第三章 强化学习基础

3.1 基本概念

强化学习中通常用大写字母表示随机变量，用小写字母表示观测值。比如 X 是随机变量，x 是 X 的一个观测值。用 $\mathbb{P}(X = x)$ 表示“事件 $X = x$ 发生”的概率，用 $\mathbb{P}(Y = y|X = x)$ 表示在“$Y = y$ 发生”这个条件下“事件 $X = x$ 发生”的概率.

强化学习较难入门的一个原因是它有许多专业术语。想入门强化学习，没有什么捷径，只能理解和记住专业术语。为了方便读者理解和记忆，本节主要用超级玛丽的例子来解释强化学习的专业术语。

图 3.1: 超级玛丽的例子中，玛丽奥是智能体，状态 s 是当前屏幕上的画面，动作空间是 $\mathcal{A} = \{左, 右, 上\}$，动作 a 是左、右、上三者中的一个。

状态 (State) 是对当前环境的一个概括。在超级玛丽的例子中，可以把屏幕当前的画面（或者最近几帧画面）看做状态。玩家只需要知道当前画面（或者最近几帧画面）就能够做出正确的决策，决定下一步是让超级玛丽向左、向右、或是向上。可以这样理解状态：状态是做决策的唯一依据。

再举一个例子，在中国象棋、五子棋游戏中，棋盘上所有棋子的位置就是状态，因为当前格局就足以供玩家做决策。假设你不是从头开始一局游戏，而是接手别人的残局。你只需要仔细观察棋盘上的格局，你就能够做出决策。知道这局游戏的历史记录（即每一步是怎么走的），并不会给你提供额外的信息。

举一个反例。星际争霸、红色警戒、英雄联盟这些游戏中，玩家屏幕上最近的 100 帧画面并不是状态，因为这些画面不是对当前环境完整的概括。在地图上某个你看不见的角落里可能正在发生些事件，这些事件足以改变游戏的结局。一个玩家屏幕上的画面只是对环境的部分观测 (Partial Observation)。最近的 100 帧画面并不足以供玩家做决策。

状态空间 (State Space) 是指所有可能存在状态的集合，记作花体字母 $\mathcal{S}$。状态空间可能是有限集合，也可能是无限集合。在超级玛丽、星际争霸、无人驾驶这些例子中，状态空间是无限集合，存在无穷多种可能的状态。围棋、五子棋、中国象棋这些游戏中，状态空间是有限集合，可以枚举出所有可能存在的状态（也就是棋盘上的格局）。

动作 (Action) 是指做出的决策。在超级玛丽的例子中，假设玛丽奥只能向左走、向右走、向上跳。那么动作就是左、右、上三者中的一种。在围棋游戏中，棋盘上有 361 个位置，于是有 361 种动作，第 i 种动作是指把棋子放到第 i 个位置上。

动作空间 (Action Space) 是指所有可能的动作的集合，记作花体字母 $\mathcal{A}$。在超级玛丽的例子中，动作空间是 $\mathcal{A}=\{$左$,$右$,$上$\}$。在围棋的例子中，动作空间是 $\mathcal{A}=\{1,2,3,\cdots,361\}$。

智能体 (Agent) 是指做动作的主体：由谁做动作，谁就是智能体。在超级玛丽游戏中，玛丽奥就是智能体。在自动驾驶的应用中，无人车就是智能体。

策略函数 (Policy Function) 的意思是根据观测到的状态，做出决策，控制智能体运动。举个例子，假设你在玩超级玛丽游戏，当前屏幕上的画面是图 3.1。请问你该做什么决策？有很大概率你会决定向上跳，这样可以避开敌人，还能吃到金币。向上跳这个动作就是你大脑中的策略做出的决策。

有不同的方式定义策略函数。这里介绍一种最常用的定义。把状态记作 S 或 s，动作记作 A 或 a。策略函数 $\pi:\mathcal{S}\times\mathcal{A}\mapsto[0,1]$ 是一个条件概率密度函数：

$$\pi(a|s) \ = \ \mathbb{P}\big(A=a \,\big|\, S=s\big).$$

策略函数的输入是状态 s 和动作 a，输出是一个 0 到 1 之间的概率值。举个例子，把图 3.1 中的屏幕画面作为状态 s 输入策略函数，策略函数输出动作的概率值：

$$\begin{aligned}\pi\big(\text{左}\,\big|\,s\big) &= 0.2,\\ \pi\big(\text{右}\,\big|\,s\big) &= 0.1,\\ \pi\big(\text{上}\,\big|\,s\big) &= 0.7.\end{aligned}$$

