Towards Safe Exploration using Demonstrations

André Correia
andre.correia@ubi.pt
Luís A. Alexandre
lfbaa@ubi.pt

Departamento de Informática
Universidade da Beira Interior
NOVA LINCS
6201-001, Covilhã, Portugal

Abstract

Reinforcement learning algorithms require large data sets to train a working policy. This is achieved by exploring the state space with trial and error interactions. However, such algorithms disregard any safety concerns. This can lead to many problems such as damage to the agent and surrounding objects or beings. Demonstrations of the task contain crucial information which the agent can use before interacting with the environment. We propose to train a safety model using demonstrations to evaluate the safety of each state. The output of this model can then be used to enhance the reward function to promote safety. We evaluate our model on four tasks and compare the safety of three baselines with and without our enhancement. Results show that our model increases the safety and performance of underlying RL models.

1 Introduction

Reinforcement learning (RL) algorithms learn a task through trial and error interactions. The agent observes the state of the environment and selects an action. The agent then receives a reward associated with the quality of the interaction [5]. The task is learned by estimating a policy that maximizes the expected accumulated rewards. To estimate a quality policy, the agent must explore the state space and try different action combinations for the different states.

The simplest form of exploration consists of adding a small random noise to the agent's action causing its trajectory to change and potentially discover a new state region [2]. Other methods separate exploration from task learning. During the exploration phase, they estimate the entropy of the visited state distribution and model the learning process towards its maximization [4, 6]. However, exploration methods disregard any safety concerns. Exploration implies testing different trajectories. Some trajectories are dangerous for the agent to take, such as dropping liquids on its circuits or colliding with surrounding objects and humans. Safe exploration of the state space remains an open problem in RL. Existing algorithms tackle safety by specifying constraints that the agent can not violate during execution [1]. However, these constraints are specific to a certain task and agent, and specifying all the constraints for each situation is impractical. Other approaches require an existing expert policy for the task [3].

The goal of this work is to create an algorithm that can enhance standalone RL algorithms by increasing their safety during exploration. Before training the RL agent, a model is trained using demonstrations, to determine the safety of interactions. The safety of the interaction is used to transform the task's reward signal for the agent to maximize. We evaluate the performance and safety of three baseline algorithms with and without the safety model on four tasks from the MuJoCo environment.

2 Methodology

We aim to create an algorithm that can be paired with an off-the-shelf RL algorithm to enhance its safety. Safety can be measured in different ways. An episode is a set of interactions between the agent and the environment. An episode can terminate if the agent successfully completes the task or if the agent performed a catastrophic action that caused the episode to terminate. In this work, we focus on preventing the agent from performing dangerous actions that cause the episode to end abruptly. Hence, we measure the safety of an algorithm by the length of the episodes.

To encourage the agent to increase the length of the episodes we provide it with an extra reward signal. This signal measures the safety of the agent's state. We define the safety of a state using the following formula: $Safety(s_i) = \log(N+\varepsilon-i)/\log(N+\varepsilon)*N/C$, where s_i is the i-th state in a trajectory $\tau = \{s_0, s_1, ..., s_N\}$, N is the length of the trajectory, ε is a small constant to avoid division by zero which we set to $1e^{-10}$ and C is a positive normalizing constant, which we set to 500 for the MuJoCo's environments because it is the maximum length of their episodes. The left hand side of the multiplication gives a reward based on how early in the trajectory a state occurred on a logarithmic scale. Later states are closer to the last state which ended the trajectory, therefore they are considered less safe. The division ensures that the first term is weighted regardless of the length of the trajectory. The right hand side of the multiplication rewards longer trajectories.

