Author：Liu Jian

Time：2021-08-19

Finite Markov Decision Processes

1MDP 是什么

MDPs are a mathematically idealized form of the reinforcement learning problem for which precise theoretical statements can be made (马尔可夫决策过程是针对强化学习问题的数学建模). The learner and decision maker is called **the agent**. The thing it interacts with, comprising everything outside the agent, is called **the environment**. MDPs 的运行过程如下图所示：

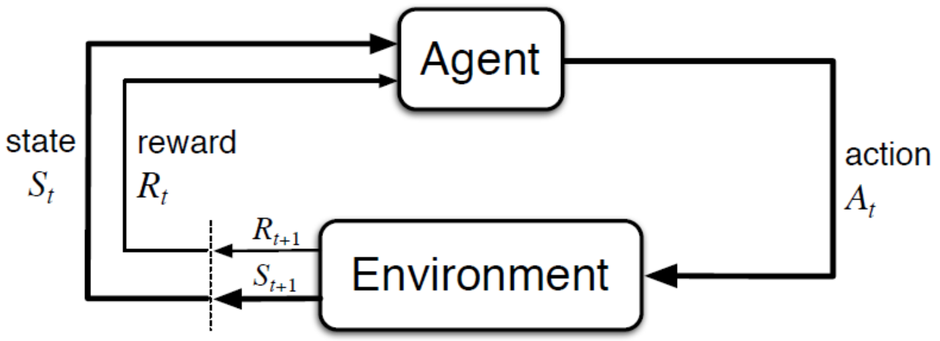


Figure 3.1: The agent–environment interaction in a Markov decision process.

The MDP thereby give rise to a sequence or trajectory that begins like this:

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

where **state** $S_t \in \mathcal{S}$, **action** $A_t \in \mathcal{A}(s)$, **reward** $R_t \in \mathcal{R} \subset \mathbb{R}$. In a **finite** MDP, the sets of states actions, and rewards ($\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$) all have a finite number of elements.

The dynamics function $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is an ordinary deterministic function of four arguments which defines the dynamics of the MDP:

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

按照书中的意思，p 是一个函数符号，而不是一般的概率符号，其中 "|" 借用自条件概率，以方便理解和使用。一般地，花体表示取值空间/集合，大写表示随机变量，小写表示具体数值。可以看到，我们假设 The state must include information about all aspects of the past agent–environment interaction that make a difference for the future. If it does, then the state is said to have the **Markov property**.

In a Markov decision process, the probabilities given by p completely characterize the environment's dynamics. From the four-argument dynamics function, p, one can compute anything else one might want to know about the environment, such as:

- the state-transition probabilities (which we denote, with a slight abuse of notation, as a three-argument function $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$) :

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

  注意，这里又定义了一个新的**转态转移函数**，为了简便起见，其函数符号依然采用 p，所以书中指明这里符号有点滥用 (a slight abuse of notation)，下同。感觉可以直接将 p 视为概率符号，这样即说得通也不存在符号滥用，不过还是尊重书中的做法吧。
- the expected rewards for state–action pairs $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$:

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

- the expected rewards for state–action–next-state triples $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$:

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

总结：

- **The MDP framework proposes that any problem of learning goal-directed behavior can be reduced to three signals passing back and forth between an agent and its environment**: one signal to represent the choices made by the agent (the actions), one signal to represent the basis on which the choices are made (the states), and one signal to define the agent's goal (the rewards). This framework may not be sufficient to represent all decision-learning problems usefully, but it has proved to be widely useful and applicable.
- Of course, the particular states and actions vary greatly from task to task, and how they are represented can strongly affect performance. In reinforcement learning, as in other kinds of learning, such representational choices are at present more art than science. In this book we offer some advice and examples regarding good ways of representing states and actions, but our primary focus is on general principles for learning how to behave once the representations have been selected.
- 建模时，agent 与 environment 的划分问题：In general, actions can be any decisions we want to learn how to make, and states can be anything we can know that might be useful in making them. The general rule we follow is that anything that cannot be changed arbitrarily by the agent is considered to be outside of it and thus part of its environment. The agent–environment boundary represents the limit of the agent's absolute control, not of its knowledge. The agent–environment boundary can be located at different places for different purposes.

