Soft Safe Policy Improvement with Baseline Bootstrapping

Kimia Nadjahi KIMIANADJAHI@GMAIL.COM
Romain Laroche ROMAIN.LAROCHE@MICROSOFT.COM
Rémi Tachet des Combes REMI.TACHET@MICROSOFT.COM
Microsoft Research, Montréal, Canada

Abstract

Batch Reinforcement Learning is a common setting in sequential decision-making under uncertainty. It consists in finding an optimal policy using trajectories collected with another policy, called the baseline. Previous work shows that safe policy improvement (SPI) methods improve mean performance compared to the basic algorithm (Laroche and Trichelair, 2017). Here, we build on that work and improve the algorithm by allowing finer optimization under the safety constraint. 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 by considering locally the error due to the model uncertainty. The method takes the right amount of risk to try uncertain actions all the while remaining safe in practice, and therefore is less conservative than the state-of-the-art methods. We propose four algorithms for this constrained optimization problem and empirically show a significant improvement over existing SPI methods.

Keywords: safe policy improvement, batch reinforcement learning, model uncertainty.

1. Introduction

In sequential decision-making problems, a common goal is to find a good policy using a limited number of trajectories that were generated with another policy, called the *behavioral policy* or *baseline*. This approach, also known as *Batch Reinforcement Learning* (Batch RL, Lange et al. (2012)), is motivated by the many real-world applications that naturally fit this setting, where data collection and optimization are decoupled (contrary to online learning which integrates the two tasks): dialogue systems (Singh et al., 1999; El Asri et al., 2016), technical process control (Ernst et al., 2005; Riedmiller, 2005), and 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 et al. (2015); Petrik et al. (2016)), where the goal is to train a target policy that approximately performs at least as well as the baseline with high confidence from a batch of data. SPI thus learns a policy that outperforms the baseline with high probability, which makes it very useful in real-world applications where bad decisions may lead to harmful consequences.

Among the existing SPI algorithms, one recent computationally efficient and provably-safe methodology is *SPI with Baseline Bootstrapping* (SPIBB) (Laroche and Trichelair, 2017). The idea is to build a set of rare state-action pairs in the dataset, called the *bootstrapped set*, to copy the baseline policy for all pairs in this set and train greedily on the

©2018 .
License: CC-BY 4.0, see `https://creativecommons.org/licenses/by/4.0/`.

rest. While the empirical results show that SPIBB is safe and performs significantly better than the standard model-based approach, it strongly relies on the binary classification of the bootstrapped set: a pair is either in it and the policy cannot be changed, otherwise, the policy can be changed entirely.

As the main contribution of the paper, we propose a reformulation of the SPIBB objective that allows slight policy changes for uncertain state-action pairs while remaining empirically safe. In that sense, the safety constraint is softer, we thus coin this SPI methodology *Soft Safe Policy Improvement with Baseline Bootstrapping* (Soft-SPIBB). We develop four algorithms to compute a policy that is an approximate solution of the new objective. We empirically evaluate the performance and safety of our algorithms on a gridworld task and explain their significant advantages.

2. Background

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 a set of states, $\mathcal{A}$ a set of actions, P^* the unknown transition probability function, R^* the unknown stochastic bounded 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 $\rho(\pi, M^*) = V^{\pi}_{M^*}(x_0) = \mathbb{E}_{\pi, M^*}\left[\sum_{t \geq 0} \gamma^t R^*(x_t, a_t))\right]$, where x_0 is the initial state and $V^{\pi}_{M^*}(x)$ is the value of being in a 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. Given a dataset of transitions $\mathcal{D}$, we denote the state-action pair counts by $N_{\mathcal{D}}(x, a)$, and its Maximum Likelihood Estimator (MLE) MDP by $\widehat{M} = \langle \mathcal{X}, \mathcal{A}, \widehat{P}, \widehat{R}, \gamma \rangle$.

