

Deep Exploration via Randomized Value Functions

Ian Osband

March 2, 2016

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. Statistically efficient reinforcement learning requires “deep exploration” which is defined as 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 thesis we present an alternative approach to deep exploration through the use of randomized value functions. This algorithm is inspired by the Thompson sampling heuristic for multi-armed bandits and achieves the first regret bounds $\tilde{O}(\sqrt{SAT})$ where S is the number of states, A is the number of actions and T is the time elapsed. Importantly, and unlike previous approaches to deep exploration, this approach is also computationally efficient and scales gracefully to complex domains with generalization. We establish several state of the art theoretical results and support our analysis through extensive experiments; these include several didactic examples as well as a recommendation system, Tetris, and Atari 2600 games.

Contents

1	Introduction	6
1.1	Learning to act	6
1.2	Deep exploration	8
1.3	Randomized value functions	9
1.4	Organization	11
1.5	Contributions	14
I	Tabula Rasa	15
2	Tabular Reinforcement Learning	16
2.1	Problem formulation	17
2.2	A wish list for learning	19
2.3	Fundamental limits to learning	23
2.4	Inefficient algorithms	25
3	Optimism in the Face of Uncertainty	31
3.1	Efficient reinforcement learning via optimism	32
3.2	Shortcomings of optimistic algorithms	39
3.3	Stochastic optimism	42

CONTENTS	3
3.4 Gaussian-Dirichlet optimism	46
4 Posterior Sampling for Reinforcement Learning	58
4.1 Thompson sampling and probability matching	59
4.2 (Even more) efficient reinforcement learning	62
4.3 Relating Bayesian regret to frequentist guarantees	69
4.4 On the horizon length and learning	76
4.5 Experimental evaluation	84
II Model-based Generalization	90
5 Factored MDPs	92
5.1 Problem formulation	93
5.2 Results	95
5.3 Analysis	96
6 Parameterized MDPs	99
6.1 Problem formulation	99
6.2 Results	102
6.3 The eluder dimension	104
6.4 Analysis	106
6.5 Technical proofs	109
III Value-based Generalization	117
7 Linear Value Functions	119
7.1 Randomized Least Squares Value Iteration	120

CONTENTS	4
7.2 Provably efficient tabular learning	123
7.3 Testing for efficient exploration	126
7.4 Learning to play Tetris	130
7.5 A simple recommendation system	132
8 Nonlinear Value Functions	135
8.1 Bayes by bootstrap	136
8.2 Bootstrapped Q-learning	139
8.3 Deep exploration with deep learning	142
8.4 Experiments on Atari 2600	145
Bibliography	153

Acknowledgements

- Family
- Ben
- Dan + Zheng + MS & E
- Friends etc
- Stanford
- Google + DeepMind

Chapter 1

Introduction

1.1 Learning to act

The “Big Data” revolution is spawning systems designed to make decisions from data. There is a hope that we might be able to 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 any fixed dataset. More and more, automated systems are being deployed to pro-actively make decisions without human oversight. However, in these settings the decisions an algorithm makes will actually affect the data it gathers. If you want to make good decisions from data, you first need good data. Even an *optimal* algorithm for learning from any fixed dataset may lead to substantially *suboptimal* performance in systems where the data is not independent of the decisions you make. Reinforcement learning is a framework for learning to act in systems where the decisions you make can affect the data you get. Reinforcement learning extends upon learning from datasets (which includes both supervised and unsupervised learning) in two key ways.

The first extension is partial feedback; an agent may only gather data about the course of action it *actually* chooses. For example in medical testing, a doctor might only gather data about the efficacy of a certain treatment by actually trying it out on a real patient. 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. Traditional methods for learning from datasets usually address this issue by separating exploration into an initial experimentation phase and then exploiting once more data is gathered. However, in modern systems for online data-driven decisions this separation may be extremely suboptimal. This problem of exploration vs exploitation 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 is long term consequences; an agent may make decisions at one timestep that actually alter the underlying system in future timesteps. For example in agriculture, the amount of water, fertilizer or pesticide which a farmer applies on any single day will affect the appropriate control in subsequent days. 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 (RL) is a general framework for artificial intelligence that combines learning from datasets with partial feedback and long term consequences to learn 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.

Designing algorithms which can facilitate generalization, partial feedback and long term consequences can be challenging. However, if you want to design a general artificial intelligence to learn to *act* then learning from datasets is not enough; you will need RL. In order to learn efficiently, an algorithm must direct its exploration towards this potentially informative data. However, for efficient RL with long term consequences it is not enough for this exploration to be directed, the exploration must also be deep.

1.2 Deep exploration

A learning algorithm which is purely exploitative, seeking only to maximize its performance given its current knowledge, may never learn the optimal policy. This can be true no matter how much data is gathered or what advanced technique is used to analyse this data. If the agent does not explore, it might never gather the data it needs to learn about actions which it never chooses.

The most common approach to this problem is through separate experimentation phases. The traditional statistics approach is to first run an experiment to first gather its exploratory data. Then based upon this dataset, we can perform an analysis to determine the best estimated policy for all subsequent timesteps or re-run the experiment if necessary. This procedure can be extremely wasteful both in terms of the quality of its decisions (a test a level $p = 0.05$ will be incorrect one in twenty times) and its performance during experimentation (we may use far more data than we require). In addition, many modern data-driven decision systems do not naturally separate in to a exploration then exploitation phase. We want algorithms that can learn to make good decisions *while* they are learning. The problem formulation known as the multi-armed bandit deals with exactly this.

In the multi-armed bandit an agent must learn to optimize an unknown function through sequential periods of interaction. The agent only receives observations at the actions which it actually takes, this leads to the fundamental tradeoff of exploration vs exploitation. The most common approach to exploration is given by dithering. Dithering strategies explore at random times in order to spread their observations beyond just the greedy exploitative strategy. Dithering strategies will *eventually* learn the optimal policy, since they will sample every action infinitely often, but they typically take exponentially long to learn. In bandits, this choice of directed exploration rather than dithering generally categorizes efficient algorithms.

Reinforcement learning extends upon the multi-armed bandit to allow long term consequences. In this RL setting directed exploration is not enough to guarantee efficiency; the exploration 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. Unlike bandit problems, which balance actions which are immediately rewarding or

immediately informative, RL settings require planning over several time steps [44]. For exploitation, this means that an efficient agent must consider the future rewards over several 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.

To illustrate this distinction, consider a simple deterministic chain $\{s_{-3}, \dots, s_{+3}\}$ with three step horizon starting from state s_0 . This MDP is known to the agent a priori, with deterministic actions “left” and “right”. 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 is unknown. 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. We illustrate this environment in Figure 1.1. Figure 1.2 depicts the planning and look ahead trees for several algorithmic approaches in this example MDP. The action “left” is gray, the action “right” is black. Rewarding states are depicted as red, informative states as blue. Dashed lines indicate that the agent can plan ahead for either rewards or information. Unlike bandit algorithms, an RL agent can plan to exploit future rewards. Only an RL agent with deep exploration can plan to learn.

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 [18]. Despite this observation, the many high profile applications of RL have either been purely exploitative [84] or employed inefficient dithering schemes for exploration [57]. At the same time, algorithms which do exhibit deep exploration can learn exponentially faster [44] or even tackle extremely complex learning problems [75]. The main reason that 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 thesis we present a computationally approach to deep exploration via randomized value functions.

1.3 Randomized value functions

If you want to make good decisions from data, you need good data. In order to design an algorithm which is statistically efficient you must direct the data it gathers towards

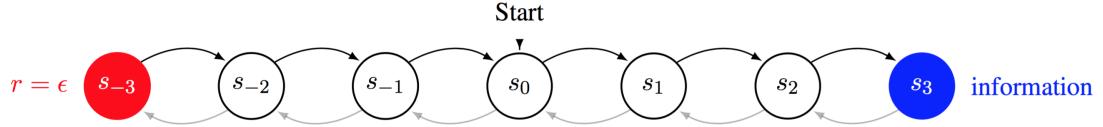


Figure 1.1: An example environment designed to illustrate deep exploration.

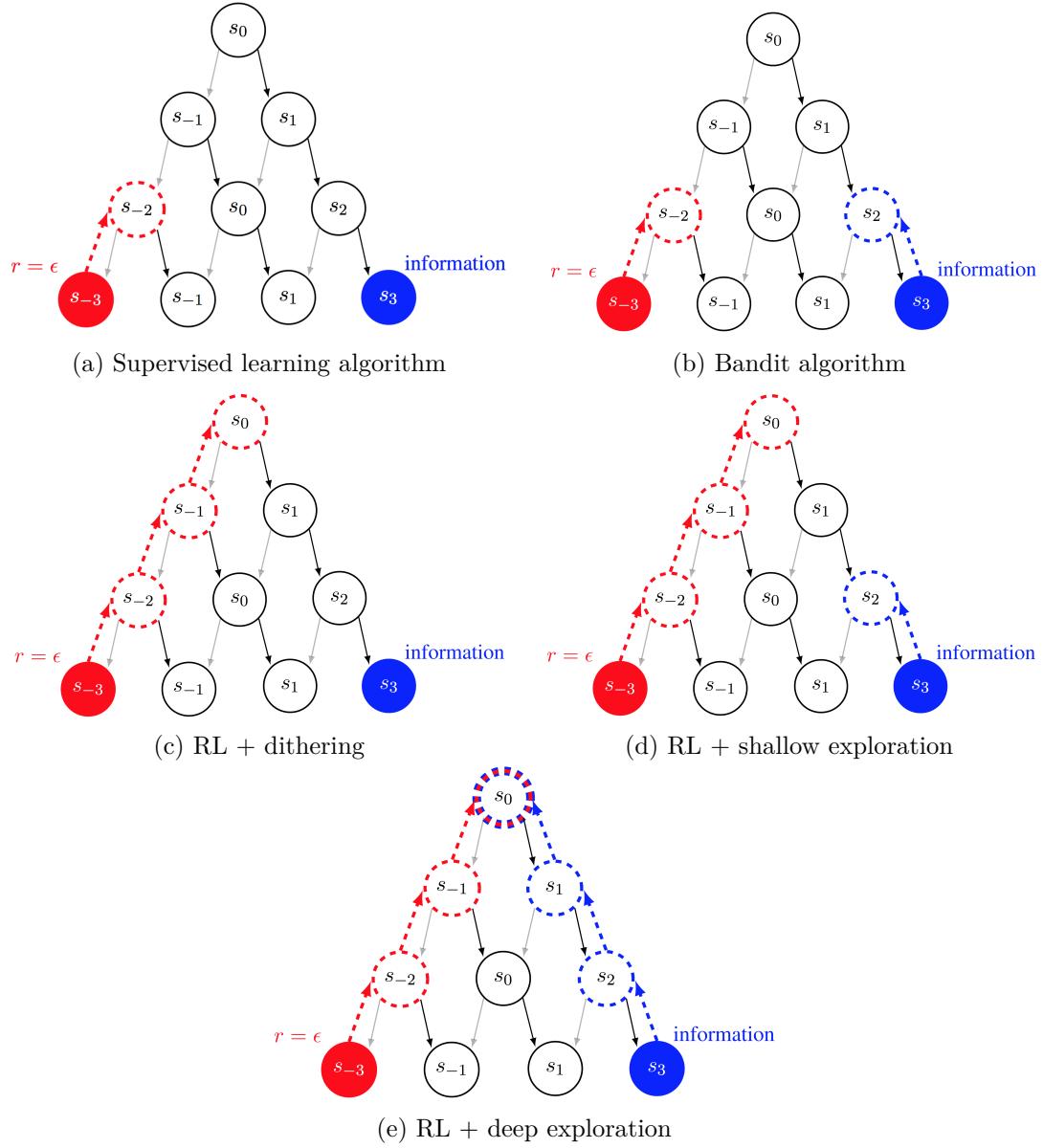


Figure 1.2: Planning, learning and exploration for the environment in Figure 1.1.

data which is potentially informative. Perhaps the most principled approach to statistically efficient learning is given by the Bayesian perspective. In this framework the agent models its uncertainty by a prior distribution over the environment. The agent then plans ahead to balance the potential information and rewards of future actions and observations. This framing is principled and effective, but for anything beyond tiny problems it will be computationally intractable, since the amount of computation grows exponentially in the problem size and lookahead horizon. For this reason, most practical implementations default to dithering strategies for exploration which remain tractable even in large domains with generalization.

In this thesis we present an alternative approach to generalization and exploration via randomized value functions. At a high level, this algorithm replaces a point estimate for the value of each action by a *distribution* over the potential values of each action. At each episode of interaction the agent will then sample a realization of this value function from its distribution and then follow the actions which are optimal according to this *sample* for the duration of this episode. If the distribution of this randomized value function is designed to effectively capture the agent's uncertainty over the potential value of each action, then this algorithm can drive efficient and deep exploration. Our algorithm is inspired by the Thompson sampling, or probability matching, heuristic for multi-armed bandits where an agent randomly chooses an action according to the probability that it is the optimal action. We show that this simple algorithm provides a unifying approach to statistically and computationally efficient reinforcement learning.

1.4 Organization

We begin our investigation to efficient reinforcement learning in the setting of tabula rasa, or blank slate environments. In this formulation the agent has little or no prior knowledge, but can learn through its interaction with the environment. This formulation is usually not practical, since it does not allow for generalization across learning the environment. However, it does allow us to study the RL problem in a simpler setting for analysis and also in a general framework for learning *any* underlying system. **Chapter 2** introduces this tabula rasa problem formulation in terms of an unknown finite Markov decision process (MDP). We outline several standard approaches to learning from tabula rasa 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 analyse 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 simulation.

Although the tabula rasa setting is important for insight and analysis, practical reinforcement learning in large scale systems will require generalization. The second part of our thesis focuses on settings where the underlying environment has some low-dimensional structure. 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. In both settings, we show that the same simple and intuitive algorithm PSRL satisfies these strong regret bounds and all 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 sucessfully 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 fucntion representation for the value function. Our algorithm, *randomized least squared value iteration* (RLSVI) approximates the intractable posterior over value functions by a simple Gaussian posterior. 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 over linear basis functions. 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 this approach to deep exploration via randomized value functions can be effectively combined with value functions with nonlinear generalization. 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 generlization 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 thesis,

- We bring attention to the importance of deep exploration. Crucially, we give a clear name to this concept 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 computationally tractable algorithm for reinforcement learning from tabula rasa with $\tilde{O}(\sqrt{SAT})$ bounds on expected regret, where S is the number of states, A is the number of actions and T is the time elapsed.
- 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 this approach can lead to exponentially faster learning in complex domains with generalization. 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 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 will learn, the amount of data and computational resources which they require is completely intractable for anything beyond the simplest of settings. In fact, we will go further than this prove lower bounds for learning without prior knowledge no matter what algorithm you use. Although tabula rasa reinforcement learning is not a practical approach to large scale problems, it will provide an invaluable opportunity to develop understanding and analysis in a simple setting. In the remainder of this thesis we will extend these insights to more practical problems.

In this chapter, we formalize the tabular RL problem in terms of an unknown finite Markov decision process (MDP). We review some standard definitions and results from the literature together with several criteria for efficient reinforcement learning. Next we establish several lower bounds which apply to *any* algorithm for RL, 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. $\mathcal{S} = \{1, \dots, S\}$ is the state space, $\mathcal{A} = \{1, \dots, A\}$ 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 $h = 1, \dots, H$ within an episode the agent observes state $s_h \in \mathcal{S}$, selects action $a_h \in \mathcal{A}$, receives a reward $r_h \sim R^M(s_h, a_h, h)$ in $[0, 1]$ and transitions to a new state $s_{h+1} \sim P^M(s_h, a_h, h)$. We say that the MDP M is time-homogenous if $R^M(s, a, h) = R^M(s, a, 1)$ and $P^M(s, a, h) = P^M(s, a, 1)$ for all $h = 1, \dots, H$. 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 $h = 1, \dots, H$ to an action $a \in \mathcal{A}$. For each MDP M and policy μ we define the Q-value function from step h :

$$Q_{\mu,h}^M(s, a) := \mathbb{E}_{M,\mu} \left[\sum_{j=h}^H \bar{r}^M(s_j, a_j, j) \mid s_h = s, a_h = a \right], \quad (2.1)$$

where $\bar{r}^M(s, a, h) = \mathbb{E}[r \mid r \sim R^M(s, a, h)]$. Theis 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

$$V_{\mu,h}^M(s) := Q_{\mu,h}^M(s, \mu(s, h)). \quad (2.2)$$

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

Let $\mathcal{H}_t = (s_1, a_1, r_1, \dots, s_{t-1}, a_{t-1}, r_{t-1})$ 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 $\{\pi_k \mid k = 1, 2, \dots\}$ 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 k th episode. We define the regret incurred by a reinforcement learning algorithm π up to time T to be:

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

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

$$\Delta_k := \sum_s \rho(s) (V_{\mu^*, 1}^{M^*}(s) - V_{\mu_k, 1}^{M^*}(s)) \quad (2.4)$$

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 The infinite horizon problem

Section 2.1 presents a reinforcement learning problem in terms of the sum of expected rewards through time, with a fixed horizon $H < \infty$. However, the setting without episodic reset and $H = \infty$ is also of significant interest in both reinforcement learning and control [11, 80]. For meaningful learning in this setting the MDP M must be time homogeneous. However 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. The key difference in the infinite horizon $H = \infty$ the definition of the value function, where (2.2) may not 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 value of a policy μ in an MDP M starting from s is given

$$V_{\mu, \gamma}^M(s) := \mathbb{E}_{M, \mu} \left[\sum_{t=1}^{\infty} \gamma^t \bar{r}^M(s_t, a_t) \middle| s_1 = s \right]. \quad (2.5)$$

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 $V_{\mu,\gamma}^M(s) \geq \max_{\mu'} V_{\mu,\gamma}^M(s)$ for all $s \in \mathcal{S}$. Actually, this problem is almost exactly equivalent to a finite horizon problem with $H \simeq \frac{1}{1-\gamma}$ [13, 61]. We choose to focus on the finite horizon problem since we find it is more amenable to clean analytical results, but almost all results in this setting can extend to discounted infinite horizon MDPs. We provide more details on why we prefer the convention with finite horizon and without discounting in Section 2.2.

For 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

$$V_{\mu,\text{ave}}^M(s) := \lim_{T \rightarrow \infty} \mathbb{E}_{M,\mu} \left[\frac{1}{T} \sum_{t=1}^T \bar{r}^M(s_t, a_t) \middle| s_1 = s \right]. \quad (2.6)$$

This expression is well-defined and bounded between zero and one. A policy μ is optimal for the MDP M if $V_{\mu,\text{ave}}^M(s) \geq \max_{\mu'} V_{\mu,\text{ave}}^M(s)$ for all $s \in \mathcal{S}$. Under some suitable technical conditions, for example if MDP M is weakly communicating [10], this optimal average value is independent of initial state so that $\rho(M) = \rho(M, s) = \max'_{\mu} V_{\mu}^M(s)$ for all $s \in \mathcal{S}$. Typically, extending results from finite horizon to infinite horizon involves nothing more than some increase in technical conditions and analytical rigour. However, in some specific situations the infinite horizon problem requires special consideration. We expose these technical difficulties in detail in Section 4.4.

2.2 A wish list for learning

We want to design algorithms which will learn to make good decisions through time and maximize the cumulative sum of rewards an agent receives in any environment. However, for any unknown MDP M^* the amount of rewards that can be attained by an algorithm π may be very different. In order to assess the performance of a learning algorithm given its environment we focus upon the regret as defined in equation 2.3. The regret quantifies the suboptimality 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 with respect to some risk measure [6] u over a family of MDPs \mathcal{M} up to time T we can control

$$u_{\mathcal{M}} [\text{Regret}(T, \pi, M^*)] \leq f_{\pi}(u_{\mathcal{M}}, T). \quad (2.7)$$

To make this abstract concept more concrete the family \mathcal{M} might include all the possible MDPs defined in Section 2.1. The risk measure u might be an expectation over a distribution of MDPs [71, 61], or an upper quantile of the regret [18, 47, 42]. The function $f_{\pi}(u_{\mathcal{M}}, T)$ provides the strength of our analytical guarantees. By our problem definition regret is always bounded $\text{Regret}(T, \pi, M^*) \leq T$ trivially. An algorithm π is asymptotically no-regret if $\lim_{T \rightarrow \infty} \frac{f(u_{\mathcal{M}}, T)}{T} = 0$, 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. The scaling of the bound in (2.7) determines the strength of the analytical guarantees.

2.2.1 Statistical efficiency

Another similar line of research has focussed upon various flavours of “probably approximately correct” (PAC) bounds for reinforcement learning [18, 47, 78, 77]. 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, with respect to some risk measure u over a family of MDPs \mathcal{M} , for any $\epsilon > 0$

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

This notion of statistical efficiency is very similar to notions of regret, but for some reason the two literatures are somewhat disconnected. 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 stronger than PAC results. We present more detailed discussion of these issues in Section 4.4.

Proposition 2.1 (Regret bounds imply PAC bounds).

Suppose that we have a regret bound $\text{Regret}(T) \leq KT^{1/\alpha}$ for some $\alpha > 1, K > 0$ and all $T > 0$. Then for all $T' > \left(\frac{K}{\epsilon}\right)^{\frac{\alpha}{\alpha-1}}$ the average regret per timestep must be less than ϵ .

Proof. Simple algebraic manipulation. \square

Proposition 2.2 (PAC bounds imply regret bounds).

Suppose that we have a PAC bound $\sum_{k=1}^{\infty} \mathbb{1}\{\Delta_k > \epsilon\} < L\epsilon^{-\beta}$ for some $\beta > 0, L > 0$ and all $\epsilon > 0$. Then for all $T > 0$ the regret is bounded $\text{Regret}(T) \leq (\beta + 1)\beta^{\frac{-\beta}{\beta+1}}L^{\frac{1}{\beta+1}}H^{\frac{1-\beta}{1+\beta}}T^{\frac{\beta}{\beta+1}}$.

Proof. For any time T the regret is bounded by $L\epsilon^{-\beta}H + \epsilon(\frac{T}{H} - L\epsilon^{-\beta})$. We use calculus to optimize this bound over choice of ϵ suggests a choice of $\left(\frac{T}{LH^2\beta}\right)^{\frac{-1}{\beta+1}}$. Substituting this expression into the above and ignoring the subdominant terms gives the desired bounds. \square

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 $\text{Regret}(T) \leq KT^{\frac{1}{2}}$ implies a PAC bound $O(K^2\epsilon^{-2})$. However, a PAC bound $O(K^2\epsilon^2)$ can only guarantee a regret bound $O(K^{\frac{2}{3}}T^{\frac{2}{3}})$, the dependence upon T is degraded. Superficially it might seem that although this bound is worse in terms of T , it improves the scaling in K . Actually, since the regret to time T is always bounded by T these guarantees are a strictly weaker result. To show this we consider any $\alpha > 0$

$$\text{Regret}(T) \leq KT^{\frac{1}{2}} \implies \text{Regret}(T) \leq \left(KT^{\frac{1}{2}} \times T^{\alpha}\right)^{\frac{1}{\alpha+1}} = K^{\frac{1}{\alpha+1}}T^{\frac{2\alpha+1}{2\alpha+2}},$$

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 thesis 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 sample complexity of the learning algorithm.

2.2.2 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 even 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.

This thesis presents an approach to reinforcement learning which aims to be both statistically and computationally efficient. Just as in supervised learning settings [26], these efficiency goals can be conflicting for reinforcement learning. 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 thesis is designing algorithms which synthesize statistically efficient approaches to reinforcement learning in a computationally tractable manner.

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 ,

$$\mathbb{E}[\Delta_k] = \mathbb{E}\left[V_{\mu^*,1}^{M^*} - V_{\mu_k,1}^{M^*}\right]. \quad (2.9)$$

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,

$$\mathbb{E}[\Delta_k] \leq \frac{1}{\lfloor T/H \rfloor} \mathbb{E}[\text{Regret}(T, \pi, M^*)]. \quad (2.10)$$

Proof. This result is established in Proposition 1 of [?]. \square

2.3 Fundamental limits to learning

Reinforcement learning is a very general framework for learning to make good decisions in an unknown environment. Section 2.2 lists several criteria for algorithms, for example the regret, which we seek to optimize through efficient algorithm design. In this section we 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”. We review a selection of such lower bounds below. 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 a time-homogeneous MDP M^ as per Section 2.1 such that the regret to time T is bounded,*

$$\text{Regret}(T, \pi, M^*) = \Omega(\sqrt{HSAT}). \quad (2.11)$$

Proof. This result can be obtained from the lower bounds in multi-armed bandits [19], or as a specializaiton of the more general results in infinite horizon MDPs [42]. Previous works have claimed lower bounds in the infinite hoirzon setting which, if they were correct, would imply lower bounds $\Omega(H\sqrt{SAT})$ [10]. However, the analysis in this paper is not rigorous

and a more careful examination suggests this results is not correct. For time-inhomogenous MDPs it is possible to improve these lower bounds to $\Omega(H\sqrt{SAT})$. \square

Corollary 2.1 (Lower bounds on Bayesian regret).

For any learning algorithm π , there exists a distribution over time-homogeneous MDPs such that expected regret to time T is bounded,

$$\mathbb{E} [\text{Regret}(T, \pi, M^*)] = \Omega(\sqrt{HSA\bar{T}}). \quad (2.12)$$

Proof. Consider a family of priors over the MDPs from Theorem 2.1 [19]. \square

Corollary 2.2 (Lower bounds on statistical efficiency).

