Safe Policy Improvement with Baseline Bootstrapping

Anonymous Authors[1]

Abstract

A common goal in Reinforcement Learning is to derive a good strategy given a limited batch of data. In this paper, we adopt the safe policy improvement (SPI) approach: we compute a target policy guaranteed to perform at least as well as a given baseline policy. Our SPI strategy, inspired by the knows-what-it-knows paradigm, consists in bootstrapping the target policy with the baseline policy when it does not know. We develop two computationally efficient bootstrapping algorithms, a value-based and a policy-based, both accompanied with theoretical SPI bounds. Three algorithm variants are proposed. We empirically show the literature algorithms limits on a small stochastic gridworld problem, and then demonstrate that our algorithms both improve the worst case scenario and the mean performance.

1. Introduction

Reinforcement Learning (RL, (Sutton & Barto, 1998)) consists in discovering by *trial-and-error*, in an unknown uncertain environment, which action is the most valuable in a particular situation. In an online learning setting, trial-and-error works optimally, because a good outcome brings a policy improvement, and even an error leads to learning not to do it again at a lesser cost. However, most real-world algorithms are to be widely deployed on independent devices/systems, and as such their policies cannot be updated as often as online learning would require. In this offline setting, batch RL algorithms should be applied (Lange et al., 2012). But, the trial-and-error paradigm shows its limits when the policy updates are rare, because the commitment on the trial is too strong and the error impact may be severe. In this paper, we endeavour to build batch RL algorithms that are safe in this regard.

The notion of safety in RL has been defined in several contexts (Garcıa & Fernández, 2015). Two notions of uncertainty: the internal and the parametric, are defined in (Ghavamzadeh et al., 2016): *internal uncertainty reflects the uncertainty of the return due to stochastic transitions and rewards, for a single known MDP, while parametric uncertainty reflects the uncertainty about the unknown MDP parameters: the transition and reward distributions.* In short, internal uncertainty intends to guarantee a certain level of return for each individual trajectory (Schulman et al., 2015; 2017), which is critical in view of their potential harmful behaviour (Amodei et al., 2016) or in the catastrophe avoidance scenarios (Geibel & Wysotzki, 2005; Lipton et al., 2016). In this paper, we focus more specifically on the parametric uncertainty in order to guarantee a given expected return for the trained policy in the batch RL setting (Thomas et al., 2015a; Petrik et al., 2016).

More specifically, we seek high confidence that the trained policy approximately outperforms a given baseline policy (not necessarily the behavioural policy). The goal is therefore to improve the policy, even in the worst case scenario. As such, this family of algorithms can be seen as pessimistic: the *optimism in the face of uncertainty* (Szita & Lőrincz, 2008) counterpart. Section 2 recalls the necessary background on MDPs and safe policy improvement (SPI).

Section 3 presents our novel optimization formulation: SPI by Baseline Bootstrapping (SPIBB). It consists in bootstrapping the trained policy with the baseline policy in the state-action pair transitions that were not probed sufficiently often in the dataset. We develop two novel computationally efficient SPIBB algorithms, a value-based and a policy-based, both accompanied with theoretical SPI bounds. At the expense of theoretical guarantees, we implement three additional algorithms variants. Then, we develop the related work positioning where we argue that the algorithms found in the literature are impractical for the following reasons: they are intractable in non-small MDPs and/or make unreasonable assumptions on the dataset size and distribution.

Then, Section 4 empirically validates the theoretical results on a gridworld problem, where our algorithms are compared to the state of the art algorithms. The results show that our algorithms significantly outperform the competitors, both on the mean performance and on the worst case scenario, while being as computationally efficient as a standard model-based algorithm.

Finally, Section 5 concludes the paper with prospective ideas of improvement. Appendix includes the proof of all theorems and some additional experimental results.

2. Background

2.1. The MDP framework

Markov Decision Processes (MDPs) (Bellman, 1957) are a widely used framework to address the problem of optimizing a sequential decision making. In our work, we assume that the true environment is modelled as an unknown MDP $M^* = \langle \mathcal{X}, \mathcal{A}, R^*, P^*, \gamma \rangle$, where $\mathcal{X}$ is the state space, $\mathcal{A}$ is the action space, $R^*(x,a) \in [-R_{max}, R_{max}]$ is the true bounded stochastic reward function, $P^*(\cdot|x,a)$ is the true transition probability, and $\gamma \in [0,1[$ is the discount factor. Without loss of generality, we assume that the process deterministically begins in state x_0, the stochastic initialization being modelled by $P^*(\cdot|x_0,a_0)$, and leading the agent to state x_1. The agent then makes a decision about which action a_1 to select. This action leads to a new state that depends on the transition probability and the agent receives a reward $R^*(x_1,a_1)$ reflecting the quality of the decision. This process is then repeated until the end of the episode. We denote by π the policy which corresponds to the decision making mechanism that assigns actions to states. We denote by $\Pi = \{\pi : \mathcal{X} \to \Delta_{\mathcal{A}}\}$ the set of stochastic policies, with $\Delta_{\mathcal{A}}$ the set of probability distributions over the set of actions $\mathcal{A}$.

The state value function $V^\pi_M(x)$ (resp. state-action value function $Q^\pi_M(x,a)$) evaluates the performance of policy $\pi \in \Pi$ starting from state $x \in \mathcal{X}$ (resp. performing action $a \in \mathcal{A}$ in state $x \in \mathcal{X}$) in the MDP $M = \langle \mathcal{X}, \mathcal{A}, R, P, \gamma \rangle$:

$$V^\pi_M(x) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R(x_t,a_t) \middle| \begin{array}{l} x_0 = x \\ a_t \sim \pi(\cdot|x_t) \\ x_{t+1} \sim P(\cdot|x_t,a_t) \end{array}\right]$$

$$Q^\pi_M(x,a) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R(x_t,a_t) \middle| \begin{array}{l} x_0 = x, a_0 = a \\ a_t \sim \pi(\cdot|x_t) \\ x_{t+1} \sim P(\cdot|x_t,a_t) \end{array}\right]$$

The goal of a reinforcement learning algorithm is to discover the unique optimal state value function V^*_M (resp. action-state value function Q^*_M). We define the performance of a policy by its expected value $\rho(\pi,M) = V^\pi_M(x_0)$. Given a policy subset $\Pi' \subseteq \Pi$, a policy π' is said to be Π'-optimal for an MDP M when it maximises its performance: $\rho(\pi',M) = \max_{\pi\in\Pi'} \rho(\pi,M)$. Later, we also make use of the notation $V_{max} \leq \frac{R_{max}}{1-\gamma}$ as a known upper bound of the return absolute value.

2.2. Percentile criterion

We transpose here the *percentile criterion* (Delage & Mannor, 2010) to the safe policy improvement objective:

$$\pi_C = \operatorname*{argmax}_{\pi\in\Pi} \mathbb{E}\left[\rho(\pi,M) \,|\, M \sim \mathbb{P}_{\text{MDP}}(\cdot|\mathcal{D})\right], \qquad (1)$$
$$\text{s.t. } \mathbb{P}\left(\rho(\pi,M) \geq \rho(\pi_b,M) - \zeta \,|\, M \sim \mathbb{P}_{\text{MDP}}(\cdot|\mathcal{D})\right) \geq 1-\delta,$$

where $\mathbb{P}_{\text{MDP}}(\cdot|\mathcal{D})$ is the posterior probability of the MDP parameters, where $1-\delta$ is the high probability meta-parameter, and where ζ is the error meta-parameter. (Petrik et al., 2016) bound from below the constraint by considering $\Xi(\widehat{M}, e)$ as the set of admissible MDP with high probability $1-\delta$, where $\widehat{M} = \langle \mathcal{X}, \mathcal{A}, \widehat{P}, \widehat{R}, \gamma \rangle$ is the MDP parameters estimator, and $e : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is an error function parametrised with the dataset $\mathcal{D}$ and the meta-parameter δ:

$$\begin{aligned} \Xi(\widehat{M}, e) = \{M : &\forall (x,a) \in \mathcal{X} \times \mathcal{A}, \\ &||P(\cdot|x,a) - \widehat{P}(\cdot|x,a)||_1 \leq e(x,a), \\ &||R(\cdot|x,a) - \widehat{R}(\cdot|x,a)||_1 \leq e(x,a)R_{max}\} \end{aligned}$$

Instead of the immeasurable expectation in Equation 1, Robust MDP (Iyengar, 2005; Nilim & El Ghaoui, 2005) classically consider the worst case scenario in Ξ (from now on, the $\Xi(\widehat{M}, e)$ notation is simplified) of the maximization of the performance: $\rho(\pi,M)$. Rather, (Petrik et al., 2016) contemplate the worst case scenario in Ξ of the SPI problem: the maximization the gap between the target and the baseline performances:

$$\pi_S = \operatorname*{argmax}_{\pi\in\Pi} \min_{M\in\Xi} \left(\rho(\pi,M) - \rho(\pi_b,M)\right) \qquad (2)$$

Unfortunately, they prove that this is an NP-hard problem. They propose two algorithms approximating the solution without any formal proof. First, Approximate Robust Baseline Regret Minimizatrion (ARBRM) assumes that there is no error in the transition probabilities of the baseline policy, which is a hazardous assumption. Also, considering its high complexity (polynomial time), it is difficult to empirically assess its percentile criterion safety. Second, the Robust MDP solver uses a Ξ-worst-case safety test to guarantee safety, which is very conservative.

3. SPI with Baseline Bootstrapping

3.1. SPIBB methodology

As evoked in Section 2.2, we endeavour in this section to further reformulate the percentile criterion in order to find an efficient and provably-safe policy within a tractable amount of computer time. Our new criterion consists in optimising the policy with respect to its performance in the MDP estimate $\widehat{M}$, while being guaranteed to be ζ-approximately at least as good as π_b in the admissible MDP set Ξ, with high probability $1-\delta$. More formally, we write it as follows:

$$\max_{\pi\in\Pi} \rho(\pi, \widehat{M}), \text{ s.t. } \forall M \in \Xi, \rho(\pi,M) \geq \rho(\pi_b,M) - \zeta \qquad (3)$$

In order to have this constraint fulfilled with high probability $1-\delta$, for a model-based RL learner, the choice of ζ

Algorithm 1 Construction of the set of bootstrapped pairs

```
Data: Dataset $\mathcal{D}$
Data: Parameters $\epsilon$, $\delta$, and $\gamma$
Data: State and action sets: $\mathcal{X}$ and $\mathcal{A}$
Initiate the $Q^{\pi_b}$-bootstrapped state-action pair set: $\mathfrak{B} = \emptyset$.
$N_\wedge = \frac{2}{\epsilon^2} \log \frac{|\mathcal{X}||\mathcal{A}|2^{|\mathcal{X}|}}{\delta}$
for $(x, a) \in \mathcal{X} \times \mathcal{A}$ do
    Compute $N_\mathcal{D}(x, a)$ the transition count from the state-action $(x, a)$ couple in $\mathcal{D}$.
    if $N_\mathcal{D}(x, a) < N_\wedge$ then
        $\mathfrak{B} = \mathfrak{B} \cup \{(x, a)\}$
    end
end
return $\mathfrak{B}$
```

is determined by Theorem 8 from (Petrik et al., 2016) that we reformulate hereinbelow with the consideration of the uncertainty on the reward model.

