Deep Learning Lab 2022 SS

Worst-Case SAC for Safety-Constrainted RL: A PyTorch Toolkit

Luca Pfrang, Akshay L Chandra, Sri Harsha Koribille
{pfrangl, lagandua, koribils}@cs.uni-freiburg.de

Supervior: Baohe Zhang

Abstract

In safety-critical domains where free exploration is expensive or infeasible, we turn to safety-constrained reinforcement learning (Safe-RL) methods that incentivize safe exploration. In this setting, the agent gets additional safety signal a.k.a cost. With separate reward and cost signals, Safe-RL methods are often modelled as a constrained optimization problem where policies are learned by estimating both expected long-term rewards as well as costs. In this project, we explore, implement and slightly extend a recent work [8] that learns optimal policies by estimating long-term cost distribution (i.e., mean and variance). We also extend WCSAC by smoothing the policy through adversarial regularization. Our code, results and this report are available at https://github.com/acl21/wcsac/.

1. Introduction

In traditional RL problems, agents are typically allowed to explore environments without having to worry about their safety. However, unsafe exploration in safety-critical problems can lead to either severe damage of the environment or the agent itself. That is why research interest in RL methods with safety as priority is growing ever more but it is still an open problem. We refer the readers to [1, 6–8] for a thorough review of recent efforts in Safe-RL. In this work, we focus our attention on an algorithm proposed by Yang et al. [8] that uses a separate safety critic to estimate the distribution of long-term accumulated cost to achieve risk control. Section 2 includes necessary background, Section 3 contains the main algorithm we re-implement in PyTorch and Section 4 explains our contributions, findings, and our attempt to extend [8]. Section 5 has all the experimental details, hyperparameters, architectural choices, etc. Section 6 has our experimental findings and discussion.

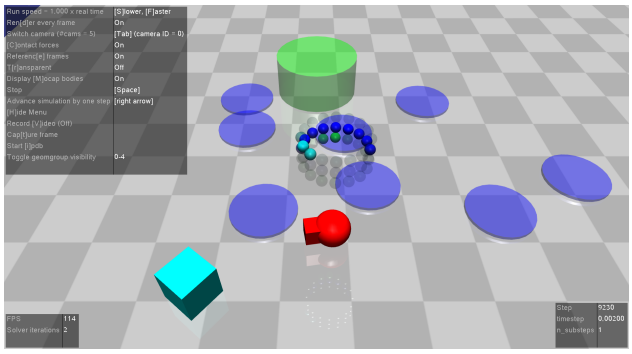


Figure 1. Screenshot of the `Safexp-PointGoal1-v0` environment in Safety-Gym [3]. Agent in red is expected to reach the green cylindrical goals, avoiding circular hazards in purple on the way. The cube in cyan is a vase, the agent is not penalized for hitting it.

2. Background

In Safe-RL, the goal of the agent is to maximize the (discounted) sum of rewards while adhering to the safety cost limits provided by the environment.

2.1. Constrained Markov Decision Processes

We assume that the environment is modelled by a Constrained Markov Decision Process (CMDP), formally defined as a tuple $(S, A, p, r, c, d, \gamma)$: state space S, action space A, a probabilistic transition function $p : S \times A \times S \rightarrow \mathbb{R}$ that assigns a probability to the transitions from a state by taking an action to the next state, a reward function $r : S \times A \rightarrow [r_{min}, r_{max}]$ provides the instant reward and a cost function $c : S \times A \rightarrow [c_{min}, c_{max}]$ provides the safety cost incurred, when action $a \in A$ is taken in state $s \in S$, a given safety threshold d and a discount factor $\gamma \in [0, 1)$.

At each discrete timestep t, the agent receives the current state $s \in S$, performas action $a \in A$ and receives reward $r(s, a)$ and a cost $c(s, a)$. Then it proceeds to the next timestep and starts a new episode upon reaching a terminal state. 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:

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

where ρ_π is the trajectory distribution induced by following policy π.

