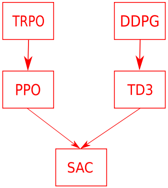
Soft Actor Critic

Olivier Sigaud
with the help of Thomas Pierrot

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

Soft Actor Critic: The best of two worlds



- The TRPO and PPO approach: stochastic policies, on-policy, low sample efficiency, stable
- The DDPG and TD3 approach: deterministic policies, replay buffer, better sample efficiency, unstable
- SAC: stochastic policies + replay buffer + entropy regularization, stable and sample efficient

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*

Exploration through Entropy regularization

- "Soft" means "entropy regularized"
- Follow up of several papers: Soft Q-learning, ...
- Can use a replay buffer → improve sample efficiency
- Claimed to be "off-policy", although this is a relative notion
- Adds entropy regularization to favor exploration
- More entropy means larger choice of actions, avoids collapsing to determinism
- Note: by removing all stochasticity from SAC, we get ≈ TD3

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

Evolution of concepts: three successive versions

- In the first two versions, uses a value function network in addition to an action-value function network
- The second version uses the double critic trick from TD3
- The third version removes the value function network (proved not so useful) and anneals the entropy regularization term

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*

Soft Actor-Critic

SAC learns a **stochastic** policy π^* maximising both rewards and entropy such that:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[r(s_t,a_t) + \alpha\mathcal{H}(\pi(.|s_t))\right]$$

- The entropy is defined as: $\mathcal{H}(\pi(.|s_t)) = \mathbb{E}_{a\sim\pi(.|s)}\left[-\log\left(\pi(a|s)\right)\right]$

Contrast to A2C-like algorithms

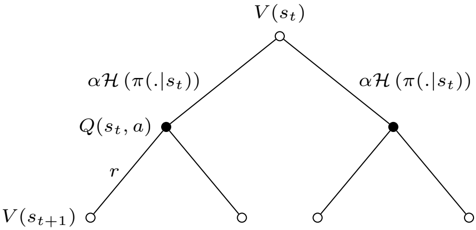
In most RL algorithms:

$$\pi^* = \operatorname*{argmax}_{\pi} \sum_t \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} \left[r(s_t, a_t) \right]$$

- In A2C-like algorithms, the entropy is used externally as an additional exploration mechanism
- SAC changes the traditional MDP objective
- Thus, it converges towards different solutions
- Consequently, it introduces a new value function, the soft action-value function
- SAC parameterizes a policy π_ϕ and a soft Q-value function Q_θ

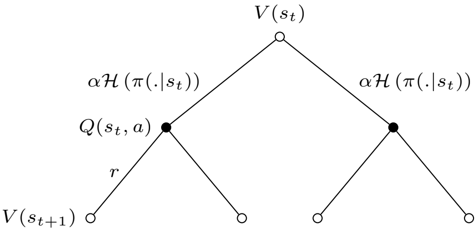
Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. (2016) Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*

Soft policy evaluation


$V(s_t)$
$\alpha\mathcal{H}\left(\pi(.|s_t)\right)$
$\alpha\mathcal{H}\left(\pi(.|s_t)\right)$
$Q(s_t, a)$
r
$V(s_{t+1})$

$$
\begin{aligned}
V(s_t) &= \alpha\mathcal{H}(\pi_\phi(.|s_t)) + \mathbb{E}_{a\sim\pi_\phi(.|s_t)}\left[Q_\theta(s_t, a)\right] \\
&= \alpha\mathbb{E}_{a\sim\pi_\phi(.|s_t)}\left[-\log\left(\pi_\phi(a|s_t)\right)\right] + \mathbb{E}_{a\sim\pi_\phi(.|s_t)}\left[Q_\theta(s_t, a)\right] \\
&= \mathbb{E}_{a\sim\pi_\phi(.|s_t)}\left[Q_\theta(s_t, a) - \alpha\log\left(\pi_\phi(a|s_t)\right)\right]
\end{aligned}
$$

