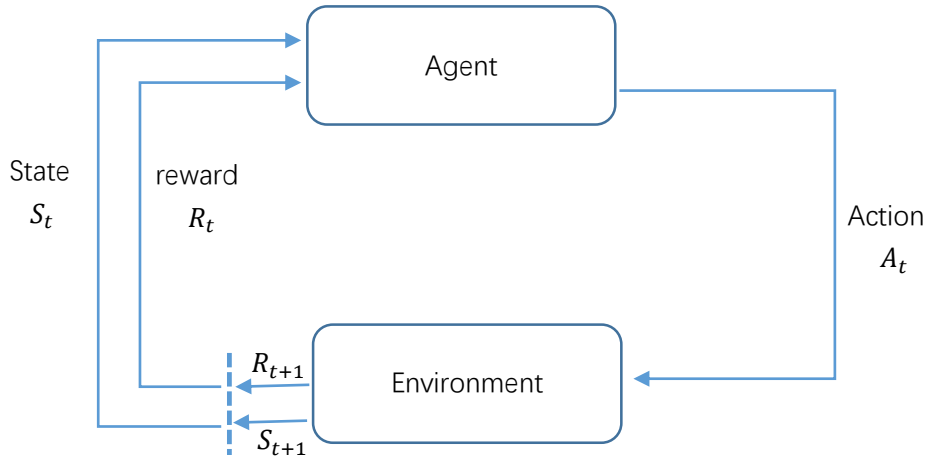
马尔可夫决策过程

本章我们介绍马尔科夫决策基本过程的基本问题，或者说有限 MDPs,这个问题包含了赌博机问题的评价反馈机制，同时又有联想——在不同情况下选择不同动作。马尔科夫是一种经典的系列决策形式，动作影响的不仅仅是立即奖赏，同时也影响后续奖赏或者状态，以及未来奖赏。因此 MDPs 涉及了延迟奖赏和延迟奖赏和立即奖赏之间的平衡。

MDPs 是可以做出精确理论陈述的理想化数学模型的强化学习问题。我们这里介绍这个问题的关键数学结构比如：返回值、值函数、Bellman 方程。我们尝试宽泛 MDPs 的应用，将尽可能多的问题划归为 MDPs 问题。和所有的人工智能一样，MDPs 也存在应用广泛性和数据可追踪性之间的矛盾问题，在本节中我们将介绍这种矛盾并且讨论其中暗含的一些平衡方法和挑战。

Agent 与环境的接口

MDPs 是从与环境的交互中学习以实现目标的简单框架问题。学习者和决策者称为 agent 而它与之交互的东西，除 agent 意外的一切都成为 Environment。交互持续，agent 选择动作然后环境响应这些动作，并向 agent 提供新的情况。环境同样会给与相应的奖赏（一般是数值奖赏），随着时间的推移，agent 会选择恰当的动作以获取最大化奖赏。



更确切的说，agent 与环境在一系列的离散时间步 t=0,1,2,3……中每一步都在进行着交互，在每个时间步 t 中，agent 收到一些环境的状态表示，$S_t \in \mathcal{S}$,在此基础上选择一个动作，$A_t \in \mathcal{A}(s)$。在一个时间步后，采取这个动作的结果是：agent 收到了一个数值上的奖赏，$R_{t+1} \in \mathcal{R} \subset \mathbf{R}$，并且到达下一个新状态，$S_{t+1}$ 。所以 **MDP** 和 **agent** 共同产生一个序列或者轨迹：

$$S_0,\ A_0,\ R_1,\ S_1,\ A_1,\ R_2,\ S_2,\ A_2,\ R_3 \ldots\ldots$$

在一个有限的 **MDP** 中，状态集、动作集、奖赏集都是一个确定的数值元素。在这种情况下，由随机变量 S_t 和 R_t 所构成的离散概率分布只取决于先前的状态和动作，也就是说，对于这些随机变量的特征值，**s'** ∈ **S** 和 **r** ∈ **R**，这些值在 **t** 时刻发生的概率，通过之前的状态动作给定特征值：

