Safe Policy Improvement with Soft Baseline Bootstrapping

Kimia Nadjahi[1(✉)], Romain Laroche[2(✉)], and Rémi Tachet des Combes[2]

[1] LTCI, Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France
kimia.nadjahi@telecom-paris.fr
[2] Microsoft Research Montréal, Montreal, Canada
{romain.laroche,remi.tachet}@microsoft.com

Abstract. Batch Reinforcement Learning (Batch RL) consists in training a policy using trajectories collected with another policy, called the behavioural policy. Safe policy improvement (SPI) provides guarantees with high probability that the trained policy performs better than the behavioural policy, also called baseline in this setting. Previous work shows that the SPI objective improves mean performance as compared to using the basic RL objective, which boils down to solving the MDP with maximum likelihood (Laroche et al. 2019). Here, we build on that work and improve more precisely the SPI with Baseline Bootstrapping algorithm (SPIBB) by allowing the policy search over a wider set of policies. Instead of binarily classifying the state-action pairs into two sets (the *uncertain* and the *safe-to-train-on* ones), we adopt a softer strategy that controls the error in the value estimates by constraining the policy change according to the local model uncertainty. The method can take more risks on uncertain actions all the while remaining provably-safe, and is therefore less conservative than the state-of-the-art methods. We propose two algorithms (one optimal and one approximate) to solve this constrained optimization problem and empirically show a significant improvement over existing SPI algorithms both on finite MDPS and on infinite MDPs with a neural network function approximation.

1 Introduction

In sequential decision-making problems, a common goal is to find a good policy using a limited number of trajectories generated by another policy, usually called the behavioral policy. This approach, also known as Batch Reinforcement Learning (Lange et al. 2012), is motivated by the many real-world applications that

K. Nadjahi and R. Laroche—Equal contribution.
K. Nadjahi—Work done while interning at Microsoft Research Montréal.
Finite MDPs code available at https://github.com/RomainLaroche/SPIBB.
SPIBB-DQN code available at https://github.com/rems75/SPIBB-DQN.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-46133-1_4) contains supplementary material, which is available to authorized users.

U. Brefeld et al. (Eds.): ECML PKDD 2019, LNAI 11908, pp. 53–68, 2020.
https://doi.org/10.1007/978-3-030-46133-1_4

naturally fit a setting where data collection and optimization are decoupled (contrary to online learning which integrates the two): *e.g.* dialogue systems (Singh et al. 1999), technical process control (Ernst et al. 2005; Riedmiller 2005), medical applications (Guez et al. 2008).

While most reinforcement learning techniques aim at finding a high-performance policy (Sutton and Barto 1998), the final policy does not necessarily perform well once it is deployed. In this paper, we focus on Safe Policy Improvement (SPI, Thomas 2015; Petrik et al. 2016), where the goal is to train a policy on a batch of data and guarantee with high probability that it performs at least as well as the behavioural policy, called baseline in this SPI setting. The safety guarantee is crucial in real-world applications where bad decisions may lead to harmful consequences.

Among the existing SPI algorithms, a recent computationally efficient and provably-safe methodology is SPI with Baseline Bootstrapping (SPIBB, Laroche et al. 2019; Simão and Spaan 2019). Its principle consists in building the set of state-action pairs that are only encountered a few times in the dataset. This set is called the bootstrapped set. The algorithm then reproduces the baseline policy for all pairs in that set and trains greedily on the rest. It therefore assumes access to the baseline policy, which is a common assumption in the SPI literature (Petrik et al. 2016). Other SPI algorithms use as reference the baseline performance, which is assumed to be known instead (Thomas 2015; Petrik et al. 2016). We believe that the known policy assumption is both more informative and more common, since most Batch RL settings involve datasets that were collected using a previous system based on a previous algorithm (*e.g.* dialogue, robotics, pharmaceutical treatment). While the empirical results show that SPIBB is safe and performs significantly better than the existing algorithms, it remains limited by the binary classification of the bootstrapped set: a pair either belongs to it, and the policy cannot be changed, or it does not, and the policy can be changed entirely.

Our contribution is a reformulation of the SPIBB objective that allows slight policy changes for uncertain state-action pairs while remaining safe. Instead of binarily classifying the state-action pairs into two sets, the uncertain and the safe-to-train-on ones, we adopt a strategy that extends the policy search to soft policy changes, which are constrained by an error bound related to the model uncertainty. The method is allowed to take more risks than SPIBB on uncertain actions, and still has theoretical safety guarantees under some assumptions. As a consequence, the safety constraint is softer: we coin this new SPI methodology *Safe Policy Improvement with Soft Baseline Bootstrapping* (Soft-SPIBB). We develop two algorithms to tackle the Soft-SPIBB problem. The first one solves it exactly, but is computationally expensive. The second one provides an approximate solution but is much more efficient computation-wise. We empirically evaluate the performance and safety of our algorithms on a gridworld task and analyze the reasons behind their significant advantages over the competing Batch RL algorithms. We further demonstrate the tractability of the approach by designing a DQN algorithm enforcing the Soft-SPIBB constrained policy optimization. The empirical results, obtained on a navigation task, show that Soft-SPIBB safely improves the baseline, and again outperforms all competing algorithms.

