SAAC: Safe Reinforcement Learning as an Adversarial Game of Actor-Critics

Yannis Flet-Berliac[*]
Stanford University
Stanford, CA, USA
yfletberliac@stanford.edu

Debabrota Basu
Équipe Scool, Inria Lille- Nord Europe
Université de Lille, CNRS, Centrale Lille, UMR 9189 – CRIStAL
F-59000 Lille, France
debabrota.basu@inria.fr

Abstract

Although Reinforcement Learning (RL) is effective for sequential decision-making problems under uncertainty, it still stumbles to thrive in real-world systems where *risk* or *safety* is a binding constraint. In this paper, we formulate the RL problem with safety constraints as a non-zero-sum game. While deployed with maximum entropy RL, this formulation leads to a soft adversary guided soft actor-critic framework, called SAAC. In SAAC, the adversary aims to break the safety constraint while the RL agent aims to maximize the constrained value function given the adversary's policy. In SAAC, the safety constraint on the agent's value function manifests only as a repulsion term between the agent's and the adversary's policies. Unlike previous approaches, SAAC can address different safety criteria such as safe exploration, mean-variance risk sensitivity, and CVaR-like coherent risk sensitivity. We illustrate the design of the adversary for these constraints. Then, in each of these variations, we show the agent differentiates itself from the adversary's unsafe actions in addition to learning to solve the task. Finally, for challenging continuous control tasks, we demonstrate that SAAC achieves faster convergence, better efficiency, and less number of failures to satisfy the safety constraints than the risk-averse distributional RL and risk-neutral soft actor-critic algorithms.

Keywords: Safe reinforcement learning, Maximum entropy RL, Constrained Markov decision processes, Soft actor-critic

Acknowledgements

...

[*] This work was done during Yannis's PhD at Scool, Inria Lille.

1 Introduction

Designing a Reinforcement Learning (RL) algorithm requires both efficient quantification of uncertainty regarding the incomplete information and the probabilistic decision making policy, and effective design of a policy that can leverage these quantifications to achieve optimal performance. Instead of recent success of RL in structured games and simulated environments, real-world deployment of RL in industrial processes, unmanned vehicles, robotics etc., does not only require efficiency in terms of performance but also being sensitive to risks involved in decisions [2]. This has propelled works quantifying risks in RL and designing safe (or robust, or risk-sensitive) RL algorithms [10, 4].

RL Formalization: Markov Decision Process (MDP). We consider the RL problems that can be modelled as a *Markov Decision Process (MDP)*. An MDP is defined as a tuple $\mathcal{M} \triangleq (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma)$. $\mathcal{S} \subseteq \mathbb{R}^d$ is the *state space*. $\mathcal{A}$ is the admissible *action space*. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the *reward function* that quantifies the goodness or badness of a state-action pair (s, a). $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta_{\mathcal{S}}$ is the *transition kernel* that dictates the probability to go to a next state given the present state and action. Here, $\gamma \in (0, 1]$ is the *discount factor* that quantifies the effect of the reward at present step to the next one. The goal of the agent is to compute a *policy* $\pi : \mathcal{S} \rightarrow \Delta_{\mathcal{A}}$ that maximizes the expected value of cumulative rewards obtained by a time horizon $T \in \mathbb{N}$. For a given policy π, the *value function* or the expected value of discounted cumulative rewards is $V_\pi(s) \triangleq \underset{s_t \sim \mathcal{T}(s_{t-1}, a_{t-1}), a_t \sim \pi(s_t)}{\mathbb{E}} \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t) | s_0 = s\right] \triangleq \mathbb{E}_{\pi \mathcal{M}}[Z_\pi^T(s)]$. Here, $Z_\pi^T(s)$ is the *return* of policy π.

Safe RL: Safe Exploration and Risk Measures. In safe RL, risk-sensitivity or safety is embedded mainly using two approaches. The first approach is constraining the RL algorithm to converge in a restricted, 'safe' region of the state space [5, 10]. Here, the 'safe' region is the part of the state space that obeys some external risk-based constraints, such as the non-slippery part of the floor for a walker. RL algorithms developed using this approach either try to construct policies that generate trajectories which stay in this safe region with high probability [5], or to start with a conservative 'safe' policy and then to incrementally estimate the maximal safe region [1]. Due to existence of these 'error' or 'unsafe' states, even a policy with low variance can produce large risks (e.g. falls or accidents) [10].

