

目录	1
----	---

## 目录

1 模版备用	2
2 策略迭代算法	3
3 价值迭代算法	4
4 首次访问蒙特卡洛算法	5
5 Q learning 算法	6
6 Sarsa 算法	7
7 DQN 算法	8
8 DRQN 算法	9
9 PER-DQN 算法	10
10 Policy Gradient 算法	11
11 Advantage Actor Critic 算法	12
12 PPO-Clip 算法	13
13 PPO-KL 散度算法	14
14 DDPG 算法	15
15 SoftQ 算法	16
16 SAC-S 算法	17
17 SAC 算法	18

目录	2
18 GAIL 算法	19
19 MAPPO 算法	20

1 模版备用

算法 <sup>①</sup>
1: 测试

<sup>①</sup>脚注

## 2 策略迭代算法

---

### 策略迭代算法

---

```

1: 初始化状态价值函数  $V(s)$  和策略  $\pi(s)$ 
2: 策略估计:
3: repeat
4:    $\Delta \leftarrow 0$ 
5:   repeat
6:      $v \leftarrow V(s)$ 
7:      $V(s) \leftarrow \sum_{s',r} p(s', r | s, \pi(s)) [r + \gamma V(s')]$ 
8:      $\Delta \leftarrow \max(\Delta, |v - V(s)|)$ 
9:   until 遍历所有的状态  $s \in S$ 
10: until  $\Delta < \theta$ 
11: 策略改进:
12:  $stable\_flag \leftarrow true$ 
13: repeat
14:   根据策略  $\pi(a|s)$  生成动作  $a_{temp}$ 
15:   更新策略:  $\pi(a|s) \leftarrow \arg \max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$ 
16:   if  $a_{temp} \neq \pi(a|s)$  then
17:     说明策略还未收敛,  $stable\_flag \leftarrow false$ 
18:   end if
19: until 遍历所有的状态  $s \in S$ 
20: if  $stable\_flag \leftarrow true$  then
21:   结束迭代并返回最优策略  $\pi \approx \pi_*$  和状态价值函数  $V \approx V_*$ 
22: else
23:   继续执行策略估计
24: end if

```

---

### 3 价值迭代算法

---

#### 价值迭代算法

---

- 1: 初始化一个很小的参数阈值  $\theta > 0$ ，以及状态价值函数  $V(s)$ ，注意终止状态的  $V(s_T) = 0$
  - 2: **repeat**
  - 3:    $\Delta \leftarrow 0$
  - 4:   **repeat**
  - 5:      $v \leftarrow V(s)$
  - 6:      $V(s) \leftarrow \max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$
  - 7:      $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  - 8:   **until** 遍历所有的状态  $s \in S$
  - 9: **until**  $\Delta < \theta$
  - 10: 输出一个确定性策略  $\pi \approx \pi_*$ ，  
且  $\pi(s) = \arg \max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$
-

## 4 首次访问蒙特卡洛算法

---

### 首次访问蒙特卡洛算法

---

```

1: 初始化价值函数  $V(s)$ , 一个空的回报列表  $Returns(s_t)$ 
2: for 回合数  $= 1, M$  do
3:   根据策略  $\pi$  采样一回合轨迹  $\tau = \{s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T, \}$ 
4:   初始化回报  $G \leftarrow 0$ 
5:   for 时步  $t = T-1, T-2, \dots, 0$  do
6:      $G \leftarrow \gamma G + R_{t+1}$ 
7:     repeat
8:       将  $G$  添加到  $Returns(s_t)$ 
9:        $V(s_t) \leftarrow \text{average}(Returns(s_t))$ 
10:    until  $s_t$  第二次出现, 即与历史某个状态  $s_0, \dots, s_{t-1}$  相同
11:   end for
12: end for

