# SAFELY TRANSFERRING TO UNSAFE ENVIRONMENTS WITH CONSTRAINED REINFORCEMENT LEARNING

**Ethan Knight**
OpenAI, Stanford Univeristy
ehk@stanford.edu

**Joshua Achiam**
OpenAI, UC Berkeley
jachiam@openai.com

## ABSTRACT

Agents deployed in the real world should operate safely, under constraints appropriate to the environment around them. In this work, we consider the problem of safe transfer: learning a safe, general policy from a low-stakes environment, and then transferring that policy to a more complex, high-stakes environment while continuing to satisfy safety constraints. In our experiments, we investigate safe transfer in an obstacle-avoidance setting, where we train a vision-based locomotion agent for transfer between simulated environments with different kinds of obstacles. In the low-stakes environment, the agent navigates around walls in its path, and in the complex high-stakes environment, the agent must avoid bumping into humanoids that are performing random actions from a motion capture dataset. We find that agents pre-trained in the low-stakes environment incur much lower cumulative cost than agents trained from scratch in the high-stakes environment while maintaining comparable performance, providing evidence and hope that future large-scale constrained reinforcement learning deployments can benefit from the safe transfer approach.

## 1 INTRODUCTION

Reinforcement learning (RL) is a promising technology for solving a variety of problems currently unsolvable by other methods, like tasks that involve high-bandwidth communication and collaboration with humans. However, there are two main obstacles to widespread adoption: first, RL is typically sample inefficient, requiring the collection of thousands or millions of environment interactions to train an agent. Second, the trial-and-error nature of RL makes it inherently unsafe in some settings, especially those where agents interact with humans directly.

Figure 1: Through its first-person camera, the "Doggo" agent learns to perceive and safely navigate around simulated humanoids.

Transfer learning approaches can be used to address the sample efficiency problem, for instance by pretraining agents on a set of related tasks before fine-tuning on the downstream task of interest. While the problem of maximizing task performance through transfer learning is well-studied, satisfying safety requirements through transfer learning is less frequently considered, and is the focus of this work. Additionally, RL algorithms deployed in complex real-world environments may be better suited to dealing with distributional shift on a higher environment level during safe transfer than on a lower physics level, which work in domain randomization has revealed to have a highly variable impact on performance (Tobin et al., 2017; OpenAI et al., 2019).

For a motivating example, consider a robot intended to operate in a warehouse and move boxes. In order to be safe, it must avoid crashing into shelves or humans. A transfer approach is desirable here, because training from scratch in the deployment environment would interfere with normal

warehouse operations in an expensive and possibly unsafe way. Zero-shot transfer is unlikely to be robust to the distributional shift, so fine-tuning in the deployment environment should be applied as well. To minimize risk, the agent should come as close as possible to satisfying safety requirements before being introduced to the deployment environment, and should preserve that constraint-satisfaction on continued training.

In this work, we investigate the feasibility of an approach to safe transfer that uses model-free constrained RL for both pre-training and fine-tuning. Our main contribution is an empirical demonstration that this approach can achieve safe transfer—retention of basic skills and knowledge of safety specifications—between simulated high-dimensional, partially-observed environments involving locomotion, vision, and navigation. Based on our results, we claim that this approach is promising for safe transfer problems in the real world.

## 2 Constrained Reinforcement Learning

To formalize what it means for an RL agent to be safe, we operate in the constrained RL framework (Achiam et al., 2017; Ray et al., 2019). In this framework, the environment defines a cost function $c(s)$ on states $s$, and some episodic cost limit $d$, under which an agent is considered safe, and over which an agent is considered unsafe. Thus, agent learning not only consists of optimizing for the expected return, but carries with it an additional constraint. Any policy with an episodic cost under the cost limit is strictly better than any policy with an episodic cost over the cost limit.