Theorem 1 (Near optimality of model-based RL)**.** *Let $\pi^*_{\widehat{M}}$ be an optimal policy in the MDP $\widehat{M}$ constructed from the dataset. Let π^*_M be an optimal policy in the true MDP M. If at each state-action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$, we define:*

$$e(x, a) = \sqrt{\frac{2}{N_\mathcal{D}(x, a)} \log \frac{|\mathcal{X}||\mathcal{A}|2^{|\mathcal{X}|}}{\delta}}, \tag{4}$$

then, with high probability $1 - \delta$,

$$\rho(\pi^*_{\widehat{M}}, M) \geq \rho(\pi^*_M, M) - \frac{2V_{max}}{1-\gamma} ||e||_\infty. \tag{5}$$

Inversely, given a desired ζ, the count should satisfy for every state-action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$:

$$N_\mathcal{D}(x, a) \geq N_\wedge = \frac{8V^2_{max}}{\zeta^2(1-\gamma)^2} \log \frac{|\mathcal{X}||\mathcal{A}|2^{|\mathcal{X}|}}{\delta} \tag{6}$$

In this paper, we extend this previous result by allowing this constraint to be only partially satisfied in a subset of $\mathcal{X} \times \mathcal{A}$. Its complementary subset, the set of uncertain state-action pairs, is called the bootstrapped pairs and is denoted by $\mathfrak{B}$ in the following. $\mathfrak{B}$ is dependent on the state set $\mathcal{X}$, on the action set $\mathcal{A}$, on the dataset $\mathcal{D}$ and on a parameter $N_\wedge$, which itself depends on three parameters: the return precision level ζ, or equivalently the MDP model precision level $\epsilon = ||e||_\infty$, the high probability $1 - \delta$, and the discount factor γ. For ease of notation those dependencies are omitted. The pseudocode for the construction of the set of bootstrapped state-action pairs is presented in Algorithm 1.

We call SPI with Baseline Bootstrapping (SPIBB) the methodology of bootstrapping the uncertain state-action pairs with low variance value estimators/policies obtained from the baseline policy and then to use RL to train a policy. We implement in the next subsections two novel SPIBB algorithms. We show that this approach is safe and prove SPI bounds. We derive three additional SPIBB variants that work better to some extent in our experiments.

3.2. Value-based SPIBB

In this section, we consider bootstrapping the uncertain state-action pairs $(x, a) \in \mathfrak{B}$ with a transition to a terminal state yielding an immediate reward equal to the baseline policy expected return estimate: $\widehat{Q}^{\pi_b}(x, a)$. Considering that the baseline policy is the behavioural policy used for the generation of dataset $\mathcal{D}$, the estimates $\widehat{Q}^{\pi_b}(x, a)$ can be obtained by averaging the returns obtained in the dataset after (x, a) transitions[1]. Indeed, the constraint on $N_\mathcal{D}(x, a)$ for estimation at precision ϵ with probability $1 - \delta$ grows logarithmically with the state set size:

Proposition 1. *If for all state action pairs $(x, a) \in \mathfrak{B}$, $\sqrt{\frac{2}{N_\mathcal{D}(x,a)} \log \frac{2|\mathcal{X}||\mathcal{A}|}{\delta}} \leq \epsilon$, then, with probability at least $1 - \delta$:*

$$\begin{cases} \forall (x, a) \notin \mathfrak{B}, ||P^*(\cdot|x, a) - \widehat{P}(\cdot|x, a)||_1 \leq \epsilon \\ \forall (x, a) \notin \mathfrak{B}, |R^*(x, a) - \widehat{R}(x, a)| \leq \epsilon R_{max} \\ \forall (x, a) \in \mathfrak{B}, |Q^{\pi_b}(x, a) - \widehat{Q}^{\pi_b}(x, a)| \leq \epsilon V_{max} \end{cases} \tag{7}$$

Inversely, given a desired ϵ, the count should satisfy:

$$N_\mathcal{D}(x, a) \geq N_\perp = \frac{2}{\epsilon^2} \log \frac{2|\mathcal{X}||\mathcal{A}|}{\delta} \tag{8}$$

In the rest of this subsection, we assume that this inequality is satisfied for every state-action pair $(x, a) \in \mathfrak{B}$. If so, we can bootstrap the uncertain state-action pairs with the Q-function estimates for the baseline policy by creating a Q^{π_b}-bootstrapped MDP $\tilde{M}$ as described earlier, and formalised in Algorithm 2. Then, we solve the estimated Q^{π_b}-bootstrapped MDP $\widehat{\tilde{M}}$ and let $\pi^\odot_{val}$ denote an optimal policy. Hereinbelow, Theorem 2 provides bounds on its near optimality in $\tilde{M}$, while Theorem 3 offers guarantees on improving the baseline policy in M^*.

Theorem 2 (Near optimality of Q^{π_b}-SPIBB-1)**.** *Let $\pi^\odot_{val}$ be an optimal policy of the reward maximization problem of an estimated Q^{π_b}-bootstrapped MDP $\widehat{\tilde{M}}$. Then, under the construction properties of $\tilde{M}$ and under the assumption of Proposition 1, the performance of $\pi^\odot_{val}$ in $\tilde{M}$ is near-*

[1] If the baseline policy is not the behavioural policy, the estimates $\widehat{Q}^{\pi_b}(x, a)$ can still be obtained through importance sampling, but it suffers from a large variance that may compromise the use of a small ϵ.

```
Algorithm 2 $Q^{\pi_b}$-SPIBB-1 algorithm
Data: Dataset $\mathcal{D}$
Data: Set of bootstrapped state-action pairs $\mathfrak{B}$
Data: State and action sets: $\mathcal{X}$ and $\mathcal{A}$
Compute the estimates $\widehat{P}$, $\widehat{R}$, and $\widehat{Q}^{\pi_b}$ from $\mathcal{D}$.
Construct the estimated $\widehat{Q}^{\pi_b}$-bootstrapped MDP: $\widehat{\tilde{M}} = \langle \mathcal{X}, \mathcal{A}, \widehat{\tilde{P}}, \widehat{\tilde{R}}, \gamma \rangle$ such that:
    • $\widehat{\tilde{P}}(x'|x,a) = \begin{cases} \widehat{P}(x'|x,a) & \text{if } (x,a) \notin \mathfrak{B} \\ \widehat{\tilde{P}}(x_f|x,a) = 1 & \text{otherwise} \end{cases}$
    • $\widehat{\tilde{R}}(x,a) = \begin{cases} \widehat{R}(x,a) & \text{if } (x,a) \notin \mathfrak{B} \\ \widehat{Q}^{\pi_b}(x,a) & \text{otherwise} \end{cases}$
return $\pi^{\odot}_{val} = \underset{\pi \in \Pi}{\text{argmax}}\, \rho(\pi, \widehat{\tilde{M}})$
```

optimal:

$$\rho(\pi^{\odot}_{val}, \tilde{M}) \geq \max_{\pi \in \Pi} \rho(\pi, \tilde{M}) - \frac{2\epsilon V_{max}}{1-\gamma} \tag{9}$$

Theorem 3 (Safe policy improvement of Q^{π_b}-SPIBB-1)**.** *Let $\pi^{\odot}_{val}$ be an optimal policy of the reward maximization problem of an estimated Q^{π_b}-bootstrapped MDP $\widehat{\tilde{M}}$. Then, under the construction properties of $\tilde{M}$ and under the assumption of Proposition 1, $\pi^{\odot}_{val}$ applied in $\tilde{M}$ is an approximate safe policy improvement over the baseline policy π_b with high probability $1-\delta$:*

$$\rho(\pi^{\odot}_{val}, \tilde{M}) \geq \rho(\pi_b, M^*) - \frac{2\epsilon V_{max}}{1-\gamma} \tag{10}$$

$\pi^{\odot}_{val}$ is trained on the estimated Q^{π_b}-bootstrapped MDP, which can be performed with any RL algorithm with the same computational efficiency.

During utilization of $\pi^{\odot}_{val}$, casting the environment into its Q^{π_b}-bootstrapped version means that once a Q^{π_b}-bootstrapped state-action pair $(x,a) \in \mathfrak{B}$ has been performed, the reached state in $\tilde{M}$ is terminal and the baseline policy π_b should take control of the trajectory until its end. This has two practical shortcomings: 1/ it means that π_b must be known, and 2/ it does not take advantage of the fact that $\pi^{\odot}_{val}$ is defined over all state-action pairs, and expected to be more efficient that π_b.

As a consequence, despite the lack of theoretical guarantees, the experimental section also assesses the empirical safety of continuing to control the trajectory with $\pi^{\odot}_{val}$ after choosing a bootstrapping action. These two variants of value-based SPIBB are respectively referred as Q^{π_b}-SPIBB-1 and Q^{π_b}-SPIBB-∞.

