DEEP EXPLORATION VIA RANDOMIZED VALUE FUNCTIONS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF
MANAGEMENT SCIENCE AND ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Ian Osband

December 2016

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Benjamin Van Roy) Principal Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(John Duchi)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Ramesh Johari)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Mykel Kochenderfer)

Approved for the Stanford University Committee on Graduate Studies

Abstract

The "Big Data" revolution is spawning systems designed to make decisions from data. Standard statistical and machine learning techniques have made great strides in prediction and estimation from any fixed dataset. However, if you want to learn to take actions, where your choices can affect the underlying system and the data you observe, you need reinforcement learning. Reinforcement learning builds upon learning from datasets, but additionally address issues of partial feedback and long term consequences. In a reinforcement learning problem the decisions you make may affect the data you get, and even alter the underlying system in future timesteps. Statistically efficient reinforcement learning requires "deep exploration" or the ability to plan to learn. Previous approaches to deep exploration have not been computationally tractable beyond small scale problems. For this reason, most practical implementations use statistically inefficient methods for exploration such as ϵ-greedy dithering, which can lead to exponentially slower learning.

In this dissertation we present an alternative approach to deep exploration through the use of randomized value functions. Our work is inspired by the Thompson sampling heuristic for multi-armed bandits. We provide insight into why this algorithm can be simultaneously more statistically efficient and more computationally efficient than existing approaches. We leverage these insights to establish several state of the art theoretical results and performance guarantees. Importantly, and unlike previous approaches to deep exploration, this approach also scales gracefully to complex domains with generalization. We support our analysis through extensive experiments; these include several didactic examples as well as a recommendation system, Tetris, and Atari 2600 games.

Acknowledgments

This dissertation is the product of a four year collaboration with my advisor, Benjamin Van Roy. Ben is a true legend of the field and I am incredibly lucky to have worked so closely wth him. Ben's clarity of thought and his vision towards greater goals has been a constant inspiration to me. What's even better is that working with Ben has also been really fun - I'll always be grateful for his support and guidance.

Far before any of this dissertation business started the biggest influence on my academic life has to be from my dad Kent. He's always tried to make learning fun and I know that there's no chance I would have made it through even half of my challenges without him. At the same time, I'm incredibly lucky to have my mum Jan to balance out any rough edges to give me enough love and support to try keep me somewhat balanced.

I would like to thank my lab partners Zheng Wen and Dan Russo for the many hours of discussion we have shared. Working together with them has both greatly shaped my thinking and work in this dissertation and I've really enjoyed the journey together. Another thank you to my dissertation committee Ramesh Johari, John Duchi, Mykel Kokenderfer and chair Percy Liang for all your help and guidance in preparing this dissertation.

Outside of research, I've been incredibly lucky to find a group of friends at Stanford to enrich my life. I'd like to give a special shout out to Justin Sirignano, Gustavo Schwenkler and Enzo Busseti within the Economics and Finance group where I began my Ph.D; David Luenberger for convincing me to come to Stanford and MS&E; Scott Wong for all our times getting started with "machine learning"; the Stanford GSB rugby team for all the good times and even an individual one for the mate Clark Burckle. Finally, a huge thank you to my collaborators at Google and Deepmind, where I rather optimistically began working before this dissertation was completed. Next steps... solve artificial intelligence!

Contents

Chapter 1

Introduction

1.1 Learning to act

The "Big Data" revolution is spawning systems designed to make decisions from data. There is hope that we might leverage this data to improve the quality of our decisions across a wide range of important fields. From healthcare, to agriculture to robotics the potential impact for effective artificial intelligence is huge. Advances in statistics and machine learning have made great strides to extract actionable insights from large datasets together with powerful computation. The key ingredient for effective learning from datasets is effective *generalization*, which allows an algorithm to perform well even in situations which are not exactly the same as the training data.

Standard statistical techniques, including both supervised and unsupervised learning, focus on learning from a fixed dataset. More and more, automated systems are deployed to make decisions proactively and which affect the data which is gathered. In such settings, we must assess these algorithms not only in terms of the quality of their decisions given their initial data, but also in terms of the quality of the data which they gather for future decisions. If you want to make good decisions from data, you need good data. Even an optimal algorithm for learning from any fixed dataset may lead to substantially suboptimal performance in systems where the data you gather depends on the decisions you make. Reinforcement learning (RL) is a framework for learning to act in systems where the decisions you make can affect the data you get; this problem formulation extends upon learning from datasets in two key ways.

The first extension which reinforcement learning adds to learning from datasets is *partial feedback*; an agent may only gather data about the actions it actually chooses. This leads to a fundamental tradeoff of exploration vs exploitation; the agent may be able to gather better data by exploring its poorly-understood actions, but it might be able to achieve better short term performance by exploiting its existing knowledge. This optimal experimentation problem has been most thoroughly addressed in the multi-armed bandit literature, which combines learning from datasets with decision making under partial feedback.

The second extension which reinforcement learning adds is *long term consequences*; an agent may make decisions at one timestep that actually alter the underlying system for the future. This leads to the planning problem of how to influence the underlying system to maximize long run cumulative rewards, rather than each timestep myopically. This planning problem has been most thoroughly addressed in the control literature, which seeks to optimize a known underlying system.

Reinforcement learning combines learning from datasets with partial feedback and long term consequences. The RL problem is the problem of learning to control an unknown system. We present three simple examples where all three of these aspects will be important.

- **Personalized medicine**, to enhance a patient's wellbeing throughout their lifetime.
- **Agriculture**, to maximize crop yields over an entire season or even multiple years.
- **Online services**, to optimize customer interactions over multiple sessions.

If you want to design a general artificial intelligence which learns to act then learning from datasets is not enough; you need RL. In particular, in order to learn efficiently, an algorithm must direct its exploration towards this potentially informative data. In fact, for environments with long term consequences it is not enough for this exploration to be directed at each individual timestep; the exploration must also be deep.

1.2 Deep exploration

Traditional methods for learning to act typically address the exploration-exploitation problem in two separate stages: estimation then optimization. This approach may be extremely suboptimal. Too much experimentation means the agent doesn't optimize its short term performance. Too little experimentation may mean the agent doesn't gather enough data to

make the right decision. Even worse, it may be difficult to perform meaningful exploration in complex systems before the problem is at least somewhat well-understood.

This exploration-exploitation dilemma has been most thoroughly studied in a stylized problem known as the multi-armed bandit (Thompson, 1933). In these problems an agent learns to optimize a fixed (but unknown) function through sequential periods of interaction. The most naive solution is given by "greedy" methods, which produce an estimate for the function and then choose the action which is optimal given this estimate. Unfortunately, these types of strategies may *never* learn the optimal policy, since they may converge to suboptimal decisions and then never gather the necessary data to learn. To combat this premature convergence, the most common strategy is to inject some additional random actions or "dithering" for exploration. Dithering strategies will *eventually* learn the optimal policy, since they will sample every action infinitely often, but can be far less efficient than optimal allocation rules (Lai and Robbins, 1985). Contrasted with dithering are *directed* strategies for exploration, which intelligently balance their choices between potentially exploitative and informative actions at each timestep. In bandit learning, the choice of directed exploration rather than undirected generally categorizes efficient algorithms (Thrun, 1992).

Reinforcement learning (RL) extends upon the multi-armed bandit to allow long term consequences; actions at one timestep can influence the system at future timesteps. Unlike bandit learning, which balances actions which are immediately rewarding or immediately informative, this may require planning over several time steps (Kearns and Singh, 2002). For exploitation, this means that an efficient agent must consider the future value of an action over several future time steps and not simply the myopic rewards. In exactly the same way, efficient exploration may require taking actions which are neither immediately rewarding, nor immediately informative but instead may lead towards some potentially informative future state. As a result, and unlike the bandit setting, directed exploration at each individual timestep is not enough to guarantee efficiency; efficient exploration in RL must also be deep. Deep exploration means exploration which is directed over multiple time steps; it can also be called "planning to learn" or "far-sighted" exploration.

To illustrate this concept of deep exploration, consider a simple deterministic Markov decision process (MDP) with states $\{s_{-3}, s_{-2}.., s_{+3}\}$ arranged in a chain. The agent begins in state s_0 and must interact with the MDP over three timesteps. The agent has a choice of actions "left" and "right" with deterministic and known transition dynamics. All states

have zero reward, except for the leftmost state s_{-3} which has known reward $\epsilon > 0$ and the rightmost state s_3 which will provide some information if visited. In particular, this information might be the unknown reward of visiting state s_3. However, for this example we will keep the concept of information more abstract so that we do not yet need to consider *how* to balance exploration with exploitation. Our only hope with this example is to highlight that just like it may be necessary to plan for future rewards, it may be necessary to plan to learn. We illustrate this environment in Figure 1.1. The action "left" is gray, the action "right" is black. Rewarding states are shown in red, informative states in blue.

An agent which begins at state s_0 cannot take any action to obtain a reward or information at timestep $t = 0$. In order to reach either a rewarding state or an informative state within three steps from s_0 the agent must plan a consistent strategy over several time steps. In Figure 1.2 we depict the planning and look ahead trees for several stylized algorithmic approaches to reinforcement learning in this environment. We use dashed lines passing backwards through time to indicate planning for rewards and information respectively and describe the meaning of each diagram below:

(a) A standard supervised learning algorithms for rewards would only recognize the value of action "left" in state s_{-2}. All other actions in all other states produce zero immediate reward and so would not be valued by a supervised learning algorithm for the immediate rewards at each timestep.

(b) A bandit algorithm for rewards balances potentially exploitative and exploratory actions at each timestep. However, only the action "left" in s_{-2} and "right" in s_2 are immediately rewarding or informative. All other actions in all other states are neither rewarding nor informative, so are not valued by a standard bandit algorithm.

(c) Reinforcement learning with dithering can plan for future rewards by heading left in states $\{s_{-2}, s_{-1}, s_0\}$, but does not account for the potential benefit of exploration.

(d) RL with shallow exploration can plan for rewards as above, but can also value the information gain of "right" in state s_2. However, shallow exploration does not propagate the potential benefit of "right" in $\{s_0, s_1\}$, which although not immediately informative could take the agent towards the information in s_2.

(e) Reinforcement learning with deep exploration is able to plan ahead for *both* rewards and exploration. This means that the agent can value the future information which actions "left" may provide, as well as the future rewards of "right".

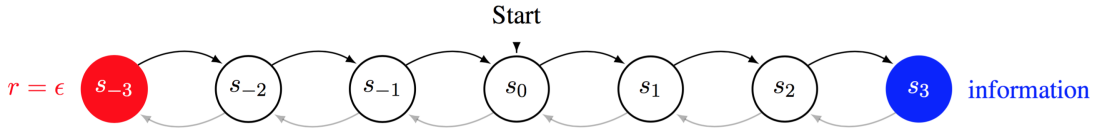
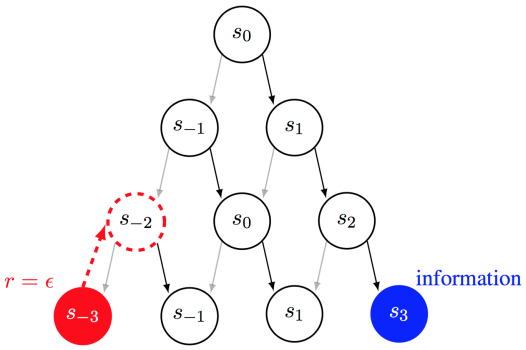
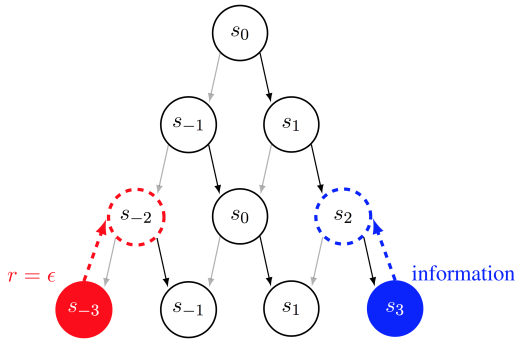


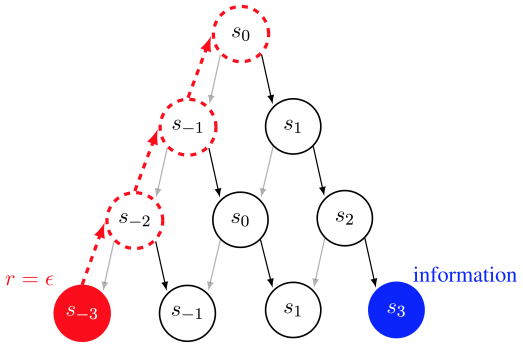
Figure 1.1: An example environment designed to illustrate deep exploration.



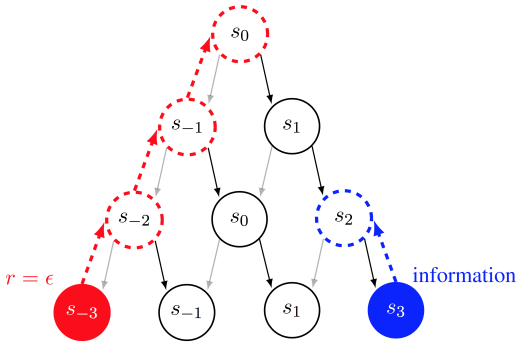
(a) Supervised learning to learn the immediate rewards



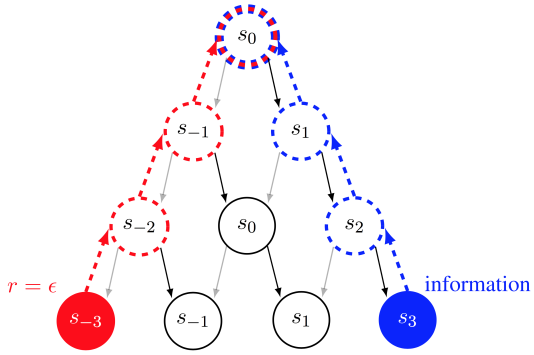
(b) Bandit algorithm for immediate rewards and immediate information



(c) RL for rewards + dithering



(d) RL for rewards + shallow exploration



(e) RL for rewards + deep exploration

Figure 1.2: Lookahead trees for the environment in Figure 1.1 over three timesteps. Dashed lines indicate planning for future rewards and/or information.

The name "deep exploration" is relatively new, but the observation that temporally extended exploration is necessary for efficient reinforcement learning has been known for a long time (Brafman and Tennenholtz, 2002). Despite this observation, the many high profile applications of RL have either been purely exploitative (Tesauro, 1995) or employed inefficient dithering schemes for exploration (Mnih et al., 2015). At the same time, algorithms which do exhibit deep exploration can learn exponentially faster (Kearns and Singh, 2002) or even tackle extremely complex learning problems (Silver et al., 2016). One reason many practitioners overlook the benefits of deep exploration is that, although these algorithms may be much more statistically efficient, their implementation is often computationally intractable. In this dissertation we present a computationally efficient approach to deep exploration via randomized value functions.

1.3 Randomized value functions

The most common approach to learning to act is the classical "estimate then optimize". This approach uses a point estimate for unknown parameters and then plans with respect to this estimate together with some dithering for exploration. This approach is practical, but since it does not perform deep exploration it can be extremely statistically efficient. At the other end of the spectrum, a principled approach to statistically efficient learning is given by the Bayesian perspective. Here the agent models its uncertainty by a prior distribution. The agent then explicitly plans ahead to balance the potential information and rewards of future actions and observations. This framing is provably optimal, but for anything beyond tiny problems it will be computationally intractable.

In this dissertation we present an alternative approach that aims to be simultaneously computationally and statistically efficient: randomized value functions. This algorithm is inspired by the Thompson sampling, or probability matching, heuristic for multi-armed bandits. At a high level, this principle suggests:

> "Randomly select a policy according to the probability it is the optimal policy."

This algorithm is somewhat similar to the Bayesian approach, since we replace a point estimate of the value with a distribution over potential values. However, instead of attempting to find the optimal policy over the entire *distribution* of beliefs, the agent simply follows

the policy which is optimal for a single *sample* from this distribution. This exploration is directed, since the random policy is sampled according to the posterior probability it is the best policy. This exploration is deep, since a single sample drives consistent exploration over several timesteps. In this dissertation we will show that this simple and intuitive approach can drive algorithms with both theoretical guarantees and state of the art empirical performance.

1.4 Organization

We begin our investigation to efficient reinforcement learning in the setting of *tabula rasa*, or learning from "blank slate". In this formulation the agent has little or not prior knowledge, but can learn through its interaction with the environment. We further specialize our consideration in this Part I to tabular representations, where each state and action is learned independently. This formulation may not be practical for large systems, since it does not allow for generalization across the environment. However, it allows us to study the RL problem in a simpler setting for analysis and with a general framework for learning *any* underlying system. **Chapter 2** introduces this tabular problem in terms of a unknown finite Markov decision process (MDP). We outline several standard approaches to tabular reinforcement learning and highlight the ways in which many common approaches to learning can be statistically and/or computationally intractable.

In **Chapter 3** we introduce the principle of "optimism in the face of uncertainty", which has been the driving principle behind almost all algorithms which are simultaneously statistically and computationally efficient. We show how optimism can drive efficient and deep exploration with a particular focus on the state of the art optimistic approaches based upon *upper confidence bounds for reinforcement learning* (UCRL). We review the analytical techniques than can be used to design near-optimal implementations of UCRL, but also highlight several of the shortcomings of these optimistic methods. We introduce the concept of *stochastic optimism* which generalizes the optimistic principle from point estimates to distributions.

In **Chapter 4** we introduce our first approach to deep exploration via randomized value functions, *posterior sampling for reinforcement learning* (PSRL). This algorithm samples a randomized value function from the posterior distribution and follows the policy which is

optimal for this sample for the duration of the episode. We show that this algorithm, which is a natural adaptation of Thompson sampling to the reinforcement learning problem, satisfies similar regret bounds to any well-designed optimistic algorithm. We analyze PSRL as a stochastically optimistic algorithm and provide new insight and analysis which explains why PSRL is simultaneously more statistically efficient and computationally efficient than *any* other existing optimistic algorithm. These theoretical results are backed up with extensive experiments and simulations in didactic tabular domains.

The tabula rasa formulation is an important tool for insight and analysis, but practical reinforcement learning in large scale systems requires generalization. The second part of our dissertation focuses on settings where the underlying environment has some low-dimensional structure which facilitates generalization. We show that, when this is the case, an efficient algorithm for RL can exploit this low-dimensional structure to learn in data that scales with the dimensionality, rather than the cardinality, of the underlying system. **Chapter 5** focuses on the setting of factored MDPs, which exhibit some conditional independence structure. In **Chapter 6** we extend this analysis to any parameterized MDP and characterize the complexity of the learning problem in terms of the eluder dimension of the function classes to be learned. Further, we show that in both settings the same simple and intuitive algorithm, PSRL, satisfies these strong regret bounds and at a computational cost no greater than solving a single known MDP. This leads to new state of the art theoretical bounds across a wide range of environments, which can be exponentially better than learning from *tabula rasa*.

Part II shows that PSRL successfully reduces learning to planning with near-optimal statistical efficiency and a computational cost no greater than solving a single known MDP. However, for large systems with complex generalization even the planning problem itself can be computationally intractable. In Part III we focus on settings with value-based generalization, that can directly combine the learning and planning steps without the need for explicit MDP planning or exact posterior influence over models for the underlying MDP.

In **Chapter 7** we investigate learning with a linear basis function representation for the value function. Our algorithm, *randomized least squared value iteration* (RLSVI) iteratively approximates the intractable posterior over value functions through Gaussian sampling. We show that, in the tabular setting of independent basis functions, this algorithm recovers state of the art regret bounds for the *tabula rasa* setting. However, unlike PSRL this

algorithm naturally facilitates generalization via a linearly parameterized value function. We demonstrate through simulation that this algorithm can demonstrate efficient directed, and deep, exploration in settings where existing dithering methods are provably inefficient. We demonstrate the practical applicability and scalability of this algorithm in learning to play the game Tetris as well as an example recommendation system.

Finally, in **Chapter 8** we show that deep exploration via randomized value functions can be effectively applied with nonlinearly parameterized value functions. We propose *bootstrapped Q learning* to use the nonparametric bootstrap to give approximate posterior estimates for the value function even when the underlying model for generalization is nonlinear and complex. In particular, we focus on the setting where the value function is estimated by some "deep" neural network for our algorithm *bootstrapped deep Q networks* (DQN). We demonstrate that this algorithm can efficiently synthesize deep exploration with deep learning even in complex stochastic environments. This method is also scalable to large systems; we evaluate our algorithm on the Arcade Learning Environment where we learn to play over 50 Atari games from raw pixel input. Bootstrapped DQN significantly improves learning time and performance across most Atari games.

1.5 Contributions

We now summarize the key contributions of this dissertation,

- We bring attention to the importance of deep exploration. Crucially, we give a clear name to this concept of "planning to learn" which is essential for statistically efficient reinforcement learning.
- We highlight several shortcomings of existing optimistic approaches to exploration, both in terms of statistical and computational efficiency. We present randomized value functions as a unifying approach to simultaneously statistically and computationally efficient reinforcement learning.
- We introduce the concept of *stochastic optimism*, which extends the optimistic principle from point estimates to distributions. We use this concept to relate exploration via randomized value functions to optimistic algorithms. This leads to the first $\tilde{O}\left(\sqrt{SAT}\right)$ bounds on the Bayesian expected regret for any RL algorithm, where S is

the number of states, A is the number of actions and T is the time elapsed. Further, the proposed algorithm *posterior sampling for reinforcement learning* (PSRL) has a computational cost no greater than solving a single known MDP.

- We show that this same computationally tractable algorithm (PSRL) extends naturally to domains with generalization. We present the first general regret bounds for reinforcement learning in terms of the dimensionality, rather than the cardinality of the underlying environment.
- We prove that this approach to deep exploration can be effective even when the random samples are not drawn from the correct posterior distribution for the value function. In particular, we present simple and computationally tractable approaches to deep exploration via randomized value functions in domains with value-based generalization where exact posterior inference is intractable. We show that these approximations can be provably-efficient in certain settings.
- We demonstrate through extensive simulation that deep exploration via randomized value functions can lead to exponentially faster learning in complex domains with generalization when compared to the previous state of the art. We present the first computationally tractable approach to deep exploration in these domains and state of the art results in Tetris, a recommendation system and Atari 2600 games.

Part I

Tabula Rasa

Chapter 2

Tabular Reinforcement Learning

We begin our investigation on reinforcement learning in the *tabula rasa* or "blank state" setting, where little or no prior knowledge is available to the agent. In this setting the agent must learn about every aspect of the environment using a separate tabular encoding for each state and action. An algorithm which can learn from *tabula rasa* is general, since it can be applied to any possible system, from driving a car to making medical diagnoses to composing a Grammy-winning album. We might hope that a well-designed algorithm could learn to make good decisions without prior knowledge no matter the underlying environment.

Perhaps surprisingly, we will outline several existing algorithms which are already guaranteed to learn the optimal solution for any environment. The problem is that although these algorithms are guaranteed to learn the optimal solution, the amount of data and/or computational resources which they require is practically intractable for anything beyond the simplest of settings. In fact, there are fundamental lower bounds on the amount of data and computation required for any algorithm to learn the optimal policy in any environment. Based upon these results we will see that *tabula rasa* reinforcement learning is not a practical approach to large scale problems. However, the insights and analysis we can perform in this simple setting will provide an invaluable opportunity to develop our understanding of the essential problems in reinforcement learning. The purpose of this chapter is to introduce the basic problem formulation and to motivate the essential issues which we will adress in later chapters of this dissertation. In the remainder of this dissertation we will extend these insights to more practical problems.

In this chapter, we formally introduce the problem of reinforcement learning in terms of an unknown finite Markov decision process (MDP). We review some standard definitions and results from existing literature together with several criteria for efficient reinforcement learning. We consider the problem of learning to optimize an unknown MDP with little or no prior knowledge, a setting which we call *tabula rasa*. Next, we review several lower bounds which apply to *any* algorithm for RL in these environemnts, this will allow us to assess our subsequent upper bounds to show that they are near optimal. Finally, we discuss several common algorithmic approaches to RL with special attention to the shortcomings of these methods.

2.1 Problem formulation

We consider the problem of learning to optimize a random finite horizon MDP $M = (\mathcal{S}, \mathcal{A}, R^M, P^M, H, \rho)$ in repeated finite episodes of interaction. $\mathcal{S} = \{1, .., S\}$ is the state space, $\mathcal{A} = \{1, .., A\}$ is the action space, H is the length of the episode and ρ is the initial state distribution. At the start of each episode the initial state s_1 is drawn from the distribution ρ. At each timestep $h = 1, .., H$ within an episode the agent observes state $s_h \in \mathcal{S}$, selects action $a_h \in \mathcal{A}$, receives a reward $r_h \sim R^M(s_h, a_h)$ in $[0, 1]$ and transitions to a new state $s_{h+1} \sim P^M(s_h, a_h)$. We define the MDP and all other random variables we will consider with respect to a probability space $(\Omega, \mathcal{F}, \mathbb{P})$.

A deterministic policy μ is a function mapping each state $s \in \mathcal{S}$ and timestep $h = 1, .., H$ to an action $a \in \mathcal{A}$. For each MDP M and policy μ we define the Q-value function from step h:

$$Q^M_{\mu,h}(s, a) := \mathbb{E}_{M,\mu}\left[\sum_{j=h}^{H} \overline{r}^M(s_j, a_j) \Big| s_h = s, a_h = a\right], \tag{2.1}$$

where $\overline{r}^M(s, a) = \mathbb{E}[r | r \sim R^M(s, a)]$. The expectation indicates that the state $s_h = s$, the action $a_h = a$ and thereafter actions are selected according to the policy μ. We define the value function

$$V^M_{\mu,h}(s) := Q^M_{\mu,h}(s, \mu(s, h)). \tag{2.2}$$

A policy μ is optimal for the MDP M if $V^M_{\mu,h}(s) = \max_{\mu'} V^M_{\mu',h}(s)$ for all $s \in \mathcal{S}$ and $h = 1, \ldots, H$. We will associate with each MDP M a policy μ^M that is optimal for M.

Let $\mathcal{H}_t = (s_1, a_1, r_1, \ldots, s_{t-1}, a_{t-1}, r_{t-1})$ denote the history of observations made *prior* to time t. To highlight this time evolution within episodes we will index $s_t = s_{kh}$ for the state at timestep t which is h steps into the k^{th} episode or $t = (k-1)H + h$ and similarly for $\mathcal{H}_{kh}$. A reinforcement learning algorithm is a deterministic sequence $\{\pi_k | k = 1, 2, \ldots\}$ of functions, each mapping $\mathcal{H}_{k1}$ to a probability distribution $\pi_k(\mathcal{H}_{k1})$ over policies which the agent will employ during the kth episode. We define the regret incurred by a reinforcement learning algorithm π up to time T to be:

$$\text{Regret}(T, \pi, M^*) := \sum_{k=1}^{\lceil T/H \rceil} \Delta_k, \tag{2.3}$$

where Δ_k denotes regret over the kth episode, defined with respect to the MDP M^* by

$$\Delta_k := \sum_{\mathcal{S}} \rho(s)(V^{M^*}_{\mu^*,1}(s) - V^{M^*}_{\mu_k,1}(s)) \tag{2.4}$$

with $\mu^* = \mu^{M^*}$ and $\mu_k \sim \pi_k(\mathcal{H}_{k1})$. Note that regret is not deterministic since it can depend on the random MDP M^*, the algorithm's internal random sampling and, through the history $\mathcal{H}_{k1}$, on previous random transitions and random rewards. We will assess and compare algorithm performance in terms of regret and its expectation.

2.1.1 Infinite horizon problem

Section 2.1 presents a reinforcement learning problem in terms of the sum of expected rewards through time, with a fixed (and known) horizon $H < \infty$. However, the setting without episodic reset and $H = \infty$ is also of significant interest in both reinforcement learning and control (Bertsekas et al., 1995; Sutton and Barto, 1998). In fact, the finite horizon problem is a special case of an infinite horizon problem. To see this clearly note that we can augment the state $\tilde{s} = (s, h)$ so that even time-inhomogeneous finite horizon problems can be expressed as homogeneous with infinite horizon. Dealing with the infinite horizon problem properly raises several technical issues. In particular, one important difference is that the value function given by (2.2) may no longer be well-defined. There are two standard approaches to dealing with this issue: discounted reward and average reward.

In the discounted reward setting, the future rewards are discounted by a factor of $\gamma \in [0, 1)$ for each timestep. This means that immediate rewards are valued higher than future rewards. The γ-discounted value of a policy μ in an MDP M starting from s is given

$$V^M_{\mu,\gamma}(s) := \mathbb{E}_{M,\mu}\left[\sum_{t=1}^{\infty} \gamma^t \overline{r}^M(s_t, a_t) \Big| s_1 = s\right]. \tag{2.5}$$

This expression is well-defined and bounded between zero and $\frac{1}{1-\gamma}$. A policy μ is optimal for the MDP M and discount rate γ if $V^M_{\mu,\gamma}(s) \geq \max_{\mu'} V^M_{\mu,\gamma}(s)$ for all $s \in \mathcal{S}$. In fact, discounted problems lend themselves to an effective horizon length $\tilde{H} \simeq \frac{1}{\gamma}$, since after for any $t > \tilde{\mathcal{H}}$, $\gamma^t \ll 1$. In fact, there are several more connections between discounting and finite horizon problems (Bertsekas and Tsitsiklis, 1996). We choose to focus on the finite horizon problem since we find it is more amenable to clean analytical results, as we justify in Section 2.2. Exploring the connections between the discounted problem formulation and finite horizon problems is an interesting area and active area of research (Dann and Brunskill, 2015).

In the average reward setting, the value of a policy μ in an MDP M is the long run expected average of rewards starting from state s

$$V^M_{\mu,\text{ave}}(s) := \lim_{T\to\infty} \mathbb{E}_{M,\mu}\left[\frac{1}{T}\sum_{t=1}^{T} \overline{r}^M(s_t, a_t) \Big| s_1 = s\right]. \tag{2.6}$$

This expression is well-defined and bounded between zero and one. A policy μ is optimal for the MDP M if $V^M_{\mu,\text{ave}}(s) \geq \max_{\mu'} V^M_{\mu,\text{ave}}(s)$ for all $s \in \mathcal{S}$. Under some suitable technical conditions, for example if MDP M is weakly communicating (Bartlett and Tewari, 2009), this optimal average value is independent of initial state so that $\rho(M) = \rho(M, s) = \max'_\mu V^M_{\mu'}(s)$ for all $s \in \mathcal{S}$. Extending analytical results from finite horizon to infinite horizon average reward can be a significant undertaking. In many cases the infinite horizon problem requires special consideration and a naive application of methods tailored to the formulation in Section 2.1 can lead to delicate errors (Osband and Van Roy, 2016a). We believe that, at a high level, many of the key problem insights from finite horizon problems will transfer to the infinite horizon problem. Or perhaps at least, in many practical applications, it may be sufficient to artificially model a problem with an artificial finite horizon. Formalizing and understanding these limitations is another important area for future research, but one which is outside the scope of this dissertation.

2.2 Objectives in the design of learning algorithms

Now that we have defined the reinforcement learning problem, our next step is to consider what would make a "good" reinforcement learning algorithm. In this section we will review some of the standard objectives which we would like our learning algorithms to obey. For clarity, we will consider these objectives in terms of a tabular reinforcement learning algorithm, which builds up estimates for each state and action independently. However, the high level objectives which we discuss (namely computational and statistical efficiency) will *not* be confined to the tabular setting. The objectives we consider in this section will help to guide our design and analysis of algorithms for all forms of reinforcement learning.

2.2.1 Regret bounds

We want to design algorithms which will learn to make good decisions through time. We assess the quality of the decisions through the cumulative sum of rewards which the agent receives. However, the reward attained by an algorithm π depends not only on the quality of the algorithm but also the unknown MDP M^*. To assess the performance of a learning algorithm given its environment we focus upon the regret as defined in (2.3). The regret quantifies the sub-optimality of the policies chosen by the learning algorithm π versus the optimal policy up to time T. Maximizing the cumulative sum of rewards is equivalent to minimizing the regret.

For provably efficient reinforcement learning, we would like to establish some guarantee that bounds the regret of an algorithm π. In general, for stochastic and unknown environments, it may not be possible to guarantee low regret. However, we might hope that we can bound the *expected* regret to time T,

$$\mathbb{E}\left[\text{Regret}(T, \pi, M^*)\right] \leq f_\pi(u_\mathcal{M}, T). \tag{2.7}$$

In this dissertation we will focus on the Bayesian expected regret where this expectation is taken with respect to the unknown MDP $M^* \sim \phi$ is drawn from the prior distribution, as well as the noise in π and M^*. This is importantly different from other works which bound the worst case frequentist regret for any specific $M^* \in \mathcal{M}$ (Jaksch et al., 2010), or risk-sensitive utility functions (Artzner et al., 1999).

The function $f_\pi(u_\mathcal{M}, T)$ provides the strength of our analytical guarantees given by (2.7). By our problem definition regret is always bounded $\text{Regret}(T, \pi, M^*) \leq T$ trivially. An algorithm π is asymptotically no-regret if $\lim_{T\to\infty} \frac{f(u_\mathcal{M},T)}{T} = 0$, or $f_\pi(u_\mathcal{M}, T) = o(T)$. An algorithm which is asymptotically no-regret is guaranteed to learn the optimal policy eventually, but "eventually" might take an extremely long time.

2.2.2 Statistical efficiency

Another similar line of research has focused upon various flavors of "probably approximately correct" (PAC) bounds for reinforcement learning (Brafman and Tennenholtz, 2002; Kearns and Singh, 2002; Strehl and Littman, 2005; Strehl et al., 2006). The precise definition of these bounds vary from paper to paper but generally they bound the number of suboptimal actions taken by the agent. For example, an algorithm may guarantee that the number of ϵ-suboptimal episodes is bounded, for any $\epsilon > 0$

$$\mathbb{E}\left[\sum_{k=1}^{\infty} \mathbb{1}\{\Delta_k > \epsilon\}\right] < f(u_\mathcal{M}, \epsilon). \tag{2.8}$$

This notion of statistical efficiency is clearly quite similar to notions of regret bounds, but for some reason the two literatures are somewhat disconnected (Jaksch et al., 2010; Lattimore et al., 2013; Dann and Brunskill, 2015). In this dissertation we focus on the regret, since its objective is more closely aligned with the standard RL objective. However, we will now present two simple equivalences that allow comparison between the quality of bounds on regret and PAC-style bounds in the case of finite horizon episodic MDPs. In infinite horizon problems the differences can be more subtle, since PAC-MDP guarantees typically only hold with respect to the states visited by the agent, unlike regret bounds which hold relative to the optimal policy. Generally speaking, regret bounds are provide stronger guarantees than PAC analyses. We present more detailed discussion of these issues in Section 4.4.2.

Proposition 2.1 (Regret bounds imply PAC bounds)**.**
Let π be a learning algorithm, $\alpha > 1$ and $K > 0$ such that for timesteps $T > 0$,

$$\mathbb{E}\left[\text{Regret}(T, \pi, M^*)\right] \leq KT^{1/\alpha}. \tag{2.9}$$

Then for any $\epsilon > 0$, for all $T' > \left(\frac{K}{\epsilon}\right)^{\frac{\alpha}{\alpha-1}}$ the average regret per timestep is bounded,

$$\mathbb{E}\left[\frac{\text{Regret}(T', \pi, M^*)}{T'}\right] \leq \epsilon. \tag{2.10}$$

Proof. Simple algebraic manipulation. □

Proposition 2.2 (PAC bounds imply regret bounds)**.**
Let π be a learning algorithm, $\beta > 0$ and $L > 0$ such that for all $\epsilon > 0$,

$$\mathbb{E}\left[\sum_{k=1}^{\infty} \mathbb{1}\{\Delta_k > \epsilon\}\right] < L\epsilon^{-\beta}. \tag{2.11}$$

Then for all $T > 0$, the regret of algorithm π is bounded,

$$\mathbb{E}\left[\text{Regret}(T, \pi, M^*)\right] \leq (\beta + 1)\beta^{\frac{-\beta}{\beta+1}} L^{\frac{1}{\beta+1}} H^{\frac{1-\beta}{1+\beta}} T^{\frac{\beta}{\beta+1}}. \tag{2.12}$$

Proof. For any time $T > 0$ the expected regret is bounded by from above by,

$$B := L\epsilon^{-\beta}H + \epsilon(\frac{T}{H} - L\epsilon^{-\beta}).$$

We use calculus to optimize $B(\epsilon)$ over choice of ϵ. This suggests a choice of $\epsilon^* = \left(\frac{T}{LH^2\beta}\right)^{\frac{-1}{\beta+1}}$. Evaluating $B(\epsilon^*)$ and ignoring the subdominant terms gives the desired bounds. □

These results demonstrate that the two criterion for statistically efficient reinforcement learning are largely equivalent. However, we should note that the quality of the regret bound which you obtain from PAC bounds is not as strong as the corresponding PAC bound which you obtain from a regret bound. To highlight this difference note that Proposition 2.1 shows that $\mathbb{E}\left[\text{Regret}(T, \pi, M^*)\right] \leq KT^{\frac{1}{2}}$ implies for all $T' > K^2\epsilon^{-2}$, the average regret per timestep is less than ϵ. In a sense this bound is comparable to PAC $O(K^2\epsilon^{-2})$. However, from Proposition 2.2 a bound $\mathbb{E}\left[\sum_{k=1}^{\infty} \mathbb{1}\{\Delta_k > \epsilon\}\right] < K^2\epsilon^{-2}$ can only guarantee a regret bound $O(K^{\frac{2}{3}}T^{\frac{2}{3}})$. Importantly, the dependence upon T is degraded in going from PAC to regret.

Superficially it might seems that although this bound is worse in terms of T, it improves the scaling in K. However, since the regret to time T is always bounded by T these guarantees are a strictly weaker result. To show this we consider any $\alpha > 0$

$$\mathbb{E}\left[\text{Regret}(T, \pi, M^*)\right] \leq KT^{\frac{1}{2}} \implies \mathbb{E}\left[\text{Regret}(T, \pi, M^*)\right] \leq \left(KT^{\frac{1}{2}} \times T^{\alpha}\right)^{\frac{1}{\alpha+1}} = K^{\frac{1}{\alpha+1}} T^{\frac{2\alpha+1}{2\alpha+2}}, \tag{2.13}$$

where the choice of $\alpha = \frac{1}{2}$ recovers the scalings $O(K^{\frac{2}{3}}T^{\frac{2}{3}})$. To simplify our treatment in this dissertation we will focus upon the criterion of regret, rather than PAC-MDP results. It should be clear from these results however, that any bound on regret implies strong guarantees upon the PAC sample complexity of the learning algorithm.

On pure exploration problems

The problem formulation in Section 2.1 seeks to optimize the cumulative rewards through time. In particular, this formulation seeks to guarantee good performance *during* learning. However, in some settings, the learning problem can be divided into separate exploration and exploitation phases. In these settings the agent may wish to minimize the expected loss of the best policy at the end of episode k,

$$\mathbb{E}\left[\Delta_k\right] = \mathbb{E}\left[V^{M^*}_{\mu^*,1} - V^{M^*}_{\mu_k,1}\right]. \tag{2.14}$$

This is known as a pure exploration problem, since the agent only cares about the regret of the best policy available after k episodes and not the cumulative regret over the first k episodes. This criterion is often referred to as "simple regret" or "terminal regret" after k episodes. Proposition 2.3 indicates that these criteria linked to cumulative regret in a natural way.

Proposition 2.3 (Bounds on the expected regret imply bounds on the simple regret)**.** *If actions are selected according the learning algorithm π then,*

$$\mathbb{E}\left[\Delta_k\right] \leq \frac{1}{\lfloor T/H \rfloor} \mathbb{E}\left[\text{Regret}(T, \pi, M^*)\right]. \tag{2.15}$$

Proof. This result is established in Proposition 1 of (Russo, 2015). □

2.2.3 Computational efficiency

We want our algorithms to learn to make good decisions as quickly as possible. So far we have assessed this statistical efficiency in terms of the number of interactions with the environment until learning a near optimal policy. However, we also care about the *computational* efficiency of our learning algorithm. This computational efficiency should be assessed in many ways including the number of floating point operations, memory requirements and practical implementations on available hardware.

For tabular reinforcement learning, we typically describe any algorithm with computational requirements to find an ϵ-optimal policy bounded by any $\text{PolyLog}(S, A, H, \epsilon)$ as "efficient". However, it is clear that in large MDPs even these polynomial bounds might be impractical. In Section 2.3 we will present lower bounds which establish that, without prior structural knowledge, computing an optimal policy for large MDPs may require a similarly large amount amount of computation regardless of algorithm. Nevertheless, we hope that some of the insights and analysis we derive in this simple tabular case will help to guide our algorithm design in future work.

The desire for multiple optimality guarantees, and in particular statistical and computational efficiency, can lead to conflicting requirements for RL, just as in the supervised learning setting (Duchi, 2014). This dissertation presents an approach to reinforcement learning which aims to be both statistically and computationally efficient. In Section 2.4 we will see that it is relatively easy to design algorithms which are statistically efficient given unbounded computation or computationally efficient with poor statistical guarantees. The key contribution of this dissertation is designing algorithms which synthesize statistically efficient approaches to reinforcement learning in a computationally tractable manner.

2.3 Fundamental limits to learning

Reinforcement learning is a very general framework for learning to make good decisions in an unknown environment. The prospect of an algorithm which can address the goals of Section 2.2 could be immensely impactful. However, before we jump to looking for positive results, we first quickly review the sort of bounds which are *not* possible. In this section, we will address the positive present several lower bounds that establish the limits for what could

be possible for *any* algorithm. These lower bounds are important for our understanding of the limitations of RL and give us a measure to assess when a given algorithms performance can be "near-optimal". Our first results establish bounds on the statistical efficiency for any algorithm, regardless of computational cost.

Theorem 2.1 (Lower bounds on regret)**.**
For any learning algorithm π, there exists an MDP M^* as per Section 2.1 such that the regret to time T is bounded,

$$\text{Regret}(T, \pi, M^*) \geq \frac{1}{20}\sqrt{HSAT}. \tag{2.16}$$

Proof. This result is established for $H = 1$ in (Bubeck and Cesa-Bianchi, 2012), the results for $H \geq 2$ can be obtained by a simple extension to absorbing states in the remainder of each episode. Previous works have claimed lower bounds in the infinite horizon setting which, if they were correct, would imply lower bounds $\Omega(H\sqrt{SAT})$ (Bartlett and Tewari, 2009). However, the analysis in this paper is not rigorous and a more careful examination suggests this results is not correct (Osband and Van Roy, 2016b). □

We can establish similar bounds on the computational cost of solving for the optimal policy in a fixed and known MDP M^*. In particular, we note that in a tabular representation where each state and action is represented independently the computational requirements grow with the number of states and actions in the underlying MDP.

Theorem 2.2 (Computational cost of solving a known MDP)**.**
Given a known MDP $M^* = (S, A, R^*, P^*, H)$, solving for the optimal policy μ^* requires at least S^2A floating point operations.

Proof. In order to solve for the optimal policy for an arbitrary MDP M^*, any algorithm must first be able to load in the transition dynamics P^* at each state and action. Simply iterating through the transition dynamics $P^*(s, a) \in [0, 1]^S$ for each (s, a) requires S^2A operations, which completes the proof. □

This dissertation aims to design algorithms for reinforcement learning which are simultaneously statistically efficient and computationally tractable. It is clear that learning an unknown MDP M^* is at least as hard as solving a single known MDP. We have several

efficiency results for learning with unlimited computation and computation given full knowledge. However, for the setting of simultaneously statistically and computationally efficient learning there are relatively few known lower bounds. This represents an important area of future research.

In a sense the result of 2.2 is quite trivial; all it says is that to be able to solve a known MDP requires at least as much computation as it does to specify the transition dynamics. However this should be a reminder that for systems with S and A extremely large the idea of "solving" for the optimal policy of a single known MDP is itself not a practical consideration - even when the transition dynamics are completely known. To highlight this problem, consider the ancient game of Go, played on a 19x19 board whose dynamics are simple, deterministic and fully known (Boorman, 1969). Nonetheless the number of possible board states is at least $S \geq 10^{170}$, far greater than the number of atoms in the universe, which is typically estimated at around 10^{80}. Nonetheless, recent breakthroughs in machine learning have even made progress on the game of Go, but doing this required some significant *generalization* between board states (Silver et al., 2016). In the later part of this disseration we will look at how we can extend the insights and intuition from tabular learning to learning with generalization; for now, we simply advise the reader to remember than learning from *tabula rasa* might not be a practical approach to realistic large-scale problems.

2.4 Inefficient algorithms

We will now review some of the most common approaches to reinforcement learning and highlight why these methods are statistically or computationally inefficient. To motivate this discussion we will begin by describing a family of environments that highlight the need for deep exploration in RL. At a high level they all contain a single optimal policy which is hard to find via random actions, but can be discovered efficiently through deep exploration. The spirit of these environments is similar to *RiverSwim* (Strehl and Littman, 2005). Although these problems are heavily stylized, they capture the essential problems of learning to act in an environment where the optimal policy is difficult to find by naive gradient descent methods. We present an illustration of this environment in Figure 2.1.

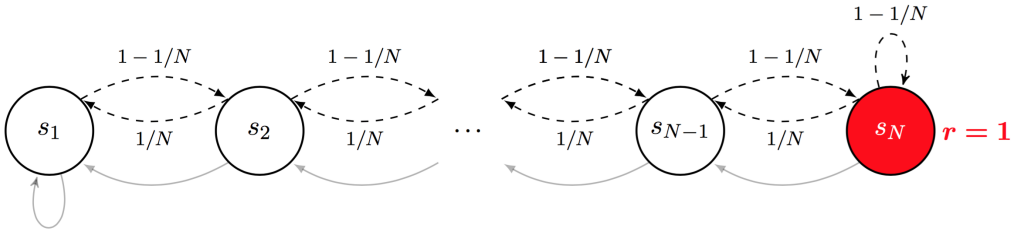


Figure 2.1: An MDP where algorithms without deep exploration are highly inefficient.

This environment is made up of a long chain of states $\mathcal{S} = \{1, .., N\}$. Each step the agent can take the action "left" or "right". The action "left" is deterministic, but the action "right" only succeed with probability $1 - 1/N$, otherwise they go left. All states have zero reward except for the far right N which gives a reward of 1. Each episode is of length $H = N - 1$ and the agent will begin each episode at state 1. The optimal policy is to go right at every step to receive an expected reward of $p^* = (1 - \frac{1}{N})^{N-1}$ each episode, all other policies give no reward.

In this section we will consider several families of algorithms for RL at a high level, rather than delve into the specific implementation for any specific algorithm. To maintain this generality we will consider an arbitrary estimator for MDPs, which is nothing more than a function from past data to an estimate of the underlying MDP. For example, we might simply take the empirical mean of observed transitions or rewards for each state, action and timestep and use these values as estimates, although a proper implementation of this would require special rules for states and actions which had not yet been observed. Alternatively we could use a Bayesian procedure to maintain a posterior distribution over beliefs for each reward and transition, this can be implemented trivially using conjugate prior families. For more details on tractable Bayesian models for tabular MDPs see (Strens, 2000).

2.4.1 Greedy control

The most common approach to reinforcement learning treats the estimation and control portions separately. This algorithm approximates the true underlying MDP M^*, which is unknown, by its best point estimate $\hat{M}_k$ and follows the policy which is optimal for that estimate. This purely exploitative policy is called greedy, since it does not account for the potential future value of exploratory actions.

Algorithm 2.1 Greedy reinforcement learning

Input: Point estimator for MDPs ϕ.
for episode $k = 1, 2, ..$ **do**
  Estimate MDP $\hat{M}_k = \phi(\mathcal{H}_{k1})$
  compute $\mu_k \in \arg\max_\mu V^{\hat{M}_k}_{\mu,1}$
  for time $h = 1, 2, .., H$ **do**
    take action $a_{kh} = \mu_k(s_{kh}, h)$
    observe r_{kh} and s_{kh+1}
    update $\mathcal{H}_{kh} = \mathcal{H}_{kh} \cup (a_{kh}, r_{kh}, s_{kh+1})$
  end for
end for

The greedy approach to RL is computationally efficient, since it requires nothing more than solving a single known MDP. For many settings, this algorithm works reasonably well and actually some of the early successes in reinforcement learning used precisely than this approach (Samuel, 1959; Sutton, 1984; Tesauro, 1995). Although greedy algorithms can be effective in many settings, they may be extremely statistically inefficient. A purely exploitative greedy agent may never learn the optimal policy, even with exact inference and an infinite amount of data (Thompson, 1933).

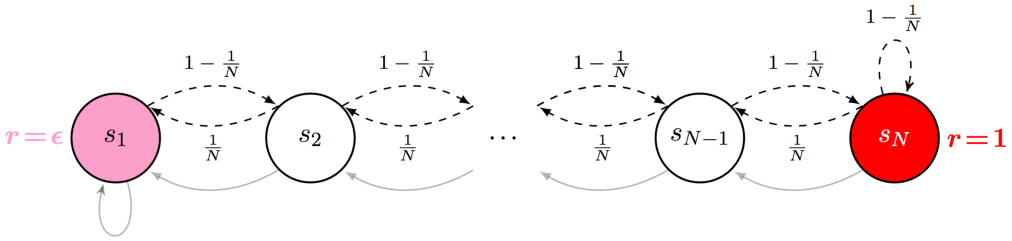


Figure 2.2: An MDP where greedy algorithms may never learn the optimal policy.