2 Background

2.1 Markov Decision Processes

We consider problems in which the agent interacts with an environment modeled as a *Markov Decision Process* (MDP): $M^* = \langle \mathcal{X}, \mathcal{A}, P^*, R^*, \gamma \rangle$, where $\mathcal{X}$ is the set of states, $\mathcal{A}$ the set of actions, P^* the unknown transition probability function, R^* the unknown stochastic reward function bounded by $\pm R_{max}$, and $\gamma \in [0, 1)$ the discount factor for future rewards. The goal is to find a policy $\pi : \mathcal{X} \to \Delta_{\mathcal{A}}$, with $\Delta_{\mathcal{A}}$ the set of probability distributions over the set of actions $\mathcal{A}$, that maximizes the expected return of trajectories $\rho(\pi, M^*) = V^{\pi}_{M^*}(x_0) = \mathbb{E}_{\pi, M^*}\left[\sum_{t \geq 0} \gamma^t R^*(x_t, a_t)\right]$. x_0 is the initial state of the environment and $V^{\pi}_{M^*}(x)$ is the value of being in state x when following policy π in MDP M^*. We denote by Π the set of stochastic policies. Similarly to $V^{\pi}_{M^*}(x)$, $Q^{\pi}_{M^*}(x, a)$ denotes the value of taking action a in state x. $A^{\pi}_M(x, a) = Q^{\pi}_M(x, a) - V^{\pi}_M(x)$ quantifies the advantage (or disadvantage) of action a in state x.

Given a dataset of transitions $\mathcal{D} = \langle x_j, a_j, r_j, x'_j \rangle_{j \in [\![1, |\mathcal{D}|]\!]}$, we denote the state-action pair counts by $N_{\mathcal{D}}(x, a)$, and its Maximum Likelihood Estimator (MLE) MDP by $\widehat{M} = \langle \mathcal{X}, \mathcal{A}, \widehat{P}, \widehat{R}, \gamma \rangle$, with:

$$\widehat{P}(x'|x, a) = \frac{\sum_{\langle x_j = x, a_j = a, r_j, x'_j = x' \rangle \in \mathcal{D}} 1}{N_{\mathcal{D}}(x, a)} \quad \text{and} \quad \widehat{R}(x, a) = \frac{\sum_{\langle x_j = x, a_j = a, r_j, x'_j \rangle \in \mathcal{D}} r_j}{N_{\mathcal{D}}(x, a)}.$$

The difference between an estimated parameter and the true one can be bounded using classic concentration bounds applied to the state-action counts in $\mathcal{D}$ (Petrik et al. 2016; Laroche et al. 2019): for all state-action pairs (x, a), we know with probability at least $1 - \delta$ that,

$$||P^*(\cdot|x, a) - \widehat{P}(\cdot|x, a)||_1 \leq e_P(x, a), \ |R^*(x, a) - \widehat{R}(x, a)| \leq e_P(x, a) R_{max}, \quad (1)$$

$$\left|Q^{\pi_b}_{M^*}(x, a) - Q^{\pi_b}_{\widehat{M}}(x, a)\right| \leq e_Q(x, a) V_{max}, \quad (2)$$

where $V_{max} \leq \dfrac{R_{max}}{1 - \gamma}$ is the maximum of the value function, and the two error functions may be derived from Hoeffding's inequality (see A.2) as

$$e_P(x, a) := \sqrt{\frac{2}{N_{\mathcal{D}}(x, a)} \log \frac{2|\mathcal{X}||\mathcal{A}|2^{|\mathcal{X}|}}{\delta}} \quad \text{and} \quad e_Q(x, a) := \sqrt{\frac{2}{N_{\mathcal{D}}(x, a)} \log \frac{2|\mathcal{X}||\mathcal{A}|}{\delta}}.$$

We will also use the following definition:

Definition 1. *A policy π is said to be a policy improvement over a baseline policy π_b in an MDP $M = \langle \mathcal{X}, \mathcal{A}, P, R, \gamma \rangle$ if the following inequality holds in every state $x \in \mathcal{X}$:*

$$V^{\pi}_M(x) \geq V^{\pi_b}_M(x) \quad (3)$$

2.2 Safe Policy Improvement with Baseline Bootstrappping

Our objective is to maximize the expected return of the target policy under the constraint of improving with high probability $1-\delta$ the baseline policy. This is known to be an NP-hard problem (Petrik et al. 2016) and some approximations are required to make it tractable. This paper builds on the Safe Policy Improvement with Baseline Bootstrapping methodology (SPIBB, Laroche et al. 2019). SPIBB finds an approximate solution to the problem by searching for a policy maximizing the expected return in the MLE MDP $\widehat{M}$, under the constraint that the policy improvement is guaranteed in the set of plausible MDPs Ξ:

$$\underset{\pi}{\text{argmax}}\, \rho(\pi, \widehat{M}), \text{ s.t. } \forall M \in \Xi, \rho(\pi, M) \geq \rho(\pi_b, M) - \zeta \tag{4}$$

$$\Xi = \left\{ M = \langle \mathcal{X}, \mathcal{A}, R, P, \gamma \rangle \text{ s.t. } \forall x, a, \begin{array}{l} ||P(\cdot|x,a) - \widehat{P}(\cdot|x,a)||_1 \leq e_P(x,a), \\ |R(x,a) - \widehat{R}(x,a)| \leq e_P(x,a) R_{max} \end{array} \right\} \tag{5}$$

The error function e_P is such that the true MDP M^* has a high probability of at least $1-\delta$ to belong to Ξ (Iyengar 2005; Nilim and El Ghaoui 2005). In other terms, the objective is to optimize the target performance in $\widehat{M}$ such that its performance is ζ-approximately at least as good as π_b in the admissible MDP set, where ζ is a precision hyper-parameter. Expressed this way, the problem is still intractable. SPIBB is able to find an approximate solution within a tractable amount of time by applying a special processing to state-action pair transitions that were not sampled enough in the batch of data. The methodology consists in building a set of rare thus uncertain state-action pairs in the dataset $\mathcal{D}$, called the bootstrapped set and denoted by $\mathcal{B}$: the bootstrapped set contains all the state-action pairs $(x, a) \in \mathcal{X} \times \mathcal{A}$ whose counts in $\mathcal{D}$ are lower than a hyper-parameter $N_\wedge$. SPIBB algorithms then construct a space of allowed policies, *i.e* policies that are constrained on the bootstrapped set $\mathcal{B}$, and search for the optimal policy in this set by performing policy iteration. For example, Π_b-SPIBB is a provably-safe algorithm that assigns the baseline π_b to the state-action pairs in $\mathcal{B}$ and trains the policy on the rest. $\Pi_{\leq b}$-SPIBB is a variant that does not give more weight than π_b to the uncertain transitions.

