Learning Transferable Domain Priors for Safe Exploration in Reinforcement Learning

Thommen George Karimpanal[1], Santu Rana[1], Sunil Gupta[1], Truyen Tran[1], and Svetha Venkatesh[1]

[1]*Applied Artificial Intelligence Institute, Deakin University, Australia*

Abstract

Prior access to domain knowledge could significantly improve the performance of a reinforcement learning agent. In particular, it could help agents avoid potentially catastrophic exploratory actions, which would otherwise have to be experienced during learning. In this work, we identify consistently undesirable actions in a set of previously learned tasks, and use pseudo-rewards associated with them to learn a prior policy. In addition to enabling safer exploratory behaviors in subsequent tasks in the domain, we show that these priors are transferable to similar environments, and can be learned off-policy and in parallel with the learning of other tasks in the domain. We compare our approach to established, *state-of-the-art* algorithms in both discrete as well as continuous environments, and demonstrate that it exhibits a safer exploratory behavior while learning to perform arbitrary tasks in the domain. We also present a theoretical analysis to support these results, and briefly discuss the implications and some alternative formulations of this approach, which could also be useful in certain scenarios.

1 Introduction

Reinforcement learning (RL) [29] has proven to be a versatile and powerful tool for effectively dealing with sequential decision making problems. In addition to requiring only a scalar reward feedback from the environment, its reliance on the knowledge of a state transition model is limited. This has resulted in RL being successfully used to solve a range of highly complex tasks [31, 21, 27, 22].

However, RL algorithms are typically not sample efficient, and desired behaviors are achieved only after the occurrence of several unsafe agent-environment interactions, particularly during the initial phases of learning. Even while operating within the same domain, commonly undesirable actions (such as bumping into a wall in a navigation environment) have to be learned to be avoided each time a new task (navigating to a new goal location) is learned. This can largely be attributed to the fact that in RL, behaviors are generally learned *tabula-rasa* (from scratch) [9], without contextual information of the domain it is operating in. This lack of contextual knowledge is usually a limiting factor when it comes to deploying RL algorithms in real world systems, where executing sub-optimal actions during learning could be highly dangerous to the agent or to elements in its environment. Providing RL agents with domain-specific contexts in the form of suitable initializations and/or domain-specific, reusable priors could greatly help mitigate this problem.

The challenge of addressing the issue of avoiding undesirable actions during learning has been the primary focus of the field of safe RL [12], and consequently, a number of methods have been proposed to enable RL agents to learn to solve tasks, with due consideration given to the aspect of safety. These methods aim to bias RL agents against such actions, broadly, by means of modifying either the optimization criterion or the exploration process [12]. In either case, the nature of the bias is to directly or indirectly equip the agent with prior information regarding its domain, which is subsequently used to enable safer learning behaviors. Safe RL approaches where such prior knowledge is extracted from already learned tasks in the domain share similarities with the ideology of transfer learning [30], in the sense that they both reuse previously acquired knowledge to achieve a specific behavior. Perhaps the main distinction between the two is that the former focuses on using domain-specific knowledge to achieve safe behaviors,

whereas the focus of the latter is more generally, to reuse previously acquired task knowledge to achieve good learning performance on a new task. Previous works [10, 16, 5] have explored the idea of exploiting known task knowledge for improving learning performance, but ignore aspects relating to safety. Other approaches which were specifically designed to enable safe exploration [1, 23, 11] were based on strong assumptions such as the availability of a safe baseline policy or the explicit specification of a constraint function. Although the idea of excluding unsafe actions during learning has been explored in previous works [2, 33], they too are reliant on explicit domain or safety specifications. In addition, previous works that incorporate safe behaviors in RL agents have not considered the issue of the ease of adaptation of the safe policy in new, but related domains.

In this work, we propose an approach to learn a transferable domain prior for safe exploration by incrementally extracting, refining and reusing common domain knowledge from already learned policies, an approach consistent with the ideology of continual learning [24]. The reward function used for learning this prior is constructed by approximating rewards from the Q- functions of the previously learned tasks for state-action pairs consistently associated with undesirable agent behaviors. Unlike other safe RL approaches, our approach does not require the explicit specification of a safety or constraint function to encode safe behaviors, or prior access to a safe policy. The focus is to instead, extract knowledge from previously learned tasks to learn a safety prior, which is subsequently used to bias an agent's exploratory behavior while it learns arbitrary tasks in the domain. The intuition behind this approach is that for a given domain, there exist behaviors that are commonly undesirable for any arbitrary task in that domain. As the prior is stored in the form of a Q- function, it can be learned off-policy [13], in parallel with an arbitrary task that the agent is learning, without the need for additional interactions with the environment. The prior can also be transferred or reused, and is capable of quickly adapting to other similar environments, under the assumption that there exists a considerable overlap in the set of undesirable actions in the two environments. We demonstrate this claim in a simple tabular environment, while also demonstrating the effectiveness of the proposed approach in more complex environments with continuous states and/or continuous actions. We also quantify the effectiveness of our approach in enabling safe exploration in tabular domains by analytically deriving an expression that relates the probability of executing unsafe actions using our approach, relative to an ϵ-greedy exploration strategy, for a given degree of correctness of the learned priors.

In summary, the main contributions of this work are:

- A novel framework for learning domain priors from previously known tasks.
- A theoretical relation between correctness of a prior and the relative probability of unsafe exploratory actions.
- Experimental results in both discrete as well as continuous environments, validating the benefits of learning and using the described priors.
- Experimental results in the discrete action setting, demonstrating the transferability of the learned priors to other similar environments.

2 Related Work

The goal of our approach is to achieve safe exploratory actions during the learning process by making use of existing knowledge of other tasks in the domain, an ideology that is typical of many transfer RL [30] frameworks. Specifically, we consider the case where the tasks differ only in the reward functions [6, 18]. In one of the popular approaches [10] that addressed this case, past policies were reused based on their similarity to the task being solved. In addition to being able to effectively reuse past policies, the approach was also shown to be capable of extracting a set of "core" policies to solve any task in a given domain. A recent method by Li and Zhang [16] improved this policy reuse approach by optimally selecting the source policies online. However, these approaches, along with several others [25, 28] are only concerned with the problem of reusing past policies to achieve quicker learning in the target task, without consideration to the cost of executing poor exploratory actions during learning. More recent works [23, 15] have emphasized this problem in greater detail, with accompanying environments that demonstrate the distinction between reward-maximization behavior and safety performance for a range of tasks.

Most approaches that are directly concerned with achieving safe behaviors during learning, do so by incorporating domain knowledge, and biasing the actions of the learning agent by modifying either the optimization criterion or the exploration process. A detailed summary of such approaches can be found in Garcia and Fernandez [12]. Among these, a few consider the problem of safety at the policy level [8, 3], while others aim to improve safety at the level of states and actions, much like the approach described in the present work. The PI-SRL approach by Garcia and Fernandez [11] avoids the exploration of unsafe states by using a known safe baseline policy, coupled with case-based reasoning. However, the maintenance of their case-base of known states is based on a Euclidean similarity metric, which may not be a useful measure in many situations, and hence limits the generalizability of the approach. Additionally, their assumption regarding the availability of a safe baseline policy may not be reasonable in many practical circumstances. The Lagrangian and constrained policy optimization approaches [1, 23] greatly improve safety performance. However, they require the explicit specification of a safety performance metric or a constraint function, which may not always be available.

The idea of achieving safe learning behaviors by biasing against certain actions has also been proposed in other recent work. Zahavy et al. [33] proposed the approach of action elimination deep Q-networks [21], which essentially eliminates sub-optimal actions, and performs Q-learning on a subset of the state-action space. The elimination of actions is based on a binary elimination signal which is computed using a contextual bandits framework. Similar to this, the idea of shielding was proposed by Alshiekh et al.[2], where unsafe actions were disallowed based on a shielding signal. The authors synthesize the shield separately, from a safety game between an environment and a system player. Akin to these approaches, the basis of our approach is to bias the agent against certain actions that are considered to be undesirable, as per a learned prior policy. However, the key idea is to obviate the need for domain-specific safety constraints, and instead, learn a safety prior from a set of previously learned tasks, in an online and off-policy manner, without the requirement of additional interactions with the environment.