For any learning algorithm π , there exists a time-homogeneous MDP M^ the number of episodes with regret at least ϵ is bounded,*

$$\mathbb{E} \left[\sum_{k=1}^{\infty} \mathbb{1}\{\Delta_k > \epsilon\} \right] = \Omega(HSA\epsilon^{-2}). \quad (2.13)$$

Proof. This result follows from the bounds on regret [42] and is also established in [23]. \square

We can establish similar bounds on the computational cost of solving for the optimal policy in a fixed and known MDP M^* .

Theorem 2.2 (Computational cost of solving a known MDP).

For any planning method, there exists an MDP M^ such that the cost of solving for the optimal policy of M^* is bounded $\Omega(HS^2A)$.*

Proof. The optimal policy may require exact dynamic programming [13]. \square

This thesis aims 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 [26].

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 with by describing a family of environments that highlight the need for deep exploration in RL. This spirit of these environments is similar to *RiverSwim* [78]. We present an illustration of this environment in Figure 2.1.

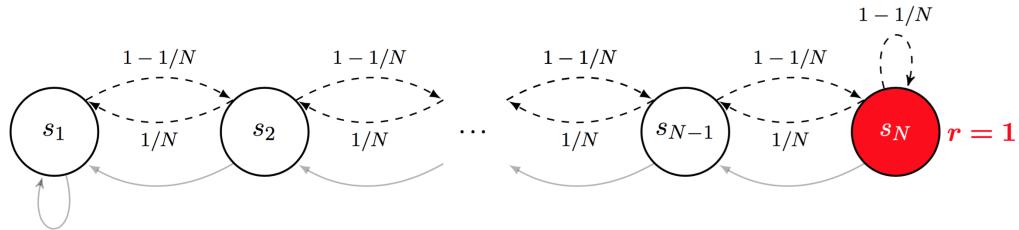


Figure 2.1: An MDP where algorithms without deep exploration are highly inefficient.

This environment is made up of a long chain of states $\mathcal{S} = \{1, \dots, N\}$. 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 will consider several families of algorithms for reinforcement learning 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 [79].

2.4.1 Greedy control

The most common approach to reinforcement learning in practical systems 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

```

1: Input: Point estimator for MDPs  $\phi$ .
2: for episode  $k = 1, 2, \dots$  do
3:   Estimate MDP  $\hat{M}_k = \phi(\mathcal{H}_{k1})$ 
4:   compute  $\mu_k \in \arg \max_{\mu} V_{\mu,1}^{\hat{M}_k}$ 
5:   for time  $h = 1, 2, \dots, H$  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 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 this approach [72, 81, 84]. 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 [85].

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 $0 < \epsilon \ll \frac{1}{N}$. A greedy policy 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 to learn an optimal policy. 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.

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 \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 [80, 57]. 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 $\epsilon, \eta \rightarrow 0$ at an appropriate rate) these dithering strategies may even be asymptotically no-regret [88]. However, dithering strategies can be remarkably statistically inefficient and make take an exponentially long time to learn [52, 71, 18]. Example 2.1 highlights a simple example where this can be the case.

Example 2.1 (Dithering strategies may take exponentially long to learn).

For a finite MDP $M = (\mathcal{S}, \mathcal{A}, R, P, H, \rho)$ dithering strategy for exploration can guarantee the number of steps until the average regret is less than $\epsilon = 0.1$ in fewer than $T^{\min} = 2^{S-1} - 1$.

Proof. Consider the example MDP from Figure 2.1. 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 $\mathbb{E}[k^*] > 2^{S-1} - 1$. \square

Algorithm 2.2 RL with ϵ -greedy exploration

```

1: Input: Point estimator for MDPs  $\phi$ ,  $\epsilon \in [0, 1]$ 
2: for episode  $k = 1, 2, \dots$  do
3:   Estimate MDP  $\hat{M}_k = \phi(\mathcal{H}_{k1})$ 
4:   compute  $\mu_k \in \arg \max_{\mu} V_{\mu,1}^{\hat{M}_k}$ 
5:   for time  $h = 1, 2, \dots, H$  do
6:     sample  $w_{kh} \sim \text{Unif}([0, 1])$ 
7:     if  $w_{kh} \leq \epsilon$  then
8:       take action  $a_{kh} \sim \text{Unif}(\mathcal{A})$ 
9:     else
10:      take action  $a_{kh} = \mu_k(s_{kh}, h)$ 
11:    end if
12:    observe  $r_{kh}$  and  $s_{kh+1}$ 
13:    update  $\mathcal{H}_{kh} = \mathcal{H}_{kh} \cup (a_{kh}, r_{kh}, s_{kh+1})$ 
14:  end for
15: end for

```

Algorithm 2.3 RL with Boltzmann exploration

```

1: Input: Point estimator for MDPs  $\phi$ ,  $\eta > 0$ 
2: for episode  $k = 1, 2, \dots$  do
3:   Estimate MDP  $\hat{M}_k = \phi(\mathcal{H}_{k1})$ 
4:   compute  $\mu_k \in \arg \max_{\mu} V_{\mu,1}^{\hat{M}_k}$ 
5:   for time  $h = 1, 2, \dots, H$  do
6:     sample  $\tilde{a} \propto \exp(Q_{\mu_k,h}^{\hat{M}_k}(s_{kh}, a)/\eta)_{a \in \mathcal{A}}$ 
7:     take action  $a_{kh} = \tilde{a}$ 
8:     observe  $r_{kh}$  and  $s_{kh+1}$ 
9:     update  $\mathcal{H}_{kh} = \mathcal{H}_{kh} \cup (a_{kh}, r_{kh}, s_{kh+1})$ 
10:   end for
11: end for

```

On optimistic initialization

One common practical approach to improve the performance for dithering methods for exploration is through “optimistic initialization” [80]. Here all states and actions are imagined to have a maximally high value, which is only adjusted downwards through experience. This can incentivize efficient exploration, but only if special care is taken to maintain optimistic estimates in a carefully balanced manner. We examine precisely this approach in Chapter 3. Naive optimistic initialization may never learn the optimal policy in stochastic environments, and even when combined with dithering may take exponentially long [52].

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.

One natural approach to remedy this shortcoming is through exploration in the policy rather than the action space [90, 74, 55]. 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).

For a finite MDP $M = (\mathcal{S}, \mathcal{A}, R, P, H, \rho)$ no pure policy search algorithm can guarantee the number of steps until the average regret is less than $\epsilon = 0.1$ in fewer than $T^{\min} = 2^{SH}$.

Proof. Consider the example MDP from Figure 2.1. 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 number of possible policies is generally 2^{SH} . It follows that $\mathbb{E}[k^*] > 2^{SH}$. \square

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 these algorithms, the agent explicitly builds a lookahead tree to imagine possible future states and actions. The agent then simulates these future outcomes, either with access to a generative model of the environment [46, 50], or some Bayesian posterior belief [87, 35]. 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, these 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 [86]. However, these methods and the Bayes-optimal framework for reinforcement learning all share several shortcomings.

First, these algorithms make heavy use of an accurate generative model (or prior description) of the environment. In many settings this may not be available without significant data gathering in the real environment. Further, 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 [36]. Second, these algorithms are extremely computationally expensive. In general, tree-based lookahead methods require $\Omega(A^H)$ calls to the generative model [44]. Approximations which use truncated computational approximations can fail spectacularly badly and may require super-exponential amounts of time to learn [58].

2.4.5 Towards efficient reinforcement learning

In this section we have outlined several broad algorithmic approaches to reinforcement learning. Perhaps surprisingly, 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. However, 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” [58]. 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 behaviour approaches optimality. This principle can successfully guide efficient reinforcement learning.

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 [42, 23]. 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 point estimates. 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 thesis.

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 learn and control an unknown system. The optimistic principle is simple, consider the set of all statistically plausible environments and plan for the policy which would obtain the best possible rewards in the best of these possible environments. 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  $\Phi$ 
2: for episode  $k = 1, 2, \dots$  do
3:   form confidence set  $\mathcal{M}_k \sim \Phi(\cdot | \mathcal{H}_{k1})$ 
4:   compute  $\mu_k \in \arg \max_{\mu, M \in \mathcal{M}_k} V_{\mu, 1}^M$ 
5:   for time  $h = 1, 2, \dots, H$  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; this 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 [78, 42, 10].

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 [52, 51, 5, 18, 42]. We will now outline the general flavour 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 $V_{\mu^{M^*},1}^{M^*}$. 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 write M_k for the optimistic MDP $\in \arg \max_{\mu, M \in \mathcal{M}_k} V_{\mu,1}^M$ and condense the value $V_{*,1}^*$ for $V_{\mu^{M^*}}^{M^*}$ and similarly for $V_{k,1}^*$ for $V_{\mu_k,1}^{M_k}$. Using this notation we can decompose the per episode regret into two components

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

The regret from optimism Δ_k^{opt} is the difference in value between the true optimal value function $V_{*,1}^*$ and the *imagined* optimal value function $V_{k,1}^k = \max_{\mu, M \in \mathcal{M}_k} V_{\mu,1}^M$. For a well

designed optimistic algorithm, the true MDP M^* lies within the confidence set \mathcal{M}_k with high probability and so $\Delta_k^{\text{opt}} \leq 0$. If the confidence sets hold we can bound the regret by the regret from concentration $\Delta_k \leq \Delta_k^{\text{conc}}$. Crucially, Δ_k^{conc} does not depend upon the unknown optimal policy μ^* , but only on 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 $V_{k,1}^*$ with standard supervised learning results.

3.1.2 Near-optimal regret bounds with UCRL

We now reproduce a simple optimistic algorithm 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 [42].

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 $R(s, a, h)$ and transition $P(s, a, h)$ 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 guarantees 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, \dots, y_n be IID samples from a distribution P^* over a finite set $\mathcal{S} = \{1, \dots, S\}$. 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^* to the empirical estimate \hat{P} is bounded in probability

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

We can rearrange this result to give bounds which hold with high probability, for any $\delta > 0$

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

Proof. This is a relaxation of the result proved by Weissman [89]. \square

We state our next result in terms of sub Gaussian random variables [21], which is a technical property for random variables which regulates the tail behaviour. 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 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 mean zero and bounded $\in [-1, 1]$.

Lemma 3.2 (Tail bounds for sub Gaussian random variables).

Let x_1, \dots, x_n be independent samples from sub Gaussian random variables. For any $\epsilon > 0$

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

We can rearrange this result to give bounds which hold with high probability, for any $\delta > 0$

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

Proof. This is a standard result for sub Gaussian random variables [21]. \square

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 at that timestep prior to episode k

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

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

```

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

```

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, regardless of the exploratory policies μ_1, μ_2, \dots 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 $\delta > 0$

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

Proof. Combine Lemmas 3.1 and 3.2 together with a naive union bound over all states, actions, timesteps and episodes. See [42] for more details. \square

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 [78, 42]. This algorithm increases the computational cost by a factor of S versus solving a single MDP.

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 we define for any MDP $M = (\mathcal{S}, \mathcal{A}, R, P, H, \rho)$ and policy μ to act upon value functions $V : \mathcal{S} \rightarrow \mathbb{R}$ by

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

This returns the expected value of state s where we follow the policy μ under the laws of M for one timestep from h . 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 $x_{kh} := (s_{kh}, \mu_k(s_{kh}, h), h)$ under the policy μ_k .

We can now 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 concentratin error into the error in estimation of \bar{r}^*, P^*

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

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 this argument to express the concentration error

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

The term from the martingale difference d_{kh} can be controlled by standard concentration results such as Azuma-Hoeffding [16] and bounded $\tilde{O}(H\sqrt{T})$. Since this term is ultimately subdominant [42] 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 \bar{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

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

We conclude our argument by upper bounding the expression in equation (3.11) over any possible trajectory of states and actions $\mathcal{H}_{11}, \mathcal{H}_{21}, \dots$. 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 $\sum_{k=1}^{[T/H]} \sum_{h=1}^H n_{kh} = T$. 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 [61, 63, 62].

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 $\delta > 0$ with probability at least $1 - \delta$ the regret of UCRL is bounded*

$$\text{Regret}(T, \pi^{\text{UCRL}}, M^*) = O\left(H^{\frac{3}{2}} S \sqrt{AT \log(T/\delta)}\right). \quad (3.12)$$

Further, in the case of time homogeneous MDP we can improve these bounds

$$\text{Regret}(T, \pi^{\text{UCRL}}, M^*) = O\left(H S \sqrt{AT \log(T/\delta)}\right). \quad (3.13)$$

Theorem 3.1 is a significant statistical guarantee as its scaling is close to the fundamental lower bound $\Omega(\sqrt{HSAT})$. 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 [42]. For more details on these lower bounds see Section 2.3.

3.2 Shortcomings of optimistic algorithms

Algorithms in the style of UCRL [9, 10, 42, 23] represent the culmination of many years of research in optimistic reinforcement learning [18, 47, 78, 77]. The construction of the confidence sets \mathcal{M}_k and the optimistic planning step have been honed over many years 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 actual performance of the algorithm. Second, we show that even if it were possible to design better confidence

sets for optimistic algorithms, this would typically lead to a computationally intractable optimistic planning step.

3.2.1 Inefficient confidence sets

The statistical efficiency of these optimistic algorithms is driven by the quality of the analytical concentration results in the spirit of Lemmas 3.1 and 3.2. For example, we can improve upon the confidence sets for transition functions by replacing Lemma 3.1 with more nuanced results that include second order information [23] or even the entire moment hierarchy via Kullback-Leibler divergence to give KL-UCRL [29]. These improved confidence sets will lead to better algorithms, since the estimates of the rewards and transitions at each (s, a, h) are more appropriately calibrated to the data at hand. Better confidence sets in each (s, a, h) can lead to better performance and better bounds but will not change the overall $\tilde{O}(S\sqrt{AT})$ scaling [29]. In fact, there is a more fundamental flaw in the design of these algorithms which can cause even greater inefficiency in large systems, the union bound which optimizes optimistically over each (s, a, h) simultaneously.

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 optimistic estimates for the rewards $\bar{r}_k(s, a, h)$ and transitions $P_k(s, a, h)$ simultaneously for each (s, a, h) . 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, h) , but it has largely ignored the effects of this over optimism that comes from the union bound.

In a system with many states and actions, this union bound can be extremely inefficient. If each $\bar{r}_k(s, a, h), P_k(s, a, h)$ is taken individually at the $1 - \delta$ optimistic quantile, the resulting MDP $M_k = (\mathcal{S}, \mathcal{A}, R_k, P_k, H)$ will be at optimistic at the $1 - \delta^{2SAH}$ quantile. This resulting MDP will be exponentially less likely than any of the individual components. To see this problem geometrically, UCRL optimizes optimistically over the union of optimistic estimates for each (s, a, h) . In effect this creates a hypercube in dimension SAH , since we allow for simultaneously optimistic estimates in every component. An efficient optimistic algorithm should be optimistic in each component independently, or more like a hypersphere, rather than hypercube. Intuitively this may not seem like a big difference,

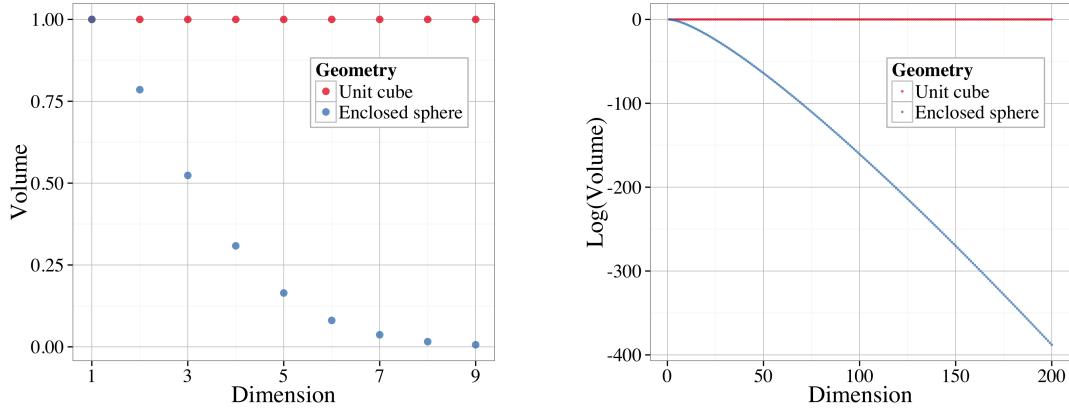


Figure 3.1: In high dimensions a sphere fills an exponentially small portion of the enclosing cube.

however for high dimensions the sphere takes up only an exponentially small portion of the enclosing cube [39]. This means that algorithms such as UCRL which build up optimistic estimates for each state and action independently and take a union bound over them all may end up over-optimistic by an exponential factor.

The correct geometry for confidence sets in \mathcal{M}_k should be ellipsoidal based upon 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, h) individually to give improved ellipsoidal, rather than L_1 , confidence balls around their estimates. These changes do nothing to improve the union bound across states and actions and so KL-UCRL effectively induces hypercylindrical confidence sets and no change to the overall scaling of the optimistic algorithm. However, 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 Intractable optimistic optimization

The hyperrectangular 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 hyperrectangular structure is absolutely crucial for the efficient computational implementation via extended value iteration [77, 42]. The hyperrectangular \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, h) is not a tractable problem [69, 71, 70, 29].

So here we have the bad news for optimistic algorithms. First, we do not have the analytical tools to design effective confidence sets for MDPs. Worse still, we can see 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 be faced with an intractable optimization problem. One 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 [51, 5, 7, 18, 47, 58, 50] and even provably near optimal algorithms such as UCRL2 [42]. At the same time we have outlined several fundamental shortcomings of these algorithms in terms of both statistical and computational efficiency. In this thesis we will present one way to move beyond these limits of existing optimistic algorithms. To do this we require a more nuanced approach to optimism, one that works in terms of optimistic *distributions* rather than point estimates.

3.3 Stochastic optimism

Early approaches to optimistic exploration in reinforcement learning categorized states and actions as either “known” or “unknown” [18, 47]. 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 [78, 77, 9, 42]. We now present a natural generalization of the optimistic principle from point estimates to optimistic random variables in terms of the entire distribution.

Definition 3.1 (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 $u : \mathbb{R} \rightarrow \mathbb{R}$:

$$\mathbb{E}[u(X)] \geq \mathbb{E}[u(Y)]. \quad (3.14)$$

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

For brevity if $X \succ_{\text{so}} Y$ we will sometimes write that X is optimistic for Y . This characterization of optimism is closely linked to “second order stochastic dominance” (SSD) [38]. In fact, the two conditions are in some sense dual in that $X \succ_{\text{so}} Y$ 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. $X \succ_{\text{so}} Y$
2. For any $u : \mathbb{R} \rightarrow \mathbb{R}$ 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 $\mathbb{E}[W|Y + A] = 0$ for all values $y + a$.

Proof. This follows from a simple integration by parts [38]. □

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. For example, if $X \sim N(\mu_X, \sigma_X^2)$ and $Y \sim N(\mu_Y, \sigma_Y^2)$ then $X \succ_{\text{so}} Y$ 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 3.2 (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 $t > 0$:

$$\mathbb{E}[\exp(tX)] \geq \mathbb{E}[\exp(tY)]. \quad (3.15)$$

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

Definition 3.3 (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:

$$F_X(s) \geq F_Y(s) \iff s \leq a. \quad (3.16)$$

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

$$X \succ_{\text{sc}} Y \iff X \succ_{\text{so}} Y \iff X \succ_{\text{mgf}} Y \iff \mu_X \geq \mu_Y \text{ and } \sigma_X \geq \sigma_Y. \quad (3.17)$$

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

$$X \succ_{\text{sc}} Y \implies X \succ_{\text{so}} Y \implies X \succ_{\text{mgf}} Y. \quad (3.18)$$

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

Suppose $X \succ_{\text{so}} Y$, 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 $X \succ_{\text{mgf}} Y$. \square

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 [21] and $X \succ_{\text{mgf}} Y$. 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 . \square

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 $X \succ_{\text{so}} Y$, 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 $x = -1, 0, 1$. \square

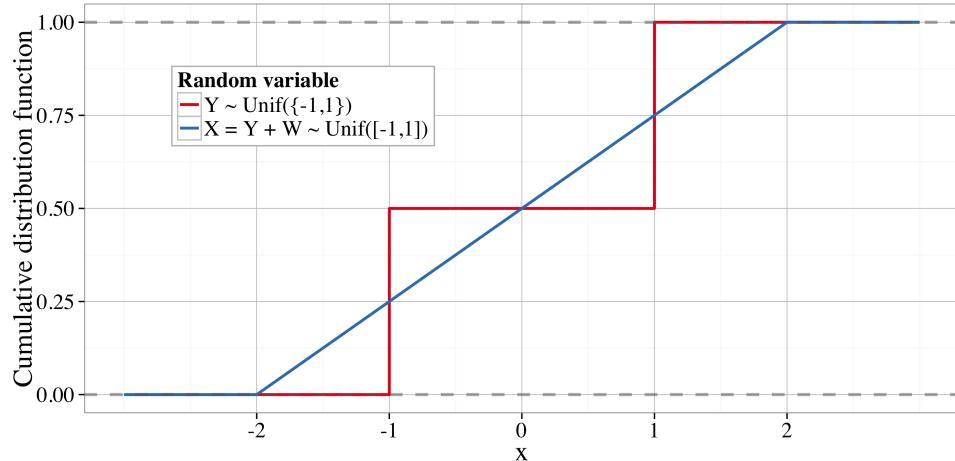


Figure 3.2: Stochastic optimism does not imply single crossing dominance

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 known as the multi-armed bandit. The standard optimistic approach to exploration, which for bandits is known as UCB rather than UCRL [52, 8] 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 hyperrectangle. 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 $r_k(a) \succ_{\text{so}} r^*(a)$ independently for each action a then Proposition 3.2 guarantees that the resulting *imagined* optimal reward is optimistic for the true optimal reward $\max_a r_k(a) \succ_{\text{so}} \max_a r^*(a)$. 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 [85, 71, 3] 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 [32]. 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 $D = y_1, \dots, y_n \sim N(\mu^*, 1)$, then the posterior distribution for $\mu^*|D \sim N(\mu, \sigma^2)$. These posterior parameters are given by

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

Example 3.4 (Updating a Dirichlet posterior from categorical data).

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

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

Lemma 3.3 concerns a posterior distribution over P^* and observe categorical results $s_i \in \mathcal{S} = \{1, \dots, S\}$. 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 $X \succ_{\text{so}} Y$.

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 $X \succ_{\text{sc}} \tilde{Y}$. Therefore, by Proposition 3.3 $X \succ_{\text{so}} Y$.

The main difficulty in this proof comes in establishing $X \succ_{\text{sc}} \tilde{Y}$. 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 thesis 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 $\gamma_1 \sim \text{Gamma}(k_1, \theta)$ and $\gamma_2 \sim \text{Gamma}(k_2, \theta)$, $\mathbb{E}[\gamma_1 | \gamma_1 + \gamma_2] = \frac{k_1}{k_1+k_2}(\gamma_1 + \gamma_2)$ 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 $V_1 \leq V_2 \leq \dots \leq V_S$. 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 $\gamma_1, \dots, \gamma_d$ independent, and let $\bar{\gamma} = \sum_{i=1}^d \gamma_i$, so that $P \equiv_D \gamma/\bar{\gamma}$. 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 $\bar{\gamma}^0 = \sum_{i=1}^d \gamma_i^0$ and $\bar{\gamma}^1 = \sum_{i=1}^d \gamma_i^1$, so that $1 - \tilde{p} \equiv_D \bar{\gamma}^0/\bar{\gamma}$ and $\tilde{p} \equiv_D \bar{\gamma}^1/\bar{\gamma}$. Couple these variables so that $1 - \tilde{p} = \bar{\gamma}^0/\bar{\gamma}$ and $\tilde{p} = \bar{\gamma}^1/\bar{\gamma}$.

We can now say

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

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, $X \succ_{\text{so}} \tilde{Y} \succ_{\text{so}} Y$. 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 $\tilde{Y} \sim \text{Beta}(\alpha, \beta)$ for any $\alpha > 0, \beta > 0$ and $x \sim N\left(\mu = \frac{\alpha}{\alpha+\beta}, \sigma^2 = \frac{1}{\alpha+\beta}\right)$. Then, $X \succ_{\text{sc}} \tilde{Y}$ 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 [68], 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 will 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:

$$h(x) := l''_N(x) - l''_B(x) = \frac{\alpha - 1}{x^2} + \frac{\beta - 1}{(1 - x)^2} - (\alpha + \beta) \tag{3.19}$$

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 analyse 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.

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

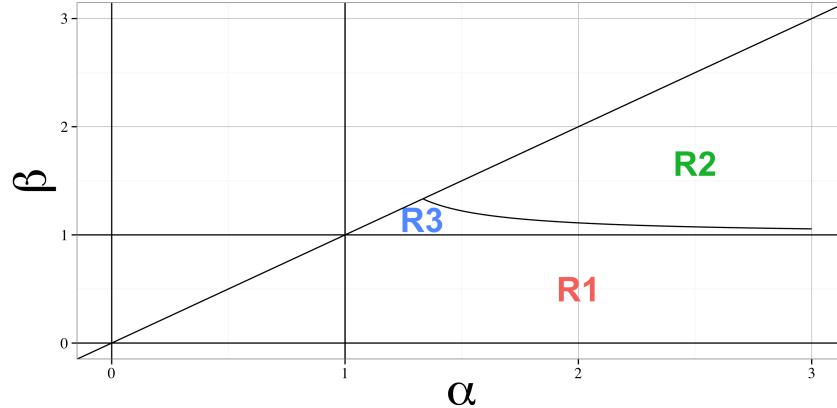


Figure 3.3: Parameter regions for proof.

These regions are displayed in Figure 3.3. The special case $\alpha = \beta = 1$ can be verified individually, in this case the PDFs do not intersect at any point. We will now deal with each region in turn.

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 $h'(x) = 0$

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

For convenience we write $K = (A/B)^{1/3} > 0$. 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 a simpler characterization of A, B that guarantees this condition:

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

And so we are done with this region.

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.

$$B(\alpha, \beta) = \int_{x=0}^1 x^{\alpha-1} (1-x)^{\beta-1} dx \leq \int_{x=0}^1 x^{\alpha-1} dx = \frac{1}{\alpha} \quad (3.20)$$

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

$$\text{Gap} : l_B(x_i) - l_N(x_i) \geq f(x_i) := \tilde{l}_B(x_i) - \max_x l_N(x) > 0. \quad (3.21)$$

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

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

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:

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

This provides a monotonicity result which states that both f_1, f_2 are minimized at 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.

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

We can use calculus to say that:

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

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. \square

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 4.3.

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 uncertainty over the environment through a prior distribution. At the start of each episode the agent samples an environment from its posterior distribution 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 [85].

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 demonstrate that PSRL attains statistical efficiency comparable to any possible optimistic algorithm but at a computational cost no greater than solving a single known MDP. In particular, we provide the first $\tilde{O}(\sqrt{SAT})$ 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 approach is also known as Thompson sampling after one of its original proponents for clinical trials [85].

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 [22, 73]. This prompted a surge of interest, and a variety of strong theoretical guarantees are now available in the bandit setting [71, 70, 3, 45, 4]. 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

