REINFORCEMENT LEARNING ALGORITHMS FOR REPRESENTING AND MANAGING UNCERTAINTY IN ROBOTICS

by

John D. Martin Jr.

A DISSERTATION

Submitted to the Faculty of the Stevens Institute of Technology
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

STEVENS INSTITUTE OF TECHNOLOGY
Castle Point on Hudson
Hoboken, NJ 07030
2021

REINFORCEMENT LEARNING ALGORITHMS FOR REPRESENTING AND MANAGING UNCERTAINTY IN ROBOTICS

ABSTRACT

Robots encounter many forms of uncertainty during their operation. Whether related to the state of their knowledge or to the random variation of their environments, generally-applicable robots should be capable of making decisions in the presence of uncertainty. This dissertation considers how a robot can learn to make decisions through a trial-and-error process known as Reinforcement Learning (RL), and its contributions focus on how uncertainty can be handled throughout this process.

The first set of contributions study how to represent and manage uncertainty from a stochastic environment. We claim that a robot learner can reduce its exposure to this uncertainty by approximating distributions of the return (i.e. total future reward), then selecting actions whose outcomes stochastically dominate all others in the second order. Two algorithms are presented that support this claim. Although these algorithms are formalized for the general RL setting, we demonstrate they readily apply to robot navigation problems, where updates are performed incrementally online, and observations come from a high-dimensional stream of sensory data.

An additional contribution studies how to represent epistemic uncertainty related to the learner's knowledge of the expected return, also known as its value function. Using Bayesian non-parametric regression, we describe how to learn an effective value function and express confidence in value predictions after relatively few interactions with the environment. This algorithm broadens the applicability of robots operating in low-data regimes, particularly those using acoustic sensors, because they can learn to navigate from a sensory stream that otherwise requires a hand-engineered decision policy. This claim is empirically supported with results in simulation and from a physical underwater robot.

A final algorithm is introduced to represent a specific kind of spatially-varying uncertainty, commonly found when using acoustic localization sensors. The algorithm can help to improve a decision policy when used within a model-based RL framework. Results show this to be effective for simulated robot navigation problems.

Taken together, these contributions underscore the significance of representing and managing uncertainty as robots learn to make decisions with RL. This work provides a step toward designing more generally-applicable robots that can operate with less expert knowledge and can be deployed in a wider range of stochastic environments.

Author: John D. Martin Jr.

Advisor: Brendan Englot

Date: April 19, 2021

Department: Mechanical Engineering

Degree: Doctor of Philosophy

To Preetha and Marvin.

Acknowledgments

First, I would like to acknowledge my advisor, Brendan Englot. Brendan inspired me to pursue a career in research, back when we were industry collaborators at UTRC and Sikorsky. He opened my eyes to modern robotics and to other fascinating subjects like optimization and planning. During my time at Stevens, he was encouraging and would always take my ideas seriously. His constant positivity and his relentless commitment to the success of his students is something I aspire to emulate. And it's for all these reasons and the other ways he supported me that I am immensely grateful.

Secondly, I would like to thank my committee members: Darinka Dentcheva, Hamid Jafarnejad Sani, and Damiano Zanotto for their thoughtful participation and helpful feedback on my proposal and dissertation.

I would also like to thank the members of the Robust Field Autonomy Laboratory for being great collaborators and colleagues throughout my time at Stevens. I've always looked forward to our interactions and especially to their talks at our group meetings. Their open-mindedness and curiosity made for a welcoming and productive research environment.

Thank you to Stevens' Maritime Security Center for catalyzing my early research efforts. In particular, I want to thank Beth Austin-DeFares, LeAnn Blunt, and Hady Salloum for coordinating my first fellowship and always involving me with interesting activities.

During two summers as a research intern, I was fortunate to interact with many people who helped me to become a better researcher. I want to acknowledge Joesph Modayil, who mentored me at DeepMind. Our time together was a period of improved clarity for me; through our many video conferences, Joesph completely changed the way I view science and artificial intelligence. I also want to acknowledge

Marc G. Bellemare for mentoring me at Google Brain. Marc plugged me into the RL community, he brought me up to speed on many important results in RL, and he also taught me how to make espresso – an essential research technique I use daily. During those times, my general style of research was also shaped through conversations with folks at DeepMind and Google Brain, and the University of Alberta. A special thanks goes to Mike Bowling, Rich Sutton, Adam White, Brian Tanner, and Neil Burch.

I am also thankful to have collaborated with Kevin Doherty, John Leonard, Caralyn Cyr, Dibya Ghosh, William Fedus, Hugo Larochelle, and Yoshua Bengio, on projects that were completed during my graduate studies but do not appear in this dissertation.

Considerable thanks is due to my family and friends. Particularly, I want to thank my parents for their persistent encouragement toward my studies. Despite some initial setbacks, things ultimately turned out well. Thank you to all my friends for helping me unwind when I needed it. Thank you also to my late grandfather, undergraduate roommate, and friend, Eugene, for always being there and for supporting me during a critical time in my life.

Finally, I would like to thank my incredible wife Preetha, for all the love and support she has given me throughout graduate school (and since we've met). Our journey has been long and sometimes unpredictable, but her willingness to embrace the uncertainty was reassuring, and it compelled me to keep pushing until the end.

Funding: My studies were funded by a number of fellowships, including the Department of Homeland Security doctoral fellowship, through Stevens' Maritime Security Center (MSC), the Robert Crooks Stanley Graduate Fellowship, grant support from the National Science Foundation, Office of Naval Research, and my research assistantship from the Department of Mechanical Engineering at Stevens.

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

The science of Artificial Intelligence (AI) is often concerned with computing systems that achieve goals by making decisions. A primary objective of this science is to embed AI onto a physical device, and to scale the system so that it can solve increasingly large and complex problems to eventually surpass human capabilities. Robots serve a critical role to this endeavor; they are the physical devices that couple AI systems to reality. In this dissertation, AI systems are the software that give life to a robot's hardware. They are the medium through which a robot's sensory observations are processed and mapped to its actions.

This dissertation is particularly concerned with systems that learn through a sequential process of trial-and-error, known as Reinforcement Learning (RL). The process involves an interaction between a learning system and an environment. At any given time, the environment exists in some state. The learner experiences information about the state through its sensory stream. Whenever it takes an action, the state will change, upon which the learner will observe a scalar reward and sense the next state. This process can repeat indefinitely, and the learner's goal is often to take actions that maximize the amount of reward it expects to experience in the future.

The success of reinforcement learning in recent years has compelled many researchers to apply RL algorithms to robot decision making (Sutton et al., 2011; Kalashnikov et al., 2018; Bellemare et al., 2020). RL algorithms require less expert knowledge than classic methods involving feedback control (Åström & Murray, 2010) or motion planning (LaValle, 2006), both of which assume the environment dynamics are known a priori. Reducing prior knowledge can be beneficial for scaling robots to more general settings, where the environment dynamics are complex or unknown. Instead of using a fixed model to determine actions, a learning robot adapts its actions based on direct experience with the environment. Consequently, RL enables robots to be deployed in a wide range of settings that can be impractical without adaptation.

This dissertation studies reinforcement learning as it pertains to uncertainty in various forms, arising when a robot is scaled to more complex settings, and the robot's reliance on expert knowledge is reduced. Often the robot's performance can be significantly affected by the way uncertainty is represented and managed. In some cases, its reliability can improve when actions are selected to avoid highly-stochastic parts of the environment. In the past, improving reliability this way has involved replacing the RL objective with a similar term that penalizes undesirable uncertainty (Delage & Mannor, 2010; Nevmyvaka et al., 2006). Yet this approach almost always leads to conservative behavior and suboptimal performance. This dissertation contributes algorithms that improve reliability while simultaneously preserving optimality. Another problem robots can encounter is learning from small amounts of experience. Bayesian nonparametric regression has been identified through prior work as one potential solution (Engel et al., 2003), but these methods require the robot to store all the observations it experiences. This dissertation provides physical evidence that within low-data regimes, experience can be effectively consolidated with data compression informed by the posterior's uncertainty.

1.1 Thesis Statement

The central claim of this work is that **reinforcement learning algorithms that represent and manage uncertainty can be used to broaden the applicability of a robot decision making system.**

In RL, uncertainty can manifest in several forms, and this dissertation is concerned with two of them. The first is *aleatoric uncertainty*; this arises from random variability in the environment dynamics. In most cases, the learner does not know precisely what rewards to expect or what outcomes its action(s) will produce. The second form, *epistemic uncertainty*, arises from a lack of knowledge about some aspect of the problem; whether the learner's knowledge is adequately represented and contains all operationally-necessary information. In contrast to aleatoric, epistemic uncertainty can be reduced with more experience and continued learning.

Representing uncertainty is accomplished with a distribution or statistics of the random variable in question. This dissertation focuses on distributional representations of the return (i.e. the system's learning objective) and observation noise, and concentrates on those which are efficiently-computable and relevant to a robot's specific resource constraints. In the case of aleatoric uncertainty, the representation applies to the random return. In the case of epistemic uncertainty, a posterior distribution is placed over the expected return. The posterior is used to predict the utility of a policy when few transitions have been experienced.

Managing uncertainty is accomplished in several ways. Distributional information is employed to define useful decision making criteria. To improve reliability, the learner may wish to reduce its exposure to aleatoric uncertainty, or parts of the environment it believes are highly stochastic and potentially harmful to performance. It is also possible to inform action selection using the principle of optimism (Brafman & Tennenholtz, 2002). This approach prioritizes transitions that are experienced less frequently, so that in the limit of infinite experience, the learner will have complete knowledge of the transition structure.

1.2 Approach

Support for the thesis statement comes from several algorithms, each of which has its own mathematical justification and experimental validation. The algorithms are applied to autonomous navigation; where a robot learns to reach a goal-point. However, the proposed algorithms differ on their assumptions and on their treatment of uncertainty.

Throughout this dissertation, algorithms are evaluated using experience from both simulated environments and, when possible, from physical hardware. Simulation experiments are conducted for various dynamic systems which are controlled by both continuous and discrete actions. The low-dimensional variants are each intended to provide a proof-of-concept for my ideas and provide intuition about the proposed algorithms. The other experiments on physical platforms or with high-dimensional simulators are intended to demonstrate the generality of the proposed algorithms and their applicability to environments outside of those considered in the study.

1.3 Contributions

The primary contributions of this dissertation are based on papers that have been peer reviewed, published in archival proceedings, or that will be submitted. Chapters 1, 2, and 7 are original to this dissertation. Chapters 3, 5, and 6 contain work that was published in conference proceedings: (Martin et al., 2020), (Martin et al., 2018), (Martin & Englot, 2017), respectively. Chapter 4 contains results to be submitted at a forthcoming date. These papers are listed below.

Stochastically Dominant Distributional Reinforcement Learning

John D. Martin, Michal Lyskawinski, Xiaohu Li, Brendan Englot, 37th International Conference on Machine Learning (ICML), 2020.

Uncertainty Aware Navigation using Online Distributional Reinforcement Learning
John D. Martin, Paul Szenher, Brendan Englot,
Transactions on Robotics (TRO), 2021, (In Preparation).

Sparse Gaussian Process Temporal Difference Learning for Marine Robot Navigation
John D. Martin, Jinkun Wang, Brendan Englot
2nd Annual Conference on Robot Learning (CoRL), 2018.

Extending Model-based Policy Gradients for Robots in Heteroscedastic Environments
John D. Martin, Brendan Englot
1st Annual Conference on Robot Learning (CoRL), 2017.

Other contributions: Several contributions of mine have been omitted, because they are less related to the narrative this dissertation presents. However, they relate to at least one of the dissertation's main topics: RL, uncertainty, and robotics. These include the following works.

Adapting the Function Approximation Architecture in Online Reinforcement Learning
John D. Martin and Joesph Modayil,
38th International Conference on Machine Learning (ICML), 2021, (Under review).

Variational Filtering with Copula Models for SLAM
John D. Martin and Kevin Doherty, Caralyn Cyr, Brendan Englot, John Leonard,
International Conference on Intelligent Robots and Systems (IROS) (2020).

Autonomous Exploration Under Uncertainty via Deep Reinforcement Learning on Graphs
Fanfei Chen, John D. Martin, Yewei Huang, Jinkun Wang, Brendan Englot,
International Conference on Intelligent Robots and Systems (IROS) (2020).

Fusing Concurrent Orthogonal Wide-aperture Sonar Images for Dense Underwater 3D Reconstruction
John McConnell, John D. Martin, Brendan Englot,
International Conference on Intelligent Robots and Systems (IROS) (2020).

On Catastrophic Interference in Atari 2600 Games
William Fedus and Dibya Ghosh, John D. Martin, Marc G. Bellemare, Yoshua Bengio, Hugo Larochelle,
Biological and Artificial Reinforcement Learning (BARL) workshop,
33rd Conference on Neural Information Processing Systems (NeurIPS), (2019).

1.4 Dissertation Layout

This dissertation contains seven chapters. Chapter 2 covers foundational material needed to understand the primary contributions. Chapter 3 focuses on representing and managing aleatoric uncertainty; it introduces an algorithm for learning return distributions and using them to improve outcome reliability. These principles are expanded in Chapter 4, to address concerns that are specific to robot sensory streams and data gathering. Chapter 5 considers epistemic uncertainty in the expected return, and introduces an algorithm for learning in low-data regimes, relevant to underwater robots. Chapter 6 introduces an algorithm that can learn from sensory streams of common underwater robots, where aleatoric uncertainty in the observations is heteroscedastic. The dissertation concludes with some potential ideas for future work and a discussion of how the primary contributions support the thesis statement.

Chapter 2
Foundations

This chapter introduces the reinforcement learning problem and several algorithmic solution methods. In addition, it covers other mathematical and algorithmic concepts needed to formalize the primary contributions of the remaining chapters, including gradient flows and Gaussian process regression.

Common notation is used throughout this document. Random variables are denoted with capital letters (e.g. X). Their realizations are denoted with lowercase letters (e.g. x). Vectors are denoted with boldface letters (e.g. $\mathbf{x}$), and sets are denoted in calligraphic font (e.g. $\mathcal{X}$).

2.1 Reinforcement Learning

Reinforcement learning describes a sequential decision making problem, whereby a computing system learns to act optimally from rewards collected after taking actions. The interaction between the learning system and an environment is modeled as a Markovian decision process $\langle \mathcal{S}, \mathcal{A}, p, p_1, \gamma \rangle$, with states $S \in \mathcal{S}$, actions $A \in \mathcal{A}$, transition distribution $p(S', R | S = s, A = a)$, initial state distribution $p_1(S_1)$, and discount factor $\gamma \in [0, 1)$ (Puterman, 1994). At each time step $t \in \mathbb{N}$, the learner takes an action A_t and transitions from its current state S_t to a next state S_{t+1} while collecting a reward $R_{t+1} \in \mathbb{R}$. Repeating this sequence produces a trajectory of alternating states, actions, and rewards:

$$S_1, A_1, R_2, S_2, A_2, R_3, S_3, A_3, \ldots .$$

The outcome of an action sequence starting from state s and action a, which thereafter follows policy[1] $\pi\colon \mathcal{S} \to \mathcal{P}(\mathcal{A})$ is given by a *return*. The return at time t is defined as the discounted sum of future rewards over the corresponding state-action trajectory:

$$G_t^\pi \triangleq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s, A_t = a. \tag{2.1}$$

Since returns are random, the learner typically reasons about its expected value. The expected return is called the *value function* (Sutton & Barto, 2018), or value for short. The expectation is conditional on either a state or a state-action pair:

$$v^\pi(s) \triangleq \mathbf{E}[G_t^\pi | S_t = s], \qquad\qquad q^\pi(s,a) \triangleq \mathbf{E}[G_t^\pi | S_t = s, A_t = a].$$

The state-value function v^π is primarily used for prediction, and the action-value function q^π is used in control problems, to evaluate the outcome of an immediate action. The action-value function will be applied in the remaining chapters for the problem of autonomous navigation.

Value functions are useful for defining decision policies. The *greedy policy* is one such example where at every state, the action maximizing q is returned:

$$\pi^*(s) \triangleq \operatorname*{arg\,max}_{a \in \mathcal{A}} q(s,a), \forall s \in \mathcal{S}. \tag{2.2}$$

In the event of a tie, actions are broken arbitrarily (e.g. uniformly at random). In some cases, the learning system may wish to randomly deviate from the greedy policy, for the sake of diversifying its experience. The ε-*greedy* policy allows the system to do this with a simple modification to the greedy policy; with probability $\varepsilon \in (0,1)$, it selects a uniformly-random action, and otherwise returns the greedy action.

[1]Policies in this dissertation are viewed as stationary distributions over actions.

2.1.1 Value-based Algorithms

This dissertation is primarily focused on learning systems that estimate the value function, then use their estimates to define policies for robot control. The value-based algorithms in this dissertation are based on a well-known decomposition of the value function, known as Bellman's equation (Bellman, 1957). Here, Bellman's equation is presented in two forms, for the action-value function. The expectation equation is

$$\begin{aligned} q^\pi(s,a) &= \sum_{r,s'} p(s',r|s,a)\left[r + \gamma v^\pi(s')\right], \\ &= \sum_{r,s',a'} p(s',r|s,a)\left[r + \gamma q^\pi(s',a')\pi(a'|s')\right]. \end{aligned} \tag{2.3}$$

The equation connects the state- and action-value functions. The optimality equation

$$q(s,a) = \sum_{r,s',a'} p(s',r|s,a)\left[r + \gamma \max_{a'\in\mathcal{A}} q(s',a')\right] \tag{2.4}$$

evaluates the greedy policy. Using Bellman's equations, many RL algorithms can be defined to evaluate and improve policies.

2.1.2 Model-free Algorithms

Model-free algorithms learn value functions without explicitly representing the transition distribution $p(s',r|s,a)$. In the absence of a transition model, the learning system estimates the value function by directly experiencing transition samples (s,a,r,s'), and using them in (2.3) or (2.4). Transition samples are drawn from the true transition distribution, simply by taking actions and observing the outcomes. Two model-free algorithms considered in this dissertation are Sarsa (Rummery & Niranjan, 1994) and Q-learning (Watkins & Dayan, 1992).

Sarsa is an *on-policy* algorithm, meaning it estimates the value of the policy used to collect experience. Given a suitable step size $\alpha \in \mathbb{R}$, its update rule adjusts the current estimate in the direction of what is referred to as the *temporal difference error*: $r + \gamma q^\pi(s', a') - q^\pi(s, a)$. At each step, the update rule is given by

$$q^\pi(s, a) \leftarrow q^\pi(s, a) + \alpha \left(r + \gamma q^\pi(s', a') - q^\pi(s, a)\right). \tag{2.5}$$

Sarsa is a member of the temporal difference (TD) family of algorithms, originally introduced by Sutton (1988). In on-policy control problems, Sarsa is commonly implemented with the ε-greedy policy.

Q-learning is an *off-policy* algorithm, meaning it estimates the value of a *target policy* that is different than what is used to collect experience, the *behavior policy*. In this dissertation, the behavior policy is denoted β. Q-learning uses the greedy policy as its target and applies the update rule

$$q(s, a) \leftarrow q(s, a) + \alpha \left(r + \gamma \max_{a' \in \mathcal{A}} q(s', a') - q(s, a)\right). \tag{2.6}$$

Although a is selected with the behavior policy β, the target estimate $r + \gamma \max_{a' \in \mathcal{A}} q(s', a')$ uses the value of the greedy (target) action.

2.1.3 Model-based Approaches

When a transition model of the environment is available, the learner can use it to estimate a value function. With these algorithms, the learner would apply an update rule (e.g. Q-learning) on simulated experience, generated by the model. Several kinds of updates are available to model-based learners. In some cases, the learner may update the model from its experience in the true environment. It could also be given a model which is left fixed. In some settings, model-based algorithms are more sample-efficient than model-free algorithms, because they require fewer interactions with the environment. However, when the model's accuracy is poor, these learning systems can potentially diverge from the excessive estimation bias (Atkeson & Santamaria, 1997). Algorithm 1 outlines a common procedure that many model-based algorithms follow. This uses a model $m(\cdot|s,a) \approx p(\cdot|s,a)$ and a starting state-action distribution d.

Algorithm 1 Model-based Reinforcement Learning

```
1: for $t = 1, 2, \cdots$ do
2:     $s', r \sim p(\cdot|s,a)$ # Sample experience.
3:     $m, d \leftarrow$UpdateModel($s, a, r, s'$)
4:     $v \leftarrow$UpdateLearner($s, a, r, s'$)
5:     for $t = 1, 2, \cdots$Number of model updates do
6:         $\hat{s}, \hat{a} \sim d$ # Sample state and action.
7:         $\hat{s}', \hat{r} \sim m(\cdot|\hat{s},\hat{a})$ # Simulate experience.
8:         $v \leftarrow$UpdateLearner($\hat{s}, \hat{a}, \hat{r}, \hat{s}'$)
```

2.1.4 Value Representations

A reinforcement learning system is partly characterized by the way its value function is represented. Tabular representations are perhaps the simplest; they store a separate value for each state or state-action pair and primarily apply to finite domains. However, as the number of states and actions become large, tabular representations quickly become impractical to use. In addition, tabular representations are only applicable to settings where the full state can be directly observed. Fully-observed states are uncommon in robotics; their observations typically provide partial information about the state, from their sensors.

Instead of experiencing states directly, a general learning system receives a stream of observation vectors and rewards. Here an observation vector is generated at each time step t by an unknown random function of the state: $\mathbf{o}_t \triangleq o(S_t) \in \mathbb{R}^d$.

The learner's only knowledge of the observation function and of the environmental dynamics comes from this single stream of experience.

With no direct access to the environment's state, the learner forms an *approximate value function* to represent the value function. For example, an approximate value can be represented as a linear function of a feature vector $\mathbf{x}_t \in \mathbb{R}^\ell$, separately for each action:

$$\hat{q}(\mathbf{x}_t, a_t; \mathbf{w}_t) \triangleq \mathbf{w}_t^\top \mathbf{x}_t, \qquad \hat{q}(\mathbf{x}_t, a_t; \mathbf{w}_t) \approx q(S_t, a_t). \tag{2.7}$$

The feature vector is computed as a function of the observation. In general, the approximate value function can be a non-linear function of the features. They are often chosen from a family of parametric functions, and their parameters are sometimes optimized while learning (Mnih et al., 2015). In this dissertation, we consider instances of all these representations. Recently, the issue of optimizing the weights $\mathbf{w}$ and internal feature parameters have been addressed with deep reinforcement learning. For example, the Deep Q Network (DQN) (Mnih et al., 2015) approximates the action-value function using a convolutional neural network architecture (LeCun et al., 1998); it then optimizes the weights and feature parameters jointly through stochastic gradient-based optimization.

2.1.5 Distributional Reinforcement Learning

In settings where it is beneficial to represent the uncertainty of outcomes, systems may learn distributions over the return. A return distribution μ describes the possible outcomes a learner could experience under its current policy. This is defined as

$$\mu(s, a) \triangleq \mathrm{Law}(G_t | S_t = s, A_t = a).$$

Bellemare et al. (2017) first showed the return distribution satisfies a distributional variant of Bellman's equation. The backup operator applies the push-forward measure

$$f_{\#}\mu^{(A)} \triangleq \mu^{(f_{\#}^{-1}(A))} = \nu^{(A)}, \tag{2.8}$$

defined for all Borel measurable sets A, to the measurable mapping $f^{(r,\gamma)}(x) = r + \gamma x$. The mapping encodes the recursive decomposition of an expected sample. The distributional backup operator under policy π is defined to be

$$\mathcal{T}^{\pi}\mu(s,a) \triangleq \int_{\mathbb{R}} \sum_{s',a'} f_{\sharp}^{(r,\gamma)}\mu(s,a)\pi(a'|s')p(dr,s'|s,a). \tag{2.9}$$

Just as the standard Bellman equation is the focus of standard value-based RL, the distributional operator (2.9) plays the central role in distributional RL; it motivates algorithms which attempt to represent μ and approximate it with repeated application of the update $\mu_{t+1}(s,a) = \mathcal{T}^{\pi}\mu_t(s,a)$, for any $(s,a) \in \mathcal{S} \times \mathcal{A}$ and steps $t = 0, 1, 2, \cdots$. The distributional operator that uses the greedy policy is denoted without the superscript: $\mathcal{T}$.

Return distributions have been represented in several ways. Using features from a convolutional neural network (CNN), Bellemare et al. (2017) computes probabilities of a Boltzmann distribution. Dabney et al. (2017) consider sample-based representations generated by CNN features. And sample-based representations have been used in subsequent work (Dabney et al., 2018; Rowland et al., 2019; Martin et al., 2020).

2.2 Gradient Flows

Chapter 3 introduces an algorithm based on a gradient flow within the space of probability measures. Given a smooth function $F\colon \mathbb{R}^d \to \mathbb{R}$, and a starting point

$\mathbf{x}_0 \in \mathbb{R}^d$, the gradient flow of $F(\mathbf{x})$ is defined as the solution of the differential equation: $d\mathbf{x} = -\nabla F(\mathbf{x}(\tau))$, for time $d\tau$, $\tau > 0$ and initial condition $\mathbf{x}(0) = \mathbf{x}_0$. This is a standard Cauchy problem, with a unique solution if ∇F is Lipschitz continuous. When F is non-differentiable, the gradient is replaced with its subgradient, which gives a similar definition, omitted for simplicity.

An exact solution to the above gradient-flow problem is typically intractable. A standard numerical method, called the Minimizing Movement Scheme (MMS) (Gobbino, 1999), evolves $\mathbf{x}$ iteratively for small steps along the gradient of F at the current point. Denoting the current point as $\mathbf{x}_k$, the next point is $\mathbf{x}_{k+1} = \mathbf{x}_k - \nabla F(\mathbf{x}_{k+1})h$, with step size $h \in \mathbb{R}$. Note $\mathbf{x}_{k+1}$ is equivalent to solving optimization problem $\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} F(\mathbf{x}) + ||\mathbf{x} - \mathbf{x}_k||_2$, where $||\mathbf{x} - \mathbf{x}_k||_2$ is the 2-norm. Convergence of the sequence $\{\mathbf{x}_k\}$ to the exact solution has been established (Ambrosio, 2005).

2.3 Gaussian Process Regression

Gaussian process (GP) regression is a Bayesian non-parametric technique for learning unknown functions from data (Rasmussen & Williams, 2005). For this purpose, a random process is defined as a set of random variables indexed by $\mathbf{x} \in \mathcal{X}$. A Gaussian process is one where variables of any finite subset are jointly Gaussian. In general, a Gaussian process can represent a random vector if $\mathcal{X}$ is finite, and a random series if it is countably infinite. When $\mathcal{X}$ is uncountably infinite, the process can represent a random function $F\colon \mathcal{X} \to \mathbb{R}$. Since subsets of F are Gaussian, its distribution is fully specified by its mean and covariance, which are functions of the inputs $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$:

$$f_0(\mathbf{x}) \triangleq \mathbf{E}[F(\mathbf{x})], \tag{2.10}$$

$$k(\mathbf{x}, \mathbf{x}') \triangleq \mathbf{Cov}[F(\mathbf{x}), F(\mathbf{x}')]. \tag{2.11}$$

In the supervised regression setting, a learning system experiences an i.i.d sample of tuples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots$. The system wishes to compute a posterior distribution of F conditioned on a finite history of these observations. Observations are assumed to come from a model of the form $Y = F(\mathbf{x}) + \varepsilon$, where $F(\mathbf{x}) \sim \mathcal{N}(f_0(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ denotes a random function value, and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ denotes an independent noise variable. With GP regression, the learner analytically computes the posterior using standard Gaussian formulas.

The joint distribution of two Gaussian variables A and B, whose respective distributions are parameterized by means $\boldsymbol{\mu}_A$, $\boldsymbol{\mu}_B$, covariance matrices $\mathbf{K}_{AA}, \mathbf{K}_{BB}$, and cross covariance matrices $\mathbf{K}_{AB} = \mathbf{K}_{BA}^\top$ is

$$\begin{pmatrix} A \\ B \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_A \\ \boldsymbol{\mu}_B \end{pmatrix}, \begin{pmatrix} \mathbf{K}_{AA} & \mathbf{K}_{AB} \\ \mathbf{K}_{BA} & \mathbf{K}_{BB} \end{pmatrix} \right). \tag{2.12}$$

The conditional distribution of A on B is Gaussian $\mathcal{N}(\boldsymbol{\mu}_{B|A}, \boldsymbol{\Sigma}_{B|A})$ with parameters

$$\boldsymbol{\mu}_{B|A} = \boldsymbol{\mu}_A + \mathbf{K}_{AB}\mathbf{K}_{BB}^{-1}(B - \boldsymbol{\mu}_B), \tag{2.13}$$

$$\mathbf{K}_{B|A} = \mathbf{K}_{AA} - \mathbf{K}_{AB}\mathbf{K}_{BB}^{-1}\mathbf{K}_{BA}. \tag{2.14}$$

For a history of length $n \in \mathbb{N}$, we define a covariance vector for some input $\mathbf{x}$ to be $\mathbf{k}(\mathbf{x}) \triangleq (k(\mathbf{x}, \mathbf{x}_1), \cdots, k(\mathbf{x}, \mathbf{x}_n))$, and the elements of the covariance matrix to be $\mathbf{K}_{ij} \triangleq k(\mathbf{x}_i, \mathbf{x}_j)$ for $i, j \in \{1, \cdots, n\}$. A GP regression learning system considers the joint distribution

$$\begin{pmatrix} F(\mathbf{x}) \\ \mathbf{y} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} f_0(\mathbf{x}) \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} k(\mathbf{x}, \mathbf{x}') & \mathbf{k}(\mathbf{x})^\top \\ \mathbf{k}(\mathbf{x}) & \mathbf{K} + \boldsymbol{\Sigma} \end{pmatrix} \right). \tag{2.15}$$

To obtain predictive moments of the posterior distribution over $F(\mathbf{x})$, the learner applies (2.13) and (2.14):

$$f(\mathbf{x}) = f_0(\mathbf{x}) + \mathbf{k}(\mathbf{x})^\top (\mathbf{K} + \mathbf{\Sigma})^{-1}\mathbf{y}, \tag{2.16}$$

$$\sigma_f^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^\top (\mathbf{K} + \mathbf{\Sigma})^{-1}\mathbf{k}(\mathbf{x}). \tag{2.17}$$

The learner can use (2.16) to predict function values at unexperienced samples $\mathbf{x}$. Furthermore, it can use (2.17) to predict the epistemic uncertainty associated with the prediction. Based on the experience observed so far, the uncertainty provides a measure of confidence that the prediction is accurate. As more data is collected, this uncertainty will eventually vanish.

