Safe Policy Improvement with Baseline Bootstrapping in Factored Environments

Thiago D. Simão and **Matthijs T. J. Spaan**
Delft University of Technology, The Netherlands
{t.diassimao, m.t.j.spaan}@tudelft.nl

Abstract

We present a novel safe reinforcement learning algorithm that exploits the factored dynamics of the environment to become less conservative. We focus on problem settings in which a policy is already running and the interaction with the environment is limited. In order to safely deploy an updated policy, it is necessary to provide a confidence level regarding its expected performance. However, algorithms for safe policy improvement might require a large number of past experiences to become confident enough to change the agent's behavior. Factored reinforcement learning, on the other hand, is known to make good use of the data provided. It can achieve a better sample complexity by exploiting independence between features of the environment, but it lacks a confidence level. We study how to improve the sample efficiency of the safe policy improvement with baseline bootstrapping algorithm by exploiting the factored structure of the environment. Our main result is a theoretical bound that is linear in the number of parameters of the factored representation instead of the number of states. The empirical analysis shows that our method can improve the policy using a number of samples potentially one order of magnitude smaller than the flat algorithm.

Introduction

Reinforcement Learning (RL) deals with sequential decision making in environments with unknown dynamics. We focus on RL applications in which an existing policy (e.g., engineered by hand) is already in operation, which is relevant in industrial and robotic settings, for instance. The goal is to improve this *behavior policy* while limiting the risks incurred by changing it. Therefore, an important concern here is the *safety* of the proposed solutions. In the literature safety might refer to avoiding catastrophic outcomes or ensuring reasonable performance (García and Fernández 2012). We refer to safety as the guarantee that the new policy will outperform the behavior policy with high probability (Thomas, Theocharous, and Ghavamzadeh 2015a).

There are two major issues for safe RL. The first is related to off-policy evaluation, where given a batch of past experiences the RL agent must estimate the performance of a candidate policy. From a safety perspective it is necessary to have a confidence bound on the predicted performance, considering the stochasticity of the previous experiences. Methods have been proposed to improve the confidence in off-policy evaluation algorithms using both flat and factored representations of the problem (Thomas, Theocharous, and Ghavamzadeh 2015a; Hallak et al. 2015).

The second issue refers to the problem of computing the candidate policy, called Safe Policy Improvement (SPI) (Thomas, Theocharous, and Ghavamzadeh 2015b). In this case the algorithm must compute a new policy at least as good as the behavior policy and if it has a high probability of returning an improved policy, this algorithm is considered safe (Laroche and Trichelair 2018; Thomas, Theocharous, and Ghavamzadeh 2015b; Cohen, Yu, and Wright 2018; Petrik, Ghavamzadeh, and Chow 2016).

A major challenge for current SPI algorithms, however, is that they rely on flat representations, limiting their scalability. In particular, when states are described by a set of features, the number of states grows exponentially in the number of features. In this case, the number of samples necessary to estimate the model or the performance of a policy precisely might be prohibitive, making the application of flat algorithms infeasible.

On the other hand, *factored reinforcement learning* can exploit independence present in the environment and generalize past experiences to new states, which allows the agent to reduce the number of non-optimal actions it takes (Ross and Pineau 2008; Degris, Sigaud, and Wuillemin 2006; Strehl, Diuk, and Littman 2007). However, such algorithms have been proposed for RL settings that ignore safety and in which an agent can explore freely. We aim to bridge the gap between safe and factored RL algorithms.

Our main contribution is an SPI algorithm that uses a factored representation to estimate the dynamics of the environment, assuming that the structure of the problem is known a priori. We prove that by exploiting independence between environment features, our safe RL algorithm obtains a better estimate of the environment dynamics and therefore requires less samples to improve the behavior policy. These results are demonstrated empirically in three experiments with different domains and behavior policies. A highlight of the proposed algorithm is the capability to improve even when the behavior policy is deterministic, which is a strong limitation of previous SPI algorithms.

Background

This section summarizes the formalisms involved in RL and problems with large state space. It focuses on model-based algorithms that explicitly keep track of the uncertainty over the estimated parameters to guide the exploration of the agent. The notion of known and unknown parts of the problem will be used to define safe RL algorithms later.

Markov Decision Process

A Markov Decision Process (MDP) (Puterman 1994) is defined by a tuple $M = \langle S, A, T, R, \gamma \rangle$: a set of states S representing the world, a set of actions A that the agent can take, a probabilistic transition function $T(s' \mid s, a)$ that defines the probability of moving to state $s' \in S$ after action $a \in A$ is executed in state $s \in S$, a reward function $R : S \times A \to \mathbb{R}$ that indicates the immediate reward after executing action $a \in A$ in state $s \in S$, and a discount factor $\gamma \in [0, 1)$. RL deals with problems that can be modeled as an MDP where T is unknown. The goal of an RL agent is to interact with the environment in such a way that maximizes its expected discounted future rewards. In general, the solutions to these problems are based on a trial-and-error strategy where the agent learns through its experiences how the world behaves and how to improve its performance.

The solution of an MDP is a policy $\pi : S \times A \to [0, 1]$, a probability distribution over the actions for each state $s \in S$, such that $\forall s \in S : \sum_{a \in A} \pi(s, a) = 1$. A sequence of interactions between the agent and the environment is represented by a batch $\mathcal{D} = [(s_t, a_t, r_t, s_{t+1}), \cdots]$ where $a_t \sim \pi(s_t)$ is the action applied in the state s_t, $r_t = R(s_t, a_t)$ is the reward obtained and $s_{t+1} \sim T(\cdot \mid s_t, a_t)$. The expected return of a policy is given by $V(\pi, M) = \mathbb{E}_M[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0, \pi]$. The optimal solution of an MDP M is the policy that maximizes the expected return $\pi^* = \arg\max_\pi V(\pi, M)$.

