&\overset{(d)}{\leq} \mathcal{L}(\pi_{\mathcal{D}}) + 2\epsilon' && \text{(D.10)}
\end{aligned}$$

where (a) comes from the fact that π_b maximizes $\mathcal{L}$, (b) comes from (D.5), (c) comes from the fact that $\pi_{\mathcal{D}}$ maximizes $\widehat{\mathcal{L}}(\cdot|D_m(\omega))$, and (d) comes from (D.6). Considering (D.7) and (D.10), it follows that $|\mathcal{L}(\pi_{\mathcal{D}}) - \mathcal{L}(\pi_b)| < 2\epsilon'$. Thus, (D.4) implies that:

$$\forall \epsilon' > 0, \Pr\left(\liminf_{m\to\infty}\{\omega \in \Omega : |\mathcal{L}(\pi_{\mathcal{D}}) - \mathcal{L}(\pi_b)| < 2\epsilon'\}\right) = 1.$$

Using $\epsilon'' \coloneqq 2\epsilon'$ we obtain:

$$\forall \epsilon'' > 0, \Pr\left(\liminf_{m\to\infty}\{\omega \in \Omega : |\mathcal{L}(\pi_{\mathcal{D}}) - \mathcal{L}(\pi_b)| < \epsilon''\}\right) = 1$$

From the definition of the KL-Divergence,

$$\mathcal{L}(\pi_{\mathcal{D}}) - \mathcal{L}(\pi_b) = \mathbf{E}[\delta_{\mathtt{KL}}(h_n)|h_n \sim d_{\pi_b,\mathcal{H}_n}]$$

and we obtain that:

$$\forall \epsilon > 0, \Pr\left(\liminf_{n\to\infty}\{\omega \in \Omega : |-\mathbf{E}[\delta_{\mathtt{KL}}(h_n)|h_n \sim d_{\pi_b,\mathcal{H}_n}]| < \epsilon\}\right) = 1$$

And finally, since the KL-Divergence is non-negative:

$$\forall \epsilon > 0, \Pr\left(\liminf_{m\to\infty}\{\omega \in \Omega : \mathbf{E}[\delta_{\mathtt{KL}}(h_n)|h_n \sim d_{\pi_b,\mathcal{H}_n}]| < \epsilon\}\right) = 1,$$

which, by the definition of almost sure convergence, means that

$$\mathbf{E}[\delta_{\mathtt{KL}}(h_n)|h_n \sim d_{\pi_b,\mathcal{H}_n}] \xrightarrow{a.s.} 0.$$

□

Proposition 5.1. *If Assumptions D.1 and D.2 hold, then* $\forall n$, $\mathrm{RIS}(n)$ *is a consistent estimator of* $v(\pi_e)$*:* $\mathrm{RIS}(n)(\pi_e, \mathcal{D}) \xrightarrow{a.s.} v(\pi_e)$.

Proof. Lemma D.1 shows that as the amount of data increases, the behavior policy estimated by RIS will almost surely converge to the true behavior policy:

$$\pi_{\mathcal{D}}{}^{(n)} \xrightarrow{a.s.} \pi_b.$$

Almost sure convergence to the true behavior policy means that RIS almost surely converges to the OIS estimate. Consider the difference, $\mathrm{RIS}(n)(\pi_e, \mathcal{D}) - \mathrm{OIS}(\pi_e, \mathcal{D})$. Since $\pi_{\mathcal{D}}{}^{(n)} \xrightarrow{a.s.} \pi_b$, we have that:

$$\mathrm{RIS}(n)(\pi_e, \mathcal{D}) - \mathrm{OIS}(\pi_e, \mathcal{D}) \xrightarrow{a.s.} 0.$$

Thus, with probability 1, RIS(n) and OIS converge to the same value. Since OIS is a consistent estimator of $v(\pi_e)$, then with probability 1 we have that $\text{OIS}(\pi_e, \mathcal{D})$ converges to $v(\pi_e)$. Thus $\text{RIS}(n)(\pi_e, \mathcal{D}) \xrightarrow{a.s.} v(\pi_e)$. □

D.2 Asymptotic Variance of Regression Importance Sampling

In this section we prove that, $\forall n$, RIS(n) has asymptotic variance at most that of OIS. We give this result as a corollary to Theorem 1 of Henmi et al. (2007) that holds for general Monte Carlo integration. Note that while we define distributions as probability mass functions, this result can be applied to continuous-valued state and action spaces by replacing probability mass functions with density functions.

Corollary 5.1. *Let $\Pi^n_{\boldsymbol{\theta}}$ be a class of twice differentiable policies, $\pi_{\boldsymbol{\theta}}(\cdot|s_{t-n}, a_{t-n}, \ldots, s_t)$. If $\exists\tilde{\boldsymbol{\theta}}$ such that $\pi_{\tilde{\boldsymbol{\theta}}} \in \Pi^n_{\boldsymbol{\theta}}$ and $\pi_{\tilde{\boldsymbol{\theta}}} = \pi_b$ then*

$$\text{Var}_{\mathtt{A}}(\text{RIS}(n)(\pi_e, \mathcal{D})) \leq \text{Var}_{\mathtt{A}}(\text{OIS}(\pi_e, \mathcal{D}, \pi_b))$$

where $\text{Var}_{\mathtt{A}}$ *denotes the asymptotic variance.*

Corollary 5.1 states that the asymptotic variance of RIS(n) must be at least as low as that of OIS.

We first present Theorem 1 from Henmi et al. (2007) and adopt their notation for its presentation. Consider estimating $v = \mathbf{E}_p[f(x)]$ for probability mass function p and real-valued function f. Given parameterized and twice differentiable probability mass function $q(\cdot|\tilde{\boldsymbol{\theta}})$, we define the ordinary importance sampling estimator of v as $\tilde{v} = \frac{1}{m}\sum_{i=1}^m \frac{p(x_i)}{q(x_i,\tilde{\boldsymbol{\theta}})} f(x_i)$. Similarly, define $\hat{v} = \frac{1}{m}\sum_{i=1}^m \frac{p(x_i)}{q(x_i,\hat{\boldsymbol{\theta}})} f(x_i)$ where $\hat{\boldsymbol{\theta}}$ is the maximum likelihood estimate of $\tilde{\boldsymbol{\theta}}$ given samples from $q(\cdot|\tilde{\boldsymbol{\theta}})$. The following theorem relates the asymptotic variance of $\hat{v}$ to that of $\tilde{v}$.

Theorem D.1.

$$\mathrm{Var}_{\mathtt{A}}(\hat{v}) \leq \mathrm{Var}_{\mathtt{A}}(\tilde{v})$$

where $\mathrm{Var}_{\mathtt{A}}$ *denotes the asymptotic variance.*

Proof. See Theorem 1 of Henmi et al. (2007). □

Theorem D.1 shows that the maximum likelihood estimated parameters of the sampling distribution yield an asymptotically lower variance estimate than using the true parameters, $\tilde{\boldsymbol{\theta}}$. To specialize this theorem to our setting, we show that the maximum likelihood behavior policy parameters are also the maximum likelihood parameters for the trajectory distribution of the behavior policy. First specify the class of sampling distribution: $\Pr(h;\boldsymbol{\theta}) = p(h)w_{\pi_{\boldsymbol{\theta}}}(h)$ where $p(h) = d_0(s_0)\prod_{t=1}^{L-1} P(s_t|s_{t-1},a_{t-1})$ and $w_{\pi_{\boldsymbol{\theta}}}(h) = \prod_{t=0}^{L-1}\pi_{\boldsymbol{\theta}}(a_t|s_{t-n},a_{t-n},\ldots,s_t)$. We now present the following lemma:

Lemma D.2.

$$\begin{aligned}&\operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{h\in\mathcal{D}}\sum_{t=0}^{L-1}\log\pi_{\boldsymbol{\theta}}(a_t|s_{t-n},a_{t-n},\ldots,s_t)\\ =&\operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{h\in\mathcal{D}}\log\Pr(h;\boldsymbol{\theta})\end{aligned}$$

Proof.

$$
\begin{aligned}
&\operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{h\in\mathcal{D}} \sum_{t=0}^{L-1} \log \pi_{\boldsymbol{\theta}}(a_t|s_{t-n}, a_{t-n}, \ldots, s_t) \\
=&\operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{h\in\mathcal{D}} \sum_{t=0}^{L-1} \log \pi_{\boldsymbol{\theta}}(a_t|s_{t-n}, a_{t-n}, \ldots, s_t) \\
&\underbrace{+\log d(s_0) + \sum_{t=1}^{L-1} \log P(s_t|s_{t-1}, a_{t-1})}_{\text{const w.r.t. } \boldsymbol{\theta}} \\
=&\operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{h\in\mathcal{D}} \log w_{\pi_{\boldsymbol{\theta}}}(h) + \log p(h) \\
\boldsymbol{\theta} =&\operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{h\in\mathcal{D}} \log \Pr(h;\boldsymbol{\theta})
\end{aligned}
$$

□

Finally, we combine Lemma D.2 with Theorem D.1 to prove Corollary 5.1:

