Safe Policy Improvement with Baseline Bootstrapping

Editor:

Abstract

In this paper, we consider the Batch Reinforcement Learning task and adopt the safe policy improvement (SPI) approach: we compute a target policy guaranteed to perform at least as well as a given baseline policy, approximately and with high probability. Our SPI strategy, inspired by the knows-what-it-knows paradigm, consists in bootstrapping the target with the baseline when the target does not know. We develop a policy-based computationally efficient bootstrapping algorithm, accompanied by theoretical SPI bounds for the tabular case. We empirically show the limits of the existing algorithms on a small stochastic gridworld problem, and then demonstrate that our algorithm not only improve the worst-case scenario but also the mean performance.

1. Introduction

Reinforcement Learning (RL, Sutton and Barto (1998)) consists in discovering through trial-and-error in an unknown uncertain environment, which action is the most valuable in a particular situation. This *optimism in the face of uncertainty* (OFU, Szita and Lőrincz (2008)), or at the very least, this recklessness in an online learning setting: algorithms based on OFU, such as R-MAX or UCRL, are provably efficient. Indeed, a good outcome brings a policy improvement, and even an error leads to learning not to do it again at a lesser cost (Brafman and Tennenholtz, 2002; Auer and Ortner, 2007). However, most real-world algorithms are to be widely deployed on independent devices/systems, and as such their policies cannot be updated as often as online learning would require. Therefore, the same mistake may be repeated during a time long enough to lose the user's trust. In this offline setting, batch RL algorithms are one approach that has been recommended (Lange et al., 2012). But, the OFU paradigm shows its limits when the policy updates are rare, because the commitment on the optimism is too strong and the error impact may be severe. In this paper, we endeavour to build batch algorithms that are safe in this regard.

The concept of safety in RL has been defined in several contexts (Garcıa and Fernández, 2015). Two notions of uncertainty, the internal and the parametric, are defined in (Ghavamzadeh et al., 2015): *internal uncertainty reflects the uncertainty of the return due to stochastic transitions and rewards, for a single known MDP, while parametric uncertainty reflects the uncertainty about the unknown MDP parameters: the transition and reward distributions*. In short, internal safety guarantees a certain level of return for each individual trajectory, which is critical for potential harmful behaviour or catastrophe avoidance scenarios.

In this paper, we focus more specifically on parametric safety: guarantee of a given expected return for the trained policy in the batch RL setting (Thomas et al., 2015; Petrik et al., 2016). More specifically, we aim to train a policy, called *target*, which approximately outperforms the behavioral policy, called *baseline*, with high confidence. The goal is therefore to improve the baseline, even in

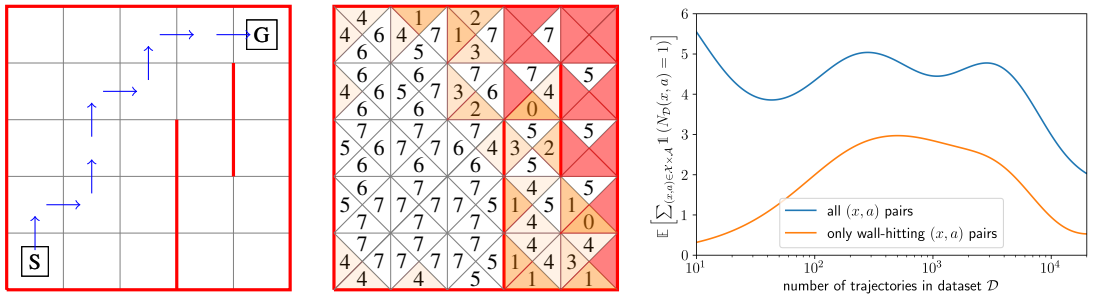
the unfavorable scenarios. As such, this family of algorithms can be seen as *pessimistic in the face of uncertainty*, the flip side of OFU.

Our contributions are summarized as follows. Firstly, we identify a small batch RL task where existing algorithms fail to be safe (Section 2). We analyze the reasons for this failure. Secondly, we present a novel methodology for Safe Policy Improvement (SPI) called SPI with Baseline Bootstrapping (SPIBB). It consists in bootstrapping the trained policy with the baseline in the state-action pair transitions that were not probed enough in the dataset (Section 3). Thirdly, we develop a novel provably-safe and computationally efficient algorithm, and two algorithm variants (Section 4). Fourthly, we empirically analyze the safety and performances of our algorithms on the gridworld task. The results show that our algorithms significantly outperform the competitors, both in mean and the worst-percentile performance (Section 5). Finally, Section 6 concludes the paper.

2. Illustrative scenario: Gridworld

The stochastic gridworld task is motivated by the fact that existing batch RL algorithms already fail to be safe in this simple environment. This task is also small enough to empirically assess the worst-percentile performance: the mean performance over the $X\%$ worst runs. The task is illustrated on Figure 1a and we refer the interested reader to Appendix A for the full experimental setting. We introduce the MDP framework and our notations in Appendix B.

