《强化学习》课程之第六讲

时序差分法

苏州大学计算机科学与技术学院

主讲：刘 全

引言

- 在未知环境模型的情况下，虽然蒙特卡洛法能够求解MDP问题，但需要等到一个情节结束后才能够更新值函数；
- 在实际场景中，有些任务没有终止状态或者难以到达终止状态；
- 结合DP的自举性和MC的采样性，学习由时间间隔产生的差分数据，通过迭代更新来求解未知环境模型的MDP问题的方法（时序差分法） 。

目 录

6.1 时序差分预测（1）

- **利用时序差分解决预测问题：**

状态值函数增量式更新递归式为：

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

在时序差分预测中，每前进1-步或***n***-步，就可以直接计算状态值（动作值）函数。本章仅讨论单步情况的时序差分算法，即TD(0)（单步TD）。

TD(0)算法是***n***-步TD算法的特列，适用于小批量状态的更新方法。其中“0”表示向前行动1步。

6.1 时序差分预测（2）

TD(0)算法的核心思想是向前行动1步后，使用得到的立即奖赏R_{t+1}和下一个状态的状态值函数的估计值$V(S_{t+1})$来进行更新。这也是TD(0)被称为单步TD方法的原因。更新递归式为：

$$V(S_t) \leftarrow V(S_t) + \alpha\left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]$$

- ✓ $R_{t+1} + \gamma V(S_{t+1})$表示根据样本得到的时序差分目标值，$V(S_{t+1})$表示 $t+1$ 时刻状态值函数的估计值；
- ✓ 右侧的 $V(S_t)$表示t 时刻的状态值函数 $v_\pi(s)$ 的估计值，通过迭代收敛来逼近真实状态值函数；
- ✓ $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ 为时序差分误差（TD error）。记为δ_t，表示t 时刻的估计误差。

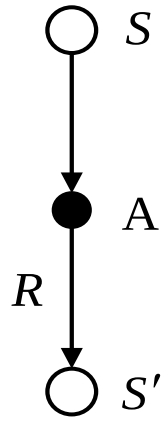
6.1 时序差分预测（3）

（1）对于DP、MC与DP的更新图比较

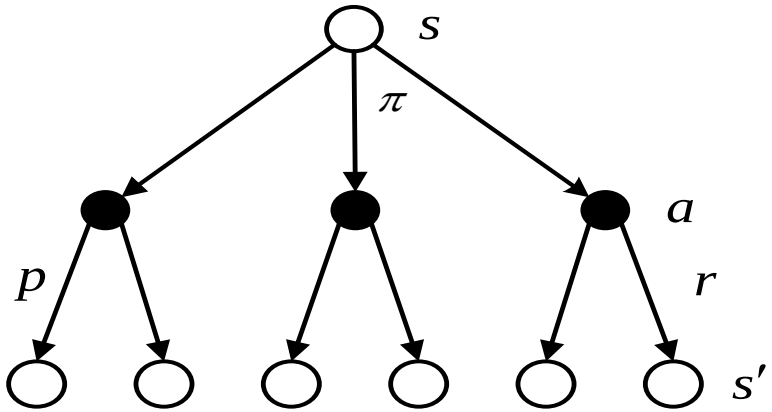
MC和TD算法对值函数的更新方法称为采样更新或样本更新。所谓的采样更新是通过采样得到一个即时后继状态（或状态-动作对），并使用即时后继状态的价值和迁移得到的奖赏来计算目标值，然后更新值函数估计值。

MC和TD算法的采样更新与DP法的期望更新思想不同，采样更新利用下一时刻单一的样本转换，而期望更新利用所有可能的下一状态的分布。

6.1 时序差分预测（4）



TD(0)更新图



DP更新图

6.1 时序差分预测（5）

（2）对DP、MC、TD的误差来源进行分析。根据值函数计算公式与贝尔曼方程可知：

$$
\begin{aligned}
\mathrm{v}_\pi(s) &= \mathbb{E}_\pi(G_t \mid S_t = s) && \text{MC法} \\
&= \mathbb{E}_\pi(R_{t+1} + \gamma G_{t+1} \mid S_t = s) \\
&= \mathbb{E}_\pi(R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s) && \text{DP法}
\end{aligned}
$$

