Safe Policy Improvement with Baseline Bootstrapping

Romain Laroche [1] Paul Trichelair [1] Remi Tachet des Combes [1]

Abstract

This paper considers Safe Policy Improvement (SPI) in Batch Reinforcement Learning (Batch RL): from a fixed dataset and without direct access to the true environment, train a policy that is guaranteed to perform at least as well as the baseline policy used to collect the data. Our approach, called SPI with Baseline Bootstrapping (SPIBB), is inspired by the knows-what-it-knows paradigm: it bootstraps the trained policy with the baseline when the uncertainty is high. Our first algorithm, Π_b-SPIBB, comes with SPI theoretical guarantees. We also implement a variant, $\Pi_{\leq b}$-SPIBB, that is even more efficient in practice. We apply our algorithms to a motivational stochastic gridworld domain and further demonstrate on randomly generated MDPs the superiority of SPIBB with respect to existing algorithms, not only in safety but also in mean performance. Finally, we implement a model-free version of SPIBB and show its benefits on a navigation task with deep RL implementation called SPIBB-DQN, which is, to the best of our knowledge, the first RL algorithm relying on a neural network representation able to train efficiently and reliably from batch data, without any interaction with the environment.

1. Introduction

Most real-world Reinforcement Learning agents (Sutton & Barto, 1998, RL) are to be deployed simultaneously on numerous independent devices and cannot be patched quickly. In other practical applications, such as crop management or clinical tests, the outcome of a treatment can only be assessed after several years. Consequently, a bad update could be in effect for a long time, potentially hurting the user's trust and/or causing irreversible damages. Devising safe algorithms with guarantees on the policy performance is a key challenge of modern RL that needs to be tackled before any wide-scale adoption.

Batch RL is an existing approach to such offline settings and consists in training a policy on a fixed set of observations without access to the true environment (Lange et al., 2012). It should not be mistaken with the multi-batch setting where the learner trains successive policies from small batches of interactions with the environment (Duan et al., 2016). Current Batch RL algorithms are however either unsafe or too costly computationally to be used in real-world applications. Safety in RL (García & Fernández, 2015) is an overloaded term, as it may be considered with respect to parametric uncertainty (Thomas et al., 2015a; Petrik et al., 2016), internal uncertainty (Altman, 1999; Carrara et al., 2019), interruptibility (Orseau & Armstrong, 2016; Guerraoui et al., 2017), or as exploration in a hazardous environment (Schulman et al., 2015; 2017; Fatemi et al., 2019). We focus on the former.

In this paper, we develop novel *safe and efficient* Batch RL algorithms. Our methodology for Safe Policy Improvement (SPI), called SPI with Baseline Bootstrapping (SPIBB), is introduced in Section 2. It consists in bootstrapping the trained policy with the behavioral policy, called *baseline*, in the state-action pair transitions that were not probed enough in the dataset. It therefore assumes access to the baseline, an assumption already made in the SPI literature (Petrik et al., 2016). Other SPI algorithms assume knowledge of the baseline performance instead (Thomas et al., 2015a;b). We argue that our assumption is more natural since SPI aims to improve an existing policy. This scenario is typically encountered when a policy is trained in a simulator and then run in its real environment, for instance in Transfer RL (Taylor & Stone, 2009); or when a system is designed with expert knowledge and then optimized, for example in Dialogue applications (Laroche et al., 2010).

Still in Section 2, we implement a computationally efficient algorithm, Π_b-SPIBB, that provably approximately outperforms the baseline with high confidence. At the expense of theoretical guarantees, we design a variant, $\Pi_{\leq b}$-SPIBB, that is even more efficient in practice. Moreover, we implement an equivalent model-free version. Coupled with a pseudo-count implementation (Bellemare et al., 2016), it allows applying SPIBB algorithms to tasks requiring a

[1]Microsoft Research, Montréal, Canada. Correspondence to: Romain Laroche <romain.laroche@microsoft.com>.

Proceedings of the 36^th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

neural network representation. Finally, we position our algorithm with respect to competing SPI algorithms found in the literature.

Then, in Section 3, we motivate our approach on a small stochastic gridworld domain and further demonstrate on randomly generated MDPs the superiority of SPIBB compared to existing algorithms, not only in safety but also in mean performance. Furthermore, we apply the model-free version to a continuous navigation task. It is, to the best of our knowledge, the first RL algorithm relying on a neural network representation able to train efficiently and reliably from batch data, without any interaction with the environment (Duan et al., 2016).

Finally, Section 4 concludes the paper. The appendix includes the proofs, thorough experiment details, and the complete results of experiments. The code may be found at `https://github.com/RomainLaroche/SPIBB` and `https://github.com/rems75/SPIBB-DQN`.

2. SPI with Baseline Bootstrapping

A proper introduction to Markov Decision Processes (Bellman, 1957, MDPs) and Reinforcement Learning (Sutton & Barto, 1998, RL) is available in Appendix A.1. Due to space constraint, we only define our notations here.

An MDP is denoted by $M = \langle \mathcal{X}, \mathcal{A}, R, P, \gamma \rangle$, where $\mathcal{X}$ is the state space, $\mathcal{A}$ is the action space, $R^*(x,a) \in [-R_{max}, R_{max}]$ is the bounded stochastic reward function, $P^*(\cdot|x,a)$ is the transition distribution, and $\gamma \in [0,1)$ is the discount factor. The true environment is modelled as an unknown finite MDP $M^* = \langle \mathcal{X}, \mathcal{A}, R^*, P^*, \gamma \rangle$ with $R^*(x,a) \in [-R_{max}, R_{max}]$. $\Pi = \{\pi : \mathcal{X} \to \Delta_{\mathcal{A}}\}$ is the set of stochastic policies, where $\Delta_{\mathcal{A}}$ denotes the set of probability distributions over the set of actions $\mathcal{A}$.