### 2.1 Algorithm

We perform both pre-training and fine-tuning using a distributed, recurrent Soft Actor-Critic (Haarnoja et al., 2018) with an extra Q-network $Q_c$ to predict future episodic cost. Constraints on average episodic cost are approximately enforced using the method of Lagrange multipliers, as in Ray et al. (2019). Thus, the constrained entropy-regularized reinforcement learning optimization objective can be described as follows:

$$\max_{\theta} R(\theta) + \alpha\mathcal{H}(\theta),\ C(\theta) \leq d \implies \max_{\theta} \min_{\lambda \geq 0} R(\theta) + \alpha\mathcal{H}(\theta) - \lambda(C(\theta) - d),$$

where $R(\theta)$ is the expected return of the policy, $C(\theta)$ is the expected cost, and $\mathcal{H}(\theta)$ is the average entropy of the policy. We optimized this objective by alternating between optimizing the Lagrange multiplier $\lambda$ and optimizing the parameters of the model $\theta$ using Adam (Kingma & Ba, 2014).

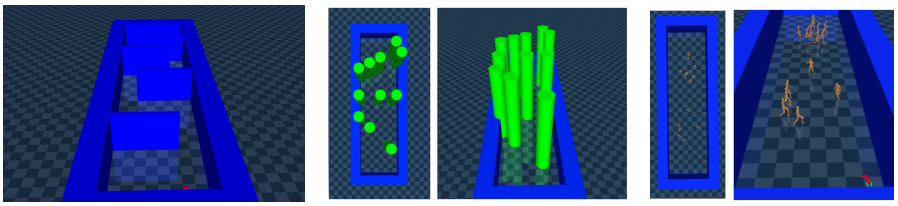
In our implementation, a central learner node performed the optimization using SAC and passed parameters to actor nodes, which passed trajectories to the learner node. The distributed backend used Ray (Moritz et al., 2017). Because the agent had to cope with partial observability, we also cached hidden states from actor rollouts and loaded them into the learner. The resultant setup was similar to ApeX (Horgan et al., 2018) with no burn-in steps.

We found it crucial to also cache the last seen reward and costs and pass them to the recurrent network as inputs concurrent to the observation (see model details in Appendix A). Without information about the last reward and cost, learning in both the low-stakes and high-stakes environments were dramatically less stable. We conjecture that quickly adapting on the fly is a crucial skill of safe complex RL agents, and that providing immediate feedback provides important signal during an episode. In our experiments, the agent learned to take a step back when it received a cost – without information about the previous cost, small imperfections in cost modeling could cause the agent to lean against a wall, repeatedly incurring cost. We present an ablation testing this in Appendix C. Additionally, the full pseudocode is provided in Appendix D.

## 3 Environments

We use environments where an agent must locomote and navigate from vision to travel from one end of a long, cluttered hallway to the other, where obstacles must be avoided for safety reasons. The agent is given a reward for moving forward, and incurs a cost for touching an obstacle.

**Low-stakes environment.** For the low-stakes pre-training environment, the obstacles are walls whose locations are randomly perturbed at the start of each episode. This weak form of domain ran-



(a) (b) (c)

Figure 2: Agents are given rewards for navigating to the far end of the environment. In the low-stakes environment (Figure 2a), the agent must use its vision to navigate around the walls. In the Poles high-stakes environment (Figure 2a) the walls are replaced by poles, and in the Humanoid high-stakes environment (Figure 2c), the agent must perceive and react to humanoids performing behaviors sampled randomly from the CMU MoCap database.

domization is built in so the agent does not memorize the locations of the walls, and is instead forced to respond to novel sensory inputs. The starting location of the agent is also lightly randomized.

**High stakes environments.** Our first "high-stakes" deployment environment has little real-world analogue, but nonetheless tests the ability of the agent to adapt to novel environments where the cost landscape changes. In this "Poles" environment, the reward and cost functions remain unchanged, but the obstacles are poles instead of walls. This presents a challenge, as a transferred agent will have never seen circular objects and can get stuck trying to navigate between poles.

For our second high-stakes environment, we replace all obstacles with simulated people, which perform various movements from the CMU Motion Capture database (De la Torre et al., 2009). This is an extremely difficult transfer problem – not only do the obstacles change form, but they are no longer stationary or entirely predictable. As before, the reward function is unchanged, but because the humanoids can fall on top of the agent, the agent is only penalized the first time it touches a given humanoid.

