Reinforcement Learning
The Reinforcement Learning Problem

Contents

1 Vocubalary and notations

▶ The reinforcement learning problem is trying to make an *agent* (or learner, decision-maker) learns from its interactions with its *environment*. This environment generates *rewards* (scalar values) that the agent is seeking to maximize over time.

Given a sequence of time steps $t = 0, 1, \dots, n$, the agent receives at each t_i a representation of the environment state, denoted $s_t \in \mathcal{S}$ where $\mathcal{S}$ denotes the set of all possible states. On the basis of this *state signal*, the agent selects an action $a_t \in \mathcal{A}(s)$ where $\mathcal{A}(s)$ denote all the actions available in the state $s \in \mathcal{S}$. One time step later, the agent receives a numerical reward $r_{t+1} \in \mathcal{R}$ and find itself in a new state s_{t+1}.

At each time step, the agent implement a mapping from state to probabilities of selecting each possible action. Such a mapping is called the agent's **policy** and is noted π_t :

$$\pi_t(s, a) = \text{ probability of picking action } a \text{ in state } s_t \tag{1}$$

The key goal of reinforcement learning is to teach the agent how to change its policy as a result of a experience in order to maximize its reward over the long run.

▶ We seek the maximize the expected return R_t, which is a mapping from the reward sequence to $\mathbb{R}$. For instance :

$$R_t = \sum_{i=0}^{T} r_{t+1+i} \tag{2}$$

where T is the final step. That case is the particular one where each episode ends in a *terminal state*, which is followed by a standard reset state. We denote $\mathcal{S}^+ = \{s \in \ \mathcal{S} \,|\, s \text{ is terminal}\}$.

We can also face continuing tasks (i.e $T = +\infty$). In such case, the agents seek to maximize the *discounted return* :

$$R_t = \sum_{k=0}^{+\infty} \gamma^k r_{t+k+1} \tag{3}$$

where $\gamma < 1$ is the discounting rate.

2 Markov Decision Process

2.1 The Markov property

To make the choice on action tractable, it would need to rely only on the basis of the latest state signal, which should thus summarize past sensations compactly, retaining all relevant information.

As in many artificial intelligence algorithms, we ask for the state signal to be **Markov**, which would mean that the knowledge contained in a state is equivalent to to knowledge of the full history contained in the time series of the state signals.

Assuming a finite number number of state the Markov property writes (known as strong Markov property) for the state and reward signal :

$$\mathbb{P}\left(s_{t+1}=s',\, r_{t+1}=r \,|\, s_t, a_t, r_t, \ldots, s_1, a_1, r_1\right) = \mathbb{P}\left(s_{t+1}=s',\, r_{t+1}=r \,|\, s_t, a_t\right) \tag{4}$$

for all s', r and sequence $(s_t, a_t, r_t)_{t\in\mathbb{N}}$.

2.2 Reinforcement Learning under the Markov assumption

A reinforcement learning that satisfy the Markov property is called a **Markov Decision Process**. When the set of state and space is finite, we call it a finite MDP.

▶ A particular FMDP is defined by its state and action step and by the one-step dynamic of its environment :

- Transition probabilities :
$$\mathcal{P}^a_{ss'} = \mathbb{P}\left(s_{t+1}=s' \,|\, s_t = s, a_t = a\right) \tag{5}$$
- Expected value of the next reward :
$$\mathcal{R}^a_{ss'} = \mathbb{E}_\pi\left[r_{t+1} \,|\, s_t = s, a_t = a, s_{t+1} = s'\right] \tag{6}$$

2.2.1 The state value function

Almost all RL algorithms are based on estimating value functions. A value function is a function that maps the state space in $\mathbb{R}$, and estimates how good (defined in terms of future expected reward) it is for an agent to be at a given state.

Before going further, let us remind the policy definition :

$$\begin{aligned} \pi \,:\, (\mathcal{S}\times\mathcal{A}(S)) &\to [0,1] \\ (s,a) &\to \pi(s,a) \end{aligned} \tag{7}$$

where $\pi(s,a)$ is the probability of taking action a when $s_t = s$.

Hence the value of a state s under a policy π, noted $V^\pi(s)$ **is the expected return of $s \in \mathcal{S}$ when following the policy π.**

State-value function

$$\begin{aligned} \forall s \in \mathcal{S}, \quad V^\pi(s) &= \mathbb{E}_\pi\left[R_t \,|\, s_t = s\right] \\ &= \mathbb{E}_\pi\left[\sum_{k=0}^{+\infty} \gamma^k r_{t+k+1} \,|\, s_t = s\right] \end{aligned} \tag{8}$$

This function is also called the state-value function for the policy π.

2.2.2 The action value function

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, then following action a and then following the policy π. This function is called the action-value function for the policy π.