SPIBB's principle amounts to search over a policy space constrained such that the policy improvement may be precisely assessed in M^*. Because of the hard definition of the bootstrapped set, SPIBB relies on a binary decision-making and may be too conservative. Our novel method, called Soft-SPIBB, follows the same principle, but relaxes this definition by allowing soft policy changes for the uncertain state-action pairs, and offers more flexibility than SPIBB while remaining safe.

This idea might seem similar to Conservative Policy Iteration (CPI), Trust Region Policy Optimization (TRPO), or Proximal Policy Optimization (PPO) in that it allows changes in the policy under a proximity regularization to the old policy (Kakade and Langford 2002; Schulman et al. 2015, 2017). However, with Soft-SPIBB, the proximity constraint is tightened or relaxed according to the amount of samples supporting the policy change (see Definition 2). Additionally,

CPI, TRPO, and PPO are designed for the online setting. In the batch setting we consider, they would be either too conservative if the proximity regularization is applied with respect to the fixed baseline, or would converge to the fixed point obtained when solving the MLE MDP if the proximity regularization is moving with the last policy update (Corollary 3 of Geist et al. 2019).

2.3 Linear Programming

Linear programming aims at optimizing a linear objective function under a set of linear in-equality constraints. The most common methods for solving such linear programs are the simplex algorithm and interior point methods (IPMs, Dantzig 1963). Even though the worst-case computational complexity of the simplex is exponential in the dimensions of the program being solved (Klee and Minty 1972), this algorithm is efficient in practice: the number of iterations seems polynomial, and sometimes linear in the problem size (Borgwardt 1987; Dantzig and Thapa 2003). Nowadays, these two classes of methods continue to compete with one another: it is hard to predict the winner on a particular class of problems (Gondzio 2012). For instance, the hyper-sparsity of the problem generally seems to favour the simplex algorithm, while IPMs can be much more efficient for large-scale linear programming.

3 Safe Policy Improvement with Soft Baseline Bootstrapping

SPIBB allows to make changes in state-action pairs where the model error does not exceed some threshold ϵ, which may be expressed as a function of $N_\wedge$. This may be seen as a hard condition on the bootstrapping mechanism: a state-action pair policy may either be changed totally, or not at all. In this paper, we propose a softer mechanism where, for a given error function, a local error budget is allocated for policy changes in each state x. Similarly to SPIBB, we search for the optimal policy in the MDP model $\widehat{M}$ estimated from the dataset $\mathcal{D}$, but we reformulate the constraint by using Definitions 2 and 3.

Definition 2. *A policy π is said to be (π_b, e, ϵ)-constrained with respect to a baseline policy π_b, an error function e, and a hyper-parameter ϵ if, for all states $x \in \mathcal{X}$, the following inequality holds:*

$$\sum_{a \in \mathcal{A}} e(x,a)\big|\pi(a|x) - \pi_b(a|x)\big| \leq \epsilon.$$

Definition 3. *A policy π is said to be π_b-advantageous in an MDP $M = \langle \mathcal{X}, \mathcal{A}, P, R, \gamma \rangle$ if the following inequality holds in every state $x \in \mathcal{X}$:*

$$\sum_{a \in \mathcal{A}} A_M^{\pi_b}(x,a)\pi(a|x) \geq 0 \tag{6}$$

Remark 1. By the policy improvement theorem, a π_b-advantageous policy is a policy improvement over π_b. The converse is not guaranteed.

3.1 Theoretical Safe Policy Improvement Bounds

We show that constraining π_b-advantageous policies appropriately allows safe policy improvements. Due to space limitation, all proofs have been moved to the appendix, Section A.

Theorem 1. *Any (π_b, e_Q, ϵ)-constrained policy π that is π_b-advantageous in $\widehat{M}$ satisfies the following inequality in every state x with probability at least $1-\delta$:*

$$V^{\pi}_{M^*}(x) - V^{\pi_b}_{M^*}(x) \geq -\frac{\epsilon V_{max}}{1-\gamma}. \tag{7}$$

Constraining the target policy to be advantageous over the baseline is a strong constraint that leads to conservative solutions. To the best of our findings, it is not possible to prove a more general bound on (π_b, e_Q, ϵ)-constrained policy improvements. However, the search over (π_b, e_P, ϵ)-constrained policies, where e_P is an error bound over the probability function P (Eq. 2), allows us to guarantee safety bounds under Assumption 1, which states:

Assumption 1. *There exists a constant $\kappa < \frac{1}{\gamma}$ such that, for all state-action pairs $(x,a) \in \mathcal{X} \times \mathcal{A}$, the following inequality holds:*

$$\sum_{x',a'} e_P(x',a')\pi_b(a'|x')P^*(x'|x,a) \leq \kappa e_P(x,a). \tag{8}$$

Lemma 1, which is essential to prove Theorem 2 below, relies on Assumption 1.

Lemma 1. *Under Assumption 1, any (π_b, e_P, ϵ)-constrained policy π satisfies the following inequality for every state-action pair (x,a) with probability at least $1-\delta$:*

$$\left|Q^{\pi}_{M^*}(x,a) - Q^{\pi}_{\widehat{M}}(x,a)\right| \leq \left(\frac{e_P(x,a)}{1-\kappa\gamma} + \frac{\gamma\epsilon}{(1-\gamma)(1-\kappa\gamma)}\right) V_{max}.$$

Theorem 2. *Under Assumption 1, any (π_b, e_P, ϵ)-constrained policy π satisfies the following inequality in every state x with probability at least $1-\delta$:*

$$V^{\pi}_{M^*}(x) - V^{\pi_b}_{M^*}(x) \geq V^{\pi}_{\widehat{M}}(x) - V^{\pi_b}_{\widehat{M}}(x) - 2\left\|d^{\pi_b}_{M^*}(\cdot|x) - d^{\pi_b}_{\widehat{M}}(\cdot|x)\right\|_1 V_{max} - \frac{1+\gamma}{(1-\gamma)^2(1-\kappa\gamma)}\,\epsilon V_{max}. \tag{9}$$

Remark 2. The theorems hold for any error function e_P verifying 2 w.p. $1-\delta$.

