WCSAC: Worst-Case Soft Actor Critic for Safety-Constrained Reinforcement Learning

Qisong Yang, Thiago D. Simão, Simon H. Tindemans, Matthijs T. J. Spaan
Delft University of Technology, The Netherlands
{q.yang, t.diassimao, s.h.tindemans, m.t.j.spaan}@tudelft.nl

Abstract

Safe exploration is regarded as a key priority area for reinforcement learning research. With separate reward signals and safety signals, it is natural to cast it as constrained reinforcement learning, where expected long-term costs of policies are constrained. However, it can be hazardous to set constraints on the expected safety signal without considering the tail of the distribution. For instance, in safety-critical domains, worst-case analysis is required to avoid disastrous results. We present a novel reinforcement learning algorithm called Worst-Case Soft Actor Critic, which extends the Soft Actor Critic algorithm with a safety critic to achieve risk control. More specifically, a certain level of conditional Value-at-Risk from the distribution is regarded as a safety measure to judge the constraint satisfaction, which guides the change of adaptive safety weights to achieve a trade-off between reward and safety. As a result, we can optimize policies on the premise that its worst-case performance satisfies the constraints. The empirical analysis shows that our algorithm attains better risk control compared to the expectation-based methods.

Introduction

In traditional reinforcement learning (RL) problems (Sutton and Barto 2018), agents can explore environments to learn optimal policies without safety concerns. However, unsafe interactions with environments are unacceptable in many safety-critical problems, for instance in robot navigation tasks. Even though RL agents can be trained in simulators, there are many real-world problems without simulators of sufficient fidelity. Constructing safe exploration algorithms for dangerous environments is challenging because we have to optimize policies under the premise of safety (Pecka and Svoboda 2014). In general, safety is still an open problem that hinders the wider application of RL (García and Fernández 2015).

Ray, Achiam, and Amodei (2019) propose to make constrained RL the main formalism of safe exploration, where the reward function and cost function (related to safety) are distinct. This framework tries to mitigate the problem of designing a single reward function that needs to carefully select a trade-off between safety and performance, which is problematic in most instances. In addition, it is generally desirable to optimize the sample efficiency, i.e., to minimize the number of samples required to learn safe optimal policies. Off-policy methods can reuse past experience to be more sample efficient, and which safe exploration can benefit from (Mnih et al. 2015). Thus off-policy methods are preferred over on-policy methods, which need new experiences to evaluate a policy (Schulman et al. 2015, 2017; Achiam et al. 2017).

In this paper, we focus on designing an off-policy algorithm for safety-constrained RL. Soft actor-critic (SAC; Haarnoja et al. 2018a,b) is an off-policy method built on the actor-critic framework, which encourages agents to explore by including a policy's entropy as a part of the reward. SAC shows better sample efficiency and asymptotic performance compared to prior on-policy and off-policy methods. SAC-Lagrangian (Ha et al. 2020) combines SAC with Lagrangian methods to address safety-constrained RL with local constraints, i.e., constraints are set for each timestep instead of each episode. The empirical analysis of SAC-Lagrangian shows that the optimal policy with constraint-satisfying expected long-term costs can be learned with a low number of constraint violations. However, the cost of individual episodes might exceed the expected-cost bound with a high probability. For safety-critical problems, it can be hazardous to use the expected long-term costs as safety evaluation. Instead, better alternatives for safety-constrained RL are algorithms that compute policies based on varying risk requirements, specialized to risk-neutral or risk-averse behavior (Duan et al. 2020; Ma et al. 2020).

We propose the Worst-Case Soft Actor Critic (WCSAC) algorithm that uses a separate safety-critic to estimate the distribution of accumulated cost to achieve risk control. We focus on the upper tail of the cost distribution, represented by the conditional Value-at-Risk (CVaR; Tyrrell Rockafellar and Uryasev 2000). In this way, policies can be optimized given different levels of CVaR, which determine the degree of risk aversion from a safety perspective. In addition, we endow safety and entropy with weights that are automatically adapted according to the performance of current policies. Experimental analysis shows that by setting the level of risk control, WCSAC algorithm attains stronger adaptability (compared to expectation-based baselines) when facing RL problems with higher safety requirements.

Background

In this work, we solve safety-constrained reinforcement learning problems. The goal of the agent is to maximize the (discounted) sum of rewards, while adhering to the cost limits provided.

Constrained Markov Decision Processes

We assume an environment modeled by a Constrained Markov Decision Process (CMDP; Altman 1999; Borkar 2005). Similar to a Markov Decision Process (MDP; Puterman 1994), the CMDP is formally defined as a tuple $(S, A, p, r, c, d, \gamma)$: a (multi-dimensional, continuous and bounded) state space S, a (multi-dimensional, continuous and bounded) action space A, a probabilistic transition function $p : S \times A \times S \to \mathbb{R}$ that represents the transition probabilities from a state by taking an action to the next state, a reward function $r : S \times A \to [r_{min}, r_{max}]$ that indicates the instant reward after taking action $a \in A$ in state $s \in S$, a cost function $c : S \times A \to [c_{min}, r_{max}]$ for evaluating constraint satisfaction, a given safety threshold d, and a discount factor $\gamma \in (0.0, 1.0)$.

The agent interacts with the environment in discrete timesteps. At each timestep t, it receives the current state $s \in S$ of the environment. Next, it performs an action $a \in A$, receives a reward $r(s, a)$ and a cost $c(s, a)$. Then it proceeds to the next timestep. Upon reaching a terminal state, the agent starts a new episode beginning in an arbitrary state $s_0 \sim p(s_0)$ in a new instance of the environment. The goal of the agent is to learn a policy that maximizes the expected return for each episode such that the safety constraint violation costs remain below the given threshold:

$$\begin{aligned} &\max_{\pi} \underset{(s_t,a_t)\sim\rho_\pi}{\mathbb{E}} \left[\sum_t \gamma^t r(s_t, a_t)\right] \\ &\text{s.t.} \underset{(s_t,a_t)\sim\rho_\pi}{\mathbb{E}} \left[\sum_t \gamma^t c(s_t, a_t)\right] \le d, \end{aligned} \tag{1}$$

where ρ_π is the trajectory distribution induced by π. Thus the expectation of long-term costs generated by the feasible policies is less than or equal to d.

