1. 对于蒙特卡洛算法的理解

Monte Carlo method，也叫做统计模拟方法

理解为暴力枚举(通过伪随机数大量采样)足够多的情况，然后据此获得对应分布

作用很广泛，伪随机数，通过公式算出来

利用随机数种子，算出来很多随机数

如果用的相同随机数种子，随机数序列相同

步骤：

1. 构造或描述概率过程：首先要人为构造一个概率过程
2. 根据已知概率分布抽样，根据已知的概率分布抽样
3. 然后根据获得的样本进行统计分析，找到解的近似分布

个人理解：针对一个强化学习问题，希望通过策略迭代进行求解

假设由于对应策略的状态值或者动作值函数未知，此时复杂度十分高，难以通过方程求解

那么可以通过采样许多样本，就是在该策略下让 agent 不断实验

然后根据实验数据样本，我们可以估计状态值函数(动作值函数也可以被估计)

得到值函数后，根据对应值函数进行策略迭代，依次循环，获得最优策略

对比理解：动态规划算法与其相比

都是策略迭代

前者

1. 初始化策略与其对应的值函数
2. 策略评估，评估该策略的好坏，实际更新的是值函数



1. 根据更新的值函数，选择使自己奖励最大的动作，在某个状态下

此时就已经更新了策略

然后返回第二步，进行策略评估，再次更新，知道策略不再改变

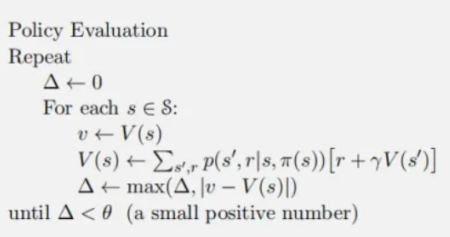
思考，其实初始化策略，伴随的值函数并非对应策略的值函数

根据策略评估，我们获得更有可能的对应策略的好坏

实际上策略评估是一个循环过程，我们要保证评估的状态值函数

足够接近真实的该策略下的状态值函数

因此有设置一个参数θ保证评估的是最优的，直到差别小于θ



很明显这种离散的策略评估的算法需要保证 状态空间是离散的(动作空间个人不确定)

离散的状态空间，才能根据以上的算法不断更新状态值函数，找到一个符合接近策略Π的

**总结：如果状态空间非离散，基于动态规划的策略迭代算法并不使用，原因为上**

1. 蒙特卡洛算法实例
2. 随即投点法计算Π值
3. 随机投点法计算定积分

3. 蒙特卡洛强化学习

1. 为什么要用蒙特卡洛算法(跟自己推理的不太一样，但认为课程的理解，区别更为深刻)

实际上动态规划是基于模型的方法，也就是动态规划算法已经提前知道了环境(环境的状态转移概率已知)

