第四讲 无模型预测 (Model-Free Prediction)

4.1 简介

上一讲主要内容是在已知环境模型的情况下，通过动态规划来求解一个马尔科夫决策过程（MDP）问题，主要介绍了策略迭代和价值迭代两种方法。而本讲的主要内容是在环境模型未知的情况下，来估计一个未知MDP的价值函数，也称为无模型预测。在下一讲中来优化这个未知MDP的价值函数，也称为无模型控制。

本讲内容分为三个小部分，分别是蒙特卡洛强化学习、时序差分强化学习和介于两者之间的λ时序差分强化学习。

4.2 蒙特卡洛学习（Monte-Carlo Learning）

蒙特卡洛（MC）方法只需要经历（experience），即从智能体实际（或模拟）与环境交互中采样序列：状态，动作，奖励。也就是直接从经历的完整的回合（Episode）中来学习。蒙特卡洛的思想很简单，某状态的价值等于多个采样回合中该状态的收获的均值。

> 注：MC方法只能用于完整回合的MDP中，即所有的回合都有终止状态。

定义： 完整回合是智能体从某个状态开始与环境交互，直到终止状态。例如，玩游戏的智能体直到赢了或输了构成一个完整回合。

蒙特卡洛学习的特点：不基于模型本身，直接从实际经历过的回合中学习，且必须是完整的回合，其基本思想就是用平均收获值来近似价值。理论上经历越多，结果越接近真实的价值。

1. 蒙特卡洛策略评估 Monte-Carlo Policy Evaluation

目标：从给定策略下的多个经历回合中学习该策略下的状态价值函数，即

$$S_1, A_1, R_2, \ldots, S_k \sim \pi$$

t时刻状态 S_t 的回报：

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

状态s的价值函数为回报的期望：

$$v_\pi = \mathbb{E}_\pi [G_t | S_t = s]$$

蒙特卡洛策略评估用经验回报的均值，而不是回报的期望，来计算状态的价值函数。也就是智能体和环境交互后，产生多个完整的经历回合，计算每个回合状态s的回报，再求平均。

在一个回合中，有些状态可能出现多次，即一个状态经过状态转移后又返回该状态。在这个回合中如何计算这个状态的收获呢？有两种方法：首次访问蒙特卡洛策略评估和每次访问蒙特卡洛策略评估。

2. 首次访问蒙特卡洛策略评估

对于每一个回合，仅当该状态**第一次**出现在回合中时列入计算:

- 状态出现次数加1： $N(s) \leftarrow N(s)+1$
- 增加总的收获：$S(s) \leftarrow S(s)+G_t$
- 用回报的均值估计状态价值：$V(s)=S(s)/N(s)$
- 根据大数定律： 随着$N(s)$趋近无穷大，$V(s)$趋近于$V\pi(s)$

3. 每次访问蒙特卡洛策略评估

对于每一个回合，当该状态**每一次**出现在回合中时列入计算:

- 状态出现次数加1： $N(s) \leftarrow N(s)+1$
- 增加总的收获：$S(s) \leftarrow S(s)+G_t$
- 用回报的均值估计状态价值：$V(s)=S(s)/N(s)$
- 如上，根据大数定律： $V(s) \rightarrow V\pi(s)\ as\ N(s) \rightarrow \infty$

例4-1 二十一点游戏 Blackjack
蒙特卡洛方法的使用可以很好地通过该例子来示范。

游戏基本信息：

牌面为：2，3，4，5，6，7，8，9，10，J，Q，K, A
牌的点数：

- 人头牌（face card）J、Q、K点数都计为10
- 牌面为2-10的点数等于面数
- 牌面A的点数为1或11

参与者：玩家（player）和庄家（dealer）
游戏目的：得到点数尽量接近21点的牌，但不得超过，并使玩家的点数大于庄家。

游戏规则：

游戏开始时每人得到两张牌，庄家开始的两张牌一张明、一张暗。

如果开始两张牌的总点数恰为21（A-10或A-花牌），称为“天成（natural）”，自动成为胜者（若玩家和庄家都得到21点，则为平局，玩家的赌注仍在台上）。

如果玩家前两张牌不是21点，可以一张一张的叫牌（hits）,直到玩家打住（stick）或超过21点输了（go bust）。

如果玩家打住，就轮到庄家叫牌，庄家叫牌或打住根据固定的策略：

- 当庄家的点数大于等于17点时，打住
- 否则就叫牌（小于等于16点叫牌）
  庄家总把A的点数记为11，除非这样使他超过21（这时A的点数记为1）
  如果庄家超过21点，玩家赢；其他情况，根据最后谁的总点数接近21点分为：赢，输，平。

游戏建模

二十一点游戏可以建模成一个回合有限长度的MDP。每一局是一个回合。赢，输，平分别对应的奖励为1，-1，0。游戏进行中的所有奖励为0，且不考虑带折扣系数（$\gamma = 1$）。因此，终止奖励就是回报。假设是无限发牌的，所以记住已发的牌得不到优势。不超过21点时，玩家总是把A当作11点，称为A是可用的（usable）。这种情况下总是把A当作11点，因为如果当作1点，总点数会小于等于11点。在小于11点时，玩家总是叫牌，没有其他决策。因此玩家的决策基于三个变量：玩家当前的总点数（12-21）点，庄家的一张明牌（A-10），玩家是否有可用的A。

状态空间：（多达200种，根据对状态的定义可以有不同的状态空间，这里采用的定义是牌的分数，不包括牌型）

1. 当前牌的点数（12 - 21），低于12时，玩家可以安全的再叫牌，所以没意义。
2. 庄家的明牌（A - 10），庄家会显示一张牌面给玩家
3. 玩家是否有“useable” ace”， A既可以当1点也可以当11点。

行为空间：

1.停止要牌 stick
2.继续要牌 twist

奖励（停止要牌）：

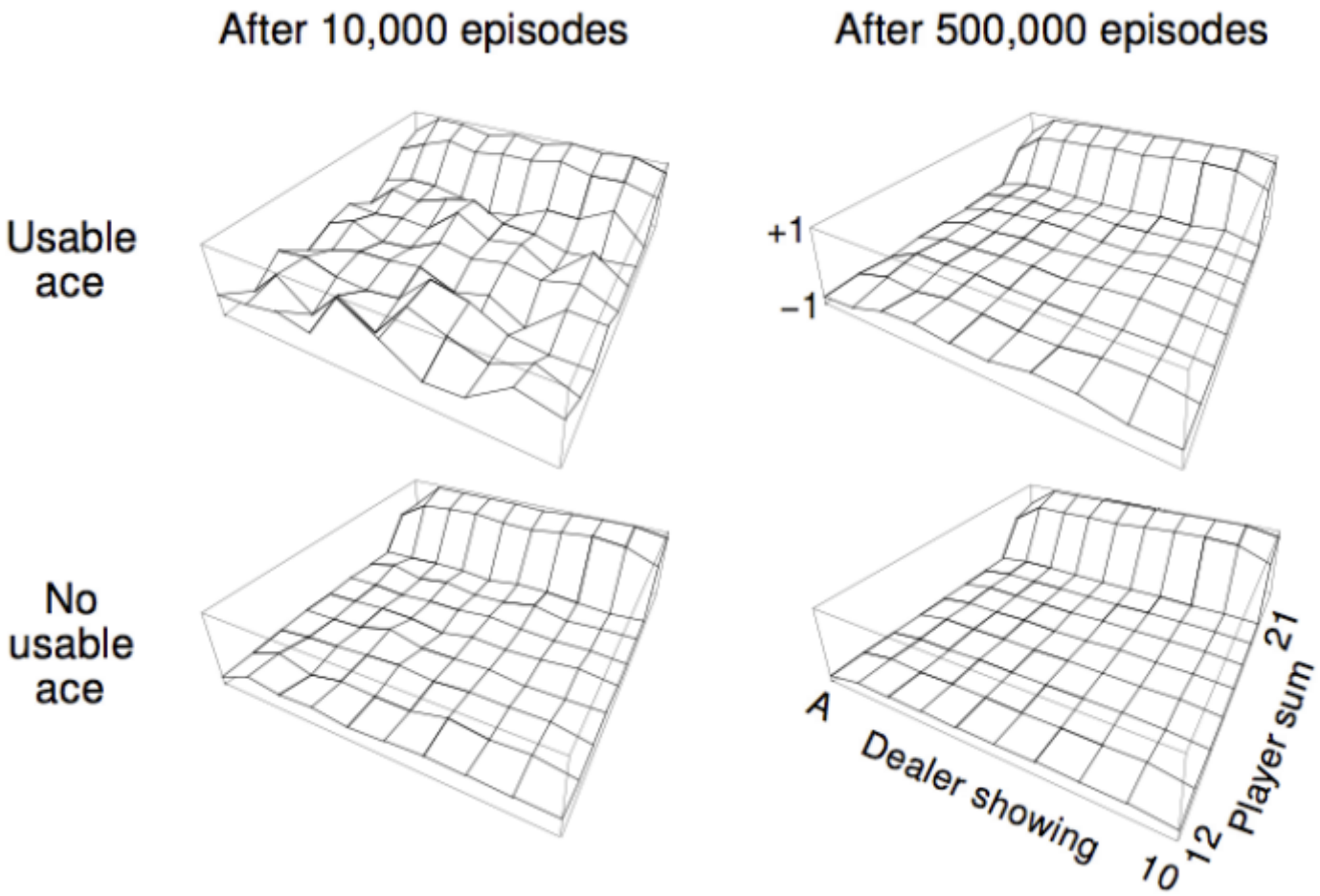
（1）+1：如果玩家的牌点数大于庄家点数
（2） 0： 如果两者点数相同
（3）-1：如果玩家的点数小于庄家点数

