Generalization and Exploration via Randomized Value Functions

Benjamin Van Roy*[1] and Zheng Wen†[1]

[1]Stanford University

Abstract

We consider the problem of reinforcement learning with an orientation toward contexts in which an agent must generalize from past experience and explore to reduce uncertainty. We propose an approach to exploration based on randomized value functions and an algorithm – randomized least-squares value iteration (RLSVI) – that embodies this approach. We explain why versions of least-squares value iteration that use Boltzmann or ϵ-greedy exploration can be highly inefficient and present computational results that demonstrate dramatic efficiency gains enjoyed by RLSVI. Our experiments focus on learning over episodes of a finite-horizon Markov decision process and use a version of RLSVI designed for that task, but we also propose a version of RLSVI that addresses continual learning in an infinite-horizon discounted Markov decision process.

1 Introduction

Reinforcement learning is concerned with how an agent can learn to make effective decisions over time through interacting with an environment. To make efficient use of data and computational resources, the agent must generalize from observations and select actions that strike a balance between exploitation of current knowledge to maximize near-term rewards and exploration to acquire new information that can improve long-term prospects. An important element of reinforcement learning problems, and one that distinguishes them from multi-armed bandit problems, is that the agent may have to learn from delayed consequences; that is, an action can impact the environment over multiple time periods.

An important challenge that remains is to design statistically and computationally efficient reinforcement learning algorithms that simultaneously generalize, explore, and learn from delayed consequences. Prior algorithms can be shown to require learning times exponential in the number of model parameters and/or the planning horizon or to be computationally intractable. In this paper,

*bvr@stanford.edu

†zhengwen@stanford.edu

we aim to make progress on this front. Similarly with the substantial value-based reinforcement learning literature, we focus on an approach that applies generalizes through modeling the state-action value function as a linear combination of pre-selected basis functions. What distinguishes our approach from the pre-existing value function generalization literature is in how the agent explores. In particular, we consider incentivizing exploring through randomly perturbing value functions. As a specific algorithm of this kind, we propose randomized least-squares value iteration (RLSVI). This algorithm fits value functions using randomly perturbed version of least-squares value iteration and selects actions that are greedy with respect to these value functions.

To the best of our knowledge, this paper is the first to propose exploration schemes that randomly perturb value functions for reinforcement learning problems with value function generalization. The use of least-squares value iteration, however, is quite common to the reinforcement learning literature. For example, it is commonly applied in conjunction with Boltzmann or ϵ-greedy exploration. We will explain in this paper why these forms of exploration can lead to highly inefficient learning. We will also present computational results that demonstrate dramatic efficiency gains enjoyed by RLSVI relative to these alternatives.

The remainder of the paper is organized as follows. First, we discuss related literature to put the contributions of this paper in perspective. Then, in Section 3, we briefly review the problem of reinforcement learning over episodes of a finite-horizon MDP. We explain in Section 4 why Boltzmann and ϵ-greedy exploration can be highly inefficient. Next, in Section 5, we motivate and describe RLSVI. Experimental results are presented in Section 6. The version of RLSVI presented in Section 5, which serves as the subject of our experiments, is designed for episodic learning in a finite-horizon MDP. In Section 7, we present a variation of RLSVI that addresses continual learning in an infinite-horizon discounted MDP.

It is worth pointing out that reinforcement learning algorithms are often used to approximate solutions to large-scale dynamic programs. In such contexts, problems are purely computational rather than statistical since the initial problem data provide all required information about the environment. Nevertheless, reinforcement learning algorithms make up popular solution as techniques. As such, our algorithm and results also serve as contributions to approximate dynamic programming.

2 Literature Review

A growing body of work has produced reinforcement learning algorithms with statistical and computational efficiency guarantees [Kearns and Singh, 2002, Brafman and Tennenholtz, 2002, Auer and Ortner, 2006, Strehl et al., 2006, Bartlett and Tewari, 2009, Jaksch et al., 2010]. This literature highlights the point that an effective exploration scheme is critical to the design of any efficient reinforcement learning algorithm. In particular, popular exploration schemes such as ϵ-greedy, Boltzmann, and knowledge gradient can require learning times that grow exponentially in the number of states and/or the planning horizon.

The aforementioned literature focuses on *tabula rasa* learning; that is, algorithms aim to learn

with little or no prior knowledge about transition probabilities and rewards. The sample complexities and computational complexities of such algorithms grow at least linearly with the number of states. Despite the valuable insights that have been generated through their design and analysis, these algorithms are of limited practical import because state spaces in most contexts of practical interest are enormous. There is a need for algorithms that generalize from past experience in order to learn how to make effective decisions in reasonable time.

