Deep Exploration via Randomized Value Functions

Ian Osband[1,2], Daniel J. Russo[3], Zheng Wen[4], and Benjamin Van Roy[2]

[1]DeepMind
[2]Stanford University
[3]Northwestern University
[4]Adobe Research

March 23, 2017

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.

1 Introduction

Reinforcement learning holds promise to provide the basis for an artificial intelligence that will manage a wide range of systems and devices to better serve society's needs. 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 tens of 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 here lies in 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 try 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.

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.

In Section 7, we present computational results generated by several reinforcement learning algorithms that use randomized value functions. Results with a simple toy example illustrate dramatic efficiency gains relative to dithering approaches and the synthesis of randomization with generalization via parameterized value functions.

2 Literature review

The Bayes optimal policy serves as a gold standard for exploration in reinforcement learning. In particular, beginning with a prior distribution over Markov decision processes, one can formulate a problem of maximizing expected reward over a prescribed time frame by taking an action at each future time contingent on the prevailing posterior distribution. A policy attaining this maximum must explore judiciously. Unfortunately, for problems of practical interest, computing a Bayes optimal policy is intractable. There is a literature on heuristics that aim to approximate Bayes optimal policies (see [1] for a survey). Randomized value function approaches we introduce in this paper can be viewed as contributing to this literature a practical technique that operates effectively together with value function learning methods commonly used to address large scale reinforcement learning problems and that comes with provable efficiency guarantees in tabular settings.

There is a substantial body of work on provably efficient exploration in tabular reinforcement learning. This begins with the seminal work of Kearns and Singh [2], 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 [3, 4, 5, 6, 7, 8, 9, 10]. An important implication of this literature is that 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., [11, 12]). We discuss this phenomenon further in Section 4.

Despite valuable insights that have been generated through the design and analysis of tabular algorithms, such algorithms are of limited practical import because, due to the curse of dimensionality, state spaces are typically enormous. Practical reinforcement learning algorithms must generalize across states to learn to make effective decisions with limited data, and the literature offers a rich collection of such algorithms (see, e.g., [13, 14, 15, 16] and references therein). Though algorithms of this genre have achieved impressive outcomes, notably in games such as backgammon[17], Atari arcade games [18], and go [19], 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 [19], 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. There is work on model learning algorithms [20, 21, 22, 8, 23, 24, 25, 26], which apply to specific model classes and become statistically or computationally intractable for problems of practical scale. Policy learning algorithms [11, 27, 28] identify high-performers among a set of policies. These lines of work have produced several interesting results, particularly when the space of possible optimal policies is small in some sense. However, each of these existing works either entails overly restrictive assumptions or does 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 [29, 30]. 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) [31], a provably efficient reinforcement learning algorithm for exploration and value function generalization in deterministic systems, and C-PACE [32], 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. Prior reinforcement learning algorithms that generalize in this manner require, in the worst case, learning times exponential in the number of model parameters and/or the planning horizon. The algorithms we propose, which we refer to collectively as randomized least-squares value iteration (RLSVI), aim to overcome these inefficiencies. They operate in a manner similar to well-known approaches such as least-squares value iteration (LSVI) and SARSA (see, e.g., [14]). What fundamentally distinguishes RLSVI is exploration through randomly sampling statistically plausible value functions, whereas 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 other work we know of involving exploration through random sampling of value functions is [33], which proposes a tabular algorithm. A preliminary version of part of this work has also been reported in a short paper [34].

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. 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 [7] can be adapted to the episodic finite-horizon context to produce a $\Omega(H\sqrt{SAL})$ lower bound on expected regret conditioned on the true Markov decision process for any learning algorithm. This differs from our upper bound by a factor of $\sqrt{H}$, though the comparison may not be meaningful, since the lower bound is on a maximum over Markov decision processes and it may not hold for the expectation over Markov decision processes, taken with respect to the distribution we posit.

A very recent thread of work builds on count-based (or upper-confidence-bound-based) exploration schemes that operate with value function learning [35, 36]. These methods maintain a density over the state-action space of pseudo-counts, which represent the quantity of data gathered that is relevant to each state-action pair. Such algorithms may offer a viable approach to deep exploration with generalization. There are, however, some potential drawbacks. One is that a separate representation is required to generalize counts, and its not clear how to design an effective approach to this. As opposed to the optimal value function, which is fixed by the environment, counts are generated by the agent's choices, so there is no single target function to learn. Second, the count model generates reward bonuses that distort data used to fit the value function, so the value function representation needs to be designed to not only capture properties of the true optimal value function but also such distorted versions. Finally, these approaches treat uncertainties as uncoupled across state-action pairs, and this can incur a substantial negative impact on statistical efficiency, as discussed in [37]. That said, there may be a range of problems where count-based schemes prove effective, and some promising computational results are reported in [35, 36].

It is worth noting that our approach is inspired by Thompson sampling [38]. 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 [39, 40, 41, 42, 43, 26]. 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 [42, 43]. On the other hand, for some problems with more complex information structures, it is possible to explore much more efficiently than does Thompson sampling [44]. As such, we should expect that for some reinforcement learning problems and value function representations, the randomized value function approaches we put forth, as well as "well-designed" upper-confidence-bound-based approaches, will leave substantial room for improvement.

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 $s \in \mathcal{S}$ 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 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"

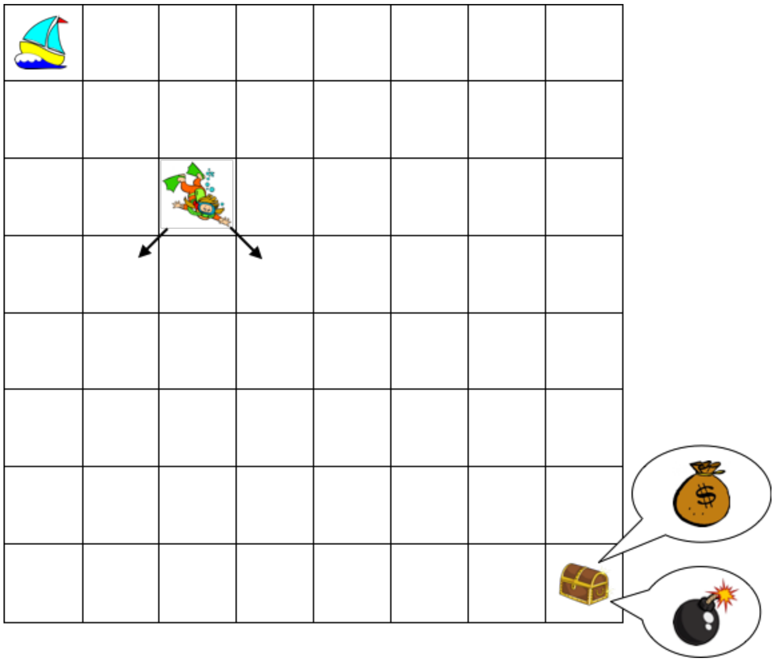


Figure 1: Deep-sea exploration: a simple example where deep exploration is critical.

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 "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 concepts. The first is that of an *optimal state-action value function*, defined by $Q^*_{\mathcal{M}}(s,a) = \mathbb{E}_{\mathcal{M}}\left[r + V^*_{\mathcal{M}}(s')\right]$, 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:

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

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.

As a first reinforcement learning algorithm, we consider a **pure-exploitation** algorithm that aims to maximize expected reward in the current episode, ignoring benefits of active exploration. A pure-exploitation algorithm maximizes the expectation $\hat{Q}_L = \mathbb{E}\left[Q^*_{\mathcal{M}} | \mathcal{H}_{L-1}\right]$, applying in episode L the greedy policy with respect to $\hat{Q}_L$. While this algorithm is applicable to any reinforcement learning problem, its behavior in Example 1 reveals severe inefficiencies. In particular, 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

$$a^L_t \sim \frac{\exp\left(\hat{Q}_L(s^L_t, \cdot)/\eta\right)}{\sum_{a\in\mathcal{A}} \exp\left(\hat{Q}_L(s^L_t, a)/\eta\right)}.$$

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 $\Theta(2^N)$.

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 is that there has been a lack of computationally efficient approaches that adequately address complex reinforcement learning 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 $\hat{Q}(s_t, a_t)$ is large or the observed reward and/or transition are expected to provide useful information. However, an agent applying such an approach, which we refer to as **myopic exploration**, to the problem of Example 1 would once again avoid moving toward the chest once it learns action associations in the initial state. This is because there is a cost to moving right, and once the action associations at that state 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.

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 [13] and Sutton and Barto [14], 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. 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 $\tilde{Q}$ of state-action value functions indexed by a set Θ; each $\theta \in \Theta$ identifies a state-action value function $\tilde{Q}_\theta : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. As a simple example of such a family, consider representing value functions as linear combinations of fixed feature vectors. 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 $\tilde{Q}_\theta(s,a) = \theta^\top \phi(s,a)$, with $\theta \in \Theta = \mathbb{R}^D$.

`live` (Algorithm 1) 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. In addition to an index set and a parametrized family of value functions, it takes as arguments three algorithms. The `cache` algorithm maintains a buffer of observations. The `learn` algorithm produces a value function index $\tilde{\theta} \in \Theta$ based on data in the buffer. The `act` algorithm generates actions based on the estimate. We will consider several versions of each of these algorithms.