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

对于所有的 **s',s∈ S, r∈ R, a∈ A(s).**上式都只是一个定义而不是从之前定义中的一个事实。中间的 '|' 表示条件概率，在这里提示我们 **p** 是一个针对每次选择的 **s** 和 **a** 的概率分布。

$$\sum_{s' \in S} \sum_{r \in R} p(s', r|s, a) = 1 \qquad , for \ \ all \ \ s \in S, a \in A(s)$$

通过四个参数方程给定的概率 **p** 完整的描述了有限马尔科夫决策（**finite MDP**）的动态性。从中我们可以计算出任何想知道的有关环境的信息，比如状态转移概率：

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

我们同样可以通过状态-动作对计算出期望奖赏：**S** X A→R：

$$r(s, a) = E(R_t|S_{t-1} = s, A_{t-1} = a] = \sum_{r \in R} r \sum_{s' \in S} p(s', r|s, a)$$

或者通过状态-动作-下一个状态三个参数计算出期望奖赏:S X A X A→R：

$$r(s, a, s') = E(R_t|S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in R} r \frac{p(s', r|s, a)}{p(s'|s, a)}$$

MDP 框架抽象且灵活，并且可以通过多种不同的方法应用到多种不同的问题上。列如，时间步长不仅仅指固定的真实时间间隔，也可以指任意连续的决策和行动阶段。动作可以是简单的控制，比如机器人手臂的电压供应，也可以是高等的决策，如是否午饭或者去上大学。同样的，状态也可以采取多种形式，他们可以完全取决于低等的感知系统比如直接传感器读取，也可以是高等的或抽象的，比如一个房间的物品描述标记。构成状态的东西可以是基于过去感应的记忆或者是完全精神的、主观的记忆。例如，一个机器人可以处于一个不确定物体是啥的状态，也可以对于一些明显的感觉感到"惊讶"。相似之下，一些动作可能是纯精神上的或者计算上的，例如，一些动作可以去控制 agent 选择思考什么，把它的注意力集中在哪里。大体上，动作可以是我们想要学习的任何决策，而状态可以是任何我们可以知道的对决策有利的东西。

通常情况下，agent 和环境之间并没有明显的界限。而我们遵循的一般规则是任何不能被 agent 随意更改的东西都是在它之外的，因此被认定为环境的一部分。我们通常不会假设 agent 对环境一无所知。例如，agent 通常会知道在状态和动作函数下怎样计算出奖赏。但我们总是把奖赏计算考虑为 agent 之外的环境，因为它定义了 agent 所面临的任务而且超出了它能随意更改的能力范围。事实上，在某些情况下，agent 会知道一些，包括环境怎样工作，但它仍然面临困难的强化学习任务。正如我们知道模仿是怎样工作的，但依旧无法解开它。agent-environment 界限要求 agent 的绝对控制，而不是知识上的限制。

MDP 框架是交互的目标向导学习问题中一个相当抽象的概念，它认为无论感觉、记忆、控制装置的细节是什么，不管它想实现的目标是什么，学习目标导向行为的任何问题都可以归结为三个信号在一个agent和环境之间的来回传递：一个代表了agent所做出的选择(action)、信号二是 agent 做出选择的基础（state)、信号三定义了 agent 的目标（rewards)。这个框架或许不能充分代表所有的决策-学习问题，但它已被证明是广泛有用的和合适的。

目标和奖赏

在强化学习中，agent 的目标根据特殊信号来形式化，我们把这种由环境反馈到训练实体（agent）的信号称为奖赏。在每个时间步里面，奖赏是一个简单的数值，$R_t \in \mathrm{R}$。非形式化的讲，agent 的目标就是最大化其获得的奖赏总值。这意味着最大化的并不是立即奖赏，而是在长时间运行下的积累奖赏。我们可以称这种非形式化的想法为奖励假设：

我们的目标可以被认为是最大化所获标量奖赏信号累计值的期望值

利用奖赏信号去目标形式化的思想是强化学习的最显著特征之一。

尽管使用奖赏信号形式化目标在最开始会有所限制，但实践证明它是灵活的且广泛应用的。我们将通过一些已经使用的或者可以运用这种方式的例子来说明这一点，例如：为了让机器人学会走路，研究者在每一步中提供了恰当的奖赏作为机器人前进的动力。在让机器人学会逃跑的例子中，在每一个时间步逃跑前进时的奖赏为-1，这是为了鼓励 agent 尽可能尽快逃跑。让机器人学习发现回收空的塑料罐，大多数情况下其奖赏值为 0，只有当能回收塑料罐时奖赏为+1。而在机器人撞到人或东西时将奖赏值设置为负数。而对于一个下下跳棋或国际象棋的机器人，其赢棋时奖赏为+1，输棋时奖赏为 0,中间步骤奖赏为 0.

