4 Model-Free Policy Evaluation

在上一节中，我们首先讨论了三个日益复杂的问题公式，我们将在下面重述：

1.马尔可夫过程是一个具有马尔可夫性质的随机过程。

2.马尔可夫奖励过程是一个马尔可夫过程，在每个时间步都有奖励以及折扣奖励的累积，称为价值(values)。

3.马尔可夫决策过程(MDP)是在每一个状态下都增加了选择或行动的马尔可夫奖励过程。

在上节课的后半部分，我们讨论了 MDP policy evaluation 的两种方法和确定 MDP optimal policy 的三种方法。policy evaluation 的两种方法是通过线性方程组和动态规划直接求解。三种控制方法是 brute force policy search、policy iteration 和 value iteration。

所有这些方法都隐含着这样一个假设，即我们知道每次状态转换的 reward 和 probabilities。然而，在许多情况下，我们不容易获得这样的信息，这就需要 model-free 算法。在章中我们将讨论 model-free policy evaluation。也就是说，我们将给定一个不知道 reward 和 transition probabilities 的 policy，并且将尝试学习该 policy 的价值。注意，在下一章节之前，我们不会讨论如何在 model-free 的情况下改进我们的 policy。

4.1 Notation Recap

在深入探讨 model-free policy evaluation 的一些方法之前，我们首先回顾一下 MDP 在上节课中的一些注释，这是我们在这节课中需要的。

回想一下，我们把 MRP 的回报定义为从时间步长 t 开始到时间域 H(H 可以是有限的)结束的折扣奖励总和。从数学上来说，这表现为：

$$G_t = \sum_{i=t}^{H-1} \gamma^{i-t} r_i \qquad (1)$$

对于 0≤t≤H-1，其中 $0 < \gamma \leq 1$ 是折扣系数，r_i 是时间步 i 的奖励。对于 MDP，回报 G_t 的定义相同，奖励 r_i 通过遵循策略π(a|s)生成。

状态值函数 $V^{\pi}(s)$ 是遵循固定策略π下从状态 s 开始的预期回报。数学上，我们可以把它表达为：

$$V^{\pi}(s) = E_{\pi}\left[G_t | s_t = s\right] \quad (2)$$

$$= E_{\pi}\left[\sum_{i=t}^{H-1} \gamma^{i-t} r_i \mid s_t = s\right] \quad (3)$$

state-action 值函数$Q^{\pi}(s,a)$是从状态 s 开始，采取动作 a，然后对此后的所有转换都遵循固定策略π的预期回报。数学上，我们可以把它表达为：

$$Q^{\pi}(s,a) = E_{\pi}\left[G_t | s_t = s, a_t = a\right] \quad (4)$$

$$= E_{\pi}\left[\sum_{i=t}^{H-1} \gamma^{i-t} r_i \mid s_t = s, a_t = a\right] \quad (5)$$

在这个章节中，我们将假设一个无限时间域以及固定的 reward、transition probabilities 和 policy。这允许我们有独立于时间的状态和 state-action 值函数，正如上节课推导出来的那样。

在这个章节，我们将使用一个新的定义，那就是历史(history)的定义。

Definition 4.1.**history** 是 agent 经历的 state、action 和 rewards 的有序元组。在 episodic 领域，我们将用 history 置换 episode，当考虑已经进行过很多次交互时，我们将以下列方式索引历史:其中第 j 个 history 是:

$$h_j = (s_{j,1}, a_{j,1}, r_{j,1}, s_{j,2}, a_{j,2}, r_{j,2}, ..., s_{j,L_j}),$$

其中L_j是互动的长度(episode 的长度,history 的长度,也就是运行了多少次),$s_{j,t}, a_{j,t}, r_{j,t}$,分别是 history j 在时间步 t 的 state,action,reward。

定义:估计器(估计变量,估计函数) $\hat{\theta}$的偏置:

$$Bias_{\theta}(\hat{\theta}) = E_{x|\theta}[\hat{\theta}] - \theta$$

定义:估计器(估计变量,估计函数) $\hat{\theta}$的方差:

$$Var_{\theta}(\hat{\theta}) = E_{x|\theta}[(\hat{\theta} - E[\hat{\theta}])^2]$$

定义:估计器$\hat{\theta}$的均方误差(MSE):

$$MSE(\hat{\theta}) = Var(\hat{\theta}) + Bias_{\theta}(\hat{\theta})^2$$

4.2 Dynamic Programming

Dynamic Programming for policy Evaluation:

- Initialize $V_0^\pi(s) = 0$ for all s
- For $k = 1$ until convergence $\quad |V_k^\pi - V_{k-1}^\pi| < \varepsilon$
  - For all s in S

$$V_k^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V_{k-1}^\pi(s')$$

Dynamic Programming for Policy π,Value Evaluation:

- Initialize $V_0^\pi(s) = 0$ for all s
- For $k = 1$ until convergence
  - For all s in S

$$V_k^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V_{k-1}^\pi(s')$$

- $V_k^\pi(s)$ is exact value of k-horizon value of state s under policy π
- $V_k^\pi(s)$ is an estimate of infinite horizon value of state s under policy π

$$V^\pi(s) = \mathbb{E}_\pi[G_t | s_t = s] \approx \mathbb{E}_\pi[r_t + \gamma V_{k-1} | s_t = s]$$

上个章节中的动态规划算法是在策略π下计算γ< 1 的 infinite horizon MDP 的值，为了方便起见，我们在此将其改写为算法 1。

Algorithm 1 Iterative algorithm to calculate MDP value function for a stationary policy π

procedure POLICY EVALUATION(M, π, ϵ)
  For all states $s \in S$, define $R^\pi(s) = \sum_{a \in A} \pi(a|s) R(s, a)$
  For all states $s, s' \in S$, define $P^\pi(s'|s) = \sum_{a \in A} \pi(a|s) P(s'|s, a)$
  For all states $s \in S$, $V_0(s) \leftarrow 0$
  $k \leftarrow 0$
  while $k = 0$ or $||V_k - V_{k-1}||_\infty > \epsilon$ **do**
    $k \leftarrow k + 1$
    For all states $s \in S$, $V_k(s) = R^\pi(s) + \gamma \sum_{s' \in S} P^\pi(s'|s) V_{k-1}(s')$
  return V_k

用这种形式写成，我们可以从几个方面来思考V_k。首先，$V_k(s)$是从状态 s 开始遵循策略π并经过 k 次转换的精确值。其次，对于很大的 k 值，或者当算法 1 终止时，$V_k(s)$的值逼近无限时间域状态 s 的值$V^\pi(s)$。

我们还可以通过备份图来表达该算法的行为，如图 1 所示。该备份图是自上而下读取的，白色圆圈代表状态，黑色圆圈代表行动和弧线代表期望。该图显示了从顶部的状态 s 开始，向下移动两个时间步长的分支过程。它还显示了从状态 s 开始并根据策略 π 采取行动后，我们如何获取下一个状态的期望值。在动态规划中，我们使用当前估计值$V_{k-1}(s')$并采用 bootstrap 或者估计求解出下一个状态的值(In dynamic programming,we bootstrap, or estimate, the value of the next state using our current estimate $V_{k-1}(s')$)。

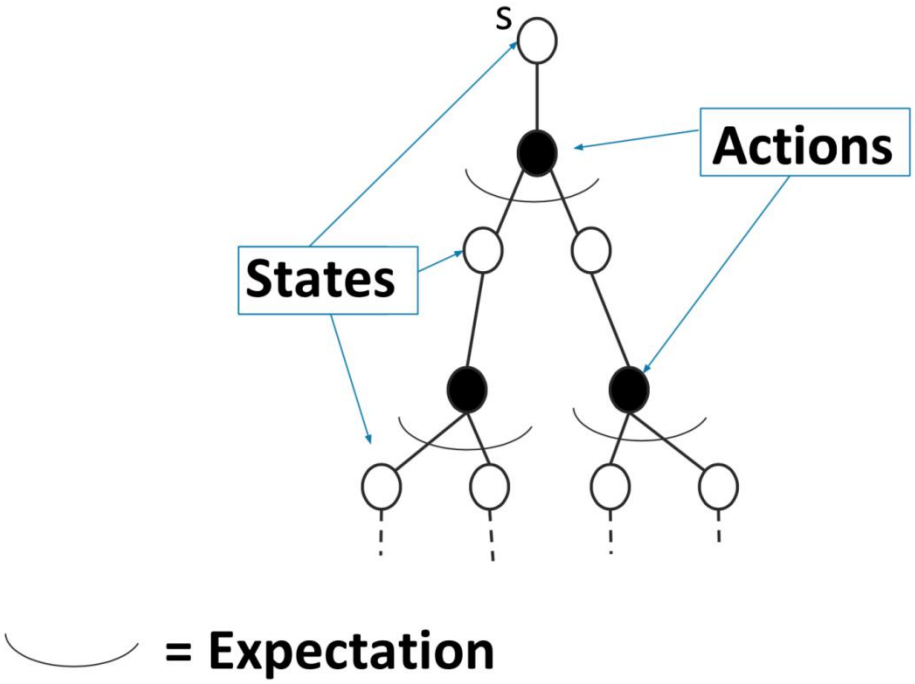


Figure 1:Backup diagram for the dynamic programming policy evaluation algorithm.

