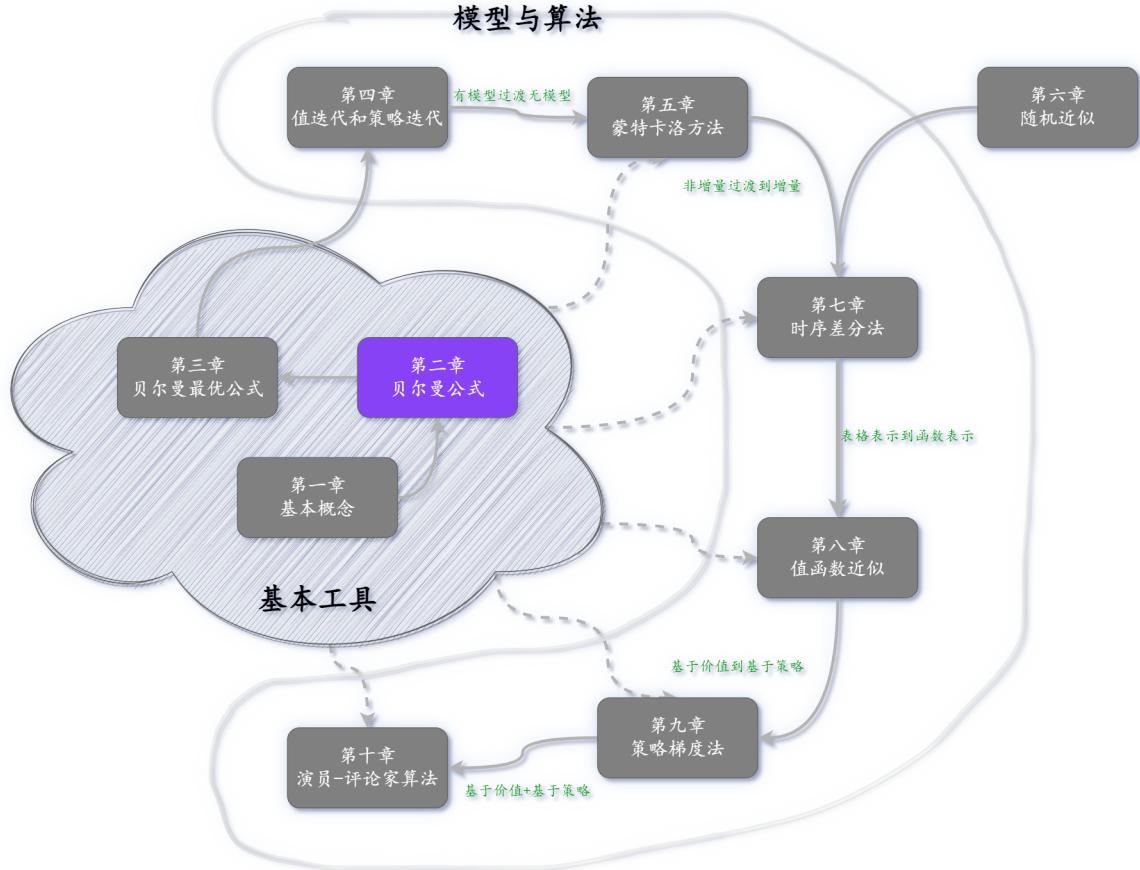


第二章 状态值与贝尔曼方程



本章介绍了一个核心概念和一个重要工具。核心概念是状态值（State Value），它被定义为如果智能体遵循给定的策略，它可以得到的平均奖励。状态值越大，对应的策略越好。状态值可以用作评估策略是否良好的度量。虽然状态值很重要，但我们如何分析它们？答案是贝尔曼方程（Bellman Equation），它是分析状态值的重要工具。简而言之，贝尔曼方程描述了所有状态的价值之间的关系。通过求解贝尔曼方程，我们可以得到状态值。这个过程被称为策略评估（Policy Evaluation），这是强化学习中的一个基本概念。最后，本章介绍了另一个重要的概念，即动作价值（Action Value）。

2.1 示例1：为什么回报很重要？

前一章介绍了回报的概念。事实上，回报在强化学习中起着重要作用，因为它们可以评估策略的好坏。以下示例说明了这一点。

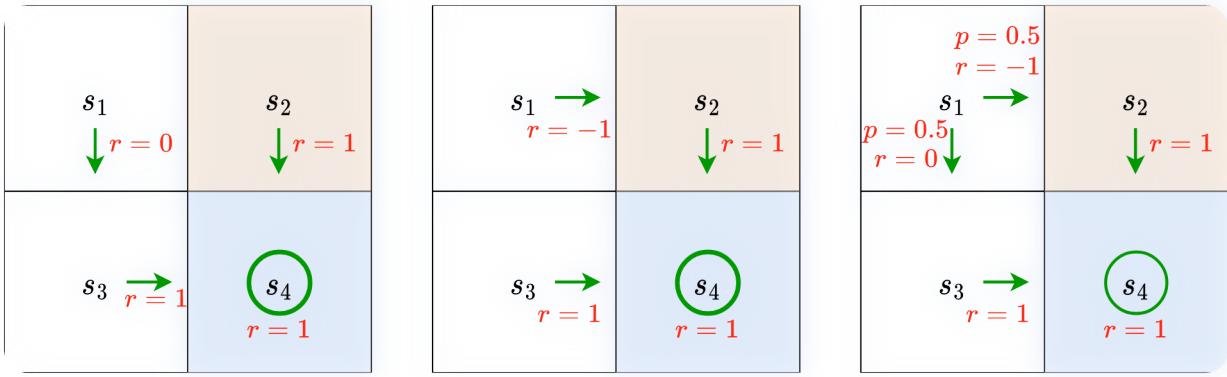


图2.2

考虑图2.2中所示的三种策略。可以看出，这三种策略在 s_1 是不同的。哪个是最好的，哪个是最坏的？直观地说，最左边的策略是最好的，因为从 s_1 开始的智能体可以避开禁区。中间策略直观上更糟，因为从 s_1 开始的智能体直接移动到了禁区。最右边的策略介于其他政策之间，因为其进入禁区的概率为 0.5。