状态转换（Transitions）： 如果牌点数小于12时，自动要牌

当前策略： 牌点数只要小于20就继续要牌。

求解问题： 评估该策略的好坏

结果如下图：无论玩家手中是否有A牌，该策略在绝大多数情况下各状态价值都较低，只有在玩家拿到21点时状态价值有一个明显的提升。可以看出，牌的点数小于20继续要牌不是一个很好的策略，因为很容易超过21点而输。



为了更直观的理解本例中的MC方法，我们模拟多个二十一点游戏牌局信息（多个回合MDP），假设我们仅研究初始状态下庄家一张明牌为10，玩家手中前两张牌和为14的情形。在给定策略下，小于20点玩家继续要牌，则可能会出现如下多种情形，如表4-1所示：

牌局	庄家最终序列	玩家最终序列	玩家获得奖励	估计的状态价值
1	10，5，2	6，8，2，4	+1	1
2	10，4	5，9，Q	-1	0
3	10，3，J	4，K，6	+1	0.333
4	10，6，5	3，A，J，6	-1	0
5	J，9	4，10，5，6	-1	-0.2

牌局	庄家最终序列	玩家最终序列	玩家获得奖励	估计的状态价值
...	...	...	...	...

可以看到，采取当牌点数小于20就继续叫牌这个策略，前5次平均价值为-0.2，如果玩的牌局足够多，按照这样的方法可以针对每一个状态（庄家第一张明牌，玩家手中前两张牌的合计点数）都可以制作这样一张表，进而计算玩家奖励的平均值。通过结果，可以发现这个策略并不能带来很高的玩家奖励。

第一个对局对应的回合信息，如表4-2所示：

回合序列	S_0	A_0	R_1	S_1	A_1	R_2	S_2	R_3
序列值	(10,14)	要牌	0	(10,16)	要牌	0	(10,20)	+1

从上面信息可以看出，这个完整的回合中包含三个状态，其中前两个状态的即时奖励为0，第三个状态是终止状态，根据规则，玩家赢得对局，获得终止状态的即时奖励+1。

在使用蒙特卡洛方法求解平均收获时，需要计算平均值。通常计算平均值要预先存储所有的数据，最后用收获总和除以回合总数。如表4-1中最后两列所示，这样每增加一个回合，就需要计算一次收获总和及回合总数，再相除。下面介绍了一种更简单实用的方法

4. 累计更新平均值（Incremental Mean）

序列 $x_1, x_2, \ldots$ 的均值 $\mu_1, \mu_2, \ldots$ 可以累计计算，这是一个实时更新均值的方法：

$$
\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} X_j \\
&= \frac{1}{k}(X_k + \sum_{j=1}^{k-1} X_j) \\
&= \frac{1}{k}(X_k + (k-1)\mu_{k-1}) \\
&= \mu_{k-1} + \frac{1}{k}(X_k - \mu_{k-1})
\end{aligned}
$$

如表4-1中收获分别为：1，-1，1，-1，-1，收获均值为1，0.0，0.333，0.0，-0.2。利用上述公式：
第二局收获均值： $\mu_2 = \mu_1 + \frac{1}{2}(X_2 - \mu_1) = 1 + 0.5(-1 - 1) = 0$
第三局收获均值： $\mu_3 = \mu_2 + \frac{1}{3}(X_3 - \mu_2) = 0 + 0.333(1 - 0) = 0.333$
...

因此，不再需要存储所有的收获，只需要上一个回合的均值，就可以实时更新收获均值。把这个方法应用于蒙特卡洛策略评估，就得到下面的蒙特卡洛累计更新。

5.蒙特卡洛累进更新

- 对于一系列回合（Episodes）中，每增加一个回合， $S_1, A_1, R_2, \ldots, S_T$ 累计地更新V(s)的值。
- 对于回合中的每一个状态 S_t ，有一个收获 G_t ，每碰到一次 S_t ,使用下式计算状态的平均价值 $V(S_t)$ ：

$$N(S_t) \leftarrow N(S_t) + 1$$
$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$$

在处理非静态问题时，使用这个方法跟踪一个实时更新的平均值是非常有用的，可以扔掉那些已经计算过的Episode信息。此时可以引入参数 α 来更新状态价值：

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

到此，我们介绍了MC学习方法的主要内容。因为MC方法只能用于完整的回合，即所有的回合都要终止。这样才能计算回报Gt，因为Gt是从t时刻开始，对整个回合的奖励进行累加。显然，这种方法计算量会比较大，实际应用不多。所以，引出了时序差分方法

4.3 时序差分学习（ Temporal-Difference Learning）

时序差分（TD）学习是蒙特卡洛思想和动态规划思想的完美结合。一方面，像蒙特卡洛方法一样，时序差分不需要知道环境的动力学模型，可以从经历的回合中直接学习；另一方面，像动态规划一样，时序差分更新估计值基于部分已知的估计值，不需要等到最后的结果（完整的回合），这就是引导（bootstrap）的思想。因此，TD是无模型的，通过引导，从不完整的回合中学习，用一个估计来更新另一个估计。

4.3.1 MC 和 TD

- 目标：TD和MC都是用经历来求解预测问题。从遵循策略 π 的经历中来学习出现在经历中的非终止状态 S_t 的值 V_π 。
- 大体上说，MC方法是等到该状态的回报已知后，以该回报 G_t 为目标，更新状态价值函数 $V(S_t)$ 朝着实际的回报 G_t 逼近，即

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

其中， α 是步长参数，是一个常量。

- 然而，在TD学习中，只需要等到下一步，即在 $t+1$ 时刻，就立刻确定了一个目标，用观测到的奖励 R_{t+1} 和下一状态的价值函数的估计值 $V(S_{t+1})$ 来更新状态价值函数 $V(S_t)$ ，即

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

式中，
$R_{t+1}+\gamma V(S_{t+1})$ 称为TD目标
$\delta_t = R_{t+1}+\gamma V(S_{t+1}) - V(S_t)$ 称为TD误差

- 与MC的目标 G_t 相比，TD的目标为 $R_{t+1}+\gamma V(S_{t+1})$ 。
- 这种TD称为TD(0)或one-step TD，因为它是后面即将介绍的TD (λ) 和n-step TD的特殊情况。
- 因为TD(0)更新的一部分是基于现有的估计，称之为引导（BootStrapping），即TD（0）目标的一部分是 $V(S_{t+1})$ ，而 $V(S_{t+1})$ 是未知的，用的是现有的估计值 $V(S_{t+1})$ 。

MC和TD的更新规则都如下：

> 新估计 ← 旧估计 + 步长*(目标 - 旧估计)