- DP算法具有自举性。通常使用估计值来代替真实函数。
- 在MC算法中，通过采样回报的期望代替真实回报。
- 在TD算法中，TD误差是由采样获得的，另外由于自举特性 $v_\pi(S_{t+1})$ 也是未知的。

6.1 时序差分预测（6）

（3）有偏估计和无偏估计：

- 无偏估计指估计量 $\hat{\theta}$ 的数学期望等于被估计参数 θ 的真实值，则称此估计量 $\hat{\theta}$ 为被估计参数 θ 的无偏估计，即 $\mathbb{E}(\hat{\theta}) = \theta$。
- 有偏估计指样本求得的的估计参数不等于待估计参数的真实值， $\mathbb{E}(\hat{\theta}) \neq \theta$ 。
- MC算法是高方差的无偏估计。因为它是以真实回报作为目标值。但是它的方差较高。
- TD算法是低方差的有偏估计。因为TD算法使用了自举思想。

6.1 时序差分预测（7）

（4）MC误差与TD误差：

在一个情节中，当状态值函数估计值 没有发生变化时，MC误差可写为TD误差之和：

$$
\begin{aligned}
G_t - V(S_t) &= R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1}) \\
&= \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right) + \gamma G_{t+1} - \gamma V(S_{t+1}) \\
&= \delta_t + \gamma\left(G_{t+1} - V(S_{t+1})\right) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2\left(G_{t+2} - V(S_{t+2})\right) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \ldots + \gamma^{\mathrm{T}-t-1}\delta_{\mathrm{T}\text{-}1} + \gamma^{\mathrm{T}-t}\left(G_{\mathrm{T}} - V(S_{\mathrm{T}})\right) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \ldots + \gamma^{\mathrm{T}-t-1}\delta_{T-1} + \gamma^{\mathrm{T}-t}\left(0-0\right) \quad \text{终止状态没有奖赏，} G_{\mathrm{T}} = 0 \;\; V(S_{\mathrm{T}}) = 0 \\
&= \sum_{k=1}^{\mathrm{T}-1} \gamma^{k-t}\delta_k
\end{aligned}
$$

6.1 时序差分预测（8）

（5）TD(0)伪代码：

算法6.1 用于估计 v_π 的表格型 TD(0) 算法

输 入：

待评估的策略 π，折扣因子 γ，学习率 $\{\alpha_k\}_{k=0}^{\infty} \in [0,1]$

初始化：

1. 对任意 $s \in \mathcal{S}^+$；初始化状态值函数，如 $V(s) \leftarrow \mathbb{R}$; $V(s^{\mathrm{T}}) \leftarrow 0$

2. **repeat** 对每个情节 $k = 0,1,2,\cdots$
3. 初始状态 S
4. **while**
5. 根据策略 π，选择动作 A
6. 在状态 S 下执行动作 A，到达下一状态 S'，并得到奖赏 R
7. $V(S) \leftarrow V(S) + \alpha_k [R + \gamma V(S') - V(S)]$
8. $S \leftarrow S'$
9. **until** $S = S^{\mathrm{T}}$

输 出：$v_\pi = V$

目 录

6.2 时序差分控制（1）

TD控制算法和MC控制算法的共同点：

- 都遵循GPI；
- 评估的目标都是状态-动作对的最优动作值函数。

为了平衡探索与利用，TD控制算法分为：

- 基于策略迭代的同策略Sarsa算法；
- 基于值迭代的异策略Q-Learning算法。

Sarsa算法与Q-Learning算法都属于TD(0)算法，它们的收敛性同样遵循MC增量式中的随机近似条件。

目 录

6.2.1 Sarsa算法（1）

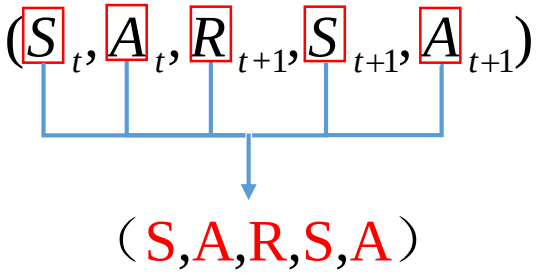
- **Sarsa算法**

  Sarsa算法的动作值函数更新迭代式为：

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right]$$

  - ✓ 目标值： $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$
  - ✓ TD误差： $\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$

