# 17 Reinforcement Learning

This chapter presents an introduction to reinforcement learning, a rich area of machine learning with connections to control theory, optimization, and cognitive sciences. Reinforcement learning is the study of planning and learning in a scenario where a learner actively interacts with the environment to achieve a certain goal. This active interaction justifies the terminology of *agent* used to refer to the learner. The achievement of the agent's goal is typically measured by the reward it receives from the environment and which it seeks to maximize.

We first introduce the general scenario of reinforcement learning and then introduce the model of Markov decision processes (MDPs), which is widely adopted in this area, as well as essential concepts such as that of *policy* or *policy value* related to this model. The rest of the chapter presents several algorithms for the *planning* problem, which corresponds to the case where the environment model is known to the agent, and then a series of *learning* algorithms for the more general case of an unknown model.

## 17.1 Learning scenario

The general scenario of reinforcement learning is illustrated by figure 17.1. Unlike the supervised learning scenario considered in previous chapters, here, the learner does not passively receive a labeled data set. Instead, it collects information through a course of *actions* by interacting with the *environment*. In response to an action, the learner or *agent*, receives two types of information: its current *state* in the environment, and a real-valued *reward*, which is specific to the task and its corresponding goal.

The objective of the agent is to maximize its reward and thus to determine the best course of actions, or *policy*, to achieve that objective. However, the information he receives from the environment is only the immediate reward related to the action just taken. No future or long-term reward feedback is provided by the environment. An important aspect of reinforcement learning is to consider delayed rewards or

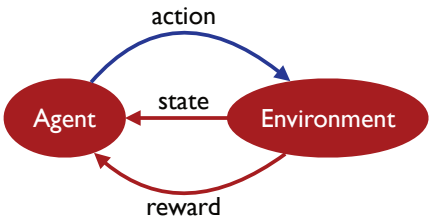


**Figure 17.1**
Representation of the general scenario of reinforcement learning.

penalties. The agent is faced with the dilemma between exploring unknown states and actions to gain more information about the environment and the rewards, and exploiting the information already collected to optimize its reward. This is known as the *exploration versus exploitation trade-off* inherent to reinforcement learning.

Note that there are several differences between the learning scenario of reinforcement learning and that of supervised learning examined in most of the previous chapters. Unlike supervised learning, in reinforcement learning there is no fixed distribution according to which instances are drawn; it is the choice of a policy that defines the distribution over observations. In fact, slight changes to the policy may have dramatic effects on the rewards received. Furthermore, in general, the environment may not be fixed and could vary as a result of the actions selected by the agent. This may be a more realistic model for some learning problems than the standard supervised learning. Finally, note that, unlike supervised learning, in reinforcement learning, training and testing phases are intermixed.

Two main settings can be distinguished here: the one where the environment model is known to the agent, in which case its objective of maximizing the reward received is reduced to a *planning problem*; and the one where the environment model is unknown, in which case the agent faces a *learning problem*. In the latter setting, the agent must learn from the state and reward information gathered to both gain information about the environment and determine the best action policy. This chapter presents algorithmic solutions for both of these settings.

## 17.2 Markov decision process model

We first introduce the model of Markov decision processes (MDPs), a model of the environment and interactions with the environment widely adopted in reinforcement learning. An MDP is a Markovian process defined as follows.

**Definition 17.1 (MDPs)** *A Markov decision process (MDP) is defined by:*

- *a set of* states $S$*, possibly infinite.*

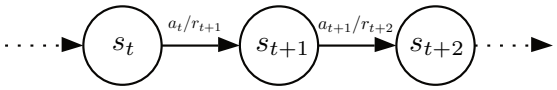


**Figure 17.2**
Illustration of the states and transitions of an MDP at different times.

- *a* start state *or* initial state $s_0 \in S$.
- *a set of* actions $A$, *possibly infinite.*
- *a* transition probability $\mathbb{P}[s'|s,a]$*: distribution over destination states* $s' = \delta(s,a)$.
- *a* reward probability $\mathbb{P}[r'|s,a]$*: distribution over rewards returned* $r' = r(s,a)$.

The model is Markovian because the transition and reward probabilities depend only on the current state $s$ and not the entire history of states and actions taken. This definition of MDP can be further generalized to the case of non-discrete state and action sets.

In a discrete-time model, actions are taken at a set of *decision epochs* $\{0, \ldots, T\}$, and this is the model we will adopt in what follows. This model can also be straightforwardly generalized to a continuous-time one where actions are taken at arbitrary points in time.

When $T$ is finite, the MDP is said to have a *finite horizon*. Independently of the finiteness of the time horizon, an MDP is said to be *finite* when both $S$ and $A$ are finite sets. Here, we are considering the general case where the reward $r(s,a)$ at state $s$ when taking action $a$ is a random variable. However, in many cases, the reward is assumed to be a deterministic function the state and action pair $(s,a)$.

Figure 17.2 illustrates the model corresponding to an MDP. At time $t \in \{0, \ldots, T\}$ the state observed by the agent is $s_t$ and it takes action $a_t \in A$. The state reached is $s_{t+1}$ (with probability $\mathbb{P}[s_{t+1}|s_t, a_t]$) and the reward received $r_{t+1} \in \mathbb{R}$ (with probability $\mathbb{P}[r_{t+1}|s_t, a_t]$).

Many real-world tasks can be represented by MDPs. Figure 17.3 gives the example of a simple MDP for a robot picking up balls on a tennis court.

## 17.3 Policy

The main problem for an agent in an MDP environment is to determine the action to take at each state, that is, an action *policy*.

### 17.3.1 Definition

**Definition 17.2 (Policy)** *A* policy *is a mapping* $\pi\colon S \to \Delta(A)$*, where* $\Delta(A)$ *is the set of probability distributions over* $A$*. A policy* $\pi$ *is* deterministic *if for any* $s$*, there exists a unique* $a \in A$ *such that* $\pi(s)(a) = 1$*. In that case, we can identify* $\pi$ *with a mapping from* $S$ *to* $A$ *and use* $\pi(s)$ *to denote that action.*

More precisely, this is the definition of a *stationary policy* since the choice of the distribution of actions does not depend on time. More generally, we could define a *non-stationary policy* as a sequence of mappings $\pi_t \colon S \to \Delta(A)$ indexed by $t$. In particular, in the finite horizon case, a non-stationary policy is typically necessary for optimizing rewards.

The agent's objective is to find a policy that maximizes its expected (reward) *return*. The return it receives following a deterministic policy $\pi$ along a specific sequence of states $s_0, \ldots, s_T$ is defined as follows:

- for a finite horizon ($T < \infty$): $\sum_{t=0}^{T} r\big(s_t, \pi(s_t)\big)$.
- for an infinite horizon ($T = \infty$): $\sum_{t=0}^{+\infty} \gamma^t r\big(s_t, \pi(s_t)\big)$, where $\gamma \in [0, 1)$ is a constant factor less than one used to discount future rewards.

Note that the return is a single scalar summarizing a possibly infinite sequence of immediate rewards. In the discounted case, early rewards are viewed as more valuable than later ones.

### 17.3.2 Policy value

This leads to the following definition of the value of a policy at each state.

**Definition 17.3 (Policy value)** *The* value $V_\pi(s)$ *of a policy* $\pi$ *at state* $s \in S$ *is defined as the expected reward returned when starting at* $s$ *and following policy* $\pi$*:*

- *finite horizon:* $V_\pi(s) = \mathbb{E}_{a_t \sim \pi(s_t)} \left[ \sum_{t=0}^{T} r\big(s_t, a_t\big) \;\middle|\; s_0 = s \right]$;
- *infinite discounted horizon:* $V_\pi(s) = \mathbb{E}_{a_t \sim \pi(s_t)} \left[ \sum_{t=0}^{+\infty} \gamma^t r\big(s_t, a_t\big) \;\middle|\; s_0 = s \right]$,

*where the expectations are over the random selection of an action* $a_t$ *according to the distribution* $\pi(s_t)$*, which is explicitly indicated, and over the random states* $s_t$ *reached and the reward values* $r\big(s_t, a_t\big)$.[22] *An infinite undiscounted horizon is also often considered based on the limit of the average reward, when it exists.*

### 17.3.3 Optimal policies

Starting from a state $s \in S$, to maximize its reward, an agent naturally seeks a policy $\pi$ with the largest value $V_\pi(s)$. In this section, we will show that, remarkably, for any finite MDP in the infinite horizon setting, there exists a policy that is *optimal* for *any* start state, that is one with the following definition.

**Definition 17.4 (Optimal policy)** *A policy* $\pi^*$ *is* optimal *if its value is maximal for every state* $s \in S$*, that is, for any policy* $\pi$ *and any state* $s \in S$*,* $V_{\pi^*}(s) \geq V_\pi(s)$.

[22] More generally, in all that follows, the randomization with respect to the reward function and the next state will not be explicitly indicated to simplify the notation.

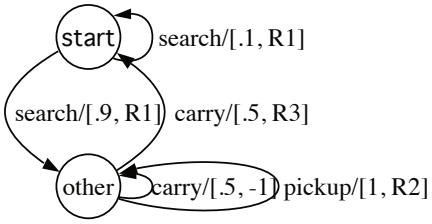


**Figure 17.3**
Example of a simple MDP for a robot picking up balls on a tennis court. The set of actions is $A = \{search, carry, pickup\}$ and the set of states reduced to $S = \{start, other\}$. Each transition is labeled with the action followed by the probability of the transition probability and the reward received after taking that action. $R_1$, $R_2$, and $R_3$ are real numbers indicating the reward associated to each transition (case of deterministic reward).

Moreover, we will show that for any MDP there exists a deterministic optimal policy. To do so, it is convenient to introduce the notion of *state-action value function*.

