表格方法求解RL笔记

Thursday, June 18, 2020 3:45 PM

2020.6.18 晴

知识复习:

强化学习：机器学习的一个领域，强调如何基于环境而行动，以取得最大化的预期利益。（维基百科）

强化学习的基本要素：环境（environment），行为主体（agent），状态（state），收益（reward）、行动（action）

强化学习过程的简单描述：agent在environment中可以采取多种actions，通过environment和agent的交互设立每次可能呈现的state，并给出每个state的reward来决策最优的action选择。

强化学习的方法：分为value-base方法，policy-base方法

常用算法库：parl、baseline、garage、coach、dopamine、ray rillib

常用环境库：gym（离散控制环境（动作可数）：atari，连续控制环境（动作连续）：mujoco）

推荐书籍与深入课程：

书籍推荐：《Reinforcement Learning：An Introduction（强化学习导论）》

课程推荐： 2015 David Silver经典强化学习公开课、 UC Berkeley CS285 、斯坦福 CS234、伯克利 2018 Deep RL课程：http://rail.eecs.berkeley.edu/deeprlcourse/

强化学习经典论文：

DQN. Playing atari with deep reinforcement learning
https://arxiv.org/pdf/1312.5602.pdfA3C.

Asynchronous methods for deep reinforcement learning
http://www.jmlr.org/proceedings/papers/v48/mniha16.pdfDDPG.

Continuous control with deep reinforcement learning.
https://arxiv.org/pdf/1509.02971

PPO. Proximal policy optimization algorithms.
https://arxiv.org/pdf/1707.06347

SARSA方法:

简介:

sarsa方法是一个基于表格搜索的on-policy方法，on-policy方法可以简单理解为一种在线的学习方法，在强化学习领域，为解决一个问题需要实行两个策略，一个是行为策略一个是目标策略。行为策略是用来与environment互动产生数据的策略。而目标策略是在行为策略产生的数据中进行学习、优化最后拿来应用的策略。这里提到的概念比较多，可能不容易理解。我们简单的举一个例子：国王派士兵去攻打城墙，士兵负责收集城墙哪里容易攻破的信息，属于行为策略。而国王负责确定真正攻打城墙的地点，也就是目标策略。即行为策略更像是探子，目标策略更像是军师。而on-policy方法是一种行为策略和目标策略统一的方法，也就是说负责收集信息和决策攻城地点的是同一个人。相对应的off-policy方法就是指行为策略和目标策略是分开的方法，在后面提到的QLearning就是一种off-policy方法。

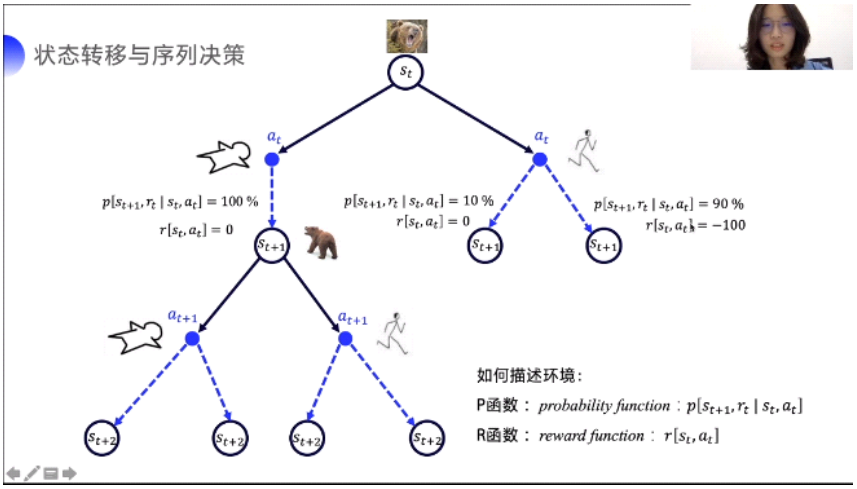
MDP和Q表格：

通过对强化学习的初步了解，我们知道关于强化学习的过程，实际上是一系列有条件的决策所组成的时间序列：agent采取了一个action，与environment交互后得到了一个state和一个reward，根据reward的大小决定下一个action的选择，期望最终的reward最大。可以看到每次的下一时刻的state只取决于当前的state，跟前面的state是无关的，这就是马尔科夫决策过程。那么我们如何来描述决策的environment呢？这里涉及到两个函数：状态转移概率函数P函数和奖励函数R函数。