```

---



---

<sup>①</sup>脚注

## 5 Q learning 算法

---

### Q-learning 算法<sup>①</sup>

---

- 1: 初始化 Q 表  $Q(s, a)$  为任意值, 但其中  $Q(s_{terminal}, \cdot) = 0$ , 即终止状态对应的 Q 值为 0
  - 2: **for** 回合数 =  $1, M$  **do**
  - 3:   重置环境, 获得初始状态  $s_1$
  - 4:   **for** 时步 =  $1, T$  **do**
  - 5:     根据  $\varepsilon - greedy$  策略采样动作  $a_t$
  - 6:     环境根据  $a_t$  反馈奖励  $r_t$  和下一个状态  $s_{t+1}$
  - 7:     更新策略:
  - 8:      $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$
  - 9:     更新状态  $s_{t+1} \leftarrow s_t$
  - 10:   **end for**
  - 11: **end for**
- 

---

<sup>①</sup>Reinforcement Learning: An Introduction

## 6 Sarsa 算法

---

### Sarsa 算法<sup>①</sup>

---

- 1: 初始化 Q 表  $Q(s, a)$  为任意值, 但其中  $Q(s_{terminal}, \cdot) = 0$ , 即终止状态对应的 Q 值为 0
  - 2: **for** 回合数  $= 1, M$  **do**
  - 3:   重置环境, 获得初始状态  $s_1$
  - 4:   根据  $\varepsilon - greedy$  策略采样初始动作  $a_1$
  - 5:   **for** 时步  $= 1, t$  **do**
  - 6:     环境根据  $a_t$  反馈奖励  $r_t$  和下一个状态  $s_{t+1}$
  - 7:     根据  $\varepsilon - greedy$  策略  $s_{t+1}$  和采样动作  $a_{t+1}$
  - 8:     **更新策略:**
  - 9:      $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
  - 10:    更新状态  $s_{t+1} \leftarrow s_t$
  - 11:    更新动作  $a_{t+1} \leftarrow a_t$
  - 12:   **end for**
  - 13: **end for**
- 

---

<sup>①</sup>Reinforcement Learning: An Introduction



## 7 DQN 算法

---

### DQN 算法<sup>①</sup>

---

```

1: 初始化策略网络参数  $\theta$ 
2: 复制参数到目标网络  $\hat{Q} \leftarrow Q$ 
3: 初始化经验回放  $D$ 
4: for 回合数 = 1,  $M$  do
5:   重置环境, 获得初始状态  $s_t$ 
6:   for 时步 = 1,  $t$  do
7:    根据  $\varepsilon - greedy$  策略采样动作  $a_t$ 
8:    环境根据  $a_t$  反馈奖励  $r_t$  和下一个状态  $s_{t+1}$ 
9:    存储 transition 即  $(s_t, a_t, r_t, s_{t+1})$  到经验回放  $D$  中
10:   更新环境状态  $s_{t+1} \leftarrow s_t$ 
11:   更新策略:
12:   从  $D$  中采样一个 batch 的 transition
13:   计算实际的  $Q$  值, 即  $y_j$ ②
14:   对损失  $L(\theta) = (y_i - Q(s_i, a_i; \theta))^2$  关于参数  $\theta$  做随机梯度下降③
15:   end for
16:   每  $C$  个回合复制参数  $\hat{Q} \leftarrow Q$ ④
17: end for

```

---



---

<sup>①</sup>Playing Atari with Deep Reinforcement Learning

<sup>②</sup>
$$y_i = \begin{cases} r_i & \text{对于终止状态 } s_i \\ r_i + \gamma \max_{a'} Q(s_{i+1}, a'; \theta) & \text{对于非终止状态 } s_i \end{cases}$$

<sup>③</sup> $\theta_i \leftarrow \theta_i - \lambda \nabla_{\theta_i} L_i(\theta_i)$

