WHEN SHOULD OFFLINE REINFORCEMENT LEARNING BE PREFERRED OVER BEHAVIORAL CLONING?

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline reinforcement learning (RL) algorithms can acquire effective policies by utilizing only previously collected experience, without any online interaction. While it is widely understood that offline RL is able to extract good policies even from highly suboptimal data, in practice offline RL is often used with data that resembles demonstrations. In this case, one can also use behavioral cloning (BC) algorithms, which mimic a subset of the dataset via supervised learning. It seems natural to ask: *under what environment and dataset conditions can an offline RL method outperform BC with an equal amount of expert data, even when BC is a natural choice?* To answer this question, we characterize the properties of environments that allow offline RL methods to perform better than BC methods even when only provided with expert data. Additionally, we show that policies trained on suboptimal data that is sufficiently noisy can attain better performance than even BC algorithms with expert data, especially on long-horizon problems. We validate our theoretical results via extensive experiments on both diagnostic and high-dimensional domains including robot manipulation, maze navigation and Atari games, when learning from a variety of data sources. We observe that modern offline RL methods trained on suboptimal, noisy data in sparse reward domains outperform cloning the expert data in several practical problems.

1 INTRODUCTION

Offline reinforcement learning (RL) algorithms aim to leverage large, existing datasets of previously collected data to produce effective policies that generalize across a wide range of scenarios, without the need for costly active data collection. Many recent offline RL algorithms (Fujimoto et al., 2018; Kumar et al., 2019; Wu et al., 2019; Kumar et al., 2020; Yu et al., 2020; Sinha et al., 2021; Kostrikov et al., 2021) can work well even highly suboptimal data. With recent advances, the performance of offline RL algorithms has improved significantly, and a number of these approaches have been studied theoretically (Wang et al., 2021; Zanette, 2020; Rashidinejad et al., 2021). While it is clear that offline RL algorithms are a good choice when the available data is either random or highly suboptimal, it is less clear if these methods should even be tried when the datasets consist of demonstration that come from expert or near-expert demonstrations. In these cases, imitation learning algorithms, such as behavorial cloning (BC), can be used to train policies via supervised learning. It then seems natural to ask: *When should we prefer to use offline RL over imitation learning?*

To our knowledge, there has not been a rigorous characterization of when offline RL perform better than imitation learning. Existing empirical studies comparing offline RL to imitation learning have mixed conclusions. Some works show that offline RL methods appear to greatly outperform imitation learning, specifically in environments that require "stitching" parts of suboptimal trajectories (Fu et al., 2020). In contrast, a number of recent works have argued that BC performs better than offline RL on both expert and suboptimal demonstration data over a variety of tasks (Mandlekar et al., 2021; Florence et al., 2021; Hahn et al., 2021). This makes it confusing for practitioners to understand whether to use offline RL or simply run BC on collected demonstrations. Thus, in this work we aim to understand if there are conditions on the environment or the dataset under which an offline RL algorithm might outperform BC for a given task, even when BC is provided with expert data or is allowed to use reward as side information. Our insights can inform a practitioner with sufficient domain knowledge on whether offline RL is a good idea, even when expert or near-expert data is available and BC is considered to be a natural choice.

Our contribution in this paper is a theoretical and empirical characterization of certain conditions when offline RL can outperform BC. Theoretically, we are the first to derive conditions on the environment or offline dataset where offline RL achieves better worst-case guarantees than even the *best-case lower-bound* for BC using *expert demonstrations*. These conditions are grounded in practical problems, and provide guidance to the practitioner as to whether they should use RL or BC. Concretely, we show that in the case of expert data, the error incurred by offline RL algorithms can scale significantly more favorably when the MDP enjoys some structure including horizon-independent returns (*i.e.*, sparse rewards), or a low volume of states where it is "critical" to take the same action as the expert (Section 4.2) Meanwhile, in the case of sufficiently noisy data, we show that offline RL again enjoys better guarantees on long-horizon tasks (Section 4.3). Finally, since traditional BC methods ignore rewards, we consider generalized BC methods that use the observed rewards to inform learning, and show that it is still preferable to perform offline RL (Section 4.4).

Empirically, we validate our theoretical conclusions on diagnostic gridworld domains (Fu et al., 2019) and large-scale benchmark problems in robotic manipulation and navigation and Atari games, using human data (Fu et al., 2020), scripted data (Singh et al., 2020) and data generated from RL policies (Agarwal et al., 2020b). We verify that in multiple long-horizon problems where the conditions we propose are likely to be satisfied, practical offline RL methods can outperform BC and generalized BC methods. Using careful offline tuning practices, we show that it is possible for offline RL to outperform cloning an expert dataset for the same task, given equal amounts of data.

2 Related Work

Offline RL (Lange et al., 2012; Levine et al., 2020) has shown promise in domains such as robotic manipulation (Kalashnikov et al., 2018b; Mandlekar et al., 2020; Singh et al., 2020; Kalashnikov et al., 2021), NLP (Jaques et al., 2020) and healthcare (Shortreed et al., 2011; Wang et al., 2018). The major challenge in offline RL is distribution shift (Fujimoto et al., 2018; Kumar et al., 2019), where the learned policy might execute out-of-distribution actions. Prior offline RL methods can broadly be characterized into two categories: **(1)** *policy-constraint* methods that regularize the learned policy to be "close" to the behavior policy either explicitly (Fujimoto et al., 2018; Kumar et al., 2019; Liu et al., 2020; Wu et al., 2019; Fujimoto & Gu, 2021) or implicitly (Siegel et al., 2020; Peng et al., 2019; Nair et al., 2020), or via importance sampling (Liu et al., 2019; Swaminathan & Joachims, 2015; Nachum et al., 2019), and **(2)** *conservative* methods that learn a lower-bound, or conservative, estimate of return and optimize the policy against it (Kumar et al., 2020; Kostrikov et al., 2021; Kidambi et al., 2020; Yu et al., 2020; 2021). Our goal is not to devise a new offline RL algorithm, but rather to understand when existing offline RL methods can outperform BC.

When do offline RL methods outperform BC? Rashidinejad et al. (2021) derive a conservative offline RL algorithm based on lower-confidence bounds (LCB) that provably outperforms BC in the simpler contextual bandits (CB) setting, but do not extend it to MDPs. While this CB result signals the possibility that offline RL can outperform BC in theory, this generalization is not trivial, as RL suffers from compounding errors (Munos, 2003; 2005; Wang et al., 2021). Laroche et al. (2019); Nadjahi et al. (2019); Kumar et al. (2020); Liu et al. (2020); Xie et al. (2021a) present safe policy improvement bounds expressed as improvements over the behavior policy, which imitation aims to recover, but these bounds do not clearly indicate when offline RL is better or worse. Empirically, Fu et al. (2020) show that offline RL considerably outperforms BC for tasks that require "stitching" trajectory segments to devise an optimal policy. In contrast, Mandlekar et al. (2021); Brandfonbrener et al. (2021); Chen et al. (2021); Hahn et al. (2021) suggest that BC or filtered BC using the top fraction of the data performs better on other tasks. While results on Adroit domains in D4RL (Fu et al., 2020) show that offline RL outperforms BC even on expert data, Florence et al. (2021) reported superior BC results, making it unclear. Kurenkov & Kolesnikov (2022) emphasize the importance of the online evaluation budget for offline RL methods and show that BC is more favorable in a limited budget. We provide a characterization of certain scenarios where we would expect offline RL to be better than BC, and empirical results verifying that offline RL performs well on such problems, spanning robotics, navigation, games (Fu et al., 2020; Singh et al., 2020; Bellemare et al., 2013).

Our theoretical analysis combines tools from a number of prior works. We analyze the total error incurred by RL via an error propagation analysis (Munos, 2003; 2005; Farahmand et al., 2010; Chen & Jiang, 2019; Xie & Jiang, 2020; Liu et al., 2020), which gives rise to bounds with *concentrability coefficients* that bound the total distributional shift between the learned policy and the data distribution (Xie & Jiang, 2020; Liu et al., 2020). We use tools from Ren et al. (2021), which provide

horizon-free bounds for standard (non-conservative) offline Q-learning but relax their strict coverage assumptions. While our analysis studies a LCB-style algorithm similar to Rashidinejad et al. (2021), we modify it to use tighter Bernstein bonuses (Zhang et al., 2021; Agarwal et al., 2020a), which is key to improving its suboptimality guarantee. Xie et al. (2021b) consider a similar algorithm with Bernstein bonuses, but take a different approach to analysis it and use it for policy finetuning.

3 Problem Setup and Preliminaries

The goal in reinforcement learning is to learn a policy $\pi(\cdot|\mathbf{s})$ that maximizes the expected cumulative discounted reward in a Markov decision process (MDP), which is defined by a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$. $\mathcal{S}, \mathcal{A}$ represent state and action spaces, $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ and $r(\mathbf{s}, \mathbf{a})$ represent the dynamics and mean reward function, and $\gamma \in (0, 1)$ represents the discount factor. The effective horizon of the MDP is given by $H = 1/(1-\gamma)$. The Q-function, $Q^\pi(\mathbf{s}, \mathbf{a})$ for a given policy π is equal to the discounted long-term reward attained by executing $\mathbf{a}$ at the state $\mathbf{s}$ and then following policy π thereafter. Q^π satisfies the recursion: $\forall \mathbf{s}, \mathbf{a} \in \mathcal{S} \times \mathcal{A}$, $Q^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim P(\cdot|\mathbf{s},\mathbf{a}), \mathbf{a}' \sim \pi(\cdot|\mathbf{s}')} [Q(\mathbf{s}', \mathbf{a}')]$. The value function V^π considers the expectation of the Q-function over the policy $V^\pi(\mathbf{s}) = \mathbb{E}_{\mathbf{a} \sim \pi(\cdot|\mathbf{s})} [Q^\pi(\mathbf{s}, \mathbf{a})]$. Meanwhile, the Q-function of the optimal policy, Q^*, satisfies the recursion: $Q^*(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \mathbb{E}_{\mathbf{s}' \sim P(\cdot|\mathbf{s},\mathbf{a})} [\max_{\mathbf{a}'} Q^*(\mathbf{s}', \mathbf{a}')]$, and the optimal value function is given by $V^*(\mathbf{s}) = \max_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a})$. Finally, the expected cumulative discounted reward is given by $J(\pi) = \mathbb{E}_{\mathbf{s}_0 \sim \rho} [V^\pi(\mathbf{s}_0)]$.

In offline RL, we are provided with a dataset $\mathcal{D}$ of transitions, $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}_i')\}_{i=1}^N$ of size $|\mathcal{D}| = N$. We assume that the dataset $\mathcal{D}$ is generated i.i.d. from a distribution $\mu(\mathbf{s}, \mathbf{a})$ that specifies the effective behavior policy $\pi_\beta(\mathbf{a}|\mathbf{s}) := \mu(\mathbf{s}, \mathbf{a}) / \sum_{\mathbf{a}} \mu(\mathbf{s}, \mathbf{a})$. Note that holds even if the data itself is generated by running a non-Markovian policy π_β (Puterman, 1994). Let $n(\mathbf{s}, \mathbf{a})$ be the number of times $(\mathbf{s}, \mathbf{a})$ appear in $\mathcal{D}$, and $\widehat{P}(\cdot|\mathbf{s}, \mathbf{a})$ and $\widehat{r}(\mathbf{s}, \mathbf{a})$ denote the empirical dynamics and reward distributions in $\mathcal{D}$, which may be different from P and r due to stochasticity. Following Rashidinejad et al. (2021), the goal is to minimize the suboptimality of the learned policy $\widehat{\pi}$:

$$\mathsf{SubOpt}(\widehat{\pi}) = \mathbb{E}_{\mathcal{D} \sim \mu} [J(\pi^*) - J(\widehat{\pi})] = \mathbb{E}_{\mathcal{D}} \left[\mathbb{E}_{\mathbf{s}_0 \sim \rho} \left[V^*(\mathbf{s}_0) - V^{\widehat{\pi}}(\mathbf{s}_0) \right] \right]. \tag{1}$$

Dataset and MDP conditions. Here we introduce some conditions on the offline dataset and MDP structure that we make for our analysis. The first characterizes the distribution shift between the data distribution $\mu(\mathbf{s}, \mathbf{a})$ and the normalized state-action marginal of π^*, given by $d^*(\mathbf{s}, \mathbf{a}) = (1-\gamma) \sum_{t=0}^\infty \gamma^t \mathbb{P}(\mathbf{s}_t = s, \mathbf{a}_t = a; \pi^*)$, via a *concentrability coefficient* C^*.

Condition 3.1 (Rashidinejad et al. (2021), Concentrability of the data distribution)**.** *Define C^* to be the smallest, finite constant that satisfies: $d^*(\mathbf{s}, \mathbf{a}) / \mu(\mathbf{s}, \mathbf{a}) \leq C^* \;\; \forall \mathbf{s} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}$.*

Intuitively, the coefficient C^* scales with how suboptimal the data $\mu(\mathbf{s}, \mathbf{a})$ is relative to the optimal π^*, where $C^* = 1$ corresponds to data from π^*. The next condition we consider is that the discounted return for any trajectory in the MDP is bounded by a constant, which w.l.o.g., we assume to be 1.

Condition 3.2 (Ren et al. (2021), the value of any trajectory is bounded by 1)**.** *The infinite-horizon discounted return for any trajectory $\tau = (\mathbf{s}_0, \mathbf{a}_0, r_0, \mathbf{s}_1, \cdots)$ is bounded as $\sum_{t=0}^\infty \gamma^t r_t \leq 1$.*

This condition holds in sparse-reward tasks, particularly those where an agent succeeds or fails at its task once per episode. This is common in domains such as robotics (Singh et al., 2020; Kalashnikov et al., 2018b) and games (Bellemare et al., 2013), where the agent receives a signal upon succeeding a task or winning. This condition also appears in prior work deriving suboptimality bounds for RL algorithms (Ren et al., 2021; Zhang et al., 2021).

Notation. Let $n \wedge 1 = \max\{n, 1\}$. Denote $\iota = \mathrm{polylog}(|\mathcal{S}|, H, N)$. We let ι be a polylogarithmic quantity, changing with context. For d-dimensional vectors $\mathbf{x}, \mathbf{y}$, $\mathbf{x}(i)$ denotes its i-th entry, and define $\mathbb{V}(\mathbf{x}, \mathbf{y}) = \sum_i \mathbf{x}(i)\mathbf{y}(i)^2 - (\sum_i \mathbf{x}(i)\mathbf{y}(i))^2$.

4 Theoretical Comparison of BC and Offline RL

In this section, we present performance guarantees for BC and offline RL, and characterize scenarios where offline RL algorithms will outperform BC. We first present general upper bounds for both algorithms in Section 4.1, by extending prior work to account for the conditions discussed in Section 3. Then, we compare the performance of BC and RL when provided with the same data generated by an expert in Section 4.2 and when RL is given noisy, suboptimal data in Section 4.3. Our goal is to characterize the conditions on the environment and offline dataset where RL can outperform BC. Furthermore, we provide intuition for when they are likely to hold in Appendix D.

4.1 General Performance Guarantees of BC and Offline RL

Our goal is to understand if there exist offline RL methods can outperform BC for a given task. As our aim is simply to provide a proof of existence, we analyze representative offline RL and BC algorithms that achieve optimal suboptimality guarantees. For brevity, we only consider a conservative offline RL algorithm (as defined in Section 2) in the main paper and defer analysis of a policy-constraint method to Appendix C. Both algorithms are detailed in Algorithms 1 and 2.

Guarantees for BC. For analysis purposes, we consider a BC algorithm that matches the empirical behavior policy on states in the offline dataset, and takes uniform random actions outside the support of the dataset. This BC algorithm was also used in prior work (Rajaraman et al., 2020), and is no worse than other schemes for acting at out-of-support states, in general. Denote the learned BC policy as $\widehat{\pi}_\beta$, then $\forall \mathbf{s} \in \mathcal{D}, \widehat{\pi}_\beta(\mathbf{a}|\mathbf{s}) \leftarrow n(\mathbf{s},\mathbf{a})/n(\mathbf{s})$, and $\forall \mathbf{s} \notin \mathcal{D}, \widehat{\pi}_\beta(\mathbf{a}|\mathbf{s}) \leftarrow 1/|\mathcal{A}|$. We adapt the results presented in Rajaraman et al. (2020) to the setting with Conditions 3.1 and 3.2. BC can only incur a non-zero asymptotic suboptimality (*i.e.*, does not decrease to 0 as $N \to \infty$) in scenarios where $C^* = 1$, as it aims to match the data distribution $\mu(\mathbf{s},\mathbf{a})$, and a non-expert dataset will inhibit the cloned policy from matching the expert π^*. The performance for BC is bounded in Theorem 4.1.

Theorem 4.1 (Performance of BC)**.** *Under Conditions 3.1 and 3.2, the suboptimality of BC satisfies*

$$\mathsf{SubOpt}(\widehat{\pi}_\beta) \lesssim \frac{(C^*-1)H}{2} + \frac{|\mathcal{S}|H\iota}{N}.$$

A proof of Theorem 4.1 is presented in Appendix B.1. The first term is the additional suboptimality incurred due to discrepancy between the behavior and optimal policies. The second term in this bound is derived by bounding the expected visitation frequency of the learned policy $\widehat{\pi}_\beta$ onto states not observed in the dataset. The analysis is similar to that for existing bounds for imitation learning (Ross & Bagnell, 2010; Rajaraman et al., 2020). We achieve $\tilde{\mathcal{O}}(H)$ suboptimality rather than $\tilde{\mathcal{O}}(H^2)$ due to Condition 3.2, since the worst-case suboptimality of any trajectory is 1 rather than H.

