
Decision-Point Guided Safe Policy Improvement

Abhishek Sharma
Harvard University
Cambridge, USA

Leo Benac
Harvard University
Cambridge, USA

Sonali Parbhoo
Imperial College
London, UK

Finale Doshi-Velez
Harvard University
Cambridge, USA

Abstract

Within batch reinforcement learning, safe policy improvement seeks to ensure that the learnt policy performs at least as well as the behavior policy that generated the dataset. The core challenge is seeking improvements while balancing risk when many state-action pairs may be infrequently visited. In this work, we introduce Decision Points RL (DPRL), an algorithm that restricts the set of state-action pairs (or regions for continuous states) considered for improvement. DPRL ensures high-confidence improvement in densely visited states (i.e. decision points) while still utilizing data from sparsely visited states. By appropriately limiting where and how we may deviate from the behavior policy, we achieve tighter bounds than prior work; specifically, our data-dependent bounds do not scale with the size of the state and action spaces. In addition to the analysis, we demonstrate that DPRL is both safe and performant on synthetic and real datasets.

1 Introduction

Batch Reinforcement Learning (Batch RL) (Lange et al., 2012) involves developing an effective policy using a limited number of trajectories generated by another *behavior policy*. Used in settings ranging from education, robotics and medicine (Fu et al.), Batch RL is valuable when interaction with the environment may be risky (Garcia and Fernández, 2015) or expensive (Kalashnikov et al., 2018). However, the benefits of batch RL rely on a sufficiently exploratory behavior

policy (Sutton and Barto, 2018; Kumar et al., 2022). In scenarios with limited exploration and experts making systematic errors, learning an optimal policy may not be feasible without risking the adoption of an unsafe policy that performs poorly compared to the behavior. Additionally, since real-world Batch RL deployments are often incremental (Fu et al.), it is sufficient—and perhaps preferable—to implement changes relative to the existing behavior policy. Our focus is on providing safe, high-confidence modifications in settings with limited exploration in the data.

The main challenge for safe policy improvement (SPI) is appropriately restricting the learned policy to be close to the behavior data, while still identifying the points of improvement over the behavior. Density-based safe RL, such as conservative Q-learning (CQL) (Kumar et al., 2019) and behavior cloning (BC), constrain the learned policy to be close to behavior policy. However, this approach is too conservative if the behavior policy is both stochastic and suboptimal as it will not choose the better of the explored actions. Pessimism-based planning (Liu et al., 2020; Yu et al., 2020; Kidambi et al., 2021) incorporates pessimism in the value estimates of state-action pairs (proportional to the uncertainty in the estimates), but can be overly conservative because it will penalize actions that lead to states that are not frequently visited (even if the action itself is observed often enough).

To address the limitations of pessimistic planning and density-based constraints, support-constrained policies Wu et al. (2022) restrict the learned policy to the support of the behavior policy. However, these policies can become unsafe with even a few noisy actions or rewards, which are common in practice. Among support-based methods, count-based techniques like Safe Policy Improvement with Baseline Bootstrapping (SPIBB) Laroche et al. (2019) impose a count-based constraint on (s, a) pairs to ensure policy improvements occur only over sufficiently observed state-action pairs. However, SPIBB requires access to the behavior policy function, which is often impractical in real-world applications. We demonstrate that its per-

formance significantly deteriorates when the behavior policy must be estimated, and its improvement guarantees are not tight in practice.

Our key insight is that we need to identify only those behavior changes that yield the most improvement. We achieve this by constraining changes to (s, a) pairs observed frequently in the dataset, specifically those with a count $\geq N_\wedge$ (termed ‘Decision Points’). For states where we lack high-confidence improvements, we defer to the current behavior policy. This approach enhances performance over SPIBB in two ways: it allows for better returns without needing access to the true behavior policy and enables an explicit ‘DEFER’ flag when we are unsure about achieving safe improvements. The hyperparameter N_\wedge allows us to balance confidence in performance improvement, safety, and the number of changes. Additionally, this method results in a learned policy with a few high-improvement changes that can be easily reviewed and implemented in practice (e.g., by a clinician).

Our work makes the following contributions: (i) We introduce Decision Points RL (DPRL), a safe batch RL algorithm that restricts improvements to specific state-action pairs (or regions in continuous states), deviating from the behavior policy only when confident and deferring otherwise. (ii) Unlike previous methods, DPRL operates without needing knowledge of the behavior policy to ensure safety. (iii) DPRL offers significantly tighter theoretical guarantees than existing algorithms for both discrete and continuous state problems, with bounds that depend on the safety threshold parameter N_\wedge rather than the state-action space size. (iv) Empirically, we demonstrate that DPRL better balances safety and improvement compared to alternatives in both synthetic and real-world medical datasets while making minimal changes.

2 Related Work

Policy regularization in Offline RL. Many RL applications require agents to learn from a fixed batch of pre-collected data, limiting further data collection. Various methods constrain policies to this data. Density-constraining methods, such as (Fujimoto et al., 2019; Kumar et al., 2019, 2020; Yu et al., 2022; Thomas et al., 2015b), keep the action space close to the behavior policy. For instance, (Kumar et al., 2019) constrains action selection based on bootstrapping errors for actions outside the training data, while (Kumar et al., 2020) introduces a conservative Q-learning penalty (CQL) to address distribution shifts between the dataset and the learned policy. However, we demonstrate that density regularization techniques can be suboptimal when the behavior policy

is stochastic and suboptimal in certain states. In contrast, our approach does not impose restrictions on the policy distribution, only limiting support to frequently observed actions. Support-constraining methods, such as (Singh et al., 2022), restrict actions to those within the support of the behavior data but lack guarantees. Unlike our method, policy regularization techniques in offline RL do not prioritize safe policy improvement.

Safe Policy Improvement. Several studies have addressed safe policy improvement in batch RL settings (e.g., (Thomas et al., 2015a; Ghavamzadeh et al., 2016; Laroche et al., 2019)). (Ghavamzadeh et al., 2016) utilizes pessimism to regularize infrequently occurring state-action pairs. Meanwhile, (Laroche et al., 2019; Nadjahi et al., 2019) propose an algorithm that bootstraps a trained policy with a baseline when uncertainty is high, offering SPI guarantees only when insufficiently observed pairs adhere to the behavior policy. However, these results only hold if we have access to the behavior policy a priori. Other works, such as Scholl et al. (2022) and Wienhöft et al. (2023a), provide additional guarantees for bootstrapping methods. Some researchers have used pessimism to regularize the reward or action-value function for rarely observed pairs. (Liu et al., 2020) presents guarantees for batch RL via pessimistic formulations of policy and Q-iteration algorithms, while (Kidambi et al., 2021) and (Yu et al., 2020) focus on learning pessimistic MDPs for near-optimal policies. (Kim and Oh, 2023) constructs a penalized reward function based on state-action counts but fails to exclude rarely observed pairs. In contrast, our approach does not regularize the value function or reward model but directly constrains the policy, avoiding excessive conservatism. Notably, we do not require explicit access to the behavior policy and provide tighter improvement guarantees.

Non-parametric RL and Trajectory Stitching. Instead of directly constraining actions and states for safety and conservatism, there is extensive research on improving models through nonparametric methods or heuristics in areas with uneven data coverage. Among these, Shrestha et al. (2020) use k-nearest neighbors to estimate reward and transition models, which works well when neighbors are nearby but fails if neighbors are distant, while others such as Gottesman et al. (2019) combine parametric and nonparametric methods for better policy evaluation. Unlike these, our work makes no assumptions of the form of value function and offers a general analysis for safe policy improvement. Other methods, like those by Char et al. (2022); Hepburn and Montana (2022), use distance-based metrics for value function approximation but do not focus on safety policy improvement and assume

accurate model estimation, which is challenging with uneven data coverage. (Zhang et al., 2022) introduce decision regions for safe policy learning using a heuristic approach without theoretical guarantees, unlike our method. Notably, none of these heuristic or nonparametric approaches offer any theoretical guarantees as we do here. More closely related, (Lederer et al., 2019) use Gaussian Processes (GPs) to estimate transition models. Though their approach does provide theoretical bounds, our guarantees are tighter; we also focus explicitly on the task of safe policy improvement.

Learning to Defer to Human Expertise. Policy regularization, SPI, and trajectory stitching methods do not effectively allow for selective use of the behavior policy during decision-making or deferring to it when the learned policy may not be significantly better. Some studies focus on learning to defer to human expertise in both static and sequential contexts. For instance, (Madras et al., 2018) and (Mozannar and Sontag, 2020) investigate this in static classification tasks, while (Li et al., 2011) explore it in online settings that require a polynomial number of deferrals. In offline settings, however, more frequent deferrals may be more reasonable, as proposed here. (Straitouri et al., 2021; Joshi et al., 2022) also study deferring to the behavior policy when uncertain but lack theoretical guarantees.

3 Background

An MDP is a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ of discount $\gamma \in [0, 1)$, states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition probabilities $P(s'|s, a)$, and rewards $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. In this work, we assume that R is bounded in $[0, R_{\max}]$, \mathcal{A} is discrete, the starting state is fixed, and that R and P are unknown.

The behavior policy $\pi_b(a|s)$ is the policy that generated the observed trajectories, and $\pi(a|s)$ is the policy we are trying to learn. Given a policy π , we call $\eta_h^\pi(s) = \Pr[S_h = s|\pi]$ the marginal distribution of S_h under π . Then, $\eta^\pi(s, a) = \eta_h^\pi(s)\pi(a|s) = (1 - \gamma) \sum_{h=0}^{\infty} \gamma^h \eta_h^\pi(s, a)$ is called the marginal distribution of (s, a) . We are given a dataset $\mathcal{D} = \{S_0^n, A_0^n, R_0^n, \dots, S_{T_n}^n, A_{T_n}^n, R_{T_n}^n\}_{n=1}^N$ consisting of N trajectories, with actions taken by π_b . State-action pairs in \mathcal{D} can also be assumed to drawn *i.i.d.* from a *behavior distribution* $\mu(s, a) = \eta^\pi(s, a)$. We overload the notation and also denote the marginal distribution over states by $\mu(s) = \sum_{a \in \mathcal{A}} \mu(s, a)$.

