深度确定策略梯度 (Deep Deterministic Policy Gradient, DDPG)

1. 策略网络 $\pi(s;\theta)$；价值网络 $q(s,a;\omega)$；target network $\pi(s;\theta^-)$，$q(s,a;\omega^-)$；其中 $\theta^-=\theta$，$\omega^-=\omega$
2. 根据当前状态 s_t 执行动作 $a_t=\pi(s_t;\theta_t)$，得到奖励 r_t 和下一个状态 s_{t+1}
3. 计算独立策略梯度 $g=\frac{\partial J}{\partial\theta}=\frac{\partial q(s,a;\omega)}{\partial a}\cdot\frac{\partial\pi(s;\theta)}{\partial\theta}\Big|_{\theta=\theta_t}=\frac{\partial q_t(a_t;\omega)}{\partial a_t}\cdot\frac{\partial\pi_t(s_t;\theta)}{\partial\theta_t}\Big|_{\theta=\theta_t}$
4. 使用策略梯度上升更新策略网络 $\theta_{t+1}\leftarrow\theta_t+\beta\cdot g$
5. 使用时值网络计算 $q_t=q(s_t,a_t;\omega_t)$
6. 使用 target network 计算 $q_{t+1}=q(s_{t+1},a_{t+1};\omega_t^-)$，其中 $a_{t+1}^-=\pi(s_{t+1};\theta_t^-)$
7. 计算 TD target $y_t=r_t+\gamma\cdot q_{t+1}$
8. 计算 TD error $\delta_t=q_t-y_t$
9. 使用梯度下降更新价值网络 $\omega_{t+1}\leftarrow\omega_t-\alpha\cdot\delta_t\cdot\frac{\partial q(s_t,a_t;\omega)}{\partial\omega}\Big|_{\omega=\omega_t}$

每 N_{update} 个 step 后更新 target network 的参数：

- $\theta^-\leftarrow\tau\cdot\theta+(1-\tau)\cdot\theta^-$
- $\omega^-\leftarrow\tau\cdot\omega+(1-\tau)\cdot\omega^-$

确定策略梯度 $\frac{\partial g(\pi(s;\theta);\omega)}{\partial\theta}=\frac{\partial g}{\partial a}\cdot\frac{\partial g(\pi;\omega)}{\partial\pi}$

计算确定策略梯度

$\theta_{t+1}\leftarrow\theta_t+\beta\cdot\frac{\partial g(\pi(s;\theta);\omega)}{\partial\theta}$

使用确定策略梯度上升
$q(s,a;\omega)\to q(s,\pi(s;\theta);\omega)$
即：最大化动作的评分

解决 DDPG 的高估问题

Clipped Double-Q Learning

- 使用两个价值网络 $q(s,a;\omega_1)$ 和 $q(s,a;\omega_2)$
- 对应两个 target network $q(s,a;\omega_1^-)$ 和 $q(s,a;\omega_2^-)$
- 在计算 TD target 时，q_{t+1} 的值选取两个 target network 输出中较小的值 $q_{t+1}=\min_{i=1,2}q(s_{t+1},a_{t+1}^-;\omega_i^-)$

"Delayed" Policy Updates

- 策略网络的更新频率比价值网络低
- 每更新两次价值网络之后，更新一次策略网络
- 每更新策略网络和价值网络 target network 之后，更新一次策略网络的 target network

Target Policy Smoothing

- 在计算 q_{t+1} 时所用的 target action a_{t+1}^- 上添加截断的高斯噪声 $a_{t+1}^-=\pi(s_{t+1};\theta_t^-)+\text{clip}(\epsilon,-c,c)$，其中 $\epsilon\sim N(0,\sigma^2)$
- 通过给 a_{t+1}^- 添加噪声，可以避免动作值网络估计误差导致的过度自信的问题
- 如果价值网络在更新过程中对某个动作的评分过高，那么策略网络会误以为该动作最好，这时策略网络输出的动作选项就算再以最大化动作评分为目标，从而规避该问题。

孪生延迟深度确定性策略梯度 (Twin Delayed DDPG, TD3)

初始化策略网络 $\pi(s;\theta)$；两个价值网络 $q(s,a;\omega_1)$，$q(s,a;\omega_2)$
三个网络对应的 target network $\pi(s;\theta^-)$，$q(s,a;\omega_1^-)$，$q(s,a;\omega_2^-)$；其中 $\theta^-=\theta$，$\omega_1^-=\omega_1$，$\omega_2^-=\omega_2$
设置策略网络，所有 target network 的更新频率为 N_{update}

