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$Q(s_{terminal},) = 0$，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

3   Sarsa算法

Sarsa算法[1]

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

4   DQN算法

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**

---

[1]Playing Atari with Deep Reinforcement Learning

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

[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   DRQN算法

DRQN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θ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]$\theta_i \leftarrow \theta_i - \lambda \nabla_{\theta_i} L_i(\theta_i)$

6 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$\theta$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**

[①]Playing Atari with Deep Reinforcement Learning

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

[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

7 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 $\boldsymbol{0}$)
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$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi (A_t \mid S_t, \boldsymbol{\theta})$
7:   **end for**
8: **end for**

---

[1]Reinforcement Learning: An Introduction

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

9   PPO-Clip算法

PPO-Clip算法[①][②]

1: 初始化策略网络(Actor)参数θ和价值网络(Critic)参数ϕ

2: 初始化Clip参数ϵ

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:    根 据 策 略π_θ采 样C个 时 步 数 据,收 集 轨 迹τ $=$ $s_0, a_0, r_1, ..., s_t, a_t, r_{t+1}, \cdots$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:       $L^{CLIP}(\theta) = \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} min(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)))$[③]

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**

[①]Proximal Policy Optimization Algorithms

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

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

10　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θ和价值网络(Critic)参数ϕ
2: 初始化KL散度参数λ
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
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_{Φ_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θ:
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

11   DDPG算法

DDPG算法[①]

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R

4: **for** 回合数 $= 1, M$ **do**

5:     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

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$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1} \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right)$

12:     更新critic参数，其损失为：$L = \frac{1}{N} \sum_i \left(y_i - Q\left(s_i, a_i \mid \theta^Q\right)\right)^2$

13:     更新actor参数：$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q\left(s, a \mid \theta^Q\right)\big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu\left(s \mid \theta^\mu\right)\big|_{s_i}$

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**

[①]Continuous control with deep reinforcement learning

12   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$\xi \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
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θ
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**

---

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

13 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$\boldsymbol{a}_t \sim \pi_\phi(\boldsymbol{a}_t \mid \mathbf{s}_t)$采样动作$a_t$
5:       环境反馈奖励和下一个状态，$\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$
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$s_{t+1} \leftarrow s_t$
8:       **更新策略：**
9:       $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
10:      $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ 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**

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

14 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$\boldsymbol{a}_t \sim \pi_\phi\left(\boldsymbol{a}_t \mid \mathbf{s}_t\right)$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 $i \in \{1,2\}$[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**

[②]Soft Actor-Critic Algorithms and Applications

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

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

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

[⑤]$J(\alpha) = \mathbb{E}_{\mathbf{a}_t \sim \pi_t}\left[-\alpha \log \pi_t\left(\mathbf{a}_t \mid \mathbf{s}_t\right) - \alpha\overline{\mathcal{H}}\right]$