环境已知 可以offline优化 dp

环境带有随机性 使用 model 提前计算 stochastic dp

环境完全未知 model free， episodic：让 Agent多次执行 标准：total或discounted runs长度 目标：别太笨的asap找到最优解 有exploration和exploitation之争：完全新路vs走当前最好 算法：启发式RL Q-learning 贝叶斯dp

核心概念：

Time：做 sequential决策时，当前决策有 direct和future影响，direct反映在reward，future则是未来的state

域T：无限域中的有限域

States：链接time periods的，包含预测未来所需的全部知识

Markov property：添加更多信息不增加reward，

对决策有暗示，有物理状态和用于学习的信息状态

域x

Reward：三种：有限奖励，long-run average，discounted 最大化reward 也是最小化 costs

1, 2, 4, 1, 2, 4, 1, 2, 4 例子：finite：加一块 long-run average：加一块除以步数（找规律）

Discounted：1 + 2β + 4β2 + β3 + ..... = ? for say β = 1/2 带有方系数相乘的累加

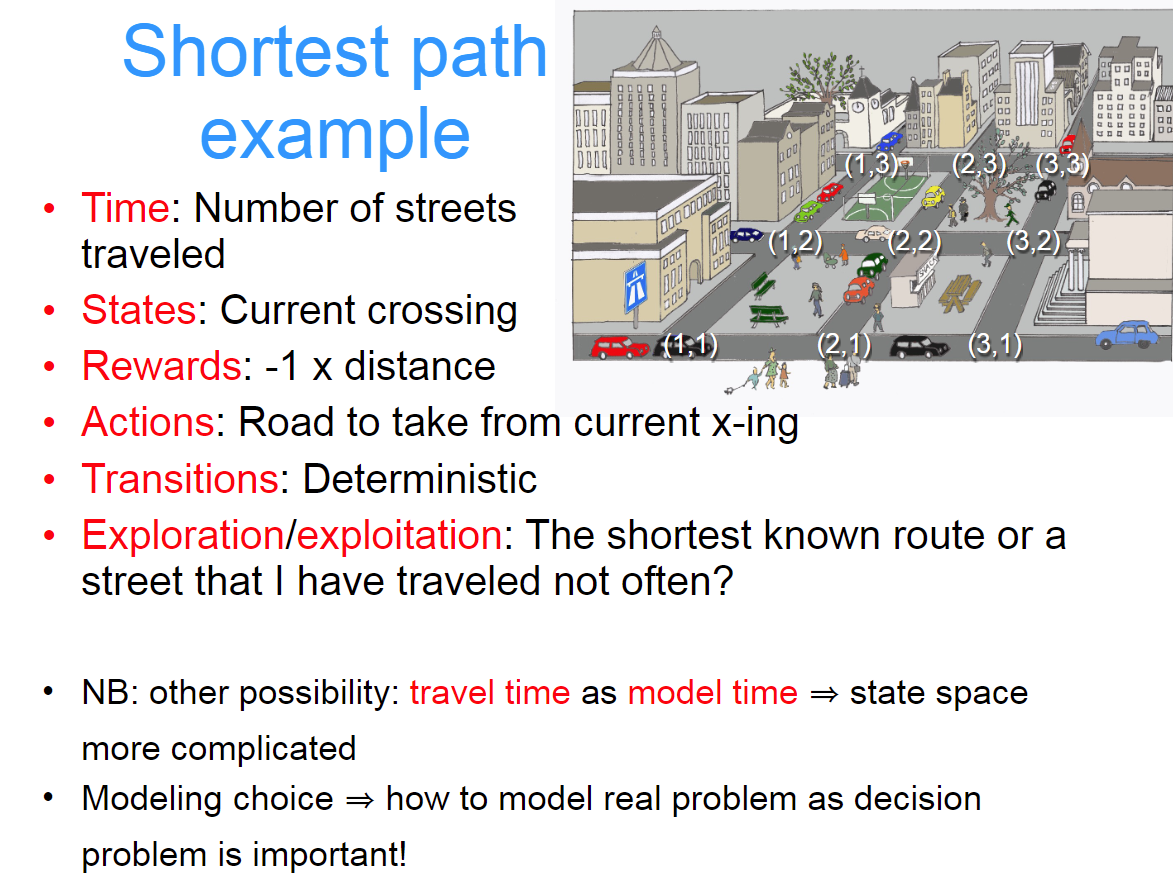
Policies（Strategies）：a：在每个time和每个state的一个action 属于A a:X x [1,T]属于A

Optimal的a应该是最大expected reward的，如不依赖time则为time齐次

Transition：state间的转化

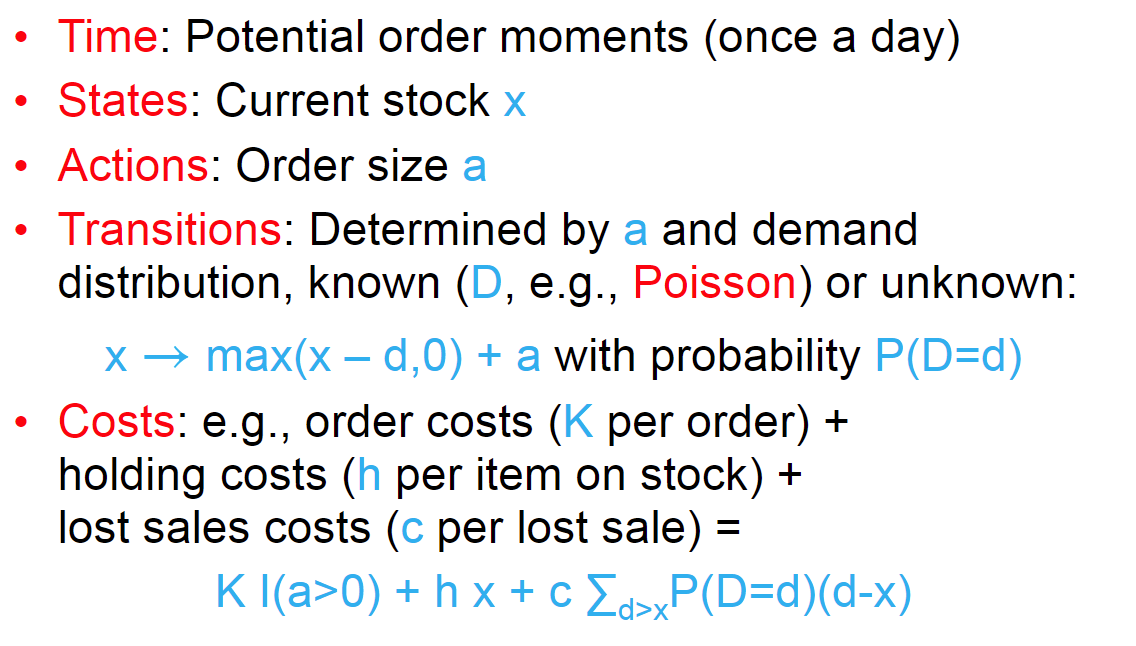
Deterministic：随时间的evolution完全可预测 Stochastic：下一状态是随机的，但已知distribution Learning：distribution还在学习，分为 implicit explicit

Exploration vs exploitation: 在learning中，exploitation是根据已知最大化reward，exploration是 做次优action来学习 信息状态制定trade off implicit 启发式RL是explicit



存货问题

顾客一个一个买存货，货物订单1天后抵达，存货不足则lost sales



维度爆炸问题 curse of dimensionality

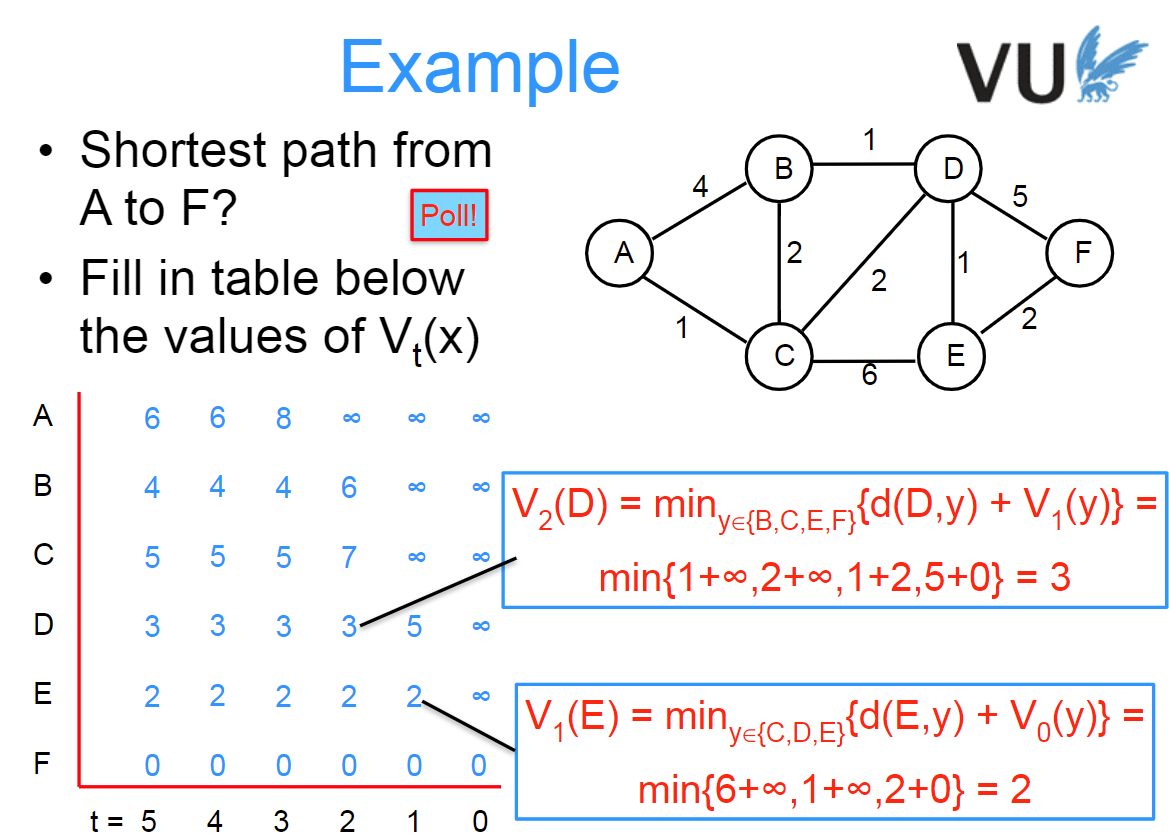
如果订单交付时间L>1天，则有 stock/order N 变为（N+1）^L问题变大