2.2. SAC-Lagrangian

SAC-Lagrangian [1] is a SAC-based method that has two critics (one to estimate long-term rewards and one to estimate long-term costs) and uses adaptive safety weights to manage a trade-off between reward and safety. The authors solve the constrained optimization problem in Equation 1 by the Lagrange-multiplier method, rewrite the constrained optimization with a Lagrange-multiplier κ as:

$$\begin{aligned} &\min_{\kappa\geq 0}\max_{\pi} \mathcal{L}(\pi,\kappa) \doteq f(\pi) - \kappa g(\pi), \\ &\text{where } f(\pi) = \mathop{\mathbb{E}}_{(s_t,a_t)\sim\rho_\pi}\left[\sum_t \gamma^t r(s_t,a_t)\right] \\ &\text{and } g(\pi) = \mathop{\mathbb{E}}_{(s_t,a_t)\sim\rho_\pi}\left[\sum_t \gamma^t c(s_t,a_t)\right] - d \end{aligned} \tag{2}$$

In SAC-Lagrangian, two separate critics (Q functions are trained), one is a reward critic Q^r for the reward and the entropy and the other one is safety critic Q^c for the safety cost. In this work, the actor loss is formulated as:

$$\begin{aligned} J_\pi(\theta) = \mathop{\mathbb{E}}_{s_t\sim D_t, a_t\sim\pi_\theta}[&\beta \log\pi_\theta(a_t|s_t) - Q^r_{\pi_\theta}(s_t,a_t) \\ &+\kappa Q^c_{\pi_\theta}(s_t,a_t)] \end{aligned} \tag{3}$$

where β, entropy weight (Lagrange multiplier) controls *softness* or stochasticity of π and also signifies the relative importance of the entropy term w.r.t. reward and cost. D is the replay buffer and θ indicates parameters of the policy π. Consequently, κ is learnt by minimizing the loss:

$$J_s(\theta) = \mathop{\mathbb{E}}_{s_t\sim D_t, a_t\sim\pi_\theta}[\kappa(d - Q^c_{\pi_\theta}(s_t,a_t))] \tag{4}$$

so κ decreases when $d \geq Q^c_{\pi_\theta}$ and increases otherwise to enforce safety more.

3. Worst-Case Soft Actor Critic

SAC-Lagrangian models the long-term cost of following a policy as its expected value. However, given uncertainty inherent to the environment and the policy, the expected long-term cost can consistently underestimate the potential risk of the agents trajectory. Worse, this behavior can go undetected by the algorithm, if the expected cost is satisfying the constraint, while the true observed cost violates the constraint. For this reason, the authors of [8] follow [7] and learn the distributional safety critic. Consequently, [8] propose focusing on the right tail of the cost distribution for worst case awareness.

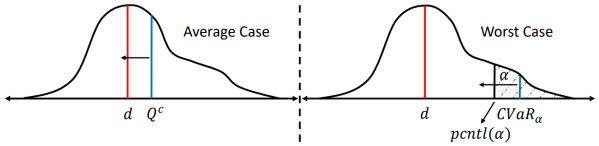


Figure 2. Comparison of risk avoidance strategy in SAC-Lagrangian (left) and WCSAC (right). SAC-Lagrangian pushes the mean of the distribution to satisfy the safety threshold d, resulting in *average case* safety. In contrast, WCSAC forces the right tail of the distribution to be beneath the safety threshold, resulting in *worst case* safety. Source: [8]

Similar to [7], [8] approximate the distributional safety critic $C_\pi(s,a)$ as a Gaussian, 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, the standard Bellman function is used:

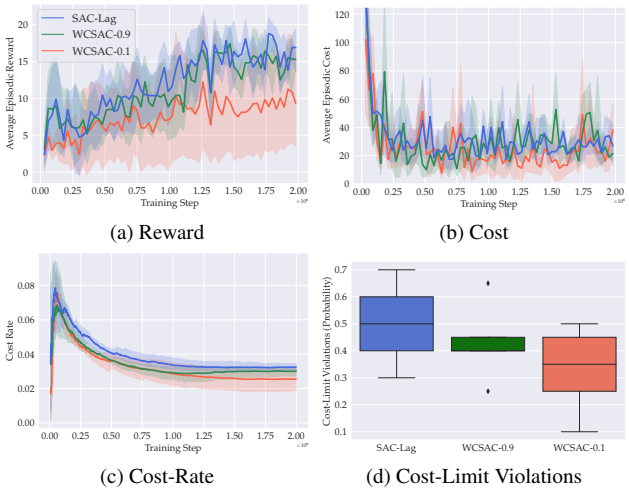
$$\begin{aligned} Q^c_\pi(s,a) = c(s,a) \\ + \gamma \sum_{s'\in S} p(s'|s,a) \sum_{a'\in A} \pi(a'|s') Q^c_\pi(s',a'). \end{aligned} \tag{5}$$

