目录

1 模版备用

算法[①]

1: 测试

[①]脚注

2 策略迭代算法

策略迭代算法

初始化状态价值函数 $V(s)$ 和策略 $\pi(s)$
策略估计：
repeat
  $\Delta \leftarrow 0$
  repeat
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_{s',r} p\left(s', r \mid s, \pi(s)\right)\left[r + \gamma V\left(s'\right)\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  until 遍历所有的状态 $s \in S$
until $\Delta < \theta$
策略改进：
$stable\_flag \leftarrow true$
repeat
  根据策略 $\pi(a|s)$ 生成动作 a_{temp}
  更新策略：$\pi(a|s) \leftarrow \arg\max_a \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma V\left(s'\right)\right]$
  if $a_{temp} \neq \pi(a|s)$ **then**
    说明策略还未收敛，$stable\_flag \leftarrow false$
  end if
until 遍历所有的状态 $s \in S$
if $stable\_flag \leftarrow true$ **then**
  结束迭代并返回最优策略 $\pi \approx \pi_*$ 和状态价值函数 $V \approx V_*$
else
  继续执行策略估计 •
end if

3 价值迭代算法

价值迭代算法

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 $V(s_T) = 0$
repeat
  $\Delta \leftarrow 0$
  repeat
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma V\left(s'\right)\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  until 遍历所有的状态 $s \in S$
until $\Delta < \theta$
输出一个确定性策略 $\pi \approx \pi_*$，
且 $\pi(s) = \arg\max_a \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma V\left(s'\right)\right]$

4 首次访问蒙特卡洛算法

首次访问蒙特卡洛算法

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

[1] 脚注

5 Q learning 算法

Q-learning 算法[①]

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

[①]Reinforcement Learning: An Introduction

6 Sarsa 算法

Sarsa 算法[①]

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

[①]Reinforcement Learning: An Introduction

7 DQN 算法

DQN 算法[①]

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

[①] Playing Atari with Deep Reinforcement Learning

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

[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

8 DRQN 算法

DRQN 算法[①]

初始化策略网络参数 θ
复制参数到目标网络 $\hat{Q} \leftarrow Q$
初始化经验回放 D
for 回合数 $= 1, M$ **do**
  重置环境, 获得初始状态的观测 o_t
  $h_0 \leftarrow 0$
  for 时步 $= 1, t$ **do**
    根据 $\varepsilon - greedy$ 策略采样动作 a_t
    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}
    存储 transition 即 (o_t, a_t, r_t, o_{t+1}) 到经验回放 D 中
    更新环境状态对应的观测 $o_{t+1} \leftarrow o_t$
    更新策略：
    从 D 中采样一个 batch 的 transition, 即
    $B = \left\{(s_j, a_j, r_j, s_j^{'}) \ldots (s_{j+\tau}, a_{j+\tau}, r_{j+\tau}, s_{j+\tau}^{'})\right\}_{j=1}^{\text{batch size}} \subseteq D$
    for 这个 batch 中的每个 transition **do**
      $h_{j-1} \leftarrow 0$
      for $k = j$ to $k = j + \tau$ **do**
        更新 LSTM 网络的隐藏状态 $h_k = Q(s_k, h_{k-1}|\theta_i)$
      end for
      计算实际的 Q 值，即 y_j[②]
      计算损失 $L(\theta) = (y_i - Q(s_{j+\tau}, a_{j+\tau}, h_{j+\tau-1}; \theta))^2$
    end for
    关于参数 θ 做随机梯度下降[③]
    每 C 个回合复制参数 $\hat{Q} \leftarrow Q$[④]]
  end for
end for

[①] Deep Recurrent Q-Learning for Partially Observable MDPs

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

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

9 PER-DQN 算法

PER-DQN 算法[①]