> transition graph: a useful way of summarizing the dynamics of a finite MDP, 详见 P52 Example 3.3.

2 如何进行强化学习

第一小节介绍了用于对强化学习理论建模的有限马尔可夫模型，下面介绍强化学习的目标，即 the maximization of the expected value of the cumulative sum of a received scalar signal (called reward, 最大化期望累积奖赏). Informally, the agent's goal is to maximize the total amount of reward it receives. This means maximizing not immediate reward, but cumulative reward in the long run.

The use of a reward signal to formalize the idea of a goal is one of the most distinctive features of reinforcement learning. Although formulating goals in terms of reward signals might at first appear limiting, in practice it has proved to be flexible and widely applicable.

建模时如何设置 reward：The agent always learns to maximize its reward. If we want it to do something for us, we must provide rewards to it in such a way that in maximizing them the agent will also achieve our goals. It is thus critical that the rewards we set up truly indicate what we want accomplished.

In particular, the reward signal is not the place to impart to the agent prior knowledge about how to achieve what we want it to do (better places for imparting this kind of prior knowledge are the initial policy or initial value function). For example, a chess-playing agent should be rewarded only for actually winning, not for achieving subgoals such as taking its opponent's pieces or aining control of the center of the board. **If achieving these sorts of subgoals were rewarded, then the agent might find a way to achieve them without achieving the real goal**. For example, it might find a way to take the opponent's pieces even at the cost of losing the game. **The reward signal is your way of communicating to the agent what you want achieved, not how you want it achieved**.

2.1 优化的目标函数与学习对象

the agent's goal is to maximize the cumulative reward it receives in the long run, 也称 the expected return, 我们记 return 为 G_t. 根据任务是否会终止还是会一直进行下去，即是否存在 terminal state, 强学学习任务可分为 episodic tasks 和 continuing tasks：

- episodic tasks: In episodic tasks we sometimes need to distinguish the set of all nonterminal states, denoted $\mathcal{S}$, from the set of all states plus the terminal state, denoted $\mathcal{S}^+$. The time of termination, T, is a random variable that normally varies from episode to episode. 此时，G_t 通常采用如下定义：

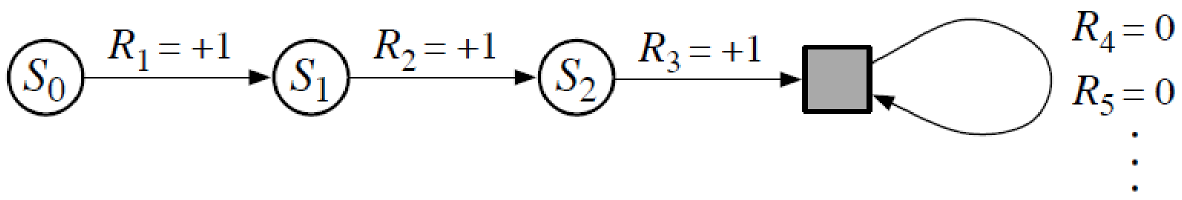
$$G_t \triangleq R_{t+1} + R_{t+2} + \cdots + R_T$$

- continuing tasks: the agent–environment interaction does not break naturally into identifiable episodes, but goes on continually without limit. 此时，return 采用和 episodic tasks 一样的定义会导致 return 趋于无穷。为此，我们引入 discounting，discounted return 定义如下：

$$G_t \triangleq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

  where γ is a parameter, $0 \leqslant \gamma \leqslant 1$, called the discount rate. If $\gamma < 1$, the infinite sum has a finite value as long as the reward sequence R_k is bounded.

可以看到，不同形式的任务，return 的形式不同。为此，It is therefore useful to establish one notation that enables us to talk precisely about both cases simultaneously. 首先，对于某个 episodic task, 不同的 episode 按理来说应该加以区分. However, it turns out that when we discuss episodic tasks we almost never have to distinguish between different episodes. We are almost always considering a particular episode, or stating something that is true for all episodes. Accordingly, in practice we almost always abuse notation slightly by dropping the explicit reference to episode number. That is, we write S_t to refer to $S_{t,i}$ (表示 the state representation at time t of episode i), and so on. 此外，我们可以认为 episode termination to be the entering of a special absorbing state that transitions only to itself and that generates only rewards of zero. For example, consider the state transition diagram:



由此，我们可以统一标记如下 (两种记法任意一种均可)：(1) Thus, we can define the return, in general, according to $G_t = \sum_{k=0}^{\infty}\gamma^k R_{t+k+1}$, using the convention of omitting episode numbers when they are not needed, and including the possibility that $\gamma=1$ if the sum remains defined (e.g., because all episodes terminate). (2) Alternatively, we can write $G_t = \sum_{k=t+1}^{T}\gamma^{k-t-1}R_k$ including the possibility that $T=\infty$ or $\gamma=1$ (but not both).

