Proceedings of Machine Learning Research vol xxx:1–12, 2021

Assured RL: Reinforcement Learning with Almost Sure Constraints

Agustin Castellano ACASTELLANO@FING.EDU.UY
Juan Bazerque JBAZERQUE@FING.EDU.UY
Universidad de la República, Julio Herrera y Reissig 565, Montevideo 11400, Uruguay
Enrique Mallada MALLADA@JHU.EDU
Johns Hopkins University, 3400 N. Charles St., Baltimore, MD 21218, USA

Abstract

We consider the problem of finding optimal policies for a Markov Decision Process with almost sure constraints on state transitions and action triplets. We define value and action-value functions that satisfy a barrier-based decomposition which allows for the identification of feasible policies independently of the reward process. We prove that, given a policy π, certifying whether certain state-action pairs lead to feasible trajectories under π is equivalent to solving an auxiliary problem aimed at finding the probability of performing an unfeasible transition. Using this interpretation, we develop a Barrier-learning algorithm, based on Q-Learning, that identifies such unsafe state-action pairs. Our analysis motivates the need to enhance the Reinforcement Learning (RL) framework with an additional signal, besides rewards, called here *damage function* that provides feasibility information and enables the solution of RL problems with model-free constraints. Moreover, our Barrier-learning algorithm wraps around existing RL algorithms, such as Q-Learning and SARSA, giving them the ability to solve almost-surely constrained problems.

Keywords: Reinforcement Learning, Constrained MDPs, Safety-critical Systems

1. Introduction

The last decade has witnessed a resurgence of Artificial Intelligence (AI) reaching to levels never experienced before. At the center of many of these successes, Reinforcement Learning (Sutton and Barto, 2018) has occupied a critical role, which when combined with modern Machine Learning techniques—such as deep neural networks (Goodfellow et al., 2016)— and increased (energy-efficient) computational power (Jouppi et al., 2017), has led to astonishing demonstrations of super-human performance. While there were already instances of such accomplishments in the turn of the century (Campbell et al., 2002; Schaeffer et al., 1996, 2001) [1], today's successes are pervasive and more impressive. Examples include Jeopardy! (Ferrucci, 2012), Atari (Mnih et al., 2015), Go (Silver et al., 2016), StarCraft II (Vinyals et al., 2017), and even Poker (Nichols et al., 2019).

However, this success is overwhelmingly limited to virtual domains and only ported to the physical realm after training for more than hundreds of equivalent human years (Andrychowicz et al., 2020). There are several challenges that prevent the full realization of RL in physical environments, all of which are closely intertwined. Firstly, due to the high dimensionality of the spatio-temporal domain where learning occurs, thousands of trials (episodes) are needed to achieve accurate policies. Secondly, general-purpose RL algorithms lack the necessary safety guarantees that are required in such applications. While several methods have been proposed to solve safe RL problems (Garcia

1. The prominent instance is the defeat of Garry Kasparov by IBM Deep Blue (Campbell et al., 2002).

and Fernandez, 2015), they often assess safety via soft penalties that lack hard guarantees. This leads to the need for training using simulated environments. Finally, policies learned in simulated environments tend to perform poorly in practice. This further points to the need to enhance training with random permutations of the environment, which in turn makes training even more computational intensive.

Such cyclic dependence between computing requirements, safety requirements and randomized virtual training has given rise to a renewed interest on the study of Constrained Markov Decision Processes (CMDPs) (Altman, 1998). In this setting the conventional return to be maximized is supplemented with one or several additional returns that are to satisfy certain lower bound, in expectation. Due to the additive structure of both the objective and constraints, it is possible to frame such problems as constrained optimization problem where the decision variables correspond to occupational distributions. Such approach has led to a rich body of literature that proposes RL algorithms using penalized primal methods (Geibel, 2006), primal-dual methods (Ding et al., 2020; Paternain et al., 2019), and methods with additional assumptions on the model (Zanon and Gros, 2019; Cheng et al., 2019). Unfortunately, there are some caveats with these approaches. Firstly, while expectation-based constraints of sum of rewards lead to tractable problems, such constraints are not satisfactory for safety-critical applications. Secondly, the above-mentioned solution methods either only approximately satisfy the constraints, or guarantee constraint satisfaction asymptotically. As a result, such schemes tend to experience a large number of constraint violations during training.