Transition distribution imbalance: All the state-of-the-art algorithms for batch RL assume that every state-action has been experienced a certain amount of times (Delage and Mannor, 2010; Petrik et al., 2016). In this paragraph, we aim to empirically demonstrate that this assumption is generally transgressed even in our small gridworld domain. To do so, we collect 12 million trajectories with the baseline. The map of the state-action pairs count $\log_{10}$ logarithm (see Figure 1b) shows how unbalanced they are: some pairs are experienced in each trajectory, some once every million trajectories, and some are never seen. Moreover, the actions that are rarely chosen are likely to be the dangerous ones, and for those ones, a bad model might lead to a catastrophic policy. Figure 1c displays the expected number of transitions that are seen exactly once in a dataset as a function of its size (it is similar to a multimodal Poisson point process). This curve is computed analytically from the state-action counts in the 12 million trajectory dataset. It decreases slowly as more trajectories



(a) Stochastic gridworld domain and its optimal trajectory.

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

(c) Expectation of the number of (x, a) pairs counting a unique transition.

Figure 1: Gridworld domain, state-action counts and unique transitions count expectation.

are collected. But, if we look more specifically at dangerous transitions, *i.e.* the ones that direct the agent to a wall, we observe a peak around 1,000 trajectories.

Failure of existing algorithms: In Figure 2, we see that, surprisingly, the Basic RL policies trained with 10 trajectories yield better returns than the ones trained with 1,000 trajectories on average. We interpret it as the consequence of the rare pair count effect developed in the previous paragraph, that create situations where the basic RL trains catastrophic policies. We conjecture that this issue is faced at different scales in most practical applications. Neither Robust MDP nor Reward Adjusted MDP Petrik et al. (2016) seem to improve the safety when the safety test is omitted. We did so in our curves to make a relevant comparison: this test always fails because of its wide confidence interval. Reward Adjusted MDP is overwhelmed by the penalty from performing actions that were rarely taken in the dataset, and ignores the environment rewards. As a result, it converges to remaining in the safe zone in the middle of the grid where no positive reward may be received, hence it is not visible on Figure 2.

Figure 2: Existing algorithms benchmark: basic RL, Robust MDP, and Reward Adjusted MDP are compared to our $\Pi_{\leq b}$-SPIBB on mean and worst-decile performances.

We solve this issue by allowing to optimize only on the state-action pairs that have been sufficiently sampled in the dataset to infer tangible knowledge. We propose a provably-safe way of bootstrapping the uncertain state-action pairs with low variance estimators obtained from the baseline. The SPIBB algorithms are described and analyzed in Section 4. Their comparative empirical results are lengthily discussed in Section 5. But first, Section 3 introduces the SPIBB methodology.

3. Safe Policy Improvement with Baseline Bootstrapping

Percentile criterion and Robust MDPs: We adapt here the *percentile criterion* Delage and Mannor (2010) to the objective of safe policy improvement over the baseline policy π_b:

$$\pi_C = \underset{\pi \in \Pi}{\operatorname{argmax}}\, \mathbb{E}\left[\rho(\pi, M) \,|\, M \sim \mathbb{P}_{\text{MDP}}(\cdot|\mathcal{D})\right], \tag{1}$$

$$\text{such that } \mathbb{P}\left(\rho(\pi, M) \geq \rho(\pi_b, M) - \zeta \,|\, M \sim \mathbb{P}_{\text{MDP}}(\cdot|\mathcal{D})\right) \geq 1 - \delta, \tag{2}$$

where $\mathbb{P}_{\text{MDP}}(\cdot|\mathcal{D})$ is the posterior probability of the MDP parameters, $1-\delta$ and ζ are the high probability and error meta-parameters. Petrik et al. (2016) use Robust MDP (Iyengar, 2005; Nilim and El Ghaoui, 2005) to bound from below the constraint (2) by considering $\Xi = \Xi(\widehat{M}, e)$ as a set of admissible MDPs:

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

where $\widehat{M} = \langle \mathcal{X}, \mathcal{A}, \widehat{P}, \widehat{R}, \gamma \rangle$ is the MDP parameters estimator, and $e : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is an error function depending on the dataset $\mathcal{D}$ and the high probability meta-parameter $1-\delta$. Instead 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. Thus, Petrik et al. (2016) contemplate the policy improvement worst-case scenario: $\pi_S = \text{argmax}_{\pi\in\Pi} \min_{M\in\Xi} (\rho(\pi, M) - \rho(\pi_b, M))$. Unfortunately, they prove that this optimization is an NP-hard problem. They propose an algorithm approximating the solution: Approximate Robust Baseline Regret Minimization (ARBRM). There are three problems with ARBRM. First, ARBRM assumes that there is no error in the transition probabilities of the baseline, which is a hazardous assumption. Second, considering its high complexity (polynomial time), it is difficult to empirically assess its percentile criterion safety even in simple tasks. Third, the Robust MDP solver uses a conservative worst-case safety test.