The simplest version of `act` generates greedy actions, as expressed by `act_greedy` (Algorithm 2). If `act_greedy` is passed to `learn`, each selected action maximizes over estimated state-action values. If multiple actions attain the maximum, one is sampled uniformly from among them. A basic version of `cache` is given by `cache_infinite` (Algorithm 3), which simply accumulates all observation. Our next three sections present algorithms that es-

Algorithm 1 live

```
Input:  $\Theta$     value function index set
        $\tilde{Q}$     value function family ($\forall \theta \in \Theta,\ \tilde{Q}_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$)
        cache   algorithm for updating memory buffer of observations
        act     algorithm for selecting action given value function
        learn   algorithm for updating value function estimate
buffer $\leftarrow$ cache(null, null)
$\tilde{\theta} \leftarrow$ learn($\tilde{Q}$, buffer, null)
for $\ell$ in $(1, 2, 3, \ldots)$ do
    $t \leftarrow 0$
    observe $s_0^\ell$
    while not terminated do
        $a_t^\ell \leftarrow$ act$\left(\tilde{Q}_{\tilde{\theta}}(s_t^\ell, \cdot)\right)$
        observe $r_{t+1}^\ell$ and $s_{t+1}^\ell$ (let $s_{t+1}^\ell$ = null if terminated)
        buffer $\leftarrow$ cache(buffer, $(s_t^\ell, a_t^\ell, r_{t+1}^\ell, s_{t+1}^\ell)$)
        $t \leftarrow t + 1$
    end while
    $\tilde{\theta} \leftarrow$ learn($\tilde{Q}$, buffer, $\tilde{\theta}$)
end for
```

Algorithm 2 act_greedy

```
Input:  $J$   value function mapping $\mathcal{A}$ to $\mathbb{R}$
return unif_sample$\left(\underset{\alpha \in \mathcal{A}}{\operatorname{argmax}}\ J(\alpha)\right)$
```

timate value function parameters, which can serve as candidates for the `learn` argument required by `live`, as well as a couple variations of `act`.

Algorithm 3 cache_infinite

```
Input:  buffer   memory buffer of observations
        $o$        observation
if buffer = null then
    return new queue
else
    return buffer.enqueue($o$)
end if
```

5.2 Least-squares value iteration

Given an MDP $\mathcal{M}$, one can apply the value iteration algorithm (Algorithm 4) 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 [13], 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 4 `vi`

```
Input:   $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \rho)$   MDP
         $H \in \mathbb{Z}_{++}$                                                 planning horizon
Output:  $Q^*_H$                                                                optimal value function for $H$-period problem
$Q^*_0 \leftarrow 0$
for $h$ in $(0, \dots, H-1)$ do
    Compute value function for $(h+1)$-period problem
    $$Q^*_{h+1}(s,a) \leftarrow \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,a}(s') \left( \int r \mathcal{R}_{s,a,s'}(dr) + \max_{a' \in \mathcal{A}} Q^*_h(s', a') \right)$$
end for
return $Q^*_H$
```

`learn_lsvi` (Algorithm 5) approximates operations carried out by value iteration. The algorithm operates with a family $\tilde{Q}$ of value functions indexed by $\Theta = \mathbb{R}^D$. Instead of the MDP realization, `learn_lsvi` takes as input the data buffer, which is used to fit the value function. Also taken as input are regularization parameters: $\overline{\theta}$ and λ can be interpreted as the "prior mean and variance," or the expectation and degree of uncertainty in θ, while v can be interpreted as "noise variance," or the degree to which value estimates based on individual transitions differ from their expectations.

Similarly to `vi`, `learn_lsvi` computes a sequence of value functions $(\tilde{Q}_{\theta_h} : h = 0, \dots, H)$, 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 $\max_{a' \in \mathcal{A}} \tilde{Q}_{\tilde{\theta}_h}(s', a')$ at the next state onto the value estimate $\tilde{Q}_{\tilde{\theta}_{h+1}}(s, a)$ for the current state-action pair.

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 $\overline{\theta}$ 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 $Q_{\tilde{\theta}_H}$ 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, $Q_{\tilde{\theta}_H}$ 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

Algorithm 5 `learn_lsvi`

```
Input:   $\tilde{Q}$                    value function family ($\forall\theta \in \mathbb{R}^D,\ \tilde{Q}_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$)
         buffer                         memory buffer of observations
         $\tilde{\theta}$               previous value function parameters
         $\overline{\theta} \in \mathbb{R}^D$    regularization parameter (prior mean)
         $\lambda \in \mathbb{R}_{++}$   regularization parameter (prior variance)
         $v \in \mathbb{R}_{++}$         regularization parameter (noise variance)
         $H \in \mathbb{Z}_{++}$         planning horizon
Output:  $\tilde{\theta}$               updated value function parameters

$\tilde{\theta}_0 \leftarrow$ null
for $h$ in $(0, \dots, H-1)$ do
    Regress
        $$\tilde{\theta}_{h+1} \leftarrow \underset{\theta \in \mathbb{R}^D}{\operatorname{argmin}} \left( \frac{1}{v} \sum_{(s,a,r,s') \in \texttt{buffer}} \left( r + \max_{a' \in \mathcal{A}} \tilde{Q}_{\tilde{\theta}_h}(s', a') - \tilde{Q}_\theta(s,a) \right)^2 + \frac{1}{\lambda} \|\theta - \overline{\theta}\|^2 \right)$$
    (let $\tilde{Q}.(\textbf{null}, \cdot) = 0$ and $\tilde{Q}_{\textbf{null}}(\cdot,\cdot) = 0$ )
end for
return $\tilde{\theta}_H$
```

of the parameter vector $\tilde{\theta}$, the number of historical observations, and the time required to maximize over actions at any given state.

Applying `live` with `learn_lsvi`$(\cdot, \cdot, \overline{\theta}, \lambda, \nu, H)$ and `act_greedy` as arguments gives rise to a simple reinforcement learning system. Such a system may work reasonably for problems that do not require active exploration. However, lack of exploration typically hinders the agent from discovering high-value policies. `act_boltzmann_greedy` (Algorithms 6) and `act_epsilon_greedy` (7) induce dithering exploration by randomly perturbing greedy actions. Each takes an additional parameter as input to control the intensity of this randomness: `act_boltzmann_greedy` takes a temperature parameter η and `act_epsilon_greedy` takes an exploration probability ϵ. Passing `act_boltzmann_greedy`$(\cdot, \eta)$ or `act_epsilon_greedy`$(\cdot, \epsilon)$ to `learn` results in Botlzmann or ϵ-greedy exploration.

Algorithm 6 `act_boltzmann_greedy`

```
Input:   $J$      value function mapping $\mathcal{A}$ to $\mathbb{R}$
         $\eta$   Boltzmann temperature

return multinomial_sample$\left( \frac{\exp(J(\cdot)/\eta)}{\sum_{a \in \mathcal{A}} \exp(J(a)/\eta)} \right)$
```

5.3 Randomized least-squares value iteration

As discussed in Section 4, randomly perturbing greedy actions – or dithering – does not achieve deep exploration. In this section, we consider randomized value function estimates as an alternative. At a high level, the idea is to randomly sample from among statistically

Algorithm 7 `act_epsilon_greedy`

```
Input: $J$  value function mapping $\mathcal{A}$ to $\mathbb{R}$
       $\epsilon$  exploration intensity parameter
if unif_sample($[0,1]$) $\le \epsilon$ then
    return unif_sample($\mathcal{A}$)
else
    return unif_sample$\left(\operatorname{argmax}_{\alpha \in \mathcal{A}} J(\alpha)\right)$
end if
```

plausible parameter vectors. This approach is inspired by Thompson sampling, an algorithm widely used in bandit learning. 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 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 achieve 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 $N(\overline{\theta}, \lambda I)$ prior and data $\mathcal{D} = ((x_n, y_n) : n = 1, \ldots, N)$, 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

$$\mathbb{E}[\theta|\mathcal{D}] = \operatorname*{argmin}_{\theta \in \mathbb{R}^D} \left(\frac{1}{v} \sum_{n=1}^{N} (y_n - x_n \theta)^2 + \frac{1}{\lambda} \|\overline{\theta} - \theta\|^2 \right) = \left(\frac{1}{v} X^\top X + \frac{1}{\lambda} I \right)^{-1} \left(\frac{1}{v} X^\top y + \frac{1}{\lambda} \overline{\theta} \right) \tag{5.1}$$

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

$$\tilde{\theta} \leftarrow \underset{\theta\in\mathbb{R}^D}{\operatorname{argmin}}\left(\frac{1}{v}\sum_{n=1}^{N}(y_n+z_n-x_n\theta)^2+\frac{1}{\lambda}\|\hat{\theta}-\theta\|^2\right)=\left(\frac{1}{v}X^\top X+\frac{1}{\lambda}I\right)^{-1}\left(\frac{1}{v}X^\top(y+z)+\frac{1}{\lambda}\hat{\theta}\right), \tag{5.2}$$

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

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

Suppose $\tilde{Q}_\theta(s,a) = \theta^\top\phi(s,a)$, where $\phi : \mathcal{S}\times\mathcal{A} \mapsto \mathbb{R}^D$ extracts a row vector of features from each state-action pair. Given this parameterized value function, Line 3 of `learn_lsvi` (Algorithm 5) carries out a linear regression, using the formula on the right-hand-side of Equation 5.1, with target values $y_n = r + \max_{a'\in\mathcal{A}}\tilde{Q}_{\tilde{\theta}_h}(s',a')$, for each nth observation (s,a,r,s') in `buffer`. Line 9 of `learn_grlsvi` (Algorithm 8) is similar, but uses a target of $y_n = r + \max_{a'\in\mathcal{A}}\tilde{Q}_{\tilde{\theta}_h}(s',a') + z_n$ and regularizes toward $\hat{\theta}$ rather than the origin. As such, $\tilde{Q}_{\tilde{\theta}_{h+1}}$ can be thought of as a statistically plausible random sample of Q^*_{h+1}, conditioned on $Q^*_h = \tilde{Q}_{\tilde{\theta}_h}$ and the buffered observations. An informal inductive argument then suggests that $\tilde{Q}_{\tilde{\theta}_H}$ can be thought of as a statistically plausible random sample of Q^*_H conditioned only on buffered observations.

