Efficient Motor Control Learning

Gregory Donnell Lawrence

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2009-53
http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-53.html

April 30, 2009

Efficient Motor Control Learning

by

Gregory Donnell Lawrence

B.S. (University of California, Berkeley) 1998

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Stuart J. Russell, Chair
Professor Sosale S. Sastry
Professor Peter Bartlett
Professor Robert J. Full

Spring 2009

The dissertation of Gregory Donnell Lawrence is approved:

Chair Date

Date

Date

Date

University of California, Berkeley

Spring 2009

Efficient Motor Control Learning

Gregory Donnell Lawrence

Abstract

Efficient Motor Control Learning

by

Gregory Donnell Lawrence

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Stuart J. Russell, Chair

There are many challenges to learning optimal motor control. These challenges include noisy environments and sensors, nonlinear dynamics, continuous variables, high-dimensional problem domains, and redundancy. Reinforcement learning can be used, in principle, to find optimal controllers; however, the traditional learning algorithms are often too slow because obtaining training data is expensive. Although policy gradient methods have shown some promising results, they are limited by the rate at which they can estimate the gradient of the objective function with respect to a given policy's parameters. These algorithms typically estimate the gradient from a number of policy trials. In the noisy setting, however, many policy trials may be necessary to achieve a desired level of performance. This dissertation presents techniques that may be used to minimize the total number of trials required.

The main difficulty arises because each policy trial returns a noisy estimate of the performance measure. As a result, we have noisy gradient estimates. One source of noise

is caused by the use of randomized policies (often used for exploration purposes). We use response surface models to predict the effect that this noise has on the observed performance. This allows us to reduce the variance of the gradient estimates, and we derive expressions for the minimal-variance model for a variety of problem settings. Other sources of noise come from the environment and from the agent's actuators. Sensor data, which partially measures the effect of this noise, can be used to explain away the noise-induced perturbations in the expected performance. We show how to incorporate the sensor information into the gradient estimation task, further reducing the variance of the gradient estimates. In addition, we show that useful sensor encodings have the following properties: the sensor data is uncorrelated with the agent's choice of action and the sensor data is correlated with the perturbations in performance. Finally, we demonstrate the effectiveness of our approach by learning controllers for a simulated dart thrower and quadruped locomotion task.

Professor Stuart J. Russell
Dissertation Committee Chair

To my parents.

Contents

List of Figures

Acknowledgments

Many people have helped me during my time in graduate school. I would first like to thank my instructors. They taught me a variety of skills and most importantly, how to properly think about research problems. My dissertation committee members allowed me to discuss ideas, and they made several suggestions for improvements. My advisor, Stuart Russell, was instrumental in providing guidance and supporting me throughout this process.

I would also like to thank the administrators in the EECS department. Sheila Humphreys, in particular, gave me plenty of advice and encouragement. In addition, Beatriz Lopez-Flores and the rest of the Center for Underrepresented Engineering Students office (CUES) gave me a great deal of support.

My interactions with fellow graduate students have contributed to my research. Russell's Unusual Group of Students (RUGS) allowed me to share ideas with others and provided me with useful feedback. These members have participated in this process: Norm Aleks, Nimar Arora, Rodrigo de Salvo Braz, Kevin Canini, Shaunak Chatterjee, Bhaskara Marthi, Brian Milch, Mark Paskin, Erik Sudderth, Satish Kumar Thittamaranahalli, Jason Wolfe, and Andy Zimdars. I would especially like to acknowledge Eyal Amir, David Andre, Noah Cowan, Jeffrey Forbes, Daishi Harada, Vassilis Papavassiliou, and Hanna Pasula. Their friendship and support made my experience at Berkeley more enjoyable.

The Black Graduate Engineering and Science Students group (BGESS) provided me with a strong social support system. I would like to thank Kofi Boakye, Tiffany Crawford, Doug Densmore, Javit Drake, Mark McKelvin, Kimani Stancil, Nerayo Teclemariam, Jennifer Wade, and Hakim Weatherspoon.

My family has always been very supportive. Although my father, Dymus Lawrence, is no longer with us, he instilled in me the values that have allowed me to be successful. My mother, Dorothy Lawrence, has always been my biggest advocate, and she encouraged me throughout this process. My two older brothers, Michael and Brian Lawrence, gave additional support and supplied useful suggestions. Finally, I would like to thank my fiancée, Emille Davie. I greatly appreciate her love and support.

Chapter 1

Introduction

This chapter provides a description of the problems addressed in this dissertation. We present our approach to solving these problems, introduce the main contributions of the dissertation, and give an outline of the topics discussed in the remaining chapters.

1.1 Problem description and motivation

Suppose that we are given the task of controlling a robot, and that our goal is to get this robot to perform some desired motion. Possible motions include those that humans typically perform (e.g., walking, running, throwing, swimming, climbing stairs). For illustrative purposes, suppose that our specific goal is to supply the low-level control commands that will cause the robot to throw a dart towards a given target. The robot is made of several body parts that are connected to each other by different types of joints; motors are attached to each of these joints so that the robot may be actuated by a controller. Figure 1.1 shows how a computer controller typically interacts with the physical hardware

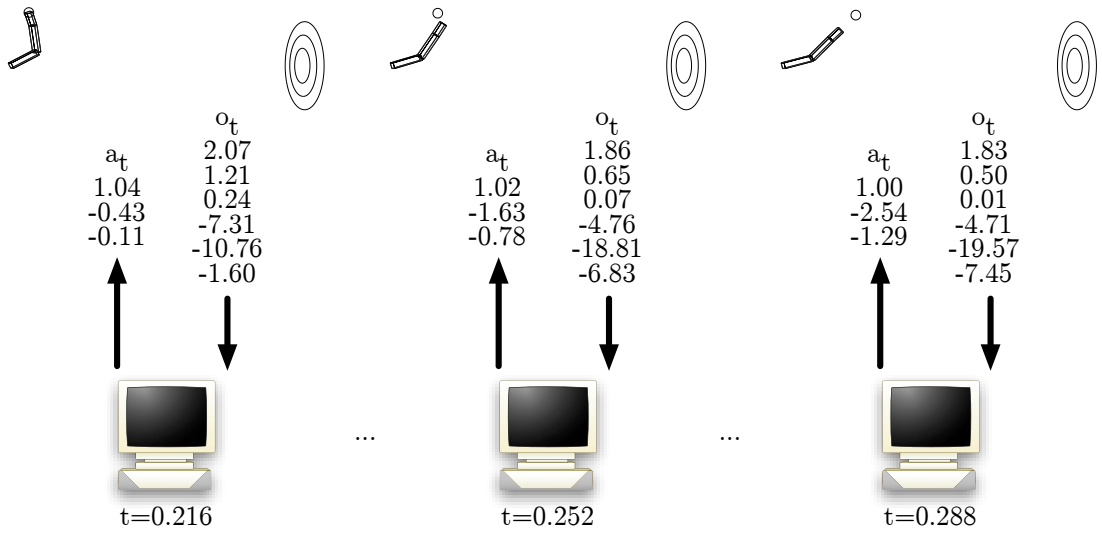


Figure 1.1: How a controller actuates a robot.

of the robot. At each time step, the controller supplies an action a_t that specifies the desired positions of each controllable joint. In addition, the controller receives noisy observations o_t from the environment; these observations typically include measurements of the joint angles and joint velocities. There are two general approaches to obtaining an effective sequence of low-level control commands. If we have an accurate model of the dynamics, we may be able to compute an optimal controller with respect to a suitably chosen objective function. Unfortunately, even in cases where we can obtain an accurate model of the dynamics, the task of finding an optimal controller is often difficult (Chapter 2 provides several reasons). *Motor control learning*, the alternative approach, is the process that an agent uses to adapt its controls, using sensory feedback from the environment, so that it effectively performs a desired motion. The ability to learn is important because it allows a system to adapt to

a constantly changing world. Fast learning is preferable because it reduces the amount of time spent acting inefficiently.

There are a number of companies and organizations working to expand the range of tasks that robots can perform. At the time of writing, over 2 million iRobot Roomba vacuuming robots have been sold and more than 150,000 Sony AIBO entertainment robots have been sold. Honda's humanoid robot Asimo is capable of performing some advanced tasks (e.g., greeting people and following them), Boston Dynamics' Big Dog can maneuver over rough terrain without falling, and the NASA Jet Propulsion Laboratory's rover explores the surface of Mars. Schaal [2007] suggests that in the near future, robots may be able to perform such complicated tasks as taking care of the sick and elderly, performing hazardous waste clean-up, and educating children. Bill Gates [2006] believes that the robotics industry may soon start to grow quickly, similar to the rapid growth of the personal computer industry in the 1980s. We believe that learning will play a significant role in the design and creation of these new robotic systems.

Since humans and other biological systems learn new motions throughout their lifetime, examining these systems may provide insight when designing effective controllers and efficient learning algorithms. For example, *central pattern generators*, which trigger behaviors according to a cyclical pattern, have been used to design controllers for locomotion [Billard and Ijspeert, 2000, Bruce *et al.*, 2002, Fukuoka *et al.*, 2003]. The six-legged RHex robot, inspired by the cockroach, can maneuver over rough terrain using simple open-loop control strategies [Buehler *et al.*, 2000, Saranli *et al.*, 2001]. One interesting question is, what criteria do biological systems use to measure the performance of each task? Consider

the following qualitative description of a control strategy for dart throwing: throw dart with maximal velocity straight toward the target. Although this strategy may work in a noise-free computer simulation, the resulting motion looks unrealistic. This is because real systems have other objectives (e.g., conserve energy and minimize probability of injury) and act in noisy environments. Nelson [1983] observed that optimizing a controller with respect to a minimal energy or minimal jerk criterion produces a smooth motion, a property shared by biological systems. More recent work suggests that these systems act in ways that are robust with respect to *multiplicative noise*, i.e., noise that is proportional to the desired force [Harris and Wolpert, 1998, Todorov and Jordan, 2002]. By determining the objectives that biological systems aim to achieve and designing computer simulations that accurately model these systems, we can produce realistic motion. In fact, using a multiplicative noise model, Todorov and Jordan [2002] were able to obtain controllers whose resulting motions qualitatively matched biological systems. Our prior work with a simulated dart thrower also shows, anecdotally, that learning robust controllers in noisy environments produces realistic looking motions [Lawrence *et al.*, 2003].

1.2 Short summary of previous work

This section briefly summarizes previous work (Chapter 2 provides more detail). Motor control learning algorithms typically consider a parameterized family of control strategies, and their goal is to find a strategy that maximizes a given objective. A *performance measure* uses a single numeric value to capture the quality of a given motion; it is important because it gives an agent a clearly defined goal to achieve, i.e., to maximize the

performance. In a noisy environment, an agent's performance will vary each time it executes a given control strategy and so learning algorithms focus on maximizing the expected performance. Genetic algorithms maintain a set of control strategies and evolve them over many generations to improve the performance. These algorithms have been used to learn new controllers for quadruped locomotion [Hornby *et al.*, 1999, Chernova and Veloso, 2004]. Another class of algorithms perform hill-climbing by repeatedly estimating *policy gradients*, i.e., the gradient of the expected performance with respect to a set of policy parameters. These algorithms differ in how they estimate the gradient. Stone and Kohl [2004] estimate the gradient from a set of trials distributed around the current parameter setting. They improve the learning rate by examining the correlations between each policy parameter and the performance; the correlations are then used to construct an estimate of the gradient.

REINFORCE uses stochastic policies for exploration purposes and can form an unbiased estimate of the gradient from a single policy trial [Williams, 1992]. Unfortunately the resulting gradient estimates often suffer from a high amount of variance. One way to reduce the variance is to examine multiple policy trials, for a single parameter setting, and take the average of the resulting gradient estimates. This is undesirable because obtaining policy trial data can be expensive. As an alternative, a number of techniques have been developed to reduce the variance of the standard REINFORCE gradient estimates. By incorporating a baseline term, the variance of the resulting gradient estimates can be reduced substantially [Weaver and Tao, 2001]. The GPOMDP algorithm uses a discounting factor to reduce the variance of the gradient of the average reward with respect to the policy parameters [Baxter and Bartlett, 2001]. Actor-critic methods incorporate a value function to

improve learning [Sutton *et al.*, 2000, Greensmith *et al.*, 2001, Konda *et al.*, 2003]. Using the natural gradient [Amari, 1998] may also give substantial improvements [Kakade, 2002, Peters and Schaal, 2006]. Normalized importance sampling can be used for sample reuse, i.e., exploiting policy trial data from previous hill-climbing steps [Shelton, 2001]. In simulation environments, PEGASUS increases the speed of learning by converting the stochastic problem into a deterministic one [Ng and Jordan, 2000]. By removing the stochasticity, they can then use standard optimization techniques to find a locally optimal control strategy.

1.3 Our approach

We extend the REINFORCE family of algorithms by continuing to lower the variance of the gradient estimates. This is important because policy trials, even in simulated environments, are often expensive to obtain. For illustrative purposes, let us first consider the setting in which gradient estimation is easy. In a d-dimensional deterministic setting we can compute the exact gradient using finite differences with $d+1$ policy trials. In the noisy setting, however, an agent may need to obtain many more policy trials. Although each policy trial produces an unbiased estimate of the expected performance, the variance adds noise to the gradient estimates. The amount of noise in the gradient estimates decreases as we move towards the deterministic setting. Thus if we could reduce the noise in each policy evaluation, we will reduce the noise in the gradient estimates.

One source of noise is introduced by the use of randomized policies; these policies perturb the nominal control commands issued by the controller. Other sources of noise, including both actuator and environmental noise, affect the actual torques (or forces) gen-

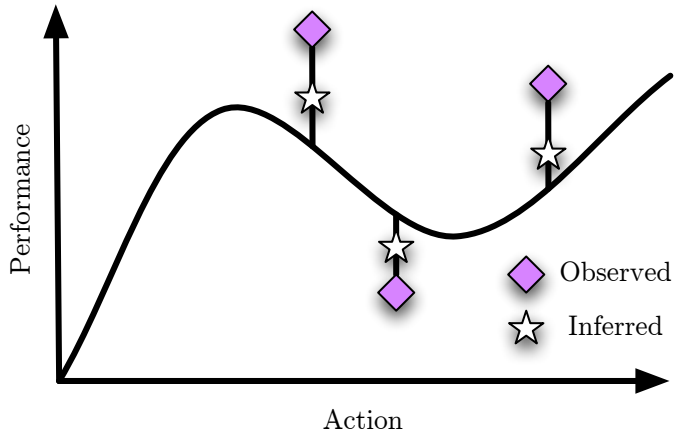


Figure 1.2: The expected performance of a one-dimensional toy problem (solid line). The diamonds denote the actual performances of three policy trials, and the stars denote *a posteriori* estimates of the corresponding expected performances.

erated by these commands. Thus if we were to execute the same policy multiple times, each trial will return a different value for the performance. Figure 1.2 illustrates the challenge via a one-dimensional problem. From three policy trials, the agent would ideally like to determine the expected performances. Because of the noisy environment, it can only observe rough estimates (denoted by the diamonds). If we could explain away the noise-induced perturbations of the performances, moving the rough estimates closer to the true mean (this move is illustrated by the stars), then we move ourselves closer to the deterministic setting. This dissertation presents algorithms that exploit this idea and as a result, produce gradient estimates that are less noisy.

The first class of algorithms that we explore in this dissertation considers the effect that exploration noise has on the performance; most gradient estimation algorithms do not fully consider this effect. For example, in the dart thrower task the agent may choose to explore, during a particular policy trial, by throwing the dart with more velocity than what

the current control parameters specify. Suppose, as a result, that the increased velocity causes the dart to land closer to the target. We capture relationships like this by learning a *response surface model* (RSM) which, in our case, predicts the value of the expected performance as a function of a set of user-defined features evaluated during each trial. By using the appropriate model, we can reduce the variance of the resulting gradient estimator.

The second class of algorithms exploits sensory information (e.g., measurements of each joint angle); this information can be used to explain away the noise-induced perturbations in the performance. While most controllers respond to changes in the sensor data, learning algorithms typically ignore this information when estimating the gradient. For example, suppose that the dart thrower executes a policy and senses that it released the dart too late. This causes the dart to land below the intended target and results in a poor performance. Since this performance can be explained by the sensor data, perhaps the agent should conclude that the policy parameters were not responsible for the miss (the culprit was probably the release-time noise). This kind of reasoning allows us to effectively reduce the noise in each performance estimate, thereby reducing the variance of the gradient estimates. This reasoning does, however, require knowledge of the relationships between the sensor data, policy parameters, and performances. A key step to performing these types of inferences is to use a sensor encoding that allows us to perform these kinds of inferences. We show that good sensor encodings have the following properties: the distribution of encoded sensor values is uncorrelated with the exploration noise and the sensor values correlate with the noise-induced perturbations of the performances. One useful sensor encoding uses the difference between the observed motion of the system and the expected motion at each

time step. The idea of removing the contribution of one's own motion from the sensory data is also present in the biology literature [Wolpert *et al.*, 2001]. Using these ideas, we can reduce the number of policy trials needed before an agent can construct an accurate estimate of the gradient and thus quickly achieve proficient behavior.

1.4 Thesis

Reasoning about the noise-induced perturbations in the performance allows an agent to more quickly learn optimal motor control policies. By constructing response surface models, we can explain away a portion of the noisy perturbations generated by the use of randomized policies. Furthermore, the sensor data obtained during learning partially explains the noise-induced perturbations, reducing the agent's uncertainty over the true value. The amount of experience required for learning can thus be reduced substantially by designing learning algorithms that explicitly reason about the exploration noise and sensor data obtained during each policy trial.

1.5 Main contributions

This dissertation presents efficient gradient estimation algorithms which are then used to quickly learn effective controllers for a simulated dart thrower and quadruped locomotion task. We use response surface models to predict the performance of a given policy trial as a function of the exploration noise. Given a set of linear basis functions, we derive an expression for the coefficients that minimize the variance of the corresponding gradient estimator. We also provide expressions for the variance in situations where the agent ex-

plores using Gaussian noise and the performance is locally linear in the exploration noise. In addition, we provide natural gradient versions of the RSM gradient estimators.

We also incorporate sensor data into the gradient estimation task to further reduce the variance. We derive expressions for the variance of a gradient estimator that uses sensor data and show that a good sensor is uncorrelated with the exploration noise, but correlated with the perturbations in the performance. Useful sensor encodings are presented for motor control problems and we present a technique that can be used when the ideal conditions do not hold. These techniques are used to improve the learning rates for a simulated dart thrower and a quadruped locomotion problem.

1.6 Outline of dissertation

Chapter 2 describes background work in motor control and reinforcement learning. We discuss the following challenges that are faced when learning effective controllers: noisy environments and sensors, nonlinear dynamics, continuous variables, high-dimensional problem domains, and redundancy. The chapter reviews a number of reinforcement learning techniques that can be applied to motor control problems. One of these techniques, policy search using gradient estimation via the REINFORCE algorithm, serves as the basic algorithm that will be extended in later chapters.

Chapter 3 describes three problem domains that will be used to compare the learning performances of the various gradient estimation algorithms discussed throughout the dissertation. We present a toy cannon problem that is useful because it allows us to visualize several key ideas in gradient estimation. (We can write an analytical expression

that closely approximates the true value function and its gradient.) This chapter also presents the simulated dart throwing and quadruped locomotion problems; these problems share many of the challenges described in Chapter 2.

Chapter 4 describes how we improve gradient estimation by learning a response surface model. Given a parameterized family of models, we derive equations for the parameters that produce the minimal-variance gradient estimator. We also derive expressions for the variance of the gradient estimator when an agent uses Gaussian exploration noise. We show that the maximum likelihood estimation of a linear-Gaussian model is mathematically equivalent to the natural gradient estimator. Using response surface models allows us to quickly learn effective controllers for the dart thrower and quadruped problems.

Chapter 5 describes how improvements can be made by incorporating the sensory data obtained during each policy trial. While agents may use sensors to determine what actions to perform, policy search algorithms typically ignore this information when estimating the gradient. We reduce the variance by incorporating the sensor data, and we derive expressions for the variance of the gradient estimators. We also show how to take raw sensor data from motor control problems and produce useful encodings. Incorporating the sensor data improves the learning performance of both the dart thrower and quadruped tasks.

Chapter 6 presents conclusions and discusses some ideas for future work.

Chapter 2

Background: motor control and reinforcement learning

This chapter presents some background work in both motor control and reinforcement learning. We explain that although an agent may face several difficulties when learning effective policies, there will always be real-world problems where learning is necessary. This chapter also reviews a number of reinforcement learning techniques that are often applied to motor control problems. In Chapter 4 and Chapter 5 we extend some of these algorithms to increase the learning performance.

2.1 Motor control learning

2.1.1 Problem description

Motor control learning is the process that an agent uses to adapt its controls, based on sensory feedback from the environment, to better perform a desired motion. Figure 2.1

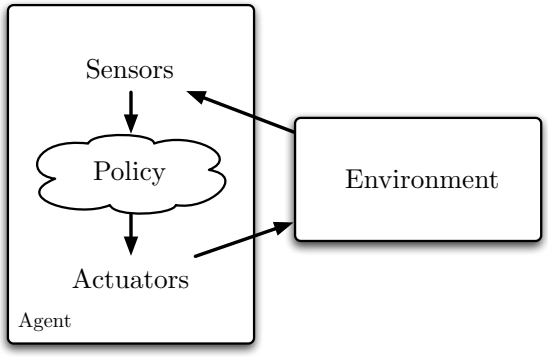


Figure 2.1: How an agent interacts with its environment.

shows the basic setup of how an agent interacts with its environment. Examples of motions that an agent may perform include those that humans typically perform: reaching for a cup of coffee, walking up a flight of stairs, throwing a dart towards the bull's-eye, and taking a morning swim. An agent performs these motions by controlling its actuators. In biological systems muscles create motion by contracting in accordance to the signals received from the central nervous system. In robotic systems, electrical motors produce motion by producing forces and torques based on the commands received from a controller. Sensory feedback may be used during movements to compensate for the noise experienced throughout a particular motion. This task usually requires some form of state estimation, i.e., the process of using sensors to update and maintain a prediction of an agent's true state. Sensory data may also be used to measure an agent's current performance so that an agent can adapt its controls to increase its future performance. An agent's actuators and sensors make it possible to perform and learn new motor control skills, a key component to an agent's survival in a complex world filled with novel situations.

Performance measures

A performance measure uses a single numeric value to capture the quality of a given motion; it is important because it gives an agent a clearly defined goal to achieve, i.e., to maximize the performance. Performance measures are often defined with respect to a particular task and they often need to incorporate several different objectives. For example, the performance is often measured in part by how well an agent achieves its direct goal. Did the agent successfully pick up the cup of coffee without spilling? Did the agent hit the bull's-eye in a game of darts? An example of a performance measure can be created by penalizing squared errors or by simply assigning a high value to a successful motion (e.g., coffee cup in hand) and a low value to failures (e.g., falling down the stairs). Agents also have to incorporate other needs such as preserving energy and minimizing the physical wear and tear on the body.

Several hypotheses have been made to describe the performance measures used by biological systems. Nelson [1983] compared the motions obtained by optimizing the following objective functions: time cost with maximal force constraint, force cost, impulse cost, energy cost, and jerk cost. All objective functions, except for time cost, use a fixed time for when the agent should reach the goal state. The task was to control a one-dimensional linear system (the state space includes position and velocity) so that it reaches the goal state with zero velocity. The energy cost is written as $\int_{t=0}^{t_f} u(t)^2 dt$ and the jerk cost is written as $\int_{t=0}^{t_f} \dot{a}(t)^2 dt$, where $u(t)$ is the force or torque at time t, $a(t)$ is the acceleration at time t, and $|\dot{a}(t)|$ is the jerk. Minimizing the time cost and force cost performance measures results in a bang-bang controller, i.e., apply a constant force in one direction followed by

the same constant force in the opposite direction. This results in an abrupt change in acceleration causing a jerk in the motion; in fact it has an infinite jerk according to the above definition. Using the energy cost and jerk cost metrics resulted in smooth motions. Because the jerk performance measure does not directly incorporate the mass of the system, the jerk produced by moving a heavy limb is penalized the same as an equivalent motion of a lighter body part. Uno *et al.* [1989] suggest using the minimum-torque-change model which aims to minimize $\int_{t=0}^{t_f} \dot{\tau}(t)^2 dt$ where $\tau(t)$ is the torque at time t.

2.1.2 Challenges in finding optimal controllers

There are many challenges in finding effective controllers!

Noisy environments and sensors

As an agent interacts with its environment, noise from several sources affects its behavior. Actuators are imperfect and so the actual force, torque, or level of muscle contraction produced by an agent may differ from the commanded value. An agent's sensors also give noisy measurements of the values that are to be measured. Unfortunately, the sensors may not be capable of directly measuring important quantities (e.g., joint sensors in a quadruped robot only provide relative information and thus cannot measure the absolute position and absolute rotation of the torso). The effect that the environment has on an agent is often modeled as a noisy process because the set of complex interactions is hard to quantify. An example of environmental noise is the effect that bumpy ground has on locomotion. Noise makes learning difficult because it creates uncertainty over the outcome of performing an action. Thus it typically increases the amount of experience an agent must

acquire in an environment before an effective controller can be learned.

Recent work demonstrates that a key component in biological systems is that they learn to minimize the effect that noise has on the performance. Optimizing a controller with respect to a multiplicative noise model (i.e., the variance of the noise is proportional to the desired force) has been shown to match biological data. Harris and Wolpert [1998] used a model of eye saccade motion to find the optimal force profile by minimizing the expected squared error in the final eye position. They found that the resulting motion matched real biological data. Using a linear system model with feedback control, Todorov and Jordan [2002] were able to qualitatively match a number of motions performed by human subjects. These motions include moving a pointer through various targets and folding a piece of paper into a ball. They also introduce the minimal intervention principle which states that an agent should adapt its controls to reduce the effects of noise along directions that affect task performance. As a result, they argue that noise will tend to accumulate in the directions that are irrelevant to the performance.

Nonlinearities

Most motor control problems have nonlinear dynamics. The equations of motion, as given by the laws of physics, are written as a set of nonlinear differential equations. Physical contacts may introduce a large nonlinear component into the system. A common example is the force that the ground exerts on the feet during locomotion. Nonlinear dynamics rule out many standard techniques that only apply to linear dynamical systems. An example of a solution that works for linear problems is the linear quadratic regulator where an agent's goal is to minimize a quadratic cost function when the dynamics are

governed by a linear system. One possible approach for nonlinear systems is to linearize the system around a nominal state or path. For example, the dynamics of the classical cart-pole problem are often approximated as a linear system. While this may work well for certain problems, it cannot be applied to domains where the system behavior deviates significantly from the nominal state or trajectory. It may also be difficult to determine what the nominal trajectory should be for more complicated tasks.

Continuous states and actions

Motor control problems have continuous state variables and they often have a continuous action space. Many learning algorithms only work with problems that have a discrete state space. Some approaches convert continuous problems by discretizing the state and action spaces. A major drawback is that the size of the resulting state and action spaces become prohibitively large for high dimensional problems. Other approaches use function approximators to encode a value function. For example, the Cerebellar Model Articulation Controllers (CMACs) representation discretizes features of a state space at multiple coarseness levels and has been used successfully for a number of problems [Albus, 1975]. Neural networks, radial basis functions, and several other function approximators have been used to encode an agent's value function [Tesauro, 1992, Sutton, 1996, Atkeson *et al.*, 1997]. When these methods are employed, convergence to an optimal policy may not be guaranteed.

High-dimensionality and redundancy

High-dimensional state and action spaces are common in motor control problems. Human bodies are estimated to have over 100 degrees of freedom. The Sarcos humanoid

robot DB has 30 degrees of freedom and the Sony AIBO robot has 26 degrees of freedom. Motor control learning problems will thus often be susceptible to the curse of dimensionality which states that the complexity grows exponentially with the number of degrees of freedom. The high number of degrees of freedom often makes these systems highly redundant, i.e., there are many ways to perform a particular task. Bernstein [1967] observed that biological systems must be able to master these redundant degrees of freedom to produce good motion. For example in a noise-free dart throwing task, there are an infinite number of ways to move the arm so that when the dart is released it exactly hits the bull's-eye. If the goal is to learn motions that match biological systems, this redundancy must be properly reduced by making the model more realistic. Properties of realistic simulations include accurate noise models (i.e., multiplicative noise) and performance measurements that capture the objectives of real systems (e.g., energy conservation and jerk minimization).

2.1.3 Advantages of taking an online learning approach

Online learning occurs when an agent continuously learns in the same environment that it acts, by using the data received during each policy trial execution. It has the potential to avoid many of the problems associated with an offline approach, where an agent learns a complete control strategy before it acts. If we are given an accurate dynamical system model for certain classes of control problems (e.g., the linear quadratic regulator problem), it may sometimes be possible to find effective controllers by directly solving for the optimal controller. In other situations, offline learning could take place inside a simulator to find a good controller. In either case, if these controllers were then implemented on a real system, inefficiencies would almost surely arise because the real system and the simulator are not

equivalent. Simulators are often inaccurate for several reasons. For example, the masses and lengths of an agent's body parts are often estimated for simulation purposes. In addition, it may be hard to determine the exact relationship between motor commands and the corresponding torques produced by the motors. Finally, there are unmodelled aspects: slack, hysteresis, vibrations, bending, stiction, etc. There are similar difficulties in simulating an accurate sensor model. Thus while offline learning may be useful as a bootstrap, further gains will be possible when learning takes place online. Another advantage of online learning is that environments change and agents may find themselves needing to perform novel tasks.

2.2 Reinforcement learning

Before we discuss reinforcement learning algorithms, we must first define the decision processes that they solve.