**Definition 17.5 (State-action value function)** *The* state-action value function $Q$ *associated to a policy* $\pi$ *is defined for all* $(s, a) \in S \times A$ *as the expected return for taking action* $a \in A$ *at state* $s \in S$ *and then following policy* $\pi$*:*

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}[r(s,a)] + \mathop{\mathbb{E}}_{a_t \sim \pi(s_t)} \left[ \sum_{t=1}^{+\infty} \gamma^t r\big(s_t, a_t\big) \;\middle|\; s_0 = s, a_0 = a \right] \\ &= \mathbb{E}\left[ r(s,a) + \gamma V_\pi(s_1) \;\middle|\; s_0 = s, a_0 = a \right]. \end{aligned} \tag{17.1}$$

Observe that $\mathbb{E}_{a \sim \pi(s)} \big[ Q_\pi(s,a) \big] = V_\pi(s)$ (see also proposition 17.9)

**Theorem 17.6 (Policy improvement theorem)** *For any two policies* $\pi$ *and* $\pi'$ *the following holds:*

$$\Big( \forall s \in S, \mathop{\mathbb{E}}_{a \sim \pi'(s)} \big[ Q_\pi(s,a) \big] \geq \mathop{\mathbb{E}}_{a \sim \pi(s)} \big[ Q_\pi(s,a) \big] \Big) \Rightarrow \Big( \forall s \in S, V_{\pi'}(s) \geq V_\pi(s) \Big).$$

*Furthermore, a strict inequality for at least one state* $s$ *in the left-hand side implies a strict inequality for at least one* $s$ *in the right-hand side.*

Proof: Assume that $\pi$ and $\pi'$ verify the left-hand side. For any $s \in S$, we have

$$\begin{aligned}
V_\pi(s) &= \mathop{\mathbb{E}}_{a\sim\pi(s)} \big[Q_\pi(s,a)\big] \\
&\leq \mathop{\mathbb{E}}_{a\sim\pi'(s)} \big[Q_\pi(s,a)\big] \\
&= \mathop{\mathbb{E}}_{a\sim\pi'(s)} \Big[r(s,a) + \gamma V_\pi(s_1) \;\Big|\; s_0 = s\Big] \\
&= \mathop{\mathbb{E}}_{a\sim\pi'(s)} \Big[r(s,a) + \gamma \mathop{\mathbb{E}}_{a_1\sim\pi(s_1)} \big[Q_\pi(s_1,a_1)\big] \;\Big|\; s_0 = s\Big] \\
&\leq \mathop{\mathbb{E}}_{a\sim\pi'(s)} \Big[r(s,a) + \gamma \mathop{\mathbb{E}}_{a_1\sim\pi'(s_1)} \big[Q_\pi(s_1,a_1)\big] \;\Big|\; s_0 = s\Big] \\
&= \mathop{\mathbb{E}}_{\substack{a\sim\pi'(s)\\ a_1\sim\pi'(s_1)}} \Big[r(s,a) + \gamma r(s_1,a_1) + \gamma^2 V_\pi(s_2) \;\Big|\; s_0 = s\Big].
\end{aligned}$$

Proceeding in this way shows that for any $T \geq 1$:

$$V_\pi(s) \leq \mathop{\mathbb{E}}_{a_t\sim\pi'(s_t)} \left[\sum_{t=0}^{T} \gamma^t \,\mathbb{E}[r(s_t,a_t)] + \gamma^{T+1} V_\pi(s_{T+1}) \;\middle|\; s_0 = s\right].$$

Since $V_\pi(s_{T+1})$ is bounded, taking the limit $T \to +\infty$ gives

$$V_\pi(s) \leq \mathop{\mathbb{E}}_{a_t\sim\pi'(s_t)} \left[\sum_{t=0}^{+\infty} \gamma^t \,\mathbb{E}[r(s_t,a_t)] \;\middle|\; s_0 = s\right] = V_{\pi'}(s).$$

Finally, any strict inequality in the left-hand side property results in a strict inequality in the chain of inequalities above. □

**Theorem 17.7 (Bellman's optimality condition)** *A policy $\pi$ is optimal iff for any pair $(s,a) \in S \times A$ with $\pi(s)(a) > 0$ the following holds:*

$$a \in \operatorname*{argmax}_{a' \in A} Q_\pi(s,a'). \tag{17.2}$$

Proof: By Theorem 17.6, if the condition (17.2) does not hold for some $(s,a)$ with $\pi(s)(a) > 0$, then the policy $\pi$ is not optimal. This is because $\pi$ can then be improved by defining $\pi'$ such that $\pi'(s') = \pi(s)$ for $s' \neq s$ and $\pi'(s)$ concentrated on any element of $\operatorname{argmax}_{a'\in A} Q_\pi(s,a')$. $\pi'$ verifies $\mathbb{E}_{a\sim\pi'(s)}\big[Q_\pi(s',a)\big] = \mathbb{E}_{a\sim\pi(s)}\big[Q_\pi(s',a)\big]$ for $s' \neq s$ and $\mathbb{E}_{a\sim\pi'(s)}\big[Q_\pi(s,a)\big] > \mathbb{E}_{a\sim\pi(s)}\big[Q_\pi(s,a)\big]$. Thus, by Theorem 17.6, $V_{\pi'}(s) > V_\pi(s)$ for at least one $s$ and $\pi$ is not optimal.

Conversely, let $\pi'$ be a non-optimal policy. Then there exists a policy $\pi$ and at least one state $s$ for which $V_{\pi'}(s) < V_\pi(s)$. By Theorem 17.6, this implies that there exists some state $s \in S$ with $\mathbb{E}_{a\sim\pi'(s)}\big[Q_\pi(s,a)\big] < \mathbb{E}_{a\sim\pi(s)}\big[Q_\pi(s,a)\big]$. Thus, $\pi'$ cannot satisfy the condition (17.2). □

**Theorem 17.8 (Existence of an optimal deterministic policy)** *Any finite MDP admits an optimal deterministic policy.*

Proof: Let $\pi^*$ be a deterministic policy maximizing $\sum_{s\in S} V_\pi(s)$. $\pi^*$ exists since there are only finitely many deterministic policies. If $\pi^*$ were not optimal, by Theorem 17.7, there would exist a state $s$ with $\pi(s) \notin \operatorname{argmax}_{a'\in A} Q_\pi(s,a')$. By theorem 17.6, $\pi^*$ could then be improved by choosing a policy $\pi$ with $\pi(s) \in \operatorname{argmax}_{a'\in A} Q_\pi(s,a')$ and $\pi$ coinciding with $\pi^*$ for all other states. But then $\pi$ would verify $V_{\pi*}(s) \leq V_\pi(s)$ with a strict inequality at least for one state. This would contradict the fact that $\pi^*$ maximizes $\sum_{s\in S} V_\pi(s)$. □

In view of the existence of a deterministic optimal policy, in what follows, to simplify the discussion, we will consider only deterministic policies. Let $\pi^*$ denote a (deterministic) optimal policy, and let $Q^*$ and $V^*$ denote its corresponding state-action value function and value function. By Theorem 17.7, we can write

$$\forall s \in S,\ \pi^*(s) = \operatorname*{argmax}_{a\in A} Q^*(s,a). \tag{17.3}$$

Thus, the knowledge of the state-action value function $Q^*$ is sufficient for the agent to determine the optimal policy, without any direct knowledge of the reward or transition probabilities. Replacing $Q^*$ by its definition gives the following system of equations for the optimal policy values $V^*(s) = Q^*(s,\pi^*(s))$:

$$\forall s \in S,\ V^*(s) = \max_{a\in A} \Big\{ \mathbb{E}[r(s,a)] + \gamma \sum_{s'\in S} \mathbb{P}[s'|s,a] V^*(s') \Big\}, \tag{17.4}$$

also known as *Bellman equations*. Note that this system of equations is not linear due to the presence of the max operator.

### 17.3.4 Policy evaluation

The value of a policy at state $s$ can be expressed in terms of its values at other states, forming a system of linear equations.

**Proposition 17.9 (***Bellman equations***)** *The values $V_\pi(s)$ of policy $\pi$ at states $s \in S$ for an infinite horizon MDP obey the following system of linear equations:*

$$\forall s \in S,\ V_\pi(s) = \mathop{\mathbb{E}}_{a_1\sim\pi(s)}[r(s,a_1)] + \gamma \sum_{s'} \mathbb{P}[s'|s,\pi(s)] V_\pi(s'). \tag{17.5}$$

Proof: We can decompose the expression of the policy value as a sum of the first term and the rest of the terms, which admit $\gamma$ as a multiplier:

$$\begin{aligned} V_\pi(s) &= \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma^t r\big(s_t, \pi(s_t)\big) \;\middle|\; s_0 = s\right]. \\ &= \mathbb{E}[r(s,\pi(s))] + \gamma\, \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma^t r\big(s_{t+1}, \pi(s_{t+1})\big) \;\middle|\; s_0 = s\right] \\ &= \mathbb{E}[r(s,\pi(s))] + \gamma\, \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma^t r\big(s_{t+1}, \pi(s_{t+1})\big) \;\middle|\; s_1 = \delta(s,\pi(s))\right] \\ &= \mathbb{E}[r(s,\pi(s)] + \gamma\, \mathbb{E}[V_\pi(\delta(s,\pi(s)))]. \end{aligned}$$

This completes the proof. □

This a linear system of equations, also known as Bellman equations, that is distinct from the non-linear system (17.4). The system can be rewritten as

$$\mathbf{V} = \mathbf{R} + \gamma \mathbf{P}\mathbf{V}, \tag{17.6}$$

using the following notation: $\mathbf{P}$ denotes the transition probability matrix defined by $\mathbf{P}_{s,s'} = \mathbb{P}[s'|s,\pi(s)]$ for all $s, s' \in S$; $\mathbf{V}$ is the value column matrix whose $s$th component is $\mathbf{V}_s = V_\pi(s)$; and $\mathbf{R}$ the reward column matrix whose $s$th component is $\mathbf{R}_s = \mathbb{E}[r(s,\pi(s)]$. $\mathbf{V}$ is typically the unknown variable in the Bellman equations and is determined by solving for it.