虽然上述分析是基于直觉的，但随之而来的一个问题是，我们是否可以用数学来描述这种直觉。答案是肯定的，这取决于回报的概念。特别是，假设智能体从 s_1 开始。

- 根据第一个策略，轨迹为 $s_1 \rightarrow s_3 \rightarrow s_4 \rightarrow s_4 \dots$ 。相应的折扣回报为

$$\begin{aligned}\text{return}_1 &= 0 + \gamma 1 + \gamma^2 1 + \dots \\ &= \gamma(1 + \gamma + \gamma^2 + \dots) \\ &= \frac{\gamma}{1 - \gamma}, \quad \gamma \in (0, 1)\end{aligned}$$

- 根据第二个策略，轨迹为 $s_1 \rightarrow s_2 \rightarrow s_4 \rightarrow s_4 \dots$ 。相应的折扣回报为

$$\begin{aligned}\text{return}_2 &= -1 + \gamma 1 + \gamma^2 1 + \dots \\ &= -1 + \gamma(1 + \gamma + \gamma^2 + \dots) \\ &= -1 + \frac{\gamma}{1 - \gamma}, \quad \gamma \in (0, 1)\end{aligned}$$

- 根据第三种策略，可以获得两个轨迹。一个是 $s_1 \rightarrow s_3 \rightarrow s_4 \rightarrow s_4 \dots$ ，另一个是 $s_1 \rightarrow s_2 \rightarrow s_4 \rightarrow s_4 \dots$ 。这两个轨迹中的任何一个轨迹的概率都是 0.5。那么，从 s_1 开始可以获得的平均回报是

$$\begin{aligned}\text{return}_3 &= 0.5 \times \left(\frac{\gamma}{1 - \gamma}\right) + 0.5 \times \left(-1 + \frac{\gamma}{1 - \gamma}\right) \\ &= -0.5 + \frac{\gamma}{1 - \gamma}, \quad \gamma \in (0, 1)\end{aligned}$$

通过比较这三种策略的回报，我们注意到

$$\text{return}_1 > \text{return}_3 > \text{return}_2 \tag{2.1}$$

不等式(2.1)表明，第一种策略是最好的，因为其回报最大，而第二种策略是最差的，因为它的回报最小。这个数学结论与前面提到的直觉一致：第一种策略是最好的，因为它可以避免进入禁区，第二种策略是最差的，因为这会导致禁区。

上面的例子表明，回报可以用于评估策略：如果遵循该策略获得的回报更大，则该策略更好。最后，值得注意的是 return_3 并没有严格遵守回报的定义，因为它更像是一个期望值。稍后将清楚地看到， return_3 实际上是一个状态值。

2.2 示例二：如何计算回报？

虽然我们已经证明了回报的重要性，但随之而来的一个问题是，在给定策略时，如何计算回报。

计算回报的方法主要有两种：

- 第一个是简单的定义：回报等于沿着轨迹收集的所有奖励的折扣总和。请考虑图2.3中的示例。设 v_i 表示当 $i=1, 2, 3, 4$ 时从 s_i 开始获得的回报。然后，从图2.3中的四种状态开始时获得的回报可以计算为

$$\begin{aligned} v_1 &= r_1 + \gamma r_2 + \gamma^2 r_3 + \dots, \\ v_2 &= r_2 + \gamma r_3 + \gamma^2 r_4 + \dots, \\ v_3 &= r_3 + \gamma r_4 + \gamma^2 r_1 + \dots, \\ v_4 &= r_4 + \gamma r_1 + \gamma^2 r_2 + \dots, \end{aligned} \tag{2.2}$$

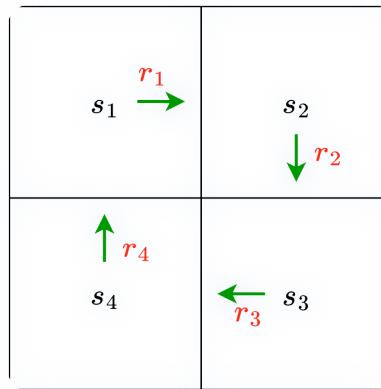


图2.3

- 第二种方式更为重要，它基于**自举（Bootstrapping）**的思想。通过观察式子(2.2)，我们可以重写为

$$\begin{aligned} v_1 &= r_1 + \gamma(r_2 + \gamma r_3 + \dots) = r_1 + \gamma v_2, \\ v_2 &= r_2 + \gamma(r_3 + \gamma r_4 + \dots) = r_2 + \gamma v_3, \\ v_3 &= r_3 + \gamma(r_4 + \gamma r_1 + \dots) = r_3 + \gamma v_4, \\ v_4 &= r_4 + \gamma(r_1 + \gamma r_2 + \dots) = r_4 + \gamma v_1, \end{aligned} \tag{2.3}$$

上述方程表明了一个有趣的现象，即回报的值相互依赖。更具体地说， v_1 依赖于 v_2 ， v_2 依赖于 v_3 ， v_3 依赖于 v_4 ， v_4 依赖于 v_1 。这反映了自举的思想，即从自身获得一些值。

乍一看，自举是一个无休止的循环，因为未知值的计算依赖于另一个未知值。事实上，如果我们从数学的角度来看，自举更容易理解。特别地，(2.3) 中的方程可以被重组为线性矩阵-向量方程：

$$\underbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}}_v = \underbrace{\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix}}_r + \underbrace{\begin{bmatrix} \gamma v_2 \\ \gamma v_3 \\ \gamma v_4 \\ \gamma v_1 \end{bmatrix}}_{\gamma v} = \underbrace{\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix}}_r + \gamma \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_P \underbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}}_v,$$