The *value* $V_\pi(s)$ of policy π at state s is the expected discounted sum of rewards starting from s following policy π : $V_\pi(s) = \mathbb{E}_\pi[\sum_{t=1}^T \gamma^{t-1} R_t | s_1 = s]$. The action-value function (or Q-function) $Q_\pi(s, a)$

is the value of performing action a in state s and then performing policy π after: $Q_\pi(s, a) = \mathbb{E}_\pi[\sum_{t=1}^T \gamma^{t-1} R_t | s_1 = s, a_1 = a]$. Let Q_{\max} or V_{\max} be upper bounds on $Q(s, a)$ and $V(s)$. We denote $\rho(\pi)$ as the value of the start state.

4 Challenges with Prior Algorithms

Issues with Standard Assumptions. Many existing algorithms make assumptions about the data distribution. One common assumption is concentrability, i.e., $\|\nu(s, a)/\mu(s, a)\|_\infty \leq C$, where $\nu(s, a)$ is any distribution reachable for some non-stationary policy. The improvement guarantee scales with C . However, C can be arbitrarily large for real-world dataset, and algorithms relying on this assumption often diverge in practice (Liu et al., 2020). To remedy this, (Liu et al., 2020) assumed bounded density instead, i.e., $\eta_h^\pi(s, a) \leq U$ for any non-stationary policy π . The improvement guarantee scales with U/b , where $\mathbf{I}[b \leq \hat{\mu}(s, a)]$ is the filter chosen to exclude low-density (s, a) pairs in the empirical data density $\hat{\mu}$. However, the improvement guarantee is arbitrarily loose when selecting (s, a) pairs with sufficient observations but low value under the behavior policy.

Illustrative Example. We now show how existing SPI baselines (PQI, SPIBB, CQL) fail to choose optimal actions while avoiding risky ones. To demonstrate how a pessimism-based approach fails, consider the first MDP in Figure 1. There are three actions at the starting state s_0 : an optimal action, a_0 , leading to a ‘forest’ of sparsely-visited states with good outcomes; a risky action, a_2 , leading to a ‘forest’ of states with bad outcomes; and a suboptimal, risk-free action, a_1 , that leads to a middle road and is chosen by the behavior most of the time. A pessimism-based approach like PQI—which penalizes value estimates based on uncertainty—fails to learn that, even though the (s, a) pairs in the optimal forest are sparse in the data, we have enough observations to conclude that the good forest is the better choice. It should be noted that reducing the threshold for (s, a) density may result in taking the risky action.

For the second scenario, suppose there are 10 actions, 8 of which lead to extremely low values (and hence are risky), and it is difficult to determine that they are risky. One action, a_1 , is optimal, while another, a_2 , is frequently chosen by the behavior but yields suboptimal rewards. A density regularization method like CQL chooses the suboptimal action if it increases its regularization penalty (i.e. favoring the behavior policy) and chooses the risky actions if it reduces the penalty.

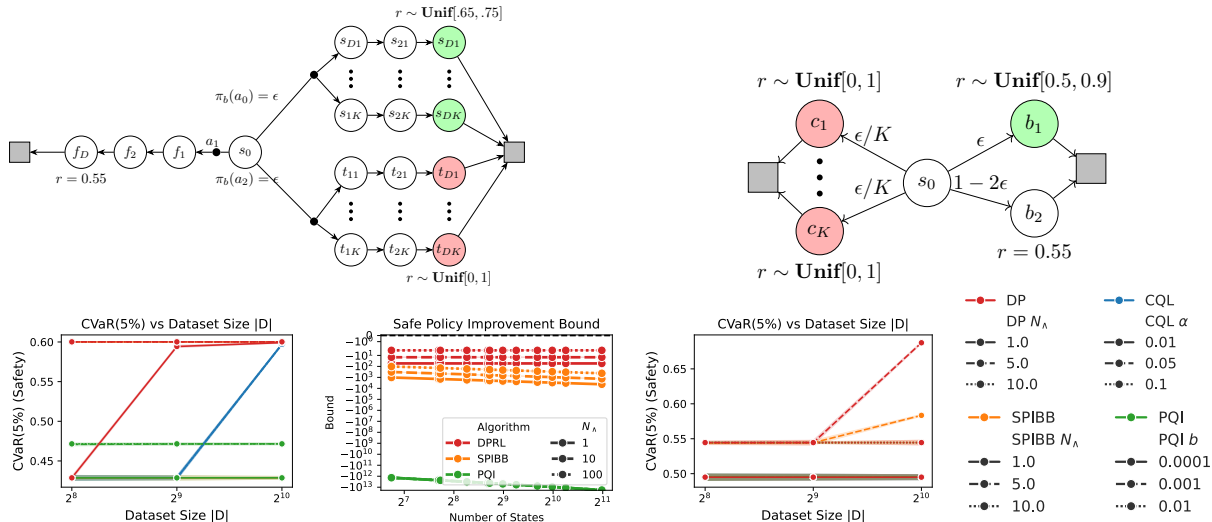


Figure 1: Top: Challenging MDPs prior approaches. In the first MDP, the behavior goes left with high probability. In the second MDP, the behavior goes to state b_2 with high probability. Green states are part of optimal trajectories, and red states are part of risky trajectories. Bottom L to R: PQI does poorly on the first MDP, DP has the tightest safety bounds, and CQL does poorly on the second MDP.

These two MDP scenarios illustrate how current SPI methods struggle with different types of action-risk dynamics. This intuition is supported empirically. In Figure 1 (left plot), we see that PQI fails to attain a high 5% conditional value at risk (CVaR 5%) (a metric used to measure safe improvement) even for relatively large dataset sizes in the “forest” MDP. Similarly, in the second MDP (right plot), CQL and SPIBB are not able to consistently learn a safe policy. In both cases, we can set an N_{min} parameter in DPRL to ignore actions that cannot be reliably estimated to have a good value. These examples also illuminate the settings in which DPRL performs particularly well: when there are systematic deviations from the behavior policy and when we are not required to act at all points in the state-action space.

Access to behavior. While SPIBB appears to be similar to DPRL in its use of a count parameter like N_\wedge , it requires access to the true behavior policy at training time—like most SPI approaches (Laroche et al., 2019; Scholl et al., 2022; Wienhöft et al., 2023b). We show in Section 7 that its performance can be significantly worse without access to the behavior policy. In contrast, we do not require access to the behavior policy during training time. This is an important consideration in real-world systems, where it can be difficult to elicit the behavior policy. For example, when working with doctors, we cannot expect to know the functional form of their behavior. However, we do require access to the behavior policy for *evaluation*, which can be achieved by either running a silent trial or by using off-policy evaluation (OPE). OPE in

a learning-to-assist framework is a promising area of research, and beyond the scope of this work.

5 Method

We now describe our decision-point RL method, which addresses several of the challenges with existing algorithms. We provide bounds for this method in Sec. 6.

Discrete Case. Define the following sets constructed from the dataset \mathcal{D} :

$$\begin{aligned} \mathcal{A}_s^{\text{DP}} &= \{a \in \mathcal{A} : n(s, a) \geq N_\wedge \text{ and } \hat{Q}^{\pi_b}(s, a) \geq \hat{V}^{\pi_b}(s)\}, \\ \mathcal{S}^{\text{DP}} &= \{s \in \mathcal{S} : \mathcal{A}_s^{\text{DP}} \neq \emptyset\} \end{aligned} \quad (1)$$

where $\hat{Q}(s, a)$ and $\hat{V}(s)$ are the estimated values under the behavior policy π_b . The set $\mathcal{A}_s^{\text{DP}} = \{\mathcal{A}_s^{\text{DP}}\}$ represents the set of actions in each state that we are confident are advantageous over the behavior policy (in the sense that the number of visits $n(s, a) \geq N_\wedge$). The set \mathcal{S}^{DP} is the set of states where at least one advantageous action exists. These states are the *decision points* on which we will recommend changes. We will defer to the behavior policy in the set $\Phi = \{s \in \mathcal{S} : \mathcal{A}_s^{\text{DP}} = \emptyset\}$ because cannot be confident that an advantageous action exists in those states.

Once the decision points are determined, we create an “elevated” Semi-MDP (SMDP) $\tilde{M} = (\mathcal{S}^{\text{DP}}, \mathcal{A}, \tilde{P}, \tilde{R}, \tilde{\gamma})$, where the state space is restricted to the decision points \mathcal{S}^{DP} , and the transition, reward, discount functions are estimated using the dataset \mathcal{D} . We use the SMDP framework since the number of time steps to reach the next decision point is not fixed, and

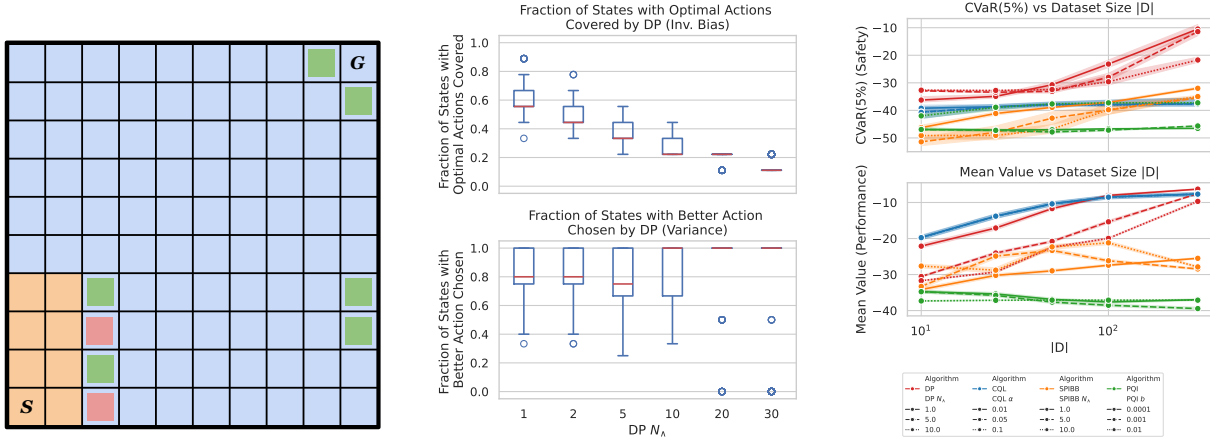


Figure 2: **GridWorld**: (left) Illustration of our Gridworld environment, (middle) Bias-variance trade-off managed by N_\wedge , (right) Performance of DPRL in terms of CVaR and Mean Value. DPRL provides safe policy improvement (CVaR), while matching baselines on mean value.