2.4 Summary

This chapter provided the technical foundation to frame the primary contributions of chapters 3, 4, 5, and 6. All of the subsequent chapters treat the problem of autonomous navigation as an instance of reinforcement learning, and each one applies a subset of this chapter's material to design its solution. Gradient flows and distributional RL are applied in Chapter 3. Distributional RL is also considered in Chapter 4, in addition to the concept of value representations. Gaussian process regression is applied in chapters 5 and 6, for both model-free and model-based learning.

Chapter 3

Handling Aleatoric Uncertainty of the Return

This chapter introduces a new policy to manage aleatoric uncertainty of the return. This applies specifically to distributional RL systems (Chapter 2), which learn to represent distributions of the random return. The chapter combines the proposed policy with a new distributional RL algorithm to reduce exposure to unwanted stochasticity from the environment. The chapter also presents a novel way to analyze the proposed algorithm as a Wasserstein gradient flow, which allows the algorithm's convergence to be theoretically understood. When the environment contains multiple solution paths, this proposed policy is shown to improve the reliability of the learner's actions. The chapter presents empirical results for simulated robot navigation problems.

3.1 Motivation

In reinforcement learning, a learner typically selects actions to maximize the expected sum of future rewards, or the average return. But sometimes this performance criterion can lead to undesirable outcomes, particularly when the environment is stochastic, and uncertainty prevents the learner from achieving consistently good performance. The expected return can also lead to degenerate solutions, where multiple policies have the same maximum expected return. Such scenarios frequently arise in finance, when multiple portfolios have the same average return but vary in different ways (Dentcheva & Ruszczyński, 2006). These scenarios are also prevalent in robot navigation, whenever multiple routes lead to the same goal. In each case, the expected return does not capture the full state of uncertainty, and it can be useful to include metrics of uncertainty in the learning objective.

The Conditional Value at Risk (CVaR_α) is one popular measure of uncertainty, defined as the expected value under an α-fraction of outcomes (Artzner et al., 1999). CVaR improves risk sensitivity by measuring expected utility over a set of undesirable outcomes while ignoring the others. Given a distribution of the return, this can be practical to implement and efficient to compute. The CVaR is employed in a great deal of RL research to express uncertainty for policy optimization (Chow & Ghavamzadeh, 2014; Tamar et al., 2015), and when return distributions are leared (Dabney et al., 2018; Keramati et al., 2020).

Although powerful, one question CVaR-based methods often leave unanswered is how to specify the fraction of undesirable outcomes, i.e. the risk level $\alpha \in (0, 1)$. This information typically comes from expert knowledge of the environment and is held fixed throughout learning. In large environments, however, it could be useful to consider multiple α-subsets that apply at different state and actions. This work studies an approach that does not require prior knowledge of α. In doing so, we claim that learning systems can apply more generally: to environments where specifying α is difficult or not practical.

We propose a new distributional policy that simultaneously captures many risk levels, therefore removing the need to select one. Specifically, the learner represents finite distributions over the return and considers a point-wise statistic of all CVaR values. We claim that our proposed approach allows learning systems to reduce their exposure to uncertainty better than systems that represent uncertainty with only a single CVaR statistic.

Our proposed policy is based on the Second Order Stochastic Dominance (SSD) relation. The SSD relation is defined using distribution functions and compared over the continuum of their realizable values. We say that X stochastically dominates Y in the second order when their cumulative CDFs, $F^{(2)}(z) \triangleq \int_{-\infty}^{z} F(x)dx$, satisfy the following equation, and we denote the relation $X \succeq_{(2)} Y$:

$$X \succeq_{(2)} Y \iff F_X^{(2)}(z) \leq F_Y^{(2)}(z),\ \forall\ z \in \mathbb{R}. \tag{3.1}$$

The function $F^{(2)}$ defines the frontier of what is known as the *dispersion space* (Figure 3.1), whose volume reflects the degree to which a random variable differs from its expected value, or its deterministic behavior. Outcomes that are disperse have more uncertainty and are considered risky for a decision maker. Indeed, a fundamental result from expected utility theory states that rational risk-averse agents prefer outcome X to Y when $X \succeq_{(2)} Y$ (Dentcheva & Ruszczyński, 2006). Drawing inspiration from this idea, we apply SSD to random returns of competing actions, as means to select the least-risky decision for off-policy RL.

This section presents the following contributions:

A domain-general means to measure risk: As we will show, the SSD relation eliminates the need to select and tune the CVaR's fraction of outcomes α. We apply the relation in settings where there are multiple solutions, and the learner wishes to select the most certain option.

A new uncertainty-sensitive policy: We use the SSD relation to define a new policy with a distributional action-selection criterion. Our proposed policy is both risk sensitive and able to preserve the expected return's performance.

A new RL algorithm to learn return distributions: SSD implies an ordering on the first two moments of random variables (Fishburn, 1980). To guarantee learned return distributions converge in these moments, we propose a new algorithm that is theoretically guaranteed to have this property, and we validate the theory with several

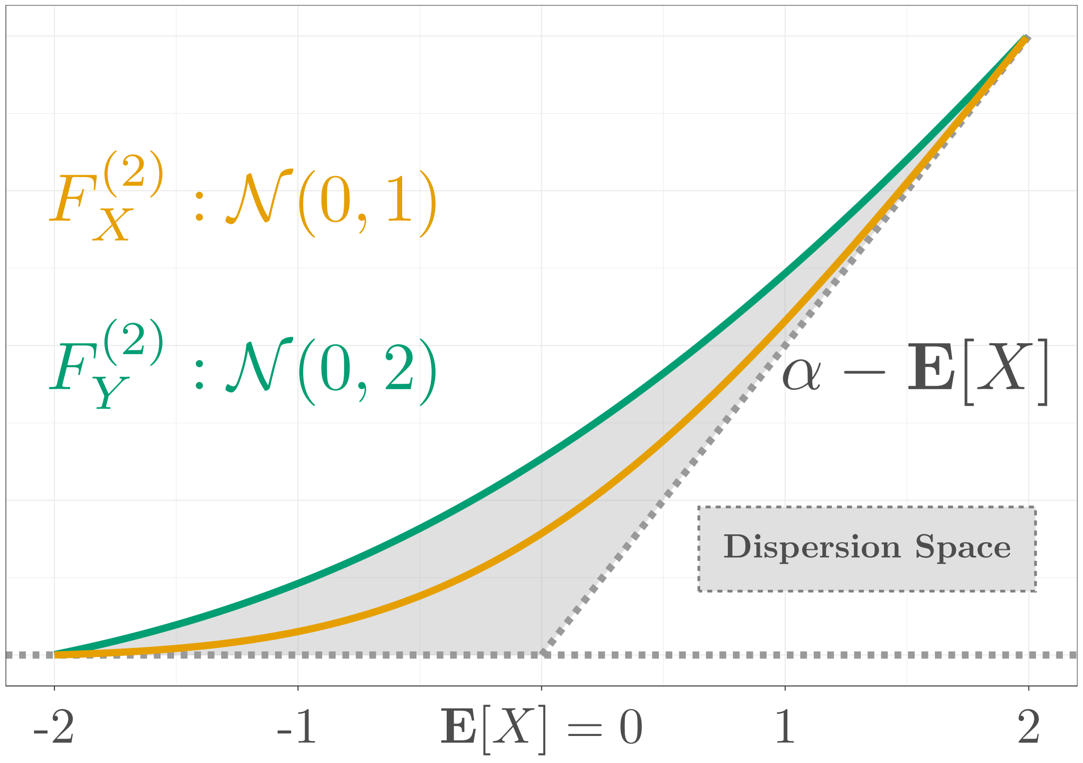


Figure 3.1: **Dispersion space:** The relative uncertainty of a random variable is shown as the space between its cumulative CDF $F_X^{(2)}$ and the asymptotes (dotted). Here, the line $\alpha - \mathbf{E}[X]$ defines the behavior of X as its uncertainty vanishes.

targeted experiments.

3.2 Action Selection with SSD

Here we define the SSD decision policy (Fig. 3.2c) as a means to mitigate uncertainty during learning. Given return distributions induced by each action in state s, the policy selects the action whose corresponding return dominates all others in the second order. This involves a point-wise comparison of their cumulative CDFs, $F_a^{(2)}(z)$, for all $a \in \mathcal{A}$, and the dominating action is the one who compares lowest at every z in the return's domain. For any state, we denote the return after taking action a to be G_a and define the SSD action as the singular element of the set

$$\mathcal{A}_{\text{SSD}} \triangleq \{a \in \mathcal{A} : G_a \succeq_{(2)} G_{a'}, \forall\, a' \in \mathcal{A} \setminus \{a\}\}. \tag{3.2}$$

At first, constructing $\mathcal{A}_{\text{SSD}}$ appears to be computationally intractable, since it involves an infinite number of point-wise comparisons (over $\mathbb{R}$). Fortunately, we can circumvent this problem by using an alternative definition of SSD involving cumulative quantile functions (Dentcheva & Ruszczyński, 2006):

$$F^{-2}(\tau) \triangleq \int_0^{\tau} F^{-1}(t)dt. \tag{3.3}$$

Here, $\tau \in (0,1)$ is a cumulative probability. With this, the alternative definition becomes:

$$X \succeq_{(2)} Y \Longleftrightarrow F_X^{-2}(\tau) \geq F_Y^{-2}(\tau)\ \forall\ \tau \in (0,1), \tag{3.4}$$

where we assume that $F_Y^{-2}(0) = 0$, and $F_Y^{-2}(1) = \infty$. However, this still involves an infinite number of comparisons. Define $\xi^{(\tau)} \triangleq F_X^{-1}(\tau)$, and notice the CVaR for risk level τ is $F_X^{-2}(\tau)/\tau = \mathbf{E}[X|X \leq \xi^{(\tau)}]$. This means the SSD relation can be interpreted as a continuum of CVaR comparisons for all $\tau \in (0,1)$. From this we can surmise that points along the boundary of dispersion space (Figure 3.1) represent unconditional Values at Risk (VaR), suggesting a computationally-tractable way to evaluate SSD.

Lemma 1. *Let $\tau \in (0,1)$ and consider $\xi^{(\tau)} = F_X^{-1}(\tau)$. Then $F_X^{-2}(\tau) = \mathbf{E}[X \leq \xi^{(\tau)}]$.*

Lemma 1 makes it possible to compare total expectations on subsets of the return space instead of dealing with probability integrals over an unbounded domain. Computations simplify even further when we consider discrete measure approximations to the return distribution. We consider a Lagrangian (particle-based) discretization, where $\mu(s,a)$ is supported on $N \in \mathbb{N}$ equally-likely Diracs at locations $z^{(i)} \in \mathbb{R}$:

$$\mu(s,a) \approx \frac{1}{N} \sum_{i=1}^{N} \delta_{z^{(i)}}^{(s,a)}.$$

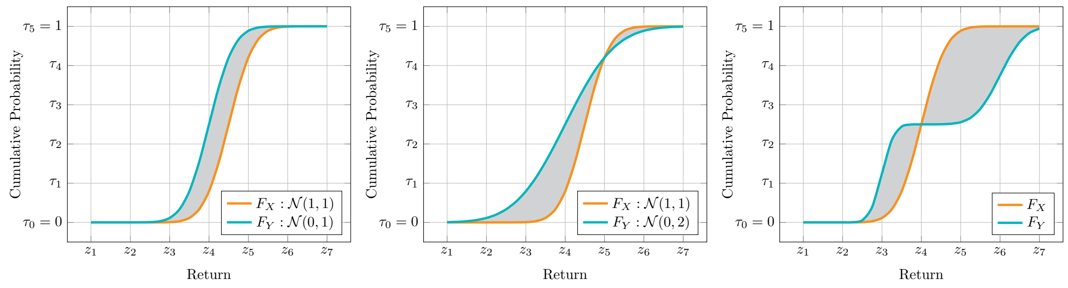
Values are simple to compute from the corresponding samples, using the empirical mean: $Q(s,a) = \frac{1}{N}\sum_{i=1}^{N} z^{(i)}$.

To apply (3.4), denote the ordered coordinates of a return distribution to be $z^{[1]} \leq z^{[2]} \leq \cdots \leq z^{[N]}$. Then given particle sets for two random returns, $G_a, G_{a'}$, induced by the actions a and a', we have the following result.

Proposition 1. *$G_a \succeq_{(2)} G_{a'}$ if, and only if*

$$\sum_{i=1}^{j} z_a^{[i]} \geq \sum_{i=1}^{j} z_{a'}^{[i]}, \ \forall\, j = 1, \cdots, N. \tag{3.5}$$

The SSD policy is executed at each step by constructing $\mathcal{A}_{\mathrm{SSD}}$ using (3.5) and a discrete representation of $\mu(s,a)$. In some cases $\mathcal{A}_{\mathrm{SSD}}$ may be empty (Figure 3.2c),



(a) $X \succeq_{(1)} Y$ and $X \succeq_{(2)} Y$ (b) $X \not\succeq_{(1)} Y$ and $X \succeq_{(2)} Y$ (c) $X \not\succeq_{(1)} Y$ and $X \not\succeq_{(2)} X$

Figure 3.2: **Stochastic dominance action selection:** Our action selection rule can be visualized with plots of the CDF. In Fig. 3.2a, $X \succeq_{(1)} Y$, because X places more mass on points larger than α. In Fig. 3.2b, the area left of z_5 is greater than the area to its right; hence $X \succeq_{(2)} Y$, because the enclosed area is always non-negative. However, in Fig. 3.2c, neither variable dominates, because after z_4, the enclosed area becomes negative with respect to X.

indicating that total dominance cannot be established. Several heuristics could be employed to handle this outcome, such as a next-best strategy. We choose to sample the greedy actions uniformly at random, thus increasing uniform exploration when dominance cannot be established while still constituting a strict enhancement of the greedy policy when multiple solutions are present.

The following proposition guarantees that the SSD action will always have the (equally) highest mean and the lowest second moment relative to the other returns.

Proposition 2 (Fishburn (1980))**.** *Assume μ has two finite moments, denoted with superscript $(\cdot)$. Then $X \succeq_{(2)} Y$ if, and only if $\mu_X^{(1)} \geq \mu_Y^{(1)}$, or $\mu_X^{(1)} = \mu_Y^{(1)}$ and $\mu_X^{(2)} \leq \mu_Y^{(2)}$.*

This implies that SSD orders random variables according to their mean unless there is a tie, in which case the second moment is used. When learning return distributions, Proposition 2 requires estimates of their first two moments be correctly approximated; otherwise, SSD orderings may be invalid.

3.3 Distributional Convergence in Moments

Moving forward, we seek algorithms for learning return distributions that are convergent in the first two moments. This can be accomplished by showing convergence in the second-order Wasserstein distance. The k-th order Wasserstein distance for any two univariate measures $\mu, \nu \in \mathcal{P}(\mathbb{R})$, is defined as

$$\mathcal{W}_k(\mu, \nu) \triangleq \inf_{\gamma \in \mathcal{P}_k(\mu,\nu)} \left\{ \int_{\mathbb{R}^2} |x-y|^k d\gamma(x,y) \right\}^{1/k}, \tag{3.6}$$

where $\mathcal{P}_k(\mu, \nu)$ is the set of all joint distributions with marginals μ and ν having k finite moments. The distance describes an optimal transport problem, where one seeks to transform μ to ν with minimum cost (Villani, 2008). Here, the cost is $|x-y|^k$. Since its convergence implies convergence in the first k moments, the $\mathcal{W}_k$ distance is appealing as a distributional learning objective.

3.4 Distributional RL as Free-energy Minimization

Our proposed algorithm is based on a minimization of the free-energy functional

$$E(\mu) \triangleq F(\mu) + \beta^{-1} H(\mu). \tag{3.7}$$

Here, F is the potential and H is the entropy of a single probability measure μ, with inverse temperature parameter $\beta \in \mathbb{R}_+$. The potential energy defines what it means for a distribution to be optimal. Its low-energy equilibrium is chosen to coincide with minimum expected Bellman error, formed from the optimality version of the distributional Bellman operator $\mathcal{T}$. The algorithm reaches a fixed point when free energy is minimized, meaning for some (s, a), $\mathcal{T}\mu(s, a) = \mu(s, a)$. Given a transition sample (s, a, r, s'), the algorithm computes samples of the target distribution $\mathcal{T}\mu(s, a)$, which are denoted $\mathcal{T}z(s,a)$. Our algorithm considers the potential energy

$$F(\mu) \triangleq \frac{1}{2}\int (\mathcal{T}z(s,a) - z(s,a))^2 \, d\mu = \int U(z) d\mu. \tag{3.8}$$

The optimal probability measure for these models is known to be the Gibbs measure: $\mu_*(z) = Z^{-1}\exp\{-\beta U(z)\}$, where $Z \triangleq \int \exp\{-\beta U(z)\}dz$.

How can the convergence of return distributions be understood as free-energy is minimized? Free-energy-based algorithms can be modeled as continuous-time stochastic diffusion processes, where distributions $\{\mu_t\}_{t\in[0,1]}$ evolve over a smooth manifold of probability measures from $\mathcal{P}_2(\mathbb{R})$. The dynamics of μ_t are known to obey a diffusive partial differential equation, called the Fokker-Planck equation:

$$\partial_t \mu_t = \nabla \cdot \left(\mu_t \nabla (\frac{\delta E}{\delta \mu_t})\right). \tag{3.9}$$

Here, the sub-gradient with respect to time is denoted ∂_t, and the first variation (Gâteaux derivative) of free energy $\frac{\delta E}{\delta \mu}$. For energy-based algorithms, the Fokker-Plank equation describes the solution path, or gradient flow, of μ as their updates evolve over a manifold of probability measures. The following result makes this claim formally.

Proposition 3 (Ambrosio (2005))**.** *Let $\{\mu_t\}_{t\in[0,1]}$ be an absolutely-continuous curve in $\mathcal{P}_2(\mathbb{R})$. Then for $t \in [0,1]$, the vector field $\mathbf{v}_t = \nabla(\frac{\delta E}{\delta t}(\mu))$ defines a gradient flow on $\mathcal{P}_2(\mathbb{R})$ as $\partial_t \mu_t = -\nabla \cdot (\mu_t \mathbf{v}_t)$, where $\nabla \cdot \mathbf{u}$ is the divergence of the vector $\mathbf{u}$.*

Convergence to an optimal point can be guaranteed provided E is convex, which we know to be the case for (3.7), which is quadratic and logarithmic in μ.

To approximately solve (3.9), we adopt an iterative procedure due to Jordan et al. (1998). The procedure discretizes time in steps of $h \in \mathbb{R}_+$ and applies the proximal operator

$$\mathrm{Prox}^{\mathcal{W}}_{hE}(\mu_k) \triangleq \underset{\mu \in \mathcal{P}_2(\mu,\mu_k)}{\arg\min}\ \mathcal{W}_2^2(\mu, \mu_k) + 2hE(\mu). \tag{3.10}$$

For every step $k \in \mathbb{N}$, the operator generates a path of distributions $\{\mu_t\}_{t=1}^{K}$ such that $\mu_{k+1} = \mathrm{Prox}^{\mathcal{W}}_{hE}(\mu_k)$ is equivalent to μ_K. In contrast with distributional RL algorithms that apply $\mathcal{T}$, we apply (3.10) to minimize a Wasserstein-regularized free energy. Since E is convex, this method obtains the unique solution to (3.9), described in the following result

Proposition 4 (Jordan et al. (1998))**.** *Let $\mu_0 \in \mathcal{P}_2(\mathbb{R})$ have finite free energy $E(\mu_0) < \infty$, and for a given $h > 0$, let $\{\mu_t^{(h)}\}_{t=1}^{K}$ be the solution of the discrete-time variational problem* (3.10)*, with measures restricted to $\mathcal{P}_2(\mathbb{R})$, the space with finite second moments. Then as $h \to 0$, $\mu_K^{(h)} \to \mu_T$, where μ_T is the unique solution of* (3.9) *at $T = hK$.*

One can evaluate the free-energy (3.8) over the solution sequence and observe it becomes a decreasing function of time (i.e. a Lyapunov function). This implies that the expected distributional Bellman error is minimized when using the JKO approach.

Proposition 5. *Let $\{\mu_k^{(h)}\}_{k=0}^{K}$ be the solution of the discrete-time variational problem* (3.10)*, with measures restricted to $\mathcal{P}_2(\mathbb{R})$, the space with finite second moments. Then $E(\mu_k)$ is a decreasing function of time.*

Finally, we show that as β is annealed, the output of our proposed free-energy optimization (3.10) is equivalent to the solution obtained from the distributional Bellman operator $\mathcal{T}$.

Theorem 1. *If $\mathcal{T}\mu = \mu$, then $\mathrm{Prox}^{\mathcal{W}}_{hE}(\mu) = \mu$ as $\beta \to \infty$.*

3.5 Discrete Measure Solutions

Given an initial set of particles at some state-action pair $z(s,a) = \{z^{(1)}, \cdots, z^{(N)}\}$, we evolve them forward in time with steps of h to obtain the solution at $t+h$. We apply a finite number of gradient steps to approximate the convergence limit $T = hK$. Finally we consider an entropic-regulated form of $\mathcal{W}_2^2$ (Cuturi, 2013) for two finite distributions $\mu = \sum_{i=1}^{N} \mu_i \delta_{x^{(i)}}$ and $\nu = \sum_{j=1}^{N} \nu_j \delta_{y^{(j)}}$:

$$\mathcal{W}_\beta(\mu,\nu) \triangleq \inf_{P \in \mathbb{R}_+^{N\times N}} \langle P, C\rangle + \beta \mathsf{KL}(P|\mu \otimes \nu),$$
$$\text{s.t.} \sum_{j=1}^{N} P_{ij} = \mu_i, \sum_{i=1}^{N} P_{ij} = \nu_j.$$

Here, $\langle P, C\rangle$ denotes the Frobenius norm between the joint P and the square Euclidean cost $C_{ij} = (x_i - y_j)^2$, and $\mathsf{KL}(P|\mu\otimes\nu) = \sum_{i,j}[P_{ij}\log(P_{ij}/\mu_i\nu_j) - P_{ij} + \mu_i\nu_j]$. The entropic term promotes numerical stability by acting as a barrier function in the positive octant. JKO steps under this new distance are defined as

$$\operatorname{Prox}_{hF}^{\mathcal{W}_\beta}(\mu_k) \triangleq \operatorname*{arg\,min}_{\mu \in \mathcal{P}_2(\mu,\mu_k)} \mathcal{W}_\beta(\mu,\mu_k) + 2hF(\mu). \tag{3.11}$$

The entropic-regularized distance, $\mathcal{W}_\beta$, can be computed using Sinkhorn iterations (Sinkhorn, 1967). This procedure (detailed in the appendix) is differentiable, which allows us to update represent particle locations with parametric models and update their predictions with gradient steps computed through auto-differentiation.

3.6 Proposed Algorithm for Learning Return Distributions

We are now ready to describe our proposed algorithm, Online WGF Fitted Q-iteration (Algorithm. 2). Our algorithm combines the solution of (3.11) into a Fitted Q-iteration algorithm (Riedmiller, 2005) to learn return distributions. The loss is computed with Algorithm 3. The algorithm can apply to both on-policy and off-policy settings. Here, we consider the off-policy case, and we compare different behavior policies that are sensitive to aleatoric uncertainty. Both distributional policies and those based on point estimates are represented with the operator $\mathcal{B}\colon \mathcal{P}(\mathbb{R})^{|\mathcal{A}|} \to \mathcal{P}(\mathcal{A})$. Given a set of return distributions, this outputs a distribution over actions.

Algorithm 2 Online WGF Fitted Q-iteration

```
1: $z(s,a) = \{z^{(i)}\}_{i=1}^{N} \ \forall \ (s,a) \in \mathcal{S} \times \mathcal{A}$
2: for $t = 1, 2, \cdots$ do
3:     $s', r \sim p(\cdot|s,a)$ with $a \sim \mathcal{B}z(s,:)$
4:     $a^* \leftarrow \arg\max_{a \in \mathcal{A}}\{\frac{1}{N}\sum_{i=1}^{N} z^{(i)}(s',a)\}$
5:     $\mathcal{T}z^{[i]} \leftarrow r + \gamma z^{[i]}(s',a^*) \ \forall \ i \in [N]$
6:     $z(s,a) \leftarrow \arg\min_z L_{hF_{\mathcal{T}}}^{\mathcal{W}_\beta}(z, z(s,a))$
```

Algorithm 3 Proximal Loss

```
1: input: Source and target particles $z, z_0; \mathcal{T}z$
2: $F_{\mathcal{T}}(z) \leftarrow \frac{1}{2N}\sum_{i=1}^{N}[\mathcal{T}z^{[i]} - z^{[i]}]^2$
3: $\mathcal{W}_\beta(z, z_0) \leftarrow \text{Sinkhorn}_\beta(z, z_0)$
4: output: $L_{hF_{\mathcal{T}}}^{\mathcal{W}_\beta} = \mathcal{W}_\beta(z, z_0) + 2hF_{\mathcal{T}}(z)$
```

3.7 Related Work

Modeling Risk in RL: Many have employed measures of uncertainty to replace or regulate the optimization objective in RL using the Markowitz mean-variance model (Markowitz, 1952). Among these include policy optimization and actor-critic algorithms (Sato et al., 2001; Tamar et al., 2013). Constraint techniques have also been considered using CVaR within a policy gradient and actor-critic framework (Chow et al., 2017). In contrast to methods that directly constrain the policy parameters, we constrain the data distribution with action selection using SSD among the return distributions. Dabney et al. (2018) trains risk-averse and risk-seeking agents from return distributions sampled from various distortion risk measures. However, they do not address problems involving multiple solutions. Furthermore, is it unclear how to sample from SSD-equivalent distortions when total dominance cannot be established. This investigation is left for future work.

Distributional RL: Our learning algorithm is inspired by the class of distributional RL algorithms (Bellemare et al., 2017). These methods represent and learn a distribution over the return, and use it to evaluate and optimize a policy (Barth-Maron et al., 2018; Hessel et al., 2018). Bellemare et al. (2017) first showed the distributional Bellman operator contracts in the supremal Wasserstein distance. They proposed a discrete-measure approximation algorithm (C51) using a fixed mesh in return space and later showed it converges in the Cramer distance (Rowland et al., 2018). Particle-based methods that use Quantile Regression (QR), have shown encouraging progress on empirical benchmarks (Dabney et al., 2017, 2018). However, understanding their convergence beyond the first moment has been more challenging. By casting the optimization problem as free-energy minimization in the space of probability measures, we show that distributional RL can be modeled as the evolution of a WGF. Updates in this framework have well-defined dynamics, permitting us to better understand convergence and optimality.

Wasserstein Gradient Flows in RL: To our knowledge WGF solutions have only been applied to policy gradient algorithms. Zhang et al. (2018) models stochastic policy inference as free-energy minimization, and applies the JKO scheme to derive

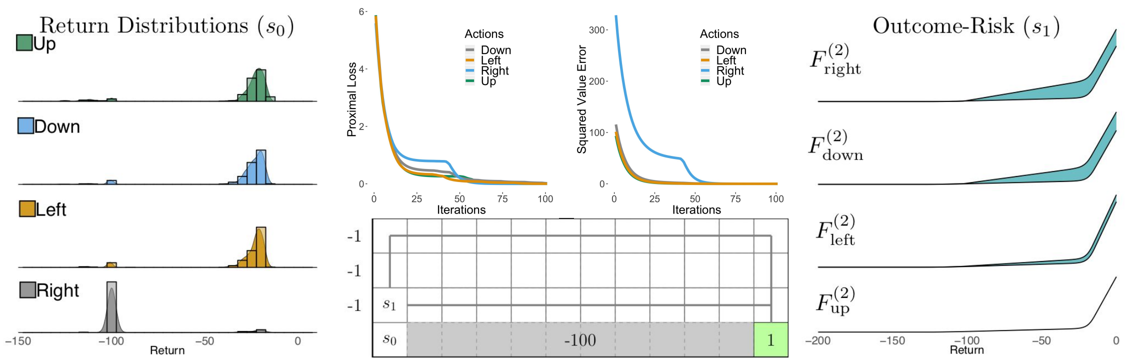


Figure 3.3: **Policy evaluation in the CliffWalk domain:** The left plot shows WGF estimates of the smoothed target distributions. Convergence of the proximal loss and the squared value error are shown in the top two plots. Outcome-risk diagrams (right) derived from distribution estimates illustrate the relative dispersion space size at s_1. The two inflections represent the bimodality of the distribution.

a policy gradient algorithm. Their method is couched within the Soft-Q learning paradigm (Haarnoja et al., 2017, 2018). These algorithms train a deep neural network to sample from a target Gibbs density using Stein Variational Gradient Descent (Liu & Wang, 2016). Our algorithm learns distributions of the underlying return and thus can be considered value-based. Furthermore, we are concerned with decision making in the presence of aleatoric uncertainty, and when the agent must select the most certain outcome from among many alternatives.

3.8 Empirical Results

In this section we verify several prior claims. Namely, we test the hypothesis that WGF regression produces two accurate moment estimates. Next we show WGF solutions from Algorithm 2 can recover return distributions in a policy evaluation setting. We extend these results to the control setting with bootstrapped off-policy updates under function approximation. In our final experiment, we quantify an agent's ability to mitigate uncertainty while gathering data with the SSD behavior policy. Details of each experiment can be found in the Appendix.