如果环境的状态转移概率已知，那么无论是策略迭代还是值迭代

更新状态值函数都会必然会使用 状态转移概率 来确定下一个可能的状态值

因此如果环境模型对于agent是未知的，也就是只能观察环境，对于如何变换，状态转移概率是位未知的

此时对于动态规划算法，无论是策略迭代，还是值迭代，都是无效的

解决方案1：通过机器学习算法来模拟环境知识，也就是模拟环境的变化，利用模型来拟合状态转移函数

然而这种模拟需要大量真实的数据

输入 动作与状态 输出 下一个可能的状态

这种方法需要大量真实的数据，对于小问题可能适用，但对复杂问题就十分困难了

这种方法还是基于模型的，因为最后你尝试建模，重现状态转移函数

解决方案2：免模型学习，其中蒙特卡洛算法就是其中之一

不需要直到环境的状态转移概率，通过大量采样本，模拟得出值迭代 的 状态转移概率

2. 基本思想

策略迭代框架

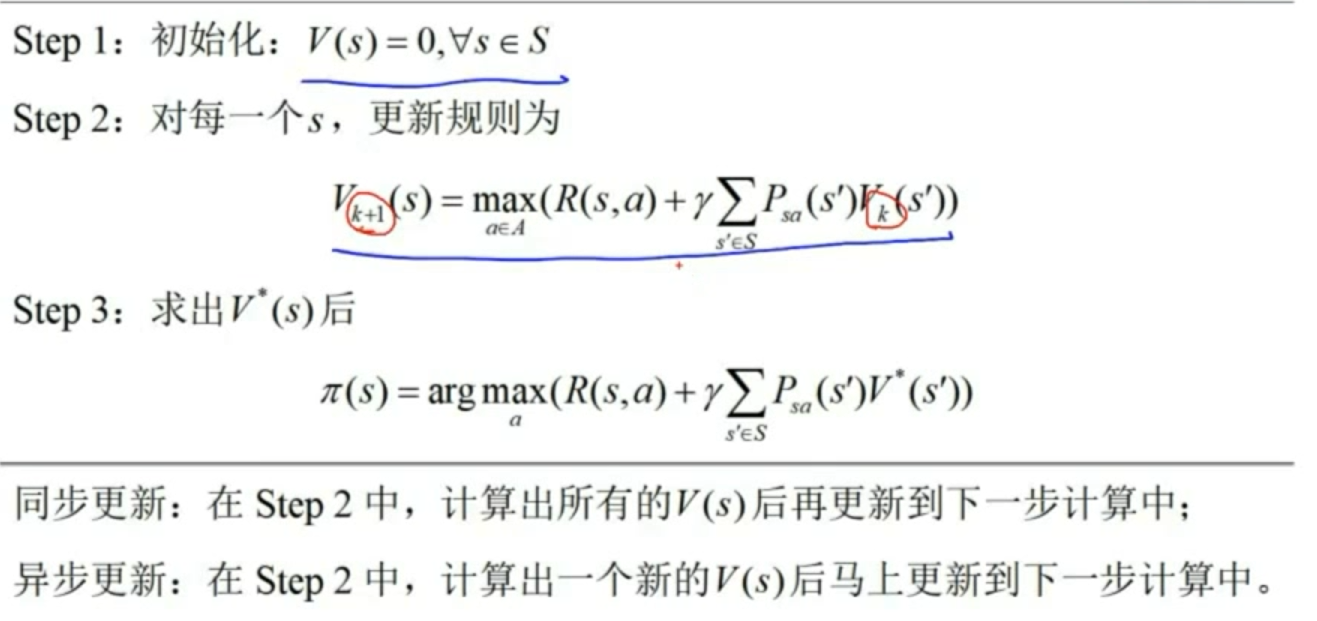
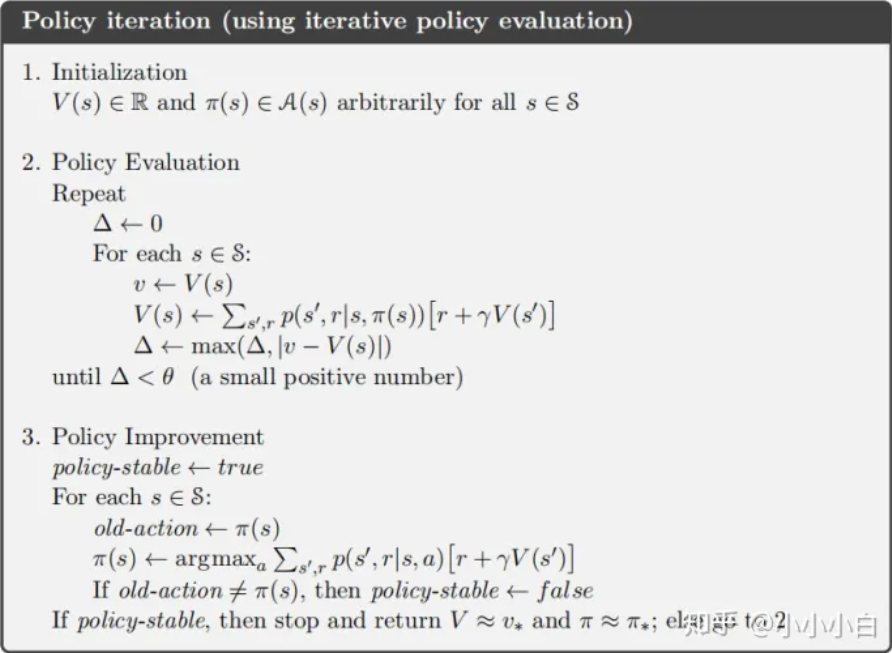
初始策略(包含策略与初始的值函数)->策略评估(不断更新值函数，直到稳定，即为该策略的评估)