can vary with each (s, a, s') triplet. The policy set Π^{DP} is defined as the set of deterministic policies that select can only select advantageous actions in each state:

$$\Pi^{\text{DP}} = \{\pi : \pi(a|s) = 0 \quad \forall a \notin \mathcal{A}_s^{\text{DP}}\} \quad (2)$$

For each state $s \in \mathcal{S}^{\text{DP}}$, action $a \in \mathcal{A}_s^{\text{DP}}$, and next state $s' \in \mathcal{S}^{\text{DP}}$, we say that a transition (s, a, s', k) exists in a trajectory if s' is the first decision point in the trajectory starting from (s, a) and taking k steps (only considering first visits), and $\tilde{r}(n, s, a, s', k)$ is the discounted sum of rewards in the trajectory n starting from (s, a) and ending in s' after k steps, and 0 if no such transition exists. Then, the counts $\tilde{n}(s, a, s', k)$ are defined as the number of times (s, a, s', k) exists in the dataset, and $\tilde{P}(s'|s, a)$, $\tilde{\gamma}(s, a, s')$ and $\tilde{R}(s, a)$ are computed from these counts (see Algorithm 2 in the appendix for details):

$$\tilde{P}(s'|s, a) = \frac{\sum_{k=1}^T \tilde{n}(s, a, s', k)}{\sum_{s' \in \mathcal{S}^{\text{DP}}} \sum_{k=1}^T \tilde{n}(s, a, s', k)}, \quad (3)$$

$$\tilde{\gamma}(s, a, s') = \frac{\sum_{k=1}^T \tilde{n}(s, a, s', k) \gamma^k}{\sum_{k=1}^T \tilde{n}(s, a, s', k)},$$

$$\tilde{R}(s, a) = \frac{\sum_{s' \in \mathcal{S}^{\text{DP}}} \sum_{k=1}^T \sum_{n=1}^N \tilde{r}(n, s, a, s', k)}{\sum_{s' \in \mathcal{S}^{\text{DP}}} \sum_{k=1}^T \tilde{n}(s, a, s', k)} \quad (4)$$

We optimize over the policy set Π^{DP} by using policy iteration (Bradtke and Duff, 1994; Sutton, 1998) on \tilde{M} . In each iteration i , the policy $\pi^{(i)}$ is evaluated to get $V_M^{(i)}$, which is then improved to get $\pi^{(i+1)}$:

$$\pi^{(i+1)}(s) = \arg \max_{a \in \mathcal{A}_s^{\text{DP}}} \tilde{R}(s, a) + \mathbb{E}_{\tilde{P}(s'|s, a)} \tilde{\gamma}(s, a, s') V_M^{(i)}(s') \quad (5)$$

If the policy iteration converges after K steps to give $\pi^{(K)}$, the final policy π_{DP} can then be used to either defer to the behavior policy or to make a better decision for a given state s :

$$\pi_{\text{DP}}(s) = \begin{cases} \text{DEFER} & \text{if } s \in \Phi \\ \pi^{(K)}(s) & \text{otherwise} \end{cases} \quad (6)$$

Continuous Case. For continuous state spaces, we describe a variant of the DP algorithm that can be used to provide safe policy improvements for any given state (for pseudocode, see Algorithm 3 in the appendix). The algorithm involves storing the dataset \mathcal{D} . We define the following distance metric over state-action pairs (s, a) and states s :

$$d((s, a), (s', a')) = d(s, s') \text{ if } a = a', \text{ and } \infty \text{ otherwise} \\ d(s, s') = \|s - s'\|$$

Given a state s , define the set of neighbors of s within a ball of radius r as $\mathcal{N}(s) = \{S_t^n : d(s, S_t^n) \leq r \text{ for } n = 1, \dots, N\}$ and the count of neighbors as $n(s) = |\mathcal{N}(s)|$. Note that our analysis assumes the first neighboring state is included per trajectory n . However, in practice, including all neighbors helps reduce the variance of the estimates. Further, for each action a , $\mathcal{N}(s, a) = \{(S_t^n, A_t^n) : d((s, a), (S_t^n, A_t^n)) \leq r \text{ for } n = 1, \dots, N\}$ and $n(s, a) = |\mathcal{N}(s, a)|$. We use a Ball-Tree data structure to efficiently find a neighbor in $\mathcal{O}(\log K)$ time after $\mathcal{O}(K \log K)$ preprocessing time, where K is the total number of points to search over. Analogous to the discrete case, we define the set of advantageous actions in state s as:

$$\mathcal{A}_s^{\text{DP}} = \{a \in \mathcal{A} : n(s, a) \geq N_\wedge \text{ and } \hat{Q}^{\pi_b}(s, a) \geq \hat{V}^{\pi_b}(s)\} \quad (7)$$

where $\hat{Q}^{\pi_b}(s, a)$ and $\hat{V}^{\pi_b}(s)$ are estimated by averaging the returns of the neighbors in $\mathcal{N}(s, a)$ and $\mathcal{N}(s)$, respectively. We defer to the behavior if no advantageous actions exist in the state, and return the action with the highest estimated Q value otherwise:

$$\pi_{\text{DP}}(s) = \begin{cases} \text{DEFER} & \text{if } \mathcal{A}_s^{\text{DP}} = \emptyset \\ \arg \max_{a \in \mathcal{A}_s^{\text{DP}}} \hat{Q}^{\pi_b}(s, a) & \text{otherwise} \end{cases} \quad (8)$$

The policy is implicitly defined using the dataset \mathcal{D} , and we compute it for any state. The above procedure uses non-parametric estimation of Q^{π_b} - and V^{π_b} -values using the dataset \mathcal{D} . The r hyperparameter controls the trade-off between bias and variance in the estimates, as well as the sparsity of improvements. A smaller r will achieve lower bias and fewer possible decision points, while a larger r will result in a higher bias of estimates and more decision points. Note that the variance of the estimates is still controlled because we average over at least N_\wedge neighbors in the estimate. The trade-off versus a parametric approach is that we do not need to assume a particular form of the value function, at the expense of storing the dataset \mathcal{D} . This can be a reasonable trade-off in practice when the dataset is not too large. We show in the analysis section that this non-parametric approach can achieve tighter bounds than a parametric approach.

6 Analysis

We now provide performance bounds for the algorithms DPRL-D and DPRL-C presented in previous section. The following theorem proves that π_{DP} is a safe policy improvement over the behavior π_b .

Theorem 1 (DPRL Discrete) *Let π_{DP} be the policy obtained by the DP algorithm. Then π_{DP} is a safe policy improvement over the behavior policy π_b , with probability at least $1 - \delta$*

$$\rho(\pi_{\text{DP}}) - \rho(\pi_b) \geq -\frac{V_{\max}}{1 - \gamma} \sqrt{\frac{1}{N_\wedge} \log \frac{C(N_\wedge)}{\delta}} \quad (9)$$

where $C(N_\wedge)$ is the count of the number of (s, a) pairs that are observed at least N_\wedge times in the dataset:

$$C(N_\wedge) = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mathbf{I}[n(s, a) \geq N_\wedge] \quad (10)$$

Proof sketch. The key fact we exploit is the property that π_{DP} always takes an advantageous action with respect to $\hat{Q}_b^\pi - \hat{V}_b^\pi$. We express the \hat{Q}_b^π and \hat{V}_b^π as the first-visit monte-carlo average of the observed returns, making sure to split the returns into independent random variables (since the returns used to estimate the Q -values and V -values may overlap). We then bound

the advantage by using the fact that the advantage is zero for (s, a) pairs for the states we defer, and bounded for the (s, a) pairs that are observed at least N_\wedge times. (See Appendix 10 for the full proof.)

Discussion. Our bound is a function of hyperparameters N_\wedge and δ , and the data-dependent term $C(N_\wedge)$. The $C(N_\wedge)$ term in our bound is a count of the number of (s, a) pairs that are observed at least N_\wedge times in the dataset. This term is much smaller than $|\mathcal{S}| |\mathcal{A}|$ when the behavior policy visits only a small subset of the state-action space. The hyperparameter N_\wedge allows us to directly control the trade-off between high-confidence policy improvement (high N_\wedge) and higher performance improvement at the cost of safety (low N_\wedge).

Comparison to baselines. We summarize the main differences between our bound and prior work here. The bounds in prior work are reproduced in Appendix 11 for reference. Most importantly, our dependence on $|\mathcal{S}|$ and $|\mathcal{A}|$ comes indirectly through the $C(N_\wedge)$ term, which differs from the direct dependence on $|\mathcal{S}|$ and $|\mathcal{A}|$ by SPI and pessimism-based methods (Liu et al., 2020; Kim and Oh, 2023). Thus, our bound is much tighter when the behavior policy has only visited a small subset of the state-action space more than N_\wedge times: we scale in the number of visited parts of the state-action space $C(N_\wedge)$ rather than $|\mathcal{S}| |\mathcal{A}|$. This difference would be most present when the behavior policy and transition dynamics are close to deterministic, or when the size of dataset is small relative to the size of the state-action space. On the other hand, when the size of the dataset is large and the behavior distribution is closer to uniform, all bounds will be tight. Our dependence on the N_\wedge parameter and effective horizon $1/(1 - \gamma)$ matches the SPI (Laroche et al., 2019; Scholl et al., 2022; Wienhöft et al., 2023b) literature. However, that we do not require access to π_b . Unlike Corollary 2 of (Liu et al., 2020), our bound does not have a dependence on the threshold b of the state-action density $\hat{\mu}(s, a)$, which has a direct correspondence to our N_\wedge parameter as $b = N_\wedge/|D|$. As a result, their bound gets looser as b gets smaller. This happens when the size of the dataset gets larger while keeping the set of (s, a) pairs with N_\wedge observations unchanged (i.e., more trajectories were added in the low-density regions). Our DPRL bound is unaffected by this superfluous extra data.