Corollary 5.1. *Let $\Pi^n_{\boldsymbol{\theta}}$ be a class of policies, $\pi_{\boldsymbol{\theta}}(\cdot|s_{t-n}, a_{t-n}, \ldots, s_t)$ that are twice differentiable with respect to $\boldsymbol{\theta}$. If $\exists \boldsymbol{\theta} \in \Pi^n_{\boldsymbol{\theta}}$ such that $\pi_{\boldsymbol{\theta}} = \pi_b$, then*

$$\mathrm{Var}_{\mathtt{A}}(\mathrm{RIS}(n)(\pi_e, \mathcal{D})) \leq \mathrm{Var}_{\mathtt{A}}(\mathrm{OIS}(\pi_e, \mathcal{D}))$$

where $\mathrm{Var}_{\mathtt{A}}$ *denotes the asymptotic variance.*

Proof. Define $f(h) = g(h)$, $p(h) = \Pr(h|\pi_e)$ and $q(h|\boldsymbol{\theta}) = \Pr(h|\pi_{\boldsymbol{\theta}})$. Lemma D.2 implies that:

$$\hat{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta}\in\Pi_{\boldsymbol{\theta}}} \sum_{h\in\mathcal{D}} \sum_{t=0}^{L-1} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)$$

is the maximum likelihood estimate of $\tilde{\boldsymbol{\theta}}$ (where $\pi_{\tilde{\boldsymbol{\theta}}} = \pi_b$ and $\Pr(h|\tilde{\boldsymbol{\theta}})$ is the probability of h under π_b) and then Corollary 5.1 follows directly from Theorem D.1. □

Note that for RIS(n) with $n > 0$, the condition that $\pi_{\tilde{\boldsymbol{\theta}}} \in \Pi^n$ can hold even if

the distribution of $A_t \sim \pi_{\tilde{\boldsymbol{\theta}}}$ (i.e., $A_t \sim \pi_b$) is only conditioned on s_t. This condition holds when $\exists \pi_{\boldsymbol{\theta}} \in \Pi^n$ such that $\forall s_{t-n}, a_{t-n}, \dots a_{t-1}$:

$$\pi_{\tilde{\boldsymbol{\theta}}}(a_t|s_t) = \pi_{\boldsymbol{\theta}}(a_t|s_{t-n}, a_{t-n}, \dots, s_t),$$

i.e., the action probabilities only vary with respect to s_t.

D.3 Connection between RIS and REG

In this section we show that RIS($L-1$) is an approximation of the REG estimator studied by Li et al. (2015). This connection is notable because Li et al. showed REG is asymptotically minimax optimal, however, in MDPs, REG requires knowledge of the environment's state transition probabilities and initial state distribution probabilities while RIS($L-1$) does not (2015). For this discussion, we recall the definition of the probability of a trajectory for a given MDP and policy:

$$\Pr(h|\pi) = d_0(s_0)\pi(a_0|s_0)P(s_1|s_0, a_0) \cdots P(s_{L-1}|s_{L-2}, a_{L-2})\pi(a_{L-1}|s_{L-1}).$$

We also define $\mathcal{H}$ to be the set of all state-action trajectories possible under π_b of length L: $s_0, a_0, ...s_{L-1}, a_{L-1}$.

Li et al. introduce the regression estimator (REG) for multi-armed bandit problems (2015). This method estimates the mean reward for each action as $\hat{r}(a, \mathcal{D})$ and then computes the REG estimate as:

$$\text{REG}(\pi_e, \mathcal{D}) = \sum_{a \in \mathcal{A}} \pi_e(a)\hat{r}(a, \mathcal{D}).$$

This estimator is identical to RIS(0) in multi-armed bandit problems (Li et al., 2015). The extension of REG to finite-horizon MDPs estimates the mean return for each

trajectory as $\hat{g}(h, \mathcal{D})$ and then computes the estimate:

$$\text{REG}(\pi_e, \mathcal{D}) = \sum_{h \in \mathcal{H}} \Pr(h|\pi_e)\hat{g}(h, \mathcal{D}).$$

Since this estimate uses $\Pr(h|\pi_e)$ it requires knowledge of the initial state distribution, d_0, and transition probabilities, P.

We now elucidate a relationship between RIS$(L-1)$ and REG even though they are different estimators. Let $c(h)$ denote the number of times that trajectory h appears in $\mathcal{D}$. Similar to the bandit case, we can rewrite REG as an importance sampling method:

$$\begin{aligned}
\text{REG}(\pi_e, \mathcal{D}) =& \sum_{h \in \mathcal{H}} \Pr(h|\pi_e)\hat{g}(h, \mathcal{D}) && \text{(D.11)}\\
=& \frac{1}{m} \sum_{h \in \mathcal{H}} c(h) \frac{\Pr(h|\pi_e)}{c(h)/m} \hat{g}(h, \mathcal{D}) && \text{(D.12)}\\
=& \frac{1}{m} \sum_{i=1}^{m} \frac{\Pr(h_i|\pi_e)}{c(h_i)/m} g(h_i) && \text{(D.13)}
\end{aligned}$$

The denominator in (D.13) can be re-written as a telescoping product to obtain an estimator that is similar to RIS$(L-1)$:

$$\begin{aligned}
\text{REG}(\pi_e, \mathcal{D}) =& \frac{1}{m} \sum_{i=1}^{m} \frac{\Pr(h_i|\pi_e)}{c(h_i)/m} g(h_i)\\
=& \frac{1}{m} \sum_{i=1}^{m} \frac{\Pr(h_i|\pi_e)}{\frac{c(s_0)}{m} \frac{c(s_0,a_0)}{c(s_0)} \cdots \frac{c(h_i)}{c(h_i/a_{L-1})}} g(h_i)\\
=& \frac{1}{m} \sum_{i=1}^{m} \frac{d_0(s_0)\pi_e(a_0|s_0)P(s_1|s_0,a_0)\cdots}{\hat{d}(s_0)\pi_{\mathcal{D}}(a_0|s_0)\hat{P}(s_1|s_0,a_0)\cdots}\\
& \frac{\cdots P(s_{L-1}|s_{L-2},a_{L-2})\pi_e(a_{L-1}|s_{L-1})}{\cdots \hat{P}(s_{L-1}|h_{0:L-1})\pi_{\mathcal{D}}(a_{L-1}|h_{i:j})} g(h_i).
\end{aligned}$$

This expression differs from RIS$(L-1)$ in two ways:

1. The numerator includes the initial state distribution and transition probabilities

of the environment.

2. The denominator includes count-based estimates of the initial state distribution and transition probabilities of the environment where the transition probabilities are conditioned on all past states and actions.

If we assume that the empirical estimates of the environment probabilities in the denominator are equal to the true environment probabilities then these factors cancel and we obtain the RIS($L-1$) estimate. This assumption will almost always be false except in deterministic environments. However, showing that RIS($L-1$) is approximating REG suggests that RIS($L-1$) may have similar theoretical properties to those derived for REG by Li et al. (2015). Our SinglePath experiment (See Figure 5.3 in Chapter 5) supports this conjecture: RIS($L-1$) has high bias in the low to medium sample size but have asymptotically lower MSE compared to other methods. REG has even higher bias in the low to medium sample size range but has asymptotically lower MSE compared to RIS($L-1$). RIS with smaller n appear to decrease the initial bias but have larger MSE as the sample size grows. The asymptotic benefit of RIS for all n is also corroborated by Corollary 5.1 in Appendix D.2 though Corollary 5.1 does *not* tell us anything about how different RIS methods compare. The asymptotic benefit of REG compared to RIS methods can be understood as REG correcting for sampling error in both the action selection and state transitions.

D.4 Sampling Error Corrected Policy Gradient Estimator Variance

In this section we prove Proposition 6.1 from Chapter 6. That is, we show – under a specific set of assumptions – that the sampling error corrected policy gradient estimator has lower variance than the Monte Carlo policy gradient estimator.

Before we present the proof, we recall the assumptions made in Section 6.3:

1. The action space is discrete and if a state is observed then all actions have also been observed in that state.

2. The return estimates $\hat{q}^{\pi_{\boldsymbol{\theta}}}$ for any $(s, a, \cdot)$ is a fixed constant that is independent of $\mathcal{T}$.

3. For all observed states, our estimated policy π_ϕ is equal to $\pi_{\mathcal{T}}$, i.e., if action a occurs k times in state s and s occurs n times in $\mathcal{T}$ then $\pi_\phi(a|s) = \frac{k}{n}$.

See Section 6.3 for discussion of these assumptions and Section 6.4 for experimental results with relaxed assumptions.

Recall that $\mathcal{T}$ is a set of state-action pairs collected by running the current policy $\pi_{\boldsymbol{\theta}}$. Let $\mathbb{S}$ be the random variable representing the states observed in $\mathcal{T}$ and let $\mathbb{A}$ be the random variable representing the actions observed in $\mathcal{T}$. We will sometimes write $\{\mathbb{S}, \mathbb{A}\}$ in place of $\mathcal{T}$ to make the composition of $\mathcal{T}$ explicit. Let $\mathrm{Var}_{\mathbb{X}}(g)$ denote the variance of estimator g with respect to random variable $\mathbb{X}$. Let $\mathrm{Var}_{\mathbb{X}}(g|\mathbb{Y})$ denote the variance of estimator g with respect to random variable $\mathbb{X}$ *given a fixed value for* $\mathbb{Y}$.

