动态规划、蒙特卡洛、时序差分

一. 动态规划（DP）：用动态规划求解

1. 动态规划和强化学习问题的联系

(1) 动态规划的两个关键点：

a. 问题的最优解可以由若干小问题的最优解构成，

即通过寻找子问题的最优解来得到问题的最优解；

b. 可以找到子问题状态之间的递推关系，

通过较小的子问题状态递推出较大的子问题的状态。

强化学习的问题恰好是满足这两个条件的。

(2) 强化学习的两个基本问题：

a. 预测：求解给定策略的状态价值函数的问题

给定：强化学习的 6 个要素（状态集 S，动作集 A，状态转移概率矩阵 P，即时奖励 R，衰减因子 γ，策略 π）

求解：给定策略 π 的状态价值函数 v(π)

b. 控制：求解最优的价值函数和策略

给定：强化学习的 5 个要素（状态集 S，动作集 A，状态转移概率矩阵 P，即时奖励 R，衰减因子 γ）

求解：最优的状态价值函数 v*和最优策略 π*

(3) 动态规划和强化学习的关系

马尔科夫决策过程(MDP)中状态价值函数的贝尔曼方程

$$v_\pi(s) = \sum_{a \in A} \pi(a|s)\left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')\right)$$

a. 定义出子问题求解每个状态的状态价值函数

b. 递推的式子（可以使用上一个迭代周期内的状态价值来计算更新当前迭代周期

某状态 s 的状态价值）

使用动态规划来求解强化学习问题是比较自然的。

2. 策略评估求解预测问题

用动态规划来求解强化学习的预测问题，即求解给定策略的状态价值函数的问题

该问题求解过程通常叫做策略评估(Policy Evaluation)

基本思路：从任意一个状态价值函数开始，依据给定的策略，结合贝尔曼期望方程、状态转移概率和奖励同步迭代更新状态价值函数，直至其收敛，得到该策略下最终的状态价值函数。

假设我们在第 k 轮迭代已经计算出了所有的状态的状态价值，那么在第 k+1 轮我们可以利用第 k 轮计算出的状态价值计算出第 k+1 轮的状态价值。这是通过贝尔曼方程来完成的，即：

$$v_{k+1}(s)=\sum_{a\in A}\pi(a|s)(R_s^a+\gamma\sum_{s'\in S}P_{ss'}^a v_k(s'))$$

和上一节的马尔科夫决策过程(MDP)中状态价值函数的贝尔曼方程的唯一区别：

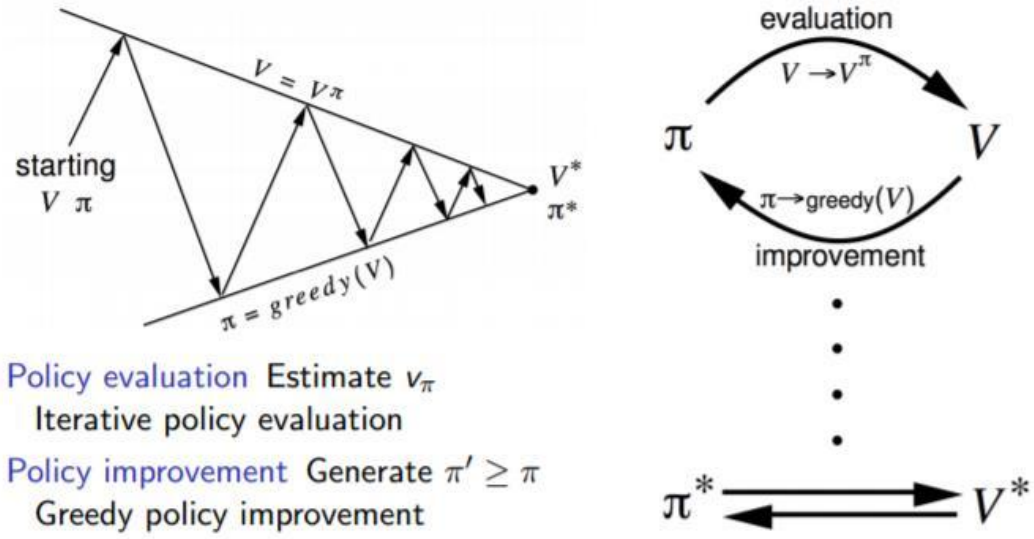
$$v_{\pi}(s)=\sum_{a\in A}\pi(a|s)(R_s^a+\gamma\sum_{s'\in S}P_{ss'}^a v_{\pi}(s'))$$