- 称作Sarsa算法的原因：
  - ✓ 每次更新都需要获取五元组：



6.2.1. Sarsa算法（2）

➢**Sarsa算法**

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

注：**Sarsa算法是同策略时序差分法，更新策略与行动策略相同。**

6.2.1. Sarsa算法（3）

- 为了能收敛到最优Q值函数q_*和最优策略π_*，Sarsa算法要求以一定的概率进行探索的同时，探索策略必须逐渐变得贪心。例如在使用ε–贪心策略时，其探索概率ε逐渐衰减到0（用$\{\varepsilon_k\}_{k=0}^{\infty}$表示衰减过程）。
- 如果使用Boltzmann探索，其探索温度参数τ_k逐渐衰减到0。这样当Sarsa算法使用具有贪心性质的策略时也会逐渐变得贪心。

6.2.1. Sarsa算法（4）

- 使用Sarsa算法解决确定环境扫地机器人问题

初始化：

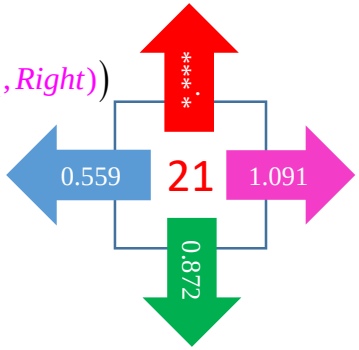
- ✓ 动作值函数：24×4 的二维数组，且初值都为**0**
- ✓ 学习率参数α_0：**0.05**
- ✓ γ：**0.8**
- ✓ ε-贪心策略：ε_0=0.5

6.2.1. Sarsa算法（5）

基于Sarsa算法的确定环境扫地机器人问题的Q值迭代过程

Q_k 状态	ε	α	S_{16}	…	S_{20}	S_{21}	S_{22}	…
Q_0	0.500	0.050	0.000 0.000; 0.000 0.000	…	*.*** *.***; 0.000 0.000	*.*** 0.000; 0.000 0.000	*.*** 0.000; 0.000 0.000	…
Q_1	0.500	0.050	0.000 0.000; 0.000 0.000	…	*.*** *.***; 0.000 0.000	*.*** 0.000; 0.000 0.000	*.*** 0.000; 0.000 0.000	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Q_{3000}	0.425	0.0425	0.706 0.566; 1.379 0.509	…	*.*** *.***; 0.787 0.565	*.*** 0.557; 1.086 0.872	*.*** 0.704; 1.483 1.365	…
Q_{3001}	0.425	0.0425	0.706 0.566; 1.379 0.509	…	*.*** *.***; 0.777 0.565	*.*** ; 1.091 0.872	*.*** 0.704; 1.486 1.365	…

$$Q(S_{21},Right)=Q(S_{21},Right)+\alpha_{3001}*\left(R+\gamma*Q\left(S_{22},Right\right)-Q(S_{21},Right)\right)$$
$$=1.086+0.0425*\left(0+0.8*1.483-1.086\right)$$
$$=1.091$$



6.2.1. Sarsa算法（6）

基于Sarsa算法的确定环境扫地机器人问题的Q值迭代过程（续）

Q_k 状态	ε	α	S_{16}	···	S_{20}	S_{21}	S_{22}	···
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Q_{19999}	0.000	0.000	1.053 0.898; 1.805 0.781	···	*.*** *.***; 1.205 1.019	*.*** 0.897; 1.518 1.352	*.*** 1.127; 1.907 1.734	···
π_{19999}			0.000 0.000; 0.000 1.000	⋮	0.000 0.000; 1.000 0.000	0.000 0.000; 1.000 0.000	0.000 0.000; 1.000 0.000	⋮
Q_{20000}	0.000	0.000	1.053 0.898; 1.805 0.781	···	*.*** *.***; 1.205 1.019	*.*** 0.897; 1.518 1.352	*.*** 1.127; 1.907 1.734	···
π_{20000}			0.000 0.000; 0.000 1.000	···	0.000 0.000; 1.000 0.000	0.000 0.000; 1.000 0.000	0.000 0.000; 1.000 0.000	···
π_*	0.000	0.000	0.000 0.000; 0.000 1.000	···	0.000 0.000; 1.000 0.000	0.000 0.000; 1.000 0.000	0.000 0.000; 1.000 0.000	

