NEAR-OPTIMAL POLICY IDENTIFICATION IN ACTIVE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Many real-world reinforcement learning tasks require control of complex dynamical systems that involve both costly data acquisition processes and large state spaces. In cases where the transition dynamics can be readily evaluated at specified states (e.g., via a simulator), agents can operate in what is often referred to as planning with a *generative model*. We propose the AE-LSVI algorithm for best-policy identification, a novel variant of the kernelized least-squares value iteration (LSVI) algorithm that combines optimism with pessimism for active exploration (AE). AE-LSVI provably identifies a near-optimal policy *uniformly* over an entire state space and achieves polynomial sample complexity guarantees that are independent of the number of states. When specialized to the recently introduced offline contextual Bayesian optimization setting, our algorithm achieves improved sample complexity bounds. Experimentally, we demonstrate that AE-LSVI outperforms other RL algorithms in a variety of environments when robustness to the initial state is required.

1 INTRODUCTION

Reinforcement learning (RL) algorithms are increasingly applied to complex domains such as robotics (Kober et al., 2013), magnetic tokamaks (Seo et al., 2021; Degrave et al., 2022), and molecular search (Simm et al., 2020a;b). A central challenge in such environments is that data acquisition is often a time-consuming and expensive process, or may be infeasible due to safety considerations. A common approach is therefore to train policies offline by interacting with a simulator.

However, even when a simulator is available, such applications require algorithms that are capable of learning and planning in *large state spaces*. Many existing approaches require a large amount of training data to obtain good policies, and efficient active exploration in large state spaces is still an open problem. Moreover, when deploying policies trained on simulators in real-world applications, a crucial requirement is that the policy performs well in *any* state that it might encounter. In particular, at training time, the learning approach has to sufficiently explore the state space. This is of particular importance when at test time, the system's state is partly out of the control of the learning algorithm—e.g., for a self-driving car or robot, which may be influenced by human actions.

In this work, we formally study the setting of *reinforcement learning with a generative model*. Our objective is to learn a near-optimal policy by actively querying the simulator with a state-action pair chosen by the learning algorithm. The simulator then returns a new state that is sampled from the transition model of the (simulated) environment. Inspired by previous works, we make a structural assumption in the kernel setting, which states that the Bellman operator maps any bounded value function to one with a bounded reproducing kernel Hilbert space (RKHS) norm. In particular, this assumption implies that the reward and the optimal Q-function can be represented by an RKHS function. We propose a novel approach based on least-squares value iteration (LSVI). The algorithm is designed to actively explore uncertain states based on the uncertainty in the Q-estimates, and makes use of optimism for action selection and pessimism for estimating a near-optimal policy.

Contributions We propose a novel kernelized algorithm for best policy identification in reinforcement learning with a generative model. Our sampling strategy actively explores (i) states for which the best action is the most uncertain and (ii) the corresponding "optimistic" actions. We prove sample complexity guarantees for finding an ϵ-optimal policy *uniformly over any given initial state*. Our bounds scale with the maximum information gain of the corresponding reproducing kernel Hilbert

space but *do not* explicitly scale with the number of states or actions. When specialized to the offline contextual Bayesian optimization (BO) setting (Char et al., 2019), we improve upon sample complexity guarantees from prior work. Finally, we include experimental evaluations on several RL and BO benchmark tasks. The former of these includes one of the first empirical evaluations of the model-free optimistic value iteration algorithms with function approximation (Yang et al., 2020).

1.1 Related Work

Reinforcement learning with function approximation dates back to at least (Bellman et al., 1963; Daniel, 1976; Schweitzer & Seidmann, 1985). A majority of work is in the *online* setting where the learning agent interacts with the environment while (typically) minimizing regret. Upper confidence bound algorithms, originally developed in the bandit setting (Lattimore & Szepesvári, 2020), have been successfully applied to tabular Markov decision processes (MDPs) (Auer & Ortner, 2006; Auer et al., 2008), and extended to RL with function approximation. Jin et al. (2020) propose the LSVI-UCB algorithm in the linear MDP setting that achieves a near-optimal regret bound. Yang et al. (2020); Domingues et al. (2021) extend this work to the non-linear function approximation setting. These work are closely related to ours in that we make use of LSVI and confidence bounds for the Q-function in the kernelized setting. Unlike previous works, we consider the generative model setting and derive bounds on the sample complexity that hold uniformly over the initial state. There are many more alternative parametric models that admit sample efficient algorithms (e.g., Ayoub et al., 2020; Zhou et al., 2021; Du et al., 2021; Zanette et al., 2020; Liu & Su, 2022). While here we are primarily interested in sample complexity guarantees, bounds on the cumulative regret can be translated to a PAC-sample complexity bound using *online-to-batch* conversion (Cesa-Bianchi et al., 2004). The online-to-batch policy however is arguably difficult to deploy and sample complexity guarantees can only be obtained for the initial state distribution used during training.

In the *generative model* setting, the learner has access to a simulator that for any given state-action pair returns a next-state sample from the transition kernel. This provides additional flexibility to obtain data from states that are otherwise hard to reach in the environment. For the tabular case, matching upper and lower bounds are shown by Azar et al. (2012); Gheshlaghi Azar et al. (2013). In the generative model setting with function approximation, Lattimore et al. (2020) show that policy iteration can be used to compute a near-optimal policy given features such that the Q-function of any policy can be approximated by a linear function. Their algorithm uses a D-experimental design to roll out policies from a sufficiently diverse set of states. The POLITEX algorithm (Abbasi-Yadkori et al., 2019; Szepesvári, 2022) can be used in lieu of policy iteration and leads to tighter bounds on the approximation error. A similar approach based on LSVI is analyzed by Agarwal et al. (2020, Chapter 3). The idea of using a *core set* of states to obtain sufficient data coverage was also used by Wang et al. (2021) for the case of linear transition models, and by Shariff & Szepesvári (2020) for the case where the V^* function can be expressed as a linear function.

In practical applications of RL, simpler approaches to exploration are often used or exploration techniques inspired by upper-confidence bound algorithms or Thompson sampling are combined with deep learning function approximation. To list a few, the ϵ-greedy approach (Mnih et al., 2013), upper confidence bounds (UCB) (Chen et al., 2017), Thompson sampling (TS) (Osband et al., 2016), added Ornstein-Uhlenbeck action noise (Lillicrap et al., 2015), and entropy bonuses (Haarnoja et al., 2018) are all widely applied. More sophisticated methods actively plan to encounter *novel* states (Shyam et al., 2019; Ecoffet et al., 2021). Though these methods serve as reasonable heuristics and are usually computationally efficient, they either lack theoretical guarantees or lead to methods that require large numbers of samples. One recent practical work (Mehta et al., 2022b) gives an *acquisition function* for the generative model setting based on methods from Bayesian experimental design (Neiswanger et al., 2021), and achieves good policies with small numbers of samples; however, this model-based method assumes access to the MDP reward function and is computationally expensive.

An important special case of the MDP setting is the *contextual bandit setting*. When combined with linear function approximation, this recovers the contextual linear bandit setting (Abbasi-Yadkori et al., 2011), and contextual Bayesian optimization when using kernel features (Srinivas et al., 2010; Krause & Ong, 2011). Various works consider the case where the learner has control over the choice of context during training time. Char et al. (2019) propose a variant based on Thompson sampling. Pearce & Branke (2018); Pearce et al. (2020) also propose variants that leverage ideas from the knowledge gradient (Frazier et al., 2009). The latter works lack theoretical guarantees, while our result (from Section 5) improves upon the sample complexity guarantee of Char et al. (2019). The

approach by Kirschner et al. (2020) for the distributionally robust setting can be specialized to our setting, in which case they recover similar bounds but only for a fixed context distribution.

2 Problem Statement

We consider an episodic MDP $\big(\mathcal{S}, \mathcal{A}, H, (\mathbb{P}_h)_{h\in[H]}, (r_h)_{h\in[H]}\big)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, horizon $H \in \mathbb{N}$, Markov transition kernel $(\mathbb{P}_h)_{h\in[H]}$ and deterministic reward functions $(r_h : \mathcal{S}\times\mathcal{A} \to [0,1])_{h\in[H]}$. In particular, for each $h \in [H]$, we let $\mathbb{P}_h(\cdot|s,a)$ denote the probability transition kernel when action a is taken at state $s \in \mathcal{S}$ in step $h \in [H]$. A *policy* consists of H functions $\pi = (\pi_h)_{h\in[H]}$ where for all $h \in [H]$, $\pi_h(\cdot|s)$ is a probability distribution over the action set $\mathcal{A}$. In particular, $\pi_h(a|s)$ is the probability that the agent takes action a in state s at step h.

We assume the *generative* (or random) access model, in which the agent interacts with the environment in the following way: Let T denote the number of episodes and H the horizon, i.e., the number of steps in each episode. Then for each $t \in [T], h \in [H]$, the agent chooses $s_h^t \in \mathcal{S}$, $a_h^t \in \mathcal{A}$, and obtains the reward $r_h(s_h^t, a_h^t)$ and observes the new state $s'_{h,t} \sim \mathbb{P}_h(\cdot|s_h^t, a_h^t)$.

To measure the performance of an agent, we use the *value* function. For a policy π, $h \in [H]$, $s \in S$, and $a \in \mathcal{A}$, the value function $V_h^\pi : \mathcal{S} \to \mathbb{R}$ and the Q-function $Q_h^\pi : \mathcal{S}\times\mathcal{A} \to [0,H]$ are given by:

$$V_h^\pi(s) = \mathbb{E}_\pi\Big[\sum_{h'=h}^{H} r_{h'}(s_{h'}, a_{h'})\Big| s_h = s\Big], \tag{1}$$

$$Q_h^\pi(s,a) = \mathbb{E}_\pi\Big[\sum_{h'=h}^{H} r_{h'}(s_{h'}, a_{h'})\Big| s_h = s, a_h = a\Big], \tag{2}$$

where $\mathbb{E}_\pi$ denotes the expectation with respect to the randomness of the trajectory $\{(s_h, a_h)\}_{h=1}^H$ that is obtained by following the policy π. We use π^* to denote the optimal policy, and we abbreviate $V_h^{\pi*}, Q_h^{\pi*}$ as V_h^*, Q_h^*, respectively. We also have $V_h^*(s) = \sup_\pi V_h^\pi(s)$ for all $s \in \mathcal{S}$ and $h \in [H]$.

