有限马尔可夫决策过程

基本概念

多臂老虎机仅涉及**评价性反馈**，即动作的即时奖励，估计每个动作 a 的价值 $q_*(a)$。**有限马尔可夫决策过程（Finite MDP）**引入了**关联性因素**，即在**不同状态（情境）下选择不同动作**，动作不仅影响**即时奖励**，还通过改变环境状态影响**未来的奖励**，因此涉及**延迟奖励**以及**长期与短期奖励之间的权衡。**是**强化学习中序贯决策问题的经典数学建模方式**，扩展了多臂老虎机问题。在MDP中需要更精细的价值估计：

- 在状态 s 下动作 a 的最优价值：

$$q_*(s,a)$$

- 状态 s 的最优价值（在最优策略下）：

$$v_*(s)$$

这些状态相关的价值函数是将**长期后果归因于具体动作选择**的关键工具。

智能体-环境交互接口

实体	定义
智能体（Agent）	进行学习及实施决策的机器
环境（Environment）	智能体之外所有与其相互作用的事物

智能体与环境之间持续进行交互，智能体选择动作，环境对这些动作做出响应，并向智能体呈现新的情境。环境还会产生**奖励**，智能体的目标就是通过选择动作，从长远来看最大化所获得的奖励总量。

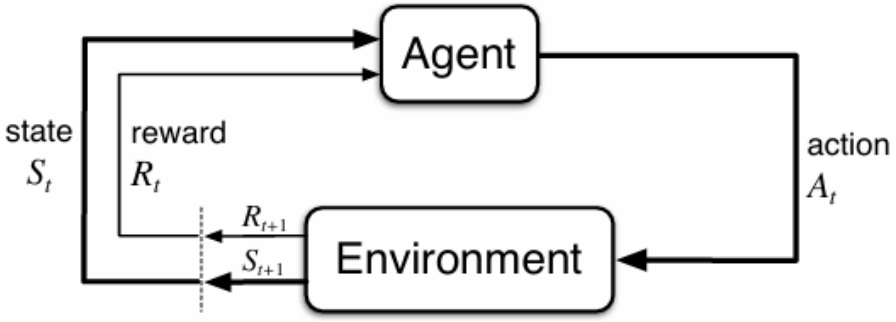


图3.1：马尔可夫决策过程中智能体与环境的交互。

交互流程

在离散的时间步$t = 0, 1, 2, \ldots$内，智能体与环境持续交互，每个时间步发生以下事件序列：

$$S_t \xrightarrow{\text{agent}} A_t \xrightarrow{\text{environment}} R_{t+1}, S_{t+1}$$

即：

1. 智能体观察当前状态 $S_t \in \mathcal{S}$
2. 选择动作 $A_t \in \mathcal{A}(s)$
3. 环境响应：
    - 给出奖励 $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$

- 进入新状态 S_{t+1}

整个交互过程形成一个序列，称为**轨迹**：

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots \tag{3.1}$$

有限MDP

状态集 $\mathcal{S}$、动作集 $\mathcal{A}$、奖励集 $\mathcal{R}$：有限个可能取值都只有**有限个元素。**

随机变量 R_t 和 S_t 具有明确定义的离散概率分布，且这些分布仅依赖于**前一个状态和动作。**

对于这些随机变量的特定取值 $s' \in \mathcal{S}$ 和 $r \in \mathcal{R}$，在给定前一个状态 s 和动作 a 的条件下，它们在时间 t 出现的概率为：

$$p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\} \tag{3.2}$$

> **Note**
>
> 符号 $\doteq$ 表示"定义为"
>
> 同时有

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s) \tag{3.3}$$

在**马尔可夫**决策过程中，条件概率 $p(s', r|s, a)$ 决定了所有未来动态。S_t 和 R_t 的分布**仅依赖于前一状态 S_{t-1} 和动作 A_{t-1}**，与更早的历史无关。这要求**状态必须包含所有对未来有影响的信息。**马尔可夫性质是对"状态"的限制，而非过程本身；若状态满足此性质，则称其具有**马尔可夫性。**

相关计算量

状态转移概率

$$p(s'|s, a) \doteq \Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r|s, a) \tag{3.4}$$

状态-动作对的期望奖励

$$r(s, a) \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a) \tag{3.5}$$

给定下一状态的期望奖励

$$r(s, a, s') \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \cdot \frac{p(s', r|s, a)}{p(s'|s, a)} \tag{3.6}$$

MDP框架

框架中**时间步**的含义不必对应真实时间间隔，可表示任意**决策阶段的连续序列。**

动作可以是：低层次控制信号（如电机电压）也可以是高层次决策（如"去吃午饭"）或者心理或计算性动作（如"集中注意力"、"思考某个问题"）。

状态可基于原始传感器数据、抽象符号描述（如物体位置）、记忆信息或者主观心理状态（如"不确定"、"惊讶"）。

一般来说, 动作可以是任何我们想要做的决策, 而状态则 可以是任何对决策有所帮助的事情。

智能体与环境

任何不能被智能体任意改变的事物都属于环境。同时并不假设智能体对环境一无所知，**智能体-环境边界代表的是智能体绝对控制能力的极限，而非其知识的极限。** 智能体可以了解环境机制，但不能随意更改任务目标。可根据任务需求设置不同边界，复杂系统中可存在多个智能体，各自有独立边界。高层智能体的输出可以成为低层智能体的状态输入。

MDP将学习目标导向行为的问题简化为**三个核心信号的交互**：

信号	含义
状态 S_t	决策的基础（当前情境）
动作 A_t	智能体的选择
奖励 R_t	目标定义（标量数值）

状态和动作的具体表示方式**极大影响性能**，表示选择目前更多是**艺术而非科学**，后续的重点**在表示确定后，学习行为的一般性原则。**

实例分析

生物反应器

- **任务**：控制温度和搅拌速率以最大化化学品产出
- **动作**：目标温度、目标搅拌速率（向量）
- **状态**：传感器读数 + 符号输入（原料/目标化学品）
- **奖励**：单位时间产出速率（瞬时度量）

拾取-放置机器人

- **任务**：实现快速、平滑的抓取放置
- **动作**：施加到关节电机的电压
- **状态**：关节角度和速度的实时读数
- **奖励**：
    - 成功完成任务：+1
    - 每步根据“抖动”程度给予小负奖励（鼓励平滑）

回收机器人

- 办公室中移动机器人收集空苏打罐
- 配备传感器、机械臂、夹爪
- 电池供电，电量分为两种状态

状态空间

两种电量水平

$$\mathcal{S} = \{high, low\}$$

动作集

在每种状态下，智能体可以决定：(1) 主动搜索罐子一段时间，(2) 原地静止等待别人送来罐子，或 (3) 返回基地为电池充电。

- $\mathcal{A}(high) = \{search, wait\}$
- $\mathcal{A}(low) = \{search, wait, recharge\}$
    - 高电量时不提供充电选项

奖励机制

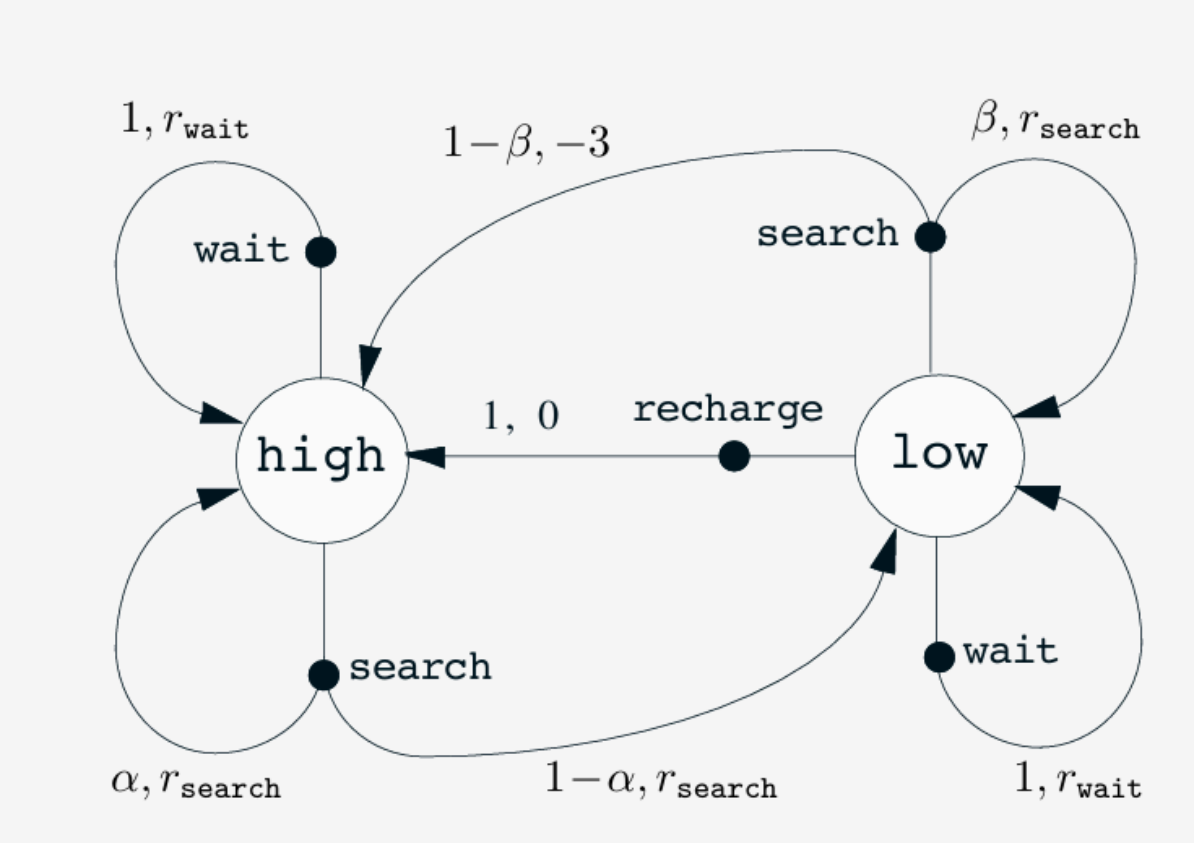
- 收集一个罐子：+1
- 电池耗尽需救援：-3
- 其他时间：奖励为0
- 充电过程中无法收集罐子
- 耗尽电池的那一步也无法收集

