**Model-free Prediction**

本节课程讲解model-free prediction即所谓的无模型方法，包括：蒙特卡洛、TD、TD(λ)，所谓无模型应该是相对于动态规划而言，因为动规的基础是要至少转移概率，相当于对环境有充分的认识。无模型不是说不用建模，而是没有环境的模型。

# 蒙特卡洛学习

计算平均值的三种方法：

Method1：(2+4)/2=3

Method2: 2+(4-2)/2=3（左优先）

Method3: 4-(4-2)/2=3（右优先）

采用第二种方法给我们的工程实践带来方便。因为我们可以采用在线增量的方法来逐渐将value function收敛，这个地方可以联想下β与多项式分布的共轭关系使得β分布逐渐收敛是一个道理。

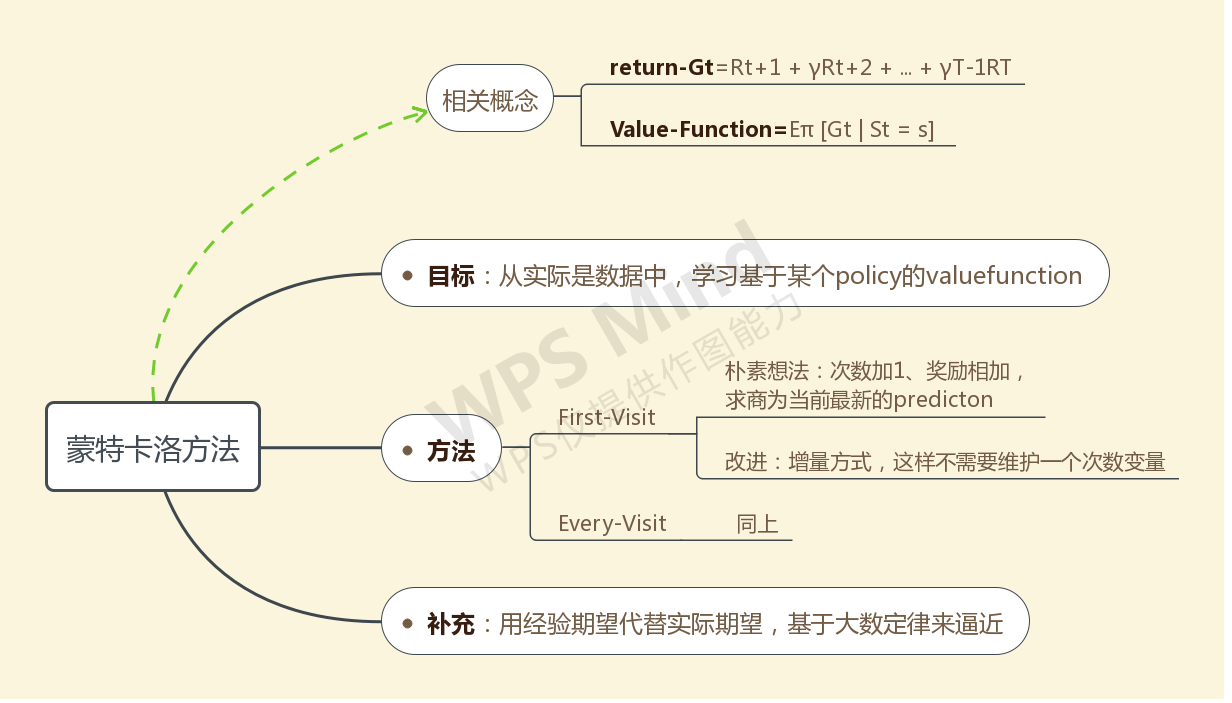
言归正传，蒙特卡洛学习的两种方法：区别在于对state的采样方法，不管怎样，评估一个方法要依据具体的环境，不能笼统的说哪一种好。