Guarantees for conservative offline RL. We consider guarantees for a class of offline RL algorithms that maintain conservative value estimator such that the estimated value lower-bounds the true one, *i.e.*, $\widehat{V}^\pi \leq V^\pi$ for policy π. Existing offline RL algorithms achieve this by subtracting a penalty from reward either explicitly (Yu et al., 2020; Kidambi et al., 2020) or implicitly (Kumar et al., 2020). We only analyze one such algorithm that does the former, but we believe the algorithm can serve as a theoretical model for general conservative offline RL algorithms, where analyzing similar algorithms can be accomplished using the same outline. The algorithm we consider is similar in spirit to `VI-LCB` proposed by Rashidinejad et al. (2021) that subtracts penalty $b(\mathbf{s},\mathbf{a})$ from the reward during value iteration; we consider a different penalty that results in a tighter bound. The estimated Q-values are obtained by iteratively solving the following Bellman backup: $\widehat{Q}(\mathbf{s},\mathbf{a}) \leftarrow \widehat{r}(\mathbf{s},\mathbf{a}) - b(\mathbf{s},\mathbf{a}) + \sum_{\mathbf{s}'} \widehat{P}(\mathbf{s}'|\mathbf{s},\mathbf{a}) \max_{\mathbf{a}'} \widehat{Q}(\mathbf{s}',\mathbf{a}')$. The learned policy is then given by $\widehat{\pi}^*(\mathbf{s}) \leftarrow \arg\max_{\mathbf{a}} \widehat{Q}(\mathbf{s},\mathbf{a})$. Building on work in online RL (Zhang et al., 2021), our specific $b(\mathbf{s},\mathbf{a})$ is derived using Bernstein's inequality, and is shown below:

$$b(\mathbf{s},\mathbf{a}) \leftarrow \sqrt{\frac{\mathbb{V}(\widehat{P}(\cdot|\mathbf{s},\mathbf{a}),\widehat{V})\iota}{(n(\mathbf{s},\mathbf{a})\wedge 1)}} + \sqrt{\frac{\widehat{r}(\mathbf{s},\mathbf{a})\iota}{(n(\mathbf{s},\mathbf{a})\wedge 1)}} + \frac{\iota}{(n(\mathbf{s},\mathbf{a})\wedge 1)}.$$

The performance of the learned policy $\widehat{\pi}^*$ can then be bounded as:

Theorem 4.2 (Performance of conservative offline RL)**.** *Under Conditions 3.1 and 3.2, the policy $\widehat{\pi}^*$ found by conservative offline RL algorithm can satisfy*

$$\mathsf{SubOpt}(\widehat{\pi}^*) \lesssim \sqrt{\frac{C^*|\mathcal{S}|H\iota}{N}} + \frac{C^*|\mathcal{S}|H\iota}{N}.$$

We defer a proof for Theorem 4.2 to Appendix B.2. On a high level, we first show that our algorithm is always conservative, *i.e.*, $\forall \mathbf{s}$, $\widehat{V}(\mathbf{s}) \leq V^{\widehat{\pi}^*}(\mathbf{s})$, and then bound the total suboptimality incurred as a result of being conservative. **Novelty**: Our bound in in Theorem 4.2 improves on existing bounds several ways: **(1)** by considering pessimistic value estimates, we are able to remove the strict coverage assumptions used by Ren et al. (2021), **(2)** by using variance recursion techniques on the Bernstein bonuses, with Condition 3.2, we save a $O(H^2)$ factor over `VI-LCB` in Rashidinejad et al. (2021), and **(3)** we shave a $|\mathcal{S}|$ factor by introducing s-absorbing MDPs for each state as in Agarwal et al. (2020a).

4.2 Comparison Under Expert Data

We first compare the performance bounds from Section 4.1 when the offline dataset is generated from expert demonstrations. In relation to Condition 3.1, this corresponds to small C^*. Specifically, we consider $C^* \in [1, 1+\tilde{\mathcal{O}}(1/N)]$ so that the suboptimality of BC in Theorem 4.1 scales as $\tilde{\mathcal{O}}(|\mathcal{S}|H/N)$. In this regime, we perform a nuanced comparison by analyzing specific scenarios where RL may outperform BC. We consider the case of $C^* = 1$ and $C^* = 1 + \tilde{\mathcal{O}}(1/N)$ separately.

What happens when $C^* = 1$? In this case, we derive a information-theoretic lower-bound of $|\mathcal{S}|H/N$ for any offline algorithm. Our result in Theorem 4.3 utilizes the analysis of Rajaraman et al. (2020) but additionally factoring in Condition 3.2.

Theorem 4.3 (Information-theoretic lower-bound for offline learning with $C^* = 1$)**.** *For any learner $\widehat{\pi}$, there exists an MDP $\mathcal{M}$ satisfying Assumption 3.2, and a deterministic expert π^*, such that the expected suboptimality of the learner is lower-bounded:*

$$\sup_{\mathcal{M},\pi^*} \mathsf{SubOpt}(\widehat{\pi}) \gtrsim \frac{|\mathcal{S}|H}{N}$$

The proof of Theorem 4.3 uses the same hard instance from Theorem 6.1 of Rajaraman et al. (2020), except that one factor of H is dropped due to Condition 3.2. The other factor of H arises from the performance difference lemma and is retained. In this case, where BC achieves the lower bound up to logarithmic factors, we argue that we cannot improve over BC. This is because the suboptimality of BC is entirely due to encountering states that do not appear in the dataset; without additional assumptions on the ability to generalize to unseen states, offline RL must incur the same suboptimality, as both methods would choose actions uniformly at random.

Practical Insight 4.1. *When no assumptions are made on the environment structure, both offline RL and BC perform equally poorly with trajectories from an expert demonstrator.*

However, we argue that even with expert demonstrations as data, $C^* = 1$ is an unrealistic assumption. Naively, it seems plausible that the expert who collected the dataset did not perform optimally at every transition; this is often true for humans, or stochastic experts as ϵ-greedy of maximum-entropy policies. In addition, a scenario where $C^* > 1$ even though the expert behaves optimally is under distribution shift of the environment. One practical example of this is when the initial state distribution changes between dataset collection and evaluation (*e.g.*, in robotics (Singh et al., 2020), or self-driving (Bojarski et al., 2016)). Since the normalized state-action marginals $d^*(\mathbf{s}, \mathbf{a}), \mu(\mathbf{s}, \mathbf{a})$ are impacted by $\rho(\mathbf{s})$, this would lead to $C^* > 1$ even when the expert behaves exactly as π^*.

What happens when $C^* = 1 + \tilde{\mathcal{O}}(1/N)$? Here C^* is small enough that BC still achieves the same optimal $\tilde{\mathcal{O}}(|\mathcal{S}|H/N)$ performance guarantee. However, there is suboptimality incurred by BC for even states that appear in the dataset due to distribution shift, which allows us to argue about structural properties of MDPs that allow offline RL to perform better across those states, particularly for problems with large effective horizon H. We motivate one such structure below.

In several practical problem domains, the return of any trajectory can mostly be explained by the actions taken at a small fraction of states, which we call *critical states*. This can occur when there exist a large proportion of states where it is not costly to recover after deviating from the optimal trajectory, or when there exist a large volume of optimal trajectories. We give two example domains: robotic manipulation and navigation. In manipulation tasks such as grasping, if the robot is not near the object, it can take many different actions and still pick up the object by the end; this is because unless the object breaks, actions taken by the robot are typically reversible (Kalashnikov et al., 2018a). It is only at a few "critical states" when the robot grasps the object, where the robot should be careful to not drop and break the object. In navigation, there may exist multiple paths that end at the same goal, particularly in large, unobstructed areas (Savva et al., 2019). For example, while navigating through a wide tunnel, the exact direction the agent takes may not matter so much as multiple directions will take the agent through the tunnel, in contrast, there are "critical states" like narrow doorways where taking a specific action is important. An illustration is shown in Figure 3. Domains that do not satisfy this structure include cliffwalk environments, where a incorrect action at any state will cause the agent to fall off the cliff (Schaul et al., 2015). Formally:

Definition 4.1 (Critical states)**.** *A state $\mathbf{s}$ is said to be non-critical, i.e., $\mathbf{s} \in \mathcal{S} \setminus \mathcal{C}$ if the advantage of any action $\mathbf{a} \in \mathcal{A}$ is close to 0 under the optimal policy, i.e, $|\max_{\mathbf{a}'} Q^*(\mathbf{s}, \mathbf{a}') - Q^*(\mathbf{s}, \mathbf{a})| \leq \varepsilon/H$.*

Condition 4.1 (Volume of critical states is small.)**.** $\exists p_c \in (0,1)$ *that satisfies:* $|\mathcal{C}| \leq p_c|S|$.

We can show that, if the MDP satisfies having a small fraction or volume of critical states, then conservative offline RL enjoys stronger guarantees than BC.

Corollary 4.1 (Performance of conservative offline RL with critical states)**.** *Under Conditions 4.1, 3.1 and 3.2, a policy $\widehat{\pi}^*$ found by the conservative offline RL can satisfy*

$$\mathsf{SubOpt}(\widehat{\pi}^*) \leq \sqrt{\frac{p_c C^* |\mathcal{S}| H \iota}{N}} + \frac{p_c C^* |\mathcal{S}| H \iota}{N} + \varepsilon\,.$$

For $\varepsilon = \mathcal{O}(\sqrt{H})$, if the environment satisfies $p_c = \mathcal{O}(1/\sqrt{H})$, meaning we encounter $\mathcal{O}(\sqrt{H})$ critical states on average in any trajectory, then we achieve better scaling in H with offline RL than BC. Note that BC does not enjoy the benefit of few critical states because it is agnostic to the reward of the environment and therefore limited by the $\mathcal{O}(H)$ suboptimality of the behavior policy.

Practical Insight 4.2. *Offline RL is preferred over BC, even with expert or near-expert data, when either the initial state distribution changes during deployment and when the environment has a few "critical" states, i.e., states where precisely taking the expert action is necessary.*

4.3 Comparison Under Noisy Data

In practice, it is often much more tractable to obtain suboptimal demonstrations rather than expert ones. From Theorem 4.1, we see that for $C^* = 1 + \Omega(1/\sqrt{N})$, BC will incur suboptimality that is worse asymptotically than offline RL. In constrast, from Theorem 4.2, we note that offline RL does not scale nearly as poorly with increasing C^*. Since offline RL is not as reliant on the performance of the behavior policy due to explicitly modeling the rewards, we hypothesize that RL can actually benefit from suboptimal data that is well-explored. In this section, we aim to answer the following question: *Can offline RL with suboptimal data outperform BC with an equal amount of expert data?*

We show in Corollary 4.2 that if the suboptimal dataset $\mathcal{D}$ satisfies an additional coverage condition, then running conservative offline RL can attain $\tilde{\mathcal{O}}(\sqrt{H})$ suboptimality in the horizon. This implies, perhaps surprisingly, that for long horizon tasks, offline RL using noisy, suboptimal data can outperform even BC on expert data. Formally, our coverage condition is the following:

Condition 4.2 (Coverage of the optimal policy)**.** $\exists b \in [\log H/N, 1)$ *such that μ satisfies:* $\forall (\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$ *where* $d^*(\mathbf{s}, \mathbf{a}) \geq b/H$*, we have* $\mu(\mathbf{s}, \mathbf{a}) \geq b$.

Intuitively, this means that the data distribution puts sufficient mass on states that have non-negligible density in the optimal policy distribution. Note that this is a weaker condition than prior works that require **(1)** full coverage of the state-action space, and **(2)** enforce a constraint on the empirical state-action visitations $\widehat{\mu}(\mathbf{s}, \mathbf{a})$ instead of $\mu(\mathbf{s}, \mathbf{a})$ (Ren et al., 2021; Zhang et al., 2021). This condition is reasonable when the dataset is collected by ϵ-greedy or a maximum-entropy expert, which is a standard assumption in MaxEnt IRL (Ziebart et al., 2008). Even if the expert is not noisy, we argue that in several real-world applications, creating noisy-expert data is feasible. In many robotics applications, it is practical to augment expert demonstrations with counterfactual data by using scripted exploration policies (Kalashnikov et al., 2018a; 2021). Existing work has also simulated trajectories to increase coverage, as was done in self-driving by perturbing the vehicle location (Bojarski et al., 2016), or in robotics using learned simulators (Rao et al., 2020). For intuition, we provide an illustrative example of the noisy data that can help offline RL in Figure 4.

Corollary 4.2 (Performance of conservative offline RL with noisy data)**.** *If μ satisfies Condition 4.2, and under Conditions 3.1 and 3.2, the policy $\widehat{\pi}^*$ found by conservative offline RL can satisfy:*

$$\mathsf{SubOpt}(\widehat{\pi}^*) \lesssim \sqrt{\frac{H\iota}{bN}} + \frac{H\iota}{bN} + \sqrt{b\iota} + \frac{C^*|\mathcal{S}|\iota}{N}\,.$$

For some b, offline RL can improve upon the lower-bound for BC in Theorem 4.3, e.g, if $b = \mathcal{O}(\sqrt{H}/N)$, then the bound in Corollary 4.2 has $\tilde{\mathcal{O}}(\sqrt{H})$ scaling rather than $\tilde{\mathcal{O}}(H)$ for BC. Thus, when the data satisfies the practical coverage conditions (more discussion in Appendix D), offline RL performs better in long-horizon tasks compared to BC with the same amount of expert data.

Practical Insight 4.3. *Offline RL outperforms BC on expert data on long-horizon tasks, when provided with an equal amount of noisy-expert data. So, if noisy-expert data is easy to collect, doing so and running offline RL can be a much better option.*

4.4 Comparison of Generalized BC Methods and Offline RL

So far we have studied scenarios where offline RL can outperform naive BC. One might now wonder how offline RL methods perform relative to generalized BC methods that additionally use reward information to inform learning. We study two such approaches: **(1)** filtered BC (Chen et al., 2021), which only fits to the top k-percentage of trajectories in $\mathcal{D}$, measured by the total reward, and **(2)** BC with one-step policy improvement (Brandfonbrener et al., 2021), which fits a Q-function for the behavior policy, then uses the values to perform one-step of policy improvement over the behavior policy. In this section, we aim to answer how these methods perform relative to RL.

Filtered BC. In expectation, this algorithm uses αN samples of the offline dataset $\mathcal{D}$ for $\alpha \in [0, 1]$ to perform BC on. This means that the upper bound (Theorem 4.1) will have worse scaling in N. For $C^* = 1$, this leads to a strictly worse bound than regular BC. However, for suboptimal data, the filtering step could decrease C^* by filtering out suboptimal trajectories, allowing filtered BC to outperform traditional BC. Nevertheless, from our analysis in Section 4.3, offline RL is still preferred to filtered BC because RL can leverage the noisy data and potentially achieve sub-linear suboptimality $O(\sqrt{H})$ in the horizon, whereas even filtered BC would always incur $O(H)$ suboptimality.

BC with policy improvement. This algorithm utilizes the entire dataset to estimate the Q-value of the behavior policy, $\widehat{Q}^{\widehat{\pi}_\beta}$, and performs one step of policy improvement using the estimated Q-function, typically via an advantage-weighted update: $\widehat{\pi}^1(\mathbf{a}|\mathbf{s}) = \widehat{\pi}_\beta(\mathbf{a}|\mathbf{s}) \exp(\eta H \widehat{A}^{\widehat{\pi}_\beta}(\mathbf{s}, \mathbf{a}))/\mathbb{Z}_1(\mathbf{s})$. *When would this algorithm perform poorly compared to offline RL?* Intutively, this would happen when multiple steps of policy improvement are needed to effectively discover high-advantage actions under the behavior policy. This is the case when the the behavior policy puts low density on high-advantage transitions. In Theorem 4.4, we show that more than one step of policy improvement can improve the policy under Condition 4.2 for the softmax policy parameterization (Agarwal et al., 2021a).

Theorem 4.4 (One-step is worse than k-step policy improvement)**.** *Assume that the learned policies are represented via a softmax parameterization (Equation 3, Agarwal et al. (2021a)). Let $\widehat{\pi}^k$ denote the policy obtained after k-steps of policy improvement using exponentiated advantage weights. Then, under Condition 4.2, the performance difference between $\widehat{\pi}^k$ and $\widehat{\pi}^1$ is lower-bounded by:*

$$J(\widehat{\pi}^k) - J(\widehat{\pi}^1) \gtrsim \frac{k}{H\eta} \mathbb{E}_{\mathbf{s}\sim\mu}\left[\frac{1}{k}\sum_{t=1}^{k} \log \mathbb{Z}_t(\mathbf{s})\right] - \sqrt{\frac{C^* H \iota}{N}}.$$

A proof of Theorem 4.4 is provided in Appendix B.5. This result implies that when the average exponentiated empirical advantage $1/k \sum_{i=1}^{k} \log \mathbb{Z}_t(\mathbf{s})$ is large enough (*i.e.*, $\geq c_0$ for some universal constant), which is usually the case when the behavior policy is highly suboptimal, then for $k = \mathcal{O}(H)$, multiple steps of policy improvement will improve performance, *i.e.*, $J(\widehat{\pi}^k) - J(\widehat{\pi}^1) = \tilde{\mathcal{O}}(H - \sqrt{H/N})$, where the gap increases with a longer horizon. This is typically the case when the structure of the MDP allow for stitching parts of poor-performing trajectories. One example is in navigation, where trajectories that fail may still contain segments of a successful trajectory.

Practical Insight 4.4. *Using multiple policy improvement steps (i.e., full offline RL) can lead to greatly improved performance on long-horizon tasks, particularly when parts of various trajectories can be concatenated or stitched together to give better performance.*

5 Empirical Evaluation of BC and Offline RL

Having characterized scenarios where at least some offline RL algorithms can outperform BC in theory, we now validate our results empirically. Concretely, we aim to answer the following questions: **(1)** Does some offline RL method trained on expert data outperform BC on expert data in practice? **(2)** Can offline RL trained on noisy data outperform BC on expert data?, and **(3)** How does full offline RL compare to the generalized BC methods studied in Section 4.4? We will first validate our findings on a tabular gridworld domain, and then on several high-dimensional offline RL problems.

Diagnostic experiments in gridworld. We first evaluate tabular versions of the BC and offline RL methods analyzed in Section 4.1 on sparse-reward 10×10 gridworlds environments (Fu et al., 2019). Complete details about the setup can be found in Appendix E.1. On a high-level, we consider three different environments, each with varying number of critical states, from "Single Critical" with

exactly one, to "Cliffwalk" where every state is critical and veering off yields zero reward. The methods we consider are: naive BC (BC), conservative RL (RL-C), policy-constraint RL (RL-PC), and generalized BC with one-step and k-step policy improvement (BC-PI, BC-kPI).