To see how this can happen, imagine if the example in Figure 2.1 were altered so that action "left" in state 1 produced a tiny reward of $0 < \epsilon \ll \frac{1}{N}$; we depict such an environment in Figure 2.2. A greedy agent who begins in state s_1 might discover this rewarding policy, which produces the highest expected value given the data it has already seen. However, once it employs this policy to head "left" in any episode it will *never* be able to explore the chain far enough to find the optimal policy.

2.4.2 Dithering strategies for exploration

Purely exploitative strategies for reinforcement learning can fail entirely to learn an optimal policy. This is because they do not account for the value of information through exploratory actions. One approach to counter premature and suboptimal exploitation, is to employ a dithering strategy for action selection. Dithering strategies purposefully inject some random noise to their choice of action so that they are no longer purely greedy. This means that the agent will still gather some information about actions which are not estimated to be the "best" action, so that eventually they should be able to learn the true optimal policy. However, dithering schemes do not direct their random exploration towards potentially informative states and actions, which can lead to exponentially slower learning.

The two most common dithering strategies for exploration are called ϵ-greedy and Boltzmann exploration. Both of these algorithms are identical to greedy reinforcement learning algorithm 2.1 with the exception of step (7.) and their choice of action. For exploration by ϵ-greedy is exactly the same as greedy but at each timestep with probability ϵ the agent selects an action uniformly at random from the action set $\mathcal{A}$. Boltzmann exploration selects actions at random proportional to the estimated Q-values of each action. The scaling of this sampling distribution is typically controlled by a temperature parameter $\eta > 0$. We present these approaches in more detail in Algorithms 2.2 and 2.3.

Dithering strategies for exploration are very popular in many practical applications (Sutton and Barto, 1998; Mnih et al., 2015). They are conceptually simple, computationally efficient and very easy to implement. Compared to purely greedy strategies they are typically more robust, since the inclusion of selectively random actions induces more exploration. Asymptotically, dithering strategies are guaranteed to explore every possible state and action. Combined with a specialized annealing exploration strategy (such that $\epsilon, \eta \to 0$ at an appropriate rate) these dithering strategies may even be be asymptotically no-regret (Watkins and Dayan, 1992). However, dithering strategies can be remarkably statistically inefficient and make take an exponentially long time to learn (Lai and Robbins, 1985; Russo and Van Roy, 2014; Brafman and Tennenholtz, 2002). Example 2.1 highlights a simple example where this can be the case.

Algorithm 2.2 RL with ϵ-greedy exploration

Input: Point estimator for MDPs ϕ, $\epsilon \in [0, 1]$
for episode $k = 1, 2, ..$ **do**
  Estimate MDP $\hat{M}_k = \phi(\mathcal{H}_{k1})$
  compute $\mu_k \in \arg\max_\mu V^{\hat{M}_k}_{\mu,1}$
  for time $h = 1, 2, .., H$ **do**
    sample $w_{kh} \sim \text{Unif}([0, 1])$
    if $w_{kh} \leq \epsilon$ **then**
      take action $a_{kh} \sim \text{Unif}(\mathcal{A})$
    else
      take action $a_{kh} = \mu_k(s_{kh}, h)$
    end if
    observe r_{kh} and s_{kh+1}
    update $\mathcal{H}_{kh} = \mathcal{H}_{kh} \cup (a_{kh}, r_{kh}, s_{kh+1})$
  end for
end for

Algorithm 2.3 RL with Boltzmann exploration

Input: Point estimator for MDPs ϕ, $\eta > 0$
for episode $k = 1, 2, ..$ **do**
  Estimate MDP $\hat{M}_k = \phi(\mathcal{H}_{k1})$
  compute $\mu_k \in \arg\max_\mu V^{\hat{M}_k}_{\mu,1}$
  for time $h = 1, 2, .., H$ **do**
    sample $\tilde{a} \propto \exp\left(Q^{\hat{M}_k}_{\mu_k,h}(s_{kh}, a)/\eta\right)_{a\in\mathcal{A}}$
    take action $a_{kh} = \tilde{a}$
    observe r_{kh} and s_{kh+1}
    update $\mathcal{H}_{kh} = \mathcal{H}_{kh} \cup (a_{kh}, r_{kh}, s_{kh+1})$
  end for
end for

Example 2.1 (Dithering strategies may take exponentially long to learn)**.**
Let π be a reinforcement learning algorithm which uses a dithering scheme for action selection. Then there exists a distribution over MDPs $M^* = (\mathcal{S}, \mathcal{A}, R^*, P^*, H, \rho)$ such that required the number of steps T until the average regret is less than $\epsilon = 0.1$ is at least $T^{\min} = 2^{S-1} - 1$.

Proof. We consider a distribution over MDPs in the style of the example MDP from Figure 2.1, where the label of each $s_1, .., s_N$ is permuted. Let k^* be the first episode in which the

state N is visited and a reward is received. For all $t < k^*H$ there has been zero observed reward, so under any dithering strategy all actions are sampled uniformly at random. Thus in any episode the rewarding state will be reached with probability less $(1 - \frac{1}{N})^{N-1}2^{-(N-1)}$. It follows that $\mathbb{E}[k^*] > 2^{S-1} - 1$. □

2.4.3 Exploration in policy space

Dithering strategies for exploration are inefficient for several reasons. One of their failings is that they do not engage in temporally extended or consistent exploration. In an environment with delayed consequences this may be problematic. Dithering exploration will be exponentially unlikely to sample an extended exploratory policy over several timesteps.

A natural approach to remedy this shortcoming is through exploration in the policy rather than the action space (Wierstra et al., 2008; Sehnke et al., 2010; Maei et al., 2010). An algorithm using this approach would build up an estimate of the optimal *policy* in the environment, parameterized by some $\mu = \mu_\theta$. In order to perform exploration, the agent then perturbs $\tilde{\theta} = \theta + w$ and then follows the policy $\mu_{\tilde{\theta}}$ for that episode. Typical approaches in this family to policy exploration include policy gradient, evolutionary strategies and random noise. This approach can have several benefits over dithering. First, this allows for exploration which is temporally extended and consistent over multiple timesteps. Second, in some domains the space of policies may be much simpler than the space of possible MDPs. However, as Example 2.2 shows, exploration in policy space is not enough to guarantee efficient reinforcement learning.

Example 2.2 (Policy exploration may take exponentially long to learn)**.**
Let π be a reinforcement learning algorithm which explores purely based upon policy search. Then there exists a distribution over MDPs $M^* = (\mathcal{S}, \mathcal{A}, R^*, P^*, H, \rho)$ such that required the number of steps T until the average regret is less than $\epsilon = 0.1$ is at least $T^{\min} = 2^{S-1} - 1$.

Proof. We consider a distribution over MDPs in the style of the example MDP from Figure 2.1, where the label of each $s_1, .., s_N$ is permuted. Let k^* be the first episode in which the state N is visited and a reward is received. For all $t < k^*H$ there has been zero observed reward, so no pure policy search algorithm can tell the difference between any sampled policies. The total number of possible policies to explore is 2^{SH}, therefore $\mathbb{E}[k^*] = \Omega(2^{SH})$ in the worst case. □

2.4.4 Tree-based sampling

Policy search allows for temporally extended exploration, however this exploration may not be directed at potentially informative states and actions. Efficient reinforcement learning requires deep exploration, which is both temporally extended and directed towards potentially informative states and actions. We will now consider a family of algorithms which are explicitly designed for deep exploration.

In tree-based sampling algorithms, the agent explicitly builds a lookahead tree to imagine possible future states and actions. The agent then simulates these future outcomes, either with a access to a generative model of the environment (Kearns et al., 2002; Kocsis and Szepesvári, 2006), or some Bayesian posterior belief (Wang et al., 2005; Guez et al., 2012). They then imagine future trajectories under temporally extended sequences of actions and take the action with the largest expected long term return.

Given enough computation, tree-based sampling methods can be extremely statistically efficient, since they explicitly balance the value of exploration vs exploitation in a principled manner. In a sense, these models come closest to the "gold-standard" of principled reinforcement learning in the Bayesian model. However, these methods and the Bayes-optimal framework for reinforcement learning all share several shortcomings.

First, these algorithms make heavy use of a generative model (or prior description) of the environment. One problem is that in many settings it may be difficult to accurately encode a prior distribution over these beliefs; doing this correctly would lead to extremely complex models. As a result, many practical applications typically encode rough approximations. However, combining highly optimized lookahead planning with an inaccurate model will typically lead to poor performance as the planning tree exploits shortcomings in the model and can be extremely fragile to model misspecification (Guez et al., 2013). Second, these algorithms are extremely computationally expensive. In general, tree-based lookahead methods require $\Omega(A^H)$ calls to the generative model (Kakade, 2003). Approximations which use truncated computational approximations can fail spectacularly badly and may require super-exponential amounts of time to learn (Munos, 2014).

2.4.5 Towards efficient reinforcement learning

In this section we have outlined several broad algorithmic approaches to reinforcement learning. Further, we have even outlined several computationally tractable approaches which are asymptotically no regret. However, these computationally tractable heuristics can be extremely statistically inefficient since they do not employ deep exploration. At the other end of the spectrum, we have introduced several tree-based approaches to deep exploration which can be statistically efficient; the problem is that these methods are not computationally tractable. In the next chapter we will introduce a simple and tractable approach to deep exploration, "optimism in the face of uncertainty". We will see that this principle can allow for algorithms which are both statistically efficient and computationally tractable.

Chapter 3

Optimism in the Face of Uncertainty

We have seen that naive approaches to reinforcement learning without deep exploration can be extremely statistically inefficient. At the same time, algorithms which explicitly compute the future value of exploratory actions can be computationally intractable. In this chapter we outline the principle of *optimism in the face of uncertainty* (OFU) which has driven the majority of progress in the field of efficient reinforcement learning. This heuristic approximates the benefits of exploration by assigning an optimistic bonus to poorly understood states and actions. Actions are then selected with respect to this optimistic model of the world, which incentivizes exploration. As the agent resolves its uncertainty, the effect of optimism is reduced and the agent's behavior approaches optimality.

We begin this chapter with an investigation of optimism in reinforcement learning. In particular, we will focus upon variants of the *upper confidence for reinforcement learning* (UCRL) which have established the existing state of the art for provably efficient RL (Jaksch et al., 2010). We will outline the main arguments and tools in their analysis, which will form an essential core for our own subsequent algorithms. Following this analysis, we will highlight several manners in which UCRL and other existing optimistic algorithms are inefficient. These shortcomings will lead us to consider a new approach to optimism, based upon entire distributions rather than confidence sets. We define this notion of *stochastic optimism* and place this definition in the context of several similar existing concepts and explain why

this concept is important for optimism. Finally, we present a technical lemma which relates Gaussian and Dirichlet posteriors in terms of stochastic optimism. The proof of this lemma is long and somewhat uninspiring, however the result will be of great importance for our design of provably efficient algorithms based upon stochastic optimism. In particular, this result is the key to our improved analysis of the algorithm *posterior sampling for reinforcement learning* (PSRL) and provides motivation for several other algorithmic approaches within this dissertation.

3.1 Efficient reinforcement learning via optimism

Efficient reinforcement learning requires efficiently balancing the conflicting goals of exploration and exploitation. This is difficult since the algorithm must simultaneously lean and control an unknown system. The optimistic principle is simple:

> "Select the policy which would obtain the best possible rewards in the best plausible environment."

We present one such high level approach for model-based RL in Algorithm 3.1.

Algorithm 3.1 Optimistic model-based RL

1: **Input:** Confidence set generator for MDPs Φ
2: **for** episode $k = 1, 2, ..$ **do**
3: form confidence set $\mathcal{M}_k \sim \Phi(\cdot|\mathcal{H}_{k1})$
4: compute $\mu_k \in \arg\max_{\mu, M \in \mathcal{M}_k} V^M_{\mu,1}$
5: **for** time $h = 1, 2, .., H$ **do**
6: take action $a_{kh} = \mu_k(s_{kh}, h)$
7: observe r_{kh} and s_{kh+1}
8: update $\mathcal{H}_{kh} = \mathcal{H}_{kh} \cup (a_{kh}, r_{kh}, s_{kh+1})$
9: **end for**
10: **end for**

The key steps in such an optimistic algorithm are the generation of confidence sets (3.) and optimistic planning (4.). The procedure Φ should generate confidence sets that contain the true MDP M^* with high probability, but should also concentrate as fast as possible given the data; the quality of this confidence set is the key to statistical efficiency. At the same time, the algorithm must be able to plan optimistically over all policies μ and all

MDPs $M \in \mathcal{M}_k$; this is the key step for computational efficiency. To be more concrete, one such procedure Φ is given by the UCRL confidence sets in Algorithm 3.2 and optimistic value iteration presents a natural approach to optimistic planning (Strehl and Littman, 2005; Jaksch et al., 2010; Bartlett and Tewari, 2009).

Optimism in the face of uncertainty acts as a tractable heuristic to balance exploration versus exploitation. These algorithms afford some optimistic bonus to poorly-understood states and actions in order to approximate the value of exploratory information. With careful analytical design, this optimism can be tuned to give near-optimal performance guarantees (Lai and Robbins, 1985; Kolter and Ng, 2009; Araya et al., 2012; Brafman and Tennenholtz, 2002; Jaksch et al., 2010). We will now outline the general flavor for proving regret bounds in optimistic algorithms, together with a sketch of the state of the art bounds in UCRL.

3.1.1 Regret bounds for optimistic algorithms

The key difficulty in bounding the regret of any learning algorithm is the dependence upon the optimal value of the optimal policy $V^{M_*}_{\mu^{M_*},1}$. It is absolutely central to the RL problem that the agent does not know this value a priori, if they did they would just be able to employ the optimal policy and we would not need any learning. However, from the point of view of analysis, it is difficult to prove efficient convergence to a quantity which you don't even know! The optimistic principle allows us to sidestep the issue of bounding the regret from the optimal policy, by instead bounding the regret from the *imagined* optimistic optimal policy.

To simplify our notation, we will denote by M_k an optimistic MDP, which attains the maximum value of $V^M_{\mu,1}$ over all $M \in \mathcal{M}_k$ and all policies μ. For shorthand we will write $V^*_{*,h}$ for $V^{M_*}_{\mu^*,h}$, $V^k_{k,h} for V^{M_k}_{\mu_k,1}$ and similarly the combinations $V^*_{k,1} = V^{M_*}_{\mu_k,1}$. Using this notation we can decompose the per episode regret into two components, the regret from optimism

Δ_k^{opt} and the regret from concentration Δ_k^{conc}.

$$\begin{aligned} \Delta_k &= V_{\mu^{M_*},1}^{M_*} - V_{\mu_k,1}^{M_*} \\ &= V_{*,1}^{*} - V_{k,1}^{*} + V_{k,1}^{k} - V_{k,1}^{k} \\ &= \underbrace{V_{k,1}^{k} - V_{k,1}^{*}}_{\Delta_k^{\text{conc}}} + \underbrace{V_{*,1}^{*} - V_{k,1}^{k}}_{\Delta_k^{\text{opt}}}. \end{aligned} \tag{3.1}$$

The regret from optimism Δ_k^{opt} is the difference between the optimal value of the true MDP M^* and optimal value of the imagined optimistic MDP M_k. In a well designed optimistic algorithm, the true MDP M^* lies within the confidence set $\mathcal{M}_k$ with high probability. By construction, the imagined optimal value $V_{k,1}^k = \max_{M,\mu} V_{\mu,1}^M \geq V_{*,1}^*$ whenever $M^* \in \mathcal{M}_k$. Therefore, with high probability the regret from optimism is less than zero $\Delta_k^{\text{opt}} \leq 0$.

The regret from concentration Δ_k^{conc} is the difference between the imagined value of the policy μ_k under the optimistic MDP M_k and actual value of the policy μ_k in the true MDP M^*. Importantly, and unlike the regret Δ_k, this term does not depend upon the unknown optimal policy μ^* but instead only upon the policy μ_k which the agent actually follows. As the agent gathers more data in its environment (by following μ_k) its imagined value $V_{k,1}^k$ should concentrate around the true value $V_{k,1}^*$ at a rate which can be recovered from standard supervised learning results. At a high level, this describes the proof strategy for a large literature on optimistic algorithms in reinforcement learning:

- Design confidence sets $\mathcal{M}_k$ which contain M^* with high probability.
- Specify optimistic variant of Algorithm 3.1 using $\mathcal{M}_k$.
- Bound $\text{Regret}(T, \pi, M^*) \leq \sum_{k=1}^{\lceil T/H \rceil} \Delta_k^{\text{conc}} \leq f(T, S, A, H)$ with high probability.

3.1.2 Near-optimal regret bounds with UCRL

We now reproduce a simple optimistic algorithm "upper confidence reinforcement learning" (UCRL) that attains state of the art (and near optimal) regret bounds for the problem formulation of Section 2.1. This algorithm is the natural adaptation of UCRL2 to finite episode reinforcement learning. We present only the key arguments of the proof with a focus upon the intuition, for more detail readers should see the journal paper (Jaksch et al., 2010).

Designing confidence sets for MDPs

The key element of any optimistic algorithm is the design of the confidence sets $\mathcal{M}_k$ so that they contain the true MDP $M^* \in \mathcal{M}_k$ with high probability. In order to do this, UCRL generates confidence sets for each reward $R(s,a)$ and transition $P(s,a,)$ independently and takes the confidence set $\mathcal{M}_k$ to be the set of all MDPs whose rewards and transitions lie within these sets. These confidence sets will be guided by concentration which guarantee that, in some sense, the empirical mean will be a reasonable approximation to the unknown mean. Most importantly, these results bound the degree of mis-estimation with high probability even for finite sample estimates.

Lemma 3.1 (L1 bounds for the empirical distribution over finite sets)**.**
Let $y_1, .., y_n$ be IID samples from a distribution P^* over a finite set $\mathcal{S} = \{1, .., S\}$. Define the empirical distribution $\hat{P} := \frac{1}{n}\sum_{i=1}^{n} \delta_{y_i}(s)$, where δ_s is the probability mass function over $\mathcal{S}$ that assigns all mass to s. Then, for any $\epsilon > 0$ the deviation of the true distribution P^* from the empirical estimate $\hat{P}$ is bounded in probability:

$$\mathbb{P}\left(\|P^* - \hat{P}\|_1 \geq \epsilon\right) \leq \exp\left(S\log(2) - \frac{n\epsilon^2}{2}\right). \tag{3.2}$$

Further, for any $\delta > 0$

$$\mathbb{P}\left(\|P^* - \hat{P}\|_1 \geq \sqrt{\frac{2S\log(2/\delta)}{n}}\right) \leq \delta. \tag{3.3}$$

Proof. Equation (3.2) is a special case of the result established by Weissman (Weissman et al., 2003). Equation (3.3) can be obtained through algebraic manipulation of (3.2). □

We state our next result in terms of sub-Gaussian random variables (Buldygin and Kozachenko, 1980), which is a technical property for random variables which regulates the tail behavior. A random variable X is σ-sub-Gaussian if and only if, for all $t > 0$ $\mathbb{E}[\exp(tX)] \leq \exp\left(\frac{t\sigma^2}{2}\right)$. We will say that X is sub-Gaussian if X is 1 sub-Gaussian. Intuitively a random variable is is σ-sub-Gaussian if its tails are dominated by a Gaussian random variable with standard deviation of σ. For our purposes it will be enough to note that the reward noise is sub-Gaussian, since it is zero-mean and bounded $\in [-1, 1]$.

Lemma 3.2 (Tail bounds for sub-Gaussian random variables)**.**
Let $x_1, .., x_n$ be independent samples from sub-Gaussian random variables. For any $\epsilon > 0$:

$$\mathbb{P}\left(\frac{1}{n}|\sum_{i=1}^{n} x_i| \geq \epsilon\right) \leq \exp\left(\log(2) - \frac{n\epsilon^2}{2}\right). \tag{3.4}$$

Further, for any $\delta > 0$

$$\mathbb{P}\left(\frac{1}{n}|\sum_{i=1}^{n} x_i| \geq \sqrt{\frac{2\log(2/\delta)}{n}}\right) \leq \delta. \tag{3.5}$$

Proof. Equation (3.4) is a standard result for sub-Gaussian random variables (Buldygin and Kozachenko, 1980). Equation (3.5) can be obtained through algebraic manipulation of (3.4). □

Lemmas 3.1 and 3.2 allow us to guarantee with high probability that the true unknown transition and reward lie close to their empirical estimates. Importantly, at episode k we can quantify these bounds in terms of the number of times each state and action have been sampled prior to episode k:

$$n_k(s,a) := \sum_{j=1}^{k-1}\sum_{h=1}^{H} \mathbb{1}\{(s_{jh}, a_{jh}) = (s,a)\}. \tag{3.6}$$

To guarantee that each reward and transition lies within the confidence set at each timestep for each state and action, we use a simple union bound over all states, actions, timesteps and episodes. This just means that we rescale $\delta \leftarrow \tilde{\delta} := \frac{\delta}{SAHk^2}$ in episode k, to guarantee that the true MDP lies within confidence at each and every timestep. This step in the analysis may be extremely loose, as we will examine in Section 3.2, but since the confidence sets depend only logarithmically in δ this step will not make any difference to the $\tilde{O}(\cdot)$ scaling of the proof. This argument allows us to produce efficient confidence sets in UCRL, which we present in Algorithm 3.2. We write $\hat{r}_k(s,a,h)$ for the empirical mean reward from (s,a,h) observed up to episode k.

Algorithm 3.2 Confidence sets for UCRL

Input: History of past observation $\mathcal{H}_{k1}$.
for each $s \in \mathcal{S}, a \in \mathcal{A}$ **do**
  compute empirical estimates $\hat{P}_k(s,a), \hat{r}_k(s,a)$ and counts $n_k(s,a)$ from $\mathcal{H}_{k1}$
  if $n_k(s,a) = 0$ **then**
    allow any transition $\mathcal{P}_k(s,a) \leftarrow \{P \in [0,1]^S \mid P^T \mathbb{1} = 1\}$
    allow any reward $\mathcal{R}_k(s,a) \leftarrow [0,1]$
  end if
  transition set $\mathcal{P}_k(s,a) \leftarrow \left\{P \in [0,1]^S \mid P^T\mathbb{1}=1,\ \|\hat{P}_k(s,a) - P\|_1 \leq \sqrt{\frac{4S\log(2SAHk/\delta)}{n_k(s,a)}}\right\}$
  reward set $\mathcal{R}_k(s,a) \leftarrow \left\{r \in [0,1] \mid \|\hat{r}_k(s,a,h) - r\|_1 \leq \sqrt{\frac{4\log(2SAHk/\delta)}{n_k(s,a)}}\right\}$
end for
Return: $\mathcal{M}_k = \{M \mid R(s,a) \in \mathcal{R}_k(s,a),\ P(s,a) \in \mathcal{P}_k(s,a)\ \forall (s,a,h)\}$

Algorithm 3.2 is designed with reference to the concentration Lemmas 3.1 and 3.2. This allows us to provide an analytical guarantee that, regardless of the exploratory policy of the algorithm, the true MDP M^* will lie within the confidence sets $\mathcal{M}_k$ for all episodes k.

Proposition 3.1 (Confidence sets hold with high probability)**.**
Let M^* be the true unknown MDP from Section 2.1 then, for all exploratory policies $\mu_1, \mu_2, ..$ the sequence of confidence sets $\mathcal{M}_1, \mathcal{M}_2$ defined by Algorithm 3.2 will contain the true MDP with high probability. In particular, for any $\delta > 0$

$$\mathbb{P}\left(\bigcup_{k=1}^{\infty} \{M^* \notin \mathcal{M}_k\}\right) \leq \frac{\pi^2 \delta}{6}. \tag{3.7}$$

Proof. Combine Lemmas 3.1 and 3.2 together with a naive union bound over all states, actions, timesteps and episodes. See (Jaksch et al., 2010) for more details. □

Optimizing optimistically over MDPs

A result like Proposition 3.1 is essential to any optimistic algorithm. If we can plan optimistically over these confidence sets then we will have no positive regret from optimism with high probability. For optimistic planning over $\mathcal{M}_k$ it is obvious that the planner should just use $r_k(s,a,h) \in \max \mathcal{R}_k(s,a,h)$ for each (s,a,h). The question of how to assign optimistic transitions $P_k(s,a,h) \in \mathcal{P}_k(s,a,h)$ is more subtle, but can be accomplished by an algorithm called extended value iteration (Strehl and Littman, 2005; Jaksch et al., 2010).

This algorithm increases the computational cost by a factor of S versus solving a single MDP (Jaksch et al., 2010)

Now that we have established a tractable method to generate optimistic confidence sets and an algorithm to optimize the policy optimistically over these confidence sets we have essentially defined the UCRL algorithm. The remainder of the proof is now simply to show that, as more data is collected, these confidence sets converge around the true value estimates. Intuitively this should be clear, since the width of the estimates at each (s, a, h) contract with every visit $\tilde{O}(\frac{1}{\sqrt{n}})$. The rest of the proof is nothing more than showing this leads to concentration in the value estimate Δ_k^{conc}.

Concentration in the value estimate

We now turn our attention to bounding the regret from concentration $\Delta_k^{\text{conc}} = V_{k,1}^k - V_{k,1}^*$. This represents the estimation error in the value of policy μ_k under the imagined optimistic MDP M_k and the true underlying MDP M^*. Importantly, this quantity only depends upon the policy μ_k which the agent actually follows. We will show that, as the agent gathers more information about the policy through experience, this value estimate will concentrate around the truth. A central tool to our analysis will be the Bellman operator $\mathcal{T}_{\mu,h}^M$, which returns the expected value of state s where we follow the policy μ under the laws of M for one timestep from h.

Definition 3.1 (Bellman operator)**.**
For any MDP $M = (\mathcal{S}, \mathcal{A}, R, P, H, \rho)$ and policy μ we define the Bellman operator to act upon value functions $V : \mathcal{S} \to \mathbb{R}$ by

$$\mathcal{T}_{\mu,h}^M V(s) := \overline{r}(s, \mu(s,h)) + \sum_{s' \in \mathcal{S}} P(s'|s, \mu(s,h)) V(s'). \tag{3.8}$$

We will streamline our discussion of $R, P, V, \mathcal{T}$ by simply writing $*$ in place of M^* or μ^* and k in place of M_k or μ_k where appropriate. We will also condense the state-action-timestep variable $x_{kh} := (s_{kh}, \mu_k(s_{kh}, h))$ under the policy μ_k. We can then decompose the regret from concentration by repeated applications of dynamic programming. Since we are only interested in finite MDPs it suffices to consider the deterministic initial distribution ρ which always begins from s_0. We can decompose the concentration error into the error in

estimation of $\overline{r}^*, P^*$

$$\begin{aligned}
\Delta_k^{\text{conc}} &= V_{k,1}^k(s_{k1}) - V_{k,1}^*(s_{k1}) \\
&= \mathcal{T}_{k,1}^k V_{k,2}^k(s_{k1}) - \mathcal{T}_{k,1}^* V_{k,2}^*(s_{k1}) \\
&= (\overline{r}_k - \overline{r}^*)(x_{k1}) + \left(P_k(x_{k1})^T V_{k,2}^k - P^*(x_{k1})^T V_{k,2}^*\right) \\
&= (\overline{r}_k - \overline{r}^*)(x_{k1}) + ((P_k - P^*)(x_{k1}))^T V_{k,2}^k + P^*(x_{k1})^T (V_{k,2}^k - V_{k,2}^*). \qquad (3.9)
\end{aligned}$$

Now we note that $P^*(x_{k1})^T(V_{k,2}^k - V_{k,2}^*) = \mathbb{E}\left[(V_{k,2}^k - V_{k,2}^*)(s') \mid s' \sim P^*(x_{k1})\right]$. This means that we write $P^*(x_{k1})^T(V_{k,2}^k - V_{k,2}^*) = (V_{k,2}^k - V_{k,2}^*)(s_{k2}) + d_{k1}$, where d_{k1} is some martingale difference bounded by H. We can repeat the argument of (3.9) to express the concentration error

$$\Delta_k^{\text{conc}} = \sum_{h=1}^{H} \left(\overline{r}_k - \overline{r}^*\right)(x_{kh}) + \sum_{h=1}^{H} \left((P_k - P^*)(x_{kh})\right)^T V_{k,h+1}^k + \sum_{h=1}^{H} d_{kh}. \qquad (3.10)$$

The term from the martingale difference d_{kh} can be controlled by standard concentration results such as the Azuma-Hoeffding concentration lemma. The resulting term is ultimately subdominant $\tilde{O}(H\sqrt{T})$ (Jaksch et al., 2010), for this reason we will ignore its contribution in the rest of our analysis to give a shortened proof. The key argument is that since the true MDP M^* lies within the confidence set $\mathcal{M}_k$ with high probability, we can upper bound the estimation error in both $\overline{r}^*$ and P^* by the maximum deviation within $\mathcal{M}_k$ at each (s, a, h) by using a Holder decomposition on the future value $P^T V$. This ensure that we can guarantee with high probability

$$\begin{aligned}
\text{Regret}(T, \pi^{\text{UCRL}}, M^*) &\leq \sum_{k=1}^{\lceil T/H \rceil} \sum_{h=1}^{H} \sup_{\mathcal{R}_k} |r_k^+ - r_k^-|(x_{k1}) + \sup_{\mathcal{P}_k} \|(P_k^+ - P_k^-)(x_{kh})\|_1 \|V_{k,h+1}^k\|_\infty \\
&\leq \sum_{k=1}^{\lceil T/H \rceil} \sum_{h=1}^{H} \left(1 + H\sqrt{S}\right) \sqrt{\frac{4\log(2SAHk/\delta)}{n_k(x_{kh})}}. \qquad (3.11)
\end{aligned}$$

We conclude our argument by upper bounding the expression in equation (3.11) over any possible trajectory of states and actions $\mathcal{H}_{11}, \mathcal{H}_{21}, \ldots$ We use an integral comparison $\sum_{n=1}^{T} \frac{1}{\sqrt{n}} \leq \int_{t=0}^{T} \frac{1}{\sqrt{t}} dt \leq 2\sqrt{T}$ together with the constraint that $\sum_{k=1}^{\lceil T/H \rceil} \sum_{h=1}^{H} n_{kh} = T$. There are some small details to do with the mismatch in terms of visits within an episode, since the quantities n_k are only updated by Algorithm 3.2 at the start of each episode.

These turn out to not be important, but with a little more care we can establish the results of Theorem 3.1. For more details see (Osband et al., 2013; Osband and Van Roy, 2014b,a).

Theorem 3.1 (Near-optimal regret bounds for UCRL)**.**
Let the true environment M^* be any finite MDP as defined in Section 2.1. For any $T > 0$ and any $\delta > 0$ the regret of UCRL is bounded

$$\text{Regret}(T, \pi^{\text{UCRL}}, M^*) = O\left(HS\sqrt{AT\log(T/\delta)}\right), \tag{3.12}$$

with probability at least $1 - \delta$.

Theorem 3.1 is a significant statistical guarantee as its scaling is close to the fundamental lower bound $\Omega(\sqrt{HSAT})$ (Jaksch et al., 2010; Osband and Van Roy, 2016b). At the same time UCRL is also computationally tractable, in the sense that the optimistic planning algorithm extended value iteration requires only a polynomial number of operations in the problem parameters S, A, H. Roughly speaking optimistic value iteration requires a factor S more operations than solving a single known MDP (Jaksch et al., 2010). For more details on these lower bounds see Section 2.3.

3.2 Shortcomings of existing optimistic algorithms

Algorithms in the style of UCRL (Auer et al., 2009; Bartlett and Tewari, 2009; Jaksch et al., 2010; Dann and Brunskill, 2015) represent the culmination of many years of research in optimistic reinforcement learning (Brafman and Tennenholtz, 2002; Kearns and Singh, 2002; Strehl and Littman, 2005; Strehl et al., 2006). The construction of the confidence sets $\mathcal{M}_k$ and the optimistic planning step have been honed to admit theoretical analysis and improved performance. However, there remain some significant shortcomings even in these "near-optimal" algorithms for reinforcement learning. First, it is extremely difficult to design confidence sets for MDPs which are statistically efficient. We will highlight several manners in which the confidence sets generated by this algorithm are loose, this hurts both the quality of the bounds and the empirical performance of the algorithm. Second, although we believe it is possible to design confidence sets that make optimistic algorithms statistically efficient, using such confidence sets appears to call for intractable computations in optimistic planning (Osband and Van Roy, 2016c).

3.2.1 Statistical inefficiency

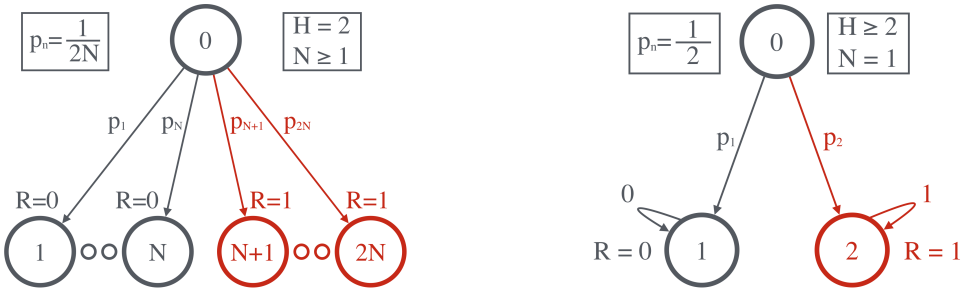
We should be careful to distinguish the statistical efficiency of the *algorithm* with the statistical efficiency of the *analysis*. Certainly, any particular analysis may provide a statistically inefficient upper bound on the regret of an algorithm without implying the algorithm itself is statistically inefficient. However, classical optimistic RL algorithms (in the style of UCRL) tightly couple the algorithm with the analysis. In a sense, these algorithms are all special cases of Algorithm 3.1 specified by their particular choice of confidence set generator Φ. In turn, these confidence sets are typically dictated via the choice of analytical concentration result, in the spirit of Lemmas 3.1 and 3.2.

As such, optimistic algorithms in the style of UCRL are not separated from their analysis in the sense that inefficient confidence sets in the analysis lead to inefficient confidence sets in the actual algorithm (Russo and Van Roy, 2014). For example, we can improve upon the confidence sets for transition functions in UCRL by replacing the confidence sets $\mathcal{P}_k$ based upon Lemma 3.1 by more nuanced confidence sets $\tilde{\mathcal{P}}_k$ which include second order information (Dann and Brunskill, 2015) or even the entire moment hierarchy via Kullback-Leibler divergence to give KL-UCRL (Filippi et al., 2010). These improved confidence sets will lead to better algorithms, since the confidence sets at each (s, a) are more appropriately calibrated to the data at hand. Better confidence sets in each (s, a) can lead to better empirical performance and better analytical bounds but do not change the overall $\tilde{O}(S\sqrt{AT})$ scaling (Filippi et al., 2010). In fact, there is a more fundamental flaw in the design of these algorithms which can cause even greater inefficiency.

Current optimistic algorithms for reinforcement learning compute confidence sets $\mathcal{R}_k, \mathcal{P}_k$ for each state and action independently and then take the overall confidence set to be the union of each of these confidence sets. This allows for tractable computation of the optimistic optimization via extended value iteration (Jaksch et al., 2010), and also a simple analysis via union bound. However, this also allows the underlying algorithm to use optimistic estimates for the rewards $\overline{r}_k(s, a)$ and transitions $P_k(s, a)$ simultaneously for each (s, a). Together, these give a confidence set of MDPs $\mathcal{M}_k$ which is far too optimistic overall. So far the literature has made great progress in designing improved confidence sets for each (s, a), but it has largely ignored the effects of this over optimism that comes from the union bound (Osband and Van Roy, 2016c).

Rectangular confidence sets

In this section, we highlight the statistical inefficiency that arises from independent confidence sets on each state and action. In order to do this, we consider the confidence sets which are generated for some trivial MDPs with only one action. We describe these MDPs in Figure 3.1. The simplicity of the examples means it is easy to see how to design more effective confidence sets in these contexts. Nonetheless, we will see that confidence sets suggested by OFU-RL can become incredibly mis-calibrated as H and S grow.



(a) MDPs to illustrate the scaling with S.

(b) MDPs to illustrate the scaling with H.

Figure 3.1: Simple environments designed to highlight potential problems with OFU-RL.

Consider the problem of estimating the value $Q(0,0)$ after K episodes of interaction for any MDP in the family of Figure 3.1a. Without any knowledge of the underlying rewards or transitions structure we could apply Lemma 3.2 to the monte-carlo reward per episode $\in \{0,1\}$ to obtain a confidence set with width $\tilde{O}\left(\sqrt{\frac{1}{K}}\right)$. We note that even using this naive approach this bound has no dependence on the number of successor states N. This result should not be particularly surprising since the problem of estimating the expected value Q is not made more difficult by the inclusion of many equivalent outcomes. Surprisingly we will now show that, due to the inefficient rectangular confidence sets at each state, many standard approaches to optimistic RL such as UCRL2 do not fare well in this example, as we will now explain.

Consider the confidence set given by UCRL2 for $Q(0,0)$ after K episodes of interaction with an MDP from Figure 3.1a. For clarity we will consider the even simpler task of estimating Q when only one the rewards or transitions are unknown, but not both. One might reasonably expect that, given this extra prior knowledge, an efficient RL algorithm should be able to estimate confidence sets at least as well as the naive monte-carlo estimates.

In fact, in both cases UCRL2 will result in confidence width $\tilde{O}\left(\sqrt{\frac{N}{K}}\right)$ that degrade with the number of successor states N:

1. **R known, P unknown.** UCRL2 a Hölder decomposition to bound $|Q - \hat{Q}| \leq \|P - \hat{P}\|_1$ in terms of the L1 concentration of the $2N$-dimensional transition estimate P. This term concentrates $\tilde{O}\left(\sqrt{\frac{N}{K}}\right)$ as per Lemma 3.1.
2. **R unknown, P known.** After K episodes the reward estimates of each s will be known up to $\tilde{O}\left(\sqrt{\frac{1}{K(s)}}\right)$, where $K(s)$ is the number of visits to s in the first K episodes. The resultant confidence set $\frac{1}{2N}\sum_{s=1}^{2N}\sqrt{\frac{1}{K(s)}} = \tilde{O}\left(\sqrt{\frac{N}{K}}\right)$ once more.

The looseness of UCRL-style algorithms with respect to (1) and the number of successor states of an unknown transition is well known (Szita and Szepesvári, 2010; Dann and Brunskill, 2015). That this looseness can even occur in (2) with *known* transition dynamics is perhaps less well understood. We now present a geometric argument that highlights the shortcomings of the independent confidence sets used by UCRL2 and other optimistic algorithms (Osband and Van Roy, 2016c).

Typical optimistic approaches to exploration build up confidence sets at each state and action independently and then optimize optimistically over the union of optimistic estimates at each (s, a). In effect, this creates a hypercube which allows for simultaneously optimistic estimates in every component. This resultant optimistic estimate may be far too optimistic even when the estimates at each state and action are optimal. We depict this effect in Figure 3.2, which presents a stylized posterior distribution for the value at any pair of states $s, s' \in \mathcal{S}$. Even if the confidence sets at each state s are well-calibrated, the resultant hyper-rectangle over all $s, s' \in \mathcal{S}$ will be $\sqrt{S}$-times miscalibrated at the extremal corners. The correct geometry for confidence sets in $\mathcal{M}_k$ should be ellipsoidal based upon something similar to the KL-divergence both in estimates for each (s, a, h) and, more importantly, when taken in union across the entire MDP.

Algorithms like KL-UCRL improve the quality of the confidence bounds in each (s, a) individually to give improved ellipsoidal, rather than $L1$, confidence balls around their estimates (Filippi et al., 2010). A similar observation, but only for the second moment of estimates, gives rise to the UCFH algorithm (Dann and Brunskill, 2015). These changes do nothing to improve the union bound across states and actions and so KL-UCRL effectively induces hyper-cylindrical confidence sets and no change to the overall scaling of

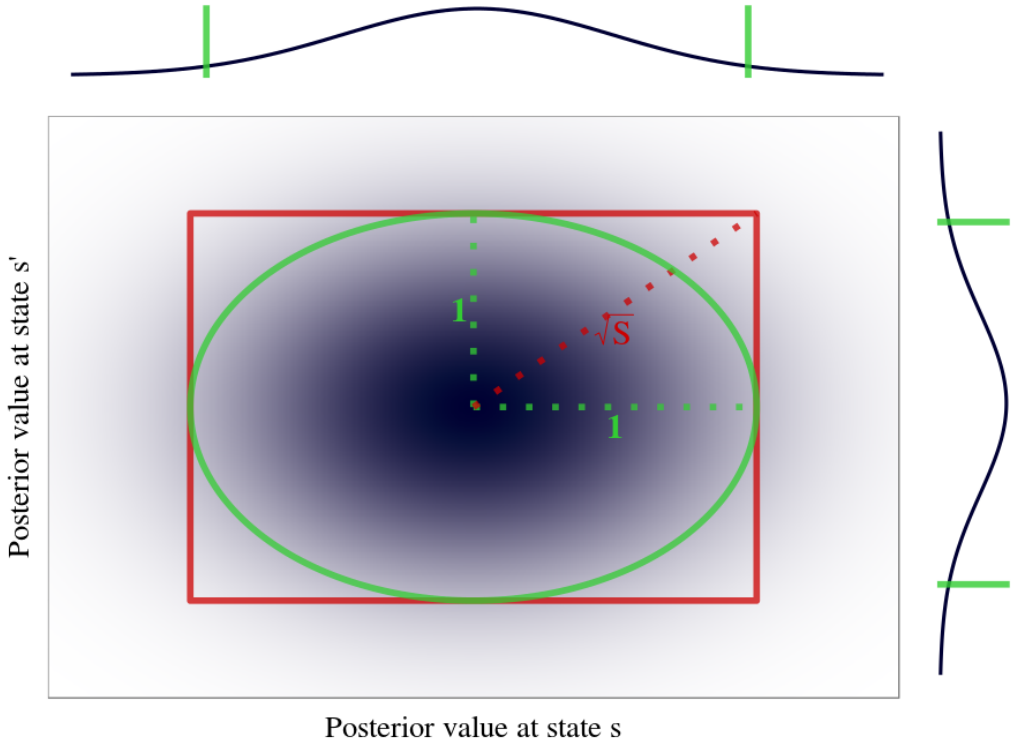


Figure 3.2: Union bounds lead to hyper-rectangles as opposed to an ellipse.

the optimistic algorithm. Worse still, as we will see in the next section, even if an oracle provided a well-calibrated confidence set $\mathcal{M}_k$, the resulting optimistic optimization might computationally intractable.

3.2.2 Computational inefficiency

The hyper-rectangular structure of confidence sets $\mathcal{M}_k$ designed by UCRL leave a lot of room for improvement in terms of statistical efficiency. The problem is that this hyper-rectangular structure is absolutely crucial for the efficient computational implementation via extended value iteration (Strehl et al., 2006; Jaksch et al., 2010). The hyper-rectangular $\mathcal{M}_k$ admit efficient computational solutions via linear programming over linear constraints. By contrast, linear optimization over ellipsoidal confidence sets that do not decompose across each (s, a) is not a tractable problem (Rusmevichientong and Tsitsiklis, 2010; Russo and Van Roy, 2014, 2013; Filippi et al., 2010).

So here we have the bad news for classical optimistic algorithms based upon confidence sets in the style of UCRL. First, we do not have the analytical tools to design effective confidence sets for MDPs which properly address the statistical inefficiencies outlined in Section 3.2.1. Second, this inefficiency is not simply in the analysis, but as we outline in Section 4.3 translates to the actual performance of the optimistic *algorithms* based upon these confidence sets. However, the bigger problem is that even if we did manage to obtain these "correct" confidence sets $\mathcal{M}_k^*$ that held at exactly the desired level of optimism we would be faced with an intractable optimization problem (Osband and Van Roy, 2016c). The problem is that ellipsoidal confidence sets which are coupled through states across time lead to combinatorial constraints which may be NP-hard to optimize. One practical approach to solving such an intractable linear optimization would be bound the confidence set $\mathcal{M}_k^*$ within a larger polygonal confidence set $\mathcal{M}_k$. We could then solve over the larger polygonal confidence set $\mathcal{M}_k$ that admits somewhat efficient computation, but at the cost of decreased statistical efficiency. Even if we had the right confidence sets, we'd still end up with an algorithm very similar to UCRL and with many of the same shortcomings.

Optimism in the face of uncertainty represents a key guiding principle for efficient reinforcement learning. At its core OFU approximates the intractable computation of the value of exploration by a heuristic bonus to encourage exploration. This principle has driven almost the majority progress in efficient RL (Kolter and Ng, 2009; Araya et al., 2012; Asmuth et al., 2009; Brafman and Tennenholtz, 2002; Kearns and Singh, 2002; Munos, 2014; Kocsis and Szepesvári, 2006) and even provably near optimal algorithms such as UCRL2 (Jaksch et al., 2010). At the same time we have outlined several fundamental shortcomings of these algorithms in terms of both statistical and computational efficiency. In this dissertation we will present one way to move beyond these limits of existing optimistic algorithms for simultaneously computationally and statistically efficient RL.

The approach which we explore in this dissertation is to replace the optimistic optimization over confidence sets with a randomized sampling scheme for the value function. In particular, we might consider sampling a single MDP M_k from the posterior distribution for the true MDP M^*, estimating the value function for M_k and following the policy which is optimal for this *sample*. In fact, this algorithm which we call *posterior sampling for reinforcement learning* (PSRL) is the subject of Chapter 4 (Strens, 2000; Osband et al., 2013). Randomization strategies may hold an advantage over deterministically optimistic

strategies, since they can sidestep the (potentially computationally intractable) optimistic optimization step. In fact, we will develop a strong connection between optimism and posterior sampling which has been noted in bandit learning (Russo and Van Roy, 2014). To do this we require a more nuanced approach to optimism, one that works in terms of optimistic random variables rather than deterministic confidence sets.

3.3 Stochastic optimism

Early approaches to optimistic exploration in reinforcement learning categorized states and actions as either "known" or "unknown" (Brafman and Tennenholtz, 2002; Kearns and Singh, 2002). Those states and actions which were "unknown" were afforded some optimistic imagined value until such time as they had been visited sufficiently often to be classified as "known", at which point they replaced this optimistic value with the empirical estimate. These algorithms established the first polynomial learning guarantees, but both the quality of their bounds and their practical performance leave a lot to be desired. Later works replaced this binary classification with a more nuanced region of plausible parameter values for greatly improved practical performance and analytical bounds (Strehl and Littman, 2005; Strehl et al., 2006; Auer et al., 2009; Jaksch et al., 2010).

In this section we will introduce the concept of *stochastic optimism*, which extends the notion of optimism beyond confidence intervals to entire distributions of random variables. Our main goal in this section is to establish conditions under which a randomized value function can drive efficient exploration. We will see that, so long as the sampling distribution is stochastically optimistic for the true distribution, then several of the nice properties and analyses for traditionally optimistic algorithms also hold for randomized value functions. This is important, since we will go on to show that exploration via randomized value functions can demonstrate superior statistical efficiency at lower computational cost than traditional optimistic algorithms. We believe that the condition of stochastic optimism, as given by Definition 3.2 provides a simple and elegant partial ordering amongst random variables.

Definition 3.2 (Stochastic optimism)**.**
Let X and Y be real-valued random variables with finite expectation. We will say that X is stochastically optimistic for Y if and only if for any convex and increasing $u : \mathbb{R} \to \mathbb{R}$:

$$\mathbb{E}\left[u(X)\right] \geq \mathbb{E}\left[u(Y)\right]. \tag{3.13}$$

We will write $X \succcurlyeq_{\text{so}} Y$ for this relation.

An intuitive understanding for Definition 3.2 is that *any* risk-seeking agent with utility $u : \mathbb{R} \to \mathbb{R}$ would prefer payout X to Y if and only if $X \succcurlyeq_{\text{so}} Y$. For brevity if $X \succcurlyeq_{\text{so}} Y$ we will sometimes write that X is optimistic for Y. This characterization of optimism is closely linked to "second order stochastic dominance" (SSD) (Hadar and Russell, 1969). In fact, the two conditions are in some sense dual in that $X \succcurlyeq_{\text{so}} Y$ if and only if $-Y$ is SSD for $-X$. In Proposition 3.2 we present several other equivalent characterizations of stochastic optimism.

Proposition 3.2 (Optimism equivalence)**.**
Let X and Y be real-valued random variables with finite expectation. The following are equivalent:

1. $X \succcurlyeq_{\text{so}} Y$
2. *For any $u : \mathbb{R} \to \mathbb{R}$ convex and increasing* $\mathbb{E}[u(X)] \geq \mathbb{E}[u(Y)]$
3. *For any $\alpha \in \mathbb{R}$,* $\int_{\alpha}^{\infty} \{\mathbb{P}(X \geq s) - \mathbb{P}(Y \geq s)\}\, ds \geq 0$.
4. *For any Z independent of X and Y,* $\mathbb{E}[\max(X, Z)] \geq \mathbb{E}[\max(Y, Z)]$.
5. $X =_D Y + A + W$ *for* $A \geq 0$ *and* $\mathbb{E}\left[W | Y + A\right] = 0$ *for all values* $y + a$.