1950 Richard Bellman: “dynamic programming

1990 reinforcement learning”, heuristics for big problems

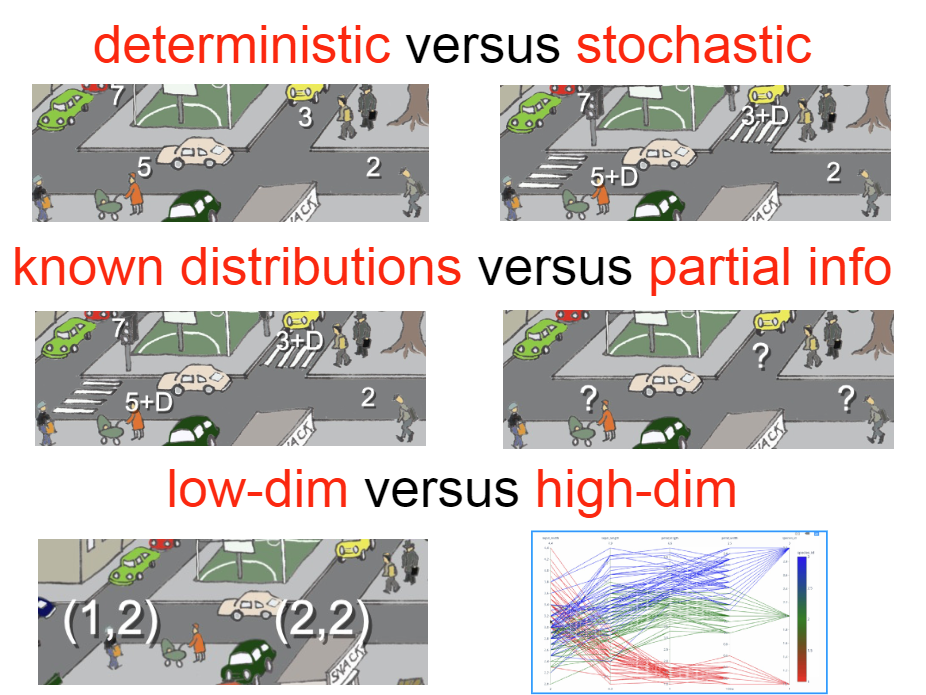
最短路径 反向算法

D(x,y)表示x当前位置 y上一状态位置，如无路径则infinite 最小化总路径



声明一个NxN矩阵，填满infinite，t0终点初始化为0, 向前更新t-1，遍历所有连接点，根据t时数据进行更新

Finite horizon DP

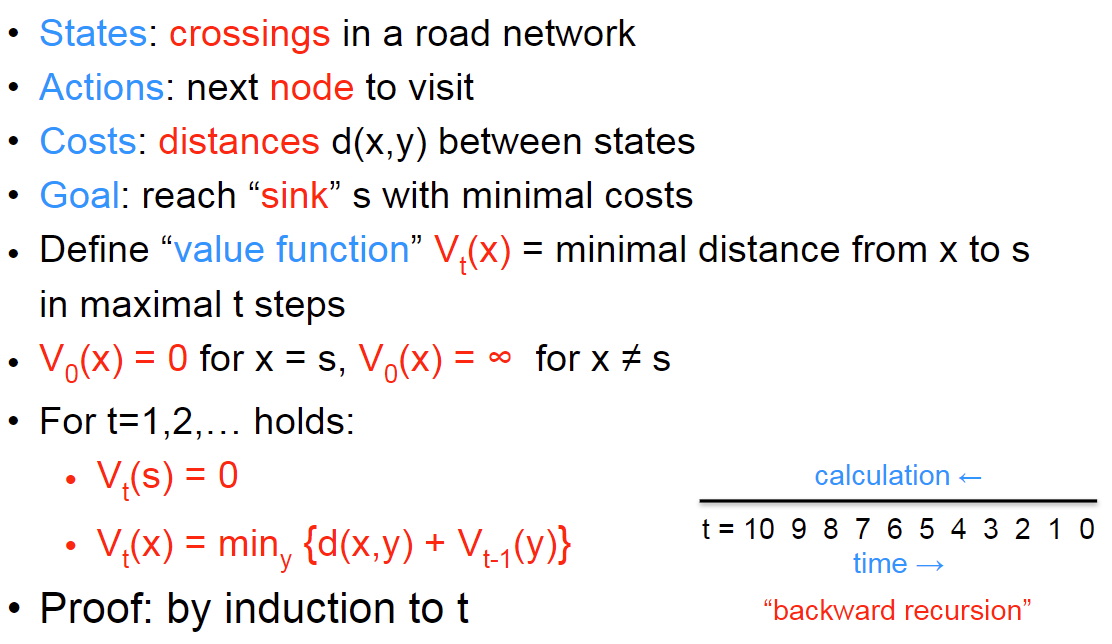
三种问题

低维 确定或随机 with 已知distribution 使用 DP 如 最短路径，库存管理

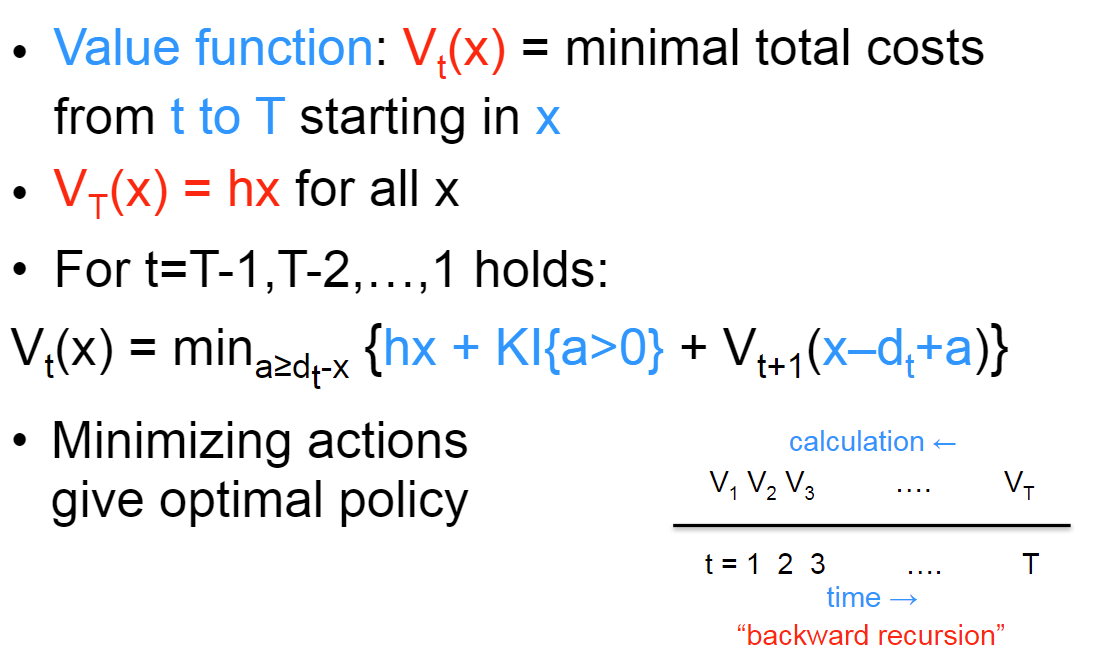
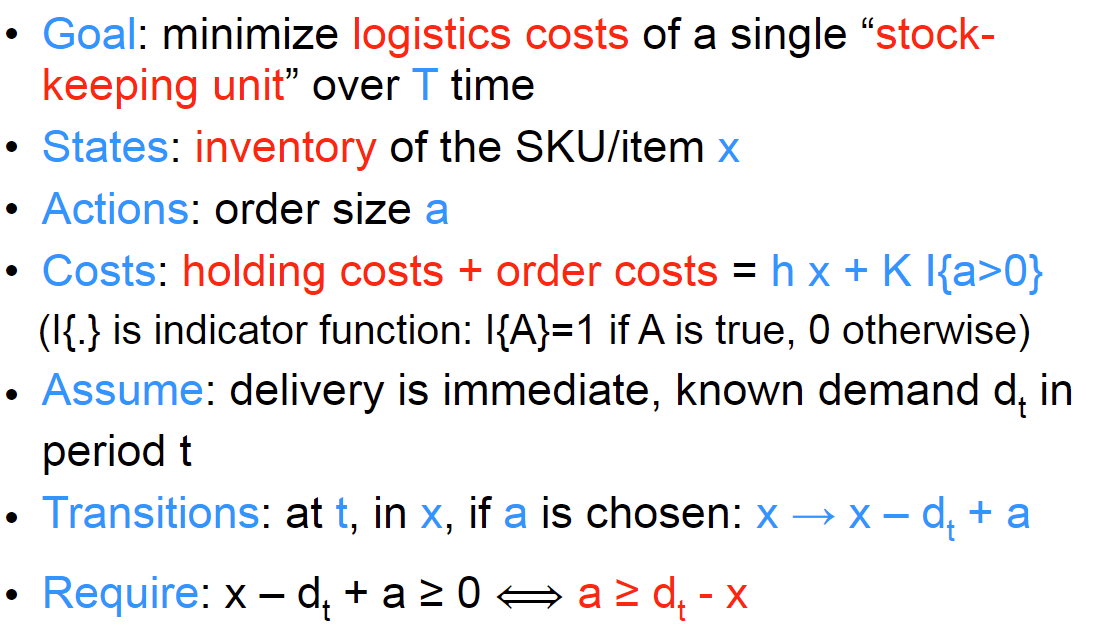
高维 确定或随机 with 已知distribution 使用 approximations近似 多item库存管理 chess

随即问题 with unknown distribution（partial info）使用 learning 如 shortest path **without prior info**, integrating learning about treatment effects with treatment decisions

最短路径例子

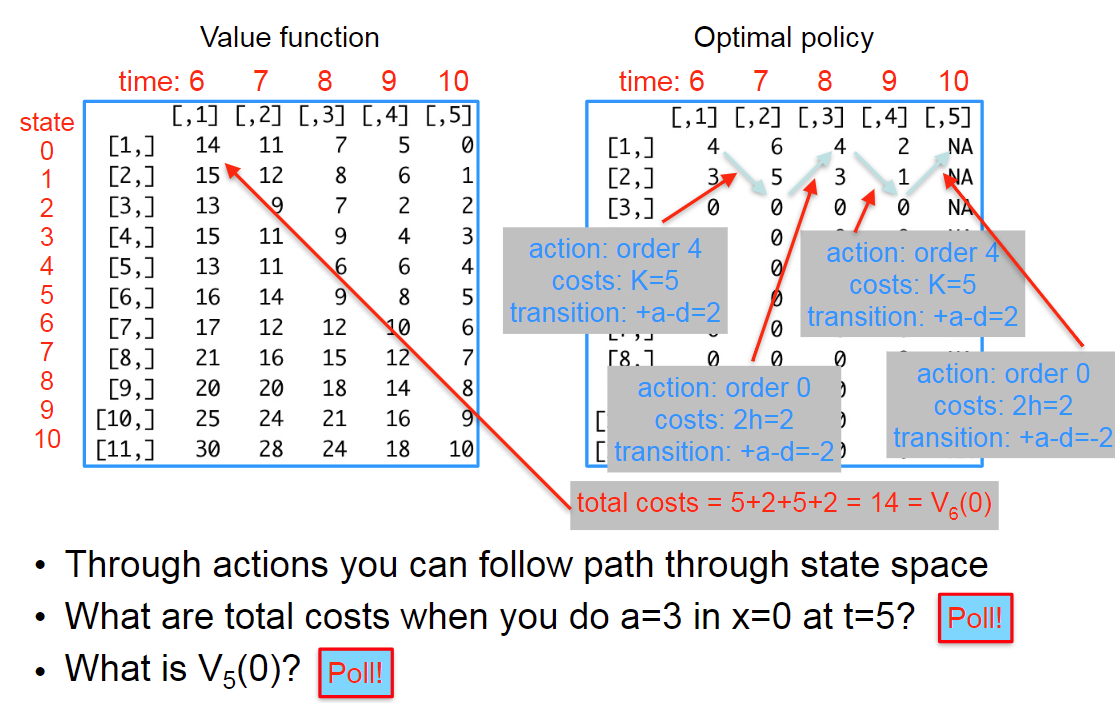


库存管理例子



存着货x有cost，下订单买货a也有cost，要这俩cost和最小从t到T （已有存货x）

每回合 交 2 个，订货不管定多少一次5，存货系数1\*n



问题2：在t为5:时，执行订3个货的行为（with 存货x为0），则为执行右边t=6时的需要3个货，订一次货+5，同位置左图值+5=20

问题3：在t为5时，无存货x=0的行为，则要t=6订货，右图t=6订0的同位置在左图为13+5

DP 优化原则：Bellman原则 t+1时刻的最优是t时刻最优的子集

What is optimal from t+1 on is part of the optimal policy from t on

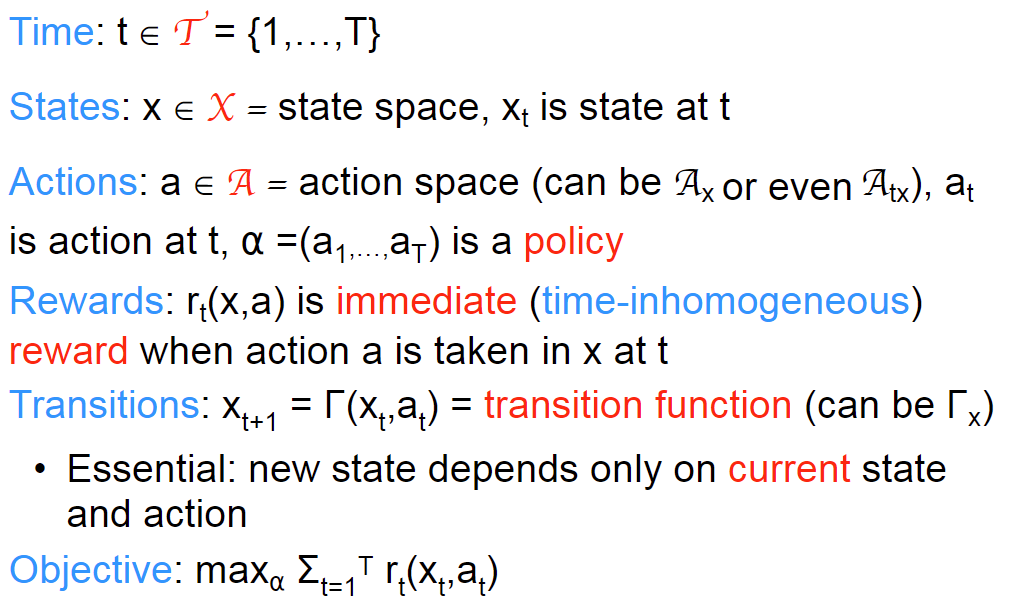
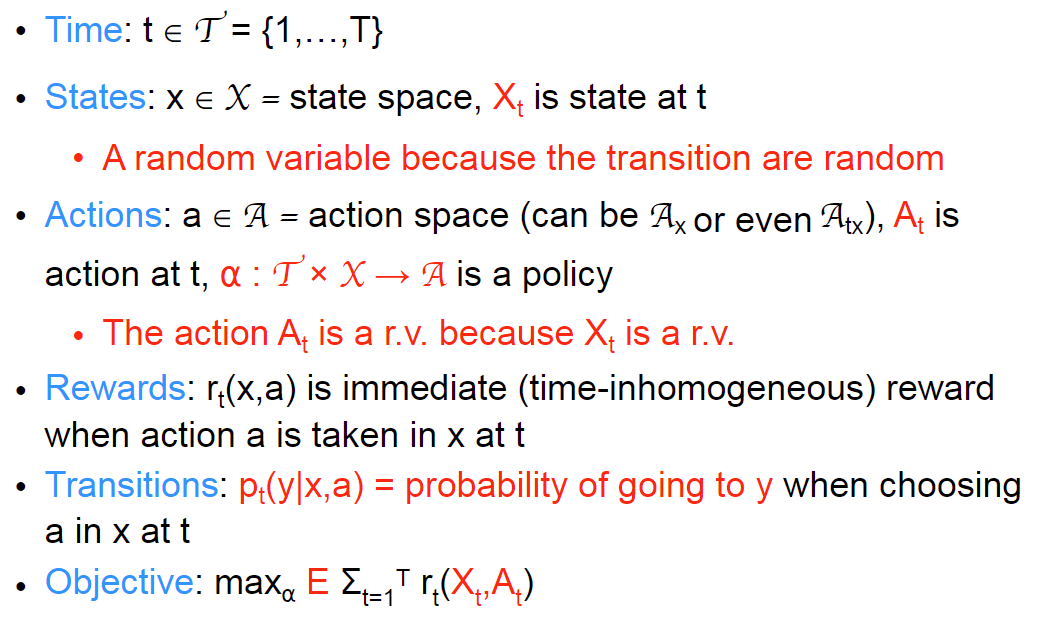
Vt(x) = optimal reward from t on starting in x 从t时刻开始，初始值为x的costs

Vt(x) = max a { rt(x,a) + Vt+1(Γ(x,a)) } 更新函数，比较前一时刻和当前时刻

向后递归，优化等式

Bellman的DP是线性规划的一种，主要在 operations research，不用 优化 是因为 太数学了， cs中多是 递归解决子问题

General formulation 有限 horizon DP 决策问题



Stochastic Deterministic

例子：背包问题，关注价值V 和W-w

期望问题：丢撒子的期望值die =Σx=1 6 x P(X=x) = 1 1/6 + 2 1/6 +...+ 6 1/6 = 3.5

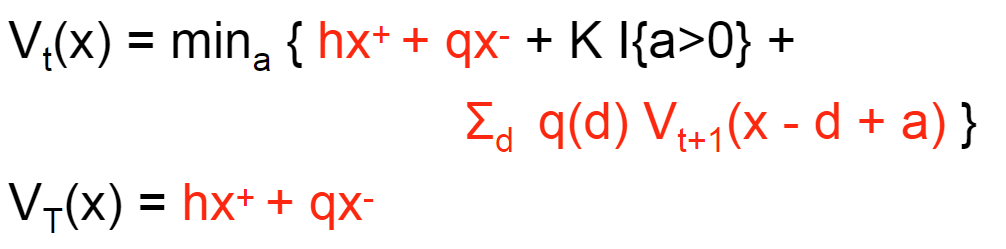
丢塞子的平方期望：91/6

期望主要和 frequencies有关

在stochastic情况下的bellman principle是：Vt(x) = maxa { rt(x,a) + Σy pt(y|x,a) Vt+1(y) }