->策略改进(通过评估策略的值函数来改进策略)

如果策略不再变化，或者不满足稳定标准；那么回到第二不进行策略评估

直到获取最优策略

但是由于现在状态转移概率是未知的，那么就无法进行策略评估，也无法进行策略改进



实际对于值迭代方法，如果没有状态转移概率，也是无法完成的

那么蒙特卡罗方法需要尝试解决策略迭代中的两个问题，策略评估与策略改进

1. 蒙特卡洛算法的策略评估
2. 经历完整的MDP序列，也就是序列必须能达到终止状态

个人理解：如果没有终止状态，agent一直保持在原位不动，这种策略评估也是无意义的

s0, a0, r1, s1, a1, r2, s2, a2, r3,....sT-1,aT-1, rT, sT

1. 大量抽样获取若干条经历完整的MDP 序列, 假设有m条

<1> s0, a0, r1, s1, a1, r2, s2, a2, r3,....sT-1,aT-1, rT, sT

<2> s0, a0, r1, s1, a1, r2, s2, a2, r3,....sT-1,aT-1, rT, sT

.....................

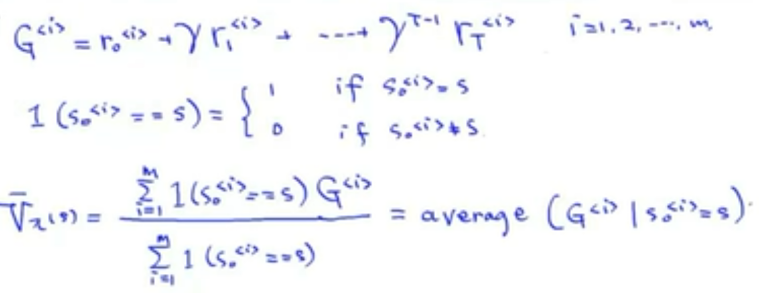
<m> s0, a0, r1, s1, a1, r2, s2, a2, r3,....sT-1,aT-1, rT, sT