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.

Algorithm 8 `learn_grlsvi`

```
Input:  $\tilde{Q}$                     value function family ($\forall \theta \in \mathbb{R}^D,\ \tilde{Q}_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$)
        buffer                          memory buffer of observations
        $\tilde{\theta}$                previous value function parameters
        $\overline{\theta} \in \mathbb{R}^D$    regularization parameter (prior mean)
        $\lambda \in \mathbb{R}_{++}$   regularization parameter (prior variance)
        $v \in \mathbb{R}_{++}$         regularization parameter (noise variance)
        $H \in \mathbb{Z}_{++}$         planning horizon
Output: $\tilde{\theta}$                updated value function parameters
$\tilde{\theta}_0 \leftarrow$ null
Sample from prior: $\hat{\theta} \sim N(\overline{\theta}, \lambda I)$
buffer_noise ← cache_infinite(null, null)
for $(s, a, r, s') \in$ buffer do
    Sample noise: $z \sim N(0, v)$
    buffer_noise ← cache_infinite(buffer_noise, $(s, a, r, s', z)$)
end for
for $h$ in $(0, \ldots, H-1)$ do
    Regress
    $\tilde{\theta}_{h+1} \leftarrow \underset{\theta \in \mathbb{R}^D}{\operatorname{argmin}} \left( \frac{1}{v} \sum_{(s,a,r,s',z) \in \texttt{buffer\_noise}} \left( r + \max_{a' \in \mathcal{A}} \tilde{Q}_{\tilde{\theta}_h}(s', a') + z - \tilde{Q}_\theta(s, a) \right)^2 + \frac{1}{\lambda} \|\hat{\theta} - \theta\|^2 \right)$
    (let $\tilde{Q}_\cdot(\mathbf{null}, \cdot) = 0$ and $\tilde{Q}_{\mathbf{null}}(\cdot, \cdot) = 0$ )
end for
return $\tilde{\theta}_H$
```

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 $(4, \textit{down})$, where we are highly uncertain and any transition is plausible. The shade of 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 $Q^*_h(s, a)$.

For the case of $h = 1$, only immediate rewards influence Q^*_1, and as such we are only uncertain about $Q^*_1(4, \textit{down})$. Stepping back to $h = 2$, in addition to being highly uncertain about $Q^*_2(4, \textit{down})$, we are somewhat uncertain about $Q^*_2(4, \textit{up})$ and $Q^*_2(3, \textit{down})$, since these pairs can transition to $(4, \textit{down})$ and be exposed to the uncertainty associated with that state-action pair. We are not as uncertain about $Q^*_2(4, \textit{up})$ and $Q^*_2(3, \textit{down})$ as we are about $Q^*_2(4, \textit{down})$ because from $(4, \textit{up})$ and $(3, \textit{down})$, there is reasonable chance that we will never see $(4, \textit{down})$. Continuing to work our way leftward in the diagram, it is easy to visualize how uncertainty propagates as h increases.

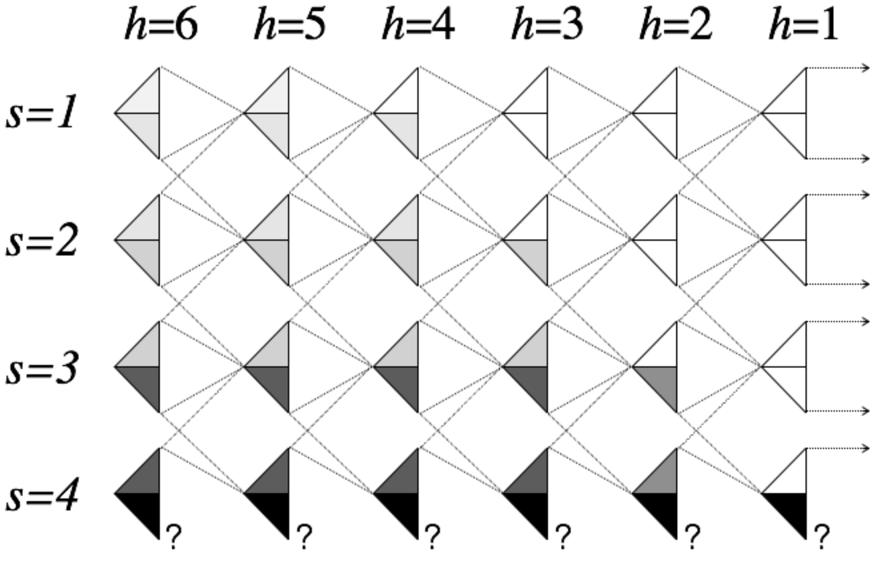


Figure 2: Illustration of how `learn_grlsvi` achieves deep exploration.

Let us now turn our attention to the variance of samples $\tilde{Q}_{\tilde{\theta}_H}(s,a)$ 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 $\tilde{Q}_\theta(s,a)$ 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

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

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, $\tilde{Q}_{\tilde{\theta}_{h+1}}(s,a)$ is essentially equal to $\mathbb{E}_{\mathcal{M}}[r_{t+1} + \max_{\alpha\in\mathcal{A}} \tilde{Q}_{\tilde{\theta}_h}(s_{t+1},\alpha)|s_t = s, a_t = a]$. 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 $\mathbb{E}[r + \max_{a'\in\mathcal{A}} \tilde{Q}_{\tilde{\theta}_h}(s',a')|\tilde{\theta}_h, \mathcal{M}]$. 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 $\tilde{Q}_{\tilde{\theta}_{h+1}}(4, down)$.

Based on this observation, for the case of $h = 1$, for $(s,a) \neq (4, down)$, $\tilde{Q}_{\tilde{\theta}_1}(s,a)$ is virtually equal to Q^*_1, while for $\tilde{Q}_{\tilde{\theta}_1}(4, down)$ exhibits variance of at least v. For $h = 2$, $\tilde{Q}_{\tilde{\theta}_2}(4, down)$ again exhibits variance of at least v, but unlike the case of $h = 1$, $\tilde{Q}_{\tilde{\theta}_2}(4, up)$ and $\tilde{Q}_{\tilde{\theta}_2}(3, down)$ also exhibit non-negligible variance since these pairs can transition to $(4, down)$ and therefore depend on the noise-corrupted realization of $\tilde{Q}_{\tilde{\theta}_1}(4, down)$. 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 $\tilde{Q}_{\tilde{\theta}_h}(s,a)$ exhibit high variance where the variance of $Q^*_h(s,a)$ is large.

This relationship drives deep exploration. In particular, a high variance sample $\tilde{Q}_{\tilde{\theta}_H}(s,a)$ 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 `gaussian_rlsvi`, the output $\tilde{\theta}_H$ is not Gaussian. This is because θ_{h+1} depends nonlinearly on θ_h due to the maximization over actions in Line 9 of the algorithm. Second, it is important that `learn_grlsvi` uses the same noise samples z in across iterations of the for loop of Line 8. 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 $(4, \mathit{down})$, while in other iterations, we would be pessimistic about that. Now consider a sample $\tilde{Q}_{\tilde{\theta}_H}(1, \mathit{up})$ for large H. This sample would be perturbed by a combination of optimistic and pessimistic noise terms influencing assessments at $(4, \mathit{down})$ to the right. The resulting averaging effect could erode the chances that $\tilde{Q}_{\tilde{\theta}_H}(1, \mathit{up})$ 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 $r + \max_{a' \in \mathcal{A}} \tilde{Q}_{\tilde{\theta}_h}(s', a')$ and expectations $\mathbb{E}[r + \max_{a' \in \mathcal{A}} \tilde{Q}_{\tilde{\theta}_h}(s', a') | \tilde{\theta}_h, \mathcal{M}]$. The statistical bootstrap[1] offers an alternative approach to randomization which may often more accurately capture characteristics of the generating process.

`learn_brlsvi` (Algorithm 9) is a version of RLSVI that makes use of the bootstrap. As with `learn_grlsvi`, a nominal parameter vector $\hat{\theta}$ is sampled from $N(\overline{\theta}, \lambda I)$. The difference comes with the random sampling of `buffer_boot`, which includes as many elements as `buffer`, each sampled uniformly from `buffer` (with replacement). The randomness induced by this sampling procedure substitutes for the randomness induced by Gaussian noise in `learn_grlsvi`.

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 terms 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.

One could consider many variations of `learn_brlsvi`. For starters, there are many

[1] 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 [45].

