Reinforcement Learning With Imperfect Safety Constraints

Jin Woo Ro, Gerald Lüttgen, Diedrich Wolter

University of Bamberg, Germany

{jin.ro, gerald.luettgen, diedrich.wolter}@uni-bamberg.de

Abstract

An obvious approach for ensuring safety in applying deep reinforcement learning is to use it with safety monitors. An ideal monitor captures all unsafe system states, but this is hardly possible in practical systems due to uncertainties in the environment, complicated system dynamics, and limited environment knowledge. Even with a perfect monitor, dynamic adjustments may still be required to account for system changes such as ageing and damage. Therefore, what to do if the monitors are imperfect? This paper proposes an approach for estimating the undetected states of imperfect monitors in conjunction with deep Q-learning. A new Q-value function is developed to capture the target states effectively, and a system is designed to manage control and near-critical safety separately. Our experiments show that the approach proposed in this paper better considers unsafe states than a classic learning-based approach and achieves a faster convergence to the maximum control reward.

Introduction

Deep reinforcement learning (DRL) excels when it is used to control a continuous system in complex and uncertain environments. Thus, cyber-physical systems (CPSs) such as robotics and intelligent transportation systems can be the most appropriate applications for DRL. However, the deployment of DRL in real systems is currently facing difficulties of ensuring safety requirement of CPS, where a set of constraints needs to be always satisfied to prevent catastrophic system failures. In particular, some control actions selected by DRL can violate the safety constraints. To overcome this problem, recent studies utilise a safety monitor in conjunction with DRL to detect safety violations and block the causing control action (Junges, Torfah, and Seshia 2021). If the safety monitor perfectly captures all unsafe system states and the actions leading to them, DRL can be enforced to select only safe control actions (Mirchevska et al. 2018).

However, a perfect safety monitor is generally not realistic in many CPS due to the complexities and uncertainties in the system environment. Furthermore, there maybe only limited known information about the environment (e.g., robots in an unknown environment (Kantaros et al. 2020)). Conservative approaches assume the worst-case scenario and over-approximate the unsafe states at the price of degraded system performance. However, finding the worst-case scenario itself is generally non-trivial, and there may be unrealistic assumptions. Even if there is a perfect monitor, it may become ineffective after some time due to external factors, such as system ageing and damage. To fill the gap between ideal and realistic monitors, there is a need for learning components dedicated to capturing the undetected unsafe states.

This paper proposes an approach for capturing the undetected states of imperfect monitors using deep Q-learning. The idea of detecting unsafe states using a learning model is not new; however, existing studies (Fisac et al. 2018; Berkenkamp et al. 2018) assume that the perfect safety constraints are known a priori. They trust that the safety interpretation given by the constraints are always correct and use them directly in the learning process. However, the imperfect monitor can produce false-negative outputs (i.e., the system is unsafe but interpreted as safe). Thus, we differentiate the cases where one can or cannot trust the monitor's interpretation. For this, we define a new Q-value function that is suitable for finding the undetected unsafe states. We mathematically prove that the proposed new Q-value function can eventually find all undetected unsafe states. Lastly, through benchmarking based on an inverse pendulum example, we compare the deep Q-learning performance with and without our safety estimation. The results show that safety violations can be reduced noticeably, and also that the learning converges faster to the optimal control policy.

Motivating Example

Consider the inverse pendulum example shown in Figure 1. The bar can rotate either clockwise or anticlockwise about a fixed anchor. The gravity g and the force F are the two factors affecting the rotation. Two variables can represent the instantaneous state of the bar: the angle θ and the angular velocity ω (i.e., two-dimensional state space).

We assume that the safety constraints cannot be specified completely, and a safety monitor is constructed as a finite state machine as shown in Figure 2a. The constraint $|\theta| \geq 0.35$ is essentially our guess for identifying two states: the Upright state (i.e., *safe*) and the Fallen state (i.e., *unsafe*). The monitor outputs 1 or -1 to indicate safe and unsafe, re-

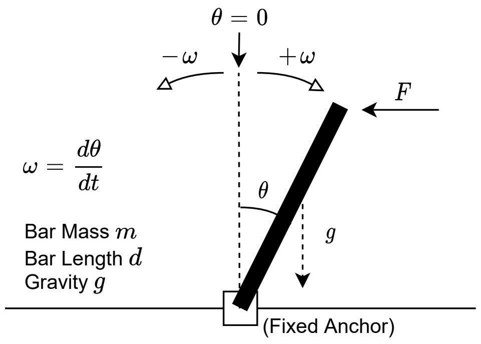


Figure 1: Inverse pendulum system, where a bar rotates about the fixed anchor. We assume that $m = 2\text{kg}$, $d = 1\text{m}$, $g = -9.81\text{ms}^{-2}$ and θ is given in radian. The force $|F| = 2\text{N}$ is a constant magnitude horizontal force applied at the tip of the bar. The direction of F can be controlled (left: $F = -2\text{N}$ or right: $F = 2\text{N}$) to keep the bar upright.

spectively.

There is an equilibrium point where the gravity force and F cancel out: $Fd\cos\theta + \frac{mgd}{2}\sin\theta = 0$. Based on our parameter values, the equilibrium is at $\theta = 0.201$. If θ exceeds this point, the angular acceleration due to the gravity becomes larger than that of F, and the bar will always accelerate towards the ground. Also, we need to consider the effect of instantaneous angular velocity ω. A large ω cannot be instantaneously reduced to zero (i.e., deceleration takes time). Therefore, the bar can still exceed the equilibrium point even if the force F is applied to the opposite direction.