1. 根据当前状态 s_t 执行动作 $a_t=\pi(s_t;\theta_t)$，得到奖励 r_t 和下一个状态 s_{t+1}
2. 选择 target action $a_{t+1}^-=\pi(s_{t+1};\theta_t^-)+\text{clip}(\epsilon,-c,c)$，其中 $\epsilon\sim N(0,\sigma^2)$
3. 使用 target network 计算 $q_{t+1}=\min_{i=1,2}q(s_{t+1},a_{t+1}^-;\omega_i^-)$
4. 计算 TD target $y_t=r_t+\gamma\cdot q_{t+1}$
5. 分别计算两个价值网络的 TD error $\delta_{i,t}=q(s_t,a_t;\omega_{i,t})-y_t$，其中 $i=1,2$
6. 使用梯度下降更新两个价值网络 $\omega_{i,t+1}\leftarrow\omega_{i,t}-\alpha\cdot\delta_{i,t}\cdot\frac{\partial q(s_t,a_t;\omega)}{\partial\omega}\Big|_{\omega=\omega_{i,t}}$，其中 $i=1,2$
7. 如果当前 step 计数 $\text{%}N_{update}=0$
   1. 使用第一个价值网络计算确定策略梯度 $g=\frac{\partial g}{\partial a}\cdot\frac{\partial q(s,a;\omega_1)}{\partial a}\cdot\frac{\partial\pi(s;\theta)}{\partial\theta}\Big|_{\theta=\theta_t}=\frac{\partial q_t(s_t,a_t;\omega_1)}{\partial a_t}\cdot\frac{\partial\pi_t(s_t;\theta_t)}{\partial\theta_t}\Big|_{\theta=\theta_t}$
   2. 使用策略梯度上升更新策略网络 $\theta_{t+1}\leftarrow\theta_t+\beta\cdot g$
   3. 更新所有 target network 参数
      - $\theta^-\leftarrow\tau\cdot\theta+(1-\tau)\cdot\theta^-$
      - $\omega_i^-\leftarrow\tau\cdot\omega_i+(1-\tau)\cdot\omega_i^-$，其中 $i=1,2$

实现算法时应当先判断，更新策略网络，再更新价值网络，因为更新策略网络需要用到价值网络更新后的参数 $\omega_{1,t}$

深度强化学习

术语	未来观测时使用随机变量
状态 $s\in S$	状态 S
动作 $a\in A$	动作 A
策略 $\pi(a\|s)$	
奖励 r	奖励 R

奖励之间的关系：折扣回报 $U_t=R_t+\gamma R_{t+1}+\gamma^2R_{t+2}+\gamma^3R_{t+3}+\cdots$，$\gamma$：折扣率

动作的概率密度函数（策略函数）$P\{A=a|S=s\}=\pi(a|s)$

状态的概率密度函数（状态转移函数）$P\{S'=s'|S=s,A=a\}=p(s'|s,a)$

策略学习

奖励 R_t 与当前状态 S_t 和做出的动作 A_t 有关

使用神经网络 $\pi(a|s;\theta)$ 近似策略函数 $\pi(a|s)$

指导 $\pi(a|s;\theta)$ 的训练

$V_\pi(s_t)=\mathbb{E}_A\left[Q_\pi(s_t,A)\right]=\sum_a\pi(a|s_t)\cdot Q_\pi(s_t,a)$

在 t 时刻观测到状态 s_t 并做出动作 a_t
对未来的状态和动作 $S_{t+1},S_{t+2},\cdots,A_{t+1},A_{t+2},\cdots$ 求期望

动作价值函数 $Q_\pi(s_t,a_t)=\mathbb{E}_{S_{t+1},S_{t+2},\cdots,A_{t+1},A_{t+2},\cdots}[U_t|S_t=s_t,A_t=a_t]$ —对动作 A 求期望→ 状态价值函数 $V_\pi(s_t)=\mathbb{E}_A\left[Q_\pi(s_t,A)\right]$ 表示当前状态 s_t 的评分 —对状态 S 求期望→ $\mathbb{E}_S[V_\pi(S)]$ 可以评估策略 π 的好坏

推导 A2C

$Q_\pi(s_t,a_t)=\mathbb{E}_{S_{t+1},A_{t+1}}[R_t+\gamma\cdot Q_\pi(S_{t+1},A_{t+1})|S_t=s_t,A_t=a_t]$
使用了递归定义：$U_t=R_t+\gamma\cdot U_{t+1}$ 然后等式两边求期望

R_t 与 A_{t+1} 无关，对 A_{t+1} 求期望可以放到里面
省略 $[S_t=s_t,A_t=a_t]$

$Q_\pi(s_t,a_t)=\mathbb{E}_{S_{t+1}}\{R_t+\gamma\cdot\mathbb{E}_{A_{t+1}}[Q_\pi(S_{t+1},A_{t+1})]\}$