The following theorem shows that, for a finite MDP, this system of linear equations admits a unique solution.

**Theorem 17.10** *For a finite MDP, Bellman's equations admit a unique solution given by*

$$\mathbf{V}_0 = (\mathbf{I} - \gamma \mathbf{P})^{-1}\mathbf{R}. \tag{17.7}$$

Proof: The Bellman equations (17.6) can be equivalently written as

$$(\mathbf{I} - \gamma \mathbf{P})\mathbf{V} = \mathbf{R}.$$

Thus, to prove the theorem it suffices to show that $(\mathbf{I} - \gamma\mathbf{P})$ is invertible. To do so, note that the infinity of $\mathbf{P}$ can be computed using its stochasticity properties:

$$\|\mathbf{P}\|_\infty = \max_s \sum_{s'} |\mathbf{P}_{ss'}| = \max_s \sum_{s'} \mathbb{P}[s'|s,\pi(s)] = 1.$$

This implies that $\|\gamma\mathbf{P}\|_\infty = \gamma < 1$. The eigenvalues of $\gamma\mathbf{P}$ are thus all less than one, and $(\mathbf{I} - \gamma\mathbf{P})$ is invertible. □

Thus, for a finite MDP, when the transition probability matrix $\mathbf{P}$ and the reward expectations $\mathbf{R}$ are known, the value of policy $\pi$ at all states can be determined by inverting a matrix.

## 17.4 Planning algorithms

In this section, we assume that the environment model is known. That is, the transition probability $\mathbb{P}[s'|s,a]$ and the expected reward $\mathbb{E}[r(s,a)]$ for all $s, s' \in S$ and $a \in A$ are assumed to be given. The problem of finding the optimal policy then does not require learning the parameters of the environment model or estimating other quantities helpful in determining the best course of actions, it is purely a *planning* problem.

This section discusses three algorithms for this planning problem: the value iteration algorithm, the policy iteration algorithm, and a linear programming formulation of the problem.

### 17.4.1 Value iteration

The *value iteration algorithm* seeks to determine the optimal policy values $V^*(s)$ at each state $s \in S$, and thereby the optimal policy. The algorithm is based on the Bellman equations (17.4). As already indicated, these equations do not form a system of linear equations and require a different technique to determine the solution. The main idea behind the design of the algorithm is to use an iterative method to solve them: the new values of $V(s)$ are determined using the Bellman equations and the current values. This process is repeated until a convergence condition is met.

For a vector $\mathbf{V}$ in $\mathbb{R}^{|S|}$, we denote by $V(s)$ its $s$th coordinate, for any $s \in S$. Let $\mathbf{\Phi}\colon \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ be the mapping defined based on Bellman's equations (17.4):

$$\forall s \in S, [\mathbf{\Phi}(\mathbf{V})](s) = \max_{a \in A} \Big\{ \mathbb{E}[r(s,a)] + \gamma \sum_{s' \in S} \mathbb{P}[s'|s,a] V(s') \Big\}. \tag{17.8}$$

The maximizing actions $a \in A$ in these equations define an action to take at each state $s \in S$, that is a policy $\pi$. We can thus rewrite these equations in matrix terms as follows:

$$\mathbf{\Phi}(\mathbf{V}) = \max_{\pi} \{ \mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V} \}, \tag{17.9}$$

where $\mathbf{P}_\pi$ is the transition probability matrix defined by $(\mathbf{P}_\pi)_{ss'} = \mathbb{P}[s'|s,\pi(s)]$ for all $s, s' \in S$, and $\mathbf{R}_\pi$ the reward vector defined by $(\mathbf{R}_\pi)_s = \mathbb{E}[r(s,\pi(s)]$, for all $s \in S$.

The algorithm is directly based on (17.9). The pseudocode is given above. Starting from an arbitrary policy value vector $\mathbf{V}_0 \in \mathbb{R}^{|S|}$, the algorithm iteratively applies

VALUEITERATION($\mathbf{V}_0$)

1 $\mathbf{V} \leftarrow \mathbf{V}_0$ ▷ $\mathbf{V}_0$ arbitrary value
2 **while** $\|\mathbf{V} - \mathbf{\Phi}(\mathbf{V})\| \geq \frac{(1-\gamma)\epsilon}{\gamma}$ **do**
3 $\mathbf{V} \leftarrow \mathbf{\Phi}(\mathbf{V})$
4 **return** $\mathbf{\Phi}(\mathbf{V})$

**Figure 17.4**
Value iteration algorithm.

$\mathbf{\Phi}$ to the current $\mathbf{V}$ to obtain a new policy value vector until $\|\mathbf{V} - \mathbf{\Phi}(\mathbf{V})\| < \frac{(1-\gamma)\epsilon}{\gamma}$, where $\epsilon > 0$ is a desired approximation. The following theorem proves the convergence of the algorithm to the optimal policy values.

**Theorem 17.11** *For any initial value $\mathbf{V}_0$, the sequence defined by $\mathbf{V}_{n+1} = \mathbf{\Phi}(\mathbf{V}_n)$ converges to $\mathbf{V}^*$.*

Proof: We first show that $\mathbf{\Phi}$ is $\gamma$-Lipschitz for the $\|\cdot\|_\infty$.[23] For any $s \in S$ and $\mathbf{V} \in \mathbb{R}^{|S|}$, let $a^*(s)$ be the maximizing action defining $\mathbf{\Phi}(\mathbf{V})(s)$ in (17.8). Then, for any $s \in S$ and any $\mathbf{U} \in \mathbb{R}^{|S|}$,

$$\begin{aligned}\mathbf{\Phi}(\mathbf{V})(s) - \mathbf{\Phi}(\mathbf{U})(s) &\leq \mathbf{\Phi}(\mathbf{V})(s) - \Big(\mathbb{E}[r(s, a^*(s))] + \gamma \sum_{s' \in S} \mathbb{P}[s' \mid s, a^*(s)]\mathbf{U}(s')\Big) \\ &= \gamma \sum_{s' \in S} \mathbb{P}[s'|s, a^*(s)][\mathbf{V}(s') - \mathbf{U}(s')] \\ &\leq \gamma \sum_{s' \in S} \mathbb{P}[s'|s, a^*(s)]\|\mathbf{V} - \mathbf{U}\|_\infty = \gamma\|\mathbf{V} - \mathbf{U}\|_\infty.\end{aligned}$$

Proceeding similarly with $\mathbf{\Phi}(\mathbf{U})(s) - \mathbf{\Phi}(\mathbf{V})(s)$, we obtain $\mathbf{\Phi}(\mathbf{U})(s) - \mathbf{\Phi}(\mathbf{V})(s) \leq \gamma\|\mathbf{V} - \mathbf{U}\|_\infty$. Thus, $|\mathbf{\Phi}(\mathbf{V})(s) - \mathbf{\Phi}(\mathbf{U})(s)| \leq \gamma\|\mathbf{V} - \mathbf{U}\|_\infty$ for all $s$, which implies

$$\|\mathbf{\Phi}(\mathbf{V}) - \mathbf{\Phi}(\mathbf{U})\|_\infty \leq \gamma\|\mathbf{V} - \mathbf{U}\|_\infty,$$

[23] A $\beta$-Lipschitz function with $\beta < 1$ is also called $\beta$-*contracting*. In a *complete metric space*, that is a metric space where any Cauchy sequence converges to a point of that space, a $\beta$-contracting function $f$ admits a *fixed point*: any sequence $(f(x_n))_{n\in\mathbb{N}}$ converges to some $x$ with $f(x) = x$. $\mathbb{R}^N$, $N \geq 1$, or, more generally, any finite-dimensional vector space, is a complete metric space.

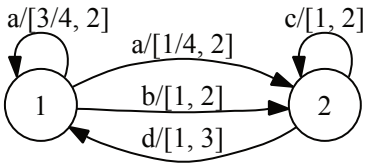


**Figure 17.5**
Example of MDP with two states. The state set is reduced to $S = \{1, 2\}$ and the action set to $A = \{a, b, c, d\}$. Only transitions with non-zero probabilities are represented. Each transition is labeled with the action taken followed by a pair $[p, r]$ after a slash separator, where $p$ is the probability of the transition and $r$ the expected reward for taking that transition.

that is the $\gamma$-Lipschitz property of $\mathbf{\Phi}$. Now, by Bellman equations (17.4), $\mathbf{V}^* = \mathbf{\Phi}(\mathbf{V}^*)$, thus for any $n \in \mathbb{N}$,

$$\|\mathbf{V}^* - \mathbf{V}_{n+1}\|_\infty = \|\mathbf{\Phi}(\mathbf{V}^*) - \mathbf{\Phi}(\mathbf{V}_n)\|_\infty \leq \gamma \|\mathbf{V}^* - \mathbf{V}_n\|_\infty \leq \gamma^{n+1} \|\mathbf{V}^* - \mathbf{V}_0\|_\infty,$$

which proves the convergence of the sequence to $\mathbf{V}^*$ since $\gamma \in (0, 1)$. □

The $\epsilon$-optimality of the value returned by the algorithm can be shown as follows. By the triangle inequality and the $\gamma$-Lipschitz property of $\mathbf{\Phi}$, for any $n \in \mathbb{N}$,

$$\begin{aligned}\|\mathbf{V}^* - \mathbf{V}_{n+1}\|_\infty &\leq \|\mathbf{V}^* - \mathbf{\Phi}(\mathbf{V}_{n+1})\|_\infty + \|\mathbf{\Phi}(\mathbf{V}_{n+1}) - \mathbf{V}_{n+1}\|_\infty \\ &= \|\mathbf{\Phi}(\mathbf{V}^*) - \mathbf{\Phi}(\mathbf{V}_{n+1})\|_\infty + \|\mathbf{\Phi}(\mathbf{V}_{n+1}) - \mathbf{\Phi}(\mathbf{V}_n)\|_\infty \\ &\leq \gamma \|\mathbf{V}^* - \mathbf{V}_{n+1}\|_\infty + \gamma \|\mathbf{V}_{n+1} - \mathbf{V}_n\|_\infty.\end{aligned}$$