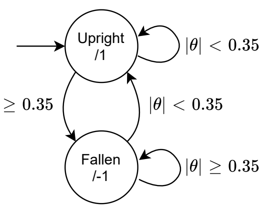
Figure 2b shows the state space of the inverse pendulum system. First, the areas with labels ① and ② represent the set of states that satisfy $|\theta| \geq 0.35$ (i.e., the Fallen state). The areas ③, ④, and ⑤ represent the possible Upright states. However, the states in ③ and ④ always converge to ① and ②, respectively, due to either exceeding the equilibrium angle or a large angular velocity. Although ③ and ④ also need to be considered as unsafe states, the imperfect safety monitor actually recognises them as safe (i.e., false-negative detections). In this paper, we call such states (that are incorrectly recognised) as *hidden unsafe states*, and our objective is not only to find them but also to identify the actions leading to them.

Problem Formulation

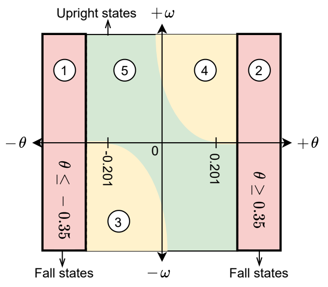
In this section, we formalise the hidden unsafe states detection problem, given that there is a safety monitor that can provide the ground truth about the safe and unsafe states. We start with a recap of markov decision process (MDP), and formally define the safety monitor. Finally, we define our problem.

Markov Decision Processes

In reinforcement learning, a control agent interacts with the environment by reading the environment state and performing an action that updates that state. Such a system is typically modelled as a MDP (Sutton and Barto 2018).



(a) Monitor



(b) State space

Figure 2: The safety monitor for the inverse pendulum example, and the visualisation of safe states, unsafe states, and hidden unsafe states as regions in the two-dimensional state space.

Definition 1 (Markov Decision Process)**.** A MDP is a tuple $\mathcal{M} = \langle S, A, P, R, \gamma \rangle$, where:

- S is the set of states, i.e., the state space.
- A is the set of actions, i.e., the action space.
- $P(s'|s, a)$ is the probability distribution over state s' given state s and action a.
- $R(s, a)$ is the immediate reward when taking action a from state s.
- $\gamma \in [0, 1]$ is the discount factor.

In every discrete time step t, the system control takes an action $a_t \in A$ from a state $s_t \in S$ based on policy $\pi : S \to A$. Then, the system reaches the next state $s_{t+1} \in S$ and the agent receives the reward $R(s_t, a_t)$. The action-value function $Q(s, a)$ – or simply called Q-value – defines the value of taking action a from state s:

$$Q(s, a) = \mathbb{E}_s^\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_t = s, a_t = a] \quad (1)$$

where $\mathbb{E}_s^\pi$ is the expected value of following policy π from state s. Intuitively, the returned value from Equation 1 is the expected sum of the rewards over all t. The Bellman optimality equation for the optimal action-value function $Q^*(s, a)$ that yields the maximum cumulative reward is

$$Q^*(s, a) = \mathbb{E}[R(s, a) + \gamma \max_{a'} Q^*(s', a')] \quad (2)$$

where s' is the successor state and a' is an action that can be taken from state s'. In this paper, we consider a MDP with a continuous state space and a discrete action space, as in the case with the inverse pendulum system example.

Safety Monitors

Generally, a safety monitor is a finite state machine. In our study, the monitor takes the inputs s and a from the MDP and outputs value 1 to indicate safe or -1 to indicate unsafe.

Definition 2 (Safety Monitor)**.** A safety monitor is a deterministic Moore machine defined as a tuple $\mathcal{A} = \langle L, l_0, I, \delta, G \rangle$, where:

- L is the set of finite discrete locations.

- l_0 is the singleton initial location.
- I is the set of input variables.
- $\delta : L \times 2^{AP(I)} \rightarrow L$ is the transition function, where $AP(I)$ is a set of boolean expressions over the input variables.
- $G : L \rightarrow \{-1, 1\}$ maps each location $l \in L$ to either -1 or 1. We denote the set of safe locations by $L_{safe} = \{l \in L | G(l) = 1\}$. Similarly, the set of unsafe locations is $L_{unsafe} = \{l \in L | G(l) = -1\}$.

If there are multiple monitors in the system, we can represent them as a single monitor whose output is 1 if the outputs of all monitors are 1, otherwise, -1. Therefore, in this paper, we can simply assume that there is exactly a single safety monitor in the system.

At any time instant, the state of the entire system including the monitor can be expressed as a state pair (s, l), where $s \in S$ is the state of the MDP and $l \in L$ is the location of the monitor. Also, an infinite path can be generated by following the policy π starting from (s_0, l_0):

$$p^{\pi}_{(s_0,l_0)} = (s_0, l_0) \xrightarrow{a_0} (s_1, l_1) \xrightarrow{a_1} (s_2, l_2) \xrightarrow{a_2} ... \quad (3)$$

In the path, a state pair (s, l) is identified as unsafe if $l \in L_{unsafe}$. Also, an action a is identified as an unsafe action in (s, l) if there exist a transition $(s, l) \xrightarrow{a} (s', l')$ where $l' \in L_{unsafe}$.

