# lec10

# 奖励塑形、Actor-Critic 方法、策略梯度方法、多步时序差分（TD）学习、Eligibility Traces（资格迹）以及处理大状态和动作空间的函数近似（Function Approximation）

## 奖励塑形（Reward Shaping）

奖励塑形是一种在强化学习中通过引入额外的奖励来加速智能体学习的方法。通过引入领域知识，在每个时间步长内提供额外的奖励，引导智能体更快地学习

奖励塑形的额外奖励函数是基于潜在函数（potential function）来定义的，它不会改变最优策略，只会加速学习过程

*Q-learning 更新公式：*

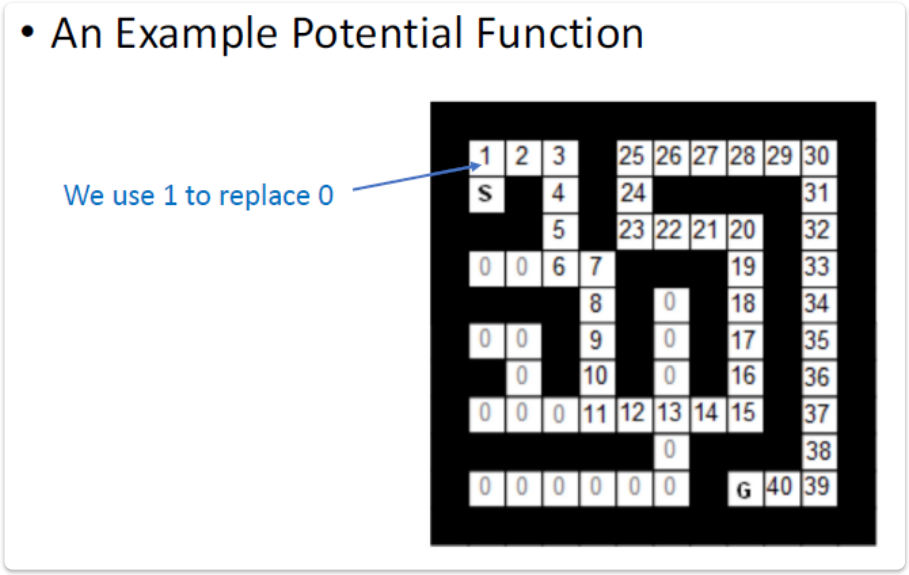
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

*奖励塑形：*

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

其中

- $F(s, s') = \gamma(\Phi(s') - \Phi(s))$：基于潜力的奖励塑形：通过定义潜力函数来确保最优策略保持不变
  - $\Phi(s)$ 是状态 $s$ 的潜力
  - $\gamma$ 是折扣因子




## Actor-Critic 方法

- Actor（行动者）：负责选择动作，它根据策略 $\pi_\theta$ 选择动作，优化目标是最大化累计奖励

- 损失函数：

$$L_{actor} = -\sum_{t=1}^{T} \log \pi_\theta(a_t|s_t)[G(s_t, a_t) - V_\theta^\pi(s_t)]$$