Thus, if $\mathbf{V}_{n+1}$ is the policy value returned by the algorithm, we have

$$\|\mathbf{V}^* - \mathbf{V}_{n+1}\|_\infty \leq \frac{\gamma}{1-\gamma} \|\mathbf{V}_{n+1} - \mathbf{V}_n\|_\infty \leq \epsilon.$$

The convergence of the algorithm is in $O(\log \frac{1}{\epsilon})$ number of iterations. Indeed, observe that

$$\|\mathbf{V}_{n+1} - \mathbf{V}_n\|_\infty = \|\mathbf{\Phi}(\mathbf{V}_n) - \mathbf{\Phi}(\mathbf{V}_{n-1})\|_\infty \leq \gamma \|\mathbf{V}_n - \mathbf{V}_{n-1}\|_\infty \leq \gamma^n \|\mathbf{\Phi}(\mathbf{V}_0) - \mathbf{V}_0\|_\infty.$$

Thus, if $n$ is the largest integer such that $\frac{(1-\gamma)\epsilon}{\gamma} \leq \|\mathbf{V}_{n+1} - \mathbf{V}_n\|_\infty$, it must verify $\frac{(1-\gamma)\epsilon}{\gamma} \leq \gamma^n \|\mathbf{\Phi}(\mathbf{V}_0) - \mathbf{V}_0\|_\infty$ and therefore $n \leq O\big(\log \frac{1}{\epsilon}\big)$.[24]

[24] Here, the $O$-notation hides the dependency on the discount factor $\gamma$. As a function of $\gamma$, the running time is not polynomial.

PolicyIteration($\pi_0$)

1 $\pi \leftarrow \pi_0$ ▷ $\pi_0$ arbitrary policy
2 $\pi' \leftarrow$ NIL
3 **while** $(\pi \neq \pi')$ **do**
4 $\quad \mathbf{V} \leftarrow \mathbf{V}_\pi$ ▷ policy evaluation: solve $(\mathbf{I} - \gamma \mathbf{P}_\pi)\mathbf{V} = \mathbf{R}_\pi$.
5 $\quad \pi' \leftarrow \pi$
6 $\quad \pi \leftarrow \text{argmax}_\pi\{\mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V}\}$ ▷ greedy policy improvement.
7 **return** $\pi$

**Figure 17.6**
Policy iteration algorithm.

Figure 17.5 shows a simple example of MDP with two states. The iterated values of these states calculated by the algorithm for that MDP are given by

$$\mathbf{V}_{n+1}(1) = \max\left\{2 + \gamma\Big(\frac{3}{4}\mathbf{V}_n(1) + \frac{1}{4}\mathbf{V}_n(2)\Big), 2 + \gamma \mathbf{V}_n(2)\right\}$$
$$\mathbf{V}_{n+1}(2) = \max\left\{3 + \gamma \mathbf{V}_n(1), 2 + \gamma \mathbf{V}_n(2)\right\}.$$

For $\mathbf{V}_0(1) = -1$, $\mathbf{V}_0(2) = 1$, and $\gamma = 1/2$, we obtain $\mathbf{V}_1(1) = \mathbf{V}_1(2) = 5/2$. Thus, both states seem to have the same policy value initially. However, by the fifth iteration, $\mathbf{V}_5(1) = 4.53125$, $\mathbf{V}_5(2) = 5.15625$ and the algorithm quickly converges to the optimal values $\mathbf{V}^*(1) = 14/3$ and $\mathbf{V}^*(2) = 16/3$ showing that state 2 has a higher optimal value.

### 17.4.2 Policy iteration

An alternative algorithm for determining the best policy consists of using policy evaluations, which can be achieved via a matrix inversion, as shown by theorem 17.10. The pseudocode of the algorithm known as *policy iteration algorithm* is given in figure 17.6. Starting with an arbitrary action policy $\pi_0$, the algorithm repeatedly computes the value of the current policy $\pi$ via that matrix inversion and greedily selects the new policy as the one maximizing the right-hand side of the Bellman equations (17.9).

The following theorem proves the convergence of the policy iteration algorithm.

**Theorem 17.12** *Let $(\mathbf{V}_n)_{n\in\mathbb{N}}$ be the sequence of policy values computed by the algorithm, then, for any $n \in \mathbb{N}$, the following inequalities hold:*

$$\mathbf{V}_n \leq \mathbf{V}_{n+1} \leq \mathbf{V}^*. \tag{17.10}$$

Proof: Let $\pi_{n+1}$ be the policy improvement at the $n$th iteration of the algorithm. We first show that $(\mathbf{I} - \gamma\mathbf{P}_{\pi_{n+1}})^{-1}$ preserves ordering, that is, for any column matrices $\mathbf{X}$ and $\mathbf{Y}$ in $\mathbb{R}^{|S|}$, if $(\mathbf{Y} - \mathbf{X}) \geq \mathbf{0}$, then $(\mathbf{I} - \gamma\mathbf{P}_{\pi_{n+1}})^{-1}(\mathbf{Y} - \mathbf{X}) \geq \mathbf{0}$. As shown in the proof of theorem 17.10, $\|\gamma\mathbf{P}\|_\infty = \gamma < 1$. Since the radius of convergence of the power series $(1-x)^{-1}$ is one, we can use its expansion and write

$$(\mathbf{I} - \gamma\mathbf{P}_{\pi_{n+1}})^{-1} = \sum_{k=0}^{\infty}(\gamma\mathbf{P}_{\pi_{n+1}})^k.$$

Thus, if $\mathbf{Z} = (\mathbf{Y} - \mathbf{X}) \geq \mathbf{0}$, then $(\mathbf{I} - \gamma\mathbf{P}_{\pi_{n+1}})^{-1}\mathbf{Z} = \sum_{k=0}^{\infty}(\gamma\mathbf{P}_{\pi_{n+1}})^k\mathbf{Z} \geq \mathbf{0}$, since the entries of matrix $\mathbf{P}_{\pi_{n+1}}$ and its powers are all non-negative as well as those of $\mathbf{Z}$.

Now, by definition of $\pi_{n+1}$, we have

$$\mathbf{R}_{\pi_{n+1}} + \gamma\mathbf{P}_{\pi_{n+1}}\mathbf{V}_n \geq \mathbf{R}_{\pi_n} + \gamma\mathbf{P}_{\pi_n}\mathbf{V}_n = \mathbf{V}_n,$$

which shows that $\mathbf{R}_{\pi_{n+1}} \geq (\mathbf{I} - \gamma\mathbf{P}_{\pi_{n+1}})\mathbf{V}_n$. Since $(\mathbf{I} - \gamma\mathbf{P}_{\pi_{n+1}})^{-1}$ preserves ordering, this implies that $\mathbf{V}_{n+1} = (\mathbf{I} - \gamma\mathbf{P}_{\pi_{n+1}})^{-1}\mathbf{R}_{\pi_{n+1}} \geq \mathbf{V}_n$, which concludes the proof of the theorem. □

Note that two consecutive policy values can be equal only at the last iteration of the algorithm. The total number of possible policies is $|A|^{|S|}$, thus this constitutes a straightforward upper bound on the maximal number of iterations. Better upper bounds of the form $O\big(\frac{|A|^{|S|}}{|S|}\big)$ are known for this algorithm.

For the simple MDP shown by figure 17.5, let the initial policy $\pi_0$ be defined by $\pi_0(1) = b, \pi_0(2) = c$. Then, the system of linear equations for evaluating this policy is

$$\begin{cases} V_{\pi_0}(1) = 1 + \gamma V_{\pi_0}(2) \\ V_{\pi_0}(2) = 2 + \gamma V_{\pi_0}(2), \end{cases}$$

which gives $V_{\pi_0}(1) = \frac{1+\gamma}{1-\gamma}$ and $V_{\pi_0}(2) = \frac{2}{1-\gamma}$.

**Theorem 17.13** *Let $(\mathbf{U}_n)_{n\in\mathbb{N}}$ be the sequence of policy values generated by the value iteration algorithm, and $(\mathbf{V}_n)_{n\in\mathbb{N}}$ the one generated by the policy iteration algorithm. If $\mathbf{U}_0 = \mathbf{V}_0$, then,*

$$\forall n \in \mathbb{N},\ \mathbf{U}_n \leq \mathbf{V}_n \leq \mathbf{V}^*. \tag{17.11}$$

Proof: We first show that the function $\mathbf{\Phi}$ previously introduced is monotonic. Let $\mathbf{U}$ and $\mathbf{V}$ be such that $\mathbf{U} \leq \mathbf{V}$ and let $\pi$ be the policy such that $\mathbf{\Phi}(\mathbf{U}) = \mathbf{R}_\pi + \gamma\mathbf{P}_\pi\mathbf{U}$.

Then,

$$\mathbf{\Phi}(\mathbf{U}) \leq \mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V} \leq \max_{\pi'} \{\mathbf{R}_{\pi'} + \gamma \mathbf{P}_{\pi'} \mathbf{V}\} = \mathbf{\Phi}(\mathbf{V}).$$

The proof is by induction on $n$. Assume that $\mathbf{U}_n \leq \mathbf{V}_n$, then by the monotonicity of $\mathbf{\Phi}$, we have

$$\mathbf{U}_{n+1} = \mathbf{\Phi}(\mathbf{U}_n) \leq \mathbf{\Phi}(\mathbf{V}_n) = \max_{\pi} \{\mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V}_n\}.$$

Let $\pi_{n+1}$ be the maximizing policy, that is, $\pi_{n+1} = \text{argmax}_\pi \{\mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V}_n\}$. Then,

$$\mathbf{\Phi}(\mathbf{V}_n) = \mathbf{R}_{\pi_{n+1}} + \gamma \mathbf{P}_{\pi_{n+1}} \mathbf{V}_n \leq \mathbf{R}_{\pi_{n+1}} + \gamma \mathbf{P}_{\pi_{n+1}} \mathbf{V}_{n+1} = \mathbf{V}_{n+1},$$

and thus $\mathbf{U}_{n+1} \leq \mathbf{V}_{n+1}$. □

The theorem shows that the policy iteration algorithm converges in a smaller number of iterations than the value iteration algorithm due to the optimal policy. But, each iteration of the policy iteration algorithm requires computing a policy value, that is, solving a system of linear equations, which is more expensive to compute than an iteration of the value iteration algorithm.