To find the optimal policy quickly, the Rmax algorithm incentivizes the agent to explore unknown parts of the environment in early stages of the learning process (Brafman and Tennenholtz 2002). To do so, it keeps track of a set of state-action pairs considered known, defined as:

$$K_m = \{(s, a) \in S \times A \mid n(s, a) \geq m\}, \tag{1}$$

where $n(s, a)$ is the number of times the agent has applied action a in state s, and m is a threshold to consider a state-action pair known.

Factored MDPs

Often, the state space S can be represented by a set of state factors $X = \{X_1, \cdots, X_{|X|}\}$ where each factor has a domain $Dom(X_i)$, therefore $S = \times_{X_i \in X} Dom(X_i)$. When these features are highly independent, an MDP can be compactly represented by a Factored MDP (FMDP) (Boutilier, Dearden, and Goldszmidt 1995; 2000).

To capture the independence between state factors we adopt the framework defined by Strehl (2007) that uses a dependence function $D : S \times A \times X \to \mathcal{I}$ to indicate the commonalities among different factors, where $\mathcal{I}$ is a set of dependency identifiers. This way, the probabilistic transition function can be represented in the form:

$$T(s' \mid s, a) = \prod_{i=1}^{|X|} P(s'_i \mid D(s, a, X_i)),$$

where s'_i is the value of X_i in the next state s'. Note that this representation of the transition function assumes that the realization of each factor is independent of the realization of the other factors given $D(s, a, X_i)$.

Some definitions follow from this framework: the relevant dependency pairs of a state-action pair $(s, a) \in S \times A$ are denoted by

$$D_{s,a} = \{(X_i, j) \in X \times \mathcal{I} \mid j = D(s, a, X_i)\}$$

and the set of all transition components by $\mathcal{Q} = \bigcup_{(s,a) \in S \times A} D_{s,a}$. The size of $\mathcal{Q}$ denotes the number of transition components that must be estimated by an RL algorithm.

Reinforcement Learning in Factored MDPs

Next, we review how an algorithm that exploits the independence between the factors of an FMDP defines which state-action pair is known or not. We use this criterion in the SPI algorithm for environments with factored dynamics.

To measure the exploration efficiency of an RL agent one can evaluate its *sample complexity*: the expected number of non-optimal actions the agent will take before it starts acting optimally. Rmax has a sample complexity polynomial in the number of states, which means exponential in the number of factors. Factored RL algorithms can have a sample complexity that scales only polynomially in the number parameters of the FMDP (Kearns and Koller 1999; Guestrin, Patrascu, and Schuurmans 2002; Strehl 2007).

The factored Rmax algorithm is a direct extension of Rmax for FMDPs (Guestrin et al. 2003). It maintains an estimate of each transition component distribution and decides which state-action pairs are known or not according to these estimates. This algorithm keeps track of two types of counters: *realization counters*

$$n(x_i, j) = \sum_{s,a,s' \in \mathcal{D}} \mathbb{1}(D(s, a, X_i) = j \text{ and } s'_i = x_i)$$

and *component counters*

$$n(j) = \sum_{s,a,s' \in \mathcal{D}} \sum_{X_i \in X} \mathbb{1}(D(s, a, X_i) = j)$$

where $x_i \in Dom(X_i)$ and $j \in \mathcal{I}$. The distribution of a transition component is given by

$$\hat{P}(s'_i \mid D(s, a, X_i)) = \frac{n(s'_i, D(s, a, X_i))}{n(D(s, a, X_i))} \tag{2}$$

and the transition function is estimated as

$$\hat{T}(s' \mid s, a) = \prod_{i=1}^{|X|} \hat{P}(s'_i \mid D(s, a, X_i)). \tag{3}$$

Unlike Rmax, this algorithm only considers as known parts of the environment where the estimate of all transition components have been experienced enough times. In particular, given a minimum number of samples for each factor

$\vec{m} = \langle m_1, \cdots, m_{|X|} \rangle$ and the counters of each transition component $n(j)$, the set of known state-action pairs is constructed as follows:

$$K_{\vec{m}} = \{(s,a) \in S \times A \mid \forall X_i : n(D(s,a,X_i)) \geq m_i\}\,. \quad (4)$$

In situations where an arbitrary policy is already in execution and the experiences with the environment were recorded in a batch $\mathcal{D}$, an RL algorithm algorithm can use $\mathcal{D}$ to compute a new policy π'. The next section presents an emergent area of RL where the agent must have a high confidence in the performance of π' given $\mathcal{D}$.

Safe Policy Improvement

This section reviews the SPI problem and a state-of-the-art method to solve it, which will be extended in the next section to factored MDPs.

Optimization Criterion

SPI addresses the question of how to compute a new policy π that outperforms the behavior policy π_b with high confidence $1 - \delta$, given a batch of previous interactions $\mathcal{D}$ and an admissible error ζ. Before formalizing the safety criterion used in this paper we present a few definitions.