强化学习学习的是策略 (policy)，而策略的学习基于最大化价值函数 (value function，也就是 expected return)。 **value functions**—functions of states (or of state–action pairs) that estimate how good it is for the agent to be in a given state (or how good it is to perform a given action in a given state). Of course the rewards the agent can expect to receive in the future depend on what actions it will take. Accordingly, value functions are defined with respect to particular ways of acting, called **policies**. Formally, a policy is a mapping from states to probabilities of selecting each possible action. Reinforcement learning methods specify how the agent's policy is changed as a result of its experience.

the state-value function for policy π: The value function of a state s under a policy π, denoted $v_\pi(s)$, is the expected return when starting in s and following π thereafter:

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

for all $s\in\mathcal{S}$. 注意，和传统期望记号 $\mathbb{E}_x$ 表示对下标 x 服从的概率分布求期望不同，$\mathbb{E}_\pi$ 中的下标 π 虽然也可以表示一个概率分布，但这里是表示采用策略 π (策略 π 会影响所有的时间步) 时能获得的 expected return，并不表示仅仅只基于概率分布 π 求一次期望，即 $v_\pi(s)\neq\sum_a \pi(a|s)G_t$。

the action-value function for policy π: Similarly, we define the value of taking action a in state s under a policy π, denoted $q_\pi(s,a)$, as the expected return starting from s, taking the action a, and thereafter following policy π:

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

价值函数的评估方法：

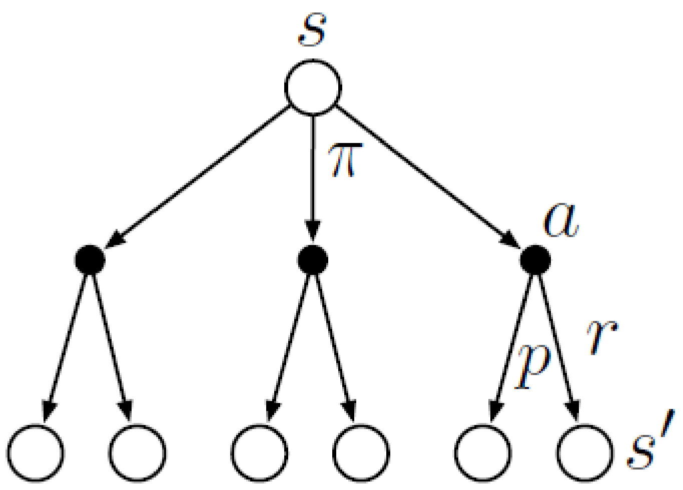
1. 蒙特卡洛法 (We call estimation methods of this kind Monte Carlo methods because they involve averaging over many random samples of actual returns. These kinds of methods are presented in Chapter 5)：if an agent follows policy π and maintains an average, for each state encountered, of the actual returns that have followed that state, then the average will converge to the state's value, $v_\pi(s)$, as the number of times that state is encountered approaches infinity. If separate averages are kept for each action taken in each state, then these averages will similarly converge to the action values, $q_\pi(s,a)$.
2. 参数化近似法：Of course, if there are very many states, then it may not be practical to keep separate averages for each state individually. Instead, the agent would have to maintain v_π and q_π as parameterized functions (with fewer parameters than states) and adjust the parameters to better match the observed returns. This can also produce accurate estimates, although much depends on the nature of the parameterized function approximator. These possibilities are discussed in Part II of the book.