SPIBB methodology: As evoked earlier, we endeavour to further reformulate the percentile criterion in order to be able to search for an efficient and provably-safe policy within a tractable amount of computer time. Our new criterion consists in optimizing the policy with respect to its performance in the MDP estimate $\widehat{M}$, while being guaranteed to be ζ-approximately at least as good as π_b in the admissible MDP set Ξ. More formally, we write it as follows:

$$\max_{\pi\in\Pi} \rho(\pi, \widehat{M}), \text{ such that } \forall M \in \Xi, \rho(\pi, M) \geq \rho(\pi_b, M) - \zeta \tag{3}$$

From Theorem 8 of Petrik et al. (2016), it is immediate to prove that, if all state-action pair counts satisfy $N_{\mathcal{D}}(x, a) \geq \frac{8V_{max}^2}{\zeta^2(1-\gamma)^2} \log \frac{2|\mathcal{X}||\mathcal{A}|2^{|\mathcal{X}|}}{\delta}$, and if $\widehat{M}$ is the Maximum Likelihood Estimation (MLE) MDP, then $\pi^{\odot} = \text{argmax}_{\pi\in\Pi} \rho(\pi, \widehat{M})$ implies that $\rho(\pi^{\odot}, M) \geq \rho(\pi^*, M) - \zeta \geq \rho(\pi_b, M) - \zeta$.

In this paper, we extend this result by allowing this constraint to be only partially satisfied in a subset of $\mathcal{X} \times \mathcal{A}$. Its complementary subset, the set of uncertain state-action pairs, is called the bootstrapped set and is denoted by $\mathfrak{B}$ in the following. $\mathfrak{B}$ is dependent on the dataset $\mathcal{D}$ and on a hyper-parameter: minimal count $N_\wedge$. For ease of notation those dependencies are omitted. Algorithm 1 in Appendix C formalizes the construction of $\mathfrak{B}$.

4. Policy-based SPIBB

We adopt a policy bootstrapping. More precisely, when a state-action pair (x, a) is rarely seen in the dataset, *i.e.* $(x, a) \in \mathfrak{B}$, instead of relying on a noisy value estimate, we propose to rely on the baseline policy by copying the probability to take this action in this state: $\pi(a|x) = \pi_b(a|x)$ if $(x, a) \in \mathfrak{B}$. Algorithm 2 in Appendix C, referred as Π_b-SPIBB, provides the pseudo-code of the baseline bootstrapping. It consists in constructing the set of allowed policies Π_b and then searching the Π_b-optimal policy $\pi_{pol}^{\odot}$ in the MDP model $\widehat{M}$ estimated from dataset $\mathcal{D}$. In practice, the optimisation process may be performed by policy iteration (Howard, 1966; Puterman and Brumelle, 1979): the current policy $\pi^{(i)}$ is evaluated with $Q^{(i)}$, and then the next iteration policy $\pi^{(i+1)}$ is made greedy with respect to $Q^{(i)}$ under the constraint of belonging to Π_b.

Theorem 1 states that Π_b-SPIBB converges to a Π_b-optimal policy $\pi_{pol}^{\odot}$. Below, Theorem 2 states that $\pi_{pol}^{\odot}$ safely improves the baseline in the true environment.

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

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

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

Π_b-SPIBB has a tendency to reproduce the rare actions from the baseline. Even though this is what allows us to guarantee a performance almost as good as the baseline, it may prove to be problematic when the baseline is already near optimal for two reasons: first, the low visited state-action pairs are generally the actions for which the behavioral policy probability is lower, meaning that these actions are more likely to be bad; second, the baseline exploratory strategies fall into this category, and copying the baseline is reproducing these strategies.

Another way to look at the problem is therefore to consider that rare actions must be avoided, because they are risky, and therefore to force the policy to assign a probability of 0 to perform this action[1]. Algorithm 2 remains unchanged with the policy search space Π_0 instead of Π_b:

$$\Pi_0 = \{\pi \in \Pi \,|\, \pi(a|x) = 0 \text{ if } (x, a) \in \mathfrak{B}\} \tag{5}$$

Theorem 1 applies to this variant, referred as Π_0-SPIBB, but the empirical analysis of Section 5 reports that it sometimes proves to be unsafe. We believe that a better policy-improvement SPIBB lies in-between: the space of policies to search in should be constrained not to give more weight than π_b to actions that were not tried out enough to significantly assess their performance, but still leave the possibility to completely cut off bad performing actions even though this evaluation is uncertain. The resulting algorithm is referred as $\Pi_{\leq b}$-SPIBB. Again, Theorem 1 applies to $\Pi_{\leq b}$-SPIBB and Algorithm 2 is reused by replacing the policy search space Π_b with $\Pi_{\leq b}$ defined as follows:

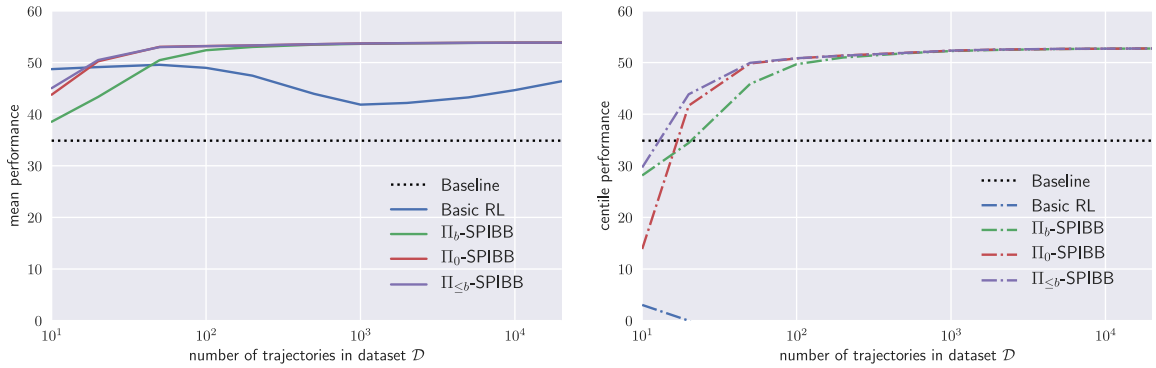
$$\Pi_{\leq b} = \{\pi \in \Pi \,|\, 0 \leq \pi(a|x) \leq \pi_b(a|x) \text{ if } (x, a) \in \mathfrak{B}\} \tag{6}$$

5. SPIBB empirical evaluation and benchmark

The SPIBB algorithms are evaluated and compared on the simple gridworld task illustrated on Figure 1a. We repeated 25000+ runs for each 5 SPIBB algorithms, each 11 dataset sizes[2], and each 8 $N_\wedge$ values logarithmically ranging from 5 to 1000[3]. Another baseline is used to generate the dataset and to bootstrap on. It differs in that it favours walking along the walls, although it should avoid it to prevent bad stochastic transitions. This baseline was constructed in order to demonstrate the jeopardy of algorithm Π_0-SPIBB when the optimal policy is different from the action with the highest likelihood in the baseline.

The safety is strictly assessed by a worst-centile measure: average performance of the 1% worst runs. The basic RL worst-centile curve is too low to appear. The main lessons are that the safety of improvement over the baseline is not much impacted by the choice of $N_\wedge$, but that a higher $N_\wedge$ implies the SPIBB algorithms to be more conservative and to bootstrap more often on the baseline.

1. If, in a given state x, all action are boostrapped, then $\pi^{\odot}_{pol}(x, \cdot)$ is set to $\pi_b(x, \cdot)$.
2. Our SPIBB algorithms are so efficient and safe that we expand the dataset size range to [10,20000].
3. Since safety bounds can be loose, we use $N_\wedge$ directly as a hyper-parameter instead of calculating it from ζ and δ.



(a) Mean performance.

(b) Worst-centile performance

Figure 3: SPIBB benchmark ($N_\wedge = 5$): mean and worst-centile performance

Even though the theory would advise to use higher values, our best empirical results was obtained with $N_\wedge = 5$. Due to space limitations, we only illustrate on figures the $N_\wedge = 5$ results.

Π_b-SPIBB and $\Pi_{\leq b}$-SPIBB get a worst-centile scenario only 5 points below the baseline, which is explained by the variance in the evaluation, and is not a consequence of a policy worse than the baseline. Π_0-SPIBB lacks safety with very small datasets, because it tends to completely abandon actions that are not sampled enough in the dataset, regardless of their performance. Results with higher $N_\wedge$ values show that there is a dataset size for which Π_0-SPIBB tends to cut the optimal actions (of not walking along the wall), which causes a strong performance drop, both in worst-centile scenario and in mean performance (see Figure 3a). Π_b-SPIBB is more conservative and fails to improve as fast as the two other policy-based SPIBB algorithms, but it does it safely. $\Pi_{\leq b}$-SPIBB is the best of both worlds: safe although still capable of cutting bad actions even with only a small number of samples. However, for multi-batch settings, it is better to keep on trying out the actions that were not sufficiently explored yet, and Π_b-SPIBB might be the best algorithm in this setting.