```

1: Input: Prior distribution for MDPs  $\phi$ .
2: for episode  $k = 1, 2, \dots$  do
3:   for time  $h = 1, 2, \dots, H$  do
4:     sample MDP  $M_{kh} \sim \phi(\cdot | \mathcal{H}_{k1})$ 
5:     compute  $\mu_k \in \arg \max_{\mu} V_{\mu,h}^{M_{kh}}$ 
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
```

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 [71].

In bandits, this choice of directed exploration rather than dithering 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 [60]. 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 $\mathcal{S} = \{-N, -N-1, \dots, N\}$, 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 $\Omega(2^N) = \Omega(2^S)$. \square

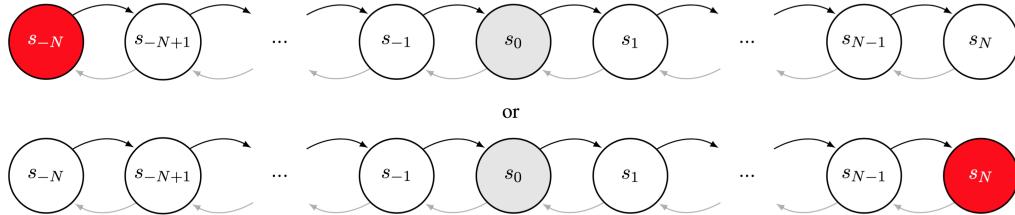


Figure 4.1: The possible MDPs for Example 4.1

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 [24, 31, 76] but this lack of deep exploration can lead to exponentially inefficient learning. However, 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.

PSRL was initially proposed by Strens under the name of Bayesian dynamic programming [79]. This older name is quite misleading however, since the Bayes-optimal exploration policy is available by dynamic programming in the belief MDP [37]. PSRL is neither a Bayes-optimal policy, nor does it approximate one directly instead it takes a single sample from the posterior and follows the policy which is optimal for that sample for the duration of the episode. For this reason we encourage the name *posterior sampling for reinforcement learning* [61]. We describe the algorithm below.

Algorithm 4.2 Posterior sampling for reinforcement learning

```

1: Input: Prior distribution for MDPs  $\phi$ .
2: for episode  $k = 1, 2, \dots$  do
3:   sample MDP  $M_k \sim \phi(\cdot | \mathcal{H}_{k1})$ 
4:   compute  $\mu_k \in \arg \max_{\mu} V_{\mu,h}^{M_k}$ 
5:   for time  $h = 1, 2, \dots, H$  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

```

PSRL is a very simple algorithm, 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. These are close the bare minimum requirements for any general approach to tabular reinforcement learning. This algorithm is extremely computationally efficient, since it requires exactly the same computation as solving a single known MDP. In fact, we will demonstrate that this simple heuristic is also extremely statistically efficient.

4.2 (Even more) efficient reinforcement learning

In this section we will demonstrate that PSRL satisfies strong bounds on the expected regret with two key results. 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. 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 [61] and mirrors the same observation in the bandit setting [71]; 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. These match the worst case frequentist lower bounds up to logarithmic factors [42].

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 [54, 53]. 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 [61]. 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 4.3.

4.2.1 PSRL matches the bounds for any optimistic algorithm

We will now present a general result that guarantees that relates the expected regret of PSRL to the optimal high probability bounds for any optimtistic algorithm. This result establishes that PSRL is in some sense equivalent to an well-tuned optimistic algorithm and 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 [56, 23] immediately apply to PSRL without need for any modification to the actual algorithm. By contrast, optimistic algorithms need good analysis before they can get good performance, PSRL works well regardless of what analysis you use. Together this means that PSRL attains statistical guarantees similar 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. This is an incredible result which we outline in Theorem 4.1.

Theorem 4.1 (PSRL matches any optimistic algorithm).

Let π^{opt} be any optimistic algorithm for reinforcement learning in the style of Algorithm 3.1. Suppose 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 $1 - \delta$*

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

Then, if the true MDP M^ is distributed according to ϕ this result implies a bound on the expected regret for PSRL for all timesteps $T > 0$*

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

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 achieve the scalings for UCRL in expectation. This result is the main contribution of the paper [61].

Corollary 4.1 (PSRL matches UCRL).

If ϕ is the distribution of M^ , then for any $T > 0$*

$$\mathbb{E} [\text{Regret}(T, \pi^{\text{PSRL}}, M^*)] = O\left(H^{\frac{3}{2}}S\sqrt{AT\log(T)}\right). \quad (4.3)$$

In the case that M^ is time homogeneous we can improve these bounds*

$$\mathbb{E} [\text{Regret}(T, \pi^{\text{PSRL}}, M^*)] = O\left(HS\sqrt{AT\log(T)}\right). \quad (4.4)$$

Proof of Theorem 4.1

Theorem 4.1 mirrors a similar result for multi-armed bandits that establishes Thompson sampling as equivalent to a well-tuned optimistic algorithm [71]. Although this result is powerful and extremely general, 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 simple 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 [25]. 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 \mathcal{H}_{k1} . Lemma 4.1 is an immediate consequence of this observation.

Lemma 4.1 (Posterior sampling).

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

$$\mathbb{E}[f(M_k)|\mathcal{H}_{k1}] = \mathbb{E}[f(M^*)|\mathcal{H}_{k1}]. \quad (4.5)$$

This simple lemma will allow us to relate the regret of any optimistic algorithm to PSRL. First we decompose the regret in episode k exactly as per (3.1). The only difference is that in this case, 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,

$$\begin{aligned} \Delta_k &= V_{\mu^{M^*},1}^{M^*} - V_{\mu_k,1}^{M^*} \\ &= V_{*,1}^* - V_{k,1}^* + V_{k,1}^k - V_{k,1}^k \\ &= \underbrace{V_{k,1}^k - V_{k,1}^*}_{\Delta_k^{\text{conc}}} + \underbrace{V_{*,1}^* - V_{k,1}^k}_{\Delta_k^{\text{opt}}}. \end{aligned} \quad (4.6)$$

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 .

In order to compare PSRL with any 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 $V_{\tilde{k},1}^{\tilde{k}}$ as the imagined optimistic value and $\tilde{\Delta}_k = V_{\tilde{k},1}^{\tilde{k}} - V_{k,1}^*$ 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 [42, 10],

$$\left\{ \text{Regret}(T, \pi^{\text{opt}}, M^*) \mid M^* \in \mathcal{M}_k(\mathcal{H}_{k1}) \text{ for all } k \right\} \leq \sum_{k=1}^{\lceil T/H \rceil} |\tilde{\Delta}_k^{\text{conc}}| \leq f(S, A, H, T, \delta). \quad (4.7)$$

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 (4.7), 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 .

$$\begin{aligned} \mathbb{E}[\text{Regret}(T, \pi^{\text{PSRL}}, M^*)] &\leq \mathbb{E}\left[\sum_{k=1}^{\lceil T/H \rceil} \mathbb{E}[\Delta_k^{\text{conc}} \mid \mathcal{H}_{k1}]\right] \\ &\leq \mathbb{E}\left[\sum_{k=1}^{\lceil T/H \rceil} \mathbb{E}\left[|V_{k,1}^k - V_{\tilde{k},1}^{\tilde{k}}| + |V_{\tilde{k},1}^{\tilde{k}} - V_{k,1}^*| \mid \mathcal{H}_{k1}\right]\right] \\ &\leq \mathbb{E}\left[\sum_{k=1}^{\lceil T/H \rceil} \mathbb{E}\left[2|\tilde{\Delta}_k^{\text{conc}}| \mid M^*, M_k \in \mathcal{M}_k(\mathcal{H}_{k1})\right]\right] + 2\delta T \end{aligned} \quad (4.8)$$

This completes the proof of Theorem 4.1. \square

4.2.2 PSRL satisfies the first $\tilde{O}(\sqrt{SAT})$ 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 analyse 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 $V_{k,1}^k$ is sampled from a distribution which is *stochastically optimistic* for true value function $V_{*,1}^*$, rather than the precise posterior distribution. The flavour of this result mirrors some of the bounds for Thompson sampling in bandits in the Frequentist perspective [3, 4]. Our work represents some of the first steps towards a robustness result for PSRL with a misspecified prior.

Theorem 4.2 (PSRL satisfies $\tilde{O}(\sqrt{SAT})$ regret bounds).

Let M^* be the true MDP distributed according to prior ϕ with any independent Dirichlet prior over transitions. Then the regret for PSRL is bounded

$$\mathbb{E} [\text{Regret}(T, \pi^{\text{PSRL}}, M^*)] = \tilde{O}\left(H^{\frac{3}{2}}\sqrt{SAT}\right). \quad (4.9)$$

Proof of Theorem 4.2

Our analysis begins with the familiar strategy of optimismistic 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

$$\mathbb{E} \left[\Delta_k^{\text{conc}} \mid \mathcal{H}_{k1} \right] = \mathbb{E} \left[\sum_{h=1}^H (\bar{r}_k - \bar{r}^*) (x_{kh}) + \sum_{h=1}^H ((P_k - P^*)(x_{kh}))^T V_{k,h+1}^k \mid \mathcal{H}_{k1} \right] \quad (4.10)$$

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 $\hat{r}_k(x) := \mathbb{E}[\bar{r}^*(x) \mid \mathcal{H}_{k1}]$ and also $\hat{P}_k(x) := \mathbb{E}[P^*(x) \mid \mathcal{H}_{k1}]$. 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 $\mathbb{E}[\bar{r}^*(x) \mid \mathcal{H}_{k1}] = \hat{r}_k(x)$ and $\mathbb{E}[P^*(x) \mid \mathcal{H}_{k1}] = \hat{P}_k(x)$. 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

$$w_k^R(x_{kh}) := \hat{r}_k(x_{kh}) - r_k(x_{kh}), \quad (4.11)$$

$$w_k^P(x_{kh}) := (\hat{P}_k(x_{kh}) - P_k(x_{kh}))^T V_{k,h+1}^k. \quad (4.12)$$

We can rewrite the concentration error (4.10) in terms of these variables,

$$\mathbb{E} \left[\Delta_k^{\text{conc}} \mid \mathcal{H}_{k1} \right] = \mathbb{E} \left[\sum_{h=1}^H (w_k^R(x_{kh}) + w_k^P(x_{kh})) \mid \mathcal{H}_{k1} \right]. \quad (4.13)$$

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

Up until this point, our analysis for Theorem 4.2 has followed the standard analysis for optimistic algorithms and in the spirit of Theorem 3.1. In addition we will bound the contribution from the unknown rewards w_k^R by Lemma 3.2 in exactly the same manner as UCRL to give a term $\tilde{O}(\sqrt{SAT})$ [42]. The only difference to our analysis for Theorem 4.2, and the key to removing a factor \sqrt{S} from our regret bounds, is how we control the deviation from unknown transitions w_k^P .

Typical optimistic algorithms, such as UCRL, use a Hölder inequality

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

They then bound the $L1$ deviation in the estimate of P_k and a universal upper bound on the span of the imagined optimal value via Lemma 3.1 [10, 42]. As we discuss in Section 3.2 these $L1$ confidence sets can be inefficient estimates for the distribution P^* . To remedy this problem, several other paper have addressed this bound via some other dual decomposition

$$|w_k^P(x_{kh})| \leq \|\hat{P}_k(x_{kh}) - P_k(x_{kh})\|_\nu \|V_{k,h+1}^k\|_{\nu^*}. \quad (4.15)$$

where ν and ν^* are some appropriate choice of norms which is better suited to the problem of estimating transitions and values in MDPs [56, 29]. These approaches can show some improvement over naive $L1$ confidence sets, but at the cost of an intractable optimization problem for optimistic planning.

However, even in terms of statistical efficiency these algorithms leave a lot to be desired. The equations (4.14) and (4.15) both bound the convergence in w_k^P , which is one-dimensional, by convergence in the estimated transition and value function, which are both S -dimensional. It seems inefficient that, in order to guarantee a good estimate of $P^T V \in \mathbb{R}$, we require a good estimate of $P \in \mathbb{R}^S$ and a good estimate of $V \in \mathbb{R}^S$. This looseness is the source of the sub-optimal scaling in S for all existing optimistic algorithms for RL, it is even shared by those variants which employ sophisticated dual norm approaches [56]. Lemma 3.3 allows us to bound this noise from transition directly in PSRL. This leads to the first regret bounds which attain the optimal $\tilde{O}(\sqrt{SAT})$ scaling¹.

Lemma 3.2 means we can bound the contribution from learning an unknown reward function, at any individual timestep, with probability at least $1 - \delta$,

$$\left\{ |w_k^R(x_{kh})| \mid \mathcal{H}_{k1} \right\} \leq \sqrt{\frac{2 \log(2/\delta)}{n_k(x_{kh})}}. \quad (4.16)$$

This term is identical to what we would obtain from the UCRL-style analysis of Theorem 4.1. This term is ultimately subdominant, the true difficulty in reinforcement learning is learning the transition function. For the transition noise w_k^P we first make use of Lemma 3.3. This result says that, for each x_{kh} , the contribution of the transition noise can be approximated optimistically by some Gaussian noise,

$$\tilde{w}_k^P(x_{kh}) \sim N\left(\mu = \hat{P}_k(x_{kh}), \sigma^2 = \frac{H^2}{n_k(x_{kh})}\right) \succ_{\text{so}} \left\{ |w_k^P(x_{kh})| \mid \mathcal{H}_{k1} \right\}. \quad (4.17)$$

Importantly, the rate at which this Gaussian optimistic bound converges $\tilde{w}_k^P \simeq O(H/\sqrt{n})$ does not depend on S . In particular, since \tilde{w}_k^P is Gaussian, we can apply Lemma 3.2 to say that at any individual timestep, with probability at least $1 - \delta$,

$$|\tilde{w}_k^P(x_{kh})| \leq H \sqrt{\frac{2 \log(2/\delta)}{n_k(x_{kh})}}. \quad (4.18)$$

We finish with a union bound over each (s, a, h) in each episode k and rescaling $\delta \leftarrow (SAHk^2T)$ and conclude with some manipulations of summation exactly as per Theorem

¹Technically we have only established this Lemma in the case where $\alpha_0(x_{kh})^T \mathbf{1} + n_k(x_{kh}) \geq 2$. However this term is incremented by one every time a given x_{kh} is visited. This means that this term can give us at worst an additive (and subdominant) $2H^2SA$ term to regret, regardless of prior

3.1 [42, 61]. Our analysis guarantees that the regret for PSRL is bounded,

$$\begin{aligned}
\mathbb{E} \left[\text{Regret}(T, \pi^{\text{PSRL}}, M^*) \right] &\leq \mathbb{E} \left[\sum_{k=1}^{\lceil T/H \rceil} \mathbb{E} [\Delta_k^{\text{conc}} \mid \mathcal{H}_{k1}] \right] \\
&\leq \sum_{k=1}^{\lceil T/H \rceil} \sum_{h=1}^H (1+H) \sqrt{\frac{4 \log(2SAHkT)}{n_k(x_{kh})}} + \frac{H\pi^2}{6} + 2H^2SA \\
&= \tilde{O} \left(H^{\frac{3}{2}} \sqrt{SAT} \right).
\end{aligned} \tag{4.19}$$

This completes the proof of Theorem 4.2. \square

In this chapter we have established state of the art statistical bounds for the performance of PSRL. However, it can be confusing to place the quality of these bounds in the context of the existing papers, each of which investigates slightly different guarantees. In particular, our bounds hold in expectation over a Bayesian prior and for MDPs with finite horizon H . We relate these Bayesian guarantees to frequentist notions of statistical efficiency in Section 4.3. We discuss the effects of learning with different horizon in Section 4.4, with special attention to the technical difficulties that arise in environments with infinite horizon. Both of these sections are technical and involved. We argue that, for the most part, these settings are intimately linked and provide several connections between these settings that are typically handled separately in the literature. Readers who are more interested in the practical performance of these algorithms should skip forward to Section 4.5, where we provide extensive experimental evaluation of these learning algorithms.

4.3 Relating Bayesian regret to frequentist guarantees

Theorems 4.1 and 4.2 establish state of the art bounds for PSRL in terms of the expected regret over *any* prior distribution of MDPs. These results are extremely general, and apply to an immense class of potential models. To accomodate 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 [42]. Typically, papers in the literature focus on either the Bayesian or the frequentist regret with little as if the two criteria were entirely separate considerations. Our next result [71, 61] 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 with non-zero probability under the prior distribution ϕ . Suppose that π is any learning algorithm such that the expected regret is bounded,

$$\mathbb{E} [\text{Regret}(T, \pi, M^*) \mid M^* \sim \phi] \leq D\sqrt{T},$$

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

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

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

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

We can then rearrange this expression to see that,

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

□

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 [43]. However, even with these maximally uninformative priors we have the problem that the probability of any specific MDP M^* will still be zero under a continuous prior. Although even a small region around a particular MDP will have non-zero mass under the uniform distribution, this mass will typically be exponentially small in the number of states and actions. As it stands, these Bayesian results for PSRL are not as strong in the worst case as the frequentist guarantees for UCRL [42, 61].

Part of the reason we cannot guarantee good frequentist performance is that we perform our analysis over an arbitrary family of priors ϕ . If the prior ϕ is misspecified to the true MDP M^* then it is intuitively clear that PSRL may not converge to the optimal policy. At

first sight this appears to be a failing of PSRL relative to UCRL, which satisfies frequentist bounds without an explicit prior. In fact, UCRL is similarly vulnerable to any MDP M^* which does not precisely fall within the categories of the formualtion in Section 2.1. For example, if the true MDP has rewards in $[0, 2]$ then UCRL may not learn the optimal policy, since its optimistic confidence sets are misspecified. In effect, UCRL operates over a specific prior class of MDPs just as PSRL, they just specify this prior structure in different ways.

A similar relationship has been studied in more detail for the multi-armed bandit, which is just a degenerate MDP without persistent consequences and $H = 1$. In this literature the corresponding result to Theorem 4.1 establishes the connection between posterior (Thompson) sampling and optimistic algorithms in terms of the Bayesian regret [71]. At the same time, more careful analysis has also established *frequentist* bounds on the regret for this same algorithm in certain settings and with certain choices of prior distribution [3, 45, 4]. These proofs are long and involved, but we believe that it will be possible to extend similar results to PSRL. One piece of insight which is encouraging for the prospect of thes frequentist bounds comes from our concept of *stochastic optimism*. In fact, in our proof of Theorem 4.2 we did not require that PSRL generate samples from the *true* posterior distribution, the only thing we required was that the sampled value functions were stochastically optimistic for the true posterior distribution. This mirrors some of the earlier results in bandit theory. Thompson sampling will perform well even the prior and likelihood are misspecified, as long as the posterior sampling distribution remains stochastically optimistic.

Optimistic posterior sampling via Gaussian PSRL

To make this point more clear we will consider a variant of PSRL that samples according to an optimistic Gaussian approximate posterior for $V_{k,1}^k$ even though the true posterior is not Gaussian. We call this algorithm Gaussian PSRL and we will see that, despite its completely misspecified posterior sampling methodology, this algorithm also satisfies the optimal $\tilde{O}(\sqrt{SAT})$ regret bounds from Theorem 4.2. In practice, we would not expect this algorithm to be as effective as PSRL, since it samples from misspecified confidence sets that do not exploit the natural structure of an MDP. However, this algorithm is interesting for several reasons. First, this algorithm is amenable to more simple analysis than PSRL directly, as we will see in section 4.4 we can provide a simple proof for the time-homogenous setting with $\tilde{O}(H\sqrt{SAT})$ bounds for *Gaussian PSRL* while the same result in PSRL is much

more involved. Second, and more importantly, Gaussian PSRL clearly demonstrates that exploration via posterior sampling can be provably efficient even when both the prior weight and likelihood are severely misspecified. The key to efficient exploration via randomized value functions is given by stochastic optimism, rather than the requirement that the posteriors are matched precisely. Finally, since Gaussian PSRL does not build up an explicit model of the transition dynamics it lends itself naturally to a form of value-based generalization. We investigate this potential in detail in Chapter 7.

Algorithm 4.3 Gaussian PSRL

```

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

```

Algorithm 4.4 Optimistic value sampling for Gaussian PSRL

