

Chapter 1 Introduction

of Reinforcement Learning: An Introduction

Xinyan Wang

School of Statistics, East China Normal University

October 9, 2020

- 1 What is Reinforcement Learning (RL)?
- 2 687-Gridworld: A Simple Environment
- 3 Describing the Agent and Environment Mathematically
- 4 Additional Terminology, Notation, and Assumptions

What is Reinforcement Learning (RL)?

Reinforcement Learning (RL)

Reinforcement learning is an area of machine learning, inspired by behaviorist psychology, concerned with how an agent can learn from interactions with an environment.-Wikipedia, Sutton and Barto (1998), Phil



What is Reinforcement Learning (RL)?

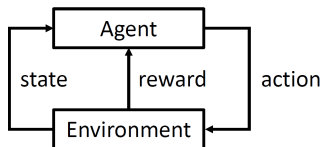


Figure 1: Agent-environment diagram.

Reinforcement Learning (RL)?

- Agent: Child, dog, robot, program, etc.
- Environment: World, lab, software environment, etc.
- Evaluative Feedback: Rewards convey how "good" an agent's actions are, not what the best actions would have been (supervised learning). If the agent was given instructive feedback (what action it should have taken) this would be a supervised learning problem, not a reinforcement learning problem.

What is Reinforcement Learning (RL)?

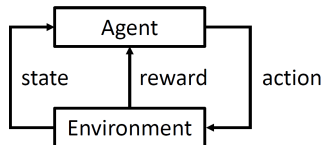


Figure 1: Agent-environment diagram.

Two characteristics

- Trial-and-error search
- Delayed reward

What is Reinforcement Learning (RL)?

Idea

Learning from interaction.

- How an agent can learn from interactions with an environment to maximize a numerical reward signal.
- A computational approach (Markov Decision Process, MDP).

687-Gridworld: A Simple Environment



Start State 1	State 2	State 3	State 4	State 5
State 6		State 8	State 9	State 10
State 11	State 12	Obstacle	State 13	State 14
State 15	State 16	Obstacle	State 17	State 18
State 19	State 20		State 22	Goal State 23

Figure 2: 687-Gridworld, a simple example environment we will reference often.

State: Position of robot.

687-Gridworld: A Simple Environment



Start State 1	State 2	State 3	State 4	State 5
State 6		State 8	State 9	State 10
State 11	State 12	Obstacle	State 13	State 14
State 15	State 16	Obstacle	State 17	State 18
State 19	State 20		State 22	Goal State 23

Figure 2: 687-Gridworld, a simple example environment we will reference often.

Actions: Attempt Up, Attempt Down, Attempt Left, Attempt Right. We abbreviate these as: AU, AD, AL, AR.

687-Gridworld: A Simple Environment



Start State 1	State 2	State 3	State 4	State 5
State 6		State 8	State 9	State 10
State 11	State 12	Obstacle	State 13	State 14
State 15	State 16	Obstacle	State 17	State 18
State 19	State 20		State 22	Goal State 23

Figure 2: 687-Gridworld, a simple example environment we will reference often.

Environment Dynamics: With probability 0.8 the robot moves in the specified direction. With probability 0.05 it gets confused and veers to the right-moves $+90^\circ$ from where it attempted to move (that is, AU results in the robot moving right, AL results in the robot moving up, etc.). With probability 0.05 it gets confused and veers to the left-moves 90° from where it attempted to move (that is, AU results in the robot moving left, AL results in the robot moving down, etc.). With probability 0.1 the robot temporarily breaks and does not move at all. If the movement defined by these dynamics would cause the agent to exit the grid (e.g., move up from state 2) or hit an obstacle (e.g., move right from state 12), then the agent does not move. The robot starts in state 1, and the process ends when the robot reaches state 23.

687-Gridworld: A Simple Environment



Start State 1	State 2	State 3	State 4	State 5
State 6		State 8	State 9	State 10
State 11	State 12	Obstacle	State 13	State 14
State 15	State 16	Obstacle	State 17	State 18
State 19	State 20		State 22	Goal State 23

Figure 2: 687-Gridworld, a simple example environment we will reference often.

Rewards: The agent receives a reward of -10 for entering the state with the water and a reward of $+10$ for entering the goal state. Entering any other state results in a reward of zero. If the agent is in the state with the water (state 21) and stays in state 21 for any reason (hitting a wall, temporarily breaking), it counts as "entering" the water state again and results in an additional reward of -10 . We use a reward discount parameter (the purpose of which is described later) of $\gamma = 0.9$.

Describing the Agent and Environment Mathematically

Mathematical model

Markov decision processes (MDPs).