$V_\pi(S_{t+1})=\mathbb{E}_{A_{t+1}}[Q_\pi(S_{t+1},A_{t+1})]$

$Q_\pi(s_t,a_t)=\mathbb{E}_{S_{t+1}}[R_t+\gamma\cdot V_\pi(S_{t+1})]$ —根据轨迹 (s_t,a_t,r_t,s_{t+1}) 使用蒙特卡洛近似→ $Q_\pi(s_t,a_t)\approx r_t+\gamma\cdot V_\pi(s_{t+1})$ —用于近似 $Q_\pi(s_t,a_t)$

$V_\pi(s_t)=\mathbb{E}_A[Q_\pi(s_t,A)]=\mathbb{E}_A\left[\mathbb{E}_{S_{t+1}}[R_t+\gamma\cdot V_\pi(S_{t+1})]\right]=\mathbb{E}_{A,S_{t+1}}[R_t+\gamma\cdot V_\pi(S_{t+1})]$ —根据轨迹 (s_t,a_t,r_t,s_{t+1}) 使用蒙特卡洛近似→ $V_\pi(s_t)\approx r_t+\gamma\cdot V_\pi(s_{t+1})$ —用于 TD 算法更新 $V_\pi(s_t)$

$a^*=\underset{a}{\operatorname{argmax}}\,Q^*(s,a)$ ← 使用 Q^* 决策 ← 最优动作价值函数 $Q^*(s_t,a_t)=\max_\pi Q_\pi(s_t,a_t)$ 表示状态 s_t 下做出动作 a_t 所能达到的最高分数，与策略 π 无关

最优状态价值函数 $V^*(s_t)=\max_\pi V_\pi(s_t)$ 表示状态 s_t 的好坏程度，与策略 π 无关

使用神经网络 $Q(s,a;\omega)$ 近似 $Q^*(s,a)$

价值学习

最优优势函数 $A^*(s_t,a_t)=Q^*(s_t,a_t)-V^*(s_t)$ 表示动作 a_t 相对于 baseline $V^*(s_t)$ 的优势大小

等式变形

$Q^*(s,a)=V^*(s)+A^*(s,a)-\max_a A^*(s,a)$

对动作 a_t 求最大值

$\max_a A^*(s,a)=\max_a Q^*(s,a)-V^*(s)$

使用已知结论 $V^*(s)=\max_a Q^*(s,a)$

减0

$\max_a A^*(s,a)=0$

记 $Q_\pi(s,a)-V_\pi(s)=A_\pi(s,a)$ 为优势函数 (Advantage Function)
优势函数可以指出当前动作 a_t 所获得的回报 $Q_\pi(s_t,a_t)$ 与当前状态 s_t 所期待的回报 $V_\pi(s_t)$ 之间的差距
优势函数大于0说明当前动作比预期的效果好；优势函数小于0说明当前动作比预期的效果差

Deep Q-Network (DQN)

1. 观察状态 $S_t=s_t$ 并作出动作 $a_t=\underset{a}{\operatorname{argmax}}\,Q(s_t,a)$
2. 计算价值 $q_t=Q(s_t,a_t;\omega)$
3. 环境给出奖励 r_t 和新的状态 s_{t+1}
4. 计算 TD target 为 $y_t=r_t+\gamma\cdot Q(s_{t+1},a_{t+1};\omega_t)=r_t+\gamma\cdot\max_a Q(s_{t+1},a;\omega_t)$
5. 计算 TD error 为 $\delta_t=q_t-y_t$
6. 最小化 TD error，定义损失函数 $L_t=\frac{1}{2}\delta_t^2$
7. 使用梯度下降最小化损失函数 $\omega_{t+1}\leftarrow\omega_t-\alpha\cdot\frac{\partial L_t}{\partial\omega}\Big|_{\omega=\omega_t}$

其中 $\frac{\partial L_t}{\partial\omega}=(q_t-y_t)\cdot\frac{\partial Q(s_t,a_t;\omega)}{\partial\omega}\Big|_{\omega=\omega_t}$

DQN高级技巧1：Experience Replay

- 将轨迹 (s_t,a_t,r_t,s_{t+1}) 存入 Replay Buffer 中
- Replay Buffer 中储存最新的 $N(10^5\sim10^6)$ 条轨迹
- 从 Replay Buffer 中随机抽样 T (batch size) 条轨迹
- 使用所有轨迹的均方误差（TD error 平方的平均值）作为损失函数：$L=\frac{1}{T}\sum_{t=1}^{T}\frac{1}{2}\delta_t^2$

