【David Silver强化学习公开课之二】马尔可夫决策过程MI

发表于 2016-06-12 | 分类于 project experience | | 4104

本文是David Silver强化学习公开课第二课的总结笔记。主要介绍了马尔可夫过程(MP)、马尔可夫奖赏过
(MRP)、马尔可夫决策过程(MDP)是什么，以及它们涉及到的一些概念，结合了课程ppt给出的例子对概念有
些直观的了解。

【转载请注明出处】chenrudan.github.io

本文是David Silver强化学习公开课第二课的总结笔记。主要介绍了马尔可夫过程(MP)、马尔可夫奖
(MRP)、马尔可夫决策过程(MDP)是什么，以及它们涉及到的一些概念，结合了课程ppt给出的例子对概念
些直观的了解。

本课视频地址:RL Course by David Silver - Lecture 2: Markov Decision Process。

本课ppt地址:http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/MDP.pdf。

文章的内容是课程的一个总结和讨论，会按照自己的理解来组织。个人知识不足再加上英语听力不是那么好
有一些理解不准的地方，欢迎一起讨论。

建了一个强化学习讨论qq群，有兴趣的可以加一下群号595176373或者扫描下面的二维码。

1.基础概念

状态集合S: 有限状态state集合，s表示某个特定状态

动作集合A: 有限动作action集合，a表示某个特定动作

状态转移矩阵P: 矩阵每一项是从S中一个状态s转移到另一个状态{s'}的概率$P_{ss'} = P[S_{t+1} = s'|S_t = s]$ 以
动作a后从一个状态转移到另一个概率为$P^a_{ss'} = P[S_{t+1} = s'|S_t = s, A_t = a]$ 。这里的状态转移矩阵决定
可夫性质，即未来状态只与当前状态有关而与过去状态无关。矩阵一行之和为1。

策略π: 状态s下执行动作a的概率，$\pi(a|s) = P[A_t = a|S_t = s]$

reward函数ER: 这个函数是immediate reward的期望，即在时刻t的时候，agent执行某个action后下一个时刻能得到的reward R_{t+1}的期望，它由当前的状态决定。状态s下immediate reward为$ER_s = E[R_{t+1}|S_t = s]$，状态s下执行动作a后immediate reward期望为$ER_s^a = E[R_{t+1}|S_t = s, A_t =$

Return G_t与discount γ: G_t是t时刻之后未来执行一组action能够获得的reward，即t+1、t+2、t+3...未来所有reward之和，是未来时刻reward在当前时刻的体现，但是越往后的时刻它能反馈回来的reward需要乘以discount系数，系数$\gamma \in [0,1]$会产生一个打折的效果，这是因为并没有一个完美的模型能拟合出未来会发么，未来具有不确定性，同时这样计算会方便，避免了产生状态的无限循环，在某些情况下，即时产生的r即R_{t+1}会比未来时刻更值得关注，符合人的直觉。因此$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

状态值函数$v(s)$: 即基于t时刻的状态s能获得的return的期望，$v(s) = E[G_t|S_t = s]$，这里是仅按照状态转阵选择执行何种动作，如果加入动作选择策略，那么函数就变成了$v_\pi(s) = E_\pi[G_t|S_t = s]$

动作值函数$q_\pi(s,a)$: 基于t时刻的状态s，选择特定的一个action后能获得的return期望，这里的选择过程就入了策略。$q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]$

2. MDP与实例分析

马尔可夫链/过程(**Markov Chain/Process**)，是具有markov性质的随机状态s1,s2,...序列。由$[S, P]$组成。如圆圈内是状态，箭头上的值是状态之间的转移概率。class是指上第几堂课，facebook指看facebook网页，pu酒吧，pass指通过考试，sleep指睡觉。例如处于class1有0.5的概率转移到class2，或者0.5的概率转facebook。

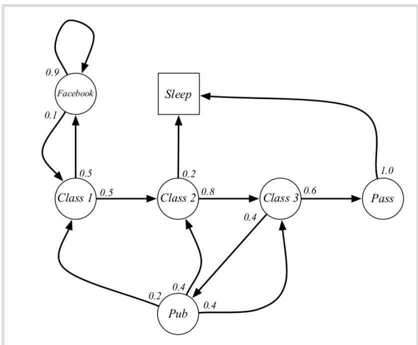


图1 Markov Process Example(图片来源[1])