3 Methodology

We consider the objective of learning a prior policy π_P by learning the corresponding Q-function Q_P in a domain $\mathcal{D} =< \mathcal{S}, \mathcal{A}, \mathcal{T} >$, where the tasks $\mathcal{M} =< \mathcal{D}, \mathcal{R} >$ share a common state-space $\mathcal{S}$, action-space $\mathcal{A}$ and state-transition function $\mathcal{T}$, and differ solely in the reward function $\mathcal{R}$. The purpose of this prior is to bias the agent against exploratory actions that have a high degree of undesirability , which we define as follows:

Definition 1. The undesirability of an action a is the absolute value of the optimal advantage $A^*(s,a)$ for that action, where $A^*(s,a) = Q^*(s,a) - \underset{a' \in \mathcal{A}}{max}\, Q^*(s,a')$.

The optimal advantage function $A^*(s,a)$ [4] measures the deviation of the Q -value for a particular state-action pair (s,a) from the maximum Q -value associated with the state s. Thus, $|A^*(s,a)|$ is indicative of how much worse action a is, in relation to the best action in that state.

In order to learn Q_P, we assume that we know the optimal Q -functions corresponding to N arbitrary tasks in the domain $\mathcal{D}$. For the sake of argument, let us consider the case where $N > 1$, which implies there exist at least a few tasks $\boldsymbol{M} = \{\mathcal{M}_1 ... \mathcal{M}_i ... \mathcal{M}_N\}$ whose optimal Q- functions $\boldsymbol{Q^*} = \{Q_1^* ... Q_i^* ... Q_N^*\}$ are known. In the proposed approach, Q_P corresponds to a pseudo-task $\mathcal{M}_\mathcal{P} =< \mathcal{D}, \mathcal{R}_\mathcal{P} >$ that is learned off-policy by sampling state-action pairs in the given domain, for example, by executing random exploratory actions in the environment. More practically, they are sampled as per a behavior policy π_B corresponding to an arbitrary task $\mathcal{M}_\Omega =< \mathcal{D}, \mathcal{R}_\Omega >$, that is being learned in parallel. Although in general, any off-policy approach could be used to learn Q_P, for simplicity, here, we show the learning of Q_P using Q-learning [32].

The basis of our approach is to construct the pseudo-reward function $\mathcal{R}_\mathcal{P}$ based on state-action pairs that are consistently undesirable across the N known tasks. We infer rewards that would likely be associated with such state-action pairs and subsequently construct $\mathcal{R}_\mathcal{P}$ as a weighted sum of these inferred rewards. Once $\mathcal{R}_\mathcal{P}$ is constructed, Q_P is learned off-policy, and is subsequently used to bias the exploratory actions of the agent. Corresponding to this description, our methodology is composed of the following steps:

3.1 Identification of Suitable State-Action Pairs

The first step in our approach is to identify state-action pairs that are consistently associated with undesirable agent behaviors. Once a state-action pair (s, a) has been sampled using the agent's behavior policy π_B, for each task $\mathcal{M}_i$ of the N known tasks, we measure the undesirability $w_i(s, a)$ of the action as a quantity proportional to the action's undesirability, as per Definition 1. In order to scale these values to be ≤ 1 , we measure the scaled undesirability $w_i(s, a)$ as:

$$w_i(s,a) = \left| \frac{A_i^*(s,a)}{\underset{a' \in \mathcal{A}}{max}\, Q_i^*(s,a')} \right| \tag{1}$$

We repeat this procedure for each of the N tasks, and store the obtained measures in a sequence $W(s, a)$ as follows:

$$W(s,a) = \{w_1(s,a), ...w_i(s,a), ...w_N(s,a)\} \tag{2}$$

The overall consensus on the undesirability of action a in state s, as per the N known tasks can then be measured by quantifying the consistency in the values stored in $W(s, a)$. We do this by converting $W(s, a)$ into a probability distribution $W^{'}(s, a)$ and then measuring the normalized entropy $\mathcal{H}(W^{'}(s, a))$ associated with it:

$$\mathcal{H}(W'(s,a)) = -\frac{\sum_{i=1}^{N} w_i'(s,a)\, log(w_i^{'}(s,a))}{log(N)} \tag{3}$$

where $W^{'}(s, a) = \{w_1^{'}(s, a), ...w_i^{'}(s, a), ...w_N^{'}(s, a)\}$, and $w_i^{'}(s, a)$, the i^{th} element of $W^{'}(s, a)$, is computed using the softmax function:

$$w_i^{'}(s,a) = \frac{e^{w_i(s,a)}}{\sum_{i=1}^{N} e^{w_i(s,a)}} \tag{4}$$

In order to construct the pseudo-reward function $\mathcal{R}_{\mathcal{P}}$, we select state-action pairs which are associated with high values of $w_i(s, a)$, as well as a high normalized entropy value $\mathcal{H}(W^{'}(s, a))$. The former criterion, quantified by the mean $\mu(W(s, a)) = \frac{\sum_{i=1}^{N} w_i(s,a)}{N}$ of the values in $W(s, a)$, prioritizes state-action pairs that are highly undesirable. The latter criterion $\mathcal{H}(W^{'}(s, a))$ quantifies the consistency of the undesirability of the state-action pair across the known tasks. To account for both these criteria, we use a threshold t, and select state-action pairs for which:

$$\mathcal{H}(W'(s,a)) * \mu(W(s,a)) > t \tag{5}$$

The general idea is to select state-action pairs associated with highly and consistently undesirable behaviors across the known tasks in the domain. The selection of state-action pairs using Equation 5 depends heavily on the choice of a suitable threshold value t, for which a rough guideline can be obtained by considering the ranges of $\mathcal{H}(W'(s, a))$ and $\mu(W(s, a))$. $H(W'(s, a))$ lies in the range $[0, 1]$, while the range of $\mu(W(s, a))$ depends on that of the function $\left| \frac{A^*(s,a)}{\underset{a' \in \mathcal{A}}{max}\, Q^*(s,a')} \right|$, or equivalently, using Definition 1, $\left| \frac{Q^*(s,a) - \underset{a' \in \mathcal{A}}{max}\, Q^*(s,a')}{\underset{a' \in \mathcal{A}}{max}\, Q^*(s,a')} \right|$. The minimum value of this function is 0, which corresponds to the case when $a = \underset{a' \in \mathcal{A}}{argmax}\, Q^*(s, a')$. The maximum value corresponds to the case when $Q^*(s, a)$ is as low as possible, and $\underset{a' \in \mathcal{A}}{max}\, Q^*(s, a')$ is as large as possible. If r_{min} and r_{max} represent the lowest and highest possible rewards in the domain, then using the lower and upper bounds of $\frac{r_{min}}{1-\gamma}$ and $\frac{r_{max}}{1-\gamma}$ for the Q- function, the maximum possible value of $\left| \frac{A_i^*(s,a)}{\underset{a' \in \mathcal{A}}{max}\, Q_i^*(s,a')} \right|$ would be: $\left| \frac{r_{min} - r_{max}}{r_{max}} \right|$. Hence, threshold t must be selected to be in the range $[0, \left| \frac{r_{min} - r_{max}}{r_{max}} \right|]$. In general, a lower threshold value results in a larger number of state-action pairs being selected for the construction of $\mathcal{R}_{\mathcal{P}}$, possibly leading to a more conservative prior.

3.2 Constructing Pseudo-rewards and Learning Q_P

The next step is to use the identified state-action pairs to construct a safety prior. Consider an arbitrary task $\mathcal{M}$ in the domain for which the policy is learned using Q- learning. The corresponding standard update equation is given by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a,s') + \gamma \max_{a' \in \mathcal{A}} Q(s',a') - Q(s,a)] \tag{6}$$

Here, s and a represent the current state and action, γ is the discount factor ($0 \leq \gamma \leq 1$), s' is the next state, and $r(s,a,s')$ is the reward associated with the transition.