Safe Policy Improvement with Baseline Bootstrappping: The 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 a 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 and Trichelair (2017)). SPIBB approximates the problem by searching for a policy maximizing the expected return in the MLE MDP $\widehat{M}$ under the constraint of guaranteeing a policy improvement with high probability $1 - \delta$:

$$\pi^{\odot} = \underset{\pi \in \Pi}{\operatorname{argmax}}\, \rho(\pi, \widehat{M}), \quad \text{s.t. } \forall M \in \Xi(\widehat{M}, e_\delta), \;\; \rho(\pi, M) \geq \rho(\pi_b, M) - \zeta \tag{1}$$

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

where ζ is a precision hyperparameter and e_δ is an error function such that the true MDP M^* has a high probability of at least $1 - \delta$ to belong to $\Xi(\widehat{M}, e_\delta)$ (Iyengar, 2005; Nilim and El Ghaoui, 2005). The error function is classically bounded from concentration bounds over the state-action counts in the dataset $\mathcal{D}$ (Petrik et al., 2016):

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

In other terms, SPIBB optimizes the target performance in $\widehat{M}$ such that this performance is ζ-approximatively at least as good as π_b in the admissible MDP set. Expressed this way, the problem is untractable and the SPIBB methodology proposes a sub-optimal solution by building a set of 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 $(x,a) \in \mathcal{X} \times \mathcal{A}$ that were not sampled enough in $\mathcal{D}$, *i.e.* whose counts are lower than a hyperparameter $N_\wedge$. Then, the policy-based SPIBB approach constructs a set of allowed policies and searches the optimal policy in this space by performing policy iteration. Π_b-SPIBB is a provably-safe greedy algorithm that assigns π_b to the state-action pairs in the bootstrapped set. Our novel method, called Soft-SPIBB, relaxes the hard definition of the bootstrapping set and allows soft policy changes for the uncertain state-action pairs. It is shown to improve the empirical performance.

3. Soft Safe Policy Improvement with baseline Bootstrapping

We reformulate the SPIBB objective as follows:

$$\pi^\odot = \operatorname*{argmax}_{\pi \in \Pi} \rho(\pi, \widehat{M}), \quad \text{s.t. } \forall x \in \mathcal{X}, \sum_{a \in \mathcal{A}} |\pi(a|x) - \pi_b(a|x)| e'_\delta(x,a) \leq 2\epsilon, \tag{3}$$

where ϵ is a hyperparameter that may be interpreted as a local error budget, and $e'_\delta(x,a)$ is the error function on the baseline policy evaluation: $|Q^{\pi_b}_{M^*}(x,a) - Q^{\pi_b}_{\widehat{M}}(x,a)| \leq e'_\delta(x,a) V_{max}$ holds with high probability $1 - \frac{\delta}{|\mathcal{X}||\mathcal{A}|}$. In comparison with $e_\delta(x,a)$, concentration bounds are tighter when using the Monte Carlo estimator for $Q^{\pi_b}_{\widehat{M}}(x,a)$:

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

Π_b-SPIBB (Laroche and Trichelair, 2017) is the exact solution of Equation 3, when the error function is loosely bounded by $e'_\delta(x,a) \leq \infty$ if $(x,a) \in \mathcal{B}$ and $e'_\delta(x,a) \leq \epsilon$ otherwise.

In this paper, we propose to use the finer concentration bounds error function of Equation 4. However, finding $\pi^\odot$ requires optimization under constraint. In order to contain the computational complexity, we propose two algorithms that return sub-optimal target policies $\pi^\odot_\sim$. They both rely on policy iteration, where the policy improvement step is performed under the constraint defined in Equation 3. This approach guarantees to improve the baseline in $\widehat{M}$: $\rho(\pi^\odot_\sim, \widehat{M}) \geq \rho(\pi_b, \widehat{M})$.