In this work, we aim at developing Reinforcement Learning algorithms that can learn and impose safety constraints during training. Unlike Achiam et al. (2017) and Wachi and Sui (2020), who either require prior knowledge of the constraints or seek to learn the constraint function, our algorithm is model-free. Precisely, we consider a finite Markov Decision Process where state transitions and action triplets (s, a, s') must lie within a feasibility set $\mathcal{F}$, almost surely. We show that given a policy π, the value and action value functions satisfy a barrier-based decomposition that decouples the problem of learning the feasibility set $\mathcal{F}$ from the reward process (Section 2). Moreover, certifying whether an initial state-action pair leads to a feasible trajectory with probability one, is equivalent to computing the action-value function of an auxiliary MDP, with identical transition probabilities, but different reward function (called here *damage*), that seeks to quantify the probability of preforming an unsafe transition. Using this equivalence, we develop a barrier-learning algorithm that learns an action-barrier function that implicitly characterizes the set of all state-action pairs (s, a) that with probability one lead to a transition $(s, a, s') \in \mathcal{F}$. Our barrier-learning algorithm wraps around any standard RL algorithm such as Q-Learning and SARSA , and incrementally restricts the exploration of infeasible triplets that are learnt *during training* (Section 4). We illustrate our findings with numerical experiments in Section 5 and conclude in Section 6.

2. Value function decomposition

Consider a Constrained Markov Decision Process (CMDP) with finite state space $\mathcal{S}$ and finite action space $\mathcal{A}$, a reward set $\mathcal{R}$ and a transition kernel p which specifies the conditional transition probability $p(s', r \mid s, a)$, from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ with reward $r \in \mathcal{R}$ under action $a \in \mathcal{A}$. There are constraints that the agent must satisfy almost surely, which are specified through a set $\mathcal{F}$. A triplet (s, a, s') is safe if it belongs to $\mathcal{F}$, and is unsafe otherwise. As usual, a policy π induces a probability distribution over the action space for a given state $\pi(\cdot \mid s)$. In this context, our goal is to maximize the value function for each possible starting state while ensuring constraint satisfaction at

all times:

$$V^*(s) := \max_{\pi} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s \right] \tag{1}$$
$$\text{s.t.:} \quad (S_t, A_t, S_{t+1}) \in \mathcal{F} \quad a.s. \ \forall t$$

where the expectation in the objective is taken over the trajectories induced by π. Let us then define the value function V^π for a specific policy π, in which the constraints in (1) are embedded inside the expectation.

$$V^\pi(s) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \left(\gamma^t R_{t+1} - \mathbb{I}_{\{(S_t, A_t, S_{t+1}) \in \mathcal{F}\}} \right) \mid S_0 = s \right] \tag{2}$$

where we introduced the barrier indicator function

$$\mathbb{I}_{\{\cdot\}} = \begin{cases} 0 & \text{if } \cdot \text{ is true} \\ \infty & \text{if } \cdot \text{ is false} \end{cases} \tag{3}$$

The proposed value function definition will prove useful in two senses: firstly, we will show that maximizing (2) is the same as (1). Secondly, the additional term in (2) will allow for a barrier-based decomposition of the value function, which will aid in the learning of constraints.
Before proceeding any further, we make a technical remark regarding the barrier term in (2).

Remark 1 (A note on the barrier term) *The reason for the addition of the indicator function term in (2) is to make unsafe policies yield $V^\pi(s) = -\infty$. Special care must be taken, though. Suppose there is an unsafe triplet $(s, a, s') \notin \mathcal{F}$ that occurs with zero probability under π. Unrolling the expectation in (2) would give a term $p(s' \mid s, a)\, \mathbb{I}_{\{(s,a,s') \in \mathcal{F}\}}$, which is a zero-times-infinity indetermination. To remedy this we instead consider the following definition for the value function*

$$V^\pi(s) = \lim_{\lambda \to \infty} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \left(\gamma^t R_{t+1} - \lambda \mathbb{1}_{\{(S_t, A_t, S_{t+1}) \notin \mathcal{F}\}} \right) \mid S_0 = s \right] \tag{4}$$

where $\mathbb{1}_{\{\cdot\}}$ is one if the condition is true and zero otherwise. Throughout the paper we will abuse notation and often speak of (2). The reader should bear in mind that we are instead referring to (4).

With that remark aside, we go on to show the equivalence between (1) and the maximization of (2).