When the optimal Q-function Q^* is learned, the temporal difference (TD) error: $[r(s,a,s') + \gamma \max_{a' \in \mathcal{A}} Q^*(s',a') - Q^*(s,a)]$ would reduce to 0. Using this fact, we can infer the original reward $r(s,a,s')$ associated with the transition:

$$r(s,a,s') = Q^*(s,a) - \gamma \max_{a' \in \mathcal{A}} Q(s',a') \tag{7}$$

In reality, the above equality seldom holds, as the TD error may not be exactly 0. However, the inferred reward may still be a reasonable approximation if the Q-function is close to optimal ($Q \approx Q^*$). With this assumption in mind, we apply Equation 7 to each of the known tasks, and construct the rewards associated with those state-action pairs (s_c, a_c) which satisfy the condition in Equation 5. The pseudo-reward r_P is computed as a sum of these inferred rewards, weighted by the corresponding elements of $W'(s_c, a_c)$:

$$r_p(s_c,a_c,s'_c) = \sum_{i=1}^{N} w'_i(s_c,a_c)[Q^*_i(s_c,a_c) - \gamma \max_{a' \in \mathcal{A}} Q^*_i(s'_c,a')] \tag{8}$$

r_P is capped to have a maximum absolute value of 1, and for state-action pairs that do not satisfy Equation 5, r_P is set to a default value of 0. r_P is then used to update the Q- function Q_P via the standard Q- learning update equation (Equation 6). By continuously sampling state-action pairs, determining the corresponding pseudo-reward r_P and updating Q_P, the optimal Q- function Q^*_P, is learned. It is worth mentioning that Q_P is updated using what ever state-action pairs are sampled by the behavior policy π_B. Hence, no additional interactions with the environment are required for its computation. However, learning Q^*_P is subject to the condition that π_B sufficiently explores the state-action space. The additional requirements for learning a prior policy are the additional memory and computations corresponding to inferring r_P, and storing and updating Q_P. The overall process of updating Q_P is summarised in Algorithm 1.

3.3 Biasing Exploration Using Q^*_P

Following the construction of the domain priors, the final step is to use these priors to bias the exploratory behavior of the agent. Q^*_P is learned based on a reward function $\mathcal{R}_\mathcal{P}$, which is specifically constructed using state-action pairs that are consistently associated with undesirable actions. Hence, in order to avoid catastrophic actions during learning, we simply bias the agent's behavior against taking undesirable actions, as determined by Q^*_P. If such an action happens to be suggested by the agent during learning, with a high probability ρ, we disallow it from being executed, and force the agent to pick an alternative action whose Q^*_P value is at least equal to the mean value of Q^*_P over all actions. The threshold of $\underset{a' \in \mathcal{A}}{mean} Q^*_P(s,a')$ was chosen, simply to ensure that better-than-average actions are executed during exploration. More conservative (higher) or radical (lower) threshold values could also be considered, although it must be noted that choosing a very high threshold would limit the extent of exploration, while a very low threshold would fail to leverage the safe exploratory behaviors enabled by Q^*_P. Algorithm 2 outlines the process of biasing the agent against undesirable exploratory actions.

4 Theoretical Analysis

Biasing the exploratory actions as described would, in an ideal case, help avoid unsafe actions. However, the effectiveness of using the learned priors to bias against these actions is highly dependent on how correct

Algorithm 1 Algorithm for updating prior Q-function Q_P

Input:
Set of N optimal Q- functions $\boldsymbol{Q^*} = \{Q_1^*...Q_i^*...Q_N^*\}$, Estimate of prior Q-function Q_P, maximum number of steps per episode H, behavior policy π_B, threshold t
Output: updated estimate of Q_P
for H steps **do**
  Execute behavior policy π_B to take action a from state s, and obtain next state s'
  Initialize $W(s,a)$ as an empty set
  for each task i of the N known tasks **do**
    Compute $A_i^*(s,a) = Q_i^*(s,a) - \underset{a' \in \mathcal{A}}{max}\, Q_i^*(s,a')$
    $w_i(s,a) = \left| \frac{A_i^*(s,a)}{\underset{a' \in \mathcal{A}}{max}\, Q_i^*(s,a')} \right|$
    $W(s,a) = W(s,a) \cup w_i(s,a)$
  end for
  Normalize $W(s,a)$ using Equation 4 to obtain $W^{'}(s,a) = \{w_1^{'}(s,a)...w_i^{'}(s,a)...w_N^{'}(s,a)\}$
  Compute $\mathcal{H}(W^{'}(s,a))$ (Equation 3)
  Compute $\mu(W(s,a)) = \frac{1}{N}\sum_{i=1}^{N} w_i(s,a)$
  Initialize pseudo-reward $r_P(s,a,s')$ as 0
  if $\mu(W(s,a)) * \mathcal{H}(W^{'}(s,a)) > t$ (threshold) **then**
    $r_P(s,a,s')$
    $= \sum_{i=1}^{N} w_i'(s,a)[Q_i^*(s,a) - \gamma \underset{a' \in \mathcal{A}}{max}\, Q_i^*(s',a')]$
  end if
  $Q_P(s,a) \longleftarrow Q_P(s,a) + \alpha[r_P(s,a,s') + \gamma \underset{a' \in \mathcal{A}}{max} Q_P(s',a') - Q_P(s,a)]$
end for

Algorithm 2 Biasing against undesirable exploration

Input:
Proposed exploratory action a_0, state s, optimal Q- function of prior Q_P^*, probability of using priors ρ
Output: selected action a
With a probability ρ:
while $Q_P^*(s,a) < \underset{a' \in \mathcal{A}}{mean} Q_P^*(s,a')$ **do**
  Pick random action from $\mathcal{A}$: $a_0 = random(\mathcal{A})$
end while
$a = a_0$

the priors are. In this section, we consider the discrete actions setting, and derive a relation between the correctness of a prior and the probability of taking unsafe actions using our approach, relative to an ϵ-greedy exploration policy. We first define the terms 'unsafe actions' and 'correctness of a prior' for the purpose of our analysis, as follows:

Definition 2. An action a is considered unsafe in a state s if $Q_P^*(s,a) < \underset{a' \in \mathcal{A}}{mean} Q_P^*(s,a')$ in that state.

Definition 2 was chosen to be consistent with the biasing criteria used in Algorithm 2.

Definition 3. The correctness $C_{Q_P,\mathcal{D}}$ of a prior Q_P, with respect to a domain $\mathcal{D}$ is the probability with which it avoids deeming an action to be safe, when it is actually unsafe.
$C_{Q_P,\mathcal{D}} = 1 - \frac{n_{FN}}{n_I - n_{FP} + n_{FN}}$

where n_{FP} and n_{FN} are respectively the number of false positives (cases where the action has been incorrectly classified by Q_P as unsafe) and false negatives (cases where the action has been incorrectly classified by Q_P as safe), and n_I is the number of unsafe actions identified by Q_P. It is worth noting that only the false negative cases affect the probability of encountering truly unsafe actions. The effect of false

positives would be to simply slow down learning. The extent to which the correctness $C_{Q_P,\mathcal{D}}$ affects the probability of encountering unsafe actions, relative to the case of ϵ-greedy exploration, is presented in the following theorem:

Theorem 4. *If a prior Q_P with a correctness of $C_{Q_P,\mathcal{D}}$, is used to bias the exploratory actions with a probability of ρ, then relative to the case of standard ϵ-greedy exploration, the probability of executing unsafe exploratory actions in a given state is reduced by a factor of $1 - \frac{\rho(|\mathcal{A}|C_{Q_P,\mathcal{D}} - U)}{|\mathcal{A}| - U}$, where $\mathcal{A}$ is the action space associated with the domain, and U is the number of unsafe actions associated with that state.*

Proof. For the case of standard $\epsilon - greedy$ exploration, the agent takes exploratory actions with a probability of ϵ, in each instance of which, the probability of picking an unsafe action is $\frac{U}{|\mathcal{A}|}$. Hence, the probability of unsafe exploratory actions for an $\epsilon - greedy$ strategy is: $p_{\epsilon-greedy} = \frac{\epsilon U}{|\mathcal{A}|}$

Now, in the case of biased exploration, exploratory actions occur with a probability of ϵ, and are biased using the priors, with a probability ρ. When the bias is used, the agent eliminates unsafe actions (as determined by Q_P), and uniformly and randomly selects from the remaining $|\mathcal{A}| - U$ actions. However, the selected action may still be unsafe due to the presence of false negatives, which occur with a probability of $1 - C_{Q_P,\mathcal{D}}$. With the remaining probability of $(1 - \rho)$, exploration occurs exactly as in the $\epsilon - greedy$ case. Hence, the total probability of unsafe actions occuring during exploring is: $p_{priors} = \frac{\epsilon U \rho (1 - C_{Q_P})}{|\mathcal{A}| - U} + \frac{\epsilon U (1-\rho)}{|\mathcal{A}|}$. The ratio $\frac{p_{priors}}{p_{\epsilon-greedy}}$ can then be simplified to: $1 - \frac{\rho(|\mathcal{A}|C_{Q_P} - U)}{|\mathcal{A}| - U}$ □

This implies that fewer unsafe actions can be expected when $C_{Q_P,\mathcal{D}}$ and ρ have values close to 1. Although a large value of ρ is favorable, in order to maintain a non-zero probability of visiting every state-action pair (and thus ensure convergence), it is set to be slightly lesser than 1. For the purpose of this analysis, we only considered environments with discrete actions. However, in practice, our approach was also used to bias exploration in continuous action environments in Section 4.3. This was done by randomly sampling a large number of actions from a uniform distribution, and applying the exploration bias on this set of discretized actions.