6. Conclusion and future work

In this paper, we tackle the problem of Batch Reinforcement Learning and its safety. We reformulate the percentile criterion without compromising its safety. We lose optimality but keep a PAC-style guarantee of policy improvement. The gain is that it allows to implement an algorithm Π_b-SPIBB that runs as fast as a basic model-based RL algorithm, while generating a provably safe policy improvement over a known baseline π_b. The empirical analysis shows that, even on a very simple domain, the basic RL algorithm fails to be safe, and the state-of-the-art safe batch RL algorithms almost never accept a policy change. And they do no better than the basic RL algorithm when the policy improvement safety test is omitted. The SPIBB algorithms show significantly better results: their worst-centile performance even surpassing the basic RL mean performance.

Future work includes developing model-free versions of our algorithms in order to ease their use in continuous state MDPs and complex real-world applications, with state representation approximation, using density networks (Veness et al., 2012; Van Den Oord et al., 2016) to compute pseudo-counts (Bellemare et al., 2016), in a similar way to that of optimism-motivated online RL (Osband et al., 2016; Laroche and Barlier, 2017; Ostrovski et al., 2017).

References

Peter Auer and Ronald Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Proceedings of the 21st Advances in Neural Information Processing Systems (NIPS)*, 2007.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS)*, 2016.

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

Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 2002.

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

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

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 2015.

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

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

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

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

Romain Laroche and Merwan Barlier. Transfer reinforcement learning with shared dynamics. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017.

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

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS)*, 2016.

Georg Ostrovski, Marc G Bellemare, Aaron van den Oord, and Rémi Munos. Count-based exploration with neural density models. *arXiv preprint arXiv:1703.01310*, 2017.

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

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

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

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

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

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

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

Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML)*, 2016.

Joel Veness, Kee Siong Ng, Marcus Hutter, and Michael Bowling. Context tree switching. In *Proceedings of the 22nd Data Compression Conference (DCC)*, 2012.

Appendix A. Stochastic gridworld task and experimental setting

Our case study is a straightforward discrete, stochastic 5×5 gridworld (see Figure 1a). We use four actions: up, down, left and right. The transition function is stochastic and the actions move the agent in the specified direction with 75% chance, in the opposite direction with 5% chance and sideway with 10% chance each. The initial and final states are respectively the bottom left and top right corners. The reward is -10 when hitting a wall (in which case the agent does not move) and $+100$ if the final state is reached. Each run consists in generating a dataset, training policies with the algorithms, and evaluating the trained policies. The results are presented in two forms: the mean performance on all the runs and the worst-decile performance: average performance of the 10% worst runs. Each point is the result of 25,000+ independent runs.

Basic RL stands for computing the Maximum Likelihood Estimation MDP of the environment, and solving it with dynamic programming. In order to cover the state-action pairs absent from the dataset, several Q initializations were investigated: optimistic (V_{max}), null (0), and pessimistic ($-V_{max}$). The first yielded awful performances, and the two last yielded the same performances (indeed the Q-function is positive everywhere in our task). All the presented results are obtained with the null initialization. The baseline is obtained with a softmax exploration around the optimal Q-function. Two safe batch RL algorithms are added to the benchmark: Robust MDP and Reward Adjusted MDP (Petrik et al., 2016).

Appendix B. The MDP framework

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

The state value function $V_M^\pi(x)$ (resp. state-action value function $Q_M^\pi(x,a)$) is the expectation of the discounted sum of rewards when following $\pi \in \Pi$ and starting from state $x \in \mathcal{X}$ (resp. performing action $a \in \mathcal{A}$ in state $x \in \mathcal{X}$) in the MDP $M = \langle \mathcal{X}, \mathcal{A}, R, P, \gamma \rangle$. The goal of a reinforcement learning algorithm is to discover the unique optimal state value function V_M^* (resp. action-state value function Q_M^*). We define the performance of a policy by its expected value $\rho(\pi, M) = V_M^\pi(x_0)$. Given a policy subset $\Pi' \subseteq \Pi$, a policy π' is said to be Π'-optimal for an MDP M when it maximises its performance: $\rho(\pi', M) = \max_{\pi \in \Pi'} \rho(\pi, M)$. Later, we also make use of the notation V_{max} as a known upper bound of the return absolute value: $V_{max} \leq \frac{R_{max}}{1-\gamma}$. Given a dataset of transitions $\mathcal{D}$, we denote the state-action pair counts by $N_{\mathcal{D}}(x,a)$.