Let $\hat{M}$ be the maximum likelihood estimate of the underlying MDP built according to the past experiences $\mathcal{D}$. Let $e : S \times A \to \mathbb{R}$ be an arbitrary error function that represents the parametric uncertainty over the transition function $\hat{T}(\cdot \mid s,a)$. The uncertainty set $\Xi(\hat{M}, e)$ is the set of MDPs with transition function T', such the L_1 distance between $T'(\cdot \mid s,a)$ and $\hat{T}(\cdot \mid s,a)$ is smaller than $e(s,a)$ for every state-action pair, that is

$$\left\| \hat{T}(\cdot \mid s,a) - T'(\cdot \mid s,a) \right\|_1 \leq e(s,a) \quad \forall (s,a) \in S \times A.$$

The idea is to define e in such a way that $\Xi(\hat{M}, e)$ includes the true MDP with high probability.

Laroche and Trichelair (2018) proposed the SPI by Baseline Bootstrapping (SPIBB) criterion defined only over the maximum likelihood estimate of the MDP, which is easier to solve. According to this new criterion, an RL algorithm is called *safe* if, given a confidence level δ and a level of precision ζ, it has a high probability $1 - \delta$ of returning a policy that is ζ-approximate as good as the behavior policy π_b on all MDPs in $\Xi(\hat{M}, e)$:

$$\max_{\pi \in \Pi} V(\pi, \hat{M}) \text{ s.t.}$$
$$\forall M' \in \Xi(\hat{M}, e) : V(\pi, M') \geq V(\pi_b, M') - \zeta. \quad (5)$$

Note that according to this criterion an algorithm that always returns the behavior policy is considered safe.

SPI by Baseline Bootstrapping Algorithms

The SPIBB framework is a model-based approach that guarantees safety by bootstrapping unknown parts of the approximated model with the behavior policy π_b (Laroche and Trichelair 2018). Formally, the set of bootstrapped state-action pairs $\mathfrak{B}_m$ is the complement of K_m (1). This way, the SPIBB algorithms guarantee to perform at least as well as the behavior policy and does not rely on a safety test, in contrast to other SPI algorithms.

Algorithm 1 Policy-based SPIBB (Π_b-SPIBB).

```
Input: Previous experiences $\mathcal{D}$
Input: Parameters $\epsilon, \delta$
Input: Behavior policy $\pi_b$
Output: Safe Policy
Estimate $\hat{T}$
Compute $\mathfrak{B}_m = \overline{K_m}$
Compute $\Pi_b$                                    ▷ Equation 6
return $\arg\max_{\pi \in \Pi_b} V(\pi, \hat{M})$
```

The SPIBB algorithm has two variants that bootstrap the behavior policy in different ways. The *value-based* algorithm uses the estimated performance of the elements of $\mathfrak{B}_m$ during the planning phase and if a state-action pair from $\mathfrak{B}_m$ is used during execution, control is returned to the behavior policy. The *policy-based* Π_b-SPIBB algorithm attributes the same probability to bootstrapped pairs as the behavior policy, which restricts the policy space to

$$\Pi_b = \{\pi \mid \pi(s,a) = \pi_b(s,a) : \forall \pi \in \Pi, \forall (s,a) \in \mathfrak{B}_m\}. \quad (6)$$

Laroche and Trichelair (2018) proved that if $m = \frac{2}{\epsilon^2} \log \frac{|S||A|2^{|S|}}{\delta}$ then the Π_b-SPIBB algorithm is safe, where ϵ is a bound on the L_1 distance between the estimated transition function and the true transition function, that depends on the precision parameter ζ. Algorithm 1 gives a brief description of the Π_b-SPIBB approach.

The $\Pi_{\leq b}$-SPIBB algorithm is a variation of the Π_b-SPIBB algorithm where the constrained space of policies is defined as follows:

$$\Pi_{\leq b} = \{\pi | \pi(s,a) \leq \pi_b(s,a) : \forall \pi \in \Pi, \forall (s,a) \in \mathfrak{B}_m\}. \quad (7)$$

In this case, it is possible to reduce the probability attributed to bootstrapped actions in case other actions, that have already being sampled enough times, have a better performance.

In their experimental analysis, Laroche and Trichelair (2018) used a stochastic baseline policy with softmax exploration over the optimal value function. As expected, the Π_b-SPIBB algorithm displayed a safe behavior. Although the $\Pi_{\leq b}$-SPIBB algorithm has not been proven to be safe (in contrast to the Π_b-SPIBB algorithm), the experimental analysis showed that it can also have a safe behavior.

Since the Π_b-SPIBB and $\Pi_{\leq b}$-SPIBB algorithms can change the policy in only a subset of the state-action pairs, they were demonstrated to be less conservative than other SPI algorithms. Nevertheless, when the problem is described by a set of factors, m grows exponentially in the number of factors. The next section shows that, by taking in account the independence between features, it is possible to exploit the factored representation of the problem using a minimum number of samples that is only polynomial in the number of parameters of the FMDP.

Factored Safe Policy Improvement

This section shows how to adapt the SPIBB methodology to environments with factored dynamics, assuming that the dependence between the factors is known a priori, although the distribution of each factor is unknown. First, we describe how the first two steps of the policy-based SPIBB algorithm can be adapted to this setting. Next, we prove that this algorithm is safe.

Factored Policy-Based SPIBB

Algorithm 2 presents the Factored Π_b-SPIBB algorithm, an adaptation of Π_b-SPIBB algorithm for factored environments. Note that this algorithm takes an extra input: the dependency function D used to determine which transition components must be estimated.