The goal is to find an ϵ-optimal policy while minimizing the number of necessary episodes T. More precisely, for a fixed precision $\epsilon > 0$ and horizon $H \in \mathbb{N}$, the goal of the learner is to output a policy $\hat{\pi}_T$ after a suitable number of episodes $T > 0$ such that $\|V_1^* - V_1^{\hat{\pi}_T}\|_{\ell^\infty(\mathcal{S})} \le \epsilon$.

Finally, we also recall the Bellman equation that is associated to some policy π:

$$V_{H+1}^\pi = 0, \quad Q_h^\pi(s,a) = r_h(s,a) + \mathbb{E}_{s'\sim\mathbb{P}_h(\cdot|s,a)}[V_{h+1}^\pi(s')], \quad V_h^\pi(s) = \mathbb{E}_{a\sim\pi_h(a|s)}[Q_h^\pi(s,a)], \tag{3}$$

and the Bellman optimality equation:

$$V_{H+1}^* = 0, \quad Q_h^*(s,a) = r_h(s,a) + \mathbb{E}_{s'\sim\mathbb{P}_h(\cdot|s,a)}[V_{h+1}^*(s')], \quad V_h^*(s) = \max_{a\in\mathcal{A}} Q_h^*(s,a). \tag{4}$$

It follows that the optimal policy π^* is the greedy policy with respect to $\{Q_h^*\}_{h\in[H]}$, a property that is going to be useful later on when defining our active exploration strategy. We use the reproducing kernel Hilbert space (RKHS) function class to represent functions such as the reward functions $\{r_h\}_{h\in[H]}$ and the optimal Q-functions $\{Q_h^*\}_{h\in[H]}$ (see the formal statement in Assumption 1). In particular, we consider a space of well-behaved functions defined on $\mathcal{X} = \mathcal{S}\times\mathcal{A}$, where $\mathcal{H}$ denotes an RKHS defined on $\mathcal{X}$ induced by some continuous, positive definite kernel function $k : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$. We also assume that (i) $\mathcal{X} \subset \mathbb{R}^d$ is a compact set, (ii) the kernel function is bounded $k(x,x') \le 1$ for all $x, x' \in \mathcal{X}$, and (iii) every $f \in \mathcal{H}$ has a bounded RKHS norm, i.e., $\|f\|_{\mathcal{H}} \le B_Q H$ for some fixed positive constant $B_Q > 0$.

3 AE-LSVI Algorithm

Our algorithm runs in episodes $t \in [T]$ of horizon H. As in the kernel least-squares value iteration (Yang et al., 2020), at the beginning of every episode t, it solves a sequence of kernel ridge regression problems based on the data obtained in the previous $t-1$ episodes to obtain value function estimates $\{\hat{Q}_h^t\}_{h=1}^H$:

$$\hat{Q}_h^t \in \operatorname*{arg\,min}_{f\in\mathcal{H}} \Big\{ \sum_{i=1}^{t-1} \big(r_h(s_h^i, a_h^i) + V_{h+1}^t(s'_{h,i}) - f(s_h^i, a_h^i)\big)^2 + \lambda\|f\|_{\mathcal{H}}^2 \Big\}, \tag{5}$$

where λ is the regularization parameter. Recalling that $x \in \mathcal{X} = \mathcal{S} \times \mathcal{A}$, the solution of the problem in Eq. (5) can be written in closed form as follows:

$$\hat{Q}_h^t(x) = k_h^t(x)^T (K_h^t + \lambda I)^{-1} Y_t^h, \tag{6}$$

where $k_h^t(x) \in \mathbb{R}^{t-1}$, the kernel matrix $K_h^t \in \mathbb{R}^{(t-1)\times(t-1)}$ and observations $Y_h^t \in \mathbb{R}^{t-1}$ are given as follows:

$$k_h^t(x) = [k(x_h^1, x), \ldots, k(x_h^{t-1}, x)],\ K_h^t = \left[k(x_h^i, x_h^{i'})\right]_{i,i' \in [t-1]},\ [Y_h^t]_i = r_h(s_h^i, a_h^i) + V_{h+1}^t(s'_{h,i}).$$

Next, we can also compute the uncertainty function $\sigma_h^t(\cdot,\cdot)$ in the closed form:

$$\sigma_h^t(s,a) = \tfrac{1}{\lambda^{1/2}} \left(k(x,x) - k_h^t(x)^T (K_h^t + \lambda I)^{-1} k_h^t(x)\right)^{1/2}. \tag{7}$$

We recall that each reward function is bounded in $[0,1]$. We use $[\,\cdot\,]_0^{H-h+1}$ to denote the truncation to the interval $[0, H-h+1]$ and we define the optimistic $\overline{Q}_h^t$ and pessimistic $\underline{Q}_h^t$ value estimates (i.e., upper and lower confidence bound of Q_h^*; see Lemma 4.1 and Lemma A.1):

$$\overline{Q}_h^t(\cdot,\cdot) = \left[\hat{Q}_h^t(\cdot,\cdot) + \beta\sigma_h^t(\cdot,\cdot)\right]_0^{H-h+1}, \quad \overline{V}_h^t(\cdot) = \max_{a\in\mathcal{A}} \overline{Q}_h^t(\cdot, a), \tag{8}$$

$$\hat{Q}_h^t(\cdot) = k_h^t(\cdot)^T (K_h^t + \lambda I)^{-1} \overline{Y}_h^t, \quad [\overline{Y}_h^t]_i = r_h(s_h^i, a_h^i) + \overline{V}_{h+1}^t(s'_{h,i}). \tag{9}$$

Similarly, we have

$$\underline{Q}_h^t(\cdot,\cdot) = \left[\check{Q}_h^t(\cdot,\cdot) - \beta\sigma_h^t(\cdot,\cdot)\right]_0^{H-h+1}, \quad \underline{V}_h^t(\cdot) = \max_{a\in\mathcal{A}} \underline{Q}_h^t(\cdot, a), \tag{10}$$

$$\check{Q}_h^t(\cdot) = k_h^t(\cdot)^T (K_h^t + \lambda I)^{-1} \underline{Y}_h^t, \quad [\underline{Y}_h^t]_i = r_h(s_h^i, a_h^i) + \underline{V}_{h+1}^t(s'_{h,i}). \tag{11}$$

Our proposed algorithm AE-LSVI is presented in Algorithm 1. At each h, the algorithm uses optimistic and pessimistic value estimates from Eqs. (8) and (10) (computed based on the data collected in previous episodes), and selects s_h^t and a_h^t as:

$$s_h^t \in \arg\max_{s\in S} \left[\max_{a\in A} \overline{Q}_h^t(s,a) - \max_{a\in A} \underline{Q}_h^t(s,a)\right], \tag{12}$$

$$a_h^t \in \arg\max_{a\in\mathcal{A}}\ \overline{Q}_h^t(s_h^t, a). \tag{13}$$

The main intuition behind the proposed sampling rules is as follows. Since the optimal policy π^* is the greedy policy with respect to $\{Q_h^*\}_{h\in[H]}$, we do not need to learn $\{Q_h^*\}_{h\in[H]}$ everywhere on $\mathcal{S} \times \mathcal{A}$. Hence, it is sufficient to focus on discovering best actions for each state. Our active exploration strategy is explicitly designed to focus on (i) states for which the best action is the most uncertain (Eq. (12)) and (ii) corresponding best "optimistic" actions (Eq. (13)).

We use $\hat{\pi}_T$ to denote the final reported policy returned by AE-LSVI (see Algorithm 1). There are various reasonable greedy-based choices for $\hat{\pi}_T$. The simplest one is to return $\hat{\pi}_{T,h}(\cdot) = \arg\max_{a\in\mathcal{A}} \hat{Q}_h^T(\cdot, a)$, but in our theory and experiments, we focus on equating $\hat{\pi}_T$ with the policy with the highest lower confidence estimate $\underline{Q}_h^t(s,a)$. Our sampling strategy combined with the proposed policy reporting rule allows for discovering an ϵ-optimal policy uniformly over any given initial state as we formally show in the next section.

4 Theoretical results

In this work, we make use of the structural assumption for the kernel setting from Yang et al. (2020) which states that the Bellman operator maps any bounded value function to a function with a bounded RKHS norm.

Assumption 1. *Let $B_Q > 0$ be a fixed positive constant. Let $k : (\mathcal{S} \times \mathcal{A})^2 \to \mathbb{R}$ be a continuous kernel function on a compact set $\mathcal{S} \times \mathcal{A} \subset \mathbb{R}^d$ such that $\sup_{x,x'\in\mathcal{S}\times\mathcal{A}} k(x,x') \le 1$. We assume that $\|T_h^* Q\|_{\mathcal{H}} \le B_Q H$ for all functions $Q : \mathcal{S} \times \mathcal{A} \to [0,H]$ and all $h \in [H]$, where T_h^* denotes the Bellman optimality operator, i.e.,*

$$T_h^* Q(s,a) = r(s,a) + \mathbb{E}_{s'\sim\mathbb{P}_h(\cdot|s,a)}\left[\max_{a'\in\mathcal{A}} Q(s',a')\right]. \tag{14}$$

Algorithm 1 AE-LSVI (Active Exploration with Least-Squares Value Iteration)

Require: kernel function $k(\cdot,\cdot)$, exploration parameter $\beta > 0$, regularizer $\lambda \geq 1$

for $t = 1, \ldots, T$ **do**
  for $h \in \{1, \ldots, H\}$ **do**
    Set $\overline{Q}^t_{H+1}, \underline{Q}^t_{H+1}$ as the zero functions
    for $h = H, \ldots, 1$ **do**
      Obtain $\overline{Q}^t_h$ and $\underline{Q}^t_h$ from Eq. (8) and Eq. (10)
    end for
    Choose $s^t_h \in \arg\max_{s\in S} \left[\max_{a\in A} \overline{Q}^t_h(s,a) - \max_{a\in A} \underline{Q}^t_h(s,a)\right]$
    Choose $a^t_h \in \arg\max_{a\in\mathcal{A}} \overline{Q}^t_h(s^t_h, a)$
    Observe the reward $r_h(s^t_h, a^t_h)$ and the next state $s'_{h,t} \sim \mathbb{P}_h\big(\cdot\,|s^t_h, a^t_h\big)$
  end for
end for
Output the policy estimate $\hat{\pi}_T$ such that $\hat{\pi}_{T,h}(\cdot) = \arg\max_{a\in\mathcal{A}} \max_{t\in[T]} \underline{Q}^t_h(s,a)$

Assumption 1 implies that for every $h \in [H]$, both $r_h(\cdot,\cdot)$ and $Q^*_h(\cdot,\cdot)$ are elements of the set $\{f \in \mathcal{H} : \|f\|_{\mathcal{H}} \leq B_Q H\}$.