策略 π 已经给定，这里不再写出，对应加上了迭代轮数的下标。我们每一轮可以对计算得到的新的状态价值函数再次进行迭代，直至状态价值的值改变很小(收敛)，那么就得出了预测问题的解，即给定策略的状态价值函数 v(π)。

3. 策略迭代求解控制问题

用动态规划求解强化学习的控制问题，即根据之前基于任意一个给定策略评估得到的状态价值来及时调整动作策略，通常叫做策略迭代(Policy Iteration)。

基本思路：贪婪法。贪婪策略：个体在某个状态下选择的行为是其能够到达后续所有可能的状态中状态价值最大的那个状态。



在策略迭代过程中，我们循环进行两部分工作，第一步是使用当前策略 π*评估计算当前策略的最终状态价值 v*，第二步是根据状态价值 v*根据一定的方法（比如贪婪法）更新策略 π*，接着回到第一步，一直迭代下去，最终得到收敛的策略 π*和状态价值 v*。

4. 价值迭代求解控制问题

不是等到状态价值收敛才调整策略，而是随着状态价值的迭代及时调整策略，这样可以大大减少迭代次数。此时我们的状态价值的更新方法也和策略迭代不同。现在的贝尔曼方程迭代式子如下：

$$v_{k+1}(s) = \max_{a \in A}(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_k(s'))$$

可见由于策略调整，我们现在价值每次更新倾向于贪婪法选择的最优策略对应的后续状态价值，这样收敛更快。

5. 异步动态规划算法

在前几节我们讲的都是同步动态规划算法，即每轮迭代我会计算出所有的状态价值并保存起来，在下一轮中，我们使用这些保存起来的状态价值来计算新一轮的状态价值。

另一种动态规划求解是异步动态规划算法，在这些算法里，每一次迭代并不对所有状态的价值进行更新，而是依据一定的原则有选择性的更新部分状态的价值，这类算法有自己的一些独特优势，当然有额会有一些额外的代价。

常见的异步动态规划算法有三种：

第一种是原位动态规划 **(in-place dynamic programming)**：此时我们不会另外保存一份上一轮计算出的状态价值。而是即时计算即时更新。这样可以减少保存的状态价值的数量，节约内存。代价是收敛速度可能稍慢。

第二种是优先级动态规划 **(prioritised sweeping)**：该算法对每一个状态进行优先级分级，优先级越高的状态其状态价值优先得到更新。通常使用贝尔曼误差来评估状态的优先级，贝尔曼误差即新状态价值与前次计算得到的状态价值差的绝对值。这样可以加快收敛速度，代价是需要维护一个优先级队列。

第三种是实时动态规划 **(real-time dynamic programming)**：实时动态规划直接使用个体与环境交互产生的实际经历来更新状态价值，对于那些个体实际经历过的状态进行价值更新。这样个体经常访问过的状态将得到较高频次的价值更新，而与个体关系不密切、个体较少访问到的状态其价值得到更新的机会就较少。收敛速度可能稍慢。

6. 动态规划求解强化学习问题小结

动态规划是我们讲到的第一个系统求解强化学习预测和控制问题的方法。它的算法思路比较简单，主要就是利用贝尔曼方程来迭代更新状态价值，用贪婪法之类的方法迭代更新最优策略。

动态规划算法使用全宽度（**full-width**）的回溯机制来进行状态价值的更新，也就是说，无论是同步还是异步动态规划，在每一次回溯更新某一个状态的价值时，都要回溯到该状态的所有可能的后续状态，并利用贝尔曼方程更新该状态的价值。