There has been much work on reinforcement learning algorithms that generalize (see, e.g., [Bertsekas and Tsitsiklis, 1996, Sutton and Barto, 1998, Szepesvári, 2010, Powell and Ryzhov, 2011] and references therein). Most of these algorithms do not come with statistical or computational efficiency guarantees, though there are a few noteworthy exceptions, which we now discuss. A number of results treat policy-based algorithms (see [Kakade, 2003, Azar et al., 2013] and references therein), in which the goal is to select high-performers among a pre-specified collection of policies as learning progresses. Though interesting results have been produced in this line of work, each entails quite restrictive assumptions or does not make strong guarantees. Another body of work focuses on model-based algorithms. An algorithm is proposed in [Kearns and Koller, 1999] that fits a factored model to observed data and makes decisions based on the fitted model. The authors establish a sample complexity bound that is polynomial in the number of model parameters rather than the number of states, but the algorithm is computationally intractable because of the difficulty of solving factored MDPs. A recent paper [Lattimore et al., 2013] proposes a novel algorithm for the case where the true environment is known to belong to a finite or compact class of models, and shows that its sample complexity is polynomial in the cardinality of the model class if the model class is finite, or the ϵ-covering-number if the model class is compact. Though this result is theoretically interesting, for most model classes of interest, the ϵ-covering-number is enormous since it typically grows exponentially in the number of free parameters. Another recent paper [Ortner and Ryabko, 2012] establishes a regret bound for an algorithm that applies to problems with continuous state spaces and Hölder-continuous rewards and transition kernels. Though the results represent an interesting contribution to the literature, a couple features of the regret bound weaken its practical implications. First, regret grows linearly with the Hölder constant of the transition kernel, which for most contexts of practical relevance grows exponentially in the number of state variables. Second, the dependence on time becomes arbitrarily close to linear as the dimension of the state space grows. Reinforcement learning in linear systems with quadratic cost is treated in [Abbasi-Yadkori and Szepesvári, 2011]. The method proposed is shown to realize regret that grows with the square root of time. The result is interesting and the property is desirable, but to the best of our knowledge, expressions derived for regret in the analysis exhibit an exponential dependence on the number of state variables, and further, we are not aware of a computationally efficient way of implementing the proposed method. This work was extended by [Ibrahimi et al., 2012] to address linear systems with sparse structure. Here, there are efficiency guarantees that scale gracefully with the number of state variables, but only under sparsity and other technical assumptions.

The most popular approach to generalization in the applied reinforcement learning literature involves fitting parameterized value functions. Such approaches relate closely to supervised learning

in that they learn functions from state to value, though a difference is that value is influenced by action and observed only through delayed feedback. One advantage over model learning approaches is that, given a fitted value function, decisions can be made without solving a potentially intractable control problem. We see this as a promising direction, though currently there lack both provably efficient algorithms and practically efficient algorithms for reinforcement learning problems with value function generalization. Perhaps one exception is a recent paper [Wen and Van Roy, 2013] that proposes a provably efficient algorithm for reinforcement learning in deterministic systems with value function generalization. However, developing such algorithms for reinforcement learning in stochastic systems remains an open problem. Another relevant paper along these lines [Li and Littman, 2010] studies the efficient reinforcement learning with value function generalization in the KWIK framework (see [Li et al., 2008]), and reduces the efficient reinforcement learning problem to the efficient KWIK online regression problem. However, the authors do not show how to solve the general KWIK online regression problem efficiently, and it is not even clear whether this is possible. Thus, though the result of [Li and Littman, 2010] is interesting, it does not provide an efficient algorithm.

To put RLSVI, our proposed exploration scheme, in perspective, it is also useful to review existing work on exploration schemes that randomly perturb value functions. It is worth pointing out that for multi-armed bandit (MAB) problems, which can be considered very special cases of reinforcement learning problems, which are restricted to have a single state, Thompson sampling constitutes such an exploration scheme [Thompson, 1933, Wyatt, 1997]. Recent work [Russo and Van Roy, 2013] has established efficiency guarantees in this context and may serve as a stepping stone toward the analysis of RLSVI. Another related line of work [Dearden et al., 1998] proposes Q-value sampling, a Bayesian exploration scheme with randomized value functions. The idea of Q-value sampling is to first sample state-action values (Q-values) based on the agent's posterior beliefs, and then select actions greedy with respect to the sampled state-action values. This is equivalent to sampling actions according to the posterior probability that they are optimal. This work does not, however, offer an approach to generalization. Furthermore, as the authors point out, certain assumptions (e.g. Assumption 4) in that paper are likely to be violated in practice.

3 Reinforcement Learning in Episodic MDP

We consider a reinforcement learning (RL) problem in which an agent interacts with a discrete-time finite-horizon Markov decision process (MDP) over a sequence of episodes. The MDP is identified by a sextuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, P, R, \pi)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, H is the horizon length, P encodes transition probabilities, R encodes reward distributions, and π is a distribution over $\mathcal{S}$. In each episode, the initial state x_0 is sampled from π. Then, in period $h = 0, 1, \cdots, H-1$, if the state is x_h and an action a_h is selected then a subsequent state x_{h+1} is sampled from $P(\cdot|x_h, a_h)$ and a reward r_h is sampled from $R(\cdot|x_h, a_h, x_{h+1})$. The episode terminates at the end of period $H-1$, after which a new episode begins.

To represent the history of actions and observations over multiple episodes, we will often index

variables by both episode and period. For example, x_{ih}, a_{ih} and r_{ih} respectively denote the state, action, and reward observed during period h in episode i [1]. The duration over which the agent has operated up to the beginning of period h of episode i is $iH + h$.

A policy $\mu = (\mu_0, \mu_1, \cdots, \mu_{H-1})$ is a sequence of functions, each mapping $\mathcal{S}$ to $\mathcal{A}$. For each policy μ, we define a value function $V_h^\mu(x) = \mathbb{E}\left[\sum_{\tau=h}^{H-1} r_\tau \middle| x_h = x, \mu\right]$, where $x_{\tau+1} \sim P(\cdot|x_\tau, a_\tau)$, $r_\tau \sim R(\cdot|x_\tau, a_\tau, x_{\tau+1})$, $x_h = x$, and $a_\tau = \mu_\tau(x_\tau)$. The optimal value function is defined by $V_h^*(x) = \sup_\mu V_h^\mu(x)$ for $h = 0, 1, \cdots, H-1$. A policy μ^* is said to be optimal if $V^{\mu^*} = V^*$. Throughout this paper, we will restrict attention to MDPs with finite state and action spaces. Such MDPs always admit optimal policies.