<sup>④</sup>此处也可像原论文中放到小循环中改成每  $C$  步, 但没有每  $C$  个回合稳定

## 8 DRQN 算法

---

**DRQN 算法<sup>①</sup>**

---

```

1: 初始化策略网络参数  $\theta$ 
2: 复制参数到目标网络  $\hat{Q} \leftarrow Q$ 
3: 初始化经验回放  $D$ 
4: for 回合数 = 1,  $M$  do
5:   重置环境, 获得初始状态的观测  $o_t$ 
6:    $h_0 \leftarrow 0$ 
7:   for 时步 = 1,  $t$  do
8:     根据  $\varepsilon - greedy$  策略采样动作  $a_t$ 
9:     环境根据  $a_t$  反馈奖励  $r_t$  和下一个状态, 生成下一状态的观测  $o_{t+1}$ 
10:    存储 transition 即  $(o_t, a_t, r_t, o_{t+1})$  到经验回放  $D$  中
11:    更新环境状态对应的观测  $o_{t+1} \leftarrow o_t$ 
12:    更新策略:
13:    从  $D$  中采样一个 batch 的 transition, 即
      
$$B = \left\{ (s_j, a_j, r_j, s'_j) \dots (s_{j+\tau}, a_{j+\tau}, r_{j+\tau}, s'_{j+\tau}) \right\}_{j=1}^{\text{batch size}} \subseteq D$$

14:    for 这个 batch 中的每个 transition do
15:       $h_{j-1} \leftarrow 0$ 
16:      for  $k = j$  to  $k = j + \tau$  do
17:        更新 LSTM 网络的隐藏状态  $h_k = Q(s_k, h_{k-1} | \theta_i)$ 
18:      end for
19:      计算实际的  $Q$  值, 即  $y_j$ ②
20:      计算损失  $L(\theta) = (y_i - Q(s_{j+\tau}, a_{j+\tau}, h_{j+\tau-1}; \theta))^2$ 
21:    end for
22:    关于参数  $\theta$  做随机梯度下降③
23:    每  $C$  个回合复制参数  $\hat{Q} \leftarrow Q$ ④
24:  end for
25: end for

```

---



---

<sup>①</sup>Deep Recurrent Q-Learning for Partially Observable MDPs

<sup>②</sup>
$$y_j = \begin{cases} r_j & \text{对于终止状态 } s_{i+1} \\ r_j + \gamma \max_{a'} Q(s_{j+\tau}, a_{j+\tau}, h_{j+\tau-1}; \theta) & \text{对于非终止状态 } s_{i+1} \end{cases}$$

<sup>③</sup> $\theta_i \leftarrow \theta_i - \lambda \nabla_{\theta_i} L_i(\theta_i)$

## 9 PER-DQN 算法

---

### PER-DQN 算法<sup>①</sup>

---

- 1: 初始化策略网络参数  $\theta$
  - 2: 复制参数到目标网络  $\hat{Q} \leftarrow Q$
  - 3: 初始化经验回放  $D$
  - 4: **for** 回合数 = 1,  $M$  **do**
  - 5:   重置环境, 获得初始状态  $s_t$
  - 6:   **for** 时步 = 1,  $t$  **do**
  - 7:     根据  $\varepsilon - greedy$  策略采样动作  $a_t$
  - 8:     环境根据  $a_t$  反馈奖励  $r_t$  和下一个状态  $s_{t+1}$
  - 9:     存储 transition 即  $(s_t, a_t, r_t, s_{t+1})$  到经验回放  $D$ , 并根据 TD-error 损失确定其优先级  $p_t$
  - 10:    更新环境状态  $s_{t+1} \leftarrow s_t$
  - 11:    **更新策略:**
  - 12:    按照经验回放中的优先级别, 每个样本采样概率为  $P(j) = p_j^\alpha / \sum_i p_i^\alpha$ , 从  $D$  中采样一个大小为 batch 的 transition
  - 13:    计算各个样本重要性采样权重  $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
  - 14:    计算 TD-error  $\delta_j$ ; 并根据 TD-error 更新优先级  $p_j$
  - 15:    计算实际的  $Q$  值, 即  $y_j$ <sup>②</sup>
  - 16:    根据重要性采样权重调整损失  $L(\theta) = (y_j - Q(s_j, a_j; \theta) \cdot w_j)^2$ , 并将其关于参数  $\theta$  做随机梯度下降<sup>③</sup>
  - 17:    **end for**
  - 18:    每  $C$  个回合复制参数  $\hat{Q} \leftarrow Q$ <sup>④</sup>
  - 19: **end for**
- 

