Deep Exploration via Randomized Value Functions

Ian Osband   IOSBAND@GOOGLE.COM
DeepMind

Benjamin Van Roy   BVR@STANFORD.EDU
Stanford University

Daniel J. Russo   DJR2174@GSB.COLUMBIA.EDU
Columbia University

Zheng Wen   ZWEN@ADOBE.COM
Adobe Research

Abstract

We study the use of randomized value functions to guide deep exploration in reinforcement learning. This offers an elegant means for synthesizing statistically and computationally efficient exploration with common practical approaches to value function learning. We present several reinforcement learning algorithms that leverage randomized value functions and demonstrate their efficacy through computational studies. We also prove a regret bound that establishes statistical efficiency with a tabular representation.

Keywords: Reinforcement learning, exploration, value function, neural network

1. Introduction

Reinforcement learning might provide the basis for an artificial intelligence that can manage a wide range of systems and better serve the needs of society. To date, its potential has primarily been assessed through learning in simulated systems, where data generation is relatively unconstrained and algorithms are routinely trained over millions to trillions of episodes. Real systems, where data collection is costly or constrained by the physical context, call for a focus on statistical efficiency. A key driver of statistical efficiency is how the agent explores its environment.

The design of reinforcement learning algorithms that efficiently explore intractably large state spaces remains an important challenge. Though a substantial body of work addresses efficient exploration, most of this focusses on tabular representations in which the number of parameters learned and the quantity of data required scale with the number of states. Despite valuable insights that have been generated through design and analysis of tabular reinforcement learning algorithms, they are of limited practical import because, due to the curse of dimensionality, state spaces in most contexts of practical interest are enormous. There is a need for algorithms that generalize across states while exploring intelligently to learn to make effective decisions within a reasonable time frame.

In this paper, we develop a new approach to exploration that serves this need. We build on value function learning, which underlies the most popular and successful approaches to reinforcement learning. In common value function learning approaches, the agent maintains

a point estimate of a function mapping state-action pairs to expected cumulative future reward. This estimate typically takes a parameterized form, such as a linear combination of features or a neural network, with parameters fit to past observations. The estimate approximates the agent's prevailing expectation of the true value function, and can be used to guide action selection. As actions are applied and new observations gathered, parameters are adapted to fit the growing data set. The hope is that this process quickly converges on a mode in which the agent selects near optimal actions and new observations reinforce prevailing value estimates.

In using the value function estimate to guide actions, the agent could operate according to a greedy policy, which at any given state, applies the action that maximizes estimated value. However, such a policy does not investigate poorly-understood actions that are assigned unattractive point estimates. This can forgo enormous potential value; it is worthwhile to experiment with such an action since the action could be optimal, and learning that can provide cumulating future benefit over subsequent visits to the state. Thoughtful exploration can be critical to effective learning.

The simplest and most widely used approaches to exploration perturb greedy actions with random *dithering*. An example is ϵ-greedy exploration, which selects the greedy action with probability $1 - \epsilon$ and otherwise selects uniformly at random from all currently available actions. Dithering induces the experimentation required to learn about actions with unattractive point estimates. However, such approaches waste much exploratory effort because they do not "write-off" actions that are known to be inferior. This is because exploratory actions are selected without regard to the level of uncertainty associated with value estimates. Clearly, it is only worth experimenting with an action that is expected to be undesirable if there is sufficient uncertainty surrounding that assessment. As we will discuss further in Section 4, this inefficiency can result in learning times that grow exponentially with the number of states.

A more sophisticated approach might only experiment with an action when applying the action will reveal useful information. We refer to such approaches as *myopic*, since they do not account for subsequent learning opportunities made possible by taking an action. Though myopic approaches do "write off" actions where dithering approaches fail to, as we will discuss in Section 4, myopic exploration can also require learning times that grow exponentially with the number of states or even entirely fail to learn.

Reliably efficient reinforcement learning calls for *deep exploration*. By this we mean that the exploration method does not only consider immediate information gain but also the consequences of an action on future learning. A deep exploration method could, for example, choose to incur losses over a sequence of actions while only expecting informative observations after multiple time periods. Dithering and myopic approaches do not exhibit such strategic pursuit of information.

In this paper, we develop a new approach to deep exploration. The idea is to apply actions that are greedy with respect to a randomly drawn statistically plausible value function. Roughly speaking, we aim to sample from a proxy of the posterior distribution over value functions. Such randomized value functions incentivize experimentation with actions of highly uncertain value, since this uncertainty translates into variance in the sampled value estimate. This randomness often generates positive bias and therefore induces exploration.

2

There is much more to be said about the design of algorithms that leverage randomized value functions, and we cover some of this ground in Section 5. It is worth mentioning here, though, that this concept is abstract and broadly applicable, transcending specific algorithms. Randomized value functions can be synthesized with the multitude of useful algorithmic ideas in the reinforcement learning literature to produce custom approaches for specific contexts.

To provide insight into the efficacy of randomized value functions, in Section 6, we establish a strong bound on the Bayesian regret of a tabular algorithm. This is not the first result to establish strong efficiency guarantees for tabular reinforcement learning. However, previous algorithms that have been shown to satisfy similar regret bounds do not extend to contexts involving generalization via parameterized value functions. In this regard, the approach we present is the first to satisfy a strong regret bound with tabular representations while also working effectively with the wide variety of practical value function learning methods that generalize over states and actions.

Section 7 presents computational results guided by randomized value functions that synthesize efficient exploration with generalization. Experiments with a family of simple toy examples demonstrate dramatic efficiency gains relative to dithering approaches for exploration and that our randomized approaches are compatible with linearly parameterized generalizing value functions. We also consider a cart-pole balancing problem that requires both deep exploration and generalization. We address this problem through a combination of randomization and deep learning, with value functions represented by neural networks.

2. Literature review

The Bayes-optimal policy serves as a gold standard for statistically efficient exploration in reinforcement learning (RL). Given a prior distribution over Markov decision processes, one can formulate a problem to maximize expected cumulative reward by taking an action at each future time contingent on the prevailing posterior distribution. A policy that attains this maximum is Bayes-optimal, and to do this it must explore judiciously. Unfortunately, for problems of practical interest, computing a Bayes-optimal policy is intractable; the computational requirements grow exponentially in the problem parameters (Szepesvári, 2010). We introduce an approach based on *randomized value functions* that offers a computationally tractable approach to statistically efficient reinforcement learning. Exploration via randomized value functions is not generally Bayes-optimal but, as we will argue, offers a practical approach to deep exploration, which common exploration schemes fail to address, sometimes at enormous cost to statistical efficiency.

There is a substantial body of work on simultaneously computationally and statistically efficient exploration in tabular RL. This begins with the seminal work of Kearns and Singh (Kearns and Singh, 2002), which identified the necessity of multi-period exploration strategies – for which we adopt the term *deep exploration* – to polynomial-time learning and established a polynomial-time learning guarantee for a particular tabular algorithm. Subsequent papers proposed and analyzed alternative tabular algorithms that carry out deep exploration with varying degrees of efficacy (Brafman and Tennenholtz, 2002; Auer and Ortner, 2006; Strehl et al., 2006; Jaksch et al., 2010; Osband et al., 2013; Dann and Brunskill, 2015; Azar et al., 2017). None of these algorithms are Bayes-optimal, but they do

bound the level of sub-optimality by some polynomial function of states and/or planning horizon. By contrast, popular schemes such as ϵ-greedy and Boltzmann exploration can require learning times that grow exponentially in the number of states and/or the planning horizon (see, e.g., Kakade (2003); Strehl (2007)). We discuss this phenomenon further in Section 4.

The design and analysis of tabular algorithms has generated valuable insights, but the resultant algorithms are of little practical importance since, for practical problems the state space is typically enormous (due to the curse of dimensionality). To learn effectively, practical RL algorithms must generalize across states to make effective decisions with limited data. The literature offers a rich collection of such algorithms (e.g. Bertsekas and Tsitsiklis (1996); Sutton and Barto (2018); Szepesvári (2010); Powell and Ryzhov (2011) and references therein). Though algorithms of this genre have achieved impressive outcomes, notably in games such as backgammon (Tesauro, 1995), Atari arcade games (Mnih et al., 2015), and go (Silver et al., 2016, 2017), they use naive exploration schemes that can be highly inefficient. Possibly for this reason, these applications required enormous quantities of data. In the case of Silver et al. (2016), for example, neural networks were trained over hundreds of billions to trillions of simulated games.

The design of reinforcement learning algorithms that efficiently explore intractably large state spaces remains an important challenge. Model learning algorithms exploit generalization in an underlying model of the environment (Kearns and Koller, 1999; Abbasi-Yadkori and Szepesvári, 2011; Ibrahimi et al., 2012; Ortner and Ryabko, 2012; Osband and Van Roy, 2014a,b; Gopalan and Mannor, 2015). However, these are typically restricted to simple model classes and become statistically or computationally intractable for problems of practical scale. Policy learning algorithms, and the closely-related 'evolutionary' algorithms identify high-performers among a set of policies (Kakade, 2003; Wierstra et al., 2008; Deisenroth et al., 2013; Plappert, 2017). These algorithms can perform well, particularly when the space of possible optimal policies is parameterized to be small. However, in a typical problem the space of policies is exponentially large; existing works either entail overly restrictive assumptions or do not make strong efficiency guarantees.

Value function learning has the potential to overcome computational challenges and offer practical means for synthesizing efficient exploration and effective generalization. A relevant line of work establishes that efficient reinforcement learning with value function generalization reduces to efficient "knows what it knows" (KWIK) online regression (Li and Littman, 2010; Li et al., 2008). However, it is not known whether the KWIK online regression problem can be solved efficiently. In terms of concrete algorithms, there is optimistic constraint propagation (OCP) (Wen and Van Roy, 2013), a provably efficient reinforcement learning algorithm for exploration and value function generalization in deterministic systems, and C-PACE (Pazis and Parr, 2013), a provably efficient reinforcement learning algorithm that generalizes using interpolative representations. These contributions represent important developments, but OCP is not suitable for stochastic systems and is highly sensitive to model misspecification, and generalizing effectively in high-dimensional state spaces calls for methods that extrapolate.

In this paper, we leverage randomized value functions to explore efficiently while generalizing via parameterized value functions. Algorithms of the kind we consider, which we refer to collectively as randomized least-squares value iteration (RLSVI), were first introduced in

Wen (2014). Prior reinforcement learning algorithms that generalize via parameterized value functions require, in the worst case, learning times exponential in the number of model parameters and/or the planning horizon. RLSVI aims to overcome these inefficiencies. While RLSVI operates in a manner similar to well-known approaches such as least-squares value iteration (LSVI) and SARSA (see, e.g. Sutton and Barto (2018)), what fundamentally distinguishes RLSVI is exploration through randomly sampling statistically plausible value functions. Alternatives such as LSVI and SARSA are typically applied in conjunction with action-dithering schemes such as Boltzmann or ϵ-greedy exploration, which lead to highly inefficient learning.

This paper aims to establish the use of *randomized value functions* as a promising approach to tackling a critical challenge in reinforcement learning: synthesizing efficient exploration and effective generalization. The only preceding work that advocates exploration through random samples of the value function comes from Dearden et al. (1998). This paper proposes a tabular algorithm that resamples every timestep and so does not perform deep exploration. A preliminary version of part of this work appeared in a short conference paper (Osband et al., 2016b). While that paper proposed a specific algorithm that is compatible with linear function approximation, this paper develops the concept of deep exploration via randomized value functions in much greater depth and generality. We provide a general template for building algorithms that perform randomized value function learning and propose several specific instantiations of this idea. These algorithms are evaluated in entirely new simulations experiments, including an example in which neural networks are used for function approximation. While a proof sketch for a regret bound was given in (Osband et al., 2016b), this paper gives a full and careful proof and develops new recursive stochastic dominance arguments that simplify the analysis. Following (Osband et al., 2016b), several papers have proposed adaptations to neural network function approximation via bootstrap sampling (Osband et al., 2016a), linear final layer approximation (Azizzadenesheli et al., 2018) or variational inference (Lipton et al., 2018; Fortunato et al., 2018).

The mathematical analysis we present in Section 6 establishes a bound on expected regret for a tabular version of RLSVI applied to an episodic finite-horizon problem, where the expectation is taken with respect to a particular uninformative distribution. We view this result as a sanity check that, although it is designed for exploration with generalization, RLSVI recovers state-of-the-art efficiency guarantees in the simple tabular setting. Our bound is $\tilde{O}(H\sqrt{SAHL})$, where S and A denote the cardinalities of the state and action spaces, L denotes the number of episodes elapsed, and H denotes the episode duration. The lower bound of Jaksch et al. (2010) can be adapted to the episodic finite-horizon context to produce a $\Omega(H\sqrt{SAL})$ lower bound that applies to any algorithm. This differs from our upper bound by a factor of $\sqrt{H}$, though this is not an apples-to-apples comparison, since the lower bound applies to a maximum over Markov decision processes and may not hold for the expectation over Markov decision processes, taken with respect to a prior distribution we posit. Follow up work by Russo (2019) shows that RLSVI also satisfies worst-case regret bounds in tabular environments.

In recent years there has been significant interest in alternative methods to incentivize exploration. One popular method uses a density model or "pseudocount" to assign a bonus to states that have been visited infrequently (Bellemare et al., 2016; Tang et al., 2016). These methods can perform well, but only when the generalization of the density model

is aligned with the task objective. Crucially, this generalization is not learned from the task and, unlike the optimal value function, "counts" are generated by the agent's choices so there is no single target function to learn. Further, these approaches add uncertainty bonus that is uncoupled across states, which can lead to a substantial negative impact on statistical efficiency, as discussed in (Osband and Van Roy, 2017; O'Donoghue et al., 2017).

Exploration via randomized value functions is inspired by Thompson sampling (Thompson, 1933; Russo et al., 2018). In particular, when generating a randomized value function, the aim is to approximately sample from the posterior distribution of the optimal value function. There are problems where Thompson sampling is in some sense near-optimal (Agrawal and Goyal, 2012, 2013a,b; Russo and Van Roy, 2013, 2014a; Gopalan and Mannor, 2015). Further, the theory suggests that "well-designed" upper-confidence-bound-based approaches, which appropriately couple uncertainties across state-action pairs, but are often computationally intractable, are similarly near-optimal (statistically) and competitive with Thompson sampling in such contexts (Russo and Van Roy, 2013, 2014a). On the other hand, for some problems with more complex information structures, it is possible to explore much more efficiently than do Thompson sampling or upper-confidence-bound methods (Russo and Van Roy, 2014b). As such, for some RL problems and value function representations, the randomized value function approaches we put forth will leave substantial room for improvement.

At a high level, randomized value functions replaces a point estimate of the value function by a distribution of plausible value functions. Recently, another approach called "distributional RL" also suggests replacing a scalar value estimate by a distribution (Bellemare et al., 2017). Although both might reasonably claim to offer a distributional perspective on reinforcement learning, the meaning and utility of the two distributions are quite distinct. Randomized value functions aim to sample from a distribution that captures the Bayesian uncertainty in the unknown optimal value function; this concentrates around the true value function as more data is gathered. By contrast, "distributional RL" fits a distribution to the realized value under stochastic outcomes. For efficient exploration of *unknown* rather than stochastic outcomes, it is important to use the correct notion of "distributional RL".

3. Reinforcement learning problem

We consider a reinforcement learning problem in which an agent interacts with an unknown environment over a sequence of episodes. We model the environment as a Markov decision process, identified by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \rho)$. Here, $\mathcal{S}$ is a finite state space, $\mathcal{A}$ is a finite action space, $\mathcal{R}$ is a reward model, $\mathcal{P}$ is a transition model, and $\rho \in \mathcal{S}$ is an initial state distribution. For each s, $\rho(s)$ is the probability that an episode begins in state s. For any $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$, $\mathcal{R}_{s,a,s'}$ is a distribution over real numbers and $\mathcal{P}_{s,a}$ is a sub-distribution over states. In particular, $\mathcal{P}_{s,a}(s')$ is the conditional probability that the state transitions to s' from state s and action a. Similarly, $\mathcal{R}_{s,a,s'}(dr)$ is the conditional probability that the reward is in the set dr. By *sub-distribution*, we mean that the sum can be less than one. The difference $1 - \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,a}(s')$ represents the probability that the process terminates upon transition.

We will denote by r_t^ℓ, s_t^ℓ, a_t^ℓ the state, action, and reward observed at the start of the tth time period of the ℓth episode. In each ℓth episode, the agent begins in a random

6

state $s_0^\ell \sim \rho$ and selects an action $a_0^\ell \in \mathcal{A}$. Given this state-action pair, a reward and transition are generated according to $r_1^\ell \sim \mathcal{R}_{s_0^\ell, a_0^\ell, s_1^\ell}$ and $s_1^\ell \sim \mathcal{P}_{s_0^\ell, a_0^\ell}$. The agent proceeds until termination, in each tth time period observing a state s_t^ℓ, selecting an action a_t^ℓ, and then observing a reward r_{t+1}^ℓ and transition to s_{t+1}^ℓ. Let τ_ℓ denote the random time at which the process terminates, so that the sequence of observations made during episode ℓ is $\mathcal{O}_\ell = \left(s_0^\ell, a_0^\ell, r_1^\ell, s_1^\ell, a_1^\ell, \ldots, s_{\tau_\ell - 1}^\ell, a_{\tau_\ell - 1}^\ell, r_{\tau_\ell}^\ell\right)$.

We define a *policy* to be a mapping from $\mathcal{S}$ to a probability distribution over $\mathcal{A}$, and denote the set of all policies by Π. We will denote by $\pi(a|s)$ the probability that π assigns to action a at state s. Without loss of generality, we will consider states and actions to be integer indices, so that $\mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$ and $\mathcal{A} = \{1, \ldots, |\mathcal{A}|\}$. As such, we can define a substochastic matrix whose (s, s')th element is $\sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}_{s,a}(s')$. We make the following assumption to ensure finite episode duration:

Assumption 1 *For all policies $\pi \in \Pi$, if each action a_t is sampled from $\pi(\cdot|s_t)$, then the MDP $\mathcal{M}$ almost surely terminates in finite time. In other words, $\lim_{t \to \infty} P_\pi^t = 0$, where P_π is the matrix whose (s, s')th element is $\sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}_{s,a}(s')$.*

For any MDP $\mathcal{M}$ and policy $\pi \in \Pi$, we define a value function $V_{\mathcal{M}}^\pi : \mathcal{S} \mapsto \mathbb{R}$ by

$$V_{\mathcal{M}}^\pi(s) = \mathbb{E}_{\mathcal{M}, \pi}\left[\sum_{t=1}^{\tau} r_t \,\Big|\, s_0 = s\right],$$

where r_t, s_t, a_t, and τ denote rewards, states, actions, and termination time of a generic episode, and the subscripts of the expectation indicate that actions are sampled according to $a_t \sim \pi(\cdot|s_t)$ and transitions and rewards are generated by the MDP $\mathcal{M}$. Further, we define an optimal value function:

$$V_{\mathcal{M}}^*(s) = \max_{\pi \in \Pi} V_{\mathcal{M}}^\pi(s).$$

The agent's behavior is governed by a reinforcement learning algorithm alg. Immediately prior to the beginning of episode L, the algorithm produces a policy $\pi^L = \text{alg}(\mathcal{S}, \mathcal{A}, \mathcal{H}_{L-1})$ based on the state and action spaces and the history $\mathcal{H}_{L-1} = (\mathcal{O}_\ell : \ell = 1, \ldots, L-1)$ of observations made over previous episodes. Note that alg may be a randomized algorithm, so that multiple applications of alg may yield different policies.

In episode ℓ, the agent enjoys a cumulative reward of $\sum_{t=1}^{\tau_\ell} r_t^\ell$. We define the *regret* over episode ℓ to be the difference between optimal expected value and the expected value under algorithm alg. This can be written as $\mathbb{E}_{\mathcal{M}, \text{alg}}\left[V^*(s_0^\ell) - V^{\pi^\ell}(s_0^\ell))\right]$, where the subscripts of the expectation indicate that each policy π^ℓ is produced by algorithm alg and state transitions and rewards are generate by MDP $\mathcal{M}$. Note that this expectation integrates over all initial states, actions, state transitions, rewards, and any randomness generated within alg, while the MDP $\mathcal{M}$ is fixed. We denote *cumulative regret* over L episodes by

$$\text{Regret}(\mathcal{M}, \text{alg}, L) = \sum_{\ell=1}^{L} \mathbb{E}_{\mathcal{M}, \text{alg}}\left[V^*(s_0^\ell) - V^{\pi^\ell}(s_0^\ell))\right].$$

We will generally refer to *cumulative regret* simply as *regret*.

When used as a measure for comparing algorithms, one issue with regret is its dependence on $\mathcal{M}$. One way of addressing this is to assume that $\mathcal{M}$ is constrained to a pre-defined set and to design algorithms with an aim of minimizing worst-case regret over this set. This tends to yield algorithms that behave in an overly conservative manner when faced with representative MDPs. An alternative is to aim at minimizing an average over representative MDPs. The distribution over MDPs can be thought of as a prior, which captures beliefs of the algorithm designer. In this spirit, we define *Bayesian regret*:

$$\text{BayesRegret}(\text{alg}, L) = \mathbb{E}\left[\text{Regret}(\mathcal{M}, \text{alg}, L)\right].$$

Here, the expectation integrates with respect to a prior distribution over MDPs.

It is easy to see that minimizing regret or Bayesian regret is equivalent to maximizing expected cumulative reward. These measures are useful alternatives to expected cumulative reward, however, because for reasonable algorithms, $\text{Regret}(\mathcal{M}, \text{alg}, L)/L$ and $\text{BayesRegret}(\text{alg}, L)/L$ should converge to zero. When it is not feasible to apply an optimal algorithm, comparing how quickly these values diminish and how that depends on problem parameters can yield insight.

To denote our prior distribution over MDPs, as well as distributions over any other randomness that is realized, we will use a probability space $(\Omega, \mathbb{F}, \mathbb{P})$. With this notation, the probability that $\mathcal{M}$ takes values in a set $\mathbb{M}$ is written as $\mathbb{P}(\mathcal{M} \in \mathbb{M})$. In fact, the probability of any measurable event $\mathcal{E}$ is written as $\mathbb{P}(\mathcal{E})$.

4. Deep exploration

Reinforcement learning calls for a sophisticated form of exploration that we refer to as *deep exploration*. This form of exploration accounts not only for information gained upon taking an action but also for how the action may position the agent to more effectively acquire information over subsequent time periods. We will use the following simple example to illustrate the critical role of deep exploration as well as how common approaches to exploration fall short on this front.

Example 1 (Deep-sea exploration)
Consider an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \rho)$ with $|\mathcal{S}| = N^2$ states, each of which can be thought of as a square cell in an $N \times N$ grid, as illustrated in Figure 1. The action space is $\mathcal{A} = \{1, 2\}$. At each state, one of the actions represents "left" and the other represents "right," with the indexing possibly differing across states. In other words, for a pair of distinct states $s, s' \in \mathcal{S}$, action 1 could represent "left" at state s and "right" at state s'. Any transition from any state in the lowest row leads to termination of the episode. At any other state, the "left" action transitions to the cell immediately to the left, if possible, and below. Analogously, the "right" action transitions to the cell immediately to the right, if possible, and below. The agent begins every episode in the upper-left-most state (where her boat sits). Note that, given the dynamics we have described, each episode lasts exactly N time periods.

From any cell along the diagonal, there is a cost of $0.01/N$ incurred each time the "right" action is chosen. No cost is incurred for the left action. The only other situation that leads to an additional reward or cost arises when the agent is in the lower-right-most cell, where there is a chest. There is an additional reward of 1 (treasure) or cost of 1 (bomb) when the
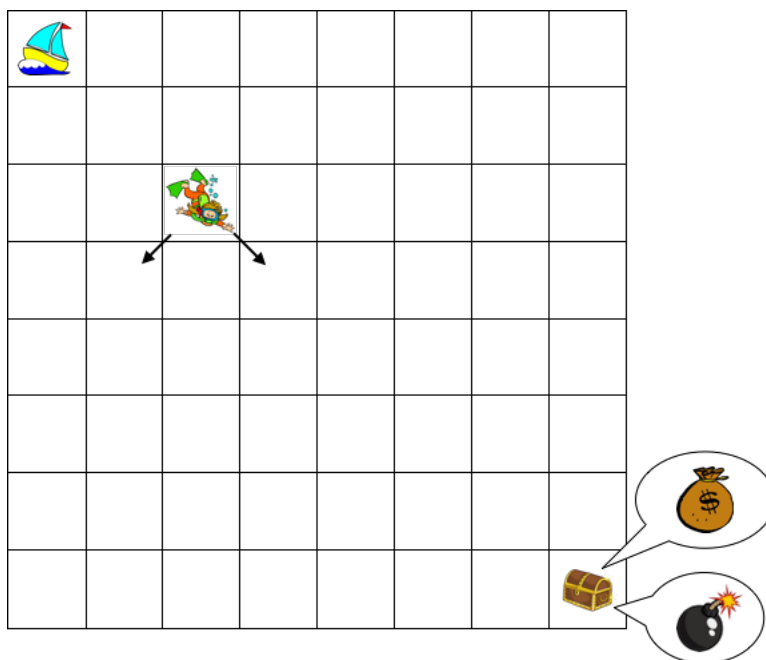
Figure 1: Deep-sea exploration: a simple example where deep exploration is critical.