Soft policy evaluation


$V(s_t)$
$\alpha\mathcal{H}\left(\pi(.|s_t)\right)$
$\alpha\mathcal{H}\left(\pi(.|s_t)\right)$
$Q(s_t, a)$
r
$V(s_{t+1})$

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

SAC critic updates

- Given the specific Bellman operator
- Minimization of a specific temporal difference error
- This is the policy evaluation step

The critic parameters can be learned by minimizing:

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}}\left[\left(Q_\theta(s_t,a_t) - (r(s_t,a_t) + \gamma V_\theta(s_{t+1}))\right)^2\right]$$

where $V_\theta(s_{t+1}) = \mathbb{E}_{a\sim\pi_\phi(.|s_{t+1})}\left[Q_\theta(s_{t+1},a) - \alpha\log\left(\pi_\phi(a|s_{t+1})\right)\right]$

SAC 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

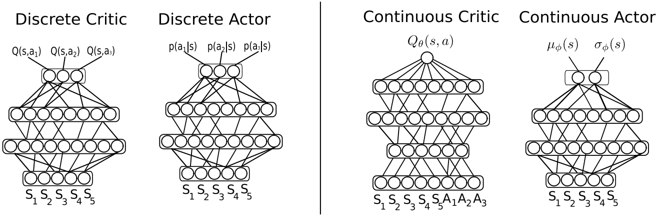
$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}}[KL(\pi_\phi(.|\mathbf{s}_t))||\frac{\exp(\frac{1}{\alpha}Q_\theta(\mathbf{s}_t, .))}{Z_\theta(\mathbf{s}_t)}].$$

- $Z_\theta(\mathbf{s}_t)$ is just a normalizing term to have a distribution
- We do not minimize directly this expression but a surrogate one that has the same gradient w.r.t ϕ
- This is the policy improvement step

The policy parameters can be learned by minimizing:

$$J_\pi(\phi) = \mathbb{E}_{s \sim \mathcal{D}}\left[\mathbb{E}_{a \sim \pi_\phi(.|s)}\left[\alpha \log \pi_\phi(a|s) - Q_\theta(s, a)\right]\right]$$

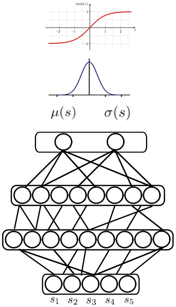
Network architectures


Discrete Critic
Q(s,a$_1$) Q(s,a$_2$) Q(s,a$_3$)
S$_1$ S$_2$ S$_3$ S$_4$ S$_5$
Discrete Actor
p(a$_1$|s) p(a$_2$|s) p(a$_3$|s)
S$_1$ S$_2$ S$_3$ S$_4$ S$_5$
Continuous Critic
$Q_\theta(s,a)$
S$_1$ S$_2$ S$_3$ S$_4$ S$_5$ A$_1$ A$_2$ A$_3$
Continuous Actor
$\mu_\phi(s)$ $\sigma_\phi(s)$
S$_1$ S$_2$ S$_3$ S$_4$ S$_5$

SAC works in both the continuous action and discrete action setting.

- In the discrete action settings:
  - The actor takes a state and returns probabilities over actions
  - The critic takes a state and returns one Q-value per action
- In the continuous action settings:
  - The critic takes a state and an action vector and returns a scalar Q-value
  - Need to choose a distribution parameterisation for the actor
  - SAC uses a squashed Gaussian: $a = \tanh(n)$ where $n \sim \mathcal{N}(\mu_\phi, \sigma_\phi)$
  - In this case, the actor returns μ_ϕ and σ_ϕ