Algorithm 9 `learn_brlsvi`

```
Input:   $\tilde{Q}$                      value function family ($\forall \theta \in \mathbb{R}^D,\ \tilde{Q}_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$)
         buffer                           memory buffer of observations
         $\tilde{\theta}$                 previous value function parameters
         $\overline{\theta} \in \mathbb{R}^D$    regularization parameter (prior mean)
         $\lambda \in \mathbb{R}_{++}$    regularization parameter (prior variance)
         $v \in \mathbb{R}_{++}$          regularization parameter (noise variance)
         $H \in \mathbb{Z}_{++}$          planning horizon
Output:  $\tilde{\theta}$                 updated value function parameters

$\tilde{\theta}_0 \leftarrow$ null
Sample from prior: $\hat{\theta} \sim N(\overline{\theta}, \lambda I)$
buffer_boot ← bootstrap_sample(buffer)
for $h$ in $(0, \ldots, H-1)$ do
    Regress
    $$\tilde{\theta}_{h+1} \leftarrow \underset{\theta \in \mathbb{R}^D}{\operatorname{argmin}} \left( \frac{1}{v} \sum_{(s,a,r,s') \in \texttt{buffer\_boot}} \left( r + \max_{a' \in \mathcal{A}} \tilde{Q}_{\tilde{\theta}_h}(s, a') - \tilde{Q}_\theta(s, a) \right)^2 + \frac{1}{\lambda} \|\hat{\theta} - \theta\|^2 \right)$$
    (let $\tilde{Q}_\cdot(\textbf{null}, \cdot) = 0$ and $\tilde{Q}_{\textbf{null}}(\cdot, \cdot) = 0$ )
end for
return $\tilde{\theta}_H$
```

variations of the statistical bootstrap that can be brought to bear. We could use the Bayesian bootstrap [46], for instance, instead of the standard statistical bootstrap. This would involve generating an independent identically distributed variable z_o for each $o \in$ `buffer` and regressing according to

$$\tilde{\theta}_{h+1} \leftarrow \underset{\theta \in \mathbb{R}^D}{\operatorname{argmin}} \left(\frac{1}{v} \sum_{(s,a,r,s') \in \texttt{buffer}} z_{(s,a,r,s')} \left(r + \max_{a' \in \mathcal{A}} \tilde{Q}_{\tilde{\theta}_h}(s, a') - \tilde{Q}_\theta(s, a) \right)^2 + \frac{1}{\lambda} \|\hat{\theta} - \theta\|^2 \right)$$

It is sometimes also useful to base the regularization term on a statistical bootstrap. Suppose, for example, we obtain or generate a buffer of data samples – `buffer_prior` – before initiating `live`. As an alternative to the regularization penalty $\psi(\theta) = \|\hat{\theta} - \theta\|^2 / \lambda$, we could generate a random bootstrap sample `buffer_prior_boot`, and apply a regularization penalty of the form

$$\psi(\theta) = \frac{1}{\lambda} \sum_{(s,a,r,s') \in \texttt{buffer\_prior\_boot}} \left(r + \max_{a' \in \mathcal{A}} \tilde{Q}_{\tilde{\theta}_h}(s, a') - \tilde{Q}_\theta(s, a) \right)^2 .$$

Here, the bootstrap sample induces randomization of the prior penalty, substituting for the effect of a randomly sampled $\hat{\theta}$. A similar approach to regularization could be based on a buffer of data pairs of the form (s, a, y), where y represents a plausible value of $Q^*(s, a)$. Given such a buffer, we can generate bootstrap sample and apply a regularization penalty of the form

$$\psi(\theta) = \frac{1}{\lambda} \sum_{(s,a,y) \in \texttt{buffer\_prior\_boot}} \left(y - \tilde{Q}_\theta(s, a) \right)^2 .$$

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. This is accomplished by using `cache_finite` (Algorithm 10) instead of `cache_infinite` (Algorithm 3) in `live` (Algorithm 1).

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 [47]. This is likely due to model mis-specification. In particular, if Q^* can not be represented by $\tilde{Q}_\theta$ 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 [48].

Algorithm 10 `cache_finite`

```
Input:  buffer   memory buffer of observations
        $o$        observation
        $N$        cache size
if buffer = null then
    return new finite_queue($N$)
else
    return buffer.enqueue($o$)
end if
```

5.4.2 Temporal-difference learning

Even with a finite buffer, when the value function is parameterized by a high-dimensional vector θ, enormous computational resources can be required to produce an estimate $\tilde{\theta}$. In such contexts, there can be substantial benefit to using first-order algorithms in the vein of stochastic gradient descent. Temporal-difference learning (TD) offers such an approach as a proxy to RLSVI. `learn_td` (Algorithm 11) is such an algorithm. In addition to $\tilde{Q}$, `buffer`, and an initial $\tilde{\theta}$, the algorithm takes several inputs, most of which relate to inputs of `learn_lsvi` (Algorithm 5):

- `sample`: an algorithm that samples a minibatch from the buffer
- γ: a discount factor that plays a role analogous to the horizon parameter H in `learn_lsvi`
- ψ: a regularization penalty function; for `learn_lsvi`, this was fixed to $\psi(\theta) = \|\theta - \overline{\theta}\|^2/\lambda$
- v: a regularization parameter that represents noise variance, as in `learn_lsvi`
- α: a learning rate parameter that determines the step size of TD updates
- N: the number of minibatches to sample, which is the same as the number of TD updates applied

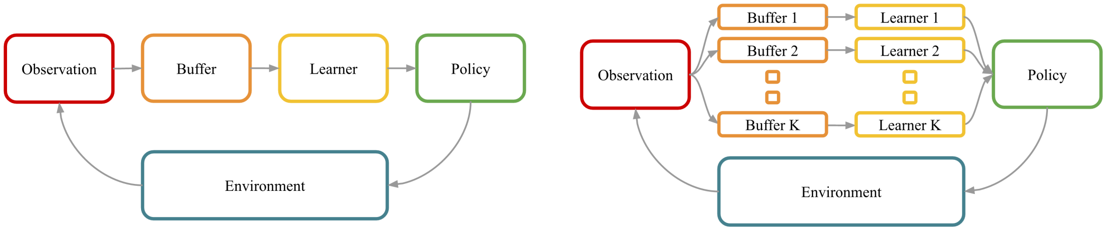
As opposed to `learn_lsvi`, `learn_td` makes use of the previously computed parameter vector $\tilde{\theta}$. This offers a "hot start" that could reduce the number of updates N required to arrive at a reasonable fit to the buffered data if the previous parameter vector was fit to a similar data set. The combination of `learn_td` and `cache_finite` with a neural network value function representation is the essence of "deep Q networks" [49], which have achieved notable success in producing high-performing strategies for Atari arcade games.

Algorithm 11 `learn_td`

```
Input:   $\tilde{Q}$                      value function family ($\forall \theta \in \mathbb{R}^D$, $\tilde{Q}_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$)
         buffer                   memory buffer of observations
         $\tilde{\theta} \in \mathbb{R}^D$             previous value function parameters
         sample                   method for sampling experience from buffer
         $\gamma \in [0,1]$            discount factor
         $\psi$                      regularization penalty function
         $v \in \mathbb{R}_{++}$              regularization parameter (noise variance)
         $\alpha \in \mathbb{R}_{++}$              learning rate
         $N \in \mathbb{N}$                number of minibatches
Output:  $\tilde{\theta}$                      updated value function parameters
for $n$ in $(1, \ldots, N)$ do
    batch ← sample(buffer)
    define loss function
        $\mathcal{L}(\theta) \equiv \frac{1}{v} \sum_{(s,a,r,s') \in \mathtt{batch}} \left( r + \gamma \max_{a' \in \mathcal{A}} \tilde{Q}_{\tilde{\theta}}(s', a') - \tilde{Q}_\theta(s,a) \right)^2 + \psi(\theta)$
    apply TD update
        $\tilde{\theta} \leftarrow \tilde{\theta} - \alpha \nabla_\theta \mathcal{L}(\tilde{\theta})$
end for
return $\tilde{\theta}$
```

5.4.3 RLSVI via parallelization

`learn_td` (Algorithm 11) offers a finite-buffer first-order variation of LSVI. We now consider such a variation of RLSVI, designed to carry out deep exploration. Our approach involves learning K value functions in parallel, each from a different buffer of randomly perturbed samples. When viewed as an ensemble, the K value functions act as an approximation to the distribution over value functions induced by RLSVI. In each episode ℓ we select one of these value function with a random index $k \sim \text{unif}(\{1,..,K\})$ and follow the resulting greedy policy. We present a high level illustration of this design in Figure 3, which we will make concrete in the remainder of this section.



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

Figure 3: RLSVI via K parallel value functions, each produced by LSVI applied to perturbed data.

We will make use of `cache_parallel_gauss` to represent a K parallel buffers. Each time an observation is incorporated, it is enqueued to each of the K buffers, but in each case with the reward perturbed by a different independent Gaussian noise term. Each of these buffers plays the role of a perturbed data set that could be fit to produce a single randomized value function. Given this set of parallel buffers, `learn_parallel_rlsvi` (Algorithm 13) produces K parameter vectors $\tilde{\theta}^1, \ldots, \tilde{\theta}^K$ and a random index $\tilde{k}$. Given this parallel structure, the value function produced is taken to be parameterized by $\tilde{\theta} = (\tilde{\theta}^1, .., \tilde{\theta}^K, \tilde{k})$, and with some abuse of notation, can be written as $\tilde{Q}_{\tilde{\theta}} = \tilde{Q}_{\tilde{\theta}^{\tilde{k}}}$.

To apply our parallel version of RLSVI we would invoke `live` (Algorithm 1) with:

- `cache` = `cache_parallel_gauss`$(\cdot, \cdot, v, K, N)$
- `act` = `act_greedy`
- `learn` = `learn_parallel_rlsvi`$(\cdot, \cdot,$ `learn_internal`$)$
- `learn_internal`$[k]$ = `learn_td`$(\cdot, \cdot, \cdot,$ `sample`$, \gamma, \psi_k, v, \alpha, N)$
- `sample(buffer)` = [`buffer`$[m]$ for m in `random_choice`$(M,$ `buffer.length`$)$]

Note that `random_choice`(M, N) selects a random subset of M elements of $1, \ldots, N$. Hence, `sample` simply samples a random minibatch of size M. Each ψ_k should be a random regularization penalty function, and other parameters are fixed and selected for the problem at hand.