The state and state-action value functions are respectively denoted by $V^\pi_M(x)$ and $Q^\pi_M(x,a)$. We define the performance of a policy by its expected return, starting from the initial state x_0: $\rho(\pi, M) = V^\pi_M(x_0)$. Given a policy subset $\Pi' \subseteq \Pi$, a policy π' is said to be Π'-optimal for an MDP M when it maximizes its performance on Π': $\rho(\pi', M) = \max_{\pi \in \Pi'} \rho(\pi, M)$. We will also make use of the notation V_{max} as a known upper bound of the return's absolute value: $V_{max} \leq \frac{R_{max}}{1-\gamma}$.

In this paper, we focus on the batch RL setting where the algorithm does its best at learning a policy from a fixed set of experience. Given a dataset of transitions $\mathcal{D} = \langle x_j, a_j, r_j, x'_j \rangle_{j \in [\![1,N]\!]}$, we denote by $N_{\mathcal{D}}(x,a)$ the state-action pair counts; and by $\widehat{M} = \langle \mathcal{X}, \mathcal{A}, \widehat{R}, \widehat{P}, \gamma \rangle$ the Maximum Likelihood Estimation (MLE) MDP of the environment, where $\widehat{R}$ is the reward mean and $\widehat{P}$ is the transition statistics observed in the dataset. Vanilla batch RL, referred hereinafter as Basic RL, looks for the optimal policy in $\widehat{M}$. This policy may be found indifferently using dynamic programming on the explicitly modelled MDP $\widehat{M}$, Q-learning with experience replay until convergence (Sutton & Barto, 1998), or Fitted-Q Iteration with a one-hot vector representation of the state space (Ernst et al., 2005).

2.1. Percentile criterion and Robust MDPs

We start from the *percentile criterion* (Delage & Mannor, 2010) on the safe policy improvement over the baseline π_b:

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

where $\mathbb{P}_{\text{MDP}}(\cdot|\mathcal{D})$ is the posterior probability of the MDP parameters, $1-\delta$ is the high probability meta-parameter, and ζ is the approximation meta-parameter. (Petrik et al., 2016) use Robust MDP (Iyengar, 2005; Nilim & El Ghaoui, 2005) to bound from below the constraint in (1) by considering a set of admissible MDPs $\Xi = \Xi(\widehat{M}, e)$ defined as:

$$\Xi(\widehat{M}, e) := \{M = \langle \mathcal{X}, \mathcal{A}, R, P, \gamma \rangle \quad \text{s.t. } \forall (x,a) \in \mathcal{X} \times \mathcal{A}, \left. \begin{array}{r} ||P(\cdot|x,a) - \widehat{P}(\cdot|x,a)||_1 \leq e(x,a), \\ |R(x,a) - \widehat{R}(x,a)| \leq e(x,a) R_{max} \end{array} \right\} \tag{2}$$

where $e : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is an error function depending on $\mathcal{D}$ and δ. In place of the intractable expectation in Equation (1), Robust MDP classically consider optimizing the policy performance $\rho(\pi, M)$ of the worst-case scenario in Ξ:

$$\pi_R = \underset{\pi \in \Pi}{\operatorname{argmax}} \min_{M \in \Xi} \rho(\pi, M). \tag{3}$$

In our benchmarks, we use the Robust MDP solver described in Petrik et al. (2016). Petrik et al. (2016) also contemplate the policy improvement worst-case scenario:

$$\pi_S = \underset{\pi \in \Pi}{\operatorname{argmax}} \min_{M \in \Xi} \left(\rho(\pi, M) - \rho(\pi_b, M)\right). \tag{4}$$

They prove that this optimization is an NP-hard problem and propose an algorithm approximating the solution without any formal proof: Approximate Robust Baseline Regret Minimization (ARBRM). There are three problems with ARBRM. First, it assumes that there is no error in the transition probabilities of the baseline, which is very restrictive and amounts to Basic RL when the support of the baseline is the full action space in each state (as is the case in all our experiments). Second, given its high complexity, it is difficult to empirically assess its percentile criterion safety except on simple tasks. Third, in order to retain safety guarantees, ARBRM requires a conservative safety test that consistently fails in our experiments. These are the reasons why our benchmarks do not include ARBRM.

2.2. SPIBB methodology

In this section, we reformulate the percentile criterion to make searching for an efficient and provably-safe policy tractable in terms of computer time. Our new criterion consists in optimizing the policy with respect to its performance in the MDP estimate $\widehat{M}$, while guaranteeing it to be ζ-approximately at least as good as π_b in the admissible MDP set Ξ. Formally, we write it as follows:

$$\max_{\pi\in\Pi} \rho(\pi, \widehat{M}), \text{ s.t. } \forall M \in \Xi, \rho(\pi, M) \geq \rho(\pi_b, M) - \zeta. \tag{5}$$

From Theorem 8 of Petrik et al. (2016), it is direct to guarantee that, if all the state-action pair counts satisfy:

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

and if $\widehat{M}$ is the Maximum Likelihood Estimation (MLE) MDP, then, with high probability $1-\delta$, the optimal policy $\pi^\odot = \operatorname{argmax}_{\pi\in\Pi} \rho(\pi, \widehat{M})$ in $\widehat{M}$ is ζ-approximately safe with respect to the true environment M^*:

$$\rho(\pi^\odot, M^*) \geq \rho(\pi^*, M^*) - \zeta \geq \rho(\pi_b, M^*) - \zeta. \tag{7}$$

In the following, we extend this result by allowing constraint (6) to be violated on a subset of the state-action pairs $\mathcal{X} \times \mathcal{A}$, called the bootstrapped set and denoted by $\mathfrak{B}$. $\mathfrak{B}$ is the set of state-action pairs with counts smaller than $N_\wedge$.