目 录

6.2.2. Q-Learning算法（1）

- **Q-Learning算法**

Q-Learning算法的动作值函数更新迭代式为：

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

✓动作值函数Q的更新方向是最优动作值函数q_*,而与Agent所遵循的行为策略b无关。

✓在这里评估动作值函数Q时，更新目标值为最优动作值函数q_*的直接近似，故需要遍历当前状态的所有动作。

6.2.2. Q-Learning算法（2）

- **Q-Learning算法**

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

6.2.2. Q-Learning算法（3）

- **使用Q-Learning算法解决确定环境扫地机器人问题**

  初始化：

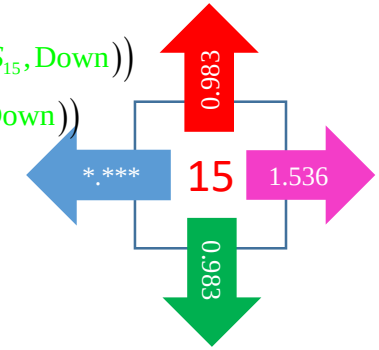
  - ✓ 动作值函数：24×4 的二维数组，且初值都为**0**
  - ✓ 学习率参数α_0：**0.05**
  - ✓ γ：**0.8**
  - ✓ ε-贪心策略：ε_0=0.5

6.2.2. Q-Learning算法（4）

基于Q-Learning算法的确定环境扫地机器人问题的Q值迭代过程

Q_k 状态	ε	α	S_{10}	…	S_{15}	S_{16}	S_{20}	…
Q_0	0.500	0.050	0.000 *.***; 0.000 0.000	…	0.000 *.***; 0.000 0.000	0.000 0.000; 0.000 0.000	*.*** *.***; 0.000 0.000	…
Q_1	0.500	0.050	0.000 *.***; 0.000 0.000	…	0.000 *.***; 0.000 0.000	0.000 0.000; 0.000 0.000	*.*** *.***; 0.000 0.000	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Q_{10999}	0.280	0.028	1.229 *.***; 1.228 0.800	…	0.983 *.***; 1.536 0.983	1.229 1.229; 1.920 1.229	*.*** *.***; 1.229 1.229	…
Q_{11000}	0.280	0.028	1.229 *.***; 1.228 0.800	…	0.983 *.***; 0.983	1.229 1.229; 1.920 1.229	*.*** *.***; 1.229 1.229	…

$$
\begin{aligned}
Q(S_{15},\text{Down}) &= Q(S_{15},\text{Down}) + \alpha_{11000}\times\left(R+\gamma * \max\left(Q(S_{10},*)\right)-Q(S_{15},\text{Down})\right)\\
&= Q(S_{15},\text{Down}) + \alpha_{11000}\times\left(R+\gamma * Q(S_{10},\text{Up})-Q(S_{15},\text{Down})\right)\\
&= 0.983+0.028\times(0+0.8\times 1.229-0.983)\\
&\approx 0.983
\end{aligned}
$$



6.2.2. Q-Learning算法（5）

基于Q-Learning算法的确定环境扫地机器人问题的Q值迭代过程（续）

Q_k 状态	ε	α	S_{16}	…	S_{20}	S_{21}	S_{22}	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Q_{24999}	0.000	0.000	1.229 *.***; 1.228 0.800	…	0.983 *.***; 1.536 0.983	1.229 1.229; 1.920 1.229	*.*** *.***; 1.229 1.229	…
b_{24999}			1.000 0.000; 0.000 0.000	⋮	0.000 0.000; 1.000 0.000	0.000 0.000; 1.000 0.000	0.000 0.000; 1.000 0.000	⋮
Q_{25000}	0.000	0.000	1.229 *.***; 1.228 0.800	…	0.983 *.***; 1.536 0.983	1.229 1.229; 1.920 1.229	*.*** *.***; 1.229 1.229	…
b_{25000}			1.000 0.000; 0.000 0.000	…	0.000 0.000; 1.000 0.000	0.000 0.000; 1.000 0.000	0.000 0.000; 1.000 0.000	…
π_*	0.000	0.000	1.000 0.000; 0.000 0.000	…	0.000 0.000; 1.000 0.000	0.000 0.000; 1.000 0.000	0.000 0.000; 1.000 0.000	