The regularization penalty functions ψ_k can play an important role in exploration. These penalty functions reflect prior uncertainty and guide initial exploration as observations are accumulated. Perhaps the simplest way to generate these functions is to proceed in a manner similar to `learn_grlsvi`, and sample independent parameter vectors $\hat{\theta}^1, \ldots, \hat{\theta}^K$ from a $N(\overline{\theta}, \lambda I)$ prior, setting $\psi_k(\theta^k) = \|\hat{\theta}^k - \theta^k\|^2/\lambda$. Alternatively, as discussed in the end

Algorithm 12 `cache_parallel_gauss`

```
Input:  buffer       memory buffer of observations
        $(s, a, r, s')$   observation
        $v$          noise variance
        $K$          number of parallel caches
        $N$          size of each parallel cache
if buffer = null then
    return [new finite_queue($N$) for $k$ in $1, \dots, K$]
else
    for $k$ in $1, \dots, K$ do
        Sample noise $z \sim N(0, v)$
        buffer[k].enqueue($(s, a, r + z, s')$)
    end for
    return buffer
end if
```

Algorithm 13 `learn_parallel_rlsvi`

```
Input:   $\tilde{Q}$                     value function family ($\forall \theta \in \mathbb{R}^D,\ \tilde{Q}_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$)
         buffer                parallel memory buffer of observations
         $\tilde{\theta} = (\tilde{\theta}^1, .., \tilde{\theta}^K, \tilde{k})$   parallel previous parameters with selection
         learn_internal        parallel internal learning routines
Output:  $\tilde{\theta}$                     updated value function parameters
$\tilde{\theta}^k \leftarrow$ learn_internal[$k$]($\tilde{Q}$, buffer[$k$], $\tilde{\theta}^k$) for $k$ in $1, .., K$
Sample $\tilde{k} \sim \text{unif}(1, \dots, K)$
return $\tilde{\theta}$
```

of Section 5.3.3, regularization penalties can be produced based on previously gathered or synthetic data. This approach could leverage, for example, data from observing operation of a system under an arbitrary policy or while performing a task with a different objective, as studied in the areas of as *off-policy learning* [50] and *transfer learning* [51].

We note that, when used off-policy, the `learn_td` update may lead to unstable learning and even cause the value function estimate to diverge [52]. This instability can be exacerbated by algorithms such as `learn_parallel_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 `learn_td` update with an alternative designed for off-policy learning [53, 54].

The algorithm we have described induces randomness in value functions through perturbing observations with Gaussian noise, similarly with `learn_grlsvi` (Algorithm 8). It is also possible to design a variation that leverages the statistical bootstrap, as does `learn_brlsvi` (Algorithm 9). This can be done by designing an alternative to `cache_parallel_gauss` that randomizes the number of copies of each observation placed in each of the parallel buffers [55, 56, 57].

6 Regret bound

This section provides a regret analysis of RLSVI in a particularly simple special case of the general problem of Section 3. The version of RLSVI we consider involves application of `live` (Algorithm 1) invoked with `cache_infinite` (Algorithm 3), `act_greedy` (Algorithm 2), and `learn_grlsvi` (Algorithm 8). 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 $\mathcal{S}_0, ..., \mathcal{S}_{H-1}$ contains an equal number of elements. This 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-inhomegenous 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

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

and define $V^*_{\mathcal{M},t}(x) := \max_\pi V^{\pi}_{\mathcal{M},t}(x)$. 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

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

for $\alpha_{0,t,x,a} \in \mathbb{R}_+^{2|\mathcal{X}|}$ and each $\mathcal{P}^O_{t,x,a}$ is drawn independently across (t,x,a). Assume there is $\beta \geq 2$ 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 a version of RLSVI that consists of `live` *(Algorithm 1) invoked with* `cache_infinite` *(Algorithm 3),* `act_greedy` *(Algorithm 2), and* `learn_grlsvi` *(Algorithm 8). Under Assumption 3 with $\beta \geq 2$, if this version of RLSVI is applied with planning horizon H, and parameters $v = H^2$, $\bar{\theta} = H\mathbb{1}$ and $v/\lambda = \beta$, then for all $L \in \mathbb{N}$,*

$$\text{(6.1)} \quad \mathrm{BayesRegret}(\mathrm{RLSVI}_{\bar{\theta},v,\lambda}, L) \leq 4H^2\sqrt{\beta|\mathcal{X}||\mathcal{A}|L\log_+(1+|\mathcal{X}||\mathcal{A}|HL)}\log_+\left(1+\frac{L}{|\mathcal{X}||\mathcal{A}|}\right),$$

and

$$\text{(6.2)} \quad \begin{aligned}\mathrm{BayesRegret}(\mathrm{RLSVI}_{\bar{\theta},v,\lambda}, L) \quad \leq \quad & 4\beta H^3|\mathcal{X}|||\mathcal{A}|\sqrt{\log_+(1+|\mathcal{X}||\mathcal{A}|HL)}\log\left(1+\frac{L}{|\mathcal{X}||\mathcal{A}|}\right) \\ & +2H^2\sqrt{2|\mathcal{X}||\mathcal{A}|L\log(|\mathcal{X}||\mathcal{A}|)}\end{aligned}$$

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 the 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,

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

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 [6] and [7] 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 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 1 (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 $\hat{P}^O_{\ell,y}(r', x')$ 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

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

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

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

where

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

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

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

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

$$\begin{aligned} F_{\ell,t}Q(x,a) &:= \sigma^2_\ell(t,x,a)\left(\frac{\bar{\theta}_{t,x,a}}{\lambda} + \frac{1}{v}\left(\sum_{\substack{(r,x')\in \\ D_{\ell-1}(t,x,a)}} r + \max_{a' \in \mathcal{A}} Q(x',a')\right)\right) + w_\ell(t,x,a) \\ \sigma^2_\ell(t,x,a) &= \left(\frac{1}{\lambda} + \frac{n_\ell(t,x,a)}{v}\right)^{-1} = \frac{v}{n_\ell(t,x,a) + v/\lambda} \\ w_\ell(t,x,a) \mid \mathcal{H}_{\ell-1} &\sim N(0, \sigma^2_\ell(t,x,a)) \end{aligned}$$

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 $\hat{P}^O_{\ell,y}$ over past outcomes resulting from $y = (t, x, a)$. We have

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

Direct calculation gives the following alternate expression

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

This shows that the Bellman update of RLSVI differs from the empirical Bellman update $V_Q^T \hat{\mathcal{P}}^O_{\ell,y}$ 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 1 (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 1. *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 1 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 1 is a powerful tool for studying the regret of optimistic algorithms. The regret of the policy π in Lemma 1 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 1. 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:

$$
\begin{aligned}
V^*_{\mathcal{M},0}(x) - V^{\pi}_{\mathcal{M},0}(x) = &\quad \max_{a\in\mathcal{A}} Q^*_{\mathcal{M},0}(x,a) - \max_{a\in\mathcal{A}} Q_0(x,a) \qquad \text{(pessimism of } Q_0\text{)} \\
(6.5) \qquad + &\quad \mathbb{E}_{\mathcal{M},\pi}\left[\textstyle\sum_{t=0}^{H-1}(Q_t - F_{\mathcal{M},t}Q_{t+1})(x_t,a_t) \mid x_0 = x\right] \quad \text{(on policy Bellman error).}
\end{aligned}
$$

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 $(Q_0,\dots,Q_H)$.

One can apply this regret decomposition to study RLSVI by taking $(Q_0,\dots,Q_H)$ to be the sequence $(Q_{\ell,0},\dots,Q_{\ell,H})$ 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 1 then follows by taking exceptions on both sides of equation (6.5).

Corollary 1 (Optimistic regret bounds)**.**
For any episode $\ell \in \mathbb{N}$, if

$$
\mathbb{E}\left[\max_{a\in\mathcal{A}} Q_{\ell,0}(x_0^\ell,a)\right] \geq \mathbb{E}\left[\max_{a\in\mathcal{A}} Q^*_{\mathcal{M},0}(x_0^\ell,a)\right] \tag{6.6}
$$

then

$$
\mathbb{E}\left[V^*_{\mathcal{M},0}(x_0^\ell) - V^{\pi_\ell}_{\mathcal{M},0}(x_0^\ell)\right] \leq \mathbb{E}\left[\sum_{t=0}^{H}(F_{\ell,t}Q_{\ell,t+1} - F_{\mathcal{M},t}Q_{\ell,t+1})(x_t^\ell,a_t^\ell)\right].
$$

Corollary 1 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 2 (Stochastic optimism).
A random variable X is stochastically optimistic with respect to another random variable Y, written $X \geq_{SO} Y$, if for all convex increasing functions $u : \mathbb{R} \to \mathbb{R}$

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

This definition closely mirrors that of "second order stochastic dominance", which is widely used in decision theory [58]. 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 $X \geq_{SO} Y$ 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 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 $X \geq_{SO} Y$ 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 2 (Preservation of optimism under convex operations).
For any two collections $(X_1, ..., X_n)$ and $(Y_1, ..., Y_n)$ of independent random variables with $X_i \geq_{SO} Y_i$ for each $i \in \{1, ...n\}$ and any convex increasing function $f : \mathbb{R}^n \to \mathbb{R}$,

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

Two special cases of Lemma 2 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[2] $X + Z \geq_{SO} Y + Z$. 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 3 (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

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

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

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 2. □

Consider the random variable $Y = P^TV$ 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 pesudocount $\mathbb{1}^T\alpha = \sum_{i=1}^n \alpha_i$, 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 with large enough mean and and variance is stochastically optimistic with respect to Y. This result is established in a separate technical note [59].

Lemma 4 (Gaussian vs Dirichlet optimism).
Let $Y = P^TV$ for $V \in \mathbb{R}^n$ fixed and $P \sim \text{Dirichlet}(\alpha)$ with $\alpha \in \mathbb{R}^n_+$ and $\sum_{i=1}^n \alpha_i \geq 2$. Let $X \sim N(\mu,\sigma^2)$ with $\mu \geq \frac{\sum_{i=1}^n \alpha_i V_i}{\sum_{i=1}^n \alpha_i}$, $\sigma^2 \geq (\sum_{i=1}^n \alpha_i)^{-1}\,\text{Span}(V)^2$, then $X \geq_{SO} Y$.

With Lemma 4 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 1, which defines the value over outcomes (r,x') under Q by $V_Q(r,x') \equiv r + \max_{a'\in\mathcal{A}} Q(x',a')$.