注意序列长度并不一定相同，只是观察到了有m条抵达终点的MDP 序列

1. 求解出某个状态值的近似值，即某个状态的累折扣奖励的期望

那么我们可以通过多条G(st)从何获得，注意选定的s0必须与我们需要求解的si相同

也就是MDP的初始状态与我们需要评估的状态值是相同的

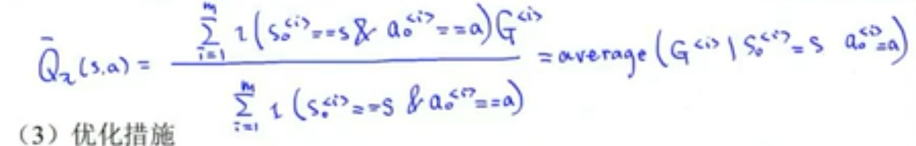


理解：会发现m条MDP序列初始状态并不一样

个人理解：假定初始状态都相同，那么如果对非初始状态的状态值的期望

个人认为可以从MDP序列中的某一点开始，即部分选取，仅仅个人想法

同样对与动作值函数的估计也是如此，并且策略改进过程不得不使用动作来优化



实际上一般都是只计算Q值，动作价值函数，这样能直接用在优化中

1. 优化措施
2. 同一个状态可能在一个完整的MDP序列中出现多次

首次访问：First Visit仅考虑MDP序列中出现在第一个时刻的状态

缺点：很明显很多状态很难出现在初始状态，并且需要大量采样，完整序列

优点：计算量小，针对小问题，随机初始化效果比较好

每次访问：Every Visit，此时针对MDP序列中每次出现的状态动作对，都纳入到期望计算中

优点：不需要大量采样，就能获得每个状态动作的期望

缺点：需要多次计算，计算量大

个人理解，可以边采样便更新，如果期望变化值很小，认为拟合比较接近->减小采样次数

1. 只保存当前均值与计算次数，也就是希望边抽样边计算

根据以往经验+课程个人理解

1. 状态值函数的均值计算公式

Vk+1(s) = Vk(s) + 1/N+1(Gk+1-Vk(s))

