《强化学习》课程之第七讲

n-步时序差分

苏州大学计算机科学与技术学院

主讲：刘 全

目 录

引言

TD(0)算法：

- 每一时间步均更新值函数，相比MC方法更充分考虑环境变化；
- 通常情况下，实际环境不会频繁地发生变化，而是一段时间间隔后才会发生显著变化。

***n*-步TD方法（*n*-步自举方法）：**

- TD(0)和MC方法的推广；
- 介于TD(0)和MC方法之间的方法。

7.1 *n*-步TD预测及资格迹（1）

7.1.1 *n*-步TD预测

Agent通过与环境交互，采样得情节 $S_0, A_0, R_1, S_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$ 。

- **MC状态值函数更新递归式：**

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

目标值计算公式：

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-t-1} R_T$$

- **TD(0)状态值函数更新递归式：**

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

目标值计算公式：

$$G_{t:t+1} = R_{t+1} + \gamma V_t(S_{t+1})$$

7.1 *n*-步TD预测及资格迹（2）

$G_{t:t+1}$：从t时刻到$t+1$时刻的截断回报（**truncated return**），即**1-**步回报；

V_t：t 时刻状态值函数v_π的估计值（简记为t时刻状态值）；

$\gamma V_t(S_{t+1})$：带折扣评估值，用于替代全额回报G_t 中的

$$\gamma R_{t+2}+\gamma^2 R_{t+3}+\ldots+\gamma^{\mathrm{T}-t-1}R_{\mathrm{T}}$$

7.1 *n*-步TD预测及资格迹（3）

***n*-步TD算法：**

- 更新方式介于**TD(0)**和**MC**之间；
- 利用未来多步奖赏和多步之后的值函数估计求得目标值。

例：2-步回报：

$$G_{t:t+2} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+2})$$

$\gamma^2 V_{t+1}(S_{t+2})$ 替代全额回报 G_t 中的 $\gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \ldots + \gamma^{\mathrm{T}-t-1} R_{\mathrm{T}}$ 。

***n*-步回报：**

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}), \qquad n \geq 1,\ 0 \leq t < \mathrm{T} - n$$

当 $t \geq \mathrm{T} - n$ 时，

$$G_{t:t+n} = G_t$$

7.1 *n*-步TD预测及资格迹（4）

- ***n*-步TD属于TD方法，当前状态的更新目标是当前Agent采集到的后续*n*步奖赏与后继第*n*个状态的带权价值估计之和。**
- **与TD(0)不同的是：这里的后继状态是*n*步后的状态，而不是当前状态的下一状态。**
- ***n*-步TD预测算法状态值函数更新递归式：**

$$V_{t+n}(S_t)=V_{t+n-1}(S_t)+\alpha\left[G_{t:t+n}-V_{t+n-1}(S_t)\right],\quad 0\le t<\mathrm{T}$$

在 $t+n$ 时刻，除状态 S_t的状态值得到更新外，其他状态的状态值均保持不变，即对于所有满足 $s\neq S_t$ 的 S来说，有

$$V_{t+n}(s)=V_{t+n-1}(s)$$

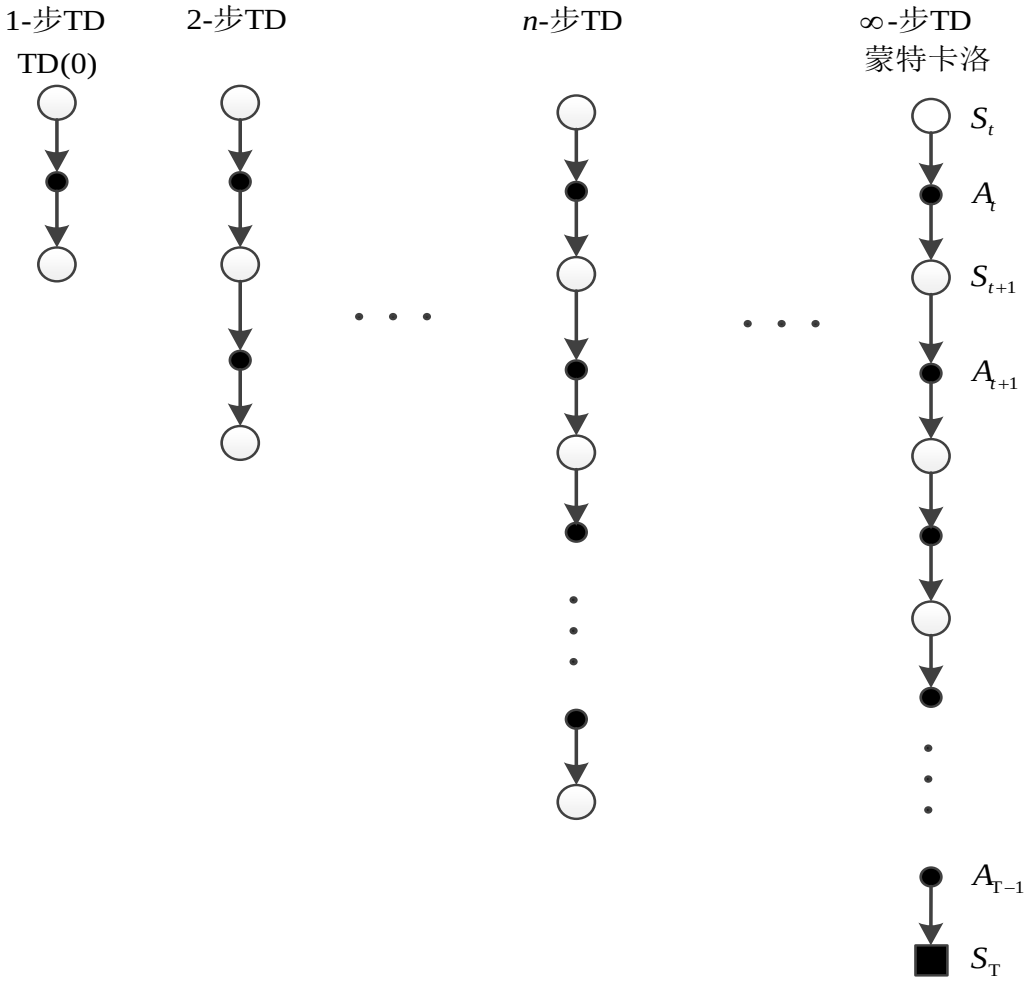
7.1 *n*-步TD预测及资格迹（5）

***n*-步TD算法的更新过程：**

- 离线过程：起始的$0\cdots n-1$时刻，所有状态的状态值均不会更新，也即 $V_0 \to V_{n-1}$均不变；
- 在线过程：n 时刻观察到状态S_n开始，更新状态值函数；
- 补偿过程：**Agent**到达终止状态且开始下一个情节采样之前，为补偿离线过程中状态值函数更新滞后现象，值函数还会进行　次更新，目标值 $G_{t:t+n} = G_t$ 。

7.1 *n*-步TD预测及资格迹（6）

***n*-步TD更新图：**



最左侧的序列为TD(0)算法，最右侧的序列为MC算法，***n*-步TD算法则介于这两个算法之间。**

7.1 n-步TD预测及资格迹（7）

算法 7.1 基于状态值函数 v_π 的 n-步 TD 预测算法

输　入：

初始策略 $\pi(a|s)$，折扣因子 γ，学习率 $\{\alpha_k\}_{k=0}^{\infty} \in [0,1]$，$n \leftarrow$ 正整数

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 $S_0 \neq S^{\mathrm{T}}$
4. $\mathrm{T} \leftarrow \infty$
5. **repeat** 对每个时间步 $t = 0, 1, 2, \cdots$
6. **if** $t < \mathrm{T}$ **then**
7. 遵循策略 $\pi(\cdot|S_t)$，选择动作 A_t
8. 执行动作 A_t，到达下一时刻的状态 S_{t+1}，并获得奖赏 R_{t+1}
9. **if** S_{t+1} 为终止状态 **then**
10. $\mathrm{T} \leftarrow t+1$
11. **end if**
12. **end if**

7.1 *n*-步TD预测及资格迹（8）

13. $\tau \leftarrow t-n+1$

14. **if** $\tau \geq 0$ **then**

15. $$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,\,\mathrm{T})} \gamma^{i-\tau-1} R_i = \begin{cases} R_{\tau+1}+\gamma R_{\tau+2}+\ldots+\gamma^{n-1}R_{\tau+n} & , \quad \tau+n<\mathrm{T} \\ R_{\tau+1}+\gamma R_{\tau+2}+\ldots+\gamma^{\mathrm{T}-\tau-1}R_{\mathrm{T}} & , \quad \tau+n\geq \mathrm{T} \end{cases}$$

16. **if** $\tau+n<\mathrm{T}$ **then**

17. $G \leftarrow G+\gamma^{n}V(S_{\tau+n})$

18. **end if**

19. $V(S_\tau) \leftarrow V(S_\tau)+\alpha_k\left[G-V(S_\tau)\right]$

20. **end if**

21. **until** $\tau=\mathrm{T}-1$

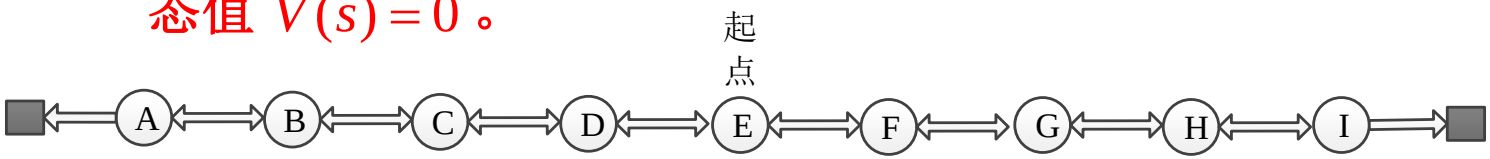
输　出：

$v_\pi = V$

7.1 *n*-步TD预测及资格迹（9）

例7.1 ***n*-步TD预测算法在随机漫步实例中的应用。**

- **Agent起始位置为E点，在A点左侧和I点右侧有两个终止位置，即左终点和右终点；**
- **Agent从A点向左走到左终点，获得-1的奖赏，从I点向右走到右终点，获得+1的奖赏，其他奖赏均为0；**
- **除了终点外，在每个点，Agent都可以等概率采取向左或向右两个动作，即** $\pi(a|s)=0.5$ **；**
- **固定学习率** $\alpha=0.5$ **，设定折扣率** $\gamma=1$ **，初始化所有状态值** $V(s)=0$ **。**



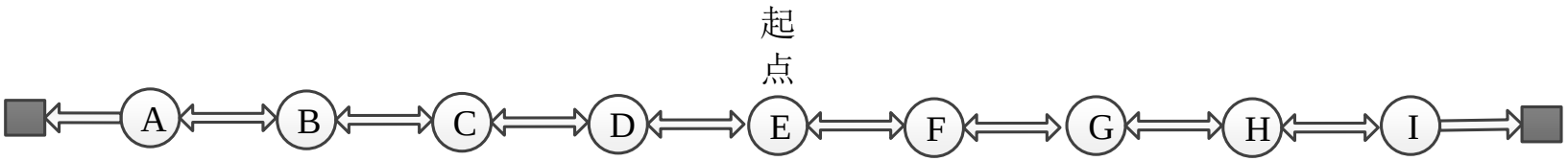
7.1 *n*-步TD预测及资格迹（10）

（1） $n=3$ 时，假设**Agent**直接从**E**一直向右走到右终点，轨迹为：

E, *Right*，0，F, *Right*，0，G，*Right*，0，H，*Right*，0，I，*Right*，+1，右终止

即有：

$$S_0=\mathrm{E},\ S_1=\mathrm{F},\ S_2=\mathrm{G},\ S_3=\mathrm{H},\ S_4=\mathrm{I},\ S_5=\text{右终止}$$



7.1 *n*-步TD预测及资格迹（11）

- 不更新阶段——离线过程

$t=0,1$ 时，$\tau=-2,-1$ ，***V***不更新；

- 正常更新阶段——在线过程

$t=2$ 时，$\tau=0$ ，开始更新 $V(\mathrm{E})$ ：

$$G_{0:3}=R_1+\gamma R_2+\gamma^2 R_3+\gamma^3 V_2(\mathrm{H})$$
$$=0+1*0+1^2*0+1^3*0$$
$$=0$$
$$V_3(\mathrm{E})=V_2(\mathrm{E})+\alpha\left[G_{0:3}-V_2(\mathrm{E})\right]$$
$$=0+0.5*(0-0)$$
$$=0$$

repeat 对每个情节 $k=0,1,2,\cdots$

初始状态 $S_0\neq S^{\mathrm{T}}$

$\mathrm{T}\leftarrow\infty$

repeat 对每个时间步 $t=0,1,2,\cdots$

if $t<\mathrm{T}$ **then**

遵循策略 $\pi(\cdot|S_t)$，选择动作 A_t

执行动作 A_t，到达下一时刻的状态 S_{t+1}，并获得奖赏 R_{t+1}

if S_{t+1} 为终止状态 **then**

$\mathrm{T}\leftarrow t+1$

end if

end if

$\tau\leftarrow t-n+1$

if $\tau\geq 0$ **then**

$$G\leftarrow\sum_{i=\tau+1}^{\min(\tau+n,\mathrm{T})}\gamma^{i-\tau-1}R_i=\begin{cases}R_{\tau+1}+\gamma R_{\tau+2}+\ldots+\gamma^{n-1}R_{\tau+n} & ,\ \tau+n<\mathrm{T}\\ R_{\tau+1}+\gamma R_{\tau+2}+\ldots+\gamma^{\mathrm{T}-\tau-1}R_{\mathrm{T}} & ,\ \tau+n\geq\mathrm{T}\end{cases}$$

if $\tau+n<\mathrm{T}$ **then**

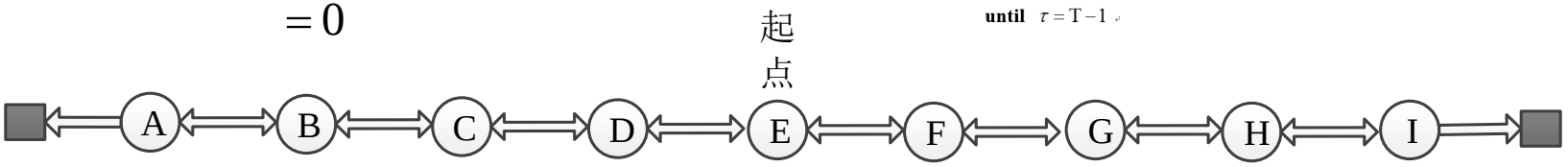
$G\leftarrow G+\gamma^n V(S_{\tau+n})$

end if

$V(S_\tau)\leftarrow V(S_\tau)+\alpha_k\left[G-V(S_\tau)\right]$

end if

until $\tau=\mathrm{T}-1$



7.1 *n*-步TD预测及资格迹（12）

- 正常更新阶段——在线过程

$t=3$ 时，$\tau=1$，开始更新 $V(\mathrm{F})$：

$$
\begin{aligned}
G_{1:4} &= R_2+\gamma R_3+\gamma^2 R_4+\gamma^3 V_3(\mathrm{I}) \\
&= 0+1*0+1^2*0+1^3*0 \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
V_4(\mathrm{F}) &= V_3(\mathrm{F})+\alpha\left[G_{1:4}-V_3(\mathrm{F})\right] \\
&= 0+0.5*(0-0) \\
&= 0
\end{aligned}
$$

repeat 对每个情节 $k=0,1,2,\cdots$

初始状态 $S_0 \neq S^{\mathrm{T}}$

$\mathrm{T} \leftarrow \infty$

repeat 对每个时间步 $t=0,1,2,\cdots$

if $t<\mathrm{T}$ **then**

遵循策略 $\pi(\cdot|S_t)$，选择动作 A_t

执行动作 A_t，到达下一时刻的状态 S_{t+1}，并获得奖赏 R_{t+1}

if S_{t+1} 为终止状态 **then**

$\mathrm{T} \leftarrow t+1$

end if

end if

$\tau \leftarrow t-n+1$

if $\tau \geq 0$ **then**

$$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,\,\mathrm{T})} \gamma^{i-\tau-1} R_i = \begin{cases} R_{\tau+1}+\gamma R_{\tau+2}+\ldots+\gamma^{n-1}R_{\tau+n} & \tau+n<\mathrm{T} \\ R_{\tau+1}+\gamma R_{\tau+2}+\ldots+\gamma^{\mathrm{T}-\tau-1}R_{\mathrm{T}} & \tau+n \geq \mathrm{T} \end{cases}$$

if $\tau+n<\mathrm{T}$ **then**

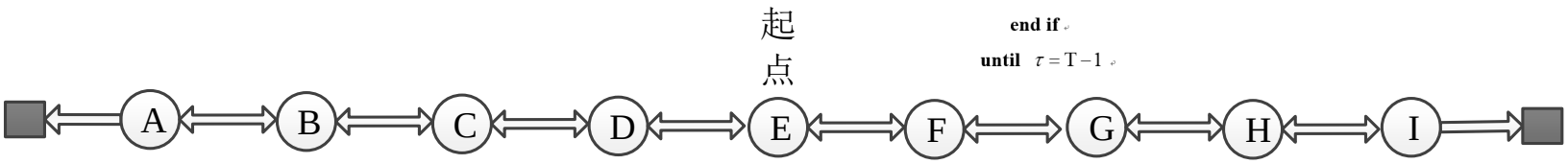
$G \leftarrow G+\gamma^n V(S_{\tau+n})$

end if

$V(S_\tau) \leftarrow V(S_\tau)+\alpha_k\left[G-V(S_\tau)\right]$

end if

until $\tau=\mathrm{T}-1$



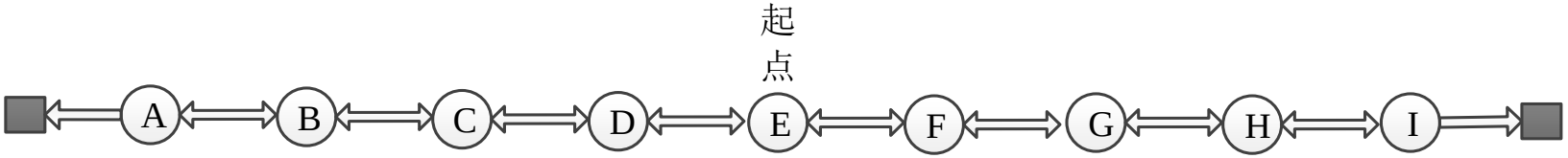
7.1 *n*-步TD预测及资格迹（13）

- 补偿阶段——当前情节采样结束，此后只更新状态值函数

$t=4$ 时，$\tau=2$，下一状态为终止状态 S_5，当前情节采样结束，开始更新 $V(\mathrm{G})$：

$$
\begin{aligned}
G_{2:5} &= R_3+\gamma R_4+\gamma^2 R_5 \\
&= 0+1*0+1^2*1 \\
&= 1
\end{aligned}
$$

$$
\begin{aligned}
V_5(\mathrm{G}) &= V_4(\mathrm{G})+\alpha\left[G_{2:5}-V_4(\mathrm{G})\right] \\
&= 0+0.5*(1-0) \\
&= 0.5
\end{aligned}
$$



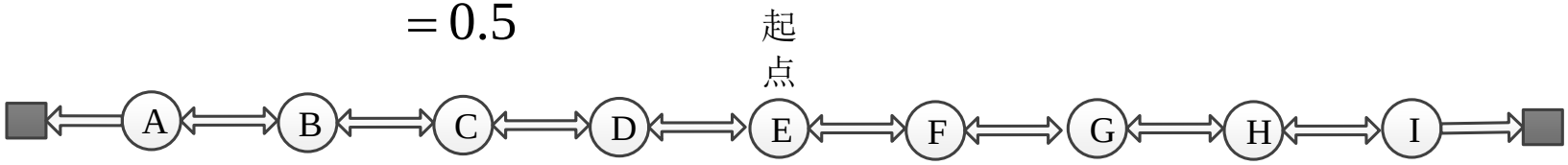
7.1 *n*-步TD预测及资格迹（14）

- 补偿阶段——当前情节采样结束，此后只更新状态值函数

$t=5$ 时，$\tau=3$，开始更新 $V(\mathrm{H})$：

$$
\begin{aligned}
G_{3:6} &= R_4 + \gamma R_5 \\
&= 0 + 1*1 \\
&= 1
\end{aligned}
$$

$$
\begin{aligned}
V_6(\mathrm{H}) &= V_5(\mathrm{H}) + \alpha\left[G_{3:6} - V_5(\mathrm{H})\right] \\
&= 0 + 0.5*(1-0) \\
&= 0.5
\end{aligned}
$$



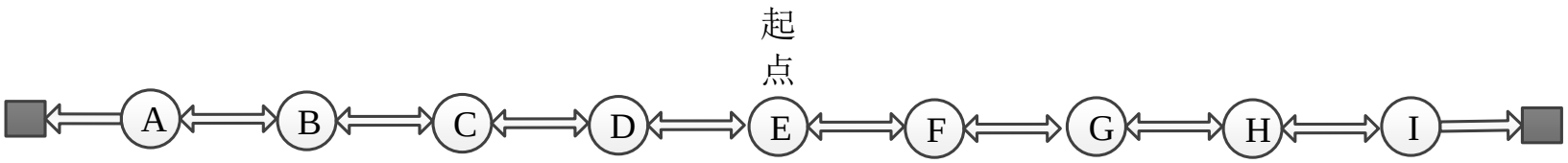
7.1 *n*-步TD预测及资格迹（15）

- 补偿阶段——当前情节采样结束，此后只更新状态值函数

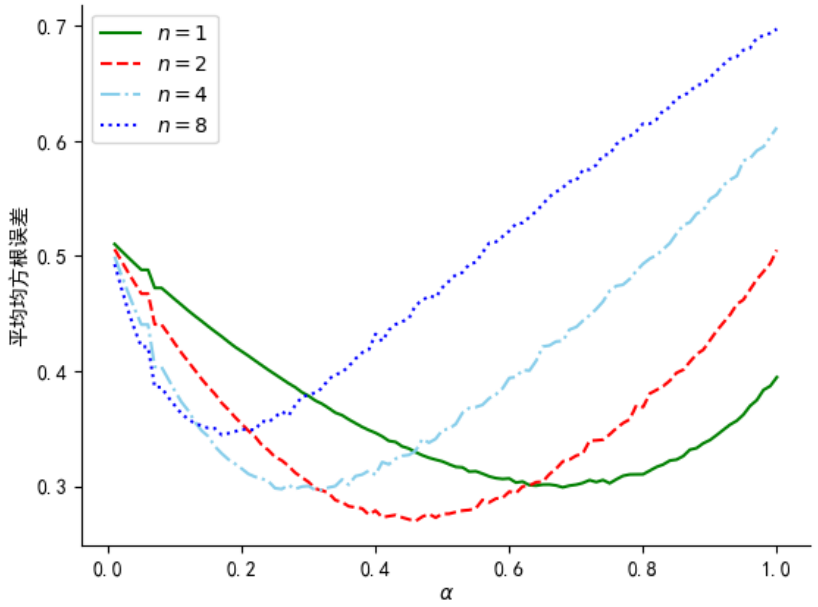
$t=6$ 时，$\tau=4$，到达迭代终止条件，开始更新 $V(\mathrm{I})$：

$$G_{4:7}=R_5=1$$

$$\begin{aligned}V_7(\mathrm{I})&=V_6(\mathrm{I})+\alpha\left[G_{4:7}-V_6(\mathrm{I})\right]\\&=0+0.5*(1-0)\\&=0.5\end{aligned}$$



