1.强化学习的最终目的是得到最优策略,最佳状态值,分析工具为贝尔曼公式

1.π(a|s)=1;描述s状态下执行a这个动作的概率,π也叫策略,还有一种概率是p(s|a,s),显然这个具有历史无关特性,意思是我采取什么行动为到达什么位置,因为到达的位置不一定由行动百分百决定,可能你向上的action但是有0.1的概率向下

2.agent执行一次action时会给他一个reward,一般正数代表鼓励,负数代表惩罚

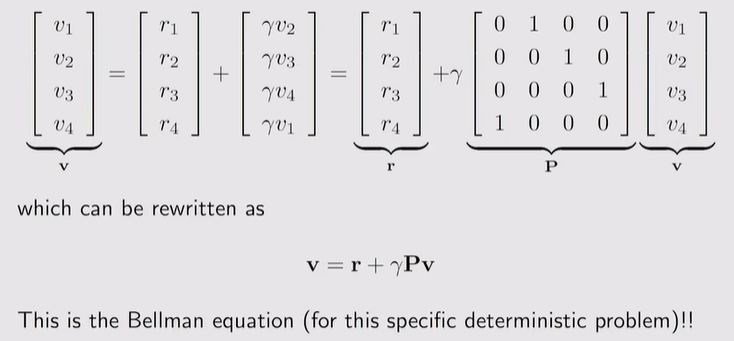
3.trajectory和path区别是他包含了r,a,s是一个链,对于一个trajectory会有一个return是把所有reward加起来

4.discount rate(0到1)使用得到discounted return,第n次得到的reward乘以.discount rate的n次方, .discount rate趋于0的时得到的return被最开始的reward影响更大,趋向1时被未来的reward影响更大,避免了到达终点后反复reward得到无穷的return

5.episode:有限步的trajectory,episode tasks一般可以转换成continue tasks,需要对target state进行设置

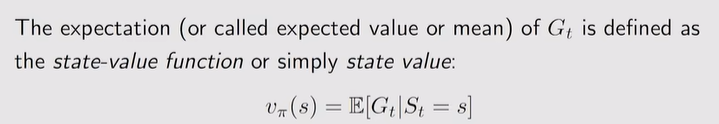
6:MDP:马尔科夫链,即上面提到的trajectory

7.一个状态的reward其实是依赖于其他状态的reward:如:vi=ri+yvi+1,表面上上是自己等于未来但是其实列成矩阵形式时是可以有解的,这就是简单情况(vi=ri+yvi+1)下的贝尔曼公式

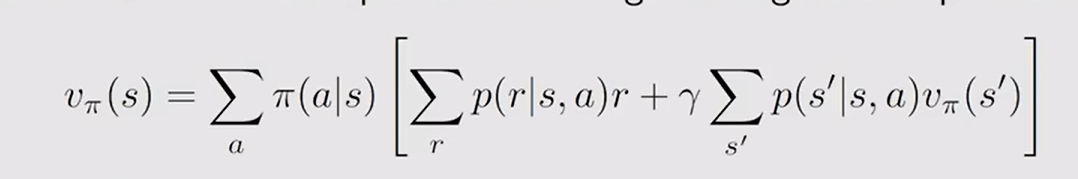


引出来两个思想: 一个状态的reward其实是依赖于其他状态的;求解问题可以用众多未知的关系式利用矩阵求解

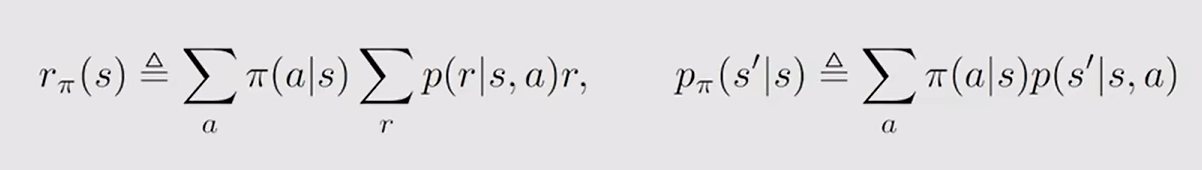
8.对于多种不确定的情况,将s,a,reward,return均变为随机变量,因此有了数学期望的概念,vπ(s)就是return这个随机变量的期望,即多条trajectory的评价return.称为state value



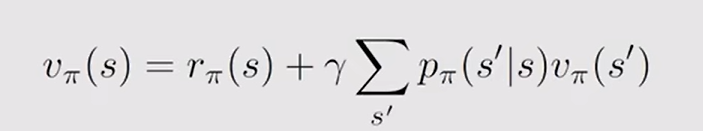
9.贝尔曼公式:

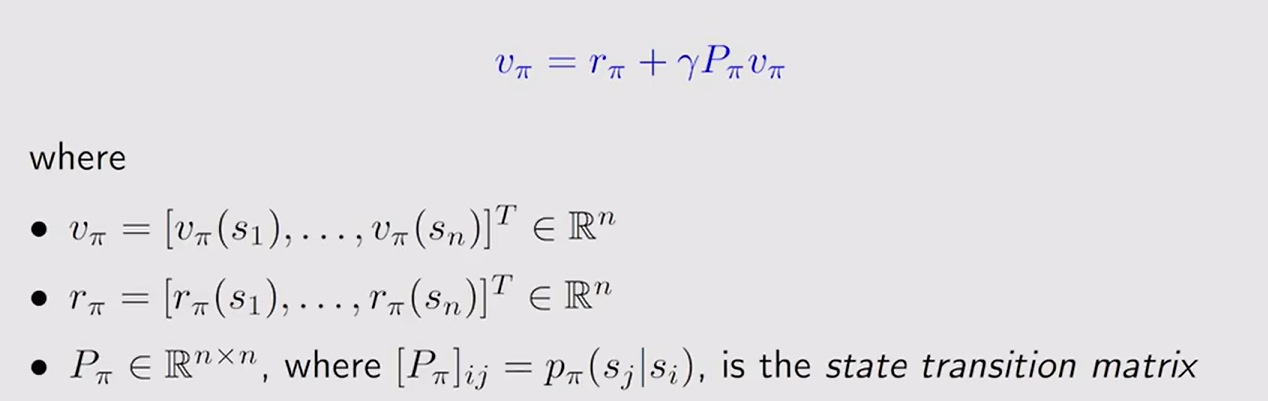


为了构建矩阵求解显然这个公式太长了,可以进行合并:

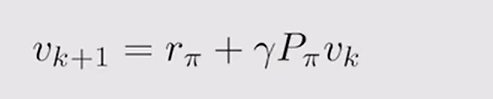


从而得到:



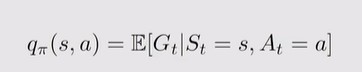


以上求逆难度较大,因此有递推式:

得到vk表达式后k趋于无穷就是vπ

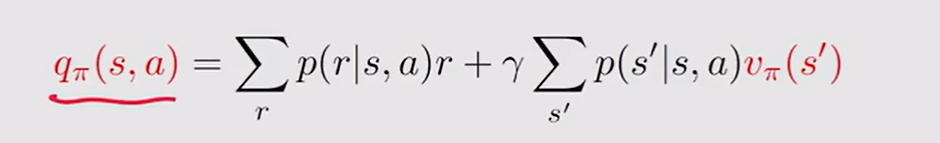
10.action value:区别与state value的policy的不固定,他的考虑了在state下会进行的确定的可能,因此state value就是无数个action value乘以发生这个action value时的π再累加.