5 Results

Benchmark Environments and Baselines

In order to test the learning and safety performance of the described approach, we chose three different environments. The first is a classical navigation environment shown in Figure 1(a), first introduced by Fernandez and Veloso [10], where the state and action spaces are discrete. For this tabular environment, we use OPS-TL[16], PRQL[10], PI-SRL[11] and Q- learning[32] as baselines for performance comparison.

Next, we show the agent's performance in a safety grid world 'Island Navigation' environment, shown in Figure 1(b), which was first introduced by Leike et al. [15] as a benchmark designed to evaluate safe exploration performance. The choice of baselines for this environment was A2C[19], SARSA[29] and DQN[21].

Lastly, we demonstrate the performance of our approach on a safe exploration task, shown in Figure 1(c), in the 'Safety Gym' environment, a continuous action environment recently introduced by Ray et al. [23]. For this environment, the chosen baselines were PPO[26], PPO-Lagrangian (a version of PPO with explicit constraints[23]) and DDPG[17].

We chose to validate our approach using these selected environments, as typical RL tasks in environments such as Atari [20] or OpenAI Gym [7] are set up largely with a focus on learning performance, without much consideration given to aspects relating to safety.

5.1 Classical Navigation Environment

We first demonstrate extensive results from our approach on a classical 21×24 grid-world navigation environment shown in Figure 1(a), before proceeding to more complex and continuous environments in Sections 4.2 and 4.3. The environment settings are consistent with those reported in [10]. Here, each state is represented by a 1×1 grid cell, with darker colored cells representing obstacles, and other cells representing

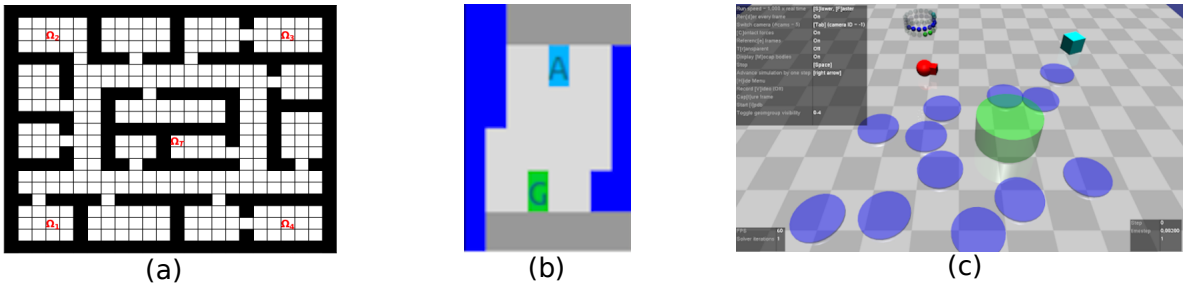

Figure 1: (a) shows the classical navigation environment, with goal locations $\Omega_1, \Omega_2, \Omega_3, \Omega_4$ of the known tasks, and goal location Ω_{T} of the task to be learned. (b) shows the agent 'A', the goal 'G' and the 'water' locations in blue in the island navigation environment and (c) shows a task in the Safety Gym PointGoal environment, where the green area is the navigation target, and the purple areas represent hazards which need to be avoided by the agent (in red).

free positions. The agent's state is represented by its (x, y) coordinates, and at each state, it is allowed to take one of four actions - moving up, down, left or right. Following the execution of an action, the agent moves to a new state, which is noised by random values sampled from a uniform distribution in the range (-0.2,0.2).

When the agent executes an action that causes it to bump into an obstacle, it retains its original state, without moving and receives a reward of -1. Goal states are terminal, and transitions leading into them are associated with a reward of 1. For all other transitions, the agent receives a small negative reward of -0.1. This penalises behaviors such as moving back and forth between two non-goal states.

For each task, the agent is allowed to interact with the environment for K episodes. Each episode starts with the agent in a random, non-goal state, following which, it could execute upto H actions to try and reach the terminal goal state. The performance W of the agent is evaluated by computing the discounted sum of rewards per episode as follows:

$$W = \frac{1}{K} \sum_{k=0}^{K} \sum_{h=0}^{H} \gamma^h r_{k,h} \tag{9}$$

where $r_{k,h}$ is the reward received from the environment at step h of episode k. We use the same metric to evaluate the performances in the continuous environments.

In order to obtain source policies, the agent is initially trained to learn the tasks $\mathcal{M}_{\Omega_1}, \mathcal{M}_{\Omega_2}, \mathcal{M}_{\Omega_3}$ and $\mathcal{M}_{\Omega_4}$, corresponding to the navigation target locations $\Omega_1, \Omega_2, \Omega_3$ and Ω_4. The label Ω_{T} in Figure 1(a) marks the goal location of the target task $\mathcal{M}_{\Omega_{\mathrm{T}}}$, which the agent aims to learn.

The prior is learned using the optimal Q-functions of tasks $\mathcal{M}_{\Omega_1}, \mathcal{M}_{\Omega_2}, \mathcal{M}_{\Omega_3}$ and $\mathcal{M}_{\Omega_4}$, as described in Algorithm 1. Figure 2 depicts the set of consistently undesirable actions identified using these known tasks, which is then used for learning the prior Q_P. The red, green, blue and orange arrows represent actions that move the agent up, right, down and left respectively. As observed in Figure 2, most of the identified actions correspond to those that would cause collisions with obstacles in the environment. The task $\mathcal{M}_{\Omega_{\mathrm{T}}}$ is then learned by biasing the exploratory actions of the agent using the learned prior, as described in Algorithm 2.

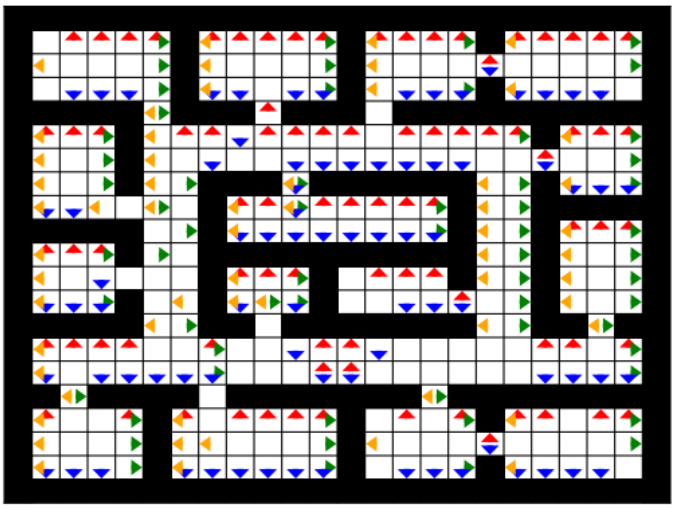

Figure 2: Identified set of consistently undesirable actions extracted from known tasks for the environment in Figure 1(a).