The other approach is to define a risk-measure on the return $Z_\pi^T(s)$ of a policy π, and then to minimize the corresponding total risk [7, 9, 3]. A risk-measure is a statistics computed on the cumulative return and it quantifies either the spread of the return distribution around its mean value or the heaviness of this distribution's tails. Example of such risk measures are conditional value-at-risk (CVaR) [3], exponential utility [7], variance [9], etc. At tandem to investigating the risk quantifiers, researchers aimed to make the safe RL algorithms scalable [3] and to extend to the continuous MDPs [10]. Our approach is flexible to consider all these risk measures and both discrete and continuous MDP settings.

Our Contributions. In this paper, we unify both of these approaches as a constrained RL problem, and further derive an equivalent non-zero sum (NZS) stochastic game formulation [11] of it. In our NZS game formulation, *risk-sensitive RL reduces to a game between an agent and an adversary* (Sec. 2). The adversary tries to break the *safety constraints*, i.e. either to move out of the 'safe' region or to increase the risk measures corresponding to a given policy. In contrast, the agent tries to construct a policy that maximizes its expected long-term return given the adversarial feedback, which is a statistics computed on adversary's constraint breaking. Given this formulation, we propose a generic actor-critic framework where any two compatible actor-critic RL algorithms are employed to enact as the agent and the adversary to ensure risk-sensitive performance (Sec. 3). In order to instantiate our approach, we propose a specific algorithm, *Safe Adversarially guided Actor-Critic* (`SAAC`), that deploys two Soft Actor-Critics (SAC) [6] as the agent and the adversary. We further derive the policy gradients for the SACs corresponding to the agent and the adversary, which shows that the risk-sensitivity of the agent is ensured by a term repulsing it from the adversary in the policy space. Interestingly, this term can also be used to seek risk and explore more. In Sec. 4, we experimentally verify the risk-sensitive performance of `SAAC` under safe region, CVaR, and variance constraints for continuous control tasks from real-world RL suite [2]. We show that `SAAC` is not only risk-sensitive but it outperforms the state-of-the-art risk-sensitive RL and distributional RL algorithms.

2 Safe RL as a Non-Zero Sum Game

Safe RL as Constrained MDP (CMDP). Both the safe exploration and risk-measure based approaches can be expressed as a CMDP that aims to maximize the value function V_π of a policy π while constraining the total risk ρ_π below a threshold δ:

$$\arg\max_{\pi} V_\pi(s) \text{ s.t. } \rho_\pi(s) \leq \delta \text{ for } \delta > 0. \tag{1}$$

If Mean-Standard Deviation (MSD) [9] is the risk measure[1], $\rho_\pi(s) \triangleq \mathbb{E}\left[Z_\pi^T(s)|\pi, s_0 = s\right] + \lambda\sqrt{\mathbb{V}\left[Z_\pi^T(s)|\pi, s_0 = s\right]}$ $(\lambda < 0)$. If CVaR is the risk measure, $\rho_\pi(s) \triangleq \mathrm{CVaR}_\lambda\left[Z_\pi^T(s)|\pi, s_0 = s\right]$ for $\lambda \in [0, 1)$. For the constraint of staying in the 'safe' or 'non-error' states $\mathcal{S} \setminus \mathcal{E}$, $\rho_\pi(s) \triangleq \mathbb{E}\left[\sum_{t=0}^{T} \mathbb{1}(s_{t+1} \in \mathcal{E})|\pi, s_0 = s \in \mathcal{S} \setminus \mathcal{E}\right] = \sum_{t=0}^{T} \mathbb{P}_\pi[s_{t+1} \in \mathcal{E}]$ such that $s_0 = s$ is a non-error state and $\mathcal{E}$ is the set of 'unsafe' or 'error' states. We refer to this as *subspace risk* $\mathrm{Risk}(\mathcal{E}, \mathcal{S})$ for $\mathcal{E} \subseteq \mathcal{S}$.

