第5章

蒙特卡洛法

动态规划法属于基于模型的 MDP 问题求解方法。在环境模型已知的情况下，动态规划法不需要对环境采样，只需要通过迭代计算，就可以得到问题的最优策略。然而在实际任务中，完整的环境模型通常是难以获取的。无模型的强化学习，状态转移概率是未知的，也就是说无模型强化学习无法利用动态规划方法来求解值函数。不妨尝试通过值函数的原始定义来求解无模型强化学习问题：

$$v_\pi(s)=\mathbb{E}_\pi(G_t|S_t=s)$$

$$q_\pi(s,a)=\mathbb{E}_\pi(G_t|S_t=s,A_t=a)$$

本章将主要介绍如何利用**蒙特卡洛法**(Monte Carlo，MC)来求解无模型强化学习问题，以 GPI 来保证其最优性。这里首先考虑预测问题中的策略评估，然后再考虑控制问题中的策略改进。

在线视频 19

5.1 蒙特卡洛法的基本概念

在实际问题中，通常不易获得完整的环境知识。例如在 21 点游戏中，只根据玩家已知的牌面，无法直接获得状态转移概率。相对而言，采样数据则更容易获得。例如可以多次进行游戏对弈，然后估计专家的叫牌和停牌套路，计算出最优的游戏策略。这种基于统计学的思想，通过大量采样获取数据来进行学习的方法称为经验方法。MC 正是基于经验方法，在环境模型未知的情况下，采用时间步有限的、完整的情节，根据经验进行学习，并通过平均采样回报来解决强化学习问题。

5.1.1 MC 的核心要素

(1) **经验**(experience)：经验是从环境交互中获得的(s,a,r)序列，它是情节的集合，

也就是样本集。经验既可以是**真实经验**(real experience),也可以是**模拟经验**(simulator experience)。其中模拟经验是通过**模拟模型**(simulated model)得到的,这里的模拟模型只需要生成状态转移的一些样本,不需要像 DP 那样需要环境中所有可能的状态转移概率。

(2) **情节**(episode):一段经验可以分为多个情节,每一情节都是一个完整的(s,a,r)序列,即必有终止状态,形如$s_0,a_0,r_1,\cdots,s_{T-1},a_{T-1},r_T,s_T$。经常与情节混淆的概念是**轨迹**(trajectory),轨迹可以没有终止状态,形如$s_0,a_0,r_1,s_1,a_1,r_2,\cdots$。所以通常情况下,可以认为:

$$\text{序列} \subseteq \text{情节} \subseteq \text{经验(轨迹)}$$

(3) **完整回报**(full return)与目标值:因为只有到达终止状态才能计算回报,所以将情节的回报G_t称为完整回报。此外,G_t也称为 MC 的目标值。

5.1.2　MC 的特点

(1) 不需要知道状态转移概率p,直接从环境中进行采样来处理无模型任务。

(2) 利用情节进行学习,并采用**情节到情节**(episode-by-episode)的**离线学习**(off-line)方式来求解最优策略π_*。与之相对的,DP 和后续介绍的时序差分算法则采用**步到步**(step-by-step)的**在线学习**(on-line)方式来求解最优策略。

- 离线学习:先完整地采集数据,然后以离线方式优化学习目标。
- 在线学习:边采集数据边优化学习目标。

(3) MC 是一个非平稳问题,其表现在:某个状态采取动作之后的回报,取决于在同一个情节内后续状态所采取的动作,而这些动作通常是不确定的。如果说 DP 是在 MDP 模型中计算值函数,那么 MC 就是学习值函数。

(4) 在 MC 中,对每个状态值函数的估计都是独立的。也就是说,对状态的值函数估计不依赖于其他任何状态,这也说明了 MC 不是自举过程。

(5) MC 在估计每个状态的值函数时,其计算成本与状态总数无关,因为它只需要计算指定状态的回报,不需要考虑所有的状态。

实际上,MC 泛指任何包含大量随机成分的估计方法,通常利用采样数据来估算某一事件的发生概率。在数学领域中,它的应用可以用例 5.1 来说明。

例 5.1　在边长为 1 米的正方形S_1内构建一个扇形S_2,如图 5.1 所示。利用扇形面积计算公式,可以计算出$S_2=\frac{1}{4}\pi r^2\approx\frac{1}{4}\times3.14\times1^2=0.785$。现在利用 MC 方法计算$S_2$的面积。均匀地向$S_1$内撒$n$个黄豆,经统计得知有$m$个黄豆在$S_2$内部,那么有$\frac{S_2}{S_1}\approx\frac{m}{n}$,即$S_2\approx\frac{m}{n}S_1$,且$n$越大,计算得到的面积越精确。

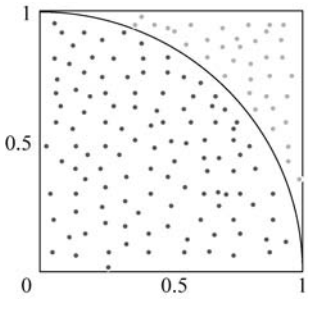

图 5.1　用 MC 方法计算扇形面积(见彩插)

在程序中分别设置$n=100$、10000 和 1000000 共 3 组数据,统计结果如表 5.1 所示。

表 5.1 MC 方法计算扇形面积结果统计表

n	S_1	S_2	m	$\frac{m}{n}$
100	1.0	0.785	79	0.79
10000	1.0	0.785	7839	0.7839
1000000	1.0	0.785	786018	0.786018

由实验数据可知，当 n 值越大时，得到的扇形面积越精确，即接近 0.785。

由此获得的 MC 采样公式为

$$\mathbb{E}_{x\sim p}[f(x)]=\sum_x p(x)f(x)\approx\frac{1}{N}\sum_{x_i\sim p,i=1}^{N}f(x_i) \tag{5.1}$$

在线视频 20

5.2 蒙特卡洛预测

根据状态值函数的初始定义，MC 预测算法以情节中初始状态 s 的回报期望作为其值函数 $v_\pi(s)$的估计值，对策略 π 进行评估。在求解状态 s 的值函数时，先利用策略 π 产生 n 个情节 $s_0,a_0,r_1,\cdots,s_{T-1},a_{T-1},r_T,s_T$，然后计算每个情节中状态 s 的折扣回报：

$$G_t^{(i)}=r_{t+1}+\gamma r_{t+2}+\cdots+\gamma^{T-t-1}r_T$$

这里，$G_t^{(i)}$ 表示在第 i 个情节中，从 t 时刻到终点时刻 T 的回报。该回报是基于某一策略下的状态值函数的无偏估计(由于 G_t 是真实获得的，所以属于无偏估计，但是存在高方差)。

在 MC 中，每个回报都是对 $v_\pi(s)$独立同分布的估计，通过对这些折扣回报求期望(均值)来评估策略 π 为

$$v_\pi(s)=\mathbb{E}_\pi(G_t|s\in\mathcal{S})=\text{average}(G_t^{(1)}+G_t^{(2)}+\cdots+G_t^{(i)}+\cdots+G_t^{(n)}|s\in\mathcal{S})$$