2.2.1 Markov decision processes

A *Markov decision process* (MDP) can be used to represent a wide range of sequential decision problems [Bellman, 1957]. In this framework an agent is always assumed to be in one of several states $s \in \mathcal{S}$ and it has perfect sensing (i.e., the agent always knows what state it is in). At the beginning of the process the agent is placed into an initial state according to the initial state distribution D. At each time step the agent decides which action $a \in \mathcal{A}$ to perform. Each action causes a stochastic state transition whose distribution depends on the current state and choice of action. These next-state probabilities are given by the *transition function* T. In problems that have a discrete state space, this transition

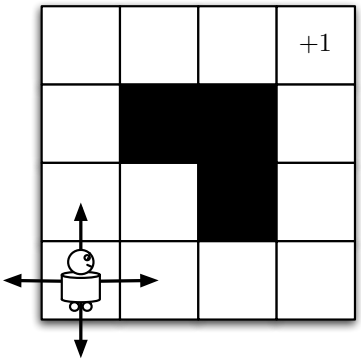


Figure 2.2: Example of a Markov decision process (MDP).

function may be written as a conditional probability distribution $T(s, a, s')$, where the agent transitions from state s to state s' after performing action a. For continuous state spaces, the transition function may be written as conditional probability density function. A policy π specifies what action the agent should take in each state and is represented by a mapping between states and actions. We write $\pi(s)$ to denote the action that π takes in state s. The agent receives a *reward* $R(s, a)$ as a function of the current state and action. This reward signal measures the single-step desirability of performing an action from some state. Problems also include a *discounting factor* γ which discounts the importance of reward signals obtained in the future. After defining the components, we can write an MDP as a tuple $(\mathcal{S}, D, \mathcal{A}, R, T, \gamma)$.

Figure 2.2 shows an example of a discrete state MDP where $\mathcal{S}$ contains the grid coordinates of a robot. In each state the agent may choose one of four actions $\mathcal{A} = \{\text{north}, \text{south}, \text{east}, \text{west}\}$. The agent receives a $+1$ reward when it performs any action in the north-east corner and a reward of zero otherwise. The stochastic transitions

take place so that with probability $(1-p_a)$, the robot successfully executes its desired action provided that there is no wall blocking its path. With probability p_a the agent accidentally executes one of the other actions, where each undesired action occurs with probability $p_a/3$. To encourage progress towards the goal we let $\gamma = 0.9$.

The goal is to find an optimal policy π^* which maximizes the agent's expected sum of discounted rewards. This expectation is captured by the value function $V_\pi(s)$, where we use subscripts to denote the policy. To understand the value function, imagine that an agent executes a policy starting from some initial state s_0. At each time step it will perform an action according to $\pi(s)$ which will cause it to stochastically transition to a new state. The agent receives rewards as a function of the states that it enters and the reward received at time step t will be discounted by a factor of γ^t. Randomness in the state transitions induces a distribution over the discounted sum of rewards obtained by executing π from s_0. The value function gives the average of this distribution and it can be expressed as follows:

$$V_\pi(s) = \mathrm{E}_\pi\Big[\sum_{i=0}^{\infty} \gamma^i \mathbf{r}_{t+i} \Big| s_t = s\Big], \tag{2.1}$$

where $\mathbf{r}_{t+i}$ is a random variable representing the reward received at time step $t+i$.

2.2.2 Partially observable Markov decision processes

Partially observable Markov decision processes (POMDPs) are capable of representing an even wider range of learning tasks [Smallwood and Sondik, 1973]. These decision problems are complicated due to the fact that the states are only partially observable. Instead of an agent knowing what state it is in, it receives an observation $o \in \Omega$ whose value is distributed conditioned on the current state. The distribution of these observa-

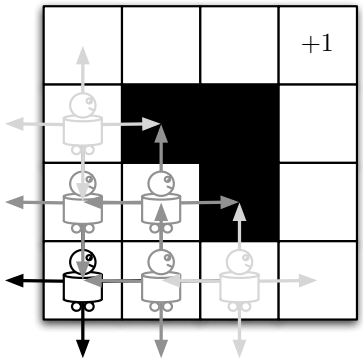


Figure 2.3: Example of a partially observable Markov decision process (POMDP).

tions is give by the observation model O. In problems that have a discrete state space, this distribution may be written as a conditional probability distribution $O(s, o)$, where the agent observes o from state s. For continuous state spaces, the observation model may be written as a conditional probability density function. In either case, the agent may be uncertain over its true state. It may elect to maintain a *belief state* b_t which gives the posterior probability of being in each state as a function of the entire history of observations. Maintaining the current belief state is an example of filtering. We can write a POMDP as follows: $(\mathcal{S}, D, \mathcal{A}, R, T, \Omega, O, \gamma)$.

Figure 2.3 shows an example of a discreet state POMDP where $(\mathcal{S}, \mathcal{A}, R, T, \gamma)$ are equivalent to the MDP shown in Figure 2.2. However in this environment the agent's sensors properly detect the grid coordinate with probability p_s and otherwise the sensor fails to return a reading. We may define the set of observations by augmenting the state space with a symbol denoting failure ($O = \mathcal{S} \cup \{\text{fail}\}$). This observation model causes the agent to be uncertain about which state it happens to be in.

The *Markov property* states that the distribution of future states is independent of the past given that we know the current state; it is essential to algorithms that solve MDPs. Although this property does not apply to the observations received in partially observable environments, it does hold for the belief states. Thus we could, in principle, apply any algorithm capable of solving an MDP with a continuous state space to a POMDP via a conversion that treats the belief states as regular states in an MDP. In discrete state spaces of significant size, reasoning about the belief states may be computationally difficult because the belief states are continuous. A key observation is that while the value function is defined over a continuous space, it can still be represented compactly for small problems. However, algorithms for directly solving POMDPs for continuous state space problems are hopeless with a few exceptions. For example, the regulation task has an analytical solution for linear control problems with a quadratic cost function and Gaussian noise.

2.2.3 Problem description

Reinforcement learning is the process of learning an effective policy from an agent's experience; Kaelbling *et al.* [1996] provide a good survey. Typically an agent does not know *a priori* the transition or reward functions for a particular problem. The goal is for an agent to efficiently improve a given policy π based on experience. Each state transition can be thought of as a random sample from the transition and reward functions. This information is used to optimize the current policy.

We must distinguish between model-free and model-based methods. A *model-free* method attempts to learn the optimal policy without learning the transition or reward functions. The Q-learning algorithm is one such method [Watkins and Dayan, 1992]. It

maintains a Q-function, written as $Q_\pi(s, a)$, which gives the expected sum of discounted rewards given that an agent executes a from state s and then follows π. Q-learning uses temporal differencing to update $Q_\pi(s, a)$ and under the right conditions it will learn the optimal policy. *Model-based* methods either have access to the transition function or learn an approximation from experience. Since each state transition is a random sample from the true transition function, the agent can estimate T and R from its experience. Given this model, the agent can periodically solve the approximate MDP to extract the corresponding optimal policy.

Value-based methods are very popular for solving reinforcement learning problems. These algorithms continuously refine an estimate of a value function (or Q-function) based on an agent's experience. It will be valuable to explain why these methods are generally hard to apply to partially observable domains. If an agent does not know the transition function or even the dimensionality of the hidden state space, then it is unclear how to properly maintain a belief state. Without a belief state, it will be very difficult to learn the value function using the standard techniques. Policy search methods can avoid this problem by working with a carefully chosen set of parameterized policies; these methods have shown promising results in partially observable domains.

2.2.4 Policy search methods

Policy search methods search through a space of parameterized policies to find an effective policy. In this framework, a policy can be represented by a d-dimensional vector $\pi \in \mathbb{R}^d$, whose entries determine the behavior of the system. We can define a value function over policies where we write $V(\pi)$ to denote the expected sum of future rewards obtained

by executing π from the initial state distribution D. An agent's experience can be used to estimate $\nabla_\pi V(\pi)\Big|_{\pi=\pi_0}$, the gradient of the value with respect to π, evaluated at the current policy π_0. Improvements to the policy are then made by adjusting the parameters in the uphill direction. Although these algorithms can get stuck in a local maximum, they have been successfully applied to many motor control problems.

We must distinguish between episodic and non-episodic algorithms. *Episodic* algorithms estimate the gradient from a number of policy trials where each policy trial consists of executing a policy from some initial state. After the agent reaches a terminal state, the process is repeated by executing a policy from a new initial state. In a non-episodic algorithm, the agent acts continuously without its state being reset to some initial value. In both settings, improvements to the policy are periodically made by adjusting the parameters in the direction of the gradient. For episodic algorithms it is useful to define the *score* f as the discounted sum of rewards obtained during a single policy trial. Policy search algorithms differ in how they estimate the gradient.

Gradient estimation using linear regression

One of the simplest gradient algorithms uses linear regression to estimate the gradient at the current policy π_0. Suppose we have n policy trials where the parameters of each trial are chosen so that they collectively form a spanning set of the entire policy space. Let $\pi^{(i)}$ contain the parameters for the ith trial and let $f^{(i)}$ be the ith score. We can locally approximate $V(\pi)$ by fitting a linear function: $f^{(i)} = {\pi^{(i)}}^T a_\pi + b$. This linear approximation is only valid when we choose policy trials that are close to π_0. Linear regression can be performed to learn a_π which may then be used as an estimate of the gradient.

REINFORCE

One of the earliest gradient estimation algorithms used for reinforcement learning was REINFORCE [Williams, 1992]. This episodic algorithm can produce an unbiased estimate of the gradient from a single policy trial. This algorithm uses randomized policies for exploration purposes. Thus at each time step, an agent performs a random action according to a distribution conditioned on the policy parameters. Each policy trial returns a history h, a sequence of observation-action-reward values, and from this history we can measure the score $f(h)$ by taking the sum of the reward values. The value function may be defined over policies $V(\pi) = \mathrm{E}[f(\mathbf{h})|\pi]$ where the histories $\mathbf{h}$ are generated from π.

Assuming that h is written in vector form, the gradient of the value with respect to π can be written as an integral expression:

$$\nabla_\pi V(\pi) = \nabla_\pi \int f(h) P(h|\pi) dh. \tag{2.2}$$

We can multiply the contents of the integral by the expression $\frac{P(h|\pi_0)}{P(h|\pi_0)}$. By moving the gradient operator inside the integral and approximating the integral by a sum (using the histories from n policy trials) we may derive the gradient estimator as follows:

$$\begin{aligned}
\nabla_\pi V(\pi) &= \nabla_\pi \int \frac{P(h|\pi)}{P(h|\pi_0)} f(h) P(h|\pi_0) dh \\
&= \int \frac{\nabla_\pi P(h|\pi)}{P(h|\pi_0)} f(h) P(h|\pi_0) dh \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \frac{\nabla_\pi P(h^{(i)}|\pi)}{P(h^{(i)}|\pi_0)} f(h^{(i)}),
\end{aligned}$$

where $h^{(i)}$ is the ith history. In policy search algorithms we are typically interested in

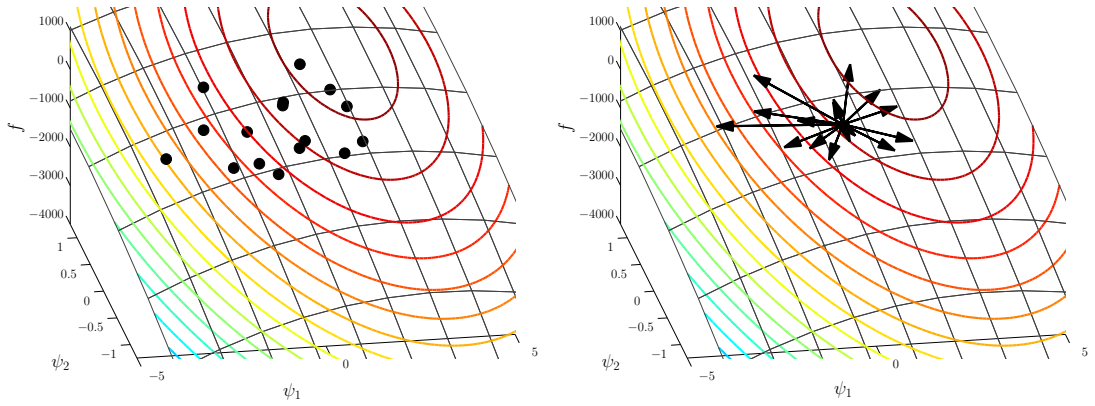


Figure 2.4: (a) Scores from several policy trials drawn from a toy problem. (b) Corresponding eligibility vectors.

evaluating the gradient at π_0.

$$\begin{aligned} \nabla_\pi V(\pi)\Big|_{\pi=\pi_0} &\approx \frac{1}{N}\sum_{i=1}^{N} \frac{\nabla_\pi P(h^{(i)}|\pi)\big|_{\pi=\pi_0}}{P(h^{(i)}|\pi_0)} f(h^{(i)}) \\ &= \frac{1}{N}\sum_{i=1}^{N} \psi(h^{(i)}|\pi_0) f(h^{(i)}), \end{aligned} \tag{2.3}$$

where the *eligibility vectors* $\psi(h^{(i)}|\pi_0)$ are defined as follows:

$$\psi(h^{(i)}|\pi_0) = \frac{\nabla_\pi P(h^{(i)}|\pi)\big|_{\pi=\pi_0}}{P(h^{(i)}|\pi_0)} = \nabla_\pi \log P(h^{(i)}|\pi)\Big|_{\pi=\pi_0}. \tag{2.4}$$

Intuitively, this vector points in the direction that an agent should move in policy space to make executing a given history more likely. This is demonstrated in Figure 2.4(a) which plots the scores of sixteen samples drawn from a two-dimensional toy problem as a function of ψ (Figure 2.4(b) plots the eligibility vectors). An estimate of the gradient is produced by weighting each eligibility vector by its corresponding score. This weighting increases the likelihood that an agent repeats the histories of the high scoring policy trials when it follows the gradient.

In Equation 2.4 we saw that ψ can be written as the gradient of a log-likelihood expression. Because of the log term, the eligibility for a policy trial can be decomposed as the sum of single-step eligibility vectors. In the fully observable case, the probability of a given history can be written as the following product:

$$P(h|\pi) = P(s_0)\prod_{t=0}^{t_f} P(a_t|s_t,\pi)P(s_{t+1}|s_t,a_t)P(r_t|s_t,a_t). \tag{2.5}$$

The only terms in the above expression that depend on the policy are the $P(a_t|s_t,\pi)$ terms. Thus we can write the gradient of the log-likelihood as follows:

$$\nabla_\pi \log P(h|\pi)\Big|_{\pi=\pi_0} = \sum_{t=0}^{t_f} \nabla_\pi \log P(a_t|s_t,\pi)\Big|_{\pi=\pi_0} = \sum_{t=0}^{t_f} \psi_{(t)}(a_t|s_t,\pi_0), \tag{2.6}$$

where the single-step eligibility is written as follows: $\psi_{(t)}(a_t|s_t,\pi_0) := \nabla_\pi \log P(a_t|s_t,\pi)\Big|_{\pi=\pi_0}$.

In the case of reactive policies, where the action depends only on the current observation, partially observable problems decompose in a similar way. The probability of a given history is written as the following product:

$$\begin{aligned} P(h|\pi) = P(o_0)&\prod_{t=0}^{t_f} P(a_t|o_t,\pi)P(o_{t+1}|o_0,\ldots,o_t,a_0,\ldots,a_t,r_0,\ldots,r_t) \\ &\prod_{t=0}^{t_f} P(r_t|o_0,\ldots,o_t,a_0,\ldots,a_t,r_0,\ldots,r_{t-1}). \end{aligned} \tag{2.7}$$

Because the Markov property no longer holds, both the observations and reward signals depend on the previous observations, actions, and reward signals. However, the only terms that depends on the policy are the $P(a_t|o_t,\pi)$ terms. Thus we can write the gradient of the log-likelihood as follows:

$$\nabla_\pi \log P(h|\pi)\Big|_{\pi=\pi_0} = \sum_{t=0}^{t_f} \nabla_\pi \log P(a_t|o_t,\pi)\Big|_{\pi=\pi_0} = \sum_{t=0}^{t_f} \psi_{(t)}(a_t|o_t,\pi_0), \tag{2.8}$$

where the single-step eligibility is written as follows: $\psi_{(t)}(a_t|o_t,\pi_0) := \nabla_\pi \log P(a_t|o_t,\pi)\Big|_{\pi=\pi_0}$. This shows that even though an agent may not have access to the entire history for a given policy trial, it can usually access the components that are necessary for gradient estimation.

REINFORCE with a baseline

It was noted in the original description of REINFORCE that improvements could be made by subtracting a baseline term from the score. This does not introduce bias into the algorithm because the gradient of a function does not change if we add any constant to its value. The variance, however, can change substantially. A common heuristic used for selecting a baseline term was to use the expected sum of rewards. The baseline that minimizes the variance is actually formed using a weighted average [Weaver and Tao, 2001].

$$b^* = \frac{\mathrm{E}[\psi(\mathbf{h}|\pi_0)^T\psi(\mathbf{h}|\pi_0)f(\mathbf{h})|\pi_0]}{\mathrm{E}[\psi(\mathbf{h}|\pi_0)^T\psi(\mathbf{h}|\pi_0)|\pi_0]}. \tag{2.9}$$

Thus an improved REINFORCE estimator is written as follows:

$$\nabla_\pi V(\pi)\Big|_{\pi=\pi_0} \approx \frac{1}{N}\sum_{i=1}^{N}\psi(h_i|\pi_0)(f(h_i)-b^*). \tag{2.10}$$

In the episodic setting, we can estimate the optimal baseline from the n policy trials and then plug this back into the estimator. This process does introduce bias into the procedure. Imagine that we estimate the gradient from a single policy trial. In this case the optimal baseline will be set to the score received from the single trial causing the estimator to return 0 for the gradient, which is clearly biased (unless the true gradient happens to be 0 at π_0). However for a sufficient number of samples, the bias is minimal and using a baseline term improves the quality of the gradient estimates.

GPOMDP

REINFORCE has been known to suffer from high variance which means that it may require many policy trials to be confident in the estimate of the gradient. This tends to be true in problems where the agent acts for a large number of steps in each policy trial. The GPOMDP algorithm approximates the gradient of the average reward with respect to the policy parameters [Baxter and Bartlett, 2001]. To reduce the variance of the estimator it uses a discounting factor.

From a single policy trial, we can estimate the gradient as

$$\nabla_\pi V(\pi)\Big|_{\pi=\pi_0} \approx \sum_{t=0}^{t_f} \psi_{(t)}(a_t|o_t,\pi_0) \sum_{u=t}^{t_f} r_u. \tag{2.11}$$

This equation is formed by using the sum of reward signals as the score. We do not include terms that multiply an eligibility vector to a reward signal obtained at a previous time step. This is because reward signals do not depend on future actions. GPOMDP introduces a discounting factor to reduce the variance of the estimator while introducing some bias. The gradient estimator can be written as follows:

$$\nabla_\pi V(\pi)\Big|_{\pi=\pi_0} \approx \sum_{t=0}^{t_f} \psi_{(t)}(a_t|o_t,\pi_0) \sum_{u=t}^{t_f} \gamma^{u-t} r_u. \tag{2.12}$$

The bias does not appear to affect the performance in practice assuming that one chooses a suitable discounting term. The amount of bias depends on the discounting term and the mixing time of the induced Markov chain. One advantage of this approach is that it can be used in both an episodic and non-episodic setting. In the non-episodic setting, an agent maintains an estimate of the gradient and the accuracy increases the longer it interacts with the environment. Periodically, it may choose to adjust its controls in the

direction of the gradient.

Actor-critic techniques

Actor-critic algorithms have been shown to reduce the variance of gradient estimation [Sutton *et al.*, 2000, Greensmith *et al.*, 2001, Konda *et al.*, 2003]. These algorithms have a critic learn $Q(s,a)$, which predicts the expected sum of discounted rewards obtained after performing an action from a particular state; some variations have their critic learn $V(s)$ or an advantage function. The Q-function is then used by the actor in estimating the gradient. The actual sum of discounted rewards obtained during each policy trial is replaced by $Q(s,a)$. This gives the following gradient estimator:

$$\nabla_\pi V(\pi)\Big|_{\pi=\pi_0} \approx \sum_{t=0}^{t_f} \psi_{(t)}(a_t|s_t,\pi_0)Q(s_t,a_t). \tag{2.13}$$

When we previously used the actual sum of rewards, there was variance in this quantity because of the stochastic nature of the problem. By replacing this sum with $Q(s,a)$ we remove one source of noise which helps to reduce the variance of the gradient estimator.

If we use a linear function approximator for $Q(s,a)$, where the bases include the components of the eligibility vector, then the resulting estimator is unbiased. The simplest choice of features is to use the components of the eligibility vector. This gives the following function approximator $Q(s,a) = \psi_{(t)}(a_t|s_t,\pi_0)^T w$, where w is a parameter to be learned by the critic. It is not difficult to show that $\mathrm{E}[\psi_{(t)}(\mathbf{a}_t|s_t,\pi_0)^T w|\pi_0] = 0$ which means that the Q-function must predict that the expected sum of rewards from any state is zero. This provides a poor representation of the true Q-function. An improvement is made by adding a baseline term $\phi(s_t)$ that is parameterized by the current state. With the addition of this

term we can construct a Q-function by using $\psi_{(t)}(a_t|s_t,\pi_0)^T w$ to approximate the advantage function, which is defined as $Q(s,a) - V(s)$, and by adding a parameterized baseline $\phi(s_t)$ that approximates the value function. This gives the following gradient estimator

$$\nabla_\pi V(\pi)\Big|_{\pi=\pi_0} \approx \sum_{t=0}^{t_f} \psi_{(t)}(a_t|s_t,\pi_0)(\psi_{(t)}(a_t|s_t,\pi_0)^T w + \phi(s_t)), \tag{2.14}$$

where w and the parameters of $\phi(s_t)$ can be learned using temporal difference methods.

Natural gradient algorithms

Gradient estimators can suffer from high variance in cases where some of the parameters are nearly deterministic. Suppose we have a single-step problem where a randomized policy chooses an action from a Gaussian distribution centered around the policy parameters $\pi \in \mathbb{R}^2$. In this case, the eligibility of a policy trial is written as $\psi(a|\pi_0) = \Sigma^{-1}(a - \pi_0)$. Suppose that the covariance Σ of the action selection is written as $\Sigma = \begin{bmatrix} {\sigma_1}^2 & 0 \\ 0 & {\sigma_2}^2 \end{bmatrix}$ and that the noise in a_1 is much less than that of a_2 (i.e., $\sigma_1 \ll \sigma_2$). The variance of the first component of the gradient estimator will tend to be very high. Intuitively, the variation in the score is mostly due to variations in the second component since that is where most of the noise enters the system. However, the gradient estimator has no way to infer this. This means that small fluctuations in control that are mostly deterministic will dominate the gradient estimation. This is counterintuitive because in most problems, these small variations give very little information about the gradient.

The natural gradient estimators compensate for this by pre-multiplying the gradient estimator by the inverse of the Fisher Information matrix: $\mathcal{F} = \mathrm{E}\big[\psi(\mathbf{h}|\pi_0)\psi(\mathbf{h}|\pi_0)^T|\pi_0\big]$ [Amari, 1998, Kakade, 2002, Peters and Schaal, 2006]. The following estimator is biased

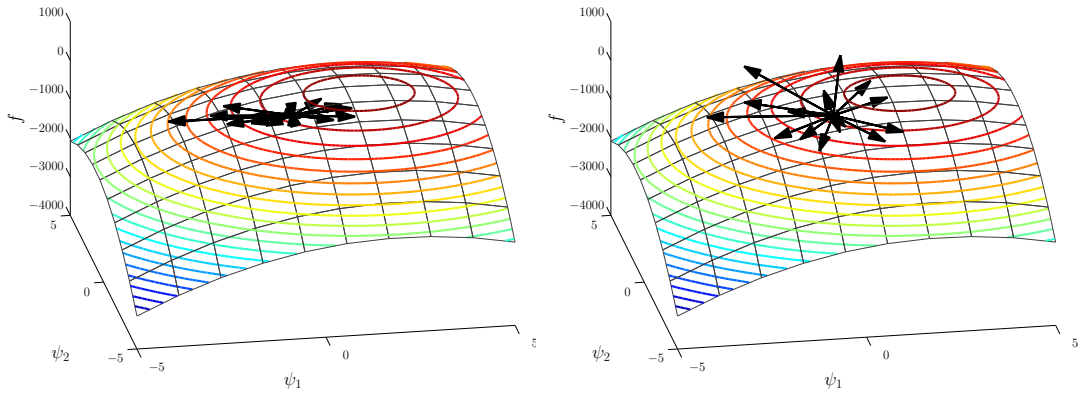


Figure 2.5: (a) Eligibility vectors of several policy trials. (b) Eligibilities scaled by the inverse of the Fisher Information matrix $\mathcal{F}^{-1}$.

but the bias is guaranteed to result in an estimate that is less than $\pi/2$ radians in expectation away from the true gradient:

$$\nabla_\pi V(\pi)\Big|_{\pi=\pi_0} \approx \mathcal{F}^{-1}\frac{1}{N}\sum_{i=1}^{N}\psi(h^{(i)}|\pi_0)f(h^{(i)}). \tag{2.15}$$

This expression improves the standard REINFORCE estimator but similar extensions can be made to the GPOMDP, baseline, and actor-critic algorithms. Figure 2.5 shows this effect for a toy two-dimensional problem. Although the agent has a larger exploration variance in a_2, the eligibilities vary the most along the first dimension. By scaling ψ by the inverse of the Fisher Information matrix, we compensate for this effect.

PEGASUS

The PEGASUS algorithm provides a substantial reduction in the variance of the gradient estimator [Ng and Jordan, 2000]. To gain some intuition, suppose that we evaluate two nearby policies and our task is to estimate the difference in value. The stochastic

nature of the problem makes this task difficult. Imagine a situation in which noise causes a fortuitous score in one trial, and an unlucky score for the other trial. In this situation the estimate of their difference will be much larger than the expected value. However if the noise is correlated in the two trials, then the task of estimating the difference becomes more accurate. This is because in cases where the noise causes a fortuitous score in one trial, correlated noise will tend to cause a fortuitous score in the other. Likewise when noise causes the agent to receive a lower than expected score in one trial, correlated noise will tend to cause the other trial to underperform.

PEGASUS exploits this idea by reusing the same noise when evaluating policies. In situations where one has control over the randomness (e.g., simulator environments), one can reuse the same noise variates for difference policy trials. This can be done by reseting the random number generator seeds, although some care must be taken to ensure a proper reuse of the noise. For example, PEGASUS can be used for gradient estimation by evaluating n policies with k different sources of noise (total of nk trials). From these value estimates, we can estimate the gradient using standard techniques.

Sample reuse using importance sampling

Many gradient estimators throw away the policy trial data after the agent takes a hill-climbing step. However this data may still be used to improve the gradient estimator at the current step. In the case of randomized policies, there is often a non-zero probability that a policy would generate the history observed in a policy trial from a prior hill-climbing step. By storing past histories, an agent can reduce the variance of the gradient estimator.

Importance sampling is a technique that is used to estimate a statistic (e.g., the

mean of a random variable) with respect to a distribution that differs from the sampling distribution. This idea can be applied to reinforcement learning by estimating the mean score with respect to the policy parameters. Suppose that an agent has executed a single policy n times for j different policies. This is typical in the hill-climbing setting where an agent has evaluated a policy at j different hill-climbing steps. The importance sampling equation for the value of a policy is written as

$$V(\pi) \approx \frac{1}{m} \sum_{j=1}^{m} \frac{1}{n} \sum_{i=1}^{n} \frac{P(h^{(ij)}|\pi)}{P(h^{(ij)}|\pi^{(j)})} f(h^{(ij)}), \tag{2.16}$$

where $h^{(ij)}$ is the history of the ith policy trial in the jth hill-climbing step. This approximation could be used for policy search by taking steps in the direction that maximizes the approximate value.

Improvements can be made by considering the normalized importance sampler [Shelton, 2001]. The distribution of the weight terms $\left(\frac{P(h^{(ij)}|\pi)}{P(h^{(ij)}|\pi^{(j)})}\right)$ in the above estimator can have a large variance. By replacing the normalization terms with the sum of the weights, we can reduce the variance of the estimator. The normalized importance sampler is given by the following equation:

$$V(\pi) \approx \frac{1}{\sum_{j=1}^{m} \sum_{i=1}^{n} \frac{P(h^{(ij)}|\pi)}{P(h^{(ij)}|\pi^{(j)})}} \sum_{j=1}^{m} \sum_{i=1}^{n} \frac{P(h^{(ij)}|\pi)}{P(h^{(ij)}|\pi^{(j)})} f(h^{(ij)}). \tag{2.17}$$

Chapter 3

Experimental problems

3.1 Cannon problem

3.1.1 Problem description

The toy cannon problem (Figure 3.1) can be used to help provide insight into the policy search algorithms that will be presented in this dissertation. It is also useful because we can write an analytical expression that closely approximates the true value function and its gradient. The agent's goal is to fire a cannonball so that it hits a target. The state space contains a single state s_0 and for each policy trial, the agent selects an action $a = (a_\theta, a_v)^T$ which consists of a desired cannon angle, $0 \leq a_\theta \leq \pi/2$, and desired initial velocity, $a_v > 0$. We assume that a is perturbed by zero-mean Gaussian noise with covariance matrix Σ_u to give the actual control $u = (u_\theta, u_v)^T$. The agent also has access to a noisy sensor that measures the perturbation $(u-a)$ and we let $o = (o_\theta, o_v)^T$ denote its value; there is additive, zero-mean Gaussian noise in o with covariance matrix Σ_o. From the actual control u, the

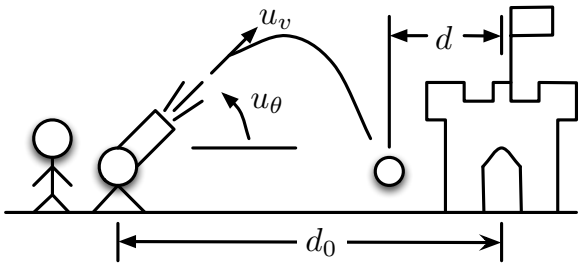


Figure 3.1: Toy cannon problem.

equations of projectile motion are used to determine the distance d from the target to where the cannonball lands. The reward is defined to be $r = -d^2$ and because this is a single-step problem, the score $f(u_\theta, u_v)$ equals r. The history h for this problem contains the action a, the observation o, and reward r. Maximizing $V(\pi) = \mathrm{E}[f(\boldsymbol{u_\theta}, \boldsymbol{u_v})|\pi]$ is equivalent to minimizing the expected squared distance error. For exploration purposes, the agent uses a randomized policy $\pi = (\pi_\theta, \pi_v)^T$ which consists of a nominal cannon angle, $0 \leq \pi_\theta \leq \pi/2$, and nominal initial velocity, $\pi_v > 0$. The agent selects actions by drawing samples from a Gaussian distribution centered around π with covariance matrix Σ_e.