7.1 *n*-步TD预测及资格迹（16）



随机漫步*n*-步TD算法性能图

- 在11个状态的随机漫步问题中， 取值略大于1时，效果会更好；
- 选取介于TD(0)和MC之间的*n*-步TD可能会得到更好的结果。

7.1 *n*-步TD预测及资格迹（17）

- ***n*-步回报误差缩减性**

$$\max_s \left|\mathbb{E}_\pi[G_{t:t+n} \mid S_t = s] - v_\pi(s)\right| \le \gamma^n \max_s \left|V_{t+n-1}(s) - v_\pi(s)\right| \qquad n \ge 1$$

- ✓ 所有的***n*-步TD**算法在合适的条件下都能收敛至正确的预测；
- ✓ 随着***n***的增加， ***n*-步回报**相对真实值函数 v_π 偏差越来越小，但是计算需要更多时间步数据，方差相对会增加；
- ✓ 选取合适的***n***，能够实现偏差和方差之间较好的权衡。

7.1 *n*-步TD预测及资格迹（18）

7.1.2 前向TD(λ)算法

目标值更新时不仅可以为单一*n*-步回报，还可以为平均*n*-步回报：

$$G_t^{ave} = \frac{1}{2} G_{t:t+2} + \frac{1}{2} G_{t:t+4}$$

- 复杂更新：
  - ✓ 平均单一轨迹回报作为目标值对值函数更新；
  - ✓ 权重之和为1；
  - ✓ 通用目标值 $G_t^{(w)}$，计算公式为：

$$G_t^{(w)} = \sum_i w_i G_{t:t+i} \quad , \quad \sum_i w_i = 1$$

7.1 *n*-步TD预测及资格迹（19）

- **TD(λ)：**
  - ✓ 一种特殊形式的平均*n*-步方法；
  - ✓ 通过引入参数λ，对所有的*n*-步回报加权求和得到目标值，目标值称为λ-回报G_t^λ：

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_{t:t+n} \quad , \quad \lambda \in [0,1]$$

7.1 *n*-步TD预测及资格迹（20）

对λ-回报进一步分离，得到 G_t^λ 的更一般化形式：

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\mathrm{T}-t-1}\lambda^{n-1}G_{t:t+n} + \lambda^{\mathrm{T}-t-1}G_t$$

可证明，所有权重和为**1**，即：

$$(1-\lambda)\sum_{n=1}^{\mathrm{T}-t-1}\lambda^{n-1} + \lambda^{\mathrm{T}-t-1}$$

$$= (1-\lambda)*(1+\lambda+\cdots+\lambda^{\mathrm{T}-t-2}) + \lambda^{\mathrm{T}-t-1}$$

$$= (1-\lambda)*\frac{1*(1-\lambda^{\mathrm{T}-t-1})}{1-\lambda} + \lambda^{\mathrm{T}-t-1}$$

$$= 1$$

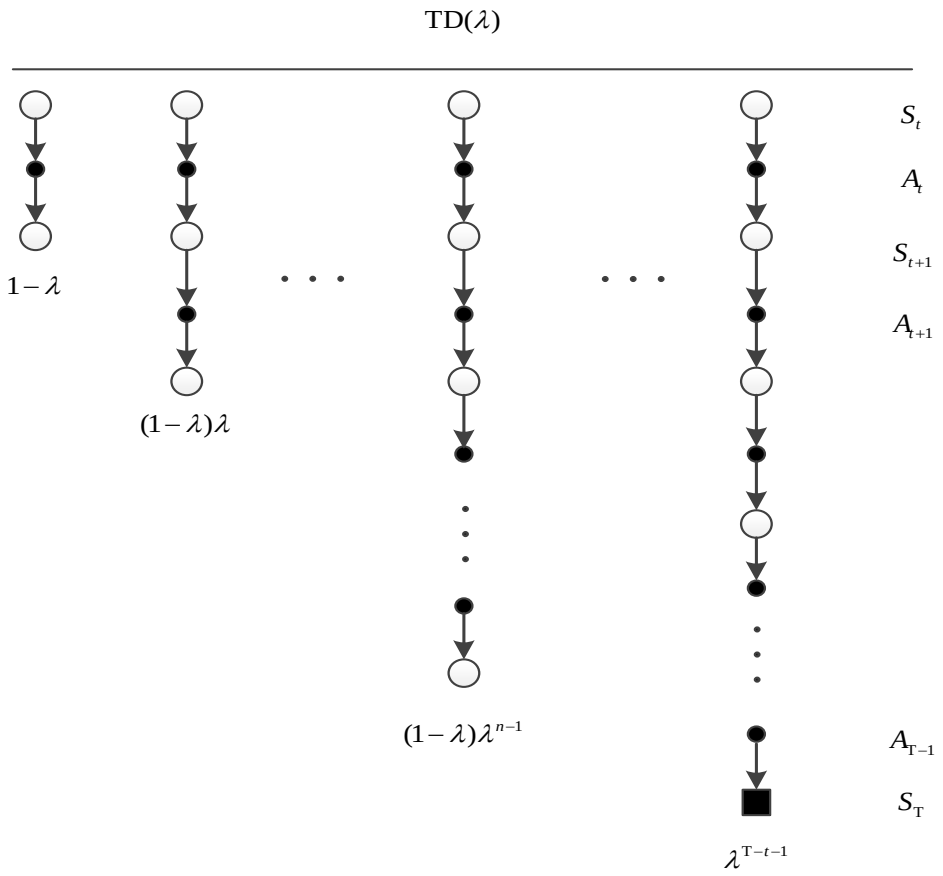
7.1 *n*-步TD预测及资格迹（21）

λ-回报：

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\mathrm{T}-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{\mathrm{T}-t-1} G_t$$

- $\lambda = 1$，与**MC**目标值 G_t 一致；
- $\lambda = 0$，与**TD(0)**（**1-步TD**）目标值 $G_{t:t+1}$ 一致；
- **TD(λ)算法介于MC和1-步TD算法之间。**

7.1 *n*-步TD预测及资格迹（22）



TD(λ)更新图

7.1 *n*-步TD预测及资格迹（23）

基于λ-回报的前向TD(λ)算法递归式：

$$V(S_t) \leftarrow V(S_t) + \alpha\left[G_t^{\lambda} - V(S_t)\right]$$

- 相比*n*-步TD预测方法，λ-回报方法不再需要人为设定参数*n*，而是使用参数λ实现偏差和方差之间的权衡；
- 通过对λ的合理选取，可以使得状态值函数的收敛速度更快，误差更小。

7.1 *n*-步TD预测及资格迹（24）

- **TD(λ)算法的前向观点：**
  - ✓ 基于未来的奖赏和状态值函数估计值，对当前状态的状态值进行更新；
  - ✓ 从存储空间角度考虑，该方法需要将未来若干步奖赏和状态等信息进行存储，存储代价较大；
  - ✓ 从逻辑角度考虑，该方法每次更新所需信息都需要在未来很多步后才能获取，不符合因果关系；
  - ✓ 方法通常不能直接实现，更多是停留在理论层面。

7.1 *n*-步TD预测及资格迹（25）

7.1.3 后向TD(λ)算法

- 资格迹（**eligibility trace**）：
  - ✓ 每个状态都有的附加内存变量；
  - ✓ 能够利用当前状态、已访问过的状态和先前已获得的奖赏，在线、增量式地对状态值函数进行更新，而无需对各时刻状态、奖赏进行存储；
  - ✓ 基本思想包含频率启发思想和就近启发思想，即将某种现象的结果归结于出现频率最高的状态或出现时间最近的状态。

7.1 *n*-步TD预测及资格迹（26）

- 资格迹（**eligibility trace**）：
  - ✓ 通俗来说，就是用“迹”（**trace**）来表达每个状态进行学习变化的“资格”（**eligible**）；
  - ✓ “资格”由状态出现频率的高低以及状态出现时间距离当前状态时间的远近来反映。
  - ✓ 基于以上思想，分为两种类型：
    - (1) 累积资格迹（**accumulating trace**）
    - (2) 替代资格迹（**alternative trace**）

7.1 *n*-步TD预测及资格迹（27）

（1）累积资格迹（accumulating trace）

- 资格迹在状态被 s 访问时进行累积，然后在该状态未被访问时逐渐衰减；
- 每走一步所有已访问过的状态的资格迹都会衰减 $\gamma\lambda$，当前状态 S_t 的资格迹则加**1**；
- 定义：

$$E_t(s)=\begin{cases}\gamma\lambda E_{t-1}(s) & s\neq S_t\\ \gamma\lambda E_{t-1}(s)+1 & s=S_t\end{cases}$$

$E_t(s)$表示 t 时刻状态 s 的资格迹，γ表示折扣因子，λ表示迹衰减系数。

7.1 *n*-步TD预测及资格迹（28）

（2）替代资格迹（alternative trace）

- 资格迹在状态被 s 访问时直接被替换为**1**，然后在该状态未被访问时逐渐衰减；
- 每走一步所有已访问过的状态的资格迹都会衰减 $\gamma\lambda$，当前状态 S_t 的资格迹则用**1**直接替代；
- 定义：

$$E_t(s)=\begin{cases}\gamma\lambda E_{t-1}(s) & s\neq S_t\\ 1 & s=S_t\end{cases}$$

7.1 *n*-步TD预测及资格迹（29）

- **TD(λ)使用的误差为1-步TD误差：**

$$\delta_t = R_{t+1} + \gamma V_t(S_{t+1}) - V_t(S_t)$$

- **利用这样的TD误差，结合资格迹，全局式成比例更新状态值函数，得到后向TD(λ)算法的状态值函数更新递归式：**

$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s) \quad , \quad s \in \mathcal{S}$$

7.1 n-步TD预测及资格迹（30）

TD(λ)优势：

- 每次更新通过引入资格迹，考虑所有访问过的状态，这样数据得到充分利用；
- 更新所需目标值的计算量大幅减少；
- 因为资格迹的存在，状态S上一次出现时刻与当前时刻越远，出现频率越低，对应值函数更新幅度也就越小。

7.1 *n*-步TD预测及资格迹（31）

前向观点和后向观点的的等价性：

- 在后向TD(λ)算法中，值函数既可以通过增量式计算来实现在线式更新，也可以先存储更新误差，累积到情节结束后离线式更新。
- 前向观点和后向观点中分别累积每个状态的值函数更新量直至情节结束，得到的两个量是相等的。