Lemma 5 (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 $\text{Span}(V_Q)^2 \leq v$ and $\max_{x\in\mathcal{X}} V_Q(x) \leq \min_{t,x,a}\bar{\theta}_{t,x,a}$.

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

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

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

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

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

The result follows from Lemma 4 if we establish $\sigma_y^2 \geq (\mathbb{1}^T \alpha_{\ell,y})^{-1}\text{Span}(V_Q)^2$ and $\mu_y \geq V_Q^T \alpha_{\ell,y}/\mathbb{1}^T\alpha_{\ell,y}$. We have

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

because of the assumption that $v \geq \text{Span}(V_Q)^2$. 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. □

Lemmas 3 and 5 together imply the stochastic optimism of the state-action value functions $Q_{\ell,0}$ generated by RLSVI.

Corollary 2. *If Assumption 3 holds and RLSVI is applied with parameters $(\bar{\theta}, v, \lambda)$ satisfying $(v/\lambda) = \beta$, $v \geq H^2$ 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 5,

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

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

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

Combining this with Lemma 3 shows

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

where the final step uses Lemma 5 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 $\text{Span}(V_Q) \leq H \leq v$ 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 slight 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 [60].

Lemma 6. *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 3. *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 4. *If RLSVI is applied with parameters $(\lambda, v, \bar{\theta})$ with $v/\lambda = \beta \geq 2$, $v = H^2$ and $\bar{\theta} = H\mathbb{1}$,*

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

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

Proof. Set

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

By Corollary 1 and Corollary 2,

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

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

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

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,

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

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

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

Each term can be bounded separately. By Corollary 3

$$\begin{aligned}
\mathbb{E}\sum_{\ell\le L, t<H} w_\ell(t, x_t^\ell, a_t^\ell) \le \mathbb{E}\sum_{\ell\le L, t<H}\sigma_\ell(t, x_t^\ell, a_t^\ell)\sqrt{2\log(|\mathcal{X}||\mathcal{A}|)} &= \mathbb{E}\sum_{\ell\le L, t<H}\sqrt{\frac{2v\log(|\mathcal{X}||\mathcal{A}|)}{\beta + n_\ell(t, x_t^\ell, a_t^\ell)}} \\
&\overset{(a)}{\le} 2\sqrt{2vH^2|\mathcal{X}|\mathcal{A}|L\log(|\mathcal{X}||\mathcal{A}|)} \\
&= 2H^2\sqrt{2|\mathcal{X}|\mathcal{A}|L\log(|\mathcal{X}||\mathcal{A}|)}
\end{aligned}$$

where the second to last inequality is proved in Lemma 7, provided below. The other term can be bounded as,

$$\begin{aligned}
&\mathbb{E}\left[\beta\left(\|\bar{\theta}\|_\infty + \max_{\ell\le L, t\le H}\|V_{Q_{\ell,t+1}}\|_\infty\right)\sum_{t\le T,\ell\le L}\frac{1}{\beta + n_\ell(t, x_t^\ell, a_t^\ell)}\right] \\
\overset{(b)}{\le}\ &\beta\left(\|\bar{\theta}\|_\infty + \mathbb{E}\left[\max_{\ell\le L, t\le H}\|V_{Q_{\ell,t+1}}\|_\infty\right]\right)H|\mathcal{X}||\mathcal{A}|\log\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right) \\
\overset{(c)}{\le}\ &\beta\left(H + 2H + H^2\sqrt{\log(1 + |\mathcal{X}||\mathcal{A}|HL)}\right)H|\mathcal{X}||\mathcal{A}|\log\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right) \\
\le\ &4\beta H^3|\mathcal{X}|||\mathcal{A}|\sqrt{\log_+(|1 + \mathcal{X}||\mathcal{A}|HL)}\log\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right)
\end{aligned}$$

where the bound on the sum in inequality (b) is from Lemma 7 (proof in the Appendix), and inequality (c) applies Corollary 4.

Lemma 7. *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

$$\mathbb{E}\sum_{\ell=1}^{L} \Delta_\ell \leq 4\beta H^3 |\mathcal{X}|||\mathcal{A}| \sqrt{\log_+(1 + |\mathcal{X}||\mathcal{A}|HL)} \log\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right) + 2H^2\sqrt{2|\mathcal{X}|\mathcal{A}|L\log(|\mathcal{X}||\mathcal{A}|)}.$$

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 $L \geq 16\beta H^2|\mathcal{X}||\mathcal{A}|$. Then

$$\begin{aligned}
\mathbb{E}\sum_{\ell=1}^{L} \Delta_\ell \quad & \leq \quad H^2\sqrt{|\mathcal{X}|\mathcal{A}|L\log_+(1 + |\mathcal{X}||\mathcal{A}|HL)}\left(2\sqrt{2} + 4\beta H\sqrt{|\mathcal{X}||\mathcal{A}|/L}\log\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right)\right) \\
& \leq \quad H^2\sqrt{|\mathcal{X}||\mathcal{A}|L\log_+(1 + |\mathcal{X}||\mathcal{A}|HL)}\left(2\sqrt{2} + \sqrt{\beta}\log_+\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right)\right) \\
& \overset{(*)}{\leq} \quad 4H^2\sqrt{\beta|\mathcal{X}||\mathcal{A}|L\log_+(1 + |\mathcal{X}||\mathcal{A}|HL)}\log_+\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right)
\end{aligned}$$

which is the desired bound. When $L \leq 16H^2|\mathcal{X}||\mathcal{A}|$, we use the naive bound

$$\mathbb{E}\sum_{\ell=1}^{L} \Delta_\ell \leq HL \leq H\sqrt{L}\sqrt{16H^2|\mathcal{X}||\mathcal{A}|} = 4H^2\sqrt{\beta|\mathcal{X}||\mathcal{A}|L},$$

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 working 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 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 randomly generate random "deep-sea" environments according to Example 1 and empirically evaluate performance over many simulations.

Specifically, we apply `learn_grlsvi` (Algorithm 8) with a tabular representation, where 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. We compare the performance of `learn_grlsvi` against two well-studied reinforcement learning algorithms specifically designed to explore efficiently with tabular representations: UCRL2 [7] and PSRL [9]. We similarly modify UCRL2 and PSRL that accelerate learning in the deterministic deep-sea environment[3]. For each of the algorithms our modifications reduce learning times but do not affect rates at which learning times scale with problem size.

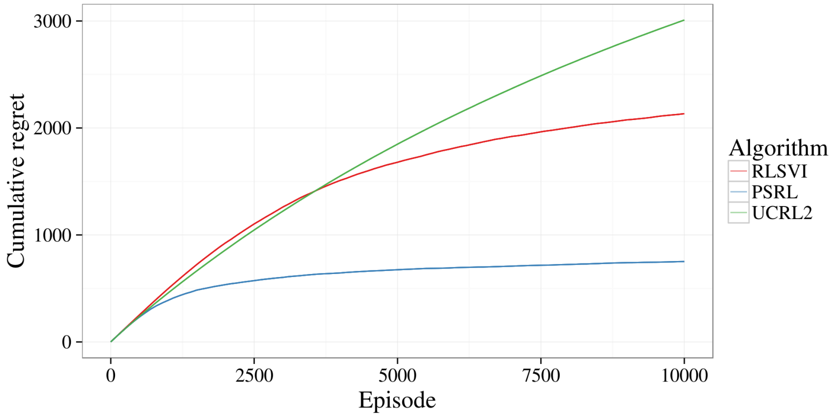


Figure 4: RLSVI is competitive with algorithms designed for tabular exploration ($N = 10$).

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,

[3] 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.

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,

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

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 $\tilde{O}(\sqrt{H^3SAL})$ *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 $\tilde{O}(\sqrt{H^3S^2AL})$ 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 [37], and the associated plot in Figure 5 strengthens the case for that hypothesis.

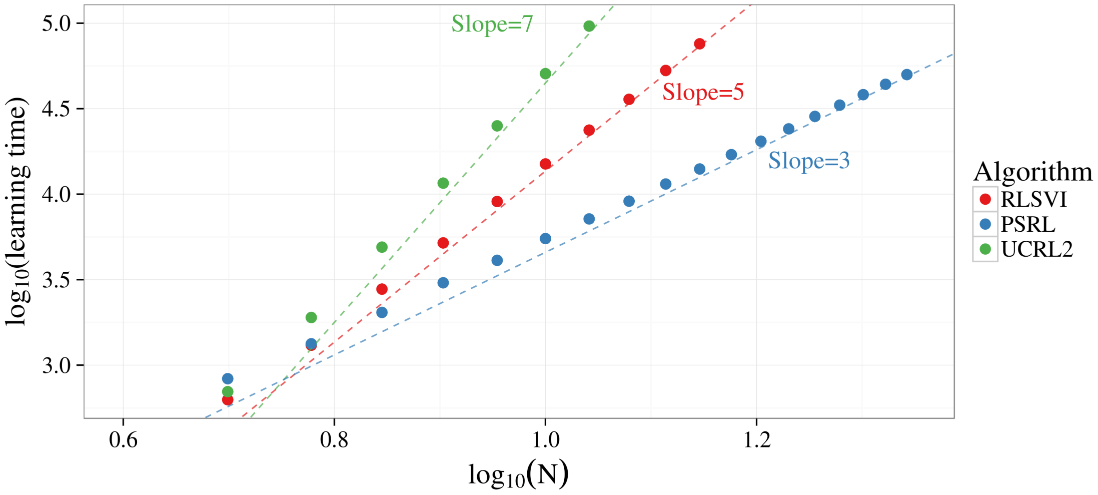


Figure 5: Scaling with tabular learning.

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!