1、First-visit：在一个trajectory中，只get第一次出现某个state的位置，然后计算奖励值，更新value-function。

2、Every-visit：在一个trajectory中，考虑所有出现某个state的位置。

在一开始学习强化学习的时候，特别是有很多的例子是小游戏，我就有一个疑问：状态空间那么大，出现相同state的几率应该很小吧，这样模型训练起来应该很难，现在为止我对该问题的看法是：1、采用深度强化学习可以让模型具有泛化能力，在当前所有states训练的结果下，就算看到一个全新的state也可以做出不俗是预测。所以基础还是大量的训练数据，好在强化学习可以自己生成数据。所以感觉强化学习是一种性局部到整体的扩张式学习方法，扩张的边界就是环境能给出的所有state，而实际上可能并不需要触摸到边界就可以训练的不错，就像在一条马路上学骑车，我们并不需要压过每一寸土地。

再次言归正传，MC方法就是骑车直到某种终结状态（摔倒或者累了停下来）然后基于这些back up的奖励来训练value-function。



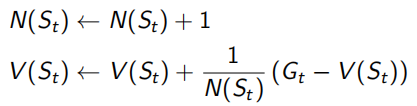
## 二．TD-learning

自举的思想-我不知道某个state的value，但是我知道两个state的value的差。就根据这个真实的差来训练网络。显然这样的训练有一个弊端就是你不知道自己估计得值是不是真值，例如将value function整体加1，那么这个训练依赖的差值是不变的。比如已知即时奖励为1，可以把网络训练成3和4，也可以是4和5吧,但是我目前觉得这没有关系，因为我们实际上想要的是一个策略，而这个策略是value之间的相对值，而不是绝对值，也就是别管我有高，只要比你高就行。

课上有学生问TD能否最终学到真值函数，David说能，但是我不明白为什么能，这也是一个问题，先放在这里。我现在觉得这句取决于终态的奖励设定。

**比较环节**

首先，对于蒙特卡洛，严格的更新公式如下：



Gt是什么？以first state为例，Gt就是一笔实验数据中对应的奖励求和。但是，对于某些动态问题这样可能不好，所以一个直觉的想法就是把全局期望迁移到近端的期望，例如维护一个固定大小的数据缓冲区，这样公式更新为：



α可以理解为缓冲区长度的倒数。

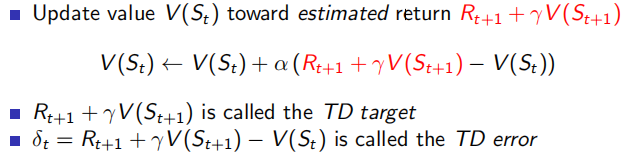
问题：error放缩后，会不会使得更新不准确？

前提条件是数据要足够多，如果只有很少的样本，显然公式是错误的。如果α等于N，那么求得就是严格意义上的平均值，如果α等于1，那么求得就相当于当前值作为平均值，所以公式是最新值与平均值的折中，如果设置合理，这样可以使得收敛速度加快。而且还有一个好处就是：开端不会震荡严重。