Proof. This follows from a simple integration by parts (Hadar and Russell, 1969). □

Stochastic optimism induces a partial ordering over random variables in a natural manner. A random variable X is optimistic for Y if it is optimistic under any risk seeking measure. Intuitively this X must have a both a higher mean and be more "spread out" under any measure of "spread". For example, if $X \sim N(\mu_X, \sigma_X^2)$ and $Y \sim N(\mu_Y, \sigma_Y^2)$ then $X \succcurlyeq_{\text{so}} Y$ if and only if $\mu_X \geq \mu_Y$ and $\sigma_X \geq \sigma_Y$. We now relate stochastic optimism to several similar orderings for random variables.

Definition 3.3 (Moment generating dominance)**.**
Let X and Y be real-valued random variables with finite expectation. We will say that X dominates Y in moment generating function (MGF) if and only if for all $t > 0$:

$$\mathbb{E}\left[\exp(tX)\right] \geq \mathbb{E}\left[\exp(tY)\right]. \tag{3.14}$$

We will write $X \succcurlyeq_{\text{mgf}} Y$ for this relation.

Definition 3.4 (Single crossing dominance)**.**
Let X and Y be real-valued random variables with CDFs F_X, F_Y and finite expectation. We say that X single-crossing dominates Y if and only if $\mathbb{E}[X] \geq \mathbb{E}[Y]$ and there a crossing point $a \in \mathbb{R}$ such that:

$$F_X(s) \geq F_Y(s) \iff s \leq a. \tag{3.15}$$

Each of these conditions captures a similar notion that the moments of Y should in some sense be dominated by X. In fact, for the special case $X \sim N(\mu_X, \sigma_X^2)$ and $Y \sim N(\mu_Y, \sigma_Y^2)$ then these three conditions are precisely equivalent

$$X \succcurlyeq_{\text{sc}} Y \iff X \succcurlyeq_{\text{so}} Y \iff X \succcurlyeq_{\text{mgf}} Y \iff \mu_X \geq \mu_Y \text{ and } \sigma_X \geq \sigma_Y. \tag{3.16}$$

However, for the general distributions that are not Gaussian this may not be the case, as the next proposition shows.

Proposition 3.3 (Relating stochastic dominance)**.**
Let X and Y be real-valued random variables with finite expectation then

$$X \succcurlyeq_{\text{sc}} Y \implies X \succcurlyeq_{\text{so}} Y \implies X \succcurlyeq_{\text{mgf}} Y. \tag{3.17}$$

Proof. Suppose $X \succcurlyeq_{\text{sc}} Y$ with single crossing point a, let $I(\alpha) = \int_\alpha^\infty \{\mathbb{P}(X \geq s) - \mathbb{P}(Y \geq s)\}\, ds$. By $X \succcurlyeq_{\text{sc}} Y$ we know $I(\alpha) > 0$ for all $\alpha > a$ and that $I(\alpha)$ is increasing for all $\alpha < a$. But the limit $\lim_{\alpha \to -\infty} I(\alpha) = \mathbb{E}[X] - \mathbb{E}[Y] \geq 0$. Hence $I(\alpha) \geq 0$ for all $\alpha \in \mathbb{R}$. Proposition 3.2 shows that $X \succcurlyeq_{\text{so}} Y$.

Suppose $X \succcurlyeq_{\text{so}} Y$, then $\mathbb{E}[u(X)] \geq \mathbb{E}[u(Y)]$ for all u increasing and convex. The function $u(x) = \exp(tx)$ is increasing and convex for any $t > 0$. Therefore $X \succcurlyeq_{\text{mgf}} Y$. □

Comparatively speaking there is a much greater literature in machine learning regarding moment dominance than stochastic dominance. One natural question is why we require this new definition when the existing literature already has this similar notion. However, in general the converse to Proposition 3.3 will not be true. We provide two simple examples of this in Examples 3.1 and 3.2.

Example 3.1 (Moment generating dominance does not imply optimism)**.**
Consider $X \sim N(0,1)$ *and* $Y \sim \text{Unif}(\{-1,1\})$*. The random variable* Y *has mean zero and bounded in* $[-1,1]$ *and so it is sub-Gaussian with* $X \succcurlyeq_{\text{mgf}} Y$ *(Buldygin and Kozachenko, 1980). However,* X *is not stochastically optimistic for* Y*.*

Proof. Consider the convex and increasing utility function $u(x) = 2x\mathbb{1}\{x > 0\}$. We can compute the expected values $\mathbb{E}[u(X)] = \sqrt{\frac{2}{\pi}} < \mathbb{E}[u(Y)] = 1$. Therefore X is not stochastically optimistic for Y. □

Example 3.2 (Optimism does not imply single crossing dominance)**.**
Consider $Y \sim \text{Unif}(\{-1,1\})$ *and let* $X = Y + W$ *where* $W \sim \text{Unif}([-1,1])$ *and independent of* Y*. By Proposition 3.2* $X \succcurlyeq_{\text{so}} Y$*, however* X *is not single crossing dominant for* Y*.*

Proof. We display the CDFs of these variables in the figure below, they are not single crossing. In particular the ordering of F_X, F_Y switches at least three points $x = -1, 0, 1$. □

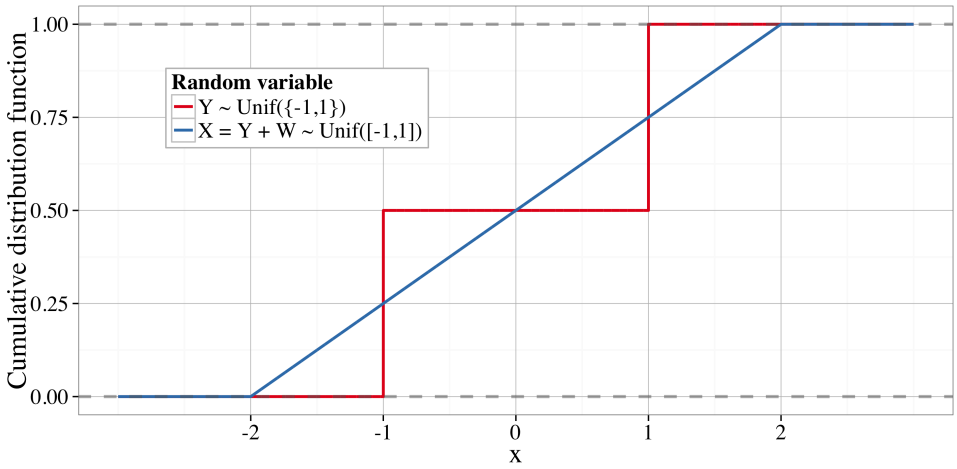


Figure 3.3: Stochastic optimism does not imply single crossing dominance

3.3.1 Posterior sampling

We have now spent several sections deriving mathematical results for this condition of stochastic optimism. Before we dive into another involved analytical proof we provide a simple example that highlights *why* this concept will be so critical to our investigation. Imagine a trivial MDP with $S = H = 1$, this problem is also known as the multi-armed bandit with independent arms. The standard optimistic approach to exploration, which for bandits is known as UCB rather than UCRL (Lai and Robbins, 1985; Auer, 2003) would design confidence sets $\mathcal{R}_k(a)$ for each action at each episode (Algorithm 3.2) and then choose optimistically among these arms. As we detail in Section 3.2 this effectively optimizes optimistically across all arms simultaneously in an A-dimensional hyper-rectangle. This may be far too optimistic, particularly in large action spaces.

Stochastic optimism allows us to consider an alternative approach to optimistic exploration which does not require an inefficient union bound. If we can generate stochastically optimistic samples $r_k(a) \succcurlyeq_{\text{so}} r^*(a)$ independently for each action a then Proposition 3.2 guarantees that the resulting *imagined* optimal reward is optimistic for the true optimal reward $\max_a r_k(a) \succcurlyeq_{\text{so}} \max_a r^*(a)$. Unlike traditional optimistic algorithms the performance of this sampling algorithm is not impacted by any loose union bounds. In particular, if we generate the samples $r_k(a)$ from the posterior distribution from $r^*(a)$ then these samples are guaranteed to be stochastically optimistic, since they are drawn from the same conditional distribution given any prior data. This algorithm is known as posterior or Thompson sampling (Thompson, 1933; Russo and Van Roy, 2014; Agrawal and Goyal, 2012a) and is the focus of our next Chapter 4. Before we move on to this algorithm for improved reinforcement learning we have one important technical lemma.

3.4 Gaussian-Dirichlet optimism

We now present a result that relates the posterior distributions of a matched Gaussian and Dirichlet posterior distribution in terms of stochastic optimism. These two distributions are of fundamental importance to Bayesian inference. In some senses they represent the fundamental conjugate priors for continuous and categorical data respectively. For a refresher on this topic see for example (Gelman et al., 2014). We present two simple pragmatic examples of how to update these Bayesian posteriors below.

Example 3.3 (Updating a Gaussian posterior with known variance)**.**
If the prior distribution for $\mu^* \sim N(\mu_0, \sigma_0^2)$ *and you observe* $D = y_1, .., y_n \sim N(\mu^*, 1)$*, then the posterior distribution for* $\mu^*|D \sim N(\mu, \sigma^2)$*. These posterior parameters are given by*

$$\mu = \frac{\sum_{i=1}^n y_i + \mu_0/\sigma_0}{1/n + 1/\sigma_0} \ , \ \sigma = 1/n + 1/\sigma_0.$$

Example 3.4 (Updating a Dirichlet posterior from categorical data)**.**
The prior distribution for the categorical distribution $P^* \sim \text{Dirichlet}(\alpha_0)$ *over* $\mathcal{S} = \{1, .., S\}$ *with* $\alpha_0 \in \mathbb{R}_+^S$*. If you observe* $D = y_1, .., y_n \sim P^*$ *then the posterior distribution for* $P^*|D \sim \text{Dirichlet}(\alpha))$*. These posterior parameters are given by*

$$\alpha(s) = \alpha_0(s) + \sum_{i=1}^n \mathbb{1}\{y_i = s\} \text{ for } s = 1, .., S.$$

Lemma 3.3 concerns a posterior distribution over P^* and observe categorical results $s_i \in \mathcal{S} = \{1, .., S\}$. We imagine that for each $s \in \mathcal{S}$ the agent receives a known associated reward $V_s \in [0, 1]$. Our result establishes that the resulting posterior distribution for $Y = (P^*)^T V$ (which does not admit a closed form solution) can be related to an approximate Gaussian posterior which ignores the underlying categorical structure and simply updates as if the underlying system were drawn IID from $N(\mu, 1)$. In particular, no matter what data we see the approximate Gaussian posterior will always be stochastically optimistic for Y. This result will be important for efficient exploration in reinforcement learning, since it establishes that it is possible to learn the future value function without explicitly bounding the estimate of P^*.

Lemma 3.3 (Gaussian vs Dirichlet optimism)**.**
Let $Y = P^T V$ *for* $V \in [0, 1]^S$ *fixed and* $P \sim \text{Dirichlet}(\alpha)$ *with* $\alpha \in \mathbb{R}_+^S$ *and* $\sum_{i=1}^S \alpha_i \geq 2$*. Let* $X \sim N(\mu, \sigma^2)$ *with* $\mu = \frac{\sum_{i=1}^S \alpha_i V_i}{\sum_{i=1}^S \alpha_i}$, $\sigma^2 = \left(\sum_{i=1}^S \alpha_i\right)^{-1}$*, then* $X \succcurlyeq_{\text{so}} Y$*.*

The complete proof of this lemma is long, but the essential argument is simple. We outline the main arguments below and fill in the details in Section 3.4.1 and 3.4.2.

First, we prove that $\tilde{Y} \sim \text{Beta}(\tilde{\alpha}, \tilde{\beta})$ is stochastically optimistic for Y when $\tilde{\alpha} = \sum_{i=1}^S \alpha_i V_i$ and $\tilde{\beta} = \sum_{i=1}^S \alpha_i - \tilde{\alpha}$ via direct algebra. This is the content of Lemma 3.5. Next, we show that the Gaussian approximate posterior X is single crossing dominant for this Beta distribution $X \succcurlyeq_{\text{sc}} \tilde{Y}$. Therefore, by Proposition 3.3 $X \succcurlyeq_{\text{so}} Y$.

The main difficulty in this proof comes in establishing $X \succcurlyeq_{\text{sc}} \tilde{Y}$. To do this we use a laborious calculus argument together with repeated applications of the mean value theorem. Our proof requires separate upper and lower bounds for different regions of $\tilde{\alpha}$ and $\tilde{\beta}$, but no real insight beyond that. Most readers will probably be better served by skipping over the details of this proof upon a first reading of this dissertation and moving directly to Chapter 4. We believe that there should be a much more enlightened and elegant method to obtain these results.

3.4.1 Beta vs Dirichlet

In order to prove Lemma 3.3 we will first prove an intermediate result that shows a particular Beta distribution $\tilde{Y}$ is optimistic for Y. Before we can prove this result we first state a more basic result that we will use on Gamma distributions.

Lemma 3.4. *For independent random variables* $\gamma_1 \sim \mathit{Gamma}(k_1, \theta)$ *and* $\gamma_2 \sim \mathit{Gamma}(k_2, \theta)$, $\mathbb{E}[\gamma_1|\gamma_1+\gamma_2] = \frac{k_1}{k_1+k_2}(\gamma_1+\gamma_2)$ *and* $\mathbb{E}[\gamma_2|\gamma_1+\gamma_2] = \frac{k_2}{k_1+k_2}(\gamma_1+\gamma_2)$.

We can now present our first result, relating Dirichlet to Beta distributions.

Lemma 3.5. *Let* $Y=P^\top V$ *for the random variable* $P\sim\text{Dirichlet}(\alpha)$ *and constants* $V\in\mathbb{R}^S$ *and* $\alpha\in\mathbb{R}^S_+$. *Without loss of generality, assume* $V_1\leq V_2\leq\cdots\leq V_S$. *Let* $\tilde{\alpha}=\sum_{i=1}^s\alpha_i(V_i-V_1)/(V_d-V_1)$ *and* $\tilde{\beta}=\sum_{i=1}^d\alpha_i(V_d-V_i)/(V_d-v_1)$. *Then, there exists a random variable* $\tilde{p}\sim\text{Beta}(\tilde{\alpha},\tilde{\beta})$ *such that, for* $\tilde{Y}=\tilde{p}V_d+(1-\tilde{p})V_1$, $\mathbb{E}[\tilde{Y}|Y]=\mathbb{E}[Y]$.

Proof. Let $\gamma_i = \text{Gamma}(\alpha, 1)$, with $\gamma_1, \ldots, \gamma_d$ independent, and let $\overline{\gamma} = \sum_{i=1}^d \gamma_i$, so that $P \equiv_D \gamma/\overline{\gamma}$. Let $\alpha_i^0 = \alpha_i(V_i-V_1)/(V_d-V_1)$ and $\alpha_i^1 = \alpha_i(V_d-V_i)/(V_d-V_1)$ so that $\alpha = \alpha^0+\alpha^1$. Define independent random variables $\gamma^0 \sim \text{Gamma}(\alpha_i^0, 1)$ and $\gamma^1 \sim \text{Gamma}(\alpha_i^1, 1)$ so that $\gamma \equiv_D \gamma^0 + \gamma^1$.

Take γ^0 and γ^1 to be independent, and couple these variables with γ so that $\gamma = \gamma^0+\gamma^1$. Note that $\tilde{\beta} = \sum_{i=1}^d \alpha_i^0$ and $\tilde{\alpha} = \sum_{i=1}^d \alpha_i^1$. Let $\overline{\gamma}^0 = \sum_{i=1}^d \gamma_i^0$ and $\overline{\gamma}^1 = \sum_{i=1}^d \gamma_i^1$, so that $1-\tilde{p} \equiv_D \overline{\gamma}^0/\overline{\gamma}$ and $\tilde{p} \equiv_D \overline{\gamma}^1/\overline{\gamma}$. Couple these variables so that $1-\tilde{p} = \overline{\gamma}^0/\overline{\gamma}$ and $\tilde{p} = \overline{\gamma}^1/\overline{\gamma}$.

We can now say

$$\begin{aligned}
\mathbb{E}[\tilde{Y}|Y] &= \mathbb{E}[(1-\tilde{p})V_1 + \tilde{p}V_d|Y] = \mathbb{E}\left[\frac{V_1\overline{\gamma}^0}{\overline{\gamma}} + \frac{V_d\overline{\gamma}^1}{\overline{\gamma}}\Big|Y\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[\frac{V_1\overline{\gamma}^0 + V_d\overline{\gamma}^1}{\overline{\gamma}}\Big|\gamma, Y\right]\Big|Y\right] = \mathbb{E}\left[\frac{V_1\mathbb{E}[\overline{\gamma}^0|\gamma] + V_d\mathbb{E}[\overline{\gamma}^1|\gamma]}{\overline{\gamma}}\Big|Y\right] \\
&= \mathbb{E}\left[\frac{V_1\sum_{i=1}^d \mathbb{E}[\gamma_i^0|\gamma_i] + V_d\sum_{i=1}^d xp[\gamma_i^1|\gamma_i]}{\overline{\gamma}}\Big|Y\right] \\
&\overset{(a)}{=} \mathbb{E}\left[\frac{V_1\sum_{i=1}^d \gamma_i\alpha_i^0/\alpha_i + V_d\sum_{i=1}^d \gamma_i\alpha_i^1/\alpha_i}{\overline{\gamma}}\Big|Y\right] \\
&= \mathbb{E}\left[\frac{V_1\sum_{i=1}^d \gamma_i(V_i - V_1) + V_d\sum_{i=1}^d \gamma_i(V_d - V_i)}{\overline{\gamma}(V_d - V_1)}\Big|Y\right] \\
&= \mathbb{E}\left[\frac{\sum_{i=1}^d \gamma_i V_i}{\overline{\gamma}}\Big|Y\right] = \mathbb{E}\left[\sum_{i=1}^d p_i V_i\Big|Y\right] = Y,
\end{aligned}$$

where (a) follows from Lemma 3.4. This completes our proof of stochastic optimism.

□

3.4.2 Gaussian vs Beta

We will now show that the Gaussian random variable X is optimistic for $\tilde{Y}$ and so complete the proof of Lemma 3.3, $X \succcurlyeq_{\text{so}} \tilde{Y} \succcurlyeq_{\text{so}} Y$. The remainder of this appendix is dedicated to proving Lemma 3.6, which will complete the proof of Lemma 3.3 through Proposition 3.3.

Lemma 3.6. *Let $\tilde{Y} \sim Beta(\alpha, \beta)$ for any $\alpha > 0, \beta > 0$ and $x \sim N\left(\mu = \frac{\alpha}{\alpha+\beta}, \sigma^2 = \frac{1}{\alpha+\beta}\right)$. Then, $X \succcurlyeq_{\text{sc}} \tilde{Y}$ whenever $\alpha + \beta \geq 2$.*

Double crossing PDFs

We want to prove that the CDFs cross at most once on $(0, 1)$. By the mean value theorem (Rudin, 1964), it is sufficient to prove that the PDFs cross at most twice on the same interval. We lament that the proof as it stands is so laborious, but our attempts at a more elegant solution has so far been unsuccessful. The remainder of this appendix is devoted to proving this "double-crossing" property via manipulation of the PDFs for different values of α, β.

We write f_N for the density of the Normal X and f_B for the density of the Beta $\tilde{Y}$ respectively. We know that at the boundary $f_N(0-) > f_B(0-)$ and $f_N(1+) > f_B(1+)$ where the $\pm$ represents the left and right limits respectively. As these densities are positive over the interval, we can consider the log PDFs

$$l_B(x) = (\alpha - 1)\log(x) + (\beta - 1)\log(1 - x) + K_B$$

$$l_N(x) = -\frac{1}{2}(\alpha + \beta)\left(x - \frac{\alpha}{\alpha + \beta}\right)^2 + K_N.$$

The function $\log(x)$ is injective and increasing, if we can show that $l_N(x) - l_B(x) = 0$ has at most two solutions on the interval we would be done.

Instead we will attempt to prove an even stronger condition, that $l'_N(x) - l'_B(x) = 0$ has at most one solution in the interval. This sufficient condition may be easier to deal with since we can ignore the distributional normalizing constants.

$$l'_B(x) = \frac{\alpha - 1}{x} - \frac{\beta - 1}{1 - x}\ , \quad l'_N(x) = \alpha - (\alpha + \beta)x$$

Finally we consider an even stronger condition, if $l''_N(x) - l''_B(x) = 0$ has no solution then $l'_B(x) - l'_N(x)$ must be monotone over the region and so it can have at most one root.

$$l''_B(x) = -\frac{\alpha - 1}{x^2} - \frac{\beta - 1}{(1 - x)^2}\ , \quad l''_N(x) = -(\alpha + \beta)$$

With these definitions now let us define:

$$h(x) := l''_N(x) - l''_B(x) = \frac{\alpha - 1}{x^2} + \frac{\beta - 1}{(1 - x)^2} - (\alpha + \beta) \tag{3.18}$$

Our goal now is to show that $h(x) = 0$ does not have any solutions for $x \in [0, 1]$. Once again, we will look at the derivatives and analyze them for different values of $\alpha, \beta > 0$.

$$h'(x) = -2\left(\frac{\alpha - 1}{x^3} - \frac{\beta - 1}{(1 - x)^3}\right)$$

$$h''(x) = 6\left(\frac{\alpha - 1}{x^4} + \frac{\beta - 1}{(1 - x)^4}\right)$$

Proof regions

Our proof will proceed by considering specific ranges for the values of $\alpha, \beta > 0$ and use different calculus arguments for each of these regions. By symmetry in the problem, we only need to prove the result for $\alpha > \beta$. Within this section of possible parameter values we will need to subdivide the quadrant into three proof regions. These regions completely cover all $\alpha + \beta \geq 2$ and so are sufficient to complete the proof of Lemma 3.6.

- $R1 := \{\alpha > 1 \geq \beta \geq 0\}$.
- $R2 := \{\alpha > 1, \beta > 1, (\alpha - 1)(\beta - 1) \geq 1/9\}$.
- $R3 := \{\alpha > 1, \beta > 1, (\alpha - 1)(\beta - 1) < 1/9\}$.

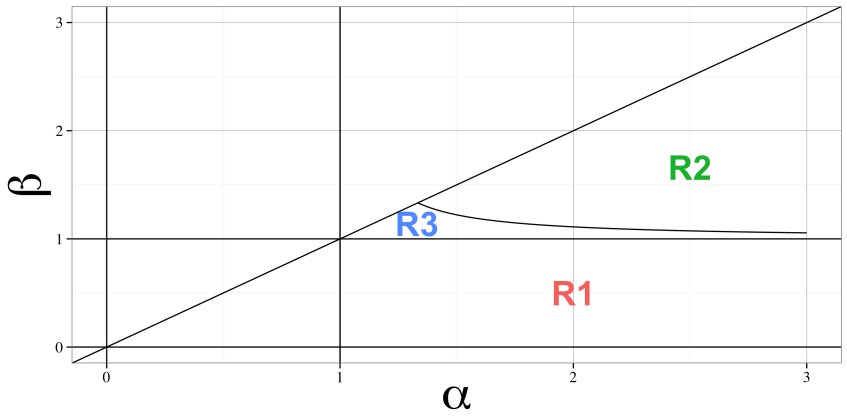


Figure 3.4: Parameter regions for proof. The special case $\alpha = \beta = 1$ can be verified individually, in this case the PDFs do not intersect at any point.

Region $R1 = \{\alpha \geq 1 \geq \beta \geq 0\}$

In this region we will show that $g(x) = l'_N(x) - l'_B(x)$ has no solutions. We write $A = \alpha - 1 > 0$ and $B = \beta - 1 \leq 0$ as before.

$$g(x) = \alpha - (\alpha + \beta)x + \frac{\beta - 1}{1 - x} - \frac{\alpha - 1}{x}$$

$$g'(x) = h(x) = \frac{A}{x^2} + \frac{B}{(1 - x)^2} - (\alpha + \beta)$$

$$g''(x) = h'(x) = -2\left(\frac{A}{x^3} - \frac{B}{(1 - x)^3}\right)$$

We note that $g''(x) \leq 0$ and so $g(x)$ is a concave function. If we can show that the maximum of g lies below 0 then we know that there can be no roots. We now attempt to solve $g'(x) = 0$:

$$\begin{aligned} g'(x) &= \frac{A}{x^2} + \frac{B}{(1-x)^2} = 0 \\ \implies -A/B &= \left(\frac{x}{1-x}\right)^2 \\ \implies x &= \frac{K}{1+K} \in (0,1) \end{aligned}$$

Where here we write $K = \sqrt{-A/B} > 0$, we ignore the case $B = 0$ as this is trivial. We write $C = -B \geq 0$ and evaluate the function g at its minimum $x_K = \frac{K}{1+K}$.

$$\begin{aligned} g(x_K) &= (A+1) - (A+B+2)\frac{K}{1+K} + B(1+K) - A\frac{1+K}{K} \\ &= -AK^2 - AK - A + BK^3 + BK^2 + BK - K^2 + K \\ &= -AK^2 - AK - A - CK^3 - CK^2 - CK - K^2 + K \\ &= -A(A/C) - A(A/C)^{1/2} - A - C(A/C)^{3/2} - C(A/C) - C(A/C)^{1/2} - A/C + (A/C)^{1/2} \\ &= -A^2C^{-1} - A^{3/2}C^{-1/2} - A - A^{3/2}C^{-1/2} - A - A^{1/2}C^{1/2} - AC^{-1} + A^{1/2}C^{1/2} \\ &= -A^2C^{-1} - 2A^{3/2}C^{-1/2} - 2A - AC^{-1} \leq 0 \end{aligned}$$

Therefore the Lemma holds for all $\alpha, \beta \in R1$

Region $R2 = \{\alpha > 1, \beta > 1, (\alpha-1)(\beta-1) \geq 1/9\}$

In the case of $\alpha, \beta > 1$ we know that $h(x)$ is a convex function on $(0,1)$. If we solve $h'(x^*) = 0$ and $h(x^*) > 0$ then we have proved our statement. We will write $A = \alpha - 1, B = \beta - 1$ for convenience.

First we solve $h'(x) = 0$ in terms of $K = (A/B)^{1/3} > 0$,

$$\begin{aligned} h'(x) &= \frac{A}{x^3} - \frac{B}{(1-x)^3} = 0 \\ \implies A/B &= \left(\frac{x}{1-x}\right)^3 \\ \implies x &= \frac{K}{1+K} \in (0,1). \end{aligned}$$

We can now evaluate the function h at its minimum $x_K = \frac{K}{1+K}$.

$$\begin{aligned} h(x_K) &= A\frac{(K+1)^2}{K^2} + B(K+1)^2 - (A+B+2) \\ &= A(2/K + 1/K^2) + B(K^2+2K) - 2 \\ &= 3(A^{2/3}B^{1/3} + A^{1/3}B^{2/3}) - 2. \end{aligned}$$

As long as $h(x_K) > 0$ we have shown that the CDFs are single crossing. We note that for all $\alpha, \beta \in R2$

$$A, B \geq 1/3 \implies AB \geq 1/9 \implies (A^{2/3}B^{1/3} + A^{1/3}B^{2/3}) \geq 2/3.$$

This completes the proof for $R2$.

Region $R3 = \{\alpha > 1, \beta > 1, (\alpha-1)(\beta-1) < 1/9\}$

Our argument for this final region is really no different than before, the only difficulty is that it is not enough to only look at the derivatives of the log likelihoods, we need to use some bound on the normalizing constants to get our bounds.

In $R3$, we know that $\beta \in (1, \frac{4}{3})$ so we will make use of an upper bound to the normalizing constant of the Beta distribution, the Beta function.

$$B(\alpha, \beta) = \int_{x=0}^{1} x^{\alpha-1}(1-x)^{\beta-1}dx \leq \int_{x=0}^{1} x^{\alpha-1}dx = \frac{1}{\alpha} \tag{3.19}$$

Our thinking is that, because in $R3$ the value of $\beta - 1$ is relatively small, this approximation will not be too bad. Therefore, we can explicitly bound the log likelihood of the Beta distribution:

$$l_B(x) \geq \tilde{l}_B(x) := (\alpha-1)\log(x) + (\beta-1)\log(1-x) + \log(\alpha)$$

We now use a similar calculus argument to before. We want to find two points $x_1 < x_2$ for which $h(x_i) = l''_N(x) - l''_B(x) > 0$. Since $\alpha, \beta > 1$ we know that h is convex and so for all $x \notin [x_1, x_2]$ then $h > 0$. We define the gap of the Beta over the maximum of the normal

log likelihood

$$\text{Gap} : l_B(x_i) - l_N(x_i) \geq f(x_i) := \tilde{l}_B(x_i) - \max_x l_N(x) > 0. \tag{3.20}$$

If we can show the gap is positive then it must mean there are no crossings over the region $[x_1, x_2]$. This is because $\tilde{l}_B$ is concave and therefore totally above the maximum of l_N over the whole region $[x_1, x_2]$.

Consider any $x \in [0, x_1)$, we know from the ordering of the tails of the CDF that if there is more than one root in this segment then there must be at least three crossings. If there are three crossings, then the second derivative of their difference h must have at least one root on this region. However we know that h is convex, so if we can show that $h(x_i) > 0$ this cannot be possible. We use a similar argument for $x \in (x_2, 1]$. We will now complete this proof by lengthy amounts of calculus.

Let's remind ourselves of the definition:

$$h(x) := l''_N(x) - l''_B(x) = \frac{\alpha - 1}{x^2} + \frac{\beta - 1}{(1-x)^2} - (\alpha + \beta)$$

For ease of notation we will write $A = \alpha - 1, B = \beta - 1$. We note that:

$$h(x) \geq h_1(x) = \frac{A}{x^2} - (A + B + 2)$$

$$h(x) \geq h_2(x) = \frac{B}{(1-x)^2} - (A + B + 2)$$

and we solve for $h_1(x_1) = 0, h_2(x_2) = 0$. This means that

$$x_1 = \sqrt{\frac{A}{A+B+2}}\ ,\ x_2 = 1 - \sqrt{\frac{B}{A+B+2}}$$

and clearly $h(x_1) > 0, h(x_2) > 0$. Now, if we can show that, for all possible values of A, B in this region $f(x_i) = l_B(x_i) - \max_x l_N(x) > 0$, our proof will be complete.

To make the dependence on A, B more clear we write $f(x_i) = f_i(A, B)$ below

$$f_1(A,B) = \log(1+A) + A\log\left(\sqrt{\frac{A}{A+B+2}}\right) + B\log\left(1 - \sqrt{\frac{A}{A+B+2}}\right) + \frac{1}{2}\log(2\pi) - \frac{1}{2}\log(A+B+2),$$

$$f_2(A,B)=\log(1+A)+A\log\left(1-\sqrt{\frac{B}{A+B+2}}\right)+B\log\left(\sqrt{\frac{B}{A+B+2}}\right)+\frac{1}{2}\log(2\pi)-\frac{1}{2}\log(A+B+2).$$

We will demonstrate that $\frac{\partial f_i}{\partial B}\leq 0$ for all of the values in our region $A>B>0$.

$$\begin{aligned}
\frac{\partial f_1}{\partial B} &= -\frac{A}{2(A+B+2)}+\log\left(1-\sqrt{\frac{A}{A+B+2}}\right)+\frac{B\sqrt{A}}{2(A+B+2)^{3/2}\left(1-\sqrt{\frac{A}{A+B+2}}\right)}-\frac{1}{2(A+B+2)} \\
&= \frac{1}{2(A+B+2)}\left(\frac{B\sqrt{A}}{\sqrt{A+B+2}\left(1-\sqrt{\frac{A}{A+B+2}}\right)}-A-1\right)+\log\left(1-\sqrt{\frac{A}{A+B+2}}\right) \\
&= \frac{1}{2(A+B+2)}\left(\frac{B\sqrt{A}}{\sqrt{A+B+2}-\sqrt{A}}-A-1\right)+\log\left(1-\sqrt{\frac{A}{A+B+2}}\right) \\
&\leq \frac{1}{2(A+B+2)}\left(\frac{\sqrt{B}/3}{\sqrt{A+B+2}-\sqrt{A}}-A-1\right)-\sqrt{\frac{A}{A+B+2}} \\
&\leq \frac{1}{2(A+B+2)}\left(\frac{1}{3}\sqrt{\frac{B}{B+2}}-A-1\right)-\sqrt{\frac{A}{A+B+2}} \\
&\leq -\frac{A}{2(A+B+2)}-\sqrt{\frac{A}{A+B+2}} \ \leq 0.
\end{aligned}$$

Similarly,

$$\begin{aligned}
\frac{\partial f_2}{\partial B} &= -A\left(\frac{\sqrt{\frac{B}{A+B+2}}}{2B}+\frac{1}{2(A+B+2)}\right)+\log\left(\sqrt{\frac{B}{A+B+2}}\right)+B\left(\frac{A+2}{2B(A+B+2)}\right)-\frac{1}{2(A+B+2)} \\
&= \frac{1}{2(A+B+2)}\left(A+2-A-1-A\sqrt{\frac{A+B+2}{B}}\right)+\log\left(\sqrt{\frac{B}{A+B+2}}\right) \\
&= \frac{1}{2(A+B+2)}\left(1-A\sqrt{\frac{A+B+2}{B}}\right)+\frac{1}{2}\log\left(\frac{B}{A+B+2}\right).
\end{aligned}$$

Therefore, for any $A\geq 0$ this means that $\frac{\partial^2 f_2}{\partial A\partial B}<0$. Therefore this expression $\frac{\partial f_2}{\partial B}$ is maximized over A for $A=0$. We can evaluate this expression explicitly:

$$\frac{\partial f_2}{\partial B}\Big|_{A=0} = \frac{1}{2(B+2)}+\frac{1}{2}\log\left(\frac{B}{B+2}\right) \leq \frac{1}{2}\left(\frac{1}{B+2}+\frac{B}{B+2}-1\right) \leq 0.$$

This provides a monotonicity result which states that both f_1, f_2 are minimized at at the largest possible $B=\frac{1}{9A}$ for any given A over our region. We will now write $g_i(A):=f_i(A,\frac{1}{9A})$. If we can show that $g_i(A)\geq 0$ for all $A\geq\frac{1}{3}$ and $i=1,2$ we will be done with our proof. We

will perform a similar argument to show that g_i is monotone increasing for all $A \geq \frac{1}{3}$.

$$\begin{aligned} g_1(A) &= \log(1+A) + A\log\left(\sqrt{\frac{A}{A+\frac{1}{9A}+2}}\right) + \frac{1}{9A}\log\left(1-\sqrt{\frac{A}{A+\frac{1}{9A}+2}}\right) + \frac{1}{2}\log(2\pi) - \frac{1}{2}\log(A+\frac{1}{9A}+2) \\ &= \log(1+A) + \frac{A}{2}\log(A) - \frac{1}{2}(1+A)\log(A+\frac{1}{9A}+2) + \frac{1}{9A}\log\left(1-\sqrt{\frac{A}{A+\frac{1}{9A}+2}}\right) + \frac{1}{2}\log(2\pi) \end{aligned}$$

Note that the function $p(A) = A + \frac{1}{9A}$ is increasing in A for $A \geq \frac{1}{3}$. We can conservatively bound g from below noting $\frac{1}{9A} \leq 1$ in our region.

$$\begin{aligned} g_1(A) &\geq = \log(1+A) + \frac{A}{2}\log(A) - \frac{1}{2}(1+A)\log(A+3) + \frac{1}{9A}\log\left(1-\sqrt{\frac{A}{A+2}}\right) + \frac{1}{2}\log(2\pi) \\ &\geq \log(1+A) + \frac{A}{2}\log(A) - \frac{1}{2}(1+A)\log(A+3) - \frac{1}{9A}\sqrt{A} + \frac{1}{2}\log(2\pi) =: \tilde{g}_1(A). \end{aligned}$$

We can use calculus to say that:

$$\begin{aligned} \tilde{g}_1'(A) &= \frac{1}{A+1} + \frac{1}{A+3} + \frac{\log(A)}{2} + \frac{1}{18A^{3/2}} - \frac{1}{2}\log(A+3) \\ &\geq \frac{1}{A+1} + \frac{1}{A+3} + \frac{1}{18A^{3/2}} + \frac{1}{2}\log(\frac{A}{A+3}) \end{aligned}$$

This expression is monotone decreasing in A and with a limit ≥ 0. Therefore $g_1(A) \geq \tilde{g}_1(A) \geq \tilde{g}_1(1/3)$ for all A. We can explicitly evaluate this numerically and $\tilde{g}_1(1/3) > 0.01$ so we are done. The final piece of this proof involves a similar argument for $g_2(A)$.

$$\begin{aligned} g_2(A) &= \log(1+A) + A\log\left(1-\sqrt{\frac{\frac{1}{9A}}{A+\frac{1}{9A}+2}}\right) + \frac{1}{9A}\log\left(\sqrt{\frac{\frac{1}{9A}}{A+\frac{1}{9A}+2}}\right) \\ &\quad + \frac{1}{2}\log(2\pi) - \frac{1}{2}\log(A+\frac{1}{9A}+2) \\ &= \log(1+A) + A\log\left(1-\sqrt{\frac{1}{9A^2+18A+1}}\right) + \frac{1}{2}\left(\frac{1}{9A}\log\left(\frac{1}{9A}\right)\right) \\ &\quad - \frac{1}{2}\left(\frac{1}{9A}+1\right)\log\left(A+\frac{1}{9A}+2\right) + \frac{1}{2}\log(2\pi) \\ &\geq \log(1+A) + A\left(-\frac{1}{\sqrt{9A^2}}\right) + \frac{1}{2}\left(\frac{1}{9A}\log\left(\frac{1}{9A}\right)\right) - \frac{1}{2}\left(\frac{1}{3}+1\right)\log\left(A+\frac{1}{3}+2\right) + \frac{1}{2}\log(2\pi) \\ &\geq \log(1+A) - \frac{1}{3} - \frac{1}{2e} - \frac{2}{3}\log(A+\frac{7}{3}) + \frac{1}{2}\log(2\pi) =: \tilde{g}_2(A) \end{aligned}$$

Once again we can see that $\tilde{g}_2$ is monotone increasing

$$\tilde{g}_2'(A) = \frac{1}{1+A} - \frac{2/3}{A+7/3} = \frac{A+5}{(A+1)(3A+7)} \geq 0.$$

We complete the argument by noting $g_2(A) \geq \tilde{g}_2(A) \geq \tilde{g}_2(1/3) > 0.01$. This concludes our proof of the PDF double crossing in this region. □

Recap

By symmetry, the results above established Lemma 3.6 for a region that includes all $\alpha+\beta \geq 2$. We can combine Lemma 3.5 with Lemma 3.6 and Proposition 3.3 to complete the proof of Lemma 3.3. The resulting constraint that $\sum_{i=1}^{S} \alpha_i \geq 2$ is technical and does not pose any real difficulties for our analysis going forward.

The analysis in this section is involved, laborious and uninspiring, even the result we prove in Lemma 3.3 is of little practical interest in and of itself. However, as we will see in the next chapter, this fundamental lemma allows us to produce new concentration results for the posterior distribution of the value function (2.2) without generating individual confidence sets for the transition dynamics first. This result is key to establishing the first optimal $\tilde{O}(\sqrt{SAT})$ regret bounds in Theorem 4.2 for reinforcement learning guided by *stochastic* optimism. We believe that a more elegant and potentially far more general result may be available to future work, we provide hints towards this in Section 4.4.1 and in particular through Conjecture 4.2.

Chapter 4

Posterior Sampling for Reinforcement Learning

Almost all analytical progress in efficient exploration has been guided by the principle of optimism in the face of uncertainty. In this chapter we study an alternative approach to exploration based upon posterior sampling. We model the agent's initial uncertainty over the environment through a prior distribution. At the start of each episode the agent samples an environment from its posterior distribution, conditional upon the data which it has observed, and follows the policy which is optimal for that *sample* for the duration of the episode. This simple approach to exploration is one of the oldest and is often known as Thompson sampling or probability matching (Thompson, 1933).

We begin this chapter with a review of the Thompson sampling heuristic. We demonstrate that naive applications of Thompson sampling are not efficient for RL, since they do not perform deep exploration. To remedy this shortcoming, we present the simple algorithm *posterior sampling for reinforcement learning* (PSRL), which uses posterior sampling to drive deep exploration. Next, we relate PSRL to well-designed optimistic algorithms via stochastic optimism. We provide insight to why PSRL can remedy the statistical inefficiencies of traditional approaches to optimistic RL while remaining computationally tractable. We leverage this insight to establish the first $\tilde{O}(\sqrt{SAT})$ Bayesian regret bounds for reinforcement learning, which match the fundamental lower bounds up to logarithmic factors.

We place these bounds in the context of the literature, with special attention to their differences and shortcomings. These theoretical results are supported by simulation.

4.1 Thompson sampling and probability matching

One of the oldest heuristics for balancing exploration with exploitation is probability matching. In this algorithm, the agent maintains a posterior distribution for its beliefs over the optimal action. A greedy algorithm would then pick the arm with the highest expected return but, as we saw in Section 2.4, this purely exploitative strategy might never learn the optimal policy. Probability matching instead randomly chooses its action randomly, according the probability that it is the optimal action. This randomized sampling incentives exploration of poorly-understood states and actions through the variance of the posterior. As more data is gathered, the posterior samples will concentrate around the true value, naturally balancing exploration with exploitation. This approach is also known as Thompson sampling after one of its original proponents for clinical trials (Thompson, 1933).

Despite its long history, Thompson sampling was largely neglected by the multi-armed bandit literature until several empirical studies demonstrated state of the art performance (Chapelle and Li, 2011; Scott, 2010). This prompted a surge of interest, and a variety of strong theoretical guarantees are now available in the bandit setting (Russo and Van Roy, 2014, 2013; Agrawal and Goyal, 2012a; Kaufmann et al., 2012; Agrawal and Goyal, 2012b). Thompson sampling can be applied to reinforcement learning problems as well, as we outline in Algorithm 4.1. At every timestep, Thompson sampling takes a single sample from its posterior for the environment and then follows the policy which is optimal for that sample.

Algorithm 4.1 Thompson sampling for reinforcement learning

Input: Prior distribution for MDPs ϕ.
for episode $k = 1, 2, ..$ **do**
  for time $h = 1, 2, .., H$ **do**
    sample MDP $M_{kh} \sim \phi(\cdot|\mathcal{H}_{kh})$
    compute $\mu_k \in \arg\max_{\mu} V_{\mu,h}^{M_{kh}}$
    take action $a_{kh} = \mu_k(s_{kh}, h)$
    observe r_{kh} and s_{kh+1}
    update $\mathcal{H}_{kh} = \mathcal{H}_{kh} \cup (a_{kh}, r_{kh}, s_{kh+1})$
  end for
end for

At first glance Thompson sampling is relatively similar to Boltzmann exploration, both algorithms randomize they policy proportional to some estimate of the value. Importantly, and unlike Boltzmann exploration, Thompson sampling directs its exploration efficiently towards policies that are either exploratory or exploitative. This key difference means that Thompson sampling is provably efficient for bandits, where Boltzmann exploration is provably inefficient (Russo and Van Roy, 2014).

In bandits, this choice of directed exploration rather than undirected generally categorizes efficient algorithms. The story in RL is not as simple, directed exploration is not enough to guarantee efficiency; the exploration must also be deep (Osband et al., 2016). Thompson sampling for reinforcement learning does not implement deep exploration, since it resamples a new policy every single timestep. This can lead to exponentially slower learning, as we present in the example below.

Example 4.1 (Thompson sampling for RL may take exponentially long to learn)**.** *For a finite MDP* $M^* = (\mathcal{S}, \mathcal{A}, R, P, H, \rho)$ *Thompson sampling may take* $\Omega(2^S)$ *steps to learn an* ϵ*-optimal policy, even when the prior is correctly specified* $M^* \sim \phi$*.*

Proof. Imagine a long chain of states with $\mathcal{S} = \{-N, -N-1, .., N\}$, the actions are deterministic "left" and "right". The episode length $H = N$ and very episode the agent deterministically begins in state 0. All rewards in the MDP are zero, but at precisely one of the ends the reward is 1. The agent does not know which end is rewarding a priori, but they do know that each end is equally likely. We depict these simple MDPs in Figure 4.1. The optimal exploration strategy can learn this MDP within a single episode. The agent should visit one end of the chain in the first episode, if there is a reward they should repeat this policy otherwise they go to the other. Despite this simple structure, an agent using Thompson sampling will take exponentially long to see any reward.

To see why this is the case we must first note that an agent receives no information or reward for any state which is not a the end, since they are identical in either possible MDP. Let k^* be the first episode in which the agent reaches an informative state. For all episodes $k < k^*$ each time the agent samples a policy it chooses to go left or go right with equal probability. The probability of reaching an informative state under this policy is 2^{-N}, therefore the expected number of episodes is $\Omega(2^N) = \Omega(2^S)$. □

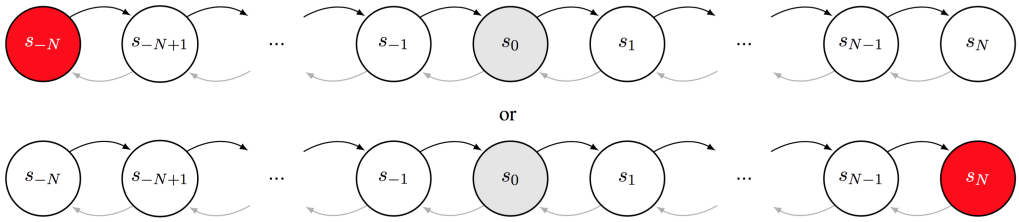


Figure 4.1: The possible MDPs for Example 4.1

Thompson sampling generates exploratory policies which are directed, but the exploration is not consistent across multiple timesteps. Several authors have proposed Thompson sampling for RL (Dearden et al., 1998; Gal and Ghahramani, 2015; Stadie et al., 2015). Further, Thompson sampling is asymptotically optimal in a wide range of environments (Leike et al., 2016), the problem is that this lack of deep exploration can lead to exponentially inefficient learning. That said, a relatively simple modification to this algorithm can adapt the probability matching principle to reinforcement learning much more effectively. Posterior sampling for reinforcement learning (PSRL) accomplishes exactly this, instead of resampling a policy every timestep the algorithm samples a single policy every *episode* and follows this policy for the duration of the episode. This adapts the Thompson sampling heuristic to allow for deep exploration.

Algorithm 4.2 Posterior sampling for reinforcement learning

Input: Prior distribution for MDPs ϕ.
for episode $k = 1, 2, ..$ **do**
  sample MDP $M_k \sim \phi(\cdot|\mathcal{H}_{k1})$
  compute $\mu_k \in \arg\max_{\mu} V^{M_k}_{\mu,h}$
  for time $h = 1, 2, .., H$ **do**
    take action $a_{kh} = \mu_k(s_{kh}, h)$
    observe r_{kh} and s_{kh+1}
    update $\mathcal{H}_{kh} = \mathcal{H}_{kh} \cup (a_{kh}, r_{kh}, s_{kh+1})$
  end for
end for

PSRL was initially proposed by Strens under the name of Bayesian dynamic programming (Strens, 2000). This older name is quite misleading however, since the Bayes-optimal exploration policy is available by dynamic programming in the belief MDP (Guez et al., 2014). PSRL is neither a Bayes-optimal policy, nor does it approximate one directly. PSRL

takes a single sample from the posterior and follows the policy which is optimal for that sample for the duration of the episode. We present a simple illustration of how this algorithm deviates from Bayes-optimal in Example 4.2.

Example 4.2 (PSRL does not approach Bayes-optimal)**.**
We consider a situation similar to Example 4.1 but where the start state is s_{-N+1} *and the episode length* $H > 2N + 1$*. We illustrate this environment in Figure 4.2. The maximally efficient exploration strategy would first head left to investigate* s_{-N} *in the first step. At this point the agent has full knowledge of its environment and can follow the optimal policy thereafter. The optimal exploration policy has regret uniformly bounded by* 2 *over all timesteps, whereas PSRL will incur expected regret* $\Omega(H)$*.*

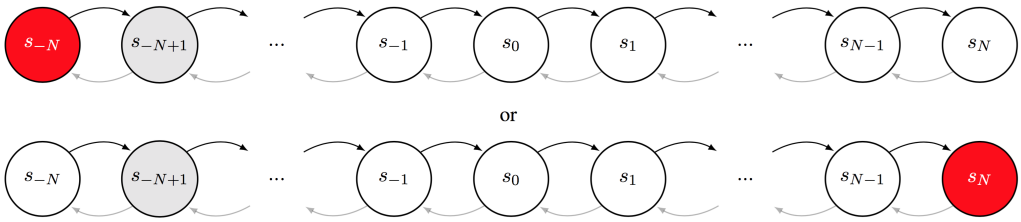


Figure 4.2: The possible MDPs for Example 4.2

PSRL is one of the simplest possible approaches to reinforcement learning. All it requires is that the agent can take a sample from its posterior belief and, that if it did know the true underlying system it could compute the optimal policy. This algorithm is computationally efficient, since it requires exactly the same computation as solving a single known MDP. Surprisingly, we will demonstrate that this simple heuristic is also statistically efficient.

