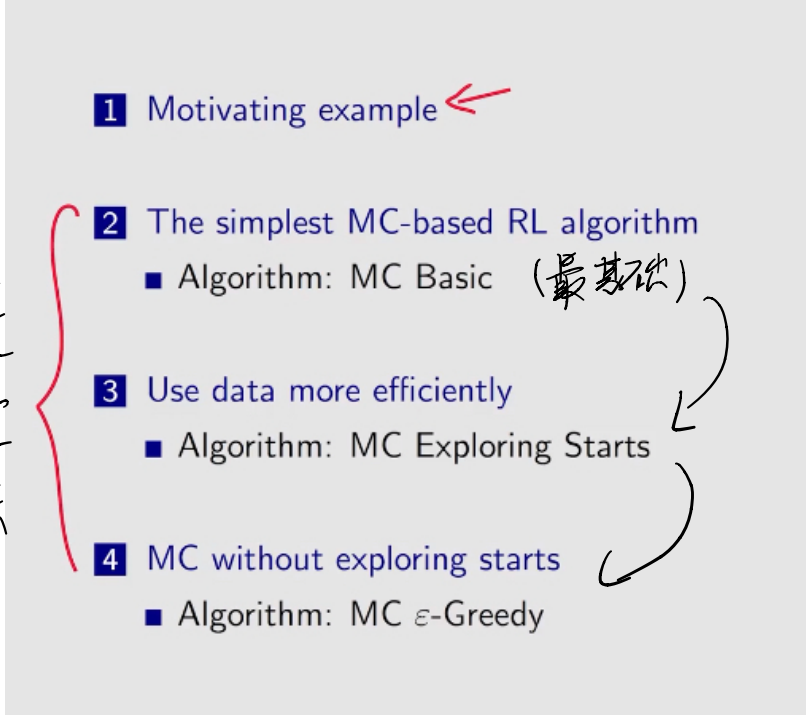
Chapter 5: Monte Carlo Learning (model-free)

MBRL：数据估计出一个模型.

1. Outline:



1. Motivating example:

Monte Carlo estimation:

How can we estimate sth without models?

The simplest idea: Monte Carlo estimation.

Example: Flip a coin;

The result (either head or tail) is denoted as a random variable X

- if the result is head, then $X = +1$ ✓
- if the result is tail, then $X = -1$ ✓

The *aim* is to compute $\mathbb{E}[X]$.

Method 1: Model-based.

Suppose the probabilistic model is known as:

$$p(X=1) = 0.5 \quad p(X=-1) = 0.5$$

Then by definition:

$$E[X] = \sum_x x\,p(x) = 1 \times 0.5 + (-1) \times 0.5 = 0$$

problem: it may be impossible to know the precise distribution

Method 2 : Model - Free.

Idea : Flip coin many times, calculate the average of the outcome

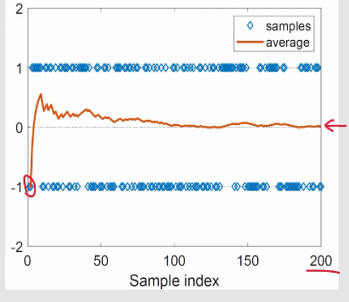
- Suppose we get a sample sequence: $\{x_1, x_2, \ldots, x_N\}$.
  Then, the mean can be approximated as

$$\mathbb{E}[X] \approx \bar{x} = \frac{1}{N}\sum_{j=1}^{N} x_j.$$

This is the idea of Monte Carlo estimation!

▷ **Question:** Is the Monte Carlo estimation accurate?

- When N is *small*, the approximation is inaccurate.
- As N *increases*, the approximation becomes more and more accurate.



大数定律.

Law of Large Numbers

For a random variable X. Suppose $\{x_j\}_{j=1}^N$ are some iid samples. Let $\bar{x} = \frac{1}{N}\sum_{j=1}^{N} x_j$ be the average of the samples. Then,

$$\mathbb{E}[\bar{x}] = \mathbb{E}[X],$$
$$\mathrm{Var}[\bar{x}] = \frac{1}{N}\mathrm{Var}[X]. \to 0 \quad \bar{x} \to$$

As a result, $\bar{x}$ is an unbiased estimate of $\mathbb{E}[X]$ and its variance decreases to zero as N increases to infinity.