式中，（目标-旧估计）称为在估计中的误差，这个误差通过朝目标走一步来减小，假设目标指明了一个向往的方向。

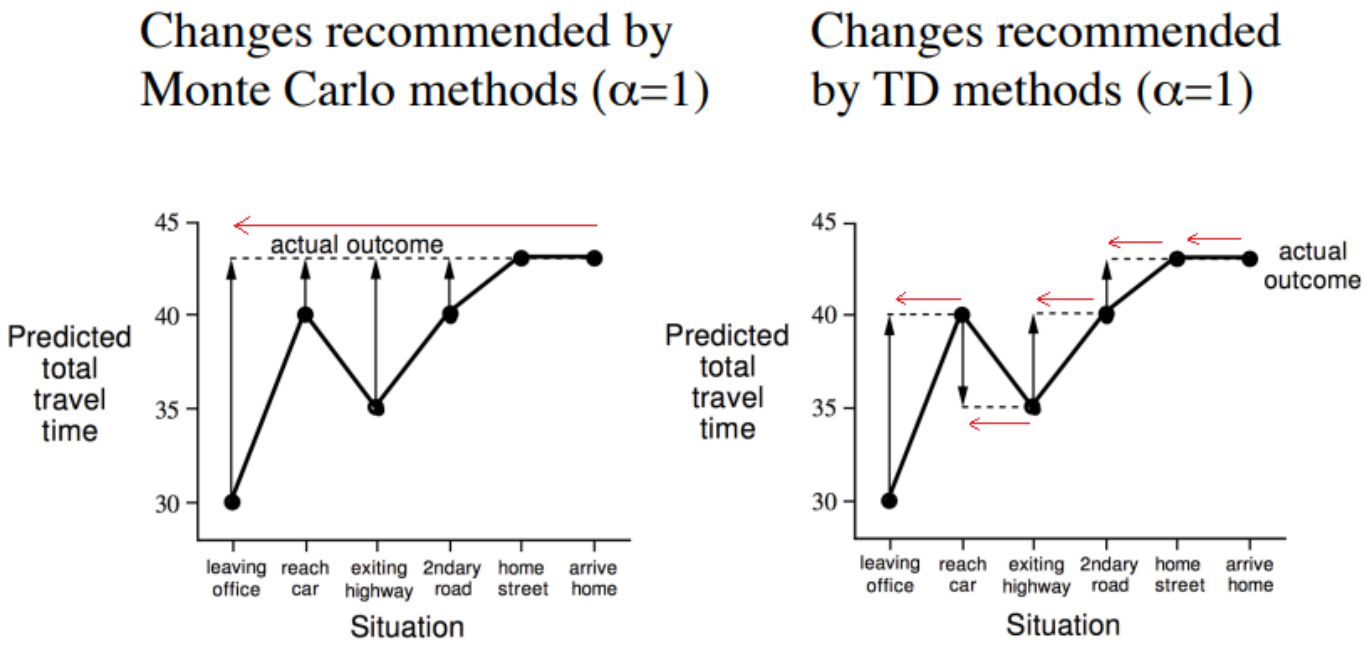
例4-2 开车回家，用来解释MC策略评估和TD策略评估的差别

你每天下班开车回家，你试图预测开车回家需要花费多长时间。
在回家的路上你会依次经过一段高速公路、普通公路、和你家附近街区三段路程。由于你经常开车上下班，在下班的路上多次碰到过各种情形，比如取车的时候发现下雨，高速路况的好坏、普通公路是否堵车等等。在每一种状态下时，你对还需要多久才能到家都有一个经验性的估计。下表中的“预计仍需耗时”列给出了这个经验估计，这个经验估计基本反映了各个状态对应的价值，通常你对下班回家总耗时的预估是30分钟。
假设这周五下午六点，你又下班准备回家了，当花费了5分钟从办公室到车旁时，发现下雨了。此时根据经验，下雨会慢一些，估计还需要35分钟才能到家，因此整个行程将耗费40分钟。随后你进入了高速公路，高速路况非常好，行驶了15分钟就离开了高速公路，此时已经耗时 20 分钟。根据经验只需要再开 15 分钟就能到家，此时将这次回家的预计总耗时修正为35分钟，比取车时的估计少了5分钟。但是当你进入普通公路时，发现交通流量较大，不得不跟在一辆卡车后面龟速行驶，这个时候距离出发已经过去30分钟了，根据以往的经验，你还需要10分钟才能到家，现在对于回家总耗时的预估又回到了40分钟。最后在出发40分钟后到达了家附近的街区，根据经验，还需要3分钟就能到家，此后没有再出现新的情况，最终你在43分钟的时候到达家中。

状态	已耗时间	预计仍需耗时	预计总耗时
离开办公室	0	30	30
取车时下雨	5	35	40
离开高速	20	15	35
跟在卡车后面	30	10	40

状态	已耗时间	预计仍需耗时	预计总耗时
家附近街区	40	3	43
回到家中	43	0	43

每个状态的奖励为已经耗时，即第一列。每个状态的价值为期望仍需耗时。遇到每个状态时，当前估计的状态价值（即预计仍需耗时）在上面表格第二列中给出。下图根据序列中各状态时预计的总耗时（上表最后一列）绘制。



如上图左边所示，使用MC算法，箭头表示各状态预计仍需耗时和实际仍需耗时的误差。例如，当你下高速时，已经耗时20分钟，你认为仅需再行驶15分钟就能到家，而实际上需要23分钟，这个误差为8分钟。在整个驾车返家的过程中，你对于所处的每一个状态，例如“取车时下雨”，“离开高速公路”，“被迫跟在卡车后”、“进入街区”等时，都不会立即更新这些状态对应的返家还需耗时的估计。这些状态的返家仍需耗时仍然分别是先前的35分钟、15分钟、10分钟和3分钟。但是当你到家发现整个行程耗时43分钟后，通过用实际总耗时减去到达某状态的已耗时，你发现在本次返家过程中在实际到达上述各状态时，仍需时间则分别变成了：38分钟（43-5）、23 分钟（43-20）、13 分钟（43-30）和3分钟（43-40）。在任何状态中，预计仍需耗时只能离线更新，即直到你回到家后，才能对每个状态预计仍需耗时进行更新，因为此时，你才知道真正的回报。

如上图右边所示，使用TD算法时，使用下一个状态的预估价值函数，来更新当前状态的的价值函数，即图中虚线到箭头所示。当取车发现下雨时，根据经验你会认为还需要35分钟才能返家，此时，你将立刻更新从离开办公室到回家总耗时的估计，为仍需的35分钟加上你离开办公室到取车现场花费的5分钟，即40分钟。同理，当驶离高速公路，根据经验，你对到家还需时间的预计为15分钟，但由于之前高速路况较好，在第20分钟时已经驶离高速，则此时你又立刻更新了取车时下雨这个状态，预估回家所需总耗

时更新为35分钟，向下箭头所示。当你在驶离高速在普通公路上又行驶了10分钟被堵，你预计还需10分钟才能返家时，你对于刚才驶离高速公路返家还需耗时又做了更新，将不再是根据既往经验预估的15分钟，而是现在的20分钟，加上从出发到驶离高速已花费的20分钟，整个行程耗时预估因此被更新为40分钟。直到你花费了40分钟才到达家附近的街区，还预计有3分钟才能到家时，你更新了在普通公路上对于返家还需耗时的预计为13分钟。最终你按预计3分钟后进入家门，不再更新剩下的仍需耗时。

从上述分析可知，TD方法中，当后一个状态的估计更新后，前一个状态的估计随之朝着后一个状态的估计方向逼近。例如，出高速时已耗时20分钟，根据高速路况较好，估计此时回家仍需15分钟，然后，立即更新前一个状态，即取车下雨这个状态的价值，即此状态下预估回家的总时间如图中箭头下降到了35，也就是从取车到家仍需时间的估计从35变为了30（取车的状态价值），把高速路况好的信息体现在这个状态的估计中，取车时的估计误差为-5，即对此时回家所需时间多估计了五分钟。图中箭头的长度表示为估计的误差，每个状态的误差与每个状态预测随时间的变化成正比，即与预测的时序差异成正比。所谓预测的时序差异指的是，根据下一个状态的预测更新本状态的预测后与在本状态时的预测的差异。

2. MC，DP 和 TD

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t|S_t=s] && (1)\\
&= \mathbb{E}_\pi[R_{t+1}+\gamma G_{t+1}|S_t=s] && (2)\\
&= \mathbb{E}_\pi[R_{t+1}+\gamma V_\pi(t+1)|S_t=s] && (3)
\end{aligned}
$$

