Context-Dependent Upper-Confidence Bounds for Directed Exploration

Raksha Kumaraswamy[1], **Matthew Schlegel**[1], **Adam White**[1,2], **Martha White**[1]
[1]Department of Computing Science, University of Alberta; [2]DeepMind
{kumarasw, mkschleg}@ualberta.ca, adamwhite@google.com, whitem@ualberta.ca

Abstract

Directed exploration strategies for reinforcement learning are critical for learning an optimal policy in a minimal number of interactions with the environment. Many algorithms use optimism to direct exploration, either through visitation estimates or upper-confidence bounds, as opposed to data-inefficient strategies like ϵ-greedy that use random, undirected exploration. Most data-efficient exploration methods require significant computation, typically relying on a learned model to guide exploration. Least-squares methods have the potential to provide some of the data-efficiency benefits of model-based approaches—because they summarize past interactions—with the computation closer to that of model-free approaches. In this work, we provide a novel, computationally efficient, incremental exploration strategy, leveraging this property of least-squares temporal difference learning (LSTD). We derive upper-confidence bounds on the action-values learned by LSTD, with context-dependent (or state-dependent) noise variance. Such context-dependent noise focuses exploration on a subset of variable states, and allows for reduced exploration in other states. We empirically demonstrate that our algorithm can converge more quickly than other incremental exploration strategies using confidence estimates on action-values.

1 Introduction

Exploration is crucial in reinforcement learning, as the data gathering process significantly impacts the optimality of the learned policies and values. The agent needs to balance the amount of time taking exploratory actions to learn about the world, versus taking actions to maximize cumulative rewards. If the agent explores insufficiently, it could converge to a suboptimal policy; exploring too conservatively, however, results in many suboptimal decisions. The goal of the agent is *data-efficient exploration*: to minimize how many samples are wasted in exploration, particularly exploring parts of the world that are known, while still ensuring convergence to the optimal policy.

To achieve such a goal, directed exploration strategies are key. Undirected strategies, where random actions are taken such as in ϵ-greedy, are a common default. In small domains these methods are guaranteed to find an optimal policy [35], because the agent is guaranteed to visit the entire space—but may take many many steps to do so, as undirected exploration can interfere with improving policies in incremental control. In this paper we explore the idea of constructing confidence intervals around the agent's value estimates. The agent can use these learned confidence intervals to select actions with the highest upper-confidence bound ensuring actions selected are of high value or whose values are highly uncertain. This optimistic approach is promising for directed exploration, but as yet there are few such methods that are model-free, incremental and computationally efficient.

Directed exploration strategies have largely been explored under the framework of "optimism in the face of uncertainty" [13]. These can generally be categorized into count-based approaches and confidence-based approaches. Count-based approaches estimate the "known-ness" of a state,

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

typically by maintaining counts for finite state-spaces [16, 6, 36, 37, 43] and extensions on counting for continuous states [14, 10, 26, 19, 33, 15, 32, 21]. Confidence interval estimates, on the other hand, depend on variance of the target, not just on visitation frequency for states. Confidence-based approaches can be more data-efficient for exploration, because the agent can better direct exploration where the estimates are less accurate. The majority of confidence-based approaches compute confidence intervals on model parameters, both for finite state-spaces [12, 47, 16, 6, 2, 3, 9, 43, 29] and continuous state-spaces [11, 27, 8, 1, 28]. There is early work quantifying uncertainty for value estimates directly for finite state-spaces [22], describing the difficulties with extending the local measures of uncertainty from the bandit literature to RL, since there are long-term dependencies.

These difficulties suggest why using confidence intervals directly on value estimates for exploration in RL has been less explored, until recently. More approaches are now being developed that maintain confidence intervals on the value function for continuous state-spaces, by maintaining a distribution over value functions [8, 31], or by maintaining a randomized set of value functions from which to sample [46, 31, 30, 34, 25]. Though significant steps forward, these approaches have limitations particularly in terms of computational efficiency. Delayed Gaussian Process Q-learning (DGPQ) [8] requires updating two Gaussian processes, which is cubic in the number of basis vectors for the Gaussian process. RLSVI [31] is relatively efficient, maintaining a Gaussian distribution over parameters with Thompson sampling to get randomized values. Their staged approach for finite-horizon problems, however, does not allow for value estimates to be updated online, as the value function is fixed per episode to gather an entire trajectory of data. Moerland et al. [25], on the other hand, sample a new parameter vector from the posterior distribution each time an action is considered, which is expensive. The bootstrapping approaches can be efficient, as they simply have to store several value functions, either for training on a bootstrapped subset of samples—such as in Bootstrapped DQN [30]—or for maintaining a moving bootstrap around the changing parameters themselves, for UCBootstrap [46]. For both of these approaches, however, it is unclear how many value functions would be required, which could be large depending on the problem.

In this paper, we provide an incremental, model-free exploration algorithm with fast converging upper-confidence bounds, called UCLS: Upper-Confidence Least-Squares. We derive the upper-confidence bounds for Least-Squares Temporal Difference learning (LSTD), taking advantage of the fact that LSTD has an efficient summary of past interaction to facilitate computation of confidence intervals. Importantly, these upper-confidence bounds have context-dependent variance, where variance is dependent on state rather than a global estimate, focusing exploration on states with higher-variance. Computing confidence intervals for action-values in RL has remained an open problem, and we provide the first theoretically sound result for obtaining upper-confidence bounds for policy evaluation under function approximation, without making strong assumptions on the noise. We demonstrate in several simulated domains that UCLS outperforms DGPQ, UCBootstrap, and RLSVI. We also empirically show the benefit of using UCLS to a simplified version that uses a global variance estimate, rather than context-dependent variance.

2 Background

We focus on the problem of learning an optimal policy for a Markov decision process, from on-policy interaction. A Markov decision process consists of $(\mathcal{S}, \mathcal{A}, \Pr, r, \gamma)$ where $\mathcal{S}$ is the set of states; $\mathcal{A}$ is the set of actions; $\Pr : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, \infty)$ provides the transition probabilities; $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function; and $\gamma : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the transition-based discount function which enables either continuing or episodic problems to be specified [45]. On each step, the agent selects action A_t in state S_t, and transitions to S_{t+1}, according to $\Pr$, receiving reward $R_{t+1} \stackrel{\text{def}}{=} r(S_t, A_t, S_{t+1})$ and discount $\gamma_{t+1} \stackrel{\text{def}}{=} \gamma(S_t, A_t, S_{t+1})$. For a policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, where $\sum_{a \in \mathcal{A}} \pi(s, a) = 1 \ \forall s \in \mathcal{S}$, the value at a given state s, taking action a, is the expected discounted sum of future rewards, with actions selected according to π into the future,

$$Q^\pi(s, a) = \mathbb{E}\Big[R_{t+1} + \gamma_{t+1} \sum_{a \in \mathcal{A}} \pi(S_{t+1}, a) Q^\pi(S_{t+1}, a) \Big| S_t = s, A_t = a\Big]$$

For problems in which Q^π can be stored in a table, a fixed point for the action-values Q^π exists for a given π. In most domains, Q^π must be approximated by $Q^\pi_{\mathbf{w}}$, parametrized by $\mathbf{w} \in \mathcal{W} \subset \mathbb{R}^d$.

In the case of linear function approximation, state-action features $\mathbf{x}(s_t, a_t)$ are used to approximate action-values $Q^\pi_{\mathbf{w}}(s_t, a_t) = \mathbf{x}(s_t, a_t)^\top \mathbf{w}$. The weights $\mathbf{w}$ can be learned with a stochastic approximation algorithm, called temporal difference (TD) learning [39]. The TD update [39] processes

samples one at a time, $\mathbf{w} = \mathbf{w} + \alpha\delta_t\mathbf{z}_t$, with $\delta_t \stackrel{\text{def}}{=} R_{t+1} + \gamma_{t+1}\mathbf{x}_{t+1}^\top\mathbf{w} - \mathbf{x}_t^\top\mathbf{w}$ for $\mathbf{x}_t \stackrel{\text{def}}{=} \mathbf{x}(S_t, A_t)$. The eligibility trace $\mathbf{z}_t = \mathbf{x}_t + \gamma_{t+1}\lambda\mathbf{z}_{t-1}$ facilitates multi-step updates via an exponentially weighted memory of previous feature activations decayed by $\lambda \in [0, 1]$ and $\mathbf{z}_0 = \mathbf{0}$. Alternatively, we can directly compute the weight vector found by TD using least-squares temporal difference learning (LSTD) [5]. The LSTD solution is more data-efficient, and can avoid the need to tune TD's stepsize parameter $\alpha > 0$. The LSTD update can be efficiently computed incrementally without approximation or storing the data [5, 4], by maintaining a matrix $\mathbf{A}_T$ and vector $\mathbf{b}_T$,

$$\mathbf{A}_T \stackrel{\text{def}}{=} \frac{1}{T}\sum_{t=0}^{T-1}\mathbf{z}_t(\mathbf{x}_t - \gamma_{t+1}\mathbf{x}_{t+1})^\top \qquad \mathbf{b}_T \stackrel{\text{def}}{=} \frac{1}{T}\sum_{t=0}^{T-1}\mathbf{z}_t R_{t+1} \tag{1}$$

The value function approximation at time step T is the weights that satisfy the linear system $\mathbf{A}_T\mathbf{w} = \mathbf{b}_T$. In practice, the inverse of the matrix $\mathbf{A}^{-1}$ is maintained using a Sherman-Morrison update, with a small regularizer η added to the matrix $\mathbf{A}$ to guarantee invertibility [41].

One approach to ensure systematic exploration is to initialize the agent's value estimates optimistically. The action-value function must be initialized to predict the maximum possible return (or greater) from each state and action. For example, for cost-to-goal problems, with -1 per step, the values can be initialized to zero. For continuing problems, with constant discount $\gamma_c < 1$, the values can be initialized to $\text{G}_{\text{max}} = \text{R}_{\text{max}}/(1 - \gamma_c)$, if the maximum reward R_{max} is known. For fixed features that are non-negative and encode locality—such as tile coding or radial basis functions—the weights $\mathbf{w}$ can be simply set to G_{max}, to make $Q_\mathbf{w}$ optimistic.

More generally, however, it can be problematic to use optimistic initialization. Optimistic initialization assumes the beginning of time is special—a period when systematic exploration should be performed after which the agent should more or less exploit its current knowledge. Many problems are non-stationary—or at least benefit from a tracking approach due to aliasing caused by function approximation—and benefit from continual exploration. Further, unlike for fixed features, it is unclear how to set and maintain initial values at G_{max} for learned features, such as with neural networks. Optimistic initialization is also not straightforward for algorithms like LSTD, which completely overwrite the estimate $\mathbf{w}$ on each step with a closed-form solution. In fact, we have found that this issue with LSTD has been obfuscated, because the regularizer η has inadvertently played a role in providing optimism (see Appendix A). Rather, to use optimism in LSTD for control, we need to explicitly compute upper-confidence bounds.

Confidence intervals around action-values, then, provide another mechanism for exploration in reinforcement learning. Consider action selection with explicit confidence intervals around mean estimates $\hat{Q}_\mathbf{w}(S_t, A_t)$, with estimated radius $\hat{U}(S_t, A_t)$. The action selection is greedy w.r.t. to these optimistic values, $\text{argmax}_a\, \hat{Q}_\mathbf{w}(S_t, a) + \hat{U}(S_t, a)$, which provides a high-confidence upper bound on the best possible value for that action. The use of upper-confidence bounds on value estimates for exploration has been well-studied and motivated theoretically in online learning [7]. In reinforcement learning, there have only been a few specialized proofs for particular algorithms using optimistic estimates [8, 31], but the result can be expressed more generally by using the idea of stochastic optimism. We extract the central argument by Osband et al. [31] to provide a general Optimistic Values Theorem in Appendix B. In particular, similar to online learning, we can guarantee that greedy-action selection according to upper-confidence values will converge to the optimal policy, if the confidence interval radius shrinks to zero, if the algorithm to estimate action-values for a policy converges to the corresponding actions and if upper-confidence estimates are stochastically optimal—remain above the optimal action-values in expectation.

Motivated by this result, we pursue principled ways to compute upper-confidence bounds for the general, online reinforcement learning setting. We make a step towards computing such values incrementally, under function approximation, by providing upper-confidence bounds for value estimates made by LSTD, for a fixed policy. We approximate these bounds to create a new algorithm for control—called Upper-Confidence-Least-Squares (UCLS).

3 Estimating Upper-Confidence Bounds for Policy Evaluation using LSTD

Consider the goal of obtaining a confidence interval around value estimates learned incrementally by LSTD for a fixed policy π. The value estimate is $\mathbf{x}^\top\mathbf{w}$ for state-action features $\mathbf{x}$ for the current state and action. We would like to guarantee, with probability $1 - p$ for a small $p > 0$, that the confidence

interval around this estimate contains the value $\mathbf{x}^\top \mathbf{w}^*$ given by the optimal $\mathbf{w}^* \in \mathcal{W}$. To estimate such an interval without parametric assumptions, we use Chebyshev's inequality which—unlike other concentration inequalities like Hoeffding or Bernstein—does not require independent samples.

To use this inequality, we need to determine the variance of the estimate $\mathbf{x}^\top \mathbf{w}$; the variance of the estimate, given $\mathbf{x}$, is due to the variance of the weights. Let $\mathbf{w}^*$ be fixed point solution for the projected Bellman operator for the λ-return—the TD fixed point, for a fixed policy π. To characterize the noise for this optimal estimator, let ν_t be the TD-error for the optimal weights $\mathbf{w}^*$, where

$$r_{t+1} = (\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^\top \mathbf{w}^* + \nu_t \qquad \text{with } \mathbb{E}[\nu_t \mathbf{z}_t] = 0. \tag{2}$$

The expectation is taken across all states weighted by the sampling distribution, typically the stationary distribution $\mathbf{d}_\pi : \mathcal{S} \to [0, \infty)$ or in the off-policy case the stationary distribution of the behaviour policy. We know that $\mathbb{E}[\nu_t \mathbf{z}_t] = 0$, by the definition of the Projected Bellman Error fixed point.