转移动态

当前状态	动作	下一状态	转移概率	期望奖励
high	search	high	α	r_{search}
high	search	low	$1-\alpha$	r_{search}
high	wait	high	1	r_{wait}
low	search	low	β	r_{search}
low	search	**耗尽**	$1-\beta$	-3（且无法收集）
low	wait	low	1	r_{wait}
low	recharge	high	1	0（充电中无收益）

转移图表示法

- 图中两类节点：
    - **状态节点**：大空心圆，标记状态名
    - **动作节点**：小实心圆，标记动作名，连接至对应状态
- 流程：
    1. 从状态节点 s → 动作节点 (s, a)
    2. 从动作节点发出箭头 → 下一状态节点 s'
- 每条箭头标注：
    - 转移概率 $p(s'|s, a)$
    - 期望奖励 $r(s, a, s')$



目标与奖励

在强化学习中，智能体的**目标或目的**通过一种特殊信号来形式化：**奖励（Reward）**奖励由**环境**传递给智能体，是一个**标量数值**$R_t \in \mathbb{R}$。每个时间步 t，环境根据当前状态转移和动作结果生成奖励 R_t智能体的目标不是最大化**即时奖励**，而是最大化**长期累积奖励**；更准确地说，是最大化**未来奖励总和的期望值**。

奖励假说

> **所有我们所说的“目标”和“目的”，都可以被很好地理解为：最大化一个接收到的标量信号（称为奖励）的累积和的期望值。**

任何形式的目标（如行走、下棋、回收物品等），都可以通过设计合适的奖励信号来引导智能体实现。强化学习不预设“目标”的具体内容，而是将其**完全交给奖励函数来定义**。智能体本身并不“知道”任务目标，它只知道“要最大化奖励”。

尽管用单一数值表示目标看似受限，但实际上非常**灵活且强大**。

应用场景	奖励设计	目的
机器人走路	每步给予与前进速度成正比的奖励	学会高效前行
迷宫逃脱	每步给予 -1 的奖励，逃出后终止	鼓励尽快逃脱（最小化步数）
回收空罐	收集成功时 +1，其余时间 0	鼓励多收集罐子
安全行为	撞到物体或被呵斥时给予负奖励	鼓励避免危险或不当行为
下跳棋/国际象棋	获胜：+1，失败：-1，平局/中间状态：0	学会赢棋

智能体总是学习去最大化它的奖励，因此，**设定的奖励必须真正反映出我们希望达成的目标。**我们希望使智能体在最大化奖励的同时也实现我们的目标。但**奖励信号并不是用来向智能体传授"如何"实现目标的先验知识的地方，奖励信号是你向智能体传达"你想要它实现什么"的方式，而不是"你希望它如何实现"的方式。**例如对于下棋智能体**不应奖励**吃掉对方棋子、控制棋盘中心等子目标，**只应在游戏结束时**根据胜负给予奖励。因为如果对"吃子"给予奖励，智能体可能学会牺牲大局去换取吃子机会；故意制造吃子局面，即使会导致最终输棋。

回报与回合

定义

智能体的目标是**从长远来看最大化累积奖励。**现在需要对回报进行数学定义。

令 $R_{t+1}, R_{t+2}, R_{t+3}, \ldots$ 表示从时间步 $t+1$ 开始的奖励序列。我们定义时间步 t 的**回报** G_t 为这些奖励的某个函数。

最简单的情况，回报是所有奖励的总和：

$$G_t \doteq R_{t+1} + R_{t+2} + \cdots + R_T = \sum_{k=1}^{T-t} R_{t+k} = \sum_{k=0}^{\infty} R_{t+k+1} \cdot \mathbf{1}_{\{t+k+1 \leq T\}} \tag{3.7}$$

其中：

- T：该回合的终止时间步（随机变量）
- 当 $k > T - t$ 时，$R_{t+k+1} = 0$（已终止）

在存在自然终止状态的任务中，交互可划分为**独立的回合。**每个回合以**终止状态**结束，终止后系统重置至初始状态或初始状态分布。下一回合的开始与上一回合的结果无关。这类任务称为**回合制任务**，比如下棋等。

在回合制任务中，我们有时需要区分两个集合：

- $\mathcal{S}$：所有**非终止状态**的集合
- $\mathcal{S}^+$：所有状态的集合，包括终止状态

持续性任务

持续性任务没有自然终止点，交互无限持续，如永不停止的机器人服务系统。此时不适用使用奖励综合定义汇报(任务的汇报会是无穷大)，因此使用折扣回报。

引入**折扣率** $\gamma \in [0, 1]$，定义：

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{3.8}$$

越远的奖励被赋予越小的权重，R_{t+k+1} 的当前价值为 $\gamma^k R_{t+k+1}$。

γ 值	含义
$\gamma = 0$	**短视型**（myopic）：只关心 R_{t+1}，忽略未来
$\gamma \to 1$	**远见型**：高度重视长期奖励
$0 < \gamma < 1$	平衡当前与未来奖励

> Important

证明：

> 若 $|R_k| \leq r_{\max}$（即奖励有界），且 $\gamma < 1$，则折扣回报
>
> $$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$
>
> 是**收敛的**，且其绝对值有界，即 $|G_t| < \infty$

前提

奖励序列有界：存在常数 $r_{\max} > 0$，使得对所有 k，都有

$$|R_{t+k+1}| \leq r_{\max}$$

（例如：奖励总是在 $[-10, 10]$ 之间）。折扣率满足：

$$0 \leq \gamma < 1$$

由于 $|R_{t+k+1}| \leq r_{\max}$，我们有：

$$|G_t| = \left| \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \right| \leq \sum_{k=0}^{\infty} \left| \gamma^k R_{t+k+1} \right| = \sum_{k=0}^{\infty} \gamma^k |R_{t+k+1}|$$

再利用 $|R_{t+k+1}| \leq r_{\max}$，得到：

$$|G_t| \leq \sum_{k=0}^{\infty} \gamma^k \cdot r_{\max} = r_{\max} \sum_{k=0}^{\infty} \gamma^k$$

我们知道，当 $0 \leq \gamma < 1$ 时，几何级数收敛：

$$\sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}$$

所以：

$$|G_t| \leq r_{\max} \cdot \frac{1}{1-\gamma}$$

这个上界是一个**有限的正数**（因为 $\gamma < 1$，分母不为零）。

因此，G_t **是绝对收敛的，且其值有限。**

由公式 (3.8) 可得：

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned} \tag{3.9}$$

这个式子 对所有 $t < T$ 成立，只要定义 $G_T = 0$

若每步奖励为 +1，且 $\gamma < 1$，则：

$$G_t = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma} \tag{3.10}$$

练习1

> 请修改式 (3.3) 使其适于分幕式任务。

在持续性任务（continuing tasks）中，式 (3.3) 定义为：

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s) \tag{3.3}$$

此式假设：

- 所有状态 $s \in \mathcal{S}$ 均可执行动作（即 $\mathcal{A}(s) \neq \emptyset$），
- 不存在终止状态，过程无限持续。

在**分幕式任务（episodic tasks）**中：

- 状态空间包含**非终止状态**（nonterminal states）和**终止状态**（terminal states）。
- 在终止状态 s_{term}，动作集为空（$\mathcal{A}(s_{\text{term}}) = \emptyset$），幕结束。
- 原式中 $\forall s \in \mathcal{S}$ 包含终止状态，导致左侧 $p(s', r|s, a)$ 未定义（因无动作 a）。

为使式 (3.3) 适用于分幕式任务：

- 将状态量化范围从 $\forall s \in \mathcal{S}$ 收窄为 $\forall s \in \mathcal{S}^+$，
- 其中 $\mathcal{S}^+ \subset \mathcal{S}$ 表示**所有非终止状态集合**，
- 转移目标 s' 仍可为任意状态（包括终止状态），因为从非终止状态可能转移到终止状态。

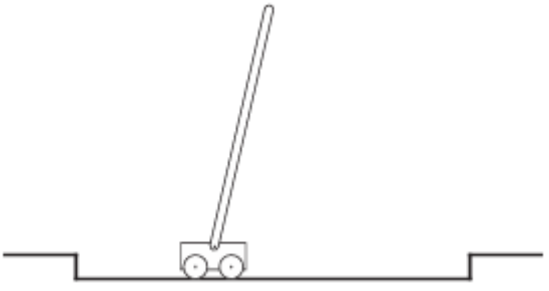
修改后等式

$$\boxed{\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1, \quad \forall s \in \mathcal{S}^+, a \in \mathcal{A}(s)} \tag{3.3e}$$

其中：

- $\mathcal{S}$：所有状态集合（含终止状态），
- $\mathcal{S}^+$：非终止状态集合（$\mathcal{S}^+ = \mathcal{S} \setminus \mathcal{S}_{\text{terminal}}$），
- $\mathcal{R}$：奖励集合，
- $p(s', r|s, a)$：状态转移与奖励联合概率函数。