2.3. Π_b-SPIBB

In this section, we develop two novel algorithms based on policy bootstrapping and prove associated SPI bounds. More precisely, when a state-action pair (x, a) is rarely seen in the dataset, we propose to rely on the baseline by copying its probability to take action a:

$$\pi_{spibb}^\odot(a|x) = \pi_b(a|x) \text{ if } (x, a) \in \mathfrak{B}. \tag{8}$$

We let Π_b denote the set of policies that verify (8) for all state-action pairs. Our first algorithm, coined Π_b-SPIBB, consists in the usual policy optimization of the expected return $\rho(\pi, \widehat{M})$ under constraint (8). In practice, it may be achieved in a model-based manner by explicitly computing the MDP model $\widehat{M}$, constructing the set of allowed policies Π_b and finally searching for the Π_b-optimal policy $\pi_{spibb}^\odot$ in $\widehat{M}$ using policy iteration over Π_b (Howard, 1966; Puterman & Brumelle, 1979). In the policy evaluation step, the current policy $\pi^{(i)}$ is evaluated as $Q_{\widehat{M}}^{(i)}$. In the policy improvement step, $\pi^{(i+1)}$ is defined as the greedy policy with respect to $Q^{(i)}$ under the constraint of belonging to Π_b (Algorithm 1 describes how to enforce this constraint in linear time).

Algorithm 1 Greedy projection of $Q^{(i)}$ on Π_b

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

The following theorems prove that Π_b-SPIBB converges to a Π_b-optimal policy $\pi_{spibb}^\odot$, and that $\pi_{spibb}^\odot$ is a safe policy improvement over the baseline in the true MDP M^*.

Theorem 1 (Convergence)**.** *Π_b-SPIBB converges to a policy $\pi_{spibb}^\odot$ that is Π_b-optimal in the MLE MDP $\widehat{M}$.*

Theorem 2 (Safe policy improvement)**.** *Let Π_b be the set of policies under the constraint of following π_b when $(x, a) \in \mathfrak{B}$. Then, $\pi_{spibb}^\odot$ is a ζ-approximate safe policy improvement over the baseline π_b with high probability $1-\delta$, where:*

$$\zeta = \frac{4V_{max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge}\log\frac{2|\mathcal{X}||\mathcal{A}|2^{|\mathcal{X}|}}{\delta}} - \rho(\pi_{spibb}^\odot, \widehat{M}) + \rho(\pi_b, \widehat{M})$$

Proofs of both theorems are available in Appendix A.3. Theorem 1 is a direct application of the classical policy iteration theorem. Theorem 2 is a generalization of Theorem 8 in Petrik et al. (2016). The resulting bounds may look very similar at first. The crucial difference is that, in our case, $N_\wedge$ is not a property of the dataset, but a hyper-parameter of the algorithm. In all our experiments, $\|e\|_\infty$ from Theorem 8 would be equal to 2, leading to a trivial bound. In comparison, Π_b-SPIBB allows safe improvement if $N_\wedge$ is chosen large enough to ensure safety and small enough to ensure improvement.

SPIBB takes inspiration from Petrik et al. (2016)'s idea of finding a policy that is guaranteed to be an improvement for any realization of the uncertain parameters, and similarly estimates the error on those parameters, as a function of the state-action pair counts. But instead of searching for the analytic optimum, SPIBB looks for a solution that improves the baseline when it can guarantee improvement and falls back on the baseline when the uncertainty is too high. One can see it as a *knows-what-it-knows* algorithm, asking for help from the baseline when it *does not know whether it knows* (Li et al., 2008). As such, our algorithms can be seen as pessimistic, the flip side of *optimism in the face of uncertainty* (Szita & Lőrincz, 2008). As a consequence, Π_b-

SPIBB is not optimal with respect to the criterion in Equation (5). But in return, it is inherently safe as it only allows to search in a set of policies for which the improvement over the baseline can be safely evaluated (Thomas et al., 2017). It is also worth mentioning that SPIBB is computationally simple, which allows us to develop the SPIBB-DQN algorithm in the next section.

2.4. Model-free Π_b-SPIBB and SPIBB-DQN

The Π_b-SPIBB policy optimization may indifferently be achieved in a model-free manner by fitting the Q-function to the following target $y_j^{(i)}$ over the transition samples in the dataset $\mathcal{D} = \langle x_j, a_j, r_j, x'_j \rangle_{j \in [\![1,N]\!]}$:

$$y_j^{(i)} = r_j + \gamma \sum_{a' | (x'_j, a') \in \mathfrak{B}} \pi_b(a'|x'_j) Q^{(i)}(x'_j, a') \qquad (9)$$
$$+ \gamma \left(\sum_{a' | (x'_j, a') \notin \mathfrak{B}} \pi_b(a'|x'_j) \right) \max_{a' | (x'_j, a') \notin \mathfrak{B}} Q^{(i)}(x'_j, a')$$

The first term r_j is the immediate reward observed during the recorded transition, the second term is the return estimate of the bootstrapped actions (where the trained policy is constrained to the baseline policy), and the third term is the return estimate maximized over the non-bootstrapped actions. SPIBB-DQN is the DQN algorithm fitted to these targets $y_j^{(i)}$ (Mnih et al., 2015). Note that computing the SPIBB targets requires determining the bootstrapped set $\mathfrak{B}$, which relies on an estimate of the state-action counts $\widetilde{N}_{\mathcal{D}}(x, a)$, also called pseudo-counts (Bellemare et al., 2016; Fox et al., 2018; Burda et al., 2019).

Theorem 3. *In finite MDPs, Equation 9 admits a unique fixed point that coincides with the Q-value of the policy trained with model-based* Π_b*-SPIBB.*

2.5. $\Pi_{\leq b}$-SPIBB

In our empirical evaluation, we consider a variant of Π_b-SPIBB: the space of policies to search is relaxed to $\Pi_{\leq b}$, the set of policies that do not to give more weight than π_b to bootstrapped actions. As a consequence, in comparison with Π_b-SPIBB, it allows to cut off bad performing actions even when their estimate is imprecise:

$$\Pi_{\leq b} = \{ \pi \in \Pi \,|\, \pi(a|x) \leq \pi_b(a|x) \text{ if } (x, a) \in \mathfrak{B} \} \quad (10)$$

The resulting algorithm is referred as $\Pi_{\leq b}$-SPIBB and amounts, as for Π_b-SPIBB, to perform a policy iteration under the policy constraint to belong to $\Pi_{\leq b}$. The convergence guarantees of Theorem 1 still apply to $\Pi_{\leq b}$-SPIBB, but we lose the SPI ones.