```

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

```

Optimistic posterior sampling via multiple samples

Another approach to generating optimistic posterior samples for the value function can be obtained by generating *multiple* samples from the posterior and then selecting the sample with the highest estimated value. At the start of each episode the agent would sample

$M_k^1, \dots, M_k^J \sim \phi(\cdot | \mathcal{H}_k)$ and for each of these MDPs compute the imagined optimal value function. The agent then selects the action which is optimal with respect to the most optimistic value function. Intuitively, choosing the maximum over J samples roughly corresponds to a Bayesian $1 - 1/J$ optimistic quantile for the posterior distribution. We call this algorithm “optimistic posterior sampling for reinforcement learning” (OPSRL) and in the case $J = 1$ it reduces to PSRL. We note that, the original implementation of OPSRL [30] actually resamples an MDP every single timestep so might more appropriately be called “optimistic Thompson sampling for reinforcement learning” (OTSRL).

Through the tools of stochastic optimism we see that every single bound we established for PSRL can be adapted to apply to OPSRL for $J > 1$. The only difference to Theorem 4.1 and Theorem 4.2 is we must alter the δ to union bound each and every J sample, but this only affects the bounds up to logarithmic factors. Empirical analysis in [30] find that OTSRL outperformed TSRL with $J = 1$. This is not entirely surprising since, for $J > 1$ OTSRL can perform deep exploration and TSRL can not. To see why this is the case we imagine the setting of large J and low prior knowledge. Even if the agent resamples MDPs every single timestep the estimate for the value function will remain consistently high around the $1 - 1/J$ quantile, so you can perform consistent and directed exploration over multiple timesteps. By contrast with $J = 1$ in TSRL the policy will change drastically at every timestep and the agent may never commit to gain useful information, as per Example 4.1.

One downside of OPSRL versus PSRL is that it requires a factor J more computation. Although this computation is amenable to parallelization, it might be undesirable for large scale applications. Nevertheless, the approach may hold some advantages over vanilla PSRL with $J = 1$. We frame these advantages in terms of stochastic optimsim. PSRL generates samples which are optimistic for the true posterior, since they are equal in distribution. However, even a small mispecification to the posterior would mean that PSRL may no longer satisfy this relation. OPSRL generates samples which are more optimistic, since it takes the maximum over several samples. We believe that this algorithm may be more amenable to provably efficient analysis in the frequentist worst case. In particular, since this generates samples which are more optimistic, we might imagine that this algorithm is more robust to mis-specified prior and posterior distribution. However, we believe that a more practical approach to this problem is simply to add extra noise to the sampling distribution for $J = 1$. In effect this will accomplish the same goal of OPSRL and an

increase in optimism but without the increase in computational cost.

A more general result on stochastic optimism

Our analysis in Theorem 4.2 demonstrates some robustness in posterior sampling, so long as the sample is taken from a distribution which is stochastically optimistic for the true posterior. However, in order to guarantee this result we rely upon Lemma 3.3 which mechanically establishes that given *any* history of data \mathcal{H}_{k1} the sample is optimistic conditional upon that data. In general this is a very difficult property to establish and we do not believe that it is necessary for the algorithm to perform well. In Conjecture 4.1 we present a simple example that suggest the flavour of result we believe to be true.

Conjecture 4.1 (Posterior sampling and stochastic optimism).

Let M^* be a independent multi-armed bandit (i.e. $S = H = 1$) and with Gaussian mean rewards $\bar{r}^*(a) \sim N(\mu_0, \sigma_0^2)$. The realized rewards are drawn $r_t(a) = \bar{r}^*(a) + \epsilon_t(a)$, where $\epsilon_t(a) \sim F_t(a)$ is zero mean noise for each t and each a . We will not make any assumptions over the distribution $F(a)$ except to say that $Z \sim N(0, 1)$ is stochastically optimistic for $\epsilon_t(a)$ for all t and for all a . We write ϕ for this correct true prior distribution.

Now imagine that we run PSRL, which for $H = 1$ is just Thompson sampling, with a mis-specified prior $\tilde{\phi}$ that begins with the correct prior for mean rewards $\bar{r}^*(a) \sim N(\mu_0, \sigma_0^2)$ but updates the posterior as if errors were drawn from $\epsilon_t(a) \sim N(0, 1)$. We write $\tilde{\pi}^{\text{PSRL}}$ for this algorithm. Then, we conjecture that the regret of this misspecified algorithm is bounded,

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

Intuitively this result would say that it doesn't matter if the true underlying noise is actually Gaussian, so long as the Gaussian noise is optimistic for the true noise. By contrast, we know that employing Thompson sampling with too small an estimate for the underlying noise will be too greedy and may not learn the optimal policy.

Proof. We do not have a complete proof of this result, but we have some intuitive arguments that suggest a result in this flavour might be possible. We begin with an application of Proposition 3.2 for stochastically optimistic random variables. For any distribution $F_t(a)$ we could inject additional noise $w_t(a)$ at every timestep such that $\mathbb{E}[w_t(a)|r_t(a)] = 0$ but the

resulting $\tilde{r}_t(a) = r_t(a) + w_t(a) \sim N(\bar{r}^*(a), 1)$. 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 $\tilde{\phi}$. Theorem 4.1 and we could recover our bounds upon the expected regret [71].

The setting we want to investigate is where the algorithm uses this incorrect prior $\tilde{\phi}$ on the real environment ϕ . However, for any fixed number of samples from any arm a and any mean value $\bar{r}^*(a)$ 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 has proven extremely difficult. There is a subtle step in our proofs of Theorem 4.1 and 4.2 that conditions on any arbitrary history \mathcal{H}_{k1} . We show that, regardless of the history, the samples which PSRL generates will be optimistic for the true posterior. For 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^* . However, since the data the agent obtains is not independent of the noisy samples $\epsilon_t(a)$, the distribution of this expectation is quite difficult to control. Nonetheless, we believe that techniques similar to the existing literature on frequentist bounds [3] may provide an avenue to this sort of result. \square

We cannot quite complete the proof of Conjecture 4.1, but the flavour of the argument seems to mirror the rigorous analysis provided in some of the Thompson sampling literature [3]. We believe that it might be possible to obtain regret bounds for misspecified prior distributions, so long as the prior over value functions is optimistic and it is updated according to a likelihood which has noise which is optimistic (and therefore by Proposition 3.2 more spread out) than the true noise. We leave the analysis for this flavour of result in both bandits and reinforcement learning as an open problem for future research.

Lower bounds on Bayesian regret

The results in Section 2.3 establish lower bounds $\Omega(\sqrt{SAT})$ on the regret for any MDP M^* in the worst case [42]. This immediately implies a lower bound on the Bayesian regret $\Omega(\sqrt{HSAT})$ where the prior ϕ is structured over only these worst case MDPs [19]. We note that Theorem 4.2 gives Bayesian regret bounds that match this $\tilde{O}(\sqrt{SAT})$ scaling up to logarithmic factors over a wide family of priors ϕ . However, this family does not include the prior with established lower bounds $\Omega(\sqrt{HSAT})$ as described in [42]. For an arbitrary prior structure ϕ we cannot necessarily use the arguments in Theorem 4.2, which rely upon an independent Dirichlet prior across transitions. We still have the guarantees from Theorem 4.1, but so far these are only $\tilde{O}(S\sqrt{AT})$. It is an open question whether the lower bound of $\Omega(\sqrt{SAT})$ applies to the class of priors specified by Theorem 4.2.

4.4 On the horizon length and learning

So far in this thesis 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 [80]. 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 $H \simeq \frac{1}{1-\gamma}$ [44]. In general, the insights and analysis from the setting with finite episodes can be 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.

4.4.1 MDP connectedness

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 4.2.

Example 4.2 (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. One of these states is “heaven” in that the agent will receive the maximum reward of one for all time, the other one is “hell” and it will never get a reward again. The problem is that the agent does not know which one is which. Even if the agent knows the entire structure of the problem, 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 [44]. We find this formulation somewhat unsatisfying for two reasons. First, unlike regret bounds, strong sample complexity bounds do not indicate strong performance. In fact, as we can see in Example 4.2 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 of interest the imposition of the discount factor $\gamma \in [0, 1)$ is contrived and does not come from the actual environment. Overall the goal of PAC-MDP seems disconnected from the actual goal of the reinforcement learning 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 4.2 from their consideration. The precise requirements used in each paper are slightly different, these include [10],

- **Ergodic**, it is possible to reach any state from any other state under any policy.
- **Unichain**, every policy induces a single recurrent class plus a set of transient states.
- **Communicating**, for every $s_1, s_2 \in \mathcal{S}$ there is *some* policy that takes an agent from s_1 to s_2 .
- **Weakly communicating**, The state space \mathcal{S} decomposes into two sets: in the first, each state is reachable from every other state from every other state under some policy; in the second, all states are transient under all policies.

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. For an infinite horizon problem with discounting there is an effective horizon $H \simeq \frac{1}{1-\gamma}$. In the average cost problem without discounting there are multiple ways to think about this MDP connectedness. Two common quantities that emerge from these analyses are the span $\Psi(M^*)$ and the diameter $D(M^*)$.

Definition 4.1 (MDP span).

The span of an infinite horizon MDP M^ is defined,*

$$\Psi(M^*) := \max_{s,s' \in \mathcal{S}} |V_{*,\text{ave}}^*(s) - V_{*,\text{ave}}^*(s')| \quad (4.23)$$

Definition 4.2 (MDP diameter).

The diameter of an infinite horizon MDP M^ is defined,*

$$D(M^*) := \max_{s \neq s'} \min_{\mu} T_{s \rightarrow s'}^{M^*, \mu}, \quad (4.24)$$

where $T_{s \rightarrow s'}^{M^*, \mu}$ is the expected number of timesteps to get from s to s' under policy μ in the MDP M^* . We note that $D(M^*) > \Psi(M^*)$ and may even be exponentially larger.

Although we have presented their analysis for finite horizon MDPs, UCRL was originally analysed in the infinite horizon setting. In this setting the algorithm obtains regret bounds $\tilde{O}(DS\sqrt{AT})$ where the horizon H is replaced by the diameter D in Theorem 3.1 [42]. A subsequent analysis presents the REGAL algorithm that improves upon this algorithm to prioritize MDPs with lower mixing times, this improves the scaling to $\tilde{O}(\Psi S\sqrt{AT})$ but the resulting algorithm is not computationally tractable. Their paper presents a clear review of several measures of MDP connectedness and their interaction with the design of effective UCRL-style algorithms [10]. This paper also claimed to modify the lower bound on regret from $\Omega(\sqrt{DSAT})$ [42] to $\Omega(\Psi\sqrt{SAT})$, however as we outline in Section 2.3 this proof is not rigorous and seems to be incorrect.

Ultimately, most of the differences between the finite horizon setting and the infinite horizon setting come down to a case of analysis and book-keeping with no real change to the underlying performance of an implementable algorithm [10]. The one exception to this is the question of how and when to resample policies in the absence of episodic reset. We address this question in the next subsection.

4.4.2 Imposing an episode length

Example 4.1 demonstrates that Thompson sampling every timestep is not an effective algorithm for reinforcement learning. In fact, the same is true for a variant of UCRL that updates its confidence sets \mathcal{M}_{kh} every timestep [42]. The problem with both of these approaches is that they do not perform deep, or temporally extended, exploration. In the case of finite episodes we solve this problem by only sampling a new policy at the beginning of each episode and committing to that policy for the duration of the episode. When compared to the Bayes-optimal solution or some tree-based sampling methods this can be inefficient, since the agent must wait until the end of the episode to incorporate any new information. We provide an example of this phenomenon in Example 4.3

Example 4.3 (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 regret $\Omega(H)$.

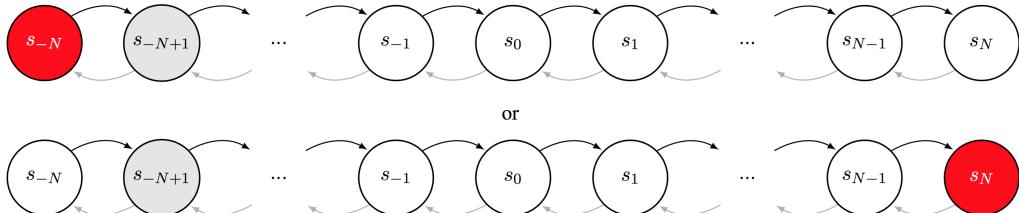


Figure 4.2: The possible MDPs for Example 4.3

Example 4.3 is investigated in more detail in [37]. PSRL can never learn in time $T < H$, but a general Bayes-optimal agent may be able to learn in a single step. However, although PSRL performs worse than optimal, even in this example designed to highlight the failings of PSRL it still satisfies the strong polynomial regret bounds of Theorem 4.2. Although the Bayes-optimal solution may be much better in a given prior structure ϕ , in general it cannot improve over these worst case scalings in S, A, T [19]. PSRL is not designed for efficient learning in $T < H$, in fact it does not do *any* learning within the first episode. PSRL is designed for good finite time performance, but only in the setting where T is relatively

large compared to any single episode H . If this is not the case, you should use a different algorithm and you must change the sort of bounds you are looking for.

The practical question remains, when should PSRL resample policies in a setting without episodic resets. One approach is to simply impose an artificial episode length of H and resample a policy every H timesteps regardless of the underlying system. This approach is guaranteed to perform well relative to any policy with timescale less than H , but might fail to learn an optimal policy over a longer horizon since the depth of its exploration is truncated by H . In addition, the algorithm may incur significant switching cost from switching policies at sampling timepoints. In the worst case this might contribute a linear term of the span Ψ every H timesteps. One strategy is to only resample at some distinguished initial state s_0 that is known to make up part of the optimal policy [33]. However, in a general MDP it is not guaranteed that such a state will exist and, where it does, the problem is effectively a finite horizon problem with the horizon H as the expected recurrent time to s_0 . Proving results in this setting is slightly more technical, but provides no real added value.

In the more general setting of MDPs which are not ergodic but are communicating the sampling strategy for exploration needs a little more care. The most common approach to this problem is known as “the doubling trick”[42]. The doubling trick dynamically grows the length of the episodes of fixed policy. For the infinite horizon average cost problem UCRL only resamples a new policy whenever it significantly improves its posterior understanding. In their proposed implementation the algorithm only begins a new policy when the total number of visits to any state and action has doubled during the current episode [42, 10]. Similar approaches to dynamically growing the episode length are common across various implementations [1]. These results are insightful and consistent with a more intuitive argument, an agent should only resample its exploration policy once it has substantially altered its understanding. Incorporating similar approaches to posterior sampling can be effective in practice [61, 2], however it is not possible to naively carry over the bounds from UCRL-style algorithms in the presence of dynamic episode sampling strategies. In particular the Lemma 4.1 cannot be applied to the average regret in an episode k that occurs at a random timestep, since the regret incurred during this episode is not $\sigma(\mathcal{H}_{k1})$ -measurable. This is a very delicate point which has not been addressed correctly in the existing literature [2].

Example 4.4 (The problem with dynamic episode lengths in PSRL).

Suppose that we employ a variant of the PSRL algorithm in an infinite horizon problem. The

algorithm uses some dynamic strategy to determine the length of each “episode”, which here is defined as a period of fixed policy. For clarity, we will consider the exact same doubling trick as UCRL2 which can be used to give $\tilde{O}(DS\sqrt{AT})$ regret bounds in communicating MDPs [42, 61]. Then, it is not possible to naively mirror the analysis of Theorem 4.1, since we do not have a corresponding result to Lemma 4.1.

The issue in this example is extremely subtle and rests on issues of $\sigma(\mathcal{H}_{k1})$ -measurability. For finite horizon MDPs, PSRL is guaranteed to follow a fixed policy for exactly H timesteps, this means that the expected cumulative value of the imagined MDP over episode k is $\sigma(\mathcal{H}_{k1})$ -measurable. However, if the length of an episode k is random and, in particular, if it depends on both the sampled MDP M_k and the true underlying MDP M^* then the expected cumulative mismatch in imagined optimal rewards is not measurable given \mathcal{H}_{k1} . This issue is technical and it might be possible to overcome this issue with improved analysis or additional assumptions, however a naive application of Lemma 4.1 to dynamic episode lengths is not correct. Some published papers have incorrectly overlooked this issue and so their results are not justified in a rigorous way [2].

