Reinforcement Learning Notes

Faraaz Akhtar

October 2024

1 Introduction

Reinforcement learning (RL) is a type of machine learning in which an agent learns to make decisions by interacting with an environment. The agent takes actions in the environment to maximize some notion of cumulative reward over time. Unlike supervised learning, where the model is trained on a fixed dataset, in RL the agent learns from the consequences of its actions through a process of trial and error.

2 How does reinforcement learning work?

At each step t the agent:

- Receives observation O_t (and reward R_t)
- Executes action A_t

and the environment:

- Receives action A_t
- Emits observation O_{t+1} (and reward R_{t+1})

The agent's job is to maximize cumulative reward

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots$$

Reinforcement learning is based on the hypothesis that any goal can be formalized as the outcome of maximizing a cumulative reward.

We call the expected cumulative reward, from a state s, the value

$$v(s) = E\left[G_t \mid S_t = s\right]$$

$$= E\left[R_{t+1} + R_{t+2} + R_{t+3} + \dots \mid S_t = s\right]$$

Returns and values can be defined recursively

$$G_t = R_{t+1} + G_{t+1}$$

$$v(s) = E\left[R_{t+1} + v(S_{t+1}) \mid S_t = s\right]$$

The environment state is the environment's internal state and is usually invisible to the agent. Even if it is visible, it may contain lots of irrelevant information.

The history is the full sequence of observations, actions, rewards

$$\mathcal{H}_t = O_0, A_0, R_1, O_1, \ldots, O_{t-1}, A_{t-1}, R_t, O_t$$

(This could, for example, include the sensorimeter stream of a robot.)

This history is used to construct the agent state S_t. The agent state could be the environment state if the agent has full observability. i.e., $S_t = O_t =$ environment state.

3 Markov Decision Processes (MDPs)

A decision process is Markov if

$$p(r, s \mid S_t, A_t) = p(r, s \mid \mathcal{H}_t, A_t)$$

This means that the future is independent of the past given the present state. In other words, the probability of transitioning to a new state s and receiving a reward r depends only on the current state S_t and the action A_t, and not on the full history $\mathcal{H}_t$.

This simplifies the decision-making process because instead of needing to keep track of the entire history of states and actions, the agent only needs to know the current state and current action to make predictions about the future. This assumption is what makes Markov Decision Processes (MDPs) computationally feasible and widely used in reinforcement learning.

The fact that a state is Markov doesn't mean it contains everything, only that adding more history doesn't help in predicting the future. Once the state is known, the history can effectively be discarded. The full environment and agent state together form a Markov process, though it may be large. Similarly, the full history is also Markov, but it keeps growing over time. Typically, the agent state is a compressed version of the full history.

Often, it's not possible to construct a fully Markovian agent state. In such cases, it may be more useful to focus on the policies and value prediction.

4 Policy

A policy defines the agent's behavior by mapping the agent's state to an action. It can either be deterministic, where the action A is a specific function of the state S (i.e., $A = \pi(S)$), or stochastic, where the policy defines a probability distribution over actions given the state, $\pi(A|S) = p(A|S)$.

5 Value Functions

The actual value function is the expected return:

$$v_\pi(s) = E[G_t \mid S_t = s, \pi] = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s, \pi]$$

In the above equation, the discount factor, $\gamma \in [0, 1]$, trades off importance of immediate vs long-term rewards.This can also be used to evaluate the desirability of states.

The return has a recursive form:

$$G_t = R_{t+1} + \gamma G_{t+1}$$

And therefore, the value function has a recursive form as well:

$$v_\pi(s) = E\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t \sim \pi(s)\right]$$
$$= E\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t \sim \pi(s)\right]$$

Where, $A_t \sim \pi(s)$ means action a is chosen by policy π in state s, even if π is deterministic. This recursive equation is known as the Bellman equation.

A similar equation holds for the optimal value function, $v_*(s)$, which gives the highest possible value from any state s:

$$v_*(s) = \max_a E[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

This equation does not depend on a specific policy because it always chooses the action a that maximizes the expected return. This means the agent selects the optimal action for each state, regardless of a fixed policy. The equation is fundamental in algorithms, where agents aim to learn the optimal value function and select actions that maximize reward over time.

While learning the optimal value function is powerful, learning a policy directly can be more practical because it provides an explicit mapping from states to actions. Policies handle stochastic environments, balance exploration vs. exploitation, and can be more efficient in high-dimensional or continuous action spaces.

6 Model

A model predicts what the environment will do next.

- $\mathscr{P}$ predicts the next state:

$$\mathscr{P}(s, a, s') \approx p(S_{t+1} = s' \mid S_t = s, A_t = a)$$

- $\mathscr{R}$ predicts the next (immediate) reward:

$$\mathscr{R}(s, a) \approx E[R_{t+1} \mid S_t = s, A_t = a]$$

A model does not directly give us a good policy — we still need to plan based on the model. Additionally, we can also consider stochastic (generative) models, which introduce randomness in the model predictions.