提高轨迹的利用率；消除轨迹之间的相关性

DQN高级技巧2：target network

- 定义 target network $Q(s,a;\omega^-)$
- 初始时 DQN 的网络参数跟随 target network：$\omega^-\leftarrow\omega$
- 每 N_{update} 个 step 后更新 target network 的参数
  - 直接使用 DQN 的参数：$\omega^-\leftarrow\omega$
  - 使用 DQN 和 target network 的加权平均：$\omega^-\leftarrow\tau\cdot\omega+(1-\tau)\cdot\omega^-$
- 在计算 TD target 时使用 target network：$y_t=r_t+\gamma\cdot\max_a Q(s_{t+1},a;\omega_t^-)$

DQN高级技巧3：Double DQN
在计算 TD target 时

- 先使用 DQN 选择动作 $a^*=\underset{a}{\operatorname{argmax}}\,Q(s_{t+1},a;\omega_t)$
- 再使用 target network 计算 TD target $y_t=r_t+\gamma\cdot Q(s_{t+1},a^*;\omega_t^-)$

缓解自举（bootstrap）更新网络参数导致的高估问题

Dueling Network

- 使用神经网络 $A(s,a;\omega^A)$ 近似优势函数 $A^*(s,a)$，其结构与 DQN 相同，输出所有动作的评分
- 使用神经网络 $V(s;\omega^V)$ 近似状态价值函数 $V^*(s)$，输出一个标量
- 构建 Dueling Network $Q(s,a;\omega^A,\omega^V)=V(s;\omega^V)+A(s,a;\omega^A)-\max_a A(s,a;\omega^A)$
  - Q 输出的每一个分量等于 A 输出的各个分量加上 V 的值，再减去 A 输出的最大值
  - A 和 V 的参数需要同时训练

Dueling Network 的训练方法与 DQN 完全相同，都是使用 TD 算法、经验回放、target network、Double DQN 这些高级技巧同样适用于 Dueling Network。

$Q(s,a;\omega^A,\omega^V)=V(s;\omega^V)+A(s,a;\omega^A)-\text{mean}_a A(s,a;\omega^A)$ 减去 A 输出的平均值比减去 A 输出的最大值效果更好

假设动作空间维度为 d
$\mathbf{a}=[a_1,a_2,\ldots,a_d]$
$a_i\sim N(\mu_i,\sigma_i^2),\ i=1,2,\ldots,d$

—动作 a 相互独立→ $\pi(\mathbf{a}|s)=\prod_{i=1}^{d}\frac{1}{\sqrt{2\pi\sigma_i}}\cdot\exp\left(-\frac{(a_i-\mu_i)^2}{2\sigma_i^2}\right)$ —为了求策略梯度对策略函数取对数→ $\ln\pi(\mathbf{a}|s)=\sum_{i=1}^{d}\left[-\ln\sigma_i-\frac{(a_i-\mu_i)^2}{2\sigma_i^2}\right]+\text{const}$ —令 $\rho_i=\ln\sigma_i^2,\ i=1,2,\ldots,d$→ $\ln\pi(\mathbf{a}|s)=\sum_{i=1}^{d}\left[-\frac{\rho_i}{2}-\frac{(a_i-\mu_i)^2}{2\cdot\exp(\rho_i)}\right]+\text{const}$ —定义辅助函数 $f(s,\mathbf{a})=\sum_{i=1}^{d}\left[-\frac{\rho_i}{2}-\frac{(a_i-\mu_i)^2}{2\cdot\exp(\rho_i)}\right]$→ $\ln\pi(\mathbf{a}|s)=f(s,\mathbf{a})+\text{const}$

计算策略梯度

随机高斯策略网络

使用神经网络 $\mu(s;\theta^\mu)$ 输出给定状态下各个动作分量的均值
使用神经网络 $\rho(s;\theta^\rho)$ 输出给定状态下各个动作分量的方差对数
记两个网络的参数为 $\theta=(\theta^\mu,\theta^\rho)$ 则 $\ln\pi(\mathbf{a}|s;\theta)=f(s,\mathbf{a};\theta)+\text{const}$
等式两边分别对 θ 求偏导，可得策略梯度中对策略函数的网络参数的偏导等于辅助函数对网络函数的偏导
$\frac{\partial\ln\pi(\mathbf{a}|s;\theta)}{\partial\theta}=\frac{\partial f(s,\mathbf{a};\theta)}{\partial\theta}$

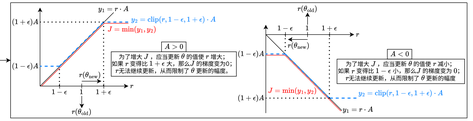
随机策略梯度 $g(\mathbf{a},\theta)=\frac{\partial\ln\pi(\mathbf{a}|s;\theta)}{\partial\theta}\cdot Q_\pi(s,\mathbf{a})=\frac{\partial f(s,\mathbf{a};\theta)}{\partial\theta}\cdot Q_\pi(s,\mathbf{a})$

