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

```
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:         $V(s) \leftarrow \sum_{s',r} p\left(s', r \mid s, \pi(s)\right)\left[r + \gamma V\left(s'\right)\right]$
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：$\pi(a|s) \leftarrow \arg\max_a \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma V\left(s'\right)\right]$
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:       $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma V(s')]$
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$\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma V(s')]$
```

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$\tau = \{s_0, a_0, r_1, \cdots, s_{T-1}, a_{T-1}, r_T,\}$
4:     初始化回报 $G \leftarrow 0$
5:     **for** 时步 $t = T-1,, T-2, \cdots, 0$ **do**
6:         $G \leftarrow \gamma G + R_{t+1}$
7:         **repeat**
8:             将 $G$ 添加到 $Returns(s_t)$
9:             $V(S_t) \leftarrow \text{average}(\text{Returns}(S_t))$
10:        **until** $s_t$ 第二次出现，即与历史某个状态$s_0, \cdots, s_{t-1}$相同
11:    **end for**
12: **end for**
```

[1]脚注

5 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$Q(s_{terminal},)=0$，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
```

[1]Reinforcement Learning: An Introduction

6 Sarsa算法

Sarsa算法[1]

```
1: 初始化Q表$Q(s,a)$为任意值，但其中$Q(s_{terminal},)=0$，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
```

[1]Reinforcement Learning: An Introduction

7 DQN算法

DQN算法

```
1: 初始化当前网络参数 $\theta$ 和目标网络参数 $\hat{\theta} \leftarrow \theta$
2: 初始化经验回放$D$
3: for 回合数 $m = 1, 2, \cdots, M$ do
4:     重置环境，获得初始状态$s_0$
5:     for 时步 $t = 1, 2, \cdots, T$ do
6:         交互采样：
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:         存储样本$(s_t, a_t, r_t, s_{t+1})$到经验回放$D$中
10:        更新环境状态$s_{t+1} \leftarrow s_t$
11:        策略更新：
12:        从$D$中随机采样一个批量的样本
13:        计算$Q$的期望值，即$y_i = r_t + \gamma \max_{a_{i+1}} Q(s_{i+1}, a; \hat{\theta})$
14:        计算损失$L(\theta) = (y_i - Q(s_i, a_i; \theta))^2$，并关于参数$\theta$做随机梯度下降
15:        每$C$步复制参数到目标网络$\hat{\theta} \leftarrow \theta$
16:     end for
17: end for
```

8 DRQN算法

DRQN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
           $B = \left\{(s_j, a_j, r_j, s_j^{'}) \ldots (s_{j+\tau}, a_{j+\tau}, r_{j+\tau}, s_{j+\tau}^{'})\right\}_{j=1}^{\text{batch size}} \subseteq D$
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
```

[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: 初始化当前网络参数 $\theta$
2: 复制参数到目标网络$\hat{\theta} \leftarrow \theta$
3: 初始化经验回放$D$
4: for 回合数 $m = 1, 2, \cdots, M$ do
5:     重置环境，获得初始状态$s_0$
6:     for 时步 $t = 1, 2, \cdots, T$ do
7:         交互采样：
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$s_{t+1}$
10:        存储样本$(s_t, a_t, r_t, s_{t+1})$到经验回放$D$中，并根据$TD$误差损失确定其优先级$p_t$
11:        更新环境状态$s_{t+1} \leftarrow s_t$
12:        模型更新：
13:        根据每个样本的优先级计算采样概率$P(j) = p_j^\alpha / \sum_i p_i^\alpha$，从$D$中采样一个批量的样本
14:        计算各个样本重要性采样权重 $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
15:        计算$TD$误差$\delta_j$；并根据$TD$误差更新优先级$p_j$
16:        计算$Q$的估计值，即$y_j$
17:        根据重要性采样权重调整损失 $L(\theta) = \left(y_j - Q\left(s_j, a_j; \theta\right) \cdot w_j\right)^2$，并关于$\theta$做随机梯度下降
18:        每$C$步复制参数$\hat{Q} \leftarrow Q$
19:    end for
20: end for
```

10 Policy Gradient算法

REINFORCE算法：Monte-Carlo Policy Gradient[1]

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

[1]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$\theta \leftarrow \theta + \alpha_\theta Q_w(s, a) \nabla_\theta \log \pi_\theta(a \mid s)$
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

12 PPO-Clip算法

PPO-Clip算法

```
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 $= 1, 2, \cdots, M$ do
6:     使 用 策 略$\pi_\theta$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
7:     for epoch数 $k = 1, 2, \cdots, K$ 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:        $L^{CLIP}(\theta) = \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} min(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)))$
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
```

13 PPO-KL散度算法

PPO-KL散度算法[①][②]

```
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 $= 1, 2, \cdots, M$ 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 $= 1, 2, \cdots, K$ 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
```