▷ The samples must be iid (independent and identically distributed)

▷ For the proof, see the book.

2. The simplest MC-based RL algorithm

Convert policy iteration algorithm to the model free.

Policy iteration

$$\begin{cases} \text{Policy evaluation} : v_{\pi_k} = r_{\pi_k} + \gamma P_{\pi_k} v_{\pi_k} \\ \text{Policy improvement} : \pi_{k+1} = \arg\max_{\pi} (r_{\pi} + \gamma P_{\pi} v_{\pi_k}) \end{cases}$$

The elementwise form:

$$\pi_{k+1}(s) = \arg\max_{\pi} \sum_a \pi(a|s) \left[\sum_r p(r|s,a) r + \gamma \sum_{s'} p(s'|s,a) v_{\pi_k}(s') \right]$$

$$= \arg\max_{\pi} \sum_a \pi(a|s) q_{\pi_k}(s,a) \qquad s \in S$$

The key is q_{π_k}

① requires the model:

$$q_{\pi_k}(s,a) = \sum_r p(r|s,a) r + \gamma \sum_{s'} p(s'|s,a) v_{\pi_k}(s')$$

② not requires the model:

$$q_{\pi_k}(s,a) = E[G_t | S_t = s, A_t = a]$$

Idea to achieve model-free RL: We can use expression 2 to calculate $q_{\pi_k}(s,a)$ based on data (sample or experience)

The Monte Carlo estimation of action value:

① Starting from (s,a) following policy π_k, generate an episode

② The return of this episode is $g(s,a)$

③ $g(s,a)$ is a sample of G_t in

$$q_{\pi_k}(s,a) = \mathbb{E}[G_t \mid S_t = s, A_t = a]$$

④ Suppose we have a set of episodes and $\{g^{(i)}(s,a)\}$

Then $q_{\pi_k}(s,a) = \mathbb{E}[G_t \mid S_t = s, A_t = a] \approx \frac{1}{N}\sum_{i=1}^{N} g^{(i)}(s,a)$

Fundamental idea: When model is unavailable, we can use data. { sample experience.

▷ Description of the algorithm:

Given an initial policy π_0, there are two steps at the kth iteration.

- **Step 1: policy evaluation.** This step is to obtain $q_{\pi_k}(s,a)$ for all (s,a). Specifically, for each action-state pair (s,a), run an infinite number of (or sufficiently many) episodes. The average of their returns is used to approximate $q_{\pi_k}(s,a)$.
- **Step 2: policy improvement.** This step is to solve $\pi_{k+1}(s) = \arg\max_\pi \sum_a \pi(a|s) q_{\pi_k}(s,a)$ for all $s \in \mathcal{S}$. The greedy optimal policy is $\pi_{k+1}(a_k^*|s) = 1$ where $a_k^* = \arg\max_a q_{\pi_k}(s,a)$.

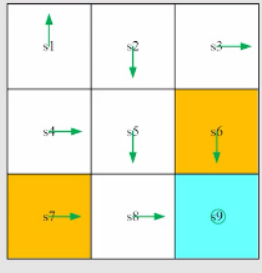
Exactly the same as the policy iteration algorithm, except

- Estimate $q_{\pi_k}(s,a)$ directly, instead of solving $v_{\pi_k}(s)$.

MC Basic is a variant of policy iteration algorith

- MC Basic is a variant of the policy iteration algorithm.
- The model-free algorithms are built up based on model-based ones. It is, therefore, necessary to understand model-based algorithms first before studying model-free algorithms.
- MC Basic is useful to reveal the core idea of MC-based model-free RL, but not practical due to low efficiency.
- Why does MC Basic estimate *action values* instead of state values? That is because state values cannot be used to improve policies directly. When models are not available, we should directly estimate action values.
- Since policy iteration is convergent, the *convergence* of MC Basic is also guaranteed to be convergent given sufficient episodes.