- 使用 REINFORCE 更新参数 θ
- 使用 Actor-Critic 更新参数 θ

带基线的策略梯度 $g(\mathbf{a},\theta)=\frac{\partial\ln\pi(\mathbf{a}|s;\theta)}{\partial\theta}\cdot Q_\pi(s,\mathbf{a})=\frac{\partial f(s,\mathbf{a};\theta)}{\partial\theta}\cdot[Q_\pi(s,\mathbf{a})-V_\pi(s)]$

- 使用 REINFORCE with baseline 更新参数 θ
- 使用 A2C 更新参数 θ

$J=\min[r\cdot A,h(\epsilon,A)],\quad\text{where }h(\epsilon,A)=\begin{cases}(1+\epsilon)\cdot A, & A\geq0;\\(1-\epsilon)\cdot A, & A<0.\end{cases}$



等价形式

使用 REINFORCE with baseline 或 A2C 的方法估计完优势函数 $A_\pi(s,a)\approx A$ 后
目标函数 J 的值只与 r 有关 $J=\min[r\cdot A,\text{clip}(r,1-\epsilon,1+\epsilon)\cdot A]$

算法解读

PPO 最大化目标函数
$J(\theta)=\mathbb{E}_{S,A}\left\{\min[r(\theta)\cdot A_\pi(S,A),\text{clip}(r(\theta),1-\epsilon,1+\epsilon)\cdot A_\pi(S,A)]\right\}$

令 $r(\theta)=\frac{\pi(A|S;\theta)}{\pi(A|S;\theta_{old})}$，$\theta$ 离 θ_{old} 越远，$r(\theta)$ 越远离 1 $r(\theta_{old})=1$
鼓励动作值函数 $Q_\pi(S,A)$ 替换为优势函数 $A_\pi(S,A)$

$J(\theta)=\mathbb{E}_S\left\{\mathbb{E}_A\left[\frac{\pi(A|S;\theta)}{\pi(A|S;\theta_{old})}\cdot Q_\pi(S,A)\right]\right\}=\mathbb{E}_{S,A}\left[\frac{\pi(A|S;\theta)}{\pi(A|S;\theta_{old})}\cdot Q_\pi(S,A)\right]$
其中 $A\sim\pi(\cdot|s;\theta_{old})$

代入 $J(\theta)$

$V(s;\theta)=\sum_a\pi(a|s;\theta)\cdot Q_\pi(s,a)$
$=\sum_a\pi(a|s;\theta_{old})\cdot\frac{\pi(a|s;\theta)}{\pi(a|s;\theta_{old})}\cdot Q_\pi(s,a)$
$=\mathbb{E}_{A\sim\pi(\cdot|s;\theta_{old})}\left[\frac{\pi(A|s;\theta)}{\pi(A|s;\theta_{old})}\cdot Q_\pi(s,A)\right]$

使用置信域 (Trust Region) 最大化 $J(\theta)$

更新策略网络参数 θ 最大化 $J(\theta)=\mathbb{E}_S[V(S;\theta)]$
即：使策略函数最好

—使用策略梯度上升最大化 $J(\theta)$→ $\theta_{t+1}\leftarrow\theta_t+\beta\cdot\frac{\partial V(s;\theta)}{\partial\theta}$

策略梯度 $\frac{\partial V(s;\theta)}{\partial\theta}=\sum_a\frac{\partial\pi(a|s;\theta)}{\partial\theta}\cdot Q_\pi(s,a)$

$\frac{\partial\ln\pi(\theta)}{\partial\theta}=\frac{1}{\pi(\theta)}\cdot\frac{\partial\pi(\theta)}{\partial\theta}$

$\frac{\partial V(s;\theta)}{\partial\theta}=\sum_a\pi(a|s;\theta)\cdot\frac{\partial\ln\pi(a|s;\theta)}{\partial\theta}\cdot Q_\pi(s,a)$

$\frac{\partial V(s;\theta)}{\partial\theta}=\mathbb{E}_A\left[\frac{\partial\ln\pi(A|s;\theta)}{\partial\theta}\cdot Q_\pi(s,A)\right]$

减0

$\frac{\partial V(s;\theta)}{\partial\theta}=\mathbb{E}_A\left[\frac{\partial\ln\pi(A|s;\theta)}{\partial\theta}\cdot Q_\pi(s,A)\right]-\mathbb{E}_A\left[b\cdot\frac{\partial\ln\pi(A|s;\theta)}{\partial\theta}\right]$