It is also useful to define an action-contingent optimal value function:

$$Q_h^*(x,a) = \begin{cases} \mathbb{E}\left[r_h + V_{h+1}^*(x_{h+1}) \middle| x_h = x, a_h = a\right] & \text{if } h < H-1 \\ \mathbb{E}\left[r_h | x_h = x, a_h = a\right] & \text{if } h = H-1 \end{cases}$$

where $x_{h+1} \sim P(\cdot|x_h, a_h)$ and $r_h \sim R(\cdot|x_h, a_h, x_{h+1})$. Then, a policy μ^* is optimal if and only if

$$\mu_h^*(x) \in \arg\max_{\alpha \in \mathcal{A}} Q_h^*(x, \alpha) \quad \forall x, h.$$

A reinforcement learning algorithm generates each action a_{ih} based on observations made up to period h of the episode i, including all states, actions, and rewards observed in previous episodes and earlier in the current episode, as well as the state space $\mathcal{S}$, action space $\mathcal{A}$, horizon H, and possible prior information. In each episode, the algorithm realizes reward $R^{(i)} = \sum_{h=0}^{H-1} r_{ih}$. Note that $\mathbb{E}\left[R^{(i)} \middle| x_{i0}\right] \leq V_0^*(x_{i0})$ for all i. One way to quantify the performance of a reinforcement learning algorithm is in terms of the *expected cumulative regret* over j episodes, defined by

$$\text{Regret}(j) = \sum_{i=0}^{j-1} \mathbb{E}\left[V_0^*(x_{0i}) - R^{(i)}\right]. \tag{1}$$

In this paper, we consider a scenario in which the agent has prior knowledge that for $h = 0, 1, \cdots, H-1$, Q_h^* lies within a linear subspace span $[\Phi_h]$, where $\Phi_h \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$. We henceforth refer to Φ_h as a *generalization matrix* and use $\Phi_h(x,a)$ to denote the row of matrix Φ_h associated with state-action pair (x,a). For $k = 1, 2, \cdots, K$, we denote by ϕ_{hk} the kth column of Φ_h and we will sometimes refer to ϕ_{hk} as a basis function.

We will consider two learning problems that reflect different interpretations of the agent's prior knowledge. In the first case, which we refer to as *coherent learning*, the optimal value function satisfies $Q_h^* \in \text{span}[\Phi_h]$ for $h = 0, \ldots, H-1$. In the other case, which we refer to as *agnostic learning*, the agent may be misinformed and the aforementioned inclusion may not be satisfied; in particular, it could be that $Q_h^* \notin \text{span}[\Phi_h]$ for some choices of h.

[1] If necessary, we use a comma to separate the episode-subscript and the period-subscript.

4 Randomized Actions

Reinforcement learning algorithms we consider in this paper will be based on least-squares value iteration (LSVI). LSVI – as presented below as Algorithm 1[2] – can be applied by the agent at the beginning of each episode to estimate the optimal value function Q^* from data gathered over previous episodes as well as prior knowledge encoded in terms of generalization matrices and a regularization parameter λ. The algorithm iterates backwards over time periods in the planning horizon, in each iteration fitting a value function to the sum of immediate rewards and value estimates of the next period. Each value function is fitted via least-squares: note that vectors θ_{jh} satisfy

$$\theta_{jh} \in \arg\min_{\zeta \in \Re^K} \left(\|A\zeta - b\|^2 + \lambda \|\zeta\|^2 \right).$$

Algorithm 1 Least-Squares Value Iteration

Input: $\Phi_0, \ldots, \Phi_{H-1} \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, $\lambda \in \Re_{++}$, $\{(x_{ih}, a_{ih}, r_{ih}) : i < j, h = 0, \ldots, H-1\}$
Output: $\theta_{j0}, \ldots, \theta_{j,H-1}$

1: $\theta_{jH} \leftarrow 0$, $\Phi_H \leftarrow \mathbf{0}$
2: **for** $h = H-1, \ldots, 1, 0$ **do**
3: Generate regression matrix and vector

$$A \leftarrow \begin{bmatrix} \Phi_h(x_{0h}, a_{0h}) \\ \vdots \\ \Phi_h(x_{j-1,h}, a_{j-1,h}) \end{bmatrix} \qquad b \leftarrow \begin{bmatrix} r_{0,h} + \max_{\alpha \in \mathcal{A}} \left(\Phi_{h+1}\theta_{j,h+1}\right)(x_{0,h+1}, \alpha) \\ \vdots \\ r_{j-1,h} + \max_{\alpha \in \mathcal{A}} \left(\Phi_{h+1}\theta_{j,h+1}\right)(x_{j-1,h+1}, \alpha) \end{bmatrix}$$

4: Estimate value function

$$\theta_{jh} \leftarrow (A^\top A + \lambda I)^{-1} A^\top b$$

5: **end for**

To fully specify a reinforcement learning algorithm, we must describe how the agent selects actions. Let us consider in this section algorithms that, during each episode, select actions that are nearly greedy with respect to value estimates, differing from greedy behavior only due to random perturbations. The form of these random perturbations is governed by an exploration scheme. We consider two popular exploration schemes: Boltzmann exploration and ϵ-greedy exploration (see, e.g., [Powell, 2007]). Reinforcement learning algorithms produced by synthesizing each of these schemes with LSVI are presented as Algorithms 2 and 3. Note that the "temperature" parameters η in Boltzmann exploration and ϵ in ϵ-greedy exploration control the degree to which random perturbations distort greedy actions.