where $s' \sim p(\cdot|s,a)$, $a' \sim \pi(\cdot|s')$. The projection equation for estimating $V^c_\pi(s,a)$ is:

$$\begin{aligned} V^c_\pi(s,a) = c(s,a)^2 - Q^c_\pi(s,a)^2 \\ + 2\gamma c(s,a) \sum_{s'\in S} p(s'|s,a) \sum_{a'\in A} \pi(a'|s') Q^c_\pi(s',a') \\ + \gamma^2 \sum_{s'\in S} p(s'|s,a) \sum_{a'\in A} \pi(a'|s') V^c_\pi(s',a') \\ + \gamma^2 \sum_{s'\in S} p(s'|s,a) \sum_{a'\in A} \pi(a'|s') Q^c_\pi(s',a') \end{aligned} \tag{6}$$

We refer the readers to [7] for the proof of this equation. Consequently, the authors train two neural networks to estimate the safety critic, i.e., Q^c_π and V^c_π.

Built on the distributional safety critic, the authors replace the expected long-term costs with a new safety metric to guide safe exploration, i.e., Conditional Value-at-Risk (CVaR) ([4]), a coherent risk measure. As shown in Figure 2, the CVaR estimates the right tail of the cost distribution leading to higher risk sensitivity due to worst case awareness. This contrasts the average case risk aversion as introduced by the SAC-Lagrangian algorithm and is the main



(a) Reward (b) Cost

(c) Cost-Rate (d) Cost-Limit Violations

Figure 3. Our WCSAC benchmarks on Safety-Gym's `Safexp-PointGoal1-v0` environment. The lines are the average of all runs, and the shaded area is the standard deviation.

source of the superiority of WCSAC over SAC-Lagrangian. By varying the risk level α, the CVaR allows us to control the strength of risk aversion we want to enforce on the policy. We refer the readers to Equation (9) in [8] for more information.

Given the worst case estimation of the expected cost of our current policy the actor loss in Equation 3 is modified to incorporate the $CVaR_\alpha$ (given a risk level α):

$$J_{\pi_1}(\theta) = \mathop{\mathbb{E}}_{s_t\sim D_t, a_t\sim\pi_\theta} [\beta \log\pi_\theta(a_t|s_t) - Q^r_{\pi_\theta}(s_t,a_t) + \kappa\Gamma_\pi(s_t,a_t,\alpha)], \tag{7}$$

with $CVaR_\alpha = \Gamma_\pi(s,a,\alpha)$. Subsequently, $Q^c_{\pi_\theta}(s_t,a_t)$ is substituted by $\Gamma_\pi(s,a,\alpha)$ in Equation 4, as well.

4. Our Contribution

4.1. A PyTorch Toolkit for WCSAC and Safe-RL

We re-implemented and benchmarked the original TensorFlow implementation of WCSAC[1] in PyTorch, the most popular deep learning framework as of August 2022[2]. Compared to the original implementation, our toolkit is more modular, easier to extend, uses a better configuration management framework, i.e., Hydra, and can act as a base code repository for any future Safe-RL projects. Built around OpenAI's Safety-Gym [3], our benchmarks on more complex safety environments are in parallel to the official results reported in [8]. However, our toolkit doesn't support multi-GPU training at the moment and we hope to change this in the near future.

[1] https://github.com/AlgTUDelft/WCSAC/

[2] https://paperswithcode.com/trends

4.2. WCASC+: WCSAC with Robust and Smooth Policy

In theory, learning worst case estimates of the long term expected cost is a principled way to solve the safe exploration problem. Additionally, our experiments show that we can replicate the performance as shown in the WCSAC paper [8]. We even generalize to a more complex environment.

In practice, we observe that increased safety also reliably comes at the cost of reduced reward. As much as this outcome is expected, we also observe some unwanted agent behaviours. When we looked closely at the safest policies in action, we noticed a consistent yet undesirable agent behaviour whenever the agent was in immediate danger. The agent essentially faces a conflict between going through a hazardous area and avoiding it, resulting in it being paralysed in some situations. Roughly speaking, the agent almost looks *scared* to make a move when in front of a hazard. Therefore, despite the principled approach we hypothesize that the WCSAC algorithm suffers from various practical limitations arising from long understood RL optimization artifacts.