7.1 *n*-步TD预测及资格迹（32）

算法 7.2 用于估计状态值函数 v_π 的后向在线表格式 TD(λ)算法

输　入：

策略 $\pi(a|s)$，折扣因子 γ，学习率 $\{\alpha_k\}_{k=0}^{\infty} \in [0,1]$，迹衰减系数 λ

初始化：

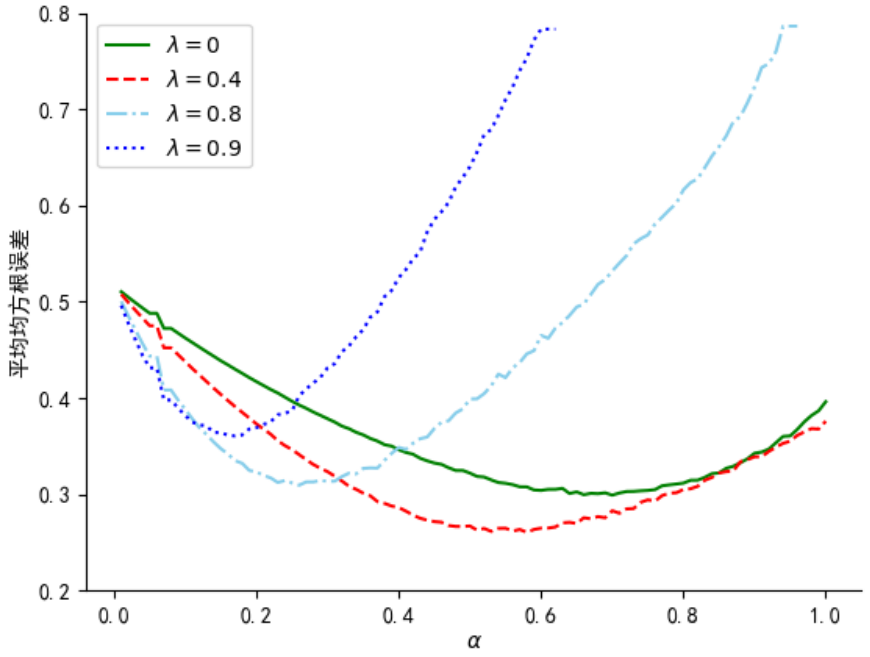
1. 对任意 $s \in S^+$，初始化状态值函数，如 $V(s) \leftarrow \mathbb{R}$；$V(s^{\mathrm{T}}) \leftarrow 0$

2. **repeat** 对每个情节 $k = 0,1,2,\cdots$
3. 　对 $s \in S$，初始化 $E(s) = 0$
4. 　初始状态 $S_0 \neq S^{\mathrm{T}}$
5. 　$\mathrm{T} \leftarrow \infty$
6. 　**repeat** 对每个时间步 $t = 0,1,2,\cdots$
7. 　　根据策略 $\pi(\cdot|S)$，选择动作 A
8. 　　执行动作 A，到达下一时刻的状态 S'，并获得奖赏 R
9. 　　$\delta \leftarrow R - V(S)$
10. 　　**if** $S' = S^{\mathrm{T}}$ **then**
11. 　　$\mathrm{T} \leftarrow t+1$
12. 　　**else**
13. 　　$\delta \leftarrow \delta + \gamma V(S')$
14. 　　**end if**
15. 　　$E(S) \leftarrow E(S)+1$（累积资格迹）　或　$E(S) \leftarrow 1$（替代资格迹）
16. 　　**for** $s \in S$ **do**
17. 　　$V(s) \leftarrow V(s) + \alpha_k \delta E(s)$
18. 　　$E(s) \leftarrow \gamma\lambda E(s)$
19. 　　**end for**
20. 　　$S \leftarrow S'$
21. 　**until** $t = \mathrm{T} - 1$

输　出：

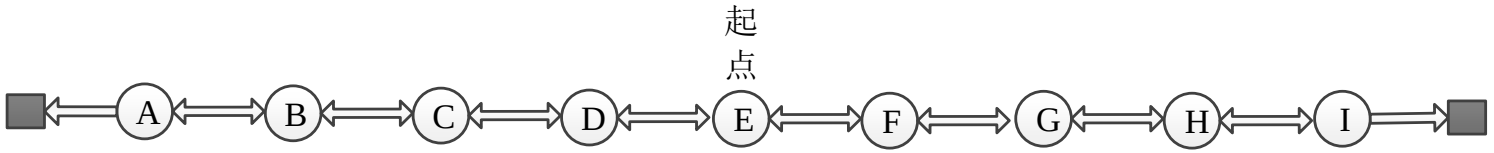
$v_\pi = V$

7.1 *n*-步TD预测及资格迹（33）



- 在11个状态的随机漫步问题中，λ取0与1之间的中间值，效果会更好；
- 选取介于TD(0)和MC之间的TD(λ)可能会得到更好的结果。

基于资格迹的后向在线表格式TD(λ)算法的随机漫步性能图



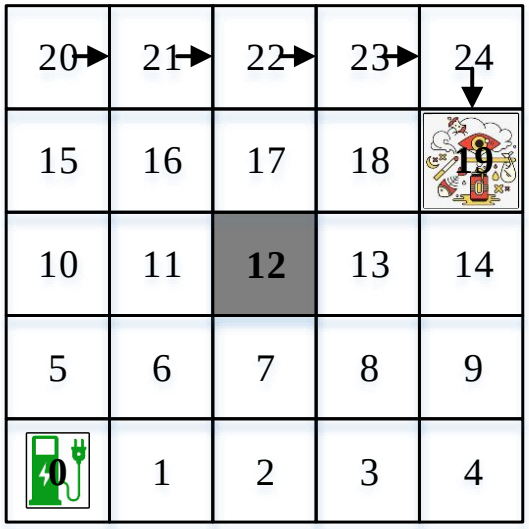
7.1 *n*-步TD预测及资格迹（34）

- 虽然使用资格迹能够在线、增量式地对状态值函数进行更新，但是本节所讨论的依然是针对表格式问题的TD(λ)算法，每一步都需要遍历整个状态空间，对每个状态的资格迹和状态值进行更新。与*n*-步TD方法相比较，该算法的计算复杂度更高；
- Sarsa(λ)算法除了遍历状态空间外，还要遍历整个动作空间，计算成本和内存开销更大；
- 但是在需要使用函数近似方法的大规模问题中，资格迹起着重要的作用。在应用这些算法时，只需要在表格式方法基础上进行微小调整即可，整体流程与表格式方法类似。

7.1 *n*-步TD预测及资格迹（35）

例7.3 ***n*-步TD预测、TD(0)以及TD(λ)算法应用于确定环境扫地机器人问题中的对比分析。**

设Agent所有状态值函数初始值为零，固定 $\alpha = 0.2$ **，令折扣系数** $\gamma = 0.8$ **，假定Agent采取确定性策略。**



7.1 *n*-步TD预测及资格迹（36）

TD(0)更新过程

	S_{20}	S_{21}	S_{22}	S_{23}	S_{24}
V_0	0.000	0.000	0.000	0.000	0.00
V_1	0.000	0.000	0.000	0.000	0.600
V_2	0.000	0.000	0.000		1.080
V_3	0.000	0.000	0.015		1.464
V_4	0.000	0.002	0.052	0.434	1.771
⋮	⋮	⋮	⋮	⋮	⋮
V_{20}	0.455	0.904	1.524	2.234	2.965
V_{21}	0.509	0.967	1.577	2.262	2.972
⋮	⋮	⋮	⋮	⋮	⋮
V_{88}	1.229	1.536	1.920	2.400	3.000
V_{89}	1.229	1.536	1.920	2.400	3.000

7.1 *n*-步TD预测及资格迹（37）

***n*-步TD预测（$n=3$）更新过程**

	S_{20}	S_{21}	S_{22}	S_{23}	S_{24}
V_0	0.000	0.000	0.000	0.000	0.000
V_1	0.000	0.000	0.384	0.480	0.600
V_2	0.049	0.061		0.864	1.080
V_3	0.128	0.160	0.937	1.171	1.464
V_4	0.222	0.278	1.134	1.417	1.771
⋮	⋮	⋮	⋮	⋮	⋮
V_{20}	1.144	1.430	1.898	2.372	2.965
V_{21}	1.158	1.447	1.902	2.378	2.972
⋮	⋮	⋮	⋮	⋮	⋮
V_{53}	1.229	1.536	1.920	2.400	3.000
V_{54}	1.229	1.536	1.920	2.400	3.000

7.1 n-步TD预测及资格迹（38）

TD(λ)（$\lambda = 0.8$）更新过程

	S_{20}	S_{21}	S_{22}	S_{23}	S_{24}
V_0	0.000	0.000	0.000	0.000	0.000
V_1	0.100	0.157	0.246	0.384	0.600
V_2	0.201	0.307		0.710	1.080
V_3	0.299	0.446	0.664	0.987	1.464
V_4	0.393	0.575	0.839	1.220	1.771
⋮	⋮	⋮	⋮	⋮	⋮
V_{20}	1.135	1.451	1.848	2.345	2.965
V_{21}	1.149	1.464	1.860	2.355	2.972
⋮	⋮	⋮	⋮	⋮	⋮
V_{58}	1.229	1.536	1.920	2.400	3.000
V_{59}	1.229	1.536	1.920	2.400	3.000

7.1 *n*-步TD预测及资格迹（39）

- 第1个情节结束后：
  - ✓ TD(0)只更新1个状态的状态值；
  - ✓ *n*-步TD预测算法（$n=3$）更新3个状态的状态值，效率相对有所提升；
  - ✓ TD(λ)算法（$\lambda=0.8$）对Agent所走轨迹上所有状态的状态值进行了更新。

7.1 *n*-步TD预测及资格迹（40）

- 第2个情节结束后：
  - ✓ 由于给定轨迹从起点到终点仅需要5步，3-步TD预测算法对Agent所走轨迹上所有点对应状态的状态值都实现了更新；
  - ✓ TD(0)算法累计只更新了2个状态的状态值；
  - ✓ TD(λ)算法继续对Agent所走轨迹上所有对应状态的状态值进行更新，但在同等步长情况下，由于参数λ的存在，接近垃圾处对应的状态值更新速度，相较3-步TD预测有所下滑。

7.1 *n*-步TD预测及资格迹（41）

- **3种算法收敛情况：**

✓ ***n*-步TD预测算法（$n=3$）与TD(λ)算法（$\lambda=0.8$）收敛迭代轮次几乎相同；**

✓ **两者收敛速度均较TD(0)算法有明显提升。**

小规模问题:

若选取合适的*n*值和λ值，*n*-步TD预测算法与TD(λ)算法性能相近，且均较TD(0)预测算法有明显更高的表现水平；

大规模问题:

TD(λ)算法第1轮更新便能覆盖Agent所走轨迹上的所有点，而*n*-步TD预测可能需要很多轮，此时相比较而言，TD(λ)算法效率会有显著提升。

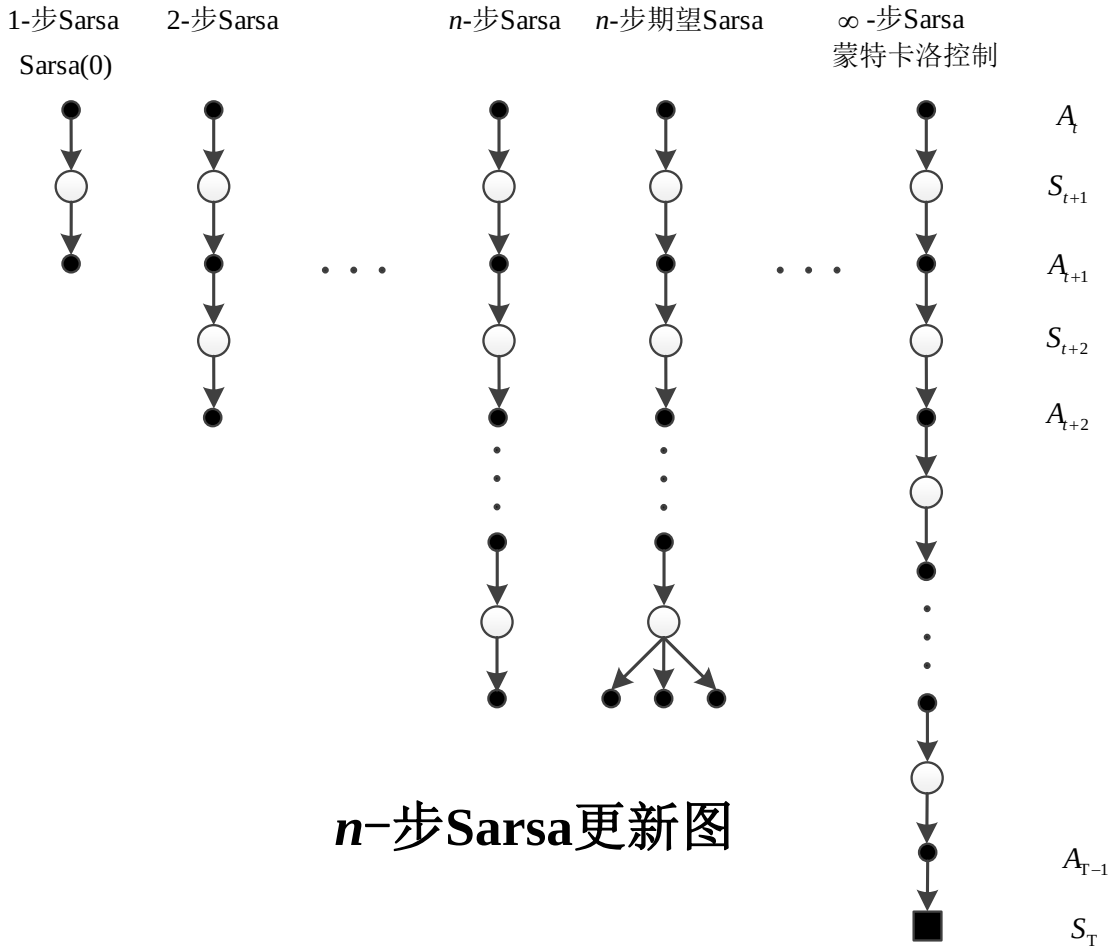
目 录

7.2 *n*-步TD控制及其资格迹实现（1）

- ***n*-步TD控制算法与预测算法一样，分为同策略方法（如*n*-步Sarsa）和异策略方法；**
- **异策略方法包括重要性采样方法、非重要性采样方法（如Tree Backup算法）；**
- **通过引入资格迹还可以构建Sarsa(λ)算法。**

7.2 *n*-步TD控制及其资格迹实现（2）

7.2.1 同策略*n*-步Sarsa算法



***n*−步Sarsa更新图**

***n*-步Sarsa与*n*-步TD更新图类似，但图的首末端都是动作而不是状态。**

7.2 *n*-步TD控制及其资格迹实现（3）

- ***n*-步Sarsa算法目标值（*n*-步回报）：**

$$G_{t:t+n}=\begin{cases}R_{t+1}+\gamma R_{t+2}+\ldots+\gamma^{n-1}R_{t+n}+\gamma^{n}Q_{t+n-1}(S_{t+n},A_{t+n}), & n\geq 1;0\leq t<\mathrm{T}\text{-}n\\ G_t, & t+n\geq \mathrm{T}\end{cases}$$

- ***n*-步Sarsa动作值函数更新迭代式：**

$$Q_{t+n}(S_t,A_t)=Q_{t+n-1}(S_t,A_t)+\alpha\left[G_{t:t+n}-Q_{t+n-1}(S_t,A_t)\right],\qquad 0\leq t<\mathrm{T}$$

除了当前状态-动作对之外，所有其他状态-动作对的值函数估计值都保持不变，即对于所有满足 $s\neq S_t$ 或 $a\neq A_t$ 的 (s,a) 来说，有 $Q_{t+n}(s,a)=Q_{t+n-1}(s,a)$。

7.2 n-步TD控制及其资格迹实现（4）

算法 7.3 用于估算最优策略或固定策略的 n-步 Sarsa 算法

输　入：

折扣因子 γ，学习率 $\{\alpha_k\}_{k=0}^{\infty} \in [0,1]$，$\{\varepsilon_k\}_{k=0}^{\infty} \in [0,1]$，$n \leftarrow$ 正整数

初始化：

1. 对 $s \in \mathcal{S}$，$a \in \mathcal{A}(s)$，初始化动作值函数，如 $Q(s,a) \in \mathbb{R}$；$Q(s^{\mathrm{T}},a) \leftarrow 0$
2. $\pi \leftarrow$ 基于 Q 的 ε -贪婪策略（或固定策略）

3. **repeat** 对每个情节 $k = 0,1,2,\cdots$
4. 初始状态 $S_0 \neq S^{\mathrm{T}}$
5. 根据策略 $\pi(\cdot \mid S_0)$，选择动作 A_0
6. $\mathrm{T} \leftarrow \infty$
7. **repeat** 对每个时间步 $t = 0, 1, 2, \cdots$
8. **if** $t < \mathrm{T}$ **then**
9. 执行动作 A_t，到达下一时刻的状态 S_{t+1}，并获得奖赏 R_{t+1}
10. **if** $S_{t+1} = S^{\mathrm{T}}$ **then**
11. $\mathrm{T} \leftarrow t+1$

7.2 n-步TD控制及其资格迹实现（5）

12. **else**
13. 根据策略 $\pi(\cdot|S_{\tau+1})$，选择动作 $A_{\tau+1}$
14. **end if**
15. **end if**
16. $\tau \leftarrow t-n+1$
17. **if** $\tau \geq 0$ **then**
18. $$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,\mathrm{T})} \gamma^{i-\tau-1} R_i = \begin{cases} R_{\tau+1}+\gamma R_{\tau+2}+\ldots+\gamma^{n-1}R_{\tau+n} & , \quad \tau+n<\mathrm{T} \\ R_{\tau+1}+\gamma R_{\tau+2}+\ldots+\gamma^{\mathrm{T}-\tau-1}R_{\mathrm{T}} & , \quad \tau+n\geq\mathrm{T} \end{cases}$$
19. **if** $\tau+n<\mathrm{T}$ **then**
20. $G \leftarrow G+\gamma^n Q(S_{\tau+n},A_{\tau+n})$
21. **end if**
22. $Q(S_\tau,A_\tau) \leftarrow Q(S_\tau,A_\tau)+\alpha\left[G-Q(S_\tau,A_\tau)\right]$
23. $A^* \leftarrow \arg\max_a Q(S_\tau,a)$

7.2 n-步TD控制及其资格迹实现（6）

24. **for** $a \in \mathcal{A}(S_\tau)$ **do**

25. $$\pi(a \mid S_\tau) \leftarrow \begin{cases} 1-\varepsilon_k+\dfrac{\varepsilon_k}{|\mathcal{A}(S_\tau)|} & a = A^* \\ \dfrac{\varepsilon_k}{|\mathcal{A}(S_\tau)|} & a \neq A^* \end{cases}$$

26. **end for**

27. **end if**

28. **until** $\tau = \mathrm{T}-1$

输　出：

$q_* = Q$，　$\pi_* = \pi$

7.2 *n*-步TD控制及其资格迹实现（7）

- 对于***n*-步Sarsa算法，可以进行适当改进，得到*n*-步期望Sarsa算法；**
- **该算法与*n*-步Sarsa算法的更新图基本一致，区别仅在于*n*-步期望Sarsa更新图最后只有一个分支；**
- **计算*n*-步回报时最后一步需要考虑所有可能的动作，并对其动作值采用策略π下的概率加权求和。**

7.2 n-步TD控制及其资格迹实现（8）

- ***n*-步期望Sarsa的目标值：**

$$G_{t:t+n}=\begin{cases}R_{t+1}+\gamma R_{t+2}+\ldots+\gamma^{n-1}R_{t+n}+\gamma^{n}\bar{V}_{t+n-1}(S_{t+n}), & n\geq 1;0\leq t<\mathrm{T}-n\\ G_t, & t+n\geq \mathrm{T}\end{cases}$$

$\bar{V}_t(s)$：状态 S的期望近似价值，满足：

$$\bar{V}_t(s)=\sum_a \pi(a\mid s)Q_t(s,a)\quad ,\quad s\in\mathcal{S}$$

S 为终止状态时，它的期望近似价值为**0**。

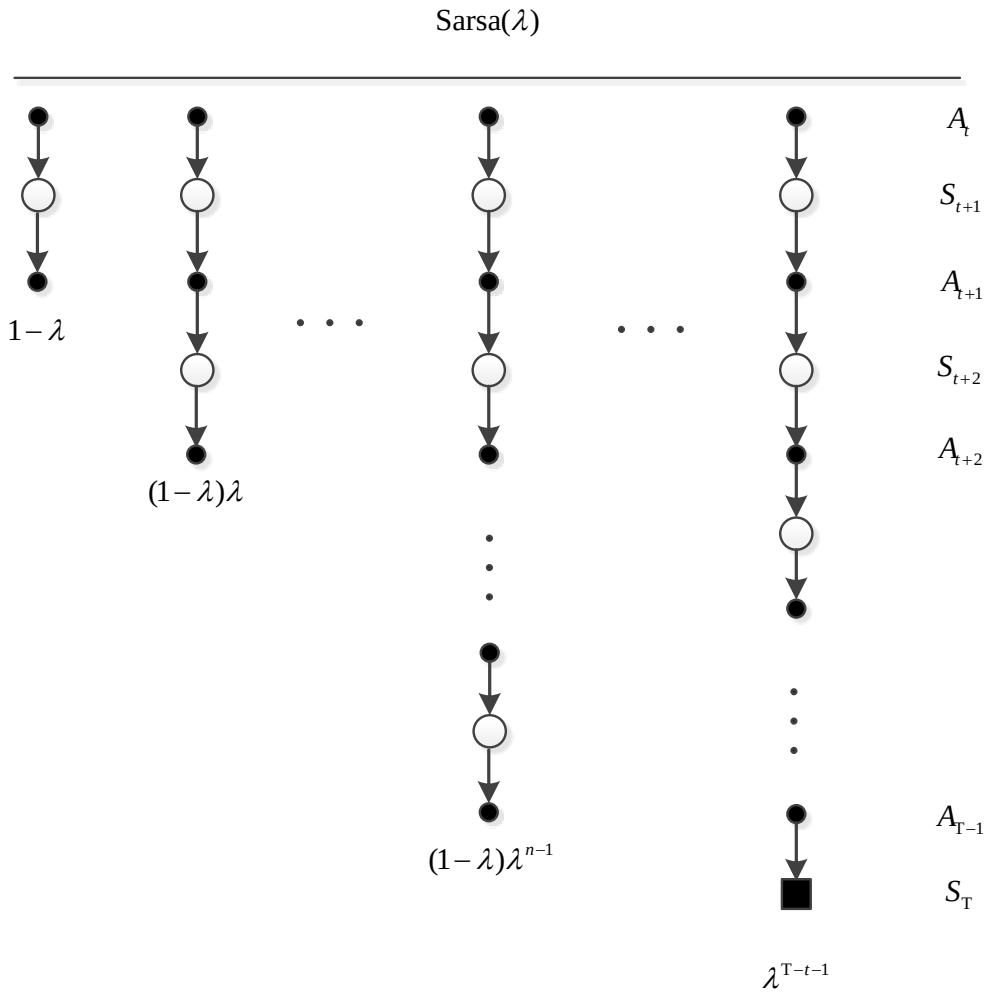
7.2 n-步TD控制及其资格迹实现（9）

7.2.2 Sarsa(λ)算法

- 前向Sarsa(λ)算法基于当前状态-动作对(S_t, A_t)向前看，同样属于同策略方法；
- 基于λ-回报的Sarsa(λ)动作值函数更新递归式：

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[G_t^{\lambda} - Q(S_t, A_t)]$$