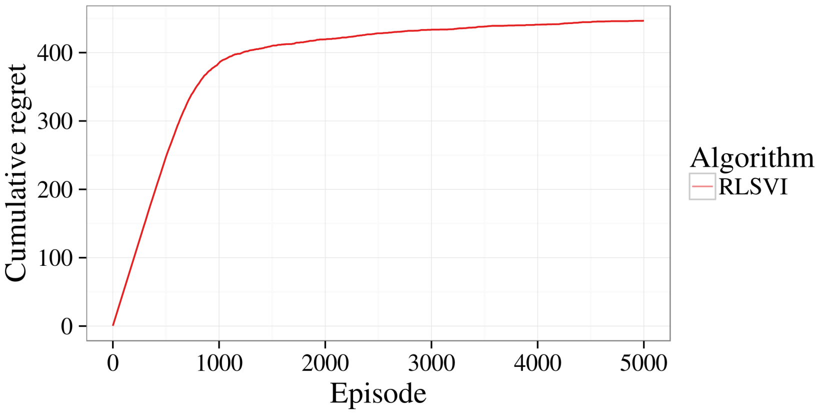
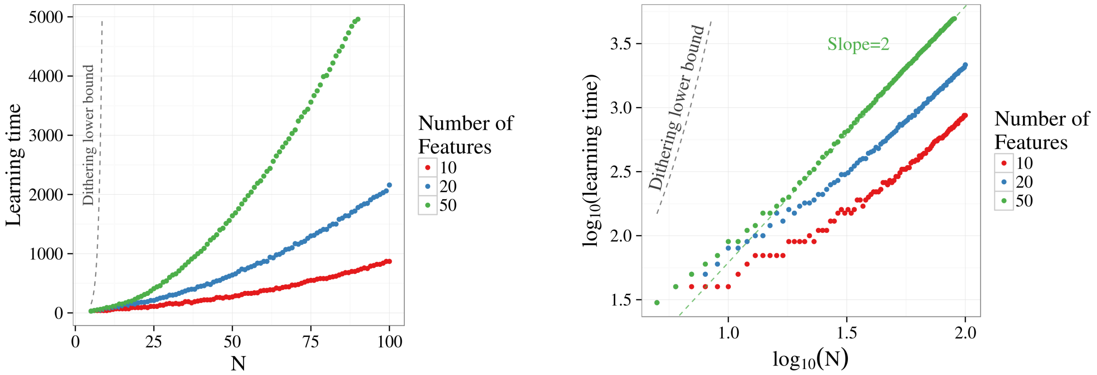


Figure 6: Regret with $N = 50$ and $M = 50$

It is the cases with treasure rather than a bomb 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.

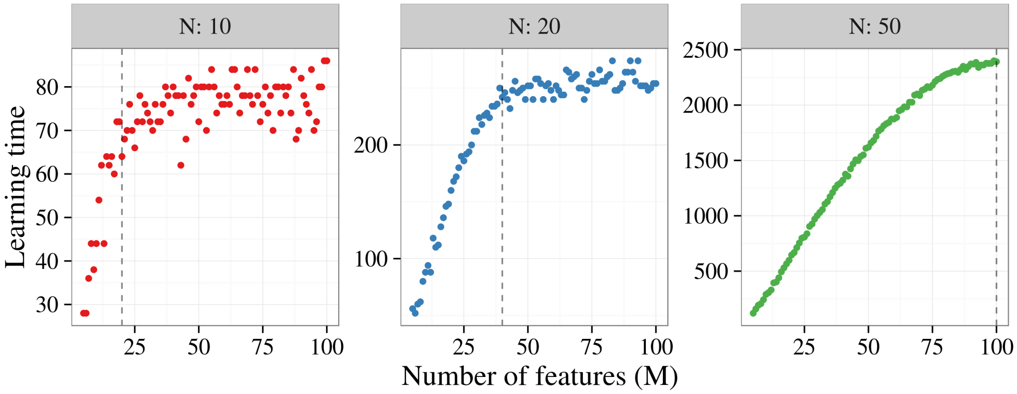


Figure 8: Scaling with number of features.

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 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-mispecification 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.

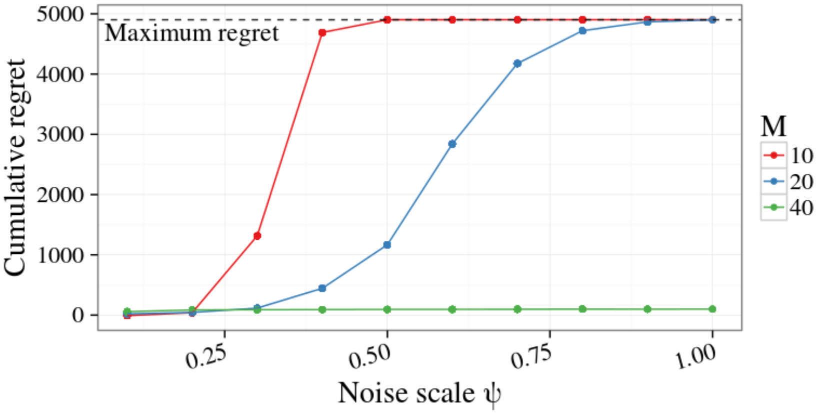


Figure 9: Robustness to misspecification with $N = 20$.

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 consider 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.

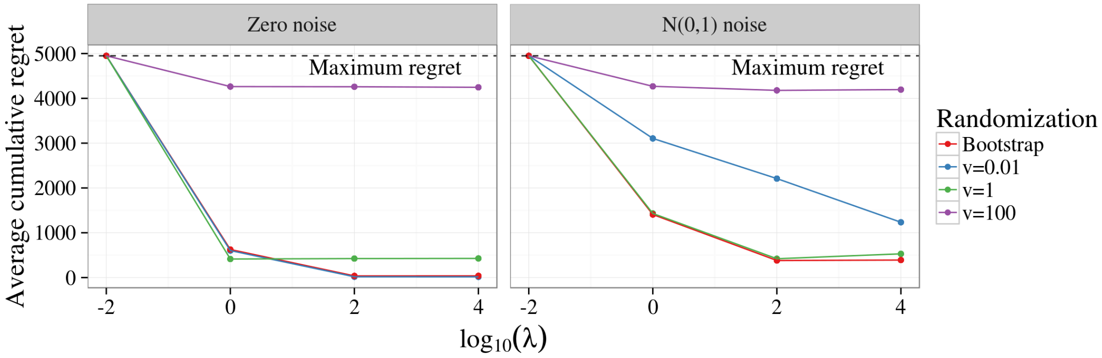


Figure 10: Robustness to prior and noise variance parameters.

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.

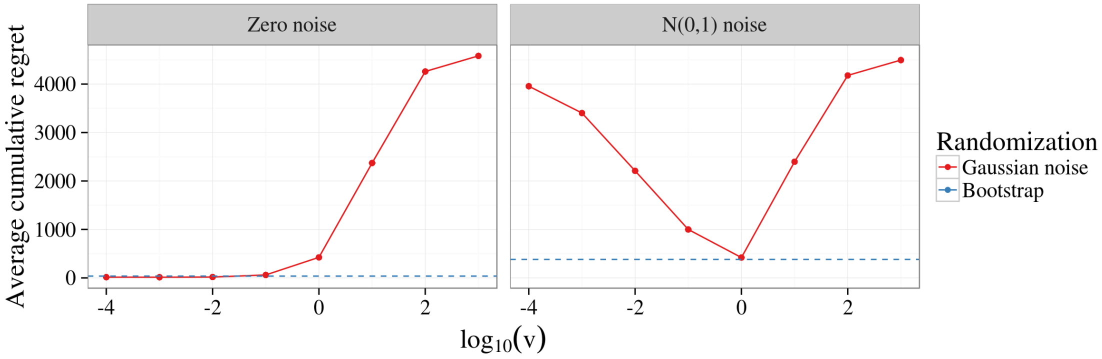


Figure 11: Bootstrap is competitive with the best choice of v across levels of noise.

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.

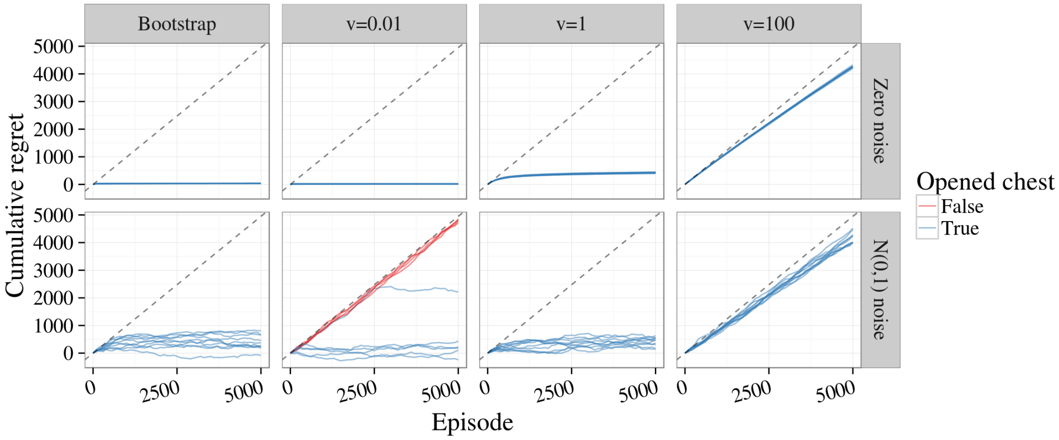
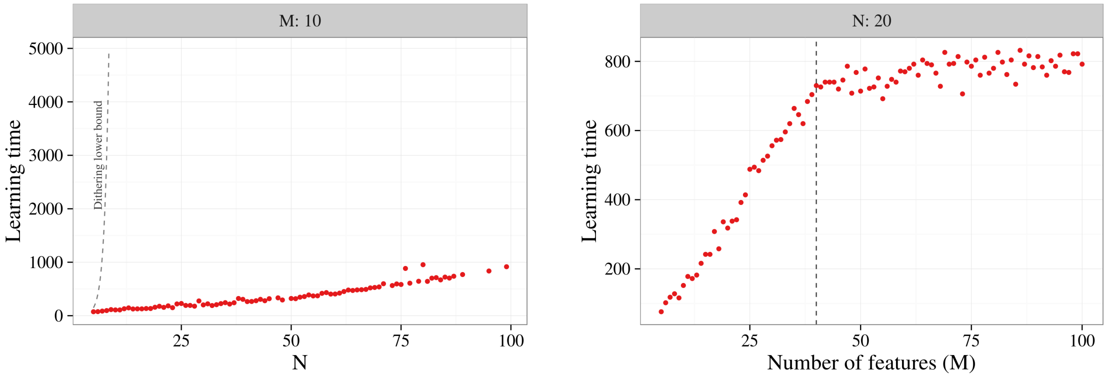


Figure 12: Higher v is more robust to stochastic environments but learns more slowly.