问题2:MC的要等到序列结束才能计算解决了吗？

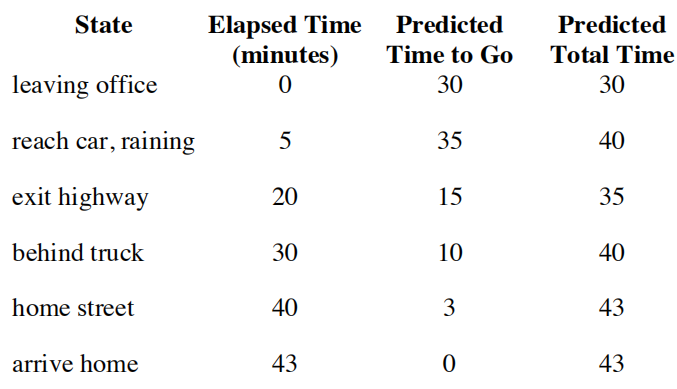
没有，必须等到序列结束才能得到Gt

接下来是TD方法：



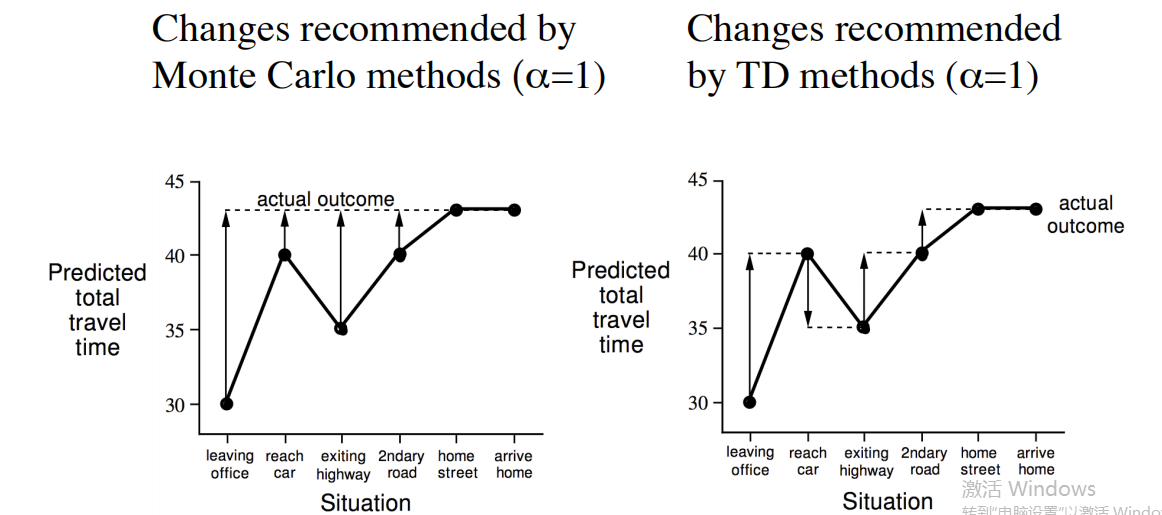
到此我们发现两者的结构具有某种相似性，这是david课程的特点就是很有体系，让人看到不同概念之间的联系。无论是MC还是TD都考虑了折扣引因子只不过MC把它写到了Gt中。

**一个实例-回家所用时间预测**



**Elapsed time**表示到达当前状态已经消耗的时间；

**Predict time**是预测的回到家还需要多长时间。



MC-方法：当α=1时，目标值就是当前值。所谓当前值就是trajectory结束后的实际值。即时奖励是否可以等到序列结束后再设定为实际值和预测值之差呢？

TD方法：当α=1时，目标值就是即时奖励（第一列）加上下一状态的耗时估计（第二列），等价于以第三列作为目标函数。

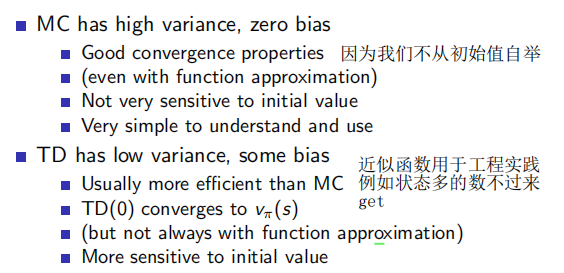
**方差-偏差trade-off**

先给出一个结论：MC高方差，低偏差，TD高偏差，低方差，这里的高和低是相对的概念。

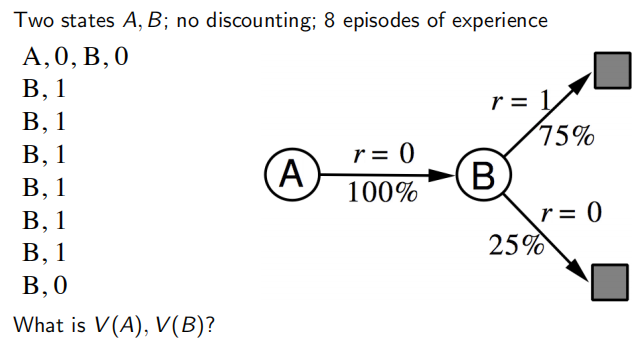
为什么呢？首先所谓的偏差指的是模型的拟合能力，而方差指的是模型泛化能力。对于MC而言，由于是看到整个的真实的奖励来更新，所以偏差较小，但是由于涉及到一些列action、rewards、transition所以鲁棒性不好也就是方差会比较大，相比较而言TD只看一步所以偏差大，方差小。