从以上例子中你可以看出，agent 总是在学会最大化其奖赏值。如果我们需要 agent 为我们做什么的时候，设置的奖赏就必须满足当其达到最大值时我们的目标也被实现了。因此，我们设定的奖赏是至关重要的，真正表面我们需要实现的是什么。特别的是，我么需要完成的目标的先验知识并不是通过奖赏信号传递给训练实体。例如：一个象棋 agent 只有在真正赢得比赛的时候才会有奖赏，而不是实现中间目标。如果实现中间目标也有奖赏，那么它会找到实现中间目标而不实现终极目标的方法。奖赏信号是你和机器交流你想实现什么目标，而不是规定他如何完成目标的一种方式。

返回值和集

至此，我们非形式化的讨论的强化学习的目标。我们说 agent 的目标是在长期运作中最大化累计奖赏。但怎么把这一目标形式化的定义呢？如果在时间 t 之后获得的系列奖赏标记为R_{t+1}，R_{t+2}，R_{t+3}，R_{t+4}……，那么在这个序列中我们想要最大化的具体目标是什么？通常，我们寻找最大化期望返回，这个返回值G_t定义为返回序列的特殊的函数，最为简单的方式则是把这个函数返回值定为奖赏之和：

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T \qquad (3.7)$$

T 是最后一个时间步，在“最后一步”有精确定义的应用中，这个方法有实际意义，也就是说：当 agent 与环境之间的交互可以自然的分成子序列，我们称这些子序列为情节（episodes），比如玩游戏，走迷宫，或者任意短暂的循环交互，每一个情节都在特殊状态结束，称为终态，或结束状态。接着一个新的情节会以一个标准的开始状态或者是从开始状态的标准分布中取出一个样本开始。即使你认为情节以不同的方式结束，比如游戏会赢或输，下一个情节的开始独立于先前的情节是如何结束的。因此，在相同的终止状态下的情节都可以被认定为结束情节，只是不同的输出有不同的奖赏。这种情节组成的任务称为情节式任务。在情节式任务中我们需要从包括终态集合所有情节中区别出所有的非终态集合，非终态集合称为 S，所有情节集合称为S^+。而结束时间记为 T，一般情况下不同情节结束时间不一样。

另一方面，在许多情况下 agent 与环境的交互并不能自然的分成可识别的独立情节，而是没有限制的进行下去。比如：这将是一种自然的方式来制定一个正在进行的过程控制任务，或一个应用到一个具有很长的寿命的机器人。我们称为连续任务。那么对于连续任务，公式（3.7）就存在问题，因为结束时间步会是 T = ∞，而我们尝试最大化的返回值则很容易就是无穷大。所以，在此我们通常使用一种概念上复杂一些但实际计算上简单一些的返回值定义。

我们需要的额外概率是折合因子（discounting），根据这个方法，agent 尝试选择使折扣后的总奖赏最大化的动作。特别的，它选择A_t去最大化期望折扣奖赏。

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

在此，γ是一个 0≤γ≤1 的参数,称为折合率

折合率决定了未来奖赏的当前价值：在未来时间步 k 收到的奖赏只是在 0 时刻收到奖赏的γ^{k-1}倍。如果 γ<1，只要奖赏序列$\{R_k\}$有界，无穷级数将会收敛到一个值。如果 γ=0，agent 是：“短见的”，它值关注立即奖赏的最大化：他的目标是学习如何选择A_t使得R_{t+1}最大化。如果每一个 agent 动作发生都只影响到立即奖赏而不影响未来奖赏，那么一个短见的 agent 可以通过划分每一个立即奖赏最大化（3.8）。但通常情况下，采用最大化立即奖赏会减少对未来奖赏的获取，从而减少回报。当 γ 趋向于 1 时，回报目标更加强烈的考虑到了未来奖赏，agent 变得更有远见。