First, the algorithm estimates each transition component according to $\mathcal{D}$ using the same counters as the factored Rmax algorithm. The set of state-action pairs to be bootstrapped $\mathfrak{B}_{\vec{m}}$ is the complement of the set of known state-action pairs $K_{\vec{m}}$ (4). In the next section, we show how each value in $\vec{m}$ must be defined to ensure the safety of this algorithm. Given $\mathfrak{B}_{\vec{m}}$ and the behavior policy π_b, the constrained policy space Π_b is computed using Equation 6. Finally, the algorithm searches for an optimal policy in Π_b, however, in this case the transition function $\hat{T}(\cdot \mid s, a)$ is estimated according to the estimate of each transition component (3).

Replacing (6) by (7) in Algorithm 2, we obtain the Factored $\Pi_{\leq b}$-SPIBB algorithm, the factored version of the $\Pi_{\leq b}$-SPIBB algorithm. As we mentioned before, Laroche and Trichelair (2018) also proposed a value-based SPIBB algorithm. However, the development of an effective factored version of this method would require a factored representation of the value function, which is typically not compactly factorized.

Benefits of a Factored Representation

There are two main benefits of exploiting factored representation in the safe RL setting that we are considering: reduced sample complexity and being able to generalize better from deterministic behavior policies. We detail both advantages below.

First, similar to factored Rmax, these algorithms enumerate the set of states to define $\mathfrak{B}_{\vec{m}}$ and compute a new policy. However, we would like to point out that the main goal of the new Factored Π_b-SPIBB algorithm is to improve the precision of the estimated transition function, which allows these algorithms to become less conservative. For example, consider the problem of controlling the temperature of three rooms, with temperatures T_1, T_2 and T_3. Because the first room only shares a wall with the second room, the next value of T_1 is conditionally independent of T_3 given T_1 and T_2. To estimate the future value of T_1 given $T_1 = t_1$, $T_2 = t_2$ and $T_3 = t_3$, all past experiences where $T_1 = t_1$ and $T_2 = t_2$ can be used, regardless of the temperature of T_3. This way, using a factored representation, a safe RL algorithm would need less samples to change how it controls the temperature of the first room, since it has a better estimate of the environment dynamics.

Algorithm 2 Factored Π_b-SPIBB.

```
Input: Previous experiences $\mathcal{D}$
Input: Parameters $\epsilon, \delta$
Input: Behavior policy $\pi_b$
Input: Dependency function $D$
Output: Safe Policy
  Estimate $\hat{P}(\cdot \mid j), \forall (X_i, j) \in \mathcal{Q}$        ▷ Equation 2
  Compute $\mathfrak{B}_{\vec{m}} = \overline{K_{\vec{m}}}$
  Compute $\Pi_b$                                                           ▷ Equation 6
  return $\arg\max_{\pi \in \Pi_b} V(\pi, \hat{M})$
```

Second, we would like to point out that by using a factored representation the Factored SPI algorithms can choose an action a in a state s even if $n(s, a) = 0$, a common case when the behavior policy is deterministic. In this case, a flat algorithm would never execute a in s. Therefore, the application of flat SPI algorithms is limited, since they are not able increase the probability of a if $\pi_b(s, a) = 0$. Consider for example a deterministic behavior policy that always executes a in the state s. In this case, $n(s, a') = 0, \forall a' \in A \setminus \{a\}$ and a flat SPI algorithm would always return a policy π where $\pi(s, a) = 1$ and $\pi(s, a') = 0, \forall a' \in A \setminus \{a\}$. The capacity of generalization of a factored representation can deal with this limitation: the agent can estimate the transition components of each factor X_i using past experiences where a was executed in other states $s' \in S \setminus \{s\}$ where $s'_i = s_i$. Using this estimate of each component the agent can estimate $\hat{T}(\cdot \mid s, a)$ and, eventually, choose to execute a in s. We demonstrate this feature in an experiment with a deterministic behavior policy.

Theoretical Analysis

In this section, we show that given the admissible error parameter ζ, the Factored Π_b-SPIBB algorithm satisfies the SPIBB safety criterion (5).

First, we show that the error of the transition function is bounded with high probability in all state-action pairs that are not bootstrapped by the Factored Π_b-SPIBB algorithm. With Corollary 1 by Strehl (2007) we can bound the L_1 distance between the estimated transition function computed by the product of a set of components and the true transition function, given that the error of each component is bounded.

Lemma 1 (Corollary 1 by Strehl (2007))**.** *Let M be any factored MDP. Suppose that for each transition component $P(\cdot \mid j)$ we have an estimate $\hat{P}(\cdot \mid j)$ such that*

$$\|P(\cdot \mid j) - \hat{P}(\cdot \mid j)\|_1 \leq \epsilon/|X|.$$

Then, for all state action pairs

$$\|T(\cdot \mid s, a) - \hat{T}(\cdot \mid s, a)\|_1 \leq \epsilon.$$

Next, we redefine Proposition 3 by Laroche and Trichelair (2018) for the case where the transition function is estimated according to the estimate of each transition component (3).