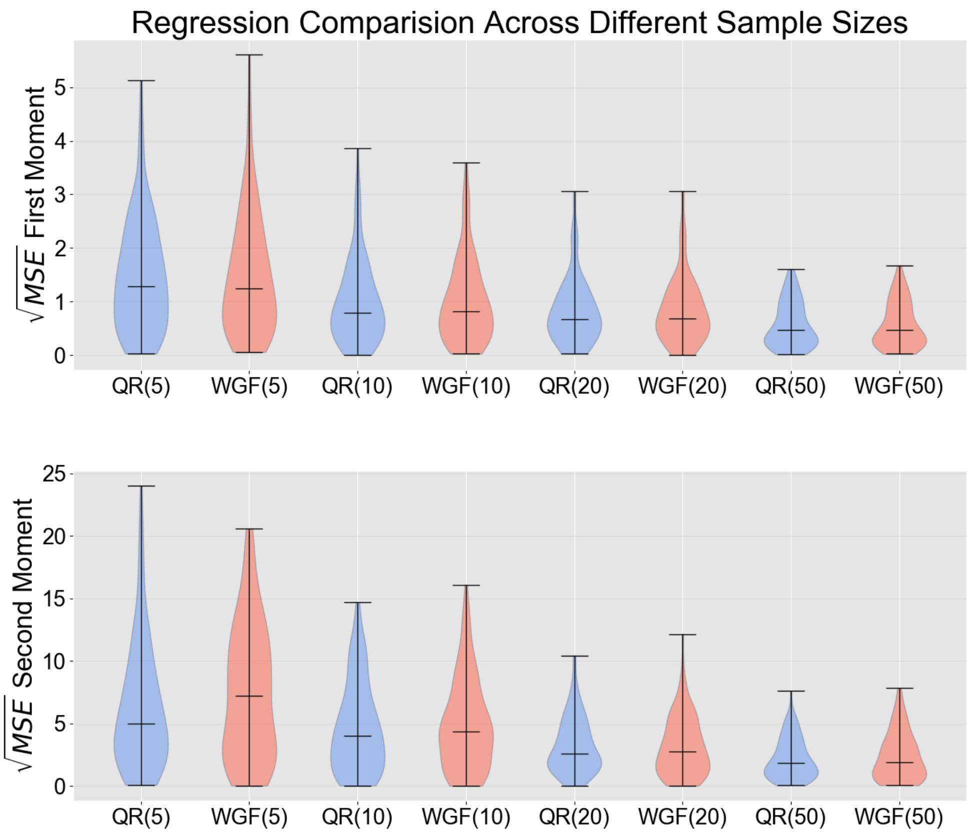


Figure 3.4: **Distributions of moment estimation error:** Quantile regression and WGF regression produce similar estimates in one dimension. The number of support samples is shown in parentheses.

3.8.1 Moment Comparison

Given that standard quantile regression learns samples from a uniform mesh in probability space, theory suggests accuracy improvements can be gained with a non-uniform mesh produced from the solution of a WGF. To evaluate this hypothesis, we compared the root mean squared error on a five component one-dimensional Gaussian mixture model, intended to be representative of a geometrically-complex return distribution. Ablations informed the parameterization of the proximal loss (See appendix). We collected data from 100 independent trials, varying the number of samples each method regressed. Our data shows there to be no statistical difference between QR and WGF regression (Figure 3.4).

We interpret the observed insignificance as a consequence of using low-dimensional data. The error from a uniform grid is expected to become more pronounced as dimensionality increases. Given that we are concerned with one-dimensional return distributions, however, these results inform different message within our problem setting; the distributions regressed through QR may be reasonably employed for SSD action selection. We believe practitioners will find this result valuable when choosing a regression method where two accurate moment estimates are required.

3.8.2 Policy Evaluation with WGF Fitted Q

Proposition 5 argues that repeated application of the proximal step (3.10) produces a decreasing function of time, implying that the Bellman free energy is minimized at convergence. Here, we verify this is indeed the case by learning the return distribution in a policy evaluation setting. The problem is set within the CliffWalk domain (Fig. 3.3). The transition dynamics follow those in Sutton & Barto (1998). However, we include a five-percent chance of falling off the cliff from adjacent states. We used fixed

Monte Carlo (MC) targets from the optimal greedy policy.

Figure 3.3 shows the convergence of the proximal loss and the mean square value error from the start state. As we can see, the estimated distribution (the histogram) accurately captures the target's features: the near certainty of walking off the cliff when moving right, the added chance of doing the same when choosing left or down, and finally the most profitable choice, moving up.

3.8.3 Control with WGF Fitted Q

Control under function approximation: In this experiment we test the hypothesis that WGF Fitted Q-iteration is scalable to function approximation in the control setting. We parameterize return distributions with a two-layer fully-connected neural network of 256 hidden units. We use off-policy updates with bootstrapped targets and compare performance results with an agent trained using the QR loss (Dabney et al., 2017) on three common control tasks from the OpenAI Gym (Brockman et al., 2016): MountainCar, CartPole, and LunarLander. The results in Figure 3.5 show that the WGF method matches the performance of QR.

Control in the presence of uncertainty: This experiment studies how aleatoric uncertainty is handled during training. Specifically, we compare different policies for selecting among a multiplicity of competing solutions. We consider the ε-greedy, SSD, and CVaR$_\alpha$ behavior policies for $\alpha \in \{0.05, 0.25, 0.45\}$. Each policy gathers data to update a greedy target policy. Different data distributions arise from the way each measures uncertainty.

We expect the data distribution under the SSD policy to favor outcomes with higher certainty, because SSD compares the expected outcome over all represented risk levels. CVaR policies consider the expected outcome for a single risk level. Uncertainty drives action selection only when the specified risk level captures the true risk in the current state. Otherwise, we expect CVaR policies to become risk neutral.

We revisit the CliffWalk domain with a modified reward structure (See appendix). Traversing the top and bottom rows have equal value. Each path has different reward uncertainty; the top row is deterministic, whereas the bottom row samples rewards from the Gaussian $\mathcal{N}(-1, 10^{-3})$. Under these conditions, we expect the SSD policy to prefer the top path and risk neutral methods to prefer the bottom row, since it will be more likely under a risk neutral policy.

Figure 3.6 shows the average episodic step count and return, along with their 95% confidence intervals computed from 50 trials. The step count data confirms our hypothesis that the SSD policy induces the least-disperse data distribution, since it takes the top path on average. We can also confirm that the ε-greedy policy chooses the bottom path, which is more likely under the sampling distribution from s_0 and incidentally more dispersed. We observe similar behavior between QR to WGF Q iteration, consistent with results in Figure 3.4. Both methods induce similar data distributions over the top path at around the 75th episode. And in this domain, the WGF method learns the quickest.

We find the greatest differences between the SSD and CVaR policies occur in the transient phase of learning (Figure 3.7). The CVaR agent takes more exploratory steps as a result of using a single uniform risk level. In high-stakes settings, the consequence of exploration can vary from undesirable to catastrophic. Here a cliff fall models a very costly outcome. Figure 3.7 shows the number of cliff falls encountered throughout learning. Using the SSD policy results in a significantly lower number of these experiences. We interpret this as positive evidence to suggest that SSD provides a more comprehensive measure of uncertainty than CVaR.

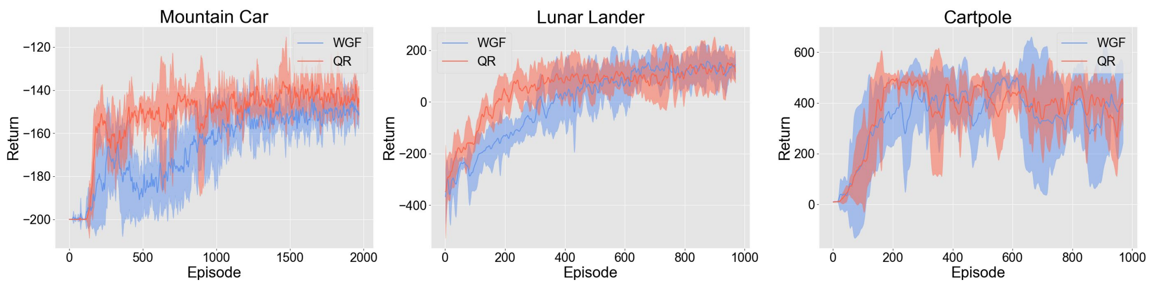


Figure 3.5: **Performance on control problems:** The WGF method matches the final average return of quantile regression.

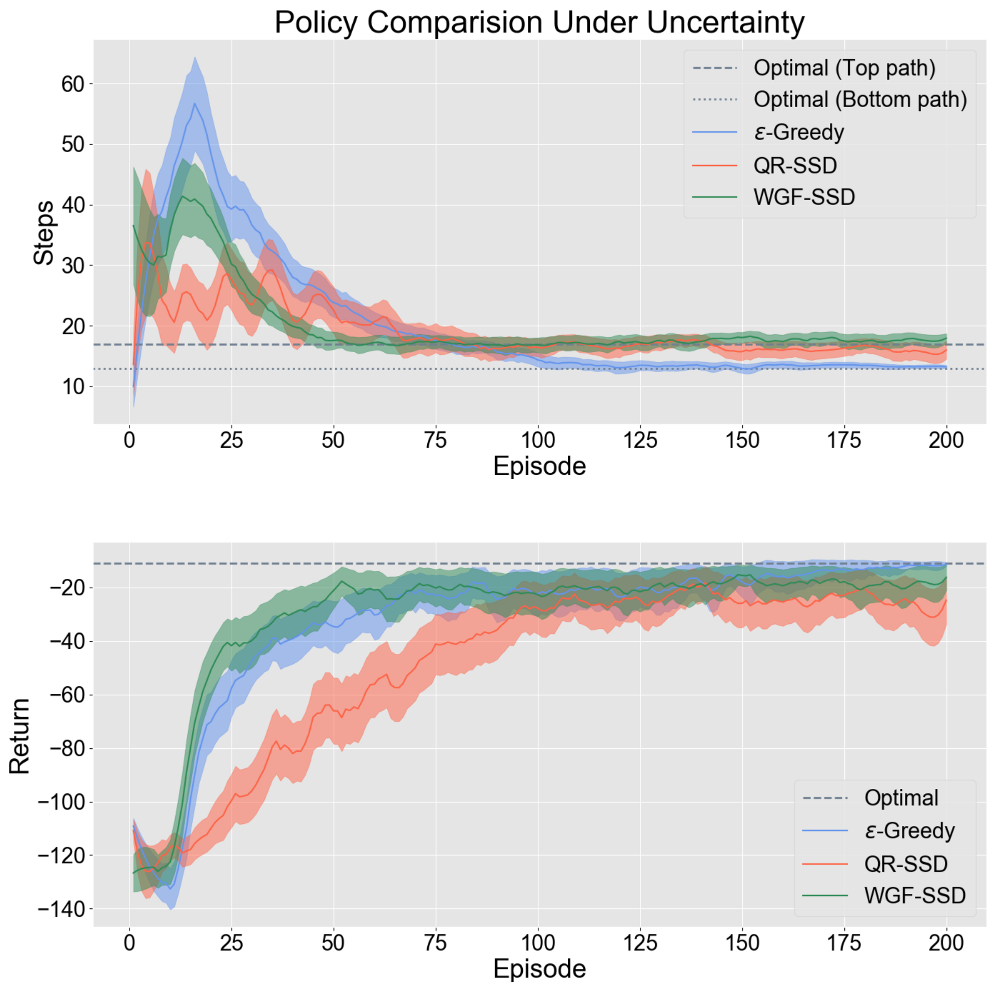


Figure 3.6: **The SSD behavior policy recovers the optimal target policy using samples from the least-disperse data distribution:** We compare the episodic step count and return using the SSD and ε-greedy policy. The distributional methods differ in their sample complexity but realize the same final solution.

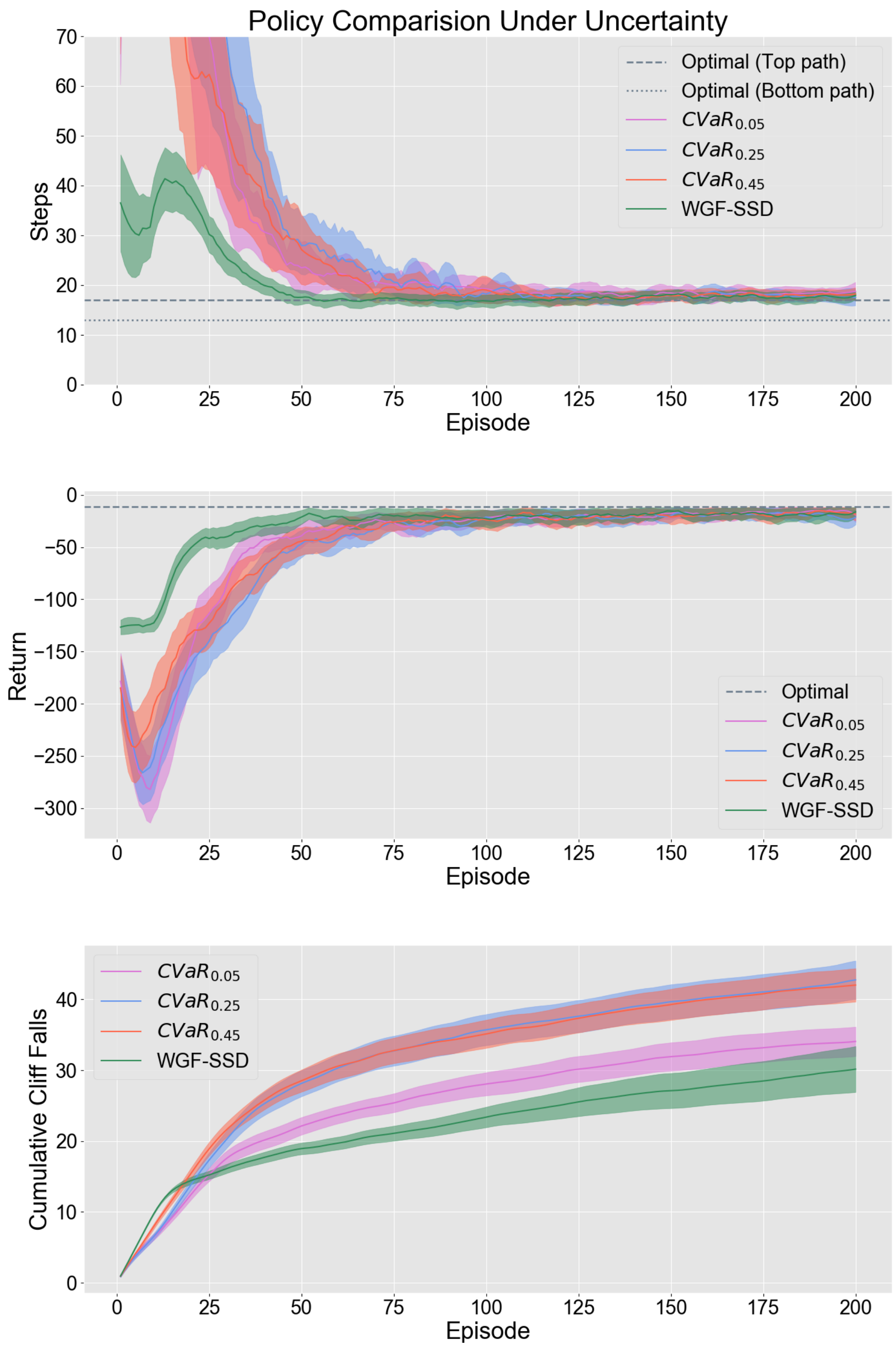


Figure 3.7: **Using many risk levels can improve exploration:** One risk level is not always appropriate for every state. Here, the CVaR policy leads the agent away from its goal, causing it to explore more than with the SSD policy, which uses many risk levels.

3.9 Discussion

This paper argues for the use of SSD to select among a multiplicity of competing solutions. This can be useful in settings where one wishes to minimize exposure to uncertainty. We presented a convergent, online algorithm for learning return distributions (WGF Fitted Q-iteration). Our simulations demonstrated the algorithm can learn good policies, and that it scales up to function approximation. Based on our experimental results, we concluded that the SSD behavior policy can reduce dispersion in the data distribution and improve exploration in the presence of uncertainty.

Chapter 4

Handling Aleatoric Uncertainty of the Return in Robotics

This chapter introduces a distributional RL algorithm for representing aleatoric uncertainty of the return. It extends several ideas from the previous chapter (Chapter 3), so they apply to robot learners using high-dimensional sensory streams.

Three main contributions are presented. The first is a new data-gathering process for mobile robots. With this process, mobile robots can naturally learn to navigate without having to impose episodic resets. The second contribution is a new distributional RL algorithm. We show this algorithm is particularly well-suited to high-dimensional sensory streams of unknown structure. Additionally, the algorithm is designed to be incremental, so it can be useful to resource-constrained systems that do not store histories of experience. The final contribution is an empirical finding that shows distributional algorithms under function approximation struggle to actualize the benefits of the SSD policy (Martin et al., 2020). This finding motivates future work discussed in Chapter 7.

4.1 Continual Reinforcement Learning for Mobile Robots

Mobile robots are natural continual learners (Ring et al., 1994). Many of them operate in large, expansive spaces, with discernibly unbounded volumes. Their experiences can span long stretches of time that conceivably never end. Yet many mobile robots restrict their applicability to more conservative environments when learning to act. Instead, these robots consider only environments that can be experienced through episodic interactions.

Episodes can be thought of as segments of experience. These begin at some starting state and continue until a termination state. Upon termination, a reset will occur, assuming the robot can be brought back to the starting state (according to its distribution). The episodic construct can be realistic for some robots, such as manipulators. But for a number of reasons, the episodic construct limits the applicability of mobile robot systems learning to navigate. This chapter argues that by abandoning episodes in favor of continual RL, the environments to which mobile robots can apply become less restrictive.

Consider the following example to motivate our proposed learning process. A mobile robot taxi has been programmed to service transport requests across a city. Based on prior knowledge, perhaps from past learning experiences, the robot is able to complete these requests autonomously. For our purposes, this implies the robot can experience the entire map in which it operates, and it can operate over an indefinite period of time. It is assumed the actions used to service an individual request come from a single policy that terminates at a goal-point, and a collection of these constitute the learner's behavior policy. As traffic conditions may randomly vary, the robot stands to benefit from representing the inherent uncertainty of returns and adapting a policy to avoid unnecessary variation. To reduce the chance of any unintended outcomes, the robot does not update the policy it currently follows. Instead, it uses the experience to update a separate target policy, using off-policy RL. At some point in time, when the taxi receives a request for the target's goal, it will stop learning and switch to the target policy. It is generally assumed the target policy has sufficiently converged when this happens. No learning occurs while the robot executes the target policy. After the goal is reached, the system switches back to the behavior policy and continues learning as before.

Compared to episodic learning, the continual learning process just described is a more natural model of mobile robot operation. Episodic learning requires the robot to teleport back to the start state after servicing a request. Not only is this awkward for mobile robots, but such resets can also be impractical to implement in the kinds of large domains in which mobile robots can operate. The proposed continual process allows for a robot to operate over long periods of time without interruption. In addition, the robot will be better equipped to deal with environmental changes, such as shifting traffic conditions and other environmental uncertainties.

4.1.1 Learning from observations with unknown structure

Another issue that precludes broad application of learning robots is the need to structure the representation. When observations are known to come from a spatial process, then representations based on convolutional neural networks (LeCun et al., 1998) are known to be effective; for instance, in standard distributional RL (Bellemare et al., 2017; Dabney et al., 2017), where observations arrive as standard images. However, a robot's sensory stream can output data in many other forms, and these representations may be uninformative of any underlying observation structure.

Lidars can observe spatial phenomena, but often these sensors output information as unstructured point clouds: sets of vectors expressed in body-centric coordinates. The dimensionality of a point cloud can change from scan to scan, so it is reasonable to use an occupancy vector as an alternative; a binary vector of fixed dimensionality, representing the occupancy of a return within a three-dimensional spatial grid. Occupancy vectors are considered unstructured, because their arrangement can be arbitrary, and generally they are not indicative of any mutual dependencies. Alas, lidar data is often preprocessed to provide spatial structure to common robot learning systems. In comparison, learners that support observations of unknown structure have potential to apply more broadly: to domains where sensory structure is unknown or is too complex to be specified a priori.

4.2 Distributional Continual Reinforcement Learning

In what follows, this chapter formalizes the continual RL setting and proposes an incremental RL algorithm for learning return distributions as functions of observations with unknown structure. The chapter's main claim is the proposed algorithm broadens the applicability of robot learners representing and managing uncertainty.

Algorithm 4 describes the observation process, where a robot experiences an unending stream of observations and rewards. The robot has no knowledge of the underlying state dynamics or observation function aside from the information provided from its single stream of experience.

The robot's goal is to learn a policy that maximizes the action-value at every state while also minimizing uncertainty in the return as a secondary objective. In the types of navigational domains this chapter considers, multiple actions whose values are close (or equal) will exist, but their associated uncertainties may be quite different. Similar to Chapter 3, the aleatoric uncertainty is represented with learned distributions over the random returns

$$\mu(s_t, a_t) \triangleq \mathrm{Law}(G_t | S = s_t, A = a_t).$$

The robot's learning system does not have direct access to the environment state – only the observation stream. Furthermore, the functional form of the return distribution is unknown. We therefore follow a conventional approach of approximating it with an empirical distribution supported on $n \in \mathbb{N}$ equally-weighted atoms at the locations $g^{(i)}(s, a) \in \mathbb{R}$ (Dabney et al., 2017, 2018), $i = 1, \cdots, n$:

$$\mu(s, a) \approx \sum_{i=1}^{n} \delta(g^{(i)}(s, a)).$$

Each support point is further approximated as a linear function of the features $x_t \in \mathbb{R}^l$,

$$\hat{g}^{(i)}(x_t, a_t; w_t^{(i)}) \triangleq \left\langle w_t^{(i)} | x_t \right\rangle, \qquad \hat{g}^{(i)}(x_t, a_t; w_t^{(i)}) \approx g^{(i)}(s, a).$$

The features are in general nonlinear functions of the observations, provided as part of the problem specification. Often, these functions are taken to be the final layer of a neural network. For each action, the set of weights $w_t^{a_t} \triangleq \{w_t^{(i)} \in \mathbb{R}^l, i = 1, \cdots, n\}$ is updated according to an online temporal difference learning algorithm. Taken together, the features and weights define the approximate return distribution $\hat{\mu}(x_t, a_t; w_t^{a_t}) \approx \mu(s, a)$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$:

$$\hat{\mu}(x_t, a_t; w_t^{a_t}) \triangleq \sum_{i=1}^{n} \delta(\hat{g}^{(i)}(x_t, a_t; w_t^{(i)})).$$

4.2.1 Quantile Regression with Temporal Differences

The proposed approximation uses a uniform discretization of the probability space $\{\tau_i = \frac{2(i-1)+1}{2n}, i = 1, \cdots, n\}$, and it fits each weight vector by minimizing a function of the temporal difference error

$$D_t^{(j)} \triangleq r_{t+1} + \gamma \hat{g}^{(j)}(s_{t+1}, a_{t+1}; w_t^{(j)}) - \hat{g}^{(i)}(s_t, a_t; w_t^{(i)}).$$

Here $\hat{g}^{(j)}(s_{t+1}, a_{t+1}; w_t^{(j)})$ denotes the j-th sample of the next return distribution under the weights of action a_{t+1}. We consider an off-policy learning algorithm; the next action is chosen with the target policy π, and updates are made with experience gathered under the behavior policy β. Experience is used to minimize the quantile regression loss $L_{\text{QR}}^{(i)}(w) \triangleq \mathbf{E}[D_t^{(j)}[\tau^{(i)} - \mathrm{I}(D_t^{(j)} < 0)]]$. We approximate this with an empirical average of samples from the target distribution $\hat{L}^{(i)}_{\text{QR}}(w) \approx L^{(i)}_{\text{QR}}(w)$:

$$\hat{L}^{(i)}_{\text{QR}}(w) \triangleq \frac{1}{n}\sum_{j=1}^{n} D^{(j)}_t[\tau^{(i)} - \mathrm{I}(D^{(j)}_t < 0)], \tag{4.1}$$

$$w^{(i)}_{t+1} = \operatorname*{arg\,min}_{w\in\mathbb{R}^l} \hat{L}^{(i)}_{\text{QR}}(w),\ \forall\ i = 1, \cdots, n. \tag{4.2}$$

The loss depends only on the weights of the term $\hat{g}^{(i)}(x_t, a_t; w^{(i)}_t)$, with the target term held fixed. The weights are updated in steps of $\alpha_t \in \mathbb{R}$ in the direction of the loss function's semi-gradient, which ignore the dependence of the target term:

$$w^{(i)}_{t+1} = w^{(i)}_t + \alpha_t \nabla \hat{L}^{(i)}_{\text{QR}}(w^{(i)}_t).$$

This algorithm follows the same principles as the algorithm described by Dabney et al. (2017). However, the proposed algorithm represents atoms as linear functions of features for the first time.

Note that each atom is trained in isolation from the other atoms. Since they do not share weights, and because there is no constraint enforcing a particular order, the full collection cannot be guaranteed to represent a proper probability distribution at every point in time. However, after a sufficient amount of data has been experienced, the atoms tend to converge to the correct ordering.

Algorithm 4 Continual Off-policy Learning

```
input: $\alpha_t$, $\gamma$, $\beta$
initialize: $o_1 \sim p_1(\cdot)$, $\mathbf{w}_1$
for $t = 1, 2, \cdots$ do
    $a_t \sim \beta(x_t)$
    $o_{t+1}, r_{t+1} \sim p(\cdot|o_t, a_t)$
    $x_{t+1} = \text{concatenate}(o_{t+1}, 1)$
    $w_{t+1} \leftarrow \text{QTD}(x_t, a_t, r_{t+1}, x_{t+1}, a_{t+1}, w_t)$
```

Algorithm 5 Quantile Temporal Difference Regression (QTD)

```
**input:** $x, a, r, x', a', w$
**parameters:** $\gamma, \alpha$
**do in parallel for** $i = 1, \cdots, n$
    $w^{(i)} \leftarrow w^{(i)} + \alpha \nabla \hat{L}^{(i)}_{\mathrm{QR}}(w^{(i)})$
```

4.3 Empirical Results

This section presents experimental results showing there are incremental distributional RL algorithms that can learn from high-dimensional sensory streams with unknown structure. Experiments additionally show the proposed algorithm can be applied to a realistic robot navigation problem. Finally, the section analyzes the approximate return distribution's representation in the context of managing aleatoric uncertainty with the SSD policy.

4.3.1 A Simulation Environment for Autonomous Navigation

Experiments consider the problem of autonomous navigation in a simulated suburban environment. The environment is filled with buildings, houses, trees, traffic lights, lampposts, one-way lanes, benches, signs, and rocks. All of these elements are three-dimensional, and their geometry is highly emblematic of the physical space it strives to model (Figure 4.1). The environment's layout and dynamics come from the Carla simulator (Dosovitskiy et al., 2017), in particular its Town-02 map (Figure 4.2). At the most basic level, the environment's underlying state is a continuously-valued position of a simulated driving robot. Carla makes it possible to model multiple vehicles, but our experiment only considers one. Our experiments map Carla's continuous state space to a fixed network of 134 waypoints, over which the vehicle traverses.

The network of waypoints approximates the roadway as a directed graph, where directionality imposes the flow of traffic. Each vertex has two outgoing edges and

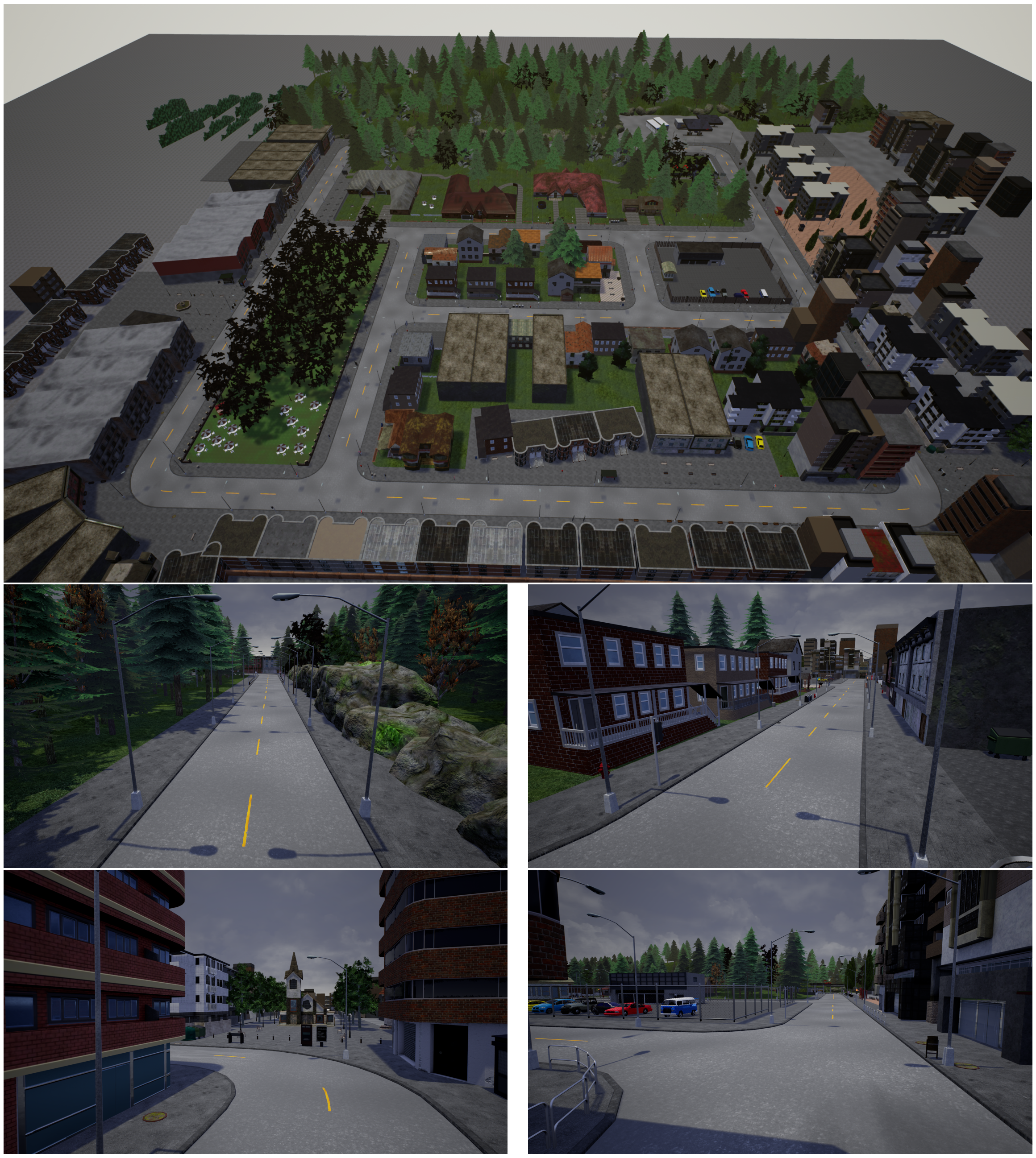


Figure 4.1: **Carla Simulation Environment:** Three-dimensional scenes are shown from Carla's Town02 environment. The environment contains a variety of structures that a self-driving robot would be expected to sense and process.