Motivated to avoid the above-mentioned behaviour, we propose a two-fold solution. Firstly, we attempt to obtain smoother policies by optimizing actor weights with adversarial regularization [5]. Secondly, we improve the actor's ability to generalize with weight decay [2], to that end we simply optimize the actor with AdamW. To performer the former, we modified the proposed method in [5] - we define the adversarial regularization term as follows: for a fixed state $s \in S$ and a policy π, we encourage the output $\pi(s)$ and $\pi(\tilde{s})$ to be similar, where state $\tilde{s}$ is obtained by injection of a small perturbation to state s. We assume the perturbation set $\mathcal{B}(s,\epsilon) = \{\tilde{s} : ||s-\tilde{s}||_2 \leq \epsilon\}$ is an ϵ-radius ball around s in Euclidean space. To obtain a smooth policy , we encourage smoothness at each state of the entire trajectory. We achieve this by minimizing the following smoothing loss:

$$J_{\pi_2}(\theta) = \mathop{\mathbb{E}}_{s_t\sim D_t, \tilde{s}_t\sim\mathcal{B}(s,\epsilon)} [\lambda_s(D_J(\pi_\theta(\cdot|s), \pi_\theta(\cdot|\tilde{s})))] \tag{8}$$

where $D_J(\cdot,\cdot)$ is Jeffery's Divergence and λ_s is the adversarial regularization coefficient. The Jeffrey's divergence for two distributions P, Q is defined by:

$$D_J(P,Q) = \frac{D_{KL}(P||Q) + D_{KL}(Q||P)}{2} \tag{9}$$

where D_{KL} is Kullback–Leibler divergence.

We argue that our proposed variant could side-step some of the practical limitations, as explained above. Specifically, we hope that smoothing the actor prevents the agent from paralyzing, while regularizing improves the actor's ability to incorporate constraints. Note that we fully understand the

risk of smoothing the policy as the agent might have to make sharp turns in some situations to avoid hazards. However, as our environment uses static hazards (within episodes) we argue that our choice is reasonable for this setting. We emphasize that a deeper investigation of the described limitations is necessary, but out of scope for our project.

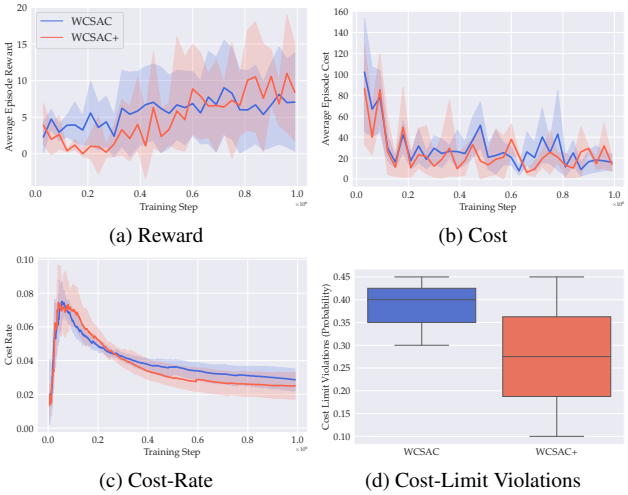
5. Experimental Setup

We benchmark WCSAC and evaluate our variant, WCSAC+, against the former on the Safety-Gym[3] benchmark [3]. We specifically chose a more complex safety environment than used in [8] for novel contribution reasons. As shown in Figure 1, a point agent navigates in a 2D environment to reach the green cylindrical goal while trying to avoid the hazardous areas in purple. All our experiments were evaluated on `Safexp-PointGoal1-v0`, in this, one vase is randomly placed in the scene, but the agent is not penalized for hitting it.

To the best of our abilities, we set the hyperparameters exactly similar to the original implementation, except the learning rate for optimizing κ, i.e., Equation (4). The learning rate set in the original implementation, 0.05, made κ explode in our runs after a few iterations. The following are the hyperparameters used in our benchmark experiments: $lr = 0.001$ with Adam optimizers for all learners (actor, double-Q critic, safety-critic, also β and κ learners), discount $\gamma = 0.99$, target network smoothing temperature $\tau = 0.005$, cost-limit $d = 15$, batch size 256. For all learners, we used an MLP with two intermediate layers, 256 neurons each. A full list of these hyperparameters are available at `config/agent/wcsac.yaml` in our repository. All the agents are trained for 2 million episodes, where the maximal length of each episode is 1000 environment interaction steps. The agent is evaluated every 30000 steps on 20 episodes. All benchmarks were averaged over 5 random seeds. Overall, we compare SAC-Lagrangian (SAC-Lag) and WCSAC at two different risk levels in safety, i.e., WCSAC-0.9 ($\alpha = 0.9$: almost risk-neutral) and WCSAC-0.1 ($\alpha = 0.1$: highly risk-averse).