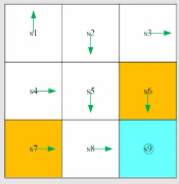
Illustrative example:



π_0

Task:

- An initial policy is shown in the figure.
- Use MC Basic to find the optimal policy.
- $r_{\text{boundary}} = -1$, $r_{\text{forbidden}} = -1$, $r_{\text{target}} = 1$, $\gamma = 0.9$.



Outline: given the current policy π_k

- Step 1 - policy evaluation: calculate $q_{\pi_k}(s, a)$
  How many state-action pairs? 9 states $\times$ 5 actions =45 state-action pairs!
- Step 2 - policy improvement: select the greedy action
  $a^*(s) = \arg\max_{a_i} q_{\pi_k}(s, a)$

3. Use data more efficiently

Algorithm: MC Exploring Starts

The MC Basic algorithm:

- **Advantage**: reveal the core idea clearly!
- **Disadvantage**: too simple to be practical.

However, MC Basic can be extended to be more efficient.

episode such as

$$s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_4} s_4 \xrightarrow{a_5} s_5 \xrightarrow{a_1} \cdots$$

① Visit: a state-action pair appear in the episode

② Method use the data: Initial-visit method.

③ Just calculate the return and approximate $q_\pi(s_1, a_2)$

▷ The episode also visits other state-action pairs.

$$\begin{aligned}
s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_4} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \cdots \quad & \text{[original episode]} \\
s_2 \xrightarrow{a_4} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \cdots \quad & \text{[episode starting from } (s_2, a_4)\text{]} \\
s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \cdots \quad & \text{[episode starting from } (s_1, a_2)\text{]} \\
s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \cdots \quad & \text{[episode starting from } (s_2, a_3)\text{]} \\
s_5 \xrightarrow{a_1} \cdots \quad & \text{[episode starting from } (s_5, a_1)\text{]}
\end{aligned}$$

Can estimate $q_\pi(s_1, a_2)$, $q_\pi(s_2, a_4)$, $q_\pi(s_2, a_3)$, $q_\pi(s_5, a_1)$,...

Data-efficient methods:

- first-visit method
- every-visit method

update the policy more efficiently

▷ Another aspect in MC-based RL is when to update the policy. There are two methods.

- **The first method** is, in the policy evaluation step, to collect all the episodes starting from a state-action pair and then use the average return to approximate the action value.
  - This is the one adopted by the MC Basic algorithm.
  - The problem of this method is that the agent has to wait until all episodes have been collected.
- **The second method** uses the return of a single episode to approximate the action value.
  - In this way, we can improve the policy episode-by-episode.

Generalized policy iteration (GPI)

- Not a specific algorithm.
- It refers to the general idea or framework of switching between policy-evaluation and policy-improvement processes.
- Many model-based and model-free RL algorithms fall into this framework.

▷ If we use data and update estimate more efficiently, we get a new algorithm called MC Exploring Starts:

Pseudocode: MC Exploring Starts (a sample-efficient variant of MC Basic)

Initialization: Initial guess π_0.
Aim: Search for an optimal policy.

For each episode, do
  Episode generation: Randomly select a starting state-action pair (s_0, a_0) and ensure that all pairs can be possibly selected. Following the current policy, generate an episode of length T: $s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T$.
  Policy evaluation and policy improvement:
  Initialization: $g \leftarrow 0$
  For each step of the episode, $t = T-1, T-2, \ldots, 0$, do
    $g \leftarrow \gamma g + r_{t+1}$
    Use the first-visit strategy:
    If (s_t, a_t) does not appear in $(s_0, a_0, s_1, a_1, \ldots, s_{t-1}, a_{t-1})$, then
      $Returns(s_t, a_t) \leftarrow Returns(s_t, a_t) + g$
      $q(s_t, a_t) = \text{average}(Returns(s_t, a_t))$
      $\pi(a|s_t) = 1$ if $a = \arg\max_a q(s_t, a)$

4. MC without exploring starts.

Algorithm: MC ε-Greedy.

soft-policy

{ deterministic policy ← greedy policy

stochastic policy ← { soft policy
ε greedy policy

▷ A policy is called *soft* if the probability to take any action is positive.

