Selective Uncertainty Propagation in Offline RL

Sanath Kumar Krishnamurthy[1], Tanmay Gangwani[2], Sumeet Katariya[1], Branislav Kveton[3], Shrey Modi[4], Anshuka Rangi[2]

[1]Meta
[2]Amazon
[3]Adobe
[4]Indian Institute of Technology, Bombay
sanathsk@meta.com

Abstract

We consider the finite-horizon offline reinforcement learning (RL) setting, and are motivated by the challenge of learning the policy at any step h in dynamic programming (DP) algorithms. To learn this, it is sufficient to evaluate the treatment effect of deviating from the behavioral policy at step h after having optimized the policy for all future steps. Since the policy at any step can affect next-state distributions, the related distributional shift challenges can make this problem far more statistically hard than estimating such treatment effects in the stochastic contextual bandit setting. However, the hardness of many real-world RL instances lies between the two regimes. We develop a flexible and general method called selective uncertainty propagation for confidence interval construction that adapts to the hardness of the associated distribution shift challenges. We show benefits of our approach on toy environments and demonstrate the benefits of these techniques for offline policy learning.

1 Introduction

We study the finite-horizon offline reinforcement learning (RL) problem, focusing on algorithms that adapt to instance hardness. At a high-level, we study algorithms that provide better guarantees for contextual bandit (CB) like instances while being able to plan in more dynamic RL-like instances.

Our work is motivated by real-world RL problems, such as user interaction with an e-commerce search engine (recommendation system). Here, the state can be a user query, and the action is the product recommendation from the engine. When the user wants to buy a particular product, the user often only enters a single product query unrelated to the previous one; thus resembling a sequence of CB problems. On the other hand, when the user explores products, the exploration queries are related through the user's intent, and the recommendation system may want to steer the user toward the ideal product. Hence, this resembles the RL setting. This indicates the need to develop unified solutions that integrate CB and RL techniques – adapting to instance hardness. We now introduce the CB and RL frameworks in more detail.

Stochastic contextual bandits (CBs) (Langford and Zhang 2008; Li et al. 2010) and finite-horizon *reinforcement learning (RL)* (Sutton 1988; Williams 1992; Sutton and Barto 1998) are two fundamental frameworks for decision-making under uncertainty. In stochastic CBs, the environment samples the context and corresponding rewards (for each action) from a fixed but unknown distribution; the agent then observes the context and learns to select the most rewarding action conditioned on the context.

Finite-horizon RL is a generalization of CBs where contexts become *states* and a sequence of decisions are to be made over H steps. Similar to the CB problem, at each step, the agent observes the current state, selects an action conditioned on the current state, and receives a reward sampled by the environment from a corresponding conditional distribution. However, unlike the CB problem, while the initial state is sampled from a fixed but unknown distribution, the next state at any step depends on the current state and the agent's action. Hence, the agent can plan to attain high cumulative reward by learning to reach high-value future states.

Unfortunately, the fact that actions can influence future states implies that the agent needs to learn under state-distribution shifts making the RL setting much more statistically harder than CBs in the worst case. For example, (Foster et al. 2021) show that the worst-case sample complexity to learn a non-trivial offline RL policy is either polynomial in the state space size or exponential in other parameters.[1] On the other hand, if actions do not influence next-state distributions at any step, the RL instance would be equivalent to solving H stochastic CB instances. On such instances, offline bandit algorithms (Foster and Syrgkanis 2019) would enjoy a polynomial sample complexity for policy learning with no dependence on state space size. Hence, for such instances, state-of-the-art offline RL algorithms such as pessimistic value function optimization (Jin, Yang, and Wang 2021) may be unnecessarily conservative.

We formalize this dichotomy and show that the statistical hardness of offline RL instances can be captured by the size of actions' impact on the next state's distribution. To show this, we consider the high-level structure of dynamic programming (DP) algorithms for offline RL (e.g. Jin, Yang, and Wang 2021). DP algorithms construct a policy itera-

[1](Foster et al. 2021) consider the discounted infinite horizon offline RL formulation. However, one should expect similar lower bounds for the finite horizon offline RL formulation.

tively starting from the policy for the final step and ending by constructing the policy for the first step. At any step h, DP algorithms can be viewed to select the policy at step h that maximizes the treatment effect of deviating from the behavioral policy at step h after having optimized the policy for all future steps. The goal of this paper is to estimate and construct good confidence intervals for this treatment effect at step h.

Our primary focus is on confidence interval (CI) construction, which is motivated by the fact that many successful offline RL algorithms learn a policy that maximizes the lower bound of constructed CIs (Jin, Yang, and Wang 2021). To account for estimation errors from future steps, standard methods for CI construction at any step propagate uncertainty from future steps to the current step h. This paper seeks to construct better CIs that adapt to instance hardness by selectively propagating uncertainty. In cases where all actions have zero estimated impact on next-state distributions, our procedure does not propagate any uncertainty from later steps and still constructs valid CIs for the treatment effect of deviating from the behavioral policy at step h after having optimized the policy for all future steps. It treats the instance like a CB problem – hence enjoying a polynomial sample complexity with no dependence on state space size for treatment effect estimation. For more dynamic instances, our procedure must unavoidably propagate more uncertainty from future steps in order to continue constructing valid CIs. In this way, we adapt to the hardness of the instance for CI construction at any step. We also show the benefits of this approach for offline policy learning by proposing an algorithm that optimizes our constructed CIs. Simple simulations further support our claim.

Related Work: Both bandits and RL have been studied extensively (Lattimore and Szepesvari 2019; Sutton and Barto 1998; Foster and Rakhlin 2023). In bandits, the focus has been on achieving higher statistical efficiency by using the reward distribution of actions (Garivier and Cappe 2011), prior distribution of model parameters (Thompson 1933; Agrawal and Goyal 2012; Chapelle and Li 2012; Russo et al. 2018), parametric structure (Dani, Hayes, and Kakade 2008; Abbasi-Yadkori, Pal, and Szepesvari 2011; Agrawal and Goyal 2013), or agnostic methods (Agarwal et al. 2014). In RL, the focus has been on different means of learning to plan for longer horizons, such as the value function (Sutton 1988), policy (Williams 1992), or their combination (Sutton et al. 2000). Just as in our work, causal inference insights have helped improve the statistical efficiency of both CB and RL algorithms (Krishnamurthy, Wu, and Syrgkanis 2018; Carranza, Krishnamurthy, and Athey 2023; Syrgkanis and Zhan 2023). However, bridging the gap between bandits and RL is an exciting and relatively underexplored research direction. One way to define this gap is to argue that in bandit-like environments, the state never changes once initially sampled. These bandit-like environments can be viewed as a special case of the situation where actions do not impact next-state distributions. With bridging this gap as one motivation, (Zanette and Brunskill 2019; Yin and Wang 2021) have used variance-dependent Bernstein bounds to limit uncertainty propagation when there is a lack of next-step value function heterogeneity. Another approach is to define this gap in a binary fashion. Either there is no impact of actions on next state distributions, or we are in a dynamic MDP environment. In an online setting, (Zhang, Gottesman, and Doshi-Velez 2022) develop hypothesis tests to differentiate between the two situations and then select the most appropriate exploration algorithm. While their higher-level framing is similar to ours and their approach is novel, their approach cannot outperform existing RL algorithms in MDP environments. By interpolating between the two regimes, we hope to outperform bandit and existing RL algorithms that either forgo planning or are too conservative in accounting for actions' impact on next-state distributions.