新状态由概率期望相乘累加得到，不是简单的计算，因此 path不能直接计算得到，只是一种simulation

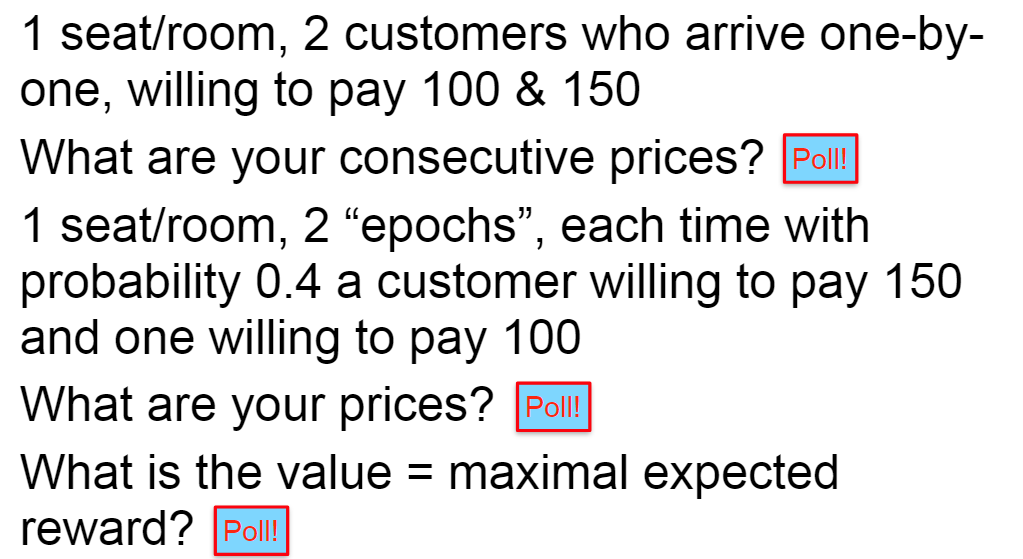
库存问题：每次的交货d是random，P(D=d) = q(d)，向后订单则costs q per item

Optimal等式则为：

原来的直接继承原状态变为概率累加

例子 Revenue Management 收入管理 RM

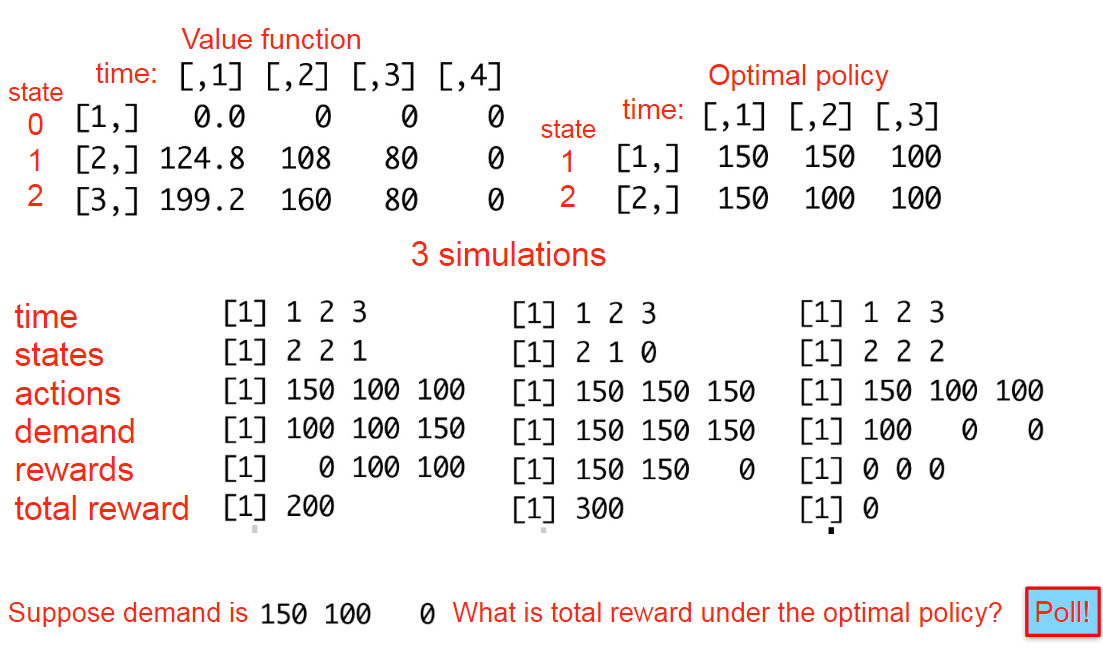
Airplane RM 有常规和特价机票，有限座位，提前订票。问题：提供多少个seats在不同的booking classes，何时关闭预定？



问题1：连续定价 150和150，第二个人肯定付款

问题2：根据概率定价 100和150

问题3：最大期望值



有2个座位的dp情况，问题：模拟的最优情况，state 2 2 1，250

顾客消费模型 每个顾客有消费意愿f 1 period 1 request，request的概率基于class和time

价格有波动，添加组件记住上一价格，价格下降增加penalty

RM多维：空中网络，联程，状态成为（x1，x2…）维度爆炸，n个航班，每个C载客则有C（C+1）^n。解决办法 近似

预测和学习

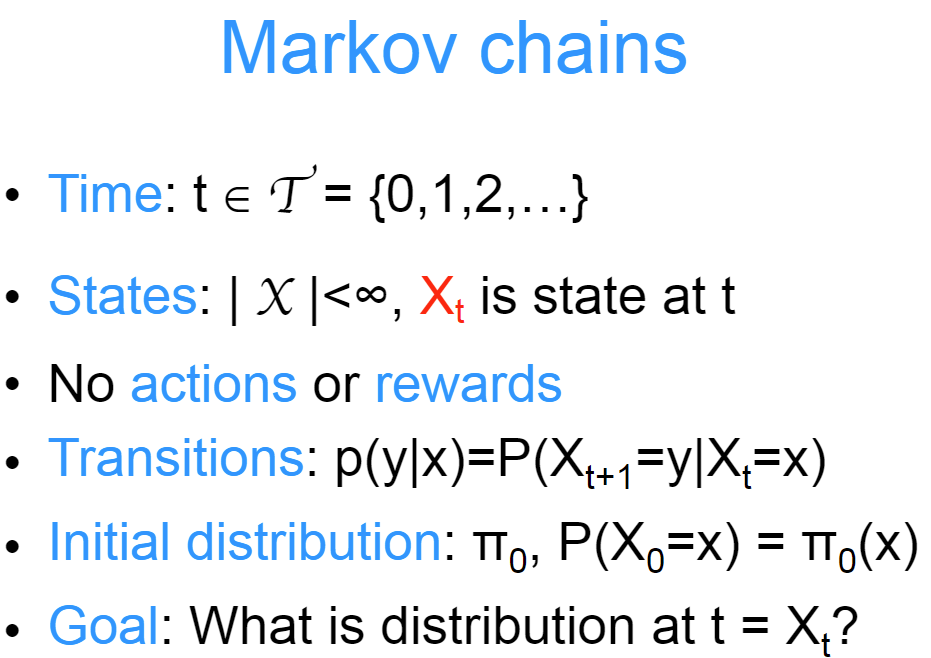
用统计的办法预测 partial info model 在抵达状态时根据更新的unknown数据学习 online

**Markov (reward) chains 马尔科夫链**

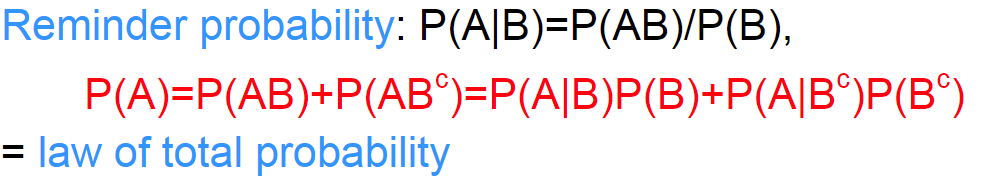
Model for times dp中的时间模型：

1. Finite horizon 有限域，T={1..T}, 算total reward 好例子 RM问题，knapsack 坏例子 没有clear的T，最短路径，库存管理
2. Infinite horizon 无限域，T={1….} 算total reward 直接的reward必须为0，不然不好计算 如最短路径 在时间T够大的情况下，也是无限域问题
3. Infinite horizon，average reward
4. Infinite horizon，discounted reward 这俩例子都是 库存管理
5. Continuous time 连续时间，infinite horizon 无限域，不管discounted还是average

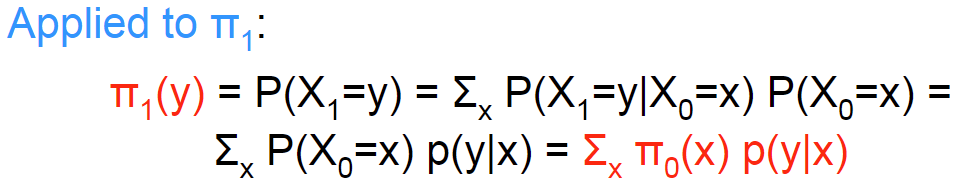
Markov chain是 long-run average reward的背景



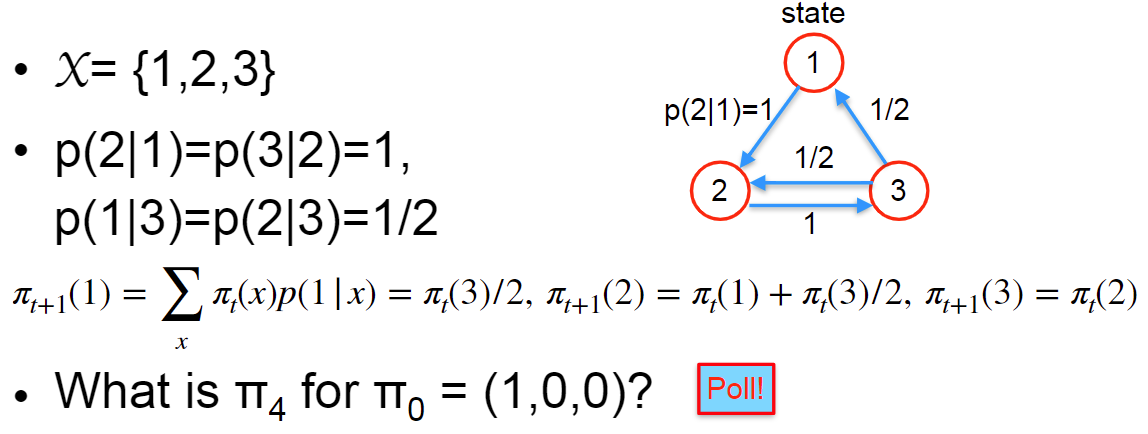
时间是无限的，所以state数量也是无限的，没有任何reward，单纯的概率转移从x到y，目标是找到t时刻Xt的分布（概率）



从B到A的概率计算，全概率公式 应用到分布pi1状态y，初始状态x有



初始x分布的概率 \* 从x转移到y的概率（源节点概率\*转移概率）

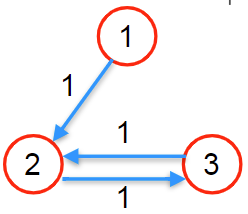


初始状态为1，pi1=(0,1,0)，pi2=(0,0,1), pi3=(1/2,1/2,0),pi4=(0,1/2,1/2)

Simulation 是 alternative 所以要模拟多次，采frequency 上面的转世过程中：traces 1,2,3,1,2 and 1,2,3,2,3 both occur with probability 0.5

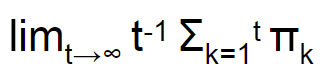
MC的性质：定义：path是chain of states x1 x2 xn… for which p(xk+1|xk)>0

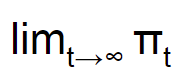
MC应该是 1. communicating 交流的：任意两个状态之间存在path 2. Aperiodic 非周期的：所有从x开始的path的长度最大公约数 应该是 x=1到x之间的任意值

这个MC是不 communicating的，3到1没path，而且是周期的 2 3循环

Pi4 = （0,0,1）

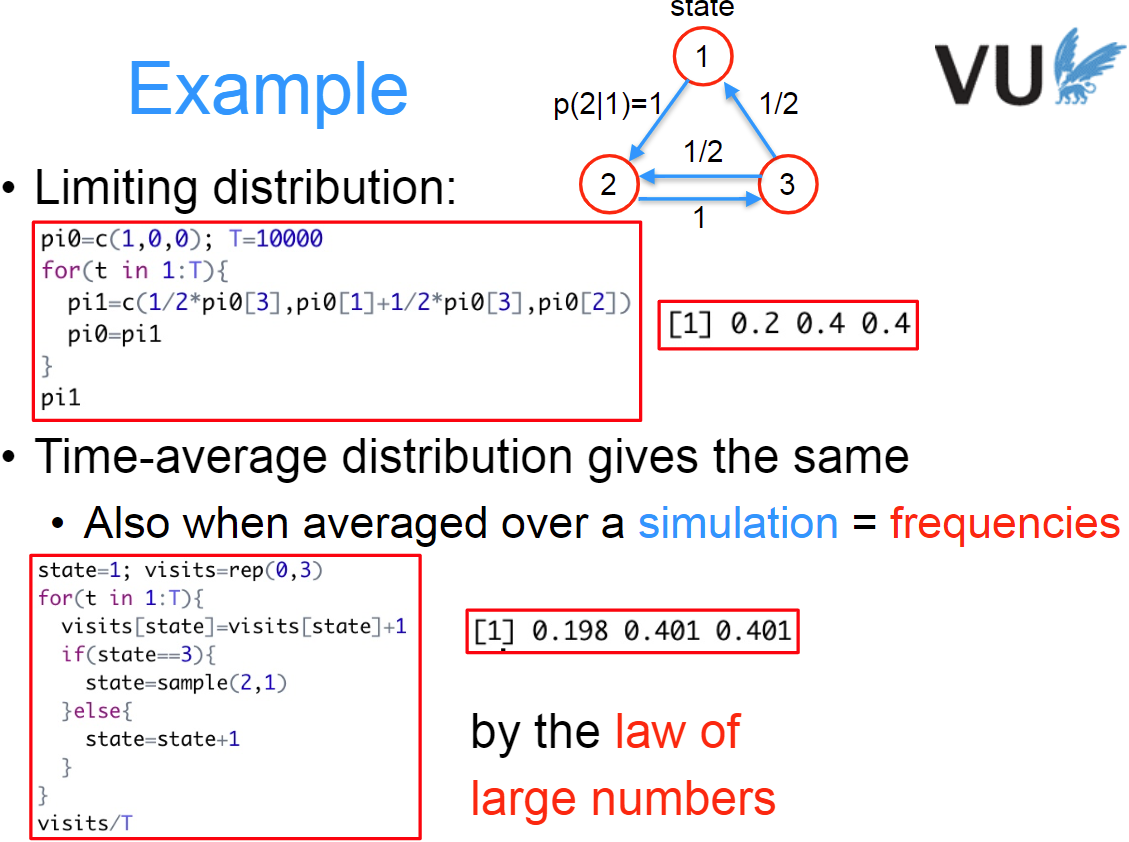
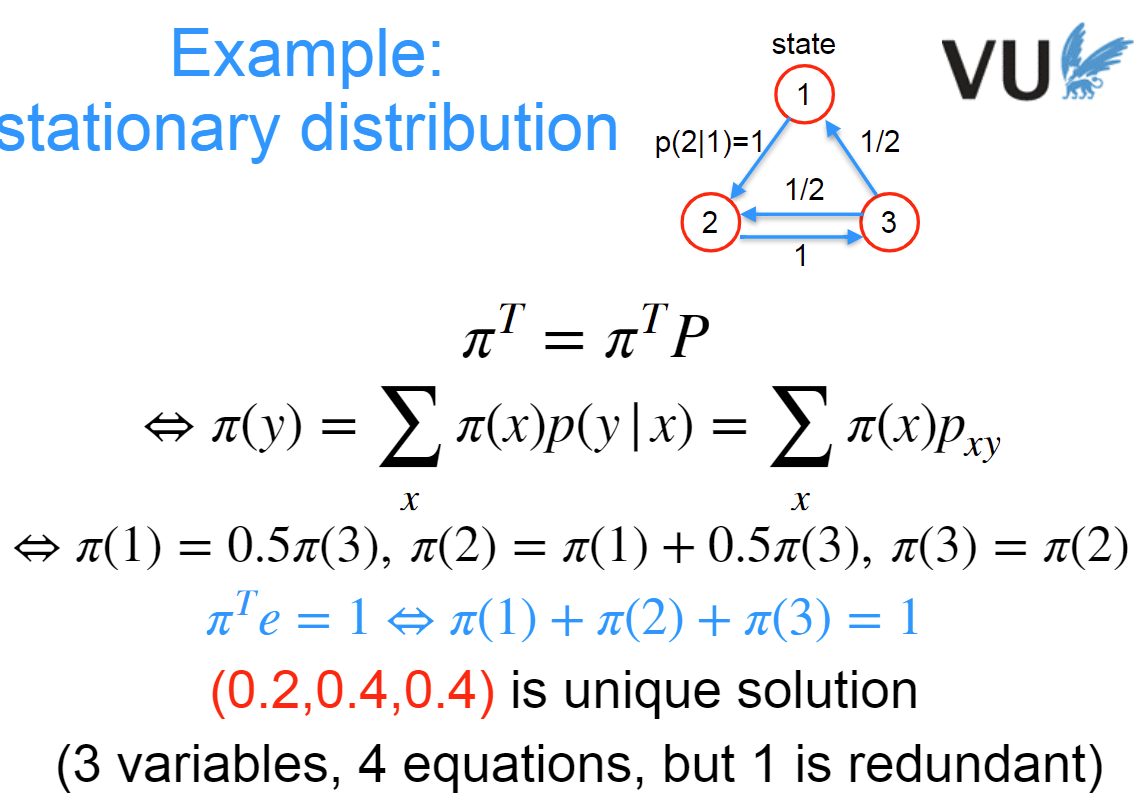
Long-run behavior