Action value:



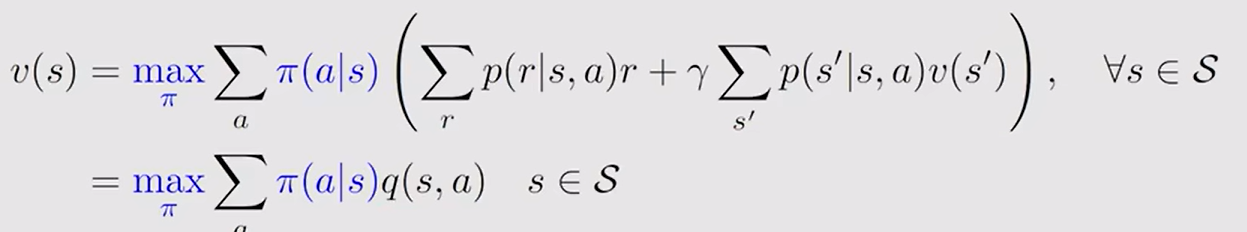


除了顺推得到action value还能借助state value逆推得到:



11.利用action value更新policy的概率,但是只是确保了当前action value最大不能确保policy最优,在过程中可能存在着无数不优的action组合

12.贝尔曼最优公式:

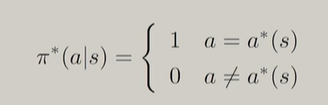


引出问题:是否有解,解是否唯一,解是否能保证最优,怎么求解,一系列证明自行去看,最后的解法是策略使得action value最大的为100%,求解是用迭代法(这一块还不太懂以后有机会再看看)

解释:假设所有action均选择最大的,则原先两个变量的贝尔曼最优公式变成了贝尔曼最优方程,v=f(v),可以证明这个方程符合压缩映射的定理存在唯一不动点,即唯一解,而所有action均选择最大的则表明了这是最优解.因此只要使用压缩映射求解不动点的方法即可得到所有value.然后反求出所有策略

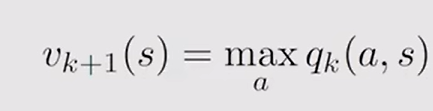
后续补充如何反推.

13.值迭代::k代表迭代次数,每一次对应一组action的策略,计算得到初始的vk,就可以得到初始的qk,然后选择qk最大的为下一次迭代的v,即vk+1,顺带也有了πk+1的策略.因此说是反推策略其实是选择q最大的就是策略.



上面的笔记则是为了证明为什么可以迭代,已经为什么迭代无穷后就是最优解.当vk-vk-1是非常小的值是即找到不动点就可以停在迭代.

最后体现出来就结果其实非常简单:

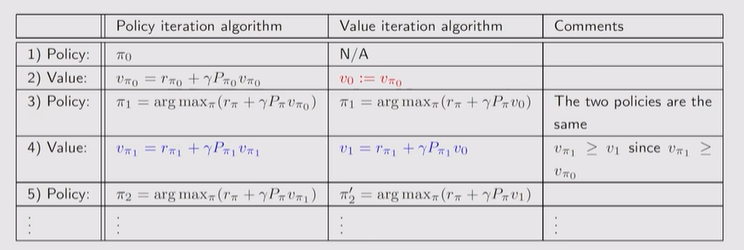


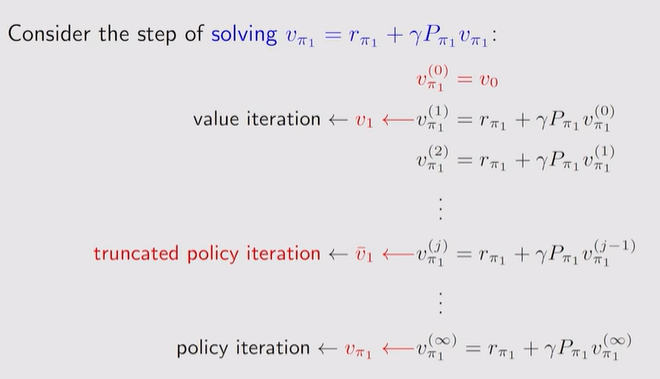
14.值迭代是从值出发一开始是没策略的,初始v0是任意的

多个迭代:贝尔曼公式的求解迭代:不用逆,给定初值v0迭代出这个v0的实际值

策略迭代:只存在理论,实际不可能存在,因为需要迭代无穷次,实际上为截断策略迭代,误差足够小就停

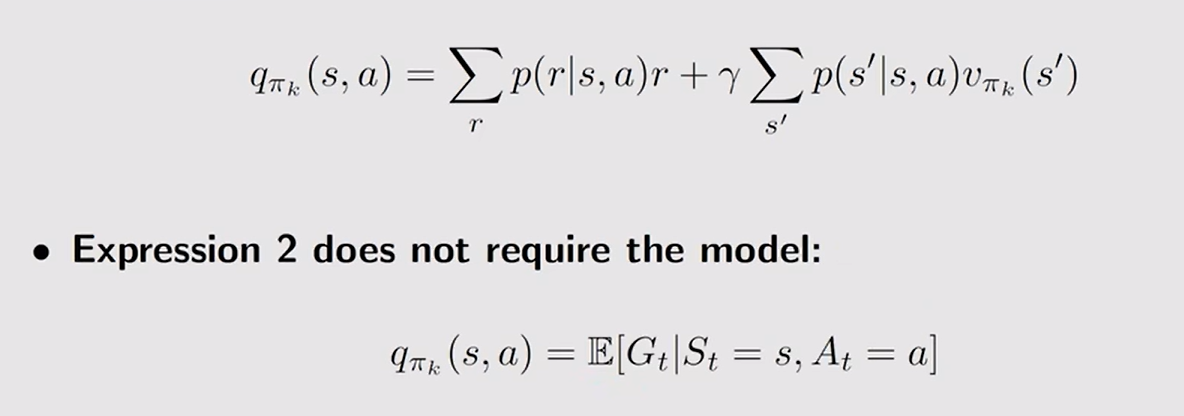
贴两个总体概述防止思路跑偏:



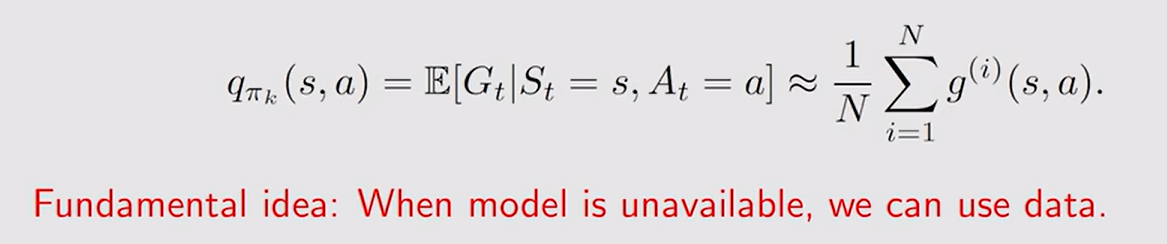


15.12-14的内容有误,详细的去看csdn上的笔记

16.值迭代和策略迭代都需要知道q的公式,按照上面的求解q是需要知道model的,但是其实还有不用model的方法:



他可以使用期望求,而期望可以用大数定理,蒙特卡洛方法代替:



其中g是一组(s,a)的episode求得的reward即return,q则是期望,q里面是任意的s和a,所以至少要有s的数量\*a的数量的组数,后面的步骤和策略迭代一样了

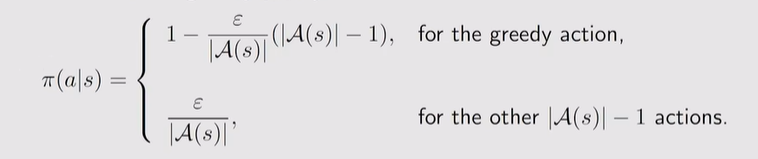
他有关键是episode的长度,如果epsisode长度没有照顾到一些s,a对,那么这些s,a对就无法得到纠正.体现到图了就是假如episode只有一步,那只有在终点附近的可以得到优化,因为其他的一步之内的范围没有正的reward.

因为直接走了定义,所以他实际上就是reward累加,加上γ值约束,

17.以上是MC basic,显然需要求很多reward效率有点低,一次episode其实可以充分利用.

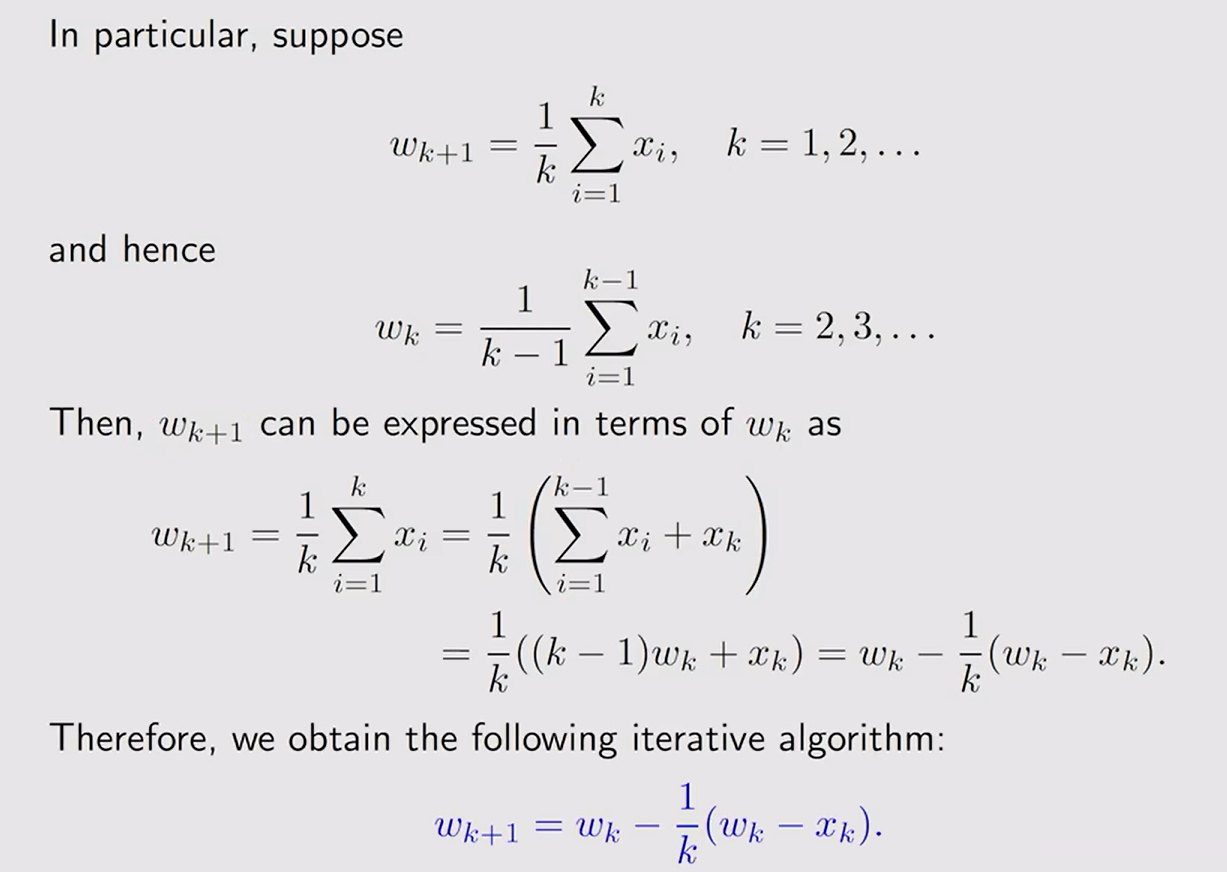
方法1:每次出现需要的s,a对都利用他后面的return,方法2:只利用第一次出现的.他要实现必须满足explore starts,探索完全.,要从每个s,a出发防止漏掉.

