免模型预测、免模型控制

免模型预测

Basic information

有模型强化学习：尝试先学习一个环境模型，然后由智能体来计划最佳的行动策略，例如通过模拟可能的未来状态来预测哪个动作会导致最大的累积奖励。它可以在不与真实环境交互的情况下学习，节省实验成本。

免模型学习：直接学习在特定状态下执行特定动作的价值或优化策略。它直接从与环境的交互中学习，不需要建立任何预测环境动态的模型。

预测：对于免模型算法，环境的状态转移概率是未知的，这种情况下会去近似环境的状态价值函数。预测的主要目的是估计或计算环境中的某种期望值，比如状态价值函数或动作价值函数。

控制：的目标是找到一个最优策略，该策略可以最大化期望的回报。

预测和控制问题经常交织在一起。例如，在使用Q-learning时，我们同时进行预测和控制。

Monte Carlo估计

回顾状态价值函数

$$V_\pi(s) = \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots | S_t = s] = \mathbb{E}_\pi[G_t | S_t = s]$$

蒙特卡洛方法的思路是我们可以采样大量的轨迹，对于每个轨迹计算对应状态的回报然后取平均近似，称之为经验平均回报。根据大数定律，只要采样的轨迹数量足够多，计算出的经验平均回报就能趋近于实际的状态价值函数。
局限性：只适用于有终止状态的马尔可夫决策过程。

蒙特卡洛方法主要分成两种算法，一种是首次访问蒙特卡洛（first-visit monte carlo, FVMC）。
FVMC 方法主要包含两个步骤，首先是产生一个回合的完整轨迹，然后遍历轨迹计算每个状态的回报。注意，只在第一次遍历到某个状态时会记录并计算对应的回报。

另外一种是每次访问蒙特卡洛（every-visit Monte Carlo，EVMC）方法。在 EVMC 方法中不会忽略统一状态的多个回报。更加精确，但是计算成本也更高。

每次计算得到新的回报，以一种递进更新的方式进行，如：

新的估计值 <- 旧的估计值 + 步长 * （目标值-旧的估计值）

步长可以理解为学习率。应用到MC方法中，更新公式可以表示为

$$V(s_t) \leftarrow V(s_t) + \alpha[G_t - V(s_t)]$$

时序差分估计

这是一种基于经验的动态规划方法。最简单的时序差分可以表示为

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

这种算法一般称为**单步时序差分**（ one-step TD ），即 TD(0)。

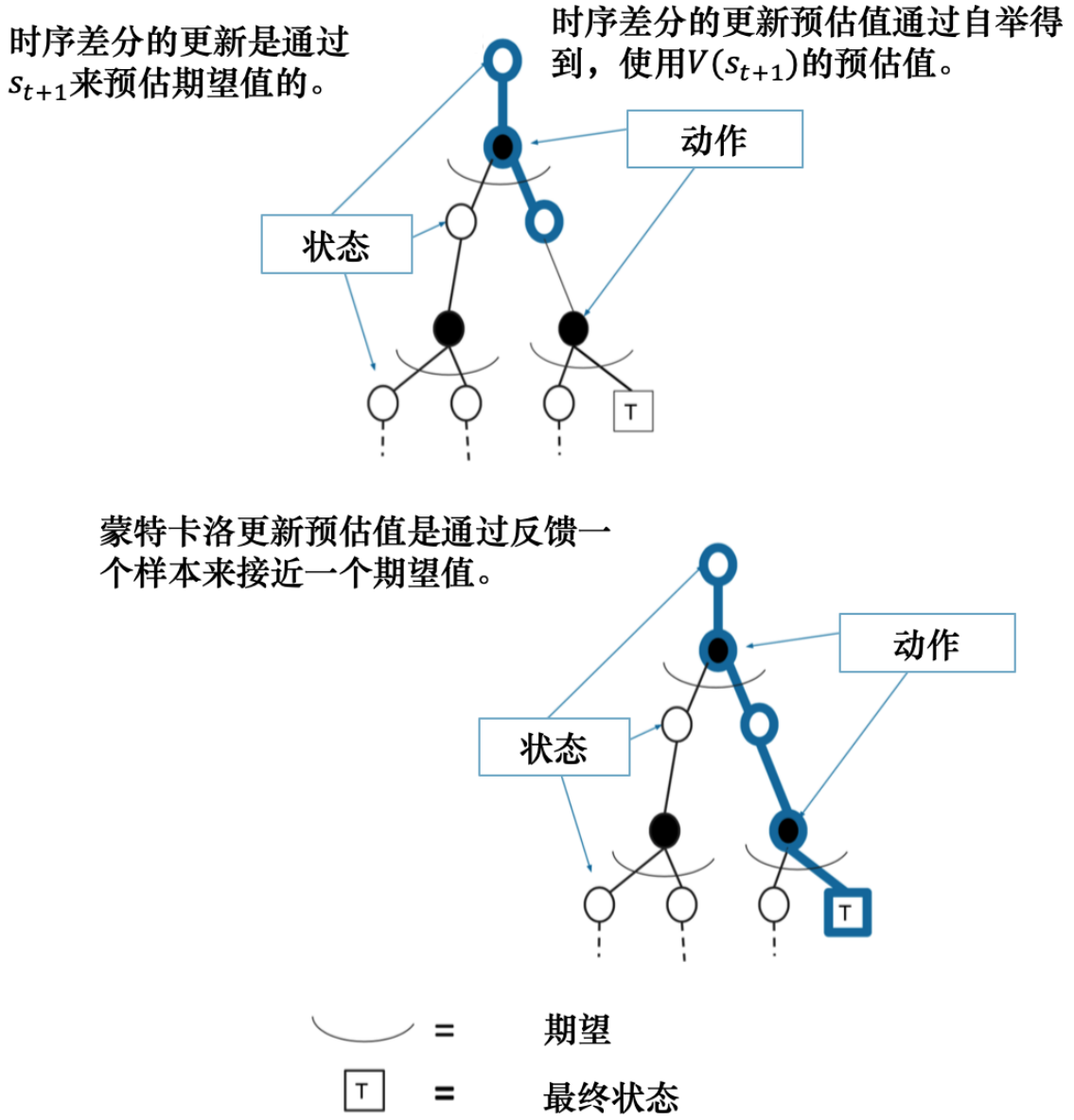
在这个更新过程中使用了当前奖励和后继状态的估计，这是类似于蒙特卡罗方法的；但同时也利用了贝尔曼方程的思想，将下一状态的值函数作为现有状态值函数的一部分估计来更新现有状态的值函数。此外，时序差分还结合了自举的思想，即未来状态的价值是通过现有的估计$r_{t+1} + \gamma V(s_{t+1})$
（也叫做**时序差分目标**）进行计算的，即使用一个状态的估计值来更新该状态的估计值，没有再利用后续状态信息的计算方法。这种方法的好处在于可以将问题分解成只涉及一步的预测，从而简化计算。$\delta = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$被定义为时序差分误差（TD error）。

