

Soft Actor Critic

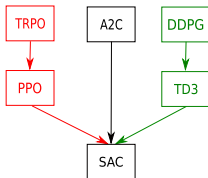
Olivier Sigaud
with the help of Thomas Pierrot

Sorbonne Université
<http://people.isir.upmc.fr/sigaud>



SAC

Soft Actor Critic: The best of two worlds



- ▶ TRPO and PPO: π_θ stochastic, on-policy, **low sample efficiency**, **stable**
- ▶ DDPG and TD3: π_θ deterministic, replay buffer, **better sample efficiency**, **unstable**
- ▶ SAC: “Soft” means “entropy regularized”, π_θ stochastic, replay buffer
- ▶ Adds entropy regularization to favor exploration (follow-up of several papers)
- ▶ **Attempt to be stable and sample efficient**
- ▶ Three successive versions



Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A. Abbeel, P. et al. (2018) Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*



Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018) Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*



Haarnoja, T. Tang, H., Abbeel, P. and Levine, S. (2017) Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*



Soft Actor-Critic

SAC learns a **stochastic** policy π^* maximizing both rewards and entropy:

$$\pi^* = \arg \max_{\pi_{\theta}} \sum_t \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_{\pi_{\theta}}} [r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi_{\theta}(\cdot | \mathbf{s}_t))]$$

- ▶ The entropy is defined as: $\mathcal{H}(\pi_{\theta}(\cdot | \mathbf{s}_t)) = \mathbb{E}_{\mathbf{a}_t \sim \pi_{\theta}(\cdot | \mathbf{s}_t)} [-\log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)]$
- ▶ SAC changes the traditional MDP objective
- ▶ Thus, it converges toward different solutions
- ▶ Consequently, it introduces a new value function, the soft value function
- ▶ As usual, we consider a policy π_{θ} and a soft action-value function $\hat{Q}_{\phi}^{\pi_{\theta}}$



Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. (2016) Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*

Soft policy evaluation

- Usually, we define $\hat{V}_\phi^{\pi_\theta}(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi_\theta(\cdot|\mathbf{s}_t)} [\hat{Q}_\phi^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)]$
- In soft updates, we rather use:

$$\begin{aligned}\hat{V}_\phi^{\pi_\theta}(\mathbf{s}_t) &= \mathbb{E}_{\mathbf{a}_t \sim \pi_\theta(\cdot|\mathbf{s}_t)} \left[\hat{Q}_\phi^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi_\theta(\cdot|\mathbf{s}_t)) \right] \\ &= \mathbb{E}_{\mathbf{a}_t \sim \pi_\theta(\cdot|\mathbf{s}_t)} \left[\hat{Q}_\phi^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] + \alpha \mathbb{E}_{\mathbf{a}_t \sim \pi_\theta(\cdot|\mathbf{s}_t)} [-\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)] \\ &= \mathbb{E}_{\mathbf{a}_t \sim \pi_\theta(\cdot|\mathbf{s}_t)} \left[\hat{Q}_\phi^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) \right]\end{aligned}$$

Critic updates

- We define a standard Bellman operator:

$$\begin{aligned}\mathcal{T}^{\pi} \hat{Q}_{\phi}^{\pi_{\theta}}(s_t, \mathbf{a}_t) &= r(s_t, \mathbf{a}_t) + \gamma V_{\phi}^{\pi_{\theta}}(s_{t+1}) \\ &= r(s_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{a}_t \sim \pi_{\theta}(\cdot | s_{t+1})} \left[\hat{Q}_{\phi}^{\pi_{\theta}}(s_{t+1}, \mathbf{a}_t) - \alpha \log \pi_{\theta}(\mathbf{a}_t | s_{t+1}) \right]\end{aligned}$$

Critic parameters can be learned by minimizing the loss associated to $J_Q(\theta)$:

$$loss_Q(\phi) = \mathbb{E}_{(s_t, \mathbf{a}_t, s_{t+1}) \sim \mathcal{D}} \left[\left(r(s_t, \mathbf{a}_t) + \gamma \hat{V}_{\phi}^{\pi_{\theta}}(s_{t+1}) - \hat{Q}_{\phi}^{\pi_{\theta}}(s_t, \mathbf{a}_t) \right)^2 \right]$$

$$\text{where } V_{\phi}^{\pi_{\theta}}(s_{t+1}) = \mathbb{E}_{\mathbf{a} \sim \pi_{\theta}(\cdot | s_{t+1})} \left[\hat{Q}_{\phi}^{\pi_{\theta}}(s_{t+1}, \mathbf{a}) - \alpha \log \pi_{\theta}(\mathbf{a} | s_{t+1}) \right]$$