Lemma 2 (Equivalence) *Under the common extension where an unfeasible policy for (1) yields $V^\pi(s) = -\infty$, Problem (1) is equivalent to the maximization of (2), that is*

$$\max_{\pi} V^\pi(s) = \max_{\pi} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \left(\gamma^t R_{t+1} - \mathbb{I}_{\{(S_t, A_t, S_{t+1}) \in \mathcal{F}\}} \right) \mid S_0 = s \right] \tag{5}$$

Proof *If a policy π_0 is unfeasible for Problem (1), then $\exists t : P\left((S_t, A_t, S_{t+1}) \notin \mathcal{F}\right) > 0$. This non-zero probability renders the expected value in (5) to $-\infty$ for π_0. Conversely, if a policy π_1 is feasible for (5) then it must necessarily hold that $(S_t, A_t, S_{t+1}) \in \mathcal{F}$ almost surely $\forall t$, and hence π_1 is feasible for (1) as well. Therefore the feasible sets of both problems coincide. Lastly, for every feasible policy it must hold $\mathbb{I}_{\{(S_t, A_t, S_{t+1}) \in \mathcal{F}\}} = 0 \ \forall t$, in which case the function being maximized is the same. Then the optimal sets of the two problems coincide.* ■

While solving (1) is of our utmost interest, we have just shown that, to this end, we can solve (5) instead. In what follows we will take this one step further, and show that (2) admits a *barrier-based decomposition* and can be cast as the sum of two value functions: one that checks only whether the policy in consideration is feasible (which will be the main focus of this work) and one that optimizes the return, provided the policy is feasible. The main idea behind this decoupling being that the search for feasible policies will be, in practice, an easier task to undergo.
To this end we define an auxiliary *barrier-based* value function F^π that will relate to V^π.

$$F^\pi(s) = \mathbb{E}_\pi \left[-\sum_{t=0}^{\infty} \mathbb{I}_{\{(S_t,A_t,S_{t+1})\in\mathcal{F}\}} \mid S_0 = s \right] \tag{6}$$

We proceed similarly for the action-value function Q^π and its counterpart B^π

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \left(\gamma^t R_{t+1} - \mathbb{I}_{\{(S_t,A_t,S_{t+1})\in\mathcal{F}\}}\right) \mid S_0 = s, A_0 = a \right] \tag{7}$$

$$B^\pi(s,a) = \mathbb{E}_\pi \left[-\sum_{t=0}^{\infty} \mathbb{I}_{\{(S_t,A_t,S_{t+1})\in\mathcal{F}\}} \mid S_0 = s, A_0 = a \right] \tag{8}$$

Searching for policies that are optimal for (5) for each possible state is our original goal. By contrast, a problem such as maximizing (6) is one that seeks to find *safe* policies, in the sense that they are *feasible* for (5). The main idea underpinning our work is that we can jointly work on optimizing (6), which reduces the search over the policy space, while at the same time maximizing the return present in (5). In the following Theorem we establish a fundamental decomposition relationship between the value functions and their auxiliary counterparts.

Theorem 3 (Barrier-based decomposition) *Assume rewards R_{t+1} are bounded almost surely for all t. Then, for every policy π*

$$V^\pi(s) = V^\pi(s) + F^\pi(s) \tag{9}$$

$$Q^\pi(s,a) = Q^\pi(s,a) + B^\pi(s,a) \tag{10}$$

Proof *We shall prove the relationship in (9) regarding the value functions (2),(6), noting that the proof for (10) is similar. The following identities hold, as explained below.*

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \left(\gamma^t R_{t+1} - \mathbb{I}_{\{(S_t,A_t,S_{t+1})\in\mathcal{F}\}}\right) \mid S_0 = s \right] \tag{11}$$

$$= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \left(\gamma^t R_{t+1} - \mathbb{I}_{\{(S_t,A_t,S_{t+1})\in\mathcal{F}\}}\right) \mid S_0 = s \right] + \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} -\mathbb{I}_{\{(S_t,A_t,S_{t+1})\in\mathcal{F}\}} \mid S_0 = s \right] \tag{12}$$

To show that (11) can be separated as in (12), first suppose the policy considered in (11) is feasible (for Problem (5)). This necessarily implies that $(S_t, A_t, S_{t+1}) \in \mathcal{F}$ a.s. $\forall t$, which makes the second term in (12) vanish. Conversely, suppose that the policy in consideration is infeasible. This together with the fact that rewards are bounded almost surely makes (11) yield $V^\pi(s) = -\infty$, which is the same value attained by both terms in (12). ■