"right" action is selected at that cell. Conditioned on the $\mathcal{M}$, this reward is deterministic, so once the agent discovers whether there is treasure or a bomb, she knows in subsequent episodes whether she wants to reach or avoid that cell. In particular, given knowledge of $\mathcal{M}$, the optimal policy is to select the "right" action in every time period if there is treasure and, otherwise, to choose the "left" action in every time period. Doing so accumulates a reward of 0.99 if there is treasure and 0 if there is a bomb. It is interesting to note that a policy that randomly explores by selecting each action with equal probability is highly unlikely to reach the chest. In particular, the probability such a policy reaches that cell in any given episode is $(1/2)^N$. Hence, the expected number of episodes before observing the chest's content is 2^N. Even for a moderate value of $N = 50$, this is over a quintillion episodes.

Let us now discuss the agent's beliefs, or state of knowledge, about the MDP $\mathcal{M}$, prior to the first episode. The agent knows everything about $\mathcal{M}$ except:

- *Action associations. At each state, the agent does not know which action index is associated with "right" or "left", and assigns equal probability to either association. These associations are independent across states.*

- *Reward. The agent does not know whether the chest contains treasure or a bomb and assigns equal probability to each of these possibilities.*

Before learning action associations and rewards, the distribution over optimal value at the initial state is given by $\mathbb{P}(V_{\mathcal{M}}^*(s_0) = 0.99) = \mathbb{P}(V_{\mathcal{M}}^*(s_0) = 0) = 1/2$. Because the MDP is deterministic, when an agent transitions from any state, she learns the action associations for that state, and when the agent selects the "right" action at the lower-right-most state, she learns whether there is treasure or a bomb.

Note that the reinforcement learning problem presented in this example is easy to address. In particular, it is straightforward to show that the minimal expected time to learn an optimal policy is achieved by an agent who chooses the "right" action whenever she knows which action that is, and otherwise, applies a random action, until she discovers the content of the chest, at which point she knows an optimal policy. This algorithm identifies an optimal policy within N episodes, since in each episode, the agent learns how to move right from at least one additional cell along the diagonal. Further, the expected learning time is $(N+1)/2$ episodes, since whenever at a state that has not previously been visited, the agent takes the wrong action with probability $1/2$. Unfortunately, this algorithm is specialized to Example 1 and does not extend to other reinforcement learning problems. For our purposes, this example will serve as a sanity check and context for illustrating flaws and features of algorithms designed for the general reinforcement learning problem.

To facilitate our discussion, it is useful to define a couple of concepts. The first is that of an *optimal state-action value function*, defined by $Q^*_{\mathcal{M}}(s, a) = \mathbb{E}_{\mathcal{M}}[r + V^*_{\mathcal{M}}(s')]$, where r and s' represent the reward and transition following application of action a in state s. Second, for any $Q : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, the *greedy policy* with respect to Q selects an action that maximizes Q, sampling randomly among alternatives if there are multiple:

$$(4.1) \qquad a \sim \texttt{unif}\left(\operatorname*{argmax}_{\alpha \in \mathcal{A}} Q(s, \alpha)\right).$$

Note that the greedy policy with respect to $Q^*_{\mathcal{M}}$ is optimal for the MDP $\mathcal{M}$. This policy depends on the random MDP $\mathcal{M}$, and therefore can not be applied in the process of learning.

The first reinforcement learning algorithm we consider is **pure-exploitation** and aims to maximize expected reward in the current episode, ignoring benefits of active exploration. This algorithm estimates a "best guess" MDP $\hat{\mathcal{M}}_L$ based upon the data it has gathered up until episode L. To offer a representative approach, we will take $\hat{\mathcal{M}}_L$ to be the MDP with rewards and transition probabilities given by their expectations conditioned on the data. The pure-exploitation algorithm then follows the policy that would be greedy with respect to $\hat{Q}_L = Q^*_{\hat{\mathcal{M}}_L}$ during episode L. While this algorithm is applicable to any reinforcement learning problem, its behavior in Example 1 reveals severe inefficiencies. Note that the algorithm is indifferent about finding the chest, since the expected reward associated with that is 0. Further, since moving toward the chest incurs cost, the algorithm avoids that, and therefore never visits the chest. As such, the algorithm is unlikely to ever learn an optimal policy.

Dithering approaches explore by selecting actions that randomly perturb what a pure-exploitation algorithm would do. As an example, one form of dithering, known as *Boltzmann exploration* selects actions according to

$$(4.2) \qquad a_t \sim \texttt{multinomial}\left(\frac{\exp\left(\hat{Q}(s_t, \cdot)/\eta\right)}{\sum_{a \in \mathcal{A}} \exp\left(\hat{Q}(s_t, a)/\eta\right)}\right).$$

Here, η represents a "temperature" parameter. As η approaches zero, actions become the same as those that would be selected by a pure-exploitation algorithm. As η increases, the selection becomes noisier, eventually converging to a uniform distribution over actions. In Example 1, a dithering algorithm is biased against moving toward the chest because of the

associated cost. Only the random perturbations can lead the agent to the chest. As such, the expected learning time is $\Theta(2^N)$.[1]

It is well known that dithering can be highly inefficient, even for bandit learning. A key shortcoming is that dithering algorithms do not write-off bad actions. In particular, even when observations make clear that a particular action is not worthwhile, dithering approaches can sample that action. Despite this understanding, dithering is the most widely used exploration method in reinforcement learning. The primary reason for this has been lack of computationally efficient approaches that adequately address the complex problems that arise in practical contexts. This paper aims to fill that need.

Bandit learning can be thought of as a special case of reinforcement learning for which actions bear no delayed consequences. The bandit learning literature offers sophisticated methods that overcome shortcomings of dithering. Such methods write-off bad actions, only selecting an action when it is expected to generate desirable reward or yield useful information or both. A naive way of applying such an algorithm to a reinforcement learning problem involves selecting an action a_t only if the expected value $\hat{Q}_L(s_t, a_t)$ is large or the observed reward and/or transition are expected to provide useful information. We call this approach **myopic exploration**, since it incentivizes exploration over a single timestep. However, applying this approach to Example 1 would once again avoid moving toward the chest as soon as it had learned the action associations in the initial state. This is because there is a cost to moving right, but once the action associations are learned, there is no immediate benefit to applying the "right" action. As such, myopic exploration is unlikely to ever learn an optimal policy.

Myopic exploration does not adequately address reinforcement learning because, in reinforcement learning, there is an additional motivation that should not be overlooked: an action can be desirable even if expected to yield no value or immediate information if the action may place the agent in a state that leads to subsequent learning opportunities. This is the essence of *deep exploration*; the agent needs to consider how actions influence downstream learning opportunities. Viewed in another way, when considering how to explore, the agent should probe *deep* in his decision tree.

Optimism serves as another guiding principle in much of the bandit learning literature and can provide a basis for deep exploration as well. In Example 1, if the agent takes most optimistic plausible view, it would assume that the chest offers treasure rather than a bomb, so long as this hypothesis has not been invalidated. In each Lth episode, the agent follows a greedy policy with respect to a value function Q_L that assigns to each state-action pair the maximal expected value under this assumption. When at a cell along the diagonal of the grid, this policy selects the "right" action whenever the agent knows which that is. Hence, this optimistic algorithm learns the optimal policy within N episodes.

The optimistic algorithm attains its strong performance in Example 1 through carrying out deep exploration. In particular, by assuming treasure rather than a bomb, the agent is incentivized to move right whenever it can, since that is the only way to obtain the posited treasure. This exploration strategy is deep since the agent does not seek only immediate information but also a learning opportunity that will only arise after consecutively moving right over multiple time periods.

---

1. A similar observation holds true for another popular dithering strategy: ϵ-greedy. This approach selects random actions according to probability ϵ and greedy (pure-exploitation) otherwise.

There are reasonably effective optimistic algorithms that apply to reinforcement learning problems with small (tractably enumerated) state and action spaces. However, the design of such algorithms that adequately address reinforcement learning problems of practical scale in a computationally tractable manner remains a challenge.

An alternative approach studied in the bandit learning literature involves randomly sampled instead of optimistic estimates. A focus of this paper is to extend this approach – known as Thompson sampling – to accommodate deep exploration in complex reinforcement learning problems. Applied to Example 1, this **randomized** approach would sample before each episode a random estimate $\tilde{Q}_L$ from the agent's posterior distribution over $Q^*_{\mathcal{M}}$, conditioned on observations made over previous episodes, or an approximation of this posterior distribution. Before the agent's first visit to the chest, she assigns equal probability to treasure and a bomb, and therefore, the sample $\tilde{Q}_L$ has an equal chance of being optimistic or pessimistic. The agent selects actions according to the greedy policy with respect to $\tilde{Q}_L$ and therefore on average explores over half of the episodes in a manner similar to an optimistic algorithm. As such, the randomized algorithm can expect to learn the optimal policy within $2N$ episodes.

As applied to Example 1, there is no benefit to using a randomized rather than optimistic approach. However, in the face of in complex reinforcement learning problems, the randomized approach can lead to computationally tractable algorithms that carry out deep exploration where the optimistic approach does not.

Table 1 summarizes our discussion of learning times of various exploration methods applied to Example 1. The minimal time required to learn an optimal policy, which is achieved by an agent who moves right whenever she knows how to, is $\Theta(N)$ episodes. The pure-exploitation algorithm avoids *any* active exploration and requires $\Theta(2^N)$ episodes to learn. Dithering does not help for our problem. Though more sophisticated, myopic approaches do not carry out deep exploration, and as such, still require $\Theta(2^N)$ episodes. Optimistic and randomized approaches require only $\Theta(N)$ episodes.

exploration method	expected episodes to learn
optimal	$\Theta(N)$
pure exploitation	∞
myopic	∞
dithering	$\Theta(2^N)$
optimistic	$\Theta(N)$
randomized	$\Theta(N)$

Table 1: Expected number of episodes required to learn an optimal policy for Example 1.

5. Algorithms

The field of reinforcement learning has produced a substantial body of algorithmic ideas that serve as ingredients to mix, match, and customize in tailoring solutions to specific applications. Such ideas are well-summarized in the textbooks of Bertsekas and Tsitsiklis (1996) and Sutton and Barto (2018), among others. The aim of this paper is to contribute to this corpus a new approach to exploration based on randomized value functions, with

the intention that this additional ingredient will broadly enable computationally efficient deep exploration.

Much of the literature and most notable applications build on value function learning. This involves fitting a parameterized value function to observed data in order to estimate the optimal value function. The algorithms we present will be of this genre. As a starting point, in Section 5.2, we will describe least-squares value iteration (LSVI), which is perhaps the simplest of value function learning algorithms. In Section 5.3, we consider modifying LSVI by injecting randomness in a manner that incentivizes deep exploration. This gives rise to a new class of algorithms, which we will refer to as randomized least-squares value iteration (RLSVI), and which offer computationally tractable means to deep exploration.

LSVI plays a foundational role in the sense that most popular value function learning algorithms can be interpreted as variations designed to improve computational efficiency or robustness to mis-specification of the parameterized value function. The reinforcement learning literature presents many ideas that address such practical considerations. In Section 5.4, we will discuss how such ideas can be brought to bear in tandem with RLSVI.

5.1. Value function learning

Before diving into specific reinforcement learning algorithms, let us discuss general concepts that apply to all of them. Value function learning algorithms make use of a family $\mathcal{Q}$ of state-action value functions indexed by $\theta \in \mathbb{R}^d$. Each $\mathcal{Q}_\theta : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ identifies a state-action value function.[2] As a simple example of such a family, consider representing value functions as linear combinations of fixed features. In particular, if $\phi(s, a) \in \mathbb{R}^d$ is a vector of features designed to capture salient characteristics of the state-action pair (s, a), it is natural to consider the family of functions taking the form $\mathcal{Q}_\theta(s, a) = \theta^\top \phi(s, a)$, with $\theta \in \mathbb{R}^d$.

Algorithm 1 (`live`) provides a template for reinforcement learning algorithms we will consider. It operates over an endless sequence of episodes, accumulating observations, learning value functions, and applying actions. We use a Pythonic pseudocode, with an object-oriented division into `agent` and `environment`. We use

$$\texttt{transition} = \texttt{NamedTuple}(\texttt{old\_state}, \texttt{action}, \texttt{reward}, \texttt{new\_state}, \texttt{timestep}),$$

to describe the evolution of the system. Where convenient, we will alternatively write `transition` $= (s_t, a_t, r_t, s'_t, t)$. We highlight three key methods the `agent` must implement:

- `act` – select actions given its internal value estimates, (e.g. greedy action selection).
- `update_buffer` – incorporate observations to its memory buffer, (e.g. append to list).
- `learn_from_buffer` – update value estimate given the data in the buffer, (e.g. LSVI).

The agents that we discuss will be distinguished through their implementation of these methods, which we will now outline.

The simplest form of `act` is given by the greedy strategy `act_greedy`. An agent that uses this approach will select actions that maximize its estimated state-action value. If multiple actions attain the maximum, one is sampled uniformly from among them.[3]

---

2. We adopt notation that for all $\mathcal{Q}$, parameter $\theta_0 = \textbf{null}$ indicates $\mathcal{Q}_{\theta_0} \equiv 0$.

3. We might also consider action selection via `act_epsilon_greedy` or `act_boltzmann`, as described in Section 4. As we saw before, these dithering approaches do not perform *deep exploration* and thus can lead to exponentially slower learning.

Algorithm 1 `live`

Input: `agent`   methods `act, update_buffer, learn_from_buffer`
              `environment`   methods `reset, step`

1: **for** ℓ in $(1, 2, \ldots)$ **do**
2:     `agent.learn_from_buffer()`
3:     `transition` ← `environment.reset()`
4:     **while** `transition.new_state` is not **null do**
5:         `action` ← `agent.act(transition.new_state)`
6:         `transition` ← `environment.step(action)`
7:         `agent.update_buffer(transition)`

Similarly, the simplest form of `update_buffer` is to simply accumulate all observed data `update_buffer_queue`. Our next two sections will investigate agents that store all observed data and take greedy actions; we investigate the effects of `learn_from_buffer` and explain why training least-squares value iteration on randomly perturbed versions of the data can offer a computationally tractable means to deep exploration.

5.2. Least-squares value iteration

Given an MDP $\mathcal{M}$, one can apply the value iteration algorithm (Algorithm 2) to compute an arbitrarily close approximation to Q^*. The algorithm takes $\mathcal{M}$ and a planning horizon H as input and computes Q_H^*, the optimal value over the next H time periods of the episode as a function of the current state and action. The computation is recursive: given Q_h^*, the algorithm computes Q_{h+1}^* by taking the expected sum of immediate reward and Q_h^*, evaluated at the next state, maximized over actions. Under Assumption 1, the mapping from Q_h^* to Q_{h+1}^* is a weighted-maximum-norm contraction mapping (Bertsekas and Tsitsiklis, 1996), and as such, Q_h^* converges to Q^* at a geometric rate. Hence, for any $\mathcal{M}$ satisfying Assumption 1 and sufficiently large H, the greedy policy with respect to Q_H^* is optimal.

Algorithm 2 `vi`

Input: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \rho)$ MDP
              $H \in \mathbb{N}$ planning horizon
Output: Q_H^* optimal value function for H-period problem

1: $Q_0^* \leftarrow 0$
2: **for** h in $(0, \ldots, H-1)$ **do**
3:    $Q_{h+1}^*(s, a) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,a}(s') \left(\int r \mathcal{R}_{s,a,s'}(dr) + \max_{a' \in \mathcal{A}} Q_h^*(s', a') \right)$ $\forall s, a \in \mathcal{S} \times \mathcal{A}$
4: **return** Q_H^*

Value iteration offers an idealized approach to evaluating Q^*, given knowledge of the underlying MDP $\mathcal{M}$ and the necessary computational power. Least-squares value iteration (LSVI) adapts `vi` to an RL setting with imperfect statistical knowledge and limited computation. For a value function family $\mathcal{Q} = \{\mathcal{Q}_\theta : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}\}$, observed data $\mathcal{D} = \{(s_t, a_t, r_t, s'_t, t)\}$ and target parameters θ^- we define the empirical temporal difference

(TD) loss:

$$\mathcal{L}(\theta; \theta^-, \mathcal{D}) \coloneqq \sum_{t \in \mathcal{D}} \left(r_t + \max_{a' \in \mathcal{A}} \mathcal{Q}_{\theta^-}(s'_t, a') - \mathcal{Q}_\theta(s_t, a_t) \right)^2 . \tag{5.1}$$

Note that, if $\mathcal{Q}$ spans the true value function Q^* and the data $\mathcal{D}$ matches the distribution of $\mathcal{M}$ then the minimizer of $\mathcal{L}$ matches the solution of Algorithm 2; for more information see Sutton and Barto (2018).

Algorithm 3 (`learn_lsvi`) describes the `learn_from_buffer` method for LSVI, whcih approximates the operations carried out by value iteration. The algorithm successively minimizes the empirical temporal difference loss (5.1) plus a regularization term:

$$\mathcal{R}(\theta; \theta^p) \coloneqq \frac{v}{\lambda} \|\theta^p - \theta\|_2^2. \tag{5.2}$$

Here θ^p can be interpreted as a prior for θ and $\frac{v}{\lambda}$ determines the strength of the regularization coefficient. In a linear system these correspond to a prior belief $\theta \sim N(\theta^p, \lambda I)$ with observations $y_t = x_t \theta + z_t$ for $z_t \sim N(0, v)$. Similarly to `vi`, `learn_lsvi` computes a sequence of value functions $(\mathcal{Q}_{\theta_h} : h = 0, \dots, H)$, reflecting optimal expected rewards over an expanding horizon. However, while value iteration computes optimal values using full knowledge of the MDP, LSVI produces estimates based only on observed data. In each iteration, for each observed transition (s, a, r, s'), `learn_lsvi` regresses the sum of immediate reward r and the value estimate $\max_{a' \in \mathcal{A}} \mathcal{Q}_{\tilde{\theta}_h}(s', a')$ at the next state onto the value estimate $\mathcal{Q}_{\tilde{\theta}_{h+1}}(s, a)$ for the current state-action pair.

Algorithm 3 `learn_lsvi`

Agent:	$\mathcal{L}(\theta = \cdot\,; \theta^- = \cdot\,, \mathcal{D} = \cdot\,)$	TD error loss function
	$\mathcal{R}(\theta = \cdot\,; \theta^p = \cdot\,)$	regularization function
	`buffer`	memory buffer of observations
	`prior`	prior distribution of θ
	$H \in \mathbb{N}$	planning horizon
Updates:	$\tilde{\theta}$	agent value function estimate

1: $\tilde{\theta}_0 \leftarrow$ **null**
2: Data $\tilde{\mathcal{D}} \leftarrow$ `buffer.data()`
3: Prior parameter $\tilde{\theta}^p \leftarrow$ `prior.mean()`
4: **for** h in $(0, \dots, H-1)$ **do**
5: $\quad \left| \quad \tilde{\theta}_{h+1} \leftarrow \underset{\theta \in \mathbb{R}^D}{\text{argmin}} \left(\mathcal{L}(\theta; \tilde{\theta}_h, \tilde{\mathcal{D}}) + \mathcal{R}(\theta; \tilde{\theta}^p) \right) \right.$
6: update value function estimate $\tilde{\theta} \leftarrow \tilde{\theta}_H$

In the event that the parameterized value function is flexible enough to represent every function mapping $\mathcal{S} \times \mathcal{A}$ to $\mathbb{R}$, it is easy to see that, for any $\bar{\theta}$ and any positive λ and v, as the observed history grows to include an increasing number of transitions from each state-action pair, value functions $\mathcal{Q}_{\tilde{\theta}_H}$ produced by LSVI converge to Q_H^*. However, in practical contexts, the data set is finite and the parameterization is chosen to be less flexible in order to enable generalization. As such, $\mathcal{Q}_{\tilde{\theta}_H}$ and Q_H^* can differ greatly.

In addition to inducing generalization, a less flexible parameterization is critical for computational tractability. In particular, the compute time and memory requirements of value iteration scale linearly with the number of states, which, due to the curse of dimensionality, grows intractably large in most practical contexts. LSVI sidesteps this scaling, instead requiring compute time and memory that scale polynomially with the dimension of the parameter vector $\tilde{\theta}$, the number of historical observations, and the time required to maximize over actions at any given state.

An LSVI `agent` may also be paired with some dithering strategy for exploration, such as ϵ-greedy or Boltzmann exploration (4.2) in place of `act_greedy`. As discussed in Section 4, randomly perturbing greedy actions – or dithering – does not achieve deep exploration and so can lead to exponentially poor performance. Our next subsection introduces randomized value function estimates as an alternative.

5.3. Randomized least-squares value iteration

At a high level, the idea is to randomly sample an imagined optimal parameter $\tilde{\theta}$ according to the probability that it is optimal. This approach is inspired by Thompson sampling, an algorithm widely used in bandit learning (Thompson, 1933). In the context of a multi-armed bandit problem, Thompson sampling maintains a belief distribution over models that assign mean rewards to arms. As observations accumulate, this belief distribution evolves according to Bayes rule. When selecting an arm, the algorithm samples a model from this belief distribution and then selects the arm to which this model assigns largest mean reward.

To address a reinforcement learning problem, one could in principle apply Thompson sampling to value function learning. This would involve maintaining a belief distribution over candidates for the optimal value function. Before each episode, we would sample a function from this distribution and then apply the associated greedy policy over the course of the episode. This approach could be effective if it were practically viable, but distributions over value functions are complex to represent and exact Bayesian inference would likely prove computationally intractable.

Randomized least-squares value iteration (RLSVI) is modeled after this Thompson sampling approach and serves as a computationally tractable method for sampling value functions. RLSVI was first introduced in Wen (2014), and subsequent work has examined variations of RLSVI and their performance with both linear and nonlinear function approximation (Osband et al., 2016b,a; Osband, 2016). RLSVI does not explicitly maintain and update belief distributions and does not optimally synthesize information, as a coherent Bayesian method would. Regardless, as we will later establish through computational and mathematical analyses, RLSVI can offer an effective approach to deep exploration.

5.3.1. RANDOMIZATION VIA GAUSSIAN NOISE

We first consider a version of RLSVI that induces exploration through injecting Gaussian noise into calculations of the form carried out by LSVI. To understand the role of this noise, let us first consider a conventional linear regression problem. Suppose we wish to estimate a parameter vector $\theta \in \mathbb{R}^D$, with $N(\bar{\theta}, \lambda I)$ prior and data $\mathcal{D} = ((x_n, y_n) : n = 1, \dots, N)$, where each "feature vector" x_n is a row vector with K components and each "target value" y_n is scalar. Given the parameter vector θ and feature vector x_n, the target y_n is generated

according to $y_n = x_n\theta + w_n$, where w_n is independently drawn from $N(0, v)$. Conditioned on $\mathcal{D}$, θ is Gaussian with

$$(5.3) \quad \mathbb{E}[\theta|\mathcal{D}] = \underset{\theta \in \mathbb{R}^D}{\arg\min} \left(\frac{1}{v} \sum_{n=1}^{N} (y_n - x_n\theta)^2 + \frac{1}{\lambda} \|\overline{\theta} - \theta\|^2 \right) = \left(\frac{1}{v} X^\top X + \frac{1}{\lambda} I \right)^{-1} \left(\frac{1}{v} X^\top y + \frac{1}{\lambda} \overline{\theta} \right)$$

and

$$\mathrm{Cov}[\theta|\mathcal{D}] = \left(\frac{1}{v} X^\top X + \frac{1}{\lambda} I \right)^{-1},$$

where $X \in \mathbb{R}^{N \times D}$ is a matrix whose nth row is x_n and $y \in \mathbb{R}^n$ is a vector whose nth component is y_n.