初始化策略网络参数 θ
复制参数到目标网络 $\hat{Q} \leftarrow Q$
初始化经验回放 D
for 回合数 $= 1, M$ **do**
  重置环境，获得初始状态 s_t
  for 时步 $= 1, t$ **do**
    根据 $\varepsilon - greedy$ 策略采样动作 a_t
    环境根据 a_t 反馈奖励 r_t 和下一个状态 s_{t+1}
    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
    更新环境状态 $s_{t+1} \leftarrow s_t$
    更新策略：
    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
    计算各个样本重要性采样权重 $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
    计算 TD-error δ_j ; 并根据 TD-error 更新优先级 p_j
    计算实际的 Q 值，即 y_j[②]
    根据重要性采样权重调整损失 $L(\theta) = \left(y_j - Q\left(s_j, a_j; \theta\right) \cdot w_j\right)^2$，并将其关于参数 θ 做随机梯度下降[③]
  end for
  每 C 个回合复制参数 $\hat{Q} \leftarrow Q$[④]
end for

[①] Playing Atari with Deep Reinforcement Learning

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

[③] $\theta_i \leftarrow \theta_i - \lambda \nabla_{\theta_i} L_i\left(\theta_i\right)$

[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

10 Policy Gradient 算法

REINFORCE 算法：Monte-Carlo Policy Gradient[①]

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

[①]Reinforcement Learning: An Introduction

11 Advantage Actor Critic 算法

Q Actor Critic 算法

初始化 Actor 参数 θ 和 Critic 参数 w
for 回合数 $= 1, M$ **do**
  根据策略 $\pi_\theta(a|s)$ 采样一个 (或几个) 回合的 transition
  更新 Critic 参数[①]
  for 时步 $= t+1, 1$ **do**
    计算 Advantage，即 $\delta_t = r_t + \gamma Q_w(s_{t+1}, a_{t+1}) - Q_w(s_t, a_t)$
    $w \leftarrow w + \alpha_w \delta_t \nabla_w Q_w(s_t, a_t)$
    $a_t \leftarrow a_{t+1}$,$s_t \leftarrow s_{t+1}$
  end for
  更新 Actor 参数 $\theta \leftarrow \theta + \alpha_\theta Q_w(s, a) \nabla_\theta \log \pi_\theta(a \mid s)$
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

12 PPO-Clip 算法

PPO-Clip 算法[①②]

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

[①] Proximal Policy Optimization Algorithms

[②] https://spinningup.openai.com/en/latest/algorithms/ppo.html

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

13 PPO-KL 散度算法

PPO-KL 散度算法[①②]

初始化策略网络 (Actor) 参数 θ 和价值网络 (Critic) 参数 ϕ
初始化 KL 散度参数 λ
初始化回合数量 M
初始化 epoch 数量 K
初始化经验回放 D
for 回合数 $= 1, 2, \cdots, M$ **do**
  根据策略 π_{θ_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
  for epoch 数 $= 1, 2, \cdots, K$ **do**
    计算折扣奖励 $\hat{R}_t$
    根据值函数 V_{Φ_m}, 用某种优势估计方法计算优势函数 $\hat{A}_t$
    通过最大化目标函数 $J_{PPO}(\theta)$ 更新参数 θ:
    $J_{PPO}(\theta) = \sum_{t=1}^{T} \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)} \hat{A}_t - \lambda KL[\pi_{old}|\pi_\theta]$
    典型方法是 Adam 随机梯度上升
    根据均方误差回归拟合值函数，更新 Critic 参数:
    $\Phi_{m+1} \leftarrow \frac{1}{|D_m|T} \sum_{\tau \in D_m} \sum_{t=0}^{T} (V_{\Phi_m}(s_t) - \hat{R}_t)^2$
    运用某些梯度下降算法
    if $KL[\pi_{old}|\pi_\theta] > \beta_{high} KL_{target}$ **then**
      $\lambda \leftarrow \alpha\lambda$
    else if $KL[\pi_{old}|\pi_\theta] < \beta_{low} KL_{target}$ **then**
      $\lambda \leftarrow \frac{\lambda}{\alpha}$
    end if
  end for
end for