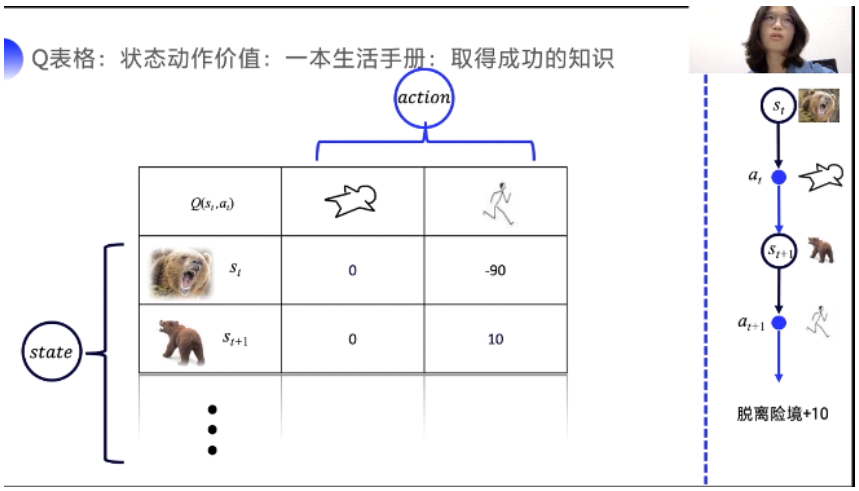
P函数指的是在决策的过程中，每个行为导致产生不同状态的条件概率。

R函数指的是不同行为不同状态与所得奖励之间的函数关系。

拿森林遇熊的故事举例：P函数指的是我看到熊后选择逃跑或装死（action）时活下来或被熊吃掉（state）的概率，R函数指的是我想要活下来，那么从以往的经验中在看到熊后选择逃跑或装死活下来几率大的那个行为会成为我优先选择的行为（给该行为匹配一个较高的reward）。



在描述环境的过程中，当我们的P函数和R函数已知时，这就是一个model-based问题，而当P函数和R函数未知时，这就是一个model-free问题。对于model-based问题我们可以通过动态规划找到最优路径，而SARSA解决的是model-free问题。对model-free问题，我们可以通过大量试错（探索）来学习到一些经验，为了表示这些经验，我们来建立一个知识表格，也就是Q表格（状态动作价值表格），表格中包含的是取得成功的知识。表格的样式如下图：

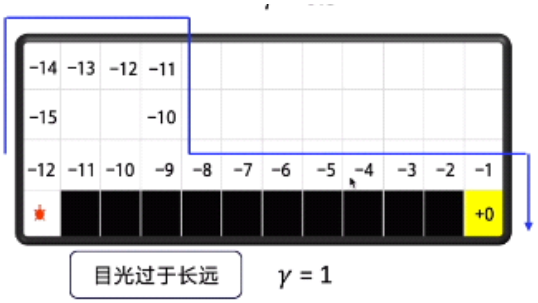


横坐标：actions、纵坐标：states、填入值：总收益G。

G实际上代表的是之后的每次决策带来的reward的总和，但我们并不想让非常遥远的未来的决策对当前的收益产生过大的影响，所以我们设置一个收益衰减，则得到如下的G公式：

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

我们可以看下强化学习在迷宫寻路问题中实际的训练过程：首先我们设立一个奖惩机制，对于每一个state来说，我们都设置一个均等的reward，在Q表格为0的时候，我们让机器随便的去探索和试错，在没有到达终点前，机器在每一个state中采取任意action的概率是均等的，在无数次训练中，由于到达终点时所走过的路径长短存在区别，每条路径中经过的states所得到的总reward也是不同的，越短的路径，总reward也就越高.假设我们先取收益衰减为1，则得到下图：



这里提到了两条路径，表格上显示的是该条路径中每个state的总收益，可以看到在第一个点中向上走的总reward是要比向右走的低的。也就是说向右走代表了可能的更优策略。（这里我们要强调一下，该图片表示的并不是Q表格的内容，而只是将两条路径的G值标注在不同的state上。Q表格是一个纵坐标为states，横坐标为actions的二维表格）

SARSA方法：

那么我们如何更新这个Q表格呢?