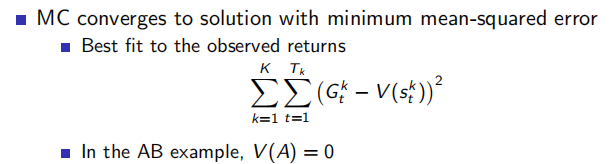
举个例子，**MC看完整战例**，理解的不错，现在假如战场环境和兵书上差不多那他会打的不错，但是如果战场环境变化较大，他可能就不知道怎么办了，相较而言**TD看的是战争片段，**那样他对战场环境的依赖就被比较小，即小方差。

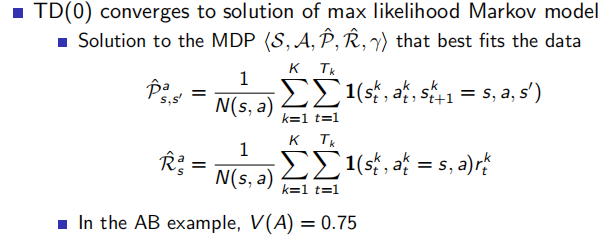
另外，MC对初始化不是很敏感，良好的收敛性能；TD更加高效、但是采用近似估计的时候不一定收敛。



**AB-example**







TD方法更适合MDP背景。

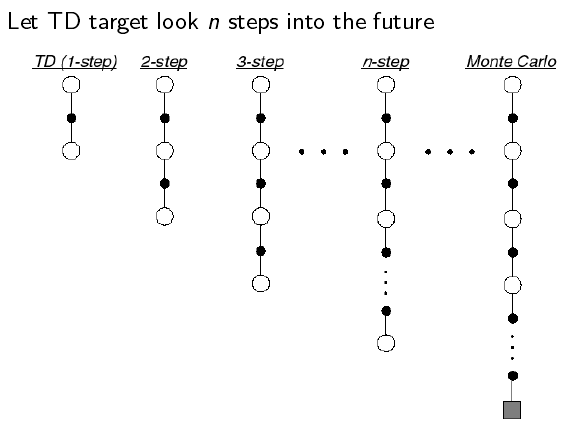
**对比总结：**

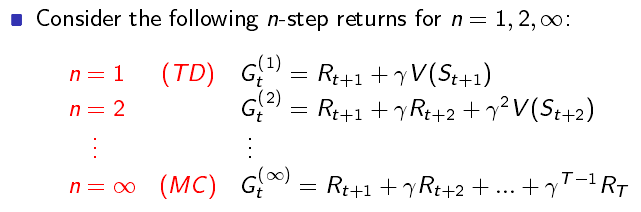
MC要在某一笔实验数据到达终结状态后学习，低偏差、高方差、更适合非马尔科夫环境。