他也可以等价地写成

$$v = r + \gamma P v$$

因此， v 的值可以很容易地计算为 $v = (I - \gamma P)^{-1} r$ ，其中 I 是具有适当维度的单位矩阵。人们可能会问， $I - \gamma P$ 是否总是可逆的。答案是肯定的，并在第2.7.1节中进行了解释。

事实上，(2.3) 就是这个简单例子的贝尔曼方程。尽管它很简单，(2.3) 证明了贝尔曼方程的核心思想：从一个状态开始获得的回报取决于从其他状态开始时获得的回报。一般情况下的自举思想和贝尔曼方程将在下文正式介绍。

2.3 状态值

我们提到回报可以用于评估策略。然而，它们不适用于随机系统，因为从一个状态开始可能会导致不同的回报。

受这个问题的启发，我们在本节中引入了**状态值（State Value）**的概念。

首先，我们需要引入一些必要的符号。考虑一系列时间步长 $t = 0, 1, 2, \dots$ 。在时间 t ，智能体处于状态 S_t ，并且按照策略 π 采取的动作是 A_t 。下一个状态是 S_{t+1} ，并且获得的即时奖励是 R_{t+1} 。这个过程可以简明地表示为

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

注意， S_t 、 s_{t+1} 、 A_t 、 R_{t+1} 都是**随机变量（Random Variables）**。此外， $S_t, S_{t+1} \in \mathcal{S}$ ， $A_t \in \mathcal{A}(S_t)$ ， $R_{t+1} \in \mathcal{R}(S_t, A_t)$ 。

从 t 开始，我们可以得到一个状态-动作-奖励轨迹：

$$S_t \xrightarrow{A_t} S_{t+1}, R_{t+1} \xrightarrow{A_{t+1}} S_{t+2}, R_{t+2} \xrightarrow{A_{t+2}} S_{t+3}, R_{t+3} \dots$$

根据定义，沿着轨迹的折扣回报是

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots, \quad \gamma \in (0, 1)$$

注意， G_t 是随机变量，因为 R_{t+1} ， R_{t+2} 都是随机变量。由于 G_t 是一个随机变量，我们可以计算它的期望值（也称为期望值或均值）：

$$v_\pi(s) \doteq \mathbb{E}[G_t | S_t = s].$$

这里， $v_\pi(s)$ 被称为**状态价值函数（State-Value Function）** 或简单地称为 s 的状态值。下面给出了一些重要的注释。

- $v_\pi(s)$ 取决于 s 。这是因为它的定义是一个条件期望，条件是智能体从 $S_t = s$ 开始。
- $v_\pi(s)$ 依赖于 π 。这是因为轨迹是通过遵循策略 π 生成的。对于不同的策略，状态值可能是不同的。
- $v_\pi(s)$ 不依赖于 t 。如果智能体在状态空间中移动，则 t 表示当前时间步长。一旦给定策略，就确定 $v_\pi(s)$ 的值。

状态值和回报之间的关系进一步阐明如下。当策略和系统模型都是确定性的时，从一个状态开始总是会导致相同的轨迹。在这种情况下，从一个状态开始获得的回报等于该状态的值。相比之下，当策略或系统模型是随机的时，从同一状态开始可能会产生不同的轨迹。在这种情况下，不同轨迹的回报是不同的，状态值是这些回报的平均值。

2.4 贝尔曼方程

现在介绍贝尔曼方程，这是一种用于分析状态值的数学工具。简而言之，贝尔曼方程是一组描述所有状态值之间关系的线性方程。接下来我们推导出贝尔曼方程。首先，注意 G_t 可以重写为：

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \\ &= R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

其中 $G_{t+1} = R_{t+2} + \gamma R_{t+3} + \dots$ 。这个方程建立了 G_t 和 G_{t+1} 之间的关系。然后，状态值可以写为：

$$\begin{aligned} v_\pi(s) &= \mathbb{E}[G_t | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \mathbb{E}[R_{t+1} | S_t = s] + \gamma \mathbb{E}[G_{t+1} | S_t = s] \end{aligned} \tag{2.4}$$

下面分析公式(2.4)的相关术语：

- 第一个术语 $\mathbb{E}[R_{t+1} | S_t = s]$ 是对即时奖励的期望。通过使用总期望定律（附录A），它可以计算为：

$$\begin{aligned} \mathbb{E}[R_{t+1} | S_t = s] &= \sum_{a \in \mathcal{A}} \pi(a | s) \mathbb{E}[R_{t+1} | S_t = s, A_t = a] \\ &= \sum_{a \in \mathcal{A}} \pi(a | s) \sum_{r \in \mathcal{R}} p(r | s, a) r. \end{aligned} \tag{2.5}$$

这里， \mathcal{A} 和 \mathcal{R} 分别是可能的动作和奖励的集合。需要注意的是，对于不同的状态， \mathcal{A} 可能是不同的。在这种情况下， \mathcal{A} 应该写成 $\mathcal{A}(s)$ 。类似地， \mathcal{R} 也可能取决于 (s, a) 。在这本书中，为了简洁起见，我们放弃了对 s 或 (s, a) 的依赖。尽管如此，在存在依赖性的情况下，这些结论仍然有效。

- 第二个术语 $\mathbb{E}[G_{t+1} | S_t = s]$ 是对未来奖励的期望。它可以计算为：

$$\begin{aligned} \mathbb{E}[G_{t+1} | S_t = s] &= \sum_{s' \in \mathcal{S}} \mathbb{E}[G_{t+1} | S_t = s, S_{t+1} = s'] p(s' | s) \\ &= \sum_{s' \in \mathcal{S}} \mathbb{E}[G_{t+1} | S_{t+1} = s'] p(s' | s) \\ &= \sum_{s' \in \mathcal{S}} v_\pi(s') p(s' | s) \\ &= \sum_{s' \in \mathcal{S}} v_\pi(s') p(s' | s, a) \pi(a | s) \end{aligned} \tag{2.6}$$