Unfortunately, both reinforcement learning algorithms we have described exhibit worst-case regret that grows exponentially in H and/or $|\mathcal{S}|$. More intelligent exploration schemes are necessary to achieve polynomial regret, even in the case of *tabula rasa* learning, in which each Φ_h is an identity

[2]Notice that in Algorithm 1, when $j = 0$, matrix A and vector b are empty. In this case, we simply set $\theta_{j0} = \theta_{j1} = \cdots = \theta_{j,H-1} = 0$.

Algorithm 2 LSVI with Boltzmann exploration

Input: $\Phi_0, \ldots, \Phi_{H-1} \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, $\lambda \in \Re_{++}$, $\eta \in \Re_+$

for $j = 0, 1, \cdots$ **do**
  Compute $\theta_{j0}, \ldots, \theta_{j,H-1}$ based on Algorithm 1
  Observe x_{j0}
  for $h = 0, 1, \ldots, H-1$ **do**
    Sample $a_{jh} \sim \exp\left[(\Phi_h \theta_{jh})(x_{jh}, a)/\eta\right]$
    Observe r_{jh} and $x_{j,h+1}$
  end for
end for

Algorithm 3 LSVI with ϵ-greedy exploration

Input: $\Phi_0, \ldots, \Phi_{H-1} \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, $\lambda \in \Re_{++}$, $\epsilon \in [0, 1]$

for $j = 0, 1, \ldots$ **do**
  Compute $\theta_{j0}, \ldots, \theta_{j,H-1}$ using Algorithm 1
  Observe x_{j0}
  for $h = 0, 1, \cdots, H-1$ **do**
    Sample $\xi \sim$ Bernoulli(ϵ)
    if $\xi = 1$ **then**
      Sample $a_{jh} \sim$ unif($\mathcal{A}$)
    else
      Sample $a_{jh} \sim$ unif$\left(\arg\max_{\alpha \in \mathcal{A}} (\Phi_t \theta_{jh})(x_{jh}, \alpha)\right)$
    end if
    Observe r_{jh} and $x_{j,h+1}$
  end for
end for

matrix and, therefore, the agent does not generalize across state-action pairs. In the remainder of this section, we provide an example for which LSVI with Boltzmann exploration or ϵ-greedy exploration require exponentially many episodes to learn an optimal policy, even in a coherent learning context with a small number of basis functions.

Example 1 *Consider the MDP illustrated in Figure 1. Each node represents a state, and each arrow corresponds to a possible state transition. The state space is* $\mathcal{S} = \{0, 1, \cdots, N-1\}$ *and the action space is* $\mathcal{A} = \left\{a^{(1)}, a^{(2)}\right\}$. *If the agent takes action* $a^{(1)}$ *at state* $x = 0, 1, \cdots, N-2$ *(the red nodes), the the state transitions to* $y = [x-1]^+$. *On the other hand, if the agent takes action* $a^{(2)}$ *at state* $x = 0, 1, \cdots, N-2$, *the state transitions to* $y = x+1$. *State* $N-1$ *(the green node) is*

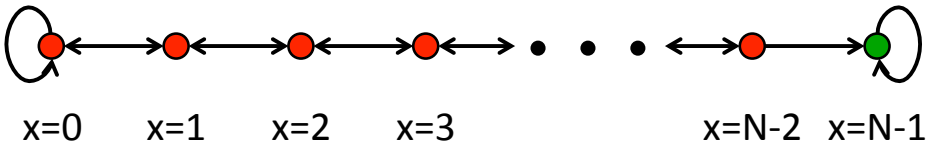

Figure 1: Deterministic system for which Boltzmann/ϵ-greedy exploration is inefficient.

absorbing. We assume a reward of 0 *is realized upon any transition from a red node and a reward of* 1 *is realized upon any transition from the green node. We take the horizon* H *to be equal to the number of states* N*. The initial state in any episode is* 0*.*

Suppose we apply LSVI with either Boltzmann or ϵ*-greedy exploration. Let* i^* *be the first episode during which state* $N-1$ *is visited. It is easy to see that* $\theta_{jh} = 0$ *for all* h *and all* $i < i^*$*. Furthermore, with either exploration scheme, actions are sampled uniformly at random over episodes* $i < i^*$*. Thus, in any episode* $i < i^*$*, the green node will be reached if and only if the algorithm samples* $a^{(2)}$ *consecutively in periods* $t = 0, 1, \cdots, H-2$*. The probability of this event is* $2^{-(H-1)} = 2^{-(N-1)} = 2^{-(|\mathcal{S}|-1)}$*. It follows that* $E[i^*] \geq 2^{|\mathcal{S}|-1}$*. Further, since the optimal expected value over each episode is* 1 *and the realized reward over each episode* $i < i^*$ *is* 0*, it follows that*

$$\begin{aligned}
\text{Regret}(j) &\geq \sum_{i=0}^{j-1} \Pr(i < i^*) \\
&= \sum_{i=0}^{j-1} \left(1 - 2^{-(|\mathcal{S}|-1)}\right)^{i+1} \\
&= \left(2^{|\mathcal{S}|-1} - 1\right)\left(1 - \left[1 - 2^{-(|\mathcal{S}|-1)}\right]^{j}\right). \qquad (2)
\end{aligned}$$

Further, $\liminf_{j\to\infty} \text{Regret}(j) \geq 2^{|\mathcal{S}|-1} - 1$*.*

This example establishes that regret realized by LSVI with Boltzmann or ϵ-greedy exploration can grow exponentially in the number of states. This is far from what one would hope for, which is regret that is independent of the number of states and instead a low-order polynomial in the number of basis functions. Note that the lower bound established for this example applies to both coherent and agnostic learning for any choice of the generalization matrices $\Phi_0, \ldots, \Phi_{H-1}$, any regularization parameter $\lambda > 0$, and any temperature parameter $\eta \geq 0$ or $\epsilon \in [0, 1]$.

