Sarsa算法

单步Sarsa

时序差分算法基于价值迭代，Sarsa则基于策略迭代。注意，这里并不像动态规划的策略迭代那样要更新所有状态的价值函数，因为它是无模型的，并且动作价值的估计只依赖执行这个动作的单步reward和之前的动作价值。

5.3 Sarsa 算法

既然我们可以用时序差分方法来估计价值函数，那一个很自然的问题是，我们能否用类似策略迭代的方法来进行强化学习。策略评估已经可以通过时序差分算法实现，那么在不知道奖励函数和状态转移函数的情况下该怎么进行策略提升呢？答案是时可以直接用时序差分算法来估计动作价值函数Q：

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t)+\alpha[r_t+\gamma Q(s_{t+1},a_{t+1})-Q(s_t,a_t)]$$

然后我们用贪婪算法来选取在某个状态下动作价值最大的那个动作，即$\arg\max_a Q(s,a)$。这样似乎已经形成了一个完整的强化学习算法：用贪婪算法根据动作价值选取动作来和环境交互，再根据得到的数据用时序差分算法更新动作价值估计。

然而这个简单的算法存在两个需要进一步考虑的问题。第一，如果要用时序差分算法来准确地估计策略的状态价值函数，我们需要用极大量的样本来进行更新。但实际上我们可以忽略这一点，直接用一些样本来评估策略，然后就可以更新策略了。我们可以这么做的原因是策略提升可以在策略评估未完全进行的情况进行，回顾一下，价值迭代（参见 4.4 节）就是这样，这其实是**广义策略迭代**（generalized policy iteration）的思想。第二，如果在策略提升中一直根据贪婪算法得到一个确定性策略，可能会导致某些状态动作对(s,a)永远没有在序列中出现，以至于无法对其动作价值进行估计，进而无法保证策略提升后的策略比之前的好。我们在第 2 章中对此有详细讨论。简单常用的解决方案是不再一味使用贪婪算法，而是采用一个ϵ-贪婪策略：有$1-\epsilon$的概率采用动作价值最大的那个动作，另外有ϵ的概率从动作空间中随机采取一个动作，其公式表示为：

$$\pi(a|s)=\begin{cases}\epsilon/|\mathcal{A}|+1-\epsilon & \text{如果} a=\arg\max_{a'} Q(s,a')\\ \epsilon/|\mathcal{A}| & \text{其他动作}\end{cases}$$

多步Sarsa

蒙特卡洛方法利用当前状态之后每一步的奖励而不使用任何价值估计，时序差分算法只利用一步奖励和下一个状态的价值估计。那它们之间的区别是什么呢？总的来说，蒙特卡洛方法是**无偏**（unbiased）的，但是具有比较大的方差，因为每一步的状态转移都有不确定性，而每一步状态采取的动作所得到的不一样的奖励最终都会加起来，这会极大影响最终的价值估计；时序差分算法具有非常小的方差，因为只关注了一步状态转移，用到了一步的奖励，但是它是有偏的，因为用到了下一个状态的价值估计而不是其真实的价值。那有没有什么方法可以结合二者的优势呢？答案是**多步时序差分**！多步时序差分的意思是使用n步的奖励，然后使用之后状态的价值估计。用公式表示，将

$$G_t=r_t+\gamma Q(s_{t+1},a_{t+1})$$

替换成

$$G_t=r_t+\gamma r_{t+1}+\cdots+\gamma^n Q(s_{t+n},a_{t+n})$$

于是，相应存在一种多步 Sarsa 算法，它把 Sarsa 算法中的动作价值函数的更新公式（参见 5.3 节）

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t)+\alpha[r_t+\gamma Q(s_{t+1},a_{t+1})-Q(s_t,a_t)]$$

替换成

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t)+\alpha[r_t+\gamma r_{t+1}+\cdots+\gamma^n Q(s_{t+n},a_{t+n})-Q(s_t,a_t)]$$