[①] Proximal Policy Optimization Algorithms

[②] Emergence of Locomotion Behaviours in Rich Environments

14 DDPG 算法

DDPG 算法[①]

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

[①] Continuous control with deep reinforcement learning

15 SoftQ 算法

SoftQ 算法

初始化参数 θ 和 ϕ
复制参数 $\bar{\theta} \leftarrow \theta, \bar{\phi} \leftarrow \phi$
初始化经验回放 D
for 回合数 $= 1, M$ **do**
  for 时步 $= 1, t$ **do**
    根据 $\mathbf{a}_t \leftarrow f^{\phi}(\xi; \mathbf{s}_t)$ 采样动作，其中 $\xi \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
    环境根据 a_t 反馈奖励 s_t 和下一个状态 s_{t+1}
    存储 transition 即 (s_t, a_t, r_t, s_{t+1}) 到经验回放 D 中
    更新环境状态 $s_{t+1} \leftarrow s_t$
    更新 soft Q 函数参数：
    对于每个 $s_{t+1}^{(i)}$ 采样 $\left\{\mathbf{a}^{(i,j)}\right\}_{j=0}^{M} \sim q_{\mathbf{a}'}$
    计算 empirical soft values $V_{\text{soft}}^{\theta}(\mathbf{s}_t)$①
    计算 empirical gradient $J_Q(\theta)$②
    根据 $J_Q(\theta)$ 使用 ADAM 更新参数 θ
    更新策略：
    对于每个 $s_t^{(i)}$ 采样 $\left\{\xi^{(i,j)}\right\}_{j=0}^{M} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
    计算 $\mathbf{a}_t^{(i,j)} = f^{\phi}\left(\xi^{(i,j)}, \mathbf{s}_t^{(i)}\right)$
    使用经验估计计算 $\Delta f^{\phi}(\cdot; \mathbf{s}_t)$③
    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 $\hat{\nabla}_{\phi} J_{\pi}$
    根据 $\hat{\nabla}_{\phi} J_{\pi}$ 使用 ADAM 更新参数 ϕ

  end for
  每 C 个回合复制参数 $\bar{\theta} \leftarrow \theta, \bar{\phi} \leftarrow \phi$
end for

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

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

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

16 SAC-S 算法

SAC-S 算法[①]

初始化参数 $\psi, \bar{\psi}, \theta, \phi$
for 回合数 $= 1, M$ **do**
  for 时步 $= 1, t$ **do**
    根据 $\boldsymbol{a}_t \sim \pi_\phi\left(\boldsymbol{a}_t \mid \mathbf{s}_t\right)$ 采样动作 a_t
    环境反馈奖励和下一个状态，$\mathbf{s}_{t+1} \sim p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right)$
    存储 transition 到经验回放中，$\mathcal{D} \leftarrow \mathcal{D} \cup \left\{\left(\mathbf{s}_t, \mathbf{a}_t, r\left(\mathbf{s}_t, \mathbf{a}_t\right), \mathbf{s}_{t+1}\right)\right\}$
    更新环境状态 $s_{t+1} \leftarrow s_t$
    更新策略：
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q\left(\theta_i\right)$ 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

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

17 SAC 算法

SAC 算法[①]