4.3 Monte Carlo On Policy Evaluation

Monte Carlo policy Evaluation 仅应用于回合制。

我们现在描述我们的第一个 model-free policy evaluation 算法，它使用一种被称为蒙特卡罗方法的流行计算方法。我们首先在强化学习的背景之外浏览蒙特卡罗方法的一个例子，然后更一般地讨论该方法，最后将蒙特卡罗应用于强化学习。我们在这里强调，这种方法只在 episode 环境中工作，在本节中我们将更仔细地检查算法时并看到为什么会这样。

假设我们想估计今天从你家到斯坦福大学的通勤时间。假设我们也有一个通勤模拟器，它可以模拟我们对交通状况、天气、施工延误和其他变量的不确定性，以及这些变量之间的相互作用。估计预期通勤时间的一种方法是在模拟器上模拟我们的通勤多次，然后取模拟通勤时间的平均值。这被称为我们通勤时间的蒙特卡罗估计。

一般来说，我们通过观察在现实生活中或通过模拟如何产生某个量的多次迭代，然后对观察到的量进行平均，得到该量的蒙特卡罗估计。根据大数定律，这个平均数收敛于数量的期望值。

在强化学习的背景下，我们想要估计的量是$V^{\pi}(s)$，它是指遵循 policy π，然后从状态 s 开始的平均回报G_t，因此我们可以通过三个步骤得到的蒙特卡罗估计值$V^{\pi}(s)$:

1. 多次从初始状态开始执行策略π，直到终止
2. 记录从状态 s 开始后我们所观察到的回报G_t。
3. 用我们得到的G_t的平均值来估计$V^{\pi}(s)$。

课件算法如下(MC On Policy Evaluation):

- Aim: estimate $V^{\pi}(s)$ given episodes generated under policy π
  - $s_1, a_1, r_1, s_2, a_2, r_2, \ldots$ where the actions are sampled from π
- $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots$ in MDP M under policy π
- $V^{\pi}(s) = \mathbb{E}_{\pi}[G_t | s_t = s]$
- MC computes empirical mean return
- Often do this in an incremental fashion
  - After each episode, update estimate of V^{π}

蒙特卡洛策略估计的备份图如图 2 所示。蓝线指明我们从状态 s 开始到终止采样的完整 episode 过程。

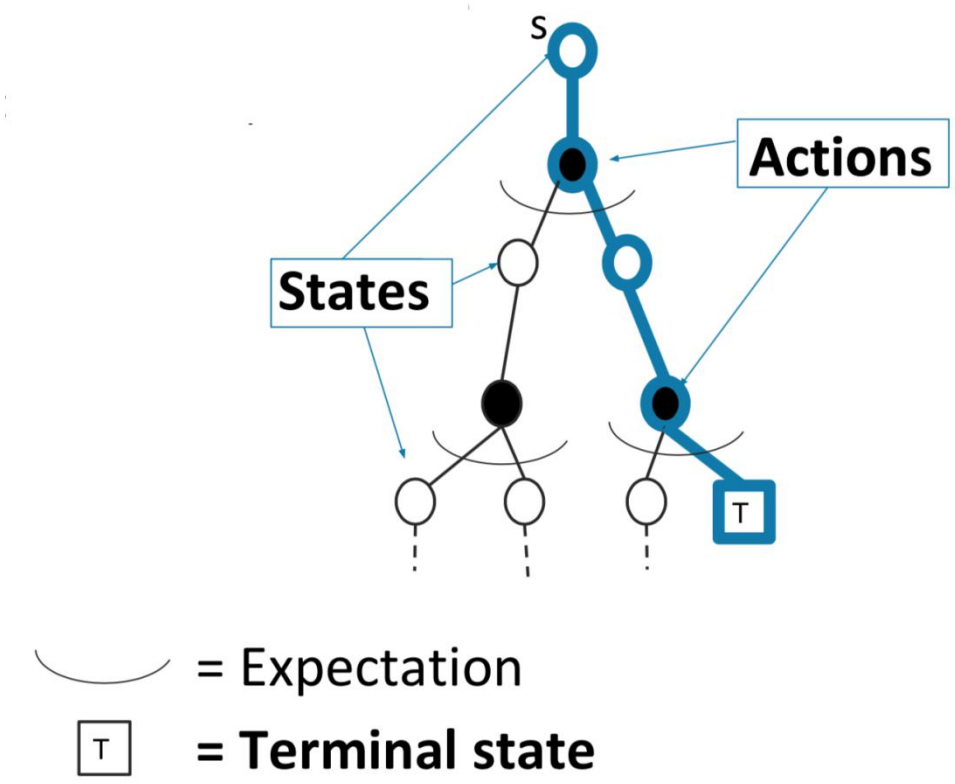


Figure 2:Backup diagram for the Monte Carlo policy evaluation algorithm.

蒙特卡洛策略估计有两种形式，这取决于我们是在每次首次访问某个 state 时取平均值，还是每次访问该 state 时取平均值。他们分别被成为 First-Visit Monte Carlo 和 Every-Visit Monte Carlo On Policy Evaluation。关于两者的算法详见 lnote3 第四页的 algorihm 2,algorithm 3,algorithm 4,algorithm 5。

课件 First-Visit MC On Policy Evaluation(N(s)首次访问该状态的次数,显然在该算法中取值为 0 或 1):

Initialize $N(s) = 0,\ G(s) = 0\ \forall s \in S$

Loop

- Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \ldots, s_{i,T_i}$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \cdots \gamma^{T_i-1} r_{i,T_i}$ as return from time step t onwards in ith episode
- For each state s visited in episode i
  - For **first** time t that state s is visited in episode i
    - Increment counter of total first visits: $N(s) = N(s) + 1$
    - Increment total return $G(s) = G(s) + G_{i,t}$
    - Update estimate $V^\pi(s) = G(s)/N(s)$