Time-average 分布：

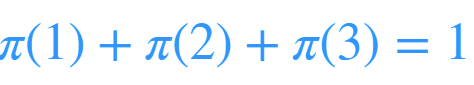
Limiting 分布：

Stationary 分布：solution of π^T = π^T P with π^T e = 1

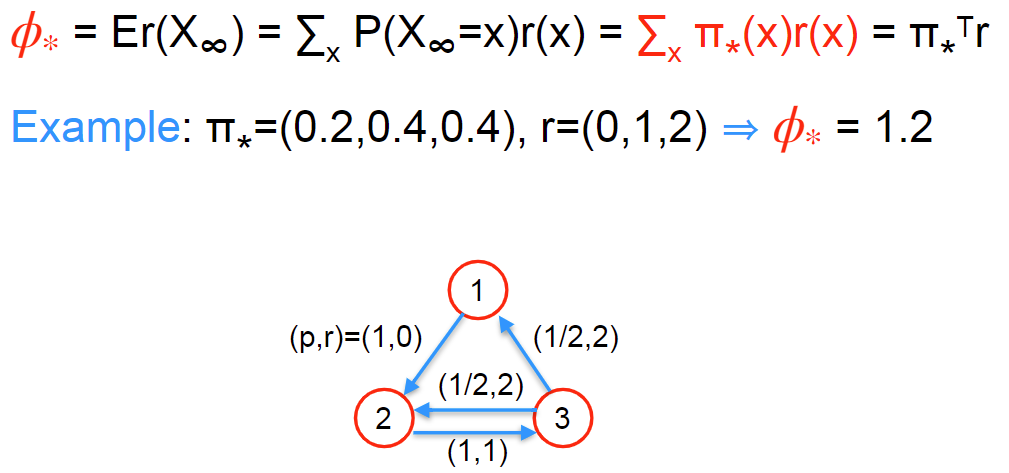
理论上：对于非周期 交流的MCs，上面仨都一样

在limiting分布中，给出指定时间T，计算当时的概率。 在time-average时候，采用frequencies，统计每个state访问的次数，除以任意时间长度T。在 stationary 分布中，解方程组，求pi123

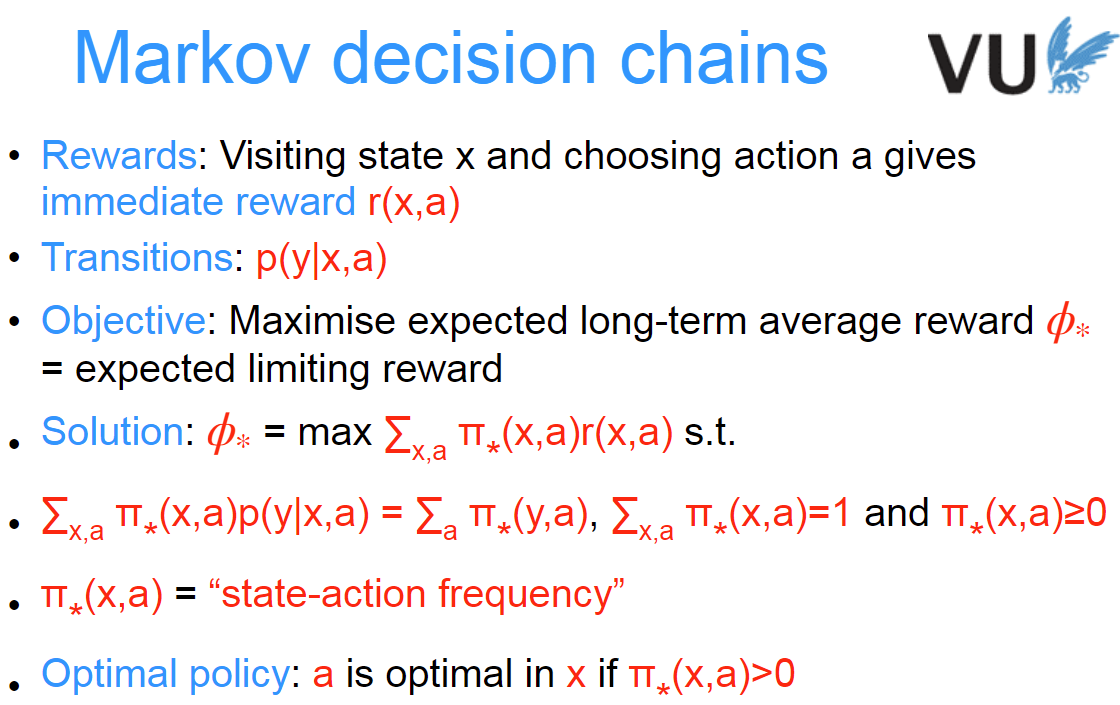
 

**Markov reward chain**：访问state x立马给reward r(x)，with reward long-term函数如下

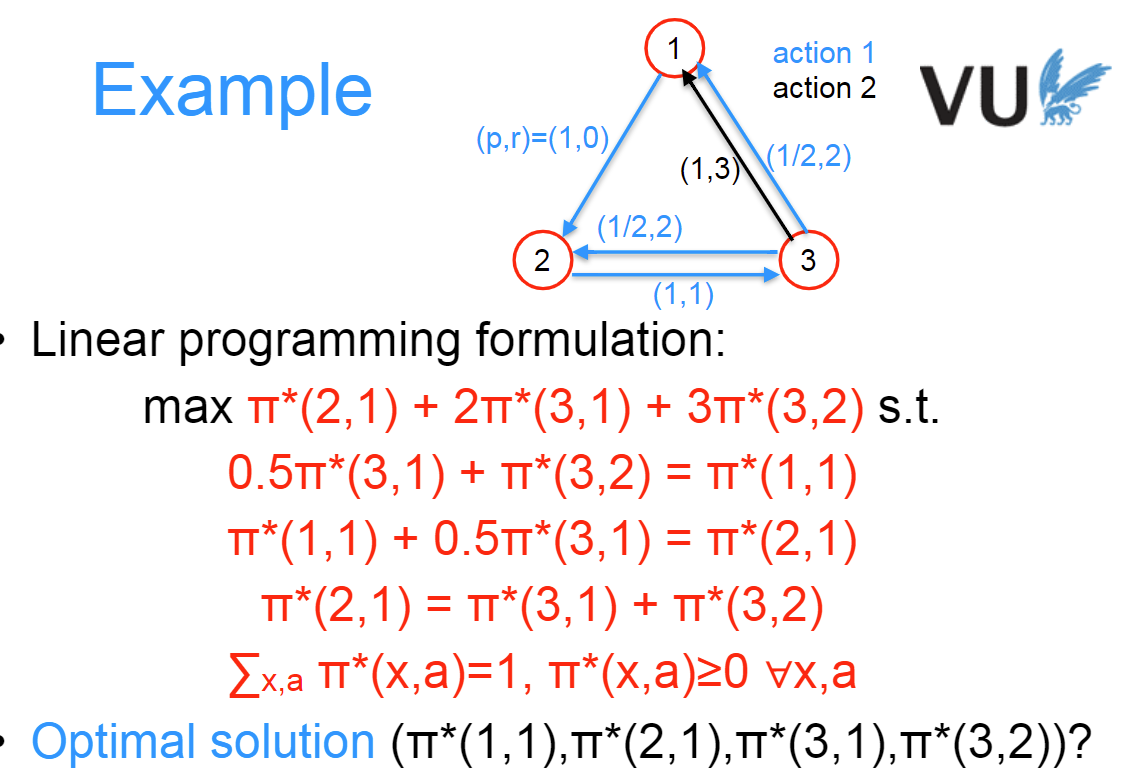


在原先的基础上，每个状态都 乘上 reward即可

Markov 决策链



到一个state的时候，还会立即给一个 reward r(x,a)，a就是action，同一个state间的转移有多个actions，goal变成找最大的reward了，即optimal



该例子中，变量数大于equation数，所以必须有一个solution是0，不然无法解

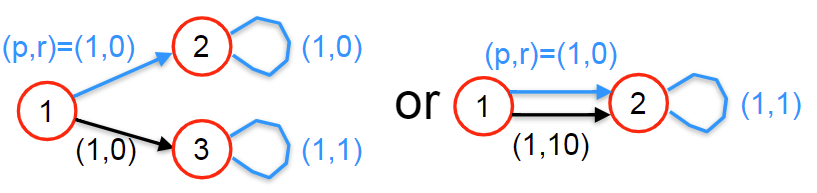
3到1有俩action，一种概率1，得3，一种概率0.5得2. 求解最大化总reward，期望用加法累加（在某一个state上概率和大于1） 但是 **总的概率和必须为1**

答案 （0.2, 0.4, 0.4, 0）根据答案反代符合要求即可

但如果直接计算 (1/3, 1/3, 0, 1/3) 不走state3的action1，optimal reward有4/3

用线性规划库 CPLEX at neos-server.org 求解

LP的disadvantage：计算慢，只能求解小instances；只能计算communicating chain的long-run average，左图无法计算，因为不communicating，除了初始无法走state1, pi1只能为0



好处：很方便增加constraints

**The Bellman equation** 贝尔曼等式

LP求解 MC很慢，要faster且sensitive的办法

**Poisson equation** 泊松方程

反向计算 φ\* = long-term average reward 在Markov reward chain中

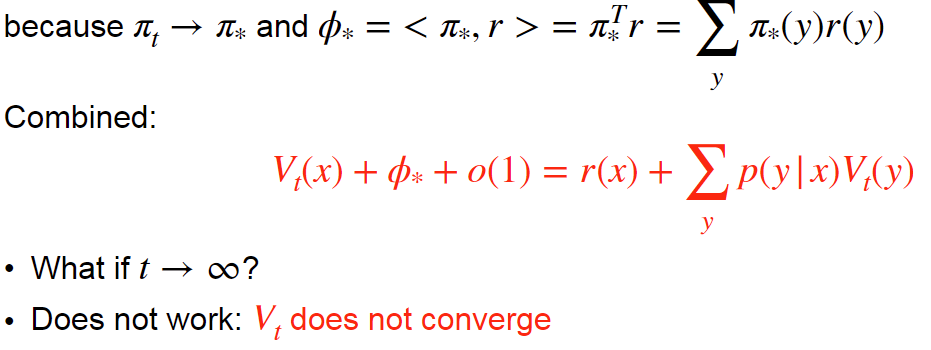
使用 Vt(x) = total expected reward over t time starting in x 从x状态开始经过t的总期望值

需要 o(f(t)) 在 g(t)=o(f(t)) if g(t)/f(t) → 0 as t →∞ 要么f无穷大, g无穷小, 时间趋近无限

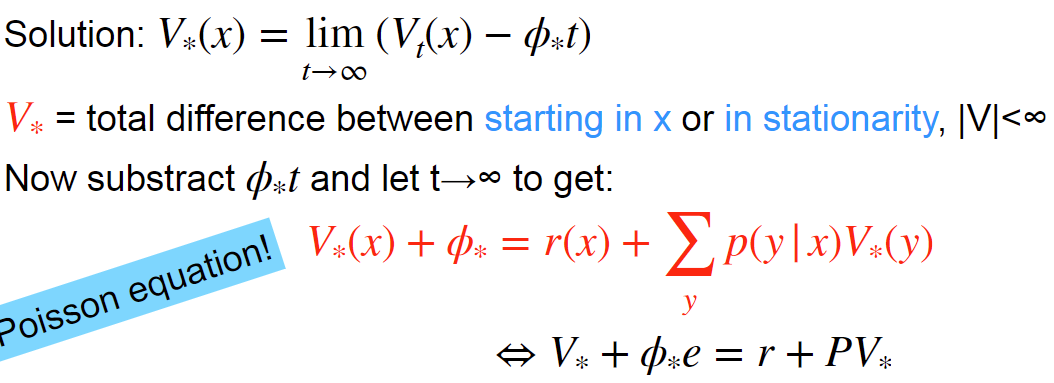
主要用途: g(t) = o(1) if limt→∞ g(t) = 0 在时间t趋近无穷时, 有g(t)=0, 则有该等式

已知 Vt+1(x)=r(x)+∑y p(y|x) \* Vt(y) 在t+1时刻的 reward

和 Vt+1(x)=Vt(x)+∑y πt(y)\*r(y)=Vt(x)+φ\*+o(1) 从t时刻reward推t+1的reward的等式



化简得到:

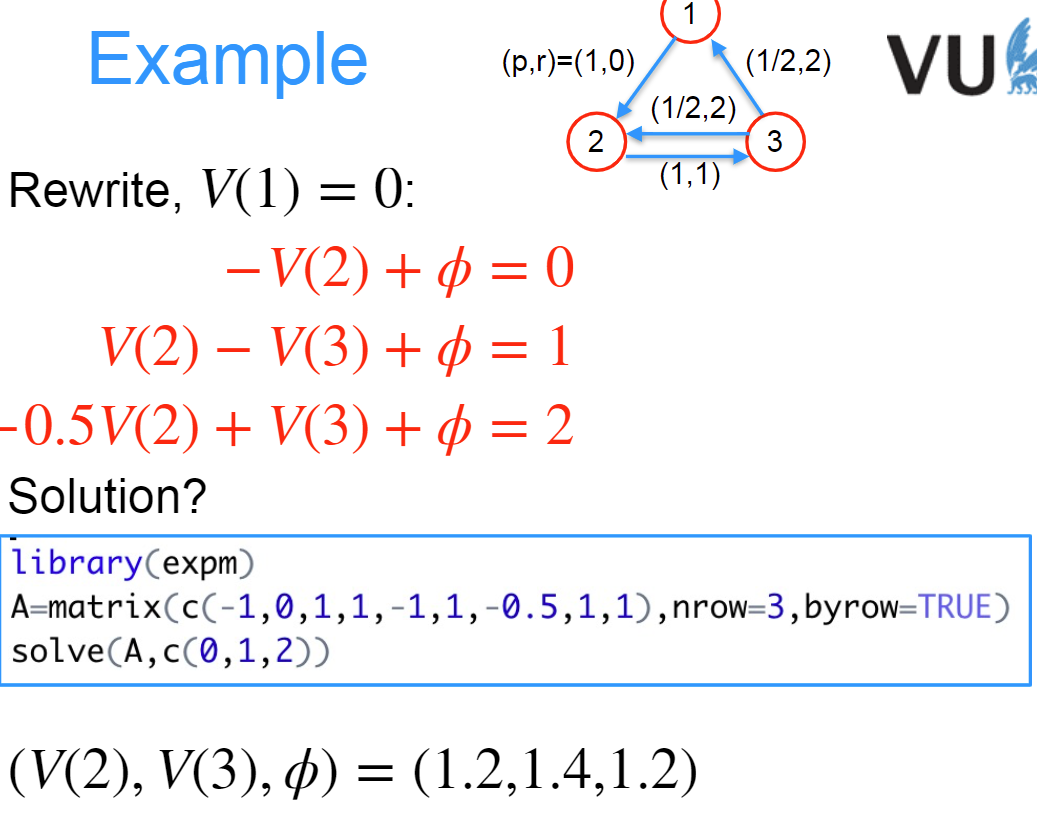
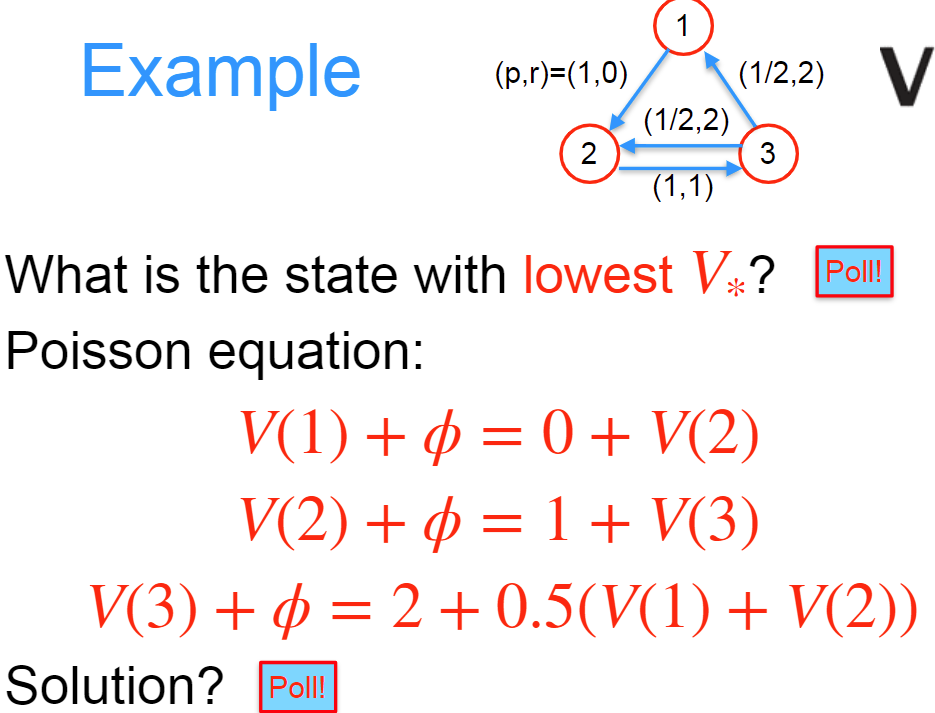


V\* 是初始状态x和经过时间t之后的稳定状态之间的总差

V+φe=r+PV 没有唯一解, (V\*+ce,φ\*)∀c都是解, 还需要 <π\*,V\*>=0



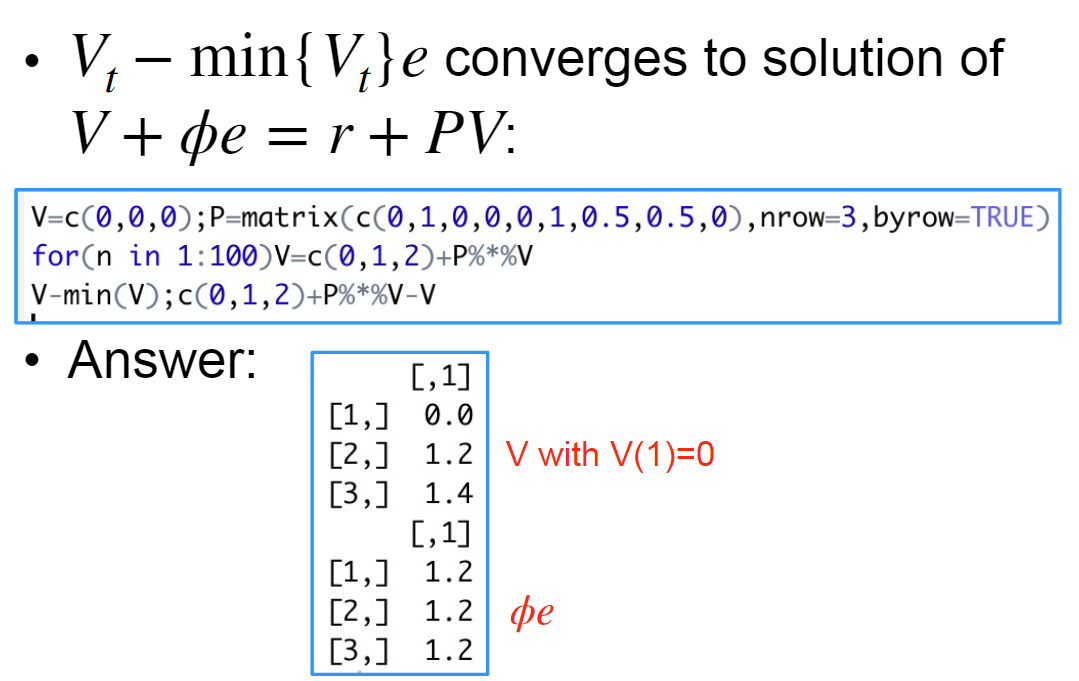
例子



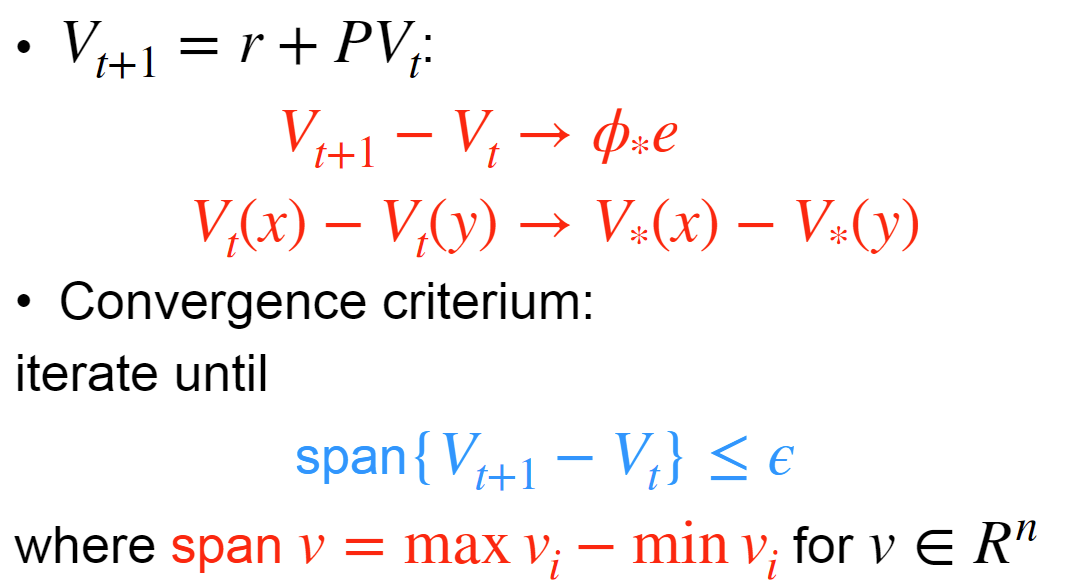
从 State 1开始有最低的V\*, 因为从他开始的reward最低 计算poisson 从答案反代

把有最低V的状态1,V(1)设为0,计算剩下V和φ

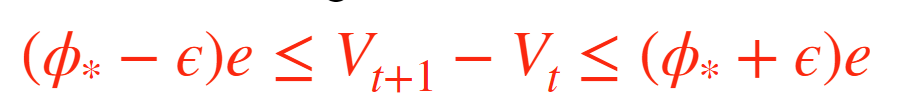
递归解决



反向递归 收敛 收敛条件为 span{Vt+1−Vt}≤ε

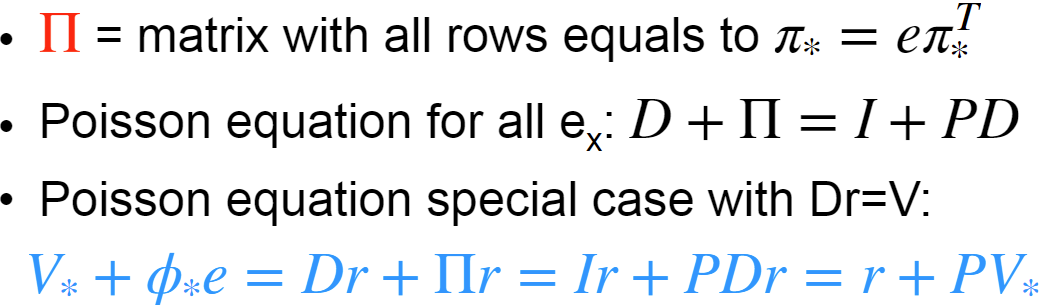


算法: 1. Take 任意一个 V0 2. 迭代Vt+1=r+PVt直到收敛条件 span{Vt+1−Vt}≤ε

理论上:算法会在这个区间停止

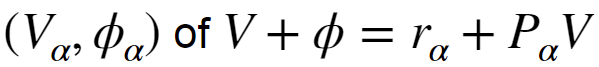
计算方法: Deviation matrix 差矩阵

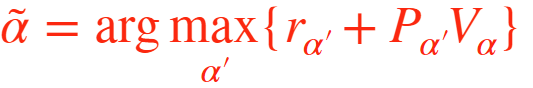
让 r = ey则有 V\*(x) = 从x开始,访问y的次数与平稳状态相比 D(x,y) 就是 deviation矩阵



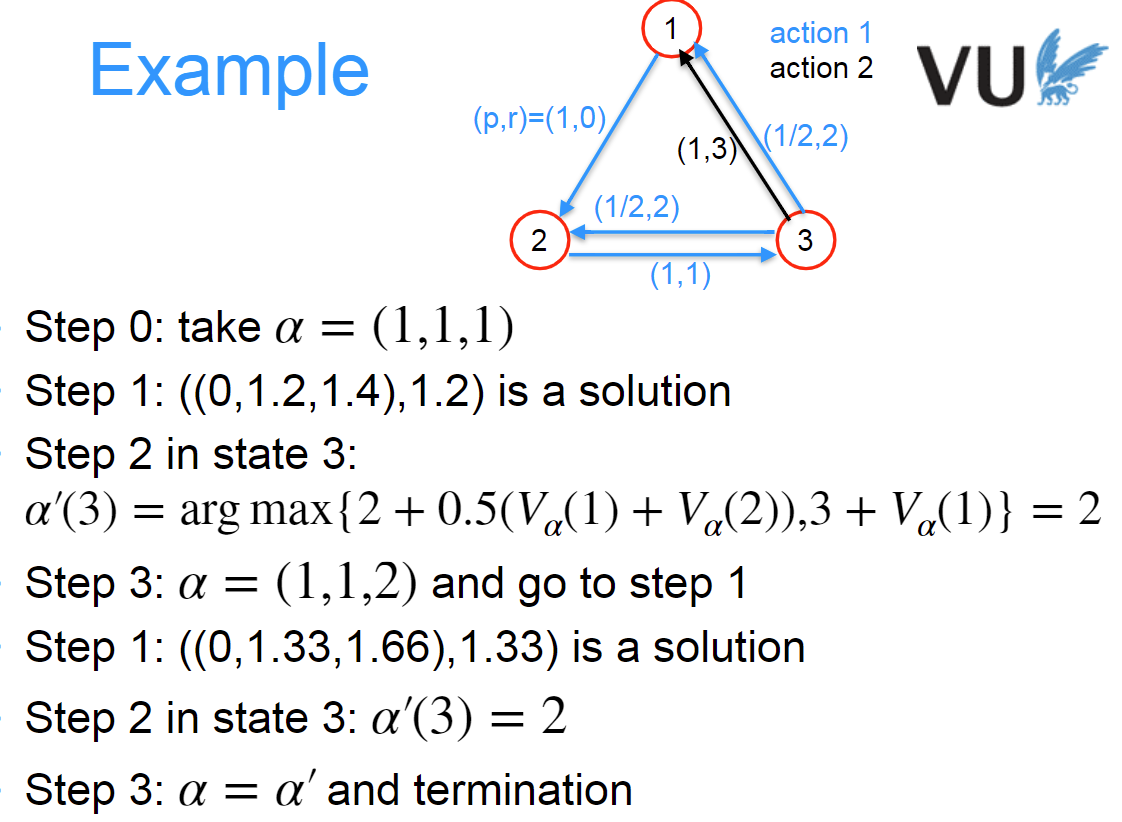
MDC policy迭代

定义: 

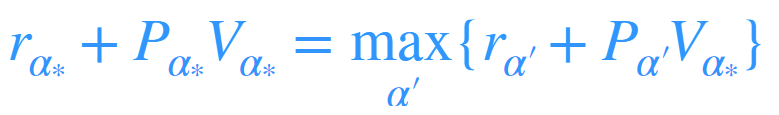
Policy 迭代: 1. 固定policy a 2. 找到一个solution 

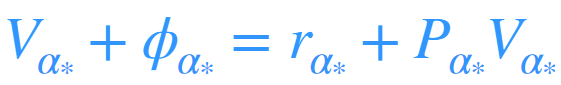
3.  4. 如果 a 和 a~相同则有最优policy, 否则将a=a~,重复1