Continuous case: DPRL-C. The following theorem proves that π_{DP} is a safe policy improvement over the behavior π_b in the continuous case. Define $M(r, N_\wedge)$ as the number of balls of radius r needed to cover the subset of $\mathcal{X} \subset \mathcal{S} \times \mathcal{A}$ where each $(s, a) \in \mathcal{X}$ has at least N_\wedge data points in the ball $B_r(s, a)$. Similarly, define $M(r)$ as the number of balls of radius r

needed to cover the entire state-action space. Also define ϵ_r as the maximum error in the Q -values in the ball $B_r(s, a)$. Finally, we assume that the error in estimating the action-values is bounded for all points $(s', a') \in B_r(s, a)$: $|Q(s, a) - Q(s', a')| \leq \epsilon_r$

Theorem 2 (DPRL Continuous) *Let the constant $M(r, N_\wedge)$ be a measure of the volume on $\mathcal{S} \times \mathcal{A}$ that the dataset \mathcal{D} covers, and let the error in estimating the Q -values using a neighbor is bounded by ϵ_r , then π_{DP} is a safe policy improvement over the behavior policy π_b . That is, with probability at least $1 - \delta$:*

$$\rho(\pi_{DP}) - \rho(\pi_b) \geq -\frac{V_{\max}}{1-\gamma} \sqrt{\frac{1}{2N_\wedge} \log \frac{M(r, N_\wedge)}{\delta}} - 3\epsilon_r$$

Discussion. The $M(r, N_\wedge)$ term is a measure of the size of the region in the state-action space for which the policy improvement is guaranteed. For datasets with density only on a small subset of the state-action space, the $M(r, N_\wedge)$ term will be much smaller than $M(r)$. The second term in the bound quantifies the penalty paid for using neighborhood-based estimates of the Q -values. The hyperparameter N_\wedge influences the bound directly and indirectly through the $M(r, N_\wedge)$ term. Higher N_\wedge leads to a high-confidence estimation of the Q -values, and also leads to a smaller $M(r, N_\wedge)$ term since fewer (s, a) pairs can be improved upon. The hyperparameter r influences the bound through the $M(r, N_\wedge)$ term and the ϵ_r term. Smaller r leads to a low-bias estimate of the Q -values, but also leads to a larger $M(r, N_\wedge)$ term.

Comparison to baselines. DeepAveragers (Shrestha et al., 2020) (which uses kNN to identify neighbors) also have a dependence on a term similar to $M(r, N_\wedge)$ in our bound. However, their term approaches $M(r)$ if the dataset is sparse in the sense that each (s, a) pair has very few neighborhood points in the dataset (for DPRL, the $M(r, N_\wedge)$ will be close 0 in this case). The $\log M(r, N_\wedge)$ term in our bound is a non-parametric alternative to the $\log |\mathcal{F}|$ term in bounds for methods which use a parametric function class \mathcal{F} (e.g., (Liu et al., 2020)) The inherent trade-off is that the parametric methods optimize for a *global* estimation error $\epsilon_{\mathcal{F}}$ over the dataset (and hence have a dependence on terms such as $\epsilon_{\mathcal{F}}$, $|\mathcal{F}|$ and $|\mathcal{D}|$), while our non-parametric method optimizes for a *local* estimation error over the neighborhood of each (s, a) pair (and hence has a dependence on ϵ_r , $M(r, N_\wedge)$ and N_\wedge). For large datasets with uniform exploration, both parametric and non-parametric methods can have a low generalization error over the dataset, and thus the improvement is similar. However, for small datasets or for datasets with non-uniform exploration, parametric methods can have a looser bound because $\epsilon_{\mathcal{F}}$ can be large, while our non-parametric method can have a tighter bound because $M(r, N_\wedge)$ can be

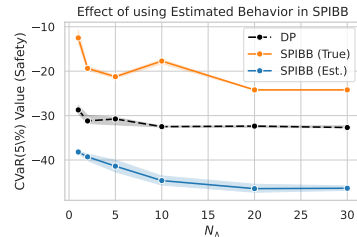


Figure 3: Gridworld: Replacing true behavior with estimated behavior in SPIBB leads to degraded CVaR. DP performs better than SPIBB without access to the true behavior.

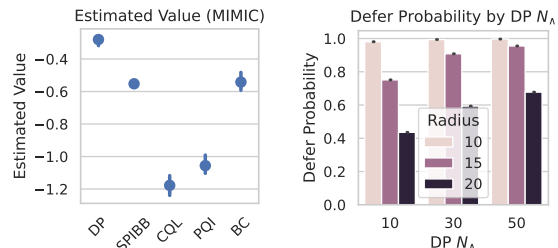


Figure 4: (Left) OPE value estimates of the learned policies on the MIMIC dataset: DPRL achieves the highest estimated value. (Right) The reason for DPRL’s strong performance: for the chosen hyperparameters ($N_\wedge = 50$, $r = 10$), DPRL defers to the behavior in nearly all states except where it is confident it can achieve a better outcome.

much smaller because we only incur the errors in dense regions of the dataset. Finally, we note that our bound is more actionable than the bounds in prior work because we can estimate $M(r, N_\wedge)$ using DBSCAN (Ester et al., 1996) (by computing the total volume occupied by the core points).

7 Experimental Evaluation

We first evaluate the DPRL on datasets from discrete-state-action MDPs and GridWorld and from

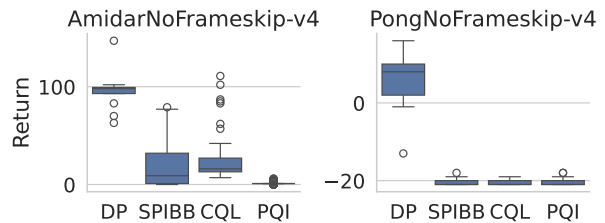


Figure 5: DP consistently learns good policies from suboptimal behavior data across Atari environments. Each algorithm is trained on 100,000 samples and evaluated on 20 episodes after training.

continuous-state, discrete-action Atari environments, where we can use the simulators to accurately estimate performance. Finally, we use our algorithm on continuous-state, discrete-action-real-world dataset of hypotensive patients in the ICU. We compare DPRL to SPIBB (Laroche et al., 2019), PQI (Liu et al., 2020), and CQL (Kumar et al., 2020), which are typical batch RL methods used for safe policy improvement and span the density-based, pessimism-based, and support-based constraints.

In what follows, we briefly describe the domain and then discuss key results from our experiments. Specifically, we consider results over discrete state and action spaces using Toy MDP and Gridworld as domains; as well as continuous state and discrete action spaces based on Atari and MIMIC III.

Discrete State and Action Spaces: MDP and GridWorld

We describe the MDPs in Section 4. For both MDPs, we vary the number of trajectories in the datasets. For our 10×10 GridWorld, we create datasets by simulating a ‘careless’ expert which is optimal everywhere except in a few states, where it chooses the worst action with probability 0.9 (marked orange in Figure 2). We sample datasets of n trajectories, $n \in \{10, 25, 50, 100, 500\}$.

DPRL achieves better theoretical bounds. On the MDP and GridWorld (Figures 1 and 2), we see that the DPRL provides safe policy improvement (CVaR). In GridWorld plots, we see that the CVaR without mean value being affected. The N_\wedge parameter allows us to control the trade-off between performance and safety, as shown in Figure 2(right): as N_\wedge increases, the mean value (performance) decreases, but the safety increases. On both the MDP and GridWorld, we see that the DPRL provides tighter bounds compared to the baselines, as shown in Figure 1(center). The bounds are tighter because the $C(N_\wedge)$ term in our bound is much smaller than the $|\mathcal{S}| |\mathcal{A}|$ term in the SPIBB and PQI bounds. Also, since the bound is data-dependent, it does not degrade as quickly as SPIBB and PQI for increase in $|\mathcal{S}|$.

DPRL better manages the bias/variance trade-off than existing methods. Figure 2(center) provides an insight into how N_\wedge achieves the trade-off between safety and performance. As N_\wedge increases, the fraction of states where the optimal action is allowed decreases, leading to increase in bias. At the same time, the fraction of states where a better action is chosen increases (considering states where multiple actions are allowed). This suggests reduced variance in value estimation.

Atari and Hypotension Datasets

DPRL achieves good performance in high-dimensional settings. On Atari domains, in Figure 5, we observe that DPRL achieves good performance even with 100K samples and a ‘careless’ expert who takes the worst action 50% of the time. In the hypotension dataset with continuous states, DPRL is the only method that achieves higher OPE estimates than the behavior baseline (Figure 4). We also see how the ability to defer can be useful in such a complex task: for the chosen parameters $N_\wedge = 50, r = 10$, DPRL defers more than 95% of the time! Such a policy is also actionable, since the physician can review the actions much faster than with the other methods.

8 Limitations and Discussion

The DPRL approach we present is non-parametric and requires storing the entire training data which can be costly. While we used BallTrees for efficient neighbor search, it would be interesting to explore how compression methods (e.g., coresets) can be used to store only a subset of the data. DPRL-C uses the euclidean distance with continuous states. Future work can look at more sophisticated metrics (e.g. bisimulation distance) and explore their influence on the safety bounds. Finally, we developed the SMDP formulation for the DPRL-D algorithm which performed multi-step planning but the DPRL-C algorithm only performed 1-step planning over behavior. In the future, we plan to extend the DPRL-C algorithm to multi-step planning while still maintaining safety guarantees.

9 Conclusion and Future Work

We introduced a decision points-based RL algorithm for performing safe policy improvement with guarantees. Our approach explicitly constrains the set of state-action pairs or regions of states considered to those areas that are most densely visited, while leveraging data from sparsely visited states to determine the ways in which we may deviate from the behaviour policy. Our experiments demonstrated that our approach lead to safer policy improvement with more confident estimates and tighter guarantees on policy improvement. Future work could explore how to extend the DPRL-C algorithm to multi-step planning while still maintain safety guarantees and utilize embeddings which are more sophisticated than euclidean distance and to determine their influence on the safety bounds.