从而可以产生非常多的随机序列，例如C1 C2 C3 Pass Sleep或者C1 FB FB C1 C2 C3 Pub C1 FB FB FB C1 Pub C2 Sleep等。这些随机状态的序列就是马尔可夫过程。这里可以看到有一些状态发生了循环。

马尔可夫奖赏过程(**Markov Reward Process**)，即马尔可夫过程加上value judgement，value judegment即判像上面一个特定的随机序列有多少累积reward，也就是计算出$v(s)$。它由$[S, P, R, \gamma]$组成，示意图如下。

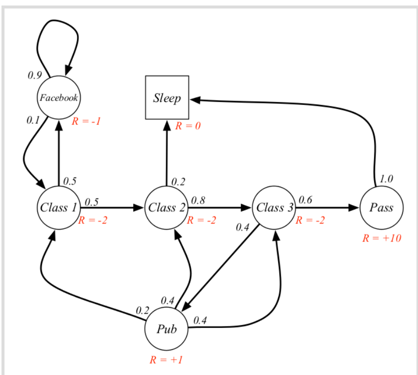


图2 Markov Reward Process Example(图片来源[1])

可以看出比图1多了红色部分即R，但是R的取值只决定了immediate reward，在实际过程中肯定是需要考虑到步骤的reward才能确定当前的选择是否正确。而实际上$v(s)$由两部分组成，一个是immediate reward，一个续状态产生的discounted reward，推导如下(这里我觉得视频里似乎把ER_s与R_{t+1}的取值当成一样的了)，来的这个式子称为Bellman方程。

$$v(s) = E[G_t|S_t = s] \quad = E[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots)|S_t = s] \quad = E[R_{t+1} + \gamma G_{t+1}|S_t = s] \quad = ER$$

文章目录 站点概览

那么每一个状态下能得到的状态值函数取值或者说累积reward如下所示，即原来写着class、sleep状态的地方成了数字(这里假设$\gamma = 1$)。可以从sleep状态出发，推导出每个状态的状态值函数取值，如右上角红色公式所最右的-23与-13，列出二元一次方程组即可求出。

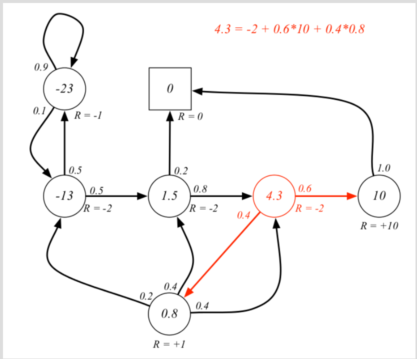


图3 State Value Function Example(图片来源[1])

将Bellman方程表达成矩阵形式，变成了$v = ER + \gamma Pv$，是个线性等式，直接求解得到$v = (I - \gamma P)^{-1}$而这样求解的话计算复杂度是$O(n^3)$，所以一般通过动态规划、蒙特卡洛估计与Temporal-Difference learnin迭代的方式求解。

马尔可夫决策过程(**Markov Decision Process**)，它是拥有决策能力的马尔可夫奖赏过程，个人理解是MRP是情况都遍历，而MDP则是选择性的遍历某些情况。它由$[S, A, P, R_s^a, \gamma, \pi(a|s)]$组成，并且拥有两个数$v_{\pi(s)}$和$q_\pi(s,a)$。根据这两个值函数的定义，它们之间的关系表示为$v_\pi(s) = \sum_{a\in A}\pi(a|s)q_\pi(s$及$q_\pi(s,a) = R_s^a + \gamma\sum_{s'\in S}P_{ss'}^a v_\pi(s')$。第二个式子是说当选择一个action之后，转移到不同状态下之后reward之和是多少。将两个式子互相代入，可以得到如下的Bellman期望方程。

$$v_\pi(s) = \sum_{a\in A}\pi(a|s)(R_s^a + \gamma\sum_{s'\in S}P_{ss'}^a v_\pi(s')) \quad (2)$$

$$q_\pi(s,a) = R_s^a + \gamma\sum_{s'\in S}P_{ss'}^a\sum_{a'\in A}\pi(a'|s')q_\pi(s',a') \quad (3)$$