Algorithm 2 in Appendix A.4, describes the greedy projection of $Q^{(i)}$ on $\Pi_{\leq b}$. Appendix A.5 also includes a comprehensive example that illustrates the difference between the Π_b-SPIBB and $\Pi_{\leq b}$-SPIBB policy improvement steps. Despite the lack of safety guarantees, our experiments show $\Pi_{\leq b}$-SPIBB to be even safer than Π_b-SPIBB while outperforming it in most scenarios. Multi-batch settings – where it may be better to keep exploring the bootstrapped pairs – might be an exception (Lange et al., 2012).

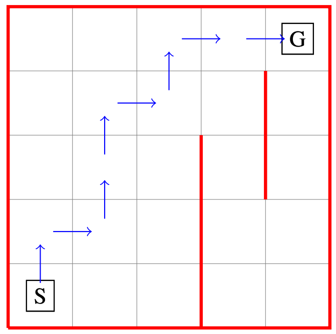
2.6. Other related works

High-Confidence PI refers to the family of algorithms introduced in Paduraru (2013); Mandel et al. (2014); Thomas et al. (2015a), which rely on the ability to produce high-confidence lower bound on the Importance Sampling (IS) estimate of the trained policy performance. IS and SPIBB approaches are very different in nature: IS provides frequentist bounds, while SPIBB provides Bayesian bounds. In comparison to SPIBB, IS has the advantage of not depending on the MDP model and as a consequence may be applied to infinite MDPs with guarantees. However, the IS estimates are known to be high variance. Another drawback of the IS approach is that it fails for long horizon problem. Indeed, Guo et al. (2017) show that the amount of data required by IS-based SPI algorithms scales exponentially with the horizon of the MDP. Regarding the dependency in the horizon of SPIBB algorithms, the discount factor γ is often translated as a planning horizon: $H = \frac{1}{1-\gamma}$. This is the case in UCT for instance (Kocsis & Szepesvári, 2006). As a consequence, Theorem 2 tells us that the safety is linear in the horizon (given a fixed V_{max}).

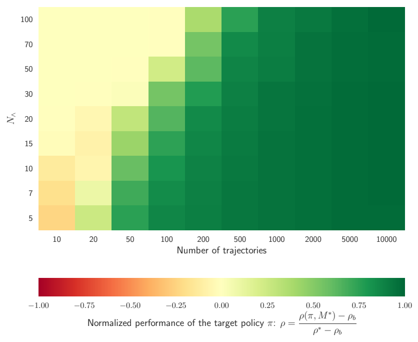
In Kakade & Langford (2002), Conservative Policy Iteration (CPI) not only assumes access to the environment, but also to a μ-restart mechanism which can basically sample at will from the environment according to a distribution μ. This is used in step (2) of the CPI algorithm to build an estimate of the advantage function precise enough to ensure policy improvement with high probability. SPIBB does not have access to the true environment: all it sees are the finite samples from the batch. Similarly, Pirotta et al. (2013a;b) consider a single safe policy improvement in order to speed up training of policy gradients (use less policy iterations). These are however not safe in the sense of finding a policy that improves a previous policy with high confidence: they will converge to the same policy asymptotically, the optimal one in the MLE MDP. Additionally, they are not considering the batch setting.

3. SPIBB Empirical Evaluation

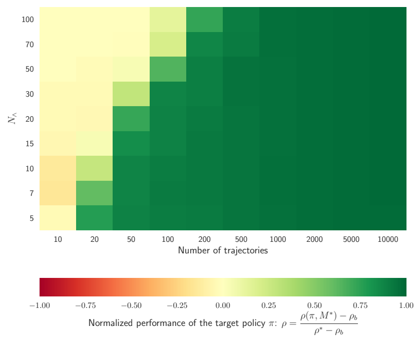
The performance of Batch RL algorithms can vary greatly from one dataset to another. To properly assess existing and SPIBB algorithms, we evaluate their ability to generate policies that consistently outperform the baseline. Practically, we repeated 100k times the following procedure on



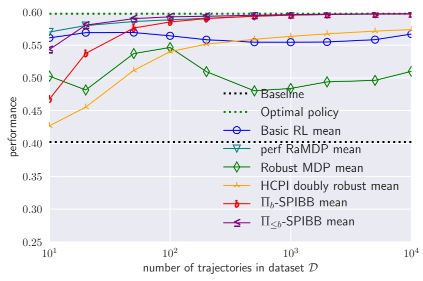
(a) Gridworld domain.



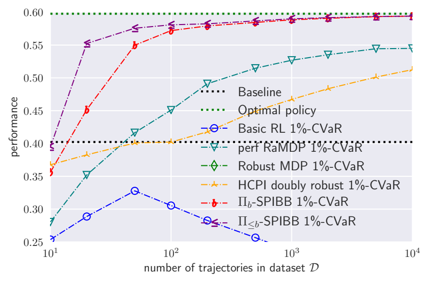
(b) 1%-CVaR heatmap: Π_b-SPIBB.



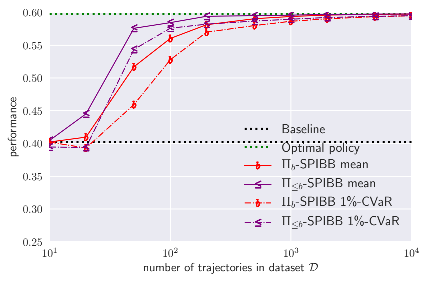
(c) 1%-CVaR heatmap: $\Pi_{\leq b}$-SPIBB.



(d) Mean: benchmark with $N_\wedge = 5$.



(e) 1%-CVaR: benchmark with $N_\wedge = 5$.



(f) Mean & 1%-CVaR: SPIBB w. $N_\wedge = 20$.

