

目录

1 模版备用

算法 ^①
1: 测试

^①脚注

2 Q learning 算法

Q-learning 算法^①

- 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**
-

^①Reinforcement Learning: An Introduction

3 Sarsa 算法

Sarsa 算法^①

- 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**
-

^①Reinforcement Learning: An Introduction

4 DQN 算法

DQN 算法^①

```

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

```

^①Playing Atari with Deep Reinforcement Learning

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

^③ $\theta_i \leftarrow \theta_i - \lambda \nabla_{\theta_i} L_i(\theta_i)$

^④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

5 PER_DQN 算法

PER_DQN 算法^①

- 1: 初始化策略网络参数 θ
 - 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 δ_j ; 并根据 TD-error 更新优先级 p_j
 - 15: 计算实际的 Q 值, 即 y_j ^②
 - 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 θ 做随机梯度下降^③
 - 17: **end for**
 - 18: 每 C 个回合复制参数 $\hat{Q} \leftarrow Q$ ^④
 - 19: **end for**
-

^①Playing Atari with Deep Reinforcement Learning

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

^③ $\theta_i \leftarrow \theta_i - \lambda \nabla_{\theta_i} L_i(\theta_i)$

^④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

6 Policy Gradient 算法

REINFORCE 算法: Monte-Carlo Policy Gradient^①

```
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
```

^①Reinforcement Learning: An Introduction

7 Advantage Actor Critic 算法

Q Actor Critic 算法

- 1: 初始化 Actor 参数 θ 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**
-

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

8 PPO-Clip 算法

PPO-Clip 算法^{①②}

```

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 数量  $K$ 
4: 初始化经验回放  $D$ 
5: 初始化总时步数  $c = 0$ 
6: for 回合数  $= 1, 2, \dots, M$  do
7:   重置环境, 获得初始状态  $s_0$ 
8:   for 时步  $t = 1, 2, \dots, T$  do
9:     计数总时步  $c \leftarrow c + 1$ 
10:    根据策略  $\pi_\theta$  选择  $a_t$ 
11:    环境根据  $a_t$  反馈奖励  $r_t$  和下一个状态  $s_{t+1}$ 
12:    存储  $(s_t, a_t, r_t, s_{t+1})$  到经验回放  $D$  中
13:    if  $c$  被  $C$  整除③ then
14:      for  $k = 1, 2, \dots, K$  do
15:        测试
16:      end for
17:      清空经验回放  $D$ 
18:    end if
19:  end for
20: end for

```

^①Proximal Policy Optimization Algorithms

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

^③即每 C 个时步更新策略

9 DDPG 算法

DDPG 算法^①

- 1: 初始化 critic 网络 $Q(s, a | \theta^Q)$ 和 actor 网络 $\mu(s | \theta^\mu)$ 的参数 θ^Q 和 θ^μ
 - 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**
-

^①Continuous control with deep reinforcement learning

10 SoftQ 算法

SoftQ 算法

- 1: 初始化参数 θ 和 ϕ
 - 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 $V_{\text{soft}}^\theta(\mathbf{s}_t)$ ^①
 - 13: 计算 empirical gradient $J_Q(\theta)$ ^②
 - 14: 根据 $J_Q(\theta)$ 使用 ADAM 更新参数 θ
 - 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 $\Delta f^\phi(\cdot; \mathbf{s}_t)$ ^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 ϕ
 - 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**
-

$$\textcircled{1} V_{\text{soft}}^\theta(\mathbf{s}_t) = \alpha \log \mathbb{E}_{q_{\mathbf{a}'}} \left[\frac{\exp(\frac{1}{\alpha} Q_{\text{soft}}^\theta(\mathbf{s}_t, \mathbf{a}'))}{q_{\mathbf{a}'}(\mathbf{a}')} \right]$$

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

$$\begin{aligned} \textcircled{3} \Delta f^\phi(\cdot; \mathbf{s}_t) = & \mathbb{E}_{\mathbf{a}_t \sim \pi^\phi} \left[\kappa \left(\mathbf{a}_t, f^\phi(\cdot; \mathbf{s}_t) \right) \nabla_{\mathbf{a}'} Q_{\text{soft}}^\theta(\mathbf{s}_t, \mathbf{a}') \Big|_{\mathbf{a}' = \mathbf{a}_t} \right. \\ & \left. + \alpha \nabla_{\mathbf{a}'} \kappa \left(\mathbf{a}', f^\phi(\cdot; \mathbf{s}_t) \right) \Big|_{\mathbf{a}' = \mathbf{a}_t} \right] \end{aligned}$$

11 SAC-S 算法

SAC-S 算法^①

```

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

```

^①Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor

12 SAC 算法

SAC 算法^①

- 1: 初始化网络参数 θ_1, θ_2 以及 ϕ
 - 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 $i \in \{1, 2\}$ ^{②③}
 - 13: 更新策略权重, $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ ^④
 - 14: 调整 temperature, $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$ ^⑤
 - 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**
-

^②Soft Actor-Critic Algorithms and Applications

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

^③ $\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\mathbf{a}_t, \mathbf{s}_t) (Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - (r(\mathbf{s}_t, \mathbf{a}_t) + \gamma (Q_{\bar{\theta}}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - \alpha \log(\pi_\phi(\mathbf{a}_{t+1} | \mathbf{s}_{t+1}))))$

^④ $\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \alpha \log(\pi_\phi(\mathbf{a}_t | \mathbf{s}_t)) + (\nabla_{\mathbf{a}_t} \alpha \log(\pi_\phi(\mathbf{a}_t | \mathbf{s}_t)) - \nabla_{\mathbf{a}_t} Q(\mathbf{s}_t, \mathbf{a}_t)) \nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t), \mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t)$

^⑤ $J(\alpha) = \mathbb{E}_{\mathbf{a}_t \sim \pi_t} [-\alpha \log \pi_t(\mathbf{a}_t | \mathbf{s}_t) - \alpha \bar{\mathcal{H}}]$