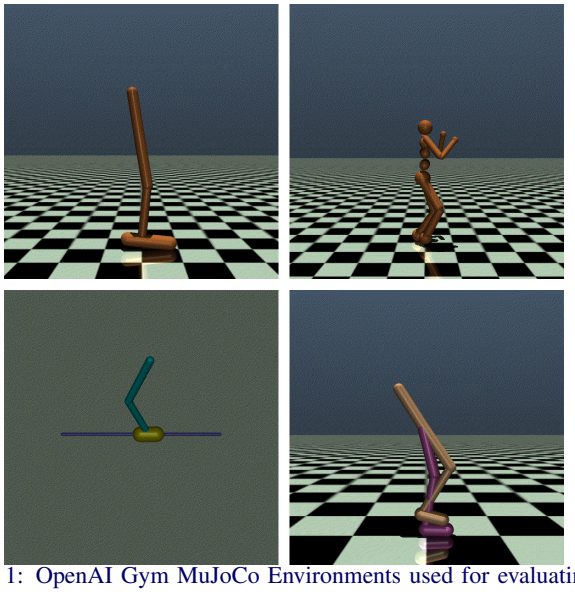

Figure 1: OpenAI Gym MuJoCo Environments used for evaluating the methods: Hopper, Humanoid, Inverted Double Pendulum and Walker2D.

We train a neural network, with parameters θ, using L2 loss to regress the safety value of a state. The network has two hidden layers each with 256 neurons as the safety model. We train this model alongside the RL agent using the same batches of data. Before updating the RL agent, the safety model predicts the safety values for the state batch. Then, we transform the environment's reward with the output of the model: $reward(s_i) = reward_i * (1 + \alpha * \theta(s_i))$, where α is a safety coefficient which we linearly decrease throughout training. At the beginning of training, the model is initialized with random values hence its output will not contribute much towards safety. Because of this, we pre-train the safety model on a data set of demonstrations of the task. A demonstration is a pre-recorded trajectory of N transitions performed. Each transition contains a state, action and reward $\tau = \{(s_0, a_0, r_0), ..., (s_{N-1}, a_{N-1}, r_{N-1})\}$. We use 100 demonstrations for each task and use 10 of them for validation. The task demonstrations are not optimal. We save the best model in the validation set to avoid over-fitting on the small training set.

3 Experiments

We evaluate the algorithms using four tasks from OpenAI Gym Mujoco's set: Hopper, Humanoid, Inverted Double Pendulum, and Walker-2D. We selected these tasks due to a set of reasons. The input is non-visual, allowing the focus to be on learning the task safely and disregarding feature estimation. Additionally, an episode can achieve 500 steps but can end abruptly if the agent performs an unsafe action. We can measure the safety of an episode by its length. A longer episode corresponds to a longer sequence of interactions where the agent did not perform unsafe actions. Hence, the algorithms are evaluated using the accumulated rewards and the length of the trajectories as performance and safety metrics, respectively.

We use three algorithms as baselines. Soft-Actor Critic (SAC) [2],

	Hopper			Humanoid			InvertedDoublePendulum			Walker2d		
	Acc Reward	Exp. Length	Task Length	Acc Reward	Exp. Length	Task Length	Acc Reward	Exp. Length	Task Length	Max. Acc Reward	Exp. Length	Task Length
SAC	**3659**	241	486	826	24	38	**9360**	196	498	2671	412	493
SAC+Ours	3539	**386**	**487**	**1387**	**49**	**68**	**9360**	**296**	**499**	**2853**	**438**	**495**
Proto	3322	415	415	**4877**	38	103	**9360**	**4**	**394**	2763	**349**	465
Proto+Ours	**3428**	**450**	**447**	4664	**45**	**136**	**9360**	**4**	392	**3851**	339	**468**
L2E	3056	93	**427**	129	8	7	3882	2	21	1236	67	**410**
L2E+Ours	**3576**	**100**	404	**189**	**11**	**16**	**9360**	**3**	**161**	**1374**	**151**	391

Table 1: Accumulated rewards, average episode length during the exploration phase and average episode length during the task phase of the baseline algorithms with and without our safety enhancement for the four MuJoCo tasks.

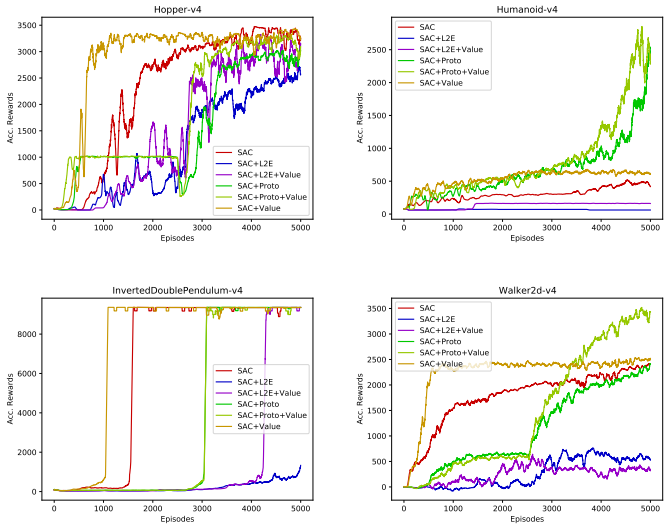

Figure 2: Accumulated rewards over 5000 learning episodes for the 3 baselines with and without our method.