Proposition 1. *Consider an environment modeled by a semi-MDP M and the empirical semi-MDP $\hat{M}$ estimated* *from a dataset $\mathcal{D}$. If in every state s where option ${o_a}^1$ may be initiated, $s \in \mathcal{I}_a$, we have that for all relevant components $(X_i, j) \in D_{s,a}$*

$$\sqrt{\frac{2|X|^2}{n(j)} \log \frac{|\mathcal{Q}|2^{|Dom(X_i)|}}{\delta}} \leq \epsilon, \tag{8}$$

then holds that

$$Pr(\|T(\cdot \mid s,a) - \hat{T}(\cdot \mid s,a)\|_1 \geq \epsilon) \leq \delta, \forall (s,a) \notin \mathfrak{B}_{\tilde{m}}.$$

Proof. Using Weissman et al. (2003)'s Theorem 2.1, for each transition component $(X_i, j) \in \mathcal{Q}$, we may write:

$$\begin{aligned} Pr(\|P(\cdot \mid j) - \hat{P}(\cdot \mid j)\|_1 \geq \epsilon_1) \\ \leq (2^{|Dom(X_i)|} - 2) \exp\left(-\frac{n(j)\epsilon_1^2}{2}\right). \end{aligned} \tag{9}$$

This equation bounds the error of each transition function component. To use Lemma 1, we set $\epsilon_1 = \epsilon / |X|$ and rewrite (9) as

$$Pr\Big(\|P(\cdot \mid j) - \hat{P}(\cdot \mid j)\|_1 \geq \epsilon/|X|\Big) \tag{10}$$

$$\leq (2^{|Dom(X_i)|} - 2) \exp\Big(-\frac{n(j)\epsilon^2}{2|X|^2}\Big) \tag{11}$$

$$\leq 2^{|Dom(X_i)|} \exp\Big(-\frac{n(j)}{2|X|^2}\frac{2|X|^2}{n(j)} \log \frac{|\mathcal{Q}|2^{|Dom(X_i)|}}{\delta}\Big) \tag{12}$$

$$= 2^{|Dom(X_i)|} \exp\Big(-\log \frac{|\mathcal{Q}|2^{|Dom(X_i)|}}{\delta}\Big) \tag{13}$$

$$= 2^{|Dom(X_i)|} \exp\Big(\log \frac{\delta}{|\mathcal{Q}|2^{|Dom(X_i)|}}\Big) \tag{14}$$

$$= 2^{|Dom(X_i)|} \frac{\delta}{|\mathcal{Q}|2^{|Dom(X_i)|}} \tag{15}$$

$$= \frac{\delta}{|\mathcal{Q}|}. \tag{16}$$

Given a bound on the probability of each component being inaccurate, we can now bound the probability that there exists a state-action pair whose transition distribution is inaccurate. In the following derivation (a) comes from Lemma 1, (b) is an application of the union bound over all components in $\mathcal{Q}$ and (c) comes from (16).

$$Pr(\|T(\cdot \mid s,a) - \hat{T}(\cdot \mid s,a)\|_1 \geq \epsilon) \tag{17}$$

$$\overset{(a)}{=} Pr\Big(\bigcup_{(X_i,j)\in\mathcal{Q}} \|P(\cdot \mid j) - \hat{P}(\cdot \mid j)\|_1 \geq \frac{\epsilon}{|X|}\Big) \tag{18}$$

$$\overset{(b)}{\leq} \sum_{i=1}^{|\mathcal{Q}|} Pr\Big(\|P(\cdot \mid j) - \hat{P}(\cdot \mid j)\|_1 \geq \epsilon/|X|\Big) \tag{19}$$

$$\overset{(c)}{\leq} \sum_{i=1}^{|\mathcal{Q}|} \frac{\delta}{|\mathcal{Q}|} \tag{20}$$

$$= \delta, \tag{21}$$

which concludes our proof. □

Now, to ensure that the conditions for Proposition 1 hold, we can set the minimum number of observations for each component to $m_i = \frac{2|X|^2}{\epsilon^2} \log \frac{|\mathcal{Q}|2^{|Dom(X_i)|}}{\delta}$, which is derived from (8) by isolating $n(j)$. This guarantees that all components relevant for non-bootstrapped state-action pairs were experienced enough times, such that the error of the transition function of these state-action pairs is smaller than ϵ. This way, we can use Proposition 1 to replace Proposition 3 in the proof of Theorem 3 (Laroche and Trichelair 2018).

Theorem 1. *(Safe Policy Improvement of the Factored Π_b-SPIBB Algorithm). Let Π_b be the set of policies under the constraint of following π_b in every bootstrapped state-action pair$(s,a) \in \mathfrak{B}_{\tilde{m}}$. Then, the policy π_{pol} computed by the Factored Π_b-SPIBB algorithm, is at least a ζ-approximate safe policy improvement over π_b with high probability $1-\delta$, with*

$$\zeta = \frac{4\epsilon V_{\max}}{(1-\gamma)} - V(\pi_{pol}, \hat{M}) + V(\pi_b, \hat{M}).$$

Proof. The proof is similar to that of Theorem 3 (Laroche and Trichelair 2018). □

These results show that it is possible to bound the probability that the Factored Π_b-SPIBB algorithm computes a policy worse than the behavior policy. The main difference with the original Π_b-SPIBB algorithm is the way we bound the error of the transition function. Given a desired ϵ, the term $|A||S|$ is replaced by $|\mathcal{Q}|$ and $|S|$ is now reduced to $|Dom(X_i)|$. This comes at the lower cost of adding a term polynomial in the number of variables $|X|^2$ (necessary to bound the error of each component distribution). In domains where the features are highly independent from each other these results can reduce significantly the number of samples necessary to improve the behavior policy, as demonstrated in the empirical analysis.

Empirical Analysis

We evaluate the proposed factored approaches for the SPI problem focusing on their sample efficiency and generalization capability. All algorithms use a flat representation to estimate the transition function, as in the Π_b-SPIBB algorithm, and flat Value Iteration with a discount factor of 0.99 to compute the new policy. We assume that the reward function is known in all algorithms.