价值函数存在递归关系 (recursive relationships)：

1. Bellman equation for v_π:

$$\begin{aligned} v_\pi(s) &\triangleq \mathbb{E}_\pi[G_t|S_t=s] \\ &= \mathbb{E}_\pi[R_{t+1}+\gamma G_{t+1}|S_t=s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s',r|s,a) \Big[r+\gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1}=s']\Big] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \Big[r+\gamma v_\pi(s')\Big], \quad \text{for all } s \in \mathcal{S} \end{aligned}$$

Backup diagram for v_π (Each open circle represents a state and each solid circle represents a state–action pair，**使用 backup diagram 能很方便地帮助我们进行公式推导**):



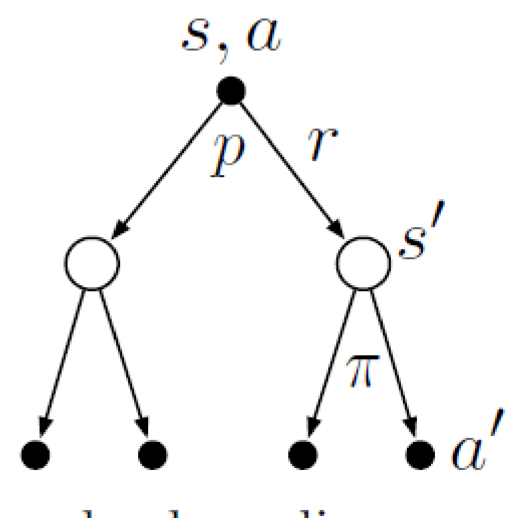
Backup diagram for v_π

The value function v_π is the unqiue solution to its Bellman equation. Bellman equation forms the basis of a number of ways to compute, approximate, and learn v_π.

2. 类似地，还有 Bellman equation for action values q_π:

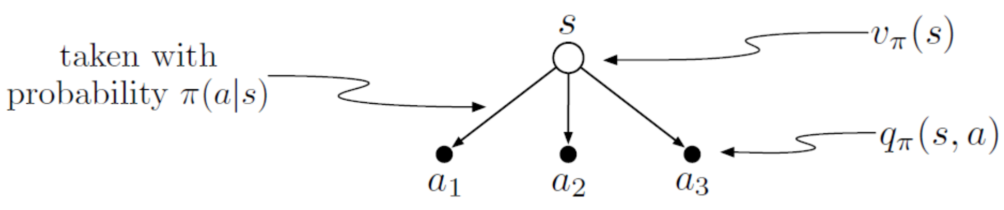
$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi[G_t|S_t=s,A_t=a] \\ &= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\middle|S_t=s,A_t=a\right] \\ &= \sum_{a'} \pi(a'|s') \sum_{s'} \sum_r p(s',r|s,a) \Big[r+\gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1}=s',A_{t+1}=a']\Big] \\ &= \sum_{a'} \pi(a'|s') \sum_{s',r} p(s',r|s,a) \Big[r+\gamma q(s',a')\Big] \end{aligned}$$

backup diagram:



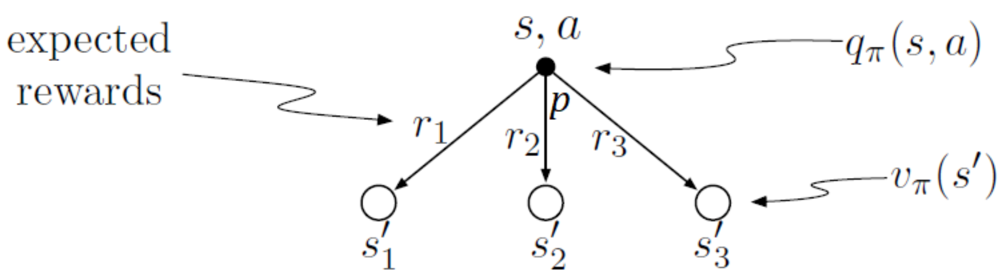
q_π backup diagram

3. The value of a state depends on the values of the actions:



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

The value of an action, $q_\pi(s,a)$, depends on the expected next reward and the expected sum of the remaining rewards:



$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)(r + \gamma v_\pi(s'))$$

综上，**强化学习最大化的目标函数是 expected return，也就是价值函数；而价值函数依赖于 agent 的策略 π 和 environment 的动态特性 p，且满足 Bellman equation；强化学习的对象为使价值函数最大的策略**.