上述推导使用了 $\mathbb{E}[G_{t+1} | S_t = s, S_{t+1} = s'] = \mathbb{E}[G_{t+1} | S_{t+1} = s']$ ，这是由于马尔可夫性质（Markov Property），即未来的奖励仅取决于当前状态而不是以前的状态。

将 (2.5) – (2.6) 代入 (2.4) 得到

$$\begin{aligned} v_\pi(s) &= \mathbb{E}[R_{t+1} | S_t = s] + \gamma \mathbb{E}[G_{t+1} | S_t = s] \\ &= \underbrace{\sum_{a \in \mathcal{A}} \pi(a | s) \sum_{r \in \mathcal{R}} p(r | s, a) r}_{\text{mean of immediate rewards}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} v_\pi(s') p(s' | s, a) \pi(a | s)}_{\text{mean of future rewards}}, \\ &= \sum_{a \in \mathcal{A}} \pi(a | s) \left[\sum_{r \in \mathcal{R}} p(r | s, a) r + \gamma \sum_{s' \in \mathcal{S}} p(s' | s, a) v_\pi(s') \right], \quad \text{for all } s \in \mathcal{S} \end{aligned} \tag{2.7}$$

这个方程就是贝尔曼方程（Bellman Equation），它描述了状态值之间的关系。它是设计和分析强化学习算法的基本工具。贝尔曼方程乍一看似乎很复杂。事实上，它有一个清晰的结构。以下是一些注意点：

- $v_\pi(s)$ 和 $v_\pi(s')$ 是待计算的未知状态值。对于初学者来说，如何计算未知的 $v_\pi(s)$ 可能会感到困惑，因为它依赖于另一个未知的 $v_\pi(s')$ 。必须注意的是，贝尔曼方程指的是所有状态的一组线性方程，而不是单个方程。如果我们把这些方程放在一起，就可以清楚地计算出所有的状态值。详细信息将在第2.7节中给出。
- $\pi(a | s)$ 是一个给定的策略。由于状态值可以用于评估策略，因此从贝尔曼方程中求解状态值是一个策略评估（Policy Evaluation）过程，这是许多强化学习算法中的一个重要过程，我们将在本书后面看到。
- $p(r | s, a)$ 和 $p(s' | s, a)$ 表示系统模型。我们将在第2.7节中首先展示如何使用该模型计算状态值，然后在本书稍后部分展示如何在没有模型的情况下使用无模型算法来计算状态值。

除了(2.7)中的表达式外，读者还可能在文献中遇到贝尔曼方程的其他表达式。接下来我们介绍两个等价的表达式。首先，根据全概率公式得到

$$\begin{aligned} p(s' | s, a) &= \sum_{r \in \mathcal{R}} p(s', r | s, a), \\ p(r | s, a) &= \sum_{s' \in \mathcal{S}} p(s', r | s, a). \end{aligned}$$

然后，方程(2.7)可以改写为：

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a | s) \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, a) [r + \gamma v_\pi(s')].$$

其次，在某些问题中，奖励 r 可能完全取决于下一个状态 s' 。因此，我们可以将奖励写为 $r(s')$ ，因此 $p(r(s') | s, a) = p(s' | s, a)$ ，将其代入(2.7)中，得到：

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a | s) \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, a) [r(s') + \gamma v_\pi(s')].$$

2.5 贝尔曼方程的实例

接下来，我们用两个例子来演示如何写出贝尔曼方程并逐步计算状态值。建议读者仔细阅读这些例子，以更好地理解贝尔曼方程。

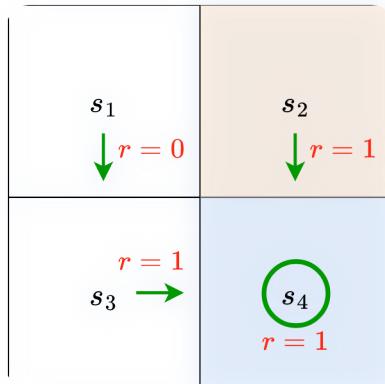


图2.4

- 考虑图2.4显示的第一个示例，其中的策略是确定性的。接下来我们写出贝尔曼方程，然后从中求解状态值。

首先，考虑状态 s_1 。在该策略下，采取动作的概率为 $\pi(a = a_3 | s_1) = 1$ 和 $\pi(a \neq a_3 | s_1) = 0$ 。状态转移概率为 $p(s' = s_3 | s_1, a_3) = 1$ 和 $p(s' \neq s_3 | s_1, a_3) = 0$ 。奖励概率为 $p(r=0 | s_1, a_3) = 1$ 和 $p(r \neq 0 | s_1, a_3) = 0$ 。将这些值代入(2.7)得出：

$$v_\pi(s_1) = 0 + \gamma v_\pi(s_3).$$

有趣的是，尽管(2.7)中贝尔曼方程的表达式看起来很复杂，但这种特定状态的表达式非常简单。

类似地，可以得到：

$$\begin{aligned} v_\pi(s_2) &= 1 + \gamma v_\pi(s_4), \\ v_\pi(s_3) &= 1 + \gamma v_\pi(s_4), \\ v_\pi(s_4) &= 1 + \gamma v_\pi(s_4). \end{aligned}$$

我们可以从这些方程中求解状态值。由于方程很简单，我们可以手动求解。更复杂的方程可以通过第2.7节中介绍的算法求解。在这里，状态值可以求解为：

$$\begin{aligned} v_\pi(s_2) &= v_\pi(s_3) = v_\pi(s_4) = \frac{1}{1 - \gamma} \\ v_\pi(s_1) &= \frac{\gamma}{1 - \gamma} \end{aligned}$$

