

# Reinforcement Learning with Deep Energy-Based Policies

Tuomas Haarnoja<sup>\*1</sup> Haoran Tang<sup>\*2</sup> Pieter Abbeel<sup>1,3,4</sup> Sergey Levine<sup>1</sup>

## 【强化学习算法 10】SQL



张楚珩

清华大学 交叉信息院博士在读

23 人赞同了该文章

并不是数据库的那个 SQL，这里指的是 soft Q-learning，而 soft 指的是策略的形式是价值函数 softmax 的形式，下面就会看到。

原文传送门：

Haarnoja, Tuomas, et al. "Reinforcement learning with deep energy-based policies." arXiv preprint arXiv:1702.08165 (2017).

特色：

从最大熵出发，推导到类似Q-Learning中的value iteration的算法形式；使用了基于能量的策略表示，使得策略表示能力更强。训练得到的结果显示，该算法产生的模型探索更充分，探索到有用的子模式更多，能用于当做初始权值来学习其他类似任务。（从此基础上可能能做transfer learning，甚至meta-learning）

分类：

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

过程：

1. 从最大熵出发，认为最优的策略不仅最大化总的奖励，而且策略足够随机，即在每个状态下产生的action的熵比较大。有

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

2. 很多算法里面随机策略用discrete multinomial distributions或者Gaussian distributions，这样的分布只能表示unimodal的分布形式，这样的表示形式最后收敛的结果基本上是near-deterministic的。要想表示multi-modal的分布，我们需要更强大的表示形式，比如这里energy-based的表示形式

$$\pi(\mathbf{a}_t | \mathbf{s}_t) \propto \exp(-\mathcal{E}(\mathbf{s}_t, \mathbf{a}_t)),$$

3. 在最优情形下，有如下定义的Q函数和V函数，可以看出，V函数相当于对于Q函数的softmax操作，在energy-based的表示里面，V相当于配分函数（指数表示下的归一化因子）。

$$Q_{\text{soft}}^*(\mathbf{s}_t, \mathbf{a}_t) = r_t + \mathbb{E}_{(\mathbf{s}_{t+1}, \dots) \sim \rho_\pi} \left[ \sum_{l=1}^{\infty} \gamma^l (r_{t+l} + \alpha \mathcal{H}(\pi_{\text{MaxEnt}}^*(\cdot | \mathbf{s}_{t+l}))) \right]$$

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

$$\pi_{\text{MaxEnt}}^*(\mathbf{a}_t | \mathbf{s}_t) = \exp \left( \frac{1}{\alpha} (Q_{\text{soft}}^*(\mathbf{s}_t, \mathbf{a}_t) - V_{\text{soft}}^*(\mathbf{s}_t)) \right)$$

4. 把下面式子中的代换看做是一次操作的话，类似Bellman operator，下面操作也是contraction，具有上一条里面写出来的最优不动点。按照下面式子反复迭代就可以收敛到最优的解，文中称soft Q-iteration。为了理解该式子，可以对比Q-learning的表达形式  $Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow r_t + \gamma \max_{\mathbf{a}} Q(\mathbf{s}_{t+1}, \mathbf{a})$ ，并且注意到第二式可以看做是softmax。

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

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

5. 第一个困难是：  $V_{\text{soft}}(\mathbf{s}_t)$  式子里面有对于整个action space的积分，在连续空间里面不可能精确求解。解决方案是进行采样+importance ratio，使用sample sum来代替integration，在初期进行随机均匀采样，后期根据policy来采样。

6. 第二个困难是：即使我们得到了  $Q_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t)$  和  $V_{\text{soft}}(\mathbf{s}_t)$ ，也知道了策略的表示形式  $\pi(\mathbf{a}_t | \mathbf{s}_t) \propto \exp \left( -\frac{1}{\alpha} (Q_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) - V_{\text{soft}}(\mathbf{s}_t)) \right)$ ，但由于这个分布实在比较复杂，当给定一个状态  $\mathbf{s}_t$  时，我们没法根据这个策略采样得到一个action。这里考虑使用一个state-conditioned stochastic network来直接从state到action，写作  $\mathbf{a}_t = \mathbf{f}^{\theta}(\xi; \mathbf{s}_t)$ 。该网络的输入是state和一个高斯采样的随机变量，输出是action。要考虑的就是缩小该网络产生的分布和实际的分布的差别，即最小化

$$J_{\pi}(\phi; \mathbf{s}_t) = \text{D}_{\text{KL}} \left( \pi^{\phi}(\cdot | \mathbf{s}_t) \parallel \exp \left( \frac{1}{\alpha} (Q_{\text{soft}}^{\theta}(\mathbf{s}_t, \cdot) - V_{\text{soft}}^{\theta}) \right) \right)$$