One way of generating a random sample $\tilde{\theta}$ with the same conditional distribution as θ is simply to sample from $\tilde{\theta} \sim N(\mathbb{E}[\theta|\mathcal{D}], \mathrm{Cov}[\theta|\mathcal{D}])$. An alternative construction is given by

$$(5.4) \quad \tilde{\theta} \leftarrow \underset{\theta \in \mathbb{R}^D}{\arg\min} \left(\frac{1}{v} \sum_{n=1}^{N} (y_n + z_n - x_n\theta)^2 + \frac{1}{\lambda} \|\hat{\theta} - \theta\|^2 \right) = \left(\frac{1}{v} X^\top X + \frac{1}{\lambda} I \right)^{-1} \left(\frac{1}{v} X^\top (y+z) + \frac{1}{\lambda} \hat{\theta} \right),$$

where $\hat{\theta} \sim N(\overline{\theta}, \lambda I)$ and $z_n \sim N(0, v)$ are sampled independently. To see why this $\tilde{\theta}$ takes on the same distribution, first note that $\tilde{\theta}$ is Gaussian, since it is affine in $\overline{\theta}$ and z. Further, $\tilde{\theta}$ exhibits the appropriate mean and covariance matrix, since

$$\mathbb{E}[\tilde{\theta}|\mathcal{D}] = \left(\frac{1}{v} X^\top X + \frac{1}{\lambda} I \right)^{-1} \left(\frac{1}{v} X^\top (y + \mathbb{E}[z|\mathcal{D}]) + \frac{1}{\lambda} \mathbb{E}[\hat{\theta}|\mathcal{D}] \right) = \mathbb{E}[\theta|\mathcal{D}],$$

and

$$
\begin{aligned}
\mathrm{Cov}[\tilde{\theta}|\mathcal{D}] &= \left(\frac{1}{v} X^\top X + \frac{1}{\lambda} I \right)^{-1} \left(\frac{1}{v^2} X^\top \mathbb{E}[zz^\top|\mathcal{D}] X + \frac{1}{\lambda^2} \mathbb{E}[\hat{\theta}\hat{\theta}^\top|\mathcal{D}] \right) \left(\frac{1}{v} X^\top X + \frac{1}{\lambda} I \right)^{-1} \\
&= \left(\frac{1}{v} X^\top X + \frac{1}{\lambda} I \right)^{-1} \left(\frac{1}{v} X^\top X + \frac{1}{\lambda} I \right) \left(\frac{1}{v} X^\top X + \frac{1}{\lambda} I \right)^{-1} \\
&= \mathrm{Cov}[\theta|\mathcal{D}].
\end{aligned}
$$

Equation (5.4) is signficant, since it allows us to understand Bayesian linear regression through a purely computational perspective. For the linear setting, we see that training a least-squares solution on perturbed versions of the data is equivalent to conjugate Bayesian posterior. This suggests that, in order to generate approximate posterior samples for Q^* we can replace the least-square computation of Algorithm 3 with an alternative value iteration that trains on randomly perturbed versions of the data. We call this algorithm *randomized least-squares value iteration*, which we now outline.

Algorithm 4 `learn_rlsvi`

Agent:	$\mathcal{L}(\theta=\cdot\,;\theta^-=\cdot\,,\mathcal{D}=\cdot\,)$	TD error loss function
	$\mathcal{R}(\theta=\cdot\,;\theta^p=\cdot\,)$	regularization function
	`buffer`	memory buffer of observations
	`prior`	prior distribution of parameters
	$H \in \mathbb{N}$	planning horizon
Updates:	$\tilde{\theta}$	agent value function estimate

1: $\tilde{\theta}_0 \leftarrow$ **null**
2: Data $\tilde{\mathcal{D}} \leftarrow$ `buffer.sample_perturbed_data()`
3: Prior parameter $\tilde{\theta}^p \leftarrow$ `prior.sample()`
4: **for** h in $(0,\ldots,H-1)$ **do**
5: $\quad\Big|\quad \tilde{\theta}_{h+1} \leftarrow \underset{\theta\in\mathbb{R}^D}{\operatorname{argmin}}\big(\mathcal{L}(\theta;\tilde{\theta}_h,\tilde{\mathcal{D}}) + \mathcal{R}(\theta;\tilde{\theta}^p)\big)$
6: update value function estimate $\tilde{\theta} \leftarrow \tilde{\theta}_H$

Note that `learn_rlsvi` is identical to `learn_lsvi` except that the optimization happens over randomized versions of the underlying data and prior. For correspondence with the Gaussian derivation above we would implement:

$$(5.5) \quad \texttt{buffer.sample\_perturbed\_data()} := \big[(s_t,a_t,r_t+z_t,s'_t,t) \ \forall t \in \texttt{buffer}, \ z_t \sim N(0,v)\big].$$

We call this version of RLSVI with additive Gaussian noise `learn_grlsvi`, indicating the specific choice of data randomization and prior $\theta \sim N(0,\lambda I)$. In Section 6 we prove that this method recovers a polynomial regret bound when used with linear value functions and tabular representation. This is significant, because LSVI with any dithering action selection scheme can not recover such a bound (Section 4). Before we jump to analysis, we provide some intuition for how this simple modification can lead to deep exploration.

5.3.2. HOW DOES RLSVI DRIVE DEEP EXPLORATION?

To understand the role of injected noise and how this gives rise to deep exploration, let us discuss a simple example, involving an MDP $\mathcal{M}$ with four states $\mathcal{S} = \{1,2,3,4\}$ and two actions $\mathcal{A} = \{up,down\}$. Let $\mathcal{H}$ be a list of all transitions we have observed, and partition this into sublists $\mathcal{H}_{s,a} = ((\tilde{s},\tilde{a},r,s') \in \mathcal{H} : (\tilde{s},\tilde{a}) = (s,a))$, each containing transitions from a distinct state-action pair. Suppose that $|\mathcal{H}_{(4,down)}| = 1$, while for each state-action pair $(s,a) \neq (4,down)$, $|\mathcal{H}_{s,a}|$ is virtually infinite. Hence, we are highly uncertain about the expected immediate rewards and transition probabilities at $(4,down)$ but can infer these quantities with extreme precision for every other state-action pair.

Given our uncertainty about $\mathcal{M}$, Q_H^* for each planning horizon H is a random variable. Figure 2 illustrates our uncertainty in these values. Each larger triangle represents a pair (s,h), where h is the horizon index. Note that these triangles represent possible future states, and h represents the number of periods between a visit to the state and the end of the planning horizon. Each of these larger triangles is divided into two smaller ones, associated with up and $down$ actions. The dotted lines indicate plausible transitions, except at $(4,down)$, where we are highly uncertain and any transition is plausible. The shade of

18

each smaller triangle represents our degree of uncertainty in the value of $Q_h^*(s,a)$. To be more concrete, take our measure of uncertainty to be the variance of $Q_h^*(s,a)$.
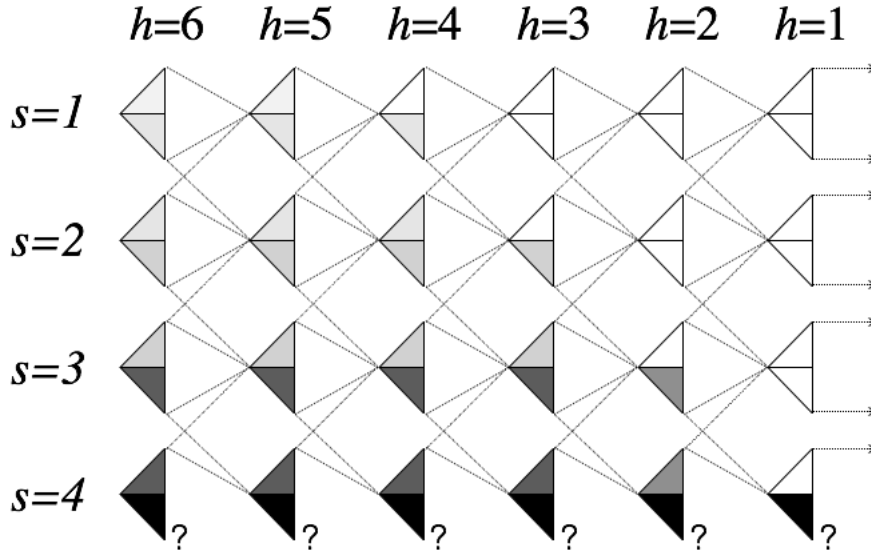

Figure 2: Illustration of how `learn_grlsvi` achieves deep exploration.

For the case of $h = 1$, only immediate rewards influence Q_1^*, and as such we are only uncertain about $Q_1^*(4, down)$. Stepping back to $h = 2$, in addition to being highly uncertain about $Q_2^*(4, down)$, we are somewhat uncertain about $Q_2^*(4, up)$ and $Q_2^*(3, down)$, since these pairs can transition to $(4, down)$ and be exposed to the uncertainty associated with that state-action pair. We are not as uncertain about $Q_2^*(4, up)$ and $Q_2^*(3, down)$ as we are about $Q_2^*(4, down)$ because from $(4, up)$ and $(3, down)$, there is reasonable chance that we will never see $(4, down)$. Continuing to work our way leftward in the diagram, it is easy to visualize how uncertainty propagates as h increases.

Let us now turn our attention to the variance of samples $\mathcal{Q}_{\tilde{\theta}_H}(s,a)$ generated by `learn_grlsvi`, which, for reasons we will explain, tend to grow and shrink with the variance of $Q_H^*(s,a)$. To keep things simple, assume $\lambda = \infty$ and that we use an exhaustive – or "tabular" – representation of value functions. In particular, each component of the parameter vector $\theta \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ encodes the value $\mathcal{Q}_\theta(s,a)$ of a single state-action pair. This parameterized value function can represent any mapping from $\mathcal{S} \times \mathcal{A}$ to $\mathbb{R}$.

Under our simplifying assumptions, it is easy to show that

$$\mathcal{Q}_{\tilde{\theta}_{h+1}}(s,a) = \frac{1}{|\tilde{\mathcal{H}}_{s,a}|} \sum_{(\tilde{s},\tilde{a},r,s',z) \in \tilde{\mathcal{H}}_{s,a}} \left(r + \max_{a' \in \mathcal{A}} \mathcal{Q}_{\tilde{\theta}_h}(s',a') + z \right).$$

The right-hand-side is an average of target values. Recall that, for any $(s,a) \neq (4, down)$, $|\tilde{\mathcal{H}}_{s,a}|$ is so large that any sample average is extremely accurate, and therefore, $\mathcal{Q}_{\tilde{\theta}_{h+1}}(s,a)$ is essentially equal to $\mathbb{E}_{\mathcal{M}}[r_{t+1} + \max_{\alpha \in \mathcal{A}} \mathcal{Q}_{\tilde{\theta}_h}(s_{t+1}, \alpha)|s_t = s, a_t = a]$. For the distinguished case of $(4, down)$, $|\tilde{\mathcal{H}}_{4,down}| = 1$, and the average target value may therefore differ substantially from its expectation $\mathbb{E}[r + \max_{a' \in \mathcal{A}} \mathcal{Q}_{\tilde{\theta}_h}(s',a')|\tilde{\theta}_h, \mathcal{M}]$. Notably, the noise term z does not

average out as it does for other state-action pairs and should contribute variance v to the sample $\mathcal{Q}_{\tilde{\theta}_{h+1}}(4, down)$.

Based on this observation, for the case of $h = 1$, for $(s, a) \neq (4, down)$, $\mathcal{Q}_{\tilde{\theta}_1}(s, a)$ is virtually equal to Q_1^*, while for $\mathcal{Q}_{\tilde{\theta}_1}(4, down)$ exhibits variance of at least v. For $h = 2$, $\mathcal{Q}_{\tilde{\theta}_2}(4, down)$ again exhibits variance of at least v, but unlike the case of $h = 1$, $\mathcal{Q}_{\tilde{\theta}_2}(4, up)$ and $\mathcal{Q}_{\tilde{\theta}_2}(3, down)$ also exhibit non-negligible variance since these pairs can transition to $(4, down)$ and therefore depend on the noise-corrupted realization of $\mathcal{Q}_{\tilde{\theta}_1}(4, down)$. Working leftward through Figure 2, we can see that noise propagates and influences value estimates in a manner captured by the shading in the figure. Hence, samples $\mathcal{Q}_{\tilde{\theta}_h}(s, a)$ exhibit high variance where the variance of $Q_h^*(s, a)$ is large.

This relationship drives deep exploration. In particular, a high variance sample $\mathcal{Q}_{\tilde{\theta}_H}(s, a)$ will be overly optimistic in some episodes. Over such episodes, the agent will be incentivized to try the associated action. This is appropriate because the agent is uncertain about the optimal value $Q_H^*(s, a)$ over the planning horizon. Note that this incentive is not only driven by uncertainty concerning the immediate reward and transition. As illustrated in Figure 2, uncertainty propagates to offer incentives for the agent to pursue information even if it will require multiple time periods to arrive at an informative observation. This is the essence of deep exploration.

It is worth commenting on a couple subtle properties of `learn_grlsvi`. First, given θ_h and $\mathcal{H}$, θ_{h+1} is sampled from a Gaussian distribution. However, given the inputs to `learn_grslvi`, the output $\tilde{\theta}$ is not Gaussian. This is because at each step θ_{h+1} depends nonlinearly on θ_h due to the maximization over actions in the TD loss (5.1). Second, it is important that `learn_grlsvi` uses the same noise samples z in across iterations of the for loop in line 5. To see why, suppose `learn_grlsvi` used independent noise samples in each iteration. Then, when applied to the example of Figure 2, in some iterations, we would be optimistic about the reward at $(4, down)$, while in other iterations, we would be pessimistic about that. Now consider a sample $\mathcal{Q}_{\tilde{\theta}_H}(1, up)$ for large H. This sample would be perturbed by a combination of optimistic and pessimistic noise terms influencing assessments at $(4, down)$ to the right. The resulting averaging effect could erode the chances that $\mathcal{Q}_{\tilde{\theta}_H}(1, up)$ is optimistic.

5.3.3. RANDOMIZATION VIA STATISTICAL BOOTSTRAP

With `learn_grlsvi`, the Gaussian distribution of noise serves as a coarse model of errors between targets $r + \max_{a' \in \mathcal{A}} \mathcal{Q}_{\tilde{\theta}_h}(s', a')$ and expectations $\mathbb{E}[r + \max_{a' \in \mathcal{A}} \mathcal{Q}_{\tilde{\theta}_h}(s', a') | \tilde{\theta}_h, \mathcal{M}]$. The statistical bootstrap[4] offers an alternative approach to randomization which may often more accurately capture characteristics of the generating process. In its classic form, the bootstrap takes a dataset $\mathcal{D}$ of size N and generates a sampled dataset $\tilde{\mathcal{D}}$ also of size N drawn uniformly with replacement from $\mathcal{D}$ (Efron and Tibshirani, 1994). The bootstrap serves as a form of data-based simulation and, in certain cases, recovers strong convergence guarantees (Bickel and Freedman, 1981; Fushiki, 2005).

---

4. We are overloading the term *bootstrap* here. In reinforcement learning, bootstrapping is commonly used to refer to the calculation of a state-action value estimate based on value estimates at states to which the agent may transition. Here, we refer to the *statistical* bootstrap of data-based simulation. The most common form of statistical bootstrap uses the sample data as an approximation to its generating distribution (Efron, 1982).

learn_brlsvi is a version of RLSVI that makes use of the bootstrap in place of additive Gaussian noise. This algorithm follows learn_rlsvi (Algorithm 4) and implements

(5.6)   buffer.sample_perturbed_data():=bootstrap_sample(buffer.data()).

Bootstrap sampling for value function randomization may present several benefits over additive Gaussian noise. First, most bootstrap resampling schemes do not require a 'noise variance' as input, which simplifies the algorithm from a user perspective. Related to this point, the bootstrap can effectively induce a state-dependent and heteroskedastic randomization which may be more appropriate in complex environments. More generally, we can consider bootstrapped RLSVI as a non-parameteric randomization for the value function estimate and this opens a wide range of potential bootstrap variants and prior mechanisms that could be employed with RLSVI (Osband and Van Roy, 2015).

Eckles and Kaptein (2019) were the first to propose using bootstrap samples as an approximation to the posterior samples used in the Thompson sampling algorithm. Unfortunately, bootstrapping does not provide meaningful uncertainty estimates in early periods. If applied without modification, the algorithms in Eckles and Kaptein (2019) incur regret that scales linearly in the time horizon. Value function estimates in learn_brlsvi are instead randomized not just through bootstrap sampling, but through regularizing toward a random prior sample. The effect of the random prior sample vanishes as diverse data is collected, but it is critical to driving exploration in early periods.

5.4. Practical variants of RLSVI

In this section, we will present variants of RLSVI designed to address the important practical considerations of computational efficiency and robustness to mis-specification of the parameterized value function. There are many ideas in the reinforcement learning literature that can be brought to bear for these purposes, and we will by no means cover an exhaustive list. Rather, we will present a mix of ideas that lead to a particular algorithm that effectively addresses a broad range of complex problems. This algorithm also serves to illustrate the many degrees of freedom in mixing and matching ingredients from the reinforcement learning literature when randomized value functions are part of the recipe.

5.4.1. Finite buffer experience replay

The use of a buffer of past observations to fit a value function is sometimes referred to as *experience replay*. The algorithms we have presented so far use an infinite buffer and thus require memory and compute time that grow linearly in the number of observations. For complex problems that require substantial learning times, such a requirement becomes onerous. To overcome this, we can restrict the buffer to some finite size, treating it as a FIFO queue.

Computational requirements aside, there can be other substantial benefits to using a finite buffer. In particular, the agent may learn to make more effective decisions within fewer episodes (Adam et al., 2012). This is likely due to model mis-specification. In particular, if Q^* can not be represented by $\mathcal{Q}_\theta$ for any θ, it is helpful to restrict attention to the most relevant data when regressing, as this focusses on minimizing errors at relevant states and actions. Restricting the buffer to recent observations may serve as a reasonable heuristic

here. Recent work has also demonstrated benefit from more sophisticated prioritization of data for storage in a finite buffer (Schaul et al., 2015).

5.4.2. DISCOUNTED TD AND INCREMENTAL LEARNING

Both LSVI and RLSVI take a planning horizon $H \in \mathbb{N}$ as an argument. These approaches can be wasteful in that they compute separate estimates $\tilde{\theta}_h$ for each $h = 1, .., H$. Further, these algorithms are "batch," in the sense that they require computation over all observed data at the start of each episode; this leads to computational costs that grow with time. In this subsection we introduce a discounted formulation that admits an incremental computational approach.

Let $\gamma \in (0, 1)$ be a discount factor that induces a time preference over future rewards. A discount γ approximates an effective planning horizon $H \simeq \frac{1}{1-\gamma}$, but affords a solution to the discounted Bellman equation $Q^*_\gamma(s, a) = \mathbb{E}_\mathcal{M}\left[R(s, a) + \gamma \max_{a'} Q^*_\gamma(s', a')\right]$ (Blackwell, 1965). Inspired by this relationship we define the γ-discounted empirical TD loss:

$$(5.7) \qquad \mathcal{L}\gamma(\theta; \theta^-, \mathcal{D}) := \sum_{t \in \mathcal{D}} \left(r_t + \gamma \max_{a' \in \mathcal{A}} \mathcal{Q}_{\theta^-}(s'_t, a') - \mathcal{Q}_\theta(s_t, a_t)\right)^2.$$

Algorithm 5 (`learn_online_lsvi`) presents an incremental variant of LSVI. Rather than recompute $\tilde{\theta}$ from scratch each episode, `learn_online_lsvi` updates its previous estimate by gradient descent over a subset of the data. This algorithm is a form of temporal difference learning (Sutton, 1988) and, at a high level, this approach broadly describes famous approaches such as TD-gammon (Tesauro, 1995) and DQN (Mnih et al., 2015). For more background and discussion of this family of algorithms we refer to Sutton and Barto (2018).

Algorithm 5 `learn_online_lsvi`

Agent:	$\tilde{\theta}$	parameter estimate
	$\tilde{\theta}^p$	prior mean of parameter estimate
	$\mathcal{L}_\gamma(\theta = \cdot; \theta^- = \cdot, \mathcal{D} = \cdot)$	TD error loss function
	$\mathcal{R}(\theta = \cdot; \theta^p = \cdot)$	regularization function
	`buffer`	memory buffer of observations
	α	learning rate
Updates:	$\tilde{\theta}$	agent value function estimate

1: Data $\tilde{\mathcal{D}} \leftarrow$ `buffer.sample_minibatch()`
2: $\delta \leftarrow$ `buffer.minibatch_size / buffer.size`
3: $\tilde{\theta} \leftarrow \tilde{\theta} - \alpha \nabla_{\theta|\theta=\tilde{\theta}}\left(\mathcal{L}_\gamma(\theta; \tilde{\theta}, \tilde{\mathcal{D}}) + \delta \mathcal{R}(\theta; \tilde{\theta}^p)\right)$

5.4.3. RANDOMIZATION VIA ENSEMBLE SAMPLING

Algorithm 5 (`learn_online_lsvi`) presents an incremental version of LSVI. However, RLSVI in its purest form calls for an estimate trained on randomly perturbed data at the beginning of each episode $l = 1, 2, ..$; this is not immediately amenable to such an incremental algorithm. Instead, our solution approximates the effects of RLSVI in an online algorithm via *ensemble sampling*.

Ensemble sampling approximates the distribution induced by RLSVI through an ensemble of $K \in \mathbb{N}$ estimates trained in parallel $(\tilde{\theta}_1, .., \tilde{\theta}_K)$ (Osband and Van Roy, 2015; Lu and Van Roy, 2017). At the start of any episode l, we can approximate the distribution of $\tilde{\theta}$ under RLSVI via a sample of $\tilde{\theta}_k$ for $k \sim \mathrm{Unif}(1, .., K)$. Figure 3 presents an illustration of such a system.
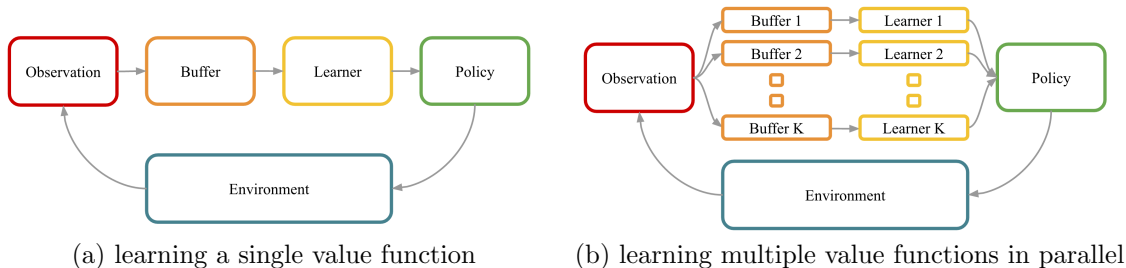

(a) learning a single value function (b) learning multiple value functions in parallel

Figure 3: RLSVI via ensemble sampling, each member produced by LSVI on perturbed data.

One method to maintain K estimates in parallel is to implement an ensemble memory buffer `ensemble_buffer`. An ensemble buffer should function similarly to `buffer`, but maintain K distinct perturbations of the observed data $\tilde{\mathcal{D}}_k$, each associated with their appropriate ensemble value estimate $\tilde{\theta}_k$. We use a Pythonic notation that `ensemble_buffer[k]` is a `buffer` for each `k = 1, .., K`; although in a practical implementation it will be sensible to share appropriate parts of the memory requirements. For an online variant of `learn_grlsvi` (additive Gaussian noise) we can store K distinct samples of additive noise, as described in Algorithm 6. Similarly, for an online approximation to bootstrap sampling we might use "double-or-nothing" sampling per Algorithm 7 (Owen and Eckles, 2012).

Algorithm 8 (`learn_ensemble_rlsvi`) presents the ensemble RLSVI algorithm. An agent that employs `learn_ensemble_rlsvi` maintains ensemble parameter estimates $\tilde{\theta}_1, .., \tilde{\theta}_K$ with random prior samples $\tilde{\theta}_1^p, .., \tilde{\theta}_K^p$. The regulatization effects of $\mathcal{R}(\theta; \tilde{\theta}_k^p)$ can play an important role in exploration. Although we have suggested a particular form in (5.2), we would consider alternative approaches based on prior observations as studied in *off-policy learning* (Precup et al., 2001) and *transfer learning* (Taylor and Stone, 2009).

We note that, when used off-policy, minimizing the TD loss may lead to unstable learning and even cause the value function estimate to diverge (Tsitsiklis and Van Roy, 1997). This instability can be exacerbated by algorithms such as `learn_ensemble_rlsvi`, which may lead to value estimates computed from data more off-policy than a single value estimate. In this context, it may be beneficial to replace the naive TD loss with an alternative designed for off-policy learning (Sutton et al., 2009; Munos et al., 2016).