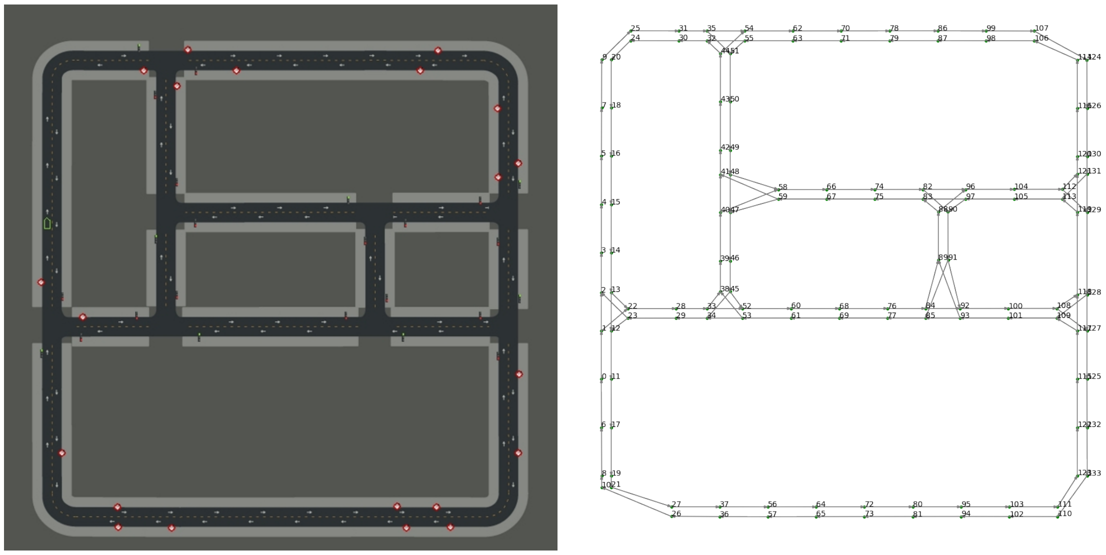


Figure 4.2: **Road network for simulated environment:** The left image shows the Carla Town-02 map. The right image shows the corresponding state-space graph after a skeletonization routine was applied from the full-scale simulator.

at most two incoming edges. The outgoing edges represent two available actions: a transition with the outgoing edge in the body-centric lefthand plane, or either a self-looping edge or an outgoing edge in the righthand plane. Self-looping transitions occur only along the straightaways to represent a braking action. At intersections, there are always two outgoing edges.

The observation vector comes from a simulated lidar with thirty-two beams and a range of 50m. At each step, the range returns are used to populate two planar occupancy grids at different elevations (Figure 4.3). Each occupancy grid is a 256×256 mesh centered on the sensor and placed on top of the vehicle. When vectorized, this produces a binary observation vector with $d = 256 \times 256 \times 2 = 131,072$ dimensions.

Occupancy grids were chosen, because of their constant dimensionality. Each of its cells encode the presence of a range return for some spatial region. Vectorizing the occupancy grids ignores the mesh's planar arrangement and effectively removes the spatial inductive bias. This choice is important, as it reflects the more general

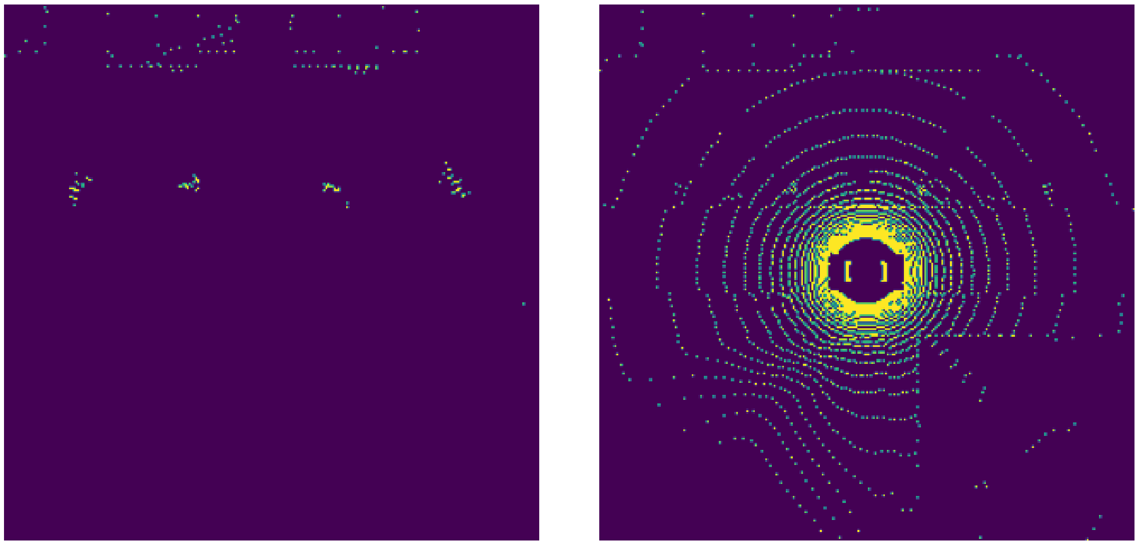


Figure 4.3: **Occupancy grids:** Grid cells reflect presence of a lidar return in a cubic spatial region centered on the vehicle. Data is shown for the evaluation start state (123). The left image captures the high-elevation features, and the right image captures those close to the ground.

case, when the observation structure is unknown.

Several different rewards are experienced throughout the environment. At each regular transition, a time penalty of -1 is given. A bonus of one is given for reaching the goal, making the reward zero at both goal states (58,59). Stochastic rewards are observed at a simulated sidewalk placed at 88 and 90. This is drawn from a Gaussian distribution $\mathcal{N}(-1, 0.2)$ with values clipped at -2 and 0.

Empirical Methodology: Experiments compare the performance of different target policies. Since the target policy is learned off-policy, from experience collected in a continual manner, it is less straightforward to empirically measure its performance online. Therefore, evaluations are performed with counterfactual simulations. Every 10000 steps, the discounted return is approximated over a trajectory of maximum length of 1000 steps using data in a separate (counterfactual) environment instance. Although this evaluation strategy is not physically realistic, it is appropriate for our

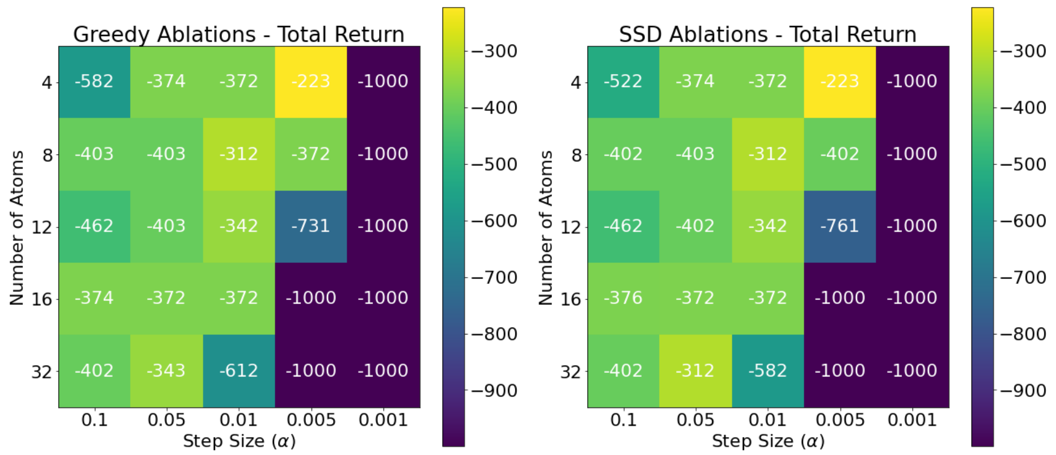


Figure 4.4: Ablations for considered policies under function approximation.

simulated setting, and it permits us to carefully study the proposed algorithm. Results from this data are averaged over fifty seeds, and its standard error bars reflect the random variation in the data due to different behavior actions and reward samples.

A different step size is used for each algorithm and policy. The step size is reduced by half every 10000 steps. Ablations of this along with the number of atoms used to approximate return distributions are shown in Figure 4.4.

The behavior policy β is a discrete distribution over the two actions $\mathcal{A} = \{a_1, a_2\}$. Based on studies with a greedy target policy, the best total performance over one-hundred thousand steps of learning occurred with $\Pr[a_1] = 0.7$. This was held fixed and used for the other policies.

4.3.2 Performance Comparison of Proposed Policies

This experiment addresses the question of whether the proposed algorithm can be applied to solve autonomous navigation problems from high-dimensional sensory streams of unknown structure. The experiment follows the methodology described previously, and it generates experience from the modified Carla environment.

Evaluations all start from state 123. This is significant, because there exist two

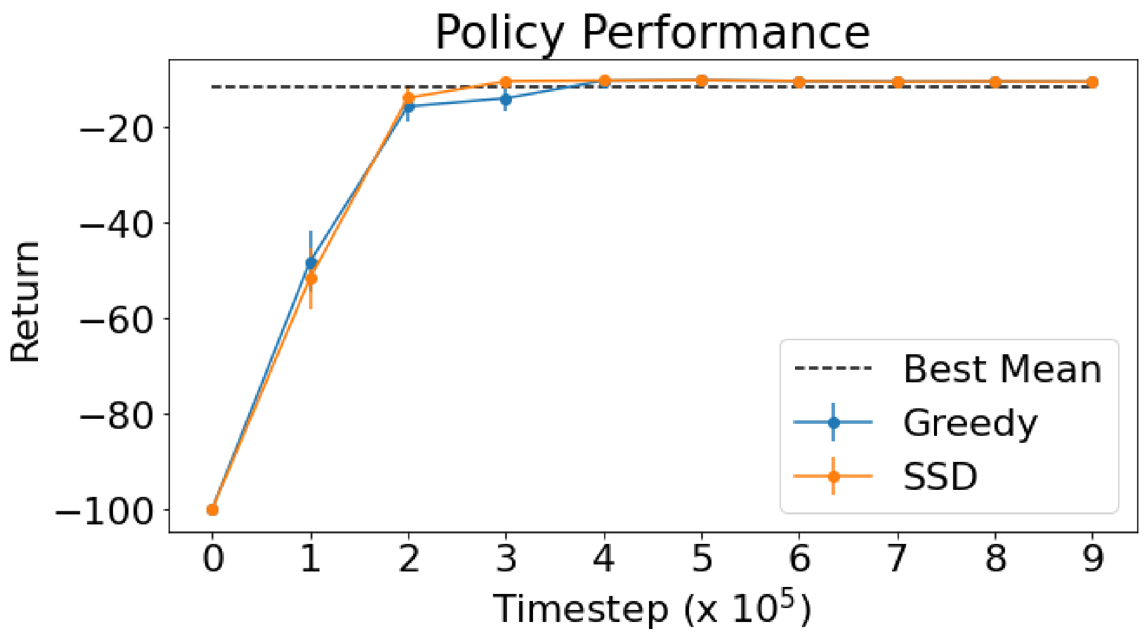


Figure 4.5: Policy performance comparison with proposed algorithm.

paths that reach the goal with equal value. The paths split at state 117; the left path intersects the simulated crosswalk and adds dispersion in the return; the forward path can reach the goal with deterministic rewards in the optimal case. The experiment compares the performance and the selected trajectory using a greedy target policy and the SSD target policy from Chapter 3.

Results are shown in Figure 4.5. As expected, the SSD policy and the greedy policy realize the same average performance. Both policies are able to reach the goal in the minimum number of steps. However, the SSD policy was expected to select the more deterministic path (Figure 4.6), but this was not observed in our data. Despite the increased uncertainty along the path, the deterministic path is infrequently chosen by the SSD policy. This finding motivates our next experiment.

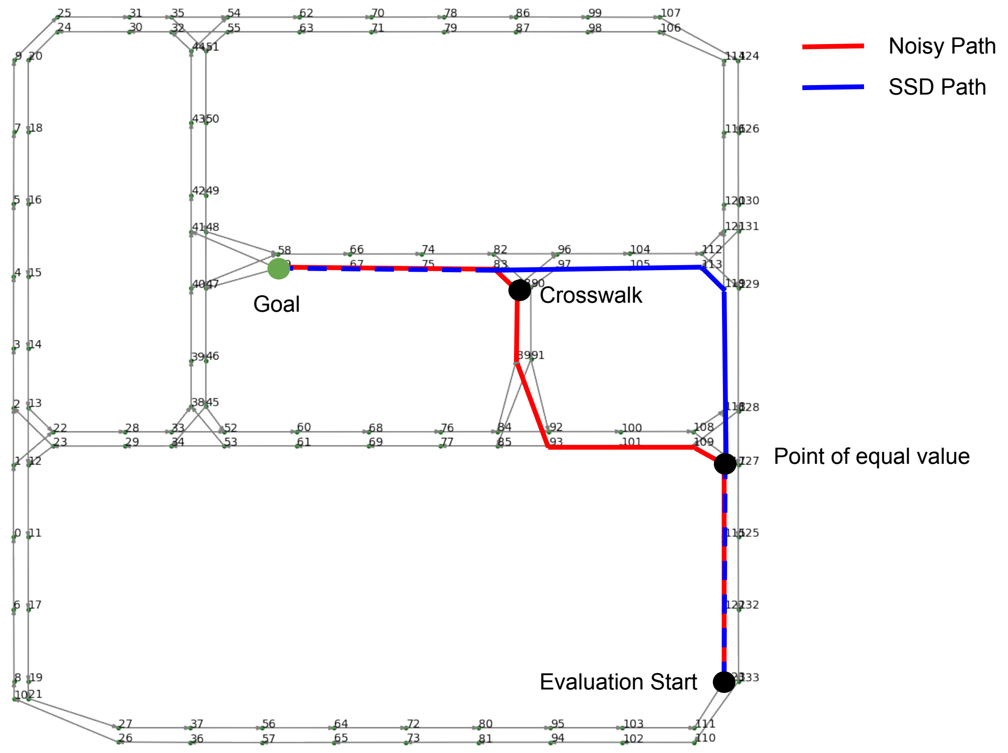


Figure 4.6: **Multiple solution paths:** Two paths of equal value are present in the environment. The noisy path intersects a simulated sidewalk, where the time penalty is random. The other follows a path of deterministic rewards. The SSD policy is expected to choose the deterministic option.

4.3.3 Learning the SSD Policy with Different Representations

This experiment investigates why the proposed algorithm, with an SSD target policy, does not favor the more deterministic path in the waypoint network (Figure 4.6). The hypothesis is that the linear representation is preventing the algorithm from learning an accurate return distribution, and that is producing a high number of false negative comparisons at the waypoint where both solution paths split (117). To investigate this, classification statistics of the proposed algorithm are compared with those of a tabular learner, which uses oracle state information. Both systems learn with the same update rule; they differ only in their representation. If the linear representation is preventing the proposed algorithm from making correct SSD classifications, then its associated true-positive rate will be lower than the tabular baseline.

Indeed, data in Figure 4.7 confirms this hypothesis. The data shows distributions of the true positive rate as box plots. Box plots are shown for each atom of the approximate return distribution learned with the best parameterization according to the ablation study. Each box plot is supported on twenty samples of data computed by drawing five evaluations without replacement from a population of fifty independent evaluations. Note how the error rates are largely uniform over the samples; meaning false classifications do not appear to be from a distinct subset of samples.

To investigate whether this trend held as the number of atoms increased, the experiment was repeated. Data in Figure 4.8 shows classification errors and the performance of both representations using 32 atoms. According to the data, each learner is able to find a useful policy in terms of average reward. However, the function approximation learner is never able to correctly classify SSD with its distribution. Additional results which support this hypothesis were obtained in a modified environment, with lower stochasticity; where the crosswalk affected only one

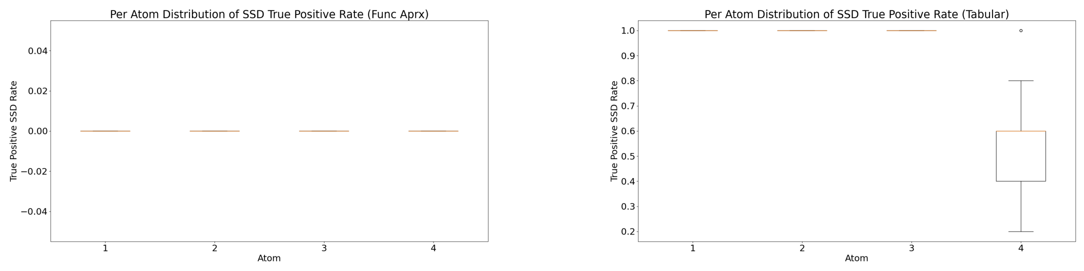
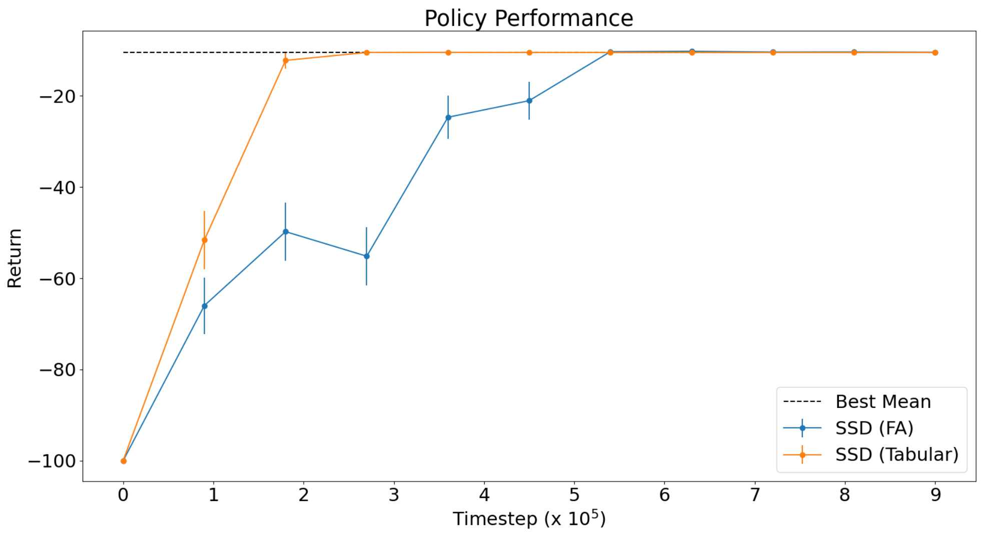


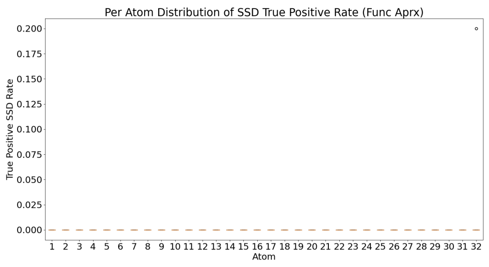
Figure 4.7: **SSD classification accuracy with 4-atom return distributions:** The function approximation learner (left) is unable to correctly classify SSD. The tabular learner (right) has a nearly perfect classification rate, except for its most positive sample, which correctly classifies SSD in only half of the trials.

direction of traffic. These results were unchanged from Figure 4.7, and thus have been omitted for clarity. Even in this simplified setting, the function approximation learner was unable to make correct SSD classifications. All the evidence suggests it may be challenging to approximate an accurate distribution within the data regime of one-hundred thousand steps of experience, using functions of high-dimensional sensory data. And although the learner can find a useful greedy policy, it may require a significant amount of additional experience to improve accuracy well enough for correct SSD classfications. This finding motivates ongoing work that studies different policies to address this issue.

4.4 Discussion

This chapter argued for the use of continual RL for autonomous mobile robot navigation. To this end, the chapter introduced an incremental distributional RL algorithm. The algorithm was shown to learn return distributions from functions of high-dimensional sensory data with unknown structure. The chapter evaluated the algorithm using simulated lidar-based observations with over one-hundred thousand dimensions, suggesting that it could be applied to other robotic systems to autonomously navigate





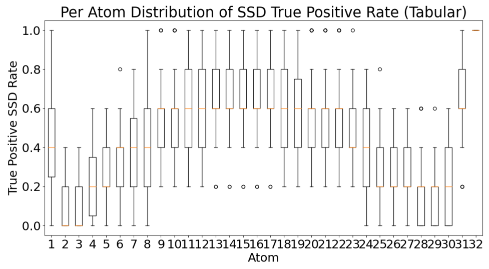


Figure 4.8: **SSD classification accuracy with 32-atom return distributions:** Using more atoms to represent a return distribution makes the learning problem more challenging. However, it also reveals whether the data distribution favors one part of the approximation. The performance plot on top shows that both representations are learning a useful policy that achieves the minimum number of time steps. This makes the representations appear the same. However, their respective classification errors show large differences. Under function approximation, the learner cannot make correct classifications (bottom left). The tabular agent shows that accuracy is biased toward the support points (i.e. atoms) that are sampled most often (bottom right). These come from clipped Gaussian distributed rewards, with likely values in its center and extremes, which consume the probability mass of the tails.

from sensory streams of lidar data. Although the proposed algorithm was able to learn a useful policy, the chapter also revealed a limitation, relating to the system's representation. Additional work is needed to understand how approximation error could be reduced or tolerated within uncertainty aware policies.

Chapter 5
Handling Epistemic Uncertainty of the Expected Return

This chapter considers epistemic uncertainty in the expected return, and it proposes an algorithm that applies to low-data regimes. These settings are particularly relevant to underwater robots whose sensory data comes from acoustic localization beacons. The proposed algorithm uses Gaussian process regression to maintain a predictive posterior over the expected return, giving the learner a way to represent epistemic uncertainty of its value predictions. Using the posterior's covariance, which is a measure of uncertainty, the proposed algorithm employs a maximum-likelihood data compression to consolidate experience into smaller datasets used for online prediction. This improves runtime and data efficiency and enables robots operating in low-data regimes to learn navigation policies with RL. The chapter presents empirical results for several simulated robotic systems and on a physical underwater robot. This provides evidence that these robots can scale to settings that require adaptation.

5.1 Reinforcement Learning in Low-data Regimes

Underwater robots have a unique potential to benefit from reinforcement learning. They operate in complex environments, where transition dynamics are difficult for experts to model. Normally this would motivate the application of model-free algorithms that learn from direct interaction with the environment. However, many underwater robots sense their environment at low rates (e.g. 1Hz), through noisy acoustic positioning sensors which can rapidly drift and require the system to be reset after only a few minutes of operation. This makes RL difficult to apply, since a standard model-free algorithm requires thousands or millions of transitions to learn a useful decision policy.

Despite such challenges, underwater robots could learn to navigate if they had RL algorithms that were effective in low data regimes.

Prior work in RL has addressed the problem of sample efficiency within the context of on-policy evaluation. In particular, Engel et al. (2003) introduced an algorithm based on Gaussian process regression (Rasmussen & Williams, 2005) to improve sample efficiency over conventional TD algorithms. Their algorithm, GP-Sarsa, incorporates experience by forming a posterior over the value function and conditioning on all past transition samples. This can be more efficient than successive applications of TD updates, which discard experience after each step. GP-Sarsa places a posterior distribution over the value function (i.e. the expected return), providing learners with potentially useful information about epistemic uncertainty. GP-Sarsa's main drawback is its prediction complexity, which prohibitively scales as $\mathcal{O}(t^3)$, where t is the number of transitions experienced.

In this work, we ask whether GP-Sarsa can be extended to realistic robotic settings, where updates are applied online with small amounts of data (e.g. $t \approx 1000$). We propose a sparse approximation to reduce the runtime complexity of GP-Sarsa's exact posterior predictions. The proposed approximation replaces the full history of transitions with a smaller set containing m transition samples trained from the full set to provide the same posterior support. This is known as the sparse pseudo-input approximation (Snelson & Ghahramani, 2006). Conditioning on the reduced set, our algorithm achieves the same prediction complexity as the state-of-the-art approximations (Csato & Opper, 2002; Engel et al., 2003, 2005; Jakab & Csato, 2011), while incurring less approximation error.

The contributions of this chapter are as follows

A sparse approximation Although SPGP regression is well-known, it has never been applied to TD estimation, where latent variables exhibit serial correlation.

5.2 On Policy Evaluation with Gaussian Process Regression

We begin by describing the mathematical formalism for GP-Sarsa. The formalism follows a Bayesian methodology to derive an exact posterior distribution over the expected return $Q(s,a)$. First, the random return G_t is expressed as a sum of its mean, $Q(s,a)$, and zero-mean residual, $\Delta Q(s,a) = G_t - Q(s,a)$. This is used to define the residual of a Gaussian reward model:

$$\varepsilon(s,a,s',a') \triangleq \Delta Q(s,a) - \gamma \Delta Q(s',a'), \tag{5.1}$$

$$R = Q(s,a) - \gamma Q(s',a') + \varepsilon(s,a,s',a'). \tag{5.2}$$

To reduce notation, we drop the functional state-action dependence on residuals. Our work makes the simplifying assumption that residuals, ε, are i.i.d random variables from a stationary distribution, $\varepsilon \sim \mathcal{N}(0,\sigma^2)$. After t time steps, the full reward model can be written with matrix notation:

$$\begin{pmatrix} R_2 \\ R_3 \\ \vdots \\ R_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & -\gamma & 0 & \cdots & 0 \\ 0 & 1 & -\gamma & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \cdots & 1 & -\gamma \end{pmatrix} \begin{pmatrix} Q(s_1,a_1) \\ Q(s_2,a_2) \\ \vdots \\ Q(s_{t+1},a_{t+1}) \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_{t+1} \end{pmatrix}. \tag{5.3}$$

We denote the vector of rewards $\mathbf{r} \in \mathbb{R}^t$, the matrix of discounts $\mathbf{\Gamma} \in \mathbb{R}^{t\times(t+1)}$, values $\mathbf{q} \in \mathbb{R}^{t+1}$, and residuals $\boldsymbol{\varepsilon} \in \mathbb{R}^{t+1}$ such that $\mathbf{r} = \mathbf{\Gamma q} + \boldsymbol{\varepsilon}$. Similar to the supervised setting in which GPs are typically applied, we assume $\mathbf{q}$ is a latent function whose

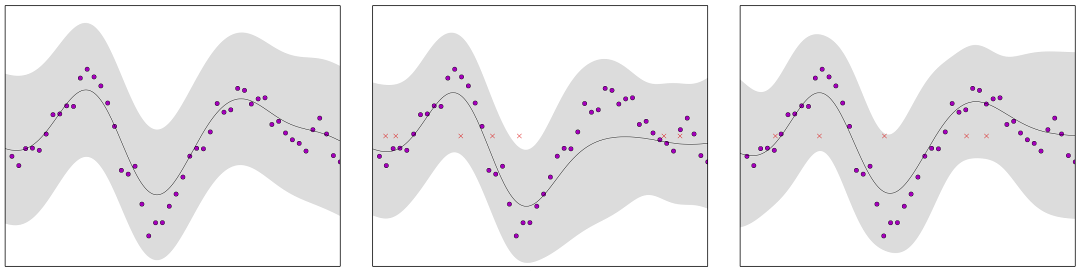


Figure 5.1: **Visualizing the GP-Sarsa posterior:** We plot the predictive distributions of GP-Sarsa (left) and our approximate method, SPGP-Sarsa. We used $m = 7$ randomly-initialized pseudo inputs (red crosses) and $t = 50$ training samples (magenta) taken from the prior. The predictive mean is black, and two standard deviations of uncertainty are shown in gray. We plot SPGP-Sarsa before (center) and after pseudo input optimization (right). After training, the sparse method is nearly identical to the exact method.

outputs are observed through the noisy reward process. However, unlike the supervised setting, the matrix $\mathbf{\Gamma}$ imposes temporal correlations between latent function values. The value function's distribution model considers the covariance matrix $\mathbf{K}_{qq}$ with elements $[\mathbf{K}_{qq}]_{ij} \triangleq k(s_i, a_i, s_j, a_j)$, for $i, j \in \{1, \cdots, t+1\}$: $\mathbf{q} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{qq})$. We also define $\mathbf{K}_{rr} \triangleq \mathbf{\Gamma}\mathbf{K}_{qq}\mathbf{\Gamma}^\top$, and $\mathbf{k}_{r*} \triangleq \mathbf{\Gamma}\mathbf{k}_*$, where $[\mathbf{k}_*]_i \triangleq k(s_i, a_i, s, a)$, and follow conventional notation where subscripts denote dimensionality, e.g. $\mathbf{K}_{qq} \in \mathbb{R}^{|\mathbf{q}|\times|\mathbf{q}|}$.

With a fully Gaussian model, the exact posterior mean and variance are

$$\hat{q}(s, a) = \mathbf{k}_{r*}^\top(\mathbf{K}_{rr} + \sigma^2\mathbf{I})^{-1}\mathbf{r}, \tag{5.4}$$

$$\hat{\sigma}_q^2(s, a) = k(s, a, s, a) - \mathbf{k}_{r*}^\top(\mathbf{K}_{rr} + \sigma^2\mathbf{I})^{-1}\mathbf{k}_{r*}. \tag{5.5}$$

GP-Sarsa approximates the on policy value function with the posterior mean (5.4). This requires an expensive $t \times t$ matrix inversion, costing $\mathcal{O}(t^3)$ floating point operations.

5.3 On Policy RL with Sparse Gaussian Process Regression

To improve the time complexity needed to evaluate a policy, we propose an approximation that induces sparsity in the GP-Sarsa likelihood model (5.3). This is based on Sparse Pseudo-input GP regression (Snelson & Ghahramani, 2006) (Figure 5.1). Our approximation expands the probability space with $m \ll t$ additional pseudo values, denoted $\tilde{\mathbf{q}} \in \mathbb{R}^m$. Here, pseudo values serve as parametric support points, providing probability mass at locations $\mathcal{Z} \triangleq \{\tilde{s}_1, \tilde{a}_1, \cdots, \tilde{s}_m, \tilde{a}_m\}$. These extra latent variables obey the same observation process as $\mathbf{q}$, but without any epistemic uncertainty, since they are treated as hyperparameters and specified as part of the RL problem. Conditioning predictions on the pseudo variables collapses the predictive probability space such that all dense matrix inversions are of rank m. Furthermore, the hyperparameters are optimized to maximize likelihood and improve the approximate value function's fit.