Remark 3. Π_b-SPIBB (Laroche et al. 2019) is a particular case of Soft-SPIBB where the error function $e_P(x,a)$ equals ∞ if $(x,a) \in \mathcal{B}$ and $\frac{\epsilon}{2}$ otherwise.

Remark 4. Theorem 2 has a cubic dependency in the horizon $\frac{1}{1-\gamma}$, which is weaker than SPIBB's bounds, but allow us to safely search over more policies, when using tighter error functions. We will observe in Sect. 4 that Soft-SPIBB empirically outperforms SPIBB both in mean performance and in safety.

3.2 Algorithms

In this section, we design two safe policy improvement algorithms to tackle the problem defined by the Soft-SPIBB approach. They both rely on the standard policy iteration process described in Pseudo-code 1, where the policy improvement step consists in solving in every state $x \in \mathcal{X}$ the locally constrained optimization problem below:

$$\pi^{(i+1)}(\cdot|x) = \underset{\pi \in \Pi}{\operatorname{argmax}} \sum_{a \in \mathcal{A}} Q^{(i)}_{\widehat{M}}(x,a)\pi(a|x) \tag{10}$$

subject to:

Constraint 1: π being a probability: $\sum_{a \in \mathcal{A}} \pi(a|x) = 1$ and $\forall a$, $\pi(a|x) \geq 0$.

Constraint 2: π being (π_b, e, ϵ)-constrained.

Pseudo-code 1: Policy iteration process for Soft-SPIBB

Input: Baseline policy π_b, MDP model precision level ϵ and dataset $\mathcal{D}$.
Compute the model error concentration bounds $e(x,a)$.
Initialize $i = 0$ and $\pi^{(0)}(\cdot|x) = \pi_b(\cdot|x)$.
while *policy iteration stopping criterion not met* **do**
  Policy evaluation: compute $Q^{(i)}_{\widehat{M}}$ with dynamic programming.
  Policy improvement: set $\pi^{(i+1)}(\cdot|x)$ as the (exact or approximate) solution of the optimization problem defined in Equation 10.
  $i \leftarrow i + 1$
return $\pi^{(i)}$

Exact-Soft-SPIBB: The Exact-Soft-SPIBB algorithm computes the exact solution of the local optimization problem in (10) during the policy improvement step. For that, we express the problem as a Linear Program (LP) and solve it by applying the simplex algorithm. Note that we chose the simplex over IPMs as it turned out to be efficient enough for our experimental settings. For tractability in large action spaces, we reformulate the non-linear Constraint 2 as follows: we introduce $|\mathcal{A}|$ auxiliary variables $\{z(x,a)\}_{(x,a) \in \mathcal{X} \times \mathcal{A}}$, which bound from above each element of the sum. For a given $x \in \mathcal{X}$, Constraint 2 is then replaced by the following $2|\mathcal{A}| + 1$ linear constraints:

$$\forall a \in \mathcal{A}, \qquad \pi(a|x) - \pi_b(a|x) \leq z(x,a), \tag{11}$$

$$\forall a \in \mathcal{A}, \qquad -\pi(a|x) + \pi_b(a|x) \leq z(x,a), \tag{12}$$

$$\sum_a e(x,a)z(x,a) \leq \epsilon. \tag{13}$$

Approx-Soft-SPIBB: We also propose a computationally-efficient algorithm, which returns a sub-optimal target policy $\pi_{\tilde{\odot}}$. It relies on the same policy iteration, but computes an approximate solution to the optimization problem. The approach still guarantees to improve the baseline in $\widehat{M}$: $\rho(\pi_{\tilde{\odot}}, \widehat{M}) \geq \rho(\pi_b, \widehat{M})$, and falls under the Theorems 1 and 2 SPI bounds. Approx-Soft-SPIBB's local policy improvement step consists in removing, for each state x, the policy probability mass m^- from the action a^- with the lowest Q-value. Then, m^- is attributed to the action that offers the highest Q-value improvement by unit of error $\partial\epsilon$:

$$a^+ = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \frac{\partial \pi(a|x)}{\partial \epsilon} \left(Q^{(i)}_{\widehat{M}}(x,a) - Q^{(i)}_{\widehat{M}}(x,a^-) \right) \tag{14}$$

$$= \underset{a \in \mathcal{A}}{\operatorname{argmax}} \frac{Q^{(i)}_{\widehat{M}}(x,a) - Q^{(i)}_{\widehat{M}}(x,a^-)}{e(x,a)} \tag{15}$$

Once m^- has been reassigned to another action with higher value, the budget is updated accordingly to the error that has been spent, and the algorithm continues with the next worst action until a stopping criteria is met: the budget is fully spent, or $a^- = a^*$, where a^* is the action with maximal state-action value. The policy improvement step of Approx-Soft-SPIBB is further formalized in Pseudo-code 2, found in the appendix, Section A.8.

Theorem 3. *The policy improvement step of Approx-Soft-SPIBB generates policies that are guaranteed to be* (π_b, e, ϵ)*-constrained.*

Remark 5. The argmax operator in the result returned by Pseudo-code 2 is a convergence condition. Indeed, the approximate algorithm does not guarantee that the current iteration policy search space includes the previous iteration policy, which can cause divergence: the algorithm may indefinitely cycle between two or more policies. To ensure convergence, we update $\pi^{(i)}$ with $\pi^{(i+1)}$ only if there is a local policy improvement, *i.e.* when $\mathbb{E}_{a\sim\pi^{(i+1)}(\cdot|x)}[Q^{(i)}_{\widehat{M}}(x,a)] \geq \mathbb{E}_{a\sim\pi^{(i)}(\cdot|x)}[Q^{(i)}_{\widehat{M}}(x,a)]$.