Algorithm 6 `ensemble_buffer.update_gaussian_noise`$(\cdot)$

Input:	`transition`	(s_t, a_t, r_t, s'_t, t)
Agent:	v	noise variance
Updates:	`ensemble_buffer`	replay buffer of K-parallel perturbed data

1: **for** k in $(1, \ldots, K)$ **do**
2: $\quad$ `ensemble_buffer[k].enqueue`$((s_t, a_t, r_t + z_t^k, s'_t, t))$ where $z_t^k \sim N(0, v)$

Algorithm 7 `ensemble_buffer.update_bootstrap(·)`

Input: `transition` (s_t, a_t, r_t, s'_t, t)

Updates: `ensemble_buffer`   replay buffer of K-parallel perturbed data

1: **for** k in $(1, \ldots, K)$ **do**
2:     **if** $m_t^k \sim \mathrm{Unif}(\{0,1\}) = 1$ **then**
3:       `ensemble_buffer[k].enqueue(`(s_t, a_t, r_t, s'_t, t)`)`

Algorithm 8 `learn_ensemble_rlsvi`

Agent: $\tilde{\theta}_1, .., \tilde{\theta}_K$ ensemble parameter estimates
              $\tilde{\theta}_1^p, .., \tilde{\theta}_K^p$ prior samples of parameter estimates
              $\mathcal{L}_\gamma(\theta = \cdot\, ; \theta^- = \cdot, \mathcal{D} = \cdot)$ TD error loss function
              $\mathcal{R}(\theta = \cdot\, ; \theta^p = \cdot)$ regularization function
              `ensemble_buffer`   replay buffer of K-parallel perturbed data
              α Learning rate

Updates: $\tilde{\theta}$ agent value function estimate

1: **for** k in $(1, \ldots, K)$ **do**
2:     Data $\tilde{\mathcal{D}}_k \leftarrow$ `ensemble_buffer[k].sample_minibatch()`
3:     $\delta \leftarrow$ `buffer.minibatch_size` / `buffer.size`
4:     $\tilde{\theta}_k \leftarrow \tilde{\theta}_k - \alpha \nabla_{\theta | \theta = \tilde{\theta}_k} \left(\mathcal{L}_\gamma(\theta; \tilde{\theta}_k, \tilde{\mathcal{D}}_k) + \mathcal{R}(\theta; \tilde{\theta}_k^p) \right)$
5: update $\tilde{\theta} \leftarrow \tilde{\theta}_j$ for $j \sim \mathrm{Unif}(1, .., K)$

---

6. Regret bound

This section provides a regret analysis of RLSVI in a particularly simple type of decision problem (Section 3). We consider an RLSVI `agent` with an infinite buffer, greedy actions and `learn_grlsvi` (Algorithm 4 with additive Gaussian noise (5.5)) and a tabular representation.[5] The bound we establish applies to a tabular time-inhomogeneous MDP with transition kernel drawn from a Dirichlet prior. This stylized setting provides rigorous confirmation that RLSVI is capable of performing provably efficient deep exploration in tabular environments. In addition, we hope this analysis provides a framework for establishing more general guarantees – for example those applying to RLSVI with linearly parameterized value functions. Several intermediate lemmas used in the analysis hold under much less restrictive assumptions, and could be useful beyond the setting studied here.

6.1. Formulation of a time-inhomogenous MDP

We consider a class of *finite-horizon time-inhomogeneous MDPs*. This can be formulated as a special case the paper's general formulation as follows. Assume the state space factorizes as $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_{H-1}$ where the state always advances from some state $s_t \in \mathcal{S}_t$ to $s_{t+1} \in \mathcal{S}_{t+1}$ and the process terminates with probability 1 in period H. For notational convenience, we assume each set $\mathcal{S}_0, \ldots, \mathcal{S}_{H-1}$ contains an equal number of elements. This

---

5. A tabular representation for RLSVI means that $\mathcal{Q}_\theta = \theta \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ and $\mathcal{Q}_\theta(s, a) = \theta_{s,a}$.

is stated formally in the next assumption, which is maintained for all statements in this section.

Assumption 2 (Finite-horizon time-inhomogeneous MDP)
The state space factorizes as $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_{H-1}$ where $|\mathcal{S}_0| = \cdots = |\mathcal{S}_{H-1}| < \infty$. For any MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \rho)$,

$$\sum_{s' \in \mathcal{S}_{t+1}} \mathcal{P}_{s,a}(s') = 1 \qquad \forall t \in \{0, ..., H-2\}, \, s \in \mathcal{S}_t, \, a \in \mathcal{A},$$

and

$$\sum_{s' \in \mathcal{S}} \mathcal{P}_{s,a}(s') = 0 \qquad \forall s \in \mathcal{S}_{H-1}, \, a \in \mathcal{A}.$$

Each state $s \in \mathcal{S}_t$ can be written as a pair $s = (t, x)$ where $t \in \{0, ..., H-1\}$ and $x \in \mathcal{X} = \{1, ..., |\mathcal{S}_0|\}$. Similarly, a policy $\pi : \mathcal{S} \to \mathcal{A}$ can be viewed as a sequence $\pi = (\pi_0, ..., \pi_{H-1})$ where $\pi_t : x \mapsto \pi((t, x))$. Our notation can be specialized to this time-inhomogenous problem, writing transition probabilities as $\mathcal{P}_{t,x,a}(x') \equiv \mathcal{P}_{(t,x),a}((t+1, x'))$ and reward probabilities as $\mathcal{R}_{t,x,a,x'}(r) \equiv \mathcal{R}_{(t,x),a,(t+1,x')}(r)$. For consistency, we also use different notation for the optimal value function, writing

$$V_{\mathcal{M},t}^{\pi}(x) \equiv V_{\mathcal{M}}^{\pi}((t, x))$$

and define $V_{\mathcal{M},t}^{*}(x) := \max_{\pi} V_{\mathcal{M},t}^{\pi}(x)$. Similarly, we can define the state-action value function under the MDP at timestep $t \in \{0, ..., H-1\}$ by

$$Q_{\mathcal{M},t}^{*}(x, a) = \mathbb{E}[r_{t+1} + V_{\mathcal{M},t+1}^{*}(x_{t+1}) \mid \mathcal{M}, x_t = x, a_t = a] \qquad \forall x \in \mathcal{X}, a \in \mathcal{A}.$$

This is the expected reward accrued by taking action a in state x and proceeding optimally thereafter.

Upon choosing an action, the algorithm observes a pair $o = (x', r)$ consisting of a state transition and a reward. We will refer to this pair o as an *outcome* of the decision. Assumptions about the distribution of rewards and state-transitions can be more compactly written as an assumption about outcome distributions. We study the regret of RLSVI under the following Bayesian model for the MDP $\mathcal{M}$. This assumption is not required for some of the results in this section, and we will specify when it is needed.

Assumption 3 (Independent Dirichlet prior for outcomes)
Rewards take values in $\{0, 1\}$ and so the cardinality of the outcome space is $|\mathcal{X} \times \{0, 1\}| = 2|\mathcal{X}|$. For each, $(t, x, a) \in \{0, ..., H-2\} \times \mathcal{X} \times \mathcal{A}$, the outcome distribution is drawn from a Dirichlet prior

$$\mathcal{P}_{t,x,a}^{O}(\cdot) \sim \mathrm{Dirichlet}(\alpha_{0,t,x,a})$$

for $\alpha_{0,t,x,a} \in \mathbb{R}_+^{2|\mathcal{X}|}$ and each $\mathcal{P}_{t,x,a}^{O}$ is drawn independently across (t, x, a). Assume there is $\beta \geq 3$ such that $\mathbb{1}^T \alpha_{0,t,a,x} = \beta$ for all (t, x, a).

6.2. Bayesian regret bound

The following theorem is the main result of this section, and establishes a polynomial bound on the Bayesian regret of RLSVI.

Theorem 1 (Bayesian regret bound for RLSVI)
Consider an RLSVI `agent` *with an infinite buffer, greedy actions and* `learn_grlsvi` *with tabular representation.[6] Under Assumption 3 with $\beta \geq 3$, if this version of RLSVI is applied with planning horizon H, and parameters $v = 3H^2$, $\bar{\theta} = H\mathbb{1}$ and $v/\lambda = \beta$, then for all $L \in \mathbb{N}$,*

$$(6.1) \quad \text{BayesRegret}(\text{RLSVI}_{\bar{\theta},v,\lambda}, L) \leq 6H^2\sqrt{\beta|\mathcal{X}||\mathcal{A}|L\log_+(1+|\mathcal{X}||\mathcal{A}|HL)}\log_+\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right),$$

and

$$(6.2) \quad \text{BayesRegret}(\text{RLSVI}_{\bar{\theta},v,\lambda}, L) \leq 5\beta H^3|\mathcal{X}|||\mathcal{A}|\sqrt{\log_+(1+|\mathcal{X}||\mathcal{A}|HL)}\log_+\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right)$$
$$+ 2H^2\sqrt{6|\mathcal{X}||\mathcal{A}|L\log(|\mathcal{X}||\mathcal{A}|)}$$

where $\log_+(x) = \max\{1, \log(x)\}$.

Let us focus on the first bound given in equation (6.1). The parameter β governs the relative strength of prior mean $\bar{\theta}$ in the Q-functions sampled by RLSVI. We typically think of β as a constant, reflecting situations with weak prior knowledge of the optimal value function that does not grow with variables H, S, A, L. In this case, this regret bound is $\tilde{O}(H^2\sqrt{|\mathcal{X}||\mathcal{A}|L})$ where $\tilde{O}$ ignores poly-logarithmic factors. Note that since $|\mathcal{S}_0| = ... = |\mathcal{S}_{H-1}| = |\mathcal{X}|$ then $|\mathcal{S}| = |\mathcal{X}|H$ and for $T = LH$ denoting the number of periods,

$$\text{BayesRegret}(\text{RLSVI}_{\bar{\theta},v,\lambda}, L) = \tilde{O}(H\sqrt{|\mathcal{S}||\mathcal{A}|T}).$$

This bound reveals that RLSVI requires a number of episodes that is just linear in the number of states to reach near optimal performance. Indeed, it is possible to guarantee cumulative Bayesian regret less than $L\epsilon$ with a value of L that scales with $|\mathcal{X}|/\epsilon^2$. In general, at least order $|\mathcal{X}|^2$ samples are required to learn the transition kernel $\mathcal{P}_{t,x,a}$. Therefore, for large $|\mathcal{X}|$ we prove that RLSVI learns to make near-optimal decisions using fewer samples than would be required to learn the transition dynamics of the MDP.

It is interesting to compare this Bayesian regret bound with bounds that have been established for other tabular reinforcement learning algorithms. The results of Bartlett and Tewari (2009) and Jaksch et al. (2010) are not directly comparable to the bound established for RLSVI, as they develop bounds on minimax, rather than Bayesian regret, and study classes of MDPs satisfying recurrence assumptions, rather than episodic MDPs. However, it is worth noting that because these algorithms attempt to represent each transition probability $\mathcal{P}_{t,x,a}(x')$ accurately, applying their analysis to our problem yields a regret bound of $\tilde{O}(H^2|\mathcal{X}|\sqrt{|\mathcal{A}|L})$, which has is larger dependence on $|\mathcal{X}|$.

The second bound given in equation (6.2) reveals the dependence of regret on β more precisely. This bound is $\tilde{O}(\beta H^3|\mathcal{X}||\mathcal{A}| + H^2\sqrt{|\mathcal{X}||\mathcal{A}|L})$. The first term in this regret bound

---

6. `learn_grlsvi` is `learn_rlsvi` (Algorithm 4) with additive Gaussian reward noise of variance v (5.5).
   A tabular representation for RLSVI means that $\mathcal{Q}_\theta = \theta \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ and $\mathcal{Q}_\theta(s,a) = \theta_{s,a}$.

can roughly be thought of as a bound on the regret incurred throughout an initial phase of the algorithm, during which it gathers data that overwhelms the prior mean. When the number of episodes L is large, the dominant term is the second one, which is $\tilde{O}(H^2 \sqrt{|\mathcal{X}||\mathcal{A}|L})$ and has no dependence on β.

6.3. Stochastic Bellman operators

Any state-action value function $Q \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$ induces a value function $V(x) = \max_{a \in \mathcal{A}} Q(x, a)$ that maps each state to a real number. To simplify the analysis, it is useful to introduce nonstandard notation for the value function over outcomes $o = (r, x)$.

Definition 2 (Induced value function)
For a state-action value function $Q \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$ define the corresponding value function $V_Q \in \mathbb{R}^{2|\mathcal{X}|}$ over outcomes by $V_Q(r, x') := r + \max_{a \in \mathcal{A}} Q(x', a)$ for all $x' \in \mathcal{X}$ and $r \in \{0, 1\}$.

It is useful to also keep notation for the empirical distribution over observed outcomes. Let

$$D_{\ell-1}(t, x, a) = \{(r_{t+1}^k, x_{t+1}^k) : k < \ell, x_t^k = x, a_t^k = a\}$$

be the set of data observed up to episode ℓ when action a was chosen in (t, x), and set $n_\ell(t, x, a) = |D_{\ell-1}(t, x, a)|$ to be number of past observations of the triple (t, x, a). For ease of notation we will write y for the timestep, state, and action $y := (t, x, a)$. Denote by $\hat{P}_{\ell,y}^O(r', x')$ the empirical distribution over outcomes (r', x') in the dataset $D_{\ell-1}(y)$.

This section introduces the Bellman operator underlying the MDP $\mathcal{M}$ and a notion of a Bellman operator that underlies the recursion defining RLSVI. Due to the randomness in $\mathcal{M}$ under Assumption 3 and the Gaussian noise added by RLSVI iterations both of these can be viewed as *stochastic Bellman operators*, as applying one of these operators to a state action value function $Q \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$ generates a random state-action value function as output.

True Bellman Operator. For $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ the true Bellman operator at timestep t applied to Q is defined by

$$
\begin{aligned}
F_{\mathcal{M},t} Q(x, a) &= \mathbb{E}\left[r_{t+1} + \max_{a' \in \mathcal{A}} Q(x_{t+1}, a') \mid \mathcal{M}, x_t = x, a_t = a\right] \\
&= \mathbb{E}\left[V_Q(r_{t+1}, x_{t+1}) \mid \mathcal{M}, x_t = x, a_t = a\right] \\
&= V_Q^T \mathcal{P}_{t,x,a}^O.
\end{aligned}
$$

Applying $F_{\mathcal{M},t}$ backward in time produces a sequence of optimal state-action value functions satisfying $Q_{\mathcal{M},H}^* = 0$ and the Bellman equation $Q_{\mathcal{M},t}^* = F_{\mathcal{M},t} Q_{\mathcal{M},t+1}^*$ for $t < H$. Under Assumption 3, this can be viewed as a randomized Bellman operator due to the randomness in the MDP $\mathcal{M}$.

Under Assumption 3, the posterior transition probabilities are distributed as

$$\mathcal{P}_y^O(\cdot) | \mathcal{H}_{\ell-1} \sim \text{Dirichlet}(\alpha_{\ell,y})$$

where

(6.3) $$\alpha_{\ell,y} = \alpha_{0,y} + n_\ell(y) \hat{P}_{\ell,y}^O \in \mathbb{R}^{2|\mathcal{X}|}$$

for any triple $y = (t, x, a)$. These determine the posterior mean of $\mathcal{P}_y^O$ as a weighted linear combination of the prior and the empirical observations:

$$\mathbb{E}[\mathcal{P}_y^O \mid \mathcal{H}_{\ell-1}] = \frac{\alpha_{0,y} + n_\ell(y)\hat{P}_{\ell,y}^O}{\beta + n_\ell(y)}.$$

Bellman operator of RLSVI. In episode ℓ, we can define a notion of a Bellman operator underlying the recursion of RLSVI. Define

$$F_{\ell,t}Q(x,a) := \sigma_\ell^2(t,x,a)\left(\frac{\bar{\theta}_{t,x,a}}{\lambda} + \frac{1}{v}\left(\sum_{\substack{(r,x')\in \\ D_{\ell-1}(t,x,a)}} r + \max_{a'\in\mathcal{A}} Q(x',a')\right)\right) + w_\ell(t,x,a)$$

$$\sigma_\ell^2(t,x,a) = \left(\frac{1}{\lambda} + \frac{n_\ell(t,x,a)}{v}\right)^{-1} = \frac{v}{n_\ell(t,x,a) + v/\lambda}$$

$$w_\ell(t,x,a) \mid \mathcal{H}_{\ell-1} \sim N(0, \sigma_\ell^2(t,x,a))$$

where $w_\ell(y)/\sigma_\ell(y) \sim N(0,1)$ is drawn independently across episodes ℓ and triples $y = (t,x,a)$.

In episode ℓ RLSVI generates a sequence of state-action value functions $Q_{\ell,1}, ..., Q_{\ell,H}$ where $Q_{\ell,H} = 0 \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$ consists of all zeros and for all $t < H$, $Q_{\ell,t} = F_{\ell,t}Q_{\ell,t+1}$. RLSVI chooses actions greedily with respect to this sequence of state-action value functions. Note that because of the Gaussian sampling noise, the action $\arg\max_{a\in\mathcal{A}} Q_\ell(x,a)$ is unique with probability one for any x and ℓ. Therefore the policy applied by RLSVI in an episode is completely determined by the state-action value functions it samples.

We can also express the RLSVI Bellman update in a simple way using the empirical distribution $\hat{P}_{\ell,y}^O$ over past outcomes resulting from $y = (t,x,a)$. We have

$$\sum_{(r,x')\in D_{\ell-1}(y)}\left(r + \max_{a'\in\mathcal{A}} Q(x',a')\right) = n_\ell(y)V_Q^T\hat{P}_{\ell,y}^O.$$

Direct calculation gives the following alternate expression

$$(6.4) \qquad F_{\ell,t}Q(x,a) = \frac{(v/\lambda)\bar{\theta} + n_\ell(y)V_Q^T\hat{\mathcal{P}}_{\ell,y}^O}{(v/\lambda) + n_\ell(y)} + w_\ell(y) \qquad \forall y = (t,x,a).$$

This shows that the Bellman update of RLSVI differs from the empirical Bellman update $V_Q^T\hat{\mathcal{P}}_{\ell,y}^O$ in two ways: there is slight regularization toward the prior mean $\bar{\theta}$, and more importantly, RLSVI adds independent Gaussian noise to each update.

6.4. Optimism and regret decompositions

The next lemma forms a crucial element of the proof.

Lemma 3 (Planning Error to On Policy Bellman Error)
Let $Q_0, Q_1, Q_2, ..., Q_H \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$ be any sequence with $Q_H = 0 \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$ and take $\pi = (\pi_0, \pi_1, ..., \pi_{H-1})$

to be a policy with $\pi_t(x) \in \arg\max_{a \in \mathcal{A}} Q_t(x, a)$ for all x. Then for any MDP $\mathcal{M}$ and initial state $x \in \mathcal{X}$,

$$Q_0(x, \pi_0(x)) - V^\pi_{\mathcal{M},0}(x) = \mathbb{E}_{\mathcal{M},\pi}\left[\sum_{t=0}^{H-1}(Q_t - F_{\mathcal{M},t}Q_{t+1})(x_t, a_t) \mid x_0 = x\right].$$

Remark 4 *To interpret this lemma, consider an algorithm that generates a sequence of state-action value functions $Q_0, ..., Q_H \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$ and chooses actions greedily with respect to this sequence. We can interpret $Q_0(x, \pi_0(x))$ to be the algorithm's estimate of the value of following this greedy policy throughout the episode from a starting state x, while $V^\pi_{\mathcal{M},0}(x)$ denotes the true expected value. One can interpret $Q_t - F_{\mathcal{M}}Q_{t+1}$ as the error in Bellman's equation at stage t. The right hand side of of the equation in Lemma 3 measures Bellman error on policy, i.e. at the states and actions that the agent is expected to sample by following the policy throughout the episode. This lemma says that the prediction $Q_0(x, \pi_0(x))$ can be far from the true value function only when on policy Bellman error is large.*

Lemma 3 is a powerful tool for studying the regret of optimistic algorithms. The regret of the policy π in Lemma 3 incurred in a single episode can always be decomposed as

$$V^*_{\mathcal{M},0}(x) - V^\pi_{\mathcal{M},0}(x) = \left(\max_{a \in \mathcal{A}} Q^*_{\mathcal{M},0}(x, a) - \max_{a \in \mathcal{A}} Q_0(x, a)\right) + \left(\max_{a \in \mathcal{A}} Q_0(x, a) - V^\pi_{\mathcal{M},0}(x)\right),$$

where we have used the fact that $V^*_{\mathcal{M},0}(x) = \max_{a \in \mathcal{A}} Q^*_{\mathcal{M},0}(x, a)$. The second term in this decomposition can be rewritten using Lemma 3. In particular, for any sequence $Q_0, ..., Q_H \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$ with $Q_H = 0$ and policy $\pi = (\pi_0, ..., \pi_{H-1})$ under which actions $\pi_t(x) = \arg\max_{a \in \mathcal{A}} Q_t(x, a)$ are chosen greedily with respect these Q–functions, regret can be decomposed as follows:

$$V^*_{\mathcal{M},0}(x) - V^\pi_{\mathcal{M},0}(x) = \quad \max_{a \in \mathcal{A}} Q^*_{\mathcal{M},0}(x, a) - \max_{a \in \mathcal{A}} Q_0(x, a) \qquad \text{(pessimism of } Q_0\text{)}$$

$$(6.5) \qquad + \quad \mathbb{E}_{\mathcal{M},\pi}\left[\sum_{t=0}^{H-1}(Q_t - F_{\mathcal{M},t}Q_{t+1})(x_t, a_t) \mid x_0 = x\right] \quad \text{(on policy Bellman error)}.$$

If the function Q_0 is optimistic at an initial state x, in the sense that $\max_a Q_0(x, a) \geq \max_a Q^*_{\mathcal{M},0}(x, a)$, then regret in the episode is bounded by on policy Bellman error under $(Q_0, ..., Q_H)$.

One can apply this regret decomposition to study RLSVI by taking $(Q_0, ..., Q_H)$ to be the sequence $(Q_{\ell,0}, ..., Q_{\ell,H})$ generated by RLSVI in some episode ℓ. On policy Bellman error can be simplified further by plugging in $Q_{\ell,t} = F_{\ell,t}Q_{\ell,t+1}$. The next corollary of Lemma 3 then follows by taking expectations on both sides of equation (6.5).

Corollary 5 (Optimistic regret bounds)
For any episode $\ell \in \mathbb{N}$, if

$$(6.6) \qquad \mathbb{E}\left[\max_{a \in \mathcal{A}} Q_{\ell,0}(x^\ell_0, a)\right] \geq \mathbb{E}\left[\max_{a \in \mathcal{A}} Q^*_{\mathcal{M},0}(x^\ell_0, a)\right]$$

then

$$\mathbb{E}\left[V^*_{\mathcal{M},0}(x^\ell_0) - V^{\pi_\ell}_{\mathcal{M},0}(x^\ell_0)\right] \leq \mathbb{E}\left[\sum_{t=0}^{H}(F_{\ell,t}Q_{\ell,t+1} - F_{\mathcal{M},t}Q_{\ell,t+1})(x^\ell_t, a^\ell_t)\right].$$

Corollary 5 forms the core of our analysis. The next section establishes that (6.6) holds in every episode $\ell \in \mathbb{N}$. We then complete the proof by bounding the cumulative on policy Bellman error throughout L episodes.

6.5. Stochastic optimism

Our goal is to show equation 6.6 holds when RLSVI is applied with appropriate parameters. We will instead prove that under Assumption 3 the stronger condition that

$$\mathbb{E}\left[\max_{a\in\mathcal{A}} Q_{\ell,0}(x_0^\ell, a) \mid \mathcal{H}_{\ell-1}\right] \geq \mathbb{E}\left[\max_{a\in\mathcal{A}} Q_{\mathcal{M},0}^*(x_0^\ell, a) \mid \mathcal{H}_{\ell-1}\right]$$

holds for any history $\mathcal{H}_{\ell-1}$. By the tower property of conditional expectation, this clearly implies equation (6.6).

As highlighted in Subsection 6.3, both $Q_{\ell,0} = F_{\ell,0}\cdots F_{\ell,H-1}0$ and $Q_{\mathcal{M},0}^* = F_{\mathcal{M},0}\cdots F_{\mathcal{M},H-1}0$ are calculated through recursive backward application of stochastic Bellman operators. The distributions of $Q_{\ell,0}$ and $Q_{\mathcal{M},0}^*$ generated in this fashion is complicated and difficult to study directly. Instead, we study properties of the stochastic Bellman operators themselves. We establish a strong sense in which $F_{\ell,t}$ generates random Q-functions $F_{\ell,t}Q$ that are optimistic compared to those generated by applying $F_{\mathcal{M},t}$ to Q. We then show this optimism is preserved under recursive application of the stochastic Bellman operators, which will imply the optimism of the final iterate $Q_{\ell,0}$. This strong notion of optimism is defined below.

Definition 6 (Stochastic optimism)
A random variable X is stochastically optimistic with respect to another random variable Y, written $X \succeq_{SO} Y$, if for all convex increasing functions $u : \mathbb{R} \to \mathbb{R}$

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

This definition closely mirrors that of "second order stochastic dominance", which is widely used in decision theory (Hadar and Russell, 1969). A random payout X is second order stochastically dominant with respect to Y if (6.7) holds for all *concave* increasing function u. This means that any rational *risk-averse* agent prefers X to Y, while $X \succeq_{SO} Y$ implies that any rational *risk-loving* agent prefers X to Y. Intuitively, this requirement means that draws of X generate payouts that are larger and noisier than Y. Our goal then is to show if RLSVI is applied with appropriate parameters, it generates iterates that are larger and noisier than the true $Q - functions$.

Example 2 (Stochastic optimism in Gaussian random variables)
If $X \sim N(\mu_X, \sigma_X^2)$ and $Y \sim N(\mu_Y, \sigma_Y^2)$ then $X \succeq_{SO} Y$ if and only if $\mu_X \geq \mu_Y$ and $\sigma_X^2 \geq \sigma_Y^2$.