Chapter 4 and Chapter 5 compare the performance of various gradient estimation algorithms using two versions of the cannon problem. In `cannon0` the only stochastic component is due to the use of a randomized policy with exploration noise, $\Sigma_e = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 4 \end{smallmatrix}\right]$. In `cannon1` there is some noise in the controller, $\Sigma_e = \left[\begin{smallmatrix} \frac{1}{10} & 0 \\ 0 & \frac{4}{10} \end{smallmatrix}\right]$, and additional noise in the actuators, $\Sigma_u = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 4 \end{smallmatrix}\right]$. There is also noise in the sensor that measures the difference between the actual and intended controls, $\Sigma_o = \left[\begin{smallmatrix} \frac{1}{10} & 0 \\ 0 & \frac{4}{10} \end{smallmatrix}\right]$.

3.1.2 Analysis

In this subsection we assume that the agent executes a deterministic policy without exploration, i.e., the action a always equals π. Therefore in a completely noise-free setting, the actual control u equals π. In this setting there are an infinite number of policies that exactly hit the target. An equation for the score in the noise-free setting can be derived using the equations of projectile motion:

$$f(\pi_\theta, \pi_v) = -\Big(\frac{{\pi_v}^2 \sin 2\pi_\theta}{g_0} - d_0\Big)^2, \tag{3.1}$$

where d_0 is the distance between the cannon and the target and g_0 is the acceleration due to gravity. Using the above equation we can write an expression for the optimal ${\pi_v}^*$ as a function of π_θ.

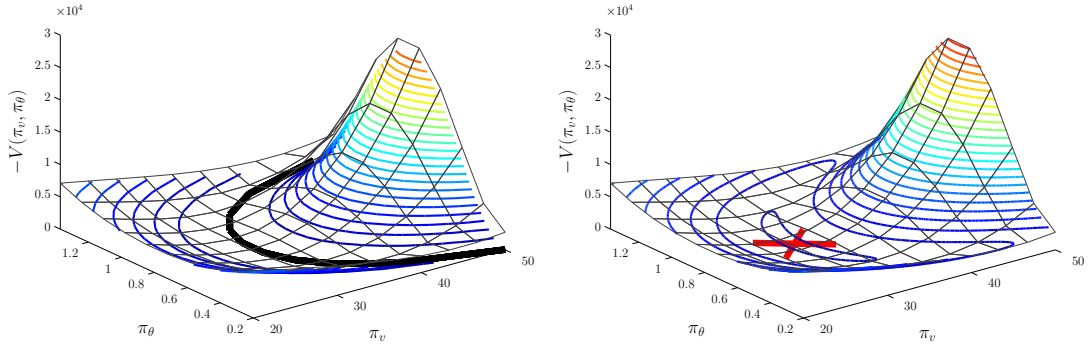
$${\pi_v}^*(\pi_\theta) = \sqrt{\frac{d_0 g_0}{\sin 2\pi_\theta}}. \tag{3.2}$$

Figure 3.2(a) shows contours of the score function with the curve of noise-free optimal solutions superimposed.

In the noisy setting, the value function is written as follows:

$$\begin{aligned} -V(\pi_\theta, \pi_v) &= \mathrm{E}\Big[\Big(\frac{(\pi_v + \boldsymbol{w_v})^2 \sin 2(\pi_\theta + \boldsymbol{w_\theta})}{g_0} - d_0\Big)^2\Big] \\ &= \mathrm{E}\Big[\frac{(\pi_v + \boldsymbol{w_v})^4 \sin^2 2(\pi_\theta + \boldsymbol{w_\theta})}{{g_0}^2} - \frac{2d_0(\pi_v + \boldsymbol{w_v})^2 \sin 2(\pi_\theta + \boldsymbol{w_\theta})}{g_0} + {d_0}^2\Big], \end{aligned}$$

where $\boldsymbol{w_\theta}$ and $\boldsymbol{w_v}$ are zero-mean Gaussian random variables with covariance Σ_u. Obtaining a closed-form expression for the above expectation is difficult but we can approximate the solution, for diagonal $\Sigma_u = \begin{bmatrix} {\sigma_v}^2 & 0 \\ 0 & {\sigma_\theta}^2 \end{bmatrix}$, by using a truncated Taylor series expansion. The



(a) (b)

Figure 3.2: (a) Contours of the value function for a noise-free version of the cannon problem. In this setting there are an infinite number of policies that exactly hit the target (indicated by the thick curve). (b) Contours of the value function of the cannon problem with noise. There is a single optimal policy indicated by the 'X'.

approximate value function is written as follows:

$$-V(\pi_\theta, \pi_v) \approx \frac{({\pi_v}^4 + 6{\pi_v}^2{\sigma_v}^2 + 3{\sigma_v}^4)(\sin^2 2\pi_\theta + 4{\sigma_\theta}^2 \cos 4\pi_\theta)}{{g_0}^2} - \frac{2d({\pi_v}^2 + {\sigma_v}^2)(\sin 2\pi_\theta - 2{\sigma_\theta}^2 \sin 2\pi_\theta)}{g_0} + {d_0}^2.$$

By setting the partial derivatives of $V(\pi_\theta, \pi_v)$ to zero (with respect to π_v and π_θ), we can show that the approximate solution is

$${\pi_v}^* = \sqrt{\frac{d_0 g_0 (1 - 2{\sigma_\theta}^2)}{(1 - 4{\sigma_\theta}^2)} - 3{\sigma_v}^2}, \qquad {\pi_\theta}^* = \pi/4. \tag{3.3}$$

Figure 3.2(b) shows the value function with the optimal policy (denoted by the 'X'). We have found a solution to the Bernstein problem, i.e., the agent has learned how to properly control its redundant degrees of freedom. The noisy setting eliminates the infinite manifold of possible optimal policies and yields a single optimal policy. Notice that the optimal solution is approximately located where the contours lines in the noise-free setting are

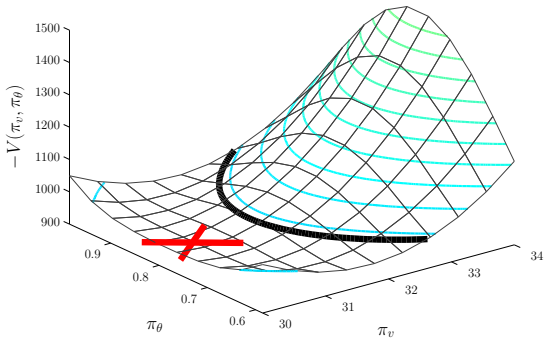


Figure 3.3: A closer look at the contours centered around the optimal policy indicated by the 'X'. The noise-free optimal policies are indicated by the thick curve.

furthest apart. This location is where the policy is least sensitive to perturbations in the control.

In Figure 3.3 we see that the optimal policy does not lie on the noise-free optimal curve. There are two possible reasons for this, which are best explained by considering the noise in each parameter separately. Suppose that there is zero noise in π_v and $\pi_\theta = \pi/4$, then the agent should fire the cannon at a slightly higher velocity than that required in a noise-free setting. This is because errors in the angle, whether positive or negative, will cause the cannonball to land short of the target. If, on the other hand, there is zero noise in the angle, but noise in the velocity, then the agent should fire the cannonball at a slightly lower velocity than that required in the noise-free setting. This is because the squared error of an overshoot is greater than an undershoot of the same magnitude. The relative strengths of these two effects determine how the agent should adjust its velocity in a noisy environment.

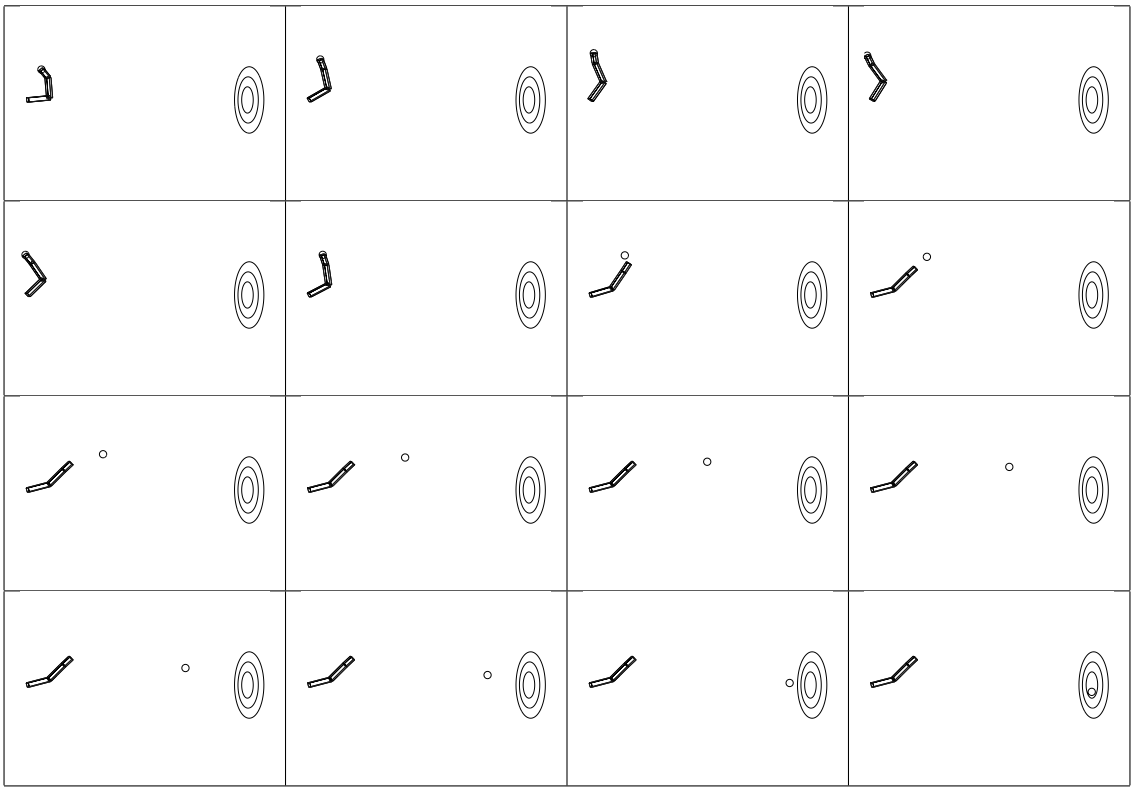


Figure 3.4: The dart problem.

3.2 Dart thrower

The dart problem involves actuating a simulated arm so that it throws a dart with minimal mean squared error (measured from where the dart hits the wall to the center of the dart board). Figure 3.4 shows sixteen frames from a single policy trial. The arm is modeled as a three-link rigid body with dimensions based on biological measurements [Garner and Pandy, 1999]. The links correspond to the upper arm, forearm, and hand and are connected using a single degree of freedom rotational joint. The upper arm is connected to the shoulder at a fixed location. This gives us a 7-dimensional continuous state space ($s \in \mathbb{R}^7$) which consists of the physical state $(x, \dot{x})$ and simulator time t. The physical state

consists of 3 joint angles that describe the position of the arm and 3 corresponding angular velocities. We use SD/Fast to simulate the dynamics of the system [Hollars *et al.*, 1990].

The arm is actuated by applying torques at each joint. These torques are generated by a proportional-derivative controller (PD-controller) that attempts to move the arm through a desired trajectory, specified by a cubic spline for each joint angle. The starting posture of the arm is fixed in advanced and the path is determined by interpolating between three other knot positions. At the start of each policy trial, the agent chooses an action $a \in \mathbb{R}^9$ that contains the knot positions for each joint. In the noise-free setting the torques at each time step are given by the following equation:

$$\tau_t = K_1(\Phi_t a - x_t) - K_2 \dot{x}_t, \tag{3.4}$$

where K_1 is the proportional gain matrix, K_2 is the derivative gain matrix, and Φ_t is a matrix that contains a set of basis functions constructed using cubic splines.

$$\Phi_t = \begin{bmatrix} \phi_1(t) & \phi_2(t) & \phi_3(t) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \phi_1(t) & \phi_2(t) & \phi_3(t) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \phi_1(t) & \phi_2(t) & \phi_3(t) \end{bmatrix},$$

where $\phi_i(t)$ is the ith basis function of a cubic-spline.

Noise enters the system by perturbing the torques given by the PD-controller by additive and multiplicative noise; the torques are written as follows:

$$\tau_t = \Big(I + \text{diag}(\zeta_t)\Big)\Big(K_1(\Phi_t a - x_t) - K_2 \dot{x}_t\Big) + \xi_t, \tag{3.5}$$

where (ζ_t, ξ_t) are sources of noise and diag is a function that takes a column vector as input and returns a diagonal matrix with the input's entries placed on the diagonal. The dynamics are simulated for approximately 0.2 seconds and then the dart is released; there is Gaussian noise added to the release time with $\sigma = 0.01$.

The additive and multiplicative sources of noise are drawn from a stationary Ornstein-Uhlenbeck process [Uhlenbeck and Ornstein, 1930]. This process is similar to a Brownian motion except that it is biased towards some mean value. Samples from these processes at discrete time intervals can be obtained using the following equations:

$$\begin{aligned} \xi_t &= \mu + e^{-\lambda \Delta t}(\xi_{t-1} - \mu) + \sigma\Big(\frac{1 - e^{-2\lambda \Delta t}}{2\lambda}\Big)^{1/2} w_{\xi(t)} \\ \zeta_t &= \mu + e^{-\lambda \Delta t}(\zeta_{t-1} - \mu) + \sigma\Big(\frac{1 - e^{-2\lambda \Delta t}}{2\lambda}\Big)^{1/2} w_{\zeta(t)}, \end{aligned} \tag{3.6}$$

where μ is the mean, λ is the mean reversion rate, σ is the standard deviation parameter, $(w_{\xi(t)}, w_{\zeta(t)})$ are zero-mean multivariate Gaussians with covariance matrix I, and Δt is the simulator's time step. The agent observes noisy measurements of the joint angles at each of these time steps. The observations are drawn from a Gaussian centered around the true state with covariance matrix Σ_o. From these measurements, the angular velocities may be approximated using standard techniques.

The simulation ends when the agent releases the dart. At this time, the equations of projectile motion are used to to determine the distance d from the bull's-eye to where the dart lands. The reward is defined to be $r = -d^2$ and the history h for this problem contains the actions a_t, the observations o_t, and reward r. Maximizing $V(\pi) = \mathrm{E}[f(\mathbf{h})|\pi]$ is equivalent to minimizing the expected squared distance error. For exploration purposes, the agent draws a single action a from a Gaussian distribution with mean π and covariance matrix Σ_e at the beginning of each policy trial.

We consider three versions of the dart throwing problem. In `dart0` the agent executes a single action (i.e., chooses coefficients for the cubic spline bases) for the entire policy trial. The only stochastic component is the use of a randomized policy. In `dart1` there is noise in the actuators and the agent has access to a noisy sensor. The release time,

however, is deterministic. The third version, `dart2`, has additional noise in the release time. This version of the problem contains all of the difficulties discussed in Section 2.1.2. There is noise in the actuators, sensors, and release time. The dynamics are nonlinear and the seven dimensional state space is continuous. Policy search is a good general strategy, but the speed of learning is limited by the ability of the agent to quickly estimate the gradient. Our general approach is to treat this problem as an episodic policy search learning task where the search is done via hill-climbing. From n policy trials we estimate the gradient, take a step in that direction, and repeat the process. The effectiveness of this approach critically depends on our ability to quickly produce accurate estimates of the gradient.

3.3 Quadruped locomotion

The quadruped problem involves actuating a simulated quadruped robot so that it achieves its maximum speed; Figure 3.5 shows sixteen frames from a single policy trial. Each leg of the quadruped has four degrees of freedom (three at the shoulder joint and one at the elbow joint). The absolute position and orientation of the torso add an additional 6 degrees of freedom to the system. This gives us a 45-dimensional continuous state ($s \in \mathbb{R}^{45}$) which consists of the following values: the physical state $(x, \dot{x})$ and the simulator time t. The physical state contains both the joint angles that describe the positions of the legs and the absolute position and orientation of the torso. We use SD/Fast to simulate the dynamics of the system.

The quadruped is actuated by applying torques at each joint. These torques are generated by a PD-controller that attempts to move each leg through a desired trajectory,

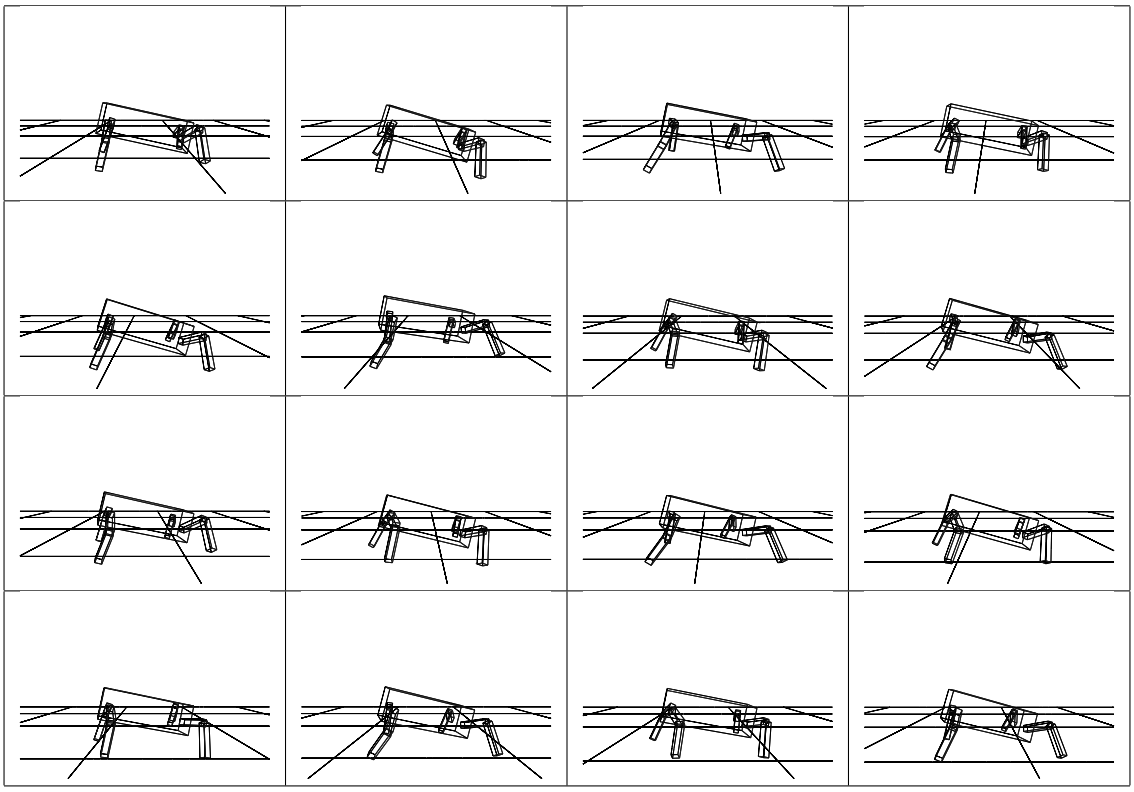


Figure 3.5: The quadruped problem.

specified by a truncated Fourier series for each joint angle. Each controllable degree of freedom has three corresponding parameters and the right side of the body is constrained to move the same as the left except offset by 180 degrees. At the start of each policy trial, the agent chooses an action $a \in \mathbb{R}^{24}$ that contains the coefficients of the Fourier series. The torques and noise are described in Equation 3.5 and Equation 3.6, where Φ_t is a matrix that contains a set of basis functions constructed using a truncated Fourier series. The dynamics are simulated for 3 seconds for each trial.

The reward r_t at each time step is defined to be the forward distance travelled during the preceding interval. The history h for this problem contains the actions a_t, the

observations o_t, and rewards r_t. Maximizing $V(\pi) = \mathrm{E}[f(\mathbf{h})|\pi]$ is equivalent to maximizing the expected speed of the robot. The policy determines the coefficients of the truncated Fourier series for each joint which gives us a compact policy representation (24 policy parameters). For exploration purposes, the agent draws a single action a from a Gaussian distribution with mean π and covariance matrix Σ_e at the beginning of each policy trial.

We consider four versions of the quadruped problem. In `quadruped0` the agent executes a single action (i.e., chooses coefficients for the truncated Fourier series) for the entire policy trial, and in `quadruped1` the agent selects a new action after every 0.3 seconds. The only stochastic component in both of these problems is the use of a randomized policy. We also consider `quadruped2` which is similar to `quadruped0` except that the quadruped experiences actuator noise and is limited by noisy sensors (`quadruped3` is the noisy version of `quadruped1`). These problems present the same challenges that the dart throwing problems present, except some of the challenges are more difficult. In this problem the dynamics are highly nonlinear because the interactions with the ground cause impulse forces on the feet. The 45-dimensional continuous state is also very large. We adopt the same general approach by treating each problem as an episodic policy search learning task where the search is done via hill-climbing. In Chapter 4 and Chapter 5 we present techniques that may be used to reduce the number of policy trials needed before an accurate gradient estimate can be produced. Estimating the gradient with fewer policy trials is very important for this problem because it is expensive to obtain each policy trial (even in simulation). At the time of writing, each policy trial takes approximately 4 seconds on a modern computer.

Chapter 4

Improving gradient estimation using response surface models

This chapter introduces gradient estimation algorithms that reduce the number of policy trials needed for policy search algorithms. We describe how to extend the idea of the baseline by using a response surface model to reduce the variance of the gradient estimator.

4.1 Motivation

The rate at which an agent learns is limited primarily by the number of policy trials it takes before it can produce an accurate estimate of the gradient. In a deterministic smooth setting, a perfect estimate of the gradient evaluated at the current policy π_0 may be obtained by first evaluating a set of policies tightly centered around π_0. The gradient can then be computed using linear regression, provided that the evaluated policy parameters span the entire policy space. In noisy environments, an agent observes perturbations of $V(\pi)$

for each trial. Since each score is an approximation to the true value an agent may need many more trials, when compared to the deterministic case, to construct a good gradient estimate. If we could partially explain the cause of these perturbations, then we should be able to construct better gradient estimators. This chapter exploits this idea and presents several improved gradient estimators.

The REINFORCE estimators (Section 2.2.4) can be very sensitive to both the noise coming from the environment and the randomness embedded in the controller for exploration purposes. This randomness may have a large impact on the quality of the gradient estimates even in cases where there is no environmental noise. In general, we do not have control over the environment but we do have access to the artificially injected random noise used for exploration purposes. This knowledge may be exploited to build better gradient estimators by learning how an agent's choice of exploration influences the score. If this relationship can be learned from a few policy trials, then we can use it to reduce the variance of the gradient estimator.

Figure 4.1 shows a two-dimensional toy problem with four different deterministic scoring functions. Each subfigure shows the deterministic scoring function with four policy trials superimposed, each plotted with respect to the eligibilities. These figures help illustrate why the baseline estimators improve upon the basic REINFORCE algorithm and how the idea can be extended. In this single-time-step problem, the policy parameter π_0 determines the nominal control and the agent explores by adding Gaussian noise to π_0 to determine the action a. The eligibility ψ for this toy problem may be written as $\Sigma^{-1}(a-\pi_0)$, where Σ is the variance of the exploration distribution.

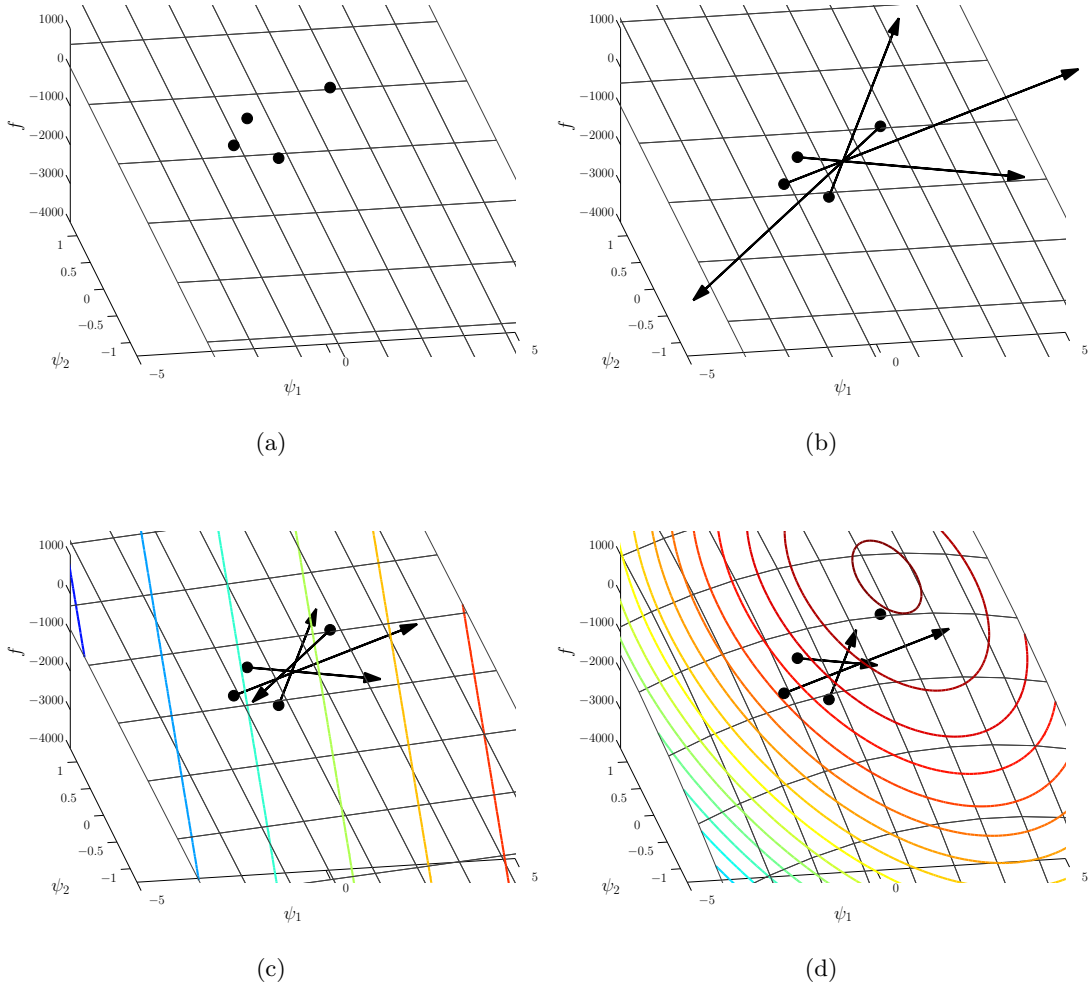


Figure 4.1: Two dimensional toy problem with four different scoring functions: (a) zero, (b) constant, (c) linear, and (d) nonlinear. Emanating from each policy trial is its contribution to the gradient estimate.

In each scenario, the agent executes four policy trials and by chance, its happens to explore more in one direction. Figure 4.1(a) shows a constant score function $f(a) = 0$. The REINFORCE gradient estimate (Equation 2.3) is given by the following equation: $\frac{1}{4}\sum_{i=1}^{4} \psi^{(i)} f^{(i)} = 0$, where superscripts denote the ith policy trial. This gives the correct estimate in the first scenario but fails in Figure 4.1(b), where we use a non-zero constant

score function $f(a) = \beta$. Although these two scenarios differ in only a constant offset term, the estimate is skewed to one side in the second case simply because the agent happened to explore more in that direction. This asymmetric exploration pulls the estimate to one side. This example illustrates the value of including a baseline term. By subtracting β from the observed scores, the agent places itself into the first scenario and correctly estimates the true gradient. The baseline estimate of the gradient is given by the following equation: $\frac{1}{4}\sum_{i=1}^{4} \psi^{(i)}(f^{(i)} - \beta) = 0$. Figure 4.1(c) uses a linear scoring function and again, the gradient estimate is not perfect despite this being a deterministic problem. Later we will see that if we compute the natural gradient using an empirical estimate of the Fisher information matrix, then we could get an error-free estimate of the natural gradient for this particular example. Figure 4.1(d) uses a nonlinear scoring function and this causes an agent to be unable to accurately estimate the gradient from a few policy trials using REINFORCE, the baseline extension, or the natural gradient version.

4.2 Linear response surface models

A response surface model (RSM) predicts a response, i.e., the value of some unknown function, as a function of some input variables [Myers and Montgomery, 1995]. We can construct a response surface model $\hat{f}(h)$ that predicts the scores of histories drawn from a set of policy trials. The general idea is that by choosing a model that closely matches $f(h)$, we will be able to improve the quality of the gradient estimator. This improvement comes from decomposing the estimation of the gradient into two subtasks: computing $\nabla_\pi \mathrm{E}\big[\hat{f}(\mathbf{h})|\pi\big]\Big|_{\pi=\pi_0}$ and estimating $\nabla_\pi \mathrm{E}\big[f(\mathbf{h}) - \hat{f}(\mathbf{h})|\pi\big]\Big|_{\pi=\pi_0}$. The key insight is that we can

often find an exact solution to the first task. Furthermore the variance of the second task is often smaller than the variance encountered when directly estimating $\nabla_\pi \mathrm{E}\big[f(\mathbf{h})|\pi\big]\Big|_{\pi=\pi_0}$. We may write an expression for the gradient using these two components:

$$\nabla_\pi \mathrm{E}\big[f(\mathbf{h})|\pi\big]\Big|_{\pi=\pi_0} = \nabla_\pi \mathrm{E}\big[\hat{f}(\mathbf{h})|\pi\big]\Big|_{\pi=\pi_0} + \nabla_\pi \mathrm{E}\big[f(\mathbf{h}) - \hat{f}(\mathbf{h})|\pi\big]\Big|_{\pi=\pi_0}.$$

When used in this way, $\hat{f}(h)$ is referred to as an additive control variate. This idea has also been used to find optimal value functions in the actor-critic setting [Greensmith *et al.*, 2004].