通俗地说，MC以估计式（1）作为目标，而DP以估计式（3）为目标。MC目标是一个估计值，这是由于式（1）中的期望值是未知的，用采样回报的均值来估计回报的期望的真实值。而DP目标为一个估计值，不是因为期望值，而是因为 $V(S_{t+1})$ 是未知的，用的是现有的估计值 $V(S_{t+1})$ 。TD目标是一个估计值由两个因素导致：首先，对式（3）做采样，其次，TD用的是当前的估计值V而非真实值 v_π 。因此，TD把MC的采样同DP的引导结合起来。所以，TD同时具备MC和DP的优点。

3 MC和TD的优缺点对比

TD 可以在知道最终结果之前可以学习，

- 可以在每一步之后在线学习
- MC必须等到回合的最后，知道回报才能学习

TD 可以从没有最终结果的序列中学习

- TD从不完整序列中学习
- MC只能从完整的序列中学习
- TD可以在持续进行（没有终止）的环境里学习
- MC只能在回合（有终止状态）的环境中学习

偏差与方差的权衡 （Bias/Variance Trade-off)

- 回报 G_t ，是在遵循某一策略下，状态价值 $V_\pi(S_t)$ 的无偏估计。
- 真实的TD目标 $R_{t+1}+\gamma V_\pi(S_{t+1})$ 是 $V_\pi(S_t)$ 的无偏估计。
- TD目标 $R_{t+1}+\gamma V(S_{t+1})$ 是 $V_\pi(S_t)$ 的有偏估计。
- TD目标的方差比回报的方差小很多，这是因为回报依赖于许多随机的动作，随机的转移，随机的奖励，而TD 目标仅依赖于一个随机的动作，随机的转移，随机的奖励。

MC和TD的优缺点对比二

- MC 方差大，无偏差
  MC有很好的收敛性（即使采用函数逼近），对初始值不敏感，理解和使用比较简单
- TD方差小，但有一些偏差
  TD通常比MC更高效，TD(0)收敛于 $V_\pi(S_t)$ （函数逼近时不一定），TD对初始值比较敏感

例4-3 随机行走,用来对比MC和TD在下图中MRP的预测能力

状态空间： 如下图：A、B、C、D、E为中间状态，C同时作为起始状态。灰色方格表示终止状态；

即时奖励： 右侧的终止状态得到即时奖励为1，左侧终止状态得到的即时奖励为0，在其他状态间转化得到的即时奖励是0；

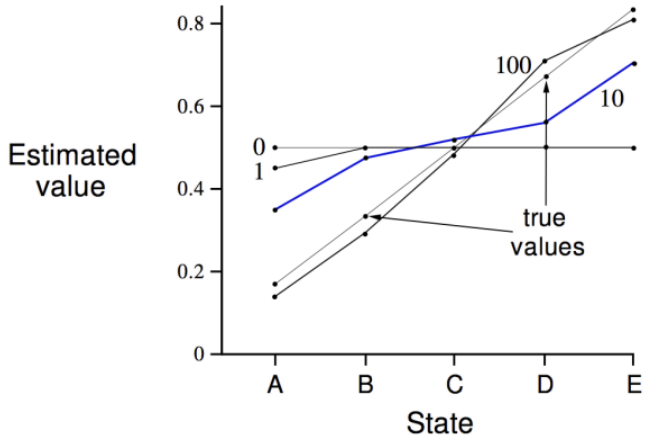
状态转移： 任意状态下等概率向左或向右转移一步，进入终止状态即终止；

折扣系数： 1；

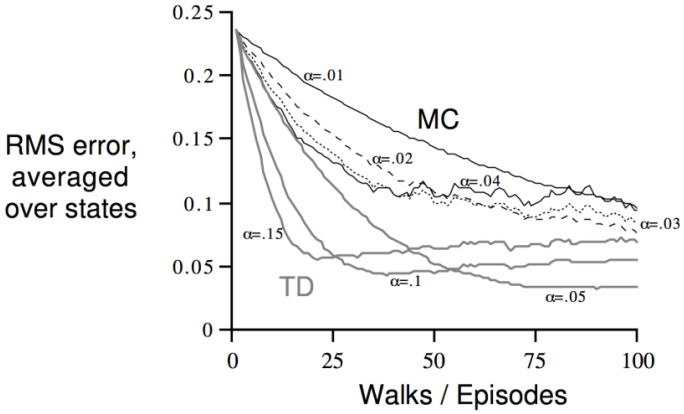
给定的策略： 随机选择向左、向右两个行为。

问题： 对这个MRP问题进行预测，也就是评估随机行走这个策略的状态价值，即确定该MRP问题的状态价值函数。由于MRP是无折扣的，且只有右边的终止状态奖励为1，其他奖励为0，所以，某个状态的真实价值就是假如从该状态开始，终止于右边终止状态的概率。因此，A到E的真实状态价值为1/6,2/6,3/6,4/6,5/6。

求解： 下图是使用TD算法得到的结果。横坐标显示的是状态，纵坐标是各状态的价值估计，一共5条折线，数字表明的是实际经历的Episode数量，true value所指的那根折线反映的是各状态的实际价值。第0次时，各状态的价值被初始化为0.5，经过1次、10次、100次后得到的价值函数越来越接近实际状态价值函数。



下图比较了MC和TD算法在不同步长参数 α 下的学习曲线。横坐标是经历的episode数量，纵坐标是计算得到的状态函数和实际状态函数下各状态价值的均方差(五个状态的平均，然后跑了100次做平均)。黑色是MC算法在不同step-size下的学习曲线，灰色的曲线使用TD算法。可以看出TD较MC更高效。此图还可以看出当step-size不是非常小的情况下，TD有可能得不到最终的实际价值，将会在某一区间震荡。



4.3.2 Batch MC和TD

在理想情况下，即经验次数趋于无穷大时，MC和TD(0)所估计得到的价值函数都将收敛于真实的价值函数，数学描述为：

$$V(S_t) \rightarrow V_\pi(S_t) \; as \; experience \rightarrow \infty$$

但在实际中，我们只有有限的经验，如10个回合或100个时间步。此时，对于增量学习方法来说，常用的手段就是重复利用这些经验，直到方法收敛于一个答案。如对于TD，价值函数更新如下，

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

非终止状态访问每个时间步t时，计算增量 $\alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ ，但是价值函数只改变一次，也就是用所有增量的和来更新一次。然后，所有有限经验被新的价值函数再一次用来计算一个新的总增量，更新价值函数，如此迭代，直到价值函数收敛。把这个过程称为Batch Updating，因为更新只发生在处理完每一批完整的训练数据后。

例如，有下面这K个回合：

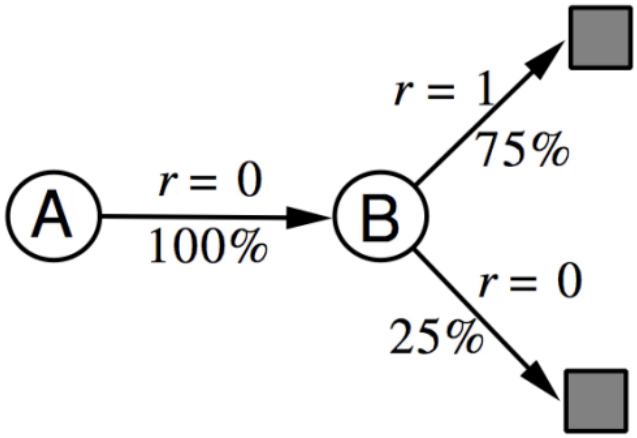
$$s_1^1, a_1^1, r_2^1, \ldots, s_{T_1}^1$$
$$\ldots$$
$$s_1^K, a_1^K, r_2^K, \ldots, s_{T_K}^K$$

我们重复地采样回合 $k \in [1, K]$，对于回合k应用MC或TD方法，有什么结论呢？对于TD来说，确定会收敛于一个答案，且和步长无关。而MC方法在一些条件下，确定会收敛，但是收敛的答案不同。

例4-4 AB两状态MRP，
现有两个状态(A和B)的MRP，衰减系数为1，8个完整回合的经验及对应的即时奖励如下表，其中第1个回合从状态A开始，转移到状态B，奖励为0，然后在B终止，奖励为0。其余7个均只有一个状态，即从B开始,立即终止于B。如下表所示：