Soft-SPIBB-max-Q: In the appendix, Algorithm 1 provides the pseudo-code of Soft-SPIBB-max-Q. The policy evaluation works as in any standard policy iteration algorithm: $Q^{(i)}_{\widehat{M}}$ is the evaluation of policy $\pi^{(i)}$ computed at iteration (i). However, the policy improvement must satisfy the Soft-SPIBB constraint. As a consequence, in each state, the Soft-SPIBB-max-Q algorithm computes the local error expense E by moving greedily all the policy probability mass to the action with maximal state-action value: $a^* = \text{argmax}_{a \in \mathcal{A}} Q^{(i)}_{\widehat{M}}(s,a)$.

$$E = (1 - \pi_b(a^*|x))e'_\delta(x,a^*) + \sum_{a \neq a^*} \pi_b(a|x)e'_\delta(x,a) \tag{5}$$

If E remains under the budget 2ϵ, then the greedy policy is allowed, else, the budget is consumed in proportion:

$$\pi^{(i+1)}(a^*|x) = \pi_b(a^*|x) + \tfrac{2\epsilon}{E}\left(1 - \pi_b(a^*|x)\right) \tag{6}$$

$$\forall a \neq a^*, \quad \pi^{(i+1)}(a|x) = \left(1 - \tfrac{2\epsilon}{E}\right)\pi_b(a|x) \tag{7}$$

Note that with this update, the local error expense is exactly 2ϵ. Soft-SPIBB-max-Q has the advantage of being simple, but also presents two drawbacks. First, if the maximal action has a large error, then the policy probability mass move remains small, even though the error budget could have been used more efficiently. Second, the suboptimal actions are all consumed equally and it takes only one with a large error to prevent the other policy probability mass moves. Soft-SPIBB-sort-Q solves these issues with a more sophisticated mass redistribution and without significant computational complexity increase.

Soft-SPIBB-sort-Q: In the appendix, Algorithm 2 provides the pseudo-code of Soft-SPIBB-sort-Q. Soft-SPIBB-sort-Q local policy improvement consists in removing 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 $\partial\epsilon$:

$$a^+ = \underset{a \in \mathcal{A}}{\text{argmax}} \frac{\partial \pi}{\partial \epsilon}(x,a)\left(Q^{(i)}_{\widehat{M}}(x,a) - Q^{(i)}_{\widehat{M}}(x,a^-)\right) \tag{8}$$

$$= \underset{a \in \mathcal{A}}{\text{argmax}} \frac{Q^{(i)}_{\widehat{M}}(x,a) - Q^{(i)}_{\widehat{M}}(x,a^-)}{e'_\delta(x,a)} \tag{9}$$

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 $a^- = a^*$.

Soft-SPIBB-max-Q-λe and Soft-SPIBB-sort-Q-λe: In Soft-SPIBB algorithms, a rare state-action pair with an overestimated state-action value can receive a higher probability value than the baseline, which might constitute a significant advantage over SPIBB when the optimal action is rarely taken by the baseline, but not when the rare actions are yielding bad returns. Generally, the baseline has a bias towards the good actions. We derive versions of Soft-SPIBB-max-Q and Soft-SPIBB-sort-Q, respectively called Soft-SPIBB-max-Q-λe and Soft-SPIBB-sort-Q-λe, that exploit this bias by considering a lower confidence bound on $Q^{(i)}_{\widehat{M}}(x,a)$. A proper lower confidence bound is hard to compute. Instead, we estimate it as $Q^{(i)}_{\widehat{M}}(x,a) - \lambda e'_\delta(x,a)$. λ is a hyperparameter that controls the desired amount of bias.

Convergence condition In our algorithms, there is no guarantee that the current iteration policy search space includes the previous iteration policy, which can cause divergence: the algorithm can indefinitely cycle between two policies. To ensure convergence, we decide to update $\pi^{(i)}$ with $\pi^{(i+1)}$ only if there is a local policy improvement, *i.e.* when $\mathbb{E}_{a\sim\pi^{(i+1)}(\cdot|x)}[Q^{(i)}_{\widehat{M}}(x,a)] \geq \mathbb{E}_{a\sim\pi^{(i)}(\cdot|x)}[Q^{(i)}_{\widehat{M}}(x,a)]$.

