Reinforcement Learning

Lidor Erez

June 2024

1 Introduction

Reinforcement learning is a straightforward approach to the problem of learning from interaction to achieve a specific goal. The learner and decision-maker are referred to as the *agent*, and the entity it interacts with is called the *environment*. At each time step, the agent selects an action, and the environment responds to this action. These responses present new situations to the agent, requiring it to choose subsequent actions accordingly. Additionally, the environment returns a reward that the agent aims to maximize over time. In summary, the agent chooses an action in a given state of the environment, which then provides a reward and transitions to a new state. The agent's objective is to respond with actions that maximize the cumulative rewards over time.

2 Agent-Environment Interface

At each time step $t = 0, 1, 2, 3, \ldots$, the agent receives a state $s_t \in S$, where S is the set of possible states. The agent then selects an action $a_t \in A(s_t)$, where $A(s_t)$ is the set of actions available to the agent in state s_t. Before transitioning to state s_{t+1}, the agent receives a reward $r_{t+1} \in \mathcal{R}$ for the action taken in state s_t. To select an action when in a given state, the agent uses a mapping function π_t, which is called a policy. For a given state s_t, π_t returns the probabilities of selecting each action in $A(s_t)$. Thus, $\pi_t(s, a)$ is the probability that $a_t = a$ if $s_t = s$. Reinforcement learning methods determine how the agent changes its policy as a result of its experience.

3 Goals and Rewards

The agent acquire special rewards from the environment when taking an action. At each time step, the reward is simply a number $r_t \in \mathcal{R}$ and the goal of an agent is to maximize the total amount of reward it receives over time. The use of reward signal from the environment is what differentiate reinforcement learning from other machine learning problems such as supervised learning. Although, the rewarding system might seem to be limited it has proved to be working in

various of tasks such as teaching robots to walk, escaping from a maze and even collecting empty soda cans for recycling. It's crucial to reward an agent only when it achieves what we want it to. Namely, one shouldn't reward an agent for achieving sub-goals that leads to the final goal as this might make the agent to learn how to achieve these sub-goals rather than achieving the main goal we attend it to. In summary, we use to rewards to communicate the agent what we want it to achieve rather than tell him how to achieve it.

4 Returns

In the aforementioned sections, we have said that the goal of an agent is to maximize the cumulative reward in the long run. Consider the sequence of rewards that has been received after time step t, denoted as $r_{t+1}, r_{t+2}, \ldots$. We seek to maximize the expected return, which is defined as a function.

4.1 Episodic Tasks

For episodic tasks, the return R_t can be defined as the sum of rewards:

$$R_t = r_{t+1} + r_{t+2} + \ldots + r_T$$

where T is the final time step. This approach is suitable for applications where the agent-environment interaction can naturally be divided into subsequences. Such as: plays of a game or trips through a maze. Each episode ends in a special state called the terminal state, followed by a reset to a standard starting state. We call this type of task *episodic tasks*. In episodic tasks, it is sometimes crucial to distinguish between the set of all non-terminal states S and the set S^+ of all states (including the terminal state).

4.2 Continuous Tasks

In continuous tasks, the agent-environment interaction goes on continually. In this case, the final time step would be $T = \infty$, which can lead to an infinite return. To address this, we introduce a discount factor γ, where $0 \leq \gamma \leq 1$. The discount factor is used to decrease the value of rewards that might come in the far future, ensuring that the sum of rewards remains finite. Although, it can be finite if $\gamma < 1$ and $\{r_k\}$ is bounded. The return R_t can be defined as follows:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

if $\gamma = 0$ then the agent is concerned about maximizing immediate rewards. Namely, he chooses a_t to maximize only r_{t+1}. On the other hand, as γ approaches to 1 the agent is becoming more farsighted.