连续时间步的返回值是相互影响的，这点对于强化学习的理论和算法是很重要的

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

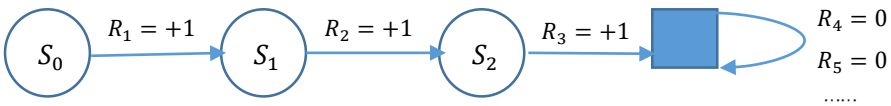
注意到这个公式在所有的结束时刻前 t<T 都是有效的，即使在 t + 1 时刻结束,如果我们定义 $G_t = 0$，这便于我们根据奖赏数列计算奖赏值。

情节式任务和连续任务的统一标识

在开始的章节中我们讨论的两种强化学习任务中，一种是 agent-环境之间的交互自然分成一些列的独立情节（情节式任务），另一种则相反（连续任务），前者在数学上容易些因为每个动作只影响情节中是后续收到的有穷奖赏序列。

为了精确描述情节式任务，还需要一些额外的表示符号。相较于长时间步序列，我们需要考虑一系列的情节，每个情节由有限的时间步长组成。每个情节的时间步从 0 开始。因此，我们不仅需要关心 t 时刻的状态S_t，还需要关心 t 时刻在情节 i 下的状态$S_{t,i}$（同理有$A_{t,i}$，$R_{t,i}$，$\pi_{t,i}$，T_i等）。但是，当讨论情节式任务时，几乎不去区分情节。大多数情况下都只考虑一个特殊的单个情节，或者讨论对于所有情节都适用的情况。于是，在实践中我们通常会省略对于情节索引的显示引用，这稍微有点滥用符号了。也就是用S_t代替$S_{t,i}$，以此类推。

我们需要另一种惯例来获取一种能涵盖情节式任务和连续任务的单一表示法。我们在（3.7）式中把返回定义为有穷项的累积和，而在（3.8）式中则把返回定义为无穷项的累积和。考虑情节在一个特殊的吸收状态（absorbing state）下终止，我们就可以把两种情况统一起来。吸收状态只向自身状态迁移并产生 0 奖赏。如下转换图所示：



这里实心正方形是一个特殊的吸收状态，对应最后一个情节。从情节S_0开始，我们得到了一系列的奖赏 +1，+1，+1,0,0,0……，对系列奖赏求和，不管是对前 T（T=3）个奖赏求和还是对整个无限序列的奖赏求和，所得结果一样。即使我们引入折扣依旧成立。因此，大体上可以根据（3.8）式，在不需要的情况下使用省略情节数量的惯例，如果总和保留定义，存在γ = 1 的可能性。另外，返回也可以写成（3.11）式：

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \qquad (3.11)$$

公式包含了 T=∞或γ=1 的情况（但没有同时包含）我们使用这个惯例来简化符号表示和表达情节式任务和连续任务之间的相似之处。

策略和值函数

几乎所有的强化学习算法都包含了估计值函数——状态的函数（状态-动作对的函数），它能估计在一个给定状态下 agent 的好坏程度（或者在一个给定状态下执行一个给定动作的好坏程度）。“好坏程度”在此是依据可预期的未来奖赏定义的，或者更精确地说是根据期望回报定义。当然 agent 在未来预期获得奖赏取决于将要采取的动作。一般的，值函数从特定的行为方式方面来定义的，这种特定的行为称为策略。

形式化地，策略是从状态到选择每个可能动作的概率的映射。如果 agent 在时间 t 采取了策略π，则π（a|s）在$S_t = s$下采取动作$A_t = a$的概率。和 p 函数一样，π是一个普通函数；π（a | s）中间的“|”仅仅是提醒到它定义了在每个 s ∈ S 状态下所有 a ∈ A(s)的概率分布。强化学习方法说明了 agent 策略怎样依据其经验进行改变。

策略π下的状态 s 值标记为$v_\pi(s)$,是一个从 s 状态开始并遵循策略π得到的期望回报。对于 MDPs，我们可以如下定义v_π:

$$v_\pi(s) = E_\pi[\ G_t|S_t = s] = E_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \;\middle|\; S_t = s\ \right] \quad ,\text{for all } s \in S$$

其中$E_\pi[.]$是 agent 遵循策略π后给出的随机变量期望值，T 是任意时间步。注意到终止状态的值（若存在）为 0.我们称函数v_π为策略π的状态-值函数。