Note

A common misconception is that RL is only about MDPs. This is not the case: MDPs are just one way of formalizing the environment of an RL problem.

Describing the Agent and Environment Mathematically

- Let $t \in N_{\geq 0}$ be the *time step* (iteration of the agent-environment loop).
- Let S_t be the state of the environment at time t .
- Let A_t be the action taken by the agent at time t .
- Let $R_t \in \mathbb{R}$ be the reward received by the agent at time t . That is, when the state of the environment is S_t , the agent takes action A_t , and the environment transitions to state S_{t+1} , the agent receives the reward R_t . This differs from some other sources where in this reward is called R_{t+1} .

Formally, a finite MDP is a tuple, $(S, A, P, d_R, d_0, \gamma)$, where:

- S is the set of all possible states of the environment. The state at time t , S_t , always takes values in S . For now we will assume that $|S| < \infty$ - that the set of states is finite.
- A is the set of all possible actions the agent can take. The action at time t , A_t , always takes values in A . For now we will assume that $|A| < \infty$.
- P is called the **transition function**, and it describes how the state of the environment changes.

$$P : S \times A \times S \rightarrow [0, 1]. \quad (1)$$

Describing the Agent and Environment Mathematically

For all $s \in S, a \in A, s' \in S$, and $t \in N_{\geq 0}$:

$$P(s, a, s') := \Pr(S_{t+1} = s' | S_t = s, A_t = a). \quad (2)$$

- d_R describes how rewards are generated. Intuitively, it is a conditional distribution over R_t given S_t, A_t , and S_{t+1} . For now we assume that the rewards are bounded - that $|R_j| \leq R_{\max}$ always, for all $t \in N_{\geq 0}$ and some constant $R_{\max} \in \mathbb{R}$.
- R is a function called the *reward function*, which is implicitly defined by d_R . Other sources often define an MDP to contain R rather than d_R . Formally

$$R : S \times A \rightarrow \mathbb{R}, \quad (3)$$

and

$$R(s, a) := E[R_t | S_t = s, A_t = a], \quad (4)$$

for all s, a , and t . Although the reward function, R , does not precisely define how the rewards, R_t , are generated (and thus a definition of an MDP with R in place of d_R would in a way be incomplete), it is often all that is necessary to reason about how an agent should act.

- d_0 is the **initial state distribution**:

$$d_0 : S \rightarrow [0, 1], \quad (5)$$

and for all s :

$$d_0(s) = \Pr(S_0 = s). \quad (6)$$

- $\gamma \in [0, 1]$ is a parameter called the *reward discount parameter*, and which we discuss later.

Describing the Agent and Environment Mathematically

Just as we have defined the environment mathematically, we now define the agent mathematically. A policy is a decision rule - a way that the agent can select actions. Formally, a policy, π , is a function:

$$\pi : S \times A \rightarrow [0, 1], \quad (7)$$

and for all $s \in S$, $a \in A$, and $t \in N_{\geq 0}$,

$$\pi(s, a) := Pr(A_t = a | S_t = s). \quad (8)$$

Thus, a policy is the conditional distribution over actions given the state. That is, π is not a distribution, but a collection of distributions over the action set - one per state. There are an infinite number of possible policies, but a finite number of **deterministic policies** (policies for which $\pi(s; a) \in \{0, 1\}$ for all s and a). We denote the set of all policies by Π . Figure 3 presents an example of a policy for 687-Gridworld. We will sometimes write $\pi : S \rightarrow A$ to denote a deterministic policy, where $A_t = \pi(S_t)$.

Describing the Agent and Environment Mathematically

	AU	AD	AL	AR
1	0	0.1	0.3	0.6
2	0.8	0	0	0.2
3	0.1	0.1	0.5	0.3
4	0.25	0.25	0.25	0.25
5	0.25	0.25	0.5	0
6	0.2	0.3	0.5	0
...

Figure 3: Example of a tabular policy. Each cell denotes the probability of the action (specified by the column) in each state (specified by the row). In this format, Π is the set of all $|\mathcal{S}| \times |\mathcal{A}|$ matrices with non-negative entries and rows that all sum to one.

