lec09

Sarsa 与 Q-learning

时序差分预测（TD Prediction）

简单的每次访问蒙特卡罗方法（Simple every-visit Monte Carlo method）：

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

其中，R_t 是时间 t 之后的实际回报，称为 target

最简单的时序差分方法，TD(0)（The simplest TD method, TD(0)）：

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

其中，$r_{t+1} + \gamma V(s_{t+1})$ 是回报的估计值（TD误差）

时序差分（TD）方法中的引导和采样（Bootstrap and Sample）

引导（Bootstrapping）

- 定义：基于其他估计值更新估计值
- 方法比较：
  - 蒙特卡罗方法（MC）：不进行引导
  - 动态规划方法（DP）：进行引导
  - 时序差分方法（TD）：进行引导

采样（Sampling）

- 定义：更新过程中不涉及期望值（涉及期望就不是采样）
- 方法比较：
  - 蒙特卡罗方法（MC）：进行采样
  - 动态规划方法（DP）：不进行采样
  - 时序差分方法（TD）：进行采样

时序差分（TD）学习的优势

1. TD方法不需要环境的模型，仅需经验
2. TD方法（但MC方法不行）可以完全增量化：
   - 你可以在不知道最终结果前学习：
     - 更少的内存需求
     - 更少的峰值计算量
   - 你可以在没有最终结果的情况下学习：
     - 可以从不完整的序列中学习

强化学习方法的分类

Model of the environment?

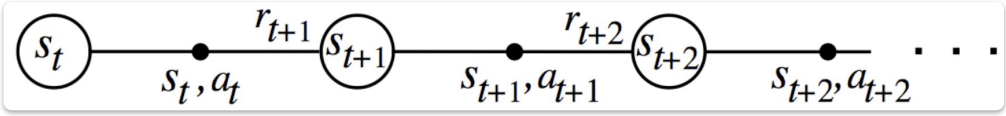
Bootstrap?	YES	NO
YES	Dynamic programming	Temporal Difference (TD)
NO		Monte Carlo Methods

1. 是否有环境模型（Model of the environment）：
   - YES（有环境模型）：可以使用环境的动态来帮助估计值
   - NO（无环境模型）：只能基于经验进行学习
2. 是否使用引导（Bootstrap）：
   - YES（使用引导）：使用当前估计值来更新值
   - NO（不使用引导）：完全基于样本进行更新

- 动态规划（Dynamic Programming，DP）：有环境模型，并且使用引导
- 时序差分（Temporal Difference，TD）：无环境模型，但使用引导
- 蒙特卡罗方法（Monte Carlo Methods，MC）：无环境模型，不使用引导

学习动作-状态值函数（Learning an action-state value function）

1. 估计当前行为策略 π 的 Q 值函数 Q^{π}：



2. 更新公式：
   - 在每次从非终止状态的转换后，执行以下更新：

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

   其中，：
   - s_t 是当前状态
   - a_t 是当前动作
   - r_{t+1} 是执行动作 a_t 后的即时奖励
   - γ 是折扣因子
   - α 是学习率
3. 终止状态处理：
   - 如果 s_{t+1} 是终止状态，则 $Q(s_{t+1}, a_{t+1}) = 0$

SARSA算法（State-Action-Reward-State-Action）在策略内（on-policy）时序差分控制中的应用

SARSA是一个基于时序差分的强化学习算法，用于学习状态-动作值函数。该算法通过策略内方法更新策略，即在评估策略的同时进行策略改进

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until s is terminal

SARSA: State Action Reward State Action

- 初始化 $Q(s,a)$ 值函数，可以随意初始化。
- 对每个情节重复以下步骤：
  1. 初始化状态 s。
  2. 从状态 s 中选择动作 a，使用由 Q 导出的策略（例如，ϵ-贪婪策略）。
  3. 对每个情节步骤重复以下操作，直到到达终止状态：
     - 执行动作 a，观察奖励 r 和下一个状态 s'。
     - 从状态 s' 中选择动作 a'，使用由 Q 导出的策略。
     - 更新 $Q(s,a)$：

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$$

     - 将 s 更新为 s'，将 a 更新为 a'。

SARSA代表了算法在更新过程中的五个元素：状态 s，动作 a，奖励 r，下一个状态 s'，以及下一个动作 a'

Q-learning：异策略（off-policy）的时序差分（Temporal Difference, TD）控制方法

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)\right]$$

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
    $s \leftarrow s'$;
  until s is terminal

Why off-policy?

- 初始化 $Q(s,a)$ 值函数，可以随意初始化。
- 对每个情节重复以下步骤：
  1. 初始化状态 s。
  2. 对每个情节步骤重复以下操作，直到到达终止状态：
     - 从状态 s 中选择动作 a，使用由 Q 导出的策略（例如，ϵ-贪婪策略）。
     - 执行动作 a，观察奖励 r 和下一个状态 s'。
     - 更新 $Q(s,a)$：

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

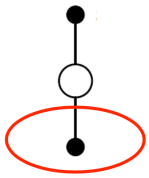
     - 将 s 更新为 s'。

为什么是异策略（off-policy）：
Q-learning是异策略的，因为它使用的行为策略（behavior policy）和目标策略（target policy）不同

- 行为策略是用于选择当前动作 a 的策略，例如 ϵ-贪婪策略
- 目标策略是用于更新 Q 值的策略，即找到使 $Q(s',a')$ 最大化的动作 a'

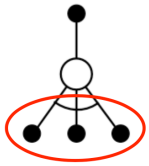
On-policy vs. Off-policy

Sarsa:



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

Q-learning:



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

举例说明

假设在一个简单的网格世界中，智能体需要学习如何从起始点到达目标点。在某一状态 s 下，有两个可能的行动 $a1$ 和 $a2$，其 Q 值分别为 $Q(s,a1) = 5$ 和 $Q(s,a2) = 10$。

- **Q-learning**：
  - 即使当前策略选择了行动 $a1$，Q-learning 在更新时仍然考虑到所有可能的行动，并选择 Q 值最大的行动进行更新。
  - 更新公式：$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
  - 由于 Q-learning 考虑到所有可能的行动，它能够更快速地识别和更新到最优策略。
- **SARSA**：
  - SARSA 更新时只考虑当前策略下的行动。
  - 更新公式：$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$
  - SARSA 必须依赖当前策略进行更新，这使得它在策略探索和更新上需要更多的时间和步骤。

Sarsa和Q-learning两种算法收敛到最优解的条件

1. Sarsa：
   - 收敛较困难，但在某些假设下可以收敛：
     - 所有状态-动作对都被无限次访问
     - 策略在极限情况下趋近于贪婪策略
2. Q-learning：
   - 由于是异策略，收敛相对更容易：
     - 在某些假设下可以收敛：
       - 所有状态-动作对被无限次访问
       - 步长参数 α 逐渐减小（但不能减小得太快）