CMDP as a Non-Zero Sum (NZS) Game. We address the constraint optimization in Eq. (1) by formulating its Lagrangian.

$$\mathcal{L}(\pi, \beta) \triangleq V_\pi(s) - \beta_0 \rho_\pi(s), \text{ for } \beta_0 \geq 0. \tag{2}$$

[1] Variance is not a coherent risk but standard deviation is. Thus, we choose to use Mean-Standard Deviation than Mean-Variance.

For $\beta_0 = 0$, this reduces to its risk-neutral counterpart. Instead, as $\beta_0 \to \infty$, this reduces to the unconstrained risk-sensitive approach. Thus, the choice of β_0 is important. We automatically tune this trade-off in our proposed method. Now, the important question is to estimate the risk function $\rho_\pi(s)$. Researchers have either solved an explicit optimization problem to estimate the parameter or subspace corresponding to the risk measure, or used a stochastic estimator of the risk gradients. These approaches are poorly scalable and lead to high variance estimates as there is no provably convergent CVaR estimator in RL settings. In order to circumvent these issues, we aim to sequentially learn and then adapt to these constraints. Specifically, we deploy *an adversary* that aims to maximize the cumulative risk $\rho_\pi(s)$ given the same initial state s and trajectory τ as *the agent* maximizing Eq. (2) and use it as a proxy for the risk constraint.

$$\theta^* \triangleq \arg\max_\theta \mathcal{L}(\theta, \beta) = V_{\pi_\theta}(s) - \beta_0 V_{\pi_\omega}(s), \qquad \omega^* \triangleq \arg\max_\omega V_{\pi_\omega}(s). \tag{3}$$

Here, we consider that the policies of the agent and the adversary are parameterized by θ and ω respectively. The value function of the adversary $V_{\pi_\omega}(s, \cdot)$ is designed to estimate the corresponding risk $\rho_\pi(s)$. This is a non-zero sum game (NZS) as the objectives of the adversary and the agent are not the same and does not sum up to 0. Following this formulation, any safe RL problem expressed as a CMDP (Eq. (1)), can be reduced to a corresponding agent-adversary non-zero sum game (Eq. (3)). The adversary tries to maximize the risk, and thus to shrink the feasibility region of the agent's value function. The agent tries to maximize the regularized Lagrangian objective in the shrinked feasibility region. We refer to this duelling game as *Risk-sensitive Non-zero Sum (RNS)* game.

3 SAAC: Safe Adversarial Soft Actor-Critics

Background: Maximum-Entropy RL. In this paper, we adopt the Maximum-Entropy RL (MaxEnt RL) framework [4], also known as entropy-regularized RL. In MaxEnt RL, we aim to maximize the sum of value function and the conditional action entropy, $\mathcal{H}_\pi(a|s)$, i.e. for a policy π: $\arg\max_\pi \quad V_\pi(s) + \mathcal{H}_\pi(a|s) = \mathop{\mathbb{E}}_{s_t \sim \mathcal{T}(s_{t-1}, a_{t-1}), a_t \sim \pi(s_t)} \left[Z^T_\pi(s) - \log \pi(a_t|s_t) \mid s_0 = s \right]$. Unlike the classical value function maximizing RL that always has a deterministic policy as a solution, MaxEnt RL tries to learn stochastic policies such that states with multiple near-optimal actions has higher entropy and states with single optimal action has lower entropy. Solving MaxEnt RL is equivalent to computing a policy π that has minimum KL-divergence from a target trajectory distribution $\mathcal{T} \circ \mathcal{R}$:

$$\arg\max_\pi V_\pi(s) + \mathcal{H}_\pi(a|s) = \arg\min_\pi D_{\mathrm{KL}} \left(\pi(\tau) \parallel \mathcal{T} \circ \mathcal{R}(\tau) \right). \tag{4}$$

Here, τ is a trajectory $\{(s_0, a_0), \ldots, (s_T, a_T)\}$. Target distribution $\mathcal{T} \circ \mathcal{R}$ is a softmax or Boltzmann distribution on the cumulative rewards given the trajectory: $\mathcal{T} \circ \mathcal{R}(\tau) \propto p_0(s) \prod_{t=0}^T \mathcal{T}(s_{t+1}|s_t, a_t) \exp[Z^T_\pi(s)]$. Policy distribution is the distribution of generating trajectory τ given the policy π and MDP $\mathcal{M}$: $\pi(\tau) \propto p_0(s) \prod_{t=0}^T \mathcal{T}(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$. Thus in MaxEnt RL, the optimal policy is a softmax or Boltzmann distribution over the expected future return of state-action pairs.