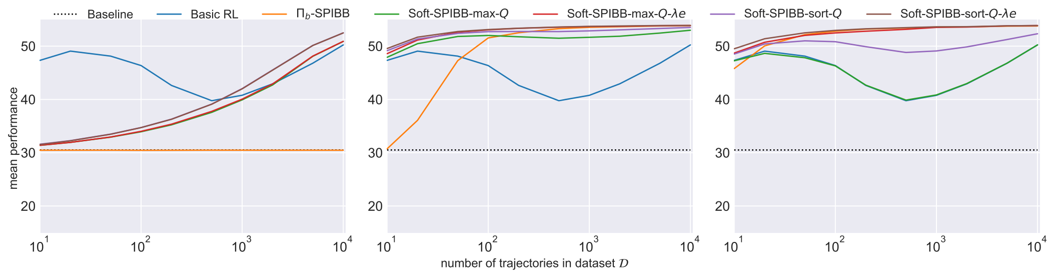
4. Soft-SPIBB Empirical Evaluation

The Gridworld Task We consider the exact same setting as in Laroche and Trichelair (2017): a discrete and stochastic 5×5 gridworld with 4 actions (up, down, left and right). The goal is to go from the bottom left corner to the top right one with a maximum cumulative reward. The action taken moves the agent in the specified direction with 75% chance, in the opposite direction with 5% chance, and sideways with 10% chance each. The agent gets a -10 reward when hitting a wall (and does not move in this case), $+100$ when reaching the final state, and 0 elsewhere. Each run consists in generating trajectories with the baseline, training the algorithms on this batch of data and evaluating the performance of the learned policies. We analyze the mean performance of all the runs and the mean performance over the 1% worst runs (worst-centile performance) to empirically assess safety.

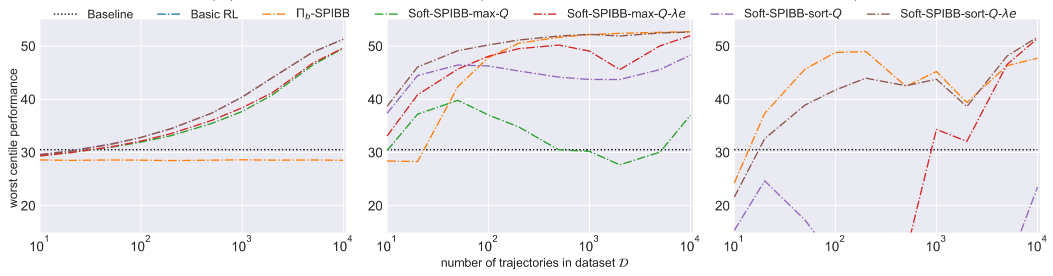
Results We evaluate Soft-SPIBB-max-Q, Soft-SPIBB-sort-Q, Soft-SPIBB-max-Q-λe and Soft-SPIBB-sort-Q-λe with $\lambda = 1$ on the gridworld task and compare them against Π_b-SPIBB (Laroche and Trichelair, 2017) and *basic RL* (which consists in computing the MLE MDP of the environment and solving it with dynamic programming). The Q-function was initialized with a pessimistic value (0), and the baseline policy is obtained with a softmax exploration around the optimal Q-function. We repeated 8000+ runs for each algorithm in the benchmark, 10 dataset sizes ranging from 10 to 10000, 10 ϵ values from 0.01 to 5 and $\delta = 1$. Figure 1 shows the results for $\epsilon = 0.05$, 2 and 5.

Basic RL fails at being safe: its worst-centile performance is too low to appear on the plots. For low ϵ values, Π_b-SPIBB does not outperform the baseline as it bootstraps on π_b for each state-action pair (see figures where $\epsilon = 0.05$ *i.e.* $N_\wedge > 23500$). Our Soft-SPIBB methods perform better than Π_b-SPIBB in both mean and worst-case scenarios. The improvement is all the more significant than the dataset is large: with 10000 trajectories, Soft-SPIBB gives a mean and worst-case performance at least 20 points better than the baseline and Π_b-SPIBB. Even with very few trajectories, Soft-SPIBB outperforms SPIBB, and its worst-centile performance is at most less than 2 points below the baseline. This confirms that our new safety constraint is less binding than the bootstrapped set.