This noise ν_t is incurred from the variability in the reward, the variability in the transition dynamics and potentially the capabilities of the function approximator. A common assumption—when using linear regression for contextual bandits [20] and for reinforcement learning [31]—is that the variance of the target is a constant value σ^2 for all contexts $\mathbf{x}$. Such an assumption, however, is likely to produce larger confidence intervals than necessary. For example, consider a one-state world with two actions, where one action has a high variance reward and the other has a lower variance reward (see Appendix A, Figure 4). A global sample variance will encourage both actions to be taken many times. For data-efficient exploration, however, the agent should take the high-variance action more, and only needs a few samples from the low-variance action.

We derive a confidence interval for LSTD, in Theorem 1. We also derive the confidence interval assuming a global variance in Corollary 1, to provide a comparison. We compare to using this global-variance upper-confidence bound in our experiments, and show that it results in significantly worse performance than using a context-dependent variance. Note that we do not assume $\mathbf{A}_T$ is invertible; if we did, the big-O term in (3) below would disappear. We include this term for preciseness of the result—even though we will not estimate it—because for smaller T, $\mathbf{A}_T$ is unlikely to be invertible. However, we expect this big-O term to get small quickly, and be dominated by the other terms. In our algorithm, therefore, we ignore the big-O term.

Theorem 1. *Let* $\bar{\boldsymbol{\nu}}_T \overset{\text{def}}{=} \frac{1}{T}\sum_{t=0}^{T-1} \mathbf{z}_t \nu_t$ *and* $\mathbf{w}_T = \mathbf{A}_T^+ \mathbf{b}_T$ *where* $\mathbf{A}_T^+$ *is the pseudoinverse of* $\mathbf{A}_T$. *Let* $\boldsymbol{\epsilon}_T^* \overset{\text{def}}{=} (\mathbf{A}_T^+ \mathbf{A}_T - \mathbf{I})\mathbf{w}^*$ *reflect the degree to which* $\mathbf{A}_T$ *is not invertible; it is zero when* $\mathbf{A}_T$ *is invertible. Assume that the following are all finite:* $\mathbb{E}[\mathbf{A}_T^+ \bar{\boldsymbol{\nu}}_T + \boldsymbol{\epsilon}_T^*]$, $\mathbb{V}[\mathbf{A}_T^+ \bar{\boldsymbol{\nu}}_T + \boldsymbol{\epsilon}_T^*]$ *and all state-action features* $\mathbf{x}$. *With probability at least* $1 - p$, *given state-action features* $\mathbf{x}$,

$$\mathbf{x}^\top \mathbf{w}^* \leq \mathbf{x}^\top \mathbf{w}_T + \sqrt{\tfrac{p+1}{p}} \sqrt{\mathbf{x}^\top \mathbb{E}[\mathbf{A}_T^+ \bar{\boldsymbol{\nu}}_T \bar{\boldsymbol{\nu}}_T^\top \mathbf{A}_T^{+\top}] \mathbf{x}} + O\left(\mathbb{E}[(\mathbf{x}^\top \boldsymbol{\epsilon}_T^*)^2]\right) \tag{3}$$

Proof: First we compute the mean and variance for our learned parameters. Because $r_{t+1} = (\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^\top \mathbf{w}^* + \nu_t$,

$$\begin{aligned}
\mathbf{w}_T &= \left(\tfrac{1}{T} \sum_{t=0}^{T-1} \mathbf{z}_t (\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^\top \right)^{-1} \left(\tfrac{1}{T} \sum_{t=0}^{T-1} \mathbf{z}_t r_{t+1} \right) \\
&= \mathbf{A}_T^+ \left(\tfrac{1}{T} \sum_{t=0}^{T-1} \mathbf{z}_t ((\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^\top \mathbf{w}^* + \nu_t) \right) \\
&= \mathbf{A}_T^+ \mathbf{A}_T \mathbf{w}^* + \mathbf{A}_T^+ \left(\tfrac{1}{T} \sum_{t=0}^{T-1} \mathbf{z}_t \nu_t \right) \\
&= \mathbf{w}^* + \mathbf{A}_T^+ \bar{\boldsymbol{\nu}}_T + \boldsymbol{\epsilon}_T^*
\end{aligned}$$

This estimate has a small amount of bias, that vanishes asymptotically. But, for a finite sample,

$$\mathbb{E}\left[\mathbf{A}_T^+ \left(\tfrac{1}{T} \sum_{t=0}^{T-1} \mathbf{z}_t \nu_t \right) \right] \neq \mathbb{E}[\mathbf{A}_T^+] \mathbb{E}\left[\tfrac{1}{T} \sum_{t=0}^{T-1} \mathbf{z}_t \nu_t \right] = \mathbf{0}.$$

Further, because $\mathbf{A}_T$ may not be invertible, there is an additional error $\boldsymbol{\epsilon}_T^*$ term which will vanish with enough samples, i.e., once $\mathbf{A}_T$ can be guaranteed to be invertible.

For covariance, because

$$\begin{aligned}\mathbf{w}_T - \mathbb{E}[\mathbf{w}_T] &= \left(\mathbf{w}^* + \mathbf{A}_T^+\bar{\boldsymbol{\nu}}_T + \boldsymbol{\epsilon}_T^*\right) - \mathbb{E}\left[\mathbf{w}^* + \mathbf{A}_T^+\bar{\boldsymbol{\nu}}_T + \boldsymbol{\epsilon}_T^*)\right] \\ &= \mathbf{A}_T^+\bar{\boldsymbol{\nu}}_T + \boldsymbol{\epsilon}_T^* - \mathbb{E}\left[\mathbf{A}_T^+\bar{\boldsymbol{\nu}}_T + \boldsymbol{\epsilon}_T^*\right]\end{aligned}$$

the covariance of the weights is

$$\mathbb{V}[\mathbf{w}_T] = \mathbb{V}\left[\mathbf{A}_T^+\bar{\boldsymbol{\nu}}_T + \boldsymbol{\epsilon}_T^*\right]$$

The goal for computing variances is to use a concentration inequality. Chebyshev's inequality[1] states that for a random variable X, if the $\mathbb{E}[X]$ and $\mathbb{V}[X]$ are bounded, then for any $\epsilon \geq 0$:

$$\Pr\left(|X - \mathbb{E}[X]| < \epsilon\sqrt{\mathbb{V}[X]}\right) \geq 1 - \frac{1}{\epsilon^2}$$

If we set $\epsilon = \sqrt{1/p}$, then this gives

$$\Pr\left(|X - \mathbb{E}[X]| < \sqrt{\tfrac{1}{p}}\sqrt{\mathbb{V}[X]}\right) \geq 1 - p$$

Now we have characterized the variance of the weights, but what we really want is to characterize the variance of the value estimates. Notice that the variance of the value-estimate, for state-action $\mathbf{x}$ is

$$\begin{aligned}\mathbb{V}[\mathbf{x}^\top\mathbf{w}_T|\mathbf{x}] &= \mathbb{E}[\mathbf{x}^\top\mathbf{w}_T\mathbf{w}_T^\top\mathbf{x}|\mathbf{x}] - \mathbb{E}[\mathbf{x}^\top\mathbf{w}_t|\mathbf{x}]^2 \\ &= \mathbf{x}^\top\left(\mathbb{E}[\mathbf{w}_T\mathbf{w}_T^\top] - \mathbb{E}[\mathbf{w}_T]\mathbb{E}[\mathbf{w}_T]^\top\right)\mathbf{x} \\ &= \mathbf{x}^\top\mathbb{V}[\mathbf{w}_T]\mathbf{x}\end{aligned}$$

Therefore, the variance of the estimate is characterized by the variance of the weights. With high probability,

$$\begin{aligned}\left|\mathbf{x}^\top\mathbf{w}_T - \mathbf{x}^\top\mathbf{w}^*\right| &= \left|\mathbf{x}^\top(\mathbf{w}_T - \mathbb{E}[\mathbf{w}_T]) + \mathbf{x}^\top(\mathbb{E}[\mathbf{w}_T] - \mathbf{w}^*)\right| \\ &\leq \left|\mathbf{x}^\top(\mathbf{w}_T - \mathbb{E}[\mathbf{w}_T])\right| + \left|\mathbf{x}^\top(\mathbb{E}[\mathbf{w}_T] - \mathbf{w}^*)\right| \\ &\leq \frac{1}{\sqrt{p}}\sqrt{\mathbf{x}^\top\mathbb{V}\left[\mathbf{A}_T^+\bar{\boldsymbol{\nu}}_T + \boldsymbol{\epsilon}_T^*\right]\mathbf{x}} + \left|\mathbf{x}^\top\mathbb{E}[\mathbf{A}_T^+\bar{\boldsymbol{\nu}}_T + \boldsymbol{\epsilon}_T^*]\right| \qquad (4) \\ &= \frac{1}{\sqrt{p}}\sqrt{\mathbf{x}^\top\left(\mathbb{E}\left[\mathbf{A}_T^+\bar{\boldsymbol{\nu}}_T\bar{\boldsymbol{\nu}}_T^\top\mathbf{A}_T^{+\top} + \boldsymbol{\Sigma}_T^*\right] - \boldsymbol{\mu}_T^*\boldsymbol{\mu}_T^{*\top}\right)\mathbf{x}} + \sqrt{\mathbf{x}^\top\boldsymbol{\mu}_T^*\boldsymbol{\mu}_T^{*\top}\mathbf{x}} \qquad (5)\end{aligned}$$

where Equation 4 uses Chebyshev's inequality, and the last step is a rewriting of Equation 4 using the definitions $\boldsymbol{\mu}_T^* \stackrel{\text{def}}{=} \mathbb{E}[\mathbf{A}_T^+\bar{\boldsymbol{\nu}}_T + \boldsymbol{\epsilon}_T^*]$ and $\boldsymbol{\Sigma}_T^* \stackrel{\text{def}}{=} \mathbf{A}_T^+\bar{\boldsymbol{\nu}}_T\boldsymbol{\epsilon}_T^{*\top} + \boldsymbol{\epsilon}_T^*(\mathbf{A}_T^+\bar{\boldsymbol{\nu}}_T)^\top + \boldsymbol{\epsilon}_T^*\boldsymbol{\epsilon}_T^{*\top}$.

To simplify (5), we need to determine an upper bound for the general formula $c\sqrt{a^2 - b^2} + b$ where $a \geq b \geq 0$. Because $p < 1$, we know that $c = \sqrt{1/p} \geq 1$. Therefore, the extremal points for b, $b = a$ and $b = 0$, both result in an upper bound of ca. Taking the derivative of the objective, gives a single stationary point in-between $[0, a]$, with $b = \frac{a}{\sqrt{c^2+1}}$. The value at this point evaluates to be $a\sqrt{c^2+1}$. Therefore, this objective is upper-bounded by $a\sqrt{c^2+1}$.

Now for $a^2 = \mathbf{x}^\top\mathbb{E}\left[\mathbf{A}_T^+\bar{\boldsymbol{\nu}}_T\bar{\boldsymbol{\nu}}_T^\top\mathbf{A}_T^{+\top} + \boldsymbol{\Sigma}_T^*\right]\mathbf{x}$, the term involving $\mathbf{x}^\top\mathbb{E}\left[\boldsymbol{\Sigma}_T^*\right]\mathbf{x}$ should quickly disappear, since it is only due to the potential lack of invertibility of $\mathbf{A}_T$. This term is equal to $\mathbb{E}\left[2(\mathbf{x}^\top\mathbf{A}_T^+\bar{\boldsymbol{\nu}}_T)(\mathbf{x}^\top\boldsymbol{\epsilon}_T^*) + (\mathbf{x}^\top\boldsymbol{\epsilon}_T^*)^2\right]$, which results in the additional $O(\mathbb{E}[(\mathbf{x}^\top\boldsymbol{\epsilon}_T^*)^2])$ in the bound.

■

Corollary 1. *Assume that ν_t are i.i.d., with mean zero and bounded variance σ^2. Let $\bar{\mathbf{z}}_T = \frac{1}{T}\sum_{t=0}^{T-1}\mathbf{z}_t$ and assume that the following are finite: $\mathbb{E}[\boldsymbol{\epsilon}_T^*]$, $\mathbb{V}[\boldsymbol{\epsilon}_T^*]$, $\mathbb{E}[\mathbf{A}_T^+\bar{\mathbf{z}}_T\bar{\mathbf{z}}_T^\top\mathbf{A}_T^{+\top}]$ and all state-action features $\mathbf{x}$. With probability at least $1 - p$, given state-action features $\mathbf{x}$,*

$$\mathbf{x}^\top\mathbf{w}^* \leq \mathbf{x}^\top\mathbf{w}_T + \sigma\sqrt{\tfrac{p+1}{p}}\sqrt{\mathbf{x}^\top\mathbb{E}[\mathbf{A}_T^+\bar{\mathbf{z}}_T\bar{\mathbf{z}}_T^\top\mathbf{A}_T^{+\top}]\mathbf{x}} + O\left(\mathbb{E}[(\mathbf{x}^\top\boldsymbol{\epsilon}_T^*)^2]\right) \qquad (6)$$

[1]Bernstein's inequality cannot be used here because we do not have independent samples. Rather, we characterize behaviour of the random variable $\mathbf{w}$, using variance of $\mathbf{w}$, but cannot use bounds that assume $\mathbf{w}$ is the sum of independent random variables. The bound with Chebyshev will be loose, but we can better control the looseness of the bound with the selection of p and the constant in front of the square root.