3.3. Policy-based SPIBB

In the previous section, we propose to bootstrap the uncertain state-action pairs with a Monte Carlo evaluation of the baseline policy. In this section, we adopt a policy bootstrapping. More precisely, when a state-action pair (x,a) is rarely seen in the dataset, *i.e.* $(x,a) \in \mathfrak{B}$, the batch algorithm is unable to assess its performance and instead it relies on the baseline policy by copying the probability to take this action in this particular situation: $\pi(a|x) = \pi_b(a|x)$ if $(x,a) \in \mathfrak{B}$. Algorithm 3 provides the pseudo-code of the baseline policy bootstrapping. It consists in constructing the set of allowed policies Π_b and then to search the Π_b-optimal policy $\pi^{\odot}_{pol}$ in the MDP model $\widehat{M}$ estimated from dataset $\mathcal{D}$. In practice, the optimisation process may be performed by policy iteration (Howard, 1960; Puterman & Brumelle, 1979): the current policy $\pi^{(i)}$ is evaluated with $Q^{(i)}$, and then the next iteration policy $\pi^{(i+1)}$ is made greedy with respect to $Q^{(i)}$ under the constraint of belonging to Π_b (see Algorithm 4 in Appendix).

```
Algorithm 3 $\Pi_b$-SPIBB algorithm
Data: Dataset $\mathcal{D}$
Data: Baseline policy $\pi_b$
Data: Set of bootstrapped state-action pairs $\mathfrak{B}$
Data: State and action sets: $\mathcal{X}$ and $\mathcal{A}$
Compute estimated MDP: $\widehat{M} = \langle \mathcal{X}, \mathcal{A}, \widehat{P}, \widehat{R}, \gamma \rangle$ where $\widehat{P}$
  and $\widehat{R}$ are the model estimates obtained from $\mathcal{D}$.
Define $\Pi_b = \{\pi \in \Pi \,|\, \pi(a|x) = \pi_b(a|x) \text{ if } (x,a) \in \mathfrak{B}\}$
return $\pi^{\odot}_{pol} = \underset{\pi \in \Pi_b}{\text{argmax}}\, \rho(\pi, \widehat{M})$
```

Similarly to Theorems 2 and 3 for Q^{π_b}-SPIBB-1, the near Π_b-optimality and the SPI of the baseline policy can be proven for Π_b-SPIBB:

Theorem 4 (Near Π_b-optimality of Π_b-SPIBB)**.** *Let Π_b be the set of policies under the constraint of following π_b when $(x,a) \in \mathfrak{B}$. Let $\pi^{\odot}_{pol}$ be a Π_b-optimal policy of the reward maximization problem of an estimated MDP $\widehat{M}$. Then, the performance of $\pi^{\odot}_{pol}$ is near Π_b-optimal in the true MDP M^*:*

$$\rho(\pi^{\odot}_{pol}, M^*) \geq \max_{\pi \in \Pi_b} \rho(\pi, M^*) - \frac{2\epsilon V_{max}}{1-\gamma}. \tag{11}$$

Theorem 5 (Safe policy improvement of Π_b-SPIBB)**.** *Let Π_b be the set of policies under the constraint of following π_b when $(x,a) \in \mathfrak{B}$. Let $\pi^{\odot}_{pol}$ be a Π_b-optimal policy of the reward maximization problem of an estimated MDP $\widehat{M}$. Then, $\pi^{\odot}_{pol}$ is an approximate safe policy improvement over the baseline policy π_b with high probability $1-\delta$:*

$$\rho(\pi^{\odot}_{pol}, M^*) \geq \rho(\pi_b, M^*) - \frac{2\epsilon V_{max}}{1-\gamma}. \tag{12}$$

Q-value	Baseline policy	Boostrapped state	Π_b-SPIBB	Π_0-SPIBB	$\Pi_{\leq b}$-SPIBB
$Q^{(i)}(x,a_1)=1$	$\pi_b(x,a_1)=0.1$	$(x,a_1)\in\mathfrak{B}$	$\pi^{(i)}(x,a_1)=0.1$	$\pi^{(i)}(x,a_1)=0$	$\pi^{(i)}(x,a_1)=0$
$Q^{(i)}(x,a_2)=2$	$\pi_b(x,a_2)=0.4$	$(x,a_2)\notin\mathfrak{B}$	$\pi^{(i)}(x,a_2)=0$	$\pi^{(i)}(x,a_2)=0$	$\pi^{(i)}(x,a_2)=0$
$Q^{(i)}(x,a_3)=3$	$\pi_b(x,a_3)=0.3$	$(x,a_3)\notin\mathfrak{B}$	$\pi^{(i)}(x,a_3)=0.7$	$\pi^{(i)}(x,a_3)=1$	$\pi^{(i)}(x,a_3)=0.8$
$Q^{(i)}(x,a_4)=4$	$\pi_b(x,a_4)=0.2$	$(x,a_4)\in\mathfrak{B}$	$\pi^{(i)}(x,a_4)=0.2$	$\pi^{(i)}(x,a_4)=0$	$\pi^{(i)}(x,a_4)=0.2$

Table 1: Policy improvement step at iteration (i) for the three policy-based SPIBB algorithms.

Algorithm 3, referred as Π_b-SPIBB, has the tendency to reproduce the rare actions from the baseline policy. Even though this is what allows to guarantee a performance almost as good as the baseline policy's one, it may prove to be toxic when the baseline policy is already near optimal for two reasons: 1/ the low visited state-action pairs are generally the actions for which the behavioural policy probability is lower, meaning that the actions are likely to be bad, 2/ the exploratory strategies that are embedded in the baseline policy fall into this category, and reproducing the baseline policy in this case is reproducing these strategies.

Another way to look at the problem is therefore to consider that those rare actions must be avoided, because they are risky, and therefore to force the policy to assign a probability of 0 to perform this action. Algorithm 3 remains unchanged except that the policy search space Π_b has to be replaced with Π_0 (see Algorithm 5 in Appendix) defined as follows:

$$\Pi_0 = \{\pi \in \Pi \,|\, \pi(a|x) = 0 \text{ if } (x,a) \in \mathfrak{B}\} \tag{13}$$

The empirical analysis of Section 4 shows that this variant, referred as Π_0-SPIBB, often proves to be unsafe. We believe that a better policy-improvement SPIBB lays in-between: the space of policies to search in should be constrained not to give more weight than π_b to actions that were not tried out enough to significantly assess its performance, but still leave the possibility to completely cut off bad performing actions even though this evaluation is uncertain. The resulting algorithm is referred as $\Pi_{\leq b}$-SPIBB. Once again, Algorithm 3 remains unchanged except for the policy search space Π_b that has to be replaced with $\Pi_{\leq b}$ defined as follows:

$$\Pi_{\leq b} = \{\pi \in \Pi \,|\, \pi(a|x) \leq \pi_b(a|x) \text{ if } (x,a) \in \mathfrak{B}\} \tag{14}$$

Algorithm 4 in Appendix describes the greedy projection of $Q^{(i)}$ on $\Pi_{\leq b}$. Despite the lack of theoretical guarantees, in our experiments, $\Pi_{\leq b}$-SPIBB proves to be safe while outperforming Π_b-SPIBB. However, in a growing batch batch setting (Lange et al., 2012), it might be valuable to keep exploring the way Π_b-SPIBB does.

Table 1 shows the difference of policy projection during the policy improvement step of the policy iteration process. It shows how the baseline policy probability mass is redistributed among the different actions according to the three policy-based SPIBB algorithms. We observe that for Π_b-SPIBB, the boostrapped state-action pairs probabilities are untouched. At the opposite, Π_0-SPIBB removes all mass from the boostrapped state-action pairs. And finally $\Pi_{\leq b}$-SPIBB lies in-between.

3.4. Discussion

Table 2 summarizes the algorithms strengths and weaknesses. High-Confidence PI refers to the family of algorithms introduced in (Thomas et al., 2015a), which rely on the ability to produce high-confidence policy evaluation (Mandel et al., 2014; Thomas et al., 2015b) of the trained policy, which is known to be high variance (Paduraru, 2013). As a consequence, they formally rely on a policy improvement safety test that is unlikely to be positive without an immense dataset.

(Petrik et al., 2016) propose a large variety of algorithms. The basic (model-based) RL relates to searching the optimal policy in the MDP estimate $\widehat{M}$. It is proved to converge to the optimal policy, but this proof relies on the unpractical assumption of having sampled every state-action pair a huge number of times (see Theorem 1 in Section 4). ARBRM assumes the transition model known around the baseline policy, which is a strong assumption that cannot be made in most practical problems. ARBRM and Robust MDP to a lesser extent suffer from high complexity even with a finite state space MDP: NP-hard reduced through approximation to polynomial time; and they also lack safety guarantees with respect to their approximation to make it tractable. Finally, Reward-Adjusted MDP's algorithm has no proven safety and relies as a consequence on a safety test, similarly to High-Confidence PI.

(Berkenkamp et al., 2017) assume the existence of a local, stable policy, and their Safe Lyapunov RL algorithm allows to safely explore outside from the safe region without ever leaving it. It exclusively addresses the stabilization tasks in the online scenario. (Roy et al., 2017) develop robust versions of the main model-free algorithms, offering algorithms that robustly converge online to the optimal policy. It does not help in the general batch setting, where the trust regions are infinite if a state-action pair has never been ob-

algorithm name	safe PI	PI test	known π_b?	$\pi_{beh} = \pi_b$?	$N_\mathcal{D}(x,a) \geq N_\bullet$?
Basic model-based RL	yes	no	no	no	$N_\wedge = \frac{2}{\epsilon^2} \log \frac{\lvert\mathcal{X}\rvert\lvert\mathcal{A}\rvert 2^{\lvert\mathcal{X}\rvert}}{\delta}$
ARBRM	yes	yes	yes	yes	no
Robust MDP ξ	yes	yes	yes	no	no
Reward-Adjusted MDP	yes	yes	no	no	no
High-Confidence PI	no	yes	no	no	no
Q^{π_b}-SPIBB-1	Th. 3	no	yes	yes	$N_\perp = \frac{2}{\epsilon^2} \log \frac{2\lvert\mathcal{X}\rvert\lvert\mathcal{A}\rvert}{\delta}$
Q^{π_b}-SPIBB-∞	no	n/a	no	yes	$N_\perp = \frac{2}{\epsilon^2} \log \frac{2\lvert\mathcal{X}\rvert\lvert\mathcal{A}\rvert}{\delta}$
Π_b-SPIBB	Th. 5	no	yes	no	no
Π_0-SPIBB	no	n/a	yes	no	no
$\Pi_{\leq b}$-SPIBB	no	n/a	yes	no	no

Table 2: Brief summary of safe improvement algorithms. The columns are from left to right: name of the algorithm, existence of safe improvement guarantees, does it rely on a policy improvement test?, does the baseline policy need to be known?, is the baseline policy required to be the behavioural policy?, is there a constraint on the action-state pair counts? The safety of basic model-based RL is proved in (Petrik et al., 2016). ARBRM, Robust MDP, and Reward-Adjusted MDP are respectively Algorithms 1, 2, and 3 in (Petrik et al., 2016). High-Confidence PI is the general approach to policy improvement guaranteed by off-policy evaluation confidence intervals defended in (Thomas, 2015).

served. (Papini et al., 2017) propose to adapt the batch size to ensure that the gradients safely improve the policy. It has two main shortcomings: it requires the use of policy gradient updates, which is less sample-efficient than model-based methods, and it commands the batch size, which is usually not a feature over which the system has control in an offline setting like ours. These three very recent algorithms use different assumptions and cannot be directly compared to our algorithms.