这种全宽度的价值更新方式：

优点：对于状态数较少的强化学习问题还是比较有效的

缺点：当问题规模很大的时候，动态规划算法将会因贝尔曼维度灾难而无法使用。因此我们还需要寻找其他的针对复杂问题的强化学习问题求解方法。

二. 蒙特卡罗（MC）：用蒙特卡罗求解

问题：

(1) 需要在每一次回溯更新某一个状态的价值时，回溯到该状态的所有可能的后续状态。导致对于复杂问题计算量很大。

(2) 环境的状态转移模型 **P** 都无法知道，这时动态规划法根本没法使用。这时候蒙

特卡罗(Monte-Calo, MC)就是一种可行的方法。

1. 不基于模型的强化学习问题定义

基于模型的强化学习问题：模型状态转移概率矩阵 P 始终已知（即 MDP 已知）的强化学习问题

不基于模型的强化学习问题：模型状态转化概率矩阵 P 未知的强化学习问题

强化学习的两个基本问题：

预测问题：给定强化学习的 5 个要素：状态集 S，动作集 A，即时奖励 R，衰减因子 γ，给定策略 π， 求解该策略的状态价值函数 v(π)

控制问题：求解最优的价值函数和策略。给定强化学习的 5 个要素：状态集 S，动作集 A，即时奖励 R，衰减因子 γ，探索率 ϵ，求解最优的动作价值函数 q∗和最优策略 π∗

本文要讨论的蒙特卡罗法就是上述不基于模型的强化学习问题。

2. 蒙特卡罗法求解特点

一种通过采样近似求解问题的方法

蒙特卡罗法通过采样若干经历完整的状态序列(episode)来估计状态的真实价值。所谓的经历完整，就是这个序列必须是达到终点的。比如下棋问题分出输赢，驾车问题成功到达终点或者失败。有了很多组这样经历完整的状态序列，我们就可以来近似的估计状态价值，进而求解预测和控制问题了。

蒙特卡罗法和动态规划的不同点：

a. 它不需要依赖于模型状态转移概率 P；

b. 它从经历过的完整序列学习，完整的经历越多，学习效果越好。

3. 蒙特卡罗法求解强化学习预测问题

这里我们先来讨论蒙特卡罗法求解强化学习预测问题的方法，即策略评估。一个给定策略 π 的完整有 T 个状态的状态序列如下：

$$S_1, A_1, R_2, S_2, A_2, \ldots S_t, A_t, R_{t+1}, \ldots R_T, S_T$$

价值函数 $v_\pi(s)$的定义

$$v_\pi(s) = \mathbb{E}_\pi(G_t|S_t = s) = \mathbb{E}_\pi(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots |S_t = s)$$

可以看出每个状态的价值函数等于所有该状态收获的期望，同时这个收获是通过后

续的奖励与对应的衰减乘积求和得到。那么对于蒙特卡罗法来说，如果要求某一个状态的状态价值，只需要求出所有的完整序列中该状态出现时候的收获再取平均值即可近似求解，也就是：

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

$$v_\pi(s) \approx average(G_t), s.t. S_t = s$$

可以看出，预测问题的求解思路还是很简单的。不过有几个点可以优化考虑。

第一个点是同样一个状态可能在一个完整的状态序列中重复出现，那么该状态的收获该如何计算？有两种解决方法。第一种是仅把状态序列中第一次出现该状态时的收获值纳入到收获平均值的计算中；另一种是针对一个状态序列中每次出现的该状态，都计算对应的收获值并纳入到收获平均值的计算中。两种方法对应的蒙特卡罗法分别称为：首次访问(first visit) 和每次访问(every visit) 蒙特卡罗法。第二种方法比第一种的计算量要大一些，但是在完整的经历样本序列少的场景下会比第一种方法适用。

第二个点是累进更新平均值（incremental mean)。在上面预测问题的求解公式里，我们有一个 average 的公式，意味着要保存所有该状态的收获值之和最后取平均。这样浪费了太多的存储空间。一个较好的方法是在迭代计算收获均值，即每次保存上一轮迭代得到的收获均值与次数，当计算得到当前轮的收获时，即可计算当前轮收获均值和次数。通过下面的公式就很容易理解这个过程：

$$\mu_k = \frac{1}{k}\sum_{j=1}^{k} x_j = \frac{1}{k}(x_k + \sum_{j=1}^{k-1} x_j) = \frac{1}{k}(x_k + (k-1)\mu_{k-1}) = \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})$$