2 Preliminaries

Setting: We consider an episodic Markov Decision Process (MDP) setting with state space $\mathcal{X}$, action space $\mathcal{A}$, horizon H, and transition kernel $P = (P^{(h)})_{h=1}^{H}$. At every episode, the environment samples a starting state $x^{(1)}$ and a set of realized rewards $r = (r^{(h)})_{h=1}^{H}$ from a fixed but unknown distribution D. Here $r^{(h)}$ is a map from $\mathcal{X} \times \mathcal{A}$ to $[0, 1]$. For any states $x, x' \in \mathcal{X}$ and action $a \in \mathcal{A}$, $P^{(h)}(x'|x, a)$ denotes the probability density of transitioning to state x' conditional on taking action a at state x during step h. A trajectory τ is a sequence of states, actions, and rewards. That is, any trajectory τ is given by $\tau = (x^{(h)}, a^{(h)}, r^{(h)}(x^{(h)}, a^{(h)}))_{h=1}^{H}$.

A policy π is a sequence of H action sampling kernels $\{\pi^{(h)}\}_{h=1}^{H}$, where $\pi^{(h)}(a|x)$ denotes the probability of sampling action a at state x during step h under the policy π. We let $D(\pi)$ denote the induced distribution over trajectories under the policy π. For any policy π, we define the (state-) value function $V_{\pi}^{(h)} : \mathcal{X} \to [0, H - h + 1]$ at each step $h \in [H]$ such that,

$$V_{\pi}^{(h)}(x) = \mathbb{E}_{D(\pi)}\left[\sum_{i=h}^{H} r^{(i)}(x^{(i)}, a^{(i)}) \middle| x^{(h)} = x\right]. \quad (1)$$

The value of policy π is given by $\mathbb{E}_D[V_{\pi}^{(1)}(x^{(1)})]$. We can take expectation over D instead of $D(\pi)$ here since the only random variable in $V_{\pi}^{(1)}(x^{(1)})$ is the initial state $x^{(1)}$ which does not depend on the choice of the policy π.

For any step $h \in [H]$, we let $R^{(h)}$ be a function from $\mathcal{X} \times \mathcal{A}$ to $[0, 1]$ denoting the expected reward function for step h. That is, $R^{(h)}(x, a) = \mathbb{E}_D[r^{(h)}(x, a)]$. With some abuse of notation, for any $x, x' \in \mathcal{X}$, we let $R^{(h)}(x, \pi) = \sum_a \pi^{(h)}(a|x) R^{(h)}(x, a)$ and $P^{(h)}(x'|x, \pi) = \sum_a \pi^{(h)}(a|x) P^{(h)}(x'|x, a)$. That is, $R^{(h)}(x, \pi)$ is the expected reward at state x and step h under the policy π. Similarly, $P^{(h)}(x'|x, \pi)$ is the expected transition probability from x to x' at step h under the policy π. For any step $h \in [H]$, we also let $V_{\max}{}^{(h)}$ denote a bound on the maximum value $V_{\pi}^{(h)}(x)$ can take for any state x and policy π.

It is also equivalent to define the value functions $(V_{\pi}^{(h)})$

using the iterative definition in (2), where ${V_\pi}^{(H+1)} \equiv 0$.

$$\forall h \in [H], x \in \mathcal{X},$$
$$V_\pi{}^{(h)}(x) = R^{(h)}(x,\pi) + \int_{x'} {V_\pi}^{(h+1)}(x') P^{(h)}(x'|x,\pi). \tag{2}$$

Data Collection Process: In this paper, we focus on the offline setting (Levine et al. 2020) with training data collected under a behavioral policy π_b. Apart from the policy π_b, the learner only has access to a dataset S consisting of T trajectories sampled from the distribution $D(\pi_b)$, where $D(\pi_b)$ is the data sampling distribution induced by π_b. That is, $S = \{\tau_t\}_{t=1}^T$, where $\tau_t = (x_t^{(h)}, a_t^{(h)}, r_t^{(h)}(x^{(h)}, a^{(h)}))_{h=1}^H \sim D(\pi_b)$. Since the transitions in these trajectories are induced by the behavioral policy, for notational convenience, we let $P_b = ({P_b}^{(h)})_{h=1}^H$ denote the transition kernel under the policy π_b. That is, for any $x, x' \in \mathcal{X}$, ${P_b}^{(h)}(x'|x) = P^{(h)}(x'|x, \pi_b)$.

2.1 Estimand of Interest

We now turn our attention to defining our target estimand, which refers to the specific quantity we aim to estimate. Consider a fixed policy π and suppose we would like to estimate its value. Since estimating the value of the behavioral policy π_b is easy (empirical average of total observed reward in each trajectory), we argue that that it is sufficient to estimate $\mathbb{E}_D[V_\pi^{(1)}(x^{(1)}) - V_{\pi_b}^{(1)}(x^{(1)})]$ – the difference in values between evaluation and behavioral policy. This difference can be further decomposed. For each step h, let $\tilde{\pi}_h = ({\pi_b}^{(1)}, \ldots, {\pi_b}^{(h-1)}, \pi^{(h)}, \ldots, \pi^{(H)})$ be the policy that follows the behavioral policy upto step $h-1$ and then follows the evaluation policy. In (3), we decompose the difference in policy value between the evaluation and behavioral policy into the sum of differences in policy value between $\tilde{\pi}_h$ and $\tilde{\pi}_{h+1}$ for each step h.