In the left plot of Figure 1, we show the return (normalized by return of the optimal policy) across all the different environments for optimal data ($C^* = 1$) and data generated from the optimal policy but with a different initial state distribution ($C^* > 1$ but $\pi_\beta(\cdot|\mathbf{s}) = \pi^*(\cdot|\mathbf{s})$). As expected from our discussion in Section 4.2, BC performs best under $C^* = 1$, but RL-C and RL-PC performs much better when $C^* > 1$; also BC with one-step policy improvement outperforms naive BC for $C^* > 1$, but does not beat RL. In Figure 1 (right), we vary C^* by interpolating the dataset with one generated by a random policy, where α is the proportion of random data. RL performs much better over all BC methods, when the data supporting our analysis in Section 4.3. Finally, BC with multiple policy improvement steps performs better than one step when the data is noisy, which validates Theorem 4.4.

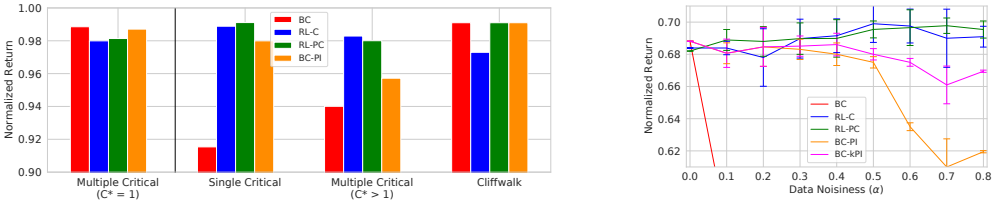

Figure 1: Offline RL vs BC on gridworld domains. *Left*: We compare offline RL and BC algorithms on three different gridworlds with varying number of critical points for expert and near-expert data. *Right*: Taking the "Multiple Critical" domain, we examine the effect of increasing the noisiness of the dataset by interpolating it with one generated by a random policy, and show that RL improves drastically with increased noise over BC.

Evaluation in high-dimensional tasks. Next, we turn to deep offline RL. We consider a diverse set of domains (shown on the right) and behavior policies that are representative of scenarios where we would decide between offline RL and BC: multi-stage robotic manipulation tasks from state (Adroit domains from Fu et al. (2020)) and image observations (Singh et al., 2020), antmaze navigation (Fu et al., 2020), and 7 Atari games (Agarwal et al., 2020b). We use the scripted expert provided by Fu et al. (2020) for antmaze and the one provided by Singh et al. (2020) for manipulation, an RL-trained expert for Atari, and human expert for Adroit (Rajeswaran et al., 2018). In scenarios where suboptimal data is used to train offline RL, we use failed attempts to solve the task from a noisy expert policy (*i.e.*, previous policies in the replay buffer for Atari, and noisy scripted experts for antmaze and manipulation). All these tasks utilize sparse rewards such that the return of any trajectory is bounded by a constant much smaller than the horizon. We use CQL (Kumar et al., 2020) as a representative offline RL method, and utilize Brandfonbrener et al. (2021) as a representative BC-PI method.

Tuning offline RL and BC. Naïvely running offline RL can lead to poor performance, as noted by prior works (Mandlekar et al., 2021; Florence et al., 2021). This is also true for BC, but, some solutions such as early stopping based on validation losses, can help improve performance. We claim that a similar tuning strategy is also crucial for offline RL. In our experiments we utilize the offline workflow proposed by Kumar et al. (2021c) to perform policy selection, and address overfitting and underfitting, purely offline. When the Q-values learned by CQL are extremely negative (typically on the Adroit domains), we utilize dropout with probability 0.4 on the layers of the Q-function to combat overfitting. On the other hand, when the Q-values exhibit a relatively stable trend (*e.g.*, in Antmaze or Atari), we utilize the DR3 regularizer (Kumar et al., 2021a) to increase capacity. Consistent with prior work, we find that naïve offline RL generally performs worse than BC without offline tuning, but we find that *offline-tuned* offline RL generally outperforms BC. To make a stronger comparison, we tuned BC using the online rollouts. We applied regularizers such as dropout on the BC policy to prevent overfitting in Adroit, and utilized a larger ResNet (He et al., 2016) architecture for the robotic manipulation tasks and Atari domains. For BC, we report the performance of the *best* checkpoint found during training, giving BC an unfair advantage, but we still find that *offline-tuned* offline RL performs better. More details about tuning can be found in Appendix F.

Answers to questions (1) to (3). For **(1)**, we run CQL and BC on expert data in each task, and present the comparison in Table 1 and Figure 2. While naïve CQL performs comparable or worse than BC in this case, after offline tuning, CQL outperforms BC. This tuning does not require any

Domain / Behavior Policy	Task/Data Quality	BC	Naïve CQL	Tuned CQL
AntMaze (scripted)	Medium, Expert	53.2%±8.7%	20.8% ± 1.0%	55.9% ± 3.2%
	Large, Expert	**4.83%**±0.8%	0.0% ± 0.0%	0.0% ± 0.0%
	Medium, Expert w/ diverse initial	55.2%±6.7%	19.0% ± 5.2%	**67.0%** ± 7.3%
	Large, Expert w/ diverse initial	1.3%±0.5%	0.0% ± 0.0	**5.1%** ± 6.9%
Manipulation (scripted)	pick-place-open-grasp, Expert	14.5%±1.8%	12.3%±5.3%	**23.5%**±6.0%
	close-open-grasp, Expert	17.4%±3.1%	20.0%±6.0%	**49.7%**±5.4%
	open-grasp, Expert	33.2%±8.1%	22.8%±5.3%	**51.9%**±6.8%
Adroit (Human)	hammer-human-v1, Expert	71.0% ± 9.3%	62.5% ± 39.0%	**78.1%** ± 6.7%
	door-human-v1, Expert	**86.3%** ± 6.5%	70.3% ± 27.2%	79.1% ± 4.7%
	pen-human-v1, Expert	**73.0** % ± 9.1%	64.0% ± 6.9%	**74.1%** ± 6.1%
	relocate-human-v1, Expert	0.0% ± 0.0%	0.0% ± 0.0%	0.0% ± 0.0%

Table 1: Offline CQL vs. BC with expert dataset compositions averaged over 3 seeds. While naïve offline CQL often performs comparable or worse than BC, the performance of offline RL improves drastically after offline tuning. Also note that offline RL can improve when provided with diverse initial states in the Antmaze domain. Additionally, note that offline-tuned offline RL outperforms BC significantly in the manipulation domains.

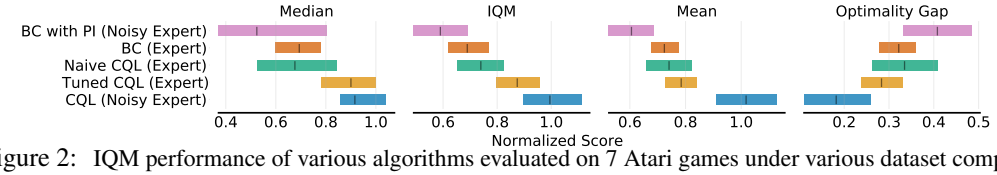

Figure 2: IQM performance of various algorithms evaluated on 7 Atari games under various dataset compositions (per game scores in Table 3). Note that offline-tuned CQL with expert data ("Tuned CQL") outperforms cloning the expert data ("BC (Expert)"), even though naïve CQL is comparable to BC in this setting. When CQL is provided with noisy-expert data, it significantly outperforms cloning the expert policy.

additional online rollouts. Note that while BC performs better or comparable to RL for antmaze (large) with expert data, it performs worse than RL when the data admits a more diverse initial state distribution such that $C^* \neq 1$, even though the behavior policy matches the expert.

For **(2)**, we compare offline RL trained on noisy-expert data with BC trained on on an equal amount of expert data, on domains where noisy-expert data is easy to generate: (a) manipulation domains (Table 2) and (b) Atari games (Figure 2). Observe that CQL outperforms BC and also improves over only using expert data. The performance gap also increases with H, *i.e.*, open-grasp ($H = 40$) vs pick-place-open-grasp ($H = 80$) vs Atari domains ($H = 27000$). This validates that some form of offline RL with noisy-expert data can outperform BC with expert data, particularly on long-horizon tasks.

Task	BC (Expert)	CQL (Noisy Expert)
pick-place-open-grasp	14.5% ± 1.8%	**85.7%** ± 3.1%
close-open-grasp	17.4% ± 3.1%	**90.3%** ± 2.3%
open-grasp	33.2% ± 8.1%	**92.4%** ± 4.9%

Table 2: CQL with noisy-expert data vs BC with expert data with equal dataset size on manipulation tasks. CQL outperforms BC as well as CQL with only expert data.

Finally, for **(3)**, we compare CQL to a representative BC-PI method (Brandfonbrener et al., 2021) trained using noisy-expert data on Atari domains, which present multiple stitching opportunities. The BC-PI method estimates the Q-function of the behavior policy using SARSA and then performs one-step of policy improvement. The results in Figure 2 support what is predicted by our theoretical results, *i.e.*, BC-PI still performs significantly worse than CQL with noisy-expert data, even though we utilized online rollouts for tuning BC-PI and report the best hyperparameters found.

6 Discussion

We sought to understand if offline RL is at all preferable over running BC, even provided with expert or near-expert data. While in the worst case, both methods attain similar performance on expert data, additional assumptions on the environment can provide certain offline RL methods with an advantage. We also show that running RL on noisy-expert, suboptimal data attains more favorable guarantees compared to running BC on expert data for the same task, using equal amounts of data. Empirically, we observe that offline-tuned CQL can outperform BC on various practical problem domains, with different kinds of expert policies. While our work is an initial step towards understanding when RL presents a favorable approach, there is still plenty of room for further investigation. Our theoretical analysis can be improved to handle function approximation. Understanding if offline RL is preferred over BC for other real-world data distributions is also important. Finally, our work focuses on analyzing cases where we might expect offline RL to outperform BC. An interesting direction is to understand cases where the opposite holds; such analysis would further contribute to this discussion.

REPRODUCIBILITY STATEMENT

For theoretical results, we provide explanations of all the assumptions and a complete proof of the claims in Appendix B. We provide complete experimental details regarding the tasks and tuning strategy in the Appendix F. Additionally, we follow the recommendations of Agarwal et al. (2021b) for reliable evaluation in deep RL and report the statistical uncertainty in reported results including aggregate performance metrics (Figure 2). We provide individual scores in Table 3 and we will open-source our code.

REFERENCES

Alekh Agarwal, Sham Kakade, and Lin F Yang. Model-based reinforcement learning with a generative model is minimax optimal. In *Conference on Learning Theory*, pp. 67–83. PMLR, 2020a.

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research*, 22(98):1–76, 2021a.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020b.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021b.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Int. Res.*, 47(1):253–279, May 2013. ISSN 1076-9757.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

David Brandfonbrener, William F Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *arXiv preprint arXiv:2106.08909*, 2021.

Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. *ICML*, 2019.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 456–464, 2019.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, 2018.

Amir-massoud Farahmand, Csaba Szepesvári, and Rémi Munos. Error propagation for approximate policy and value iteration. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=rif3a5NAxU6.

Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. Diagnosing bottlenecks in deep Q-learning algorithms. *arXiv preprint arXiv:1902.10250*, 2019.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *arXiv preprint arXiv:2106.06860*, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

Meera Hahn, Devendra Chaplot, Shubham Tulsiani, Mustafa Mukadam, James M. Rehg, and Abhinav Gupta. No rl, no simulation: Learning to navigate without navigating. In *Advances in Neural Information Processing Systems*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Natasha Jaques, Judy Hanwen Shen, Asma Ghandeharioun, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Shane Gu, and Rosalind Picard. Human-centric dialog training via offline reinforcement learning. *arXiv preprint arXiv:2010.05848*, 2020.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pp. 651–673, 2018a.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pp. 651–673. PMLR, 2018b.

Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

Ilya Kostrikov, Jonathan Tompson, Rob Fergus, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. *arXiv preprint arXiv:2103.08050*, 2021.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pp. 11761–11771, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

Aviral Kumar, Rishabh Agarwal, Aaron Courville, Tengyu Ma, George Tucker, and Sergey Levine. Value-based deep reinforcement learning requires explicit regularization. In *RL for Real Life Workshop & Overparameterization: Pitfalls and Opportunities Workshop, ICML*, 2021a. URL https://drive.google.com/file/d/1Fg43H5oagQp-ksjpWBf_aDYEzAFMVJm6/view.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=O9bnihsFfXU.

Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021c. URL https://openreview.net/forum?id=fy4ZBWxYbIo.

Vladislav Kurenkov and Sergey Kolesnikov. Showing your offline reinforcement learning work: Online evaluation budget matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

Sascha Lange, Thomas Gabel, and Martin A. Riedmiller. Batch reinforcement learning. In *Reinforcement Learning*, volume 12. Springer, 2012.

Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning*, pp. 3652–3661. PMLR, 2019.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *CoRR*, abs/1904.08473, 2019.

Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch reinforcement learning without great exploration. *arXiv preprint arXiv:2007.08202*, 2020.

Ajay Mandlekar, Fabio Ramos, Byron Boots, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Dieter Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4414–4420. IEEE, 2020.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Fei-Fei Li, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=JrsfBJtDFdI.

Andreas Maurer and Massimiliano Pontil. Empirical bernstein bounds and sample variance penalization. *arxiv preprint arxiv:0907.3740*, 2009.

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

Rémi Munos. Error bounds for approximate policy iteration. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pp. 560–567. AAAI Press, 2003. ISBN 1577351894.

Rémi Munos. Error bounds for approximate value iteration. In *AAAI Conference on Artificial intelligence (AAAI)*, pp. 1006–1011. AAAI Press, 2005.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

Kimia Nadjahi, Romain Laroche, and Rémi Tachet des Combes. Safe policy improvement with soft baseline bootstrapping. *arXiv preprint arXiv:1907.05079*, 2019.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Transactions On Graphics (TOG)*, 37(6):1–14, 2018.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

Nived Rajaraman, Lin F Yang, Jiantao Jiao, and Kannan Ramachandran. Toward the fundamental limits of imitation learning. *arXiv preprint arXiv:2009.05990*, 2020.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Robotics: Science and Systems*, 2018.

Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11157–11166, 2020.

Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *arXiv preprint arXiv:2103.12021*, 2021.

Tongzheng Ren, Jialian Li, Bo Dai, Simon S Du, and Sujay Sanghavi. Nearly horizon-free offline reinforcement learning. *arXiv preprint arXiv:2103.14077*, 2021.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 661–668, 2010.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *International Conference on Learning Representations (ICLR)*, 2015.

Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. Ving: Learning open-world navigation with visual goals. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13215–13222. IEEE, 2021.

Susan M Shortreed, Eric Laber, Daniel J Lizotte, T Scott Stroup, Joelle Pineau, and Susan A Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine learning*, 84(1-2):109–136, 2011.

Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. *arXiv preprint arXiv:2010.14500*, 2020.

Samarth Sinha, Ajay Mandlekar, and Animesh Garg. S4RL: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=8xC5NNej-l_.

Adith Swaminathan and Thorsten Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *J. Mach. Learn. Res*, 16:1731–1755, 2015.

L. Wang, Wei Zhang, Xiaofeng He, and H. Zha. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

Ruosong Wang, Dean Foster, and Sham M. Kakade. What are the statistical limits of offline {rl} with linear function approximation? In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=30EvkP2aQLD.

Ziyu Wang, Alexander Novikov, Konrad Żołna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *arXiv preprint arXiv:2006.15134*, 2020.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Tengyang Xie and Nan Jiang. Q* approximation schemes for batch reinforcement learning: A eoretical comparison. 2020.

Tengyang Xie, Ching-An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal. Bellman-consistent pessimism for offline reinforcement learning. *arXiv preprint arXiv:2106.06926*, 2021a.

Tengyang Xie, Nan Jiang, Huan Wang, Caiming Xiong, and Yu Bai. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021b.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *arXiv preprint arXiv:2102.08363*, 2021.

Andrea Zanette. Exponential lower bounds for batch reinforcement learning: Batch rl can be exponentially harder than online rl. *arXiv preprint arXiv:2012.08005*, 2020.

Yufeng Zhang, Qi Cai, Zhuoran Yang, Yongxin Chen, and Zhaoran Wang. Can temporal-difference and q-learning learn representation? a mean-field theory. *arXiv preprint arXiv:2006.04761*, 2020.

Zihan Zhang, Xiangyang Ji, and Simon Du. Is reinforcement learning more difficult than bandits? a near-optimal algorithm escaping the curse of horizon. In *Conference on Learning Theory*, pp. 4528–4531. PMLR, 2021.

B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *International Conference on Artificial Intelligence (AAAI)*, 2008.