Action-value function

$$\begin{aligned}\forall (s,a) \in \mathcal{S} \times \mathcal{A}(s), \quad Q^\pi(s,a) &= \mathbb{E}_\pi \left[R_t \,|\, s_t = s, a_t = a \right] \\ &= \mathbb{E}_\pi \left[\sum_{k=0}^{+\infty} \gamma^k r_{t+k+1} \,|\, s_t = s, a_t = a \right]\end{aligned} \tag{9}$$

2.3 Bellman equations for MDP

Both those value functions satisfy particular recursive relationships. Indeed, for any policy and state, the following consistency holds between the value of the state and its successors :

Since :

$$\begin{aligned}V^\pi(s) &= \mathbb{E}_\pi \left[\sum_{k=0}^{+\infty} \gamma^k r_{t+k+1} \,|\, s_t = s \right] \\ &= \mathbb{E}_\pi \left[r_{t+1} + \gamma \sum_{k=0}^{+\infty} \gamma^k r_{t+k+2} \,|\, s_t = s \right]\end{aligned} \tag{10}$$

conditioning with respect to the possible action a :

$$\begin{aligned}V^\pi(s) &= \sum_{a \in \mathcal{A}(s)} \mathbb{E}_\pi \left[r_{t+1} + \gamma \sum_{k=0}^{+\infty} \gamma^k r_{t+k+2} \,|\, s_t = s, a_t = a \right] \mathbb{P}\left(a_t = a \,|\, s_t = s\right) \\ &= \sum_{a \in \mathcal{A}(s)} \sum_{s_{t+1} = s'} \mathbb{E}_\pi \left[r_{t+1} + \gamma \sum_{k=0}^{+\infty} \gamma^k r_{t+k+2} \,|\, s_t = s, a_t = a, s_{t+1} = s' \right] \mathbb{P}\left(a_t = a \,|\, s_t = s\right) \mathbb{P}\left(s_{t+1} = s' \,|\, s_t = s, a_t = a\right)\end{aligned} \tag{11}$$

and therefore :

$$V^\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(s,a) \sum_{s_{t+1} = s'} \mathcal{P}^a_{ss'} \left(\mathbb{E}_\pi \left[r_{t+1} + \gamma \sum_{k=0}^{+\infty} \gamma^k r_{t+k+2} \,|\, s_t = s, a_t = a, s_{t+1} = s' \right] \right) \tag{12}$$

Hence we obtain what is called the **Bellman equation for** V^π :

Bellman equation for V^π

$$\forall s \in \mathcal{S}, \quad V^\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V^\pi(s') \right) \tag{13}$$

2.3.1 Optimal value functions

Value functions define a partial ordering in the policies space. Indeed, we say that $\pi' \geq \pi$ if $\forall s \in \mathcal{S}$, $V^{\pi'}(s) \geq V^\pi(s)$. In a FMDP, there is always at least one policy that is better our equal to all other policies. We call this policy the optimal policy π^*.

All optimal policies share the same state-value function called the *optimal state value function* V^* :

$$\forall s \in \mathcal{S}, \quad V^*(s) = \max_\pi V^\pi(s) \tag{14}$$

and the same optimal action-value function :

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}(S), \quad Q^*(s,a) = \max_\pi Q^\pi(s,a) \tag{15}$$

There exists a straight-forward relation between those two optimal functions :

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}(S), \quad Q^*(s,a) = \mathbb{E}_\pi \left[r_{t+1} + \gamma V^*(s_{t+1}) \,|\, s_t = s, a_t = a \right] \tag{16}$$

Also, because V^* is the value function for a policy, it satisfies the self consistency given by the Bellman equation, giving rise to the **Bellman optimality equation**. Indeed, since $\forall s \in \mathcal{S}$:

$$\begin{aligned} V^*(s) &= \max_{a \in \mathcal{A}(s)} Q^*(s,a) \\ &= \max_a \mathbb{E}_\pi \left[R_t \,|\, s_t = s, a_t = a \right] \\ &= \max_a \sum_{s'} P^a_{ss'} \mathbb{E}_\pi \left[R_t \,|\, s_t = s, a_t = a, s_{t+1} = s' \right] \\ &= \max_{a \in \mathcal{A}(s)} P^a_{ss'} \mathbb{E}_\pi \left[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \,|\, s_t = s, a_t = a, s_{t+1} = s' \right] \end{aligned} \tag{17}$$

leading to the Bellman optimality equations :

Bellman optimality equations

Optimality equation for V^* :

$$\forall s \in \mathcal{S}, \quad V^*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s'} P^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V^*(s') \right) \tag{18}$$

Optimality equation for Q^* :

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}(s), \quad Q^*(s,a) = \sum_{s'} P^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma \max_{a' \in \mathcal{A}(s')} Q^*(s',a') \right) \tag{19}$$

For FMDP, the Bellman optimality equation has a unique solution, independant of the policy. Indeed, the optimal policy is the *greedy policy* (follows action with argmax action-value function) with respect to the optimal value function.

However, optimal policies (or equivalently optimal value functions) require a long time to compute, and extreme computational cost for large state spaces.

3 Quick summary

- The agent learns how to behave thanks to its interaction with the environment.
- Actions are choices made by the agent. States are basis for making this choices and rewards are the basis for evaluating them.
- A policy is a stochastic rule by which the agent selects actions as a function of its current state.
- The return is a function of future rewards the agents seeks to maximize over time.
- The Markov property is the basis for MDP. It indicates that the state signals compactly summarize the past without degrading the ability to predict the future.
- Even in a FMDP with complete knowledge of its environment, the memory load may be too heavy to store the value function : one may need to find a good, quick approximation of such a function.