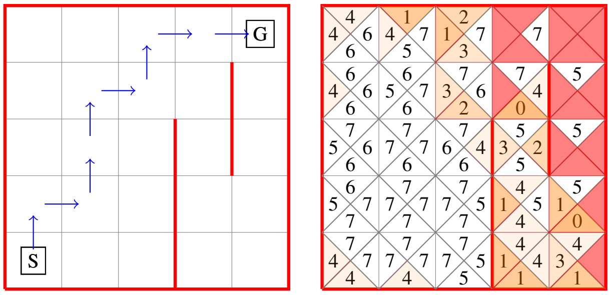
Our algorithms take inspiration from the ARBRM idea of finding a policy that is guaranteed to be an improvement for any realization of the uncertain parameters. Still, like ARBRM, they do so by taking into account the estimation of the error, as a function of the state-action pair counts. But instead of searching for the analytic optimum, it goes straightforwardly to a solution improving the baseline policy where it can guarantee the improvement, and bootstrapping on the baseline policy where the uncertainty is too high. One can see it as a *knows-what-it-knows* algorithm (Li et al., 2008), asking for help from the baseline policy when it *does not know whether it knows*. As a consequence, our proofs do not require the policy improvement safety test. Q^{π_b}-SPIBB-1 is proved to be safe under conditions of application that are widely relaxed, compared with the basic model-based RL Theorem 1. Π_b-SPIBB algorithm does not rely on any condition of application else than knowing the baseline policy. Additionally, contrary to the other robust/safe batch RL algorithms in the literature, all SPIBB algorithms maintain a computational cost equal to the basic RL algorithms.

4. Experimental evaluation

4.1. Gridworld setting

Our case study is a straightforward discrete, stochastic 5×5 gridworld (see Figure 1a). We use four actions: up, down, left and right. The transition function is stochastic and the actions move the agent in the specified direction with 75% chances, in the opposite direction with 5% chances and with 10% to each side. The initial and final states are respectively the bottom left and top right corners. The reward is -10 when hitting a wall (in which case the agent does not move) and $+100$ if the final state is reached. Each run consists in generating a dataset, training a policy from it, and evaluating the trained policy.

The gridworld domain is justified by the fact that basic



(a) Stochastic gridworld domain and its optimal trajectory

(b) $\log_{10}$ state-action pair counts after 1.2×10^7 trajectories

Figure 1: Gridworld domain and dataset distribution.

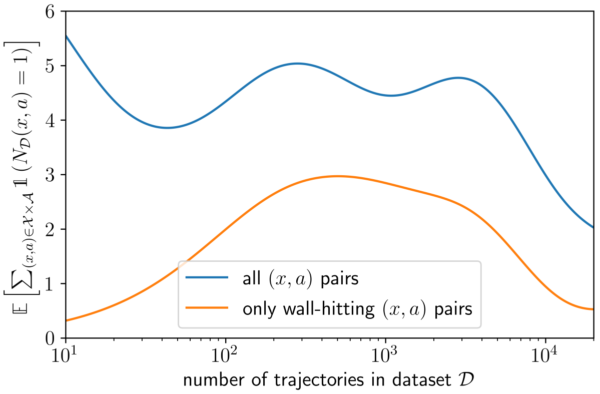


Figure 2: Expectation of the number of (x,a) pairs counting a unique transition as a function of the dataset size.

model-based RL already fails to be safe in this simple environment, and by the empirical worst-case evaluation that requires to run 1,000 runs for 8 algorithms (the 5 SPIBB algorithms, the basic RL, Robust MDP, and the Reward-Adjusted MDP), 11 dataset sizes, and 8 $N_\wedge$ values. For Basic RL, several Q-functions initializations were investigated: the optimistic (V_{max}), the null (0), and the pessimistic ($-V_{max}$). The two first yielded awful performances. All the presented results are obtained with the pessimistic initialization.

Two baseline policies were used to generate the dataset and to bootstrap on. The literature benchmark is performed on the first one, a strong softmax exploration around the optimal Q-function. The SPIBB benchmark is performed on the second one, which differs in that it favours walking along the walls, although it should avoid it to prevent bad stochastic transitions. This baseline was constructed in order to demonstrate the unsafety of algorithm Π_0-SPIBB.

The results are presented in two forms: the mean performance on all the runs; and the worst-case performance of the 10% (decile) or 1% (centile) worst runs. For the SPIBB algorithms, values of $N_\wedge$ from 5 to 1000 are tested.

4.2. Basic model-based RL failure

All the state-of-the-art algorithms for batch RL assume that every state-action has been experienced a certain amount of times (Delage & Mannor, 2010; Petrik et al., 2016). In this subsection, we aim to empirically demonstrate that this assumption is generally transgressed even in our simple grid-world domain. To do so, we collect 12 millions trajectories with the first baseline. The map of the state-action count $\log_{10}$ logarithm (see Figure 1b) shows how unbalanced the $N_\mathcal{D}(x,a)$ counts are: some transitions are experienced in each trajectory, some only once every few million trajectories, and some are even never seen once.

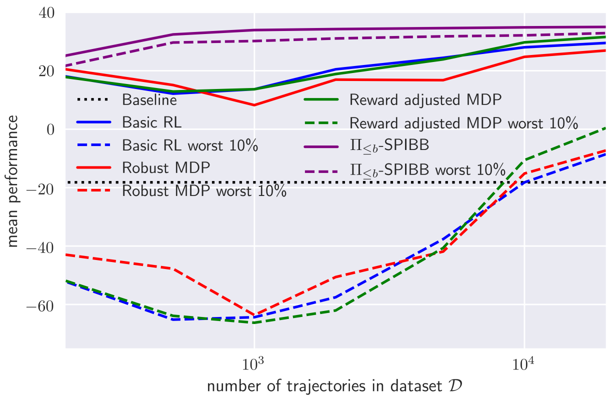
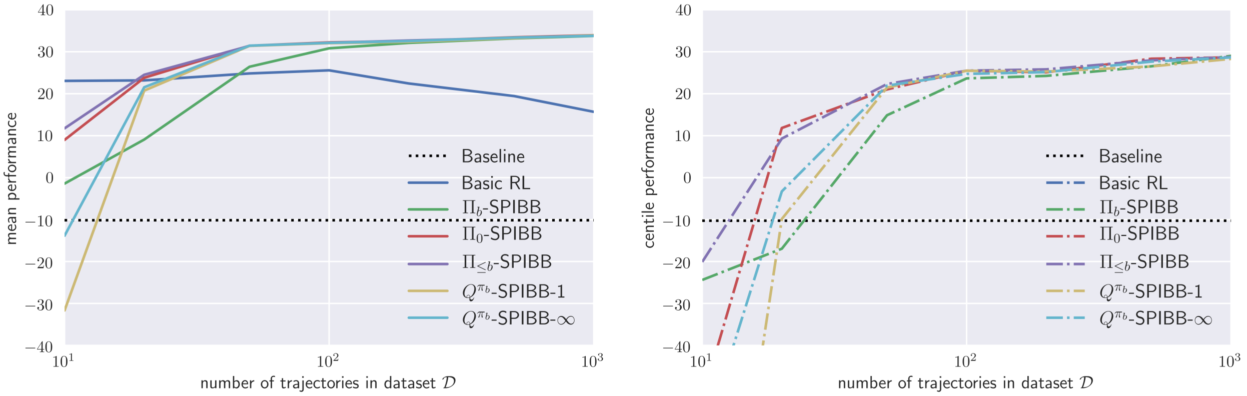


Figure 3: Literature benchmark: Basic RL, Robust MDP, and Reward Adjusted MDP are compared to our $\Pi_{\leq b}$-SPIBB with $N_\wedge = 100$ on mean and safe performances.

Moreover, the actions that are rarely chosen are likely to be the dangerous ones, and for those ones, a bad model might lead to a catastrophic policy. Figure 2 displays the expected number of transitions that are seen exactly once in a dataset as a function of its size. This is a curve that decreases slowly as more trajectories are collected. But, if we look more specifically at dangerous transitions, *i.e.* the ones that direct the agent to a wall, we observe a peak around 1,000 trajectories. In the next subsection, we see that it strongly affects the basic RL safety: surprisingly, the models trained with 10 trajectories yield better returns than the ones trained with 1,000 trajectories on average. We conjecture that this issue is faced in most practical applications too. For instance, in dialogue, all the collected human dialogue transitions are relevant to what is being discussed.

4.3. Results

Figure 3 shows the literature benchmark results against our best algorithm $\Pi_{\leq b}$-SPIBB. The basic RL algorithm performs reasonably well on average, but fails to be safe, and sometimes outputs a policy that is disastrous. We can notice that the performance reaches a valley for datasets around 1,000 trajectories. We interpret it as the consequence of the rare pair count effect developed in the previous subsection. Neither Robust MDP, nor Reward Adjusted MDP, seem to improve the safety when the safety test is omitted. We did so in our curves to make a relevant comparison: this test appears to be always negative because of its wide confidence interval. It is also worth mentioning that the Reward Adjusted MDP algorithm tends to become suicidal in environments where it can get killed (not the case in our domain). Indeed, the intrinsic penalty adjustment may be overwhelming the environment reward and the optimal strategy may be to stop the trajectory as fast as possible. Our algorithm $\Pi_{\leq b}$-SPIBB with $N_\wedge = 100$ is safe. Its worst decile performance is even significantly higher than the other algorithms mean performance. The



(a) Mean performance

(b) Worst centile performance

Figure 4: SPIBB benchmark ($N_\wedge = 5$): mean and worst centile performance

SPIBB algorithms empirical results are lengthily discussed hereinbelow.

Our SPIBB algorithms are so efficient and safe that we shift the dataset size range to the [10,1000] window. The safety is assessed by a worst-centile measure: mean of the performance of the 1% worst runs. The basic RL worst centile is too low to appear. A wide range of values for $N_\wedge$ are evaluated: from 5 to 1000. The main lessons are that the safety of improvement over the baseline is not much impacted by the choice of $N_\wedge$, but that a higher $N_\wedge$ implies the SPIBB algorithms to be more conservative and to bootstrap more often on the baseline. For complete results, we refer the interested reader to the Appendix. Even though the theory would advise to use higher values, we report here our best empirical results: with $N_\wedge = 5$.

