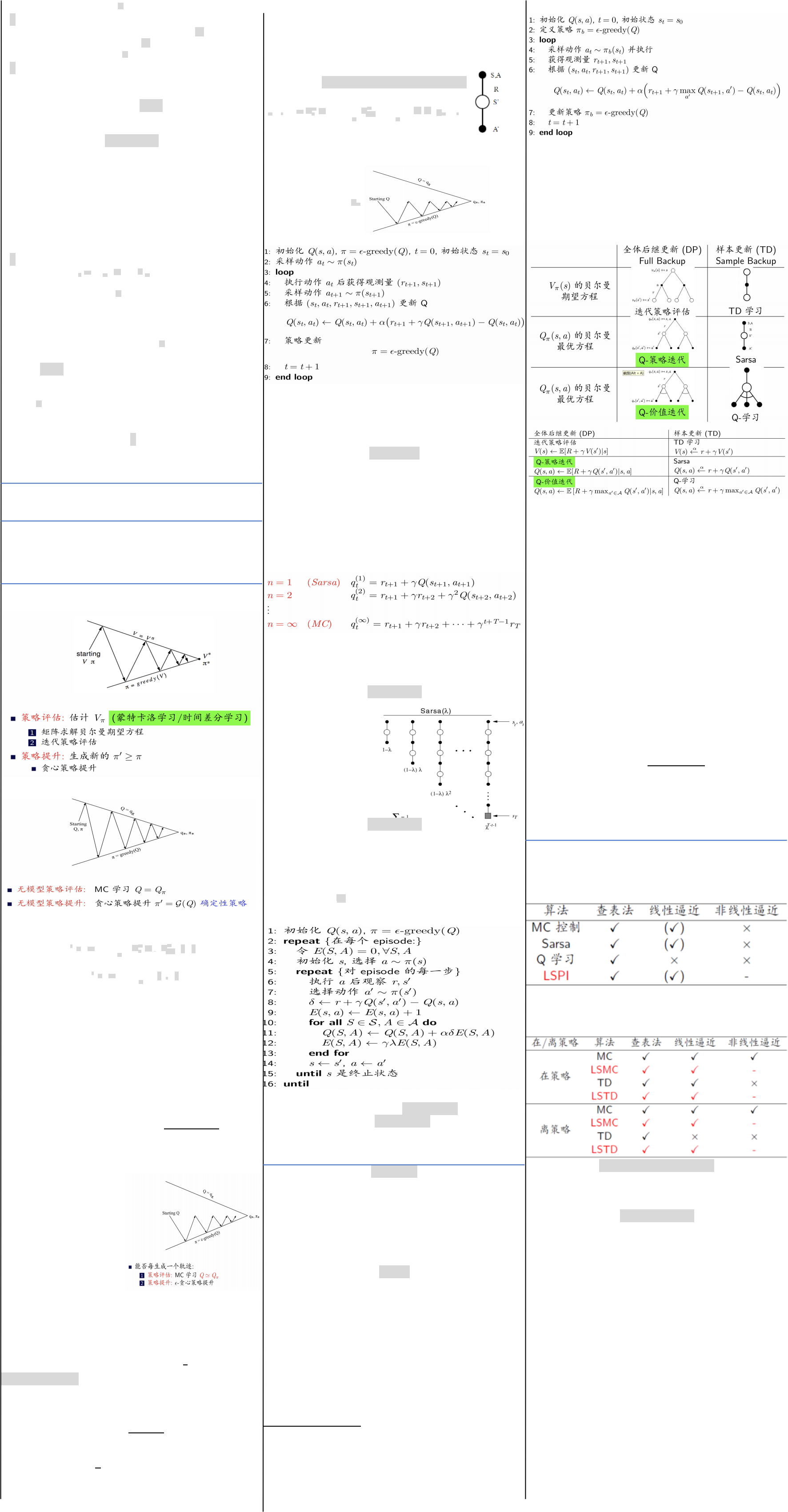
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **贝尔曼最优方程**：  **动态规划方法**  **强化学习：**强化学习是一种优化智能体在环境中行为的一种方法。 根据环境反馈的奖励，调整智能体的行为策略，提升智能体实现目 标的能力。  〇强化学习考虑的是**序贯决策过程**：  智能体处在特定的环境中产生一系列的动作，而环境能够根 据这些动作改变智能体的当前状态。  举例  遥控直升飞机的特技表演；打败围棋世界冠军；管理股票证券；发 电厂调控；控制人型机器人双足行走；视频游戏上超越人类  目标：选择一组动作使未来奖励和最大化  动作可能在未来很久才会产生影响；奖励可能是延时的；有可能会 牺牲短期利益从而获得长期的回报  举例：  投资（要几个月才会收益）；直升机加油（防止几小时后没油坠机） **强化学习：**  ~~v\*(s)~~ = ~~[~~T~~(v\*)](s)~~ = ~~mx (~~R ~~+ Y Σ~~sI∈sP~~s/v\*(s~~9~~))~~  v\*(s) = mx (R + Y ΣsI∈sP/v\*(s9))  价值迭代收敛性，Y  < 1(此时价值迭代算子是收缩算子)  **动作规划:** 价值迭代/策略迭代  使用∞-范数比较两个价值函数的不同, 即状态集上两个函数最大  通过迭代有效求解原始问题中非线性的 max 算子 依赖系统模型, R, P  的差值Ⅱu 一 vⅡ∞ = Iu(s) 一 v(s)I  足够的计算空间记录每个状态的函数值  收缩算子：对于任意两个函数f, g, 如果两个函数的距离经过一个 算子T后能够被缩小, 那么这个算子被称为收缩算子  价值迭代v\*(s)  策略迭代vπ(s), π(s)  ⅡT(f 一 g)Ⅱ∞ < Ⅱf 一 gⅡ∞  **状态-动作价值函数**  q\* s, a) = R + Y ΣsI / q\* s9, a9)  ∈s  ●基于 Q 函数的价值迭代  初始化q1 , (e.g. q1(s; a) = 0; 丫s ∈ S; 丫a ∈ A) 迭代计算新的 Q 函数  q s, a) = R + Y ΣsI Ps/  max qk s9, a9)   |  |  | | --- | --- | | ①产生的结果(动作) 能够改变数据的分布(状态)  ②最终的目标可能要很长时间才能观察到(下棋)  ③没有明确的标签(label) 数据  ④根据当前的奖励，最终实现长远的目标  **监督学习(Supervised Learning, SL)/非监督学习：**  ①产生的结果(输出) 不会改变数据的分布②结果是瞬时的③要么 有明确的标签数据(SL)④要么完全没有任何标签数据(USL) | | | **●马尔可夫性 Markov property**  智能体未来的状态只与当前时刻的状态st 有关, 而与过去的 状态{s1, … , st-1}无关，那么称智能体的模型具有马尔可夫性。  P[st+1Ist] = P[st+1 Is1, … , st]  未来只与当前有关，与历史无关s1:t → st → st+1:∞  一旦当前状态确定了，历史状态都可以丢弃了，也就是说当前状态 足以决定未来状态是什么样的 | | | 强化学习主要研究的是具有马尔可夫性的问题. | | | **状态转移矩阵：**  对于一个马尔可夫状态s和后继状态sI，状态转移概率定义为  PssI = P(st+1 = sI Ist = s)  状态转移矩阵P 定义从所有状态s到所有后继状态sI 的转移概率 to | | | P = from [氵1 | …、 氵n丨其中矩阵的每行和为 1 | | **●马尔可夫过程 Markov Process**  一个马尔可夫过程是一个无记忆的随机过程，即一组具有马尔可 夫性的随机状态序列 S1; S2; … | | | 定义：  马尔可夫过程（马尔可夫链）可以用一组〈s, P〉表示 s是（有限）状态集  P是状态转移概率矩阵P = P = sI s = s) | | | **●马尔可夫奖励过程 Markov Reward Process** | | | 一个马尔可夫奖励过程是一个马尔可夫链加上奖励 | | | 定义：  一个马尔可夫奖励过程由一组〈s, P, R, Y〉构成 s是一组有限状态集  P是状态转移概率矩阵PssI = P t+1 = sI st = s) R是奖励函数，Rs = E Rt+1 st = s)  Y是折扣因子，Y ∈ [0,1] | | | Gt = Rt+1 + YRt+2 + … = Σ=OYkRt+k+1  折扣因子Y代表未来的奖励在当前时刻贡献的价值  k + 1时刻后的奖励R对当前回报的贡献只有YkR 0=目光短浅；1=目光长远  如果想要调整奖励的重要性，数学上很方便，在循环马尔可夫 过程问题中能避免无穷回报。但是有可能会忽视未来奖励。如果奖 励代表金钱，近期的奖励会比远期产生更多的收益。自然界的人类 或动物行为模式更倾向于近期奖励。有时候也会使用无折扣的马 尔可夫奖励过程（即 1）。