如果你让策略函数 π 来自动操作玛丽奥打游戏，它就会做一个随机抽样：以 0.2 的概率向左走，0.1 的概率向右走，0.7 的概率向上跳。三种动作都有可能发生，但是向上的概率最大，向左概率较小，向右概率最小。

强化学习学什么？就是学这个策略函数 π。只要有了策略函数，就可以让它自动控制玛丽奥打赢游戏。

奖励 (Reward) 是在智能体执行一个动作之后，环境返回给智能体的一个数值。奖励往往由我们自己来定义；奖励定义得好坏非常影响强化学习的结果。比如可以这样定义，玛丽奥吃到一个金币，获得奖励 $+1$；如果玛丽奥通过一局关卡，奖励是 $+1000$；如果玛丽奥碰到敌人，游戏结束，奖励是 -1000；如果这一步什么都没发生，奖励就是 0。怎么定义奖励就见仁见智了。我们应该把打赢游戏的奖励定义得大一些，这样才能鼓励玛丽奥通过关卡，而不是一味地收集金币。

状态转移 (State Transition) 是指当前状态 s 变成新的状态 s'。给定当前状态 s，智能体执行动作 a，环境 (Environment) 给出下一时刻的状态 s'。请问 s' 是如何产生的呢？s' 是由环境根据某个函数计算出来的，这个函数叫做状态转移函数，它把 (s,a) 映射到 s'；稍后详细解释状态转移函数。

环境 (Environment) 又是什么呢？在超级玛丽的例子中，游戏程序就是环境。在围棋、象棋的例子中，游戏规则就是环境。在无人驾驶的应用中，真实的物理世界就是环

境。谁能生成新的状态，谁就是环境。

状态转移函数 (State-Transition Function) 是环境用于生成新的状态 s' 时用到的函数。在超级玛丽的例子中，基于当前状态（屏幕上的画面），玛丽奥向上跳了一步，那么环境（即游戏程序）就会计算出新的状态（即下一帧画面）。在中国象棋的例子中，基于当前状态（棋盘上的格局），红方让“车”走到黑方“马”的位置上，那么环境（即游戏规则）就会将黑方的“马”移除，生成新的状态（棋盘上新的格局）。

状态转移函数可以是确定的。比如中国象棋的状态转移函数就是确定的：给定当前状态 s，玩家执行动作 a，那么新的状态 s' 是确定的，没有随机性。状态转移函数也可能是随机的；我们通常认为状态转移是随机的。状态转移的随机性是从环境来的。图 3.2 中的例子说明状态转移的随机性。

图 3.2: 这个例子说明状态转移的随机性。如果玛丽奥向上跳，玛丽奥的位置就到上面来了；这个是确定的。但是标出的敌人 Goomba 有可能往左，也有可能往右。Goomba 移动的方向可以是随机的。即使当前状态 s 和智能体的动作 a 确定了，也无法确定下一个状态 s'。

随机状态转移函数记作 $p(s'|s,a)$，它是一个条件概率密度函数：

$$p(s'|s,a) \;=\; \mathbb{P}\big(S'=s' \,\big|\, S=s, A=a\big).$$

意思是如果观测到当前状态 s 以及动作 a，那么 p 函数输出状态变成 s' 的概率。本书中只考虑随机状态转移，因为确定状态转移是是随机状态转移的一个特例：概率质量全部集中在一个状态 s' 上。