公式理解，对于多出来的部分，按照k+1份平均分给大家，这样每个人仍然是相等的

1. 动作值函数的均值计算公式

同理有

1. 策略改进

这里使用的是基于动作值函数的策略改进，策略评估也是动作值函数

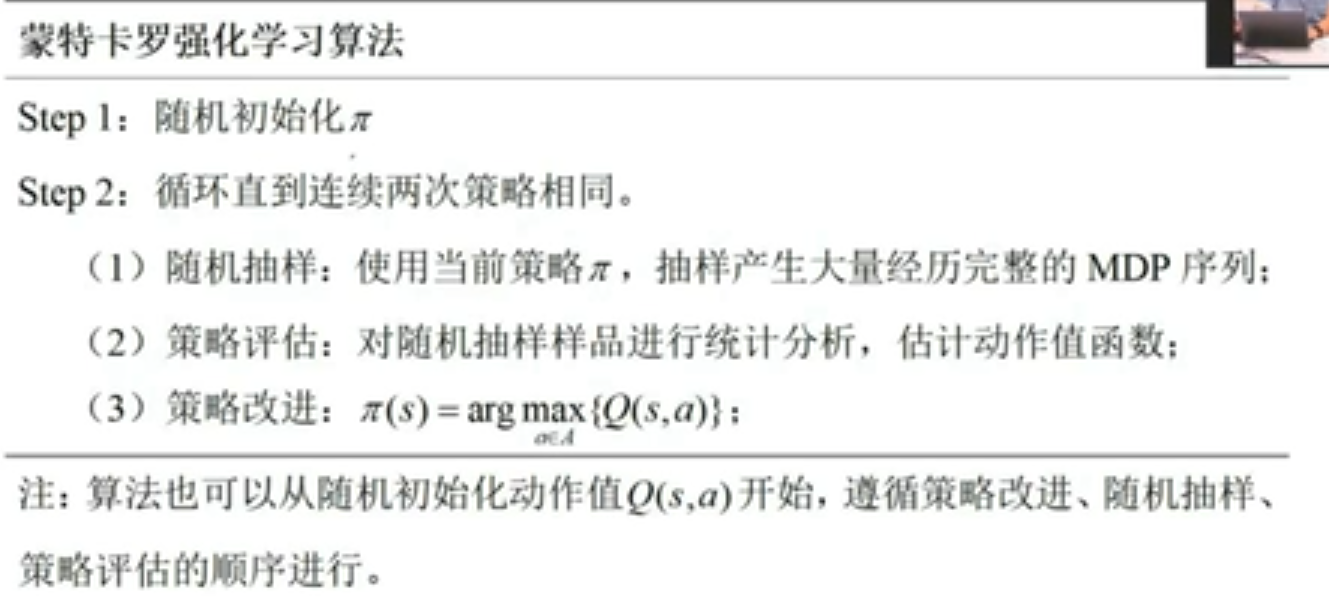
前者由于不知道状态转移概率只能使用

后者则是计算出来对于策略改进没有直接作用

因此基于蒙特卡洛算法的策略迭代，只能是采用动作值函数作为更新值函数

对于策略Π(a|s) argmax a∈A

对于所有动作，找到一个动作使得其值最大



蒙特卡洛算法的策略迭代新理解

1. 如果采用动作值函数作为初始化对象，然后依此来计算策略

那么省去了原本初始策略，然后求解值函数的过程

直接步入第二布，进行策略评估

如此理解，对于状态值函数的动态规划策略迭代，也是可以

不过只是省去了初步解决方程的麻烦

1. 问题：抽样应该从哪一个状态开始

如果初始状态是确定的，那么就从那两个状态开始

然而有些问题，可以从中间状态开始，依据问题情况来决定

如果总是从任意的起点开始，就是起点探索

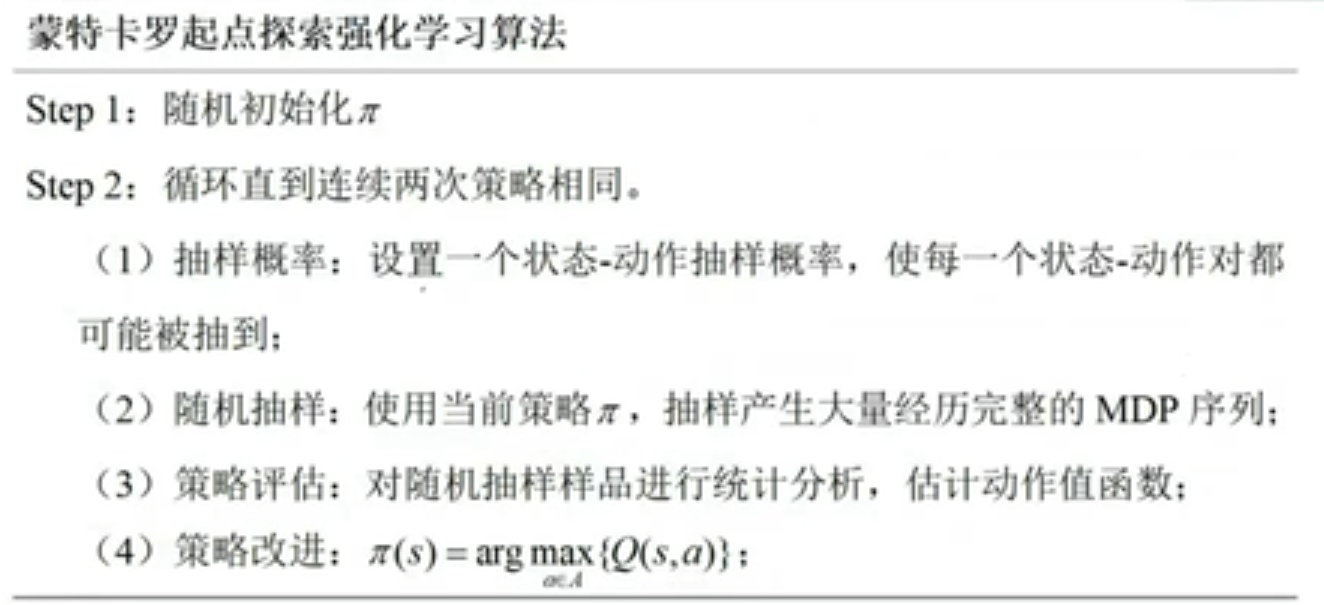