例如所有的事件序列都有终止状态  **价值函数**v(s) **代表智能体在状态**s **下的长期价值**  一个马尔可夫奖励过程的状态价值函数等于从状态s出发的期望 回报  v(s) = E[Gt Ist = s]  **MRPs 的贝尔曼方程 Bellman equation**  价 值 函 数拆 分 成两 部 分 :瞬 间 奖励Rt+1 ; 后 继 状 态 的 折扣价 值 Yv(st+1)  v(s) = E[Gt Ist = s] = [Rt+1 + YRt+2 + …Ist = s] =  E[Rt+1 + Y(Rt+2 + … )Ist = s] = E[Rt+1 + YGt+1Ist = s] = E Rt+1 + Yv(t+1) st = s]  v(s) = E[Rt+1 + Yv(st+1)Ist = s]  v(s) = Rs + Y Σs/ ∈sPssIv(sI) 矩阵：v = R + YPv  [(())丨 = [(())丨 + Y [氵1 …、 氵n丨 [(())丨  v = (I 一 YP)-1R: n个状态下的计算复杂度 O(n3) | | | **●马尔可夫决策过程 Markov Decision Process, MDP** | | | 马尔可夫决策过程是马尔可夫奖励过程加上决策 . 问题的所有状 态都具有马尔可夫性. | | | 一个马尔可夫决策过程由一组〈s, A, P, R, Y〉构成 | | | s是一组有限状态集 A是有限动作集  P是状态转移概率矩阵  I *’* = P t+1 = sI st = s, At = a) R是奖励函数，*’* = E Rt+1 st = s, At = a) Y是折扣因子，Y ∈ [0,1] | | | ●强化学习的主要组成元素：  **策略**：智能体的行为  **价值函数**（值函数、性能指标函数）：智能体在某一状态 | | | 和/或某一动作时是好还是坏 | | | **模型**：智能体对真实环境的估计 | | | **策略：代表了智能体是如何行为的，是从状态到动作的映射，是状** | | | **态到动作的一种分布** | | | 确定性策略：a = π(s)  随机性策略：π aIs) = P(At = aIst = s)  一个策略定义了一个智能体的行为，MDP 问题的策略取决于当前  时刻的状态(与历史无关)，即策略是静态的(时不变性)，迷宫中的  箭头（上下左右）代表策略π(s)在状态s的动作 给定一个 MDP 的M =〈s, A, P, R, Y〉和策略π:  状态序列s1, s2, … 是一个马尔可夫过程〈s, Pπ〉  状态和奖励序列s1, R1, s2, … 是一个马尔可夫奖励过程〈s, Pπ , Rπ , Y〉 PI = Σa∈A π(aIs) PI R = Σa∈A π(aIs) R | |   k+1  aI  ∈s  ●基于 Q 函数的策略迭代  给定一个策略π1  **策略评估**: 计算 k 的状态-动作价值函数  qπk s, a) = R + Y ΣsI P/ ΣaI∈Aπ(a9 Is9) qπk s9, a9)  ∈s  共同的不足: 对模型依赖, R;P  要么是 MDP 问题的模型已知  要么智能体对环境建模 无模型的策略评估方法：  蒙特卡洛方法；时间差分方法 **策略提升**: 根据 πk提取出新的策略  q  πk+1(s) = arg x , a)  qπk s  **迭代策略评估:**  当  1  2  k → ∞, vk → v\*  问题:对给定策略π计算它的价值函数  **策略**  智能体可根据给定的策略决定在当前状态下如何采取下一步动作 确定性策略：at = π(st)  随机性策略: at~π(st)  给定一个策略, 就可以对它的好坏进行评估  智能体按照给定的策略执行动作，获得的期望回报称为该策略的 价值函数vπ (s) = Eπ [Rt+1 + YRt+2 + … Ist = s, At~π(st)]  价值函数越高的策略, 性能越好  方法:基于贝尔曼期望方程迭代更新价值函数 初始化一个价值函数v1 , v1 → v2 → … → vπ 同步更新：  在每次迭代中，对所有的状态s ∈ s，根据vk (s9)更新vk+1(s)，其中 s9是s的后继状态，最终收敛到真实价值函数vπ  **贝尔曼期望方程**  价值函数也可以写成动态规划的形式  对于最优策略来讲  ~~vπ~~ = ~~Σ~~a~~π(aIs) (~~R ~~+ Y~~P~~s/vπ(s~~9~~))~~  vk+1(s) = Σa∈Aπ(aIs) (R + Y ΣsI∈sP/vk(s9))  π\*(s) = arg x ~~.~~   ∈si i)  它的价值函数就是最优价值函数, 满足贝尔曼最优方程  -  .   ∈si i) 相比于其它策略，最优策略具有最大的价值函数  在给定策略π的情况下, 智能体有限个状态之间的转移可以用矩阵 形式表示  其中 π 代表了在策略π下, 从状态i转移到 j 的概率 **贝尔曼期望方程:**vπ = (I 一YPπ)-1Rπ**:矩阵很大,矩阵稀疏**  (R + Y ΣsI Ps/v\*(s9)  贝尔曼期望更新算子Tπ , Tπ (v) = Rπ + YPπv  t  +1  ssI  t  该算子是Y-收缩的, 即经过该算子两个价值函数的距离变为原来 的Y倍  (R + Y ΣsI Ps/v\*(s9)  v\*(s) = max  a  v\*(s) = vπ\*(s) ≥ vπ(s), 丫sEs  Pπ  …、 ,,丨  s1,s1 氵  所以迭代策略评估收敛到 π  v  Pπ =  **回报 Return:回报**  t**时刻往后所有的折扣奖励:**  