When $\epsilon = 2$, our Soft-SPIBB algorithms are safe and give the best mean and worst-centile performances for small datasets. For all dataset sizes, Soft-SPIBB-sort-Q-λe performs the best on average. We also notice that Soft-SPIBB-sort-Q and Soft-SPIBB-sort-Q-λe yield better mean and worst-centile results than Soft-SPIBB-max-Q and Soft-SPIBB-max-Q-λe respectively, which was expected since Soft-SPIBB-sort-Q allows a smarter and more efficient use of the local error budget. However, we notice that Π_b-SPIBB outperforms Soft-SPIBB with larger datasets. This is explained by our experiments settings: when generating more trajectories, Soft-SPIBB is less constrained and can favor rare state-action pairs with high estimated values, but since we use an optimal baseline, actions that are



(a) Mean performance (from left to right: $\epsilon = 0.05$, 2, 5)



(b) Worst-centile performance (from left to right: $\epsilon = 0.05$, 2, 5)

Figure 1: Soft-SPIBB benchmark: mean and worst-centile performances

rarely sampled actually yield bad returns. In that sense, Soft-SPIBB-max-Q-λe and Soft-SPIBB-sort-Q-λe naturally perform better than their respective non-regularized versions. In particular, Soft-SPIBB-sort-Q-λe is only less than 1 point below Π_b-SPIBB for a small range of dataset sizes, and performs better on the rest. We expect our non-regularized Soft-SPIBB methods to perform better than SPIBB in situations where the optimal actions are not necessarily predominantly selected by the baseline policy.

Finally, figures for $\epsilon = 5$ show that the choice of ϵ is key to ensuring safety: Soft-SPIBB fails at being safe because its constraint is too easily satisfied. Soft-SPIBB-max-Q has the same behavior as basic RL, which makes sense since the former is reduced to the latter when its constraint is removed. Nevertheless, in the worst-centile scenario, Soft-SPIBB-sort-Q-λe is at most only less than 10 points below the baseline and gives the best performance for large datasets, followed by Soft-SPIBB-max-Q-λe.

5. Conclusion

In this paper, we tackle the fundamental problem of learning safe policies given a batch of data. We propose a novel model-based approach for safe policy improvement, Soft-SPIBB, based on a reformulation of the SPIBB objective. Our methodology is more flexible as it allows slight policy changes for uncertain state-action pairs. We derive two computationally-efficient algorithms that return sub-optimal target policies and two variants, and empirically evaluate their mean and worst-centile performances on the gridworld task. Our Soft-SPIBB algorithms remain safe with reasonable ϵ values and show significant advantages: they are less conservative and improve faster than Π_b-SPIBB, and they perform better than basic RL in most settings.

References

Layla El Asri, Bilal Piot, Matthieu Geist, Romain Laroche, and Olivier Pietquin. Score-based inverse reinforcement learning. In *Proceedings of the 15th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 457–465. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

Damien Ernst, Mevludin Glavic, Pierre Geurts, and Louis Wehenkel. Approximate value iteration in the reinforcement learning context. application to electrical power system control. *International Journal of Emerging Electric Power Systems*, 3(1), 2005.

Arthur Guez, Robert D Vincent, Massimo Avoli, and Joelle Pineau. Adaptive treatment of epilepsy via batch-mode reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1671–1678, 2008.

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 Paul Trichelair. Safe policy improvement with baseline bootstrapping. *CoRR*, abs/1712.06924, 2017. URL `http://arxiv.org/abs/1712.06924`.

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

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 Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the 16th European Conference on Machine Learning (ECML)*, pages 317–328. Springer, 2005.

Satinder P Singh, Michael J Kearns, Diane J Litman, and Marilyn A Walker. Reinforcement learning for spoken dialogue systems. In *Proceedings of the 13st Advances in Neural Information Processing Systems (NIPS)*, pages 956–962, 1999.