The main quantity that characterizes the complexity of the RKHS function class in the kernelized setting is the maximum information gain (Srinivas et al., 2010)

$$\Gamma_k(T,\lambda) := \sup_{D\subseteq\mathcal{S}\times\mathcal{A}, |D|\leq T} \tfrac{1}{2}\ln|I + \lambda^{-1}K_{D,D}|, \tag{15}$$

where $K_{D,D}$ denotes the Gram matrix, $|\cdot|$ denotes the determinant, $\lambda > 0$ is a regularization parameter, and the index k indicates the kernel. This quantity is known to be sublinear in T for most of the popularly used kernels (Srinivas et al., 2010).

Further, we define the set of possible optimistic and pessimistic value functions

$$\mathcal{Q}(T,h,b) = \Big\{Q(\cdot,\cdot) = \big[\hat{Q}(\cdot,\cdot) \pm \beta\sigma_D(\cdot,\cdot)\big]_0^{H-h+1} : \\ \hat{Q} \in \mathcal{H}, \|\hat{Q}\|_{\mathcal{H}} \leq 2H\sqrt{\Gamma_k(T,\lambda)}, \beta \in [0,b], D \subseteq \mathcal{S}\times\mathcal{A}, |D| \leq T\Big\}, \tag{16}$$

where $b > 0$ and $\sigma_D(\cdot,\cdot)$ is of the form Eq. (7) computed with a data set $D \subseteq \mathcal{S}\times\mathcal{A}$, and denote its ℓ^∞-covering number as $N_\infty(\epsilon,T,h,b)$. If $h \in [H]$, $\epsilon \in (0, 1/e)$, and λ is bounded by $[c_1, c_2]$ for absolute constants $c_1, c_2 > 0$, then

$$\log\big(N_\infty(\epsilon,T,h,b)\big) = \mathcal{O}\big(\gamma^2\cdot[1+\log(b/\epsilon)] + \gamma\cdot[1+\log(2H\sqrt{\Gamma_k(T,\lambda)}) - \log(\epsilon)]\big) \tag{17}$$

holds for kernels with γ-finite spectrum (includes the finite-dimensional case), and similarly,

$$\log\big(N_\infty(\epsilon,T,h,b)\big) = \mathcal{O}\big([1+\log(b/\epsilon)]^{1+2/\gamma} + \gamma\cdot[1+\log(2H\sqrt{\Gamma_k(T,\lambda)}) - \log(\epsilon)]^{1+1/\gamma}\big) \tag{18}$$

for kernels with γ-exponential decay (includes the squared exponential kernel).[1]

Our sample complexity bounds depend on $b_T > 0$ defined as the smallest number that satisfies the following inequality:

$$8\Gamma_k\big(T, \tfrac{T+1}{T}\big) + 8\log N_\infty(H/T, T, h, b_T) + 16\log(2TH) + 22 + 2B_Q^2\big(\tfrac{T+1}{T}\big) \leq (b_T/H)^2 \tag{19}$$

For many kernel functions, b_T has a sublinear dependence on T. For instance, $b_T = \mathcal{O}(\gamma H\sqrt{\log(\gamma TH)})$ for bounded and continuously differentiable kernels with γ-finite spectrum and $b_T = \mathcal{O}(H\sqrt{TH}\log(T)^{1/\gamma})$ for bounded and continuously differentiable kernels with γ-exponential decay. See (Yang et al., 2020, Corollary 4.4) for more details.

We recall that the Bellman equation implies that $\overline{Q}^t_{h+1}(\cdot)$, $\underline{Q}^t_h(\cdot)$ are upper and lower confidence bounds for Q^*_h for all $h \in [H]$, respectively (see Lemma A.1), while the target functions of kernel ridge regressions are $T^*_h\overline{Q}^t_{h+1}(\cdot)$ and $T^*_h\underline{Q}^t_{h+1}(\cdot)$. As a technical tool, we use the following concentration result that follows from (Yang et al., 2020, Lemma 5.2).

[1] Eqs. (17) and (18) hold despite Yang et al. (2020, Lemma D.1) being stated only in the case of the smaller class obtained from only adding $+\beta\sigma_D(\cdot,\cdot)$ in the definition of $Q(\cdot,\cdot)$ in $\mathcal{Q}(T,h,b)$ in Eq. (16).

Lemma 4.1. *Consider the setup of Assumption 1, and* $\overline{Q}^t_{h+1}(\cdot)$, $\underline{Q}^t_h(\cdot)$ $\sigma^t_h(\cdot)$ *from Eqs.* (7), (8) *and* (10) *computed with* $\lambda = 1 + 1/T$ *and* $\beta = b_T$ *from Eq.* (19)*. Then with probability at least* $1 - (2T^2H^2)^{-1}$*, the following holds for all* $t \in [T]$, $h \in [H]$ *and all* $(s, a) \in \mathcal{S} \times \mathcal{A}$*:*

$$0 \leq \overline{Q}^t_h(s,a) - T^*_h \overline{Q}^t_{h+1}(s,a) \leq 2\beta\sigma^t_h(s,a), \tag{20}$$

$$0 \leq T^*_h \underline{Q}^t_{h+1}(s,a) - \underline{Q}^t_h(s,a) \leq 2\beta\sigma^t_h(s,a). \tag{21}$$

With the previous confidence lemma in place, we state our main theorem that characterizes the sample complexity of AE-LSVI . The proof is given in Appendix A.2.

Theorem 4.2. *Consider the setting of Lemma 4.1 and let* $H \in \mathbb{N}$ *be a fixed horizon. When running Algorithm 1 for* T *episodes, then with probability at least* $1 - (2T^2H^2)^{-1}$*, the best-policy estimate* $\hat{\pi}_T$ *(Algorithm 1, Line 12) satisfies:*

$$\|V^*_1 - V^{\hat{\pi}_T}_1\|_{\ell^\infty(\mathcal{S})} \leq 2\sqrt{3}\beta H(H+1)\sqrt{\tfrac{\Gamma_k(T,\lambda)}{T}}. \tag{22}$$

In other words, for a given fixed precision $\epsilon > 0$*, after* $T = O\Big(\frac{\beta^2 H^4 \Gamma_k(T,\lambda)}{\epsilon^2}\Big)$ *episodes (or* $O\Big(\frac{\beta^2 H^5 \Gamma_k(T,\lambda)}{\epsilon^2}\Big)$ *samples)* $\|V^*_1 - V^{\hat{\pi}_T}_1\|_{\ell^\infty(\mathcal{S})} \leq \epsilon$ *holds with probability at least* $1 - (2T^2H^2)^{-1}$.

The obtained result is general since it holds for any kernel function that satisfies Assumption 1. To obtain concrete kernel-dependent regret bounds it remains to specify the kernel and the bounds for the corresponding maximum information gain in Eq. (19). These are summarized in Yang et al. (2020) for the most widely used kernels (see Assumption 4.3 and its discussion).

In the special case of linear kernels with the feature dimension d, our sample complexity guarantee reduces to $\tilde{O}(\frac{d^3H^7}{\epsilon^2})$. Better bounds (in terms of d) for this special case are known $\tilde{O}(\frac{d^2H^7}{\epsilon^2})$, see, e.g., Agarwal et al. (2019, Theorem 3.3). These bounds are obtained by the LSVI algorithm with D-optimal design. Unlike this algorithm, AE-LSVI uses optimism for active exploration and such a performance gap is present even in the simpler linear bandit setting where optimistic algorithms are known to attain worse sample complexity guarantees (Lattimore & Szepesvári, 2020, Chapter 22).

Finally, as a comparison, we note that the result of Yang et al. (2020) implies that their resulting policy π satisfies $V_1(s_1) - V^\pi_1(s_1) \leq \epsilon$ using $O(H^5\beta^2\Gamma_k(T,\lambda)/\epsilon^2)$ samples. However, this result holds with at least a constant probability and requires that the initial state s_1 is fixed for all episodes. In contrast, the result of Theorem 4.2 holds uniformly over the entire state space.

5 Application to offline contextual Bayesian optimization

In this section, we specialize Algorithm 1 to the offline contextual Bayesian optimization setting (Char et al., 2019). We show that in this setting the proposed active exploration scheme leads to new sample complexity bounds that hold *uniformly* over the context space.

The offline contextual Bayesian optimization setting is similar to the one considered in Section 2 when $H = 1$. In particular, instead of having H different functions to learn, we have a single unknown objective $Q^* : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that we learn about (from noisy point evaluations). Here, we refer to $\mathcal{S}$ as the context space, and assume that both $\mathcal{S}$ and $\mathcal{A}$ are compact sets. As before, we use a shorthand notation $\mathcal{X} = \mathcal{S} \times \mathcal{A}$. In each round $t \in [T]$, the learner chooses a context-action pair $(s^t, a^t) \in \mathcal{S} \times \mathcal{A}$ and observes $y_t = Q^*(s^t, a^t) + \eta_t$ (with independent sub-Gaussian noise). To choose (s^t, a^t) at each round t, we make use of the same active exploration strategy from Eqs. (12) and (13). Our complete algorithm for the offline BO setting can be found in Appendix A.3 (see Algorithm 2).

