

# SOFT ACTOR-CRITIC FOR DISCRETE ACTION SETTINGS

A PREPRINT

Petros Christodoulou  
Imperial College London  
petros.christodoulou18@imperial.ac.uk

## 【强化学习 113】Development on SAC



张楚珩

清华大学 交叉信息院博士在读

14 人赞同了该文章

### 原文传送门

Haarnoja, Tuomas, et al. "Soft actor-critic algorithms and applications." arXiv preprint arXiv:1812.05905 (2018).

Christodoulou, Petros. "Soft Actor-Critic for Discrete Action Settings." arXiv preprint arXiv:1910.07207 (2019).

原始 SAC: Haarnoja, Tuomas, et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." *arXiv preprint arXiv:1801.01290* (2018).

### 特色

SAC (soft actor critic) 可以使用 max-entropy 的框架来解释, 策略在动作空间朝着 Q 值较大的动作更新的 greedy 程度可以用一个温度超参数来控制, 在原始的 SAC 版本中, 这个温度参数是固定的。第一篇文章讲了一种自适应调节 SAC 中的温度参数的方法。第二篇文章/Notes 介绍了如何将 SAC 更有效地应用到离散动作空间的任務上。

### 过程

#### 1、回顾

SAC 在每一轮中一边根据样本来更新 Q 函数, 一边更新策略。它们的目标函数如下

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}} \left[ \frac{1}{2} (Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - (r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p} [V_{\bar{\theta}}(\mathbf{s}_{t+1})]))^2 \right], \quad (5)$$

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}} [\alpha \log \pi_\phi(f_\phi(\epsilon_t; \mathbf{s}_t) | \mathbf{s}_t) - Q_\theta(\mathbf{s}_t, f_\phi(\epsilon_t; \mathbf{s}_t))], \quad (9)$$

#### 2、温度参数自适应条件

注意到上面关于策略的目标函数中有一个参数  $\alpha$ 。这一项表示 entropy 项的权重, 注意到第一项是一个 log-probability, 而第二项是 cumulative reward。在不同的任务中 reward 尺度不同, 因此

对应的最优  $\alpha$  也会不同；同时，随着训练的进行，策略不断改善，相应的  $Q$  值也会变化，因此最优的  $\alpha$  也会随之变化。因此这里提出把  $\alpha$  也作为一个参数来动态地训练。

目标如下：

$$\max_{\pi_{0:T}} \mathbb{E}_{\rho_{\pi}} \left[ \sum_{t=0}^T r(s_t, a_t) \right] \text{ s.t. } \mathbb{E}_{(s_t, a_t) \sim \rho_{\pi}} [-\log(\pi_t(a_t|s_t))] \geq \mathcal{H} \quad \forall t \quad (11)$$

即，希望找到一个策略，使得其在样本上的平均 entropy 不小于  $\mathcal{H}$ ，同时该策略需要达到最大的 cumulative reward。可以解得最优的温度  $\alpha$  为

$$\alpha_t^* = \arg \min_{\alpha_t} \mathbb{E}_{a_t \sim \pi_t^*} [-\alpha_t \log \pi_t^*(a_t|s_t; \alpha_t) - \alpha_t \mathcal{H}] . \quad (17)$$

在实际操作的时候，可以制定一个关于参数  $\alpha$  的优化目标并对其做梯度上升：

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi_t} [-\alpha \log \pi_t(a_t|s_t) - \alpha \mathcal{H}] . \quad (18)$$

### 3、SAC-Discrete

为了更好地将 SAC 应用到离散动作空间的任務上，本文做了以下几点变化：

- **Critic Network:** 之前 critic network（ $Q$  函数）的输入是 state 和 action，输出是一个实数；现在 critic network 输入是一个 state 输出是  $A$  个端，分别代表不同动作对应的  $Q$  值；
- **Policy Network:** 以前 policy network 的输出是  $2 * \dim\_A$  维的向量，分别代表高斯分布（with diagonal variance）的均值和方差；现在 policy network 输出的是  $A$  维的 simplex，代表选择到各个动作的概率；
- **Bootstrapped Value:** 以前对于每个状态计算 target  $Q$  value 的时候需要用到下一个状态的  $V$  函数（下式中的红框），而  $V$  函数的计算中的期望（下式中的蓝框）用从策略输出的分布的 MC 采样估计来代替。在离散状态空间的情况下，可以使用下面的 (10) 式来代替 (2) 式，这样能够有效减小 variance。