We can give a simple illustration of this potential problem. Suppose the agent exists in an MDP such that the true MDP M^* is either M^+ or M^- with equal probability. The MDP M^+ has optimal average reward of $\rho(M^+) = 1$ and M^- has zero rewards and $\rho(M^-) = 0$. Imagine we set up a dynamic episode length such that if you sample a belief of M^+ , but the true MDP is M^- then the episode will last 1000 time steps. For all other combinations the episode will last only one timestep. At the end of the first episode the agent will know the true value of the MDP and all episodes last one step. We can construct an MDP like this with an imbalanced version of Example 4.3 where the agent resamples its policy only when it observes new information [61, 2, 37]. We write L_k for the length of episode k and $k(t)$ for the index of the current episode at time t .

Now, Lemma 4.1 implies that, for any fixed dataset \mathcal{H}_{k1} , $\mathbb{E}[\rho(M^*)|\mathcal{H}_{k1}] = \mathbb{E}[\rho(M_k)|\mathcal{H}_{k1}]$. Previous analysis [2] had previously argued that, for any $T > 0$,

$$\mathbb{E} \left[\sum_{t=1}^T (\rho(M^*) - \rho(M_{k(t)})) \right] = \mathbb{E} \left[\sum_{k=1}^{k(T)} \mathbb{E} \left[\sum_{h=1}^{L_k} (\rho(M^*) - \rho(M_h) \mid \mathcal{H}_{k1}) \right] \right] = 0. \quad (4.25)$$

However, this result is incorrect since it ignores that the length of the episode k is a random variable which is not in general $\sigma(\mathcal{H}_{k1})$ -measurable and so the results of Lemma 4.1 do not

apply. In fact, we can clearly see for this example with the choice of $T = 1000$ that,

$$\mathbb{E} \left[\sum_{t=1}^{1000} (\rho(M^*) - \rho(M_{k(t)})) \right] = \frac{1}{4} \times 1000 - \frac{1}{4} \times 1 > 0. \quad (4.26)$$

This completes the counter-example to the proof technique of Theorem 2 in [2]. \square

The discussion of how to dynamically sample new episodes in a communicating, but not necessarily recurrent, MDP with infinite horizon is an interesting open question. However, we believe that the finite horizon case provides a much simpler setting in which to build intuition. For the most part, this will also be sufficient for the practical settings of interest [23]. In the course of Theorem 4.2 we have established the first $\tilde{O}(\sqrt{SAT})$ bounds for regret in the finite horizon setting. However, there remain some questions about the scaling in the horizon H . In the next subsection we address some of the issues regarding the interaction of learning in finite horizon MDPs.

4.4.3 Learning in finite horizon MDPs

In Section 2.1 we introduced finite horizon setting that allowed both the transition dynamics and reward structure to vary between timesteps. However, many problems of interest are time-homogeneous and, in these settings, we would like to our algorithm to exploit this structure. Theorem 4.2 attains the optimal scaling in S, A and T however there remains a gap to the lower bounds in terms of H . As we outlined in Section 2.3, it is not even clear what the true optimal scaling in H should be. Previous works had claimed $\Omega(H\sqrt{SAT})$ for time homogeneous MDPs, but this analysis was not rigorous [10]. The best established lower bounds are $\Omega(\sqrt{HSAT})$ for the homogenous setting and $\Omega(H\sqrt{SAT})$ in the time-inhomogeneous setting. In this section we try to clarify some of these issues.

Modifying Theorem 4.2 for time-homogeneous MDPs

PSRL as an *algorithm* can naturally work with time-homogenous MDPs. The analysis in Theorem 4.1 gives regret bounds for time inhomogeneous MDPs of $\tilde{O}(H^{3/2}S\sqrt{AT})$ which improve to $\tilde{O}(HS\sqrt{AT})$ for time homogeneous MDPs, just like UCRL. However, theorem 4.2 for time-inhomogeneous MDPs gives bounds $\tilde{O}(H^{3/2}\sqrt{SAT})$ for PSRL and we cannot

naively improve the H -dependence in the same way for time homogeneous MDPs in the same way. This is a technical issue that rests upon the shortcoming of our analysis, rather than any underlying issue of the algorithm.

The problem is that we have only established Lemma 3.3 for fixed future value V . In PSRL for time-homogeneous systems the sample at any $P_k(s, a, h)$ may impact the future value as well as the deviation from \hat{P} . Interestingly, this issue does not arise in the analysis of Gaussian PSRL, so that we are able to establish $\tilde{O}(H\sqrt{SAT})$ bounds on the expected regret for Gaussian PSRL in the time homogeneous setting. However, despite these mismatches in the bounds we actually believe that PSRL is the better algorithm and that it will be able to improve the regret bounds for PSRL to $\tilde{O}(\sqrt{HSAT})$. This is in contrast to the previously stated lower bounds of $\Omega(H\sqrt{SAT})$ [10].

Hypothesis for an $\tilde{O}(\sqrt{HSAT})$ regret bound

Previous works had claimed a result which, if true, would establish a lower bound $\Omega(H\sqrt{SAT})$ on regret in finite horizon time-homogenous MDPs [10]. In this setting, our proof of Theorem 4.2 would translate to expected regret bounds for Gaussian PSRL which match this scaling. Even though we have not successfully matched these results for PSRL, we believe that PSRL is a better algorithm since its samples are more accurately adjusted to the posterior distribution for MDPs. In fact, we believe that the true regret scaling for PSRL is $\tilde{O}(\sqrt{HSAT})$, but for Gaussian PSRL correctly $\tilde{O}(H\sqrt{SAT})$. We present some intuition for this hypothesis below.

Recent analysis of UCRL in finite horizon settings has established a sample complexity bound $\tilde{O}(H^2S^2A\epsilon^{-2})$ in the number of episodes until an ϵ -suboptimal policy [23]. These bounds are optimal in their H -dependence, whereas previous analysis for UCRL had an extra factor of H . Importantly, by the arguments in Theorem 4.1, these bounds also apply to PSRL in expectation without requiring any change to the underlying algorithm. The key step in this analysis is to note that, during a single episode, we cannot repeatedly make a bad decision that gives us $O(H)$ regret, since the total value within any episode is bounded by H . To exploit this structure they make use of a Bellman relation, which mirrors the arguments in [?] for discounted MDPs. Although this analytical trick would apply to PSRL, and thus lead to $\tilde{O}(\sqrt{HSAT})$ regret bounds it would not apply to Gaussian PSRL. We are

working hard to establish exactly this result in future work.

4.5 Experimental evaluation

In this section we will evaluate several learning algorithms on a family of environments designed to highlight the need for directed and deep exploration. The example is designed as a scalable version of *RiverSwim* [78, 61] and similar to Example 2.1 with stochastic rewards. The environment is a chain of $S = N$ states with deterministic transitions “left” and stochastic transitions “right”. All states give zero reward except for s_1 which gives $r_t \sim N(0, 1)$ and s_N which gives $r_t \sim N(1, 1)$. The agent begins each episode in state s_1 with deterministic reset every $H = N$ steps. The optimal policy is to choose the action “right” in every timestep, and receive an expected value of $\left(1 - \frac{1}{N}\right)^{N-1} \rightarrow \exp(-1)$, all other policies have zero expected value. We depict this environment in Figure 4.3.

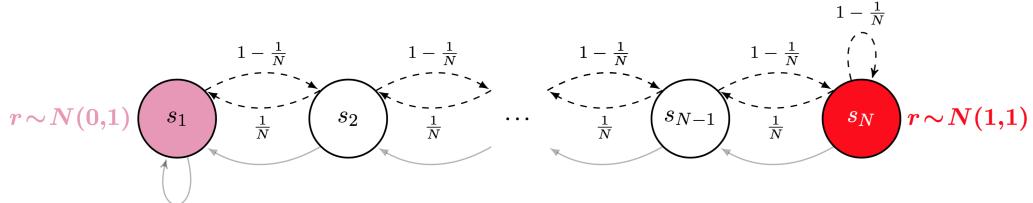
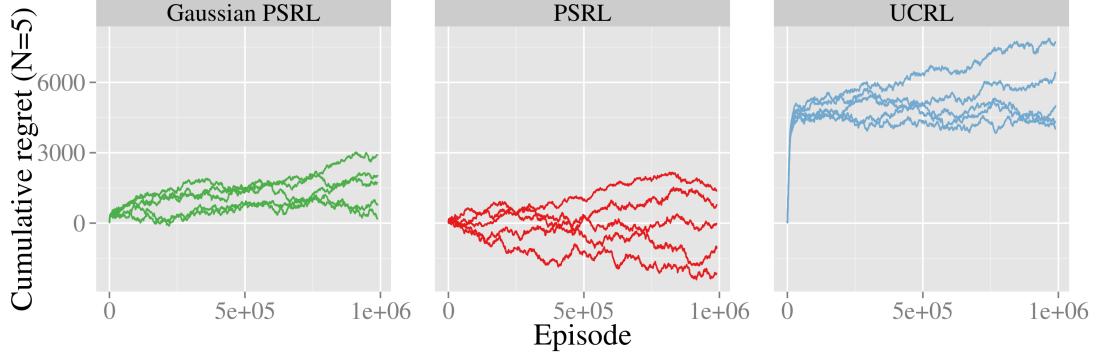
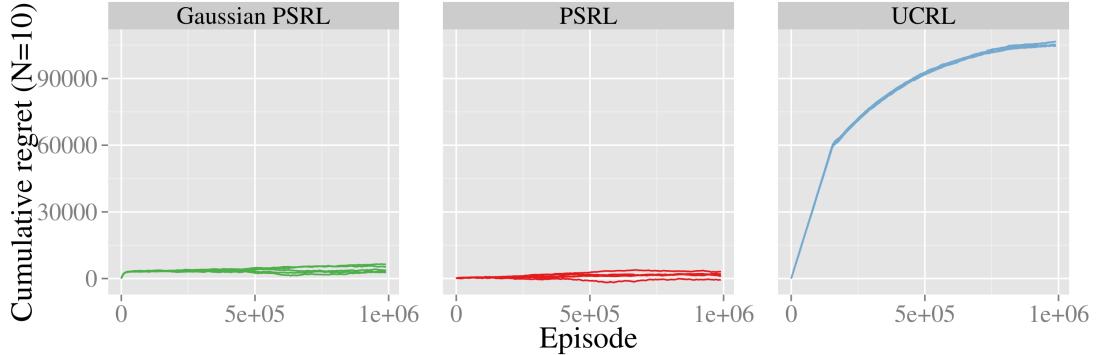


Figure 4.3: Our environments for scalable and deep exploration

4.5.1 Performance

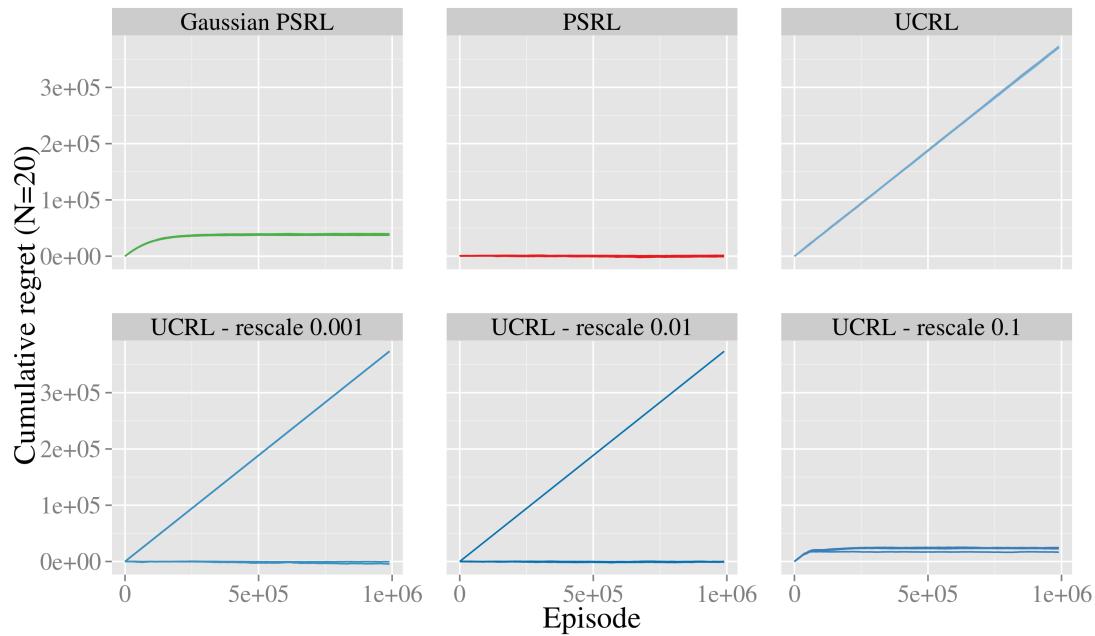
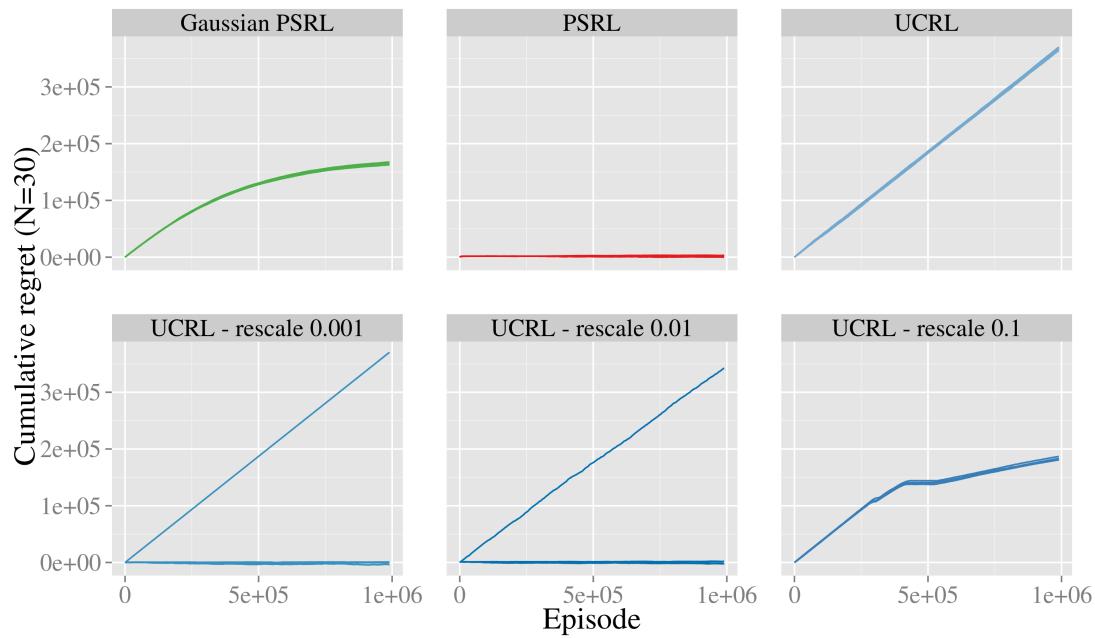
We implement three algorithms; UCRL2, PSRL and Gaussian PSRL. For UCRL2 we use the confidence parameter $\delta = 0.05$ and confidence sets from Algorithm 3.2. For PSRL we use the Jeffrey’s prior for transitions $\alpha^P = \frac{1}{2}\mathbb{1}$ and a flat uninformative prior for mean rewards $\bar{r}^* \sim N(0, 1)$. For Gaussian PSRL we use the values that correspond to this same prior for PSRL from Algorithm 4.3.

Figure 4.4 displays the cumulative regret for these algorithms on a chain with $N = S = H = 5$. We can see that, in this environment the performance of these algorithms is ordered with PSRL better than Gaussian PSRL which is in turn better than UCRL2. As we increase the size of the chain to $N = 10$ this difference becomes even more pronounced in Figure 4.5.

Figure 4.4: Regret in a stochastic chain of length $N = 5$ Figure 4.5: Regret in a stochastic chain of length $N = 10$

For larger domains UCRL2 is orders of magnitude slower than either Gaussian PSRL or PSRL. One practical approach to improving performance in UCB-style methods is to “tune” the confidence sets to give better performance [71]. In this method we rescale the confidence sets from Algorithm 3.2 by some constant multiple. Although the algorithm may no longer satisfy analytical regret bounds, in some situations this may lead to improved performance. For the larger MDPs we include this rescaled version of UCRL2 in our consideration.

We can see that rescaling to give more aggressive confidence sets can lead to improved performance. However, this is at the cost of some simulations which completely fail to learn, this is because the algorithm is now too exploitative and may fail to explore its environment adequately. For the large chain $N = 50$ we can see that PSRL learns orders of magnitude faster than competing methods in Figure 4.8. We highlight the performance of PSRL alone in Figure 4.9, this is similar to UCRL2 on a chain of length $N = 5$.

Figure 4.6: Regret in a stochastic chain of length $N = 20$ Figure 4.7: Regret in a stochastic chain of length $N = 30$

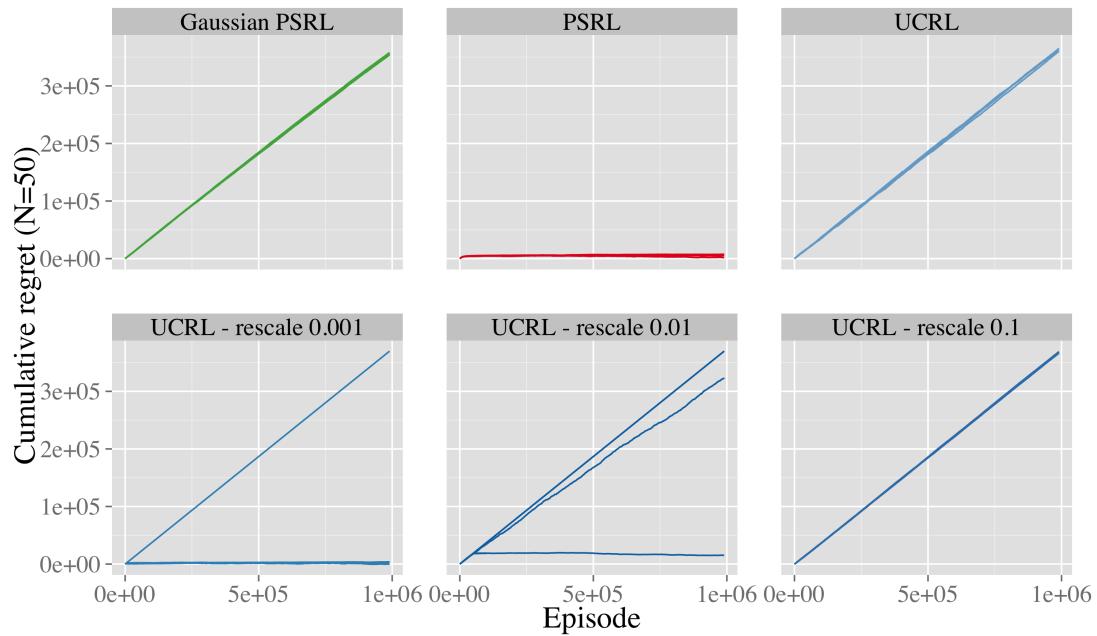
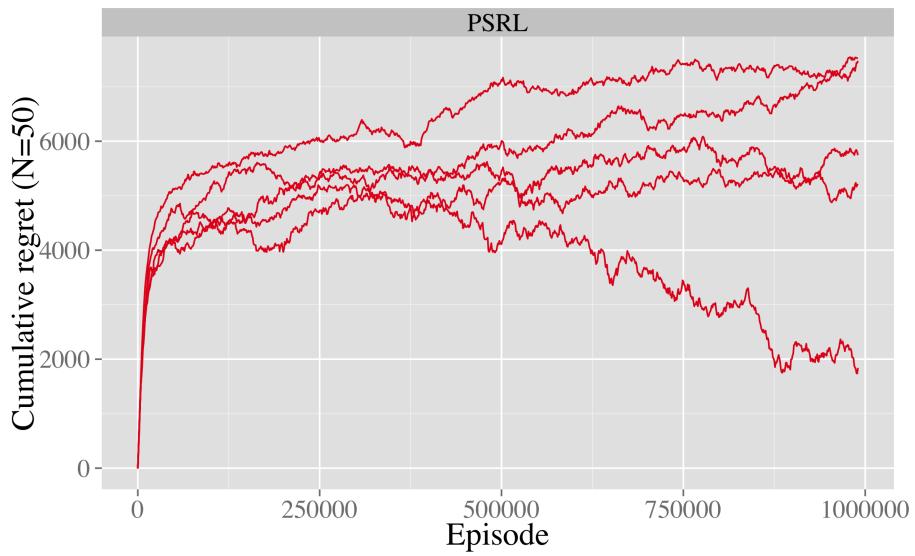
Figure 4.8: Regret in a stochastic chain of length $N = 50$ 

Figure 4.9: Our environments for scalable and deep exploration

4.5.2 Scaling

We now present some indicative plots to examine the empirical scaling of each algorithm with the growth of the chain N . To do this we plot the time each algorithm takes to reach an average regret of less than 0.1 against the length of the chain N on a logarithmic scale. In this setting, a linear relationship of slope k indicates that the time to learn scales with N^k . We now use proposition 2.1 to relate the regret bounds for each algorithm to scalings in learning time. UCRL2 has regret bounds $\tilde{O}(HS\sqrt{AT})$, which should scale like N^4 . Gaussian PSRL has regret bounds $\tilde{O}(H\sqrt{SAT})$, which should scale like N^3 . We have not completed the proof for PSRL, but in this chapter we present arguments that it should scale $\tilde{O}(\sqrt{HSAT})$, which should scale like N^2 . In Figure 4.10 we present the learning times for each algorithm together with indicative scalings of these slopes. We can see that our data fits this hypothesis extremely well.

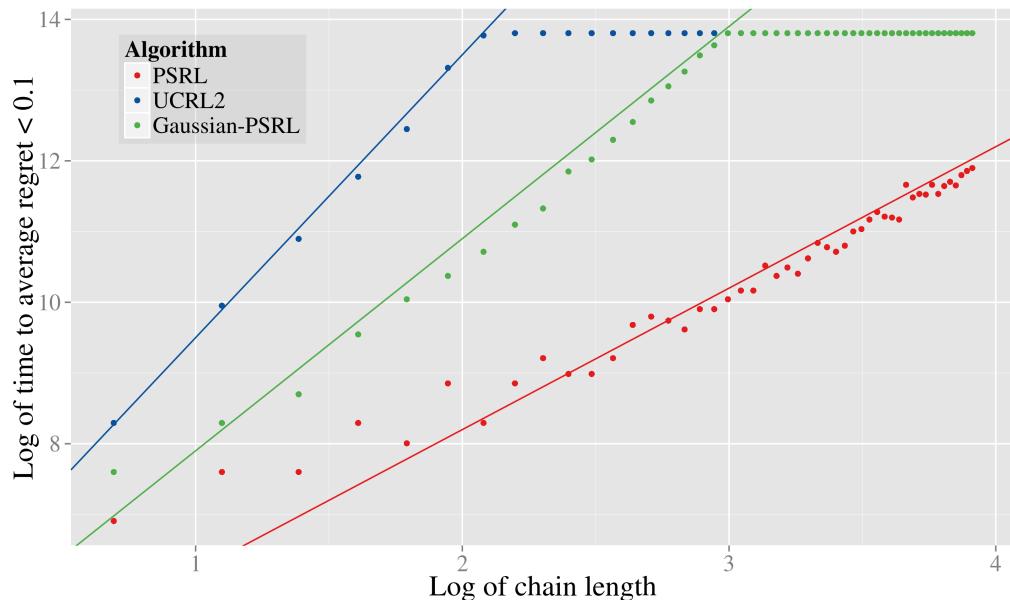


Figure 4.10: Our environments for scalable and deep exploration

We repeat this exercise for the rescaled versions of UCRL2 in Figure 4.11. In this figure we plot all five seeds and discount those simulations which fail to converge to the optimal policy. We see that, although the rescaled versions of UCRL2 can produce better performance this rescaling does not alter the fundamental scaling of the algorithm. We note that these rescaled variants also do not even learn the optimal policy. By contrast, every

single seed of PSRL learned the optimal policy in every single chain of length $N = 5, \dots, 50$ and learned it extremely quickly.

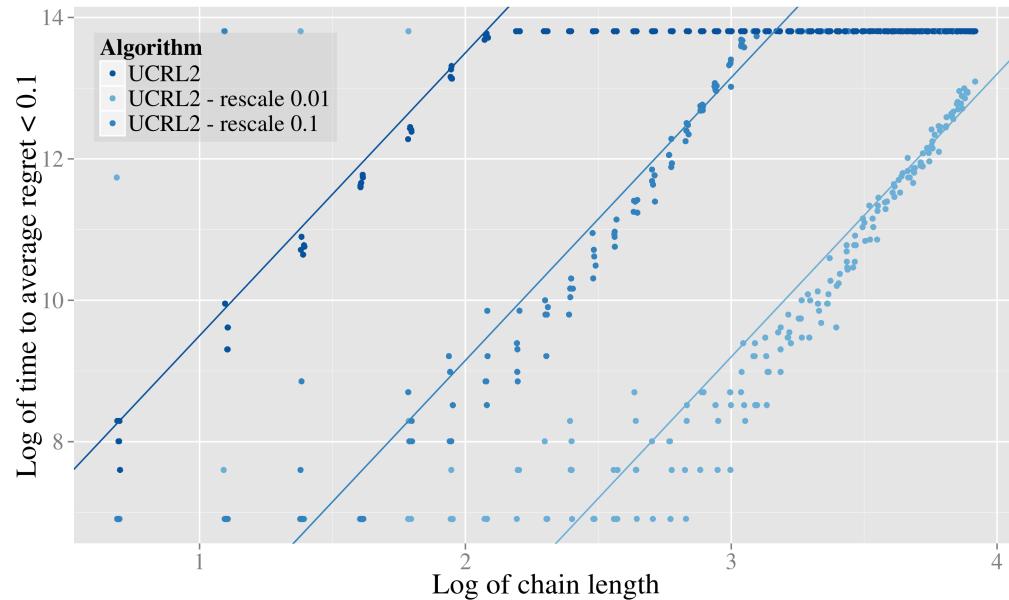


Figure 4.11: Our environments for scalable and deep exploration

Part II

Model-based Generalization

Part I 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. 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 thesis 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 [17]. The essential material for this chapter is taken from our previous work [63]. 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, \dots, 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 $s_h \in \{1, \dots, J\}^L$. Similarly the choice of action at every timestep $a_h \in \{1, \dots, K\}^L$. Therefore, even for a reasonably small systems¹ 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-dimesnional structure.

¹Consider, 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} .

We will imagine that, in any single timestep, the evolution of any single machine only depends directly upon its *local* interactions with its neighbours. To be more concrete we imagine that, over one timestep, 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. This sort of conditional independence is exactly what is captured by a factored MDP. We present an illustration of this example in Figure 5.1. Note that in this example a malfunction in any machine can still affect any other machine in the production line, just not over a single timestep.

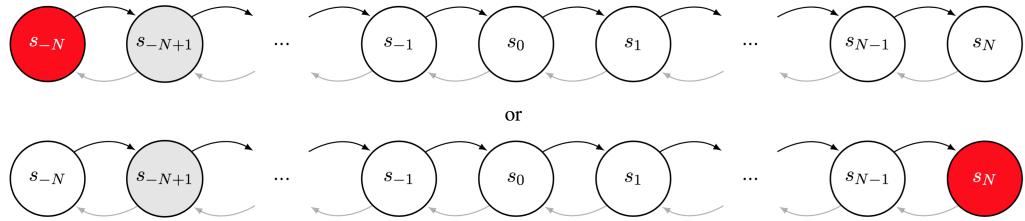


Figure 5.1: Factored representation for production line. CHANGE PICTURE.

Factored MDPs are good models for large systems with several weakly-connected components. More generally they are any Markov decision process whose reward and transition can be represented as a dynamic Bayesian network (DBN). Examples, examples, examples, examples, examples, examples, examples, examples, examples, examples. Examples, examples, examples, examples, examples, examples, examples, examples, examples, examples. Examples, examples, examples, examples, examples, examples, examples, examples, examples, examples.

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 \mathcal{X} is a shorthand for $\mathcal{S} \times \mathcal{A}$.

Definition 5.1 (Scope operation for factored sets $\mathcal{X} = \mathcal{X}_1 \times \dots \times \mathcal{X}_n$).

For any subset of indices $Z \subseteq \{1, 2, \dots, n\}$ 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 $\mathcal{P}_{\mathcal{X}, \mathbb{R}}^{C, \sigma}$ to be the set of distributions mapping elements of \mathcal{X} to distributions over $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$ such that, for any $R \in \mathcal{P}_{\mathcal{X}, \mathbb{R}}^{C, \sigma}$ and $x \in \mathcal{X}$ we can write

$$R(x) =_D \bar{r}(x) + W \quad (5.1)$$

where $\bar{r}(x) \in [0, C]$ 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 $R \in \mathcal{R} \subseteq \mathcal{P}_{\mathcal{X}, \mathbb{R}}^{C, \sigma}$).

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

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

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 $\mathcal{S} \times \mathcal{A} = \mathcal{X} = \mathcal{X}_1 \times \dots \times \mathcal{X}_n$ and $\mathcal{S} = \mathcal{S}_1 \times \dots \times \mathcal{S}_m$ with scopes Z_1, \dots, 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,

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

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 $Z_i^R, Z_i^P \subseteq \{1, \dots, n\}$ 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:

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

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 $\tilde{O} (J^L \sqrt{K^L T})$, Theorem 4.2 can improve upon this scaling to give $\tilde{O} (LJ \sqrt{KT})$.

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 [63]. 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 PSRL with probability at least $1 - \delta$,

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

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. For a full specification of UCRL-Factored we refer the reader to the paper [63].

5.2.1 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. 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.

5.2.2 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.

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,

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

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 \bar{r}_i^* individually, rather than the total $\bar{r}^* = \sum_{i=1}^l \bar{r}_i^*$ 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, 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 $\mathcal{X} = \mathcal{X}_1 \times \dots \times \mathcal{X}_n$ and $\mathcal{S} = \mathcal{S}_1 \times \dots \times \mathcal{S}_m$ with scopes Z_1, \dots, 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:

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

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 $\{1, \dots, d_1\}$ and q, \tilde{q} over $\{1, \dots, d_2\}$. We let $Q = pq^T, \tilde{Q} = \tilde{p}\tilde{q}^T$ be the joint probability distribution over $\{1, \dots, d_1\} \times \{1, \dots, d_2\}$. 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

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

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 than the overall reward or transition function. 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 $\Omega(L\sqrt{SAT})$.

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 [62]. 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 [70, 71], to capture the complexity of the reinforcement learning problem. 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 with a $\|\cdot\|_2$ -norm induced by an inner product. This allows us to decompose the transition function as a mean value in \mathcal{S} plus additive noise $s' \sim P^M(\cdot|s, a) \implies s' = \bar{p}^M(s, a) + \epsilon_P$.

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 $\mathcal{P}_{\mathcal{X}, \mathcal{Y}}^{C, \sigma}$ to be the set of distributions mapping elements of \mathcal{X} to distributions over $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$ such that, for any $P \in \mathcal{P}_{\mathcal{X}, \mathcal{Y}}^{C, \sigma}$ and $x \in \mathcal{X}$ we can write:

$$P(x) =_D \bar{p}(x) + W \quad (6.1)$$