由此我们能计算得到其梯度

$$\frac{\partial J_{\pi}(\phi; \mathbf{s}_t)}{\partial \phi} \propto \mathbb{E}_{\xi} \left[ \Delta f^{\phi}(\xi; \mathbf{s}_t) \frac{\partial f^{\phi}(\xi; \mathbf{s}_t)}{\partial \phi} \right]$$

$$\begin{aligned} \Delta f^{\phi}(\cdot; \mathbf{s}_t) = & \mathbb{E}_{\mathbf{a}_t \sim \pi^{\phi}} \left[ \kappa(\mathbf{a}_t, f^{\phi}(\cdot; \mathbf{s}_t)) \nabla_{\mathbf{a}'} Q_{\text{soft}}^{\theta}(\mathbf{s}_t, \mathbf{a}') \Big|_{\mathbf{a}' = \mathbf{a}_t} \right. \\ & \left. + \alpha \nabla_{\mathbf{a}'} \kappa(\mathbf{a}', f^{\phi}(\cdot; \mathbf{s}_t)) \Big|_{\mathbf{a}' = \mathbf{a}_t} \right] \end{aligned}$$

其中  $\kappa$  是kernel。上述梯度的计算通过采样可以做到。

**结果：**

该算法的主要特点是能很好的抓住multi-modal的分布。即，如果有若干个不同的最优解的时候，该算法能够把这些解都学习到，并且在实际中随机地去执行这些最优解。相比之下，其他的算法（主要对比了DDPG），最后会收敛到其中的任意一个解上。这个的用处是，在学习了一个任务之后，想要迁移学习其他任务的时候，将已经学习到的模型作为初始值，能够更快地学习到新任务。（即，transfer learning）

**算法：**

---

**Algorithm 1** Soft Q-learning

---

$\theta, \phi \sim$  some initialization distributions.  
Assign target parameters:  $\bar{\theta} \leftarrow \theta, \bar{\phi} \leftarrow \phi$ .  
 $\mathcal{D} \leftarrow$  empty replay memory.

**for** each epoch **do**

**for** each  $t$  **do**

**Collect experience**

      Sample an action for  $\mathbf{s}_t$  using  $f^\phi$ :

$\mathbf{a}_t \leftarrow f^\phi(\xi; \mathbf{s}_t)$  where  $\xi \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ .

      Sample next state from the environment:

$\mathbf{s}_{t+1} \sim p_{\mathbf{s}}(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$ .

      Save the new experience in the replay memory:

$\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$ .

**Sample a minibatch from the replay memory**

$\{(\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)}, r_t^{(i)}, \mathbf{s}_{t+1}^{(i)})\}_{i=0}^N \sim \mathcal{D}$ .

**Update the soft Q-function parameters**

      Sample  $\{\mathbf{a}^{(i,j)}\}_{j=0}^M \sim q_{\mathbf{a}'}$  for each  $\mathbf{s}_{t+1}^{(i)}$ .

      Compute empirical soft values  $\hat{V}_{\text{soft}}^{\bar{\theta}}(\mathbf{s}_{t+1}^{(i)})$  in (10).

      Compute empirical gradient  $\hat{\nabla}_{\theta} J_Q$  of (11).

      Update  $\theta$  according to  $\hat{\nabla}_{\theta} J_Q$  using ADAM.

**Update policy**

      Sample  $\{\xi^{(i,j)}\}_{j=0}^M \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  for each  $\mathbf{s}_t^{(i)}$ .

      Compute actions  $\mathbf{a}_t^{(i,j)} = f^\phi(\xi^{(i,j)}, \mathbf{s}_t^{(i)})$ .

      Compute  $\Delta f^\phi$  using empirical estimate of (13).

      Compute empirical estimate of (14):  $\hat{\nabla}_{\phi} J_{\pi}$ .

      Update  $\phi$  according to  $\hat{\nabla}_{\phi} J_{\pi}$  using ADAM.

**end for**

**if** epoch  $\bmod$  update\_interval = 0 **then**

    Update target parameters:  $\bar{\theta} \leftarrow \theta, \bar{\phi} \leftarrow \phi$ .

**end if**

**end for**

---

另外：文章说明了soft Q-learning和entropy-regularized policy gradient的等价性。

另外：为啥这篇文章的算法这么复杂？尤其是，为什么会有后面复杂的approximate inference部分？因为为了做到off-policy，拿到了之前的  $\pi_{\theta}$  都没法用，都需要从当前的类似actor的这个state-conditioned stochastic network中重新采样去重算。

编辑于 2018-10-01

算法

机器学习

强化学习 (Reinforcement Learning)

▲ 赞同 23 ▼

💬 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