TechX 深度强化学习课程（五）

何舜成

宽带网数字媒体实验室
清华大学自动化系

2018 年 7 月 29 日

目录

蒙特卡洛方法

蒙特卡洛方法计算圆周率

蒙特卡洛是运用统计模拟指导数值计算的一类方法。

图中扇形面积占正方形面积的 $\pi/4$，那么如果随机向正方形中投豆子，那么豆子落在扇形区域的概率为 $\pi/4$。由大数定律可知，随着投掷的豆子数量增加，落在扇形区域的豆子数目占总数的比值将逐渐趋近于 $\pi/4$，因此可以通过模拟的方式求得 π 的一个近似值。

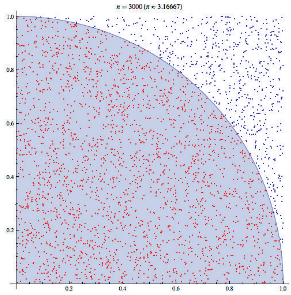


图 1: 数量达到 3000 时，误差不足 1%

蒙特卡洛估值

在 DP 小节里，对某策略 π 求估值函数利用了 Bellman 方程做迭代

$$v_{k+1}(s)=\sum_{a}\pi(a\mid s)\sum_{s',r}p(s',r\mid s,a)[r+\gamma v_k(s')] \qquad (1)$$

但这种方法要求环境的模型（转移概率）$p(s',r\mid s,a)$ 是已知的。在不少问题里，这个概率是未知的。联想到蒙特卡洛方法的统计模拟，我们可以多次模拟，通过统计手段估计转移概率，进而优化策略。

蒙特卡洛估值

由于 $v_\pi(s) = \mathbb{E}[G_t \mid S_t = s]$，那么可以根据策略 π 产生多个 episodes，计算所有 s 之后的累积回报的平均值。有两种略微不同的算法，一种算法只计算 episode 中首次遇到 s 之后的累积回报，一种算法计算所有遇到 s 之后的累积回报。

Initialize:

  $\pi \leftarrow$ policy to be evaluated

  $V \leftarrow$ an arbitrary state-value function

  $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

  Generate an episode using π

  For each state s appearing in the episode:

    $G \leftarrow$ the return that follows the first occurrence of s

    Append G to $Returns(s)$

    $V(s) \leftarrow \text{average}(Returns(s))$

图 2: Alg: First-visit MC Prediction

蒙特卡洛估值

在转移概率未知时，估计状态值函数并没有太大用处，因为在改进策略时，用到公式

$$\pi'(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_\pi(s')] \tag{2}$$

仍然无法求解。

此时更重要的是估计动作值函数 $q_\pi(s, a)$。类似于前页算法，对遇到的 (s, a) 对之后的累积回报做统计平均即可（first-visit 或 every-visit）。

当 π 是确定性策略时，会存在很多 (s, a) 对无法模拟出来，也就无法计算相应的 $q_\pi(s, a)$。若计算不全面，则无法进行策略改进，此时需要加入一些探索机制。

一种解决方法是令起始状态为 s，起始动作为 a，遍历所有这样的 (s, a) 对，这种方法称为**探索起始值**（exploring starts）。

蒙特卡洛控制

类似策略迭代，通过循环估计值函数、改进策略，最终将达到最优策略

$$\pi_0 \rightarrow q_{\pi_0} \rightarrow \pi_1 \rightarrow q_{\pi_1} \rightarrow \pi_2 \rightarrow \cdots \rightarrow \pi_* \rightarrow q_{\pi_*}$$

策略的选择仍然使用贪心法

$$\pi(s) = \arg\max_a q(s, a) \tag{3}$$

策略改进定理仍然生效因为下式成立

$$\begin{aligned} q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \arg\max_a q_{\pi_k}(s, a)) \\ &= \max_a q_{\pi_k}(s, a) \\ &\geq q_{\pi_k}(s, \pi_k(s)) \\ &\geq v_{\pi_k}(s) \end{aligned} \tag{4}$$

蒙特卡洛控制

由于蒙特卡洛估值也需要很多的 episode，做精确估值将耗费大量时间，能否仿照值迭代法只用一个 episode 更新呢?