<sup>①</sup>Playing Atari with Deep Reinforcement Learning

<sup>②</sup>
$$y_i = \begin{cases} r_i & \text{对于终止状态 } s_{i+1} \\ r_i + \gamma \max_{a'} Q(s_{i+1}, a'; \theta) & \text{对于非终止状态 } s_{i+1} \end{cases}$$

<sup>③</sup> $\theta_i \leftarrow \theta_i - \lambda \nabla_{\theta_i} L_i(\theta_i)$

<sup>④</sup>此处也可像原论文中放到小循环中改成每  $C$  步, 但没有每  $C$  个回合稳定

## 10 Policy Gradient 算法

---

### REINFORCE 算法: Monte-Carlo Policy Gradient<sup>①</sup>

---

```

1: 初始化策略参数  $\theta \in \mathbb{R}^{d'}$  ( e.g., to  $\mathbf{0}$  )
2: for 回合数 =  $1, M$  do
3:   根据策略  $\pi(\cdot | \cdot, \theta)$  采样一个 (或几个) 回合的 transition
4:   for 时步 =  $0, 1, 2, \dots, T-1$  do
5:     计算回报  $G \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k$ 
6:     更新策略  $\theta \leftarrow \theta + \alpha \gamma^t G \nabla \ln \pi(A_t | S_t, \theta)$ 
7:   end for
8: end for

```

---



---

<sup>①</sup>Reinforcement Learning: An Introduction

## 11 Advantage Actor Critic 算法

---

### Q Actor Critic 算法

---

```

1: 初始化 Actor 参数  $\theta$  和 Critic 参数  $w$ 
2: for 回合数 = 1,  $M$  do
3:   根据策略  $\pi_\theta(a|s)$  采样一个 (或几个) 回合的 transition
4:   更新 Critic 参数①
5:   for 时步 =  $t + 1, 1$  do
6:     计算 Advantage, 即  $\delta_t = r_t + \gamma Q_w(s_{t+1}, a_{t+1}) - Q_w(s_t, a_t)$ 
7:      $w \leftarrow w + \alpha_w \delta_t \nabla_w Q_w(s_t, a_t)$ 
8:      $a_t \leftarrow a_{t+1}, s_t \leftarrow s_{t+1}$ 
9:   end for
10: 更新 Actor 参数  $\theta \leftarrow \theta + \alpha_\theta Q_w(s, a) \nabla_\theta \log \pi_\theta(a | s)$ 
11: end for

```

---



---

<sup>①</sup>这里结合 TD error 的特性按照从  $t + 1$  到 1 计算法 Advantage 更方便

## 12 PPO-Clip 算法

---

### PPO-Clip 算法<sup>①②</sup>

---

- 1: 初始化策略网络 (Actor) 参数  $\theta$  和价值网络 (Critic) 参数  $\phi$
  - 2: 初始化 Clip 参数  $\epsilon$
  - 3: 初始化 epoch 数  $K$
  - 4: 初始化经验回放  $D$
  - 5: **for** 回合数  $= 1, 2, \dots, M$  **do**
  - 6:   根据策略  $\pi_\theta$  采样  $C$  个时步数据, 收集轨迹  $\tau = s_0, a_0, r_1, \dots, s_t, a_t, r_{t+1}, \dots$  到经验回放  $D$  中
  - 7:   **for** epoch 数  $k = 1, 2, \dots, K$  **do**
  - 8:     计算折扣奖励  $\hat{R}_t$
  - 9:     计算优势函数, 即  $A^{\pi_{\theta_k}} = V_{\phi_k} - \hat{R}_t$
  - 10:    结合重要性采样计算 Actor 损失, 如下:
  - 11:     $L^{CLIP}(\theta) = \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^T \min(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)))$ <sup>③</sup>
  - 12:    梯度下降更新 Actor 参数:  $\theta_{k+1} \leftarrow \theta_k + \alpha_\theta L^{CLIP}(\theta)$
  - 13:    更新 Critic 参数:
  - 14:     $\phi_{k+1} \leftarrow \phi_k + \alpha_\phi \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^T (V_{\phi_k}(s_t) - \hat{R}_t)^2$
  - 15:   **end for**
  - 16: **end for**
- 