案例1



考虑**杆平衡任务**（Cart-Pole）：

- **失败条件**：杆倾角过大 或 小车移出轨道
- **重置机制**：失败后杆重置回垂直位置，任务重新开始

我们需要分析两种建模方式下的回报计算：

方案一：分幕式任务（Episodic Task）

- 每步未失败 → 奖励 **+1**
- 失败时 → 无额外奖励
- 目标：最大化单幕生存时间

方案二：持续性任务（Continuing Task）

- 每步未失败 → 奖励 **0**
- 失败时 → 奖励 **-1**
- 目标：最小化长期失败频率

问题：

1. 在**分幕式任务**中，回报 G_t 如何计算?
2. 与**持续性任务**中的回报有何本质区别

在**标准分幕式任务**中，通常采用**生存奖励**（survival reward）机制：

- 每步成功平衡 → 奖励 **+1**
- 失败时 → 奖励 **0**

设：从时间步 t 开始，到失败发生在时间步 $t+K$（即 K 步后失败）

则回报为：

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{K-1} R_{t+K}$$

由于 $R_{t+1} = R_{t+2} = \cdots = R_{t+K-1} = 1$ 且 $R_{t+K} = 0$（失败）：

$$G_t = 1 + \gamma + \gamma^2 + \cdots + \gamma^{K-2} = \frac{1-\gamma^{K-1}}{1-\gamma} \quad (\text{当 } \gamma < 1)$$

> **关键理解**：
>
> - 当 $\gamma = 1$（无折扣）时，$G_t = K - 1$，即**直到失败前的步数**
> - 当 $\gamma < 1$ 时，G_t 是**折扣后的生存时间**，但仍随 K 增大而增大
> - 优化目标：**最大化** G_t ⇨ **最大化生存时间** K

因此，在标准分幕式任务中：

$$\boxed{G_t = \sum_{k=0}^{K-2} \gamma^k = \frac{1-\gamma^{K-1}}{1-\gamma}}$$

其中 K 是从时间步 t 到失败所经历的总步数（$K \geq 1$）。

> **示例**：若 $\gamma = 0.9$，生存 10 步后失败（$K = 10$）
>
> $$G_t = 1 + 0.9 + 0.9^2 + \cdots + 0.9^8 = \frac{1-0.9^9}{1-0.9} \approx 5.31$$

在**持续性任务**中，通常采用**失败惩罚**（failure penalty）机制：

- 每步成功平衡 → 奖励 **0**
- 失败时 → 奖励 **-1**

设：

- 第一次失败发生在 $t+K$（经历 K 步）
- 重置后，第二次失败发生在 $t+K+K'$（再经历 K' 步）
- 依此类推...

则回报为：

$$G_t = \underbrace{0 + \cdots + \gamma^{K-1}(-1)}_{\text{第一次失败}} + \underbrace{\gamma^K \cdot 0 + \cdots + \gamma^{K+K'-1}(-1)}_{\text{第二次失败}} + \cdots$$

即：

$$G_t = -\gamma^{K-1} - \gamma^{K+K'-1} - \gamma^{K+K'+K''-1} - \cdots$$

因此，在持续性任务中：

$$\boxed{G_t = -\sum_{i=1}^{\infty} \gamma^{T_i - t - 1}}$$

其中 T_i 是第 i 次失败发生的时间步。

> **关键理解**：
>
> - G_t 为负值，且**绝对值越小越好**（越接近 0 越好）
> - G_t 的绝对值反映**长期失败频率**：失败越频繁，$|G_t|$ 越大
> - 优化目标：**最大化** G_t（即**最小化** $|G_t|$）⇨ **降低长期失败频率**

> **示例**：若 $\gamma = 0.9$，且每次失败后都能平衡 10 步（$K = K' = K'' = \cdots = 10$）
>
> $$G_t = -0.9^9 - 0.9^{19} - 0.9^{29} - \cdots = -0.9^9(1 + 0.9^{10} + 0.9^{20} + \cdots) \approx -0.39$$

在标准分幕式任务中（使用生存奖励 +1/步），从时间步 t 开始的回报为：

$$\boxed{G_t = \sum_{k=0}^{K-2} \gamma^k = \frac{1 - \gamma^{K-1}}{1 - \gamma}}$$

其中 K 是从 t 到失败所经历的总步数。

与持续性任务的本质区别在于：

- 分幕式任务使用**生存奖励**（+1/步），回报为**正值**，直接反映**单幕生存时间**
- 持续性任务使用**失败惩罚**（-1/失败），回报为**负值**，反映**长期失败频率**
- 两种设置下优化目标一致（减少失败），但**数学表达和直观解释不同**，导致算法对策略的评估方式有本质差异。

练习2

考虑一个迷宫逃脱任务：

- **奖励规则**：
  - 逃脱迷宫 → 奖励 **+1**
  - 其他情况 → 奖励 **0**
- 任务被建模为**分幕式任务**（episodic task）：
  - 每幕从起点开始，到逃脱结束
  - 目标：最大化预期总收益（无折扣，$\gamma = 1$）：

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T = \sum_{k=1}^{T-t} R_{t+k}$$

    其中 T 是逃脱时间

问题现象：机器人表现不再提升——它可能能逃脱，但效率不高（绕远路、耗时长），或根本没学会高效路径。

核心问题：

1. 可能出了什么问题？
2. 是否有效地向机器人传达了你想要它实现的目标？

问题根源：在当前奖励设计（逃脱+1，其他0）和无折扣（$\gamma = 1$）条件下，**所有成功逃脱的策略具有相同的回报值**（$G_t = 1$），无论路径长短。数学上：

$$\boxed{G_t^{\pi_{\text{opt}}} = G_t^{\pi_{\text{sub}}} = 1}$$

机器人学会"最终逃脱"就达到了回报最大化，**没有动力去寻找更短路径**，导致效率低下或学习停滞。

目标传达有效性：你没有有效传达目标。你希望机器人"高效逃脱"，但回报函数只鼓励"最终逃脱"，无论效率如何。这是典型的**奖励设计偏差**——智能体只优化明确指定的回报函数，而非你的隐含期望。

解决方案：修改奖励函数，使更短路径获得更高回报：

1. **每步小负奖励**（推荐）：$R = -0.01$（每步），$R = +1$（逃脱）$\rightarrow G_t = 1 - 0.01(K-1)$
2. **使用折扣因子**：$\gamma < 1$（如 $\gamma = 0.99$）$\rightarrow G_t = \gamma^{K-1}$
3. **势能塑形奖励**：根据状态接近出口的程度提供中间奖励

这些修改确保**回报函数与真实目标对齐**，驱动机器人学习高效策略。其中，**每步小负奖励是最简单有效的解决方案**，它直接将"最小化逃脱步数"编码到回报函数中。

给定：

- 折扣因子 $\gamma = 0.5$
- 终止时间步 $T = 5$
- 收益序列：$R_1 = +1, R_2 = 2, R_3 = 6, R_4 = 3, R_5 = 2$

任务：计算 $G_0, G_1, \ldots, G_5$

> **提示**：使用反向计算（从 G_5 开始向前计算）

在分幕式任务中，回报 G_t 定义为：

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T$$

或者使用递归关系（更便于计算）：

$$G_t = R_{t+1} + \gamma G_{t+1}$$

其中 $G_T = 0$（终止状态后无回报）。

逐步计算（从后向前）

1. 计算 G_5

- 在终止时间步 $T = 5$ 之后，无未来收益

$$G_5 = 0$$

2. 计算 G_4

- 只包含 R_5

$$G_4 = R_5 + \gamma G_5 = 2 + 0.5 \times 0 = 2$$

3. 计算 G_3

- 包含 R_4 和 R_5（折扣后）

$$G_3 = R_4 + \gamma G_4 = 3 + 0.5 \times 2 = 3 + 1 = 4$$

4. 计算 G_2

- 包含 R_3, R_4, R_5（适当折扣）

$$G_2 = R_3 + \gamma G_3 = 6 + 0.5 \times 4 = 6 + 2 = 8$$

5. 计算 G_1

- 包含 R_2 至 R_5

$$G_1 = R_2 + \gamma G_2 = 2 + 0.5 \times 8 = 2 + 4 = 6$$

6. 计算 G_0

- 包含 R_1 至 R_5

$$G_0 = R_1 + \gamma G_1 = 1 + 0.5 \times 6 = 1 + 3 = 4$$

给定：

- 折扣因子：$\gamma = 0.9$
- 收益序列：
  - $R_1 = 2$
  - $R_2 = R_3 = R_4 = \cdots = 7$（从 $t = 2$ 开始无限循环 7）

任务：计算 G_0 和 G_1

在强化学习中，折扣回报 G_t 定义为：

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

其中：

- γ 是折扣因子（$0 \leq \gamma < 1$）
- 当 $|\gamma| < 1$ 时，无限几何级数 $\sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}$

计算 G_1

从时间步 $t = 1$ 开始，回报 G_1 包含从 R_2 开始的所有未来奖励：

$$G_1 = R_2 + \gamma R_3 + \gamma^2 R_4 + \gamma^3 R_5 + \cdots$$

已知 $R_2 = R_3 = R_4 = \cdots = 7$，代入得：

$$G_1 = 7 + \gamma \cdot 7 + \gamma^2 \cdot 7 + \gamma^3 \cdot 7 + \cdots = 7 \cdot (1 + \gamma + \gamma^2 + \gamma^3 + \cdots)$$

这是一个首项为 1、公比为 $\gamma = 0.9$ 的**无限几何级数**：

$$G_1 = 7 \cdot \sum_{k=0}^{\infty} \gamma^k = 7 \cdot \frac{1}{1-\gamma} = 7 \cdot \frac{1}{1-0.9} = 7 \cdot \frac{1}{0.1} = 7 \cdot 10 = 70$$

$$\boxed{G_1 = 70}$$

从时间步 $t = 0$ 开始，回报 G_0 包含从 R_1 开始的所有未来奖励：

$$G_0 = R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \cdots$$

已知 $R_1 = 2$ 且 $R_2 = R_3 = R_4 = \cdots = 7$，代入得：

$$G_0 = 2 + \gamma \cdot 7 + \gamma^2 \cdot 7 + \gamma^3 \cdot 7 + \cdots$$

将后半部分提取出来：

$$G_0 = 2 + \gamma \cdot (7 + \gamma \cdot 7 + \gamma^2 \cdot 7 + \cdots) = 2 + \gamma \cdot G_1$$