$$\begin{aligned}
&\mathbb{E}_D[V_\pi^{(1)}(x^{(1)}) - V_{\pi_b}^{(1)}(x^{(1)})] \\
&\overset{(i)}{=} \sum_{h=1}^H \mathbb{E}_D[V_{\tilde{\pi}_h}^{(1)}(x^{(1)}) - V_{\tilde{\pi}_{h+1}}^{(1)}(x^{(1)})] \\
&\overset{(ii)}{=} \sum_{h=1}^H \mathbb{E}_{D(\pi_b)}[V_{\tilde{\pi}_h}^{(h)}(x^{(h)}) - V_{\tilde{\pi}_{h+1}}^{(h)}(x^{(h)})].
\end{aligned} \tag{3}$$

Here (i) follows from telescoping and (ii) follows from the fact that the policies $\tilde{\pi}_h$ and $\tilde{\pi}_{h+1}$ agree with the behavioral policy for the first $h-1$ steps. *We let $\alpha_\pi^{(h)}$, the term corresponding to step h in the above decomposition, be our estimand of interest.* That is, our estimand ($\alpha_\pi^{(h)}$) is the difference in value of policies $\tilde{\pi}_h$ and $\tilde{\pi}_{h+1}$ – these policies only differ in the current step h, which may cause difference in immediate rewards and may also cause a difference in in next-state distributions (affecting future rewards even if the policies at future steps are the same).

$$\alpha_\pi^{(h)} = \mathbb{E}_{D(\pi_b)}[V_{\tilde{\pi}_h}^{(h)}(x^{(h)}) - V_{\tilde{\pi}_{h+1}}^{(h)}(x^{(h)})] \tag{4}$$

We now seek to justify $\alpha_\pi^{(h)}$ as an important estimand, and start by arguing that it is a reasonable estimand to care about. Note that, given the decomposition in (3), estimating and constructing CIs for $\{\alpha_\pi^{(h)}\}_{h=1}^H$ allows us to estimate and construct CIs for $\mathbb{E}_D[V_\pi^{(1)}(x^{(1)}) - V_{\pi_b}^{(1)}(x^{(1)})]$ (the difference in policy value between evaluation and behavioral policies) – and thus allows us to estimate and construct CIs for $\mathbb{E}_D[V_\pi^{(1)}(x^{(1)})]$ (evaluation policy value).

Beyond being an effective surrogate for policy evaluation, $\alpha_\pi^{(h)}$ is an important quantity to consider in dynamic programming (DP) algorithms. DP algorithms construct the policy for the final step ($\pi^{(H)}$) and iteratively construct policies for earlier steps. At step h, the policy at steps $h+1$ to H are already fixed/computed. Hence at this step, one can interpret DP algorithms as attempting to select $\pi^{(h)}$ in order to maximize $\alpha_\pi^{(h)}$ – that is, maximize the treatment effect of deviating from the behavioral policy at step h after having optimized the policy for all future steps. Hence, for any step h, $\alpha_\pi^{(h)}$ is a helpful estimand to consider for decision-making at step h.

Importantly for us, when actions at step h do not affect next state distributions, the problem of choosing a policy at step h can be viewed as a CB problem. Helpfully in this case, unlike policy value, $\alpha_\pi^{(h)}$ only depends on immediate rewards and can be estimated via offline stochastic CB techniques. However, when actions at step h do influence next state distributions, RL techniques are necessary for estimating $\alpha_\pi^{(h)}$. Hence, beyond being a critical quantity for decision-making at step h, it is also a quantity that is amenable to interpolating between CB and RL techniques. Thus, our paper focuses on estimating and constructing tight confidence intervals (CIs) for this estimand ($\alpha_\pi^{(h)}$).

3 Shift Model

Offline RL is more challenging than offline policy learning in the stochastic CB setting (Foster et al. 2021). The primary reason for the difference between the two settings is due to state distribution shift induced due to deviating from the behavioral policy. Distribution shift makes any statistical learning theory problem challenging (Vergara et al. 2012; Bobu et al. 2018; Farshchian et al. 2018). Hence methods that adapt to instance hardness must rely on some implicit or explicit approach to measure this state-distribution shift. To this end, we model the "heterogeneous treatment effect" (Künzel et al. 2019; Nie and Wager 2021) of actions on the next-state distribution and refer to this effect as the "shift model". More precisely, we define the shift model $\Delta = (\Delta^{(h)})_{h=1}^H$ in (5).

$$\forall (x,a), \Delta^{(h)}(\cdot|x,a) = P^{(h)}(\cdot|x,a) - {P_b}^{(h)}(\cdot|x). \tag{5}$$

Here $\Delta^{(h)}(x'|x,a)$ captures the shift in the probability of transitioning from x to x' due to selecting action a at state x instead of following the behavioral policy. With some abuse of notation, for any $x, x' \in \mathcal{X}$, we let $\Delta^{(h)}(x'|x,\pi) = \sum_a \pi^{(h)}(a|x)\Delta^{(h)}(x'|x,a)$. That is, $\Delta^{(h)}(x'|x,\pi)$ is the expected shift (w.r.t to P_b) in probability of transitioning from x to x' at step h under the policy π. It is worth

noting that shifts are bounded. For all (x, a), since the $\Delta^{(h)}(\cdot|x,a)$ is a difference of two state-distributions, we have $\|\Delta^{(h)}(\cdot|x,a)\|_1 \leq 2$ from triangle inequality.

We argue that shift helps capture instance hardness for estimating $\alpha_\pi^{(h)}$. To see this, we provide a shift-dependent expression for $\alpha_\pi^{(h)}$.

$$\begin{aligned}\alpha_\pi^{(h)} &\overset{(i)}{=} \mathbb{E}_{D(\pi_b)}[V^{(h)}_{\tilde{\pi}_h}(x^{(h)}) - V^{(h)}_{\tilde{\pi}_{h+1}}(x^{(h)})] \\ &\overset{(ii)}{=} \mathbb{E}_{D(\pi_b)}[R^{(h)}(x^{(h)},\pi) - R^{(h)}(x^{(h)},\pi_b)] \\ &+ \mathbb{E}_{D(\pi_b)}\left[\int_{x'} V_\pi{}^{(h+1)}(x')\Delta^{(h)}(x'|x^{(h)},\pi)\right].\end{aligned} \tag{6}$$

Here (i) follows from (4) (definition of $\alpha_\pi^{(h)}$); and (ii) follows from (2) and (5). Note that, in the final expression of (6), the first term can be estimated using stochastic CB techniques and the dependence on next-step value function is scaled by the size of this shift. This hints at the possibility of developing methods that interpolate between CB and RL techniques. More formally, in Section 4 , we show shift estimates enable us to adapt to the hardness of our setting – when estimating and constructing CIs for $\alpha_\pi^{(h)}$.