### 3.1 Agent embodiment

In our experiments, agents are embodied as either a simple "Point" robot or a quadrupedal "Doggo." The Point robot can directly control its heading and forward speed, whereas the Doggo agent must learn to walk.

The agent perceives the world through a camera mounted on its head as a $50 \times 50$ pixel array, in addition to proprioceptive input (joint angles, etc.). In some experiments, we used an RGB representation, but for the experiments shown here, we equip the agent with a depth camera for a visual field of size $50 \times 50 \times 1$. Due to the agents' limited senses, they must learn cope with partial observability.

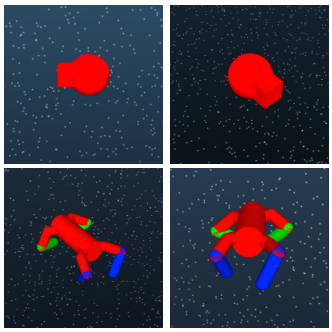
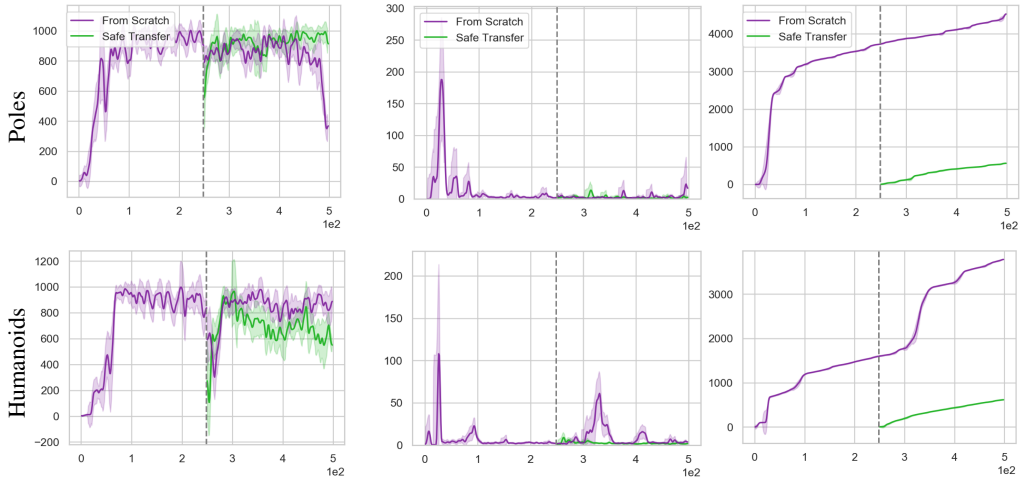


Figure 3: Renderings of Point (top) and Doggo (bottom).

## 4 Experiments and Results

We first demonstrate that unconstrained reward maximization does not trivially achieve safe policies by running our algorithm without training with respect to the cost Q-network. The results from this experiment are shown in Appendix C. One can observe that the optimal unconstrained policy incurs an average episodic cost in the hundreds, which is well outside the realm of acceptability.

In our main experiments, we demonstrate that safe transfer using constrained RL for pre-training and fine-tuning can meet the task and safety performance of training from scratch with constrained RL in the deployment environment, while incurring far fewer lifetime costs in the deployment environment.




(a) Episodic Return (b) Episodic Cost (c) Regret (total cost)

Figure 4: Safe adaptation to high-stakes environments with the Doggo agent. Safe Transfer agents train an equivalent number of epochs in the low-stakes environment, and then transfer to high-stakes environment, indicated by the grey dotted line. Safe Transfer agents incur much less lifetime regret (total accumulated cost) than agents trained from scratch in high-stakes environments.