The following observation is key to our analysis.

Lemma 7 (Preservation of optimism under convex operations)
For any two collections $(X_1, ..., X_n)$ and $(Y_1, ..., Y_n)$ of independent random variables with $X_i \succeq_{SO} Y_i$ for each $i \in \{1, ...n\}$ and any convex increasing function $f : \mathbb{R}^n \to \mathbb{R}$,

$$f(X_1, ..., X_n) \succeq_{SO} f(Y_1..., Y_n).$$

Two special cases of Lemma 7 imply that the partial ordering of stochastic optimism is preserved under convolution and maximization. In particular, for any independent random

variables (X, Y, Z) if $X \geq_{SO} Y$ we can conclude[7] $X + Z \geq_{SO} Y + Z$. For two pairs of independent random variables (X_1, X_2) and (Y_1, Y_2) with $X_1 \geq_{SO} Y_1$ and $X_2 \geq_{SO} Y_2$,

$$\max\{X_1, X_2\} \geq_{SO} \max\{Y_1, Y_2\}.$$

This implies the following monotonicity property of the Bellman operator $F_{\ell,t}$ underlying RLSVI. This will later enable us to show that if initial iterates of RLSVI are stochastically optimistic, then this optimism is preserved under recursive application of the stochastic Bellman operators $F_{\ell,0}\cdots F_{\ell,H-1}$.

Lemma 8 (Monotonicity)
Fix two random Q functions $Q_1, Q_2 \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$. Suppose that conditioned on $\mathcal{H}_{\ell-1}$, for each $i = 1, 2$ the entries of $Q_i(x, a)$ are drawn independently across x, a, and drawn independently of the RLSVI noise terms $w_\ell(t, x, a)$. Then

$$Q_1(x, a) \mid \mathcal{H}_{\ell-1} \geq_{SO} Q_2(x, a) \mid \mathcal{H}_{\ell-1} \qquad \forall (x, a) \in \mathcal{X} \times \mathcal{A}$$

implies

$$F_{\ell,t} Q_1(x, a) \mid \mathcal{H}_{\ell-1} \geq_{SO} F_{\ell,t} Q_2(x, a) \mid \mathcal{H}_{\ell-1} \qquad \forall (x, a) \in \mathcal{X} \times \mathcal{A}, t \in \{0, ..., H-1\}.$$

Proof Conditioned on $\mathcal{H}_{\ell-1}$,

$$F_{\ell,t} Q(x, a) = \frac{\sigma_\ell^2(t, x, a) \bar{\theta}_{t,x,a}}{\lambda} + \frac{\sigma_\ell^2(t, x, a)}{v} \left(\sum_{\substack{(r, x') \in \\ D_{\ell-1}(t, x, a)}} r + \max_{a' \in \mathcal{A}} Q(x', a') \right) + w_\ell(t, x, a)$$

is a convex function of $(Q(x', a'))_{x' \in \mathcal{X}, a' \in \mathcal{A}}$ convolved with the independent noise term $w_\ell(t, x, a)$. The result therefore follows by Lemma 7. ∎

Consider the random variable $Y = P^T V$ where $V \in \mathbb{R}^n$ and $P \sim \text{Dirichlet}(\alpha)$. Then, Y has mean $V^T \alpha / \mathbb{1}^T \alpha$. The size of its fluctuations depends on how concentrated P is around its mean, captured by the pseudocount $\mathbb{1}^T \alpha = \sum_{i=1}^n \alpha_i$, and spread of the elements in V, captured by $\text{Span}(V) \equiv \max_i V_i - \min_j V_j$. The next lemma shows that a Gaussian random variable large enough mean and variance is stochastically optimistic with respect to Y. This result is established in the appendix.

Lemma 9 (Gaussian vs Dirichlet optimism)
Let $Y = P^T V$ for $V \in \mathbb{R}^n$ fixed and $P \sim \text{Dirichlet}(\alpha)$ with $\alpha \in \mathbb{R}_+^n$ and $\sum_{i=1}^n \alpha_i \geq 3$. Let $X \sim N(\mu, \sigma^2)$ with $\mu \geq \frac{\sum_{i=1}^n \alpha_i V_i}{\sum_{i=1}^n \alpha_i}$, $\sigma^2 \geq 3 \left(\sum_{i=1}^n \alpha_i \right)^{-1} \text{Span}(V)^2$, then $X \geq_{SO} Y$.

With Lemma 9 in place, we can now establish a sense in which the Bellman operator underlying RLSVI is stochastically optimistic relative to the true Bellman operator. Recall definition 2, which defines the value over outcomes (r, x') under Q by $V_Q(r, x') \equiv r + \max_{a' \in \mathcal{A}} Q(x', a')$.

---

7. This follows from Lemma 7 by looking at the pairs (X, Z) and (Y, Z) and taking $f : \mathbb{R}^2 \to \mathbb{R}$ to be $f(x_1, x_2) = x_1 + x_2$.

Lemma 10 (Stochastically optimistic operators)
Suppose Assumption 3 holds and RLSVI is applied with parameters $(\bar{\theta}, v, \lambda)$ satisfying $(v/\lambda) = \beta$. Then for any episode ℓ with history $\mathcal{H}_{\ell-1}$, time $t \in \{0, ..., H-1\}$, and pair $(x, a) \in \mathcal{X} \times \mathcal{A}$,

$$F_{\ell,t}Q(x,a) \mid \mathcal{H}_{\ell-1} \geq_{SO} F_{\mathcal{M},t}Q(x,a) \mid \mathcal{H}_{\ell-1}$$

for any fixed $Q \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$ such that $v \geq 3\mathrm{Span}(V_Q)^2$ and $\max_{x \in \mathcal{X}} V_Q(x) \leq \min_{t,x,a} \bar{\theta}_{t,x,a}$.

Remark 11 *When $Q(x,a) \geq 0$ for all (x,a), $\mathrm{Span}(V_Q) \leq \|V_Q\|_\infty \leq \|Q\|_\infty + 1$. Therefore it suffices that $v \geq 3(1 + \|Q\|_\infty)^2$ and $\min_y \bar{\theta}_y \geq \|Q\|_\infty + 1$.*

Proof Recall the Bellman update of Q under the true MDP $\mathcal{M}$ is

$$F_{\mathcal{M},t}Q(x,a) = V_Q^T \mathcal{P}_{t,x,a}^O$$

For each $y = (t, x, a)$, $\mathcal{P}_y^O | \mathcal{H}_{\ell-1} \sim \mathrm{Dirichlet}(\alpha_{\ell,y})$ with $\alpha_{\ell,y} = \alpha_{0,y} + n_\ell(y)\hat{P}_{\ell,y}^O \in \mathbb{R}^{2|\mathcal{X}|}$. Similarly, for each $y = (t, x, a)$, plugging in $\beta = v/\lambda$ we have

$$F_{\ell,t}Q(x,a) \mid \mathcal{H}_{\ell-1} \sim N(\mu_y, \sigma_y^2)$$

where

$$\mu_y \equiv \frac{\beta \bar{\theta}_y + n_\ell(y) V_Q^T \hat{\mathcal{P}}_{\ell,y}^O}{\beta + n_\ell(y)} \qquad \sigma_y^2 \equiv \frac{v}{n_\ell(y) + \beta}.$$

The result follows from Lemma 9 if we establish $\sigma_y^2 \geq (\mathbb{1}^T \alpha_{\ell,y})^{-1}\mathrm{Span}(V_Q)^2$ and $\mu_y \geq V_Q^T \alpha_{\ell,y}/\mathbb{1}^T \alpha_{\ell,y}$. We have

$$\frac{3 \cdot \mathrm{Span}(V_Q)^2}{\mathbb{1}^T \alpha_{\ell,y}} = \frac{3 \cdot \mathrm{Span}(V_Q)^2}{\beta + n_\ell(y)} \leq \sigma_y^2$$

because of the assumption that $v \geq 3 \cdot \mathrm{Span}(V_Q)^2$. Next we have

$$\frac{V_Q^T \alpha_{\ell,y}}{\mathbb{1}^T \alpha_{\ell,y}} = \frac{V_Q^T \alpha_{0,y} + n_\ell(y) V_Q^T \hat{\mathcal{P}}_y^O}{\beta + n_\ell(y)} \leq \frac{\beta \max_{x \in \mathcal{X}} V_Q(x) + n_\ell(y) V_Q^T \hat{\mathcal{P}}_y^O}{\beta + n_\ell(y)} \leq \mu_y$$

because of the assumption that $V_Q(x) \leq \min_y \bar{\theta}_y$ for all x. $\blacksquare$

Lemmas 8 and 10 together imply the stochastic optimism of the state-action value functions $Q_{\ell,0}$ generated by RLSVI.

Corollary 12 *If Assumption 3 holds and RLSVI is applied with parameters $(\bar{\theta}, v, \lambda)$ satisfying $(v/\lambda) = \beta$, $v \geq 3H^2$ and $\min_y \bar{\theta}_y \geq H$,*

$$Q_{\ell,0}(x,a) \mid \mathcal{H}_{\ell-1} \geq_{SO} Q_{\mathcal{M},0}^*(x,a) \mid \mathcal{H}_{\ell-1}$$

for any history $\mathcal{H}_{\ell-1}$ and state-action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$.

Proof To reduce notation, we prove this for episode $\ell = 1$, but the proof follows identically for general ℓ by conditioning on the history $\mathcal{H}_{\ell-1}$ at every step. Recall that $Q_{1,0} = F_{1,0}F_{1,1}\cdots F_{1,H-1}0$ and $Q_{\mathcal{M},0}^* = F_{\mathcal{M},0}F_{\mathcal{M},1}\cdots F_{\mathcal{M},H-1}0$.

By Lemma 10,

$$(F_{1,H-1}0)(x,a) \succeq_{SO} (F_{\mathcal{M},H-1}0)(x,a) \qquad \forall x,a.$$

Proceeding by induction, suppose for some $t \leq H - 1$

$$(F_{1,t+1}F_{1,t+2}\cdots F_{1,H-1}0)(x,a) \succeq_{SO} (F_{\mathcal{M},t+1}F_{\mathcal{M},t+2}\cdots F_{\mathcal{M},H-1}0)(x,a) \qquad \forall x,a.$$

Combining this with Lemma 8 shows

$$
\begin{aligned}
F_{1,t}(F_{1,t+1}F_{1,t+2}\cdots F_{1,H-1}0)(x,a) \quad &\succeq_{SO} \quad F_{1,t}(F_{\mathcal{M},t+1}F_{\mathcal{M},t+2}\cdots F_{\mathcal{M},H-1}0)(x,a) \\
&\succeq_{SO} \quad F_{\mathcal{M},t}(F_{\mathcal{M},t+1}F_{\mathcal{M},t+2}\cdots F_{\mathcal{M},H-1}0)(x,a)
\end{aligned}
$$

where the final step uses Lemma 10 combined with the fact that for any $t \in \{0,..,H-1\}$,

$$Q \equiv F_{\mathcal{M},t+1}F_{\mathcal{M},t+2}\cdots F_{\mathcal{M},H-1}0$$

satisfies $3 \cdot \mathrm{Span}(V_Q) \leq 3H \leq v$ and $Q \leq \bar{\theta}$. ∎

6.6. Analysis of on-policy Bellman error: proof of Theorem 1

The proof relies on the following bound. For standard Gaussian random variables $X_1,...,X_n$, a basic Gaussian maximal inequality implies $\mathbb{E}[\max_i X_i] \leq \sqrt{2\log(n)}$. The next lemma is a generalization of this result, which can be seen by taking $J = \arg\max_j X_j$. This lemma is implied by Proposition A.1. of (Russo and Zou, 2015).

Lemma 13 *Let (X, J) be jointly distributed random variables where $X \in \mathbb{R}^n$ follows a multivariate Gaussian distribution with $X_j \sim N(0, \sigma_j^2)$ and $J \in \{1,...n\}$ is a random index. Then*
$$\mathbb{E}[X_J] \leq \sqrt{2\log(n)\mathbb{E}[\sigma_J^2]}.$$

Applying this leads to two bounds that are used in our analysis. The first bounds the noise terms $w_\ell(t, x_t, a_t)$ of RLSVI at the state and action visited by RLSVI, and the second bounds the norm of the value function sampled by RLSVI.

Corollary 14 *For each $t \leq H$ and $\ell \leq L$*

$$\mathbb{E}[w_\ell(t, x_t, a_t)] \leq \sqrt{2\log(|\mathcal{A}||\mathcal{X}|)\mathbb{E}[\sigma_\ell(t, x_t, a_t)^2]}.$$

Corollary 15 *If RLSVI is applied with parameters $(\lambda, v, \bar{\theta})$ with $v/\lambda = \beta \geq 3$, $v = 3H^2$ and $\bar{\theta} = H\mathbb{1}$,*

$$\mathbb{E}\big[\max_{\ell \leq L, t < H} \|V_{Q_{\ell,t+1}}\|_\infty\big] \leq 2H + H^2\sqrt{2\log(1 + |\mathcal{X}||\mathcal{A}|HL)}.$$

A proof of this corollary is provided in the appendix. We now complete the regret analysis of RLSVI and establish Theorem 1.

Proof Set
$$\Delta_\ell = V^*_{\mathcal{M},0}(x_0^\ell) - V^{\pi_\ell}_{\mathcal{M},0}(x_0^\ell).$$

By Corollary 5 and Corollary 12,
$$\mathbb{E}\left[\sum_{\ell=1}^{L} \Delta_\ell\right] \le \mathbb{E}\left[\sum_{\ell=1}^{L}\sum_{t=0}^{H-1}(F_{\ell,t}Q_{\ell,t+1} - F_{\mathcal{M},t}Q_{\ell,t+1})(x_t^\ell, a_t^\ell)\right].$$

The posterior-mean Bellman update of Q under $\mathcal{M}$ is
$$\mathbb{E}[F_{\mathcal{M},t}Q(x,a)|\mathcal{H}_{\ell-1}] = V_Q^T\mathbb{E}[\mathcal{P}^O_{t,x,a}|\mathcal{H}_{\ell-1}].$$

Recall as well that for each $y = (t,x,a)$, $\mathcal{P}^O_y|\mathcal{H}_{\ell-1} \sim \text{Dirichlet}(\alpha_{\ell,y})$ with
$$\alpha_{\ell,y} = \alpha_{0,y} + n_\ell(y)\hat{P}^O_{\ell,y} \in \mathbb{R}^{2|\mathcal{X}|}.$$

Since the prior over $\mathcal{P}^O_{t,x,a}(\cdot)$ is distributed independently across states and actions (t,x,a), and (t,x,a) cannot be visited prior to period t in any episode, we have also that
$$\mathcal{P}^O_{t,x,a}|\mathcal{H}_{\ell-1}, x_0^\ell, a_0^\ell, .., x_t^\ell, a_t^\ell \sim \text{Dirichlet}(\alpha_{\ell,y}).$$

As a result
$$
\begin{aligned}
\mathbb{E}[F_{\mathcal{M},t}Q(x,a)|\mathcal{H}_{\ell-1}, x_0^\ell, a_0^\ell, .., x_t^\ell, a_t^\ell] &= \mathbb{E}[F_{\mathcal{M},t}Q(x,a) \mid \mathcal{H}_{\ell-1}]\\
&= \frac{V_Q^T\alpha_{0,y} + n_\ell(y)V_Q^T\hat{P}^O_{\ell,y}}{\beta + n_\ell(y)}\\
&\ge \frac{-\beta\|V_Q\|_\infty}{\beta + n_\ell(y)} + \frac{n_\ell(y)V_Q^T\hat{P}^O_{\ell,y}}{\beta + n_\ell(y)}.
\end{aligned}
$$

By equation (6.4), we find
$$F_{\ell,t}Q(x,a) - \mathbb{E}[F_{\mathcal{M},t}Q(x,a)|\mathcal{H}_{\ell-1}, x_1^\ell, a_1^\ell, .., x_t^\ell, a_t^\ell] \le \frac{\beta(\|\bar{\theta}\|_\infty + \|V_Q\|_\infty)}{\beta + n_\ell(y)} + w_\ell(y).$$

Then,
$$
\begin{aligned}
\mathbb{E}[\Delta_\ell] &\le \mathbb{E}\left[\sum_{t=0}^{H-1}(F_{\ell,t}Q_{\ell,t+1} - F_{\mathcal{M},t}Q_{\ell,t+1})(x_t^\ell, a_t^\ell)\right]\\
&= \mathbb{E}\left[\sum_{t=0}^{H-1}((F_{\ell,t}Q_{\ell,t+1})(x_t^\ell, a_t^\ell) - \mathbb{E}[F_{\mathcal{M},t}Q_{\ell,t+1}(x_t^\ell, a_t^\ell) \mid \mathcal{H}_{\ell-1}, x_1^\ell, a_1^\ell, .., x_t^\ell, a_t^\ell])\right]\\
&\le \mathbb{E}\left[\sum_{t=0}^{H-1}\frac{\beta(\|\bar{\theta}\|_\infty + \|V_{Q_{\ell,t+1}}\|_\infty)}{\beta + n_\ell(t, x_t^\ell, a_t^\ell)} + w_\ell(t, x_t^\ell, a_t^\ell)\right]
\end{aligned}
$$

where the second inequality uses that $Q_{\ell,t+1}$ and $F_{\mathcal{M},t}$ are independent conditioned on $\mathcal{H}_{\ell t}$. Summing over episodes $\ell \in \{1, .., L\}$ implies
$$\mathbb{E}\sum_{\ell=1}^{L}\Delta_\ell \le \mathbb{E}\left[\beta\left(\|\bar{\theta}\|_\infty + \max_{\ell \le L, t < H}\|V_{Q_{\ell,t+1}}\|_\infty\right)\sum_{t<H,\ell\le L}\frac{1}{\beta + n_\ell(t, x_t^\ell, a_t^\ell)} + \sum_{\ell\le L, t\le H}w_\ell(t, x_t^\ell, a_t^\ell)\right].$$

34

Each term can be bounded separately. By Corollary 14

$$\mathbb{E} \sum_{\ell \leq L, t < H} w_\ell(t, x_t^\ell, a_t^\ell) \leq \mathbb{E} \sum_{\ell \leq L, t < H} \sigma_\ell(t, x_t^\ell, a_t^\ell)\sqrt{2\log(|\mathcal{X}||\mathcal{A}|)} \quad = \quad \mathbb{E} \sum_{\ell \leq L, t < H} \sqrt{\frac{2v\log(|\mathcal{X}||\mathcal{A}|)}{\beta + n_\ell(t, x_t^\ell, a_t^\ell)}}$$

$$\overset{(a)}{\leq} \quad 2\sqrt{2vH^2|\mathcal{X}||\mathcal{A}|L\log(|\mathcal{X}||\mathcal{A}|)}$$

$$= \quad 2H^2\sqrt{6|\mathcal{X}||\mathcal{A}|L\log(|\mathcal{X}||\mathcal{A}|)}$$

where the second to last inequality is proved in Lemma 16, provided below. The other term can be bounded as,

$$\mathbb{E}\left[\beta\left(\|\bar{\theta}\|_\infty + \max_{\ell \leq L, t \leq H}\|V_{Q_{\ell, t+1}}\|_\infty\right)\sum_{t \leq T, \ell \leq L}\frac{1}{\beta + n_\ell(t, x_t^\ell, a_t^\ell)}\right]$$

$$\overset{(b)}{\leq} \quad \beta\left(\|\bar{\theta}\|_\infty + \mathbb{E}\left[\max_{\ell \leq L, t \leq H}\|V_{Q_{\ell, t+1}}\|_\infty\right]\right)H|\mathcal{X}||\mathcal{A}|\log\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right)$$

$$\overset{(c)}{\leq} \quad \beta\left(H + 2H + H^2\sqrt{2\log(1 + |\mathcal{X}||\mathcal{A}|HL)}\right)H|\mathcal{X}||\mathcal{A}|\log\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right)$$

$$\leq \quad 5\beta H^3|\mathcal{X}|\|\mathcal{A}|\sqrt{\log_+(1 + |\mathcal{X}||\mathcal{A}|HL)}\log\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right)$$

where the bound on the sum in inequality (b) is from Lemma 16 (proof in the Appendix), and inequality (c) applies Corollary 15.

Lemma 16 *If $\beta \geq 2$ then with probability 1,*

$$\sum_{\ell \leq L}\sum_{t \leq H}\frac{1}{\beta + n_\ell(t, x_t^\ell, a_t^\ell)} \leq H|\mathcal{X}||\mathcal{A}|\log\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right)$$

and

$$\sum_{\ell \leq L}\sum_{t \leq H}\sqrt{\frac{1}{\beta + n_\ell(t, x_t^\ell, a_t^\ell)}} \leq 2\sqrt{H^2|\mathcal{X}||\mathcal{A}|L}.$$

Together, the calculations above yield the regret bound

$$\mathbb{E}\sum_{\ell=1}^{L}\Delta_\ell \leq 5\beta H^3|\mathcal{X}|\|\mathcal{A}|\sqrt{\log_+(1 + |\mathcal{X}||\mathcal{A}|HL)}\log\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right) + 2H^2\sqrt{6|\mathcal{X}||\mathcal{A}|L\log(|\mathcal{X}||\mathcal{A}|)}.$$

Unfortunately, this alone does not yield the desired bound of order $\tilde{O}(H^2\sqrt{|\mathcal{X}||\mathcal{A}|L})$. To complete the proof, we consider two cases. First suppose $L \geq 25\beta H^2|\mathcal{X}||\mathcal{A}|$. Then

$$\mathbb{E}\sum_{\ell=1}^{L}\Delta_\ell \quad \leq \quad H^2\sqrt{|\mathcal{X}||\mathcal{A}|L\log_+(1 + |\mathcal{X}||\mathcal{A}|HL)}\left(2\sqrt{6} + 5\beta H\sqrt{|\mathcal{X}||\mathcal{A}|/L}\log\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right)\right)$$

$$\leq \quad H^2\sqrt{|\mathcal{X}||\mathcal{A}|L\log_+(1 + |\mathcal{X}||\mathcal{A}|HL)}\left(2\sqrt{6} + \sqrt{\beta}\log_+\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right)\right)$$

$$\overset{(*)}{\leq} \quad 6H^2\sqrt{\beta|\mathcal{X}||\mathcal{A}|L\log_+(1 + |\mathcal{X}||\mathcal{A}|HL)}\log_+\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right)$$

35

which is the desired bound. When $L \le 25H^2|\mathcal{X}||\mathcal{A}|$, we use the naive bound

$$\mathbb{E}\sum_{\ell=1}^{L}\Delta_\ell \le HL \le H\sqrt{L}\sqrt{25H^2|\mathcal{X}||\mathcal{A}|} = 5H^2\sqrt{\beta|\mathcal{X}||\mathcal{A}|L},$$

which is also less than the term in $(*)$. This completes the proof of Theorem 1. ∎

7. Computational studies

In Section 6 we established formal guarantees for a tabular version of RLSVI. This result serves as a sanity check, demonstrating that RLSVI carries out efficient deep exploration, but the tabular nature and prior structure of the setting limits the scope of our theoretical results. Perhaps most importantly, the results do not apply when parameterized representations are used to generalize across states and actions. In this section, we present computational results that offer further assurances. In particular, we discuss results from a series of experiments designed to enhance insight into the workings of RLSVI beyond the scope of our theoretical analysis. The focus of these experiments is to improve understanding, rather than to solve challenging problems. Nevertheless, we believe that observations from these didactic examples will prove valuable toward the design of practical systems that require the synthesis of efficient deep exploration with effective generalization.

7.1. Deep-sea exploration

We begin our computational experiments with an empirical study of the "deep-sea exploration" problem from Example 1. This offers a simple illustration of the importance of deep exploration. Although the associated MDP has only N^2 states, dithering schemes require a number of episodes that grows exponentially in N to effectively explore the environment. Deep exploration approaches, on the other hand, can effectively explore the environment within a sub-exponential number of episodes. Our results verify the efficacy randomized value functions and that RLSVI carries out deep exploration.

7.1.1. TABULAR REPRESENTATION

We begin with an investigation into RLSVI with a tabular representation. Our goal will be to study the behavior of RLSVI in a simple setting similar to that addressed by Theorem 1. To do this, we generate random "deep-sea" environments according to Example 1 and empirically evaluate performance over many simulations.

We apply RLSVI `agent` with infinite buffer, greedy actions and `learn_grlsvi`[8] with a tabular representation. Specifically, each component of the parameter vector $\theta \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ provides a value estimate for one state-action pair. We set the tuning parameters to $v = H^2/25$, $\overline{\theta} = 0$, and $\lambda = v$. Note that, compared to the setting specified in Theorem 1 we rescaled v by a constant in order to accelerate learning in the deterministic deep-sea environment. Further our prior parameter $\overline{\theta}$ is not optimistic. This choice is not particularly important in terms of performance on this task, in fact setting $\overline{\theta} = H\mathbb{1}$ leads to almost

---

8. `learn_grlsvi` is `learn_rlsvi` (Algorithm 4) with additive Gaussian reward noise of variance v (5.5)

identical results. The reason we do not rely on an optimistic prior is to highlight the practical efficacy of RLSVI even without optimistic prior. This will be important in later sections where we study RLSVI in domains with generalization, for which the notion of an universally 'optimistic prior' does not carry over from the tabular setting (Osband et al., 2016b).