Under our assumptions, we make two claims about the SEC gradient estimate, $g_{\texttt{sec}}$:

Claim 6.1. $\mathrm{Var}_{\mathbb{A}}(g_{\texttt{sec}}(\{\mathbb{S}, \mathbb{A}\}|\mathbb{S})) = 0$.

Proof. Recall from Chapter 6, that we can write either $g_{\texttt{mc}}$ or $g_{\texttt{sec}}$ as:

$$g(\{\mathbb{S}, \mathbb{A}\}) = \sum_{s \in \mathcal{S}} d_{\mathcal{T}}(s) \sum_{a \in \mathcal{A}} \pi_{\mathcal{T}}(a|s) w(s, a) \bar{q}(s, a) \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s). \tag{D.14}$$

In Claim 6.1, the sampled states are fixed and variance only arises from $\pi_{\mathcal{T}}$ and $w(s, a)$ which vary for different realizations of $\mathbb{A}$. When we choose $w(s, a) = \frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\mathcal{T}}(a|s)}$

(as SEC does) the $\pi_{\mathcal{T}}(a|s)$ factors cancel in D.14. Since $\pi_{\mathcal{T}}$ is the only part of $g_{\texttt{sec}}$ that depends on the random variable $\mathbb{A}$, using $w(s,a)$ eliminates variance due to action selection in the estimator. This proves Claim 6.1.

□

Claim 6.2. $\mathbf{E}_{\mathbb{A}}\left[g_{\texttt{sec}}(\{\mathbb{S},\mathbb{A}\})|\mathbb{S}\right] = \mathbf{E}_{\mathbb{A}}\left[g_{\texttt{mc}}(\mathbb{S},\mathbb{A})|\mathbb{S}\right]$.

Proof. Claim 6.2 also follows from the same logic as Claim 6.1. The cancellation of the $\pi_{\mathcal{T}}(a|s)$ factors converts the inner summation over actions into an exact expectation under $\pi_{\boldsymbol{\theta}}$. Since $g_{\texttt{mc}}$ is an unbiased estimator, the inner summation over actions must be equal to the exact expectation under $\pi_{\boldsymbol{\theta}}$ *in expectation*. Thus the expectation of both estimators *conditioned on* $\mathbb{S}$ is:

$$\mathbf{E}_{\mathbb{A}}\left[g(\{\mathbb{S},\mathbb{A}\})|\mathbb{S}\right] = \sum_{s\in\mathcal{S}} d_{\mathcal{T}}(s) \sum_{a\in\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) w(s,a) \bar{q}(s,a) \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s). \tag{D.15}$$

This proves Claim 6.2. □

We can now prove Proposition 6.1.

Proposition 6.1. *Let* $\mathbb{S}$ *be the random variable representing the states observed in* $\mathcal{T}$ *and let* $\mathbb{A}$ *be the random variable representing the actions observed in* $\mathcal{T}$*. Let* $\mathrm{Var}_{\mathbb{S},\mathbb{A}}(g)$ *denote the variance of estimator* g *with respect to random variables* $\mathbb{S}$ *and* $\mathbb{A}$*. For the Monte Carlo estimator,* $g_{\texttt{mc}}$*, and the* SEC *estimator,* $g_{\texttt{sec}}$*:*

$$\mathrm{Var}_{\mathbb{S},\mathbb{A}}\left(g_{\texttt{sec}}(\{\mathbb{S},\mathbb{A}\})\right) \leq \mathrm{Var}_{\mathbb{S},\mathbb{A}}\left(g_{\texttt{mc}}(\{\mathbb{S},\mathbb{A}\})\right)$$

Proof. Again, recall that both $g_{\texttt{sec}}$ and $g_{\texttt{mc}}$ can be written as:

$$g(\{\mathbb{S},\mathbb{A}\}) = \sum_{s\in\mathcal{S}} d_{\mathcal{T}}(s) \sum_{a\in\mathcal{A}} \pi_{\mathcal{T}}(a|s) w(s,a) \bar{q}^{\pi_{\boldsymbol{\theta}}}(s,a) \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) \tag{D.16}$$

where $w(s,a) = \frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\mathcal{T}}(a|s)}$ for $g_{\texttt{sec}}$ and $w(s,a) = 1$ for $g_{\texttt{mc}}$. Using the law of total variance, the variance of (D.16) can be decomposed as:

$$\text{Var}_{\mathbb{S},\mathbb{A}}\left(g(\{\mathbb{S},\mathbb{A}\})\right) = \underbrace{\mathbf{E}_{\mathbb{S}}\left[\text{Var}_{\mathbb{A}}\left(g(\{\mathbb{S},\mathbb{A}\}|\mathbb{S})\right)\right]}_{\Sigma_{\mathbb{A}}} + \underbrace{\text{Var}_{\mathbb{S}}\left(\mathbf{E}_{\mathbb{A}}\left[g(\{\mathbb{S},\mathbb{A}\})|\mathbb{S}\right]\right)}_{\Sigma_{\mathbb{S}}}$$

The first term, $\Sigma_{\mathbb{A}}$, is the variance due to stochasticity in the action selection. From Claim 6.1, we know that for $g_{\texttt{sec}}$ this term is at most that for $g_{\texttt{mc}}$. The second term, $\Sigma_{\mathbb{S}}$, is the variance due to only visiting a limited number of states before estimating the gradient. We know this term is equal for both $g_{\texttt{sec}}$ and $g_{\texttt{mc}}$. Thus the variance of $g_{\texttt{sec}}$ is at most that of $g_{\texttt{mc}}$. □

Appendix E

Model-based Error: Derivations and Proofs

This appendix proves the bounds on error in a model-based estimate of $v(\pi_e)$ from Chapter 8.

E.1 Error Bound in Terms of Trajectories

In this section we make the additional assumption that the true MDP reward function is known and bounded in $[0, r_{\texttt{max}}]$. This assumption is only used for these theoretical results.

This section will reference the MDP of interest, $\mathcal{M}$, and a model of that MDP, $\widehat{\mathcal{M}}$. The model is an identical MDP to $\mathcal{M}$ except for the transition function and initial state distribution. To make this difference explicit we will use $P_{\mathcal{M}}$ and $d_{0,\mathcal{M}}$ to refer to these probabilities for $\mathcal{M}$ and we will use $P_{\widehat{\mathcal{M}}}$ and $d_{0,\widehat{\mathcal{M}}}$ to refer to these probabilities for $\widehat{\mathcal{M}}$.

E.1.1 On-Policy Model Error Bound

Lemma E.1. *For any policy π, let $\Pr(\cdot|\pi, \mathcal{M})$ be the distribution of trajectories generated by π and $\Pr(\cdot|\pi, \widehat{\mathcal{M}})$ be the distribution of trajectories generated by π in an approximate model, $\widehat{\mathcal{M}}$. The error of the estimate, $v(\pi, \widehat{\mathcal{M}})$, under $\widehat{\mathcal{M}}$ is upper bounded by:*

$$\left|v(\pi, \mathcal{M}) - v(\pi, \widehat{\mathcal{M}})\right| \leq 2\sqrt{2}L \cdot r_{\texttt{max}} \sqrt{D_{\texttt{KL}}(\Pr(\cdot|\pi, \mathcal{M}) || \Pr(\cdot|\pi, \widehat{\mathcal{M}}))}$$

where $D_{\texttt{KL}}(\Pr(\cdot|\pi, \mathcal{M}) || \Pr(\cdot|\pi, \widehat{\mathcal{M}}))$ is the Kullback-Leibler (KL) divergence between probability distributions $\Pr(\cdot|\pi, \mathcal{M})$ and $\Pr(\cdot|\pi, \widehat{\mathcal{M}})$.

Proof.

$$\left|v(\pi, \mathcal{M}) - v(\pi, \widehat{\mathcal{M}})\right| = \left|\sum_h \Pr(h|\pi, \mathcal{M}) g(h) - \sum_h \Pr(h|\pi, \widehat{\mathcal{M}}) g(h)\right|$$

From Jensen's inequality and the fact that $g(h) \geq 0$:

$$\left|v(\pi, \mathcal{M}) - v(\pi, \widehat{\mathcal{M}})\right| \leq \sum_h \left|\Pr(h|\pi, \mathcal{M}) - \Pr(h|\pi, \widehat{\mathcal{M}})\right| g(h).$$

After replacing $g(h)$ with the maximum possible return, $g_{\texttt{max}} \coloneqq L \cdot r_{\texttt{max}}$, and factoring it out of the summation, we can use the definition of the total variation distance between two probability distributions ($D_{\texttt{TV}}(p||q) = \frac{1}{2} \sum_x |p(x) - q(x)|$) to obtain:

$$\left|v(\pi, \mathcal{M}) - v(\pi, \widehat{\mathcal{M}})\right| \leq 2 D_{\texttt{TV}}(\Pr(\cdot|\pi, \mathcal{M}) || \Pr(\cdot|\pi, \widehat{\mathcal{M}})) \cdot g_{\texttt{max}}.$$

The definition of $g_{\texttt{max}}$ and Pinsker's inequality ($D_{\texttt{TV}}(p||q) \leq \sqrt{2 D_{\texttt{KL}}(p||q)}$) completes the proof. □

E.1.2 Off-Policy Model Error Bound

We now use Lemma E.1 to prove the main theoretical result from Chapter 8.