回合	状态转移及概率
1	A，0,B，0
2	B，1
3	B，1
4	B，1
5	B，1
6	B，1
7	B，1
8	B，0

问题：依据上面有限的经验，用MC和TD计算状态A，B的价值分别是多少，即V(A)=？， V(B)=？
答案：显而易见V(B) = 6/8，而V(A)根据不同算法结果不同，用MC算法结果为0，TD则得出6/8。
分析：应用MC算法，需要完整的回合,因此仅回合1可以用来计算A的状态价值，且从A开始的回报为0，V(A)=0；同时B的价值是6/8。应用TD算法时，TD算法试图利用现有的回合经验构建一个MRP（如下图），由于存在一个回合使得状态A有后继状态B，因此状态A的价值是通过状态B的价值来计算的，且A状态的即时奖励是0，且没有折扣，因此A的状态价值等于B的状态价值，这也是Batch TD的答案。



5 确定性等价估计 Certainty Equivalence estimate

- MC算法收敛至一个最小化均方差的解，即状态价值与实际收获的均方差，这个解能够很好的拟合观测到的收获，均方差为：

$$\sum_{k=1}^{K}\sum_{t=1}^{T_k}(G_t^k - V(s_t^k))^2$$

AB 例子中，V(A)=0
式中， k 表示的是回合序号， K为总的回合数量，t为一个回合内状态序号（第1,2,3...个状态等），T_k 表示第k个回合总的状态数，G_t^k 表示第k个回合里t时刻状态 S_t 获得的最终收获，$V(s_t^k)$ 表示的是第k个回合里算法估计的t时刻状态 S_t 的价值。

- TD算法收敛至一个最大似然马尔可夫模型的解，MDP $< S, A, \widehat{P}, \widehat{R}, \gamma >$ 的解很好地拟合数据。TD算法首先根据已有经验估计状态间的转移概率，

$$\widehat{P}_{s,s'}^a = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k} 1(s_t^k, a_t^k, s_{t+1}^k = s, a, s')$$

同时估计某一个状态的即时奖励，

$$\widehat{R}_s^a = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k} 1(s_t^k, a_t^k = s, a) r_t^k$$

AB 例子中，V(A)=0.75

可以理解为，MC方法的解对现有的数据表现得很好，而TD方法的解对未来数据产生较小的误差。

TD和MC的对比

TD算法利用了马尔可夫性质，通常在马尔可夫环境下更有效；而是MC算法没有利用马尔可夫性质，通常在非马尔可夫环境下更有效。

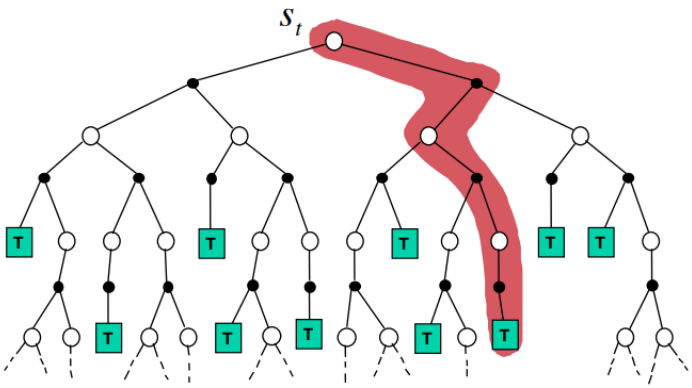
4.3.3 统一视图（Unified View）

DP，MC，DP是计算状态价值的三种方法。DP算法需要知道环境的动力学模型，通过计算一个状态S所有可能的转移状态S'及其转移概率以及对应的即时奖励来计算这个状态S的价值，即通过一步引导。而MC和TD都是不知道模型的情况下，估计状态S的价值。其中，MC需要一个完整的回合来估计状态价值，因为MC用采样回报作为目标。TD方法结合了DP的一步引导和MC的采样，用下一个状态的估计来估计当前状态的价值，所以不需要知道完整的回合。
引导就是用估计来更新。采样就是用采样的均值来更新。关于三者是否有引导和采样的对比如下，

- MC没有引导，DP和TD利用引导。
- DP没有采样，MC和TD利用采样
  TD结合了引导和采样的优点。三者的备份图如下：

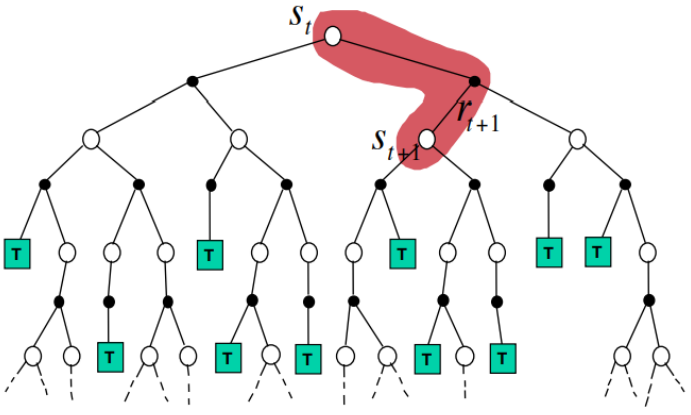
1. MC备份,采样，一次完整经历，用采样的收获更新状态预估价值

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



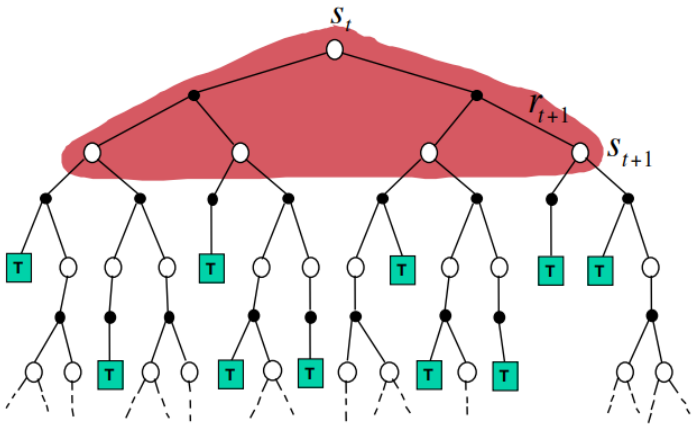
1. TD备份，经历可不完整，用下一个状态的预估状态价值来更新当前状态的预估状态价值

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

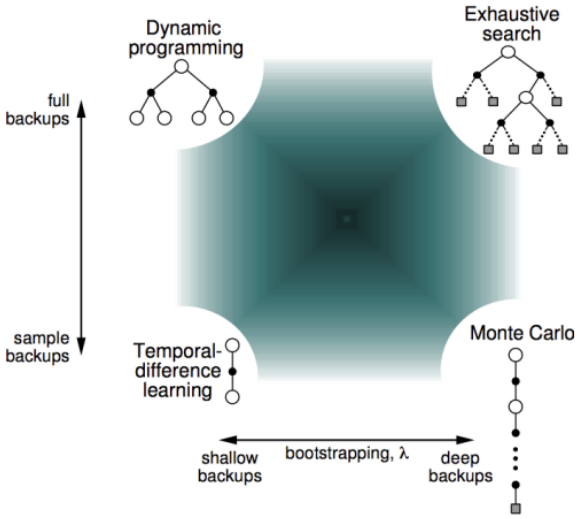


1. DP备份，没有采样，根据完整模型，用预估数据来更新状态价值

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1}) \right]$$



强化学习统一视图



上图中包含的前面介绍的三种算法，再加一个穷举法。从两个维度描述了四种算法的联系与差别。这两个维度分别是：采样深度和广度。当使用单个采样，不走完整个回合就是TD；当使用单个采样但走完整个回合就是MC；当考虑全部样本可能性，但对每一个样本并不走完整个回合时，就是DP；当既考虑所有回合又把回合从开始到终止遍历完，就变成了穷举法。