Proof: The result follows similarly to above, with some simplifications due to global-variance:

$$\mathbb{E}\left[\mathbf{A}_T^+ \bar{\boldsymbol{\nu}}_T\right] = \mathbb{E}\left[\mathbb{E}\left[\mathbf{A}_T^+ \bar{\boldsymbol{\nu}}_T \middle| S_0, ..., S_T\right]\right] = \mathbb{E}\left[\mathbf{A}_T^+ \frac{1}{T}\sum_{t=0}^{T-1} \mathbf{z}_t \mathbb{E}\left[\nu_t \middle| S_0, ..., S_T\right]\right] = \mathbf{0}$$

$$\mathbb{E}[\mathbf{A}_T^+ \bar{\boldsymbol{\nu}}_T \bar{\boldsymbol{\nu}}_T^\top \mathbf{A}_T^{+\top}] = \sigma^2 \mathbb{E}[\mathbf{A}_T^+ \bar{\mathbf{z}}_T \bar{\mathbf{z}}_T^\top \mathbf{A}_T^{+\top}]$$

■

4 UCLS: Estimating upper-confidence bounds for LSTD in control

In this section, we present Upper-Confidence-Least-Squares (UCLS)[2], a control algorithm, which incrementally estimates the upper-confidence bounds provided in Theorem 1, for guiding on-policy exploration. The upper-confidence bounds are sound without requiring i.i.d. assumptions; however, they are derived for a fixed policy. In control, the policy is slowly changing, and so instead we will be slowly tracking this upper bound. The general strategy, like policy iteration, is to slowly estimate both the value estimates and the upper-confidence bounds, under a changing policy that acts greedily with respect to the upper-confidence bounds. Tracking these upper bounds incurs some approximations; we identify and address potential issues here. The complete psuedocode for UCLS is given in the Appendix (Algorithm 2).

First, we are not evaluating one fixed policy; rather, the policy is changing. The estimates $\mathbf{A}_T$ and $\mathbf{b}_T$ will therefore be out-of-date. As is common for LSTD with control, we use an exponential moving average, rather than a sample average, to estimate $\mathbf{A}_T$, $\mathbf{b}_T$ and the upper-confidence bound. The exponential moving average uses $\mathbf{A}_T = (1-\beta)\mathbf{A}_{T-1} + \beta \mathbf{z}_T(\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^\top$, for some $\beta \in [0, 1]$. If $\beta = 1/T$, then this reduces to the standard sample average; otherwise, for a fixed β, such as $\beta = 0.01$, more recent samples have a higher weight in the average. Because an exponential average is unbiased, the result in Theorem 1 would still hold, and in practice the update will be more effective for the control setting.

Second, we cannot obtain samples of the noise $\nu_t = r_{t+1} + \gamma_{t+1}\mathbf{x}_{t+1}^\top \mathbf{w}^* - \mathbf{x}_t^\top \mathbf{w}^*$, which is the TD-error for the optimal value function parameters $\mathbf{w}^*$ (see Equation (2)). Instead, we use δ_t as a proxy. This proxy results in an upper bound that is too conservative—too loose—because δ_t is likely to be larger than ν_t. This is likely to ensure sufficient exploration, but may cause more exploration than is needed. The moving average update

$$\bar{\boldsymbol{\nu}}_t = \bar{\boldsymbol{\nu}}_{t-1} + \beta_t(\delta_t \mathbf{z}_t - \bar{\boldsymbol{\nu}}_{t-1}) \tag{7}$$

should also help mitigate this issue, as older δ_t are likely larger than more recent ones.

Third, the covariance matrix $\mathbf{C}$ estimating $\mathbb{E}[\mathbf{A}_T^{-1}\bar{\boldsymbol{\nu}}_T\bar{\boldsymbol{\nu}}_T^\top \mathbf{A}_T^{-1}]$ could underestimate covariances, depending on a skewed distribution over states and depending on the initialization. This is particularly true in early learning, where the distribution over states is skewed to be higher near the start state; a sample average can result in underestimates in as yet unvisited parts of the space. To see why, let $\mathbf{a} = \mathbf{A}_T^{-1}\bar{\boldsymbol{\nu}}_T$. The covariance estimate $\mathbf{C}_{ij} = \mathbb{E}[\mathbf{a}_i \mathbf{a}_j]$ corresponds to feature i and j. The agent begins in a certain region of the space, and so features that only become active outside of this region will be zero, providing samples $\mathbf{a}_i \mathbf{a}_j = 0$. As a result, the covariance is artificially driven down in unvisited regions of the space, because the covariance accumulates updates of 0. Further, if the initialization to the covariance $\mathbf{C}_{ii}$ is an underestimate, a visited state with high variance will artificially look more optimistic than an unvisited state.

We propose two simple approaches to this issue: updating $\mathbf{C}$ based on locality and adaptively adjusting the initialization to $\mathbf{C}_{ii}$. Each covariance estimate $\mathbf{C}_{ij}$ for features i and j should only be updated if the sampled outer-product is relevant, with the agent in the region where i and j are active. To reflect this locality, each $\mathbf{C}_{ij}$ is updated with the $\mathbf{a}_i \mathbf{a}_j$ only if the eligibility traces is non-zero for i and j. To adaptively update the initialization, the maximum observed $\mathbf{a}_i^2$ is stored, as $c_{\max}$, and the initialization c_0 to each $\mathbf{C}_{ii}$ is retroactively updated using

$$\mathbf{C}_{ii} = \mathbf{C}_{ii} - (1-\beta)^{c_i} c_0 + (1-\beta)^{c_i} c_{\max}$$

[2]We do not characterize the regret of UCLS, and instead similarly to policy iteration, rely on a sound update under a fixed policy to motivate incrementally estimating these values as if the policy is fixed and then acting according to them. The only model-free algorithm that achieves a regret bound is RLSVI, but that bound is restricted to the finite horizon, batch, tabular setting. It would be a substantial breakthrough to provide such a regret bound, and is beyond the scope of this work.

where c_i is the number of times $\mathbf{C}_{ii}$ has been updated. This update is equivalent to having initialized $\mathbf{C}_{ii} = c_{\text{max}}$. We provide a more stable retroactive update to $\mathbf{C}_{ii}$, in the pseudocode in Algorithm 2, that is equivalent to this update.

Fourth, to improve the computational complexity of the algorithm, we propose an alternative, incremental strategy for estimating $\mathbf{w}$, that takes advantage of the fact that we already need to estimate the inverse of $\mathbf{A}$ for the upper bound. In order to do so, we make use of the summarized information in $\mathbf{A}$ to improve the update, but avoid directly computing $\mathbf{A}^{-1}$ as it may be poorly conditioned. Instead, we maintain an approximation $\mathbf{B} \approx \mathbf{A}^{-\top}$ that uses a simple gradient descent update, to minimize $\|\mathbf{A}^\top \mathbf{B}\mathbf{x}_t - \mathbf{x}_t\|_2^2$. If $\mathbf{B}$ is the inverse of $\mathbf{A}^\top$, then this loss is zero; otherwise, minimizing it provides an approximate inverse. This estimate $\mathbf{B}$ is useful for two purposes in the algorithm. First, it is clearly needed to estimate the upper-confidence bound. Second, it also provides a pre-conditioner for the iterative update $\mathbf{w} = \mathbf{w} + \mathbf{G}(\mathbf{b} - \mathbf{A}\mathbf{w})$, for preconditioner $\mathbf{G}$. The optimal preconditioner is in fact the inverse of $\mathbf{A}$, if it exists. We use $\mathbf{G} = \mathbf{B}^\top + \eta\mathbf{I}$ for a small $\eta > 0$ to ensure that the preconditioner is full rank. Developing this stable update for LSTD required significant empirical investigation into alternatives; in addition to providing a more practical UCLS algorithm, we hope it can improve the use of LSTD in other applications.

5 Experiments

We conducted several experiments to investigate the benefits of UCLS' directed exploration against other methods that use confidence intervals for action selection, to evaluate sensitivity of UCLS's performance with respect to its key parameter p, and to contrast the advantage contextual variance estimates offer over global variance estimates in control. Our experiments were intentionally conducted in small—though carefully selected—simulation domains so that we could conduct extensive parameter sweeps, hundreds of runs for averaging, and compare numerous state-of-the-art exploration algorithms (many of which are computationally expensive on larger domains). We believe that such experiments constitute a significant contribution, because effectively using confidence bounds for model free-exploration in RL is still in its infancy—not yet at the large-scale demonstration state–with much work to be done. This point is highlighted nicely below as we demonstrate that several recently proposed exploration methods fail on these simple domains.

5.1 Algorithms

We compare UCLS to DGPQ [8], UCBootstrap [46], our extension of LSPI-Rmax to an incremental setting [19] and RLSVI [31]. In-depth descriptions of each algorithm and implementation details can be found in the Appendix. These algorithms are chosen because they either keep confidence intervals explicitly, as in UCBootstrap, or implicitly as in DGPQ and RLSVI. In addition, we included LSPI-Rmax as a natural alternative approach to using LSTD to maintain optimistic value estimates.

We also include Sarsa with ϵ-greedy, with ϵ optimized over an extensive parameter sweep. Though ϵ-greedy is not a generally practical algorithm, particularly in larger worlds, we include it as a baseline. We do not include Sarsa with optimistic initialization, because even though it has been a common heuristic, it is not a general strategy for exploration. Optimistic initialization can converge to suboptimal solutions if initial optimism fades too quickly [46]. Further, initialization only happens once, at the beginning of learning. If the world changes, then an agent relying on systematic exploration due to its initialization may not react, because it no longer explores. For completeness comparing to previous work using optimistic initialization, we include such results in Appendix G.

5.2 Environments

Sparse Mountain Car is a version of classic mountain car problem Sutton and Barto [40], only differing in the reward structure. The agent only receives a reward of $+1$ at the goal and 0 otherwise, and a discounted, episodic γ of 0.998. The start state is sampled from the range $[-0.6, -0.4]$ with velocity zero. This domain is used to highlight how exploration techniques perform when the reward signal is sparse, and thus initializing the value function to zero is not optimistic.

Puddle World is a continuous state 2-dimensional world with $(x, y) \in [0, 1]^2$ with 2 puddles: (1) $[0.45, 0.4]$ to $[0.45, 0.8]$, and (2) $[0.1, 0.75]$ to $[0.45, 0.75]$ - with radius 0.1 and the goal is the region $(x, y) \in ([0.95, 1.0], [0.95, 1.0])$. The agent receives a reward of $-1 - 400 * d$ on each time step, where d denotes the distance between the agent's position and the center of the puddle, and an undiscounted, episodic γ of 1.0. The agent can select an action to move $0.05 + \zeta$, $\zeta \sim N(\mu = 0, \sigma^2 = 0.01)$.

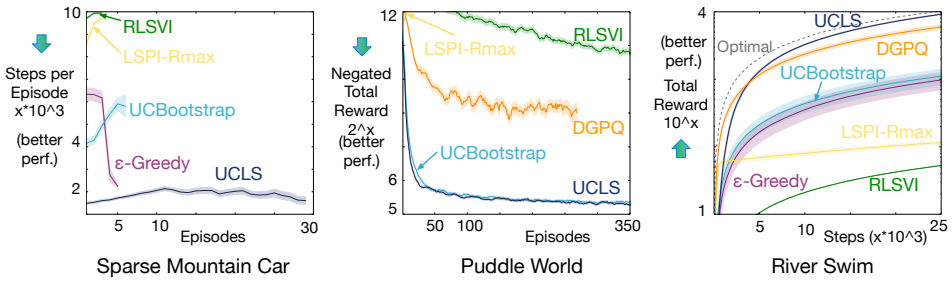

Figure 1: A comparison of speed of learning in Sparse Mountain Car, Puddle World and River Swim. In plots (a) and (b) lower on y-axis are better, whereas in (c) curves higher along y-axis are better. Sparse Mountain Car and Puddle World are episodic problems with a fixed experience budget. Thus the length of the lines in plots (a) and (b) indicate how many episodes each algorithm completed over 50,000 steps, and the height on the y-axis indicates the quality of the learned policy—lower indicates better performance. Note RLSVI did not show significant learning after 50,000 steps. The RLSVI result in Puddle World uses a budget of 1 million.

The agent's initial state is uniformly sampled from $(x, y) \in ([0.1, 0.3], [0.45, 0.65])$. This domain highlights a common difficulty for traditional exploration methods: high magnitude negative rewards, which often cause the agent to erroneously decrease its value estimates too quickly.

River Swim is a standard continuing exploration benchmark [42] inspired by a fish trying to swim upriver, with high reward (+1) upstream which is difficult to reach and, a lower but still positive reward (+0.005), which is easily reachable downstream. We extended this domain to continuous states in $[0, 1]$, with a stochastic displacement of 0.1 when taking an action up or down, with low-probability of success for up. The starting position is sampled uniformly in $[0, 0.1]$, and $\gamma = 0.99$.

5.3 Experimental Setup

We investigate a learning regime where the agents are allowed a fixed budget of interaction steps with the environment, rather than allowing a finite number of episodes of unlimited length. Our primary concern is early learning performance, thus each experiment is restricted to 50,000 steps, with an episode cutoff (in Sparse Mountain Car and Puddle World) at 10,000 steps. In this regime, an agent that spends a significant time exploring the world during the first episode may not be able to complete many episodes, the cutoff makes exploration easier given the strict budget on experience. Whereas, in the more common framework of allowing a fixed number of episodes, an agent can consume many steps during the first few episodes exploring, which is difficult to detect in the final performance results. We average over 100 runs in River Swim and 200 runs for the other domains . For all the algorithms that utilize eligibility traces we set λ to be 0.9. For algorithms which use exponential averaging, β is set to 0.001, and the regularizer η is set to be 0.0001. The parameters for UCLS are fixed. RLSVI's weights are recalculated using all experienced transitions at the beginning of an episode in Puddle World and Sparse Mountain Car, and every 5,000 steps in River Swim. The parameters of competitors, where necessary, are selected as the best from a large parameter sweep.

All the algorithms except DGPQ use the same representation: (1) Sparse Mountain Car - 8 tilings of 8x8, hashed to a memory space of 512, (2) River Swim - 4 tilings of granularity 32, hashed to a memory space of 128, and (3) Puddle World - 5 tilings of granularity 5x5, hashed to a memory space of 128. DGPQ uses its own kernel-based representation with normalized state information.