For example, $c(s_t, a_t)$ can be the aggregate indicator cost function for the environment, so $c(s_t, a_t) = 1$ if the agent has taken an unsafe action in (s_t, a_t), otherwise $c(s_t, a_t) = 0$. In this case, we can set $d \ll 1$ to constrain the agent to stay away from the circumstances that lead to harm, or we can make $d > 1$ to allow the agent to have several unsafe interactions with the environment. In safety-critical domains, it is risky to build safety constraints in the expected long-term costs. In order to avoid disastrous events, the constraint in Equation 1 will be reformulated.

Maximum Entropy Reinforcement Learning

When the agent knows nothing about the environment, the safety constraint cannot be strictly fulfilled during exploration. During the early steps of learning, we still hope to encourage exploration to learn more about the environment. But the policy's entropy must be carefully balanced with the safety constraint, and the policy must be allowed to converge to a relatively deterministic policy, which reduces risks in terms of (safety-related) cost. SAC-based methods with entropy constraints and adaptive entropy weights are candidates to meet these conditions.

We focus on SAC-based methods to solve the safety-constrained RL problems. Thus it is essential to further formulate the problem as maximum entropy RL (Haarnoja et al. 2017) with safety constraints:

$$\begin{aligned} &\max_{\pi} \underset{(s_t,a_t)\sim\rho_\pi}{\mathbb{E}} \left[\sum_t \gamma^t r(s_t, a_t)\right] \\ &\text{s.t.} \begin{cases} \underset{(s_t,a_t)\sim\rho_\pi}{\mathbb{E}} [\sum_t \gamma^t c(s_t, a_t)] \le d \\ \underset{(s_t,a_t)\sim\rho_\pi}{\mathbb{E}} [-\log(\pi_t(a_t|s_t))] \ge \mathcal{H}_0 \quad \forall t \end{cases}, \end{aligned} \tag{2}$$

where entropy weight β manages the stochasticity of π and also determines the relative importance of the entropy term compared to reward and cost. $\mathcal{H}_0$ is the given entropy threshold to ensure $\mathcal{H}(\pi_t) \ge \mathcal{H}_0$ at each timestep t, which enforces a minimum degree of exploration. After maximum entropy RL is augmented with safety constraints, the regular SAC without safety term is not applicable anymore.

SAC-Lagrangian

Next, we describe the SAC-Lagrangian method for the maximum entropy RL with safety constraints (Ha et al. 2020). In general, SAC-Lagrangian is a SAC-based method that has two critics and uses adaptive safety weights to manage a trade-off between reward and safety. For clarity of exposition, we use the reward-critic to indicate the estimation of long-term rewards (possibly with entropy) to promote reward during learning. And the safety-critic expresses the estimation of long-term costs to encourage safety.

Constrained optimization problems can usually be solved by the Lagrange-multiplier method (Bertsekas 1982). Lagrangian methods introduce a Lagrange-multiplier κ to the constrained optimization:

$$\max_{\pi} \min_{\kappa \ge 0} \mathcal{L}(\pi, \kappa) \doteq f(\pi) - \kappa g(\pi), \tag{3}$$

$$\text{where } f(\pi) = \underset{(s_t,a_t)\sim\rho_\pi}{\mathbb{E}} \left[\sum_t \gamma^t r(s_t, a_t)\right]$$

$$\text{and } g(\pi) = \underset{(s_t,a_t)\sim\rho_\pi}{\mathbb{E}} \left[\sum_t \gamma^t c(s_t, a_t)\right] - d$$

in the case of Equation 1. So the adaptive penalty coefficients κ can be employed to enforce the constraint by gradient ascent on π and descent on κ.

Ha et al. (2020) developed SAC-Lagrangian for local constraints, which means that the safety cost is constrained at each timestep. However, it can be easily generalized to the constraint optimization with global constraints[1].

In SAC-Lagrangian for global constraints, two separate critics (Q functions) are trained, where one is a reward-critic Q^r for the reward and the entropy, and the other one is

[1]A similar approach has been used in the code available at https://github.com/openai/safety-starter-agents.

safety-critic Q^c for the safety term. In this paper, J is used to represent loss functions. By using a formulation similar to Haarnoja et al. (2018b), we can get the actor loss:

$$J_\pi(\theta) = \mathop{\mathbb{E}}_{s_t\sim\mathcal{D},a_t\sim\pi_\theta}[\beta\log\pi_\theta(a_t|s_t) - Q^r_{\pi_\theta}(s_t,a_t) + \kappa Q^c_{\pi_\theta}(s_t,a_t)], \quad (4)$$

where $\mathcal{D}$ is the replay buffer and θ indicates the parameters of the policy π. We learn the adaptive safety weight κ (Lagrangian multiplier) by minimizing the entropy loss $J_s(\kappa)$:

$$J_s(\kappa) = \mathop{\mathbb{E}}_{s_t\sim\mathcal{D},a_t\sim\pi_\theta}\left[\kappa(d - Q^c_{\pi_\theta}(s_t,a_t))\right], \quad (5)$$

so the entropy weight κ will be decreased if $d \geq Q^c_{\pi_\theta}$, otherwise κ will be increased to emphasize safety more.

Worst-Case Soft Actor Critic

SAC-Lagrangian maximizes the long-term return under the premise that the average performance in safety satisfies the constraint. In this way, RL agents are not aware of the potential risks because of the randomness in long-term costs, which is generated by the stochastic policy and the dynamics of the environment. In expectation-based cases, if a safe policy has higher returns and higher variance in safety costs, it will be preferred over another safe policy with lower returns and lower variance in safety costs. In safety-critical domains, the optimal policies are expected to be more robust, i.e., to have a lower risk of hazardous events even for the stochastic or heavy-tailed long-term costs.