Properties:

- V^π estimator is an unbiased estimator of true $\mathbb{E}_\pi[G_t|s_t = s]$
- By law of large numbers, as $N(s) \to \infty$, $V^\pi(s) \to \mathbb{E}_\pi[G_t|s_t = s]$

课件 Every-Visit MC On Policy Evaluation:

Initialize $N(s) = 0,\ G(s) = 0\ \forall s \in S$

Loop

- Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \ldots, s_{i,T_i}$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \cdots \gamma^{T_i-1} r_{i,T_i}$ as return from time step t onwards in ith episode
- For each state s visited in episode i
  - For **every** time t that state s is visited in episode i
    - Increment counter of total first visits: $N(s) = N(s) + 1$
    - Increment total return $G(s) = G(s) + G_{i,t}$
    - Update estimate $V^\pi(s) = G(s)/N(s)$

Properties:

- V^π every-vist MC estimator is an **biased** estimator of V^π
- But consistent estimator and often has better MSE

如果我们在一个真正的马尔可夫域，Every-Visit Monte Carlo 将会更高效，因为我们每次访问一个状态时都会更新该状态的平均回报率。

练习 4.1，以下图为例，初始化每个状态值的为 0，现有 history 如下：

h=(S3,TL,+0，S2,TL,+0，S1,TL,+1，terminal)

如下问题：

1. first-visit monte carlo 在每一个状态估计值 V 是多少?V=[1,1,1,0,0,0,0]
2. every-visit monte carlo 在每个状态估计值 V 是多少 V=[1,1,1,0,0,0,0]
3. incremental first-visit monte carlo 在 $\alpha = \frac{2}{3}$ 的条件下每个状态的估计值是多少?V=[2/3,0,0,0,0,0,0]

4. incremental every-visit monte carlo 在$\alpha=\frac{2}{3}$的条件下每个状态的估计值是多少?V=[2/3,0,0,0,0,0,0]

S1	S2	S3	S4	S5	S6	S7
Okay Field Site R=+1	R=0	R=0	 R=0	R=0	R=0	Fantastic Field Site R=+10

$P(s'|s,TL)=$

1	0	0	0	0	0	0
1	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	1	0

$P(s'|s,TR)=$

0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	1
0	0	0	0	0	0	1

Figure 3:Mars Rover Markov Decision Process with actions Try Left(TL) and Try Right(TR)

4.4 Monte Carlo Off Policy Evaluation

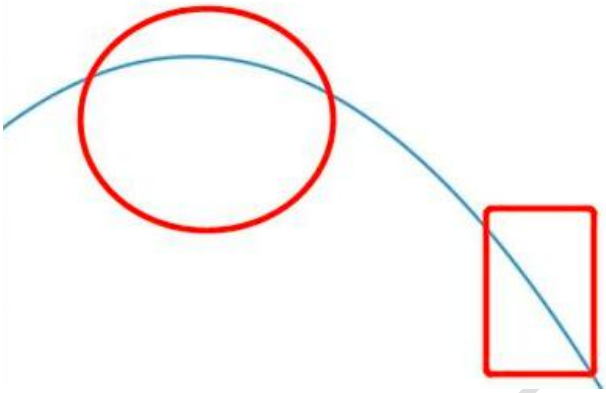
在上一节中，我们讨论了基于 policy π，我们能够获得G_t的许多实现的情况。然而，在许多高成本或高风险的情况下，我们无法根据我们上 4.3 节的策略来获得G_t的值。例如，我们可能有与一个医疗方案相关的数据，但我们想确定这些医疗方案的价值。在本节中，我们将描述 Monte Carlo off policy evaluation,这是一种使用从一个策略中获取的数据来评估不同策略的方法。

4.4.1 Importance Sampling

重要性采样。off policy evaluation 的关键要素被称为 importance sampling。importance sampling 的目标是估计函数 f(x)的期望值，x 服从 q 分布，而数据$f(x_1),...,f(x_n)$中的x_i是采用 p 分布(从 x 服从 p 分布中采样，但是求 x 服从 q 分布的均值，因为 q 求不出来，显然 q 是期望的正式分布，p 是估计。

为什么要使用重要性采样，蒙特卡洛是一种随机采样方法，该方法随着取样的增长而越发的精确，那么有什么方法能够在一定的抽样数量基础上来增加准确度，减少方差呢？这就需要我们人为地对抽样的分布进行干预，如下图所示，显然在红色圆内采样要比方形内采样要来的准确(圆内蓝线面积比率更大)，所以我们在抽样时以更大的概率抽取圆形区域的样本，一来就能够提高估计的准确度。假设我们用分布 p(x)在原函数上采样，如下图三角状的图形，依照这个分布进行采样我们一定程度上可以使得在原函数对积分贡献大的区域获得更多的采样机