7.2 n-步TD控制及其资格迹实现（10）



Sarsa(λ)更新图

- 第1条轨迹向前走完整的一步，到达下一个状态-动作对；
- 第2条轨迹向前走了两步，以此类推，最后一条轨迹对应于完整的情节；
- 每一条轨迹的权重与TD(λ)中的权重一样。

7.2 n-步TD控制及其资格迹实现（11）

- **后向Sarsa(λ)算法：**

✓ 利用资格迹，基于当前状态-动作对(S_t, A_t)向后看；

✓ 考虑已经访问过的，从当前状态–动作对(S_t, A_t) 到初始状态–动作对(S_0, A_0)的所有状态–动作对；

✓ 体现的是各个状态-动作对与当前Agent获得回报变化的因果关系，考虑的是当前状态-动作对以及在此之前发生的状态-动作对，受到该结果的影响程度；

✓ 资格迹分为累积资格迹和替代资格迹。

7.2 n-步TD控制及其资格迹实现（12）

（1）累积资格迹：

- 在状态-动作对 (s,a) 被访问时，资格迹进行累积，然后在该状态-动作对未被访问时逐渐衰减；
- 每走一步所有已访问过的状态的资格迹都会衰减$\gamma\lambda$，当前状态-动作对的资格迹则加1；
- 定义：

$$E_t(s,a)=\begin{cases}\gamma\lambda E_{t-1}(s,a)+1 & s=S_t \text{且} a=A_t \\ \gamma\lambda E_{t-1}(s,a) & \text{其他}\end{cases}$$

$E_t(s,a)$表示 t 时刻状态-动作对 (s,a) 的资格迹。

7.2 n-步TD控制及其资格迹实现（13）

（2）替代资格迹：

- 在状态-动作对(s,a)被访问时，资格迹直接替换为“1”值，然后在该状态-动作对未被访问时逐渐衰减；
- 每走一步所有已访问过的状态的资格迹都会衰减$\gamma\lambda$，当前状态-动作对的资格迹则用“1”直接替代；
- 定义：

$$E_t(s,a)=\begin{cases}1 & s=S_t\text{且}a=A_t\\ \gamma\lambda E_{t-1}(s,a) & \text{其他}\end{cases}$$

7.2 n-步TD控制及其资格迹实现（14）

- 后向Sarsa(λ)算法的动作值函数更新递归式：

$$Q(s,a) \leftarrow Q(s,a) + \alpha\delta_t E_t(s,a)$$

δ_t 同样为1-步TD误差：

$$\delta_t = R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1}) - Q_t(S_t, A_t)$$

- 后向Sarsa(λ)算法可以在线、增量式地学习，无需Agent采集完整的情节，数据用毕即可丢弃。

7.2 n-步TD控制及其资格迹实现（15）

算法 7.4 用于估计最优策略或固定策略的后向在线表格式 Sarsa(λ)算法

输　入：

折扣因子 γ，学习率 $\{\alpha_k\}_{k=0}^{\infty} \in [0,1]$，$\{\varepsilon_k\}_{k=0}^{\infty} \in [0,1]$，迹衰减系数 $\lambda \in [0,1]$

初始化：

1. 对 $s \in \mathcal{S}$，$a \in \mathcal{A}(s)$，初始化动作值函数，如 $Q(s,a) \in \mathbb{R}$；$Q(s^{\mathrm{T}},a) \leftarrow 0$
2. $\pi \leftarrow$ 基于 Q 的 ε -贪婪策略（或固定策略）
3. **repeat** 对每个情节 $k = 0,1,2,\cdots$
4. 　　对 $s \in \mathcal{S}$，$a \in \mathcal{A}(s)$ 初始化 $E(s,a) = 0$
5. 　　初始状态 $S_0 \neq S^{\mathrm{T}}$
6. 　　根据策略 $\pi(\cdot|S)$，选择动作 A
7. 　　$\mathrm{T} \leftarrow \infty$
8. 　　**repeat** 对每个时间步 $t = 0, 1, 2, \cdots$
9. 　　执行动作 A
10. 　　到达下一时刻的状态 S' 并获得奖赏 R
11. 　　$\delta \leftarrow R - Q(S,A)$

7.2 n-步TD控制及其资格迹实现（16）

12. **if** $S' = S^{\mathrm{T}}$ **then**
13. $\mathrm{T} \leftarrow t+1$
14. **else**
15. 根据策略 $\pi(\cdot \mid S')$ 选择动作 A'
16. $\delta \leftarrow \delta + \gamma Q(S', A')$
17. **end if**
18. $E(S,A) \leftarrow E(S,A)+1$（累积资格迹）或 $E(S,A) \leftarrow 1$（替代资格迹）
19. **for** $s \in \mathcal{S}$， $a \in \mathcal{A}(s)$ **do**
20. $Q(s,a) \leftarrow Q(s,a) + \alpha_k \delta E(s,a)$
21. $E(s,a) \leftarrow \gamma\lambda E(s,a)$
22. **end for**
23. $S \leftarrow S'$， $A \leftarrow A'$

7.2 n-步TD控制及其资格迹实现（17）

24. **for** $s \in \mathcal{S}$ **do**
25. $A^* \leftarrow \arg\max_a Q(s,a)$
26. **for** $a \in \mathcal{A}(s)$ **do**
27. $$\pi(a|s) \leftarrow \begin{cases} 1-\varepsilon_k+\dfrac{\varepsilon_k}{|\mathcal{A}(s)|} & a = A^* \\ \dfrac{\varepsilon_k}{|\mathcal{A}(s)|} & a \neq A^* \end{cases}$$
28. **end for**
29. **end for**
30. **until** $t = \mathrm{T}-1$

输　出：

$q_* = Q$，　$\pi_* = \pi$

7.2 n-步TD控制及其资格迹实现（18）

例7.5 将Sarsa(0)、n-步Sarsa以及Sarsa(λ)算法应用于确定环境扫地机器人任务中的对比分析。

- Agent所有动作值函数初始值均为0；
- 初始$\alpha_0 = 0.05$，折扣系数 $\gamma = 0.8$，并采用 ε-greedy策略（ $\varepsilon_0 = 0.5$ ）。

7.2 n-步TD控制及其资格迹实现（19）

Sarsa(0)更新过程

	S_5	S_{10}	S_{18}	S_{20}	S_{24}
Q_0	0.00;0.00;*.**;0.00	0.00;0.00;*.**;0.00	0.00;0.00;0.00;0.00	*.**;0.00;*.**;0.00	*.**;0.00;0.00;*.**
π_0	0.67;0.17;0.00;0.17	0.67;0.17;0.00;0.17	0.63;0.12;0.12;0.12	0.00;0.75;0.00;0.25	0.00;0.75;0.25;0.00
Q_1	0.00;0.00;*.**;0.00	0.00;0.00;*.**;0.00	0.00;0.00;0.00;0.15	*.**;0.00;*.**;0.00	*.**;0.00;0.00;*.**
π_1	0.67;0.17;0.00;0.17	0.67;0.17;0.00;0.17	0.12;0.12;0.12;0.63	0.00;0.75;0.00;0.25	0.00;0.75;0.25;0.00
⋮	⋮	⋮	⋮	⋮	⋮
Q_{7500}	0.35;1.00;*.**;0.40	0.38;0.69;*.**;-0.22	1.65;0.86;1.09;3.00	*.**;0.48;*.**;0.83	*.**;3.00;1.72;*.**
π_{7500}	0.10;0.79;0.00;0.10	0.10;0.79;0.00;0.10	0.08;0.08;0.08;0.77	0.00;0.16;0.00;0.84	0.00;0.84;0.16;0.00
⋮	⋮	⋮	⋮	⋮	⋮

7.2 n-步TD控制及其资格迹实现（20）

Sarsa(0)更新过程

Q_{12500}	0.40;1.00;*.**;0.40	0.44;0.70;*.**;-0.58	1.72;0.79;1.36;3.00	*.**;0.57;*.**;1.04	*.**;3.00;1.71;*.**
π_{12500}	0.06;0.88;0.00;0.06	0.06;0.88;0.00;0.06	0.05;0.05;0.05;0.86	0.00;0.09;0.00;0.91	0.00;0.91;0.09;0.00
⋮	⋮	⋮	⋮	⋮	⋮
Q_{19999}	0.40;1.00;*.**;0.40	0.44;0.73;*.**;-0.55	1.75;0.88;1.47;3.00	*.**;0.66;*.**;1.20	*.**;3.00;1.85;*.**
π_{19999}	0.00;1.00;0.00;0.00	0.00;1.00;0.00;0.00	0.00;0.00;0.00;1.00	0.00;0.00;0.00;1.00	0.00;1.00;0.00;0.00
Q_{20000}	0.40;1.00;*.**;0.40	0.44;0.73;*.**;-0.55	1.75;0.88;1.47;3.00	*.**;0.66;*.**;1.20	*.**;3.00;1.85;*.**
π_{20000}	0.00;1.00;0.00;0.00	0.00;1.00;0.00;0.00	0.00;0.00;0.00;1.00	0.00;0.00;0.00;1.00	0.00;1.00;0.00;0.00
π_*	0.00;1.00;0.00;0.00	0.00;1.00;0.00;0.00	0.00;0.00;0.00;1.00	0.00;0.00;0.00;1.00	0.00;1.00;0.00;0.00

7.2 n-步TD控制及其资格迹实现（21）

n-步Sarsa（$n=3$）更新过程

	S_5	S_{10}	S_{18}	S_{20}	S_{24}
Q_0	0.00;0.00;*.**;0.00	0.00;0.00;*.**;0.00	0.00;0.00;0.00;0.00	*.**;0.00;*.**;0.00	*.**;0.00;0.00;*.**
π_0	0.67;0.17;0.00;0.17	0.67;0.17;0.00;0.17	0.63;0.12;0.12;0.12	0.00;0.75;0.00;0.25	0.00;0.75;0.25;0.00
Q_1	0.00;0.00;*.**;0.00	0.00;0.00;*.**;0.00	0.00;0.00;0.00;0.15	*.**;0.00;*.**;0.00	*.**;0.00;0.00;*.**
π_1	0.67;0.17;0.00;0.17	0.67;0.17;0.00;0.17	0.12;0.12;0.12;0.63	0.00;0.75;0.00;0.25	0.00;0.75;0.25;0.00
⋮	⋮	⋮	⋮	⋮	⋮
Q_{7500}	0.17;1.00;*.**;0.32	0.37;0.70;*.**;-0.51	1.72;0.46;1.10;3.00	*.**;0.49;*.**;0.95	*.**;3.00;1.76;*.**
π_{7500}	0.10;0.79;0.00;0.10	0.10;0.79;0.00;0.10	0.08;0.08;0.08;0.77	0.00;0.16;0.00;0.84	0.00;0.84;0.16;0.00
⋮	⋮	⋮	⋮	⋮	⋮