### 17.4.3 Linear programming

An alternative formulation of the optimization problem defined by the Bellman equations (17.4) or the proof of Theorem 17.8 is via linear programming (LP), that is an optimization problem with a linear objective function and linear constraints. LPs admit (weakly) polynomial-time algorithmic solutions. There exist a variety of different methods for solving relatively large LPs in practice, using the simplex method, interior-point methods, or a variety of special-purpose solutions. All of these methods could be applied in this context.

By definition, the equations (17.4) are each based on a maximization. These maximizations are equivalent to seeking to minimize all elements of $\{V(s) \colon s \in S\}$ under the constraints $V(s) \geq \mathbb{E}[r(s,a)] + \gamma \sum_{s' \in S} \mathbb{P}[s'|s,a] V(s')$, $(s \in S)$. Thus, this can be written as the following LP for any set of fixed positive weights $\alpha(s) > 0$, $(s \in S)$:

$$\min_{\mathbf{V}} \quad \sum_{s \in S} \alpha(s) V(s) \tag{17.12}$$

$$\text{subject to} \quad \forall s \in S, \forall a \in A, V(s) \geq \mathbb{E}[r(s,a)] + \gamma \sum_{s' \in S} \mathbb{P}[s'|s,a] V(s'),$$

where $\boldsymbol{\alpha} > \mathbf{0}$ is the vector with the $s$th component equal to $\alpha(s)$.[25] To make each coefficient $\alpha(s)$ interpretable as a probability, we can further add the constraints that $\sum_{s \in S} \alpha(s) = 1$. The number of rows of this LP is $|S||A|$ and its number of columns $|S|$. The complexity of the solution techniques for LPs is typically more favorable in terms of the number of rows than the number of columns. This motivates a solution based on the equivalent dual formulation of this LP which can be written as

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{s \in S, a \in A} \mathbb{E}[r(s,a)]\, x(s,a) \\ \text{subject to} \quad & \forall s \in S, \sum_{a \in A} x(s',a) = \alpha(s') + \gamma \sum_{s \in S, a \in A} \mathbb{P}[s'|s,a]\, x(s',a) \\ & \forall s \in S, \forall a \in A, x(s,a) \geq 0, \end{aligned} \tag{17.13}$$

and for which the number of rows is only $|S|$ and the number of columns $|S||A|$. Here $x(s,a)$ can be interpreted as the probability of being in state $s$ and taking action $a$.

## 17.5 Learning algorithms

This section considers the more general scenario where the environment model of an MDP, that is the transition and reward probabilities, is unknown. This matches many realistic applications of reinforcement learning where, for example, a robot is placed in an environment that it needs to explore in order to reach a specific goal.

How can an agent determine the best policy in this context? Since the environment models are not known, it may seek to learn them by estimating transition or reward probabilities. To do so, as in the standard case of supervised learning, the agent needs some amount of training information. In the context of reinforcement learning with MDPs, the training information is the sequence of immediate rewards the agent receives based on the actions it has taken.

There are two main learning approaches that can be adopted. One known as the *model-free approach* consists of learning an action policy directly. Another one, a *model-based* approach, consists of first learning the environment model, and then of using that to learn a policy. The Q-learning algorithm we present for this problem is widely adopted in reinforcement learning and belongs to the family of model-free approaches.

[25] Let us emphasize that the LP is only in terms of the variables $V(s)$, as indicated by the subscript of the minimization operator, and not in terms of $V(s)$ and $\alpha(s)$.

The estimation and algorithmic methods adopted for learning in reinforcement learning are closely related to the concepts and techniques in *stochastic approximation*. Thus, we start by introducing several useful results of this field that will be needed for the proofs of convergence of the reinforcement learning algorithms presented.

### 17.5.1 Stochastic approximation

Stochastic approximation methods are iterative algorithms for solving optimization problems whose objective function is defined as the expectation of some random variable, or to find the fixed point of a function $H$ that is accessible only through noisy observations. These are precisely the type of optimization problems found in reinforcement learning. For example, for the Q-learning algorithm we will describe, the optimal state-action value function $Q^*$ is the fixed point of some function $H$ that is defined as an expectation and thus not directly accessible.

We start with a basic result whose proof and related algorithm show the flavor of more complex ones found in stochastic approximation. The theorem is a generalization of a result known as the *strong law of large numbers*. It shows that under some conditions on the coefficients, an iterative sequence of estimates $\mu_m$ converges almost surely (a.s.) to the mean of a bounded random variable.

**Theorem 17.14 (Mean estimation)** *Let $X$ be a random variable taking values in $[0,1]$ and let $x_0, \ldots, x_m$ be i.i.d. values of $X$. Define the sequence $(\mu_m)_{m\in\mathbb{N}}$ by*

$$\mu_{m+1} = (1-\alpha_m)\mu_m + \alpha_m x_m, \tag{17.14}$$

*with $\mu_0 = x_0$, $\alpha_m \in [0,1]$, $\sum_{m\geq 0} \alpha_m = +\infty$ and $\sum_{m\geq 0} \alpha_m^2 < +\infty$. Then,*

$$\mu_m \xrightarrow{a.s.} \mathbb{E}[X]. \tag{17.15}$$

Proof: We give the proof of the $L_2$ convergence. The a.s. convergence is shown later for a more general theorem. By the independence assumption, for $m \geq 0$,

$$\operatorname{Var}[\mu_{m+1}] = (1-\alpha_m)^2 \operatorname{Var}[\mu_m] + \alpha_m^2 \operatorname{Var}[x_m] \leq (1-\alpha_m)\operatorname{Var}[\mu_m] + \alpha_m^2. \tag{17.16}$$

Let $\epsilon > 0$ and suppose that there exists $N \in \mathbb{N}$ such that for all $m \geq N$, $\operatorname{Var}[\mu_m] \geq \epsilon$. Then, for $m \geq N$,

$$\operatorname{Var}[\mu_{m+1}] \leq \operatorname{Var}[\mu_m] - \alpha_m \operatorname{Var}[\mu_m] + \alpha_m^2 \leq \operatorname{Var}[\mu_m] - \alpha_m \epsilon + \alpha_m^2,$$

which implies, by reapplying this inequality, that

$$\operatorname{Var}[\mu_{m+N}] \leq \underbrace{\operatorname{Var}[\mu_N] - \epsilon \sum_{n=N}^{m+N} \alpha_n + \sum_{n=N}^{m+N} \alpha_n^2}_{\to -\infty \text{ when } m \to \infty},$$

contradicting $\text{Var}[\mu_{m+N}] \geq 0$. Thus, this contradicts the existence of such an integer $N$. Therefore, for all $N \in \mathbb{N}$, there exists $m_0 \geq N$ such that $\text{Var}[\mu_{m_0}] \leq \epsilon$.

Choose $N$ large enough so that for all $m \geq N$, the inequality $\alpha_m \leq \epsilon$ holds. This is possible since the sequence $(\alpha_m^2)_{m\in\mathbb{N}}$ and thus $(\alpha_m)_{m\in\mathbb{N}}$ converges to zero in view of $\sum_{m\geq 0} \alpha_m^2 < +\infty$. We will show by induction that for any $m \geq m_0$, $\text{Var}[\mu_m] \leq \epsilon$, which implies the statement of the theorem.

Assume that $\text{Var}[\mu_m] \leq \epsilon$ for some $m \geq m_0$. Then, using this assumption, inequality 17.16, and the fact that $\alpha_m \leq \epsilon$, the following inequality holds:

$$\text{Var}[\mu_{m+1}] \leq (1 - \alpha_m)\epsilon + \epsilon\alpha_m = \epsilon.$$

Thus, this proves that $\lim_{m\to+\infty} \text{Var}[\mu_m] = 0$, that is the $L_2$ convergence of $\mu_m$ to $\mathbb{E}[X]$. □

Note that the hypotheses of the theorem related to the sequence $(\alpha_m)_{m\in\mathbb{N}}$ hold in particular when $\alpha_m = \frac{1}{m}$. The special case of the theorem with this choice of $\alpha_m$ coincides with the strong law of large numbers. This result has tight connections with the general problem of stochastic optimization.

Stochastic optimization is the general problem of finding the solution to the equation

$$\mathbf{x} = H(\mathbf{x}),$$

where $\mathbf{x} \in \mathbb{R}^N$, when

- $H(x)$ cannot be computed, for example, because $H$ is not accessible or because the cost of its computation is prohibitive;
- but an i.i.d. sample of $m$ noisy observations $H(\mathbf{x}_i) + \mathbf{w}_i$ are available, $i \in [m]$, where the noise random variable $\mathbf{w}$ has expectation zero: $\mathbb{E}[\mathbf{w}] = \mathbf{0}$.

This problem arises in a variety of different contexts and applications. As we shall see, it is directly related to the learning problem for MDPs.