t**代表从**  G  ●异步动态规划:就地动态规划;优先动态规划;实时动态规划  Pπ  sn,s1  **蒙特卡洛方法 Monte-Carlo, MC**  s  s  P  si,sj  MC 方法直接从经历过的事件中学习  **无模型方法:** 不需要 MDP 的转移/奖励函数  Σaπ(aIs1) R1    vπ = [((s))丨 , Rπ =  如何提升策略—————————— 给定一个策略π计算π的价值函数  **从完整的事件中学习:** 没有自举      **基于最简单的思想:** 价值= 平均回报 运行 MC 方法通常要求:  …  Σaπ(a sn) Rsn  a」  s  所有事件都到达终止状态{s1, A1, … , sterminal}  或者事件的时序足够长{s1, A1, … , sT}; T > 1 蒙特卡洛策略评估  vπ (s) = Eπ [Rt+1 + YRt+2 + … Ist = s, At~π(st)] vπ提取出贪心策略  从  v  目标: 从策略π产生的事件中学习 π：s1, A1, R2, … , sk~π 回忆下回报的定义是所有折扣奖励和  πI = greedy(vπ)  πI(s) = arg  ( )  max  R + Y Σ  P/vπ (s9)  Gt = Rt+1 + YRt+2 + … YT-1Rt  a∈A  回忆下价值函数的定义是回报的期望vπ (s) = Eπ [Gt Ist = s]  sI∈s  新的策略继续上述过程,经过多次策略迭代后, 策略收敛到  **策略迭代**  考虑一个确定性的策略，a = π(s) 根据贪心策略对策略进行提升  πI(s) = arg x /  ) = arg x )  上述过程提升了任意 s 一步的价值  qπ(s, πI(s)) = x qπ(s, a) ≥ qπ(s, π(s)) = vπ(s)  因此新策略的价值函数得到了提升, vπI(s) ≥ vπ(s)  因为vπ1 ≤ vπ2 ≤ … ≤ vπk ≤ v\***单调递增有上界，所以收敛** 当策略不再变化时满足贝尔曼最优方程  价值迭代  ~~vk+1(s)~~ = ~~mx(~~R ~~+ Y~~P~~s/vk(s~~9~~))~~  只在收敛得到v\*后计算π\* , 中间过程不产生策略 涉及赋值操作, 计算量小,0(IsI2 IAI)，迭代次数多 策略迭代  π  \*  **蒙特卡洛策略评估方法使用回报的经验均值作为回报的期望 基于样本更新：**  接下来的课程将主要使用智能体从环境获得的样本进行更新 ，使 用样本的奖励和状态转移 , s9〉而不是奖励函数 R 和转移函  〈  s, A, , R  数 P 。好处包括:  无模型: 不需要预前知道 MDP 的模型信息  通过样本避免整个状态空间的维数灾问题  每次更新的计算量是固定 ，与后继的状态空间 n = |S| 无关 首次经过的 MC 策略评估  为了评估状态 s，假设在一次事件中第一次经过状态 s 的时刻为 t， 计数加一 N(s) ← N(s) + 1 ，全部回报相加 S(s) ← S(s)+Gt，估计的 价值等于回报均值 V(s) = S(s)/N(s) ，根据大数定理, 当 N(s) → ∞ 时, V(s) →  vπ (s)  每次经过的 MC 策略评估  为了评估状态 s，对一次事件中每一次经过状态 s 的时刻 t，计数 加一 N(s) ← N(s)+1，全部回报相加 S(s) ← S(s) +Gt，估计的价值 等于回报均值 V(s) = S(s)/N(s) ，同样当 N(s) → ∞ 时, V(s) →  vπ (s)  增量计算均值  一个序列x1, x2, … 的均值μ1,μ2, …可以通过增量的方式计算  Σ= 1xj = (xk + Σ=-xj) = (xk + (k 一 1)μk-1)  1  k  μk =  1  (xk 一 μk-1)  = μk-1 + k  增量式的 MC 更新  根据事件s1, A1, R1, … , sT 对V(s)增量式地更新 对状态st , 回报是Gt  N(st) ← N(st) + 1  1  V(st) ← V(st) + N(st) (Gt 一 V(st))  如果环境是动态、不断变化的, 我们更希望是能够跟踪当前不断变  化的均值, 遗忘掉很久之前的事件  (R + Y Σs Ps/vπ(sI)  qπ (s, a  V(st) ← V(st) + α(Gt 一 V(st))  时间差分学习 Temporal Difference (TD) Learning  TD 方法直接从智能体经历的事件中学习  无模型的: 不知道 MDP 问题的转移和奖励函数 可以从非完整的事件中学习, 借助自举法  根据一个猜测值更新另一个猜测值  MC 和 TD  **目标:** 根据智能体在策略π作用下产生的经历在线学习vπ  **增量式的 MC 方法**:调整价值V(st)向真实的回报Gt逼近  V(st) ← V(st) + α(Gt 一 V(st)) 最简单形式的 TD 学习算法: TD(0)  **价值函数 Value Function**  价值函数是对未来奖励的预测，评估智能体在某一状态下是好还 是坏，因而可以用来选择对智能体最有利的动作  ①***状态-价值函数***vπ(s) = Eπ [Gt Ist = s]  ②***动作-价值函数***qπ (s, a) = Eπ [Gt Ist = s, At = a]  qπ (s, a) = Eπ [Rt+1 + Yqπ (st+1, At+1)Ist = s, At = a]  调整价值V(st)向估计的回报  )逼近  R + YV(  t+1  t+1  V(st) ← V(st) + α **TD 目标**：  ) 一 V(st))  R + YV( t+1  **价值迭代 VS 策略迭代**  t+1  R + YV( t+1)  ●  t+1  δt = R + YV( t+1  **TD 误差**： ) 一 V(st)  t+1  TD 学习算法：  ●  ~~vπ(s)~~ = ~~Σ~~a~~π(aIs) (~~R ~~+ Y~~P~~s/vπ(s~~9~~))~~  (R + Y Σs Ps/vπ(sI)  πI(s) = arg x   /  )  每次迭代开始时给定一个π , 结束时产生一个新π9**,**迭代次数少 求解方程, 计算量大, 矩阵求逆0(IsI3), 策略提升,0(IsI2 IAI)  vπ (s) = Σa∈A π(aIs) qπ s, a)  qπ s, a) =  + Y ΣsI∈s .vπ (s9)  ss/  qπ s, a) = R + Y ΣsI∈s Pss/  qπ  π(a9 Is9)  s9, a9)  aI∈A  a Σ  ~~vπ~~ = Σa∈Aπ(aIs) ( + Y ΣsI∈s ~~vπ~~ (s9))  ss/ |
| vπ = (I 一 YPπ)-1Rπ |
| **最优价值函数：**  **最优状态-价值函数：**在所有策略中价值函数最大的 v\*(s) = mπaxvπ (s) |
| **最优动作-价值函数：**所有策略中动作价值函数最大 |
| q\*(s, a) = mπax qπ (s, a)  最优价值函数代表了智能体在该 MDP 问题下最好的性能；如果 得到了最优值函数, 那么 MDP 问题就已经求解了  **最优策略：**π ≥ πIif vπ (s) ≥ vπ/(s), 丫s 定理  对任意马尔可夫决策过程  总是存在一个最优策略π\* 比其它所有策略都不差 π\* ≥ π 丫π  所有最优策略的价值函数都相等, 且等于最优价值 函数, Vπ\* (s) = v\*(s)  所有最优策略的动作-价值函数都相等, 且等于最优 动作-价值函数, qπ\*(s, a) = q\*(s, a)  **寻找最优策略**  一个最优策略可以通过最大化q\*(s, a)来确定  π\*(aIs) = { if a = i(s, a)  对任何 MDP 都存在一个确定性的最优策略 如果q\*(s, a)已知, 即可得到最优策略  ***最优化原理：***  强化学习目标是找到一组时间序列的动作{A0, A1, … }使得智能体  从s0 出发得到的期望累加奖励最大化 **最优价值函数**：  v\*(s0) = E [Ax, (R1 + YR2 + Y2R3 + … )]  最优策略具有如下性质：不论初始状态和初始决策(第一步决策) 如何，以第一步决策所形成的阶段和状态作为初始条件来考虑时， 余下的决策对余下的问题而言也必构成最优策略  v\*(s0) = E [Ax, (R1 + YR2 + Y2R3 + … )]  = E [R1 + Y A2x, (R2 + YR3 + … )]  = max E[R1 + YE[v\*(s1)]]  A0 |
|  |
| v\*(s) = max q\*(s, a)  a  v\* 的 Bellman **最优方程**：  q\* 的： q\* = R + Y Σ  Pa  ss/  s, a)  s9, a9)  q\*  max  sI∈s  aI  关于  q\* s, a) =  + Y ΣsI∈s/v\*(s9)      v\*(s)=mx(+YΣsI∈s/v\*(s9))  关于 |
| 确定性的最优策略：  π\*(s) = arg  q\* s, a)  求解贝尔曼最优方程需要：  1 求解非线性算子 max  2 模型已知  3 足够的计算空间 |
| **动态规划 DP**  通过把原问题分解为相对简单的子问题来求解复杂问题的方法， 适用于：  **最优子结构性质:** 问题的最优解所包含的子问题的解也是最优的 满足最优化原理  最优解划分成子问题  **子问题重叠性质:** 在用递归算法自顶向下对问题进行求解时，每次 产生的子问题并不总是新问题，有些子问题会被重复计算多次  子问题重复出现多次  每个子问题计算一次, 保存起来, 再次需要求解子问题 时直接查找结果  **马尔可夫决策过程满足如下两个属性**：  贝尔曼方程具有递归形式  价值函数可以保存和重复利用 |
| **●价值迭代** |
| 贝尔曼最优方程的难点在于求解的v\* 同时存在于等式左右两边 价值迭代的基本思路:  1 对v\*定义一个估计函数 v  2 将估计函数代入方程右边, 等式左边得到一个新函数 v′  3 vI是对 v\* 更为准确的估计  4 将 v′ 代入右式继续上述过程 |
|  |
| vk+1(s) =  vk+1 = (Ra + YPavk)  **●价值迭代算子**  价值迭代定义一个以函数作为输入的算子 T , 对给定的函数vk计 算新的函数  vk+1(s) = [T(vk)](s) = mx (R + Y ΣsI∈sP/vk(s9))  T为价值迭代算子 |