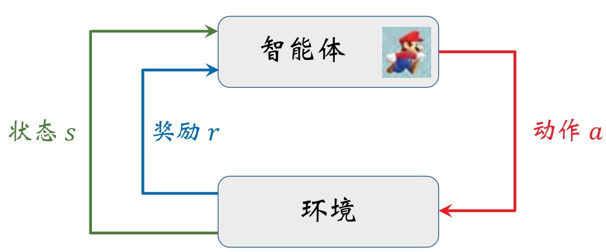

图 3.3: 智能体与环境交互。

智能体与环境交互 (Agent Environment Interaction) 是指智能体观测到环境的状态

s，做出动作 a，动作会改变环境的状态，环境反馈给智能体奖励 r 以及新的状态 s'。图 3.3 是智能体与环境交互的示意图。在超级玛丽的游戏中，智能体是玛丽奥，环境是游戏程序。AI 以下面的方式控制玛丽奥跟游戏程序交互。观测到当前状态 s，AI 用策略函数 $\pi(a|s)$ 算出所有动作的概率，比如算出

$$\pi\big(\text{左}\,\big|\,s\big) \;=\; 0.2, \qquad \pi\big(\text{右}\,\big|\,s\big) \;=\; 0.1, \qquad \pi\big(\text{上}\,\big|\,s\big) \;=\; 0.7.$$

按照概率做随机抽样，得到其中一个动作（比如向上），记作 a，然后玛丽奥执行这个动作。游戏程序会用状态转移函数 $p(s'|s,a)$ 随机生成新的状态 s'，并反馈给玛丽奥一个奖励 r。

3.2 随机性的来源

这一节的内容是强化学习中的随机性。随机性有两个来源：策略函数与状态转移函数。搞明白随机性的两个来源，对之后的学习很有帮助。

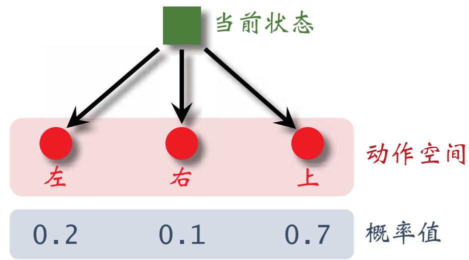

图 3.4: 状态空间是 $\mathcal{A}=\{左,中,右\}$。把当前状态 s 输入策略函数，策略函数输出三个概率值：0.2, 0.1, 0.7。所以，对于确定的状态 s，智能体执行的动作是不确定的，三个动作都可能被执行。

动作的随机性来自于**策略函数**。给定当前状态 s，策略函数 $\pi(a|s)$ 会算出动作空间 $\mathcal{A}$ 中每个动作 a 的概率值。智能体执行的动作是随机抽样的结果，所以带有随机性。见图 3.4 中的例子。

状态的随机性来自于**状态转移函数**。当状态 s 和动作 a 都被确定下来，下一个状态仍然有随机性。环境（比如游戏程序）用状态转移函数 $p(s'|s,a)$ 计算所有可能的状态的概率，然后做随机抽样，得到新的状态。见图 3.5 中的例子。

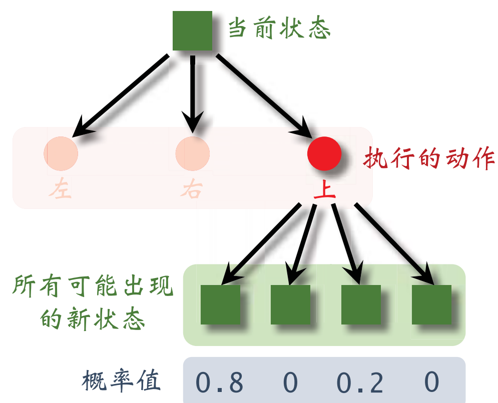

图 3.5: 已知当前状态 s，智能体已经做出决策——向上跳，那么环境会更新状态。环境把 s 和 a 输入状态转移函数，得到所有可能的状态的概率值。环境根据概率值做随机抽样，得到新的状态 s'。

奖励可以看做状态和动作的函数。给定当前状态 s_t 和动作 a_t，那么奖励 r_t 就是唯一确定的。假设给定当前状态 s_t，但智能体尚未做决策，也就是说 t 时刻动作还未知，应当记作随机变量 A_t（而非 a_t）；那么 t 时刻的奖励仍然未知，应当记作随机变量 R_t（而非 r_t），它的随机性从未知的动作 $A_t\sim\pi(\cdot|s_t)$ 中来。

注 在很多应用中，奖励 r_t 取决于 s_t、a_t、s_{t+1}。在这种情况下，即使给定当前状态 s_t 和动作 a_t，奖励 R_t 仍然是未知的变量，它的随机性从未知的新状态 $s_{t+1}\sim p(\cdot|s_t,a_t)$ 中来。