已经计算出 $G_1 = 70$，所以：

$$G_0 = 2 + 0.9 \cdot 70 = 2 + 63 = 65$$

$$\boxed{G_0 = 65}$$

统一记号

上文中同时描述了分慕式任务和持续性任务，现在要为这两种任务建立统一的表达方式。

省略回合编号

在严格意义上，每个回合 i 都有自己的时间步序列。因此应使用：

- $S_{t,i}, A_{t,i}, R_{t,i}$：第 i 个回合中时间步 t 的状态、动作、奖励
- T_i：第 i 个回合的终止时间

但这会使表达变得复杂，

在实际讨论中，我们通常要么只关注**某一个特定回合**，要么陈述的是**对所有回合都成立**的结论。因此，我们可以**轻微滥用记号**，省略回合下标 i： 使用 S_t, A_t, R_t, T 等符号，隐含地表示“当前回合”的变量。

将终止视为“吸收状态”

将**回合的终止**看作进入一个特殊的**吸收状态**（absorbing state），满足一旦进入，永远停留（自循环）；每步产生**零奖励**；不再0有实际动作或状态变化.

例如假设一个回合有 3 步奖励：$R_1 = +1, R_2 = +1, R_3 = +1$，然后终止。

我们将其扩展为无限序列：

$$R_1 = +1,\ R_2 = +1,\ R_3 = +1,\ R_4 = 0,\ R_5 = 0,\ R_6 = 0,\ \dots$$

并定义状态转移图如下：

$$S_0 \xrightarrow{+1} S_1 \xrightarrow{+1} S_2 \xrightarrow{+1} \boxed{\text{Absorbing State}} \xrightarrow{0} \boxed{\text{Absorbing State}} \xrightarrow{0} \cdots$$

现在，则可以**统一使用折扣回报公式**：

$$G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3.8}$$

在回合制任务中的效果：

- 当 $t \geq T$（已终止），所有后续 $R_{t+k+1} = 0$
- 所以：

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} + \sum_{k=T-t}^{\infty} \gamma^k \cdot 0 = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$$

- 若 $\gamma = 1$，则退化为原始的有限和：$G_t = \sum_{k=t+1}^{T} R_k$

在持续性任务中的效果：

- 无需终止，直接使用无限折扣和
- 吸收状态不出现，或仅用于理论构造

策略与价值函数

强化学习的核心是**估计价值函数**（value functions），用于衡量状态或动作的“好坏”,而“好坏”由**期望回报**定义，既从某状态出发，未来能获得多少奖励。但未来奖励取决于**行为方式**，即**策略**（policy），因此，**价值函数是相对于策略定义的。**

策略

一个**策略** π 是从状态到动作概率的映射：

$$\pi(a|s) = \Pr\{A_t = a \mid S_t = s\}$$

- 表示在状态 s 下选择动作 a 的概率
- 对每个 s，$\pi(\cdot|s)$ 是动作集 $\mathcal{A}(s)$ 上的概率分布：

$$\sum_{a \in \mathcal{A}(s)} \pi(a|s) = 1, \quad \pi(a|s) \geq 0$$

强化学习方法的核心就是：**如何根据智能体的经验来改变其策略。**

练习3

给定：

- 当前状态 S_t
- 随机策略 π，其中 $\pi(a|s)$ 表示在状态 s 下选择动作 a 的概率
- 状态转移函数 $p(s', r|s, a)$（式 3.2）：

$$p(s', r|s, a) \doteq \Pr\{S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\}$$

任务：用 π 和 p 表示 R_{t+1} 的期望 $\mathbb{E}[R_{t+1}|S_t]$

需要计算**给定当前状态 S_t 时，下一步奖励 R_{t+1} 的期望值。**

关键点：

1. 智能体首先根据策略 π 从当前状态 S_t 选择动作 A_t
2. 然后环境根据状态转移函数 p 产生下一个状态 S_{t+1} 和奖励 R_{t+1}

因此，R_{t+1} 的期望需要考虑两个随机过程：

- 动作选择的随机性（由 π 决定）
- 状态转移和奖励的随机性（由 p 决定）

根据全期望公式，R_{t+1} 的条件期望可分解为：

$$\mathbb{E}[R_{t+1}|S_t = s] = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \cdot \mathbb{E}[R_{t+1}|S_t = s, A_t = a]$$

其中：

- $\pi(a|s)$ 是在状态 s 选择动作 a 的概率
- $\mathbb{E}[R_{t+1}|S_t = s, A_t = a]$ 是在给定状态 s 和动作 a 时，下一步奖励的期望

现在计算 $\mathbb{E}[R_{t+1}|S_t = s, A_t = a]$：

根据状态转移函数 $p(s', r|s, a)$ 的定义，我们有：

$$\mathbb{E}[R_{t+1}|S_t = s, A_t = a] = \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} r \cdot p(s', r|s, a)$$

代入前式，得到最终表达式：

$$\mathbb{E}[R_{t+1}|S_t = s] = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \left(\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} r \cdot p(s', r|s, a) \right)$$

最终答案

$$\boxed{\mathbb{E}[R_{t+1}|S_t] = \sum_{a \in \mathcal{A}(S_t)} \pi(a|S_t) \left(\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} r \cdot p(s', r|S_t, a) \right)}$$

或者，如果明确指定当前状态为 s：

$$\boxed{\mathbb{E}[R_{t+1}|S_t=s]=\sum_{a\in\mathcal{A}(s)}\pi(a|s)\left(\sum_{s'\in\mathcal{S}}\sum_{r\in\mathcal{R}}r\cdot p(s',r|s,a)\right)}$$

假设：

- 状态 s 有两个可能动作：a_1 和 a_2
- $\pi(a_1|s)=0.7$，$\pi(a_2|s)=0.3$
- 对于 a_1：$p(s'_1,1|s,a_1)=0.6$，$p(s'_2,2|s,a_1)=0.4$
- 对于 a_2：$p(s'_1,3|s,a_2)=1.0$

则：

- $\mathbb{E}[R_{t+1}|s,a_1]=1\times0.6+2\times0.4=1.4$
- $\mathbb{E}[R_{t+1}|s,a_2]=3\times1.0=3.0$
- $\mathbb{E}[R_{t+1}|s]=0.7\times1.4+0.3\times3.0=0.98+0.9=1.88$

价值函数

状态价值函数 $v_\pi(s)$

定义为：从状态 s 开始，遵循策略 π 的**期望回报**：

$$v_\pi(s)\doteq\mathbb{E}_\pi[G_t\mid S_t=s]=\mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\middle|S_t=s\right],\quad\forall s\in\mathcal{S}\tag{3.12}$$

- $\mathbb{E}_\pi[\cdot]$：在策略 π 下取期望
- 终止状态的价值定义为 0：$v_\pi(s_{\text{terminal}})=0$

动作价值函数 $q_\pi(s,a)$

定义为：在状态 s 执行动作 a 后，再遵循策略 π 的期望回报：

$$q_\pi(s,a)\doteq\mathbb{E}_\pi[G_t\mid S_t=s,A_t=a]=\mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\middle|S_t=s,A_t=a\right]\tag{3.13}$$

练习4

> 写出用 q_π 和 π 来表达的 v_π 公式。

$$\boxed{v_\pi(s)=\sum_{a\in\mathcal{A}}\pi(a|s)\,q_\pi(s,a)}$$

在强化学习中：

- $v_\pi(s)$ 是**状态价值函数**，表示在策略 π 下，从状态 s 开始、之后始终遵循 π 所能获得的期望回报。
- $q_\pi(s,a)$ 是**动作价值函数**，表示在状态 s 下先执行动作 a，之后遵循策略 π 所能获得的期望回报。
- $\pi(a|s)$ 是策略函数，表示在状态 s 下选择动作 a 的概率。

由于在状态 s 中，智能体会根据策略 π 以概率 $\pi(a|s)$ 选择动作 a，因此状态 s 的期望价值 $v_\pi(s)$ 应等于所有可能动作 a 对应的 $q_\pi(s,a)$ 值按其选择概率 $\pi(a|s)$ 加权求和。

换句话说，$v_\pi(s)$ 是 $q_\pi(s,a)$ 关于动作 a 的期望值，期望由策略 π 定义：

$$v_\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[q_\pi(s,a)\right] = \sum_a \pi(a|s)\, q_\pi(s,a)$$

这个公式建立了状态价值函数与动作价值函数之间的基本联系，是策略评估和值函数转换中的核心关系之一。

> 写出用 v_π 和四参数函数 p 表达的 q_π 公式。

$$\boxed{q_\pi(s,a) = \sum_{s',r} p(s',r \mid s,a)\left[r + \gamma\, v_\pi(s')\right]}$$

在强化学习中：

- $q_\pi(s,a)$：在状态 s 下执行动作 a，之后遵循策略 π 的期望回报。
- $v_\pi(s')$：从状态 s' 开始遵循策略 π 的期望回报（折扣后）。
- $p(s',r \mid s,a)$：环境的**动态函数**（dynamics function），是一个四参数概率函数，表示在状态 s 执行动作 a 后，转移到状态 s' 并获得即时奖励 r 的概率。
- γ：折扣因子，$0 \le \gamma \le 1$。