Appendix C. Algorithms

Algorithm 1: Construction of $\mathfrak{B}$

Input: Dataset $\mathcal{D}$
Input: Parameter $N_\wedge$

Initialize $\mathfrak{B}$: $\mathfrak{B} = \emptyset$.
for $(x, a) \in \mathcal{X} \times \mathcal{A}$ **do**
  if $N_{\mathcal{D}}(x, a) < N_\wedge$ **then**
    $\mathfrak{B} = \mathfrak{B} \cup \{(x, a)\}$.
  end
end
return $\mathfrak{B}$

Algorithm 2: Π_b-SPIBB algorithm

Input: Dataset $\mathcal{D}$, bootstrapped set $\mathfrak{B}$
Input: Baseline policy π_b

Compute $\widehat{M} = \langle \mathcal{X}, \mathcal{A}, \widehat{P}, \widehat{R}, \gamma \rangle$.
$\Pi_b = \{\pi \in \Pi \,|\, \pi(a|x) = \pi_b(a|x) \text{ if } (x, a) \in \mathfrak{B}\}$
return $\pi^{\odot}_{pol} = \underset{\pi \in \Pi_b}{\operatorname{argmax}}\, \rho(\pi, \widehat{M})$

Appendix D. Proofs for Π_b-SPIBB (Section 4)

Proposition 1 *Consider an environment modelled with a semi-MDP Parr and Russell (1998); Sutton et al. (1999)* $\ddot{M} = \langle \mathcal{X}, \Omega_{\mathcal{A}}, \ddot{P}^*, \ddot{R}^*, \Gamma^* \rangle$*, where* Γ^* *is the discount rate inferior or equal to* γ *that varies with the state action transitions and the empirical semi-MDP* $\widehat{\ddot{M}} = \langle \mathcal{X}, \Omega_{\mathcal{A}}, \widehat{\ddot{P}}, \widehat{\ddot{R}}, \widehat{\Gamma} \rangle$ *estimated from a dataset* $\mathcal{D}$*. If in every state* x *where option* o_a *may be initiated:* $x \in \mathcal{I}_a$*, we have:*

$$\sqrt{\frac{2}{N_{\mathcal{D}}(x,a)} \log \frac{2|\mathcal{X}||\mathcal{A}|2^{|\mathcal{X}|}}{\delta}} \leq \epsilon, \tag{7}$$

then, with probability at least $1 - \delta$*:*

$$\forall a \in \mathcal{A}, \forall x \in \mathcal{I}_a, \begin{cases} \|\Gamma^* \ddot{P}^*(x, o_a) - \widehat{\Gamma}\widehat{\ddot{P}}(x, o_a)\|_1 \leq \epsilon \\ |\ddot{R}^*(x, o_a) - \widehat{\ddot{R}}(x, o_a)| \leq \epsilon \ddot{R}_{max} \end{cases} \tag{8}$$

Proof The proof is similar to that of Proposition 9 of Petrik et al. (2016). ■

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

Lemma 1 (Q-function error bounds with Π_b-SPIBB) *Consider two semi-MDPs* $M_1 = \langle \mathcal{X}, \Omega_{\mathcal{A}}, P_1, R_1, \Gamma_1 \rangle$ *and* $M_2 = \langle \mathcal{X}, \Omega_{\mathcal{A}}, P_2, R_2, \Gamma_2 \rangle$*. Consider a policy* π*. Also, consider* Q_1 *and* Q_2 *be the state-action value function of the policy* π *in* M_1 *and* M_2*, respectively. If:*

$$\forall a \in \mathcal{A}, \forall x \in \mathcal{I}_a, \begin{cases} |R_1 \mathbb{1}_{x,o_a} - R_2 \mathbb{1}_{x,o_a}| \leq \epsilon R_{max} \\ ||(\Gamma_1 \circ P_1)\mathbb{1}_{x,o_a} - (\Gamma_2 \circ P_2)\mathbb{1}_{x,o_a}||_1 \leq \epsilon, \end{cases} \tag{9}$$

then, we have:

$$\forall a \in \mathcal{A}, \forall x \in \mathcal{I}_a, \quad |Q_1 \mathbb{1}_{x,o_a} - Q_2 \mathbb{1}_{x,o_a}| \leq \frac{2\epsilon V_{max}}{1-\gamma}, \tag{10}$$

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

Proof We adopt the matrix notations. The difference between the two state-option value functions can be written:

$$\begin{aligned} Q_1 - Q_2 &= R_1 + Q_1 \pi(\Gamma_1 \circ P_1) - R_2 - Q_2 \pi(\Gamma_2 \circ P_2) && (11) \\ &= R_1 + Q_1 \pi(\Gamma_1 \circ P_1) - R_2 - Q_2 \pi(\Gamma_2 \circ P_2) + Q_2 \pi(\Gamma_1 \circ P_1) - Q_2 \pi(\Gamma_1 \circ P_1) && (12) \\ &= R_1 - R_2 + (Q_1 - Q_2)\pi(\Gamma_1 \circ P_1) + Q_2 \pi((\Gamma_1 \circ P_1) - (\Gamma_2 \circ P_2)) && (13) \\ &= [R_1 - R_2 + Q_2 \pi((\Gamma_1 \circ P_1) - (\Gamma_2 \circ P_2))] \, (\mathbb{I} - \pi(\Gamma_1 \circ P_1))^{-1}. && (14) \end{aligned}$$