References

- Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. *Reinforcement Learning: Theory and Algorithms*. 200.
- Steven Bradtke and Michael Duff. Reinforcement Learning Methods for Continuous-Time Markov Decision Problems. In *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994. URL <https://proceedings.neurips.cc/paper/1994/hash/07871915a8107172b3b5dc15a6574ad3-Abstract.html>.
- Ian Char, Viraj Mehta, Adam Villafior, John M. Dolan, and Jeff Schneider. BATS: Best Action Trajectory Stitching, April 2022. URL <http://arxiv.org/abs/2204.12026>. arXiv:2204.12026 [cs].
- Martin Ester, Hans-Peter Kriegel, Jorg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.
- Yuwei Fu, Wu Di, and Benoit Boulet. Batch reinforcement learning in the real world: A survey. In *Offline RL Workshop, NeurIPS*.
- Scott Fujimoto, David Meger, and Doina Precup. Off-Policy Deep Reinforcement Learning without Exploration. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2052–2062. PMLR, May 2019. URL <https://proceedings.mlr.press/v97/fujimoto19a.html>. ISSN: 2640-3498.
- Javier Garcia and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.
- Mohammad Ghavamzadeh, Marek Petrik, and Yinlam Chow. Safe Policy Improvement by Minimizing Robust Baseline Regret. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL <https://proceedings.neurips.cc/paper/2016/hash/9a3d458322d70046f63dfd8b0153ece4-Abstract.html>.
- Omer Gottesman, Yao Liu, Scott Sussex, Emma Brunskill, and Finale Doshi-Velez. Combining parametric and nonparametric models for off-policy evaluation. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2366–2375. PMLR, May 2019. URL <https://proceedings.mlr.press/v97/gottesman19a.html>. ISSN: 2640-3498.
- Omer Gottesman, Joseph Futoma, Yao Liu, Sonali Parbhoo, Leo Anthony Celi, Emma Brunskill, and Finale Doshi-Velez. Interpretable Off-Policy Evaluation in Reinforcement Learning by Highlighting Influential Transitions, August 2020. URL <http://arxiv.org/abs/2002.03478>. arXiv:2002.03478 [cs, stat].
- Bruce Hajek and Maxim Raginsky. ECE 543: Statistical Learning Theory, 2019. URL <https://courses.engr.illinois.edu/ece543/sp2019/SLT.pdf>.
- Charles A. Hepburn and Giovanni Montana. Model-based Trajectory Stitching for Improved Offline Reinforcement Learning, November 2022. URL <http://arxiv.org/abs/2211.11603>. arXiv:2211.11603 [cs, stat].
- Nan Jiang and Lihong Li. Doubly Robust Off-policy Value Evaluation for Reinforcement Learning.
- Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, et al. Mimic-iv, a freely accessible electronic health record dataset. *Scientific data*, 10(1):1, 2023.
- Shalmali Joshi, Sonali Parbhoo, and Finale Doshi-Velez. Learning-to-defer for sequential medical decision-making under uncertainty. *Transactions on Machine Learning Research*, December 2022. ISSN 2835-8856. URL <https://openreview.net/forum?id=0pn3KnbH5F>.
- 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*, pages 651–673. PMLR, 2018.
- Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOREL : Model-Based Offline Reinforcement Learning, March 2021. URL <http://arxiv.org/abs/2005.05951>. arXiv:2005.05951 [cs, stat].
- Byeongchan Kim and Min-Hwan Oh. Model-based Offline Reinforcement Learning with Count-based Conservatism. In *Proceedings of the 40th International Conference on Machine Learning*, pages 16728–16746. PMLR, July 2023. URL <https://proceedings.mlr.press/v202/kim23q.html>. ISSN: 2640-3498.
- Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019.
- Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning

- for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc., 2020. URL <https://proceedings.neurips.cc/paper/2020/hash/0d2b2061826a5df3221116a5085a6052-Abstract.html>.
- Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. When Should We Prefer Offline Reinforcement Learning Over Behavioral Cloning?, April 2022. URL <http://arxiv.org/abs/2204.05618>. arXiv:2204.05618 [cs].
- Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, pages 45–73. Springer, 2012.
- Romain Laroche, Paul Trichelair, and Rémi Tachet des Combes. Safe Policy Improvement with Baseline Bootstrapping, June 2019. URL <http://arxiv.org/abs/1712.06924>. arXiv:1712.06924 [cs, stat] version: 5.
- Armin Lederer, Jonas Umlauf, and Sandra Hirche. Uniform Error Bounds for Gaussian Process Regression with Application to Safe Control. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/fe73f687e5bc5280214e0486b273a5f9-Abstract.html.
- Lihong Li, Michael L. Littman, Thomas J. Walsh, and Alexander L. Strehl. Knows what it knows: a framework for self-aware learning. *Machine Learning*, 82(3):399–443, March 2011. ISSN 1573-0565. doi: 10.1007/s10994-010-5225-4. URL <https://doi.org/10.1007/s10994-010-5225-4>.
- Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably Good Batch Off-Policy Reinforcement Learning Without Great Exploration. In *Advances in Neural Information Processing Systems*, volume 33, pages 1264–1274. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/0dc23b6a0e4abc39904388dd3ffadcd1-Abstract.html.
- David Madras, Toni Pitassi, and Richard Zemel. Predict responsibly: improving fairness and accuracy by learning to defer. *Advances in neural information processing systems*, 31, 2018.
- Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert. In *International Conference on Machine Learning*, pages 7076–7087. PMLR, 2020.
- Kimia Nadjahi, Romain Laroche, and Rémi Tachet des Combes. Safe Policy Improvement with Soft Baseline Bootstrapping, July 2019. URL <https://arxiv.org/abs/1907.05079v1>.
- Philipp Scholl, Felix Dietrich, Clemens Otte, and Steffen Udfluft. Safe Policy Improvement Approaches and their Limitations, August 2022. URL <http://arxiv.org/abs/2208.00724>. arXiv:2208.00724 [cs] version: 1.
- Aayam Shrestha, Stefan Lee, Prasad Tadepalli, and Alan Fern. DeepAveragers: Offline Reinforcement Learning by Solving Derived Non-Parametric MDPs, October 2020. URL <http://arxiv.org/abs/2010.08891>. arXiv:2010.08891 [cs, stat].
- Anikait Singh, Aviral Kumar, Quan Vuong, Yevgen Chebotar, and Sergey Levine. Offline RL With Realistic Datasets: Heteroskedasticity and Support Constraints, November 2022. URL <http://arxiv.org/abs/2211.01052>. Issue: arXiv:2211.01052 arXiv:2211.01052 [cs, stat].
- Eleni Straitouri, Adish Singla, Vahid Balazadeh Meresht, and Manuel Gomez-Rodriguez. Reinforcement Learning Under Algorithmic Triage, September 2021. URL <http://arxiv.org/abs/2109.11328>. arXiv:2109.11328 [cs].
- Richard S Sutton. Between mdps and semi-mdps: Learning, planning, and representing knowledge at multiple temporal scales. 1998.
- Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- Philip Thomas, Georgios Theodorou, and Mohammad Ghavamzadeh. High-Confidence Off-Policy Evaluation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), February 2015a. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v29i1.9541. URL <https://ojs.aaai.org/index.php/AAAI/article/view/9541>.
- Philip Thomas, Georgios Theodorou, and Mohammad Ghavamzadeh. High Confidence Policy Improvement. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2380–2388. PMLR, June 2015b. URL <https://proceedings.mlr.press/v37/thomas15.html>. ISSN: 1938-7228.
- Patrick Wienhöft, Marnix Suilen, Thiago D. Simão, Clemens Dubsflaff, Christel Baier, and Nils Jansen. More for Less: Safe Policy Improvement With Stronger Performance Guarantees, May 2023a. URL <http://arxiv.org/abs/2305.07958>. arXiv:2305.07958 [cs].

Patrick Wienhöft, Marnix Suilen, Thiago D. Simão, Clemens Dubschlaff, Christel Baier, and Nils Jansen. More for less: safe policy improvement with stronger performance guarantees. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23, pages 4406–4415, <conf-loc>, <city>Macao</city>, <country>P.R.China</country>, </conf-loc>, August 2023b. ISBN 978-1-956792-03-4. doi: 10.24963/ijcai.2023/490. URL <https://doi.org/10.24963/ijcai.2023/490>.

Jialong Wu, Haixu Wu, Zihan Qiu, Jianmin Wang, and Mingsheng Long. Supported policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:31278–31291, 2022.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based Offline Policy Optimization, November 2020. URL <http://arxiv.org/abs/2005.13239>. arXiv:2005.13239 [cs, stat].

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: Conservative Offline Model-Based Policy Optimization, January 2022. URL <http://arxiv.org/abs/2102.08363>. arXiv:2102.08363 [cs].

Kristine Zhang, Henry Wang, Jianzhun Du, Brian Chu, Aldo Robles Arévalo, Ryan Kindle, Leo Anthony Celi, and Finale Doshi-Velez. An interpretable RL framework for pre-deployment modeling in ICU hypotension management. *npj Digital Medicine*, 5(1):1–10, November 2022. ISSN 2398-6352. doi: 10.1038/s41746-022-00708-4. URL <https://www.nature.com/articles/s41746-022-00708-4>. Publisher: Nature Publishing Group.

10 Proofs

Lemma 3 (Performance Difference Lemma) *Let π_{DP} and π_b be two policies. Then, the difference in performance between the two policies is given by:*

$$\rho(\pi_{DP}) - \rho(\pi_b) = \frac{1}{1-\gamma} \mathbb{E}_{\eta^{\pi_{DP}(s,a)}} [A^{\pi_b}(s, a)] \quad (11)$$

where $A^{\pi_b}(s, a)$ is the advantage of taking action a in state s under policy π_b .