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 with those of `learn_grlsvi`.



(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 in high-dimensional systems

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. We are in the process of expanding upon these experimental work to include an investigation of incremental algorithms applied to high-dimensional systems. Preliminary investigations have been presented for several arcade games, including Tetris [34], Angry Birds [61], and Atari 2600 games [56], as well as for a recommendation system model [34].

8 Closing remarks

Much of the applied reinforcement literature has focussed on simulated systems and eventual performance of policies after learning, often over billions to 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. With this in mind, it may be more appropriate, for example, to evaluate algorithms based on performance 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

This work was generously supported by a research grant from Boeing, a Marketing Research Award from Adobe, and Stanford Graduate Fellowships, courtesy of PACCAR, Burt and Deedee McMurty, and ST Microelectronics. We thank Xiuyuan Lu, Shuhui Ku, and Kuang Xu for pointing out errors and ambiguities in an earlier draft, and more broadly, students who participated in Stanford University's 2017 offering of Reinforcement Learning, for feedback and stimulating discussions on this work.

References

[1] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. Bayesian reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 8(5-6):359–483, 2015.

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

[3] 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.

[4] Peter Auer and Ronald Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In *NIPS*, pages 49–56, 2006.

[5] Alexander L. Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L. Littman. PAC model-free reinforcement learning. In *ICML*, pages 881–888, 2006.

[6] 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.

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

[8] Ronald Ortner and Daniil Ryabko. Online regret bounds for undiscounted continuous reinforcement learning. In *NIPS*, 2012.

[9] Ian Osband, Daniel Russo, and Benjamin Van Roy. (More) efficient reinforcement learning via posterior sampling. In *NIPS*, pages 3003–3011. Curran Associates, Inc., 2013.

[10] Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2818–2826. Curran Associates, Inc., 2015.

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

[12] Alexander L Strehl. *Probably approximately correct (PAC) exploration in reinforcement learning.* ProQuest, 2007.

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

[14] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction.* MIT Press, 2017.

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

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

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

[18] Yitao Liang, Marlos C. Machado, Erik Talvitie, and Michael H. Bowling. State of the art control of Atari games using shallow reinforcement learning. *CoRR*, abs/1512.01563, 2015.

[19] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

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

[21] 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.

[22] Morteza Ibrahimi, Adel Javanmard, and Benjamin Van Roy. Efficient reinforcement learning for high dimensional linear quadratic systems. In *NIPS*, 2012.

[23] Tor Lattimore, Marcus Hutter, and Peter Sunehag. The sample-complexity of general reinforcement learning. In *ICML*, 2013.

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

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

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

[27] 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.

[28] Mohammad Gheshlaghi Azar, Alessandro Lazaric, and Emma Brunskill. Regret bounds for reinforcement learning with policy advice. *CoRR*, abs/1305.1027, 2013.

[29] Lihong Li and Michael Littman. Reducing reinforcement learning to KWIK online regression. *Annals of Mathematics and Artificial Intelligence*, 2010.

[30] Lihong Li, Michael L. Littman, and Thomas J. Walsh. Knows what it knows: a framework for self-aware learning. In *ICML*, pages 568–575, 2008.

[31] Zheng Wen and Benjamin Van Roy. Efficient exploration and value function generalization in deterministic systems. In *NIPS*, pages 3021–3029, 2013.

[32] Jason Pazis and Ronald Parr. PAC optimal exploration in continuous space Markov decision processes. In *AAAI*. Citeseer, 2013.

[33] Richard Dearden, Nir Friedman, and Stuart J. Russell. Bayesian Q-learning. In *AAAI/IAAI*, pages 761–768, 1998.

[34] 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, 2016.

[35] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1471–1479. Curran Associates, Inc., 2016.

[36] 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.

[37] Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning. *arXiv preprint arXiv:1607.00215*, 2016.

[38] W.R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[39] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem.

[40] Shipra Agrawal and Navin Goyal. Further optimal regret bounds for Thompson sampling. *arXiv preprint arXiv:1209.3353*, 2012.

[41] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs.

[42] Dan Russo and Benjamin Van Roy. Eluder dimension and the sample complexity of optimistic exploration. In *NIPS*, pages 2256–2264. Curran Associates, Inc., 2013.

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

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

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

[46] Donald B Rubin et al. The Bayesian bootstrap. *The annals of statistics*, 9(1):130–134, 1981.

[47] 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.

[48] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[49] 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.

[50] Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, pages 417–424, 2001.

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

[52] John N Tsitsiklis, Benjamin Van Roy, et al. An analysis of temporal-difference learning with function approximation. *IEEE transactions on automatic control*, 42(5):674–690, 1997.

[53] Richard S 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.

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

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

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

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

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

[59] Ian Osband and Benjamin Van Roy. Gaussian-dirichlet posterior dominance in sequential learning. *arXiv preprint arXiv:1702.04126*, 2017.

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

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

APPENDIX

A Proofs of technical lemmas

A.1 Proof of Lemma 1

Lemma (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 $x \in \mathcal{X}$,*

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

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

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

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

$$Q^{\pi}_{\mathcal{M},t} = F^{\pi}_{\mathcal{M},t} Q^{\pi}_{\mathcal{M},t+1} \quad t \in \{0, ..., H-1\}.$$

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

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

We have

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

By definition, this means

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

The result follows by iterating this relation. □

A.2 Proof of Lemma 2

Lemma (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 $U_Z : \mathbb{R} \to \mathbb{R}$ 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 $f_z(X_2, ...X_n) \geq_{SO} f_z(Y_2, ...Y_n)$ 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 4

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

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

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,

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

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

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

In addition

$$\begin{aligned}
\mathbb{E}[w_{\max}] = \mathbb{E}\left[\left(\max_{t\leq H,\ell\leq L,a\in\mathcal{A},x\in\mathcal{X}} w_\ell(t,x,a)\right)_+\right] &= \mathbb{E}\left[\left(\max_{t<H,\ell\leq L,a\in\mathcal{A},x\in\mathcal{X}} \sigma_\ell(t,x,a)\frac{w_\ell(t,x,a)}{\sigma_\ell(t,x,a)}\right)_+\right] \\
&\leq \sqrt{\frac{v}{\beta}}\mathbb{E}\left[\left(\max_{t<H,\ell\leq L,a\in\mathcal{A},x\in\mathcal{X}} \frac{w_\ell(t,x,a)}{\sigma_\ell(t,x,a)}\right)_+\right] \\
&\leq \sqrt{2v/\beta\log(1+|\mathcal{X}||\mathcal{A}|HL)}
\end{aligned}$$

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

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

The result then follows by plugging in for β, v, and $\bar{\theta}$. □

A.4 Proof of Lemma 7

Lemma. *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

$$\begin{aligned}
\sum_{t\le H,\ell\le L} \frac{1}{\beta + n_\ell(t, x_t, a_t)} = \sum_{y\in\mathcal{Y}} \sum_{i=0}^{n_L(y)-1} \frac{1}{\beta + i} \quad &\le \quad \sum_{y\in\mathcal{Y}} \int_{\beta-1}^{n_L(y)+\beta-1} \frac{1}{z} dz \\
&= \quad \sum_{y\in\mathcal{Y}} \log\left(\frac{\beta - 1 + n_L(y)}{\beta - 1}\right) \\
&\le \quad \sum_{y\in\mathcal{Y}} \log\left(1 + n_L(y)\right) \\
&\le \quad |\mathcal{Y}| \log\left(\frac{\sum_{y\in\mathcal{Y}}(1 + n_L(y))}{|\mathcal{Y}|}\right) \\
&= \quad |\mathcal{Y}| \log\left(1 + \frac{LH}{|\mathcal{Y}|}\right). \\
&= \quad H|\mathcal{X}||\mathcal{A}| \log\left(1 + \frac{L}{|\mathcal{X}||\mathcal{A}|}\right).
\end{aligned}$$

In addition

$$\begin{aligned}
\sum_{t\le H,\ell\le L} \sqrt{\frac{1}{\beta + n_\ell(t, x_t, a_t)}} = \sum_{y\in\mathcal{Y}} \sum_{i=0}^{n_L(y)-1} \sqrt{\frac{1}{\beta + i}} \quad &\le \quad \sum_{y\in\mathcal{Y}} \int_{z=0}^{n_L(y)} \frac{1}{(\beta - 1 + z)^{1/2}} dz \\
&\le \quad \sum_{y\in\mathcal{Y}} \int_{z=0}^{n_L(y)} \frac{1}{z^{1/2}} dz \\
&= \quad \sum_{y\in\mathcal{Y}} 2\sqrt{n_L(y)} \\
&\le \quad 2\sqrt{|\mathcal{Y}| \sum_{y\in\mathcal{Y}} n_L(y)} \\
&= \quad 2H\sqrt{|\mathcal{X}||\mathcal{A}|L}.
\end{aligned}$$

□