4 Theory: Selective Propagation

In Section 2, we motivated and defined our estimand $\alpha_\pi^{(h)}$ (see (4)) – which is the treatment effect for deviating from the behavioral policy at step h after having already deviated from the behavioral policy for all future steps. We now present an approach to estimate and construct tight valid CIs for $\alpha_\pi^{(h)}$ – with interval size adapting to instance hardness. Here harder instances have a larger next-state distribution shifts when deviating from the behavioral policy. When shifts are smaller, we can rely more on statistically efficient CB methods. However when shifts are larger (instance is more dynamic), we unavoidably must rely more on RL methods that account for worst-case distribution shifts.

Our approach to estimate and construct tight valid CIs for $\alpha_\pi^{(h)}$ requires several inputs. These inputs, described in the following subsection, allow us to abstract away existing approaches to tackle well-studied estimation problems in CB and RL settings. In Section 4.2, we describe how to combine these existing tools to achieve guarantees that adapt to instance hardness.

4.1 Inputs

Our method interpolates between existing tools for CB and RL settings, by leveraging shift estimates. To simplify our analysis and generalize our results, we assume access to these estimates as inputs to our interpolation method. In particular, we take as input: (1) offline CB treatment effect estimate and corresponding CI, (2) optimistic and pessimistic offline RL value function estimates, and (3) shift estimates with average error bounds. As the quality of our inputs improve (potentially as better estimators get developed), the quality of our outputs will correspondingly improve.

We now formally describe these inputs – requiring all the associated high-probability bounds to hold simultaneously with probability at least $1 - \delta_{\text{in}}$. We start by describing the first input, which is based on CB methods.

Input 1 (CB estimates): This input provides an estimate and CI for $\theta^{(h)}$ (formally defined in (7)) – which is the average treatment effect on the immediate reward for deviating from the behavioral policy at step h.

$$\theta_\pi^{(h)} = \mathbb{E}_{D(\pi_b)}\Big[R^{(h)}(x^{(h)},\tilde{\pi}_h) - R^{(h)}(x^{(h)},\tilde{\pi}_{h+1})\Big] \tag{7}$$

Since $\theta_\pi^{(h)}$ only depends on the immediate reward, well-established offline CB techniques (e.g., Dudik et al. 2014) can be used to estimate and construct CIs for the difference (in terms of immediate rewards) between these policies. We let $\hat{\theta}_\pi^{(h)}$ be our input estimate and let $\kappa_{\pi,\theta}^{(h)}$ be the input CI radius. That is, the confidence interval is given by (8).

$$|\theta_\pi^{(h)} - \hat{\theta}_\pi^{(h)}| \leq \kappa_{\pi,\theta}^{(h)} \tag{8}$$

When deviating from the behavioral policy at step h has no impact on next-state distributions, the estimate and CI for $\theta_\pi^{(h)}$ can be used as the estimate and CI for $\alpha_\pi^{(h)}$. However, when there is an impact on next-state distributions, valid estimation and CI construction for $\alpha_\pi^{(h)}$ requires us to propagate estimates and uncertainty from future steps to the current step. To enable this propagation, we take estimates for $V_\pi{}^{(h+1)}$ as our second input.

Input 2 (RL estimates): This input provides pessimistic, standard, and optimistic estimates for $V_\pi{}^{(h+1)}$ – denoted by $\hat{V}_{\pi,p}^{(h+1)}$, $\hat{V}_\pi^{(h+1)}$, and $\hat{V}_{\pi,o}^{(h+1)}$ respectively – such that the ordering in (9) holds.[2] Further, with high probability, we require (10) holds – that is, the true value function is bounded by the pessimistic and optimistic value function estimates.

$$\forall x,\ 0 \leq \hat{V}_{\pi,p}^{(h+1)}(x) \leq \hat{V}_\pi^{(h+1)}(x) \leq \hat{V}_{\pi,o}^{(h+1)}(x) \leq V_{\max}^{(h+1)} \tag{9}$$

$$\forall x,\ V_\pi{}^{(h+1)}(x) \in [\hat{V}_{\pi,p}^{(h+1)}(x), \hat{V}_{\pi,o}^{(h+1)}(x)] \tag{10}$$

There is a large and growing literature on value function estimation in RL, including optimistic and pessimistic value function estimation that are designed to satisfy (10) (e.g., Martin et al. 2017; Wang et al. 2019; Jin, Yang, and Wang 2021). Thus, we can employ the most cutting-edge methods to construct these next-step value function estimates.

Input 2 gave us estimates for $V_\pi{}^{(h+1)}$ (next-step value), which we may need to propagate to the current step h – when constructing an estimate and CI for $\alpha_\pi^{(h)}$. Since our goal is to interpolate between tight CB guarantees and always valid RL guarantees, unlike traditional RL algorithms, we want to be selective in propagating next-step estimates/uncertainties. Our final input, shift estimates, allows us to only propagate these estimates when required – enabling our adaptation to instance hardness.

Input 3 (Shift estimates): This input provides a estimate for $\Delta^{(h)}$ (see (5)) and an associated error bound – denoted

[2] Note that (9) can be enforced during input construction. Recall that $V_{\max}{}^{(h+1)}$ denotes a bound on the maximum value $V_\pi^{(h+1)}(x)$ can take for any state x.

by $\hat{\Delta}^{(h)}$ and $\kappa^{(h)}_{\pi,\Delta}$ respectively – such that (11) holds (recall from Section 3 that $\Delta^{(h)}$ satisfies the same bound). We also require (12) holds with high-probability.

$$\forall(x,a),\ \|\hat{\Delta}^{(h)}(\cdot|x,a)\|_1 \leq 2 \tag{11}$$

$$\mathbb{E}_{D(\pi_b)}\left\|\hat{\Delta}^{(h)}(\cdot|x^{(h)},\pi) - \Delta^{(h)}(\cdot|x^{(h)},\pi)\right\|_1 \leq \kappa^{(h)}_{\pi,\Delta} \tag{12}$$

Since the true shift model is a function of the true transition model (see Section 3), shift can be estimated via first estimating transition model and then calculating the treatment effect (due to deviating from the behavioral policy) of transitioning between any pair of states (see Moerland et al. 2023, for a survey on model-based RL and transition model estimation.). [3]

4.2 Combining Inputs

We now have all our required input estimates, and can state our main result (Theorem 4.1) on constructing an estimate and CI for $\alpha^{(h)}_\pi$ – in a way that adapts to instance hardness.