Figure 1. Gridworld experiment: Figure (a) illustrates the domain with an optimal trajectory. Figures (b-c) are heatmaps of the 1%-CVaR normalized performance of the SPIBB algorithms as a function of $N_\wedge$. Figures (d-e) show the benchmark for the mean and 1%-CVaR performance. Figure (f) displays additional curves for another value of $N_\wedge$.

various environments: randomly generate a dataset, train a policy on that dataset using each algorithm and each hyper-parameter in the benchmark and compute the performance of the trained policy (with $\gamma = 0.95$). We formalize this experimental protocol in Appendix B.1.1. The algorithms are then evaluated using the mean performance and conditional value at risk performance (CVaR, also called expected shortfall) of the policies they produced. The $X\%$-CVaR is the mean performance over the $X\%$ worst runs. Given the high number of runs, all the results that are visible to the naked eye are significant.

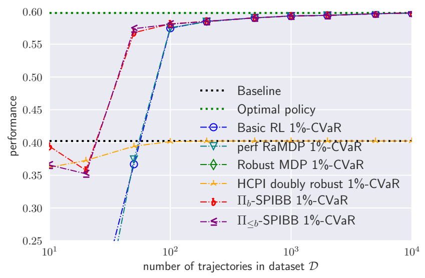
In addition to the SPIBB algorithms, our finite MDP benchmark contains four algorithms: Basic RL, HCPI (Thomas et al., 2015a), Robust MDP, and RaMDP (Petrik et al., 2016). RaMDP stands for Reward-adjusted MDP and applies an exploration penalty when performing actions rarely observed in the dataset. At the exception of Basic RL, they all rely on one hyper-parameter: δ_{hcpi}, δ_{rob} and κ_{adj} respectively. We performed a grid search on those parameters and for HCPI compared 3 versions. In the main text, we only report the best performance we found ($\delta_{hcpi} = 0.9$, $\delta_{rob} = 0.1$, and $\kappa_{adj} = 0.003$), the full results can be found in Appendix B.2. Additionally, Robust MDP and RaMDP depend on a safety test that always failed in our experiments. We still report their performance.

3.1. Does SPIBB outperform existing algorithms?

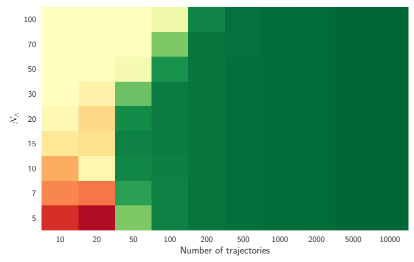
Our first domain is a discrete, stochastic, 5×5 gridworld (see Figure 1(a)), with 4 actions: up, down, left and right. The transitions are stochastic: the agent moves in the requested direction with 75% chance, in the opposite one with 5% chance and to either side with 10% chance each. The initial and final states are respectively the bottom left and top right corners. The reward function is $+1$ when the final state is reached and 0 everywhere else. The baseline we use in this experiment is a fixed stochastic policy with a 0.4 performance, the optimal policy has a 0.6 performance.

We start by analysing the sensitivity of Π_b-SPIBB and $\Pi_{\leq b}$-SPIBB with respect to $N_\wedge$. We visually represent the results as two 1%-CVaR heatmaps: Figures 1(b) and 1(c) for Π_b-SPIBB and $\Pi_{\leq b}$-SPIBB. They read as follows: the colour of a cell indicates the improvement over the baseline normalized with respect to the optimal performance: red, yellow, and green respectively mean below, equal to, and above baseline performance. We observe for SPIBB algorithms that the policy improvement is safe (at the slight exception of Π_b-SPIBB with a low $N_\wedge$ on 10-trajectory datasets), that the bigger the $N_\wedge$, the more conservative SPIBB gets, and that $\Pi_{\leq b}$-SPIBB outperforms Π_b-SPIBB.

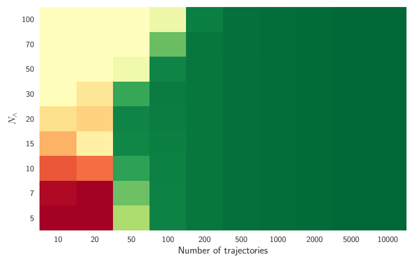
In Figure 1(d), we see that Basic RL improves the base-



(a) 1%-CVaR benchmark with $N_\wedge = 20$.



(b) 1%-CVaR heatmap: Π_b-SPIBB.



(c) 1%-CVaR heatmap: $\Pi_{\leq b}$-SPIBB.

Figure 2. Gridworld experiment with random behavioural policy: Figure (a) shows the benchmark for the 1%-CVaR performance, with SPIBB using $N_\wedge = 20$. Figures (b-c) are the heatmaps of the 1%-CVaR normalized performance of the SPIBB algorithms as a function of $N_\wedge$ (same heat colours as in Figures 1(b) and 1(c)).

line on average, but not monotonically with the size of the dataset, and remains quite far from optimal. That fact is explained by the fairly frequent learning of catastrophic policies and will be analyzed in details with the 1%-CVaR results. HCPI is more conservative for small datasets but slightly outperforms Basic RL for bigger ones, still remaining away from optimal. We also observe that Robust MDPs do even worse than Basic RL; in fact, they learn policies that remain at the center of the grid where the dataset contains a maximum of transitions and therefore where the Robust MDPs have a minimal estimate error, and completely ignore the goal. A similar behaviour is observed with RaMDP when its hyper-parameter is set too high (≥ 0.004). Inversely, when it is set too low (≤ 0.002), RaMDP behaves like Basic RL. But in the tight spot of 0.003, RaMDP is very efficient. We refer the interested reader to Appendix B.2 for the analysis of hyper-parameter search for the benchmark algorithms. Overall, RaMDP and $\Pi_{\leq b}$-SPIBB win this benchmark based on mean performance, with Π_b-SPIBB not far behind.