Latent Value Likelihood Model: Given a state-action pair, the likelihood of the value, $Q(s, a)$, is the conditional probability

$$p(Q|s, a, \mathcal{Z}, \tilde{\mathbf{q}}) = \mathcal{N}(Q|\mathbf{k}_{\tilde{q}}^\top \mathbf{K}_{\tilde{q}\tilde{q}}^{-1} \tilde{\mathbf{q}}, k(s, a, s, a) - \mathbf{k}_{\tilde{q}}^\top \mathbf{K}_{\tilde{q}\tilde{q}}^{-1} \mathbf{k}_{\tilde{q}}), \tag{5.6}$$

where $[\mathbf{K}_{\tilde{q}\tilde{q}}]_{ij} \triangleq k(\tilde{s}_i, \tilde{a}_i, \tilde{s}_j, \tilde{a}_j)$, $[\mathbf{k}_{\tilde{q}}]_i \triangleq k(\tilde{s}_i, \tilde{a}_i, s, a)$. The complete likelihood is obtained by stacking the t independent single transition likelihoods over the full history $\mathcal{H} \triangleq \{(s_i, a_i)\}_{i=1}^t$:

$$p(\mathbf{q}|\mathcal{H}, \mathcal{Z}, \tilde{\mathbf{q}}) = \prod_{i=1}^{t} p(Q_t|s_i, a_i, \mathcal{Z}, \tilde{\mathbf{q}}) = \mathcal{N}(\mathbf{g}, \widetilde{\mathbf{K}}); \tag{5.7}$$

where we define $\mathbf{g} \triangleq \mathbf{K}_{q\tilde{q}}\mathbf{K}_{\tilde{q}\tilde{q}}^{-1}\tilde{\mathbf{q}}$, $\widetilde{\mathbf{K}} \triangleq \mathrm{diag}(\mathbf{K}_{qq} - \mathbf{K}_{q\tilde{q}}\mathbf{K}_{\tilde{q}\tilde{q}}^{-1}\mathbf{K}_{\tilde{q}q})$, with $[\mathbf{K}_{q\tilde{q}}]_{ij} \triangleq k(s_i, a_i, \tilde{s}_j, \tilde{a}_j)$.

Conditioned Observation Likelihood Model: Here we derive the likelihood distribution of the observed rewards conditioned on the pseudo values $p(\mathbf{r}|\mathcal{H}, \mathcal{Z}, \tilde{\mathbf{q}})$. Consider the transformed joint distribution over values, rewards, and pseudo values

$$\begin{pmatrix} \mathbf{q} \\ \mathbf{r} \\ \tilde{\mathbf{q}} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{K}_{qq} & \mathbf{K}_{qq}\boldsymbol{\Gamma}^\top & \mathbf{K}_{q\tilde{q}} \\ \boldsymbol{\Gamma}\mathbf{K}_{qq} & \boldsymbol{\Gamma}\mathbf{K}_{qq}\boldsymbol{\Gamma}^\top + \sigma^2\mathbf{I} & \boldsymbol{\Gamma}\mathbf{K}_{\tilde{q}q} \\ \mathbf{K}_{\tilde{q}q} & \mathbf{K}_{\tilde{q}q}\boldsymbol{\Gamma}^\top & \mathbf{K}_{\tilde{q}\tilde{q}} \end{pmatrix}\right).$$

The likelihood is obtained by conditioning $\mathbf{r}$ on $\tilde{\mathbf{q}}$ and invoking transition independence:

$$p(\mathbf{r}|\mathcal{H}, \mathcal{Z}, \tilde{\mathbf{q}}) = \mathcal{N}(\mathbf{K}_{r\tilde{q}}\mathbf{K}_{\tilde{q}\tilde{q}}^{-1}\tilde{\mathbf{q}}, \mathbf{Q} + \sigma^2\mathbf{I}), \qquad \mathbf{Q} \triangleq \mathrm{diag}(\mathbf{K}_{rr} - \mathbf{K}_{r\tilde{q}}\mathbf{K}_{\tilde{q}\tilde{q}}^{-1}\mathbf{K}_{\tilde{q}r}). \tag{5.8}$$

Posterior of Pseudo Values: To obtain the posterior $p(\tilde{\mathbf{q}}|\mathbf{r}, \mathcal{H}, \mathcal{Z})$, we use Bayes' rule. Given the marginal $p(\tilde{\mathbf{q}}|\mathcal{Z}) = \mathcal{N}(\tilde{\mathbf{q}}|\mathbf{0}, \mathbf{K}_{\tilde{q}\tilde{q}})$ and the conditional for $\mathbf{r}$ given $\tilde{\mathbf{q}}$ (5.8), the posterior for $\tilde{\mathbf{q}}$ given $\mathbf{r}$ is

$$\begin{aligned} p(\tilde{\mathbf{q}}|\mathbf{r}, \mathcal{H}, \mathcal{Z}) &= \mathcal{N}(\tilde{\mathbf{q}}|\mathbf{L}\mathbf{K}_{\tilde{q}\tilde{q}}^{-1}\mathbf{K}_{\tilde{q}r}(\mathbf{Q} + \sigma^2\mathbf{I})^{-1}\mathbf{r}, \mathbf{L}); \\ \mathbf{L} &\triangleq \mathbf{K}_{\tilde{q}\tilde{q}}(\mathbf{K}_{\tilde{q}\tilde{q}} + \mathbf{K}_{\tilde{q}r}(\mathbf{Q} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_{r\tilde{q}})^{-1}\mathbf{K}_{\tilde{q}\tilde{q}}. \end{aligned} \tag{5.9}$$

Latent Value Predictive Posterior: The predictive posterior is obtained by marginalizing the pseudo values:

$$p(Q|s, a, \mathbf{r}, \mathcal{H}, \mathcal{Z}) = \int p(Q|s, a, \mathcal{Z}, \tilde{\mathbf{q}})p(\tilde{\mathbf{q}}|\mathbf{r}, \mathcal{H}, \mathcal{Z})d\tilde{\mathbf{q}}.$$

Let $\mathbf{M} \triangleq \mathbf{K}_{\tilde{q}\tilde{q}} + \mathbf{K}_{\tilde{q}r}(\mathbf{Q} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_{r\tilde{q}}$. The new predictive posterior is Gaussian with mean and variance

$$\tilde{q}(s,a) = \mathbf{k}_{\tilde{q}*}^{\top}\mathbf{M}^{-1}\mathbf{K}_{\tilde{q}r}(\mathbf{Q} + \sigma^2\mathbf{I})^{-1}\mathbf{r}, \tag{5.10}$$

$$\tilde{\sigma}_q^2(s,a) = k(s,a,s,a) - \mathbf{k}_{\tilde{q}*}^{\top}(\mathbf{K}_{\tilde{q}\tilde{q}}^{-1} - \mathbf{M}^{-1})\mathbf{k}_{\tilde{q}*}. \tag{5.11}$$

Equations 5.10 and 5.11 represent our main contribution. The predictions depend on two matrix inversions. The first is the dense m-rank matrix, $\mathbf{K}_{\tilde{q}\tilde{q}}$. The second is the t-rank diagonal matrix $(\mathbf{Q} + \sigma^2\mathbf{I})$. When $m \ll t$, both matrices are easier to invert than a dense t-rank matrix. Thus, (5.10) provides motivation for estimating and predicting latent values efficiently.

5.3.1 Policy Evaluation with Sparse GP-Sarsa

We now turn to the question of how a learning system could apply our sparse approximation to evaluate a policy. We introduce SPGP-Sarsa (Algorithm 6). Given a transition sample along with the history of observations and pseudo inputs, the algorithm incorporates new experience into $\mathbf{r}$, $\mathcal{H}$, and the matrices used to represent the predictive mean and variance. Algorithm 6 returns the updated representations such that they can be used to approximate the expected value function and its predictive variance for any state-action pair. With an abuse of notation, the moment representations are denoted $\tilde{q}(s,a)$, and $\tilde{\sigma}_q^2(s,a)$.

Algorithm 6 SPGP-Sarsa

```
1: input $s, a, r, s', a'$, $\mathbf{r}$, $\mathcal{H}$
2: Hyperparameters: $\boldsymbol{\Theta}$
3: $\mathbf{r} \leftarrow \text{concatenate}(\mathbf{r}, r)$
4: $\mathcal{H} \leftarrow \mathcal{H} \cup \{(s', a')\}$
5: Update representation of $\tilde{q}(s, a)$ from (5.10).
6: Update representation of $\tilde{\sigma}_q^2(s, a)$ from (5.11).
7: return $\tilde{q}(s, a)$, $\tilde{\sigma}_q^2(s, a)$, $\mathbf{r}$, $\mathcal{H}$
```

5.3.2 Optimizing Hyperparameters

Hyperparameters are fit to the observed data by maximizing the marginal likelihood. Hyperparameters include kernel parameters $\boldsymbol{\theta}$, aleatoric variance σ^2, and pseudo inputs $\mathcal{Z}$. The full set is defined to be $\boldsymbol{\Theta} \triangleq \{\boldsymbol{\theta}, \sigma^2, \mathcal{Z}\}$. Unlike prior sparse approximations which were based on rejection sampling, our approximation has the benefit of being differentiable in its hyperparameters, thereby permitting them to be precisely tuned with gradient-based optimization. The marginal likelihood is Gaussian and given by

$$p(\mathbf{r}|\mathcal{H}, \mathcal{Z}, \boldsymbol{\Theta}) = \int p(\mathbf{r}|\mathcal{H}, \mathcal{Z}, \tilde{\mathbf{q}})p(\tilde{\mathbf{q}}|\mathcal{Z})d\tilde{\mathbf{q}} = \mathcal{N}(\mathbf{r}|\mathbf{0}, \mathbf{Q} + \sigma^2\mathbf{I} + \mathbf{K}_{r\tilde{q}}\mathbf{K}_{\tilde{q}\tilde{q}}^{-1}\mathbf{K}_{\tilde{q}r}). \quad (5.12)$$

Instead of maximizing (6.9) directly, we use its logarithm, $\mathcal{L}(\boldsymbol{\Theta}) \triangleq \log p(\mathbf{r}|\mathcal{H}, \mathcal{Z}, \boldsymbol{\Theta})$. Full details of the gradient computation are provided in the appendix.

Overfitting is an important consideration when optimizing hyperparameters of GP models. This occurs when the predictive variance collapses around the training data, and it can weaken the representation's ability to generalize to unexperienced settings. Overfitting can be prevented by adding a regularization term to $\mathcal{L}(\boldsymbol{\Theta})$ and penalize parameters with high magnitude.

5.3.3 Policy Improvement with Sparse GP-Sarsa

Defining the approximate value function to be the predictive mean $\hat{q}(s,a) = \tilde{q}(s,a)$, Algorithm 7 is can be used to update a policy. A learner gathers experience online and updates its representation of the predictive moments. These are subsequently used to update the policy with the current representation. For instance, a robot learning to navigate may consider the ε-greedy policy that returns the action maximizing the predictive mean with probability $1 - \varepsilon$.

Algorithm 7 On-policy Improvement with SPGP-Sarsa

```
1: for $t = 1, 2, \cdots T$ do
2:     Observe $s, a, r, s', a'$ from $p(\cdot|s,a)$ and $\pi(\cdot)$
3:     $\hat{q}(s,a), \hat{\sigma}_q^2(s,a), \mathbf{r}, \mathcal{H} \leftarrow$SPGP-Sarsa$(s, a, r, s', a', \mathbf{r}, \mathcal{H}, \mathcal{Z})$.
4:     Update policy with $\hat{q}(s,a)$ and $\hat{\sigma}_q^2(s,a)$.
5:     if $t \mod T_{\text{fit}} = 0$ then
6:         $\mathbf{\Theta} \leftarrow \arg\max_{\mathbf{\Theta}'} \mathcal{L}(\mathbf{\Theta}')$
```

Our proposed RL algorithm interleaves hyperparameter optimization into the online learning process. In comparison to GP-Sara, which fits $|\mathbf{\Theta}|$ parameters, our proposed algorithm fits an additional m d-dimensional variables. To reduce the computational burden of tuning these variables, our algorithm only optimizes every T_{fit} steps The frequency which optimization will be useful depends on how well $\mathbf{H}$ and $\mathcal{Z}$ support the posterior distribution.

5.4 Related Work

GP regression has been applied to RL in several contexts. Algorithms based on GP-Sarsa are most relevant to our work (Engel et al., 2003, 2005; Engel, 2005; Engel et al., 2006). Engel et al. (2006) uses this to control a set of high-dimensional manipulators, which emulate an octopus arm. Others have applied GP regression to the value function while ignoring sequential correlation of the observation process (Deisenroth et al., 2009; Rasmussen & Kuss, 2004).

Prior approximations to the GP posterior used rejection sampling (Csato & Opper, 2002; Engel et al., 2003, 2005; Jakab & Csato, 2011) so their predictions never cost more than tm^2 operations, where $m \ll t$. Besides discarding potentially useful information, rejection sampling can interfere with hyperparameter optimization by causing sharp changes to appear in the marginal likelihood. Unlike these methods, our approximation is continuous in the hyperparameters, which makes it amenable to gradient-based optimization.

The state-of-the-art approximation of the GP-Sarsa posterior uses a low-rank approximation to the full covariance matrix: $\mathbf{K}_{qq} \approx \mathbf{A}\tilde{\mathbf{K}}^{-1}\mathbf{A}^\top$, where $\mathbf{A}$ is a projection matrix (Engel et al., 2005). Before adding new data to the active set, Low Rank GP-Sarsa checks if new samples increase the conditional covariance by a desired error threshold, ν.

5.5 Empirical Results

Here, we empirically verify several prior claims. The first is that SPGP-Sarsa permits data-efficient policy evaluation while learning to navigate. Furthermore, we test whether our proposed algorithm for policy improvement can be applied to a physical underwater robot. All experiments use the covariance function:

$$k(s_i, a_i, s_j, a_j) = \sigma_f \exp\left[-0.5(\Delta^\top \boldsymbol{\ell} \Delta)\right],$$

where $\Delta \triangleq (s_i - s_j, a_i - a_j)$, and $\boldsymbol{\ell}$ is a diagonal matrix of length scales.

5.5.1 Approximation Quality of Sparse GP-Sarsa

In Figure 5.2, the approximation quality of SPGP-Sarsa is examined as a function of the number of pseudo inputs m. Approximation quality is measured with the ratio of sparse-to-complete log-likelihood, $\mathcal{L}_{\text{sparse}}(m)/\mathcal{L}_{\text{exact}}$. As m increases, and the pseudo inputs are placed at the maximum likelihood locations, they begin to coincide with observed history $\mathcal{H}$. Although more pseudo inputs improve the approximation and eventually match the exact predictive posterior, the approximation starts to lose its computational benefit.

5.5.2 Approximation Adjustability of Low-rank GP-Sarsa

In Figure 5.4 the approximation adjustability is recorded as the amount of experience accepted by the algorithm as a function of the error parameter ν which tunes the algorithm's sparsity. Results show the adjustable range can be limited to only extreme amounts of data retention or rejection.

Retaining an excessive amount of data reduces the computational benefit of the low-rank approximation. Conversely, rejecting too much data is counterproductive when very little arrives, particularly in an underwater robotics setting.

5.5.3 Learning Navigation Policies in Simulation

This experiment considers the problem of learning to control a simulated robot to a goal point. It compares learning performance for three environments, whose dynamics model the classic Mountain Car (Sutton & Barto, 2018), a planar autonomous surface vehicle (ASV), and an autonomous underwater robot (AUV). We ask whether our proposed sparse approximation can realize the same asymptotic performance as the exact solution, and we additionally compare this to Low-rank GP-Sarsa, the best

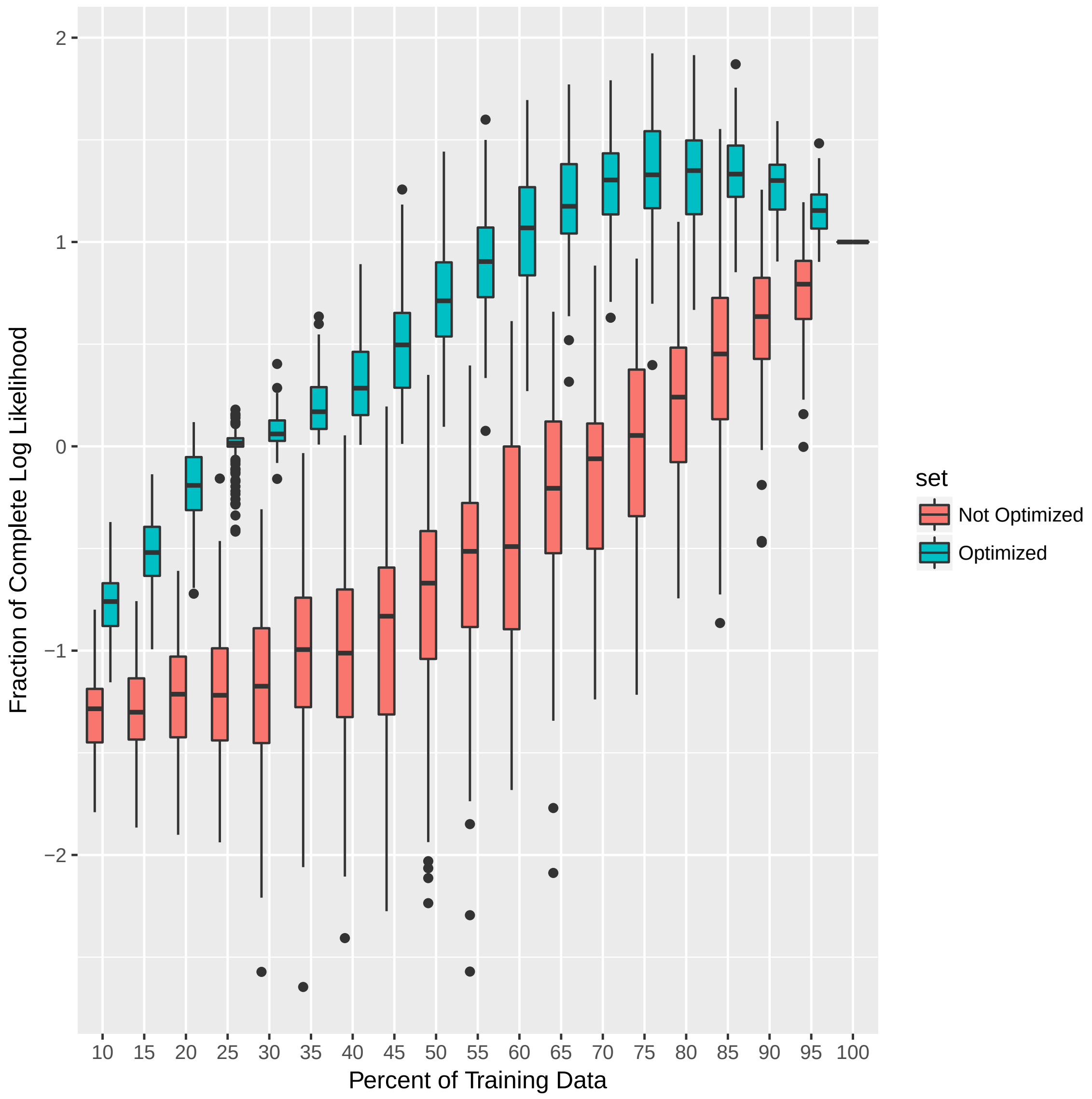


Figure 5.2: SPGP-Sarsa likelihood disparity

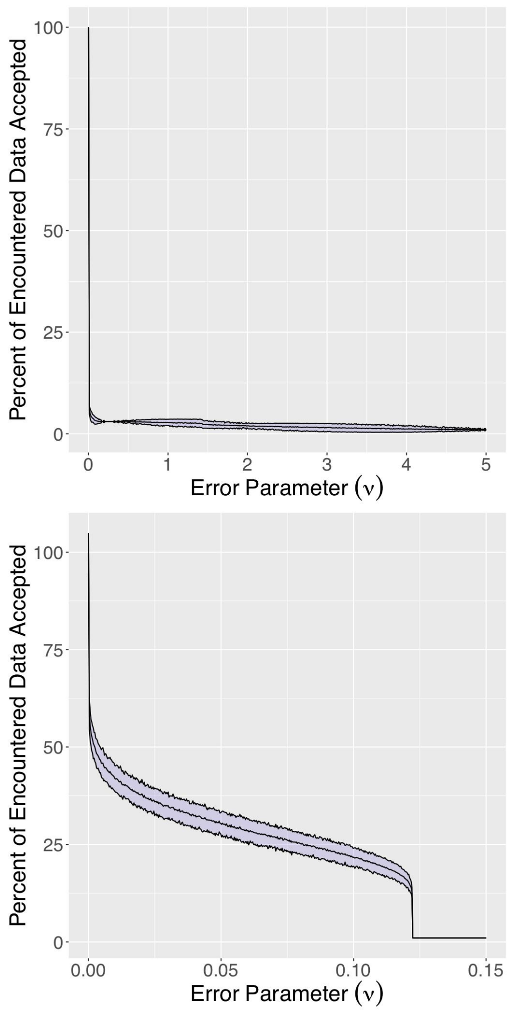


Figure 5.3: Low Rank Sarsa rejection statistics

Figure 5.4: **Visualizing the effect of parameter changes on approximation quality:** Figure 5.2 shows how the inclusion of more pseudo-inputs improves approximation quality, and how optimizing their locations is additionally beneficial. We fixed a synthetic dataset of 100 samples from the prior, $\mathcal{N}(\mathbf{0}, \mathbf{K}_{qq})$. As a reflection of quality, we uniformly sampled subsets of the data, computed the log likelihood with SPGP-Sarsa, both before and after optimization, and normalized it by the log likelihood of the full set. Boxplots were computed for 100 random subsets. Figure 5.3 shows how the low-rank method cannot always induce sparsity. We vary the error threshold, ν, and plot the percentage of samples Low Rank Sarsa retained from 100 random trajectories. Trajectories from the Mountain Car system (top), and prior, $\mathcal{N}(\mathbf{0}, \mathbf{K}_{qq})$ (bottom) are shown.

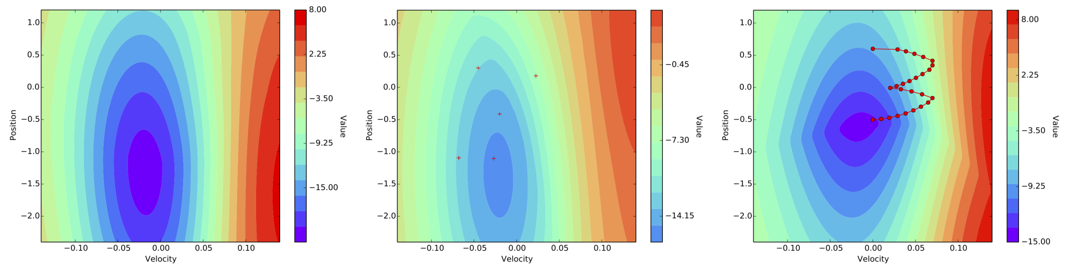
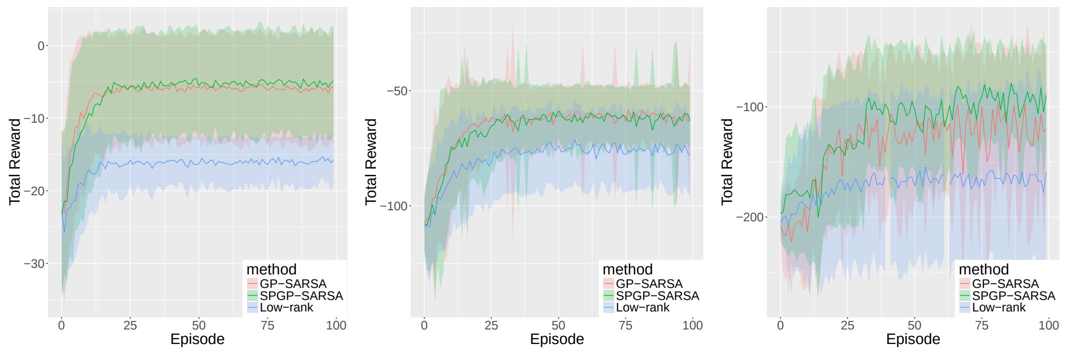


Figure 5.5: **SPGP-Sarsa replicates value functions with a small active set:** Using the same covariance parameters, we plot value predictions from GP-Sarsa (left) and SPGP-Sarsa before (center) and after (right) pseudo-input optimization. With only a 5×5 matrix inversion, SPGP-Sarsa nearly replicates the true value landscape, while GP-Sarsa used a 302×302 inversion. The pseudo inputs (red crosses) moved beyond plot boundaries after optimization. A sample trajectory is shown at right.



(a) Mountain Car (b) ASV (c) AUV

Figure 5.6: **Learning performance during policy improvement:** We plot average total reward over 100 episodes, along with one standard deviation. In each case, SPGP-Sarsa performs as well as GP-Sarsa; Low Rank Sarsa is the least optimal, and all policies exhibit considerable variance.

rejection-based sparse approximation.

Data in Figure 5.6 provide two pieces of information. All the algorithms converge with a relatively small amount of data: in around fifty episodes with . These results are consistent with prior work by Engel *et. al*(Engel et al., 2006), where GP-Sarsa was applied to control a high dimensional octopus arm having 88 continuous state variables and 6 action variables. Performance differences between the two approximate methods are due to their approximation quality (Figure 5.4). As expected, SPGP-Sarsa is able to learn on par with the exact method, because it can replicate the predictive posterior better than Low Rank Sarsa.

The experiment was designed with several aspects held fixed. First, the learner used an ε-greedy policy with $\varepsilon = 0.1$ to experience the environment in 100 episodes. Hyperparameters that control the degree of sparsity between SPGP-Sarsa and Low Rank GP-Sarsa were both set to induce fifty percent sparsity over the maximum amount of data the learner could experience – allowing for a fair comparison between both algorithms. Finally, the pseudo inputs were initialized randomly.

Mountain Car: The Mountain Car is a canonical RL problem, where an underpowered dynamical system must learn to exploit momentum to reach the crest of a hill. The environment state is given by a position and velocity, $(x, \dot{x})$, where $x \in [-1.2, 0.6]$, and $\dot{x} \in [-0.07, 0.07]$. Rewards are $R = \epsilon - x$, with $\epsilon \sim \mathcal{N}(0, 0.001)$, until the goal is reached, where $R = 1$. Episodes start at $(-0.5, 0.0)$, and the goal is located at 0.6. We let $m = 5$, $\nu = 0.1$, and learning evolve over 50 transitions.

Autonomous Surface Vehicle The second environment simulates a planar ASV, which has been considered in prior RL research (Ghavamzadeh et al., 2016; **?**). The robot must navigate within 10m of (50m, 50m) using 100 transitions. States contain position x, y, heading θ, and heading rate $\dot{\theta}$. The speed is held constant at $V = 3$ m/s, and the angular rate $\omega \in [-15°/\text{s}, 15°/\text{s}]$ controls the robot through the equations of motion

$$x_{t+1} = x_t + \Delta V \cos\theta_t, \qquad y_{t+1} = y_t + \Delta V \sin\theta_t,$$
$$\theta_{t+1} = \theta_t + \Delta\dot{\theta}_t, \qquad \dot{\theta}_t = \dot{\theta}_t + \frac{\Delta}{T}(\omega_t - \dot{\theta}_t).$$

We use time steps of $\Delta = 1.0$s, and a $T = 3$ step time delay for the command ω to be realized. The delay models resistance of surface currents and actuator limitations. Rewards are assigned with $R = R_{\text{min}} - (R_{\text{goal}} - R_{\text{min}})\exp(-d/\delta) + \epsilon$, where $R_{\text{min}} = -1.0$, $R_{\text{goal}} = 10.0$, ϵ as before, $d = ||\mathbf{x}_{\text{goal}} - \mathbf{x}||$, and $\delta = 10$. The policy was linear: $\omega = K_\omega e_\theta$, where $e_\theta = \arctan[(50 - y)/(50 - x)] - \theta$. K_ω was updated with a line search, to maximize the first moment of the posterior. We selected $\nu = 0.1$ and $m = 50$.

Autonomous Underwater Vehicle For the third system we consider a common AUV design with differential control. Actions command forward acceleration v and turn rate ω through port and starboard actions. The dynamics are an extension of the ASV with the additional dimension, $V_{t+1} = V_t + \Delta v_t$. The policy uses Fourier basis functions, with e_θ defined as before, and $e_r = ||\mathbf{x}_{\text{goal}} - \mathbf{x}||$:

$$v = K_r e_r \cos(e_\theta), \qquad \omega = K_r \cos(e_\theta)\sin(e_\theta) + K_\theta e_\theta.$$

These map to actions through $v_t = K_v(a_{\text{port},t} + a_{\text{star},t})$ and $\omega_t = K_w(a_{\text{port},t} - a_{\text{star},t})$, where we let $K_v = K_w = 1$. Policy updates select parameters K_r and K_θ to maximize the value posterior mean, found through a 100×100 grid search. Learning occurs over 100 episodes of 200 transitions, with $\nu = 5$ and $m = 50$.