### 4.1 Safe Transfer.

We compare baseline agents trained from scratch in high-stakes environments to agents pre-trained in the low-stakes environment, presenting zero-shot transfer results (Table 1) and fine-tuning results (Figure 4). Overall, the safe transfer agents incur significantly less total lifetime regret than agents trained from scratch, while achieving comparable task performance. While this result is attributable mostly to the similarity between pre-training and deployment environments, it nonetheless demonstrates that this approach is feasible and scalable, and as a result is promising for problems of interest in the real world.

| | **From Scratch** | | **Zero-shot** | |
|---|---|---|---|---|
| | *Return* | *Cost* | *Return* | *Cost* |
| **P-Poles** | 1090 | 0.4 | 756 | 1000 |
| **P-Humanoids** | 1040 | 16 | 935 | 5.5 |
| **D-Poles** | 800 | 1.8 | 877 | 3.1 |
| **D-Humanoids** | 965 | 2.2 | 946 | 3.9 |

Table 1: Agents transferred from the low-stakes environment do nearly as well as agents trained from scratch in the deployment environment, and can improve in performance via fine-tuning (see Figure 4). Here, "P-" means Point and "D-" means Doggo.

## 5 Conclusion

In this paper, we claimed that safely transferring from a simulated low-stakes environment to a simulated high-stakes environment is an important step towards safe deployment of autonomous learning agents. We were surprised to find that model-free methods worked well in transferring across the environments we tested, in both the zero-shot and few-shot regimes. In future work, we hope to expand on the success of this initial trial by rigorously exploring more approaches, and testing on even more difficult, perhaps real-world environments.

As a closing thought, we highlight that the success of transfer approaches depends on the similarity between the transfer tasks. In future work, we would be interested in exploring how the satisfaction of safety constraints can be assured as the similarity between tasks is varied.

## REFERENCES

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

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014. URL https://arxiv.org/abs/1409.1259.

Fernando De la Torre, Jessica Hodgins, Adam Bargteil, Xavier Martin, Justin Macey, Alex Collado, and Pep Beltran. Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. 2009. URL https://kilthub.cmu.edu/ndownloader/files/12037214.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018. URL http://arxiv.org/abs/1801.01290.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. *CoRR*, abs/1803.00933, 2018. URL http://arxiv.org/abs/1803.00933.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014. URL http://arxiv.org/abs/1412.6980.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pp. 971–980, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging AI applications. *CoRR*, abs/1712.05889, 2017. URL http://arxiv.org/abs/1712.05889.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand, 2019.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. 2019. URL https://cdn.openai.com/safexp-short.pdf.

Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017. URL http://arxiv.org/abs/1703.06907.

## A AGENT DETAILS

To cope with vision input and partial observability, our agent's network incorporates an RNN and a CNN. A schematic is provided in Figure 5, and a parameter count is shown in Table 2.

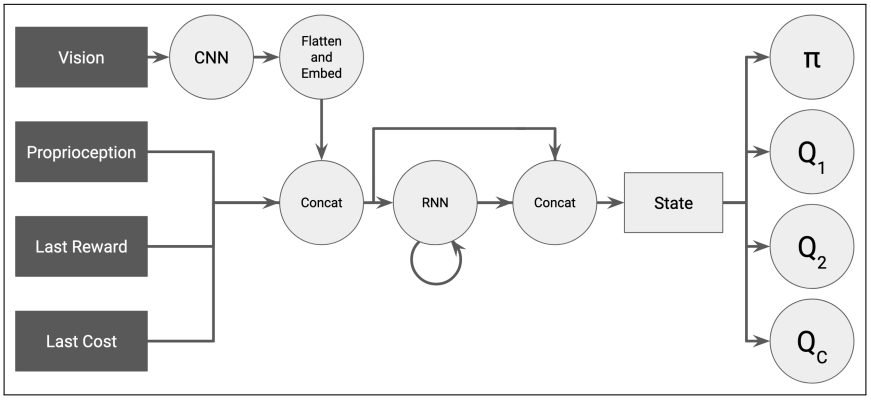



Figure 5: Schematic diagram of model-free constrained agent.

| **Module** | **# Parameters** |
|---|---|
| Vision CNN | $1.4e5$ |
| CNN Embedding | $1.3e5$ |
| RNN (GRU) (Cho et al., 2014) | $4.3e5$ |
| Policy MLP | $3.5e5$ |
| Q MLPs ($\times 3$) | $3.5e5$ |
| Total | $2.1e6$ |

Table 2: Parameter count of final model-free agent.