A PSEUDOCODE FOR ALGORITHMS

Algorithm 1 Conservative Offline RL Algorithm

Require: Offline dataset $\mathcal{D}$, discount factor γ, and confidence level δ

1: Compute $n(\mathbf{s}, \mathbf{a})$ from $\mathcal{D}$, and estimate $\widehat{r}(\mathbf{s}, \mathbf{a})$, $\widehat{P}(s'|\mathbf{s}, \mathbf{a})$, $\forall(\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$

2: Initialize $\widehat{Q}(\mathbf{s}, \mathbf{a}) \leftarrow 0, \widehat{V}(\mathbf{s}) \leftarrow 0, \forall(\mathbf{s}, \mathbf{a})$

3: **for** $i = 1, 2, \ldots, m$ **do**

Calculate $b(\mathbf{s}, \mathbf{a})$ as:

$$b(\mathbf{s}, \mathbf{a}) \leftarrow \sqrt{\frac{\mathbb{V}(\widehat{P}(\mathbf{s}, \mathbf{a}), \widehat{V}) \log(|\mathcal{S}||\mathcal{A}|m/\delta))}{(n(\mathbf{s}, \mathbf{a}) \wedge 1)}} + \sqrt{\frac{\widehat{r}(\mathbf{s}, \mathbf{a}) \log(|\mathcal{S}||\mathcal{A}|m/\delta)}{(n(\mathbf{s}, \mathbf{a}) \wedge 1)}} + \frac{\log(|\mathcal{S}||\mathcal{A}|m/\delta)}{(n(\mathbf{s}, \mathbf{a}) \wedge 1)}$$

Calculate $\widehat{\pi}^*(\mathbf{s})$ as:

$$\widehat{Q}(\mathbf{s}, \mathbf{a}) \leftarrow \widehat{r}(\mathbf{s}, \mathbf{a}) - b(\mathbf{s}, \mathbf{a}) + \gamma \widehat{P}(\mathbf{s}, \mathbf{a}) \cdot \widehat{V}$$
$$\widehat{V}(\mathbf{s}) \leftarrow \max_a \widehat{Q}(\mathbf{s}, \mathbf{a})$$
$$\widehat{\pi}^*(\mathbf{s}) \leftarrow \arg\max_a \widehat{Q}(\mathbf{s}, \mathbf{a})$$

4: Return $\widehat{\pi}^*$

Algorithm 2 Policy-Constraint Offline RL Algorithm

Require: Offline dataset $\mathcal{D}$, discount factor γ, and threshold b

1: Compute $n(\mathbf{s}, \mathbf{a})$ from $\mathcal{D}$, and estimate $\widehat{r}(\mathbf{s}, \mathbf{a})$, $\widehat{P}(s'|\mathbf{s}, \mathbf{a})$, $\widehat{\mu}(\mathbf{s}, \mathbf{a})$, $\forall(\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$

2: Compute $\zeta(\mathbf{s}, \mathbf{a}) \leftarrow 1\{\widehat{\mu}(\mathbf{s}, \mathbf{a}) \geq b\}$, $\forall(\mathbf{s}, \mathbf{a})$

3: Initialize $\widehat{\pi}^*(\mathbf{a}|\mathbf{s}) \leftarrow \frac{1}{|\mathcal{A}|}$, $\widehat{Q}_\zeta^{\widehat{\pi}^*}(\mathbf{s}, \mathbf{a}) \leftarrow 0$, $\widehat{V}_\zeta^{\widehat{\pi}^*}(\mathbf{s}) \leftarrow 0$, $\forall(\mathbf{s}, \mathbf{a})$

4: **for** $\ell = 1, 2, \ldots, k$ **do**

5: **for** $i = 1, 2, \ldots, m$ **do**

Update $\widehat{Q}_\zeta^{\widehat{\pi}^*}(\mathbf{s}, \mathbf{a}), \widehat{V}_\zeta^{\widehat{\pi}^*}(\mathbf{s})$ as:

$$\widehat{Q}_\zeta^{\widehat{\pi}^*}(\mathbf{s}, \mathbf{a}) \leftarrow \widehat{r}(\mathbf{s}, \mathbf{a}) + \gamma \widehat{P}(\mathbf{s}, \mathbf{a}) \cdot \widehat{V}_\zeta^{\widehat{\pi}^*}$$
$$\widehat{V}_\zeta^{\widehat{\pi}^*}(\mathbf{s}) \leftarrow \sum_{\mathbf{a}} \widehat{\pi}^*(\mathbf{a}|\mathbf{s}) \zeta(\mathbf{s}, \mathbf{a}) \cdot \widehat{Q}(\mathbf{s}, \mathbf{a})$$

Compute $\widehat{\pi}^*$ as:

$$\widehat{\pi}^* \leftarrow \arg\max_\pi \mathbb{E}_{\mathbf{s} \sim \mathcal{D}} \left[\mathbb{E}_{\mathbf{a} \sim \pi'} \left[\zeta(\mathbf{s}, \mathbf{a}) \cdot \widehat{Q}_\zeta^\pi(\mathbf{s}, \mathbf{a}) \right] \right]$$

6: Return $\widehat{\pi}^*$.

B PROOFS

B.1 PROOF OF THEOREM 4.1

Let π_β be the behavior policy that we fit our learned policy $\widehat{\pi}_\beta$ to. Recall that the BC algorithm we analyze fits $\widehat{\pi}_\beta$ to choose actions according to the empirical dataset distribution for states that appear in dataset $\mathcal{D}$, and uniformly at random otherwise. We have

$$\mathbb{E}_{\mathcal{D}} \left[J(\pi^*) - J(\widehat{\pi}_\beta) \right] \leq J(\pi^*) - J(\pi_\beta) + \mathbb{E}_{\mathcal{D}} \left[J(\pi_\beta) - J(\widehat{\pi}_\beta) \right]$$

The following lemma from Rajaraman et al. (2020) bounds the suboptimality from performing BC on a (potentially stochastic) expert, which we adapt below factoring in bounded returns of trajectories from Condition 3.2.

Lemma B.1 (Theorem 4.4, Rajaraman et al. (2020))**.** *The policy returned by BC on behavior policy π_β has expected error bounded as*

$$\mathbb{E}_{\mathcal{D}} \left[J(\pi_\beta) - J(\widehat{\pi}_\beta) \right] \leq \frac{SH \log N}{N},$$

where π_β could be stochastic.

Using Lemma B.1, we have $\mathbb{E}_{\mathcal{D}}\left[J(\pi_\beta) - J(\widehat{\pi}_\beta)\right] \leq SH\iota/N$. What remains is bounding the suboptimality of the behavior policy, which we can upper-bound as

$$\begin{aligned}
J(\pi^*) - J(\pi_\beta) &\leq \sum_{t=0}^{\infty}\sum_{\mathbf{s}} \gamma^t \mathbb{P}\left(s_t = s\right) \mathbb{E}_{\pi_\beta(\cdot|s)}\left[1\{\mathbf{a} \neq \pi_t^*(\mathbf{s})\}\right] \\
&= \frac{1}{2}\sum_{t=0}^{\infty}\sum_{\mathbf{s}} \gamma^t d_t^*(\mathbf{s}) \sum_{\mathbf{a}} |\pi_\beta(\mathbf{a}|\mathbf{s}) - 1\{\mathbf{a} = \pi_t^*(\mathbf{s})\}| \\
&= \frac{1}{2}\sum_{t=0}^{\infty}\sum_{(\mathbf{s},\mathbf{a})} \gamma^t \left|d_t^*(\mathbf{s})\pi_\beta(\mathbf{a}|\mathbf{s}) - d_t^*(\mathbf{s},\mathbf{a})\right| \\
&\leq \frac{C^* - 1}{2} H \sum_{(\mathbf{s},\mathbf{a})} \mu(\mathbf{s},\mathbf{a}) \\
&= \frac{(C^* - 1)H}{2},
\end{aligned}$$

where we use the definition of C^* in Condition 3.1. Taking the sum of both terms yields the desired result.

B.2 Proof of Theorem 4.2

In this section, we proof the performance guarantee for the conservative offline RL algorithm detailed in Algorithm 1. Recall that the algorithm we consider builds upon empirical value iteration but subtracts a penalty during each Q-update. Specifically, we initialize $Q_0(\mathbf{s},\mathbf{a}) = 0, V_0(\mathbf{s}) = 0$ for all $(\mathbf{s},\mathbf{a})$. Let $n(\mathbf{s},\mathbf{a})$ be the number of times $(\mathbf{s},\mathbf{a})$ appeared in $\mathcal{D}$, and let $\widehat{r}(\mathbf{s},\mathbf{a})$, $\widehat{P}(\mathbf{s},\mathbf{a})$ be the empirical estimates of their reward and transition probabilities. Then, for each iteration $i \in [m]$:

$$\begin{aligned}
\widehat{Q}_i(\mathbf{s},\mathbf{a}) &\leftarrow \widehat{r}(\mathbf{s},\mathbf{a}) - b_i(\mathbf{s},\mathbf{a}) + \gamma \widehat{P}(\mathbf{s},\mathbf{a}) \cdot \widehat{V}_{i-1}, \quad \text{for all } \mathbf{s},\mathbf{a}, \\
\widehat{V}_i(\mathbf{s}) &\leftarrow \max\{\widehat{V}_{t-1}(\mathbf{s}), \max_a \widehat{Q}_i(\mathbf{s},\mathbf{a})\}, \quad \text{for all } s,
\end{aligned}$$

In our algorithm we define the penalty function as

$$b_i(\mathbf{s},\mathbf{a}) \leftarrow \sqrt{\frac{\mathbb{V}(\widehat{P}(\mathbf{s},\mathbf{a}), \widehat{V}_{i-1})\iota}{(n(\mathbf{s},\mathbf{a}) \wedge 1)}} + \sqrt{\frac{\widehat{r}(\mathbf{s},\mathbf{a})\iota}{(n(\mathbf{s},\mathbf{a}) \wedge 1)}} + \frac{\iota}{(n(\mathbf{s},\mathbf{a}) \wedge 1)},$$

where we let ι to capture all poly-logarithmic terms. As notation, we drop the subscript i to denote the final $\widehat{Q}$ and $\widehat{V}$ at iteration m, where $m = H \log N$. Finally, the learned policy $\widehat{\pi}^*$ satisfies $\widehat{\pi}^*(\mathbf{s}) \in \arg\max_a \widehat{Q}(\mathbf{s},\mathbf{a})$ for all s, if multiple such actions exist, then the policy samples an action uniformly at random.

B.2.1 Technical Lemmas

Lemma B.2 (Bernstein's inequality)**.** *Let $X, \{X_i\}_{i=1}^n$ be i.i.d random variables with values in $[0,1]$, and let $\delta > 0$. Then we have*

$$\mathbb{P}\left(\left|\mathbb{E}\left[X\right] - \frac{1}{n}\sum_{i=1}^n X_i\right| > \sqrt{\frac{2\mathrm{Var}\left[X\right]\log(2/\delta)}{n}} + \frac{\log(2/\delta)}{n}\right) \leq \delta\,.$$

Lemma B.3 (Theorem 4, Maurer & Pontil (2009))**.** *Let $X, \{X_i\}_{i=1}^n$ with $n \geq 2$ be i.i.d random variables with values in $[0,1]$. Define $\bar{X} = \frac{1}{n}\sum_{i=1}^n X_i$ and $\widehat{\mathrm{Var}}(X) = \frac{1}{n}\sum_{i=1}^n (X_i - \bar{X})^2$. Let $\delta > 0$. Then we have*

$$\mathbb{P}\left(\left|\mathbb{E}\left[X\right] - \frac{1}{n}\sum_{i=1}^n X_i\right| > \sqrt{\frac{2\widehat{\mathrm{Var}}(\bar{X})\log(2/\delta)}{n-1}} + \frac{7\log(2/\delta)}{3(n-1)}\right) \leq \delta\,.$$

Lemma B.4 (Lemma 4, Ren et al. (2021)). *Let $\lambda_1, \lambda_2 > 0$ be constants. Let $f : \mathbb{Z}_{\geq 0} \to \mathbb{R}$ be a function such that $f(i) \leq H$, $\forall i$ and $f(i)$ satisfies the recursion*

$$f(i) \leq \sqrt{\lambda_1 f(i+1)} + \lambda_1 + 2^{i+1}\lambda_2 \,.$$

Then, we have that $f(0) \leq 6(\lambda_1 + \lambda_2)$.

B.2.2 Pessimism Guarantee

The first thing we want to show is that with high probability, the algorithm provides pessimistic value estimates, namely that $\widehat{V}_i(\mathbf{s}) \leq V^*(\mathbf{s})$ for all $t \in [T]$ and $\mathbf{s} \in \mathcal{S}$. To do so, we introduce a notion of a "good" event, which occurs when our empirical estimates of the MDP are not far from the true MDP. We define $\mathcal{E}_1$ to be the event where

$$\left|(\widehat{P}(\mathbf{s},\mathbf{a}) - P(\mathbf{s},\mathbf{a})) \cdot \widehat{V}_i\right| \leq \sqrt{\frac{\mathbb{V}(\widehat{P}(\mathbf{s},\mathbf{a}), \widehat{V}_i)\iota}{(n(\mathbf{s},\mathbf{a}) \wedge 1)}} + \frac{\iota}{(n(\mathbf{s},\mathbf{a}) \wedge 1)} \tag{2}$$

holds for all $i \in [m]$ and $(\mathbf{s},\mathbf{a}) \in \mathcal{S} \times \mathcal{A}$. We also define $\mathcal{E}_2$ to be the event where

$$|\widehat{r}(\mathbf{s},\mathbf{a}) - r(\mathbf{s},\mathbf{a})| \leq \sqrt{\frac{\widehat{r}(\mathbf{s},\mathbf{a})\iota}{(n(\mathbf{s},\mathbf{a}) \wedge 1)}} + \frac{\iota}{(n(\mathbf{s},\mathbf{a}) \wedge 1)} \tag{3}$$

holds for all $(\mathbf{s},\mathbf{a})$.

We want to show that the good event $\mathcal{E} = \mathcal{E}_1 \cap \mathcal{E}_2$ occurs with high probability. The proof mostly follows from Bernstein's inequality in Lemma B.2 . Note that because $\widehat{P}(\mathbf{s},\mathbf{a}), \widehat{V}_i$ are not independent, we cannot straightforwardly apply Bernstein's inequality. We instead use the approach of Agarwal et al. (2020a) who, for each state s, partition the range of $\widehat{V}_i(\mathbf{s})$ within a modified s-absorbing MDP to create independence from $\widehat{P}$. The following lemma from Agarwal et al. (2020a) is a result of such analysis, and is slightly modified below to account for bounded returns of trajectories, *i.e.*, $\widehat{V}_i(\mathbf{s}) \leq 1$:

Lemma B.5 (Lemma 9, Agarwal et al. (2020a)). *For any iteration t, state-action $(\mathbf{s},\mathbf{a}) \in \mathcal{S} \times \mathcal{A}$ such that $n(\mathbf{s},\mathbf{a}) \geq 1$, and $\delta > 0$, we have*

$$\mathbb{P}\left(\left|(\widehat{P}(\mathbf{s},\mathbf{a}) - P(\mathbf{s},\mathbf{a})) \cdot \widehat{V}_i\right| > \sqrt{\frac{\mathbb{V}(\widehat{P}(\mathbf{s},\mathbf{a}), \widehat{V}_i)\iota}{n(\mathbf{s},\mathbf{a})}} + \frac{\iota}{n(\mathbf{s},\mathbf{a})}\right) \leq \delta \,.$$

Using this, we can show that $\mathcal{E}$ occurs with high probability:

Lemma B.6. $\mathbb{P}(\mathcal{E}) \geq 1 - 2|\mathcal{S}||\mathcal{A}|m\delta$.

Proof. For each i and $(\mathbf{s},\mathbf{a})$, if $n(\mathbf{s},\mathbf{a}) \leq 1$, then equation 2 and equation 3 hold trivially. For $n(\mathbf{s},\mathbf{a}) \geq 2$, we have from Lemma B.5 that

$$\mathbb{P}\left(\left|(\widehat{P}(\mathbf{s},\mathbf{a}) - P(\mathbf{s},\mathbf{a})) \cdot \widehat{V}_i\right| > \sqrt{\frac{\mathbb{V}(\widehat{P}(\mathbf{s},\mathbf{a}), \widehat{V}_i)\iota}{n(\mathbf{s},\mathbf{a})}} + \frac{\iota}{n(\mathbf{s},\mathbf{a})}\right) \leq \delta \,.$$

Similarly, we can use Lemma B.3 to derive

$$\begin{aligned}
&\mathbb{P}\left(|\widehat{r}(\mathbf{s},\mathbf{a}) - r(\mathbf{s},\mathbf{a})| > \sqrt{\frac{\widehat{r}(\mathbf{s},\mathbf{a})\iota}{n(\mathbf{s},\mathbf{a})}} + \frac{\iota}{n(\mathbf{s},\mathbf{a})}\right) \\
&\leq \mathbb{P}\left(|\widehat{r}(\mathbf{s},\mathbf{a}) - r(\mathbf{s},\mathbf{a})| > \sqrt{\frac{\widehat{\mathrm{Var}}(\widehat{r}(\mathbf{s},\mathbf{a}))\iota}{2(n(\mathbf{s},\mathbf{a})-1)}} + \frac{\iota}{2(n(\mathbf{s},\mathbf{a})-1)}\right) \leq \delta \,,
\end{aligned}$$

where we use that $\widehat{\mathrm{Var}}(\widehat{r}(\mathbf{s},\mathbf{a})) \leq \widehat{r}(\mathbf{s},\mathbf{a})$ for $[0,1]$ rewards, and with slight abuse of notation, let ι capture all constant factors. Taking the union bound over all i and $(\mathbf{s},\mathbf{a})$ yields the desired result. □

Now, we can prove that our value estimates are indeed pessimistic.

Lemma B.7 (Pessimism Guarantee)**.** *On event $\mathcal{E}$, we have that $\widehat{V}_i(\mathbf{s}) \leq V^{\widehat{\pi}^*}(\mathbf{s}) \leq V^*(\mathbf{s})$ for any iteration $i \in [m]$ and state $\mathbf{s} \in \mathcal{S}$.*

Proof. We aim to prove the following for any i and s: $\widehat{V}_{i-1}(\mathbf{s}) \leq \widehat{V}_i(\mathbf{s}) \leq V^{\widehat{\pi}^*}(\mathbf{s}) \leq V^*(\mathbf{s})$. We prove the claims one by one.

$\widehat{V}_{i-1}(\mathbf{s}) \leq \widehat{V}_i(\mathbf{s})$: This is directly implied by the monotonic update of our algorithm.

$\widehat{V}_i(\mathbf{s}) \leq V^{\widehat{\pi}^*}(\mathbf{s})$: We will prove this via induction. We have that this holds for $\widehat{V}_0$ trivially. Assume it holds for $t-1$, then we have

$$\begin{aligned}
V^{\widehat{\pi}^*}(\mathbf{s}) &\geq \mathbb{E}_{a\sim\widehat{\pi}^*(\cdot|s)}\left[r(\mathbf{s},\mathbf{a}) + \gamma P(\mathbf{s},\mathbf{a})\cdot \widehat{V}_{i-1}\right] \\
&\geq \mathbb{E}_a\left[\widehat{r}(\mathbf{s},\mathbf{a}) - b_i(\mathbf{s},\mathbf{a}) + \gamma \widehat{P}(\mathbf{s},\mathbf{a})\cdot \widehat{V}_{t-1}\right] + \\
&\qquad \mathbb{E}_a\left[b_i(\mathbf{s},\mathbf{a}) - (\widehat{r}(s,a) - r(\mathbf{s},\mathbf{a})) - \gamma(\widehat{P}(\mathbf{s},\mathbf{a}) - P(\mathbf{s},\mathbf{a}))\cdot \widehat{V}_{i-1}\right] \\
&\geq \widehat{V}_t(\mathbf{s})\,,
\end{aligned}$$

where we use that

$$\begin{aligned}
b_i(\mathbf{s},\mathbf{a}) &= \sqrt{\frac{\mathbb{V}(\widehat{P}(\mathbf{s},\mathbf{a}),\widehat{V}_{i-1})\iota}{(n(\mathbf{s},\mathbf{a})\wedge 1)}} + \sqrt{\frac{\widehat{r}(\mathbf{s},\mathbf{a})\iota}{(n(\mathbf{s},\mathbf{a})\wedge 1)}} + \frac{\iota}{(n(\mathbf{s},\mathbf{a})\wedge 1)} \\
&\geq (\widehat{r}(s,a) - r(\mathbf{s},\mathbf{a})) + \gamma(\widehat{P}(\mathbf{s},\mathbf{a}) - P(\mathbf{s},\mathbf{a}))\cdot \widehat{V}_{i-1}
\end{aligned}$$

under event $\mathcal{E}$.