5 Randomized Value Functions

RL algorithms of the previous section explore through randomly perturbing greedy actions. We now consider an alternative approach to exploration that involves randomly perturbing value functions rather than actions. As a specific scheme of this kind, we propose randomized least-squares value iteration (RLSVI), which we present as Algorithm 4.[3]

To obtain an RL algorithm, we simply select greedy actions in each episode, as specified in Algorithm 5. Note that RLSVI randomly perturbs value estimates in directions of significant uncertainty to incentivize exploration. This approach relates to Thompson sampling (see, e.g., Thompson [1933], Russo and Van Roy [2013]). We conjecture that Algorithm 5 is statistically efficient in the sense that for any j, Regret(j) is bounded by a low-order polynomial function of K and H. Further, it is easy to see that this RL algorithm is computationally efficient in the sense

[3] Similarly as in Algorithm 1, when $j = 0$, we set $\hat{\theta}_{j0} = \hat{\theta}_{j1} = \cdots = \hat{\theta}_{j,H-1} = 0$.

Algorithm 4 Randomized Least-Squares Value Iteration

Input: $\Phi_0, \ldots, \Phi_{H-1} \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, $\sigma \in \Re_{++}$, $\lambda \in \Re_{++}$, $\{(x_{ih}, a_{ih}, r_{ih}) : i < j, h = 0, \ldots, H-1\}$
Output: $\hat{\theta}_{j0}, \ldots, \hat{\theta}_{j,H-1}$

$\hat{\theta}_{jH} \leftarrow 0$, $\Phi_H \leftarrow \mathbf{0}$
for $h = H-1, \ldots, 1, 0$ **do**
  Generate regression matrix and vector

$$A \leftarrow \begin{bmatrix} \Phi_h(x_{0h}, a_{0h}) \\ \vdots \\ \Phi_h(x_{j-1,h}, a_{j-1,h}) \end{bmatrix} \qquad b \leftarrow \begin{bmatrix} r_{0,h} + \max_{\alpha \in \mathcal{A}} \left(\Phi_{h+1}\hat{\theta}_{j,h+1}\right)(x_{0,h+1}, \alpha) \\ \vdots \\ r_{j-1,h} + \max_{\alpha \in \mathcal{A}} \left(\Phi_{h+1}\hat{\theta}_{j,h+1}\right)(x_{j-1,h+1}, \alpha) \end{bmatrix}$$

  Estimate value function

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

  Sample $\hat{\theta}_{jh} \sim N(\overline{\theta}_{jh}, \Sigma_{jh})$
end for

Algorithm 5 RLSVI with Greedy Action

Input: $\Phi_0, \ldots, \Phi_{H-1} \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, $\sigma \in \Re_{++}$, $\lambda \in \Re_{++}$
for $j = 0, 1, \cdots$ **do**
  Compute $\hat{\theta}_{j0}, \ldots, \hat{\theta}_{j,H-1}$ using Algorithm 4
  Observe x_{j0}
  for $h = 0, \cdots, H-1$ **do**
    Sample $a_{jh} \sim \text{unif}\left(\arg\max_{\alpha \in \mathcal{A}} \left(\Phi_h \hat{\theta}_{jh}\right)(x_{jh}, \alpha)\right)$
    Observe r_{jh} and $x_{j,h+1}$
  end for
end for

that the computational requirements per time period can be bounded by a low-order polynomial function of K, H, and $|\mathcal{A}|$.

6 Experimental Results

In this section, we report results of applying RLSVI to Example 1, with $N = |\mathcal{S}| = H = 50$. We consider both coherent and agnostic learning contexts. Our results demonstrate dramatic efficiency gains relative to LSVI with Boltzmann or ϵ-greedy exploration. The results also illustrate how performance depends on the number of basis functions, algorithm parameters, and, in the agnostic case, the distance between the optimal value function and the span of the basis functions.

To estimate the performance of RLSVI, we average the computation results over $M = 200$ independent simulations, each over $j = 400$ episodes. Based on data produced from these simulations,

we compute for each ith episode an estimate of the episode regret

$$\hat{\Delta}(i) = V_0^*(x_{i0}) - \frac{1}{M}\sum_{m=1}^{M} R^{(mi)},$$

where $R^{(mi)}$ is the reward realized over the ith episode of the mth simulation.

Each of the M independent simulations is of the same MDP. If we also used the same basis functions and algorithm parameters in each case, the accumulation

$$\sum_{i=0}^{j-1}\left(V_0^*(x_{i0}) - \frac{1}{M}\sum_{m=1}^{M} R^{(mi)}\right)$$

of our episode regret estimates would be an unbiased and consistent estimator of Regret(j). However, so that our results do not depend on a specific selection of basis functions, we will randomly sample a separate set of basis functions for each of the M simulations, using a sampling scheme that we will describe later.

As discussed in Section 4, for the MDP under consideration with *any* choice of basis functions and algorithm parameters, LSVI with either Boltzmann exploration or ϵ-greedy exploration, Regret(400) $\geq 400 - 6.82 \times 10^{-13} \approx 400$. This implies that the episode regret is approximately 1 for each of the first 400 episodes. This level of performance offered by the alternative algorithms will serve as baseline as we assess performance of RLSVI.