---

<sup>①</sup>Proximal Policy Optimization Algorithms

<sup>②</sup><https://spinningup.openai.com/en/latest/algorithms/ppo.html>

<sup>③</sup> $L^{CLIP}(\theta) = \hat{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)]$

## 13 PPO-KL 散度算法

---

### PPO-KL 散度算法<sup>①②</sup>

---

```

1: 初始化策略网络 (Actor) 参数  $\theta$  和价值网络 (Critic) 参数  $\phi$ 
2: 初始化 KL 散度参数  $\lambda$ 
3: 初始化回合数量  $M$ 
4: 初始化 epoch 数量  $K$ 
5: 初始化经验回放  $D$ 
6: for 回合数  $= 1, 2, \dots, M$  do
7:   根据策略  $\pi_{\theta_m}$  采样一个或几个回合数据, 收集  $(s_t, a_t, r_t)$  到经验回放
    $D_m = \{\tau_i\}$  中
8:   for epoch 数  $= 1, 2, \dots, K$  do
9:     计算折扣奖励  $\hat{R}_t$ 
10:    根据值函数  $V_{\Phi_m}$ , 用某种优势估计方法计算优势函数  $\hat{A}_t$ 
11:    通过最大化目标函数  $J_{PPO}(\theta)$  更新参数  $\theta$ :
12:     $J_{PPO}(\theta) = \sum_{t=1}^T \frac{\pi_{\theta}(a_t|s_t)}{\pi_{old}(a_t|s_t)} \hat{A}_t - \lambda KL[\pi_{old}|\pi_{\theta}]$ 
13:    典型方法是 Adam 随机梯度上升
14:    根据均方误差回归拟合值函数, 更新 Critic 参数:
15:     $\Phi_{m+1} \leftarrow \frac{1}{|D_m|T} \sum_{\tau \in D_m} \sum_{t=0}^T (V_{\Phi_m}(s_t) - \hat{R}_t)^2$ 
16:    运用某些梯度下降算法
17:    if  $KL[\pi_{old}|\pi_{\theta}] > \beta_{high} KL_{target}$  then
18:       $\lambda \leftarrow \alpha \lambda$ 
19:    else if  $KL[\pi_{old}|\pi_{\theta}] < \beta_{low} KL_{target}$  then
20:       $\lambda \leftarrow \frac{\lambda}{\alpha}$ 
21:    end if
22:  end for
23: end for

```

---



---

<sup>①</sup>Proximal Policy Optimization Algorithms

<sup>②</sup>Emergence of Locomotion Behaviours in Rich Environments

## 14 DDPG 算法

---

### DDPG 算法<sup>①</sup>

---

- 1: 初始化 critic 网络  $Q(s, a | \theta^Q)$  和 actor 网络  $\mu(s | \theta^\mu)$  的参数  $\theta^Q$  和  $\theta^\mu$
  - 2: 初始化对应的目标网络参数, 即  $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
  - 3: 初始化经验回放  $R$
  - 4: **for** 回合数  $= 1, M$  **do**
  - 5:   选择动作  $a_t = \mu(s_t | \theta^\mu) + \mathcal{N}_t$ ,  $\mathcal{N}_t$  为探索噪声
  - 6:   环境根据  $a_t$  反馈奖励  $s_t$  和下一个状态  $s_{t+1}$
  - 7:   存储  $\text{transition}(s_t, a_t, r_t, s_{t+1})$  到经验回放  $R$  中
  - 8:   更新环境状态  $s_{t+1} \leftarrow s_t$
  - 9:   **更新策略:**
  - 10:   从  $R$  中取出一个随机批量的  $(s_i, a_i, r_i, s_{i+1})$
  - 11:   求得  $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} | \theta^{\mu'}) | \theta^{Q'})$
  - 12:   更新 critic 参数, 其损失为:  $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i | \theta^Q))^2$
  - 13:   更新 actor 参数:  $\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a | \theta^Q) \Big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s | \theta^\mu) \Big|_{s_i}$
  - 14:   软更新目标网络:  $\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}, \theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$
  - 15: **end for**
- 