因此时序差分方法在实践过程中会把终止状态单独做一个判断，即将对应未来状态的估计值设置为 0，然后更新当前状态的估计值，这个过程也被称作**回溯**。

$$\begin{cases} V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} - V(s_t)] & 对于终止状态V(s_t) \\ V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] & 对于非终止状态V(s_t) \end{cases}$$

比较

时序差分 VS MC
在线学习 游戏结束时才可以学习
可从不完整序列上学习 只能从完整序列上学习
可在连续环境下（无终止）学习 只能在有终止的情况下学习
马尔可夫环境下有更高的学习效率 无限制



n步时序差分

$$
\begin{aligned}
&n=1(\mathrm{TD}) \quad G_t^{(1)}=r_{t+1}+\gamma V\left(s_{t+1}\right)\\
&n=2 \quad G_t^{(2)}=r_{t+1}+\gamma r_{t+2}+\gamma^2 V\left(s_{t+2}\right)\\
&n=\infty(\mathrm{MC}) \quad G_t^{\infty}=r_{t+1}+\gamma r_{t+2}+\cdots+\gamma^{T-t-1} r_T
\end{aligned}
$$

当 n趋近于无穷大时，就变成了蒙特卡洛方法，因此可以通过调整自举的步数，来实现蒙特卡洛方法和时序差分方法之间的权衡。这个n我们通常会用λ来表示，即 TD(λ) 方法。

选择λ的几种方法：

- 网络搜索
- 随机搜索
- 自适应选择
- 交叉验证
- 经验取值

免模型控制

Q-learning算法

根据策略与状态价值函数之间是存在的联系：

$$V_\pi(s)=\sum_{a\in A}\pi(a\mid s)Q_\pi(s,a)$$

为了解决控制问题，我们只需要直接预测动作价值函数，然后在决策时选择动作价值即 Q值最大对应的动作即可。这样一来，策略和动作价值函数同时达到最优，相应的状态价值函数也是最优的。

Q-learning算法更新公式：

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

与时序差分方法中状态价值函数的更新公式是一样的。动作价值函数更新时是直接拿最大的未来动作价值的来估计的，而在状态价值函数更新中相当于是拿对应的平均值来估计的。这就会导致这个估计相当于状态价值函数中的估计更不准确，一般称为 **Q 值的过估计**。

Q-table

状态和动作组成的双轴表格。需要初始化。 Q值的更新过程：让机器人自行在网格中走动，走到一个状态，就把对应的Q值更新一次，这个过程就叫做**探索** 。

探索策略

直接根据Q函数（即每次选择Q值最大对应的动作）来探索是没有问题的。但是由于在探索的过程中 Q值也是估计出来的，然后还需要利用先前的估计值来更新 Q值（也就是自举），换句话说，由于自举依赖于先前的估计值，因此这可能会导致估计出的价值函数存在某种程度上的偏差。

故， Q-learning 算法采用了$\epsilon-greedy$探索策略，即智能体在探索的过程中，会以 $1-\epsilon$ 的概率按照 Q函数来执行动作（利用，exploitation），然后以剩下ϵ的概率随机动作（探索，exploration）。

Sarsa算法

Q值更新公式：

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

Sarsa算法是直接用下一个状态和动作对应的Q值来作为估计值的，而Q-learning算法则是用下一个状态对应的最大Q值。

同策略与异策略

Sarsa 算法在训练的过程中当前策略来生成数据样本，并在其基础上进行更新。换句话说，策略评估和策略改进过程是基于相同的策略完成的，这就是**同策略算法**。相应地，像 Q-learning 算法这样从其他策略中获取样本然后利用它们来更新目标策略，我们称作**异策略算法**。也就是说，异策略算法基本上是从经验池或者历史数据中进行学习的

同策略算法：稳定，但效率较低
异策略算法：高效，但是需要让获取样本的策略和更新的策略具备一定的分布匹配条件，以避免偏差。