5.4 Results & Analysis

Our first experiment simply compares UCLS against other control algorithms in all the domains. Figure 1 shows the early learning results across all three domains. In all three domains UCLS achieves the best final performance. In Sparse Mountain Car, UCLS learns faster than the other methods, while in River Swim DGPQ learns faster initially. UCBootstrap and UCLS learn at a similar rate in Puddle World, which is a cost-to-goal domain. UCBootstrap, and bootstrapping approaches generally, can suffer from insufficient optimism, as they rely on sufficiently optimistic or diverse initialization strategies [46, 30]. LSPI-Rmax and RLSVI do not perform well in any of the domains. DGPQ does not perform as well as UCLS in Puddle World, and exhibits high variance compared with the other methods. In Puddle World, UCLS goes on to finish 1200 episodes in the alloted budget of steps,

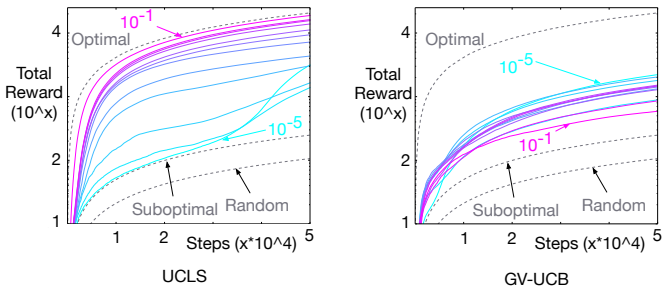

Figure 2: The effect of the confidence parameter p on the policy, in River Swim, using context-dependent variance (UCLS) and global variance (GV-UCB). The values for p are $\{10^{-5}, [1, 2, \ldots, 9] \times 10^{-3}, 10^{-2}, 10^{-1}\}$.

whereas in River Swim both UCLS and DGPQ get close to the optimal policy by the end of the experiment.

The DGPQ algorithm uses the maximum reward (Rmax) to initialize the Gaussian processes. In Sparse Mountain Car this effectively converts the problem back into the traditional -1 per-step formulation. In this traditional variant of Mountain Car UCLS significantly outperforms DGPQ (Appendix G). Sarsa with ϵ-greedy learns well in Puddle world as it is a cost-to-goal problem in which by default Sarsa uses optimistic initialization, and therefore is reported in the Appendix. .

Next we investigated the impact of the confidence level $1 - p$, on the performance of UCLS in River Swim. The confidence interval radius is proportional to $\sqrt{1 + 1/p}$; smaller p should correspond to a higher rate of exploration. In Figure 2, smaller p resulted in a slower convergence rate, but all values eventually reach the optimal policy.

Finally, we investigate the benefit using contextual variance estimates over global variance estimates within UCLS. In Figure 2, we also show the effect of various p values on the performance of the algorithm resulting from Corollary 1, which we call Global Variance-UCB (GV-UCB) (see Appendix E.1 for more details about this algorithm). For this range of p, UCLS still converges to the optimal policy, albeit at different rates. Using a global variance estimates (GV-UCB), on the other hand, results in significant over-estimates of variance, resulting in poor performance.

6 Conclusion and Discussion

This paper develops a sound upper-confidence bound on the value estimates for least-squares temporal difference learning (LSTD), without making i.i.d. assumptions about noise distributions. In particular, we allow for context-dependent noise, where variability could be due to noise in rewards, transition dynamics or even limitations of the function approximator. We then introduce an algorithm, called UCLS, that estimates these upper-confidence bounds incrementally, for policy iteration. We demonstrate empirically that UCLS requires far fewer exploration steps to find high-quality policies compared to several baselines, across domains chosen to highlight different exploration difficulties.

The goal of this paper is to provide an incremental, model-free, data-efficient, directed exploration strategy. The upper-confidence bounds for action-values for fixed policies are one of the few available under function approximation, and so a step towards exploration with optimistic values in the general case. A next step is to theoretically show that using these upper bounds for exploration ensures stochastic optimism, and so converges to optimal policies.

One promising aspect of UCLS is that it uses least-squares to efficiently summarize past experience, but is not tied to a specific state representation. Though we considered a fixed representation for UCLS, it is feasible that an analysis for the non-stationary case could be used as well for the setting where the representation is being adapted over time. If the representation drifts slowly, then UCLS may be able to similarly track the upper-confidence bounds. Recent work has shown that combining deep Q-learning with Least-squares can result in significant performance gains over vanilla DQN[18]. We expect that combining deep networks and UCLS could result in even larger gains, and is a natural direction for future work.

7 Acknowledgements

We would like to thank Bernardo Ávila Pires and Jian Qian for their helpful comments, alongwith Calcul Québec (`www.calculquebec.ca`) and Compute Canada (`www.computecanada.ca`) for the computing resources used in this work.

References

[1] Y. Abbasi-Yadkori and C. Szepesvari. Bayesian Optimal Control of Smoothly Parameterized Systems: The Lazy Posterior Sampling Algorithm. In *Uncertainty in Artificial Intelligence*, 2014.

[2] P. Auer and R. Ortner. Logarithmic Online Regret Bounds for Undiscounted Reinforcement Learning. *Advances in Neural Information Processing Systems*, 2006.

[3] P. L. Bartlett and A. Tewari. REGAL - A Regularization based Algorithm for Reinforcement Learning in Weakly Communicating MDPs. In *Conference on Uncertainty in Artificial Intelligence*, 2009.

[4] J. A. Boyan. Technical update: Least-squares temporal difference learning. *Machine learning*, 49(2-3): 233–246, 2002.

[5] S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine learning*, 22(1-3):33–57, 1996.

[6] R. Brafman and M. Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 2003.

[7] W. Chu, L. Li, L. Reyzin, and R. E. Schapire. Contextual Bandits with Linear Payoff Functions. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[8] R. Grande, T. Walsh, and J. How. Sample Efficient Reinforcement Learning with Gaussian Processes. In *International Conference on Machine Learning*, 2014.

[9] T. Jaksch, R. Ortner, and P. Auer. Near-optimal Regret Bounds for Reinforcement Learning. *The Journal of Machine Learning Research*, 2010.

[10] N. Jong and P. Stone. Model-based exploration in continuous state spaces. *Abstraction, Reformulation, and Approximation*, 2007.

[11] T. Jung and P. Stone. Gaussian processes for sample efficient reinforcement learning with RMAX-like exploration. In *Machine Learning: ECML PKDD*, 2010.

[12] L. P. Kaelbling. *Learning in embedded systems*. MIT press, 1993.

[13] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 1996.

[14] S. Kakade, M. Kearns, and J. Langford. Exploration in metric state spaces. In *International Conference on Machine Learning*, 2003.

[15] K. Kawaguchi. Bounded Optimal Exploration in MDP. In *AAAI Conference on Artificial Intelligence*, 2016.

[16] M. J. Kearns and S. P. Singh. Near-Optimal Reinforcement Learning in Polynomial Time. *Machine Learning*, 2002.

[17] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 2003.

[18] N. Levine, T. Zahavy, D. J. Mankowitz, A. Tamar, and S. Mannor. Shallow updates for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3138–3148, 2017.

[19] L. Li, M. Littman, and C. Mansley. Online exploration in least-squares policy iteration. In *International Conference on Autonomous Agents and Multiagent Systems*, 2009.

[20] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *World Wide Web Conference*, 2010.

[21] J. Martin, S. N. Sasikumar, T. Everitt, and M. Hutter. Count-Based Exploration in Feature Space for Reinforcement Learning. In *International Joint Conference on Artificial IntelligenceI*, 2017.

[22] N. Meuleau and P. Bourgine. Exploration of Multi-State Environments - Local Measures and Back-Propagation of Uncertainty. *Machine Learning*, 1999.

[23] C. D. Meyer, Jr. Generalized inversion of modified matrices. *SIAM Journal on Applied Mathematics*, 24 (3):315–323, 1973.

[24] K. S. Miller. On the inverse of the sum of matrices. *Mathematics magazine*, 54(2):67–72, 1981.

[25] T. M. Moerland, J. Broekens, and C. M. Jonker. Efficient exploration with Double Uncertain Value Networks. In *Advances in Neural Information Processing Systems*, 2017.

[26] A. Nouri and M. L. Littman. Multi-resolution Exploration in Continuous Spaces. In *Advances in Neural Information Processing Systems*, 2009.

[27] R. Ortner and D. Ryabko. Online Regret Bounds for Undiscounted Continuous Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2012.

[28] I. Osband and B. Van Roy. Why is Posterior Sampling Better than Optimism for Reinforcement Learning? In *International Conference on Machine Learning*, 2017.

[29] I. Osband, D. Russo, and B. Van Roy. (More) Efficient Reinforcement Learning via Posterior Sampling. In *Advances in Neural Information Processing Systems*, 2013.

[30] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep Exploration via Bootstrapped DQN. In *Advances in Neural Information Processing Systems*, 2016.

[31] I. Osband, B. Van Roy, and Z. Wen. Generalization and Exploration via Randomized Value Functions. In *International Conference on Machine Learning*, 2016.

[32] G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos. Count-Based Exploration with Neural Density Models. In *International Conference on Machine Learning*, 2017.

[33] J. Pazis and R. Parr. PAC optimal exploration in continuous space Markov decision processes. In *AAAI Conference on Artificial Intelligence*, 2013.

[34] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz. Parameter Space Noise for Exploration. *arXiv.org*, 2017.

[35] S. P. Singh, T. S. Jaakkola, M. L. Littman, and C. Szepesvari. Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms. *Machine Learning*, 2000.

[36] A. Strehl and M. Littman. Exploration via model based interval estimation. In *International Conference on Machine Learning*, 2004.

[37] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman. PAC model-free reinforcement learning. In *International Conference on Machine Learning*, 2006.

[38] R. Sutton, C. Szepesvári, A. Geramifard, and M. Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. In *Conference on Uncertainty in Artificial Intelligence*, 2008.

[39] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 1988.

[40] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[41] C. Szepesvari. *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers, 2010.

[42] I. Szita and A. Lorincz. The many faces of optimism. In *International Conference on Machine Learning*, 2008.

[43] I. Szita and C. Szepesvari. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *International Conference on Machine Learning*, 2010.

[44] H. van Seijen and R. Sutton. A deeper look at planning as learning from replay. In *International Conference on Machine Learning*, 2015.

[45] M. White. Unifying task specification in reinforcement learning. In *International Conference on Machine Learning*, 2017.

[46] M. White and A. White. Interval estimation for reinforcement-learning algorithms in continuous-state domains. In *Advances in Neural Information Processing Systems*, 2010.

[47] M. A. Wiering and J. Schmidhuber. Efficient Model-Based Exploration. In *Simulation of Adaptive Behavior From Animals to Animats*, 1998.

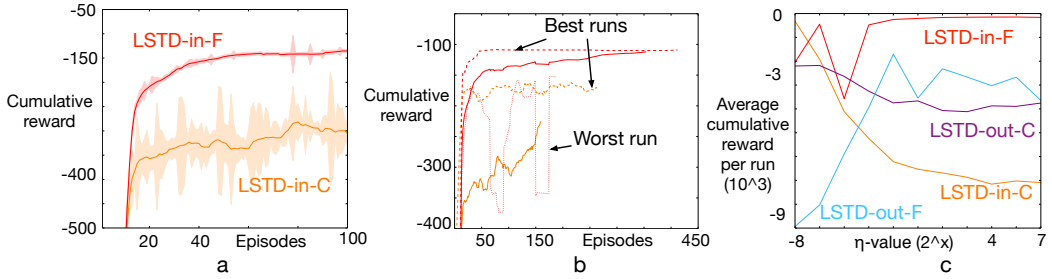

Figure 3: Learning performance in Mountain Car for LSTD-in and LSTD-out with η kept constant through learning (-C) and η fading with time (-F). (a) Early learning curves for LSTD-in. This plot does not include LSTD-out as it performed too poorly to be visible. (b) Learning curves for LSTD-in with best and worst runs. LSTD-in-C's worst run performed too poorly to be visible. (c) Parameter sensitivity for both variants LSTD-in and LSTD-out to η/η_r.

A Issues with LSTD for control

LSTD is a more data-efficient algorithm than its incremental counterpart TD, and typically performs quite well in policy evaluation. This is primarily due to TD only using each sample once for a stochastic update with a tuned stepsize parameter. In the case of control, LSTD performs surprisingly well without ϵ-greedy exploration and lack of an optimism strategy. We highlight here the inadvertent use of the regularization parameter as a form of optimism for LSTD in control, and empirically show when this strategy fails leading us to UCLS as a sound approach in using LSTD in control.

In practice, the inverted matrix $\mathbf{A}^{-1}$ is often directly maintained using a Sherman-Morrison update, with a small regularizer η added to the matrix $\mathbf{A}$ to guarantee invertibility [41].

There are two objectives that can be solved when dealing with an ill-conditioned system $\mathbf{A}\mathbf{w} = \mathbf{b}$. The most common is to use Tikohonov regularization solving, referred to here as *LSTD-out*.

$$\min_{\mathbf{w}} \|\mathbf{A}\mathbf{w} - \mathbf{b}\|_2^2 + \eta_r \|\mathbf{w}\|_2^2$$

Another approach is to solve the system

$$\min_{\mathbf{w}} \|(\mathbf{A} + \eta\mathbf{I})\mathbf{w} - \mathbf{b}\|_2^2$$

The second approach is implicitly what is solved when a Sherman-Morrison update is used for $\mathbf{A}^{-1}$, with a small regularizer η added to the matrix $\mathbf{A}$ to guarantee invertibility. This approach is referred to here as *LSTD-in*. When $\eta = 0$, both approaches are solving $\|\mathbf{A}\mathbf{w} - \mathbf{b}\|_2^2$, which may have infinitely many solutions if $\mathbf{A}$ is not full rank. While the Tikohonov regularization strategy is more common, the second approach is useful for enabling use of the incremental Sherman-Morrison update to facilitate maintaining $\mathbf{A}^{-1}$ directly.

Another choice in regularizing the ill-conditioned system is in how η decays over time. A small fixed η can be used as a constant regularizer, even as the number of samples increases, because the true $\mathbf{A}$ may be ill-conditioned. However, more regularization could also be used at the beginning and then decayed over time. The incremental Sherman-Morrison update implicitly decays η proportionally to $\frac{1}{t}$.

We conducted an empirical study using LSTD without an ϵ-greedy exploration strategy in two domains: Mountain Car and a new One-State world. One-State world—depicted in Figure 4—simulates a typical setting where sufficient exploration is needed: one outcome with low variance and lower expected value and one outcome with high variance and higher expected value. For an algorithm that does not explore sufficiently, it is likely to settle on the suboptimal action, but more immediately rewarding low-variance outcome. This world simulates a larger continuous navigation task from 46. We include results for both systems described above and consider a fading version (shown by *-F*) or a constant regularization parameter (shown by *-C*).