例子 a就是决策,走action 几 当前测试的决策和最优决策一样就ok



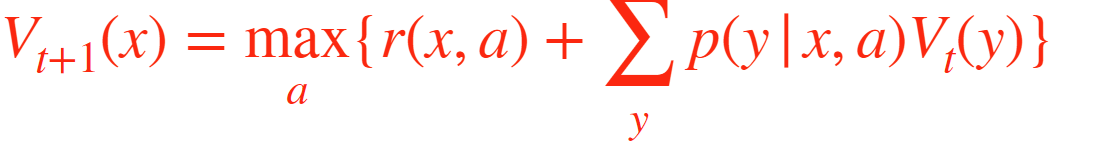
**Bellman 等式**

对于最优决策a\* 有 

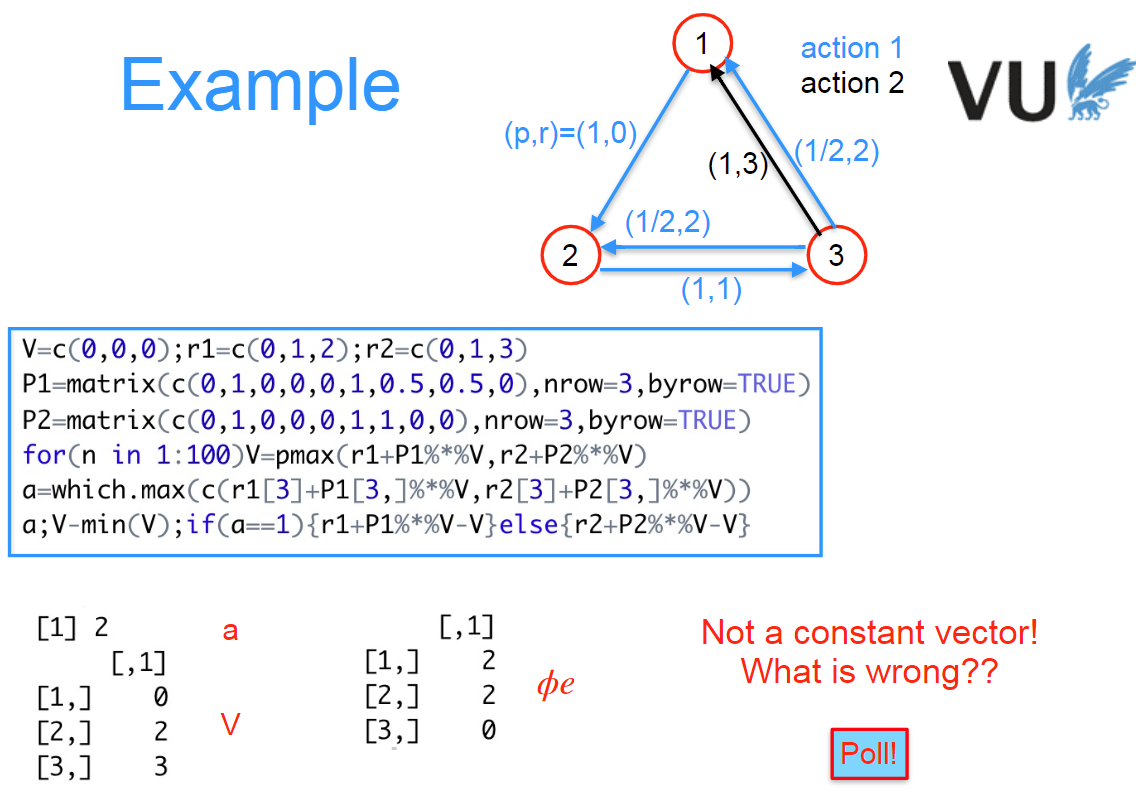
对于任意决策 a\* 有 

将两个结合就有 就是 bellman 等式 也叫最优等式

新的反向递归 MDC 将递归换为 

每个状态就是一个vector: 

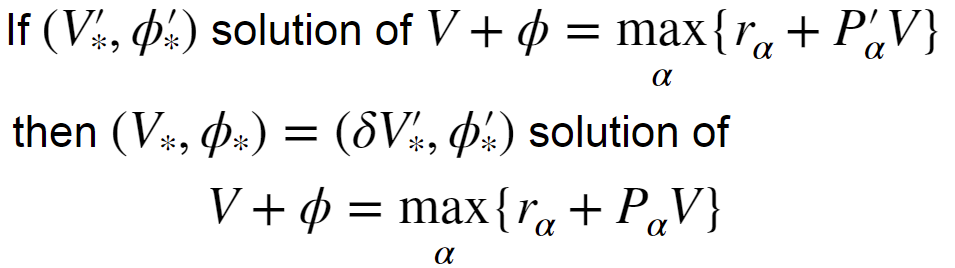
最大policy for t 就是 long-run optimal policy

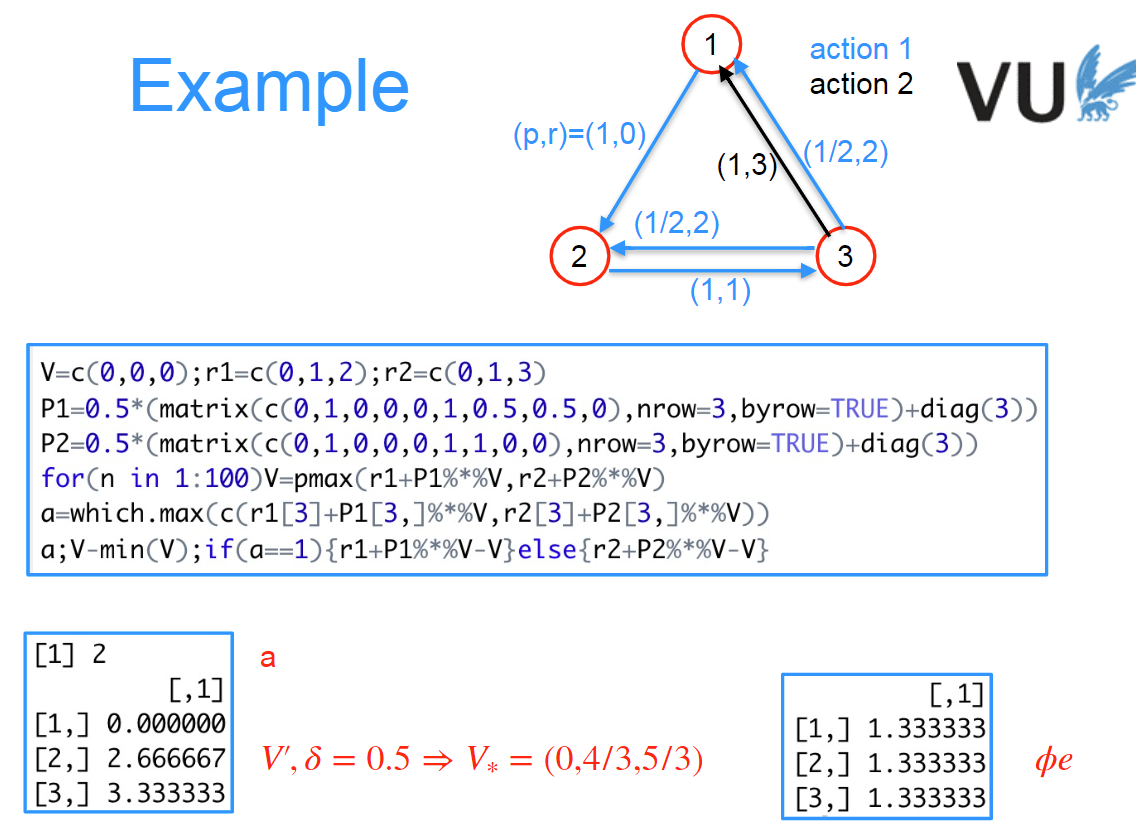


Periodicity 周期性

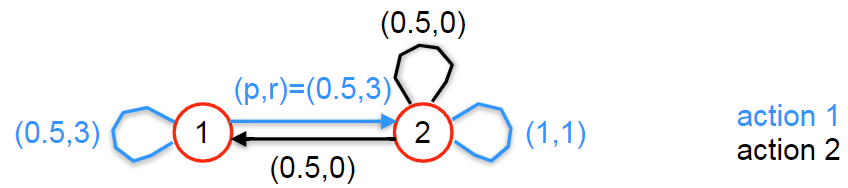
考虑一个 Markov周期决策链 用poisson/bellmen解决好 但是并**不会收敛**

将收敛条件改为: by P’=δP+(1-δ)I P`是非周期的, 这样就能收敛了

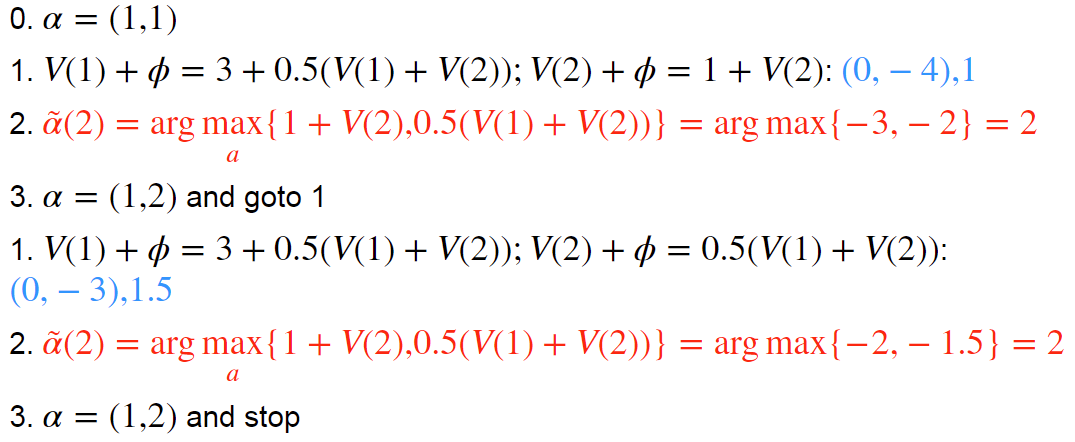
例子



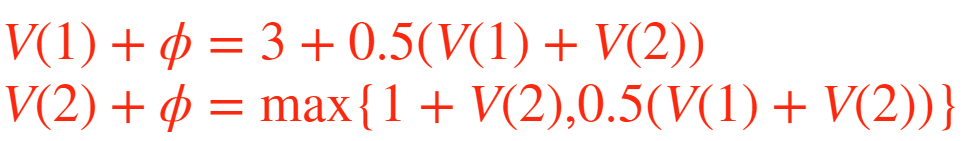
**非连通链**: 对于连通链, long-run average reward是独立的于初始状态和分布, 但是非连通链和初始有关系 例子



策略迭代



贝尔曼等式

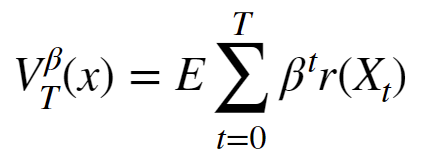


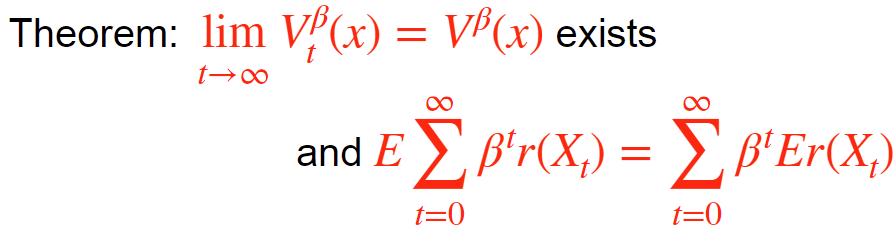


Discounted rewards 打折奖励

现在的reward比future的更有吸引力 有参数ρ reward变为 x(1+ρ)^t at t

其中 β = (1+ρ)^t就是 discounted factor, β属于(0,1) **初始状态很关键**

Total discounted reward 理论:

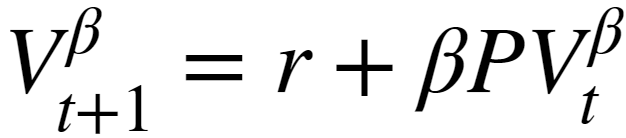


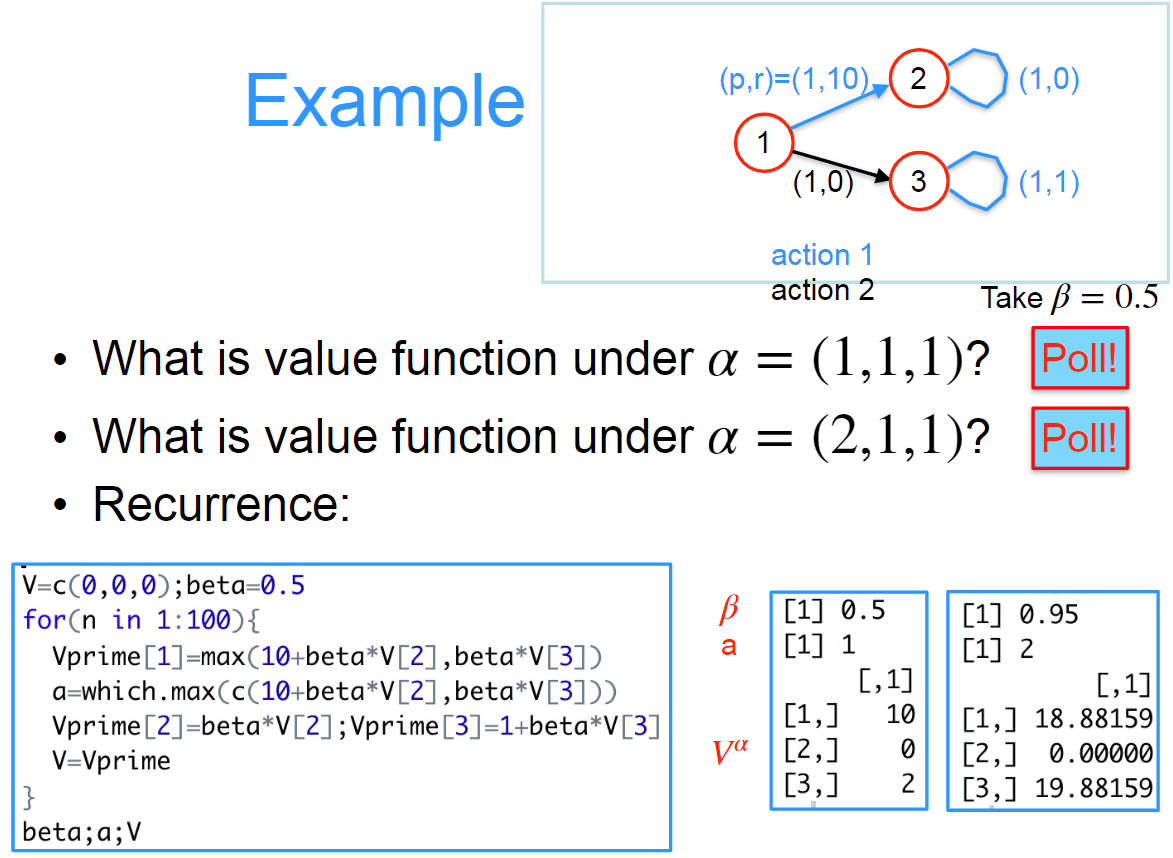
时间趋近无穷时, 二者相似

Discounting vs averaging

时间久了都一样, 但是呢, discounted factor beta影响的是未来因素下降程度, discounting是一种更简单的数学模型, 另一种解释: 依然是total reward但是 在每一步都会 stop with Probability 1-beta **reward在t时刻, 有 beta^t的概率不停止**