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s,a) \leftarrow$ arbitrary
  $\pi(s) \leftarrow$ arbitrary
  $Returns(s,a) \leftarrow$ empty list

Repeat forever:
  Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
  Generate an episode starting from S_0, A_0, following π
  For each pair s, a appearing in the episode:
    $G \leftarrow$ the return that follows the first occurrence of s, a
    Append G to $Returns(s,a)$
    $Q(s,a) \leftarrow \text{average}(Returns(s,a))$
  For each s in the episode:
    $\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$

图 3: Alg: Monte Carlo Exploring Starts

放弃 Exploring Starts

控制初始状态 S_0 通常是不可能的，如果仍然要尽可能多地遍历到 (s, a) 对，需要考虑一些随机策略。$\epsilon - soft$ 策略指的是有 *epsilon* 的概率，动作是随机的，而有 $1 - \epsilon$ 的概率动作是确定性的。

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s, a) \leftarrow$ arbitrary
  $Returns(s, a) \leftarrow$ empty list
  $\pi(a|s) \leftarrow$ an arbitrary ε-soft policy

Repeat forever:
  (a) Generate an episode using π
  (b) For each pair s, a appearing in the episode:
    $G \leftarrow$ the return that follows the first occurrence of s, a
    Append G to $Returns(s, a)$
    $Q(s, a) \leftarrow \text{average}(Returns(s, a))$
  (c) For each s in the episode:
    $A^* \leftarrow \arg\max_a Q(s, a)$ (with ties broken arbitrarily)
    For all $a \in \mathcal{A}(s)$:

$$\pi(a|s) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq A^* \end{cases}$$

图 4: Alg: MC Control (For ϵ-soft Policies)

放弃 Exploring Starts

只要能证明 $q_\pi(s, \pi'(s)) \geq v_\pi(s)$，那么策略改进定理将生效。

$$
\begin{aligned}
& q_\pi(s, \pi'(s)) \\
= & \sum_a \pi'(a \mid s) q_\pi(s, a) \\
= & \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) + (1-\epsilon) \max_a q_\pi(s, a) \\
\geq & \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) + (1-\epsilon) \sum_a \frac{\pi(a \mid s) - \frac{\epsilon}{|\mathcal{A}(s)|}}{1-\epsilon} q_\pi(s, a) \\
= & \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) - \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) + \sum_a \pi(a \mid s) q_\pi(s, a) \\
= & v_\pi(s)
\end{aligned}
\tag{5}
$$

On-policy & Off-policy

蒙特卡洛方法始终面临权衡，一是要优化当前策略，二是要探索更好的策略。某种程度上二者是矛盾的，在优化的过程中，往往不得已要去选择非优的动作，去探索更多的可能性。$\epsilon - greedy$ 策略是一种折中，既逼近了最优策略，又一定程度上进行探索。

另一种可能则是在优化时使用一种策略（target policy），在估值的时候使用另一种策略（behavior policy）。

由于优化和估值使用了不同策略，这种学习方式称为 off-policy。

重要性采样

设 target policy 为 π，behavior policy 为 b，若不加改动地利用 b 产生的样本去估计 $V(s)$，将产生偏差，可以对这些样本进行重要性加权，称为**重要性采样**。

假设 π 和 b 都是固定的。在 π 下，从初始值 S_t 开始，产生轨迹 $A_t, S_{t+1}, A_{t+1}, \ldots, S_T$ 的概率为

$$
\begin{aligned}
& P(A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi) \\
=& \pi(A_t \mid S_t) p(S_{t+1} \mid S_t, A_t) \pi(A_{t+1} \mid S_{t+1}) \cdots p(S_T \mid S_{T-1}, A_{T-1}) \\
=& \prod_{k=t}^{T-1} \pi(A_k \mid S_k) p(S_{k+1} \mid S_k, A_k)
\end{aligned}
\tag{6}
$$

重要性采样

据此可以计算两种不同策略下，产生同一种轨迹的概率之比（重要性比例）

$$\rho_{t:T-1} = \frac{\prod_{k=t}^{T-1} \pi(A_k \mid S_k) p(S_{k+1} \mid S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k \mid S_k) p(S_{k+1} \mid S_k, A_k)} = \prod_{k=t}^{T} \frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)} \tag{7}$$