Value-based SPIBB algorithms Q^{π_b}-SPIBB-1 and Q^{π_b}-SPIBB-∞ fail at being safe with small datasets. The reason is that these algorithms rely on the assumption that even bootstrapped state-action pairs must have been experienced a small amount of times. We also notice that, despite the lack of guarantees, Q^{π_b}-SPIBB-∞ improves the safety as compared to Q^{π_b}-SPIBB-1.

Π_b-SPIBB and $\Pi_{\leq b}$-SPIBB get a worst case scenario only 10 points below the baseline, which is partially explained by the variance in the evaluation, and is not likely to be a consequence of a bad policy. Π_0-SPIBB lacks safety with very small datasets, because it tends to completely abandon actions that are not sampled enough in the dataset, regardless of their performance. Results with higher $N_\wedge$ values show that there is a dataset size for which Π_0-SPIBB tends to cut the optimal actions (of not walking along the wall), which causes a strong performance drop, both in worst case scenario and in mean performance (see the Appendix). Π_b-SPIBB is more conservative and fails to improve as fast as the two other policy-based SPIBB algorithms, but it does it safely. $\Pi_{\leq b}$-SPIBB is the best of both worlds: safe although still capable of cutting bad actions even with only a small number of samples. However, for growing batch settings, it might be better to keep on trying out the actions that were not sufficiently explored yet, and Π_b-SPIBB might be the best algorithm in this setting.

5. Conclusion and future work

In this paper, we tackle the problem of Batch Reinforcement Learning and its safety. We reformulate the percentile criterion without compromising its safety at the expense of the optimality of the safe solution. The gain is that it allows to implement two algorithms Q^{π_b}-SPIBB-1 and Π_b-SPIBB that run as fast as a basic model-based RL algorithm, while generating a provably safe policy improvement over a known baseline π_b. Three other SPIBB algorithms are derived without any safety guarantees: Q^{π_b}-SPIBB-∞, Π_0-SPIBB, and $\Pi_{\leq b}$-SPIBB.

The empirical analysis shows that, even on a very simple domain, the basic RL algorithm fails to be safe, and the state-of-the-art safe batch RL algorithms does no better when the policy improvement safety test is omitted. This safety test has also proven to be (almost) always negative in our tests consequently preventing any improvement over the baseline. The SPIBB algorithms show significantly better results: their worst-centile performance even surpassing the basic RL mean performance in most settings.

Future work includes developing model-free versions of our algorithms in order to ease their use in continuous state MDP and real-world applications, designing a Bayesian policy projection to take into account the uncertainty of local policy evaluation, and demonstrating that our algorithms may be used in conjunction with imitation learning to compute the baseline policy estimate.

Bibliography

Amodei, Dario, Olah, Chris, Steinhardt, Jacob, Christiano, Paul, Schulman, John, and Mané, Dan. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Bellman, Richard. A markovian decision process. *Journal of Mathematics and Mechanics*, 1957.

Berkenkamp, Felix, Turchetta, Matteo, Schoellig, Angela, and Krause, Andreas. Safe model-based reinforcement learning with stability guarantees. In *Proceedings of the 30th Advances in Neural Information Processing Systems (NIPS)*. 2017.

Delage, Erick and Mannor, Shie. Percentile optimization for markov decision processes with parameter uncertainty. *Operations research*, 2010.

Garcıa, Javier and Fernández, Fernando. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 2015.

Geibel, Peter and Wysotzki, Fritz. Risk-sensitive reinforcement learning applied to control under constraints. 2005.

Ghavamzadeh, Mohammad, Mannor, Shie, Pineau, Joelle, and Tamar, Aviv. Bayesian reinforcement learning: A survey. *CoRR*, abs/1609.04436, 2016.

Howard, Ronald A. Dynamic programming and markov processes. 1960.

Iyengar, Garud N. Robust dynamic programming. *Mathematics of Operations Research*, 2005.

Lange, Sascha, Gabel, Thomas, and Riedmiller, Martin. Batch reinforcement learning. In *Reinforcement learning*. 2012.

Li, Lihong, Littman, Michael L, and Walsh, Thomas J. Knows what it knows: a framework for self-aware learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

Lipton, Zachary C, Kumar, Abhishek, Gao, Jianfeng, Li, Lihong, and Deng, Li. Combating deep reinforcement learning's sisyphean curse with reinforcement learning. *arXiv preprint arXiv:1611.01211*, 2016.

Mandel, Travis, Liu, Yun-En, Levine, Sergey, Brunskill, Emma, and Popovic, Zoran. Offline policy evaluation across representations with applications to educational games. In *Proceedings of the 13th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2014.

Nilim, Arnab and El Ghaoui, Laurent. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 2005.

Paduraru, Cosmin. *Off-policy Evaluation in Markov Decision Processes*. PhD thesis, PhD thesis, McGill University, 2013.

Papini, Matteo, Pirotta, Matteo, and Restelli, Marcello. Adaptive batch size for safe policy gradients. In *Proceedings of the 30th Advances in Neural Information Processing Systems (NIPS)*. 2017.

Petrik, Marek, Ghavamzadeh, Mohammad, and Chow, Yinlam. Safe policy improvement by minimizing robust baseline regret. In *Proceedings of the 29th Advances in Neural Information Processing Systems*, 2016.

Puterman, Martin L and Brumelle, Shelby L. On the convergence of policy iteration in stationary dynamic programming. *Mathematics of Operations Research*, 1979.

Roy, Aurko, Xu, Huan, and Pokutta, Sebastian. Reinforcement learning under model mismatch. In *Proceedings of the 30th Advances in Neural Information Processing Systems (NIPS)*. 2017.

Schulman, John, Levine, Sergey, Abbeel, Pieter, Jordan, Michael, and Moritz, Philipp. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sutton, Richard S. and Barto, Andrew G. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

Szita, István and Lőrincz, András. The many faces of optimism: a unifying approach. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

Thomas, Philip, Theocharous, Georgios, and Ghavamzadeh, Mohammad. High confidence policy improvement. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015a.

Thomas, Philip S. *Safe reinforcement learning*. PhD thesis, Stanford university, 2015.

Thomas, Philip S, Theocharous, Georgios, and Ghavamzadeh, Mohammad. High-confidence off-policy evaluation. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015b.

A. Proofs for Q^{π_b}-SPIBB-1 (Section 3.2)

A.1. ϵ error with high probability $1-\delta$

Proposition 1. *If for all state action pairs $(x,a) \in \mathfrak{B}$, $\sqrt{\frac{2}{N_\mathcal{D}(x,a)} \log \frac{2|\mathcal{X}||\mathcal{A}|}{\delta}} \leq \epsilon$, then, with probability at least $1-\delta$:*

$$\begin{cases} \forall (x,a) \notin \mathfrak{B}, \|P^*(\cdot|x,a) - \widehat{P}(\cdot|x,a)\|_1 \leq \epsilon \\ \forall (x,a) \notin \mathfrak{B}, |R^*(x,a) - \widehat{R}(x,a)| \leq \epsilon R_{max} \\ \forall (x,a) \in \mathfrak{B}, |Q^{\pi_b}(x,a) - \widehat{Q}^{\pi_b}(x,a)| \leq \epsilon V_{max} \end{cases} \tag{15}$$

Proof. From construction of $\mathfrak{B}$, and from Proposition 9 of (Petrik et al., 2016), the first condition is satisfied for every state-action pair $(x,a) \notin \mathfrak{B}$ individually with probability $\frac{\delta}{|\mathcal{X}||\mathcal{A}|}$.

The second and the third inequalities are obtained similarly. The proof is further only detailed for the third inequality hereinafter: given $(x,a) \in \mathfrak{B}$, and from the two-sided Hoeffding's inequality:

$$\mathbb{P}\big(|Q^{\pi_b}(x,a) - \widehat{Q}^{\pi_b}(x,a)| > \epsilon V_{max}\big) = \mathbb{P}\left(\frac{|Q^{\pi_b}(x,a) - \widehat{Q}^{\pi_b}(x,a)|}{2V_{max}} > \sqrt{\frac{1}{2N_\mathcal{D}(x,a)} \log \frac{2|\mathcal{X}||\mathcal{A}|}{\delta}}\right) \tag{16}$$

$$\leq 2\exp\left(-2N_\mathcal{D}(x,a)\frac{1}{2N_\mathcal{D}(x,a)} \log \frac{2|\mathcal{X}||\mathcal{A}|}{\delta}\right) \tag{17}$$

$$\leq \frac{\delta}{|\mathcal{X}||\mathcal{A}|} \tag{18}$$

Adding up all $|\mathcal{X}||\mathcal{A}|$ state-action pairs probabilities lower than $\frac{\delta}{|\mathcal{X}||\mathcal{A}|}$ gives a result lower than δ, which proves the proposition. □

A.2. Value function error bounds

Lemma 1 (Value function error bounds)**.** *Consider two transition probability matrices P_1 and P_2, two reward functions R_1 and R_2, and two bootstrapping Q-function Q_1 and Q_2, used to bootstrap two MDPs M_1 and M_2. Consider a policy $\pi \in \Pi$. Let V_1 and V_2 be the state value function of the policy π given (P_1, R_1, Q_1) and (P_2, R_2, Q_2), respectively.*

$$\text{If} \quad \begin{cases} \forall (x,a) \notin \mathfrak{B}, \|P_1(\cdot|x,a) - P_2(\cdot|x,a)\|_1 \leq \epsilon \\ \forall (x,a) \notin \mathfrak{B}, |R_1(x,a) - R_2(x,a)| \leq \epsilon R_{max} \\ \forall (x,a) \in \mathfrak{B}, |Q_1(x,a) - Q_2(x,a)| \leq \epsilon V_{max} \end{cases} \quad \text{then} \quad |V_1 - V_2| \leq (\mathbb{I} - \gamma\dot{\pi}P_1)^{-1}\epsilon V_{max}, \tag{19}$$

where V_{max} is the known maximum of the value function.

