lec08

贝尔曼最优方程及其求解方法

- 动态规划（Dynamic Programming）
- 蒙特卡罗方法（Monte Carlo）
- 时序差分方法（Temporal Difference）

许多强化学习方法可以理解为近似求解贝尔曼最优方程

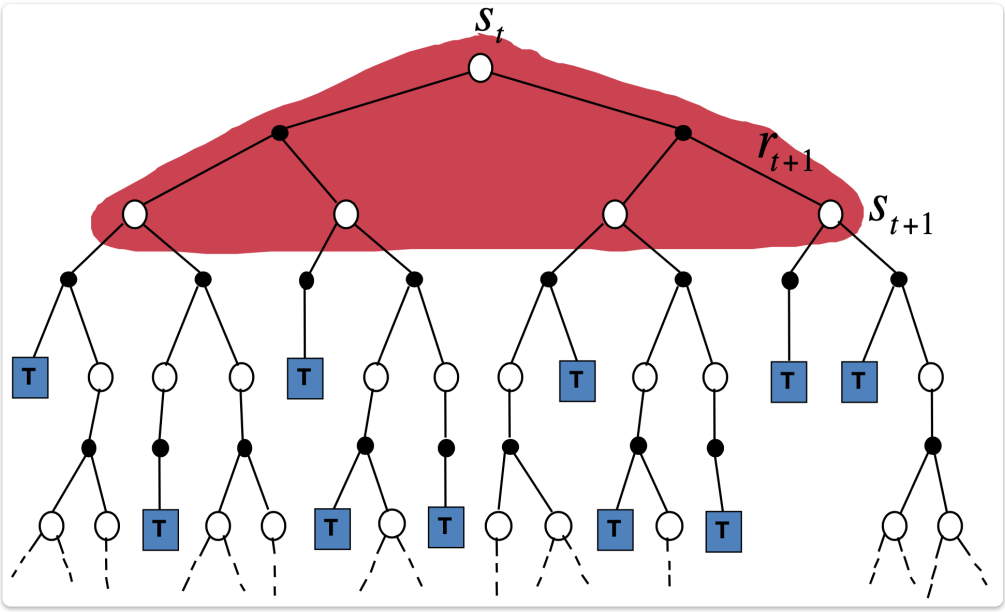
动态规划（Dynamic Programming）

目标：通过显式求解贝尔曼方程来找到最优策略

$$V(s_t) \leftarrow \mathbb{E}_\pi\{R_{t+1} + \gamma V(s_{t+1})\}$$
$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^\pi(s')\right]$$

动态规划的要求：

- 准确了解环境动态（accurate knowledge of environment dynamics）
- 有足够的空间和时间进行计算（enough space and time to do the computation）
- 满足马尔可夫性质（the Markov Property）



但实际中很少应用

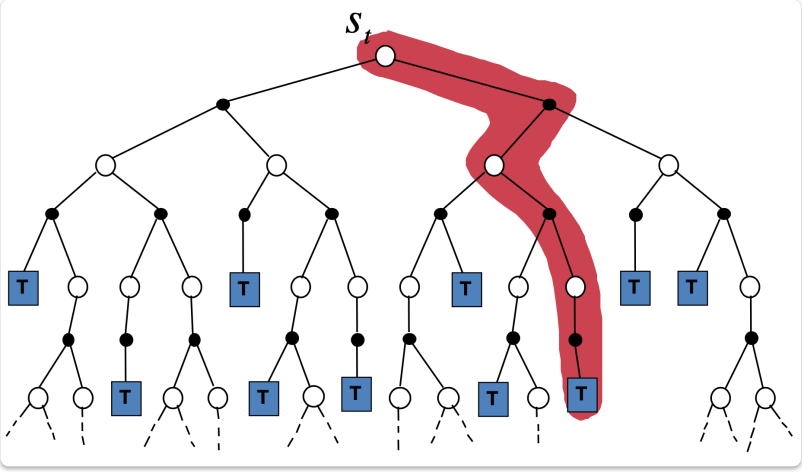
蒙特卡罗方法（Monte Carlo）

- 从完整的样本回报中学习：蒙特卡罗方法从完整的样本回报中学习，适用于情节任务（episodic tasks）
- 直接从经验中学习：蒙特卡罗方法直接从经验中学习
- 当在一个大的状态空间中只有少数几个可能性需要估值时，使用蒙特卡罗方法非常好

蒙特卡罗策略评估（Monte Carlo Policy Evaluation）

- 目标：
  - 学习 $V^{\pi}(s)$
- 给定条件：
  - 给定一些包含状态 s 的 episodes
- 基本思想：
  - 观察经过状态 s 后的平均回报
- 方法：
  - Every-Visit MC：
    - 计算在一个 episode 中每次访问状态 s 的平均回报
  - First-Visit MC：
    - 只计算在一个 episode 中首次访问状态 s 的平均回报
- 收敛性：
  - 两种方法渐近收敛