Hidden Unsafe States Detection Problem

We denote by S_{unsafe} the set of all state pairs (s, l) satisfying $l \in L_{unsafe}$. Similarly, S_{safe} denotes the set of all state pairs (s, l) satisfying $l \in L_{safe}$.

Definition 3 (Hidden Unsafe States Detection Problem)**.** Let $S_{hidden} \subseteq S_{safe}$ represent a set of hidden unsafe states, where all possible paths starting from a state $(s, l) \in S_{hidden}$ always eventually reach a state $(s', l') \in S_{unsafe}$. Then, the hidden unsafe states detection problem is to obtain the actual unsafe set

$$S^*_{unsafe} = S_{hidden} \cup S_{unsafe} \quad (4)$$

by finding S_{hidden} based on S_{unsafe} which is known a priori.

Intuitively, S_{unsafe} is what the imperfect monitor recognises as unsafe, and S^*_{unsafe} is what the perfect monitor should recognise as unsafe. Thus, S_{hidden} is the gap between the perfect and imperfect monitors, and this is what we aim to find in our approach.

System Model

This section presents our approach to estimate S^*_{unsafe} and block the actions that immediately lead to such states. We first give an overview of our system model.

Model Overview

The system overview is shown in Figure 3. First, the environment passes the state s and the reward $R(s, a)$ to the controller based on action a. Such a feedback loop between environment and controller is the standard design of reinforcement learning. State s and action a are also passed to the

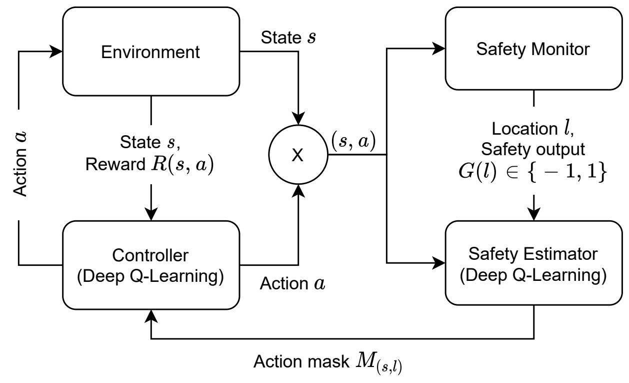

Environment
State s
Safety Monitor
State s, Reward $R(s, a)$
X
(s, a)
Location l, Safety output $G(l) \in \{-1, 1\}$
Action a
Controller (Deep Q-Learning)
Action a
Safety Estimator (Deep Q-Learning)
Action mask $M_{(s,l)}$

Figure 3: System Model Overview

safety monitor for updating the monitor location and generating the output $G(l)$. The *safety estimator* is an additional learning component in the system. It aims to learn S^*_{unsafe} and produces the output for indicating which actions need to be blocked. It requires information s, a, and l to learn the system path (Equation 3), and also $G(l)$ to know what is safe and unsafe. The action mask output $M_{(s,l)}$ is an array of binary values with size equal to the number of control actions:

$$M_{(s,l)} = [m_0, m_1, ..., m_n] \quad (5)$$

Intuitively, each binary value is associated with a certain action. For example, $m_i = 0$ means that action $a_i \in A$ needs to be blocked. In this way, we can selectively block unsafe actions.

Controller

The controller is based on the deep Q-Learning method proposed in (Mnih et al. 2015). Generally, the optimal Q-value $Q^*(s, a)$ in Equation 2 is not known to the controller. Hence, we use a neural network to estimate $Q^*(s, a)$ with $Q(s, a; \theta)$, where θ is the parameters of the neural network (i.e., weights and bias). Note that θ is the standard notation for neural network parameters, and it should not be confused with the angle θ in Figure 1.

For each discrete time step, the *experience replay buffer* U stores the controller's experience (s, a, r, s'), where s is current the state, a is the action, r is the reward, and s' is the next state. Note that the use of replay buffer is based on the work in (Mnih et al. 2015). By using the data stored in buffer U, the neural network is trained using the following loss function:

$$Loss(\theta) = \mathbb{E}_{(s,a,r,s') \sim U} \left[\left(Q_{target} - Q_{pred} \right)^2 \right] \quad (6)$$

where,

$$Q_{target} = r + \gamma \max_{a'} Q(s', a'; \theta^-) \quad (7)$$

$$Q_{pred} = Q(s, a; \theta) \quad (8)$$

where γ is the discount factor. Here, $(s, a, r, s') \sim U$ means that the experience data are sampled from buffer U. The term $(Q_{target} - Q_{pre})^2$ in Equation 6 represents the squared error between Q_{pred} (the prediction of the Q-value) and Q_{target}

(the target Q-value). In the equation of Q_{target}, symbol θ^- represents the previous value of θ. More precisely, we update $\theta^- = \theta$ every k steps, but during the k steps, θ^- is kept fixed. Note that k is a hyperparameter used to control the learning process based on (Mnih et al. 2015). Finally, the neural network learns to minimise the loss function.