This perspective of MaxEnt RL allows us to design `SAAC` which transforms the robust RL into an adversarial game in the softmax policy space. MaxEnt RL is widely used in solving complex RL problems as: it enhances exploration [6], it transforms the optimal control problem in RL into a probabilistic inference problem [12], and it modifies the optimization problem by smoothing the value function landscape.

Risk-sensitive Non-zero Sum (RNS) Game with MaxEnt RL. In order to perform the RNS game with MaxEnt RL, we substitute the Q-values in Eq. (3) with corresponding soft Q-values. Thus, the adversary's objective is maximizing:

$$\omega^* = \arg\max_\omega \mathbb{E}_{\pi_\omega}[Q_\omega(s, \cdot)] + \alpha_0 \, \mathcal{H}_{\pi_\omega}(\pi_\omega(.|s)) = \arg\min_\omega D_{\mathrm{KL}} \left(\pi_\omega(.|s) \, \big\| \, \exp\left(\alpha_0^{-1} Q_\omega(s, \cdot)\right) / Z_\omega(s) \right). \tag{5}$$

for $\pi_\omega \in \Pi_\omega$, and the agent's objective is maximizing:

$$\begin{aligned} \theta^* &= \arg\max_\theta \quad \mathbb{E}_{\pi_\theta}[Q_\theta(s, \cdot)] + \alpha_0 \, \mathcal{H}_{\pi_\theta}(\pi_\theta(.|s)) - \beta_0(\mathbb{E}_{\pi_\theta}[Q_\omega(s, \cdot)] + \alpha_0 \, \mathcal{H}_{\pi_\omega}(\pi_\omega(.|s))) \qquad (6) \\ &= \arg\min_\theta D_{\mathrm{KL}} \left(\pi_\theta(.|s) \, \big\| \, \exp\left(\alpha^{-1} Q_\theta(s, \cdot)\right) / Z_\theta(s) \right) - \beta D_{\mathrm{KL}} \left(\pi_\theta(\cdot|s) \parallel \pi_{\omega^*}(\cdot|s) \right). \qquad (7) \end{aligned}$$

for $\pi_\theta \in \Pi_\theta$. Here, $\alpha = \alpha_0(1 + \beta_0)$ and $\beta = \alpha_0 \beta_0$. The last equality holds true as $\pi_{\omega^*}(.|s) = \exp\left(\alpha_0^{-1} Q_{\omega^*}(s, \cdot)\right) / Z_{\omega^*}(s)$ for the adversary's optimal policy π_{ω^*}, and since the optimization is over θ, adding $\ln Z_\omega(s)$ does not make a change.

Additionally, for $\omega \neq \omega^*$, the relaxed objective $-(D_{\mathrm{KL}} \left(\pi_\theta(.|s) \, \big\| \, \exp\left(\alpha^{-1} Q_\theta(s, \cdot)\right) / Z_\theta(s) \right) - \beta D_{\mathrm{KL}} \left(\pi_\theta(\cdot|s) \parallel \pi_\omega(\cdot|s) \right))$ is a strict lower bound of the goal of the agent in Eq. (6). Thus, maximizing it is similar to maximizing the lower bound on the actual objective. This is similar to the general EM algorithms for maximising likelihoods. Thus, not only in asymptotics, but at every step optimizing the reduced objective allows to maximize the agent's risk-sensitive soft Q-value.

Following this reduction, we observe that performing the RNS game with MaxEnt RL is equivalent to performing the traditional MaxEnt RL for adversary with a risk-seeking Q-function Q_ω, and a modified MaxEnt RL for the agent that includes the usual soft Q-function and a KL-divergence term repulsing the agent's policy π_θ from the adversary's policy π_ω. This behaviour of RNS game in policy space allows to propose a duelling soft actor-critic algorithm, namely `SAAC`.