2.2 最优价值函数与最优策略

由上一小节可知，强化学习的对象是策略 π，而最优策略的 expected return，也就是价值函数是最大的：A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states. In other words, $\pi \geqslant \pi'$ if and only if $v_\pi(s) \geqslant v_{\pi'}(s)$ for all $s \in \mathcal{S}$. There is always at least one policy that is better than or equal to all other policies. This is an optimal policy. Although there may be more than one, we denote all the optimal policies by π_*. 最优策略的价值函数记为 v_* (the optimal state-value function) 和 q_* (the optimal a-action-value function)

$$
\begin{aligned}
v_*(s) &= \max_\pi v_\pi(s) \qquad\qquad\qquad , \text{for all } s \in \mathcal{S} \\
q_*(s,a) &= \max_\pi v_\pi(s) \quad , \text{for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}(s)
\end{aligned}
$$

对于所在的环境，任何策略 π 的价值函数 v_π, q_π 总满足 Bellman equation，对于最优策略 π_* 显然也是如此：

$$
\begin{aligned}
v_*(s) &= \sum_a \pi_*(a|s) \sum_{s',r} p(s',r|s,a) \Big[r + \gamma v_*(s')\Big] \\
q_*(s,a) &= \sum_{a'} \pi_*(a'|s') \sum_{s',r} p(s',r|s,a) \Big[r + \gamma q_*(s',a')\Big]
\end{aligned}
$$

但我们可以进一步将上述等式写成不依赖最优策略 π_* 的形式，即 Bellman optimality equation. 注意到，the value of a state under an optimal policy must equal the expected return for the best action from that state:

$$
\begin{aligned}
v_*(s) \quad &= \max_{a\in\mathcal{A}(s)} q_*(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}[G_t|S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a] \\
&= \max_a \textstyle\sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')]
\end{aligned}
$$

类似地，the Bellman optimality equation for q_* is:

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

上述 Bellman optimality equations 对应的 backup diagrams 如下：

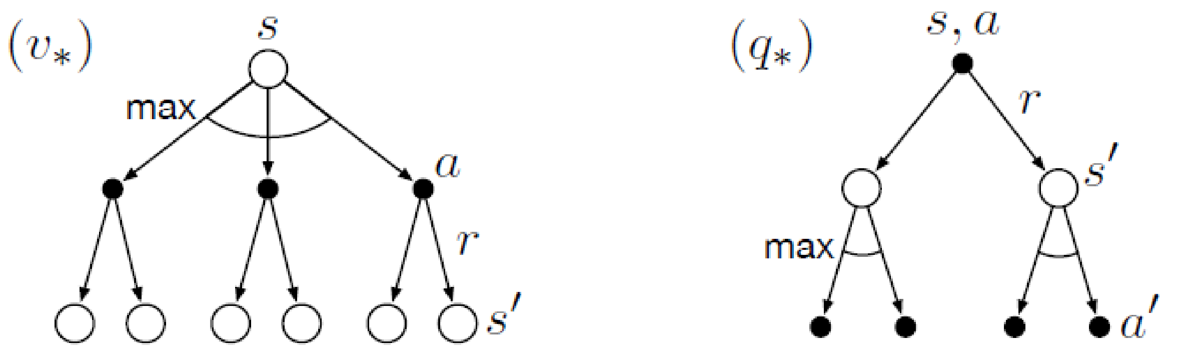


Figure 3.4: Backup diagrams for v_* and q_*

For finite MDPs, the Bellman optimality equation for v_* has a unique solution，也就是说，满足上述方程的价值函数就是最优价值函数. The Bellman optimality equation is actually a system of equations, one for each state, so if there are n states, then there are n equations in n unknowns. 当环境的动态特性 p 已知时，原则上 one can solve this system of equations for v_* using any one of a variety of methods for solving systems of nonlinear equations. One can solve a related set of equations for q_*.

下面讨论当最优价值函数已知时，如何求最优策略 π_*：

1. 基于价值函数 v_*：any policy that is **greedy** with respect to the optimal evaluation function v_* is an optimal policy, 具体操作而言，For each state s, there will be one or more actions at which the maximum is obtained in the Bellman optimality equation. Any policy that assigns nonzero probability **only** to these actions is an optimal policy. You can think of this as a one-

step search.

  - 可以看到, By means of v_* (already takes into account the reward consequences of all possible future behavior), the optimal expected long-term return is turned into a quantity that is locally and immediately available for each state. Hence, a one-step-ahead search yields the long-term optimal actions.