在一组采样(一个情节)中状态 s 可能多次出现，以更新图的方式表示，如图 5.2 所示。对同一情节中重复出现的状态 s，有如下两种处理方法。

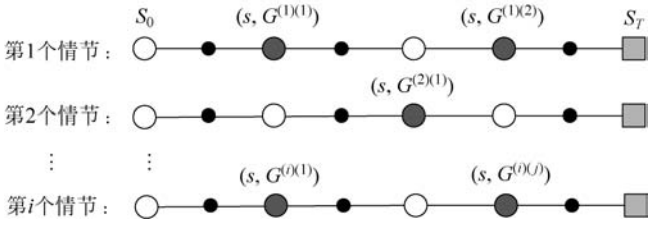

图 5.2 在情节中状态 s 的出现情况(MC 更新图)(见彩插)

(1) 首次访问(first-visit)：在对状态 s 的回报 $v_\pi(s)$进行估计时，只对每个情节中第 1 次访问到状态 s 的回报值作以统计：

$$V(s)=\frac{G_t^{(1)(1)}(s)+G_t^{(2)(1)}(s)+\cdots+G_t^{(i)(1)}(s)}{i} \tag{5.2}$$

(2) 每次访问(every-visit)：在对状态 s 的回报 $v_\pi(s)$进行估计时，对所有访问到状态 s 的回报值都作以统计：

$$V(s)=\frac{G_t^{(1)(1)}(s)+G_t^{(1)(2)}(s)+\cdots+G_t^{(2)(1)}(s)+\cdots+G_t^{(i)(1)}(s)+\cdots+G_t^{(i)(j)}(s)}{N(s)} \tag{5.3}$$

其中，i 表示第 i 个情节；j 表示第 j 次访问到状态 s；$N(s)$表示状态 s 被访问过的总次数。根据大数定理，当 MC 采集的样本足够多时，计算出来的状态值函数估计值 $V_\pi(s)$就会逼近真实状态值函数 $v_\pi(s)$。

从图 5.2 中可以看出，MC 更新图只显示在当前情节中，采样得到的状态转移，且包含了到该情节结束为止的所有状态转移；而 DP 更新图显示在当前状态下所有可能的状态转移，且仅包含单步转移。

基于首次访问的 MC 预测算法，如算法 5.1 所示。

算法 5.1　基于首次访问的 MC 预测算法

输　入：待评估的随机策略 $\pi(a|s)$

初始化：

1. 对所有 $s\in\mathcal{S}$，初始化 $V(s)\in\mathbb{R}$，$V(s^T)=0$；状态 s 被统计的次数 $\text{count}(s)=0$

2. **repeat** 对每一个情节 $k=0,1,2,\cdots$
3. 根据策略 $\pi(a|s)$，生成一个情节序列 $S_0,A_0,R_1,\cdots,S_{T-1},A_{T-1},R_T,S_T$
4. $G\leftarrow 0$
5. **for** 本情节中的每一步 $t=T-1$　**downto**　0 **do**
6. $G\leftarrow\gamma G+R_{t+1}$
7. **if** $S_t\notin\{S_0,S_1,\cdots,S_{t-1}\}$ **then**
8. $\text{count}(S_t)\leftarrow\text{count}(S_t)+1$
9. $V(S_t)\leftarrow V(S_t)+\frac{1}{\text{count}(S_t)}(G-V(S_t))$
10. **end if**
11. **end for**

输　出：$v_\pi=V$

例 5.2　以例 3.1 扫地机器人为例。给出机器人经过图 5.3 的每条轨迹后，相对应的状态值。

如图 5.3 所示，选取了 5 个经过状态 16 的情节，5 个情节依次设置为情节 1、情节 2、情节 3、情节 4 和情节 5。

情节 1：16→15→10→5→0

$$G_{16}^{(1)(1)}=0+0.8\times(0+0.8\times(0+0.8\times(1+0.8\times 0)))=0.8^3\times 1=0.512$$

情节 2：16→17→17→18→19

$$\begin{aligned}G_{16}^{(2)(1)}&=0+0.8\times(-10+0.8\times(0+0.8\times(3+0.8\times 0)))\\&=0.8\times(-10)+0.8^3\times 3=-6.464\end{aligned}$$

情节 3：16→11→6→7→8→9→14→19

$$\begin{aligned}G_{16}^{(3)(1)}=0+0.8\times(0+0.8\times(0+0.8\times(0+0.8\times(0+0.8\times\\(0+0.8\times(3+0.8\times 0))))))=0.8^6\times 3\approx 0.786\end{aligned}$$

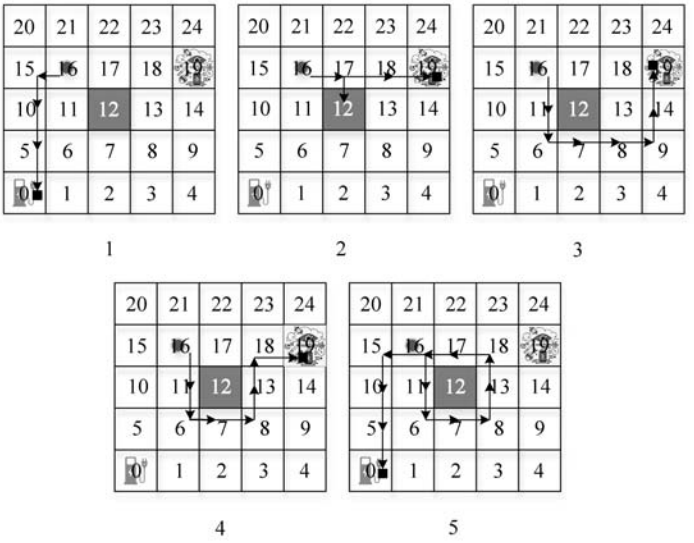

图 5.3 经过状态 16 的其中 5 个情节(见彩插)

也可以直接利用关于回报的定义计算：

$$G_{16}^{(3)(1)} = 0 + 0.8 \times 0 + 0.8^2 \times 0 + 0.8^3 \times 0 + 0.8^4 \times 0 + 0.8^5 \times 0 + 0.8^6 \times 3$$
$$= 0.8^6 \times 3 \approx 0.786$$

情节 4：16→11→6→7→8→13→18→19

$$G_{16}^{(4)(1)} = 0 + 0.8 \times (0 + 0.8 \times (0 + 0.8 \times (0 + 0.8 \times (0 + 0.8 \times$$
$$(0 + 0.8 \times (3 + 0.8 \times 0)))))) = 0.8^6 \times 3 \approx 0.786$$

情节 5：首次访问状态 16：16→11→6→7→8→13→18→17→16→15→10→5→0

$$G_{16}^{(5)(1)} = 0 + 0.8 \times (0 + 0.8 \times (0 + 0.8 \times (0 + 0.8 \times (0 + 0.8 \times (0 + 0.8 \times (0 + 0.8 \times$$
$$(0 + 0.8 \times (0 + 0.8 \times (0 + 0.8 \times (0 + 0.8 \times (1 + 0.8 \times 0)))))))))))$$
$$= 0.8^{11} \times 1 \approx 0.086$$

第 2 次访问状态 16：16→15→10→5→0

$$G_{16}^{(5)(2)} = 0 + 0.8 \times (0 + 0.8 \times (0 + 0.8 \times (1 + 0.8 \times 0))) = 0.8^3 \times 1 = 0.512$$