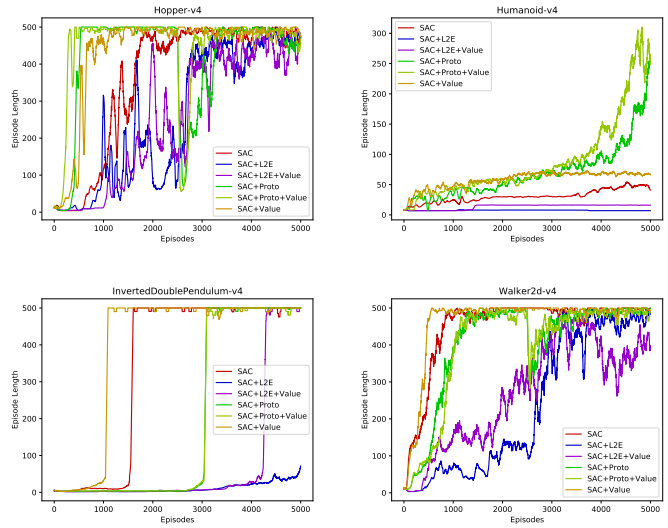

Figure 3: Episode lengths throughout training for 5000 episodes for the 3 baselines with and without our method.

is the underlying RL algorithm to be augmented by the different models. We then use Proto [6] and Learn2Explore (L2E) [4], which augment SAC with an exploration goal. The exploration goal is to maximize the estimated entropy of the explored state space. Proto clusters the state space using a set of centroids. The entropy is estimated using the distance of the current state to the nearest centroid. In L2E, the entropy is estimated by the variance of predictions given by an ensemble of neural networks.

We divide training into two phases as in [4, 6]: exploration and task learning. In the exploration phase, the agent is encouraged to search the state space while in the task learning phase it is encouraged to maximize the expected rewards. We train the agents for 2500 episodes in each phase. For the safety augmented algorithms, we pre-train the safety model beforehand for 1000 epochs on 90 demonstrations and validate the model every 100 epochs using 10 demonstrations to avoid over-fitting. Lastly, we linearly decrease α from 1 to 0 during the exploration phase.

The accumulated rewards and episode lengths throughout training for the different algorithms across the four different tasks are shown in Fig. 2 and Fig.3, respectively. Additionally, we present values of the average accumulated rewards and average episode lengths for the exploration and task phases in Table 3. Results show that the safety enhancement increases the length of the episodes during the exploration phase by 55% for the standalone SAC and for the SAC with Proto, and by 6% for the SAC with L2E. This advantage comes at no clear cost of performance. In most settings, the accumulated rewards of algorithms enhanced with the safety model surpass their baseline. This indicates that the safety model successfully transforms the reward to promote the safety of the agent during training while additionally increasing its learning capabilities.

4 Conclusion

We present a method that can enhance off-the-shelf RL algorithms by increasing their safety during training. We pre-train a model on demonstrations to predict the safety value of a state. The output of the model is used to transform the environment's reward to promote safety. We evaluate the proposed method by training three baseline algorithms and evaluating their performance and safety during training with and without our method. Results show that our method successfully increases the safety of the algorithms by generating longer trajectories during training. This increase in safety increased the performance of the agent in most settings.

Acknowledgements

This work was supported by NOVA LINCS (UIDB/04516/2020) with the financial support of FCT-Fundação para a Ciência e a Tecnologia, through national funds.

References

[1] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

[2] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[3] Kunal Menda, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Ensembledagger: A bayesian approach to safe imitation learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5041–5048. IEEE, 2019.

[4] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.

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

[6] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pages 11920–11931. PMLR, 2021.