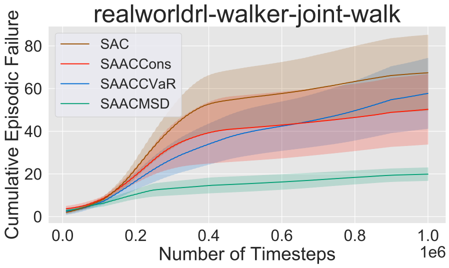

Figure 1: Constraints: `SAAC` variants.

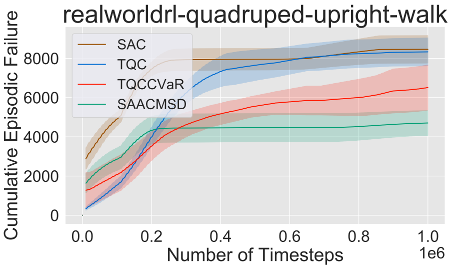

Figure 2: Constraints: `SAAC` vs. baselines.

Figure 3: In *walker-joint-walk*.

Method	Efficiency (xSAC)	# Failures $\pm\sigma$
SAC	$\times 1$	65.88 ± 17.25
`SAAC-Cons`	$\times 1.33$	48.66 ± 15.99
`SAAC-CVaR`	$\times 2.02$	54.39 ± 15.37
`SAAC-MSD`	$\times\mathbf{2.21}$	$\mathbf{19.31 \pm 3.02}$

Figure 4: In *quadruped-upright-walk*.

Method	Efficiency (xSAC)	# Failures $\pm\sigma$
SAC	$\times 1$	8443.93 ± 696.47
TQC	$\times 0.97$	8297.63 ± 697.88
TQC-CVaR	$\times 1.03$	6298.33 ± 1078.50
`SAAC-MSD`	$\times\mathbf{1.19}$	$\mathbf{4632.80 \pm 657.35}$

The `SAAC` Algorithm. We propose an algorithm `SAAC` to solve the objectives of the agent (Eq. (6)) and of the adversary (Eq. (5)). In `SAAC`, we deploy two soft actor-critics (SACs) to enact the agent and the adversary respectively.

As a building block for `SAAC`, we deploy the recent version of SAC [6] that uses two soft Q-functions to mitigate positive bias in the policy improvement step. In the design of `SAAC`, we introduce two new ideas: an off-policy deep actor-critic algorithm within the MaxEnt RL framework and a Risk-sensitive Non-zero Sum (RNS) game. `SAAC` engages the agent in safer strategies while finding the optimal actions to *maximize* the expected returns. The role of the adversary is to find a policy that maximizes the probability of breaking the constraints given by the environment. The adversary is trained online with off-policy data given by the agent. We denote the parameter of the adversary policy using ω^2. For each sequence of transition from the replay buffer, the adversary should find actions that minimize the following loss:

$$J(\pi_\omega) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\mathbb{E}_{a_t \sim \pi_\omega} \left[\alpha \log \left(\pi_\omega \left(a_t | s_t \right) \right) - Q_\psi \left(s_t, a_t \right) \right] \right].$$

Finally, leveraging the RNS based reduced objective, `SAAC` makes the agent's actor minimize $J(\pi_\theta)$:

$$J(\pi_\theta) = \mathbb{E}_{s_t \sim \mathcal{D}} \Big[\mathbb{E}_{a_t \sim \pi_\theta} \Big[\alpha \log \left(\pi_\theta \left(a_t | s_t \right) \right) - Q_\phi \left(s_t, a_t \right) - \beta \Big(\log \pi_{\theta_{\text{old}}} (a_t | s_t) - \log \pi_{\omega_{\text{old}}} (a_t | s_t) \Big) \Big] \Big].$$

In blue is the repulsion term introduced by `SAAC`. The method alternates between collecting samples from the environment with the current agent's policy and updating the function approximators, namely the adversary's critic Q_ψ, the adversary's policy π_ω, the agent's critic Q_ϕ and the agent's policy π_θ. It performs stochastic gradient descent on corresponding loss functions with batches sampled from the replay buffer. Now, we provide a few examples of designing the adversary's critic Q_ψ for different safety constraints.