在情节 5 中状态 16 被访问了两次。利用首次访问的 MC 预测方法时，计算累计回报值只使用该情节中第一次访问状态 16 的回报，即 $G_{16}^{(5)(1)}$。所以使用这 5 条情节，利用首次访问方式计算状态 16 的估计值为 $V(S_{16}) = (G_{16}^{(1)(1)} + G_{16}^{(2)(1)} + G_{16}^{(3)(1)} + G_{16}^{(4)(1)} + G_{16}^{(5)(1)})/5 = -0.859$；而利用每次访问方式计算状态 16 的估计值为 $V(S_{16}) = (G_{16}^{(1)(1)} + G_{16}^{(2)(1)} + G_{16}^{(3)(1)} + G_{16}^{(4)(1)} + G_{16}^{(5)(1)} + G_{16}^{(5)(2)})/6 = -0.630$。

在线视频 21

5.3 蒙特卡洛评估

由最优策略的两种求解方式可知，利用动作值函数更适合于无模型求解最优策略。于是将估计状态值函数的 MC 预测问题转化为估计动作值函数的 MC 评估问题，也就是

说，对状态-动作对(s,a)进行访问而不是对状态s进行访问。根据策略π进行采样，记录情节中(s,a)的回报G_t，并对(s,a)的回报求期望，得到策略π下的动作值函数$q_\pi(s,a)$的估计值。同样地，MC评估方法对每一组状态-动作对(s,a)的评估方法也分为首次访问和每次访问两种。

为了保证算法中值函数和策略的收敛性，DP算法会对所有状态进行逐个扫描。在MC评估方法中，根据动作值函数计算的性质，必须保证每组状态-动作对(s,a)都能被访问到，即得到所有(s,a)的回报值，才能保证样本的完备性。针对该问题，我们设定**探索始点**(exploring starts)：每一组(s,a)都有非0的概率作为情节的起始点(s_0,a_0)。

实际上，探索始点在实际应用中是难以达成的，需要配合无限采样才能保证样本的完整性。通常的做法是采用那些在每个状态下所有动作都有非0概率被选中的随机策略。这里我们先从简单的满足探索始点的MC控制算法开始讨论，然后引出基于同策略和异策略方法的MC控制算法。

5.4　蒙特卡洛控制

预测和控制的思想是相同的，它们都是基于带奖赏过程的马尔可夫链来对目标进行更新，其区别在于以下两点。

(1) MC预测：求解在给定策略π下，状态s的状态值函数$v_\pi(s)$。

(2) MC控制：基于GPI，包含策略评估和策略改进两部分。这里的策略评估是求解在给定策略π下，状态-动作对(s,a)的动作值函数$q_\pi(s,a)$。其策略迭代过程为

$$\pi_0 \xrightarrow{\text{E}} q_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} q_{\pi_1} \xrightarrow{\text{I}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} q_*$$

5.4.1　基于探索始点的蒙特卡洛控制

当采样过程满足探索始点时，对任意策略π_k，MC算法都可以准确地计算出指定状态-动作对的值函数$q_{\pi_k}(s,a)$。一旦得到了动作值函数，可以直接利用基于动作值函数的贪心策略来对策略进行更新。此时对于所有$s\in\mathcal{S}$，任意策略π_k以及更新后的π_{k+1}都将满足策略改进原理：

$$\begin{aligned}
q_{\pi_k}[s,\pi_{k+1}(s)] &= q_{\pi_k}[s,\arg\max_a q_{\pi_k}(s,a)] && \text{—— 根据 } \pi_{k+1}(s)=\arg\max_a q_{\pi_k}(s,a)\\
&= \max_a q_{\pi_k}(s,a) && \text{—— 将最优策略提到公式外面}\\
&\geqslant q_{\pi_k}[s,\pi_k(s)] && \text{—— 使用上一个策略的动作值函数}\\
&\geqslant v_{\pi_k}(s)
\end{aligned} \tag{5.4}$$

式(5.4)表明，采用基于动作值函数的贪心策略，改进后的策略π_{k+1}一定优于或等价于π_k。这一过程保证了MC控制算法能够收敛到最优动作值函数和最优策略。

对于5.3节介绍的无穷采样假设，由于在实际环境中无法实现这一条件，我们使用**基于探索始点的蒙特卡洛**(Monte Carlo based on Exploring Starts，MCES)控制算法来进行规避。

MCES控制算法通过情节到情节的方式对策略进行评估和更新，每一情节结束后，使用观测到的回报进行策略评估，然后在该情节访问到的每一个状态上进行策略改进。

MC控制算法主要分为以下两个阶段。

(1) 策略评估：根据当前策略 π 进行采样，得到一条完整情节，估计每一组(s,a)的动作值函数。

(2) 策略改进：基于动作值函数 $q_\pi(s,a)$，直接利用贪心策略对策略进行改进。

算法5.2给出了MCES控制算法。

算法 5.2 MCES控制算法

输　入：待评估的确定策略 $\pi(s)$

初始化：

对所有 $s\in\mathcal{S}^+$，$a\in\mathcal{A}(s)$，初始化 $Q(s,a)\in\mathbb{R}$，$Q(s^T,a)=0$

对所有 $s\in\mathcal{S}^+$，$a\in\mathcal{A}(s)$，状态-动作对(s,a)被统计的次数 $\text{count}(s,a)=0$

repeat 对每一个情节 $k=0,1,2,\cdots$

　　以非0概率随机选取初始状态-动作对(S_0,A_0)

　　根据策略 $\pi(s)$，从初始状态-动作对(S_0,A_0)开始，生成一个情节序列：

$$S_0,A_0,R_1,\cdots,S_{T-1},A_{T-1},R_T,S_T$$

　　$G\leftarrow 0$

　　for 本情节中的每一步 $t=T-1$　**downto**　0 **do**

　　$G\leftarrow\gamma G+R_{t+1}$

　　if $S_t,A_t\notin\{S_0,A_0,S_1,A_1,\cdots,S_{t-1},A_{t-1}\}$ **then**

　　$\text{count}(S_t,A_t)\leftarrow\text{count}(S_t,A_t)+1$

　　$Q(S_t,A_t)\leftarrow Q(S_t,A_t)+\dfrac{1}{\text{count}(S_t,A_t)}[G-Q(S_t,A_t)]$

　　end if

　　$\pi(S_t)\leftarrow\arg\max_a Q(S_t,a)$

　end for

输　出：$\pi_*=\pi$

在算法5.2中，$Q(s,a)$的初始值一般设置为0。若使用非0初始化，则可以通过$Q[s][a]=\text{random}()/10$来控制。$Q(s,a)$表示动作值函数估计值，所以使用大写$Q$表示。在程序中，常以defalutdict()声明未定型字典$Q(s,a)$，其参数可表示为{s:(action1_value,action2_value),…}形式，并通过$Q[s][a]$来调用(s,a)的动作值函数。

虽然可以通过探索始点来弥补无法访问到所有状态-动作对的缺陷，但这一方法并不合理，普遍的解决方法就是保证Agent能够持续不断地选择所有可能的动作，这也称为无探索始点方法，该方法分为同策略方法与异策略方法两种。

在线视频 22

5.4.2 同策略蒙特卡洛控制

在同策略MC控制算法中，策略通常是**软性**的(soft)，即对于所有的 $s\in\mathcal{S}$、$a\in\mathcal{A}(s)$，均有 $\pi(a|s)>0$。但策略都必须逐渐变得贪心，以逐渐逼近一个确定性策略。同策略方