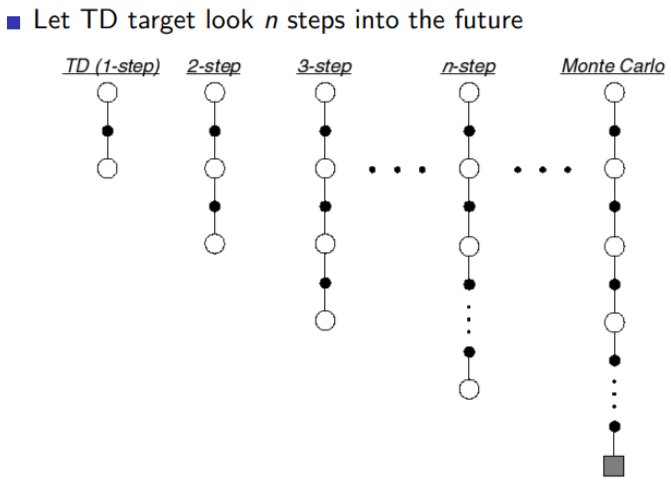
注：DP利用的是整个MDP模型，也就是状态转移概率，虽然它并不实际利用样本，但是它利用了整个模型的规律，因此认为是Full Width的。

4.4 λ时序差分强化学习 （TD(λ))

之前介绍的TD称为一步TD或TD(0)，更新基于下一步奖励和下一个状态的价值函数的估计（下一个状态价值的估计可以看作从下一步开始所有剩余奖励的近似），而MC更新用了一个回合的所有奖励。简单地说，就是TD用了一步奖励及状态估计来更新，MC用了整个回合的所有奖励来更新。一个折中的方法就是，更新价值基于中间步数的奖励，这个步数大于1，但小于整个回合步数,即步数介于TD(0)和MC之间。如，两步更新基于前两个奖励和两步之后的状态的价值估计。这就引入了n-step TD的概念，基本思想是更新的时候比TD看得更远一些。

1. n步TD（n-step TD）

从当前状态往前n步，计算n步的return，同样TD target 也由2部分组成，已走的步数使用确定的即时reward，剩下的使用估计的状态价值替代。



注：空心圆圈表示状态，实心小圆圈表示动作

n步回报（n-step Return）

考虑n= 1，2， ∞ 时，n-step回报。D(0)是基于1-步预测的，MC则是基于∞-步预测的

步数n	更新名称	回报
n=1	TD or TD(0)	$G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1})$
n=2	n-step TD	$G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$
...	n-step TD	...
n= ∞	MC	$G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$

定义n-步收获：

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

n步TD学习状态价值函数的更新：

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t^{(n)} - V(S_t))$$

TD(0)和MC是 n-step TD的两个极端情况。

在线更新与离线更新

n-steps 备份（更新）计算一个增量：

$$\Delta_t(S_t) = \alpha\left[G_t^{(n)} - V_t(S_t)\right] \qquad \Delta_t(s) = 0, \forall s \neq S_t$$

在线更新为：

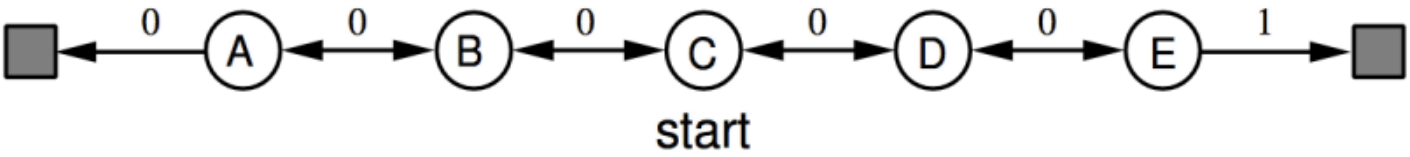
$$V_{t+1}(s) = V_t(s) + \Delta_t(s), \qquad \forall s \in \mathbb{S}$$

离线更新为：

$$V(s) \leftarrow V(s) + \sum_{t=0}^{T-1} \Delta_t(s) \qquad \forall s \in \mathbb{S}$$

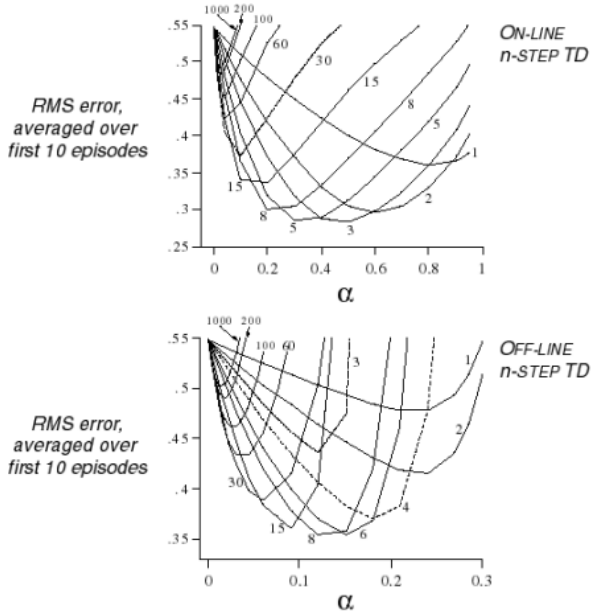
在线更新，每一个时间步更新一次价值函数；而离线更新，一个回合内计算所有的增量之和，一个回合只更新一次价值函数

例4-5 大规模随机行走，示例不同n值的效果



考虑把n-step TD方法使用在例4-4的五-状态的随机行走任务种。假设第一个回合从C开始，通过D,E，然后终止于右边,回报为1。回顾之前估计任何以中间点为起点的状态价值为V(s)=0.5。因此在这个实验中，一步TD只会改变最后一个状态的估计，即V(E)将会逼近观测回报值1。而两步TD将会更新终止状态的前两个状态，V(D)和V(E)，两个估计值都将逼近1。三步TD或n步TD （n大于2时），所有的三个状态值会逼近1，且每次更新的增量相同。那么n值多少时效果最好呢?

下图展示了19个状态的大规模随机行走的简单经验测试的结果（左边的结果为-1，所有价值初始化为0），实验使用多个不同步数预测联合不同步长（step-size，公式里的系数α）时，对于每个参数性能的衡量，体现在纵坐标，即在回合的末尾19个状态的估计和它们的真实值的均方误差的平方根（RMS error），实验使用前10个回合及重复100次的平均值。可以从下图看出，n取中间值时最有效。那么，如何将TD和MC方法推广到n步TD中,才可能使得效果比两种极端的方法都要好?



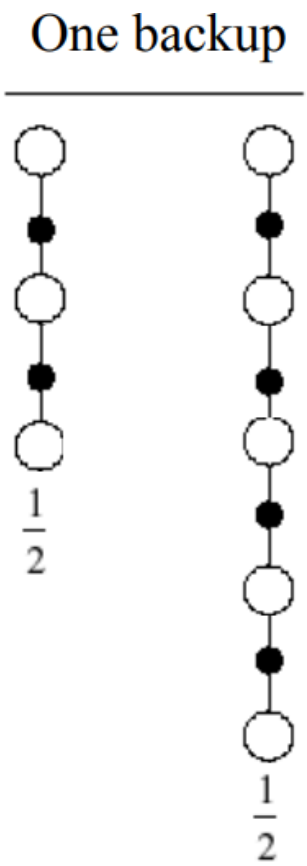
上下两个图分别表示离线与在线，其主要区别在于，离线是在经历所有10个Episode后进行状态价值更新；而在线则至多经历一个Episode就更新一次状态价值。离线和在线之间曲线的形态差别不明显；从步数上来看，步数越长，越接近MC算法，均方差越大。因此选择多少步数作为一个较优的计算参数也是一个问题。