`SAAC-Cons`: *Subspace Risk.* At every step, the environment signals whether the constraints have been satisfied or not. We construct a reward signal based on this information. This constraint reward, denoted as r_c, is 1 if all the constraints have been broken, and 0 otherwise. $J(Q_\psi)$ is the soft Bellman residual for the critic responsible with constraint satisfaction:

$$J(Q_\psi) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}} \Big[\frac{1}{2} \Big(Q_\psi \left(s_t, a_t \right) - \big(r_c \left(s_t, a_t \right) + \gamma \mathbb{E}_{s_{t+1} \sim \rho} \mathbb{E}_{a_t \sim \pi_\omega} \left[Q_{\bar{\psi}} \left(s_t, a_t \right) - \alpha \log \pi \left(a_t | s_t \right) \right] \Big)^2 \Big]. \tag{8}$$

`SAAC-MSD`: *Mean-Standard Deviation (MSD).* In this case, we consider optimizing a Mean-Standard Deviation risk [9], which we estimate using: $Q_\psi(s,a) = Q_\phi(s,a) + \lambda \sqrt{\mathbb{V}[Q_\phi(s,a)]}$. $\lambda < 0$ is a hyperparameter that dictates the lower $\lambda-\mathrm{SD}$ considered to represent the lower tail. In the experiments, we use $\lambda = -1$. In practice, we approximate the variance $\mathbb{V}[Q_\phi(s,a)]$ using the state-action pairs in the current batch of samples. We refer to the associated method as `SAAC-MSD`.

`SAAC-CVaR`: *CVaR.* Given a state-action pair (s,a), the Q-value distribution is approximated by a set of quantile values at quantile fractions [3]. Let $\{\tau_i\}_{i=0,\dots,N}$ denote a set of quantile fractions, which satisfy $\tau_0 = 0$, $\tau_N = 1$, $\tau_i < \tau_j \, \forall i < j$, $\tau_i \in [0,1] \, \forall i = 0, \dots, N$, and $\hat{\tau}_i = \left(\tau_i + \tau_{i+1} \right)/2$. If $Z^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{Z}$ denotes the soft action-value of policy π, $Q_\psi(s,a) = -\sum_{i=0}^{N-1} \left(\tau_{i+1} - \tau_i \right) g' \left(\hat{\tau}_i \right) Z_{\hat{\tau}_i}^{\pi_\theta}(s, a; \phi)$ with $g(\tau) = \min\{\tau/\lambda, 1\}$, where $\lambda \in (0,1)$. In the experiments, we set $\lambda = 0.25$, i.e. we truncate the right tail of the return distribution by dropping 75% of the topmost atoms.

4 Experimental Analysis

Experimental Setup. To validate the proposed framework, we conduct a set of experiments in the real-world RL challenge [2], such as *realworldrl-walker-joint-walk*, and *realworldrl-quadruped-upright-walk*. Note that for all the experiments, the agents are trained for 1M timesteps and their performance is evaluated at every 1000-th step. Similar to [6], the adversary temperature β and the entropy temperature are automatically adjusted.

Comparison between Risk Quantifiers of `SAAC`. First, we compare the different variants of `SAAC` allowed by the method's framework in the *realworldrl-walker-joint-walk* task. From Table 3 and Fig. 1 (lines are average performances and shaded

[2] resp. ω_{old} the parameter at the previous iteration.