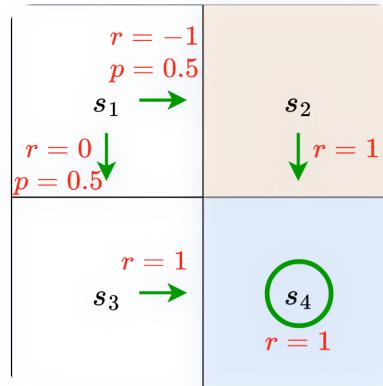


图2.5

- 考虑图2.5中所示的第二个示例，其中的策略是随机的。接下来我们写出贝尔曼方程，然后从中求解状态值。

在状态 s_1 ，向右和向下的概率等于 0.5。在数学上，我们有 $\pi(a = a_2 | s_1) = 0.5$ 和 $\pi(a = a_3 | s_1) = 0.5$ 。由于 $p(s' = s_3 | s_1, a_3) = 1$ 和 $p(s' = s_2 | s_1, a_3) = 1$ ，状态转移概率是确定的。奖励概率也是确定性的，因为 $p(r = 0 | s_1, a_3) = 1$ 并且 $p(r = -1 | s_1, a_2) = 1$ 。将这些值代入(2.7)得出：

$$v_\pi(s_1) = 0.5 \times [0 + \gamma v_\pi(s_3)] + 0.5 \times [-1 + \gamma v_\pi(s_2)].$$

类似地，可以得到：

$$\begin{aligned} v_\pi(s_2) &= 1 + \gamma v_\pi(s_4), \\ v_\pi(s_3) &= 1 + \gamma v_\pi(s_4), \\ v_\pi(s_4) &= 1 + \gamma v_\pi(s_4). \end{aligned}$$

得到状态值：

$$v_{\pi}(s_2) = v_{\pi}(s_3) = v_{\pi}(s_4) = \frac{1}{1-\gamma}$$

$$v_{\pi}(s_1) = -0.5 + \frac{\gamma}{1-\gamma}$$

如果我们比较上述示例中两种策略的状态值，可以看出：

$$v_{\pi_1}(s_i) \geq v_{\pi_2}(s_i), \quad i = 1, 2, 3, 4,$$

这表明图2.4中的策略更好，因为它具有更大的状态值。这个数学结论与直觉一致，即第一个策略更好，因为当代理从 s_1 开始时，它可以避免进入禁区。因此，以上两个示例表明，状态值可以用于评估策略。

2.6 贝尔曼方程的矩阵 - 向量形式

(2.7)中的贝尔曼方程是元素形式 (Elementwise Form) 的。由于它对每个状态都有效，我们可以将所有这些方程组合起来，并将它们简明地写成矩阵 - 向量形式 (Matrix - Vector Form)，这将经常用于分析贝尔曼方程。

为了推导矩阵向量形式，我们首先将 (2.7) 中的贝尔曼方程改写为：

$$v_{\pi}(s) = r_{\pi}(s) + \gamma \sum_{s' \in \mathcal{S}} p_{\pi}(s' | s) v_{\pi}(s') \quad (2.8)$$

其中：

$$r_{\pi}(s) \doteq \sum_{a \in \mathcal{A}} \pi(a | s) \sum_{r \in \mathcal{R}} p(r | s, a) r,$$

$$p_{\pi}(s' | s) \doteq \sum_{a \in \mathcal{A}} p(s' | s, a) \pi(a | s).$$

这里， $r_{\pi}(s)$ 表示即时奖励的平均值， $p_{\pi}(s' | s)$ 是在策略 π 下从 s 过渡到 s' 的概率。

我们标记一下所有的状态 s_i , $i = 1, 2, \dots, n$, $n = |\mathcal{S}|$ ，对于每个 s_i 都可写出一个式子 (2.8)：

$$v_{\pi}(s_i) = r_{\pi}(s_i) + \gamma \sum_{s_j \in \mathcal{S}} p_{\pi}(s_j | s_i) v_{\pi}(s_j) \quad (2.9)$$

让 $v_{\pi} = [v_{\pi}(s_1), \dots, v_{\pi}(s_n)]^T \in \mathbb{R}^n$, $r_{\pi} = [r_{\pi}(s_1), \dots, r_{\pi}(s_n)]^T \in \mathbb{R}^n$, $P_{\pi} \in \mathbb{R}^{n \times n}$, $[P_{\pi}]_{i,j} = p_{\pi}(s_j | s_i)$, 然后，(2.9)可以写成以下矩阵向量形式：

$$v_{\pi} = r_{\pi} + \gamma P_{\pi} v_{\pi} \quad (2.10)$$

其中 v_{π} 是待解的未知数， r_{π} 、 P_{π} 是已知的。

矩阵 P_{π} 具有一些有趣的性质。首先，它是一个非负矩阵，意味着它的所有元素都等于或大于零。这个性质表示为 $P_{\pi} \geq 0$ ，其中 0 表示具有适当维数的零矩阵。在本书中， \geq 或 \leq 表示元素比较运算。第二， P_{π} 是一个随机矩阵，意味着每行中的值之和等于1。这个性质表示为 $P_{\pi} \mathbf{1} = \mathbf{1}$ ，其中 $\mathbf{1} = [1, \dots, 1]^T$ 具有适当的维数。