Both implementation of the Soft-SPIBB strategy comply to the requirements of Theorem 1 if only one policy iteration is performed. In Sect. 4.1, we empirically evaluate the 1-iteration versions, which are denoted by the '1-step' suffix.

Complexity Analysis: We study the computational complexity of Exact-Soft-SPIBB and Approx-Soft-SPIBB. The error bounds computation and the policy evaluation step are common to both algorithms, and have a complexity of $\mathcal{O}(|\mathcal{D}|)$ and $\mathcal{O}(|\mathcal{X}|^3|\mathcal{A}|^3)$ respectively. The part that differs between them is the policy improvement.

Exact-Soft-SPIBB solves the LP with the simplex algorithm, which, as recalled in Sect. 2.3, is in practice polynomial in the dimensions of the program being solved. In our case, the number of constraints is $3|\mathcal{A}| + 1$.

Theorem 4. *Approx-Soft-SPIBB policy improvement has a complexity of* $\mathcal{O}(|\mathcal{X}||\mathcal{A}|^2)$.

Model-Free Soft-SPIBB: The Soft-SPIBB fixed point may be found in a model-free manner by fitting the Q-function to the target $y^{(i+1)}$ on the transition samples $\mathcal{D} = \langle x_j, a_j, r_j, x'_j \rangle_{j \in [\![1,N]\!]}$:

$$y_j^{(i+1)} = r_j + \gamma \sum_{a' \in \mathcal{A}} \pi^{(i+1)}(a'|x'_j) Q^{(i)}(x'_j, a'), \tag{16}$$

where $\pi^{(i+1)}$ is obtained either exactly or approximately with the policy improvement steps described in Sect. 3.2. Then, the policy evaluation consists in fitting $Q^{(i+1)}(x, a)$ to the set of $y_j^{(i+1)}$ values computed using the samples from $\mathcal{D}$.

Theorem 5. *Considering an MDP with exact counts, the model-based policy iteration of (Exact or Approx)-Soft-SPIBB is identical to the model-free policy iteration of (resp. Exact or Approx)-Soft-SPIBB.*

The model-free versions are less computationally efficient than their respective model-based versions, but are particularly useful since it makes function approximation easily applicable. In our infinite MDP experiment, we consider Approx-Soft-SPIBB-DQN as the DQN algorithm fitted to the model-free Approx-Soft-SPIBB targets. The Exact-Soft-SPIBB counterpart is not considered for tractability reasons. We recall that the computation of the policy improvement step relies on the estimates of an error function e_P, which may, for instance, be indirectly inferred from pseudo-counts $\widetilde{N}_{\mathcal{D}}(x, a)$ (Bellemare et al. 2016; Fox et al. 2018; Burda et al. 2019).

4 Soft-SPIBB Empirical Evaluation

This section intends to empirically validate the advances granted by Soft-SPIBB. We perform the study on two domains: on randomly generated finite MDPs, where the Soft-SPIBB algorithms are compared to several Batch RL competitors: basic RL, High Confidence Policy Improvement (Thomas 2015, HCPI), Reward-Adjusted MDPs (Petrik et al. 2016, RaMDP), Robust MDPs (Iyengar, 2005; Nilim and El Ghaoui 2005), and to Soft-SPIBB natural parents: Π_b-SPIBB and $\Pi_{\leq b}$-SPIBB (Laroche et al. 2019); and on a helicopter navigation task requiring function approximation, where Soft-SPIBB-DQN is compared to basic DQN, RaMDP-DQN, and SPIBB-DQN. All the benchmark algorithms had their hyper-parameters optimized beforehand. Their descriptions and the results of the hyper-parameter search is available in the appendix, Section B.2 for finite MDPs algorithms and Section C.3 for DQN-based algorithms.

In order to assess the safety of an algorithm, we run a large number of times the same experiment with a different random seed. Since the environments and the baselines are stochastic, every experiment generates a different dataset, and the algorithms are evaluated on their mean performance over the experiments, and on their conditional value at risk performance (CVaR), sometimes also called the expected shortfall: $X\%$-CVaR corresponds to the mean performance over the $X\%$ worst runs.

4.1 Random MDPs

In the random MDPs experiment, the MDP and the baseline are themselves randomly generated too. The full experimental process is formalized in Pseudo-code 2 found in the appendix, Section B.1. Because every run involves different MDP and baseline, there is the requirement for a normalized performance. This is further defined as $\overline{\rho}$:

$$\overline{\rho}(\pi, M^*) = \frac{\rho(\pi, M^*) - \rho(\pi_b, M^*)}{\rho(\pi^*, M^*) - \rho(\pi_b, M^*)}. \tag{17}$$

In order to demonstrate that Soft-SPIBB algorithms are safely improving the baselines on most MDPs in practice, we use a random generator of MDPs. All the details may be found in the appendix, Section B.1. The number of states is set to $|\mathcal{X}| = 50$, the number of actions to $|\mathcal{A}| = 4$ and the connectivity of the transition function to 4, *i.e.*, for a given state-action pair (x, a), its transition function $P(x'|x, a)$ is non-zero on four states x' only. The reward function is 0 everywhere except when entering the goal state, which is terminal and where the reward is equal to 1. The goal is chosen in such a way that the optimal value function is minimal.

Random Baseline: For a randomly generated MDP M, baselines are generated according to a predefined level of performance $\eta \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$: $\rho(\pi_b, M) = \eta\rho(\pi^*, M) + (1 - \eta)\rho(\tilde{\pi}, M)$, where π^* and $\tilde{\pi}$ are respectively the optimal and the uniform policies. The generation of the baseline consists in three steps: optimization, where the optimal Q-value is computed; softening, where a softmax policy is generated; and randomization, where the probability mass is randomly displaced in the baseline. The process is formally and extensively detailed in the appendix, Section B.1.