相似的，我们定义$q_\pi(s,a)$为策略π下状态 S 下采取动作 a 的值，是一个从 s 状态开始采取动作 a 并遵循策略π得到的期望回报

$$q_\pi(s,a) = E_\pi[\ G_t|S_t = s, A_t = a] = E_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \;\middle|\; S_t = s\ , A_t = \text{a}\right] \quad ,\text{for all } s \in S$$

我们称q_π为策略π的动作-值函数。

值函数v_π和q_π可以通过经验估计。例如，对于每个可达状态，如果 agent 遵循策略π并储存所遇状态下每个实际回报的平均值，那么这个平均值随着所遇状态数趋向无穷时会收敛到状态值，$V_\pi(s)$。如果每个状态下采取的每个动作都保留一个单独的平均值，那么这些平均值同样也会收敛于动作值$q_\pi(s,a)$。这种估计方法称为蒙克卡罗方法，因为它涉及了很多随机样本的实际回报的平均。当然，如果状态数过多，对每个装填单独保存平均值可能不太实际。相替代的，agent 会建立v_π和q_π作为参数化函数（相比状态函数参数较少），并且调整参数以便于更好的匹配观察到的回报。这也可以产生准确的估计，尽管这在很大程度上取决于参数化函数近似器的内在性质。

强化学习和动态规划都使用了值函数的一个基本属性，就是他们满足一定的递归递归，类似

于我们已经建立的回报。对于任意策略π和状态 s，状态 s 的值和可能的后继状态值之间存在以下一致性条件：

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

其中，动作 a 是从动作集 A(s)中得到，下一个状态 s'是从状态集合 S 中得到的（或者从情节式任务S^+中得到的），奖赏值 r 是从集合 R 中得到的。还要注意在最后一个方程中，我们如何将在' s 的所有值上与在 r 的所有值上的和，合并成一个值。通常用这种合并方式来简化公式。作为一个期望值怎样才能使最后的表达式更便于阅读，者实际上是三个变量 s',a r 的所有可能值得总和。

方程（3.14）是v_π的 Bellman 方程，它表达出当前状态值和后继状态值之间的关系。

最优策略和最优值函数

简而言之，解决一个强化学习任务以为着找到一个能在长期运行中获得更多奖赏的策略。对于有限马尔科夫策略，我们可以通过如下方法精确地定义一个最佳策略。值函数定义了对策略的部分排序。如果一个策略π的期望返回大于或等于策略π'的期望返回，则定义策略π优于或者等于策略π'。换句话说，只要对于所有的 s∈S，$v_\pi(s) \geq v_{\pi'}(s)$，那么必有π≥π'。至少总有一个策略比其他策略好或者等于其他策略，这个策略称为最优策略。尽管会存在多个最优策略，我们标记这些最优策略为π_*。他们或许有相同的状态-值函数，称为最优状态-值函数，表示为v_*。并且定义

$$V_*(s) = \max_\pi v_\pi(s) \quad , for \ \ all \ \ s \in S \tag{3.15}$$

最优策略同样也有最优动作-值函数，表示为q_*,定义如下：

$$q_*(s, a) = \max_\pi q_\pi(s, a) \quad , for \ \ all \ \ s \in S \ \ and \ \ a \in A(s) \tag{3.16}$$

对于状态-动作对，这个函数给出了遵循最佳策略且在状态 s 下采取动作 a 的期望返回。因此，我们可以根据v_*写出q_*如下：

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

因为v_π是一个策略的值函数，所以它必须满足 Bellman 方程（3.14 式）对状态值得自我一致性条件。但是，因为这是最优值函数，v_*的一致性条件可以不参照任何特殊策略而写成一个特定的形式。这是v_*的 Bellman 等式，或者称为 Bellman 最优等式。直观的说，Bellman 最优方程表达了在一个最优策略下的状态的值必须等于这个状态下采取最佳动作的期望返回这样一个事实：

$$
\begin{aligned}
v_*(s) &= \max_{a \epsilon A(s)} q_{\pi*}(s,a) \\
&= \max_a E_{\pi*}[G_t | S_t = s, A_t = a] \\
&= \max_a E_{\pi*}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] && (by (3.9)) \\
&= \max_a E_{\pi*}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] && (3.18) \\
&= \max_a \sum_{s',r} p(s', r | s, a)[r + \gamma v_*(s')] && (3.19)
\end{aligned}
$$