考虑图2.6中所示的示例。贝尔曼方程的矩阵 - 向量形式为：

$$\underbrace{\begin{bmatrix} v_{\pi}(s_1) \\ v_{\pi}(s_2) \\ v_{\pi}(s_3) \\ v_{\pi}(s_4) \end{bmatrix}}_{v_{\pi}} = \underbrace{\begin{bmatrix} r_{\pi}(s_1) \\ r_{\pi}(s_2) \\ r_{\pi}(s_3) \\ r_{\pi}(s_4) \end{bmatrix}}_{r_{\pi}} + \gamma \underbrace{\begin{bmatrix} p_{\pi}(s_1 | s_1) & p_{\pi}(s_2 | s_1) & p_{\pi}(s_3 | s_1) & p_{\pi}(s_4 | s_1) \\ p_{\pi}(s_1 | s_2) & p_{\pi}(s_2 | s_2) & p_{\pi}(s_3 | s_2) & p_{\pi}(s_4 | s_2) \\ p_{\pi}(s_1 | s_3) & p_{\pi}(s_2 | s_3) & p_{\pi}(s_3 | s_3) & p_{\pi}(s_4 | s_3) \\ p_{\pi}(s_1 | s_4) & p_{\pi}(s_2 | s_4) & p_{\pi}(s_3 | s_4) & p_{\pi}(s_4 | s_4) \end{bmatrix}}_{P_{\pi}} \underbrace{\begin{bmatrix} v_{\pi}(s_1) \\ v_{\pi}(s_2) \\ v_{\pi}(s_3) \\ v_{\pi}(s_4) \end{bmatrix}}_{v_{\pi}}.$$

将特定的值带入到上述方程：

$$\underbrace{\begin{bmatrix} v_\pi(s_1) \\ v_\pi(s_2) \\ v_\pi(s_3) \\ v_\pi(s_4) \end{bmatrix}}_{v_\pi} = \underbrace{\begin{bmatrix} 0.5 \times 0 + 0.5 \times (-1) \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{r_\pi} + \gamma \underbrace{\begin{bmatrix} 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{P_\pi} \underbrace{\begin{bmatrix} v_\pi(s_1) \\ v_\pi(s_2) \\ v_\pi(s_3) \\ v_\pi(s_4) \end{bmatrix}}_{v_\pi}.$$

可以看出， P_π 满足 $P_\pi \mathbf{1} = \mathbf{1}$ 。

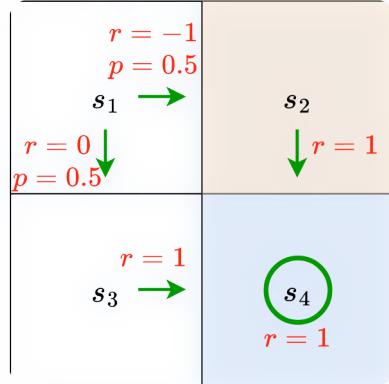


图2.7

2.7 从贝尔曼方程求解状态值

计算给定策略的状态值是强化学习中的一个基本问题。这个问题通常被称为策略评估（Policy Evaluation）。在本节中，我们介绍了从贝尔曼方程计算状态值的两种方法。

2.7.1 闭式解（解析解）

由于 $v_\pi = r_\pi + \gamma P_\pi v_\pi$ 是一个简单的线性方程，因此它的解析解可以很容易地得到为：

$$v_\pi = (I - \gamma P_\pi)^{-1} r_\pi$$

$(I - \gamma P_\pi)$ 的一些性质：

- $(I - \gamma P_\pi)$ 是可逆的阵：

补充：Gershgorin圆定理¹：

设 A 是 n 阶复数矩阵， $A = (a_{ij})_{n \times n}$ ，则 A 的特征值在复平面的下列圆盘中：

$$|z - a_{ii}| < \sum_{i \neq j}^n |a_{ij}|$$

[证明]：

根据 Gershgorin 圆定理¹： $(I - \gamma P_\pi)$ 的每一个特征值都至少在一个 Gershgorin 圆里面，第 i 个 Gershgorin 圆的中心是： $[I - \gamma P_\pi]_{ii} = 1 - \gamma p_\pi(s_i | s_i)$ ，半径是 $\sum_{i \neq j} [I - \gamma P_\pi]_{ij} = -\sum_{i \neq j} \gamma p_\pi(s_j | s_i)$ ， $\because \gamma < 1$ ，我们知道半径要小于中心的大小，

$$\therefore \sum_{i \neq j} \gamma p_\pi(s_j | s_i) < 1 - \gamma p_\pi(s_i | s_i)$$

\therefore 所有 Gershgorin 圆都不包围原点，因此 $I - \gamma P_\pi$ 的特征值不为零。

- $(I - \gamma P_\pi)^{-1} > I$ ，这就意味着 $(I - \gamma P_\pi)^{-1}$ 的每个元素都是非负的，更具体地说，不小于单位矩阵的元素。这是因为 P_π 具有非负项，因此， $(I - \gamma P_\pi)^{-1} = I + \gamma P_\pi + \gamma^2 P_\pi + \dots > I > 0$

2.7.2 迭代解

尽管解析解可用于理论分析目的，但它在实践中不适用，因为它涉及矩阵求逆运算，仍需要通过其他数值算法进行计算。事实上，我们可以使用以下迭代算法直接求解贝尔曼方程：

$$v_{k+1} = r_\pi + \gamma P_\pi v_k, \quad k = 0, 1, 2, \dots, \quad (2.11)$$

该算法生成一个值序列 $\{v_0, v_1, v_2, \dots\}$ ，其中 $v_0 \in \mathbb{R}^n$ 是 v_π 的初始猜测值。它认为：

$$v_k \rightarrow v_\pi = (I - \gamma P_\pi)^{-1} r_\pi, \quad k \rightarrow \infty \quad (2.12)$$