We evaluate the Π_b-SPIBB and Factored Π_b-SPIBB algorithms and their respective relaxations $\Pi_{\leq b}$-SPIBB and Factored $\Pi_{\leq b}$-SPIBB. We compare these results with two basic model-based RL algorithms that simply estimate the underlying model and compute a greedy policy according to this estimate. The first, called Basic Flat RL, uses a flat representation and the second, called Basic Factored RL, uses a factored representation.

Experimental Setup

We use two domains with known independence between features: i) the Taxi domain (Dietterich 1998) that has 4 conditionally independent features, 500 states, 6 actions and a horizon of 200 steps, and ii) the SysAdmin domain with

[1] Option o_a is the counterpart of the original action a in the semi-MDP.

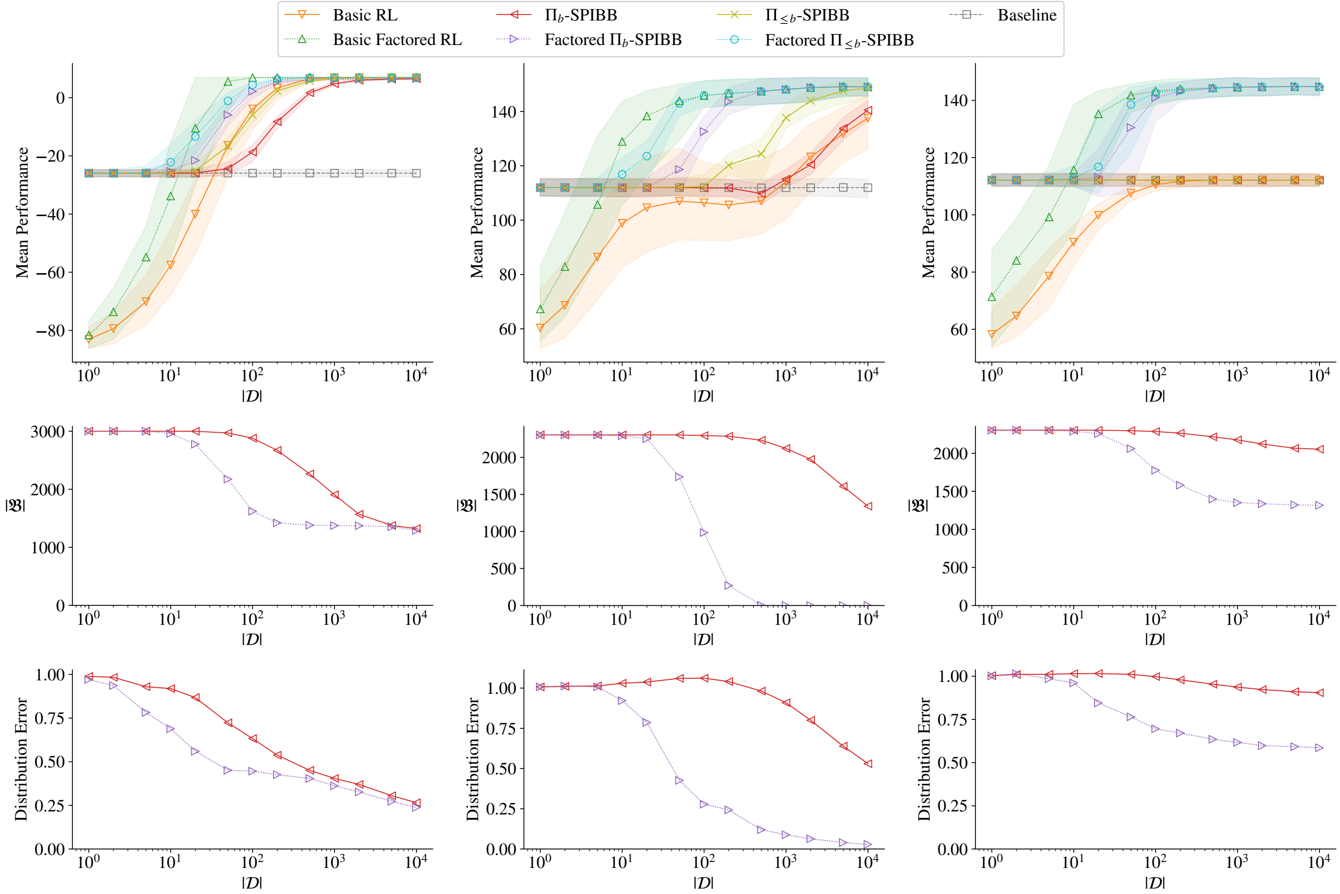

Figure 1: In every plot the x-axis shows the number of trials in the batch. Each column shows the results of a different experiment: Taxi with softmax policy (left), SysAdmin with softmax policy (middle) and SysAdmin with deterministic policy (right). The rows present: i) the average performance of the computed policy (top), ii) the number of bootstrapped state-action pairs (middle), and iii) the average distribution error of the estimated transition function (bottom).

8 machines in a bidirectional ring topology (Guestrin et al. 2003), that has 256 states, 9 actions and a horizon of 40.

Our analysis uses three metrics: i) the performance of the policies computed; ii) the size of the set of bootstrapped state-action pairs; and iii) the distribution error of $\hat{T}$, defined as the average of the L_1 distance between the estimated transition function and the true transition function.

The first two experiments have a setup similar to the empirical evaluation of the SPIBB methodology (Laroche and Trichelair 2018). A baseline policy π_b is computed using softmax exploration over the optimal value function of each state-action pair (temperature 2 for the Taxi domain and 3 for the SysAdmin domain). Next, a batch of past experiences $\mathcal{D}$ is generated following π_b. Note that $\mathcal{D}$ is composed of a set of trajectories, therefore $|\mathcal{D}|$ denotes the number of trajectories in $\mathcal{D}$. Each algorithm uses $\mathcal{D}$ and π_b to compute a new policy π. Finally, the policies computed are evaluated by averaging the returns of 1000 simulations.