Theorem 8.1. *For* MDP $\mathcal{M}$*, any policies* π_e *and* π_b*, and an approximate model,* $\widehat{\mathcal{M}}$*, estimated with i.i.d. trajectories,* $H \sim \Pr(\cdot|\pi_b, \mathcal{M})$*, the error in the model-based estimate of* $v(\pi_e, \mathcal{M})$ *with* $\widehat{\mathcal{M}}$*,* $v(\pi_e, \widehat{\mathcal{M}})$*, is upper bounded by:*

$$\left|v(\pi_e, \widehat{\mathcal{M}}) - v(\pi_e, \mathcal{M})\right| \leq 2\sqrt{2}L \cdot r_{\texttt{max}} \sqrt{\mathbf{E}\left[\rho_{L-1}^{(H)} \log \frac{\Pr(H|\pi_e, \mathcal{M})}{\Pr(H|\pi_e, \widehat{\mathcal{M}})} \middle| H \sim \pi_b\right]}$$

where $\rho_{L-1}^{(H)}$ *is the importance weight of trajectory* H *at step* L*.*

Proof. Theorem 8.1 follows from Lemma E.1 with the importance-sampling identity (i.e., importance-sampling the expectation in Lemma E.1 so that it is an expectation with $H \sim \pi_b$). The transition probabilities cancel in the importance weight, $\frac{\Pr(H|\pi_e, mdp)}{\Pr(H|\pi_b, mdp)}$, leaving us with the importance weight $\rho_{L-1}^{(H)}$ and completing the proof. □

E.1.3 Bounding Theorem 8.1 in terms of a Supervised Loss Function

We now express Theorem 8.1 in terms of an expectation over transitions that occur along sampled trajectories.

Corollary 8.1. *For* MDP*,* $\mathcal{M}$*, any policies* π_e *and* π_b *and an approximate model,* $\widehat{\mathcal{M}}$*, with transition probabilities,* $P_{\widehat{\mathcal{M}}}$*, estimated with trajectories* $H \sim \pi_b$*, the bias of the approximate model's estimate of* $v(\pi_e, \mathcal{M})$*,* $v(\pi_e, \widehat{\mathcal{M}})$*, is upper bounded by:*

$$|v(\pi_e, \widehat{\mathcal{M}}) - v(\pi_e, \mathcal{M})| \leq 2\sqrt{2}L \cdot r_{\texttt{max}} \sqrt{\epsilon_0 + \sum_{t=1}^{L-1} \mathbf{E}\left[\rho_t^{(H)} \epsilon(S_t, A_t) \middle| S_t, A_t \sim d_{\pi_b, \mathcal{M}}^t\right]}$$

where $d_{\pi_b, \mathcal{M}}^t$ *is the distribution of states and actions observed at time* t

when executing π_b *in the true* MDP, $\epsilon_0 := D_{\mathtt{KL}}(d_{0,\mathcal{M}}||d_{0,\widehat{\mathcal{M}}})$, *and* $\epsilon(s,a) = D_{\mathtt{KL}}(P_{\mathcal{M}}(\cdot|s,a)||P_{\widehat{\mathcal{M}}}(\cdot|s,a)))$.

Corollary 8.1 follows from Theorem 8.1 by equating the expectation to an expectation in terms of (S_t, A_t, S_{t+1}) samples:

Proof.

$$\mathbf{E}\left[\rho_L^H \log \frac{\Pr(H|\pi_e,\mathcal{M})}{\Pr(H|\pi_e,\widehat{\mathcal{M}})}\middle| H \sim \pi_b\right] = \sum_h \Pr(h|\pi_b,\mathcal{M})\rho_{L-1}^{(H)} \log \frac{\Pr(h|\pi_e,\mathcal{M})}{\Pr(h|\pi_e,\widehat{\mathcal{M}})}$$

$$= \sum_{s_0}\sum_{a_0}\cdots\sum_{s_{L-1}}\sum_{a_{L-1}} d_{0,\mathcal{M}}(s_0)\pi_b(a_0|s_0)\cdots P_{\mathcal{M}}(s_{L-1}|s_{L-2},a_{L-2})\cdot$$
$$\pi_b(a_{L-1}|s_{L-1})\rho_{L-1}^{(H)} \log \frac{d_{0,\mathcal{M}}(s_0)\cdots P_{\mathcal{M}}(s_{L-1}|s_{L-2},a_{L-2})}{d_{0,\widehat{\mathcal{M}}}(s_0)\cdots P_{\widehat{\mathcal{M}}}(s_{L-1}|s_{L-2},a_{L-2})}$$

Using the logarithm property that $\log(ab) = \log(a) + \log(b)$ and rearranging the summations allows us to marginalize out the probabilities that do not appear in the logarithm.

$$= \sum_{s_0} d_{0,\mathcal{M}}(s_0) \log \frac{d_{0,\mathcal{M}}(s_0)}{d_{0,\widehat{\mathcal{M}}}(s_0)} + \sum_{t=1}^{L-1}\sum_{s_0} d_{0,\mathcal{M}}(s_0)\cdots$$
$$\sum_{s_t} \rho_L^H P_{\mathcal{M}}(s_t|s_{t-1},a_{t-1}) \log \frac{P_{\mathcal{M}}(s_t|s_{t-1},a_{t-1})}{P_{\widehat{\mathcal{M}}}(s_t|s_{t-1},a_{t-1})} \tag{E.1}$$

Define the probability of observing s and a at time $t+1$ when following π_b in MDP $\mathcal{M}$ as:

$$d_{\pi_b,\mathcal{M}}^{t+1}(s,a) := \sum_{s_t,a_t} d_{\pi_b,\mathcal{M}}^{t}(s_t,a_t)P_{\mathcal{M}}(s|s_t,a_t)\pi_b(a|s)$$

where $d_{\pi_b,\mathcal{M}}^{1}(s,a) := d_{0,\mathcal{M}}(s)\pi_b(a|s)$. Using this definition, we can simplify Equation

E.1:

$$= D_{\mathtt{KL}}(d_{0,\mathcal{M}}||d_{0,\widehat{\mathcal{M}}})+$$
$$\sum_{t=1}^{L-1} \mathbf{E}\left[\rho_{L-1}^{(H)} D_{\mathtt{KL}}(P_{\mathcal{M}}(\cdot|S_t,A_t)||P_{\widehat{\mathcal{M}}}(\cdot|S_t,A_t))\Big|S_t,A_t \sim d_{\pi_b,\mathcal{M}}^t\right]$$

□

We relate $D_{\mathtt{KL}}$ to two common supervised learning loss functions so that we can express Corollary 8.1 with training error over (s_t, a_t, s_{t+1}) samples. $D_{\mathtt{KL}}(P||Q) = H[P,Q] - H[P]$ where $H[P]$ is the entropy of P and $H[P,Q]$ is the cross-entropy of P with respect to Q. For distributions of discrete random variables, $H[P,Q] - H[P] \leq H[P,Q]$ since entropy is always positive. This fact allows us to upper bound $D_{\mathtt{KL}}$ with the cross-entropy loss function. The cross-entropy loss function is equivalent to the expected negative log likelihood loss function:

$$H[P_{\mathcal{M}}(\cdot|s,a), P_{\widehat{\mathcal{M}}}(\cdot|s,a)] = \mathbf{E}_{S'\sim P_{\mathcal{M}}(\cdot|s,a)}[-\log P_{\widehat{\mathcal{M}}}(S'|s,a)].$$

Thus building a maximum likelihood model corresponds to minimizing this model error bound. For continuous domains where the transition function is a probability density function, entropy can be negative so the negative log-likelihood or cross-entropy loss functions will not always upper bound model error. In this case, our bound approximates the true error bound to within a constant.

E.2 Finite-sample Error Bound

Theorem 8.1 can be expressed as a finite-sample bound by applying Hoeffding's inequality to bound an estimate of the expectation for a finite number of observed trajectories.