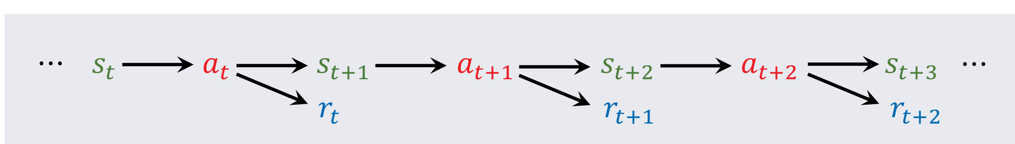

图 3.6: 智能体的轨迹。

轨迹 (Trajectory) 是指一回合 (Episode) 游戏中，智能体观测到的所有的状态、动作、奖励：

$$s_1,a_1,r_1,\quad s_2,a_2,r_2,\quad s_3,a_3,r_3,\quad\cdots$$

图 3.6 描绘了轨迹中状态、动作、奖励的依赖关系。在 t 时刻，给定状态 $S_t=s_t$，下面

这些都是观测到的值：

$$s_1, a_1, r_1, \quad s_2, a_2, r_2, \quad \cdots, \quad s_{t-1}, a_{t-1}, r_{t-1}, \quad s_t,$$

而下面这些都是随机变量（尚未被观测到）：

$$A_t, R_t, \quad S_{t+1}, A_{t+1}, R_{t+1}, \quad S_{t+2}, A_{t+2}, R_{t+2}, \quad \cdots$$

3.3 回报与折扣回报

本节介绍回报 (Return) 和折扣回报 (Discounted Return) 这两个概念，并且讨论其随机性来源。由于回报是折扣率等于 1 的特殊折扣回报，后面的章节中用“回报”指代“折扣回报”，不再区分两者。

3.3.1 回报

回报 (Return) 是从当前时刻开始到一回合结束的所有奖励的总和，所以回报也叫做**累计奖励** (Cumulative Future Reward)。把 t 时刻的回报记作随机变量 U_t；如果一局游戏结束，已经观测到所有奖励，那么就把回报记作 u_t。回报的定义是这样的：

$$U_t \;=\; R_t + R_{t+1} + R_{t+2} + R_{t+3} + \cdots$$

回报有什么用呢？回报是未来获得的奖励总和，所以智能体的目标就是让回报尽量大，越大越好。强化学习的目标就是寻找一个策略，使得回报的期望最大化。

注 强化学习的目标是最大化回报，而不是最大化当前的奖励。打个比方，下棋的时候，你的目标是赢得一局比赛（回报），而非吃掉对方一个棋子（奖励）。

3.3.2 折扣回报

思考一个问题：在 t 时刻，请问奖励 r_t 和 r_{t+1} 同等重要吗？假如我给你两个选项：第一，现在我立刻给你 100 元钱; 第二，等一年后我给你 100 元钱。你选哪个？理性人应该都会选现在得到 100 元钱。这是因为未来的不确定性很大，即使我现在答应明年给你 100 元，你也未必能拿到。大家都明白这个道理：明年得到 100 元不如现在立刻拿到 100 元。

要是换一个问题，现在我立刻给你 80 元钱，或者是明年我给你 100 元钱。你选哪一个？或许大家会做不同的选择，有的人愿意拿现在的 80，有的人愿意等一年拿 100。如果两种选择一样好，那么就意味着一年后的奖励的重要性只有今天的 $\gamma = 0.8$ 倍。这里的 $\gamma = 0.8$ 就是**折扣率** (Discount Factor)。

同理，在强化学习中，通常使用**折扣回报 (Discounted Return)**，给未来的奖励做折扣。这是折扣回报的定义：

$$U_t \;=\; R_t + \gamma \cdot R_{t+1} + \gamma^2 \cdot R_{t+2} + \gamma^3 \cdot R_{t+3} + \cdots$$

这里的 $\gamma \in [0,1]$ 叫做折扣率。对待越久远的未来，给奖励打的折扣越大。折扣率是个超参数，需要手动调；折扣率的设置会影响强化学习的结果。

3.3.3 回报中的随机性

假设一回合游戏一共有 n 步。当完成这一回合之后，我们观测到所有 n 个奖励：$r_1, r_2, \cdots, r_n$。此时这些奖励不是随机变量，而是实际观测到的数值。此时我们可以实