The preceding result implies non-trivial consequences. If the learning agent can interact with the environment and have access to rewards R_{t+1} and queries of whether a transition has been safe (i.e. queries of the type $\mathbb{I}_{\{(S_t,A_t,S_{t+1})\in\mathcal{F}\}}$), then it can independently learn both $Q^\pi(s,a)$ and $B^\pi(s,a)$. Learning (and improving) B^π is of great utility, since infeasible state-action pairs are readily coded as $-\infty$ in the corresponding function. This is further discussed in the next remark.

Remark 4 (Properties of the optimal barrier action-value function B^*)

$$B^*(s,a) = \max_\pi B^\pi(s,a) = \max_\pi \mathbb{E}_\pi \left[-\sum_{t=0}^{\infty} \mathbb{I}_{\{(S_t,A_t,S_{t+1})\in\mathcal{F}\}} \;\middle|\; S_0 = s, A_0 = a \right] \tag{13}$$

It is evident that for any policy π the entries of $B^\pi(s,a)$ will either be 0 or $-\infty$. Having $B^*(s,a) = -\infty$ means that if starting at s with action a and then following any policy, there is an (albeit small) non-zero probability that an unsafe event will be encountered.
Conversely, if $B^*(s,a) = 0$ then following the optimal policy guarantees we will always ensure constraints while starting from state s and action a. Furthermore, if B^* is available, then any policy that chooses an action such that $B^*(s,a) = 0$ is a feasible policy for that state. If, on the other hand, the policy chooses an action which leads to $B^*(s,a) = -\infty$, then that policy can be deemed unsafe immediately, and can be discarded. Notice, however, that this is a particular property of the optimal function B^*: for a sub-optimal π, B^π can be padded with lots of $-\infty$ in places where B^* has zeros. Intuitively, this would mean that the sub-optimal policy starting from (s,a) makes unsafe transitions along its trajectory. In this sense, having access to the optimal action-value function B^* helps constraint the search of any other known algorithms (such as Q-Learning) to only feasible policies. This becomes a joint work of optimizing the return (using one of many possible techniques), while learning the feasible set of actions at the same time.

3. Almost sure constraints

In this section we model the probability of satisfying the constraints, laying the ground to construct the algorithm that guarantees safety almost surely. In our way there we connect to the state of the art on safe learning with cumulative expected constraints.

In this direction, and given the triplet (S_t, A_t, S_{t+1}), consider the auxiliary random variable $D_{t+1} \in \{0,1\}$ which indicates if $(S_t, A_t, S_{t+1}) \in \mathcal{F}$, that is, $D_{t+1} = 1$ if $(S_t, A_t, S_{t+1}) \in \mathcal{F}$ and zero otherwise. Intuitively, D_{t+1} indicates if there is *damage* in the transition from S_t to S_{t+1}, and it is related to the indicator function (3) through $D_{t+1} = \exp\left(-\mathbb{I}_{\{(S_t,A_t,S_{t+1}\in\mathcal{F})\}}\right)$.

Accordingly, the conditional transition probability model is augmented to incorporate this new variable, so that $p(s', d|s, a)$ represents the probability of evolving from state s to state s' facing damage d, under action a. We further make the following assumptions on $p(s', d|s, a)$ that will allow us to learn in episodes. We assume that the trajectory ends when there is damage, which is modeled by moving to an absorbent terminal state s_D. This state is reached the first time there is damage $D_{t+1} = 1$, yielding $D_t = 0$ thereafter. Specifically, we set the following conditions on the transition probabilities

$$P(S_{t+1} = s_D, D_{t+1} = 0 | S_t \neq s_D, A_t = a_t) = 0 \tag{14}$$

$$P(S_{t+1} \neq s_D, D_{t+1} = 1 | S_t \neq s_D, A_t = a_t) = 0 \tag{15}$$

$$p(S_{t+1} = s_D, D_{t+1} = 0 | S_t = s_D, A_t = a_t) = 1 \tag{16}$$

The first and second conditions state that the transition to the terminal state s_D happens if and only if there is damage $D_{t+1} = 1$, while the third condition ensures that s_D is absorbent with $D_{t+1} = 0$ thereafter. Next, we consider the probability of violating the constraints at some point through an entire trajectory starting from $S_0 = s$, with initial action $A_0 = a$ and selecting subsequent actions according to a particular policy π.