To summarize so far, the interaction between the agent and environment proceeds as follows (where $R_t \sim d_R(S_t, A_t, S_{t+1}, \cdot)$ denotes that R_t is sampled according to d_R):

$$S_0 \sim d_0 \tag{9}$$

$$A_0 \sim \pi(S_0, \cdot) \tag{10}$$

$$S_1 \sim P(S_0, A_0, \cdot) \tag{11}$$

$$R_0 \sim d_R(S_0, A_0, S_1, \cdot) \tag{12}$$

$$A_1 \sim \pi(S_1, \cdot) \tag{13}$$

$$S_2 \sim P(S_1, A_1, \cdot) \tag{14}$$

$$\dots \tag{15}$$

Describing the Agent and Environment Mathematically

In pseudocode:

Algorithm 1: General flow of agent-environment interaction.

```
1  $S_0 \sim d_0$ ;  
2 for  $t = 0$  to  $\infty$  do  
3    $A_t \sim \pi(S_t, \cdot)$ ;  
4    $S_{t+1} \sim P(S_t, A_t, \cdot)$ ;  
5    $R_t \sim d_R(S_t, A_t, S_{t+1}, \cdot)$ ;
```

The running of an MDP is also presented as a Bayesian network in Figure 4.

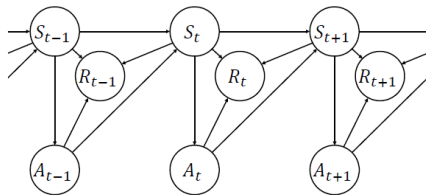


Figure 4: Bayesian network depicted the running of an MDP.

Describing the Agent and Environment Mathematically

Agent's goal: Find a policy, π^* , called an **optimal policy**. Intuitively, an optimal policy maximizes the expected total amount of reward that the agent will obtain.

Objective function: $J : \Pi \rightarrow \mathbb{R}$, where for all $\pi \in \Pi$,

$$J\pi := E \left[\sum_{t=0}^{\infty} R_t \middle| \pi \right] \quad (16)$$

Note: Later we will revise this definition - if you are skimming looking for the correct definition of J , it is in (18).

Optimal Policy: An optimal policy, π^* , is any policy that satisfies:

$$\pi^* \in \arg \max_{\pi \in \Pi} J(\pi) \quad (17)$$

Note: Much later we will define an optimal policy in a different and more strict way.

Property 1 (Existence of an optimal policy)

If $|S| < \infty$, $|A| < \infty$, $R_{max} < \infty$, and $\gamma < 1$, then an optimal policy exists.

Describing the Agent and Environment Mathematically

Reward Discounting: If you could have one cookie today or two cookies on the last day of class, which would you pick? Many people pick one cookie today when actually presented with these options. This suggests that rewards that are obtained in the distant future are worth less to us than rewards in the near future. The reward discount parameter, γ , allows us to encode, within the objective function, this discounting of rewards based on how distant in the future they occur.

Recall that $\gamma \in [0,1]$. We redefine the objective function, J , as:

$$J(\pi) := E \left[\sum_{t=0}^{\infty} \gamma^t R_t \middle| \pi \right] \quad (18)$$

for all $\pi \in \Pi$. So, $\gamma < 1$ means that rewards that occur later are worth less to the agent - the utility of a reward, r , t time steps in the future is $\gamma^t r$. Including γ also ensures that $J(\pi)$ is bounded, and later we will see that smaller values of γ make the MDP easier to solve (**solving** an MDP refers to finding or approximating an optimal policy).

To summarize, the agent's goal is to find (or approximate) an optimal policy, π^* , as defined in (17), using the definition of J that includes reward discounting - (18).

Describing the Agent and Environment Mathematically

When we introduced 687-Gridworld, we said that the agent-environment interactions terminate when the agent reaches state 23, which we called the goal. This notion of a **terminal state** can be encoded using our definition of an MDP above. Specifically, we define a **terminal state** to be any state that **always** transitions to a special state, s_∞ , called the **terminal absorbing state**. Once in s_∞ , the agent can never leave (s_∞ is **absorbing**) - the agent will forever continue to transition from s_∞ back into s_∞ . Transitioning from s_∞ to s_∞ always results in a reward of zero. Effectively, when the agent enters a terminal state the process ends. There are no more decisions to make (since all actions have the same outcome) or rewards to collect. Thus, an episode **terminates** when the agent enters s_∞ . Notice that terminal states are optional - MDPs need not have any terminal states. Also, there may be states that only sometimes transition to s_∞ , and we do not call these terminal states. Notice also that s_∞ is an element of S . Lastly, although terminal states are defined, **goal states** are **not** defined - the notion of a goal in 687-Gridworld is simply for our own intuition.

When the agent reaches s_∞ , the current trial, called an **episode** ends and a new one begins. This means that t is reset to zero, the initial state, S_0 , is sampled from d_0 , and the next episode begins (the agent selects A_0 , gets reward R_0 , and transitions to state s_∞). The agent is noticed that this has occurred, since this reset may change its behavior (e.g., it might clear some sort of short-term memory).