The third experiment uses the same instance of the SysAdmin domain, but with a deterministic behavior policy that always takes the action with the second highest expected value. This way, there is space for improvement in every state. Note that because the policy is deterministic, this experiment requires the agent to be able to generalize its past experiences to new states.

We performed a parameter search in each domain to choose the minimum value for m and m_i that maintains the safety of the algorithm. For the Taxi domain we set $m = 10$ and $m_i = 20$ for $0 < i \leq |X|$. In the case of the SysAdmin problem we set $m = 50$ and $m_i = 10$ for $0 < i \leq |X|$.

Experimental Results

Figure 1 shows the results obtained. Each column presents a different experiment and each row a different metric. The first row shows the average performance of the policies computed over 1000 repetitions and the 1%-quantile of these values, which lets us assess the safety of each method. On the second and third row we omit the results of Basic Flat RL, Basic Factored RL, $\Pi_{\leq b}$-SPIBB, Factored $\Pi_{\leq b}$-SPIBB that are equal to Π_b-SPIBB and Factored Π_b-SPIBB respectively or do not apply. Note that solid (dotted) lines are used for algorithms that use a flat (factored) representation, and dashed lines are used for the baseline policy.

In the Taxi experiment (Figure 1, first column) we no-

Algorithm	**Mean**	**1%-quantile**
Baseline Value	-25.91	-27.09
Basic RL	-16.43	-27.24
Basic Factored RL	5.56	-4.71
Π_b-SPIBB	-24.36	-26.54
Factored Π_b-SPIBB	-5.94	-9.72
$\Pi_{\leq b}$-SPIBB	-16.6	-19.7
Factored $\Pi_{\leq b}$-SPIBB	-1.08	-4.32

Table 1: Performance of policies computed when $|\mathcal{D}| = 50$.

tice that although the unsafe algorithms (Basic Flat RL and Basic Factored RL) can improve their performance quickly, they obtain policies worse than the behavior policy when the batch contains only a few trajectories, which is exactly what safe reinforcement learning tries to avoid. All the other algorithms are shown to be safe as expected.

The main result of this paper is the difference in the number of samples necessary to change the behavior policy of flat SPI algorithms and their factored counterparts. The Factored Π_b-SPIBB algorithm manages to compute policies better than the behavior policy given batches with only 20 trajectories, in contrast to the Π_b-SPIBB algorithm, that only shows improvement when $|\mathcal{D}| \geq 50$. As already demonstrated, the $\Pi_{\leq b}$-SPIBB algorithm can be less conservative (Laroche and Trichelair 2018), and is able to find better improvements when $|\mathcal{D}| = 50$, while its factored version (Factored $\Pi_{\leq b}$-SPIBB) is even less conservative finding improvements when $|\mathcal{D}| = 5$ and already getting close to the optimal policy when $|\mathcal{D}| = 50$. To provide a more precise measure of these differences, Table 1 shows the results when $|\mathcal{D}| = 50$.

When comparing the number of bootstrapped state-action pairs by each algorithm (Figure 1, second row) and the performance of the policy computed we see a strong correlation between both. That is, a smaller number of bootstrapped state-action pairs results in a higher performance. The quicker reduction of the distribution error (Figure 1, third row) shows why the Factored SPIBB algorithm can bootstrap from fewer state-action pairs; this is a clear result of the generalization capacity of factored representations.

In the SysAdmin with softmax policy experiment (Figure 1, second column) the results are similar, although the differences between (Factored) Π_b-SPIBB and (Factored) $\Pi_{\leq b}$-SPIBB algorithms are much larger than in the first experiment. We also notice that for the factored algorithms $|\mathfrak{B}|$ drops quickly between $|\mathcal{D}| = 20$ and $|\mathcal{D}| = 200$ when these algorithms stop bootstrapping and achieve the same performance as Basic Factored RL.

Finally, in the SysAdmin with deterministic behavior policy experiment (Figure 1, third column), the factored algorithms are the only ones that manage to improve the behavior policy. As expected, none of the flat algorithms can find a policy better than the behavior policy, given that the behavior policy is deterministic. We notice that the distribution error of the flat representation only drops slightly while for the factored representation it drops to an average of 0.5, which is enough to let the factored algorithms stop bootstrapping some of the state-action pairs.

Related work

Factored RL has also been studied in settings where the dependence structure is unknown. These approaches include methods to search for the underlying structure with and without guarantees of sample complexity (Strehl, Diuk, and Littman 2007; Diuk, Li, and Leffler 2009; Chakraborty and Stone 2011; Degris, Sigaud, and Wuillemin 2006). Although these methods do not have a safety guarantee, we believe they could naturally be coupled to our algorithm.

Factored representations have also been used to tackle the off-policy policy evaluation problem. High confidence off-policy evaluation (HCOPE) is a model-free method based on importance sampling that estimate the performance of a candidate policy with a given confidence level (Thomas, Theocharous, and Ghavamzadeh 2015a). From a model-based perspective, Hallak et al. (2015) showed that using a factored representation to estimate the model one can find accurate estimates of the performance of a target policy.