Figure 1(e) displays the 1%-CVaR performance of the algorithms. We observe that the very good mean performance of RaMDP hides some catastrophic runs where the trained policy under-performs for small datasets. In contrast, $\Pi_{\leq b}$-SPIBB's curve remains over the baseline. Π_b-SPIBB is again a bit behind. HCPI also proves to be near safe. We explained in the previous paragraph why Robust MDP often generates bad policies. It actually does it so often, and the policies are so bad, that its curve does not even show on the graph. Let us now consider Basic RL and explain why it does so poorly, even at times on very large datasets (considering that the MDP has 25 states and 4 actions). The dataset is collected using a baseline that performs some actions only very rarely. As a consequence, even in big datasets, some state-action pairs are observed only once or twice. Given the stochasticity of the environment, the MLE MDP might be quite different from the true MDP in those states, leading to policies falsely taking advantage of those chimaeras. SPIBB algorithms are not allowed to jump to conclusions without sufficient proof and have to conservatively reproduce the baseline policy in those configurations.

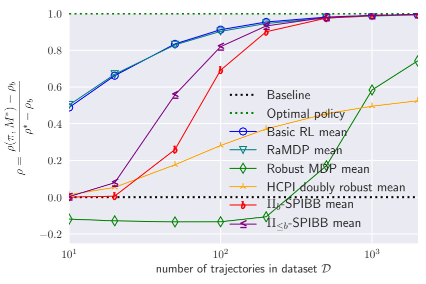
Figure 1(f) shows the SPIBB curves for a higher value of $N_\wedge = 20$. There, the algorithms are more conservative and therefore safe, while still achieving near optimality on big datasets. Full results may be found in Appendix C.1.

3.2. Must the dataset be collected with the baseline?

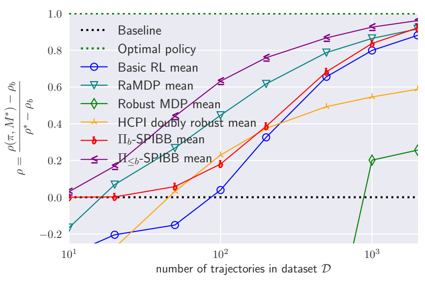
SPIBB theory relies on the assumption that the baseline was used for the data collection, which is a limiting factor of the method. In practice, this assumption simply ensures that the preferential trajectories of the baseline are experienced in the batch of trajectories used for training. We modify the previous experiment by producing datasets using a uniform random policy, while keeping the same Gridworld environment and the same baseline for bootstrapping. In this setting, Basic RL does not have its non-monotonic behaviour anymore, but both our algorithms, Π_b-SPIBB and $\Pi_{\leq b}$-SPIBB, still significantly outperform their competitors (see Figure 2(a)). Note however the following differences: Basic RL becomes safe with 100 trajectories, RaMDP does not improve Basic RL anymore, and HCPI has more difficulty improving the baseline. Robust MDP still does not show on the 1%-CVaR figure. Focusing more specifically on the SPIBB algorithms and their $N_\wedge$ sensitivity, Figures 2(b) and 2(c) show that they fail to be completely safe when $N_\wedge \leq 10$ and $|\mathcal{D}| \leq 20$; and that Π_b-SPIBB slightly outperforms $\Pi_{\leq b}$-SPIBB. Indeed, $\Pi_{\leq b}$-SPIBB cannot take advantage anymore of the bias that the behavioural policy tends to take actions that are better than average. Full results may be found in Appendix C.2.

3.3. Does SPIBB achieve SPI in most domains?

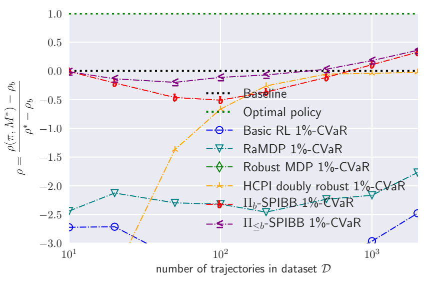
In this section, we study the conditions required on the environment and on the baseline for SPIBB to be helpful. To do so, we use a generator of Random MDPs where the



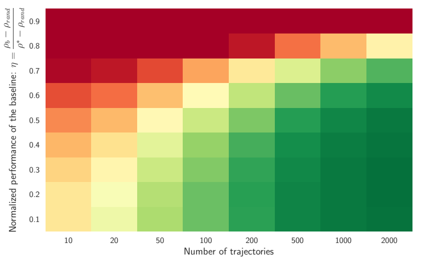
(a) Mean: with $\eta = 0.1$ and $N_\wedge = 10$.



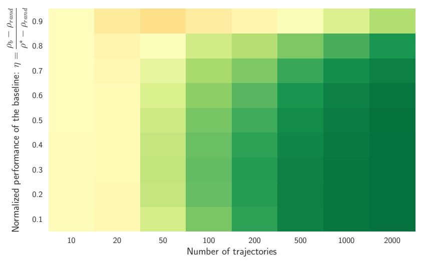
(b) Mean: with $\eta = 0.9$ and $N_\wedge = 10$.



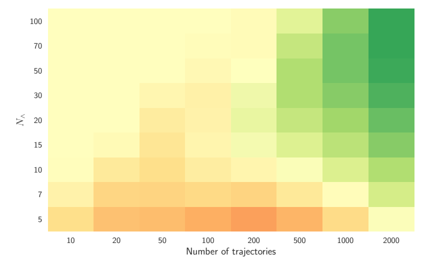
(c) 1%-CVaR: with $\eta = 0.9$ and $N_\wedge = 10$.



(d) 1%-CVaR: RaMDP with $\delta_{adj} = 0.003$.



(e) 1%-CVaR: $\Pi_{\leq b}$-SPIBB with $N_\wedge = 10$.



(f) 1%-CVaR: $\Pi_{\leq b}$-SPIBB with $\eta = 0.9$.

Figure 3. Random MDPs domain: Figures (a-c) show the mean and 1%-CVaR performances for η values of 0.1 and 0.9 and SPIBB with $N_\wedge = 10$. Figures (d-e) are the 1%-CVaR as a function of η for RaMDP and $\Pi_{\leq b}$-SPIBB respectively. Figure (f) is the 1%-CVaR heatmap for $\Pi_{\leq b}$-SPIBB as a function of $N_\wedge$ with $\eta = 0.9$.