Finally, the claim of $V^{\widehat{\pi}^*}(\mathbf{s}) \leq V^*(\mathbf{s})$ is trivial, which completes the proof of our pessimism guarantee. □

B.2.3 Value Difference Lemma

Now, we are ready to derive the performance guarantee from Theorem 4.2. The following lemma is a bound on the estimation error of our pessimistic Q-values.

Lemma B.8. *On event $\mathcal{E}$, the following holds for any $i \in [m]$ and $(\mathbf{s},\mathbf{a}) \in \mathcal{S}\times\mathcal{A}$:*

$$Q^*(\mathbf{s},\mathbf{a}) - \widehat{Q}_i(\mathbf{s},\mathbf{a}) \leq \gamma P(\mathbf{s},\mathbf{a})\cdot(Q^*(\cdot;\pi^*) - \widehat{Q}_{i-1}(\cdot;\pi^*)) + 2b_i(\mathbf{s},\mathbf{a})\,, \tag{4}$$

where $f(\cdot;\pi)$ satisfies $f(s;\pi) = \sum_{\mathbf{a}}\pi(\mathbf{a}|\mathbf{s})f(\mathbf{s},\mathbf{a})$.

Proof. We have,

$$\begin{aligned}
&Q^*(\mathbf{s},\mathbf{a}) - \widehat{Q}_i(\mathbf{s},\mathbf{a}) \\
&\quad= r(\mathbf{s},\mathbf{a}) + \gamma P(\mathbf{s},\mathbf{a})\cdot V^* - (\widehat{r}(\mathbf{s},\mathbf{a}) - b_i(\mathbf{s},\mathbf{a}) + \gamma\widehat{P}(\mathbf{s},\mathbf{a})\cdot\widehat{V}_{t-1}) \\
&\quad= b_i(\mathbf{s},\mathbf{a}) + r(\mathbf{s},\mathbf{a}) - \widehat{r}(\mathbf{s},\mathbf{a}) + \gamma P(\mathbf{s},\mathbf{a})\cdot(V^* - \widehat{V}_{t-1}) + \gamma(P(\mathbf{s},\mathbf{a}) - \widehat{P}(\mathbf{s},\mathbf{a}))\cdot\widehat{V}_{t-1} \\
&\quad\leq \gamma P(\mathbf{s},\mathbf{a})\cdot(V^* - \widehat{V}_{t-1}) + 2b_i(\mathbf{s},\mathbf{a}) \\
&\quad\leq \gamma P(\mathbf{s},\mathbf{a})\cdot(Q^*(\cdot;\pi^*) - \widehat{Q}_{t-1}(\cdot;\pi^*)) + 2b_i(\mathbf{s},\mathbf{a})\,.
\end{aligned}$$

The first inequality is due by definition of $\mathcal{E}$ and the second is because $\widehat{V}_{t-1} \geq \max_a \widehat{Q}_{t-1}(\cdot,a) \geq \widehat{Q}_i(\cdot,\pi^*)$. □

By recursively applying Lemma B.8, we can derive the following value difference lemma:

Lemma B.9 (Value Difference Lemma)**.** *On event $\mathcal{E}$, at any iteration $i \in [m]$, we have*

$$J(\pi^*) - J(\widehat{\pi}^*) \leq \gamma^i + 2\sum_{t=1}^{i}\sum_{(\mathbf{s},\mathbf{a})}\gamma^{i-t}d^*_{i-t}(\mathbf{s},\mathbf{a})b_t(\mathbf{s},\mathbf{a})\,, \tag{5}$$

where $d^*_t(\mathbf{s},\mathbf{a}) = \mathbb{P}(s_t = \mathbf{s}, \mathbf{a}_t = a;\pi^*)$.

Proof. We have,

$$J(\pi^*) - J(\widehat{\pi}^*) = \mathbb{E}_\rho\left[V^*(\mathbf{s}) - V^{\widehat{\pi}^*}(\mathbf{s})\right] \le \mathbb{E}_\rho\left[V^*(\mathbf{s}) - \widehat{V}_i(\mathbf{s})\right] \le \rho(Q^*(\cdot;\pi^*) - \widehat{Q}_i(\cdot;\pi^*))$$

where we use Lemma B.7 in the first inequality. As shorthand, let $P^\pi \in \mathbb{R}^{(\mathcal{S}\times\mathcal{A})\times(\mathcal{S}\times\mathcal{A})}$ where $P^\pi(\mathbf{s},\mathbf{a},s',a') = P(s'|\mathbf{s},\mathbf{a})\pi(a'|s')$ be the transition matrix for policy π. Now, we can apply Lemma B.8 recursively to derive

$$\begin{aligned}
\rho^{\pi^*}(Q^* - \widehat{Q}_i) &\le \rho^{\pi^*}\left(\gamma P^{\pi^*}(Q - \widehat{Q}_{i-1}) + 2b_i\right) \\
&\le \rho^{\pi^*}\left(\gamma P^{\pi^*}\left(\gamma P^{\pi^*}(Q^* - \widehat{Q}_{i-2}) + 2b_{i-1}\right) + 2b_i\right) \\
&\le \dots \\
&\le \rho^{\pi^*}(\gamma P^{\pi^*})^i(Q^* - \widehat{Q}_0) + 2\sum_{t=1}^{i}\rho^{\pi^*}(\gamma P^{\pi^*})^{i-t}b_t \\
&\le \gamma^i\mathbf{1} + 2\sum_{t=1}^{i}\gamma^{i-t}d^*_{i-t}b_t
\end{aligned}$$

where we use that $d^*_t = \rho^{\pi^*}(P^{\pi^*})^t$. This yields the desired result. □

Now, we are ready to bound the desired quantity $\mathsf{SubOpt}(\widehat{\pi}^*) = \mathbb{E}_\mathcal{D}\left[J(\pi^*) - J(\widehat{\pi}^*)\right]$. We have

$$\begin{aligned}
\mathbb{E}_\mathcal{D}\left[J(\pi^*) - J(\widehat{\pi}^*)\right] &= \mathbb{E}_\mathcal{D}\left[\sum_s \rho(\mathbf{s})(V^*(\mathbf{s}) - V^{\widehat{\pi}^*}(\mathbf{s}))\right] \\
&= \underbrace{\mathbb{E}_\mathcal{D}\left[1\{\bar{\mathcal{E}}\}\sum_s \rho(\mathbf{s})(V^*(\mathbf{s}) - V^{\widehat{\pi}^*}(\mathbf{s}))\right]}_{:=\Delta_1} \\
&\quad + \underbrace{\mathbb{E}_\mathcal{D}\left[1\{\exists \mathbf{s}\in\mathcal{S},\ n(\mathbf{s},\pi^*(\mathbf{s})) = 0\}\sum_s \rho(\mathbf{s})(V^*(\mathbf{s}) - V^{\widehat{\pi}^*}(\mathbf{s}))\right]}_{:=\Delta_2} \\
&\quad + \underbrace{\mathbb{E}_\mathcal{D}\left[1\{\forall \mathbf{s}\in\mathcal{S},\ n(\mathbf{s},\pi^*(\mathbf{s})) > 0\}\,1\{\mathcal{E}\}\sum_s \rho(\mathbf{s})(V^*(\mathbf{s}) - V^{\widehat{\pi}^*}(\mathbf{s}))\right]}_{:=\Delta_3}.
\end{aligned} \tag{6}$$

We bound each term individually. The first is bounded as $\Delta_1 \le \mathbb{P}\left(\bar{\mathcal{E}}\right) \le 2|\mathcal{S}||\mathcal{A}|m\delta \le \frac{\iota}{N}$ for choice of $\delta = \frac{1}{2|\mathcal{S}||\mathcal{A}|HN}$.

B.2.4 Bound on Δ_2

For the second term, we have

$$\begin{aligned}
\Delta_2 &\le \sum_s \rho(\mathbf{s})\mathbb{E}_\mathcal{D}\left[1\{n(\mathbf{s},\pi^*(\mathbf{s})) = 0\}\right] \\
&\le H\sum_{\mathbf{s}} d^*(\mathbf{s},\pi^*(\mathbf{s}))\mathbb{E}_\mathcal{D}\left[1\{n(\mathbf{s},\pi^*(\mathbf{s})) = 0\}\right] \\
&\le C^*H\sum_{\mathbf{s}}\mu(\mathbf{s},\pi^*(\mathbf{s}))(1-\mu(\mathbf{s},\pi^*(\mathbf{s})))^N \\
&\le \frac{4C^*|\mathcal{S}|H}{9N},
\end{aligned}$$

where we use that $\rho(\mathbf{s}) \le Hd^*(\mathbf{s},\pi^*(\mathbf{s}))$, and that $\max_{p\in[0,1]} p(1-p)^N \le \frac{4}{9N}$.

B.2.5 BOUND ON Δ_3

What remains is bounding the last term, which we know from Lemma B.9 is bounded by

$$\Delta_3 \leq \frac{1}{N} + 2\mathbb{E}_{\mathcal{D}}\left[1\{\forall \mathbf{s} \in \mathcal{S},\ n(\mathbf{s}, \pi^*(\mathbf{s})) > 0\} \sum_{t=0}^{m} \sum_{(\mathbf{s},\mathbf{a})} \gamma^{m-t} d^*_{m-t}(\mathbf{s},\mathbf{a}) b_t(\mathbf{s},\mathbf{a})\right],$$

where we use that $\gamma^m \leq \frac{1}{N}$ for $m = H \log N$. Recall that $b_t(\mathbf{s},\mathbf{a})$ is given by

$$b_t(\mathbf{s},\mathbf{a}) = \sqrt{\frac{\mathbb{V}(\widehat{P}(\mathbf{s},\mathbf{a}), \widehat{V}_{t-1})\iota}{n(\mathbf{s},\mathbf{a})}} + \sqrt{\frac{\widehat{r}(\mathbf{s},\mathbf{a})\iota}{n(\mathbf{s},\mathbf{a})}} + \frac{\iota}{n(\mathbf{s},\mathbf{a})}$$

We can bound the summation of each term separately. For the third term we have,

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}\left[\sum_{t=1}^{m} \sum_{(\mathbf{s},\mathbf{a})} \gamma^{m-t} d^*_{m-t}(\mathbf{s},\mathbf{a}) \frac{\iota}{n(\mathbf{s},\mathbf{a})}\right] &\leq \sum_{t=0}^{m-1} \sum_{(\mathbf{s},\mathbf{a})} \gamma^t d^*_t(\mathbf{s},\mathbf{a}) \mathbb{E}_{\mathcal{D}}\left[\frac{\iota}{n(\mathbf{s},\mathbf{a})}\right] \\
&\leq \sum_{\mathbf{s}} \sum_{t=0}^{\infty} \gamma^t d^*_t(\mathbf{s}, \pi^*(\mathbf{s})) \frac{\iota}{N\mu(\mathbf{s}, \pi^*(\mathbf{s}))} \\
&\leq \frac{H\iota}{N} \sum_{\mathbf{s}} \left((1-\gamma)\sum_{t=0}^{\infty} \gamma^t d^*_t(\mathbf{s}, \pi^*(\mathbf{s}))\right) \frac{1}{\mu(\mathbf{s}, \pi^*_h(\mathbf{s}))} \\
&\leq \frac{C^*|\mathcal{S}|H\iota}{N}.
\end{aligned}$$

Here we use Jensen's inequality and that $(1-\gamma)\sum_{t=1}^{\infty} \gamma^t d^*_t(\mathbf{s},\mathbf{a}) \leq C^* \mu(\mathbf{s},\mathbf{a})$ for any $(\mathbf{s},\mathbf{a})$. For the second term, we similarly have

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}}\left[\sum_{t=1}^{m} \sum_{(\mathbf{s},\mathbf{a})} \gamma^{m-t} d^*_{m-t}(\mathbf{s},\mathbf{a}) \sqrt{\frac{\widehat{r}(\mathbf{s},\mathbf{a})\iota}{n(\mathbf{s},\mathbf{a})}}\right] \\
&\leq \mathbb{E}_{\mathcal{D}}\left[\sqrt{\sum_{t=1}^{m} \sum_{(\mathbf{s},\mathbf{a})} \gamma^{m-t} d^*_{m-t}(\mathbf{s},\mathbf{a}) \frac{\iota}{n(\mathbf{s},\mathbf{a})}}\right] \sqrt{\sum_{t=1}^{m} \sum_{(\mathbf{s},\mathbf{a})} \gamma^{m-t} d^*_{m-t}(\mathbf{s},\mathbf{a}) \widehat{r}(\mathbf{s},\mathbf{a})} \\
&\leq \sqrt{\frac{C^*|\mathcal{S}|H\iota}{N}},
\end{aligned}$$

where we use Cauchy-Schwarz, then Condition 3.2 to bound the total estimated reward. Finally, we consider the first term of $b_t(\mathbf{s},\mathbf{a})$

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}}\left[\sum_{t=1}^{m} \sum_{(\mathbf{s},\mathbf{a})} \gamma^{m-t} d^*_{m-t}(\mathbf{s},\mathbf{a}) \sqrt{\frac{\mathbb{V}(\widehat{P}(\mathbf{s},\mathbf{a}), \widehat{V}_{t-1})\iota}{n(\mathbf{s},\mathbf{a})}}\right] \\
&\leq \mathbb{E}_{\mathcal{D}}\left[\sqrt{\sum_{t=1}^{m} \sum_{(\mathbf{s},\mathbf{a})} \gamma^{m-t} d^*_{m-t}(\mathbf{s},\mathbf{a}) \frac{\iota}{n(\mathbf{s},\mathbf{a})}}\right] \sqrt{\sum_{t=1}^{m} \sum_{(\mathbf{s},\mathbf{a})} \gamma^{m-t} d^*_{m-t}(\mathbf{s},\mathbf{a}) \mathbb{V}(\widehat{P}(\mathbf{s},\mathbf{a}), \widehat{V}_{t-1})} \\
&\leq \sqrt{\frac{C^*|\mathcal{S}|H\iota}{N}} \sqrt{\sum_{t=1}^{m} \sum_{(\mathbf{s},\mathbf{a})} \gamma^{m-t} d^*_{m-t}(\mathbf{s},\mathbf{a}) \mathbb{V}(\widehat{P}(\mathbf{s},\mathbf{a}), \widehat{V}_{t-1})}.
\end{aligned}$$

Similar to what was done in Zhang et al. (2020); Ren et al. (2021) for finite-horizon MDPs, we can bound this term using variance recursion for infinite-horizon ones. Define

$$f(i) := \sum_{t=1}^{\infty} \sum_{(\mathbf{s},\mathbf{a})} \gamma^{m-t} d^*_{m-t}(\mathbf{s},\mathbf{a}) \mathbb{V}(\widehat{P}(\mathbf{s},\mathbf{a}), (\widehat{V}_{t-1})^{2^i}). \tag{7}$$

Using Lemma 3 of Ren et al. (2021) for the infinite-horizon case, we have the following recursion:

$$f(i) \leq \sqrt{\frac{C^*|\mathcal{S}|H\iota}{N} f(i+1)} + \frac{C^*|\mathcal{S}|H\iota}{N} + 2^{i+1}(\Phi+1)\,,$$

where

$$\Phi := \sqrt{\frac{C^*|\mathcal{S}|H\iota}{N}} \sqrt{\sum_{t=1}^{m} \sum_{(\mathbf{s},\mathbf{a})} \gamma^{m-t} d^*_{m-t}(\mathbf{s},\mathbf{a}) \mathbb{V}(\widehat{P}(\mathbf{s},\mathbf{a}), \widehat{V}_{t-1})} + \frac{C^*|\mathcal{S}|H\iota}{N} \tag{8}$$

Using Lemma B.4, we can bound $f(0) = \mathcal{O}\left(\frac{C^*|\mathcal{S}|H\iota}{N} + \Phi + 1\right)$. Using that for constant c,

$$\begin{aligned}
\Phi &= \sqrt{\frac{C^*|\mathcal{S}|H\iota}{N} f(0)} + \frac{C^*|\mathcal{S}|H\iota}{N} \\
&\leq \sqrt{\frac{C^*|\mathcal{S}|H\iota}{N}\left(\frac{cC^*|\mathcal{S}|H\iota}{N} + c\Phi + c\right)} + \frac{C^*|\mathcal{S}|H\iota}{N} \\
&\leq \frac{c\Phi}{2} + \frac{2cC^*|\mathcal{S}|H\iota}{N} + \frac{c}{2}
\end{aligned}$$

we have that

$$\Phi \leq c + \frac{4cC^*|\mathcal{S}|H\iota}{N}\,.$$

Substituting this back into the inequality for Φ yields,

$$\Phi = \mathcal{O}\left(\sqrt{\frac{C^*|\mathcal{S}|H\iota}{N}} + \frac{C^*|\mathcal{S}|H\iota}{N}\right)$$

Finally, we can bound

$$\Delta_3 \leq \sqrt{\frac{C^*|\mathcal{S}|H\iota}{N}} + \frac{C^*|\mathcal{S}|H\iota}{N}\,.$$

Combining the bounds for the three terms yields the desired result.

B.3 Proof of Corollary 4.1

The proof of Corollary 4.1 mostly relies on the existing machinery in Appendix B.2. In this section, we will only point out the differences and defer details to the comprehensive proof in Appendix B.2. Recall that $\mathcal{C}$ is the set of critical states, and from Definition 4.1, that all $\mathbf{s} \in \mathcal{S} \setminus \mathcal{C}$ satisfy having negligible advantage, *i.e.*, $Q^*(\mathbf{s}, \pi^*(\mathbf{s})) - Q^*(\mathbf{s}, \mathbf{a}) \leq \varepsilon/H$ for any suboptimal action a.