会。但我们不能通过对{f(x1),f(x2)...f(xn)}进行简单的求和平均来获取估计值，因为这是不均匀采样生成的数据，必须要进行加权，这个权重就是重要性权重。



那么为什么要引入一个新的分布 p(x)呢?因为原函数 f(x)也许本身就是定义在一个分布之上的，我们定义这个分布为 q(x),我们没有办法直接找到它没所以无法对其进行采样，所以另辟蹊径找到一个更加简单分布 p(x)，并对其采样，希望间接求出 f(x)在分布 q(x)下的期望。

小结一下,给定 $q(x_i), p(x_i), f(x_i)\ for\ 1 \le x_i, \le n$，我们想要估计 $E_{x\sim q}[f(x)]$。我们能够通过以下的公式来近似得到：

$$\mathbb{E}_{x\sim q}[f(x)] = \int_x q(x)f(x)dx \tag{8}$$

$$= \int_x p(x)\left[\frac{q(x)}{p(x)}f(x)\right]dx \tag{9}$$

$$= \mathbb{E}_{x\sim p}\left[\frac{q(x)}{p(x)}f(x)\right] \tag{10}$$

$$\approx \sum_{i=1}^{n}\left[\frac{q(x_i)}{p(x_i)}f(x_i)\right]. \tag{11}$$

公式(11)给出了利用分布 p 对 f 进行采样后在分布 q 下的 importance sampling f 的估计值 。注意第一步仅支持 q(x)f(x)>0 的情况，意味着 p(x)>0 for all x。

4.4.2 Importance Sampling off Policy Evaluation

我们现在将重要性抽样估计的通用结果应用于强化学习。在这种情况下，我们想要使用基于策略π_2生成的 n 个历史$h_1,...,h_n$,并通过公式$V^{\pi_1}(s)=E[G_t \mid s_t=s]$近似求解基于策略$\pi_1$的状态 s 的值。利用重要性抽样估计结果，我们可以得出：

$$V^{\pi_1}(s) \approx \frac{1}{n}\sum_{j=1}^{n}\frac{p(h_j \mid \pi_1, s)}{p(h_j \mid \pi_2, s)}G(h_j) \qquad (12)$$

$$G(h_j)=\sum_{t=1}^{L_j-1}\gamma^{t-1}r_{j,t}$$

G 是历史h_j的累积折扣回报。

现在对于一个 policy π,我们能够得到基于 policy π的历史 h_j 的概率:

$$p(h_j|\pi, s=s_{j,1}) = \prod_{t=1}^{L_j-1} p(a_{j,t}|s_{j,t})p(r_{j,t}|s_{j,t},a_{j,t})p(s_{j,t+1}|s_{j,t},a_{j,t}) \quad (13)$$

$$= \prod_{t=1}^{L_j-1} \pi(a_{j,t}|s_{j,t})p(r_{j,t}|s_{j,t},a_{j,t})p(s_{j,t+1}|s_{j,t},a_{j,t}) \quad (14)$$

其中L_j是第 j 个剧集的长度。第一行是从每次转换的三个组件开始的，三个组件是：

1. $p(a_{j,t} \mid s_{j,t})$：我们在状态$s_{j,t}$采取行动$a_{j,t}$的概率
2. $p(r_{j,t} \mid s_{j,t}, a_{j,t})$：我们在状态$s_{j,t}$采取行动$a_{j,t}$后获得的奖励$r_{j,t}$的概率
3. $p(s_{j,t+1} \mid s_{j,t}, a_{j,t})$：我们在状态$s_{j,t}$采取行动$a_{j,t}$后转换到状态$s_{j,t+1}$的概率

现在，结合我们对V^{π_1}的重要抽样估计和我们对历史概率$p(h_j \mid \pi, s=s_{j,1})$的分解，我们得到

$$V^{\pi_1}(s) \approx \frac{1}{n}\sum_{j=1}^{n}\frac{p(h_j|\pi_1,s)}{p(h_j|\pi_2,s)}G(h_j) \quad (15)$$

$$= \frac{1}{n}\sum_{j=1}^{n}\frac{\prod_{t=1}^{L_j-1}\pi_1(a_{j,t}|s_{j,t})p(r_{j,t}|s_{j,t},a_{j,t})p(s_{j,t+1}|s_{j,t},a_{j,t})}{\prod_{t=1}^{L_j-1}\pi_2(a_{j,t}|s_{j,t})p(r_{j,t}|s_{j,t},a_{j,t})p(s_{j,t+1}|s_{j,t},a_{j,t})}G(h_j) \quad (16)$$

$$= \frac{1}{n}\sum_{j=1}^{n}G(h_j)\prod_{t=1}^{L_j-1}\frac{\pi_1(a_{j,t}|s_{j,t})}{\pi_2(a_{j,t}|s_{j,t})}. \quad (17)$$

请注意，我们现在可以在没有转移概率或奖励的情况下准确地估计表达式，因为所有涉及 model dynamic 的术语都在等式的第二步中被消元了。特别是，我

们给出了历史h_j，因此我们可以计算出$G(h_j)=\sum_{t=1}^{L_j-1}\gamma^{t-1}r_{j,t}$，并且我们知道两个策略$\pi_1$和$\pi_2$，因此我们也可以解出第二项。