where $\|\bar{p}(x)\|_2 \in [0, C]$ 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 this broad class of functions. 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^* . In order to do this it will be helpful for us to 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 ,

$$U_i^M(\Phi) := \mathbb{E}_{M, \mu^M} [V_{\mu^M, i+1}^M(s) | s \sim \Phi]. \quad (6.2)$$

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,

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

We define

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

to be a global Lipschitz constant for the future value function with state distributions from \mathcal{D} . Where appropriate, we will 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 than other works have assumed [2, 59]. 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 [10].

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,

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

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 ,

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

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,

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

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. Our main result in this chapter, Theorem 6.1, bounds the expected regret of PSRL at any time T .

Theorem 6.1 (Regret bounds for PSRL in parameterized MDPs).

Fix a state space \mathcal{S} , action space \mathcal{A} , function families $\mathcal{R} \subseteq \mathcal{P}_{\mathcal{S} \times \mathcal{A}, \mathbb{R}}^{C_{\mathcal{R}}, \sigma_{\mathcal{R}}}$ and $\mathcal{P} \subseteq \mathcal{P}_{\mathcal{S} \times \mathcal{A}, \mathcal{S}}^{C_{\mathcal{P}}, \sigma_{\mathcal{P}}}$ 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,*

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

This result is established in more detail in the paper [62] as the result of a 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:*

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

Here λ_1 is the largest eigenvalue of the matrix Q given as the solution of the Riccati equations for the unconstrained optimal value function $V(s) = -s^T Q s$ [11].

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 [62]. \square

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. 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 5.2.1 applies equally well to this chapter as well.

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 [8, 20, 71]. 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 [?], but with constants that grow exponentially with dimension. Later works remove this exponential dependence, but only under significant sparsity assumptions [41]. Following our analysis in [62], later works attempted to extend these results to linear quadratic systems without episodic regret [2]. However, as we outline in Example 4.4 their analysis is not rigorous and the proof of their theorems may not hold be correct. The most general previous analysis considers rewards and transitions that are α -Hölder in a d -dimensional space to establish regret bounds $\tilde{O}(T^{(2d+\alpha)/(2d+2\alpha)})$ [59]. 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 [71, 70] to vector-valued functions. We call this the eluder dimension because we imagine a game where one agent attempts to learn the function $f : \mathcal{X} \rightarrow \mathcal{Y}$, 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 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 [71].

Definition 6.7 $((\mathcal{F}, \epsilon) - \text{dependence})$.

We will say that $x \in \mathcal{X}$ is (\mathcal{F}, ϵ) -dependent on $\{x_1, \dots, x_n\} \subseteq \mathcal{X}$

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

$x \in \mathcal{X}$ is (ϵ, \mathcal{F}) -independent of $\{x_1, \dots, x_n\}$ if and only 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}, ϵ') -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 $\mathcal{G} \subseteq \mathcal{P}_{\mathcal{X}, \mathcal{Y}}^{C, \sigma}$ 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}, ϵ . However, for any given \mathcal{F}, ϵ 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 [?]. 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 $\mathcal{F} = \{f \mid f(x) = \theta\phi(x) \text{ for } \theta \in \mathbb{R}^{n \times p}, \phi \in \mathbb{R}^p, \|\theta\|_2 \leq C_\theta, \|\phi\|_2 \leq C_\phi\}$ then $\forall \mathcal{X}$:

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

Proposition 6.3 (Eluder dimension for quadratic functions).

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

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

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 $\in [\underline{h}, \bar{h}]$ with $\underline{h} > 0$. Define $r = \frac{\bar{h}}{\underline{h}} > 1$ 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} :

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

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 [62] 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,

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

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

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

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 $\mathcal{F}_t = \mathcal{F}_t(\beta_t) := \{f \in \mathcal{F} \mid \|f - \hat{f}_t^{LS}\|_{2,E_t} \leq \sqrt{\beta_t}\}$ 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:

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

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 [71], but extends statements about \mathbb{R} to vector-valued functions. A full derivation is available in the Section 6.5. \square

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 L_2 deviation between any two members of \mathcal{F} evaluated at x .

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

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 [71] but with a small adjustment to account for episodes. A full proof is given in Section 6.5. \square

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:

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

Proof. Once again we follow the analysis of Russo [71] and strealine 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 $(w_1, \dots, w_T) \rightarrow (w_{i_1}, \dots, w_{i_T})$ such that $w_{i_1} \geq \dots \geq w_{i_T}$ 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:

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

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

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

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,

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

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 [71] 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 $\psi_i(\lambda) = \log \mathbb{E}[\exp(\lambda(Z_i - \mu_i)) | \mathcal{H}_{i-1}]$.

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

Both of these lemmas are available in earlier discussion for real-valued variables [?]. We specialize our discussion to the vector space $\mathcal{Y} \subseteq \mathbb{R}^d$ where the inner product $\langle y, y \rangle = \|y\|_2^2$. 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

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

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:

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

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}^α so that $\forall f \in \mathcal{F}$:

$$\begin{aligned}
L_{2,t}(f) - L_{2,t}(f^*) &\geq \frac{1}{2}\|f - f^*\|_{2,E_t} - 4\sigma^2 \log(N(\mathcal{F}, \alpha, \|\cdot\|_2)/\delta) + DE(\alpha) \quad (6.15) \\
\text{For } DE(\alpha) &= \min_{f^\alpha \in \mathcal{F}^\alpha} \left\{ \frac{1}{2}\|f^\alpha - f^*\|_{2,E_t}^2 - \frac{1}{2}\|f - f^*\|_{2,E_t}^2 + L_{2,t}(f) - L_{2,t}(f^\alpha) \right\}
\end{aligned}$$

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

$$\begin{aligned} \|f^\alpha(x) - f^*(x)\|_2^2 - \|f(x) - f^*(x)\|_2^2 &= \|f^\alpha(x)\|_2^2 - \|f(x)\|_2^2 + 2 < f^*(x), f(x) - f^\alpha(x) > \leq 4C\alpha \\ \|y - f(x)\|_2^2 - \|y - f^\alpha(x)\|_2^2 &= 2 < y, f^\alpha(x) - f(x) > + \|f(x)\|_2^2 - \|f^\alpha(x)\|_2^2 \leq 2\alpha|y| + 2C\alpha + \alpha^2 \end{aligned}$$

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

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, \dots, x_{t-1} . If x_t is ϵ -dependent on K disjoint subsequences then there exist $\|\bar{f} - \underline{f}\|_{2,E_t} > K\epsilon^2$. By the triangle inequality $\|\bar{f} - \underline{f}\|_{2,E_t} \leq 2\sqrt{\beta_t} \leq 2\sqrt{\beta_T}$ 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 [71]. 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. \square

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, \dots, x_d such that for some $\epsilon' \geq \epsilon$:

$$w_k = \sup \left\{ \|(\bar{f} - \underline{f})(x_k)\|_2 \mid \|\bar{f} - \underline{f}\|_{2, E_t} \leq \epsilon' \right\} > \epsilon' \quad (6.16)$$

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

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

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:

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

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

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

Here we solve for the dual-optimal $\lambda^* \nabla_{\lambda}g = 0 \implies \frac{1}{2\lambda^*} \text{Sum}\left(\frac{1}{2\lambda}\tilde{\Phi}V^{-1} \otimes \tilde{\Phi}\right) - \frac{1}{4\lambda^*} \text{Trace}\left(\frac{1}{4\lambda}\tilde{\Phi}V^{-1}\tilde{\Phi}^T\right) = \epsilon^2$. From the definition of $\tilde{\Phi}$, $\text{Sum}(\tilde{\Phi}V^{-1} \otimes \tilde{\Phi}) = n\phi^TV^{-1}\phi$ and $\text{Trace}(\tilde{\Phi}V^{-1}\tilde{\Phi}^T) = \phi^TV^{-1}\phi$.

From this we can simplify our expression to conclude:

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

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

Using this lemma, we will be able to address 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 $\|\phi_k\|_{V_k^{-1}}^2 > \frac{1}{4n-2} > 0$.

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:

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

for $\lambda := \left(\frac{\epsilon}{2C_\theta}\right)^2$. To get an upper bound on the determinant we note that $\det(V_k)$ is maximized when all eigenvalues are equal or equivalently:

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

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 express 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 our 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 $\tilde{\epsilon} = \frac{\epsilon}{pC_\theta}$. 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 $\in [\underline{h}, \bar{h}]$ with $\underline{h} > 0$. Define $r = \frac{\bar{h}}{\underline{h}} > 1$ 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} :

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

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

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

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.

6.5.2 Bounded LQR control

We imagine a standard linear quadratic controller with rewards with $x = (s, a)$ the state-action vector. The rewards and transitions are given by:

$$R(x) = -x^T A x + \epsilon_R, \quad P(x) = \Pi_C(Bx + \epsilon_P),$$

where $A \succcurlyeq 0$ is positive semi-definite and Π_C projects x onto the $\|\cdot\|_2$ -ball at radius C .

In the case of unbounded states and actions the Riccati equations give the form of the optimal value function $V(s) = -s^T Q s$ for $Q \succcurlyeq 0$. In this case we can see that the difference in values of two states:

$$|V(s) - V(s')| = | -s^T Q s + s'^T Q s'| = | - (s + s')^T Q (s - s')| \leq 2C\lambda_1 \|s - s'\|_2$$

where λ_1 is the largest eigenvalue of Q and C is an upper bound on the $\|\cdot\|_2$ -norm of both s and s' . We note that $2C\lambda_1$ works as an effective Lipschitz constant when we know what C can bound s, s' .

We observe that for any projection $\Pi_C(x) = \alpha x$ for $\alpha \in (0, 1]$ and that for all positive semi-definite matrices $A \succcurlyeq 0$, $(\alpha x)^T A (\alpha x) = \alpha^2 x^T A x \leq x^T A x$. Using this observation together with reward and transition functions we can see that the value function of the bounded LQR system is always greater than or equal to that of the unconstrained value function. The effect of excluding the low-reward outer region, but maintaining the higher-reward inner region means that the value function becomes more flat in the bounded case, and so $2C\lambda_1$ works as an effective Lipschitz constant for this problem too.

Part III

Value-based Generalization

In Part II we demonstrated 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, for truly large scale systems even the problem of solving a single known MDP may itself be intractable. In these environments an algorithm for efficient RL should be able to exploit low dimensional structure not just in the underlying environment, but also its solution to the optimal policy.

In this part of the thesis, we explore an approach to generalization and exploration in large scale MDPs that synthesize the learning and planning steps of randomized value functions. PSRL maintains a posterior distribution over the underlying environment, which induces a posterior distribution over policies through solving for the optimal policy given any sample. We will seek to obviate the need for this planning step by directly maintaining a distribution over the value function, without an explicit posterior for the underlying environment and without the need for a separate MDP planning algorithm.

Once again, we present randomized value functions as an effective approach to deep exploration. However, unlike PSRL, the randomized value function will not be drawn from the exact posterior distribution for the value function. Instead the algorithms in this part will be drawn from an *approximate* posterior distribution which captures some aspects of the agent's uncertainty, but should be computationally tractable even in large systems. 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 experiments which suggest that randomized value functions can drive efficient exploration in high dimensional and complex environments.

Chapter 7

Linear Value Functions

We now turn our attention to 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* [80]. In these settings, one common approach is to approximate the value of a policy by some shared functional representation. For now, we will focus upon the linear setting where the agent approximates the value function as the linear combination of some given basis functions. These basis functions can be chosen to encode some 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 [65]. The core idea of RLSVI is very simple; since building up an exact posterior over value functions via PSRL is intractable, we approximate the posterior for the value function by a Gaussian. 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 such as TD, LSTD, and SARSA (see, e.g., [80, 82]). 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 the problem of 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 a linear value function to model that, for each h the optimal Q-value function,

$$Q_h^* \in \text{span} [\Phi_h] \text{ for some } \Phi_h \in \mathbb{R}^{SA \times D}. \quad (7.1)$$

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 $d = 1, 2, \dots, D$, we write the d th column of Φ_h as ϕ_{hd} and refer to ϕ_{hd} as a basis function¹. We refer to contexts where the agent's belief is correct as *coherent learning*, and refer the alternative as *agnostic learning*.

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., [80, 82]). 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.

¹We note that we had previously used D as the diameter of the MDP in Definition 4.2. For the remainder of this chapter we will use D only as the dimension for the linear basis functions.

Algorithm 7.1 Least-Squares Value Iteration

Input: Data $\Phi_1(s_{k1}, a_{k1}), r_{k1}, \dots, \Phi_H(s_{kH}, a_{kH}), r_{kH} : k < K$, Parameters $\lambda > 0, \sigma > 0$

Output: $\tilde{\theta}_{k1}, \dots, \tilde{\theta}_{kH}$

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

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

- 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 $\Phi_1, \dots, \Phi_H; \sigma > 0, \lambda > 0, \epsilon > 0$

- 1: **for** episode $k = 1, 2, \dots$ **do**
 - 2: Compute $\tilde{\theta}_{k1}, \dots, \tilde{\theta}_{kH}$ using LSVI
 - 3: **for** $h = 1, \dots, H$ **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: $a_{kh} \in \arg \max_{\alpha \in \mathcal{A}} (\Phi_h \tilde{\theta}_{kh})(s_{kh}, \alpha)$
 - 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 $\Phi_1, \dots, \Phi_H; \sigma > 0, \lambda > 0, \eta > 0$

- 1: **for** episode $k = 1, 2, \dots$ **do**
 - 2: Compute $\tilde{\theta}_{k1}, \dots, \tilde{\theta}_{kH}$ using LSVI
 - 3: **for** $h = 1, \dots, H$ **do**
 - 4: Sample $a_{kh} \sim \exp \left[(\Phi_h \tilde{\theta}_{kh})(s_{kh}, \alpha) \right]^\infty$
 - 5: Observe r_{kh} and $s_{k,h+1}$
 - 6: **end for**
 - 7: **end for**
-

These algorithms are natural adaptations of Algorithms 2.2 and 2.3 to linearly parameterized value functions. Although 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, we can adapt Example 2.1 to *any* choice of basis function. Both ϵ -greedy and Boltzmann exploration will take $\Omega(2^S)$ timesteps to learn the optimal policy, regardless of the choice of basis function [65]. In order to guarantee efficient

learning we need to have directed (and deep) exploration.

7.1.2 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 [61]. However, exact MDP planning does not scale well to large environments. RLSVI sidesteps the intractable planning step by instead building up an approximate posterior distribution in the value function directly. To do this, RLSVI simply performs a linear regression on the Bellman error as if the one step Bellman error were Gaussian. This admits a simple closed form posterior update via conjugate Gaussian prior.

Algorithm 7.4 Randomized Least-Squares Value Iteration

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

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

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

- 3: Bayesian linear regression for the value function