依据该重要性比例，对累积回报进行缩放，即可无偏差地估计 $v_\pi(s)$。

重要性采样

将所有 episodes 连接起来，采用统一且唯一的时间戳 t。令

- $\mathcal{T}(s)$ 为所有 episode 中首次遇到状态 s 时的时间戳集合
- $T(t)$ 为 t 时刻所在 episode 的终结时间
- G_t 为 t 到 $T(t)$ 的累积回报
- $\{G_t\}_{t\in\mathcal{T}(s)}$ 为所有与 s 相关的累积回报的集合
- $\{\rho_{t:T(t)-1}\}_{t\in\mathcal{T}(s)}$ 为对应的重要性比例

那么 $v_\pi(s)$ 的一个估计为

$$V(s) = \frac{\sum_{t\in\mathcal{T}(s)} \rho_{t:T-1} G_t}{|\mathcal{T}(s)|} \tag{8}$$

重要性采样

还有一种加权重要性采样的方法

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T-1}} \tag{9}$$

（以简单例子说明二者的区别）

- 无偏 vs 有偏
- 方差无界 vs 有限方差

示例：无界方差

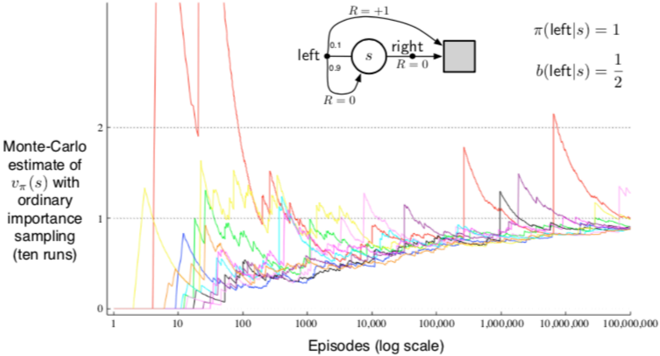


图 5: 不收敛的构造样例

示例：无界方差

计算原始版本的重要性采样的方差，对于任意随机变量

$$Var[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 \tag{10}$$

由于这里 $(\mathbb{E}[X])^2$ 必定是有限值，那么 $V(s)$ 是否无界仅与 $\mathbb{E}[X^2]$ 有关。该例中这一项对应的是

$$\mathbb{E}\left[\left(\prod_{t=0}^{T-1} \frac{\pi(A_t \mid S_t)}{b(A_t \mid S_t)} G_0\right)^2\right] \tag{11}$$

示例：无界方差

分项计算期望得（因权值为 0，一旦选择向右，相应的回报将不计入）

$$
\begin{aligned}
=&\frac{1}{2}\cdot 0.1\cdot 2^2\\
&+\frac{1}{2}\cdot 0.9\frac{1}{2}\cdot 0.1(2\dot{2})^2\\
&+\frac{1}{2}\cdot 0.9\frac{1}{2}\cdot 0.9\frac{1}{2}\cdot 0.1(2\cdot 2\cdot 2)^2\\
&+\cdots\\
=&0.1\sum_{k=0}^{\infty}0.9^k\cdot 2^{k+1}=0.2\sum_{k=0}^{\infty}1.8^k
\end{aligned}
\tag{12}
$$

对于加权重要性采样，一旦第一次选择向左且直接到达终结状态，那么 $V(s)$ 将一直保持为 1。

递推算法

每当有新 episode 出现时，希望我们不需要将之前所有的数据全拿出计算。此时需要一种递推算法。

假设 $G_1, G_2, \ldots, G_{n-1}$ 都是同一个状态之后的累积回报，$W_i = \rho_{t:T(t)-1}$ 是对应的比例，要计算

$$V_n = \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}, n \geq 2 \tag{13}$$

递推式可以写成

$$V_{n+1} = V_n + \frac{W_n}{C_n}(G_n - V_n), n \geq 1 \tag{14}$$

以及

$$C_{n+1} = C_n + W_{n+1} \tag{15}$$

递推算法

Input: an arbitrary target policy π

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s, a) \leftarrow$ arbitrary
    $C(s, a) \leftarrow 0$