n步回报求平均（averaging n-step returns）

我们可以对不同n的n步回报求平均，如求两步回报和四步回报的平均：

$$\frac{1}{2}(G^{(2)} + G^{(4)})$$

备份图如下，



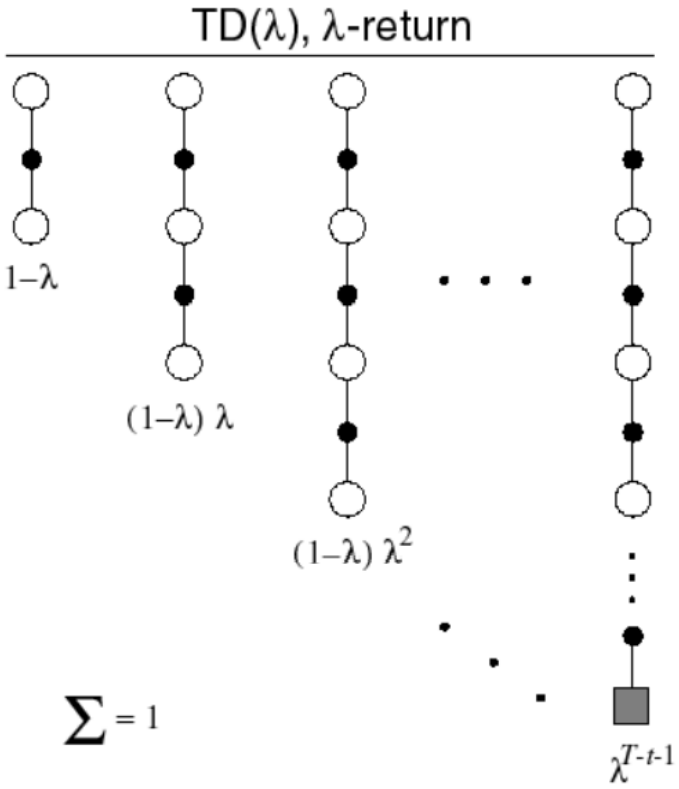
这样就把两种不同时间步的信息结合起来了，那么问题来了，我们能否把所有的时间步的信息有效地结合起来呢？这就引出了 λ 回报 （λ-return），即 λ 回报 G_t^λ 结合了所有n步回报 G_t^n ，而n步回报的权重为 $(1-\lambda)\lambda^{n-1}$ ， λ-return 如下:

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^n$$

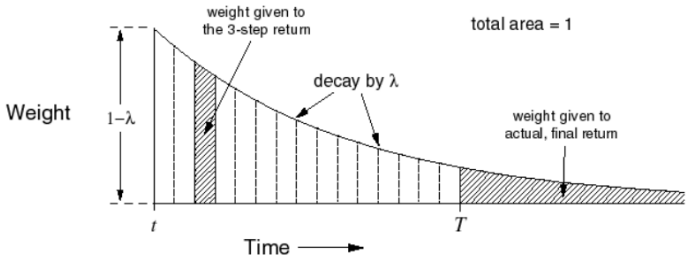
λ-return 对应的 λ 更新,又称为前向 $TD(\lambda)$,如下:

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t^\lambda - V(S_t))$$

备份图如下，



TD(λ)权重分配函数示意图，



$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^{(n)}$$

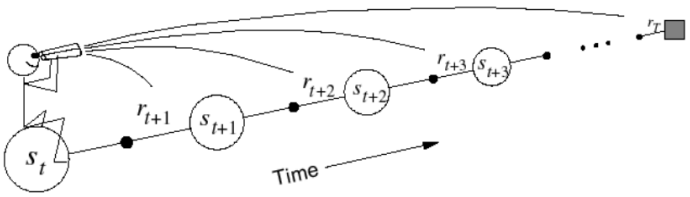
上图中描述了n步权重随着步数增加的变化，例如对于n=3的3-步收获，赋予其在λ收获中的权重如左侧阴影部分面积，对于终止状态的T-步收获，T以后的所有阴影部分面积。所有面积之和为1。这种几何级数的设计也考虑了算法实现的方便性。

与离线λ回报算法相比，TD((λ)改进了三个方面，首先，TD((λ)的权重向量不是在回合的结尾才更新，而是回合的每一步都更新。其次，其计算量不是都集中在回合的结尾，而是随时间平均分配。最后，不是只能应用于回合性的问题中，TD(λ)可以应用于连续的问题中。我们可以从两个方向来理解TD(λ)。

2. TD（λ）前向视角（Forward view）

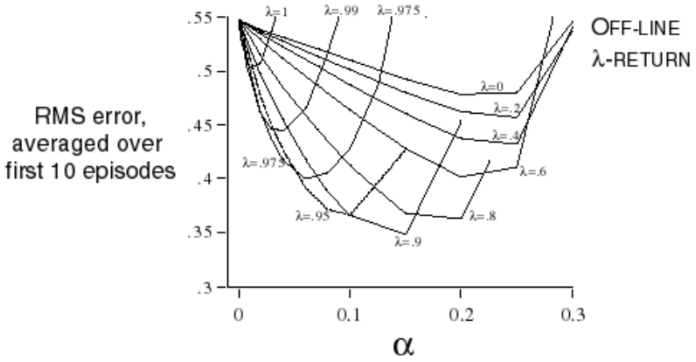
前向视角指用未来的奖励更新当前状态，即基于向前看来更新状态。在这里指计算回报 G_t^λ 时朝未来看，朝着λ回报方向更新价值函数。引入了λ之后，会发现要更新一个状态的状态价值，必须要走完整个回合，获得每一个状态的即时奖励以及最终状态获得的即时奖励。这和MC算法的要求一样，因此TD(λ)

算法有着和MC方法一样的劣势，只能工作在完整的回合的MDP。λ取值区间为[0,1]，当λ=1时对应的就是MC算法。



前向视角的解释：假设一个智能体处于状态St，当智能体更新状态St价值函数的估计时，智能体需要拿望远镜看向前方的状态（St的后续状态），用向前看到的奖励和后续状态价值来更新当前状态的价值函数。从TD(λ)的定义中可以看到，它需要用到将来时刻的值函数。

前向视角TD(λ)大规模随机行走的例子如下图，可以看出更像MC方法了，因为回报朝着MC方向靠近，远离n-step TD方向，也就是计算回报的步数越来越多了。

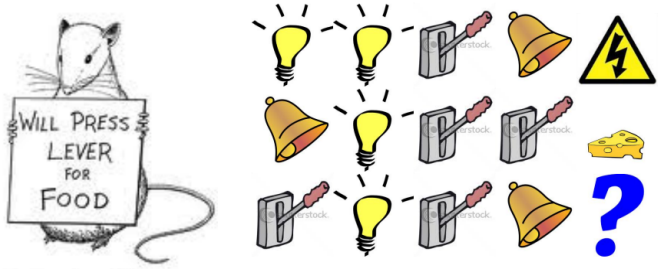


3. TD（λ）反向视角 （Backward view）

前向视角提供了理论基础，反向视角提供了机制。反向视角在线更新每一步，且从不完整的序列中更新。

资格迹（eligibility traces）

示例——被电击的原因



这是第一讲的一个例子，老鼠在连续接受了3次响铃和1次亮灯信号后遭到了电击，那么在分析遭电击的原因时，到底是响铃的因素较重要还是亮灯的因素更重要呢?

两个概念：
频率启发（Frequency heuristic）：将原因归因于出现频率最高的状态

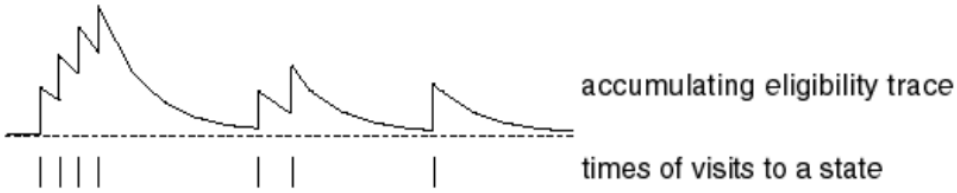
就近启发（Recency heuristic）：将原因归因于较近的几次状态

如何把上面两个概念结合起来呢？直观上理解，通过同时捕获频率启发（状态被访问越多，信用积分越多）和就近启发（状态访问时间越近，信用积分就越多）来处理信用积分分配问题（credit assignment）。 资格迹（Eligibility Traces）就是结合上述两个启发，给每一个状态在时间t引入一个数值，因此是关于状态的函数，定义：

$$E_0(s) = 0$$
$$E_t(s) = \gamma\lambda E_{t-1}(s) + 1(S_t = s)$$

式中 $1(S_t = s)$ 是一个条件判断表达式。

下图给出了 $E_t(s)$ 对于访问状态s关于时间t的累积资格迹：



该图横坐标是时间，横坐标下有竖线的位置代表当前进入了状态s，纵坐标是资格迹E。可以看出当某一状态连续出现，E值会在一定衰减的基础上有一个单位数值的提高，此时将增加该状态对于最终收获贡献的权重，因而在更新该状态价值的时候可以较多地考虑最终收获的影响。同时如果该状态距离最终状态较远，则其对最终收获的贡献越小，在更新该状态时也不需要太多的考虑最终收获。