Corollary 8.2. *For* MDP $\mathcal{M}$*, any policies* π_e *and* π_b *and an approximate model,* $\widehat{\mathcal{M}}$*, with transition probabilities,* $P_{\widehat{\mathcal{M}}}$*, estimated with* (s, a) *transitions from trajectories* $H \sim \pi_b$*, and after observing* m *trajectories then with probability* α*, the error of the approximate model's estimate of* $v(\pi_e, \mathcal{M})$*,* $v(\pi_e, \widehat{\mathcal{M}})$*, is upper bounded by:*

$$\left| v(\pi_e, \widehat{\mathcal{M}}) - v(\pi_e, \mathcal{M}) \right| \leq 2L \cdot r_{\texttt{max}} \cdot$$

$$\sqrt{2\bar{\rho}_{L-1}\sqrt{\frac{\ln(\frac{1}{\alpha})}{2m}} - \frac{1}{m}\sum_{j=1}^{m} \rho_{L-1}^{j} \left(\log d_{0,\widehat{\mathcal{M}}}(s_1^j) + \sum_{t=1}^{L-1} \log P_{\widehat{\mathcal{M}}}(s_{t+1}^j | s_t^j, a_t^j) \right)}$$

where $\bar{\rho}_{L-1}$ *is an upper bound on the importance ratio, i.e., for all* h*,* $\rho_{L-1}^{(h)} < \bar{\rho}_{L-1}$*.*

Proof. Corollary 8.2 follows from applying Hoeffding's Inequality to Theorem 8.1 and then expanding $D_{\texttt{KL}}(\Pr(\cdot|\pi_b, \mathcal{M}) || \Pr(\cdot|\pi_b, \widehat{\mathcal{M}}))$ to be in terms of samples as done in the derivation of Corollary 8.1. We then drop logarithm terms which contain the unknown $d_{0,\mathcal{M}}$ and $P_{\mathcal{M}}$ probabilities. Dropping these terms is equivalent to expressing Corollary 8.2 in terms of the cross-entropy or negative log-likelihood loss functions.

□

Appendix F

Additional Empirical Details

This appendix contains additional details of experiments described in Chapters 3, 4, 5, and 8. All other chapters that include experiments contain a full description of those experiments.

F.1 Chapter 3: Behavior Policy Search

This appendix contains experimental details in addition to the details contained in Chapter 3.

F.1.1 Grid World

This domain is a 4x4 Grid World with a terminal state with reward 10 at $(3, 3)$, a state with reward -10 at $(1, 1)$, a state with reward 1 at $(1, 3)$, and all other states having reward -1. The action set contains the four cardinal directions and actions move the agent in its intended direction (except when moving into a wall which produces no movement). The agent begins in $(0, 0)$, $\gamma = 1$, and $L = 100$. All policies use softmax action selection with temperature 1 where the probability of taking an

action a in a state s is given by:

$$\pi(a|s) = \frac{e^{\theta_{sa}}}{\sum_{a'} e^{\theta_{sa'}}}.$$

We obtain two evaluation policies by applying REINFORCE to this task, starting from a policy that selects actions uniformly at random. We then select one evaluation policy from the early stages of learning – an improved policy but still far from converged –, π_1, and one after learning has converged, π_2. We run our set of experiments once with $\pi_e \coloneqq \pi_1$ and a second time with $\pi_e \coloneqq \pi_2$. The ground truth value of $v(\pi_e)$ is computed with finite-horizon dynamic programming for both π_e.

F.1.2 Continuous Control

We evaluate BPG on two continuous control tasks: Cart Pole Swing Up and Acrobot. Both tasks are implemented within RLLAB (Duan et al., 2016). The single task modification we make is that in Cart Pole Swing Up, when a trajectory terminates due to moving out of bounds we give a penalty of -1000. This modification increases the variance of π_e. We use $\gamma = 1$ and $L = 50$. Policies are represented as conditional Gaussians with mean determined by a neural network with two hidden layers of 32 tanh units each and a state-independent diagonal covariance matrix. In Cart Pole Swing Up, π_e was learned with 10 iterations of the TRPO algorithm (Schulman et al., 2015a) applied to a randomly initialized policy. In Acrobot, π_e was learned with 60 iterations. The ground truth value of $v(\pi_e)$ in both domains is computed with 1,000,000 Monte Carlo roll-outs.

F.1.3 Domain Independent Details

In all experiments we subtract a constant control variate (or baseline) in the gradient estimate from Theorem 3.1. The baseline, b_i is an estimate of:

$$\mathbf{E}\left[-\operatorname{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}_{i-1}})^2 \middle| H \sim \pi_{\boldsymbol{\theta}_{i-1}}\right]$$

and our new gradient is an estimate of:

$$\mathbf{E}\left[\left(-\operatorname{IS}(\pi_e, H, \pi_{\boldsymbol{\theta}})^2 - b_i\right) \sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \middle| H \sim \pi_{\boldsymbol{\theta}}\right].$$

Adding or subtracting a constant does not change the gradient in expectation since $b_i \cdot \mathbf{E}\left[\sum_{t=0}^{L-1} \frac{\partial}{\partial \boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t)\right] = 0$. However, the baseline variant of BPG has lower variance behavior policy gradient estimates so that the estimated gradient is closer in direction to the true behavior policy gradient.

We use batch sizes of 100 trajectories per iteration for Grid World experiments and size 500 for the continuous control tasks. The step-size parameter was determined by a sweep over $[10^{-2}, 10^{-6}]$.

F.2 Chapter 4: Parallel Policy Search

Trust-region policy optimization (TRPO) is an on-policy policy gradient algorithm that optimizes the target policy parameters, $\boldsymbol{\theta}$, according with the constrained optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\theta}'}{\text{maximize}} \quad & \frac{1}{m} \sum_{j=1}^{m} \frac{\pi_{\boldsymbol{\theta}'}(a_j|s_j)}{\pi_{\boldsymbol{\theta}}(a_j|s_j)} \hat{q}^{\pi_{\boldsymbol{\theta}}}(s_j, a_j, \cdot) \\ \text{subject to} \quad & \frac{1}{m} \sum_{j=1}^{m} D_{\mathtt{KL}}(\pi_{\boldsymbol{\theta}}(\cdot|s_j), \pi_{\boldsymbol{\theta}}{}'(\cdot|s_j)) < \epsilon \end{aligned} \tag{F.1}$$

where $\hat{q}^{\pi_{\boldsymbol{\theta}}}(s, a, \cdot)$ is an estimate of $q^{\pi_e}(s, a, \cdot)$, $D_{\mathtt{KL}}$ is the KL-divergence and ϵ is a constant hyperparameter. In our implementation we use the sum of rewards following taking action a in state s as our estimate of $\hat{q}^{\pi_{\boldsymbol{\theta}}}(s, a, \cdot)$.

The TRPO optimization problem uses m on-policy state-action pairs. We can make the algorithm off-policy by replacing $\hat{q}^{\pi_e}$ with an off-policy estimate of the return. Let ρ_j be the importance weight for the entire trajectory in which s_j and a_j occurred, i.e., the product of importance ratios from step 0 to step $L-1$. We replace $\hat{q}^{\pi_{\boldsymbol{\theta}}}(s_j, a_j, \cdot)$ with $\rho_j \hat{q}^{\pi_{\boldsymbol{\theta}}}(s_j, a_j, \cdot)$ to obtain the off-policy variant of TRPO.[34] Parallel policy search uses this off-policy variant of TRPO to update the target policy.

Finally, parallel policy search uses a similar optimization problem to update the behavior policy. Let $g(h_j)$ be the return of the trajectory that contains (s_j, a_j). Our implementation of parallel policy search updates the behavior policy with the constrained optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\theta}'}{\text{maximize}} \quad & \frac{1}{m} \sum_{j=1}^{m} \frac{\pi_{\boldsymbol{\theta}'}(a_j|s_j)}{\pi_{\boldsymbol{\theta}}(a_j|s_j)} (\rho_j g(h_j))^2 \\ \text{subject to} \quad & \frac{1}{m} \sum_{j=1}^{m} D_{\mathtt{KL}}(\pi_{\boldsymbol{\theta}}(\cdot|s_j), \pi_{\boldsymbol{\theta}}{}'(\cdot|s_j)) < \epsilon_b \end{aligned} \tag{F.2}$$

where ϵ_b is a constant hyperparameter.

F.3 Chapter 5: Regression Importance Sampling

In this section we provide additional details for the experiments described in Chapter 5.

[34] We use the importance weight for the entire trajectory to correct for the change in the state distribution up to step t as well as the return distribution following step t where t is the time when (s_j, a_j) was observed.

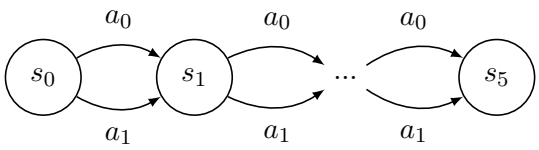


Figure F.1: The SinglePath MDP referenced in Section 4 of the main text. **Not shown:** If the agent takes action a_1 it remains in its current state with probability 0.5.

F.3.1 SinglePath

This environment is shown in Figure F.1 with horizon $L = 5$. In each state, π_b selects action, a_0, with probability $p = 0.6$ and π_e selects action, a_0, with probability $1 - p = 0.4$. Action a_0 causes a deterministic transition to the next state. Action a_1 causes a transition to the next state with probability 0.5, otherwise, the agent remains in its current state. The agent receives a reward of 1 for action a_0 and 0 otherwise. RIS uses count-based estimation of π_b and REG uses count-based estimation of trajectories. REG is also given the environment's state transition function, P.