最后两个等式是v_*的 Bellman 方程的两种形式，q_*Bellman 方程如下：

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

图 3.5 的回溯图用图形的方式描述了v_*和q_*的 Bellman 最优方程中所考虑的未来状态和动作的跨度。这和v_π和q_π的回溯图一样的，不同之处在于在 agent 选择点处加了圆弧，表示我们选择这些动作的最大值，而不是给定策略的期望值。

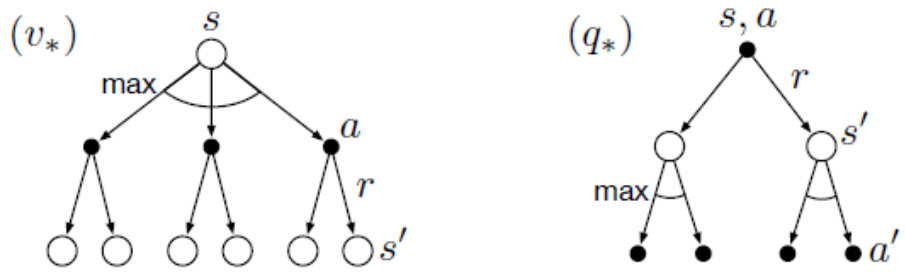


Figure 3.5: Backup diagrams for v_* and q_*

在有限 MDPs 中，v_π的 Bellman 最优方程有一个唯一的独立于策略的解。Bellman 最优方程实际上是一个方程组，在这组方程中，每个状态对应一个方程。所以如果有 n 个状态，则有 n 个方程和 n 个未知数。如果环境动态因素 p 是已知的，那么原则上可以用任意一种解非线性方程的方法来解v_*的方程组。同样也可以解出q_*的方程组。

一旦求出v_*，就可以相对简单的确定一个最优策略。对于每个状态 s，总会存在一个或多个动作使得在 Bellman 最优方程中获得值最大。任何将非零概率分配给这些动作的策略都是最优策略，可以把它认为是一个一步搜索（one-step search）。如果你有最优值函数v_*，一次一步搜索之后表现最好的动作会是最佳动作。宁一种说法是任何考虑最有评估函数v_*的贪心策略都是最佳策略。“贪婪”一词在计算机科学中被用来描述任何一种根据局部或立即因素来

进行选择的搜索或决策过程，而不考虑这样的选择会阻止未来获得更好选择的可能性。因此，它所描述的选择动作策略是基于短期结果。v_*的精妙之处在于如果用它去评价动作的短期结果（一步的结果），那么贪心策略在我们所感兴趣的长期角度来看确实是最优的，因为v_*已经把所有可能未来行为的奖赏序列都考虑到了。通过v_*，每个状态的最优长期期望回报转换成了一个局部、立即可获得的回报。因此，一步向前搜索能产生长期的最优动作。

使用q_*能使选择最优动作更简单。在q_*中，agent 甚至不需要去做一步向前搜索：对任何状态 s，可以简单地找到使$q_*(s,a)$最大化的动作。动作-值函数高效缓存所有一步向前搜索结果。它提供了最优期望长期返回作为一个为每个状态-动作对的局部、立即可用的值。因此，以用状态-动作对函数代替状态函数为代价，最优动作-值函数允许最优动作在不知道任何可能的后继状态和其值得情况下被选择，也就是说，可以不知道任何的环境动态因素信息。

显然解 Bellman 最优方程提供了一条寻找最优策略的路径，从而解决了强化学习问题。但是，这种解法并不直接有效，它类似于一种彻底搜索，展望所有的可能性，计算他们出现的概率和就期望奖赏而言的可取性。这种解法至少依赖在实践中很少正确的三个假设：(1) 我们准确知道环境动态因素；(2) 我们有充足的资源来完成计算求解；(3) 马尔可夫特性。对于我们感兴趣的这类任务，由于这些假设的各种组合相互冲突，一个人通常不能完全实现这个解决方案。例如，尽管在西洋双陆棋游戏中，假设（1）和假设（3）都成立，但假设（2）成为主要的障碍。那么这个游戏有10^{20}中状态，在今天最快的计算机上大约会花费几千年来解v_*的 Bellman 方程，对于q_*也是一样。因此在强化学习中通常不得不采取逼近的方法来解决这个问题。