这样上面的状态价值公式就可以改写成：

$$N(S_t) = N(S_t) + 1$$

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

这样我们无论数据量是多还是少，算法需要的内存基本是固定的 。

有时候，尤其是海量数据做分布式迭代的时候，我们可能无法准确计算当前的次数 $N(S_t)$,这时我们可以用一个系数 α 来代替，即：

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

对于动作价值函数 $Q(S_t,A_t)$,也是类似的，比如对上面最后一个式子，动作价值函数

版本为：

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t))$$

4. 蒙特卡罗法求解强化学习控制问题

蒙特卡罗法求解控制问题的思路和动态规划价值迭代的的思路类似。动态规划价值迭代的的思路，每轮迭代先做策略评估，计算出价值 $v_k(s)$，然后基于据一定的方法（比如贪婪法）更新当前策略 π。最后得到最优价值函数 $v*$和最优策略 $\pi*$。

和动态规划比，蒙特卡罗法不同之处体现在三点：

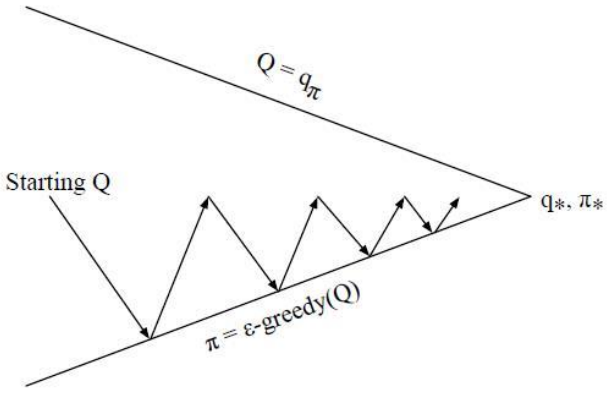
a. 预测问题策略评估的方法不同，这个第三节已经讲了。

b. 蒙特卡罗法一般是优化最优动作价值函数 $q*$，而不是状态价值函数 $v*$。

c. 动态规划一般基于贪婪法更新策略。而蒙特卡罗法一般采用 $\epsilon-$贪婪法更新。$\epsilon-$贪婪法通过设置一个较小的 ϵ 值，使用 $1-\epsilon$ 的概率贪婪地选择目前认为是最大行为价值的行为，而用 ϵ 的概率随机的从所有 m 个可选行为中选择行为。用公式可以表示为：

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & if\ a^* = \arg\max_{a \in A} Q(s,a) \\ \epsilon/m & else \end{cases}$$

在实际求解控制问题时，为了使算法可以收敛，一般 ϵ 会随着算法的迭代过程逐渐减小，并趋于 0。这样在迭代前期，我们鼓励探索，而在后期，由于我们有了足够的探索量，开始趋于保守，以贪婪为主，使算法可以稳定收敛。这样我们可以得到一张和动态规划类似的图：



5. 蒙特卡罗法控制问题算法流程

在这里总结下蒙特卡罗法求解强化学习控制问题的算法流程，这里的算法是在线

(on-policy)版本的,相对的算法还有离线(off-policy)版本的。在线和离线的区别我们在后续的文章里面会讲。同时这里我们用的是 every-visit,即个状态序列中每次出现的相同状态，都会计算对应的收获值。

在线蒙特卡罗法求解强化学习控制问题的算法流程如下:

输入：状态集 S，动作集 A，即时奖励 R，衰减因子 γ，探索率 ϵ

输出：最优的动作价值函数 q_*和最优策略 π_*

1. 初始化所有的动作价值 Q(s,a)=0， 状态次数 N(s,a)=0，采样次数 k=0，随机初始化一个策略 π

2. k=k+1，基于策略 π 进行第 k 次蒙特卡罗采样，得到一个完整的状态序列:

$$S_1, A_1, R_2, S_2, A_2, \ldots S_t, A_t, R_{t+1}, \ldots R_T, S_T$$

3. 对于该状态序列里出现的每一状态行为对(S_t,A_t)，计算其收获 G_t，更新其计数 N(s,a)和行为价值函数 Q(s,a):