4.5 Temporal Difference(TD) Learning

到目前为止我们有两个方法进行 policy evaluation:动态规划和蒙特卡洛。动态规划仅利用 boostrapping 帮助我们仅使用一次备份去获取估计值。另一方面，蒙特卡洛通过获取大量的历史数据来帮助我们从模型中解放(MC 一路走到尽头并总结奖励)。现在，我们要介绍一个结合 bootstrapping 和 sampling 的新算法，也是我们第二个 model-free 的 policy evaluation 算法(Bootstrapping 算法，指的就是利用有限的样本资料经由多次重复抽样，重新建立起足以代表母体样本分布的新样本。bootstrapping 的运用基于很多统计学假设，因此采样的准确性会影响假设的成立与否。)。

为了看出是如何结合 sampling 和 bootstrapping，让我们回到 incremental Monte Carlo update:

$$V^{\pi} \leftarrow V^{\pi}(s_t)+\alpha(G_t-V^{\pi}(s_t)). \qquad (18)$$

回顾一下G_t是从时间 t,状态 s 开始执行 policy 直到终止的回报。现在就像动态规划一样用 Bellman backup 替换G_t。也就是用$r_t+\gamma V^{\pi}(s_{t+1})$代替$G_t$，其中$r_t$是在时间 t 采样的奖励，$V^{\pi}(s_{t+1})$是对下一状态的值的估计，代入可以得到我们的 TD-Learning update:

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t)+\alpha(r_t+\gamma V^{\pi}(s_{t+1})-V^{\pi}(s_t)) \qquad (19)$$

差分:

$$\delta = r_t+\gamma V^{\pi}(s_{t+1})-V^{\pi}(s_t) \qquad (20)$$

也通常被称为 **TD error**，与下一状态值的 bootstrap estimate 相结合的采样奖励:

$$r_t+\gamma V^{\pi}(s_{t+1}), \qquad (21)$$

被称为 TD target,完整的 TD learning 算法在 Algorithm6 中给出。我们可以看出使用这个方法，我们能够在发送状态转移(s_t,a_t,r_t,s_{t+1})后立即更新我们的$V^{\pi}(s_t)$值。特别是，我们不需要像蒙特卡洛那样等待一个剧集结束。

Algorithm 6 TD Learning to evaluate policy π

procedure TDLEARNING(step size α, number of trajectories n)
  For all states s, $V^\pi(s) \leftarrow 0$
  while $n > 0$ **do**
    Begin episode E at state s
    while $n > 0$ and episode E has not terminated **do**
      $a \leftarrow$ action at state s under policy π
      Take action a in E and observe reward r, next state s'
      $V^\pi(s) \leftarrow V^\pi(s) + \alpha(R + \gamma V^\pi(s') - V^\pi(s))$
      $s \leftarrow s'$
  return V^π

我们将再次通过 figure 4 中的备份图的方式来检查这个算法。我们通过蓝线看到，我们从 s 开始对一个转换进行采样，然后我们通过当前对下一个状态的估计来估计下一个状态的值，以构建完整的 Bellman backup 估计。

实际上，我们可以用一种叫做 TD(λ)的方法将蒙特卡罗和动态规划结合起来。当λ＝0 时，我们得到上面的 TD-learning 公式，因此给出别名 TD(0)。当λ＝1 时，我们根据所使用的公式恢复 Monte Carlo policy evaluation。当 0＜λ＜1 时，我们得到这两种方法的混合。为了更彻底地处理 TD(λ)，我们请感兴趣的读者自行阅读参考书的的第 7.1 节和第 12.1-12.5 节，其中分别详细介绍了 n 步 TD 学习和 TD(λ)/eligibility traces。

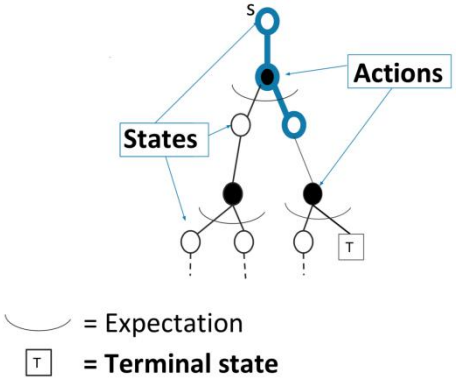


Figure 4: Backup diagram for the TD Learning policy evaluation algorithm

练习 4.2.参考 Figure 3 中的 Mars Rover 的例子。假设每个状态的初始值均为 0，生成的历史数据为：h=(S3,TL,+0, S2,TL,+0, S2,TL,+0, S1,TL,+1, terminal)，问：

1. 当 alpha =1 时,TD(0)的估计值是多少? V=[1,1,0,0,0,0,0]
2. 当 alph2=2/3 时,TD(0)的估计值是多少?V=[1,2/3,0,0,0,0,0]

4.6 Summary of Methods Discussed

在这一章节中，我们使用上一章节的动态规划重新检查了 policy evaluation，并介绍了两种新的 policy evaluation 方法，即蒙特卡罗评估和时序差分学习。

首先，我们注意到动态规划依赖于世界模型(环境模型)，从而推动了蒙特卡

罗和 TD-Learning 的引入。也就是说，我们需要给我们的动态规划策略评估算法提供奖励和转换概率。蒙特卡罗和 TD-Learning 都不受这个约束，使它们成为无模型的方法。

那么如何选择 MC 和 TD-Learning 呢?