7.2 n-步TD控制及其资格迹实现（22）

n-步Sarsa（$n=3$）更新过程

Q_{12500}	0.30;1.00;*.**;0.21	0.40;0.72;*.**;-0.49	1.72;1.03;1.00;3.00	*.**;0.61;*.**;1.07	*.**;3.00;1.74;*.**
π_{12500}	0.06;0.88;0.00;0.06	0.06;0.88;0.00;0.06	0.05;0.05;0.05;0.86	0.00;0.09;0.00;0.91	0.00;0.91;0.09;0.00
⋮	⋮	⋮	⋮	⋮	⋮
Q_{19999}	0.31;1.00;*.**;0.22	0.40;0.74;*.**;-0.47	1.73;1.07;1.21;3.00	*.**;0.71;*.**;1.21	*.**;3.00;1.86;*.**
π_{19999}	0.00;1.00;0.00;0.00	0.00;1.00;0.00;0.00	0.00;0.00;0.00;1.00	0.00;0.00;0.00;1.00	0.00;1.00;0.00;0.00
Q_{20000}	0.31;1.00;*.**;0.22	0.40;0.74;*.**;-0.47	1.73;1.07;1.21;3.00	*.**;0.71;*.**;1.21	*.**;3.00;1.86;*.**
π_{20000}	0.00;1.00;0.00;0.00	0.00;1.00;0.00;0.00	0.00;0.00;0.00;1.00	0.00;0.00;0.00;1.00	0.00;1.00;0.00;0.00
π_*	0.00;1.00;0.00;0.00	0.00;1.00;0.00;0.00	0.00;0.00;0.00;1.00	0.00;0.00;0.00;1.00	0.00;1.00;0.00;0.00

7.2 n-步TD控制及其资格迹实现（23）

Sarsa(λ)算法（$\lambda = 0.8$）更新过程

	S_5	S_{10}	S_{18}	S_{20}	S_{24}
Q_0	0.00;0.00;*.**;0.00	0.00;0.00;*.**;0.00	0.00;0.00;0.00;0.00	*.**;0.00;*.**;0.00	*.**;0.00;0.00;*.**
π_0	0.67;0.17;0.00;0.17	0.67;0.17;0.00;0.17	0.63;0.12;0.12;0.12	0.00;0.75;0.00;0.25	0.00;0.75;0.25;0.00
Q_1	0.00;0.00;*.**;0.00	0.00;0.00;*.**;0.00	0.00;0.00;0.00;0.15	*.**;-0.07;*.**;-0.11	*.**;0.00;0.00;*.**
E_1	0.00;0.00;*.**;0.00	0.00;0.00;*.**;0.00	0.00;0.00;0.00;0.64	*.**;0.04;*.**;0.07	*.**;0.00;0.00;*.**
π_1	0.67;0.17;0.00;0.17	0.17;0.67;0.00;0.17	0.12;0.12;0.12;0.63	0.00;0.75;0.00;0.25	0.00;0.75;0.25;0.00
⋮	⋮	⋮	⋮	⋮	⋮
Q_{7500}	0.41;1.00;*.**;0.44	0.40;0.72;*.**;-0.38	1.69;0.55;1.02;3.00	*.**;0.50;*.**;0.94	*.**;3.00;1.75;*.**
E_{7500}	0.00;0.00;*.**;0.00	0.00;0.00;*.**;0.00	0.00;0.00;0.00;0.64	*.**;0.00;*.**;0.06	*.**;0.00;0.26;*.**
π_{7500}	0.10;0.79;0.00;0.10	0.10;0.79;0.00;0.10	0.08;0.08;0.08;0.77	0.00;0.16;0.00;0.84	0.00;0.84;0.16;0.00
⋮	⋮	⋮	⋮	⋮	⋮

7.2 n-步TD控制及其资格迹实现（24）

Sarsa(λ)算法（$\lambda = 0.8$）更新过程

Q_{12500}	0.47;1.00;*.**;0.44	0.43;0.73;*.**;-0.41	1.72;1.05;1.46;3.00	*.**;0.58;*.**;1.07	*.**;3.00;1.74;*.**
E_{12500}	0.00;0.00;*.**;0.00	0.00;0.00;*.**;0.00	0.00;0.00;0.00;0.00	*.**;0.04;*.**;0.11	*.**;0.64;0.00;*.**
π_{12500}	0.06;0.88;0.00;0.06	0.06;0.88;0.00;0.06	0.05;0.05;0.05;0.86	0.00;0.09;0.00;0.91	0.00;0.91;0.09;0.00
⋮	⋮	⋮	⋮	⋮	⋮
Q_{19999}	0.47;1.00;*.**;0.44	0.43;0.75;*.**;-0.39	1.73;1.08;1.52;3.00	*.**;0.71;*.**;1.21	*.**;3.00;1.85;*.**
E_{19999}	0.00;0.00;*.**;0.00	0.00;0.00;*.**;0.00	0.00;0.00;0.00;0.00	*.**;0.00;*.**;0.11	*.**;0.64;0.00;*.**
π_{19999}	0.00;1.00;0.00;0.00	0.00;1.00;0.00;0.00	0.00;0.00;0.00;1.00	0.00;0.00;0.00;1.00	0.00;1.00;0.00;0.00
Q_{20000}	0.47;1.00;*.**;0.44	0.43;0.75;*.**;-0.39	1.73;1.08;1.52;3.00	*.**;0.71;*.**;1.21	*.**;3.00;1.85;*.**
E_{20000}	0.00;0.00;*.**;0.00	0.00;0.00;*.**;0.00	0.00;0.00;0.00;0.00	*.**;0.00;*.**;0.11	*.**;0.64;0.00;*.**
π_{20000}	0.00;1.00;0.00;0.00	0.00;1.00;0.00;0.00	0.00;0.00;0.00;1.00	0.00;0.00;0.00;1.00	0.00;1.00;0.00;0.00
π_*	0.00;1.00;0.00;0.00	0.00;1.00;0.00;0.00	0.00;0.00;0.00;1.00	0.00;0.00;0.00;1.00	0.00;1.00;0.00;0.00

7.2 n-步TD控制及其资格迹实现（25）

- **第1个情节后：**
- ✓ 状态 S_{20} 中向下及向右动作，**Sarsa(λ)**（$\lambda=0.8$）算法算得其对应动作值均为负值，说明该情节随机所得轨迹中到达障碍物的信息，在该算法中被反馈给较远的位置，比如左上角处状态上；
- ✓ 同样轨迹下，其他2个算法则并未对这些动作值进行更新；
- ✓ 状态 S_{10} 处各动作值， **Sarsa(λ)**（$\lambda=0.8$）算法计算结果，同样与其他算法存在差异；
- ✓ 其余所列状态，各动作对应动作值，3个算法求解情况基本一致；
- ✓ **Sarsa(λ)算法在起始阶段，可以更充分地对Agent所走轨迹信息进行利用并高效反馈。**

7.2 *n*-步TD控制及其资格迹实现（26）

- 此后所列情节：

✓ 3个算法计算所得动作值存在一定差异，而所得策略基本一致，说明策略可以较动作值更快收敛；

✓ 考虑动作值差异，对比3种算法，可以看出，相较Sarsa(0)算法、*n*−步Sarsa（$n=3$ ）与Sarsa(λ)算法（$\lambda=0.8$ ）在实际较优轨迹上动作值学习较快；

✓ 经历了较多情节后，Sarsa(λ)学习效果并不一定较*n*-步Sarsa更优；

对于Sarsa(λ)，还需要进行资格迹的计算，空间和时间成本相较其他算法会有进一步提升。

7.2 *n*-步TD控制及其资格迹实现（27）

7.2.3 异策略*n*-步Sarsa算法

- 基于重要性采样的异策略*n*-步TD算法，不但需要考虑*n*-步回报，而且需要关注*n*步中两个策略采取动作的相对概率。
- 基于重要性采样的异策略*n*-步TD预测算法的状态值函数更新递归式：

$$V_{t+n}(S_t)=V_{t+n-1}(S_t)+\alpha\rho_{t:t+n-1}\left[G_{t:t+n}-V_{t+n-1}(S_t)\right] \qquad 0\le t<\mathrm{T}$$

7.2 n-步TD控制及其资格迹实现（28）

- **重要性权重 $\rho_{t:t+n-1}$：**
  - ✓ 两种策略采取 $A_t \sim A_{t+n}$这n个动作的相对概率；
  - ✓ 计算公式：

$$\rho_{t:h} = \prod_{k=t}^{\min(h,\mathrm{T}-1)} \frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)}$$

7.2 n-步TD控制及其资格迹实现（29）

- $\rho_{t:h}$ ：

✓ 当遵循目标策略π，某状态-动作对(S_k, A_k)永远不会遇到时，$\pi(A_k \mid S_k)=0$，$\rho_{t:h}=0$，即忽略此次更新；

✓ 在任意状态S_k下，当遵循目标策略π选择动作A_k的概率，远大于遵循行为策略b选择动作A_k的概率时，$\rho_{t:h}$远大于1；

✓ 在任意状态S_k下，当遵循目标策略π选择动作A_k的概率，远小于遵循行为策略b选择动作A_k的概率时，情况反之；

✓ 当目标策略与行为策略的效果一样时，$\rho_{t:h}$恒为1。

7.2 *n*-步TD控制及其资格迹实现（30）

- 异策略***n*-步Sarsa动作值函数更新递归式：**

$$Q_{t+n}(S_t,A_t)=Q_{t+n-1}(S_t,A_t)+\alpha\rho_{t+1:t+n-1}\left[G_{t:t+n}-Q_{t+n-1}(S_t,A_t)\right],\qquad 0\le t<\mathrm{T}$$

与预测问题相比，这里ρ的起始下标要晚一步。这是因为在异策略***n*-步Sarsa**算法中，t时刻的动作A_t已经确定，重要性采样仅需应用到后续动作即可，即从$t+1$时刻开始应用，所以该算法使用$\rho_{t+1:t+n-1}$作为重要性权重。

7.2 n-步TD控制及其资格迹实现（31）

算法 7.5 基于动作值函数的异策略 n-步 Sarsa 算法

输　入：

对所有 $s \in \mathcal{S}$， $a \in \mathcal{A}(s)$，行为策略 $b(s,a) > 0$

折扣因子 γ，学习率 $\{\alpha_k\}_{k=0}^{\infty} \in [0,1]$，$n \leftarrow$ 正整数

初始化：

1. 对 $s \in \mathcal{S}$， $a \in \mathcal{A}(s)$，初始化动作值函数，如 $Q(s,a) \in \mathbb{R}$；$Q(s^{\mathrm{T}},a) \leftarrow 0$
2. $\pi \leftarrow$ 基于 Q 的贪婪策略（或固定策略）
3. **repeat** 对每个情节 $k = 0,1,2,\cdots$
4. 初始状态 $S_0 \neq S^{\mathrm{T}}$
5. 根据策略 $b(\cdot \mid S_0)$，选择动作 A_0
6. $\mathrm{T} \leftarrow \infty$
7. **repeat** 对每个时间步 $t = 0, 1, 2, \cdots$
8. **if** $t < T$ **then**
9. 执行动作 A_t
10. 到达下一时刻的状态 S_{t+1} 并获得奖赏 R_{t+1}
11. **if** S_{t+1} 为终止状态 **then**
12. $\mathrm{T} \leftarrow t+1$

7.2 n-步TD控制及其资格迹实现（32）

13. **else**
14. 根据策略 $b(\cdot \mid S_{t+1})$，选择动作 A_{t+1}
15. **end if**
16. **end if**
17. $\tau \leftarrow t-n+1$
18. **if** $\tau \geq 0$ **then**
19. $\rho \leftarrow \prod_{i=\tau+1}^{\min(\tau+n-1,T-1)} \frac{\pi(A_i \mid S_i)}{b(A_i \mid S_i)}$
20. $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
21. **if** $\tau+n<T$ **then**
22. $G \leftarrow G+\gamma^n Q(S_{\tau+n}, A_{\tau+n})$
23. **end if**
24. $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau)+\alpha_k \rho\left[G-Q(S_\tau, A_\tau)\right]$
25. $A^* \leftarrow \arg\max_a Q(S_\tau, a)$

