Lecture 2: The Reinforcement Learning Setups

Melih Kandemir

Semester: Fall 2021

1 The ingredients

Reinforcement learning is built on the following concepts. An **agent** is anything that the system has power to change arbitrarily, for instance the valid moves in chess. An **environment** is anything on which the agent has limited or no control, such as joint angles of a robot, engine horsepower in autonomous driving. All information about the situation of the environment and the agent at time point t is stored in a variable $S_t \in \mathcal{S}$, called the **state**. The set of all possible states an environment can be in $\mathcal{S}$ is called a **state space**. The agent fulfills its target task by choosing an **action** $A_t \in \mathcal{A}$. The environment change caused by an action affects the agent via a *reward*, denoted as R_t.

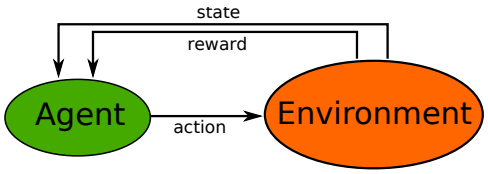


Figure 1: Reinforcement learning solves a task defined by rewards by interacting with an environment and updating its policy with respect to experience.

A **policy** is the agent's behavior function that maps states to actions. The central problem of reinforcement learning is to find a policy that maximizes reward. A **deterministic policy** is a function that always gives the same output for a fixed input, i.e. $A_t = \pi(S_t)$. A **stochastic policy** applies a random mapping on a given input, i.e. $\pi(A_t|S_t) = P(A_t|S_t)$. The policy is our central object of interest in reinforcement learning. It is the end output of the training process. We can use a policy for i) guessing the future state of the environment, called **prediction**, and ii) learning how to change the environment for a target task, called **control**.

The precise boundary between the agent and environment varies in applications. For instance, the limbs of a robot are the agent if we care only about high-level planning, and the environment if we are to control them with RL. The primary concern of RL is to design goals by rewards (i.e. to describe what the goal is, not how to achieve it). An **environment model** is the agent's internal representation of the real world it is living in.

We are interested in reinforcement learning to solve real-world problems, called **tasks**. A task is said to be **episodic** if state sequences break naturally such as a chess game, and **continuous** if never ends. For example, see M episodes from an episodic task

$$
\begin{aligned}
&S_1, A_1, R_2, S_2, A_2, R_3, S_3, A_3, R_4, S_4 = s_e \text{ (Episode 1)}\\
&S_1, A_1, R_2, S_2, A_2, R_3, S_3, A_3, R_4, S_4, A_4, R_5, S_5 = s_e \text{ (Episode 2)}\\
&\qquad\qquad\vdots\\
&S_1, A_1, R_2, S_2, A_2, R_3, S_3 = s_e \text{ (Episode M)}
\end{aligned}
$$

each of which ends when the terminal state s_e is reached at different time points.

2 The reinforcement learning setup

Reinforcement learning is in fact an ill-posed problem, where the agent aims to learn a task within an environment that it changes while learning. Breaking this cross-dependency requires applying heuristic. The

most important heuristic in reinforcement learning is to find the optimal balance between exploration and exploitation. **Exploration** is probing the environment randomly to gain information, such as try a new restaurant. **Exploitation** is using existing knowledge to maximize the reward, such as going once more to our favorite restaurant.

We need assumptions to simplify the problem even further to make it solvable. The first is to assume that the states follows the Markov property, which is defined as below.

Definition 2.1: Markov Property

A random state variable S_t is said to satisfy the Markov property if

$$P(S_{t+1}|S_t) = P(S_{t+1}|S_1, \cdots, S_t).$$

Intuitively, the Markov property assumes that the future is independent of the past given a *good enough description* of the present. This does not mean we do not care about the past, but only means we can encapsulate sufficient information about the past into a single variable *some* careful definition of state. If this is possible, we will gain enormous efficiency in information storage and our problem will simplify dramatically.

We aim to model environments that may change for arbitrarily long time. There is no time limit on the duration of a tennis game. Our time budget to swing up a cart pole could also be chosen arbitrarily. Hence, we need a mathematical definition that glues the variables of arbitrarily many time steps together.