▷ Why introduce soft policies? (S, a)

- With a soft policy, a few episodes that are sufficiently long can visit every state-action pair for sufficiently many times.
- Then, we do not need to have a large number of episodes starting from every state-action pair. Hence, the requirement of exploring starts can thus be removed.

ε-greedy policies:

$$\pi(a|s) = \begin{cases} 1 - \frac{\varepsilon}{|A(s)|}(|A(s)|+1) & \text{for the greedy action} \\ \frac{\varepsilon}{|A(s)|} & \text{for the other } |A(s)|-1 \text{ action} \end{cases}$$

$\varepsilon \in [0, 1]$ and $|A(s)|$ is the number of actions for s

- The chance to choose the greedy action is always greater than other actions, because $1 - \frac{\varepsilon}{|\mathcal{A}(s)|}(|\mathcal{A}(s)| - 1) = 1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}(s)|} \geq \frac{\varepsilon}{|\mathcal{A}(s)|}$.
- **Why use ε-greedy?** Balance between exploitation and exploration
  - When $\varepsilon = 0$, it becomes greedy! Less exploration but more exploitation!
  - When $\varepsilon = 1$, it becomes a uniform distribution. More exploration but less exploitation.

▷ **How to embed ε-greedy into the MC-based RL algorithms?**

Originally, the policy improvement step in MC Basic and MC Exploring Starts is to solve

$$\pi_{k+1}(s) = \arg\max_{\pi \in \Pi} \sum_a \pi(a|s) q_{\pi_k}(s,a).$$

where Π denotes the set of all possible policies. The optimal policy here is

$$\pi_{k+1}(a|s) = \begin{cases} 1, & a = a_k^*, \\ 0, & a \neq a_k^*, \end{cases}$$

where $a_k^* = \arg\max_a q_{\pi_k}(s,a)$.

▷ **How to embed ε-greedy into the MC-based RL algorithms?**

Now, the policy improvement step is changed to solve

$$\pi_{k+1}(s) = \arg\max_{\pi \in \Pi_\varepsilon} \sum_a \pi(a|s) q_{\pi_k}(s,a),$$

where Π_ε denotes the set of all ε-greedy policies with a fixed value of ε. The optimal policy here is

$$\pi_{k+1}(a|s) = \begin{cases} 1 - \frac{|\mathcal{A}(s)|-1}{|\mathcal{A}(s)|}\varepsilon, & a = a_k^*, \\ \frac{1}{|\mathcal{A}(s)|}\varepsilon, & a \neq a_k^*. \end{cases}$$

- MC ε-Greedy is the *same* as that of MC Exploring Starts *except* that the former uses ε-greedy policies.
- It does not require exploring starts, but still requires to visit all state-action pairs in a different form.

Pseudocode: MC ϵ-Greedy (a variant of MC Exploring Starts)

Initialization: Initial guess π_0 and the value of $\epsilon \in [0,1]$
Aim: Search for an optimal policy.

For each episode, do
  Episode generation: Randomly select a starting state-action pair (s_0, a_0). Following the current policy, generate an episode of length T: $s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T$.
  Policy evaluation and policy improvement:
  Initialization: $g \leftarrow 0$
  For each step of the episode, $t = T-1, T-2, \ldots, 0$, do
    $g \leftarrow \gamma g + r_{t+1}$
    Use the every-visit method:
    If (s_t, a_t) does not appear in $(s_0, a_0, s_1, a_1, \ldots, s_{t-1}, a_{t-1})$, then
      $Returns(s_t, a_t) \leftarrow Returns(s_t, a_t) + g$
      $q(s_t, a_t) =$ average$(Returns(s_t, a_t))$
      Let $a^* = \arg\max_a q(s_t, a)$ and

$$\pi(a|s_t) = \begin{cases} 1 - \frac{|\mathcal{A}(s_t)|-1}{|\mathcal{A}(s_t)|}\epsilon, & a = a^* \\ \frac{1}{|\mathcal{A}(s_t)|}\epsilon, & a \neq a^* \end{cases}$$