首先，我们对G公式进行一个简单的数学变换，可以得到如下的结果：

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots\right) \\ &= R_{t+1} + \gamma\, G_{t+1} \end{aligned}$$

由上式可知，Gt可以由Gt+1和t+1时刻的reward得到，因此得出一个结论，我们可以通过使用下一次state的

reward和G来更新当前state的G（总reward），我们构建一个简单的更新策略：时序差分单步更新策略，我们从当前Q表格中取得下一次action的G，此时action的选取采用的是E-greddy方法（之后会介绍这个方法），然后进行如下变换得到新的G：

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big]$$

其中a是学习率，这里为小白同学科普一下，在机器学习领域，学习率代表的是随着时间的推移，信息累计的速度。机器学习是在学习一种变化的趋势，如果我们单纯的用下一时刻的Q来替换当前Q是没有意义的。

以上就是sarsa的学习过程，接下里说说e-greddy方法，这个方法实际上是一种平衡利用和探索问题的策略，在机器学习的过程中，我们通过探索（随机试错）会得到一些知识（经验值），利用这些知识可以提高学习的效率，但是这些知识是需要不断积累的（继续随机试错），所以在每次决策的时候，我们需要利用已有的知识并提供一定的探索空间，为达到这个目的，我们首先设置一个阈值，高于这个阈值时我们随机的选择动作，低于这个阈值时，我们根据Q矩阵选择G最大的动作（这个动作可能有多个，在其中随机的选择一个就可以了）。在这种方法中，我们大概率选择了当前的最佳选项（贪婪），小概率的选择随机选项，这样可以有效的利用已经学到的知识（得出贪婪的策略），又避免对知识的过度依赖（产生局部最优解）。

QLearning方法：

简介

有了sarsa方法的基础，QLearning方法理解起来就简单的多了。对比来看，QLearning方法与sarsa最主要的不同就在于Qlearning是一种off-policy方法，也就是说它的行为策略和目标策略是分开的，这就意味着model在学习的过程中，数据的产生和学习是分开进行的，这是什么意思呢？我们在训练时，不需要实际的产生一个交互得出新的数据后再进行学习，而是从之前已经产生的数据中直接进行学习。对比sarsa方法：sarsa方法在学习的过程中需要产生下一次的action，该action一定会被执行，我们查找当前的Q表格，通过下一次的state和action取得确定的Q值后对当前Q值进行更新。而QLearning方法则不需要知道下一次的action，当更新Q值时直接通过取Q表格state t+1那一行中最大的Q值来对当前Q值进行更新，至于下一次action实际上执行的是哪个动作我们并不关心。换句话说QLearning的目标策略是绝对的贪婪策略（直接取Q的最大值）。其更新函数如下：

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big].$$

这样会导致什么结果呢？QLearning选出的策略要更加optimal而sarsa要更加的safe。这里又是个不容易理解的点，我们简单的说明一下，在sarsa方法中，更新函数是直接与action的选取有关的（行为策略和目标策略是一个策略），它是e-greddy方法选择的结果，这就意味着它有一定的概率是随机选择action的，所以在存在悬崖问题的时候，学习过程中，Q表格中是记录了action选择的信息的，也就包括了掉进悬崖的可能，这就使得Q中接近悬崖的那些点获得了较低的G值，最终的Q会偏向于给出远离悬崖的路径，而Qlearning 就没有这个问题（目标策略为绝对贪婪策略），当Q收敛时，Q中绝对不会记录可能掉进悬崖的action选择了（掉进悬崖会导致过低的reward，学习过程中这种action不可能被选择），所以最终QLearning会给出一条贴近悬崖的最短路径，这也是训练过程中sarsa的平均reward要高于qlearning的原因，也就是说sarsa的在线学习效果要优于qlearning，具体可以去看课程里面对cliffwalking问题的举例。

实现代码：

科老师在课上给出了完整的SARSA和QLearning的实现代码，感兴趣的朋友可以直接去连接里面看，此处贴出两个方法的关键代码对比：

Sarsa:

```
def learning(self,obs,action,reward,next_obs,next_action,done):
        predict_q = self.Q[obs,action]
        if done:
                target_q = reward
        else:
                Target_q = reward + self.gama*self.Q[next_obs,next_action]
        self.q[obs,action] += self.lr*(target_q - predict_q)

Qlearning:

def learning(self,obs,action,reward,next_obs,done):
        predict_q = self.Q[obs,action]
        if done:
                target_q = reward
        else:
                Target_q = reward + self.gama*np.max(self.Q[next_obs,:])
        self.q[obs,action] += self.lr*(target_q - predict_q)
```

PS：这节课实际上是昨天晚上8:30的课程，因为工作原因，是第二天抽空看的录播，所以笔记更新时间与实际上课时间基本都是间隔一天的，感兴趣的小伙伴如果还在直播时间内的话也可以直接去听科老师的课程，由于单次课程时间有限，知识点还是相对密集的，对于零基础的小伙伴可能需要反复食用。这个系列从第二节课开始，前面的内容更偏向实操和环境搭建，就不做赘述啦，课程链接在下方，欢迎取阅~

强化学习7日打卡营-世界冠军带你从零实践：

https://aistudio.baidu.com/aistudio/education/group/info/1335