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

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

Algorithm 7.5 RLSVI with greedy action

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

- 1: **for** episode $k = 1, 2, \dots$ **do**
 - 2: Compute $\tilde{\theta}_{k1}, \dots, \tilde{\theta}_{kH}$ using Algorithm 7.4
 - 3: **for** $h = 1, \dots, H$ **do**
 - 4: Sample $a_{kh} \in \arg \max_{\alpha \in \mathcal{A}} (\Phi_h \tilde{\theta}_{kh})(s_{kh}, \alpha)$
 - 5: Observe r_{kh} and $s_{k,h+1}$
 - 6: **end for**
 - 7: **end for**
-

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*, as demonstrated in Example 2.1, regardless of the choice of basis function. In this section we will establish a bound on the expected regret for RLSVI in a tabular setting without generalization where the basis functions $\Phi_h = I$.

For our analysis to hold, we will assume that rewards $R(s, a, h)$ are drawn from independent Dirichlet $\alpha^R(s, a, h) \in \mathbb{R}_+^2$ with values on $\{-1, 0\}$ and transitions Dirichlet $\alpha^P(s, a, h) \in \mathbb{R}_+^S$. Analytical techniques exist to extend similar results to general bounded distributions; see, for example [3].

Theorem 7.1 (RLSVI satisfies regret bounds in the tabular setting).

If RLSVI is executed with $\Phi_h = I$ for $h = 1, \dots, H$ with $\lambda \geq \max_{(s,a,h)} (\mathbf{1}^T \alpha^R(s, a, h) + \mathbf{1}^T \alpha^P(s, a, h))$ and $\sigma \geq \sqrt{H^2 + 1}$, then,

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

The result of Theorem 7.1 should not be entirely surprising, since for $\Phi_h = I$ this algorithm looks almost identical to Gaussian PSRL (Algorithm 4.3). We introduced Gaussian PSRL as a simple Gaussian approximation to PSRL in Chapter 4. The only difference for RLSVI with tabular basis functions from Gaussian PSRL is the global estimate shrinkage λ which brings the value estimate towards zero. However, for our rescaling rewards in $[-1, 0]$ this shrinkage is *optimistic* and so will not affect our analysis via Lemma 3.3 beyond the cost of an additive bias term.

7.2.1 Analysis

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 ,

$$\Delta_k = \underbrace{V_{k,1}^k - V_{k,1}^*}_{\Delta_k^{\text{conc}}} + \underbrace{V_{*,1}^* - V_{k,1}^k}_{\Delta_k^{\text{opt}}} .$$

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 $\lambda \geq \max_{(s,a,h)} (\mathbf{1}^T \alpha^R(s, a, h) + \mathbf{1}^T \alpha^P(s, a, h))$ are stochastically optimistic,

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

Proof. We present a simple sketch of this argument that relates RLSVI to Gaussian PSRL, for which we have already established a similar optimism result. For a more detailed proof of this Lemma see [65].

For both Dirichlet and Gaussian distributions, the posterior mean is a weighted average of the empirical mean. For $\lambda \geq \max_{(s,a,h)} (\mathbf{1}^T \alpha^R(s, a, h) + \mathbf{1}^T \alpha^P(s, a, h))$ we know that the relative weight of the ‘‘prior’’ shrinkage in RLSVI is larger than in the correctly specified prior. Since the rewards are bounded $\in [0, 1]$ we know that the prior shrinkage in RLSVI is more optimistic than for the correct posterior and Gaussian PSRL. This means that, for any dataset, the mean estimates of RLSVI will be above Gaussian PSRL.

Now, with the choice of $\sigma \geq \sqrt{H^2 + 1}$ the magnitude of the noise from RLSVI will be at least as spread out as Gaussian PSRL. The proof of Gaussian PSRL comes from Theorem 4.2 and relies upon the stochastic optimism relation between Gaussian and Dirichlet distributions in Lemma 3.3. \square

Lemma 7.1 means RLSVI generates stochastically optimistic Q-values for any history \mathcal{H}_{k1} . All that remains is to prove the remaining estimates $\mathbb{E}[\Delta_l^{\text{conc}} | \mathcal{H}_{k1}]$ 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 $\Delta_l^{\text{conc}} = V_{k1}^k(s_{k1}) - V_{k0}^*(s_{k1})$. We decompose the value estimate V_{k1}^k explicitly:

$$\begin{aligned} V_{kh}^k(s_{kh}) &= \frac{n(\hat{R} + \hat{P}^T V_{k,h+1}^k h + 1)}{n + \lambda\sigma^2} + w_{kh}^\sigma \\ &= \bar{R}_k + \bar{P}_k^T V_{k,h+1}^k + b_{kh}^R + b_{kh}^P + w_{kh}^\sigma \end{aligned} \quad (7.4)$$

where w_{kh}^σ is the Gaussian noise from RLSVI and $b_{kh}^R = b^R(s_{kh}, a_{kh}, h)$, $b_{kh}^P = b^P(s_{kh}, a_{kh}, h)$ 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,

$$\mathbb{E}[V_{k1}^k(s_{k1}) - V_{k1}^*(s_{k1}) | \mathcal{H}_{k1}] = \sum_{h=1}^H \left\{ b^R(s_{kh}, a_{kh}, h) + b^P(s_{kh}, a_{kh}, h) + w_{kh}^\sigma \right\}. \quad (7.5)$$

We can even write explicit expressions for b^R , b^P and w_h^σ .

$$\begin{aligned} b^R(s, a, h) &= \frac{n\hat{R}}{n + \lambda\sigma^2} - \frac{n\hat{R} - \alpha_1^R}{n + \mathbf{1}^T \alpha^R} \\ b^P(s, a, h) &= \frac{n\hat{P}^T \tilde{V}_{h+1}^k}{n + \lambda\sigma^2} - \frac{(n\hat{P} + \alpha^P)^T \tilde{V}_{h+1}^k}{n + \mathbf{1}^T \alpha^P} \\ w_{kh}^\sigma &\sim N\left(0, \frac{\sigma^2}{n + \lambda\sigma^2}\right) \end{aligned}$$

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 let $\lambda = \mathbf{1}^T \alpha^R + \mathbf{1}^T \alpha^P$ and $\sigma = \sqrt{H^2 + 1}$. Up to \tilde{O} notation $b^R \simeq \frac{\alpha_1^R}{n + \mathbf{1}^T \alpha^P}$, $b^P \simeq \frac{H\mathbf{1}^T \alpha^P}{n + \mathbf{1}^T \alpha^P}$ and $w_h^\sigma \simeq \frac{H}{\sqrt{n + H^2 \mathbf{1}^T \alpha^R + \mathbf{1}^T \alpha^P}}$. Summing using a pigeonhole principle for $\sum_{s,a,h} n(s, a, h) = T$ gives us an upper bound on the regret. We write $K(s, a, h) := (\alpha_1^R(s, a, h) + H\mathbf{1}^T \alpha^P(s, a, h))$ to bound the effects of the prior mismatch in RLSVI arising from the bias terms b^R, b^P . The constraint $\alpha^T \mathbf{1} \geq 2$ can only be violated twice for each s, a, h . Therefore up to $O(\cdot)$ notation:

$$\begin{aligned} \mathbb{E} \left[\sum_{l=0}^{T/H-1} \mathbb{E}[\Delta_l^{\text{conc}} | \mathcal{H}_{k1}] \right] &\leq 2SAH + \\ \sum_{s,a,h} K(s, a, h) \log(T + K(s, a, h)) + H\sqrt{SAHT \log(T)} \end{aligned}$$

This completes the proof of Theorem 7.1.

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

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

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 modelled on Example 2.1, where dithering strategies for exploration are provably inefficient. Importantly, and unlike the tabular experiments in Section 4.5, 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 $(\mathbf{1}, Q_h^*, \tilde{w}_1, \dots, \tilde{w}_{D-2})$. 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 $(\mathbf{1}, w_1, \dots, w_{k-1})$ onto V_{hD} and renormalize each component to have equal 2-norm. For more details on the practicalities of these experiments see [65]. Figure 8.4 presents the empirical regret for RLSVI with $D = 10, N = 50, \sigma = 0.1, \lambda = 1$ and an ϵ -greedy agent over 5 seeds².

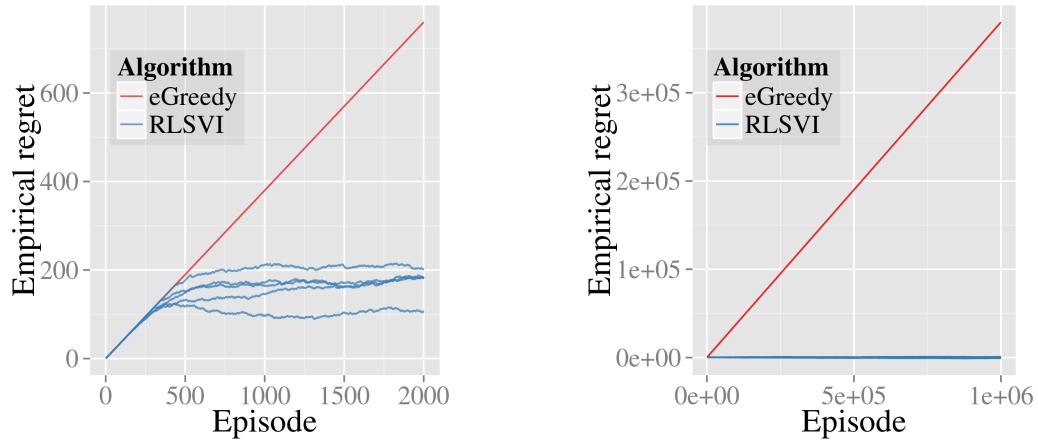


Figure 7.1: Efficient exploration on Example 2.1 with a length 50 chain.

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 [23] 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.

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

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 [23]. For $N = 100$, $2^{N-1} > 10^{28}$ and $H^2SA > 10^6$. RLSVI demonstrates scalable generalization and exploration to outperform these bounds.

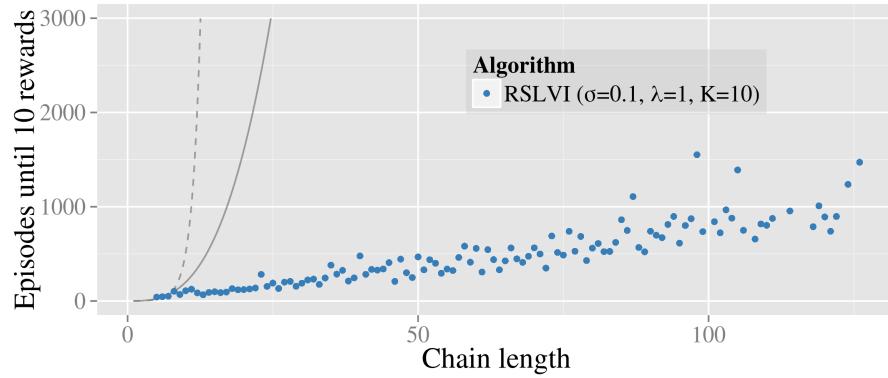


Figure 7.2: RLSVI learning time against chain length.

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 bounds³.

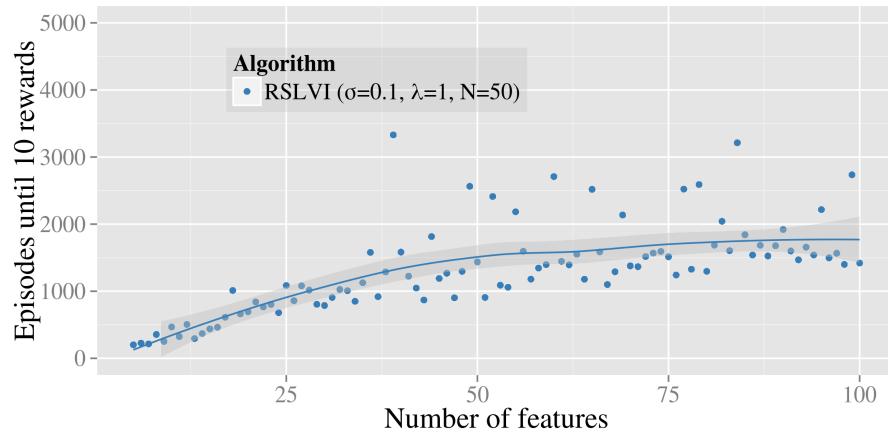


Figure 7.3: RLSVI learning time against number of basis features.

Figure 7.4 examines these same scalings on a logarithmic scale. We find the data for

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

these experiments is consistent with polynomial learning as hypothesized in Appendix ???. These results are remarkably robust over several orders of magnitude in both σ and λ . We present more detailed analysis of these sensitivities in Appendix ??.

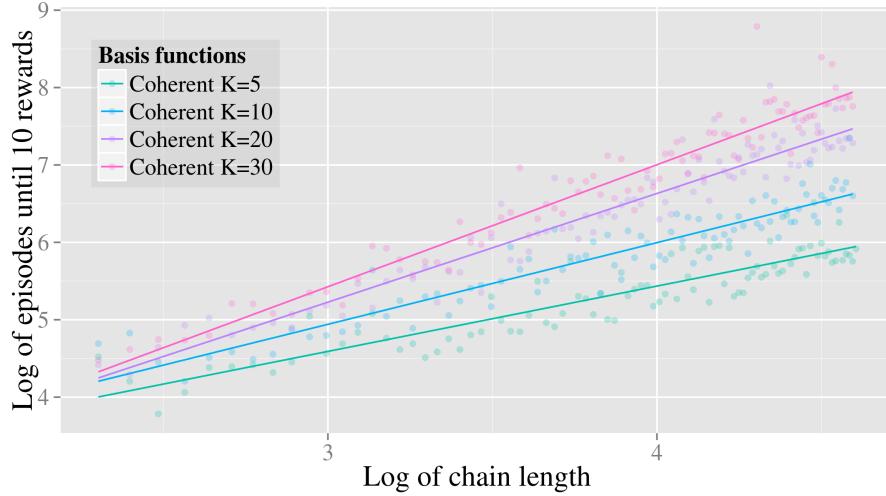


Figure 7.4: Empirical support for polynomial learning in RLSVI.

7.3.2 Agnostic learning

Unlike the example above, 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 $i = 0, \dots, 1000$ 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 stable⁴.

This simple example gives us some hope that RLSVI can be useful in the agnostic setting. In our remaining experiments we will demonstrate that RLSVI can achieve state of the art results in more practical problems with agnostic features.

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

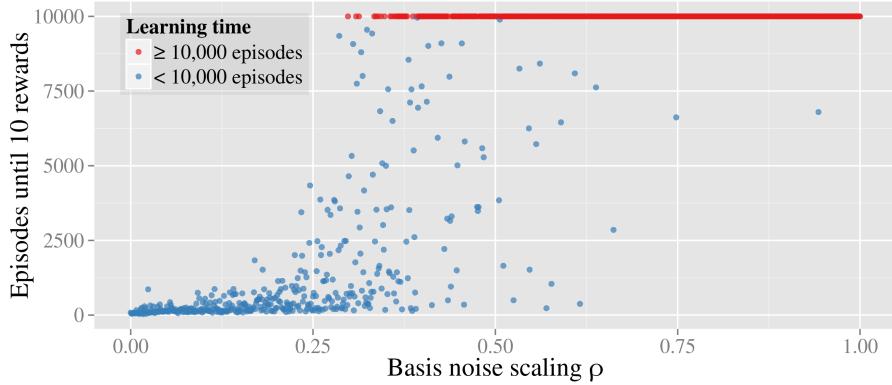


Figure 7.5: RLSVI is somewhat robust model misspecification.

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 [?]. 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 with more than 2^{200} states. In order to tackle this problem efficiently we use 22 benchmark features. These features 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, but we use these to mirror their approach.

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 keep the most recent $N = 10^5$ transitions to cap the linear growth in memory and computational requirements, similar to [57]. In Figure 7.6 we present learning curves for RLSVI $\lambda = 1, \sigma = 1$ and LSVI with a tuned ϵ -greedy exploration

schedule⁵

First, both RLSVI and LSVI make significant improvements over the previous approach of LSPI with the same basis functions [12]. 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.

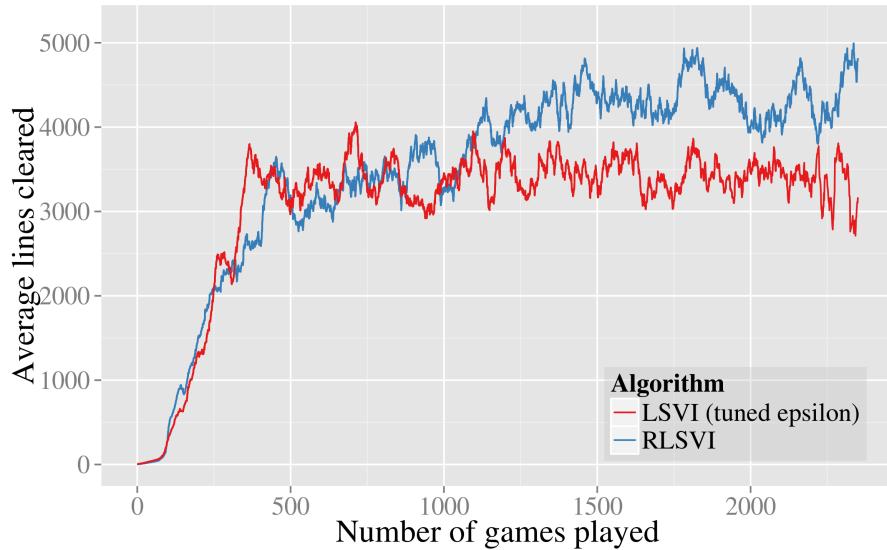


Figure 7.6: Learning to play Tetris with linear Bertsekas features.

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 [65].

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

⁵We 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. The results are significant in several ways.

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.

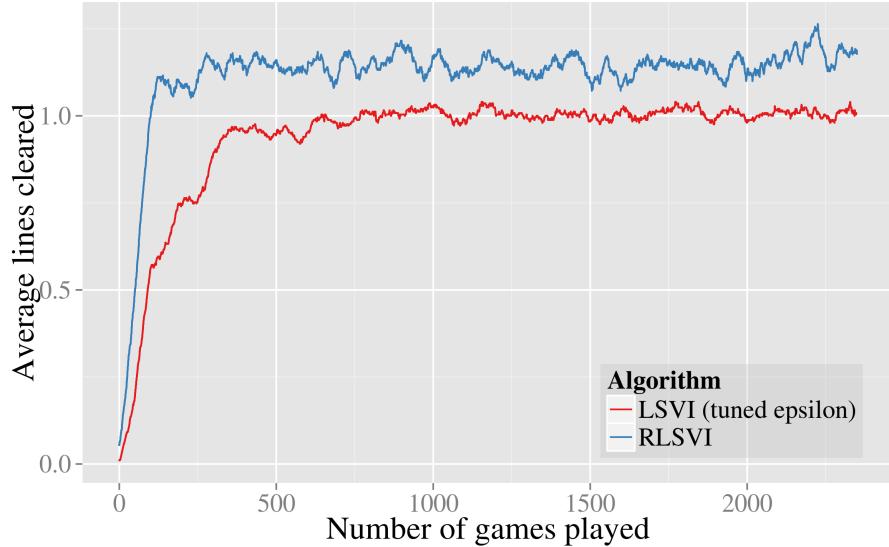


Figure 7.7: Reduced 4-row tetris with only S and Z pieces.

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 $\mathcal{Z} = \{1, 2, \dots, N\}$ 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 :

$$\mathbb{P}(a|x) = 1 / (1 + \exp(-[\beta_a + \sum_n \gamma_{an}x_n])). \quad (7.6)$$

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 .

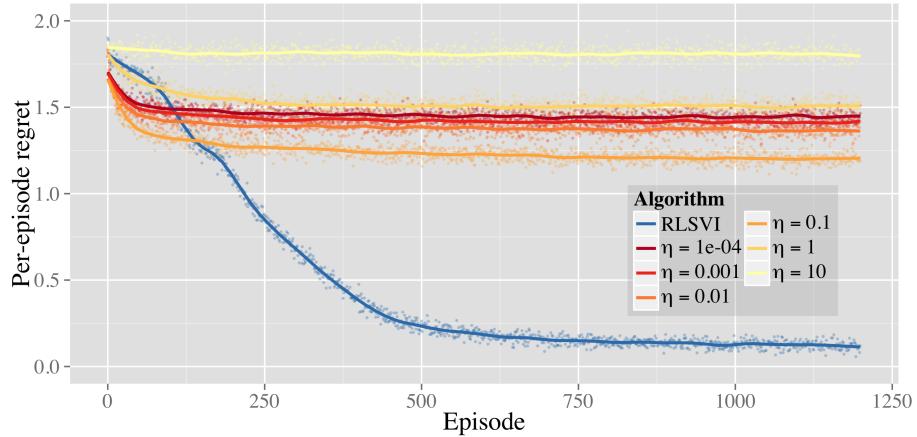


Figure 7.8: RLSVI performs better than Boltzmann exploration.

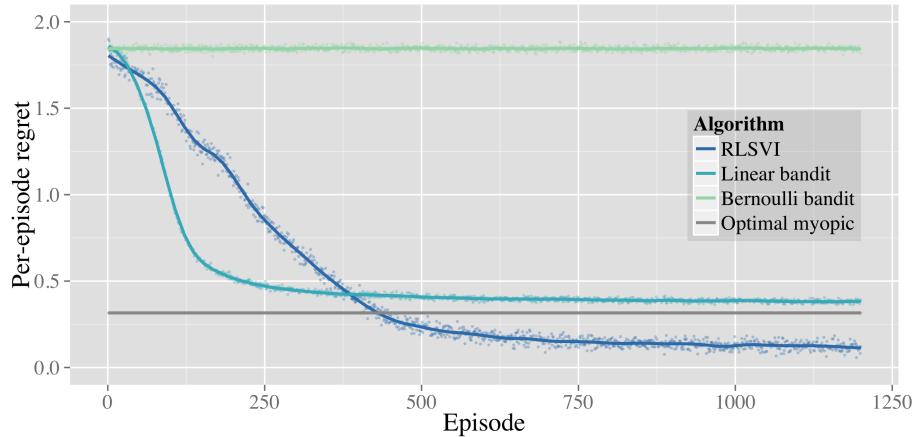


Figure 7.9: RLSVI can outperform the optimal myopic policy.

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 (7.6). 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.

In Figure 7.9 we show that, using this efficient exploration method, the reinforcement learning policy is able to outperform not only benchmark bandit algorithms but even the optimal myopic policy⁶. 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.

Appendix ?? provides pseudocode for this computational study. 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.

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 an optimal myopic system, particularly where large amounts of data are available. In some settings, efficient generalization and exploration can be crucial.

⁶The optimal myopic policy knows the true model defined in Equation 7.6, 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. Randomized least squares value iteration can combine efficient exploration with generalization over linear basis functions. The performance of this algorithm is crucially dependent on the choice of good basis functions. However, in settings with rich unstructured data it can be difficult to design effective linear features for these basis functions. In these environments nonlinear function approximation and “deep learning” have emerged as the state of the art methods for learning both the parameter values *and* the feature representation simultaneously.

In this chapter, we demonstrate a simple and tractable way to perform deep exploration via randomized value functions with nonlinear generalization. We present 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 a general extension of RLSVI to arbitrary nonlinear function approximators. We focus upon bootstrapped *deep Q networks* (DQN), which combines “deep exploration” with “deep learning”. The essential material for this chapter is taken from our papers [64] and [60]. Bootstrapped DQN can demonstrate deep exploration in complex stochastic MDPs to learn exponentially faster than DQN with dithering. Bootstrapped DQN is scalable to large problems such as the Arcade Learning Environment; it

substantially improves learning times and performance across most Atari games.

8.1 Bayes by bootstrap

The term *bootstrap* refers to a class of methods for nonparametric estimation from data-driven simulation [27]. In essence, the bootstrap uses the empirical distribution of a sampled dataset as an estimate of the population statistic [28]. 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 $\hat{\psi}_1, \dots, \hat{\psi}_K$ to approximate the distribution of ψ by a discrete distribution $\hat{P} = \sum_{k=1}^K \mathbb{1}\{\hat{\psi}_k\} = y$. 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 $x_1, \dots, x_N \in \mathcal{X}$, estimator $\psi : \mathcal{P}(\mathcal{X}) \rightarrow \mathcal{Y}$

Output: Bootstrap sample $\hat{\psi}$

- 1: Sample data $\tilde{D} = (\tilde{x}_1, \dots, \tilde{x}_N)$ from $D = (x_1, \dots, x_N)$ 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 $\hat{\psi} = \psi(\hat{P})$
-

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 [14]. 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 [49].

The output \hat{P} is reminiscent of a Bayesian posterior with an extremely weak prior. 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[67]. This can be interpreted as the

posterior of a Dirichlet process in the limit of no prior information¹.

Algorithm 8.2 A Bayesian bootstrap sample

Input: Data $x_1, \dots, x_N \in \mathcal{X}$, estimator $\psi : \mathcal{P}(\mathcal{X}) \rightarrow \mathcal{Y}$

Output: Bootstrap sample $\hat{\psi}$

- 1: Sample $w_1, \dots, w_N \sim \text{Exp}(1)$
 - 2: For all $dx \subset \mathcal{X}$ let $\hat{P}(dx) = \sum_{n=1}^N w_n \mathbf{1}\{x_n \in dx\} / \sum_{n=1}^N w_n$
 - 3: Estimate $\hat{\psi} = \psi(\hat{D})$
-

With the bootstrap approaches we have described, the support of distributions \hat{P}_k is restricted to the dataset $\{x_1, \dots, x_N\}$. In sequential decision problems this can pose a significant problem, since this may underestimate the posterior variance and lead to suboptimal exploitation [64]. To address the problem, we propose a simple extension to the bootstrap. In particular, we augment the dataset $\{x_1, \dots, x_N\}$ with artificially generated samples $\{x_{N+1}, \dots, x_{N+M}\}$ 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 [64]. In other settings, such as deep neural networks, it may actually be much more practical to specify a prior belief in terms of artificial prior data rather than complicated prior distributions.

8.1.1 Uncertainty in deep neural networks

Deep neural networks represent the state of the art in many supervised learning domains [?]. 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 [34, 15], assumed density filtering [40], dropout-based variational inference [31, 48], and stochastic gradient Langevin dynamics

¹To highlight the similarities between these methods we note that, as $N \rightarrow \infty$ the bootstrap approximates data weighted by $w_1, \dots, w_N \sim \text{Po}(1)$.

[83]. Instead of appealing to parametric assumptions about the uncertainty present in RL problems, we use the non-parametric bootstrap [27] to obtain a distribution over functions represented by the neural network.

In Figure 8.1 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 [66]. 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 [?].

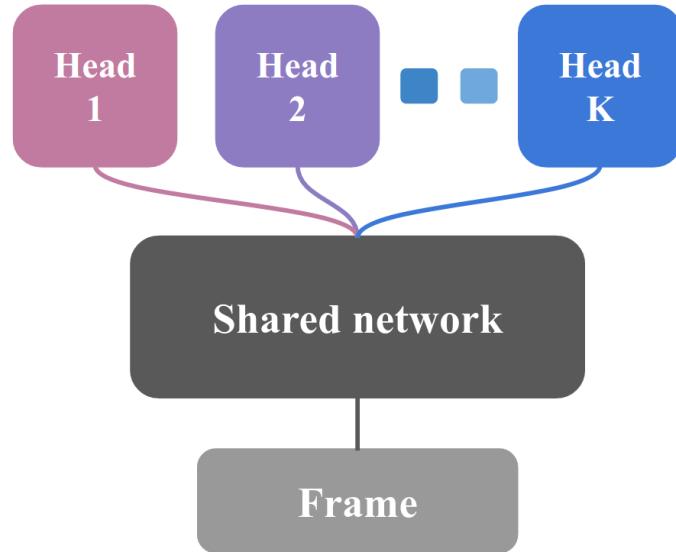


Figure 8.1: An efficient architecture for K bootstrap samples

Figure 8.2 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 [31]. Further details are provided in [60].

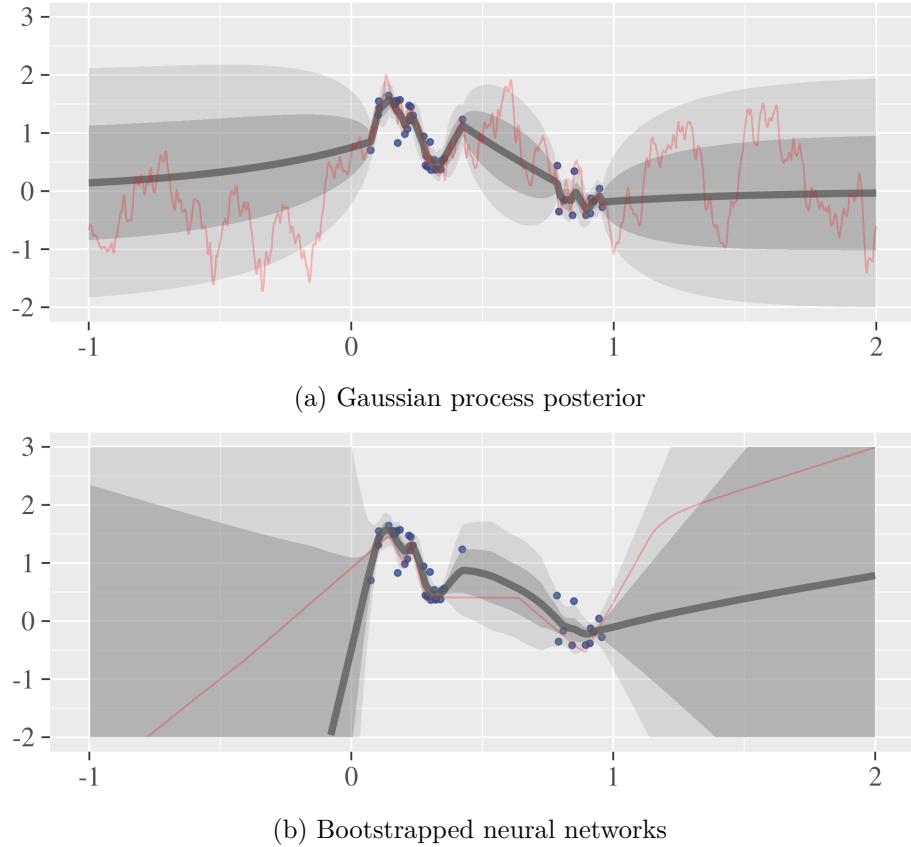


Figure 8.2: Grey regions depict the mean estimate $\pm \{1, 2\}$ standard deviations. A single sample is shown in red.

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 [65, 64, 60].

Algorithm 0 is general, since it is compatible with any Q value approximator ψ and any

Algorithm 8.3 Bootstrapped Q learning

Input: Bootstrap method B , Q value approximator ψ

- 1: **for** episode $k = 1, 2, \dots$ **do**
 - 2: Sample bootstrap estimate $\tilde{Q}_k \sim B(\psi, \mathcal{H}_{k1})$
 - 3: **for** $h = 1, \dots, H$ **do**
 - 4: Sample $a_{kh} \in \arg \max_{\alpha \in \mathcal{A}} \tilde{Q}_k(s_{kh}, \alpha)$
 - 5: Observe r_{kh} and $s_{k,h+1}$
 - 6: **end for**
 - 7: **end for**
-

bootstrap method for approximate posterior inference [64]. 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 [57]. 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

$$\theta_{t+1} \leftarrow \theta_t + \alpha(y_t^Q - Q(s_t, a_t; \theta_t)) \nabla_{\theta} Q(s_t, a_t; \theta_t) \quad (8.1)$$

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 [57]. 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 :

$$y_t^Q \leftarrow r_t + \gamma \max_a Q(s_{t+1}, \arg \max_a Q(s_{t+1}, a; \theta_t); \theta^-).$$

DDQN can demonstrate improved performance and stability [?]. In this paper we use the DDQN update for all DQN variants unless explicitly stated.

Our algorithm, *bootstrapped DQN* in 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 temporally extended (or deep) exploration [79, 61].

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 8.1. 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, \dots, Q_K provide a temporally extended (and consistent) estimate of the value uncertainty via TD estimates. We approximate a bootstrap sample by selecting $k \in \{1, \dots, K\}$ 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 resample every timestep².

Algorithm 8.4 Bootstrapped DQN

- 1: **Input:** neural network $(Q_k)_{k=1}^K$, sampling distribution P
 - 2: **for** each episode **do**
 - 3: Update network parameters via minibatches
 - 4: Sample $k \sim \text{Uniform}\{1, \dots, K\}$
 - 5: **while** not end of episode **do**