资格迹的另一种等价函数描述：

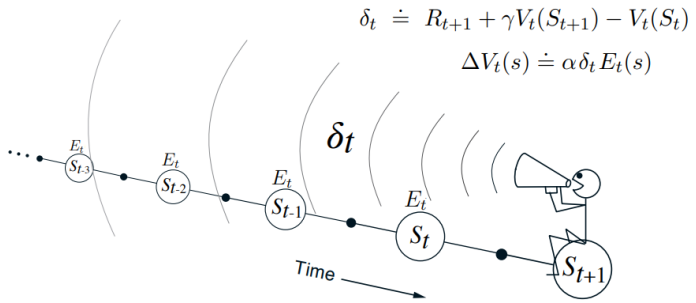
$$E_t(s) = \begin{cases} \gamma\lambda E_{t-1}(s) & if\ s \neq s_t \\ \gamma\lambda E_{t-1}(s) + 1 & if\ s = s_t \end{cases}$$

反向视角TD((λ)

E值并不需要等到完整的回合结束才能计算出来，它可以每经过一个时刻就得到更新。每一个状态s都有一个资格迹E，对于每个状态s，更新价值函数V(s)。价值的更新正比于TD-error δ_t 和资格迹 $E_t(s)$,如下：

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$
$$V(S) \leftarrow V(S) + \alpha\delta_t E_t(s)$$

反向视角TD((λ)的图解如下：



- 想象我们沿着时间计算TD误差，然后沿着时间反向，朝前面访问过的状态喊 δ_t
- 声音的强度随着时间距离以 $\gamma\lambda$ 递减

4. TD（λ）前向视角与反向视角的关系

为了更好的理解反向TD(λ)，来看看在不同的λ时会发生什么?

TD（λ）与TD（0）

- 当λ=0时，只有当前状态得到更新，等同于TD(0)算法

$$E_t(s) = 1(S_t = s)$$
$$V(S) \leftarrow V(S) + \alpha\delta_t E_t(s)$$

上式中资格迹等于1，就是TD(0)的更新。

TD（λ）与MC

- 当 λ=1时，信用积分推迟到回合的结尾；考虑回合性环境中的离线更新；在回合的过程中，TD(1)的总更新与MC的总更新相同。

定理：离线更新之和与前向及后向TD（λ）相同，即

$$\sum_{t=1}^{T} \alpha\delta_t E_t(s) = \sum_{t=1}^{T} \alpha(G_t^\lambda - V(S_t))\ 1(S_t = s)$$

5. 前向与反向等价

MC 与TD(1)

考虑一个回合中，状态s在时间步k被访问了一次，TD(1)的资格迹从被访问时间步之后随时间衰减，

$$E_t(s) = \gamma E_{t-1}(s) + 1(S_t = s)$$
$$= \begin{cases} 0 & if\ t < k \\ \gamma^{t-k} & if\ t \geq k \end{cases}$$

TD(1)在线更新累积误差，

$$\sum_{t=1}^{T-1}\alpha\delta_t E_t(s)=\alpha\sum_{t=1}^{T-1}\gamma^{t-k}\delta_t=\alpha(G_k-V(S_k))$$

在回合的的末尾累积的总误差为

$$\delta_k+\gamma\delta_{k+1}+\gamma^2\delta_{k+2}+\cdots+\gamma^{T-1-k}\delta_{T-1}$$

TD(1)的缩放

当 λ=1时，TD误差之和放缩成MC的误差，

$$\begin{aligned}
&\delta_t+\gamma\delta_{t+1}+\gamma^2\delta_{t+2}+\cdots+\gamma^{T-1-t}\delta_{T-1}\\
&=R_{t+1}+\gamma V(S_{t+1})-V(S_t)\\
&+\gamma R_{t+2}+\gamma^2V(S_{t+2})-\gamma V(S_{t+1})\\
&+\gamma^2R_{t+3}+\gamma^3V(S_{t+3})-\gamma^2V(S_{t+2})\\
&\vdots\\
&+\gamma^{T-1-t}R_T+\gamma^{T-t}V(S_T)-\gamma^{T-1-t}V(S_{T-1})\\
&=R_{t+1}+\gamma R_{t+2}+\gamma^2R_{t+3}+\ldots+\gamma^{T-1-t}R_T-V(S_t)\quad=G_t-V(S_t)
\end{aligned}$$

TD（λ）与TD（1）

- TD（1）粗略等价于每次访问的MC算法；在线更新时，误差每一步都会有积累；如果价值函数只在回合的末尾离线更新时，TD(1)等同于MC算法。

TD（λ）的放缩

对于一般的λ，TD误差也可放缩到，λ-error， $G_t^\lambda-V(S_t)$

$$\begin{aligned}
G_t^\lambda-V(S_t)=&-V(S_t)+(1-\lambda)\lambda^0(R_{t+1}+\gamma V(S_{t+1}))\\
&+(1-\lambda)\lambda^1(R_{t+1}+\gamma R_{t+2}+\gamma^2V(S_{t+2}))\\
&+(1-\lambda)\lambda^2(R_{t+1}+\gamma R_{t+2}+\gamma^2R_{t+3}+\gamma^3V(S_{t+3}))\\
&+\ldots\\
\\
=&-V(S_t)+(\gamma\lambda)^0(R_{t+1}+\gamma V(S_{t+1})-\gamma\lambda V(S_{t+1}))\\
&+(\gamma\lambda)^1(R_{t+2}+\gamma V(S_{t+2})-\gamma\lambda V(S_{t+2}))\\
&+(\gamma\lambda)^2(R_{t+3}+\gamma V(S_{t+3})-\gamma\lambda V(S_{t+3}))\\
&+\ldots
\end{aligned}$$

$$
\begin{aligned}
= \ & (\gamma\lambda)^0(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)) \\
& + (\gamma\lambda)^1(R_{t+2} + \gamma V(S_{t+2}) - V(S_{t+1})) \\
& + (\gamma\lambda)^2(R_{t+3} + \gamma V(S_{t+3}) - V(S_{t+2})) \\
& + \dots \\
\\
& = \delta_t + \gamma\lambda\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \dots
\end{aligned}
$$

前向与反向 TD（λ）

考虑一个回合中，状态s在时间步k被访问了一次，TD(λ)的资格迹从被访问时间步之后随时间衰减，

$$
\begin{aligned}
E_t(s) &= \lambda\gamma E_{t-1}(s) + 1(S_t = s) \\
&= \begin{cases} 0 & if\ t < k \\ (\lambda\gamma)^{t-k} & if\ t \geq k \end{cases}
\end{aligned}
$$

反向TD（λ）在线更新累积误差

$$
\sum_{t=1}^{T} \alpha\delta_t E_t(s) = \alpha \sum_{t=k}^{T} (\lambda\gamma)^{t-k}\delta_t = \alpha(G_k^\lambda - V(S_k))
$$

在回合的末尾，累积了λ-回报的总误差；对于多次访问状态s， $E_t(s)$累积了许多误差。

对于离线更新，更新的累积发生在回合内，但是在回合的末尾可以应用批量更新。
对于在线更新，TD(λ)更新被应用于回合内的每一个时间步；前向和反向TD(λ)有略微差别；Sutton（ICML 2014） 通过利用资格迹形式的略微差别，在线TD(λ)取得完美的等价。

前向与反向TD（λ）的总结

小结如下表所示，

Offline updates	$\lambda = 0$	$\lambda \in (0,1)$	$\lambda = 1$
Backward view	TD(0)	TD(λ)	TD(1)
	$\Vert$	$\Vert$	$\Vert$
Forward view	TD(0)	Forward TD(λ)	MC
Online updates	$\lambda = 0$	$\lambda \in (0,1)$	$\lambda = 1$
Backward view	TD(0)	TD(λ)	TD(1)
	$\Vert$	$\nparallel$	$\nparallel$
Forward view	TD(0)	Forward TD(λ)	MC
	$\Vert$	$\Vert$	$\Vert$
Exact Online	TD(0)	Exact Online TD(λ)	Exact Online TD(1)

表中的垂直等号表示在回合的末尾总更新等价。

参考

1.David Silver第4课

2.Richard Sutton 《Reinforcement Learning A Introduction》

3.叶强《David Silver强化学习公开课中文讲解及实践》第四讲