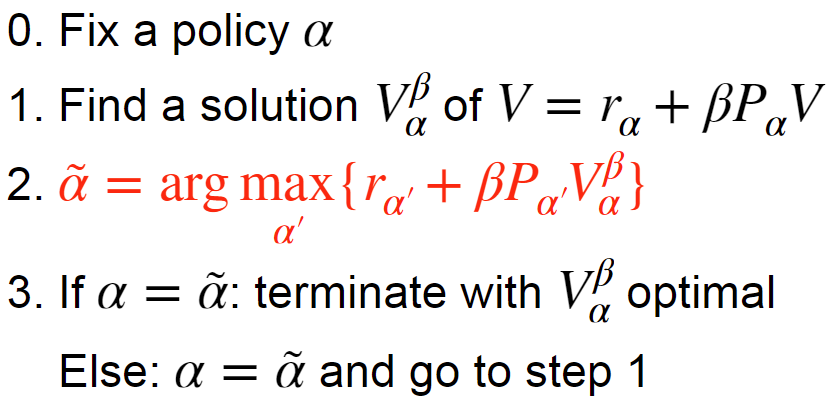
Discounted Markov reward chains

Vβ 是 discounted Poisson equation:  反向再现: 



**2怎么来的: 1是V(3)的reward 1+0.5^1 + 0.5^2 + 0.5^3 + … = 2** 级数

Discounting: policy iteration



**Policy 决策**

决策是不可以 look ahead 并 anticipate 预测的 决策可以是基于 previous states的, 也可以是 randomize 随机的

Framework: finite-horizon MDC 有限域决策链

Definition: policy是memoryless (也叫 Markovian), 当 当前策略只和当前状态有关



Transition是memoryless当 当前state转移只和当前状态和策略有关

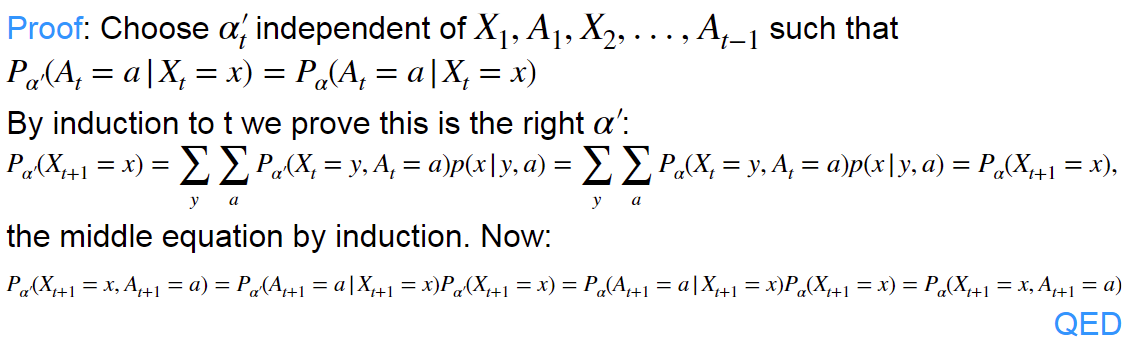


Theorem: 对于给定的state x1 和 policy a=(a1 … aT) 有一个 memoryless policy a` 满足

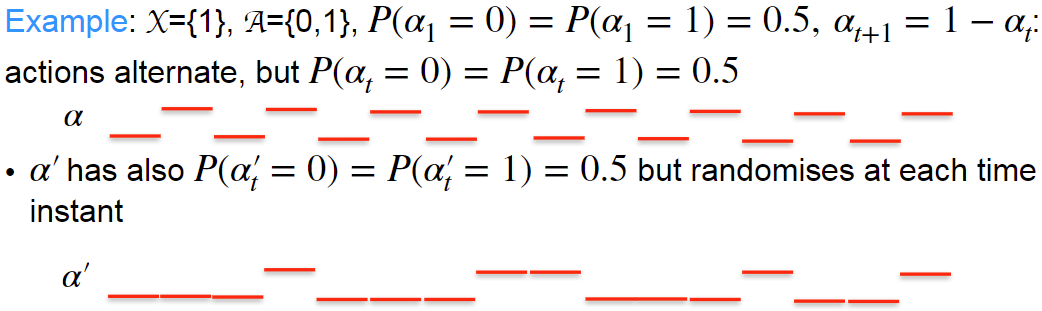
 即 存在一个策略和当前给定的一样

Consequence: 对于每个policy都存在一个Markov policy和这个policy有一样的reward

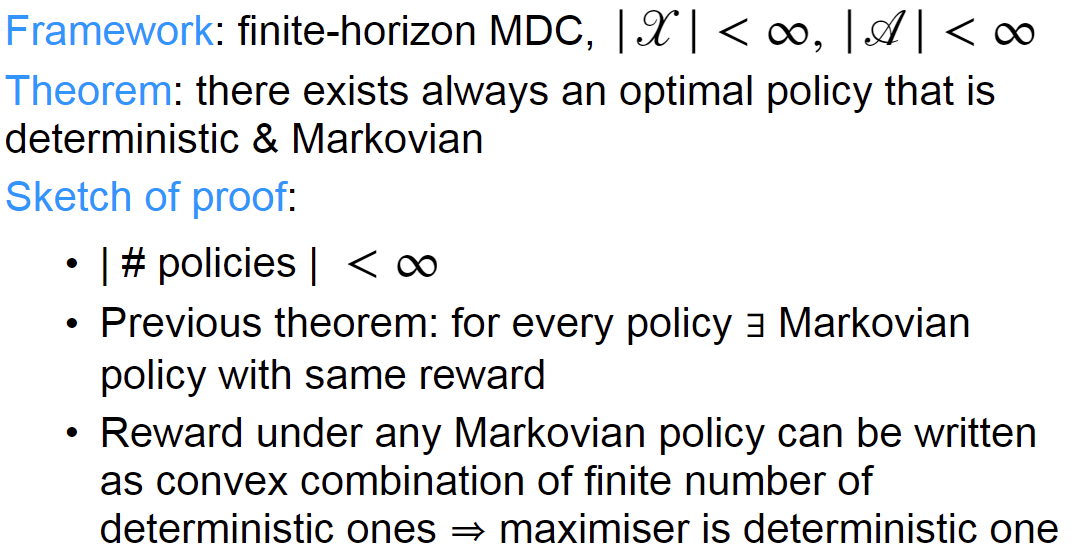
(POLICY始终存在解理论?)



Note: 边际概率可能相同, 但是过程不一样 相同概率不同过程的例子



最优policy存在理论 始终存在一个最优policy



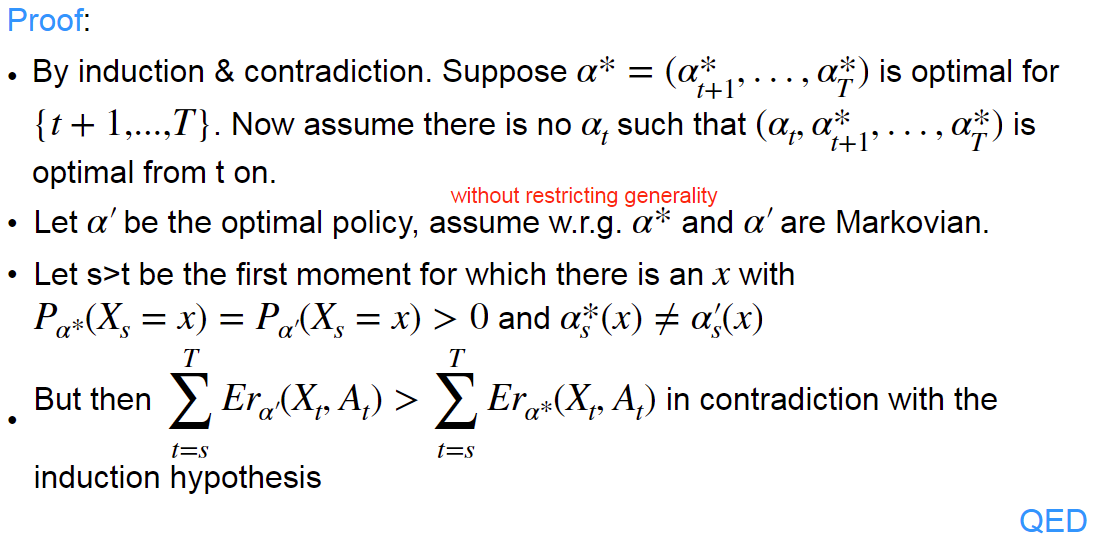
之前是任何决策都存在相同的reward, 后来是Markov决策都能写成有限的图组合, 能算

Bellman的优化原则

Framework: finite-horizon MDC

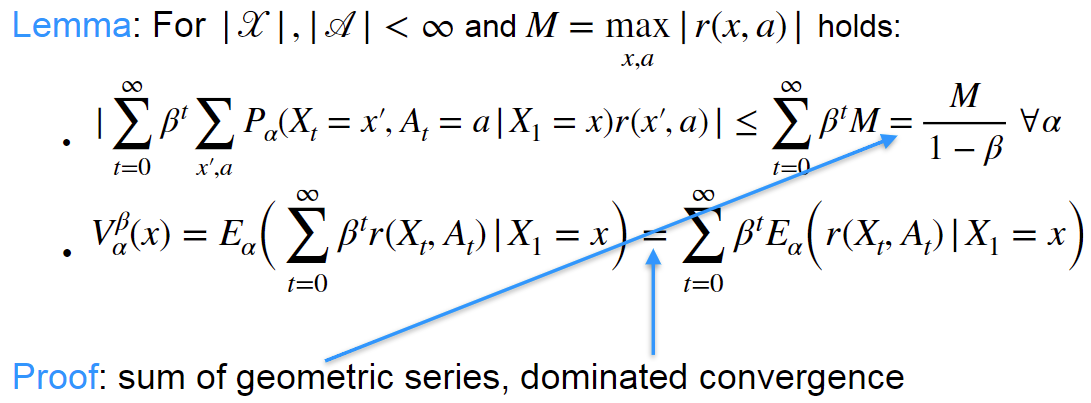
**Theorem: What is optimal from t+1 to T is part of an optimal policy from t to T**

**从t+1到T的最优也是 t到T的最优的一部分!!**



Discounting 打折模型 级数的和

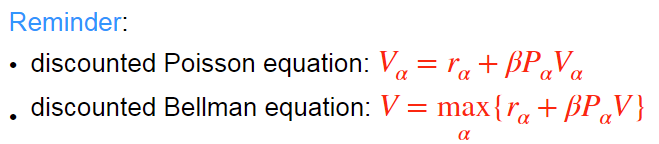
Framework: time-homogeneous discounted ∞-horizon MDC



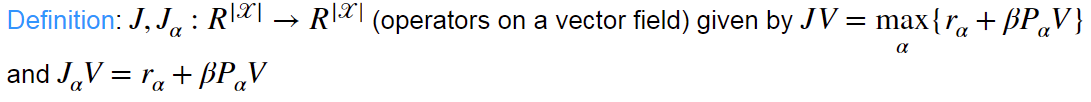
Consequence: when discounting total rewards converge, maxima exist, etc.

当reward收敛, 存在最大值

Discounted Bellman equation 打折贝尔曼等式 打折泊松和打折bellman



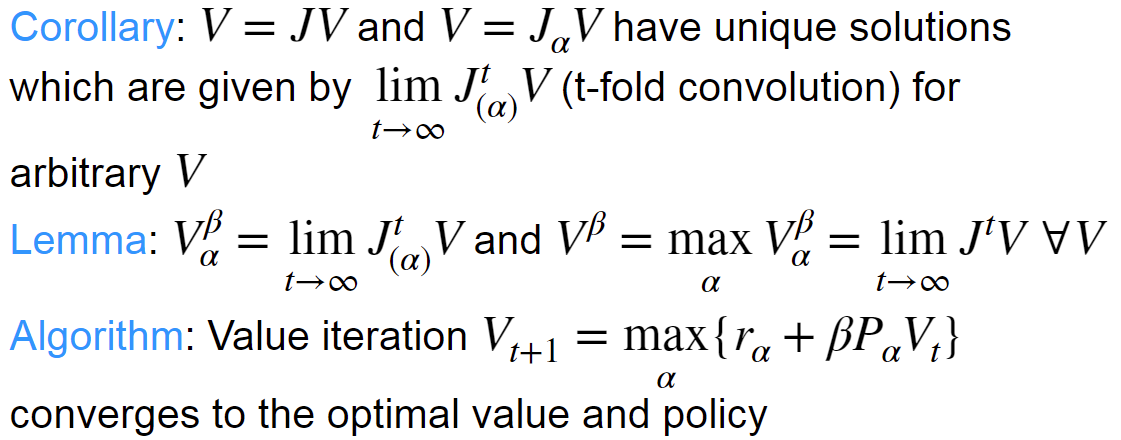
Questions: 是否存在一个独一无二的解? 怎么找?