Proof. The policy π can be decomposed as the aggregation of two partial policies: $\pi = \dot{\pi} \otimes \tilde{\pi}$, where $\dot{\pi}$ are the non-boostrapped actions probabilities, and $\tilde{\pi}$ are the bootstrapped actions probabilities

Then, the difference between the two value functions can be written:

$$V_1 - V_2 = \pi R_1 + \gamma\pi P_1 V_1 - \pi R_2 - \gamma\pi P_2 V_2 \tag{20}$$

$$= \pi R_1 + \gamma\pi P_1 V_1 - \pi R_2 - \gamma\pi P_2 V_2 + \gamma\pi P_1 V_2 - \gamma\pi P_1 V_2 \tag{21}$$

$$= \pi(R_1 - R_2) - \gamma\pi P_1(V_1 - V_2) + \gamma\pi(P_1 - P_2)V_2 \tag{22}$$

$$= \pi(R_1 - R_2) - \gamma\dot{\pi} P_1(V_1 - V_2) + \gamma\dot{\pi}(P_1 - P_2)V_2 \tag{23}$$

$$= (\mathbb{I} - \gamma\dot{\pi}P_1)^{-1}\left[\pi(R_1 - R_2) + \gamma\dot{\pi}(P_1 - P_2)V_2\right]. \tag{24}$$

Line 23 is explained by the fact that the bootstrapping action lead to a terminal state: therefore $V_1 = V_2 = 0$, and Line 24 is passing the second term to the left-hand side of the equation, factorised over $V_1 - V_2$ and divided by its factor. Now using the Holder's inequality, for any state-action couple $(x,a) \notin \mathfrak{B}$, we have:

$$|(P_1(\cdot|x,a) - P_2(\cdot|x,a))^\top V_2| \leq \|P_1(\cdot|x,a) - P_2(\cdot|x,a)\|_1 \|V_2\|_\infty \leq \epsilon V_{max}. \tag{25}$$

Also, considering the reward term, we get:

$$\begin{aligned} \pi|R_1(x,a) - R_2(x,a)| &= \dot{\pi}|R_1(x,a) - R_2(x,a)| + \tilde{\pi}|R_1(x,a) - R_2(x,a)| && (26) \\ &\leq \dot{\pi}\epsilon R_{max} + \tilde{\pi}\epsilon V_{max}. && (27) \end{aligned}$$

Inserting 25 and 27 into Equation 24 gives:

$$\begin{aligned} |V_1 - V_2| &\leq (\mathbb{I} - \gamma\dot{\pi}P_1)^{-1}\left[\tilde{\pi}\epsilon R_{max} + \tilde{\pi}\epsilon V_{max} + \gamma\dot{\pi}\epsilon V_{max}\right] && (28) \\ &\leq (\mathbb{I} - \gamma\dot{\pi}P_1)^{-1}\left[\tilde{\pi}\epsilon V_{max} + \dot{\pi}\epsilon(R_{max} + \gamma V_{max})\right] && (29) \\ &\leq (\mathbb{I} - \gamma\dot{\pi}P_1)^{-1}\left[\tilde{\pi}\epsilon V_{max} + \dot{\pi}\epsilon V_{max}\right] && (30) \\ &\leq (\mathbb{I} - \gamma\dot{\pi}P_1)^{-1}\epsilon V_{max} && (31) \end{aligned}$$

□

A.3. Near optimality

Theorem 2 (Near optimality of Q^{π_b}-SPIBB-1)**.** *Let $\pi_{val}^{\odot}$ be an optimal policy of the reward maximization problem of an estimated Q^{π_b}-bootstrapped MDP $\widehat{\tilde{M}}$. Then, under the construction properties of $\tilde{M}$ and under the assumption of Proposition 1, the performance of $\pi_{val}^{\odot}$ in $\tilde{M}$ is near-optimal:*

$$\rho(\pi_{val}^{\odot}, \tilde{M}) \geq \max_{\pi\in\Pi} \rho(\pi, \tilde{M}) - \frac{2\epsilon V_{max}}{1-\gamma} \tag{32}$$

Proof. From Lemma 1, with $\pi = \pi_{val}^{\odot}$, $P_1 = \tilde{P}$, $P_2 = \widehat{\tilde{P}}$, $R_1 = \tilde{R}$, $R_2 = \widehat{\tilde{R}}$, $Q_1 = Q^{\pi_b}$, and $Q_2 = \widehat{Q}^{\pi_b}$, we have:

$$\begin{aligned} |\rho(\pi_{val}^{\odot}, \tilde{M}) - \rho(\pi_{val}^{\odot}, \widehat{\tilde{M}})| &= |V_{\tilde{M}}^{\pi_{val}^{\odot}}(x_0) - V_{\widehat{\tilde{M}}}^{\pi_{val}^{\odot}}(x_0)| && (33) \\ &\leq (\mathbb{I} - \gamma\dot{\pi}_{val}^{\odot}\tilde{P})^{-1}\epsilon V_{max} && (34) \end{aligned}$$

And if we write $\pi^* = \operatorname{argmax}_{\pi\in\Pi} \rho(\pi, \tilde{M})$, analogously, we also have:

$$|\rho(\pi^*, \tilde{M}) - \rho(\pi^*, \widehat{\tilde{M}})| \leq (\mathbb{I} - \gamma\dot{\pi}^*\tilde{P})^{-1}\epsilon V_{max} \tag{35}$$

Thus, we may write:

$$\begin{aligned} \rho(\pi^*, \tilde{M}) - \rho(\pi_{val}^{\odot}, \tilde{M}) &\overset{(a)}{\leq} \rho(\pi^*, \tilde{M}) - \rho(\pi_{val}^{\odot}, \widehat{\tilde{M}}) + (\mathbb{I} - \gamma\dot{\pi}_{val}^{\odot}\tilde{P})^{-1}\epsilon V_{max} && (36) \\ &\overset{(b)}{\leq} \rho(\pi^*, \tilde{M}) - \rho(\pi^*, \widehat{\tilde{M}}) + (\mathbb{I} - \gamma\dot{\pi}_{val}^{\odot}\tilde{P})^{-1}\epsilon V_{max} && (37) \\ &\overset{(c)}{\leq} (\mathbb{I} - \gamma\dot{\pi}^*\tilde{P})^{-1}\epsilon V_{max} + (\mathbb{I} - \gamma\dot{\pi}_{val}^{\odot}\tilde{P})^{-1}\epsilon V_{max} && (38) \\ &\overset{(d)}{\leq} \frac{2\epsilon V_{max}}{1-\gamma}, && (39) \end{aligned}$$

where each step is obtained as follows:

(a) From equation 34.

(b) Optimality of $\pi_{val}^{\odot}$ in the estimated Q^{π_b}-bootstrapped MDP $\widehat{\tilde{M}}$.

(c) From equation 35.

(d) For any policy $\pi \in \Pi$, we have $\|(\mathbb{I} - \gamma\dot{\pi}\tilde{P})^{-1}\|_1 \leq \frac{1}{1-\gamma}$.

□

A.4. Safe policy improvement

Proposition 2 (Baseline policy value conservation under Q^{π_b}-bootstrapping)**.**

$$V^{\pi_b}_{\tilde{M}} = V^{\pi_b}_{M^*}. \tag{40}$$

Proof. The V-value function can be decomposed as follows:

$$\begin{aligned} V^{\pi_b}_{\tilde{M}} &= \pi_b \left(\tilde{R} + \tilde{P} V^{\pi_b}_{\tilde{M}}\right) && (41)\\ &= \dot{\pi}_b \left(\tilde{R} + \tilde{P} V^{\pi_b}_{\tilde{M}}\right) + \tilde{\pi}_b \left(\tilde{R} + \tilde{P} V^{\pi_b}_{\tilde{M}}\right) && (42)\\ &= \dot{\pi}_b \left(R^* + P^* V^{\pi_b}_{\tilde{M}}\right) + \tilde{\pi}_b \left(Q^{\pi_b}_{M^*}\right) && (43)\\ &= \dot{\pi}_b \left(R^* + P^* V^{\pi_b}_{\tilde{M}}\right) + \tilde{\pi}_b \left(R^* + P^* V^{\pi_b}_{M^*}\right). && (44) \end{aligned}$$

From Equation 44, it is direct to conclude that $V^{\pi_b}_{M^*}$ is the unique solution of the Bellman equation for $V^{\pi_b}_{\tilde{M}}$, and therefore that $V^{\pi_b}_{\tilde{M}} = V^{\pi_b}_{M^*}$. □

Corollary 1 (Baseline policy return conservation under bootstrapping)**.**

$$\rho(\pi_b, \tilde{M}) = \rho(\pi_b, M^*). \tag{45}$$

Proof. This is a direct consequence from Proposition 2:

$$\rho(\pi_b, \tilde{M}) = V^{\pi_b}_{\tilde{M}}(x_0) = V^{\pi_b}_{M^*}(x_0) = \rho(\pi_b, M^*). \tag{46}$$

□

Theorem 3 (Safe policy improvement of Q^{π_b}-SPIBB-1)**.** *Let $\pi^{\odot}_{val}$ be an optimal policy of the reward maximization problem of an estimated Q^{π_b}-bootstrapped MDP $\widehat{\tilde{M}}$. Then, under the construction properties of $\tilde{M}$ and under the assumption of Proposition 1, $\pi^{\odot}_{val}$ applied in $\tilde{M}$ is an approximate safe policy improvement over the baseline policy π_b with high probability $1-\delta$:*

$$\rho(\pi^{\odot}_{val}, \tilde{M}) \geq \rho(\pi_b, M^*) - \frac{2\epsilon V_{max}}{1-\gamma} \tag{47}$$

Proof.

$$\begin{aligned} \rho(\pi^{\odot}_{val}, \tilde{M}) &\overset{(a)}{\geq} \max_{\pi \in \Pi} \rho(\pi, \tilde{M}) - \frac{2\epsilon V_{max}}{1-\gamma} && (48)\\ &\overset{(b)}{\geq} \rho(\pi_b, \tilde{M}) - \frac{2\epsilon V_{max}}{1-\gamma} && (49)\\ &\overset{(c)}{\geq} \rho(\pi_b, M^*) - \frac{2\epsilon V_{max}}{1-\gamma}, && (50) \end{aligned}$$

where each step is obtained as follows:

(a) From Theorem 2.

(b) Optimality of $\max_{\pi \in \Pi} \rho(\pi, \tilde{M})$.

(c) From Corollary 1.

