SAC, Eligibility Traces

Milan Straka

📅 November 21, 2022
The paper *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor* by Tuomas Haarnoja et al. from Jan 2018 introduces a different off-policy algorithm for continuous action space.

It was followed by a continuation paper *Soft Actor-Critic Algorithms and Applications* in Dec 2018.

The general idea is to introduce entropy directly in the value function we want to maximize, instead of just ad-hoc adding the entropy penalty. Such an approach is an instance of *regularized policy optimization*. 
Soft Actor Critic Objective

Until now, our goal was to optimize

$$E_{\pi} \left[ G_0 \right].$$

Assume the rewards are deterministic and that $\mu_\pi$ is on-policy distribution of a policy $\pi$.

In the soft actor-critic, the authors instead propose to optimize the maximum entropy objective

$$\pi_* = \arg \max_{\pi} \mathbb{E}_{s \sim \mu_\pi} \left[ \mathbb{E}_{a \sim \pi(s)} \left[ r(s, a) + \alpha H(\pi(\cdot | s)) \right] \right]$$

$$= \arg \max_{\pi} \mathbb{E}_{s \sim \mu_\pi, a \sim \pi(s)} \left[ r(s, a) - \alpha \log \pi(a | s) \right].$$

Note that the value of $\alpha$ is dependent on the magnitude of returns and that for a fixed policy, the entropy penalty can be “hidden” in the reward.
To maximize the regularized objective, we define the following augmented reward:

\[ r_\pi(s, a) \overset{\text{def}}{=} r(s, a) + \mathbb{E}_{s' \sim p(s, a)} \left[ \alpha H(\pi(\cdot | s')) \right]. \]

From now on, we consider **soft action-value** function corresponding to this augmented reward.
Soft Policy Evaluation

Our goal is now to derive soft policy iteration, an analogue of policy iteration algorithm.

We start by considering soft policy evaluation. Let a modified Bellman backup operator $\mathcal{T}_\pi$ be defined as

$$
\mathcal{T}_\pi q(s, a) \overset{\text{def}}{=} r(s, a) + \gamma \mathbb{E}_{s' \sim p(s,a)} [v(s')],
$$

where the soft (state-)value function $v(s)$ is defined as

$$
v(s) = \mathbb{E}_{a \sim \pi} [q(s, a)] + \alpha H(\pi(\cdot | s)) = \mathbb{E}_{a \sim \pi} [q(s, a) - \alpha \log \pi(a | s)].
$$

This modified Bellman backup operator corresponds to the usual one for the augmented rewards $r_\pi(s, a)$, and therefore the repeated application $\mathcal{T}_\pi^k q$ converges to $q_\pi$ according to the original proof.
Soft Policy Improvement

While the soft policy evaluation was a straightforward modification of the original policy evaluation, the soft policy improvement is quite different.

Assume we have a policy $\pi$, its action-value function $q_\pi$ from the soft policy evaluation, and we want to improve the policy. Furthermore, we should select the improved policy from a family of parametrized distributions $\Pi$.

We define the improved policy $\pi'$ as

$$
\pi'(\cdot | s) \overset{\text{def}}{=} \arg \min_{\pi' \in \Pi} J_\pi(\bar{\pi}) = \arg \min_{\pi' \in \Pi} D_{\text{KL}} \left( \bar{\pi}(\cdot | s) \left\| \frac{\exp \left( \frac{1}{\alpha} q_\pi(s, \cdot) \right)}{z_\pi(s)} \right. \right),
$$

where $z_\pi(s)$ is the partition function (i.e., normalization factor such that the right-hand side is a distribution), which does not depend on the new policy and thus can be ignored.
We now prove that $q_{\pi'}(s, a) \geq q_\pi(s, a)$ for any state $s$ and action $a$.