$$q_D^\pi(s,a) := P\left(\bigcup_{t\geq 0}\{(S_t, A_t, S_{t+1}) \notin \mathcal{F}\}\Big| S_0 = s, A_0 = a\right) = P\left(\bigcup_{t\geq 0}\{D_{t+1} = 1\}\Big| S_0 = s, A_0 = a\right) \tag{17}$$

We also define the expected cumulative damage starting from $S_0 = s,\ A_0 = a$ and following policy π as

$$d^\pi(s) := \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} D_{t+1} = 1 | S_0 = s, A_0 = a\right] \tag{18}$$

It can be shown that (17) and (18) coincide (Paternain et al., 2019). Indeed, since conditions (14)-(16) ensure that damage D_{t+1} equals one at most once in the entire trajectory, the union in (17) is disjoint, and we can write

$$q_D^\pi(s,a) = \sum_{t=0}^{\infty} P\left(D_{t+1} = 1 | S_0 = s, A_0 = a\right) \tag{19}$$

$$= \sum_{t=0}^{\infty} \mathbb{E}_\pi\left[D_{t+1} = 1 | S_0 = s, A_0 = a\right] = d^\pi(s,a) \tag{20}$$

The next step is to link $q_D(s,a)$ and $d^\pi(s,a)$ with the barrier function $B^\pi(s,a)$ in (8). In the next statement we consider a policy π which satisfies all constraints almost surely when starting from $S_0 = s$ and $A_0 = a$, so that $q_D^\pi(s,a) = 0$ and prove that for such a safe policy $B^\pi(s,a) = 0$.

Theorem 5 *Under conditions (14)-(16) $q_D^\pi(s,a) = 0$ iff $B^\pi(s,a) = 0$.*

Proof *Starting from (19), and with the probabilities being non negative, we can write*

$$\begin{aligned} q_D^\pi(s,a) = 0 &\iff P(D_{t+1} = 1 | S_0 = s, A_0 = a) = 0,\ \forall t \geq 0 \\ &\iff P((S_t, A_t, S_{t+1}) \in \mathcal{F} | S_0 = s, A_0 = a) = 0,\ \forall t \geq 0 \\ &\iff \mathbb{E}_\pi\left[\mathbb{I}_{\{(S_t,A_t,S_{t+1})\in\mathcal{F}\}} | S_0 = s, A_0 = a\right] = 0,\ \forall t \geq 0 \iff B^\pi(s,a) = 0 \end{aligned}$$

The second equivalence follows from the definition of D_{t+1}, the third one from the definition of $\mathbb{I}_{\{\cdot\}}$ with the limit in (4), and the fourth one from linearity and $\mathbb{I}_{\{\cdot\}}$ being non-positive.

■

Finally we take advantage of the identity between $d^\pi(s,a)$ and $q_D^\pi(s,a)$ to derive a Bellman equation for the probability of being safe.

Theorem 6 (Bellman equation for q_D^π) *Under conditions* (14)-(16) *the conditional probability of violating the constraints satisfies* $q_D^\pi(s,a) = \mathbb{E}\left[D_{t+1} + q_D^\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a\right]$.

Proof We can set the first term of (20) apart and identify the conditional expectation of D_{t+1} in it

$$q_D^\pi(s,a) = \sum_{\tau=t}^{\infty} P(D_{\tau+1} = 1|S_t = s, A_t = a) \tag{21}$$

$$= P(D_{t+1} = 1|S_t = s, A_t = a) + \sum_{\tau=t+1}^{\infty} P(D_{\tau+1} = 1|S_t = s, A_t = a) \tag{22}$$

By setting $Z_t = (S_t, A_t)$ to reduce notation, we write the second term in as

$$\sum_{\tau=t+1}^{\infty} P(D_{\tau+1} = 1|S_t = s, A_t = a) = \sum_{\tau=t+1}^{\infty} \sum_{s'} \sum_{a'} P\left(D_{\tau+1} = 1, Z_{t+1} = (s', a')|Z_t = (s,a)\right)$$

$$= \sum_{\tau=t+1}^{\infty} \sum_{s'} \sum_{a'} P(D_{\tau+1} = 1|Z_{t+1} = (s', a'), Z_t = (s,a)) P(Z_{t+1} = (s', a')|Z_t = (s,a)) \tag{23}$$

$$= \sum_{\tau=t+1}^{\infty} \sum_{s'} \sum_{a'} P(D_{\tau+1} = 1|Z_{t+1} = (s', a')) P(Z_{t+1} = (s', a')|Z_t = (s,a)) \tag{24}$$

$$= \sum_{s'} \sum_{a'} q_D^\pi(s', a') P(Z_{t+1} = (s', a')|Z_t = (s,a)) = E[q_D^\pi(Z_{t+1})|Z_t = (s,a)] \tag{25}$$

where we used the Markov property of the transition probability for dropping Z_t in (23). ■