The ϵ-greedy action selection method can be performed with the action mask $M_{(s,l)}$. Through the neural network, we can obtain the distribution of Q-values over the action space:

$$Q^s = [Q(s, a_0), Q(s, a_1), ..., Q(s, a_n)] \tag{9}$$

The array Q^s contains the Q-values of all actions. Next, we apply the action mask to obtain the array of final Q-values Q^s_{final}:

$$\begin{aligned} Q^s_{final} &= Q^s \otimes M(s, l) \\ &= [Q(s, a_0), ..., Q(s, a_n)] \otimes [m_0, ..., m_n] \\ &= [Q(s, a_0)m_0, ..., Q(s, a_n)m_n] \end{aligned} \tag{10}$$

where $\otimes$ is the element-wise multiplication. For example, the multiplication of the first element of Q^s and $M(s, l)$ is the first element of Q^s_{final}. Because the action mask $M(s, l)$ contains binary values, the element-wise multiplication enforces the Q-values to either keep their values or become zero. During the action selection process, the actions with zero Q-values are ignored so that unsafe actions can be blocked from being selected. If Q^s_{final} only contains zeros, it means that all actions are unsafe. In this case, a predefined safe backup plan can be instantiated or the maximum Q-value in Q^s can be selected directly.

Safety Estimator

Similar to the controller that learns from the environment, the safety estimator learns from the safety monitor. Basically, $G(l)$ is the reward of the safety monitor. However, unlike $R(s, a)$ from the environment, $G(l)$ cannot be fully trusted because $G(l) = 1$ cannot differentiate between safe and hidden unsafe states. Though, we trust $G(l) = -1$.

Another important aspect is that the safety estimator aims to solve a classification problem rather than an optimisation problem. Therefore, the safety estimator does not need to learn to maximise the reward but to identify hidden unsafe states. Here, the cumulative reward value is not meaningful because we cannot know what rewards are added. For example, a few bad rewards may be overwhelmed by good ones. As such, the cumulative reward maximisation using the Bellman equation (Equation 2) is inappropriate for the goal of the safety estimation. Therefore, we define a different Q-value function for our safety estimator.

The Q-value $Q(s, l, a)$ is defined in Equation 11. We express $[x]^{u=k_1}_{l=k_2}$ to indicate that the value of x in the parenthesis is capped by the upper bound k_1 and the lower bound k_2. For example, $[5]^{u=1}_{l=-1} = 1$ and $[-2]^{u=1}_{l=-1} = -1$.

$$Q(s, l, a) = \left[G(l') + 2\gamma_s \frac{\sum\limits_{\forall a' \in A} Q(s', l', a')}{|A|} \right]^{u=1}_{l=-1} \tag{11}$$

where, $|A|$ is the number of actions. Basically, the sigma term calculates the average Q-value of all actions in (s', l'). Parameter γ_s can have two possible values as shown in Equation 12:

$$\gamma_s = \begin{cases} 0 & \text{if}\, G(l') = -1 \\ 1 & \text{if}\, G(l') = 1 \end{cases} \tag{12}$$

Lemma 1. The Q-value of an action that directly leads to an unsafe state is always -1.

Proof. We directly prove Lemma 1. Consider an arbitrary transition $(s, l) \xrightarrow{a} (s', l')$, where $(s', l') \in S_{unsafe}$. In this case, we have $G(l') = -1$ from the monitor, which is then used to get $\gamma_s = 0$ using Equation 12. By substituting $\gamma_s = 0$ and $G(l') = -1$ into Equation 11, we always have $Q(s, l, a) = [-1]^{u=1}_{l=-1} = -1$. ■

If $G(l') = -1$, the average value term in Equation 11 is eliminated by $\gamma_s = 0$, and the Q-value is just the value of $G(l')$. This reflects that we trust the case $G(l') = -1$. In contrast, the case of $G(l') = 1$ requires additional information to differentiate between the safe and hidden unsafe states. Therefore, the average term in Equation 11 is the key to identify hidden unsafe states. Intuitively, γ_s in our approach operates as a switch between two cases.

Lemma 2. The Q-value of an action that directly leads to a *hidden unsafe state* is always -1.

Proof. Consider a transition $(s, l) \xrightarrow{a} (s', l')$, where (s, l) and (s', l') are both members of S_{safe} so that $G(l) = G(l') = 1$. To identify (s', l') as a hidden unsafe state, we further assume that all actions a' taken in (s', l') directly lead to some unsafe states. In this case, we know that $Q(s', l', a') = -1$ based on Lemma 1. Furthermore, $\gamma_s = 1$ because $G(l') = 1$. Hence, Equation 11 simplifies to:

$$\begin{aligned} Q(s, l, a) &= \left[G(l') + 2\gamma_s \frac{\sum\limits_{\forall a' \in A} Q(s', l', a')}{|A|} \right]^{u=1}_{l=-1} \\ &= \left[1 + 2\gamma_s \frac{|A| \times (-1)}{|A|} \right]^{u=1}_{l=-1} \\ &= \left[1 - 2 \right]^{u=1}_{l=-1} \\ &= -1 \end{aligned} \tag{13}$$

Here, the sigma term $\sum_{\forall a' \in A} Q(s', l', a')$ is substituted with $|A| \times -1$ because all the Q-values from (s', l') are -1. Therefore, the Q-value of an action leading to a hidden unsafe state is -1. ■

If there is at least one safe action, we always have $\sum_{\forall a' \in A} Q(s', l', a') > |A| \times -1$, which makes the Q-value not equal to -1 but a larger value.

It is an important factor in reinforcement learning, whether the system reward converges to the optimal solution. In our case, the optimal solution is the actual set of unsafe states S^*_{unsafe} (Definition 3), and convergence means that our safety estimator can eventually detect all states in S^*_{unsafe} by finding the set S_{hidden} of hidden unsafe states.

Theorem 1. All hidden unsafe states and the actions leading to them are eventually detected by the safety estimator.