6.1 Coherent Learning

In the coherent learning case, $Q_h^* \in \text{span}\,[\Phi_h]$ for each period h. Recall that we use ϕ_{hk} to denote the kth basis function, which is the kth column of Φ_h. Each set of $M = 200$ simulations that we carry out to obtain estimates of episode regret is conducted in a manner specified by Algorithm 6

Algorithm 6 Coherent Learning Simulation

Input: $K \in \mathbb{Z}_{++}$, $\sigma \in \Re_{++}$, $\lambda \in \Re_{++}$, $j \in \mathbb{Z}_{++}$, $M \in \mathbb{Z}_{++}$

Output: $\hat{\Delta}(0), \ldots, \hat{\Delta}(j-1)$

```
for m = 1, ..., M do
  for h = 0, ..., H − 1 do
    for k = 1, ..., K do
      switch (k)
      case 1:
        φ_hk ← Q*_h
      case 2:
        φ_hk ← 1
      default:
        Sample φ_hk ~ N(0, I)
      end switch
    end for
  end for
  Simulate RLSVI over j episodes; observe episode rewards R^(m0), ..., R^(m,j−1)
end for
for i = 0, 1, ··· , j − 1 do
  Δ̂(i) = V*_0(x_i0) − (1/M) Σ_{m=1}^{M} R^(mi)
end for
```

We first demonstrate how the experimental results vary with K, the number of basis functions. Here, we let $\lambda = 1$, $\sigma^2 = 10^{-3}$, $j = 400$, $M = 200$, and $K = 2, 5, 10, 20$. Resulting estimates of episode regret are plotted in Figure 2. These results demonstrate that with up to 20 basis functions RLSVI with $\lambda = 1$ and $\sigma^2 = 10^{-3}$ generally takes less than 250 episodes to learn the optimal policy. This represents a dramatic efficiency gain relative to LSVI with Boltzmann or ϵ-greedy exploration, each of which would take more than 5.63×10^{14} episodes. The results also indicate that larger values of K call for longer learning times and larger cumulative regret. This makes sense since larger K implies weaker prior knowledge and therefore a more to learn.

Next, we examine how results vary with the algorithm parameters λ and σ^2. Figure 3 plots results from simulations with $K = 20$, $\lambda = 1$, $M = 200$, $j = 400$, and $\sigma^2 = 10^{-12}, 3.5 \times 10^{-12}, 10^{-9}, 10^{-6}, 10^{-4}, 10^{-2}, 1$. These results demonstrate that for the values of σ^2 considered from 10^{-9} to 10^{-4}, RLSVI generally learns the optimal policy in around 200 episodes; on the other hand, for values of 10^{-12}, 3.5×10^{-12}, 10^{-2} and 1, RLSVI does not learn the optimal policy within 400 episodes. It makes sense that large values of σ^2 and small values of σ^2 lead to a long learning times, since large values of σ^2 induce excessive exploration, while small values of σ^2 lead to insufficient exploration.

Figure 4 plots results from simulations with $K = 20$, $\sigma^2 = 10^{-4}$, $M = 200$, $j = 400$ and $\lambda = 2.5 \times 10^{-8}, 10^{-4}, 10^{-2}, 1, 10^2, 10^4$. These results indicate that for a wide range of settings for λ, RLSVI generally learns the optimal policy in less than 350 episodes. These results also suggest that large values of λ and small values of λ result in a long learning time. This makes sense, since large values of λ lead to over-regularization, which makes basis function weights very small,

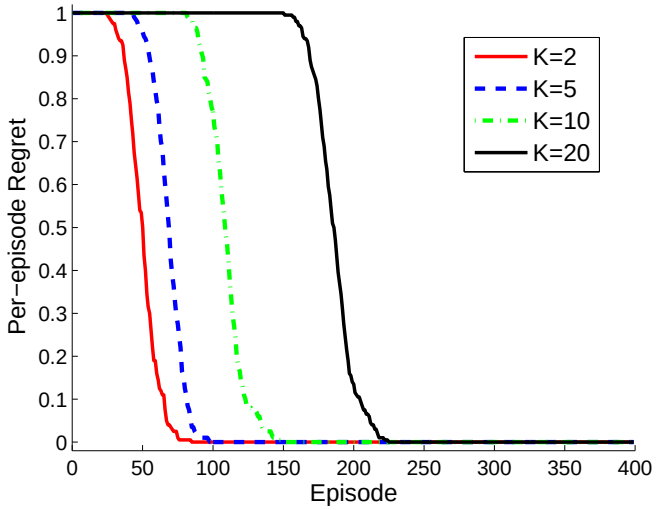

Figure 2: Episode regret estimates for various numbers of basis functions.

restricting exploration. On the other hand, small values of λ can give rise to matrices Σ_{jh} with large eigenvalues, which leads to excessive exploration.

6.2 Agnostic Learning

Our experiments with agnostic learning are conducted in a manner similar to those of the coherent learning case. The procedure, specified as Algorithm 7, differs only in its construction of the basis functions for each time period. Specifically, in the agnostic learning case, we choose $\phi_{h1} = \mathbf{1}$ and $\phi_{hk} = Q_h^* + \rho\psi_{hk}$ for any $k = 2, 3, \cdots, K$, where ψ_{hk} is sampled independently from $N(\mathbf{0}, I)$ and $\rho \geq 0$ is a chosen scalar. Note that if $\rho = 0$, this case reduces to a coherent learning case. As ρ grows, the basis functions are increasingly distorted and the distance between the optimal value function and the span of the basis functions is likely to grow.

