Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor

Tuomas Haarnoja [1] Aurick Zhou [1] Pieter Abbeel [1] Sergey Levine [1]

【强化学习算法 11】SAC

张楚珩
清华大学 交叉信息院博士在读

从文章的标题就能看得出来，SAC 代表的是 soft actor-critic。

原文传送门：

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：

在前面讲到的【强化学习算法】10. SQL 里面，为了开发一个基于最大熵的off-policy Q-learning算法，中间用到了复杂的approximate inference。相比于这个基于value iteration的SQL来说，这里要讲的基于policy iteration（policy evaluation + policy improvement）的SAC算法就更简单。其主打特色就是sample efficient（off-policy）和robust（maximum entropy framework）。

分类：

Model-free、Energy-based、Off-policy、Continuous State Space、Continuous Action Space

过程：

1. Policy Iteration：

同样从最大化entropy regularized return出发

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot \,|\mathbf{s}_t)) \right]$$

SQL里面的操作是value Iteration，相当于一直作用Bellman operator τ

$$Q_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow r_t + \gamma \, \mathbb{E}_{\mathbf{s}_{t+1} \sim p_{\mathbf{s}}} \left[V_{\text{soft}}(\mathbf{s}_{t+1}) \right], \ \forall \mathbf{s}_t, \mathbf{a}_t$$

$$V_{\text{soft}}(\mathbf{s}_t) \leftarrow \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q_{\text{soft}}(\mathbf{s}_t, \mathbf{a}') \right) d\mathbf{a}', \forall \mathbf{s}_t$$

而这里面的policy evaluation，相当于作用的是 $\mathcal{T}^{\pi}$

$$\mathcal{T}^{\pi} Q(\mathbf{s}_t, \mathbf{a}_t) \triangleq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[V(\mathbf{s}_{t+1})\right]$$

where

$$V(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi}\left[Q(\mathbf{s}_t, \mathbf{a}_t) - \log \pi(\mathbf{a}_t|\mathbf{s}_t)\right]$$

相应的policy improvement就是选择相对于Q函数较好的action，即

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi'(\cdot|\mathbf{s}_t) \,\middle\|\, \frac{\exp\left(Q^{\pi_{\text{old}}}(\mathbf{s}_t, \cdot)\right)}{Z^{\pi_{\text{old}}}(\mathbf{s}_t)}\right)$$

2. Objective Functions：

大致思路就是循环地估计 Q^{π} 和做 $\pi \leftarrow Greedy(Q^{\pi})$ ，为了更稳定，加入了状态价值函数网络 $V_{\psi}(s)$ ，和V值的target network $V_{\bar{\psi}}(s)$ 。critic的估计都用MSE loss，actor的估计使用KL divergence loss。

$$J_V(\psi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}}\left[\frac{1}{2}\left(V_{\psi}(\mathbf{s}_t) - \mathbb{E}_{\mathbf{a}_t \sim \pi_{\phi}}\left[Q_{\theta}(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_{\phi}(\mathbf{a}_t|\mathbf{s}_t)\right]\right)^2\right]$$

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}}\left[\frac{1}{2}\left(Q_{\theta}(\mathbf{s}_t, \mathbf{a}_t) - \hat{Q}(\mathbf{s}_t, \mathbf{a}_t)\right)^2\right]$$

with

$$\hat{Q}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[V_{\bar{\psi}}(\mathbf{s}_{t+1})\right]$$

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\mathrm{D}_{\mathrm{KL}} \left(\pi_\phi(\,\cdot\,|\mathbf{s}_t) \,\middle\|\, \frac{\exp\left(Q_\theta(\mathbf{s}_t, \cdot)\right)}{Z_\theta(\mathbf{s}_t)} \right) \right]$$

得到它们相应的梯度

$$\hat{\nabla}_\psi J_V(\psi) = \nabla_\psi V_\psi(\mathbf{s}_t) \left(V_\psi(\mathbf{s}_t) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t) + \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t) \right)$$

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\mathbf{a}_t, \mathbf{s}_t) \left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - r(\mathbf{s}_t, \mathbf{a}_t) - \gamma V_{\bar{\psi}}(\mathbf{s}_{t+1}) \right)$$

$$\begin{aligned} \hat{\nabla}_\phi J_\pi(\phi) = {} & \nabla_\phi \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t) \\ & + \left(\nabla_{\mathbf{a}_t} \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t) - \nabla_{\mathbf{a}_t} Q(\mathbf{s}_t, \mathbf{a}_t) \right) \nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t) \end{aligned}$$

注意到这里的actor做了和SQL一样的表示方式

$$\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t)$$

算法：

Algorithm 1 Soft Actor-Critic

Initialize parameter vectors ψ, $\bar{\psi}$, θ, ϕ.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$
  end for
end for

SAC是怎么做到off-policy的？

SAC是基于policy iteration来解决控制问题的，off-policy数据可能影响的是policy evaluation的部分，因为该部分需要逼近 Q^π 和 V^π 。 V^π 计算公式里面红框部分保证了目标计算的是当前策略下的价值函数， $Q^\pi(s_t,a_t)$ 的目标依赖的是后一步 $V^\pi(s_{t+1})$ 的估值，只要V值是相对于当前策略的，那么Q值的目标也是相对于当前策略的。此外，由于使用的MSE，因此任何的 (s_t,a_t) 分布下，都能收敛到正确的值。

SQL是怎么做到off-policy的？

SQL是基于Bellman算子 $\mathcal{T}$ 的contraction的，重复操作 $\mathcal{T}$ 就能收敛到 Q^* 。算子 $\mathcal{T}$ 的表述里面是 $\forall s,\forall a$ 的，因此，只要按照算子 $\mathcal{T}$ 规定的计算，任何数据分布都没问题。

ACER/Off-policy actor-critic是怎么做到off-policy的？

它们是基于对actor做相对于expected return的SGD来解控制问题的，求到的policy gradient里面有关于当前策略的期望，如果拿到了off-policy数据，那么就使用importance sampling ratio来保证不产生bias。

注：Off-policy actor-critic指的是Degris, Thomas, Martha White, and Richard S. Sutton. "Off-policy actor-critic."*arXiv preprint arXiv:1205.4839*(2012).

为什么要做off-policy？

现在Deep RL比较诟病的事情是需要经历远比人类高若干数量级的experience才行，在模拟环境中无非就是比较浪费训练时间，但是在实际机械应用场景中，不可能允许把一个实体的机器人摔那么多次。因此提高sample efficiency是一个很核心的问题，其主流方法就是experience replay + off-policy algorithm。

还有没有别的解决方法？

还有一些绕开此问题的方式，比如transfer learning（先学一些好学、模拟便宜的任务，然后迁移到新的任务上，快速学习）、imitation learning（人类专家先提供一些好的策略，然后算法能快速的

学到）、meta-learning（一次性学习一组任务，然后遇到特定任务的时候能够很快学到，跟few-shot learning也有些相似）。

补充

非常好的资料（比读原文更清晰）：

https://spinningup.openai.com/en/latest/algorithms/sac.html
spinningup.openai.com

编辑于 2020-03-17

算法（书籍） 强化学习 (Reinforcement Learning) 算法

文章被以下专栏收录