Figure 3 shows results for the four different LSTD strategies in Mountain Car. The Tikohonov regularization, with η_r, is unable to learn an optimal policy in this domain, whereas with either constant or fading η, the agent can learn an optimal policy. This is surprising, considering we use

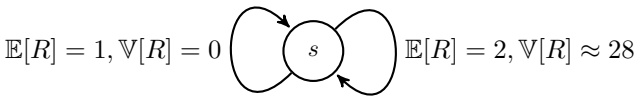

Figure 4: One-state world, where the optimal action (right) has high-variance; the reward here is uniformly sampled from within the set $\{-5, -2, 2, 5, 10\}$. LSTD, with $\epsilon = 0$ and η large, fails in this world, unlike the cost-to-goal problems.

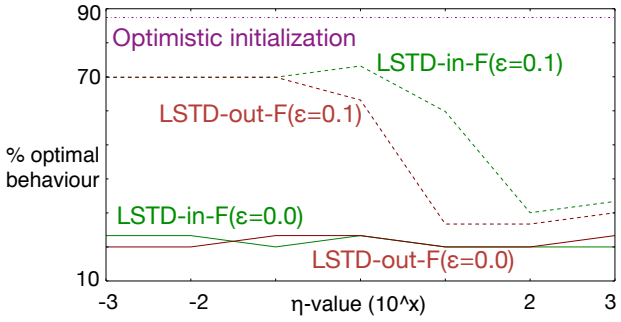

Figure 5: η-sensitivity in 1-State world with various LSTD updates. Sarsa with optimistic initialization $\alpha = 0.001$ is used as a baseline. The y-axis represents percentage optimal behaviour, where optimal behaviour is choosing to go right, in 20k steps (averaged over 30 runs). Sarsa with optimistic initialization is highly sensitive to the step-size chosen. With other stepsizes (not shown in figure), it reduces its values too quickly, and fails a significant percentage of the time. The best stepsize is chosen here to show near-optimal performance is possible in the domain.

neither randomized exploration nor optimistic initialization. The parameter sensitivity curve, shown in plot c, indicates η and η_r needs to be sufficiently large as time passes in order to find an optimal policy.

Next, we show that neither regularization strategy with fading η is effective in the One-State world. The optimal strategy is to take the Right action, to get an expected reward of 2 under a higher variance for obtaining rewards. All of the LSTD variants fail for this domain, because η no longer plays a role in encouraging exploration. To verify that a directed exploration strategy helps, we experiment with ϵ-greedy exploration, with $\epsilon = 0.1$, decayed by a factor of 0.2 every 100 steps (shown in Figure 5). With ϵ-greedy, and small values of η_r and η, the policy converges to the optimal action, whereas it fails to with higher values of η_r and η.

These results suggest that η's role in exploration has obscured our understanding of how to use LSTD for control. LSTD, with sufficient optimism does seem to reach optimal solutions, and unlike Sutton et al. [38], we did not find any issues with forgetting. This further explains why there have been previous results with small ϵ for LSTD in cost-to-goal problems, that nonetheless still obtained the optimal policy [44]. Therefore, in developing UCLS, we more explicitly add optimism to LSTD, and ensure η is strictly used as a regularization parameter (to ensure well-conditioned updates).

B Optimistic Values Theorem

The use of upper confidence bounds on value estimates for exploration has been well-studied and motivated theoretically in online learning [7]. For reinforcement learning, though, there are only specialized proofs for particular algorithms using optimistic estimates [8, 31]. To better motivate and appreciate the use of upper confidence bounds for reinforcement learning, we extract the key argument from Osband et al. [31], which uses the idea of stochastic optimism.

Under function approximation, it may not be possible to obtain the optimal policy exactly. Instead, our criterion is to obtain the optimal policy according to the following formulation, assuming greedy-action selection from action-values. Let $Q^* : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ be the action-values for the optimal policy,

under the chosen density $d : \mathcal{S} \times \mathcal{A} \to [0, \infty)$ over states and actions

$$Q^* = \operatorname*{argmax}_{Q \in \mathcal{Q}} \int_{\mathcal{S} \times \mathcal{A}} d(s, a) Q(s, a) ds da \tag{8}$$

This optimization does not preclude d being related to the trajectory of optimal policy, but generically allows specification of any density, such as one putting all weight on a set of start states or such as one that is uniform across states and actions to ensure optimality from any point in the space. The optimal policy in this setting is the policy that corresponds to acting greedily w.r.t. Q^*; depending on the function space $\mathcal{Q}$, this may only be an approximately optimal policy. The design of the agent is directed towards this goal, though we do not explicitly optimize this objective.

Let $\tilde{Q}_t = \hat{Q}_t + \hat{U}_t$ be the estimated action-values plus the confidence interval radius $\hat{U}_t$ on time step t, to get the estimated upper confidence bound which the agent uses to select actions. Let π_t be the policy induced by greedy action selection on $\tilde{Q}_t$.

Assumption 1 (Stochastic Optimism)**.** *At some point $T > 0$, the action-values at every step $t \geq T$ are* stochastically optimistic*: $\mathbb{E}[\tilde{Q}_t(S, A)] \geq \mathbb{E}[Q^*(S, A)]$, with expectation according to a specified density $d : \mathcal{S} \times \mathcal{A} \to [0, \infty)$.*

Assumption 2 (Shrinking Confidence Interval Radius)**.** *The confidence interval radius $\hat{U}_t$ goes to zero: $\mathbb{E}[\hat{U}_t(S, A)] \leq f(t)$ for some non-negative function f with $f(t) \to 0$.*

Assumption 3 (Convergent Action Values)**.** *The estimated action-values $\hat{Q}_t$ approach the true action-values for policy π_t: $\left|\mathbb{E}[\hat{Q}_t(S, A) - Q^{\pi_t}(S, A)]\right| \leq g(t)$ for some non-negative function g with $g(t) \to 0$.*

These assumptions are heavily dependent on the distribution utilized to evaluate the expectation. If the expectations are w.r.t. the stationary distribution induced by the optimal policy (d^*), it is easy to see that they could be satisfied - as the density is non-zero only for the optimal state-action pairs. In contrast, if the density is a uniform density over the space, then these assumptions may not be satisfied.

Given the three key assumptions, the theorem below is straightforward to prove. However, these three conditions are fundamental, and do not imply each other. Therefore, this result highlights what would need to be shown, to obtain the Optimistic Values Theorem. For example, Assumption 1 and 2 do not imply Assumption 3, because the confidence interval radius could decrease to zero, and $\hat{Q}_t$ still be stochastically optimistic and an over-estimate of values that correspond to a suboptimal policy. Assumption 1 and 3 do not imply Assumption 2, because $\hat{Q}_t$ could converge to the policy corresponding to acting greedily w.r.t. $\tilde{Q}_t$, but $\hat{U}_t$ may never fade away. Then, $\tilde{Q}_t$ could still be stochastically optimistic, but the policy π_t could be suboptimal because it is acting greedily according to inaccurate, inflated estimates of value $\tilde{Q}_t$.

Theorem 2 (Optimistic Values Theorem)**.** *Under Assumptions 1, 2 and 3,*

$$\begin{aligned}
\mathbb{E}[Q^*(S, A)] - \mathbb{E}[Q^{\pi_t}(S, A)] &\leq f(t) + g(t) \\
\textit{Regret}(T) &\stackrel{\text{def}}{=} \sum_{t=1}^{T} \mathbb{E}[Q^*(S, A)] - \mathbb{E}[Q^{\pi_t}(S, A)] \\
&\leq \sum_{t=1}^{T} f(t) + g(t)
\end{aligned}$$

Proof: Consider the regret across states and actions

$$\begin{aligned}
\mathbb{E}[Q^*(S, A) - Q^{\pi_t}(S, A)] &= \mathbb{E}[Q^*(S, A) - \tilde{Q}_t(S, A)] + \mathbb{E}[\tilde{Q}_t(S, A) - Q^{\pi_t}(S, A)] \\
&\leq \mathbb{E}[\tilde{Q}_t(S, A) - Q^{\pi_t}(S, A)]
\end{aligned}$$

because $\mathbb{E}[Q^*(S, A) - \tilde{Q}_t(S, A)] \leq 0$ by Assumption 1. By Assumptions 2 and 3,

$$\begin{aligned}
\mathbb{E}[\tilde{Q}_t(S, A) - Q^{\pi_t}(S, A)] &= \mathbb{E}[\hat{Q}_t(S, A) - Q^{\pi_t}(S, A)] + \mathbb{E}[\hat{U}_t(S, A)] \\
&\leq g(t) + f(t)
\end{aligned}$$

completing the proof. ■

This result is intentionally abstract, where the three assumptions could be satisfied in a variety of ways. These assumptions have been verified for one algorithm, called RLSVI, under a tabular setting using a finite-horizon specification [31], which simplifies ensuring stochastic optimism (Assumption 1). We hypothesize that the last two assumptions could be addressed with a two-timescale analysis, with confidence interval radius $\hat{U}_t$ updating more slowly than $\hat{Q}_t$. This would reflect an iterative approach, where the optimistic values are essentially held fixed—such as is done in Delayed Q-learning [8]—and Q^{π_t} estimated, before then adjusting the optimistic values. The updates to $\hat{Q}_t$, then, would be updated on a faster timescale, converging to Q^{π_t}, and the upper confidence radius $\hat{U}_t$ updating on a slower timescale.

Algorithm 1 GetOptimisticAction($\mathbf{x}_{s,\cdot}$)

$u_a \leftarrow \sqrt{\left(1+\frac{1}{p}\right)\mathbf{x}_{s,a}^\top \mathbf{C}\mathbf{x}_{s,a}} \quad \forall a \in \mathcal{A}$
$a = \text{argmax}_{a\in\mathcal{A}}\, \mathbf{x}_{s,a}^\top \mathbf{w} + u_a$
return a

Algorithm 2 UCLS(λ)

$\mathbf{A} \leftarrow \mathbf{0}$, $\mathbf{b} \leftarrow \mathbf{0}$, $\mathbf{z} \leftarrow \mathbf{0}$, $\mathbf{w} \leftarrow \mathbf{0}$
$\mathbf{B} \leftarrow \mathbf{I}$, $\mathbf{C} \leftarrow \mathbf{I}$, $\bar{\boldsymbol{\nu}} \leftarrow \mathbf{0}$, $\mathbf{c} \leftarrow \mathbf{1}$
$p = 0.1$, $\eta = 10^{-4}$, $\beta = 0.001$, $c_{\max} = 1.0$
$\mathbf{x}_{s,\cdot} \leftarrow$ initial state-action features, for any action
$a \leftarrow$ GetOptimisticAction($\mathbf{x}_{s,\cdot}$)
repeat
  Take action a and observe $\mathbf{x}_{s',\cdot}$ and r, and γ
  $a' \leftarrow$ GetOptimisticAction($\mathbf{x}_{s',\cdot}$)
  $\delta \leftarrow r + (\gamma \mathbf{x}_{s',a'} - \mathbf{x}_{s,a})^\top \mathbf{w}$
  $\mathbf{z} \leftarrow \gamma\lambda \mathbf{z} + \mathbf{x}_{s,a}$
  $\mathbf{b} \leftarrow (1-\beta)\mathbf{b} + \beta r \mathbf{z}$
  $\mathbf{A} \leftarrow (1-\beta)\mathbf{A} + \beta \mathbf{z}(\mathbf{x}_{s,a} - \gamma \mathbf{x}_{s',a'})^\top$
  ▷ Update $\mathbf{B} \approx \mathbf{A}^{-\top}$
  $\alpha = \min\left\{1.0, \frac{0.01}{||\mathbf{A}||_F^2 ||\mathbf{x}_{s,a}||_2^2 + 1.0}\right\}$
  $\mathbf{B} \leftarrow \mathbf{B} - \alpha \mathbf{A}(\mathbf{A}^\top \mathbf{B}\mathbf{x}_{s,a} - \mathbf{x}_{s,a})\mathbf{x}_{s,a}^\top$
  ▷ Update $\mathbf{C}$
  $\bar{\boldsymbol{\nu}} \leftarrow (1-\beta)\bar{\boldsymbol{\nu}} + \beta\delta\mathbf{z}$
  $\mathbf{a} \leftarrow \mathbf{B}^\top \bar{\boldsymbol{\nu}}$
  temp $= c_{\max}$
  $c_{\max} = \max(c_{\max}, \mathbf{a}_1^2, \ldots, \mathbf{a}_d^2)$
  if temp $\neq c_{\max}$ **then** ▷ Adjust initialization
    $\mathbf{C}_{ii} \leftarrow \mathbf{C}_{ii} + \mathbf{c}_i(c_{\max} - \text{temp}), \forall i$
  for i such that $\mathbf{z}_i \neq 0$ **do**
    $\mathbf{c}_i = \mathbf{c}_i(1-\beta)$
    for j such that $\mathbf{z}_j \neq 0$ **do**
      $\mathbf{C}_{ij} \leftarrow (1-\beta)\mathbf{C}_{ij} + \beta \mathbf{a}_i \mathbf{a}_j$
  ▷ Update $\mathbf{w}$
  $\mathbf{w} \leftarrow \mathbf{w} + (\mathbf{B}^\top + \eta\mathbf{I})(\mathbf{b} - \mathbf{A}\mathbf{w})$
  $\mathbf{x}_{s,a} \leftarrow \mathbf{x}_{\mathbf{s}',a'}$ and $a \leftarrow a'$
until agent done interaction with environment

C Estimating Upper Confidence Bounds for Policy Evaluation using linear TD

Recall that the TD update [39] processes one sample at a time as $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\delta_t\mathbf{z}_t$ to estimate the solution to the least-squares system $\mathbf{w}_T = \mathbf{A}_T^{-1}\mathbf{b}_T$ in an incremental manner. This is feasible as the following holds:

$$\begin{aligned}
\mathbf{w}_T &= \mathbf{A}_T^{-1}\mathbf{b}_T \\
\mathbf{A}_T\mathbf{w}_T &= \mathbf{b}_T \\
\left[\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{z}_t(\mathbf{x}_t - \gamma_{t+1}\mathbf{x}_{t+1})^\top\right]\mathbf{w}_T &= \left[\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{z}_t r_t\right] \\
\sum_{t=0}^{T-1}\mathbf{z}_t(r_t + \gamma_{t+1}\mathbf{x}_{t+1}^\top\mathbf{w}_T - \mathbf{x}_t^\top\mathbf{w}_T) &= 0 \\
\sum_{t=0}^{T-1}\mathbf{z}_t\delta_t &= 0
\end{aligned}$$

Therefore, $\mathbf{w}_t$ is updated incrementally with a constant step-size towards minimizing this error stochastically.

