Reinforcement learning

Linus Falk

March 27, 2023

Contents

1 Introduction

Machine learning can typically be separated into three branches:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Reinforcement learning is different from the two other branches in the way that it is not supervised with labeled data, but it doesn't find patterns in data completely on its own as in unsupervised learning. Instead there is a reward signal, and the feedback can be delayed, so time come in as a factor as well. The actions of the so called **agent** affect the subsequent data it receives.

The agent-environment interface

In our reinforcement learning setup we have the **agent** which is the decision maker and the **environment**, which can be seen as everything else in the setup. The goal is to maximize the cumulative reward. The agents task is to learn this from experience.

The agent observes the environments state and takes some sort of action, the agent then receives a rewards from the action and the environments state changes. The agent must now observe the new state and take a new action, and so on. Note: in reinforcement learning the environment, agent and/or reward may be stochastic.

Probability theory - review

Lets consider two random variables $X \in \chi$ and $Y \in \gamma$

- Probability: the probability that we will observe $x \in \chi$ is written as:

$$\Pr\{X = x\} \tag{1}$$

- Conditional probability: If we have already observed $y \in \gamma$ then the probability that we will observe $x \in \chi$ is written:

$$\Pr\{X = x|Y = y\} \tag{2}$$

- Probability functions:

$$\mathrm{p(x)} = \Pr\{X = x\}, \mathrm{p}(x|y) = \Pr\{X = x|Y = y\} \tag{3}$$

- Probabilities sum to 1:

$$\sum_{x \in \chi} p(x) = 1 \text{ and } \sum_{x \in \chi} p(x|y) = 1 \tag{4}$$

- The expected value:

$$\mathrm{E}[X] = \sum_{x \in \chi} xp(x), E[X|Y = y] = \sum_{x \in \chi} xp(x|y) \tag{5}$$

What is a state?

The state S_t is a representation of the environment at time t. The state space $\mathcal{S}$ is a set of all possible states. S_t will be dependent on what happened in the past, before time t. S_t contains all information that is relevant for the prediction of the future at time t.

The Markov property

$$p(S_{t+1}|S_0, A_0, S_1, A_1, \ldots, S_t, A_t) = p(S_{t+1}|S_t, A_t) \tag{6}$$

Example: The taxi environment

- Taxi: 25 different positions possible
- Passenger: 5 positions including picked up
- Destinations: 4 options
- In total 25 x 5 x 4 = 500 configurations

We can enumerate all possible configurations in some way, let $\mathcal{S} = \{0, 1, \dots, 499\}$. This is a finite state space.

Example: Inverted pendulum

When balancing a stick, the control signal or action is the torque at the bottom. The state?

- $S_t = \theta_t$

No! we don't have any information about the direction the stick is moving in.

- $S_t = \begin{bmatrix} \theta_t \\ \frac{d\theta_t}{dt} \end{bmatrix} \in \mathcal{S} \subset \mathbb{R}^2$

This is an example of a continuous state space, Infinitely many possible states.

What is the reward

The reward R_t is a scalar signal that tells how it is doing at that time step t. The goal is the maximize the cumulative reward over **long-time**.

In the taxi example could we for example see a reward of -10 if a illegal pick-up or drop-off occurred, a successful drop-off would score +20 and all other actions would score -1 in reward. So in other words, to maximize the total reward we should deliver passenger in as few steps as possible.

In the inverted pendulum example could the goal be to keep the angle as close to zero as possible, perhaps: $R_{t+1} = -\theta_t^2$ and if we also want to use a low torque we could try: $R_{t+1} = -c_1\theta_t^2 - c_2 a_t^2$. With $\theta_t = 0$ and $a_t = 0$ we would get the maximum reward $R_{t+1} = 0$

Sequential decision making

The goal is to select actions that maximize the future reward. So actions may have long term consequences and reward may be delayed. It can also be the case that sacrificing immediate reward can gain more long-term reward. Examples:

- A financial investment,
- Fueling a car, prevent fuel stop later

Designing the reward functions

In this course we will often receive the reward functions. However it is important to note that the design of the reward function is important since it will determine what the agent tries to achieve. In some cases the agent might also find unintended ways to increase the reward and the reward influences **how** the agent learns.

an example would be the mountain car with the goal of reaching the top/the flag with as few steps as possible. The reward is -1 for each step until the flag is reached. This is a sparse rewards, all action looks equally bad until the flag is reached the first time. It could be possible to use a more informative reward function, but it is important to make sure that the agent still optimize the correct thing we want.

What is an action

An action A_t is the way an agent can affect the state a time t and $\mathcal{A}$ is the set of all possible actions. In the taxi example would an action be for example: go north, go west and pick-up and in the pendulum example could it be: torque.

Stochastic environments

A deterministic environment is one which the outcome of an action is determined by the current state and the actions taken. While in the stochastic environment there is some randomness and the outcome of an action is not completely determined by the current state. This means that the same action in the same state might have different outcomes at different times. This means that it is harder for the agent to learn an optimal policy.