---

<sup>①</sup>Continuous control with deep reinforcement learning



## 15 SoftQ 算法

---

### SoftQ 算法

---

- 1: 初始化参数  $\theta$  和  $\phi$
  - 2: 复制参数  $\bar{\theta} \leftarrow \theta, \bar{\phi} \leftarrow \phi$
  - 3: 初始化经验回放  $D$
  - 4: **for** 回合数 = 1,  $M$  **do**
  - 5:   **for** 时步 = 1,  $t$  **do**
  - 6:     根据  $\mathbf{a}_t \leftarrow f^\phi(\xi; \mathbf{s}_t)$  采样动作, 其中  $\xi \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - 7:     环境根据  $a_t$  反馈奖励  $s_t$  和下一个状态  $s_{t+1}$
  - 8:     存储 transition 即  $(s_t, a_t, r_t, s_{t+1})$  到经验回放  $D$  中
  - 9:     更新环境状态  $s_{t+1} \leftarrow s_t$
  - 10:    **更新 soft Q 函数参数:**
  - 11:    对于每个  $s_{t+1}^{(i)}$  采样  $\{\mathbf{a}^{(i,j)}\}_{j=0}^M \sim q_{\mathbf{a}'}$
  - 12:    计算 empirical soft values  $V_{\text{soft}}^\theta(\mathbf{s}_t)^{\textcircled{1}}$
  - 13:    计算 empirical gradient  $J_Q(\theta)^{\textcircled{2}}$
  - 14:    根据  $J_Q(\theta)$  使用 ADAM 更新参数  $\theta$
  - 15:    **更新策略:**
  - 16:    对于每个  $s_t^{(i)}$  采样  $\{\xi^{(i,j)}\}_{j=0}^M \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - 17:    计算  $\mathbf{a}_t^{(i,j)} = f^\phi(\xi^{(i,j)}, \mathbf{s}_t^{(i)})$
  - 18:    使用经验估计计算  $\Delta f^\phi(\cdot; \mathbf{s}_t)^{\textcircled{3}}$
  - 19:    计算经验估计  $\frac{\partial J_\pi(\phi; \mathbf{s}_t)}{\partial \phi} \propto \mathbb{E}_\xi \left[ \Delta f^\phi(\xi; \mathbf{s}_t) \frac{\partial f^\phi(\xi; \mathbf{s}_t)}{\partial \phi} \right]$ , 即  $\hat{\nabla}_\phi J_\pi$
  - 20:    根据  $\hat{\nabla}_\phi J_\pi$  使用 ADAM 更新参数  $\phi$
  - 21:
  - 22:   **end for**
  - 23:   每  $C$  个回合复制参数  $\bar{\theta} \leftarrow \theta, \bar{\phi} \leftarrow \phi$
  - 24: **end for**
- 