Proof: See Lemma 1.16 of Agarwal et al. (200). \square

Lemma 4 (McDiarmid's Inequality) *Suppose a function $f : \mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_n \rightarrow \mathbb{R}$ satisfies the bounded differences property if:*

$$\sup_{x'_i, x_i \in \mathcal{X}_i} f(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n) - f(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) \leq c_i. \quad (12)$$

for all $i \in [n]$ and constants c_1, \dots, c_n . Let $X = (X_1, X_2, \dots, X_n)$ be an n -tuple of independent random variables, where $X_i \in \mathcal{X}_i$ for all $i \in [n]$. Then, for any $\epsilon > 0$,

$$\Pr[|f(X) - \mathbb{E}[f(X)]| \geq \epsilon] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^n c_i^2}\right) \quad (13)$$

Proof: See Theorem 2.3 of (Hajek and Raginsky, 2019) \square

Theorem 1 (DPRL Discrete) *Let π_{DP} be the policy obtained by the DP algorithm. Then π_{DP} is a safe policy improvement over the behavior policy π_b , with probability at least $1 - \delta$*

$$\rho(\pi_{DP}) - \rho(\pi_b) \geq -\frac{V_{\max}}{1-\gamma} \sqrt{\frac{1}{N_\wedge} \log \frac{C(N_\wedge)}{\delta}} \quad (9)$$

where $C(N_\wedge)$ is the count of the number of (s, a) pairs that are observed at least N_\wedge times in the dataset:

$$C(N_\wedge) = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mathbf{I}[n(s, a) \geq N_\wedge] \quad (10)$$

Proof: To prove the theorem, we first define a few useful quantities. Let $\Omega_s \subset \{1, \dots, N\}$ be the set of trajectories that visit state s and let $\Omega_{s,a} \subset \Omega_s$ be the set of trajectories that contain the (s, a) pair. For $n \in \Omega_s$, let t_n be the first time step that the state s is visited in the trajectory n , i.e., $S_{t_n}^n = s$. For $n \in \Omega_{s,a}$, let t_n^a be the first time step that the action a is taken in the state s in the trajectory n , i.e., $(S_{t_n^a}^n, A_{t_n^a}^n) = (s, a)$. Therefore, $t_n^a \geq t_n$. Let G_t^n be the return of the trajectory n starting from time step t , i.e., $G_t^n = \sum_{k=t}^{T_n} \gamma^{k-t} R_k^n$. We define the value function estimators $\hat{V}^{\pi_b}(s)$ and $\hat{Q}^{\pi_b}(s, a)$ as follows:

$$\hat{V}^{\pi_b}(s) = \frac{1}{|\Omega_s|} \sum_{n \in \Omega_s} G_{t_n}^n, \quad \hat{Q}^{\pi_b}(s, a) = \frac{1}{|\Omega_{s,a}|} \sum_{n \in \Omega_{s,a}} G_{t_n^a}^n \quad (14)$$

$$\hat{A}^{\pi_b}(s, a) = \hat{Q}^{\pi_b}(s, a) - \hat{V}^{\pi_b}(s) \quad (15)$$

Therefore, we have the following expectations:

$$\mathbb{E}[G_{t_n}^n] = V^{\pi_b}(s), \quad \text{and} \quad \mathbb{E}[G_{t_n^a}^n] = Q^{\pi_b}(s, a) \quad (16)$$

$$\implies \mathbb{E}[\hat{V}^{\pi_b}(s)] = V^{\pi_b}(s), \quad \mathbb{E}[\hat{Q}^{\pi_b}(s, a)] = Q^{\pi_b}(s, a), \quad \mathbb{E}[\hat{A}^{\pi_b}(s, a)] = A^{\pi_b}(s, a) \quad (17)$$

and for any $n \in \Omega_{s,a}$,

$$G_{t_n}^n = G_{t_n:t_n^a}^n + \gamma^{t_n^a - t_n} G_{t_n^a}^n \quad (18)$$

where $G_{t_n:t_n^a}^n$ is the return of the trajectory n from time step t_n to t_n^a (zero if $t_n = t_n^a$).

Now, we can write the empirical advantage $\hat{A}^{\pi_b}(s, a)$ as:

$$\hat{A}^{\pi_b}(s, a) = \frac{1}{|\Omega_{s,a}|} \sum_{n \in \Omega_{s,a}} G_{t_n^a}^n - \frac{1}{|\Omega_s|} \sum_{n \in \Omega_s} G_{t_n}^n \quad (19)$$

$$= \sum_{n \in \Omega_{s,a}} \frac{G_{t_n^a}^n}{|\Omega_{s,a}|} - \frac{\gamma^{t_n^a - t_n} G_{t_n^a}^n}{|\Omega_s|} - \sum_{n \in \Omega_s} \frac{G_{t_n:t_n^a}^n}{|\Omega_s|} - \sum_{n \in \Omega_s \setminus \Omega_{s,a}} \frac{G_{t_n}^n}{|\Omega_s|} \quad (20)$$

$$= S_1(\{G_{t_n^a}^n\}_{n \in \Omega_{s,a}}) + S_2(\{G_{t_n:t_n^a}^n\}_{n \in \Omega_s}) + S_3(\{G_{t_n}^n\}_{n \in \Omega_s}) \quad (21)$$

where all the random variables $\mathcal{G} := \{G_{t_n^a}^n\}_{n \in \Omega_{s,a}} \cup \{G_{t_n:t_n^a}^n\}_{n \in \Omega_s} \cup \{G_{t_n}^n\}_{n \in \Omega_s}$ are independent and bounded in $[0, V_{\max}]$.

We can now apply McDiarmid's inequality to the function $f(\{G_{t_n^a}^n\}_{n \in \Omega_{s,a}}, \{G_{t_n:t_n^a}^n\}_{n \in \Omega_s}, \{G_{t_n}^n\}_{n \in \Omega_s}) = S_1 + S_2 + S_3$. Therefore, we have:

$$\Pr[f(\mathcal{G}) - \mathbb{E}[f(\mathcal{G})] \geq \epsilon] \leq \exp\left(-\frac{2\epsilon^2 N_\wedge}{2V_{\max}^2}\right) \quad (22)$$

$$\implies \Pr[A^{\pi_b}(s, a) \geq -\epsilon - \hat{A}^{\pi_b}(s, a)] \leq \exp\left(-\frac{2\epsilon^2 N_\wedge}{2V_{\max}^2}\right) \quad (23)$$

$$\implies \Pr[A^{\pi_b}(s, a) \geq -\epsilon] \leq \exp\left(-\frac{2\epsilon^2 N_\wedge}{2V_{\max}^2}\right) \quad (24)$$

where the last inequality follows from the fact that $\hat{A}^{\pi_b}(s, a) \geq 0$ and N_\wedge is the minimum number of times the (s, a) pair is observed in the dataset.

Apply the union bound over all valid state-action pairs $C(N_\wedge) = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mathbf{I}[n(s, a) \geq N_\wedge]$ to get the desired result that with probability at least $1 - \delta$:

$$A^{\pi_b}(s, a) \geq -V_{\max} \sqrt{\frac{1}{N_\wedge} \log \frac{C(N_\wedge)}{\delta}} \quad (25)$$

Note that we did not need to apply the union bound over the rest of the state-action pairs because the advantage is zero as $\pi_{\text{DP}} = \pi_b$ in those states.

Finally, we can use the performance difference lemma to get the desired result:

$$\rho(\pi_{\text{DP}}) - \rho(\pi_b) = \frac{1}{1-\gamma} \mathbb{E}[A^{\pi_b}(s, a)] \geq -\frac{V_{\max}}{1-\gamma} \sqrt{\frac{1}{2N_\wedge} \log \frac{C(N_\wedge)}{\delta}} \quad (26)$$

□

Theorem 2 (DPRL Continuous) *Let the constant $M(r, N_\wedge)$ be a measure of the volume on $\mathcal{S} \times \mathcal{A}$ that the dataset \mathcal{D} covers, and let the error in estimating the Q -values using a neighbor is bounded by ϵ_r , then π_{DP} is a safe policy improvement over the behavior policy π_b . That is, with probability at least $1 - \delta$:*

$$\rho(\pi_{\text{DP}}) - \rho(\pi_b) \geq -\frac{V_{\max}}{1-\gamma} \sqrt{\frac{1}{2N_\wedge} \log \frac{M(r, N_\wedge)}{\delta}} - 3\epsilon_r$$

Proof: The proof for the discrete case is identical except that we pay a penalty of ϵ_r everytime we use (s, a) 's neighbor's action-value ($Q(s', a')$ for $(s', a') \in B_r(s, a)$) to estimate $Q(s, a)$. We create a covering \mathcal{C} of the dense region of the dataset. We assume $M(r, N_\wedge) \geq |\mathcal{C}|$

1. For any point (s, a) with at least N_\wedge neighbors in $B_r(s, a)$, the advantage $A\pi_b(s, a)$ is bounded from below because it is a monte-carlo average (like in discrete case analysis). However, using the returns of the neighbors incurs an ϵ_r error.

2. For any point $(s', a') \in \mathcal{C}$, there is at least one neighbor with at least N_\wedge neighbors. We incur another ϵ_r error. Then we apply union bound over $M(r, N_\wedge)$ points.
3. Finally, we incur another ϵ_r error for going from points in \mathcal{C} to all the points where $A^{\pi_b}(s, a) \geq 0$ (i.e. we don't defer).

□

11 Bounds in Prior Work

Here we reproduce the bounds in prior work for reference. The bound by (Laroche et al., 2019) (Theorem 2) is, with probability $1 - \delta$,

$$\rho(\hat{\pi}) - \rho(\pi_b) \geq -\frac{4V_{\max}}{1-\gamma} \sqrt{\frac{2}{N_\wedge} \log \frac{2|\mathcal{S}||\mathcal{A}|2^{|\mathcal{S}|}}{\delta}} - \rho(\pi, \hat{M}) + \rho(\pi_b, \hat{M}) \quad (27)$$

This bound also depends on the N_\wedge parameter and $(1 - \gamma)$ in the same way as our bound. However, the bound differs from ours because it has a dependence on the number of states and actions. This is much larger than the $C(N_\wedge)$ term in our bound, which only scales with the number of (s, a) pairs that are observed at least N_\wedge times.

The bound by (Liu et al., 2020) (Corollary 2) is, with probability $1 - \delta$,

$$\rho(\hat{\pi}) - \rho(\pi_b) \geq -\tilde{O} \left(\frac{V_{\max}}{b(1-\gamma)^3} \frac{|\mathcal{S}||\mathcal{A}|}{n} + \frac{V_{\max}}{b(1-\gamma)^3} \sqrt{\frac{|\mathcal{S}||\mathcal{A}|}{n}} + \frac{\gamma^K V_{\max}}{(1-\gamma)^2} \right) \quad (28)$$

This bound differs from ours because it has a dependence on $b := \min_{s,a} \eta^{\pi_b}(s, a)$, and the bound can be even more loose when $\hat{\eta}^{\pi_b}$ is used to estimate η^{π_b} . Furthermore, the bound also has a dependence on the number of states and actions, which can be large in practice. In contrast, our bound is directly in terms of the N_\wedge parameter, and does not have to assume any bounds on the data distribution. This allows for a tighter bound in practice even when the excluded (s, a) pairs are the same.