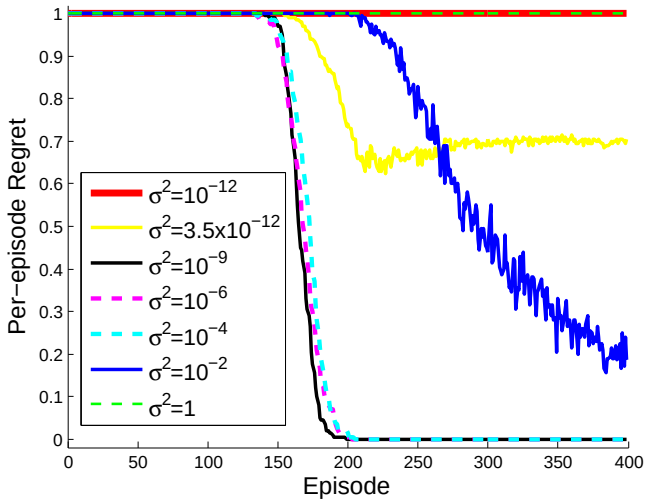

Figure 3: Episode regret estimates for various values of σ^2.

Algorithm 7 Agnostic Learning Simulation

Input: $K \in \mathbb{Z}_{++}$, $\rho \in \Re_+$, $\sigma \in \Re_{++}$, $\lambda \in \Re_{++}$, $j \in \mathbb{Z}_{++}$, $M \in \mathbb{Z}_{++}$

Output: $\hat{\Delta}(0), \ldots, \hat{\Delta}(j-1)$

for $m = 1, \ldots, M$ **do**
  for $h = 0, \ldots, H-1$ **do**
    for $k = 1, \ldots, K$ **do**
      switch (k)
      case 1**:**
        $\phi_{hk} \leftarrow \mathbf{1}$
      default:
        Sample $\psi_{hk} \sim N(\mathbf{0}, I)$
        $\phi_{hk} \leftarrow Q_h^* + \rho\psi_{hk}$
      end switch
    end for
  end for
  Simulate RLSVI over j episodes; observe episode rewards $R^{(m0)}, \ldots, R^{(m,j-1)}$
end for
for $i = 0, 1, \cdots, j-1$ **do**
  $\hat{\Delta}(i) = V_0^*(x_{i0}) - \frac{1}{M}\sum_{m=1}^{M} R^{(mi)}$
end for

Figure 5 plots cumulative regret estimates as a function of ρ, with $K = 11$, $\lambda = 1$, $\sigma^2 = 10^{-3}$, $M = 200$, $j = 400$ and $\rho = 0, 0.01, 0.02, \cdots, 0.1$. Also plotted is an upper bound on cumulative regret, which is also approximately equal to the level of regret realized by LSVI with Boltzmann or ϵ-greedy exploration. These results demonstrate that regret grows with ρ, but that regret realized by RLSVI over the first 400 episodes remains superior to the alternatives for a range of settings.

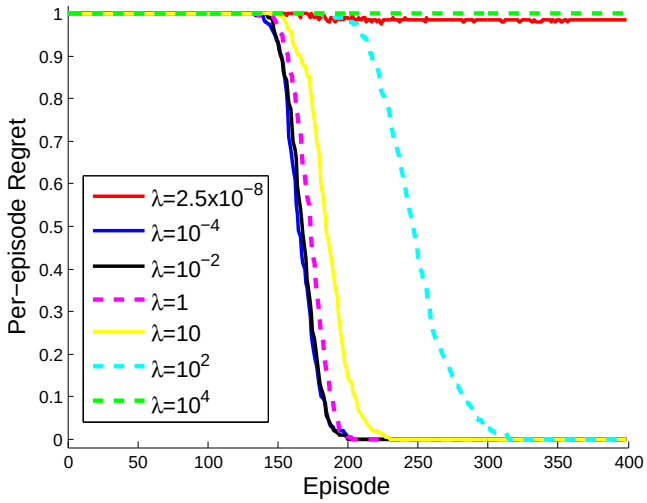

Figure 4: Episode regret estimates for various values of λ.

7 RLSVI in Discounted Infinite-Horizon MDPs

In this section, we propose a version of randomized least-squares value iteration (RLSVI) for reinforcement learning in infinite-horizon discounted MDPs. A discounted MDP is identified by a sextuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \gamma, P, R, \pi)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $\gamma \in (0,1)$ is the discount factor, P encodes transition probabilities, R encodes reward distributions, and π is a distribution over $\mathcal{S}$. In each time $t = 0, 1, \ldots$, if the state is x_t and an action a_t is selected then a subsequent state x_{t+1} is sampled from $P(\cdot|x_t, a_t)$ and a reward r_t is sampled from $R(\cdot|x_t, a_t, x_{t+1})$. The initial state x_0 is sampled from π.

A (stationary) policy μ is a function mapping $\mathcal{S}$ to $\mathcal{A}$. For each policy μ, we define a value function $V^\mu(x) = \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^\tau r_\tau | x_0 = x, \mu\right]$, where $x_{\tau+1} \sim P(\cdot|x_\tau, a_\tau)$, $r_\tau \sim R(\cdot|x_\tau, a_\tau, x_{\tau+1})$, $x_0 = x$, and $a_\tau = \mu(x_\tau)$. The optimal value function is defined by $V^*(x) = \sup_\mu V^\mu(x)$, and a policy μ^* is said to be optimal if $V^{\mu^*} = V^*$. We restrict attention to MDPs with finite state and action spaces. Such MDPs always admit optimal policies. An action-contingent optimal value function is defined by

$$Q^*(x,a) = \mathbb{E}\left[r_t + \gamma V^*(x_{t+1}) | x_t = x, a_t = a\right], \tag{3}$$

where $x_{t+1} \sim P(\cdot|x_t, a_t)$ and $r_t \sim R(\cdot|x_t, a_t, x_{t+1})$. Note that a policy μ^* is optimal if and only if

$$\mu^*(x) \in \arg\max_{\alpha \in \mathcal{A}} Q^*(x, \alpha) \quad \forall x.$$