The Bellman recursion in Theorem 6 lets us compute the probability of being unsafe recursively, as the expected damage after the first transition, plus the average probability of damage starting from the next state. As we want to know if $q_D^\pi(s,a) = 0$, we can stop if there is damage after the first transition, and otherwise step to the next state and repeat. This is the idea behind the barrier-learning Algorithm in next section .

4. Assured reinforcement learning

As introduced in previous sections, our goal is to learn a safe policy π that maximizes the cumulative reward and finds trajectories that satisfy the constraints at all time steps almost surely. This safety condition on the policy can be described by the probability $q_D^\pi(s,a)$ being null for all s in the set of starting points $\mathcal{S}_0$, or equivalently $B^\pi(s,a) = 0$ for all $s \in \mathcal{S}_0,\ a \in \mathcal{A}$. As before, we assume that each state transition yields a reward R_{t+1} and a damage index D_{t+1}, which are distributed according to the transition probability $p(s'd, r|s, a)$. Algorithm 1 specifies how to use this data in order to learn the set of safe policies. The Barrier function $B(s,a)$ is updated incorporating the new information about safety provided by the damage index D_{t+1} at each transition.
Then, Algorithm 2 uses the decomposition of Theorem 3 to embed this safety information in the q-function $Q(s,a)$. Hence, we identify that the condition $Q(s,a) = -\infty$ is equivalent to $B(s,a) = -\infty$ and this propagates information about safety from successor states. Indeed, $Q(s,a) = -\infty$ flags the pair (s,a) as one leading, with nonzero probability, to imminent damage or to a next state where all further actions lead to damage. Such a pair (s,a) does not comply with our requirement

Data: B-function (initialized as all-zeroes)
Input: (s, a, s', d) tuple
Output: Evaluated barrier-function $B(s, a)$
$B(s, a) \leftarrow B(s, a) + \log(1 - d) + \max_{a'} B(s', a')$
return $B(s, a)$
Algorithm 1: Barrier-learner

of satisfying the constraints almost surely along the trajectory, and thus it is discarded from the set of state-action pairs that ensure feasibility. This is obtained by defining safe sets $\mathcal{B}(s)$ which specify the permitted actions at state s. If an action is deemed *unsafe* (by inspecting $Q(s, a) = -\infty$) then it is taken out from $\mathcal{B}(s)$. In this sense, at every time step we are enforcing that the set $\mathcal{B}(s) = \text{argmax}_a B(s, a)$.

Data: Starting state distribution μ, discount γ, exploration ϵ, learning rate η
Result: Optimal action-value functions Q^* and B^*
Initialize safe sets $\mathcal{B}(s) = \mathcal{A} \quad \forall s$
$Q(s, a) = B(s, a) = 0 \quad \forall\, (s, a)$
for $episode = 0, 1, \ldots$ **do**
  Draw $s_0 \sim \mu$
  Set $s \leftarrow s_0$
  if $\mathcal{B}(s) = \emptyset$ **then**
    continue
  end
  $k \leftarrow 0$
  while $s \neq s_D$ **do**
    Draw action $a \sim \pi$ $\quad$ (e.g. ϵ-greedily w.r.t. $Q(s, \cdot)$, with $a \in \mathcal{B}(s)$)
    Apply a, observe (s', r, d)
    $B(s, a) \leftarrow$ `barrier_learner`(s, a, s', d)
    $Q(s, a) \leftarrow B(s, a) + (1 - \eta)Q(s, a) + \eta\big(r + \gamma \max_{a'}(Q(s', a')\big)$
    if $Q(s, a) = -\infty$ **then**
      $\mathcal{B}(s) \leftarrow \mathcal{B}(s) \setminus \{a\}$
    end
    $k \leftarrow k + 1$
    $s \leftarrow s'$
    if $\mathcal{B}(s) = \emptyset$ **then**
      break
    end
  end
end
Algorithm 2: Episodic Assured Q-Learning

This in turn narrows the set of feasible policies that are contenders for being optimal in terms of the reward. Notice that $Q(s, a)$ also incorporates the reward, and coincides with the standard q-function when the barrier is null, so that our Assured Q-Learning algorithm learns to optimize the reward while assuring feasibility on the fly.