Let us consider a linear RSM that predicts the score as a function of a feature vector computed from h.

$$\hat{f}(h) := \phi(h)^T \rho, \tag{4.1}$$

where the feature vector $\phi(h) \in \mathbb{R}^m$ is a function of the designer's choice and ρ is a vector of parameters to be fit. We begin the discussion by solely measuring the performance of each policy trial via $f(h)$, ignoring the timing of the individual reward signals received. Later we will consider algorithms that work with a series of returns for each policy trial and compute corresponding feature vectors for each of these returns.

4.2.1 RSM gradients

Suppose that an agent executes n policy trials with parameter π_0 and wants to estimate the gradient at π_0. Let $\Psi = [\psi^{(1)}, \ldots, \psi^{(n)}]^T$ be a matrix of eligibility vectors where $\psi^{(i)} = \psi(h^{(i)}|\pi_0)$, let $\Phi = [\phi^{(1)}, \ldots, \phi^{(n)}]^T$ be a matrix of feature vectors where $\phi^{(i)} = \phi(h^{(i)})$, let $f = [f^{(1)}, \ldots, f^{(n)}]^T$ be a column vector of scores, and define G as follows:

$$G := \mathrm{E}\big[\psi(\mathbf{h}|\pi_0)\boldsymbol{\phi}(\mathbf{h})^T|\pi_0\big] = \frac{1}{n}\mathrm{E}\big[\boldsymbol{\Psi}^T\boldsymbol{\Phi}|\pi_0\big]. \tag{4.2}$$

The features $\phi(h)$ must be designed so that G can be computed.

The baseline gradient estimator can be expressed using the above terms as follows:

$$g(\Psi, f) = \frac{1}{n}\Psi^T(f - 1_n\beta),$$

where 1_n is a column vector of ones. We can extend the idea of the baseline by replacing the baseline term with the response surface model prediction. Let us define the following gradient estimator:

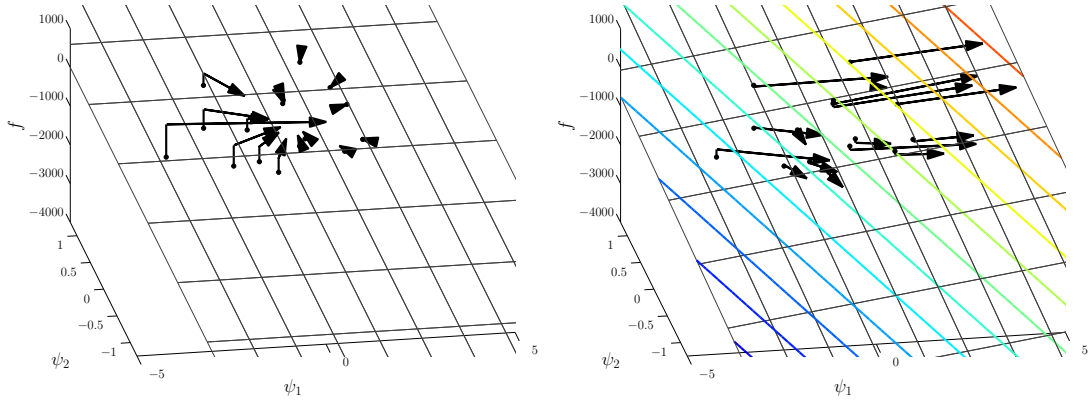
$$g_\rho(\Psi, \Phi, f, G, \rho) = G\rho + \frac{1}{n}\Psi^T(f - \Phi\rho). \tag{4.3}$$

This estimator is constructed by replacing the baseline term with a term that depends on features computed from each history. By choosing a response surface model that closely matches f for each policy trial, the estimator may experience less variance when compared to the baseline case. The term $G\rho$ is needed to ensure that the estimator remains unbiased. We can verify that this is an unbiased estimator by examining its expected value and observing that it is equivalent to the REINFORCE estimator which is unbiased [Williams, 1992].

$$\begin{aligned}\mathrm{E}\big[g_\rho(\boldsymbol{\Psi}, \boldsymbol{\Phi}, \mathbf{f}, G, \rho)|\pi_0\big] &= \frac{1}{n}\,\mathrm{E}\big[\boldsymbol{\Psi}^T\boldsymbol{\Phi}|\pi_0\big]\rho + \frac{1}{n}\,\mathrm{E}\big[\boldsymbol{\Psi}^T\mathbf{f}|\pi_0\big] - \frac{1}{n}\,\mathrm{E}\big[\boldsymbol{\Psi}^T\boldsymbol{\Phi}|\pi_0\big]\rho \\ &= \frac{1}{n}\,\mathrm{E}\big[\boldsymbol{\Psi}^T\mathbf{f}|\pi_0\big].\end{aligned}$$

By choosing an appropriate ρ, we can reduce the variance of the gradient estimator as was done in the baseline case. Note that if we set the feature vector equal to some constant $\phi(h) = k \neq 0$, then we recover the baseline estimator as a special case.

Figure 4.2(a) shows 16 policy trials where we superimpose the corresponding single-trial gradient estimates (emanating from each trial) computed using REINFORCE without a baseline. Since this algorithm is equivalent to using an RSM that predicts zero for every



(a) (b)

Figure 4.2: (a) 16 policy trials with the corresponding single-trial gradient estimates (emanating from each trial) computed using REINFORCE without a baseline. (b) 16 policy trials with the corresponding single-trial gradient estimates (emanating from each trial) computed using the optimal linear RSM.

h, for comparison purposes we also plot the zero-RSM and the residuals (i.e., the difference between the RSM predicted score and the observed score). Figure 4.2(b) shows these same policy trials with the optimal linear RSM and residuals. Notice that the residuals are much smaller and that the variance of the single-trial gradient estimates is much smaller than in the standard case. These gradient estimates all point in the general direction of the gradient whereas in Figure 4.2(a), some point in the opposite direction of the true gradient.

Optimal RSM

Equation 4.3 takes an average over many single-trial gradient estimates. Since each of the n policy trials is conditionally independent of the others (conditioned on π_0), the optimal response surface model may be approximated by minimizing the trace of the covariance matrix of these single trial estimates. This optimal model can be estimated by

taking the derivative of the trace with respect to ρ. By setting this value to zero and solving for ρ we get an estimate of the optimal response surface model ρ^* around the current policy. The variance of the estimator is approximated by using the sample variance:

$$\mathrm{var}\big[g_\rho(\mathbf{\Psi}, \mathbf{\Phi}, \mathbf{f}, G, \rho)|\pi_0\big] \approx \frac{1}{n^2}\sum_{i=1}^{n} \psi^{(i)}\big(f^{(i)} - \phi^{(i)^T}\rho\big)^2\psi^{(i)^T} + k,$$

where k is a term whose derivative does not depend on ρ. We take the derivative of the trace and solve for the optimal parameter ρ^*.

$$\begin{aligned}
\mathrm{tr}(\mathrm{var}\big[g_\rho(\mathbf{\Psi}, \mathbf{\Phi}, \mathbf{f}, G, \rho)|\pi_0\big]) &\approx \mathrm{tr}\Big(\frac{1}{n^2}\sum_{i=1}^{n} \psi^{(i)^T}\psi^{(i)}(f^{(i)} - \phi^{(i)^T}\rho)^2\Big) + \mathrm{tr}(k) \\
d\,\mathrm{tr}(\mathrm{var}\big[g_\rho(\mathbf{\Psi}, \mathbf{\Phi}, \mathbf{f}, G, \rho)|\pi_0\big]) &\approx -\frac{2}{n^2}\sum_{i=1}^{n} \psi^{(i)^T}\psi^{(i)}\big(f^{(i)} - \phi^{(i)^T}\rho\big)\phi^{(i)^T}d\rho \\
0 &\approx \sum_{i=1}^{n} f^{(i)}\psi^{(i)^T}\psi^{(i)}\phi^{(i)^T} - \rho^T\sum_{i=1}^{n}\phi^{(i)}\psi^{(i)^T}\psi^{(i)}\phi^{(i)^T} \\
\rho^* &\approx \big(\sum_{i=1}^{n}\phi^{(i)}\psi^{(i)^T}\psi^{(i)}\phi^{(i)^T}\big)^{-1}\sum_{i=1}^{n}\phi^{(i)}\psi^{(i)^T}\psi^{(i)}f^{(i)}
\end{aligned}$$

This is equivalent to performing a weighted linear regression where data from the ith policy trial is weighted by $\psi^{(i)^T}\psi^{(i)}$. When we use $g_\rho(\Psi, \Phi, f, G, \rho^*)$ to estimate the gradient, we get the lowest variance estimator in the family of estimators defined by the space of ρ. In practice, we do not know the optimal response surface model and so we must estimate it from the policy trials themselves. This process introduces bias into the gradient estimator task. We may rewrite the approximation of ρ^* as follows:

$$\rho^* \approx \big(\Phi^T\mathrm{ddiag}(\Psi\Psi^T)\Phi\big)^{-1}\Phi^T\mathrm{ddiag}(\Psi\Psi^T)f, \tag{4.4}$$

where ddiag is a function that returns its input with the non-diagonal entries set to zero.

Variance

By considering locally linear scoring functions, we may rewrite the variance of the plain REINFORCE estimator so that it is easier to analyze. This will allow us to compare the quality of the gradient estimator to others, including the baseline and RSM variants. Here we assume that the true score function is a linear function of the eligibilities, $f = \Psi a_\psi + 1_n b + w$ where w is independent additive noise (possibly non-Gaussian) with variance σ^2. We also assume that the eligibility terms are drawn from a matrix normal distribution $\Psi \sim \mathcal{N}(0, I, \Sigma)$ (see Kollo and Rosen [2005] for properties of the matrix normal distribution). This will be the case when an agent's randomized policy explores using a Gaussian distribution. Using these assumptions we can derive an expression for the mean.

$$\begin{aligned} g(\Psi, f) &= \frac{1}{n}\Psi^T(\Psi a_\psi + 1_n b + w) \\ \mathrm{E}\big[g(\mathbf{\Psi}, \mathbf{f})|\pi_0\big] &= \frac{1}{n}\,\mathrm{E}\big[\mathbf{\Psi}^T\mathbf{\Psi}a_\psi|\pi_0\big] + \frac{1}{n}\,\mathrm{E}\big[\mathbf{\Psi}^T 1_n b|\pi_0\big] + \frac{1}{n}\,\mathrm{E}\big[\mathbf{\Psi}^T\mathbf{w}|\pi_0\big] \\ &= \Sigma a_\psi. \end{aligned}$$

We can also derive an expression for the variance:

$$\begin{aligned} \mathrm{var}\big[g(\mathbf{\Psi}, \mathbf{f})|\pi_0\big] &= \frac{1}{n^2}\,\mathrm{E}\big[\mathbf{\Psi}^T(\mathbf{\Psi}a_\psi + 1_n b + \mathbf{w})(\mathbf{\Psi}a_\psi + 1_n b + \mathbf{w})^T\mathbf{\Psi}|\pi_0\big] - \Sigma a_\psi a_\psi^T\Sigma^T \\ &= \frac{1}{n^2}\,\mathrm{tr}({I_n}^2)\,\mathrm{tr}(a_\psi a_\psi^T\Sigma)\Sigma + \frac{1}{n^2}\,\mathrm{tr}(I_n)^2\Sigma a_\psi a_\psi^T\Sigma + \\ &\quad \frac{1}{n^2}\,\mathrm{tr}({I_n}^2)\Sigma a_\psi a_\psi^T\Sigma + \frac{b^2}{n^2}\,\mathrm{tr}(1_n 1_n^T)\Sigma + \frac{1}{n^2}\,\mathrm{tr}(\sigma^2 I_n)\Sigma - \Sigma a_\psi a_\psi^T\Sigma^T \\ &= \frac{1}{n}\Big(\mathrm{tr}(a_\psi a_\psi^T\Sigma)\Sigma + \Sigma a_\psi a_\psi^T\Sigma + (b^2 + \sigma^2)\Sigma\Big). \end{aligned} \tag{4.5}$$

Let us take a qualitative look at the expression for the variance. It is proportional to the inverse of the number of samples. In the first two terms of the expression, the variance increases as the magnitude of a_ψ increases and as Σ increases. The last term depends on

the variance of the eligibility terms Σ and is scaled by a factor of $(b^2 + \sigma^2)$. Here we see the advantage of using the baseline estimator. The constant offset of the scoring function b does not change the gradient but affects the variance of the estimator. By subtracting a baseline, we can make this component disappear.

We can also derive an expression for the variance of the RSM gradient estimator. Suppose that $f = \Psi a_\psi + 1_n b + w$ and $\Phi\rho = \Psi\alpha + 1_n\beta$.

$$\begin{aligned}
\operatorname{var}\big[g_\rho(\boldsymbol{\Psi}, \boldsymbol{\Phi}, \mathbf{f}, G, \rho)|\pi_0\big] &= \frac{1}{n^2}\operatorname{E}\big[\boldsymbol{\Psi}^T(\boldsymbol{\Psi}a_\psi + 1_n b + \mathbf{w} - \boldsymbol{\Psi}\alpha - 1_n\beta) \\
&\qquad (\boldsymbol{\Psi}a_\psi + 1_n b + \mathbf{w} - \boldsymbol{\Psi}\alpha - 1_n\beta)^T\boldsymbol{\Psi}|\pi_0\big] - \Sigma a_\psi a_\psi^T\Sigma^T \\
&= \frac{1}{n}\operatorname{tr}((a_\psi - \alpha)(a_\psi - \alpha)^T\Sigma)\Sigma + \frac{1}{n}\Sigma(a_\psi - \alpha)(a_\psi - \alpha)^T\Sigma + \\
&\qquad \frac{((b-\beta)^2 + \sigma^2)}{n}\Sigma.
\end{aligned}$$

By inspection we see that if we set $\alpha = a_\psi$ then the first two terms of the above expression equal zero and if we set $\beta = b$ then the third term is at its minimal possible value. Because the baseline estimators lack the linear term α, the family of baseline estimators can be formed by setting $\alpha = 0$ and considering changes in β. Notice that the difference between the RSM and baseline variants becomes more pronounced as the magnitude of a_ψ increases. For the linear scoring function (and Gaussian exploration) the best RSM estimator has a variance of $\frac{\sigma^2}{n}\Sigma$.

The RSM gradient estimator algorithm

Algorithm 4.1 shows an improved gradient estimation algorithm that uses an RSM. It requires the following inputs: Ψ, the eligibility vectors; f, the scores; $\{\Sigma_i\}_{i\in\{1,\ldots,n\}}$, covariance matrices where $\Sigma_i = \operatorname{var}\big[\psi(\mathbf{h}^{(i)}|\pi_0)|\pi_0\big]$; `features`, a user-defined function that

Algorithm 4.1: RSM Gradient Estimator

Input: Ψ, the eligibility vectors; f, the scores; $\{\Sigma_i\}_{i\in\{1,\ldots,n\}}$, covariance matrices where $\Sigma_i = \mathrm{var}\big[\psi(\mathbf{h}^{(i)}|\pi_0)|\pi_0\big]$; `features`, a user-defined function that computes Φ and G; Λ_ρ, a regularization parameter.

Output: ∇, an estimate of the gradient

$n \leftarrow$ number of rows in Ψ

$W \leftarrow \frac{1}{n}\mathrm{ddiag}(\Psi\Psi^T)$ `// ddiag clears non-diagonal entries`

$[\Phi, G] \leftarrow \mathrm{features}(\Psi, \{\Sigma_i\}_{i\in\{1,\ldots,n\}})$

$\rho \leftarrow \left(\Phi^T W \Phi + \Lambda_\rho\right)^{-1} \Phi^T W f$

$\nabla \leftarrow G\rho + \frac{1}{n}\Psi(f - \Phi\rho)$

return ∇

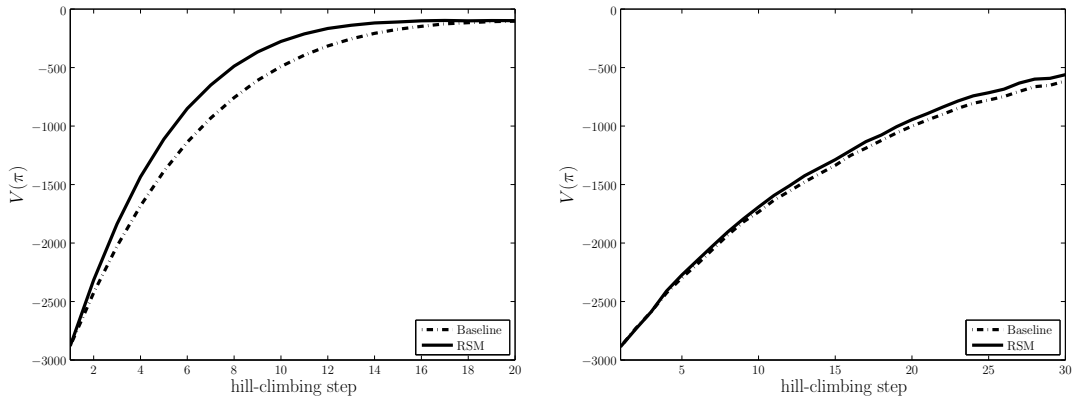
computes Φ and G; and Λ_ρ, a regularization parameter.

We applied this algorithm, with a linear response model ($\phi^{(i)} = [1, \psi^{(i)^T}]$), to the `cannon0` and `cannon1` problems described in Section 3.1. The results are shown in Figure 4.3. Each hill-climbing run lasts 30 steps (Figure 4.3(a) shows only 20 steps) and at each step we drew 5 policy trials from the current policy; the results were averaged over 1000 hill-climbing runs. Notice that the RSM gradient estimator outperforms the baseline gradient estimator. The `cannon1` learning performance improvement is relatively poor because of the presence of actuator noise. In Chapter 5 we explore ways to improve the performance in this setting by reasoning about the sensor data.

4.2.2 Natural RSM gradients

The natural gradient estimators pre-multiply the regular gradient estimators by the inverse of the Fisher information matrix: $\mathcal{F} = \mathrm{E}\big[\boldsymbol{\psi}(\mathbf{h}|\pi_0)\boldsymbol{\psi}(\mathbf{h}|\pi_0)^T|\pi_0\big]$. The following estimator is biased but the bias is guaranteed to result in an estimate that is less than $\pi/2$ radians (in expectation) away from the true gradient:

$$\widetilde{g}(\Psi, f) := \mathcal{F}^{-1} g(\Psi, f). \tag{4.6}$$



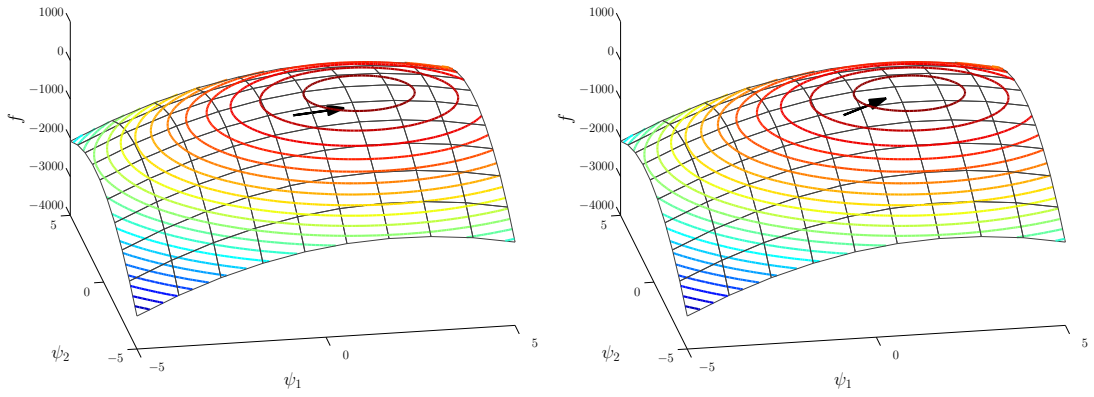
(a) (b)

Figure 4.3: (a) The learning curve performance of the baseline and RSM gradient estimators for the `cannon0` problem. (b) The learning curve performance of the baseline and RSM gradient estimators for the `cannon1` problem.

Figure 4.4(a) shows the gradient of the value with respect to π_0 superimposed on the contours of f, which are plotted in eligibility space. Figure 4.4(b) shows the natural gradient which points towards the maximum of the surface. Let us define a natural gradient estimator that incorporates the response surface model idea:

$$\begin{aligned} \widetilde{g}_\rho(\Psi, \Phi, f, \widetilde{G}, \rho) &= \widetilde{G}\rho + \mathcal{F}^{-1}\Psi^T(f - \Phi\rho) \\ \widetilde{G} &:= \frac{\mathcal{F}^{-1}}{n} \mathrm{E}\big[\boldsymbol{\Psi}^T \boldsymbol{\Phi} | \pi_0\big]. \end{aligned} \tag{4.7}$$

The natural gradient of the response surface is $\widetilde{G}\rho$, which must be added to the estimator so that it remains unbiased. We can verify that this is an unbiased estimator by taking its expected value and observing that it is equivalent to the natural REINFORCE



(a) (b)

Figure 4.4: (a) Gradient of the value with respect to π_0 superimposed on the contours of the score. (b) Natural gradient which is guaranteed to be in the correct half-space.

estimator which is known to be unbiased.

$$\begin{aligned}\mathrm{E}\big[\widetilde{g}_\rho(\boldsymbol{\Psi},\boldsymbol{\Phi},\mathbf{f},\widetilde{G},\rho)|\pi_0\big] &= \frac{\mathcal{F}^{-1}}{n}\mathrm{E}\big[\boldsymbol{\Psi}^T\boldsymbol{\Phi}|\pi_0\big]\rho + \frac{\mathcal{F}^{-1}}{n}\mathrm{E}\big[\boldsymbol{\Psi}^T\mathbf{f}|\pi_0\big] - \\ &\quad \frac{\mathcal{F}^{-1}}{n}\mathrm{E}\big[\boldsymbol{\Psi}^T\boldsymbol{\Phi}|\pi_0\big]\rho \\ &= \frac{\mathcal{F}^{-1}}{n}\mathrm{E}\big[\boldsymbol{\Psi}^T\mathbf{f}|\pi_0\big].\end{aligned}$$

By choosing an appropriate ρ, we can reduce the variance of the natural gradient estimator. Note that if we set the feature vector equal to some constant $\phi(h) = k \neq 0$, then we recover the natural baseline as a special case.

Optimal RSM

Equation 4.7 takes an average over many single-trial gradient estimates and pre-multiplies the result by an estimate of $\mathcal{F}^{-1}$. Since each of the n policy trials is conditionally independent of the others (conditioned on π_0), the optimal response surface model may be approximated by minimizing the trace of the covariance matrix of these single trial

estimates. We can estimate the optimal model by taking the derivative of the trace with respect to ρ, setting its value to zero, and solving for ρ. The variance of the estimator can be approximated by using the sample variance:

$$\mathrm{var}\big[\widetilde{g}_\rho(\mathbf{\Psi},\mathbf{\Phi},\mathbf{f},\widetilde{G},\rho)|\pi_0\big] \approx \frac{1}{n^2}\mathcal{F}^{-1}\Big(\sum_{i=1}^{n}\psi^{(i)}\big(f^{(i)}-\phi^{(i)^T}\rho\big)^2\psi^{(i)^T}\Big)\mathcal{F}^{-1}+k,$$

where k is a term whose derivative does not depend on ρ. We take the derivative of the trace and solve for the optimal parameter ρ^*.

$$\mathrm{tr}(\mathrm{var}\big[\widetilde{g}_\rho(\mathbf{\Psi},\mathbf{\Phi},\mathbf{f},\widetilde{G},\rho)|\pi_0\big]) \approx \mathrm{tr}\Big(\frac{1}{n^2}\sum_{i=1}^{n}\psi^{(i)^T}\mathcal{F}^{-2}\psi^{(i)}(f^{(i)}-\phi^{(i)^T}\rho)^2\Big)+\mathrm{tr}(k)$$

$$d\,\mathrm{tr}(\mathrm{var}\big[\widetilde{g}_\rho(\mathbf{\Psi},\mathbf{\Phi},\mathbf{f},\widetilde{G},\rho)|\pi_0\big]) \approx -\frac{2}{n^2}\sum_{i=1}^{n}\psi^{(i)^T}\mathcal{F}^{-2}\psi^{(i)}\big(f^{(i)}-\phi^{(i)^T}\rho\big)\phi^{(i)^T}d\rho$$

$$0 \approx \sum_{i=1}^{n} f^{(i)}\psi^{(i)^T}\mathcal{F}^{-2}\psi^{(i)}\phi^{(i)^T}-\rho^T\sum_{i=1}^{n}\phi^{(i)}\psi^{(i)^T}\mathcal{F}^{-2}\psi^{(i)}\phi^{(i)^T}$$

$$\rho^* \approx \big(\sum_{i=1}^{n}\phi^{(i)}\psi^{(i)^T}\mathcal{F}^{-2}\psi^{(i)}\phi^{(i)^T}\big)^{-1}\sum_{i=1}^{n}\phi^{(i)}\psi^{(i)^T}\mathcal{F}^{-2}\psi^{(i)}f^{(i)}$$

Thus, when we use $\widetilde{g}_\rho(\Psi,\Phi,f,\widetilde{G},\rho^*)$ to estimate the natural gradient, we get the lowest variance estimator in the family of estimators defined by the space of ρ. In practice, we do not know the optimal response surface model and so we must estimate it from the policy trials themselves. As before this process introduces bias into the gradient estimator task. We may rewrite the approximation of ρ^* as follows:

$$\rho^* \approx \big(\Phi^T\mathrm{ddiag}(\Psi\mathcal{F}^{-2}\Psi^T)\Phi\big)^{-1}\Phi^T\mathrm{ddiag}(\Psi\mathcal{F}^{-2}\Psi^T)f, \tag{4.8}$$

if we have an analytical solution for $\mathcal{F}^{-2}$ or use the empirical estimate:

$$\rho^* \approx \big(\Phi^T\mathrm{ddiag}(\Psi(\Psi^T\Psi)^{-2}\Psi^T)\Phi\big)^{-1}\Phi^T\mathrm{ddiag}(\Psi(\Psi^T\Psi)^{-2}\Psi^T)f, \tag{4.9}$$

Variance

By considering locally linear scoring functions we may rewrite the variance of the natural REINFORCE estimator (Equation 2.15) so that it is easier to analyze. Although we often have an analytical solution to the Fisher information matrix, we follow Peters and Schaal [2006] and use the empirical estimate from the policy trial data. The natural REINFORCE gradient estimator can be expressed as follows:

$$\widetilde{g}(\Psi, f) = (\Psi^T \Psi)^{-1} \Psi f.$$

Let us assume that the scoring function is linear in the eligibilities, i.e., $f = \Psi a_\psi + 1_n b + w$ with Gaussian noise $w \sim \mathcal{N}(0, \sigma^2 I)$, and that the eligibility terms are drawn from a matrix normal distribution $\Psi \sim \mathcal{N}(0, I, \Sigma)$. Using these assumptions we can derive an expression for the mean.

$$\begin{aligned}
\widetilde{g}(\Psi, f) &= (\Psi^T \Psi)^{-1} \Psi^T (\Psi a_\psi + 1_n b + w) \\
\mathrm{E}\big[\widetilde{g}(\mathbf{\Psi}, \mathbf{f}) | \pi_0\big] &= \mathrm{E}\big[(\mathbf{\Psi}^T \mathbf{\Psi})^{-1} \mathbf{\Psi}^T \mathbf{\Psi} a_\psi | \pi_0\big] + \mathrm{E}\big[(\mathbf{\Psi}^T \mathbf{\Psi})^{-1} \mathbf{\Psi}^T 1_n b | \pi_0\big] + \mathrm{E}\big[(\mathbf{\Psi}^T \mathbf{\Psi})^{-1} \mathbf{\Psi}^T \mathbf{w} | \pi_0\big] \\
&= a_\psi.
\end{aligned}$$

We can also derive an expression for the variance:

$$\begin{aligned}
\mathrm{var}\big[\widetilde{g}(\mathbf{\Psi}, \mathbf{f}) | \pi_0\big] &= \frac{1}{n^2} \mathrm{E}\big[(\mathbf{\Psi}^T \mathbf{\Psi})^{-1} \mathbf{\Psi}^T (\mathbf{\Psi} a_\psi + 1_n b + \mathbf{w})(\mathbf{\Psi} a_\psi + 1_n b + \mathbf{w})^T \mathbf{\Psi} (\mathbf{\Psi}^T \mathbf{\Psi})^{-1} | \pi_0\big] - \\
&\quad a_\psi a_\psi^T \\
&= a_\psi a_\psi^T + b^2 \, \mathrm{E}\big[(\mathbf{\Psi}^T \mathbf{\Psi})^{-1} \mathbf{\Psi}^T 1_n 1_n^T \mathbf{\Psi} (\mathbf{\Psi}^T \mathbf{\Psi})^{-1} | \pi_0\big] + \\
&\quad \mathrm{E}\big[(\mathbf{\Psi}^T \mathbf{\Psi})^{-1} \mathbf{\Psi}^T \mathbf{w} \mathbf{w}^T \mathbf{\Psi} (\mathbf{\Psi}^T \mathbf{\Psi})^{-1} | \pi_0\big] - a_\psi a_\psi^T \\
&= \frac{(b^2 + \sigma^2)(\Sigma^{-1})}{n - d - 1}
\end{aligned}$$

We can also derive an expression for the variance of the response surface model. Suppose that $f = \Psi a_\psi + 1_n b + w$ and $\Phi\rho = \Psi\alpha + 1_n\beta$.

$$\begin{aligned}
\mathrm{var}\big[\widetilde{g}_\rho(\boldsymbol{\Psi}, \boldsymbol{\Phi}, \mathbf{f}, \widetilde{G}, \rho)|\pi_0\big] &= \frac{1}{n^2}\,\mathrm{E}\big[(\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^T(\boldsymbol{\Psi}a_\psi + 1_n b + \mathbf{w} - \boldsymbol{\Psi}\alpha - 1_n\beta) \\
&\qquad (\boldsymbol{\Psi}a_\psi + 1_n b + \mathbf{w} - \boldsymbol{\Psi}\alpha - 1_n\beta)^T\boldsymbol{\Psi}(\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}|\pi_0\big] - a_\psi a_\psi^T \\
&= (b-\beta)^2\,\mathrm{E}\big[(\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}|\pi_0\big] + \sigma^2\,\mathrm{E}\big[(\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}|\pi_0\big] \\
&= \frac{((b-\beta)^2 + \sigma^2)\Sigma^{-1}}{n-d-1}
\end{aligned}$$

In this situation we see that the linear component of the response surface model does not play a role in reducing the variance. Thus to get a benefit in the natural gradient setting, one must include second order or higher terms in the response surface model. As before, if we set $\beta = b$ then the variance is at its minimal possible value. For the linear scoring function (and Gaussian exploration) the best natural linear RSM estimator has a variance of $\frac{\sigma^2\Sigma^{-1}}{n-d-1}$.