$$\begin{aligned}
 \textcircled{1} V_{\text{soft}}^\theta(\mathbf{s}_t) &= \alpha \log \mathbb{E}_{q_{\mathbf{a}'}} \left[ \frac{\exp(\frac{1}{\alpha} Q_{\text{soft}}^\theta(\mathbf{s}_t, \mathbf{a}'))}{q_{\mathbf{a}'}(\mathbf{a}')} \right] \\
 \textcircled{2} J_Q(\theta) &= \mathbb{E}_{\mathbf{s}_t \sim q_{\mathbf{s}_t}, \mathbf{a}_t \sim q_{\mathbf{a}_t}} \left[ \frac{1}{2} \left( \hat{Q}_{\text{soft}}^\theta(\mathbf{s}_t, \mathbf{a}_t) - Q_{\text{soft}}^\theta(\mathbf{s}_t, \mathbf{a}_t) \right)^2 \right] \\
 \textcircled{3} \Delta f^\phi(\cdot; \mathbf{s}_t) &= \mathbb{E}_{\mathbf{a}_t \sim \pi^\phi} \left[ \kappa \left( \mathbf{a}_t, f^\phi(\cdot; \mathbf{s}_t) \right) \nabla_{\mathbf{a}'} Q_{\text{soft}}^\theta(\mathbf{s}_t, \mathbf{a}') \Big|_{\mathbf{a}' = \mathbf{a}_t} \right. \\
 &\quad \left. + \alpha \nabla_{\mathbf{a}'} \kappa \left( \mathbf{a}', f^\phi(\cdot; \mathbf{s}_t) \right) \Big|_{\mathbf{a}' = \mathbf{a}_t} \right]
 \end{aligned}$$

## 16 SAC-S 算法

---

### SAC-S 算法<sup>①</sup>

---

```

1: 初始化参数  $\psi, \bar{\psi}, \theta, \phi$ 
2: for 回合数 = 1,  $M$  do
3:   for 时步 = 1,  $t$  do
4:     根据  $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t | \mathbf{s}_t)$  采样动作  $a_t$ 
5:     环境反馈奖励和下一个状态,  $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$ 
6:     存储 transition 到经验回放中,  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$ 
7:     更新环境状态  $\mathbf{s}_{t+1} \leftarrow \mathbf{s}_t$ 
8:     更新策略:
9:      $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$ 
10:     $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$  for  $i \in \{1, 2\}$ 
11:     $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ 
12:     $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$ 
13:   end for
14: end for

```

---



---

<sup>①</sup>Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor

## 17 SAC 算法

---

### SAC 算法<sup>①</sup>

---

- 1: 初始化网络参数  $\theta_1, \theta_2$  以及  $\phi$
  - 2: 复制参数到目标网络  $\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$ ,
  - 3: 初始化经验回放  $D$
  - 4: **for** 回合数 = 1,  $M$  **do**
  - 5:   重置环境, 获得初始状态  $s_t$
  - 6:   **for** 时步 = 1,  $t$  **do**
  - 7:     根据  $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t | \mathbf{s}_t)$  采样动作  $a_t$
  - 8:     环境反馈奖励和下一个状态,  $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$
  - 9:     存储 transition 到经验回放中,  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  - 10:    更新环境状态  $s_{t+1} \leftarrow s_t$
  - 11:    **更新策略:**
  - 12:     更新  $Q$  函数,  $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$  for  $i \in \{1, 2\}$ <sup>②③</sup>
  - 13:     更新策略权重,  $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ <sup>④</sup>
  - 14:     调整 temperature,  $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$ <sup>⑤</sup>
  - 15:     更新目标网络权重,  $\bar{\theta}_i \leftarrow \tau \theta_i + (1 - \tau) \bar{\theta}_i$  for  $i \in \{1, 2\}$
  - 16:   **end for**
  - 17: **end for**
- 

---

<sup>②</sup>Soft Actor-Critic Algorithms and Applications

<sup>③</sup> $J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}} \left[ \frac{1}{2} (Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - (r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p} [V_{\bar{\theta}}(\mathbf{s}_{t+1})]))^2 \right]$

## 18 GAIL 算法

---

### GAIL 算法<sup>①</sup>

---

- 1: 采样专家轨迹  $\tau_E \sim \pi_E$ , 初始化网络模型参数  $\theta_0$  和判别器  $D$  参数  $\omega_0$
- 2: **for** 回合数  $i = 1, 2, \dots$  **do**
- 3:   采样策略轨迹  $\tau_i \sim \pi_{\theta_i}$
- 4:   使用梯度下降更新判别器  $D$  的参数  $\omega_i$ , 梯度为:

$$\hat{\mathbb{E}}_{\tau_i} [\nabla_w \log (D_w(s, a))] + \hat{\mathbb{E}}_{\tau_E} [\nabla_w \log (1 - D_w(s, a))] \quad (1)$$