Given this incremental method to estimate a least-squares solution, we can notice that the covariance matrix is the outer-product of the solution to a similar least-squares system, $\mathbf{A}_T^{-1}\bar{\boldsymbol{\nu}}_T$. The solution to this least-squares system is denoted by $\mathbf{w}_{\text{var}}$, and can be estimated incrementally as:

$$\mathbf{w}_{\text{var}\,t+1} = \mathbf{w}_{\text{var}\,t} + \alpha\delta_{\text{var}\,t}\mathbf{z}_t$$

where, $\delta_{\text{var}\,t} = \delta_t + \gamma_{t+1}\mathbf{x}_{t+1}^\top\mathbf{w}_{\text{var}\,t} - \mathbf{x}_t^\top\mathbf{w}_{\text{var}\,t}$.

Therefore, for a given policy, the true action-values satisfy the following:

$$\mathbf{x}^\top\mathbf{w}^* \leq \mathbf{x}^\top\mathbf{w}_T + \sqrt{\tfrac{p+1}{p}}\sqrt{\mathbf{x}^\top\mathbf{w}_{\text{var}\,T}\mathbf{w}_{\text{var}\,T}^\top\mathbf{x}}$$

Similarly a linear variant of GV-UCB can be obtained as the upper bound again consists of an outer-product to a different least-squares system $\mathbf{A}_T^{-1}\bar{\mathbf{z}}_T$. But as shown in Figures 2 and 7, GV-UCB, the quadratic version, can be highly sample inefficient, which may worsen with the linear variant, GV-UCB-L. Therefore, we do not provide an algorithm, or any empirical results for GV-UCB-L here.

D UCLS-L: Estimating upper confidence bounds for linear TD in control

In the same spirit as UCLS utilizes the policy evaluation upper-bound of LSTD for control, with a slowly changing control policy, UCLS-L utilizes the policy evaluation upper-bound of linear TD for control. At each step, UCLS-L, given in Algorithm 4, uses a stochastic update to estimate mean action-values, and their corresponding contextual-variance estimates. These stochastic updates use fixed, and if necessary are different, step-sizes (α, and α_{var} respectively), instead of a closed-form solution as done by UCLS. The rate of change of the policy in UCLS-L is controlled by the step-size, unlike in UCLS which utilizes weighted forms of experience samples in $\mathbf{A}$ and $\mathbf{b}$. Therefore, UCLS-L can be sensitive to the step-sizes, but adapt more quickly to a changing feature-space. Further, in order to account for underestimates of variances, UCLS-L uses another vector $\mathbf{w}_{\text{varInit}}$, in a similar spirit as UCLS's retroactive initialization of covariance estimates. Additionally, as these upper-bounds are estimated incrementally, they can be quite loose, specifically so in the linear framework. Therefore, instead of choosing the best parameter p, we can choose a parameter $\bar{p} = \sqrt{1 + \frac{1}{p}}$: the loss of theoritical interpretation of the upper-bound is traded-off for better empirical performance.

With this, we investigate UCLS-L as a substitue to UCLS in the three benchmark domains. For UCLS-L, both p and $\bar{p}$ is swept, from which the best parameter is selected scale the uncertainity unstemiate, along with the learning rates α and α_{var}. The experiment configuration and the domains are the same as used in UCLS. The results are presented in Figure 6. UCLS-L does reasonably

Algorithm 3 GetOptimisticActionLinear($\mathbf{x}_{s,\cdot}$)

$u_a \leftarrow \sqrt{\left(1+\frac{1}{p}\right)\left((\mathbf{x}_{s,a}^\top \mathbf{w}_{\text{var}})^2 + ||\mathbf{x}_{s,a}||^2_{\mathbf{I}\mathbf{w}_{\text{varInit}}}\right)} \quad \forall a \in \mathcal{A}$
$a = \text{argmax}_{a\in\mathcal{A}}\, \mathbf{x}_{s,a}^\top \mathbf{w} + u_a$
return a

Algorithm 4 UCLS-L(λ)

$p = 0.1$, $\beta = 0.001$, $v_{\text{init}} = 1.0$, $\alpha = 0.01$, $\alpha_{\text{var}} = 0.1$
$\mathbf{w} \leftarrow \mathbf{0}$, $\mathbf{w}_{\text{var}} \leftarrow \mathbf{0}$, $\mathbf{w}_{\text{varInit}} \leftarrow \mathbf{1} * v_{\text{init}}$, $\mathbf{c} \leftarrow \mathbf{1}$
$\mathbf{x}_{s,\cdot} \leftarrow$ initial state-action features, for any action
$a \leftarrow$ GetOptimisticActionLinear($\mathbf{x}_{s,\cdot}$)
repeat
  Take action a and observe $\mathbf{x}_{s',\cdot}$ and r, and γ
  $a' \leftarrow$ GetOptimisticActionLinear($\mathbf{x}_{s',\cdot}$)
  $\delta \leftarrow r + (\gamma \mathbf{x}_{s',a'} - \mathbf{x}_{s,a})^\top \mathbf{w}$
  $\delta_{\text{var}} \leftarrow \delta + (\gamma \mathbf{x}_{s',a'} - \mathbf{x}_{s,a})^\top \mathbf{w}_{\text{var}}$
  $\mathbf{z} \leftarrow \gamma\lambda \mathbf{z} + \mathbf{x}_{s,a}$
  ▷ Update $\mathbf{w}_{\text{var}}$ and $\mathbf{w}_{\text{varInit}}$
  $\mathbf{w}_{\text{var}} \leftarrow \mathbf{w}_{\text{var}} + \alpha_{\text{var}} \delta_{\text{var}} \mathbf{z}$
  temp $= v_{\text{init}}$
  $v_{\text{init}} = \max(v_{\text{init}}, \mathbf{w}_{\text{var}}{}_1^2, \ldots, \mathbf{w}_{\text{var}}{}_d^2)$
  if temp $\neq v_{\text{init}}$ **then** ▷ Adjust initialization
    $\mathbf{w}_{\text{varInit}\,i} \leftarrow \mathbf{w}_{\text{varInit}\,i} + \mathbf{c}_i(v_{\text{init}} - \text{temp}), \forall i$
  for i such that $\mathbf{z}_i \neq 0$ **do**
    $\mathbf{c}_i = \mathbf{c}_i(1-\beta)$
    $\mathbf{w}_{\text{varInit}\,i} \leftarrow (1-\beta) * \mathbf{w}_{\text{varInit}\,i}, \forall i$
  ▷ Update $\mathbf{w}$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\mathbf{z}$
  $\mathbf{x}_{s,a} \leftarrow \mathbf{x}_{\mathbf{s}',a'}$ and $a \leftarrow a'$
until agent done interaction with environment

well in all the domains. While it experiences more regret in Puddle World, and River Swim during early learning, by the end of the steps budget, it learns the optimal policy. In Sparse Mountain Car, surprisingly, UCLS-L learns much faster and a better policy than UCLS. This can be attributed to the fact that the parameter p in UCLS was not swept, whereas in UCLS-L we did sweep to find the best parameter to scale the variance estimate. As the domain is a sparse-reward domain, the variance estimates play a significant role in influencing exploratory behaviour, and therefore optimizing for p would improve UCLS' performance. Nonetheless, these results show UCLS-L to be a promising algorithm for linear complexity based control, and warrant further evaluation of it.

Algorithm 5 GetOptimisticActionGlobal($\mathbf{x}_{s,\cdot}$)

$u_a \leftarrow \sigma\sqrt{\left(1+\frac{1}{p}\right)\mathbf{x}_{s,a}^\top \mathbf{C} \mathbf{x}_{s,a}} \quad \forall a \in \mathcal{A}$
$a = \text{argmax}_{a\in\mathcal{A}}\, \mathbf{x}_{s,a}^\top \mathbf{w} + u_a$
return a

E Details about other algorithms

E.1 Global variance UCB

Based on Corollary 1 to estimate a global variance σ^2, it is possible that the noise may not be 0-mean during the learning process. We account for this by estimating mean of ν_t as well. We know

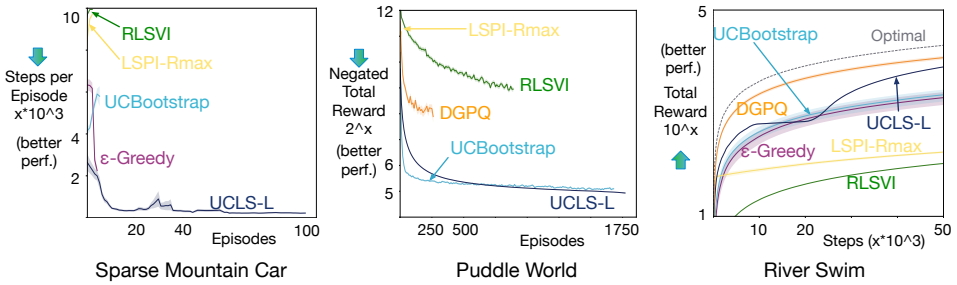

Figure 6: A comparison of speed of learning in Sparse Mountain Car, Puddle World and River Swim. In plots (a) and (b) lower on y-axis are better, whereas in (c) curves higher along y-axis are better. Sparse Mountain Car and Puddle World are episodic problems with a fixed experience budget. Thus the length of the lines in plots (a) and (b) indicate how many episodes each algorithm completed over 50,000 steps, and the height on the y-axis indicates the quality of the learned policy—lower indicates better performance. Note RLSVI did not show significant learning after 50,000 steps. The RLSVI result in Puddle World uses a budget of 1 million.

$\nu_t \sim \mathcal{N}(\bar{\nu}_t, \sigma_t^2)$. Therefore:

$$\begin{aligned}
\bar{\nu}_{t+1} &= E[r_{t+1}] && - E[\mathbf{x}_t - \gamma \mathbf{x}_{t+1}]^\top \mathbf{w}_t \\
\bar{\nu^2}_{t+1} &= E[r_{t+1}^2] && - 2E[r_{t+1}(\mathbf{x}_t - \gamma \mathbf{x}_{t+1})]^\top \mathbf{w}_t \\
& && + \mathbf{w}_t^\top E[(\mathbf{x}_t - \gamma \mathbf{x}_{t+1})(\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^\top] \mathbf{w}_t
\end{aligned}$$

These expected values are maintained incrementally. Utilizing this, $\sigma_{t+1}^2 = \bar{\nu^2}_{t+1} - \bar{\nu}_{t+1}^2$. We refer to Global variance UCB as GV-UCB. The algorithm is given in Algorithm 6.

E.2 Bootstrapped upper confidence bounds

The strategy for action selection which utilizes bootstrapped confidence intervals, as proposed by White and White [46], is given in Algorithm 7. This action selection strategy can be used in conjunction with any learning algorithm to guide on-policy control. The algorithm requires a window of recent $\mathbf{w}$'s. The window can be maintained with a circular queue. The window is updated after each learning step of the main algorithm, resulting in a new $\mathbf{w}_t$ in the queue. The original UCBootstrap paper proposed both a global and a sparse updating mechanism, where only the global approach was theoretically justified. The sparse mechanism was used to reduce the number of parameters stored, particularly by taking advantage of tile-coding representations. We found in our experiments that the global approach worked just as well as the sparse approach, and so we include only the simpler, theoretically justified algorithm.

E.3 DGPQ

Another approach to exploration is found in a model-free algorithm using gaussian processes named Delayed-GPQ (DGPQ) [8]. The pseudocode for DGPQ is in Algorithm 8. Any algorithm can be used to train the Gaussian processes, and for this paper we use the same algorithm as in [8]. The initialization of this algorithm requires the maximum reward and value, but for ease of use we transform the reward signal to $r_{new} = r - R_{max}$ so the means of the gaussian processes can be initialized to zero and $V_{max} = 0$.

A major problem with DGPQ is the large number of parameters needed to be set properly. Some intuition on setting these parameters can be found in [8] as well as in algorithm 8. As some guidance the width of the kernel determines how much a sample can generalize to other states, the thresholds $(\sigma_{tol}^2, \epsilon)$ determine how often we swap for new experience in the set basis vectors, and the Lipschitz constant L_Q tunes the tradeoff between exploration and exploitation.