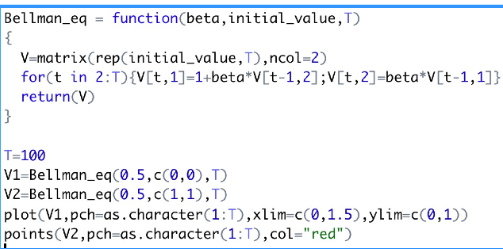
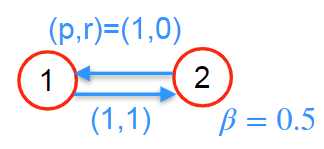


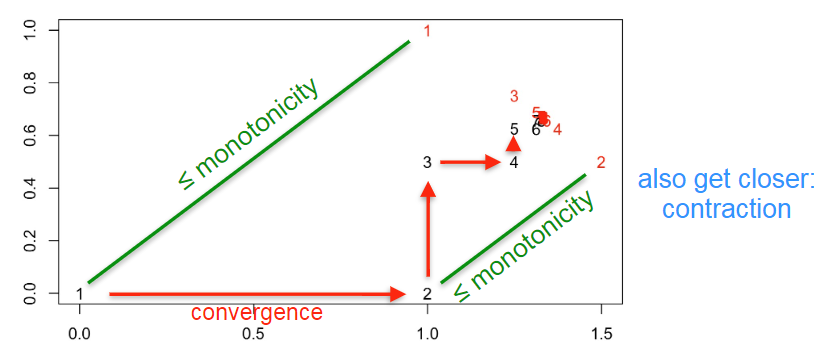
所以存在一个唯一解在 打折贝尔曼方程中

Theorem: 如果B是一个单调的收缩映射, 则 B^t \* V (the -fold convolution of ) 会收敛到唯一解 V=BV

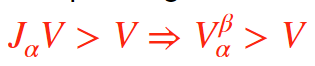


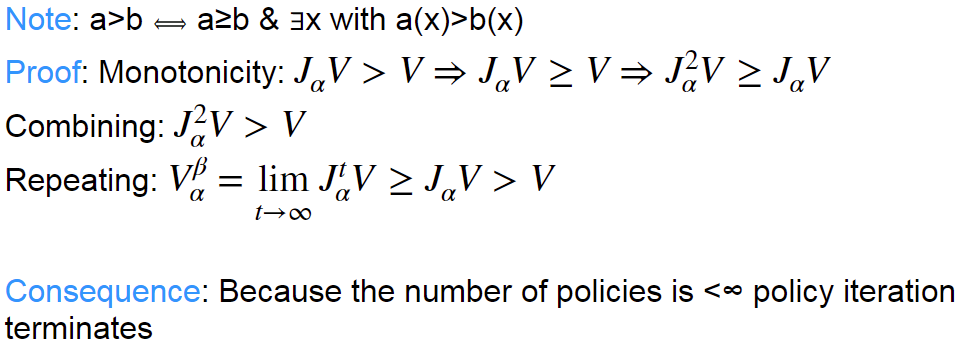
例子:



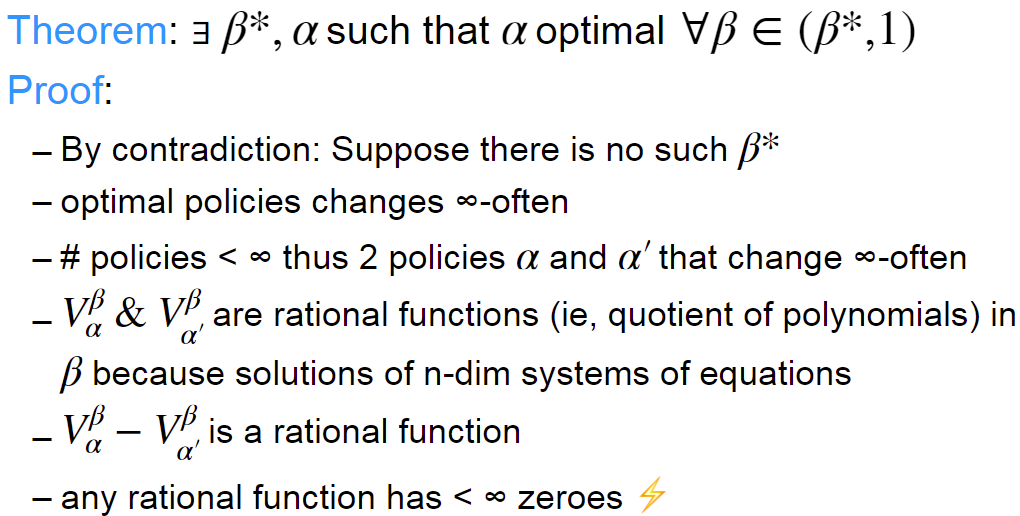


打折的 Policy iteration

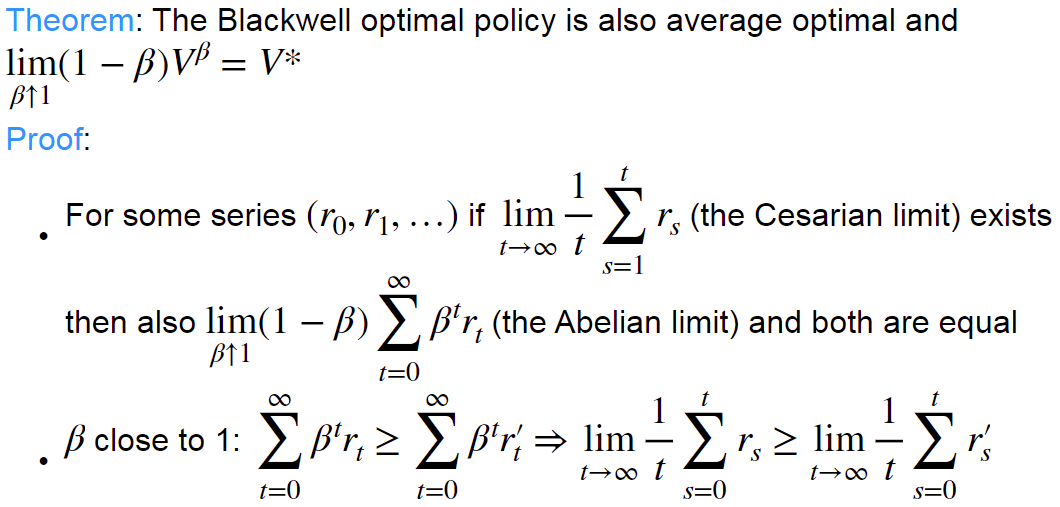
理论: 



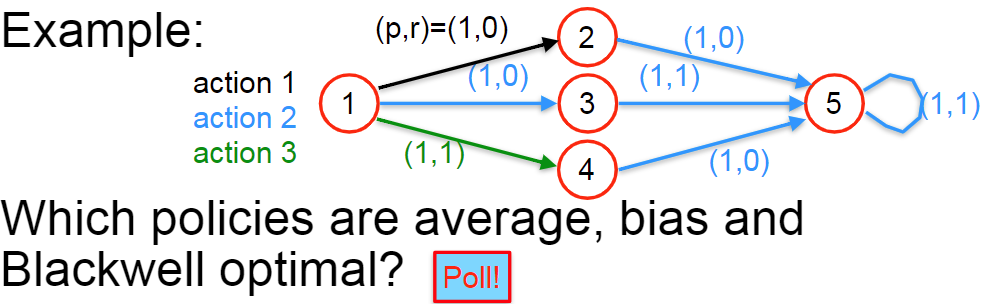
Beta close to 1的情况: 存在一个最优策略a, 在(beta, 1)的所有beta可能性上都最优



Blackwell optimality 上述这样的a是 Blackwell 最优 这个最优策略也是平均最优策略



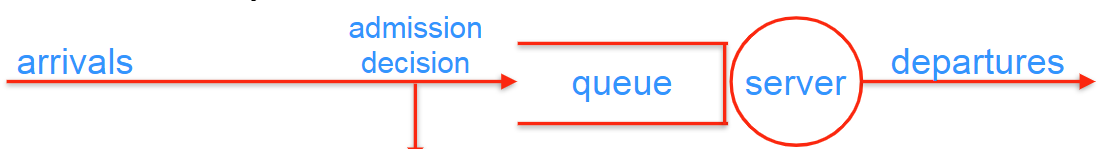
Discounted rewards 最大化V\*的策略叫做 bias optimal, V\* 也叫 bias vector



**Continuous time 连续时间问题**

Discrete vs continuous time

很多问题都是发生在连续时间的 如 排队问题 在事件发生之后才知道后续的事情



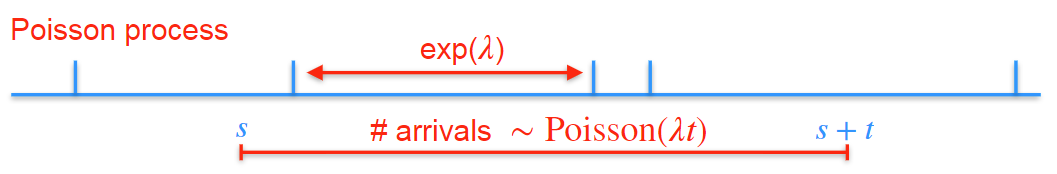
这种 到达 通常根据 Poisson process 并 按 指数分布 提供服务

到达时间 和 服务时间 通常是指数分布 Arrivals是empirical reasons的 服务时间是 数学方便 指数分布是**memoryless**的

Reminder Poisson process 泊松过程

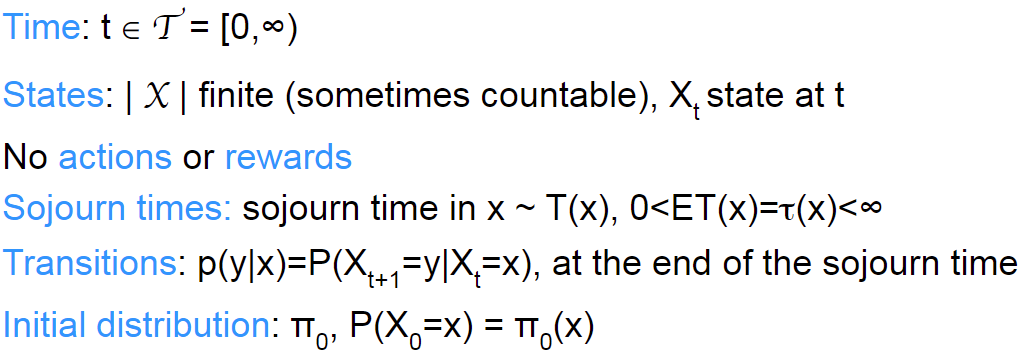
理论: 在间隔上, 有 ∼exp(λ)⇒ 到达 in [s,s+t]∼Poisson(λt)

N∼Poisson(λ)⇔P(N=n)=λnn!e−λ,EN=λ N 有 泊松分布, 到达过程就是 泊松过程

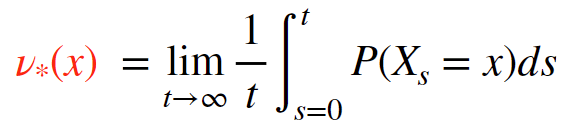


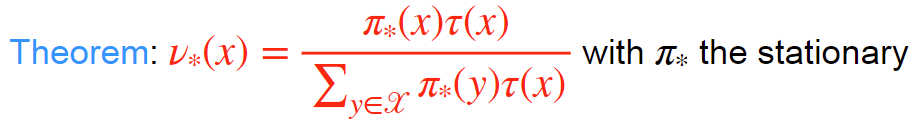
Semi-Markov processes 微马尔科夫过程

在原本的markov chain上加入 随机的 sojourn time 逗留时间 其余和以前一样

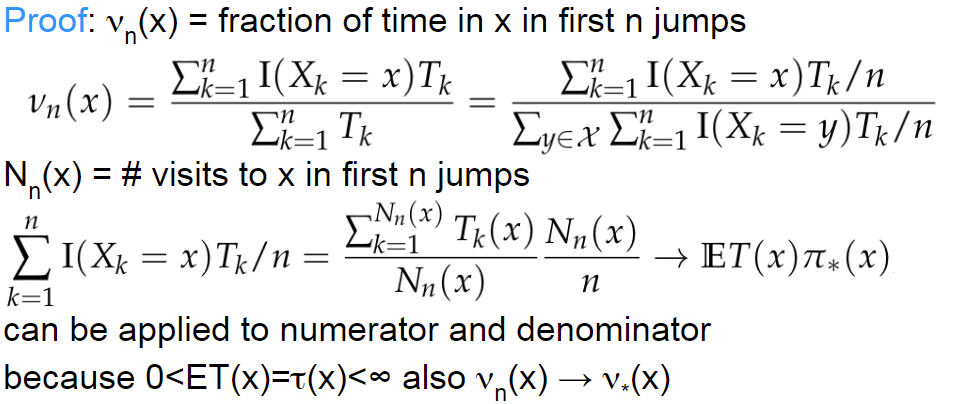


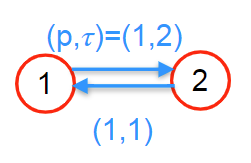
目标: 确定long-run的平均分布 v\* 通过 ν\*(x)=limt→∞1t∫ts=0P(Xs=x)ds





Pi\* 在 jump time有 平稳分布



 pi\* 是

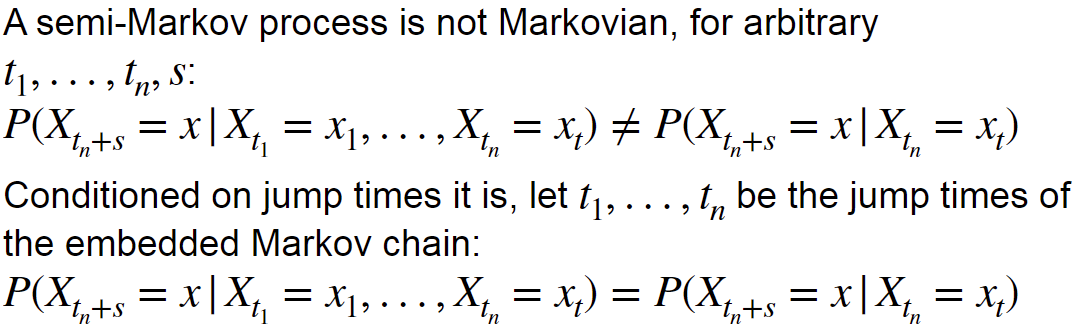
V\* 是

不存在 time-limiting distribution 有限时间分布

Markov processes 马尔科夫过程

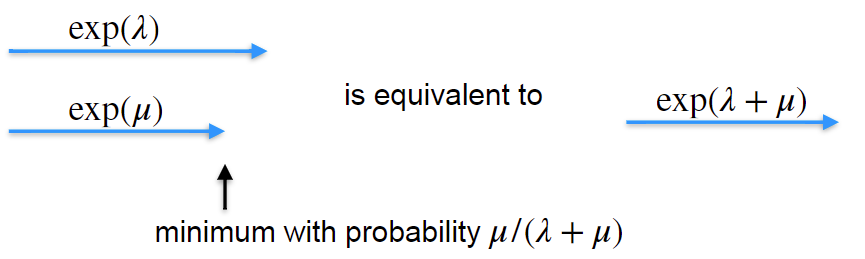
用chain来计算 离散时间下的建模, 用 过程 来计算连续时间下的建模

通过使用 微马尔科夫



当 逗留时间 是马尔科夫的 即 memoryless 则有全马尔科夫

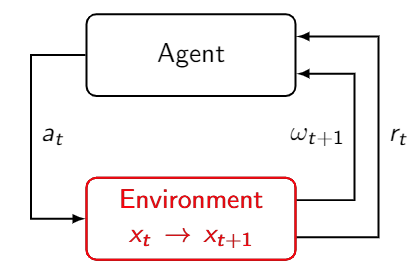
The minimum of exponentials 最小指数:

Lamda是 出生率, 进入该状态的概率 miu是死亡率 退出该状态的概率

强化学习

从 环境 中的 经验 学习智能体, 学习去做一系列的决策, to achieve goals



两个chanllenge: 1. reality gap 不可能完全真实环境 只能模拟 所以 模拟器要尽可能精确 2. limited data: 因为agent本身的限制, 不可能获取所有信息 所以 要提升generalization

在RL中, generalization 泛化指: 1. 在受限数据环境中的有好性能的能力 2. 在相关环境中获得好性能的能力. 特殊的 转移学习技巧

监督学习在有限数据下会bias 和 overfitting