F.3.2 Grid World

We use the same Grid World domain described in Section F.1.1. The off-policy set of experiments uses a behavior policy, π_b, that can reach the high reward terminal state and an evaluation policy, π_e, that is the same policy with lower entropy action selection. The on-policy set of experiments uses the same behavior policy as both behavior and evaluation policy. RIS estimates the behavior policy with the empirical frequency of actions in each state. This domain allows us to study RIS separately from questions of function approximation.

F.3.3 Linear Dynamical System

This domain is a point-mass agent moving towards a goal in a two dimensional world by setting x and y acceleration. The agent acts for $L = 20$ time-steps under linear-gaussian dynamics and receives a reward that is proportional to its distance from the goal. We use second order polynomial basis functions so that policies are non-linear in the state features but we can still estimate $\pi_{\mathcal{D}}$ efficiently with ordinary least squares. We obtain a basic policy by optimizing the parameters of a policy for 10 iterations of the Cross-Entropy optimization method (Rubinstein and Kroese, 2013). The basic policy maps the state to the mean of a Gaussian distribution over actions. The evaluation policy uses a standard deviation of 0.5 and the true π_b uses a standard deviation of 0.6.

F.3.4 Continuous Control

We also use two continuous control tasks from the OpenAI gym: Hopper and HalfCheetah[35] In each task, we use neural network policies with 2 layers of 64 hidden units each for π_e and π_b. Each policy maps the state to the mean of a Gaussian distribution with state-independent standard deviation. We obtain π_e and π_b by running the OpenAI Baselines (Dhariwal et al., 2017) version of proximal policy optimization (PPO) (Schulman et al., 2017) and then selecting two policies along the learning curve. For both environments, we use the policy after 30 updates for π_e and after 20 updates for π_b. These policies use tanh activations on their hidden units since these are the default in the OpenAI Baselines PPO implementation,

RIS estimates the behavior policy with gradient descent on the negative log-likelihood of the neural network. In all our experiments we use a learning rate of 1×10^{-3}. The multi-layer behavior policies learned by RIS have either 0, 1, 2, or 3 hidden layers with 64 hidden units with relu activations.

[35] For these tasks we use the Roboschool versions: https://github.com/openai/roboschool

F.4 Chapter 8: Combining Simulated with Off-Policy Data

In this section we provide additional details for the experiments described in Chapter 8.

F.4.1 Mountain Car

The first domain is a discretized version of the Mountain Car task from the RL literature (Sutton and Barto, 1998). In this domain an agent attempts to drive an underpowered car up a hill. States are discretized horizontal position and velocity and the agent may choose to accelerate left, right, or neither. At each time-step the reward is -1 except for in a terminal state (the top of the hill) when it is 0.

As in previous work on importance sampling, we shorten the horizon of the problem by holding action a_t constant for 4 updates of the environment state (Jiang and Li, 2016; Thomas, 2015). This modification changes the problem horizon to $L = 100$ and is done to reduce the variance of importance-sampling.

Policy π_b chooses actions uniformly at random and π_e is a sub-optimal policy that solves the task in approximately 35 steps. We use Monte Carlo rollouts to estimate $v(\pi_e, \widehat{\mathcal{M}})$ for the model-based estimator.

In this domain we build tabular models which cannot generalize from observed (s, a) pairs. As done by Jiang and Li (2016), we assume that a lack of data for a (s, a) pair causes a deterministic transition to s. We compute the model action value function, $\hat{q}_{\pi_e}$, and state value function, $\hat{v}_{\pi_e}$ with value-iteration for WDR.

F.4.2 Cliff World

Our second domain is a continuous two-dimensional Cliff World (depicted in Figure F.2) where a point mass agent navigates a series of cliffs to reach a goal, $\mathbf{g}$. An

Figure F.2: Cliff World domain in which an agent (A) must move between or around cliffs to reach a goal (G).

agent's state is a four dimensional vector of horizontal and vertical position and velocity. Actions are acceleration values in the horizontal and vertical directions. The reward is negative and proportional to the agent's distance to the goal and magnitude of the actions taken, $r(\mathbf{s}, \mathbf{a}) = ||\mathbf{s} - \mathbf{g}||_1 + ||\mathbf{a}||_1$. If the agent falls off a cliff it receives a large negative penalty. Domain dynamics are linear with additive Gaussian noise.

To create π_b and π_e, we first hand code a deterministic policy, π_d. Then the agent samples $\pi_e(\cdot|s)$ by sampling from $\mathcal{N}(\cdot|\pi_d(s), \Sigma)$ with $\Sigma = 0.5^2 I$. The behavior policy is the same except $\Sigma = I$.

We build models in two ways: linear regression (converges to true model as $m \to \infty$) and regression over nonlinear polynomial basis functions.[36] The first model class choice represents the ideal case and the second is the case when the true dynamics are outside the learnable model class. Our results refer to MB-BOOTSTRAP$^{\text{LR}}$ and MB-BOOTSTRAP$^{\text{PR}}$ as the MB estimator using linear regression and polynomial regression respectively. Similarly, we evaluate WDR-BOOTSTRAP$^{\text{LR}}$ and WDR-BOOTSTRAP$^{\text{PR}}$. These dynamics mean that the bootstrap models of MB-BOOTSTRAP$^{\text{LR}}$ and WDR-BOOTSTRAP$^{\text{LR}}$ will quickly converge to a correct

[36]For each state feature, x, we include features 1, x^2, x^3 but not x.

model as the amount of data increases since they build models with linear regression. On the other hand, these dynamics mean that the models of MB-BOOTSTRAP$^{\mathrm{PR}}$ and WDR-BOOTSTRAP$^{\mathrm{PR}}$ will quickly converge to an incorrect model since they use regression over nonlinear polynomial basis functions.

Bibliography

Abbeel, P., Quigley, M., and Ng, A. Y. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006. URL http://dl.acm.org/citation.cfm?id=1143845. 3, 150

Arouna, B. Adaptative Monte Carlo method, a variance reduction technique. *Monte Carlo Methods and Applications (MCMA)*, 10(1):1–24, 2004. 30

Asadi, K., Allen, C., Roderick, M., Mohamed, A.-r., Konidaris, G., and Littman, M. Mean actor critic. *arXiv preprint arXiv:1709.00503v1*, 2017. 84, 149

Ashar, J., Ashmore, J., Hall, B., Harris, S., Hengst, B., Liu, R., Mei, Z., Pagnucco, M., Roy, R., Sammut, C., Sushkov, O., Teh, B., and Tsekouras, L. RoboCup SPL 2014 champion team paper. In Bianchi, R. A. C., Akin, H. L., Ramamoorthy, S., and Sugiura, K., editors, *RoboCup 2014: Robot World Cup XVIII*, volume 8992 of *Lecture Notes in Artificial Intelligence*, pages 70–81. Springer International Publishing, 2015. 115

Asis, K. D., Hernandez-Garcia, J. F., Holland, G. Z., and Sutton, R. S. Multi-step reinforcement learning: A unifying algorithm. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 149

Austin, P. C. An introduction to propensity score methods for reducing the effects

of confounding in observational studies. *Multivariate behavioral research*, 46(3): 399–424, 2011. 146

Baird, L. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*. 1995. 157

Bertsekas, D. P. and Tsitsiklis, J. N. Gradient convergence in gradient methods with errors. *SIAM Journal on Optimization*, 10:627–642, 2000. 35

Boeing, A. and Bräunl, T. Leveraging multiple simulators for crossing the reality gap. In *Proceedings of the 12th International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 1113–1119. IEEE, 2012. 152

Bottou, L., Peters, J., Candela, J. Q., Charles, D. X., Chickering, M., Portugaly, E., Ray, D., Simard, P. Y., and Snelson, E. Counterfactual reasoning and learning systems: the example of computational advertising. *Journal of Machine Learning Research*, 14(1):3207–3260, 2013. 154

Bouchard, G., Trouillon, T., Perez, J., and Gaidon, A. Online learning to sample. *arXiv preprint arXiv:1506.09016*, 2016. 144

Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., Konolige, K., Levine, S., and Vanhoucke, V. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018. 154

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016. 52, 95

Brown, D. S. and Niekum, S. Efficient probabilistic performance bounds for inverse

reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 155

Chambless, L. E., Folsom, A. R., Sharrett, A. R., Sorlie, P., Couper, D., Szklo, M., and Nieto, F. J. Coronary heart disease risk prediction in the atherosclerosis risk in communities (aric) study. *Journal of Clinical Epidemiology*, 56(9):880–890, 2003. 129

Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., and Fox, D. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019. 152

Chow, Y., Petrik, M., and Ghavamzadeh, M. Robust policy optimization with baseline guarantees. *arXiv preprint arXiv:1506.04514*, 2015. 155

Christiano, P., Shah, Z., Mordatch, I., Schneider, J., Blackwell, T., Tobin, J., Abbeel, P., and Zaremba, W. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016. 103, 153

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Proceedings of the 32nd Conference on Advances in Neural Information Processing Systems (NeurIPS)*, pages 4754–4765, 2018. 156