当你在状态 s 采取动作 a：

1. 环境根据动态函数 $p(s',r \mid s,a)$ 随机转移到下一个状态 s' 并给出奖励 r。
2. 你立即获得奖励 r。
3. 之后你遵循策略 π，从新状态 s' 开始，期望获得的未来回报是 $v_\pi(s')$（已包含折扣）。
4. 所以总期望回报为：$r + \gamma v_\pi(s')$。
5. 对所有可能的 s' 和 r 按其发生概率 $p(s',r \mid s,a)$ 求期望：

$$q_\pi(s,a) = \mathbb{E}\left[r + \gamma v_\pi(s') \,\middle|\, s,a\right] = \sum_{s',r} p(s',r \mid s,a)\left[r + \gamma\, v_\pi(s')\right]$$

这个公式是**贝尔曼方程**中动作价值函数的核心表达式之一，它将 q_π 用环境动态 p 和状态价值函数 v_π 表达出来，是连接模型（p）与值函数的重要桥梁。

贝尔曼方程

价值函数满足**递归关系**：当前状态的价值 = 即时奖励 + 折扣后继状态的期望价值。

这源于回报的递推性质：

$$G_t = R_{t+1} + \gamma G_{t+1}$$

状态价值函数的贝尔曼方程

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r + \gamma\, \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s']\right] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right], \quad \forall s \in \mathcal{S} \end{aligned} \tag{3.14}$$

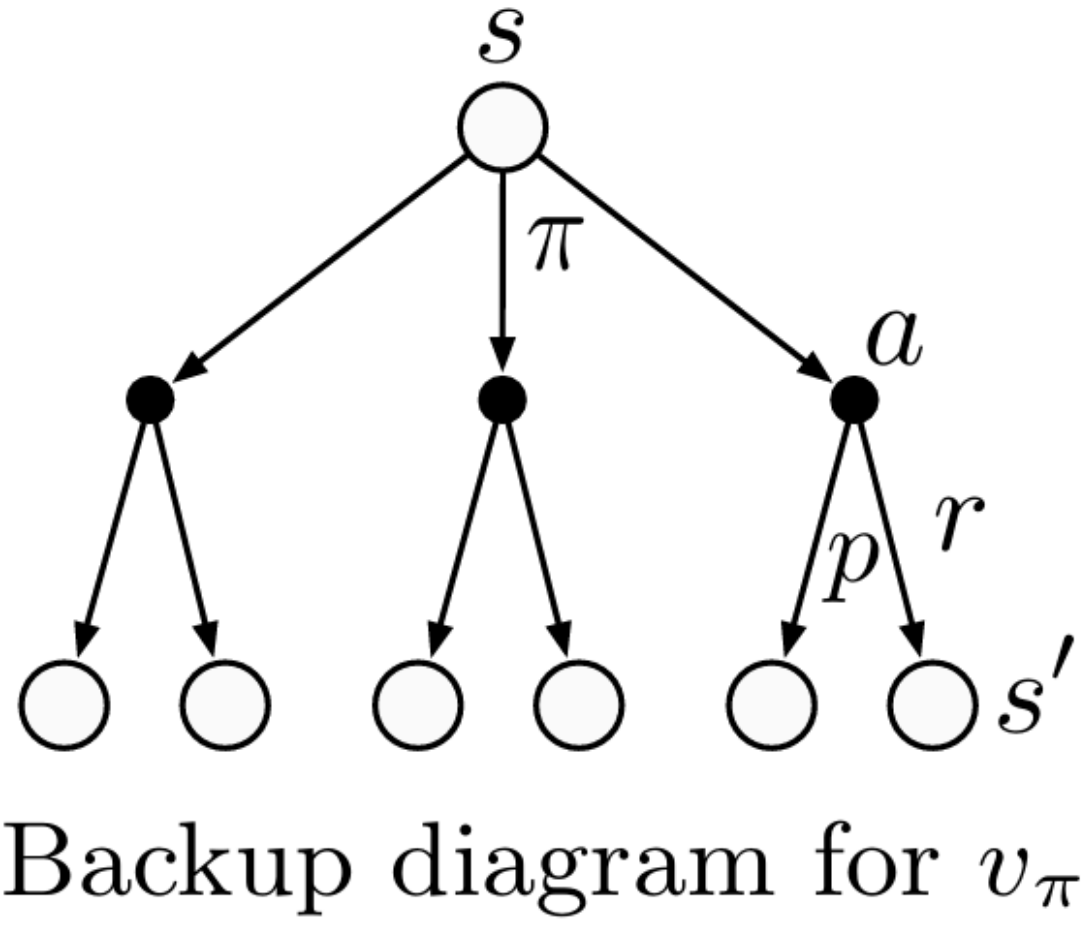
- 对每个可能动作 a，按概率 $\pi(a|s)$ 加权
- 对每个动作，考虑所有可能的转移 (s', r)，按 $p(s', r|s, a)$ 加权
- 每条路径贡献：$r + \gamma v_\pi(s')$
- 总价值是所有路径的加权平均

贝尔曼方程 (3.14) 对所有这些可能性进行加权平均，权重为其发生概率。它表明：**起始状态的价值**等于**期望下一状态的（折扣后）价值**加上**途中的期望奖励**。

备份图

备份图用于可视化贝尔曼方程中的**价值更新机制**。

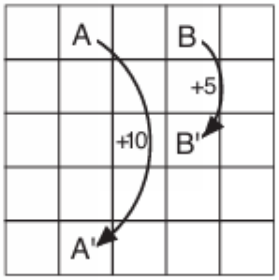
- **空心圆**：状态节点
- **实心圆**：状态-动作对节点
- 从状态 s 出发：
  - 根据策略 π 分支到各动作
  - 每个动作后根据环境动态 p 分支到 (s', r)
  - 最终连接到后继状态 s'



Backup diagram for v_π

案例2

网格世界



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

网格世界中的策略评估：计算等概率随机策略下的状态价值函数 v_π

1. 状态空间 $\mathcal{S}$

- 网格为 5×5，共 25 个状态，编号可按行优先从 (0,0) 到 (4,4)，或编号 1~25。
- 每个格子代表一个状态。
- 特殊状态：
  - **状态 A**：固定位置（图中左上角，如 (0,1)）
  - **状态 B**：固定位置（图中右下角，如 (0,3)）
  - **状态 A'**：A 被触发后转移到的位置（如 (4,1)）
  - **状态 B'**：B 被触发后转移到的位置（如 (2,3)）

2. 动作空间 $\mathcal{A}$

- 四个动作：**东 (E)、南 (S)、西 (W)、北 (N)。**
- 每个动作尝试使智能体向对应方向移动一格。

3. 状态转移与奖励函数 $p(s', r|s, a)$

一般规则（非 A/B 状态）：

- 执行动作后，若新位置在网格外 → **不移动，奖励 = -1。**
- 若新位置在网格内 → **移动成功，奖励 = 0。**

特殊规则（在状态 A 或 B）：

- 在 **状态 A**：无论执行哪个动作 →
  - **转移到 A'**
  - **奖励 = +10**
- 在 **状态 B**：无论执行哪个动作 →
  - **转移到 B'**
  - **奖励 = +5**

4. 策略 π

- **等概率随机策略**：在每个状态下，四个动作被选择的概率均为 0.25。
- 即：$\pi(a|s) = 0.25, \quad \forall a \in \{N, S, E, W\}, \forall s \in \mathcal{S}$

5. 折扣因子 γ

- $\gamma = 0.9$

状态价值函数 $v_\pi(s)$ 表示从状态 s 开始，遵循策略 π 的期望折扣回报：

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right]$$

根据贝尔曼期望方程：

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s') \right]$$

为 5×5 网格的每个状态分配唯一编号（0~24）：

```
Row 0:  0   1   2   3   4
Row 1:  5   6   7   8   9
Row 2: 10  11  12  13  14
Row 3: 15  16  17  18  19
Row 4: 20  21  22  23  24
```

→ 所以：

- A = (0,1) → 状态 1
- A' = (4,1) → 状态 21
- B = (0,3) → 状态 3
- B' = (2,3) → 状态 13

对每个状态 s，写出：

$$v_\pi(s) = \frac{1}{4} \sum_{a \in \{N,S,E,W\}} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s') \right]$$

示例：状态 A（s=1）

无论执行哪个动作：

- 转移到 s' = 21（A'）
- 奖励 r = +10

所以：

$$v_\pi(1) = \frac{1}{4} \cdot 4 \cdot [10 + 0.9 \cdot v_\pi(21)] = 10 + 0.9 \cdot v_\pi(21)$$

示例：状态 B（s=3）

同理：

$$v_\pi(3) = 5 + 0.9 \cdot v_\pi(13)$$

示例：角落状态（如 s=0，左上角）

动作分析：

- **北 (N)**：出界 → s'=0, r=-1
- **西 (W)**：出界 → s'=0, r=-1
- **东 (E)**：到 s=1, r=0
- **南 (S)**：到 s=5, r=0

所以：

$$v_\pi(0) = \frac{1}{4}[(-1 + 0.9v_\pi(0)) + (-1 + 0.9v_\pi(0)) + (0 + 0.9v_\pi(1)) + (0 + 0.9v_\pi(5))]$$