Proof. Consider an arbitrary path from (s_0, l_0) to (s_n, l_n) of length n:

$$(s_0, l_0) \xrightarrow{a_0} (s_1, l_1) \xrightarrow{a_1} ... \xrightarrow{a_{n-1}} (s_n, l_n) \qquad (14)$$

Based on the safety monitor, the unsafe states that produce $G(\cdot) = -1$ are known in a priori. Without loss of generality, we assume that (s_n, l_n) is an unsafe state so that $G(l_n) = -1$. By Lemma 1, the Q-value of the previous action $Q(s_{n-1}, l_{n-1}, a_{n-1})$ is -1 because it leads to an unsafe state. If the state (s_{n-1}, l_{n-1}) only has unsafe actions, the Q-value of its previous action $Q(s_{n-2}, l_{n-2}, a_{n-2})$ is -1 based on Lemma 2 because it leads to a hidden unsafe state. Similarly, if (s_{n-2}, l_{n-2}) only has unsafe actions, $Q(s_{n-3}, l_{n-3}, a_{n-3}) = -1$, and so on. This chain of Q-value relation uses an unsafe state as a reference point, and by inductively backtracking the path, we can find all hidden unsafe states. Because every hidden unsafe state is connected to unsafe states by definition, the safety estimator can eventually obtain S^*_{unsafe}. ■

Finally, the safety estimator's Q-value outputs are mapped into the action mask $M(s, l)$. If the Q-value is -1, the mask value is set to 0, otherwise, set to 1.

$$\forall m_a \in M(s,l),\ m_a = \begin{cases} 0 & \text{if } Q(s,l,a) = -1 \\ 1 & \text{otherwise} \end{cases} \qquad (15)$$

System Execution and Learning Process

A neural network is used to estimate the Q-value of the safety estimator. The estimated Q-value is denoted by $Q(s, l, a; \theta_s)$, where θ_s are the neural network parameters. The experience data $(s, l, a, G(l), s', l')$ is stored in two buffers U_s and U_v depending on the value of $G(l)$. If the value is 1, the experience data are stored in U_s, otherwise, in U_v.

If only a single buffer is used, keeping the unsafe experience data in the buffer is difficult as they are frequently overwritten by new experience data. In particular, as the system control becomes better, the new data are most likely to be a safe state experience. Since the neural network training relies on the training data quality, biased data leads to biased learning (i.e., only learning the safe states) that degrades the safety estimation accuracy. Hence, to consistently provide unsafe states data, we propose using double buffers so that one buffer (U_v) is dedicated for storing unsafe state data. In the experiment results section below, the safety estimator performances with respect to a single buffer case and a double buffer case are compared.

The overall system execution and the learning process are shown in Algorithm 1. We first initialise the buffers and the neural network with random parameters θ_s. An episode consists of T discrete time steps (Line 4). In each discrete time step, the current states s_t and l_t are used to compute Q^{s_t} and $M(s_t, l_t)$. The action a_t is chosen based on the ϵ-greedy method after masking. We execute the action and get the next state s_{t+1} and l_{t+1}. Based on $G(l_{t+1})$, the experience data $(s_t, l_t, a_t, G(l_{t+1}), s_{t+1}, l_{t+1})$ is stored in either U_s or U_v. $Minibatch$ is randomly sampled from both of the buffers. In our case, we sampled from U_s and U_v with a 1:1 ratio. Next, we iterate over each experience data in $Minibatch$. The target Q-value is decided by $G(l')$. The loss function computes the squared error between the target Q-value and $Q(s, l, a; \theta_s)$, and a gradient descent method is used to minimise the loss. Lastly, the controller is trained. One important note is that the safety estimator's Q-value needs to be restricted between -1 and 1. This can be done naturally by setting the output layer activation function. In our case, we use the hyperbolic tangent ($\tanh$) function.

Algorithm 1: Safety Estimator Learning

```
1:  Initialise $U_s$ and $U_v$ to capacity N
2:  Initialise a neural network $Q$ with random weights $\theta_s$
3:  **for** episode = 1, N **do**
4:      **for** t = 1, T **do**
5:          Get the current states $s_t$ and $l_t$
6:          Get $Q^{s_t}$ from the controller (Equation 9)
7:          Get $M_{(s_t,l_t)}$ based on $Q(s_t, l_t, \forall a \in A; \theta_s)$
8:          Compute $Q^{s_t}_{final}$ (Equation 10)
9:          **if** a random value between 0 and 1 $\geq \epsilon$ **then**
10:             $a_t \leftarrow \arg\max_a Q^{s_t}_{final}$
11:         **else**
12:             $a_t \leftarrow$ random non-zero element in $Q^{s_t}_{final}$
13:         **end if**
14:         Execute $a_t$ and get the next states $s_{t+1}$ and $l_{t+1}$
15:         Get $G(l_{t+1})$ from the safety monitor
16:         **if** $G(l_{t+1}) = 1$ **then**
17:             Store $(s_t, l_t, a_t, G(l_{t+1}), s_{t+1}, l_{t+1})$ in $U_s$
18:         **else if** $G(l_{t+1}) = -1$ **then**
19:             Store $(s_t, l_t, a_t, G(l_{t+1}), s_{t+1}, l_{t+1})$ in $U_v$
20:         **end if**
21:         Minibatch $\leftarrow$ Sample($U_s \cup U_v$);
22:         **for** $(s, l, a, G(l'), s', l') \in$ Minibatch **do**
23:             **if** $G(l') = 1$ **then**
24:                 $q_{target} \leftarrow \left[ 1+2\frac{\sum_{\forall a' \in A} Q(s',l',a';\theta_s)}{|A|} \right]^{u=1}_{l=-1}$
25:             **else if** $G(l') = -1$ **then**
26:                 $q_{target} \leftarrow -1$
27:             **end if**
28:             $loss \leftarrow (q_{target} - Q(s, l, a; \theta_s))^2$
29:             Minimise $loss$ via a gradient descent method
30:             Train the controller using Equation 6
31:         **end for**
32:     **end for**
33: **end for**
```

Experimental Results

This section demonstrates the efficacy of our approach for the inverse pendulum example. We first explain the experiment setup, followed by the experiment results.