[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$\theta^Q$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 $D$
4: for 回合数 $= 1, M$ do
5:     交互采样：
6:     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
7:     环境根据$a_t$反馈奖励$s_t$和下一个状态$s_{t+1}$
8:     存储样本$(s_t, a_t, r_t, s_{t+1})$到经验回放 $D$ 中
9:     更新环境状态$s_{t+1} \leftarrow s_t$
10:    策略更新：
11:    从 $D$ 中取出一个随机批量的$(s_i, a_i, r_i, s_{i+1})$
12:    求得$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1} \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right)$
13:    更新 critic 参数，其损失为：$L = \frac{1}{N}\sum_i \left(y_i - Q\left(s_i, a_i \mid \theta^Q\right)\right)^2$
14:    更新 actor 参数：$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q\left(s, a \mid \theta^Q\right)\Big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu\left(s \mid \theta^\mu\right)\Big|_{s_i}$
15:    软更新目标网络：$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$， $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
16: end for
```

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$\xi \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
 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$\left\{\mathbf{a}^{(i,j)}\right\}_{j=0}^{M} \sim q_{\mathbf{a}'}$
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$\theta$
15:         更新策略：
16:         对于每个$s_t^{(i)}$采样$\left\{\xi^{(i,j)}\right\}_{j=0}^{M} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
17:         计算$\mathbf{a}_t^{(i,j)} = f^{\phi}\left(\xi^{(i,j)}, \mathbf{s}_t^{(i)}\right)$
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$\phi$
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
```

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

16 SAC-S算法

SAC-S算法

```
 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$\boldsymbol{a}_t \sim \pi_\phi\left(\boldsymbol{a}_t \mid \mathbf{s}_t\right)$采样动作$a_t$
 5:         环境反馈奖励和下一个状态，$\mathbf{s}_{t+1} \sim p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right)$
 6:         存储transition到经验回放中，$\mathcal{D} \leftarrow \mathcal{D} \cup\left\{\left(\mathbf{s}_t, \mathbf{a}_t, r\left(\mathbf{s}_t, \mathbf{a}_t\right), \mathbf{s}_{t+1}\right)\right\}$
 7:         更新环境状态$s_{t+1} \leftarrow s_t$
 8:         更新策略：
 9:         $\psi \leftarrow \psi-\lambda_V \hat{\nabla}_\psi J_V(\psi)$
10:         $\theta_i \leftarrow \theta_i-\lambda_Q \hat{\nabla}_{\theta_i} J_Q\left(\theta_i\right)$ 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
```

[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$\theta_1, \theta_2$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$\boldsymbol{a}_t \sim \pi_\phi\left(\boldsymbol{a}_t \mid \mathbf{s}_t\right)$采样动作$a_t$
8:         环境反馈奖励和下一个状态，$\mathbf{s}_{t+1} \sim p\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right)$
9:         存储transition到经验回放中，$\mathcal{D} \leftarrow \mathcal{D} \cup\left\{\left(\mathbf{s}_t, \mathbf{a}_t, r\left(\mathbf{s}_t, \mathbf{a}_t\right), \mathbf{s}_{t+1}\right)\right\}$
10:        更新环境状态$s_{t+1} \leftarrow s_t$
11:        更新策略：
12:        更新$Q$函数，$\theta_i \leftarrow \theta_i-\lambda_Q \hat{\nabla}_{\theta_i} J_Q\left(\theta_i\right)$ for $i \in\{1,2\}$
13:        更新策略权重，$\phi \leftarrow \phi-\lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
14:        调整温度因子，$\alpha \leftarrow \alpha-\lambda \hat{\nabla}_\alpha J(\alpha)$
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
```

18 GAIL算法

GAIL算法[①]

```
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 $i = 1, 2, \cdots$ 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：
           $\hat{\mathbb{E}}_{\tau_i}\left[\nabla_w \log\left(D_w(s,a)\right)\right] + \hat{\mathbb{E}}_{\tau_E}\left[\nabla_w \log\left(1 - D_w(s,a)\right)\right]$    (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$\pi_{\theta_i}$[②]
6: 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

```
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$\pi^u_{old}$的参数$\theta^u_{old} \leftarrow \theta^u$和目标Critic网络$Q_{\overline{\phi}^u}$的参数$\overline{\phi}^u \leftarrow \phi^u$
3: 初始化epoch数$K$
4: 初始化经验回放$D$
5: for 回合数 $= 1, 2, \cdots, M$ 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$\tau^u = \{o^u_t, a^u_t, r_{t+1}\}^T_{t=1}$
8:   对每个时步的每条轨迹
9:   计算折扣奖励$\{\hat{R}^u_t\}^T_{t=1}$
10:   计算优势函数$\{A^u_t = V^u_{\phi_t} - \hat{R}^u_t\}^T_{t=1}$
11:   计算$y^u_t = V^u_{\phi_t} + A^u_t$
12:   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
13:   for epoch数 $k = 1, 2, \cdots, K$ do
14:     打乱$D$中数据顺序并重新编号
15:     for $j = 0, 1, \cdots, \frac{T}{B} - 1$ do
16:       选择$B$条数据$\{o^u_i, a^u_i, y^u_i, A^u_i\}^{B(j+1)}_{i=1+Bj}$
17:       计算梯度：
18:       $\triangle\theta^u = \frac{1}{B}\sum^B_{i=1}\{\nabla_{\theta^u} f(r_i(\theta^u), A^u_i)\}$
19:       $\triangle\phi^u = \frac{1}{B}\sum^B_{i=1}\{\nabla_{\phi^u}(y^u_i - V_{\phi^u}(o^u_i))^2\}$
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$\theta^u_{old} \leftarrow \theta^u$, $\overline{\phi}^u \leftarrow \phi^u$
24: end for
```