$\frac{\partial V(s;\theta)}{\partial\theta}=\mathbb{E}_A\left\{\frac{\partial\ln\pi(A|s;\theta)}{\partial\theta}\cdot[Q_\pi(s,A)-b]\right\}$
使用带基线的策略梯度可以降低方差，加快收敛速度

使用状态价值函数作为 baseline $b=V_\pi(s)$

$\frac{\partial V(s;\theta)}{\partial\theta}=\mathbb{E}_A\left\{\frac{\partial\ln\pi(A|s;\theta)}{\partial\theta}\cdot[Q_\pi(s,A)-V_\pi(s)]\right\}$

对于任意一个与动作 A 无关的 baseline 函数 b 具有下面的性质：
$\mathbb{E}_A\left[b\cdot\frac{\partial\ln\pi(A|s;\theta)}{\partial\theta}\right]$
$=b\cdot\mathbb{E}_A\left[\frac{\partial\ln\pi(A|s;\theta)}{\partial\theta}\right]$
$=b\cdot\sum_a\pi(a|s;\theta)\cdot\frac{\partial\ln\pi(a|s;\theta)}{\partial\theta}$
$=b\cdot\sum_a\pi(a|s;\theta)\cdot\left[\frac{1}{\pi(a|s;\theta)}\cdot\frac{\partial\pi(a|s;\theta)}{\partial\theta}\right]$
$=b\cdot\sum_a\frac{\partial\pi(a|s;\theta)}{\partial\theta}=b\cdot\frac{\partial}{\partial\theta}\sum_a\pi(a|s;\theta)$
$=b\cdot\frac{\partial}{\partial\theta}1=0$

置信域策略优化 (Trust Region Policy Optimization, TRPO)

1. 使用策略 $A\sim\pi(\cdot|s;\theta_{old})$ 完成一整局游戏 $s_1,a_1,r_1,s_2,a_2,r_2,\cdots,s_T,a_T,r_T$
2. 计算每一步 $t=1,2,\ldots,T$ 的折扣回报 $u_t=\sum_{k=1}^{T-t}\gamma^{k-1}\cdot r_k$
3. 使用均值估计期望近似目标函数 $J(\theta)=L(\theta|\theta_{old})=\frac{1}{T}\sum_{t=1}^{T}\frac{\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_{old})}\cdot u_t$
4. 求解等约束条件的优化问题 $\theta_{new}\leftarrow\underset{\theta}{\operatorname{argmax}}L(\theta|\theta_{old})$；s.t. $\theta\in\mathcal{N}(\theta_{old})$，其中邻域 $\mathcal{N}(\theta_{old})$ 可以选择：
   - $\|\theta-\theta_{old}\|_2<\Delta$
   - $\frac{1}{T}\sum_{t=1}^{T}D_{KL}[\pi(\cdot|s_t;\theta_{old})\|\pi(\cdot|s_t;\theta)]<\Delta$
5. 每个 Episode 重复进代上述 4 步

TRPO 限制了策略网络参数 θ 的更新范围，使得策略网络不会因为更新幅度过大导致策略网络发散

近端策略优化 (Proximal Policy Optimization, PPO)

目标函数中鼓励小值的第一项是 TRPO 中的目标函数
目标函数中鼓励小值的第二项是 TRPO 目标函数的 clipped version (ϵ 通常取 0.2)
$\text{clip}(x,a,b)=\begin{cases}a, & \text{if }x<a;\\x, & \text{if }a\leq x\leq b;\\b, & \text{if }x>b.\end{cases}$

训练流程：

1. 使用神经网络 $\pi(s;\omega)$ 近似状态价值函数 $V_\pi(s)$
2. 使用策略 $A\sim\pi(\cdot|s;\theta_{old})$ 完成一整局游戏 $s_1,a_1,r_1,s_2,a_2,r_2,\cdots,s_T,a_T,r_T$
3. 使用 REINFORCE with baseline 或 A2C 的方法估计每一步的优势函数值 $\hat{A}_t$，其中 $t=1,2,\ldots,T$
4. 使用价值估计和望近似目标函数
   $J(\theta)=L(\theta)=\frac{1}{T}\sum_{t=1}^{T}\min\left[r(\theta)\cdot\hat{A}_t,\text{clip}(r(\theta),1-\epsilon,1+\epsilon)\cdot\hat{A}_t\right]$
5. 求解无约束优化问题 $\theta_{new}\leftarrow\underset{\theta}{\operatorname{argmax}}L(\theta)$
6. 每个 Episode 重复进代 2-5 步

PPO 无需计算 KL 散度 D_{KL}，因此与 TRPO 相比减少了计算量；
同时 PPO 目标函数中的 clip 函数同样限制了策略网络参数 θ 的更新幅度