Remark 7 (Avoiding unsafe states) *Keeping track of the sets of safe actions $\mathcal{B}(s)$ in Algorithm 2 implies that we can identify unsafe states as those with $\mathcal{B}(s) = \emptyset$. This gives us a second level of safety during training, as we prematurely stop an episode if we reach one of such unsafe states. In this way, we prevent what we already identified as unavoidable future damage, and assume $d = 1$ for updating $B(s, a)$ and $Q(s, a)$ without actually following the trajectory until the actual damage occurs. Notice that this propagates backwards, so that we can identify unsafe states many steps ahead of where actual damage would occur.*

5. Numerical Experiments

In this section we compare the performance of Algorithm 2 against standard Q-Learning. We start by introducing the CMDP: a simple grid-world or maze that the agent must learn to navigate. We finish by comparing the learning curve of both algorithms and the number of times constraints have been violated during training.

5.1. Experimental setup

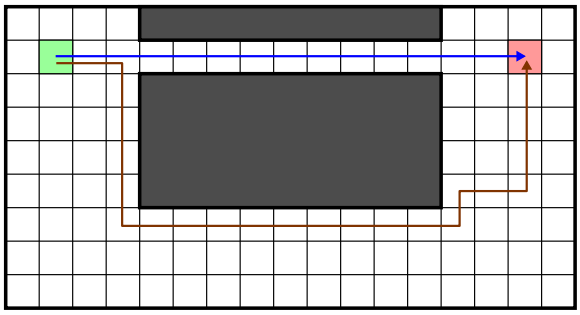

Figure 1: Example grid-world. The agent starts at the green square and if he reaches the red "goal" achieves a reward of $+10$. Moving around has null reward and bumping into walls is unsafe. With discounting ($\gamma < 1$), the optimal trajectory is the one that goes through the corridor (shown in blue). During learning, the corridor is potentially unsafe since many actions lead to a wall. As a result, some agents converge to sub-optimal trajectories that avoid the corridor (as the one shown in brown).

We test our Algorithm on a simple grid-world shown in Figure 1. The agent must learn to navigate the maze, starting from the green "start" state and finishing in the red "goal" state. Available actions are *up, down, right* and *left*, with which the agent transitions to the corresponding neighbouring state, if possible. If the agent bumps into a wall —which is considered *unsafe*— the episode terminates and the new episode begins in the preceding state. Rewards are 0 for moving around and $+10$ if the goal is reached.
We deploy Algorithm 2 and see how it compares against standard Q-Learning. To provide a fair comparison, when doing standard Q-Learning bumping into walls will have a reward $r = -\infty$. In any case, by specifying a discount factor $\gamma < 1$ the optimal policy will be one that learns to reach the goal in the least number of steps. This policy is the one that goes through the narrow corridor, which

is tricky to learn since 2 out of the 4 actions available lead to a wall inside the corridor. Suboptimal policies often learn to avoid the corridor and take the long way around (see Figure 1).

5.2. Results

We define the *total episode length* as the number of steps it takes the agent to reach the goal while starting from the "start", and consider this to be a proxy for learning. If after 100 steps the agent hasn't reached the goal, the episode is reset and the agent is taken to its starting position. We train 1000 instances of each agent on the gridworld described, setting $\gamma = 0.9,\ \epsilon = 0.1, \eta = 0.1$. Figure 2 on the left shows the total episode length during training. As can be seen, the learning curve is similar for both Algorithms. The interesting thing to check is the number of *constraint violations* during training, which is shown by Figure 2 on the right. It depicts the cumulative constraint violations while learning —that is, the number of times the agent has bumped into a wall. Notice that after some time the Assured agent's constraint violations plateau, while the Standard Q-Learning agent keeps taking unsafe actions. It is worth recalling that our Assured agent discards actions when he learns they lead to unsafe transitions, whereas the Q-Learning agent might still take an unsafe action (even if it has a corresponding value $Q(s, a) = -\infty$) as a consequence of the ϵ-induced exploration.