We compare the performance of RLSVI against two well-studied reinforcement learning algorithms specifically designed to explore efficiently with tabular representations: UCRL2 (Jaksch et al., 2010) and PSRL (Osband et al., 2013). We similarly modify UCRL2 and PSRL that accelerate learning in the deterministic deep-sea environment[9]. For each of the algorithms our modifications reduce learning times but do not affect rates at which learning times scale with problem size.
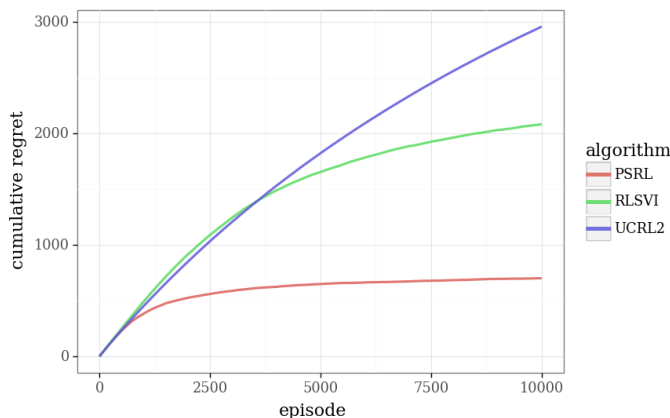
Figure 4: RLSVI is competitive with algorithms designed for tabular exploration ($N = 10$).

Figure 4 plots the average regret realized by RLSVI (specifically, `learn_grlsvi`), UCRL2 and PSRL, over five seeds with a bomb and five seeds with treasure. Among the algorithms, PSRL offers the lowest level of regret, followed by RLSVI, and then UCRL2. Hence, RLSVI is competitive with these algorithms, which are designed to yield efficient exploration with tabular representations.

One natural question is how this performance scales with the size N of the problem. To answer this we study the "learning time," defined to be the first episode where the average regret per episode is less than 0.5. Formally,

$$(7.1) \qquad \text{Learning time}(\mathcal{M}^*, \text{alg}) := \min \left\{ L > 1 \,\middle|\, \frac{\text{Regret}(\mathcal{M}^*, \text{alg}, L)}{L} \leq 0.5 \right\},$$

This quantity is random, as it depends on the realization of $\mathcal{M}^*$. For an algorithm with regret bound $\text{Regret}(\mathcal{M}^*, \text{alg}, L) \leq \sqrt{BL}$ we would expect the learning time to be $\tilde{O}(B)$.

The results of Theorem 1 suggests an $\tilde{O}(\sqrt{H^3 SAL})$ *average* scaling when the environment is drawn from a symmetric Dirichlet distribution. We can contrast this to existing performance guarantees for UCRL2 which, when adapted to this setting, provide a

---

9. Specifically, we update the confidence sets for PSRL and UCRL2 as if each observed transition (s, a, r, s') occured identically 10 times repeatedly. We also further rescale the confidence sets for UCRL2 to be 10 times smaller than prescribed by the analysis.

$\tilde{O}(\sqrt{H^3 S^2 A L})$ regret bound. For the deep-sea problem, $H = N$, $S = N^2$ and $A = 2$, and the bounds therefore suggests that learning times scale as $\tilde{O}(N^5)$ for RLSVI and $\tilde{O}(N^7)$ for UCRL2. Figure 5 shows that observed performance to a large degree matches performance suggested by these theoretical results. The best known bound for PSRL also suggests a $\tilde{O}(N^5)$ scaling. However, recent work suggests that this bound is loose (Osband and Van Roy, 2017, 2016), and the associated plot in Figure 5 strengthens the case for that hypothesis.
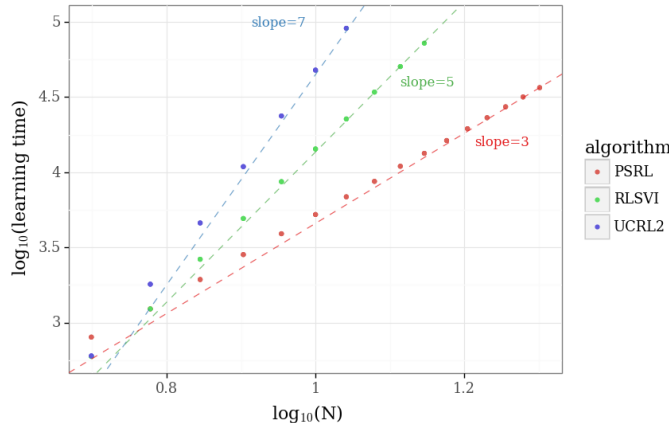
Figure 5: Scaling with tabular learning.

7.1.2. LINEARLY PARAMETERIZED VALUE FUNCTIONS

Section 7.1.1 presents evidence of the efficacy of RLSVI with a tabular representation. However, the value of RLSVI lies in its ability to function well with parameterized value functions that generalize across states and actions. Model-based algorithms such as UCRL2 or PSRL do not accommodate this form of generalization.

In this subsection, we continue our investigation of the "deep-sea" environment, but now using linear parameterized representations. To do this, we generate a random subspace of dimension D that is specifically designed to include the true optimal value function of the deep-sea environment irrespective of whether there is treasure or a bomb. We then generate a random basis of D unit vectors $\phi_1, \ldots, \phi_D \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ that span the space. Each vector ϕ_d can be thought of as representing a feature that assigns a numerical value to each state-action pair. As such, the representation can be thought of as a linear combination of features, with a dimensional parameter vector $\theta \in \mathbb{R}^D$ encoding feature weighs: $\tilde{Q}_\theta = \sum_{d=1}^{D} \theta_d \phi_d$.

To facilitate efficient computation in the deep-sea problem we restrict these D-dimensional features so that each is nonzero only at state-action pairs corresponding to one row of the grid of states. We only consider values of D that are multiples of N, and for each row assign $M = D/N$ features to generate nonzero values. With this representation, RLSVI learns a separate M-dimensional representation for each row, avoiding a costly dimension D inversion. Figure 6 plots realized regret generated by `learn_grlsvi` with $\lambda = 100$, $v = 0.01$, and $N = 50$. Once again, we simulate this problem for five random seeds with treasure and five random seeds with a bomb and report the average regret. These results demonstrate that

per-episode regret vanishes much faster than any dithering method, which would expect at least $2^{50} \simeq 10^{15}$ episodes to even reach the chest!
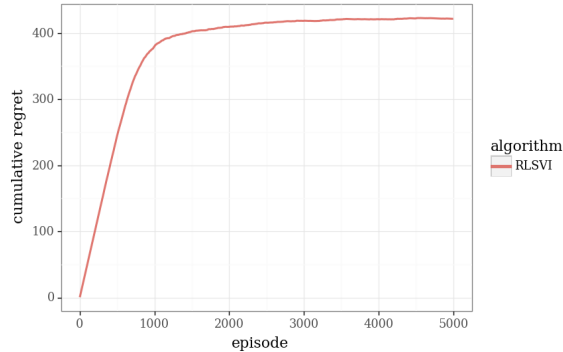

Figure 6: Regret with $N = 50$ and $M = 50$

It is the cases with treasure rather than a bomb that bind regret and learning times. For this reason we will only present results associated with the former case from here on to save on computation. Figure 7(a) plots learning times as a function of N for different numbers of features M per row. As one would expect, the learning time increases with the number of features. Importantly, this scaling with chain length N is graceful and grows much more slowly than even the lower bound for dithering methods $O(2^N)$. Figure 7(b) plots the same data on a log-log scale to highlight this sub-exponential growth. We can see empirically that the slope on this scale is approximately two, implying that learning time scales approximately quadratically in N.

(a) Raw values

(b) Log-log plot shows scaling
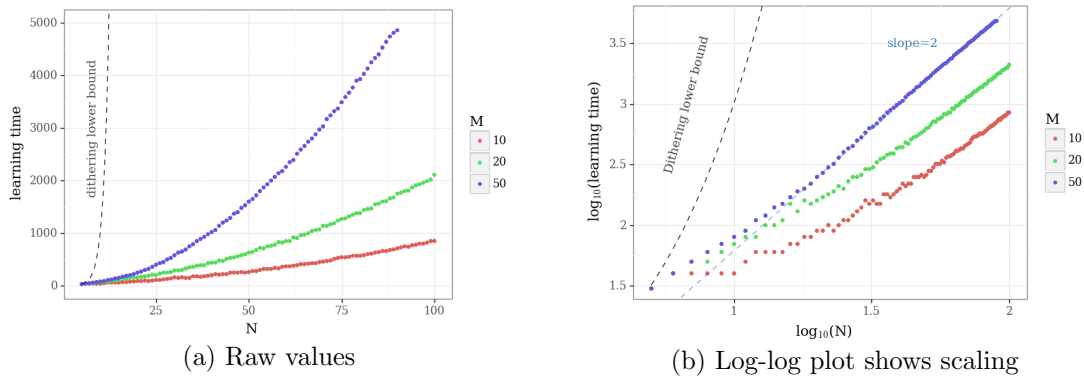
Figure 7: Effect of problem size N on learning time.

For another perspective on scaling, Figure 8 presents plots of learning times as a function of the number of features M, for several values of N. In each case, the learning time appears to grow linearly in the number of features up until some threshold and then increase much more slowly beyond this point. The vertical dotted lines in Figure 8 appear at $M = 2N$. Empirically, this seems to be the point beyond which the incremental learning time incurred with additional features is small. Intuitively, one might speculate that this is reasonable because $2N$ is equal to the maximum number of states-action pairs which can be observed in any time period. Beyond this point, additional features must be linearly dependent.
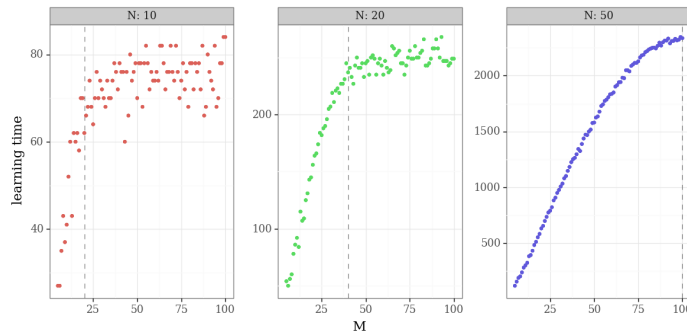
Figure 8: Scaling with number of features.

7.1.3. MISSPECIFIED REPRESENTATIONS

Let us now consider a more realistic setting in which the value function representation is mis-specified in the sense that Q^* is not equal to $\tilde{Q}_\theta$ for any vector θ. We experiment with a setting completely analogous to that of the previous section, except we add to each feature vector ϕ_d a random vector $\eta_d \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$. The random vector η_d is nonzero only only at state-action pairs associated with the feature ϕ_d. Each nonzero noise component is sampled from $N(0, \psi I)$. Hence, we make use of a representation of the form $\tilde{Q}_\theta = \sum_{d=1}^D \theta_d (\phi_d + \eta_d)$. As the parameter ψ increases, the representation becomes increasingly misspecified.

Figure 9 plots cumulative regret of `learn_grlsvi` with varying numbers of features and degrees of misspecification over 5000 episodes. Our results are the average of 20 seeds for each value of the noise scale ψ. These results indicate that RLSVI remains robust to some degree of misspecification. However, at some point the model-misspecification becomes too severe as the value of ψ increases depending upon the number of basis functions M. The power of the representation increases with the number of features, and this enables RLSVI to tolerate larger values of ψ. In the case $M \geq 2N$ random basis functions will span the true value function with high probability. As expected, we observe that for $M = 40, N = 20$ RLSVI performs similarly well irrespective of ψ.
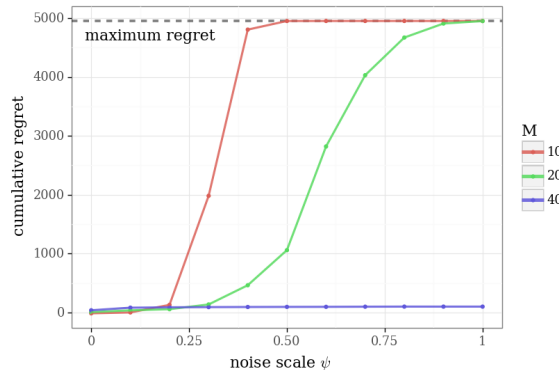
Figure 9: Robustness to misspecification with $N = 20$.

7.1.4. Parameter tuning

The computational results we have present in Sections 7.1.2 and 7.1.3 make use of particular settings for the prior and noise variance parameters $\lambda = 100, v = 0.01$. In this section, we study the dependence of results on these parameter settings. The deep-sea problem we have considered is in some sense degenerate because each problem instance is deterministic. In order to offer a more representative set of results pertaining to variance parameter tuning, we will also consider a modified version of the deep sea problem where all reward observations are corrupted by some $N(0,1)$ noise.

Figure 10 plots the cumulative average regret after 5000 episodes over 10 random seeds for various choices of prior and noise variance with $N = 20$ and $M = 10$. In both settings with and without stochastic rewards, and for all choices of noise randomization, we can see that prior variance which is too small can prohibit learning. In this problem, where our prior $\overline{\theta} = 0$ is not informative, choosing even very large λ does not degrade performance.
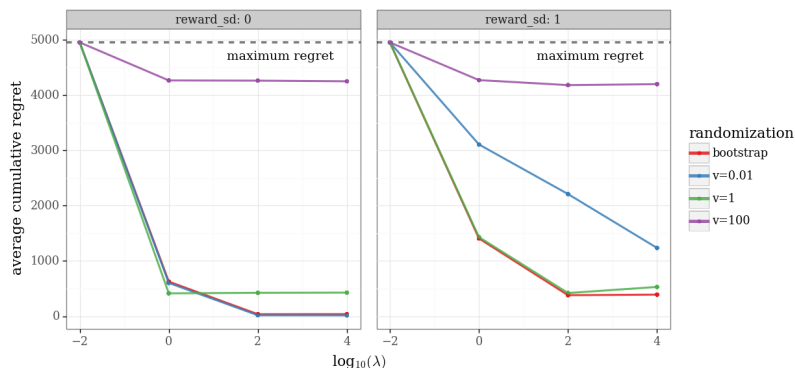
Figure 10: Robustness to prior and noise variance parameters.

Figure 11 takes the same data as Figure 10 but investigates the sensitivity of RLSVI to the noise randomization, for the choice $\lambda = 100$. We see that choice of the best-performing noise variance v is largely dependent upon the scale of the noise in the actual environment. When the underlying environment is deterministic there is no benefit to adding noise and low values of v perform best. However, when the environment is stochastic choosing v on the order of the variance of the noise in the environment is necessary to not fall victim to unlucky observations. In both settings, bootstrapping performs competitively with the ex-ante "best" choice of v but does not need to be specified in advance.
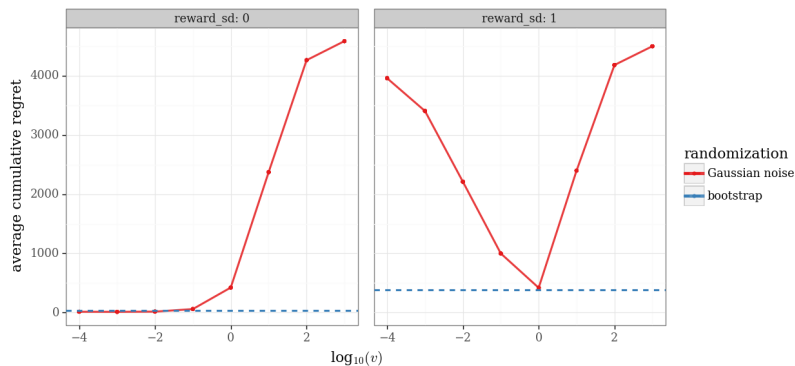

Figure 11: Bootstrap is competitive with the best choice of v across levels of noise.

To gain some more intuition for this parameter tuning we take this same data as Figure 11 and present the realized regret by random seed in Figure 12. We see v smaller than the noise in the problem can lead to premature and sub-optimal convergence that never opens the chest (and so leads to linear regret). Choices of v which are too large lead to slower learning and more exploration, but do not lead to linear regret. We note that RLSVI with $v > 0$ does not seem to significantly degrade in performance for stochastic rewards with variance up to v. Once again, we see that randomization by bootstrap is competitive with the best ex-ante choice of v but with one fewer parameter to tune.
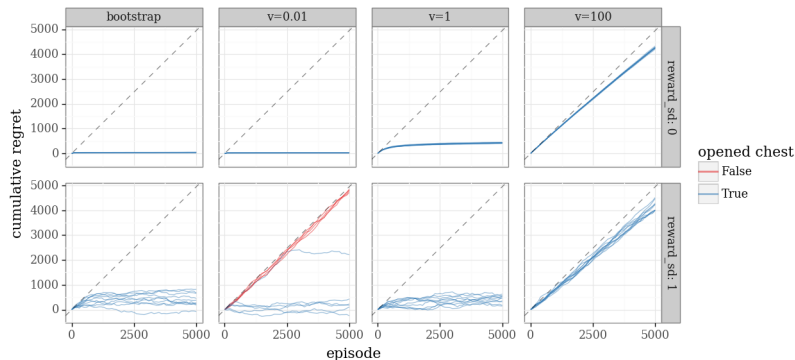
Figure 12: Higher v is more robust to stochastic environments but learns more slowly.

The bootstrap learns the "right" noise variance, and beyond that, can even learn how it should vary over states and actions. Further, Figure 13 plot learning times from applying the `learn_brlsvi` in the same settings to which `learn_grlsvi` was applied to generate Figures 6 and 7. These results suggest that learning times of `learn_brlsvi` scale similarly to those of `learn_grlsvi`. In general, our results suggest that bootstrapping offers a natural approach to setting an appropriate form of randomizing noise without prior knowledge.
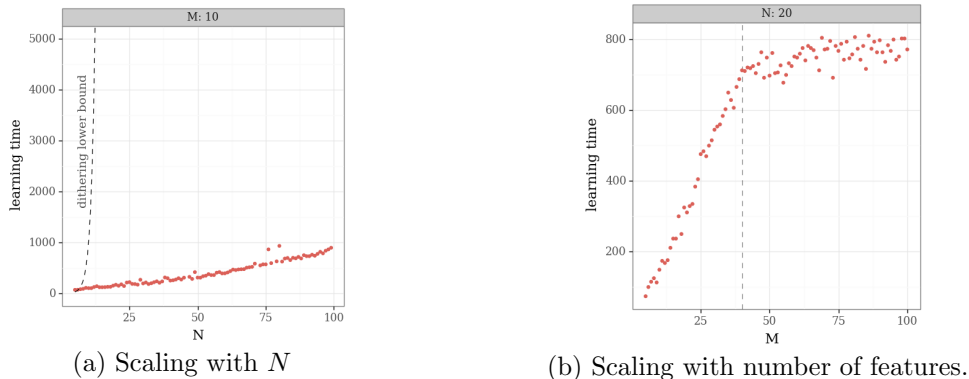

(a) Scaling with N

(b) Scaling with number of features.

Figure 13: Performance of the bootstrap scales similarly to `learn_grlsvi`.

7.2. Deep exploration with deep learning

The experiments of Section 7.1 are designed to highlight several key properties of RLSVI in a simple setting. These results demonstrate that RLSVI can successfully synthesize efficient

exploration with generalization. However, the context was a "toy" example in that the underlying system involved a tractable number of states. Further, the algorithms we evaluate in that section are not practical for large-scale learning problems for two reasons. First, they recompute the whole history of data each episode and so have computational costs that grow with the amount of data collected. Second, their performance is highly reliant upon an accurate linear basis for the value function whereas many of the recent breakthroughs in the field have come from so-called "deep" RL that uses deep neural networks for function approximation in RL.

In this section we present computational results for deep exploration with a practical variant `learn_ensemble_rlsvi` together with neural network function approximation. We begin with more experiments on the "deep sea" problem and show that a parallel RLSVI strategy can recover performance qualitatively similar to full batch resampling, but at a dramatically lower computational cost. Next we investigate the scaling properties of RLSVI with nonlinear neural network representations and find that the performance can successfully synthesize exploration with generalization in this setting. Finally, we apply these findings to a difficult task in continuous control. We find that `learn_ensemble_rlsvi` successfully demonstrates deep exploration together with complex nonlinear generalization.

7.2.1. RLSVI via ensemble sampling

In this section we apply `learn_ensemble_rlsvi` (Algorithm 8) for $K = 1, 5, 10, 20, 40$ and with an `ensemble_buffer` that stores the most recent 10^5 transitions. For `update` we use `update_bootstrap` (Algorithm 7) to approximate a "double or nothing" online bootstrap (Owen and Eckles, 2012). We use a discounted TD loss with $\gamma = 0.99$, learning rate $\alpha = 10^{-3}$ and minibatch size of 128. For our value function family $\mathcal{Q}$ we consider two-layer MLP with 50 rectified linear units in each layer.

In place of explicit prior regularization $\mathcal{R}$ we evaluate $\tilde{Q}_{\theta_k} = f^{\text{MLP}}_{\theta_k} + f^{\text{MLP}}_{\theta_k^0}$ where f^{MLP} is a 2-layer MLP and the parameters θ_k, θ_k^0 are sampled independently from Glorot initialization and henceforth θ_k^0 is held fixed (Glorot and Bengio, 2010; Osband et al., 2018). We found that this randomization plus SGD training provide sufficient regularization for deep learning without use of weight decay (Zhang et al., 2016; Bartlett et al., 2017). Using two separate networks, one with fixed weights, is useful so that the SGD training cannot easily learn to ignore the state input and learn $Q = 0$ as a degenerate global solution. More detail on this specific prior mechanism for deep neural networks is available in Osband et al. (2018).

We apply this algorithm to "deep sea" problem with a raw pixel representation in $[0,1]^{N \times N}$ where the diver's position is given by a 1 and all other entries zero. This problem is by nature tabular and does not necessitate complex representation via neural network however we use it as a simple example to investigate the importance of ensemble size K. Figure 14 presents the performance for $N = 25$ averaged over 20 seeds. We see that even a relatively small number of parallel estimates K can direct deep exploration and that, for a problem of this size, the marginal benefits seem to plateau around $K = 20$. The computational savings of this parallel approach can be quite significant. We compare the cost of $K = 20$ parallel Q-networks each with $O(1)$ computation per episode against a naive `learn_brlsvi` that computes $H = 25$ value functions each with $\Omega(L)$ computation per episode.
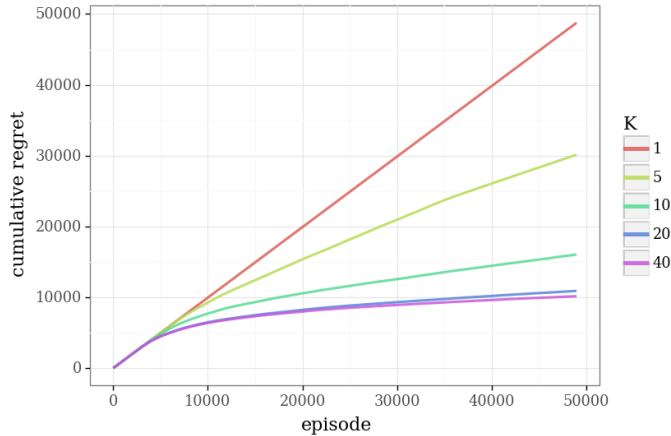
Figure 14: Investigating ensemble size K for `learn_ensemble_rlsvi` on deep sea $N = 25$.