Vulnerable reliance on expected costs

The unreliability of SAC-Lagrangian stems mainly from the use of Q^c_π as the safety-critic (Mihatsch and Neuneier 2002; Dabney et al. 2018). $Q^c_\pi(s,a)$ is the expectation of long-term cumulative costs from starting point (s,a), denoted by

$$C_\pi(s,a) = \sum_t \gamma^t c(s_t,a_t)|s_0 = s, a_0 = a, \pi. \quad (6)$$

Following policy π, the probability distribution of $C_\pi(s,a)$ is modeled as $p^\pi(C|s,a)$, such that:

$$Q^c_\pi(s,a) = \mathbb{E}_{p^\pi}\left[C \mid s,a\right].$$

Thus the realised cost is quite likely to exceed (possibly by a large margin) the expected cost constraint, even if our decisions satisfy $Q^c_\pi(s_t,a_t) < d$ at each timestep.

An example of such a case is a simple CMDP shown in Figure 1. In each state, we can choose to move or not. If we choose to move (take action m), we will get reward 1 and cost 1, otherwise both reward and cost will be 0. With an episode length of two timesteps and $d = 1.95$ for each episode, the approximate optimal policy get from Lagrangian method will be around $\pi(m|a) = 0.9, \pi(n|a) = 0.1, \pi(m|b) = 0.9, \pi(n|b) = 0.1$, which means the corresponding $Q^c(a,m) = 1.9, Q^c(a,n) = 0.9, Q^c(b,m) = 1.9, Q^c(b,n) = 0.9$ (at time $t = 0$) are within the safety threshold. But the real costs generated by the policy will be larger than the threshold with probability $p = 0.81$. Thus the optimal policy is hardly thought to be acceptable for safety-critical problems, even though it satisfies the constraint.

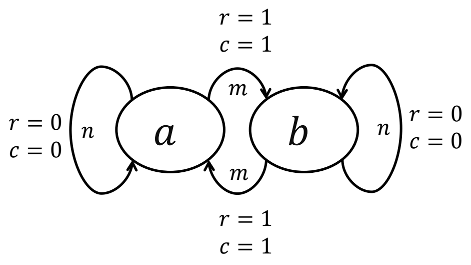


Figure 1: A CMDP example with $d = 1.95$ and $T = 2$. It has state space $\{a,b\}$ and action space $\{m,n\}$.

Distributional safety-critic

To obtain risk-averse policies, we learn the distribution of long-term costs, which helps the agent to be aware of the safety-risk at each timestep. We propose the Worst-Case Soft Actor Critic (WCSAC) algorithm, which replaces the safety-critic of SAC-Lagrangian with a distributional one to adjust policies based on the given safety-risk requirement.

To get a safety measure, we must learn the distributional safety-critic first. Following policy π, the distribution of long-term costs is defined as $C_\pi(s,a) \doteq p^\pi(C|s,a)$, so the distributional Bellman operator (Sobel 1982; Morimura et al. 2010; Tamar, Di Castro, and Mannor 2016) $\mathcal{T}^\pi$ is defined as:

$$\mathcal{T}^\pi C(s,a) \doteq c(s,a) + \gamma C(s',a'),$$

where $s' \sim p(\cdot|s,a), a' \sim \pi(\cdot|s')$. We approximate the distribution $C_\pi(s,a)$ with a Gaussian distribution (Tang, Zhang, and Salakhutdinov 2020), i.e.,

$$C_\pi(s,a) \sim \mathcal{N}(Q^c_\pi(s,a), V^c_\pi(s,a)),$$

where the variance $V^c_\pi(s,a) = \mathbb{E}_{p^\pi}[C^2|s,a] - (Q^c_\pi(s,a))^2$. To estimate Q^c_π, we can use the standard Bellman function:

$$Q^c_\pi(s,a) = c(s,a) + \gamma\sum_{s'\in S}p(s'|s,a)\sum_{a'\in A}\pi(a'|s')Q^c_\pi(s',a'). \quad (7)$$

The projection equation for estimating $V^c_\pi(s,a)$ is:

$$\begin{aligned} V^c_\pi(s,a) &= c(s,a)^2 - Q^c_\pi(s,a)^2 \\ &+ 2\gamma c(s,a)\sum_{s'\in S}p(s'|s,a)\sum_{a'\in A}\pi(a'|s')Q^c_\pi(s',a') \\ &+ \gamma^2\sum_{s'\in S}p(s'|s,a)\sum_{a'\in A}\pi(a'|s')V^c_\pi(s',a') \\ &+ \gamma^2\sum_{s'\in S}p(s'|s,a)\sum_{a'\in A}\pi(a'|s')Q^c_\pi(s',a')^2. \end{aligned} \quad (8)$$

We refer the reader to Tang, Zhang, and Salakhutdinov (2020) for the proof of Equation 8.

Safety-critic learning

In this paper, we use two neural networks parameterized by μ and η respectively to estimate the safety-critic, i.e.,

$$Q^c_\mu(s,a) \rightarrow \hat{Q}^c_\pi(s,a) \text{ and } V^c_\eta(s,a) \rightarrow \hat{V}^c_\pi(s,a).$$

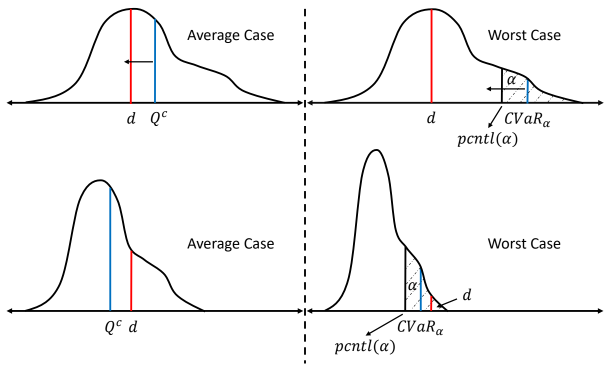


Figure 2: In the average case (left panel), the policies are optimized to ensure Q^c (blue line) is moved to the left side of the fixed boundary d (red line). In the worst-case (right panel), we optimize policies to reshape the distribution of long-term costs to ensure the $CVaR_\alpha$ (blue line) measure on the left side of the fixed boundary d (red line).