Recall that for any Q, π, we use the notation $Q(s;\pi) = \mathbb{E}_{a\sim\pi}\left[Q(\mathbf{s}, a)\right]$. The main difference between this proof and the one for Theorem 4.2 is that under the critical point structure outlined in Section 4.2, we can handle critical and non-critical states differently to derive a tighter bound. Namely, we bound the suboptimality using:

$$\begin{aligned}
J(\pi^*) - J(\widehat{\pi}^*) &= H \sum_{\underline{\mathbf{s}}\in\mathcal{S}} d^{\widehat{\pi}^*}(\mathbf{s}) \left(Q^*(s;\pi^*) - Q^*(s;\widehat{\pi}^*)\right) \\
&\leq H \sum_{\mathbf{s}\in\mathcal{C}} d^{\widehat{\pi}^*}(\mathbf{s}) \left(Q^*(s;\pi^*) - Q^*(s;\widehat{\pi}^*)\right) + H \sum_{\mathbf{s}\notin\mathcal{C}} d^{\widehat{\pi}^*}(\mathbf{s})\, \frac{\varepsilon}{H} \\
&= H \sum_{\mathbf{s}\in\mathcal{C}} d^{\widehat{\pi}^*}(\mathbf{s}) \left(Q^*(s;\pi^*) - Q^*(s;\widehat{\pi}^*)\right) + \varepsilon \\
&= H \sum_{\mathbf{s}\in\mathcal{C}} d^{*}(\mathbf{s}) \left(Q^{\widehat{\pi}^*}(s;\pi^*) - Q^{\widehat{\pi}^*}(s;\widehat{\pi}^*)\right) + \varepsilon\,,
\end{aligned}$$

where we use the performance difference lemma, then condition on whether a state is critical. We propose an alternative interpretation for the bound. The first term is the value difference under a

modified MDP where the advantage for all states that are non-critical is 0. One way to construct such an MDP is to take each $s \notin \mathcal{C}$ and modify the reward and transitions so that $r(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \pi^*(\mathbf{s}))$, $P(\mathbf{s}, \mathbf{a}) = P(\mathbf{s}, \pi^*(\mathbf{s}))$ for all actions. Let us denote the performance function for this modified MDP be $J_{\mathcal{C}}$, then we have

$$J(\pi^*) - J(\widehat{\pi}^*) \leq J_{\mathcal{C}}(\pi^*) - J_{\mathcal{C}}(\widehat{\pi}^*) + \varepsilon$$

The first term can be bounded as was done in Appendix B.2. Namely, we can show the following equivalent of Lemma B.9:

$$J_{\mathcal{C}}(\pi^*) - J_{\mathcal{C}}(\widehat{\pi}^*) \leq \gamma^i + 2\sum_{t=1}^{i}\sum_{\mathbf{s}\in\mathcal{C}} \gamma^{i-t} d^*_{i-t}(\mathbf{s}, \pi^*(\mathbf{s})) b_t(\mathbf{s}, \pi^*(\mathbf{s}))\,.$$

Bounding this for $i = H\iota$ can be done exactly as in was done in Appendix B.2.4 and B.2.5, except with the dependence on all states $\mathcal{S}$ replaced by $\mathcal{C}$ only the critical ones. We get the following bound:

$$J_{\mathcal{C}}(\pi^*) - J_{\mathcal{C}}(\widehat{\pi}^*) \leq \sqrt{\frac{C^*|\mathcal{C}|H\iota}{N}} + \frac{C^*|\mathcal{C}|H\iota}{N}$$

Combining the above result with Condition 4.1, which bounds $|\mathcal{C}| \leq p_c|\mathcal{S}|$, completes the proof.

B.4 Proof of Corollary 4.2

The proof of Corollary 4.2 is a slight modification of the one for Theorem 4.2. For brevity, we will point out the parts of the proof that change, and simply defer to the proof in Appendix B.2 for parts that are similar. Recall the decomposition for suboptimality in equation 6, which we restate below:

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}\left[J(\pi^*) - J(\widehat{\pi}^*)\right] = &\underbrace{\mathbb{E}_{\mathcal{D}}\left[1\{\bar{\mathcal{E}}\}\sum_s \rho(\mathbf{s})(V^*(\mathbf{s}) - V^{\widehat{\pi}^*}(\mathbf{s}))\right]}_{\Delta_1} \\
&+ \underbrace{\mathbb{E}_{\mathcal{D}}\left[1\{\exists \mathbf{s}\in\mathcal{S},\ n(\mathbf{s}, \pi^*(\mathbf{s})) = 0\}\sum_s \rho(\mathbf{s})(V^*(\mathbf{s}) - V^{\widehat{\pi}^*}(\mathbf{s}))\right]}_{\Delta_2} \\
&+ \underbrace{\mathbb{E}_{\mathcal{D}}\left[1\{\forall \mathbf{s}\in\mathcal{S},\ n(\mathbf{s}, \pi^*(\mathbf{s})) > 0\}\,1\{\mathcal{E}\}\sum_s \rho(\mathbf{s})(V^*(\mathbf{s}) - V^{\widehat{\pi}^*}(\mathbf{s}))\right]}_{\Delta_3}\,.
\end{aligned}$$

Δ_1 is bounded by $\frac{\iota}{N}$ as before.

B.4.1 Bound on Δ_2

The bound for Δ_2 changes slightly from Appendix B.2.4 due to accounting for the lower-bound on $\mu(\mathbf{s}, \mathbf{a}) \geq b \geq \frac{\log H}{N}$. We have

$$\begin{aligned}
\Delta_2 &\leq \sum_{\mathbf{s}} \rho(\mathbf{s})\mathbb{E}_{\mathcal{D}}\left[1\{n(\mathbf{s}, \pi^*(\mathbf{s})) = 0\}\right] \\
&\leq H\sum_{\mathbf{s}} d^*(\mathbf{s}, \pi^*(\mathbf{s}))\mathbb{E}_{\mathcal{D}}\left[1\{n(\mathbf{s}, \pi^*(\mathbf{s})) = 0\}\right] \\
&\leq H\sum_{\mathbf{s}} d^*(\mathbf{s}, \pi^*(\mathbf{s}))1\left\{d^*(\mathbf{s}, \pi^*(\mathbf{s})) \leq \frac{b}{H}\right\} + H\sum_{\mathbf{s}} d^*(\mathbf{s}, \pi^*(\mathbf{s}))\mathbb{E}_{\mathcal{D}}\left[1\{n(\mathbf{s}, \pi^*(\mathbf{s})) = 0\}\right] \\
&\leq |\mathcal{S}|c + C^*H\sum_{\mathbf{s}} \mu(\mathbf{s}, \pi^*(\mathbf{s}))(1 - \mu(\mathbf{s}, \pi^*(\mathbf{s})))^N \\
&\leq |\mathcal{S}|b + \frac{C^*|\mathcal{S}|\iota}{N}\,,
\end{aligned}$$

where we use that $\rho(\mathbf{s}) \leq Hd^*(\mathbf{s}, \pi^*(\mathbf{s}))$, and that

$$\max_{p\in[\frac{\log H}{N}, 1]} p(1-p)^N \leq \frac{\log H}{N}\left(1 - \frac{\log H}{N}\right)^N \leq \frac{\log H}{HN}\,.$$

B.4.2 BOUND ON Δ_3

Due to the lower bound on $\mu(\mathbf{s}, \mathbf{a}) \geq b$, we can instead bound,

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}\left[\sum_{t=1}^{m}\sum_{(\mathbf{s},\mathbf{a})}\gamma^{m-t}d^*_{m-t}(\mathbf{s},\mathbf{a})\,\frac{\iota}{n(\mathbf{s},\mathbf{a})}\right] &\leq \sum_{t=0}^{m-1}\sum_{(\mathbf{s},\mathbf{a})}\gamma^t d^*_t(\mathbf{s},\mathbf{a})\mathbb{E}_{\mathcal{D}}\left[\frac{\iota}{n(\mathbf{s},\mathbf{a})}\right] \\
&\leq \sum_{\mathbf{s}}\sum_{t=0}^{\infty}\gamma^t d^*_t(\mathbf{s},\pi^*(\mathbf{s}))\,\frac{\iota}{N\mu(\mathbf{s},\pi^*(\mathbf{s}))} \\
&\leq 1\left\{d^*(\mathbf{s},\mathbf{a}) \leq \frac{b}{H}\right\} H\sum_{\mathbf{s}}\left((1-\gamma)\sum_{t=0}^{\infty}\gamma^t d^*_t(\mathbf{s},\pi^*(\mathbf{s}))\right) + \\
&\quad \frac{H\iota}{Nc}\sum_{\mathbf{s}}\left((1-\gamma)\sum_{t=0}^{\infty}\gamma^t d^*_t(\mathbf{s},\pi^*(\mathbf{s}))\right) \\
&\leq b + \frac{H\iota}{bN}.
\end{aligned}$$

The analysis for bounding Δ_3 proceeds exactly as in Appendix B.2.5 but using the new bound. Namely, we end up with the recursion

$$f(i) \leq \sqrt{\frac{H\iota}{bN} + b}\,\sqrt{f(i+1)} + \frac{H\iota}{bN} + b + 2^{i+1}(\Phi + 1)\,,$$

where

$$\Phi := \sqrt{\frac{H\iota}{bN} + b}\,\sqrt{\sum_{t=1}^{m}\sum_{(\mathbf{s},\mathbf{a})}\gamma^{m-t}d^*_{m-t}(\mathbf{s},\mathbf{a})\mathbb{V}(\widehat{P}(\mathbf{s},\mathbf{a}),\widehat{V}_{t-1})} + \frac{H\iota}{Nb} + b\,.$$

Using Lemma B.4 and proceeding as in Appendix B.2.5 yields the bound

$$\Delta_3 \leq \sqrt{\frac{H\iota}{bN}} + \frac{H\iota}{bN} + \sqrt{b\iota}\,.$$

Combining the new bounds for Δ_2, Δ_3 results in the bound in the Corollary 4.2.

B.5 PROOF OF THEOREM 4.4

The proof of Theorem 4.4 builds on analysis by Agarwal et al. (2021a) that we apply to policies with a softmax parameterization, which we define below.

Definition B.1 (Softmax parameterization)**.** *For a given* $\theta \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$, $\pi_\theta(\mathbf{a}|\mathbf{s}) = \frac{\exp(\theta_{\mathbf{s},\mathbf{a}})}{\sum_{\mathbf{a}'}\exp(\theta_{\mathbf{s},\mathbf{a}'})}$.

We consider generalized BC algorithms that perform advantage-weighted policy improvement for k improvement steps. A BC algorithm with k-step policy improvement is defined as follows:

Definition B.2 (BC with k-step policy improvement)**.** *Let* $\widehat{A}^k(\mathbf{s},\mathbf{a})$ *denote the advantage of action* $\mathbf{a}$ *at state* $\mathbf{s}$ *under a given policy* $\widehat{\pi}_k$*, where the policy* $\widehat{\pi}^k(\mathbf{a}|\mathbf{s})$ *is defined via the recursion:*

$$\widehat{\pi}^{k+1}(\mathbf{a}|\mathbf{s}) := \widehat{\pi}^k(\mathbf{a}|\mathbf{s})\frac{\exp(\eta H\widehat{A}^k(\mathbf{s},\mathbf{a}))}{\mathbb{Z}_k(\mathbf{s})},$$

starting from $\widehat{\pi}^0(\mathbf{a}|\mathbf{s}) = \widehat{\pi}_\beta$*. Then, BC with* k*-step policy improvement returns* $\widehat{\pi}^k$*.*

This advantage weighted update is utilized in practical works such as Brandfonbrener et al. (2021), which first estimates the Q-function of the behavior policy using the offline dataset, i.e, $\widehat{Q}^0(\mathbf{s},\mathbf{a})$, and then computes $\widehat{\pi}^1$ as the final policy returned by the algorithm. To understand the performance difference between multiple values of k, we first utilize essentially Lemma 5 from Agarwal et al. (2021a), which we present below for completeness:

Lemma B.10 (Lower bound on policy improvement in the empirical MDP, $\widehat{M}$)**.** *The iterates $\widehat{\pi}^k$ generated by k-steps of policy improvement, for any initial state distributions $\rho_0(\mathbf{s})$ satisfy the following lower-bound on improvement:*

$$\widehat{J}(\widehat{\pi}^{k+1}) - \widehat{J}(\widehat{\pi}^k) := \mathbb{E}_{\mathbf{s}_0\sim\rho_0}\left[\widehat{V}^{\widehat{\pi}_{k+1}}(\mathbf{s}_0)\right] - \mathbb{E}_{\mathbf{s}_0\sim\rho_0}\left[\widehat{V}^{\widehat{\pi}_k}(\mathbf{s}_0)\right] \geq \frac{1}{\eta H}\mathbb{E}_{\mathbf{s}_0\sim\rho_0}\log \mathbb{Z}_t(\mathbf{s}_0). \quad (9)$$

Proof. We utilize the performance difference lemma in the empirical MDP to show this:

$$\begin{aligned}
\widehat{J}(\widehat{\pi}^{k+1}) - \widehat{J}(\widehat{\pi}^k) &= H\mathbb{E}_{\mathbf{s}\sim d^{\widehat{\pi}_{k+1}}}\left[\sum_{\mathbf{a}}\widehat{\pi}_{k+1}(\mathbf{a}|\mathbf{s})\widehat{A}^k(\mathbf{s},\mathbf{a})\right] \\
&= \frac{1}{\eta}\mathbb{E}_{\mathbf{s}\sim d^{\widehat{\pi}_{k+1}}}\left[\sum_{\mathbf{a}}\widehat{\pi}^{k+1}(\mathbf{a}|\mathbf{s})\log\frac{\widehat{\pi}^{k+1}(\mathbf{a}|\mathbf{s})\mathbb{Z}_k(\mathbf{s})}{\widehat{\pi}^k(\mathbf{a}|\mathbf{s})}\right] \\
&= \frac{1}{\eta}\mathbb{E}_{\mathbf{s}\sim d^{\widehat{\pi}_{k+1}}}\left[\mathrm{D}_{\mathrm{KL}}(\widehat{\pi}^{k+1}(\cdot|\mathbf{s})||\widehat{\pi}^k(\cdot|\mathbf{s}))\right] + \frac{1}{\eta}\mathbb{E}_{\mathbf{s}\sim d^{\widehat{\pi}_{k+1}}}\left[\log\mathbb{Z}_k(\mathbf{s})\right] \\
&\geq \frac{1}{\eta}\mathbb{E}_{\mathbf{s}\sim d^{\widehat{\pi}_{k+1}}}\left[\log\mathbb{Z}_k(\mathbf{s})\right].
\end{aligned}$$

Finally, note that the final term $\log\mathbb{Z}_t(\mathbf{s})$ is always positive because of Jensen's inequality, and the fact that the expected advantage under a given policy is 0 for any MDP. □

Utilizing Lemma B.10, we can then lower bound the total improvement of the learned policy in the actual MDP as:

$$\begin{aligned}
J(\widehat{\pi}^k) - J(\widehat{\pi}^l) &\geq \underbrace{J(\widehat{\pi}^k) - \widehat{J}(\widehat{\pi}^k)}_{\text{(a)}} + \underbrace{\widehat{J}(\widehat{\pi}^k) - \widehat{J}(\widehat{\pi}^l)}_{\text{(b)}} - \underbrace{J(\widehat{\pi}^l) - \widehat{J}(\widehat{\pi}^l)}_{\text{(c)}} \\
&\geq \frac{1}{\eta}\sum_{j=l}^{k}\mathbb{E}_{\mathbf{s}\sim d^{\widehat{\pi}_{j+1}}}\left[\log\mathbb{Z}_j(\mathbf{s})\right] - \sqrt{\frac{C^* H\iota}{N}}
\end{aligned}$$

where the $\sqrt{C^* H\iota/N}$ guarantee for terms (a) and (c) arises under the conditions studied in Section 4.3.

Interpretation of Theorem 4.4. Theorem 4.4 says that if atleast k many updates can be made to the underlying empirical MDP, $\widehat{M}$, such that each update is non-trivially lower-bounded, i.e., $\mathbb{E}_{\mathbf{s}\sim d^{\widehat{\pi}}_{k+1}}\left[\log\mathbb{Z}_k(\mathbf{s})\right] \geq c_0 > 0$, then the performance improvement obtained by k-steps of policy improvement is bounded below by $kc_0/\eta - \mathcal{O}(\sqrt{H/N})$. This result indicates that if $k = \mathcal{O}(H)$ many high advantage policy updates are possible in a given empirical MDP, then the methods with that perform $\mathcal{O}(H)$ steps of policy improvement will attain higher performance than the counterparts that perform only one update.

This is typically the case in maze navigation-style environments, where $\mathcal{O}(H)$ many possible high-advantage updates are possible on the empirical MDP, especially by "stitching" parts of suboptimal trajectories to obtain a much better trajectory. Therefore, we expect that in offline RL problems where stitching is possible, offline RL algorithms will attain an improved performance compared to one or a few-steps of policy improvement.

C Guarantees for Policy-Constraint Offline RL

In this section, we analyze a policy-constraint offline algorithm (Levine et al., 2020) that constrains the policy to choose a safe set of actions by explicitly preventing action selection from previously unseen, low-density actions. The algorithm we consider builds upon the `MBS-PI` algorithm from Liu et al. (2020), which truncates Bellman backups and policy improvement steps from low-density, out-of-support state-action pairs. The algorithm is described in detail in Algorithm 2, but we provide a summary below. Let $\widehat{\mu}(\mathbf{s},\mathbf{a})$ denote the empirical state-action distribution and choose a constant b.