法使用ε-**贪心策略**(ε-greedy policy),其公式为

$$\pi(a|s)\leftarrow\begin{cases}1-\varepsilon+\dfrac{\varepsilon}{|\mathcal{A}(s)|} & \text{当 } a=A^* \text{ 时,以概率 } 1-\varepsilon+\dfrac{\varepsilon}{|\mathcal{A}(s)|} \text{ 选择具有最大价值的动作}\\ \dfrac{\varepsilon}{|\mathcal{A}(s)|} & \text{当 } a\neq A^* \text{ 时,以概率} \dfrac{\varepsilon}{|\mathcal{A}(s)|} \text{ 随机选择动作}\end{cases} \tag{5.5}$$

其中,$\mathcal{A}(s)$表示状态 s 的动作空间;$|\mathcal{A}(s)|$表示状态 s 可采取的动作总数;A^* 表示最优动作。

GPI 并没有要求必须使用贪心策略,只要求采用的优化方法逐渐逼近贪心策略即可。根据策略改进定理,对于一个 ε-软性策略 π,任何根据 q_π 生成的ε-贪心策略都是对其所做的改进。下面对 ε-软性策略改进定理进行证明。

假设 π'为ε-greedy 策略,对任意状态 $s\in\mathcal{S}$有

$$\begin{aligned}q_\pi[s,\pi'(s)]&=\sum_a\pi'(a|s)q_\pi(s,a)\\&=\frac{\varepsilon}{|\mathcal{A}(s)|}\sum_a q_\pi(s,a)+(1-\varepsilon)\max_a q_\pi(s,a)\\&\geqslant\frac{\varepsilon}{|\mathcal{A}(s)|}\sum_a q_\pi(s,a)+(1-\varepsilon)\sum_a\frac{\pi(a|s)-\frac{\varepsilon}{|\mathcal{A}(s)|}}{1-\varepsilon}q_\pi(s,a)\\&=\frac{\varepsilon}{|\mathcal{A}(s)|}\sum_a q_\pi(s,a)-\frac{\varepsilon}{|\mathcal{A}(s)|}\sum_a q_\pi(s,a)+\sum_a\pi(a|s)q_\pi(s,a)\\&=v_\pi(s)\end{aligned} \tag{5.6}$$

所以,根据策略改进定理,$\pi'\geqslant\pi$。

基于同策略首次访问 ε-greedy 策略的 MC 算法,如算法 5.3 所示。

算法 5.3　基于同策略首次访问 ε-greedy 策略的 MC 算法

输　入:待评估的 ε-greedy 策略 $\pi(a|s)$

初始化:

对所有 $s\in\mathcal{S}^+$,$a\in\mathcal{A}(s)$,初始化 $Q(s,a)\in\mathbb{R}$,$Q(s^T,a)=0$

对所有 $s\in\mathcal{S}^+$,$a\in\mathcal{A}(s)$,状态-动作对(s,a)被统计的次数 $\text{count}(s,a)=0$

ε←(0,1)为一个逐步递减的较小的实数

repeat 对每一个情节 $k=0,1,2,\cdots$

　　根据策略 $\pi(a|s)$,生成一个情节序列 $S_0,A_0,R_1,\cdots,S_{T-1},A_{T-1},R_T,S_T$

　　$G\leftarrow 0$

　　for 本情节中的每一步 $t=T-1$　**downto**　0 **do**

　　$G\leftarrow\gamma G+R_{t+1}$

　　if $S_t,A_t\notin\{S_0,A_0,S_1,A_1,\cdots,S_{t-1},A_{t-1}\}$ **then**

　　$\text{count}(S_t,A_t)\leftarrow\text{count}(S_t,A_t)+1$

　　$Q(S_t,A_t)\leftarrow Q(S_t,A_t)+\dfrac{1}{\text{count}(S_t,A_t)}[G-Q(S_t,A_t)]$

　　$A^*\leftarrow\arg\max_a Q(S_t,a)$

13. **for** $a \in \mathcal{A}(S_t)$ **do**
14. **if** $a == A^*$ **then**
15. $\pi(a|S_t) \leftarrow 1-\varepsilon+\frac{\varepsilon}{|A(S_t)|}$
16. **else**
17. $\pi(a|S_t) \leftarrow \frac{\varepsilon}{|A(S_t)|}$
18. **end if**
19. **end for**
20. **end if**
21. **end for**

输　出：$q_* = Q$；$\pi_* = \pi$

在算法 5.3 中，基于动作值函数 Q 的 ε-贪心策略，对于贪心动作给予 $\pi(a|S_t) \leftarrow 1-\varepsilon+\frac{\varepsilon}{|A(S_t)|}$ 的概率，其他动作给予 $\pi(a|S_t) \leftarrow \frac{\varepsilon}{|A(S_t)|}$ 概率。


阅读代码 14

例 5.3　利用同策略首次访问 ε-贪心策略 MC 算法，给出扫地机器人的最优策略。扫地机器人通过多次实验，不断地更新 Q 值，最终收敛到最优策略，并得到一条回报最大的路径。同策略蒙特卡洛首次访问控制算法中，对动作值函数 Q 的计算，也是通过对每一情节中第一次访问到的该状态-动作对的回报进行平均，然后选择使该动作值函数 Q 最大的动作，作为该状态下应该采取的动作。表 5.2 给出 5 个代表性状态，基于同策略首次访问 MC 算法的扫地机器人最优策略的求解过程。


在线表格 07

表 5.2　基于同策略首次访问 MC 算法的扫地机器人最优策略计算过程($\varepsilon_0 = 0.1$)

	S_5	…	S_{10}	…	S_{18}	…	S_{20}	…	S_{24}
Q_0	0.000;0.000; *.***;0.000	…	0.000;0.000; *.***;0.000	…	0.000;0.000; 0.000;0.000	…	*.***;0.000; *.***;0.000	…	*.***;0.000; 0.000;*.***
π_0	0.667;0.167; 0.000;0.167	…	0.667;0.167; 0.000;0.167	…	0.625;0.125; 0.125;0.125	…	0.000;0.750; 0.000;0.250	…	0.000;0.750; 0.250;0.000
A_0	1;0;0;0	…	0;1;0;0	…	1;0;0;0	…	0;1;0;0	…	0;1;0;0
Q_1	0.000;0.000; *.***;0.000	…	0.000;0.000; *.***;0.000	…	−5.120;0.000; 0.000;3.000	…	*.***;−0.269; *.***;−1.605	…	*.***;0.000; 0.000;*.***
π_1	0.667;0.167; 0.000;0.167	…	0.667;0.167; 0.000;0.167	…	0.125;0.125; 0.125;0.625	…	0.000;0.750; 0.000;0.250	…	0.000;0.750; 0.250;0.000
A_1	1;0;0;0	…	1;0;0;0	…	0;0;0;1	…	0;1;0;0	…	0;1;0;0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Q_{7500}	0.391;1.000; *.***;0.368	…	0.326;0.641; *.***;−0.546	…	1.530;0.667; 0.265;3.000	…	*.***;0.396; *.***;0.710	…	*.***;3.000; 1.605;*.***
π_{7500}	0.104;0.792; 0.000;0.104	…	0.104;0.792; 0.000;0.104	…	0.078;0.078; 0.078;0.766	…	0.000;0.156; 0.000;0.843	…	0.000;0.843; 0.156;0.000
A_{7500}	0;1;0;0	…	0;0;0;1	…	0;0;0;1	…	0;0;0;1	…	0;1;0;0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