Theorem 4.1. *Suppose we have: (1) CB inputs* $(\hat{\theta}^{(h)}_\pi, \kappa^{(h)}_{\pi,\theta})$*; (2) RL inputs* $(\hat{V}^{(h+1)}_{\pi,p}, \hat{V}^{(h+1)}_{\pi}, \hat{V}^{(h+1)}_{\pi,o})$ *satisfying* (9)*; and (3) shift inputs* $(\hat{\Delta}^{(h)}, \kappa^{(h)}_{\pi,\Delta})$ *satisfying* (11) *– such that* (8)*,* (10)*, and* (12) *hold with probability at least* $1-\delta_{in}$*.*
Moreover, suppose we have a (holdout) dataset of T *trajectories* $S = \{\tau_t\}_{t=1}^T$ *– sampled from the distribution* $D(\pi_b)$ *– that were not used for constructing input estimates.*[4] *Our estimate for* $\alpha^{(h)}_\pi$ *is then denoted by* $\hat{\alpha}^{(h)}_\pi$ *and given by* (13)*.*

$$\hat{\alpha}^{(h)}_\pi = \hat{\theta}^{(h)}_\pi + \frac{1}{T}\sum_{t=1}^{T}\int_{x'} \hat{V}^{(h+1)}_\pi(x')\hat{\Delta}^{(h)}(x'|x^{(h)}_t,\pi) \tag{13}$$

Now for some fixed $\delta > 0$*, with probability at least* $1-\delta-\delta_{in}$*, we have the confidence interval in* (14) *holds.*

$$|\alpha^{(h)}_\pi - \hat{\alpha}^{(h)}_\pi| \leq L^{(h)}_\pi. \tag{14}$$

Here $L^{(h)}_\pi$ *is given by* (15)*.*

$$L^{(h)}_\pi = \kappa^{(h)}_{\pi,\theta} + {V_{\max}}^{(h+1)}\kappa^{(h)}_{\pi,\Delta} + 6V^{(h+1)}_{\max}\sqrt{\frac{\ln(4/\delta)}{2T}} + \frac{1}{T}\sum_{t=1}^{T}\int_{x'} |\mathbb{E}_{D(\pi_b)}[\hat{\Delta}^{(h)}(x'|x^{(h)}_t,\pi)]|{\Gamma_\pi}^{(h+1)}(x') \tag{15}$$

Here ${\Gamma_\pi}^{(h+1)}$ *is the difference between the optimistic and pessimistic estimates – it captures the uncertainty in the next-step value function estimates. That is, for all* $x \in \mathcal{X}$*,* ${\Gamma_\pi}^{(h+1)}(x) = \hat{V}^{(h+1)}_{\pi,o}(x) - \hat{V}^{(h+1)}_{\pi,p}(x)$*.*

[3] As a treatment effect model, shift may also be estimated via heterogeneous treatment effect estimators (e.g., Nie and Wager 2021; Künzel et al. 2019).

[4] Utilizing a holdout set for estimating and constructing CIs for $\alpha^{(h)}_\pi$ allows us to treat our input estimates as fixed quantities (independent of the randomness in the sampled holdout dataset).

One of the advantages of in-distribution supervised learning is that excess risk bounds only depend on complexity of hypothesis class (and number of training samples), with no dependence on size of feature space (see Shalev-Shwartz and Ben-David 2014). As discussed in Section 1, the statistical challenges of RL stem from the fact that learning under (state) distribution shifts is hard. For example, without additional assumptions, optimistic/pessimistic value function estimation have an unavoidable polynomial dependency on state-space size (Foster and Rakhlin 2023). Our goal is to avoid/minimize such dependencies when possible. The key benefit of Theorem 4.1 is that both our estimate $(\hat{\alpha}^{(h)}_\pi)$ and our CI width $(L^{(h)}_\pi)$ are "selective" in propagating/utilizing the RL estimates from input 2 – allowing us to only suffer from the slower worst-case RL estimation rates on harder instances. To better understand this, let us dig deeper into the terms in our CI width $(L^{(h)}_\pi)$.

Note that $\kappa^{(h)}_{\pi,\theta}$ and $\kappa^{(h)}_{\pi,\Delta}$ (see Inputs 1 and 3) bound errors averaged under the behavioral policy state-distribution – that is, they bound in-distribution average errors. Hence, with appropriate inputs, the first two terms in $L^{(h)}_\pi$ shrink quickly with no dependency on state-space size (Dudik et al. 2014; Shalev-Shwartz and Ben-David 2014). The third term in $L^{(h)}_\pi$, which enjoys a $\mathcal{O}(\sqrt{\log(1/\delta)/T})$ rate, also shrinks quickly to zero and has no dependency on state-space size.

We now only need to discuss the fourth and final term in $L^{(h)}_\pi$. Unlike the previous terms which bound in-distribution average errors, this term does depend on per-state (pointwise) errors $({\Gamma_\pi}^{(h+1)}(x'))$. The reason RL algorithms seek to bound per-state errors is because these guarantees do not depend on the state-distribution and are valid under any shift. We now illustrate how this robustness to state-distribution shift comes at a cost of larger error bounds, and argue that it is advantageous to scale these terms down with the estimated shift. First, as a sanity check, we show that this term is finite.

$$\int_{x'} |\hat{\Delta}^{(h)}(x'|{x_t}^{(h)},\pi)| \cdot {\Gamma_\pi}^{(h+1)}(x') \overset{(i)}{\leq} \|\hat{\Delta}^{(h)}(\cdot|{x_t}^{(h)},\pi)\|_1 \|{\Gamma_\pi}^{(h+1)}\|_\infty \overset{(ii)}{\leq} 2\|{\Gamma_\pi}^{(h+1)}\|_\infty. \tag{16}$$

where (i) follows from Hölder's inequality, and (ii) follows from (11). Now that we know this term is finite, we can argue that it shrinks to zero. Since ${\Gamma_\pi}^{(h+1)}(x')$ captures the size of per-state errors for pessimistic/optimistic value function estimates, we can expect this term to converges to zero in the limit with infinite data. The size of ${\Gamma_\pi}^{(h+1)}(x')$ must depend on how often states similar to x' were visited at step $h+1$ in the training data for the RL input. For simplicity, let us consider the scenario when all states are visited uniformly at step $h+1$ under the distribution $D(\pi_b)$. In such a scenario, the frequency at which states similar to x' were visited at step $h+1$ would depend on some measure of the size of the state space $\mathcal{X}$. This would imply that ${\Gamma_\pi}^{(h+1)}(x')$ shrinks at a rate that depends on some measure of the size of the state space $\mathcal{X}$. That is, while these terms shrink, they

shrink slowly. Hence per-state bounds, while independent of state-distribution, come at a cost of slower statistical rates. As shown in (Foster et al. 2021), such a dependence of confidence interval width on state-space size is unavoidable in the worst-case.