Now using the Holder's inequality and the second assumption, we have:

$$|Q_2 \pi((\Gamma_1 \circ P_1) - (\Gamma_2 \circ P_2))\mathbb{1}_{x,o_a}| \leq \|Q_2\|_\infty \|\pi\|_\infty \|(\Gamma_1 \circ P_1)\mathbb{1}_{x,o_a} - (\Gamma_2 \circ P_2)\mathbb{1}_{x,o_a}\|_1 \leq \epsilon V_{max}. \tag{15}$$

Inserting 15 into Equation 14 and using the first assumption, we obtain:

$$|Q_1 \mathbb{1}_{x,o_a} - Q_2 \mathbb{1}_{x,o_a}| \leq [\epsilon R_{max} + \epsilon V_{max}] \, \|(\mathbb{I} - \pi(\Gamma_1 \circ P_1))^{-1} \mathbb{1}_{x,o_a}\|_1 \tag{16}$$

$$\leq \frac{2\epsilon V_{max}}{1-\gamma}, \tag{17}$$

which proves the lemma. There is a factor 2 that might require some discussion. It comes from the fact that we do not control that the maximum R_{max} might be as big as V_{max} in the semi-MDP setting and we do not control the γ factor in front of the second term. As a consequence, we surmise that a tighter bound down to $\frac{\epsilon V_{max}}{1-\gamma}$ holds, but this still has to be proven. ■

D.2 Convergence and safe policy improvement of Π_b-SPIBB

Lemma 2 (Safe policy improvement of $\pi^{\odot}_{pol}$ over any policy $\pi \in \Pi_b$) *Let Π_b be the set of policies under the constraint of following π_b when $(x,a) \in \mathfrak{B}$. Let $\pi^{\odot}_{pol}$ be a Π_b-optimal policy of the reward maximization problem of an estimated MDP $\widehat{M}$. Then, for any policy $\pi \in \Pi_b$, the difference of performance between $\pi^{\odot}_{pol}$ and π is bounded as follows with high probability $1-\delta$ in the true MDP M^*:*

$$\rho(\pi^{\odot}_{pol}, M^*) - \rho(\pi, M^*) \geq \rho(\pi^{\odot}_{pol}, \widehat{M}) - \rho(\pi, \widehat{M}) - \frac{4\epsilon V_{max}}{1-\gamma}. \tag{18}$$

Proof We transform the true MDP M^* and the MDP estimate $\widehat{M}$, to their bootstrapped semi-MDP counterparts $\ddot{M}^*$ and the MDP estimate $\widehat{\ddot{M}}$. In these semi-MDPs, the actions $\mathcal{A}$ are replaced by options $\Omega_{\mathcal{A}} = \{o_a\}_{a \in \mathcal{A}}$ constructed as follows:

$$o_a = \langle \mathcal{I}_a, a{:}\pi_b, \beta \rangle = \begin{cases} \mathcal{I}_a = \{x \in \mathcal{X}, \text{ such that } (x,a) \notin \mathfrak{B}\} \\ a{:}\tilde{\pi}_b = \text{ perform } a \text{ at initialization, then follow } \tilde{\pi}_b \\ \beta(x) = \|\dot{\pi}_b(x, \cdot)\|_1 \end{cases} \tag{19}$$

where π_b has been decomposed as the aggregation of two partial policies as follows: any policy π may be decomposed as the aggregation of two partial policies: $\pi = \dot{\pi} + \tilde{\pi}$, where $\dot{\pi}$ are the non-boostrapped actions probabilities, and $\tilde{\pi}$ are the bootstrapped actions probabilities:

$$\forall a \in \mathcal{A}, \begin{cases} \dot{\pi}(x,a) = \pi(x,a) & \text{if } (x,a) \notin \mathfrak{B} \\ \dot{\pi}(x,a) = 0 & \text{if } (x,a) \in \mathfrak{B} \end{cases} \tag{20}$$

$$\forall a \in \mathcal{A}, \begin{cases} \tilde{\pi}(x,a) = \pi(x,a) & \text{if } (x,a) \in \mathfrak{B} \\ \tilde{\pi}(x,a) = 0 & \text{if } (x,a) \notin \mathfrak{B} \end{cases} \tag{21}$$

Let $\ddot{\Pi}$ denote the set of policies over the bootstrapped semi MDPs.

$\mathcal{I}_a$ is the initialization function: it determines the set of states where the option is available. $a{:}\pi_b$ is the option policy being followed during the length of the option. Finally, $\beta(x)$ is the termination function defining the probability of the option to terminate in each state[4]. Please, notice that some

4. Notice that all options have the same termination function.

states might have no available options, but this is okay since every option has a termination function equal to 0 in those states, meaning that they are unreachable. This to avoid being in this situation at the beginning of the trajectory, we use the notion of starting option: a trajectory starts with a void option $o_\emptyset = \langle \{x_0\}, \pi_b, \beta \rangle$.