其中，$G(s_t, a_t) - V_\theta^\pi(s_t)$ 表示优势函数（Advantage），用来衡量当前动作相对于平均水平的好坏

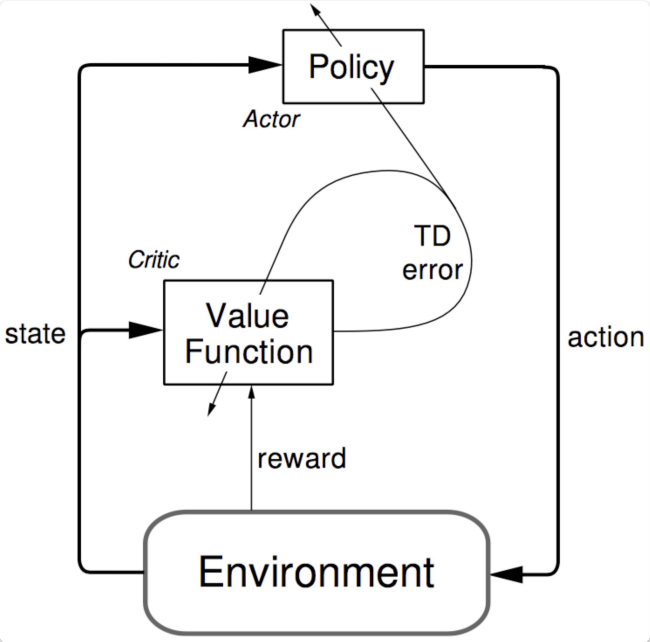
- Critic（评论者）：负责评估动作的价值，通过计算值函数 $V_\theta^\pi$，帮助 Actor 更新策略
  - 损失函数：

$$L_{critic} = L_\delta(G, V_\theta^\pi)$$

- 优势函数（Advantage）：

$$G_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$$

其中，$\gamma \in (0, 1)$




**例子：走迷宫**

假设我们有一个智能体（agent）需要在一个 5x5 的迷宫中找到出口。迷宫中的每个格子是一个状态 $s$，智能体可以采取四个动作之一：向上、向下、向左或向右。每次移动到一个新的格子，智能体都会收到一个奖励 $r$。在这个例子中，出口的奖励是 +10，其他所有格子的奖励是 -1（表示时间和步骤的消耗）。

**1. Actor-Critic 方法的设置**

- **Actor**：Actor 是策略网络，负责决定在每个状态下采取哪种行动。它输出一个概率分布，表示在当前状态下采取每个可能行动的概率。
- **Critic**：Critic 是价值网络，负责评估当前策略的好坏。它输出一个值函数 $V(s)$，表示在状态 $s$ 下执行当前策略所期望的累计奖励。

**2. 更新过程**

- **初始化**：初始化策略网络（Actor）和价值网络（Critic）的参数。
- **执行策略**：从初始状态 $s_0$ 开始，按照 Actor 提供的概率分布选择动作 $a_t$，并执行该动作。
- **观察结果**：执行动作后，观察下一个状态 $s_{t+1}$ 和即时奖励 $r_t$。
- **Critic 评估**：Critic 计算 TD 误差（时间差分误差）：

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

- **更新 Critic**：使用 TD 误差更新 Critic 的参数：

$$L_{critic} = \delta_t^2$$

- **更新 Actor**：使用 TD 误差更新 Actor 的参数：

$$L_{actor} = -\log \pi_\theta(a_t|s_t)\delta_t$$

  其中 $\pi_\theta(a_t|s_t)$ 是策略网络在状态 $s_t$ 选择动作 $a_t$ 的概率。
- **迭代**：重复上述步骤，直至智能体找到出口或者达到最大步数。

# 策略梯度方法（Policy Gradient Methods）- Learning Without Value Functions

策略梯度方法的主要思想是直接优化策略，使得在给定环境下获得的累计奖励最大化。这些方法不需要通过价值函数间接评估策略的好坏，而是直接在策略空间中进行优化

- 传统方法的共性：
  - 动态规划（Dynamic Programming, DP）、蒙特卡罗方法（Monte Carlo, MC）、时序差分方法（Temporal Difference, TD，如 Sarsa 和 Q-learning）等方法都有一个共同点：它们都使用价值函数
  - 策略 $\pi$ 是基于价值函数的
- 策略梯度方法的不同点：
  - 策略梯度方法不使用价值函数进行学习
  - 这些方法通过直接操作策略来进行学习，而不是先寻找最优的价值函数

## 公式

目标函数 $J(\theta)$ 表示策略在有限时间范围内的期望未折扣回报，假设策略 $\pi$ 在几乎所有地方都是可微的（例如，神经网络）

*梯度如下：*

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)A^{\pi_\theta}(s_t,a_t)\right]$$