Experimental Setup

The benefit of our approach can be effectively shown by directly comparing the performance of deep Q-learning with and without the safety estimator. Furthermore, we consider the safety estimator with a single buffer and with double buffers as separate models to be evaluated. Therefore, three models are evaluated and compared.

1. Conventional deep Q-learning (DQN) (Mnih et al. 2015).
2. DQN with our safety estimator (single buffer)
3. DQN with our safety estimator (double buffer)

We are interested in the following observation criteria:

- How frequently safety is violated.
- How fast control reward reaches the optimal solution (i.e., convergence).
- The evolution of estimated set of hidden unsafe states over episodes.

Lastly, we describe the inverse pendulum experiment setup. Each episode terminates if 200 discrete steps have passed or the bar is considered as fallen down based on the safety monitor in Figure 2a. The reward is 1 for each discrete time step when the bar is upright; otherwise, it is 0. Thus, the maximum cumulative reward of a single episode (or episode reward) is 200. To maximise the reward, the bar must keep the upright position as long as possible. We run 500 episodes for each model.

Inverse Pendulum Results

Figure 4 shows the plot of the episode reward (vertical axis) over the number of episodes (horizontal axis). In the first 100 episodes, we observe that DQN with safety estimator learns faster than the conventional DQN. This is because avoiding the safety violation can quickly increase the cumulative reward. In our inverse pendulum system, the maximum reward is only possible if the bar does not fall. Thus, a reward of less than 200 means an occurrence of a safety violation. In this sense, the conventional DQN and the single buffer model are experiencing frequent safety violations. However, there is a noticeable improvement in the double buffer case as the reward is maintained at the maximum value more consistently. In 500 episodes, the number of episodes violating the safety is 186 for the conventional DQN, 149 for the single buffer model, and 121 for the double buffer model. Thus, with the single buffer model, violation is reduced by 19.9%, and with the double buffer model by 34.9%.

As shown in the proof of Lemma 2, the hidden unsafe states can be recognised by checking the average Q-values. If the value is -1, the state is a hidden unsafe state. Figure 5 shows the heatmap of the average Q-value for the single buffer case at different episodes. The blue colour marks the average Q-value of 1, whereas red marks -1. Intuitively, the red areas indicate the hidden unsafe states. In Episode 0, the estimator is initialised. As it experiences more episodes, the average Q-value is updated. However, compared to the analytical solution in Figure 2b, the estimation accuracy is poor and unstable. Especially after Episode 460, every state is considered safe. This problem is caused by the biased data in the buffer. As the controller gets better and better, it becomes more difficult to collect the information about unsafe states. Consequently, the buffer becomes full of safe state information, and the neural network can only learn from safe states. Overall, the poor and unstable estimation accuracy also explains why the single buffer case in Figure 4 exhibit severe fluctuations.