The power of feedback

If no feedback is given we can pre-compute all actions $A_0, A_1, ...$ if the first state S_0 is given. While with feedback we decide the action A_t after we have observed the state at time t, S_t. Example could be to walk with or without blindfold. So instead of trying to find good actions we want to try to find a good policy.

What is a policy?

Policy is a distribution over actions given states:

$$\pi(a|s) = \Pr\{A_t = a|S_t = s\} \tag{7}$$

A policy defines how the agent will behave in different states. If we have a deterministic policy we can sometimes write:

$$a = \pi(s) \tag{8}$$

Example: The linear quadratic regulator

$$S_t = FS_t + GA_t + W_t \tag{9}$$

If we want to maximize the expected value

$$E\left[\sum_{t=0}^{\infty}(-c_1 S_t^T S_t - c_2 A_t^T A_t)\right] \tag{10}$$

Then the **Optimal policy:** when we observe s_t, choose the action:

$$a_t = \pi(s_t) = -Ls_t \tag{11}$$

for some fixed L, that we can find if F and G are know.

Terminology

Reinforcement learning	**Optimal control**
Environment	System / plant
State, S_t	State, x(t)
Action, A_t	Input, u(t)
Reward	Cost, c(t)
Policy	Controller
Maximize reward	Minimize cost
Learn policy from experience	Use model to find controller (LQC or MPC)

Table 1: example

Exploration vs exploitation

The agent should be able to learn a good policy without losing to much reward. **Exploration** is used to learn more about the environment and **exploitation** is the use of information to maximize the reward. We usually have both explore and exploit.

Model vs model-free

In model based reinforcement learning we learn a model from experience and uses the model to find a good policy and/or predictions. While in model-free learning we learn a policy and/or predictions without first learning the model. In this course the main focus will be on model-free reinforcement learning.

b

2 Markov Decision process

This lecture will explain the concepts of Markov decision process, MDP's, the Bellman equation and optimal policy

Recap

For a state to have **Markov property** means that the state contains all the information that is useful to predict the future. In other words, we don't need the previous states to predict the future, the useful information is stored in the current state. A **Policy**: π(a|s) choosing an action a when we are in state s, can be deterministic or stochastic. The **prediction** gives the future cumulative reward following a policy. We want to find the policy that maximize the cumulative future reward by **control**.

Markov decision process

If we begin with assuming that $\mathcal{S}, \mathcal{A}$ and $\mathcal{R}$ have finite numbers of elements then the **translation probabilities** are given by:

$$p(', r|s, a) = \Pr\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\} \tag{12}$$

The Markov property determines the **dynamics** of the environment, the probability of transitioning to a new state depends only on the current state and action. The **state transitions**:

$$p(s'|s, a) = \sum_{r \in \mathcal{R}} p(s', r|s, a) = \tag{13}$$

With the **expected reward**

$$r(s, a) = \mathbb{E}[R_{r+1}|S_t = s, A_t = a] = \sum_{r \in \mathcal{R}} \sum_{s' \in \mathcal{S}} rp(s', r|s, a) \tag{14}$$

Episodic vs continuing tasks

- **Episodic tasks**
  - Has terminating states and the task end in finite time.
  - When reaching the terminating state the episode stops.
  - If you reach the terminating state you will stay there forever, receiving no future rewards.
- **Continuing tasks**
  - Often not a clear way to divide the task into independent episodes.
  - No state were the task is done
  - Must take into account infinitely many future rewards.

The return

In a given state we want to maximize the future reward we can receive: $R_{t+1}, R_{t+2,...}$. To make it possible to have non finite number of rewards we introduce the **discounted reward**

The discounted reward

$$G_t = R_{t+1}R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \text{where } 0 < \gamma \leq 1 \tag{15}$$

since we put $\gamma \leq 1$ we put less value on future rewards, $\gamma = 0.5 \Rightarrow \gamma^{10} = 0.001, \gamma = 0.9 \Rightarrow \gamma^{10} = 0.35$.

And if $\gamma < 1$ we make sure that $R_t < \overline{R}$ for all t, and then G_t is bounded:

$$\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \leq \overline{R} \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma} \overline{R} \tag{16}$$

If we know that the task ends after a finite number of steps we can get away with using non-discounted returns where $\gamma = 1$

The state value function

The state-value function estimates the expected long term rewards the agent will recieve starting from a specific state and following a given policy. Since S_t and R_t are random variables, the return is therefore also a random variable:

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

We must therefore consider the expected return:

The state-value function

The state-value function $v_\pi(s)$ of a MDP is the expected return starting from the state s and then following the policy π:

$$v_\pi(s) = \mathbb{E}[G_t | S_t = s] \tag{17}$$

The prediction of cumulative reward is computed with $v_\pi(s)$

The action-value function

Another important value function is the **action-value function** that estimates the expected long-term reward an agent will receive starting from a specific state, taking a specific action and following a given policy.