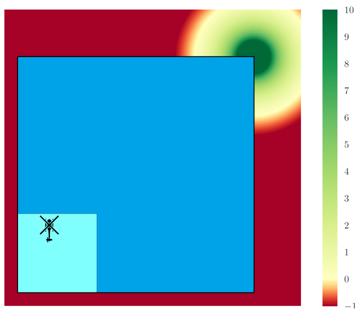
number of states has been fixed to $|\mathcal{X}| = 50$, the number of actions to $|\mathcal{A}| = 4$ and the connectivity of the transition function to 4. This means that for a given state-action pair (x, a), its transition function $P(x'|x, a)$ is non-zero on four states x' only. The initial state is fixed at x_0. The reward function is 0 everywhere except when entering the terminal state, where it equals 1. The terminal state is chosen in such a way that the optimal value function is minimal. It coarsely amounts to choosing the state that is the hardest to reach/farthest from x_0. For a randomly generated MDP M, we generate baselines with different levels of performance (the process is detailed in Appendix B.1.4). Specifically, we set a target performance for the baseline based on a hyper-parameter $\eta \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$: $\rho(\pi_b, M) = \eta\rho(\pi^*, M) + (1 - \eta)\rho(\tilde{\pi}, M)$, where π^* and $\tilde{\pi}$ are respectively the optimal and the uniform policies.

Figure 3(a) shows the mean results with a bad, highly stochastic baseline ($\eta = 0.1$). Since, the baseline is bad, it is an easy task to safely improve it. Basic RL and RaMDP dominate the benchmark in mean, but also in safety (not shown). SPIBB algorithms are too conservative for small datasets but catch up on the bigger ones. Figure 3(b) shows the mean results with a very good baseline, therefore very hard task to safely improve. On average, the podium is composed by $\Pi_{\leq b}$-SPIBB, RaMDP, Π_b-SPIBB, followed closely by Basic RL. But, when one considers more specifically the 1%-CVaR performance, all fail to be safe but the SPIBB algorithms. Note that a -0.5 normalized performance is still a good performance, and that this loss is actually predicted by the theory: Theorem 2 proves a ζ-approximate safe policy improvement.

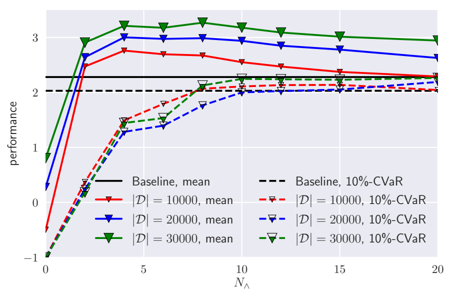
The heatmaps shown in Figures 3(d) and 3(e) allow us to compare more globally the 1%-CVaR performance of RaMDP and $\Pi_{\leq b}$-SPIBB. One observes that the former is unsafe in a large area of the map (where it is red, for high η or small datasets), while the latter is safe everywhere. Figure 3(f) displays a heatmap of the $\Pi_{\leq b}$-SPIBB 1%-CVaR performance in the hardest scenario ($\eta = 0.9$) in function of its $N_\wedge$ hyper-parameter. Unsurprisingly, the algorithm becomes slightly unsafe when $N_\wedge$ gets too low. As it increases, the red stains disappear meaning that it becomes completely safe. The green sections show that it still allows for some policy improvement. Full results may be found in Appendix C.3.

3.4. Does SPIBB scale to larger tasks?

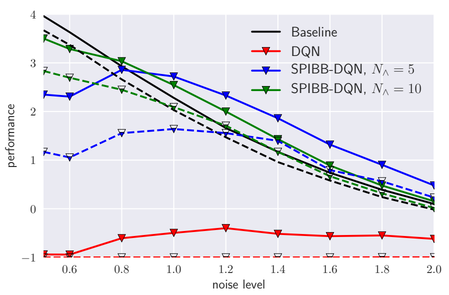
For the sake of simplicity and to be able to repeat several runs of each experiment efficiently, instead of applying pseudo-count methods from the literature (Bellemare et al., 2016; Fox et al., 2018; Burda et al., 2019), we consider here a pseudo-count heuristic based on the Euclidean state-distance, and a task where it makes sense to do so. The pseudo-count of a state-action (x, a) is defined as the sum of its similarity with the state-action pairs (x_i, a_i) found



(a) Helicopter environment.



(b) Mean and 10%-CVaR in function of $N_\wedge$.



(c) Performance in function of the noise factor.

Figure 4. SPIBB-DQN experiments: Figure (a) is an illustration of the environment. Figure (b) displays the mean and 10%-CVaR performance as a function of $N_\wedge$ for three dataset sizes. Figure (c) displays the mean and 10%-CVaR performance for the baseline, vanilla DQN, RaMDP with $\kappa_{adj} = 0.01$, SPIBB-DQN with $N_\wedge = 5$, and with $N_\wedge = 10$, as a function of the transition noise factor.

in the dataset. The similarity between (x, a) and (x_i, a_i) is equal to 0 if $a_i \neq a$, and to $\max(0, 1 - d(x, x_i))$ otherwise, where $d(\cdot, \cdot)$ is the Euclidean distance between two states.

We consider a helicopter navigation task (see Figure 4(a)). The helicopter starts from a random position in the teal area, with a random initial velocity. The 9 available actions consist in applying thrust: backward, no, or forward acceleration, along the two dimensions. The episode terminates when the velocity exceeds some maximal value, in which case it gets a -1 reward, or when the helicopter leaves the blue area, in which case it gets a reward as chromatically indicated on Figure 4(a). The dynamics of the domain follow the basic laws of physics with a Gaussian centered additive noise both on the position and the velocity, see Appendix D.1 for full details of the domain. To train our algorithms, we use a discount factor $\gamma = 0.9$, but we report in our results the undiscounted final reward. The baseline is generated as follows: we first train a policy with online DQN, stop before full convergence and then apply a softmax on the obtained Q-network. Our experiments consist in 300 runs on SPIBB-DQN with a range of $N_\wedge$ values and for different dataset sizes. SPIBB-DQN with $N_\wedge = 0$ is equivalent to vanilla DQN. We also tried RaMDP with several values of $\kappa_{adj} \in [0.001, 0.1]$ without any success. For figure clarity, we do not report RaMDP in the Main Document figures. The set of used parameters and the results of the preliminary experiments are reported in Appendices D.3 and D.4.