We start by noting that $J_{\pi'}(\pi') \leq J_\pi(\pi)$, because we can always choose $\pi$ as the improved policy. Therefore,

$$\mathbb{E}_{a \sim \pi'} \left[ \alpha \log \pi'(a|s) - q_\pi(s, a) + \alpha \log z_\pi(s) \right] \leq \mathbb{E}_{a \sim \pi} \left[ \alpha \log \pi(a|s) - q_\pi(s, a) + \alpha \log z_\pi(s) \right],$$

which results in

$$\mathbb{E}_{a \sim \pi'} \left[ q_\pi(s, a) - \alpha \log \pi'(a|s) \right] \geq v_\pi(s).$$

We now finish the proof analogously to the original one:

$$q_\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s'} \left[ v_\pi(s') \right]$$

$$\leq r(s, a) + \gamma \mathbb{E}_{s'} \left[ \mathbb{E}_{a' \sim \pi'} \left[ q_\pi(s', a') - \alpha \log \pi'(a'|s') \right] \right]$$

$$\ldots$$

$$\leq q_{\pi'}(s, a).$$
The soft policy iteration algorithm alternates between the soft policy evaluation and soft policy improvement steps.

The repeated application of these two steps produce better and better policies. In other words, we get a monotonically increasing sequence of soft action-value functions.

If the soft action-value function is bounded (the paper assumes a bounded reward and a finite number of actions to bound the entropy), the repeated application converges to some $q_*$, from which we get a $\pi_*$ using the soft policy improvement step. (It is not clear to me why the algorithm should converge in finite time, but we can make the rest of the slide conditional on “if the algorithm converges”).

It remains to show that the $\pi_*$ is indeed the optimal policy fulfilling $q_{\pi_*}(s, a) \geq q_*(s, a)$.

However, this follows from the fact that at convergence, $J_{\pi_*}(\pi_*) \leq J_{\pi_*}(\pi)$, and following the same reasoning as in the proof of the soft policy improvement, we obtain the required $q_{\pi_*}(s, a) \geq q_*(s, a)$. 
The following derivation is not in the original paper, but it is my understanding of how the softmax of the action-value function arises. For simplicity, we assume finite number of actions.

Assuming we have a policy $\pi$ and its action-value function $q_\pi$, we usually improve the policy using

$$\nu(\cdot | s) = \arg \max_{\nu} \mathbb{E}_{a \sim \nu(\cdot | s)} [q_\pi(s, a)]$$

$$= \arg \max_{\nu} \sum_{a} q_\pi(s, a) \nu(a | s)$$

$$= \arg \max_{\nu} q_\pi(s, \cdot)^T \nu(\cdot | s),$$

which results in a greedy improvement with the form of

$$\nu(s) = \arg \max_{a} q_\pi(s, a).$$
Now consider instead the regularized objective

\[ \nu(\cdot|s) = \arg\max_{\nu} \left( \mathbb{E}_{a \sim \nu(\cdot|s)} [q_{\pi}(s, a)] + \alpha H(\nu(\cdot|s)) \right) \]

\[ = \arg\max_{\nu} \left( \mathbb{E}_{a \sim \nu} [q_{\pi}(s, a) - \alpha \log \nu(a|s)] \right) \]

To maximize it for a given \( s \), we form a Lagrangian

\[ \mathcal{L} = \left( \sum_a \nu(a|s)(q_{\pi}(s, a) - \alpha \log \nu(a|s)) \right) - \lambda \left( 1 - \sum_a \nu(a|s) \right). \]

The derivative with respect to \( \nu(a|s) \) is

\[ \frac{\partial \mathcal{L}}{\partial \nu(a|s)} = q_{\pi}(s, a) - \alpha \log \nu(a|s) - \alpha + \lambda. \]

Setting it to zero, we get \( \alpha \log \nu(a|s) = q_{\pi}(s, a) + \lambda - \alpha \), resulting in \( \nu(a|s) \propto e^{\frac{1}{\alpha} q_{\pi}(s, a)} \).
Our soft actor critic will be an off-policy algorithm with continuous action space. The model consist of two critics $q_{\theta_1}$ and $q_{\theta_2}$, two target critics $q_{\bar{\theta}_1}$ and $q_{\bar{\theta}_2}$ and finally a single actor $\pi_{\phi}$.