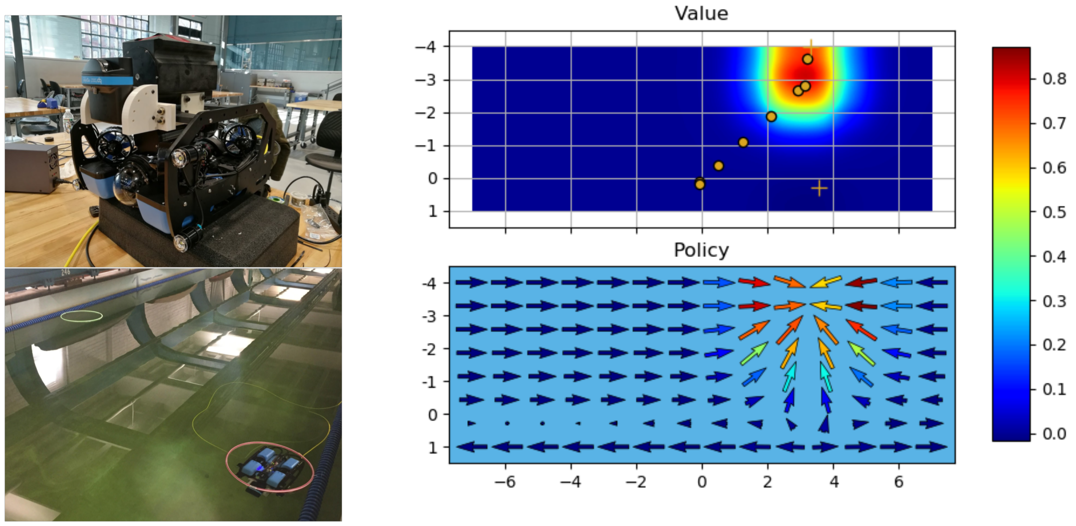


Figure 5.7: **Underwater robot navigation:** The robot (top-left) learns to navigate between two rings (bottom-left). With only a demonstration of eight transitions, SPGP-Sarsa recreates a near-optimal value (top-right) and policy (bottom-right). Robot trajectory and the pseudo inputs are plotted in gold.

Results: Results in Figure 5.6 show that SPGP-Sarsa achieves the same asymptotic performance as the exact algorithm, GP-Sarsa, and it outperforms the alternative sparse approximation which is based on rejection sampling.

5.5.4 Learning to Navigate on a Physical Underwater Robot

The final investigation applies SPGP-Sarsa to a physical BlueROV robot using observations of planar position and velocity. The purpose is to verify whether results from our simulation study transfer to a physical domain, where the robot learned to navigate between two waypoints.

The robot's sensory information comes from a Doppler velocity log, whose estimates drift on the order of 1m every 1-2 min. This bounded the number of transitions the agent could experience, before its sensory information turned into noise and, furthermore, it made learning over multiple episodes impractical. Disturbances from a data tether introduced additional noise in the robot's starting position, which sometimes made the robot drift in all degrees of freedom. The robot remained at a fixed depth throughout learning.

The robot's speed was constrained to facilitate accurate localization. Its actions were defined with a finite number of pulse-width-modulated commands to its thrusters. In particular, it was allowed to yaw and translate forward and backward for a fixed amount of time.

Results: Figure 5.7 shows the value function and greedy policy learned after only eight transitions. As the highest value is concentrated around the goal, we conclude that SPGP-Sarsa was able to recover a useful value function and policy. The experiment was repeated twenty times and the average policy update time took $0.013 \pm 7 \cdot 10^{-4}$s with a 2.8GHz i7 processor. We achieve only a modest improvement in prediction time, since $t = 8$, and we use $m = 2$ pseudo inputs. Despite this fact, our results confirm that efficient robot learning is possible with SPGP-Sarsa.

5.6 Discussion

There are several key assumptions underpinning our results. To guarantee the likelihood can be factored, we needed to assume that transitions are uncorrelated. Had we modeled serial correlation in the noise process, $\boldsymbol{\varepsilon}$ would be distributed with a tridiagonal covariance (Engel et al., 2005), which cannot be factored directly. It is possible to obtain a factorable model by applying a whitening transform. However, this changes the observation process to Monte Carlo samples, which are known to be noisy (Sutton & Barto, 1998). Under our simpler set of assumptions, we obtain an efficiently-computable, sparse representation of the value posterior that is amenable to smooth evidence maximization.

This work presented an algorithm that supports learning navigation policies in low-data regimes. We considered the class of GP-Sarsa algorithms. These algorithms can be more data-efficient than applying the standard Bellman operator, because GP-Sarsa updates are based on probabilistic conditioning, which incorporates new experience in a single step. SPGP-Sarsa was proposed as a sparse approximation to GP-Sarsa, which can be more accurate than rejection-based approximations. Our proposed algorithm was shown to be effective in simulation and on a physical marine robot with a highly-constrained sensory system. Despite these challenges, SPGP-Sarsa was still able to learn a useful value function. In closing, we believe our results highlight the efficiency of SPGP-Sarsa as a marine robot learning method.

Chapter 6

Handling Heteroscedastic Uncertainty in the Observations

Model-based algorithms for reinforcement learning can provide a data-efficient way to learn control policies for robotic systems (Deisenroth & Rasmussen, 2011; Hester et al., 2010; Ko et al., 2007). These algorithm learn a transition model of the environment to predict the next state and reward, then use the model to perform simulation-based policy evaluation and updates (Hester & Stone, 2011). Model-based algorithms can sometimes be more data efficient than their model-free counter parts, by reducing the number of costly physical learning experiences. Model-based algorithms have proven effective in environments that exhibit constant noise properties. However, they have never been extended to environments for which the aleatoric uncertainty is *heteroscedastic*, i.e. its noise varies throughout the state-action space. Many sensors used for underwater robot localization exhibit heteroscedastic noise (Figure 6.1). Failing to capture these effects in the RL process may result in inaccurate policy evaluation and poor policy updates.

To address these issues, this chapter introduces a new transition model to represent heteroscedastic noise using hierarchal Gaussian process regression. We show how to use the algorithm for indirect policy evaluation, whereby samples of the return are obtained with rollouts from the model. The framework is applied to three environments, where a simulated robot needs to navigate to a goal point. Our results demonstrate a significant performance improvement and reduction in model bias when using our framework over a standard model that assumes constant noise. Furthermore, our results indicate that model learning can be done efficiently, using reasonable amounts of data in short-horizon settings.

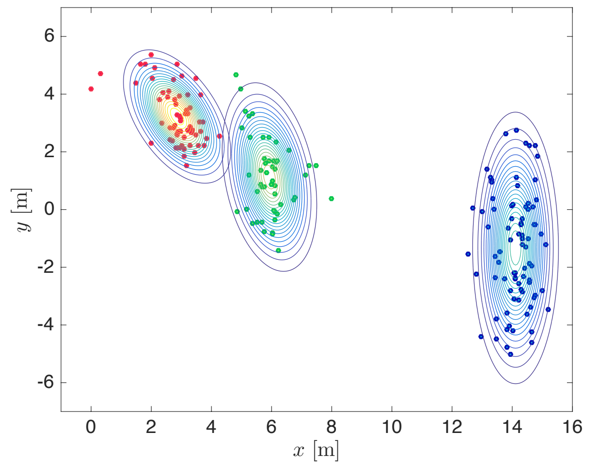


Figure 6.1: **Heteroscedastic uncertainty from underwater localization sensor:** Position samples from an ultra-short baseline (USBL) acoustic positioning sensor are shown. The sample's coordinates are relative to the sensor. The data reflects the position of an underwater robot holding position at three different locations in a shallow-water environment.

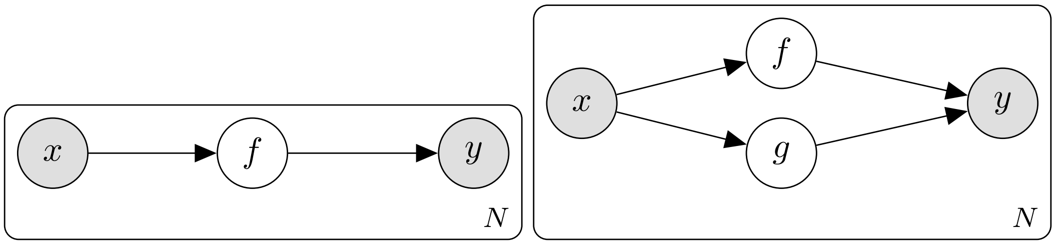


Figure 6.2: Graphical models for a standard (left) and heteroscedastic (right) Gaussian Process.

6.1 Transition Models from Hierarchal Gaussian Process Regression

Heteroscedastic models consider input-dependent noise. Suppose $\mathbf{x}$ is some vector of inputs, and y is a corresponding output from some observation process. Then heteroscedastic models can follow the form $y = f(\mathbf{x}) + \varepsilon(\mathbf{x})$, where $\varepsilon(\mathbf{x}) \sim \mathcal{N}(0, \exp(g(\mathbf{x})))$. Here, there are two latent functions, f and g, to describe the output dynamics and change in observation noise respectively. Figure 6.2 illustrates the model. By placing a Gaussian process prior over the two latent functions and using a maximum likelihood assumption on the noise variables, one can derive equations for the predictive moments of y (Goldberg et al., 1998). In what follows, we describe how to apply this model to the transition dynamics of a Markov decision process, to predict the next observation vector.

6.1.1 First-level Observation Model

This work considers distributional transition models whose moments are used to generate expected samples of experience. The model places a Gaussian process prior over a latent, nonlinear function that generates the next observation $\mathbf{o}_{t+1} = \mathbf{f}(\mathbf{o}_t, \mathbf{a}_t)$, where $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_f)$. The learner trains this model using samples[1] $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})$ from its history of experience $\mathcal{H} \triangleq \{(\mathbf{o}_t, \mathbf{a}_t)\}_{t=1}^{T}$, up to time $T \in \mathbb{N}$. Since observations are multi-dimensional, the learner considers n such models $o_{t+1}^{i} = f_i(\mathbf{o}_t, \mathbf{a}_t)$, for each component $i = 1, \cdots, n$ of $\mathbf{o}_{t+1} \triangleq (o_{t+1}^{1}, \cdots, o_{t+1}^{n})^{\top}$.

Predictions of f_i are influenced by two sources of noise. Constant transition noise is modeled with $\mathbf{K}_{f_i} \in \mathbb{R}^{T \times T}$, where its elements are defined as $[\mathbf{K}_{f_i}]_{lj} \triangleq k_f(\mathbf{o}_l, \mathbf{a}_l, \mathbf{o}_j, \mathbf{a}_j)$, for $l, j \in \{1, \cdots, T\}$. The heteroscedastic noise is modeled with a diagonal matrix $\mathbf{K}_{g_i} \triangleq \operatorname{diag}(\mathbf{k}_{g_i})$, whose elements are the exponentiated predic-

[1]This work applies to robotics problems where the reward function is given to the learner.

tive mean of an additional latent function $k_{g_i}(\mathbf{o}_t, \mathbf{a}_t) \triangleq \exp(g_i(\mathbf{o}_t, \mathbf{a}_t))$, and $\mathbf{k}_{g_i} \triangleq (k_{g_i}(\mathbf{o}_1, \mathbf{a}_1), \cdots, k_{g_i}(\mathbf{o}_T, \mathbf{a}_T))^\top$. This model structure was originally proposed by Goldberg et al. (1998).

Given the covariances and a vector of outcomes for the i-th observation $\mathbf{o}'_i \triangleq (o^i_2, \cdots, o^i_{T+1})^\top$, the learner predicts the next observation using the predictive moments of the posterior distribution over $\mathbf{f}$. Each component model has predictive moments

$$f_i(\mathbf{o}, \mathbf{a}) = \mathbf{k}_{f_i}^\top (\mathbf{K}_{f_i} + \mathbf{K}_{g_i})^{-1} \mathbf{o}'_i, \tag{6.1}$$

$$\sigma^2_{f_i}(\mathbf{o}, \mathbf{a}) = k_{f_i}(\mathbf{o}, \mathbf{a}, \mathbf{o}, \mathbf{a}) + k_{g_i}(\mathbf{o}, \mathbf{a}) - \mathbf{k}_{f_i}^\top (\mathbf{K}_{f_i} + \mathbf{K}_{g_i})^{-1} \mathbf{k}_{f_i}. \tag{6.2}$$

Here, we define $[\mathbf{k}_{f_i}]_t \triangleq k_{f_i}(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}, \mathbf{a})$ for $t = 1, \cdots, T$.

To compute the predictive posterior over $\mathbf{f}$, we condition on the history of observation-action pairs $\mathcal{H}$ and $\mathbf{o}'_i$, then we typically marginalize any additional latent variables. Candidates for marginalization in this model are the noise variables $\mathbf{k}_{g_i}$ predicted from previous experience and a prediction $k_{g_i^*} \triangleq k_{g_i}(\mathbf{o}, \mathbf{a})$, for some inputs:

$$p(\mathbf{f}|\mathbf{o}, \mathbf{a}, \mathcal{H}, \mathbf{o}'_i) = \int p(\mathbf{f}|\mathbf{o}, \mathbf{a}, \mathcal{H}, \mathbf{o}'_i, k_{g_i^*}, , \mathbf{k}_{g_i}) p(k_{g_i^*}, \mathbf{k}_{g_i}|\mathbf{o}, \mathbf{a}, \mathcal{H}, \mathbf{o}'_i) dk_{g_i^*} d\mathbf{k}_{g_i}. \tag{6.3}$$

Unfortunately, marginalization is not analytically possible with this model. The presence of the exponential on $g_i(\mathbf{o}, \mathbf{a})$ causes the distribution to become non-Gaussian.

To obtain a solution, we must appeal to an approximation. Previous approximations to this have included MCMC (Goldberg et al., 1998), variational inference (Saul et al., 2016; Titsias & Lázaro-Gredilla, 2011), Laplace approximation (Vanhatalo et al., 2013), expectation propagation (Hernández-lobato et al., 2014), and maximum likelihood (Kersting et al., 2007). In robotics, the uncertainty of a sensor is usually concentrated around its mean. Therefore, a maximum likelihood assumption can provide a fair approximation of the true posterior in these cases.

6.1.2 Second-level Heteroscedastic Noise Model

We proceed under the maximum likelihood assumption described by Kersting et al. (2007), where the approximate posterior is defined by

$$p(\mathbf{f}|\mathbf{o}, \mathbf{a}, \mathcal{H}, \mathbf{o}') \approx p(\mathbf{f}|\mathbf{o}, \mathbf{a}, \mathcal{H}, \mathbf{o}'_i, k^*_{g^*_i}, , \mathbf{k}^*_{g_i}), \tag{6.4}$$

$$(k^*_{g^*_i}, , \mathbf{k}^*_{g_i}) \triangleq \underset{k_{g^*_i}, , \mathbf{k}_{g_i}}{\arg\max}\, p(k_{g^*_i}, \mathbf{k}_{g_i}|\mathbf{o}, \mathbf{a}, \mathcal{H}, \mathbf{o}'_i). \tag{6.5}$$

Here, the latent noise variables are replaced by their maximum-likelihood realizations. To compute the predictive posterior, we condition $\mathbf{g}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{h_i})$ on the history $\mathcal{H}$ and noise observations $\mathbf{z}_i$ using the covariance $\mathbf{K}_{h_i}$, and without considering any additional aleatoric uncertainty. In practice, however, adding noise to $\mathbf{K}_{h_i}$ can increase numerical stability during inversion. The predictive moments of the noise model are

$$g_i(\mathbf{o}, \mathbf{a}) = \mathbf{k}_{h_i}^\top \mathbf{K}_{h_i}^{-1} \mathbf{z}_i, \tag{6.6}$$

$$\sigma^2_{g_i}(\mathbf{o}, \mathbf{a}) = k_{h_i}(\mathbf{o}, \mathbf{a}, \mathbf{o}, \mathbf{a}) - \mathbf{k}_{h_i}^\top \mathbf{K}_{h_i}^{-1} \mathbf{k}_{h_i}. \tag{6.7}$$

In Equation 6.6, the log variances $\mathbf{z}_i$ are treated as observed variables. However, variance is a latent variable in our setting which needs be estimated from data. To obtain estimates, we use the empirical variance with samples from the posterior over $\mathbf{f}$ (Kersting et al., 2007). First, we draw M posterior samples, assuming that each draw is an independent observation of the noise-free observation process. Then using the noisy targets $\mathbf{o}'_i$, the t-th variance observation is estimated with

$$z_i^t \approx \log\left(\frac{1}{2M}\sum_{j=1}^{M}(o_i^t - f_i^j)^2\right). \tag{6.8}$$

This implicitly assumes that f_i^j and o_i^t are observed with equal likelihood for $t = 1, \cdots, T$. For large enough M, this estimate will minimize the average distance between the predictive distribution and the observations $\mathbf{o}'_i$.

6.1.3 Optimizing Hyperparameters

As the learner experiences new transitions, it updates the transition model by optimizing its internal hyperparameters $\boldsymbol{\Theta}$ (Hester & Stone, 2011). The model's hyperparameters include those in its covariance functions, k_{f_i} and k_{g_i}.

For both the observation and noise process, the model uses an exponentiated quadratic covariance function:

$$k(\mathbf{x}, \mathbf{x}') \triangleq \sigma^2 \exp\left[-0.5(\mathbf{x} - \mathbf{x}')^\top \mathbf{L}(\mathbf{x} - \mathbf{x}')\right].$$

Here, $\mathbf{L}$ is a diagonal matrix with elements $\ell_{f,1}^{-2}, \cdots, \ell_{f,n}^{-2}$. Each $\ell_{f,i}$ represents a characteristic length scale of the i-th observation-action component, and σ_f^2 modulates the aleatoric variance of the output. The complete parameter set for each process includes their own variables $\{\ell_{f,1}, \cdots, \ell_{f,n}, \sigma_f\}$, for each output dimension. Generally speaking, the choice of covariance function is problem specific and typically reflects any prior knowledge concerning the correlation of outputs. See Rasmussen & Williams (2005) for a broader treatment of this subject.

We use an approximate form of evidence maximization starting from the marginal likelihood

$$p(\mathbf{f}_i|\mathcal{H}, \mathbf{o}') = \int p(\mathbf{f}_i|\mathcal{H}, \mathbf{o}'_i, \mathbf{k}_{g_i}) p(\mathbf{k}_{g_i}|\mathcal{H}, \mathbf{o}'_i) d\mathbf{k}_{g_i}.$$

Since we are unable to marginalize $\mathbf{k}_{g_i}$, we invoke the maximum likelihood approximation again. The approximate likelihood, $p(\mathbf{f}_i|\mathcal{H}, \mathbf{o}', \mathbf{k}^*_{g_i})$, provides a lower bound on the exact marginal, since our optimization procedure (Algorithm 8) cannot guarantee that $\mathbf{k}^*_{g_i}$ is a global maximizer. We obtain our optimization objective by taking the logarithm of the approximate likelihood:

$$\log p(\mathbf{f}_i|\mathcal{H}, \mathbf{o}', \mathbf{k}^*_{g_i}) = -\frac{T}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{K}_{f_i} + \mathbf{K}^*_{g_i}| - \frac{1}{2}\mathbf{y}^\top(\mathbf{K}_{f_i} + \mathbf{K}^*_{g_i})^{-1}\mathbf{o}_i. \quad (6.9)$$

Our complete optimization procedure is outlined in Algorithm 8. It follows the Expectation-Maximization algorithm described by Kersting et al. (2007). The expectation step estimates latent variances $\mathbf{z}$, and the maximization step maximizes the hyperparameters. We used a conjugate gradient optimization to fit the data. Evaluating the likelihood costs $\mathcal{O}(T^3)$ operations: the same as standard GP regression. The precise runtime of our proposed learning algorithm depends on the convergence tolerance $\epsilon \in \mathbb{R}$ and the maximum likelihood procedure. As with the standard EM algorithm, this procedure is not guaranteed to converge. However, we found that it performs reliably well in practice.

6.1.4 Model Improvement

6.2 On Policy RL with a Learned Transition Model

Given the current observation and action, the transition model (6.1) provides a way to generate a sequence of predicted observations which can be used in conjunction

Algorithm 8 Heteroscedastic Model Update

```
1: input: $\mathbf{\Theta}$, $\mathcal{H}$, $\mathbf{o}'$, $\mathbf{z}$, $M$
2: repeat for all $i = 1, \cdots, n$
3:     for all $(\mathbf{o}_t, \mathbf{a}_t) \in \mathcal{H}$ do
4:         $\{f_i^1, \cdots, f_i^M\} \sim \mathcal{N}(f_i(\mathbf{o}_t, \mathbf{a}_t), \sigma_{f_i}^2(\mathbf{o}_t, \mathbf{a}_t))$
5:         $z_i^t \leftarrow \log\left(\frac{1}{2M}\sum_{j=1}^{M}(o_i^t - f_i^j)^2\right)$
6:     $\mathbf{z}_i \leftarrow \text{concatenate}(z_i^1, \cdots, z_i^T)$
7:     Update $\mathbf{\Theta}_i$ to maximize (6.9) with $\mathcal{H}$, $\mathbf{o}_i'$ and $\mathbf{z}_i$.
8: until difference in subsequent $\mathbf{\Theta}_i$ is smaller than $\epsilon$.
9: output $\mathbf{\Theta}$, $\mathbf{z}$
```

with a reward model to compute sample returns. From the sample returns, policies can be evaluated and improved to maximize the expected return.

The learning system parameterizes a policy π with the vector $\boldsymbol{\theta} \in \mathbb{R}^d$ and optimizes their values to maximize the expected return. For each update, policy parameters are adjusted in steps of $\beta \in \mathbb{R}$ toward regions of increasing expected return. Algorithms such as these are known as policy optimization algorithms (Sutton & Barto, 2018).

$$\boldsymbol{\theta}_{j+1} \leftarrow \boldsymbol{\theta}_j + \beta_j \nabla_{\boldsymbol{\theta}} \mathbf{E}[G_t]. \tag{6.10}$$

Provided that rewards are bounded and the learning rate β_j is chosen from a Robbins-Monroe sequence, policy gradient methods are guaranteed to converge to a local optimum. By computing these numerically, we can support a large class of policies, even those with discontinuous parameterizations.

For d policy parameters, the partial derivative $\frac{\partial}{\partial \theta_j}$ can be estimated by perturbing the j-th parameter θ_j by an amount $\Delta\theta_j$ and computing the resulting change in the return $\Delta G^j = \eta(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}_j) - \eta(\boldsymbol{\theta} - \Delta\boldsymbol{\theta}_j)$. Sample returns come from model rollouts, where $\Delta\boldsymbol{\theta}_j$ is an elementary vector scaled by $\Delta\theta_j$ in the j-th dimension. A single gradient sample comes from computing these perturbations for every parameter dimension, and the result is averaged over m gradient samples total. Gradients are estimated with $\nabla_{\boldsymbol{\theta}}\mathbf{E}[G] \approx \frac{1}{2m\Delta\boldsymbol{\theta}}\sum_{i=1}^{m}\Delta G_i$.

Algorithm 9 describes the top-level policy update algorithm. For each update, N training trajectories are drawn from the environment to create the history $\mathcal{H}$, $\mathbf{o}'$. These are used to update the transition model with Algorithm 8. The updated model is subsequently used to estimate the policy gradient and update the policy's parameters $\boldsymbol{\theta}$. Policy updates are made with Algorithm 10. the number of policy updates U, trajectory sample size N,

Algorithm 9 Heteroscedastic Policy Gradient

```
1: input: $\beta_{1:U}$, $T$, $U$, $N$, $M$, $\mathbf{\Theta}$, $\boldsymbol{\theta}$,
2: for $u = 1, \cdots, U$ do
3:     Initialize $\mathcal{H} \leftarrow \emptyset$, $\mathbf{o}'_i \leftarrow \emptyset$, $\mathbf{z}$.
4:     for $j = 1, \cdots, N$ do
5:         $\mathbf{o}_1 \sim p_1(\cdot)$
6:         $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\mathbf{o}_t, \mathbf{a}_t) \colon \mathbf{o}_{t+1} \sim p(\cdot|\mathbf{o}_t, \mathbf{a}_t), \mathbf{a}_t \sim \pi_{\boldsymbol{\theta}_i}\ \forall\ t = 1, \cdots, T\}$
7:         $\mathbf{o}_i^{'u} \leftarrow$ concatenate$(o_2^i, \cdots, o_{T+1}^i)$ for all $i = 1, \cdots, n$.
8:         $\mathbf{o}_i^{'} \leftarrow \mathbf{o}'_i \cup \{\mathbf{o}_i^{'u}\}$ for all $i = 1, \cdots, n$.
9:     $\mathbf{\Theta}$, $\mathbf{z}$ $\leftarrow$Update Model$(\mathbf{\Theta}, \mathcal{H}, \mathbf{o}', \mathbf{z}, M)$
10:    $\nabla_{\boldsymbol{\theta}}\mathbf{E}[G] \leftarrow$ Model-based Policy Gradient$(p_1, r, \mathbf{\Theta}, \mathcal{H}, \mathbf{o}', \mathbf{z}, \pi_{\boldsymbol{\theta}}, T, \gamma)$
11:    $\boldsymbol{\theta}_{u+1} \leftarrow \boldsymbol{\theta}_u + \beta_u \nabla_{\boldsymbol{\theta}}\mathbf{E}[G]$
12: return $\boldsymbol{\theta}$, $\mathbf{\Theta}$, $\mathbf{o}'$, $\mathbf{z}$
```

Algorithm 10 Model-based Policy Gradient

```
1: input: $p_1, r, \mathbf{\Theta}, \mathcal{H}, \mathbf{o}', \mathbf{z}, \pi_{\boldsymbol{\theta}}, T, \gamma$
2: for $i = 1, \cdots, m$ do
3:     for $j = 1, \cdots, |\boldsymbol{\theta}|$ do
4:         $G(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}_j) = \sum_{t=0}^{T} \gamma^t r_t$ from rollout
5:         $G(\boldsymbol{\theta} - \Delta\boldsymbol{\theta}_j) = \sum_{t=0}^{T} \gamma^t r_t$ from rollout
6:         $\Delta G^j \leftarrow G(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}_j) - G(\boldsymbol{\theta} - \Delta\boldsymbol{\theta}_j)$
7:     $\Delta G_i \leftarrow$ concatenate$(\Delta G^1, \cdots, \Delta G^{|\boldsymbol{\theta}|})$
8: return $\frac{1}{2m\Delta\boldsymbol{\theta}}\sum_{i=1}^{m}\Delta G_i$
```

6.3 Related Work

Others have also employed GP regression to represent the transition model in RL (Deisenroth & Rasmussen, 2011; Ko et al., 2007; Nguyen-tuong et al., 2009). Deisenroth & Rasmussen (2011) present an alternative method to approximate the expected return, and they obtain a closed-form result by modeling the randomness of transition perturbations. Transitions are treated as independent and Gaussian, and moment-matching is applied to obtain expressions for at every time step. Our transition model can be contextualized more generally as a Chained Gaussian Process using an identity and a log link function with two latent processes (Saul et al., 2016).

6.4 Empirical Results

Here, two experiments are presented which study how the proposed transition model impacts the RL process. The first experiment addresses the question of sample efficiency in the context of model learning. The second experiment examines the full learning system and its ability to learn a navigation policy in the presence of simulated heteroscedastic noise. Additionally, the second experiment compares learning performance to a similar model-based algorithm that ignores heteroscedastic noise. Both of these experiments provide positive support to the hypothesis that there are environments where an RL system can improve its performance by representing aleatoric uncertainty.

Inspired by the sensing challenges faced by common underwater robots, the experiments consider three simulated environments. In particular, we considered a two degree-of-freedom (DOF) point robot (Point), a three DOF under-actuated surface vehicle (Ship), and a six DOF underwater vehicle (AUV). Each environment emits observations with spatially-varying noise according to the function

$$\varepsilon(\mathbf{x}) = \alpha_a \exp\left(\alpha_x \mathbf{x}^\top \mathbf{x}\right) \cdot \delta, \tag{6.11}$$

where $\mathbf{x}$ denotes some generic input of spatial variables, and $\delta \sim \mathcal{N}(0,1)$. This is intended to simulate the noise profile of a fixed USBL acoustic beacon (Figure 6.1). In each environment, parameters are set to $\alpha_a = 1$ and $\alpha_x = 10^{-4}$.

Experiments share several other commonalities. Rewards are assigned with a linear quadratic function $r(\mathbf{o}) = (\mathbf{o} - \mathbf{o}_{\text{goal}})^\top \mathbf{Q}(\mathbf{o} - \mathbf{o}_{\text{goal}})$, where $\mathbf{Q} = -0.1\mathbf{I}$. These encourage the system to navigate to the goal with a shortest Euclidean path in observation space. Returns are undiscounted ($\gamma = 1$), and T is specified differently for each environment. Sample trajectories were generated uniformly from the state space using the current policy with $M = 100$. Model updates are terminated either when subsequent parameters differ less than 10^{-4} in terms of the Euclidean norm, or after 1000 iterations.

6.4.1 Batch Size Ablations

This experiment investigates how model accuracy changes as the batch size N is increased. With the time horizon fixed at $T = 20$ steps, the experiment varies $N \in \{1, \cdots, 20\}$. Accuracy is measured as the mean-squared error between the model's predictions and a fixed trajectory, which is known a priori and comes from the true dynamics. The model is updated with Algorithm 8 until the convergence criteria are met.

Figure 6.3 shows the results using experience from each environment. In each case, the target tolerance of 10^{-4} was reached within about 10 trajectories. Convergence remains temporally stable, and this result is consistent with the intuition

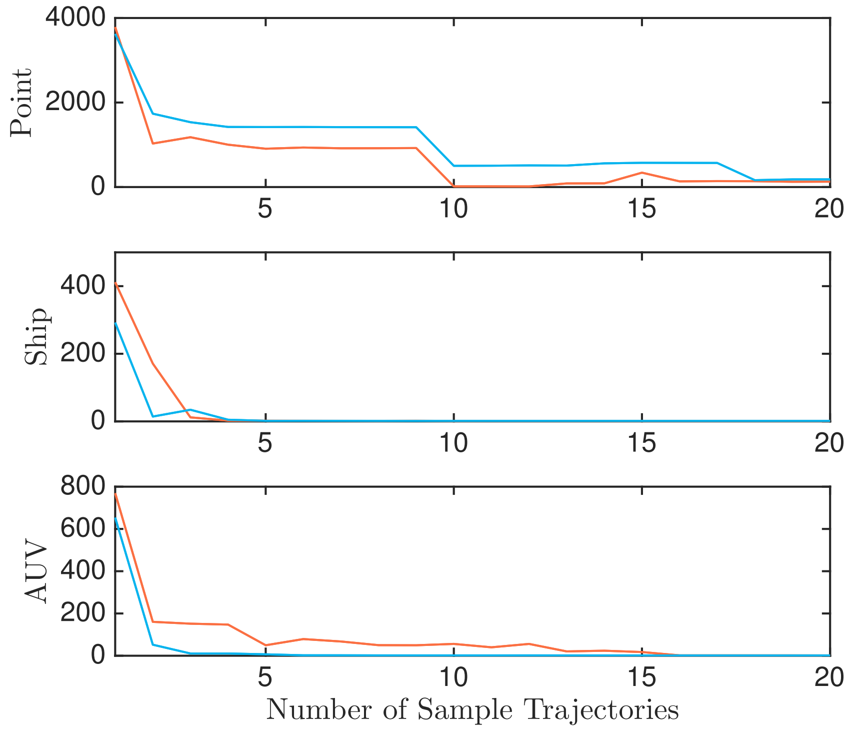


Figure 6.3: **Ablations of batch size:** The mean-squared error is shown for observation components of planar position whose uncertainty varies heteroscedastically (horizontal: red; vertical: blue). Batch size is varied for $N = 1, \cdots, 20$ trajectories of length $T = 20$. Results are averaged over 10 independent trials and show reasonable convergence after about ten trajectories.

that more training data produces more accurate predictions. We observed the point robot trajectories were highly dispersed in its observation space, and this increased its data requirements for the given correlation range.