Thomas, Theocharous, and Ghavamzadeh (2015b) proposed a model-free approach to tackle the SPI problem. It divides the dataset in two partitions $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$. Using $\mathcal{D}_{train}$, this algorithm computes a new policy and, to ensure safety, it checks with the HCOPE test if the new policy is safe using $\mathcal{D}_{test}$. If the computed policy is not considered safe it returns an indication that no improved policy was found, in which case the RL agent could keep executing the behavior policy. To exploit factored dynamics, the method proposed by Hallak et al. (2015) could be used following a similar strategy.

Conclusions and Future Work

We proposed the Factored Π_b-SPIBB algorithm, an adaptation of the Π_b-SPIBB algorithm (Laroche and Trichelair 2018) to environments with factored dynamics and proved that this algorithm is safe. The Factored Π_b-SPIBB algorithm inherits from factored RL the capability to exploit the independence between features, allowing it to reduce the number of samples necessary to stop bootstrapping the behavior policy. This new method also is able to improve deterministic policies by generalizing past experiences, a novel feature for safe RL algorithms.

For simplicity, we assumed that the reward function is known. However, this function can also be succinctly represented in cases where it has an additive property. Therefore, it would also be possible to adapt the Factored Π_b-SPIBB algorithm to environments with an unknown reward function. Furthermore, extending the Factored Π_b-SPIBB algorithm to settings with unknown dependence between features is a promising avenue of future work. Finally, we believe it is also possible to extend other model-based safe RL methods to factored environments, such as the robust approach by Petrik, Ghavamzadeh, and Chow (2016).

Acknowledgments

This research received funding from the Netherlands Organisation for Scientific Research (NWO).

References

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 1995. Exploiting Structure in Policy Construction. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 1104–1113. Morgan Kaufmann.

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 2000. Stochastic dynamic programming with factored representations. *Artificial Intelligence* 121(1-2):49–107.

Brafman, R. I., and Tennenholtz, M. 2002. R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *Journal of Machine Learning Research* 3:213–231.

Chakraborty, D., and Stone, P. 2011. Structure Learning in Ergodic Factored MDPs without Knowledge of the Transition Function's In-Degree. In *Proc. of International Conference on Machine Learning*, 737–744. Omnipress.

Cohen, A.; Yu, L.; and Wright, R. 2018. Diverse Exploration for Fast and Safe Policy Improvement. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press.

Degris, T.; Sigaud, O.; and Wuillemin, P. 2006. Learning the Structure of Factored Markov Decision Processes in Reinforcement Learning Problems. In *Proc. of International Conference on Machine Learning*, volume 148 of *ACM International Conference Proceeding Series*, 257–264. ACM.

Dietterich, T. G. 1998. The MAXQ Method for Hierarchical Reinforcement Learning. In *Proc. of International Conference on Machine Learning*, volume 98, 118–126.

Diuk, C.; Li, L.; and Leffler, B. R. 2009. The Adaptive k-meteorologists Problem and Its Application to Structure Learning and Feature Selection in Reinforcement Learning. In *Proc. of International Conference on Machine Learning*, 249–256. ACM.

García, J., and Fernández, F. 2012. Safe Exploration of State and Action Spaces in Reinforcement Learning. *Journal of Artificial Intelligence Research* 45:515–564.

Guestrin, C.; Koller, D.; Parr, R.; and Venkataraman, S. 2003. Efficient Solution Algorithms for Factored MDPs. *Journal of Artificial Intelligence Research* 19:399–468.

Guestrin, C.; Patrascu, R.; and Schuurmans, D. 2002. Algorithm-Directed Exploration for Model-Based Reinforcement Learning in Factored MDPs. In *Proc. of International Conference on Machine Learning*, 235–242. Morgan Kaufmann.

Hallak, A.; Schnitzler, F.; Mann, T. A.; and Mannor, S. 2015. Off-policy Model-based Learning under Unknown Factored Dynamics. In *Proc. of International Conference on Machine Learning*, 711–719. JMLR.org.

Kearns, M. J., and Koller, D. 1999. Efficient Reinforcement Learning in Factored MDPs. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 740–747. Morgan Kaufmann.

Laroche, R., and Trichelair, P. 2018. Safe Policy Improvement with Baseline Bootstrapping. In *14th European Workshop on Reinforcement Learning*. `http://arxiv.org/abs/1712.06924`.

Petrik, M.; Ghavamzadeh, M.; and Chow, Y. 2016. Safe Policy Improvement by Minimizing Robust Baseline Regret. In *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc. 2298–2306.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1st edition.

Ross, S., and Pineau, J. 2008. Model-Based Bayesian Reinforcement Learning in Large Structured Domains. In *Proc. of Uncertainty in Artificial Intelligence*, 476–483. AUAI Press.

Strehl, A. L.; Diuk, C.; and Littman, M. L. 2007. Efficient Structure Learning in Factored-State MDPs. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, 645–650. AAAI Press.

Strehl, A. L. 2007. Model-Based Reinforcement Learning in Factored-State MDPs. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 103–110. IEEE.

Thomas, P. S.; Theocharous, G.; and Ghavamzadeh, M. 2015a. High-Confidence Off-Policy Evaluation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 3000–3006. AAAI Press.

Thomas, P. S.; Theocharous, G.; and Ghavamzadeh, M. 2015b. High Confidence Policy Improvement. In *Proc. of International Conference on Machine Learning*, 2380–2388. JMLR.org.

Weissman, T.; Ordentlich, E.; Seroussi, G.; Verdu, S.; and Weinberger, M. J. 2003. Inequalities for the L_1 Deviation of the Empirical Distribution. *Hewlett-Packard Labs, Tech. Rep.*