Then, let $\zeta(\mathbf{s},\mathbf{a}) = 1\{\widehat{\mu}(\mathbf{s},\mathbf{a}) \geq b\}$ be the indicator of high-density state-action tuples. The algorithm we analyze performs the following update until convergence:

$$\widehat{Q}^{\pi}_{\zeta}(\mathbf{s},\mathbf{a}) \leftarrow \widehat{r}(\mathbf{s},\mathbf{a}) + \gamma \sum_{(\mathbf{s}',\mathbf{a}')} \widehat{P}(\mathbf{s}'|\mathbf{s},\mathbf{a})\pi(\mathbf{a}'|\mathbf{s}')\zeta(\mathbf{s}',\mathbf{a}') \cdot \widehat{Q}^{\pi}_{\zeta}(\mathbf{s}',\mathbf{a}'), \quad \text{for all } (\mathbf{s},\mathbf{a}),$$

$$\widehat{\pi} \leftarrow \arg\max_{\pi} \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\left[\mathbb{E}_{\mathbf{a}\sim\pi'}\left[\zeta(\mathbf{s},\mathbf{a}) \cdot \widehat{Q}^{\pi}_{\zeta}(\mathbf{s},\mathbf{a})\right]\right],$$

In order to derive performance guarantees for this generic policy-constraint algorithm, we define the notion of a $\zeta-$covered policy following Liu et al. (2020) in Definition C.1. The total occupancy of all out-of-support state-action pairs (i.e., $(\mathbf{s},\mathbf{a})$ such that $\zeta(\mathbf{s},\mathbf{a}) = 0$) under a ζ-covered policy is bounded by a small constant U, which depends on the threshold b. Let π^*_{ζ} denote the best performing ζ-covered policy.

Definition C.1 (ζ-covered)**.** *π is called ζ-covered if $\sum_{(\mathbf{s},\mathbf{a})}(1-\zeta(\mathbf{s},\mathbf{a}))d^{\pi}(\mathbf{s},\mathbf{a}) \leq (1-\gamma)U(b)$.*

Equipped with this definition C.1, Lemma C.1 shows that the total value estimation error of any given $\zeta-$covered policy, π, $|J(\pi) - \widehat{J}_{\zeta}(\pi)|$ is upper bounded in expectation over the dataset

Lemma C.1 (Value estimation error of a ζ-covered policy)**.** *For any given ζ-covered policy π, under Condition 3.2, the estimation error $|J(\pi) - \widehat{J}_{\zeta}(\pi)|$ is bounded as:*

$$\mathbb{E}_{\mathcal{D}}\left[\left|J(\pi) - \widehat{J}_{\zeta}(\pi)\right|\right] \lesssim \sqrt{\frac{C^*|\mathcal{S}|H\iota}{N}} + \frac{C^*|\mathcal{S}|H\iota}{N} + U(b) \tag{10}$$

Proof. To prove this lemma, we consider the following decomposition of the policy performance estimate:

$$\begin{aligned}
&\left|J(\pi) - \widehat{J}_{\zeta}(\pi)\right| \\
&= \sum_{t=0}^{\infty}\sum_{(\mathbf{s},\mathbf{a})} \gamma^t d^{\pi}_t(\mathbf{s},\mathbf{a}) \left[\sum_{(\mathbf{s}',\mathbf{a}')} \left(\widehat{P}(\mathbf{s}'|\mathbf{s},\mathbf{a})\zeta(\mathbf{s}',\mathbf{a}') - P(\mathbf{s}'|\mathbf{s},\mathbf{a})\right) \cdot \widehat{Q}^{\pi}(\mathbf{s}',\mathbf{a}')\right] \\
&= \underbrace{\sum_{t=0}^{\infty}\sum_{(\mathbf{s},\mathbf{a})} \gamma^t d^{\pi}_t(\mathbf{s},\mathbf{a}) \sum_{(\mathbf{s}',\mathbf{a}')} (\widehat{P}(\mathbf{s}'|\mathbf{s},\mathbf{a}) - P(\mathbf{s}'|\mathbf{s},\mathbf{a})) \cdot \zeta(\mathbf{s}',\mathbf{a}') \cdot \pi(\mathbf{a}'|\mathbf{s}') \cdot \widehat{Q}(\mathbf{s}',\mathbf{a}')}_{\Delta_1\text{:bound using concentrability and variance recursion}} \\
&\qquad + \underbrace{\sum_{t=0}^{\infty} \gamma^t d^{\pi}_t(\mathbf{s},\mathbf{a}) \sum_{(\mathbf{s}',\mathbf{a}')} P(\mathbf{s}'|\mathbf{s},\mathbf{a}) \cdot (1-\zeta(\mathbf{s}',\mathbf{a}')) \cdot \pi(\mathbf{a}'|\mathbf{s}') \cdot \widehat{Q}^{\pi}(\mathbf{s}',\mathbf{a}')}_{\Delta_2\text{:bias due to leaving support; upper bounded due to } \zeta\text{-cover}}
\end{aligned}$$

To bound the inner summation over $(\mathbf{s}',\mathbf{a}')$ in term (a), we can apply Lemma B.5 since $\widehat{P}(\mathbf{s}'|\mathbf{s},\mathbf{a})$ and $\zeta(\mathbf{s}',\mathbf{a}')$ are not independent, to obtain a horizon-free bound. Finally, we use Condition 3.1 to bound the density ratios, in expectation over the randomness in dataset $\mathcal{D}$, identical to the proof for the conservative lower-confidence bound method from before. Formally, using Lemma B.5, we get, with high probability $\geq 1-\delta$:

$$\forall(\mathbf{s},\mathbf{a}) \text{ s.t. } n(\mathbf{s},\mathbf{a}) \geq 1, \ \left|\left(\widehat{P}(\mathbf{s},\mathbf{a}) - P(\mathbf{s},\mathbf{a})\right) \cdot \widehat{V}^{\pi}_{\zeta}\right| \leq \sqrt{\frac{\mathbb{V}(\widehat{P}(\mathbf{s},\mathbf{a}),\widehat{V}^{\pi}_{\zeta})\iota}{n(\mathbf{s},\mathbf{a})}} + \frac{\iota}{n(\mathbf{s},\mathbf{a})},$$

where we utilized the fact that $\widehat{V}^{\pi}_{\zeta} \leq \widehat{V}^{\pi} \leq 1$ due to Condition 3.2. For bounding Δ_2, we note that this term is bounded by the definition of ζ-covered policy:

$$\Delta_2 \leq \sum_{t=0}^{\infty} \gamma^t (1-\gamma)U(b) \leq U(b). \tag{11}$$

Thus, the overall policy evaluation error is given by:

$$\left|J(\pi) - \widehat{J}_{\zeta}(\pi)\right| \lesssim \sum_{t=0}^{\infty} \gamma^t d^{\pi}_t(\mathbf{s},\mathbf{a}) \left[\sqrt{\frac{\mathbb{V}(\widehat{P}(\mathbf{s},\mathbf{a}),\widehat{V}^{\pi}_{\zeta})\iota}{n(\mathbf{s},\mathbf{a})}} + \frac{\iota}{n(\mathbf{s},\mathbf{a})}\right] + U(b). \tag{12}$$

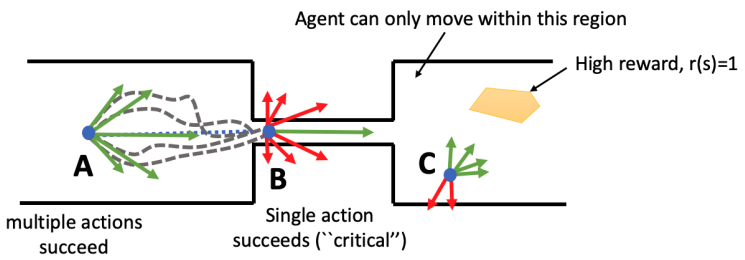

Figure 3: **Illustration showing the intuition behind critical points in a simple navigation task.** The agent is supposed to navigate to a high-reward region marked as the yellow polygon, without crashing into the walls. For different states, **A**, **B** and **C** that we consider, the agent has a high volume of actions that allow it to reach the goal at states **A** and **C**, but only few actions that allow it to do so at state **B**. States around **A** and **C** are not critical, and so this task has only a small volume of critical states (i.e., those in the thin tunnel).

Equation 12 mimics the Φ term in equation 8 that is bounded in Section B.2.5, with an additional offset $U(b)$. Hence, we can reuse the same machinery to show the bound in expectation over the randomness in the dataset, which completes the proof. $\square$

Using Lemma C.1, we can now that the policy constraint algorithm attains a favorable guarantee when compared to the best policy that is ζ-covered:

Theorem C.1 (Performance of policy-constraint offline RL)**.** *Under Condition 3.2, the policy $\widehat{\pi}^*$ incurs bounded suboptimality against the best ζ-covered policy, with high probability $\geq 1-\delta$:*

$$\mathbb{E}_{\mathcal{D}}\left[J(\pi^*_\zeta) - J(\widehat{\pi}^*)\right] \lesssim \sqrt{\frac{C^*|\mathcal{S}|H\iota}{N}} + \frac{C^*|\mathcal{S}|H\iota}{N} + 2U(b)\,.$$

To prove this theorem, we use the result of Lemma C.1 for the fixed policy, that is agnostic of the dataset, and then again use the recursion as before to bound the value of the data-dependent policy. The latter uses Lemma B.5 and ends up attaining a bound previously found in Appendix B.2.5, which completes the proof of this Theorem. When the term $U(b)$ is small, such that $U(b) \leq \mathcal{O}(H^{0.5-\varepsilon})$ for $\varepsilon > 0$, then we find that the guarantee in Theorem C.1 matches that in Theorem 4.2, modulo a term that grows slower in the horizon than the other terms in the bound. If $U(b)$ is indeed small, then all properties that applied to conservative offline RL shall also follow for policy-constraint algorithms.

Note on the bound. We conjecture that it is possible to get rid of the $U(b)$ term, under certain assumptions on the support indicator $\zeta(\mathbf{s}, \mathbf{a})$, and by relating the values of $\zeta(\mathbf{s}, \mathbf{a})$ and $\zeta(\mathbf{s}', \mathbf{a}')$, at consecutive state-action tuples. For example, if $\zeta(\mathbf{s}', \mathbf{a}') = 1 \implies \zeta(\mathbf{s}, \mathbf{a}) = 1$, then we can derive a stronger guarantee.

D Intuitive Illustrations of Derived Conditions and Practical Guidelines for Verifying Them

In this section, we present intuitive illustrations of the various conditions we study and discuss practical guidelines that allow a practitioner to verify whether they are likely to hold for their problem domain. We focus on Conditions 4.1 and 4.2, as Condition 3.2 is satisfied in very common settings such as learning from sparse rewards obtained at the end of an episode, indicating success or failure.

D.1 Condition 4.1

To provide intuition behind when this condition is satisfied, we first provide an example of a sample navigation domain in Figure 3, to build examples of critical states. As shown in the figure, the task is to navigate to the high-reward yellow-colored region. We wish to understand if states marked as **A**, **B** and **C** are critical or not. The region where the agent can move is very wide in the neighborhood around state **A**, very narrow around state **B** and wide again around state **C**. In this case, as marked on the figure if the agent executes actions shown via green arrows at the various states, then it is likely to

sill finish the task, while if it executes the actions shown via red arrows it is likely not on track to reach the high-reward region.

Several actions at states **A** and **C** allow the agent to still reach the goal, without crashing into the walls, while only one action at state **B** allows so, as shown in the figure. Thus, state **B** is a "critical state" as per Definition 4.1. But since most of the states in the wide regions of the tunnel are non-critical, this navigation domain satisfies Condition 4.1.

D.2 CONDITION 4.2

Now we discuss the intuition behind why offline RL run on a form of noisy-expert data can outperform BC on expert data. As shown in Figure 4, the task is to navigate from the **Start** to the **Goal**. In this case, when BC is trained using only expert trajectories, and the environment consists of a stochastic dynamics, then running the BC policy during evaluation may diverge to low reward regions as shown in the second row, second column of Figure 4. On the other hand, if RL is provided with some noisy-expert data that visits states around expert trajectories which might eventually lead to low-rewarding states, then an effective offline RL method should be able to figure out how to avoid such states and solve the task successfully.

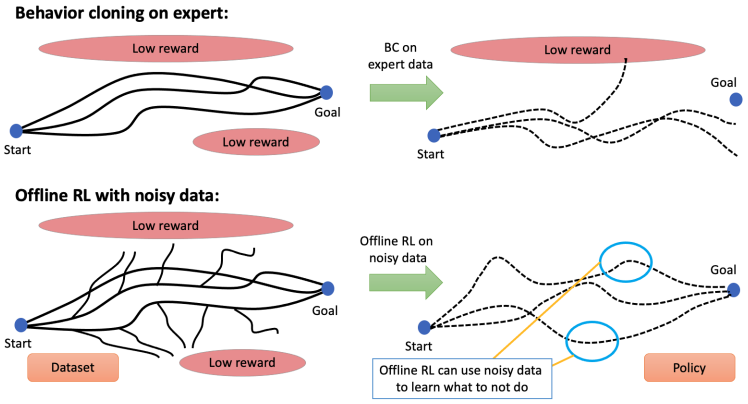

Figure 4: **Illustration showing the intuition behind suboptimal data in a simple navigation task.** BC trained on expert data (data composition is shown on the left) may diverge away from the expert and find a poor policy that does not solve the task. On the other hand, if instead of expert data, offline RL is provided with noisy expert data that sometimes ventures away from the expert distribution, RL can use this data to learn to stay on the course to the goal.

Condition 4.2 requires that state-action tuples with $d^*(\mathbf{s}, \mathbf{a}) \geq b/H$ under the expert policy have high-enough density $\mu(\mathbf{s}, \mathbf{a}) \geq b$ under the data distribution. There are two ways to attain this condition: **(1)** the expert policy already sufficiently explores the state ($d^*(\mathbf{s}, \mathbf{a}) \geq b$), for example, when the environment is mostly deterministic or when trajectories are cyclic (e.g., in locomotion tasks discussed below), or, **(2)** the offline dataset is noisy such that states that are less-frequently visited by the expert are explored more in the data. The latter can be satisfied when the agent is provided with "negative data", i.e., failed trajectories, starting from states in an optimal trajectory as shown in Figure 4. Such counterfactual trajectories allow the agent to observe more outcomes starting from a state in the optimal trajectory. And running offline RL on them will enable the agent to learn what *not* to do, as illustrated in Figure 4. On the other hand, BC cannot use this negative data.

This condition is satisfied in the following practical problems:

- **Robotics**: In robotics noisy-expert data may not be directly available via demonstrations but can be obtained by a practitioner by running some limited autonomous data collection using noisy scripted policies (e.g., Kalashnikov et al. (2018a) and Kalashnikov et al. (2021) run partly trained RL policies for noisy collection for running offline RL).
- **Autonomous driving and robotics:** In some domains such as autonomous driving and robotics, rolling out short counterfactual trajectories from states visited in the offline data, and labeling it with failures is a common strategy (Bojarski et al., 2016; Rao et al., 2020). A

practitioner could therefore satisfy this condition by tuning the amount of counterfactual data they augment to training.

- **Recommender systems**: In recommender systems, Chen et al. (2019) found that it was practical to add additional stochasticity while rolling out a policy for data collection which could be controlled, enabling the practitioner to satisfy this condition.
- **Trajectories consist of cyclic states (e.g., robot locomotion, character animation in computer graphics):** This condition can be satisfied in several applications such as locomotion (e.g., in character animation in computer graphics (Peng et al., 2018), or open-world robot navigation (Shah et al., 2021)) where the states observed by the agent are repeated over the course of the trajectory. Therefore a state with sufficient density under the optimal policy may appear enough times under μ.

D.3 Practical Guidelines For Verifying These Conditions

These conditions listed above and discussed in the paper can be difficult to check for in practice; for example, it is non-obvious how to quantitatively compute the volume of critical states, or how well-explored the dataset is. However, we believe that a practitioner who has sufficient domain-specific knowledge has enough intuition to qualitatively reason about whether the conditions hold. We believe that such practitioners can answer the following questions about their particular problem domain:

- Does there only exist a large fraction of states along a trajectory where either multiple good actions exist, or it is easy to recover from suboptimal actions?
- Is the offline dataset collected from a noisy-expert policy, and if not, can the dataset be augmented using perturbed or simulated trajectories?

If either of those questions can be answered positively, our theoretical and empirical results show that it is favorable to use offline RL algorithms, even over collecting expert data and using BC. At least, offline RL algorithms should be tried on the problem under consideration. Hence, the goal of our contributions is not to provide a rigid set of guidelines, but rather provide practical advice to the ML practitioner. We would like to highlight that such non-rigid guidelines exist in general machine learning, beyond RL. For example, in supervised learning, tuning the architecture of a deep neural network depends heavily on domain knowledge, and choosing kernel in a kernel machine again depends on the domain.

E Experimental Details

In this section we provide a detailed description of the various tasks used in this paper, and describe the data collection procedures for various tasks considered. We discuss the details of our tasks and empirical validation at the following website: `https://sites.google.com/view/shouldirunrlorbc/home`.

E.1 Tabular Gridworld Domains

The gridworld domains we consider are described by 10×10 grids, with a start and goal state, and walls and lava placed in between. We consider a sparse reward where the agent earns a reward of 1 upon reaching the goal state; however, if the agent reaches a lava state, then its reward is 0 for the rest of the trajectory. The agent is able to move in either of the four direction (or choose to stay still); to introduce stochasticity in the transition dynamics, there is a 10% chance that the agent travels in a different direction than commanded.

The exact three gridworlds we evaluate on vary in the number of critical points encountered per trajectory. We model critical states as holes in walls through which the agent must pass; if the agent chooses a wrong action at those states, it veers off into a lava state. The exact three gridworlds we evaluate on are: (a) "Single Critical" with one critical state per trajectory, (b) "Multiple Critical" with three critical states per trajectory, and (c) "Cliffwalk", where every state is critical (Schaul et al., 2015). The renderings of each gridworld are in Figure 5.