2. 基于价值函数 q_* 则更简单：With q_*, the agent does not even have to do a one-step-ahead search: for any state s, it can simply find any action that maximizes $q_*(s,a)$. The action-value function effectively caches the results of all one-step-ahead searches.

  - 可以看到，q_* provides the optimal expected long-term return as a value that is locally and immediately available for each state–action pair.

2.3 贝尔曼最优方程的近似求解

由上一小节可知，我们可以根据最优价值函数求得最优策略，而最优价值函数是 Bellman optimality equation 的唯一解得到。因此 Explicitly solving the Bellman optimality equation provides one route to finding an optimal policy, and thus to solving the reinforcement learning problem. However, this solution is rarely directly useful. 原因是 This solution relies on at least three assumptions that are rarely true in practice: (1) the dynamics of the environment are accurately known, (2) computational resources are sufficient to complete the calculation, (3) the states have the Markov property. 我们感兴趣的问题通常不会满足上述三个假设，even if we have a complete and accurate model of the environment's dynamics, it is usually not possible to simply compute an optimal policy by solving the Bellman optimality equation. A critical aspect of the problem facing the agent is always the computational power available to it, in particular, the amount of computation it can perform in a single time step. The memory available is also an important constraint.

因此 In reinforcement learning one typically has to settle for approximate solutions. Many different decision-making methods can be viewed as ways of approximately solving the Bellman optimality equation. **Reinforcement learning adds to MDPs a focus on approximation and incomplete information for realistically large problems. There is one key property that distinguishes reinforcement learning from other approaches to approximately solving MDPs**: The online nature of reinforcement learning makes it possible to approximate optimal policies in ways that put more effort into learning to make good decisions for frequently encountered states, at the expense of less effort for infrequently encountered states.

3 总结

Reinforcement learning is about learning from interaction how to behave in order to achieve a goal. The reinforcement learning agent and its environment interact over a sequence of discrete time steps. The **actions** are the choices made by the agent; the **states** are the basis for making the choices; and the **rewards** are the basis for evaluating the choices. Everything inside the agent is known and controllable. Its environment, on the other hand, is incompletely controllable and may or may not be completely known. A **policy** is a stochastic rule by which the agent selects actions as a function of states. The agent's objective is to maximize the amount of reward it receives over time.

When the reinforcement learning setup described above is formulated with well defined transition probabilities it constitutes a **Markov decision process (MDP)**. A **finite MDP** is an MDP with finite state, action, and (as we formulate it here) reward sets.

The **return** is the function of future rewards that the agent seeks to maximize (in expected value). The undiscounted formulation is appropriate for **episodic tasks**, in which the agent–environment interaction breaks naturally into **episodes**; the discounted formulation is appropriate for **continuing tasks**, in which the interaction does not naturally break into episodes but continues without limit. We try to define the returns for the two kinds of tasks such that one set of equations can apply to both the episodic and continuing cases.

A policy's **value functions** (v_π and q_π) assign to each state, or state–action pair, the expected return from that state, or state–action pair, given that the agent uses the policy. The **optimal value functions** (v_* and q_*) assign to each state, or state–action pair, the largest expected return achievable by any policy. A policy whose value functions are optimal is an **optimal policy**.

Whereas the optimal value functions for states and state–action pairs are unique for a given MDP, there can be many optimal policies. Any policy that is greedy with respect to the optimal value functions must be an optimal policy. The **Bellman optimality equations** are special consistency conditions that the optimal value functions must satisfy and that can, in principle, be solved for the optimal value functions, from which an optimal policy can be determined with relative ease.

> The counterpart of the Bellman optimality equation for continuous time and state problems is known
> as the Hamilton–Jacobi–Bellman equation (or often just the Hamilton–Jacobi equation).

A reinforcement learning problem can be posed in a variety of different ways depending on assumptions about the level of knowledge initially available to the agent. In problems of **complete knowledge**, the agent has a complete and accurate model of the environment's dynamics. In problems of **incomplete knowledge**, a complete and perfect model of the environment is not available. **Even if the agent had a complete and accurate environment model, the agent would typically be unable to fully use it because of limitations on its memory and computation per time step. In most cases of practical interest there are far more states than could possibly be entries in a table, and approximations must be made**.