实际使用优势函数 $J(\theta)=\mathbb{E}_{S,A}\left[\frac{\pi(A|S;\theta)}{\pi(A|S;\theta_{old})}\cdot A_\pi(S,A)\right]$，
当 $A_\pi>0$ 时，θ 的更新使 $\pi(a_t|s_t;\theta)$ 的值增大，会让智能体在 s_t 状态下更倾向于选择动作 a_t
当 $A_\pi<0$ 时，θ 的更新使 $\pi(a_t|s_t;\theta)$ 的值减小，会让智能体在 s_t 状态下更不可能选择动作 a_t
（优势函数的含义参见紫色加粗定义处）
（A_π 中的价值函数 V_π 使用神经网络近似，训练方法与带基线的策略梯度相同）

使用蒙特卡罗方法估计策略梯度

1. 根据策略 $\pi(\cdot|s;\theta)$ 随机抽样动作 $\hat{a}$
2. 计算随机策略梯度 $g(\hat{a},\theta)=\frac{\partial\ln\pi(\hat{a}|s;\theta)}{\partial\theta}\cdot[Q_\pi(s,\hat{a})]$

因为 $\mathbb{E}_A[g(A,\theta)]=\frac{\partial V(s;\theta)}{\partial\theta}$ 所以 $g(\hat{a},\theta)$ 是 $\frac{\partial V(s;\theta)}{\partial\theta}$ 的无偏估计

如何计算 $Q_\pi(s,\hat{a})$？

使用蒙特卡罗方法估计策略梯度

1. 根据策略 $\pi(\cdot|s;\theta)$ 随机抽样动作 $\hat{a}$
2. 计算随机策略梯度 $g(\hat{a},\theta)=\frac{\partial\ln\pi(\hat{a}|s;\theta)}{\partial\theta}\cdot[Q_\pi(s,\hat{a})-V_\pi(s)]$

如何计算 $Q_\pi(s,\hat{a})$ 以及 $V_\pi(s)$？

REINFORCE

1. 完成一整局游戏 $s_1,a_1,r_1,s_2,a_2,r_2,\cdots,s_T,a_T,r_T$
2. 使用 $u_t=\sum_{k=1}^{T-t}\gamma^{k-1}r_k$ 近似 $Q_\pi(s_t,a_t)$
3. 梯度上升更新策略网络参数
   $\theta_{t+1}\leftarrow\theta_t+\beta\cdot g(a_t,\theta_t)=\theta_t+\beta\cdot u_t\cdot\frac{\partial\ln\pi(a_t|s_t;\theta)}{\partial\theta}\Big|_{\theta=\theta_t}$

一整局游戏 $s_1,a_1,r_1,s_2,a_2,r_2,\cdots,s_T,a_T,r_T$ 结束后，进行 T 轮更新，分别使用 $t=1,2,\ldots,T$ 更新策略网络；然后进行下一局游戏

Actor-Critic

使用神经网络 $q(s,a;\omega)$ 近似 $Q_\pi(s,a)$（将状态的特征向量和动作的特征向量拼接起来作为价值网络的输出）

- Actor：策略网络 $\pi(a|s;\theta)$
- Critic：价值网络 $q(s,a;\omega)$

训练流程：

1. 观察状态 s_t 并随机抽样动作 $a_t\sim\pi(\cdot|s_t;\theta_t)$
2. 执行动作 a_t 得到环境的奖励 r_t 和状态 s_{t+1}
3. 针对新状态抽样下一个动作 $a_t^-\sim\pi(\cdot|s_{t+1};\theta_t)$ 但不执行
4. 计算 $q_t=q(s_t,a_t;\omega)$ 以及 $q_{t+1}=q(s_{t+1},a_{t+1}^-;\omega_t)$
5. 计算 TD target $y_t=r_t+\gamma\cdot q_{t+1}$
6. 计算 TD error $\delta_t=q_t-y_t$
7. 计算价值网络的梯度 $d_{\omega,t}=\frac{\partial q(s_t,a_t;\omega)}{\partial\omega}\Big|_{\omega=\omega_t}$
8. 梯度下降更新价值网络参数 $\omega_{t+1}=\omega_t-\alpha\cdot\delta_t\cdot d_{\omega,t}$
9. 计算策略网络梯度 $d_{\theta,t}=\frac{\partial\ln\pi(a_t|s_t;\theta)}{\partial\theta}\Big|_{\theta=\theta_t}$
10. 梯度上升更新策略网络参数 $\theta_{t+1}=\theta_t+\beta\cdot q_t\cdot d_{\theta,t}$