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

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

Appendix A. Soft-SPIBB-max-Q Algorithm

```
Algorithm 1: Soft-SPIBB-max-$Q$
  Input: Baseline policy $\pi_b$
  Input: Current state $x$ and action set $\mathcal{A}$
  Input: Errors $e'_\delta(x,a)$ for each $a \in \mathcal{A}$
  Input: MDP model precision level $\epsilon$
  Input: Last iteration value function $Q^{(i)}_{\widehat{M}}$
  Result: Next iteration policy $\pi^{(i+1)}$
  Initialize $\pi^{(i+1)}(\cdot|x) = 0$
  $a^* = \operatorname*{argmax}_{a\in\mathcal{A}} \widehat{Q}^{(i)}(x,a)$
  $E = (1 - \pi_b(a^*|x))e'_\delta(x,a^*) + \sum_{a\neq a^*} \pi_b(a|x)e'_\delta(x,a)$
  if $E \leq 2\epsilon$ then
    $\pi^{(i+1)}(a^*|x) = 1$
  else
    for $a \neq a^*$ do
      $\pi^{(i+1)}(a|x) = (1 - \frac{2\epsilon}{E})\pi_b(a|x)$
    end
    $\pi^{(i+1)}(a^*|x) = \pi_b(a^*|x) + \frac{2\epsilon}{E}(1 - \pi_b(a^*|x))$
  end
  if $\sum_{a\in\mathcal{A}} \pi^{(i+1)}(a|x)Q^{(i)}_{\widehat{M}}(x,a) \leq \sum_{a\in\mathcal{A}} \pi^{(i)}(a|x)Q^{(i)}_{\widehat{M}}(x,a)$ then
    $\pi^{(i+1)}(\cdot|x) = \pi^{(i)}(\cdot|x)$
  end
  return $\pi^{(i+1)}$
```

In practice, some state-action pairs might have never been sampled in the dataset, and therefore have infinite transition errors. To cancel these infinite terms in the constraint in (3), we assign the baseline probability to state-action pairs with infinite errors and apply the same greedy procedure on the rest.

Appendix B. Soft-SPIBB-sort-Q Algorithm

Algorithm 2: Soft-SPIBB-sort-Q

```
**Data:** Baseline policy $\pi_b$
**Data:** Last iteration value function $Q^{(i)}_{\widehat{M}}$
**Data:** Current state $x$ and action set $\mathcal{A}$
**Data:** Errors $e'_\delta(x, a)$ for each $a \in \mathcal{A}$
**Data:** MDP model precision level $\epsilon$
**Result:** Next iteration policy $\pi^{(i+1)}$
Initialize $\pi^{(i+1)}(\cdot|x) = \pi_b(\cdot|x)$ and $E = 2\epsilon$
Define $\mathcal{A}^-$ as $\mathcal{A}$ sorted in increasing order of $Q^{(i)}_{\widehat{M}}(x, \cdot)$
for $a^- \in \mathcal{A}^-$ do
    $m^- = \min\left(\pi^{(i+1)}(a^-|x), \frac{E}{2e'_\delta(x, a^-)}\right)$
    $a^+ = \operatorname{argmax}_{a \in \mathcal{A}} \frac{Q^{(i)}_{\widehat{M}}(x, a) - Q^{(i)}_{\widehat{M}}(x, a^-)}{e'_\delta(x, \cdot)}$
    $m^+ = \min\left(m^-, \frac{E}{2e'_\delta(x, a^+)}\right)$
    if $m^+ > 0$ then
        $\pi^{(i+1)}(a^+|x) = \pi^{(i+1)}(a^+|x) + m^+$
        $\pi^{(i+1)}(a^-|x) = \pi^{(i+1)}(a^-|x) - m^+$
        $m^- = m^- - m^+$
        $E = E - m^+ \left(e'_\delta(x, a^+) + e'_\delta(x, a^-)\right)$
    end
end
if $\sum_{a \in \mathcal{A}} \pi^{(i+1)}(a|x) Q^{(i)}_{\widehat{M}}(x, a) \leq \sum_{a \in \mathcal{A}} \pi^{(i)}(a|x) Q^{(i)}_{\widehat{M}}(x, a)$ then
    $\pi^{(i+1)}(\cdot|x) = \pi^{(i)}(\cdot|x)$
end
return $\pi^{(i+1)}$
```