 - 6: Choose $a_t \in \arg \max_a Q_k(s_t, a)$
 - 7: Receive state s_{t+1} and reward r_t from environment
 - 8: Sample bootstrap mask $m_t^k \sim P$ for all k
 - 9: Add (a_t, r_t, s_{t+1}, m_t) to replay buffer
 - 10: **end while**
 - 11: **end for**
-

²We call bootstrapped DQN which resamples a head every timestep Thompson DQN. This is similar to the “Thompson sampling” algorithm of [31] but uses the bootstrap in place of dropout as an approximate posterior.

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* [78]. 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 $\phi : \{1, \dots, N\} \rightarrow \{0, 1\}^N$. We consider two feature mappings, “one-hot” $\phi_{\text{oh}}(s_t) := (\mathbf{1}\{x = s_t\})_{x=1, \dots, N}$ and “thermometer” $\phi_{\text{therm}}(s_t) := (\mathbf{1}\{x \leq s_t\})_{x=1, \dots, N}$. 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 [60].

8.3.1 Scaling deeper

We consider a family of deterministic chains of length $N > 3$ as in Figure 8.3. 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 [65].

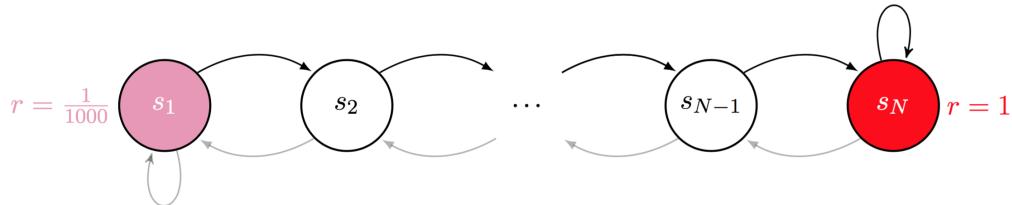


Figure 8.3: Scalable environments that requires deep exploration.

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 8.4, together with a conservative lower bound of $99 + 2^{N-11}$ on the expected time to learn for any shallow exploration strategy. Only bootstrapped DQN demonstrates a graceful scaling to long chains which require deep exploration.

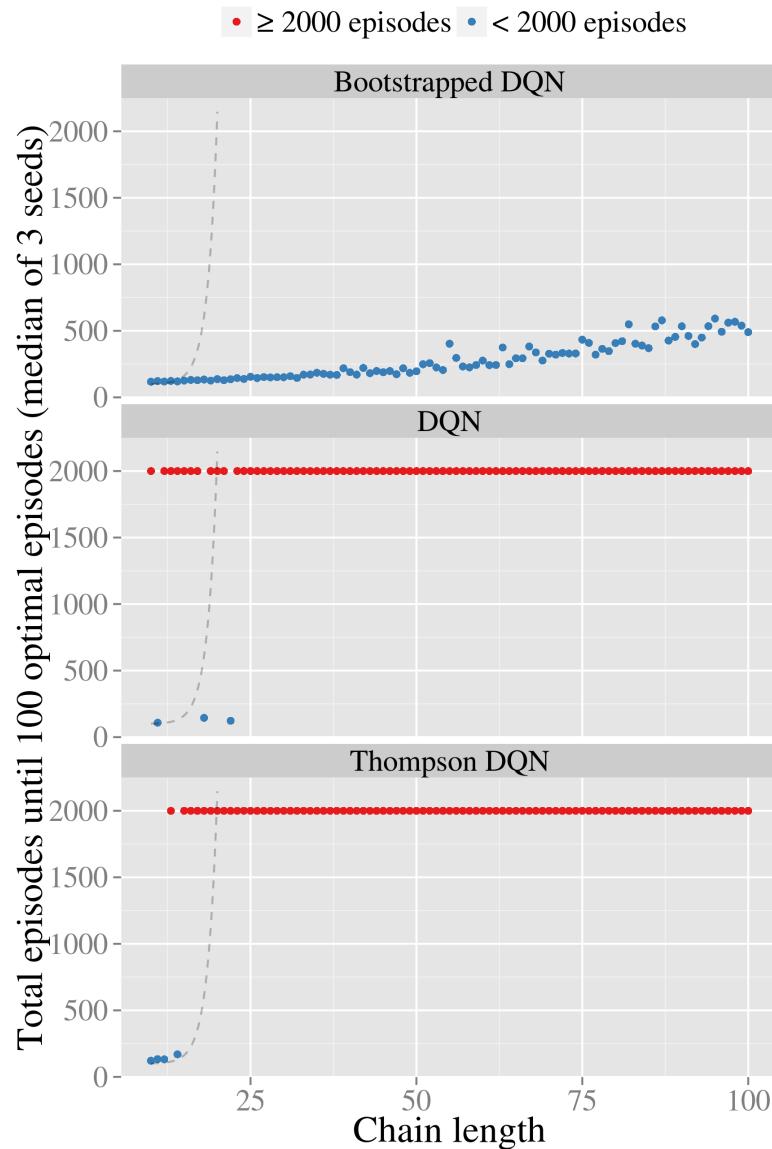


Figure 8.4: Bootstrapped DQN demonstrates deep exploration.

8.3.2 A difficult stochastic MDP

Figure 8.4 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 8.5. 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.

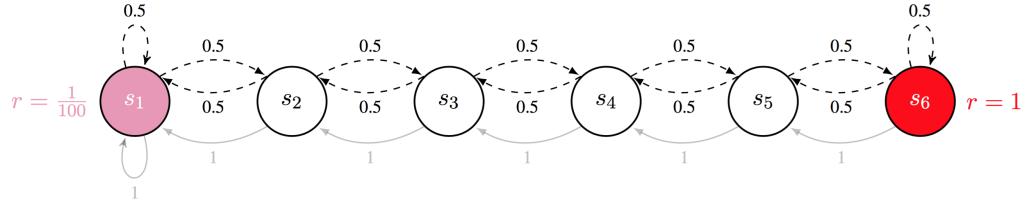


Figure 8.5: A stochastic MDP that requires deep exploration.

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 [42]) and posterior sampling (PSRL [61]).

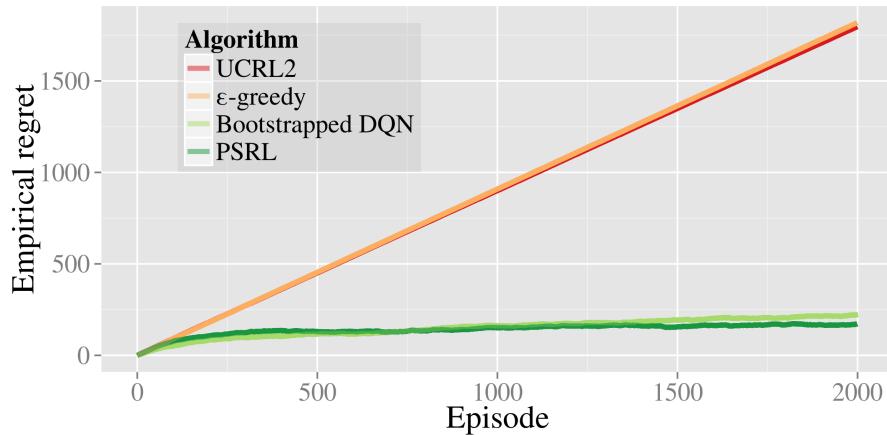


Figure 8.6: Bootstrapped DQN matches efficient tabular RL.

In Figure 8.6 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 8.6 seems to suggest UCRL2 incurs linear regret, actually it follows its bounds $\tilde{O}(S\sqrt{AT})$ [42] where S is the number of states and A is the number of actions. We demonstrate this through simulation [60].

8.4 Experiments on Atari 2600

We now evaluate our algorithm across 49 Atari games on the Arcade Learning Environment [?]. Importantly, and unlike the experiments in Section ??, 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 strategies³. 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 [?] for our network architecture and benchmark our performance against their algorithm. Our network structure is identical to the convolutional structure of DQN [57] except we split 10 separate bootstrap heads after the convolutional layer as per Figure 8.1. Recently, several authors have provided architectural and algorithmic improvements to DDQN [?, ?]. 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 Appendix ??.

³By contrast, imagine the agent received a small reward for dying; dithering strategies would be hopeless, just like Section ??.

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 use a shared convolutional network architecture with K bootstrap heads as per Figure 8.1. This leaves 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? Each of these questions involves balancing computational and statistical considerations. We found that even a small number of bootstrap heads was sufficient to incentivize efficient exploration. Figure 8.7 presents the cumulative reward of bootstrapped DQN on the game Breakout, for different number of heads K . More heads leads to faster learning, but even a small number of heads captures most of the benefits of bootstrapped DQN, we choose $K = 10$. Due to the shared convolutional network, we find this overall network retains similar compute speed to DQN.

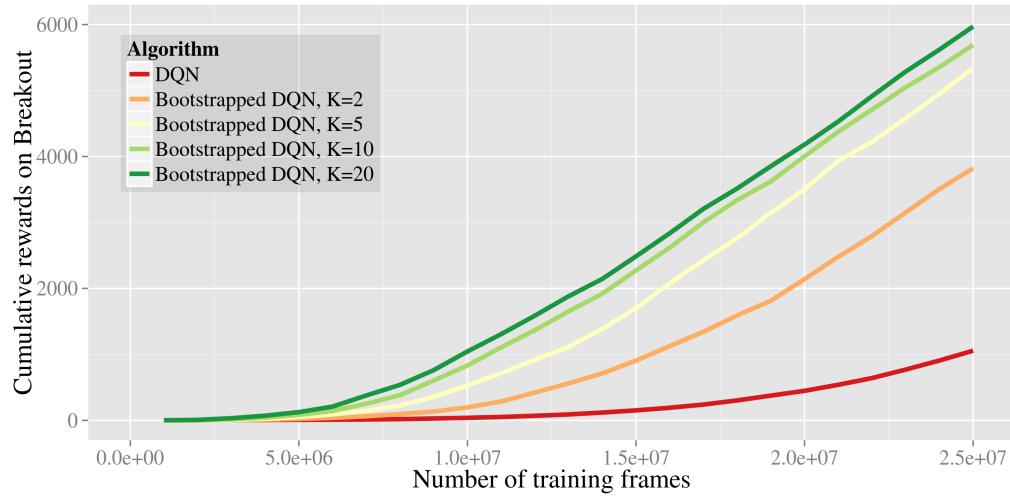


Figure 8.7: Decreasing marginal benefit to extra bootstrap heads.

This shared network architecture allows us to train this combined network via back-propagation. One question is whether to normalize gradients flowing from the network heads. 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. Since much of the literature focusses best policy after 200m frames, rather than cumulative rewards during learning, we choose

1/10 rescaling for our shared gradients⁴.

Finally we have the question of how to implement an online bootstrap for the K network heads. In order to avoid a significant increase in memory footprint we use an independent Bernoulli mask $w_1, \dots, w_K \sim \text{Ber}(p)$ for each head in each episode⁵. These flags are stored in the memory replay buffer and identify which heads are trained on which data. A smaller p means that fewer data is shared, which will maintain a higher diversity of bootstrap samples.

However, when trained using a shared minibatch the algorithm will also require an effective $1/p$ more iterations. This is undesirable since DQN is already computationally taxing. Surprisingly, we found the algorithm performed similarly irrespective of p and all outperformed DQN, as shown in Figure 8.8. This is strange, but we present a more detailed investigation of this phenomenon in Appendix ???. 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 hardware⁶.

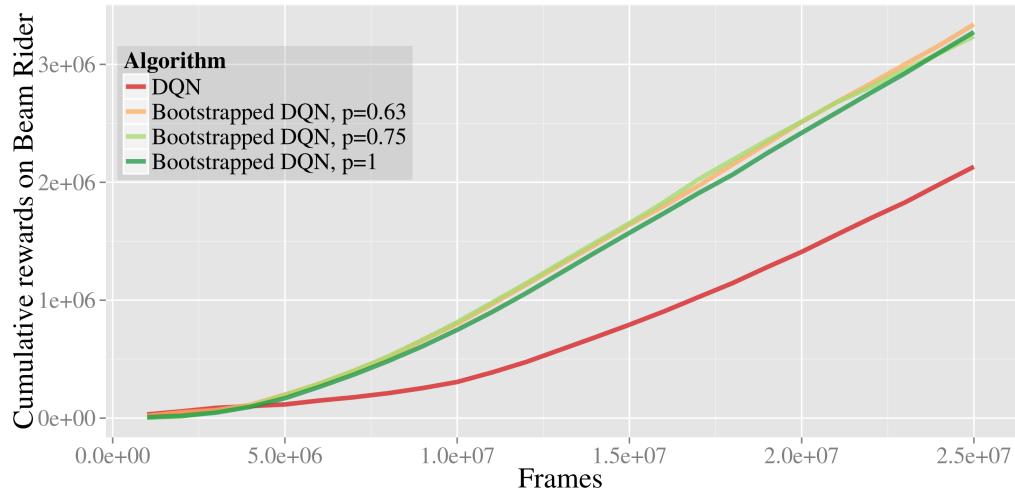


Figure 8.8: We found similar performance across a range of p .

⁴For more details on gradient rescaling see Appendix ??.

⁵ $p = 0.5$ is double-or-nothing bootstrap [66].

⁶Our implementation $K=10$, $p=1$ ran with less than a 20% increase on wall-time versus DQN for the same amount of frames.

8.4.2 Efficient exploration in Atari

We find that Bootstrapped DQN drives efficient exploration in several Atari games. This means that, 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.

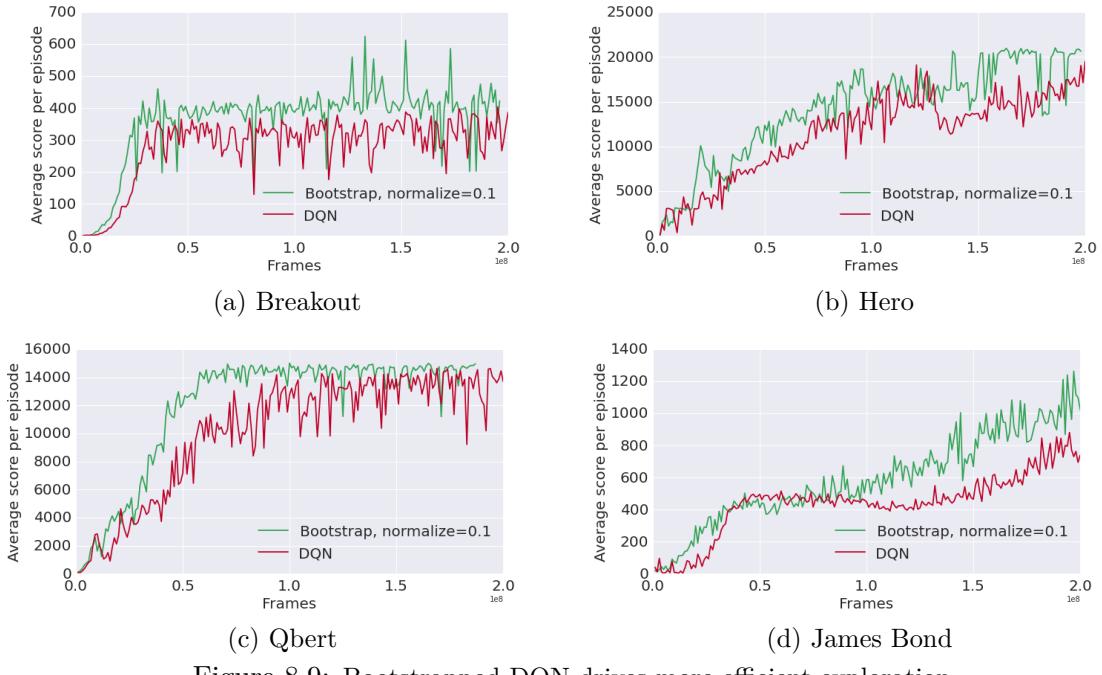


Figure 8.9: Bootstrapped DQN drives more efficient exploration.

To summarize this improvement in learning time we consider the number of frames required to reach human performance. We say algorithm A has a human speedup of x relative to algorithm B if it reaches human performance in $1/x$ as many frames as DQN. We find that, for the majority of games where DQN reaches human performance, Bootstrapped DQN reaches human performance significantly faster. We present these results in Figure 8.10.

8.4.3 Understanding bootstrapped DQN

We will now present some insight into how bootstrapped DQN uses deep exploration to improve upon DQN. First, we will examine the exploration policies for the game Hero.

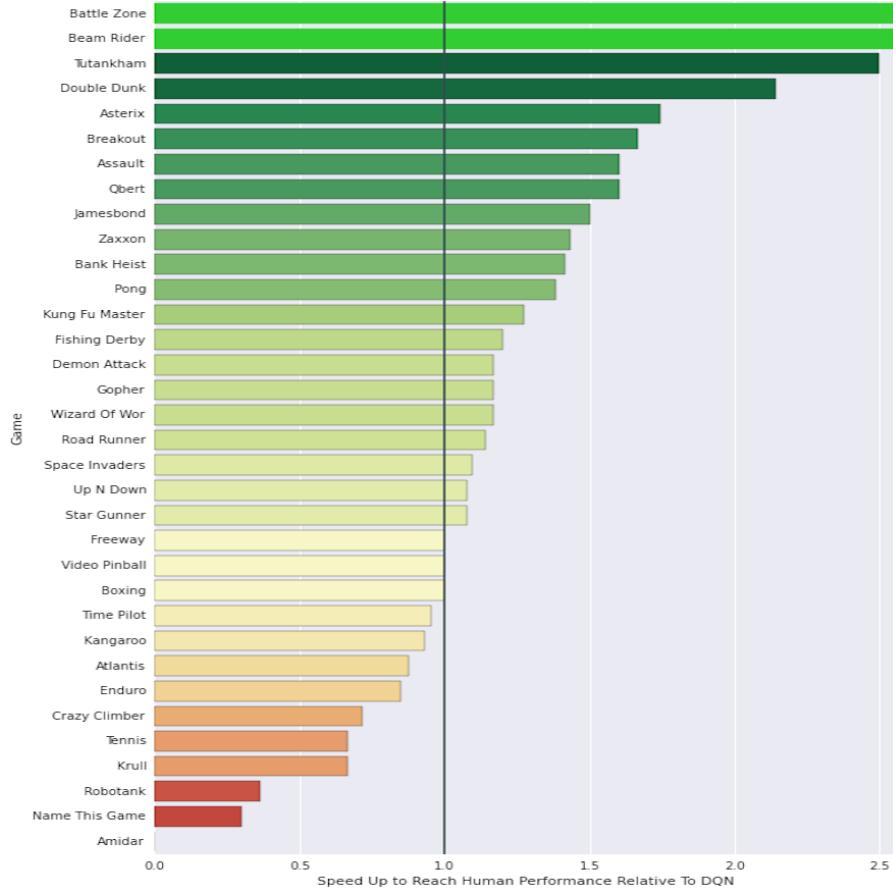


Figure 8.10: Bootstrapped DQN at human level faster than DQN.

In this game the agent controls a rescue mission which travels room to room, blows up obstacles and rescues hostages. In Figure 8.11 we show a screenshot for nine different heads of bootstrapped DQN run from the initial state for 3,200 steps. Although each individual head has learned a good representation for the value function that leads to a high-scoring policy, the policies they find are quite distinct.

In fact, this screenshot shows the nine different heads in five different game rooms. This shows that even after more than 100m training frames with the data sharing $p = 1$, we still maintain significant diversity. By contrast, ϵ -greedy strategies are almost indistinguishable for small values of ϵ and totally ineffectual for larger values. Our heads explore a diverse range of policies, but still manage to each perform well individually. We believe that this deep exploration is key to the improved learning visible in Figure 8.9b, since diverse experiences allow for better generalization.

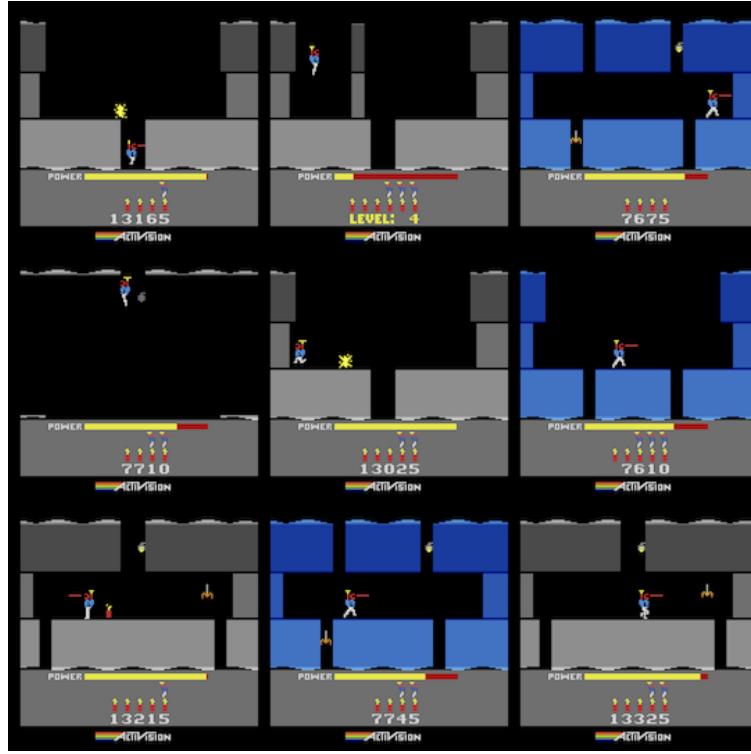


Figure 8.11: Diverse exploration policies in Hero.

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 across heads. 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.

In Figure 8.12 we present two screenshots of bootstrapped DQN playing Breakout according to an ensemble policy. In this game the agent can choose to move left, move right or stay. Beneath each screen we depict the number of heads that pick each action by the length of each arrow. We draw the action chosen by the ensemble policy in green.

When the ball is approaching the bottom of the screen and all ten heads recognize the optimal policy is to move right to catch it (Figure 8.12a). However when the ball is far from the paddle the we can see the ensemble policy is uncertain on the best action (Figure 8.12b). 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.

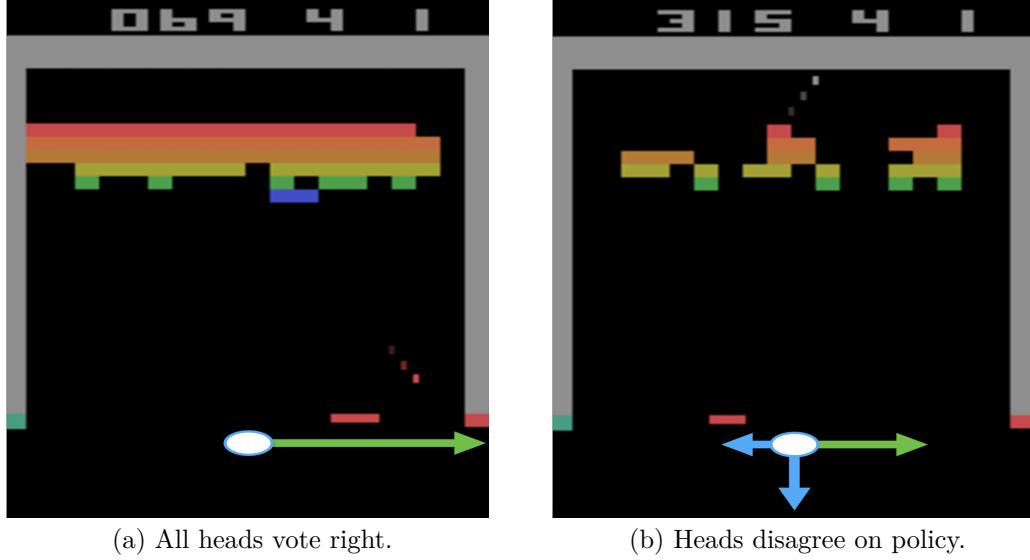


Figure 8.12: Visualizing uncertainty in an ensemble policy.

8.4.4 Overall performance

We have seen that bootstrapped DQN is able to learn much faster than DQN. In Figure 8.13 we see that best performance reached by both algorithms is similar across most games. However, the benefits of efficient exploration mean that bootstrapped DQN greatly outperforms DQN when measured in terms of *cumulative* rewards through learning. We present these results in Figure 8.14. In Appendix ?? we present full results for our algorithm across all 49 games. Bootstrapped DQN significantly outperforms several recent heuristic approaches for improved exploration on Atari [76], particularly in terms of cumulative rewards.

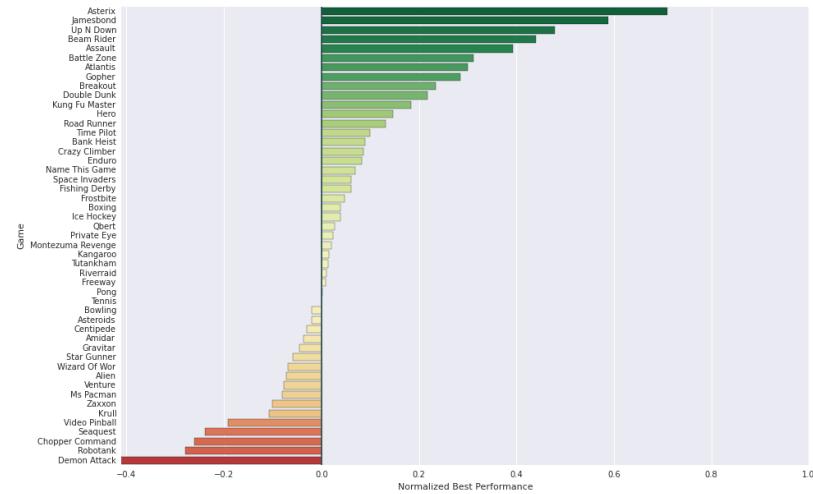


Figure 8.13: Bootstrapped DQN attains similar best policy.

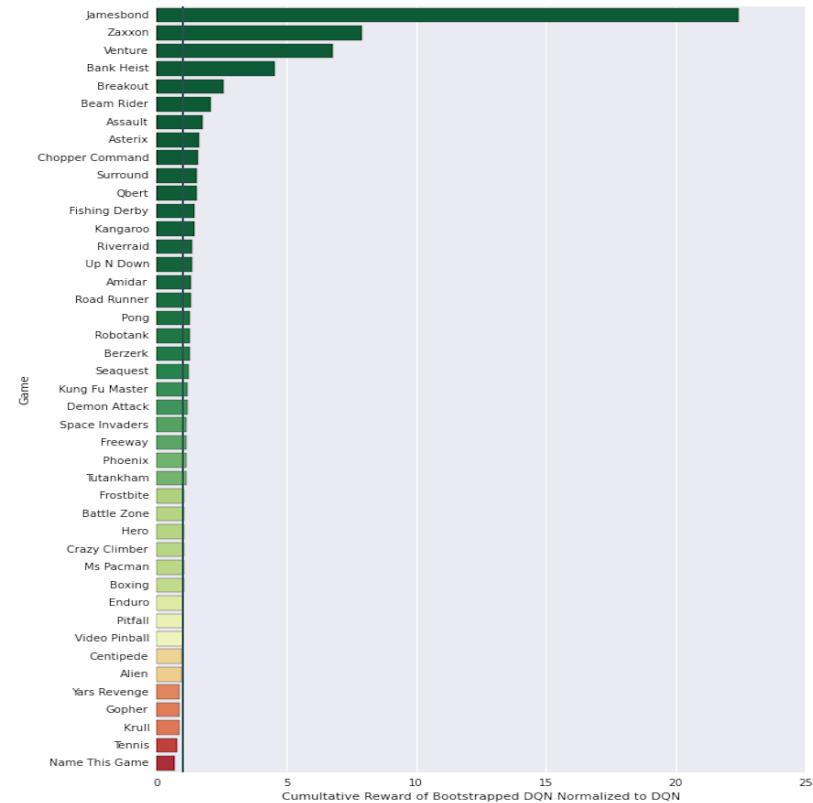


Figure 8.14: Bootstrapped DQN improves cumulative rewards.

Bibliography

- [1] Yasin Abbasi-Yadkori and Csaba Szepesvári. Regret bounds for the adaptive control of linear quadratic systems. *Journal of Machine Learning Research - Proceedings Track*, 19:1–26, 2011.
- [2] Yasin Abbasi-Yadkori and Csaba Szepesvári. Bayesian optimal control of smoothly parameterized systems. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2015.
- [3] Shipra Agrawal and Navin Goyal. Further optimal regret bounds for Thompson sampling. *arXiv preprint arXiv:1209.3353*, 2012.
- [4] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. *arXiv preprint arXiv:1209.3352*, 2012.
- [5] Mauricio Araya, Olivier Buffet, and Vincent Thomas. Near-optimal brl using optimistic local transitions. *arXiv preprint arXiv:1206.4613*, 2012.
- [6] Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. *Mathematical finance*, 9(3):203–228, 1999.
- [7] 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.
- [8] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2003.
- [9] 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.
- [10] 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.

- [11] 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.
- [12] 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.
- [13] Dimitri P. Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, September 1996.
- [14] Peter J Bickel and David A Freedman. Some asymptotic theory for the bootstrap. *The Annals of Statistics*, pages 1196–1217, 1981.
- [15] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *ICML*, 2015.
- [16] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. OUP Oxford, 2013.
- [17] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1):49–107, 2000.
- [18] 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.
- [19] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *CoRR*, abs/1204.5721, 2012.
- [20] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12:1587–1627, 2011.
- [21] Valerii V Buldygin and Yu V Kozachenko. Sub-gaussian random variables. *Ukrainian Mathematical Journal*, 32(6):483–489, 1980.
- [22] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.
- [23] Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems*, page TBA, 2015.
- [24] Richard Dearden, Nir Friedman, and Stuart J. Russell. Bayesian Q-learning. In *AAAI/IAAI*, pages 761–768, 1998.
- [25] Joseph L Doob. *Measure theory*, volume 143. Springer Science & Business Media, 2012.
- [26] John C Duchi. *Multiple Optimality Guarantees in Statistical Learning*. PhD thesis, Citeseer, 2014.

- [27] Bradley Efron. *The jackknife, the bootstrap and other resampling plans*, volume 38. SIAM, 1982.
- [28] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.
- [29] 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.
- [30] Raphaël Fonteneau, Nathan Korda, and Rémi Munos. An optimistic posterior sampling strategy for bayesian reinforcement learning. In *NIPS 2013 Workshop on Bayesian Optimization (BayesOpt2013)*, 2013.
- [31] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv preprint arXiv:1506.02142*, 2015.
- [32] Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*, volume 2. Taylor & Francis, 2014.
- [33] Aditya Gopalan and Shie Mannor. Thompson sampling for learning parameterized markov decision processes. *arXiv preprint arXiv:1406.7498*, 2014.
- [34] Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.
- [35] 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.
- [36] 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.
- [37] Arthur Guez, David Silver, and Peter Dayan. Better optimism by bayes: Adaptive planning with rich models. *arXiv preprint arXiv:1402.1958*, 2014.
- [38] Josef Hadar and William R Russell. Rules for ordering uncertain prospects. *The American Economic Review*, pages 25–34, 1969.
- [39] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.
- [40] José Miguel Hernández-Lobato and Ryan P Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. *ICML*, 2015.

- [41] Morteza Ibrahimi, Adel Javanmard, and Benjamin Van Roy. Efficient reinforcement learning for high dimensional linear quadratic systems. In *NIPS*, pages 2645–2653, 2012.
- [42] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.
- [43] 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.
- [44] Sham Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, University College London, 2003.
- [45] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *Algorithmic Learning Theory*, pages 199–213. Springer, 2012.
- [46] 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.
- [47] Michael J. Kearns and Satinder P. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.
- [48] Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *arXiv preprint arXiv:1506.02557*, 2015.
- [49] 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.
- [50] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.
- [51] 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.
- [52] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.
- [53] Erich Leo Lehmann, George Casella, and George Casella. *Theory of point estimation*. Wadsworth & Brooks/Cole Advanced Books & Software, 1991.
- [54] Erich Leo Lehmann, Joseph P Romano, and George Casella. *Testing statistical hypotheses*, volume 150. Wiley New York et al, 1986.
- [55] 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.

- [56] 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.
- [57] Volodymyr et al. Mnih. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [58] Rémi Munos. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. 2014.
- [59] Ronald Ortner and Daniil Ryabko. Online regret bounds for undiscounted continuous reinforcement learning. In *NIPS*, 2012.
- [60] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *arXiv preprint arXiv:1602.04621*, 2016.
- [61] Ian Osband, Daniel Russo, and Benjamin Van Roy. (More) efficient reinforcement learning via posterior sampling. In *NIPS*, pages 3003–3011. Curran Associates, Inc., 2013.
- [62] Ian Osband and Benjamin Van Roy. Model-based reinforcement learning and the eluder dimension. In *Advances in Neural Information Processing Systems*, pages 1466–1474, 2014.
- [63] Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored MDPs. In *Advances in Neural Information Processing Systems*, pages 604–612, 2014.
- [64] Ian Osband and Benjamin Van Roy. Bootstrapped thompson sampling and deep exploration. *arXiv preprint arXiv:1507.00300*, 2015.
- [65] Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. *arXiv preprint arXiv:1402.0635*, 2014.
- [66] Art B Owen, Dean Eckles, et al. Bootstrapping data arrays of arbitrary order. *The Annals of Applied Statistics*, 6(3):895–927, 2012.
- [67] Donald B Rubin et al. The bayesian bootstrap. *The annals of statistics*, 9(1):130–134, 1981.
- [68] Walter Rudin. *Principles of mathematical analysis*, volume 3. McGraw-Hill New York, 1964.
- [69] Paat Rusmevichientong and John N. Tsitsiklis. Linearly parameterized bandits. *Math. Oper. Res.*, 35(2):395–411, 2010.
- [70] Dan Russo and Benjamin Van Roy. Eluder dimension and the sample complexity of optimistic exploration. In *NIPS*, pages 2256–2264. Curran Associates, Inc., 2013.
- [71] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

- [72] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.
- [73] Steven L Scott. A modern bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.
- [74] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.
- [75] 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.
- [76] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.
- [77] Alexander L. Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L. Littman. PAC model-free reinforcement learning. In *ICML*, pages 881–888, 2006.
- [78] 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.
- [79] Malcolm J. A. Strens. A bayesian framework for reinforcement learning. In *ICML*, pages 943–950, 2000.
- [80] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, March 1998.
- [81] Richard Stuart Sutton. Temporal credit assignment in reinforcement learning. 1984.
- [82] Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.
- [83] 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.
- [84] Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.
- [85] 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.

- [86] Nikos Vlassis, Mohammad Ghavamzadeh, Shie Mannor, and Pascal Poupart. Bayesian reinforcement learning. In *Reinforcement Learning*, pages 359–386. Springer, 2012.
- [87] 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.
- [88] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.
- [89] Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J Weinberger. Inequalities for the l_1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep*, 2003.
- [90] 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.