We define $\hat{Q}^t : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (and $\sigma^t : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$) similarly as $\hat{Q}^t_h$ (resp. σ^t_h) from Eq. (6) (resp. Eq. (7)) but with the modification of ignoring the index h and defining $Y_t := (y_i)_{i=1}^{t-1} \in \mathbb{R}^{t-1}$. We further define the upper and lower confidence bounds for Q^* as:

$$\overline{Q}^t(\cdot,\cdot) = \hat{Q}^t(\cdot,\cdot) + \beta_t\sigma^t(\cdot,\cdot), \quad \underline{Q}^t(\cdot,\cdot) = \hat{Q}^t(\cdot,\cdot) - \beta_t\sigma^t(\cdot,\cdot). \tag{23}$$

When $Q^* \in \mathcal{H}$ and $\|Q^*\|_{\mathcal{H}} \leq B$ correspond to some known kernel (such that $k(x, x') \leq 1$ for all $x, x' \in \mathcal{X}$), then $(\beta_t)_{t\in[T]}$ is a non-decreasing sequence of parameters that can be chosen according to Abbasi-Yadkori (2012, Theorem 3.11) to yield valid confidence bounds. Similarly, in

case of $Q^* \sim \mathrm{GP}_{\mathcal{X}}(0, k)$ (Bayesian setting), we can utilize Gaussian Process confidence bounds (Srinivas et al., 2010) and use the corresponding $(\beta_t)_{t\in[T]}$ sequence. In what follows, we assume that $(\beta_t(\delta))_{t\in[T]}$ is a non-decreasing sequence such that with probability at least $1-\delta$,

$$\underline{Q}^t(s,a) \leq Q^*(s,a) \leq \overline{Q}^t(s,a) \tag{24}$$

holds for all $t \in [T]$ and $(s,a) \in \mathcal{S} \times \mathcal{A}$.

Corollary 5.1. *Assume $(\beta_t(\delta))_{t\in[T]}$ is set to satisfy Eq.* (24)*. Fix $\epsilon \in (0,1)$ and run Algorithm 2 for*

$$T \geq \frac{12\beta_T^2 \Gamma_k(T,\lambda)}{\epsilon^2} \tag{25}$$

rounds. Then, for every $s \in \mathcal{S}$, the reported policy $\hat{\pi}_T(\cdot)$ computed as in Line 6 (Algorithm 2) satisfies $Q^*(s, \hat{\pi}_T(s)) \geq \max_{a\in\mathcal{A}} Q^*(s,a) - \epsilon$ with probability at least $1-\delta$.

We briefly compare the result obtained in Corollary 5.1 with related results from the literature. Char et al. (2019, Theorem 1) obtain a sample complexity that scales as $\mathbb{E}[T] = O\big(|\mathcal{S}|^3|\mathcal{A}|\Gamma_k(T,\lambda)/\epsilon^2\big)$ in expectation for a given context distribution. In comparison, our result obtained in Eq. (25) is independent of the context space size $|\mathcal{S}|$ in general and holds in ∞-norm over the context space (i.e., implies bounds for *any* context distribution). In the setting of distributionally robust Bayesian optimization (DRBO), Kirschner et al. (2020) obtain a result with the same dependency as ours. However, their bound holds only for a *fixed* contextual distribution, and degenerates as a function of the distance between the training and test distributions.

6 Experiments

6.1 Reinforcement Learning Experiments

In the previous sections we presented the AE-LSVI algorithm, which provably identifies a near-optimal policy in polynomial time given access to a generative model of the MDP dynamics. Here, we test the AE-LSVI algorithm empirically, and additionally provide one of the first empirical evaluation of the LSVI-UCB method from Yang et al. (2020) on standard benchmarks. We evaluate AE-LSVI and LSVI-UCB on four MDPs from the literature as well as four synthetic contextual BO problems from Char et al. (2019). We discuss details of our implementation in Appendix B.1

Each environment has a discrete action space. For continuous environments, we discretize the action space into 10 bins per dimension but model the value function in the original continuous state and action space. All methods besides DDQN are initialized by executing a random policy for two episodes. In between exploration episodes, the pessimistic policy $\hat{\pi}_T$ is evaluated by executing it for 10 episodes in the environment.

Initial State Distribution To evaluate the policies found by each method, we must initialize the policy at initial states drawn from some distribution p_0 at test time. As AE-LSVI does not explicitly consider the initial state distribution, for each environment we choose both a standard p_0 from the literature as well as a an alternate distribution p_0' that is translated in the state space, i.e., $p_0'(s) = p_0(s - \Delta_s)$ for some Δ_s. The alternate distribution allows us evaluate the best policy estimate in an area of state space that is not explicitly given to agents. We evaluate each policy using initial states sampled from p_0' as a proxy for understanding how well the optimal policy has been identified in regions of the state space beyond where it was initialized. We give a complete description of the various p_0' for each environment in Appendix B.2. In Table 1, we present results for each method and environment when initialized on p_0, which is the typical setup for training and evaluating RL algorithms in the literature. In Table 2 we present results for each method evaluated starting instead from p_0'.

Comparison Methods Besides AE-LSVI and LSVI-UCB , we compare against several ablations and methods taken from the literature. As a naive baseline for performance in active exploration, we randomly sample state-action pairs from the MDP, evaluate the next states and rewards, and fit Q-functions to that data as in the other methods, executing the policy given by the Q-function mean (**Random**). We also perform uncertainty sampling (**US**) on the Q-function, choosing state-action pairs at each step that maximize $\sigma_h^t(\cdot,\cdot)$ as in Eq. (7). Additionally, we compare against two online RL baselines: the Double DQN algorithm (**DDQN**) (Van Hasselt et al., 2016), a state-of-the-art model-free algorithm that approximates value functions with neural networks, and a greedy exploration algorithm (**Greedy**) that chooses $\arg\max_a \hat{Q}_h^t(s,a)$ at every step h for a given state s but uses the same value iteration procedure used in the main methods.

Environment	AE-LSVI	Random	US	LSVI-UCB	DDQN	Greedy
Cartpole	15.2 ± 0.5	13.6 ± 0.5	13.6 ± 0.6	17.1 ± 0.7	$\mathbf{19.3 \pm 0.7}$	17.2 ± 0.4
Navigation	6.0 ± 1.7	6.7 ± 1.4	8.9 ± 0.7	$\mathbf{12.9 \pm 0.2}$	7.3 ± 1.5	10.9 ± 1.5
β Tracking	12.7 ± 0.3	11.6 ± 0.4	11.7 ± 0.2	$\mathbf{13.8 \pm 0.1}$	13.4 ± 0.2	12.9 ± 0.3
β + Rotation	15.2 ± 0.6	15.2 ± 0.6	15.1 ± 0.4	$\mathbf{17.8 \pm 0.1}$	15.1 ± 0.4	$\mathbf{17.9 \pm 0.1}$

Table 1: Average Return $\pm$ standard error of executing the identified best policy on the MDP starting from p_0 over 5 seeds after collecting 1000 timesteps of data through the use of a generative model (left of line) or episodes starting from p_0 (right of line).

Environment	AE-LSVI	Random	US	LSVI-UCB	DDQN	Greedy
Cartpole	$\mathbf{16.8 \pm 0.4}$	12.9 ± 0.4	14.5 ± 0.3	12.9 ± 0.3 (14.2 ± 0.6)	15.3 ± 0.6 (13.7 ± 1.3)	13.3 ± 0.5 $(\mathbf{16.7 \pm 0.2})$
Navigation	$\mathbf{22.3 \pm 0.4}$	15.3 ± 0.8	17.5 ± 1.3	13.6 ± 0.6 (20.6 ± 1.1)	17.1 ± 2.4 (18.1 ± 2.6)	15.2 ± 1.6 (14.0 ± 0.8)
β Tracking	$\mathbf{14.0 \pm 0.4}$	9.2 ± 0.9	12.5 ± 0.1	13.3 ± 0.3 $(\mathbf{13.7 \pm 0.2})$	$\mathbf{13.8 \pm 0.1}$ $(\mathbf{13.7 \pm 0.2})$	12.5 ± 0.4 $(\mathbf{13.8 \pm 0.1})$
β + Rotation	$\mathbf{14.3 \pm 0.2}$	12.8 ± 1.4	13.3 ± 0.5	10.1 ± 0.4 (12.7 ± 0.3)	12.9 ± 1.1 (13.4 ± 0.3)	12.8 ± 0.7 (7.5 ± 0.2)

Table 2: Average Return $\pm$ standard error of executing the identified best policy on the MDP starting from p_0' over 5 seeds after collecting 1000 timesteps of data through the use of a generative model (left) and online RL methods (right). For online methods, numbers without parentheses refer to training from episodes starting from p_0, whereas numbers in parentheses use the uniform distribution on the state space as initial states during training.

Environments We evaluate all methods on four environments: a **Cartpole** swing-up problem with dense rewards, a nonlinear **Navigation** problem, and two problems (β **Tracking** and β **+ Rotation**) in plasma control from Mehta et al. (2022a), in which plasma is driven to a desired target state. We give further information on the environments used in Appendix B.2.

Results As our bound on the value function error uses the $\ell^\infty(\mathcal{S})$-norm, our method provably finds an approximately optimal policy regardless of the initial distribution. The LSVI-UCB method is able to quickly learn a policy for the initial state distribution p_0 given at training time, as it is designed to minimize regret on the episodic MDP initialized at p_0. This can be seen clearly in Table 1, which shows that after 1000 samples, LSVI-UCB performs the best on nearly every environment. In the online setting when the start state distribution is known, greedy and ϵ-greedy methods like DDQN also perform relatively well. We also see in Table 1 that AE-LSVI does not perform particularly well compared to the online methods given the 1,000-sample budget. This is to be expected, as the online methods naturally collect data that is reachable from p_0 and in particular LSVI-UCB is designed to minimize regret on episodes beginning from p_0. However, this focus on performing well when starting from p_0 comes at the expense of active exploration and identifying the best policy uniformly across the state space.