其中，$A^{\pi_\theta}(s_t,a_t)$ 表示优势函数

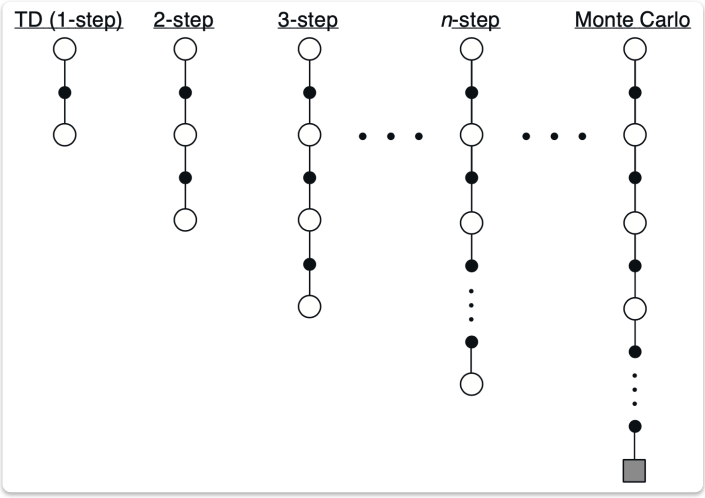
*参数更新规则：*

$$\theta_{t+1} = \theta_t + \Delta\theta_t, \quad \Delta\theta_t = G_t\nabla_\theta \log \pi(a_t|s_t)$$

# 多步时序差分（Multi-Step TD）

在更长的时间跨度内备份值而不是每个单步时间间隔后进行备份

- 传统的 TD 方法在每个单步时间间隔后备份值
- 多步TD方法可以在更长的时间跨度内备份值




$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 \max_a Q(s_{t+2},a) - Q(s_t,a_t)\right]$$

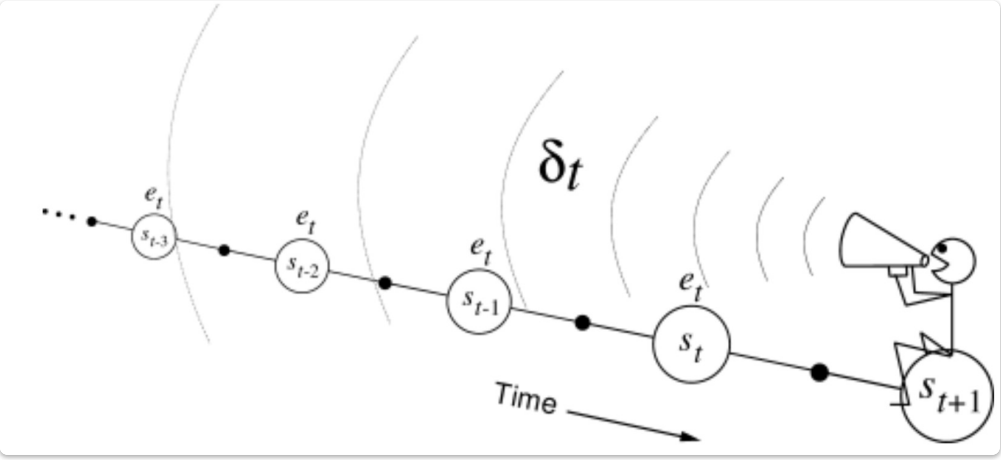
其中，$r_{t+1} + \gamma r_{t+2} + \gamma^2 \max_a Q(s_{t+2},a)$ 是两步

# Eligibility Traces（资格迹）

n 步 TD（Temporal Difference，时序差分）方法中最著名的实现

1. 定义：
   - 资格迹是 n 步 TD 方法的一种重要实现
   - 状态-动作对 对于未来的奖励是有资格的，最近的状态将获得更多的奖励
2. 机制：
   - 保持状态-动作对的踪迹或历史
   - 这些踪迹会随着时间衰减