The key message of Theorem 4.1 is that we can move beyond this worst-case scenario by scaling these point-wise errors with the estimated shifts ($\hat{\Delta}^{(h)}$). For example, when $\hat{\Delta}^{(h)} \equiv 0$, the fourth term in ${L_\pi}^{(h)}$ is zero, allowing us to recover contextual bandit-style guarantees that are independent of state-space size. It is worth noting that, even when state-space size is not a concern, being selective about error/estimate propagation can improve the resulting interval widths.

5 Modifying Pessimistic Value Iteration

Pessimistic value iteration (PVI) (Jin, Yang, and Wang 2021) is a popular family of dynamic programming (DP) algorithms for offline RL. PVI can take as input: an estimated reward model $\hat{R}$, an estimated transition model $\hat{P}$, and pointwise estimation uncertainty bonus $b = (b^{(h)})_{h=1}^{H}$. No additional data is required. The DP procedure in PVI iterates over steps $h = H$ to $h = 1$. For any step of value function estimation, the bonuses $b = (b^{(h)})_{h=1}^{H}$ must bound the cumulative error in the estimated reward and transition model inputs. [5] Hence at any step h, (17) gives a valid pessimistic Q-value estimate ($\hat{Q}_{\text{p}}^{(h)}$) – here $\hat{V}_p^{(h+1)}$ is the pessimistic value function for the constructed policy starting from step $h+1$ (computed in the previous dynamic programming step). At every step h, PVI selects the policy that maximizes the pessimistic Q-value function.

$$\hat{Q}_{\text{p}}^{(h)}(\cdot,\cdot) = \hat{R}(\cdot,\cdot) + \int_{x'} \hat{V}_p^{(h+1)}(x')\hat{P}^{(h)}(x'|\cdot,\cdot) - b^{(h)}(\cdot,\cdot) \tag{17}$$

Pessimistic Q-value maximization helps PVI avoid model exploitation – that is, PVI avoids picking policies with inaccurately high estimated values at step h by penalizing uncertainty in the value function estimate. However to do this, as we see in (17), PVI propagates all the uncertainty captured in future steps through $\hat{V}_p^{(h+1)}$. Depending on the instance, this may lead to larger than necessary uncertainty propagation for avoiding model exploitation. In particular, based on the results in Section 4, uncertainty from later steps does not always need to be fully propagated to estimate lower bounds for the effect of deviating from the behavioral policy at step h (after fixing the policy for all future steps). Since we can view maximizing this treatment effect as the goal of DP algorithms at any step h, maximizing the tighter lower bounds from Section 4 should allow us to do better on easier CB-like instances (while avoiding model exploitation).

[5] In finite-state MDPs, these bonuses can be count based, where $b^{(h)}(x,a) = \beta \cdot \sqrt{1/\max\{1, n_h(x,a)\}}$ and $n_h(x,a)$ is the number of times state x and action a are observed at step h in the data set S. Here β is an algorithmic parameter. Several papers have extended these ideas to continuous state spaces (Bellemare et al. 2016; Osband et al. 2021).

In this section, we propose a modification of PVI called selectively pessimistic value iteration (SPVI, complete pseudo-code is available in Appendix B). The key modification is that, at any step h, SPVI maximizes the selectively pessimistic Q-value (18) – which is the standard Q-value estimate ($\hat{Q}^{(h)}$) minus the bonus and the required uncertainty that needs to be propagated. Here $\hat{\Delta}$ is the induced shift model (that is, $\hat{\Delta}^{(h)}(\cdot|\cdot,\cdot) = \hat{P}^{(h)}(\cdot|\cdot,\cdot) - \hat{P}^{(h)}(\cdot|\cdot,\pi_b)$), $\hat{V}_p^{(h+1)}$ and $\hat{V}_o^{(h+1)}$ are the pessimistic and optimistic value function for the constructed policy starting from step $h+1$ (computed in the previous dynamic programming step).

$$\begin{aligned}\hat{Q}_{\text{sp}}^{(h)}(\cdot,\cdot) &= \hat{Q}^{(h)}(\cdot,\cdot) - b^{(h)}(\cdot,\cdot) \\ &- \int_{x'} |\hat{\Delta}^{(h)}(x'|\cdot,\cdot)| \cdot (\hat{V}_o^{(h+1)}(x') - \hat{V}_p^{(h+1)}(x'))\end{aligned} \tag{18}$$

Justification: As we argued in earlier sections, one can view the goal at step h as selecting $\pi^{(h)}$ in order to maximize $\alpha_\pi^{(h)}$. We construct a tight lower bound for $\alpha_\pi^{(h)}$ and argue that maximizing $\hat{Q}_{\text{sp}}^{(h)}$ maximizes this bound. Similar to standard pessimistic value estimation, we let the bonus $b^{(h)}(x,a)$ bound the total model estimation errors at (x,a,h). Now from the analysis in Theorem 4.1, we have (19) is a valid lowed bound on $\alpha_\pi^{(h)}$.

$$\begin{aligned}\alpha_\pi^{(h)} &= \mathbb{E}_{D(\pi_b)}[V_{\tilde{\pi}_h}^{(h)}(x^{(h)}) - V_{\tilde{\pi}_{h+1}}^{(h)}(x^{(h)})] \\ &\geq \mathbb{E}_{D(\pi_b)}[\hat{Q}_{\text{sp}}^{(h)}(x^{(h)},\pi^{(h)}) - \hat{Q}^{(h)}(x^{(h)},\pi_b{}^{(h)})]\end{aligned} \tag{19}$$

Importantly, by maximizing this tight lower bound we can do better on easier CB-like instances (while always avoiding model exploitation by penalizing uncertainty in estimating $\alpha_\pi^{(h)}$). This completes our justification of SPVI. The complete pseudo-code is available in Appendix B.

6 Simulation

To illustrate our insights, we consider a simple toy environment called "ChainBandit". At a high-level, the environment has two chains, a top chain, and a bottom chain. The environment also has three actions given by a_1, a_2, a_3. The top chain states are the most rewarding. The agent starts at $(1,0)$ (first state in the top chain). At any state in the bottom chain, all the actions lead to the same transition (which is to move to the next state in the bottom chain) and are essentially bandit states. In the top chain, both a_1 and a_2 lead to the same transitions (which is to move to the next state in the top chain), and a_3 makes the agent move to the next state in the bottom chain. In the top chain, the highest cumulative reward comes from never taking action a_3; however, the highest immediate reward comes from selecting the action a_3 (which makes planning beneficial in this environment). Note that at every state, action a_3 is a sub-optimal action. See Appendix C for a complete description.