际计算出折扣回报

$$u_t \;=\; r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \cdots + \gamma^{n-t} \cdot r_n, \qquad \forall\, t = 1, \cdots, n.$$

此时的折扣回报 u_t 是实际观测到的数值，不具有随机性。

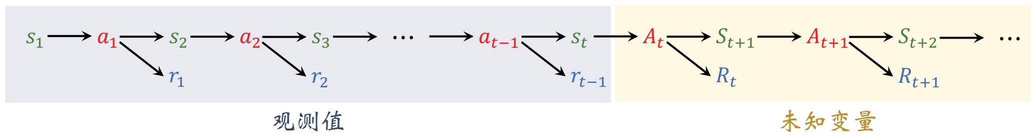

图 3.7: 智能体的轨迹中 s_t 及其之前的状态、动作、奖励都被观测到，而 A_t 及其之后的状态、动作、奖励都是未知变量。

假设我们此时在第 t 时刻，我们只观测到 s_t 及其之前的状态、动作、奖励

$$s_1, a_1, r_1, \;\; s_2, a_2, r_2, \;\; \cdots, \;\; s_{t-1}, a_{t-1}, r_{t-1}, \;\; s_t,$$

而下面这些都是随机变量（尚未被观测到）：

$$A_t, R_t, \;\; S_{t+1}, A_{t+1}, R_{t+1}, \;\; \cdots \;\; S_n, A_n, R_n.$$

见图 3.7。回报 U_t 依赖于奖励 $R_t, R_{t+1}, \cdots, R_n$，而这些奖励全都是未知的随机变量，所以 U_t 也是未知的随机变量。

请问回报 U_t 的随机性的来源是什么？奖励 R_t 依赖于状态 s_t（已观测到）与动作 A_t（未知变量），奖励 R_{t+1} 依赖于 S_{t+1} 和 A_{t+1}（未知变量），奖励 R_{t+2} 依赖于 S_{t+2} 和 A_{t+2}（未知变量），以此类推。所以 U_t 的随机性来自于这些动作和状态：

$$A_t, \;\; S_{t+1}, A_{t+1}, \;\; S_{t+2}, A_{t+2}, \;\; \cdots, \;\; S_n, A_n.$$

动作的随机性来自于策略函数，状态的随机性来自于状态转移函数。

3.4 价值函数

本节介绍动作价值函数 $Q_\pi(s,a)$，最优动作价值函数 $Q_\star(s,a)$，状态价值函数 $V_\pi(s)$。它们都是回报的期望。

3.4.1 动作价值函数

上一节介绍了（折扣）回报 U_t，它是 t 时刻之后所有奖励的（加权）和。在 t 时刻，假如我们知道 U_t 的值，我们就知道游戏是快赢了还是快输了。然而在 t 时刻我们并不知道 U_t 的值，因为此时 U_t 仍然是个随机变量。在结束本回合游戏之前，我们都会不知道 U_t 的值。在 t 时刻，我们不知道 U_t 的值，而我们又想预判 U_t 的值从而知道局势的好坏。该怎么办呢？解决方案就是对 U_t 求期望，消除掉其中的随机性。

为什么求期望可以消除掉随机性呢？打个比方，抛硬币，正面记做 $X=1$，反面记做 $X=0$。在抛硬币之前，并不知道随机变量 X 是 1 还是 0。如果对 X 求期望，可以消除掉随机性，得到一个具体的数值 $\mathbb{E}[X]=0.5$。同理，对 U_t 求期望，就能得到一个具体的数值。

假设我们已经观测到状态 s_t，而且做完决策，选中动作 a_t。那么 U_t 中的随机性来自于 $t+1$ 时刻之后的状态和动作：

$$S_{t+1}, A_{t+1}, \quad S_{t+2}, A_{t+2}, \quad \cdots, \quad S_n, A_n.$$

对 U_t 关于变量 $S_{t+1}, A_{t+1}, \cdots, S_n, A_n$ 求条件期望，得到

$$Q_\pi(s_t,a_t) \;=\; \mathbb{E}_{S_{t+1},A_{t+1},\cdots,S_n,A_n}\Big[U_t \,\Big|\, S_t=s_t, A_t=a_t\Big].$$