资格迹方法通过跟踪过去的状态-动作对，并逐渐衰减它们的重要性，帮助算法更有效地学习和调整策略

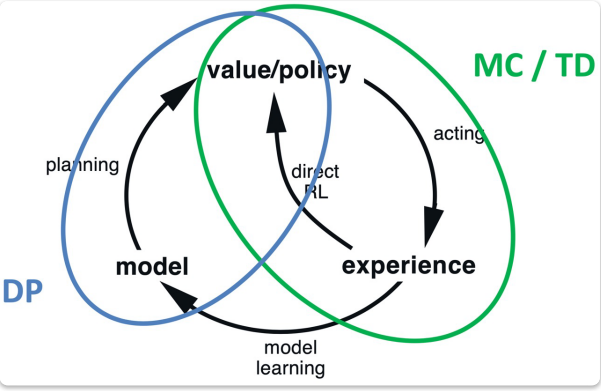



# 基于模型的时序差分（Model-based TD）

1. 计划（Planning）：
   - 计划是指任何采用模型作为输入并生成或改进与建模环境交互策略的计算方法
2. 动态规划（Dynamic Programming）：
   - 动态规划是一种计划方法，利用环境模型进行计划
3. 蒙特卡罗方法（Monte Carlo）和时序差分方法（TD）：
   - 蒙特卡罗方法和时序差分方法不是计划方法，不依赖于环境模型

利用环境模型的意思是我们拥有环境的数据，并利用环境数据进行学习
而没有利用环境数据进行学习的蒙特卡罗方法和TD方法显然不是基于模型的时序差分

## 图示




这是一个关于基于模型的时序差分（Model-based Temporal Difference, TD）方法的图示，展示了学习和计划在强化学习中的关系和相互作用。图中包含了动态规划（DP）和蒙特卡罗方法（MC）、时序差分方法（TD）在模型中的位置和相互关系。

## 图示解读：

- Model（模型）：表示环境的模型，用于模拟和预测环境的动态
- Experience（经验）：通过与环境交互获得的经验数据
- Value/Policy（价值/策略）：表示价值函数或策略，用于指导智能体的行为

## 三个主要方法：

1. 动态规划（DP）：

- Planning（计划）：使用环境模型进行计划，通过模型推断未来状态和奖励，从而更新价值函数或策略
- Model：依赖于准确的环境模型
- Value/Policy：通过计划更新价值函数或策略

2. 蒙特卡罗方法（MC）和时序差分方法（TD）：
    - Acting（行动）：基于当前的策略与环境交互，收集经验数据
    - Experience：直接利用经验数据进行学习，不依赖环境模型
    - Value/Policy：通过与环境的实际交互数据更新价值函数或策略

### 结合：

- Model Learning（模型学习）：通过经验数据学习环境模型
- Direct RL（直接强化学习）：通过直接使用经验数据更新价值函数或策略

### 总结：

- 动态规划主要依赖于环境模型进行计划，从而更新价值函数或策略
- 蒙特卡罗方法和时序差分方法通过直接与环境交互获取经验数据进行学习，不依赖环境模型
- 基于模型的TD方法结合了模型学习和直接强化学习，通过利用经验数据学习环境模型，并结合计划和直接学习来优化智能体的行为

## 函数近似（Function Approximation）

- 表格表示法的限制：目前，我们使用表格表示法来表示价值函数。但是当状态和动作空间非常大时，这种方法变得不可行
- 函数近似：可以使用函数逼近器（如高斯过程、神经网络）来泛化到大规模或连续的状态和动作空间。这些方法可以有效地处理大规模的问题，克服表格方法的限制

函数近似通过使用如高斯过程和神经网络的方法，使得在大规模或连续的状态和动作空间中进行泛化和学习成为可能，克服了表格表示法的局限性