## B HYPERPARAMETERS

| | |
|---|---|
| # of Actors | 60 |
| CNN architecture | Nature CNN (Mnih et al., 2015) |
| Activation | SeLU (Klambauer et al., 2017) |
| $\alpha$ (entreg coef.) | 0.005 |
| Learner replay size | $2 \times 10^6$ |
| Actor replay size | $10^5$ |
| Batch size | 256 |
| Backprop horizon | 30 |
| $c$ (per-timestep cost limit) | 0.01 |
| Learning rate | $3 \times 10^{-4}$ |
| Max environment timesteps | 1000 |
| Training steps per epoch | 4000 |

Table 3: Hyperparameters for constrained model-free agent

# C Ablations

## C.1 Unconstrained Training

Figures 6 and 7 show results for an unconstrained SAC agent trained on the environments presented. Episodic cost remains high throughout training, indicating that the constrained optimiation is needed to achieve a cost-limit-satisfying policy.

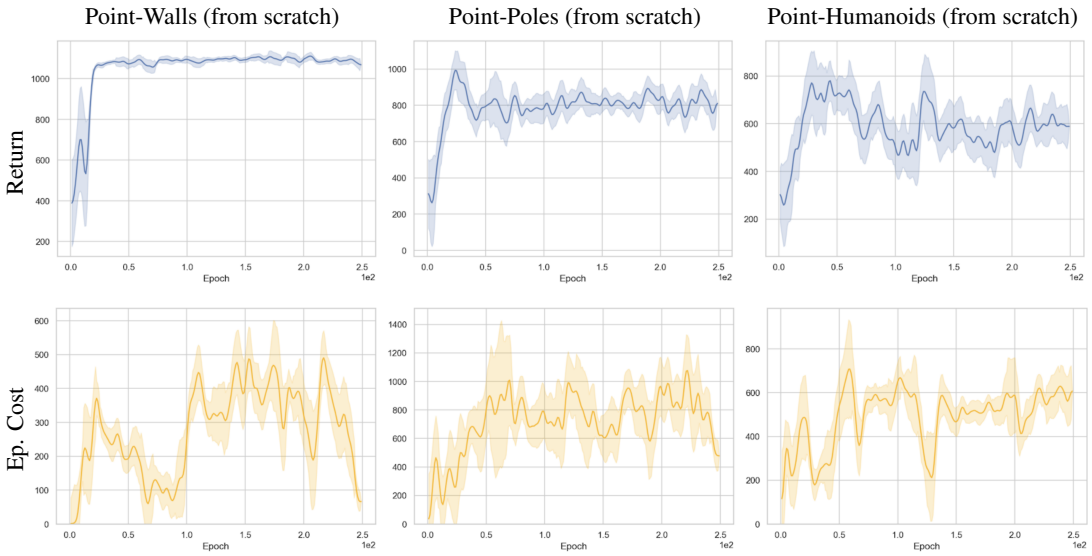



Figure 6: Unconstrained SAC on Point environments.