Repeat forever:
    $b \leftarrow$ any policy with coverage of π
    Generate an episode using b:
        $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    For $t = T-1, T-2, \ldots$ down to 0:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
        $W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$
        If $W = 0$ then exit For loop

图 6: Alg: Off-policy MC Prediction

Off-policy 蒙特卡洛控制

在递推基础上加上策略改进

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$Q(s, a) \leftarrow$ arbitrary

$C(s, a) \leftarrow 0$

$\pi(s) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)

Repeat forever:

$b \leftarrow$ any soft policy

Generate an episode using b:

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

$G \leftarrow 0$

$W \leftarrow 1$

For $t = T-1, T-2, \ldots$ down to 0:

$G \leftarrow \gamma G + R_{t+1}$

$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$

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

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)

If $A_t \neq \pi(S_t)$ then exit For loop

$W \leftarrow W \frac{1}{b(A_t|S_t)}$

图 7: Alg: Off-policy MC Control

Temporal Difference 算法

TD 估值

蒙特卡洛方法需要每一个 episode 结束之后才能更新策略估值，有些情况下，一个 episode 非常长，或者根本没有终结状态。TD 方法可以破除这个限制，并且做到实时更新。

将 $v_\pi(s)$ 展开一项

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \end{aligned} \tag{16}$$

假设 R_{t+1} 已知，可以认为等式右边的值估计更精确。仿照之前的做法可以得到

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)) \tag{17}$$

TD 估值

这种算法称为 TD(0)，是 TD(λ) 的一个特例（该算法不会在课程中讲授）。

Input: the policy π to be evaluated
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0$, for all $s \in \mathcal{S}^+$)
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\left[R + \gamma V(S') - V(S)\right]$
        $S \leftarrow S'$
    until S is terminal

图 8: Alg: Tabular TD(0)

TD 估值

更新式中的这一项通常被称为 TD error

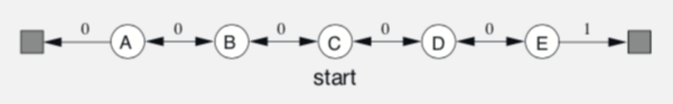
$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{18}$$

事实上，蒙特卡洛方法中的误差就可以写成 TD error 的和

$$\begin{aligned} G_t - V(S_t) &= R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1}) \\ &= \delta_t + \gamma(G_{t+1} - V(S_{t+1})) \\ &= \delta_t + \gamma\delta_{t+1} + \cdots + \gamma^{T-t-t}\delta_{T-1} + \gamma^{T-t}(G_T - V(S_T)) \\ &= \sum_{k=1}^{T-1} \gamma^{k-t}\delta_k \end{aligned} \tag{19}$$

由于估值 V 在 TD 方法中一直在变动，因此这个式子左右并不完全相等，但当 α 足够小时，二者近似相当。

示例：随机行走



在任意非终结状态，向左或向右的概率均等，从 E 向右可以获得 +1 的回报。这是一个**马尔科夫回报过程**（MRP），因为策略无法控制。现在要估计每个状态的 V(S)。

本例中从 A 到 E 状态的真实值函数分别取值 1/6，2/6，3/6，4/6 和 5/6。TD 和 MC 算法初始化对所有状态的估计均为 1/2。

示例：随机行走

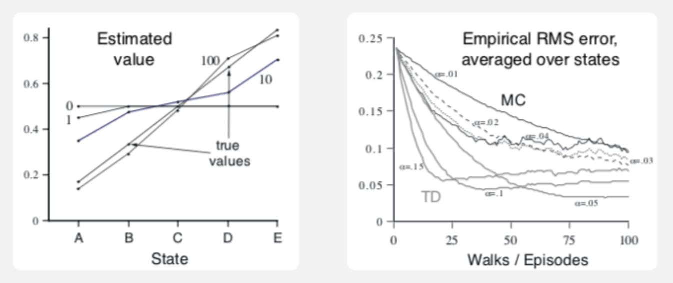


图 9: TD 和 MC 方法误差比较

TD 的预测误差收敛速度明显快于 MC 方法

Sarsa 算法

当环境模型未知时，只对状态值函数进行估计仍然无法计算最优策略。此时应当估值动作值函数，仿写出更新式：

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - A(S_t, A_t)) \tag{20}$$

每次更新依靠的是五元组 $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$，因此得名 **Sarsa** 算法。