续表

	S_5	…	S_{10}	…	S_{18}	…	S_{20}	…	S_{24}
Q_{12500}	0.402;1.000;*.***;0.379	…	0.342;0.661;*.***;−0.575	…	1.580;0.756;0.486;3.000	…	*.***;0.436;*.***;0.808	…	*.***;3.000;1.646;*.***
π_{12500}	0.062;0.875;0.000;0.062	…	0.062;0.875;0.000;0.062	…	0.047;0.047;0.047;0.859	…	0.000;0.094;0.000;0.906	…	0.000;0.906;0.094;0.000
A_{12500}	0;1;0;0	…	0;0;0;1	…	0;0;0;1	…	0;1;0;0	…	0;1;0;0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Q_{19999}	0.405;1.000;*.***;0.382	…	0.342;0.668;*.***;−0.565	…	1.590;0.752;0.553;3.000	…	*.***;0.469;*.***;0.935	…	*.***;3.000;1.680;*.***
π_{19999}	0.000;1.000;0.000;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;1.000;0.000;0.000
A_{19999}	0;1;0;0	…	0;1;0;0	…	0;0;0;1	…	0;0;0;1	…	0;1;0;0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Q_{20000}	0.405;1.000;*.***;0.382	…	0.342;0.668;*.***;−0.565	…	1.590;0.752;0.553;3.000	…	*.***;0.469;*.***;0.935	…	*.***;3.000;1.680;*.***
π_{20000}	0.000;1.000;0.000;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;1.000;0.000;0.000
A_{20000}	0;1;0;0	…	0;1;0;0	…	0;0;0;1	…	0;0;0;1	…	0;1;0;0
π_*	0;1;0;0	…	0;1;0;0	…	0;0;0;1	…	0;0;0;1	…	0;1;0;0

另外，常用 ε-柔性策略公式还有以下 4 种。

(1) 随机贪心策略。基于随机数，用一个较小的阈值 ε 来控制策略的探索性：

$$\pi(a\mid S_t)\leftarrow\begin{cases}A^*, & \text{rand}()>\varepsilon\\ \text{rand}(A), & \text{rand}()\leqslant\varepsilon\end{cases}\tag{5.7}$$

当随机数大于 ε 时，选择最大动作值函数对应的动作；当随机数小于或等于 ε 时，随机地选择动作 rand(A)。

(2) 玻耳兹曼探索。定义 t 时刻选择动作 A_t 的概率，其公式为

$$\pi(A_t\mid S_t)=\frac{\mathrm{e}^{Q_t(s_t,A_t)/\tau_t}}{\sum\limits_a \mathrm{e}^{Q_t(s_t,a)/\tau_t}}\tag{5.8}$$

其中，$\tau_t\geqslant 0$ 表示温度参数，控制探索的随机性。当 $\tau_t\to 0$ 时，选择贪心动作；当 $\tau_t\to\infty$ 时，随机选择动作。

(3) 最大置信上界法。虽然贪心策略选取的动作在当前时刻看起来是最好的，但实际上其他一些没有选取到的动作可能从长远看更好。在这些动作中，通常是根据它们的潜力来选择可能最优的动作，因此在选择动作时，一方面要考虑其估计值最大，另一方面也要考虑探索长时间没有访问到的动作，以免错过更好的动作。一个有效的方法是按照以下公式来选择动作：

$$A_t=\arg\max_a\left[Q_t(s,a)+c\sqrt{\frac{\ln t}{N_t(s,a)}}\right]\tag{5.9}$$

其中，$\ln t$ 表示 t 的自然对数；$N_t(s,a)$表示当前状态 s；在时刻 t 之前动作 a 被选择的次数；c 是一个大于 0 的数，用来控制探索的程度。如果 $N_t(s,a)=0$，则动作 a 就被认为是当前状态 s 下满足最大化条件的动作。

(4) 乐观初始值方法。给值函数赋予一个比实际价值大得多些的乐观初始值。这种乐观估计会鼓励不断地选取收益接近估计值的动作。但无论选取哪一种动作，收益都比最初始的估计值小，因此在估计值收敛之前，所有动作都会被多次尝试。即使每一次都按照贪心法选择动作，系统也会进行大量的探索。

在线视频 23

5.4.3 异策略与重要性采样

另一种常用的无探索始点蒙特卡洛方法为异策略 MC 方法。在介绍异策略方法之前，我们先来了解重要性采样，因为几乎所有的异策略方法都使用到重要性采样。所谓的重要性采样是利用来自其他分布的样本，估计当前某种分布期望值的通用方法。在引出异策略 MC 控制之前，首先阐述预测问题的**重要性采样**(importance sampling)。

1. 重要性采样

以离散型数据为例，假设 $f(x)$是一个服从 $p(x)$分布的函数，其期望公式为

$$\mathbb{E}_{x\sim p}[f(x)]=\sum_x p(x)f(x) \tag{5.10}$$

其中，$x\sim p$ 表示 x 服从 $p(x)$分布，也可记为 $f(x)\sim p$。

通常情况下，可以在服从 $p(x)$分布的离散型数据中进行采样，得到样本集 $\{x_1,x_2,\cdots,x_N\}$，则 $f(x)$在 $p(x)$分布下的期望为

$$\mathbb{E}_{x\sim p}[f(x)]\approx\frac{1}{N}\sum_{x_i\sim p,i=1}^{N}f(x_i) \tag{5.11}$$

在有些任务中，为了得到分布函数 $p(x)$，需要采集大量的样本才能拟合原期望，或存在部分极端、无法代表分布的样本。针对这些任务，在服从 $p(x)$分布的数据中采样存在困难的问题，根据重要性采样原则，可以将该任务转化为从服从简单分布 $q(x)$的数据中进行采样，得到的样本集为$\{x_1,x_2,\cdots,x_N\}$。此时 $f(x)$在 $p(x)$分布下的期望为

$$\mathbb{E}_{x\sim p}[f(x)]=\sum_x p(x)f(x)=\sum_x q(x)\left[\frac{p(x)}{q(x)}f(x)\right]=\mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)}f(x)\right] \tag{5.12}$$

其中，$\mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)}f(x)\right]$表示函数$\frac{p(x)}{q(x)}f(x)$在 $q(x)$分布下的期望。

根据 MC 采样思想，在采样数据足够多时，$f(x)$在 $p(x)$分布下的期望近似为

$$\mathbb{E}_{x\sim p}[f(x)]=\mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)}f(x)\right]\approx\frac{1}{N}\sum_{x_i\sim q,i=1}^{N}\frac{p(x_i)}{q(x_i)}f(x_i)=\frac{1}{N}\sum_{x_i\sim q,i=1}^{N}\omega(x_i)f(x_i) \tag{5.13}$$

其中，$\omega(x)$为**重要性采样比率**(importance-sampling ratio)，有 $\omega(x)=p(x)/q(x)$。

由此，我们将求解 $f(x)$在 $p(x)$分布下的函数期望问题，转换为求解包含重要性采

样比率 ω 的 $f(x)$在 $q(x)$分布下的函数期望。

重要性采样的特点：

(1) $q(x)$与 $p(x)$具有相同的定义域。

(2) 采样概率分布 $q(x)$与原概率分布 $p(x)$越接近，方差越小；反之，方差越大。

通常采用加权重要性采样来减小方差，即用 $\sum_{j=1}^{N}\omega(x_j)$ 替换 N。

① **普通重要性采样**(ordinary importance sampling)的函数估计为

$$\mathbb{E}_{x\sim p}[f(x)]\approx\sum_{x_i\sim q,i=1}^{N}\omega(x_i)f(x_i)/N \tag{5.14}$$