In order to learn the safety-critic, the distance between value distributions is measured by the p-Wasserstein distance (Bellemare, Dabney, and Munos 2017; Olkin and Pukelsheim 1982):

$$W_p(u,v) \doteq \left(\int_0^1 |F_u^{-1}(s) - F_v^{-1}(s)|^p ds\right)^{1/p},$$

where $u \sim \mathcal{N}(Q_1, V_1)$, $v \sim \mathcal{N}(Q_2, V_2)$ and F_u, F_v are the cumulative distribution functions (CDF). In this paper, we use the simplified 2-Wasserstein distance (Tang, Zhang, and Salakhutdinov 2020) to estimate the safety-critic loss:

$$\begin{aligned} W_2(u,v) &= \|Q_1 - Q_2\|_2^2 \\ &\quad + trace(V_1 + V_2 - 2(V_2^{1/2} V_1 V_2^{1/2})^{1/2}). \end{aligned}$$

The 2-Wasserstein distance can be computed as the Temporal Difference (TD) error based on the projection equations (7) and (8) to update the safety-critic, i.e., we will minimize the following values:

$$J_C(\mu) = \mathop{\mathbb{E}}_{(s_t,a_t)\sim\mathcal{D}} \|\Delta Q(s_t,a_t,\mu)\|_2^2, \text{ and}$$
$$J_V(\eta) = \mathop{\mathbb{E}}_{(s_t,a_t)\sim\mathcal{D}} trace(\Delta V(s_t,a_t,\eta)),$$

where $J_C(\mu)$ is the loss function of Q_μ^c, and $J_V(\eta)$ is the loss function of V_η^c. We can get

$$\Delta Q(s_t,a_t,\mu) = \overline{Q}_\mu^c(s_t,a_t) - Q_\mu^c(s_t,a_t),$$

where $\overline{Q}_\mu^c(s_t,a_t)$ is the TD target from Equation 7, and

$$\begin{aligned} \Delta V(s_t,a_t,\mu) &= \overline{V}_\mu^c(s_t,a_t) + V_\mu^c(s_t,a_t) \\ &\quad - 2(V_\mu^c(s_t,a_t)^{1/2} \overline{V}_\mu^c(s_t,a_t) V_\mu^c(s_t,a_t)^{1/2})^{1/2}, \end{aligned}$$

where $\overline{V}_\mu^c(s_t,a_t)$ is the TD target from Equation 8.

Safety based on CVaR

In Figure 2, the x-axis depicts the long-term cost C (Equation 6). The y-axis depicts the density of its probability distribution. The SAC-Lagrangian algorithm focuses on the average performance in safety when optimizing policies. Thus, π, Q_π^c, and the shape of $p^\pi(C|s,a)$ will be changed during the training process until Q_π^c (blue line) is shifted to the left side of the boundary (red line). After that, there is still a strong likelihood that the constraint value d is exceeded. For a policy π, Q_π^c can only be used as the evaluation of average performance in safety, however, in safety-critical domains, the worst-case performance in safety is preferred over the average performance. Therefore, we replace the expected value with the Conditional Value-at-Risk (CVaR; Tyrrell Rockafellar and Uryasev 2000), a *coherent* risk measure, to judge the safety of a policy. In the right panel of Figure 2, we set the constraint on CVaR. Thus we optimize policies that will move the tail-end of $p^\pi(C|s,a)$ (blue line) to the left side of the boundary d (red line).

Definition 1 (Risk level in safety). *A positive scalar $\alpha \in (0,1]$ is used to define the risk level in safety for WCSAC. A WCSAC with smaller α ($\alpha \to 0$) is expected to be more pessimistic and risk-averse in safety. Conversely, a larger value of α leads to less risk-averse behavior, with $\alpha = 1$ corresponding to the risk-neutral case.*

We focus on the α-percentile $F_C^{-1}(1-\alpha)$, where F_C is the CDF of $p^\pi(C|s,a)$, so we can get the CVaR:

$$CVaR_\alpha \doteq \mathbb{E}_{p^\pi}[C|C \geq F_C^{-1}(1-\alpha)].$$

The following definition gives us a new constraint to learn risk-averse policies, which differs from the traditional constraint (1).

Definition 2 (Safety based on CVaR). *Given the risk level α, a policy π is safe if it satisfies*

$$\mathbb{E}_{p^\pi}\left[C(s_t,a_t)|C(s_t,a_t) \geq F_C^{-1}(1-\alpha)\right] \leq d \quad \forall t,$$

where $(s_t,a_t) \sim \rho_\pi$ and $s_0 \sim p(s_0)$.

It is excessively expensive to compute the CVaR measure directly by sampling for long horizons (Tamar, Glassner, and Mannor 2015). In this paper, the CVaR measure is easier to be computed from the distributional safety-critic at each timestep since the Gaussian distribution results in a closed-form estimation for CVaR (Khokhlov 2016; Tang, Zhang, and Salakhutdinov 2020).