Similarly with the episodic case, a reinforcement learning algorithm generates each action a_t based on observations made up to time t, including all states, actions, and rewards observed in previous time steps, as well as the state space $\mathcal{S}$, action space $\mathcal{A}$, discount factor γ, and possible

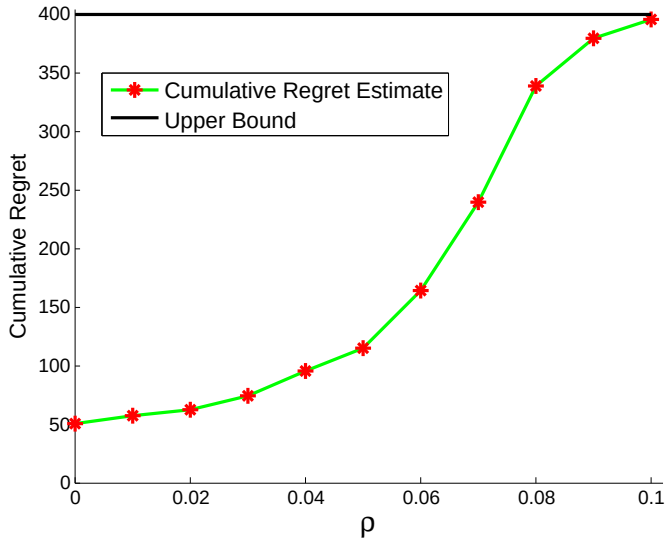

Figure 5: Estimate of cumulative regret over 400 episodes as a function of ρ.

prior information. The realized discounted reward from time t on is $R^{(t)} = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau$. Note that $\mathbb{E}\left[R^{(t)} \middle| x_t\right] \leq V^*(x_t)$ for all t. We will quantify the performance of a reinforcement learning algorithm in terms of the *expected cumulative regret* over the first T time periods, defined by

$$\text{Regret}(T) = \sum_{t=0}^{T-1} \mathbb{E}\left[V^*(x_t) - R^{(t)}\right]. \tag{4}$$

RLSVI for discounted-infinite horizon problems is presented in Algorithm 8. Similarly to Algorithm 4, Algorithm 8 randomly perturbs value estimates in directions of significant uncertainty to incentivize exploration. Note that the random perturbation vectors $w_{t+1} \sim N(\sqrt{1-\gamma^2} w_t, \gamma^2 \Sigma_{t+1})$ are sampled to ensure autocorrelation and that marginal covariance matrices of consecutive perturbations differ only slightly. In each period, a greedy action is selected. Avoiding frequent abrupt changes in the perturbation vector is important as this allows the agent to execute on multi-period plans to learn new information.

References

Yasin Abbasi-Yadkori and Csaba Szepesvári. Regret bounds for the adaptive control of linear quadratic systems. *Journal of Machine Learning Research - Proceedings Track*, 19:1–26, 2011.

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

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

Algorithm 8 Randomized Least-Squares Value Iteration (Discounted MDP)

Input: $\hat{\theta}_t \in \Re^K$, $w_t \in \Re^K$, $\Phi \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, $\sigma \in \Re_{++}$, $\lambda \in \Re_{++}$, $\gamma \in (0,1)$, $\{(x_\tau, a_\tau, r_\tau) : \tau \leq t\}$, x_{t+1}

Output: $\hat{\theta}_{t+1} \in \Re^K$, $w_{t+1} \in \Re^K$

1: Generate regression matrix and vector

$$A \leftarrow \begin{bmatrix} \Phi(x_0, a_0) \\ \vdots \\ \Phi(x_t, a_t) \end{bmatrix} \qquad b \leftarrow \begin{bmatrix} r_0 + \max_{\alpha \in \mathcal{A}} \left(\Phi\hat{\theta}_t\right)(x_1, \alpha) \\ \vdots \\ r_t + \max_{\alpha \in \mathcal{A}} \left(\Phi\hat{\theta}_t\right)(x_{t+1}, \alpha) \end{bmatrix}$$

2: Estimate value function

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

3: Sample $w_{t+1} \sim N(\sqrt{1-\gamma^2} w_t, \gamma^2 \Sigma_{t+1})$

4: Set $\hat{\theta}_{t+1} = \overline{\theta}_{t+1} + w_{t+1}$

Algorithm 9 RLSVI with Greedy Action (Discounted MDP)

Input: $\Phi_0, \ldots, \Phi_{H-1} \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, $\sigma \in \Re_{++}$, $\lambda \in \Re_{++}$, $\gamma \in (0,1)$

1: Set $\hat{\theta}_0 = 0$ and observe x_0

2: **for** $t = 0, 1, \cdots$ **do**

3: Sample $a_t \sim \text{unif}\left(\arg\max_{\alpha \in \mathcal{A}} \left(\Phi\hat{\theta}_t\right)(x_t, \alpha)\right)$

4: Observe r_t and x_{t+1}

5: Compute $\hat{\theta}_{t+1}$ using Algorithm 8

6: **end for**

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.

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

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.

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

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

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

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

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

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

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

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

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.

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

Warren Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality.* Wiley-Interscience, 2007.

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

Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *CoRR*, abs/1301.2609, 2013.

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

Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction.* MIT Press, March 1998.

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

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.

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

Jemery Wyatt. *Exploration and Inference in Learning from Reinforcement*. PhD thesis, University of Edinburgh, 1997.