4.2 (Even more) efficient reinforcement learning

In this section we will demonstrate that PSRL satisfies strong bounds statistical guarantees in terms of the Bayesian expected regret. First, we show that any high probability regret bound for *any* optimistic algorithm in the style of Algorithm 3.1 will imply that PSRL satisfies this same bound in expectation over the prior. This means that PSRL matches the state of the art regret scaling, given UCRL in Theorem 3.1, but also that if ever an improved variant of UCRL were developed these bounds would also apply to PSRL as well. This equivalence is taken from (Osband et al., 2013) and mirrors the same observation in

the bandit setting (Russo and Van Roy, 2014); it a profound result that connects Thompson sampling to any well-designed optimistic algorithm. Second, we present a new and improved analysis of PSRL guided by stochastic optimism and, in particular, the results of Lemma 3.3. This analysis demonstrates that PSRL is the first algorithm to satisfy $\tilde{O}(\sqrt{SAT})$ bounds for regret in reinforcement learning (Osband and Van Roy, 2016c) These match the worst case frequentist lower bounds up to logarithmic factors (Jaksch et al., 2010).

Before we progress any further it is worth highlighting that the regret bounds we present in this section will hold in expectation over the prior distribution. This measure is sometimes called the Bayes risk or Bayesian regret (Lehmann et al., 1986, 1991). We feel that this is a natural measure of performance, but should mention that many in the literature focus upon bounding regret for the worst-case MDP with high probability. These two criteria are intimately linked and bounds on the Bayesian regret bound the frequentist regret for any family of MDPs with nonzero probability under the prior (Osband et al., 2013). That said, bounds on the expected regret are in some sense weaker than high probability bounds in the worst case. We present a more detailed discussion of the relationship between the Bayesian and frequentist guarantees in Section 4.4.1.

4.2.1 PSRL matches the bounds for any optimistic algorithm

We will now present Theorem 4.1 that guarantees that relates the expected regret of PSRL to the optimal high probability bounds for any optimistic algorithm. This result establishes that PSRL is in some sense equivalent to an well-tuned optimistic algorithm; this is significant for several reasons. First, it means that PSRL matches the state of the art regret scalings established for any UCRL variant by Theorem 3.1. Further, and unlike typical optimistic algorithms, the performance of PSRL is separated from the analysis. This means that any future improvements to of any UCRL variants (Maillard et al., 2014; Dann and Brunskill, 2015) immediately apply to PSRL without need for any modification to the actual algorithm.

Optimistic algorithms need good analysis before they can get good performance. By contrast, PSRL works well regardless of what analysis you use. This means that PSRL attains similar statistical scaling to the optimal optimistic algorithm, even if we don't know how to design this optimistic algorithm. What is more, PSRL accomplishes this optimal

statistical efficiency at a computational cost no greater than solving a single known MDP, even if the optimal optimistic algorithm is computationally intractable. We formalize this relationship between any well-designed optimistic algorithm and the Bayesian regret of PSRL in Theorem 4.1.

Theorem 4.1 (PSRL matches any optimistic algorithm in Bayesian regret)**.**
Let π^{opt} be any optimistic algorithm for reinforcement learning in the style of Algorithm 3.1. If π^{opt} satisfies regret bounds such that, for any M^* as per Section 2.1, any $T > 0$ and any $\delta > 0$ the regret is bounded with probability at least $1 - \delta$

$$\text{Regret}(T, \pi^{\text{opt}}, M^*) \leq f(S, A, H, T, \delta). \tag{4.1}$$

Then, if ϕ is the distribution of the true MDP M^* (4.1) this result implies a bound on the expected regret for PSRL for all timesteps $T > 0$

$$\mathbb{E}\left[\text{Regret}(T, \pi^{\text{PSRL}}, M^*)\right] \leq 2f(S, A, H, T, \delta{=}T^{-1}) + 2. \tag{4.2}$$

In particular, we can choose $\pi^{\text{opt}} = \pi^{\text{UCRL}}$ in Theorem 4.1. In this case Theorem 3.1 implies Corollary 4.1 and that the bounds for PSRL acheive the scalings for UCRL in expectation. This result is the main contribution of the paper (Osband et al., 2013).

Corollary 4.1 (PSRL matches UCRL in Bayesian regret)**.**
If ϕ is the distribution of M^*, then for any $T > 0$

$$\mathbb{E}\left[\text{Regret}(T, \pi^{\text{PSRL}}, M^*)\right] = O\left(HS\sqrt{AT\log(SAT)}\right). \tag{4.3}$$

Proof of Theorem 4.1

Theorem 4.1 mirrors a similar result for multi-armed bandits (Russo and Van Roy, 2014). Before we begin this proof, we should also clearly emphasize that the bounds we obtain from Theorem 4.1 are on the Bayesian expected regret taken over the prior distribution; which is not the same as a frequentist worst-case guarantee for any MDP M^*. Nevertheless, this result is powerful and extremely general; what is more its proof is remarkably simple and elegant. The key step relies upon a delicate conditioning argument that, conditional upon *any* possible history of data $\mathcal{H}_k$ the sampled MDP M_k is equal in distribution to the

true MDP M^*.

This observation allows us to relate quantities that depend upon the true, but unknown MDP M^*, to those of the sampled MDP M_k, which is fully observed by the agent. We introduce $\sigma(\mathcal{H}_{k1})$ as the σ-algebra generated by the history up to episode k (Doob, 2012). Readers unfamiliar with measure theory can think of this as "all the information known just before the start of episode k". When we say that a random variable X is $\sigma(\mathcal{H}_{k1})$-measurable, intuitively this means that although X is random, it is deterministically known given the information contained in H_{k1}.

Lemma 4.1 (Posterior sampling)**.**
If ϕ is the distribution of M^* then for any $\sigma(\mathcal{H}_{k1})$-measurable function f,

$$\mathbb{E}\left[f(M_k)|\mathcal{H}_{k1}\right] = \mathbb{E}\left[f(M^*)|\mathcal{H}_{k1}\right]. \tag{4.4}$$

Lemma 4.1 allows us to relate the regret of any optimistic algorithm to PSRL. First we decompose the regret in episode k exactly as per (3.1). Now the imagined MDP M_k is sampled according to the posterior distribution for M^* conditional on the data $\mathcal{H}_{k1}$, rather than chosen optimistically. Just as before we decompose the regret,

$$\begin{aligned}
\Delta_k &= V^{M^*}_{\mu^{M_*},1} - V^{M^*}_{\mu_k,1} \\
&= V^*_{*,1} - V^*_{k,1} + V^k_{k,1} - V^k_{k,1} \\
&= \underbrace{V^k_{k,1} - V^*_{k,1}}_{\Delta_k^{\text{conc}}} + \underbrace{V^*_{*,1} - V^k_{k,1}}_{\Delta_k^{\text{opt}}}.
\end{aligned} \tag{4.5}$$

into the regret from optimism Δ_k^{opt} and the regret from concentration Δ_k^{conc}. Lemma 4.1 implies $\mathbb{E}[f(M_k)] = \mathbb{E}[f(M^*)]$ for all k through the tower property of conditional expectation and so $\mathbb{E}[\Delta_k^{\text{opt}}] = 0$ for all k. Our proof strategy will now conclude as per the optimistic analysis, with a concentration result for Δ_k^{conc}.

To compare PSRL with an arbitrary optimistic algorithm π^{opt} we will write that π^{opt} imagines an optimistic MDP $\tilde{M}_k$ within a confidence set $\mathcal{M}_k$ that contains the true MDP with high probability. We will use the tilde to represent policies and values for the optimistic algorithm such as in the natural way, for example $V^{\tilde{k}}_{\tilde{k},1}$ as the imagined optimistic value and $\tilde{\Delta}_k = V^{\tilde{k}}_{\tilde{k},1} - V^*_{\tilde{k},1}$ as the optimistic concentration. Optimistic algorithm conclude their proofs

by conditioning upon event that M^* lies within the confidence sets $\mathcal{M}_k$ at every episode, which holds with probability at least $1-\delta$. They then bound the regret by the absolute size of the concentration error (Jaksch et al., 2010; Bartlett and Tewari, 2009),

$$\left\{\text{Regret}(T,\pi^{\text{opt}},M^*) \mid M^* \in \mathcal{M}_k(\mathcal{H}_{k1}) \text{ for all } k\right\} \leq \sum_{k=1}^{\lceil T/H \rceil} |\tilde{\Delta}_k^{\text{conc}}| \leq f(S,A,H,T,\delta). \quad (4.6)$$

We can similarly bound regret from PSRL by the concentration error in PSRL. We then add and subtract the *imagined* optimal reward for the optimistic algorithm π^{opt}. We can mirror the analysis in (4.6), conditioned now upon the event that *both* M^* and M_k lie within the confidence sets $\mathcal{M}_k$. Lemma 4.1 means this holds with probability at least $1-2\delta$. Even when the confidence sets fail the regret is bounded by T.

$$\begin{aligned}
\mathbb{E}\Big[\text{Regret}(T,\pi^{\text{PSRL}},M^*)\Big] &\leq \mathbb{E}\left[\sum_{k=1}^{\lceil T/H \rceil} \mathbb{E}[\Delta_k^{\text{conc}} \mid \mathcal{H}_{k1}]\right] \\
&\leq \mathbb{E}\left[\sum_{k=1}^{\lceil T/H \rceil} \mathbb{E}\left[|V_{k,1}^k - V_{\tilde{k},1}^{\tilde{k}}| + |V_{\tilde{k},1}^{\tilde{k}} - V_{k,1}^*| \;\middle|\; \mathcal{H}_{k1}\right]\right] \\
&\leq \mathbb{E}\left[\sum_{k=1}^{\lceil T/H \rceil} \mathbb{E}\left[2|\tilde{\Delta}_k^{\text{conc}}| \;\middle|\; M^*, M_k \in \mathcal{M}_k(\mathcal{H}_{k1})\right]\right] + 2\delta T \quad (4.7)
\end{aligned}$$

This completes the proof of Theorem 4.1. □

4.2.2 PSRL satisfies $\tilde{O}(\sqrt{SAT})$ regret bounds

Theorem 4.1 establishes that PSRL matches the $O(\cdot)$ scalings for any optimistic algorithm. The crucial ingredient to this proof is Lemma 4.1 which requires that the sampled MDP is taken from the exact posterior distribution for $M^*|\mathcal{H}_{k1}$. In this section we will present an improved analysis for PSRL that is significant in several ways.

First we analyze PSRL without appeal to artificial confidence sets generated by any optimistic algorithm. This allows us to establish Theorem 4.2 and the first $O(\sqrt{SAT})$ regret bounds for any computationally tractable reinforcement learning algorithm. These match the worst case frequentist lower bounds in these scalings up to logarithmic factors. In addition to these bounds we establish that it is *not* necessary that the sampled MDP M_k is drawn from the exact posterior distribution. The key requirement is that the sampled

value function $V^k_{k,1}$ is sampled from a distribution which is is *stochastically optimistic* for true value function $V^*_{*,1}$, rather than the precise posterior distribution. The flavor of this result mirrors some of the bounds for Thompson sampling in bandits in the Frequentist perspective (Agrawal and Goyal, 2012a,b). Our work represents some of the first steps towards a robustness result for PSRL with a misspecified prior. Extending these results to more general frequentist guarantees in an area of ongoing research.

Theorem 4.2 (PSRL satisfies $\tilde{O}\left(\sqrt{SAT}\right)$ regret bounds)**.**
Let ϕ be the distribution of the underlying MDP M^*. For any ϕ with an independent Dirichlet prior over transitions the Bayesian expected regret of PSRL is bounded,

$$\mathbb{E}\left[\text{Regret}(T, \pi^{\text{PSRL}}, M^*)\right] = \tilde{O}\left(H\sqrt{SAT}\right). \tag{4.8}$$

Proof of Theorem 4.2

Our analysis begins with the familiar strategy of optimistic algorithms where we add and subtract the optimal value function of the *imagined* MDP M_k. We can apply Lemma 4.1 together with the the Bellman decomposition from Equation 3.10 to write the expected regret from concentration for PSRL

$$\mathbb{E}\left[\Delta^{\text{conc}}_k \;\middle|\; \mathcal{H}_{k1}\right] = \mathbb{E}\left[\sum_{h=1}^{H} \left(\overline{r}_k - \overline{r}^*\right)(x_{kh}) + \sum_{h=1}^{H} \left((P_k - P^*)(x_{kh})\right)^T V^k_{k,h+1} \;\middle|\; \mathcal{H}_{k1}\right] \tag{4.9}$$

Crucially, these expressions only depend upon the states and actions x_{kh} which are *actually* chosen by PSRL in episode k.

We now introduce some notation for the posterior mean estimate at the start of episode k. Let $\hat{r}_k(x) := \mathbb{E}\left[\overline{r}^*(x) \mid \mathcal{H}_{k1}\right]$ and also $\hat{P}_k(x) := \mathbb{E}\left[P^*(x) \mid \mathcal{H}_{k1}\right]$. We note that, conditional on the data at the start of episode k, the true MDP M^* and the sampled MDP M_k are independent and identically distributed. Therefore $\mathbb{E}[\overline{r}^*(x)|\mathcal{H}_{k1}] = \hat{r}_k(x)$ and $\mathbb{E}[P^*(x)|\mathcal{H}_{k1}] = \hat{P}_k(x)$. By contrast the sampled reward and transition functions r_k, P_k will not be unbiased since the policy PSRL chooses is optimized with respect to these imagined samples. We introduce two new variables to represent this noise from sampling

$$w^R_k(x_{kh}) := \hat{r}_k(x_{kh}) - r_k(x_{kh}), \tag{4.10}$$

$$w^P_k(x_{kh}) := \left(\hat{P}_k(x_{kh}) - P_k(x_{kh})\right)^T V^k_{k,h+1}. \tag{4.11}$$

We can rewrite the concentration error (4.9) in terms of these variables,

$$\mathbb{E}\left[\Delta_k^{\text{conc}} \;\middle|\; \mathcal{H}_{k1}\right] = \mathbb{E}\left[\sum_{h=1}^{H}\left(w_k^R(x_{kh}) + w_k^P(x_{kh})\right) \;\middle|\; \mathcal{H}_{k1}\right]. \tag{4.12}$$

We can complete the regret bound by controlling this deviation from sampling.

Controlling the contributions from rewards

We can bound the contribution from unknown rewards $w_k^R(x_{kh})$ with a standard argument from earlier work (Jaksch et al., 2010; Osband et al., 2013). We apply Lemma 3.2 to the errors of the empirical estimate for the mean reward, conditional upon the history $\mathcal{H}_{k1}$. By posterior sampling, the true rewards are identically distributed to the sampled rewards conditional on any data. We conclude via a triangle inequality to give Lemma 4.2.

Lemma 4.2 (Reward concentration)**.**
For any prior distribution ϕ over rewards in $[0,1]$, state-action pair x_{kh} and any $\delta > 0$ then conditional upon any history $\mathcal{H}_{k1}$:

$$|w^R(x_{kh})| \leq 2\sqrt{\frac{2\log(2/\delta)}{n_k(x_{kh})}} \tag{4.13}$$

with probability at least $1-\delta$.

Proof. Simple application of Lemma 3.2. □

Controlling the contributions from transitions

The key difficulty in reinforcement learning comes from the contributions of unknown transitions $w_k^P(x_{kh})$. Typical optimistic algorithms, such as UCRL (Jaksch et al., 2010), use a Hölder inequality

$$|w_k^P(x_{kh})| \leq \|\hat{P}_k(x_{kh}) - P_k(x_{kh})\|_1 \|V_{k,h+1}^k\|_\infty. \tag{4.14}$$

They then bound the L1 deviation in the estimate of P_k and an upper bound on the span of the imagined optimal value via Lemma 3.1 (Bartlett and Tewari, 2009; Jaksch et al., 2010). In Section 3.2 we discuss two important shortcomings for this approach. First, the L1 metric

can be inefficient for distributional concentration. Several papers have addressed this issue either by higher order conditions (Dann and Brunskill, 2015), KL-divergence (Filippi et al., 2010) or dual norm decomposition specialized to MDPs (Maillard et al., 2014). However, there is a larger issue that the confidence sets at each (s, a) are independent rather than coupled across the MDP. As a result, in order to guarantee a good estimate of $P^T V \in \mathbb{R}$, these algorithms require a good estimate of $P \in \mathbb{R}^S$ and a good estimate of $V \in \mathbb{R}^S$.

The key to our analysis for Theorem 4.2 comes from Lemma 4.3 which show that the estimates for PSRL concentrate at a rate independent of S. The intuition for this result is very similar to Lemma 3.3, which we establish in Section 3.4, we compare the concentration with an optimistic Gaussian approximation. The only difficulty in extending these results is that unlike Lemma 3.3, the future value V^k_{kh+1} may itself be a random variable whose value is not independent of the sampled transition $P_k(x_{kh})$. Our strategy will be to show that, although V^k_{kh+1} can vary with P_k, the structure of the MDP means that resultant $w^P(x_{kh})$ is still no more stochastically optimistic than the most optimistic possible *fixed* $V \in [0, H]^S$.

Lemma 4.3 (Transition concentration)**.**
For any independent prior over rewards with $\overline{r} \in [0, 1]$ and sub Gaussian noise and independent Dirichlet posterior over transitions at state-action pair x, then

$$w^P_h(x_{kh}) \leq 2H\sqrt{\frac{2\log(2/\delta)}{\max(n_k(x_{kh}) - 2, 1)}} \tag{4.15}$$

with probability at least $1 - \delta$.

Proof. We begin this proof only for the simply family of MDPs with $S = 2$, which we call $\mathcal{M}_2$. We write $p := P_k(x_{kh})(1)$ for the first component of the unknown transition at x_{kh} and similarly $\hat{p} := \hat{P}_k(x_{kh})(1)$. We can then bound the transition concentration,

$$\begin{aligned}
|w^P_h(x_{kh})| &= |(P_k(x_{kh}) - \hat{P}_k(x_{kh}))^T V^k_{kh+1}| \\
&= |(p - \hat{p})(V^k_{kh+1}(1) - V^k_{kh+1}(2))| \\
&\leq |(p - \hat{p})||(V^k_{kh+1}(1) - V^k_{kh+1}(2))| \\
&\leq |p - \hat{p}| \sup_{R_k, P_k} |(V^k_{kh+1}(1) - V^k_{kh+1}(2))| \\
&\leq |(p - \hat{p})|H
\end{aligned} \tag{4.16}$$

The results of Lemma 3.6 imply that for any $\alpha \in \mathbb{R}_+$ with with $\alpha^T \mathbb{1} \geq 2$, the the random variables $p \sim \text{Beta}(\alpha)$ and $X \sim N(0, \sigma^2 = 1/\alpha^T \mathbb{1})$ are ordered,

$$X \succcurlyeq_{\text{so}} p - \hat{p} \implies |X|H \succcurlyeq_{\text{so}} |p - \hat{p}|H \succcurlyeq_{\text{so}} |w_h^P(x_{kh}|. \tag{4.17}$$

We can conclude the proof for $\mathcal{M}_2$ through a concentration result for the optimistic Gaussian approximation $H|X|$ and Lemma 3.2.

The remaining task to prove Lemma 4.3 is to extend this argument to multiple states $S > 2$. To do this, we use a similar argument to Lemma 3.5. Consider $|w_h^P(x_{kh})| = |(P_k(x_{kh}) - \hat{P}_k(x_{kh}))^T V_{kh+1}^k|$. Imagine that, for any fixed $P_k(x_{kh})$ we then allow R_k and $P_k(x' \neq x)$ to vary to maximize $|(P_k(x_{kh}) - \hat{P}_k(x_{kh}))^T V_{kh+1}^k|$ through the resulting V_{kh+1}^k. For any such V_{kh+1}^k, the resulting random variable would be stochastically dominated by a matching Beta distribution by Lemma 3.5. In turn, since we know that $V_{kh+1}^k \in [0, H]^S$ we can bound this term by the results of (4.16). This completes the proof of Lemma 4.3. □

Completing the regret bound

We now combine Lemma 4.2 with Lemma 4.3 to bound the regret from concentration in PSRL. We rescale $\delta \leftarrow \delta/2SAT$ so that these confidence sets hold at each $R(s,a), P(s,a)$ in union bound

$$\mathbb{E}\left[\sum_{h=1}^H \left\{|w^R(x_{kh})| + |w_h^P(x_{kh})|\right\} \mid \mathcal{H}_{k1}\right] \leq \sum_{h=1}^H 2\,(H+1)\sqrt{\frac{2\log(2SAT)}{\max(n_k(x_{kh})-2,1)}}, \tag{4.18}$$

with probability at least $1 - \frac{1}{T}$.

We can now use (4.18) to complete our proof of Theorem 4.2 using a pigeonhole principle over the worst-case number of visits to each state and action:

$$\begin{aligned}\mathbb{E}\left[\text{Regret}(T, \pi^{\text{PSRL}}, M^*)\right] &\leq \sum_{k=1}^{\lceil T/H \rceil} \sum_{h=1}^H 2(H+1)\sqrt{\frac{2\log(2SAT)}{n_k(x_{kh})}} + 2SA + 1 \\ &\leq 10H\sqrt{SAT\log(2SAT)}.\end{aligned}$$

This completes the proof of Theorem 4.2 □

4.2.3 Towards an $\tilde{O}(\sqrt{HSAT})$ regret bound

The analysis of Theorem 4.2 provides the first Bayesian regret bounds $\tilde{O}(H\sqrt{SAT})$ for any algorithm. In fact, we believe that it is possible to improve this analysis to reduce the dependence of H to $\sqrt{H}$ following a line of argument similar to recent PAC analyses (Lattimore and Hutter, 2012; Dann and Brunskill, 2015). They note that "local value variance" satisfies a Bellman equation. Intuitively this captures that if we transition to a bad state $V \simeq 0$, then we cannot transition anywhere much worse during this episode. This relation means that $\sum_{h=1}^{H} w_h^P(x_{kh})$ should behave more as if they were independent and grow $O(\sqrt{H})$, unlike our analysis which crudely upper bounds them each in turn $O(H)$. We present this hypothesized bound as Conjecture 4.1 together with a rough sketch of a potential analysis.

Conjecture 4.1. *For any prior over rewards with $\overline{r} \in [0,1]$ and sub Gaussian noise and any independent Dirichlet prior over transitions, we conjecture that*

$$\mathbb{E}\left[\text{Regret}(T, \pi^{\text{PSRL}}, M^*)\right] = \tilde{O}\left(\sqrt{HSAT}\right), \tag{4.19}$$

and that this matches the lower bounds for any algorithm up to logarithmic factors.

Proof sketch. We follow the proof of Theorem 4.2 up to the bound for $\sum_{h=1}^{H} w_h^P(x_{kh})$. The argument in Theorem 4.2 bounds each $w_h^P(x_{kh})$ independently. Each term is $\tilde{O}\left(\sqrt{\frac{H}{n_k(x_{kh})}}\right)$ to give a resulting sum $\tilde{O}\left(H\sqrt{\frac{H}{n_k(x_{kh})}}\right)$. However, this approach is very loose. In particular, to repeat our geometric intuition, we have assumed a worst-case hyper-rectangle over all timesteps H when the actual geometry should be an ellipse.

It is not possible to sequentially get the "worst-case" transitions $O(H)$ at each and every timestep during an episode, since once your sample gets one such transition then there will be no more future value to deplete. A very similar observation is used by recent analyses in the sample complexity setting (Lattimore and Hutter, 2012) and also finite horizon MDPs (Dann and Brunskill, 2015). This seems to suggest that it should be possible to combine the insights of Lemma 3.3 with, for example, Lemma 4 of (Dann and Brunskill, 2015) to remove *both* the $\sqrt{S}$ *and* the $\sqrt{H}$ from our bounds to prove Conjecture 4.1.

□

Optimistic posterior sampling via Gaussian PSRL

To elucidate the reasoning begin Conjecture 4.1 we note that the current proof of Theorem 4.2 does not require that PSRL samples from the exact posterior. In fact, the only important argument is that the sampled value function V_{k1}^k is stochastically optimistic for the true value function. We then bound the regret from concentration Δ_k^{conc} by comparison to a value function perturbed by Gaussian noise, rather than the Dirichlet posterior samples used by PSRL. We now formalize this auxiliary algorithm below and call it Gaussian PSRL[1].

Algorithm 4.3 Gaussian PSRL

Input: Optimistic sampling procedure GaussVal of Algorithm 4.4
for episode $k = 1, 2, ..$ **do**
  sample Gaussian value $Q_k \sim \text{GaussVal}(\mathcal{H}_{k1})$
  for time $h = 1, 2, .., H$ **do**
    take action $a_{kh} \in \arg\max_a Q_k(s_{kh}, a, h)$
    observe r_{kh} and s_{kh+1}
    update $\mathcal{H}_{kh} = \mathcal{H}_{kh} \cup (a_{kh}, r_{kh}, s_{kh+1})$
  end for
end for

Algorithm 4.4 Optimistic value sampling for Gaussian PSRL

Input: History of past observation $\mathcal{H}_{k1}$.
initialize $Q_k(s, a, H+1) \leftarrow 0$ for each $s \in \mathcal{S}, a \in \mathcal{A}$
for timestep $h = H, H-1, .., 1$ **do**
  compute future value $V_{h+1}(s) = \max_\alpha Q_k(s, \alpha, h+1)$ for each $s \in \mathcal{S}$
  for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ **do**
    compute empirical estimates $\hat{P}_k(s, a, h), \hat{r}_k(s, a, h)$ and counts $n_k(s, a, h)$
    sample transition noise $w_k^P(s, a, h) \sim N\left(\mu{=}0, \sigma^2{=}\frac{H^2}{\max(n_k(s,a,h),1)}\right)$
    sample reward noise $w_k^R(s, a, h) \sim N\left(\mu{=}0, \sigma^2{=}\frac{1}{\max(n_k(s,a,h),1)}\right)$
    update $Q_k(s, a, h) \leftarrow \hat{r}_k(s, a, h) + w_k^R(s, a, h) + \hat{P}_k(s, a, h)^T V_{h+1} + w_k^P(s, a, h)$
  end for
end for
Return: $Q_k \in \mathbb{R}^{S \times A \times H}$ sampled value estimate

[1]Another interesting perspective on Gaussian PSRL is that the algorithm is equivalent to a greedy policy acting on historical batch data $\mathcal{H}_k$ whose observed rewards have been perturbed by i.i.d. $N(0, H^2)$ noise. In fact, we will revisit such an algorithm at a high level in Part III of this disseration as a practical approach to randomized value functions with nonlinear generalization.

Gaussian PSRL explicitly injects the matched Gaussian noise which we use to prove the regret bounds for PSRL in Theorem 4.2. However, we note that unlike PSRL, the arguments for Conjecture 4.1 would *not* apply to Gaussian PSRL. Interestingly, and as we will demonstrate in the next section, we find that our experimental evaluation is consistent with $\tilde{O}(HS\sqrt{AT})$, $\tilde{O}(H\sqrt{SAT})$ and $\tilde{O}(\sqrt{HSAT})$ for UCRL2, Gaussian PSRL and PSRL respectively. These results help to suggest that it might be possible to formalize the arguments in our conjecture and maybe even in terms of the problem-specific frequentist regret (Gopalan and Mannor, 2014).

4.3 Experimental evaluation

In this section we investigate the empirical performance of PSRL on a family of didactic environments designed to highlight the need for directed and deep exploration. In particular, we will pay special attention to the comparison between PSRL and algorithms based upon the "optimism in the face of uncertainty".

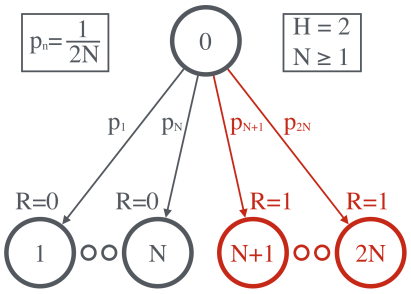
4.3.1 Shortcomings of existing optimistic algorithms

We begin with a computational illustration that the confidence sets used in common OFU-based algorithms lead to extremely inefficient estimates as S or H grow. This is an empirical validation of the argument in Section 3.2. We highlight these failures via some trivial MDPs with only one action which we describe in Figure 3.1, and which we reproduce below. In each of these environments the true value from the initial state 0 is $Q(0,1) = \frac{1}{2}(H-1)$. The simplicity of the examples means it is easy to see how to design more effective confidence sets in these contexts. Nonetheless, we will see that confidence sets suggested by OFU-RL can become incredibly mis-calibrated as H and S grow.

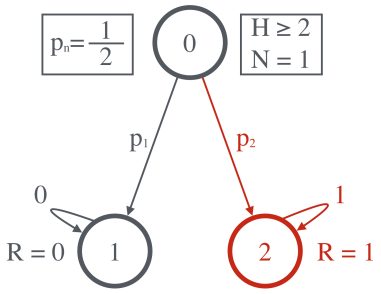
In each experiment we sample $K = 1000$ episodes of data from the MDP and then examine the optimistic/sampled Q-values for UCRL2 and PSRL. We implement a version of UCRL2 optimized for finite horizon MDPs and implement PSRL with a uniform Dirichlet prior over the initial dynamics $P(0,1) = (p_1, .., p_{2N})$ and a $N(0,1)$ prior over rewards updating as if rewards had $N(0,1)$ noise. For both algorithms, if we say that R or P are *known* then we mean that we use the true R or P inside this part of UCRL2 or PSRL. We present three simple experiments:

- Figure 4.3a shows the effect of increasing N in Figure 3.1a on an agent with full reward knowledge but unknown transitions.
- Figure 4.3b shows the effect of increasing N in Figure 3.1a on an agent with full transition knowledge but unknown rewards.
- Figure 4.3c shows the effect of increasing H in Figure 3.1b on an agent with unknown rewards and transitions.

We compare the estimates of PSRL to a version of UCRL2 modified for finite horizon settings. In each experiment, the estimates guided by OFU become extremely mis-calibrated, even for relatively small values of S, H; PSRL remains stable.

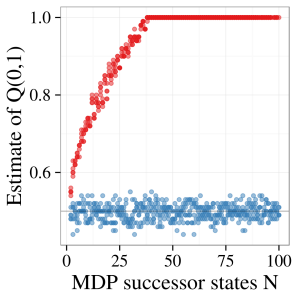


(a) MDPs to illustrate the scaling with S.

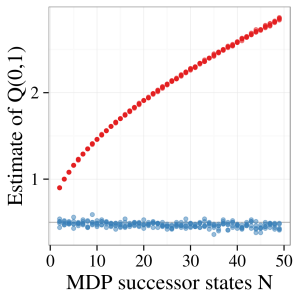


(b) MDPs to illustrate the scaling with H.

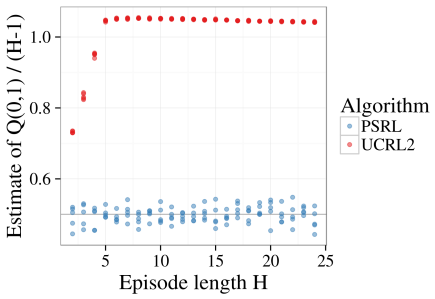
Figure 3.1: Simple environments designed to highlight potential problems with OFU-RL.



(a) R known, P unknown. Vary N in Figure 3.1a.



(b) P known, R unknown. Vary N in Figure 3.1a.



(c) R, P unknown. Vary H in 3.1b

Figure 4.3: UCRL2 leads to miscalibrated optimism, PSRL does not.

The results of Figure 4.3b are particularly revealing. They demonstrates the potential pitfalls of OFU-RL even when the underlying transition dynamics *entirely known*. Several OFU algorithms have been proposed to remedy the loose UCRL-style L1 concentration from transitions (Filippi et al., 2010; Araya et al., 2012; Dann and Brunskill, 2015) but none of these address the inefficiency from hyper-rectangular confidence sets. These loose confidence sets lead to extremely poor performance when measured in regret. To investigate this effect in a genuine sequential decision setting we augment the MDP in Figure 3.1 but to now include two actions instead of one. The first action is identical to Figure 3.1, but the second action modifies the transition probabilities to favor the rewarding states with probability $0.6/N$ and assigning only $0.4/N$ to the non-rewarding states.

We now investigate the *regret* of several learning algorithms which we adapt to this setting. These algorithms are based upon BEB (Kolter and Ng, 2009), BOLT (Araya et al., 2012), ϵ-greedy with $\epsilon = 0.1$, Gaussian PSRL (Algorithm 4.3), Optimistic PSRL (which takes $K = 10$ samples and takes the maximum over sampled Q-values similar to BOSS (Asmuth et al., 2009)), PSRL (Strens, 2000), UCFH (Dann and Brunskill, 2015) and UCRL2 (Jaksch et al., 2010).

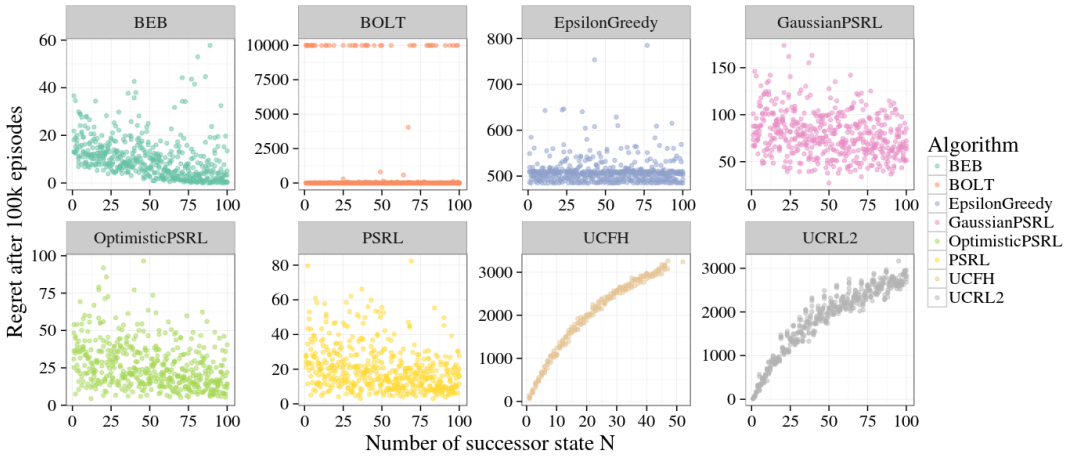


Figure 4.4: Regret with known rewards R and unknown transitions P, similar to Figure 4.3a.

We see that the loose estimates in OFU algorithms from Figure 4.3 lead to bad performance in a decision problem. This poor scaling with the number of successor states N occurs when *either* the rewards or the transition function is unknown. We note that in stochastic environments the PAC-Bayes algorithm BOLT, which relies upon optimistic fake

prior data, can sometimes concentrate too quickly and so incur the maximum linear regret. In general, although BOLT is PAC-Bayes, it concentrates too fast to be PAC-MDP just like the algorithmBEB (Kolter and Ng, 2009).

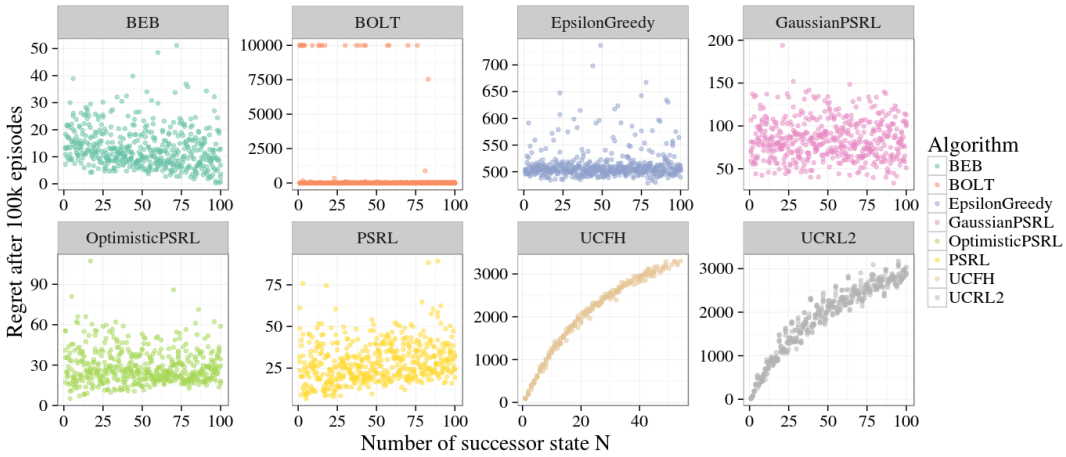


Figure 4.5: Regret with unknown rewards R and known transitions P, similar to Figure 4.3b.

In Figure 4.6 we see a similar effect as we increase the episode length H. We note the second order UCFH modification improves upon UCRL2's miscalibration with H, as is reflected in their bounds (Dann and Brunskill, 2015). We note that both BEB and BOLT scale poorly with the horizon H.

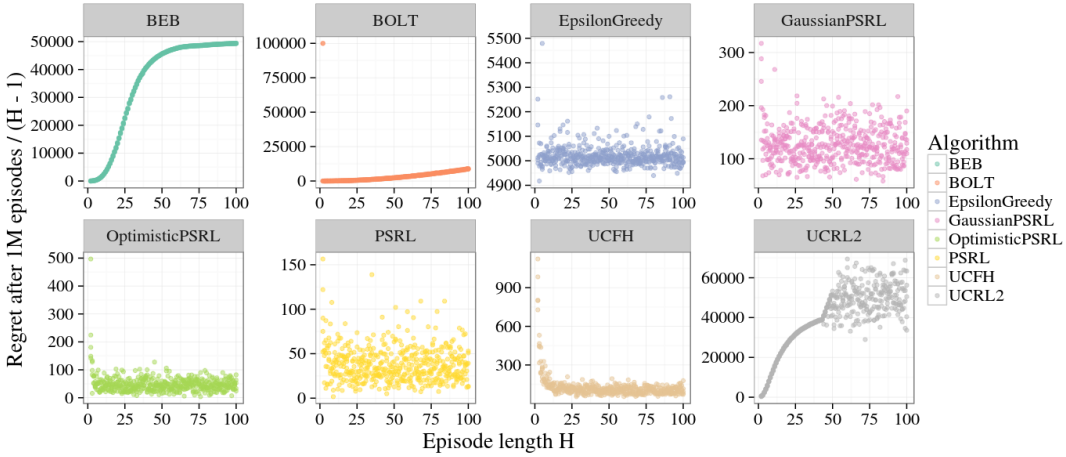


Figure 4.6: Regret with unknown rewards R and transitions P, similar to Figure 4.3c.

4.3.2 Experiments for deep exploration

We now consider an experiment designed to illustrate the importance of deep exploration. This example will highlight the joint scaling in H and S and lends some support the conjecture that PSRL reduces regret by a factor of $\sqrt{H}$ relative to UCRL2. The class of MDPs we consider involves a long chain of states with $S = H = N$ and with two actions: left and right. Each episode the agent begins in state 1. The optimal policy is to head right at every timestep, all other policies have zero expected reward. Inefficient strategies for exploration will take $\Omega(2^N)$ episodes to learn the optimal policy per Example 2.1.

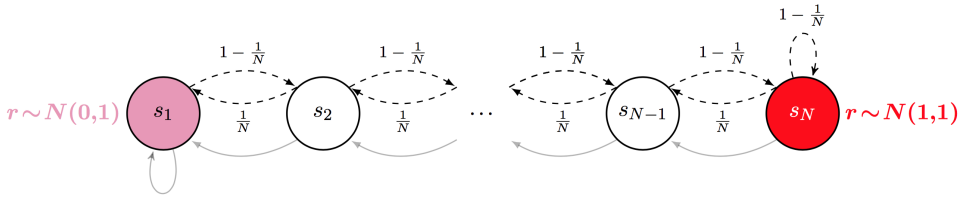


Figure 4.7: A family of MDPs designed to highlight the need for efficient exploration.

We evaluate several learning algorithms from five random seeds and $N = 2, .., 100$ for one million episodes each. Our goal is to investigate their empirical performance and scaling. We highlight results for three algorithms with $\tilde{O}(\sqrt{T})$ regret bounds: UCRL2, Gaussian PSRL and PSRL. All of the code and experiments used in this paper are available in full at https://github.com/iosband/TabulaRL. We implement UCRL2 with confidence sets optimized for finite horizon MDPs. For the Bayesian algorithms we use a uniform Dirichlet prior for transitions and $N(0, 1)$ prior for rewards.

Figure 4.8 display the regret curves for these algorithms for $N \in \{5, 10, 30, 50\}$. As suggested by our analysis, PSRL outperforms Gaussian PSRL which itself outperforms UCRL2. These differences seems to scale with the length of the chain N and that even for relatively small MDPs, PSRL is many orders of magnitude more efficient than UCRL2.

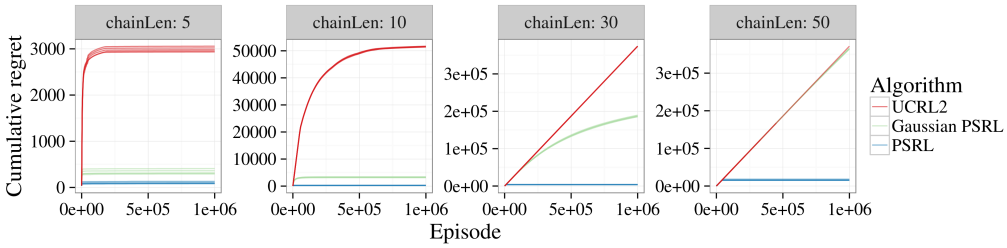


Figure 4.8: PSRL outperforms other methods by large margins.

We investigate the empirical scaling of these algorithms with respect to N and compare these results to our analysis. Suppose for some $B \in \mathbb{R}$ we can bound $\text{Regret}(T) \leq \sqrt{BT}$). This means we that $\tilde{T}(\epsilon)$:= "*#episodes* until the average regret per episode $\leq \epsilon$"= $\tilde{O}(B^2 H/\epsilon^2)$. The results of Theorem 4.2 and Conjecture 4.1 would suggest that up to $o(\log(N))$:

$$\log(\tilde{T}^{\text{UCRL2}}) \simeq 5\log(N),\ \log(\tilde{T}^{\text{GaussPSRL}}) \simeq 4\log(N),\ \log(\tilde{T}^{\text{PSRL}}) \simeq 3\log(N).$$

Figure 4.9 displays $\tilde{T}(0.1)$ for UCRL2, Gaussian PSRL and PSRL. Along side these results we also draw dotted lines with slope $5, 4$ and 3 respectively, which are the scalings associated with bounds $\tilde{O}(HS\sqrt{AT})$, $\tilde{O}(H\sqrt{SAT})$ and $\tilde{O}(\sqrt{HSAT})$ respectively. We find that the empirical results match the results of Theorem 4.2 and provide support to Conjecture 4.1. PSRL is robust over several orders of magnitude of prior specification. Full details of these computational results together with robustness over prior parameters are available in Appendix D of (Osband and Van Roy, 2016c).

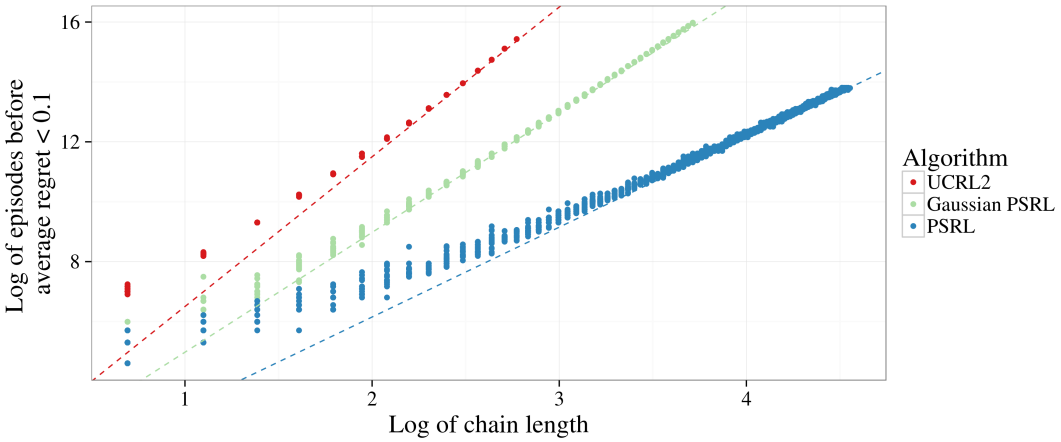


Figure 4.9: Empirical scaling matches results suggested by our analysis.

PSRL is orders of magnitude more statistically efficient than UCRL *and* S-times less computationally expensive. In the future, we believe that analysts will be able to formally specify an OFU approach to RL that remedies the issues of Section 3.2 so that its statistical efficiency matches PSRL. However, we argue that the resulting confidence sets which address both the coupling over H and S may lead to a computationally intractable optimization problem. For this reason, computationally efficient approaches to optimism may sacrifice statistical efficiency; this is why posterior sampling is better than optimism for reinforcement learning.

4.4 Relating PSRL to existing work

In this chapter we have proposed posterior sampling for reinforcement learning as an alternative driving principle to deterministic optimism in the face of uncertainty. We present some insight to why existing optimistic approaches to RL can be inefficient and demonstrate the PSRL does not suffer from these same flaws. That said, the analytical results which we derive in Theorem 4.1 and Theorem 4.2 are not exactly the same type as those popular in the literature for optimistic RL algorithms. In this section we will highlight these differences in analysis and point towards interesting avenues for future research in this area.

4.4.1 Bayesian regret and frequentist guarantees

Theorems 4.1 and 4.2 establish state of the art bounds for PSRL in terms of the expected regret over a prior distribution of MDPs. These results are extremely general, and apply to an immense class of potential models. To accommodate this generality, our bounds hold in expectation under the prior distribution, which is sometimes referred to as the *Bayes risk* or *Bayesian regret*. In particular, these results are not the same as the frequentist worst-case bounds on regret that are satisfied by UCRL2 (Jaksch et al., 2010). Typically, papers in the literature focus on either the Bayesian or the frequentist regret as if the two criteria were entirely separate considerations. Our next result (Russo and Van Roy, 2014; Osband et al., 2013) provides some link between these criteria.

Theorem 4.3 (Bounds on expected regret imply bounds in high probability)**.**
Let $\mathcal{M}$ be any family of MDPs such that $\mathbb{P}(M^* \in \mathcal{M}) > 0$. Suppose that π is any learning algorithm such that the expected regret is bounded,

$$\mathbb{E}\left[\text{Regret}(T, \pi, M^*) \;\middle|\; M^* \sim \phi\right] \leq D\sqrt{T},$$

for some $D > 0$ and all $T > 0$. Then, for any $\epsilon > 0$ and any $\alpha > \frac{1}{2}$,

$$\mathbb{P}\left(\frac{\text{Regret}(T, \pi, M^*)}{T^\alpha} > \epsilon \mid M^* \in \mathcal{M}\right) \to 0 \quad \text{as } T \to \infty. \tag{4.20}$$

Proof. For any $\epsilon > 0$ we can use Markov's inequality to bound

$$\begin{aligned}\frac{\mathbb{E}\left[\text{Regret}(T,\pi,M^*)\right]}{T^\alpha} &\geq \frac{\mathbb{E}\left[\text{Regret}(T,\pi,M^*) \mid M^* \in \mathcal{M}\right]}{T^\alpha}\mathbb{P}(M^* \in \mathcal{M}) \\ &\geq \epsilon\mathbb{P}\left(\frac{\text{Regret}(T,\pi,M^*)}{T^\alpha} > \epsilon \mid M^* \in \mathcal{M}\right)\mathbb{P}(M^* \in \mathcal{M}) \quad (4.21)\end{aligned}$$

We can then rearrange this expression to see that,

$$\mathbb{P}\left(\frac{\text{Regret}(T,\pi,M^*)}{T^\alpha} > \epsilon \mid M^* \in \mathcal{M}\right) \leq \frac{\mathbb{E}\left[\text{Regret}(T,\pi,M^*)\right]}{\epsilon\mathbb{P}(M^* \in \mathcal{M})T^\alpha} \to 0.$$

□

Theorem 4.3 shows that bounds on the Bayesian regret also imply bounds on the frequentist regret for any environment with non-zero probability under the prior. In addition, these bounds will maintain the optimal $\tilde{O}(\sqrt{T})$ convergence. For situations without strong prior knowledge, we suggest that an agent can encode this in the form of a diffuse prior for example uniform or Jeffrey's (Jeffreys, 1946). Unfortunately, this kind of diffuse prior typically place mass exponentially small in S, A around any MDP M^*. This means that the resultant bounds from Theorem 4.3 are much weaker than the corresponding guarantees for UCRL (Jaksch et al., 2010; Osband et al., 2013).

A similar analysis to Theorem 4.1 studies degenerate MDPs with $H = 1$ in terms of the Bayesian regret. At the same time, a more careful analysis also established *frequentist* bounds for the regret with certain choices of prior distribution (Agrawal and Goyal, 2012a; Kaufmann et al., 2012; Agrawal and Goyal, 2012b) that match the frequentist lower bounds. These proofs are long and involved, but we believe that it will be possible to extend a similar result to PSRL. Our remarkably robust simulation results lend some credence to this claim. To stimulate further thinking on this topic we present Conjecture 4.2 to illustrate an intermediate result which may help to establish frequentist regret bounds for PSRL.