By construction $x \in \mathcal{I}_a$ if and only if $(x, a) \notin \mathfrak{B}$, *i.e.* if and only if the condition on the state-action counts of Proposition 1 is fulfilled[5]. Also, any policy $\pi \in \Pi_b$ is implemented by a policy $\ddot{\pi} \in \ddot{\Pi}$ in a bootstrapped semi-MDP. Inversely, any policy $\ddot{\pi} \in \ddot{\Pi}$ admits a policy $\pi \in \Pi_b$ in the original MDP.

Note also, that by construction, the transition and reward functions are only defined for (x, o_a) pairs such that $x \in \mathcal{I}_a$. By convention, we set them to 0 for the other pairs. Their corresponding Q-functions are therefore set to 0 as well.

This means that Lemma 1 may be applied with $\pi = \pi^{\odot}_{pol}$ and $M_1 = \ddot{M}^*$ and $M_2 = \widehat{\ddot{M}}$. We have:

$$|\rho(\pi^{\odot}_{pol}, M^*) - \rho(\pi^{\odot}_{pol}, \widehat{M})| = |\rho(\pi^{\odot}_{pol}, \ddot{M}^*) - \rho(\pi^{\odot}_{pol}, \widehat{\ddot{M}})| \tag{22}$$

$$= |V^{\pi^{\odot}_{pol}}_{\ddot{M}^*}(x_0) - V^{\pi^{\odot}_{pol}}_{\widehat{\ddot{M}}}(x_0)| \tag{23}$$

$$= |Q^{\pi^{\odot}_{pol}}_{\ddot{M}^*}(x_0, o_\emptyset) - Q^{\pi^{\odot}_{pol}}_{\widehat{\ddot{M}}}(x_0, o_\emptyset)| \tag{24}$$

$$\leq \frac{2\epsilon V_{max}}{1-\gamma} \tag{25}$$

Analogously to 25, for any $\pi \in \Pi_b$, we also have:

$$|\rho(\pi, M^*) - \rho(\pi, \widehat{M})| \leq \frac{2\epsilon V_{max}}{1-\gamma} \tag{26}$$

Thus, we may write:

$$\rho(\pi^{\odot}_{pol}, M^*) - \rho(\pi, M^*) \geq \rho(\pi^{\odot}_{pol}, \widehat{M}) - \rho(\pi, \widehat{M}) - \frac{4\epsilon V_{max}}{1-\gamma}, \tag{27}$$

where the inequality is directly obtained from equations 25 and 26. ■

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

Proof We use the same transformation of $\widehat{M}$ as in Lemma 2. Then, the problem is cast without any constraint in a well defined semi-MDP, and Policy Iteration is known to converge in semi-MDPs to the policy optimizing the value function Gosavi (2004). ■

5. Also, note that there is the requirement here that the trajectories are generated under policy π_b, so that the options are consistent with the dataset.

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

$$\zeta = \frac{4V_{max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge}\log\frac{2|\mathcal{X}||\mathcal{A}|2^{|\mathcal{X}|}}{\delta}} - \rho(\pi^{\odot}_{pol}, \widehat{M}) + \rho(\pi_b, \widehat{M}) \tag{28}$$

Proof It is direct to observe that $\pi_b \in \Pi_b$, and therefore that Lemma 2 can be applied to π_b. We infer that, with high probability $1 - \delta$:

$$\rho(\pi^{\odot}_{pol}, M^*) - \rho(\pi_b, M^*) \geq \rho(\pi^{\odot}_{pol}, \widehat{M}) - \rho(\pi_b, \widehat{M}) - \frac{4\epsilon V_{max}}{1-\gamma}. \tag{29}$$

with:

$$\epsilon = \sqrt{\frac{2}{N_\wedge}\log\frac{2|\mathcal{X}||\mathcal{A}|2^{|\mathcal{X}|}}{\delta}} \tag{30}$$

Therefore, we obtain:

$$\zeta = \frac{4\epsilon V_{max}}{1-\gamma} - \Big(\rho(\pi^{\odot}_{pol}, \widehat{M}) - \rho(\pi_b, \widehat{M})\Big) \tag{31}$$

$$= \frac{4V_{max}}{1-\gamma}\sqrt{\frac{2}{N_\wedge}\log\frac{|\mathcal{X}||\mathcal{A}|2^{|\mathcal{X}|}}{\delta}} - \rho(\pi^{\odot}_{pol}, \widehat{M}) + \rho(\pi_b, \widehat{M}) \tag{32}$$

Quod erat demonstrandum. ■