期望中的 $S_t=s_t$ 和 $A_t=a_t$ 是条件，意思是已经观测到 S_t 与 A_t 的值。条件期望的结果 $Q_\pi(s_t,a_t)$ 被称作**动作价值函数** (Action-Value Function)。

动作价值函数 $Q_\pi(s_t,a_t)$ 依赖于 s_t 与 a_t，而不依赖于 $t+1$ 时刻及其之后的状态和动作，这是因为随机变量 $S_{t+1}, A_{t+1}, \cdots, S_n, A_n$ 都被期望消除了。从下面的公式中可以看出，$Q_\pi(s_t,a_t)$ 依赖于策略函数 $\pi(a|s)$：

$$\begin{aligned} Q_\pi(s_t,a_t) \;&=\; \mathbb{E}_{S_{t+1},A_{t+1},\cdots,S_n,A_n}\Big[U_t \,\Big|\, S_t=s_t, A_t=a_t\Big] \\ &= \int_{\mathcal{S}} d\, s_{t+1} \int_{\mathcal{A}} d\, a_{t+1} \cdots \int_{\mathcal{S}} d\, s_n \int_{\mathcal{A}} d\, a_n \underbrace{\left[\prod_{k=t+1}^{n} p\big(s_k \big| s_{k-1}, a_{k-1}\big) \cdot \pi\big(a_k \big| s_k\big)\right]}_{\text{概率密度函数}} \cdot U_t. \end{aligned}$$

公式中的 π 是动作的概率密度函数；用不同的 π，连加结果就会不同。这就是为什么动作价值函数 Q_π 有下标 π。综上所述，t 时刻的动作价值函数 $Q_\pi(s_t,a_t)$ 依赖于以下三个因素：

- 第一，当前状态 s_t。当前状态越好，那么价值 $Q_\pi(s_t,a_t)$ 越大，也就是说回报的期望值越大。在超级玛丽的游戏中，如果玛丽奥当前已经接近终点，马上就能赢一局游戏，那么 $Q_\pi(s_t,a_t)$ 就非常大。
- 第二，当前动作 a_t。智能体执行的动作越好，那么价值 $Q_\pi(s_t,a_t)$ 越大。举个例子，如果玛丽奥做正常的动作，那么 $Q_\pi(s_t,a_t)$ 就比较正常；如果玛丽奥的动作 a_t 是跳

下悬崖，那么 $Q_\pi(s_t, a_t)$ 就会非常小。

- 第三，策略函数 π。策略决定未来的动作 $A_{t+1}, A_{t+2}, \cdots, A_n$ 的好坏。策略越好，那么 $Q_\pi(s_t, a_t)$ 就越大。举个例子，顶级玩家相当于好的策略 π；新手就相当于差的策略。让顶级玩家操作游戏，回报的期望非常高；换新手操作游戏，从相同的状态出发，回报的期望会很低。

3.4.2 最优动作价值函数

怎么样才能排除掉策略 π 的影响，只评价当前状态和动作的好坏呢？解决方案就是**最优动作价值函数** (Optimal Action-Value Function)：

$$Q_\star(s_t, a_t) \;=\; \max_\pi \; Q_\pi(s_t, a_t), \qquad \forall\, s_t \in \mathcal{S}, \quad a_t \in \mathcal{A}.$$

公式的意思是有很多种策略函数 π 可供选择，而我们选择最好的策略函数：

$$\pi^\star \;=\; \underset{\pi}{\operatorname{argmax}} \; Q_\pi(s_t, a_t), \qquad \forall\, s_t \in \mathcal{S}, \quad a_t \in \mathcal{A}.$$

$Q_\star$ 和 $Q_{\pi^\star}$ 指的都是最优动作价值函数。$Q_\star(s_t, a_t)$ 只依赖于 s_t 和 a_t，而与策略 π 无关。

最优动作价值函数 $Q_\star$ 非常有用：它就像是一个先知，能指引智能体做出正确决策。比如玩超级玛丽，给定当前状态 s_t，智能体该执行动作空间 $\mathcal{A} = \{左, 右, 上\}$ 中的哪个动作呢？假设我们已知 $Q_\star$ 函数，那么我们就让 $Q_\star$ 给三个动作打分，比如：

$$Q_\star\big(s_t,\, 左\big) \;=\; 130, \qquad Q_\star\big(s_t,\, 右\big) \;=\; -50, \qquad Q_\star\big(s_t,\, 上\big) \;=\; 296.$$