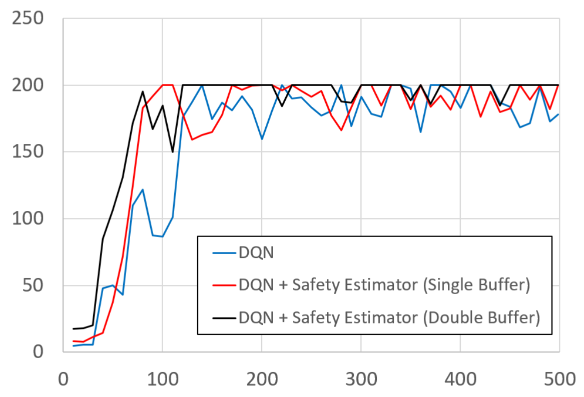


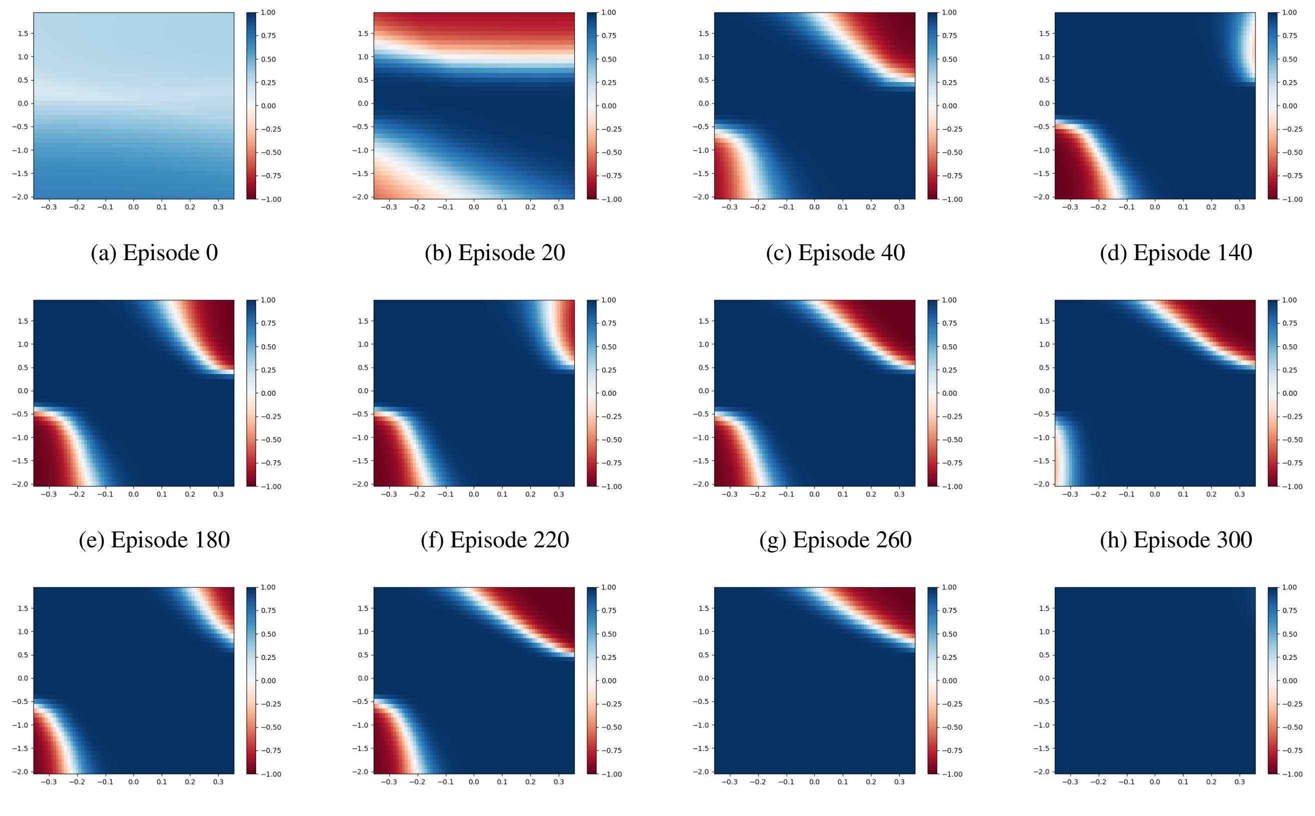
Figure 4: The cumulative reward of each episode is plotted over the number of episodes. The horizontal is the number of episodes, and the vertical axis is the cumulative reward of each episode.

The heatmaps for the double buffer case are shown in Figure 6. It is clear that the use of double buffers has significantly improved the estimation accuracy compared to the single buffer case. Furthermore, the estimated regions are not significantly changed over episodes, and there is no more biased learning problem in Episode 460. Despite the improved accuracy, the estimation is still an under-approximation of the analytic solution in Figure 2b. This is because the neural network learning also causes uncertainties. Hence, violations can still occur, but they are reduced by 34.9%.

Related work

Safe *policy extraction* (Watkins and Dayan 1992) is a technique that restricts the action space during policy extraction to ensure constraint satisfaction, which is widely used in various applications such as autonomous vehicle control (Mirchevska et al. 2018) and robot systems (Alshiekh et al. 2018). However, action masking often results in non-optimal solutions due to the lack of long-term future consideration. To address this, the study (Kalweit et al. 2020) specifies multi-step constraints using the formulation of constrained MDP (Altman 1999). However, these only look a fixed number of steps ahead. In contrast, the proposed approach do not have such a limitation. Another approach called Reward Constrained Policy Optimisation (RCPO) (Tessler, Mankowitz, and Mannor 2019) specifies penalties that are added to the reward, which essentially represent the constraints. Similar to others, RCPO requires the complete constraints to be given a priori.

Some studies combine the effort of formal methods and



(a) Episode 0 (b) Episode 20 (c) Episode 40 (d) Episode 140

(e) Episode 180 (f) Episode 220 (g) Episode 260 (h) Episode 300

(i) Episode 400 (j) Episode 420 (k) Episode 440 (l) Episode 460

Figure 5: The average Q-value heatmaps of the single buffer case at different episodes. In each heatmap, the x-axis is the angle or the bar, and the y-axis is the angular velocity.

machine learning. For example in (Junges, Torfah, and Seshia 2021), a runtime monitor is syntactically composed with the discrete MDP system. In the resultant automaton, all unsafe actions are known; hence, actions can be blocked appropriately. However, this approach cannot be used for the continuous state space due to the state explosion problem, and the given set of constraints need to be perfect.

To the best of our knowledge, no work has been done to deal with partially known (i.e., some unsafe states are not detected) and partially correct (i.e., the monitor's output can be incorrect) safety monitors.

Conclusions

This paper addressed the problem of dealing with imperfect safety monitors. Such monitors can only detect a subset of the actual unsafe states. Our approach used deep Q-learning to estimate the missing states which we called hidden unsafe states. The Q-value function was adjusted to find such states, and an action masking mechanism was employed to block unsafe actions. The experimental results of an inverse pendulum example showed that the learning speed is accelerated and the occurrence of safety violations is noticeably reduced.