Definition 2.2: Markov Process

A Markov Process (MP) is a tuple of two entities $\langle \mathcal{S}, \mathcal{P} \rangle$, where

i) $\mathcal{S}$ is the set of environment states: $S_t = s$ with $s \in \mathcal{S}, \ \forall t$.

ii) $\mathcal{P} = P(S_{t+1}|S_t)$ is the environment dynamics model.

Let us now incorporate the reward into our setup.

Definition 2.3: Markov Reward Process

A Markov Reward Process (MRP) is a tuple of four entities $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where

i) $\mathcal{S}$ is the set of environment states such that $S_t = s$ with $s \in \mathcal{S}, \ \forall t$,

ii) $\mathcal{R}$ is the set of rewards such that $R_t = r$ with $r \in \mathcal{R}$,

iii) $\gamma \in [0, 1]$ is the discount factor,

iv) $\mathcal{P} = P(R_{t+1}, S_{t+1}|S_t)$ is the environment model that decomposes as

$$P(R_{t+1}, S_{t+1}|S_t) = \underbrace{P(R_{t+1}|S_{t+1}, S_t)}_{Reward\ model} \underbrace{P(S_{t+1}|S_t)}_{Transition\ model}.$$

Now let us take a closer look at the relationship between the random variables of an episode. Take the episode S_1, R_2, S_2, R_3, S_3 and decompose its joint distribution following the product rule

$$\begin{aligned}
P(S_1, R_2, S_2, R_3, S_3) &= P(S_3, R_3, S_2, R_2|S_1)P(S_1) \\
&= P(S_3, R_3, S_2|R_2, S_1)P(R_2|S_1)P(S_1) \\
&= P(S_3, R_3|S_2, R_2, S_1)P(S_2|R_2, S_1)P(R_2|S_1)P(S_1) \\
&= \underbrace{P(S_3|R_3, S_2, R_2, S_1)}_{P(S_3|S_2)} \underbrace{P(R_3|S_2, R_2, S_1)}_{P(R_3|S_2)} \underbrace{P(S_2|R_2, S_1)}_{P(S_2|S_1)} P(R_2|S_1)P(S_1). \\
&= P(S_3|S_2)P(R_3|S_2)P(S_2|S_1)P(R_2|S_1)P(S_1),
\end{aligned}$$

where red denotes the Markov property and blue the assumption of the Markov Reward Process on the reward model.

Let us next construct a quantity of interest that measures the success of the agent in solving the task.

Definition 2.4: Return

Return is defined as the cumulative discounted reward starting from time step t on

$$G_t \triangleq R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}.$$

In this formulation, $\gamma^k R_{t+k+1}$ represents the *present* value of the *future* reward R_{t+k+1}. Setting $\gamma = 0$ gives a myopic return model and $\gamma = 1$ a far-sighted return model. Reward is typically discounted to account for two sources of uncertainty: i) the environment model is probabilistic, where uncertainty propagates through time, ii) the policy is unreliable at the early stages of training. Return has a recursive property:

$$\begin{aligned} G_t &\triangleq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma \underbrace{\left[R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right]}_{G_{t+1}} \\ &= R_{t+1} + \gamma G_{t+1}, \end{aligned}$$

making the divide-and-conquer strategy applicable to RL.

Let us next construct a mechanism into which the agent can use to aggregate its observations to sharpen its heuristic search.

Definition 2.5: Value Functions

A **state-value** function (a.k.a. value function) is the expected return of starting a from state and following a policy

$$v(s) = \mathbb{E}[G_t | S_t = s]$$

and an **action-value** function is the expected return of starting from state s, taking action a, and following policy π

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a].$$

Intuitively, the state-value function gives a measure of how good it is to be in a state, and the action-value function gives a measure of how good it is to take a certain action in a certain state. These two functions are connected. Integrating out the action variable in the action-value function with respect to the policy distribution gives the state-value function

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(A_t = a | S_t = s) q_\pi(s, a).$$

The expectation needs to be taken over all random variables

$$S_1, S_2, \cdots, R_1, R_2, \cdots$$

In more detail,

$$v(s) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots].$$

Definition 2.6: Markov Decision Process

A Markov Decision Process (MDP) is defined as a tuple of five entities $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and a policy function π, where