Our next set of experiments investigates the scalability of `learn_ensemble_rlsvi` with different feature representations. We repeat the experiment of Figure 14 with $K = 20$ varying $N = 5, .., 50$ under three separate representations and averaged over 20 seeds. First we consider the raw pixel representation as above; this is effectively a tabular problem. Next, we consider the informative linear basis of Section 7.1.2 with $M = 10$; this learning can be expedited through generalization. Finally, we consider the raw pixel representation but alter the problem formulation so that the action 1 is always "go right"; the optimal value function for this setting takes a particularly simple form and an optimal policy can easily be happened upon through random weight initialization and without any learning. Figure 15 shows that RLSVI with neural network architecture can exploit these feature representations where they are present, but defaults to an approximately tabular learning approach when they are not.
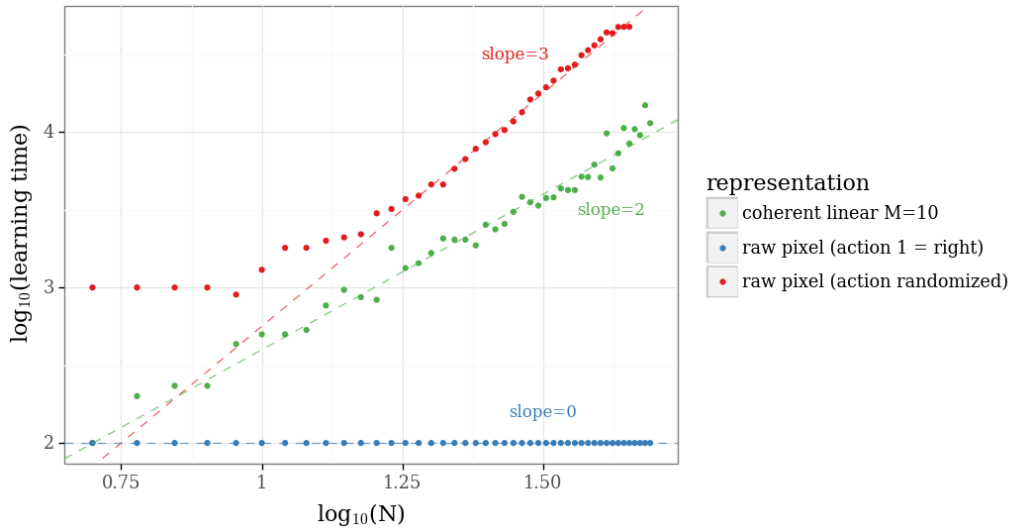

Figure 15: Log-log plot shows empirical scalings with different feature representations.

44

7.2.2. Cartpole swing up

In this section we consider the classic "cartpole" problem of a cart attached to a pole on a frictionless rail. We modify the problem so that, as in previous sections, deep exploration is crucial to finding rewarding states and thus learning the optimal policy. However, unlike "deep sea" the underlying dynamics are not governed by a small finite MDP. The cart is of mass $M = 1$ and the pole is mass $m = 0.1$ and length $l = 1$, with acceleration due to gravity $g = 9.8$. At each timestep the agent can apply a horizontal force F_t to the cart. The dynamics for this system are given by a second order differential equation in x_t, the horizontal position of the cart and θ_t, the angle of the pole from vertically upright at $\theta = 0$,

$$(7.2) \qquad \tau_t = \frac{F_t + \frac{l}{2}\dot{\theta}_t^2 \sin(\theta_t)}{m + M}, \ \ddot{\theta}_t = \frac{g\sin(\theta_t) - \cos(\theta_t)\tau_t}{\frac{l}{2}\left(\frac{4}{3} - \frac{m}{m+M}\cos(\theta_t)^2\right)}, \ \ddot{x} = \tau - \frac{m\frac{l}{2}\ddot{\theta}_t\cos(\theta_t)}{m + M}.$$

Unlike the traditional cartpole problem, where the agent begins with the pole stood upright and must learn to balance it; our agent begins each episode with the pole hanging down and has to learn to swing it up. Concretely we interact with the environment through the state $s_t := (\theta_t, \dot{\theta}_t, x_t, \dot{x}_t, t) \in \mathbb{R}^5$. Each episode begins with $s_0 = (\pi, 0, 0, 0) + w$ for $w_i \sim \text{Unif}([-0.05, 0.05])$ i.i.d. in each component. We discretize the evolution of (7.2) with timescale $\Delta t = 0.01$ and present the choice of actions $F_t \in \{-10, 0, 10\}$ for all t. The reward structure of this task is specifically designed to necessitate deep exploration; each timestep the agent pays a cost $\frac{|F_t|}{1000}$ for its action but can receive a reward of 1 if the pole is balanced upright and steady in the middle[10]. The ends of the rail at $x = -5, 5$ are rigid and immovable; an episode ends whenever $t > 10$.

Figure 16 presents results for DQN and a 50-50-MLP with rectified linear units with linear annealing ϵ-greedy dithering from 1 to 0 over varying number of episodes. Irrespective of annealing schedule, DQN is unable to gather informative data since it does not perform deep exploration. In this environment, dithering strategies for exploration are insufficient to gain information beyond the locally-attractive policy to remain motionless and receive a reward of zero.
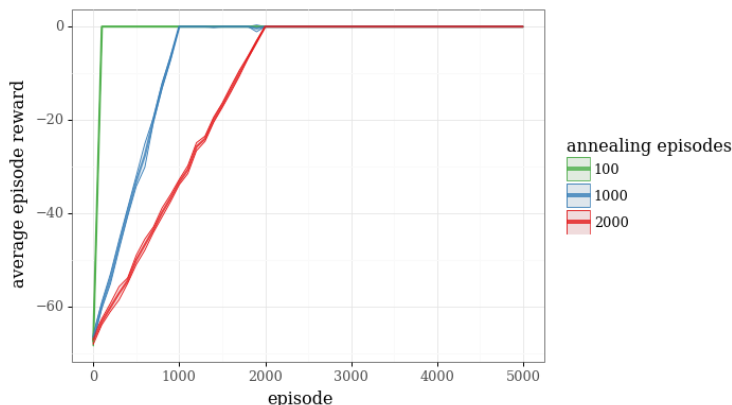

Figure 16: DQN with ϵ-greedy exploration simply learns to stay motionless.

---

10. Reward +1 received if $\cos(\theta) > 0.95$ with all of $|\dot{\theta}|, |x|, |\dot{x}| \leq 1$.

Figure 17 presents the average episodic reward for an ensemble approach to RLSVI with $K = 20$ and the same algorithmic approach as Section 7.2.1. We note that, unlike DQN with ϵ-greedy exploration, RLSVI is able to learn a successful swing up policy with an identical network architecture. In addition, we note that RLSVI implemented with linear or single-layer Q-value functions is unable to learn a successful swing up policy. This demonstrates the importance of *both* deep exploration and deep representation learning in order for a successful application of deep RL in this setting. The results of Figures 16 and 17 are averaged over 20 seeds, with confidence intervals at 1 standard error of the mean. We present visualization of this performance `https://youtu.be/ia72VyW5MfI`.
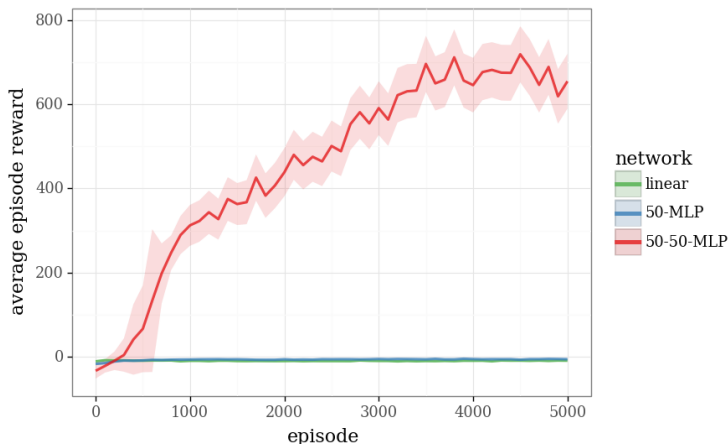

Figure 17: RLSVI with 2-layer neural network is able to learn a near-optimal policy.

The computational results we present in this paper are tailored to be simple and interpretable, with a clear focus on the importance of deep exploration and the compatibility of this approach with linear and nonlinear value function learning. Related work investigates scaling up this approach in several arcade games including Tetris, Angry Birds, Atari 2600 as well as a model of recommendation systems (Osband et al., 2016b; Ibarra et al., 2016; Osband et al., 2016a). We consciously choose to keep our empirical investigation concise and the key results sanitary, but look forward to pushing the boundaries of large-scale applications of "deep RL" via randomized value functions in future work.

8. Closing remarks

Much of the applied RL literature focuses on simulated systems and learning a good final policy, potentially over billions or trillions of episodes. Assessed in this manner, performance is driven largely by the investment of computational resources and simulation time; not just how effectively a reinforcement learning algorithm makes decisions and interprets observations. In many real systems, data collection is costly or constrained by the physical context, and this calls for a focus on statistical efficiency. In these contexts it may be more appropriate, for example, to evaluate algorithms over a fixed number of episodes.

Exploration is a key driver of statistical efficiency. As discussed in Section 4, there can be an exponentially large difference in data requirements between an agent that explores via dithering, as has commonly been done in past applications of reinforcement learning, and

an agent that carries out deep exploration. In this paper, we have developed randomized value functions as a concept that enables efficient deep exploration in conjunction with value function learning methods commonly used in reinforcement learning.

Acknowledgements

APPENDIX

Appendix A. Proofs of technical lemmas

A.1. Proof of Lemma 3

Lemma 1 (Planning Error to Bellman Error) *Let $Q_0, Q_1, Q_2, ..., Q_H \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$ be any sequence with $Q_H = 0 \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$ and take $\pi = (\pi_0, \pi_1, ...)$ to be the policy $\pi_t(x) = \arg\max_{a \in \mathcal{A}} Q_t(x, a)$ for all a, x. Then for any MDP $\mathcal{M}$ and initial state $x \in \mathcal{X}$,*

$$(A.1) \qquad Q_0(x, \pi_0(x)) - V_{\mathcal{M}}^{\pi}(x) = \mathbb{E}_{\mathcal{M}, \pi}\left[\sum_{t=0}^{H}((Q_t - F_{\mathcal{M}, t}Q_{t+1})(x_t, a_t))|x_0 = x\right]$$

Proof Define the operator $F_{\mathcal{M}, t}^{\pi}$ at time t for the MDP $\mathcal{M}$ and policy π by

$$F_{\mathcal{M}, t}^{\pi}Q(x, a) = \mathbb{E}[r_{t+1} + Q(x_{t+1}, \pi_{t+1}(x_{t+1}))|\mathcal{M}, x_t = x, a_t = a].$$

Let $Q_{\mathcal{M}, 0}^{\pi}, ..., Q_{\mathcal{M}, H}^{\pi} \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$ be defined according to $Q_{\mathcal{M}, H}^{\pi} = 0$ and

$$Q_{\mathcal{M}, t}^{\pi} = F_{\mathcal{M}, t}^{\pi}Q_{\mathcal{M}, t+1}^{\pi} \quad t \in \{0, ..., H-1\}.$$

Then $Q_{\mathcal{M}, 0}^{\pi}(x, \pi_0(x)) = V_{\mathcal{M}}^{\pi}(x)$ and, since $\pi_t(x) = \arg\max_a Q_{t+1}(x, a)$, $F_{\mathcal{M}, t}^{\pi}Q_{t+1} = F_{\mathcal{M}, t}Q_{t+1}$ for all t. We can therefore rewrite (A.1) as

$$Q_0(x, \pi_0(x)) - Q_{\mathcal{M}, 0}^{\pi}(x, \pi_0(x)) = \mathbb{E}_{\mathcal{M}, \pi}\left[\sum_{t=0}^{H-1}\left((Q_t - F_{\mathcal{M}, t}^{\pi}Q_{t+1})(x_t, a_t)\right)|x_0 = x\right].$$

We have

$$\begin{aligned} Q_0 - Q_{\mathcal{M}, 0}^{\pi} &= Q_0 - F_{\mathcal{M}, 0}^{\pi}Q_1 + F_{\mathcal{M}, 0}^{\pi}Q_1 - Q_{\mathcal{M}, 0}^{\pi} \\ &= Q_0 - F_{\mathcal{M}, 0}^{\pi}Q_1 + F_{\mathcal{M}, 0}^{\pi}Q_1 - F_{\mathcal{M}, 0}^{\pi}Q_{\mathcal{M}, 1}^{\pi}. \end{aligned}$$

By definition, this means

$$(Q_0 - Q_{\mathcal{M}, 0}^{\pi})(x, \pi_0(x)) = \left(Q_0 - F_{\mathcal{M}, 0}^{\pi}Q_1\right)(x, \pi_0(x)) + \mathbb{E}_{\mathcal{M}, \pi}[\left(Q_1 - Q_{\mathcal{M}, 1}^{\pi}\right)(x_1, a_1)|x_0 = x].$$

The result follows by iterating this relation. ∎

A.2. Proof of Lemma 7

Lemma 2 (Preservation under convex operations) *For two collections $(X_1, ..., X_n)$ and $(Y_1, ..., Y_n)$ of independent random variables with $X_i \geq_{SO} Y_i$ for each $i \in \{1, ...n\}$ and any convex increasing function $f : \mathbb{R}^n \to \mathbb{R}$,*

$$f(X_1, ..., X_n) \geq_{SO} f(Y_1..., Y_n).$$

Proof The proof proceeds by induction on n. First consider the base case $n = 1$. Fix any convex increasing function $u : \mathbb{R} \to \mathbb{R}$. Then $u \circ f$ is convex increasing and

$$\mathbb{E}[u(f(X_1))] \geq \mathbb{E}[u(f(Y_1))].$$

Now suppose the result holds for any collection of $n - 1$ random variables. Fix any convex increasing $u : \mathbb{R} \to \mathbb{R}$: Define the convex increasing functions $U_X : \mathbb{R} \to \mathbb{R}$ and $U_Y : \mathbb{R} \to \mathbb{R}$ by

$$
\begin{aligned}
U_X(z) &\equiv \mathbb{E}[u(f(z, X_2, ..., X_n))] \\
U_Y(z) &\equiv \mathbb{E}[u(f(z, Y_2, ..., Y_n))].
\end{aligned}
$$

For each fixed $z \in \mathbb{R}$, $U_X(z) \geq U_Y(z)$ since

$$U_X(z) = \mathbb{E}[u(f_z(X_2, .., X_n))] \geq \mathbb{E}[u(f_z(Y_2, .., Y_n))] = U_Y(z)$$

where $f_z : \mathbb{R}^{n-1} \to \mathbb{R}$ is the convex increasing function $f_z(x_2, ..., x_n) = f(z, x_2, ..., x_n)$ and $f_z(X_2, ...X_n) \succeq_{SO} f_z(Y_2, ...Y_n)$ by the inductive hypothesis. We conclude

$$\mathbb{E}[u(f(X_1, ..., X_n))] = \mathbb{E}[U_X(X_1)] \geq \mathbb{E}[U_Y(X_1)] \geq \mathbb{E}[U_Y(Y_1)] = \mathbb{E}[u(f(X_1, ..., X_n))]$$

where the first and last equality use the independence of $(X_1, ..., X_n)$ and $(Y_1, ..., Y_n)$ along with the Fubini–Tonelli theorem. The final inequality uses the definition of stochastic optimism. ∎

A.3. Proof of Corollary 15

Corollary 1 *If RLSVI is applied with parameters $(\lambda, v, \bar{\theta})$ with $v/\lambda = \beta \geq 3$, $v = 3H^2$ and $\bar{\theta} = H\mathbb{1}$,*

$$\mathbb{E}[\max_{\ell \leq L, t < H} \|V_{Q_{\ell, t+1}}\|_\infty] \leq 2H + H^2 \sqrt{2 \log(|\mathcal{X}||\mathcal{A}|HL)}$$

Proof To begin, we observe a basic fact about the maximum of Gaussian random variables. Fix independent Gaussian random variables $X_0, X_1, ..., X_n \sim N(0, 1)$. Let $f : (x_0, ..., x_n) \mapsto \max_i x_i$ be the maximum function, so $\mathbb{E}[f(X_0, ..., X_n)] \leq \sqrt{2 \log(n + 1)}$ by a standard Gaussian maximum inequality. Then by Jensen's inequality,

$$
\begin{aligned}
\mathbb{E}\left[\left(\max_{i \in \{1, ..., n\}} X_i\right)_+\right] = \mathbb{E}[f(0, X_1, ..., X_n)] &= \mathbb{E}[f(\mathbb{E}[(X_0, X_1, ..., X_n)|X_1, ..., X_n])] \\
&\leq \mathbb{E}[f(X_0, X_1, ..., X_n)] \\
&\leq \sqrt{2 \log(n + 1)}.
\end{aligned}
$$

(A.2)

For every state action value function $Q \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|}$, $\|V_Q\|_\infty \leq 1 + \|Q\|_\infty$. Therefore, by equation 6.4, for every episode ℓ and period t,

$$
\begin{aligned}
\|F_{\ell, t} Q\|_\infty &\leq \max\{\|\theta\|_\infty, \|V_Q\|_\infty\} + \max_{x \in \mathcal{X}, a \in \mathcal{A}} w_\ell(t, x, a) \\
&\leq \max\{\|\theta\|_\infty, \|Q\|_\infty\} + 1 + \max_{x \in \mathcal{X}, a \in \mathcal{A}} w_\ell(t, x, a) \\
&\leq \max\{\|\theta\|_\infty, \|Q\|_\infty\} + 1 + w_{\max}
\end{aligned}
$$

where $w_{\max} \triangleq \left(\max_{t \leq H, \ell \leq L, a \in \mathcal{A}, x \in \mathcal{X}} w_\ell(t, x, a) \} \right)_+$. This implies

$$\|Q_{\ell, H-1}\|_\infty = \|F_{\ell, H-1} 0\|_\infty \leq \|\theta\|_\infty + 1 + w_{\max}.$$

Then

$$\|Q_{\ell, H-2}\|_\infty = \|F_{\ell, H-2} Q_{\ell, H-1}\|_\infty \leq \max\{\|\theta\|_\infty, \|Q_{\ell, H-1}\|_\infty\} + 1 + w_{\max} \leq \|\theta\|_\infty + 2(1 + w_{\max}).$$

Repeating this by backward induction shows,

$$\max_{t < H} \|\|Q_{\ell, t+1}\|_\infty \leq \|\theta\|_\infty + (H-1)(1 + w_{\max})$$

Therefore

$$\max_{t < H} \|V_{Q_{\ell, t+1}}\|_\infty \leq 1 + \max_{t < H} \|Q_{\ell, t+1}\|_\infty \leq \|\bar{\theta}\|_\infty + H(1 + w_{\max}).$$

In addition

$$\mathbb{E}[w_{\max}] = \mathbb{E}\left[\left(\max_{t \leq H, \ell \leq L, a \in \mathcal{A}, x \in \mathcal{X}} w_\ell(t, x, a) \right)_+ \right] = \mathbb{E}\left[\left(\max_{t < H, \ell \leq L, a \in \mathcal{A}, x \in \mathcal{X}} \sigma_\ell(t, x, a) \frac{w_\ell(t, x, a)}{\sigma_\ell(t, x, a)} \right)_+ \right]$$

$$\leq \sqrt{\frac{v}{\beta}} \mathbb{E}\left[\left(\max_{t < H, \ell \leq L, a \in \mathcal{A}, x \in \mathcal{X}} \frac{w_\ell(t, x, a)}{\sigma_\ell(t, x, a)} \right)_+ \right]$$

$$\leq \sqrt{2v/\beta \log(1 + |\mathcal{X}||\mathcal{A}|HL)}$$

where the last step uses equation (A.2). Combining these results implies,

$$\mathbb{E}[\max_{\ell \leq L, t < H} \|V_{Q_{\ell, t+1}}\|_\infty] \leq \|\bar{\theta}\|_\infty + H + H \mathbb{E}[w_{\max}] \leq \|\bar{\theta}\|_\infty + H + H\sqrt{2(v/\beta) \log(1 + |\mathcal{X}||\mathcal{A}|HL)}.$$

The result then follows by plugging in for $\beta \geq 3$, $v = 3H^2$, and $\bar{\theta} = H$. ∎

A.4. Proof of Lemma 16

Lemma 3 *If $\beta \geq 2$, with probability 1,*

$$\sum_{\ell \leq L} \sum_{t \leq H} \frac{1}{\beta + n_\ell(t, x_t, a_t)} \leq H|\mathcal{X}||\mathcal{A}| \log\left(\frac{1 + L}{|\mathcal{X}||\mathcal{A}|} \right)$$

and

$$\sum_{\ell \leq L} \sum_{t \leq H} \sqrt{\frac{1}{\beta + n_\ell(t, x_t, a_t)}} \leq 2H\sqrt{|\mathcal{X}||\mathcal{A}|L}.$$

Proof Set $\mathcal{Y} = \{0, ..., H-1\} \times \mathcal{X} \times \mathcal{A}$ to be the set of valid period, state, action triples $y = (t, x, a) \in \mathcal{Y}$. Note that $|\mathcal{Y}| = H|\mathcal{X}||\mathcal{A}|$. We have

$$
\sum_{t \leq H, \ell \leq L} \frac{1}{\beta + n_\ell(t, x_t, a_t)} = \sum_{y \in \mathcal{Y}} \sum_{i=0}^{n_L(y)-1} \frac{1}{\beta + i} \leq \sum_{y \in \mathcal{Y}} \int_{\beta-1}^{n_L(y)+\beta-1} \frac{1}{z} dz
$$

$$
= \sum_{y \in \mathcal{Y}} \log\left(\frac{\beta - 1 + n_L(y)}{\beta - 1}\right)
$$

$$
\leq \sum_{y \in \mathcal{Y}} \log\left(1 + n_L(y)\right)
$$

$$
\leq |\mathcal{Y}| \log\left(\frac{\sum_{y \in \mathcal{Y}}(1 + n_L(y))}{|\mathcal{Y}|}\right)
$$

$$
= |\mathcal{Y}| \log\left(1 + \frac{LH}{|\mathcal{Y}|}\right).
$$

$$
= H|\mathcal{X}||\mathcal{A}| \log\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right).
$$

In addition

$$
\sum_{t \leq H, \ell \leq L} \sqrt{\frac{1}{\beta + n_\ell(t, x_t, a_t)}} = \sum_{y \in \mathcal{Y}} \sum_{i=0}^{n_L(y)-1} \sqrt{\frac{1}{\beta + i}} \leq \sum_{y \in \mathcal{Y}} \int_{z=0}^{n_L(y)} \frac{1}{(\beta - 1 + z)^{1/2}} dz
$$

$$
\leq \sum_{y \in \mathcal{Y}} \int_{z=0}^{n_L(y)} \frac{1}{z^{1/2}} dz
$$

$$
= \sum_{y \in \mathcal{Y}} 2\sqrt{n_L(y)}
$$

$$
\leq 2\sqrt{|\mathcal{Y}| \sum_{y \in \mathcal{Y}} n_L(y)}
$$

$$
= 2H\sqrt{|\mathcal{X}||\mathcal{A}|L}.
$$

∎

A.5. Proof of Lemma 9

This section establishes Lemma 9 through a sequence of results. First, Lemma 17 provides general conditions for stochastic optimism. Next, we use these properties to show a stochastic dominance relation between Dirichlet and Beta distributions in Lemma 18 and between Beta and Gaussian distributions in Lemma 19. The final proof of Lemma 9, given at the end of this section, uses a simple combination of these results.

We begin with Lemma 17, which reproduces several sufficient conditions for stochastic optimism. Conditions (1.) and (2.) are classic results in the theory of second order

stochastic dominance (SSD) and we refer to (Hadar and Russell, 1969; Hanoch and Levy, 1969) for proofs. [11] For a clever and explicit construction of coupled random variables per condition (1.) see (Machina and Pratt, 1997). In fact, both (1.) and (2.) are also necessary conditions for stochastic optimism, although our results do not rely upon this.

Condition (3.) is less widely stated, but is a simple consequence of (2.) and has been known since Hanoch and Levy (1969). To our knowledge, condition (4.) is a new result that provides a sufficient condition for stochastic optimism that is much easier to verify than condition (3.). Some intuition for this result is captured by Figure 18a. We provide a detailed proof that analytically establishes many of the properties observable in Figure 18.

Lemma 17 (Sufficient Conditions for Stochastic Optimism) *Consider two integrable random variables X and Y with probability density functions f and g. Let $F(s) = \int_{-\infty}^{s} f(x)dx$ and $G(s) = \int_{-\infty}^{s} g(x)dx$ denote the corresponding cumulative distribution functions. Then $X \succeq_{SO} Y$ if any of the following properties hold:*

1. *One can construct random variables $(\tilde{X}, \tilde{Y}, \tilde{W})$ on a joint probability space such that $\tilde{X}$ has marginal distribution F, $\tilde{Y}$ has marginal distribution G, $\mathbb{E}[\tilde{W}|\tilde{Y}] \geq 0$ and*

$$\tilde{X} = \tilde{Y} + \tilde{W}.$$

2. *For all $a \in \mathbb{R}$,*

$$\int_{a}^{\infty} (G(s) - F(s))\, ds \geq 0.$$

3. *$\mathbb{E}[X] \geq \mathbb{E}[Y]$ and there exists $a \in \mathbb{R}$ such that*

$$G(s) \geq F(s) \iff s \geq a.$$

4. *$\mathbb{E}[X] \geq \mathbb{E}[Y]$ and $C := \{x \in \mathbb{R} : g(x) > f(x)\}$ is convex.*

Proof (1.) follows easily by the tower property of conditional expectation and the conditional Jensen inequality. We have $\mathbb{E}[\tilde{X} \mid \tilde{Y}] \geq \tilde{Y}$ and hence for any convex increasing function $u : \mathbb{R} \to \mathbb{R}$,

$$\mathbb{E}[u(\tilde{X})] = \mathbb{E}[\mathbb{E}[u(\tilde{X}) \mid \tilde{Y}]] \geq \mathbb{E}[u(\mathbb{E}[\tilde{X} \mid \tilde{Y}])] \geq \mathbb{E}[u(\tilde{Y})].$$

(2.) follows from integration by parts and (3.) is a fairly simple consequence of (2.). See Hanoch and Levy (1969) for a proof. The claim (4.) is new and established here.