**MC vs TD 控制**

Q-学习算法实现离策略控制:

MC 和 TD 对比

**TD(**λ**)和 TD(0)**

TD 学习和 MC 方法相比有如下优势：

当λ= 0 时, 只有当前状态被更新

**MC 是高方差, 零偏差：**

低方差 ，在线学习 ，无完整事件序列 在控制方法中使用 TD 替代 MC：

Et (s) = 1(st = s) V(s) ← V(s) + αδtEt (s)

需要完整的事件序列计算出回报后学习

好的收敛性(即使是使用逼近器也能保证收敛性)； 与价值函数初始值无关；

原理简单, 使用方便；

等同于 TD(0) 方法V(s) ← V(s) + αδt

基于 TD 方法训练Q(s,A)，使用 e-贪心方法进行策略提

**TD(**λ**)和 MC**

当λ= 1 时, 可信度被推迟到事件结束，考虑每个事件都到达终止升，每个时刻都对函数更新

状态的环境, 使用离线更新，对每个事件序列, TD(1) 的所有更新 ***基于* *Sarsa* *方法更新动作-价值函数***

要求事件达到终止状态或序列足够长

能够利用马尔可夫性 ，因此在马尔可夫环境下更有效

动作-价值函数的贝尔曼期望方程：

和 MC 的所有更新是一致的

qπ s, a) = R + Y Σs/ P/ Σa/∈Aπ(aI IsI) qπ sI, aI)

**TD 是低方差, 有偏差：**

可以在智能体运行过程中的每一步在线学习

Σ=1αδtEt(s) = Σ=1α(G — V(st))1(st = s)

当使用离散更新时, 前向和后向 **TD(**λ**)**所有的更新是相同的

∈S

在某一时刻智能体观察到的一段事件(s,A, R,s/,AI) 基于样本的 TD 更新:

可以从不完整的事件序列中学习 通常情况是比 MC 更有效

**MC 和 TD(1)**

Q(s,A) ← Q(s,A) + α(R + YQ(sI,AI) — Q(s,A)) 基于 Sarsa 的在策略控制:

**Sarsa vs Q-学习**

Q-学习正确地学到了最优路径, 即沿着悬崖的边缘行走，但是由 于使用的 e-贪心策略会采取一定概率的随机动作, 有时智能体跌 入悬崖

Sarsa 学到的是安全路径,在学习过程中考虑了随机探索动作的影 响 ，即使 Sarsa 学到的安全路径比 Q-学习的最优路径行走步数要 长, 但是每次获得的奖励和却比 Q-学习的高

动态规划和时间差分学习的关系

考虑在一次事件中 s 在 k 时刻被唯一访问一次

 s)

(但是与逼近器结合后并不一定收敛)

受价值函数初始值影响

不能利用马尔可夫性 ，因此在非马尔可夫环境下更有效 偏差/方差的权衡：

vπ(

TD(0) 能够收敛到

该状态的 TD(1) 资格迹会在被访问之后随时间衰减

在每一时刻:

if t < k if t ≥ k

Et (s) = YEt-1(s) + 1(st = s) = {Y0t-k

策略评估:

使用 Sarsa 方法, Q ≈ 策略提升:

TD(1) 在更新时会在线累计误差

qπ

Σ=-αδtEt(s) = α Σ=-Yt-kδt = (Gk — V(sk))

vπ (s

回报Gt = Rt+1 + YRt+2 + ⋯ YT-1Rt是

t)的无偏估计

vπ (s

R + Yvπ (

)是

. .a )是

t)的无偏估计

真实的 TD 目标

在事件结束时累计误差等于

t+1

t+1

vπ (s

t)的有偏估计

e-贪心策略提升 算法：

TD 目标

R + YV(

δk + γδk+1 + γ2δk+2 + ⋯ + γT-1-kδT-1 当λ= 1 时, TD 误差可以变换成 MC 误差

t+1

t+1

目标比回报的方差要小很多

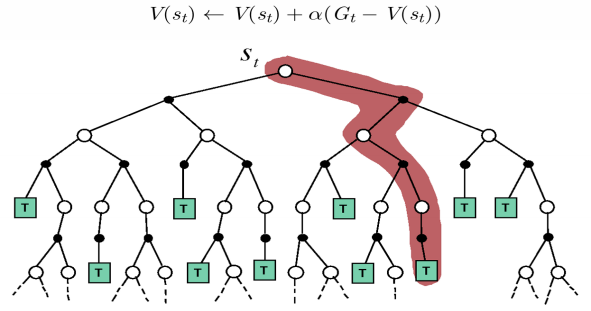
TD

δt = R + YV t

回报的计算包含整个序列中的随机动作, 转移状态, 奖励 TD 目标只包含一个时刻的随机动作, 转移状态, 奖励

t+1

+1) — V(st)

δk + γδk+1 + γ2δk+2 + ⋯ + γT-1-kδT-1 = Gt — V(st)

**TD(**

λ**) 和 TD(1)**

TD(1) 大致等价于每次经过的 MC 方法 ，误差会随在线运行逐步 累积，如果价值函数只在事件结束时离线更新，那么所有的更新量

等同于 MC 方法

λ**)**

**前向和后向 TD(**

考虑在一次事件中 s 在 k 时刻被唯一访问一次

该状态的 **TD(**λ**)**资格迹会在被访问之后随时间衰减

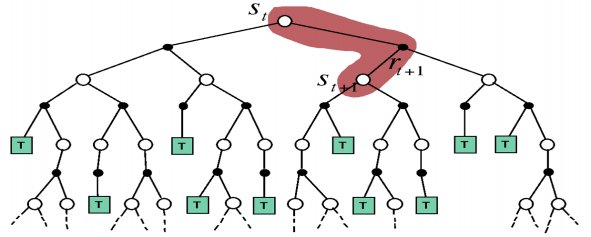
Et (s) = YλEt-1(s) + 1(st = s) = {(λ)t-kif  k

Sarsa 的收敛性：

当如下条件满足时 Sarsa 算法是收敛到最优动作-价值函数： 策略序列πt (aIs)是 GLIE 的

**后向 TD(**

λ**)**在更新时会在线累计误差

Σ=-αδtEt(s) = α Σ=-(λY)t-kδt = (G — V(sk))

更新步长αt满足 Robbins-Monro 序列要求

MC：V(st) ← V(st) + α(Gt — V(st))



在事件结束时总共累计误差等于λ -回报

如果 s 被多次访问, Et (s)也会累积多次的误差

**Sarsa vs MC**

前向提供理论原理，后向给出算法实现：在线学习；每一时刻更新；注意: 蒙特卡洛方法并不能直接应用在这个问题上

可以适用事件序列中的一小段, 非完整序列 并不是所有的策略都能到达终止状态 ，如果学到的某一策略使智

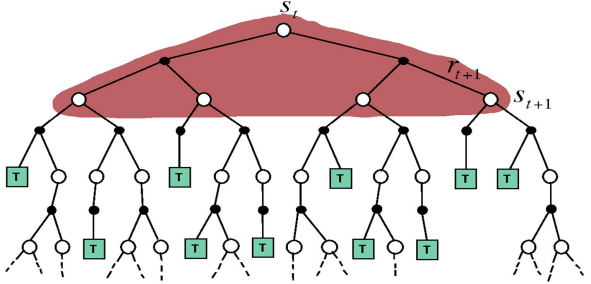
目标: 获得最优策略, 基于模型：动态规划 、价值迭代 、策略迭代能体待在原地不动，MC 在基于该策略对事件采样时永远都无法

目标: 获得策略的价值函数, 基于样本：迭代策略评估.蒙特卡洛方结束 其中，x y ≡ x ← x + α(y — x)

法.时间差分学习.TD(λ) . Sarsa 算法没有这样的问题 探索(exploration) 与利用(exploitation)

我们想获得最优策略, 但是 MDP 的模型未知，或者问题空间太大,每一步都在学习，很快就会发现这样的策略性能很差, 然后转向其在线决策过程存在的一个关键问题是:

只能考虑和智能体最相关的状态：无模型方法；使用智能体的经验它策略 利用: 根据当前的信息做出最佳的决策

样本；最优化价值函数 **n-步 Sarsa** 探索: 采样更多的信息

) — V(st))

TD：V(st) ← V(st) + α

R  + YV(t+1

t+1

上节课:无模型预测, 估计一个模型未知 MDP 问题的价值函数 考虑 n-步的回报, n = 1, 2,∞: 想要做出长期的最佳决策有时需要牺牲当前短期的部分利益