Ciosek, K. and Whiteson, S. OFFER: Off-environment reinforcement learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, 2017. 30, 143, 144

Ciosek, K. and Whiteson, S. Expected policy gradients. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 84, 140, 149, 174

Cohen, A., Yu, L., and Wright, R. Diverse exploration for fast and safe policy improvement. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 155

Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. Robots that can adapt like animals. *Nature*, 521(7553):503, 2015. 153

Cutler, M. and How, J. P. Efficient reinforcement learnng for robots using informative simulated priors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015. 153

Cutler, M., Walsh, T. J., and How, J. P. Reinforcement learning with multi-fidelity simulators. In *Proceedings of the IEEE Conference on Robotics and Automation (ICRA)*, 2014. URL http://www.research.rutgers.edu/~thomaswa/pub/icra2014Car.pdf. 3, 150

Deisenroth, M. P. and Rasmussen, C. E. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011. 153

Delyon, B. and Portier, F. Integral approximation by kernel smoothing. *Bernoulli*, 22(4):2177–2208, 2016. 146

Desai, P. Y. and Glynn, P. W. Simulation in optimization and optimization in simulation: A Markov chain perspective on adaptive Monte Carlo algorithms. In *Proceedings of the 33rd conference on Winter simulation*, pages 379–384. IEEE Computer Society, 2001. 30, 143

Devin, C., Gupta, A., Darrell, T., Abbeel, P., and Levine, S. Learning modular neural network policies for multi-task and multi-robot transfer. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2169–2176. IEEE, 2017. 154

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. OpenAI baselines. https://github.com/openai/baselines, 2017. 54, 95, 221

Doroudi, S., Thomas, P. S., and Brunskill, E. Importance sampling for fair policy selection. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 2017. 80

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016. 39, 217

Dudík, M., Langford, J., and Li, L. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011. 23, 71, 147

Dvoretzky, A., Kiefer, J., and Wolfowitz, J. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics*, 27(3):642–669, 1956. 169

Efron, B. Better bootstrap confidence intervals. *Journal of the American Statistical Association*, 82(397):171–185, 1987. 128

Fang, K., Bai, Y., Hinterstoisser, S., and Kalakrishnan, M. Multi-task domain adaptation for deep learning of instance grasping from simulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018. 154

Farajtabar, M., Chow, Y., and Ghavamzadeh, M. More robust doubly robust off-policy evaluation. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018. 71, 147

Farchy, A., Barrett, S., MacAlpine, P., and Stone, P. Humanoid robots learning to walk faster: From the real world to simulation and back. In *Proceedings*

of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2013. URL http://www.cs.utexas.edu/users/ai-lab/?AAMAS13-Farchy. 7, 103, 104, 122

Fellows, M., Ciosek, K., and Whiteson, S. Fourier policy gradients. *arXiv preprint arXiv:1802.06891*, 2018. 149

Frank, J., Mannor, S., and Precup, D. Reinforcement learning in the presence of rare events. In *Proceedings of the 25th International Conference on Machine learning*, pages 336–343. ACM, 2008. 30, 143

Fu, J. and Topcu, U. Probably approximately correct mdp learning and control with temporal logic constraints. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2014. 155

Ghahramani, Z. and Rasmussen, C. E. Bayesian monte carlo. In *Proceedings of the 16th Conference on Advances in Neural Information Processing Systems (NeurIPS)*, pages 505–512, 2003. 63

Ghavamzadeh, M., Petrik, M., and Chow, Y. Safe policy improvement by minimizing robust baseline regret. In *Proceedings of the 30th Conference on Advances in Neural Information Processing Systems (NeurIPS)*, pages 2298–2306, 2016. 155

Golemo, F., Taiga, A. A., Courville, A., and Oudeyer, P.-Y. Sim-to-real transfer with neural-augmented robot simulation. In *Proceedings of the 2nd Conference on Robot Learning (CORL)*, pages 817–828, 2018. 151

Greensmith, E., Bartlett, P. L., and Baxter, J. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530, 2004. 28

Gruslys, A., Azar, M. G., Bellemare, M. G., and Munos, R. The reactor: A

sample-efficient actor-critic architecture. *arXiv preprint arXiv:1704.04651v1*, 2017. 147

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017a. 50, 174

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., Schölkopf, B., and Levine, S. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In *Proceedings of the 31st Conference on Advances in Neural Information Processing Systems (NeurIPS)*, 2017b. 27

Hall, B., Harris, S., Hengst, B., Liu, R., Ng, K., Pagnucco, M., Pearson, L., Sammut, C., and Schmidt, P. RoboCup SPL 2015 champion team paper. In *RoboCup 2015: Robot World Cup XIX*, volume 9513 of *Lecture Notes in Artificial Intelligence*, pages 72–82. Springer International Publishing, 2016. 115

Hanna, J. P. and Stone, P. Grounded action transformation for robot learning in simulation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, 2017. 103

Hanna, J. P. and Stone, P. Towards a data efficient off-policy policy gradient algorithm. In *Proceedings of the AAAI Spring Symposium on Data Efficient Reinforcement Learning*, 2018. 49

Hanna, J. P. and Stone, P. Reducing sampling error in the monte carlo policy gradient estimator. In *Proceedings of the 19th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 2019. 83

Hanna, J. P., Stone, P., and Niekum, S. Bootstrapping with models: Confidence intervals for off-policy evaluation. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 2017a. 126

Hanna, J. P., Thomas, P. S., Stone, P., and Niekum, S. Data-efficient policy evaluation through behavior policy search. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017b. 29

Hanna, J. P., Stone, P., and Niekum, S. Importance sampling with an estimated behavior policy. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019. 60

Hansen, N., Müller, S. D., and Koumoutsakos, P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003. 112

Hengst, B. rUNSWift walk2014 report robocup standard platform league. Technical report, The University of New South Wales, 2014. 115

Henmi, M., Yoshida, R., and Eguchi, S. Importance sampling via the estimated sampler. *Biometrika*, 94(4):985–991, 2007. 70, 146, 201, 202

Hester, T. and Stone, P. Real time targeted exploration in large domains. In *Proceedings of the 9th International Conference on Development and Learning (ICDL)*, August 2010. 155

Hirano, K., Imbens, G. W., and Ridder, G. Efficient estimation of average treatment effects using the estimated propensity score. *Econometrica*, 71(4):1161–1189, 2003. 146

Iocchi, L., Libera, F. D., and Menegatti, E. Learning humanoid soccer actions interleaving simulated and real data. In *Proceedings of the 2nd Workshop on Humanoid Soccer Robots*, 2007. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.139.4931. 151

Jakobi, N., Husbands, P., and Harvey, I. Noise and the reality gap: The use of

simulation in evolutionary robotics. In *Proceedings of the European Conference on Artificial Life*, pages 704–720. Springer, 1995. 117, 152

James, S., Wohlhart, P., Kalakrishnan, M., Kalashnikov, D., Irpan, A., Ibarz, J., Levine, S., Hadsell, R., and Bousmalis, K. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12627–12637, 2019. 154

Jiang, N. and Li, L. Doubly robust off-policy evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016. 8, 23, 36, 63, 74, 154, 222

Kober, J., Bagnell, J. A., and Peters, J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 2013. URL http://ijr.sagepub.com/content/early/2013/08/22/0278364913495721.abstract. 3, 102

Koos, S., Mouret, J.-B., and Doncieux, S. Crossing the reality gap in evolutionary robotics by promoting transferable controllers. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 119–126. ACM, 2010. URL http://dl.acm.org/citation.cfm?id=1830505. 151

Lee, G., Srinivasa, S. S., and Mason, M. T. GP-ILQG: Data-driven robust optimal control for uncertain nonlinear dynamical systems. *arXiv preprint arXiv:1705.05344*, 2017. 151

Levine, S. and Koltun, V. Guided policy search. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 1–9, 2013. 147

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. 2

Li, L., Munos, R., and Szepesvári, C. Toward minimax off-policy value estimation. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, 2015. 61, 70, 73, 75, 145, 164, 170, 194, 204, 206

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 2

Liu, Q. and Lee, J. D. Black-box importance sampling. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017. 146

Liu, Y., Gottesman, O., Raghu, A., Komorowski, M., Faisal, A. A., Doshi-Velez, F., and Brunskill, E. Representation balancing MDPs for off-policy policy evaluation. In *Proceedings of the 31st Conference on Advances in Neural Information Processing Systems*, 2018. 17

Lowrey, K., Kolev, S., Dao, J., Rajeswaran, A., and Todorov, E. Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system. In *Proceedings of the IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 35–42. IEEE, 2018. 151, 152

MacAlpine, P., Depinet, M., and Stone, P. UT austin villa 2014: Robocup 3D simulation league champion via overlapping layered learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pages 2842–2848, 2015. 2

Mahmood, A. R., Yu, H., and Sutton, R. S. Multi-step off-policy learning without importance sampling ratios. *arXiv preprint arXiv:1702.03006*, 2017. 157