Conjecture 4.2 (Posterior sampling and stochastic optimism)**.**
Let M^* be an MDP with $S = H = 1$ and mean rewards $\overline{r}^*(a) \sim \phi$. The realized rewards are drawn $r_t(a) = \overline{r}^*(a) + \epsilon_t(a)$, where $\mathbb{E}[\epsilon_t(a)|\overline{r}^*(a)] = 0$ and $Z \sim N(0,1) \succcurlyeq_{\text{so}} \epsilon_t(a)$ for all a. We write $\tilde{\pi}^{\text{PSRL}}$ for PSRL with with prior $\tilde{\phi} \succcurlyeq_{\text{so}} \phi$ and posterior updating as if errors $\epsilon_t(a) \sim N(0,1)$. We conjecture that the regret of this misspecified PSRL is bounded,

$$\mathbb{E}\left[\text{Regret}(T, \tilde{\pi}^{\text{PSRL}}, M^*)|M^* \sim \phi\right] \;\leq\; \mathbb{E}\left[\text{Regret}(T, \tilde{\pi}^{\text{PSRL}}, M^*)|M^* \sim \tilde{\phi}\right] \;=\; \tilde{O}\left(\sqrt{AT}\right). \tag{4.22}$$

This result says that we do not need to use the correct posterior for the M^* so long as we use an optimistic prior and don't update the noise too fast. Therefore using a diffuse optimistic prior would satisfy frequentist regret bounds.

Proof sketch. We begin with Proposition 3.2 for stochastically optimistic random variables. For any distribution of $\epsilon_t(a)$ the agent could inject additional noise $w_t(a)$ with $\mathbb{E}[w_t(a)|r_t(a)] = 0$ to produce $\tilde{r}_t(a) = r_t(a) + w_t(a) \sim N(\overline{r}^*(a), 1)$. If the algorithm $\tilde{\pi}^{\text{PSRL}}$ updated its posterior based upon $\tilde{r}_t(a)$, rather than the actual data $r_t(a)$ then this problem would be precisely equivalent to as if the true MDP were generated by the misspecified prior $\tilde{\phi}$. Using this corrupted rewards $\tilde{r}_t$ we can recover the bounds of Theorem 4.1.

The setting we want to investigate is where the algorithm uses this incorrect prior $\tilde{\phi}$ on the real environment ϕ. For any fixed number of samples from any arm a and any mean value $\overline{r}^*(a)$ the posterior estimates based upon $r_t(a)$ are expected to be closer to the truth than those based upon the corrupted $\tilde{r}_t(a)$. In both cases the Gaussian posterior uses the identical variance, so we believe that we should be better served by using the more accurate posterior distribution. The addition of un-informative zero-mean noise $w_t(a)$ makes the problem *harder* not easier.

Although this argument is intuitively appealing, completing these steps in a mathematically rigorous manner is difficult. Our proofs of Theorem 4.1 and 4.2 rely on optimistic samples given any arbitrary history $\mathcal{H}_{k1}$. For PSRL with mismatched likelihood distributions this will not generally be the case; the single exceptional case we have for this is the content of Lemma 3.3. The thing is, we actually do not need this result to hold for every possible dataset $\mathcal{H}_{k1}$ for the final result to be true, we only need it to hold in expectation

over the data chosen by $\tilde{\pi}^{\text{PSRL}}$ in M^*. We believe that techniques similar to the existing literature on frequentist bounds (Agrawal and Goyal, 2012a) may provide an avenue to this sort of result. We leave the analysis for this flavor of result in both bandits and reinforcement learning as an open problem for future research. □

4.4.2 On the horizon length and learning

So far in this dissertation we have focused upon environments which can be described as finite horizon MDPs as per section 2.1. At the same time, a great portion of the literature in reinforcement learning has focused upon the setting with infinite horizon and without episodic structure, as outlined in Section 2.1.1 (Sutton and Barto, 1998). We believe that the great majority of reinforcement learning problems of practical interest can be appropriately mapped to settings with finite episode length. For example, any system where the optimal policy has a mixing time less than H can be effectively modeled as an episodic MDP of horizon H while any agent that uses a discount factor $\gamma \in [0, 1)$ effectively operates on an episode length $H \simeq \frac{1}{1-\gamma}$ (Kakade, 2003). In general, the insights and analysis from the setting with finite episodes can extended to the infinite horizon setting with a little more careful analysis. However, there are a few delicate difficulties that arise in the infinite horizon setting.

The first thing to note is that, in general, it is not possible to design any algorithm which is asymptotically no-regret for any infinite horizon MDP. We present a simple MDP that highlights this problem in example 4.3.

Example 4.3 (Heaven and hell)**.**
We consider a simple MDP with three states $\mathcal{S} = \{s_0, s_1, s_2\}$ and two actions $\mathcal{A} = \{a_1, a_2\}$. The agent begins in s_0, if they choose action a_1 they transition to s_1 and if they choose a_2 they transition to s_2. The states s_1 and s_2 are absorbing, so that for all subsequent steps they remain here, regardless of action. If i^* is equally likely to be 1 or 2 there is no algorithm that can provide an expected regret less than $\frac{T}{2}$.

The PAC-MDP approach to sample complexity looks at the number of steps in which the policy takes an action whose expected discounted return from that action is more than ϵ sub-optimal (Kakade, 2003; Lattimore and Hutter, 2012; Dann and Brunskill, 2015). We find this formulation somewhat unsatisfying for two reasons. First, unlike regret bounds,

strong sample complexity bounds does not indicate strong performance. In fact, as we can see in Example 4.3 an algorithm can have only one ϵ-sub optimal action and zero total return even when the optimal policy would obtain T. Second, for many settings the imposition of the discount factor $\gamma \in [0, 1)$ is contrived and does not come from the actual environment. As such, PAC-MDP guarantees are somewhat disconnected from the natural RL problem.

Algorithms that bound the regret over infinite horizon problems must impose some connectedness constraint on the underlying MDP in order to discount environments like Example 4.3 from their consideration. The precise requirements used in each paper are slightly different, but a good review is given by (Bartlett and Tewari, 2009). The essence of all these conditions are that the MDP should be connected, so that it is always possible to recover from bad mistakes during learning. In this setting a similar quantity to the horizon H emerges, and represents the maximum scale of a mistake during learning. Most of the insights from our treatment of PSRL translate to the setting of infinite horizon in a natural way, although there are some delicate technical issues that arise during this analysis. Practical approaches to PSRL in MDPs without episodic reset may include:

- Assume some ergodic positive recurrent state (Gopalan and Mannor, 2014).
- Imposing a fixed artificial episode length H.
- Slowly varying the exploration policy over a time frame $O(H)$.
- Dynamically growing the episode length as more data is gathered.

More work is needed to rigorously extend the results of this dissertation to environments without episodic reset. We refer interested readers to a technical note on this subject (Osband and Van Roy, 2016a).

Part II

Model-based Generalization

In part I we introduced reinforcement learning from *tabula rasa*, where little prior knowledge is available to the agent. This formulation is simple, powerful and general; it allows us to design a single algorithm that can learn to take good actions across a huge range of environments. However, this generality comes at a cost. The lower bounds from Section 2.3 establish that any *tabula rasa* algorithm will suffer $\Omega(\sqrt{SAT})$ regret on some MDP, where T is the elapsed time and S and A are the cardinalities of the state and action spaces (Jaksch et al., 2010; Osband and Van Roy, 2016b). This bound implies $T = \Omega(SA)$ time to guarantee a near-optimal policy.

In many interesting environments the number of states and actions could be extremely large or even infinite. For any sort of practical application these learning times $\Omega(SA)$ will be unacceptable. This means that, even if we had an *optimal* approach to reinforcement learning from *tabula rasa* it would still not be useful. The problem is with the *tabula rasa* formulation, which assumes that the agent must learn each state and action separately. Efficient reinforcement learning requires generalization across states and actions to exploit some lower-dimensional structure.

In this part of the dissertation we begin our investigation of algorithms which are able to combine efficient generalization and exploration. We develop a series of algorithms which can generalize over the model of the underlying MDP to achieve regret bounds that depend on the *dimensionality*, rather than *cardinality* of the underlying environment. Importantly, we demonstrate that the same simple and intuitive algorithm posterior sampling for reinforcement learning (PSRL) satisfies these state of the art bounds at a computational cost no greater than solving a single known MDP.

Chapter 5

Factored MDPs

In this chapter we consider a special family of MDPs with low-dimensional structure called *factored* MDPs (Boutilier et al., 2000). The essential material for this chapter is taken from our previous work (Osband and Van Roy, 2014b). Factored MDPs are MDPs whose reward and transition structure exhibit some conditional independence structure. Before we present a formal definition of a factored MDP in Section 5.1, we present a simple example of a factored MDP that helps motivate some of the discussion in this chapter.

Consider a production line with L machines $m_1, .., m_L$ in a sequence. Each machine can exist in one of J possible states and to each machine the operator can apply one of K possible actions. We consider the problem of learning to optimize production in the production line over the course of a working day of length H. We model the environment as a Markov decision process, at each timestep h the state of the system can be described by the state of every machine $s_h \in \{1, .., J\}^L$. Similarly the choice of action at every timestep $a_h \in \{1, .., K\}^L$. Therefore, even for a reasonably small systems[1] the resultant number of states and actions will quickly become greater than the number of atoms in the universe.

Even if we could design an optimal algorithm for *tabula rasa* learning, with no constraints on computation. the regret bounds $O(\sqrt{SAT})$ are entirely useless in this example. The number of potential states and actions is far too large to explore fully. However, we may be able to learn much more efficiently if the system exhibits some low-dimensional structure.

[1]Consider, for example, a production line with $L = 100$ machines, $J = 3$ states $\in \{\text{on}, \text{off}, \text{broken}\}$ and $L = 3$ actions $\in \{\text{turn on}, \text{turn off}, \text{repair}\}$. The total number of states and actions $SA = 9^{100} > 10^{95}$. Most common estimates for the number of atoms in the universe are closer to 10^{80}.

Factored MDPs are good models for large scale structure with several weakly-connected components. We present a formal definition of factored MDPs in the next section, but before we do this it is helpful to consider a motivating example of the production line (Guestrin et al., 2003).

First, we might be able to decompose the reward for the production line into a local cost/reward for each individual machine. Similarly, the evolution of each machine over a single timestep may only directly depend upon its *local* interactions with its neighbors. That is to say machine m_i evolves in a way that depends only upon m_{i-1}, m_i, m_{i+1} and that, conditional upon these machines it is independent of the rest of the production line. Of course, if any individual machine malfunctions it may affect the revenue for the entire plant but this conditional independence over each timestep may help to reduce the dimensionality of this problem exponentially. We present an illustration of this example in Figure 5.1.

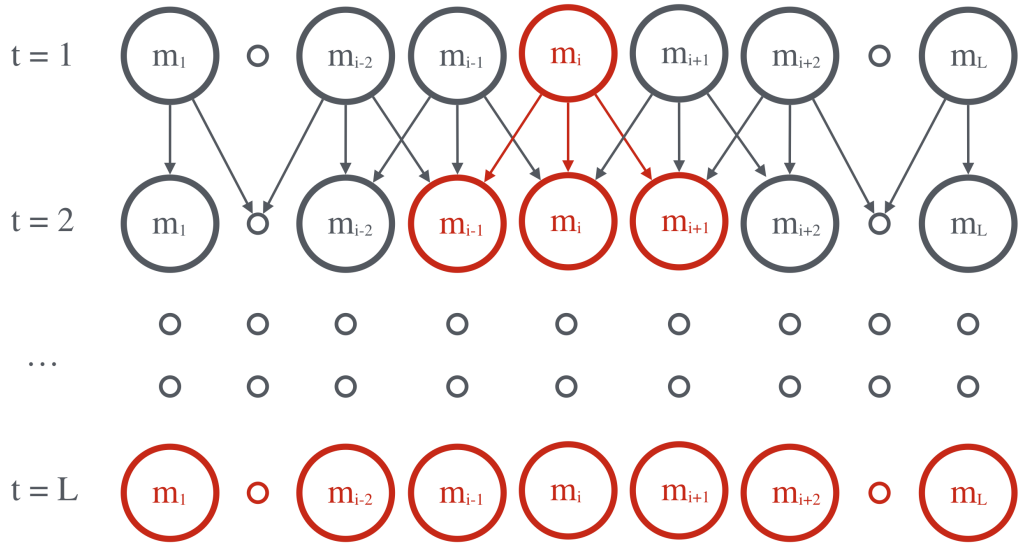


Figure 5.1: State evolution for a production line with sparse conditional dependence.

Intuitively, factored MDPs are Markov decision process whose reward and transition can be represented as a dynamic Bayesian network (DBN) (Boutilier et al., 2000; Guestrin et al., 2003). In the next section we present a formal definition of these systems.

5.1 Problem formulation

We now define what we mean by a factored MDP in a precise sense. To do this clearly we first introduce a few pieces of notation. For these definitions you should imagine that the set $\mathcal{X}$ is a shorthand for the combined state-action space $\mathcal{S} \times \mathcal{A}$.

Definition 5.1 (Scope operation for factored sets $\mathcal{X} = \mathcal{X}_1 \times .. \times \mathcal{X}_n$)**.**
For any subset of indices $Z \subseteq \{1, 2, .., n\}$ let us define the scope set $\mathcal{X}[Z] := \bigotimes_{i \in Z} \mathcal{X}_i$. Further, for any $x \in \mathcal{X}$ define the scope variable $x[Z] \in \mathcal{X}[Z]$ to be the value of the variables $x_i \in \mathcal{X}_i$ with indices $i \in Z$. For singleton sets Z we will write $x[i]$ for $x[\{i\}]$ in the natural way.

Definition 5.2 (Finite categorical distributions)**.**
For any finite sets $\mathcal{X}, \mathcal{Y}$ we define $\mathcal{P}_{\mathcal{X},\mathcal{Y}}$ to be the set of functions mapping elements of $\mathcal{X}$ to probability mass functions over $\mathcal{Y}$.

Definition 5.3 (Sub-Gaussian probability measures with bounded mean)**.**
For any finite set $\mathcal{X}$ define $\mathcal{P}^{C,\sigma}_{\mathcal{X},\mathbb{R}}$ to be the set of distributions mapping elements of $\mathcal{X}$ to distributions over $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$ such that, for any $R \in \mathcal{P}^{C,\sigma}_{\mathcal{X},\mathbb{R}}$ and $x \in \mathcal{X}$ we can write

$$R(x) =_D \overline{r}(x) + W \tag{5.1}$$

where $\overline{r}(x) \in [0, C]$ is deterministic and W is some zero mean σ sub-Gaussian noise.

We can now define what we mean by factored rewards and factored transitions.

Definition 5.4 (Factored reward functions $R \in \mathcal{R} \subseteq \mathcal{P}^{C,\sigma}_{\mathcal{X},\mathbb{R}}$)**.**
The reward function class $\mathcal{R}$ is factored over $\mathcal{S} \times \mathcal{A} = \mathcal{X} = \mathcal{X}_1 \times .. \times \mathcal{X}_n$ with scopes $Z_1, ..Z_l$ if and only if, for all $R \in \mathcal{R}, x \in \mathcal{X}$ there exist functions $\{R_i \in \mathcal{P}^{C,\sigma}_{\mathcal{X}[Z_i],\mathbb{R}}\}_{i=1}^{l}$ such that,

$$\mathbb{E}[r] = \sum_{i=1}^{l} \mathbb{E}[r_i] \tag{5.2}$$

for $r \sim R(x)$ is equal to $\sum_{i=1}^{l} r_i$ with each $r_i \sim R_i(x[Z_i])$ and individually observed.

Definition 5.5 (Factored transition functions $P \in \mathcal{P} \subseteq \mathcal{P}_{\mathcal{X},\mathcal{S}}$)**.**
The transition function class $\mathcal{P}$ is factored over $\mathcal{S} \times \mathcal{A} = \mathcal{X} = \mathcal{X}_1 \times .. \times \mathcal{X}_n$ and $\mathcal{S} = \mathcal{S}_1 \times .. \times \mathcal{S}_m$ with scopes $Z_1, ..Z_m$ if and only if, for all $P \in \mathcal{P}, x \in \mathcal{X}, s \in \mathcal{S}$ there exist some $\{P_i \in \mathcal{P}_{\mathcal{X}[Z_i],\mathcal{S}_i}\}_{i=1}^{m}$ such that,

$$P(s|x) = \prod_{i=1}^{m} P_i \left(s[i] \;\middle|\; x[Z_i] \right) \tag{5.3}$$

A factored MDP (FMDP) is then defined to be an MDP with both factored rewards and factored transitions. If we write $\mathcal{X} = \mathcal{S} \times \mathcal{A}$ we can characterize an FMDP by the tuple

$$M = (\{\mathcal{S}_i\}_{i=1}^m;\ \{\mathcal{X}_i\}_{i=1}^n;\ \{Z_i^R\}_{i=1}^l;\ \{R_i\}_{i=1}^l;\ \{Z_i^P\}_{i=1}^m;\ \{P_i\}_{i=1}^m;\ H;\ \rho),$$

where $Z_i^R, Z_i^P \subseteq \{1, .., n\}$ are the scopes for the reward and transition functions respectively from $\mathcal{X}_i$. Once again, we focus upon the setting where the horizon H is finite.

5.2 Results

Our main result shows that we can bound the expected regret of PSRL in factored MDPs in a polynomial expression the number of *parameters* of the FMDP, which may be exponentially smaller than the number of the states or actions.

Theorem 5.1 (PSRL satisfies strong regret bounds in factored MDPs)**.**
Let M^* be factored with graph structure $\mathcal{G} = (\{\mathcal{S}_i\}_{i=1}^m;\ \{\mathcal{X}_i\}_{i=1}^n;\ \{Z_i^R\}_{i=1}^l;\ \{Z_i^P\}_{i=1}^m;\ \tau)$. If ϕ is the distribution of M^* and Ψ is the span of the optimal value function then we can bound the regret of PSRL:

$$\mathbb{E}\left[\text{Regret}(T, \pi^{\text{PSRL}}, M^*)\right] = \tilde{O}\left(\sum_{i=1}^{l} \sqrt{|\mathcal{X}[Z_i^R]|T} + \Psi \sum_{j=1}^{m} \sqrt{|\mathcal{X}[Z_j^P]||\mathcal{S}_j|T} \right). \tag{5.4}$$

Theorem 5.1 is particularly significant because it removes the dependence on the number of states and replaces it with the size of the factored elements of the MDP. This can lead to an exponential improvement in the bounds. To highlight this improvement consider our earlier example of the production line with L machines, each with J states and K actions. A naive application of Theorem 3.1 would give regret bounds $\tilde{O}\left(J^L\sqrt{K^L T}\right)$, Theorem 4.2 can improve upon this scaling to give $\tilde{O}\left(LJ\sqrt{KT}\right)$. Our proof of Theorem 5.1 relies upon the construction of a new optimistic algorithm, UCRL-Factored, designed to give efficient analysis in factored MDPs (Osband and Van Roy, 2014b). We then follow a similar comparison argument to Theorem 4.1 to apply this analysis to PSRL in expectation.

Theorem 5.2 (UCRL-Factored satisfies strong regret bounds in factored MDPs)**.** *Let M^* be factored with graph structure $\mathcal{G} = (\{\mathcal{S}_i\}_{i=1}^m;\ \{\mathcal{X}_i\}_{i=1}^n;\ \{Z_i^R\}_{i=1}^l;\ \{Z_i^P\}_{i=1}^m;\ \tau)$. If D is the diameter of M^* then for any $\delta > 0$ we can bound the regret of UCRL-Factored with probability at least $1 - \delta$,*

$$\text{Regret}(T, \pi^{\text{PSRL}}, M^*) = \tilde{O}\left(\sum_{i=1}^{l} \sqrt{|\mathcal{X}[Z_i^R]|T} + D\sum_{j=1}^{m} \sqrt{|\mathcal{X}[Z_j^P]||\mathcal{S}_j|T}\right). \tag{5.5}$$

UCRL-Factored is a modification to confidence sets in UCRL to exploit the known factored structure $\mathcal{G}$. The algorithm is not particularly interesting and follows the standard format of Algorithm 3.1, together with its previous shortcomings relative to PSRL. In particular, UCRL-Factored is not computationally tractable even when given access to an approximate solution method for any fixed factored MDP. For a full specification of UCRL-Factored we refer the reader to the paper (Osband and Van Roy, 2014b).

5.2.1 Factored decomposition

Theorems 5.1 and 5.2 hold for any possible factored structure $\mathcal{G}$. However, there are at least two key issues regarding this factorization which we do not address. First, we do not address the question of how the agent acquires the factored structure $\mathcal{G}$, but instead assume it is given a priori. How an agent might uncover some factored structure where it exists is an interesting question and the subject of ongoing research. Second, we assume that the MDP M^* can be precisely captured by a low-dimensional factored representation. In many settings the underlying system may not be truly factored, but the conditional dependence between various parts may be so weak that the factored representation admits faster learning as an approximation. This ties back in to some of the Bayesian robustness guarantees we established through the concept of stochastic optimism in Section 3.3. More research is needed in this area, but it is outside of the scope of this dissertation.

5.3 Analysis

The analysis for UCRL-Factored is remarkably similar to Theorem 3.1 for UCRL in MDPs without factorization. In fact, we follow the exact same argument step for step up to equation (3.10), which we reproduce below,

$$\Delta_k^{\text{conc}} = \sum_{h=1}^{H} \left(\overline{r}_k - \overline{r}^*\right)(x_{kh}) + \sum_{h=1}^{H} \left((P^k - P^*)(x_{kh})\right)^T V_{k,h+1}^k + \sum_{h=1}^{H} d_{kh}.$$

The next step in our analysis is simply to replace the reward and transition functions by their factored representations according to Definitions 5.4 and 5.5. Dealing with the factored rewards is trivial, since these are already additive. We apply concentration results on each component $\overline{r}_i^*$ individually, rather than the to $\overline{r}^* = \sum_{i=1}^{l} \overline{r}_i^*$ in aggregate to improve our contribution from rewards. Dealing with the concentration in the factored transition function seems more subtle. Lemma 5.1 shows that, once again,a we can bound the deviation in $P^*(x)$ by the sum of the deviations in each *factor* of $P_j^*(x)$.

Lemma 5.1 (Bounding factored deviations)**.**
Let the transition function class $\mathcal{P} \subseteq \mathcal{P}_{\mathcal{X},\mathcal{S}}$ be factored over $\mathcal{X} = \mathcal{X}_1 \times .. \times \mathcal{X}_n$ and $\mathcal{S} = \mathcal{S}_1 \times .. \times \mathcal{S}_m$ with scopes $Z_1, ..Z_m$. Then, for any $P, \tilde{P} \in \mathcal{P}$ we may bound their L1 distance by the sum of the differences of their factorizations:

$$\|P(x) - \tilde{P}(x)\|_1 \leq \sum_{i=1}^{m} \|P_i(x[Z_i]) - \tilde{P}_i(x[Z_i])\|_1 \tag{5.6}$$

Proof. We begin with the simple claim that for any $\alpha_1, \alpha_2, \beta_1, \beta_2 \in (0, 1]$:

$$\begin{aligned} |\alpha_1\alpha_2 - \beta_1\beta_2| &= \alpha_2 \left|\alpha_1 - \frac{\beta_1\beta_2}{\alpha_2}\right| \\ &\leq \alpha_2 \left(|\alpha_1 - \beta_1| + \left|\beta_1 - \frac{\beta_1\beta_2}{\alpha_2}\right|\right) \\ &\leq \alpha_2 |\alpha_1 - \beta_1| + \beta_1 |\alpha_2 - \beta_2| \end{aligned}$$

This result also holds for any $\alpha_1, \alpha_2, \beta_1, \beta_2 \in [0, 1]$, where 0 can be verified case by case.

We now consider the probability distributions $p, \tilde{p}$ over $\{1, .., d_1\}$ and $q, \tilde{q}$ over $\{1, .., d_2\}$. We let $Q = pq^T, \tilde{Q} = \tilde{p}\tilde{q}^T$ be the joint probability distribution over $\{1, .., d_1\} \times \{1, .., d_2\}$. Using the claim above we bound the L1 deviation $\|Q - \tilde{Q}\|_1$ by the deviations of their

factors:

$$\begin{aligned}\|Q - \tilde{Q}\|_1 &= \sum_{i=1}^{d_1}\sum_{j=1}^{d_2} |p_i q_j - \tilde{p}_i \tilde{q}_j| \\ &\leq \sum_{i=1}^{d_1}\sum_{j=1}^{d_2} q_j |p_i - \tilde{p}_i| + \tilde{p}_i |q_j - \tilde{q}_j| \\ &= \|p - \tilde{p}\|_1 + \|q - \tilde{q}\|_1.\end{aligned}$$

We conclude the proof by applying this m times to the factored transitions P and $\tilde{P}$. $\square$

Using Lemma 5.1 we can bound the regret in any episode k

$$\begin{aligned}\Delta_k^{\text{conc}} = & \sum_{h=1}^{H}\left\{\sum_{i=1}^{l}((\overline{r}_k)_i - \overline{r}_i^*)(x_{kh}[Z_i^R]) + \sum_{j=1}^{m}\left(((P_k)_j - P_j^*)(x_{kh}[Z_j^P])\right)^T V_{k,h+1}^k + d_{kh},\right\} \\ \leq & \sum_{h=1}^{H}\left\{\sum_{i=1}^{l}|(\overline{r}_k)_i - \overline{r}_i^*|(x_{kh}[Z_i^R]) + \sum_{j=1}^{m}\|(P_k)_j - P_j^*)(x_{kh}[Z_j^P])\|_1 \|V_{k,h+1}^k\|_\infty + |d_{kh}|.\right\} \quad (5.7)\end{aligned}$$

We conclude by a concentration argument exactly as per Theorems 3.1 and 4.1. The key difference is that we now bound the concentration with appeal to the *factors* of R^* and P^* rather that the overall reward or transition function (Osband and Van Roy, 2014b). PSRL is able to exploit this structure to learn in exponentially less data.

These bounds are state of the art for reinforcement learning in factored MDPs and a significant improvement upon previous algorithms. Moreover, PSRL only requires computational cost of solving a single known factored MDP each episode. The bounds are in some sense near optimal since, for the case of L disjoint MDPs we would recover the bounds $\tilde{O}(LS\sqrt{AT})$, which is close to the lower bounds $\Omega(L\sqrt{SAT})$.

5.4 Computational considerations

Even given knowledge of a factored MDP M^*, solving for an ϵ-optimal policy is generally not polynomial in the parameters of the factored representation (Guestrin et al., 2003). Efficient approximate solutions for the optimal policy in large factored MDPs remains an interesting area of research, with several promising results. In this chapter we do not address the question of planning within a factored MDP, but assume that PSRL can access an ϵ-optimal policy for any posterior sample M_k. One piece of good news is that, just like in the

tabular case, the bounds for PSRL are robust to any planning error in any episode.

We now consider a variant of PSRL where, instead of solving for the optimal policy given sampled MDP M_k we allow ourselves to estimate a $\tilde{\delta}_k$-optimal policy. That is to say, the policy μ_k which the algorithm follows during episode k is no more than $\tilde{\delta}_k$-suboptimal for the sampled MDP M_k. We present this algorithm in Algorithm 5.1.

Algorithm 5.1 PSRL with approximate planning

Input: Prior distribution for MDPs ϕ, planning slack $\tilde{\delta}_k \geq 0$ for $k = 1, 2, ..$
for episode $k = 1, 2, ..$ **do**
  sample MDP $M_k \sim \phi(\cdot|\mathcal{H}_{k1})$
  compute μ_k such that $V_{\mu_k,h}^{M_k} \geq \max_{\mu} V_{\mu,h}^{M_k} - \tilde{\delta}_k$
  for time $h = 1, 2, .., H$ **do**
    take action $a_{kh} = \mu_k(s_{kh}, h)$
    observe r_{kh} and s_{kh+1}
    update $\mathcal{H}_{kh} = \mathcal{H}_{kh} \cup (a_{kh}, r_{kh}, s_{kh+1})$
  end for
end for

We note that for PSRL with approximate planning we can no longer apply the posterior sampling lemma, Lemma 4.1. However, we can establish a similar result that for any history $\mathcal{H}_{k1}$,

$$\mathbb{E}[V_{\mu_k,1}^{M_k}|\mathcal{H}_{k1}] \geq \mathbb{E}[V_{\mu^*,1}^{M^*}|\mathcal{H}_{k1}] - \delta_k. \tag{5.8}$$

This result is impactful, since it shows we can perform PSRL with approximate planning and maintain a regret bound similar to Theorem 5.1 with no more than an additional additive term $\sum_{k=1}^{\lceil T/H \rceil} \tilde{\delta}_k$ to the resulting regret bound. In fact, if we choose an appropriate approximation schedule $\tilde{\delta}_k$, this additional term may even be subdominant. Importantly, these results hold regardless of the choice of approximate MDP solution used in conjunction with PSRL. This means that PSRL with approximate planning can be a suitable approach for large scale reinforcement learning even when exact MDP solutions are intractable.

Chapter 6

Parameterized MDPs

We have seen that, when the underlying MDP can be represented by a factored MDP, PSRL can exploit this structure to learn good decisions faster. In this chapter, we extend these results to consider arbitrary MDPs with some parameterized structure but which may not be factored. We are motivated by models such as the linear quadratic regulator, where the underlying state and action space are continuous, but the system's dynamics can be represented concisely. The essential material for this chapter is taken from our previous work (Osband and Van Roy, 2014a).

In this chapter, we demonstrate that PSRL satisfies regret bounds that depend on the dimensionality, rather than the cardinality, of the underlying system. To characterize the complexity of this learning problem we extend the definition of the eluder dimension, previously introduced for bandits (Russo and Van Roy, 2013, 2014), to capture the complexity of the reinforcement learning problem. We use this notion of dimensionality to refine our analysis for PSRL and, in particular, to establish regret bounds for parameterized MDPs in terms of the eluder dimension. Our results provide a unified analysis of model-based reinforcement learning in general and provide new state of the art bounds in several important problem settings.

6.1 Problem formulation

We consider the problem of learning to optimize an unknown MDP where the underlying reward and transition function are known to lie within some known function class. For ease of analysis we assume that the state space $\mathcal{S}$ is a subset of $\mathbb{R}^d$ for some finite d. This allows us to decompose the transition function as a mean value in $\mathcal{S}$ plus additive noise $s' \sim P^M(\cdot|s,a) \implies s' = \overline{p}^M(s,a) + \epsilon_P$. This does not impose much restriction, since we can represent finite MDPs with S states $s_t \in \mathcal{S} = [0,1]^S \subset \mathbb{R}^S$ with a single active component equal to 1 and 0 otherwise. This is sometimes called the "one-hot" state representation. The notation $\mathcal{S} \subseteq \mathbb{R}^d$ should not impose a significant restriction for most practical settings. We extend Definition 5.3 to functions over $\mathbb{R}^d$ and domains which might be infinite.

Definition 6.1 (Sub-Gaussian probability measures with bounded mean)**.**
For any set $\mathcal{X}$ and any $\mathcal{Y} \subseteq \mathbb{R}^d$ for any $d \geq 1$ define $\mathcal{P}^{C,\sigma}_{\mathcal{X},\mathcal{Y}}$ to be the set of distributions mapping elements of $\mathcal{X}$ to distributions over $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$ such that, for any $P \in \mathcal{P}^{C,\sigma}_{\mathcal{X},\mathcal{Y}}$ and $x \in \mathcal{X}$ we can write:

$$P(x) =_D \overline{p}(x) + W \tag{6.1}$$

where $\|\overline{p}(x)\|_2 \in [0, C]$ is deterministic and W is some zero mean σ sub-Gaussian noise.

We will consider rewards and transitions which are suitably drawn from parameterized families taken from $\mathcal{P}^{C,\sigma}_{\mathcal{X},\mathcal{Y}}$. For our analysis of learning in an unknown MDP we need a notion of MDP connectedness, that can regulate the potential error in value estimates from errors in either R^* or P^*. To do this we define the future value function of an MDP.

Definition 6.2 (Future value function)**.**
For any distribution Φ over $\mathcal{S}$, we define the one step future value function U to be the expected value of the optimal policy with the next state distributed according to Φ,

$$U^M_i(\Phi) := \mathbb{E}_{M,\mu^M}\left[V^M_{\mu^M,i+1}(s) \middle| s \sim \Phi\right]. \tag{6.2}$$

We will assume that, the underlying MDP is regular in the sense that future values of similar distributions are similar. Intuitively we require that if the distribution $\Phi, \tilde{\Phi}$ are similar, then the future values under $\Phi, \tilde{\Phi}$ should be similar. We express this idea through the Lipschitz constant on the means of these state distributions.

Definition 6.3 (Lipschitz future value constant)**.**
We write $\mathcal{E}(\Phi) := \mathbb{E}[s|s \sim \Phi] \in \mathcal{S}$ for the mean of a distribution Φ. We say that the future value is Lipschitz with respect to the $\|\cdot\|_2$-norm of the mean,

$$|U_i^M(\Phi) - U_i^M(\tilde{\Phi})| \leq K_i^M(\mathcal{D})\|\mathcal{E}(\Phi) - \mathcal{E}(\tilde{\Phi})\|_2 \text{ for all } \Phi, \tilde{\Phi} \in \mathcal{D} \tag{6.3}$$

We define

$$K^M(\mathcal{D}) := \max_i K_i^M(\mathcal{D}), \tag{6.4}$$

to be a global Lipschitz constant for the future value function with state distributions from $\mathcal{D}$. We condense our notation to write $K^M := K^M(\mathcal{D}(M))$ where $\mathcal{D}(M) := \{P^M(\cdot|s,a)|s \in \mathcal{S}, a \in \mathcal{A}\}$ is the set of all possible one-step state distributions under the MDP M.

We note that, according to Definition 6.3 the future value function can be Lipschitz continuous even when the underlying value function is discontinuous. This is because the noise in the transition function will typically smooth the the future value function. However, the future value function will always be Lipschitz continuous when the underlying rewards and transitions are Lipschitz continuous; a strictly more stringent condition that other works have assumed (Abbasi-Yadkori and Szepesvári, 2015; Ortner and Ryabko, 2012). In the setting of discrete states embedded in $[0,1]^S$ a Lipschitz future value function is equivalent to an optimal value function with finite span (Bartlett and Tewari, 2009).

6.1.1 Standard notions of dimension

In order to quantify the complexity of learning in an a general parameterized environment we review several common notions of dimension.

Definition 6.4 (Mean function class)**.**
For any $\mathcal{G} \subseteq \mathcal{P}_{\mathcal{X},\mathcal{Y}}^{C,\sigma}$ we define the set of mean functions,

$$\mathcal{F} = \mathbb{E}[\mathcal{G}] := \{f|f = \mathcal{E}[G] \text{ for } G \in \mathcal{G}\} \tag{6.5}$$

Sequential observations $y_i \sim G^*(x_i)$ we can equivalently write them as $y_i = f^*(x_i) + \epsilon_i$ for some $f^*(x_i) = \mathbb{E}[y|y \sim G^*(x_i)]$ and ϵ_i zero mean noise.

Definition 6.5 (Covering number)**.**
For any metric space (M, d) we define the α-covering number of a subset $K \subseteq M$ to be the minimum number of α-balls in the metric d needed to cover K,

$$N(K, \alpha, d) = \min\{n | K \subseteq \bigcup_{i=1}^{n} B_\alpha(x_i)\}, \tag{6.6}$$

Where $B_\alpha(x) := \{x' \in M | d(x, x') \leq \alpha\}$.

We let $N(\mathcal{F}, \alpha, \|\cdot\|_2)$ be the α-covering number of $\mathcal{F}$ with respect to the $\|\cdot\|_2$-norm and write $n_\mathcal{F} = \log(8N(\mathcal{F}, 1/T^2, \|\cdot\|_2)T)$ for brevity. The covering number captures some notion of the size of a function class with respect to the metric induced by $\|\cdot\|_2$. The Kolmogorov dimension quantifies the asymptotic rate of growth in this covering number.

Definition 6.6 (Kolmogorov dimension)**.**
The Kolmogorov dimension of a function class $\mathcal{F}$ is given,

$$\dim_K(\mathcal{F}) := \limsup_{\alpha \downarrow 0} \frac{\log(N(\mathcal{F}, \alpha, \|\cdot\|_2))}{\log(1/\alpha)}. \tag{6.7}$$

6.2 Results

Our main result bounds the expected regret for PSRL over an extremely general class of environments in terms of the dimensionality, rather than the cardinality, of the underlying system. We will write $d_E(\mathcal{F}) = \dim_E(\mathcal{F}, T^{-1})$ for the eluder dimension of $\mathcal{F}$ at precision T^{-1}, a notion of dimension specialized to sequential measurements described in Section 6.3.

Theorem 6.1 (Regret bounds for PSRL in parameterized MDPs)**.**
Fix a state space $\mathcal{S}$, action space $\mathcal{A}$, function families $\mathcal{R} \subseteq \mathcal{P}^{C_\mathcal{R}, \sigma_\mathcal{R}}_{\mathcal{S}\times\mathcal{A},\mathbb{R}}$ and $\mathcal{P} \subseteq \mathcal{P}^{C_\mathcal{P}, \sigma_\mathcal{P}}_{\mathcal{S}\times\mathcal{A},\mathcal{S}}$ for any $C_\mathcal{R}, C_\mathcal{P}, \sigma_\mathcal{R}, \sigma_\mathcal{P} > 0$. Let M^* be an MDP with state space $\mathcal{S}$, action space $\mathcal{A}$, rewards $R^* \in \mathcal{R}$ and transitions $P^* \in \mathcal{P}$. If ϕ is the distribution of M^* and $K^* = K^{M^*}$ is a global Lipschitz constant for the future value function as per (6.3) *then,*

$$\mathbb{E}[\text{Regret}(T, \pi^{\text{PSRL}}, M^*)] = \tilde{O}\left(\sigma_\mathcal{R}\sqrt{d_K(\mathcal{R})d_E(\mathcal{R})T} + \mathbb{E}[K^*]\sigma_\mathcal{P}\sqrt{d_K(\mathcal{P})d_E(\mathcal{P})T} \right). \tag{6.8}$$

The result of Theorem 6.1 is established in more detail in the paper (Osband and Van Roy, 2014a) as the result of an explicit finite time analysis. Importantly this result bounds the regret of the reinforcement learning problem as a simple polynomial expression of the dimensionality of the underlying system. In Section 6.3 we provide bounds on the eluder dimension of several function classes. These lead to explicit regret bounds in a number of important domains such as discrete MDPs, linear-quadratic control and even generalized linear systems. In all of these cases the eluder dimension scales comparably with more traditional notions of dimensionality. For clarity, we present bounds in the case of linear-quadratic control.

Corollary 6.1 (Regret bounds for PSRL in bounded linear quadratic systems)**.** *Let M^* be an n-dimensional linear-quadratic system with σ-sub-Gaussian noise. If the state is $\|\cdot\|_2$-bounded by C and ϕ is the distribution of M^*, then:*

$$\mathbb{E}[\text{Regret}(T, \pi^{PS}, M^*)] = \tilde{O}\left(\sigma C \lambda_1 n^2 \sqrt{T}\right). \tag{6.9}$$

Here λ_1 is the largest eigenvalue of the matrix Q given as the solution of the Ricatti equations for the unconstrained optimal value function $V(s) = -s^T Q s$ (Bertsekas et al., 1995).

Proof. We apply the results of for eluder dimension in Section 6.3 to Theorem 6.1 and upper bound the Lipschitz constant of the constrained LQR by $2C\lambda_1$. see Appendix D in (Osband and Van Roy, 2014a). □

6.2.1 Comparison to existing bounds

These bounds are significant because they are the first regret guarantees for any algorithm in arbitrarily parameterized MDPs (Russo and Van Roy, 2013; Osband and Van Roy, 2014a) In addition, these bounds are satisfied by PSRL which is a simple and intuitive algorithm with computational cost no greater than solving a single known MDP. Similar to our analysis in factored MDPs we do not address the question of how to solve a given known MDP, which may itself be intractable. The discussion of computational issues in Section 5.4 applies equally well to this chapter as well. Once again, we also derive a formal optimistic algorithm UCRL-Eluder which matches the scalings of Theorem 6.1 but this algorithm may generally be computationally intractable (Osband and Van Roy, 2014a).

Previous analyses have studied specific classes of parameterized MDPs in an effort to achieve regret bounds that scale with the dimensionality of this underlying system. The most widely-studied parameterization is the degenerate MDP with no transitions, the multi-armed bandit (Auer, 2003; Bubeck et al., 2011; Russo and Van Roy, 2014). Another common assumption is that the transition function is linear in states and actions. Papers here establish regret bounds $\tilde{O}(\sqrt{T})$ for linear quadratic control (Abbasi-Yadkori et al., 2011), but with constants that grow exponentially with dimension. Later works remove this exponential dependence, but only under significant sparsity assumptions (Ibrahimi et al., 2012). Following our analysis in (Osband and Van Roy, 2014a), later works attempted to extend these results to linear quadratic systems without episodic regret (Abbasi-Yadkori and Szepesvári, 2015). However, as outlined in (Osband and Van Roy, 2016a) this proposed analysis is not rigorous and the conjectured theorems have not been verified. The most general previous analysis considers rewards and transitions that are α-Hölder in a d-dimensional space to establish regret bounds $\tilde{O}(T^{(2d+\alpha)/(2d+2\alpha)})$ (Ortner and Ryabko, 2012). However, the proposed algorithm UCCRL is not computationally tractable and the bounds approach linearity in many settings.

6.3 The eluder dimension

To quantify the complexity of learning in a potentially infinite MDP, we extend the existing notion of eluder dimension for real-valued functions (Russo and Van Roy, 2014, 2013) to vector-valued functions. We call this the eluder dimension because we imagine a game where one agent attempts to learn the function $f : \mathcal{X} \to \mathcal{Y}$, but the function $f \in \mathcal{F}$ is chosen by an adversary who attempts to elude the agent. At each timestep t the agent can ask the eluder for value of the the underlying function f at specific measurement points $x_t \in \mathcal{X}$. The agent wants to learn the function f for all $x \in \mathcal{X}$ as quickly as possible, but the eluder wants to stay unknown for as long as possible. The eluder dimension of a function class $\mathcal{F}$ to ϵ-accuracy is the longest possible length of time such that the eluder can avoid the agent.

Existing notions of dimension, such as the Kolmogorov complexity or VC-dimension are insufficient to capture the potential complexity of learning in a sequential problem. For example, a function class with VC-dimension of one may be arbitrarily difficult to learn through sequential experimentation. For a more complete discussion of the eluder dimension and other measures of statistical complexity see (Russo and Van Roy, 2014).

Definition 6.7 $((\mathcal{F}, \epsilon) - dependence)$**.**
We will say that $x \in \mathcal{X}$ *is* $(\mathcal{F}, \epsilon)$*-dependent on* $\{x_1, ..., x_n\} \subseteq \mathcal{X}$

$$\iff \forall f, \tilde{f} \in \mathcal{F}, \quad \sum_{i=1}^{n} \|f(x_i) - \tilde{f}(x_i)\|_2^2 \leq \epsilon^2 \implies \|f(x) - \tilde{f}(x)\|_2 \leq \epsilon.$$

$x \in \mathcal{X}$ *is* $(\epsilon, \mathcal{F})$*-independent of* $\{x_1, .., x_n\}$ *if it does not satisfy the definition for dependence.*

Definition 6.8 (Eluder Dimension)**.**
The eluder dimension $\dim_E(\mathcal{F}, \epsilon)$ *is the length of the longest possible sequence of elements in* $\mathcal{X}$ *such that for some* $\epsilon' \geq \epsilon$ *every element is* $(\mathcal{F}, \epsilon')$*-independent of its predecessors.*

The eluder dimension mirrors the linear dimension for vector spaces, which is the length of the longest sequence such that each element is linearly independent of its predecessors, but allows for nonlinear and approximate dependencies. We overload our notation for $\mathcal{G} \subseteq \mathcal{P}^{C,\sigma}_{\mathcal{X},\mathcal{Y}}$ and write $\dim_E(\mathcal{G}, \epsilon) := \dim_E(\mathbb{E}[\mathcal{G}], \epsilon)$, which should be clear from the context.

6.3.1 Eluder dimension for specific function classes

Theorem 6.1 gives regret bounds in terms of the eluder dimension, which is well-defined for any $\mathcal{F}, \epsilon$. However, for any given $\mathcal{F}, \epsilon$ actually calculating the eluder dimension may take some additional analysis. We now provide bounds on the eluder dimension for some common function classes in a similar approach to earlier work for real-valued functions (Russo and Van Roy, 2014). These proofs are available in Section 6.5.

Proposition 6.1 (Eluder dimension for finite $\mathcal{X}$)**.**
A counting argument shows that for $|\mathcal{X}| = X$ *finite, any* $\epsilon > 0$ *and any function class* $\mathcal{F}$*:*

$$\dim_E(\mathcal{F}, \epsilon) \leq X$$

This bound is tight in the case of independent measurements.

Proposition 6.2 (Eluder dimension for linear functions)**.**
Let $\mathcal{F} = \{f \mid f(x) = \theta\phi(x)$ *for* $\theta \in \mathbb{R}^{n\times p}, \phi \in \mathbb{R}^p, \|\theta\|_2 \leq C_\theta, \|\phi\|_2 \leq C_\phi\}$ *then* $\forall \mathcal{X}$*:*

$$\dim_E(\mathcal{F}, \epsilon) \leq p(4n-1)\frac{e}{e-1}\log\left[\left(1+\left(\frac{2C_\phi C_\theta}{\epsilon}\right)^2\right)(4n-1)\right] + 1 = \tilde{O}(np)$$

Proposition 6.3 (Eluder dimension for quadratic functions)**.**
Let $\mathcal{F} = \{f \mid f(x) = \phi(x)^T\theta\phi(x)$ *for* $\theta \in \mathbb{R}^{p\times p}, \phi \in \mathbb{R}^p, \|\theta\|_2 \leq C_\theta, \|\phi\|_2 \leq C_\phi\}$ *then* $\forall \mathcal{X}$*:*

$$\dim_E(\mathcal{F}, \epsilon) \leq p(4p-1)\frac{e}{e-1}\log\left[\left(1+\left(\frac{2pC_\phi^2 C_\theta}{\epsilon}\right)^2\right)(4p-1)\right] + 1 = \tilde{O}(p^2).$$

Proposition 6.4 (Eluder dimension for generalized linear functions)**.**
Let $g(\cdot)$ *be a component-wise independent function on* $\mathbb{R}^n$ *with derivative in each component bounded* $\in [\underline{h}, \overline{h}]$ *with* $\underline{h} > 0$*. Define* $r = \frac{\overline{h}}{\underline{h}} > 1$ *to be the condition number. If* $\mathcal{F} = \{f \mid f(x) = g(\theta\phi(x))$ *for* $\theta \in \mathbb{R}^{n\times p}, \phi \in \mathbb{R}^p, \|\theta\|_2 \leq C_\theta, \|\phi\|_2 \leq C_\phi\}$ *then for any* $\mathcal{X}$*:*

$$\dim_E(\mathcal{F}, \epsilon) \leq p\left(r^2(4n-2)+1\right)\frac{e}{e-1}\left(\log\left[\left(r^2(4n-2)+1\right)\left(1+\left(\frac{2C_\theta C_\phi}{\epsilon}\right)^2\right)\right]\right) + 1 = \tilde{O}(r^2 np)$$

6.4 Analysis

Our analysis for PSRL in parameterized MDPs follows the same argument as our treatment of factored MDPs. In particular, we compare the performance of PSRL to a computationally intractable optimistic algorithm UCRL-Eluder (Osband and Van Roy, 2014a) and conclude by an argument similar to Theorem 4.1. The analysis for UCRL-Eluder is remarkably similar to Theorem 3.1 for UCRL in MDPs without factorization. In fact, we follow the exact same argument step for step up to equation (3.10), which we reproduce below,

$$\Delta_k^{\text{conc}} = \sum_{h=1}^{H}\left(\overline{r}_k - \overline{r}^*\right)(x_{kh}) + \sum_{h=1}^{H}\left((P_k - P^*)(x_{kh})\right)^T V_{k,h+1}^k + \sum_{h=1}^{H} d_{kh}.$$

The next step in our analysis is to rewrite the future value $P^T V_{h+1}$ in terms f the future value $U_h(P)$. We can then leverage the Lipschitz continuity from Definition 6.3 to say,

$$|\Delta_k^{\text{conc}}| \leq \sum_{h=1}^{H}|(\overline{r}_k - \overline{r}^*)(x_{kh})| + \sum_{h=1}^{H} K^{M_k}\|(\overline{p}_k - \overline{p}^*)(x_{kh})\|_2 + \sum_{h=1}^{H} d_{kh}. \tag{6.10}$$

6.4.1 Designing confidence sets

We will use the eluder dimension build confidence sets for the reward and transition which contain the true functions with high probability and then bound the regret of our algorithm by the maximum deviation within the confidence sets. For observations from $f^* \in \mathcal{F}$ we will center the sets around the least squares estimate $\hat{f}_t^{LS} \in \arg\min_{f \in \mathcal{F}} L_{2,t}(f)$ where $L_{2,t}(f) := \sum_{i=1}^{t-1} \|f(x_t) - y_t\|_2^2$ is the cumulative squared prediction error. The confidence sets are defined $\mathcal{F}_t = \mathcal{F}_t(\beta_t) := \{f \in \mathcal{F} | \|f - \hat{f}_t^{LS}\|_{2,E_t} \leq \sqrt{\beta_t}\}$ where β_t controls the growth of the confidence set and the empirical 2-norm is defined $\|g\|_{2,E_t}^2 := \sum_{i=1}^{t-1} \|g(x_i)\|_2^2$.