Worst-case actor

Built on the distributional safety-critic, the expected long-term costs can be replaced with a new safety metric to guide safe exploration, i.e., $CVaR_\alpha$. At each iteration, $\hat{Q}_\pi^c(s,a)$ and $\hat{V}_\pi^c(s,a)$ can be estimated. Thus the new safety measure for risk level α is computed by

$$\begin{aligned} \Gamma_\pi(s,a,\alpha) &\doteq CVaR_\alpha \\ &= Q_\pi^c(s,a) + \alpha^{-1}\phi(\Phi^{-1}(\alpha))\sqrt{V_\pi^c(s,a)}, \end{aligned} \tag{9}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ denote the PDF and the CDF of the standard normal distribution (Khokhlov 2016). For a certain risk level α, we optimize policy until Γ_π satisfies

$$\Gamma_\pi(s_t,a_t,\alpha) \leq d \quad \forall t \tag{10}$$

Algorithm 1 Worst-Case Soft Actor Critic

```
1: **Inputs**: Initial parameters $\alpha$, $\psi$, $\mu$, $\eta$, $\theta$, $\tau$
2: **Initialize**: Target networks: $\langle\overline{\psi}, \overline{\mu}, \overline{\eta}\rangle \leftarrow \langle\psi, \mu, \eta\rangle$
3: **Initialize**: Replay buffer $\mathcal{D} \leftarrow \emptyset$
4: for each iteration do
5:     for each environment step do
6:         $a_t \sim \pi_\theta(a_t|s_t)$
7:         $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
8:         $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), c(s_t, a_t), s_{t+1})\}$
9:     end for
10:    for each gradient step do
11:        Sample experience from replay buffer $\mathcal{D}$
12:        Compute $\Gamma_{\pi_\theta}$ based on Equation 9
13:        $\psi_i \leftarrow \psi_i - \lambda_R \hat{\nabla}_{\psi_i} J_R(\psi_i)$ for $i \in \{1, 2\}$
14:        $\mu \leftarrow \mu - \lambda_C \hat{\nabla}_\mu J_C(\mu)$
15:        $\eta \leftarrow \eta - \lambda_V \hat{\nabla}_\eta J_V(\eta)$
16:        $\theta \leftarrow \theta - \lambda_\pi \hat{\nabla}_\theta J_\pi(\theta)$
17:        $\beta \leftarrow \beta - \lambda_\beta \hat{\nabla}_\beta J_e(\beta)$
18:        $\kappa \leftarrow \kappa - \lambda_\kappa \hat{\nabla}_\kappa J_s(\kappa)$
19:        # Update target networks weights
20:        $\overline{\psi}_i \leftarrow \tau\psi_i + (1-\tau)\overline{\psi}_i$ for $i \in \{1, 2\}$
21:        $\overline{\mu} \leftarrow \tau\mu + (1-\tau)\overline{\mu}$
22:        $\overline{\eta} \leftarrow \tau\eta + (1-\tau)\overline{\eta}$
23:    end for
24: end for
25: **Output**: Optimized parameters $\psi$, $\mu$, $\eta$, $\theta$, $\beta$, $\kappa$
```

according to Definition 2. Following a formulation similar to Haarnoja et al. (2018b), we should try to minimize the following KL divergence (Kullback and Leibler 1951) to update the policy:

$$\min_\pi D_{KL}\left(\pi(\cdot|s_t) \middle\| \frac{\exp\left(\frac{1}{\beta}(Q^r_\pi(s_t,\cdot) - \kappa\Gamma_\pi(s_t,\cdot,\alpha))\right)}{Z^\pi(s_t)}\right),$$

where $Z^\pi(s_t)$ is the partition function to normalize the distribution. The KL divergence can be rewritten as

$$D_{KL}(\pi_\theta(\cdot|s_t)|| \exp\left(\frac{1}{\beta}X^{\pi_\theta}_{\alpha,\kappa}(s_t,\cdot) - \log(Z^{\pi_\theta}(s_t))\right))$$

$$= \mathop{\mathbb{E}}_{\substack{s_t\sim\mathcal{D}\\a_t\sim\pi_\theta}}\left[-\log\left(\frac{\pi_\theta(a_t|s_t)}{\exp\left(\frac{1}{\beta}X^{\pi_\theta}_{\alpha,\kappa}(s_t,a_t) - \log(Z^{\pi_\theta}(s_t))\right)}\right)\right]$$

$$= \mathop{\mathbb{E}}_{\substack{s_t\sim\mathcal{D}\\a_t\sim\pi_\theta}}\left[\log\pi_\theta(a_t|s_t) - \frac{1}{\beta}X^{\pi_\theta}_{\alpha,\kappa}(s_t,a_t) + \log(Z^{\pi_\theta}(s_t))\right],$$

where $X^\pi_{\alpha,\kappa}(s,a) = Q^r_\pi(s,a) - \kappa\Gamma_\pi(s,a,\alpha)$. β and κ are the adaptive entropy and safety weights, respectively. $Z^{\pi_\theta}(s_t)$ has no influence on updating θ, thus it can be omitted. The new actor loss is

$$J_\pi(\theta) = \mathop{\mathbb{E}}_{\substack{s_t\sim\mathcal{D}\\a_t\sim\pi_\theta}}\left[\beta\log\pi_\theta(a_t|s_t) - X^{\pi_\theta}_{\alpha,\kappa}(s_t,a_t)\right].$$

Based on the new safety measure $\Gamma_\pi(s,a,\alpha)$, the safety weight κ can be learned by minimizing the loss $J_s(\kappa)$:

$$J_s(\kappa) = \mathop{\mathbb{E}}_{\substack{s_t\sim\mathcal{D}\\a_t\sim\pi_\theta}}\left[\kappa(d - \Gamma_{\pi_\theta}(s_t,a_t,\alpha))\right].$$

We update the reward-critic $Q^r_{\pi_\theta}$ (parameterized by ψ) and entropy weight β in the same way as the SAC method. For the loss function $J_e(\beta)$ to learn the adaptive entropy weights, and the loss function $J_R(\psi)$ of the reward-critic, we refer the reader to Haarnoja et al. (2018b).

Complete algorithm

Algorithm 1 presents our method. At each environment step, the agent executes a new action sampled from the current policy and then proceeds to the next state. The experience will be stored in the replay buffer (lines 5-9). For the gradient descent steps, the method uses batches sampled from the replay buffer to update all function parameters (lines 10-23).

In standard maximum entropy RL, the entropy of the policy is expected to be as large as possible. However, relatively deterministic policies are preferred over stochastic policies in safe exploration, even though it is essential to encourage exploration during the early steps of learning. In SAC, the entropy of the policy is constrained to ensure that the final optimal policy is more robust (Haarnoja et al. 2018b). Therefore, for safety-critical domains, it is preferred to set a relatively low minimum requirement $\mathcal{H}_0$ for the entropy, or omit this constraint altogether.