化简：

$$v_\pi(0) = \frac{1}{4}[-2 + 1.8v_\pi(0) + 0.9v_\pi(1) + 0.9v_\pi(5)]$$

→ 移项整理为线性方程：

$$v_\pi(0) - 0.45v_\pi(0) = -0.5 + 0.225v_\pi(1) + 0.225v_\pi(5)$$

$$\Rightarrow 0.55v_\pi(0) - 0.225v_\pi(1) - 0.225v_\pi(5) = -0.5$$

对每个状态 $s = 0, 1, \ldots, 24$，写出对应的贝尔曼方程，整理为标准线性形式：

$$\sum_{j=0}^{24} A_{ij} v_\pi(j) = b_i, \quad i = 0, \ldots, 24$$

其中：

- A 是 25×25 系数矩阵
- b 是 25 维常数向量
- v_π 是 25 维未知向量

使用数值方法求解：

```python
import numpy as np

# 初始化 A (25x25) 和 b (25,)
A = np.zeros((25, 25))
b = np.zeros(25)

gamma = 0.9

# 定义动作方向：N, S, E, W → (dr, dc)
actions = [(-1,0), (1,0), (0,1), (0,-1)]

# 状态坐标映射
def state_to_coord(s):
    return s // 5, s % 5

def coord_to_state(r, c):
    if 0 <= r < 5 and 0 <= c < 5:
        return r * 5 + c
    else:
        return None  # 表示出界

# 特殊状态定义
A_state = 1   # (0,1)
A_prime = 21  # (4,1)
B_state = 3   # (0,3)
B_prime = 13  # (2,3)

# 对每个状态 s
```

```
for s in range(25):
    r, c = state_to_coord(s)

    if s == A_state:
        # 特殊：所有动作 → A', +10
        A[s, s] = 1.0
        A[s, A_prime] = -gamma
        b[s] = 10.0
    elif s == B_state:
        # 特殊：所有动作 → B', +5
        A[s, s] = 1.0
        A[s, B_prime] = -gamma
        b[s] = 5.0
    else:
        # 一般状态
        for dr, dc in actions:
            nr, nc = r + dr, c + dc
            next_s = coord_to_state(nr, nc)
            if next_s is None:
                # 出界：留在原地，奖励 -1
                reward = -1
                next_s = s
            else:
                reward = 0

            # 累加贡献：π(a|s)=0.25
            A[s, s] += 0.25 * 1.0          # 自身系数（如果留在原地或转移到自己）
            if next_s != s:
                A[s, next_s] += 0.25 * gamma
            b[s] += 0.25 * reward

        # 移项：v(s) - sum(...) = b → A[s,s] 已包含 1.0，减去转移部分
        A[s, s] -= 1.0  # 因为原式是 v(s) = ...，移项后左边是 v(s) - ... = b

# 求解
v_pi = np.linalg.solve(A, b)

# 重塑为 5x5 网格
V_grid = v_pi.reshape(5,5)
print("状态价值函数 v_π (γ=0.9):")
print(np.round(V_grid, 2))
```

运行上述代码后，应得到类似如下价值分布（近似值）：

```
[[  3.3   8.8   4.4   5.3   1.5 ]
 [  1.5   3.0   2.3   1.9   0.5 ]
 [  0.1   0.7   0.7   0.4 -0.4 ]
 [ -1.0  -0.4 -0.4 -0.6 -1.2 ]
 [ -1.9  -1.3 -1.2 -1.4 -2.0 ]]
```

1. 边界状态价值为负

- 原因：随机策略下，边界状态执行某些动作（如向墙外）会获得 -1 奖励，且可能反复触发。
- 折扣累积后，期望回报为负。

2. 状态 A 价值 < 10

- 虽然立即获得 +10，但下一状态 A' 位于底部边界，后续出界概率高 → 未来折扣回报为负 → 拉低总价值。

3. 状态 B 价值 > 5

- 立即获得 +5，且转移到内部状态 B'，未来期望回报为正 → 总价值被提升。

4. 中心区域价值较高

- 远离边界，出界概率低 → 价值稳定或略正。

练习5

> 请证明：**当中心状态的四个相邻状态价值分别为 +2.3、+0.4、-0.4 和 +0.7 时，该中心状态的价值为 +0.7（精确到小数点后一位）**。

假设中心状态为 s，其四个邻居状态（按动作：北、南、东、西）的价值分别为：

- 北：$v_N = +2.3$
- 南：$v_S = +0.4$
- 东：$v_E = -0.4$
- 西：$v_W = +0.7$

由于是**内部状态**（不在边界），所有四个动作都会成功移动到相邻格子，**不会出界** → 所以每个动作的即时奖励 $r = 0$。

策略是**等概率随机策略** → $\pi(a|s) = 0.25$ 对每个动作。

根据贝尔曼期望方程：

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s') \right]$$

因为每个动作确定性地转移到对应邻居，且 $r = 0$，所以简化为：

$$v_\pi(s) = \frac{1}{4}[\gamma v_N + \gamma v_S + \gamma v_E + \gamma v_W] = \frac{\gamma}{4}(v_N + v_S + v_E + v_W)$$

代入数值：

- $\gamma = 0.9$
- $v_N = 2.3,\ v_S = 0.4,\ v_E = -0.4,\ v_W = 0.7$

计算总和：

$$v_N + v_S + v_E + v_W = 2.3 + 0.4 + (-0.4) + 0.7 = 3.0$$

代入公式：

$$v_\pi(s) = \frac{0.9}{4} \times 3.0 = \frac{2.7}{4} = 0.675$$

四舍五入到**小数点后一位**：

$$\boxed{0.7}$$

> 在网格世界例子中：

> - 到达目标（如状态 A、B）获得**正收益**；
> - 出界获得**负收益**；
> - 其他情况收益为 0。
>
> **问题：**
>
> 1. 是收益值的**符号**重要，还是**相对大小**更重要？
> 2. 用公式 (3.8) 证明：若对每个收益值都加上常数 c，则每个状态价值都会增加一个常数 v_c，因此**任何策略下，状态间的相对价值不变。**
> 3. 在给定 c 和 γ 的情况下，求 v_c 的值。

> **答：收益值的相对大小更重要，而非符号。**

- 在强化学习中，智能体的目标是**最大化累积折扣回报。**
- 如果所有奖励都加上一个常数 c，虽然每个时间步的“感觉”变了（比如从负变正），但**策略的优劣排序不会改变**，因为所有轨迹的回报都增加了相同的量（或按折扣累积的相同偏移）。
- **举例**：假设两条轨迹：
  - 轨迹1：回报 = 10
  - 轨迹2：回报 = 5
    → 轨迹1 更优。

  若所有奖励加 $c = 1$，折扣 $\gamma < 1$，则：
  - 新回报1 = 10 + v_c
  - 新回报2 = 5 + v_c
    → 差值仍为 5，**相对优劣不变。**
- **符号变化**（如原为负收益，加 c 后变正）可能影响人类对“惩罚/奖励”的直观理解，但**不影响最优策略或价值函数的相对排序。**

因此，相对大小更重要。

第 2 问：用式 (3.8) 证明价值函数平移不变性

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3.8}$$

假设我们对**所有时间步的奖励**加上一个常数 c，即定义新的奖励：

$$\tilde{R}_t = R_t + c$$

则新的回报为：

$$\tilde{G}_t = \sum_{k=0}^{\infty} \gamma^k \tilde{R}_{t+k+1} = \sum_{k=0}^{\infty} \gamma^k (R_{t+k+1} + c) = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} + \sum_{k=0}^{\infty} \gamma^k c$$

第一项就是原回报 G_t，第二项是几何级数：

$$\sum_{k=0}^{\infty} \gamma^k c = c \cdot \sum_{k=0}^{\infty} \gamma^k = c \cdot \frac{1}{1-\gamma} \quad (\text{当 } 0 \le \gamma < 1)$$

因此：

$$\tilde{G}_t = G_t + \frac{c}{1-\gamma}$$

于是，新的状态价值函数为：

$$\tilde{v}_\pi(s) = \mathbb{E}_\pi[\tilde{G}_t \mid S_t = s] = \mathbb{E}_\pi[G_t \mid S_t = s] + \frac{c}{1-\gamma} = v_\pi(s) + \frac{c}{1-\gamma}$$

令：

$$v_c = \frac{c}{1-\gamma}$$

则：

$$\boxed{\tilde{v}_\pi(s) = v_\pi(s) + v_c}$$

设 $\gamma = 0.9$，$c = 1$：

$$v_c = \frac{1}{1-0.9} = \frac{1}{0.1} = 10$$

→ 所有状态价值增加 10，但状态之间的**差值不变**，比如：

- 原：$v(A) = 8.8,\ v(B) = 5.3$ → 差 = 3.5
- 新：$v(A) = 18.8,\ v(B) = 15.3$ → 差 = 3.5

> 在**分幕式任务**（episodic task，如走迷宫）中，如果对**所有收益都加上一个常数** c，会对任务结果产生影响吗？还是会像持续性任务（continuing task）一样**没有影响**？请说明原因并举例。

在分幕式任务中，给所有收益加上常数 c 会显著影响任务结果（如最优策略、学习行为等），这与持续性任务不同。

项目	分幕式任务	持续性任务
是否终止	有明确终止状态（如迷宫出口、游戏结束）	无限持续，无终止
回报计算	从起始到终止的有限步回报总和（或折扣和）	无限折扣回报和 $\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
加常数 c 的影响	每一步 +c，总回报增加 $c \times T$（T 为幕长）→ **鼓励更长路径**	总偏移为常数 $v_c = \frac{c}{1-\gamma}$ → **不影响相对价值**

在分幕式任务中：

设一幕从时间 t 开始，到时间 T 结束（T 是随机变量，依赖策略和环境），则回报为：

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$$

若每个奖励加 c，新回报为：

$$\tilde{G}_t = \sum_{k=0}^{T-t-1} \gamma^k (R_{t+k+1} + c) = G_t + c \sum_{k=0}^{T-t-1} \gamma^k$$

→ **偏移量依赖于幕长 $T - t$!**

- 若 $\gamma = 1$（无折扣），则 $\tilde{G}_t = G_t + c \cdot (T - t)$ → **幕越长，奖励越多。**

- 即使 $\gamma < 1$，偏移量也**随路径长度变化** → 不同策略获得的"奖励加成"不同！

因此，策略的优劣排序可能改变！

场景：简单迷宫（分幕式任务）

- 起点 S，终点 G。
- 两条路径：
  - **路径 A（短）**：3 步到达终点，原奖励序列：`[0, 0, +10]` → 总回报 = 10（$\gamma = 1$）
  - **路径 B（长）**：6 步到达终点，原奖励序列：`[0, 0, 0, 0, 0, +10]` → 总回报 = 10