Dataset Generation: Given a fixed size number of trajectories, a dataset is generated on the following modification of the original MDPs: addition of another goal state (reward is set to 1). Since the original goal state was selected so as to be the hardest to reach, the new one, which is selected uniformly, is necessarily a better goal.

Complexity Empirical Analysis: In Fig. 1, we show an empirical confirmation of the complexity results on the gridworld task. Exact-Soft-SPIBB has a linear dependency in the number of actions. We also notice that Approx-Soft-SPIBB runs much faster: even faster than Π_b-SPIBB, and 2 times slower than basic RL. Note that the policy improvement step is by design exactly linearly dependent on the number of states $|\mathcal{X}|$. This is the reason why

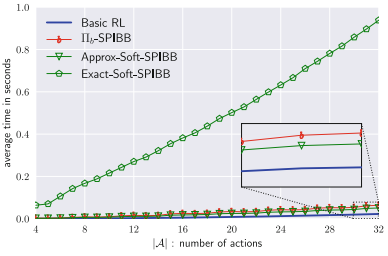


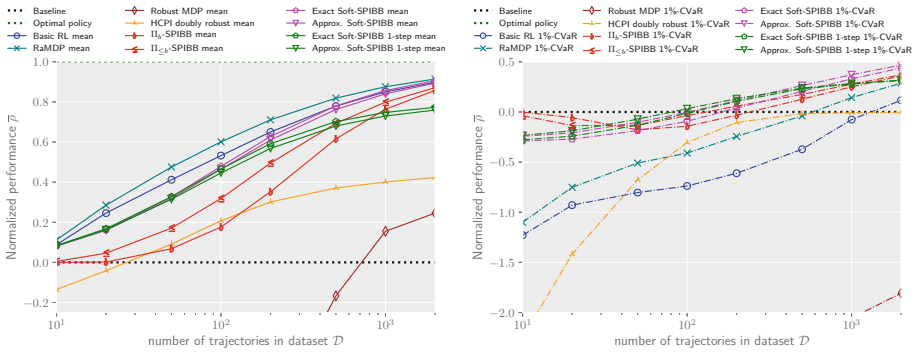
Fig. 1. Average time to convergence.

we do not report experiments on the dependency on $|\mathcal{X}|$. We do not report complexity empirical analysis of the other competitors because we do not pretend to have optimal implementations of them, and the purpose of this analysis is to show that Approx-Soft-SPIBB solves the tractability issues of Exact-Soft-SPIBB. In Theory, Robust MDPs and HCPI are more complex by at least an order of magnitude.

Benchmark Results: Figures 2a and b respectively report the mean and 1%-CVAR performances with a strong baseline ($\eta = 0.9$). Robust MDPs and HCPI perform poorly and are not further discussed. Basic RL and RaMDP win the benchmark in mean, but fail to do it safely, contrary to Soft-SPIBB and SPIBB algorithms. Exact-Soft-SPIBB is slightly better than Approx-Soft-SPIBB in mean, but also slightly worse in safety. Still in comparison to Approx-Soft-SPIBB, Exact-Soft-SPIBB's performance does not justify the computational complexity increase and will not be further discussed. Approx-Soft-SPIBB demonstrates a significant improvement over SPIBB methods, both in mean and in safety. Finally, the comparison of Approx-Soft-SPIBB with Approx-Soft-SPIBB 1-step shows that the safety is not improved in practice, and that the asymptotic optimality is compromised when the dataset becomes larger.

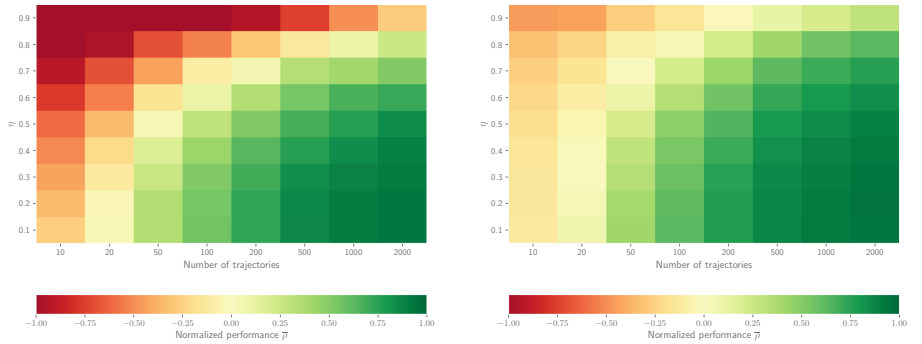
Sensitivity to the Baseline: We continue the analysis with a heatmap representation as a function of the strength of the baseline: Figs. 3a and b display heatmaps of the 0.1%-CVaR performance for RaMDP and Approx-Soft-SPIBB ($\epsilon = 2$) respectively. The colour of a cell indicates the improvement 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 that RaMDP is unsafe for strong baselines (high η values) and small datasets, while Soft-SPIBB methods become slightly unsafe only with $\eta = 0.9$ and less than 20 trajectories, but are safe everywhere else.

Sensitivity to Hyper-Parameters: We carry on with 1%-CVaR performance heatmaps as a function of the hyper-parameters for RaMDP (Fig. 4a) and Approx-Soft-SPIBB (Fig. 4b) in the hardest scenario ($\eta = 0.9$). The choice of 1%-CVaR instead of 0.1%-CVaR is justified by the fact that the 0.1%-CVaR RaMDP heatmap is almost completely red, which would not allow us to notice the interesting thresholding behaviour: when $\kappa_{adj} \geq 0.0035$, RaMDP becomes over-conservative to the point of not trying to reach the goal anymore. In contrast, Approx-Soft-SPIBB behaves more smoothly with respect to its hyper-parameter, its optimal value being in interval [0.5, 2], depending on the safety/performance trade-off one wants to achieve. In the appendix, Section B.3, the interested reader may find the corresponding heatmaps for all Soft-SPIBB algorithms for mean and 1%-CVaR performances. In particular, we may observe that, despite not having as strong theoretical guarantees as their 1-step versions, the Soft-SPIBB algorithms demonstrate similar CVaR performances.