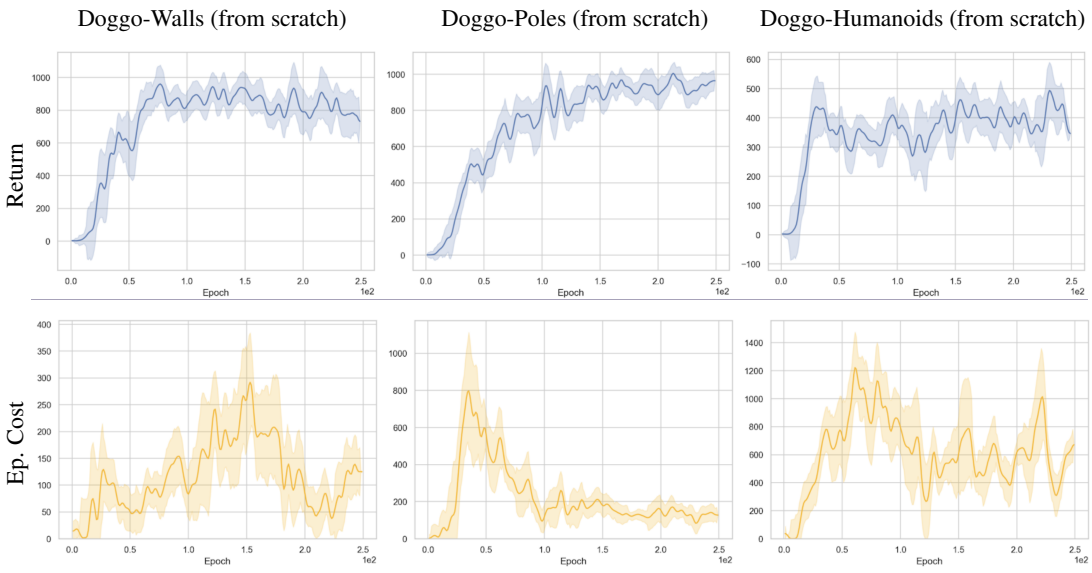



Figure 7: Unconstrained SAC on Doggo environments.

### C.2 ADAPTING ON THE FLY

We hypothesize that giving agent access to environment feedback allows for both faster learning in low-stakes environments and safer adaptation in high-stakes environments. In Figure 8, we experiment with taking away the ability for the agent to perceive its previous reward and cost.

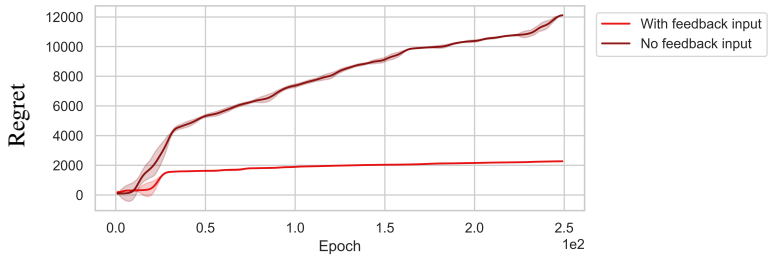



Figure 8: Ablation on providing the previous reward and cost as input to the agent in the Doggo-Walls environment. We find that removing the feedback significantly impairs the agent's ability to be safe.

## D CONSTRAINED SAC PSEUDOCODE

**Algorithm 1** Lagrangian Actor-Critic (SAC-like)

**Require:** Initial Lagrange multiplier $\beta$
**Require:** Cost limit $c$
**Require:** Agent with parameters $\theta$, policy $\pi$, return $Q$-networks $Q_1$, $Q_2$ target networks $Q_1^{\text{targ}}$, $Q_2^{\text{targ}}$, cost $Q$-network $Q_c$ and target network $Q_c^{\text{targ}}$
**while** not trained **do**
Collect experience from environment
Update $Q_1$, $Q_2$ according to entropy-regularized Bellman backup:

$$-\nabla_{\theta_{Q_{1,2}}} \mathbb{E}\left[\left(Q(s,a) - r + \gamma(1-d)\left(\min_{i\in\{1,2\}} Q_i^{\text{targ}}(s',a') - \alpha \log \pi(a'|s')\right)\right)^2\right]$$

Update $Q_c$ according to Bellman cost backup:

$$-\nabla_{\theta_{Q_c}} \mathbb{E}\left[\left(Q_c(s,a) - r + \gamma(1-d)Q_c^{\text{targ}}(s',a')\right)^2\right]$$

Update $\pi$ according to constrained, entropy-regularized objective:

$$\nabla_{\theta_\pi} \mathbb{E}\left[\min_{i\in\{1,2\}} Q_i(s,a(s)) - \alpha \log \pi(a_\theta(s)|s) - \beta Q_c(s,a_\theta(s))\right]$$

Update Lagrange multiplier $\beta$:

$$\nabla_\beta \mathbb{E}\left[\beta(c - Q_c(s,a))\right]$$

Update target networks $Q_1^{\text{targ}}$, $Q_2^{\text{targ}}$, $Q_c^{\text{targ}}$
**end while**