- $\mathcal{S}$ is the set of environment states such that $S_t = s$ with $s \in \mathcal{S}, \ \forall t$,
- $\mathcal{R}$ is the set of rewards such that $R_t = r$ with $r \in \mathcal{R}$,
- $\gamma \in [0, 1]$ is the discount factor,
- $\mathcal{A}$ is the set of actions such that $A_t \in \mathcal{A}$ follows $A_t \sim \pi(A_t|S_t)$,
- $\mathcal{P} = P(R_{t+1}, S_{t+1}|S_t, A_t)$ is the environment model that decomposes as

$$P(R_{t+1}, S_{t+1}|S_t, A_t) = \underbrace{P(R_{t+1}|S_t, A_t)}_{Reward\ model} \underbrace{P(S_{t+1}|S_t, A_t)}_{Transition\ model}.$$

Essentially, MDP can be viewed as an MRP. We will typically have an MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ with an attached policy π. Integrating out all actions with respect to the policy distribution gives an environment model

$$\mathcal{P}_\pi \triangleq P_\pi(S_{t+1} = s'|S_t = s) = \sum_{a \in \mathcal{A}} \pi(A_t = a|S_t = s) P(S_{t+1} = s'|S_t = s, A_t = a),$$

and a reward distribution

$$\mathcal{R}_\pi = P_\pi(R_{t+1}|S_t = s) = \sum_{a \in \mathcal{A}} \pi(A_t = a|S_t = s) P(R_{t+1}|S_t = s, A_t = a)$$

that are functionals of the policy π, determined by the policy in less technical terms. With these ingredients, we can construct a MRP $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$ for a given policy π.

There are cases when the environment cannot be accurately observed. In chess the environment is fully observed, while the physical conditions of the sky when flying a helicopter. Such cases can be handled by a further extension of MDPs with an observation model.

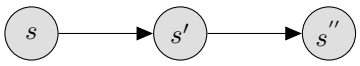
Definition 2.7: Partially Observable Markov Decision Process

A Partially Observable Markov Decision Process (POMDP) is defined as a tuple of six entities $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{O}, \mathcal{R}, \gamma \rangle$ and a policy function π, where

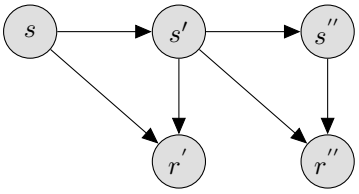
- $\mathcal{S}$ is the set of environment states such that $S_t = s$ with $s \in \mathcal{S}, \ \forall t$,
- $\mathcal{R}$ is the set of rewards such that $R_t = r$ with $r \in \mathcal{R}$,
- $\gamma \in [0, 1]$ is the discount factor,
- $\mathcal{A}$ is the set of actions such that $A_t \in \mathcal{A}$ follows $A_t \sim \pi(A_t|S_t)$,
- $\mathcal{O}$ is the set of observations such that $O_t = o$ with $o \in \mathcal{O}, \ \forall o$,
- $\gamma \in [0, 1]$ is the discount factor,
- $\mathcal{P} = P(R_{t+1}, O_{t+1}, S_{t+1}|S_t, A_t)$ is the environment model that decomposes as

$$P(R_{t+1}, O_{t+1}, S_{t+1}|S_t, A_t) = \underbrace{P(R_{t+1}|S_t, A_t)}_{Reward\ model} \underbrace{P(O_{t+1}|S_{t+1})}_{Observation\ model} \underbrace{P(S_{t+1}|S_t, A_t)}_{Transition\ model}.$$

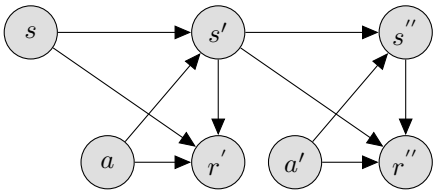
Figure 2 summarizes the introduced reinforcement learning setups as plate diagrams.



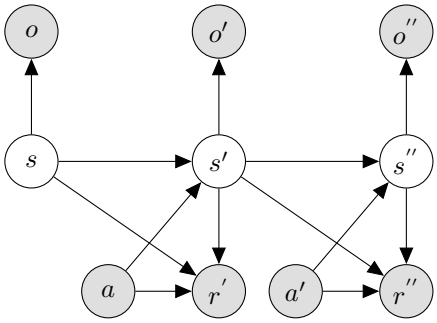
(a) Markov Process



(b) Markov Reward Process



(c) Markov Decision Process



(d) Partially Observable Markov Decision Process

Figure 2: Reinforcement Learning Setups