It would not be fair to directly compare the variances between the natural and standard RSM gradient estimators because they estimate different quantities. However, given an analytical solution to the Fisher information matrix we can convert the standard RSM gradient estimator to the natural version. Assuming a locally linear scoring function, Gaussian distribution of eligibilities, and optimal RSM parameter ρ^* we may write the variance of the converted estimator as follows:

$$\mathrm{var}\big[\mathcal{F}^{-1}g_\rho(\boldsymbol{\Psi}, \boldsymbol{\Phi}, \mathbf{f}, G, \rho^*)|\pi_0\big] = \Sigma^{-1}\left(\frac{\sigma^2}{n}\Sigma\right)\Sigma^{-1} = \frac{\sigma^2}{n}\Sigma^{-1}.$$

Thus given the optimal response surface model, the variance of the natural RSM gradient estimator ($\frac{\sigma^2\Sigma^{-1}}{n-d-1}$) is larger than the standard RSM gradient when we transform it to the

Algorithm 4.2: Natural RSM Gradient Estimator

Input: Ψ, the eligibility vectors; f, the scores; $\{\Sigma_i\}_{i\in\{1,\ldots,n\}}$, covariance matrices where $\Sigma_i = \mathrm{var}\big[\psi(\mathbf{h}^{(i)}|\pi_0)|\pi_0\big]$; `features`, a user-defined function that computes Φ and G; Λ_ρ, a regularization parameter.

Output: ∇, an estimate of the gradient

$n \leftarrow$ number of rows in Ψ

$\mathcal{F} \leftarrow \mathrm{mean}(\{\Sigma_i\}_{i\in\{1,\ldots,n\}})$

$W \leftarrow \frac{1}{n}\mathrm{ddiag}(\Psi\mathcal{F}^{-2}\Psi^T)$ `// ddiag clears non-diagonal entries`

$[\Phi, G] \leftarrow$ `features`$(\Psi, \{\Sigma_i\}_{i\in\{1,\ldots,n\}})$

$\rho \leftarrow \left(\Phi^T W \Phi + \Lambda_\rho\right)^{-1} \Phi^T W f$

$\nabla = \mathcal{F}^{-1} G\rho + (\Psi^T\Psi)^{-1}\Psi(f - \Phi\rho)$

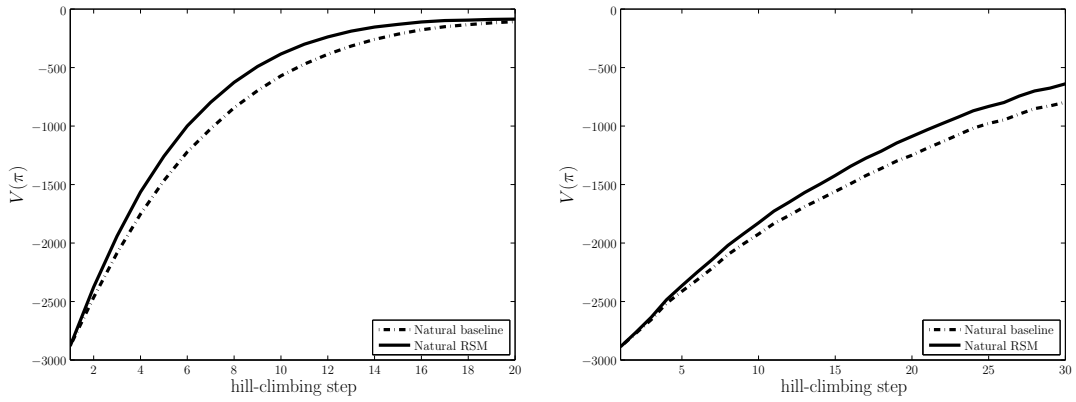
return ∇

natural setting ($\frac{\sigma^2}{n}\Sigma^{-1}$). This difference is more pronounced as the dimensionality of the policy increases. However, this assumes that an agent knows the true optimal RSM parameter. Since the linear component is unnecessary for the natural case, one only needs to learn the baseline (or 2nd order or higher terms). Thus in practice the natural gradient with constant baseline may outperform the linear RSM gradient estimator.

The natural RSM gradient estimator algorithm

Algorithm 4.2 shows an improved natural gradient estimation algorithm that uses an RSM. It requires the following inputs: Ψ, the eligibility vectors; f, the scores;$\{\Sigma_i\}_{i\in\{1,\ldots,n\}}$, covariance matrices where $\Sigma_i = \mathrm{var}\big[\psi(\mathbf{h}^{(i)}|\pi_0)|\pi_0\big]$; `features`, a user-defined function that computes Φ and G; and Λ_ρ, a regularization parameter.

We applied this algorithm, with a linear response model ($\phi^{(i)} = [1, {\psi^{(i)}}^T]$), to the `cannon0` and `cannon1` problems described in Section 3.1. The results are shown in Figure 4.5. Each hill-climbing run lasts 30 steps (Figure 4.5(a) shows only 20 steps) and at each step we drew 5 policy trials from the current policy; the results were averaged



(a) (b)

Figure 4.5: (a) The learning curve performance of the natural baseline and natural RSM gradient estimators for the `cannon0` problem. (b) The learning curve performance of the natural baseline and natural RSM gradient estimators for the `cannon1` problem.

over 1000 hill-climbing runs. The natural RSM gradient estimator outperforms the natural baseline gradient estimator. Both of these algorithms perform slightly worse than their standard counterparts. We should note, however, that the effectiveness of the natural gradient algorithms depends on the Fisher information matrix, a quantity that is problem specific. We may see improvements on problems that use different exploration distributions.

4.2.3 Time-variant RSM gradients

The RSM gradient estimator constructs a score for each policy trial by summing the individual rewards received. This approach views each policy trial as though it came from a single-time-step problem by computing an eligibility vector over the entire history. Further improvements may be possible by reasoning about the relationship between the individual rewards received and the agent's choice of actions. There is a causal relationship between actions and rewards (i.e., an agent's future choice of actions does not affect

previously received rewards). In many problems, actions often do not affect the distribution of rewards received far into the future. This is exploited in the GPOMDP algorithm (Equation 2.12) by introducing a discounting factor.

In Equation 2.11 we saw that the gradient could be expressed as follows:

$$\nabla_\pi V(\pi)\Big|_{\pi=\pi_0} \approx \sum_{t=0}^{t_f} \psi(a_t|o_t,\pi_0) \sum_{u=t}^{t_f} r_u.$$

We may improve this estimator by subtracting out a prediction of the return (i.e., $\sum_{u=t}^{t_f} r_u$) for each time step as a function of the history. By choosing a model that closely matches each of these returns, we will be able to improve the quality of the gradient estimator. Let us consider a linear response surface model that predicts the return as a function of a feature vector computed from the history h and time t.

$$\hat{f}(h,t) := \phi(h,t)^T \rho, \tag{4.10}$$

where the feature vector $\phi(h,t) \in \mathbb{R}^m$ is a function of the designer's choice and ρ is a vector of parameters to be fit.

Suppose that an agent executes n policy trials with parameter π_0 and wants to estimate the gradient at π_0. Let $\Psi_t = [\psi_t^{(1)}, \cdots, \psi_t^{(n)}]^T$ be a matrix of eligibility vectors where $\psi_t^{(i)} = \psi(a_t^{(i)}|o_t^{(i)},\pi_0)$, let $\Phi_t = [\phi_t^{(1)}, \cdots, \phi_t^{(n)}]^T$ be a matrix of feature vectors where $\phi_t^{(i)} = \phi(h^{(i)},t)$, let $f_t = \sum_{u=t}^{t_f} [r_u^{(1)}, \cdots, r_u^{(n)}]^T$ be a column vector of returns, and define G as follows:

$$G = \sum_{t=1}^{t_f} \mathrm{E}\big[\psi_t(\mathbf{a}_t|\mathbf{o}_t,\pi_0)\phi(\mathbf{h},t)^T|\pi_0\big] = \frac{1}{n}\sum_{t=1}^{t_f} \mathrm{E}\big[\mathbf{\Psi}_t^T \mathbf{\Phi}_t|\pi_0\big]. \tag{4.11}$$

The features $\phi(h,t)$ must be designed so that G can be computed. We store these terms in the following sets: $\Psi = \{\Psi_t\}_{t\in\{1,\ldots,t_f\}}$, $\Phi = \{\Phi_t\}_{t\in\{1,\ldots,t_f\}}$, and $\mathrm{f} = \{f_t\}_{t\in\{1,\ldots,t_f\}}$. The base-

line gradient estimator can be expressed by subtracting out a baseline from Equation 2.11.

$$g(\psi, \mathrm{f}) = \frac{1}{n} \sum_{t=1}^{t_f} \Psi_t^T (f_t - 1_n \beta). \tag{4.12}$$

We proceed by replacing the baseline with the response surface model prediction from the scores of the gradient estimator. This gives the following gradient estimator:

$$g_\rho(\psi, \phi, \mathrm{f}, G, \rho) := G\rho + \frac{1}{n} \sum_{t=1}^{t_f} \Psi_t^T (f_t - \Phi_t \rho), \tag{4.13}$$

To compensate for this change we need to add the term $G\rho$ so that the estimator remains unbiased. To show that this estimator is unbiased we take its expectation and observe that it is equivalent to the expected value of Equation 4.12 when $\beta = 0$.

$$\begin{aligned} \mathrm{E}\big[g_\rho(\boldsymbol{\psi}, \boldsymbol{\phi}, \mathbf{f}, G, \rho)|\pi_0\big] &= \frac{1}{n} \sum_{t=1}^{t_f} \mathrm{E}\big[\boldsymbol{\Psi}_t^T \boldsymbol{\Phi}_t|\pi_0\big]\rho + \frac{1}{n} \sum_{t=1}^{t_f} \mathrm{E}\big[\boldsymbol{\Psi}_t^T \mathbf{f}_t|\pi_0\big] - \frac{1}{n} \sum_{t=1}^{t_f} \mathrm{E}\big[\boldsymbol{\Psi}_t^T \boldsymbol{\Phi}_t|\pi_0\big]\rho \\ &= \frac{1}{n} \sum_{t=1}^{t_f} \mathrm{E}\big[\boldsymbol{\Psi}_t^T \mathbf{f}_t|\pi_0\big]. \end{aligned}$$

Optimal RSM

Equation 4.13 takes an average over many single trial gradient estimates. Since each of the n policy trials is conditionally independent of the others (conditioned on π_0), the optimal response surface model may be approximated by minimizing the trace of the covariance matrix of these single trial estimates. This optimal model can be estimated by taking the derivative of the trace with respect to ρ. By setting this value to zero and solving for ρ we get an estimate of the optimal response surface model ρ^* around the current policy. The variance of the estimator is approximated by using the sample variance:

$$\mathrm{var}\big[g_\rho(\boldsymbol{\psi}, \boldsymbol{\phi}, \mathbf{f}, G, \rho)|\pi_0\big] \approx \frac{1}{n^2} \sum_{i=1}^{n} \Big(\sum_{t=1}^{t_f} \psi_t^{(i)} \big(f_t^{(i)} - \phi_t^{(i)^T} \rho\big) \Big) \Big(\sum_{t=1}^{t_f} \big(f_t^{(i)} - \phi_t^{(i)^T} \rho\big) \psi_t^{(i)^T} \Big) + k,$$

where k is a term whose derivative does not depend on ρ. We take the derivative of the trace and solve for the optimal parameter ρ^*.

$$\begin{aligned}
\operatorname{tr}(\operatorname{var}\big[g_\rho(\boldsymbol{\psi},\boldsymbol{\phi},\mathbf{f},G,\rho)|\pi_0\big]) \approx\ & \operatorname{tr}\Big(\frac{1}{n^2}\sum_{i=1}^{n}\sum_{t=1}^{t_f}\sum_{u=1}^{t_f}\psi_t^{(i)}\big(f_t^{(i)}-{\phi_t^{(i)}}^T\rho\big) \\
& \big(f_u^{(i)}-{\phi_u^{(i)}}^T\rho\big){\psi_u^{(i)}}^T\Big)+\operatorname{tr}(k) \\
d\operatorname{tr}(\operatorname{var}\big[g_\rho(\boldsymbol{\psi},\boldsymbol{\phi},\mathbf{f},G,\rho)|\pi_0\big]) \approx\ & -\frac{2}{n^2}\sum_{i=1}^{n}\sum_{t=1}^{t_f}\sum_{u=1}^{t_f}{\psi_t^{(i)}}^T\psi_u^{(i)}\big(f_t^{(i)}-{\phi_t^{(i)}}^T\rho\big){\phi_u^{(i)}}^T d\rho \\
0 \approx\ & \sum_{i=1}^{n}\sum_{t=1}^{t_f}\sum_{u=1}^{t_f}\Big(f_t^{(i)}{\psi_t^{(i)}}^T\psi_u^{(i)}{\phi_u^{(i)}}^T-\rho^T\phi_t^{(i)}{\psi_t^{(i)}}^T\psi_u^{(i)}{\phi_u^{(i)}}^T\Big) \\
\rho^* \approx\ & \Big(\sum_{i=1}^{n}\sum_{t=1}^{t_f}\phi_t^{(i)}{\psi_t^{(i)}}^T\sum_{u=1}^{t_f}\psi_u^{(i)}{\phi_u^{(i)}}^T\Big)^{-1}\sum_{i=1}^{n}\sum_{t=1}^{t_f}\phi_t^{(i)}{\psi_t^{(i)}}^T\sum_{u=1}^{t_f}\psi_u^{(i)}f_u^{(i)}.
\end{aligned}$$

We may get better results if we assume that the individual terms used to compute the summation in Equation 4.13 are uncorrelated. The eligibility vectors from different time steps are uncorrelated but the returns and feature vectors may be highly correlated. Nevertheless, by ignoring this correlation we get many more data points which may be used for approximating ρ^*. We can approximate the variance as follows:

$$\operatorname{var}\big[g_\rho(\boldsymbol{\psi},\boldsymbol{\phi},\mathbf{f},G,\rho)|\pi_0\big] \approx \frac{1}{n^2}\sum_{i=1}^{n}\sum_{t=1}^{t_f}\psi_t^{(i)}\big(f_t^{(i)}-{\phi_t^{(i)}}^T\rho\big)^2{\psi_t^{(i)}}^T+k,$$

where k is a term whose derivative does not depend on ρ. We take the derivative of the

Algorithm 4.3: Time-variant RSM gradient estimator

Input: $\{\Psi_t\}_{t\in\{1,\dots,t_f\}}$, the eligibility vectors; $\{f_t\}_{t\in\{1,\dots,t_f\}}$, the scores; $\{\Sigma_{i,t}\}_{i\in\{1,\dots,n\},t\in\{1,\dots,t_f\}}$, covariance matrices where $\Sigma_{i,t} = \mathrm{var}\big[\psi(\mathbf{a}_t^{(i)}|o_t^{(i)},\pi_0)|\pi_0\big]$; `features`, a user-defined function that computes $\{\Phi_t\}_{t\in\{1,\dots,t_f\}}$ and G; Λ_ρ, a regularization parameter.

Output: ∇, an estimate of the gradient

$n \leftarrow$ number of rows in Ψ_1

for $t = 1$ *to* t_f **do**

$W_t \leftarrow \frac{1}{n}\mathrm{ddiag}(\Psi_t\Psi_t^T)$ `// ddiag clears non-diagonal entries`

end

$[\{\Phi\}_{t\in\{1,\dots,t_f\}}, G] \leftarrow \texttt{features}(\{\Psi_t\}_{t\in\{1,\dots,t_f\}}, \{\Sigma_{i,t}\}_{i\in\{1,\dots,n\},t\in\{1,\dots,t_f\}})$

$$\rho = \Big(\sum_{t=1}^{t_f}\Phi_t^T W_t\Phi_t + \Lambda_\rho\Big)^{-1}\sum_{t=1}^{t_f}\Phi_t^T W_t f_t$$

$$\nabla = G\rho + \tfrac{1}{n}\sum_{t=1}^{t_f}\Psi_t(f_t - \Phi_t\rho)$$

return ∇

trace and solve for the optimal parameter ρ^*.

$$\mathrm{tr}(\mathrm{var}\big[g_\rho(\boldsymbol{\psi},\boldsymbol{\phi},\mathbf{f},G,\rho)|\pi_0\big]) \approx \mathrm{tr}\Big(\frac{1}{n^2}\sum_{i=1}^{n}\sum_{t=1}^{t_f}\psi_t^{(i)^T}\psi_t^{(i)}(f_t^{(i)} - \phi_t^{(i)^T}\rho)^2\Big) + \mathrm{tr}(k)$$

$$d\,\mathrm{tr}(\mathrm{var}\big[g_\rho(\boldsymbol{\psi},\boldsymbol{\phi},\mathbf{f},G,\rho)|\pi_0\big]) \approx -\frac{2}{n^2}\sum_{i=1}^{n}\sum_{t=1}^{t_f}\psi_t^{(i)^T}\psi_t^{(i)}\big(f_t^{(i)} - \phi_t^{(i)^T}\rho\big)\phi_t^{(i)^T}d\rho$$

$$0 \approx \sum_{i=1}^{n}\sum_{t=1}^{t_f} f_t^{(i)}\psi_t^{(i)^T}\psi_t^{(i)}\phi_t^{(i)^T} - \rho^T\sum_{i=1}^{n}\sum_{t=1}^{t_f}\phi_t^{(i)}\psi_t^{(i)^T}\psi_t^{(i)}\phi_t^{(i)^T}$$

$$\rho^* \approx \big(\sum_{i=1}^{n}\sum_{t=1}^{t_f}\phi_t^{(i)}\psi_t^{(i)^T}\psi_t^{(i)}\phi_t^{(i)^T}\big)^{-1}\sum_{i=1}^{n}\sum_{t=1}^{t_f}\phi_t^{(i)}\psi_t^{(i)^T}\psi_t^{(i)}f_t^{(i)}$$

This is equivalent to performing a weighted linear regression where data from the ith policy trial (at time t) is weighted by $\psi_t^{(i)^T}\psi_t^{(i)}$. We may rewrite the approximation of ρ^* as follows:

$$\rho^* \approx \Big(\sum_{t=1}^{t_f}\Phi_t^T\mathrm{ddiag}(\Psi_t\Psi_t^T)\Phi_t\Big)^{-1}\sum_{t=1}^{t_f}\Phi_t^T\mathrm{ddiag}(\Psi_t\Psi_t^T)f_t. \tag{4.14}$$

The time-variant RSM gradient estimator algorithm

Algorithm 4.3 shows an improved gradient estimation algorithm that uses an RSM. This algorithm is suitable for problems with multiple time-steps and it requires the following

inputs: $\{\Psi_t\}_{t\in\{1,\ldots,t_f\}}$, the eligibility vectors; $\{f_t\}_{t\in\{1,\ldots,t_f\}}$, the returns for each time step; $\{\Sigma_{i,t}\}_{i\in\{1,\ldots,n\},t\in\{1,\ldots,t_f\}}$, covariance matrices where $\Sigma_{i,t} = \mathrm{var}\big[\psi(\mathbf{a}_t^{(i)}|o_t^{(i)},\pi_0)|\pi_0\big]$; `features`, a user-defined function that computes $\{\Phi\}_{t\in\{1,\ldots,t_f\}}$ and G; and Λ_ρ, a regularization parameter.

4.2.4 Time-variant natural RSM gradients

The natural gradient estimators pre-multiply the regular gradient estimators by the inverse of the Fisher information matrix. The following estimator is biased but the bias is guaranteed to result in an estimate that is less than $\pi/2$ radians (in expectation) away from the true gradient:

$$\begin{aligned}\widetilde{g}_\rho(\psi,\phi,\mathrm{f},\widetilde{G},\rho) &= \widetilde{G}\rho + \frac{\mathcal{F}^{-1}}{n}\sum_{t=1}^{t_f}\Psi_t^T(f_t-\Phi_t\rho)\\ \widetilde{G} &:= \frac{\mathcal{F}^{-1}}{n}\sum_{t=1}^{t_f}\mathrm{E}\big[\mathbf{\Psi}_t^T\mathbf{\Phi}_t|\pi_0\big].\end{aligned} \tag{4.15}$$

The natural gradient of the response surface is $\widetilde{G}\rho$, which must be added to the estimator so that it remains unbiased. We can verify that this is an unbiased estimator by taking its expected value and observing that it is equivalent to the natural REINFORCE estimator which is known to be unbiased.

$$\begin{aligned}\mathrm{E}\big[\widetilde{g}_\rho(\boldsymbol{\psi},\boldsymbol{\phi},\mathbf{f},\widetilde{G},\rho)|\pi_0\big] = {}& \frac{\mathcal{F}^{-1}}{n}\sum_{t=1}^{t_f}\mathrm{E}\big[\mathbf{\Psi}_t^T\mathbf{\Phi}_t|\pi_0\big]\rho + \frac{\mathcal{F}^{-1}}{n}\sum_{t=1}^{t_f}\mathrm{E}\big[\mathbf{\Psi}_t^T\mathbf{f}_t|\pi_0\big] -\\ & \frac{\mathcal{F}^{-1}}{n}\sum_{t=1}^{t_f}\mathrm{E}\big[\mathbf{\Psi}_t^T\mathbf{\Phi}_t|\pi_0\big]\rho\\ = {}& \frac{\mathcal{F}^{-1}}{n}\sum_{t=1}^{t_f}\mathrm{E}\big[\mathbf{\Psi}_t^T\mathbf{f}_t|\pi_0\big].\end{aligned}$$

Optimal RSM

Equation 4.15 takes an average over many single trial gradient estimates and pre-multiplies the result by an estimate of $\mathcal{F}^{-1}$. The variance of the estimator is approximated by using the sample variance:

$$\begin{aligned}\operatorname{var}\big[\widetilde{g}_\rho(\boldsymbol{\psi},\boldsymbol{\phi},\mathbf{f},\widetilde{G},\rho)|\pi_0\big] \approx \frac{1}{n^2}\mathcal{F}^{-1}\sum_{i=1}^{n}\Big(\sum_{t=1}^{t_f}\psi_t^{(i)}\big(f_t^{(i)}-{\phi_t^{(i)}}^T\rho\big)\Big)\\ \Big(\sum_{t=1}^{t_f}\big(f_t^{(i)}-{\phi_t^{(i)}}^T\rho\big){\psi_t^{(i)}}^T\Big)\mathcal{F}^{-1}+k,\end{aligned}$$

where k is a term whose derivative does not depend on ρ. We take the derivative of the trace and solve for the optimal parameter ρ^*.

$$\begin{aligned}\operatorname{tr}(\operatorname{var}\big[\widetilde{g}_\rho(\boldsymbol{\psi},\boldsymbol{\phi},\mathbf{f},\widetilde{G},\rho)|\pi_0\big]) \approx\ & \operatorname{tr}\Big(\frac{1}{n^2}\sum_{i=1}^{n}\sum_{t=1}^{t_f}\sum_{u=1}^{t_f}\mathcal{F}^{-1}\psi_t^{(i)}\big(f_t^{(i)}-{\phi_t^{(i)}}^T\rho\big)\\ &\big(f_u^{(i)}-{\phi_u^{(i)}}^T\rho\big){\psi_u^{(i)}}^T\mathcal{F}^{-1}\Big)+\operatorname{tr}(k)\\ d\operatorname{tr}(\operatorname{var}\big[\widetilde{g}_\rho(\boldsymbol{\psi},\boldsymbol{\phi},\mathbf{f},\widetilde{G},\rho)|\pi_0\big]) \approx\ & -\frac{2}{n^2}\sum_{i=1}^{n}\sum_{t=1}^{t_f}\sum_{u=1}^{t_f}{\psi_t^{(i)}}^T\mathcal{F}^{-2}\psi_u^{(i)}\big(f_t^{(i)}-{\phi_t^{(i)}}^T\rho\big){\phi_u^{(i)}}^T d\rho\\ 0 \approx\ & \sum_{i=1}^{n}\sum_{t=1}^{t_f}\sum_{u=1}^{t_f}\Big(f_t^{(i)}{\psi_t^{(i)}}^T\mathcal{F}^{-2}\psi_u^{(i)}{\phi_u^{(i)}}^T-\rho^T\phi_t^{(i)}{\psi_t^{(i)}}^T\mathcal{F}^{-2}\psi_u^{(i)}{\phi_u^{(i)}}^T\Big)\\ \rho^* \approx\ & \Big(\sum_{i=1}^{n}\sum_{t=1}^{t_f}\phi_t^{(i)}{\psi_t^{(i)}}^T\mathcal{F}^{-2}\sum_{u=1}^{t_f}\psi_u^{(i)}{\phi_u^{(i)}}^T\Big)^{-1}\sum_{i=1}^{n}\sum_{t=1}^{t_f}\phi_t^{(i)}{\psi_t^{(i)}}^T\mathcal{F}^{-2}\sum_{u=1}^{t_f}\psi_u^{(i)}f_u^{(i)}.\end{aligned}$$

As discussed in Section 4.2.3, we may get better results if we assume that the individual terms used in the summation term of Equation 4.15 are uncorrelated. We can approximate the variance as follows:

$$\operatorname{var}\big[\widetilde{g}_\rho(\boldsymbol{\psi},\boldsymbol{\phi},\mathbf{f},\widetilde{G},\rho)|\pi_0\big] \approx \frac{1}{n^2}\mathcal{F}^{-1}\sum_{i=1}^{n}\sum_{t=1}^{t_f}\psi_t^{(i)}\big(f_t^{(i)}-{\phi_t^{(i)}}^T\rho\big)^2{\psi_t^{(i)}}^T\mathcal{F}^{-1}+k,$$

where k is a term whose derivative does not depend on ρ. We take the derivative of the

trace and solve for the optimal parameter ρ^*.

$$\mathrm{tr}(\mathrm{var}\big[\widetilde{g}_\rho(\boldsymbol{\psi},\boldsymbol{\phi},\mathbf{f},\widetilde{G},\rho)|\pi_0\big]) \approx \mathrm{tr}\Big(\frac{1}{n^2}\sum_{i=1}^{n}\sum_{t=1}^{t_f}{\psi_t^{(i)}}^T\mathcal{F}^{-2}\psi_t^{(i)}(f_t^{(i)}-{\phi_t^{(i)}}^T\rho)^2\Big)+\mathrm{tr}(k)$$

$$d\,\mathrm{tr}(\mathrm{var}\big[\widetilde{g}_\rho(\boldsymbol{\psi},\boldsymbol{\phi},\mathbf{f},\widetilde{G},\rho)|\pi_0\big]) \approx -\frac{2}{n^2}\sum_{i=1}^{n}\sum_{t=1}^{t_f}{\psi_t^{(i)}}^T\mathcal{F}^{-2}\psi_t^{(i)}\big(f_t^{(i)}-{\phi_t^{(i)}}^T\rho\big){\phi_t^{(i)}}^T d\rho$$

$$0 \approx \sum_{i=1}^{n}\sum_{t=1}^{t_f} f_t^{(i)}{\psi_t^{(i)}}^T\mathcal{F}^{-2}\psi_t^{(i)}{\phi_t^{(i)}}^T - \rho^T\sum_{i=1}^{n}\sum_{t=1}^{t_f}\phi_t^{(i)}{\psi_t^{(i)}}^T\mathcal{F}^{-2}\psi_t^{(i)}{\phi_t^{(i)}}^T$$

$$\rho^* \approx \big(\sum_{i=1}^{n}\sum_{t=1}^{t_f}\phi_t^{(i)}{\psi_t^{(i)}}^T\mathcal{F}^{-2}\psi_t^{(i)}{\phi_t^{(i)}}^T\big)^{-1}\sum_{i=1}^{n}\sum_{t=1}^{t_f}\phi_t^{(i)}{\psi_t^{(i)}}^T\mathcal{F}^{-2}\psi_t^{(i)}f_t^{(i)}$$

This is equivalent to performing a weighted linear regression where data from the ith policy trial (at time t) is weighted by ${\psi_t^{(i)}}^T\mathcal{F}^{-2}\psi_t^{(i)}$. We may rewrite the approximation of ρ^* as follows:

$$\rho^* \approx \Big(\sum_{t=1}^{t_f}\Phi_t^T\mathrm{ddiag}(\Psi_t\mathcal{F}^{-2}\Psi_t^T)\Phi_t\Big)^{-1}\sum_{t=1}^{t_f}\Phi_t^T\mathrm{ddiag}(\Psi_t\mathcal{F}^{-2}\Psi_t^T)f_t, \tag{4.16}$$

if we have an analytical solution for $\mathcal{F}^{-2}$ or use the empirical estimate:

$$\rho^* \approx \Big(\sum_{t=1}^{t_f}\Phi_t^T\mathrm{ddiag}(\Psi_t(\sum_{t=1}^{t_f}\Psi_t^T\Psi_t)^{-2}\Psi_t^T)\Phi_t\Big)^{-1}\sum_{t=1}^{t_f}\Phi_t^T\mathrm{ddiag}(\Psi_t(\sum_{t=1}^{t_f}\Psi_t^T\Psi_t)^{-2}\Psi_t^T)f_t. \tag{4.17}$$

The time-variant natural gradient estimator algorithm

Algorithm 4.4 shows an improved natural gradient estimation algorithm that uses an RSM to reduce the variance. This algorithm is suitable for problems with multiple time-steps and it requires the following inputs: $\{\Psi_t\}_{t\in\{1,\ldots,t_f\}}$, the eligibility vectors; $\{f_t\}_{t\in\{1,\ldots,t_f\}}$, the returns for each time step; $\{\Sigma_{i,t}\}_{i\in\{1,\ldots,n\},t\in\{1,\ldots,t_f\}}$, covariance matrices where $\Sigma_{i,t}=\mathrm{var}\big[\psi(\mathbf{a}_t^{(i)}|o_t^{(i)},\pi_0)|\pi_0\big]$; and `features`, a user-defined function that computes $\{\Phi_t\}_{t\in\{1,\ldots,t_f\}}$ and G; and Λ_ρ, a regularization parameter.