As shown in Table 2, AE-LSVI outperforms the baselines when evaluated on a *different* initial state distribution p_0', even when the online algorithms are initialized from a uniform initial state distribution p_0 during training. This is unsurprising, as AE-LSVI is precisely built for this setting and identifies the best action uniformly across the state space, unlike LSVI-UCB which aims to minimize regret starting from an initial state distribution. We see that uncertainty sampling outperforms a random data selection strategy and is comparable to the online methods. However, as we discuss above (in Section 3), in general it is the uncertainty in the value of the best action at a state and not the uncertainty in the value of a state-action pair that needs to be reduced in order to more efficiently find the best policy. We see that, in general, the online methods perform better on p_0' when they train on episodes uniformly initialized on the state space. This suggests that in these cases, it is helpful to make sure that the evaluation distribution p_0' is supported by the training distribution p_0. We also note that (as we describe in Appendix B.2) the maximum possible score on **Navigation** starting from p_0' is higher than that from p_0 due to a starting distribution closer to the goal. We believe that these results give empirical support to the theoretical claims of Section 4.

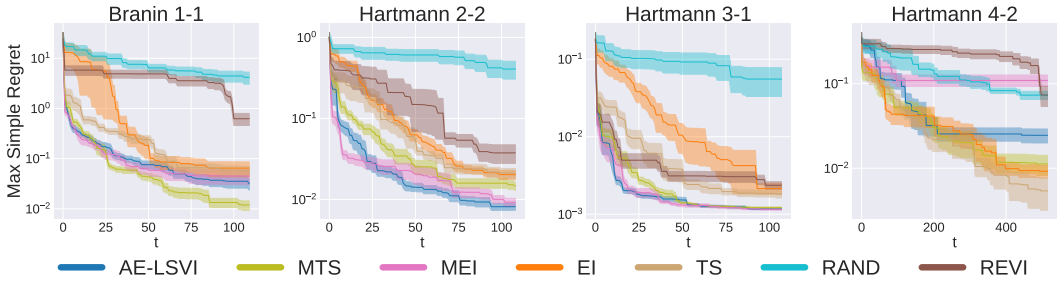

Figure 1: The maximum simple regret seen in any given context for the offline contextual Bayesian optimization experiments. The shaded regions show the standard error over 10 different seeds.

6.2 Offline Contextual Bayesian Optimization Experiments

We test the performance of AE-LSVI (Algorithm 2 in Appendix A.3) in the offline contextual Bayesian optimization setting. In particular, we test the algorithm on the optimization problems presented in Section 3 of Char et al. (2019), each having a discrete context space but continuous action space. In all experiments, we average over 10 seeds. At the beginning of each experiment, the values corresponding to five actions, chosen uniformly at random, are observed for each context. Every time new data is observed, the hyperparameters of the GP are tuned according to the marginal likelihood. We leverage the Dragonfly library for these experiments (Kandasamy et al., 2020).

Comparison Methods For baselines, we compare against the Multi-task Thompson Sampling (**MTS**) method presented by Char et al. (2019), which picks context and action based on the largest improvement over what has been seen according to samples from the posterior. In addition, we compare to the strategy of picking the context with the greatest expected improvement. This method was presented by Swersky et al. (2013), and we refer to it as Multi-task Expected Improvement (**MEI**), following Char et al. (2019). We also compare against the **REVI** algorithm (Pearce & Branke, 2018), which picks contexts and actions that will increase the posterior mean the most across all contexts. Additionally, we show the performance of naive Thompson sampling (**TS**) and expected improvement (**EI**), where contexts are picked in a round robin fashion. Lastly, we show the performance of randomly selecting contexts and actions at each time step (**RAND**).

Experiment Tasks To evaluate the method in the case where the objective function is correlated in context space, we take a higher dimensional function and assign some dimensions to context space and the rest to action space. A single GP with a squared exponential kernel is then used to model the objective function. In particular, the Branin-Hoo (Branin, 1972), Hartmann 4, and Hartmann 6 (Picheny et al., 2013) functions are used to create Branin 1-1, Hartmann 2-2, Hartmann 3-1, and Hartmann 4-2, where the first number corresponds to the context dimension and the second to the action dimension. These functions have 10, 9, 8, and 16 equispaced contexts, respectively.

Results Figure 1 shows the maximum simple regret seen in any given context as a function of t values observed. As seen from these plots, AE-LSVI often is one of the best performing methods. The only task that AE-LSVI struggles on is Hartmann 4-2. We believe that estimating the amount of improvement to be gained at each context is difficult for this benchmark task. This is supported by the fact none of the more sophisticated methods outperforms the baseline that applies **EI** in a round-robin fashion. It is likely that improved modeling or hyperparameter selection is needed in order for these methods to achieve the highest performance on this task.

7 Conclusion

We have provided a new kernelized least-squares value iteration algorithm for RL in the generative model setting, which aims to learn a near-optimal policy for all initial states by actively exploring states for which the best action is the most uncertain. Our algorithm identifies a near-optimal policy uniformly over the entire state space and attains polynomial sample complexity. Experimentally, we demonstrate that it outperforms other RL algorithms in a variety of environments when robustness to the initial state is required. Perhaps the most immediate direction for future work is to extend the algorithm to the local access model (Yin et al., 2022) in which the simulator can be queried only for states that have been encountered in previous simulation steps.

REPRODUCIBILITY STATEMENT

The proof of Theorem 4.2 is provided in Appendix A.2 and the proof of Corollary 5.1 is given in Appendix A.3. The supplementary material includes the source code for the experiments. It also includes a requirements file and README with full instructions on how to run the RL and BO experiments. Although we are not allowed to provide the data used for running the β Tracking and β + Rotation experiments at this time, all other experiments can be run using the provided code. Lastly, experimental details about the implementation and the environments used can be found in Appendix B.1 and Appendix B.2, respectively.

REFERENCES

Yasin Abbasi-Yadkori. Online learning for linearly parametrized control problems. 2012.

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24, 2011.

Yasin Abbasi-Yadkori, Peter Bartlett, Kush Bhatia, Nevena Lazic, Csaba Szepesvari, and Gellért Weisz. Politex: Regret bounds for policy iteration using expert prediction. In *International Conference on Machine Learning*, pp. 3692–3702. PMLR, 2019.

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, pp. 10–4, 2019.

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *URL https://rltheorybook. github. io*, 2020.

Peter Auer and Ronald Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. *Advances in neural information processing systems*, 19, 2006.

Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21, 2008.

Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin Yang. Model-based reinforcement learning with value-targeted regression. In *International Conference on Machine Learning*, pp. 463–474. PMLR, 2020.

Mohammad Gheshlaghi Azar, Rémi Munos, and Bert Kappen. On the sample complexity of reinforcement learning with a generative model. *arXiv preprint arXiv:1206.6461*, 2012.

Richard Bellman, Robert Kalaba, and Bella Kotkin. Polynomial approximation–a new computational technique in dynamic programming: Allocation processes. *Mathematics of Computation*, 17(82):155–161, 1963.

Ilija Bogunovic and Andreas Krause. Misspecified gaussian process bandit optimization. *Advances in Neural Information Processing Systems*, 34:3004–3015, 2021.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Franklin H Branin. Widely convergent method for finding multiple solutions of simultaneous nonlinear equations. *IBM Journal of Research and Development*, 16(5):504–522, 1972.

Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

Ian Char, Youngseog Chung, Willie Neiswanger, Kirthevasan Kandasamy, Andrew O Nelson, Mark Boyer, Egemen Kolemen, and Jeff Schneider. Offline contextual bayesian optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Ian Char, Joseph Abbate, Laszo Bardoczi, Mark D. Boyer, Youngseog Chung, Rory Conlin, Keith Erickson, Viraj Mehta, Nathan Richner, Egemen Kolemen, and Jeff Schneider. Offline model-based reinforcement learning for tokamak control. *arXiv preprint*, 2022.

Richard Y. Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. UCB and infogain exploration via q-ensembles. *CoRR*, abs/1706.01502, 2017. URL `http://arxiv.org/abs/1706.01502`.

James W Daniel. Splines and efficiency in dynamic programming. *Journal of Mathematical Analysis and Applications*, 54(2):402–407, 1976.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

Omar Darwiche Domingues, Pierre Ménard, Matteo Pirotta, Emilie Kaufmann, and Michal Valko. Kernel-based reinforcement learning: A finite-time analysis. In *International Conference on Machine Learning*, pp. 2783–2792. PMLR, 2021.

Simon Du, Sham Kakade, Jason Lee, Shachar Lovett, Gaurav Mahajan, Wen Sun, and Ruosong Wang. Bilinear classes: A structural framework for provable generalization in rl. In *International Conference on Machine Learning*, pp. 2826–2836. PMLR, 2021.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Dan Foreman-Mackey. Tinygp, 2021. URL `https://tinygp.readthedocs.io`.

Peter Frazier, Warren Powell, and Savas Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS journal on Computing*, 21(4):599–613, 2009.

Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3): 325–349, 2013.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pp. 2137–2143. PMLR, 2020.

Kirthevasan Kandasamy, Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher R Collins, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. *J. Mach. Learn. Res.*, 21(81): 1–27, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Johannes Kirschner, Ilija Bogunovic, Stefanie Jegelka, and Andreas Krause. Distributionally robust bayesian optimization. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2174–2184. PMLR, 26–28 Aug 2020. URL `https://proceedings.mlr.press/v108/kirschner20a.html`.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Andreas Krause and Cheng Ong. Contextual gaussian process bandit optimization. *Advances in neural information processing systems*, 24, 2011.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Tor Lattimore, Csaba Szepesvari, and Gellert Weisz. Learning with good feature representations in bandits and in rl with a generative model. In *International Conference on Machine Learning*, pp. 5662–5670. PMLR, 2020.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Shuang Liu and Hao Su. Provably efficient kernelized q-learning. *arXiv preprint arXiv:2204.10349*, 2022.

Viraj Mehta, Ian Char, Joseph Abbate, Rory Conlin, Mark D Boyer, Stefan Ermon, Jeff Schneider, and Willie Neiswanger. Exploration via planning for information about the optimal trajectory. In *Advances in Neural Information Processing Systems*, volume 35, 2022a.

Viraj Mehta, Biswajit Paria, Jeff Schneider, Stefano Ermon, and Willie Neiswanger. An experimental design perspective on model-based reinforcement learning. In *International Conference on Learning Representations*, 2022b.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Willie Neiswanger, Ke Alexander Wang, and Stefano Ermon. Bayesian algorithm execution: Estimating computable properties of black-box functions using mutual information. In *International Conference on Machine Learning*, pp. 8005–8015. PMLR, 2021.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/8d8818c8e140c64c743113f563cf750f-Paper.pdf.