For the reward-critic, to avoid overestimation and reduce the positive bias during the policy improvement process, we also learn two soft Q-functions independently, which are parameterized by ψ_1 and ψ_2. The minimum Q-function is used in each gradient step. For the safety-critic, we use two separate neural networks to estimate the mean function and variance function respectively. The size of each network can be smaller than using one network to estimate the mean function and variance together, so it does not add more parameters to be trained. Besides, it is much easier to compare the distributional safety-critic of WCSAC to the regular safety-critic of SAC-Lagrangian, which can be seen as ablation of WCSAC. We use 4 target networks to achieve stable updating, a common technique used in DQN (Mnih et al. 2015) and DDPG (Lillicrap et al. 2015). Specifically, the parameters of target networks (including safety-critic and reward-critic) are updated by moving averages (lines 20-22), where hyperparameter $\tau \in [0, 1]$ is used to reduce fluctuations.

Empirical Analysis

We evaluate our method based on the Safety Gym benchmark (Ray, Achiam, and Amodei 2019). In these environments (see Figure 3) a point agent (one actuator for turning

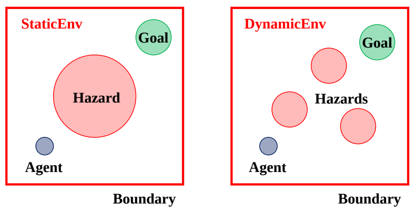


Figure 3: Point navigation domains StaticEnv and DynamicEnv. The environments differ in the number and size of hazards, and generation of goal and hazards' locations.

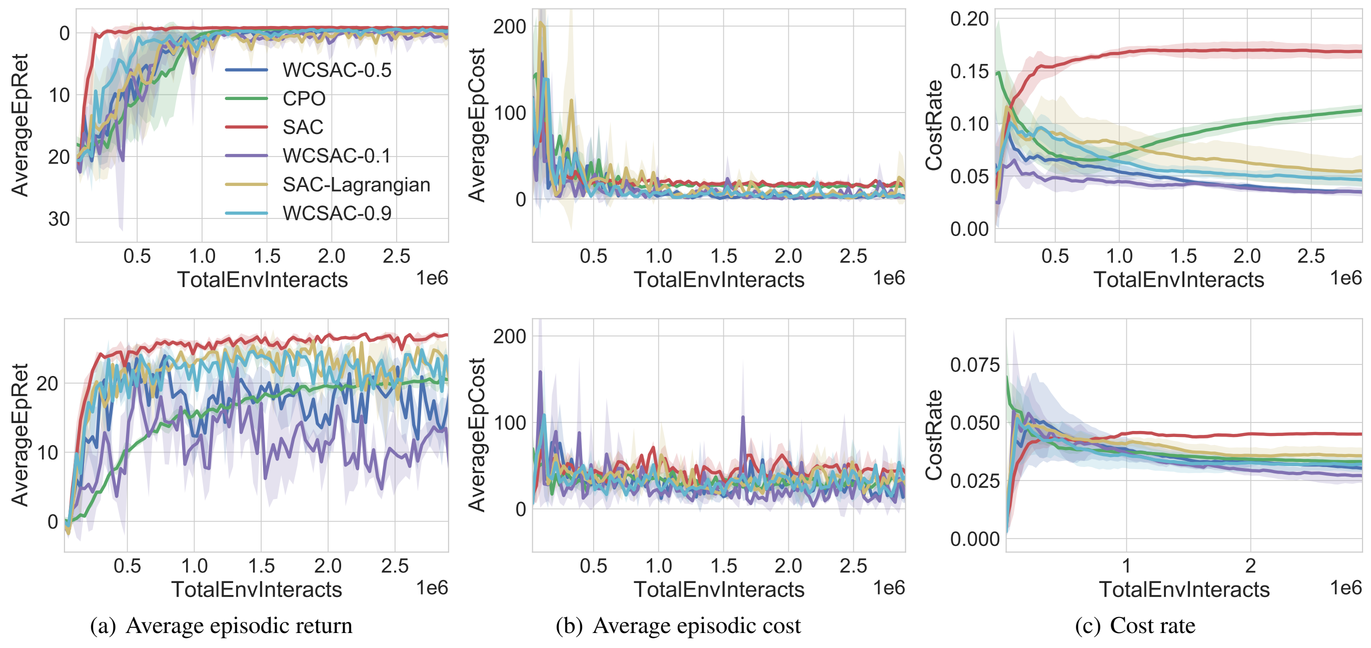


Figure 4: Comparison of SAC, CPO, SAC-Lag, and WCSAC during training in StaticEnv (top row) and DynamicEnv (bottom row). The lines are the average of all runs, and the shaded area is the standard deviation.

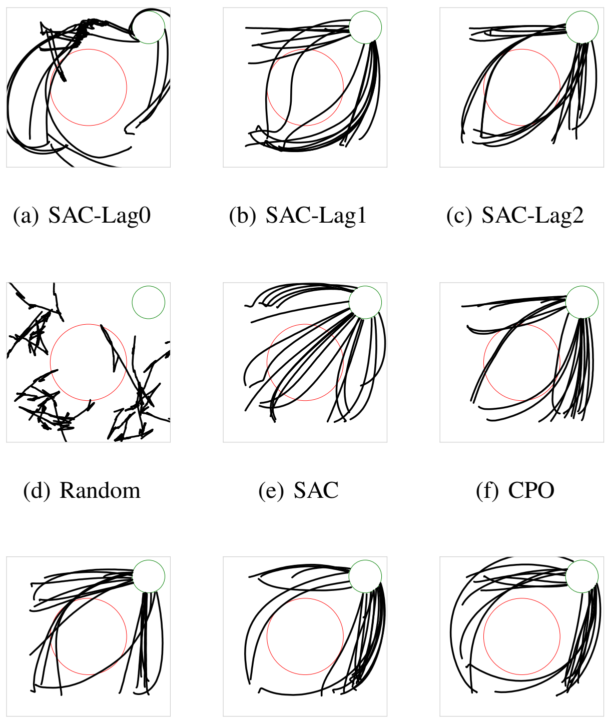
and another for moving forward/backward) navigates in a 2D map to reach the goal position while trying to avoid hazardous areas. In StaticEnv (Figure 3 left), the agent gets a reward $r - 0.2$ in each step, where r is the original reward signal of Safety Gym (distance towards goal plus a constant for being within range of goal), and the offset -0.2 incentivizes the agent to reach the target in the smallest number of time steps. DynamicEnv keeps the original reward signal. In both environments, the initial state of the agent is randomly initialized in each episode. The episodic locations of goal and hazards are also arbitrarily generated in DynamicEnv but fixed in StaticEnv. In each step, if the agent stays in the hazardous area, it incurs a cost $c = 1$, otherwise $c = 0$. We use a discount factor $\gamma = 0.99$.