Define the function

$$D(s) = G(s) - F(s) = \int_{-\infty}^{s} (g(x) - f(x))dx$$

---

11. The literature on SSD generally considers *concave* and increasing utility functions whereas we consider *convex* and increasing utility functions. Results on SSD can be easily translated into results about stochastic optimism through swapping positive and negative signs.

and note $C = \{x : g(x) - f(x) > 0\} = \{x : D'(x) > 0\}$. We have $\lim_{t \to \infty} D(t) = 0$. Two distributions yielding convex C and the corresponding functions $D'(s)$ and $D(s)$ are pictured in Figure 18.

Integration by parts shows (Hanoch and Levy, 1969, Lemma 1)

$$(\text{A.3}) \qquad \int_{-\infty}^{\infty} D(s)ds = \mathbb{E}[X] - \mathbb{E}[Y] \geq 0.$$

We first show $\sup\{C\} < \infty$. Note that we cannot have $C = (-\infty, \infty)$, as this would imply $\int g(x)dx > \int f(x)dx$, contradicting that g and f are probability density functions. Now suppose for contradition that $C = (\underline{c}, \infty)$ for some $\underline{c} \in \mathbb{R}$. Then, for $t \leq \underline{c}$ we have $D(t) = \int_{-\infty}^{t}(g(s) - f(s))ds < 0$. Since $\lim_{t \to \infty} D(t) = 0$, we have that for $t \geq \underline{c}$, $D(t) = -\int_{t}^{\infty}(g(s) - f(s))ds < 0$. Thus we have showed that $D(t) < 0$ for all $t \in \mathbb{R}$ contradicting (A.3).

Next, suppose $C = (-\infty, \overline{c})$ for $\overline{c} \in \mathbb{R}$. Then for $s \in C$, $D'(s) > 0$ by definition of C, which shows $D(t) = \int_{-\infty}^{t} D'(s)ds > 0$ for $t \in C$. But since $D'(s) < 0$ for all $s > \overline{c}$ and $\lim_{t \to \infty} D(t) = 0$, we must have $D(t) > 0$ also for all $t \geq \overline{c}$. Since D is non-negative, we have $\int_{a}^{\infty} D(s) \geq 0$ for all $a \in \mathbb{R}$ and hence $X \succeq_{SO} Y$. [12]

Finally, suppose $C = (-\underline{c}, \overline{c})$ and refer to Figures 18c and 18d for visual guidance. For $s < \underline{c}$, $D'(s) \leq 0$ and hence $D(t) = \int_{-\infty}^{t} D'(s)ds \leq 0$ for $t \leq \underline{c}$. For $s \in C$, $D'(s) > 0$. For $s > \overline{c}$, $D'(s) \leq 0$, which implies $D(t) \geq 0$ for all $t \geq \overline{c}$ since $\lim_{t \to \infty} D(t) = 0$. Since we have shown $D(\underline{c}) \leq 0 \leq D(\overline{c})$ there must exists a single crossing point $a \in (\underline{c}, \overline{c})$ with $D(a) = 0$. For this value of a, we have

$$G(s) - F(s) \iff s \geq a,$$

implying $X \succeq_{SO} Y$. This completes the proof of condition (4.). ∎

Our next result uses several relationships between Gamma, Beta, and Dirichlet random variables to establish a stochastic optimism relationship between specific matched distributions. First, for two independent Gamma distributed random variables $\gamma \sim \text{Gamma}(\alpha, 1)$ and $\gamma' \sim \text{Gamma}(\beta, 1)$, we have that $\frac{\gamma}{\gamma + \gamma'} \sim \text{Beta}(\alpha, \beta)$. Next, $\gamma + \gamma' \sim \text{Gamma}(\alpha + \beta, 1)$ and $\mathbb{E}[\gamma \mid \gamma + \gamma'] = (\gamma + \gamma')\left(\frac{\alpha}{\alpha + \beta}\right)$. Finally, for a collection of independent Gamma random variables $\gamma_i \sim \text{Gamma}(\alpha_i, 1)$, the random probability vector $P = \left(\frac{\gamma_1}{\sum_{j=1}^{n} \gamma_j}, \dots \frac{\gamma_n}{\sum_{j=1}^{n} \gamma_j}\right)$ follows a Dirichlet(α) distribution. Lemma 18 compares the distribution of the inner product $P^{\top}V$ between a Dirichlet random variable P and a fixed vector V to an appropriate Beta random variable X.

Lemma 18 (Beta-Dirichlet optimism) *If $V \in \mathbb{R}^n$ where $0 = V_1 \leq \cdots \leq V_n = 1$, $P \sim$ Dirichlet(α) where $\alpha \in \mathbb{R}_+^n$, and $X \sim \text{Beta}(\sum_{i=1}^{n} \alpha_i V_i, \sum_{i=1}^{n} \alpha_i(1 - V_i))$ then $X \succeq_{SO} P^{\top}V$.*

Proof Our proof constructs coupled Dirichlet and and Beta random variables with the marginal distributions described in the lemma's statement. Consider independent Gamma-distributed random variables $(\gamma_1^0, \gamma_1^1, \dots, \gamma_n^0, \gamma_n^1)$ where

$$\gamma_i^0 \sim \text{Gamma}(\alpha_i V_i, 1) \quad \text{and} \quad \gamma_i^1 \sim \text{Gamma}(\alpha_i(1 - V_i), 1) \qquad i = 1, \dots, n.$$

---

12. In fact, X first-order stochastically dominates Y in this case.

(a) Comparison of PDFs

(b) Comparison of CDFs

(c) Difference in PDFs $D'(s) = g(s) - f(s)$
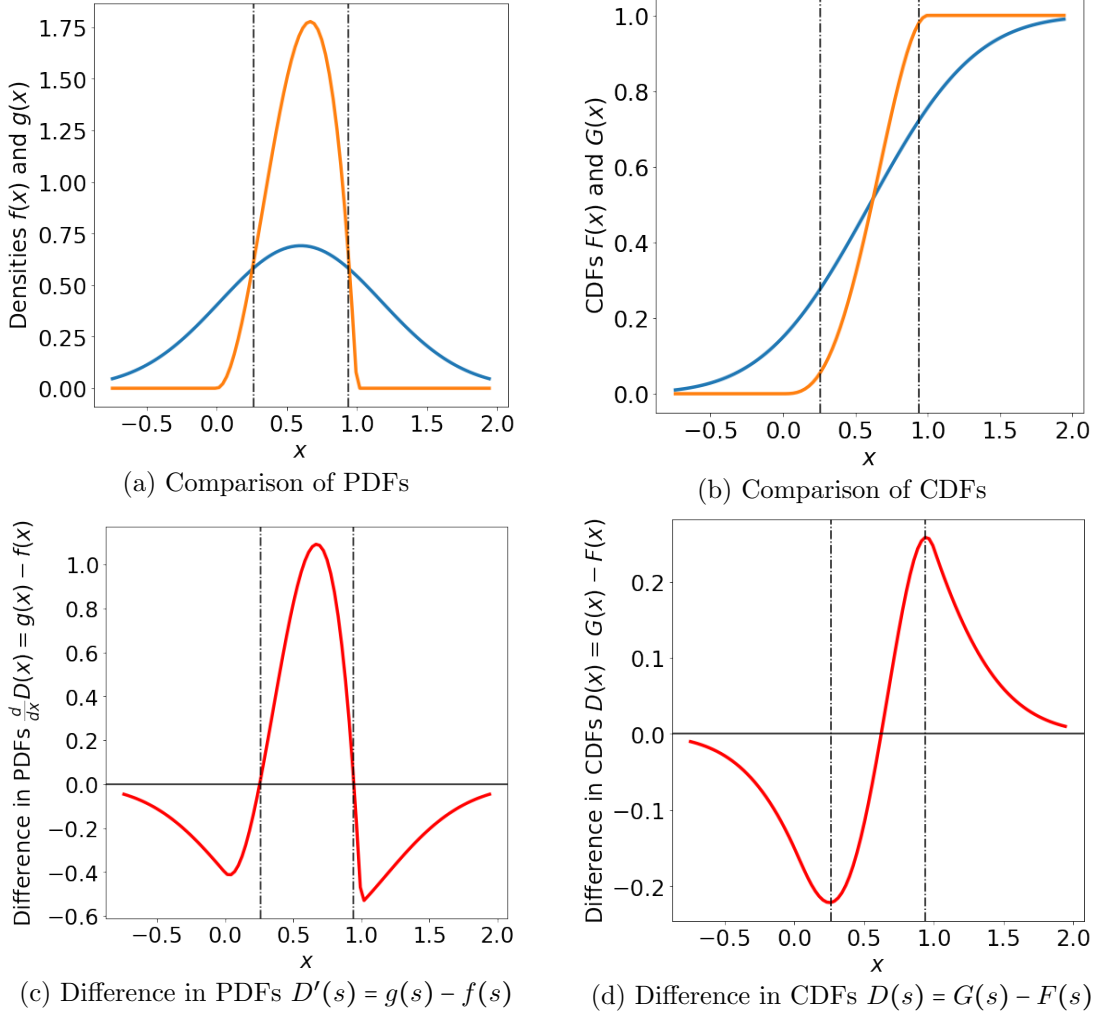
(d) Difference in CDFs $D(s) = G(s) - F(s)$

Figure 18: Comparison of Beta$(3, 2)$ and $N(3/5, 1/3)$ distributions. The dashed vertical lines indicate the boundaries of set C.

Set $\gamma_i = \gamma_i^0 + \gamma_i^1$ so that $\gamma_i \sim \mathrm{Gamma}(\alpha_i, 1)$. Then, set

$$P = \frac{(\gamma_1, \ldots, \gamma_n)}{\sum_{i=1}^n \gamma_i} \sim \mathrm{Dirichlet}(\alpha)$$

and

$$X = \frac{\sum_{i=1}^n \gamma_i^0}{\sum_{i=1}^n \gamma_i} = \frac{\sum_{i=1}^n \gamma_i^0}{\sum_{i=1}^n \gamma_i^0 + \sum_{i=1}^n \gamma_i^1} \sim \mathrm{Beta}\left(\sum_{i=1}^n \alpha_i V_i, \sum_{i=1}^n \alpha_i (1 - V_i)\right).$$

Our result follows from the fact that $\sum_{i=1}^n \gamma_i^0 \sim \mathrm{Gamma}(\sum \alpha_i V_i, 1)$ and $\sum_{i=1}^n \gamma_i^1 \sim \mathrm{Gamma}(\sum_{i=1}^n \alpha_i (1 - V_i), 1)$ and the ratio of Gamma distributed random variables follows a Beta distribution. Now, we have that

$$\mathbb{E}\left[\gamma_i^0 | \gamma_i\right] = \gamma_i \left(\frac{\mathbb{E}\left[\gamma_i^0\right]}{\mathbb{E}\left[\gamma_i^0\right] + \mathbb{E}\left[\gamma_i^1\right]}\right) = \gamma_i \left(\frac{\alpha_i V_i}{\alpha_i}\right) = \gamma_i V_i.$$

This implies,

$$\mathbb{E}[X|\gamma] = \mathbb{E}\left[\frac{\sum_{i=1}^n \gamma_i^0}{\sum_{i=1}^n \gamma_i}\,\middle|\,\gamma\right] = \frac{\sum_{i=1}^n \mathbb{E}[\gamma_i^0 \mid \gamma_i]}{\sum_{i=1}^n \gamma_i} = \frac{\sum_{i=1}^n \gamma_i V_i}{\sum_{i=1}^n \gamma_i} = P^\top V.$$

Set $Y = P^\top V$ and $W = X - Y$ so that $X = Y + W$. We have shown $\mathbb{E}[W \mid \gamma] = 0$ and so $\mathbb{E}[W \mid Y] = \mathbb{E}[\mathbb{E}[W \mid \gamma] \mid Y] = 0$ and the result follows from part 1 of Lemma 17. ∎

Our next result, Lemma 19 shows that a Gaussian distribution with sufficiently large mean and varaince stochastically dominates a Beta distribution. The proof shows this by looking at the ratio of their probability density functions and concluding this is a quasi-concave function. The basic properties of quasi-concave functions used here are in (Boyd and Vandenberghe, 2004, Section 3.4).

Lemma 19 (Gaussian-Beta optimism) *Let $X \sim N(\mu, \sigma)$ and $Y \sim \text{Beta}(\alpha, \beta)$. If $\alpha + \beta \geq 3$, $\mu = \alpha/(\alpha + \beta)$ and $\sigma^2 = (\alpha + \beta - 2)^{-1}$, then $X \succeq_{SO} Y$.*

Proof Let $f(x) = \frac{1}{\sqrt{2\pi}\sigma}e^{-(x-\mu)^2/2\sigma^2}$ and $g(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$ denote the probability density functions corresponding to the $N(\mu, \sigma)$ and $\text{Beta}(\alpha, \beta)$ distributions. We show $g(x)/f(x)$ is quasi-concave, which implies the super-level set $C = \{x : g(x)/f(x) \geq 1\}$ is convex. The result then follows by Lemma 17 and condition (4.).

Let $\ell(x) = \log(g(x)/f(x))$. Since $g(x)/f(x) = e^{\ell(x)}$ is a monotone function of ℓ, it is quasi-concave a long as $\ell(x)$ is quasi-concave. We show ℓ is quasi-concave by considering two cases. First, if $\alpha > 1$ and $\beta > 1$, we show ℓ is concave (and therefore quasi-concave). Differentiation shows

$$\ell''(x) = -\left(\frac{\alpha-1}{x^2} + \frac{\beta-1}{(1-x)^2}\right) + \frac{1}{\sigma^2} \leq -(\alpha + \beta - 2) + \frac{1}{\sigma^2} \leq 0$$

where the last step follows from our choice of σ^2. In the case where either $\alpha \in (0,1)$ or $\beta \in (0,1)$, we find that ℓ is monotone (and therefore quasi-concave). Assume $\beta \in (0,1)$, in which case $\alpha \geq 2$. Then,

$$\ell'(x) = \frac{\alpha-1}{x} - \frac{\beta-1}{1-x} + \frac{x-\mu}{\sigma^2} \geq \frac{\alpha-1}{x} + \frac{x-\mu}{\sigma^2}.$$

This is non-negative for $x \geq \mu$. For $x < \mu$, plugging in $\mu = \alpha/(\alpha + \beta)$ and $\sigma^2 \geq 1/(\alpha + \beta)$ gives

$$\ell'(x) \geq \frac{\alpha-1}{x} + \frac{x-\mu}{\sigma^2} \geq \frac{\alpha - \alpha/2}{x} + x(\alpha + \beta) - \alpha \geq \frac{\alpha}{2x} + \alpha(x-1) \geq 0$$

where the last inequality can be verified by solving $\min_{x>0} \frac{2}{x} + x - 1 = \frac{2}{\sqrt{1/2}} + \sqrt{1/2} - 1 \geq 0$. ∎

With these results we are now ready to prove Lemma 9, which we restate below.

Lemma 4 *Let $Y = P^T V$ for $V \in \mathbb{R}^n$ fixed and $P \sim \text{Dirichlet}(\alpha)$ with $\alpha \in \mathbb{R}_+^n$ and $\sum_{i=1}^n \alpha_i \geq 3$. Let $X \sim N(\mu, \sigma^2)$ with $\mu \geq \frac{\sum_{i=1}^n \alpha_i V_i}{\sum_{i=1}^n \alpha_i}$, $\sigma^2 \geq 3\left(\sum_{i=1}^n \alpha_i\right)^{-1} \text{Span}(V)^2$, then $X \succeq_{SO} Y$.*

Proof [Proof of Lemma 9] Without loss of generality, assume $V_1 \le \ldots \le V_n$. Set $V_i' = (V_i - V_1)/(V_n - V_1)$ for $i = 1, \ldots, n$. If $X \sim \text{Beta}(\sum_{i=1}^n \alpha_i V_i', \sum_{i=1}^n \alpha_i (1 - V_i'))$, then $X \succeq_{SO} P^\top V'$ by Lemma 18. Let $Z \sim N(\mu, \sigma^2)$ where $\mu = \mathbb{E}[X] = \sum_i \alpha_i V_i'/(\sum_j \alpha_j)$ and $\sigma^2 = (\sum_{i=1}^n \alpha_i - 2)^{-1}$. Then $Z \succeq_{SO} X$ by Lemma 19. Combining these results gives

$$P^\top V = V_1 + (V_n - V_1) P^\top V' \preceq_{SO} V_1 + (V_n - V_1) X \preceq_{SO} V_1 + (V_n - V_1) Z.$$

We have that $V_1 + (V_n - V_1)Z$ is normally distributed with

$$\mathbb{E}[V_1 + (V_n - V_1)Z] = V_1 + (V_n - V_1)\mu = V_1 + (V_n - V_1)\left(\frac{\sum_i \alpha_i V_i'}{\sum_j \alpha_j}\right) = \frac{\sum_{i=1}^n \alpha_i V_i}{\sum_{i=1}^n \alpha_i}$$

and

$$\text{Variance}\,(V_1 + (V_n - V_1)Z) = (V_n - V_1)^2 \sigma^2 = \text{Span}(V)^2 \left(\sum_{i=1}^n \alpha_i - 2\right)^{-1} \le 3 \cdot \text{Span}(V)^2 \left(\sum_{i=1}^n \alpha_i\right)^{-1}.$$

∎

References

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.

Sander Adam, Lucian Busoniu, and Robert Babuska. Experience replay for real-time reinforcement learning control. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(2):201–212, 2012.

Shipra Agrawal and Navin Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 39–1, 2012.

Shipra Agrawal and Navin Goyal. Further optimal regret bounds for Thompson sampling. In *Artificial Intelligence and Statistics*, pages 99–107, 2013a.

Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *Proceedings of the 30th Annual International Conference on Machine Learning*, pages 127–135, 2013b.

Peter Auer and Ronald Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Advances in Neural Information Processing Systems 19*, pages 49–56, 2006.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *Proceedings of the 34th Annual International Conference on Machine Learning*, 2017.

Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through Bayesian deep q-networks. *arXiv preprint arXiv:1802.04412*, 2018.

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.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems 30*, pages 6241–6250, 2017.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems 29*, pages 1471–1479. 2016.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Advances in Neural Information Processing Systems 30*, 2017.

Dimitri P. Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, September 1996.

Peter J Bickel and David A Freedman. Some asymptotic theory for the bootstrap. *The Annals of Statistics*, pages 1196–1217, 1981.

David Blackwell. Discounted dynamic programming. *The Annals of Mathematical Statistics*, 36(1):226–235, 1965.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

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.

Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems 28*, pages 2818–2826. 2015.

Richard Dearden, Nir Friedman, and Stuart J. Russell. Bayesian Q-learning. In *AAAI Conference on Artificial Intelligence*, pages 761–768, 1998.

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

Dean Eckles and Maurits Kaptein. Bootstrap thompson sampling and sequential decision problems in the behavioral sciences. *SAGE Open*, 9(2):2158244019851675, 2019.

Bradley Efron. *The jackknife, the bootstrap and other resampling plans*, volume 38. SIAM, 1982.

Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *International Conference on Learning Representations*, 2018.

Tadayoshi Fushiki. Bootstrap prediction and bayesian prediction under misspecified models. *Bernoulli*, pages 747–758, 2005.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th international conference on artificial intelligence and statistics*, pages 249–256, 2010.

Aditya Gopalan and Shie Mannor. Thompson sampling for learning parameterized Markov decision processes. In *Proceedings of the 28th Annual Conference on Learning Theory*, 2015.

Josef Hadar and William R Russell. Rules for ordering uncertain prospects. *The American Economic Review*, pages 25–34, 1969.

G Hanoch and H Levy. The efficiency analysis of choices involving risk. *The Review of Economic Studies*, 36(3):335–346, 1969.

Imanol Arrieta Ibarra, Bernardo Ramos, and Lars Roemheld. Angrier birds: Bayesian reinforcement learning. *arXiv preprint arXiv:1601.01297*, 2016.

Morteza Ibrahimi, Adel Javanmard, and Benjamin V Roy. Efficient reinforcement learning for high dimensional linear quadratic systems. In *Advances in Neural Information Processing Systems 25*, pages 2636–2644, 2012.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

Sham Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, University College London, 2003.

Michael J. Kearns and Daphne Koller. Efficient reinforcement learning in factored MDPs. In *IJCAI*, pages 740–747, 1999.

Michael J. Kearns and Satinder P. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

Lihong Li and Michael L Littman. Reducing reinforcement learning to kwik online regression. *Annals of Mathematics and Artificial Intelligence*, 58(3-4):217–237, 2010.

Lihong Li, Michael L Littman, and Thomas J Walsh. Knows what it knows: a framework for self-aware learning. In *Proceedings of the 25th international conference on Machine learning*, pages 568–575. ACM, 2008.

Zachary Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. BBQ-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. *AAAI Conference on Artificial Intelligence*, 2018.

Xiuyuan Lu and Benjamin Van Roy. Ensemble sampling. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3258–3266. 2017.

Mark Machina and John Pratt. Increasing risk: some direct constructions. *Journal of Risk and Uncertainty*, 14(2):103–127, 1997.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems 29*, pages 1046–1054, 2016.

Brendan O'Donoghue, Ian Osband, Remi Munos, and Volodymyr Mnih. The uncertainty Bellman equation and exploration. In *Proceedings of the 35th Annual International Conference on Machine Learning*, 2017.

Ronald Ortner and Daniil Ryabko. Online regret bounds for undiscounted continuous reinforcement learning. In *Advances in Neural Information Processing Systems 25*, pages 1763–1771, 2012.

Ian Osband. *Deep Exploration via Randomized Value Functions*. PhD thesis, Stanford University, 2016.

Ian Osband and Benjamin Van Roy. Model-based reinforcement learning and the eluder dimension. In *Advances in Neural Information Processing Systems 27*, pages 1466–1474, 2014a.

Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored MDPs. In *Advances in Neural Information Processing Systems 27*, pages 604–612, 2014b.

Ian Osband and Benjamin Van Roy. Bootstrapped Thompson sampling and deep exploration. *arXiv preprint arXiv:1507.00300*, 2015.

Ian Osband and Benjamin Van Roy. On lower bounds for regret in reinforcement learning. *arXiv preprint arXiv:1608.02732*, 2016.

Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *Proceedings of the 34th International Conference on Machine Learning*, pages 2701–2710, 2017.

Ian Osband, Dnaiel Russo, and Benjamin Van Roy. (More) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems 26*, pages 3003–3011. 2013.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Advances In Neural Information Processing Systems 29*, pages 4026–4034, 2016a.

Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2377–2386, 2016b.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8625–8637. 2018.

Art B Owen and Dean Eckles. Bootstrapping data arrays of arbitrary order. *The Annals of Applied Statistics*, pages 895–927, 2012.

Jason Pazis and Ronald Parr. PAC optimal exploration in continuous space Markov decision processes. In *AAAI Conference on Artificial Intelligence*. Citeseer, 2013.

Matthias Plappert. *Parameter Space Noise for Exploration in Deep Reinforcement Learning*. PhD thesis, Karlsruhe Institute of Technology, 2017.

Warren Powell and Ilya Ryzhov. *Optimal Learning*. John Wiley and Sons, 2011.

Doina Precup, Richard Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *Proceedings of The 18th International Conference on Machine Learning*, pages 417–424, 2001.

Daniel Russo. Worst-case regret bounds for exploration via randomized value functions. *arXiv preprint arXiv:1906.02870*, 2019.

Daniel Russo and Benjamin Van Roy. Eluder dimension and the sample complexity of optimistic exploration. In *Advances in Neural Information Processing Systems 26*, pages 2256–2264. 2013.

Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014a.

Daniel Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. In *Advances in Neural Information Processing Systems 27*, pages 1583–1591. 2014b.

Daniel Russo and James Zou. How much does your data exploration overfit? controlling bias via information usage. *arXiv preprint arXiv:1511.05219*, 2015.

Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on Thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1): 1–96, 2018.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2015.

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.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Alexander L Strehl. *Probably approximately correct (PAC) exploration in reinforcement learning*. PhD thesis, Rutgers University-Graduate School-New Brunswick, 2007.

Alexander L. Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L. Littman. PAC model-free reinforcement learning. In *Proceedings of the 23rd Annual International Conference on Machine Learning*, pages 881–888, 2006.

Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction, Second Edition*. MIT Press, 2018.

Richard Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 993–1000. ACM, 2009.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #Exploration: A study of count-based exploration for deep reinforcement learning. *CoRR*, abs/1611.04717, 2016.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

Gerald Tesauro. Temporal difference learning and TD-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

William 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.

John N Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

Zheng Wen. *Efficient reinforcement learning with value function generalization*. PhD thesis, Stanford University, 2014.

Zheng Wen and Benjamin Van Roy. Efficient exploration and value function generalization in deterministic systems. In *Advances in Neural Information Processing Systems 26*, pages 3021–3029, 2013.

Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 3381–3387. IEEE, 2008.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016.