Q-Learning

与Sarsa算法类似，修改了时序差分的更新方式。Sarsa算法的Q(st+1,at+1)依赖策略，当策略换了则下个状态也不一样，而Q-learning由于取的是动作值函数的最大值，所以他不依赖策略，这种称为离线策略算法。并且在进行完一个动作之后即可进行更新，并不需要等待整个序列更新完。

除了 Sarsa，还有一种非常著名的基于时序差分算法的强化学习算法——Q-learning。Q-learning 和 Sarsa 的最大区别在于 Q-learning 的时序差分更新方式为

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

Q-learning 算法的具体流程如下：

- 初始化$Q(s,a)$
- **for** 序列$e = 1 \rightarrow E$ **do**:
- 　　得到初始状态s
- 　　**for** 时间步$t = 1 \rightarrow T$ **do**:
- 　　　　用ϵ-greedy 策略根据Q选择当前状态s下的动作a
- 　　　　得到环境反馈的r, s'
- 　　　　$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
- 　　　　$s \leftarrow s'$
- 　　**end for**
- **end for**

我们可以用价值迭代的思想来理解 Q-learning，即 Q-learning 是直接在估计Q^*，因为动作价值函数的贝尔曼最优方程是

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a'} Q^*(s',a')$$

而 Sarsa 估计当前ϵ-贪婪策略的动作价值函数。需要强调的是，Q-learning 的更新并非必须使用当前贪心策略$\arg\max_a Q(s,a)$采样得到的数据，因为给定任意(s,a,r,s')都可以直接根据更新公式来更新Q，为了探索，我们通常使用一个ϵ-贪婪策略来与环境交互。Sarsa 必须使用当前ϵ-贪婪策略采样得到的数据，因为它的更新中用到的$Q(s',a')$的a'是当前策略在s'下的动作。我们称 Sarsa 是**在线策略**（on-policy）算法，称 Q-learning 是**离线策略**（off-policy）算法，这两个概念强化学习中非常重要。