The ChainBandit environment is constructed to have both dynamic (some actions result in a different next-state distribution) and bandit-like (non-dynamic) elements. Ensuring that while planning and some estimate/uncertainty propagation is necessary, complete uncertainty propagation can be

unnecessary to evaluate policies of interest. Throughout this section, we consider ChainBandit with a chain length of 3 and consider the following behavioral policy (π_b) – at every state and step, π_b selects action a_3 with probability 0.8 and selects the other two actions with probability 0.1 respectively. From the data collected, we evaluate standard and selective uncertainty propagation for tasks of: (1) estimating upper/lower bounds for $\alpha_\pi^{(h)}$; and (2) offline policy learning.

Since uncertainty propagation is the focus of this simulation, in order to make a fair comparison, both standard and selective propagation: use the same tabular approach to estimate a (step independent) reward/transition model; and use the same (step independent) count-based bonuses to account for model estimation errors.[6] Here bonus is given by $b(x,a) = \sqrt{\ln(|\mathcal{X}| * |\mathcal{A}| * H/\delta)/n(x,a)}$ – where $n(x,a)$ is the number of times action a was taken at state x, and confidence parameter $\delta = 0.05$.

For the step h and policy π of interest, standard CIs for $\alpha_\pi^{(h)}$ are constructed using pessimistic/optimistic value estimates at step h for policies $\tilde{\pi}_h$ and $\tilde{\pi}_{h+1}$ – i.e, utilizing (20).

$$\begin{aligned}&\mathbb{E}_{D(\pi_b)}[\hat{V}^{(h)}_{\tilde{\pi}_h,o}(x^{(h)}) - \hat{V}^{(h)}_{\tilde{\pi}_{h+1},p}(x^{(h)})] \geq \alpha_\pi^{(h)} \\ &\geq \mathbb{E}_{D(\pi_b)}[\hat{V}^{(h)}_{\tilde{\pi}_h,p}(x^{(h)}) - \hat{V}^{(h)}_{\tilde{\pi}_{h+1},o}(x^{(h)})]\end{aligned} \tag{20}$$

For selective CIs, we use (19) to construct the lower bound and similarly construct the upper bound. Note that converting inequalities like (19) and (20) into empirical bounds is straightforward since our training is from $D(\pi_b)$. Figure 1 plots CIs for both selective and standard uncertainty propagation, when varying the evaluation policy. As expected, benefits over standard pessimism are larger when next-state distribution shift is smaller – that is, when evaluation policy is closer to the behavioral policy.

In Figure 2, we plot the value of learnt policy from various algorithms as we vary the number of training episodes. In particular, we compare SPVI, PVI (Jin, Yang, and Wang 2021), and pessimistic supervised learning (PSL). Here PSL refers to a pessimistic bandit policy optimization applied to each step without planning. The ChainBandit environment benefits from planning, so PSL performs poorly as expected.

On all Chain Bandit simulations we tried, SPVI was by far the best-performing algorithm. The reason we considered the behavioral policy described earlier was that it was more disadvantageous for SPVI. In particular, since selective pessimism has an initial bias against policies that lead to significant shifts, we chose a highly sub-optimal behavioral policy (selecting a_3 with probability 0.8). While this leads to a worse start for SPVI than PVI, eventually, SPVI outperforms the other algorithms. We also run simulations for CI construction and policy learning on the standard GridWorld (see Appendix D) – since this is a very dynamic environment, both standard and selective propagation have similar performance.[7]

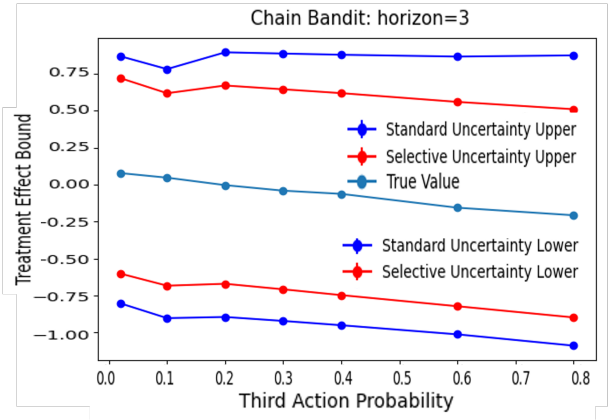

Figure 1: We plot CIs for $\alpha_\pi^{(2)}$ while varying the evaluation policy. These evaluation policies are parameterized by $\lambda \in [0,1]$. For all states/steps, the probability of selecting a_1, a_2 and a_3 are $(1-\lambda)/2$, $(1-\lambda)/2$, and λ respectively. Note that the evaluation policy is the same as the behavioral policy for $\lambda = 0.8$. The number of training episodes is 10000, and the plots are averaged over 10 runs.

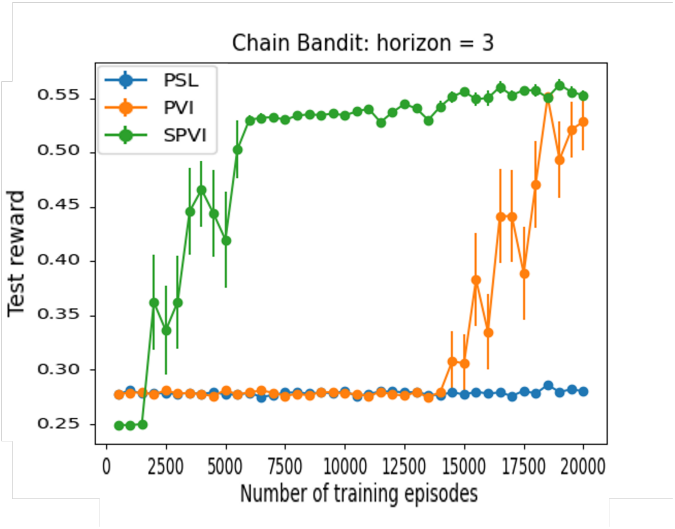

Figure 2: Policy learning with a bad behavioral policy

7 Conclusion

We introduce selective propagation, a general approach to interpolate between CB and RL techniques – achieving guarantees that adapt to instance hardness. Further developing this could impact real world problems (e.g., recommendation systems, mHealth, EdTech) that lie in between the two settings.

[6]The model estimates and bonuses are not step dependent since the reward/transition functions in Chain Bandit are the same for all steps. Further a tabular approach to reward (transition) estimation simply indicates using the average reward (average one-hot next-state vector) observed at any state-action pair as its reward (transition) estimate.