目 录

6.2.3.期望Sarsa算法（1）

- **期望Sarsa的目标值为：**

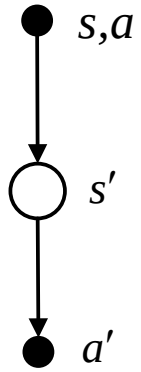
$$\begin{aligned} G_t &= R_{t+1} + \gamma \mathbb{E}\left[Q(S_{t+1}, A_{t+1}) \mid S_{t+1} \right] \\ &= R_{t+1} + \gamma \sum_a \pi(a \mid S_{t+1}) Q(S_{t+1}, a) \end{aligned}$$

- **期望Sarsa动作值函数的更新递归式为：**

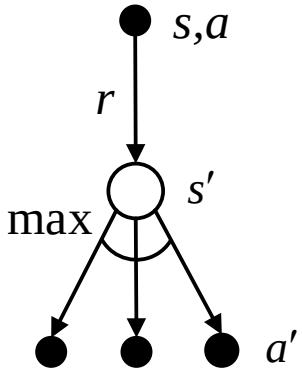
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \sum_a \pi(a \mid S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

- **期望Sarsa优于Sarsa；**
- **期望Sarsa还可以使用异策略方法，将Q-Learning进行推广，并提升性能。**

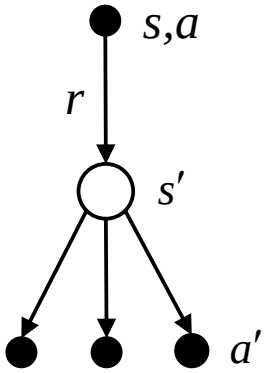
6.2.3.期望Sarsa算法（2）



Sarsa更新图



Q-Learning更新图



期望Sarsa更新图

6.2.3.期望Sarsa算法（3）

- **使用期望Sarsa算法解决确定环境扫地机器人问题**

  初始化：

  - ✓ 动作值函数：24×4 的二维数组，且初值都为**0**
  - ✓ 学习率参数α_0：**0.05**
  - ✓ γ：**0.8**
  - ✓ ε-贪心策略：ε_0=0.5

6.2.3.期望Sarsa算法（4）

基于期望Sarsa算法的确定环境扫地机器人问题的Q值迭代过程

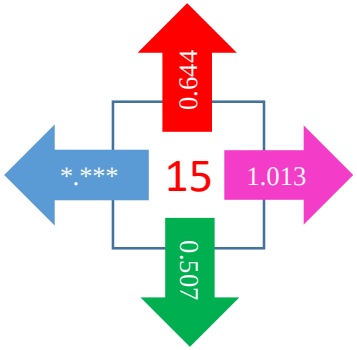
Q_k 状态	ε	α	S_{10}	···	S_{15}	S_{16}	S_{20}	···
···			···	···				···
Q_{6001}	0.350	0.035	0.533 *.***; 0.440 0.699	···	0.644 *.***; 1.013 0.507	0.879 0.683; 1.471 0.606	*.*** *.***; 0.919 0.723	···
π_{6001}			0.117 *.***; 0.117 0.767	⋮	0.117 *.***; 0.767 0.117	0.087 0.087; 0.738 0.087	*.*** *.***; 0.825 0.175	

$$Q(S_{15},\text{Down}) = Q(S_{15},\text{Down}) + \alpha_{6001} \times \left(R + \gamma \times \sum_{a \in A(10)} \left(\text{prob}[a] \times \left(Q(S_{10},a) \right) \right) - Q(S_{15},\text{Down}) \right)$$