We set $d = 15$ for the expected cost limit. All the agents are trained for 100 epochs, where the length of each epoch is 30000 interaction steps and the maximal length of each episode is 1000 interaction steps. Furthermore, all experiments were run with three random seeds. Specifically, we will compare SAC, CPO, SAC-Lagrangian (SAC-Lag) and WCSAC with different risk levels in safety, i.e., WCSAC-0.1 ($\alpha = 0.1$: highly risk-averse), WCSAC-0.5 ($\alpha = 0.5$) and WCSAC-0.9 ($\alpha = 0.9$: almost risk-neutral). Four CPUs were used in parallel for training in all cases. The detailed environment setting and all hyperparameters of the algorithms are presented in the Appendix.

Results

During training, we use the following metrics: average episodic return, average episodic cost, and cost rate (Ray, Achiam, and Amodei 2019). The cost rate at each epoch is computed by dividing the cumulative safety costs by the current environment interaction steps. We present the results in Figure 4. We observe that all algorithms find policies that can reach the goal area at the end of the training (Figure 4(a)), but with different convergence rates. Compared to safe methods (SAC-Lag, CPO, and WCSAC), SAC has better and stable performance in average episodic return obviously, however it does not satisfy the constraint. Regarding safety, figures 4(b) and 4(c) show that all safe methods converge to constraint-satisfying policies, and WCSAC-0.9 performance is close to CPO and SAC-Lag. Besides, WCSAC with lower risk level generates fewer constraint violations during training.

We execute a trajectory analysis in StaticEnv, see Figure 5. We only show the training process of SAC-Lag at different stages because other methods show similar behavior during training. At the beginning of learning (Figure 5(a)), it is possible that the agent cannot get out of the hazard once stepping into it, and the trajectory ends early before arriving at the goal area. In Figure 5(d), the trajectories of random policy are highly chaotic. The final policies from SAC-Lag, CPO, and WCSAC-0.9 perform better than before, but still prefers to take a risk within the budget to get a larger return (Figures 5(c), 5(f) and 5(g)). Conversely, the agent from WCSAC-0.5 becomes more risk-averse in Figure 5(h). The behaviors of the SAC agent are presented in Figure 5(e), since the safety constraint is not considered, the agent chooses the shortest path to reach the target directly. Finally, in Figure 5(i), we can see that the agent from WCSAC-0.1 prefers to stay away from the hazardous area for the risk level setting.


(a) SAC-Lag0 (b) SAC-Lag1 (c) SAC-Lag2

(d) Random (e) SAC (f) CPO

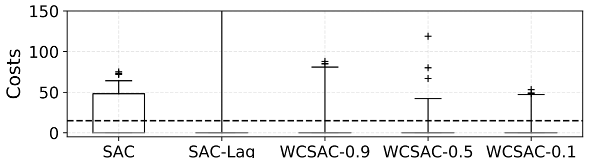
(g) WCSAC-0.9 (h) WCSAC-0.5 (i) WCSAC-0.1

Figure 5: Trajectory analysis. (a)-(c) show the trajectories generated by policies from SAC-Lag at the beginning, middle stage and end of training respectively. (d)-(i) show the final trajectories from random policy, SAC, CPO, and WCSAC separately.

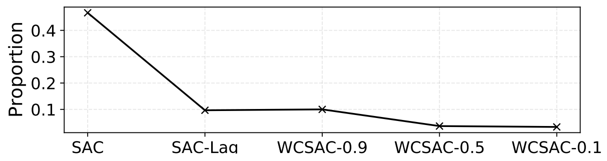
After training, we use 300 test runs (100 runs for each random seed) to evaluate the final policies of these algorithms. In Table 1, the results show that each WCSAC variation satisfies its corresponding CVaR bound (estimated by the average costs of the worst 300α trajectories), while only WCSAC-0.1 results in $CVaR_{0.1} < 15$. In Figure 6, we compare the SAC-based methods. The whiskers of the boxplot Figure 6(a) are set at the $[1, 99]$ percentiles of the data. To make the boundary d more clear, we set the y-axis view limits. So the data of SAC-Lag is out of the chart. The optimal policies learned by SAC have poor performance in safety without considering the constraint. The optimal policies from SAC-Lag can ensure that most of the trajectories are safe, but some extremely dangerous events happen, which is undesirable for safety-critical problems. As to the proportion of budget exceedance, WCSAC-0.9 has a similar average performance with SAC-Lag, but the boxplot shows that extreme cost events are much less likely to happen. Compared to SAC-Lag and WCSAC-0.9, WCSAC with lower risk levels have more stable performance in safety. Although the policies from WCSAC-0.1 and WCSAC-0.5 still generate some unsafe trajectories, the likelihood is much lower.