Algorithm 4.4: Time-variant natural gradient estimator

Input: $\{\Psi_t\}_{t\in\{1,\ldots,t_f\}}$, the eligibility vectors; $\{f_t\}_{t\in\{1,\ldots,t_f\}}$, the scores; $\{\Sigma_{i,t}\}_{i\in\{1,\ldots,n\},t\in\{1,\ldots,t_f\}}$, covariance matrices where $\Sigma_{i,t} = \mathrm{var}\big[\psi(\mathbf{a}_t^{(i)}|o_t^{(i)},\pi_0)|\pi_0\big]$; `features`, a user-defined function that computes $\{\Phi_t\}_{t\in\{1,\ldots,t_f\}}$ and G; Λ_ρ, a regularization parameter.

Output: ∇, an estimate of the gradient

$n \leftarrow$ number of rows in Ψ_1

$\mathcal{F} \leftarrow 0, \mathcal{F}_e \leftarrow 0$

for $t = 1$ *to* t_f **do**

  for $i = 1$ *to* n **do**

    $\mathcal{F} \leftarrow \mathcal{F} + \Sigma_{i,t}/n$

  end

  $\mathcal{F}_e \leftarrow \mathcal{F}_e + {\Psi_t}^T\Psi_t/n$

end

for $t = 1$ *to* t_f **do**

  $W_t \leftarrow \frac{1}{n}\mathrm{ddiag}(\Psi_t\mathcal{F}^{-2}\Psi_t^T)$ `// ddiag clears non-diagonal entries`

end

$[\{\Phi\}_{t\in\{1,\ldots,t_f\}}, G] \leftarrow$ `features`$(\{\Psi_t\}_{t\in\{1,\ldots,t_f\}}, \{\Sigma_{i,t}\}_{i\in\{1,\ldots,n\},t\in\{1,\ldots,t_f\}})$

$$\rho = \Big(\sum_{t=1}^{t_f} \Phi_t^T W_t \Phi_t + \Lambda_\rho\Big)^{-1} \sum_{t=1}^{t_f} \Phi_t^T W_t f_t$$

$$\nabla = \mathcal{F}^{-1}G\rho + \tfrac{1}{n}{\mathcal{F}_e}^{-1}\sum_{t=1}^{t_f}\Psi_t(f_t - \Phi_t\rho)$$

return ∇

4.2.5 Properties of eligibilities

Before we present some examples of features that may be used to construct an RSM, we must discuss two important properties relating to the distribution of ψ. These properties allow us to derive analytical solutions for G, a requirement shared by the RSM gradient estimators. The first well-known property is that $\mathrm{E}\big[\psi(\mathbf{a}_t|\mathbf{o}_t,\pi_0)|\pi_0\big] = 0$ and we

show this by first integrating over the random variable $\mathbf{a}_t$ for a given observation o_t.

$$\begin{aligned}
\mathrm{E}\big[\psi(\mathbf{a}_t|o_t,\pi_0)|o_t,\pi_0\big] &= \int \frac{\nabla_\pi P(a_t|o_t,\pi)\big|_{\pi=\pi_0}}{P(a_t|o_t,\pi_0)} P(a_t|o_t,\pi_0)da_t \\
&= \nabla_\pi \int P(a_t|o_t,\pi)da_t\Big|_{\pi=\pi_0} \\
&= \nabla_\pi 1\Big|_{\pi=\pi_0} \\
&= 0.
\end{aligned}$$

$\mathrm{E}\big[\psi(\mathbf{a}_t|\mathbf{o}_t,\pi_0)|\pi_0\big] = 0$ follows from the law of total expectation:

$$\mathrm{E}\big[\psi(\mathbf{a}_t|\mathbf{o}_t,\pi_0)|\pi_0\big] = \mathrm{E}\big[\mathrm{E}\big[\psi(\mathbf{a}_t|o_t,\pi_0)|\mathbf{o}_t,\pi_0\big]|\pi_0\big] = 0. \tag{4.18}$$

Thus every time an agent chooses an action, the expectation of its corresponding eligibility equals zero; this property is true regardless of an agent's current state, past history, or policy parameterization.

The second property, which naturally follows, is that eligibilities from different time steps are uncorrelated. This is shown by the following equation:

$$\begin{aligned}
\mathrm{E}\big[\psi(\mathbf{a}_t|\mathbf{o}_t,\pi_0)\psi(\mathbf{a}_{t+k}|\mathbf{o}_{t+k},\pi_0)^T|\pi_0\big] &= \mathrm{E}\big[\psi(\mathbf{a}_t|\mathbf{o}_t,\pi_0)\,\mathrm{E}\big[\psi(\mathbf{a}_{t+k}|o_{t+k},\pi_0)^T|\mathbf{o}_{t+k},\pi_0\big]|\pi_0\big] \\
&= \mathrm{E}\big[\psi(\mathbf{a}_t|\mathbf{o}_t,\pi_0)0^T|\pi_0\big] = 0.
\end{aligned}$$

In fact, ψ_{t+k} is uncorrelated with any information an agent obtains before executing the action (e.g., states, observations, and rewards). Note that it is possible that the eligibilities are statistically dependent on each other. For example, an agent's prior actions may cause it to enter some state where π chooses actions according to a different exploration distribution. This change causes the distribution of the corresponding eligibilities to change. Therefore one could infer something about past eligibilities based on the value of the current eligibility.

Useful response surface model functions

These properties allow us to construct RSMs where there is an analytical solution for G, a requirement shared by the RSM gradient estimators. An advantage function RSM may be constructed by using the following feature vector: $\phi_t(h) = [\phi_s(s_t)^T, {\psi_t}^T]^T$, where ϕ_s computes features over the current state. There are two components: $\phi_s(s_t)$ is a feature vector that is similar to an approximate value function (or parameterized baseline) and ψ_t is used to predict the advantage of performing a particular action. We may write G by observing that ψ_t is uncorrelated with the agent's state at time t.

$$G = \sum_{t=1}^{t_f} \mathrm{E}\big[\boldsymbol{\psi}_t[\phi_s(\mathbf{s}_t)^T, \boldsymbol{\psi}_t^T]|\pi_0\big] = \sum_{t=1}^{t_f} \big[0, \mathrm{E}\big[\boldsymbol{\psi}_t {\boldsymbol{\psi}_t}^T|\pi_0\big]\big].$$

Actor-critic techniques often use similar functions for their critics. These algorithms learn an approximation to the value and advantage functions and then substitute their predictions in place of the actual returns. Our algorithms subtract the predicted value from the actual returns and add $G\rho$ to compensate for this change. We also attempt to find a model that explicitly minimizes the variance of the gradient estimator. Greensmith *et al.* also learn value functions that explicitly minimize the error in gradient estimation [2001]. In the fully observable setting, the actor-critic approach may yield better results. Our approach, however, does not require full observability.

We can also use the following feature vector: $\phi_t(h) = [\phi_o(o_t)^T, \psi_t^T]^T$, where ϕ_o computes features over the current observation. Because ψ_t is uncorrelated with the agent's observation at time t, the analytical expression for G is written as follows:

$$G = \sum_{t=1}^{t_f} \mathrm{E}\big[\boldsymbol{\psi}_t[\phi_o(\mathbf{o_t})^T, \boldsymbol{\psi}_t^T]|\pi_0\big] = \sum_{t=1}^{t_f} \big[0, \mathrm{E}\big[\boldsymbol{\psi}_t {\boldsymbol{\psi}_t}^T|\pi_0\big]\big].$$

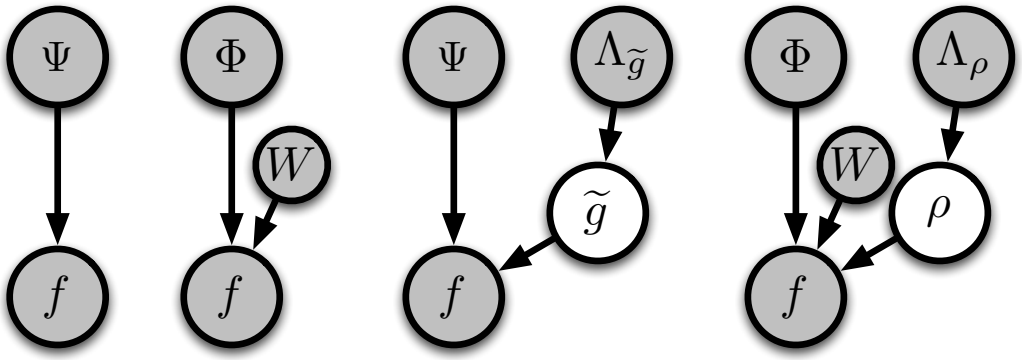
We showed that eligibilities from different time steps are uncorrelated and so we may consider the following RSM: $\phi_t(h) = [\phi_o(o_t)^T, \sum_{u=t}^{t_f} \psi_u{}^T]^T$. An analytical expression for G may be written because eligibilities from different time steps are uncorrelated:

$$
\begin{aligned}
G &= \sum_{t=1}^{t_f} \mathrm{E}\big[\boldsymbol{\psi}_t[\phi_o(\mathbf{o_t})^T, \sum_{u=t}^{t_f} \psi_u{}^T]|\pi_0\big] \\
G &= \sum_{t=1}^{t_f} \mathrm{E}\big[\boldsymbol{\psi}_t[\phi_o(\mathbf{o_t})^T, \boldsymbol{\psi}_t{}^T + \sum_{u=t+1}^{t_f} \boldsymbol{\psi}_u{}^T]|\pi_0\big] \\
&= \sum_{t=1}^{t_f} \big[0, \mathrm{E}\big[\boldsymbol{\psi}_t \boldsymbol{\psi}_t{}^T|\pi_0\big]\big].
\end{aligned}
$$

4.3 Probabilistic inference problem

In this section we show that the natural gradient estimator (when using an empirical estimate of the Fisher information matrix) is mathematically equivalent to performing maximum likelihood estimation of a parameter of a linear Gaussian model. One advantage of this view is that it allows us to extend these algorithms by improving the statistical model. For example we can further reduce the variance, with the addition of some bias, by using an appropriate prior. Despite the bias, we can show that the estimated gradient is guaranteed to be less than $\pi/2$ radians away from the true gradient.

The solution to the optimal response surface model can also be cast as a maximum likelihood estimation problem. This situation corresponds to learning the parameter of a linear Gaussian model where the sample points are weighted. Thus we have a weighted least squares regression problem.



(a) (b) (c) (d)

Figure 4.6: Four Bayesian networks that may be used for gradient estimation. They may be used to (a) estimate the natural gradient, (b) find the optimal RSM, (c) estimate the natural gradient while using a prior, and (d) find the optimal RSM using a prior.

4.3.1 Bayesian network representations

Figure 4.6(a) shows a two node graphical model that illustrates this transformation (in Chapter 5 we will expand this model by adding nodes that encode sensor data). The relationship is governed by a linear Gaussian model ($f = \Psi\widetilde{g} + w$) where the eligibility terms Ψ and corresponding scores f are obtained from the agent and w is zero-mean additive Gaussian noise. The maximum likelihood solution $\widetilde{g}^*$ to the following equations are equivalent to the natural gradient estimator.

$$P(f|\Psi,\pi_0) = (2\pi\sigma^2)^{-n/2}\exp\Big\{-\frac{1}{2\sigma^2}(f-\Psi\widetilde{g})^T(f-\Psi\widetilde{g})\Big\}$$

$$\ell(f|\Psi,\pi_0) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(f-\Psi\widetilde{g})^T(f-\Psi\widetilde{g})$$

$$\widetilde{g}^* = \widetilde{g}(\Psi, f) = (\Psi^T\Psi)^{-1}\Psi^T f$$

Figure 4.6(b) shows a two node graphical model that can be used to learn the optimal response surface model parameters. In this case the parent nodes contain the

features for each policy trial and each trial is weighted by the following weight matrix W:

$$W := \text{ddiag}(\Psi \mathcal{F}^{-2} \Psi^T).$$

The optimal parameters are given by the following weighted least squares equation.

$$\begin{aligned} P(f|\Phi, \pi_0) &= (2\pi\sigma^2)^{-\operatorname{tr}(W)/2} \exp\Big\{-\frac{1}{2\sigma^2}(f - \Phi\rho)^T W (f - \Phi\rho)\Big\} \\ \ell(f|\Phi, \pi_0) &= -\frac{\operatorname{tr}(W)}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(f - \Phi\rho)^T W (f - \Phi\rho) \\ \rho^* &= (\Phi^T W \Phi)^{-1} \Phi^T W f. \end{aligned}$$

Prior distributions can help reduce the uncertainty over the posterior which will reduce the variance of the gradient estimates. We will use a Gaussian distribution for the mean of the gradient, which we assume equals zero, with isotropic information matrix $\Lambda_{\widetilde{g}}$. An estimate of the gradient is obtained by taking the maximum *a posteriori* estimate of $\widetilde{g}$.

$$\begin{aligned} P(\widetilde{g}) &\propto \exp\Big\{-\frac{1}{2\sigma^2}\widetilde{g}^T \Lambda_{\widetilde{g}} \widetilde{g}\Big\} \\ P(f|\Psi, \pi_0) &= (2\pi\sigma^2)^{-n/2} \exp\Big\{-\frac{1}{2\sigma^2}(f - \Psi\widetilde{g})^T (f - \Psi\widetilde{g})\Big\} \\ \ell(f|\Psi, \pi_0) &= k - \frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(f - \Psi\widetilde{g})^T (f - \Psi\widetilde{g}) - \frac{1}{2\sigma^2}\widetilde{g}^T \Lambda_{\widetilde{g}} \widetilde{g} \\ \widetilde{g}^* &= (\Psi^T \Psi + \Lambda_{\widetilde{g}})^{-1} \Psi^T f, \end{aligned}$$

where k is a constant term. Leave one out cross-validation can be used to determine the appropriate prior. Because the matrix is positive definite, the expectation of the estimator is guaranteed to be less than $\pi/2$ radians away from the true gradient.

By placing a prior on the response surface model we get the following estimator:

$$
\begin{aligned}
P(\rho) &\propto \exp\Big\{-\frac{1}{2\sigma^2}\rho^T\Lambda_\rho\rho\Big\} \\
P(f|\Phi,\pi_0) &= (2\pi\sigma^2)^{-\operatorname{tr}(W)/2}\exp\Big\{-\frac{1}{2\sigma^2}(f-\Phi\rho)^T W(f-\Phi\rho)\Big\} \\
\ell(f|\Phi,\pi_0) &= k - \frac{\operatorname{tr}(W)}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(f-\Phi\rho)^T W(f-\Phi\rho) - \frac{1}{2\sigma^2}\rho^T\Lambda_\rho\rho \\
\rho^* &= (\Phi^T W\Phi + \Lambda_\rho)^{-1}\Phi^T W f.
\end{aligned}
$$

4.4 Results

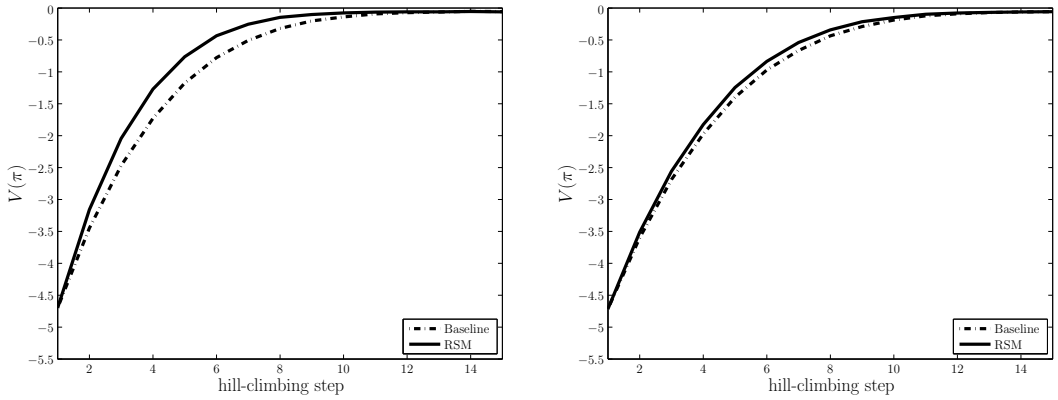
We applied Algorithm 4.1 and Algorithm 4.2 to `dart0`, `dart1`, and `dart2`; these problems are described in Section 3.2. The results are shown in Figure 4.7 and Figure 4.8. Each hill-climbing run lasts 30 steps (some plots only show a portion of this) and at each step we drew 12 policy trials from the current policy; the results were averaged over 500 hill-climbing runs. For these problems we used the following linear response model: $\phi^{(i)} = [1, {\psi^{(i)}}^T]$. We should expect the improvements in learning to be most prominent in the setting where perturbations in f are caused primarily by the randomness used for exploration purposes. This is due to the fact that our response surface models only use features that are functions of the eligibilities. The only stochastic component in `dart0`, is due to the use of a randomized policy. Therefore in this setting we see the biggest gains when using the RSM gradient estimator when compared to the baseline case (Figure 4.7(a)). In `dart1`, where there is actuator noise, we see a slight improvement. When we add noise to the release time, as is the case in `dart2`, the RSM gradient estimator performs substantially worse than the baseline gradient estimator. A similar trend exists for the natural versions of the gradient estimators except that the natural RSM gradient estimator slightly outperforms

the natural baseline gradient estimator for `dart2`. However, both of these perform worse than their standard counterparts. In Chapter 5 we will be able to improve the performance by incorporating the sensor data.

We applied Algorithm 4.1 and Algorithm 4.2 to `quadruped0` and `quadruped2`; and we applied Algorithm 4.3 and Algorithm 4.4 to `quadruped1` and `quadruped3`. The results are shown in Figure 4.9 and Figure 4.10. Each hill-climbing run lasts 20 steps and the results were averaged over 100 hill-climbing runs. In `quadruped0` and `quadruped2` we drew 30 policy trials at each step and in `quadruped1` and `quadruped3` we drew 20 policy trials. The RSM gradient estimators improve the learning performance in `quadruped0` and `quadruped1`; there are slight improvements for `quadruped2` and `quadruped3`. Since the only stochastic component present in `quadruped0` and `quadruped1` is the use of a randomized policy, we should expect greater improvements in this setting. A similar trend exists for the natural versions of the gradient estimators. For `quadruped0` and `quadruped2`, we used the following linear response model: $\phi^{(i)} = [1, \psi^{(i)^T}]$. In `quadruped1` and `quadruped3`, the agent selects a new action after every 0.3 seconds. We used the following response surface model: $\phi_t^{(i)} = [e_t, \sum_{u=t}^{t_f} \psi_u^{(i)^T}]$, where e_i is a column-vector of zeros with the ith entry set to 1. For comparison purposes, the baseline gradient estimator uses a baseline parameterized by time, i.e., we use a different baseline value for each time step. This is equivalent to using $\phi_t^{(i)} = e_t$ as the response surface model.

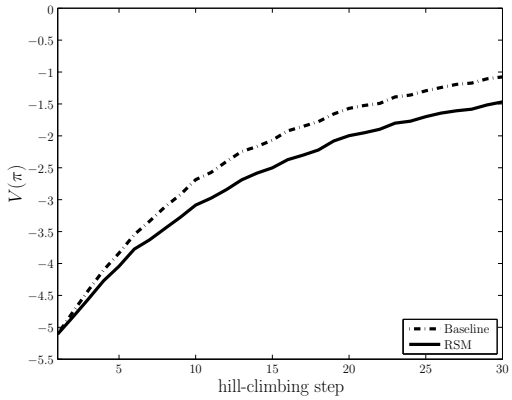
4.5 Discussion

This chapter presented techniques to improve gradient estimation. These techniques use linear response surface models to predict the expected scores of policy trials. Given a parameterized family of functions, we showed how to estimate the optimal RSM parameter, reducing the variance of gradient estimation. We developed several algorithms (including natural versions) that exploit these models and showed that they produce gains in the learning performance for the cannon, dart throwing, and quadruped problems presented in Chapter 3. Finally, we showed how the maximum likelihood estimation of a linear-Gaussian model is mathematically equivalent to the natural gradient estimator. In the next chapter, we will extend these techniques by incorporating an agent's sensor data.



(a)

(b)



(c)

Figure 4.7: (a,b,c) The learning curve performances of the baseline and RSM gradient estimators for `dart0`, `dart1`, and `dart2` respectively.

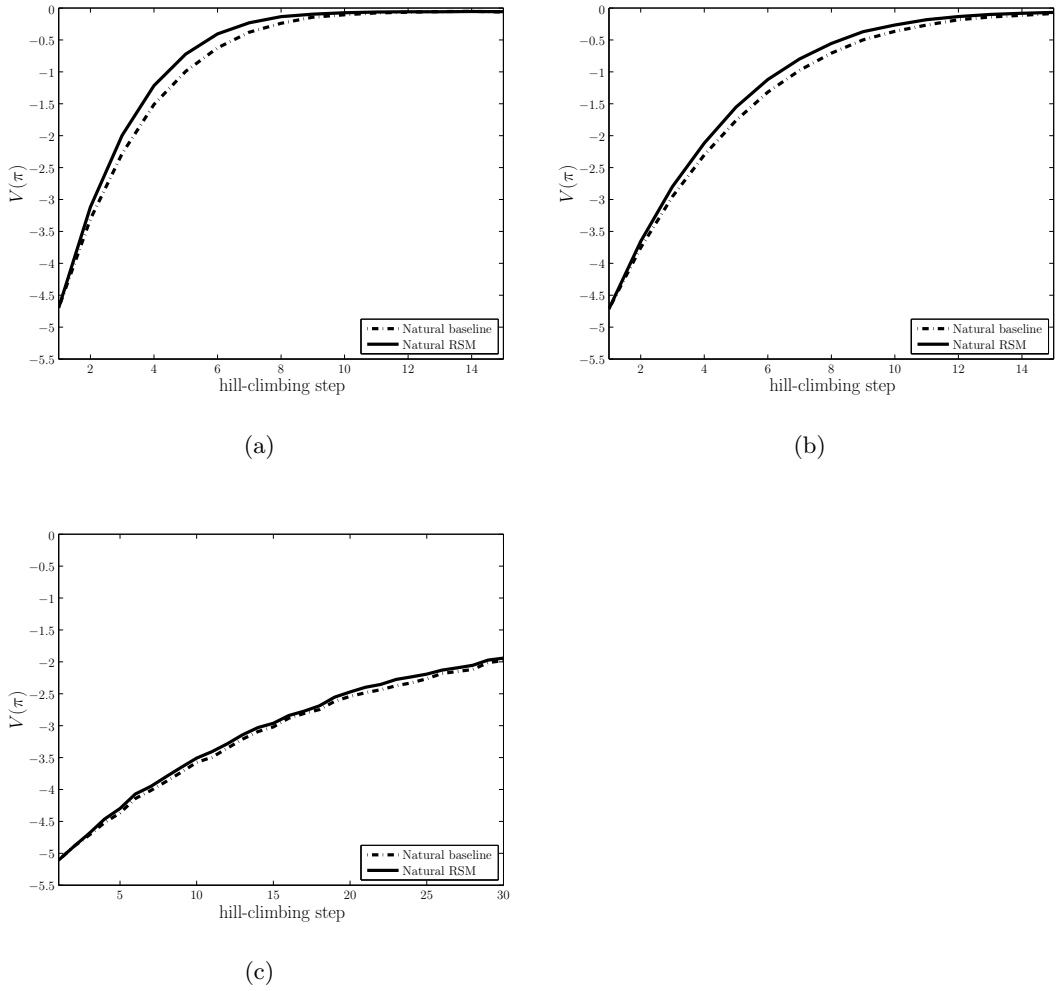


Figure 4.8: (a,b,c) The performances of the natural baseline and natural RSM gradient estimators for `dart0`, `dart1`, and `dart2` respectively.

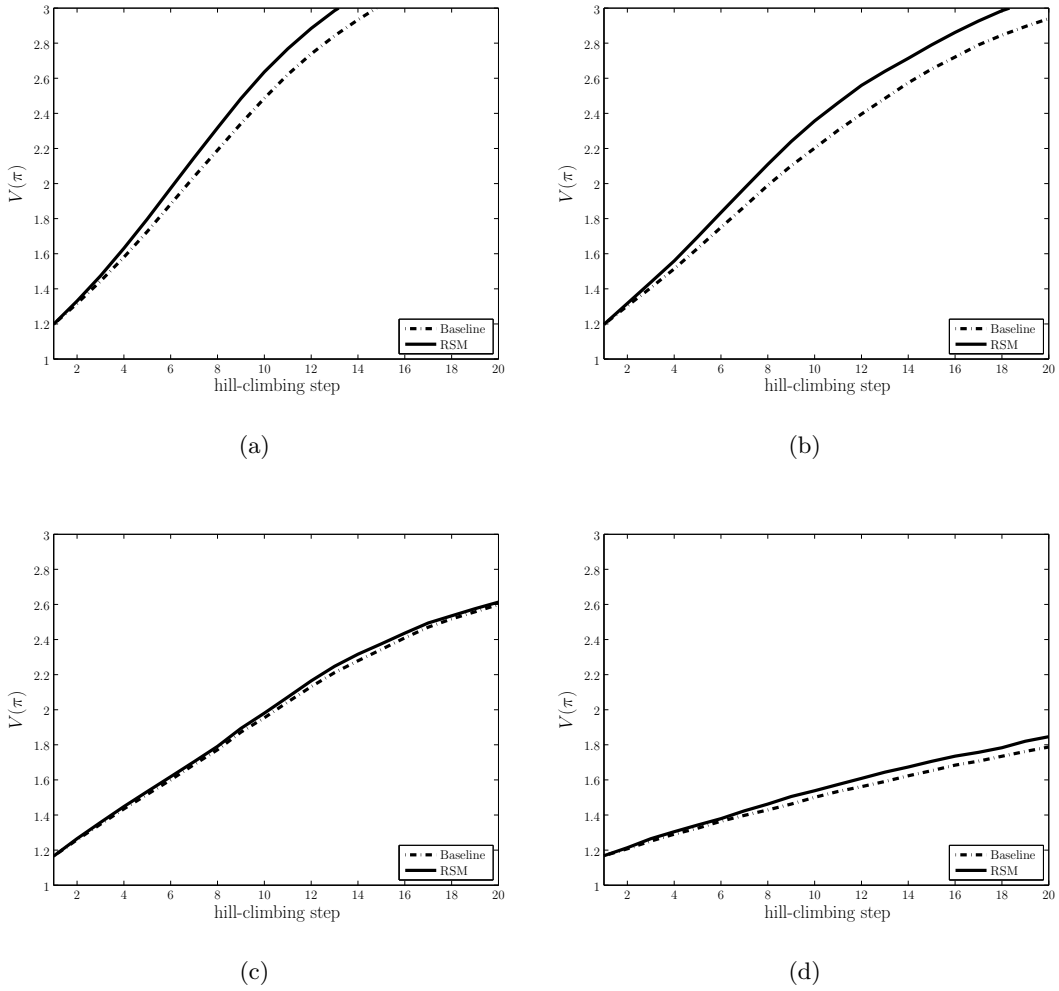


Figure 4.9: (a,b,c,d) The learning curve performances of the baseline and RSM gradient estimators for `quadruped0`, `quadruped1`, `quadruped2`, and `quadruped3` respectively.