The action-value function

The action-value function $q_\pi(s, a)$ is the exptected return starting from s, taking action a, and **then** following a policy π

$$q_\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a] \tag{18}$$

This function is also often called the Q-function.

Bellman equations

Looking at the reward we can note that:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = R_{t+1} + \gamma G_{t+1} \tag{19}$$

Hence, the value function satisfies to following equation:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \end{aligned} \tag{20}$$

The value of s is the expected immediate reward plus the discounted expected vlue of the next state. In the same way is action-value function:

$$q_\pi(s,a) = \mathbb{E}[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}|S_t = s, A_t = a)] \tag{21}$$

$$\boxed{\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] \\ q_\pi(s,a) &= \mathbb{E}[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}|S_t = s, A_t = a)] \end{aligned} \quad (22)}$$

The expectation

The state-value of a state s, is the expected action-value:

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a) \tag{23}$$

So for a deterministic policy a $= \pi(s)$ we get $v_\pi(s, \pi(s))$. And given s and a, the immediate reward r and the next state s' has probability $p(s', r|s, a)$ so that:

$$q_\pi(s,a) = \sum_{r,s'} p(s', r|s, a)(r + \gamma v_\pi(s')) \tag{24}$$

The Bellman equation for v_π and q_π

$$\begin{aligned} v_\pi &= \sum_a \pi(a|s) q_\pi(s,a) = \sum_a \pi(a,s) \sum_{r,s'} p(s', r|s, a)[r + \gamma v_\pi(s')] \\ q_\pi(s,a) &= \sum_{r,s'} p(s', r|s, a)[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a')] \end{aligned} \tag{25}$$

Solving the Bellman equation

Solving the Bellman equation is done by solving a system of linear equation, this is because the dependencies between the different states. Each state $s \in \mathcal{S}$ gives one equation. There is a unique solution that can be expressed analytically. If there is a large number of states, large $\mathcal{S}$, then there are more efficient ways to solve it with iterative solution. If $p(s', r|s, a)$ is not know, we need to **learn** $v_\pi(s)$ from **experience**. If the number of states $\mathcal{S}$ is infinite, we can't compute the value for each state individually, and instead we need to find some function $\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$.

Optimal value function

This represent the maximum expected cumulative reward that can be achieved from a given state, following the best possible policy. It quantifies the highest long-term reward that the agent can expect to obtain when makin **optimal decision** from each state. The optimal state-value function is given by:

$$v_*(s) = \max v_\pi(s), \text{for all } s \in \mathcal{S} \tag{26}$$

The optimal action-value function is given by:

$$q_*(s,a) = \max q_\pi(s,a), \text{for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A} \tag{27}$$

The optimal $v_*(s)$ should be the maximum of $q_*(s,a)$, so we have:

$$v_*(s) = \max_a q_*(s,a) \tag{28}$$

The equation for $q_*(s,a)$:

$$
\begin{aligned}
q_*(s,a) = \sum_{r,s'} p(s',r|s,a)(r+\gamma v_*(s')) &= \max_a \mathbb{E}[R_{t+1} + \gamma(S_{t+1})|S_t = s, A_t = a] \\
&= \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1},a')|S_t = s, A_t = a]
\end{aligned} \tag{29}
$$

Solving the Bellman optimality function

$$
v_*(s) = \max_a \sum_{r,s'} p(s',s|s,a)[r+\gamma v_*(s')] \tag{30}
$$

This is a system of non-linear equation (non linear because of the max function). There is one equation for each state, s. In general there is no closed form solution. But there are iterative methods to solve it.

Optimal policy

Partial ordering over policies: This is the relationship between different policies, where some of them can be ranked as better or worse than others. This is based on their performance or expected returns. For some policy pairs there is no such definitive ranking. This means that the ordering captures the hierarchy of policies in terms of effectiveness but not necessarily a linear ranking of them.

$$
\pi \geq \pi' \text{ if } v_\pi(s) \geq t_{\pi'}(s), \text{ for all } s \tag{31}
$$

Theorem

- There exists at least one optimal policy π_* such that $\pi_* \geq \pi$ for all policies π.
- All optimal policies achieve the optimal state-value function $v_{\pi *}(s) = v_*(s)$
- All optimal policies achieve the action-value function $q_{\pi *}(s,a) = q_*(s,a)$

So how do make a decision in state s?

1. Choose an action, a that maximizes the optimal action-value $q_*(s,a)$
2. Then use an optimal policy form s'

The **control** part is to find this optimal policy. Were the policy is described with:

$$
\pi_*(s) = \arg\max_a \sum_{r,s'} p(s',r|s,a)[r+\gamma v_*(s')] \tag{32}
$$

which is optimal