First-visit Monte Carlo Policy Evaluation



Initialize:
$\pi \leftarrow$ 待评估的策略
$V \leftarrow$ 任意初始化的状态值函数
$Returns(s) \leftarrow$ 一个空列表, $\forall s \in S$

Repeat forever :
(a) 使用策略 π 生成一个 episode
(b)对于 episode 中出现的每个状态 s :
$R \leftarrow$ 第一次出现 s 后返回
将 R 添加到 $Returns(s)$ 列表中
$V(s) \leftarrow Returns(s)$ 的均值

Backup diagram for Monte Carlo 蒙特卡罗备份图

- 蒙特卡罗方法考虑完整的 episode，从开始到终止状态
- 在每个状态下只有一个选择，而不像动态规划那样需要考虑所有可能的动作
- 自举（bootstrap）意味着在更新估计值时使用一个或多个估计值，而蒙特卡罗方法直接从完整情节的回报中学习，不使用部分估计值
- 估计一个状态值所需的时间与状态空间的大小无关，只与具体的情节相关

Value Iteration: Simplest Monte Carlo 值迭代：最简单的蒙特卡罗

$$V(s_t) \leftarrow V(s_t) + \alpha \left[R_{t+1} - V(s_t)\right]$$

其中：

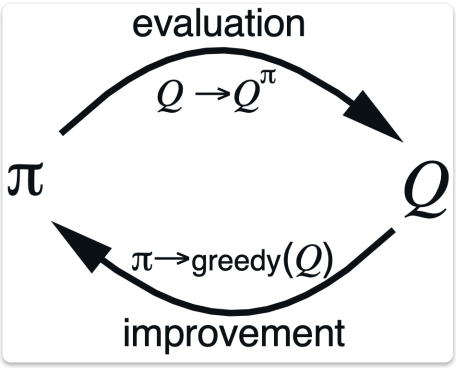
- $V(s_t)$ 是状态 s_t 的当前值函数
- α 是学习率
- R_{t+1} 是状态 s_t 后的实际回报
- $[R_{t+1} - V(s_t)]$ 是当前回报和估计值之间的差异，也称为“误差”或“增量”

Monte Carlo Estimation of Action Values (Q) 使用蒙特卡罗方法进行动作值估计

- 当环境的动态模型不可用时，蒙特卡罗方法通过直接从经验中学习变得非常有用
  - 目标是学习动作值函数 Q^*
- $Q^\pi(s, a)$ 表示在状态 s 下执行动作 a 后，遵循策略 π 的平均回报
- 如果每个状态-动作对都被访问过，蒙特卡罗方法会渐近收敛到真实的 $Q^\pi(s, a)$ 值
- 每个 状态-动作对 都有非零的概率作为起始对

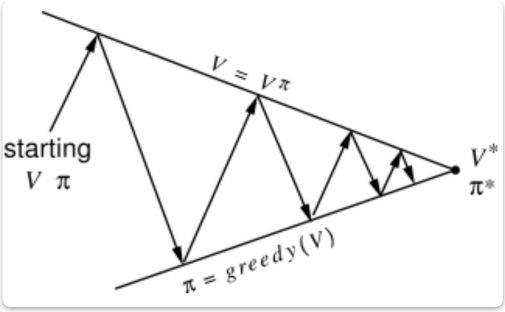
蒙特卡罗控制（Monte Carlo Control）

- 蒙特卡罗策略迭代（MC policy iteration）：
  - 使用蒙特卡罗方法进行策略评估，然后进行策略改进
  - 这个过程反复进行，直到策略收敛到最优策略
- 策略改进步骤（Policy improvement step）：
  - 相对于值函数（或动作值函数）采取贪心策略
  - 具体来说，在每次策略改进时，选择能够最大化当前值函数的动作



蒙特卡罗控制方法的收敛性 Convergence of Monte Carlo Control

- 蒙特卡罗方法在模型不可用时最有用
- 策略评估与策略改进之间的交互
- 通过迭代改进直至收敛



贪心策略满足策略改进的条件

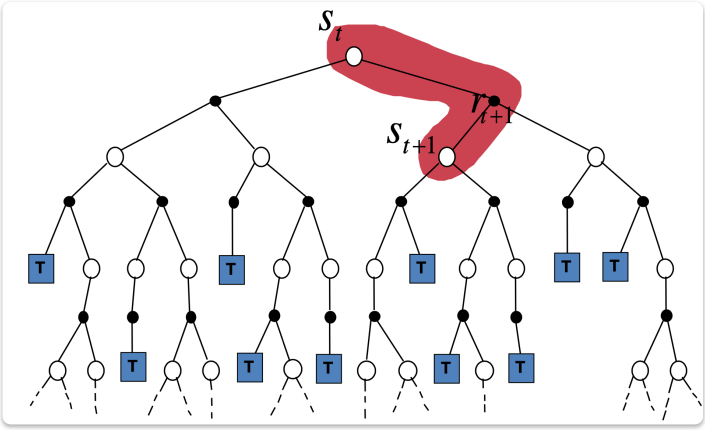
$$
\begin{aligned}
Q^{k_\pi}\left(s, \pi_{k+1}(s)\right) &= Q^{k_\pi}\left(s, \underset{a}{\operatorname{argmax}}\, Q^{\pi_k}(s,a)\right) \\
&= \max_a Q^{\pi_k}(s,a) \\
&\geq Q^{\pi_k}\left(s, \pi_k(s)\right) \\
&\geq V^{\pi_k}(s)
\end{aligned}
$$