The authors state that

- with a single critic, all the described experiments still converge;
- they adopted the two critics from the TD3 paper;
- using two critics “significantly speed up training”.
To train the critic, we use the modified Bellman backup operator, resulting in the loss

\[
J_q(\theta_i) = \mathbb{E}_{s \sim \mu, a \sim \pi}(s) \left[ (q_{\theta_i}(s, a) - (r(s, a) + \gamma \mathbb{E}_{s' \sim p(s,a)}[v_{\min}(s')]))^2 \right],
\]

where

\[
v_{\min}(s) = \mathbb{E}_{a \sim \pi}(s) \left[ \min_i (q_{\theta_i}(s, a)) - \alpha \log \pi_\varphi(a|s) \right].
\]

The target critics are updated using exponentiation moving averages with momentum \(\tau\).
Soft Actor Critic – Actor Training

The actor is updated by directly minimizing the KL divergence, resulting in the loss

\[
J_{\pi}(\varphi) = \mathbb{E}_{s \sim \mu_{\pi}, a \sim \pi_{\varphi}(s)} \left[ \alpha \log \left( \pi_{\varphi}(a, s) \right) - \min_i \left( q_{\theta_i}(s, a) \right) \right].
\]

Given that our critics are differentiable, we now reparametrize the policy as

\[
a = f_{\varphi}(s, \varepsilon).
\]

Specifically, we sample $\varepsilon \sim \mathcal{N}(0, 1)$ and let $f_{\varphi}$ produce an unbounded Gaussian distribution (a diagonal one if the actions are vectors).

Together, we obtain

\[
J_{\pi}(\varphi) = \mathbb{E}_{s \sim \mu_{\pi}, \varepsilon \sim \mathcal{N}(0,1)} \left[ \alpha \log \left( \pi_{\varphi}(f_{\varphi}(s, \varepsilon), s) \right) - \min_i \left( q_{\theta_i}(s, f_{\varphi}(s, \varepsilon)) \right) \right].
\]
In practice, the actions need to be bounded.

The authors propose to apply an invertible squashing function $\text{tanh}$ on the unbounded Gaussian distribution.

Consider that our policy produces an unbounded action $\pi(u|s)$. To define a distribution $\tilde{\pi}(a|s)$ with $a = \text{tanh}(u)$, we need to employ the change of variables, resulting in

$$\tilde{\pi}(a|s) = \pi(u|s) \left( \frac{\partial a}{\partial u} \right)^{-1} = \pi(u|s) \left( \frac{\partial \text{tanh}(u)}{\partial u} \right)^{-1}.$$

Therefore, the log-likelihood has quite a simple form

$$\log \tilde{\pi}(a|s) = \log \pi(u|s) - \log (1 - \text{tanh}^2(u)).$$
One of the most important hyperparameters is the entropy penalty $\alpha$.

In the second paper, the authors presented an algorithm for automatic adjustment of its value. Instead of setting the entropy penalty $\alpha$, they propose to specify target entropy value $H$ and then solve a constrained optimization problem

$$\pi_\ast = \arg \max_\pi \mathbb{E}_{s \sim \mu, a \sim \pi(s)} [r(s, a)] \text{ such that } \mathbb{E}_{s \sim \mu, a \sim \pi(s)} [- \log \pi(a | s)] \geq H.$$  

We can then form a Lagrangian with a multiplier $\alpha$

$$\mathbb{E}_{s \sim \mu, a \sim \pi(s)} [r(s, a) + \alpha ( - \log \pi(a | s) - H)],$$

which should be maximized with respect to $\pi$ and minimized with respect to $\alpha \geq 0$. 

---

**Soft Actor Critic – Automatic Entropy Adjustment**

---

NPFL122, Lecture 8  SAC  SPE  SPI  SACAlgorithm  ControlVariates  EligibilityTraces  15/28
To optimize the Lagrangian, we perform *dual gradient descent*, where we alternate between maximization with respect to $\pi$ and minimization with respect to $\alpha$.

While such a procedure is guaranteed to converge only under the convexity assumptions, the authors report that the dual gradient descent works in practice also with nonlinear function approximation.

To conclude, the automatic entropy adjustment is performed by introducing a final loss

$$J(\alpha) = \mathbb{E}_{s \sim \mu, a \sim \pi(s)} \left[ -\alpha \log \pi(a|s) - \alpha \mathcal{H} \right].$$
Algorithm 1 Soft Actor-Critic