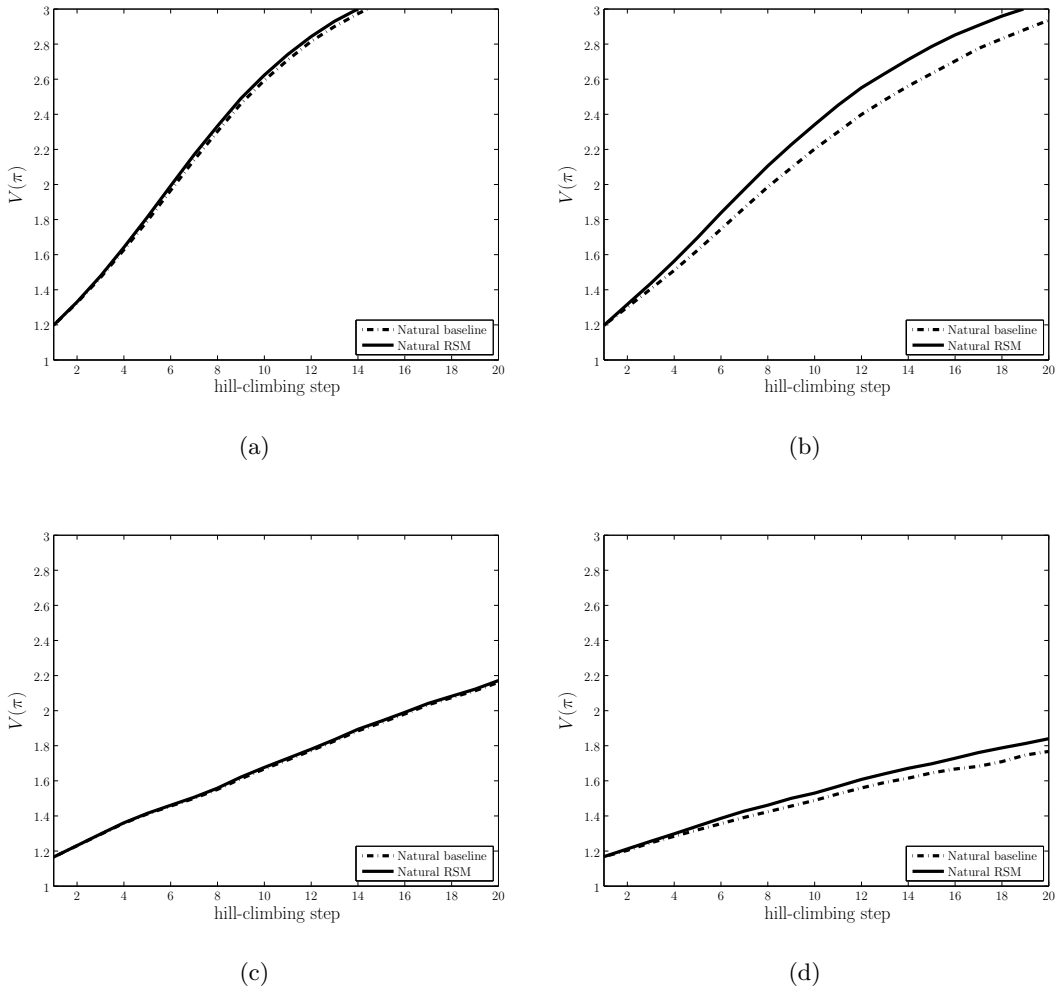


Figure 4.10: (a,b,c,d) The learning curve performances of the natural baseline and natural RSM gradient estimators for `quadruped0`, `quadruped1`, `quadruped2`, and `quadruped3` respectively.

Chapter 5

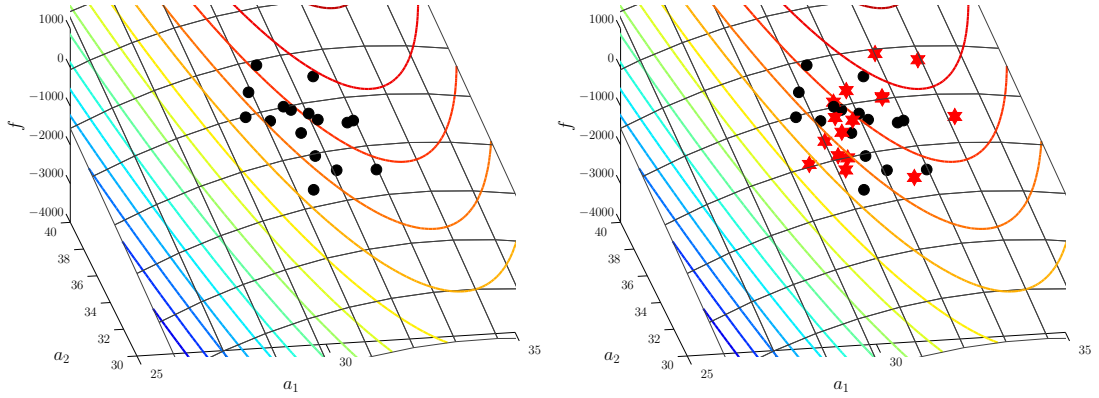
Improving gradient estimation by incorporating sensor data

This chapter discusses how we can reduce the variance of a gradient estimator by incorporating sensor data. Motor control problems typically deal with a large amount of raw sensor data for each policy trial (e.g., the sensed positions of each controllable joint at each time step and the pressure felt by each foot during locomotion). This data is often useful because it helps to explain away noise-induced perturbations in the scores of each policy trial. The sensory data must possess certain properties if our approach is to be of any value, and we use these properties to design sensor encodings that are suitable for motor control problems. Finally, we briefly discuss how to combine these ideas with the response surface model technique presented in Chapter 4.

5.1 Motivation

In the previous chapter we improved the performance of gradient estimation by learning a response surface model. This technique requires an analytical solution to the gradient, with respect to π, of the response surface itself (i.e., the agent must be able to evaluate $G := \mathrm{E}\big[\psi(\mathbf{h}|\pi_0)\boldsymbol{\phi}(\mathbf{h})^T|\pi_0\big]$). As a result, an agent may only use certain feature vectors if it wishes to have an unbiased estimate of the gradient. Since an agent chooses the randomness used for exploration purposes, features that are functions of the eligibility terms may often be used. In cases where the agent knows the sensor model and transition function, the agent may be able to compute G for features that contain sensor data. The dart thrower's release time is an example of a sensor for which we can compute G, because the distribution of the release-time noise is independent of the agent's actions. In general we do not know these relationships, but we may still get better performance if we, nevertheless, incorporate sensor data into the gradient estimation task. In this chapter we present algorithms that reduce the variance of gradient estimation by incorporating an agent's sensor data, even in cases where the agent does not know the sensor model or transition function.

Figure 5.1(b) helps illustrate the importance of using sensor data. It displays sixteen policy trials from the cannon problem with their corresponding sensors ($\pi + s$) superimposed. The contours represent the value of performing an action. Each trial was drawn from a nominal policy π_0 that selects desired actions by adding zero-mean Gaussian noise to π_0. Actuator noise perturbs the desired actions to give the actual controls which are hidden to the agent. Sensors accurately measure these perturbations allowing the agent to infer something about the actual controls. Suppose that we know, for this particular



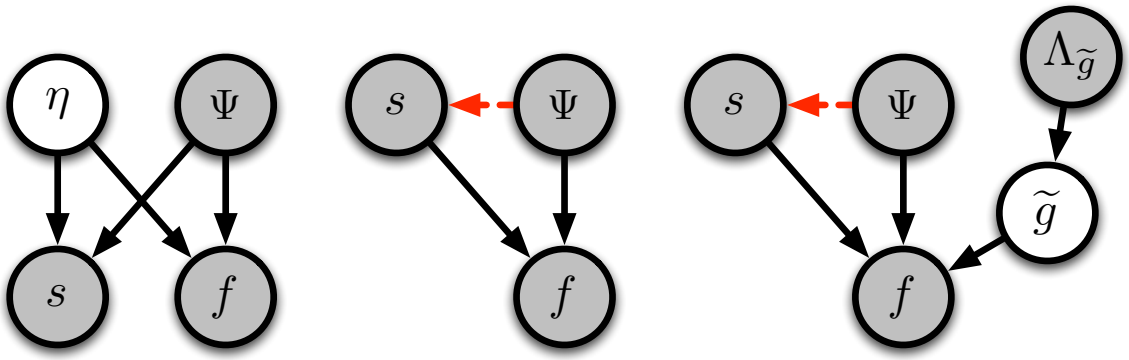
(a) (b)

Figure 5.1: (a) Scores from several policy trials drawn from a toy problem. (b) Scores from several policy trials with the sensors $(\pi + s)$ superimposed.

problem, that the sensor data provides an unbiased measurement of the difference between the actual control u and the desired action a. We may therefore want to subtract the sensors from the desired actions to determine the actual controls. In the next section we explain how this knowledge can be used to partially explain away the noise.

5.2 Probabilistic inference problem

In this section we show, under certain assumptions, how to get an unbiased estimator with reduced variance by exploiting an agent's sensor data. This requires an agent to learn the relationship between its sensors and the noise-induced perturbations in f. Our approach will be to consider a Bayesian network that captures the relationships between Ψ, f, and the sensor data. By using an appropriate encoding of the sensor data, we will be able to use maximum likelihood estimation to find a good estimate of the natural gradient.



(a) (b) (c)

Figure 5.2: (a) Bayesian network that contains latent variable η, which represents all of the noise experienced by an agent during a single policy trial, and sensor node s. (b) Network after we eliminate η and (c) after we add a prior.

5.2.1 Bayesian network representation

In Figure 5.2(a) we proceed by adding a latent variable η and sensor node s to the Bayesian network that was shown in Figure 4.6(a). The idea is that the latent variable represents all of the noise experienced by an agent during each policy trial. For example, it may represent the actuator noise, bumpiness of the ground, or the release-time noise of a throwing agent. Although we never know its true value, we do know that it often influences both the score and the observed sensor data. To help illustrate this let us consider a scenario in the cannon problem. An agent fires several shots and most of them land near the target. However during one of these shots, the cannon operator used too much gun powder, causing the shot to sail way past its target. Fortunately, the observation o_v, a noisy measurement of the perturbation between the intended velocity a_v and the actual velocity, detects the error. In this case, the unobserved human error influenced both the score (a large squared error) and the sensor data. The agent can therefore explain the cause of the miss and ignore

this outlier shot. Otherwise, it might conclude that it should quickly adjust it's policy parameters to avoid repeating this situation. Notice, though, that this decision requires knowledge of the relationship between the sensor data and the score.

If we eliminate the latent variable then we get a graph absent of any conditional independicies; let us consider the graph shown in Figure 5.2(b). Without any knowledge of the relationship between the distribution of Ψ and the sensor information, the additional sensor information will not help when estimating the gradient. While it may improve our ability to predict f, we will also need to learn the relationship between Ψ and S in order to predict the gradient. Our approach will be to assume that Ψ is independent of the sensor information. In situations where this does not hold we get a biased estimator, but we attempt to minimize the bias by penalizing directions in sensor space that are highly correlated with Ψ. The goal is to find an encoding for which Ψ and S are nearly independent and for which knowledge of S helps predict f.

5.2.2 Variance analysis

We proceed by comparing the variance of a gradient estimator that incorporates sensor data to an estimator that ignores it. By considering locally linear scoring functions, we may write expressions for the two variances. First we will assume that S is uncorrelated with Ψ (in Section 5.2.2 we give expressions for the bias and variance of a gradient estimator in the correlated setting). A biased estimator that incorporates sensor data may still outperform one that ignores it as long as the bias remains small.

Let $S = [s^{(1)}, \ldots, s^{(n)}]^T$ be a matrix whose rows contain the sensor values for a single hill-climbing step. The following analysis assumes that the sensors are distributed

from a zero-mean Gaussian with covariance matrix Σ_s. Let $w = [w_1, \ldots, w_n]$ be a column vector of zero-mean noise with variance σ^2. The score function can then be written as $f = \Psi a_\psi + S a_s + 1_n b + w$.

Ignore sensor data

From the point of view of a gradient estimator that ignores the sensor data, additional noise will appear to be added to the scores that can not be explained by perturbations in the sensor data. Let v represent this noise where each element is given by the equation $v = S a_s + w$. The variance of each entry in v is given by the expression ${a_s}^T \Sigma_s a_s + \sigma^2$. The score function can be rewritten as $f = \Psi a_\psi + 1_n b + v$. In Section 4.3.1 we learned the linear relationship between Ψ and f by performing maximum likelihood estimation.

$$\widetilde{g}(\Psi, f) = (\Psi^T \Psi)^{-1} \Psi^T f.$$

The variance may be found by substituting the variance for v into Equation 4.5:

$$\mathrm{var}\left[\widetilde{g}(\mathbf{\Psi}, \mathbf{f}) | \pi_0\right] = \frac{(\Sigma^{-1})(b^2 + {a_s}^T \Sigma_s a_s + \sigma^2)}{n - d - 1} \tag{5.1}$$

Include sensor data

A linear model that predicts the score as a function of both Ψ and S will have the noise on the output partially explained by the sensor data. Figure 5.2(b) contains a linear Gaussian relationship where f depends on both S and Ψ. If S is independent of Ψ, an unbiased estimate of the natural gradient can be found by maximizing the likelihood of

the parameters:

$$
\begin{aligned}
P(f|\Psi,S,\pi_0) &\propto (2\pi\sigma^2)^{-n/2}\exp\Big\{-\frac{1}{2\sigma^2}(f-\Psi\widetilde{g}-S\widetilde{h})^T(f-\Psi\widetilde{g}-S\widetilde{h})\Big\} \\
\ell(f|\Psi,S,\pi_0) &= -\frac{n}{2}\log(2\pi\sigma^2)-\frac{1}{2\sigma^2}(f-\Psi\widetilde{g}-S\widetilde{h})^T(f-\Psi\widetilde{g}-S\widetilde{h}) \\
\begin{bmatrix}\widetilde{g}^* \\ \widetilde{h}^*\end{bmatrix} &= \begin{bmatrix}\Psi^T\Psi & \Psi^TS \\ S^T\Psi & S^TS\end{bmatrix}^{-1}\begin{bmatrix}\Psi^Tf \\ S^Tf\end{bmatrix},
\end{aligned}
\tag{5.2}
$$

where $\widetilde{h}^*$ predicts a_s.

The variance of the above estimator is written as

$$
\mathrm{var}\Big[\begin{bmatrix}\widetilde{g}(\Psi,\mathbf{f},S) \\ \widetilde{h}(\Psi,\mathbf{f},S)\end{bmatrix}\Big|\pi_0\Big] = \begin{bmatrix}\Psi^T\Psi & \Psi^TS \\ S^T\Psi & S^TS\end{bmatrix}^{-1}(b^2+\sigma^2).
$$

We take the inverse of the Schur complement with respect to S^TS to find the variance of $\widetilde{g}(\Psi,f,S)$:

$$
\mathrm{var}\big[\widetilde{g}(\Psi,\mathbf{f},S)|\pi_0\big] = (\Psi^T\Psi-\Psi^TS(S^TS)^{-1}S^T\Psi)^{-1}(b^2+\sigma^2).
$$

This quantity is for a fixed Ψ and fixed S. The variance of $\widetilde{g}(\Psi,f,S)$ averaged over different exploration policies and sensor values, assuming that the sensors are independent of both the policy parameters and output noise w, is given by the following equation:

$$
\begin{aligned}
\mathrm{var}\Big[\begin{bmatrix}\widetilde{g}(\mathbf{\Psi},\mathbf{f},\mathbf{S}) \\ \widetilde{h}(\mathbf{\Psi},\mathbf{f},\mathbf{S})\end{bmatrix}|\pi_0\Big] &= \begin{bmatrix}\Sigma & 0 \\ 0 & \Sigma_s\end{bmatrix}^{-1}\frac{b^2+\sigma^2}{(N-d-d_s-1)} \\
\mathrm{var}\big[\widetilde{g}(\mathbf{\Psi},\mathbf{f},\mathbf{S})|\pi_0\big] &= \frac{(\Sigma^{-1})(b^2+\sigma^2)}{n-d-d_s-1},
\end{aligned}
\tag{5.3}
$$

where d_s is the dimensionality of the sensor data.

The expressions for the variance of the two gradient estimators (Equation 5.1 and Equation 5.3) differ from each other in two factors. The variance of the estimator that ignores the sensor data has a factor of $(b^2+{a_s}^T\Sigma_s a_s+\sigma^2)$ which is reduced to $(b^2+\sigma^2)$ in

the estimator that incorporates the sensor data. We see that we get a bigger reduction in variance whenever the sensor information provides more information about the score. The second difference between the two estimators favors the estimator that ignores the sensor data because the denominator in equation 5.1 has a term that is larger than the corresponding term in equation 5.3. The difference in the denominators is the dimensionality of the sensor data d_s, which suggests that we should choose sensor encodings of low dimensionality. Whether $\widetilde{g}(\Psi, f, S)$ is more efficient than $\widetilde{g}(\Psi, f)$ depends on the relative strength of these two factors.

Correlated sensors

If the sensors are correlated with Ψ then the gradient estimator that incorporates sensor data will be biased. In this situation we can represent the distribution over sensors as a linear Gaussian distribution $S = \Psi A_{\psi,s} + 1_n {b_{\psi,s}}^T + W_s$, where W_s is a matrix of zero-mean noise where each row-vector has variance Σ_{w_s}. The scores can be written as follows:

$$f = \Psi a_\psi + \Psi A_{\psi,s} a_s + 1_n {b_{\psi,s}}^T a_s + W_s a_s + 1_n b + w. \tag{5.4}$$

The natural gradient is written as $\nabla_\pi V(\pi)\Big|_{\pi=\pi_0} = a_\psi + A_{\psi,s} a_s$. Thus we can see that $\widetilde{g}(\Psi, f, S)$ is biased by $A_{\psi,s} a_s$ whenever S is correlated with Ψ. The variance of the estimator also changes in the case of correlated sensors:

$$\mathrm{var}\big[\widetilde{g}(\mathbf{\Psi}, \mathbf{f}, \mathbf{S})\big] = \frac{(\Sigma - \Sigma_{\psi s}(A_{\psi,s}{}^T \Sigma A_{\psi,s} + \Sigma_s)^{-1} \Sigma_{\psi s}{}^T)^{-1}(b^2 + \sigma^2)}{(N - d - d_s - 1)},$$

where $\Sigma_{\psi s} = \Sigma A_{\psi,s}$ is the covariance of the policy parameters and sensor values. Thus we see that it is best to choose sensor encodings where the sensor values are uncorrelated with the policy parameters.

While incorporating sensor data can improve the learning performance, the level of improvement depends on how we choose to encode the sensor data. In practice, it may be difficult to transform the raw sensor data so that S is uncorrelated with Ψ; however, we can use regularization techniques to limit the amount of bias. In Equation 5.2, $\widetilde{h}$ serves as an unbiased estimate of a_s. If we knew the linear relationship between Ψ and S (i.e., $A_{\pi,s}$) then we could introduce an ℓ_2-norm penalty of the bias as follows: $\widetilde{h}^T {A_{\pi,s}}^T A_{\pi,s} \widetilde{h}$. As an approximation to this expression, we use the empirical covariance between Ψ and S. This gives the following penalty term: $\widetilde{h}^T S^T \Psi \Psi^T S \widetilde{h}$. Further improvements can be made by adding priors to $\widetilde{g}$ (with information matrix $\Lambda_{\widetilde{g}}$) and $\widetilde{h}$ (with information matrix λI). We proceed by adding these terms to the log-likelihood expression presented in Equation 5.2:

$$
\begin{aligned}
P(\widetilde{g}) &\propto \exp\Big\{-\frac{1}{2\sigma^2}\widetilde{g}^T \Lambda_{\widetilde{g}} \widetilde{g}\Big\} \\
P(\widetilde{h}) &\propto \exp\Big\{-\frac{1}{2\sigma^2}\widetilde{h}^T (\lambda I + \lambda_S S^T \Psi \Psi^T S)\widetilde{h}\Big\} \\
\ell(f|\Psi, S, \pi_0) = &-\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(f - \Psi\widetilde{g} - S\widetilde{h})^T (f - \Psi\widetilde{g} - S\widetilde{h}) - \\
&-\frac{1}{2\sigma^2}\widetilde{g}^T \Lambda_{\widetilde{g}} \widetilde{g} - \frac{1}{2\sigma^2}\widetilde{h}^T (\lambda I + \lambda_S S^T \Psi \Psi^T S)\widetilde{h} \\
\begin{bmatrix} \widetilde{g}^* \\ \widetilde{h}^* \end{bmatrix} &= \begin{bmatrix} \Psi^T\Psi + \Lambda_{\widetilde{g}} & \Psi^T S \\ S^T\Psi & S^T S + (\lambda I + \lambda_S S^T \Psi \Psi^T S) \end{bmatrix}^{-1} \begin{bmatrix} \Psi^T f \\ S^T f \end{bmatrix}.
\end{aligned}
\tag{5.5}
$$

Offset term

The offset term b present in equations 5.1 and 5.3 may be eliminated by augmenting the sensor data S with 1_n. This complicates the variance analysis because the inverted matrix in equation 5.3 becomes non-central, but we found that it works well in practice.

Algorithm 5.1: Natural gradient estimator with sensor data

Input: Ψ, the eligibility vectors; S, the sensor data; f, the scores;
$\{\Sigma_i\}_{i\in\{1,\ldots,n\}}$, covariance matrices where $\Sigma_i = \mathrm{var}\big[\psi(\mathbf{h}^{(i)}|\pi_0)|\pi_0\big]$;
$\{\Lambda, \lambda, \lambda_S\}$, regularization parameters.

Output: ∇, an estimate of the gradient

$n \leftarrow$ number of rows in Ψ, $d \leftarrow$ number of columns in Ψ

$d_s \leftarrow$ number of columns in S

$\Lambda \leftarrow \mathrm{blockdiag}(\Lambda, 0, \lambda I_{d_s} + \lambda_S S^T \Psi \Psi^T S)$

$X \leftarrow [\Psi, 1_n, S]$

$\nabla \leftarrow \left(X^T X + \Lambda\right)^{-1} X^T f$

$\nabla \leftarrow$ the first d elements of ∇

return ∇

5.2.3 The natural gradient estimator algorithm with sensor data

Algorithm 5.1 shows an improved gradient estimation algorithm that incorporates the sensor data. It requires the following inputs: Ψ, the eligibility vectors; S, the sensor data, f, the scores; $\{\Sigma_i\}_{i\in\{1,\ldots,n\}}$, covariance matrices where $\Sigma_i = \mathrm{var}\big[\psi(\mathbf{h}^{(i)}|\pi_0)|\pi_0\big]$; and $\{\Lambda, \lambda, \lambda_S\}$, regularization parameters.

We applied this algorithm to the `cannon1` problems described in Section 3.1 and compare the results to the natural gradient estimators presented in Chapter 4. Figure 5.3 shows different learning curves for the cannon problem as we increase the level of actuator noise. We used the following amounts of actuator noise: Figure 5.3(a), $0.1\Sigma_u$; Figure 5.3(a), $0.5\Sigma_u$; Figure 5.3(a), Σ_u; and Figure 5.3(a), $2\Sigma_u$. Each hill-climbing run lasts 30 steps and at each step we drew 5 policy trials from the current policy; the results were averaged over 1000 hill-climbing runs. Notice that the performance gains from using sensor data become more pronounced as the noise level increases.

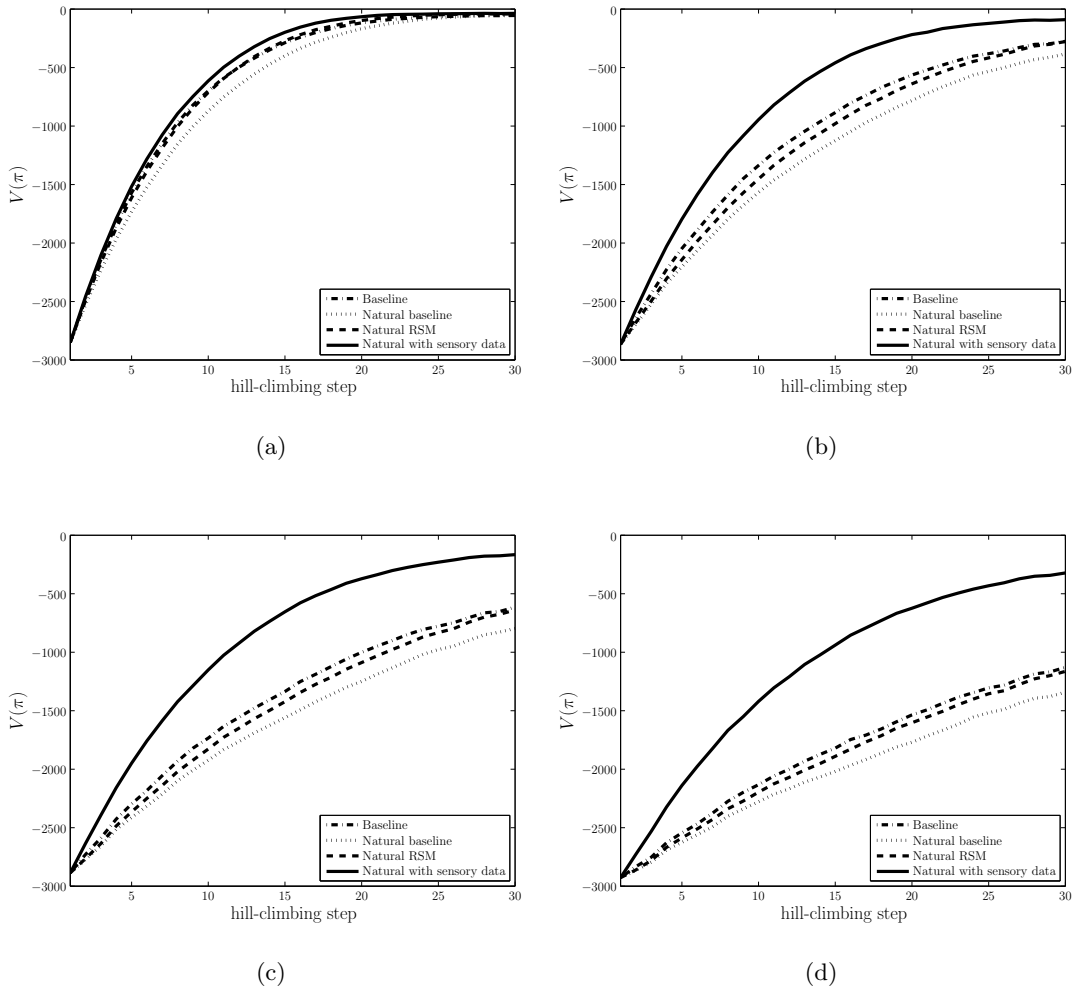


Figure 5.3: The learning curve performance of the baseline, natural baseline, natural RSM, and natural with sensor data gradient estimators for the `cannon1` problem. The figures are shown with increasing amounts of actuator noise.

5.3 Sensors for motor control

The raw sensor data for the dart and quadruped problems include noisy measurements of the joint angles at each time step. In the quadruped task, the agent can sense the positions of each leg relative to the body, but it does not have access to the absolute position and rotation of the torso. Our task is to take the sensed trajectories and transform them

into something that can improve the learning rate. This transformation should produce an encoding that is independent of Ψ (i.e., an agent's actions) and so one possible approach is to find the difference between the observed motion and expected motion at each time step.

The idea of removing the contribution of one's own motion from the sensory data is also present in the biology literature [Wolpert *et al.*, 2001].

5.3.1 Approximating the dynamical system

We approximate the difference between the expected velocities and observed velocities by learning an approximation to the dynamical system in a pre-processing phase. Using the joint-space representation of each system, the dynamics are governed by the following second-order nonlinear differential equation:

$$M(x)\ddot{x} = u(t) + g(x) + c(x, \dot{x}) + w(x, \dot{x}, t),$$

where x is the physical state of the system, $M(x)$ is the joint-space inertia matrix, $u(t)$ contains the forces and torques applied to the system, $g(x)$ contains the gravity terms, $c(x, \dot{x})$ contains the centrifugal and Coriolis terms, and $w(t)$ is the noise plus any external forces. Examples of $w(t)$ include the actuator noise and the forces caused by the ground pushing up on the feet of the quadruped. A discrete time approximation to this equation can be written as follows:

$$M(x)a = u(t) + g(x) + c(x, v) + w(x, v, t).$$

where v is the velocity and a is the acceleration of the system. Solving the above equation for a yields the following equation:

$$a = M(x)^{-1}u(t) + M(x)^{-1}g(x) + M(x)^{-1}c(x, v) + M(x)^{-1}w(x, v, t). \tag{5.6}$$

Let $\tilde{x}$ contain the sensed joint angles and let $\tilde{v}$ contain the sensed joint velocities. We approximate the expected acceleration at each time step by predicting the following quantities as a function of $\tilde{x}$ and $\tilde{v}$:

$$
\begin{aligned}
M(x)^{-1} \approx \hat{M}(\tilde{x}, \theta_M) &:= \begin{bmatrix} \phi(\tilde{x}, \theta_{M_{11}}) & \dots & \phi(\tilde{x}, \theta_{M_{1d}}) \\ \vdots & \ddots & \vdots \\ \phi(\tilde{x}, \theta_{M_{d1}}) & \dots & \phi(\tilde{x}, \theta_{M_{dd}}) \end{bmatrix}, \\
M(x)^{-1} g(x) \approx \hat{g}(\tilde{x}, \theta_g) &:= \begin{bmatrix} \phi(\tilde{x}, \theta_{g_1}) \\ \vdots \\ \phi(\tilde{x}, \theta_{g_d}) \end{bmatrix}, \\
M(x)^{-1} c(x, v) \approx \hat{c}(\tilde{x}, \tilde{v}, \theta_c) &:= \begin{bmatrix} \phi([\tilde{x}^T, \tilde{v}^T]^T, \theta_{c_1}) \\ \vdots \\ \phi([\tilde{x}^T, \tilde{v}^T]^T, \theta_{c_d}) \end{bmatrix},
\end{aligned}
\tag{5.7}
$$

where $\phi(x, \theta)$ is a function that returns a scalar value based on x and parameter θ. In our case, each θ determines the coefficients of features computed using both linear and quadratic terms of x and we use linear regression to learn each θ.

We learn θ in a pre-processing state by examining random states (x_i, v_i) in the dynamical simulator and examining the corresponding mass matrix $M(x_i)$, gravity terms $g(x_i)$, and centrifugal and Coriolis terms $c(x_i, v_i)$. The samples are drawn from a distribution of states that are likely to be encountered during policy execution. Because the sensed joint angles may contain a subset of x, our model will crudely approximation the real system in certain states. In the quadruped problem, for example, we learn these linear models without regard to the absolute rotation of the system. This is clearly an approximation for the terms that involve gravity because the direction of the gravitational force, from a frame of reference attached to the torso, depends on the quadruped's rotation relative to the ground frame.