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

$$N(S_t, A_t) = N(S_t, A_t) + 1$$

$$Q(S_t, A_t) = Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$$

4. 基于新计算出的动作价值，更新当前的 ϵ−贪婪策略:

$$\epsilon = \frac{1}{k}$$

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & if\ a^* = \arg\max_{a \in A} Q(s,a) \\ \epsilon/m & else \end{cases}$$

5. 如果所有的 Q(s,a)收敛，则对应的所有 Q(s,a)即为最优的动作价值函数 q_*。对应的策略 π(a|s)即为最优策略 π_*。否则转到第二步。

6. 蒙特卡罗法求解强化学习问题小结

蒙特卡罗法是我们第二个讲到的求解强化问题的方法，也是第一个不基于模型的强化问题求解方法。

优点：

a. 避免动态规划求解过于复杂；

b. 不事先知道环境转化模型；

因此可以用于海量数据和复杂模型。

问题：

每次采样都需要一个完整的状态序列。如果我们没有完整的状态序列，或者很难拿到较多的完整的状态序列，这时候蒙特卡罗法就不太好用了，也就是说，我们还需要寻找其他的更灵活的不基于模型的强化问题求解方法。

三. 时序差分（TD）：用时序差分求解

蒙特卡罗法来求解强化学习问题的方法：

优点：很灵活，不需要环境的状态转化概率模型

缺点：需要所有的采样序列都是经历完整的状态序列

不使用完整状态序列求解强化学习问题的方法：时序差分(Temporal-Difference, TD)。

1. 时序差分 TD 简介

时序差分法和蒙特卡罗法类似，都是不基于模型的强化学习问题求解方法。所以在上一篇定义的不基于模型的强化学习控制问题和预测问题的定义，在这里仍然适用。

预测问题：给定强化学习的 5 个要素：状态集 S, 动作集 A, 即时奖励 R，衰减因子 γ，给定策略 π， 求解该策略的状态价值函数 v(π)

控制问题：求解最优的价值函数和策略。给定强化学习的 5 个要素：状态集 S, 动作集 A, 即时奖励 R，衰减因子 γ，探索率 ϵ，求解最优的动作价值函数 q∗和最优策略 π∗

回顾蒙特卡罗法中计算状态收获的方法是：

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

对于时序差分法来说，我们没有完整的状态序列，只有部分的状态序列，那么如何可以近似求出某个状态的收获呢？

马尔科夫决策过程(MDP)中的贝尔曼方程：

$$v_\pi(s) = \mathbb{E}_\pi(R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s)$$

可以用 $R_{t+1}+\gamma v(S_{t+1})$来近似的代替收获 G_t，一般我们把 $R_{t+1}+\gamma v(S_{t+1})$称为 TD 目标值。$R_{t+1}+\gamma v(S_{t+1})-V(S_t)$称为 TD 误差,将用 TD 目标值近似代替收获 G(t)的过程称为引导(bootstrapping)。这样我们只需要两个连续的状态与对应的奖励，就可以尝试求解强化学习问题了。

现在我们有了自己的近似收获 Gt 的表达式，那么就可以去求解时序差分的预测问题和控制问题了。

2. 时序差分 TD 的预测问题求解

时序差分的预测问题求解和蒙特卡罗法类似，但是主要有两个不同点。

一是收获 G_t 的表达式不同，时序差分 G(t)的表达式为：

$$G(t) = R_{t+1} + \gamma V(S_{t+1})$$

二是迭代的式子系数稍有不同，回顾蒙特卡罗法的迭代式子是：

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

由于在时序差分我们没有完整的序列，也就没有对应的次数 $N(S_t)$,一般就用一个[0,1]的系数 α 代替。这样时序差分的价值函数迭代式子是：

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

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t))$$

蒙特卡罗法和时序差分法求解预测问题的区别：

一是时序差分法在知道结果之前就可以学习，也可以在没有结果时学习，还可以在持续进行的环境中学习，而蒙特卡罗法则要等到最后结果才能学习，时序差分法可以更快速灵活的更新状态的价值估计，这在某些情况下有着非常重要的实际意义。