Michael Pearce and Juergen Branke. Continuous multi-task bayesian optimisation with correlation. *European Journal of Operational Research*, 270(3):1074–1085, 2018.

Michael Pearce, Janis Klaise, and Matthew Groves. Practical bayesian optimization of objectives with conditioning variables. *arXiv preprint arXiv:2002.09996*, 2020.

Victor Picheny, Tobias Wagner, and David Ginsbourger. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and multidisciplinary optimization*, 48(3):607–626, 2013.

Paul J Schweitzer and Abraham Seidmann. Generalized polynomial approximations in markovian decision processes. *Journal of mathematical analysis and applications*, 110(2):568–582, 1985.

Jaemin Seo, Y-S Na, B Kim, CY Lee, MS Park, SJ Park, and YH Lee. Feedforward beta control in the kstar tokamak by deep reinforcement learning. *Nuclear Fusion*, 61(10):106010, 2021.

Roshan Shariff and Csaba Szepesvári. Efficient planning in large mdps with weak linear function approximation. *Advances in Neural Information Processing Systems*, 33:19163–19174, 2020.

Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In *International conference on machine learning*, pp. 5779–5788. PMLR, 2019.

Gregor Simm, Robert Pinsler, and José Miguel Hernández-Lobato. Reinforcement learning for molecular design guided by quantum mechanics. In *International Conference on Machine Learning*, pp. 8959–8969. PMLR, 2020a.

Gregor NC Simm, Robert Pinsler, Gábor Csányi, and José Miguel Hernández-Lobato. Symmetry-aware actor-critic for 3d molecular design. In *International Conference on Learning Representations*, 2020b.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *International Conference on Machine Learning*, 2010.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. *Advances in neural information processing systems*, 26, 2013.

Csaba Szepesvári. Lecture notes in reinforcement learning theory, Aug 2022. URL `https://rltheory.github.io/lecture-notes/planning-in-mdps/lec13/`.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Bingyan Wang, Yuling Yan, and Jianqing Fan. Sample-efficient reinforcement learning for linearly-parameterized mdps with a generative model. *Advances in Neural Information Processing Systems*, 34:23009–23022, 2021.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

Zhuoran Yang, Chi Jin, Zhaoran Wang, Mengdi Wang, and Michael I Jordan. On function approximation in reinforcement learning: Optimism in the face of large state spaces. *arXiv preprint arXiv:2011.04622*, 2020.

Dong Yin, Botao Hao, Yasin Abbasi-Yadkori, Nevena Lazić, and Csaba Szepesvári. Efficient local planning with linear function approximation. In *International Conference on Algorithmic Learning Theory*, pp. 1165–1192. PMLR, 2022.

Andrea Zanette, Alessandro Lazaric, Mykel Kochenderfer, and Emma Brunskill. Learning near optimal policies with low inherent bellman error. In *International Conference on Machine Learning*, pp. 10978–10989. PMLR, 2020.

Dongruo Zhou, Quanquan Gu, and Csaba Szepesvari. Nearly minimax optimal reinforcement learning for linear mixture markov decision processes. In *Conference on Learning Theory*, pp. 4532–4576. PMLR, 2021.

A APPENDIX

A.1 AUXILIARY RESULTS

Lemma A.1. *Let $t \in [T]$. Then, for every $(s,a) \in \mathcal{S} \times \mathcal{A}$,*

1. *If $\overline{Q}_h^t(s,a) \geq T_h^* \overline{Q}_{h+1}^t(s,a)$ holds for all $h \in [H]$, then $\overline{Q}_h^t(s,a) \geq Q_h^*(s,a)$ is true for all $h \in [H]$.*
2. *If $\underline{Q}_h^t(s,a) \leq T_h^* \underline{Q}_{h+1}^t(s,a)$ holds for all $h \in [H]$, then $Q_h^*(s,a) \geq \underline{Q}_h^t(s,a)$ is true for all $h \in [H]$.*

Proof. In order to prove part 1., let $s \in \mathcal{S}$ and $a \in \mathcal{A}$ and assume $\overline{Q}_h^t(s,a) \geq T_h^* \overline{Q}_{h+1}^t(s,a)$ for all $h \in [H]$ and $t \in [T]$. We prove $\forall h \in [H]$, $\overline{Q}_h^t(s,a) \geq Q_h^*(s,a)$ by induction on $h = H, H-1, \ldots, 1$. For the initial case $h = H$, we have

$$\begin{aligned}
\overline{Q}_H^t(s,a) &\overset{\text{assumption}}{\geq} T_h^* \overline{Q}_{H+1}^t(s,a) && (26)\\
&\overset{\text{Def. of } T_h^*}{=} r_H(s,a) + \mathbb{E}_{s' \sim \mathbb{P}_H(\cdot|s,a)} \Big[\max_{a' \in \mathcal{A}} \overline{Q}_{H+1}^t(s',a') \Big] && (27)\\
&= r_H(s,a) && (28)\\
&= Q_H^*(s,a). && (29)
\end{aligned}$$

For the inductive step, we assume that $Q_{h+1}^*(s,a) \leq \overline{Q}_{h+1}^t(s,a)$. Then,

$$\begin{aligned}
Q_h^*(s,a) &= T_h^* Q_{h+1}^*(s,a) && (30)\\
&\overset{\text{Def. of } T_h^*}{=} r_h(s,a) + \mathbb{E}_{s' \sim \mathbb{P}_h(\cdot|s,a)} \Big[\max_{a' \in \mathcal{A}} Q_{h+1}^*(s',a') \Big] && (31)\\
&\overset{\text{inductive hypothesis}}{\leq} r_h(s,a) + \mathbb{E}_{s' \sim \mathbb{P}_h(\cdot|s,a)} \Big[\max_{a' \in \mathcal{A}} \overline{Q}_{h+1}^t(s',a') \Big] && (32)\\
&\overset{\text{Def. of } T_h^*}{=} T_h^* \overline{Q}_{h+1}^t(s,a) && (33)\\
&\overset{\text{assumption}}{\leq} \overline{Q}_h^t(s,a). && (34)
\end{aligned}$$

This shows $\overline{Q}_h^t(s,a) \geq Q_h^*(s,a)$ for all $h \in [H]$ and thus concludes the proof of the first claim. The second part can be shown analogously.

□

The following is a standard result that can be found in multiple works.

Lemma A.2. *Consider a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $k(x,x) \leq 1$ for every $x \in \mathcal{X}$. Then for all $h \in [H]$ and $\lambda \geq 1$ we have*

$$\sum_{t=1}^{T} \sigma_h^t(s_h^t, a_h^t) \leq \sqrt{3\Gamma_k(T,\lambda)T}. \tag{35}$$

Proof. We can for example invoke the result of Lemma 3 in Bogunovic & Krause (2021) that in our notation reads as:

$$\sum_{t=1}^{T} \sigma_h^t(s_h^t, a_h^t) \leq \sqrt{\lambda^{-1}(2\lambda+1)\Gamma_k(T,\lambda)T}, \tag{36}$$

for $\lambda > 0$. Setting $\lambda \geq 1$, we obtain

$$\sum_{t=1}^{T} \sigma_h^t(s_h^t, a_h^t) \leq \sqrt{3\Gamma_k(T,\lambda)T}. \tag{37}$$

□

A.2 PROOF OF THEOREM 4.2

Let $\hat{\pi}_T$ be the best-policy estimate returned by the algorithm. Recall the definition of

$$\pi_T^{*\geq h} := \left(\pi_{T,h'}^{*\geq h}\right)_{h'=1}^{H} := \begin{cases} \hat{\pi}_{T,h'} & \text{for } h' = 1, \ldots, h-1 \\ \pi_{h'}^{*} & \text{for } h' = h, \ldots, H \end{cases} \tag{38}$$

as the policy that equals our best-policy estimate $\hat{\pi}_T$ until step $h-1$ and then equals the optimal policy π^*.

We start the proof with the following useful lemma.

Lemma A.3. *Let $\hat{\pi}_T$ be a best-policy estimate, let $s \in \mathcal{S}$ be an initial state, and let $h \in [H]$. Using the notation from Eq.* (38)*, we obtain*

$$V_1^{\pi_T^{*\geq h}}(s) - V_1^{\pi_T^{*\geq h+1}}(s) = \mathbb{E}_{a_1,\ldots,s_h \textit{ following } \hat{\pi}_T}\Big[Q_h^*\big(s_h, \pi_h^*(s_h)\big) - Q_h^*\big(s_h, \hat{\pi}_{T,h}(s_h)\big)\big|s_1 = s\Big].$$