□

B. Proofs for Π_b-SPIBB (Section 3.3)

B.1. Q-function error bounds with Π_b-SPIBB

Lemma 2 (Q-function error bounds with Π_b-SPIBB)**.** *Consider two transition probability matrices P_1 and P_2 and two reward functions R_1 and R_2 over an MDP M. Consider a policy $\pi \in \Pi_b$ satisfying the following constraint: if $(x,a) \in \mathfrak{B}$, $\pi(x,a) = \pi_b(x,a)$. Also, consider Q_1 and Q_2 be the state-action value function of the policy π given (P_1, R_1) and (P_2, R_2), respectively. Under the construction properties of $\mathfrak{B}$, we have:*

$$\forall (x,a) \notin \mathfrak{B}, |Q_1(x,a) - Q_2(x,a)| \leq (\mathbb{I} - \gamma P_1 \pi)^{-1} V_{max} \epsilon, \tag{51}$$

where V_{max} is the known maximum of the value function.

Proof. For any state-action pair, the difference between the two state-action value functions can be written:

$$\begin{aligned} Q_1 - Q_2 &= R_1 + \gamma P_1 \pi Q_1 - R_2 - \gamma P_2 \pi Q_2 & (52)\\ &= R_1 + \gamma P_1 \pi Q_1 - R_2 - \gamma P_2 \pi Q_2 + \gamma P_1 \pi Q_2 - \gamma P_1 \pi Q_2 & (53)\\ &= R_1 - R_2 - \gamma P_1 \pi (Q_1 - Q_2) + \gamma (P_1 - P_2) \pi Q_2 & (54)\\ &= (\mathbb{I} - \gamma P_1 \pi)^{-1} \left[R_1 - R_2 + \gamma (P_1 - P_2) \pi Q_2 \right]. & (55) \end{aligned}$$

Now using the Holder's inequality, for any state-action couple $(x,a) \notin \mathfrak{B}$, we have:

$$|(P_1(\cdot|x,a) - P_2(\cdot|x,a))^\top \pi Q_2| \leq \|P_1(\cdot|x,a) - P_2(\cdot|x,a)\|_1 \|\pi\|_\infty \|Q_2\|_\infty \leq V_{max}\epsilon. \tag{56}$$

Also, considering the reward term, for any state-action couple $(x,a) \notin \mathfrak{B}$, we get:

$$|R_1(x,a) - R_2(x,a)| \leq \epsilon R_{max} \tag{57}$$

Inserting 56 and 57 into Equation 55 gives for any state-action couple $(x,a) \notin \mathfrak{B}$:

$$\begin{aligned} |Q_1(x,a) - Q_2(x,a)| &\leq (\mathbb{I} - \gamma P_1 \pi)^{-1} \left[R_{max}\epsilon + \gamma V_{max} \epsilon \right] & (58)\\ &\leq (\mathbb{I} - \gamma P_1 \pi)^{-1} V_{max} \epsilon, & (59) \end{aligned}$$

which proves the lemma. □

B.2. Near optimality of Π_b-SPIBB

Theorem 4 (Near Π_b-optimality of Π_b-SPIBB)**.** *Let Π_b be the set of policies under the constraint of following π_b when $(x,a) \in \mathfrak{B}$. Let $\pi_{pol}^{\odot}$ be a Π_b-optimal policy of the reward maximization problem of an estimated MDP $\widehat{M}$. Then, the performance of $\pi_{pol}^{\odot}$ is near Π_b-optimal in the true MDP M^*:*

$$\rho(\pi_{pol}^{\odot}, M^*) \geq \max_{\pi \in \Pi_b} \rho(\pi, M^*) - \frac{2\epsilon V_{max}}{1-\gamma}. \tag{60}$$

Proof. From Lemma 2, with $\pi = \pi_{pol}^{\odot}$, $P_1 = P^*$, $P_2 = \widehat{P}$, $R_1 = R^*$, and $R_2 = \widehat{R}$, we have:

$$\begin{aligned} |\rho(\pi_{pol}^{\odot}, M^*) - \rho(\pi_{pol}^{\odot}, \widehat{M})| &= |V_{M^*}^{\pi_{pol}^{\odot}}(x_0) - V_{\widehat{M}}^{\pi_{pol}^{\odot}}(x_0)| & (61)\\ &= |Q_{M^*}^{\pi_{pol}^{\odot}}(x_0, a_0) - Q_{\widehat{M}}^{\pi_{pol}^{\odot}}(x_0, a_0)| & (62)\\ &\leq (\mathbb{I} - \gamma P^* \pi_{pol}^{\odot})^{-1} \epsilon V_{max} & (63) \end{aligned}$$

Let π^* denote the optimal policy in the set of admissible policies Π_b: $\pi^* = \text{argmax}_{\pi \in \Pi_b} \rho(\pi, M^*)$. Analogously to 63, we also have:

$$|\rho(\pi^*, M^*) - \rho(\pi^*, \widehat{M})| \leq (\mathbb{I} - \gamma P^* \pi^*)^{-1} \epsilon V_{max} \tag{64}$$

Thus, we may write:

$$\begin{aligned}
\rho(\pi^*, M^*) - \rho(\pi^{\odot}_{pol}, M^*) &\overset{(a)}{\leq} \rho(\pi^*, M^*) - \rho(\pi^{\odot}_{pol}, \widehat{M}) + (\mathbb{I} - \gamma P^* \pi^{\odot}_{pol})^{-1} \epsilon V_{max} && (65)\\
&\overset{(b)}{\leq} \rho(\pi^*, M^*) - \rho(\pi^*, \widehat{M}) + (\mathbb{I} - \gamma P^* \pi^{\odot}_{pol})^{-1} \epsilon V_{max} && (66)\\
&\overset{(c)}{\leq} (\mathbb{I} - \gamma P^* \pi^*)^{-1} \epsilon V_{max} + (\mathbb{I} - \gamma P^* \pi^{\odot}_{pol})^{-1} \epsilon V_{max} && (67)\\
&\overset{(d)}{\leq} \frac{2\epsilon V_{max}}{1-\gamma}, && (68)
\end{aligned}$$

where each step is obtained as follows:

(a) From equation 63.

(b) Optimality of $\pi^{\odot}_{pol}$ in the estimated MDP $\widehat{M}$.

(c) From equation 64.

(d) For any policy $\pi \in \Pi$, we have $\|(\mathbb{I} - \gamma P^* \pi)^{-1}\|_1 \leq \frac{1}{1-\gamma}$.

□

B.3. Safe policy improvement of Π_b-SPIBB

Theorem 5 (Safe policy improvement of Π_b-SPIBB)**.** *Let Π_b be the set of policies under the constraint of following π_b when $(x, a) \in \mathfrak{B}$. Let $\pi^{\odot}_{pol}$ be a Π_b-optimal policy of the reward maximization problem of an estimated MDP $\widehat{M}$. Then, $\pi^{\odot}_{pol}$ is an approximate safe policy improvement over the baseline policy π_b with high probability $1 - \delta$:*

$$\rho(\pi^{\odot}_{pol}, M^*) \geq \rho(\pi_b, M^*) - \frac{2\epsilon V_{max}}{1-\gamma}. \tag{69}$$

Proof. It is direct to observe that $\pi_b \in \Pi_b$, and therefore that:

$$\max_{\pi \in \Pi_b} \rho(\pi, M^*) \geq \rho(\pi_b, M^*) \tag{70}$$

It follows from Theorem 4 that:

$$\rho(\pi^{\odot}_{pol}, M^*) \geq \rho(\pi_b, M^*) - \frac{2\epsilon V_{max}}{1-\gamma}. \tag{71}$$

□

C. Algorithms for the greedy projection of $Q^{(i)}$ on Π_b, Π_0, and $\Pi_{\leq b}$

The policy-based SPIBB algorithms rely on a policy iteration process that requires a policy improvement step under the constraint of the generated policy to belong to Π_b, Π_0, or $\Pi_{\leq b}$. Those are respectively described in Algorithms 4, 5, and 6.

Algorithm 4 Greedy projection of $Q^{(i)}$ on Π_b

```
Data: Baseline policy $\pi_b$
Data: Last iteration value function $Q^{(i)}$
Data: Set of bootstrapped state-action pairs $\mathfrak{B}$
Data: Current state $x$ and action set $\mathcal{A}$
Initialize $\pi_{pol}^{(i)} = 0$
for $(x,a) \in \mathfrak{B}$ do $\pi_{pol}^{(i)}(x,a) = \pi_b(x,a)$ ;
$\pi_{pol}^{(i)}(x, \text{argmax}_{(x,a)\notin\mathfrak{B}} Q^{(i)}(x,a)) = 1 - \sum_{(x,a')\in\mathfrak{B}} \pi_b(x,a')$
return $\pi_{pol}^{(i)}$
```

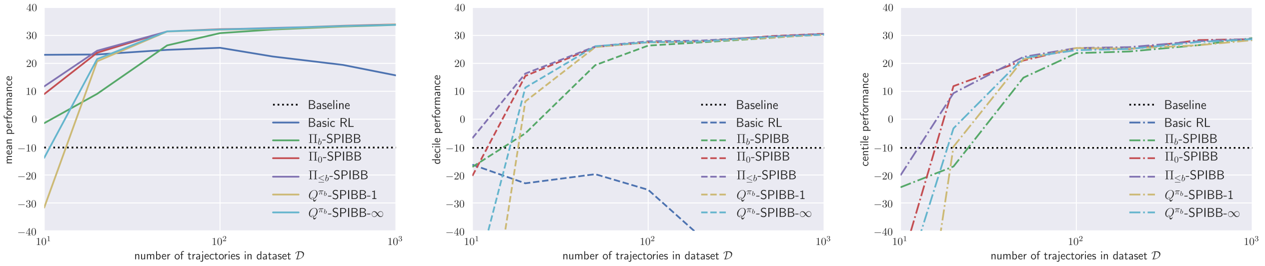
Algorithm 5 Greedy projection of $Q^{(i)}$ on Π_0

```
Data: Baseline policy $\pi_b$
Data: Last iteration value function $Q^{(i)}$
Data: Set of bootstrapped state-action pairs $\mathfrak{B}$
Data: Current state $x$ and action set $\mathcal{A}$
Initialize $\pi_{pol}^{(i)} = 0$
$\pi_{pol}^{(i)}(x, \text{argmax}_{(x,a)\notin\mathfrak{B}} Q^{(i)}(x,a)) = 1$
return $\pi_{pol}^{(i)}$
```