Squashed Gaussian distribution



- Multivariate Gaussian is a generic distribution parameterisation choice
- However its support is non finite whereas action spaces have finite supports (most of the time $[-1,1]^n$)
- Idea: sample according to a Gaussian and rescale the sampled action
- Need for the rescaling operation to be differentiable: choice of *tanh*
- The random variable obtained when "squashing" a Gaussian with a tanh function has a new density distribution
- Tensor libraries do not provide the estimation from squashed Gaussian distributions
- $\mathbf{a} = \tanh(\mathbf{n})$, where $\mathbf{n} \sim \mathcal{N}(\mu_\phi, \sigma_\phi)$
- $\log \pi_\phi(\mathbf{a}|s) = \log \eta(\mathbf{n}|s) - \sum_{i=1}^{D} \log\left(1 - \tanh(n_i)^2\right)$
- where n_i is the i-th component of $\mathbf{n}$ and η is the density of $\mathcal{N}(\mu_\phi, \sigma_\phi)$

Computing expectations

SAC updates require to estimate an expectation over actions sampled from the parameterized policy, i.e. the computation of $\mathbb{E}_{a \sim \pi_\theta(.|s)} \left[F(a, s) \right]$ where F is a scalar function.

- In the discrete action setting:
  - $\mathbb{E}_{a \sim \pi_\phi(.|s)} \left[F(a, s) \right] = \pi_\phi(.|s)^T F(.|s)$ //simple inner product

- In the continuous action setting, use the re-parameterization trick

Re-parameterization trick

- Standard likelihood ratio method, or score function trick: sample $\mathbf{a}_t \sim \pi_\phi$ and reorganize to introduce the gradient into the expectation (REINFORCE-like)
- Re-parameterization trick: rather change the expectation variable : ϵ instead of a
- Use $a = \tanh(\mu_\phi + \epsilon\sigma_\phi)$ where $\epsilon \sim \mathcal{N}(0,1)$
- Thus, $\mathbb{E}_{a\sim\pi_\phi(.|s)}\left[F(a,s)\right] = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}\left[F(\tanh(\mu_\phi + \epsilon\sigma_\phi), s)\right]$
- This expectation does not depend on ϕ anymore
- Thus it is possible to backprop through the expectation w.r.t parameters ϕ
- The re-parameterization trick has lower variance

https://www.youtube.com/watch?v=jmMsNQ2eug4 (8'30)

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel (2015) Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, pp. 3528–3536

Critic update improvements (from TD3)

- Use a target critic with weights $\bar{\theta}$
- Update $\bar{\theta}$ such as $\bar{\theta} \leftarrow (1-\tau)\bar{\theta} + \tau\theta$
- Introduce two critics Q_{θ_1} and Q_{θ_2}
- Compute the TD-target as the minimum to reduce the over-estimation bias

Finally, the TD-target becomes:

$$y_t = r + \gamma \mathbb{E}_{a\sim\pi(.|s_{t+1})}\left[\min_{i=1,2} Q_{\bar{\theta}_i}(s_{t+1}, a) - \alpha \log\left(\pi(a|s_{t+1})\right)\right]$$

And the losses:

$$\begin{cases} J_Q(\theta) = \mathbb{E}_{(s_t, a_t, s_{t+1})\sim\mathcal{D}}\left[\left(Q_{\theta_1}(s_t, a_t) - y_t\right)^2 + \left(Q_{\theta_2}(s_t, a_t) - y_t\right)^2\right] \\ J_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{a\sim\pi_\phi(.|s)}\left[\alpha \log \pi_\phi(a|s) - \min_{i=1,2} Q_{\bar{\theta}_i}(s_{t+1}, a)\right]\right] \end{cases}$$

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

Automatic Entropy Adjustment

- Choosing the optimal temperature α is non trivial
- α needs to be tuned for each task
- We can tune a lower bound $\bar{\mathcal{H}}$ for the policy entropy instead
- Use heuristic to compute $\bar{\mathcal{H}}$ from the action space size
- SAC changes the optimization problem into a constrained one

$$\begin{cases} \pi^* = \underset{\pi}{\operatorname{argmax}} \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[r(s_t,a_t)\right] \\ \text{s.t. } \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[-\log\pi(a_t|s_t)\right] \geq \bar{\mathcal{H}}, \quad \forall t \end{cases}$$