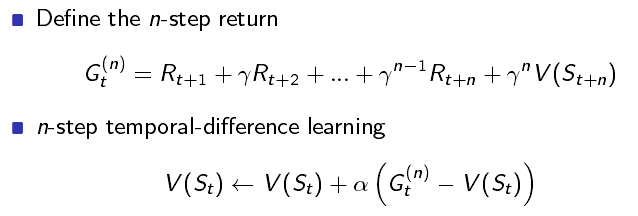
TD可以在两个state之间学习，低方差、高偏差、更适合马尔科夫环境。

## N-steps Prediction

因为TD方法具有高偏差的特点，所以就有了N-steps，考虑更长的真实奖励。无论n为多少都是对当前状态的价值评估。终端都是V(St)。

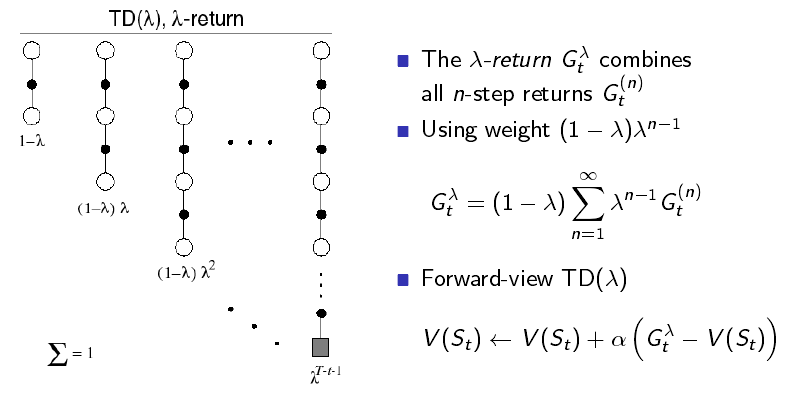


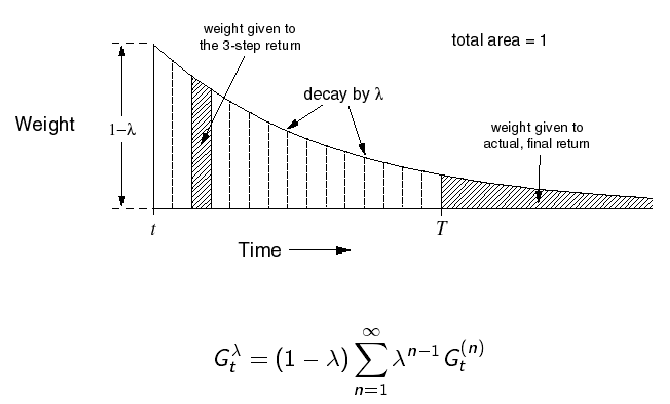




原始的Gt定义是从t时刻开始的奖励汇总，Gt(n) 是向后看了n步。实作时n就是缓冲区中一笔实验数据的长度。

N可以取很多的值，有一种方法可以把不同的n结合起来，就是Td(λ)。用G表示不同步长对应的价值函数，权重分配和更新机制如下：

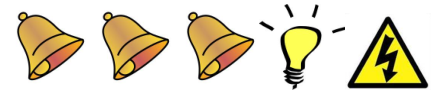




拿到实验数据，先算Gtn，再算Gtλ。

后面是讲正反向传播的，大概的意思是：从正向的角度考虑，需要考虑到episode的终点，但是从反向考虑可以此采用不完整的episode片段，原因是截止了Eligibility track技术，完整的思路没有理解，后面会来这里补充。

关于资格迹、以下面的例子说明。

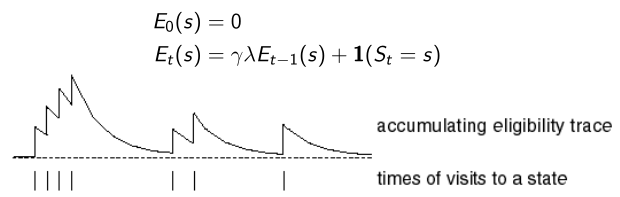


是什么引起了点击呢？

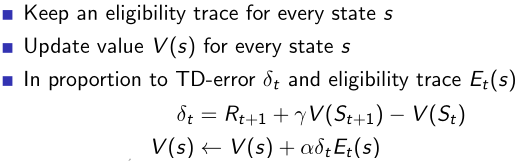
如果将信任分配到频率上，我们会认为是铃铛导致点击。

如果将信任分配到邻近的事情上，我们会认为灯光导致了电击。

资格迹综合考虑两种启发：



如果一个状态不发生：函数指数衰减。状态发生：直接增加1。



对每一个状态，维护一个资格迹，然后更新的幅度与之成正比。