	EC	C0.9	C0.5	C0.1	ER
SAC	21.7	24.1	42.8	56.5	0.97
SAC-Lag	**14.3**	15.9	28.6	141.8	0.27
WCSAC-0.9	4.2	**4.6**	8.4	31.4	0.56
WCSAC-0.5	1.8	2.0	**3.6**	17.9	0.19
WCSAC-0.1	1.4	1.6	2.9	**14.3**	-0.43
CPO	**16.3**	18.1	32.5	58.3	0.84
Random Policy	49.1	54.5	98.1	431.6	-20.10

Table 1: Performance of the agents on the StaticEnv according to multiple metrics: expected cost (EC), cost-CVaR-0.9 (C0.9), cost-CVaR-0.5 (C0.5), cost-CVaR-0.1 (C0.1), and expected return (ER). In bold we show the value of the constraint used by the agent.



(a) Boxplot of safety evaluation after training



(b) Proportion of budget exceedance

Figure 6: Evaluation after training (StaticEnv). The boxplot (a) shows the statistical properties of long-term costs generated by the SAC-based algorithms, and the dashed line indicates the safety threshold d. In (b), we use the proportion of budget exceedance to analyse the safety of the SAC-based algorithms.

Conclusion

In this paper, we propose the algorithm WCSAC to solve safety-constrained RL problems. We augment SAC with a separate distributional safety-critic (parallel to the reward-critic) to make the algorithm more adaptive when facing RL problems with higher safety requirement. In the experiments, the agent has different performance in safety under different risk level α. When $\alpha \ll 1$, we can get more risk-averse policies for a safety-critical domain. Our research is meaningful for the development of safe exploration. In this paper, the distribution of long-term costs is approximated to be a Gaussian distribution. In the future, we can further explore modeling the uncertainty of the safety-critic in different ways. Moreover, the focus of this paper has been on safety, but a similar approach could be taken to consider the variability of long-term rewards. It is then also interesting to consider trade-offs between a distributional reward-critic and a distributional safety-critic.

Acknowledgments

This research is partly funded by the Netherlands Organisation for Scientific Research (NWO), as part of the Energy System Integration: planning, operations, and societal embedding program.

References

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained Policy Optimization. In *34th International Conference on Machine Learning*, 22–31. PMLR.

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

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A Distributional Perspective on Reinforcement Learning. In *34th International Conference on Machine Learning*, 449–458. PMLR.

Bertsekas, D. P. 1982. *Constrained optimization and Lagrange multiplier methods*, volume 1. Academic press.

Borkar, V. S. 2005. An actor-critic algorithm for constrained Markov decision processes. *Systems & control letters* 54(3): 207–213.

Dabney, W.; Rowland, M.; Bellemare, M. G.; and Munos, R. 2018. Distributional reinforcement learning with quantile regression. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2892–2901. AAAI Press.

Duan, J.; Guan, Y.; Li, S. E.; Ren, Y.; and Cheng, B. 2020. Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors. arXiv:2001.02811.

García, J.; and Fernández, F. 2015. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research* 16(1): 1437–1480.

Ha, S.; Xu, P.; Tan, Z.; Levine, S.; and Tan, J. 2020. Learning to Walk in the Real World with Minimal Human Effort. arXiv:2002.08550.

Haarnoja, T.; Tang, H.; Abbeel, P.; and Levine, S. 2017. Reinforcement Learning with Deep Energy-Based Policies. In *34th International Conference on Machine Learning*, 1352–1361. PMLR.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018a. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *35th International Conference on Machine Learning*, 1861–1870. PMLR.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018b. Soft Actor-Critic Algorithms and Applications. arXiv:1812.05905.

Khokhlov, V. 2016. Conditional value-at-risk for elliptical distributions. *Evropský časopis ekonomiky a managementu* 2(6): 70–79.

Kullback, S.; and Leibler, R. A. 1951. On information and sufficiency. *The annals of mathematical statistics* 22(1): 79–86.

Lillicrap, T.; Hunt, J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations*, 1–10. ICLR.

Ma, X.; Zhang, Q.; Xia, L.; Zhou, Z.; Yang, J.; and Zhao, Q. 2020. Distributional Soft Actor Critic for Risk Sensitive Learning. arXiv:2004.14547.

Mihatsch, O.; and Neuneier, R. 2002. Risk-Sensitive Reinforcement Learning. *Machine learning* 49(2-3): 267–290.

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

Morimura, T.; Sugiyama, M.; Kashima, H.; Hachiya, H.; and Tanaka, T. 2010. Parametric Return Density Estimation for Reinforcement Learning. In *Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, 368–375. AUAI Press.

Olkin, I.; and Pukelsheim, F. 1982. The distance between two random vectors with given dispersion matrices. *Linear Algebra and its Applications* 48: 257–263.

Pecka, M.; and Svoboda, T. 2014. Safe Exploration Techniques for Reinforcement Learning – An Overview. In *First International Workshop on Modelling and Simulation for Autonomous Systems*, 357–375. Springer.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, volume 1. Wiley.

Ray, A.; Achiam, J.; and Amodei, D. 2019. Benchmarking Safe Exploration in Deep Reinforcement Learning. https://cdn.openai.com/safexp-short.pdf.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust Region Policy Optimization. In *32nd International Conference on Machine Learning*, 1889–1897. JMLR.org.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Sobel, M. J. 1982. The Variance of Discounted Markov Decision Processes. *Journal of Applied Probability* 19(4): 794–802.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*, volume 2. MIT press.

Tamar, A.; Di Castro, D.; and Mannor, S. 2016. Learning the Variance of the Reward-To-Go. *The Journal of Machine Learning Research* 17(1): 361–396.

Tamar, A.; Glassner, Y.; and Mannor, S. 2015. Optimizing the CVaR via sampling. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2993–2999. AAAI Press.

Tang, Y. C.; Zhang, J.; and Salakhutdinov, R. 2020. Worst Cases Policy Gradients. In *3rd Annual Conference on Robot Learning*, 1078–1093. PMLR.

Tyrrell Rockafellar, R.; and Uryasev, S. 2000. Optimization of conditional value-at-risk. *Journal of risk* 2(3): 21–41.