Figure 4(b) displays the mean and 10%-CVaR performances in function of $N_\wedge$ for three dataset sizes (10k, 20k, and 30k). We observe that vanilla DQN ($N_\wedge = 0$) significantly worsens the baseline in mean and achieves the worst possible 10%-CVaR performance. SPIBB-DQN not only significantly improves the baseline in mean performance for $N_\wedge \geq 1$, but also in 10%-CVaR when $N_\wedge \geq 8$. The discerning reader might wonder about the CVaR curve for the baseline. It is explained by the fact that the evaluation of the policies are not exact. The curve accounts for the evaluation errors, errors also obviously encountered with the trained policies.

We performed an additional experiment. Keeping the baseline identical, we trained on 10k-transitions datasets obtained from environments with a different transition noise. Figure 4(c) shows the mean and 10%-CVaR performances for the baseline, vanilla DQN, and SPIBB-DQN with $N_\wedge \in \{5, 10\}$. First, we observe that vanilla DQN performs abysmally. Second, we see that the baseline quickly gets more efficient when the noise is removed making the safe policy improvement task harder for SPIBB-DQN. SPIBB is efficient at dealing with stochasticity, the noise attenuation reduces its usefulness. Third, as we get to higher noise factors, the stochasticity becomes too high to efficiently aim at the goal, but SPIBB algorithms still succeed at safely improving the baseline.

4. Conclusion and Future Work

In this paper, we tackle the problem of safe Batch Reinforcement Learning. We reformulate the percentile criterion without compromising its safety. We lose optimality that way but keep a PAC-style guarantee of policy improvement. It allows the implementation of an algorithm Π_b-SPIBB that run as fast as a vanilla model-based RL algorithm, while generating a provably safe policy improvement over a known baseline π_b. A variant algorithm $\Pi_{\leq b}$-SPIBB is shown to perform better and safer on a wide range of domains, but does not come with safety guarantees. Basic Batch RL and the other benchmark competitors are shown to fall short on at least one, and generally two, of the following criteria: mean performance, safety, or domain-dependent hyper-parameter sensitivity. Finally, we implement a DQN version of SPIBB that is the first deep batch algorithm allowing policy improvement in a safe manner.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Altman, E. *Constrained Markov Decision Processes*. CRC Press, 1999.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. In *Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS)*, 2016.

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

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Carrara, N., Leurent, E., Laroche, R., Urvoy, T., Maillard, O., and Pietquin, O. Scaling up budgeted reinforcement learning. *CoRR*, abs/1903.01004, 2019. URL `http://arxiv.org/abs/1903.01004`.

Chollet, F. et al. Keras. `https://keras.io`, 2015.

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

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pp. 1329–1338, 2016.

Efron, B. Better bootstrap confidence intervals. *Journal of the American statistical Association*, 82(397):171–185, 1987.

Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

Fatemi, M., Sharma, Shikharand van Seijen, H., and Ebrahimi Kahou, S. Dead-ends and secure exploration in reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Fox, L., Choshen, L., and Loewenstein, Y. Dora the explorer: Directed outreaching reinforcement action-selection. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

García, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 2015.

Gosavi, A. A reinforcement learning algorithm based on policy iteration for average reward: Empirical results with yield management and convergence analysis. *Machine Learning*, 2004.

Guerraoui, R., Hendrikx, H., Maurer, A., et al. Dynamic safe interruptibility for decentralized multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 130–140, 2017.

Guo, Z., Thomas, P. S., and Brunskill, E. Using options and covariance testing for long horizon off-policy policy evaluation. In *Proceedings of the 30th Advances in Neural Information Processing Systems (NIPS)*, pp. 2492–2501, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015.

Howard, R. A. Dynamic programming. *Management Science*, 1966.

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

Jiang, N. and Li, L. Doubly robust off-policy value evaluation for reinforcement learning. *arXiv preprint arXiv:1511.03722*, 2015.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, volume 2, pp. 267–274, 2002.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

Lange, S., Gabel, T., and Riedmiller, M. Batch reinforcement learning. In *Reinforcement learning*. Springer, 2012.

Laroche, R., Putois, G., and Bretier, P. Optimising a handcrafted dialogue system design. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2010.

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

Mandel, T., Liu, Y.-E., Levine, S., Brunskill, E., and Popovic, Z. 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.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 2015.

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

Orseau, L. and Armstrong, S. Safely interruptible agents. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence (UAI)*, UAI'16, pp. 557–566, Arlington, Virginia, United States, 2016. AUAI Press. ISBN 978-0-9966431-1-5. URL http://dl.acm.org/citation.cfm?id=3020948.3021006.

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

Parr, R. E. and Russell, S. *Hierarchical control and learning for Markov decision processes*. University of California, Berkeley, CA, 1998.

Petrik, M., Ghavamzadeh, M., and Chow, Y. Safe policy improvement by minimizing robust baseline regret. In *Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS)*, 2016.

Pirotta, M., Restelli, M., and Bascetta, L. Adaptive step-size for policy gradient methods. In *Proceedings of the 26th Advances in Neural Information Processing Systems (NIPS)*, pp. 1394–1402, 2013a.

Pirotta, M., Restelli, M., Pecorino, A., and Calandriello, D. Safe policy iteration. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pp. 307–315, 2013b.

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

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 1999.

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

Taylor, M. E. and Stone, P. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

Thomas, P. S., Theocharous, G., and Ghavamzadeh, M. High confidence policy improvement. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015a.

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

Thomas, P. S., da Silva, B. C., Barto, A. G., and Brunskill, E. On ensuring that intelligent machines are well-behaved. *arXiv preprint arXiv:1708.05448*, 2017.

Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015. URL http://dblp.uni-trier.de/db/journals/corr/corr1509.html#HasseltGS15.