$$= Q(S_{15},\text{Down}) + \alpha_{6000} \times \left(R + \gamma \times (\text{prob}[\text{Up}] \times Q(S_{10},\text{Up}) + \text{prob}[\text{Down}] \times Q(S_{10},\text{Down}) + \text{prob}[\text{Right}] \times Q(S_{10},\text{Right})) - Q(S_{15},\text{Down}) \right)$$

$$= 0.506 + 0.035 \times \left(0 + 0.8 \times (0.117 \times 0.533 + 0.767 \times 0.699 + 0.117 \times 0.440) - 0.506 \right)$$

$$\approx 0.507$$



目 录

6.2.4. Double Q-Learning算法（1）

- **Sarsa和Q-Learning算法都是采用目标策略最大化的思想，即采用基于ε-贪心策略（Sarsa）或贪心策略（Q-Learning）,产生最大化偏差。**
- **最大化偏差不会导致算法失败，但是会让收敛速度变慢。**

6.2.4. Double Q-Learning算法（2）

Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
    Take action A, observe R, S'
    With 0.5 probabilility:

$$Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S', a)\big) - Q_1(S, A)\Big)$$

    else:

$$Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S', a)\big) - Q_2(S, A)\Big)$$

    $S \leftarrow S'$
  until S is terminal

6.2.4. Double Q-Learning算法（3）

- **使用Double Q-Learning算法解决确定环境扫地机器人问题**

  初始化：

  - ✓ 动作值函数：24×4 的二维数组，且初值都为**0**
  - ✓ 学习率参数α_0：**0.05**
  - ✓ γ：**0.8**
  - ✓ ε-贪心策略：$\varepsilon_0=0.5$

6.2.4. Double Q-Learning算法（4）

基于Double Q-Lerning算法的确定环境扫地机器人问题的Q值迭代过程

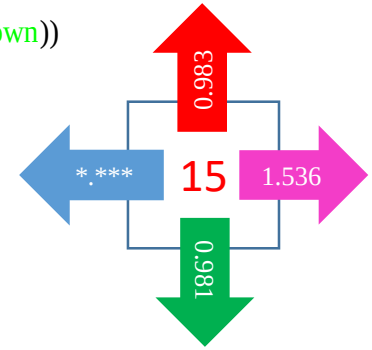
Q_k 状态	ε	α	S_{10}	…	S_{15}	S_{16}	S_{20}	…
…			…	…				…
Q^1_{8640}	0.392	0.0392	1.228 *.***; 0.973 0.800	…	0.983 *.***; 1.536 0.981	1.228 1.229; 1.920 1.227	*.*** *.***; 1.229 1.229	…
Q^2_{8640}			1.229 *.***; 1.100 0.800	⋮	0.983 *.***; 1.536 0.981	1.229 1.229; 1.920 1.225	*.*** *.***; 1.229 1.229	
π_{6001}			0.739 0.000; 0.131 0.131		0.131 0.000; 0.739 0.131	0.098 0.098; 0.706 0.098	0.000 0.000; 0.196 0.804	

$$Q_1(S_{15},\text{Down})=Q_1(S_{15},\text{Down})+\alpha_{8640}\times(R+\gamma\times Q_2(S_{10},\arg\max_a Q_1(S_{10},a))-Q_1(S_{15},\text{Down}))$$

$$=Q_1(S_{15},\text{Down})+\alpha_{8640}\times(R+\gamma\times Q_2(S_{10},\text{Up})-Q_1(S_{15},\text{Down}))$$

$$=0.981+0.0392\times(0+0.8\times 1.229-0.981)$$

$$\approx 0.981$$



习题（1）

1. 在TD控制算法中，Sarsa和Q-Learning分别是同策略算法还是异策略算法，为什么？

2. 举例说明在什么情况下使用TD算法通常优于MC算法和DP算法。

3. 在TD算法中，步长 会影响收敛速度和收敛效果吗？请简述理由。

4. 从估计值、自举和采样等方面，对DP、MC和TD三种方法进行对比，并说明其中哪些方法是有偏估计，哪些是无偏估计。

习题（2）

5. （编程）通过TD算法计算：第3章习题2（图3.12）扫地机器人在折扣率 $\gamma = 0.8$、初始策略为等概率策略的情况下，分别利用Sarsa算法和Q-Learning算法，计算每个状态的最优策略。

The End