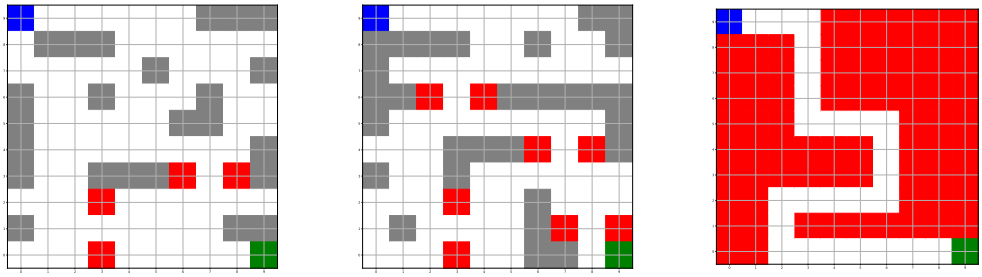

Figure 5: Renderings of three gridworld domains we evaluate on, where states are colored as: Start:blue, Goal:green, Lava:red, Wall:grey, and Open:white. The domains have varying number of critical points. *Left*: Single Critical. *Middle*: Multiple Critical. *Right*: Cliffwalk.

E.2 Multi-Stage Robotic Manipulation Domains

Overview of domains. These tasks are taken from Singh et al. (2020). The robotic manipulation simulated domains comprise of a 6-DoF WidowX robot that interacts with objects in the environment. There are three tasks of interest, all of which involve a drawer and a tray. The objective of each task is to remove obstructions of the drawer, open the drawer, pick an object and place it in a tray. The obstructions of the drawer were varied giving rise to three different domains — **open-grasp** (no obstruction of the drawer), **close-open-grasp** (an open top drawer obstructs the bottom drawer), **pick-place-open-grasp** (an object obstructs the bottom drawer).

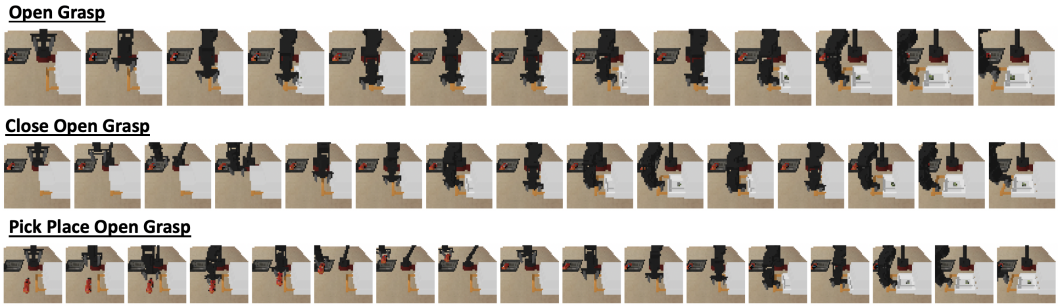

Figure 6: Filmstrip of the three tasks that we stufy for robotic manipulation – open-grasp, close-open-grasp and pick-place-open-grasp.

Reward function. For all the three tasks considered, a reward of $+1$ is provided when the robot is successfully able to open the drawer of interest (bottom drawer in close-open-grasp and pick-place-open-grasp; the only drawer in open-grasp) and is able to grasp the object inside it. If the robot fails at doing so, it gets no reward.

Dataset composition. For each task, we collected a dataset comprising of 5000 trajectories. For our experiments where we utilize expert data, we used the (nearly)-expert scripted policy for collecting trajectories and discarded the ones that failed to succeed. Thus the expert data attains a 100% success rate on this task. For our experiments with suboptimal data, which is used to train offline RL, we ran a noisy version of this near-expert scripted policy and collected 5000 trajectories. The average success rate in the suboptimal data is around 40-50% in both opening and closing the drawers with, 70% success rate in grasping objects, and a 70% success rate in place those objects at random locations in the workspace.

E.3 ANTMAZE DOMAINS

Overview of the domain. This task is based on the antmaze-medium and antmaze-large environments from Fu et al. (2020). The goal in this environment is to train an 8-DoF quadraped ant robot to successfully navigate to a given, pre-specifcied target location in a maze. We consider two different maze layouts provided by Fu et al. (2020). We believe that this domain is well-suited to test BC and RL methods in the presence of multiple critical points, and is representative of real-world navigation scenarios.

Scripted policies and datasets. We utilize the scripted policies provided by Fu et al. (2020) to generate two kinds of expert datasets: first, we generate trajectories that actually traverse the path from a given default start location to the target goal location that we consider for evaluation, and second, we generate trajectories that go from multiple random start positions in the maze to the target goal location in the maze. The latter has a wider coverage and a different initial state distribution compared to what we will test these algorithms on. We collected a dataset of 500k transitions, which was used by both BC and offline RL.

Reward functions. In this task, we consider a sparse binary reward $r(\mathbf{s}, \mathbf{a}) = +1$, if $|\mathbf{s}' - \mathbf{g}| \leq \varepsilon = 0.5$ and 0 otherwise. This reward is only provided at the end of a trajectory. This reward function is identical to the one reported by D4RL (Fu et al., 2020), but the dataset composition in our case comes from an expert policy.

E.4 ADROIT DOMAINS

Overview of the domain. The Adroit domains (Rajeswaran et al., 2018; Fu et al., 2020) involve controlling a 24-DoF simulated Shadow Hand robot tasked with hammering a nail (hammer), opening a door (door), twirling a pen (pen) or picking up and moving a ball (relocate). This domain presents itself with narrow data distributions, and we utilize the demonstrations provided by Rajeswaran et al. (2018) as our expert dataset for this task. The environments were instantiated via D4RL, and we utilized the environments marked as: `hammer-human-longhorizon`, `door-human-longhorizon`, `pen-human-longhorizon` and `relocate-human-longhorizon` for evaluation.

Reward functions. We directly utilize the data from D4RL (Fu et al., 2020) for this task. However, we modify the reward function to be used for RL. While the D4RL adroit domains provide a dense reward function, with intermediate bonuses provided for various steps, we train offline RL using a binary reward function. To compute this binary reward function, we first extract the D4RL dataset for these tasks, and then modify the reward function as follows:

$$r(\mathbf{s}, \mathbf{a}) = +1 \quad \text{if } r_{\text{D4RL}}(\mathbf{s}, \mathbf{a}) \geq 70.0 \quad \text{(hammer-human)} \tag{13}$$

$$r(\mathbf{s}, \mathbf{a}) = +1 \quad \text{if } r_{\text{D4RL}}(\mathbf{s}, \mathbf{a}) \geq 9.0 \quad \text{(door-human)} \tag{14}$$

$$r(\mathbf{s}, \mathbf{a}) = +1 \quad \text{if } r_{\text{D4RL}}(\mathbf{s}, \mathbf{a}) \geq 47.0 \quad \text{(pen-human)} \tag{15}$$

$$r(\mathbf{s}, \mathbf{a}) = +1 \quad \text{if } r_{\text{D4RL}}(\mathbf{s}, \mathbf{a}) \geq 18.0 \quad \text{(relocate-human)} \tag{16}$$

The constant thresholds for various tasks are chosen in a way that only any transition that actually activates the flag `goal_achieved=True` flag in the D4RL Adroit environments attains a reward +1, while other transitions attain a reward 0. We also evaluarte the performance of various algorithms on this new sparse reward that we consider for our setting.

E.5 ATARI DOMAINS

We utilized 7 Atari games which are commonly studied in prior work (Kumar et al., 2020; 2021b): ASTERIX, BREAKOUT, SEAQUEST, PONG, SpaceInvaders, Q*BERT, ENDURO for our experiments. We do not modify the Atari domains, directly utilize the sparse reward for RL training and operate in the stochastic Atari setting with sticky actions for our evaluations. For our experiments, we extracted datasets of different qualities from the DQN-Replay dataset provided by Agarwal et al. (2020b). The DQN-Replay dataset is stored as 50 buffers consisting of sequentially stored data observed during training of an online DQN agent over the course of training.

Expert data. To obtain expert data for training BC and RL algorithms, we utilized all the data from buffer with id 49 (i.e., the last buffer stored). Since each buffer in DQN-Replay consists of 1M transition samples, all algorithms training on expert data learn from 1M samples.

Domain / Behavior Policy	Task/Data Quality	BC	Naïve CQL	Tuned CQL
7 Atari games (RL policy)	Pong, Expert	109.78 ± 2.93	102.03 ± 4.43	105.84 ± 2.22
	Breakout, Expert	75.59 ± 21.59	71.22 ± 27.55	94.77 ± 27.02
	Asterix, Expert	41.10 ± 9.5	44.81 ± 12.0	80.19 ± 20.7
	SpaceInvaders, Expert	40.88 ± 4.17	45.27 ± 7.32	54.15 ± 2.96
	Q*bert, Expert	121.48 ± 9.06	105.83 ± 23.17	98.52 ± 18.62
	Enduro, Expert	78.67 ± 3.98	141.53 ± 18.79	127.02 ± 10.53
	Seaquest, Expert	63.15 ± 9.47	64.03 ± 27.67	85.28 ± 21.28

Table 3: Per-game results for the Atari domains with expert data. Note that while naïve CQL does not perform much better than BC (it performs similarly as BC), tuned CQL with the addition of the DR3 regularizer performs much better.

Task	BC-PI	CQL
Pong	100.03 ± 5.01	94.48 ± 8.39
Breakout	25.99 ± 1.98	86.92 ± 13.74
Asterix	29.77 ± 5.33	157.54 ± 37.94
SpaceInvaders	31.45 ± 1.96	63.7 ± 16.18
Q*bert	106.06 ± 8.63	88.72 ± 20.41
Enduro	68.56 ± 0.23	148.97 ± 12.3
Seaquest	22.51 ± 2.23	124.95 ± 43.86

Table 4: Comparing the performance of BC-PI and offline RL on noisy-expert data. Observe that in general, offline RL significantly outperforms BC-PI.

Noisy-expert data. For obtaining noisy-expert data, analogous to the gridworld domains we study, we mix data from the optimal policy (buffer 49) with an equal amount of random exploration data drawn from the initial replay buffers in DQN replay (buffers 0-5). i.e. we utilize 0.5M samples form buffer 49 in addition to 0.5M samples sampled uniformly at random from the first 5 replay buffers.

F Tuning and Hyperparameters

In this section, we discuss our tuning strategy for BC and CQL used in our experiments.

Tuning CQL. We tuned CQL offline, using recommendations from prior work (Kumar et al., 2021c). We used default hyperparameters for the CQL algorithm (Q-function learning rate = 3e-4, policy learning rate = 1e-4), based on prior works that utilize these domains. Note that prior works do not use the kind of data distributions we use, and our expert datasets can be very different in composition compared to some of the other medium or diverse data used by prior work in these domains. In particular, with regards to the hyperaprameter α in CQL that trades off conservatism and the TD error objective, we used $\alpha = 0.1$ for all Atari games (following Kumar et al. (2021b)), and $\alpha = 1.0$ for the robotic manipulation domains following (Singh et al., 2020). For the Antmaze and Adroit domains, we ran CQL training with multiple values of $\alpha \in \{0.01, 0.1, 0.5, 1.0, 5.0, 10.0, 20.0\}$, and then picked the smallest α that did not lead to eventually divergent Q-values (either positively or negatively) with more (1M) gradient steps. Next, we discuss how we regularized the Q-function training and performed policy selection on the various domains.

- **Detecting overfitting and underfitting**: Following Kumar et al. (2021c), as a first step, we detect whether the run is overfitting or underfitting, by checking the trend in Q-values. In our experiments, we found that Q-values learned on Adroit domains exhibited a decreasing trend throughout training, from which we concluded it was overfitting. On the Antmaze and Atari experiments, Q-values continued to increase and eventually stabilized, indicating that the run might be underfitting (but not overfitting).
- **Correcting for overfitting and policy selection**: As recommended, we applied a capacity decreasing regularizer to correct for overfitting, by utilizing dropout on every layer of the Q-function. We ran with three values of dropout parobability, $p \in \{0.1, 0.2, 0.4\}$, and found that 0.4 was the most effective in alleviating the monotonically decreasing trend in Q-values, so used that for our results.

Then, we performed policy checkpoint selection by picking the earliest checkpoint that appears after the peak in the Q-values for our evaluation.

- **Correcting for underfitting:** In the Atari and Antmaze domains, we observed that the Q-values exhibited a stable, convergent trend and did not decrease with more training. Following Kumar et al. (2021c), we concluded that this resembled underfitting and utilized a capacity-increasing regularizer (DR3 regularizer (Kumar et al., 2021a)) for addressing this issue. We used identical hyperparameter for the multiplier (β) on this regularizer term for both Atari and Antmaze, $\beta = 0.03$ and did not tune it.

Tuning BC. In all domains, we tested BC with different network architectures. On the antmaze domain, we evaluated two feed-forward policy architectures of sizes $(256, 256, 256)$ and $(256, 256, 256, 256, 256, 256)$ and picked the one that performed best online. ON Adroit domains, we were not able to get a tanh-Gaussian policy, typically used in continuous control to work well, since it overfitted very quickly giving rise to worse-than-random performance and therefore, we switched to utilizing a Gaussian policy network with hidden layer sizes $(256, 256, 256, 256)$, and a learned, state-dependent standard deviation. To prevent overfitting in BC, we applied a strong dropout regularization of $p = 0.2$ after each layer for Adroit domains. On Atari and the manipulation domains, we utilized a Resnet architecture borrowed from IMPALA (Espeholt et al., 2018), but without any layer norm.

More details are at: `https://sites.google.com/view/shouldirunrlorbc/home`.

Tuning BC-PI. Our BC-PI method is implemented by training a Q-function via SARSA, i.e., $Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma Q(\mathbf{s}', \mathbf{a}')$, where $(\mathbf{s}', \mathbf{a}')$ is the state-action pair that appears next in the trajectory after $(\mathbf{s}, \mathbf{a})$ in the dataset using the following Bellman error loss function to train Q_θ:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{s},\mathbf{a},\mathbf{s}',\mathbf{a}' \sim \mathcal{D}} \left(Q_\theta(\mathbf{s}, \mathbf{a}) - (r(\mathbf{s}, \mathbf{a}) + \gamma \bar{Q}_{\bar{\theta}}(\mathbf{s}', \mathbf{a}')) \right)^2,$$

and then performing advantage-weighted policy extraction: $\pi(\mathbf{a}|\mathbf{s}) \propto \widehat{\pi_\beta}(\mathbf{a}|\mathbf{s}) \cdot \exp(A(\mathbf{s}, \mathbf{a})/\eta)$ on the offline dataset $\mathcal{D}$. The loss function for this policy extraction step, following Peng et al. (2019) is given by:

$$\pi_\phi \leftarrow \max_{\pi_\phi} \sum_{\mathbf{s},\mathbf{a}} \log \pi_\phi(\mathbf{a}|\mathbf{s}) \cdot \exp\left(\frac{Q_\theta(\mathbf{s}, \mathbf{a}) - V(\mathbf{s})}{\eta} \right),$$

where the value function was given by $V(\mathbf{s}) = \sum_{\mathbf{a}'} \pi_\beta(\mathbf{a}'|\mathbf{s}) Q_\theta(\mathbf{s}, \mathbf{a}')$ and π_β is a learned model of the behavior policy, as done in the implementation of Brandfonbrener et al. (2021). This model of the behavior policy is trained according to the tuning protocol for BC, and is hence well-tuned.

What we tuned: We tuned the temperature hyperparameter η using multiple values spanning various levels of magnitude: $\{0.005, 0.05, 0.1, 0.5, 1.0, 3.0\}$ and additionally tried two different clippings of the advantage values $A(\mathbf{s}, \mathbf{a}) := Q(\mathbf{s}, \mathbf{a}) - V(\mathbf{s})$ between $[-10, 2]$ and $[-10, 4]$. The Q-function architecture is identical to tuned CQL, and the policy and the model of the behavior policy both utilize the architecture used by our BC baseline.

Our observations: We summarize our observations below:

- We found that in all the runs the temporal difference (TD) error for SARSA was in the range of $[0.001, 0.003]$, indicating that the SARSA Q-function is well behaved.
- The optimal hyperparameters that lead to the highest average performance across all games is $\eta = 0.005$, and the advantage clipping between $[-10, 2]$. We find a huge variation in the performance of a given η, but we used a single hyperparameter η across all games, in accordance with the Atari evaluation protocols (Mnih et al., 2013), and as we did for all other baselines. For example, while on some games such as Qbert, BC-PI improves quite a lot and attains 118% normalized return, on Enduro it attains only 78% and on SpaceInvaders it attains 31.7%.
- These results perhaps indicate the need for per-state tuning of η, i.e., utilizing $\eta(\mathbf{s})$, however, this is not covered in our definition of BC-PI from Section 4.4, and so we chose to utilize a single η across all states. Additionally, we are unaware of any work that utilizes per-state $\eta(\mathbf{s})$ values for exponentiated-advantage weighted policy extraction.

Finally, we would note that we tuned this implementation to the best of our abilities, within our compute capabilities. We used online rollouts for tuning BC-PI, and the results shown in Figue 2 is what we obtain.

G REGARDING OFFLINE RL LOWER BOUNDS

Here we address the connection between recently published lower-bounds for offline RL by Zanette (2020) and Wang et al. (2021) and our work.

Zanette (2020) worst-case analysis does address function approximation even under realizability and closure. However, this analysis concerns policy evaluation and applies to policies that go out of the support of the offline data. This does not necessarily prove that any conservative offline RL algorithm (i.e., one that prevents the learned policy from going out of the support of the offline dataset) would suffer from this issue during policy learning, just that there exist policies for which evaluation would be exponentially bad. Offline RL can exactly prevent the policy from going out of the support of the dataset in Zanette (2020)'s counterexample since the dynamics and reward functions are deterministic. This policy can still improve over behavior cloning by stitching overlapping trajectories, similar to the discussion of one-step vs multi-step PI in Theorem 4.4. Thus, while the lower-bound applies to off-policy evaluation of some target policies, it doesn't apply to evaluation of ***every*** policy in the MDP, and specifically ***does not*** apply to in-support policies, which modern offline RL algorithms produce, in theory and practice. Certainly, their lower bound would apply to an algorithm aiming to estimate Q^* from the offline data, but that's not the goal of pessimistic algorithms that we study. Finally, our practical results indicates that offline RL methods can be made to perform well in practice despite the possibility of any such lower bound.

The lower bound in Wang et al. (2020) only applies to non-pessimistic algorithms that do not tackle distributional shift. In fact, Section 5, Theorem 5.1 in Wang et al. (2021) provides an upper bound for the performance of offline policy evaluation with low distribution shift. The algorithms analyzed in our paper are pessimistic and guarantee low distribution shift between the learned policy and the dataset, and so the lower bound does not apply to the algorithms we analyze.