使用 TD 算法更新价值网络；使用策略梯度更新策略网络

REINFORCE with baseline

1. 完成一整局游戏 $s_1,a_1,r_1,s_2,a_2,r_2,\cdots,s_T,a_T,r_T$
2. 使用 $u_t=\sum_{k=1}^{T-t}\gamma^{k-1}r_k$ 近似 $Q_\pi(s_t,a_t)$
3. 使用神经网络 $v(s;\omega)$ 近似 $V_\pi(s)$

记 $v(s_t;\omega_t)=u_t=\delta_t$

更新策略网络 $\pi(a|s;\theta)$：

1. $\theta_{t+1}\leftarrow\theta_t+\beta\cdot g(a_t,\theta_t)=\theta_t+\beta\cdot\delta_t\cdot\frac{\partial\ln\pi(a_t|s;\theta)}{\partial\theta}\Big|_{\theta=\theta_t}$

更新价值网络 $v(s;\omega)$：

1. 最小化损失函数 $L_t=\frac{1}{2}\delta_t^2$ 使价值网络向真价值函数（即：折扣回报）
2. 梯度下降更新网络参数
   $\omega_{t+1}\leftarrow\omega_t-\alpha\cdot\frac{\partial L_t}{\partial\omega}=\omega_t-\alpha\cdot\delta_t\cdot\frac{\partial v(s;\omega)}{\partial\omega}\Big|_{\omega=\omega_t}$

一整局游戏 $s_1,a_1,r_1,s_2,a_2,r_2,\cdots,s_T,a_T,r_T$ 结束后，进行 T 轮更新，分别使用 $t=1,2,\ldots,T$ 更新策略网络和价值网络；然后进行下一局游戏

Advantage Actor-Critic (A2C)

使用 $r_t+\gamma\cdot V_\pi(s_{t+1})$ 近似 $Q_\pi(s_t,a_t)$：
$g(a_t,\theta)\approx\frac{\partial\ln\pi(a_t|s_t;\theta)}{\partial\theta}\cdot[r_t+\gamma\cdot V_\pi(s_{t+1})-V_\pi(s)]$
使用神经网络 $\pi(a|s;\omega)$ 近似 $V_\pi(s)$：
$q(a_t,\theta)\approx\frac{\partial\ln\pi(a_t|s_t;\omega)}{\partial\omega}\cdot[r_t+\gamma\cdot v(s_{t+1};\omega)-v(s_t;\omega)]$

训练流程：

1. 观察得到轨迹 (s_t,a_t,r_t,s_{t+1})
2. 计算 TD target $y_t=r_t+\gamma\cdot v(s_{t+1};\omega_t)$
3. 计算 TD error $\delta_t=v(s_t;\omega_t)-y_t$
4. 使用梯度上升更新策略网络
   $\theta_{t+1}\leftarrow\theta_t+\beta\cdot g(a_t,\theta_t)=\theta_t+\beta\cdot\delta_t\cdot\frac{\partial\ln\pi(a_t|s_t;\theta)}{\partial\theta}\Big|_{\theta=\theta_t}$
5. 使用梯度下降更新价值网络
   $\omega_{t+1}\leftarrow\omega_t-\alpha\cdot\frac{\partial L_t}{\partial\omega}=\omega_t-\alpha\cdot\delta_t\cdot\frac{\partial v(s;\omega)}{\partial\omega}\Big|_{\omega=\omega_t}$ 其中损失函数 $L_t=\frac{1}{2}\delta_t^2$

$m=T$
REINFORCE

Multi-Step TD Target

连续观察 m 条轨迹 $\{(s_{t+i},a_{t+i},r_{t+i},s_{t+i+1})\}_{i=0}^{m-1}$

计算 TD target 时使用 m 个已知奖励 $y_t=\sum_{k=0}^{m-1}\gamma^k\cdot r_{t+k}+\gamma^m\cdot v(s_{t+m};\omega)$

其他计算、更新公式均保持不变

$m=1$
One-Step TD Target

[个人主页]

参考资料

- 《深度强化学习》王树森
- 深度强化学习课程
- Stable-Baselines3
- Open AI Spinning Up in Deep RL
- PPO explanation

TD Target		
	Sarsa 算法	Q-Learning 算法
特点	计算 TD target 使用的动作来自策略函数 $a_{t+1}\sim\pi(\cdot\|s_{t+1})$	计算 TD target 使用的动作为价值最高的动作 $a_{t+1}=\underset{a}{\operatorname{argmax}}\,Q^*(s_{t+1},a)$
TD target	$y_t=r_t+\gamma\cdot q(s_{t+1},a_{t+1};\omega_t)$	$y_t=r_t+\gamma\cdot\max_a Q(s_{t+1},a;\omega_t)$
应用	Actor-Critic	DQN

This work by liuzhaoze is licensed under CC BY-NC-ND 4.0