下图是一个MDP的例子，箭头上的单词表示action，与MRP不同的是，这里给出的immediate reward是同个状态s和某个动作a条件下，所以图中R不是只取决于s，而是取决于s和a。右上角的等式表达出了这一个状态值函数求解过程。

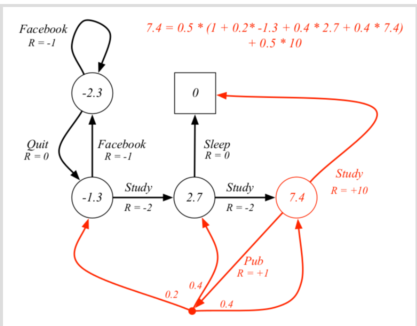
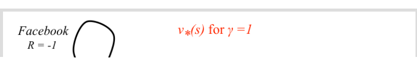


图4 Markov Decision Processes Example(图片来源[1])

由于策略$\pi(a|s)$是可以改变的，因此两个值函数的取值不像MRP一样是固定的，那么就能从不同的取值中个最大值即最优值函数(这节课没有讲如何求解)。例如下面两个图就是上面的例子能找到的最优状数$v_*(s)$与最优动作值函数$q_*(s,a)$。如果知道了$q_*(s,a)$，那么也就知道了在每一步选择过程中应该选择什动作。也就是说MDP需要解决的问题并不是每一步到底会获得多少累积reward，而是找到一个最优的解决这两个最优值函数同样存在着一定关系，$v_*(s) = \max_a q_*(s,a)$，从而可出$q_*(s,a) = R_s^a + \gamma\sum_{s'\in S}P_{ss'}^a \max_{a'} q_*(s',a')$，这个等式称为Bellman优化方程，它不是一个线性等式闭式解。通常通过值迭代、策略迭代、Q-learning、Sarsa等方法求解。

$$v_*(s) = \max_a q_*(s,a) \quad (4)$$

$$q_*(s,a) = R_s^a + \gamma\sum_{s'\in S}P_{ss'}^a \max_{a'} q_*(s',a') \quad (5)$$



文章目录 站点概览

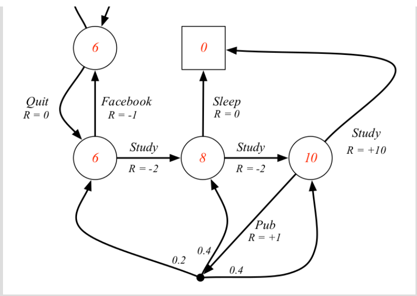


图5 Optimal State-Value Function Example(图片来源[1])

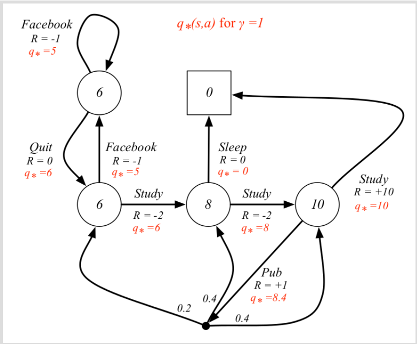


图6 Optimal Action-Value Function Example(图片来源[1])

实际上一定存在这样的一个最优策略π_*。可以通过最大化$q_*(s,a)$获得。针对任意的MDP问题，总是存在一
优的deterministic policy。

以上是第二课的主要内容，从MP开始到MRP再到MDP，了解值函数的具体概念与reward有什么联系，重点
绍MDP面对的问题，暂时没有提到如何解决。结合课程的例子对各个概念有比较直观的了解。但是这一课有
念是"MDP描述了强化学习的environment，且是fully Observable的"，这个意思我暂时没明白。如果M
environment，那么它为什么有policy，为什么需要最大化policy，个人觉得MDP应该是fully Observable的强
习，也就是说MDP本身描述的就是一个强化学习问题，它的状态转移矩阵和reward function就是environmen
的目标就是找到最优的动作值函数或者最优Policy。

[1] http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/MDP.pdf

【David Silver强化学习公开课之一】强化学习入门

【David Silver强化学习公开课之三】动态规划解决MDP的Planning问