# Reinforcement Learning Notes

## The Agent-Environment Interface

Markov Decision Process is solving the problem of learning from interaction to achieve a goal. The learner and decision maker is called an **agent**. An **agent** interacts with its **environment**. These two entities interact continuously – the agent selecting an action and the environment and the environment responding to these actions presenting new situations to the agent. The environment gives rise to **rewards**, special numerical values that the agent seeks to maximize over time through its choice of values.

A diagram of a agent and environment

Description automatically generated with low confidence

Figure: the agent-environment interaction in a Markov decision process

More specifically, the agent and environment interact at each of a sequence of discrete time steps,

Notes: 1) In *Optimal Control Theory* the terms **controller**, **controlled system**, and **control signal** are used instead of agent, environment and action.

2) We are going to restrict our attention to discrete time steps to keep the analysis as simple as possible. The framework can be extended to continuous time at the expense of analytical simplicity (see *Bertsekas, Reinforcement Learning and Optimal Control*).

At each time step the agent receives a representation of the environment’s **state** and on this basis selects an **action**, . One time step later, in part of consequence of its action, the agent receives a numerical **reward**, , and finds itself in a new state, . The MDP and the agent thereby give rise to a sequence or **trajectory** that begins like this:

In a finite MDP, the sets of states, actions, and rewards () all have finite number of elements. In this case, the random variables and have well defined probability distributions dependent only on the preceding state and action. That is, for particular values of these random variables, and , there is a probability of the values occurring at time t, given particular values of the preceding state and action:

(1)

for all and . The function defines the **dynamics** of the MDP. The **dynamics** function is an ordinary deterministic function of four arguments which specifies probability distribution for each choice of and , that is:

, for all , (2)

In a Markov decision process, the probabilities given by completely characterize the environment’s dynamics. That is, the probability of each possible value for and depends on the immediately preceding state and action, and , and given them, not at all on earlier states and actions. This is best viewed as a restriction not on the decision process, but on the *state*. 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**.

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 :

(3)

We can compute the expected rewards for state-action pairs as a two-argument function ,

(4)

and the expected rewards for state-action-next-state triples as a three-argument function ,

(5)

### Example: Recycling Robot

A mobile robot has the job of collecting empty soda cans in an office environment. It has sensors for detecting cans, and an arm and gripper that can pick them up and place them in an onboard bin; it runs on a rechargeable battery. The robot’s control system has components for interpreting sensory information, for navigating, and for controlling the arm and gripper. High-level decisions about how to search for cans are made by a reinforcement learning agent based on the current charge level of the battery. The robot’s control system has components for interpreting sensory information, for navigating, and for controlling the arm and gripper. High-level decisions about how to search for cans are made by a reinforcement learning agent based on the current charge level of the battery. To make a simple example, we assume that only two charge levels can be distinguished, comprising a small state set . In each state, the agent can decide whether to (1) actively for a can for a certain period, (2) remain stationary and for someone to bring it a can, or (3) head back to its home base to its battery. When the energy level is , recharging would always be foolish, so we do not include it in the action set for this state. The action sets are then and .

The rewards are zero most of the time but become positive when the robot secures an empty can, or large and negative if the battery runs all the way down. The best way to find cans is to actively search for them, but this runs the robot’s battery down, whereas waiting does not. Whenever the robot is searching, the possibility exists that its battery will become depleted.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| low | search | low |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Figure: for the Recycling Robot example in tabular form

## Goals and Rewards

In reinforcement learning, the purpose or goal of the agent is formalized in terms of a special signal, called the **reward**, passing from the environment to the agent. At each time step, the reward is a real number, . 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. We can state this idea as the *reward hypothesis*:

The notion of goal and purposes can be thought of as the maximization of the expected value of the cumulative sum of a received scalar signal called **reward**. The goal is the agent to learn to maximize its reward. If we want it to do something for us, we must provide rewards to in a such way that in maximizing them the agent will also achieve our goals. Thus, it is 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. Instead, the reward signal should communicate to the agent only *what* we want to achieve.

## Returns and Episodes

So far we have discussed informally the objective of learning. We have said that the agent’s goal is to maximize the cumulative reward it receives in the long run. If the sequence of rewards received after step is denoted with then what precise aspect of this sequence do we wish to maximize?