18.因此引入soft policy,每个a都有可能,那么只要episode足够长哪就可能访问到所有的s,a对,这样只需要从几个或者一个出发即可,另外episode太长会采用every visit的方法即每次出现需要的s,a对都利用他后面的return,



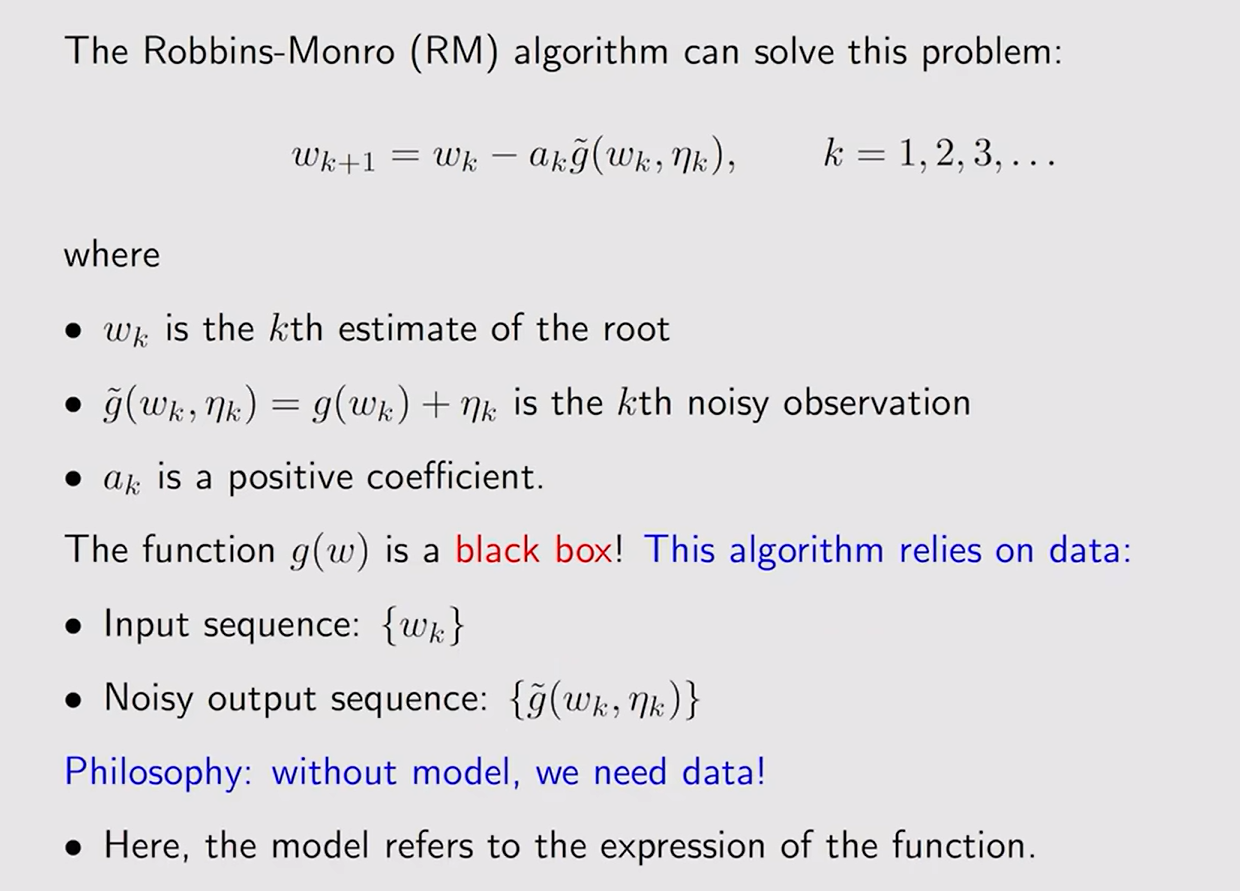
伊普西龙越大探索性越好,但是得到的episode可能不好,导致q下降,一般一开始大后面小

19.求期望时,因为他需要等数据,再去求平均效率过低,通常采用增量式的,来多少处理多少:



20:SA:随机近似,指的是一系列随机迭代算法,求解方程g(w)=0不需要得到这个方程,在这个领域有一个RM算法,而随机梯度下降是RM算法的特殊情况

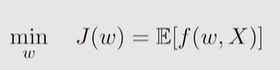
方法:k表示第k次估计,把gw看成黑盒,只需要观测他的输出,其中w为输入,η为噪音,思路其实很简单,输出大于0就减去正值,小于0减去负值



当然了使用他是有条件的,至少要确保有解,ak的和要等于无穷,但是平方要小于无穷,即要收敛但是别收敛太快,噪声期望为0,且方差有界

21.ak=1/k可以成立,同时这个形式类似于之前求期望,其实也通过证明发现他就是一个特殊的RM问题

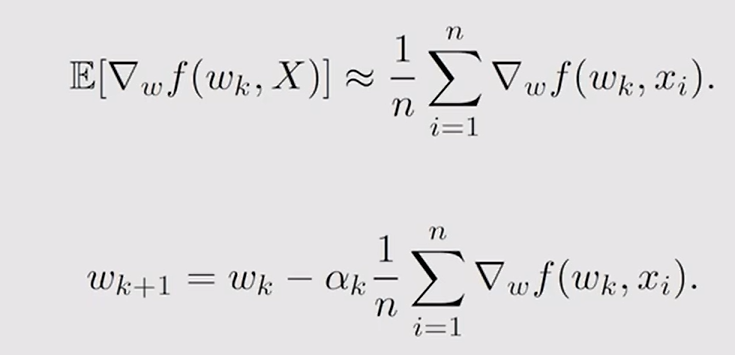
22.SGD:随机梯度下降算法,目标是为了解决这个优化问题:



有方法：GD：他同样有个问题就是期望很难求



BGD：批量梯度下降，用数据代替期望



SGD：BGD的采样要求还是太大，直接令n=1：



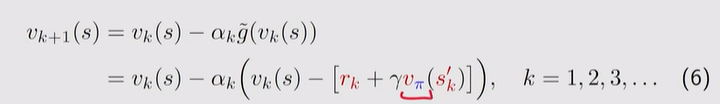
显然这样去代替期望极度的不准确，但是不妨碍他能收敛因为他是特殊的RM算法

23.MBGD：不全用采样数据，随机抽取m个，SGD是抽取1个，MBGDm=n也不是BGD因为他是随机抽取这些数据

24.TD算法：蒙特卡洛求q其实也是在求近似的v，而TD算法借用了上面的SGD的思想（没有用到梯度）能够处理黑盒函数的功能来求同样未知的v，因此是求这样一个问题：



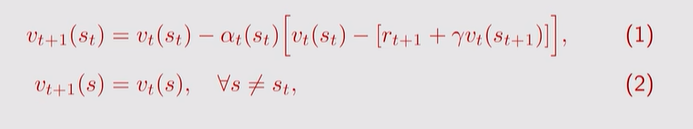
可以得到：



式中vπs’k同样未知,因此也要去估计这个值变成vks’k

这个算法比蒙特卡洛好的点时他是用当前的采样不断去优化逼近实际的value,他不需要等episode采样完.