E.4 LSPI-Rmax

LSPI-Rmax [19] combines LSPI [17] with Rmax [6] for online control in continuous state-spaces. Exploration is encouraged by determining the *knowness* of a transition, utilizing kernels. LSPI

Algorithm 6 GV-UCB(λ)

$\mathbf{A} \leftarrow \mathbf{0}$, $\mathbf{b} \leftarrow \mathbf{0}$, $\mathbf{z} \leftarrow \mathbf{0}$, $\mathbf{w} \leftarrow \mathbf{0}$,
$\mathbf{B} \leftarrow \mathbf{I}$, $\mathbf{C} \leftarrow \mathbf{I}$, $\bar{\mathbf{z}} \leftarrow \mathbf{0}$
$p = 0.01$,$\eta = 10^{-4}$, $\beta = 0.001$
$\sigma = 1.0$, $\bar{r} = 0.0$, $\bar{r^2} = 100.0$, $\bar{\mathbf{d}} \leftarrow \mathbf{0}$, $\bar{\mathbf{d}_\mathbf{r}} \leftarrow \mathbf{0}$, $\bar{\mathbf{D}} \leftarrow \mathbf{0}$
$\mathbf{x}_{s,\cdot} \leftarrow$ initial state-action features, for any action
$a \leftarrow$ GetOptimisticActionGlobal($\mathbf{x}_{s,\cdot}$)
repeat
  Take action a and observe $\mathbf{x}_{s',\cdot}$ and r, and γ
  $a' \leftarrow$ GetOptimisticActionGlobal($\mathbf{x}_{s',\cdot}$)
  $\delta \leftarrow r + (\gamma \mathbf{x}_{s',a'} - \mathbf{x}_{s,a})^\top \mathbf{w}$
  $\mathbf{z} \leftarrow \gamma\lambda\mathbf{z} + \mathbf{x}_{s,a}$
  $\mathbf{b} \leftarrow (1-\beta)\mathbf{b} + \beta r \mathbf{z}$
  $\mathbf{A} \leftarrow (1-\beta)\mathbf{A} + \beta \mathbf{z}(\mathbf{x}_{s,a} - \gamma \mathbf{x}_{s',a'})^\top$
  ▷ Update $\mathbf{C}$
  $\bar{\mathbf{z}} \leftarrow (1-\beta)\bar{\mathbf{z}} + \beta \mathbf{z}$
  $\mathbf{a} \leftarrow \mathbf{B}^\top \bar{\mathbf{z}}$
  for i such that $\mathbf{z}_i \neq 0$ **do**
    for j such that $\mathbf{z}_j \neq 0$ **do**
      $\mathbf{C}_{ij} \leftarrow (1-\beta)\mathbf{C}_{ij} + \beta \mathbf{a}_i \mathbf{a}_j$
  ▷ Update σ
  $\bar{r} \leftarrow (1-\beta)\bar{r} + \beta r$
  $\bar{r^2} \leftarrow (1-\beta)\bar{r^2} + \beta r^2$
  $\bar{\mathbf{d}} \leftarrow (1-\beta)\bar{\mathbf{d}} + \beta(\mathbf{x}_{s,a} - \gamma \mathbf{x}_{s',a'})$
  $\bar{\mathbf{d}_\mathbf{r}} \leftarrow (1-\beta)\bar{\mathbf{d}_\mathbf{r}} + \beta r(\mathbf{x}_{s,a} - \gamma \mathbf{x}_{s',a'})$
  $\bar{\mathbf{D}} \leftarrow (1-\beta)\bar{\mathbf{D}} + \beta(\mathbf{x}_{s,a} - \gamma \mathbf{x}_{s',a'})(\mathbf{x}_{s,a} - \gamma \mathbf{x}_{s',a'})^\top$
  $\bar{\nu} = \bar{r} - \bar{\mathbf{d}}^T \mathbf{w}$
  $\bar{\nu^2} = \bar{r^2} - 2\bar{\mathbf{d}_\mathbf{r}}^T \mathbf{w} + \mathbf{w}^\top \bar{\mathbf{D}} \mathbf{w}$
  $\sigma = \sqrt{\bar{\nu^2} - \bar{\nu}^2}$
  ▷ Update $\mathbf{w}$ and $\mathbf{B} \approx \mathbf{A}^{-\top}$
  $\alpha = \min\left\{1.0, \frac{0.01}{||\mathbf{A}||_F^2 ||\mathbf{x}_{s,a}||_2^2 + 1.0}\right\}$
  $\mathbf{B} \leftarrow \mathbf{B} - \alpha \mathbf{A}(\mathbf{A}^\top \mathbf{B} \mathbf{x}_{s,a} - \mathbf{x}_{s,a})\mathbf{x}_{s.a}^\top$
  $\mathbf{w} \leftarrow \mathbf{w} + (\mathbf{B} + \eta \mathbf{I})(\mathbf{b} - \mathbf{A}\mathbf{w})$
  $\mathbf{x}_{s,a} \leftarrow \mathbf{x}_{\mathbf{s}',a'}$ and $a \leftarrow a'$
until agent done interaction with environment

algorithm is designed for a batch setting, where the LSTD solution is computed in closed form for staged batches of data. However, because it accumulates optimistic values, it can be simply converted into an online algorithm using incremental updates to the matrix $\mathbf{A}$ and $\mathbf{b}$, as done in Li et al. [19].

We summarize this extension in pseudocode as Algorithm 10. Until states become known, the algorithm estimates action-values that predict the maximum possible return; once a state becomes known, it starts to use actual rewards sampled from the environment. To estimate the *knowness* of a state under function approximation, we use feature counts. Each state has a set of active features; the active feature with the minimum count reflects an upper bound on the number of times that this state has been seen. Once a states active features have been seen frequently enough, it becomes known.

E.5 RLSVI

RLSVI [31] is an algorithm that maintains a distribution over the possible value functions. The value functions are assumed to be linearly parametrized. While the main algorithm proposed uses a finite-horizon assumption, a modified version proposed in the Appendix of the paper does not, and this is the version used in the experiments here.

Algorithm 7 UCBootstrap($\mathbf{x}_{s,\cdot}$) select action from state features $\mathbf{x}_{s,\cdot}$ at time t

l = block length, B = number of bootstrap resamples, w = number (window) of value functions weights to store and confidence level α
examples: $l = 10$, $B = 50$, $w = 100$, $\alpha = 0.05$

$M \leftarrow \lfloor w/l \rfloor$ ▷ num of length l blocks to sample with replacement and concatenate
for each action a **do**
  $Q_N \leftarrow \{\mathbf{w}_{t-w}^\top \mathbf{x}_{s,a}, \ldots, \mathbf{w}_{t-1}^\top \mathbf{x}_{s,a}\}$
  $\bar{Q}_N \leftarrow$ mean(Q_N) ▷ The mean value for this (s, a), given the window of recent weights
  Blocks = $\Big\{\{[Q_N[0], \ldots, Q_N[l\text{-}1]\}, \{[Q_N[1], \ldots, Q_N[l]\}, \ldots, [Q_N[w\text{-}l], \ldots, Q_N[w\text{-}1]]\Big\}$
  for all $i = 1$ to B **do**
    for all $j = 1$ to M **do**
      $A_j^* \leftarrow$ random block from Blocks (chosen with replacement)
    $A \leftarrow (A_1^*, A_2^*, \ldots, A_M^*)$ ▷ Concatenate blocks
    $T_i^* = \frac{1}{lM} \sum_{k=1}^{lM} A[k]$ ▷ ith bootstrap estimate is the mean of the M concatenated blocks
  $T \leftarrow$ sort($\{T_1^*, \ldots, T_B^*\}$) ▷ ascending order
  $j \leftarrow \lfloor \frac{B\alpha}{2} + \frac{\alpha+2}{6} \rfloor$ ▷ j is the position of the critical samples to help estimate the continuous sample quantile
  $r \leftarrow \frac{B\alpha}{2} + \frac{\alpha+2}{6} - j$ ▷ r is the remainder
  $T_{\alpha/2}^* \leftarrow (1-r)T_j^* + rT_{j+1}^*$ ▷ the $\alpha/2$ sample quantile
  $u_a \leftarrow 2\bar{Q}_N - T_{\alpha/2}^*$
$a = \text{argmax}_{a \in \mathcal{A}} u_a$
return a

F Alternative updates for LSTD

The update for $\mathbf{w}$ using $\mathbf{A}$ and $\mathbf{b}$ in UCLS is the result of an empirical investigation into alternative linear system solvers. We investigated using a Sherman-Morrison update, with exponential averaging (in Algorithm 11) as well as improved incremental inverse updates, including one for pseudo-inverses [23]. This update has a confounding role for η, and for small η we found it less stable than our proposed update. We investigated iterative updates with a fixed stepsize, $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha(\mathbf{b}_t - \mathbf{A}_t\mathbf{w})$; the addition of the step-size, however, removes some of the parameter-free benefits of LSTD. We investigated conjugate gradient updates, as in Algorithm 12. We finally derived the iterative update proposed, for $\mathbf{B} \approx \mathbf{A}^{-\top}$, to obtain a preconditioner for the iterative update.

For completeness, we include the derivation for the Sherman-Morrison update. The derivation for $\mathbf{A}_{t+1}^{-1}$ using $\mathbf{A}_{t+1} = (1-\beta)\mathbf{A}_t + \beta uv^T$ is as follows:

$$\mathbf{A}_t^{-1}\mathbf{A}_{t+1} = (1-\beta)\mathbf{I} + \beta\mathbf{A}_t^{-1}uv^T$$

Converting it to terms of $\mathbf{A}_{t+1}^{-1}$:

$$\begin{aligned}\mathbf{A}_{t+1}^{-1} &= ((1-\beta)\mathbf{I} + \beta\mathbf{A}_t^{-1}uv^T)^{-1}\mathbf{A}_t^{-1} \\ &= \left(\frac{1}{(1-\beta)}\mathbf{I} + \frac{1}{1+\frac{\beta}{1-\beta}v^T\mathbf{A}_t^{-1}u}\frac{\beta}{(1-\beta)^2}\mathbf{A}_t^{-1}uv^T)\right)\mathbf{A}_T^{-1} \\ &= \frac{1}{(1-\beta)}\mathbf{A}_t^{-1} + \frac{\frac{\beta}{(1-\beta)}\mathbf{A}_t^{-1}uv^T\mathbf{A}_t^{-1}}{(1-\beta)+\beta v^T\mathbf{A}_t^{-1}u}\end{aligned}$$

The first step utilizes a Lemma in [24].

G Extended results

We show additional results here comparing UCLS to Sarsa with optimistic initialization and GV-UCB (p=0.5), Figure 7; along with DGPQ in MCSparse. Also included are plots showing best and worst

Algorithm 8 DGPQ($k(\cdot,\cdot), d(\cdot,\cdot), L_Q, Env, \mathcal{A}, R_{max}, s_0, \gamma, \sigma^2, \sigma^2_{tol}, \epsilon$)

$k(\cdot,\cdot), d(\cdot,\cdot)$ are typically the RBF w/ bandwidth = σ^2 and euclidean distance respectively.
L_Q correlates with exploration.
$\mathcal{A}$ is the set of possible actions.
γ is the discount factor.
σ^2_{tol} is the tolerance of induced variance of using a new point to update a GP
Found useful ranges for parameters during sweeps:
$\sigma^2 \in [0.001, 0.5]$,$\sigma^2_{tol} \in [0.01, 0.1]$, $\epsilon \in [0.01, 0.1]$, $L_Q \in [1, 20]$

$\hat{Q}(s,a) \stackrel{\text{def}}{=} \min($
$\quad V_{max},$
$\quad \min_{(s_i,a)\in\hat{Q}_a.BV} \{[\hat{\mu}_i + L_Q d((s,a),(s_i,a))]\}$
$)$
for $a \in A$ **do**
  $\hat{Q}_a.BV = \emptyset$
  $GP_a = GP.init(\mu = \frac{Rmax}{1-\gamma}, k(\cdot,\cdot))$
for $t \in [0, T]$ **do**
  $a_t = \operatorname{argmax}_a \hat{Q}(s,a)$
  //take action a_t in state s_t, observe (s_{t+1}, r_t)
  $(s_{t+1}, r_t) = Env(s_t, a_t)$
  $q_t = r_t + \gamma \max_a \hat{Q}(s_{t+1})$
  $\sigma^2_1 = GP_{a_t}.variance(s_t)$
  //If the new sample is not well covered by GP_{a_t}
  if $\sigma^2_1 > \sigma^2_{tol}$ **then**
    $GP_{a_t}.update(s_t, q_t)$
  $\sigma^2_2 = GP_{a_t}.variance(s_t)$
  //If the GP_{a_t} now well covers a previously unknown state and the new approximation is 2ϵ less than what is found in $\hat{Q}$ (i.e. is a less optimistic estimate).
  if $\left\{\sigma^2_1 > \sigma^2_{tol} \geq \sigma^2_2\right\}$ **and** $\left\{\hat{Q}_{a_t}(s_t) - GP_{a_t}.mean(s_t) > 2\epsilon\right\}$ **then**
    $\mu = GP_{a_t}.mean(s_t) + \epsilon$
    $\hat{Q}_{a_t}.BV.add((s_t, a_t), \mu)$
    for $((s_j, a_t), \mu_j) \in \hat{Q}_{a_t}.BV$ **do**
      if $\mu_j \leq \mu + L_Q d((s_t, a_t), (s_j, a_t))$ **then**
        $\hat{Q}_{a_t}.BV.delete(((s_j, a_t), \mu_j))$
    //To prevent slow learning or halted learning reset the current GPs and initialize to the current estimates.
    $\forall a \in A, GP_a = GP.init(\hat{\mu} = \hat{Q}_a, k(\cdot,\cdot))$

Algorithm 9 IsKnown(s, a)

// Uses the minimum count of the features for a state, to decide if s, a is known
// If a not given, sums over all a
$m = 5$
if a not given **then**
  $\mathbf{f} \leftarrow \sum_a \mathbf{c}(\mathbf{x}_{s,a}) \in \mathbb{R}^d$
else
  $\mathbf{f} \leftarrow \mathbf{c}(\mathbf{x}_{s,a}) \in \mathbb{R}^d$
if $\min(\mathbf{f}) > m$ **then**
  return "Known"
else
  return "Not Known"

Algorithm 10 Incremental LSPI-Rmax(m)