(a) Mean: $\eta = 0.9$, $\epsilon = 2$ (b) 1%-CVAR: $\eta = 0.9$, $\epsilon = 2$

Fig. 2. Benchmark on Random MDPs domain: mean and 1%-CVAR performances for a hard scenario ($\eta = 0.9$) and Soft-SPIBB with $\epsilon = 2$

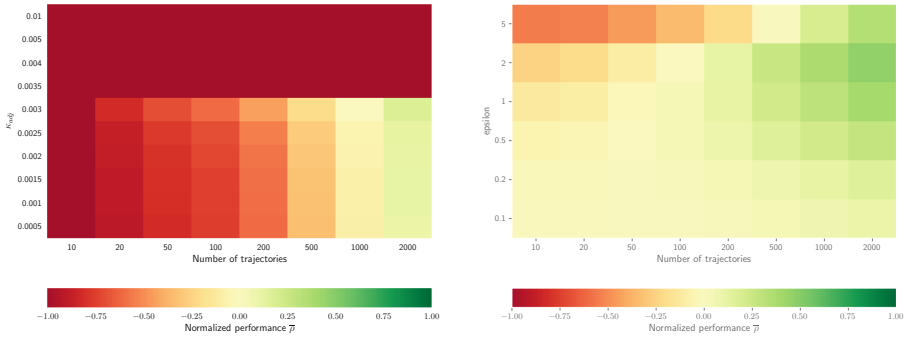


(a) 0.1%-CVaR: RaMDP (b) 0.1%-CVaR: Approx-Soft-SPIBB, $\epsilon = 2$

Fig. 3. Influence of η on Random MDPs domain: 0.1%-CVaR heatmaps as a function of η

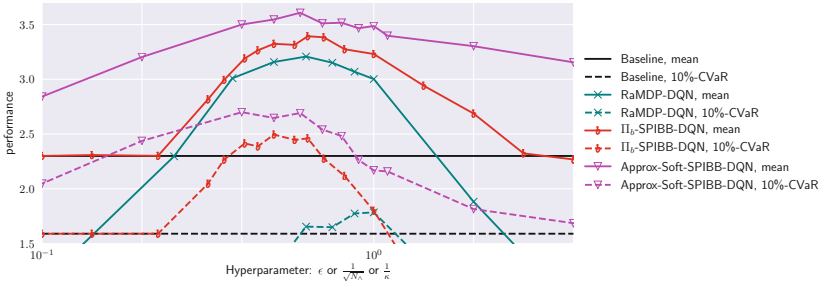
4.2 Helicopter Domain

To assess our algorithms on tasks with more complex state spaces, making the use of function approximation inevitable, we apply them to a helicopter navigation task (Fig. 5(c)). The helicopter's start point is randomly picked in the teal region, its initial velocity is random as well. The agent can choose to apply or not a fixed amount of thrust forward and backward in the two dimensions, resulting in 9 actions total. An episode ends when the agent reaches the boundary of the blue box or has a speed larger than some maximal value. In the first case, it receives a reward based on its position with respect to the top right corner of the box (the exact reward value is chromatically indicated in the figure). In the second, it gets a reward of -1. The dynamics of the helicopter obey Newton's second law

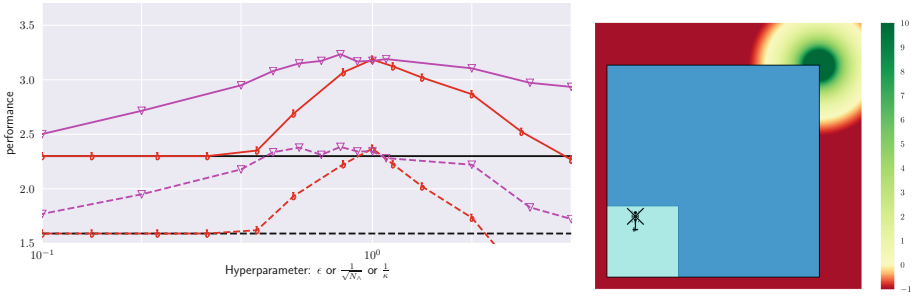


(a) 1%-CVaR: RaMDP, $\eta = 0.9$ (b) 1%-CVaR: Approx-Soft-SPIBB, $\eta = 0.9$

Fig. 4. Sensitivity to hyperparameter on Random MDPs: 1%-CVaR heatmaps for $\eta = 0.9$



(a) Helicopter benchmark with $|\mathcal{D}| = 10,000$



(b) Helicopter benchmark with $|\mathcal{D}| = 3,000$ (c) Helicopter domain

Fig. 5. Helicopter: mean and 10%-CVaR as a function of the hyper-parameter value

with an additive centered Gaussian noise applied to its position and velocity. We refer the reader to the appendix, Section C.1 for the detailed specifications. We generated a baseline by training online a DQN (Mnih et al. 2015) and applying a softmax on the learnt Q-network. During training, a discount factor of 0.9 is

used, but the reported results show the undiscounted return obtained by the agent.

The experiments consist in 300 training runs (necessary to obtain reasonable estimates of algorithms' safety, the full training procedure is described in the appendix, Section C.3) of RaMDP-DQN, SPIBB-DQN and Approx-Soft-SPIBB-DQN, for different values of their hyper-parameters (resp. κ, $N_\wedge$ and ϵ). We note that for $\kappa = 0$, $N_\wedge = 0$ or $\epsilon = +\infty$, those three algorithms become standard DQN, and that for $N_\wedge = \infty$ or $\epsilon = 0$, the SPIBB and Soft-SPIBB algorithms produce a copy of the baseline. The three algorithms rely on some estimates of the state-action counts. In this work, we used a pseudo-count estimate heuristic based on Euclidean distance, also detailed in Section C.3. For scalability, we may consider several pseudo-count methodologies from the literature Bellemare et al. (2016); Fox et al. (2018). This is left for future work.