二是时序差分法在更新状态价值时使用的是 TD 目标值，即基于即时奖励和下一状态的预估价值来替代当前状态在状态序列结束时可能得到的收获，是当前状态价值的有偏估计，而蒙特卡罗法则使用实际的收获来更新状态价值，是某一策略下状态价值的无偏估计，这一点蒙特卡罗法占优。

三是虽然时序差分法得到的价值是有偏估计，但是其方差却比蒙特卡罗法得到的方差要低，且对初始值敏感，通常比蒙特卡罗法更加高效。

从上面的描述可以看出时序差分法的优势比较大，因此现在主流的强化学习求解方法都是基于时序差分的。后面的文章也会主要基于时序差分法来扩展讨论。

3. n 步时序差分

在第二节的时序差分法中，我们使用了用 $R_{t+1}+\gamma v(S_{t+1})$来近似的代替收获 G_t。即向前一

步来近似我们的收获 G_t,那么能不能向前两步呢？当然可以，这时我们的收获 G_t 的近似表达式为：

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

从两步，到三步，再到 n 步，我们可以归纳出 n 步时序差分收获 $G_t^{(n)}$ 表达式为：

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

当 n 越来越大，趋于无穷，或者说趋于使用完整的状态序列时，n 步时序差分就等价于蒙特卡罗法了。

对于 n 步时序差分来说，和普通的时序差分的区别就在于收获的计算方式的差异。那么既然有这个 n 步的说法，那么 n 到底是多少步好呢？如何衡量 n 的好坏呢？我们在下一节讨论。

4. TD(λ)

n 步时序差分选择多少步数作为一个较优的计算参数是需要尝试的超参数调优问题。为了能在不增加计算复杂度的情况下综合考虑所有步数的预测，我们引入了一个新[0,1]的参数 λ，定义 λ–收获是 n 从 1 到∞所有步的收获乘以权重的和。每一步的权重是 $(1-\lambda)\lambda^{n-1}$，这样 λ–收获的计算公式表示为：

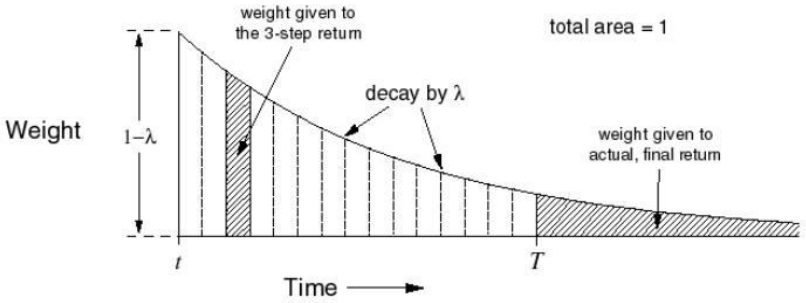
$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

进而我们可以得到 TD(λ)的价值函数的迭代公式：

$$V(S_t) = V(S_t) + \alpha(G_t^{\lambda} - V(S_t))$$

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

每一步收获的权重定义为 $(1-\lambda)\lambda^{n-1}$ 的原因是什么呢？其图像如下图所示，可以看到随着 n 的增大，其第 n 步收获的权重呈几何级数的衰减。当在 T 时刻到达终止状态时，未分配的权重全部给予终止状态的实际收获值。这样可以使一个完整的状态序列中所有的 n 步收获的权重加起来为 1，离当前状态越远的收获其权重越小。



从前向来看 TD(λ)， 一个状态的价值 $V(S_t)$由 G_t 得到，而 G_t 又间接由所有后续状态价值计算得到，因此可以认为更新一个状态的价值需要知道所有后续状态的价值。也就是说，必须要经历完整的状态序列获得包括终止状态的每一个状态的即时奖励才能更新当前状态的价值。这和蒙特卡罗法的要求一样，因此 TD(λ)有着和蒙特卡罗法一样的劣势。当 λ=0 时,就是第二节讲到的普通的时序差分法，当 λ=1 时,就是蒙特卡罗法。