② **加权重要性采样**(weighted importance sampling)的函数估计为

$$\mathbb{E}_{x\sim p}[f(x)]\approx\sum_{x_i\sim q,i=1}^{N}\omega(x_i)f(x_i)/\sum_{j=1}^{N}\omega(x_j) \tag{5.15}$$

2. 基于重要性采样的异策略方法

异策略方法目标策略和行为策略是不同的。这里假设目标策略为 π，行为策略为 b，所有情节都遵循行为策略 b，并利用行为策略 b 产生的情节来评估目标策略。这样需要满足覆盖条件，即目标策略 π 中的所有动作都会在行为策略 b 中被执行。也就是说，所有满足 $\pi(a|s)>0$ 的(s,a)均有 $b(a|s)>0$。根据轨迹在两种策略下产生的相对概率来计算目标策略 π 的回报值，该相对概率称为重要性采样比率，记为 ρ。

以 S_t 作为初始状态，其采样得到的后续状态-动作对序列为：$A_t,S_{t+1},\cdots,S_{T-1},A_{T-1},S_T$。在任意目标策略 π 下发生的概率如式(5.16)所示，在任意行为策略 b 下发生后的概率如式(5.17)所示。

$$\begin{aligned}&P(A_t,S_{t+1},A_{t+1},\cdots,S_T|S_t,A_{t:T-1}\sim\pi)\\&=\pi(A_t|S_t)p(S_{t+1}|S_t,A_t)\pi(A_{t+1}|S_{t+1})\cdots p(S_T|S_{T-1},A_{T-1})\\&=\prod_{k=t}^{T-1}\pi(A_k|S_k)p(S_{k+1}|S_k,A_k)\end{aligned} \tag{5.16}$$

$$\begin{aligned}&P(A_t,S_{t+1},A_{t+1},\cdots,S_T|S_t,A_{t:T-1}\sim b)\\&=b(A_t|S_t)p(S_{t+1}|S_t,A_t)\pi(A_{t+1}|S_{t+1})\cdots p(S_T|S_{T-1},A_{T-1})\\&=\prod_{k=t}^{T-1}b(A_k|S_k)p(S_{k+1}|S_k,A_k)\end{aligned} \tag{5.17}$$

其中，p 表示状态转移概率；T 是该情节的终止时刻。注意：公式累乘符号的上标为 $T-1$，因为最后一个动作发生在 $T-1$ 时刻。$S_t,A_{t:T-1}\sim\pi$ 表示该情节服从目标策略 π。

这样某一情节在目标策略和行为策略下发生的相对概率为

$$\rho_{t:T-1}=\frac{\prod_{k=t}^{T-1}\pi(A_k|S_k)p(S_{k+1}|S_k,A_k)}{\prod_{k=t}^{T-1}b(A_k|S_k)p(S_{k+1}|S_k,A_k)}=\prod_{k=t}^{T-1}\frac{\pi(A_k|S_k)}{b(A_k|S_k)} \tag{5.18}$$

其中，$\rho_{t:T-1}$ 表示某一情节从 t 到 T 时刻的重要性采样比率，也就是基于两种策略采取动

作序列 $A_t, A_{t+1}, \cdots, A_{T-1}$ 的相对概率，与重要性采样比率 $\omega(x)$ 相对应。从式(5.12)可以看出，尽管情节的形成依赖于状态转移概率 p，但由于分子分母中同时存在 p，可以约去，所以重要性采样比率仅仅依赖于两个策略，而与状态转移概率无关。

行为策略中的回报期望是不能直接用于评估目标策略的。根据重要性采样原则，需要使用比例系数 $\rho_{t:T-1}$ 对回报进行调整，使其矫正为正确的期望值：

$$\mathbb{E}[G_t \mid S_t = s] = v_b(s) \quad \Rightarrow \quad v_\pi(s) = \mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] \tag{5.19}$$

假设遵循行为策略 b 采样得到了一系列情节。为方便计算，将这些情节首尾相连，并按时刻状态出现的顺序进行编号。例如第 1 个情节在时刻 100 状态结束，则第 2 个情节的编号就在时刻 101 状态开始，以此类推。在每次访问方法中，存储所有访问过状态 s 的时间步，记为 $\mathcal{T}(s)$，并以 $|\mathcal{T}(s)|$ 表示状态 s 被访问过的总次数。在首次访问方法中，$\mathcal{T}(s)$ 只包括这些情节中第一次访问到 s 的时间步。此外，以 $T(t)$ 表示在 t 时刻后的第一个终止时刻，以 G_t 表示从 t 到 $T(t)$ 时刻的回报，以 $\rho_{t:T(t)-1}$ 表示回报 G_t 的重要性采样比率(在增量式计算中常简写为 W_i)。

根据重要性采样思想，状态值函数的计算方法分为两种。

1) 普通重要性采样

将回报按照权重缩放后进行平均。属于无偏估计，具有方差无界的特点。其计算如式(5.20)所示：

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|} \tag{5.20}$$

2) 加权重要性采样

将回报进行加权平均。属于有偏估计，具有方差较小的特点。其计算如式(5.21)所示：

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}} \tag{5.21}$$

分母为 0 时，$V(s)=0$。

两种方法的主要差异在于偏差和方差的不同。后面将讨论它们在首次访问方法下，获得单一情节回报后的值函数估计值。

1) 普通重要性采样的偏差与方差

采用某种方法估计值函数，当估计结果的期望恒为 $v_\pi(s)$ 时，该方法是无偏估计，但其方差可能是无界的。当 $\rho=10$ 时，表明该轨迹在目标策略下发生的可能性是行为策略下的 10 倍，$V(s)=10G_t$，根据普通重要性采样得到的估计值 $V_\pi(x)$ 是回报值的 10 倍，这就存在高方差。

2) 加权重要性采样的偏差与方差

由于比例系数 ρ 被消去，所以加权重要性采样的估计值就等于回报值，与重要性采样比例无关。因为该回报值是仅有的观测结果，所以是一个合理的估计，但它的期望是 $v_b(s)$ 而非 $v_\pi(s)$，所以该方法属于有偏估计。此外，由于加权估计中回报的最大权重是 1，所以其方差会明显低于普通估计。

而实际上，由于重要性采样比率涉及所有状态的转移概率，因此有很高的方差，从这一点来说，MC 算法不太适合于处理异策略问题。异策略 MC 算法只有理论研究价值，实际应用的效果并不明显，难以获得最优动作值函数。

5.4.4　蒙特卡洛中的增量式计算

增量式计算在强化学习中具有很重要的应用价值。本节在说明 MC 增量式计算之前，先来了解经典的增量式计算。

1. 经典的增量式计算

计算一组数据的均值，最简单的方法是使用采样平均法，即将所有的历史信息记录下来，然后求平均。假设有一组实数数据，其形式为 $x_1, x_2, \cdots, x_k, x_{k+1}, \cdots$。令 x_k 为第 k 个数据的数值，u_k 为前 k 个数据的平均值，有：

$$u_k = \frac{1}{k}\sum_{i=1}^{k} x_i \tag{5.22}$$

使用式(5.22)计算前 k 个数据的平均值时，需要对这 k 个数据进行存储，然后再求平均值。这种方法容易产生计算量大和存储消耗量大的问题。根据数学中的迭代思想，引入增量式计算方法，以简化求解过程。增量式推导如下所示：

$$\begin{aligned} u_k &= \frac{1}{k}\sum_{i=1}^{k} x_i \\ &= \frac{1}{k}\left(x_k + \sum_{i=1}^{k-1} x_i\right) \\ &= \frac{1}{k}\left(x_k + (k-1)u_{k-1}\right) \\ &= u_{k-1} + \frac{1}{k}\left(x_k - u_{k-1}\right) \end{aligned}$$