The bound by (Kim and Oh, 2023) (Theorem 2) is, with probability $1 - \delta$,

$$\rho(\hat{\pi}) - \rho(\pi_b) \geq -\frac{\gamma V_{\max}}{(1-\gamma)^2} \mathbb{E}_{(s,a) \sim \eta_{\hat{P}}^{\pi_b}} \left[\min \left(1, \sqrt{\frac{2}{n(s,a)} \log \frac{|\mathcal{S}||\mathcal{A}|}{\delta}} \right) \right] \quad (29)$$

where $\eta_{\hat{P}}^{\pi_b}(s, a)$ is the state-action visitation distribution under the behavior policy and the MLE transition model. This bound is similar to ours in that it depends on the count $n(s, a)$, but it has a dependence on the number of states and actions. Furthermore, the dependence on $n(s, a)$ is only useful when the count is large, and the bound can be loose when there are many (s, a) pairs with low counts. We avoid this by directly controlling the set of (s, a) pairs that are included in the policy set.

12 Algorithms

12.1 Algorithm: DPRL-Discrete

Algorithm 1 DPRL-D for Discrete States

- 1: **Input:** Dataset, $\mathcal{D} = \{(S_t^n, A_t^n, R_t^n)\}$
 - 2: Compute $\hat{Q}^{\pi_b}, \hat{V}^{\pi_b}$ using the dataset \mathcal{D}
 - 3: Compute $\mathcal{S}^{\text{DP}}, \mathcal{A}^{\text{DP}} = \{\mathcal{A}_s^{\text{DP}} : s \in \mathcal{S}^{\text{DP}}\}$ using Eq 5
 - 4: $\tilde{P}, \tilde{R}, \tilde{\gamma} \leftarrow$ Make SMDP Parameters($\mathcal{D}, \mathcal{S}^{\text{DP}}$)
 - 5: **Policy Iteration:**
 - 6: **Initialize:** $i \leftarrow 1$ and $V^{(1)} \leftarrow \hat{V}^{\pi_b}$
 - 7: $\pi^{(1)}(s) \leftarrow \arg \max_{a \in \mathcal{A}_s^{\text{DP}}} \hat{Q}^{\pi_b}(s, a) \quad \forall s \in \mathcal{S}^{\text{DP}}$
 - 8: **repeat**
 - 9: $i \leftarrow i + 1$
 - 10: $V^{(i)} \leftarrow$ PolicyEval $[\tilde{P}, \tilde{R}, \pi^{(i)}]$
 - 11: Update $\pi^{(i)}(s)$ using Eq 5
 - 12: **until** $\|V^{(i)} - V^{(i-1)}\|_{\infty} \leq \varepsilon$
 - 13: **Output:** $\pi^{(i)}$
-

12.2 Algorithm to Construct SMDP Parameters for Discrete States

Algorithm 2 Make SMDP Parameters

- 1: **Input:** Dataset, $\mathcal{D} = \{(S_t^n, A_t^n, R_t^n) : n = 1, \dots, N, t = 1, \dots, T_n\}$
 - 2: **Input:** Decision points, $\mathcal{S}^{\text{DP}} \subset \mathcal{S}$
 - 3: $\mathcal{S}^{\text{DP}} \leftarrow$ set of all possible states
 - 4: **for** n in $[1, \dots, N]$ **do**
 - 5: $\tau^n \leftarrow$ dict()
 - 6: **for** t in $[1, T_n]$ **do**
 - 7: **if** $S_t^n \notin \tau^n$ and $S_t^n \in \mathcal{S}^{\text{DP}}$ **then**
 - 8: $\tau^n[S_t^n] \leftarrow t$
 - 9: **end if**
 - 10: $\tau_{\text{sorted}}^n \leftarrow \text{sort}(\tau^n)$
 - 11: **for** t, t' in zip($\tau_{\text{sorted}}^n[: -1], \tau_{\text{sorted}}^n[1 :]$) **do**
 - 12: $Y(S_t^n, A_t^n, S_{t'}^n) \leftarrow Y(S_t^n, A_t^n, S_{t'}^n) + \gamma^{t'-t}$
 - 13: $G(S_t^n, A_t^n, S_{t'}^n) \leftarrow G(S_t^n, A_t^n, S_{t'}^n) + \sum_{k=t}^{t'} \gamma^{k-t} R_k^n$
 - 14: $\tilde{n}(S_t^n, A_t^n, S_{t'}^n) \leftarrow \tilde{n}(S_t^n, A_t^n, S_{t'}^n) + 1$
 - 15: **end for**
 - 16: **end for**
 - 17: **end for**
 - 18: $\tilde{\gamma}(s, a, s') \leftarrow Y(s, a, s') / \tilde{n}(s, a, s')$
 - 19: $\tilde{P}(s' | s, a) \leftarrow \tilde{n}(s, a, s') / \sum_{s' \in \mathcal{S}^{\text{DP}}} \tilde{n}(s, a, s')$
 - 20: $\tilde{r}(s, a, s') \leftarrow G(s, a, s') / \tilde{n}(s, a, s')$
 - 21: $\tilde{R}(s, a) \leftarrow \sum_{s' \in \mathcal{S}^{\text{DP}}} \tilde{r}(s, a, s') \tilde{P}(s, a, s')$
 - 22: **Output:** $\tilde{P}, \tilde{R}, \tilde{\gamma}$
-

12.3 Algorithm: DPRL-Continuous

Algorithm 3 DPRL-C for Continuous States

- 1: **Input:** State s
 - 2: **Input:** $\text{BallTree}_{sa}(\cdot, \mathcal{D}, r)$, $\text{BallTree}_s(\cdot, \mathcal{D}, r)$
 - 3: $\mathcal{N}(s) \leftarrow \text{BallTree}_s(s, \mathcal{D}, r)$
 - 4: $\mathcal{N}(s, a) \leftarrow \text{BallTree}_{sa}((s, a), \mathcal{D}, r)$ for all $a \in \mathcal{A}$
 - 5: **if** $|\mathcal{N}(s)| \leq N_\wedge$ **then**
 - 6: **Output:** DEFER
 - 7: **else**
 - 8: Compute $\hat{V}^{\pi_b}(s)$ using $\mathcal{N}(s)$
 - 9: Compute $\hat{Q}^{\pi_b}(s, a)$ using $\mathcal{N}(s, a)$
 - 10: Compute $\mathcal{A}_s^{\text{DP}}$ using Eq 7
 - 11: **if** $\mathcal{A}_s^{\text{DP}} = \emptyset$ **then**
 - 12: **Output:** DEFER
 - 13: **else**
 - 14: **Output:** $\arg \max_{a \in \mathcal{A}_s^{\text{DP}}} \hat{Q}^{\pi_b}(s, a)$
 - 15: **end if**
 - 16: **end if**
-

13 Details about the Synthetic MDPs

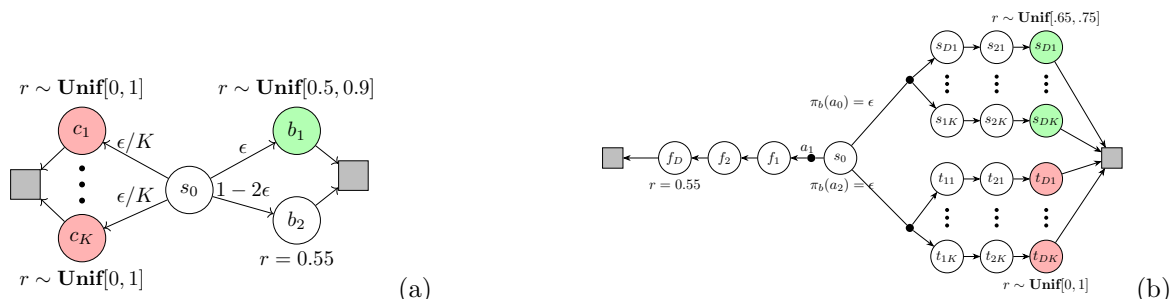


Figure 6: (a) Toy MDP for CQL: The optimal action leads to state b_1 , and the suboptimal action (frequent under behavior) leads to state b_2 . There are also risky states $\{c_1, \dots, c_K\}$. (b) Toy MDP for PQI: The optimal action (a_0) leads to the upper branch with two parallel chains indexed by depth (D) and chain number (K). a_1 leads to the middle branch, and a_2 leads to the lower branch with two parallel chains indexed by depth (D) and chain number (K). All branches converge to the final state z . Intermediary dots indicate additional potential connections between parallel chains.

Experimental details for the bounds plot. To create the bounds plot, we simulated the forests MDP for 50 trials. For each trial, we generated a new dataset, and for each dataset, we computed the bounds for three methods: DP, SPIBB, and PQI. We varied parameters N_\wedge in $\{1, 10, 100\}$ and the number of states to observe their effect on the bounds (we varied the states by varying the number of chains in the ‘forests’ in $\{10, 20, 30, 50\}$). We set the density threshold for PQI to 0.02 for all the simulations.

14 Gridworld Experimental Details

The GridWorld has 100 states and 4 actions. The agent must start at the bottom left cell and reach the top right cell. The dynamics are stochastic with going to the intended cell with a probability of 0.9 and a simulating a random action otherwise. The rewards are stochastic, and described in Figure 2(A). We sampled 500 random datasets from the environment to evaluate the reliability of the algorithms. For DPRL, SPIBB and PQI, we tested the N_\wedge parameter in $\{1, 2, 5, 10, 20, 30\}$ and found $N_\wedge = 20$ to be good for all the algorithms. For CQL, we varied α in $\{0.01, 0.05, 0.1\}$.