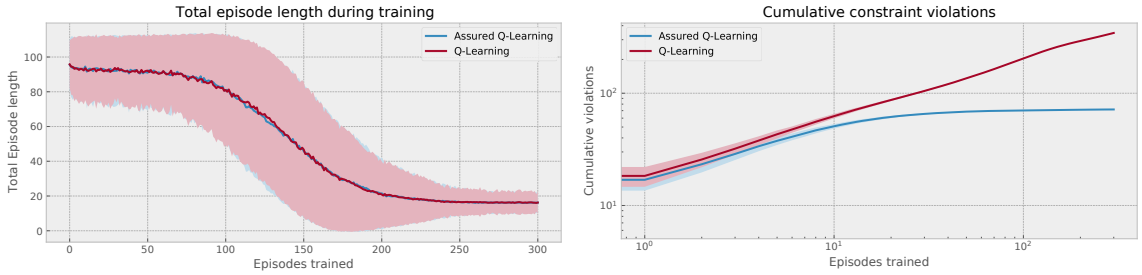

Figure 2: On the left: episode length during training, i.e., how many steps the agent takes to reach the goal. On the right: log plot of the cumulative constraint violations during training, i.e., the number of times the agent has bumped into a wall. Solid marks correspond to the average value over $N = 1000$ independent trials, while shaded regions are the average value $\pm\sigma/\sqrt{N}$, with σ the sample deviation. Notice that the learning curves (left) for both algorithms are similar. However, the Assured agents learn to avoid walls, why the other agents do not (right).

6. Conclusions

We addressed the problem of safe learning on a model-free framework, where optimal policies and constraints are hinted by the realization of rewards and damages. In this context, we defined value functions that can be decomposed in a constraint-abiding term and a term that optimizes the return. This barrier function can be inferred from damage data, and is crucial to learn the set of safe policies. Our proposed barrier-learning algorithm can be combined with Q-Learning, and learn to optimize rewards and discard unsafe policies simultaneously on the fly.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained Policy Optimization. *arXiv*, cs.LG, 2017.

Eitan Altman. *Constrained Markov Decision Process*, volume 7. CRC Press, 1998. ISBN 9780849303821. URL http://www-sop.inria.fr/members/Eitan.Altman/PAPERS/h.pdf.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, 134 (1-2):57–83, 2002.

Richard Cheng, Gabor Orosz, Richard M Murray, and Joel W Burdick. End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks. *arXiv*, 2019.

Dongsheng Ding, Xiaohan Wei, Zhuoran Yang, Zhaoran Wang, and Mihailo R Jovanović. Provably Efficient Safe Exploration via Primal-Dual Policy Optimization. *arXiv*, 2020.

David A Ferrucci. Introduction to "this is watson". *IBM Journal of Research and Development*, 56 (3.4):1–1, 2012.

Javier Garcia and Fernando Fernandez. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research*, 16(1):1437—1480, 2015.

Peter Geibel. Reinforcement Learning for MDPs with Constraints. volume 4212 of *Machine Learning: ECML 2006*, pages 646–653, 2006. ISBN 9783540453758. doi: 10.1007/11871842_63.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 1–12, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

James A Nichols, Hsien W Herbert Chan, and Matthew AB Baker. Machine learning: applications of artificial intelligence to imaging and diagnosis. *Biophysical reviews*, 11(1):111–118, 2019.

Santiago Paternain, Miguel Calvo-Fullana, Luiz F O Chamon, and Alejandro Ribeiro. Safe Policies for Reinforcement Learning via Primal-Dual Methods. *arxiv*, 2019. URL https://arxiv.org/pdf/1911.09101.pdf.

Jonathan Schaeffer, Robert Lake, Paul Lu, and Martin Bryant. Chinook the world man-machine checkers champion. *AI Magazine*, 17(1):21–21, 1996.

Jonathan Schaeffer, Markian Hlynka, and Vili Jussila. Temporal difference learning applied to a high-performance game-playing program. In *Proceedings of the 17th international joint conference on Artificial intelligence-Volume 1*, pages 529–534, 2001.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

Akifumi Wachi and Yanan Sui. Safe Reinforcement Learning in Constrained Markov Decision Processes. *arXiv*, 2020.

Mario Zanon and Sébastien Gros. Safe Reinforcement Learning Using Robust MPC. *arXiv*, 2019.