**Input:** $\theta_1, \theta_2, \phi$

$\theta_1 \leftarrow \theta_1, \theta_2 \leftarrow \theta_2$

$\mathcal{D} \leftarrow \emptyset$

**for** each iteration **do**

**for** each environment step **do**

$a_t \sim \pi_\phi(a_t|s_t)$

$s_{t+1} \sim p(s_{t+1}|s_t, a_t)$

$\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$

**end for**

**for** each gradient step **do**

$\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$

$\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$

$\alpha \leftarrow \alpha - \lambda_\alpha \hat{\nabla}_\alpha J(\alpha)$

$\bar{\theta}_i \leftarrow \tau \theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1, 2\}$

**end for**

**end for**

**Output:** $\theta_1, \theta_2, \phi$

▷ Initial parameters
▷ Initialize target network weights
▷ Initialize an empty replay pool
▷ Sample action from the policy
▷ Sample transition from the environment
▷ Store the transition in the replay pool
▷ Update the Q-function parameters
▷ Update policy weights
▷ Adjust temperature
▷ Update target network weights
▷ Optimized parameters
## Table 1: SAC Hyperparameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimizer</td>
<td>Adam (Kingma &amp; Ba, 2015)</td>
</tr>
<tr>
<td>learning rate</td>
<td>$3 \cdot 10^{-4}$</td>
</tr>
<tr>
<td>discount ($\gamma$)</td>
<td>0.99</td>
</tr>
<tr>
<td>replay buffer size</td>
<td>$10^6$</td>
</tr>
<tr>
<td>number of hidden layers (all networks)</td>
<td>2</td>
</tr>
<tr>
<td>number of hidden units per layer</td>
<td>256</td>
</tr>
<tr>
<td>number of samples per minibatch</td>
<td>256</td>
</tr>
<tr>
<td>entropy target</td>
<td>$-\text{dim} (\mathcal{A})$ (e.g., -6 for HalfCheetah-v1)</td>
</tr>
<tr>
<td>nonlinearity</td>
<td>ReLU</td>
</tr>
<tr>
<td>target smoothing coefficient ($\tau$)</td>
<td>0.005</td>
</tr>
<tr>
<td>target update interval</td>
<td>1</td>
</tr>
<tr>
<td>gradient steps</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 1: Training curves on continuous control benchmarks. Soft actor-critic (blue and yellow) performs consistently across all tasks and outperforming both on-policy and off-policy methods in the most challenging tasks.
Figure 3. Sensitivity of soft actor-critic to selected hyperparameters on Ant-v1 task. (a) Evaluating the policy using the mean action generally results in a higher return. Note that the policy is trained to maximize also the entropy, and the mean action does not, in general, correspond to the optimal action for the maximum return objective. (b) Soft actor-critic is sensitive to reward scaling since it is related to the temperature of the optimal policy. The optimal reward scale varies between environments, and should be tuned for each task separately. (c) Target value smoothing coefficient $\tau$ is used to stabilize training. Fast moving target (large $\tau$) can result in instabilities (red), whereas slow moving target (small $\tau$) makes training slower (blue).

Figure 3 of "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor" by Tuomas Haarnoja et al.
Off-policy Correction Using Control Variates

Let $G_{t:t+n}$ be the estimated $n$-step return

$$G_{t:t+n} \overset{\text{def}}{=} \left( \sum_{k=t}^{t+n-1} \gamma^{k-t} R_{k+1} \right) + \begin{cases} 0 & \text{if episode ended before } t, \\ V(S_t) & \text{if } n = 0, \\ R_{t+1} + \gamma G_{t+1:t+n} & \text{otherwise.} \end{cases}$$

which can be written recursively as

For simplicity, we do not explicitly handle the first case ("the episode has already ended") in the following.
Off-policy Correction Using Control Variates

Note that we can write

\[ G_{t:t+n} - V(S_t) = R_{t+1} + \gamma G_{t+1:t+n} - V(S_t) \]

\[ = R_{t+1} + \gamma (G_{t+1:t+n} - V(S_{t+1})) + \gamma V(S_{t+1}) - V(S_t), \]

which yields

\[ G_{t:t+n} - V(S_t) = R_{t+1} + \gamma V(S_{t+1}) - V(S_t) + \gamma (G_{t+1:t+n} - V(S_{t+1})). \]

Denoting the TD error as \( \delta_t \triangleq R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \), we can therefore write the \( n \)-step estimated return as a sum of TD errors:

\[ G_{t:t+n} = V(S_t) + \sum_{i=0}^{n-1} \gamma^i \delta_{t+i}. \]