For $\mathcal{F} \subseteq \mathcal{P}_{\mathcal{X},\mathcal{Y}}^{C,\sigma}$, we define the distinguished control parameter:

$$\beta_t^*(\mathcal{F}, \delta, \alpha) := 8\sigma^2 \log(N(\mathcal{F}, \alpha, \|\cdot\|_2)/\delta) + 2\alpha t \left(8C + \sqrt{8\sigma^2 \log(4t^2/\delta)}\right) \tag{6.11}$$

This leads to confidence sets which contain the true function with high probability.

Proposition 6.5 (Confidence sets with high probability)**.**
For all $\delta > 0$ and $\alpha > 0$ and the confidence sets $\mathcal{F}_t = \mathcal{F}_t(\beta_t^*(\mathcal{F}, \delta, \alpha))$ for all $t \in \mathbb{N}$ then:

$$\mathbb{P}\left(f^* \in \bigcap_{t=1}^{\infty} \mathcal{F}_t\right) \geq 1 - 2\delta$$

Proof. We combine standard martingale concentrations with a discretization scheme. The argument is essentially the same as Proposition 6 in (Russo and Van Roy, 2014), but extends statements about $\mathbb{R}$ to vector-valued functions. A full derivation is available in the Section 6.5. □

6.4.2 Bounding the sum of set widths

We now bound the deviation from f^* by the maximum deviation within the confidence set.

Definition 6.9 (Set widths)**.**
For any set of functions $\mathcal{F}$ we define the width of the set at x to be the maximum L2 deviation between any two members of $\mathcal{F}$ evaluated at x.

$$w_{\mathcal{F}}(x) := \sup_{\overline{f}, \underline{f} \in \mathcal{F}} \|\overline{f}(x) - \underline{f}(x)\|_2$$

We can bound for the number of large widths in terms of the eluder dimension.

Lemma 6.1 (Bounding the number of large widths)**.**
If $\{\beta_t > 0 | t \in \mathbb{N}\}$ *is a nondecreasing sequence with* $\mathcal{F}_t = \mathcal{F}_t(\beta_t)$ *then*

$$\sum_{k=1}^{m}\sum_{i=1}^{\tau} \mathbb{1}\{w_{\mathcal{F}_{t_k}}(x_{t_k+i}) > \epsilon\} \leq \left(\frac{4\beta_T}{\epsilon^2} + \tau\right) \dim_E(\mathcal{F}, \epsilon)$$

Proof. This result follows from proposition 8 in (Russo and Van Roy, 2014) but with a small adjustment to account for episodes. A full proof is given in Section 6.5. □

We can use Lemma 6.1 to control the cumulative deviation through time.

Proposition 6.6 (Bounding the sum of widths)**.**
If $\{\beta_t > 0 | t \in \mathbb{N}\}$ *is nondecreasing with* $\mathcal{F}_t = \mathcal{F}_t(\beta_t)$ *and* $\|f\|_2 \leq C$ *for all* $f \in \mathcal{F}$ *then:*

$$\sum_{k=1}^{m}\sum_{i=1}^{\tau} w_{\mathcal{F}_{t_k}}(x_{t_k+i}) \leq 1 + \tau C \dim_E(\mathcal{F}, T^{-1}) + 4\sqrt{\beta_T \dim_E(\mathcal{F}, T^{-1}) T} \tag{6.12}$$

Proof. Once again we follow the analysis of Russo (Russo and Van Roy, 2014) and streamline notation by letting $w_t = w_{\mathcal{F}_{t_k}}(x_{t_k+i})$ and $d = \dim_E(\mathcal{F}, T^{-1})$. Reordering the sequence $(w_1, .., w_T) \to (w_{i_1}, .., w_{i_T})$ such that $w_{i_1} \geq .. \geq w_{i_T}$ we have that:

$$\sum_{k=1}^{m}\sum_{i=1}^{\tau} w_{\mathcal{F}_{t_k}}(x_{t_k+i}) = \sum_{t=1}^{T} w_{i_t} \leq 1 + \sum_{i=1}^{T} w_{i_t} \mathbb{1}\{w_{i_t} \geq T^{-1}\}$$

.

By the reordering we know that $w_{i_t} > \epsilon$ means that $\sum_{k=1}^{m}\sum_{i=1}^{\tau} \mathbb{1}\{w_{\mathcal{F}_{t_k}}(x_{t_k+i}) > \epsilon\} \geq t$. From Lemma 6.1, $\epsilon \leq \sqrt{\frac{4\beta_T d}{t-\tau d}}$. So that if $w_{i_t} > T^{-1}$ then $w_{i_t} \leq \min\{C, \sqrt{\frac{4\beta_T d}{t-\tau d}}\}$. Therefore,

$$\sum_{i=1}^{T} w_{i_t} \mathbb{1}\{w_{i_t} \geq T^{-1}\} \leq \tau C d + \sum_{t=\tau d+1}^{T} \sqrt{\frac{4\beta_T d}{t-\tau d}} \leq \tau C d + 2\sqrt{\beta_T} \int_0^T \sqrt{\frac{d}{t}}\, dt \leq \tau C d + 4\sqrt{\beta_T d T}$$

□

6.4.3 Regret bounds

We can now combine Proposition 6.5 to bound the regret from equation 6.10. The posterior sampling lemma ensures that $\mathbb{E}[K^k] = \mathbb{E}[K^*]$ so that $\mathbb{E}[K^k|A,B] \leq \frac{\mathbb{E}[K^*]}{\mathbb{P}(A,B)} \leq \frac{\mathbb{E}[K^*]}{1-8\delta}$ by a union bound on $\{A^c \cup B^c\}$. We fix $\delta = 1/8T$ to see that:

$$\mathbb{E}[\text{Regret}(T, \pi^{PS}, M^*)] \leq (C_{\mathcal{R}} + C_{\mathcal{P}}) + \sum_{k=1}^{m}\sum_{i=1}^{\tau} w_{\mathcal{R}_k}(x_{k,i}) + \mathbb{E}[K^*]\left(1 + \frac{1}{T-1}\right)\sum_{k=1}^{m}\sum_{i=1}^{\tau} w_{\mathcal{P}_t}(x_{k,i})$$

We now use equation (6.11) together with Proposition 6.6 to obtain our regret bounds. For ease of notation we will write $d_E(\mathcal{R}) = \dim_E(\mathcal{R}, T^{-1})$ and $d_E(\mathcal{P}) = \dim_E(\mathcal{P}, T^{-1})$.

$$\begin{aligned}\mathbb{E}[\text{Regret}(T, \pi^{PS}, M^*)] \quad \leq \quad & 2 + (C_{\mathcal{R}} + C_{\mathcal{P}}) + \tau(C_{\mathcal{R}} d_E(\mathcal{R}) + C_{\mathcal{P}} d_E(\mathcal{P})) + \\ & 4\sqrt{\beta_T^*(\mathcal{R}, 1/8T, \alpha) d_E(\mathcal{R}) T} + 4\sqrt{\beta_T^*(\mathcal{P}, 1/8T, \alpha) d_E(\mathcal{P}) T}\end{aligned} \tag{6.13}$$

We let $\alpha = 1/T^2$ and write $n_{\mathcal{F}} = \log(8N(\mathcal{F}, 1/T^2, \|\cdot\|_2)T)$ for $\mathcal{R}$ and $\mathcal{P}$ to complete our proof of Theorem 6.1,

$$\mathbb{E}[\text{Regret}(T, \pi^{PS}, M^*)] \leq [C_{\mathcal{R}} + C_{\mathcal{P}}] + \tilde{D}(\mathcal{R}) + \mathbb{E}[K^*]\left(1 + \frac{1}{T-1}\right)\tilde{D}(\mathcal{P}) \tag{6.14}$$

Where $\tilde{D}(\mathcal{F})$ is shorthand for $1 + \tau C_{\mathcal{F}} d_E(\mathcal{F}) + 8\sqrt{d_E(\mathcal{F})(4C_{\mathcal{F}} + \sqrt{2\sigma_{\mathcal{F}}^2 \log(32T^3)})} + 8\sqrt{2\sigma_{\mathcal{F}}^2 n_{\mathcal{F}} d_E(\mathcal{F}) T}$. The first term $[C_{\mathcal{R}} + C_{\mathcal{P}}]$ bounds the contribution from missed confidence sets. The cost of learning the reward function R^* is bounded by $\tilde{D}(\mathcal{R})$. In most problems the remaining contribution bounding transitions and lost future value will be dominant. Theorem 6.1 follows from the Definition 6.6 together with $n_{\mathcal{R}}$ and $n_{\mathcal{P}}$.

6.5 Technical proofs

In this subsection we will present the remainder of the technical results that we omitted earlier in the chapter. These arguments are essentially adaptation of the technical results from (Russo and Van Roy, 2014) but with some extensions to account for vector valued functions of dimension $d \geq 1$. We will begin with a proof of Proposition 6.5, that the confidence sets defined by β^* in equation 6.11 hold with high probability. Before we can do this, we reproduce some elementary results from martingale theory.

Lemma 6.2 (Exponential martingale)**.**
Let $Z_i \in L^1$ be real-valued random variables adapted to $\mathcal{H}_i$. We define the conditional mean $\mu_i = \mathbb{E}[Z_i|\mathcal{H}_{i-1}]$ and conditional cumulant generating function $\psi_i(\lambda) = \log \mathbb{E}[\exp(\lambda(Z_i - \mu_i))|\mathcal{H}_{i-1}]$, then

$$M_n(\lambda) = \exp\left(\sum_1^n \lambda(Z_i - \mu_i) - \psi_i(\lambda)\right)$$

is a martingale with $\mathbb{E}[M_n(\lambda)] = 1$.

Lemma 6.3 (Martingale concentration guarantee)**.**
For Z_i adapted real L^1 random variables adapted to $\mathcal{H}_i$. We define the conditional mean $\mu_i = \mathbb{E}[Z_i|\mathcal{H}_{i-1}]$ and conditional cumulant generating function $\psi_i(\lambda) = \log \mathbb{E}[\exp(\lambda(Z_i - \mu_i))|\mathcal{H}_{i-1}]$.

$$\mathbb{P}\left(\bigcup_{n=1}^{\infty}\{\sum_1^n \lambda(Z_i - \mu_i) - \psi_i(\lambda) \geq x\}\right) \leq e^{-x}$$

Both of these lemmas are available in earlier discussion for real-valued variables (Russo and Van Roy, 2014). We specialize our discussion to the vector space $\mathcal{Y} \subseteq \mathbb{R}^d$ where the inner product $< y, y >= \|y\|_2^2$. To simplify notation we will write $f_t^* := f^*(x_t)$ and $f_t = f(x_t)$ for arbitrary $f \in \mathcal{F}$. We define

$$\begin{aligned} Z_t &= \|f_t^* - y_t\|_2 - \|f_t - y_t\|_2 \\ &= \langle f_t^* - y_t, f_t^* - y_t\rangle - \langle f_t - y_t, f_t - y_t, f_t - y_t\rangle \\ &= -\langle f_t - f_t^*, f_t - f_t^*\rangle + 2\langle f_t - f_t^*, y_t - f_t^*\rangle \\ &= -\|f_t - f_t^*\|_2 + 2\langle f_t - f_t^*, \epsilon_t\rangle \end{aligned}$$

so that clearly $\mu_t = -\|f_t - f_t^*\|_2$. As we have said that the noise is σ-sub-Gaussian, $\mathbb{E}[\exp(\langle\phi, \epsilon\rangle)] \leq \exp\left(\frac{\|\phi\|_2^2\sigma^2}{2}\right)\ \forall \phi \in \mathcal{Y}$. From here we can deduce that:

$$\begin{aligned} \psi_t(\lambda) &= \log \mathbb{E}[\exp(\lambda(Z_t - \mu_t))|\mathcal{H}_{t-1}] \\ &= \log \mathbb{E}[\exp(2\lambda\langle f_t - f_t^*, \epsilon_t\rangle)] \\ &\leq \frac{\|2\lambda(f_t - f_t^*)\|_2^2\sigma^2}{2}. \end{aligned}$$

We write $\sum_{i=1}^{t-1} Z_i = L_{2,t}(f^*) - L_{2,t}(f)$ according to our earlier definition of $L_{2,t}$. We can apply Lemma 6.3 with $\lambda = 1/4\sigma^2$, $x = log(1/\delta)$ to obtain:

$$\mathbb{P}\{\Big(L_{2,t}(f) \geq L_{2,t}(f^*) + \frac{1}{2}\|f - f^*\|_{2,E_t} - 4\sigma^2 \log(1/\delta)\Big) \ \forall t\} \geq 1 - \delta$$

substituting $f = \hat{f}$ to be the least squares solution which minimizes $L_{2,t}(f)$ we can remove $L_{2,t}(\hat{f}) - L_{2,t}(f^*) \leq 0$. From here we use an α-cover discretization argument to complete the proof of Proposition 6.5.

Let $\mathcal{F}^\alpha \subset \mathcal{F}$ be an α-2 cover of $\mathcal{F}$ such that $\forall f \in \mathcal{F}$ there is some $\|f^\alpha - f\|_2 \leq \alpha$. We can use a union bound on $\mathcal{F}^\alpha$ so that $\forall f \in \mathcal{F}$:

$$L_{2,t}(f) - L_{2,t}(f^*) \geq \frac{1}{2}\|f - f^*\|_{2,E_t} - 4\sigma^2 \log(N(\mathcal{F}, \alpha, \|\cdot\|_2)/\delta) + DE(\alpha) \tag{6.15}$$

$$\text{For } DE(\alpha) = \min_{f^\alpha \in \mathcal{F}^\alpha} \left\{\frac{1}{2}\|f^\alpha - f^*\|^2_{2,E_t} - \frac{1}{2}\|f - f^*\|^2_{2,E_t} + L_{2,t}(f) - L_{2,t}(f^\alpha)\right\}$$

We will seek to bound this discretization error with high probability.

Lemma 6.4 (Bounding discretization error)**.**
If $\|f^\alpha(x) - f(x)\|_2 \leq \alpha$ for all $x \in \mathcal{X}$ then with probability at least $1 - \delta$:

$$DE(\alpha) \leq \alpha t \left[8C + \sqrt{8\sigma^2 \log(4t^2/\delta)}\right]$$

Proof. For non-trivial bounds we will consider the case of $\alpha \leq C$ and note that via Cauchy-Schwarz:

$$\|f^\alpha(x)\|_2^2 - \|f(x)\|_2^2 \leq \max_{\|y\|_2 \leq \alpha} \|f(x) + y\|_2^2 - \|f\|_2^2 \leq 2C\alpha + \alpha^2.$$

From here we can say that

$$\|f^\alpha(x) - f^*(x)\|_2^2 - \|f(x) - f^*(x)\|_2^2 \ = \ \|f^\alpha(x)\|_2^2 - \|f(x)\|_2^2 + 2\langle f^*(x), f(x) - f^\alpha(x)\rangle \ \leq \ 4C\alpha,$$

and

$$\|y - f(x)\|_2^2 - \|y - f^\alpha(x)\|_2^2 \ = \ 2\langle y, f^\alpha(x) - f(x)\rangle + \|f(x)\|_2^2 - \|f^\alpha(x)\|_2^2 \ \leq \ 2\alpha|y| + 2C\alpha + \alpha^2.$$

Summing these expressions over time $i = 1, .., t-1$ and using sub-Gaussian high probability bounds on $|y|$ gives our desired result. □

Finally we apply Lemma 6.4 to equation 6.15 and use the fact that $\hat{f}_t^{LS}$ is the $L_{2,t}$ minimizer to obtain the result that with probability at least $1-2\delta$:

$$\|\hat{f}_t^{LS} - f^*\|_{2,E_t} \leq \sqrt{\beta_t^*(\mathcal{F}, \alpha, \delta)}$$

Which is our desired result. The following lemma establishes that the number of

Lemma 6.5 (Bounding the number of large widths)**.**
If $\{\beta_t > 0 | t \in \mathbb{N}\}$ *is a nondecreasing sequence with* $\mathcal{F}_t = \mathcal{F}_t(\beta_t)$ *then*

$$\sum_{k=1}^{m}\sum_{i=1}^{\tau} \mathbb{1}\{w_{\mathcal{F}_{t_k}}(x_{t_k+i}) > \epsilon\} \leq \left(\frac{4\beta_T}{\epsilon^2} + \tau\right) \dim_E(\mathcal{F}, \epsilon)$$

Proof. We first imagine that $w_{\mathcal{F}_t}(x_t) > \epsilon$ and is ϵ-dependent on K disjoint subsequences of $x_1, .., x_{t-1}$. If x_t is ϵ-dependent on K disjoint subsequences then there exist $\|\overline{f} - \underline{f}\|_{2,E_t} > K\epsilon^2$. By the triangle inequality $\|\overline{f} - \underline{f}\|_{2,E_t} \leq 2\sqrt{\beta_t} \leq 2\sqrt{\beta_T}$ so that $K < 4\beta_T/\epsilon^2$.

In the case without episodic delay, Russo went on to show that in any sequence of length l there is some element which is ϵ-dependent on at least $\frac{l}{\dim_E(\mathcal{F},\epsilon)} - 1$ disjoint subsequences (Russo and Van Roy, 2014). Our analysis follows similarly, but we may lose up to $\tau - 1$ proper subsequences due to the delay in updating the episode. This means that we can only say that $K \geq \frac{l}{\dim_E(\mathcal{F},\epsilon)} - \tau$. Considering the subsequence $w_{\mathcal{F}_{t_k}}(x_{t_k+i}) > \epsilon$ we see that $l \leq \left(\frac{4\beta_T}{\epsilon^2} + \tau\right) \dim_E(\mathcal{F}, \epsilon)$ as required. □

6.5.1 Eluder dimension for specific function classes

By Definition 6.8, the eluder dimension $\dim_E(\mathcal{F}, \epsilon)$ is the length d of the longest sequence $x_1, .., x_d$ such that for some $\epsilon' \geq \epsilon$:

$$w_k = \sup\left\{\|(\overline{f} - \underline{f})(x_k)\|_2 \;\middle|\; \|\overline{f} - \underline{f}\|_{2,E_t} \leq \epsilon'\right\} > \epsilon' \tag{6.16}$$

In this subsection we provide bounds on the eluder dimension for specific function classes.

Finite domain $\mathcal{X}$

Any $x \in \mathcal{X}$ is ϵ-dependent upon itself for all $\epsilon > 0$. Therefore for all $\epsilon > 0$ the eluder dimension of $\mathcal{F}$ is bounded by $|\mathcal{X}|$.

Linear functions $f(x) = \theta\phi(x)$

Let $\mathcal{F} = \{f \mid f(x) = \theta\phi(x) \text{ for } \theta \in \mathbb{R}^{n\times p}, \phi \in \mathbb{R}^p, \|\theta\|_2 \leq C_\theta, \|\phi\|_2 \leq C_\phi\}$. To simplify our notation we will write $\phi_k = \phi(x_k)$ and $\theta = \theta_1 - \theta_2$. From here, we may manipulate the expression

$$\|\theta\phi\|_2^2 = \phi_k^T\theta^T\theta\phi_k = \text{Trace}(\phi_k^T\theta^T\theta\phi_k) = \text{Trace}(\theta\phi_k\phi_k^T\theta)$$

$$\implies w_k = \sup_\theta\{\|\theta\phi_k\|_2 \mid \text{Trace}(\theta\Phi_k\theta^T) \leq \epsilon^2\} \text{ where } \Phi_k := \sum_{i=1}^{k-1}\phi_i\phi_i^T$$

We next require a lemma which gives an upper bound for trace constrained optimizations.

Lemma 6.6 (Bounding norms under trace constraints)**.**
Let $\theta \in \mathbb{R}^{n\times p}, \phi \in \mathbb{R}^p$ and $V \in \mathbb{R}_{++}^{p\times p}$, the set of positive definite $p \times p$ matrices, then:

$$W^2 = \max_\theta \|\theta\phi\|_2^2 \textit{ subject to } \text{Trace}(\theta V\theta^T) \leq \epsilon^2$$

is bounded above by $W^2 \leq (2n-1)\epsilon^2\|\phi\|_{V^{-1}}^2$ where $\|\phi\|_A^2 := \phi^T A\phi$.

Proof. We first note that $\|\theta\phi\|_2^2 = \text{Trace}(\theta\phi\phi^T\theta^T) = \sum_1^n(\theta\phi)_i^2 \leq (\sum_1^n(\theta\phi)_i)^2$ by Jensen's inequality. We define $\tilde{\Phi} \in \mathbb{R}^{n\times p}$ such that each row of $\tilde{\Phi} = \phi^T$. Then this inequality can be expressed as:

$$W^2 = \text{Trace}(\theta\phi\phi^T\theta^T) \leq \text{Sum}(\theta \otimes \tilde{\Phi})^2$$

Where $A \otimes B = C$ for $C_{ij} = A_{ij}B_{ij}$ and $\text{Sum}(C) := \sum_{i,j} C_{ij}$ We can now obtain an upper bound for our original problem through this convex relaxation:

$$\max_\theta \text{Sum}(\theta \otimes \tilde{\Phi}) \text{ subject to } \text{Trace}(\theta V\theta^T) \leq \epsilon^2$$

We form the Lagrangian $\mathcal{L}(\theta, \lambda) = -\text{Sum}(\theta \otimes \tilde{\Phi}) + \lambda(\text{Trace}(\theta V \theta^T) - \epsilon^2)$. Solving for first order optimality $\nabla_\theta \mathcal{L} = 0 \implies \theta^* = \frac{1}{2\lambda}\tilde{\Phi}V^{-1}$. From here we form the dual objective

$$g(\lambda) = -\text{Sum}(\frac{1}{2\lambda}\tilde{\Phi}V^{-1} \otimes \tilde{\Phi}) + \text{Trace}(\frac{1}{4\lambda}\tilde{\Phi}V^{-1}\tilde{\Phi}^T) - \lambda\epsilon^2$$

Here we solve for the dual-optimal λ^*,

$$\nabla_\lambda g = 0 \implies \frac{1}{2\lambda^*}^2 \text{Sum}(\frac{1}{2\lambda}\tilde{\Phi}V^{-1} \otimes \tilde{\Phi}) - \frac{1}{4\lambda^*}^2 \text{Trace}(\frac{1}{4\lambda}\tilde{\Phi}V^{-1}\tilde{\Phi}^T) = \epsilon^2.$$

From the definition of $\tilde{\Phi}$, $\text{Sum}(\tilde{\Phi}V^{-1} \otimes \tilde{\Phi}) = n\phi^T V^{-1}\phi$ and $\text{Trace}(\tilde{\Phi}V^{-1}\tilde{\Phi}^T) = \phi^T V^{-1}\phi$. We can simplify our expression to conclude:

$$\begin{aligned}
\frac{n}{2\lambda^{*2}}\phi^T V^{-1}\phi - \frac{1}{4\lambda^{*2}}\phi^T V^{-1}\phi = \epsilon^2 &\implies \lambda^* = \sqrt{\frac{(n-1/2)}{2\epsilon^2}}\|\phi\|_{V^{-1}} \\
&\implies g(\lambda^*) = -\frac{n}{2\lambda^*}\|\phi\|^2_{V^{-1}} + \frac{1}{4\lambda^*}\|\phi\|^2_{V^{-1}} - \lambda^*\epsilon \\
\text{strong duality} &\implies f(\theta^*) = g(\lambda^*) = \sqrt{2n-1}\epsilon\|\phi\|_{V^{-1}}.
\end{aligned}$$

Therefore, we conclude that the optimal value of $W^2 \leq f(\theta^*)^2 \leq (2n-1)\epsilon^2\|\phi\|^2_{V^{-1}}$. □

We can use Lemma 6.6, to bound the eluder dimension for linear functions. Using the definition of w_k from equation 6.16 together with Φ_k we may rewrite:

$$w_k = \max_\theta \left\{ \sqrt{\text{Trace}(\theta\phi_k\phi_k^T\theta)} \mid \text{Trace}(\theta\Phi_k\theta^T) \leq \epsilon^2 \right\}.$$

Let $V_k := \Phi_k + \left(\frac{\epsilon}{2C_\theta}\right)^2 I$ so that $\text{Trace}(\theta\Phi_k\theta^T) \leq \epsilon^2 \implies \text{Trace}(\theta V_k\theta^T) \leq 2\epsilon^2$ through a triangle inequality. Now, applying Lemma 6.6 we can say that $w_k \leq \epsilon\sqrt{4n-2}\|\phi_k\|_{V_k^{-1}}$. This means that if $w_k \geq \epsilon$ then $\|\phi_k\|^2_{V_k^{-1}} > \frac{1}{4n-2} > 0$.

We imagine that $w_i \geq \epsilon$ for each $i < k$. Then since $V_k = V_{k-1} + \phi_k\phi_k^T$ we can use the Matrix Determinant together with the above observation to say that:

$$\det(V_k) = \det(V_{k-1})(1 + \phi_k^T V_K^{-1}\phi_k) \geq \det(V_{k-1})\left(1 + \frac{1}{4n-2}\right) \geq .. \geq \lambda^p\left(1 + \frac{1}{4n-2}\right)^{k-1} \tag{6.17}$$

for $\lambda := \left(\frac{\epsilon}{2C_\theta}\right)^2$.

To obtain an upper bound on the determinant we note that $\det(V_k)$ is maximized when all eigenvalues are equal or equivalently:

$$\det(V_k) \leq \left(\frac{\text{Trace}(V_k)}{p}\right)^p \leq \left(\frac{C_\phi^2(k-1)}{p} + \lambda\right)^p \tag{6.18}$$

Using equations 6.17 and 6.18 together we see that k must satisfy the inequality,

$$\left(1 + \frac{1}{4n-2}\right)^{(k-1)/p} \leq \frac{C_\phi^2(k-1)}{\lambda p} + 1.$$

We write $\zeta_0 = \frac{1}{4n-2}$ and $\alpha_0 = \frac{C_\phi^2}{\lambda} = \left(\frac{2C_\phi C_\theta}{\epsilon}\right)^2$ so that we can epress this as:

$$(1+\zeta_0)^{\frac{k-1}{p}} \leq \alpha_0 \frac{k-1}{p} + 1$$

We can use the result that

$$B(x,\alpha) = \max\{B \mid (1+x)^B \leq \alpha B + 1\} \leq \frac{1+x}{x}\frac{e}{e-1}\{\log(1+\alpha) + \log(\frac{1+x}{x})\}.$$

We complete the proof of Proposition 6.2 through computing this upper bound at (ζ_0, α_0),

$$\dim_E(\mathcal{F}, \epsilon) \leq p(4n-1)\frac{e}{e-1}\log\left[\left(1 + \left(\frac{2C_\phi C_\theta}{\epsilon}\right)^2\right)(4n-1)\right] + 1 = \tilde{O}(np).$$

□

Quadratic functions $f(x) = \phi^T(x)\theta\phi(x)$

Let $\mathcal{F} = \{f \mid f(x) = \phi(x)^T\theta\phi(x) \text{ for } \theta \in \mathbb{R}^{p\times p}, \phi \in \mathbb{R}^p, \|\theta\|_2 \leq C_\theta, \|\phi\|_2 \leq C_\phi\}$ then for any $\mathcal{X}$ we can say that:

$$\dim_E(\mathcal{F}, \epsilon) \leq p(4p-1)\frac{e}{e-1}\log\left[\left(1 + \left(\frac{2pC_\phi^2 C_\theta}{\epsilon}\right)^2\right)(4p-1)\right] + 1 = \tilde{O}(p^2).$$

Where we have simply applied the linear result with $\tilde{\epsilon} = \frac{\epsilon}{pC_\mathcal{P}}$. This is valid since if we can identify the linear function $g(x) = \theta\phi(x)$ to within this tolerance then we will certainly know $f(x)$ as well.

Generalized linear models

Let $g(\cdot)$ be a component-wise independent function on $\mathbb{R}^n$ with derivative in each component bounded $\in [\underline{h}, \overline{h}]$ with $\underline{h} > 0$. Define $r = \frac{\overline{h}}{\underline{h}} > 1$ to be the condition number. If $\mathcal{F} = \{f \mid f(x) = g(\theta\phi(x)) \text{ for } \theta \in \mathbb{R}^{n\times p}, \phi \in \mathbb{R}^p, \|\theta\|_2 \leq C_\theta, \|\phi\|_2 \leq C_\phi\}$ then for any $\mathcal{X}$:

$$\dim_E(\mathcal{F}, \epsilon) \leq p\left(r^2(4n-2)+1\right)\frac{e}{e-1}\left(\log\left[\left(r^2(4n-2)+1\right)\left(1+\left(\frac{2C_\theta C_\phi}{\epsilon}\right)^2\right)\right]\right)+1 = \tilde{O}(r^2np)$$

This proof follows exactly as per the linear case, but first using a simple reduction on the form of equation (6.16).

$$\begin{aligned} w_k &= \sup\left\{\|(\overline{f}-\underline{f})(x_k)\|_2 \;\middle|\; \|\overline{f}-\underline{f}\|_{2,E_t} \leq \epsilon'\right\} \\ &\leq \max_{\theta_1,\theta_2}\left\{\|g(\theta_1\phi_k)-g(\theta_2\phi_k)\|_2 \mid \sum_{i=1}^{k-1}\|g(\theta_1\phi_i)-g(\theta_2\phi_i)\|_2^2 \leq \epsilon^2\right\} \\ &\leq \max_{\theta}\left\{\overline{h}\|\theta\phi_k\|_2 \mid \sum_{i=1}^{k-1}\underline{h}^2\|\theta\phi_i\|_2^2 \leq \epsilon^2\right\} \end{aligned}$$

To which we can apply Lemma 6.6 with the ϵ rescaled by r. Following the same arguments as for linear functions completes our proof.

Part III

Value-based Generalization

Part II of this dissertation shows that PSRL can effectively synthesize efficient exploration and generalization in reinforcement learning. We establish that this simple and intuitive algorithm satisfies statistical efficiency guarantees that scale with the dimensionality, rather than the cardinality, of the underlying MDP. These results effectively reduce the problem of learning to control an unknown MDP to the problem of planning in a single *known* MDP. However, even where the underlying MDP is known, solving for optimal policy might itself be an intractable problem. An example of this phenomenon is the board game Go who's dynamics are simple and deterministic. In these sorts of environments an algorithm for efficient RL must be able to exploit low dimensional structure not just in the underlying environment, but also its solution to the optimal policy (Silver et al., 2016)

In this part of the dissertation, we explore an approach to generalization and exploration in large environments where even MDP planning may be intractable. Once again, we will tackle this problem through an algorithm driven by randomized value functions. PSRL generates randomized value functions through solving a posterior sample for the underlying MDP. We will seek to obviate the need for separate MDP planning by estimating a distribution over the value function directly. At a high level, this distribution will be designed to approximate the posterior distribution for the value function while exploiting generalization in the value function for statistical and computational efficiency. We present a natural adaptation of PSRL to domains with value function generalization that can be used with both linear and nonlinear function approximation. We prove that this approach is statistically efficient in some simple settings which are amenable to analysis. We also present extensive expetiments which suggest that randomized value functions can drive simultaneously computationally and statistically efficient reinforcement learning in complex environments.

Chapter 7

Linear Value Functions

We now consider the problem of learning to control an unknown environment where the underlying number of states and actions is very large or even infinite. For such large systems an exact tabular solution to the optimal value function is not tractable, even if the agent has perfect knowledge a priori (Sutton and Barto, 1998). In these settings, one common approach is to approximate the value of a policy by some shared functional representation (Tsitsiklis and Van Roy, 1997). In this chapter we focus upon the linear setting where the agent approximates the value function as the linear combination of some known basis functions. These basis functions may encode prior structure to the learning problem, as well as generalize across states and actions. Synthesizing efficient generalization with exploration in complex domains remains a significant challenge for reinforcement learning.

In this chapter, we propose *randomized least squares value iteration* (RLSVI), an algorithm that implements deep exploration via randomized value functions with linear basis functions. The essential material from this chapter is taken from our paper (Osband et al., 2014). The core idea of RLSVI is very simple; building up an exact posterior over value functions via PSRL is intractable, instead approximate the posterior for Bellman equation by a Gaussian conjugate update (Wen, 2014). We demonstrate that this naive approximation recovers state of the art regret bounds in the tabular setting. We also present a series of experiments that highlight the benefit of this form of exploration versus common dithering approaches. These include several didactic examples designed to highlight the need for efficient (and deep) exploration, together with practical evaluation on the video game Tetris as well as a recommendation system model.

7.1 Randomized Least Squares Value Iteration

RLSVI operates in a manner similar to least-squares value iteration (LSVI) and also shares much of the spirit of other closely related approaches TD, LSTD, and SARSA (see, e.g., (Sutton and Barto, 1998; Szepesvári, 2010)). What fundamentally distinguishes RLSVI is that the algorithm explores through randomly sampling statistically plausible value functions, whereas the aforementioned alternatives are typically applied in conjunction with action-dithering schemes such as Boltzmann or ϵ-greedy exploration, which lead to highly inefficient learning.

We consider an agent learning to optimize an unknown MDP $M^* = (\mathcal{S}, \mathcal{A}, R, P, H, \rho)$. Our problem formulation follows Section 2.1. The agent will employ use a linear value function to model that, for each h the optimal Q-value function,

$$Q_h^* \in \text{span}\left[\Phi_h\right] \text{ for some } \Phi_h \in \mathbb{R}^{SA \times D}. \tag{7.1}$$

We refer this matrix Φ_h as a *generalization matrix* and use $\Phi_h(s,a)$ to denote the row of matrix Φ_h associated with state-action pair (s,a). For $d = 1, 2, \cdots, D$, we write the d^{th} column of Φ_h as ϕ_{hd} and refer to ϕ_{hd} as a basis function. We refer to contexts where the agent's belief is correct as *coherent learning*, and refer the alternative as *agnostic learning*.

We now provide an informal explanation of the model in (7.1). In a large MDP a tabular representation of $Q_h(s,a)$ will require at minimum SAH separate values. As we outline in Part II this will quickly become statistically and computationally intractable. In this chapter we consider a setting where the generalization matrix provides *features* $\Phi_h(s,a)$ which capture generalization across states and actions. In particular, equation (7.1) models that the value at (s,a) can be modeled by some linear combination of these *features* $Q_h^*(s,a) \simeq \theta_h^* \Phi_h(s,a)$. Instead of learning each value independently, the problem of learning Q^* given this model reduces to estimating the linear parameters θ_h^*. For settings with $D \ll SA$ this can provide significant statistical and computational advantages.

7.1.1 Least Squares Value Iteration

Before we describe RLSVI, it will first be helpful to introduce *least squares value iteration* (LSVI), which shares much of the spirit of other closely related approaches such as TD,

LSTD, and SARSA (see, e.g., (Sutton and Barto, 1998; Szepesvári, 2010)). LSVI approximates the optimal value function by a linear regression to minimize the Bellman error. To specify a reinforcement learning algorithm we must specify how an agent selects *actions* based upon this value function. Two common approaches are ϵ-greedy and Boltzmann exploration, which we outline below.

Algorithm 7.1 Least-Squares Value Iteration (LSVI)

Input: Data $\Phi_1(s_{k1}, a_{k1}), r_{k1}, .., \Phi_H(s_{kH}, a_{kH}), r_{kH} : k < K$, Parameters $\lambda > 0,\ \sigma > 0$
Output: $\tilde{\theta}_{k1}, .., \tilde{\theta}_{k,H}$

1: **for** timestep $h = H, .., 1$ **do**
2: Generate regression problem $A \in \mathbb{R}^{k \times D}$, $b \in \mathbb{R}^k$:

$$A \leftarrow \begin{bmatrix} \Phi_h(s_{1h}, a_{1h}) \\ \vdots \\ \Phi_h(s_{kh}, a_{kh}) \end{bmatrix}, \quad b_k \leftarrow \begin{cases} r_{kh} + \max_\alpha \left(\Phi_{h+1}\tilde{\theta}_{l,h+1}\right)(s_{k,h+1}, \alpha) & \text{if } h < H \\ r_{kh} & \text{if } h = H \end{cases}$$

3: Linear regression for the value function

$$\tilde{\theta}_{kh} \leftarrow \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}A^\top A + \lambda I\right)^{-1} A^\top b$$

4: **end for**

Algorithm 7.2
LSVI with ϵ-greedy exploration

Input: Features $\Phi_1, .., \Phi_H$;
$\sigma > 0,\ \lambda > 0,\ \epsilon > 0$

1: **for** episode $k = 1, 2, ..$ **do**
2: Compute $\tilde{\theta}_{k1}, .., \tilde{\theta}_{k,H}$ using LSVI
3: **for** $h = 1, .., H$ **do**
4: Sample $w_h \sim \text{Unif}([0, 1])$
5: **if** $w_h < \epsilon$ **then**
6: Sample $a_{kh} \sim \text{Unif}(\mathcal{A})$
7: **else**
8: $a_{kh} \in \arg\max_{\alpha \in \mathcal{A}} \left(\Phi_h\tilde{\theta}_{kh}\right)(s_{kh}, \alpha)$
9: **end if**
10: Observe r_{kh} and $s_{k,h+1}$
11: **end for**
12: **end for**

Algorithm 7.3
LSVI with Boltzmann exploration

Input: Features $\Phi_1, .., \Phi_H$;
$\sigma > 0,\ \lambda > 0,\ \eta > 0$

1: **for** episode $k = 1, 2, ..$ **do**
2: Compute $\tilde{\theta}_{k1}, .., \tilde{\theta}_{k,H}$ using LSVI
3: **for** $h = 1, .., H$ **do**
4: Sample

$$a_{kh} \propto \exp\left[\left(\Phi_h\tilde{\theta}_{kh}\right)(s_{kh}, \alpha)\right]$$

5: Observe r_{kh} and $s_{k,h+1}$
6: **end for**
7: **end for**

Algorithms 7.2 and 7.3 are natural adaptations of Algorithms 2.2 and 2.3 to linearly parameterized value functions. Even though linear value functions across an informative basis can lead to more efficient learning, in general these dithering approaches to exploration can take exponentially long to learn. In fact, Example 2.1 also applies to Algorithms 7.2 and 7.3 *any* choice of basis function. LSVI with either ϵ-greedy and Boltzmann exploration will take $\Omega(2^S)$ timesteps to learn the optimal policy (Osband et al., 2014). In order to guarantee efficient learning we need to have directed (and deep) exploration.

7.1.2 RLSVI algorithm

The manner in which RLSVI explores is inspired by PSRL. PSRL maintains a posterior distribution over MDPs which, through MDP planning, induces a posterior distribution over value functions (Osband et al., 2013). The computational problem is that exact MDP planning does not scale well to large environments. RLSVI sidesteps the intractable planning step by instead building up an approximate posterior distribution for the value function. To do this, RLSVI replaces the linear regression step of LSVI with a *Bayesian* linear regression as if the one step Bellman error were IID Gaussian. This model will not typically be correct, but this approximation admits a simple and tractable posterior update for the linear value function Q_h^*. RLSVI then uses this computationally tractable approximate posterior to drive deep exploration via randomized value functions.

Algorithm 7.4 Randomized Least-Squares Value Iteration

Input: Data $\Phi_1(s_{k1}, a_{k1}), r_{k1}, .., \Phi_H(s_{kH}, a_{kH}), r_{kH} : k < K$, Parameters $\lambda > 0,\ \sigma > 0$
Output: $\tilde{\theta}_{k1}, .., \tilde{\theta}_{k,H}$

1: **for** timestep $h = H, .., 1$ **do**
2: Generate regression problem $A \in \mathbb{R}^{k \times D}$, $b \in \mathbb{R}^k$:

$$A \leftarrow \begin{bmatrix} \Phi_h(s_{1h}, a_{1h}) \\ \vdots \\ \Phi_h(s_{kh}, a_{kh}) \end{bmatrix}, \quad b_k \leftarrow \begin{cases} r_{kh} + \max_\alpha \left(\Phi_{h+1}\tilde{\theta}_{l,h+1}\right)(s_{k,h+1}, \alpha) & \text{if } h < H \\ r_{kh} & \text{if } h = H \end{cases}$$

3: Bayesian linear regression for the value function

$$\overline{\theta}_{kh} \leftarrow \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}A^\top A + \lambda I\right)^{-1} A^\top b, \quad \Sigma_{kh} \leftarrow \left(\frac{1}{\sigma^2}A^\top A + \lambda I\right)^{-1}$$

4: Sample $\tilde{\theta}_{kh} \sim N(\overline{\theta}_{kh}, \Sigma_{kh})$ from Gaussian posterior
5: **end for**

Algorithm 7.5 RLSVI with greedy action

Input: Features $\Phi_1, .., \Phi_H$; $\sigma > 0,\ \lambda > 0$

for episode $k = 1, 2, ..$ **do**
  Compute $\tilde{\theta}_{k1}, .., \tilde{\theta}_{k,H}$ using Algorithm 7.4
  for $h = 1, .., H$ **do**
    Sample $a_{kh} \in \arg\max_{\alpha \in \mathcal{A}} \left(\Phi_h \tilde{\theta}_{kh}\right)(s_{kh}, \alpha)$
    Observe r_{kh} and $s_{k,h+1}$
  end for
end for

At an intuitive level the key difference between LSVI and RLSVI is the incorporation of uncertainty estimates. LSVI performs a linear regression for the Bellman error at each timestep. LSVI then uses its best estimate for the value at time $h+1$ to estimate values at time h. The resultant value estimate from LSVI has no notion of uncertainty, so algorithms based upon this value have no way to direct efficient exploration. RLSVI also performs a linear regression for the Bellman error, but the algorithm uses a cheap Gaussian uncertainty estimate for the resultant value function. The key observation is that even though this uncertainty estimate will not typically be the correct Bayesian distribution, it may still be a useful approximation to this uncertainty. At each timestep, RLSVI updates the values at time h based upon a *sample* from this distribution so that the resultant value estimate is a single *sample* from an approximate posterior. We can then use this distribution to assess the potential exploration exploitation tradeoff.

7.2 Provably efficient tabular learning

RLSVI is an algorithm designed for efficient exploration in large MDPs with linear value function generalization. So far, there are no algorithms with analytical regret bounds in this setting. In fact, most common methods are provably *inefficient* regardless of the choice of basis function (see Example 2.1). Theorem 7.1 establishes a bound on the expected regret for RLSVI in a tabular setting without generalization where the basis functions $\Phi_h = I$. At a high level, we rely upon Lemma 3.3 which shows that, under certain conditions, this noise will result in a value estimate which is stochastically optimistic for the true value function. We show that these optimistic samples will converge to the true optimal value as more data is gathered.

7.2.1 Problem formulation

We establish the result of Theorem 7.1 for a slightly different setting than Section 2.1 As before we consider the problem of learning to optimize an unknown MDP $M = (\mathcal{S}, \mathcal{A}, P^M, R^M, H, \rho)$. However, for each timestep h within an episode the state is s_h and an action a_h is selected then a next state s_{h+1} is sampled from $P^M(\cdot|s_h, a_h, h)$ and a reward r_h is sampled from $R^M(\cdot|s_h, a_h, h)$. The key difference here from our previous treatment is that both the rewards and the transitions may be time-inhomogeneous through the episode. This formulation is in some sense more general than Section 2.1, since it includes all time-homogeneous MDPs as a special case. However, this setting is generally more easy to analyze, since we do not need to worry about dependency between states and actions throughout an episode.

We will now introduce a family of MDPs with a simple prior structure that we will use for our analysis in Theorem 7.1. We write $\Psi_{\text{Dir}}^{\{-1,0\}}$ for the family of priors over MDPs that have rewards drawn from $\{-1, 0\}$, independent Dirichlet priors $\alpha^R(s, a) \in \mathbb{R}_+^2$ for the reward function and independent Dirichlet prior on transitions $\alpha^R(s, a) \in \mathbb{R}_+^S$[1]. The proof Theorem 7.1 is remarkably straightforward given our previous results for PSRL. For generalization matrix $\Phi_h = I$ RLSVI learns a tabular representation for each (s, a) independently at each timestep. The resultant algorithm is almost identical to PSRL, but with Gaussian noise in place of the true posterior for transition dynamics. We use Lemma 3.3 to establish that RLSVI generates value estimates which are stochastically optimistic for the true posterior resulting from $\Psi_{\text{Dir}}^{\{-1,0\}}$ and any data.

Theorem 7.1 (RLSVI satisfies regret bounds in the tabular setting)**.** *Let $\phi \in \Psi_{\text{Dir}}^{\{-1,0\}}$ be the distribution of M^*. For all $\lambda = O(SA)$ and $\sigma = O(H)$ such that $\lambda \geq \max_{(s,a,h)} \left(\mathbb{1}^T \alpha^R(s, a, h) + \mathbb{1}^T \alpha^P(s, a, h) \right)$ and $\sigma \geq \sqrt{H^2 + 1}$, then the expected regret of RLSVI executed with $\Phi_h = I$ for $h = 1, .., H$ with parameters λ and σ is bounded,*

$$\mathbb{E}\left[\text{Regret}(T, \pi^{\text{RLSVI}}, M^*)\right] \leq \tilde{O}\left(\sqrt{H^3 SAT}\right). \tag{7.2}$$

[1]Shifting the rewards to lie within $[-1, 0]$ rather than the typical $[0, 1]$ is important for our analysis of RLSVI, which introduces λ-shrinkage towards 0. The Dirichlet structure of the reward distributions is not important, and can easily be extended to arbitrary support over $[-1, 0]$ (Agrawal and Goyal, 2012a).

Proof of Theorem 7.1

We follow the standard approach for analysis of optimistic algorithms (3.1), which is to add and subtract the imagined optimal value function in each episode k,

$$\Delta_k = \underbrace{V^k_{k,1} - V^*_{k,1}}_{\Delta^{\text{conc}}_k} + \underbrace{V^*_{*,1} - V^k_{k,1}}_{\Delta^{\text{opt}}_k}.$$

Lemma 7.1 (RLSVI generates optimistic Q-values)**.**
Conditional on any data $\mathcal{H}$, the Q-values generated by RLSVI with $\sigma \geq \sqrt{H^2+1}$ and $\lambda \geq \max_{(s,a,h)} \left(\mathbb{1}^T \alpha^R(s,a,h) + \mathbb{1}^T \alpha^P(s,a,h) \right)$ are stochastically optimistic,

$$\tilde{Q}^k_h(s,a) \succcurlyeq_{\text{so}} Q^*_h(s,a) \text{ for all } s,a,h. \tag{7.3}$$

Proof. Fix any data $\mathcal{H}_k$ available and use backwards induction on $h = H-1,..,1$. For any (s,a,h) we write $n(s,a,h)$ for the amount of visits to that datapoint in $\mathcal{H}_k$. We will write $\hat{R}(s,a,h), \hat{P}(s,a,h)$ for the empirical mean reward and mean transitions based upon the data $\mathcal{H}_k$. We can now write the posterior mean rewards and transitions:

$$\overline{R}(s,a,h)|\mathcal{H}_k = \frac{-1 \times \alpha^R_1(s,a,h) + n(s,a,h)\hat{R}(s,a,h)}{\mathbb{1}^T \alpha^R(s,a,h) + n(s,a,h)}$$

$$\overline{P}(s,a,h)|\mathcal{H}_k = \frac{\alpha^P(s,a,h) + n(s,a,h)\hat{P}(s,a,h)}{\mathbb{1}^T \alpha^P(s,a,h) + n(s,a,h)}$$

Now, using $\Phi_h = I$ for all (s,a,h) we can write the RLSVI updates in similar form. Note that, Σ_{kh} is diagonal with each diagonal entry equal to $\sigma^2/(n(s,a,h) + \lambda\sigma^2)$. In the case of $h = H-1$

$$\overline{\theta}^l_{H-1}(s,a) = \frac{n(s,a,H-1)\hat{R}(s,a,H-1)}{n(s,a,H-1) + \lambda\sigma^2}$$

Using the relation that $\hat{R} \geq \overline{R}$ Lemma 3.3 means that

$$N(\overline{\theta}^l_{H-1}(s,a), \frac{1}{n(s,a,h) + \mathbb{1}^T \alpha^R(s,a,h)}) \succcurlyeq_{\text{so}} R_{H-1}|\mathcal{H}_l.$$

Therefore, choosing $\lambda > \max_{s,a,h} \mathbb{1}^T \alpha^R(s,a,h)$ and $\sigma > 1$, we must satisfy the lemma for all s,a and $h = H-1$.