Figure 3 shows the average performance over 10 trials, of different algorithms, evaluated using Equation 9. The shaded regions represent the standard errors of the mean performances for the 10 trials. The common learning parameters were set as follows: $\alpha = 0.05, \gamma = 0.95, H = 500, K = 2000$, and the probability of exploration ϵ was set to be decaying from an initial value of 1, as in [10]. Two of the performance curves in Figures 3 and 4 were obtained by combining the described approach with: a) standard Q-learning [32], and b) PRQ-learning (PRQL) [10]($\psi = 1, \nu = 0.95$). The parameters specific to our approach were chosen to be: $t = 0.35, \rho = 0.95$. As observed from the figure, these curves exhibit a superior learning and safety performance compared to their corresponding counterparts, in which the learning occurs without the use of domain priors. In particular, the use of learned priors enables a significant increase in the initial performance of the agent, due to fewer unsafe exploratory actions during the initial phases of learning. This is supported by the results in Figure 4, which depicts the trend in the number of obstacle collisions per episode in each of the tested approaches. The overall performance of the agent is also superior to that of other approaches such as OPS-TL [16]($c = 0.0049$) for selecting source tasks, and the PI-SRL approach [11] ($k = 6, \sigma = 0.5$), in which safe exploratory actions are chosen based on case-based reasoning. Although the latter approach has a marginally better initial performance as seen in Figure 3, the learned policy is very conservative, as indicated by the negligible improvement in its performance across the episodes. From these figures, it is evident that the use of domain priors brings about improvements in both safety as well as learning performance.

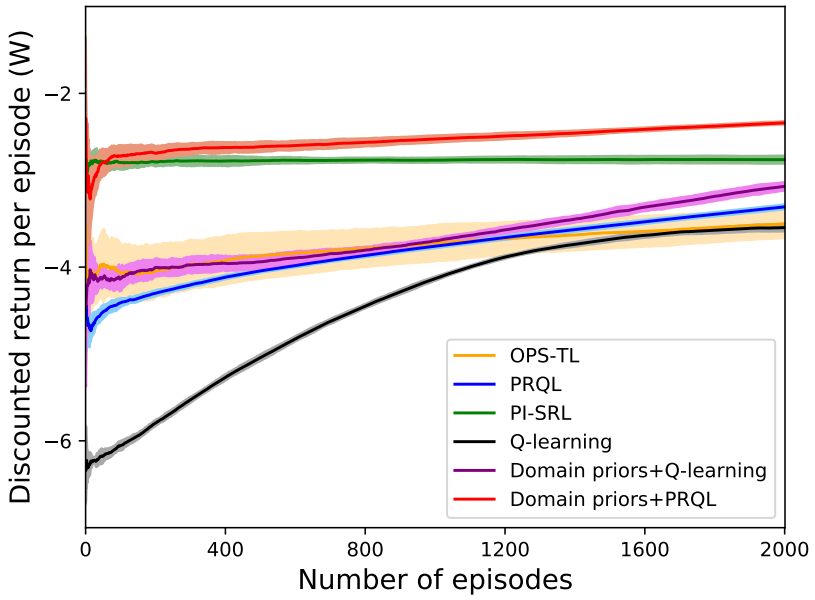

Figure 3: The average discounted returns per episode (W), computed over 10 trials, for different learning methods in the classical navigation environment.

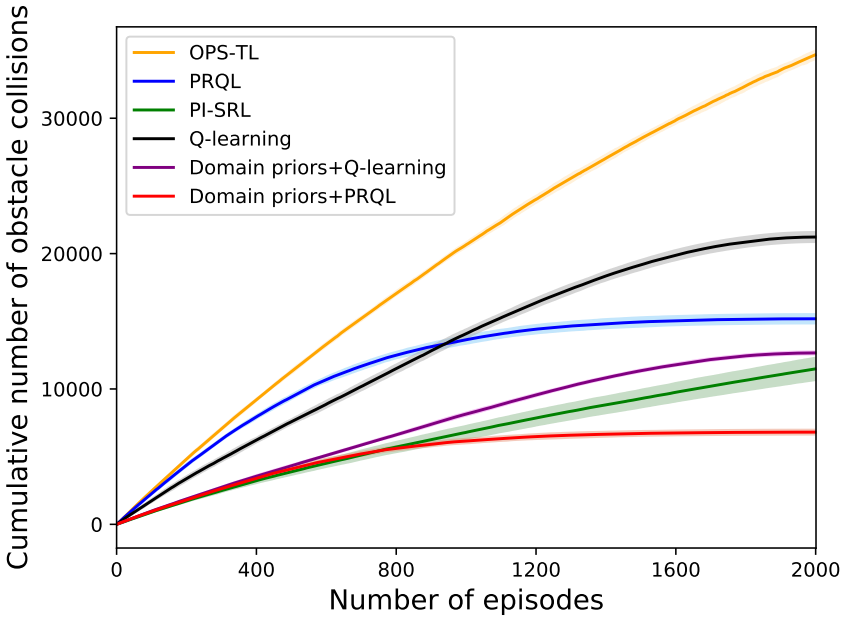

Figure 4: The cumulative number of obstacle collisions, computed over 10 trials, for different learning methods in the classical navigation environment.

5.2 Continuous State Environment

The results from Section 4.1 demonstrate the effectiveness of the proposed method in simple tabular domains. Although the nature of the task in the non-tabular 'Island Navigation' domain [15] considered in this section is roughly similar to that in Section 4.1, there exists a fundamental difference between the two, in that the states are now represented using features. The goal in this environment is for the agent to navigate to the target location using a set of discrete actions (moving left,right,up and down) without stepping into the 'water' locations. In order to obtain the source policies to construct the priors, we first solved a set of 4 random tasks using Deep Q-learning (DQN) [21] by randomly generating the target locations. Consistent with the implementation in Leike at al. [15], both the A2C as well as the DQN implementations used a 2 layered multi-layer perceptrons with 100 nodes each, trained with inputs that consisted of a matrix encoding the current configuration of the environment. The architecture for SARSA was kept identical to that for DQN, and varied only in the value function update rule. For A2C, we used an entropy penalty parameter of 0.05, which linearly decayed to 0 at the end of each trial. For optimization, we used Adam[14] with a learning rate of $5e-4$ and a batch size of 64. For each task, the agent was trained for 2000 episodes, each consisting of up to 100 steps. The other parameters used were: a discount factor of 0.99, an initial exploration parameter of 1, which decayed exponentially to a minimum of 0.1 (with a decay factor of 0.95), a replay buffer of size 2000, a threshold $t = 0.25$ and $\rho = 0.95$.

Using the obtained source policies, we implemented our approach described in Section 2, and tested the performance of the agent on a new task, while its exploration was biased using the learned priors, as described in Algorithm 2. Figures 5 and 6 depict the performance of various approaches, averaged over 15 trials. As observed, our method of biasing the exploration using the learned priors was able to improve the agent's learning performance, while simultaneously achieving a fewer visits to the 'water' locations, thereby also improving the safety performance.

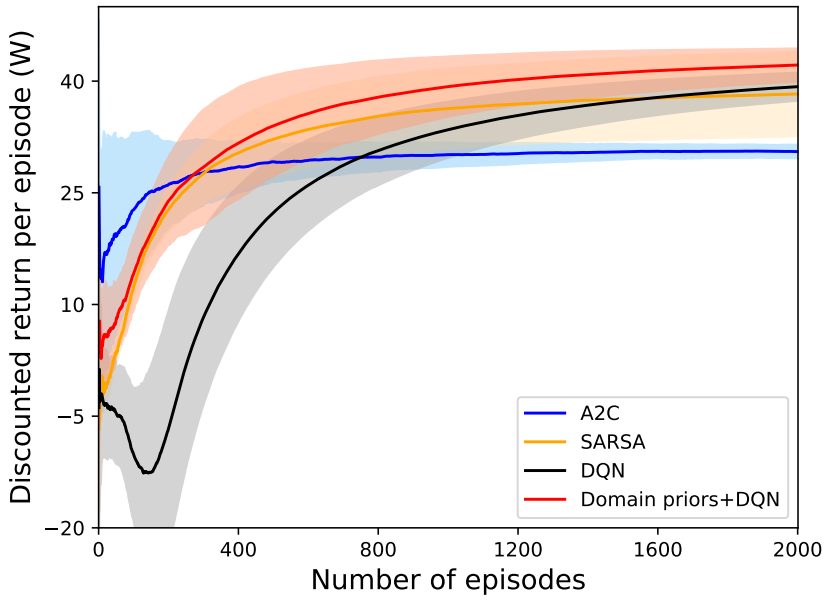

Figure 5: The average discounted returns per episode (W), computed over 15 trials, for different learning methods in the island navigation environment.