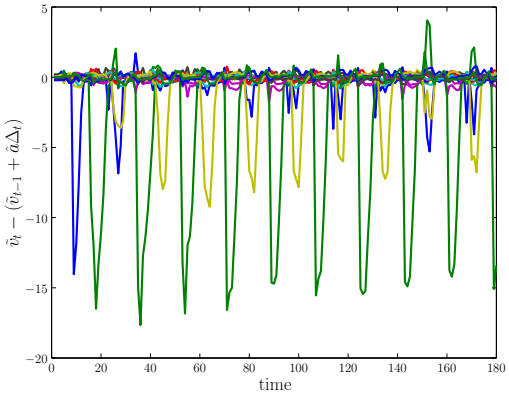
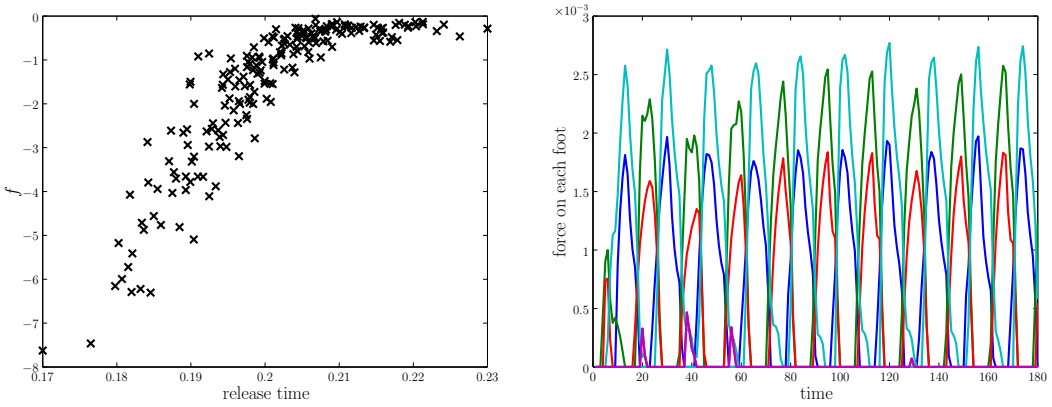


Figure 5.4: The difference between the predicted and actual velocities of 16 controllable joints during a single quadruped trial.

Given θ we can use equation 5.6 to predict the acceleration at each time step. The difference in velocity for the sensed joint angles is computed as the actual velocity at each time step $\tilde{v}_t$ minus the velocity predicted using the following expression:

$$\tilde{v}_t \approx \tilde{v}_{t-1} + \big(\hat{M}(\tilde{x}, \theta_g)u_t + \hat{g}(\tilde{x}, \theta_g) + \hat{c}(\tilde{x}, \tilde{v}, \theta_g)\big)\Delta_t, \tag{5.8}$$

where u_t is given by an agent's control and Δ_t is the time between sensor measurements. The acceleration terms are approximated using Equation 5.7. We can use the difference between $\tilde{v}$ and its predicted value as sensor data. Figure 5.4 plots this difference during a single quadruped policy trial. The spikes in the data are caused by the resistance felt by each foot as it touches the ground, as it remains planted, or as it slides across the ground. The variations in each foot step help explain the resulting perturbations in f for each policy trial. In Section 5.3.3, we describe how we transform this data into a low-dimensional encoding.



(a) (b)

Figure 5.5: (a) The scores of several policy trials plotted against their release times for the dart thrower. (b) The upward force felt by each foot of the quadruped during a single trial.

5.3.2 Additional sensor data

In the dart throwing problem, noise in the release time may cause the agent to throw the dart with different initial velocities for a fixed policy. This causes the dart to hit the wall at different distances from the target, altering the scores for each policy trial. Therefore the release time noise correlates with perturbations in f. This is demonstrated in Figure 5.5(a) which plots samples of f against the actual release time for a fixed policy. We also know that the noise is independent of Ψ because π does not control the release time. Thus using the actual release times as sensor data will not introduce bias into the gradient estimation task. Figure 5.5(b) shows the upward force felt by each foot of the quadruped during a single policy trial. In each trial, actuator noise causes the feet to hit the ground at different speeds and at different times. This information may be useful in explaining away perturbations in f.

5.3.3 Low-dimensional sensor representations

Since the variance of the gradient estimation algorithms increases with the dimensionality of the sensor data, it is critical that the difference in velocities between the expected and actual motions be projected onto a low-dimensional subspace. For each sensed joint, we can represent the difference curve $(\tilde{v}_t - (\tilde{v}_{t-1} + \hat{a}\Delta_t))$ for a particular joint as a vector. In the quadruped problem, we may also represent the force curves for each foot as a vector. We reduce the dimensionality by taking the dot-product between these values and a set of radial basis functions. We use Gaussian RBFs whose centers are evenly spaced throughout the duration of a single policy trial.

5.4 RSM gradient estimation with sensory data

We can use the same approach presented in Section 5.2.2 to incorporate sensor data into the RSM gradient estimators presented in Chapter 4. This is accomplished by limiting the bias, introduced whenever the sensory data S is correlated with Ψ_t, via regularization techniques. The first step is to augment the matrix of features Φ_t with the sensor data (i.e., $\Phi_t \leftarrow [\Phi_t, S]$). Equation 4.11 shows the expression for G. Using the augmented feature vector requires an analytical expression for $\mathrm{E}\big[\boldsymbol{\Psi}_t^T \mathbf{S} | \pi_0\big]$. If the eligibilities were uncorrelated with the sensors, then the expectation's value is zero. Using this assumption, we augment G with a matrix of zeros (i.e., $G \leftarrow [G, \mathbf{0}_{d \times d_s}]$). To limit the bias, we add a regularization parameter Λ_ρ to Equation 4.14, which estimates the optimal linear response

Algorithm 5.2: Time-variant RSM gradient estimator with sensor data

Input: $\{\Psi_t\}_{t\in\{1,\dots,t_f\}}$, the eligibility vectors; S, the sensor data; $\{f_t\}_{t\in\{1,\dots,t_f\}}$, the scores; $\{\Sigma_{i,t}\}_{i\in\{1,\dots,n\},t\in\{1,\dots,t_f\}}$, covariance matrices where $\Sigma_{i,t} = \mathrm{var}\big[\psi(\mathbf{a}_t^{(i)}|o_t^{(i)},\pi_0)|\pi_0\big]$; `features`, a user-defined function that computes $\{\Phi_t\}_{t\in\{1,\dots,t_f\}}$ and G; $\{\Lambda_\rho,\lambda,\lambda_S\}$, regularization parameters.

Output: ∇, an estimate of the gradient

$n \leftarrow$ number of rows in Ψ_1, $d \leftarrow$ number of columns in Ψ_1

$d_s \leftarrow$ number of columns in S

for $t = 1$ *to* t_f **do**

  $W_t \leftarrow \frac{1}{n}\mathrm{ddiag}(\Psi_t\Psi_t^T)$ `// ddiag clears non-diagonal entries`

end

$[\{\Phi\}_{t\in\{1,\dots,t_f\}}, G] \leftarrow$ `features`$(\{\Psi_t\}_{t\in\{1,\dots,t_f\}}, \{\Sigma_{i,t}\}_{i\in\{1,\dots,n\},t\in\{1,\dots,t_f\}})$

for $t = 1$ *to* t_f **do**

  $\Phi_t \leftarrow [\Phi_t, S]$

end

$G \leftarrow [G, \mathbf{0}_{d\times d_s}]$

$\Lambda_\rho \leftarrow \mathrm{blockdiag}(\Lambda_\rho, \lambda I_{d_s} + \lambda_S \sum_{t=1}^{t_f} S^T\Psi_t\Psi_t^T S)$

$\rho = \Big(\sum_{t=1}^{t_f} \Phi_t^T W_t \Phi_t + \Lambda_\rho\Big)^{-1} \sum_{t=1}^{t_f} \Phi_t^T W_t f_t$

$\nabla = G\rho + \frac{1}{n}\sum_{t=1}^{t_f} \Psi_t(f_t - \Phi_t\rho)$

return ∇

surface parameter ρ^*.

$$\rho^* \approx \Big(\sum_{t=1}^{t_f} \Phi_t^T \mathrm{ddiag}(\Psi_t\Psi_t^T)\Phi_t + \Lambda_\rho\Big)^{-1} \sum_{t=1}^{t_f} \Phi_t^T \mathrm{ddiag}(\Psi_t\Psi_t^T) f_t.$$

We introduce an ℓ_2-norm penalty by adding $\sum_{t=1}^{t_f} S^T\Psi_t\Psi_t^T S$ to the appropriate position of Λ_ρ. Further improvements can be made by adding additional regularization terms:

$$\Lambda_\rho := \begin{bmatrix} \Lambda & 0 \\ 0 & \lambda I_{d_s} + \lambda_S \sum_{t=1}^{t_f} S^T\Psi_t\Psi_t^T S \end{bmatrix}.$$

5.4.1 The time-variant RSM gradient est. algorithm with sensor data

Algorithm 5.2 shows an improved gradient estimation algorithm that incorporates the sensor data. It requires the following inputs: $\{\Psi_t\}_{t\in\{1,\ldots,t_f\}}$, the eligibility vectors; S, the sensor data; $\{f_t\}_{t\in\{1,\ldots,t_f\}}$, the scores; $\{\Sigma_{i,t}\}_{i\in\{1,\ldots,n\},t\in\{1,\ldots,t_f\}}$, covariance matrices where $\Sigma_{i,t} = \mathrm{var}\big[\psi(\mathbf{a}_t^{(i)}|o_t^{(i)},\pi_0)|\pi_0\big]$; `features`, a user-defined function that computes $\{\Phi_t\}_{t\in\{1,\ldots,t_f\}}$ and G; and $\{\Lambda_\rho,\lambda,\lambda_S\}$, regularization parameters.

5.5 Results

We applied Algorithm 5.1 and Algorithm 5.2 to `dart1`, and `dart2`; these problems are described in Section 3.2. The results are shown in Figure 5.6 and Figure 5.7. Each hill-climbing run lasts 30 steps (some plots only show a portion of this) and at each step we drew 12 policy trials from the current policy; the results were averaged over 500 hill-climbing runs. We used the following linear response surface model for Algorithm 5.2: $\phi^{(i)} = [1, {\psi^{(i)}}^T]$. The sensor data for `dart1` was encoded using the difference between the observed motion and expected motion at each time step (Section 5.3), and the sensor data for `dart2` was the release time. In both settings, we see an improvement in learning performance in the algorithms that incorporate sensor data. This is especially true in the noisy-release-time setting as the sensor data is able to explain away a significant portion of the noise-induced perturbations in the score.

We also applied Algorithm 5.1 and Algorithm 5.2 to `quadruped2` and `quadruped3`; these problems are described in Section 3.3. The results are shown in Figure 5.8. Each hill-climbing run lasts 20 steps and the results were averaged over 100 hill-climbing runs. In

`quadruped2` we drew 30 policy trials at each step and in `quadruped3` we drew 20 policy trials. We used the following linear response surface model for Algorithm 5.2: $\phi_t^{(i)} = [e_t, \sum_{u=t}^{t_f} \psi_u^{(i)^T}]$, where e_i is a column-vector of zeros with the ith entry set to 1. The sensor data for both problems was encoded using the upward forces felt by each of the quadruped's feet during a particular policy trial (Section 5.3) In each figure, we see an improvement in learning performance from the algorithms that incorporate sensor data.

5.6 Discussion

This chapter showed how to improve the learning performance by incorporating sensor data into the learning task. The sensor data is used to explain away the noise-induced perturbations in the score for each policy trial. We derived expressions for the variance of an estimator that incorporates sensor data and compared it to one that ignores it. The decrease in variance depends on two competing factors: the amount of noise that can be explained away via the sensor data and the dimensionality of the sensor data. We also presented useful sensor encodings for motor control problems; using these encodings improves the learning rate for the dart throwing and quadruped locomotion problems presented in Chapter 3. In the next chapter, we discuss some ideas for future improvements.

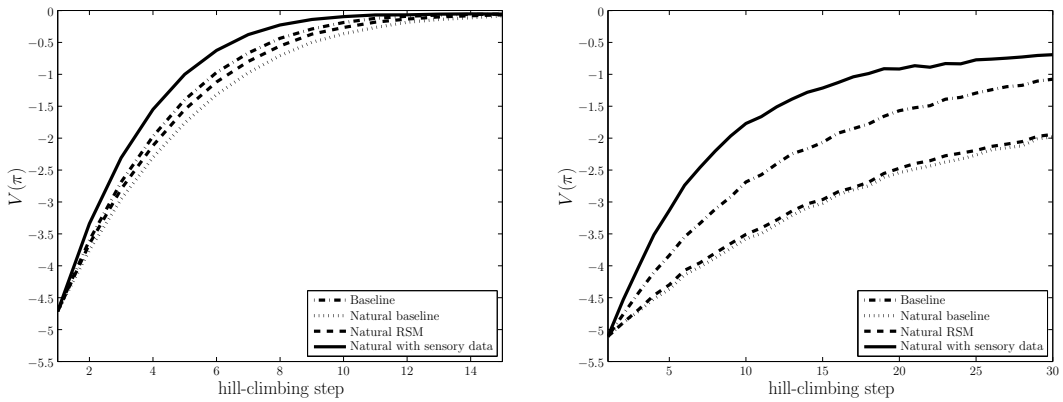


Figure 5.6: (a,b) The learning curve performances of the baseline, natural baseline, natural RSM, and natural sensor gradient estimators for `dart1` and `dart2` respectively.

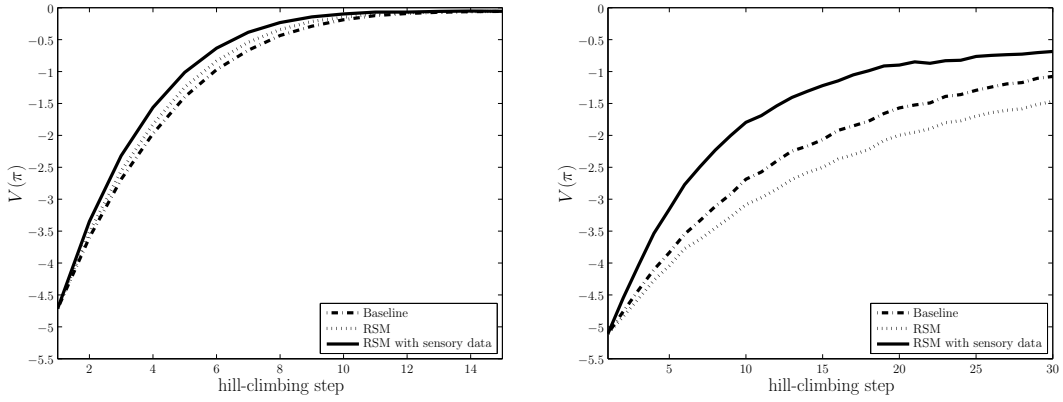


Figure 5.7: (a,b) The learning curve performances of the baseline, RSM, and time-variant RSM with sensor data gradient estimators for `dart1` and `dart2` respectively.

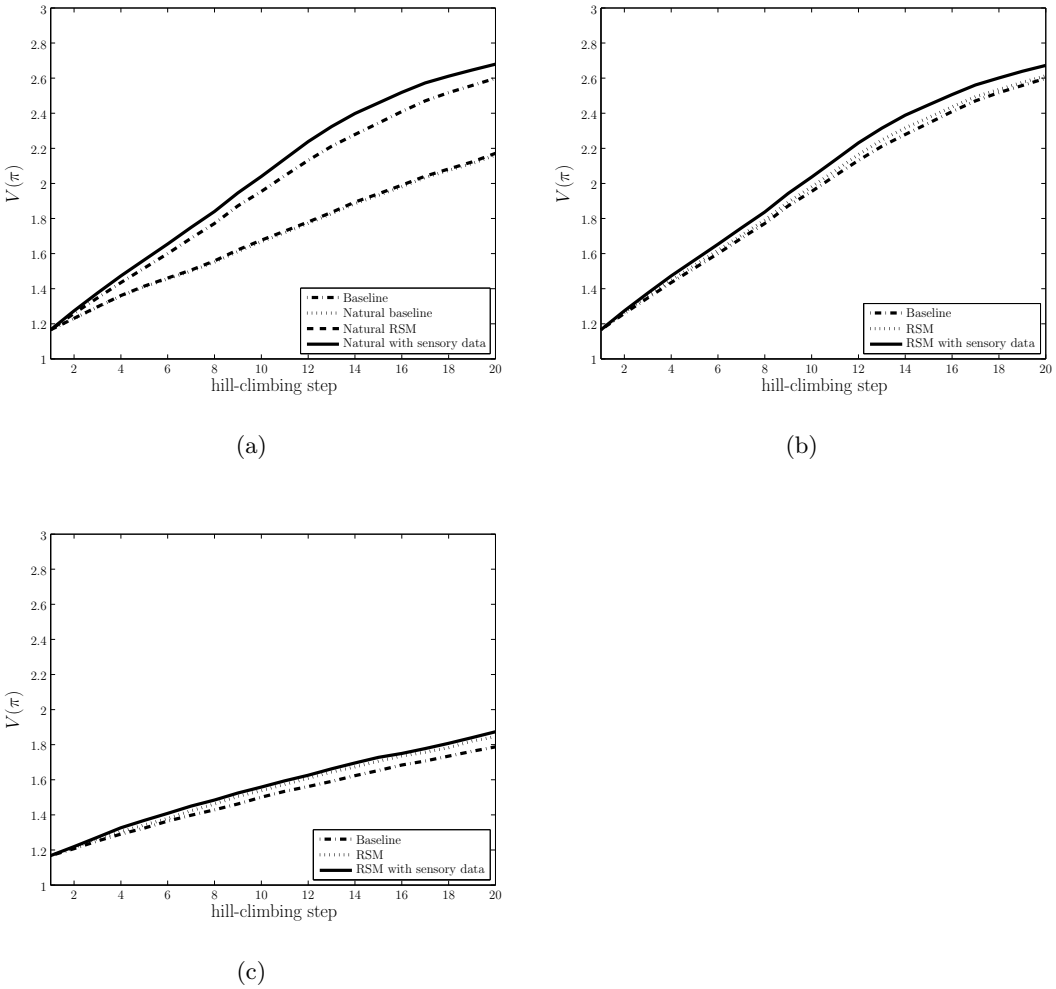


Figure 5.8: (a) The learning curve performances of the baseline, natural baseline, natural RSM, and natural with sensor data gradient estimators for `quadruped2`. (b,c) The learning curve performances of the baseline, RSM, and time-variant RSM with sensor data gradient estimators for `quadruped2` and `quadruped3` respectively.

Chapter 6

Conclusions

6.1 Summary of contributions

This dissertation presented several algorithms that improve the learning performance of policy search routines. The improvements were realized by reducing the variance of the gradient estimator, a procedure that is often difficult in noisy domains. These algorithms explain away the noise-induced perturbations in the score by examining their cause. The randomness used for exploration purposes and the environmental noise, which is measured by sensors, help an agent reason about these perturbations. This information helps explain away the perturbations, allowing us to construct better gradient estimator algorithms that we then applied to the cannon, dart throwing, and quadruped problems described in Chapter 3. Improvements were realized in each case.

Randomized policies are often used for exploration purposes; however, the stochastic choice of actions produces a noisy sample of the performance, even in deterministic domains. Fortunately, the agent has access to the artificially injected noise and it may use

this knowledge to explain away a portion of the noise-induced perturbations. Note that many other approaches effectively treat the system as a "black box." We, on the other hand, exploit this information by learning linear response surface models that predict the performance as a function of the exploration noise, as captured by the eligibility vector. Given a parameterized family of RSMs, we derived equations for the parameter that yields the minimal-variance gradient estimator. Natural gradient estimators were also produced using the same principles. We also derived expressions for the variance of these gradient estimators in the setting where an agent uses Gaussian exploration noise. Finally, we presented time-variant RSM gradient estimator algorithms that are well-suited for problems with multiple time-steps.

In addition to the noise-induced perturbations caused by randomized policies, actuator and environmental noise also produce perturbations in the observed performances. We found that the sensor data obtained during each policy trial can often be used to explain away this noise. Although an agent's policy may choose actions based on sensory feedback, most policy search algorithms typically ignore this data for gradient estimation purposes. We presented algorithms that incorporate the sensor data and derived expressions for the variance of the estimator. By comparing the amount of variance to an estimator that ignores sensor data, we found that the decrease in variance depends on two competing factors: the amount of noise that can be explained away by observing the sensor data and the dimensionality of the sensor data. We were also able to incorporate sensor data into the RSM family of algorithms. Finally, we gave some useful sensor encodings that are appropriate for motor control problems; these encodings estimate the difference between

the actual and expected motions at each time step.

6.2 Future work

This section discusses some ideas for future work.

6.2.1 Sample Reuse

Many policy search algorithms, including our implementation, effectively throw away policy trial data after each hill-climbing step. In situations where the scoring function changes slowly (relative to the step size), the old data may be used to reduce the variance at the current step. Importance sampling techniques have been used to reuse policy trial data from previous hill-climbing steps [Shelton, 2001]. The algorithms presented in this dissertation could be extended in a similar way. Another way to reuse sample data is to weight samples from prior hill-climbing steps according to a properly defined metric. This weighting scheme could also be used when estimating the optimal linear response surface model parameter. Furthermore, when incorporating sensor data, we are free to choose different weighting schemes for the coefficients corresponding to the eligibilities and for those corresponding to the sensors. This is valuable in problems where changes in the sensor data's influence evolve at a slower rate when compared to changes in the gradient. Another approach is to use Bayesian techniques, i.e., assume that the parameters from different hill-climbing steps are drawn from a distribution where the hyper-parameters are learned from experience (e.g., by using hierarchical Bayes methods). This approach allows an agent to represent problem-specific constraints via the hyper-parameters. For example, an agent

can learn over time that certain sensor variables are uncorrelated with the perturbations in performance.

6.2.2 Hierarchical control

The algorithms presented in this dissertation are used to find good low-level controllers for a single task (e.g., quadruped locomotion). Hierarchical control methods decompose a controller into several sub-controllers; these controllers are placed at different levels of a control hierarchy [Parr and Russell, 1998, Sutton *et al.*, 1999, Dietterich, 2000]. Decomposition improves the learning performance by adding structure to the policy, by allowing an agent to reuse sub-controllers in different contexts, and by allowing an agent to decompose the value function. These methods allow an agent to learn many different tasks, which in turn allows it to effectively interact with a complex world. For example, a quadruped robot can use a hierarchical controller to perform a diverse range of tasks. Low level motions might including walking straight, running, turning, and walking on an incline; higher level tasks might include navigation and search. Kolter *et al.* [2008] use a hierarchical controller for quadruped locomotion over rough terrain. Gradient based methods are probably not well suited for higher-level tasks, such as navigation, in part because of the presence of local maxima. One possible approach is to use policy search algorithms at the lowest level of a hierarchy and use value-based methods at the higher levels.

6.2.3 Quasi-newton methods

In deterministic settings, Newton, quasi-Newton, and conjugate gradient algorithms often outperform first-order techniques. Newton and quasi-Newton methods com-

pute or maintain an approximation of the Hessian of the objective function; the curvature information is used to step in the direction of the minimum (or maximum). Determining the curvature of the surface is difficult in noisy domains. By applying a few modifications to the standard quasi-Newton methods, Schraudolph *et al.* [2007] were able to make substantial improvements in the stochastic gradient setting. The authors exploited certain characteristics that, unfortunately, do not apply to the policy search setting. In their setting, stochasticity arises solely because they consider only a subset of the training data at each optimization step. Constructing policy search algorithms that exploit the curvature of the surface may provide similar improvements to the learning rate. The natural gradient methods compensate for the curvature induced by the parameterization of the log-likelihood function. However, these algorithms ignore the curvature in the performance measure itself.

Bibliography

[Albus, 1975] J. S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement, and Control*, 97:220–227, 1975.

[Amari, 1998] Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998.

[Atkeson *et al.*, 1997] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial intelligence review*, 11:11–73, 1997.

[Baxter and Bartlett, 2001] J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[Bellman, 1957] Richard Ernest Bellman. A Markov decision process. *Mathematics and Mechanics*, 6:679–684, 1957.

[Bernstein, 1967] Nikolai A. Bernstein. *The Co-ordination and Regulation of Movements*. Pergamon Press, 1967.

[Billard and Ijspeert, 2000] Aude Billard and Auke Jan Ijspeert. Biologically inspired neu-

ral controllers for motor control in a quadruped robot. In *Proceedings of the International Joint Conference on Neural Systems*, 2000.

[Bruce *et al.*, 2002] James Bruce, Scott Lenser, and Manuela Veloso. Fast parametric transitions for smooth quadrupedal motion. In *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*, 2002.

[Buehler *et al.*, 2000] M. Buehler, U. Saranli, D. Papadopoulos, and D. E. Koditschek. Dynamic locomotion with four and six-legged robots. In *International Symposium on Adaptive Motion of Animals and Machines*, 2000.

[Chernova and Veloso, 2004] Sonia Chernova and Manuela Veloso. An evolutionary approach to gait learning for four-legged robots. In *International Conference on Intelligent Robots and Systems*, 2004.

[Dietterich, 2000] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[Fukuoka *et al.*, 2003] Yasuhiro Fukuoka, Hiroshi Kimura, and Avis H. Cohen. Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. *International Journal of Robotics Research*, Vol. 22, 2003.

[Garner and Pandy, 1999] B. Garner and M. Pandy. A kinematic model of the upper limb based on the visible human project (VHP) image dataset. *Computer Methods in Biomechanics and Biomedical Engineering*, 2:107–124, 1999.

[Gates, 2006] Bill Gates. A robot in every home. *Scientific American Magazine*, December 2006.

[Greensmith *et al.*, 2001] Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1507–1514, 2001.

[Greensmith *et al.*, 2004] Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530, 2004.

[Harris and Wolpert, 1998] Christopher Harris and Daniel Wolpert. Signal-dependent noise determines motor planning. *Nature*, 394:780–784, 1998.

[Hollars *et al.*, 1990] Michael G. Hollars, Dan E. Rosenthal, and Michael A. Sherman. *SD/FAST User's Manual*. Symbolic Dynamics, Inc., Mountain View, CA, 1990.

[Hornby *et al.*, 1999] G.S. Hornby, M. Fujita, S. Takamura, T. Yamamoto, and O. Hanagata. Autonomous evolution of gaits with the Sony quadruped robot. In *Proceedings of 1999 Genetic and Evolutionary Computation Conference*, 1999.

[Kaelbling *et al.*, 1996] Leslie Pack Kaelbling, Michael L. Littman, and Andrew P. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Kakade, 2002] Sham Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, 2002.

[Kohl and Stone, 2004] N. Kohl and P. Stone. Machine learning for fast quadrupedal locomotion. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 2004.

[Kollo and von Rosen, 2005] Tonu Kollo and D. von Rosen. *Advanced Multivariate Statistics with Matrices*. Springer, 2005.

[Kolter *et al.*, 2008] J. Zico Kolter, Pieter Abbeel, and Andrew Y. Ng. Hierarchical apprenticeship learning with application to quadruped locomotion. In *Advances in Neural Information Processing Systems*, 2008.

[Konda *et al.*, 2003] Vijay R. Konda, John, and N. Tsitsiklis. Actor-critic algorithms. *Control and Optimization*, 42(4):1143–1166, 2003.

[Lawrence *et al.*, 2003] Gregory Lawrence, Noah Cowan, and Stuart Russell. Efficient gradient estimation for motor control learning. In *Proceedings of the Nineteenth International Conference on Uncertainty in Artificial Intelligence*, 2003.

[Myers and Montgomery, 1995] R.H. Myers and D.C. Montgomery. *Response surface methodology: Process and product optimization using designed experiments*. John Wiley & Sons Inc., 1995.

[Nelson, 1983] W. Nelson. Physical principles for economies of skilled movements. *Biological Cybernetics*, 46, 1983.

[Ng and Jordan, 2000] Andrew Y. Ng and Michael Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 406–415, 2000.

[Parr and Russell, 1998] Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems*, pages 1043–1049. MIT Press, 1998.

[Peters and Schaal, 2006] Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *Proceedings of the IEEE International Conference on Intelligent Robotics Systems*, 2006.

[Saranli *et al.*, 2001] U. Saranli, M. Buehler, and D.E. Koditschek. RHex: A simple and highly mobile hexapod robot. *International Journal of Robotics Research*, Vol. 20, 2001.

[Schaal, 2007] Stefan Schaal. The new robotics - towards human-centered machines. *HFSP Journal Frontiers of Interdisciplinary Research in the Life Sciences*, 1(2):115–126, 2007.

[Schraudolph *et al.*, 2007] Nicol N. Schraudolph, Jin Yu, and Simon Günter. A stochastic quasi-Newton method for online convex optimization. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 2 of *Workshop and Conference Proceedings*, pages 436–443, San Juan, Puerto Rico, 2007.

[Shelton, 2001] Christian R. Shelton. Policy improvement for POMDPs using normalized importance sampling. In *Proceedings of the Seventeenth International Conference on Uncertainty in Artificial Intelligence*, pages 496–503, 2001.

[Smallwood and Sondik, 1973] Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[Sutton *et al.*, 1999] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Journal of Artificial Intelligence Research*, 112:181–211, 1999.

[Sutton *et al.*, 2000] Richard S. Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *In Advances in Neural Information Processing Systems 12*, pages 1057–1063, 2000.

[Sutton, 1996] Rich Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*, 1996.

[Tesauro, 1992] Gerald Tesauro. Practical issues in temporal difference learning. *Machine Learning*, pages 257–277, 1992.

[Todorov and Jordan, 2002] E. Todorov and M. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5, 2002.

[Uhlenbeck and Ornstein, 1930] G. E. Uhlenbeck and L. S. Ornstein. On the theory of brownian motion. *Physical Review*, 36:823–841, 1930.

[Uno *et al.*, 1989] Y. Uno, M. Kawato, and R. Suzuki. Formation and control of optimal trajectory in human multijoint arm movement. *Biological Cybernetics*, 61, 1989.

[Watkins and Dayan, 1992] C. J. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[Weaver and Tao, 2001] Lex Weaver and Nigel Tao. The optimal reward baseline for

gradient-based reinforcement learning. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 538–545, 2001.

[Williams, 1992] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[Wolpert *et al.*, 2001] D. Wolpert, Z. Ghahramani, and J. Flanagan. Perspectives and problems in motor learning. *Trends in Cognitive Science*, 5, 2001.