总结:

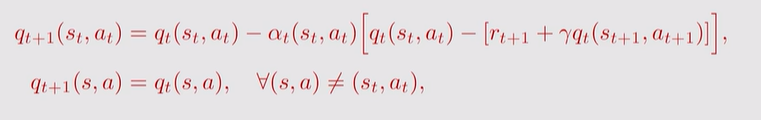


使用t时刻访问到的东西区更新value,其他状态的value不动,根据之前学到的随机近似理论,中括号里面就是要去逼近的值,因为是逼近value所以本来应该是r+γv(即reward)的期望的.然后随机近似允许只使用一次采样代表他就变成了图中的形式,但是注意到即使是这样中括号里的也应该是vπ,他是未知的,因此把他也当成估计量变成vt

显然很累TD是否成功有个error即:



25.基于TD算法还有许多变种使得他更具实用性,如sarsa可以估计q



按照前面的思路中括号里的后半应该是qπ(s,a),

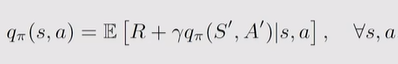
为什么可以呢:这就要回到最初的定义了:

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

其实q也是return的期望只是条件不同这同时还带了一个特别重要的return写法:

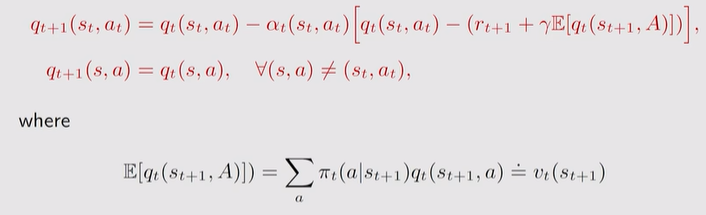


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!



26.使用TD算法时的策略迭代还是会用ε-greedy的,因为他和蒙特卡洛一样是基于采样的,他会有很多episode,只是蒙特卡洛直接得到value,TD一边采样episode一边计算value.特殊的时为了提高效率他会直接把误差很大的value拿去更新策略,然后这个策略再进行估计

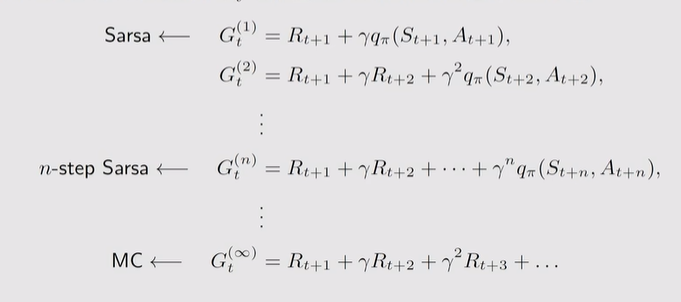
27.expected sarsa:



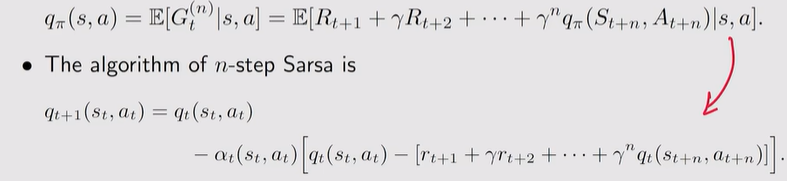
求解问题为:



N-step sarsa:

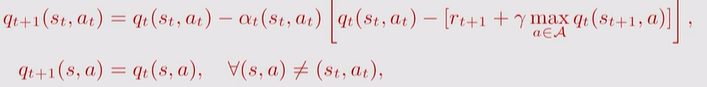


而这个就是n-step:



28.n-step sarsa算法是特殊的online他虽然要等到n但是episode但是等的少,不是offline也不是online

29.Q-Learning:直接估计出最优的v,

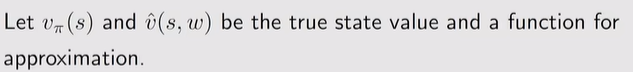


看中括号里的target,他是最优贝尔曼公式,target中的a可以是on-policy的也可以是off-policy,如果选择的π是ε-greedy,即有探索性的贪婪,他会把探索到的东西去更新下一个s状态是什么,而如果是greedy的直接选用q最大的作为下一个π,那么这个π就不会反过来影响探索的策略.

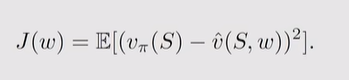
30.on-policy: ,behavior policy和target policy是同一个.环境交互得到的策略实时(behavior policy)去改进目标策略(target policy),也就是改进也环境的交互的策略,即不断变更探索方法

Off-policy: behavior policy和target policy不是同一个.环境交互得到的策略(behavior policy)去改进目标策略(target policy),但是却不影响现在的交互策略,因此可以目标策略越来越好但是探索策略却保持不变

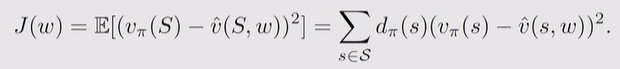
31.(s,a,q)的存储使用表格需要占用空间太大,因此采用函数近似:



为了找到拟合度最好的w需要进行下面的优化问题:



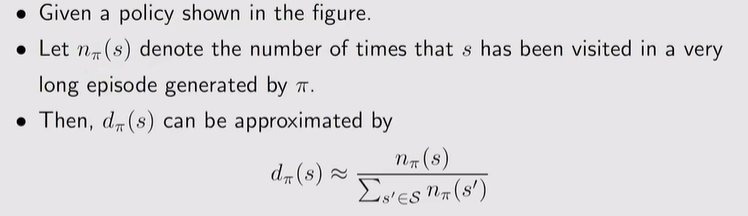
为了求解这个首先得知道S这个随机变量的概率密度函数,通常有认为每个state都是均等概率的,这样好计算但是不符合实际,因此就会映入认为state分别概率符合dπ(s)



这里的dπ(s)不仅代表了概率密度也代表着权重,即权重越大,拟合的精度越高.

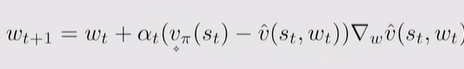
dπ是什么呢?是平稳状态概率:

如,跑的step越多越稳定

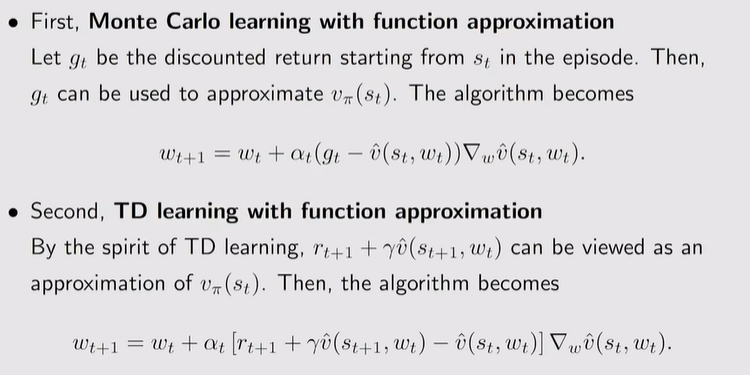


回到求解问题,通常采用随机梯度下降:



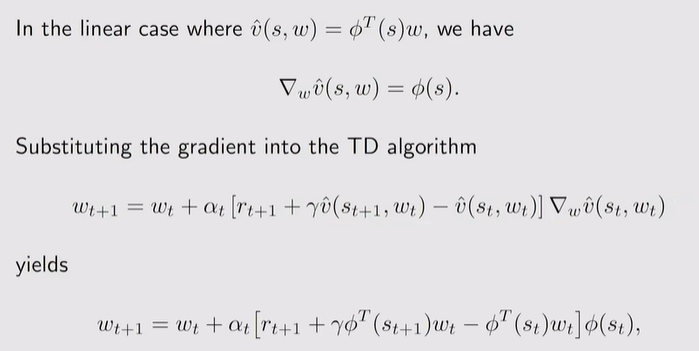


对于未知的v,按照老样子有两种方法,蒙特卡洛和TD:



32.上面解决了拟合精度问题,还剩下用什么拟合的问题,通常有linear function和神经网络两种

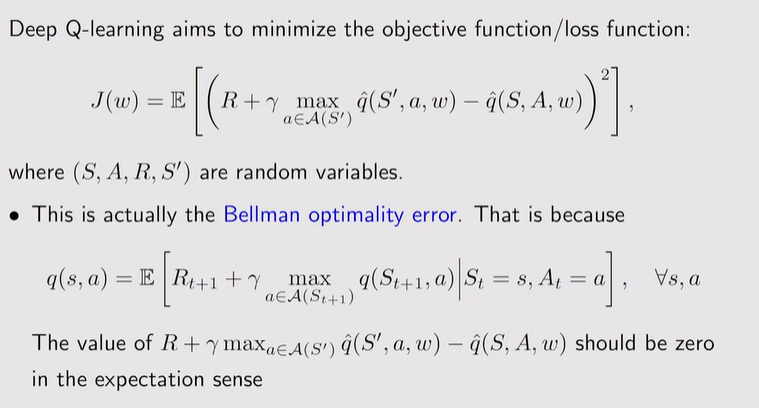
线性的:



他的特征向量如何选取是痛点,所以逐渐被神经网络取代,但是他相比于神经网络还是有着可解释性的优点

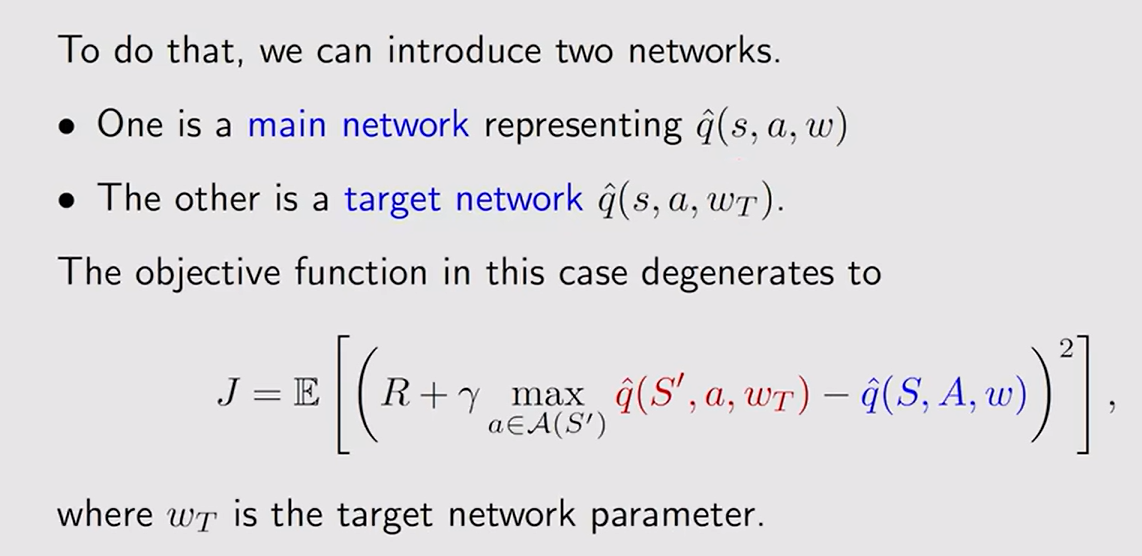
33.DQN:最早且最成功把神经网络引入Q-learning的.

要训练神经网络就需要objection function/loss function:

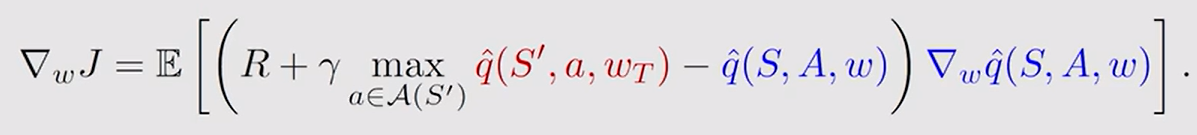


实际上就是TD error.

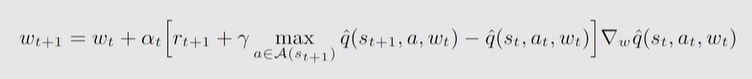
求解这个function首先想到的就是梯度下降,但是里面有w的函数过于复杂,因此就想固定一个求解一次回代再求解已到达逼近,这也算是DQN的创新点/技巧:



因此梯度变为:



34.第二个技巧:经验回放:experience replay,因为采样的时候不是按照约定好的dπ来采样但是用的时候又要求符合这个分布因此会把采样的放buffer里按照目标发布去采样称为经验回放:



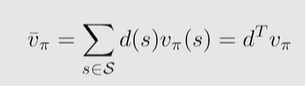
又涉及到迭代又涉及到梯度下降又涉及的贝尔曼最优,卧槽这公式真复杂吧

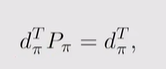
35.policy-based:首先也是要把表格变成函数,因此π变为带参数(向量)的函数:

,同样的他不是确定的θ,黑盒状态输入s得到π:同样的会有优化目标为:

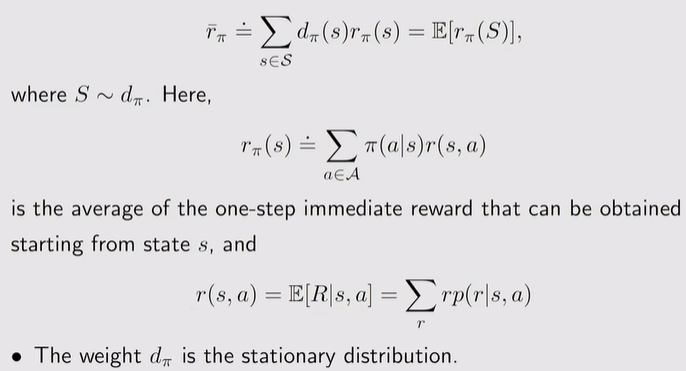


Value-based问题时目标是求出最优的value,即最大,那policy-based自然也需要最优化的目标,因此如metric指标为:

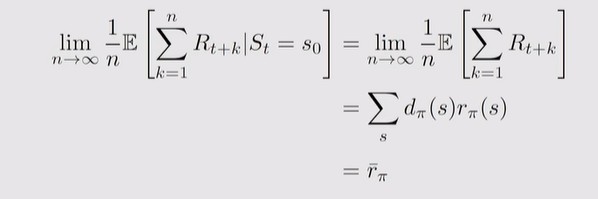
d的选择有两种如果与π无关则称为d0,另外一种与π有关则称为dπ:

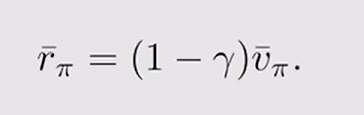


第二个metric指标是平均reward:



他还有另外一种形式:



而policy-based的目标就是优化Θ找到最优的这两个指标，且在数学上可以证明：，一个极值另外一个也极值。

36.我懵逼了，各种数学推导技巧，反正最后就是要用梯度上升求解那个指标最大就是了。。。

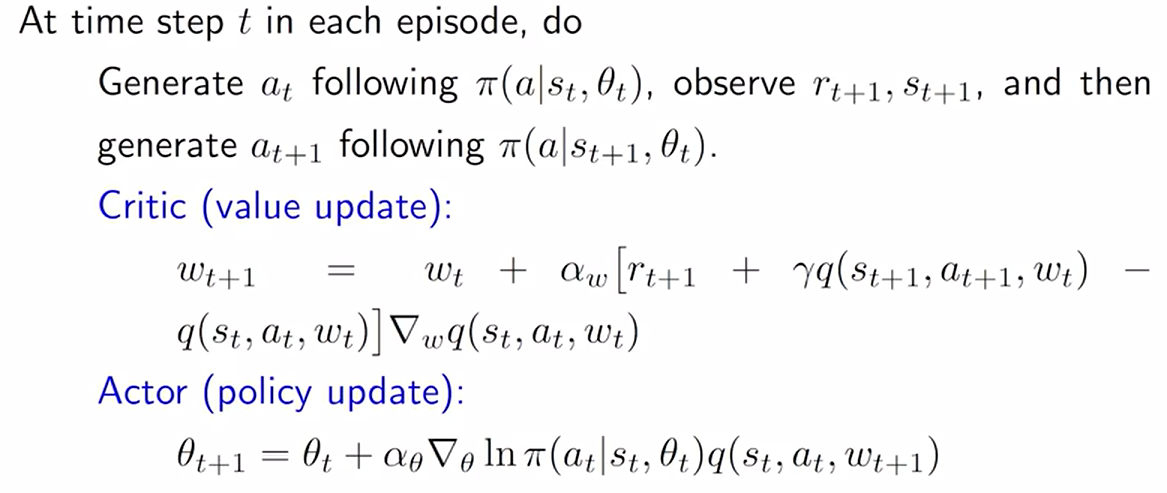
37.actor-critic：policy-based+value function

之前学习可以总结到policy-based是为了求解下面这个问题：（当然了这时这个问题的一种形式而已，选用指标不同，选用数学技巧不同都会有差别）

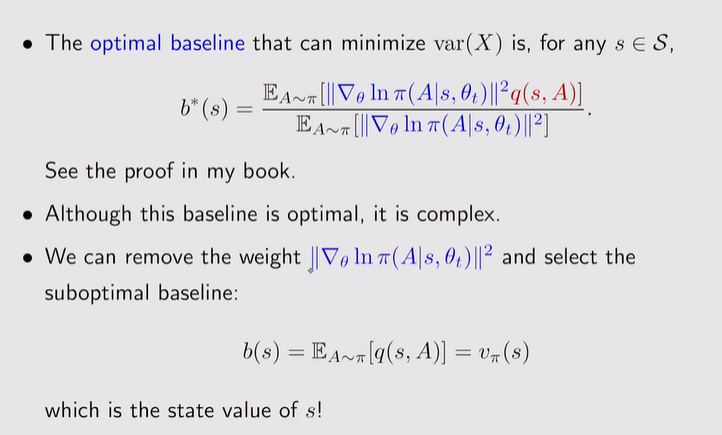


这里面q原本是qπ的,但是我懒得重新截图就用了QAC的图,qπ情况下这个式子不能求解,需要得到q,如果采用蒙特卡洛的方法那就是reinforce算法

引入wt的TD去估计q则变成了QAC算法，如下：

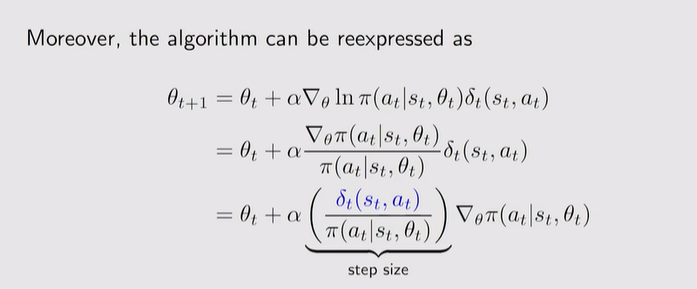


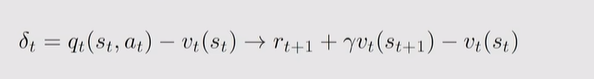
38.期望引入标量不会发生变化因此可以尝试引入baseline偏置,为什么要引入呢,因为虽然期望不影响但是方差会被影响,方差之所以小好是因为越小说明采样越接近真实值.一般选择如下:



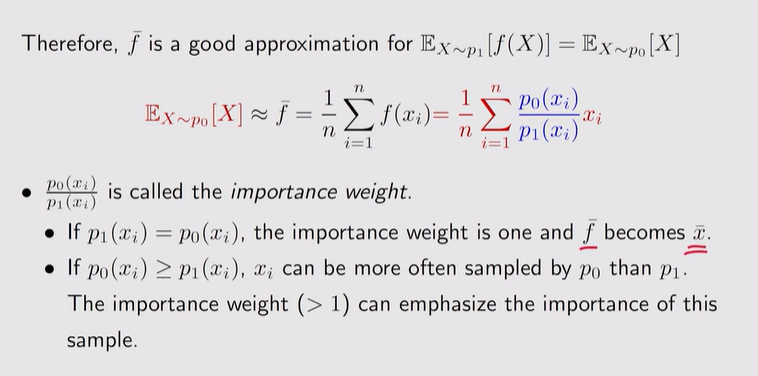
39.A2C算法:





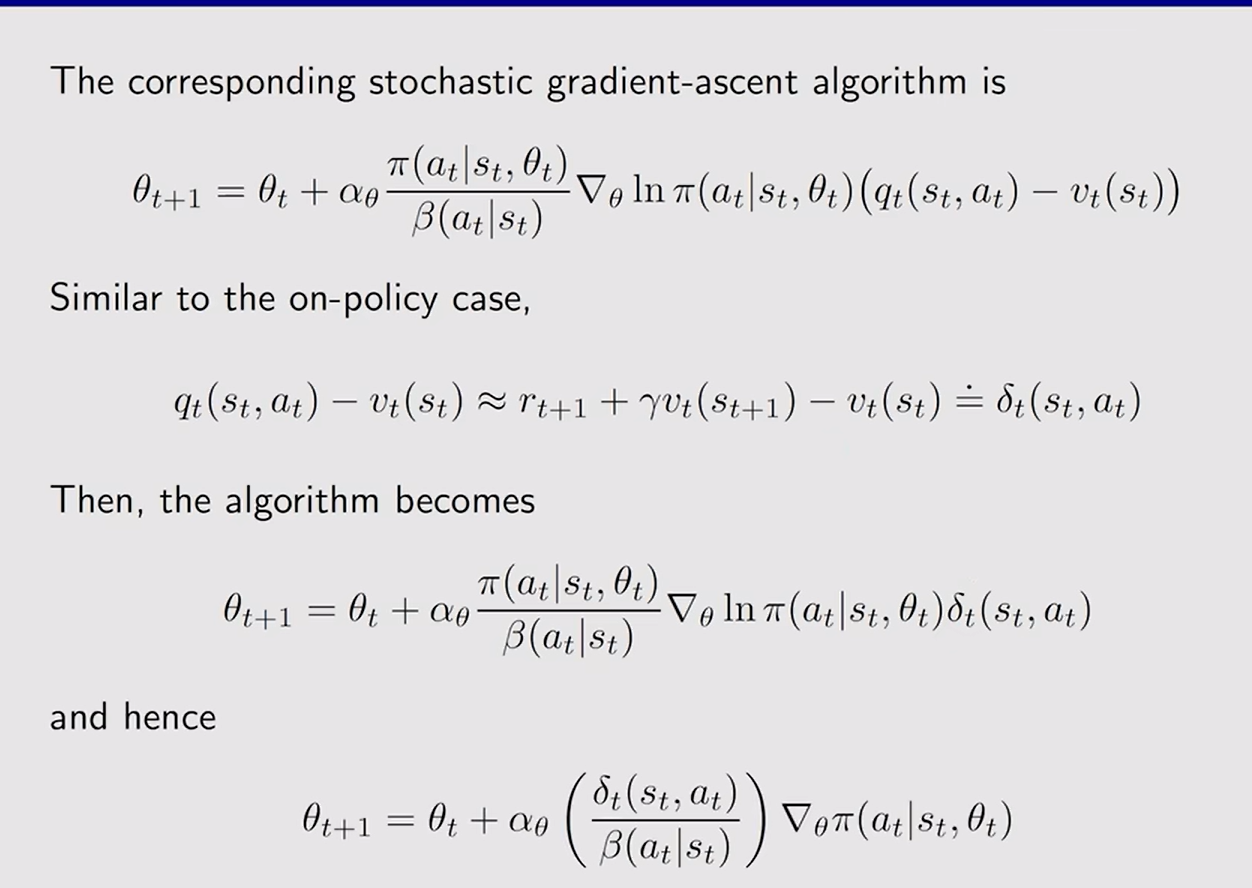


现在的q,v都是用神经网络求了,改写成这样子就只用一个神经网络

40.重要性采样:由于EX的分布未知(为神经网络),而我们的采样时的分布是已知的且一定会与未知的产生偏差,为了修正这种偏差有了重要性采样的出现: 

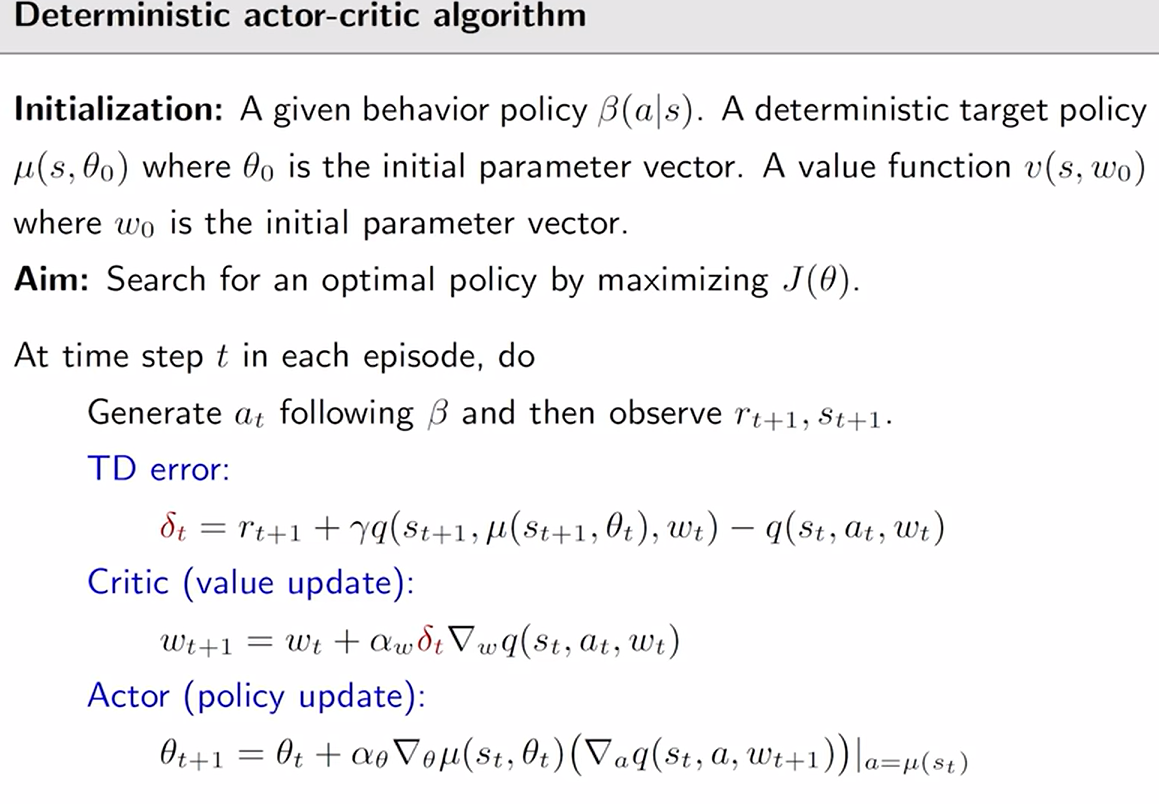
可以看到如果采样时概率很小,代表他很难采样到,他的重要性就会提高,如果实际的概率很小那么他的重要性就会降低

41.应用到算法中为:



β是p1,π是p0

41.π,a连续无穷:



用μ代替会失去概率特性导致确定性,失去探索性,因此通常会采用μ+noise