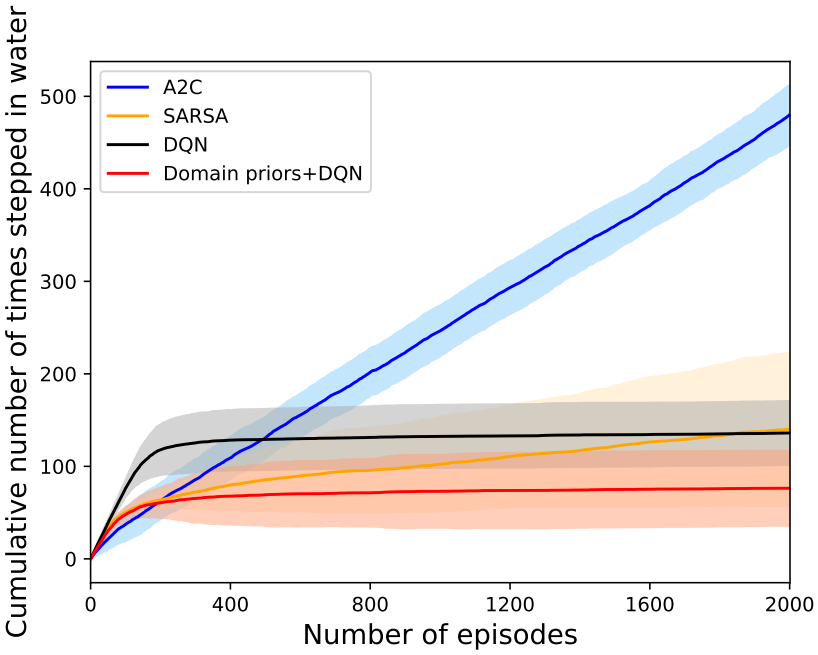

Figure 6: The cumulative number of times stepped in water, computed over 15 trials, for different learning methods in the island navigation environment.

Our method for learning domain priors naturally scales to such non-tabular environments, fundamentally because the process of inferring r_P (Equation 8) does not explicitly depend on the state complexity, and only depends on the Q-values of the N tasks for the specific transition (s_c,a_c,s'_c) under consideration. This can be obtained with a maximum of $N|\mathcal{A}|$ queries to the stored Q-networks, which depends only on $|\mathcal{A}|$ and N, and is independent of the size of the state space.

5.3 Continuous Action Environment

The 'Safety Gym' [23] environment consists of both continuous states and actions. To implement our approach in such as setting, we chose a version of the PointGoal1 environment, 'PointGoal1-12', where the number of 'hazards' were set to 12, making it a more unsafe environment than the original PointGoal1 environment. The aim of the agent in this environment is to navigate to the goal location while avoiding the 'hazard' locations. Each of the 1000 episodes are run for 1000 steps. As in the case of the other environments, we initially obtained source policies by separately training a DDPG [17] agent on 3 tasks. Using these source policies, we implemented our described approach for safe exploration. For the DDPG implementation, the critic and target networks were multi-layer perceptrons with 3 and 2 layers respectively, with the former having $1024, 512$ and 300 nodes in its three layers, and the latter with 512 and 128 nodes in its layers. The learning rates for both networks were set to $1e-4$, the soft target update parameter τ was set to $1e-2$, the discount factor was set to 0.99 and the replay buffer size was set to be 100000. For PPO, the hyperparameters used were consistent with those used in Ray et al. [23]. The hyperparameters specific to the approach described here are $\rho = 0.95$ and threshold $t = 0.1$.

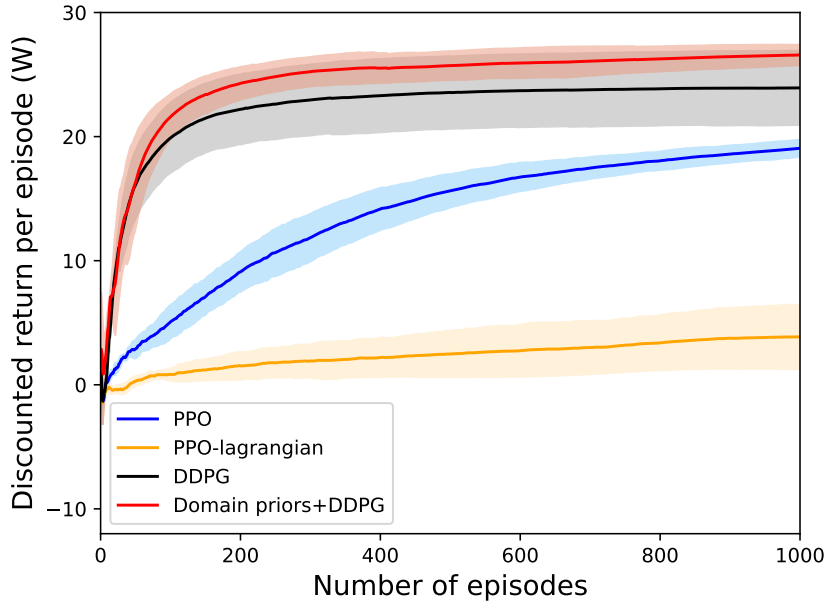

Figure 7: The average discounted returns per episode (W), computed over 3 trials, for different learning methods in the PointGoal1-12 Safety Gym environment.

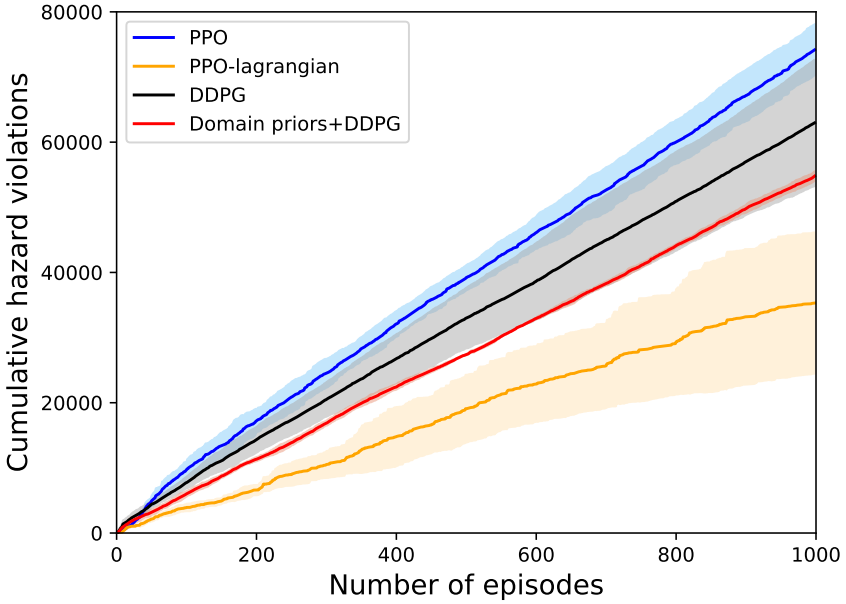

Figure 8: The cumulative number of obstacle collisions, computed over 3 trials, for different learning methods in the PointGoal1-12 Safety Gym environment.