In future work, the proposed approach can be extensively analysed using more complex benchmarking example with a larger action space, complicated reward functions, and dynamic changes in the deployment environment such as damaging. Also, further research can investigate how to reuse the learned knowledge about safety to improve/adjust safety monitors.

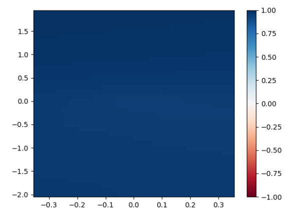
References

Alshiekh, M.; Bloem, R.; Ehlers, R.; Könighofer, B.; Niekum, S.; and Topcu, U. 2018. Safe reinforcement learning via shielding. In *32nd AAAI Conference on Artificial Intelligence*, 2669–2678.

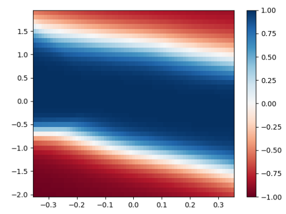
Altman, E. 1999. *Constrained Markov decision processes*, volume 7. CRC Press.

Berkenkamp, F.; Turchetta, M.; Schoellig, A. P.; and Krause, A. 2018. Safe model-based reinforcement learning with stability guarantees. In *31st Conference on Neural Information Processing Systems*, volume 2, 909–919. Curran Associates, Inc.

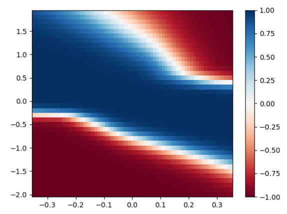
Fisac, J. F.; Akametalu, A. K.; Zeilinger, M. N.; Kaynama, S.; Gillula, J.; and Tomlin, C. J. 2018. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7): 2737–2752.



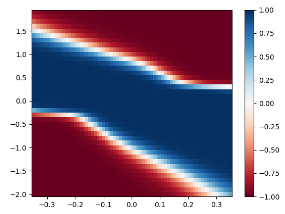
(a) Episode 0



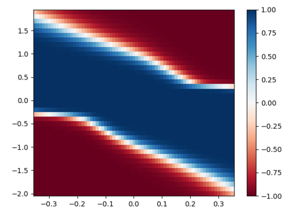
(b) Episode 20



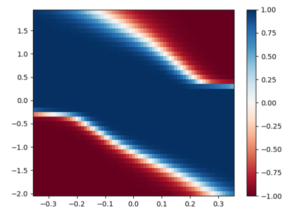
(c) Episode 40



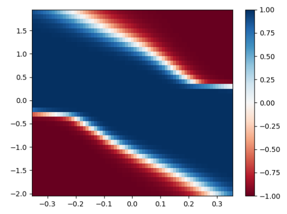
(d) Episode 140



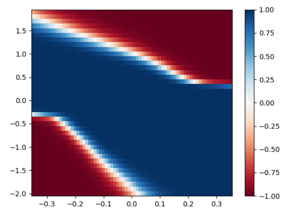
(e) Episode 180



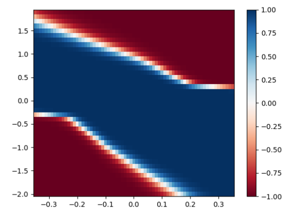
(f) Episode 220



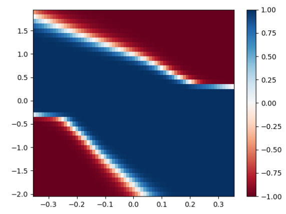
(g) Episode 260



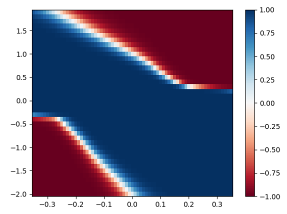
(h) Episode 300



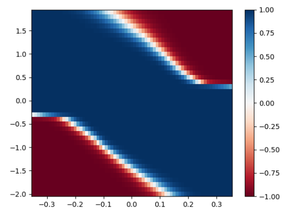
(i) Episode 400



(j) Episode 420



(k) Episode 440



(l) Episode 460

Figure 6: The average Q-value heatmaps of the double buffer case at different episodes. The x-axis is the angle or bar for each heatmap, and the y-axis is the angular velocity.

Junges, S.; Torfah, H.; and Seshia, S. A. 2021. Runtime monitors for Markov decision processes. In *International Conference on Computer Aided Verification*, 553–576. Springer.

Kalweit, G.; Huegle, M.; Werling, M.; and Boedecker, J. 2020. Deep Inverse Q-learning with Constraints. In *34th Conference on Neural Information Processing Systems*, volume 33, 14291–14302. Curran Associates, Inc.

Kantaros, Y.; Malencia, M.; Kumar, V.; and Pappas, G. J. 2020. Reactive temporal logic planning for multiple robots in unknown environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 11479–11485. IEEE.

Mirchevska, B.; Pek, C.; Werling, M.; Althoff, M.; and Boedecker, J. 2018. High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning. In *21st International Conference on Intelligent Transportation Systems (ITSC)*, 2156–2162. IEEE.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT Press.

Tessler, C.; Mankowitz, D. J.; and Mannor, S. 2019. Reward Constrained Policy Optimization. In *7th International Conference on Learning Representations, ICLR*, 1–15. OpenReview.net.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning*, 8(3-4): 279–292.

Acknowledgments

This work is carried out in the Dependable Intelligent Software Lab (DISL), funded by the German Federal Ministry of Education and Research (BMBF – Bundesministerium für Bildung und Forschung).