- Similar to DDPG update, but with entropy

Actor updates

- Update policy such as to become greedy w.r.t to the soft Q-value
- Choice: update the policy towards the exponential of the soft Q-value

$$J_{\pi}(\theta) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} [KL(\pi_{\theta}(\cdot | \mathbf{s}_t)) || \frac{\exp(\frac{1}{\alpha} \hat{Q}_{\phi}^{\pi_{\theta}}(\mathbf{s}_t, \cdot))}{Z_{\theta}(\mathbf{s}_t)}].$$

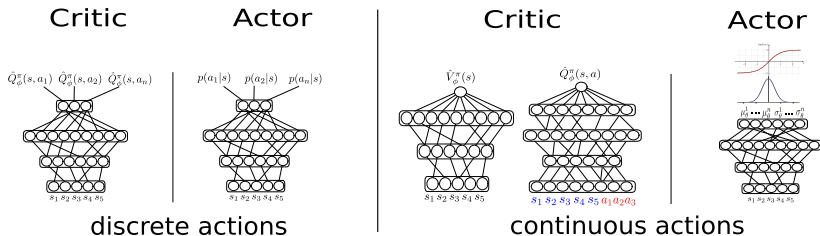
- $Z_{\theta}(\mathbf{s}_t)$ is just a normalizing term to have a distribution
- SAC does not minimize directly this expression but a surrogate one that has the same gradient w.r.t θ

The policy parameters can be learned by minimizing:

$$J_{\pi}(\theta) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\mathbb{E}_{\mathbf{a}_t \sim \pi_{\theta}(\cdot | \mathbf{s}_t)} \left[\alpha \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) - \hat{Q}_{\phi}^{\pi_{\theta}}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]$$

- Similar to DDPG update, but with entropy

Continuous vs discrete actions setting



- ▶ SAC works in both the discrete action and the continuous action setting
- ▶ Discrete action setting:
 - ▶ The critic takes a state and returns a Q-value per action
 - ▶ The actor takes a state and returns probabilities over actions
- ▶ Continuous action setting:
 - ▶ The critic takes a state and an action vector and returns a scalar Q-value
 - ▶ Need to choose a distribution function for the actor
 - ▶ SAC uses a squashed Gaussian: $\mathbf{a} = \tanh(n)$ where $n \sim \mathcal{N}(\mu_\phi, \sigma_\phi)$

Computing the actor loss

- ▶ To compute

$$J_{\pi}(\theta) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\mathbb{E}_{\mathbf{a}_t \sim \pi_{\theta}(\cdot | \mathbf{s}_t)} \left[\alpha \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) - \hat{Q}_{\phi}^{\pi_{\theta}}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]$$

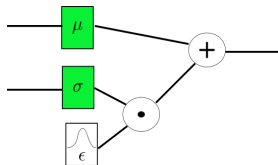
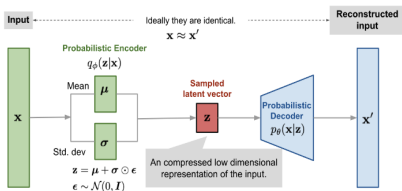
- ▶ SAC needs to estimate an expectation over actions sampled from the actor,
- ▶ That is $\mathbb{E}_{\mathbf{a}_t \sim \pi_{\theta}(\cdot | \mathbf{s}_t)} [F(\mathbf{s}_t, \mathbf{a}_t)]$ where F is a scalar function of the action.
- ▶ In the discrete action setting, $\pi_{\theta}(\cdot | \mathbf{s}_t)$ is a vector of probabilities
 - ▶ $\mathbb{E}_{\mathbf{a}_t \sim \pi_{\theta}(\cdot | \mathbf{s}_t)} [F(\mathbf{s}_t, \mathbf{a}_t)] = \pi_{\theta}(\cdot | \mathbf{s}_t)^T F(\mathbf{s}_t, \cdot)$
 - ▶ No specific difficulty
- ▶ In the continuous action setting:
 - ▶ The actor returns μ_{θ} and σ_{θ}
 - ▶ Re-parameterization trick: $\mathbf{a}_t = \tanh(\mu_{\theta} + \epsilon \cdot \sigma_{\theta})$ where $\epsilon \sim \mathcal{N}(0, 1)$
 - ▶ Thus, $\mathbb{E}_{\mathbf{a}_t \sim \pi_{\theta}(\cdot | \mathbf{s}_t)} [F(\mathbf{s}_t, \mathbf{a}_t)] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, 1)} [F(\mathbf{s}_t, \tanh(\mu_{\theta} + \epsilon \sigma_{\theta}))]$
 - ▶ This trick reduces the variance of the expectation estimate (not always!)
 - ▶ Can still backprop from samples w.r.t θ



Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. (2020) Monte carlo gradient estimation in machine learning. *J. Mach. Learn. Res.*, 21(132):1–62



Reparameterization trick



$$\epsilon \sim p(\epsilon) \quad \gg \quad \mathbf{x} = g_\theta(\epsilon) \quad \gg \quad \mathbb{E}_{\mathbf{x} \sim p_\theta(\mathbf{x})}[f(\mathbf{x})] = \mathbb{E}_\epsilon[f(g_\theta(\epsilon))]$$

Now, we can compute the gradient!

$$\nabla_\theta \mathbb{E}_{\mathbf{x} \sim p_\theta(\mathbf{x})}[f(\mathbf{x})] = \nabla_\theta \mathbb{E}_\epsilon[f(g_\theta(\epsilon))] = \mathbb{E}_\epsilon[\nabla_\theta f(g_\theta(\epsilon))]$$

- In SAC, the reparameterization trick is used to reduce the variance

Critic update improvements (from TD3)

- ▶ As in TD3, SAC uses two critics $\hat{Q}_{\phi_1}^{\pi_\theta}$ and $\hat{Q}_{\phi_2}^{\pi_\theta}$
- ▶ The TD-target becomes:

$$y_t = r + \gamma \mathbb{E}_{\mathbf{a}_{t+1} \sim \pi_\theta(\cdot | \mathbf{s}_{t+1})} \left[\min_{i=1,2} \hat{Q}_{\phi_i}^{\pi_\theta}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - \alpha \log \pi_\theta(\mathbf{a}_{t+1} | \mathbf{s}_{t+1}) \right]$$

And the losses:

$$\begin{cases} J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \sim \mathcal{D}} \left[\left(\hat{Q}_{\phi_1}^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) - y_t \right)^2 + \left(\hat{Q}_{\phi_2}^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) - y_t \right)^2 \right] \\ J_\pi(\theta) = \mathbb{E}_{s \sim \mathcal{D}} \left[\mathbb{E}_{\mathbf{a}_t \sim \pi_\theta(\cdot | \mathbf{s}_t)} \left[\alpha \log \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) - \min_{i=1,2} \hat{Q}_{\phi_i}^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \end{cases}$$

- ▶ Since the actor and critic updates are those of DDPG but with entropy, if we set $\alpha = 0$ and take a deterministic policy, we exactly get TD3



Fujimoto, S., van Hoof, H., & Meger, D. (2018) Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*



Automatic Entropy Adjustment

- ▶ The temperature α needs to be tuned for each task
- ▶ Finding a good α is non trivial
- ▶ Instead of tuning α , tune a lower bound $\mathcal{H}_0$ for the policy entropy
- ▶ And change the optimization problem into a constrained one

$$\begin{cases} \pi^* = \underset{\pi}{\operatorname{argmax}} \sum_t \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_{\pi_{\theta}}} [r(\mathbf{s}_t, \mathbf{a}_t)] \\ \text{s.t. } \forall t \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_{\pi_{\theta}}} [-\log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)] \geq \mathcal{H}_0, \end{cases}$$

- ▶ Use heuristic to compute $\mathcal{H}_0$ from the action space size

α can be learned to satisfy this constraint by minimizing:

$$J(\alpha) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} [\mathbb{E}_{\mathbf{a}_t \sim \pi_{\theta}(\cdot | \mathbf{s}_t)} [-\alpha \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) - \alpha \mathcal{H}_0]]$$

Practical algorithm

- ▶ Initialize neural networks π_θ and $\hat{Q}_\phi^{\pi_\theta}$ weights
- ▶ Play k steps in the environment by sampling actions with π_θ
- ▶ Store the collected transitions in a replay buffer
- ▶ Sample k batches of transitions in the replay buffer
- ▶ Update the temperature α , the actor and the critic using SGD
- ▶ Repeat this cycle until convergence

Any question?



Send mail to: Olivier.Sigaud@upmc.fr



Fujimoto, S., van Hoof, H., and Meger, D. (2018).

Addressing function approximation error in actor-critic methods.

In Dy, J. G. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1582–1591. PMLR.



Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. (2017).

Reinforcement learning with deep energy-based policies.

In *International conference on machine learning*, pages 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 Dy, J. G. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. 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 preprint arXiv:1812.05905.



Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016).

Asynchronous methods for deep reinforcement learning.

In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1928–1937. JMLR.org.



Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. (2020).

Monte carlo gradient estimation in machine learning.

J. Mach. Learn. Res., 21(132):1–62.