[证明](2.12)：

将误差定义为 $\delta_k = v_k - v_\pi$ 。我们只需要说明 $\delta_k \rightarrow 0$ ，将 $v_{k+1} = \delta_{k+1} + v_\pi$ 和 $v_k = \delta_k + v_\pi$ 代入 $v_{k+1} = r_\pi + \gamma P_\pi v_k$ 得到：

$$\delta_{k+1} + v_\pi = r_\pi + \gamma P_\pi(\delta_k + v_\pi),$$

可以写成：

$$\begin{aligned} \delta_{k+1} &= -v_k + r_\pi + \gamma P_\pi \delta_k + \gamma P_\pi v_\pi, \\ &= \gamma P_\pi \delta_k - v_k + (r_\pi + \gamma P_\pi v_\pi), \\ &= \gamma P_\pi \delta_k. \end{aligned}$$

因此：

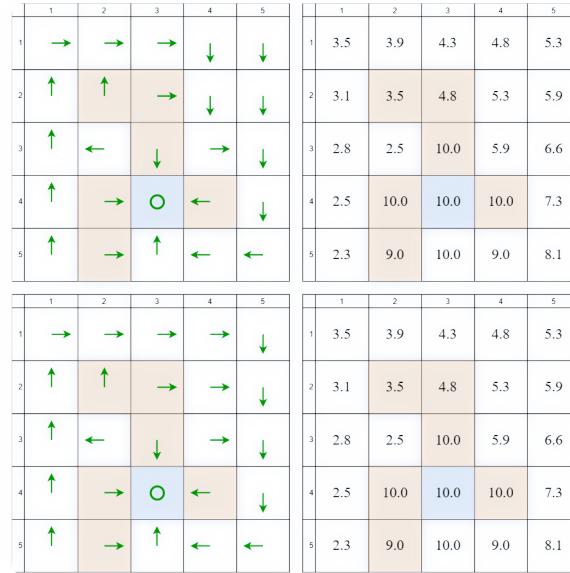
$$\delta_{k+1} = \gamma P_\pi \delta_k = \gamma^2 P_\pi^2 \delta_{k-1} = \dots = \gamma^{k+1} P_\pi^{k+1} \delta_0$$

由于 P_π 的每个项都是非负的并且不大于 1，因此对于任何 k ，我们都有 $0 \leq P_\pi^k \leq 1$ 。也就是说， P_π^k 的每个项都不大于 1。另一方面，由于 $\gamma < 1$ ，我们知道 $\gamma^k \rightarrow 0$ ，因此，当 $k \rightarrow \infty$ 时， $\delta_{k+1} = \gamma^{k+1} P_\pi^{k+1} \delta_0 \rightarrow 0$

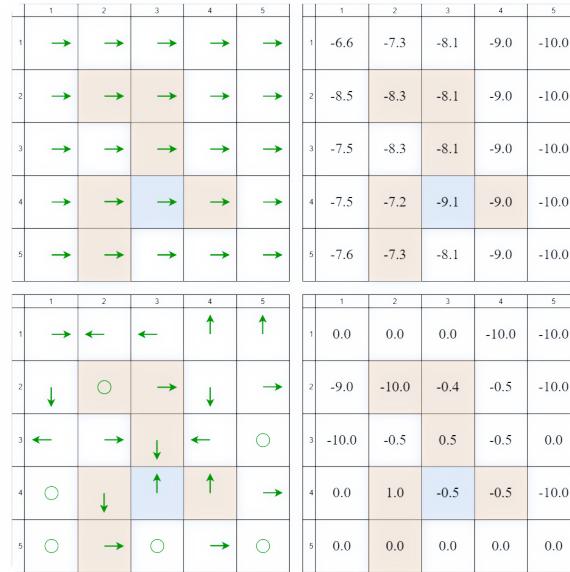
2.7.3 说明示例

接下来，我们应用(2.11)中的算法来求解一些例子的状态值。

示例如图 2.7 所示。橙色的单元格代表禁区。蓝色单元格表示目标区域。奖励设置为 $r_{boundary} = r_{forbidden} = -1$ 和 $r_{target} = 1$ ，折扣因子设为 0.9。



(a)两项“好”策略及其状态价值。这两个策略的状态值是相同的，但在第四列的前两个策略是不同的。



(b)两项“坏”政策及其状态值。状态值比“好”策略的状态值要小。

图2.7：策略及其对应的状态值的示例

图2.7(A)显示了由(2.11)获得的两个“好”策略及其对应的状态值。这两个策略具有相同的状态值，但不同在第四列的前两个状态。因此，我们知道不同的策略可能具有相同的状态值。

图2.7(B)显示了两个“坏”策略及其对应的状态值。这两个策略是糟糕的，因为许多状态的动作在直觉上是不合理的。这种直觉得到了结果的验证。如图2.7(A)所示，这两个策略的状态值都是负值，远远小于好策略的状态值。

2.8 从状态值到动作值

到目前为止，我们在本章中一直在讨论状态值，现在我们转向动作值，它表示在一个状态下采取动作的“价值”。虽然动作值的概念很重要，但本章最后一节之所以引入它，是因为它在很大程度上依赖于状态值的概念。在研究动作值之前，首先了解状态值是很重要的。

状态-动作对 (s, a) 的动作值定义为：

$$q_{\pi}(s, a) \doteq \mathbb{E}[G_t | S_t = s, A_t = a].$$

可以看出，动作值被定义为在一个状态下采取动作后可以获得的预期回报。必须注意的是， $q_{\pi}(s, a)$ 取决于状态-动作对 (s, a) ，而不是单独的动作。将这个值称为状态-动作值可能更严格，但为了简单起见，它通常被称为动作值。

动作值和状态值之间的关系是什么？

- 首先，根据条件期望的性质：

$$\underbrace{\mathbb{E}[G_t | S_t = s]}_{v_{\pi}(s)} = \sum_{a \in \mathcal{A}} \underbrace{\mathbb{E}[G_t | S_t = s, A_t = a]}_{q_{\pi}(s, a)} \pi(a | s)$$

因此：

$$v_{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a | s) q_{\pi}(s, a). \quad (2.13)$$