Proof. To formally prove the lemma, we first explicitly express $V_1^{\pi_T^{*\geq h}}(s)$ and $V_1^{\pi_T^{*\geq h+1}}(s)$ for an arbitrary initial state $s \in \mathcal{S}$ as

$$V_1^{\pi_T^{*\geq h}}(s) \tag{39}$$

$$= \mathbb{E}_{a_1,\ldots,s_H \text{ following } \pi_T^{*\geq h}|s_1=s}\Big[\sum_{h'=1}^{H} r_{h'}(s_{h'}, a_{h'})\Big] \tag{40}$$

$$= \mathbb{E}_{a_1,\ldots,s_h \text{ following } \hat{\pi}_T|s_1=s}\Bigg[\mathbb{E}_{a_h,\ldots,s_H \text{ following } \pi^*|s_h}\Big[\sum_{h'=1}^{H} r_{h'}(s_{h'}, a_{h'})\Big]\Bigg] \tag{41}$$

$$= \mathbb{E}_{a_1,\ldots,s_h \text{ following } \hat{\pi}_T|s_1=s}\Bigg[\sum_{h'=1}^{h} r_{h'}(s_{h'}, a_{h'}) + \mathbb{E}_{a_h,\ldots,s_H \text{ following } \pi^*|s_h}\Big[\sum_{h'=h+1}^{H} r_{h'}(s_{h'}, a_{h'})\Big]\Bigg], \tag{42}$$

and

$$V_1^{\pi_T^{*\geq h+1}}(s_1) \tag{43}$$

$$= \mathbb{E}_{a_1,\ldots,s_H \text{ following } \pi_T^{*\geq h+1}|s_1=s}\Big[\sum_{h'=1}^{H} r_{h'}(s_{h'}, a_{h'})\Big] \tag{44}$$

$$= \mathbb{E}_{a_1,\ldots,s_h \text{ following } \hat{\pi}_T|s_1=s}\Bigg[\mathbb{E}_{a_h,s_{h+1} \text{ following } \hat{\pi}_T|s_h}\Bigg[\mathbb{E}_{a_{h+1},\ldots,s_H \text{ following } \pi^*|s_{h+1}}$$
$$\Big[\sum_{h'=1}^{h} r_{h'}(s_{h'}, a_{h'}) + \sum_{h'=h+1}^{H} r_{h'}(s_{h'}, a_{h'})\Big]\Bigg]\Bigg] \tag{45}$$

$$= \mathbb{E}_{a_1,\ldots,s_h \text{ following } \hat{\pi}_T|s_1=s}\Bigg[\sum_{h'=1}^{h} r_{h'}(s_{h'}, a_{h'})$$
$$+ \mathbb{E}_{a_h,s_{h+1} \text{ following } \hat{\pi}_T|s_h}\Bigg[\mathbb{E}_{a_{h+1},\ldots,s_H \text{ following } \pi^*|s_{h+1}}\Big[\sum_{h'=h+1}^{H} r_{h'}(s_{h'}, a_{h'})\Big]\Bigg]\Bigg]. \tag{46}$$

Eqs. (40) and (44) use the definition of V_1^{π}, Eqs. (41) and (45) use the definition of $\pi_T^{*\geq h}$ and $\pi_T^{*\geq h+1}$ from Eq. (38), and Eqs. (42) and (46) use the property that integration is a linear operator.

Lemma A.3 then follows from Eqs. (42) and (46) as well as the definition of Q_h^*:

$$V_1^{\pi_T^{*\geq h}}(s_1) - V_1^{\pi_T^{*\geq h+1}}(s_1) = \mathbb{E}_{a_1,\ldots,s_h \text{ following } \pi^T|s_1=s}\Bigg[\sum_{h'=1}^{h} r_{h'}(s_{h'}, a_{h'}) - \sum_{h'=1}^{h} r_{h'}(s_{h'}, a_{h'})$$

$$+ \mathbb{E}_{a_h,\ldots,s_H \text{ following } \pi^*|s_h}\left[\sum_{h'=h+1}^{H} r_{h'}(s_{h'}, a_{h'})\right]$$

$$- \mathbb{E}_{a_h,s_{h+1} \text{ following } \hat{\pi}_T|s_h}\left[\mathbb{E}_{a_{h+1},\ldots,s_H \text{ following } \pi^*|s_{h+1}}\left[\sum_{h'=h+1}^{H} r_{h'}(s_{h'}, a_{h'})\right]\right] \tag{47}$$

$$= \mathbb{E}_{a_1,\ldots,s_h \text{ following } \hat{\pi}_T|s_1=s}\left[Q_h^*\big(s_h, \pi_h^*(s_h)\big) - Q_h^*\big(s_h, \hat{\pi}_{T,h}(s_h)\big)\right]. \tag{48}$$

□

We proceed with the proof by using the notation from Eq. (38). We can decompose the instantaneous regret for an arbitrary initial state $s \in \mathcal{S}$ as follows:

$$V_1^*(s) - V_1^{\hat{\pi}_T}(s) = V_1^{\pi_T^{*\geq 1}}(s) - V_1^{\pi_T^{*\geq H+1}}(s) \tag{49}$$

$$= \sum_{h=1}^{H}\left(V_1^{\pi_T^{*\geq h}}(s) - V_1^{\pi_T^{*\geq h+1}}(s)\right) \tag{50}$$

$$\overset{\text{Lemma A.3}}{=} \sum_{h=1}^{H} \mathbb{E}_{s_1,a_1,\ldots,s_h \text{following } \hat{\pi}_T}\left[Q_h^*\big(s_h, \pi_h^*(s_h)\big) - Q_h^*\big(s_h, \hat{\pi}_{T,h}(s_h)\big)\big|s_1 = s\right]. \tag{51}$$

The intuition behind Lemma A.3 used in Eq. (51) is as follows. Both $V_1^{\pi_T^{*\geq h}}(s)$ and $V_1^{\pi_T^{*\geq h+1}}(s)$ refer to the same random trajectory segment $(s_1, a_1, \ldots, s_h)$ until step h (i.e., the same initial state and policy are used), which is captured as $\mathbb{E}_{s_1,a_1,\ldots,s_h \text{ following } \hat{\pi}_T}[\cdot]$. For the remaining steps $h, \ldots, H$, the policies only differ at step h, a property which is captured in the difference $Q_h^*\big(s_h, \pi_h^*(s_h)\big) - Q_h^*\big(s_h, \hat{\pi}_{T,h}(s_h)\big)$.

Conditioning on the event in Lemma 4.1 holding true and by invoking Lemma A.1, we have that:

$$\underline{Q}_h^t(s,a) \leq Q_h^*(s,a) \leq \overline{Q}_h^t(s,a), \tag{52}$$

holds for every $h \in [H]$, $t \in [T]$, and $(s,a) \in \mathcal{S} \times \mathcal{A}$. Next, we proceed to bound $Q_h^*\big(\cdot, \pi_h^*(\cdot)\big) - Q_h^*\big(\cdot, \hat{\pi}_{T,h}(\cdot)\big)$ from Eq. (51) uniformly on $\mathcal{S}$. We have:

$$Q_h^*(s, \pi_h^*(s)) - Q_h^*(s, \hat{\pi}_{T,h}(s)) \overset{\text{Eq. (52)}}{\leq} Q_h^*(s, \pi_h^*(s)) - \max_{t\in[T]} \underline{Q}_h^t(s, \hat{\pi}_{T,h}(s)) \tag{53}$$

$$\overset{\text{Def. of } \hat{\pi}_{T,h}}{=} Q_h^*(s, \pi_h^*(s)) - \max_{a\in\mathcal{A}} \max_{t\in[T]} \underline{Q}_h^t(s,a) \tag{54}$$

$$= \min_{t\in[T]}\left(Q_h^*(s, \pi_h^*(s)) - \max_{a\in\mathcal{A}} \underline{Q}_h^t(s,a)\right) \tag{55}$$

$$\overset{\text{Def. of } \pi_h^*}{=} \min_{t\in[T]}\left(\max_{a\in\mathcal{A}} Q_h^*(s,a) - \max_{a\in\mathcal{A}} \underline{Q}_h^t(s,a)\right) \tag{56}$$

$$\overset{\text{Eq. (52)}}{\leq} \min_{t\in[T]}\left(\max_{a\in\mathcal{A}} \overline{Q}_h^t(s,a) - \max_{a\in\mathcal{A}} \underline{Q}_h^t(s,a)\right) \tag{57}$$

$$\overset{\text{Eq. (12)}}{\leq} \min_{t\in[T]}\left(\max_{a\in\mathcal{A}} \overline{Q}_h^t(s_h^t,a) - \max_{a\in\mathcal{A}} \underline{Q}_h^t(s_h^t,a)\right) \tag{58}$$

$$\overset{\text{Eq. (13)}}{\leq} \min_{t\in[T]}\left(\overline{Q}_h^t(s_h^t,a_h^t) - \underline{Q}_h^t(s_h^t,a_h^t)\right) \tag{59}$$

$$\leq \frac{1}{T}\sum_{t=1}^{T}\left(\overline{Q}_h^t(s_h^t,a_h^t) - \underline{Q}_h^t(s_h^t,a_h^t)\right). \tag{60}$$

Next, for convenience we introduce the notation

$$d_h^t := \left(\overline{Q}_h^t(s_h^t, a_h^t) - T_h^* \overline{Q}_{h+1}^t(s_h^t, a_h^t)\right) + \left(T_h^* \underline{Q}_{h+1}^t(s_h^t, a_h^t) - \underline{Q}_h^t(s_h^t, a_h^t)\right), \tag{61}$$

and obtain the following upper bound on $\overline{Q}_h^t(s_h^t, a_h^t) - \underline{Q}_h^t(s_h^t, a_h^t)$ (from Eq. (60)) for every $h \in [H]$, $t \in [T]$:

$$\overline{Q}_h^t(s_h^t, a_h^t) - \underline{Q}_h^t(s_h^t, a_h^t) \overset{\text{Eq. (61)}}{=} d_h^t + T_h^* \overline{Q}_{h+1}^t(s_h^t, a_h^t) - T_h^* \underline{Q}_{h+1}^t(s_h^t, a_h^t) \tag{62}$$

$$\overset{\text{Def. of } T_h^*}{=} d_h^t + \mathbb{E}_{s' \sim \mathbb{P}_h(\cdot|s_h^t, a_h^t)} \left(\max_{\overline{a} \in \mathcal{A}} \overline{Q}_{h+1}^t(s', \overline{a}) - \max_{\underline{a} \in \mathcal{A}} \underline{Q}_{h+1}^t(s', \underline{a}) \right) \tag{63}$$

$$\leq d_h^t + \max_{s' \in \mathcal{S}} \left(\max_{\overline{a} \in \mathcal{A}} \overline{Q}_{h+1}^t(s', \overline{a}) - \max_{\underline{a} \in \mathcal{A}} \underline{Q}_{h+1}^t(s', \underline{a}) \right) \tag{64}$$

$$\overset{\text{Eq. (12)}}{=} d_h^t + \left(\max_{\overline{a} \in \mathcal{A}} \overline{Q}_{h+1}^t(s_{h+1}^t, \overline{a}) - \max_{\underline{a} \in \mathcal{A}} \underline{Q}_{h+1}^t(s_{h+1}^t, \underline{a}) \right) \tag{65}$$

$$\overset{\text{Eq. (13)}}{\leq} d_h^t + \left(\overline{Q}_{h+1}^t(s_{h+1}^t, a_{h+1}^t) - \underline{Q}_{h+1}^t(s_{h+1}^t, a_{h+1}^t) \right). \tag{66}$$