Marco, A., Berkenkamp, F., Hennig, P., Schoellig, A. P., Krause, A., Schaal, S., and Trimpe, S. Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with bayesian optimization. In *Proceedings of the IEEE*

International Conference on Robotics and Automation (ICRA), pages 1557–1563. IEEE, 2017. 153

Miglino, O., Lund, H. H., and Nolfi, S. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4):417–434, 1996. 152

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 2

Molchanov, A., Chen, T., Hönig, W., Preiss, J. A., Ayanian, N., and Sukhatme, G. S. Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors. *arXiv preprint arXiv:1903.04628*, 2019. 152

Mozifian, M., Higuera, J. C. G., Meger, D., and Dudek, G. Learning domain randomization distributions for transfer of locomotion policies. *arXiv preprint arXiv:1906.00410*, 2019. 152

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. In *Proceedings of the 29th Conference on Advances in Neural Information Processing Systems (NeurIPS)*, pages 1054–1062, 2016. 157

Muratore, F., Treede, F., Gienger, M., and Peters, J. Domain randomization for simulation-based policy optimization with transferability assessment. In *Proceedings of the 2nd Conference on Robot Learning (CORL)*, pages 700–713, 2018. 152

Narita, Y., Yasui, S., and Yata, K. Efficient counterfactual learning from bandit feedback. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, 2019. 61, 145

Oates, C. J., Girolami, M., and Chopin, N. Control functionals for monte carlo integration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):695–718, 2017. 146

O'Hagan, A. Monte carlo is fundamentally unsound. *The Statistician*, pages 247–249, 1987. 63

OpenAI, Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018. 152

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. In *Proceedings of the 30th Conference on Advances in Neural Information Processing Systems (NeurIPS)*, pages 4026–4034, 2016. 155

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 152

Petit, B., Amdahl-Culleton, L., Liu, Y., Smith, J., and Bacon, P.-L. All-action policy gradient methods: A numerical integration approach. *arXiv preprint arXiv:1910.09093*, 2019. 149

Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., and Abbeel, P. Asymmetric actor critic for image-based robot learning. *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2017a. 154

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017b. 152

Precup, D., Sutton, R. S., and Singh, S. Eligibility traces for off-policy policy evaluation. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, pages 759–766, 2000. 18, 21, 63, 74, 149, 157

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014. 14

Raghu, A., Gottesman, O., Liu, Y., Komorowski, M., Faisal, A., Doshi-Velez, F., and Brunskill, E. Behaviour policy estimation in off-policy policy evaluation: Calibration matters. In *Proceedings of the ICML Workshop on Causal Inference, Counterfactual Prediction, and Autonomous Action*, 2018. 89, 147

Rajeswaran, A., Ghotra, S., Levine, S., and Ravindran, B. EPOpt: Learning robust neural network policies using model ensembles. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 152

Ramos, F., Possas, R. C., and Fox, D. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*, 2019. 152

Rodriguez, D., Brandenburger, A., and Behnke, S. Combining simulations and real-robot experiments for bayesian optimization of bipedal gait stabilization. In *RoboCup 2018: Robot World Cup XXII*, volume 11374 of *Lecture Notes in Artificial Intelligence*. Springer International Publishing, 2019. 151

Rosenbaum, P. R. Model-based direct adjustment. *Journal of the American Statistical Association*, 82(398):387–394, 1987. 146

Rubinstein, R. Y. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997. 30, 143

Rubinstein, R. Y. The cross-entropy method for combinatorial and continuous

optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999. 182, 188

Rubinstein, R. Y. and Kroese, D. P. *The cross-entropy method: a unified approach to combinatorial optimization, Monte Carlo simulation and machine learning*. Springer Science & Business Media, 2013. 221

Rubinstein, R. Y. and Kroese, D. P. *Simulation and the Monte Carlo method*, volume 10. John Wiley & Sons, 2016. 31

Rummery, G. A. and Niranjan, M. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, England, 1994. 148

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016a. 153

Rusu, A. A., Vecerik, M., Rothörl, T., Heess, N., Pascanu, R., and Hadsell, R. Sim-to-real robot learning from pixels with progressive nets. In *Proceedings of the 1st Conference on Robot Learning (CORL)*, 2016b. 103, 153

Sadeghi, F. and Levine, S. $(\text{CAD})^2$ RL: Real single-image flight without a single real image. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2017. 154

Schulman, J., Levine, S., Moritz, P., Jordan, M., and Abbeel, P. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015a. URL http://jmlr.csail.mit.edu/proceedings/papers/v37/schulman15.html. 39, 54, 217

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. High-dimensional

continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015b. 95

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2, 27, 50, 221

Sen, P. K. and Singer, J. M. *Large Sample Methods in Statistics: An Introduction with Applications*. Chapman & Hall, 1993. 20

Shi, L., Li, S., Cao, L., Yang, L., and Pan, G. TBQ (σ): Improving efficiency of trace utilization for off-policy reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1025–1032, 2019. 149

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. 2

Strehl, A., Langford, J., Li, L., and Kakade, S. M. Learning from logged implicit exploration data. In *Proceedings of the 23rd Conference on Advances in Neural Information Processing Systems (NeurIPS)*, pages 2217–2225, 2010. 147

Sutton, R. S. *Temporal credit assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst, 1984. 84

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 1998. 1, 56, 84, 149, 156, 157, 165, 222

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 13th*

Conference on Advances in Neural Information Processing Systems (NeurIPS), 2000a. 28, 37

Sutton, R. S., Singh, S., and McAllester, D. Comparing policy-gradient algorithms. 2000b. 84, 149

Sutton, R. S., Mahmood, A. R., and White, M. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 17(1):2603–2631, 2016. 157

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2018. 151

Thomas, P. S. *Safe Reinforcement Learning*. PhD thesis, University of Massachusetts Amherst, 2015. 22, 73, 154, 222

Thomas, P. S. and Brunskill, E. Data-efficient off-policy policy evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016a. 8, 17, 18, 23, 24, 25, 36, 37, 63, 73, 74, 156

Thomas, P. S. and Brunskill, E. Magical policy search: Data efficient reinforcement learning with guarantees of global optimality. *European Workshop On Reinforcement Learning*, 2016b. 195, 196, 197

Thomas, P. S., Theocharous, G., and Ghavamzadeh, M. High confidence off-policy evaluation. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, 2015a. 2, 29, 127, 129, 154

Thomas, P. S., Theocharous, G., and Ghavamzadeh, M. High confidence policy improvement. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015b. 126, 128, 136, 154, 155

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proceedings of the 30th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017. 103, 154

Tobin, J., Zaremba, W., and Abbeel, P. Domain randomization and generative models for robotic grasping. *Proceedings of the 31st IEEE/RSJ International Conference on Intelligent RObots and Systems (IROS)*, 2018. 154

Tzeng, E., Coline, D., Hoffman, J., Finn, C., Xingchao, P., Levine, S., Saenko, K., and Darrell, T. Towards adapting deep visuomotor representations from simulated to real environments. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2016. 154

Urieli, D., MacAlpine, P., Kalyanakrishnan, S., Bentor, Y., and Stone, P. On optimizing interdependent skills: A case study in simulated 3D humanoid robot soccer. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 769–776, 2011. 112

Van Seijen, H., Van Hasselt, H., Whiteson, S., and Wiering, M. A theoretical and empirical analysis of expected SARSA. In *Proceedings of the IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 177–184. IEEE, 2009. 148

Veness, J., Lanctot, M., and Bowling, M. Variance reduction in Monte-Carlo tree search. In *Proceedings of the 24th Conference on Advances in Neural Information Processing Systems (NeurIPS)*, pages 1836–1844, 2011. 24, 143

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and Freitas, N.de. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016. 27

Watkins, C. J. C. H. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989. 84

White, M. and Bowling, M. Learning a value analysis tool for agent evaluation. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1976–1981, 2009. 24, 143

White, M. and White, A. Interval estimation for reinforcement-learning algorithms in continuous-state domains. In *Proceedings of the 24th Conference on Advances in Neural Information Processing Systems (NeurIPS)*, pages 2433–2441, 2010. 156

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 28, 38, 39, 49, 93

Xie, Y., Liu, B., Liu, Q., Wang, Z., Zhou, Y., and Peng, J. Off-policy evaluation and learning from logged bandit feedback: Error reduction via surrogate policy. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 61, 146

Yang, L., Shi, M., Zheng, Q., Meng, W., and Pan, G. A unified approach for multi-step temporal-difference learning with eligibility traces in reinforcement learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018. 149

Zhang, F., Leitner, J., Upcroft, B., and Corke, P. Vision-based reaching using modular deep networks: from simulation to the real world. *arXiv preprint arXiv:1610.06781*, 2016. 154

Zhu, S., Kimmel, A., E. Bekris, K., and Boularias, A. Fast model identification via physics engines for data-efficient policy search. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3249–3256, 07 2018. doi: 10.24963/ijcai.2018/451. 151

Zinkevich, M., Bowling, M., Bard, N., Kan, M., and Billings, D. Optimal unbiased estimators for evaluating agent performance. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 573–578, 2006. 24, 143