6.4.2 Learning Navigation Policies in Simulation

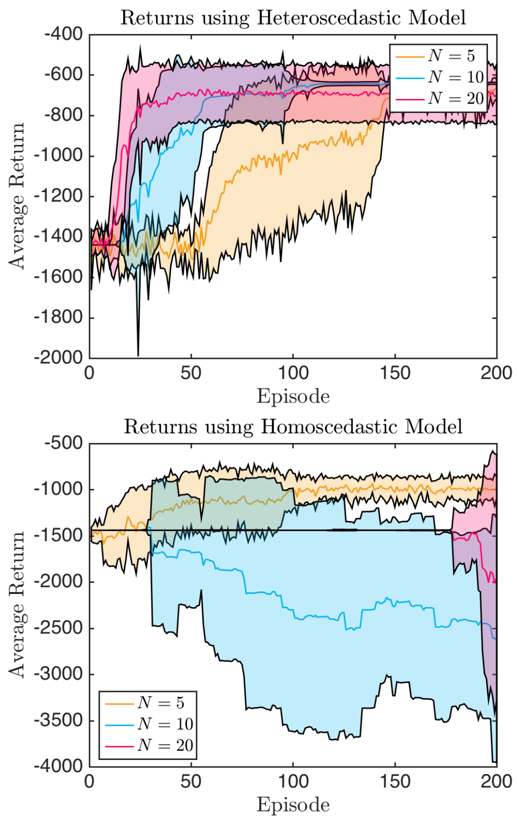
Planar point vehicle: Here the true transition dynamics are linear. The learning system experiences trajectories of $T = 20$ time steps with $\Delta = 0.5$s. Starting from $(40, 40)$, the system learns to navigate to the goal position $(100, 100)$. States are fully-observable, four-dimensional vectors $(x, \dot{x}, y, \dot{y})$ containing positions x, y and velocities $\dot{x}, \dot{y}$. Two actions (a_x, a_y) provide control with a linear policy $a = \mathbf{k}_a \mathbf{o}$ for each action, where $\mathbf{k}_a$ is a weight vector. The environment dynamics are

$$x_{t+1} = x_t + \Delta \dot{x}_t, \qquad y_{t+1} = y_t + \Delta \dot{y}_t,$$

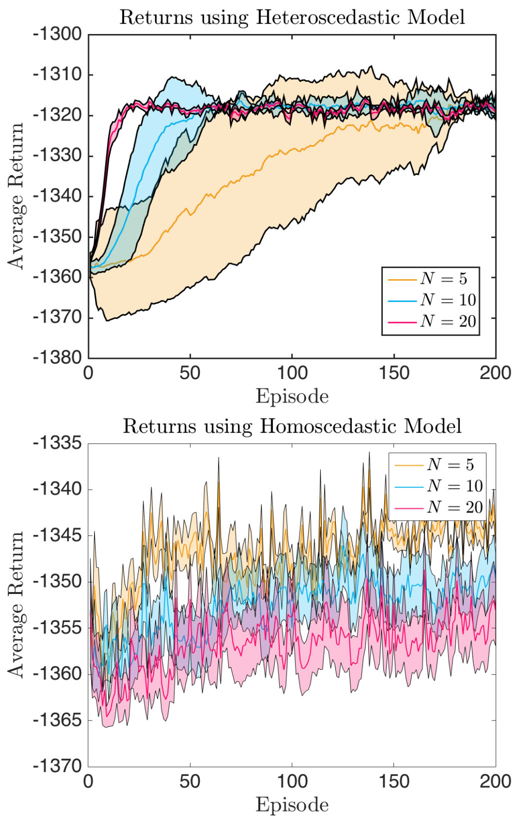
$$\dot{x}_{t+1} = \dot{x}_t + a_x \qquad \dot{y}_{t+1} = \dot{y}_t + a_y.$$

Results are shown in Figure 6.4 and discussed in Section 6.4.2.

Underactuated Ship Steering: In many cases, robots do not have full controllability of each dynamic degree of freedom. One example considered here is a planar autonomous surface vehicle, inspired by Ghavamzadeh et al. (2016). The learning system experiences trajectories of $T = 40$ time steps with $\Delta = 0.5$s. Starting from $(40, 40)$, the system learns to navigate to the goal position $(100, 100)$. The simulated vehicle moves at a constant speed of $V = 5$ m/s and asserts control by setting its angular rate $\omega \in [\pm 15°/\mathrm{s}]$. Fully-observable states $(x, y, \theta, \dot{\theta})$ contain positions $x, y,$

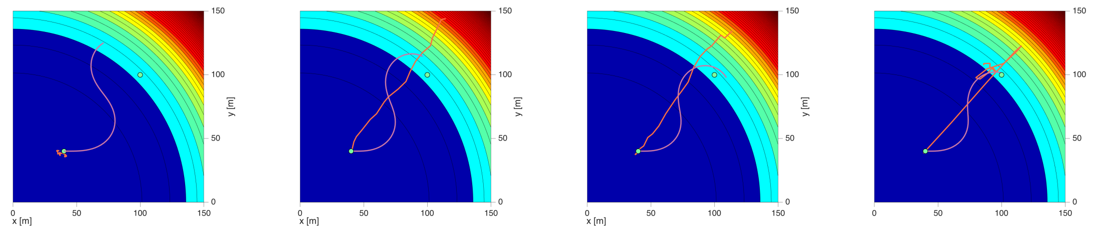


(a) Point Robot



(b) Ship

Figure 6.4: **Heteroscedastic transition models outperform naive alternatives:** We plot the evolution of cumulative reward for a point robot (Figure 6.4a) and a Ship (Figure 6.4b). The experiments were repeated with $N = 5, 10, 20$ sample trajectories and averaged over 10 independent trials. The standard Gaussian process model (bottom row) suffers greatly from bias and produces suboptimal behavior.



(a) 50 Episodes (b) 100 Episodes (c) 150 Episodes (d) 200 Episodes

Figure 6.5: **Trajectory evolution with policy improvement:** Sample trajectories are shown from the point robot (orange) and ASV (purple) as their policy improves. Initially, the path is suboptimal. Learning occurs in the presence of exponentially worsening noise whose severity is indicated by the color gradient.

heading θ, and yaw rate $\dot{\theta}$. The dynamics are

$$x_{t+1} = x_t + \Delta V \cos\theta_t, \qquad y_{t+1} = y_t + \Delta V \sin\theta_t,$$

$$\theta_{t+1} = \theta_t + \Delta\dot{\theta}_t, \qquad \dot{\theta}_{t+1} = \dot{\theta}_t + \frac{\Delta}{\tau}(\omega_t - \dot{\theta}_t).$$

The model includes a $\tau = 3$ step time delay for the command ω to be realized. This is intended to model delays from surface currents and actuator limitations. Learning rates remained fixed, and actions are chosen with a linear policy $\omega = \sum_{i=1}^{4} \mathbf{w}^\top \phi_i(\mathbf{s})$ using four basis tiles ϕ_i split around the goal position.

Results: Figure 6.4 shows the average return of an algorithm that learns with our proposed model and another model-based algorithm that ignores heteroscedastic noise. Our proposed algorithm achieves convergence within 200 episodes, and outperforms the baseline algorithm. Consistent with the batch size ablations, as N increases so does the rate of convergence. Training with more sample trajectories reduces model bias and produces more accurate gradient estimates. Conversely, ignoring heteroscedastic effects compounds model bias, which can lead to divergence from poor policy evaluations. The evolution of policy improvement can be visualized in the shortening of sample trajectories shown in Figure 6.5.

Autonomous Underwater Vehicle: Physical underwater vehicles operate in three spatial dimensions, and they often have six degrees of freedom. In this experiment, a six DOF underwater vehicle is simulated over a horizon of $T = 100$ steps of $\Delta = 0.2$s. With trajectories of this length, model bias becomes a significant concern, because new data can fall outside the model's correlational range and into the extrapolation regime. The environment we consider has a twelve-dimensional fully-observable state, containing position, x, y, z, orientation $\phi, \theta\psi$, and body rates u, v, w, p, q, r. The control system allows for forward acceleration $a_\text{f} \in [\pm 0.5 \text{ m/s}^2]$, vertical acceleration $a_\text{v} \in [\pm 1 \text{ m/s}^2]$, and heading acceleration $a_\text{d} \in [\pm 0.5 \text{ rad/s}^2]$. The system is under-actuated, since it cannot instantaneously control sway, bank, or attitude. The position rates evolve according to the equations of motion

$$\begin{aligned}
\dot{u}_{t+1} &= a_1 v_t r_t + a_2 u_t + a_3 u_t |u_t| + a_\text{f}, \\
\dot{v}_{t+1} &= b_1 u_t r_t + b_2 v_t + b_3 v_t |v_t|, \\
\dot{w}_{t+1} &= c_1 w_t + c_2 w_t |w_t| + a_\text{v}.
\end{aligned}$$

Yaw motion is described by $\dot{r}_{t+1} = d_1 r_t + d_2 r_t |r_t| + a_\text{d}$. Values used for all the numerical constants can be found in Eidsvik (2015). Starting from $(40, 40)$, the system learns to navigate to the goal position $(100, 100)$. Forward and vertical actions are computed with a linear policy, and heading actions are computed with $a_\text{d} = \sum_{i=1}^{4} w_i \phi_i(\mathbf{o})$, using the same features ϕ as the ASV problem.

Figure 6.6 shows the return averaged from 10 independent trials of data. Due to the large scale of variation, data is shown on a log scale. Both model types perform poorly when making long-range predictions. These results make intuitive sense; in the

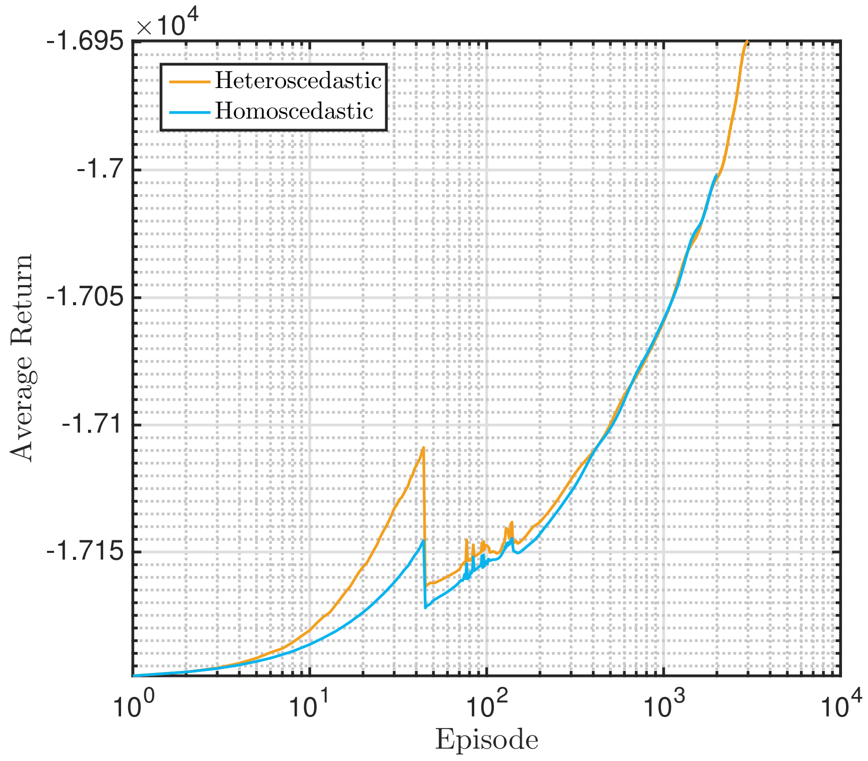


Figure 6.6: **Model training and Underwater Robot results:** The average return is shown as a simulated underwater vehicle learns to navigate. Initially, our algorithm achieves good performance. However, this eventually degrades as more bias enters the process. This leads both to experience poor asymptotic performance.

extrapolation regime, correlations are low. Thus, model bias has a more significant effect.

6.5 Discussion

This chapter presented a new model-based RL algorithm that uses hierarchal GP regression to model heteroscedastic transition dynamics of robotic systems. Our experimental results showed that model learning can be done with reasonable amounts of data for common simulated marine robotic systems.

Chapter 7

Summary and Future Work

This dissertation argues that

> *reinforcement learning algorithms that represent and manage uncertainty can be used to broaden the applicability of a robot decision making system.*

The final chapter summarizes the contributions that support this statement. Additionally, it describes some potential directions for future research.

7.1 Summary of Contributions

Chapter 3 presented an RL system that represents the aleatoric uncertainty of returns and uses that information to reduce exposure to unwanted randomness from the environment. A new algorithm was presented that learns return distributions using the optimization framework of Wasserstein gradient flows. This framework allows one to guarantee convergence of the return distribution's first and second moments, which is a necessary condition to establish stochastic dominance in the second order (Fishburn, 1980). Second order stochastic dominance was presented as a distributional relation that could be used to select actions, such that the system's expected performance is preserved, and its exposure to risk is reduced. Before this work, SSD had not appeared in the RL literature; systems typically employed conditional value at risk (Artzner et al., 1999). The experimental results from this chapter showed that our full system, which includes the WGF learning algorithm and the SSD decision policy, could broaden the class of environments to which RL systems can be applied. For these environments, data showed that an RL system's expected performance can be preserved when compared to the uncertainty-agnostic greedy policy used by conventional systems. These results were demonstrated for tabular and for moderate-scale state spaces which require function approximation. In comparison to the single risk level of conditional value at risk, the proposed SSD policy was shown to explore more consistently, because it considers multiple risk levels in its evaluation of uncertainty.

Despite the advances of Chapter 3, its results apply under particular conditions that can sometimes be unrealistic for robotic systems. For example, the SSD policy requires the environment to have multiple paths of equal expected return; otherwise, its actions are identical to the standard greedy policy. Even when this precondition occurs, it can be challenging to satisfy within physically-attainable data regimes (e.g. about 10^5 transitions), and when computations involve floating point numbers of stochastic data. Many environments relevant to robotics will seldom contain multiple trajectories with strictly equal values (i.e. zero action gap). In reality, action gaps may be close, but they are never exactly equal. Still, the principles of Chapter 3 are potentially beneficial for robots seeking reliable and optimal actions.

Chapter 4 addressed some of these challenges and extended the line of work in several ways the previous chapter did not consider. The chapter focused on RL systems aimed at mobile robots learning to navigate. A new algorithm was presented that learns return distributions incrementally online and without physically-implausible episodic resets. To support sensors with high-dimensional output, samples approximating return distributions were represented as linear functions of a feature vector. Before this work, only tabular representations had been applied to the online setting; most work treated the offline setting with deep neural networks (Dabney et al., 2017). In choosing samples to be linear functions of features, the proposed algorithm gains generality to handle observations of unknown structure. This can scale systems to complex domains, where observation structure is unknown or difficult to embed through an inductive bias. For resource-constrained systems, linear representations permit their computations to be highly parallelized for time-efficient online learning. Experimental results demonstrated that the proposed algorithm was capable of handling a realistic robot sensory stream, containing over one-hundred thousand dimensions of simulated lidar information. The algorithm was applied to improve reliability of actions from data experienced in an off-policy and continual manner; after transitioning under the behavior policy for an undetermined amount of time, the learner switches to use the target policy. After the goal has been reached, the learner would switch back to the behavior policy and continue refining its actions based on new experience. Unlike the conventional episodic paradigm, which requires the learner to be reset, and doesn't consider what happens after the goal is reached, the proposed learning process encompasses a larger and more physically-plausible scope. Results showed the system was able to reduce its exposure to uncertainty. Finally, the chapter presented evidence showing that the proposed linear representation struggles to differentiate its actions from the greedy policy, in comparison to an oracle tabular learner. These results provide further support for the thesis statement, and they motivate future work, which is discussed in the next section.

In contrast to the data regimes considered in Chapters 3 and 4, common sensory streams from underwater robots produce much less data. In addition, these robots' sensors can be highly prone to drift and struggle to provide consistent measurements across learning episodes. Issues such as these make it challenging or altogether infeasible to apply conventional RL algorithms to underwater robots.

Chapter 5 addressed the issue of on-policy evaluation in low-data regimes. The chapter presented a Bayesian non-parametric regression algorithm for estimating the expected return while representing the associated epistemic uncertainty. To improve the algorithm's online viability, the chapter proposed a sparse approximation based on the pseudo-input paradigm of Snelson & Ghahramani (2006). This uses posterior moments, including a measure of epistemic uncertainty (i.e. the covariance), to compress experience into manageable amounts with which to make predictions. Evidence was presented showing learned navigation policies could control simulated mobile robots. Additionally, evidence showed the algorithm could control a physical underwater robot, while experiencing noisy and drifting acoustic sensory data. These results imply the proposed algorithm may be applied to other robotic systems that learn to make decisions with low amounts of data. This broadens the applicability of robotic decision making systems that are difficult to model a priori, and whose sensors preclude the application of conventional RL algorithms. In particular, data demands of such systems may be drastically reduced using the proposed algorithm.

Another challenge faced by underwater robots is heteroscedastic uncertainty affecting observations. Chapter 6 presented an algorithm to learn heteroscedastic transition models. The algorithm was based on a hierarchal Gaussian process regression, similar to Kersting et al. (2007), which leverages maximum likelihood assumptions to make inference tractable. The chapter demonstrates how to integrate the proposed transition model into a model-based algorithm for RL, using batches of trajectories from which to learn. Experiments showed that by representing heteroscedastic uncertainty, predictions made by the transition model were more accurate than models assuming constant noise properties. In the data, this resulted in performance improvements for several simulated robotic vehicles. In a broader context, the proposed algorithm enables robots to apply RL in more settings: in the presence of heteroscedastic observation noise.

7.2 Discussion and Future Directions

The contributions of this dissertation offer several ways they could be extended. In some cases, their results open up new research directions which did not exist before. This section discusses a few possible avenues for future work.

7.2.1 Uncertainty, Robotics, and Continual Learning

Much RL research focuses on algorithms that experience data episodically. In simulated environments, the episodic assumption has little bearing; it is straightforward to impose resets programatically. However, episodes can often be unnatural for mobile robotic systems, which require physical involvement whenever terminal conditions are met. This dissertation showed that continual learning (Ring et al., 1994) can be a more appropriate alternative, particularly in the case of self-driving vehicles (Chapter 4). In this setting, a system is always learning, and it never resets. Many others in RL study continual learning (Kirkpatrick et al., 2017; Kaplanis et al., 2019; Fedus et al., 2020). Continual learning for mobile robotics could be very promising when combined with uncertainty-aware decision making. Not only could this help robots more naturally learn over longer periods of time, but it could reveal different settings in which to study uncertainty.

7.2.2 Balancing Uncertainty and Approximation Error

The results of Chapter 4 provide evidence that SSD is difficult to identify under function approximation. Based on this data, it is natural to wonder whether there are other criteria with similar benefits that can be realized with less data. Here, at least two strands of research can be geared toward robotics. In one case, the relation between SSD and the ordering on moments could be explored – perhaps to define a new policy that tolerates small action gaps. This would presumably apply to a broader class of environments, where small action-gaps exist; for instance, any roadmap where two routes differ in length by less than a tolerance. In the other case, the distribution's accuracy could be studied with a goal of reducing false-negative SSD comparisons. Supposing the false-negative errors were concentrated to a subset of sample estimates, then a policy could be designed to remove or weight these appropriately. If excluding erroneous samples could be guaranteed to still approximate the underlying random variable well, then the false-negative rate would be reduced. In both cases, future work would hope to show a new policy retains much of the SSD relation's desired behavior.

Instead of managing represntation error through the learner's policy, it worth investigating whether the error could be reduced directly, with a different representation using nonlinear features. Convolutional features have proven effective for representing spatial data that comes from cameras and other imaging sensors. Like many nonlinear features, convolutional features often demand an abundance of experience to optimize: often on the order of 10^8 transitions (Machado et al., 2018). In smaller domains, nonlinear features from multilayer perceptrons have been effective using roughly 10^6 steps of experience (Schulman et al., 2017). However, both convolutional and MLP features have predominately optimized offline, with batches of saved episodic experience. Future work could explore algorithms for bringing these representations to the online continual RL setting.

7.2.3 Improving Exploration with Epistemic Uncertainty

The work in Chapter 5 primarily focused on updating the posterior over a value function and using the posterior's mean for decision making and policy improvement. In the same low-data settings this work considered, it could also be interesting to incorporate the predictive variance as a measure of the value function's epistemic uncertainty. This would allow the learner to identify and explore unknown states in different and potentially-useful ways. For instance, the learner could apply the principle of optimism in the face of uncertainty (Brafman & Tennenholtz, 2002). This has been a useful strategy for incorporating epistemic uncertainty in RL, leading to optimal exploration in tabular domains. Conversely, a learner could choose actions so their outcomes are closer to what has been experienced in the past. Such approaches have been shown to help when using model-based algorithms with limited capacity (Abbas et al., 2020). Both approaches are expected to be useful in low-data regimes, where the learner would like to weight its exploration with its prediction confidence.

7.3 Conclusions

This dissertation presented several reinforcement learning algorithms for representing and managing uncertainty in robotic problem settings. Each algorithm was designed to handle a specific kind of uncertainty. The algorithms for handling aleatoric uncertainty used distributions to express the full spread of outcomes a learner could expect to encounter. These were used to inform more predicable decision making, and they were shown to be applicable to realistic robot sensory streams. Other algorithms were designed to handle epistemic uncertainty in the expected return. These algorithms were based on the data-efficient principles of Gaussian process regression. Evidence supported how both of these approaches could broaden the applicability of robot decision making systems, by reducing the amount of prior information they require to learn policies, and expanding the conditions under which they can be applied. The algorithms presented in this dissertation are important steps toward making robots more generally applicable.

Bibliography

Abbas, Z., Sokota, S., Talvitie, E., & White, M. (2020). Selective dyna-style planning under limited model capacity. In *International Conference on Machine Learning* (pp. 1–10).: PMLR.

Ambrosio, L. (2005). *Gradient Flows in Metric Spaces and in the Space of Probability Measures*. Lectures in Mathematics ETH Zurich.

Artzner, P., Delbaen, F., Eber, J.-M., & Heath, D. (1999). Coherent measures of risk. *Mathematical Finance*, 9(3), 203–228.

Åström, K. J. & Murray, R. M. (2010). *Feedback systems*. Princeton university press.

Atkeson, C. G. & Santamaria, J. C. (1997). A comparison of direct and model-based reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4 (pp. 3557–3564).

Barth-Maron, G., Hoffman, M., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N., & Lillicrap, T. (2018). Distributed distributional policy gradients. In *International Conference on Learning Representations (ICLR)*.

Bellemare, M., Dabney, W., & Munos, R. (2017). A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., & Wang, Z. (2020). Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836), 77–82.

Bellman, R. (1957). *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press, 1 edition.

Brafman, R. I. & Tennenholtz, M. (2002). R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct), 213–231.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym.

Chow, Y. & Ghavamzadeh, M. (2014). Algorithms for cvar optimization in mdps. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS)*.

Chow, Y., Ghavamzadeh, M., Janson, L., & Pavone, M. (2017). Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18.

Csato, L. & Opper, M. (2002). Sparse on-line gaussian processes. *Neural Computation*.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS)*.

Dabney, W., Ostrovski, G., Silver, D., & Munos, R. (2018). Implicit quantile networks for distributional reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

Dabney, W., Rowland, M., Bellemare, M., & Munos, R. (2017). Distributional reinforcement learning with quantile regression. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.

Deisenroth, M. & Rasmussen, C. (2011). Pilco: A model-based and data-efficient approach to policy search. In *In Proceedings of the International Conference on Machine Learning (ICML).*

Deisenroth, M. P., Rasmussen, C. E., & Peters, J. (2009). Gaussian process dynamic programming. *Neurocomput.*

Delage, E. & Mannor, S. (2010). Percentile optimization for markov decision processes with parameter uncertainty. *Oper. Res.*, 58(1), 203–213.

Dentcheva, D. & Ruszczyński, A. (2006). Inverse stochastic dominance constraints and rank dependent expected utility theory. *Mathematical Programming.*

Dentcheva, D. & Ruszczyński, A. (2006). Portfolio optimization with stochastic dominance constraints. *Journal of Banking & Finance*, 30(2), 433–451.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning* (pp. 1–16).

Eidsvik, O. (2015). Identification of hydrodynamic parameters for remotely operated vehicles. Master's thesis, Norweigian University of Science and Technology.

Engel, Y. (2005). *Algorithms and Representations for Reinforcement Learning.* PhD thesis, The Hebrew University.

Engel, Y., Mannor, S., & Meir, R. (2003). Bayes meets bellman: The gaussian process approach to temporal difference learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML).*

Engel, Y., Mannor, S., & Meir, R. (2005). Reinforcement learning with gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*.

Engel, Y., Szabo, P., & Volkinshtein, D. (2006). Learning to control an octopus arm with gaussian process temporal difference methods. In *Advances in Neural Information Processing Systems 18*. MIT Press.

Fedus, W., Ghosh, D., Martin, J. D., Bellemare, M. G., Bengio, Y., & Larochelle, H. (2020). On catastrophic interference in atari 2600 games. *arXiv preprint arXiv:2002.12499*.

Fishburn, P. (1980). Stochastic dominance and moments of distributions. *Math. Operations Research*.

Ghavamzadeh, M., Engel, Y., & Valko, M. (2016). Bayesian policy gradient and actor-critic algorithms. *Journal of Machine Learning Research*.

Gobbino, M. (1999). Minimizing movements and evolution problems in euclidean spaces. *Annali di Matematica Pura ed Applicata*.

Goldberg, P. W., Williams, C. K. I., & Bishop, C. M. (1998). Regression with input-dependent noise: A Gaussian process treatment. In *Advances in Neural Information Processing Systems 10* (pp. 493–499).

Haarnoja, T., Tang, H., Abbeel, P., & Levine, S. (2017). Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

Hernández-lobato, D., Sharmanska, V., Kersting, K., Lampert, C. H., & Quadrianto, N. (2014). Mind the nuisance: Gaussian process classification using privileged noise. In *Advances in Neural Information Processing Systems 27* (pp. 837–845).

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. G., & Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Hester, T., Quinlan, M., & Stone, P. (2010). Generalized model learning for reinforcement learning on a humanoid robot. In *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 2369–2374).

Hester, T. & Stone, P. (2011). *Learning and Using Models*. Springer.

Jakab, H. & Csato, L. (2011). Improving gaussian process value function approximation in policy gradient algorithms. In *Artificial Neural Networks and Machine Learning*.

Jordan, R., Kinderlehrer, D., & Otto, F. (1998). The variational formulation of the fokker–planck equation. *SIAM Journal on Mathematical Analysis*.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., & Levine, S. (2018). Scalable deep reinforcement learning for vision-based robotic manipulation. In *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research* (pp. 651–673).: PMLR.

Kaplanis, C., Shanahan, M., & Clopath, C. (2019). Policy consolidation for continual reinforcement learning. *arXiv preprint arXiv:1902.00255*.

Keramati, R., Dann, C., Tamkin, A., & Brunskill, E. (2020). Being optimistic to be conservative: Quickly learning a CVaR policy. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*.

Kersting, K., Plagemann, C., Pfaff, P., & Burgard, W. (2007). Most likely heteroscedastic Gaussian process regression. In *Proceedings of the 24th International Conference on Machine Learning*.

Kingma, D. P. & Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, (ICLR)*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13), 3521–3526.

Ko, J., Klein, D. J., Fox, D., & Haehnel, D. (2007). Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 742–747).

LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

Liu, Q. & Wang, D. (2016). Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in Neural Information Processing Systems 29*.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., & Bowling, M. (2018). Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61, 523–562.

Markowich, P. & Villani, C. (1999). On the trend to equilibrium for the fokker-planck equation: An interplay between physics and functional analysis. In *Physics and Functional Analysis, Matematica Contemporanea (SBM) 19*.

Markowitz, H. (1952). Portfolio selection. *Journal of Finance*.

Martin, J., Lyskawinski, M., Li, X., & Englot, B. (2020). Stochastically dominant distributional reinforcement learning. In *International Conference on Machine Learning* (pp. 6745–6754).: PMLR.

Martin, J. D. & Englot, B. (2017). Extending model-based policy gradients for robots in heteroscedastic environments. In *1st Annual Conference on Robot Learning (CoRL)*.

Martin, J. D., Wang, J., & Englot, B. (2018). Sparse gaussian process temporal difference learning for marine robot navigation. In *2nd Annual Conference on Robot Learning (CoRL)*.

Mnih, V., D., K. K., Silver, Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., D, W., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*.

Nevmyvaka, Y., Feng, Y., & Kearns, M. (2006). : New York, NY, USA: Association for Computing Machinery.

Nguyen-tuong, D., Peters, J. R., & Seeger, M. (2009). Local Gaussian process regression for real time online model learning. In *Advances in Neural Information Processing Systems 21* (pp. 1193–1200).

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1st edition.

Rasmussen, C. & Kuss, M. (2004). Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems 16*: MIT Press.

Rasmussen, C. & Williams, C. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Riedmiller, M. (2005). Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning* (pp. 317–328).: Springer.

Ring, M. B. et al. (1994). *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin Austin, Texas 78712.