[7]All algorithm runs takes less than 2 mins on a MacBook Pro M2 16GB.

References

Abbasi-Yadkori, Y.; Pal, D.; and Szepesvari, C. 2011. Improved Algorithms for Linear Stochastic Bandits. In *Advances in Neural Information Processing Systems 24*, 2312–2320.

Agarwal, A.; Hsu, D.; Kale, S.; Langford, J.; Li, L.; and Schapire, R. 2014. Taming the Monster: A Fast and Simple Algorithm for Contextual Bandits. In *Proceedings of the 31st International Conference on Machine Learning*, 1638–1646.

Agrawal, S.; and Goyal, N. 2012. Analysis of Thompson Sampling for the Multi-Armed Bandit Problem. In *Proceeding of the 25th Annual Conference on Learning Theory*, 39.1–39.26.

Agrawal, S.; and Goyal, N. 2013. Thompson Sampling for Contextual Bandits with Linear Payoffs. In *Proceedings of the 30th International Conference on Machine Learning*, 127–135.

Bellemare, M.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; and Munos, R. 2016. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29.

Bobu, A.; Tzeng, E.; Hoffman, J.; and Darrell, T. 2018. Adapting to continuously shifting domains. *workshop*.

Carranza, A. G.; Krishnamurthy, S. K.; and Athey, S. 2023. Flexible and efficient contextual bandits with heterogeneous treatment effect oracles. In *International Conference on Artificial Intelligence and Statistics*, 7190–7212. PMLR.

Chapelle, O.; and Li, L. 2012. An Empirical Evaluation of Thompson Sampling. In *Advances in Neural Information Processing Systems 24*, 2249–2257.

Dani, V.; Hayes, T.; and Kakade, S. 2008. Stochastic Linear Optimization under Bandit Feedback. In *Proceedings of the 21st Annual Conference on Learning Theory*, 355–366.

Dudik, M.; Erhan, D.; Langford, J.; and Li, L. 2014. Doubly Robust Policy Evaluation and Optimization. *Statistical Science*, 29(4): 485–511.

Farshchian, A.; Gallego, J. A.; Cohen, J. P.; Bengio, Y.; Miller, L. E.; and Solla, S. A. 2018. Adversarial domain adaptation for stable brain-machine interfaces. *arXiv preprint arXiv:1810.00045*.

Foster, D. J.; Krishnamurthy, A.; Simchi-Levi, D.; and Xu, Y. 2021. Offline Reinforcement Learning: Fundamental Barriers for Value Function Approximation. *arXiv preprint arXiv:2111.10919*.

Foster, D. J.; and Rakhlin, A. 2023. Foundations of Reinforcement Learning and Interactive Decision Making. *arXiv preprint arXiv:2312.16730*.

Foster, D. J.; and Syrgkanis, V. 2019. Orthogonal statistical learning. *arXiv preprint arXiv:1901.09036*.

Garivier, A.; and Cappe, O. 2011. The KL-UCB Algorithm for Bounded Stochastic Bandits and Beyond. In *Proceeding of the 24th Annual Conference on Learning Theory*, 359–376.

Jin, Y.; Yang, Z.; and Wang, Z. 2021. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, 5084–5096. PMLR.

Krishnamurthy, A.; Wu, Z. S.; and Syrgkanis, V. 2018. Semiparametric contextual bandits. In *International Conference on Machine Learning*, 2776–2785. PMLR.

Künzel, S. R.; Sekhon, J. S.; Bickel, P. J.; and Yu, B. 2019. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the national academy of sciences*, 116(10): 4156–4165.

Langford, J.; and Zhang, T. 2008. The Epoch-Greedy Algorithm for Contextual Multi-Armed Bandits. In *Advances in Neural Information Processing Systems 20*, 817–824.

Lattimore, T.; and Szepesvari, C. 2019. *Bandit Algorithms*. Cambridge University Press.

Levine, S.; Kumar, A.; Tucker, G.; and Fu, J. 2020. Offline Reinforcement Learning: Tutorial, Review. *and Perspectives on Open Problems*.

Li, L.; Chu, W.; Langford, J.; and Schapire, R. 2010. A Contextual-Bandit Approach to Personalized News Article Recommendation. In *Proceedings of the 19th International Conference on World Wide Web*.

Martin, J.; Sasikumar, S. N.; Everitt, T.; and Hutter, M. 2017. Count-based exploration in feature space for reinforcement learning. *arXiv preprint arXiv:1706.08090*.

Moerland, T. M.; Broekens, J.; Plaat, A.; Jonker, C. M.; et al. 2023. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1): 1–118.

Nie, X.; and Wager, S. 2021. Quasi-oracle estimation of heterogeneous treatment effects. *Biometrika*, 108(2): 299–319.

Osband, I.; Wen, Z.; Asghari, S. M.; Dwaracherla, V.; Ibrahimi, M.; Lu, X.; and Van Roy, B. 2021. Epistemic neural networks. *arXiv preprint arXiv:2107.08924*.

Russo, D.; Van Roy, B.; Kazerouni, A.; Osband, I.; and Wen, Z. 2018. A Tutorial on Thompson Sampling. *Foundations and Trends in Machine Learning*, 11(1): 1–96.

Shalev-Shwartz, S.; and Ben-David, S. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Sutton, R. 1988. Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 3: 9–44.

Sutton, R.; and Barto, A. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Sutton, R.; McAllester, D.; Singh, S.; and Mansour, Y. 2000. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems 12*, 1057–1063.

Syrgkanis, V.; and Zhan, R. 2023. Post-Episodic Reinforcement Learning Inference. *arXiv preprint arXiv:2302.08854*.

Thompson, W. R. 1933. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25(3-4): 285–294.

Vergara, A.; Vembu, S.; Ayhan, T.; Ryan, M. A.; Homer, M. L.; and Huerta, R. 2012. Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, 166: 320–329.

Wang, Y.; Wang, R.; Du, S. S.; and Krishnamurthy, A. 2019. Optimism in reinforcement learning with generalized linear function approximation. *arXiv preprint arXiv:1912.04136*.

Williams, R. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3-4): 229–256.

Yin, M.; and Wang, Y.-X. 2021. Towards instance-optimal offline reinforcement learning with pessimism. *Advances in neural information processing systems*, 34: 4065–4078.

Zanette, A.; and Brunskill, E. 2019. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. In *International Conference on Machine Learning*, 7304–7312. PMLR.

Zhang, K. W.; Gottesman, O.; and Doshi-Velez, F. 2022. A Bayesian Approach to Learning Bandit Structure in Markov Decision Processes. *arXiv preprint arXiv:2208.00250*.