One general idea for solving this problem is to use an iterative method and define a sequence $(\mathbf{x}_t)_{t\in\mathbb{N}}$ in a way similar to what is suggested by theorem 17.14:

$$\mathbf{x}_{t+1} = (1 - \alpha_t)\mathbf{x}_t + \alpha_t[H(\mathbf{x}_t) + \mathbf{w}_t] \tag{17.17}$$

$$= \mathbf{x}_t + \alpha_t[H(\mathbf{x}_t) + \mathbf{w}_t - \mathbf{x}_t], \tag{17.18}$$

where $(\alpha_t)_{t\in\mathbb{N}}$ follow conditions similar to those assumed in theorem 17.14. More generally, we consider sequences defined via

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha_t D(\mathbf{x}_t, \mathbf{w}_t), \tag{17.19}$$

where $D$ is a function mapping $\mathbb{R}^N \times \mathbb{R}^N$ to $\mathbb{R}^N$. There are many different theorems guaranteeing the convergence of this sequence under various assumptions. We will present one of the most general forms of such theorems, which relies on the following result.

**Theorem 17.15 (Supermartingale convergence)** *Let $(X_t)_{t\in\mathbb{N}}$, $(Y_t)_{t\in\mathbb{N}}$, and $(Z_t)_{t\in\mathbb{N}}$ be sequences of non-negative random variables such that $\sum_{t=0}^{+\infty} Y_t < +\infty$. Let $\mathcal{F}_t$ denote all the information for $t' \leq t$: $\mathcal{F}_t = \{(X_{t'})_{t'\leq t}, (Y_{t'})_{t'\leq t}, (Z_{t'})_{t'\leq t}\}$. Then, if $\mathbb{E}\left[X_{t+1}\middle|\mathcal{F}_t\right] \leq X_t + Y_t - Z_t$, the following holds:*

- *$X_t$ converges to a limit (with probability one).*
- *$\sum_{t=0}^{+\infty} Z_t < +\infty$.*

The following is one of the most general forms of such theorems.

**Theorem 17.16** *Let $D$ be a function mapping $\mathbb{R}^N \times \mathbb{R}^N$ to $\mathbb{R}^N$, $(\mathbf{w}_t)_{t\in\mathbb{N}}$ a sequence of random variables in $\mathbb{R}^N$, $(\alpha_t)_{t\in\mathbb{N}}$ a sequence of real numbers, and $(\mathbf{x}_t)_{t\in\mathbb{N}}$ a sequence defined by $\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha_t D(\mathbf{x}_t, \mathbf{w}_t)$ with $\mathbf{x}_0 \in \mathbb{R}^N$. Let $\mathcal{F}_t$ denote the entire history up to $t$, that is: $\mathcal{F}_t = \{(\mathbf{x}_{t'})_{t'\leq t}, (\mathbf{w}_{t'})_{t'\leq t-1}, (\alpha_{t'})_{t'\leq t}\}$, and let $\Psi$ denote the function $\mathbf{x} \to \frac{1}{2}\|\mathbf{x}-\mathbf{x}^*\|_2^2$ for some $\mathbf{x}^* \in \mathbb{R}^N$. Assume that $D$ and $(\alpha)_{t\in\mathbb{N}}$ verify the following conditions:*

- *$\exists K_1, K_2 \in \mathbb{R}\colon \mathbb{E}\left[\|D(\mathbf{x}_t, \mathbf{w}_t)\|_2^2 \,\middle|\, \mathcal{F}_t\right] \leq K_1 + K_2\,\Psi(\mathbf{x}_t)$;*
- *$\exists c \geq 0\colon \nabla\Psi(\mathbf{x}_t)^\top \mathbb{E}\left[D(\mathbf{x}_t, \mathbf{w}_t) \,\middle|\, \mathcal{F}_t\right] \leq -c\,\Psi(\mathbf{x}_t)$;*
- *$\alpha_t > 0, \sum_{t=0}^{+\infty} \alpha_t = +\infty, \sum_{t=0}^{+\infty} \alpha_t^2 < +\infty$.*

*Then, the sequence $\mathbf{x}_t$ converges almost surely to $\mathbf{x}^*$:*

$$\mathbf{x}_t \xrightarrow{a.s.} \mathbf{x}^*. \tag{17.20}$$

Proof: Since function $\Psi$ is quadratic, a Taylor expansion gives

$$\Psi(\mathbf{x}_{t+1}) = \Psi(\mathbf{x}_t) + \nabla\Psi(\mathbf{x}_t)^\top(\mathbf{x}_{t+1} - \mathbf{x}_t) + \frac{1}{2}(\mathbf{x}_{t+1} - \mathbf{x}_t)^\top \nabla^2\Psi(\mathbf{x}_t)(\mathbf{x}_{t+1} - \mathbf{x}_t).$$

Thus,

$$\begin{aligned}
\mathbb{E}\left[\Psi(\mathbf{x}_{t+1})\middle|\mathcal{F}_t\right] &= \Psi(\mathbf{x}_t) + \alpha_t \nabla\Psi(\mathbf{x}_t)^\top \mathbb{E}\left[D(\mathbf{x}_t, \mathbf{w}_t)\middle|\mathcal{F}_t\right] + \frac{\alpha_t^2}{2}\,\mathbb{E}\left[\|D(\mathbf{x}_t, \mathbf{w}_t)\|^2\middle|\mathcal{F}_t\right] \\
&\leq \Psi(\mathbf{x}_t) - \alpha_t c \Psi(\mathbf{x}_t) + \frac{\alpha_t^2}{2}(K_1 + K_2\Psi(\mathbf{x}_t)) \\
&= \Psi(\mathbf{x}_t) + \frac{\alpha_t^2 K_1}{2} - \Big(\alpha_t c - \frac{\alpha_t^2 K_2}{2}\Big)\Psi(\mathbf{x}_t).
\end{aligned}$$

Since by assumption the series $\sum_{t=0}^{+\infty} \alpha_t^2$ is convergent, $(\alpha_t^2)_t$ and thus $(\alpha_t)_t$ converges to zero. Therefore, for $t$ sufficiently large, the term $\big(\alpha_t c - \frac{\alpha_t^2 K_2}{2}\big)\Psi(\mathbf{x}_t)$ has the sign of $\alpha_t c \Psi(\mathbf{x}_t)$ and is non-negative, since $\alpha_t > 0$, $\Psi(\mathbf{x}_t) \geq 0$, and $c > 0$. Thus, by the supermartingale convergence theorem 17.15, $\Psi(\mathbf{x}_t)$ converges and $\sum_{t=0}^{+\infty} \big(\alpha_t c - \frac{\alpha_t^2 K_2}{2}\big)\Psi(\mathbf{x}_t) < +\infty$. Since $\Psi(\mathbf{x}_t)$ converges and $\sum_{t=0}^{+\infty} \alpha_t^2 < +\infty$, we have $\sum_{t=0}^{+\infty} \frac{\alpha_t^2 K_2}{2}\Psi(\mathbf{x}_t) < +\infty$. But, since $\sum_{t=0}^{+\infty} \alpha_t = +\infty$, if the limit of $\Psi(\mathbf{x}_t)$ were non-zero, we would have $\sum_{t=0}^{+\infty} \alpha_t c \Psi(\mathbf{x}_t) = +\infty$. This implies

that the limit of $\Psi(\mathbf{x}_t)$ is zero, that is $\lim_{t\to\infty} \|\mathbf{x}_t - \mathbf{x}^*\|_2 \to 0$, which implies $\mathbf{x}_t \xrightarrow{\text{a.s.}} \mathbf{x}^*$. □

The following is another related result for which we do not present the full proof.

**Theorem 17.17** *Let $\mathbf{H}$ be a function mapping $\mathbb{R}^N$ to $\mathbb{R}^N$, $(\mathbf{w}_t)_{t\in\mathbb{N}}$ a sequence of random variables in $\mathbb{R}^N$, $(\alpha_t)_{t\in\mathbb{N}}$ a sequence of real numbers, and $(\mathbf{x}_t)_{t\in\mathbb{N}}$ a sequence defined by*

$$\forall s \in [N], \quad \mathbf{x}_{t+1}(s) = \mathbf{x}_t(s) + \alpha_t(s)\big[\mathbf{H}(\mathbf{x}_t)(s) - \mathbf{x}_t(s) + \mathbf{w}_t(s)\big],$$

*for some $\mathbf{x}_0 \in \mathbb{R}^N$. Define $\mathcal{F}_t$ by $\mathcal{F}_t = \{(\mathbf{x}_{t'})_{t'\leq t}, (\mathbf{w}_{t'})_{t'\leq t-1}(\alpha_{t'})_{t'\leq t}\}$, that is the entire history up to $t$, and assume that the following conditions are met:*

- $\exists K_1, K_2 \in \mathbb{R}\colon \mathbb{E}\left[\|\mathbf{w}_t\|^2(s) \,\middle|\, \mathcal{F}_t\right] \leq K_1 + K_2 \|\mathbf{x}_t\|^2$ *for some norm $\|\cdot\|$;*
- $\mathbb{E}\left[\mathbf{w}_t \,\middle|\, \mathcal{F}_t\right] = 0$*;*
- $\forall s \in [N], \sum_{t=0}^{+\infty} \alpha_t(s) = +\infty, \sum_{t=0}^{+\infty} \alpha_t^2(s) < +\infty$*; and*
- $\mathbf{H}$ *is a $\|\cdot\|_\infty$-contraction with fixed point $\mathbf{x}^*$.*

*Then, the sequence $\mathbf{x}_t$ converges almost surely to $\mathbf{x}^*$:*

$$\mathbf{x}_t \xrightarrow{a.s.} \mathbf{x}^*. \tag{17.21}$$

The next sections present several learning algorithms for MDPs with an unknown model.