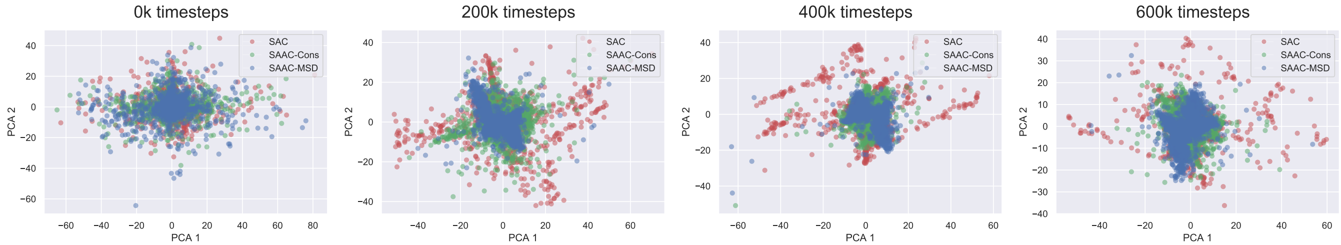

Figure 5: Visualization of visited state space at different stages of learning in the *realworldrl-walker-joint-walk* task.

areas represent one standard deviation), we evaluate how our method affects the performance and risk aversion of agents. In addition to the rate at which the maximum average return is reached by each of the methods compared to SAC, we compare the cumulative number of failures of the agents (the lower the better). Risk-sensitive agents such as SAAC decrease the probability of breaking safety constraints. Concurrently, they *achieve the maximum average return with higher sample efficiency and* SAAC-MSD *ahead*. Henceforth, we use the SAAC-MSD version of our method to compare with the baselines.

Comparison of SAAC to Baselines. Now, we compare the best performing SAAC variant SAAC-MSD with SAC [6], TQC [8] and TQC-CVaR, i.e. an extension of TQC with 16% of the topmost atoms dropped (cf. Table 6 in [8, Appendix B]) of all Q-function atoms. TQC-CVaR leverages distributional RL with risk measure to obtain safer policies. In Table 4 and Fig. 2, we evaluate SAAC-MSD in *realworldrl-quadruped-upright-walk*. Table 4 confirms the advantage of using SAAC-MSD as a risk-averse MaxEnt RL method over the risk-neutral MaxEntRL and risk-averse distributional RL baselines. SAAC *allows the agents to achieve faster convergence, using safer policies during training, better efficiency (*$\sim 1.19\times$ *more than SAC), and less number of failures to satisfy the safety constraints (*~ 26 *to* 45% *less)*.

Visualization of Safer State Space Visitation. In this experiment, we choose SAC, SAAC-Cons and SAAC-MSD to train a relatively wide spectrum of agents using the same experimental protocol as in Sec. 5.2., and on the *realworldrl-walker-joint-walk* task. We collect samples of states visited during the evaluation phase in a test environment at different stages of the training. The state vectors are projected from a 18D space to a 2D space using PCA. We present the results in Fig. 5. At the beginning of training, there is no clear distinction in terms of explored state regions, as the learning has not begun yet. On the contrary, during the 200k-600k timesteps, there is a significant difference in terms of state space visitation. In resonance with the cumulative number of failures shown in Fig. 1, the results suggest that SAC engages in actions leading to more unsafe states. Conversely, SAAC *demonstrates to successfully constraint the agents to the safe regions*.

References

[1] Felix Berkenkamp, Riccardo Moriconi, Angela P. Schoellig, and Andreas Krause. Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes. In *IEEE CDC*, pages 4661–4666, 2016.

[2] Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. An empirical investigation of the challenges of real-world reinforcement learning. 2020.

[3] Hannes Eriksson, Debabrota Basu, Mina Alibeigi, and Christos Dimitrakakis. Sentinel: Taming uncertainty with ensemble-based distributional reinforcement learning. *arXiv preprint arXiv:2102.11075*, 2021.

[4] Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. *arXiv preprint arXiv:2103.06257*, 2021.

[5] Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, 24:81–108, 2005.

[6] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[7] Ronald A Howard and James E Matheson. Risk-sensitive markov decision processes. *Management science*, 18(7):356–369, 1972.

[8] Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *ICML*, pages 5556–5566. PMLR, 2020.

[9] LA Prashanth and Mohammad Ghavamzadeh. Variance-constrained actor-critic algorithms for discounted and average reward mdps. *Machine Learning*, 105(3):367–417, 2016.

[10] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 2019.

[11] S Sorin. Asymptotic properties of a non-zero sum stochastic game. *International Journal of Game Theory*, 15(2):101–107, 1986.

[12] Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pages 1049–1056, 2009.