WCSAC+ has similar hyperparameters to WCSAC except the actor optimizer being AdamW (weight decay). Due to computational and time restrictions, we were only able to average WCSAC+ results over 3 random seeds, with training done for only 1 million steps. We set risk-level $\alpha = 0.1$ and choose $\epsilon = 0.001$ and $\lambda_s = 1$ based on a quick grid-search (evaluated on 150,000 training steps). For baseline, we re-use the 5 random seed WCSAC-0.1 runs from benchmark experiments.

[3] https://github.com/openai/safety-gym



(a) Reward (b) Cost

(c) Cost-Rate (d) Cost-Limit Violations

Figure 4. WCSAC vs WCSAC+ ($\alpha = 0.1$)

6. Results and Discussion

Our WCSAC benchmarks report the following metrics: average episodic return, average episodic cost, cost-rate at each environment interaction step (cumulative costs divided by number of steps) and cost-limit violation statistics. Cost-limit violations show how many times out of 20 episodic evaluations the agent, post training, broke the set cost-limit. Figure 3 shows all the benchmark results. We can observe in Figures 3(a) and 3(b) all algorithms learn to reach the goal (fairly high episodic rewards). However, the risk-neutral algorithms, SAC-Lag and WCSAC-0.9, do not satisfy safety constraint the most (higher episodic costs as well), which is expected of them. We can also see in Figure 3(c), that throughout the training steps, risk-averse WCSAC-0.1 has progressively lower cost-rate on average than the other algorithms. This is also supported by the evidence in Figure 3(d), where the agent trained with WCSAC-0.1 algorithm has the least probability of violating the set cost-limit. All these results are consistent with the reported results in [8]. Our key impression of WCSAC algorithm is that it is quite brittle to hyperparameters, as maybe the case with many RL algorithms out there.

Figure 4 shows the comparisons between our proposed variant WCSAC+ and WCSAC. Particularly, after 1 million trianing, we observe WCSAC+ significantly outperforms WCSAC in terms of cost-rate and cost-limit violations, see Figures 4(c) and 4(d). On average, WCSAC+ also incurs slightly lower episodic costs than WCSAC while obtaining similar episodic rewards. Note that WCSAC results were averaged over 5 random seed runs, while WCSAC+ results were averaged over only 3 random seed runs. Despite this key difference, we feel the comparisons made are fair and acceptable and that there is some merit in policy smoothing in Safe-RL setting. Perhaps with better hyperparameter

tuning and/or by evaluating after training for longer than 1 million steps and/or by making λ_s learnable, i.e., yet another Lagrange multiplier, we feel we can have a better understanding of the role of policy smoothing. We leave these investigations for future works.

7. Conclusion

In this project, we successfully benchmark WCSAC algorithm that solves safety-constrained RL problems. In addition, open-source a modular PyTorch toolkit that can act as a base code-repository to any Safe-RL problem. Our benchmark results are consistent with the results reported in the WCSAC paper. We also extend WCSAC with policy smoothing, WCSAC+ is adversarially trained to have smoother policies for better steering of the agent in the environment. Our experiments show that WCSAC+ outperforms WCSAC. We feel a more thorough investigation and hyperparameter tuning is necessary to see how policy smoothing contributes in Safe-RL setting.

References

[1] Sehoon Ha, P Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. Learning to walk in the real world with minimal human effort. In *CoRL*, 2020. 1, 2

[2] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ArXiv*, abs/1711.05101, 2017. 3

[3] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. 2019. 1, 3, 4

[4] R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000. 2

[5] Qianli Shen, Y. Li, Haoming Jiang, Zhaoran Wang, and Tuo Zhao. Deep reinforcement learning with smooth policy. *ArXiv*, abs/2003.09534, 2020. 3

[6] Krishna Parasuram Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. Learning to be safe: Deep rl with a safety critic. *ArXiv*, abs/2010.14603, 2020. 1

[7] Yichuan Tang, Jian Zhang, and Ruslan Salakhutdinov. Worst cases policy gradients. In *CoRL*, 2019. 1, 2

[8] Qisong Yang, Thiago D. Simão, Simon H Tindemans, and Matthijs T. J. Spaan. Wcsac: Worst-case soft actor critic for safety-constrained reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35, 2021. 1, 2, 3, 4