$$J_Q(\theta) = E_{(s_t, a_t) \sim D} \left[ \frac{1}{2} (Q_{\theta}(s_t, a_t) - (r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_t, a_t)} [V_{\bar{\theta}}(s_{t+1})]))^2 \right] \quad (4)$$

$$V(s_t) := E_{a_t \sim \pi_t} [Q(s_t, a_t) - \alpha \log(\pi(a_t|s_t))] \quad (2)$$

$$V(s_t) := \pi(s_t)^T [Q(s_t) - \alpha \log(\pi(s_t))] \quad (10)$$

- **Alpha Loss:** 类似地，也可以使用 (11) 式来代替 (9) 式。

$$J(\alpha) = E_{a_t \sim \pi_t} [-\alpha (\log \pi_t(a_t|s_t) + \bar{H})] \quad (9)$$

$$J(\alpha) = \pi_t(s_t)^T [-\alpha (\log(\pi_t(s_t)) + \bar{H})] \quad (11)$$

- **Policy Loss:** 动作空间离散的时候不再需要使用 reparameterization trick，可以直接对如下目标函数求导

$$J_{\pi}(\phi) = E_{s_t \sim D} [\pi_{\phi}(s_t)^T [\alpha \log(\pi_{\phi}(s_t)) - Q_{\theta}(s_t)]] \quad (12)$$

最后的算法如下

Algorithm 1 Soft Actor-Critic with Discrete Actions (SAC-Discrete)	
Initialise $Q_{\theta_1} : S \rightarrow \mathbb{R}^{ A }$ , $Q_{\theta_2} : S \rightarrow \mathbb{R}^{ A }$ , $\pi_{\phi} : S \rightarrow [0, 1]^{ A }$	▷ Initialise local networks
Initialise $\bar{Q}_{\theta_1} : S \rightarrow \mathbb{R}^{ A }$ , $\bar{Q}_{\theta_2} : S \rightarrow \mathbb{R}^{ A }$	▷ Initialise target networks
$\theta_1 \leftarrow \bar{\theta}_1$ , $\theta_2 \leftarrow \bar{\theta}_2$	▷ Equalise target and local network weights
$\mathcal{D} \leftarrow \emptyset$	▷ Initialize an empty replay buffer
<b>for</b> each iteration <b>do</b>	
<b>for</b> each environment step <b>do</b>	
$a_t \sim \pi_{\phi}(a_t s_t)$	▷ Sample action from the policy
$s_{t+1} \sim p(s_{t+1} s_t, a_t)$	▷ Sample transition from the environment
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$	▷ Store the transition in the replay buffer
<b>for</b> each gradient step <b>do</b>	
$\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} J(\theta_i)$ for $i \in \{1, 2\}$	▷ Update the Q-function parameters
$\phi \leftarrow \phi - \lambda_{\pi} \nabla_{\phi} J_{\pi}(\phi)$	▷ Update policy weights
$\alpha \leftarrow \alpha - \lambda_{\alpha} \nabla_{\alpha} J(\alpha)$	▷ Update temperature
$\bar{Q}_i \leftarrow \tau Q_i + (1 - \tau) \bar{Q}_i$ for $i \in \{1, 2\}$	▷ Update target network weights
<b>Output</b> $\theta_1, \theta_2, \phi$	▷ Optimized parameters

一点疑问，这里的策略梯度把  $\pi_{\phi}(a_t)$  看做一个 A 维的向量，但其实它应该是一个 simplex，这影响么？

发布于 2020-03-23

强化学习 (Reinforcement Learning)

机器学习

深度学习 (Deep Learning)

▲ 赞同 14 ▼

● 4 条评论

🔗 分享

♥ 喜欢

★ 收藏

...

文章被以下专栏收录



强化学习前沿  
读呀读paper

进入专栏