For the inductive step we assume that the result holds for all s, a and $j > h$, we now want to prove the result for all (s, a) at timestep h. Once again, we can express $\overline{\theta}^l_h(s, a)$ in closed form.

$$\overline{\theta}^l_h(s,a) = \frac{n(s,a,h)\left(\hat{R}(s,a,h) + \hat{P}(s,a,h)^T \tilde{V}^l_{h+1}\right)}{n(s,a,h) + \lambda\sigma^2}$$

To simplify notation we omit the arguments (s, a, h) where they should be obvious from context. The posterior mean estimate for the next step value V^*_h, conditional on $\mathcal{H}_l$:

$$\mathbb{E}[Q^*_h(s,a)|\mathcal{H}_l] = \overline{R} + \overline{P}^T V^*_{h+1} \leq \frac{n(\hat{R} + \hat{P}^T V^*_{h+1})}{n + \lambda\sigma^2}.$$

As long as $\lambda > \mathbb{1}^T\alpha^R + \mathbb{1}^T(\alpha^P)$ and $\sigma^2 > H^2$. By our induction process $\tilde{V}^l_{h+1} \succcurlyeq_{\text{so}} V^*_{h+1}$ so that

$$\mathbb{E}[Q^*_h(s,a)|\mathcal{H}_l] \leq \mathbb{E}\left[\frac{n(\hat{R} + \hat{P}^T \tilde{V}^l_{h+1})}{n + \lambda\sigma^2} \mid \mathcal{H}_l\right].$$

We can conclude through another application of 3.3: the noise from rewards is dominated by $N(0, 1)$ and the noise from transitions is dominated by $N(0, H^2)$. We know that $\sigma^2 \geq H^2 + 1$ and so this completes our result of stochastic optimism. □

Lemma 7.1 means RLSVI generates stochastically optimistic Q-values for any history $\mathcal{H}_{k1}$, so that $\mathbb{E}[\Delta^{\text{opt}}_k|\mathcal{H}_k] \leq 0$. All that remains is to prove the remaining estimates $\mathbb{E}[\Delta^{\text{conc}}_l|\mathcal{H}_{k1}]$ concentrate around the true values with data. Intuitively this should be clear, since the size of the Gaussian perturbations decreases as more data is gathered. In the remainder of this section we will sketch this result. We write the concentration error $\Delta^{\text{conc}}_k = V^k_{k1}(s_{k1}) - V^*_{k1}(s_{k1})$. We decompose the value estimate V^k_{kh} explicitly:

$$\begin{aligned} V^k_{kh}(s_{kh}) &= \frac{n(\hat{R} + \hat{P}^T V^k_{k,h+1} h + 1)}{n + \lambda\sigma^2} + w^\sigma_{kh} \\ &= \overline{R}_k + \overline{P}^T_k V^k_{k,h+1} + b^R_{kh} + b^P_{kh} + w^\sigma_{kh} \end{aligned} \tag{7.4}$$

where w^σ_{kh} is the Gaussian noise from RLSVI and $b^R_{kh} = b^R(s_{kh}, a_{kh}, h), b^P_{kh} = b^P(s_{kh}, a_{kh}, h)$ are optimistic bias terms for RLSVI. These terms emerge since RLSVI shrinks estimates towards zero rather than the Dirichlet prior for rewards and transitions. In the case of tabular basis functions this is the only difference between RLSVI and Gaussian PSRL.

We can reproduce the arguments of Bellman decomposition as per Equation 3.10 to bound the concentration,

$$\mathbb{E}[\Delta_k^{\text{conc}}|\mathcal{H}_{k1}] = \sum_{h=1}^{H} \left\{ b^R(s_{kh}, a_{kh}, h) + b^P(s_{kh}, a_{kh}, h) + w_{kh}^{\sigma} \right\}. \tag{7.5}$$

We can even write explicit expressions for b^R, b^P and w^{σ}.

$$b^R(s,a,h) = \frac{n\hat{R}}{n+\lambda\sigma^2} - \frac{n\hat{R} - \alpha_1^R}{n + \mathbb{1}^T\alpha^R} = \hat{R}\left(\frac{\mathbb{1}^T\alpha^R}{n+\mathbb{1}^T\alpha^R} - \frac{\lambda\sigma^2}{n+\lambda\sigma^2}\right) + \frac{\alpha_1^R}{n+\mathbb{1}^T\alpha^R}$$

$$b^P(s,a,h) = \frac{n\hat{P}^T\tilde{V}_{h+1}^k}{n+\lambda\sigma^2} - \frac{(n\hat{P}+\alpha^P)^T\tilde{V}_{h+1}^k}{n+\mathbb{1}^T\alpha^P} = \hat{P}^T\tilde{V}_{h+1}^k\left(\frac{\mathbb{1}^T\alpha^P}{n+\mathbb{1}^T\alpha^P} - \frac{\lambda\sigma^2}{n+\lambda\sigma^2}\right) + \frac{{\alpha^P}^T\tilde{V}_{h+1}^k}{n+\mathbb{1}^T\alpha^P}$$

$$w_{kh}^{\sigma} \sim N\left(0, \frac{\sigma^2}{n+\lambda\sigma^2}\right)$$

where $n = n_k(s,a,h)$ is the number of visits to (s,a,h) before episode k. The final details for this proof are technical but the argument is simple. We show that the contributions from b^R, b^P, w^{σ} cannot be too large as the number of visits to each state and action $n = n(s,a,h)$ grows. At a high level we can see that $b^R, b^P = O\left(\frac{1}{n}\right)$ and $w_{kh}^{\sigma} = O\left(\sqrt{\frac{1}{n}}\right)$. We make this argument more precise below.

Let $\mathcal{E}$ be the event that $\max_{h,s,a}\left(w_{kh}^{\sigma}(s,a) - \sigma\sqrt{\frac{4\log(SAHT)}{n(s,a,h)+\lambda}}\right) > 0$. It follows from Lemma 3.2 and a union bound over all SAH that $\mathbb{P}(\mathcal{E}) \leq \frac{1}{2SAHT^2}$. Even when the event $\mathcal{E}$ does hold, the resultant contribution from tail w_{kh}^{σ} cannot be too large.

Lemma 7.2. *For $X \sim N(0,1)$ and any $\lambda > 1$: $\mathbb{E}\left[X \mid X > \lambda\right] \leq \lambda + 1$.*

Proof. We bound the Gaussian CDF Φ in terms of its PDF ϕ:

$$\mathbb{E}\left[X \mid X > \lambda\right] = \frac{\phi(\lambda)}{1-\Phi(\lambda)} \leq \frac{\phi(\lambda)}{\frac{\lambda}{\lambda^2+1}\phi(\lambda)} \leq \lambda + 1.$$

□

We can use Lemma 7.2 in (7.5) to bound the estimation error under event $\mathcal{E}$:

$$\sum_{h,s,a} \mathbb{E}[w_{lh}(s,a) \mid \mathcal{H}_{k1}, \mathcal{E}] \leq \sum_{h,s,a} \frac{\sqrt{4\log(SAHT)}+1}{\sqrt{n_{lh}(s,a)+S}} \leq SAH\left(2\sqrt{\log(SAHT)}+1\right).$$

and so,

$$\begin{aligned}\mathbb{E}\left[\sum_{h=1}^{H} w_{kh}^{\sigma} \mid \mathcal{H}_{k1}\right] &\leq \mathbb{P}(\mathcal{E}) \times \mathbb{E}\left[\sum_{s,a,h} w_{kh}^{\sigma} \mid \mathcal{H}_{k1}, \mathcal{E}\right] + \mathbb{E}\left[\sum_{h=1}^{H} w_{kh}^{\sigma} \mid \mathcal{H}_{k1}, \mathcal{E}^{c}\right] \\ &\leq \frac{2SAH\sigma\sqrt{\log(SAHT)}+1}{2SAHT^2} + \sigma\sum_{h=1}^{H}\sqrt{\frac{4\log(SAHT)}{n(s,a,h)+\lambda}}. \qquad (7.6)\end{aligned}$$

We can complete the proof using a standard summation argument together with the expressions for b^R, b^P, w^σ above,

$$\begin{aligned}\mathbb{E}\left[\text{Regret}(T, \pi^{\text{RLSVI}}, M^*)\right] &= \sum_{k=1}^{\lceil T/H\rceil} \mathbb{E}\left[\mathbb{E}\left[\Delta_k^{\text{opt}} + \Delta_k^{\text{conc}} \mid \mathcal{H}_{k1}\right]\right] \\ &\leq \sum_{k=1}^{\lceil T/H\rceil} \mathbb{E}\left[\sum_{h=1}^{h} b^R(s_{kh}, a_{kh}, h) + b^P(s_{kh}, a_{kh}, h) + w_{kh}^{\sigma} \mid \mathcal{H}_{k1}\right] \\ &\leq \sum_{k=1}^{\lceil T/H\rceil} \left\{\mathbb{E}\left[\sum_{h=1}^{h} \frac{\lambda\sigma^2}{n+\lambda\sigma^2}(1+H) + \sigma\sqrt{\frac{4\log(SAHT)}{n(s,a,h)+\lambda}} \mid \mathcal{H}_{k1}\right]\right. \\ &\qquad \left. + \frac{\sigma\sqrt{\log(SAHT)}+1}{T^2}\right\}.\end{aligned}$$

We complete the proof using a standard integral comparison for summations: $\sum_{t=1}^{T}\frac{1}{t} = O(\log(T))$ and $\sum_{t=1}^{T}\sqrt{\frac{1}{t}} = O(\sqrt{T})$ together with a pigeonhole argument for the visits to each state and action. This shows that,

$$\mathbb{E}\left[\text{Regret}(T, \pi^{\text{RLSVI}}, M^*)\right] = O\left(H\lambda\sigma^2\log(T) + \sigma\sqrt{SAHT\log(SAHT)} + \frac{\sigma H\sqrt{\log(SAHT)}}{T}\right). \qquad (7.7)$$

We now make use of the the requirement in Theorem 7.1 that $\sigma \geq \sqrt{H^2+1}$ and $\lambda \geq \max_{(s,a,h)}\left(\mathbb{1}^T\alpha^R(s,a,h) + \mathbb{1}^T\alpha^P(s,a,h)\right)$; at the same time, we know that asymptotically these parameters are bounded $\lambda = O(SA)$ and $\sigma = O(H)$. As a result , the dominant term in (7.7) comes from $\sigma\sqrt{SAHT\log(SAHT)}$ which is ultimately $\tilde{O}(\sqrt{H^3SAT})$. This completes the proof of Theorem 7.1. □

7.2.2 Provably efficient learning with generalization

Theorem 7.1 shows that RLSVI with tabular basis functions acts as an effective Gaussian approximation to PSRL. This demonstrates a clear distinction between exploration via randomized value functions and dithering strategies such as Example 2.1. However, the motivating for RLSVI is not for tabular environments, where several provably efficient RL algorithms already exist, but instead for large systems that require generalization.

We believe that, under some conditions, it may be possible to establish polynomial regret bounds for RLSVI with value function generalization. To stimulate thinking on this topic we present a conjecture of result what may be possible in Conjecture 7.1.

Conjecture 7.1. *For all* $M^* = (\mathcal{S}, \mathcal{A}, P, R, H, \rho,)$, $\Phi_0, \ldots, \Phi_{H-1}$, σ, *and* λ, *if reward distributions* R *have support* $[-\sigma, \sigma]$, *there is a unique* $(\theta_0, \ldots, \theta_{H-1}) \in \mathbb{R}^{K \times H}$ *satisfying* $Q_h^* = \Phi_h \theta_h$ *for* $h = 0, \ldots, H-1$, *and* $\sum_{h=0}^{H-1} \|\theta_h\|^2 \leq \frac{KH}{\lambda}$, *then there exists a polynomial* poly *such that*

$$\mathbb{E}\left[\text{Regret}(T, \pi^{\text{RLSVI}}, M^*)\right] \leq \sqrt{T}\ \text{poly}\left(K, H, \max_{h,x,a} \|\Phi_h(x,a)\|, \sigma, \lambda^{-1}\right).$$

For now, we will present a series of experiments designed to test the applicability and scalability of RLSVI for exploration with generalization. Our experiments are divided into three sections. First, we present a series of didactic chain environments similar to Example 2.1. We show that RLSVI can effectively synthesize exploration with generalization with both coherent and agnostic value functions that are intractable under any dithering scheme. Next, we apply our Algorithm to learning to play Tetris. We demonstrate that RLSVI leads to faster learning, improved stability and a superior learned policy in a large-scale video game. Finally, we consider a business application with a simple model for a recommendation system. We show that an RL algorithm can improve upon even the optimal myopic bandit strategy. RLSVI learns this optimal strategy when dithering strategies do not.

7.3 Testing for efficient exploration

We now consider a series of environments modeled on Example 2.1, where dithering strategies for exploration are provably inefficient. Importantly, and unlike the tabular experiments in Section 4.3, our algorithm will only interact with the MDP but through a set of basis function Φ which generalize across states. We examine the empirical performance of RLSVI and find that it does efficiently balance exploration and generalization in this didactic example.

7.3.1 Coherent learning

In our first experiments, we generate a random set of D basis functions. This basis is coherent but the individual basis functions are not otherwise informative. We form a random linear subspace V_{hD} spanned by $(\mathbb{1}, Q_h^*, \tilde{w}_1, .., \tilde{w}_{D-2})$. Here w_i and $\tilde{w}_i$ are IID Gaussian $\sim N(0, I) \in \mathbb{R}^{SA}$. We then form Φ_h by projecting $(\mathbb{1}, w_1, .., w_{k-1})$ onto V_{hD} and renormalize each component to have equal 2-norm. For more details on the practicalities of these experiments see (Osband et al., 2014). Figure 8.5 presents the empirical regret for RLSVI with $D = 10, N = 50, \sigma = 0.1, \lambda = 1$ and an ϵ-greedy agent over 5 seeds[2].

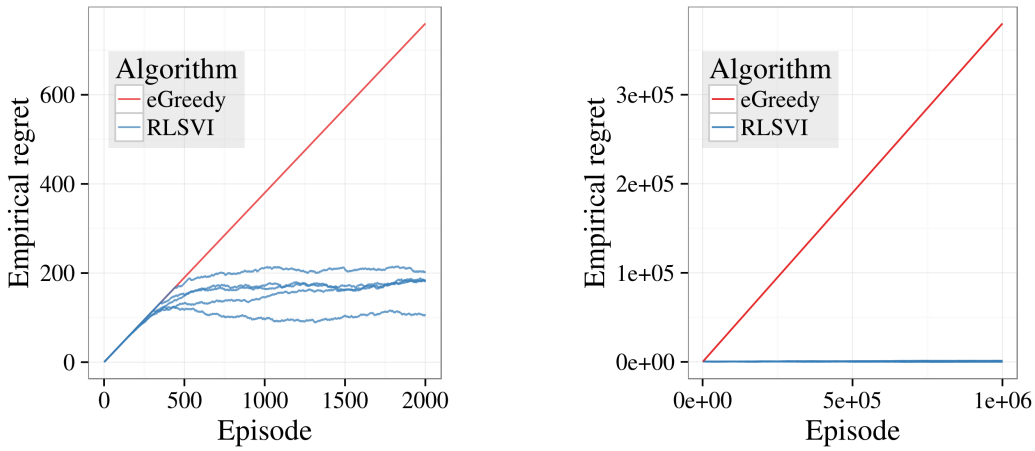


Figure 7.1: Efficient exploration on Example 2.1 with a length 50 chain.

[2]In this setting any choice of ϵ or Boltzmann η is equivalent.

Figure 7.1 shows that RLSVI consistently learns the optimal policy in roughly 500 episodes. Any dithering strategy would take at least 10^{15} episodes for this result. The state of the art upper bounds for the efficient optimistic algorithm UCRL given by appendix C.5 in (Dann and Brunskill, 2015) for $H = 15, S = 6, A = 2, \epsilon = 1, \delta = 1$ only kick in after more than 10^{10} suboptimal episodes. RLSVI is able to effectively exploit the generalization and prior structure from the basis functions to learn much faster.

We now examine how learning scales as we change the chain length N and number of basis functions D. We observe that RLSVI essentially maintains the optimal policy once it discovers the rewarding state. We use the number of episodes until 10 rewards as a proxy for learning time. We report the average of five random seeds.

Figure 7.2 examines the time to learn as we vary the chain length N with fixed $K = 10$ basis functions. We include the dithering lower bound 2^{N-1} as a dashed line and a lower bound scaling $\frac{1}{10}H^2SA$ for tabular learning algorithms as a solid line (Dann and Brunskill, 2015). For $N = 100$, $2^{N-1} > 10^{28}$ and $H^2SA > 10^6$. RLSVI demonstrates scalable generalization and exploration to outperform these bounds.

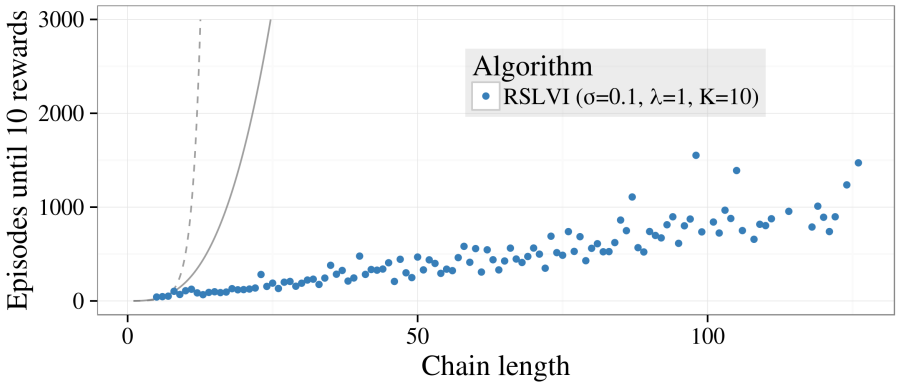


Figure 7.2: RLSVI learning time against chain length.

Figure 7.3 examines the time to learn as we vary the basis functions K in a fixed $N = 50$ length chain. Learning time scales gracefully with K. Further, the marginal effect of K decrease as $\dim(V_{hK}) = K$ approaches $\dim(\mathbb{R}^{SA}) = 100$. We include a local polynomial regression in blue to highlight this trend. Importantly, even for large K the performance is far superior to the dithering and tabular bounds[3].

[3]For chain $N = 50$, the bounds $2^{N-1} > 10^{14}$ and $H^2SA > 10^5$.

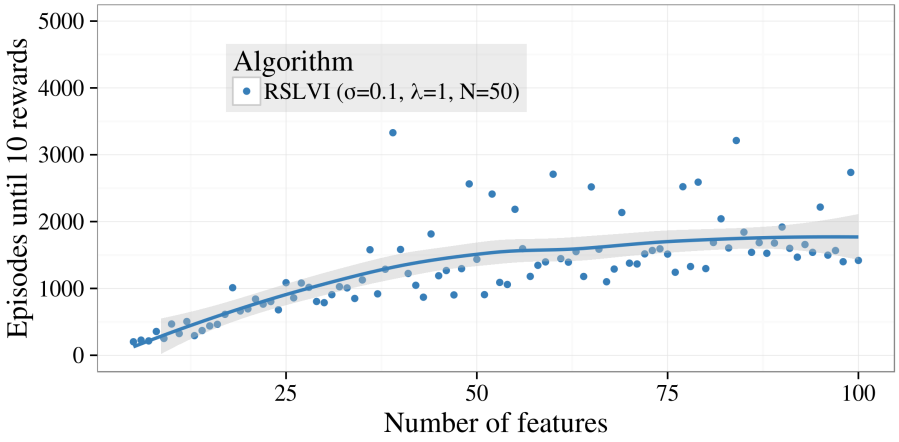


Figure 7.3: RLSVI learning time against number of basis features.

Figure 7.4 examines these same scalings on a logarithmic scale. We find the data for these experiments is consistent with polynomial learning as hypothesized in Conjecture 7.1. These results are remarkably robust over several orders of magnitude in both σ and λ. We present more detailed analysis of these sensitivities in (Osband et al., 2014).

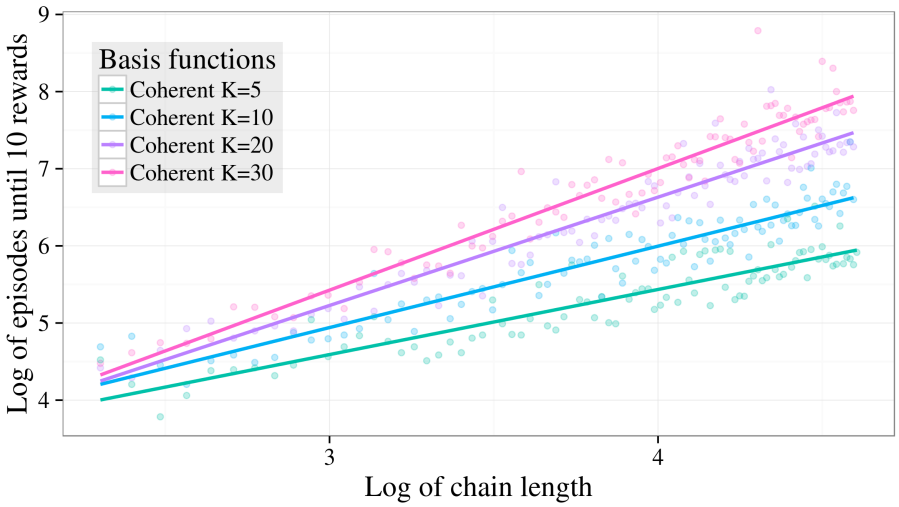


Figure 7.4: Empirical support for polynomial learning in RLSVI.

7.3.2 Agnostic learning

Unlike the examples in Section 7.3.1, practical RL problems will typically be agnostic; the true value function Q_h^* will not lie within V_{hK}. To examine RLSVI in this setting we generate basis functions by adding Gaussian noise to the true value function $\phi_{hk} \sim N(Q_h^*, \rho I)$. The parameter ρ determines the scale of this noise. For $\rho = 0$ this problem is coherent but for $\rho > 0$ this will typically not be the case. We fix $N = 20, K = 20, \sigma = 0.1$ and $\lambda = 1$.

For $i = 0, .., 1000$ we run RLSVI for 10,000 episodes with $\rho = i/1000$ and a random seed. Figure 7.5 presents the number of episodes until 10 rewards for each value of ρ. For large values of ρ, and an extremely misspecified basis, RLSVI is not effective. However, there is some region $0 < \rho < \rho^*$ where learning remains remarkably stable[4].

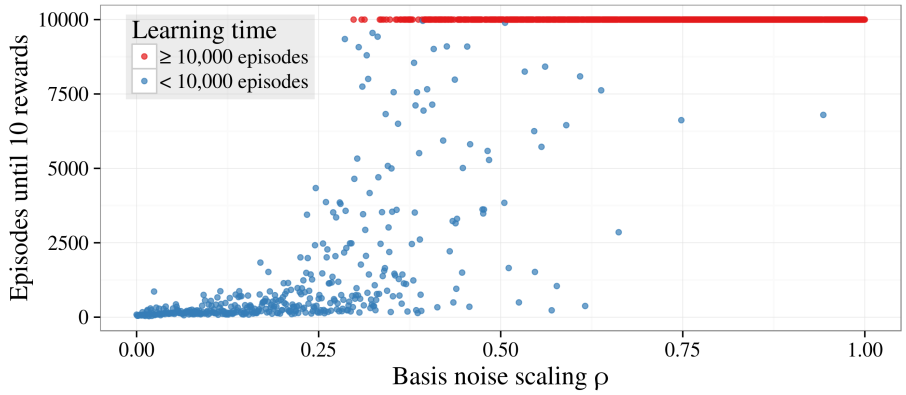


Figure 7.5: RLSVI is somewhat robust model misspecification.

Unlike many existing methods for exploration with generalization (Wen and Van Roy, 2013), RLSVI's performance degrades smoothly as the problem transitions from the agnostic to the coherent setting. This simple example gives us some hope that RLSVI can be useful in the agnostic setting, so long as the basis functions are somewhat reasonable for the problem at hand. In our remaining experiments we will demonstrate that RLSVI can achieve state of the art results in more practical problems even when the basis functions do not span the true value function.

[4]Note $Q_h^*(s, a) \in \{0, 1\}$ so $\rho = 0.5$ represents significant noise.

7.4 Learning to play Tetris

We now turn our attention to learning to play the iconic video game Tetris. In this game, random blocks fall sequentially on a 2D grid with 20 rows and 10 columns. At each step the agent can move and rotate the object subject to the constraints of the grid. The game starts with an empty grid and ends when a square in the top row becomes full. However, when a row becomes full it is removed and all bricks above it move downward. The objective is to maximize the score attained (total number of rows removed) before the end of the game.

Tetris has been something of a benchmark problem for RL and approximate dynamic programming, with several papers on this topic (Gabillon et al., 2013; Farias and Van Roy, 2006; Szita and Lörincz, 2006). Our focus is not so much to learn a high-scoring Tetris player, but instead to demonstrate the RLSVI offers benefits over other forms of exploration with LSVI. Tetris is challenging for RL with a huge state space of more than 2^{200} states. In order to tackle this problem efficiently we use 22 benchmark features (Bertsekas and Ioffe, 1996) These give the height of each column, the absolute difference in height of each column, the maximum height of a column, the number of "holes" and a constant. It is well known that you can find far superior linear basis functions (Gabillon et al., 2013), but we use these benchmark features to mirror earlier work.

In order to apply RLSVI to Tetris, which does not have fixed episode length, we made a few natural modifications to the algorithm. First, we approximate a time-homogeneous value function. We also only the keep most recent $N = 10^5$ transitions to cap the linear growth in memory and computational requirements, similar to (Mnih et al., 2015). In Figure 7.6 we present learning curves for RLSVI $\lambda = 1, \sigma = 1$ and LSVI with a tuned ϵ-greedy exploration schedule[5]. The results are significant in several ways.

First, both RLSVI and LSVI make significant improvements over the previous approach of LSPI with the same basis functions (Bertsekas and Ioffe, 1996). Both algorithms reach higher final performance ($\simeq$ 3500 and 4500 respectively) than the best level for LSPI (3183). They also reach this performance after many fewer games and, unlike LSPI do not "collapse" after finding their peak performance. We believe that these improvements are mostly due to the memory replay buffer, which stores a bank of recent past transitions, rather than LSPI which is purely online.

[5]We found that we could not achieve good performance for any fixed ϵ. We used an annealing exploration schedule that was tuned to give good performance.

Second, both RLSVI and LSVI learn from scratch where LSPI required a scoring initial policy to begin learning. We believe this is due to improved exploration schemes, LSPI is completely greedy so struggles to learn without an initial policy. LSVI with a tuned ϵ schedule is much better. However, we do see a significant improvement through exploration via RLSVI even when compared to the tuned ϵ scheme. More details are available in the Appendix of (Osband et al., 2014).

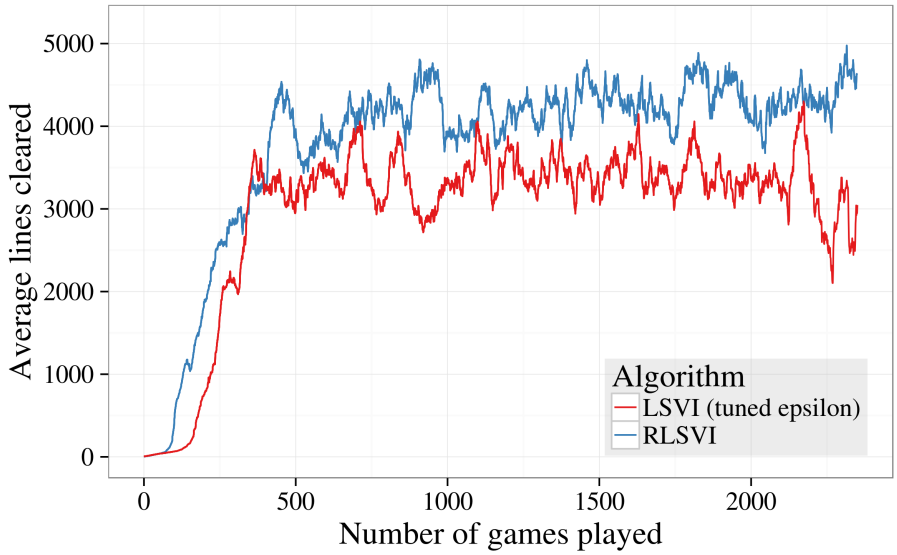


Figure 7.6: Learning to play Tetris with linear Bertsekas features.

In Figure 7.6 we show that RLSVI outperforms LSVI even with a highly tuned annealing scheme for ϵ. However, these results are much more extreme on a didactic version of mini-Tetris. We make a Tetris board with only 4 rows and only S, Z pieces. This problem is much more difficult and highlights the need for efficient exploration in a more extreme way. In Figure 7.7 we present the results for this mini-Tetris environment. As expected, this example highlights the benefits of RLSVI over LSVI with dithering. RLSVI greatly outperforms LSVI even with a tuned ϵ schedule. RLSVI learns faster and reaches a higher convergent policy.

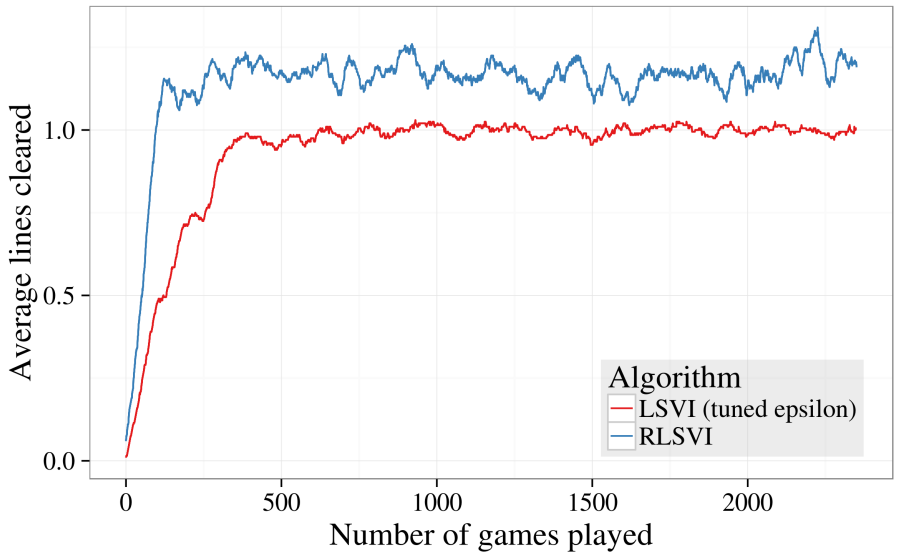


Figure 7.7: Reduced 4-row Tetris with only S and Z pieces.

7.5 A simple recommendation system

We will now show that efficient exploration and generalization can be helpful in a simple model of customer interaction. Consider an agent which recommends $J \leq N$ products from $\mathcal{Z} = \{1, 2, \ldots, N\}$ sequentially to a customer. The conditional probability that the customer likes a product depends on the product, some items are better than others. However it also depends on what the user has observed, what she liked and what she disliked. We represent the products the customer has seen by $\tilde{\mathcal{Z}} \subseteq \mathcal{Z}$. For each product $n \in \tilde{\mathcal{Z}}$ we will indicate $x_n \in \{-1, +1\}$ for her preferences {dislike, like} respectively. If the customer has not observed the product $n \notin \tilde{\mathcal{Z}}$ we will write $x_n = 0$. We model the probability that the customer will like a new product $a \notin \tilde{\mathcal{Z}}$ by a logistic transformation linear in x:

$$\mathbb{P}(a|x) = 1/\left(1 + \exp\left(-\left[\beta_a + \textstyle\sum_n \gamma_{an} x_n\right]\right)\right). \tag{7.8}$$

Importantly, this model reflects that the customers' preferences may evolve as their experiences change. For example, a customer may be much more likely to watch the second season of the TV show "Breaking Bad" if they have watched the first season and liked it.

The agent in this setting is the recommendation system, whose goal is to maximize the cumulative amount of items liked through time for each customer. The agent does not know $p(a|x)$ initially, but can learn to estimate the parameters β, γ through interactions across

different customers. Each customer is modeled as an episode with horizon length $H = J$ with a "cold start" and no previous observed products $\tilde{\mathcal{Z}} = \emptyset$. For our simulations we set $\beta_a = 0 \; \forall a$ and sample a random problem instance by sampling $\gamma_{an} \sim N(0, c^2)$ independently for each a and n.

Although this setting is simple, the number of possible states $|\mathcal{S}| = |\{-1, 0, +1\}|^H = 3^J$ is exponential in J. To learn in time less than $|\mathcal{S}|$ it is crucial that we can exploit generalization between states as per equation (7.8). For this problem we construct the following simple basis functions: $\forall 1 \leq n, m, a \leq N$, let $\phi_m(x, a) = \mathbf{1}\{a = m\}$ and $\phi_{mn}(x, a) = x_n \mathbf{1}\{a = m\}$. In each period h form $\Phi_h = ((\phi_n)_n, (\phi_m)_m)$. The dimension of our function class $K = N^2 + N$ is exponentially smaller than the number of states. However, barring a freak event, this simple basis will lead to an agnostic learning problem.

Figure 7.8 and 7.9 show the performance of RLSVI compared to several benchmark methods. In Figure 7.8 we plot the cumulative regret of RLSVI when compared against LSVI with Boltzmann exploration and identical basis features. We see that RLSVI explores much more efficiently than Boltzmann exploration over a wide range of temperatures. We set $N = 10$, $H = J = 5$, $c = 2$ and $L = 1200$. Note that such problems have $|\mathcal{S}| = 4521$ states; this allows us to solve each MDP exactly so that we can compute regret. Each result is averaged over 100 problem instances and for each problem instance, we repeat simulations 10 times. The cumulative regret for both RLSVI (with $\lambda = 0.2$ and $\sigma^2 = 10^{-3}$) and LSVI with Boltzmann exploration (with $\lambda = 0.2$ and a variety of "temperature" settings η) are plotted in Figure 7.8. RLSVI clearly outperforms LSVI with Boltzmann exploration.

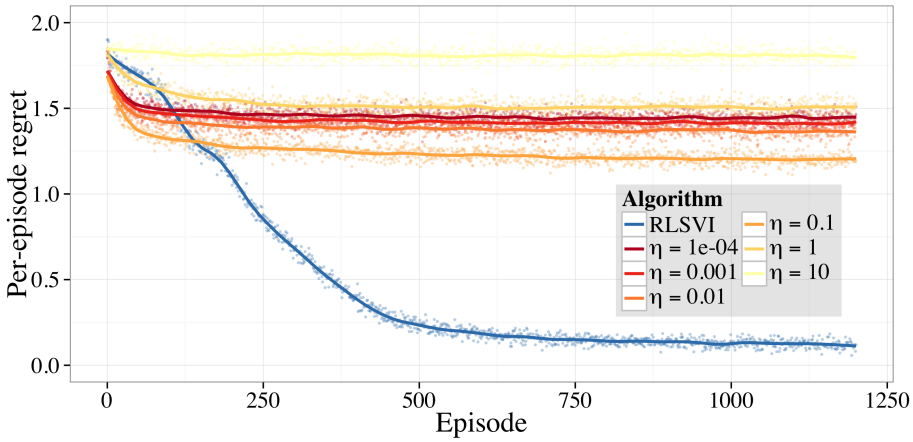


Figure 7.8: RLSVI performs better than Boltzmann exploration.

Figure 7.9 shows that, using this efficient exploration method, the reinforcement learning policy is able to outperform not only benchmark bandit algorithms but even the optimal myopic policy[6]. Bernoulli Thompson sampling does not learn much even after 1200 episodes, since the algorithm does not take *context* into account. The linear contextual bandit outperforms RLSVI at first. This is not surprising, since learning a myopic policy is simpler than a multi-period policy. However as more data is gathered RLSVI eventually learns a richer policy which outperforms the myopic policy.

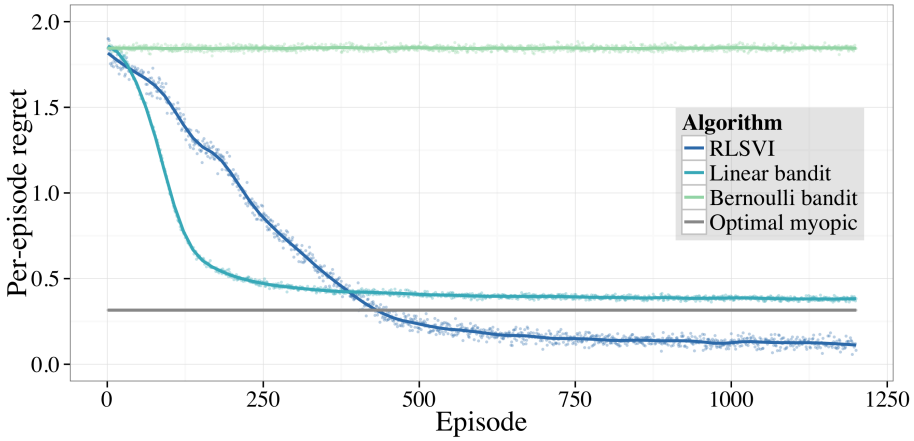


Figure 7.9: RLSVI can outperform the optimal myopic policy.

Our simulations use an extremely simplified model. Nevertheless, they highlight the potential value of RL over multi-armed bandit approaches in recommendation systems and other customer interactions. An RL algorithm may outperform even even an optimal myopic system, particularly where large amounts of data are available. In some important settings, efficient generalization and exploration can be crucial.

[6]The optimal myopic policy knows the true system parameters defined by (7.8), but does not plan over multiple timesteps.

Chapter 8

Nonlinear Value Functions

We have seen that randomized value functions can drive deep exploration even when the randomization scheme is not the correct posterior distribution. RLSVI can combine efficient exploration with generalization over linear basis functions. However, the performance of this algorithm is crucially dependent on the choice of good basis functions. In settings with rich unstructured data it can be difficult to design effective linear features for these basis functions. For these environments nonlinear function approximation and "deep learning" have emerged as the state of the art methods for learning both a the parameter values *and* the feature representation simultaneously.

In this chapter, we present a simple and tractable algorithm for deep exploration via randomized value functions with nonlinear generalization. We use the nonparametric bootstrap as a tractable and general method to generate approximate posterior samples where exact Bayesian inference is intractable. We show that, in some situations, this is equivalent to an exact Bayesian posterior. We also show that, when used with complex deep neural networks, the bootstrap can produce reasonable estimates for uncertainty. Next, we introduce bootstrapped Q-learning as an extension of RLSVI to nonlinear function approximators. We focus upon bootstrapped *deep Q networks* (DQN), which combine deep exploration with deep learning. Most material for this chapter is taken from our papers (Osband and Van Roy, 2015; Osband et al., 2016). Bootstrapped DQN can demonstrate deep exploration in complex stochastic MDPs to learn exponentially faster than DQN with dithering. Bootstrapped DQN scales to large problems such as the Arcade Learning Environment; it substantially improves learning times and performance across most Atari games.

8.1 Approximate Bayes by bootstrap

The term *bootstrap* refers to a class of methods for nonparametric estimation from data-driven simulation (Efron, 1982). In essence, the bootstrap uses the empirical distribution of a sampled dataset as an estimate of the population statistic (Efron and Tibshirani, 1994). In its most common form, the bootstrap takes as input a data set D and an estimator ψ. To generate a sample from the bootstrapped distribution, a data set $\tilde{D}$ of cardinality equal to that of D is sampled uniformly with replacement from D. The bootstrap sample estimate is then taken to be $\psi(\tilde{D})$. We can use multiple bootstrap samples $\hat{\psi}_1, .., \hat{\psi}_K$ to approximate the distribution of ψ by a discrete distribution $\hat{P} = \sum_{k=1}^{K} \mathbb{1}\{\hat{\psi}_1) = y\}$. Algorithm 8.1 outlines this approach where $\mathcal{P}(\mathcal{X})$ is the space of probability measures over a set $\mathcal{X}$.

Algorithm 8.1 A bootstrap sample

Input: Data $x_1, .., x_N \in \mathcal{X}$, estimator $\psi : \mathcal{P}(\mathcal{X}) \to \mathcal{Y}$
Output: Bootstrap sample $\hat{\psi}$

1: Sample data $\tilde{D} = (\tilde{x}_1, .., \tilde{x}_N)$ from $D = (x_1, .., x_N)$ uniformly with replacement
2: For all $dx \subset \mathcal{X}$ let $\hat{P}(dx) = \sum_{n=1}^{N} \mathbb{1}\{\tilde{x}_n \in dx\}/N$
3: Estimate $\hat{\psi} = \psi(\hat{P})$

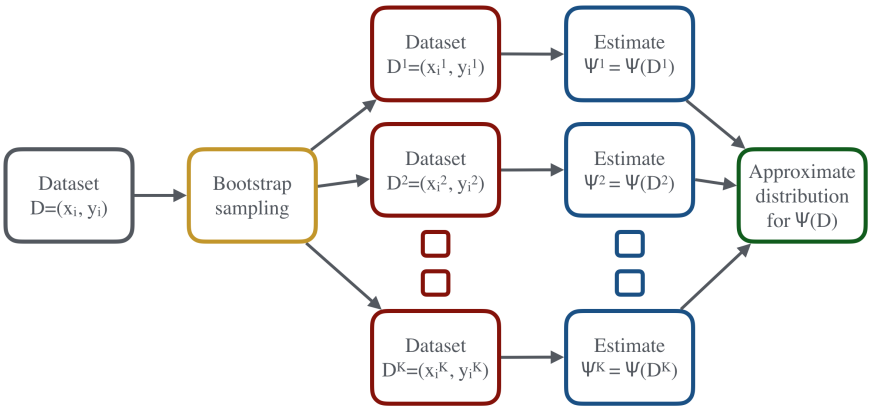


Figure 8.1: How to use Algorithm 8.1 to generate a bootstrap distribution.

This procedure allows estimates of the distribution of any unknown parameter in a flexible and non-parametric manner. Algorithm 8.1 is somewhat abstract. However, the procedure easily specializes to familiar concrete versions. For example, suppose ψ is the expectation operator. Then, each sample $\hat{\psi}_k$ is the mean of the subsampled dataset and the approximate distribution $\hat{P}$ is the relatively frequency measure of these sampled means.

The bootstrap is widely hailed as a great advance of 20th century applied statistics and comes with theoretical guarantees (Bickel and Freedman, 1981). The bootstrap is highly parallelizable and, as such, amenable to distributed computation. The approach can even scale to massive data with sub-linear computational cost (Kleiner et al., 2014).

The output $\hat{P}$ is somewhat reminiscent of a Bayesian posterior (Friedman et al., 2001). In fact, with a small modification to the weight sampling distribution we can modify Algorithm 8.1 to give the Bayesian bootstrap of Algorithm 8.2(Rubin et al., 1981). For this so-called Bayesian bootstrap each datapoint is reweighted according to an exponential random variable Exp(1), rather than resampled uniformly. In fact, this approach is quite similar as we note that for $N \to \infty$ the classical bootstrap approaches data reweighting $\sim$ Poi(1).

Algorithm 8.2 A Bayesian bootstrap sample

Input: Data $x_1, .., x_N \in \mathcal{X}$, estimator $\psi : \mathcal{P}(\mathcal{X}) \to \mathcal{Y}$
Output: Bootstrap sample $\hat{\psi}$

1: Sample $w_1, .., w_N \sim \text{Exp}(1)$
2: For all $dx \subset \mathcal{X}$ let $\hat{P}(dx) = \sum_{n=1}^{N} w_n \mathbb{1}\{x_n \in dx\} / \sum_{n=1}^{N} w_n$
3: Estimate $\hat{\psi} = \psi(\tilde{D})$

8.1.1 Uncertainty in deep neural networks

Deep neural networks represent the state of the art in many supervised learning domains (Krizhevsky et al., 2012). This is largely due to their flexibility, scalability and inductive bias, which allows them to learn effective feature representations. The exploration method we study in this paper is designed to be statistically and computationally efficient when used in conjunction with neural network representations of the value function.

To explore efficiently, it is important to quantify uncertainty in value estimates so that the agent can judge potential benefits of exploratory actions. The neural network literature presents a sizable body of work on uncertainty quantification founded on parametric Bayesian inference. These include variational Bayes (Graves, 2011; Blundell et al., 2015), assumed density filtering (Hernández-Lobato and Adams, 2015), dropout-based variational inference (Gal and Ghahramani, 2015; Kingma et al., 2015), and stochastic gradient Langevin

dynamics (Teh et al., 2015). Instead of appealing to parametric assumptions about the uncertainty present in RL problems, we use the non-parametric bootstrap (Efron, 1982) to obtain a distribution over functions represented by the neural network.

In Figure 8.2 we present an efficient and scalable method for generating bootstrap samples from a large and deep neural network. The network consists of a shared architecture with K bootstrapped "heads" branching off independently. Each head is trained only on its bootstrapped sub-sample of the data, which can be generated online (Owen et al., 2012). Thus each head represents $\psi(\tilde{D})$ in the classical view of bootstrap. The shared network learns a joint feature representation, this can provide significant computational advantages at the cost of lower diversity between heads. This type of bootstrap can even be implemented efficiently by a single forward/backward pass of backpropagation; it can be thought of as a data-dependent dropout, where the dropout mask for each head is fixed for each data point (Srivastava et al., 2014).

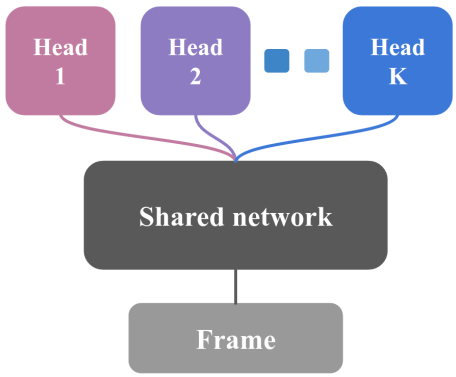
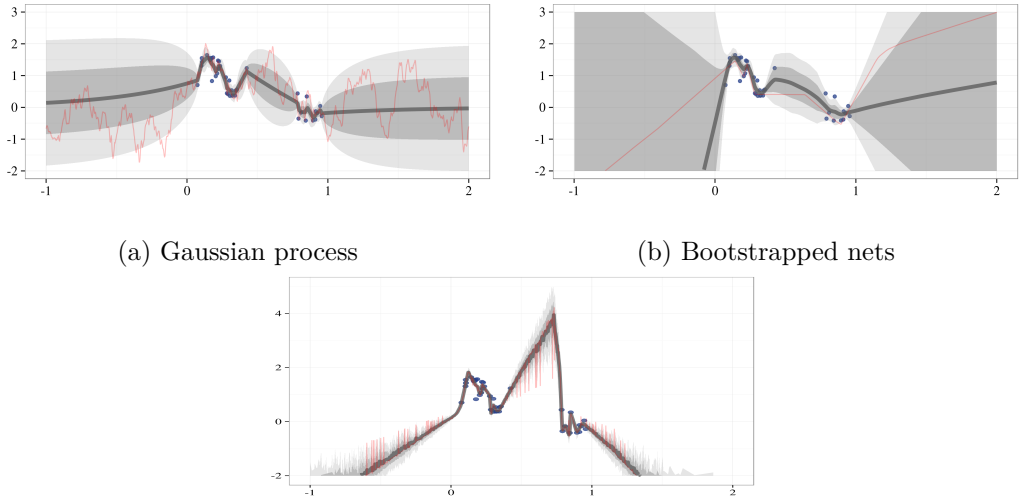


Figure 8.2: An efficient architecture for K bootstrap samples

Figure 8.3 presents an example of uncertainty estimates from bootstrapped neural networks on a regression task with noisy data. We trained a fully-connected 2-layer neural networks with 50 rectified linear units (ReLU) in each layer on 50 bootstrapped samples from the data. As is standard practice, we initialize these networks with random parameter values, this induces an important initial diversity in the models. We were unable to generate effective uncertainty estimates for this problem using the dropout approach in prior literature (Gal and Ghahramani, 2015). Further details are provided in (Osband et al., 2016).



(a) Gaussian process (b) Bootstrapped nets

(c) Dropout nets

Figure 8.3: Posterior estimates for a simple nonlinear regression task. Grey regions depict the mean estimate $\pm\{1, 2\}$ standard deviations. A single posterior sample is shown in red.

8.2 Bootstrapped Q-learning

In this section we will introduce the algorithm Bootstrapped Q-learning, which uses the nonparametric bootstrap to approximate the uncertainty over the optimal value function Q^*. At the start of each episode, bootstrapped DQN samples a single Q-value function from its approximate posterior. The agent then follows the policy which is optimal for that *sample* for the duration of the episode. This is a natural extension of RLSVI to nonlinear function approximation (Osband et al., 2014; Osband and Van Roy, 2015; Osband et al., 2016).

Algorithm 8.3 Bootstrapped Q learning

Input: Bootstrap method B, Q value approximator ψ

for episode $k = 1, 2, ..$ **do**
  Bootstrap past sample data $\tilde{\mathcal{H}}_{k1} = B(\mathcal{H}_{k1})$
  Estimate $\tilde{Q}_k \sim \psi(\tilde{\mathcal{H}}_{k1})$
  for $h = 1, .., H$ **do**
    Sample $a_{kh} \in \arg\max_{\alpha \in \mathcal{A}} \tilde{Q}_k(s_{kh}, \alpha)$
    Observe r_{kh} and $s_{k,h+1}$
  end for
end for

Algorithm 8.3 is general, since it is compatible with any Q value approximator ψ and any bootstrap method for approximate posterior inference (Osband and Van Roy, 2015). In particular, we are interested in the implementation of bootstrapped Q learning where we use a deep neural network to estimate this value.

8.2.1 Bootstrapped DQN

In *bootstrapped deep Q networks*, where we use a deep neural network to estimate the value function. Bootstrapped DQN builds upon *deep Q networks* (DQN) which has achieved high profile success in so-called "deep reinforcement learning" from raw pixels to control (Mnih et al., 2015). Before we can describe bootstrapped DQN, we first review some of the essential pieces of DQN. The Q-learning update after taking action a_t in state s_t and observing reward r_t and transitioning to s_{t+1} is given by

$$\theta_{t+1} \leftarrow \theta_t + \alpha(y_t^Q - Q(s_t, a_t; \theta_t))\nabla_\theta Q(s_t, a_t; \theta_t) \tag{8.1}$$

where α is the scalar learning rate and y_t^Q is the target value $r_t + \gamma \max_a Q(s_{t+1}, a; \theta^-)$. θ^- are target network parameters fixed $\theta^- = \theta_t$.