这节课:无模型控制, 最优化一个模型未知 MDP 问题的价值函数

收集更多的信息从而做出整体最佳的决策

在策略(on-policy) 和离策略(off-policy) 学习

Q-学习

在策略学习 ：根据策略π 产生的样本来学习关于 π 的相关知识 如果 Q-学习没有探索, 即 e = 0, 学习结果容易陷入局部最优解,

无法找到真正最优策略，如果 e 很大也不好

离策略学习：根据另一策略μ产生的样本来学习关于 π 的相关知识

1 状态, 动作空间很大时, 想要探索整个空间是费时费力的, 对

策略迭代：

定义 n-步的 Q-回报

在线学习过程不利

qt(n) = Rt+1 + YRt+2 + ⋯ + Yn-1Rt+n + YnQ(st+n,At+n) 2 随机的动作会降低智能体的实际回报

非均匀探索概率

n-步 Sarsa 算法更新 Q(s, a), 向 n-步 Q-回报逼近

Q(st,At) ← Q(st,At) + α(qt(n) — Q(st,At)) e-贪心策略除了最优动作, 其它动作以等概率随机选择

动态规划更新：V(st) ← Eπ [

R + YV( t

   +1] **自举法:** 更新时包含一个猜测量

MC 不使用自举 DP 使用自举 TD 使用自举 **采样法:** 使用采样的数据计算期望

MC 使用采样 DP 不使用采样 TD 使用采样

**前向 Sarsa(λ)** 假如: 有两个动作看起来不错, 而其它的动作完全不可行

t+1

qλ 回报包含了所有的n-步Q-回报qt(n) 那么比较合理的探索方式是选择那些看起来不错的动作，从不错

的动作中选出最优的动作 ，而不是将精力浪费在看起来不可行的 动作 ，**根据动作-价值决定动作被选中的概率**

使用(1 — λ)λn-1对回报加权

q = (1 — λ)1 λn-1qt(n)

**波尔兹曼探索, Boltzmann exploration**

eQ(s,a)⁄T

***等价模型：***

π(aIs) =

MC 收敛结果对应最小二乘误差，最佳匹配观测的回报

Σ=1 Σ1 (G — V(s))2

Σa/∈A eQ(s,a/)⁄T

a被选中的概率与eQ(s,a)⁄T呈正比(Q(s, a) ≥ 0, ∀s, a) 温度系数 T > 0 决定了策略的随机性能

基于蒙特卡洛的策略迭代：

前向 Sarsa(λ) 算法

Q(st,At) ← Q(st,At) + α(q — Q(st,At))

s,A, , ,Y〉

〈

TD(0) 收敛结果对应最大似然马尔可夫模型

如果 T 很大, 所有动作几乎以等概率选择( 探索)

1

Σk ΣTk 1(s ,a,s+1 = s, a, sI)

如果 T 很小, 高价值的动作更容易被选中( 利用) 极限情况 T→ 0, 只选择最优动作

a,s/ =

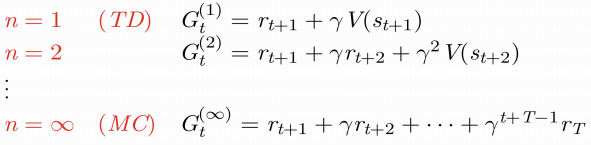
**后向 Sarsa(λ)**

N(s, a)

k=1 t=1 1

和 TD(λ) 算法一样, 我们将资格迹引入到在线的控制算法中

Σ=1 Σ11(s ,a = s, a)rtk

但是 Sarsa(λ) 算法不同的地方在于为每个状态- 动作对都记录资求解非线性算子 max

s/ =

N(s, a)

动态规划: 价值迭代/策略迭代 模型已知

格迹(之前是只对状态记录)

E0(s, a) = 0 Et (s, a) = YλEt-1(s, a) + 1(st = s,At = a)

预测(prediction): 蒙特卡洛方法, 时间差分学习, TD(λ)

在每一时刻会对每个状态s和动作a的Q(s, a)更新 更新量与 TD 误差 t和资格迹Et (s, a)呈正比

控制(control): Sarsa, Sarsa(λ), Q-学习

δ

δt = Rt+1 + YQ t+1,At+1) — Q(st,At)

动作-价值函数实现无模型的策略迭代：

Q(s, a) ← Q(s, a) + αδtEt (s, a)

n-步的回报:

基于 V(s) 进行贪心策略提升需要 MDP 的模型信息 ))

Gt(n) = Rt+1 + YRt+2 + ⋯ Yn-1Rt+n + Ynv(

st+n)

π/(s) = arg max

R + YP/V(sI

(

n-步时间差分学习

a∈A

V(st) ← V(st) + α(Gt(n) — V(st))

**TD(**λ**),**

 λ**-回报**

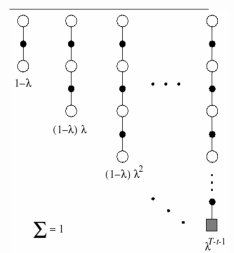
λ-回报G将所有的 n-步回报Gt(n)整

合在一起

对每项使用权重(1 — λ)λn-1

G = (1 — λ) 1 λn-1Gt(n)

基于 Q(s, a) 进行贪心策略提升是无模型的 )

a∈A

π/(s) = arg max Q(s, a

**ϵ-贪心探索**

( √)代表算法会在最优价值函数附近振荡 最小二乘策略迭代：

策略评估: 使用最小二乘学习策略的 Q 函数

策略提升: 贪心策略提升

最简单实现连续探索的方法，所有的 m 个动作都有非零概率被选 中执行 ，以 1-e 概率选择贪心动作，以 e 概率随机选择一个动作

π(aIs) = {E⁄mE — E if e Q(s, a)

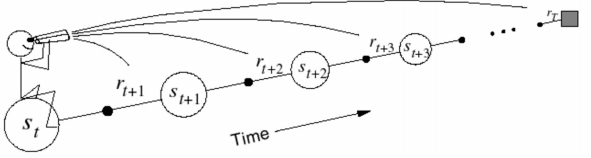
给定任意 e-贪心策略 π, 根据qπ构造出新的 e-贪心策略π/具有更 好的性能, 即vπ/(s) ≥ vπ (s)

前向更新的 TD(λ)形式

V(st) ← V(st) + α(G — V(st))

qπ (s, π/ (s)) = Σa∈Aπ/(aIs)qπ (s, a)