7.2 n-步TD控制及其资格迹实现（33）

26. **for** $a \in \mathcal{A}(S_\tau)$ **do**

27. $$\pi(a \mid S_\tau) \leftarrow \begin{cases} 1 & a = A^* \\ 0 & a \neq A^* \end{cases}$$

28. **end for**

29. **end if**

30. **until** $\tau = \mathrm{T} - 1$

输　出：

$q_* = Q$，　$\pi_* = \pi$

7.2 *n*-步TD控制及其资格迹实现（34）

异策略***n*-步期望Sarsa**算法的值函数更新递归式是在异策略***n*-步Sarsa**算法的基础上，将重要性权重 $\rho_{\tau+1:\tau+n-1}$ 转变为 $\rho_{\tau+1:\tau+n-2}$ 。

原因：异策略***n*-步期望Sarsa**算法的最后一个状态需要考虑所有可能的动作，这样就不需要使用重要性采样了。

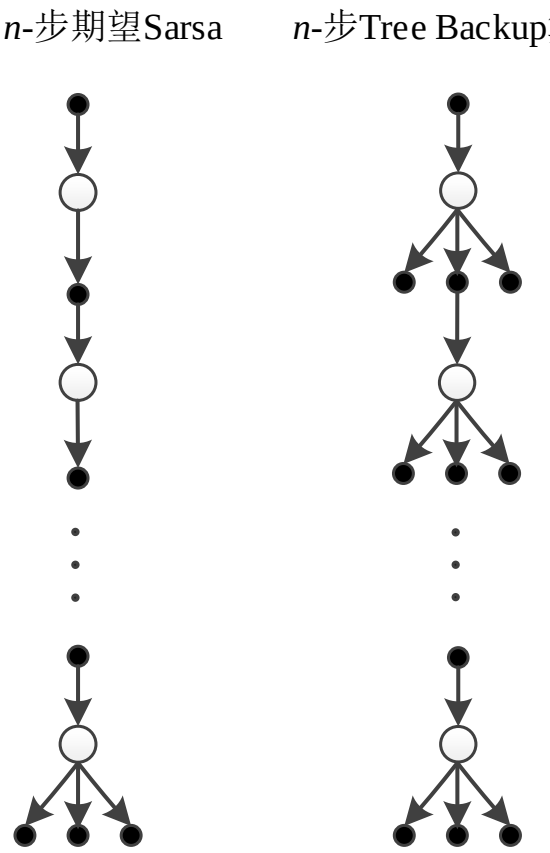
7.2 *n*-步TD控制及其资格迹实现（35）

7.2.4 *n*-步Tree Backup算法

- 重要性采样比：
  - ✓ 在修正行为策略和目标策略差异方面起着重要作用；
  - ✓ 但是具有高度变化的性质，且在考虑*n*个时间步的情况下，计算得到重要性权重变化更大，方法存在着严重的高方差问题。

不使用重要性采样的*n*-步TD方法：*n*-步Tree Backup算法。

7.2 n-步TD控制及其资格迹实现（36）



（1）中轴线为采样轨迹，从根结点(S_t, A_t)开始；

（2）主线实心圆点（不包括最后一层）表示已选动作A_t，A_{t+1}，$\cdots$，侧枝实心圆点（包括最后一层的实心原点）表示未选动作。

n-步Tree Backup算法与n-步期望Sarsa算法的更新图

7.2 *n*-步TD控制及其资格迹实现（37）

***n*-步回报** $G_{t:t+n}$：

- $n=1$时，***n*-步Tree Backup算法与期望Sarsa算法等价**；
- $n>1$ 时，总回报由3部分组成：

（1）从根结点出发，观察获得的奖赏R_{t+1}；

（2）到达下一状态 S_{t+1} 后，树开始分支，考虑侧枝实心圆点，这些状态-动作对，当前情节采样并没有取到，直接用对应动作值 $Q(S_{t+1},a)$实现截断操作，并分配权重 $\pi(a|S_{t+1})$，求得值 $\sum_{a\neq A_{t+1}}\pi(a|S_{t+1})Q_{t+n+1}(S_{t+1},a)$；

（3）考虑中轴线上部分，沿其到达下一结点，此时即面临“根结点为 (S_{t+1},A_{t+1})，求解**(*n*-1)**-步回报$G_{t+1:t+n}$”这样的子问题，对这样的回报分配权重 $\pi(A_{t+1}|S_{t+1})$，得 $\pi(A_{t+1}|S_{t+1})G_{t+1:t+n}$ 。

7.2 *n*-步TD控制及其资格迹实现（38）

- 考虑折扣系数的情况下，***n*-步回报的定义**：

$$\begin{cases} G_{t:t+1} = R_{t+1} + \gamma \sum_{a} \pi(a \mid S_{t+1}) Q_t(S_{t+1}, a), & t < \mathrm{T}-1, n = 1 \\ G_{t:t+n} = R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a \mid S_{t+1}) Q_{t+n+1}(S_{t+1}, a) + \gamma \pi(A_{t+1} \mid S_{t+1}) G_{t+1:t+n}, & t < \mathrm{T}-1, n > 1 \end{cases}$$

- ***n*-步Tree Backup算法的值函数更新递归式**：

$$Q_{t+n}(S_t, A_t) = Q_{t+n-1}(S_t, A_t) + \alpha \left[G_{t:t+n} - Q_{t+n-1}(S_t, A_t) \right], \qquad 0 \le t < \mathrm{T}$$

除了当前状态-动作对 (S_t, A_t) 之外，其他所有状态-动作对的值函数估计值都保持不变，即对于所有满足 $s \neq S_t$ 或 $a \neq A_t$ 的 (s, a) 来说，$Q_{t+n}(s,a) = Q_{t+n-1}(s,a)$ 。

7.2 n-步TD控制及其资格迹实现（39）

算法 7.6 基于动作值函数的 n-步 Tree Backup 算法

输　入：

对所有 $s \in \mathcal{S}$， $a \in \mathcal{A}(s)$，行为策略 $b(s,a) > 0$

折扣因子 γ，学习率 $\{\alpha_k\}_{k=0}^{\infty} \in [0,1]$， $\{\varepsilon_k\}_{k=0}^{\infty} \in [0,1]$， $n \leftarrow$ 正整数

初始化：

1. 对 $s \in \mathcal{S}$， $a \in \mathcal{A}(s)$，初始化动作值函数，如 $Q(s,a) \in \mathbb{R}$； $Q(s^{\mathrm{T}},a) \leftarrow 0$
2. $\pi \leftarrow$ 基于 Q 的贪婪策略（或固定策略）

3. **repeat** 对每个情节 $k = 0,1,2,\cdots$
4. 初始状态 $S_0 \neq S^{\mathrm{T}}$
5. 根据策略 $b(\cdot \mid S_0)$，选择动作 A_0
6. $\mathrm{T} \leftarrow \infty$
7. **repeat** 对每个时间步 $t = 0, 1, 2, \cdots$
8. **if** $t < \mathrm{T}$ **then**
9. 执行动作 A_t
10. 到达下一时刻的状态 S_{t+1}，并获得奖赏 R_{t+1}
11. **if** $S_{t+1} = S^{\mathrm{T}}$ **then**
12. $\mathrm{T} \leftarrow t+1$

7.2 n-步TD控制及其资格迹实现（40）

13. **else**
14. 根据策略 $b(\cdot|S_{t+1})$，选择动作 A_{t+1}
15. **end if**
16. **end if**
17. $\tau \leftarrow t-n+1$
18. **if** $\tau \geq 0$ **then**
19. **if** $t+1 \geq \mathrm{T}$ **then**
20. $G \leftarrow R_T$
21. **else**
22. $G \leftarrow R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$
23. **end if**
24. **for** $k = \min(t, \mathrm{T}-1)$ **downto** $\tau+1$ **do**
25. $G \leftarrow R_k + \gamma \sum_{a \neq A_k} \pi(a|S_k) Q(S_k, a) + \gamma \pi(A_k|S_k) G$
26. **end for**
27. $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha_k \left[G - Q(S_\tau, A_\tau) \right]$
28. $A^* \leftarrow \arg\max_a Q(S_\tau, a)$

7.2 n-步TD控制及其资格迹实现（41）

29. **for** $a \in \mathcal{A}(S_\tau)$ **do**

30. $$\pi(a \mid S_\tau) \leftarrow \begin{cases} 1 & a = A^* \\ 0 & a \neq A^* \end{cases}$$

31. **end for**

32. **end if**

33. **until** $\tau = \mathrm{T} - 1$

输　出：

$q_* = Q$，　$\pi_* = \pi$

7.2 *n*-步TD控制及其资格迹实现（42）

- ***n*-步Tree Backup算法的核心思想在于平衡探索与利用问题；**
- **与*n*-步Sarsa或者*n*-步期望Sarsa算法相比，*n*-步Tree Backup算法考虑的动作更全面，有效拓展了采样数据，进一步提升了动作值函数的利用率；**
- **与利用行为策略的采样方式不同，*n*-步Tree Backup算法对于动作的选取是随机的，在更新目标值时，也仅仅考虑目标策略。**

目 录

7.3 小结（1）

- 本章介绍了几种*n*-步TD算法，其更新时需要利用未来*n*个奖赏、动作以及状态。相比位于两个“极端”的TD(0)和MC，*n*-步TD通常表现更好。
- 本章首先详细阐述了*n*-步TD预测、同策略*n*-步Sarsa算法。这两种算法，与单步TD算法相比较，虽然总体性能更优，但是Agent与环境交互需要延迟*n*个时间步进行更新，需要更多存储空间，每步需要计算量更大。对于这些问题，资格迹可以适当解决，尤其当后续章节面对需要应用函数逼近解决的更大规模问题，或者Agent面临连续式任务时，应用资格迹效果明显更好。

7.3 小结（2）

- 本章还介绍了应用资格迹之后得到的后向TD(λ)算法以及Sarsa(λ)算法。关于异策略，介绍了利用重要性采样的异策略n-步Sarsa算法以及不利用重要性采样的n-步Tree Backup算法。
- 前者理解相对简单，但是方差较大，算法学习速度较慢，收敛所需时间较长，尤其当行为策略和目标策略相差较大时，算法低效且不实用。后者没有利用重要性采样，算法效果稍优，但计算量偏大。

7.3 小结（3）

- 本章介绍的这些算法均应用在较小规模、表格式求解的强化学习问题方面。尽管如此，实际情景中更为普遍的大规模强化学习问题，比如深度强化学习领域，在使用神经网络进行值函数逼近时，*n*-步TD方法仍然起着重要作用。例如，DQN算法可以应用多步学习进行改进，从而提升性能；A3C算法中评论家估计动作值时，也用到多步自举。因此，本章内容为后续研究奠定了非常重要的基础。

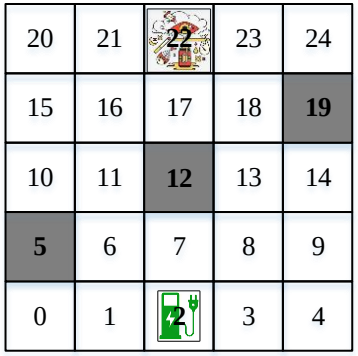
7.4 习题（1）

1.分别给出前向TD(λ)和后向TD(λ)算法的值函数更新公式。

2.什么是资格迹？在强化学习中，有哪几种资格迹？

3.（编程）通过*n*-TD算法计算：第3章习题2（图3.12）扫地机器人在折扣率、初始策略为等概率策略的情况下，分别在*n*为1步、2步、n步时，计算每个状态的最优策略，并比较对于不同的*n*，*n*-TD算法的性能。

4. 在例7.1随机漫步任务中，使用*n*-TD算法，如果用19个状态取代9个状态会产生什么样的训练效果？



The End