从反向来看 TD(λ)，它可以分析我们状态对后续状态的影响。比如老鼠在依次连续接受了 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？如果把老鼠遭到电击的原因认为是之前接受了较多次数的响铃，则称这种归因为频率启发(frequency heuristic) 式；而把电击归因于最近少数几次状态的影响，则称为就近启发(recency heuristic) 式。

如果给每一个状态引入一个数值：效用(eligibility, E) 来表示该状态对后续状态的影响，就可以同时利用到上述两个启发。而所有状态的效用值总称为效用迹(eligibility traces,ES)。定义为：

$$E_0(s) = 0$$

$$E_t(s) = \gamma\lambda E_{t-1}(s) + 1(S_t = s) = \begin{cases} 0 & t < k \\ (\gamma\lambda)^{t-k} & t \geq k \end{cases}, \quad s.t.\ \lambda, \gamma \in [0,1], s\ is\ visited\ once\ at\ time\ k$$

此时我们 TD(λ)的价值函数更新式子可以表示为：

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

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

也许有人会问，这前向的式子和反向的式子看起来不同啊，是不是不同的逻辑呢？其实两者是等价的。现在我们从前向推导一下反向的更新式子。

$$
\begin{aligned}
G_t^\lambda - V(S_t) &= -V(S_t) + (1-\lambda)\lambda^0(R_{t+1} + \gamma V(S_{t+1})) && (1)\\
&+ (1-\lambda)\lambda^1(R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})) && (2)\\
&+ (1-\lambda)\lambda^2(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 V(S_{t+3})) && (3)\\
&+ \ldots && (4)\\
&= -V(S_t) + (\gamma\lambda)^0(R_{t+1} + \gamma V(S_{t+1}) - \gamma\lambda V(S_{t+1})) && (5)\\
&+ (\gamma\lambda)^1(R_{t+2} + \gamma V(S_{t+2}) - \gamma\lambda V(S_{t+2})) && (6)\\
&+ (\gamma\lambda)^2(R_{t+3} + \gamma V(S_{t+3}) - \gamma\lambda V(S_{t+3})) && (7)\\
&+ \ldots && (8)\\
&= (\gamma\lambda)^0(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)) && (9)\\
&+ (\gamma\lambda)^1(R_{t+2} + \gamma V(S_{t+2}) - V(S_{t+1})) && (10)\\
&+ (\gamma\lambda)^2(R_{t+3} + \gamma V(S_{t+3}) - V(S_{t+2})) && (11)\\
&+ \ldots && (12)\\
&= \delta_t + \gamma\lambda\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \ldots && (13)
\end{aligned}
$$

可以看出前向 TD 误差和反向的 TD 误差实际上一致的。

5. 时序差分的控制问题求解

现在我们回到普通的时序差分，来看看它控制问题的求解方法。回想上一篇蒙特卡罗法在线控制的方法，我们使用的是 ϵ−贪婪法来做价值迭代。对于时序差分，我们也可以用 ϵ−贪婪法来价值迭代，和蒙特卡罗法在线控制的区别主要只是在于收获的计算方式不同。时序差分的在线控制(on-policy)算法最常见的是 SARSA 算法。

而除了在线控制，我们还可以做离线控制(off-policy)，离线控制和在线控制的区别主要在于在线控制一般只有一个策略(最常见的是 ϵ−贪婪法)。而离线控制一般有两个策略，其中一个策略(最常见的是 ϵ−贪婪法)用于选择新的动作，另一个策略(最常见的是贪婪法)用于更新价值函数。时序差分的离线控制算法最常见的是 Q-Learning 算法，我们在下下篇单独讲解。

6. 时序差分小结

时序差分和蒙特卡罗法比它更加灵活，学习能力更强，因此是目前主流的强化学习求解问题的方法，现在绝大部分强化学习乃至深度强化学习的求解都是以时序差分的思想为基础的。