**一步 Sarsa vs Sarsa(λ)**

= E⁄m Σa∈Aqπ (s, a) + (1 — E)arg

qπ (s, a)

max

一步 Sarsa 只对最终导致高奖励的最后一步动作强化它的价值

**前向 TD (**

λ**)**

a∈A

≥ E⁄mΣa∈Aqπ (s, a) + (1 — E)Σa∈A

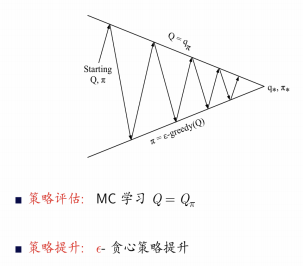
π(aIs) — E⁄m

1 — E

资格迹方法能够对事件中的多个动作强化它们的价值 强化的幅度(箭头大小) 随着离奖励步数的增加而衰减 衰减率等于γλ

qπ (s,a)

= Σa∈Aπ(aIs)qπ (s, a) = vπ (s) → vπ/(s) ≥ vπ(s)



这里γ = 1, λ = 0.9

MC 策略迭代： MC 控制

**离策略学习**

**批处理强化学习, Batch RL：**

想要对目标策略π(aIs)进行评估, 计算vπ (s)或qπ (s,a)，但是智能

梯度下降法简单, 易实现，但是对样本的利用率不足

批处理的方法能够基于智能体的经验(“训练数据”)，试着找到最合

体实际的行为策略是μ(aIs)：{s1,A1,R2, … ,sT}~μ 适的价值函数

λ -回报逼近，面向未来时刻计算G**，**与 MC 一样

更新价值函数向

为什么要考虑这种情况?

有时智能体需要观察人类或别的智能体的行为去学习

要求完整的事件序列

**使用逼近器的好处**

使用较少数量的参数就能表达复杂的函数( 计算复杂度)

有时需要重复利用旧策略π1,π2, … , πt-1产生的经验去学习

**资格迹 Eligibility Traces**

可信度: 警铃和闪光是否和高压电直接相关?

受频率启发: 对发生次数最多的状态给予最高的可信度

受近因启发: 对时间上最近发生的状态给予最高的可信度 资格迹将两种方式结合

E0(s) = 0 Et (s) = YEt-1(s) + 1(st = s)

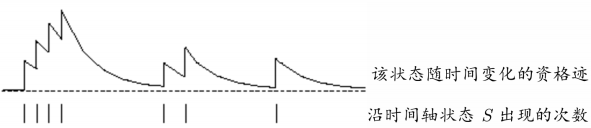
对一个权重的调整就可以影响到很多的点( 泛化能力)

执行探索性的策略去学习最优策略

多种特征表示和逼近器结构( 多样性)

执行单一的策略去学习多个策略

可微的函数逼近器, 例如：特征函数的线性组合，神经网络，决策

考虑基于动作-价值Q(s, a)的离策略学习，不再使用重要性采样 智能体下一时刻执行的动作是由行为策略产生A t-1 ∼ μ( · Ist) 但是学习算法考虑的是由另一个目标策略产生的后继动作

**Q-学习**

树 ，最近邻 ，傅立叶/小波基函数

无限探索, 无穷时刻收敛为贪心策略 (Greedy in the Limit with Infinite Exploration, GLIE)

AI ∼ π( · Ist)

智能体能无限次数地探索所有的状态-动作对：Nk(s, a) = ∞

更新Q(st,At)向另一个后继动作的价值逼近

策略在无穷时刻收敛到贪心策略

Q(st,At) ← Q(st,At) + α(Rt+1 + YQ(st+1,AI) — Q(st,At)) Q-学习实现离策略控制:

πk (aIs) = 1(a = arg  Qk (s,aI))

后向 **TD (**

λ**)**

为每个状态s记录它的资格迹，更新s的价值V(s)

e-贪心策略的探索率E能随时间衰减到零Ek = , 则它就满足 GLIE

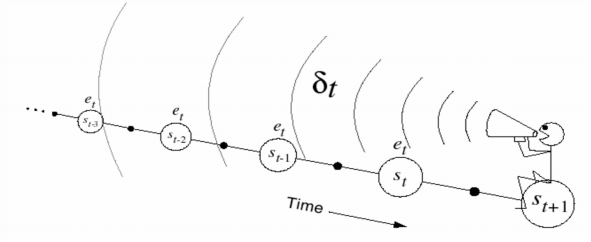
t和资格迹Et (s)呈正比

现在允许行为策略和目标策略都能提升 目标策略π是由Q(s, a)生成的贪心策略

更新量与 TD 误差

δ

GLIE 蒙特卡洛控制

   +1) — V(st) V(s) ← V(s) + αδtEt (s)

δt = R + YV t

使用策略π采集第k次事件{s1,A1,R2, … ,sT}~π 对事件中的每个状态st和动作At

t+1

π(st+1) = arg Q(st+1, aI)

N(st,At) ← N(st,At) + 1

行为策略μ是由Q(s, a)生成的例如e-贪心策略

**那么 Q-学习的目标就是**

1

Rt+1 + YQ(st+1,A/) = Rt+1 + Y arg Q(st+1, a/)

Q(st,At) ← Q(st,At) + N(st,At) (Gt — Q(st,At))

= Rt+1 + Y Q(st+1, a/)

基于新得到的动作-价值函数对策略进行提升

Q-学习控制算法

1

ε ← π ← greedy(Q)

Q(st,At) ← Q(st,At) + α (R + Y Q(sI, a/) — Q(s,A))

k

GLIE 蒙特卡洛控制能够收敛到最优动作-价值函数,

Q 学习控制能够收敛到最优动作-价值函数,

Q(s, a) → q\*(s, a)