可以证明 MC 的误差可以写成 TD error 的和（留作习题）。

Sarsa 算法

Initialize $Q(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Repeat (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

图 10: Alg: Sarsa (On-policy TD Control)

Q-learning

将 TD learning 用 off-policy 实现时，称为 **Q-learning**。Q-learning 是迄今最为广泛运用的强化学习算法，近年来强化学习复苏很大程度归功于将深度学习与 Q-learning 结合。

将 Sarsa 的更新式改动一下

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{21}$$

$\max_a$ 的操作使动作值函数 Q 直接逼近最优值函数 q_*，可以证明，只要所有的 (s, a) 的 Q 值能持续更新，Q 函数能以概率 1 收敛到 q_*。

Q-learning

Initialize $Q(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (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

图 11: Alg: Q-learning

为什么 Q-learning 是 off-policy 的?

Expected Sarsa

若将 Sarsa 更新式中的 $Q(S_{t+1}, A_{t+1})$ 换成所有动作上的均值，则称作 Expected Sarsa。

$$
\begin{aligned}
&Q(S_t, A_t) \\
&\leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma\mathbb{E}[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)) \\
&\leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \sum_a \pi(a \mid S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t))
\end{aligned} \tag{22}
$$

原始 Sarsa 中 A_{t+1} 是随机选的，因此 Expected Sarsa 有减小估值方差的作用。

估计偏差

Q-learning 和 Sarsa 中都有对 Q 取最大值的操作，这将带来正的偏差。假设对于状态 s，以及相应所有的 a，$q(s, a)$ 的真实值都为 0，但由于估计值 $Q(s, a)$ 通常有误差，数值在 0 上下浮动，取最大值之后显然导致了正向偏差。

估计偏差

构造一个简单的例子，下图的 MDP 从 A 点出发，若向右移动，直接结束，回报为 0，若向左移动，到达 B 点，B 点之后有多种选择，每种选择的回报都服从均值为-0.1，方差为 1 的正态分布，之后到达终结状态。

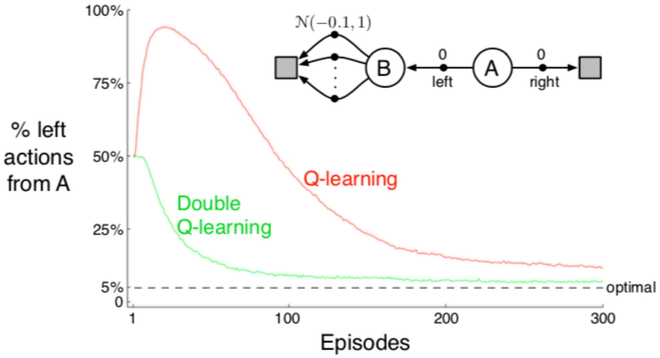


图 12: 构造估计偏差的例子

估计偏差

最大化操作导致的估计偏差来自于值函数的方差，甚至掩盖了 $Q(a)$ 之间本身的差别，估值和动作选择的耦合使得这种偏差一直存在。

设想同时学习两个 Q 函数，学习过程中产生了不同的样本。更新时，采用如下公式

$$\begin{aligned} Q_1(S_t, A_t) \leftarrow & Q_1(S_t, A_t) + \alpha(R_{t+1} \\ & + \gamma Q_2(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a)) - Q_1(S_t, A_t)) \end{aligned} \tag{23}$$

等概率地，将 Q_1 和 Q_2 交换之后进行更新。这种算法称为 **Double Q-learning**。

估计偏差

完整算法如下。由于 Q_1 和 Q_2 学习时采用的是不同的样本，即使二者都存在方差，但几乎不会相同，一定程度上可缓解估计偏差问题。

Initialize $Q_1(s,a)$ and $Q_2(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily
Initialize $Q_1(\textit{terminal-state}, \cdot) = Q_2(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q_1 and Q_2 (e.g., ε-greedy in $Q_1 + Q_2$)
    Take action A, observe R, S'
    With 0.5 probabilility:

$$Q_1(S,A) \leftarrow Q_1(S,A) + \alpha\Big(R + \gamma Q_2\big(S', \operatorname{argmax}_a Q_1(S',a)\big) - Q_1(S,A)\Big)$$

    else:

$$Q_2(S,A) \leftarrow Q_2(S,A) + \alpha\Big(R + \gamma Q_1\big(S', \operatorname{argmax}_a Q_2(S',a)\big) - Q_2(S,A)\Big)$$