→ 两条路径回报相同，智能体可能任选其一（或偏好短路径以降低风险）。

现在，给每步奖励加 $c = +1$

- 路径 A 新回报：`[1, 1, 11]` → 总 = 1+1+11 = **13**
- 路径 B 新回报：`[1,1,1,1,1,11]` → 总 = 1×5 + 11 = **16**

→ **路径 B 现在回报更高！**

智能体将**偏好更长的路径** —— 尽管目标奖励没变，但"走路也能赚钱"。

极端例子：$c = -1$（每步惩罚）

- 路径 A：`[-1, -1, 9]` → 总 = 7
- 路径 B：`[-1×5, 9]` → 总 = 4

现在短路径更优

对比：如果是持续性任务（$\gamma < 1$）

如练习 3.15 所示，所有状态价值增加常数 $v_c = \frac{c}{1-\gamma}$，**差值不变** → 策略不变。

但在分幕式任务中：

> 偏移量 = $c \cdot \sum_{k=0}^{T-t-1} \gamma^k$，**依赖于幕长 → 策略可能改变**

> 写出动作价值函数 $q_\pi(s,a)$ 的**贝尔曼方程**，并用所有可能的"后继状态-动作"对 (s', a') 对应的动作价值 $q_\pi(s', a')$ 来表达。

状态价值函数的贝尔曼期望方程：

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s') \right] \tag{3.14}$$

这是对**状态价值**的递归定义：当前状态的价值 = 所有可能动作的加权平均，每个动作的期望回报 = 即时奖励 + 折扣后的下一状态价值。

动作价值函数定义：

$$q_\pi(s,a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s, A_t = a \right]$$

根据马尔可夫性质，可拆分为：

$$= \mathbb{E}_\pi \left[R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \cdots \right) \mid S_t = s, A_t = a \right] = \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a \right]$$

注意：G_{t+1} 是从 S_{t+1} 开始、遵循策略 π 的回报。但在 $S_{t+1} = s'$ 时，下一个动作 A_{t+1} 是根据 $\pi(\cdot|s')$ 随机选择的，因此：

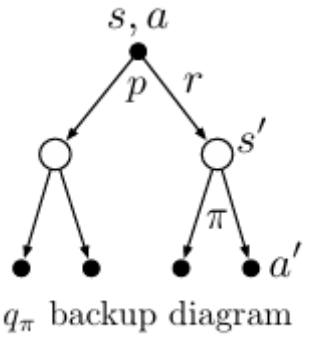
$$\mathbb{E}_\pi \left[G_{t+1} \mid S_{t+1} = s' \right] = v_\pi(s') = \sum_{a'} \pi(a'|s') q_\pi(s', a')$$

所以：

$$q_\pi(s,a) = \mathbb{E}_\pi \left[R_{t+1} + \gamma \sum_{a'} \pi(a'|S_{t+1}) q_\pi(S_{t+1}, a') \,\middle|\, S_t = s, A_t = a \right]$$

现在，将期望展开为对所有可能的 s' 和 r 求和（离散情况）：

$$q_\pi(s,a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a') \right]$$



q_π backup diagram

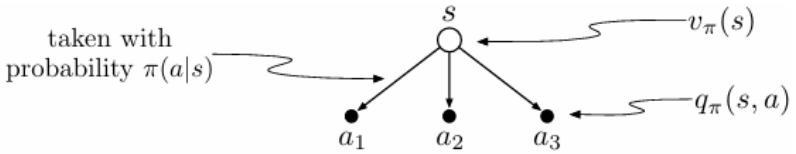
回溯图（backup diagram）中：

- 从 (s, a) 出发
- 分支到所有可能的 (s', r)
- 从每个 s'，再分支到所有可能的 a'（按 $\pi(a'|s')$ 加权）
- 终点为 $q_\pi(s', a')$

→ 正对应公式结构：

> “从 (s, a) → 环境转移 → (s', r) → 策略选择 a' → 未来价值 $q_\pi(s', a')$”

> 给定状态 $S_t = s$，根据回溯图（以状态 s 为根，分支到所有可能动作 a，叶子节点为 $q_\pi(s, a)$），写出：
>
> 1. 一个公式，用**期望符号**表达 $v_\pi(s)$，基于策略 π；
> 2. 一个**显式公式**，用 $\pi(a|s)$ 写出，**不出现期望符号。**



图中：

- **根节点**：状态 s → 对应 $v_\pi(s)$
- **第一层分支**：从 s 出发的所有可能动作 a
- **叶子节点**：每个动作 a 对应的动作价值 $q_\pi(s, a)$

根据定义，状态价值是在给定状态 s 下，遵循策略 π 的期望回报：

> 在状态 s，策略 π 会以某种概率选择动作 a，然后获得对应的动作价值 $q_\pi(s,a)$。

因此：

$$v_\pi(s) = \mathbb{E}_{A\sim\pi(\cdot|s)}\left[q_\pi(s,A)\right]$$

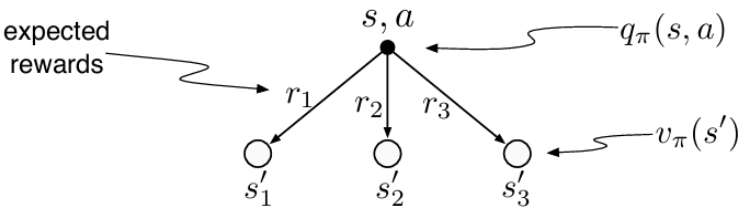
- 期望是在动作 A 上，根据策略 $\pi(\cdot|s)$ 分布取的。
- 即：从状态 s 出发，按策略 π 选动作，期望的动作价值就是状态价值。

将期望展开为对所有动作的加权求和：

$$\boxed{v_\pi(s) = \sum_a \pi(a|s)\, q_\pi(s,a)}$$

- $\pi(a|s)$：在状态 s 下选择动作 a 的概率；
- $q_\pi(s,a)$：在状态 s 下执行动作 a 后的期望回报；
- 加权平均 → 得到状态 s 下的总期望回报 $v_\pi(s)$。

> 给定状态-动作对 (s,a)，根据回溯图（根节点为 (s,a)，分支为下一时刻可能的状态 s' 和奖励 r），写出动作价值函数 $q_\pi(s,a)$ 的两个表达式：
>
> 1. 使用期望符号的公式（不显式依赖策略 π 在当前步，但隐含在后续价值中）；
> 2. 使用环境动态函数 $p(s',r\mid s,a)$ 显式展开期望，不出现期望符号。



1. 第 1 问：使用期望符号的公式

因此，公式为：

$$q_\pi(s,a) = \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \,\middle|\, S_t = s, A_t = a\right]$$

说明：

- 期望是关于环境动态（下一状态和奖励）的，不直接涉及策略 π 在当前动作的选择（因为 a 已给定）。
- 策略 π 仅隐含在 $v_\pi(S_{t+1})$ 中，因为 $v_\pi(s') = \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s']$。

第 2 问：显式写出，使用 $p(s',r\mid s,a)$，无期望符号

将期望按所有可能的下一状态 s' 和奖励 r 展开，使用权重 $p(s',r\mid s,a)$：

$$q_\pi(s,a) = \sum_{s',r} p(s',r\mid s,a)\left[r + \gamma v_\pi(s')\right]$$

说明：

- $p(s',r\mid s,a)$ 是环境四参数动态函数，表示在 (s,a) 下转移到 s' 并获得 r 的概率。
- 该公式不包含策略 π 在当前步的显式依赖（因为动作 a 已固定），但 $v_\pi(s')$ 仍依赖 π。

最优策略与最优价值函数

解决一个强化学习任务，本质上是找到一种能在长期运行中**最大化累积奖励**的策略。

对于**有限 MDP**，价值函数在策略之间定义了一种**偏序关系**，若策略 π 在所有状态下的期望回报都不小于另一策略 π'，即$v_\pi(s) \geq v_{\pi'}(s), \quad \forall s \in \mathcal{S}$，则称 π **优于或等于** π'，记作 $\pi \geq \pi'$。同时**总存在至少一个策略**，它优于或等于所有其他策略，这样的策略称为**最优策略**，记为 π_*。可能存在多个最优策略，但它们都具有相同的长期性能。

所有最优策略共享同一个状态价值函数，称为**最优状态价值函数**：

$$v_*(s) \doteq \max_\pi v_\pi(s), \quad \forall s \in \mathcal{S} \tag{3.15}$$

> $v_*(s)$ 表示从状态 s 出发，遵循某个最优策略所能获得的最大期望回报。

所有最优策略共享同一个动作价值函数，称为**最优动作价值函数**：

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s) \tag{3.16}$$

> $q_*(s,a)$ 表示在状态 s 执行动作 a 后，再遵循某个最优策略所能获得的最大期望回报。

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \tag{3.17}$$

> 执行动作 a 后的最优价值 = 即时奖励期望 + 折扣后的最优后继状态价值期望。

贝尔曼最优方程

v_* 的贝尔曼最优方程

由于 v_* 是某个最优策略的**价值函数**，它必须满足贝尔曼方程。但由于其"最优性"，该方程可以写成**不依赖具体策略**的形式。

$$\begin{aligned} v_*(s) &= \max_{a \in \mathcal{A}(s)} q_*(s,a) \\ &= \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right] \\ &= \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma v_*(s')\right] \end{aligned} \tag{3.19}$$