这三个值是什么意思呢？$Q_\star(s_t,\, 左) \,=\, 130$ 的意思是：如果现在智能体选择向左走，那么不管以后智能体用什么策略函数 π，回报 U_t 的期望最多不会超过 130。同理，如果现在向右走，则回报的期望最多不超过 -50；如果现在向上跳，则回报的期望最多不超过 296。智能体应该执行哪个动作呢？毫无疑问，智能体当然应该向上跳，这样才能有希望获得尽量高的回报。

3.4.3 状态价值函数

假设 AI 用策略函数 π 下围棋。AI 想知道当前状态 s_t（即棋盘上的格局）是否对自己有利，以及自己和对手的胜算各有多大。该用什么来量化双方的胜算呢？答案是**状态价值函数** (State-Value Function)：

$$\begin{aligned} V_\pi(s_t) \;&=\; \mathbb{E}_{A_t \sim \pi(\cdot|s_t)}\Big[Q_\pi(s_t, A_t)\Big] \\ &=\; \sum_{a \in \mathcal{A}} \pi(a|s_t) \cdot Q_\pi(s_t, a). \end{aligned}$$

公式把动作 A_t 作为随机变量，关于 A_t 求期望，把 A_t 消掉。得到的状态价值函数 $V_\pi(s_t)$ 只依赖于策略 π 与当前状态 s_t，不依赖于动作。状态价值函数 $V_\pi(s_t)$ 也是回报 U_t 的期望：

$$V_\pi(s_t) \;=\; \mathbb{E}_{A_t, S_{t+1}, A_{t+1}, \cdots, S_n, A_n}\Big[U_t \,\Big|\, S_t = s_t\Big].$$

期望消掉了 U_t 依赖的随机变量 $A_t, S_{t+1}, A_{t+1}, \cdots, S_n, A_n$。状态价值 $V_\pi(s_t)$ 越大，就意味着回报 U_t 的期望越大。用状态价值可以衡量策略 π 与状态 s_t 的好坏。

3.5 策略学习和价值学习

假如我们想设计一种 AI，让它自动打超级玛丽游戏。AI 打游戏的目标是避开敌人、通过关卡、并收集尽量多的金币。我们需要自己来定义奖励，比如每个金币的奖励是 $+1$，通过一个关卡的奖励是 $+1000$，碰到敌人或落下悬崖的奖励是 -1000。AI 的目标是最大化（折扣）回报，也就是最大化奖励的（加权）总和。定义好了目标，就可以设计强化学习方法来实现目标。强化学习方法通常分为两类：基于模型的方法 (Model-Based) 和无模型方法 (Model-Free)，本书主要介绍后者。无模型方法又可以分为价值学习和策略学习。

价值学习 (Value-Based Learning) 通常是指学习最优价值函数 $Q_\star(s,a)$（或者动作价值函数、状态价值函数）。假如我们有了 $Q_\star$，智能体就可以根据 $Q_\star$ 来做决策，选出最好的动作。每次观测到一个状态 s_t，把它输入 $Q_\star$ 函数，让 $Q_\star$ 对所有动作做评价，比如

$$Q_\star(s_t, \text{左}) \;=\; 273, \qquad Q_\star(s_t, \text{右}) \;=\; -139, \qquad Q_\star(s_t, \text{上}) \;=\; 195.$$

这些 Q 值量化每个动作的好坏。智能体应该执行 Q 值最大的动作，也就是向左移动。这个动作预计能在未来获得最高不超过 273 的期望回报；而其他两个动作的期望回报不超过 -139 和 195。智能体的决策可以用这个公式表示：

$$a_t \;=\; \underset{a\in\mathcal{A}}{\operatorname{argmax}}\; Q_\star(s_t, a).$$

如何去学习 $Q_\star$ 函数呢？我们需要用智能体收集到的状态、动作、奖励，用它们作为训练数据，学习一个表格或一个神经网络，用于近似 $Q_\star$。最有名的价值学习方法是深度 Q 网络 (DQN)，在后面章节中会详细介绍。