Rowland, M., Bellemare, M., Dabney, W., Munos, R., & Teh, Y. (2018). An analysis of categorical distributional reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Rowland, M., Dadashi, R., Kumar, S., Munos, R., Bellemare, M. G., & Dabney, W. (2019). Statistics and samples in distributional reinforcement learning. In *International Conference on Machine Learning* (pp. 5528–5536).: PMLR.

Rummery, G. A. & Niranjan, M. (1994). *On-Line Q-Learning Using Connectionist Systems*. Technical report, Cambridge University.

Sato, M., Kimura, H., & Kobayashi, S. (2001). Td algorithm for the variance of return and mean-variance reinforcement learning. *Transactions of the Japanese Society for Artificial Intelligence.*

Saul, A. D., Hensman, J., Vehtari, A., & Lawrence, N. D. (2016). Chained Gaussian processes. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (pp. 1431–1440).

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347.*

Sinkhorn, R. (1967). Diagonal equivalence to matrices with prescribed row and column sums. In *The American Mathematical Monthly.*

Snelson, E. & Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems.*

Sutton, R. & Barto, A. (1998). *Introduction to Reinforcement Learning.* Cambridge, MA, USA: MIT Press, 1st edition.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Mach. Learn.*, 3(1), 9–44.

Sutton, R. S. & Barto, A. G. (2018). *Reinforcement learning: An introduction.* MIT press.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., & Precup, D. (2011). Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In L. Sonenberg, P. Stone, K. Tumer, & P. Yolum (Eds.), *AAMAS* (pp. 761–768).: IFAAMAS.

Tamar, A., Castro, D. D., & Mannor, S. (2013). Temporal difference methods for the variance of the reward to go. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*.

Tamar, A., Glassner, Y., & Mannor, S. (2015). Optimizing the CVaR via sampling. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*.

Titsias, M. & Lázaro-Gredilla, M. (2011). Variational heteroscedastic Gaussian process regression. In *Proceedings of the 28th International Conference on Machine Learning*.

Vanhatalo, J., Riihimäki, J., Hartikainen, J., Jylänki, P., Tolvanen, V., & Vehtari, A. (2013). GPstuff: Bayesian modeling with Gaussian processes. *Journal of Machine Learning Research*, 14(Apr), 1175–1179.

Villani, C. (2008). *Optimal Transport: Old and New*. Springer.

Watkins, C. J. C. H. & Dayan, P. (1992). Technical note: q-learning. *Mach. Learn.*, 8(3-4), 279–292.

Zhang, R., Chen, C., Li, C., & Carin, L. (2018). Policy optimization as Wasserstein gradient flows. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

Chapter 8
Appendix

8.1 Chapter 3 Appendix

8.1.1 Experimental Details for the Regression Comparison

This experiment compared the first and second moments obtained from distributional regression algorithms: quantile regression and the proposed Wasserstein gradient flow algorithm. Distributions were parameterized with the same number of atoms, which varied in $\{5, 10, 20, 50\}$. The atom locations were optimized from data drawn from a five-component Gaussian mixture model. The Gaussians used the following parameters; components were equally likely to be experienced ($c_i = 1/5$); component means were $\mu_i \in \{-5, -3, 0, 5, 6, 9\}$; standard deviations $\sigma_i \in \{1, 2, 1, 2, 1, 0.5\}$, for $i = 1, \cdots, 5$. Models were evaluated on a separate draw of the same size as the training set. Target values, y, were computed using empirical estimates from $10,000$ samples. The violin plots show the distribution of root mean square error $RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(y - \hat{y})^2}$ samples between the targets and the estimates $\hat{y}$ over $N = 100$ trials.

8.1.2 Experimental Details for the Ablation Study

This experiment evaluated the proposed algorithm's performance with coarse sweeps of the minimum temperature $\beta^{-1} \in \{0.01, 0.1, 0.2, 0.25, 0.5, 0.9\}$ and step size $h \in \{0.01, 0.1, 0.5, 1., 10.\}$. Evaluations considered 50 trials. Data came from the same five-component Gaussian mixture model used in the Regression Comparison experiment. Results show the root mean square error in the first and second moments with targets computed using $10,000$ samples and empirical estimators.

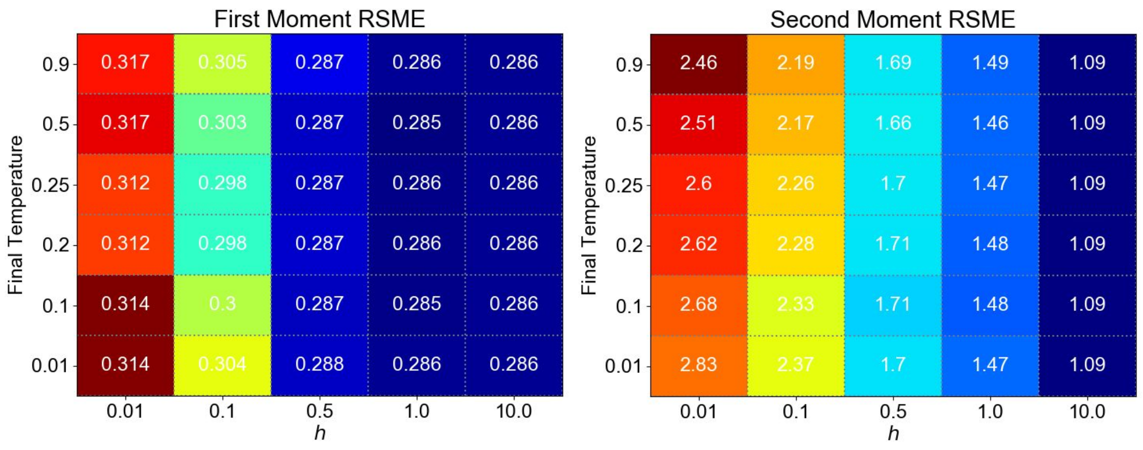


Figure 8.1: Ablations of temperature β and step size h in proximal loss.

8.1.3 Experimental Details for WGF Policy Evaluation

This experiment performance policy evaluation on Monte Carlo (MC) returns from the optimal policy. The optimal policy was obtained by running Q-learning for $10,000$ episodes with an ($\epsilon = 0.1$)-greedy behavior policy, $\gamma = 0.9$, learning rate $\alpha = 0.5$, and using an absorbing terminal state. MC returns were computed for each state from 200 rollouts of 200 time steps. A discrete distribution was parameterized with 200 atoms initialized from a standard Gaussian, then transported using 100 gradient steps with a step size of 0.5. The proximal loss was annealed down from $\beta^{-1} = 1$ to 0.25 in minimum steps of 0.5; the proximal time step was set to $h = 1$. We report the curves of the proximal loss and the squared value error at each gradient step.

8.1.4 Experimental Details for WGF in the Control Setting

This experiment used the OpenAI Gym (Brockman et al., 2016) environments MountainCar, CartPole, and LunarLander with discrete actions. Atom locations were computed using a two-layer, fully-connected neural network, where each layer had 256 hidden units. Networks were optimized with the proposed WGF proximal loss and

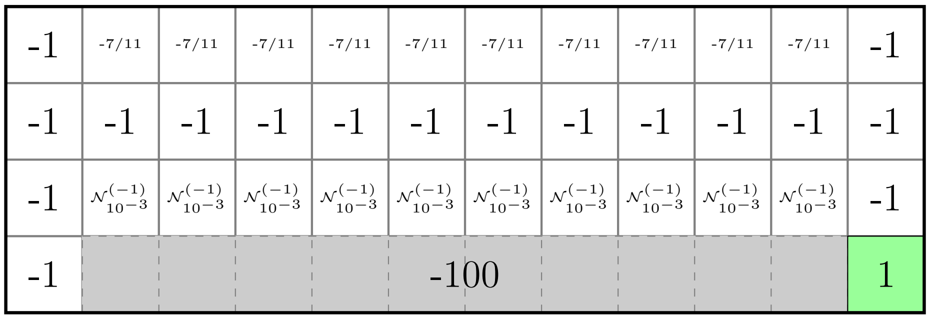

-1
-7/11
-7/11
-7/11
-7/11
-7/11
-7/11
-7/11
-7/11
-7/11
-7/11
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
$\mathcal{N}_{10^{-3}}^{(-1)}$
$\mathcal{N}_{10^{-3}}^{(-1)}$
$\mathcal{N}_{10^{-3}}^{(-1)}$
$\mathcal{N}_{10^{-3}}^{(-1)}$
$\mathcal{N}_{10^{-3}}^{(-1)}$
$\mathcal{N}_{10^{-3}}^{(-1)}$
$\mathcal{N}_{10^{-3}}^{(-1)}$
$\mathcal{N}_{10^{-3}}^{(-1)}$
$\mathcal{N}_{10^{-3}}^{(-1)}$
$\mathcal{N}_{10^{-3}}^{(-1)}$
-1
-1
-100
1

Figure 8.2: CliffWorld with a modified reward structure for multiple solutions. Here $\mathcal{N}_{\sigma}^{(\mu)}$ denotes $\mathcal{N}(\mu, \sigma)$.

compared to a network optimized with the quantile regression loss (Dabney et al., 2017). Both models regressed 2 quantiles. Network parameters were adjusted with the Adam optimizer (Kingma & Ba, 2015) using a step size of 10^{-3}. Experience replay buffers with mini-batches of size 32 and a total capacity of $10,000$ transition samples were used. Each learner explored with an ($\epsilon = 0.1$)-greedy policy, using $\gamma = 0.99$ until the absorbing state was reached. We report data for 5 independent trials.

8.1.5 Experimental Details for Control in the Presence of Uncertainty

This experiment used data generated in the CliffWalk environment (Sutton & Barto, 1998). The environment's reward structure was modified with the addition of a second optimal trajectory. One trajectory emits stochastic rewards, whereas the other trajectory's rewards are deterministic (Figure 8.2). Stochastic rewards were clipped to be within the interval $[-10, 10]$. Both the WGF and quantile regression agents used a tabular representation of 16 atoms for each return distribution. Learning occurred with $\gamma = 1$, a maximum horizon length of 500, and the same loss settings used in the policy evaluation experiment. The number of gradient steps was limited to 50, unless a tolerance of 10^{-8} was exceeded below first. Data gathered from $M = 50$ independent trials is reported. The 95% confidence intervals were computed using the standard

t-distribution with $M-1$ degrees of freedom.

8.1.6 Mathematical Proofs

This section provides proofs for the main theoretical results of Chapter 3. Supporting results include references to their original source(s). All the following results involving probability measure apply for single measures.

Lemma 1. *Let $\tau \in (0,1)$ and consider $\xi_\tau = F_X^{-1}(\tau)$. Then $F_X^{-2}(\tau) = \mathbf{E}[X \leq \xi_\tau]$.*

Proof. By conjugate duality,

$$\begin{aligned} F_X^{-2}(\tau) &= \tau\xi_\tau - F_X^{(2)}(x), \\ &= \tau\xi_\tau - \tau\mathbf{E}[X - \xi_\tau | X \leq \xi_\tau], \\ &= \tau\mathbf{E}[X | X \leq \xi_\tau], \\ &= \mathbf{E}[X \leq \xi_\tau]. \end{aligned}$$

□

Proposition 1. *$G_1 \succeq_{(2)} G_2$ if, and only if $\sum_{i=1}^{j} g_1^{[i]} \geq \sum_{i=1}^{j} g_2^{[i]}$, $\forall\ j = 1, \cdots, N$.*

Proof. We prove the result in the context of random returns. However, this holds for general random variables. We consider random returns induced by two different actions, respectively denoted G_1, G_2. Each return distribution is approximated with a discrete Lagrangian measure

$$\mu_1 \approx \frac{1}{N}\sum_{n=1}^{N} \delta(g_1^{(n)}), \qquad\qquad \mu_2 \approx \frac{1}{N}\sum_{n=1}^{N} \delta(g_2^{(n)}).$$

Given that $G_1 \succeq_{(2)} G_2$, we know by the definition that $F_1^{-2}(\tau) \geq F_2^{-2}(\tau)$ for all $\tau \in (0,1)$. Invoking Lemma 1 allows us to rewrite the definition with total expectations

$$\mathbf{E}[G_1 \leq \xi_1^{(\tau)}] \geq \mathbf{E}[G_2 \leq \xi_2^{(\tau)}],\ \forall\, \tau \in (0,1).$$

Denote the ordered samples of a return distribution to be $g^{[1]} \leq g^{[2]} \leq \cdots \leq g^{[N]}$. Then with particle sets from each measure, we have

$$\sum_{i=1}^{j} g_1^{[i]} \geq \sum_{i=1}^{j} g_2^{[i]},\ \forall\, j = 1, \cdots, N.$$

The other implication follows by normalizing the sums with $1/N$ and invoking Lemma 1 again to arrive at the definition. $\square$

Proposition 2 (Fishburn (1980))**.** *Assume μ has two finite moments. Then $X \succeq_{(2)} Y$ implies $\mu_X^{(1)} \geq \mu_Y^{(1)}$ or $\mu_X^{(1)} = \mu_Y^{(1)}$ and $\mu_X^{(2)} \leq \mu_Y^{(2)}$, where $(\cdot)$ denotes a particular moment of the distribution μ.*

Proof. This result follows from Theorem 1 of Fishburn (1980), which proves an ordering dominance of any finite degree. $\square$

Proposition 3. *Let $\{\mu_t\}_{t\in[0,1]}$ be an absolutely-continuous curve in $\mathcal{P}(\mathbb{R})$ with finite second-order moment. Then for $t \in [0,1]$, the vector field $\mathbf{v}_t = \nabla(\frac{\delta E}{\delta t}(\mu))$ defines a gradient flow on $\mathcal{P}(\mathbb{R})$ as $\partial_t \mu_t = -\nabla \cdot (\mu_t \mathbf{v}_t)$, where $\nabla \cdot \mathbf{u}$ is the divergence of some vector $\mathbf{u}$.*

Proof. See Ambrosio (2005), Theorem 8.3.1. $\square$

Proposition 4. *Let $\mu_0 \in \mathcal{P}_2(\mathbb{R})$ have finite free energy $E(\mu_0) < \infty$, and for a given $h > 0$, let $\{\mu_t^{(h)}\}_{t=0}^{K}$ be the solution of the discrete-time variational problem, with* *measures restricted to $\mathcal{P}_2(\mathbb{R})$, the space with finite second moments. Then as $h \to 0$, $\mu_K^{(h)} \to \mu_T$, where μ_T is the unique solution of the Fokker-Plank equation at $T = hK$.*

Proof. See Jordan et al. (1998), Theorem 5.1. □

Proposition 5. *Let $\{\mu_t^{(h)}\}_{t=0}^K$ be the solution of the discrete-time JKO variational problem, with measures restricted to $\mathcal{P}_2(\mathbb{R})$, the space with finite second moments. Then $E(\mu_t)$ is a decreasing function of time.*

Proof. We show that the free-energy $E(\mu) = F(\mu) + \beta^{-1}H(\mu)$ is a Lyapunov functional for the Fokker-Planck (FP) equation. Following the approach of Markowich & Villani (1999), we consider the change of variables $\mu_t = h_t e^{-U}$, where we let $\beta = 1$ without loss of generality. With this, FP is equivalent to

$$\partial_t h_t = \Delta h_t - \nabla U \cdot \nabla h_t. \tag{8.1}$$

Whenever ϕ is a convex function, one can check the following is a Lyapunov functional for (8.1), and equivalently FP:

$$\int \phi(h_t) e^{-U} dz = \int \phi(\mu_t e^U) e^{-U} dz.$$

Differentiating with respect to time shows

$$\frac{d}{dt} \int \phi(h_t) e^{-U} dz = -\int \phi''(h_t) |\nabla h_t|^2 e^{-U} dz < 0.$$

Now consider $\phi(h_t) = h_t \log(h_t) - h_t + 1$. With the identity $\int (h_t - 1) e^{-U} dz = 0$, we find

$$\int \phi(h_t) e^{-U} dz = \int \mu_t \log\left(\frac{\mu_t}{e^{-U}}\right) dz = \int \mu_t (U + \log \mu_t) dz = E(\mu).$$

Thus, the free-energy functional is a Lyapunov function for the Fokker-Planck equation, and $E(\mu_t)$ is a decreasing function of time. In the low-energy state the optimal distributional Bellman equation is satisfied with pure Brownian motion. □

Theorem 1. *If $\mathcal{T}\mu = \mu$, then $\operatorname{Prox}^{\mathcal{W}}_{hE}(\mu) = \mu$ as $\beta \to \infty$.*

Proof. Let $d(\mu, \nu)$ be some distributional distance between measures μ and ν, such as the supremal k-Wasserstein $= \sup_{s,a} \mathcal{W}_k(\mu, \nu)$. Furthermore, suppose $\mu^* = \mathcal{T}\mu^*$ is the fixed point of the optimal distributional Bellman operator $\mathcal{T}$. We consider the proximal operator

$$\operatorname{Prox}^{\mathcal{W}}_{hE}(\mu_k) = \arg\min_{\mu} \mathcal{W}_2^2(\mu, \mu_k) + 2hE(\mu).$$

It follows that $\mu^* = \mathcal{T}\mu^*$ and

$$\begin{aligned} d(\mathcal{T}\mu^*, \mu^*) &\le d(\operatorname{Prox}^{\mathcal{W}}_{hE}(\mu^*), \mu^*) = d\Big(\arg\min_{\mu} \mathcal{W}_2^2(\mu, \mu^*) + 2h \underbrace{E(\mu)}_{0 \text{ as } \beta\to\infty}, \mu^*\Big), \\ &\le d\Big(\arg\min_{\mu} \mathcal{W}_2^2(\mu, \mu^*) = \mu^*, \mu^*\Big) \\ &\le 0 \end{aligned}$$

Distance is non-negative, so it must be that $\operatorname{Prox}^{\mathcal{W}}_{hE}(\mu^*) = \mathcal{T}\mu^* = \mu^*$. □

8.1.7 Sinkhorn's Algorithm

This section presents an algorithm for approximating the $\mathcal{W}_2^2$ distance through entropy regularization. Including entropy reduces the original Optimal Transport problem to one of matrix scaling. Sinkhorn's algorithm can be applied for this purpose to admit unique solutions.

The optimal value of the Kantorovich problem is the exact $\mathcal{W}_2^2$ distance. Given probability measures $\alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}$ and $\beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}$, the problem is to compute a minimum-cost mapping, π, defined as a non-negative matrix on the product space of atoms $\{x_1, \cdots, x_N\} \times \{y_1, \cdots, y_M\}$. Denoting the cost to move x_i to y_j as $C_{ij} = ||x_i - y_j||^2$, the problem is

$$\mathcal{W}_2^2(\alpha, \beta) = \min_{\pi \in \mathbb{R}_{\geq 0}^{N \times M}} \langle \pi, C \rangle = \sum_{ij} \pi_{ij} C_{ij}, \tag{8.2}$$

$$\text{such that } \pi \mathbf{1}_M = \alpha,\ \pi^\top \mathbf{1}_N = \beta. \tag{8.3}$$

This approach constitutes a linear program, which unfortunately scales cubically in the number of atoms. The complexity can be reduced by considering an entropically regularized version of the problem. Let ε be a regularization parameter. The new problem is written in terms of the generalized Kullback-Leibler (KL) divergence:

$$\mathcal{W}_2^2(\alpha, \beta) \approx \mathcal{W}_\varepsilon(\alpha, \beta) = \min_{\pi \in \mathbb{R}_{\geq 0}^{N \times M}} \langle \pi, C \rangle + \varepsilon \mathsf{KL}(\pi || \alpha \otimes \beta), \tag{8.4}$$

$$= \sum_{i,j} \pi_{ij} C_{ij} + \varepsilon \sum_{i,j} [\pi_{ij} \log \frac{\pi_{ij}}{\alpha_i \beta_j} - \pi_{ij} + \alpha_i \beta_j], \tag{8.5}$$

$$\text{such that } \pi \mathbf{1}_M = \alpha,\ \pi^\top \mathbf{1}_N = \beta. \tag{8.6}$$

The value of $\mathcal{W}_\varepsilon(\alpha, \beta)$ occurs necessarily at the critical point of the constrained objective function

$$\begin{aligned} L_\varepsilon = \sum_{i,j} \pi_{ij} C_{ij} + \varepsilon \sum_{i,j} [\pi_{ij} \log \frac{\pi_{ij}}{\alpha_i \beta_j} - \pi_{ij} + \alpha_i \beta_j] \\ - \sum_i f_i \Big(\sum_j \pi_{ij} - \alpha_i \Big) - \sum_j g_j \Big(\sum_i \pi_{ij} - \beta_j \Big), \quad (8.7) \\ \frac{\partial L_\varepsilon}{\partial \pi_{ij}} = 0 \implies \forall\ i, j,\ C_{ij} + \varepsilon \log \frac{\pi^*_{ij}}{\alpha_i \beta_j} = f^*_i + g^*_j. \quad (8.8) \end{aligned}$$

The last line of (8.8) shows that the entropically-regularized solution is characterized by two vectors $f^* \in \mathbb{R}^N$, $g^* \in \mathbb{R}^M$. With the following definitions

$$u_i = \exp(f^*_i/\varepsilon), \qquad v_j = \exp(g^*_j/\varepsilon), \qquad K_{ij} = \exp(-C_{ij}/\varepsilon), \quad (8.9)$$

one write the optimal transport plan as $\pi^* = \mathbf{diag}(\alpha_i u_i) K \mathbf{diag}(v_j \beta_j)$. Then the approximate Wasserstein distance can be computed simply as

$$\mathcal{W}_\varepsilon(\alpha, \beta) = \langle \pi^*, C \rangle + \varepsilon \mathsf{KL}(\pi^* || \alpha \otimes \beta) = \sum_{ij} (f^*_i + g^*_j) = \langle f^*, \alpha \rangle + \langle g^*, \beta \rangle$$

We mentioned that Optimal Transport reduces to positive matrix scaling. Indeed, using the vectors u and v, Sinkhorn's algorithm provides a way to iteratively scale K such that the unique solution is π^*. Initialize $u^{(0)} = \mathbf{1}_N$, and $v^{(0)} = \mathbf{1}_M$, then perform the following iterations for all i, j

$$\begin{aligned} v_j^{(1)} &= \frac{1}{[K^\top(\alpha \odot u^{(0)})]_j}, & u_i^{(1)} &= \frac{1}{[K(\beta \odot v^{(1)})]_i}, \\ &\vdots & &\vdots \\ v_j^{(n+1)} &= \frac{1}{[K^\top(\alpha \odot u^{(n)})]_j}, & u_i^{(n+1)} &= \frac{1}{[K(\beta \odot v^{(n+1)})]_i}. \quad (8.10) \end{aligned}$$

Sinkhorn's algorithm performs coordinate ascent with f and g to maximize the dual maximization problem

$$\mathcal{W}_\varepsilon(\alpha,\beta) = \max_{f\in\mathbb{R}^N, g\in\mathbb{R}^M} \langle f,\alpha\rangle + \langle g,\beta\rangle - \varepsilon\,\langle \alpha\otimes\beta, \exp\{(f\oplus g - C)/\varepsilon\} - 1\rangle\,. \tag{8.11}$$

Each update consists of kernel products, $K^\top(\alpha\odot u)$ and $K(\beta\odot v)$, and point-wise divisions. This procedure is described in Algorithm 11, using computations in the log domain to numerically stabilize the updates. The log updates derive from (8.9) and (8.12):

$$\log v_j = -\log\sum_i K_{ij}\alpha_i u_i \qquad\qquad \log u_i = -\log\sum_j K_{ij}\beta_j v_j,$$

$$\begin{aligned} g_j &= -\varepsilon\log\sum_i \exp\{(-C_{ij}+f_i)/\varepsilon + \log\alpha_i\} \\ f_i &= -\varepsilon\log\sum_j \exp\{(-C_{ij}+g_j)/\varepsilon + \log\beta_j\}. \end{aligned} \tag{8.12}$$

Sinkhorn iterations typically loop until convergence. In practice, a decreasing temperature sequence $\{\varepsilon_n\}$ is employed with which to bound the number of iterations.

Algorithm 11 Sinkhorn's Algorithm in the log domain for $\mathcal{W}_2^2$

```
1: input: Source and target measures $\alpha = \sum_{i=1}^N \alpha_i \delta_{x_i}$, $\beta = \sum_{j=1}^M \beta_j \delta_{y_j}$, Annealing
   temperature sequence $\{\varepsilon_n\}$
2: $i \in \{1, \cdots, N\}$, $j \in \{1, \cdots, M\}$
3: $f_i \leftarrow 0$, $g_j \leftarrow 0 \ \forall \ i, j$
4: for $\varepsilon \in \{\varepsilon_n\}$ do
5:     $C_{ij} = \frac{1}{2\varepsilon}||x_i - y_j||^2 \ \forall \ i, j$
6:     $g_j^{(n+1)} \leftarrow -\varepsilon \log \sum_i \exp\{(-C_{ij} + f_i^{(n)})/\varepsilon + \log \alpha_i\} \ \forall \ j$
7:     $f_i^{(n+1)} \leftarrow -\varepsilon \log \sum_j \exp\{(-C_{ij} + g_j^{(n+1)})/\varepsilon + \log \beta_j\} \ \forall \ i$
8: output: $\langle f, \alpha \rangle + \langle g, \beta \rangle$
```

8.1.8 Proof that the Gibbs measure minimizes free energy.

Remark 1. *Let $E(\mu) = F(\mu) + \beta^{-1}H(\mu)$, with $F(\mu) = \int U(z)d\mu$. The minimizer is the Gibbs density,*

$$\mu_*(z) = Z^{-1} \exp\{-\beta\psi(z)\},$$

where $\psi(z) = U(z) + \int_0^1 \lambda(\tau)S(z,\tau)d\tau$, and $Z = \int \exp\{-\beta\psi(z)\}dz$.

Proof. Set the functional derivative of E to zero and solve for μ. The derivatives are

$$\frac{\delta F}{\delta \mu} = U(z), \qquad \frac{\delta H}{\delta \mu} = \log(\mu) + 1.$$

Solving for μ_* emits a proportionality, which can be normalized as described:

$$U(z) + \beta^{-1}(\log(\mu_*) + 1) = 0 \implies \mu_* \propto \exp\{-\beta\psi(z)\}$$

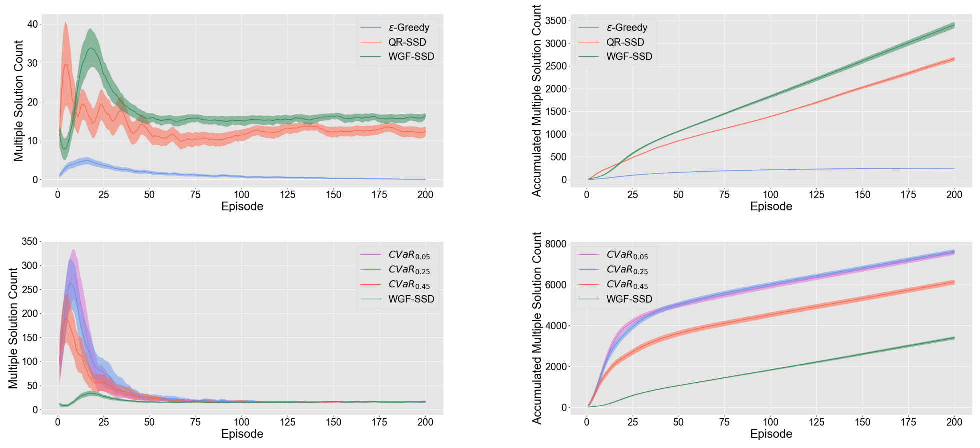


Figure 8.3: Frequency of multiple-solution events that occurred during the Control in the Presence of Uncertainty experiment.

□

8.1.9 On the prevalence of multiple solutions

Figure 8.3 shows the number of times SSD conditions were detected during the Control in the Presence of Uncertainty experiment. This data shows that multiple solutions will occur often enough in the considered domain to justify SSD action selection.

8.2 Chapter 5 Appendix

8.2.1 Gaussian Process Parameter Optimization

During the time this work was completed, most software packages required an objective function and its gradient to solve optimization problems. Below, we provide the full details of the predicted value function's log likelihood gradient computation.

Objective Gradient: Let $\mathbf{K}_r = \mathbf{Q} + \sigma^2\mathbf{I} + \mathbf{K}_{ru}\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}$ and ξ_j be the j-th optimization variable. The gradient with respect to ξ_j is

$$\frac{\partial \mathcal{L}}{\partial \xi_j} = -\frac{1}{2}\mathbf{tr}(\mathbf{K}_r^{-1}\mathbf{J}_r) + \frac{1}{2}\mathbf{r}^\top\mathbf{K}_r^{-1}\mathbf{J}_r\mathbf{K}_r^{-1}\mathbf{r}. \tag{8.13}$$

Here, $\mathbf{J}_r$ is the tangent matrix of $\mathbf{K}_r$ with respect to ξ_j. The full equations for computing $\mathbf{J}_r$ are described in the next section.

Likelihood Covariance Tangent Matrix: Denote ξ_j to be a generic optimization variable. Then the matrix $\mathbf{J}_r$ used in Equation 8.13 is given by

$$\begin{aligned}
\mathbf{J}_r = \frac{\partial \mathbf{Q}}{\partial \xi_j} + \frac{\partial}{\partial \xi_j}\sigma^2\mathbf{I} + \mathbf{K}_{ru}\frac{\partial}{\partial \xi_j}(\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}) + \mathbf{J}_{ru}(\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}),\\
\frac{\partial \mathbf{Q}}{\partial \xi_j} = \mathrm{diag}\big(\mathbf{J}_{rr} - \mathbf{K}_{ru}\frac{\partial}{\partial \xi_j}(\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}) - \mathbf{J}_{ru}(\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur})\big),\\
\frac{\partial}{\partial \xi_j}(\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}) = \mathbf{K}_{uu}^{-1}\mathbf{J}_{ur} + \frac{\partial \mathbf{K}_{uu}^{-1}}{\partial \xi_j}\mathbf{K}_{ur},\\
\frac{\partial \mathbf{K}_{uu}^{-1}}{\partial \xi_j} = -\mathbf{K}_{uu}^{-1}\mathbf{J}_u\mathbf{K}_{uu}^{-1}.
\end{aligned}$$