In general, we seek to maximize the ***expected return***, where the return, denoted , is defined as some specific function of the reward sequence. In the simplest case the return is the sum of the rewards:

(6)

where is a final time step. This approach makes sense in applications in which there is a natural notion of final time step, that is, when the agent-environment interaction breaks naturally into subsequences, which we call ***episodes***, such as plays of a game, trips through a maze or any sort of repeated interaction. Each episode ends in a special state called the ***terminal state***, followed by a reset to a standard starting state or to a sample from a standard distribution of starting states. The episodes can be considered to end in the same terminal state, with different rewards from the different outcomes. Tasks with episodes of this kind are called ***episodic tasks***.

In episodic tasks we sometimes need to distinguish the set of all nonterminal states, denoted , from the set of all states plus the terminal state, denoted . The time of termination, , is a random variable that normally varies from episode to episode.

On the other hand, in many cases the agent-environment interaction does not break naturally into identifiable episodes, but goes on continually without limit. We call these continuing tasks. The return formulation (6) is problematic for continuing tasks because the final step would be , and the return, which is what we are trying to maximize, could easily be infinite. We need an additional concept which will allows us to model continuing tasks - ***discounting***. According to this concept the agent tries to select actions so that the sum of the discounted rewards it receives over the future is maximized. In particular, it chooses to maximize the expected ***discounted return***:

(7)

where is a parameter, , called the ***discount rate***.

The discount rate determines the present value of future rewards: a reward received time steps in the future is worth only times what it would be worth if it were received immediately. If , the infinite sum in (7) has a finite value as long as the reward sequence is bounded. If , the agent is “myopic” in being concerned only with maximizing the immediate rewards: its objective in this case id to learn how to choose so as to maximize only . If each of the agent’s actions happened to influence only the immediate reward, not future rewards as well, then a myopic agent could maximize (7) by separately maximizing each immediate reward. But in general acting to maximize the immediate reward can reduce access to future rewards so that the return is reduced. As approaches 1, the return objective takes future rewards into account more strongly; the agent becomes more farsighted.

Returns at successive time steps are related to each other in a way that is important for the theory and algorithms of reinforcement learning:

(8)

Note that this works for all time steps , even if termination occurs at , provided we define . Note that although the return (7) is a sum of infinite number of terms, it is still finite if the reward is nonzero and constant – if . For example, if the reward is a constant , then the return is

(9)

## Policies and Value Functions

Almost all reinforcement learning algorithms involve estimating **value functions** – functions of state (or 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). The notion of “how good” is defined in terms of future rewards that can be expected, or to be precise, in terms of expected return. Of course, the reward 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**.

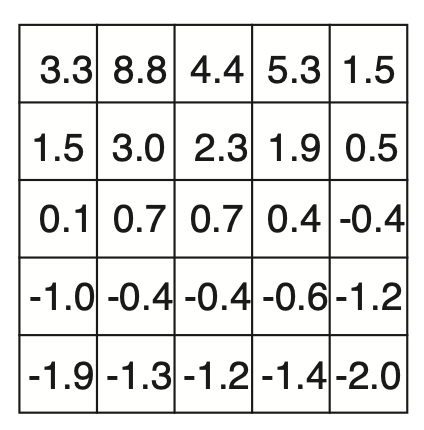
Formally, a **policy** is a mapping from states to probabilities of selecting each possible action. If an agent is following policy at time , then is the probability that if . Like , is an ordinary function which defines a probability distribution.

***Gridworld***

Gridworld is an example illustrating the use of a finite Markov Decision Process.

The cells of the grid correspond to the states of the environment. At each cell, four actions are possible: ***north***, ***south***, ***east***, and ***west***, which deterministically cause the agent to move one cell in the respective direction on the grid. Actions that would take the agent off the grid leave its location unchanged, but also result in a reward of -1.

Other actions result in a reward of 0, except those that move the agent out of the special states and . From state , all four actions yield a reward of +10 and take the agent to . From state B



A picture containing symbol, font, white, graphics

Description automatically generatedA picture containing diagram, line

Description automatically generated

//TODO: finish the discussion on Markov decision processes

# Bibliography

Berteskas, D. P. (2019). *Reinforcement Learning and Optimal Control.* Belmont, Massachusetts: Athena Scientific.

Richard S. Sutton, A. G. (2020). *Reinforcement Learning: An Introduction.* Cambridge, Massachusetts: The MIT Press.