As the environment contains a continuous action space, biasing the exploration exactly as described in Algorithm 2 is infeasible. In order to circumvent this issue, we randomly sampled 100 actions from a

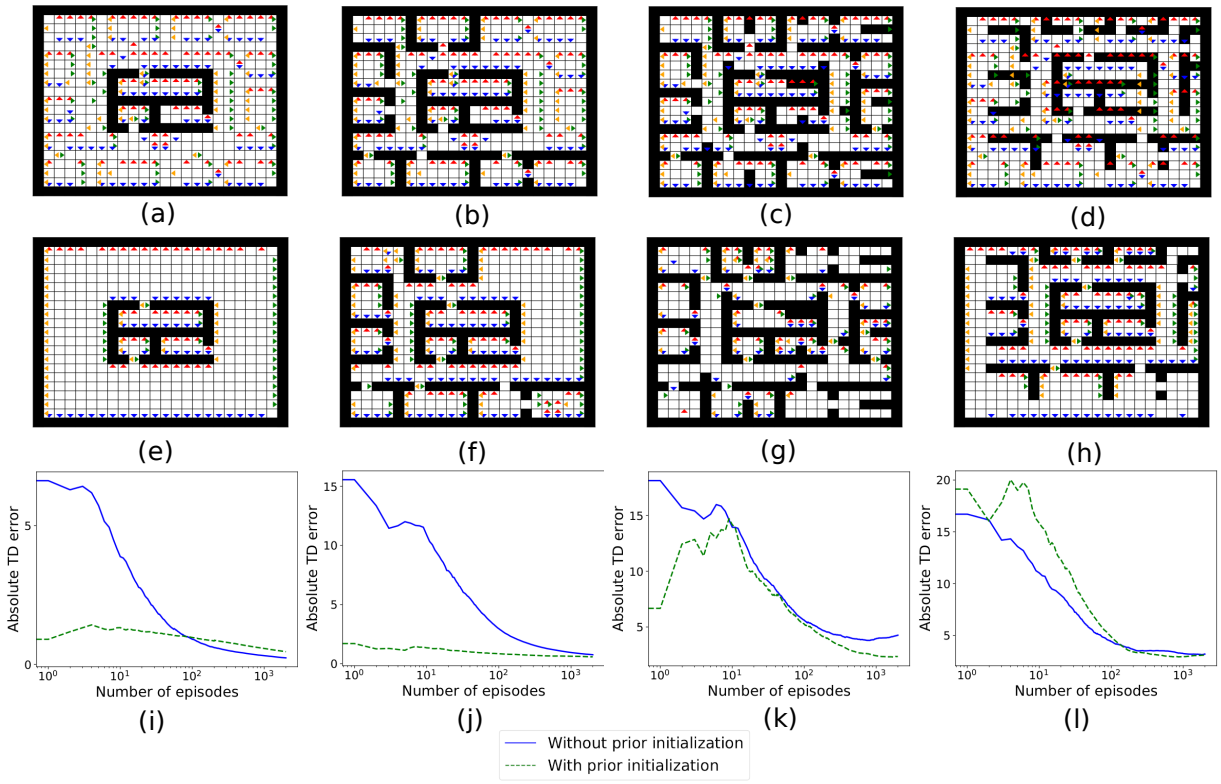

Figure 9: (a)-(d) show the consistently undesirable actions corresponding to the original environment in Figure 1(a), overlaid on top of four modified environments. (e)-(h) show these environments, with actions that are actually undesirable in them. (i)-(l) show the absolute TD errors associated with the learning of Q_P for these environments, with and without prior initialization.

uniform distribution in the allowable range of actions, $(-1, 1)$, essentially discretizing the action space. Following this, we proceeded to bias the actions as per Algorithm 2. The actions were biased with a probability proportional to an exploration bias factor, which started with an initial value of 1, and decayed exponentially by a factor of 0.95 at the end of each episode.

As depicted in Figures 7 and 8, the use of priors helps improve both learning as well as safety performance. As also noted in Ray et al. [23], although the learning performance of the PPO-Lagrangian approach is poor, it exhibits a much superior safety performance. However, it must be pointed out that this method has explicit access to a constraint violation function, while our approach does not.

5.4 Prior Adaptation to Modified Environments

As shown in Sections 4.1,4.2 and 4.3, learned priors can effectively help avoid undesirable exploratory actions while learning an arbitrary task in the domain. However, if the environment was to undergo a change in configuration, the set of actions associated with unsafe agent behaviors would not remain the same. Nevertheless, provided these changes are not too drastic, the priors learned from the original environment could still serve as a useful initialization for learning the corresponding priors in the modified environment. In other words, the priors may be transferable to the modified environments. This is an advantage that is specific to our approach, and is enabled by the fact that our priors are adaptive, and are inherently tied to the structure of the domain. In addition, the adaptive nature of the priors ensures that in time, they become well-suited to the modified environment, the with the adaptation time depending on the degree of dissimilarity between the two environments.

Here, we design experiments in the tabular environment in Section 4.1, to demonstrate this transferabil-

ity to modified versions of the original environment in Figure 1(a), shown in Figures 9(a)-(d). Obstacles were either added or removed from the original environment (Figure 1(a)) to obtain the modified environments in Figures 9(a)-(c), whereas the environment in Figure 9(d) was created by offsetting most obstacles 2 units upwards and to the right. The consistently undesirable actions for the original environment in Figure 1(a) are overlaid on top of the modified environments in Figures 9(a)-(d), whereas the correct set of consistently undesirable actions for the modified environments are shown in Figures 9(e)-(h). Despite the differences between the undesirable actions of the original and modified environments, there exists some structural similarity between them. Hence, it is reasonable to expect the priors learned in the original environment to be at least partially transferable to the modified environments. Specifically, we posit that the learned prior for the original environment forms a reasonable initial estimate for learning the corresponding priors in the modified environments, as long as the differences between the two are not drastic.

In order to test this hypothesis, the priors for the modified environments were learned with and without these initial estimates. In both cases, the associated absolute TD errors decrease, as shown in Figures 9(i)-(l), which demonstrates the capability of the priors to adapt to different environments. Figures 9(i)-(k) suggest that initialization of the priors could lead to significantly lowered initial absolute TD errors compared to the case of learning the priors from scratch (without initialization). However, initializing the priors in this manner was not found to be useful for the environment in Figure 9(d), where the effect of the initialization was to slightly increase the initial absolute TD error, as depicted in Figure 9(l). This is due to the fact that the nature of the differences in the obstacle configuration in Figure 9(d) and Figure 1(a) renders the prior learned in the latter ineffective with respect to learning the prior in the former. These experiments demonstrate that while the prior learned using the described approach is transferable to some extent, it is not transferable in general.

6 Discussion

The proposed methodology allows RL agents avoid undesirable actions during learning by making use of a learned prior policy. Although our approach as described, deals with avoiding undesirable actions, it can be easily adapted to scenarios where there exist actions that are commonly desirable across the tasks in the domain. Such an adaptation would involve replacing the advantage $A_i^*(s,a)$ with $B_i^*(s,a) = Q_i^*(s,a) - \underset{a' \in \mathcal{A}}{min} Q_i^*(s,a')$, in addition to replacing Equation 1 with $w_i(s,a) = \left| \frac{B_i^*(s,a)}{\underset{a' \in \mathcal{A}}{max}\, Q_i^*(s,a')} \right|$. The resulting prior could then simply be used to guide exploration, by taking exploratory actions that are greedy with respect to Q_P^* with a high probability. Such an approach appeared to be successful in versions of the tabular environment (similar to that described in Section 4.1) where a non-goal, rewarding state was introduced into all tasks in the domain. Although the approach is useful for such specific situations, in general, exploring the state-action space by greedily exploiting the prior in this manner could lead to poor learning performances, as it may limit the agent's exploration. Hence, achieving safe learning behaviors is a more practical use-case for the approach described in this work.

The ability to avoid undesired actions during learning makes the proposed approach potentially useful for real-world systems which are often intolerant of poor actions. Our approach would thus be useful in scenarios where the associated marginal increase in memory and computational costs are outweighed by the costs of executing unsafe actions.

Although we only consider cases where tasks vary solely in the reward function, this could lay the foundation for more general work, where tasks vary in other aspects such as the representation, transition function or the state-action space.

7 Conclusion

We presented a method to extract priors from a set of known tasks in the domain. The prior is learned in the form of a Q-function, and is based on inferred rewards corresponding to consistently undesirable actions across these tasks. The effectiveness of the prior in enabling safe learning behaviors was demonstrated in discrete as well as continuous environments, and its performance was compared to various baselines. This

was further supported by our theoretical analysis, which suggests that the use of these priors helps reduce the probability of taking unsafe exploratory actions. In addition to leading to safer learning behaviors for arbitrary tasks in the domain, the priors were shown to be transferable to some extent, and capable of adapting to changes in the environment.