The results of our evaluation can be found in Fig. 5, where we plot the mean and 10%-CVaR performances of the different algorithms for two sizes of datasets (more results may be found in the appendix, Section C.4). In order to provide meaningful comparisons, the abscissa represents the different hyper-parameters transformed to account for their dimensional homogeneity (except for a scaling factor). Both Approx-Soft-SPIBB-DQN and SPIBB-DQN outperform RaMDP-DQN by a large margin on the datasets of size 10,000. On the smaller datasets, RaMDP-DQN performs very poorly and does not even appear on the graph. For the same reason, vanilla DQN (mean: 0.22 and 10%-CVaR: -1 with $|\mathcal{D}| = 10,000$) does not appear on any of the graphs. The two SPIBB algorithms significantly outperform the baseline both in mean and 10%-CVaR. At their best hyper-parameter value, their 10%-CVaR is actually better than the mean performance of the baseline. Approx-Soft-SPIBB-DQN performs better than SPIBB-DQN both in mean and 10%-CVaR performances. Finally, it is less sensitive than SPIBB-DQN with respect to their respective hyperparameters, and demonstrates a better stability over different dataset sizes. That stability is a useful property as it reduces the requirement for hyper-parameter optimization, which is crucial for Batch RL.

5 Conclusion

We study the problem of safe policy improvement in a Batch RL setting. Building on the SPIBB methodology, we relax the constraints of the policy search to propose a family of algorithms coined Soft-SPIBB. We provide proofs of safety and of computational efficiency for an algorithm called Approx-Soft-SPIBB based on the search of an approximate solution that does not compromise the safety guarantees. We support the theoretical work with an extensive empirical analysis where Approx-Soft-SPIBB shines as the best compromise average performance vs. safety. We further develop Soft-SPIBB in a model-free manner which helps its application to function approximation. Despite the lack of theoretical safety guarantees with function approximation, we observe in our experiments where the function approximation is modelled as a neural network, that Soft-SPIBB

allows safe policy improvement in practice and significantly outperforms the competing algorithms both in safety and in performance.

References

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

Borgwardt, K.H.: The Simplex Method: A Probabilistic Analysis. Springer, Heidelberg (1987). https://doi.org/10.1007/978-3-642-61578-8

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

Dantzig, G.: Linear Programming and Extensions. Rand Corporation Research Study. Princeton Univ. Press, Princeton (1963)

Dantzig, G.B., Thapa, M.N.: Linear Programming 2: Theory and Extensions. Springer, New York (2003). https://doi.org/10.1007/b97283

Ernst, D., Geurts, P., Wehenkel, L.: Tree-based batch mode reinforcement learning. J. Mach. Learn. Res. **6**, 503–556 (2005)

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

Geist, M., Scherrer, B., Pietquin, O.: A theory of regularized Markov decision processes. In: Proceedings of the 36th International Conference on Machine Learning (ICML) (2019)

Gondzio, J.: Interior point methods 25 years later. Eur. J. Oper. Res. **218**(3), 587–601 (2012)

Guez, A., Vincent, R.D., Avoli, M., Pineau, J.: Adaptive treatment of epilepsy via batch-mode reinforcement learning. In: Proceedings of the 23rd AAAI Conference on Artificial Intelligence, pp. 1671–1678 (2008)

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

Iyengar, G.N.: Robust dynamic programming. Math. Oper. Res. **30**(2), 257–280 (2005)

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

Klee, V., Minty, G.J.: How good is the simplex algorithm? In: Shisha, O. (ed.) Inequalities, vol. III, pp. 159–175. Academic Press, New York (1972)

Lange, S., Gabel, T., Riedmiller, M.: Batch reinforcement learning. In: Wiering, M., van Otterlo, M. (eds.) Reinforcement Learning. Adaptation, Learning, and Optimization, vol. 12, pp. 45–73. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27645-3_2

Laroche, R., Trichelair, P., Tachet des Combes, R.: Safe policy improvement with baseline bootstrapping. In: Proceedings of the 36th International Conference on Machine Learning (ICML) (2019)

Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)

Nilim, A., El Ghaoui, L.: Robust control of Markov decision processes with uncertain transition matrices. Oper. Res. **53**(5), 780–798 (2005)

Paszke, A., et al.: Automatic differentiation in PyTorch. In: NIPS-W (2017)
Petrik, M., Ghavamzadeh, M., Chow, Y.: Safe policy improvement by minimizing robust baseline regret. In: Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS) (2016)
Riedmiller, M.: Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720, pp. 317–328. Springer, Heidelberg (2005). https://doi.org/10.1007/11564096_32
Schulman, J., Levine, S., Abbeel, P., Jordan, M., 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., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
Simão, T.D., Spaan, M.T.J.: Safe policy improvement with baseline bootstrapping in factored environments. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence (2019)
Singh, S.P., Kearns, M.J., Litman, D.J., Walker, M.A.: Reinforcement learning for spoken dialogue systems. In: Proceedings of the 13th Advances in Neural Information Processing Systems (NIPS), pp. 956–962 (1999)
Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge (1998)
Thomas, P.S.: Safe reinforcement learning. Ph.D. thesis, Stanford university (2015)
Tieleman, T., Hinton, G.: Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. COURSERA: Neural Netw. Mach. Learn. **4**(2), 26–31 (2012)
van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. CoRR, abs/1509.06461 (2015)
Weissman, T., Ordentlich, E., Seroussi, G., Verdu, S., Weinberger, M.J.: Inequalities for the L1 deviation of the empirical distribution. Hewlett-Packard Labs, Technical report (2003)