- 5:   使用判别器  $D$  对策略轨迹  $\tau_i$  的输出作为奖励更新策略  $\pi_{\theta_i}$ <sup>②</sup>
  - 6: **end for**
- 

---

<sup>①</sup>Generative Adversarial Imitation Learning

<sup>②</sup>策略更新方式与策略模型  $\pi_\theta$  有关, 如 PP0-Clip 等.

## 19 MAPPO 算法

---

### MAPPO 算法<sup>①②</sup>

---

- 1: 初始化每个智能体  $u$  的 Critic 网络  $Q_{\phi^u}$  和参数为  $\theta^u$  的 Actor 网络,  $u \in U$
  - 2: 初始化每个智能体  $u$  的目标 Actor 网络  $\pi_{old}^u$  的参数  $\theta_{old}^u \leftarrow \theta^u$  和目标 Critic 网络  $Q_{\bar{\phi}^u}$  的参数  $\bar{\phi}^u \leftarrow \phi^u$
  - 3: 初始化 epoch 数  $K$
  - 4: 初始化经验回放  $D$
  - 5: **for** 回合数 =  $1, 2, \dots, M$  **do**
  - 6:   初始化状态  $s_1$
  - 7:   每个智能体  $u$  都根据各自策略采样  $C$  个时步数据, 收集轨迹  $\tau^u = \{o_t^u, a_t^u, r_{t+1}\}_{t=1}^T$  <sup>③</sup>
  - 8:   对每个时步的每条轨迹
  - 9:   计算折扣奖励  $\{\hat{R}_t^u\}_{t=1}^T$
  - 10:   计算优势函数  $\{A_t^u = V_{\phi_t}^u - \hat{R}_t^u\}_{t=1}^T$
  - 11:   计算  $y_t^u = V_{\phi_t}^u + A_t^u$
  - 12:   将每个时步的数据  $\{[o_t^u, a_t^u, y_t^u, A_t^u]_{u=1}^U\}_{t=1}^T$  都存储到经验回放  $D$  中
  - 13:   **for** epoch 数  $k = 1, 2, \dots, K$  **do**
  - 14:     打乱  $D$  中数据顺序并重新编号
  - 15:     **for**  $j = 0, 1, \dots, \frac{T}{B} - 1$  **do**
  - 16:       选择  $B$  条数据  $\{o_i^u, a_i^u, y_i^u, A_i^u\}_{i=1+Bj}^{B(j+1)}$
  - 17:       计算梯度:
  - 18:        $\Delta\theta^u = \frac{1}{B} \sum_{i=1}^B \{\nabla_{\theta^u} f(r_i(\theta^u), A_i^u)\}$
  - 19:        $\Delta\phi^u = \frac{1}{B} \sum_{i=1}^B \{\nabla_{\phi^u} (y_i^u - V_{\phi^u}(o_i^u))^2\}$
  - 20:       Adam 梯度上升方法计算  $\theta^u$ , Adam 梯度下降方法计算  $\phi^u$
  - 21:     **end for**
  - 22:   **end for**
  - 23:   更新  $\theta_{old}^u \leftarrow \theta^u, \bar{\phi}^u \leftarrow \phi^u$
  - 24: **end for**
- 

<sup>①</sup>Joint Optimization of Handover Control and Power Allocation Based on Multi-Agent Deep Reinforcement Learning

<sup>②</sup>The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games

<sup>③</sup> $o_t^u$  是智能体  $u$  在时刻  $t$  的观测值, 是局部观测值, 而  $s_t$  是全局状态

<sup>④</sup>函数  $f$  是目标函数, 不同的方法如 Clip 或者 KL 散度等目标函数不同