从三个方面考虑:

1.Bias/variance characteristics;

2.data efficiency;

3.Computational efficiency

Model-free Policy Evaluation Algorithm 的偏置和方差:

- 回报G_t是$V^{\pi}(s_t)$的无偏估计
- TD target 是$V^{\pi}(s_t)$的有偏估计
- But often much lower variance than a single return G_t
- Return function of multi-step sequence of random action, state&rewards
- TD target 仅需要一个随机动作，奖励和下一状态便可计算
- MC
  - Unbiased
  - High variance
  - Consistent(converges to true) even with function approximation
- TD
  - Some bias
  - Lower variance
  - TD(0) converges to true value with tabular representation
  - TD(0) does not always converges with function approximation

在 Monte Carlo policy evaluation 中，我们生成许多历史记录，然后对我们遇到的各种状态的回报进行平均。为了让我们在有限的时间内生成这些历史，我们要求历史的数据是剧集式(episodic)的——也就是说，我们需要确保我们观察到的每个历史都终止。在动态编程和时序差分学习中，我们只备份一次转换(我们只展望未来的一步)，因此历史是否终止不是一个问题，我们可以将这两个算法应用于非剧集(non-episodic)环境。

另一方面，我们能够在 dynamic programming 和 TD-learning 学习中仅备份一次转换的原因是因为我们利用了环境的马尔可夫假设。此外，算法 4 和 5 中描述的 Incremental Monte Carlo policy evaluation 可用于非马尔可夫环境。

在所有三种方法中，我们收敛到真值函数。上一章节，我们用 Bellman backup 运算符是一个严格收敛的算子的事实证明了动态编程的这个结果。我们在今天的

讲座中看到，由于大数定律，蒙特卡罗策略估计收敛于策略的价值函数。TD(0)也收敛到真实值，我们将在下一节关于批量学习中更详细地研究 TD(0)。

因为我们获取了蒙特卡罗中真实回报分布的平均值，所以我们得到了每个状态下价值的无偏估计。另一方面，在 TD-Learning 中，我们引导下一个状态的值估计以获得当前状态的值估计，因此该估计被下一个状态的估计值所偏置。关于这一点的进一步讨论见 Sutton and Barto·[1]的第 6.2 节。

蒙特卡罗估计的方差相对高于 TD Learning，因为在蒙特卡罗估计中，我们考虑了每个剧集的许多转换，而每词转换对我们的估计贡献方差。另一方面，TD 学习每次更新只考虑一次转换，所以我们不会很快积累方差。

最后，蒙特卡罗通常比 TD(0)更有效。在蒙特卡洛中，我们根据完整剧集的回报来更新一个状态的价值，所以如果在未来的许多轨迹中有高度积极或高度消极的回报，这些回报仍然会立即被纳入我们更新的状态价值中。另一方面，在 TD(0)中，我们仅使用当前步骤中的奖励和下一个状态的值估计值(some previous estimate of the value at the next state)来更新状态的值。这意味着，如果将来出现正数很大或负数很大的奖励并且当奖励已经被用于更新下一状态值的引导估计时我们仅会把它纳入到当前状态的值更新中。这意味着，如果一个高回报的剧集长度为 1，那么我们可能需要经历该剧集 1 次，以获取高回报剧集的信息，从而一路返回到起始状态。

在表 1 中，我们总结了这里讨论的每种方法的优势和局限性。

	Dynamic Programming	Monte Carlo	Temporal Difference
Model Free?	no	yes	yes
non-episodic domains?	yes	no	yes
non-Markovian domains?	no	yes	no
converges to true value	yes	yes	yes
unbiased estimate	n/a	yes	no
variance	n/a	high	low

Table1: summary of methods in this lecture

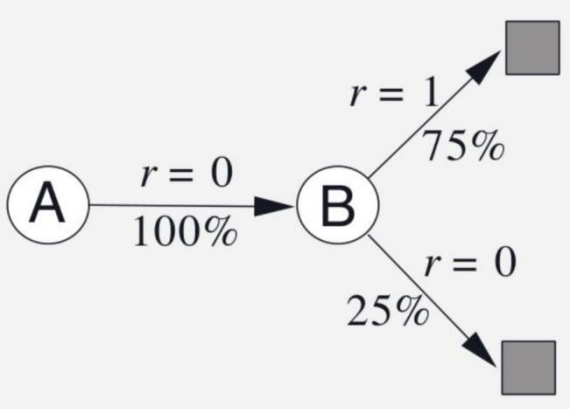


Figure 5:Example 6.4 from Sutton and Barto[1]