起点探索：任何一个状态动作对都可以作为抽样的起点。设置一个随机概率分布

使得所有可能的状态动作对都有不一定为0的概率作为起始状态

为什么要求起点探索

因为要评估Q(s,a)，解决Q(s,a) 在First Visit 中的稀疏性

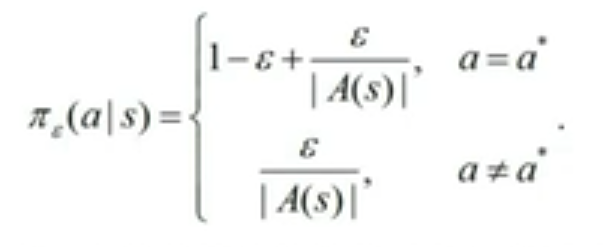
起点算法的算法流程



1. ε-贪心策略

Agent在与环境交互时，会有一定的概率选择例外的动作

以 1-ε概率选择最优动作，ε概率进行随机选择



理解：选择最优的概率还要增加ε/ |A(s)| 可以理解为 ε 随机概率被所有动作平分了

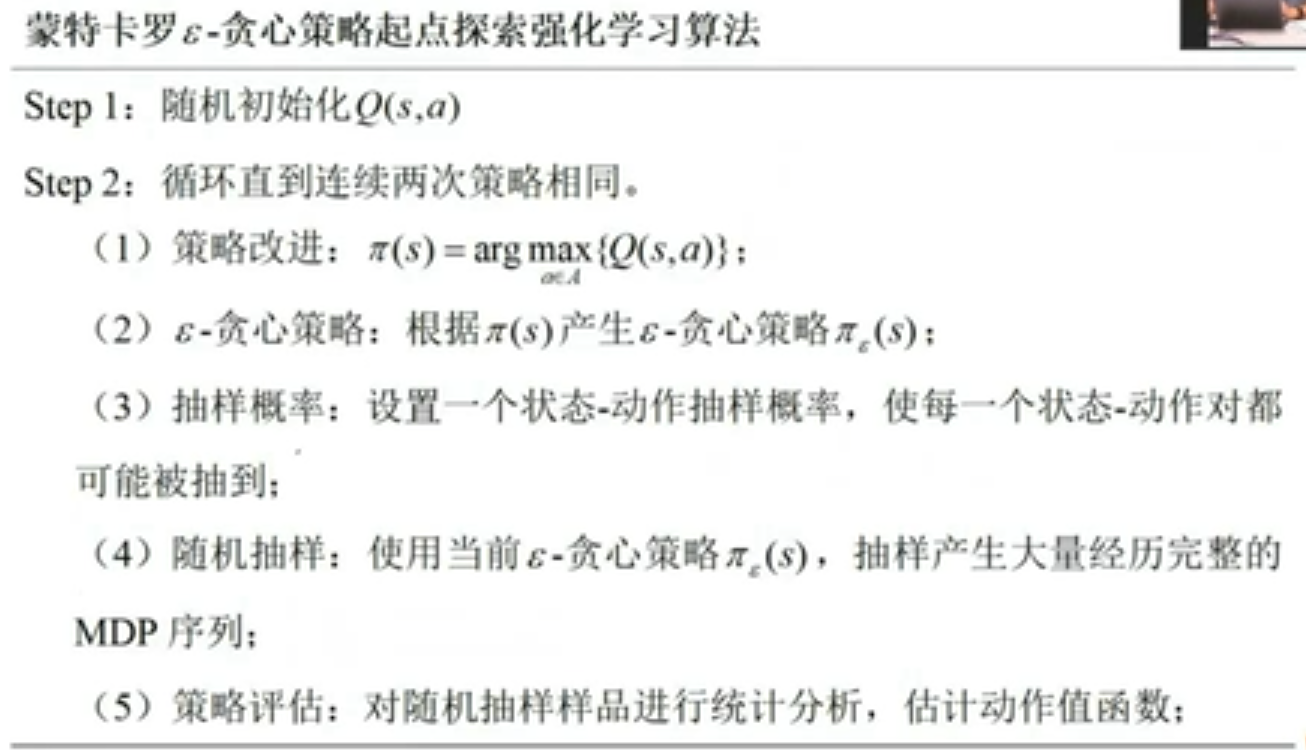
所有动作都可以得到 ε/ |A(s)| 概率

因此实际上选择最有动作概率为 1-ε+ ε/ |A(s)|

在实际算法中，ε的值一般随着算法迭代次数增加会逐渐减小，令ε= 1/k

表明只能在初期决策时，探索性更高

在中后期就变得更加保守



每次贪心获取序列，都是针对当前策略

实际上基于这种探索性的贪心算法能够 增加 收敛速度