Algorithm 6 Greedy projection of $Q^{(i)}$ on $\Pi_{\leq b}$

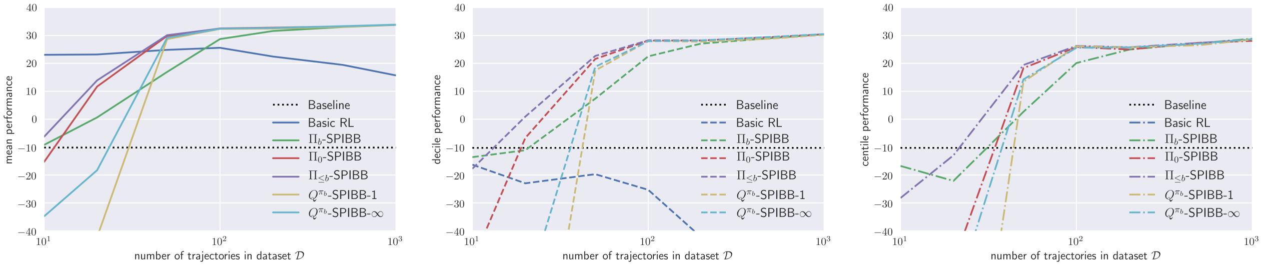
```
Data: Baseline policy $\pi_b$
Data: Last iteration value function $Q^{(i)}$
Data: Set of bootstrapped state-action pairs $\mathfrak{B}$
Data: Current state $x$ and action set $\mathcal{A}$
Sort $\mathcal{A}$ in decreasing order of the action values: $Q^{(i)}(x,a)$
Initialize $\pi_{pol}^{(i)} = 0$
for $a \in \mathcal{A}$ do
    if $(x,a) \in \mathfrak{B}$ then
        if $\pi_b(a|x) \geq 1 - \sum_{a'\in\mathcal{A}} \pi_{pol}^{(i)}(a'|x)$ then
            $\pi_{pol}^{(i)}(a|x) = 1 - \sum_{a'\in\mathcal{A}} \pi_{pol}^{(i)}(a'|x)$
            return $\pi_{pol}^{(i)}$
        else
            $\pi_{pol}^{(i)}(a|x) = \pi_b(a|x)$
        end
    else
        $\pi_{pol}^{(i)}(a|x) = 1 - \sum_{a'\in\mathcal{A}} \pi_{pol}^{(i)}(a'|x)$
        return $\pi_{pol}^{(i)}$
    end
end
```

D. Extensive SPIBB experimental results



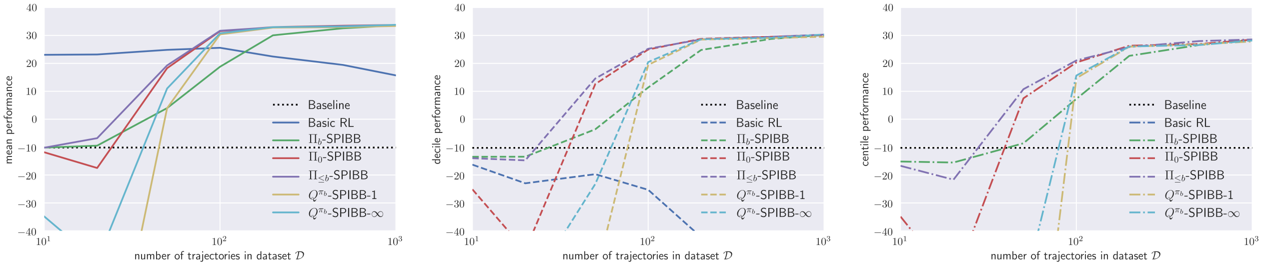
(a) Mean performance (b) Worst decile performance (c) Worst centile performance

Figure 5: SPIBB benchmark ($N_\wedge = 5$): mean, worst decile and worst centile performances.



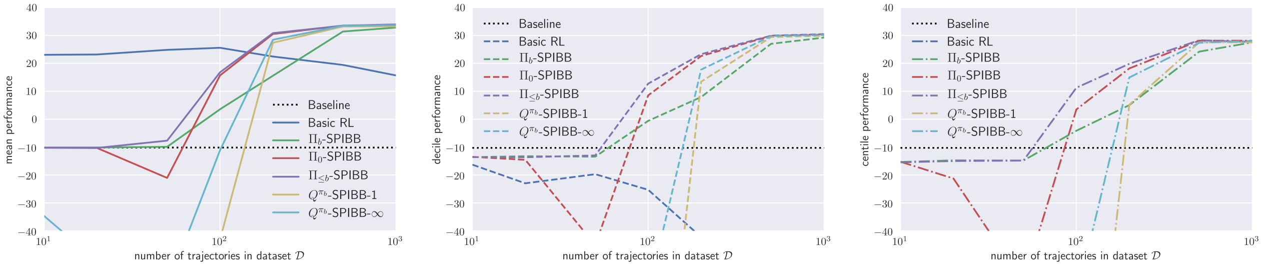
(a) Mean performance (b) Worst decile performance (c) Worst centile performance

Figure 6: SPIBB benchmark ($N_\wedge = 10$): mean, worst decile and worst centile performances.



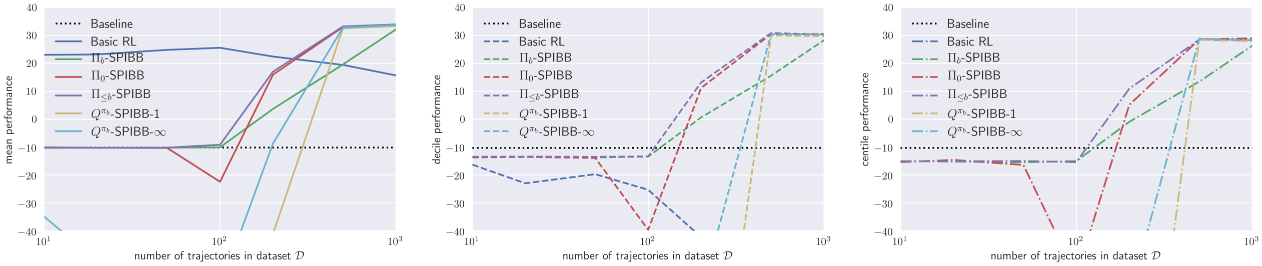
(a) Mean performance (b) Worst decile performance (c) Worst centile performance

Figure 7: SPIBB benchmark ($N_\wedge = 20$): mean, worst decile and worst centile performances.



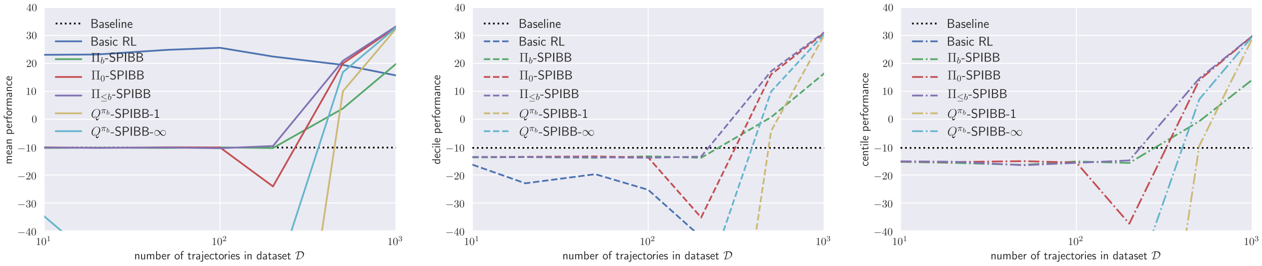
(a) Mean performance (b) Worst decile performance (c) Worst centile performance

Figure 8: SPIBB benchmark ($N_\wedge = 50$): mean, worst decile and worst centile performances.



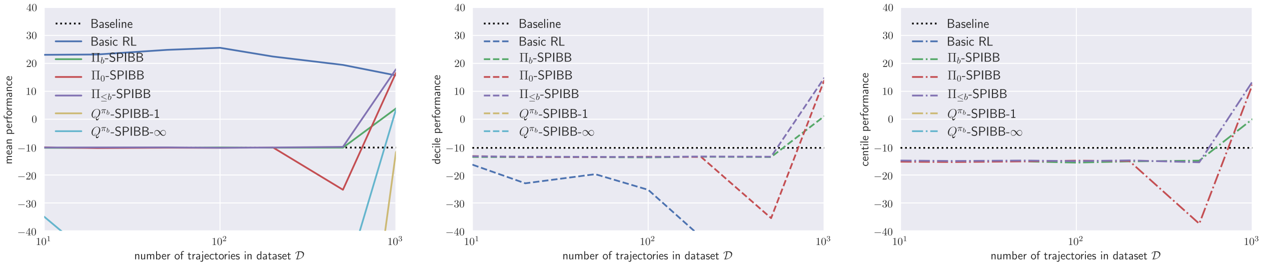
(a) Mean performance (b) Worst decile performance (c) Worst centile performance

Figure 9: SPIBB benchmark ($N_\wedge = 100$): mean, worst decile and worst centile performances.



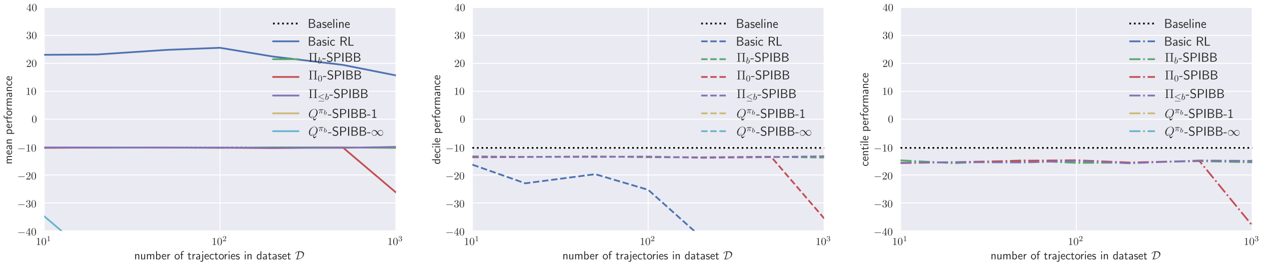
(a) Mean performance (b) Worst decile performance (c) Worst centile performance

Figure 10: SPIBB benchmark ($N_\wedge = 200$): mean, worst decile and worst centile performances.



(a) Mean performance (b) Worst decile performance (c) Worst centile performance

Figure 11: SPIBB benchmark ($N_\wedge = 500$): mean, worst decile and worst centile performances.



(a) Mean performance (b) Worst decile performance (c) Worst centile performance

Figure 12: SPIBB benchmark ($N_\wedge = 1000$): mean, worst decile and worst centile performances.