根据上式的规律，构建经典增量式公式，如式(5.23)所示：

$$\text{NewEstimate} \leftarrow \text{OldEstimate} + \text{StepSize}(\text{Target} - \text{OldEstimate}) \tag{5.23}$$

其中对式(5.23)中变量说明如下：

- Target：表示当前采样值，称其为目标值，通常伴随着噪声，即存在偏差。
- OldEstimate，NewEstimate：表示真实值的估计值，可理解为 V 或 Q，其目的是逼近真实目标值 v 或 q。
- Target－OldEstimate：表示评估误差，通常记为 δ。在递增公式中，通过向 Target 趋近而使得 δ 逐步减少。
- StepSize：表示步长(学习率)，通常记为 α。在这里用一个较小值 α 来替代 $1/k$，使公式更具一般性。

需要说明的是，之所以将 Target 称为目标值，是因为从单次更新过程来看，OldEstimate 是朝着 Target 移动的。而从全部更新结果来看，OldEstimate 是朝着真实目标值移动的。在自举方法中，Target 也常被称为**自举估计值**(bootstrapping estimate)。

由此可见，增量式计算方法是一种基于样本 Target 的随机近似过程，拆分了均值求解过程，减少了存储消耗，简化了计算过程。

2. MC 的增量式

对于 MC 预测问题，在采集情节的基础上也可以使用增量式的实现方法。其增量式的实现可以分为同策略和异策略两种模式。

同策略 MC：使用传统增量式计算公式，不涉及重要性采样，t 时刻状态 S_t 的状态值函数更新递归式如下所示：

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \tag{5.24}$$

- 公式右侧的 $V(S_t)$ 为历史状态值函数的均值，表示估计值，即 OldEstimate。
- G_t 为 t 时刻的回报，表示目标值，即 Target。
- α 为步长，当 α 是固定步长时，该式称为恒定-α MC。

异策略 MC：假设已经获得了状态 s 的回报序列 $G_1, G_2, \cdots, G_{n-1}$，每个回报都对应一个随机重要性权重 W_i（$W_i = \rho_{t:T(t)-1}$）。当获得新的回报值 G_n 时，我们希望以增量式的方式，在状态值函数估计值 V_n 的基础上估计 V_{n+1}。

在普通重要性采样中，仅仅需要对回报赋予权重 W_i，其增量式与经典增量式方程基本一致：

$$V(s) \leftarrow V(s) + \alpha[WG - V(s)] \tag{5.25}$$

在加权重要性采样中，需要为每一个状态计算前 n 个回报的累积权重 C_n：

$$C_n = \sum_{k=1}^{n} W_k \quad \Rightarrow \quad C_n = C_{n-1} + W_n \quad (C_0 = 0) \tag{5.26}$$

其中，C_n 表示前 n 个回报的重要性权重之和，右式是其增量形式。由此得到 V_n 的更新递归式为

$$V_{n+1} = V_n + \frac{W_n}{C_n}(G_n - V_n) \quad (n \geqslant 1) \tag{5.27}$$

推导过程为

$$\begin{aligned} V_{n+1} &= \sum_{k=1}^{n} \frac{W_k G_k}{C_n} = \frac{\sum_{k=1}^{n-1} W_k G_k + W_n G_n}{C_n} \\ &= \frac{\sum_{k=1}^{n-1} W_k G_k}{C_n} + \frac{W_n G_n}{C_n} = \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k} \cdot \frac{\sum_{k=1}^{n-1} W_k}{C_n} + \frac{W_n G_n}{C_n} \\ &= V_n \cdot \frac{C_n - W_n}{C_n} + \frac{W_n G_n}{C_n} \\ &= V_n + \frac{W_n}{C_n}(G_n - V_n) \quad (n \geqslant 1) \end{aligned}$$

3. 随机近似条件

由于 MC 是基于采样方法进行更新的，当状态和动作空间是离散且有穷时，只要满

足如下两个条件，动作值函数估计值 $Q_\pi(s,a)$ 就能收敛于真实动作值函数 $q_\pi(s,a)$。

(1) 满足 $\sum_{t=1}^{\infty}\alpha_t=\infty$，　$\sum_{t=1}^{\infty}\alpha_t^2<\infty$　　(5.28)

(2) 所有的状态-动作对都能够渐近地被无限次访问到。

若 α 根据随机近似条件逐渐变小，则它能以概率 1 收敛。以上收敛条件均适用于其他采样模型。所以只要所有 (s,a) 都能被无限多次地访问到，且贪心策略在极限情况下能够收敛（ε 衰减为 0），那么采样模型就能以 1 的概率收敛到最优动作值函数和最优策略。

5.4.5　异策略蒙特卡洛控制

异策略 MC 控制算法与异策略 MC 预测算法的原理一致，动作值函数更新递归式为

$$Q(S_t,A_t)\leftarrow Q(S_t,A_t)+\frac{W}{C(S_t,A_t)}[G-Q(S_t,A_t)] \tag{5.29}$$

算法 5.4 给出了用于估算最优策略的异策略 MC 控制算法。

算法 5.4　异策略 MC 控制算法

初始化：

1. 对所有 $s\in\mathcal{S}^+$，$a\in\mathcal{A}(s)$，初始化 $Q(s,a)\in\mathbb{R}$，$Q(s^T,a)=0$，$C(s,a)=0$
2. $\varepsilon\leftarrow(0,1)$为一个逐步递减的较小的实数
3. $\pi(s)\leftarrow\arg\max_a Q(s,a)$

4. **repeat** 对每一个情节 $k=0,1,2,\cdots$
5. 　$b\leftarrow$任意软性策略
6. 　根据策略 $b(s)$，从初始状态-动作对 (S_0,A_0) 开始，生成一个情节序列 $S_0,A_0,R_1,\cdots,S_{T-1},A_{T-1},R_T,S_T$
7. 　$G\leftarrow 0$
8. 　$W\leftarrow 1$
9. 　**for** 本情节中的每一步 $t=T-1$　**downto**　0 **do**
10. 　　$G\leftarrow\gamma G+R_{t+1}$
11. 　　$C(S_t,A_t)\leftarrow C(S_t,A_t)+W$
12. 　　$Q(S_t,A_t)\leftarrow Q(S_t,A_t)+\frac{W}{C(S_t,A_t)}[G-Q(S_t,A_t)]$
13. 　　$\pi(S_t)\leftarrow\arg\max_a Q(S_t,a)$
14. 　　**if** $A_t=\pi(S_t)$ **then** $W\leftarrow W\frac{1}{b(A_t|S_t)}$ **else** break
15. 　**end for**

输　出：$\pi_*=\pi$

在算法 5.4 第 14 行，如果行为策略 b 采样的动作与目标策略 π 采取的动作不同，则 $\pi(A_t|S_t)=0$，重要性权重为 0，无法更新值函数，退出本次情节的迭代。从这一步可以看出，每一次迭代都会很快停止，效率不会很高，这一点也是异策略 MC 应用性不强的原因