### 17.5.2 TD(0) algorithm

This section presents an algorithm, TD(0) algorithm, for evaluating a policy in the case where the environment model is unknown. The algorithm is based on Bellman's linear equations giving the value of a policy $\pi$ (see proposition 17.9):

$$\begin{aligned} V_\pi(s) &= \mathbb{E}[r(s,\pi(s)] + \gamma \sum_{s'} \mathbb{P}[s'|s,\pi(s)] V_\pi(s') \\ &= \mathop{\mathbb{E}}_{s'}\big[r(s,\pi(s)) + \gamma V_\pi(s')|s\big]. \end{aligned}$$

However, here the probability distribution according to which this last expectation is defined is not known. Instead, the TD(0) algorithm consists of

- sampling a new state $s'$; and
- updating the policy values according to the following, which justifies the name of the algorithm:

$$\begin{aligned} V(s) &\leftarrow (1-\alpha)V(s) + \alpha[r(s,\pi(s)) + \gamma V(s')] \\ &= V(s) + \alpha[\underbrace{r(s,\pi(s)) + \gamma V(s') - V(s)}_{\text{temporal difference of } V \text{ values}}]. \end{aligned} \tag{17.22}$$

Here, the parameter $\alpha$ is a function of the number of visits to the state $s$.

```
TD(0)()
 1  V ← V_0 ▷ initialization.
 2  for t ← 0 to T do
 3       s ← SELECTSTATE()
 4       for each step of epoch t do
 5            r' ← REWARD(s, π(s))
 6            s' ← NEXTSTATE(π, s)
 7            V(s) ← (1 − α)V(s) + α[r' + γV(s')]
 8            s ← s'
 9  return V
```

The pseudocode of the algorithm is given above. The algorithm starts with an arbitrary policy value vector $\mathbf{V}_0$. An initial state is returned by SELECTSTATE at the beginning of each epoch. Within each epoch, the iteration continues until a final state is found. Within each iteration, action $\pi(s)$ is taken from the current state $s$ following policy $\pi$. The new state $s'$ reached and the reward $r'$ received are observed. The policy value of state $s$ is then updated according to the rule (17.22) and current state set to be $s'$.

The convergence of the algorithm can be proven using theorem 17.17. We will give instead the full proof of the convergence of the Q-learning algorithm, for which that of TD(0) can be viewed as a special case.

### 17.5.3 Q-learning algorithm

This section presents an algorithm for estimating the optimal state-action value function $Q^*$ in the case of an unknown model. Note that the optimal policy or policy value can be straightforwardly derived from $Q^*$ via: $\pi^*(s) = \operatorname{argmax}_{a\in A} Q^*(s,a)$ and $V^*(s) = \max_{a\in A} Q^*(s,a)$. To simplify the presentation, we will assume a deterministic reward function.

The Q-learning algorithm is based on the equations giving the optimal state-action value function $Q^*$ (17.1):

$$\begin{aligned} Q^*(s,a) &= \mathbb{E}[r(s,a)] + \gamma \sum_{s'\in S} \mathbb{P}[s' \mid s,a] V^*(s') \\ &= \mathop{\mathbb{E}}_{s'}[r(s,a) + \gamma \max_{a\in A} Q^*(s',a)]. \end{aligned}$$

Q-LEARNING($\pi$)

```
1  Q ← Q_0    ▷ initialization, e.g., Q_0 = 0.
2  for t ← 0 to T do
3      s ← SELECTSTATE()
4      for each step of epoch t do
5          a ← SELECTACTION(π, s) ▷   policy π derived from Q, e.g., ε-greedy.
6          r' ← REWARD(s, a)
7          s' ← NEXTSTATE(s, a)
8          Q(s, a) ← Q(s, a) + α[r' + γ max_{a'} Q(s', a') − Q(s, a)]
9          s ← s'
10 return Q
```

As for the policy values in the previous section, the distribution model is not known. Thus, the Q-learning algorithm consists of the following main steps:

- sampling a new state $s'$; and
- updating the policy values according to the following:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r(s,a) + \gamma \max_{a' \in A} Q(s',a')]. \tag{17.23}$$

where the parameter $\alpha$ is a function of the number of visits to the state $s$.

The algorithm can be viewed as a stochastic formulation of the value iteration algorithm presented in the previous section. The pseudocode is given above. Within each epoch, an action is selected from the current state $s$ using a policy $\pi$ derived from $Q$. The choice of the policy $\pi$ is arbitrary so long as it guarantees that every pair $(s,a)$ is visited infinitely many times. The reward received and the state $s'$ observed are then used to update $Q$ following (17.23).

**Theorem 17.18** *Consider a finite MDP. Assume that for all $s \in S$ and $a \in A$, $\sum_{t=0}^{+\infty} \alpha_t(s,a) = +\infty$, and $\sum_{t=0}^{+\infty} \alpha_t^2(s,a) < +\infty$ with $\alpha_t(s,a) \in [0,1]$. Then, the Q-learning algorithm converges to the optimal value $Q^*$ (with probability one).*

Note that the conditions on $\alpha_t(s,a)$ impose that each state-action pair is visited infinitely many times.

Proof: Let $(Q_t(s,a))_{t\geq 0}$ denote the sequence of state-action value functions at $(s,a) \in S \times A$ generated by the algorithm. By definition of the Q-learning updates,

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha\big[r(s_t,a_t) + \gamma \max_{a'} Q_t(s_{t+1},a') - Q_t(s_t,a_t)\big].$$

This can be rewritten as the following for all $s \in S$ and $a \in A$:

$$\begin{aligned} Q_{t+1}(s,a) = Q_t(s,a) &+ \alpha_t(s,a)\left[r(s,a) + \gamma \mathop{\mathbb{E}}_{u\sim\mathbb{P}[\cdot|s,a]}\left[\max_{a'} Q_t(u,a')\right] - Q_t(s,a)\right] \\ &+ \gamma\alpha_t(s,a)\left[\max_{a'} Q_t(s',a') - \mathop{\mathbb{E}}_{u\sim\mathbb{P}[\cdot|s,a]}\left[\max_{a'} Q_t(u,a')\right]\right], \end{aligned} \tag{17.24}$$

if we define $s' = \textsc{NextState}(s,a)$ and $\alpha_t(s,a)$ as 0 if $(s,a) \neq (s_t,a_t)$ and $\alpha_t(s_t,a_t)$ otherwise. Now, let $\mathbf{Q}_t$ denote the vector with components $Q_t(s,a)$, $\mathbf{w}_t$ the vector whose $s'$th entry is

$$w_t(s) = \max_{a'} Q_t(s',a') - \mathop{\mathbb{E}}_{u\sim\mathbb{P}[\cdot|s,a]}\left[\max_{a'} Q_t(u,a')\right],$$

and $\mathbf{H}(\mathbf{Q}_t)$ the vector with components $\mathbf{H}(\mathbf{Q}_t)(s,a)$ defined by

$$\mathbf{H}(\mathbf{Q}_t)(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{u\sim\mathbb{P}[\cdot|s,a]}\left[\max_{a'} Q_t(u,a')\right].$$

Then, in view of (17.24),

$$\forall (s,a) \in S\times A, \quad \mathbf{Q}_{t+1}(s,a) = \mathbf{Q}_t(s,a) + \alpha_t(s,a)\big[\mathbf{H}(\mathbf{Q}_t)(s,a) - \mathbf{Q}_t(s,a) + \gamma \mathbf{w}_t(s)\big].$$

We now show that the hypotheses of theorem 17.17 hold for $\mathbf{Q}_t$ and $\mathbf{w}_t$, which will imply the convergence of $\mathbf{Q}_t$ to $\mathbf{Q}^*$. The conditions on $\alpha_t$ hold by assumption. By definition of $\mathbf{w}_t$, $\mathbb{E}[\mathbf{w}_t \,\big|\, \mathcal{F}_t] = 0$. Also, for any $s' \in S$,

$$\begin{aligned} |\mathbf{w}_t(s)| &\leq \max_{a'} |Q_t(s',a')| + \left| \mathop{\mathbb{E}}_{u\sim\mathbb{P}[\cdot|s,a]}\left[\max_{a'} Q_t(u,a')\right]\right| \\ &\leq 2 \max_{s'} |\max_{a'} Q_t(s',a')| = 2\|\mathbf{Q}_t\|_\infty. \end{aligned}$$

Thus, $\mathbb{E}\left[\mathbf{w}_t^2(s) \,\middle|\, \mathcal{F}_t\right] \leq 4\|\mathbf{Q}_t\|_\infty^2$. Finally, $\mathbf{H}$ is a $\gamma$-contraction for $\|\cdot\|_\infty$ since for any $\mathbf{Q}_1, \mathbf{Q}_2 \in \mathbb{R}^{|S|\times|A|}$, and $(s,a) \in S \times A$, we can write

$$\begin{aligned}
|\mathbf{H}(\mathbf{Q}_2)(x,a) - \mathbf{H}(\mathbf{Q}_1)(x,a)| &= \left|\gamma \mathop{\mathbb{E}}_{u\sim\mathbb{P}[\cdot|s,a]}\left[\max_{a'} Q_2(u,a') - \max_{a'} Q_1(u,a')\right]\right| \\
&\leq \gamma \mathop{\mathbb{E}}_{u\sim\mathbb{P}[\cdot|s,a]}\left[\left|\max_{a'} Q_2(u,a') - \max_{a'} Q_1(u,a')\right|\right] \\
&\leq \gamma \mathop{\mathbb{E}}_{u\sim\mathbb{P}[\cdot|s,a]} \max_{a'} \left[|Q_2(u,a') - Q_1(u,a')|\right] \\
&\leq \gamma \max_u \max_{a'} \left[|Q_2(u,a') - Q_1(u,a')|\right] \\
&= \gamma \|\mathbf{Q}_2 - \mathbf{Q}_1\|_\infty.
\end{aligned}$$

Since $\mathbf{H}$ is a contraction, it admits a fixed point $\mathbf{Q}^*$: $\mathbf{H}(\mathbf{Q}^*) = \mathbf{Q}^*$. □

The choice of the policy $\pi$ according to which an action $a$ is selected (line 5) is not specified by the algorithm and, as already indicated, the theorem guarantees the convergence of the algorithm for an arbitrary policy so long as it ensures that every pair $(s,a)$ is visited infinitely many times. In practice, several natural choices are considered for $\pi$. One possible choice is the policy determined by the state-action value at time $t$, $Q_t$. Thus, the action selected from state $s$ is $\operatorname{argmax}_{a\in A} Q_t(s,a)$. But this choice typically does not guarantee that all actions are taken or that all states are visited. Instead, a standard choice in reinforcement learning is the so-called *$\epsilon$-greedy policy*, which consists of selecting with probability $(1-\epsilon)$ the greedy action from state $s$, that is, $\operatorname{argmax}_{a\in A} Q_t(s,a)$, and with probability $\epsilon$ a random action from $s$, for some $\epsilon \in (0,1)$. Another possible choice is the so-called *Boltzmann exploration*, which, given the current state-action value $Q$, epoch $t \in \{0,\ldots,T\}$, and current state $s$, consists of selecting action $a$ with the following probability:

$$p_t(a|s,Q) = \frac{e^{\frac{Q(s,a)}{\tau_t}}}{\sum_{a'\in A} e^{\frac{Q(s,a')}{\tau_t}}},$$

where $\tau_t$ is the *temperature*. $\tau_t$ must be defined so that $\tau_t \to 0$ as $t \to +\infty$, which ensures that for large values of $t$, the greedy action based on $Q$ is selected. This is natural, since as $t$ increases, we can expect $Q$ to be close to the optimal function. On the other hand, $\tau_t$ must be chosen so that it does not tend to 0 too fast to ensure that all actions are visited infinitely often. It can be chosen, for instance, as $1/\log(n_t(s))$, where $n_t(s)$ is the number of times $s$ has been visited up to epoch $t$.