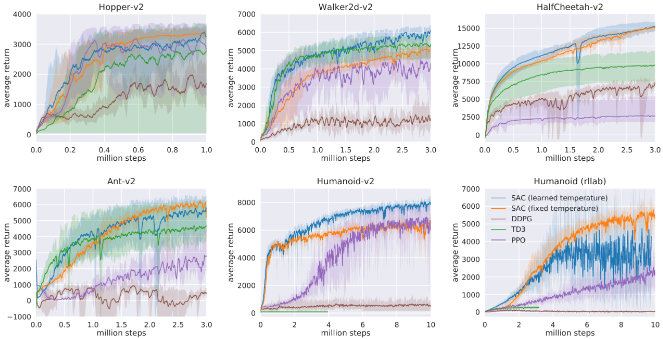
α can be learned to satisfy this constraint by minimizing:

$$\text{loss}(\alpha) = \mathbb{E}_{s_t\sim\mathcal{D}}\left[\mathbb{E}_{a\sim\pi_\phi(.|s_t)}\left[-\alpha\log\pi_\phi(a|s_t) - \alpha\bar{\mathcal{H}}\right]\right]$$

Practical algorithm

- Initialize neural networks π_ϕ and Q_θ weights
- Play k steps in the environment by sampling actions with π_ϕ
- Store the collected transitions in a replay buffer
- Sample k batches of transitions from the replay buffer
- Update the temperature α, the actor and the critic by SGD
- Repeat this cycle until convergence

Performance of SAC



- Is a state-of-the-art algorithm
- Sometimes better than TD3, sometimes not
- Good implementation and additional explanations at `https://spinningup.openai.com/en/latest/algorithms/sac.html`

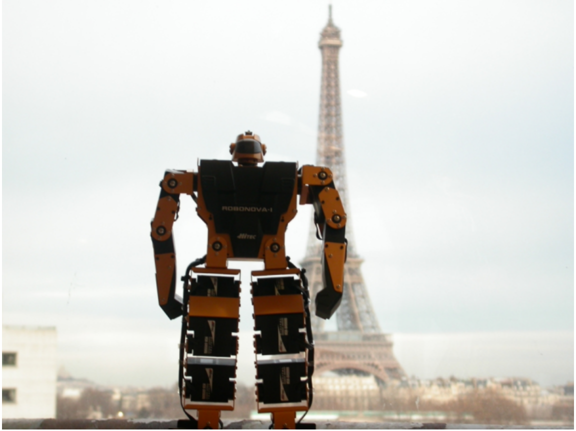
Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *International Conference on Machine Learning*, pp. 151–160, 2019

Practical tips

- Distributing the algorithm in a simple synchronous way works well
- For automatic entropy adjustment:
  - for continuous action spaces: $\bar{\mathcal{H}} = -|\mathcal{A}|$
  - for discrete action spaces: $\bar{\mathcal{H}} = 0.98 \log |\mathcal{A}|$
- Initialization of the layers matters (different init for hidden and final layers)
- Use the same learning rates for the actor, the critic and the temperature
- For MUJOCO environments, use MLPs with one hidden layer of 256 neurons

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., et al. (202) What matters in on-policy reinforcement learning? a large-scale empirical study.arXiv preprint arXiv:2006.05990

Any question?



Send mail to: `Olivier.Sigaud@upmc.fr`

Ahmed, Z., Le Roux, N., Norouzi, M., and Schuurmans, D.
Understanding the impact of entropy on policy optimization.
In *International Conference on Machine Learning*, pp. 151–160, 2019.

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., et al.
What matters in on-policy reinforcement learning? a large-scale empirical study.
arXiv preprint arXiv:2006.05990, 2020.

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

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

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

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

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K.
Asynchronous methods for deep reinforcement learning.
arXiv preprint arXiv:1602.01783, 2016.

Schulman, J., Heess, N., Weber, T., and Abbeel, P.
Gradient estimation using stochastic computation graphs.
In *Advances in Neural Information Processing Systems*, pp. 3528–3536, 2015.