新的策略必须比旧的策略更优或至少相同，并且假设有无限多的情节用于蒙特卡罗策略评估

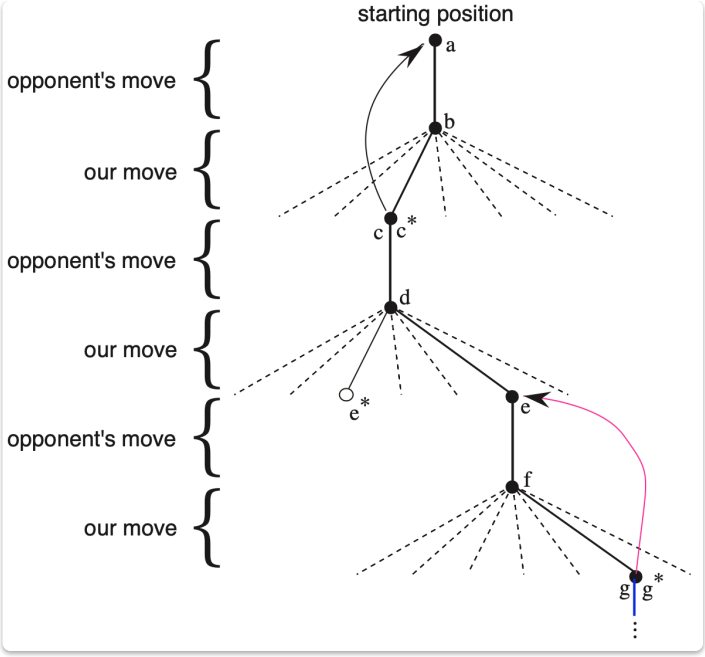
时间差分方法（Temporal Difference Method）

- 动态规划的更新规则需要了解环境的动态变化
- 典型的方法是使用时间差分方法来克服这个问题，例如Q-learning
- 查看一个状态的当前估计值与下一个状态的贴现值之间的差异以及收到的奖励

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



例子 tic-tac-toe



$$V(a) \leftarrow V(a) + \alpha[V(c) - V(a)]$$
$$V(e) \leftarrow V(e) + \alpha[V(g) - V(e)]$$

其中，$r_{t+1} = 0$，$\gamma = 1$

总结

动态规划（Dynamic Programming）

特点：

1. 要求环境模型：动态规划方法需要对环境的完全了解，包括状态转移概率和奖励函数
2. 迭代更新：通过策略评估和策略改进的迭代过程来求解最优策略和值函数
3. 计算复杂度：通常计算复杂度较高，因为需要对所有状态进行全局更新

例子：

- 值迭代（Value Iteration）
- 策略迭代（Policy Iteration）

蒙特卡罗方法（Monte Carlo）

特点：

1. 无需环境模型：蒙特卡罗方法通过采样经验数据（即通过模拟生成多个情节）进行学习
2. 基于情节：仅在一个完整情节结束后更新值函数，因此适用于终止任务
3. 期望回报：利用多个情节中的平均回报来估计值函数

例子：

- 首次访问蒙特卡罗（First-Visit Monte Carlo）
- 每次访问蒙特卡罗（Every-Visit Monte Carlo）

时序差分方法（Temporal Difference）

特点：

1. 无需环境模型：类似于蒙特卡罗方法，时序差分方法不需要环境的完全知识
2. 在线更新：可以在每个时间步进行更新，而不需要等待整个情节结束
3. 结合特性：结合了动态规划的引导和蒙特卡罗方法的样本效率

例子：

- TD(0)
- Q学习（Q-Learning）
- SARSA

具体区别：

1. 需求的环境信息：
   - 动态规划：需要完全的环境模型（转移概率和奖励函数）
   - 蒙特卡罗方法：不需要环境模型，通过模拟完整情节学习
   - 时序差分方法：不需要环境模型，可以在每个时间步进行更新
2. 更新方式：
   - 动态规划：全局更新，迭代计算所有状态的值函数
   - 蒙特卡罗方法：基于完整情节进行更新，适用于终止任务
   - 时序差分方法：在线更新，每个时间步进行更新，适用于持续任务
3. 效率和复杂度：
   - 动态规划：计算复杂度高，适用于小规模问题
   - 蒙特卡罗方法：需要大量情节，收敛较慢
   - 时序差分方法：更新及时，通常效率较高