$\mathbf{A} \leftarrow \mathbf{0}$, $\mathbf{b} \leftarrow \mathbf{0}$, $\mathbf{z} \leftarrow \mathbf{0}$, $\mathbf{w} \leftarrow \mathbf{0}$,
$\mathbf{B} \leftarrow \mathbf{I}$, $\mathbf{c} \leftarrow \mathbf{0}$
$\eta = 10^{-4}$, $\beta = 0.001$, $\lambda = 0$, $G_{\text{max}} = r_{\text{max}}/(1-\gamma)$ if continuing or $\gamma \neq 1$, else $G_{\text{max}} = r_{\text{max}} h$ for a predicted maximum episode length (e.g., $h = 10000$).
$\mathbf{x}_{s,\cdot} \leftarrow$ initial state-action features, for any action
$a \leftarrow$ greedy action according to value estimates given by $\mathbf{x}_{s,a}^\top \mathbf{w}$
repeat
    Take action a and observe $\mathbf{x}_{s',\cdot}$ and r, and γ
    $a' \leftarrow$ greedy action according to value estimates given by $\mathbf{x}_{s',a'}^\top \mathbf{w}$
    $\mathbf{z} \leftarrow \gamma\lambda \mathbf{z} + \mathbf{x}_{s,a}$
    if IsKnown(s, a) **then**
        if IsKnown(s') **then**
            $\mathbf{A} \leftarrow (1-\beta)\mathbf{A} + \beta \mathbf{z}(\mathbf{x}_{s,a} - \gamma \mathbf{x}_{s',a'})^\top$
            $\mathbf{b} \leftarrow (1-\beta)\mathbf{b} + \beta r \mathbf{z}$
        else
            $\mathbf{A} \leftarrow (1-\beta)\mathbf{A} + \beta \mathbf{x}_{s,a}\mathbf{x}_{s,a}^\top$
            $\mathbf{b} \leftarrow (1-\beta)\mathbf{b} + \beta(r + \gamma G_{\text{max}})\mathbf{x}_{s,a}$
    else
        $\mathbf{A} \leftarrow (1-\beta)\mathbf{A} + \beta \mathbf{x}_{s,a}\mathbf{x}_{s,a}^\top$
        $\mathbf{b} \leftarrow (1-\beta)\mathbf{b} + \beta G_{\text{max}} \mathbf{x}_{s,a}$
    for $\forall \tilde{a} \in A \backslash a$ **do**
        if !IsKnown$(s, \tilde{a})$ **then**
            $\mathbf{A} \leftarrow (1-\beta)\mathbf{A} + \beta \mathbf{x}_{s,\tilde{a}}\mathbf{x}_{s,\tilde{a}}^\top$
            $\mathbf{b} \leftarrow (1-\beta)\mathbf{b} + \beta G_{\text{max}} \mathbf{x}_{s,\tilde{a}}$
    $\mathbf{c} \leftarrow \mathbf{c} + \mathbf{x}_{s,a}$
    $\alpha = \min\left\{1.0, \frac{0.01}{||\mathbf{A}||_F^2 ||\mathbf{x}_{s,a}||_2^2 + 1.0}\right\}$
    $\mathbf{B} \leftarrow \mathbf{B} - \alpha \mathbf{A}(\mathbf{A}^\top \mathbf{B} \mathbf{x}_{s,a} - \mathbf{x}_{s,a})\mathbf{x}_{s,a}^\top$
    $\mathbf{w} \leftarrow \mathbf{w} + (\mathbf{B} + \eta \mathbf{I})(\mathbf{b} - \mathbf{A}\mathbf{w})$
    $\mathbf{x}_{s,a} \leftarrow \mathbf{x}_{\mathbf{s}',a'}$ and $a \leftarrow a'$
until agent done interaction with environment

Algorithm 11 LSTD(λ) with Sherman-Morrison update

$\mathbf{A}^{-1} \leftarrow \frac{1}{\eta}\mathbf{I}$, $\mathbf{b} \leftarrow \mathbf{0}$, $\mathbf{z} \leftarrow \mathbf{0}$, $\mathbf{w} \leftarrow \mathbf{0}$,
$\mathbf{x}_{s,\cdot} \leftarrow$ initial state-action features, for any action
$a \leftarrow \epsilon$-greedy action according to value estimates given by $\mathbf{x}_{s,a}^\top \mathbf{w}$
repeat
    Take action a and observe $\mathbf{x}_{s',\cdot}$ and r, and γ
    $a' \leftarrow \epsilon$-greedy action according to value estimates given by $\mathbf{x}_{s',a'}^\top \mathbf{w}$
    $\mathbf{z} \leftarrow \gamma\lambda \mathbf{z} + \mathbf{x}_{s,a}$
    $\beta = \frac{1}{t}$
    $\mathbf{b} \leftarrow \mathbf{b} + \beta(r\mathbf{z} - \mathbf{b})$
    $\mathbf{v} \leftarrow \left((\mathbf{x}_{s,a} - \gamma \mathbf{x}_{s',a'})^\top \mathbf{A}^{-1}\right)^\top$
    $\mathbf{A}^{-1} \leftarrow \frac{1}{(1-\beta)}\mathbf{A}^{-1} + \frac{\frac{\beta}{(1-\beta)}\mathbf{A}^{-1}\mathbf{z}\mathbf{v}^\top}{(1-\beta) + \beta \mathbf{v}^\top \mathbf{z}}$
    $\mathbf{w} \leftarrow \mathbf{A}^{-1}\mathbf{b}$
    $\mathbf{x}_{s,a} \leftarrow \mathbf{x}_{\mathbf{s}',a'}$ and $a \leftarrow a'$
until agent done interaction with environment

Algorithm 12 LSTD(λ) with Conjugate Gradient

$\mathbf{A} \leftarrow \mathbf{0}$, $\mathbf{b} \leftarrow \mathbf{0}$, $\mathbf{z} \leftarrow \mathbf{0}$, $\mathbf{w} \leftarrow \mathbf{0}$,
$\mathbf{x}_{s,\cdot} \leftarrow$ initial state-action features, for any action
$a \leftarrow \epsilon$-greedy action according to value estimates given by $\mathbf{x}_{s,a}^\top \mathbf{w}$
repeat
  Take action a and observe $\mathbf{x}_{s',\cdot}$ and r, and γ
  $a' \leftarrow \epsilon$-greedy action according to value estimates given by $\mathbf{x}_{s',a'}^\top \mathbf{w}_{a'}$
  $\mathbf{z} \leftarrow \gamma\lambda\mathbf{z} + \mathbf{x}_{s,a}$
  $\beta_t = \frac{1}{t}$ ▷ or constant such as $\beta_t = 0.01$
  $\mathbf{b} \leftarrow \mathbf{b} + \beta_t(r\mathbf{z} - \mathbf{b})$
  $\mathbf{A} \leftarrow \mathbf{A} + \beta_t(\mathbf{z}(\mathbf{x}_{s,a} - \gamma\mathbf{x}_{s',a'})^\top - \mathbf{A})$
  $\mathbf{w} \leftarrow$ ConjugateGradient$(\mathbf{A} + \eta\mathbf{I}, \mathbf{b}, \mathbf{w}, \eta_r)$
  $\mathbf{x}_{s,a} \leftarrow \mathbf{x}_{\mathbf{s}',a'}$ and $a \leftarrow a'$
until agent done interaction with environment

Algorithm 13 Conjugate Gradient$(\mathbf{A}, \mathbf{b}, \mathbf{w}, \eta_r)$

tol $= 0.001$
$\tilde{\mathbf{A}} = \mathbf{A}^T\mathbf{A} + \eta_r\mathbf{I}$
$\mathbf{r} \leftarrow \mathbf{b} - \tilde{\mathbf{A}}\mathbf{w}$
$\mathbf{d} \leftarrow \mathbf{r}$
repeat
  $\alpha \leftarrow \frac{\mathbf{r}^\top\mathbf{r}}{\mathbf{d}^\top\tilde{\mathbf{A}}\mathbf{d}}$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha\mathbf{d}$
  $\mathbf{r}' \leftarrow \mathbf{r} - \alpha\tilde{\mathbf{A}}\mathbf{d}$
  $\beta \leftarrow \frac{\mathbf{r}'^\top\mathbf{r}'}{\mathbf{r}^\top\mathbf{r}}$
  $\mathbf{d} \leftarrow \mathbf{r}' + \beta\mathbf{d}$
until CG converged ($||\mathbf{r}'||_2^2 \leq$ tol) or a fixed number of steps reached
return $\mathbf{w}$

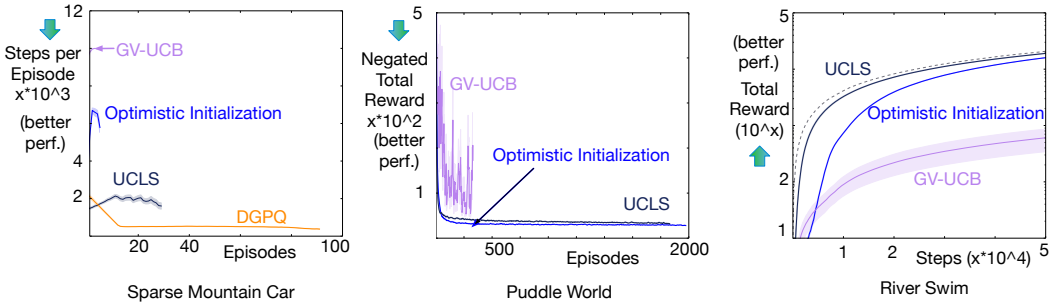

Figure 7: Learning curves in the three domains comparing UCLS to additional methods. In the first two plots lower on y-axis indicates better performance, whereas in the right-most plot higher along y-axis is better.

runs for UCLS and DGPQ — the two closest competitors — to show the variance of each algorithm Figure 8. Additionally, to empirically reinforce the utility of contextual confidence interval radius (CIR) over global CIR, we evaluate the policies obtained by UCLS and GV-UCB after 50,000 learning samples in River Swim and present the results in Figure 9.

As mentioned in the main results, Sarsa with optimistic initialization performs remarkably well in these domains. In Sparse Mountain Car, as DGPQ converts the sparse reward dynamics to a dense one, it outperforms Sarsa with optimistic initialization as well. In Puddle World, UCLS matches up to Sarsa's policy. In River Swim UCLS experiences minimal regret when compared to Sarsa's control policy. With the loss of contextual variance estimates GV-UCB explores the complete state space more thoroughly, and therefore performs poorly. The explicit upper confidence bound given by

UCLS does not suffer from this, and sufficiently explores the domain to converge to an optimal policy without excessively exploring. For regions where there is low variance, the upper-confidence-bound converges more quickly to zero, whereas it remains higher in regions of uncertainty. Therefore, contextual variance estimates provide the flexibility of variable convergence based on the variance of the region, and global variance estimates decay too slowly. When the policies obtained by GV-UCB and UCLS are evaluated, it is clear that the policy obtained by UCLS is much closer to the optimal policy than the policy obtained by GV-UCB, showing that the exploration strategy used by UCLS is more *data efficient*.

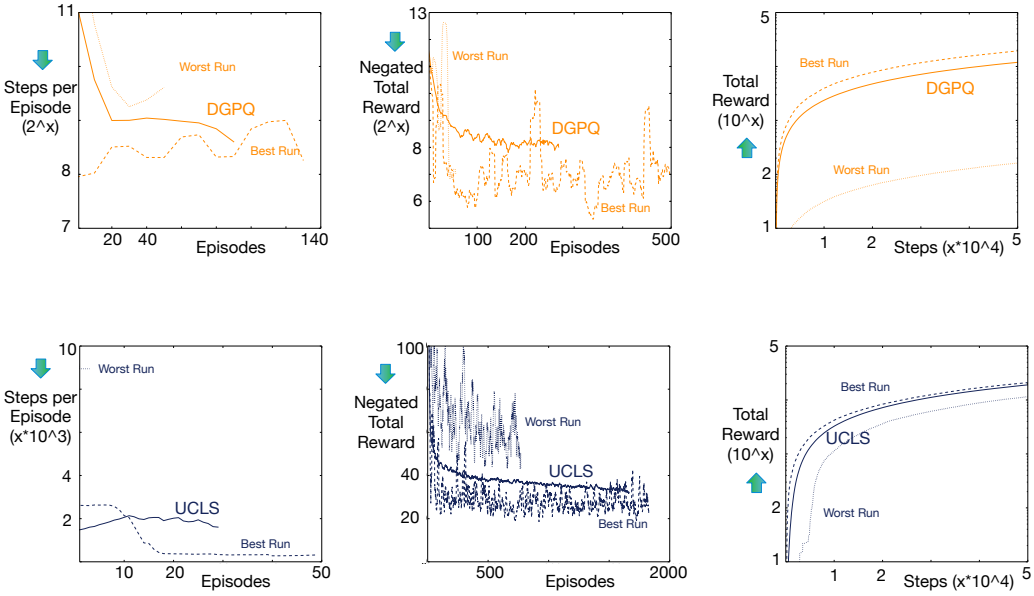

Figure 8: Best and worst run curves for DGPQ(top) and UCLS(bottom). From left to right: Sparse Mountain Car, Puddle World, River Swim.

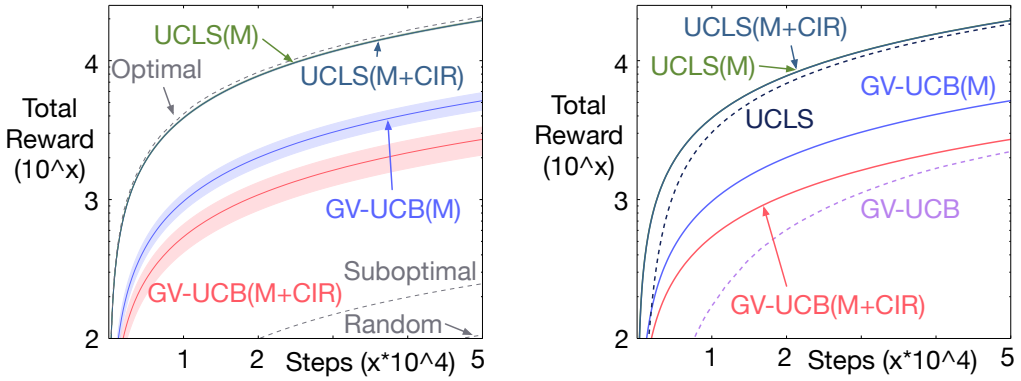

Figure 9: Policy evaluation plots comparing variations of final policy obtained by UCLS ($p = 0.1$) and GV-UCB ($p = 10e{-}5$) after 50,000 learning steps in River Swim. Policies with (M) indicate greedy policy w.r.t. mean estimates, whereas policies with (M+CIR) indicate greedy policies w.r.t. (mean + CIR) estimates. In the left plot it can be seen that UCLS(M) and UCLS(M+CIR) perform almost as well as the optimal policy, whereas both versions of GV-UCB are still sub-optimal in many parts of the state space. Additionally, the overlap of UCLS(M) and UCLS(M+CIR) indicates that contextual CIR fades faster than global CIR, and is a more data-efficient exploration strategy. The right plot helps contrast the final policies obtained to the actual control policy used during learning (indicated by just UCLS and GV-UCB).