Describing the Agent and Environment Mathematically

Consider again the definition of reinforcement learning. Notice the segment "learn from **interactions** with the environment." If P and R (or d_R) are known, then the agent does not need to interact with the environment. E.g., an agent solving 687-Gridworld can plan in its head, work out an optimal policy and execute this optimal policy from this start. This is **not reinforcement** learning - this is **planning**. More concretely, in planning problems P and R are known, while in reinforcement learning problems at least P (and usually R) is not known by the agent. Instead, the agent must learn by interacting with the environment- taking different actions and seeing what happens. Most reinforcement learning algorithms will **not** estimate P . The environment is often too complex to model well, and small errors in an estimate of P compound over multiple time steps making plans built from estimates of P unreliable. We will discuss this more later.

Additional Terminology, Notation, and Assumptions

- A **history** , H_t , is a recording of what has happened up to time t in an episode:

$$H_t := (S_0, A_0, R_0, S_1, A_1, R_1, S_2, \dots, S_t, A_t, R_t). \quad (19)$$

- A **trajectory** is the history of an entire episode: H_∞ .
- The **return or discounted return** of a trajectory is the discounted sum of rewards: $G := \sum_{t=0}^{\infty} \gamma^t R_t$. So, the objective, J , is the **expected return or expected discounted return**, and can be written as $J(\pi) := E[G|\pi]$.
- **The return from time t or discounted return from time t** , G_t , is the discounted sum of rewards starting from time t :

$$G_t := \sum_{k=0}^{\infty} \gamma^k R_{t+k}$$

Markov Property

A seemingly more general non-Markovian formulation for the transition function might be:

$$P(h, s, a, s') := \Pr(S_{t+1} = s' | H_{t-1} = h, S_t = s, A_t = a). \quad (22)$$

The **Markov assumption** is the assumption that S_{t+1} is conditionally independent of H_{t-1} given S_t . That is, for all h, s, a, s', t

$$\Pr(S_{t+1} = s' | H_{t-1} = h, S_t = s, A_t = a) = \Pr(S_{t+1} = s' | S_t = s, A_t = a) \quad (23)$$

Since we make this Markov assumption, P as defined earlier completely captures the transition dynamics of the environment, and there is no need for the alternate definition in (22). The Markov assumption is sometimes referred to as the **Markov property** (for example one would usually say that a domain has the Markov property, not that the domain satisfies the Markov assumption). It can also be stated colloquially as: the future is independent of the past given the present.

We also assume that the rewards are Markovian - R_t is conditionally independent of H_{t-1} given S_t (since A_t depends only on S_t , this is equivalent to assuming that R_t is conditionally independent of H_{t-1} given both S_t and A_t). While the previous Markov assumptions apply to the environment (and are inherent assumptions in the MDP formulation of the environment), we make an additional Markov assumption about the agent: the agent's policy is Markovian. That is, A_t is conditionally independent of H_{t-1} given S_t .

Stationary vs. Nonstationary

We assume that the dynamics of the environment are **stationary**. This means that the dynamics of the environment do not change between episodes, and also that the transition function does not change within episodes. That is, $Pr(S_0 = s)$ is the same for all episodes, and also for all s, a, t , and i :

$$Pr(S_{t+1} = s' | S_t = s, A_t = a) = Pr(S_{i+1} = s_0 | S_i = s, A_i = a). \quad (24)$$

Importantly, here t and i can be time steps from different episodes.

This is one of the assumptions that is most often **not** true for real problems.

We also assume that the rewards are stationary (that the distribution over rewards that result from taking action a in state s and transitioning to state s' does not depend on the time step or episode number).

However, usually we assume that π is **nonstationary**. This is because learning corresponds to changing (ideally, improving) the policy both within an episode and across episodes. A stationary policy is sometimes called **a fixed policy**.

The **horizon**, L , of an MDP is the smallest integer such that

$$\forall t \geq L, Pr(S_t = s_\infty) = 1. \quad (54)$$

If $L < \infty$ for all policies, then we say that the MDP is **finite horizon**. If $L = \infty$ then the domain may be **indefinite horizon or infinite horizon**. An MDP with **indefinite horizon** is one for which $L = \infty$, but where the agent will always enter s_∞ . An **infinite horizon** MDP is an MDP where the agent may never enter s_∞ .

Summary

- ➊ Reinforcement learning is a computational approach to understanding and automating goal-directed learning and decision making.
- ➋ It is distinguished from other computational approaches by its emphasis on learning by an agent from direct interaction with its environment, without requiring exemplary supervision or complete models of the environment.
- ➌ Reinforcement learning uses the formal framework of Markov decision processes to define the interaction between a learning agent and its environment in terms of states, actions, and rewards.

Thanks you!