初始化网络参数 θ_1, θ_2 以及 ϕ
复制参数到目标网络 $\bar{\theta_1} \leftarrow \theta_1, \bar{\theta_2} \leftarrow \theta_2$,
初始化经验回放 D
for 回合数 $= 1, M$ **do**
  重置环境，获得初始状态 s_t
  for 时步 $= 1, t$ **do**
    根据 $\boldsymbol{a}_t \sim \pi_\phi\left(\boldsymbol{a}_t \mid \mathbf{s}_t\right)$ 采样动作 a_t
    环境反馈奖励和下一个状态，$\mathbf{s}_{t+1} \sim p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right)$
    存储 transition 到经验回放中，$\mathcal{D} \leftarrow \mathcal{D} \cup \left\{\left(\mathbf{s}_t, \mathbf{a}_t, r\left(\mathbf{s}_t, \mathbf{a}_t\right), \mathbf{s}_{t+1}\right)\right\}$
    更新环境状态 $s_{t+1} \leftarrow s_t$
    更新策略：
    更新 Q 函数，$\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q\left(\theta_i\right)$ for $i \in \{1, 2\}$[②③]
    更新策略权重，$\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ [④]
    调整 temperature，$\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$ [⑤]
    更新目标网络权重，$\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1, 2\}$
  end for
end for

[②] Soft Actor-Critic Algorithms and Applications

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

18 GAIL 算法

GAIL 算法[①]

采样专家轨迹 $\tau_E \sim \pi_E$，初始化网络模型参数 θ_0 和判别器 D 参数 ω_0
for 回合数 $i = 1, 2, \cdots$ **do**
    采样策略轨迹 $\tau_i \sim \pi_{\theta_i}$
    使用梯度下降更新判别器 D 的参数 ω_i，梯度为：

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

    使用判别器 D 对策略轨迹 τ_i 的输出作为奖励更新策略 π_{θ_i}[②]
end for

[①] Generative Adversarial Imitation Learning

[②] 策略更新方式与策略模型 π_θ 有关，如 PP0-Clip 等.

19 MAPPO 算法

MAPPO 算法[①②]

初始化每个智能体 u 的 Critic 网络 Q_{ϕ^u} 和参数为 θ^u 的 Actor 网络，$u \in U$
初始化每个智能体 u 的目标 Actor 网络 π^u_{old} 的参数 $\theta^u_{old} \leftarrow \theta^u$ 和目标 Critic 网络 $Q_{\overline{\phi}^u}$ 的参数 $\overline{\phi}^u \leftarrow \phi^u$
初始化 epoch 数 K
初始化经验回放 D
for 回合数 $= 1, 2, \cdots, M$ **do**
  初始化状态 s_1
  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 $\tau^u = \{o^u_t, a^u_t, r_{t+1}\}^T_{t=1}$[③]
  对每个时步的每条轨迹
  计算折扣奖励 $\{\hat{R}^u_t\}^T_{t=1}$
  计算优势函数 $\{A^u_t = V^u_{\phi_t} - \hat{R}^u_t\}^T_{t=1}$
  计算 $y^u_t = V^u_{\phi_t} + A^u_t$
  将每个时步的数据 $\{[o^u_t, a^u_t, y^u_t, A^u_t]^U_{u=1}\}^T_{t=1}$ 都存储到经验回放 D 中
  for epoch 数 $k = 1, 2, \cdots, K$ **do**
    打乱 D 中数据顺序并重新编号
    for $j = 0, 1, \cdots, \frac{T}{B} - 1$ **do**
      选择 B 条数据 $\{o^u_i, a^u_i, y^u_i, A^u_i\}^{B(j+1)}_{i=1+Bj}$
      计算梯度：
      $\triangle\theta^u = \frac{1}{B}\sum^B_{i=1}\{\bigtriangledown_{\theta^u} f(r_i(\theta^u), A^u_i)\}$
      $\triangle\phi^u = \frac{1}{B}\sum^B_{i=1}\{\bigtriangledown_{\phi^u}(y^u_i - V_{\phi^u}(o^u_i))^2\}$
      Adam 梯度上升方法计算 θ^u，Adam 梯度下降方法计算 ϕ^u
    end for
  end for
  更新 $\theta^u_{old} \leftarrow \theta^u$, $\overline{\phi}^u \leftarrow \phi^u$
end for

[①] Joint Optimization of Handover Control and Power Allocation Based on Multi-Agent Deep Reinforcement Learning

[②] The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games

[③] o^u_t 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

[④] 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