    $S \leftarrow S'$
  until S is terminal

图 13: Alg: Double Q-learning

n-step 方法

TD 方法与 MC 方法的折中

TD 方法中，只得到一步的回报就对值函数进行更新，MC 方法中，模拟得到了所有的回报后才对值函数进行更新。这两种方法并不都总是最优的，如果将 TD 方法的视野延长，或将 MC 方法的视野截断，得到的是 n-step TD 方法。

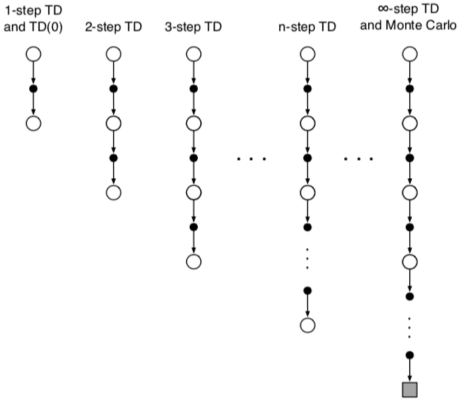


图 14: 视野的长短

n-step TD 值函数估计

定义 n-step 回报

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}) \tag{24}$$

适用于所有的 $n \geq 1$ 和 $0 \leq t < T - n$。当 $t + n \geq T$ 时，缺失项定义为 0，此时 $G_{t:t+n} = G_t$。更新式

$$V_{t+n}(S_t) = V_{t+n-1}(S_t) + \alpha(G_{t:t+n} - V_{t+n-1}(S_t)), 0 \leq t < T \tag{25}$$

n-step TD 值函数估计

Initialize $V(s)$ arbitrarily, $s \in \mathcal{S}$
Parameters: step size $\alpha \in (0,1]$, a positive integer n
All store and access operations (for S_t and R_t) can take their index mod n

Repeat (for each episode):
    Initialize and store $S_0 \neq$ terminal
    $T \leftarrow \infty$
    For $t = 0, 1, 2, \ldots$:
    | If $t < T$, then:
    | Take an action according to $\pi(\cdot|S_t)$
    | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
    | If S_{t+1} is terminal, then $T \leftarrow t+1$
    | $\tau \leftarrow t - n + 1$ (τ is the time whose state's estimate is being updated)
    | If $\tau \geq 0$:
    | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
    | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
    | $V(S_\tau) \leftarrow V(S_\tau) + \alpha \left[G - V(S_\tau)\right]$
    Until $\tau = T - 1$

图 15: Alg: n-step Value Estimation

n-step TD 值函数估计

n-step 方法误差既小于 TD 方法，又小于 MC 方法。

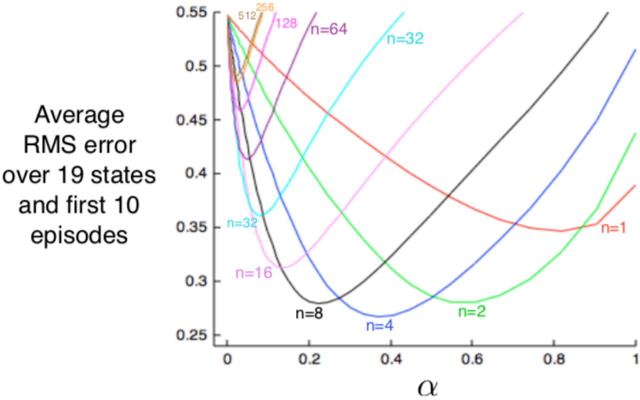


图 16: n 的取值与误差曲线的关系（随机行走）

n-step Sarsa

n-step 方法也可以自然地融入 Sarsa 算法。只要将更新式改为

$$Q_{t+n}(S_t, A_t) = Q_{t+n-1}(S_t, A_t) + \alpha(G_{t:t+n} - Q_{t+n-1}(S_t, A_t)) \tag{26}$$