4.7 Batch Monte Carlo and Temporal Difference

在今天讲座中我们有一组历史记录，用于多次更，现在我们来看下算法的批量版本。在我们讲解案例之前，让我们先看一下 Sutton 和 Barton[1]的例子 6.4 来更仔细地研究蒙特卡罗和 TD(0)之间的区别。假设γ=1，由 policy π在所有的状态上都执行动作 *act1*，最终生成八个历史:

$$h_1 = (A, act1, +0, B, act1, +0, terminal)$$

$$h_j = (B, act1, +1, terminal)\ for\ j = 2, ..., 7$$

$$h_8 = (B, act1, +0, terminal)$$

然后要么使用 batch Monte Carlo 要么使用 TD(0),并令 alpha=1,我们可以看出 V(B)=0.75。然后使用 Monte Carlo，我们可以得到 V(A)=0，因为仅有第一个剧集访问了状态 A 并且有 0 回报。另一方面，通过 TD(0),我们可以得到 V(A)=0.75，因为我们对 V(A)执行了如下的更新:

$$V(A) \leftarrow r_{1,1} + \gamma V(B)$$

。

在像图 5 所示的马尔可夫域下，由 TD(0)给出的估计更有意义。

在这个部分，我们将讨论批量数据情况下的 Monte Carlo 和 TD(0)。我们将会把一批数据或者一组历史数据$h_1, ..., h_n$多次喂给 Monte Carl 或者 TD(0)，与之前公式的唯一区别是，我们只在每次处理整个批次后更新价值函数。因此对于 TD(0) bootstrap estimate 而言只能在每次喂完一批数据后才能进行。

在蒙特卡罗批处理设置中，每个状态下的值收敛到使观察到的返回值最小化均方误差的值。这直接来自以下事实:在蒙特卡洛中我们在每个状态所得到的平均值超过回报，并且通常，样本的 MSE 最小化器恰好是样本的平均值(This follows directly from the fact that in Monte Carlo, we take an average over returns at each state, and in general, the MSE minimizer of samples is precisely the average of the samples.)。也就是说给定样本:{y1,...,yn},则最小化公式设为 Y 如下式(a):

$$\sum_{i=1}^{n}(y_i-\hat{y})^2 \qquad (a)$$

$$\hat{y}=\sum_{i=1}^{n} y_i \qquad (b)$$

当满足公式(b)时,Y 有最小值。我们在本节开头的例子中可以看到这一点。对于 Monte Carlo 而言 V(A)=0 是因为状态 A 在历史中仅仅被访问过一次。

在 TD(0)的批处理设定中，没有收敛到同蒙特卡洛同样的结果。在 TD(0)中会收敛到 policy π在最大似然估计 MDP 模型的值V^{π}

$$\hat{P}(s'|s,a)=\frac{1}{N(s,a)}\sum_{j=1}^{n}\sum_{t=1}^{L_j-1}\mathbb{1}(s_{j,t}=s,a_{j,t},s_{j,t+1}=s') \qquad (22)$$

$$\hat{r}(s,a)=\frac{1}{N(s,a)}\sum_{j=1}^{n}\sum_{t=1}^{L_j-1}\mathbb{1}(s_{j,t}=s,a_{j,t})r_{j,t}. \qquad (23)$$

换句说话最大似然估计 MDP 模型是我们基于批处理的情况下能创建的最朴素的模型(most naive model)-转换概率$\hat{P}(s'|s,a)$是我们再批处理中基于状态 s 采取行动 a 后看到转变(s,a,s')次数的一部分，以及奖励$\hat{r}(s,a)$是基于状态 s，执行动作 a 并执行完批处理后的平均奖励。

我们还在本节开头的示例中看到了这个结果。 在这种情况下，我们的最大似然模型是:

$$\hat{P}(B|A,act1)=1$$

$$\hat{P}(terminal|B,act1)=1$$

$$\hat{r}(A,act1)=0$$

$$\hat{r}(B,act1)=0.75$$

就像我们知道的一样$V^{\pi}(A)$。

从最大似然 MDP 模型导出的值函数被称为确定性等价估计(certainty equivalence estimate)。利用这种方法我们有了另一种策略估计的方法。我们首先可以利用批处理计算最大似然 MDP 模型。然后我们能够利用这个模型计算V^{π}，并且 model-based policy evaluation 的方法已经在上一节讨论过了。这种方法数据精度高，但计算量大，因为它涉及到通过动态规划求解 MDP 问题，需要时间$O(|S|^3)$和$|S|^2|A|$。

练习 4.3

查看 Figure 3 的火星漫游者例子，假设每个状态的初始值为 0，如果我们批处理两段历史数据:

$$h_1 = (S3, TL, +0, S2, TL, +0, S1, TL, +1, terminal)$$

$$h_1 = (S3, TL, +0, S2, TL, +0, S1, TL, +0, S1, TL, +1, terminal)$$

我们的策略是 TL，那么 certainty equivalence estimate 是什么?