15 Atari Experimental Details

We use Atari datasets and environments for our experiments, specifically focusing on five environments: **Qubert**, **Pong**, **Freeway**, **Booing**, and **Amidar**. These environments allow for a diverse range of behaviors and complexities to evaluate our methods effectively.

Policy Training

- **Optimal Policy:** We trained an optimal policy for each environment by running a **Deep Q-Network (DQN)** for **10 million steps**. The **Stable Baselines** implementation of DQN was used for this training.
- **Medium Policy:** In addition to the optimal policy, we defined a "medium" policy by using the state of the DQN after **1 million steps** of training, which represents a less performant but reasonable policy.

Trajectory Simulation We simulated trajectories for each environment with a structured deviation from the optimal policy:

- **Action Selection Probability:** For each action in the trajectory, the probability of taking the optimal action was set to **0.5**, while the probability of taking an action based on the medium policy was also set to **0.5**.
- **Data Generation:** We simulated **100,000 samples** for each environment, which correspond to a different number of episodes per environment depending on their characteristics.

This setup creates a structured deviation from the optimal policy, where the action taken can either be optimal or highly suboptimal, making the dataset suitable for evaluating off-policy methods.

Representation Learning To ensure consistency across all methods, we used a **16-dimensional representation** for state-action pairs, which were learned using the collected datasets. This step standardizes the inputs for the comparison of various algorithms.

- **State Representation:** We used the mean of all action representations corresponding to each state to derive the state representations.
- **Data Storage:** We utilized a **BallTree** data structure to store the data, enabling efficient nearest neighbor searches during evaluation.

Implementation Details We compared several methods using the learned state-action representations, with the following specific implementations:

- **PQI and SPIBB:** We used **ExtraTrees** (with **100 trees**) to estimate Q-values. ExtraTrees offers a highly flexible function approximation for this purpose.
- **CQL:** Since ExtraTrees is unsuitable for CQL (due to the nature of its loss function), we employed a **DQN architecture** with a single hidden layer of **16 nodes** for Q-value estimation.
- **PQI Density Estimation:** For PQI, we estimated the joint density of state-action pairs by counting the number of neighbors in the vicinity of each state-action pair using the **BallTree** data structure. This count was then divided by the dataset size to obtain the density estimate. Although this is a heuristic, it provides a feasible approach for density estimation in continuous state-action spaces.
 - **Note:** The original PQI paper employed a **variational auto-encoder** for density estimation, which was applied only to states, not state-action pairs.
- **SPIBB Behavior Policy:** We estimated the behavior policy using **random forests**, and the density of state-action pairs was computed in the same way as for PQI using the **BallTree** data structure.

Table 1: Best hyperparameter combinations for each environment and algorithm (Atari).

	hyperparameters	
AmidarNoFrameskip-v4	CQL	$\alpha = 0.0001$
	DP	$N_\wedge = 5.0, r = 0.0001$
	PQI	$b = 0.001, r = 0.001$
	SPIBB	$N_\wedge = 5.0, r = 0.001$
BowlingNoFrameskip-v4	CQL	$\alpha = 0.01$
	DP	$N_\wedge = 5.0, r = 0.0001$
	PQI	$b = 0.01, r = 0.001$
	SPIBB	$N_\wedge = 10.0, r = 0.001$
FreewayNoFrameskip-v4	CQL	$\alpha = 0.01$
	DP	$N_\wedge = 5.0, r = 0.0001$
	PQI	$b = 0.0001, r = 0.001$
	SPIBB	$N_\wedge = 30.0, r = 0.001$
PongNoFrameskip-v4	CQL	$\alpha = 0.1$
	DP	$N_\wedge = 5.0, r = 0.0001$
	PQI	$b = 0.001, r = 0.001$
	SPIBB	$N_\wedge = 10.0, r = 0.001$
QbertNoFrameskip-v4	CQL	$\alpha = 0.1$
	DP	$N_\wedge = 5.0, r = 0.0001$
	PQI	$b = 0.0001, r = 0.001$
	SPIBB	$N_\wedge = 30.0, r = 0.001$

Hyperparameters We conducted extensive experiments with the following hyperparameter settings (see Table 1 for the best hyperparameters):

- **DPRL:** $N_\wedge \in \{5, 10, 30\}$
- **SPIBB:** $N_\wedge \in \{5, 10, 30\}$
- **PQI:** Density threshold (b) $\in \{1e-4, 1e-3, 1e-2\}$
- **Radius for neighbor search:** $1e-3$ (for all methods)
- **CQL:** Alpha parameter $\alpha \in \{1e-4, 1e-2, 1e-1\}$

Table 2: Architecture used by each DQN and Representation Learning Network (Atari)

Layer	Number of outputs	Other details
Input frame size	(4x84x84)	–
Downscale convolution 1	12800	kernel 8x8, depth 32, stride 4x4
Downscale convolution 2	5184	kernel 4x4, depth 32, stride 2x2
Downscale convolution 3	3136	kernel 3x3, depth 32, stride 1x1
Layers in feedforward network of DQN	[512, 64, 16, $ A $]	–
Layers in feedforward network of representation learning model	[512, 16 * $ A $]	layer norm, tanh activation after the last layer

Table 3: All Hyperparameters for DQN used for training the expert policy

Hyperparameter	Value
Network optimizer	Adam
Learning rate	0.0001
Adam ϵ	0.000015
Discount γ	0.99
Mini-batch size	128
Target network update frequency	10k training iterations
Evaluation ϵ	0.001

15.1 Additional Plots - Atari

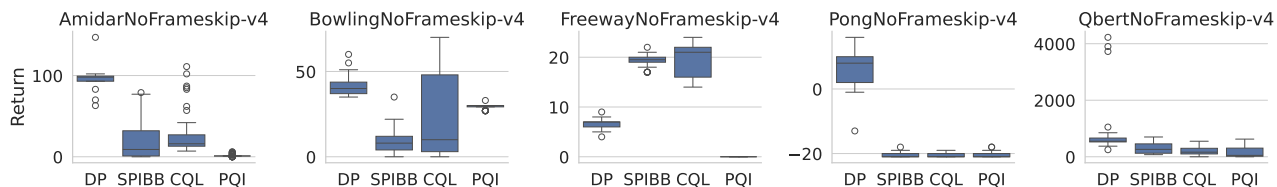


Figure 7: Performance comparison on the Atari domains.

16 MIMIC Dataset Experimental Details

Cohort and Environment Definition For the continuous states dataset, the EHR data contains deidentified clinical data of patients admitted to the Beth Israel Deaconess Medical Center ICU unit (Johnson et al., 2023). We select a cohort of patients with at least 7 mean arterial pressure (MAP) measurements of less than 65 mmHg (i.e. hypotension) within the first 72 hours of their ICU stay. The states have 11 features, and we add another 18 features as per prior work in the literature (Gottesman et al., 2020). The actions correspond to 4 levels intravenous (IV) fluid bolus therapy and 4 levels of vasopressor therapy (total 16 discrete actions). Fluid bolus and vasopressors are the first-line treatments for patients with hypotension in the ICU. We used 500 trajectories for training (28944 transitions), and 500 trajectories for validation (28710 transitions), and 1000 trajectories for final evaluation the performance of the algorithms (58033 transitions).

Preprocessing For our cohort, we selected the following features from the raw data for the analysis: *creatinine*, *fraction inspired oxygen*, *lactate*, *urine output*, *alanine aminotransferase*, *aspartate aminotransferase*, *diastolic blood pressure*, *mean blood pressure*, *partial pressure of oxygen*, *systolic blood pressure*, and *gcs*. These features have been selected for the hypotension management task in prior work Gottesman et al. (2020)

To create the distance function, we used a weighted Euclidean metric with weights derived from Gottesman et al. (2020) on this dataset and cohort. The final set of features includes the original features as well as a collection of handcrafted features, which were determined by domain experts to be relevant for the hypotension task. We provide the original set of features and the final set of features, along with their corresponding weights, in Table 4.

To ensure the state-action pairs are appropriately matched during the computation of distances, we assigned a weight of 10,000 to the action dimension when computing the distance between state-action pairs.

Training details For DPRL-C, we need to specify hyperparameters r and N_\wedge . We found $r = 10$ to give the best results. We varied the N_\wedge in $\{10, 30, 50\}$. For SPIBB and PQI, we need to provide similar sets of state-actions with sufficient count or density, and we use the same sets as the ones obtained using r and N_\wedge .

Evaluation To evaluate the MIMIC policies, we estimated the behavior policy π_b (using ExtraTrees) and the Q^π values (using Fitted Q-Evaluation with ExtraTrees). We then use Doubly Robust Off-policy evaluation (DR-OPE) (Jiang and Li) using these estimates. Note that we do expect π_b and Q^π estimates to be imperfect in

Table 4: Original and Handcrafted Features with Weights and Data Types (MIMIC Dataset)

Feature	Weight	Data Type	Present in the Raw Dataset
creatinine	3	float	Yes
fraction inspired oxygen	15	float	Yes
lactate	15	float	Yes
urine output	15	float	Yes
urine output since last action	5	bool	No
alanine aminotransferase	5	float	Yes
aspartate aminotransferase	5	float	Yes
diastolic blood pressure	5	float	Yes
mean blood pressure	15	float	Yes
partial pressure of oxygen	3	float	Yes
systolic blood pressure	5	float	Yes
gcs	15	float	Yes
gcs since last action	5	bool	No
creatinine ever recorded	3	bool	No
fraction inspired oxygen ever recorded	15	bool	No
lactate ever recorded	10	bool	No
alanine aminotransferase ever recorded	5	bool	No
aspartate aminotransferase ever recorded	5	bool	No
partial pressure of oxygen ever recorded	3	bool	No
flag vaso 1 was last action	15	bool	No
flag vaso 2 was last action	15	bool	No
flag vaso 3 was last action	15	bool	No
flag bolus 1 was last action	15	bool	No
flag bolus 2 was last action	15	bool	No
flag bolus 3 was last action	15	bool	No
total vasopressor dose	15	float	No
total bolus dose	15	float	No
total vasopressor dose last 8 hours	15	float	No
total bolus dose last 8 hours	15	float	No

parts of the state space due to the complexity and coverage challenges in this real-world dataset. Nevertheless, we used DR-OPE to mitigate some of the bias of the estimates.