Incidentally, to correctly handle the “the episode has already ended” case, it would be enough to define \( \delta_t \triangleq R_{t+1} + [-\text{done}] \cdot \gamma V(S_{t+1}) - V(S_t). \)
Now consider applying the IS off-policy correction to $G_{t:t+n}$ using the importance sampling ratio

$$\rho_t \overset{\text{def}}{=} \frac{\pi(A_t|S_t)}{b(A_t|S_t)}, \quad \rho_{t:t+n} \overset{\text{def}}{=} \prod_{i=0}^{n} \rho_{t+i}.$$

First note that

$$\mathbb{E}_{A_t \sim b} [\rho_t] = \sum_{A_t} b(A_t|S_t) \frac{\pi(A_t|S_t)}{b(A_t|S_t)} = 1,$$

which can be extended to

$$\mathbb{E}_b [\rho_{t:t+n}] = 1.$$
Until now, we used

\[ G_{t:t+n}^{\text{IS}} \overset{\text{def}}{=} \rho_{t:t+n-1} G_{t:t+n}. \]

However, such correction has unnecessary variance. Notably, when expanding \( G_{t:t+n} \)

\[ G_{t:t+n}^{\text{IS}} = \rho_{t:t+n-1} (R_{t+1} + \gamma G_{t+1:t+n}), \]

the \( R_{t+1} \) depends only on \( \rho_t \), not on \( \rho_{t+1:t+n-1} \), and given that the expectation of the importance sampling ratio is 1, we can simplify to

\[ G_{t:t+n}^{\text{IS}} = \rho_t R_{t+1} + \rho_{t:t+n-1} \gamma G_{t+1:t+n}. \]

Such an estimate can be written recursively as

\[ G_{t:t+n}^{\text{IS}} = \rho_t (R_{t+1} + \gamma G_{t+1:t+n}^{\text{IS}}). \]
We can reduce the variance even further – when $\rho_t = 0$, we might consider returning the value of $V(S_t)$ instead of $0$.

Therefore, we might add another term, the so-called **control variate**, to the estimate

$$G_{t:t+n}^{\text{CV}} \overset{\text{def}}{=} \rho_t (R_{t+1} + \gamma G_{t+1:t+n}^{\text{CV}}) + (1 - \rho_t)V(S_t),$$

which is valid, since the expected value of $1 - \rho_t$ is zero and $\rho_t$ and $S_t$ are independent.

Similarly as before, rewriting to

$$G_{t:t+n}^{\text{CV}} - V(S_t) = \rho_t (R_{t+1} + \gamma G_{t+1:t+n}^{\text{CV}}) - \rho_t V(S_t)$$

$$= \rho_t (R_{t+1} + \gamma V(S_{t+1}) - V(S_t) + \gamma (G_{t+1:t+n}^{\text{CV}} - V(S_{t+1})))$$

results in

$$G_{t:t+n}^{\text{CV}} = V(S_t) + \sum_{i=0}^{n-1} \gamma^i \rho_{t:i} \delta_{t+i}.$$
Eligibility traces are a mechanism of combining multiple $n$-step return estimates for various values of $n$.

First note instead of an $n$-step return, we can use any average of $n$-step returns for different values of $n$, for example $\frac{2}{3} G_{t:t+2} + \frac{1}{3} G_{t:t+4}$.
For a given $\lambda \in [0, 1]$, we define $\lambda$-return as

$$G_t^\lambda \overset{\text{def}}{=} (1 - \lambda) \sum_{i=1}^{\infty} \lambda^{i-1} G_{t:t+i}.$$ 

Alternatively, the $\lambda$-return can be written recursively as

$$G_t^\lambda = (1 - \lambda) G_{t:t+1} + \lambda (R_{t+1} + \gamma G_t^\lambda).$$

**Figure 12.2:** Weighting given in the $\lambda$-return to each of the $n$-step returns.

*Figure 12.2 of "Reinforcement Learning: An Introduction, Second Edition".*
In an episodic task with time of termination $T$, we can rewrite the $\lambda$-return to Figure 12.3 of "Reinforcement Learning: An Introduction, Second Edition".

$$G_t^{\lambda} = (1 - \lambda) \sum_{i=1}^{T-t-1} \lambda^{i-1} G_{t:t+i} + \lambda^{T-t-1} G_t.$$