Using the definition of $\underline{Q}_{H+1}^t$ and $\overline{Q}_{H+1}^t$ as the zero functions, we can unroll the recursive inequality from Eq. (66) and upper bound $\overline{Q}_h^t(s_h^t, a_h^t) - \underline{Q}_h^t(s_h^t, a_h^t)$ for every $h \in [H]$, $t \in [T]$ as follows:

$$\overline{Q}_h^t(s_h^t, a_h^t) - \underline{Q}_h^t(s_h^t, a_h^t) \leq \sum_{h'=h}^{H} d_{h'}^t. \tag{67}$$

$$\overline{Q}_h^t(s_h^t, a_h^t) - \underline{Q}_h^t(s_h^t, a_h^t) \tag{68}$$

$$\leq \sum_{h'=h}^{H} \left[\left(\overline{Q}_{h'}^t(s_{h'}^t, a_{h'}^t) - T_{h'}^* \overline{Q}_{h'+1}^t(s_{h'}^t, a_{h'}^t) \right) + \left(T_{h'}^* \underline{Q}_{h'+1}^t(s_{h'}^t, a_{h'}^t) - \underline{Q}_{h'}^t(s_{h'}^t, a_{h'}^t) \right) \right] \tag{69}$$

$$\overset{\text{Lemma 4.1}}{\leq} \sum_{h'=h}^{H} 4\beta \sigma_{h'}^t(s_{h'}^t, a_{h'}^t). \tag{70}$$

By substituting the bound from Eq. (70) in Eq. (60), and then in Eq. (51), we arrive at:

$$V_1^*(s) - V_1^{\hat{\pi}_T}(s) \leq 4\beta \sum_{h=1}^{H} \sum_{h'=h}^{H} \frac{1}{T} \sum_{t=1}^{T} \sigma_{h'}^t(s_{h'}^t, a_{h'}^t) \leq 2\sqrt{3}\beta H(H+1)\sqrt{\tfrac{\Gamma_k(T,\lambda)}{T}}, \tag{71}$$

where the last inequality follows from Lemma A.2. Since Eq. (71) holds for any $s \in S$, we arrive at our main result:

$$\|V_1^* - V_1^{\hat{\pi}_T}\|_{\ell^\infty(\mathcal{S})} \leq 2\sqrt{3}\beta H(H+1)\sqrt{\tfrac{\Gamma_k(T,\lambda)}{T}}. \tag{72}$$

A.3 Offline contextual Bayesian optimization

Algorithm 2 AE-LSVI for offline contextual Bayesian optimization

Require: kernel function $k(\cdot, \cdot)$, exploration parameter $\beta > 0$, regularization parameter λ,

for $t = 1, \ldots, T$ **do**
  Obtain $\overline{Q}^t$ and $\underline{Q}^t$ from Equation (23)
  Choose $s^t \in \arg\max_{s \in S} \left[\max_{a \in A} \overline{Q}^t(s, a) - \max_{a \in A} \underline{Q}^t(s, a) \right]$
  Choose $a^t \in \arg\max_{a \in \mathcal{A}} \overline{Q}^t(s^t, a)$
  Observe the reward $y_t = Q^*(s^t, a^t) + \eta_t$
end for
Output the policy estimate $\hat{\pi}_T$ such that $\hat{\pi}_T(\cdot) = \arg\max_{a \in \mathcal{A}} \max_{t \in [T]} \underline{Q}^t(s, a)$

Our algorithm for the offline contextual Bayesian optimization is presented in Algorithm 2. As a side observation, we note that similarly to Char et al. (2019), we can also simply incorporate context weights (i.e., given $\omega(s)$ that represents some weighting of context s that may depend on the probability of seeing s at evaluation time or the importance of s), in case they are available, into the proposed acquisition function, i.e.,

$$s^t \in \arg\max_{s \in S} \Big[\big(\max_{a \in A} \overline{Q}^t(s,a) - \max_{a \in A} \underline{Q}^t(s,a) \big) w(s) \Big]. \tag{73}$$

A.3.1 Proof of Corollary 5.1

Proof. In this proof, we condition on the event in Eq. (24) holding true. Similar arguments to the ones in Eq. (62) – Eq. (66) lead to the following for every $s \in \mathcal{S}$:

$$\begin{aligned}
\max_{a \in \mathcal{A}} Q^*(s,a) - Q^*(s, \hat{\pi}_T(s)) &\overset{\text{Eq. (24)}}{\leq} \max_{a \in \mathcal{A}} Q^*(s,a) - \max_{t \in [T]} \underline{Q}^t(s, \hat{\pi}_T(s)) && (74)\\
&\overset{\text{Def. of } \hat{\pi}_T}{=} \max_{a \in \mathcal{A}} Q^*(s,a) - \max_{a \in \mathcal{A}} \max_{t \in [T]} \underline{Q}^t(s,a) && (75)\\
&= \min_{t \in [T]} \Big(\max_{a \in \mathcal{A}} Q^*(s,a) - \max_{a \in \mathcal{A}} \underline{Q}^t(s,a) \Big) && (76)\\
&\overset{\text{Eq. (24)}}{\leq} \min_{t \in [T]} \Big(\max_{a \in \mathcal{A}} \overline{Q}^t(s,a) - \max_{a \in \mathcal{A}} \underline{Q}^t(s,a) \Big) && (77)\\
&\overset{\text{Def. of } s^t}{\leq} \min_{t \in [T]} \Big(\max_{a \in \mathcal{A}} \overline{Q}^t(s^t,a) - \max_{a \in \mathcal{A}} \underline{Q}^t(s^t,a) \Big) && (78)\\
&\overset{\text{Def. of } a^t}{\leq} \min_{t \in [T]} \Big(\overline{Q}^t(s^t,a^t) - \underline{Q}^t(s^t,a^t) \Big) && (79)\\
&\leq \frac{1}{T} \sum_{t=1}^{T} \Big(\overline{Q}^t(s^t,a^t) - \underline{Q}^t(s^t,a^t) \Big) && (80)\\
&\overset{\text{Eq. (23)}}{=} \frac{1}{T} \sum_{t=1}^{T} \Big(2\beta_t \sigma^t(s^t,a^t) \Big) && (81)\\
&\leq \frac{2\beta_T}{T} \sum_{t=1}^{T} \sigma^t(s^t,a^t) && (82)\\
&\overset{\text{Lemma A.2}}{\leq} \frac{2\beta_T \sqrt{3\Gamma_k(T,\lambda)}}{\sqrt{T}}. && (83)
\end{aligned}$$

Finally, by setting $\epsilon \geq \frac{2\beta_T \sqrt{3\Gamma_k(T,\lambda)}}{\sqrt{T}}$ and expressing it in terms of T, we arrive at the main result. □

B Additional Experimental Details

B.1 Implementation

We use an exact Gaussian Process with a squared exponential kernel for the value function regression in Eq. (5). We fit the kernel hyperparameters at each iteration using 1000 iterations of Adam (Kingma & Ba, 2014), maximizing the marginal log likelihood of the training data. We used the TinyGP package (Foreman-Mackey, 2021) built on top of JAX (Bradbury et al., 2018) in order to take advantage of JIT compilation. All experiments are conducted with a fixed bonus $\beta = 0.5$. We uniformly sample 1,000 points from the state space and evaluate them to find an approximate maximizer to the objective in Eq. (12).

B.2 ENVIRONMENTS

Each environment is defined with a native reward function taken from the literature. We established upper and lower bounds on the reward function value and used them to scale the reward function values to $[0, 1]$ so that our environments would match the theoretical results in this paper.

Cartpole We use a modified version of the cartpole environment from Mehta et al. (2022a) that has dense rewards as implemented in Wang et al. (2019). The state space is $4D$ and consists of the horizontal position and velocity of the cart as well as the angular position and velocity of the pole. p_0 in this environment is a normal distribution centered with the cart below the goal horizontally with the pole hanging down with very small variance. p_0' is the same distribution displaced 5 meters to the right.

Navigation This is a $2D$ navigation problem with dynamics of the form $s_{t+1} = s_t + B(s_t)a_t$, where $B(t) = \begin{bmatrix} \sin(x_2/10) + 4 & 0 \\ 0 & 1.5\cos(x_1/10) - 2 \end{bmatrix}$. The goal is fixed at $\begin{bmatrix} 6 \\ 9 \end{bmatrix}$. We define p_0 to be the uniform distribution over the axis-aligned rectangle given by corners $\begin{bmatrix} -8 \\ -9 \end{bmatrix}$ and $\begin{bmatrix} -6 \\ -6 \end{bmatrix}$. We define p_0' to be the uniform distribution over the axis-aligned rectangle given by corners $\begin{bmatrix} 1 \\ 4 \end{bmatrix}$ and $\begin{bmatrix} 3 \\ 7 \end{bmatrix}$. The reward function at every timestep is simply the negative ℓ_1-distance between the agent and the goal.

β Tracking and β + Rotation Our two simulated plasma control problems are taken from Mehta et al. (2022a), which gives a thorough description of their relevance to the problem of nuclear fusion. At a high level, β_N is a normalized plasma pressure ratio that is correlated with the economic output of a fusion reactor. Our β **Tracking** environment aims to adjust the injected power in the reactions in order to achieve a target value of $\beta_N = 2\%$. The initial state distribution p_0 is taken from a set of real datapoints from shots on the DIII-D tokamak in San Diego. Our alternate initial state distribution p_0' consists of simply adding 0.4 to each component of a vector sampled from p_0. The reward function is the negative ℓ_1-distance betwen the β_N value and 2%. The dynamics are given by a learned model of the plasma state as introduced in Char et al. (2022).

The β **+ Rotation** environment is a more complex plasma control problem, introducing an additional actuator (injected torque) and an additional control objective (controlling plasma rotation). Control of plasma rotation is key to plasma stability and this is a reduced version of the realistic problem. This environment also uses a model from Char et al. (2022) for the dynamics, real plasma states for the initial state distribution p_0, and a fixed translation for the alternate initial state distribution p_0'. We also include a randomly drawn target for β_N and rotation in the state space for every episode.