因此，状态值是与该状态相关联的动作值的期望值。

- 其次，由于状态值由：

$$v_{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a | s) \left[\sum_{r \in \mathcal{R}} p(r | s, a) r + \gamma \sum_{s' \in \mathcal{S}} p(s' | s, a) v_{\pi}(s') \right],$$

将其与(2.13)进行比较可得出：

$$q_{\pi}(s, a) = \sum_{r \in \mathcal{R}} p(r | s, a) r + \gamma \sum_{s' \in \mathcal{S}} p(s' | s, a) v_{\pi}(s') \quad (2.14)$$

可以看出，动作值由两个项组成。第一个术语是即时奖励的平均值，第二个术语是未来奖励的平均值。

(2.13)和(2.14)都描述了状态值和动作值之间的关系。它们是同一枚硬币的两面：(2.13)显示了如何从动作值中获得状态值，而(2.14)显示如何从状态值中获得动作值。

2.8.1 举例说明

接下来，我们将举一个例子来说明计算动作值的过程，并讨论初学者可能犯的一个常见错误。

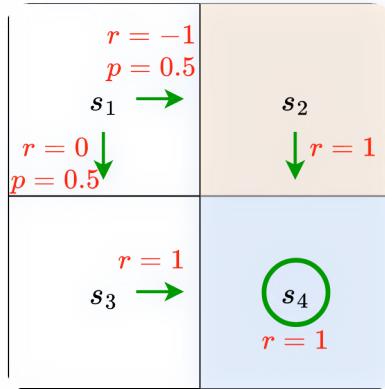


图2.8

考虑图2.8所示的随机策略。接下来我们研究 s_1 的动作。其他状态也可以进行类似的检查。 (s_1, a_2) 的动作值为：

$$q_\pi(s_1, a_2) = -1 + \gamma v_\pi(s_2),$$

其中 s_2 是下一个状态。类似地，可以得到：

$$q_\pi(s_1, a_3) = 0 + \gamma v_\pi(s_3).$$

初学者可能会犯的一个常见错误是关于给定策略未选择的动作的值。例如，图2.8中的策略只能选择 a_2 或 a_3 ，而不能选择 a_1 、 a_4 、 a_5 。有人可能会争辩说，由于策略没有选择 a_1 、 a_4 、 a_5 ，我们不需要计算它们的作用值，或者我们可以简单地设置 $q_\pi(s_1, a_1) = q_\pi(s_1, a_4) = q_\pi(s_1, a_5) = 0$ 。这是错误的。

- 首先，即使策略不会选择某个动作，它仍然具有操作值。在这个例子中，尽管策略 π 在 s_1 处不取 a_1 ，但我们仍然可以通过观察采取该动作后我们将获得的结果来计算其动作值。具体来说，在取得 a_1 后，智能体被反弹回 s_1 （因此，即时奖励为 -1 ），然后通过跟随 π 在从 s_1 开始的状态空间中继续移动（因此，未来奖励为 $\gamma v_\pi(s_1)$ ）。结果， (s_1, a_1) 的动作值为：

$$q_\pi(s_1, a_1) = -1 + \gamma v_\pi(s_1).$$

同样，对于 a_4 和 a_5 ，给定的策略也不可能选择它们，我们有：

$$q_\pi(s_1, a_4) = -1 + \gamma v_\pi(s_1),$$

$$q_\pi(s_1, a_5) = 0 + \gamma v_\pi(s_1).$$

- 第二，我们为什么关心特定策略不会选择的行动？尽管某些动作不可能由给定的策略选择，但这并不意味着这些动作不好。给定的策略可能不好，因此无法选择最佳动作。强化学习的目的是找到最优的策略。为此，我们必须不断探索所有动作，为每个状态确定更好的动作。

2.8.2 贝尔曼方程里的动作值

我们之前介绍的贝尔曼方程是基于状态值定义的。事实上，它也可以用动作值来表达。

特别是，将(2.13)代入(2.14)得到：

$$q_\pi(s, a) = \sum_{r \in \mathcal{R}} p(r | s, a)r + \gamma \sum_{s' \in \mathcal{S}} p(s' | s, a) \sum_{a' \in \mathcal{A}(s')} \pi(a' | s') q_\pi(s', a')$$

上面是动作值的方程式。上面的等式对于每个状态-动作对都是有效的。如果我们把所有这些方程放在一起，它们的矩阵向量形式是：

$$q_\pi = \tilde{r} + \gamma P \Pi q_\pi, \quad (2.15)$$

其中 q_π 是由状态-动作对索引的作用值向量：它的第 (s, a) 个元素是 $[q_\pi]_{(s,a)} = q_\pi(s, a)$ 。 \tilde{r} 是由状态-动作对索引的即时奖励向量： $[\tilde{r}]_{(s,a)} = \sum_{r \in \mathcal{R}} p(r | s, a) r$ 。矩阵 P 是概率转移矩阵，其行由状态-动作对索引，其列由状态索引： $[P]_{(s,a),s'} = p(s' | s, a)$ 。此外， Π 是一个块对角矩阵，其中每个块是一个 $1 \times |\mathcal{A}|$ 向量： $\Pi_{s',(s'|a')} = \pi(a' | s')$ ，并且 Π 的其他项为零。

与根据状态值定义的贝尔曼方程相比，根据动作值定义的方程具有一些独特的特征。例如， \tilde{r} 和 P 与策略无关，仅由系统模型确定。该策略包含在 π 中。可以验证 (2.15) 也是一个压缩映射，并且具有可以迭代求解的唯一解。更多详细信息请参见 [2](#)。

1. R. A. Horn and C. R. Johnson, Matrix analysis. Cambridge University Press, 2012. [↩](#)

2. D. P. Bertsekas and J. N. Tsitsiklis, Neuro-dynamic programming. Athena Scientific. [↩](#)