之一；第 14 行：当 $A_t=\pi(S_t)$ 时，$\pi(A_t|S_t)=1$，重要性权重更新递归式就简写为 $W\leftarrow W\frac{1}{b(A_t|S_t)}$。

阅读代码 15

例 5.4 对于扫地机器人问题，通过行为策略来生成情节，然后利用每次访问和重要性采样比率计算动作值函数 $Q(S_t,A_t)$，如果行为策略采样的动作不是目标策略采取的动作，则会结束该循环开始新一轮循环。这样也就产生很多无用的数据，使得学习效率不高。表 5.3 为异策略每次访问 MC 控制算法的 Q 值更新表。

在线表格 08

表 5.3 异策略每次访问 MC 控制算法的 Q 值更新过程

	S_5	…	S_{10}	…	S_{18}	…	S_{20}	…	S_{24}
Q_0	0.000;0.000; *.***;0.000	…	0.000;0.000; *.***;0.000	…	0.000;0.000; 0.000;0.000	…	*.***;0.000; *.***;0.000	…	*.***;0.000; 0.000;*.***
b_1	0.333; 0.333; 0.000; 0.333	…	0.333; 0.333; 0.000; 0.333	…	0.250; 0.250; 0.250; 0.250	…	0.000; 0.500; 0.000; 0.500	…	0.000; 0.500; 0.500; 0.000
π_0	1.000; 0.000; 0.000; 0.000	…	1.000; 0.000; 0.000; 0.000	…	1.000; 0.000; 0.000; 0.000	…	0.000; 1.000; 0.000; 0.000	…	0.000; 1.000; 0.000; 0.000
Q_1	0.000;0.000; *.***;0.000	…	0.000;0.000; *.***;0.000	…	1.920;0.000; 0.000;0.000	…	*.***;0.000; *.*** ;0.000	…	*.***;3.000; 0.000;*.***
b_1	0.333; 0.333; 0.000; 0.333	…	0.333; 0.333; 0.000; 0.333	…	0.250; 0.250; 0.250; 0.250	…	0.000; 0.500; 0.000; 0.500	…	0.000; 0.500; 0.500; 0.000
π_1	1.000; 0.000; 0.000; 0.000	…	1.000; 0.000; 0.000; 0.000	…	1.000; 0.000; 0.000; 0.000	…	0.000; 1.000; 0.000; 0.000	…	0.000; 1.000; 0.000; 0.000
⋮	⋮	…	⋮	…	⋮	…	⋮	…	⋮
Q_{7500}	0.960;1.000; *.***;0.977	…	1.215;0.799; *.***;1.208	…	1.917;1.889; 1.894;3.000	…	*.***;1.181; *.***;1.229	…	*.***;3.000; 1.916;*.***
b_{7500}	0.333; 0.333; 0.000; 0.333	…	0.333; 0.333; 0.000; 0.333	…	0.250;0.250; 0.250; 0.250	…	0.000; 0.500; 0.000; 0.500	…	0.000; 0.500; 0.500; 0.000
π_{7500}	0.000; 1.000; 0.000; 0.000	…	1.000; 0.000; 0.000; 0.000	…	0.000; 0.000; 0.000; 1.000	…	0.000; 0.000; 0.000; 1.000	…	0.000; 1.000; 0.000; 0.000
⋮	⋮	…	⋮	…	⋮	…	⋮	…	⋮
Q_{12500}	0.975;1.000; *.***;0.979	…	1.220;0.800; *.***;1.215	…	1.918;1.899; 1.904;3.000	…	*.***;1.200; *.***;1.229	…	*.***;3.000; 1.914;*.***
b_{12500}	0.333; 0.333; 0.000; 0.333	…	0.333; 0.333; 0.000; 0.333	…	0.250; 0.250; 0.250; 0.250	…	0.000; 0.500; 0.000; 0.500	…	0.000; 0.500; 0.500; 0.000
π_{12500}	0.000; 1.000; 0.000; 0.000	…	1.000; 0.000; 0.000; 0.000	…	0.000; 0.000; 0.000; 1.000	…	0.000; 0.000; 0.000; 1.000	…	0.000; 1.000; 0.000; 0.000
⋮	⋮	…	⋮	…	⋮	…	⋮	…	⋮
Q_{19999}	0.978;1.000; *.***;0.977	…	1.223;0.800; *.***;1.221	…	1.917;1.906; 1.906;3.000	…	*.***;1.209; *.***;1.228	…	*.***;3.000; 1.916;*.***
b_{19999}	0.333; 0.333; 0.000; 0.333	…	0.333; 0.333; 0.000; 0.333	…	0.250; 0.250; 0.250; 0.250	…	0.000; 0.500; 0.000; 0.500	…	0.000; 0.500; 0.500; 0.000

续表

	S_5	…	S_{10}	…	S_{18}	…	S_{20}	…	S_{24}
π_{19999}	0.000; 1.000; 0.000; 0.000	…	1.000; 0.000; 0.000; 0.000	…	0.000; 0.000; 0.000; 1.000	…	0.000; 0.000; 0.000; 1.000	…	0.000; 1.000; 0.000; 0.000
Q_{20000}	0.978;1.000; *.*** ;0.977	…	1.223;0.800; *.***;1.221	…	1.917;1.906; 1.906;3.000	…	*.***;1.209; *.***;1.228	…	*.***;3.000; 1.916; *.***
b_{20000}	0.333; 0.333; 0.000; 0.333	…	0.333; 0.333; 0.000; 0.333	…	0.250; 0.250; 0.250; 0.250	…	0.000; 0.500; 0.000; 0.500	…	0.000; 0.500; 0.500; 0.000
π_{20000}	0.000; 1.000; 0.000; 0.000	…	1.000; 0.000; 0.000; 0.000	…	0.000; 0.000; 0.000; 1.000	…	0.000; 0.000; 0.000; 1.000	…	0.000; 1.000; 0.000; 0.000
π_*	0.000; 1.000; 0.000; 0.000	…	1.000; 0.000; 0.000; 0.000	…	0.000;0.000; 0.000; 1.000	…	0.000; 0.000; 0.000; 1.000	…	0.000; 1.000; 0.000; 0.000

5.5　小结

本章介绍了从经验中学习价值函数和最优策略的蒙特卡洛方法，这些“经验”主要体现在从多个情节采样数据。与 DP 方法相比，其优势主要在以下 3 个方面：①MC 方法不需要完整的环境动态模型，而可以直接通过与环境的交互来学习最优的决策行为。②MC 方法可以使用数据仿真或采样模型。在很多应用中，构建 DP 方法所需要的显式状态概率转移模型通常很困难，但是通过仿真采样得到多情节序列数据却很简单。③MC 方法可以简单、高效地聚焦于状态的一个小的子集，它可以只评估关注的区域而不评估其他的状态。

5.6　习题

1. 举例说明蒙特卡洛首次访问和每次访问的异同点。

2. 蒙特卡洛方法可以解决哪些强化学习问题？

3. 给出蒙特卡洛估计 $q_\pi(s,a)$ 值的更新图。

4. 修改异策略蒙特卡洛控制算法，使之可以递增计算加权的平均值，并给出伪代码。

5.（编程）通过蒙特卡洛法计算：第 3 章习题 2（图 3.12）扫地机器人在等概率策略的情况下，分别给出实验次数为 5000 和 50000 时，每个状态的价值 $v_\pi(s)$。