最优和逼近

我们定义了最优值函数和最优策略。显然，一个学习最优策略的 agent 会表现的很好，但最优策略在实践中却很难找到，对于我们感兴趣的这类任务，要得到最优策略得付出昂贵的计算代价。我们在本书中描述了一个定义良好的最优性概念，并提供了一种理解各种学习算法的理论性质的方法，但它是一种理想的情况，agent 只能在不同程度上进行逼近。正如上述讨论，即使我们有完整准确的环境动态因素模型，通常也不可能通过简单求解 Bellman 最优方程来计算最优策略。

存储能力同样是一个重要的限制。建立近似的值函数、策略和模型通常需要较大的存储。小规模、有限状态集任务中，很容易使用每个状态（状态-动作对）的只有一个条目的数组或表来形成这些逼近，称为表格式情况，对应的方法称为表格式方法。但在许多实际例子中，往往有大量状态无法放在表格中。这种情况下方程必须使用某种更紧凑的参数化函数逼近。

强化学习问题框架迫使我们把注意力放在逼近上。但是，它也向我们呈现了一些实现有用逼近的独特条件。例如：在逼近最优行为时，agent 面对的许多状态有很低的概率使得他们在选择次有动作时对奖赏几乎不起任何作用。强化学习的在线特征使得逼近最优策略成为可能，即在平凡遇到的状态中花费更多的精力去学习制定好的决策。这是区分强化学习与其他逼近地解决马尔可夫决策过程的方法的关键特性。

总结

现在我们来总结一下强化学习问题的基本要素。强化学习是从交互中学习如何采取行动以实现目标。强化学习 agent 和其环境在一系列离散时间步中交互。它们之间相互作用的具体描述定义了一个特殊任务：动作是 agent 做出的选择，状态基于所做的选择，奖赏是做出选择的反馈。Agent 内部的所有东西都是完全已知的并且由 agent 控制的，所有外部是不完全可控的且或者完全未知。策略是由 agent 选择作为状态函数的操作的随机规则。Agent 的目标是最大化长时间的奖赏总值。

上述的强化学习设置是由定义良好的转换概率组成的，它构成了一个马尔可夫决策过程。有限 MDP 是有限状态、动作和奖赏集合的 MDP。很多当前的强化学习理论都局限于有限 MDPs，但其方法和思想运用广泛，

返回是未来奖赏的函数，agent 要寻找其最大化，根据任务特性和是否希望折扣延迟奖赏，它有几种不同定义。无折扣的公式适用于情节式任务，其中 agent-环境交互自然分解成情节。折扣公式适用于连续任务，其中交互不会自然分解成情节而是无限制的继续。我们试图定义两种任务的回报使得一组方程可以同时适用于情节式任务和连续任务两种情况。

策略值函数给每个状态或者状态-动作对赋以从这个状态或状态-动作对开始 agent 使用了该策略后的期望回报。最优值函数给每个状态或状态-动作对赋以通过任何任何策略得到的最大期望回报。对于给定 MDP，状态或状态-动作对的最优值函数值独一无二的，但可能有多个最优策略。任何考虑最优值函数的贪心策略一定是最优策略。Bellman 最优方程式最优值函数必须满足的一致性条件，并且原则上，他可以解出最优值函数，从而可以相对容易的确定一个最优策略。

一个讲话学习问题可以用很多不同方式提出，这取决于 agent 初始时能利用的知识水平的假设。在完备知识问题中，agent 有一个完整准确的环境动态因素模型。如果环境是 MDP，那么这样的模型组成了完整四参数动态函数 p(3.2)。在不完备知识问题中，是无法获取一个完整的且完美的环境模型。

即使 agent 有完整精确环境模型，它在每个时间步还是无法执行足够的计算来充分使用这个模型。存储空间是一个重要限制。我们需要存储空间来建立值函数、策略和模型。在多数实例中，状态数量太多无法存储在一张表中以至于需要做逼近。

一个明确的最优性概念组织了我们在本书中所表述的学习方法，并且提供了一种理解不同学习算法的理论属性的方法，但它只是一种理想情况，强化学习只能在不同程度上进行逼近。