Reinforcement learning algorithms include two components: a *learning policy*, which determines the action to take, and an *update rule*, which defines the new estimate of the optimal value function. For an *off-policy algorithm*, the update rule does not necessarily depend on the learning policy. Q-learning is an off-policy algorithm since its update rule (line 8 of the pseudocode) is based on the max

operator and the comparison of all possible actions $a'$, that is the greedy action, which may not coincide with the action recommended by the current the policy $\pi$. More generally, an off-policy algorithm evaluates or improves one policy, while acting based on another policy.

In contrast, the algorithm presented in the next section, SARSA, is an *on-policy algorithm*. An on-policy algorithm evaluates and improves the current policy used for control. It evaluates the return based on the algorithm's policy.

### 17.5.4 SARSA

SARSA is also an algorithm for estimating the optimal state-action value function in the case of an unknown model. The pseudocode is given in figure 17.7. The algorithm is in fact very similar to Q-learning, except that its update rule (line 9 of the pseudocode) is based on the action $a'$ selected by the learning policy. Thus, SARSA is an on-policy algorithm, and its convergence therefore crucially depends on the learning policy. In particular, the convergence of the algorithm requires, in addition to all actions being selected infinitely often, that the learning policy becomes greedy in the limit. The proof of the convergence of the algorithm is nevertheless close to that of Q-learning.

The name of the algorithm derives from the sequence of instructions defining successively $s$, $a$, $r'$, $s'$, and $a'$, and the fact that the update to the function $Q$ depends on the quintuple $(s, a, r', s', a)$.

### 17.5.5 TD($\lambda$) algorithm

Both TD(0) and Q-learning algorithms are only based on immediate rewards. The idea of TD($\lambda$) consists instead of using multiple steps ahead. Thus, for $n > 1$ steps, we would have the update

$$V(s) \leftarrow V(s) + \alpha\,(R_t^n - V(s)),$$

where $R_t^n$ is defined by

$$R_t^n = r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n}).$$

How should $n$ be chosen? Instead of selecting a specific $n$, TD($\lambda$) is based on a geometric distribution over all rewards $R_t^n$, that is, it uses $R_t^\lambda = (1-\lambda)\sum_{n=0}^{+\infty} \lambda^n R_t^n$ instead of $R_t^n$ where $\lambda \in [0, 1]$. Thus, the main update becomes

$$V(s) \leftarrow V(s) + \alpha\,(R_t^\lambda - V(s)).$$

The pseudocode of the algorithm is given above. For $\lambda = 0$, the algorithm coincides with TD(0). $\lambda = 1$ corresponds to the total future reward.

SARSA($\pi$)

$Q \leftarrow Q_0$ ▷ initialization, e.g., $Q_0 = 0$.
**for** $t \leftarrow 0$ **to** $T$ **do**
    $s \leftarrow$ SELECTSTATE()
    $a \leftarrow$ SELECTACTION($\pi(Q), s$) ▷ policy $\pi$ derived from $Q$, e.g., $\epsilon$-greedy.
    **for** each step of epoch $t$ **do**
        $r' \leftarrow$ REWARD($s, a$)
        $s' \leftarrow$ NEXTSTATE($s, a$)
        $a' \leftarrow$ SELECTACTION($\pi(Q), s'$) ▷ policy $\pi$ derived from $Q$, e.g., $\epsilon$-greedy.
        $Q(s,a) \leftarrow Q(s,a) + \alpha_t(s,a)\big[r' + \gamma Q(s',a') - Q(s,a)\big]$
        $s \leftarrow s'$
        $a \leftarrow a'$
**return** $Q$

**Figure 17.7**
The SARSA algorithm.

In the previous sections, we presented learning algorithms for an agent navigating in an unknown environment. The scenario faced in many practical applications is more challenging; often, the information the agent receives about the environment is uncertain or unreliable. Such problems can be modeled as partially observable Markov decision processes (POMDPs). POMDPs are defined by augmenting the definition of MDPs with an observation probability distribution depending on the action taken, the state reached, and the observation. The presentation of their model and solution techniques are beyond the scope of this material.

### 17.5.6 Large state space

In some cases in practice, the number of states or actions to consider for the environment may be very large. For example, the number of states in the game of backgammon is estimated to be over $10^{20}$. Thus, the algorithms presented in the previous section can become computationally impractical for such applications. More importantly, generalization becomes extremely difficult.

Suppose we wish to estimate the policy value $V_\pi(s)$ at each state $s$ using experience obtained using policy $\pi$. To cope with the case of large state spaces, we

```
TD(λ)()
 1  V ← V_0 ▷ initialization.
 2  e ← 0
 3  for t ← 0 to T do
 4       s ← SELECTSTATE()
 5       for each step of epoch t do
 6            s' ← NEXTSTATE(π, s)
 7            δ ← r(s, π(s)) + λV(s') − V(s)
 8            e(s) ← λe(s) + 1
 9            for u ∈ S do
10                 if u ≠ s then
11                      e(u) ← γλe(u)
12                 V(u) ← V(u) + αδe(u)
13            s ← s'
14  return V
```

can map each state of the environment to $\mathbb{R}^N$ via a mapping $\mathbf{\Phi}\colon S \to \mathbb{R}^N$, with $N$ relatively small ($N \approx 200$ has been used for backgammon) and approximate $V_\pi(s)$ by a function $f_{\mathbf{w}}(s)$ parameterized by some vector $\mathbf{w}$. For example, $f_{\mathbf{w}}$ could be a linear function defined by $f_{\mathbf{w}}(s) = \mathbf{w} \cdot \mathbf{\Phi}(s)$ for all $s \in S$, or some more complex non-linear function of $\mathbf{w}$. The problem then consists of approximating $V_\pi$ with $f_{\mathbf{w}}$ and can be formulated as a regression problem. Note, however, that the empirical data available is not i.i.d.

Suppose that at each time step $t$ the agent receives the exact policy value $V_\pi(s_t)$. Then, if the family of functions $f_{\mathbf{w}}$ is differentiable, a gradient descent method applied to the empirical squared loss can be used to sequentially update the weight vector $\mathbf{w}$ via:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla_{\mathbf{w}_t} \frac{1}{2}[V_\pi(s_t) - f_{\mathbf{w}_t}(s_t)]^2 = \mathbf{w}_t + \alpha[V_\pi(s_t) - f_{\mathbf{w}_t}(s_t)]\nabla_{\mathbf{w}_t} f_{\mathbf{w}_t}(s_t).$$

It is worth mentioning, however, that for large action spaces, there are simple cases where the methods used do not converge and instead cycle.

## 17.6 Chapter notes

Reinforcement learning is an important area of machine learning with a large body of literature. This chapter presents only a brief introduction to this area. For a more detailed study, the reader could consult the book of Sutton and Barto [1998], whose mathematical content is short, or those of Puterman [1994] and Bertsekas [1987], which discuss in more depth several aspects, as well as the more recent book of Szepesvári [2010]. The Ph.D. theses of Singh [1993] and Littman [1996] are also excellent sources.

Some foundational work on MDPs and the introduction of the temporal difference (TD) methods are due to Sutton [1984]. Q-learning was introduced and analyzed by Watkins [1989], though it can be viewed as a special instance of TD methods. The first proof of the convergence of Q-learning was given by Watkins and Dayan [1992].

Many of the techniques used in reinforcement learning are closely related to those of stochastic approximation which originated with the work of Robbins and Monro [1951], followed by a series of results including Dvoretzky [1956], Schmetterer [1960], Kiefer and Wolfowitz [1952], and Kushner and Clark [1978]. For a recent survey of stochastic approximation, including a discussion of powerful proof techniques based on ODE (ordinary differential equations), see Kushner [2010] and the references therein. The connection with stochastic approximation was emphasized by Tsitsiklis [1994] and Jaakkola et al. [1994], who gave a related proof of the convergence of Q-learning. For the convergence rate of Q-learning, consult Even-Dar and Mansour [2003]. For recent results on the convergence of the policy iteration algorithm, see Ye [2011], which shows that the algorithm is strongly polynomial for a fixed discount factor.

Reinforcement learning has been successfully applied to a variety of problems including robot control, board games such as backgammon in which Tesauro's TD-Gammon reached the level of a strong master [Tesauro, 1995] (see also chapter 11 of Sutton and Barto [1998]), chess, elevator scheduling problems [Crites and Barto, 1996], telecommunications, inventory management, dynamic radio channel assignment [Singh and Bertsekas, 1997], and a number of other problems (see chapter 1 of Puterman [1994]).