Several important modifications to the Q-learning update improve stability for DQN (Mnih et al., 2015). First the algorithm learns from sampled transitions from an experience buffer, rather than learning fully online. Second the algorithm uses a target network with parameters θ^- that are copied from the learning network $\theta^- \leftarrow \theta_t$ only every τ time steps and then kept fixed in between updates. Double DQN (DDQN) modifies the target y_t^Q:

$$y_t^Q \leftarrow r_t + \gamma \max_a Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta_t); \theta^-). \tag{8.2}$$

DDQN can demonstrate improved performance and stability (Van Hasselt et al., 2015). In this paper we use the DDQN update for all DQN variants unless explicitly stated.

Bootstrapped DQN (Algorithm 8.4) modifies DQN to produce *distribution* over Q-values via the bootstrap. At the start of each episode, bootstrapped DQN samples a single Q-value function from its approximate posterior. The agent then follows the policy which is optimal for that *sample* for the duration of the episode. This is a natural extension of the Thompson sampling heuristic to RL that allows for deep exploration.

In order to implement this algorithm efficiently online we build up $K \in \mathbb{N}$ bootstrapped estimates of the Q-value function in parallel as in Figure 8.2. Importantly, each one of these value function function heads $Q_k(s, a; \theta)$ is trained against its own target network $Q_k(s, a; \theta^-)$. This means that each $Q_1, .., Q_K$ provide a temporally extended (and consistent) estimate of the value uncertainty via TD estimates. We approximate a bootstrap sample by selecting $k \in \{1, .., K\}$ uniformly at random and following Q_k for the duration of that episode. Bootstrapped DQN exhibits deep exploration unlike the naive application of Thompson sampling to RL which resamples every timestep[1].

Implementation

The core idea to our implementation of bootstrapped DQN is the bootstrap mask m_t. The mask m_t decides, for each value function Q_k, whether or not it should train upon the experience generated at step t. In its simplest form m_t is a binary vector of length K, masking out or including each value function for training on that time step of experience (i.e., should it receive gradients from the corresponding $(s_t, a_t, r_{t+1}, s_{t+1}, m_t)$ tuple). The masking distribution M is responsible for generating each m_t. For example, when M yields m_t whose components are independently drawn from a Bernoulli distribution with parameter 0.5 then this corresponds to the double-or-nothing bootstrap (Owen et al., 2012). On the other hand, if M yields a mask m_t with all ones, then the algorithm reduces to an ensemble method. Poisson masks $M_t[k] \sim \text{Poi}(1)$ provides the most natural parallel with the standard non-parametric bootstrap since $\text{Bin}(N, 1/N) \to \text{Poi}(1)$ as $N \to \infty$. Exponential masks $M_t[k] \sim \text{Exp}(1)$ closely resemble the standard Bayesian nonparametric posterior of a Dirichlet process (Osband and Van Roy, 2015).

Periodically, the replay buffer is played back to update the parameters of the value function network Q. The gradients of the kth value function Q_k for the tth tuple in the replay buffer B, g_t^k is:

$$g_t^k = m_t^k (y_t^Q - Q_k(s_t, a_t; \theta)) \nabla_\theta Q_k(s_t, a_t; \theta) \tag{8.3}$$

where y_t^Q is given by (8.2). Note that the mask m_t^k modulates the gradient, giving rise to the bootstrap behavior.

[1] We call bootstrapped DQN which resamples a head every timestep Thompson DQN. This is similar to to the "Thompson sampling" algorithm of (Gal and Ghahramani, 2015) but uses the bootstrap in place of dropout as an approximate posterior.

Algorithm 8.4 Bootstrapped DQN

Input: Value function networks Q with K outputs $\{Q_k\}_{k=1}^K$. Masking distribution M.
Let B be a replay buffer storing experience for training.
for each episode **do**
  Obtain initial state from environment s_0
  Pick a value function to act using $k \sim \text{Uniform}\{1, \dots, K\}$
  for step $t = 1, \dots$ until end of episode **do**
    Pick an action according to $a_t \in \arg\max_a Q_k(s_t, a)$
    Receive state s_{t+1} and reward r_t from environment, having taking action a_t
    Sample bootstrap mask $m_t \sim M$
    Add $(s_t, a_t, r_{t+1}, s_{t+1}, m_t)$ to replay buffer B
  end for
end for

8.2.2 The importance of prior information

One important property of the bootstrap approaches we have described, the support of distributions $\hat{P}_k$ is restricted to the dataset $\{x_1, .., x_N\}$. In particular, these bootstrap distributions can differ significantly from a proper Bayesian posterior in that they lack any prior uncertainty. In sequential decision problems with sparse data this can pose a significant problem, since this may understate the posterior variance and lead to suboptimal exploitation (Osband and Van Roy, 2015).

We now examine a simple problem instance designed to highlight the importance of prior knowledge to incentivize efficient exploration in randomized value functions. Consider a trivial finite MDP with $S = H = 1$, and $A = 2$, fix $0 < \epsilon \ll 1$ and describe the underlying reward dynamics in terms of the Dirac delta function $\delta_x(y)$ which assigns all probability mass to x:

$$p_a^*(y) = \begin{cases} \delta_\epsilon(y) & \text{if } a = 1 \\ (1 - 2\epsilon)\delta_0(y) + 2\epsilon\delta_1(y) & \text{if } a = 2 \end{cases}. \tag{8.4}$$

The optimal policy is to pick $a = 2$ at every timestep, since this has an expected reward of 2ϵ instead of just ϵ. However, with probability at least $1 - 2\epsilon$, Bootstrapped Q learning will incur linear regret $\Omega(\epsilon T)$ (Osband and Van Roy, 2015).

To see why this is the case note that Algorithm 8.3 without artificial history must begin by sampling each arm once. In our system this means that with probability $1 - 2\epsilon$ the agent will receive a reward of ϵ from arm one and 0 from arm two. Given this history, the algorithm

will prefer to choose arm one for *all* subsequent timesteps, since its bootstrap estimates will always put all their mass on ϵ and 0 respectively. The problem here is that the bootstrap does not give meaningful uncertainty estimates in the limit of only one observation. When you only have a small amount of data an agent must rely on some prior to give meaningful uncertainty estimates. This shortcoming of the bootstrap can easily be remedied by either of two practical approaches.

1. **Artificial prior data**. We propose a simple and natural extension to the bootstrap: augment the dataset $\{x_1, \ldots, x_N\}$ with artificially generated samples $\{x_{N+1}, .., x_{N+M}\}$ and apply the bootstrap to the combined dataset. The artificially generated data can be viewed as inducing a prior distribution. In fact, if this artificial data is generated carefully this process is precisely equivalent to a Dirichlet process prior in some settings (Osband and Van Roy, 2015).

2. **Target smoothing**. In each boostrap sample, we can replace the observed targets y_i^k by $\tilde{y}_i^k := y_i^k + w_i^k$ for some mean zero noise w_k^i, for example $w_i^k \sim N(0, \sigma^2)$. This procedure is called "smoothed bootstrap" (Efron, 1982) and the scale of the noise σ determines the amount of smoothing. We note that, in the Gaussian linear regression setting, this scheme (without resampling) is equivalent to a Bayesian posterior with no prior conviction on the mean (Friedman et al., 2001).

In fact, for the simple multi-armed bandit example of (8.4) choosing α artificial observations at 0 and β artificial observations at 1 together with Bayesian bootstrap is precisely equivalent to a Beta(α, β) posterior distribution. Similarly, choosing smoothed bootstrap with $\sigma^2 = 1$ and no resampling results in a Gaussian posterior distribution with zero prior precision on the mean. In both cases, we can guarantee that the resulting bootstrap Thompson sampling will satisfy an optimal regret bound $\tilde{O}(\sqrt{T})$ (Agrawal and Goyal, 2012a).

Our implementation of Algorithm 8.4 does not incorporate artificial prior data, since this can be difficult to generate for arbitrary structured domains. However, we do incorporate a form of indirect target smoothing through the online DQN updates (Welling and Teh, 2011). This is because we do not train our networks to convergence before each TD-update, so each target Q may be thought of as a noisy approximation to the estimated Q. Incorporating more carefully-considered prior knowledge or structure is an interesting area for future research.

8.3 Deep exploration with deep learning

We now present a series of didactic computational experiments designed to highlight the need for deep exploration in RL. The agents will be placed in an environment made up of a long chain of states right next to a small reward. All other states have zero reward except at the far end of the chain where they can find a state with much higher reward. These experiments in this section are toy problems intended to be expository rather than entirely realistic. However, they clearly test whether an agent can efficiently balance the potential benefits of delayed information from deep exploration. Balancing a well known and mildly successful strategy versus an unknown, but potentially more rewarding, approach can emerge in many practical applications.

The environments in this section may be concisely described by a finite tabular MDP, similar to *RiverSwim* (Strehl and Littman, 2005). However, we will require our algorithm to interact with the MDP only through raw pixel features. For a chain of length N we will define features of the state $\phi : \{1, .., N\} \to \{0, 1\}^N$. We consider two feature mappings, "one-hot" $\phi_{\text{oh}}(s_t) := (\mathbb{1}\{x = s_t\})_{x=1,..,N}$ and "thermometer" $\phi_{\text{therm}}(s_t) := (\mathbb{1}\{x \leq s_t\})_{x=1,..,N}$. We find that bootstrapped DQN is able to explore efficiently with either set of features. We focus upon the thermometer encoding, since this better captures generalization between states. Full details for these experiments are given in (Osband et al., 2016).

8.3.1 Scaling deeper

We consider a family of deterministic chains of length $N > 3$ as in Figure 8.4. Each episode of interaction lasts $N + 9$ steps after which point the agent resets to the initial state s_2. We choose this so that the optimal policy is to move right at every step and receive a return of 10 in each episode. However, any shallow exploration strategy will take $\Omega(2^N)$ episodes to learn the optimal policy (Osband et al., 2014).

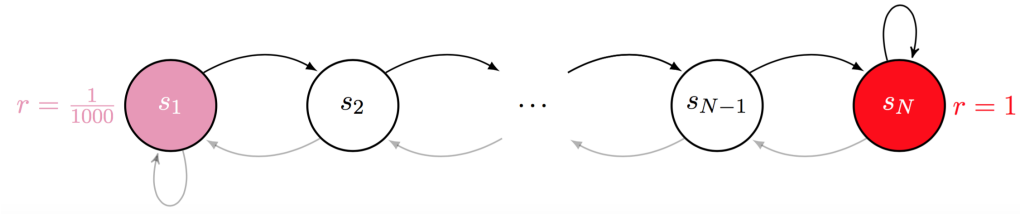


Figure 8.4: Scalable environments that requires deep exploration.

These environments may be described by a finite tabular MDP. However, we consider algorithms which interact with the MDP only through raw pixel features $\phi_{\text{therm}}(s_t) := (\mathbb{1}\{x \leq s_t\})$ in $\{0,1\}^N$ (Osband et al., 2016). We say that the algorithm has successfully learned the optimal policy when it has successfully completed one hundred episodes with optimal reward of 10. For each chain length, we ran each learning algorithm for 2000 episodes across three seeds. We plot the median time to learn in Figure 8.5, together with a conservative lower bound of $99+2^{N-11}$ on the expected time to learn for any shallow exploration strategy (Osband et al., 2014). Thompson DQN is the same as bootstrapped DQN, but resamples every timestep. Ensemble DQN uses the same architecture as bootstrapped DQN, but with an ensemble policy. Only bootstrapped DQN demonstrates a graceful scaling to long chains which require deep exploration.

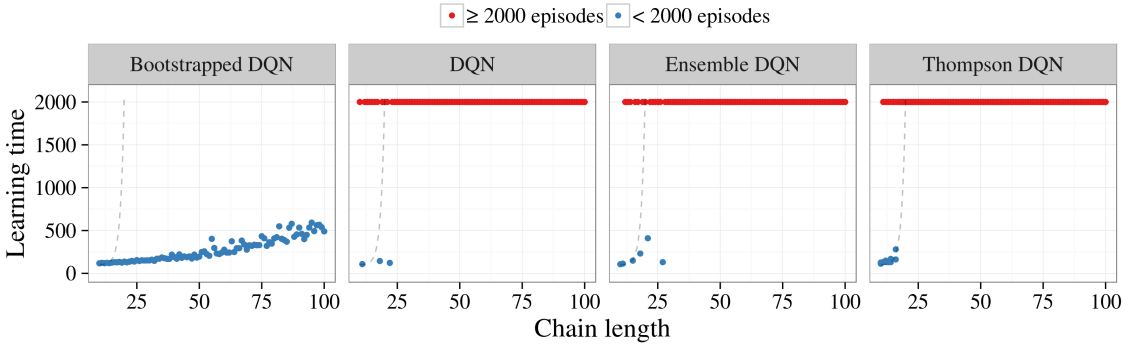


Figure 8.5: Only Bootstrapped DQN demonstrates deep exploration.

8.3.2 A difficult stochastic MDP

Figure 8.5 shows that bootstrapped DQN can implement effective (and deep) exploration where similar deep RL architectures fail. However, since the underlying system is a small and finite MDP there may be several other simpler strategies which would also solve this problem. We will now consider a difficult variant of this chain system with significant stochastic noise in transitions as depicted in Figure 8.6. Action "left" deterministically moves the agent left, but action "right" is only successful 50% of the time and otherwise also moves left. The agent interacts with the MDP in episodes of length 15 and begins each episode at s_1. Once again the optimal policy is to head right.

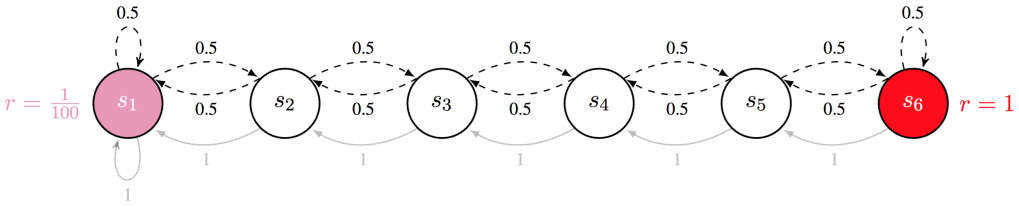


Figure 8.6: A stochastic MDP that requires deep exploration.

Bootstrapped DQN is unique amongst scalable approaches to efficient exploration with deep RL in stochastic domains. For benchmark performance we implement three algorithms which, unlike bootstrapped DQN, will receive the true tabular representation for the MDP. These algorithms are based on three state of the art approaches to exploration via dithering (ϵ-greedy), optimism (UCRL2 (Jaksch et al., 2010)) and posterior sampling (PSRL (Osband et al., 2013)).

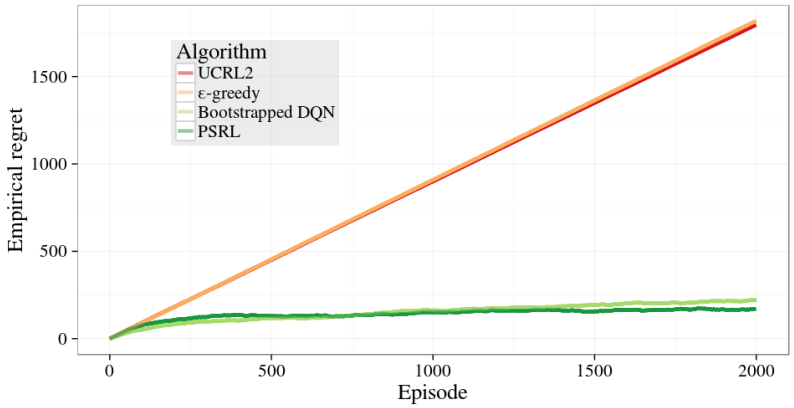


Figure 8.7: Bootstrapped DQN matches efficient tabular RL algorithms.

In Figure 8.7 we present the empirical regret of each algorithm averaged over 10 seeds over the first two thousand episodes. The empirical regret is the cumulative difference between the expected rewards of the optimal policy and the realized rewards of each algorithm. We find that bootstrapped DQN achieves similar performance to state of the art efficient exploration schemes such as PSRL even without prior knowledge of the tabular MDP structure and in noisy environments. Most telling is how much better bootstrapped DQN does than the state of the art optimistic algorithm UCRL2. Although Figure 8.7 seems to suggest UCRL2 incurs linear regret, actually it follows its bounds $\tilde{O}(S\sqrt{AT})$ (Jaksch et al., 2010) where S is the number of states and A is the number of actions. We demonstrate this through simulation (Osband et al., 2016).

8.3.3 How does bootstrapped DQN drive deep exploration?

Bootstrapped DQN explores in a manner similar to the provably-efficient algorithm PSRL (Osband et al., 2013) but it uses a bootstrapped neural network to approximate a posterior sample for the value. Unlike PSRL, bootstrapped DQN directly samples a value function and so does not require further planning steps. This algorithm is similar to RLSVI, which is also provably-efficient (Osband et al., 2014), but with a neural network instead of linear value function and bootstrap instead of Gaussian sampling. The analysis for the linear setting suggests that this nonlinear approach will work well so long as the distribution $\{Q^1, .., Q^K\}$ remains *stochastically optimistic* (Osband et al., 2014), or at least as spread out as the "correct" posterior.

Bootstrapped DQN relies upon random initialization of the network weights as a prior to induce diversity. Surprisingly, we found this initial diversity was enough to maintain diverse generalization to new and unseen states for large and deep neural networks. This is effective for our experimental setting, but will not work in all situations. In general it may be necessary to maintain some more rigorous notion of "prior", potentially through the use of artificial prior data to maintain diversity (see Section 8.2.2 or (Osband and Van Roy, 2015)). One potential explanation for the efficacy of simple random initialization is that unlike supervised learning or bandits, where all networks fit the same data, each of our Q^k heads has a unique target network. This, together with stochastic minibatch and flexible nonlinear representations, means that even small differences at initialization may become *bigger* as they refit to unique TD errors.

Bootstrapped DQN does *not* require that any single network Q^k is initialized to the correct policy of "right" at every step, which would be exponentially unlikely for large chains N. For the algorithm to be successful in this example we only require that the networks generalize in a diverse way to the actions they have never chosen in the states they have not visited very often. Imagine that, in the example above, the network has made it as far as state $\tilde{N} < N$, but never observed the action right $a = 2$. As long as one head k imagines $Q(\tilde{N}, 2) > Q(\tilde{N}, 2)$ then TD bootstrapping can propagate this signal back to $s = 1$ through the target network to drive deep exploration. The expected time for these estimates at n to propagate to at least one head grows gracefully in n, even for relatively small K, as our experiments show. We expand upon this intuition with a video designed to highlight *how* bootstrapped DQN demonstrates deep exploration `https://youtu.be/e3KuV_d0EMk`.

8.4 Arcade Learning Environment

We now evaluate our algorithm across 49 Atari games on the Arcade Learning Environment (Bellemare et al., 2012). Importantly, and unlike the experiments in Section 8.3, these domains are not specifically designed to showcase our algorithm. In fact, many Atari games are structured so that small rewards always indicate part of an optimal policy. This may be crucial for the strong performance observed by dithering strategies[2]. We find that exploration via bootstrapped DQN produces significant gains versus ϵ-greedy in this setting. Bootstrapped DQN reaches peak performance roughly similar to DQN. However, our improved exploration mean we reach human performance on average 30% faster across all games. This translates to significantly improved cumulative rewards through learning.

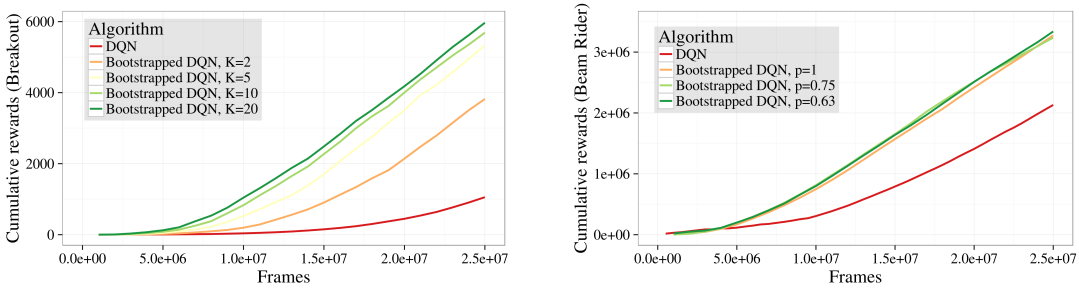
We follow the setup of (Van Hasselt et al., 2015) for our network architecture and benchmark our performance against their algorithm. Our network structure is identical to the convolutional structure of DQN (Mnih et al., 2015) except we split 10 separate bootstrap heads after the convolutional layer as per Figure 8.2. Recently, several authors have provided architectural and algorithmic improvements to DDQN (Wang et al., 2015; Schaul et al., 2015). We do not compare our results to these since their advances are orthogonal to our concern and could easily be incorporated to our bootstrapped DQN design. Full details of our experimental set up are available in (Osband et al., 2016).

8.4.1 Implementing bootstrapped DQN at scale

We now examine how to generate online bootstrap samples for DQN in a computationally efficient manner. We focus on three key questions: how many heads do we need, how should we pass gradients to the shared network and how should we bootstrap data online? We make significant compromises in order to maintain computational cost comparable to DQN.

Figure 8.8a presents the cumulative reward of bootstrapped DQN on the game Breakout, for different number of heads K. More heads leads to faster learning, but also increases the computational cost of the bootstrapped neural networks. We found that even a small number of heads captures most of the benefits of bootstrapped DQN. We choose $K = 10$ as a compromise between statistical and computational efficiency.

[2]By contrast, imagine that the agent received a small immediate reward for dying; dithering strategies would be hopeless at solving this problem, just like Section 8.3.



(a) Number of bootstrap heads K. (b) Probability of data sharing p.

Figure 8.8: Examining the sensitivities of bootstrapped DQN.

The shared network architecture allows us to train this combined network via backpropagation. Feeding K network heads to the shared convolutional network effectively increases the learning rate for this portion of the network. In some games, this leads to premature and sub-optimal convergence. We found the best final scores by normalizing the gradients by $1/K$, but this also leads to slower early learning. See (Osband et al., 2016) for more details.

To implement an online bootstrap we use an independent Bernoulli mask $w_1, .., w_K \sim \text{Ber}(p)$ for each head in each episode[3]. These flags are stored in the memory replay buffer and identify which heads are trained on which data. However, when trained using a shared minibatch the algorithm will also require an effective $1/p$ more iterations; this is undesirable computationally. Surprisingly, we found the algorithm performed similarly irrespective of p and all outperformed DQN, as shown in Figure 8.8b. This is strange and we discuss this phenomenon in (Osband et al., 2016). However, in light of this empirical observation for Atari, we chose $p = 1$ to save on minibatch passes. As a result bootstrapped DQN runs at similar computational speed to vanilla DQN on identical hardware[4].

8.4.2 Efficient exploration in Atari 2600

We find that Bootstrapped DQN drives efficient exploration in several Atari games. For the same amount of game experience, bootstrapped DQN generally outperforms DQN with ϵ-greedy exploration. Figure 8.9 demonstrates this effect for a diverse selection of games.

[3] $p = 0.5$ is double-or-nothing bootstrap (Owen et al., 2012), $p = 1$ is ensemble with no bootstrapping.
[4] Our implementation K=10, p=1 ran with less than a 20% increase on wall-time versus DQN.

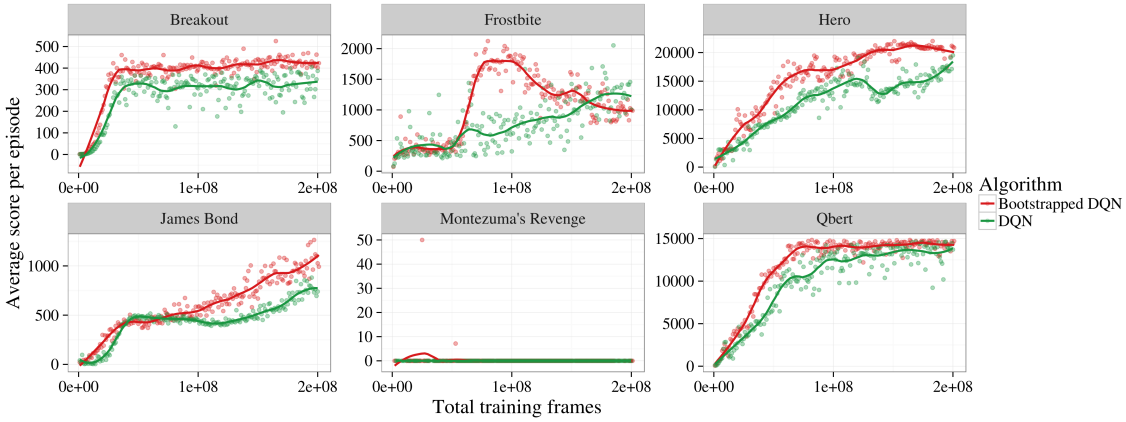


Figure 8.9: Bootstrapped DQN drives more efficient exploration.

On games where DQN performs well, bootstrapped DQN typically performs better. Bootstrapped DQN does not reach human performance on Amidar (DQN does) but does on Beam Rider and Battle Zone (DQN does not). To summarize this improvement in learning time we consider the number of frames required to reach human performance. If bootstrapped DQN reaches human performance in $1/x$ frames of DQN we say it has improved by x. Figure 8.10 shows that Bootstrapped DQN typically reaches human performance significantly faster.

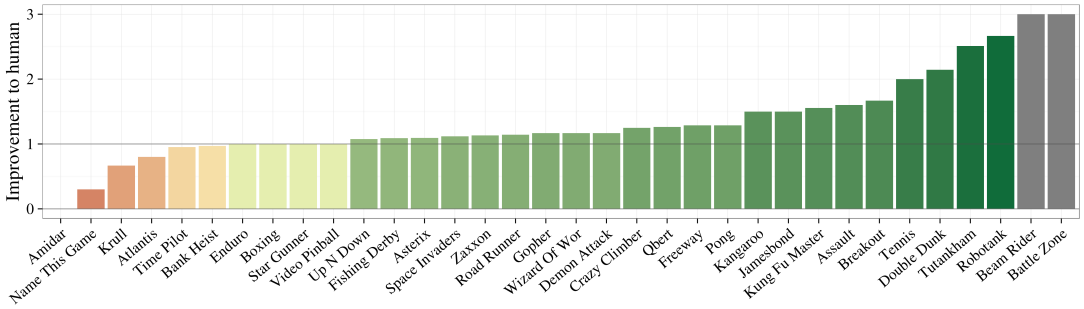


Figure 8.10: Bootstrapped DQN reaches human performance faster than DQN.

On most games where DQN does not reach human performance, bootstrapped DQN does not solve the problem by itself. On some challenging Atari games where deep exploration is conjectured to be important (Van Hasselt et al., 2015) our results are not entirely successful, but still promising. In Frostbite, bootstrapped DQN reaches the second level much faster than DQN but network instabilities cause the performance to crash. In Montezuma's Revenge, bootstrapped DQN reaches the first key after 20m frames (DQN never observes a reward even after 200m frames) but does not properly learn from this experience[5]. Our results suggest that improved exploration may help to solve these remaining games, but also highlight the importance of other problems like network instability, reward clipping and temporally extended rewards.

8.4.3 Overall performance

Bootstrapped DQN is able to learn much faster than DQN. Figure 8.11 shows that bootstrapped DQN also improves upon the final score across most games. However, the real benefits to *efficient* exploration mean that bootstrapped DQN outperforms DQN by orders of magnitude in terms of the *cumulative* rewards through learning (Figure 8.12. In both figures we normalize performance relative to a fully random policy. The most similar work to ours presents several other approaches to improved exploration in Atari (Stadie et al., 2015) they optimize for AUC-20, a normalized version of the cumulative returns after 20m frames. According to their metric, averaged across the 14 games they consider, we improve upon both base DQN (0.29) and their best method (0.37) to obtain 0.62 via bootstrapped DQN. Further details of complete scores are available in (Osband et al., 2016).

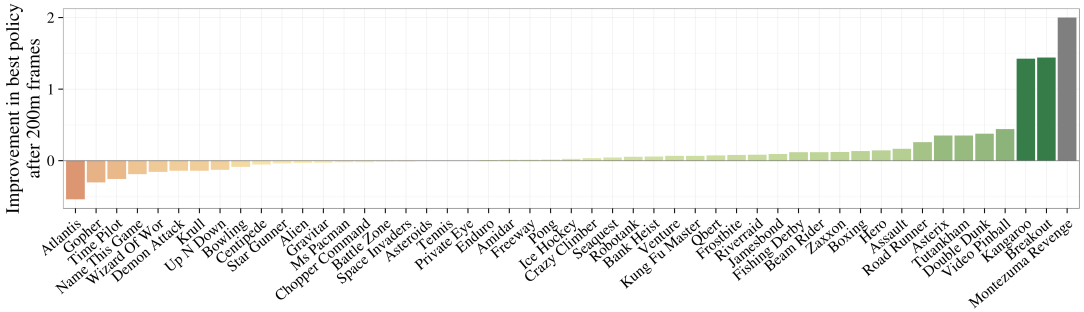


Figure 8.11: Bootstrapped DQN typically improves upon the best policy.

[5] An improved training method, such as prioritized replay (Schaul et al., 2015) may help solve this problem.

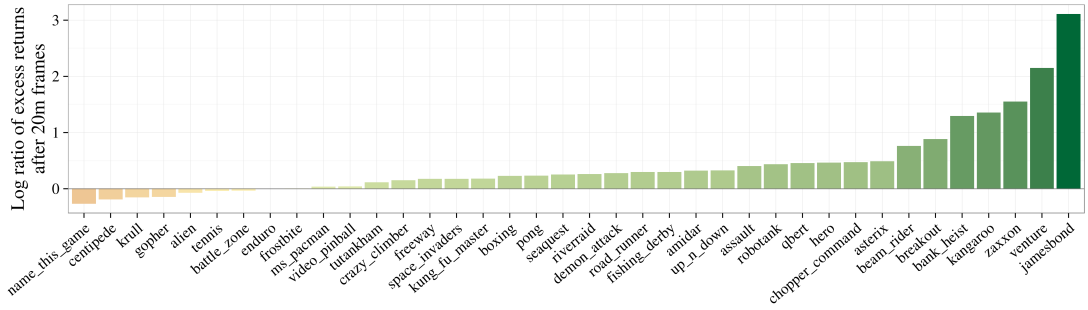


Figure 8.12: Bootstrapped DQN improves cumulative rewards by orders of magnitude.

8.4.4 Visualizing bootstrapped DQN

We now present some more insight to how bootstrapped DQN drives deep exploration in Atari. In each game, although each head $Q^1, .., Q^{10}$ learns a high scoring policy, the policies they find are quite distinct. In the video https://youtu.be/Zm2KoT82O_M we show the evolution of these policies simultaneously for several games. Figure 8.13 displays a screenshot of 9 bootstrap heads at the same intermediate timestep within an episode. Although each head performs well, they each follow a unique policy. By contrast, ϵ-greedy strategies are almost indistinguishable for small values of ϵ and totally ineffectual for larger values. We believe that this deep exploration is key to improved learning, since diverse experiences allow for better generalization.

Disregarding exploration, bootstrapped DQN may be beneficial as a purely exploitative policy. We can combine all the heads into a single ensemble policy, for example by choosing the action with the most votes; this approach might have several benefits. First, we find that the ensemble policy can often outperform any individual policy. Second, the distribution of votes across heads to give a measure of the uncertainty in the optimal policy. Unlike vanilla DQN, bootstrapped DQN can know what it doesn't know. In an application where executing a poorly-understood action is dangerous this could be crucial. Figure 8.14 presents a visualization of this uncertainty at two different stages of the game Breakout. We find that the uncertainty in this policy is surprisingly interpretable: all heads agree at clearly crucial decision points, but remain diverse at other less important steps. In the video https://youtu.be/0jvEcC5JvGY we visualize this ensemble policy across several games.

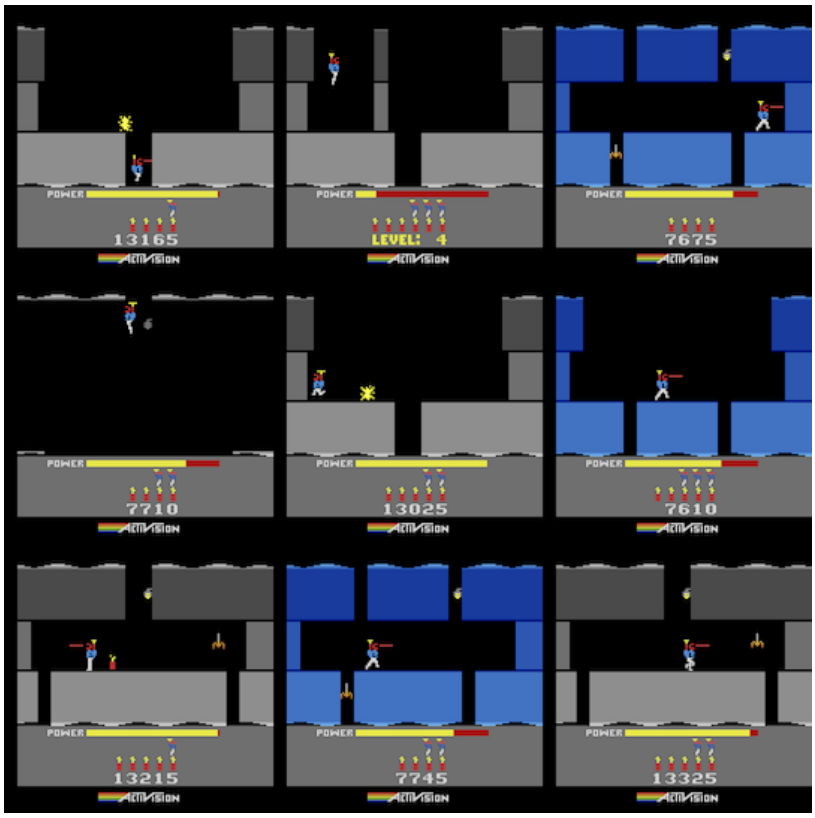
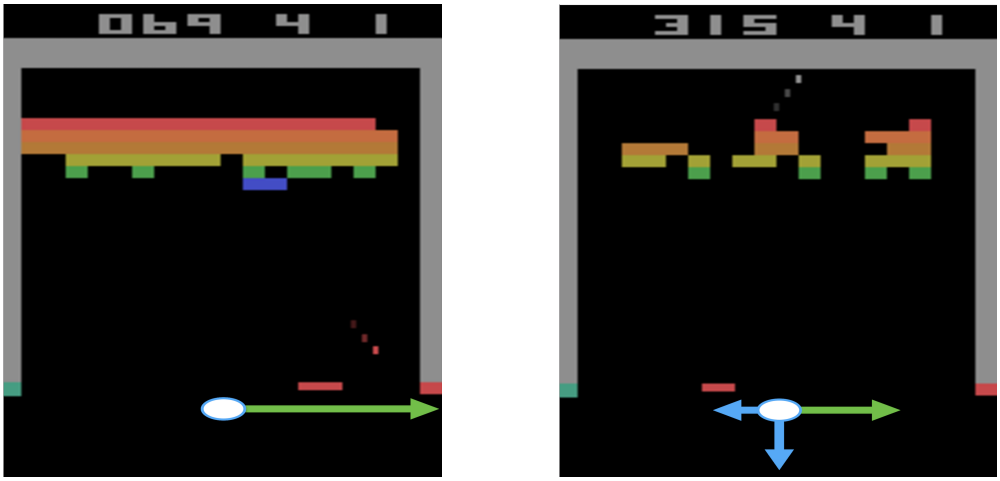


Figure 8.13: Diverse exploration policies from bootstrap heads in the game Hero.



(a) All bootstrap heads vote right. (b) Bootstrap heads disagree on policy.

Figure 8.14: Visualizing uncertainty in an ensemble policyheads in the game Breakout.

8.5 Extensions to continuous action spaces

Bootstrapped DQN can combine generalization with deep exploration solve problems with high-dimensional observation spaces, but it can only handle discrete and low-dimensional actions spaces. In particular, Algorithm 8.3 assumes that the step $\arg\max_a Q_k(s, a)$ can be computed as an intermediate step. In many situations of interest, such as physical control, the action space may be high dimensional and/or continuous (Lillicrap et al., 2015; Sutton and Barto, 1998). A standard technique to deal with this problem is the so-called actor-critic family of algorithms (Konda and Tsitsiklis, 1999) which introduce a parameterized policy $\pi(\cdot|\theta)$ for action selection. In fact, the main insights of Algorithms 8.3 and 8.4 can also be applied actor-critic architectures in the same way. We outline this algorithm, bootstrapped actor critic, in Algorithm 8.5.

Algorithm 8.5 Bootstrapped actor critic

Input: Value function networks Q with K outputs $\{Q_k\}_{k=1}^K$. Policy networks π with K outputs $\{\pi_k\}_{k=1}^K$. Masking distribution M.
Let B be a replay buffer storing experience for training Q.
for each episode **do**
  Obtain initial state from environment s_0
  Pick an actor-critic pair to act using $k \sim \text{Uniform}\{1, \ldots, K\}$
  for step $t = 1, \ldots$ until end of episode **do**
    Pick an action according to $a_t \sim \pi_k(s_t)$
    Receive state s_{t+1} and reward r_t from environment, having taken action a_t
    Sample bootstrap mask $m_t \sim M$
    Add $(s_t, a_t, r_{t+1}, s_{t+1}, m_t)$ to replay buffer B
    Update policy π_k by policy gradient with respect to Q_k
  end for
end for

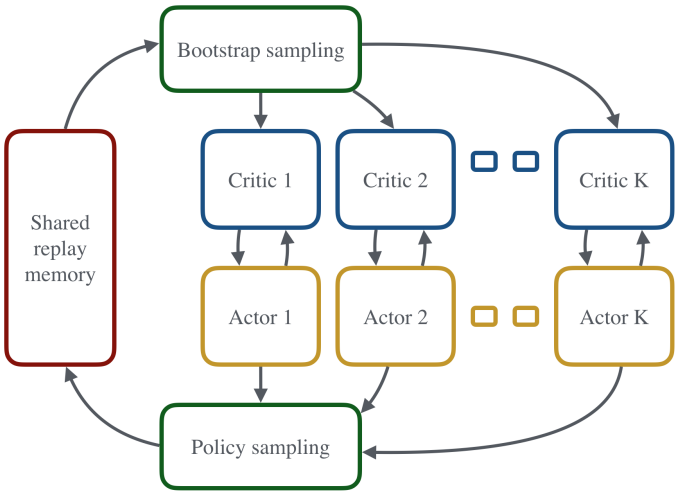


Figure 8.15: A natural extension of Bootstrapped DQN to actor critic.

Once again, instead of training a single Q-value estimate we will train K estimates in parallel on bootstrap perturbations of the observed data. For actor-critic architectures these Q-value estimates are known as the "critic". For each Q-value estimate we train an "actor" policy $\pi_k(\cdot|\theta)$ by policy gradient with respect to its own critic Q_k. For a single active agent the active policy can be selected uniformly at random for each episode. Alternatively, for large-scale computer simulations we might run each policy concurrently (Mnih et al., 2016). We present a depiction of this high-level approach in Figure 8.15. This architecture is able to combine the benefits of deep exploration via randomized value functions, together with a parameterized policy. Exploring these methods in more detail is a promising direction for future research.

Bibliography

Yasin Abbasi-Yadkori and Csaba Szepesvári. Bayesian optimal control of smoothly parameterized systems. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2015.

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *NIPS*, pages 2312–2320, 2011.

Shipra Agrawal and Navin Goyal. Further optimal regret bounds for Thompson sampling. *arXiv preprint arXiv:1209.3353*, 2012a.

Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. *arXiv preprint arXiv:1209.3352*, 2012b.

Mauricio Araya, Olivier Buffet, and Vincent Thomas. Near-optimal brl using optimistic local transitions. *arXiv preprint arXiv:1206.4613*, 2012.

Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. *Mathematical finance*, 9(3):203–228, 1999.

John Asmuth, Lihong Li, Michael L Littman, Ali Nouri, and David Wingate. A bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 19–26. AUAI Press, 2009.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2003.

Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 89–96. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3401-near-optimal-regret-bounds-for-reinforcement-learning.pdf.

Peter L. Bartlett and Ambuj Tewari. REGAL: A regularization based algorithm for reinforcement learning in weakly communicating MDPs. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI2009)*, pages 35–42, June 2009.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 2012.

Dimitri P Bertsekas and Sergey Ioffe. Temporal differences-based policy iteration and applications in neuro-dynamic programming. *Lab. for Info. and Decision Systems Report LIDS-P-2349, MIT, Cambridge, MA*, 1996.

Dimitri P. Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, September 1996.

Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

Peter J Bickel and David A Freedman. Some asymptotic theory for the bootstrap. *The Annals of Statistics*, pages 1196–1217, 1981.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *ICML*, 2015.

Scott A Boorman. *The protracted game: a wei-chÊżi interpretation of Maoist revolutionary strategy*. Oxford University Press, 1969.

Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1):49–107, 2000.

Ronen I. Brafman and Moshe Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *CoRR*, abs/1204.5721, 2012. URL http://arxiv.org/abs/1204.5721.

Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12:1587âĂŞ1627, 2011.

Valerii V Buldygin and Yu V Kozachenko. Sub-gaussian random variables. *Ukrainian Mathematical Journal*, 32(6):483–489, 1980.

Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.

Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems*, page TBA, 2015.

Richard Dearden, Nir Friedman, and Stuart J. Russell. Bayesian Q-learning. In *AAAI/IAAI*, pages 761–768, 1998.

Joseph L Doob. *Measure theory*, volume 143. Springer Science & Business Media, 2012.

John C Duchi. *Multiple Optimality Guarantees in Statistical Learning*. PhD thesis, Citeseer, 2014.

Bradley Efron. *The jackknife, the bootstrap and other resampling plans*, volume 38. SIAM, 1982.

Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

Vivek F Farias and Benjamin Van Roy. Tetris: A study of randomized constraint sampling. In *Probabilistic and Randomized Methods for Design Under Uncertainty*, pages 189–201. Springer, 2006.

Sarah Filippi, Olivier Cappé, and Aurélien Garivier. Optimism in reinforcement learning and kullback-leibler divergence. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pages 115–122. IEEE, 2010.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

Victor Gabillon, Mohammad Ghavamzadeh, and Bruno Scherrer. Approximate dynamic programming finally performs well in the game of tetris. In *Advances in neural information processing systems*, pages 1754–1762, 2013.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv preprint arXiv:1506.02142*, 2015.

Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*, volume 2. Taylor & Francis, 2014.

Aditya Gopalan and Shie Mannor. Thompson sampling for learning parameterized markov decision processes. *arXiv preprint arXiv:1406.7498*, 2014.

Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored MDPs. *J. Artif. Intell. Res.(JAIR)*, 19:399–468, 2003.

Arthur Guez, David Silver, and Peter Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems*, pages 1025–1033, 2012.

Arthur Guez, David Silver, and Peter Dayan. Scalable and efficient bayes-adaptive reinforcement learning based on monte-carlo tree search. *Journal of Artificial Intelligence Research*, pages 841–883, 2013.

Arthur Guez, David Silver, and Peter Dayan. Better optimism by bayes: Adaptive planning with rich models. *arXiv preprint arXiv:1402.1958*, 2014.

Josef Hadar and William R Russell. Rules for ordering uncertain prospects. *The American Economic Review*, pages 25–34, 1969.

José Miguel Hernández-Lobato and Ryan P Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. *ICML*, 2015.

Morteza Ibrahimi, Adel Javanmard, and Benjamin Van Roy. Efficient reinforcement learning for high dimensional linear quadratic systems. In *NIPS*, pages 2645–2653, 2012.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

Harold Jeffreys. An invariant form for the prior probability in estimation problems. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 186, pages 453–461. The Royal Society, 1946.

Sham Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, University College London, 2003.

Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *Algorithmic Learning Theory*, pages 199–213. Springer, 2012.

Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine Learning*, 49(2-3):193–208, 2002.

Michael J. Kearns and Satinder P. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *arXiv preprint arXiv:1506.02557*, 2015.

Ariel Kleiner, Ameet Talwalkar, Purnamrita Sarkar, and Michael I Jordan. A scalable bootstrap for massive data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(4): 795–816, 2014.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.

J Zico Kolter and Andrew Y Ng. Near-bayesian exploration in polynomial time. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 513–520. ACM, 2009.

Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *NIPS*, volume 13, pages 1008–1014, 1999.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

Tor Lattimore and Marcus Hutter. PAC bounds for discounted MDPs. In *Algorithmic learning theory*, pages 320–334. Springer, 2012.

Tor Lattimore, Marcus Hutter, and Peter Sunehag. The sample-complexity of general reinforcement learning. In *ICML*, 2013.

Erich Leo Lehmann, Joseph P Romano, and George Casella. *Testing statistical hypotheses*, volume 150. Wiley New York et al, 1986.

Erich Leo Lehmann, George Casella, and George Casella. *Theory of point estimation*. Wadsworth & Brooks/Cole Advanced Books & Software, 1991.

Jan Leike, Tor Lattimore, Laurent Orseau, and Marcus Hutter. Thompson sampling is asymptotically optimal in general environments. *arXiv preprint arXiv:1602.07905*, 2016.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Hamid Reza Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S. Sutton. Toward off-policy learning control with function approximation. In *ICML*, pages 719–726, 2010.

Odalric-Ambrym Maillard, Timothy A Mann, and Shie Mannor. How hard is my MDP?" the distribution-norm to the rescue". In *Advances in Neural Information Processing Systems*, pages 1835–1843, 2014.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016.

Rémi Munos. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. 2014.

Ronald Ortner and Daniil Ryabko. Online regret bounds for undiscounted continuous reinforcement learning. In *NIPS*, 2012.

Ian Osband and Benjamin Van Roy. Model-based reinforcement learning and the eluder dimension. In *Advances in Neural Information Processing Systems*, pages 1466–1474, 2014a.

Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored MDPs. In *Advances in Neural Information Processing Systems*, pages 604–612, 2014b.

Ian Osband and Benjamin Van Roy. Bootstrapped thompson sampling and deep exploration. *arXiv preprint arXiv:1507.00300*, 2015.

Ian Osband and Benjamin Van Roy. Posterior sampling for reinforcement learning without episodes. *arXiv preprint arXiv:1608.02731*, 2016a.

Ian Osband and Benjamin Van Roy. On lower bounds for regret in reinforcement learning. *arXiv preprint arXiv:1608.02732*, 2016b.

Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning. *arXiv preprint arXiv:1607.00215*, 2016c.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (More) efficient reinforcement learning via posterior sampling. In *NIPS*, pages 3003–3011. Curran Associates, Inc., 2013.

Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. *arXiv preprint arXiv:1402.0635*, 2014.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *arXiv preprint arXiv:1602.04621*, 2016.

Art B Owen, Dean Eckles, et al. Bootstrapping data arrays of arbitrary order. *The Annals of Applied Statistics*, 6(3):895–927, 2012.

Donald B Rubin et al. The bayesian bootstrap. *The annals of statistics*, 9(1):130–134, 1981.

Walter Rudin. *Principles of mathematical analysis*, volume 3. McGraw-Hill New York, 1964.

Paat Rusmevichientong and John N. Tsitsiklis. Linearly parameterized bandits. *Math. Oper. Res.*, 35(2):395–411, 2010.

Dan Russo and Benjamin Van Roy. Eluder dimension and the sample complexity of optimistic exploration. In *NIPS*, pages 2256–2264. Curran Associates, Inc., 2013.

Daniel Russo. Efficient learning in sequential optimization. *PhD Thesis*, 2015.

Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Steven L Scott. A modern bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

Alexander L Strehl and Michael L Littman. A theoretical analysis of model-based interval estimation. In *Proceedings of the 22nd international conference on Machine learning*, pages 856–863. ACM, 2005.

Alexander L. Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L. Littman. PAC model-free reinforcement learning. In *ICML*, pages 881–888, 2006.

Malcolm J. A. Strens. A bayesian framework for reinforcement learning. In *ICML*, pages 943–950, 2000.

Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, March 1998.

Richard Stuart Sutton. Temporal credit assignment in reinforcement learning. 1984.

Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.

István Szita and András Lörincz. Learning tetris using the noisy cross-entropy method. *Neural computation*, 18(12):2936–2941, 2006.

István Szita and Csaba Szepesvári. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1031–1038, 2010.

Yee Whye Teh, Leonard Hasenclever, Thibaut Lienart, Sebastian Vollmer, Stefan Webb, Balaji Lakshminarayanan, and Charles Blundell. Distributed bayesian learning with stochastic natural-gradient expectation propagation and the posterior server. *arXiv preprint arXiv:1512.09327*, 2015.

Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3): 58–68, 1995.

W.R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Sebastian B Thrun. Efficient exploration in reinforcement learning. 1992.

John N Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE transactions on automatic control*, 42(5):674–690, 1997.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR, abs/1509.06461*, 2015.

Tao Wang, Daniel J. Lizotte, Michael H. Bowling, and Dale Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *ICML*, pages 956–963, 2005.

Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J Weinberger. Inequalities for the l1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep*, 2003.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.

Zheng Wen. Efficient reinforcement learning with value function generalization. *PhD Thesis*, 2014.

Zheng Wen and Benjamin Van Roy. Efficient exploration and value function generalization in deterministic systems. In *NIPS*, pages 3021–3029, 2013.

Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 3381–3387. IEEE, 2008.