4.3 The Markov Property

The Markov Property is essential for defining Markov decision processes (MDPs), which serve as the foundational framework for modeling decision-making problems. The Markov Property asserts that the future state of an environment is conditionally independent of past states and actions, given the present state and action. To keep the mathematics simple, we will assume that there are a finite number of states and reward values so we'll work only with sums and probabilities rather than integrals and probability densities. Although, it can easily be extended to include continuous states and rewards. Consider the response of the environment at time step $t+1$ to the action taken at time step t. In general, this can be defined as:

$$Pr[s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0]$$

However, if the state signal has the *Markov property* then the environment's response at $t+1$ this former probability can be defined as:

$$Pr[s_{t+1} = s', r_{t+1} = r | s_t, a_t] \quad \forall\ s', r, s_t, a_t$$

In other words, a state s has the Markov property and is a Markov state if and only if the aforementioned probabilities equal for all s', r, and histories, $s_t, a_t, r_t, \ldots, r_1, s_0, a_0$ When this happens we say that the environment have the Markov property.

4.4 Markov Decision Processes

When a reinforcement learning task satisfies the Markov property defined in the previous section, we call it a *Markov decision process* (MDP). If the state and action spaces are finite, then it is called a *finite Markov decision process* (finite MDP). The latter is defined by its state and action sets and by the one-step dynamics of the environment. Namely, given an action a and state s, the probability of every possible next state s' is:

$$\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$$

These probabilities are called *transition probabilities*. The expected reward can also be calculated using the Markov property as follows:

$$\mathcal{R}^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}$$

4.5 Value Functions

The value function is a way to estimate how good it is for the agent to be in a given state. The term "how good" relates to future rewards that can be expected (expected return). The value function is defined with respect to a policy. Recall

that a policy is a mapping from each state $s \in S$ and action $a \in A(s)$ to the probability $\pi(s, a)$ of taking a in s. Thus, the expected return is defined as:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\}$$

where V^{π} is the state-value function for policy π. In similar means, we define the value of taking a in s under a policy π as:

$$Q^{\pi}(s, a) = E\{R_t | s_t = s, a_t = a\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

Where Q^{π} is the action-value function for policy π.

The value of both Q^{π} and V^{π} can be estimated from experience. For example, one can use Monte Carlo methods (averaging over random samples) to estimate the values of the functions.

4.6 Bellman Equation

For any policy π and any state s, the following consistency condition holds between the value of s and the value of its possible successor states:

$$V^{\pi}(s) = \sum_{a} \pi(s, a) \sum_{s'} \mathcal{P}^{a}_{ss'} [\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')]$$

The Bellman equation, averages over all the possibilities, weighting each by it's probability of occurring. It states that the value of the start state equal to the expected value of the next state, plus the reward expected along the way.

4.6.1 Optimal Value Function

In a reinforcement learning task, the objective is to find a policy that maximizes the cumulative reward over the long run. For finite Markov Decision Processes (MDPs), we can define and determine such a policy. We say that a policy π is better than another policy π' if its expected return is greater than or equal to that of π' for all states. Formally, $\pi \geq \pi'$ if and only if $V^{\pi}(s) \geq V^{\pi'}(s)$ for all $s \in S$. There may be more than one optimal policy, and we denote these as π^*. These optimal policies share the same state-value function, called the *optimal state-value function*, defined as:

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

Furthermore, these optimal policies, also share an *optimal action value function* defined as:

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$$

For the state-action pair (s, a), this function gives the expected return for taking a in s, thus Q^* can be written in terms of V^* as follows:

$$Q^*(s, a) = E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\}$$

To find these optimal value functions, one needs to solve a system of non-linear equations, one for each state (i.e., N states result in N equations). When the number of states is small (e.g., $N = 2$) and $\mathcal{R}^a_{ss'}$, $\mathcal{P}^a_{ss'}$ are known, it is feasible to solve this system of equations. However, as the number of states increases, solving this system becomes impractical due to several assumptions that must be satisfied: (1) We know $\mathcal{R}^a_{ss'}$, $\mathcal{P}^a_{ss'}$; (2) We have sufficient computational resources; (3) The Markov property holds. Consequently, this method of solving the Bellman equation is rarely used in practice.

References

[1] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.