策略学习 (Policy-Based Learning) 指的是学习策略函数 $\pi(a|s)$。假如我们有了策略函数，我们就可以直接用它计算所有动作的概率值，然后随机抽样选出一个动作并执行。每次观测到一个状态 s_t，把它输入 π 函数，让 π 对所有动作做评价，得到概率值：

$$\pi(\text{左} \,|\, s_t) \;=\; 0.6, \qquad \pi(\text{右} \,|\, s_t) \;=\; 0.1, \qquad \pi(\text{上} \,|\, s_t) \;=\; 0.3.$$

智能体做随机抽样，然后执行选中的动作。三个动作都有可能被选中。如何去学习策略 π 呢？本书后面的章节会介绍策略梯度等方法，用于学习 π。

3.6 实验环境

如果你设计出一种新的强化学习方法，你应该将其与已有的标准方法做比较，看新的方法是否有优势。比较和评价强化学习算法最常用的是 OpenAI Gym，它相当于强化学习中的 ImageNet。Gym 有几大类控制问题，比如经典控制问题、Atari 游戏、机器人。

图 3.8: 经典控制问题。

Gym 中第一类是经典控制问题，都是小规模的简单问题，比如 Cart Pole 和 Pendulum，见图 3.8。Cart Pole 要求给小车向左或向右的力，移动小车，让上面的杆子能竖起来。Pendulum 要求给钟摆一个力，让钟摆恰好能竖起来。

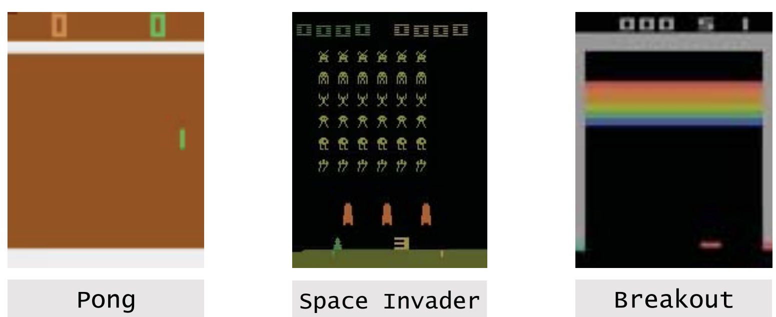

图 3.9: Atari 游戏。

第二类问题是 Atari 游戏，就是八、九十年代小霸王游戏机上拿手柄玩的那种游戏，见图 3.9。Pong 中的智能体是乒乓球拍，球拍可以上下运动，目标是接住对手的球，尽量让对手接不住球。Space Invader 中的智能体是小飞机，可以左右移动，可以发射炮弹。Breakout 中的智能体是下面的球拍，可以左右移动，目标是接住球，并且把上面的砖块都打掉。

图 3.10: 机器人连续的控制问题，用到 MuJoCo 物理模拟器。

第三类问题是机器人连续的控制问题，比如控制蚂蚁、人、猎豹等机器人走路，见图 3.10。这个模拟器叫做 MuJoCo，它可以模拟重力等物理量。机器人是智能体，AI 需要控制这些机器人站立和走路。MuJoCo 是付费软件，但是可以申请免费试用 license。

想要使用 Gym，应该先按照官方文档安装 https://gym.openai.com/。安装之后就可以在 Python 里面调用 Gym 库中的函数了。下面的程序以 Cart Pole 这个控制任务为例，说明怎么样使用 Gym 标准库。通过阅读这段程序，读者可以更好理解智能体与环境的交互。

```
import gym

env = gym.make('CartPole-v0')
state = env.reset()

for t in range(100):
        env.render()
        print(state)

        action = env.action_space.sample()

        state, reward, done, info = env.step(action)

        if done:
                print('Finished')
                break

env.close()
```

生成环境。此处的环境是CartPole游戏程序。

重置环境，让小车回到起点。并输出初始状态。

弹出窗口，把游戏中发生的显示到屏幕上。

方便起见，此处均匀抽样生成一个动作。在实际应用中，应当依据状态，用策略函数生成动作。

智能体真正执行动作。然后环境更新状态，并反馈一个奖励。

done等于1意味着游戏结束；
done等于0意味着游戏继续。