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]

```
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ε − 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) ← Q(s_t, a_t) + α[r_t + γ max_a Q(s_{t+1}, a) − Q(s_t, a_t)]
        更新状态s_{t+1} ← s_t
    end for
end for
```

[1] Reinforcement Learning: An Introduction

3 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

4 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 $L(\theta)=\left(y_i-Q\left(s_i, a_i ; \theta\right)\right)^2$关于参数$\theta$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]$y_i=\begin{cases} r_i & \text{对于终止状态} s_{i+1} \\ r_i+\gamma \max_{a^{\prime}} Q\left(s_{i+1}, a^{\prime} ; \theta\right) & \text{对于非终止状态} s_{i+1} \end{cases}$

[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

5 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_j②
      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

①Deep 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]

```
初始化策略网络参数θ
复制参数到目标网络Q̂ ← 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ε − 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} ← 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^α / Σ_i p_i^α，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 · P(j))^{−β} / 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(θ) = (y_j − Q(s_j, a_j; θ) · w_j)^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Q̂ ← 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

7 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

8 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

9 PPO-Clip算法

PPO-Clip算法[1][2]

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

10 PPO-KL散度算法

PPO-KL散度算法[1][2]

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

11 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$\theta^{\mu}$
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

12 SoftQ算法

SoftQ算法

```
初始化参数θ和φ
复制参数θ̄ ← θ, φ̄ ← φ
初始化经验回放D
for 回合数 = 1, M do
    for 时步 = 1, t do
        根据a_t ← f^φ(ξ; s_t)采样动作，其中ξ ~ N(0, 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} ← s_t
        更新soft Q函数参数：
        对于每个s_{t+1}^{(i)}采样{a^{(i,j)}}_{j=0}^{M} ~ q_{a'}
        计算empirical soft values V_soft^θ(s_t)①
        计算empirical gradient J_Q(θ)②
        根据J_Q(θ)使用ADAM更新参数θ
        更新策略：
        对于每个s_t^{(i)}采样{ξ^{(i,j)}}_{j=0}^{M} ~ N(0, I)
        计算a_t^{(i,j)} = f^φ(ξ^{(i,j)}, s_t^{(i)})
        使用经验估计计算Δf^φ(·; s_t)③
        计算经验估计∂J_π(φ; s_t)/∂φ ∝ E_ξ[Δf^φ(ξ; s_t) ∂f^φ(ξ; s_t)/∂φ]，即∇̂_φ J_π
        根据∇̂_φ J_π使用ADAM更新参数φ

    end for
    每C个回合复制参数θ̄ ← θ, φ̄ ← φ
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
$$\begin{aligned}\Delta f^{\phi}(\cdot;\mathbf{s}_t)=&\mathbb{E}_{\mathbf{a}_t\sim\pi^{\phi}}\left[\kappa\left(\mathbf{a}_t,f^{\phi}(\cdot;\mathbf{s}_t)\right)\nabla_{\mathbf{a}'}Q_{\text{soft}}^{\theta}(\mathbf{s}_t,\mathbf{a}')\Big|_{\mathbf{a}'=\mathbf{a}_t}\right.\\&\left.+\alpha\nabla_{\mathbf{a}'}\kappa\left(\mathbf{a}',f^{\phi}(\cdot;\mathbf{s}_t)\right)\Big|_{\mathbf{a}'=\mathbf{a}_t}\right]\end{aligned}$$

13 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

14 SAC算法

SAC算法[1]

```
初始化网络参数θ1, θ2以及φ
复制参数到目标网络θ̄1 ← θ1, θ̄2 ← θ2,
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a_t ~ π_φ(a_t | s_t)采样动作a_t
        环境反馈奖励和下一个状态，s_{t+1} ~ p(s_{t+1} | s_t, a_t)
        存储transition到经验回放中，D ← D ∪ {(s_t, a_t, r(s_t, a_t), s_{t+1})}
        更新环境状态s_{t+1} ← s_t
        更新策略：
        更新Q函数，θ_i ← θ_i − λ_Q ∇̂_{θ_i} J_Q(θ_i) for i ∈ {1,2}[2][3]
        更新策略权重，φ ← φ − λ_π ∇̂_φ J_π(φ) [4]
        调整temperature，α ← α − λ ∇̂_α J(α) [5]
        更新目标网络权重，θ̄_i ← τθ_i + (1 − τ)θ̄_i for i ∈ {1,2}
    end for
end for
```

[2] Soft Actor-Critic Algorithms and Applications

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

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

15 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θ_0和判别器D参数ω_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：

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

5: 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]
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.