> **最优策略下,各个状态的价值一定等于这个状态下最优动作的期望回报。**

> ⚠ **Warning**
>
> **最优贝尔曼方程（Bellman Optimality Equation）的成立和可实现**是建立在**马尔可夫决策过程（MDP）**的严格数学框架之上。
>
> **马尔可夫性（Markov Property）**即"未来只依赖当前状态，与过去历史无关。"
>
> $$\mathbb{P}(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a, S_{t-1}, A_{t-1}, \ldots) = p(s', r \mid s, a)$$
>
> 如果环境不满足马尔可夫性（如部分可观测、有隐藏状态），贝尔曼方程不再精确成立。
>
> 这是一个非常重要、也非常深刻的问题!
>
> **最优贝尔曼方程（Bellman Optimality Equation）** 的标准形式：
>
> $$v_*(s) = \max_a q_*(s,a) = \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right]$$

- $v_*(s)$：状态 s 在**最优策略**下的价值。
- $q_*(s,a)$：在状态 s 下执行动作 a，之后遵循**最优策略**的期望回报。
- $\max_a q_*(s,a)$：在 s 下选择能带来最大期望回报的动作 —— 这就是“当前的最优动作”。

在最优策略中，**每个状态的价值确实等于该状态下最优动作的期望回报。**

因为 MDP 满足：

1. **马尔可夫性**：未来只依赖当前状态，与历史无关。
2. **贝尔曼最优性原理（Principle of Optimality）**：

> “一个策略在某一状态达到最优，当且仅当它在该状态选择的动作，能最大化即时回报 + 折扣后的后续最优状态价值。”

→ 也就是说：**局部最优 ⇒ 全局最优。**

举例：

- 走迷宫：每一步都选“能最快到达终点”的动作 → 最终一定最快。
- 高尔夫（例 3.6）：每一步选“期望击球次数最少”的球杆 → 总击球次数最少。
- 游戏：每一步选“胜率最高”的动作 → 整体胜率最高。

所以只要你的“当前最优”是基于**真正的最优价值函数 v_* 或 q_*”**，那么贪心地每步选最优动作，就能得到全局最优策略。**即每一步都选“当前最优”，一定能得到最优结果。**

关键前提：**“当前最优”必须是基于最优价值函数定义的。**在部分情况下,每一步都选择最优不一定能带来最优解。

反例 1：短视贪心（Myopic Greedy）

- 假设每步奖励：
  - 动作 A：立即得 +5，但之后被困，总回报 = 5
  - 动作 B：立即得 +1，但之后能得 +10，总回报 = 11

→ 如果你只看“即时奖励”，选 A（+5 > +1）→ **结果更差！**

→ 但如果你用 $q_*(s,a)$（包含未来折扣回报），就会选 B。

所以，“当前最优”必须是 $q_*(s,a)$，而不是“即时奖励最大”。

反例 2：非平稳环境或非马尔可夫问题

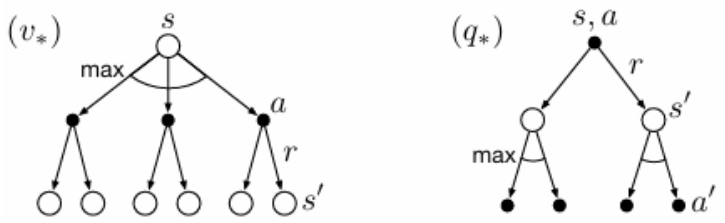
- 如果环境不是 MDP（比如有隐藏状态、长期依赖），贪心策略可能失效。
- 例如：某些博弈中，故意“示弱”引对手犯错 → 短期损失，长期收益。

→ 此时，“每步最优”可能不是全局最优。

q_* 的贝尔曼最优方程

$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \,\middle|\, S_t = s, A_t = a\right] \\
&= \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \max_{a'} q_*(s', a')\right]
\end{aligned}
\tag{3.20}
$$

区别于普通贝尔曼方程：这里对未来取**最大值**而非策略期望。



与普通备份图类似，但选择点增加**弧线**；弧线表示：此处取**最大值**，而非期望（$\mathbb{E}_\pi$）。

唯一解与最优策略的构造

对于有限 MDP，贝尔曼最优方程 (3.19) 有**唯一解**。它是一个包含 $n = |\mathcal{S}|$ 个方程和 n 个未知数（每个状态的 $v_*(s)$）的非线性方程组。若动态函数 p 已知，原则上可通过数值方法求解。

一旦求得 v_*，最优策略可通过**贪心策略**（greedy policy）构造：对每个状态 s，选择使贝尔曼最优方程中最大值成立的动作：

$$\pi_*(a|s) > 0 \quad \text{仅当} \quad a \in \arg\max_{a'} \sum_{s',r} p(s', r|s, a')[r + \gamma v_*(s')]$$

任何仅以非零概率选择这些动作的策略都是最优策略。

“贪心”通常指仅考虑短期收益的策略；但 v_* 已经包含了**所有未来最优行为的长期奖励**；因此，**基于 v_* 的一步前瞻贪心选择，实际上就是长期最优选择。**

v_* 将长期最优决策“压缩”为每个状态的局部值。

拥有 q_* 则可使决策更简单：智能体无需知道环境动态 p；无需计算后继状态价值；只需对每个状态 s，选择：

$$a^* = \arg\max_a q_*(s, a)$$

因为q_* **缓存了所有一步搜索的结果**，是最直接的最优动作指南

上文案例

网格世界

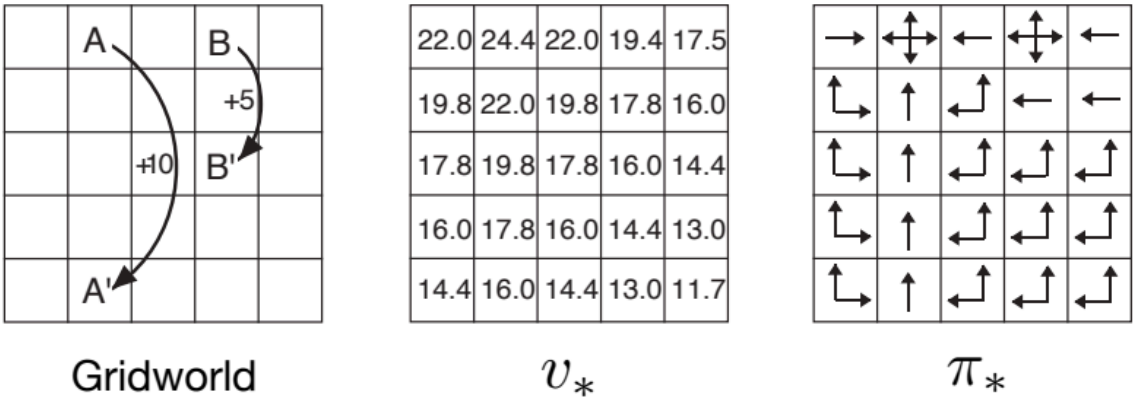


图 3.5： 网格世界的最优解

- **左图**：任务结构（同图 3.2）
- **中图**：最优状态价值函数 v_*
- **右图**：最优策略（箭头表示最优动作，多个箭头表示多个动作同优）

易见：

- A 和 B 附近价值高
- 边缘区域价值低（避免撞墙）
- 从任意位置出发，策略引导朝向 A 或 B

回收机器人

状态：`high` (h), `low` (l)
动作：`search` (s), `wait` (w), `recharge` (re)
简写期望奖励：$r_s = r_{\text{search}}$，$r_w = r_{\text{wait}}$

$v_*(h)$ 的贝尔曼最优方程

$$
\begin{aligned}
v_*(h) &= \max \begin{Bmatrix} \text{选择 search:} \quad \alpha[r_s + \gamma v_*(h)] + (1-\alpha)[r_s + \gamma v_*(l)] \\ \text{选择 wait:} \quad 1 \cdot [r_w + \gamma v_*(h)] \end{Bmatrix} \\
&= \max \{ r_s + \gamma[\alpha v_*(h) + (1-\alpha) v_*(l)], \quad r_w + \gamma v_*(h) \}
\end{aligned}
$$

$v_*(l)$ 的贝尔曼最优方程

$$
v_*(l) = \max \begin{Bmatrix} \text{search:} \quad \beta r_s - 3(1-\beta) + \gamma[(1-\beta) v_*(h) + \beta v_*(l)] \\ \text{wait:} \quad r_w + \gamma v_*(l) \\ \text{recharge:} \quad \gamma v_*(h) \end{Bmatrix}
$$

对任意 $0 \le \gamma < 1$，$0 \le \alpha, \beta \le 1$，存在**唯一解** $(v_*(h), v_*(l))$

显式求解与近似方法

虽然理论上可通过求解贝尔曼最优方程得到最优策略，但**极少直接实用**，因为依赖以下三个理想假设：

假设	是否常成立	说明
1. 环境动态 p 已知	很少	需要完整模型
2. 足够计算资源	常不成立	状态空间巨大时不可行
3. 马尔可夫性质成立	通常假设成立	本书基础

3.7 最优性与近似

理论上，实现最优策略意味着智能体表现完美。但在**实际任务中，达到最优极为罕见。 最优性是一个理想化的理论目标，实践中我们追求的是“足够好”的近似解。**

即使满足以下理想条件，求解最优仍面临巨大挑战：

挑战	说明
计算资源限制	单个时间步内可执行的计算量有限
内存限制	无法存储所有状态的价值或策略
状态空间爆炸	实际任务的状态数远超计算机存储能力

例如国际象棋状态数巨大，远超当前算力可穷举范围。即使拥有完整环境模型（即知道所有规则和转移概率），也无法直接求解贝尔曼最优方程智能体必须在**有限时间内做出决策**，不能等待“全局最优解”。

强化学习的一个关键特性是其**在线学习能力**，智能体在与环境交互中逐步学习，可以将学习资源**集中在高频出现的状态上**，对极少访问的状态，允许次优决策，不影响整体性能。 **“在重要状态上做得好”比“在所有状态上都最优”更实用**