References

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 22–31. JMLR. org, 2017.

[2] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[3] Haitham Bou Ammar, Rasul Tutunov, and Eric Eaton. Safe policy search for lifelong reinforcement learning with sublinear regret. In *International Conference on Machine Learning*, pages 2361–2369, 2015.

[4] Leemon C Baird. Advantage updating. Technical report, Wright Lab Wright-Patterson AFB OH, 1993.

[5] André Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Žídek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. *arXiv preprint arXiv:1901.10964*, 2019.

[6] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[8] Andrew Cohen, Lei Yu, and Robert Wright. Diverse exploration for fast and safe policy improvement. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[9] Rachit Dubey, Pulkit Agrawal, Deepak Pathak, Tom Griffiths, and Alexei Efros. Investigating human priors for playing video games. In *International Conference on Machine Learning*, pages 1348–1356, 2018.

[10] Fernándo Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 720–727. ACM, 2006.

[11] Javier Garcia and Fernando Fernández. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, 45:515–564, 2012.

[12] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[13] Matthieu Geist and Bruno Scherrer. Off-policy learning with eligibility traces: a survey. *Journal of Machine Learning Research*, 15(1):289–333, 2014.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. Ai safety gridworlds. *arXiv preprint arXiv:1711.09883*, 2017.

[16] Siyuan Li and Chongjie Zhang. An optimal online method of selecting source policies for reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[17] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[18] Chen Ma, Junfeng Wen, and Yoshua Bengio. Universal successor representations for transfer reinforcement learning. *arXiv preprint arXiv:1804.03758*, 2018.

[19] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[21] 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, 2015.

[22] Andrew Y. Ng, H. Jin Kim, Michael I. Jordan, and Shankar Sastry. Inverted autonomous helicopter flight via reinforcement learning. In *International Symposium on Experimental Robotics*. MIT Press, 2004.

[23] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning.

[24] Mark Bishop Ring. *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin Austin, Texas 78712, 1994.

[25] Simon Schmitt, Jonathan J Hudson, Augustin Zidek, Simon Osindero, Carl Doersch, Wojciech M Czarnecki, Joel Z Leibo, Heinrich Kuttler, Andrew Zisserman, Karen Simonyan, et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

[26] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[27] 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):484–489, 2016.

[28] Benjamin Spector and Serge Belongie. Sample-efficient reinforcement learning through transfer and architectural priors. *arXiv preprint arXiv:1801.02268*, 2018.

[29] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction, 2011.

[30] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[31] Gerald Tesauro. Temporal difference learning and td-gammon. *Commun. ACM*, 38(3):58–68, March 1995.

[32] CJCH Watkins. Learningfrom delayed rewards. *PhDthesis, Cambridge University, Cambridge, England*, 1989.

[33] Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3562–3573, 2018.

Supplementary material:

8 Application to common reward case

In domains where there exists a common, non-terminal rewarding state s_{com}, the proposed approach can be modified to positively bias the agent towards taking greedy actions with respect to the learned prior Q_P, as described in the discussion section. By doing so, we shift the focus of the algorithm to finding consistently desirable actions across the known tasks in this common reward environment. Here, we present one such environment, where in addition to the attributes of the environment in Figure 1 (a), there exists a non-terminal rewarding state s_{com} associated with a reward of 0.2, shown in Figure 10. In such a case, visiting state s_{com} becomes a desirable behavior across all tasks. Hence, the learned prior directs learning agents towards this state, as seen in Figure 11. Such a bias in the exploration policy is also reflected in the performance of the agent, as depicted in Figure 12.

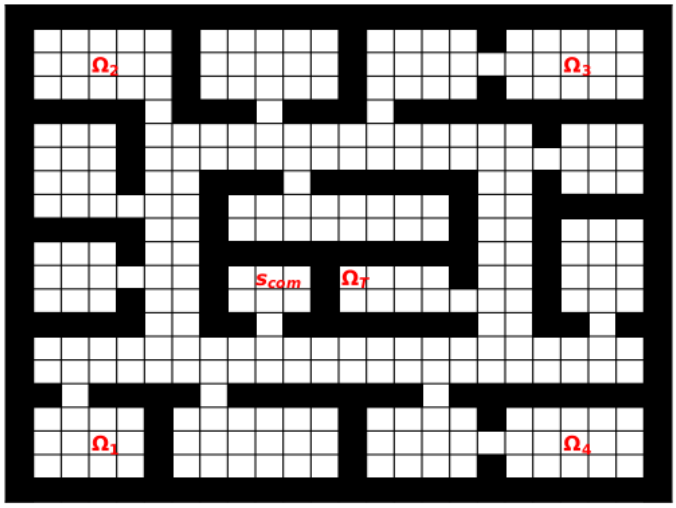

Figure 10: Navigation environment showing the goal locations $\Omega_1, \Omega_2, \Omega_3, \Omega_4$ of the known tasks, common rewarding state s_{com} and goal location Ω_{T} of the task to be learned.

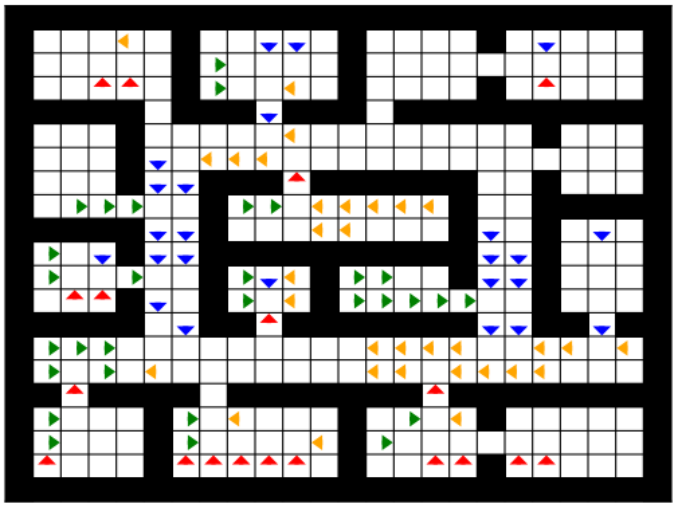

Figure 11: Identified desirable actions for the common reward environment in Figure 10.

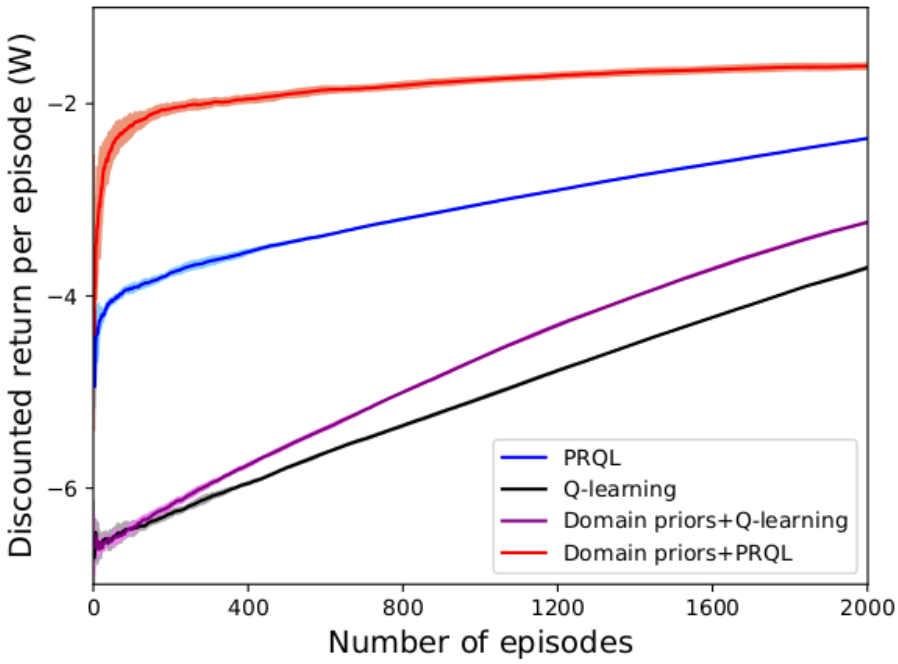

Figure 12: The average discounted return per episode (W), computed over 10 trials, for different learning methods.