Lecture 5: Monte Carlo Learning

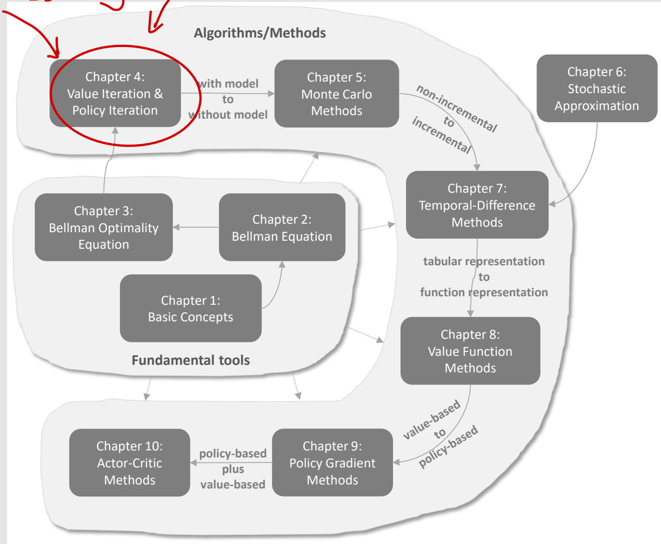
Shiyu Zhao

Department of Artificial Intelligence

Westlake University

Outline

Model-Based Reinforcement Learning (MBRL) 属于 dynamic programming



Outline

蒙特卡洛 Monte Carlo

1. Motivating example
2. The simplest MC-based RL algorithm (简单，但实际没法用)
   - Algorithm: MC Basic
3. Use data more efficiently
   - Algorithm: MC Exploring Starts
4. MC without exploring starts
   - Algorithm: MC ε-Greedy

Outline

Motivating example: Monte Carlo estimation

▷ How can we estimate something **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 $\mathbb{E}[X]$.

Motivating example: Monte Carlo estimation

有model $\longrightarrow$ 知道 probability / distribution

▷ **Method 1: with model**

- Suppose the probabilistic model is known as

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

Then by definition

$$\mathbb{E}[X] = \sum_x xp(x) = 1 \times 0.5 + (-1) \times 0.5 = 0$$

- **Problem: it may be impossible to know the precise distribution!**

▷ **Method 2: without model or called model-free**

- *Idea:* Flip the coin many times, and then calculate the average of the outcomes.
- Suppose we get a sample sequence: $\{x_1, x_2, \ldots, x_N\}$.
  Then, the mean can be approximated as

大数定律

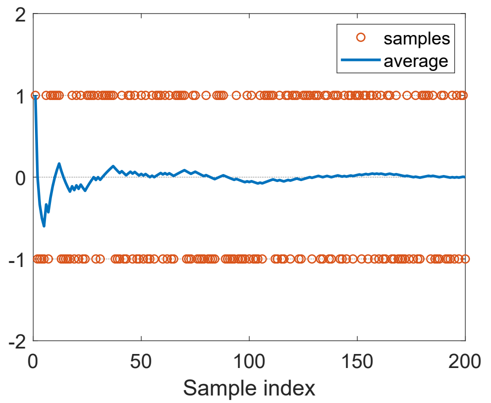
$$\mathbb{E}[X] \approx \bar{x} = \frac{1}{N}\sum_{j=1}^{N} x_j.$$

This is the idea of Monte Carlo estimation!

Motivating example: Monte Carlo estimation

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

- When N is small, the approximation is inaccurate.
- When N is large, the approximation is more accurate.



大数定律：当独立重复实验的次数足够大时，样本的平均值会越来越接近总体的期望值

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,

$$\mathbb{E}[\bar{x}] = \mathbb{E}[X],$$

$$\mathrm{Var}[\bar{x}] = \frac{1}{N}\mathrm{Var}[X]. \longrightarrow$$

$n \to \infty$ 时，方差 $\to 0$，$\bar{x} \to$ 常数

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.

▷ Summary:

- Monte Carlo estimation refers to a broad class of techniques that rely on repeated random sampling to solve approximation problems.
- Why we care about Monte Carlo estimation? Because it does not require the model!
- Why we care about mean estimation? Because state value and action value are defined as expectations of random variables!

Outline

Convert policy iteration to be model-free

The key idea is to convert the *policy iteration* algorithm to be model-free.

It is simple to understand if you

- understand the policy iteration algorithm well, and
- understand the idea of Monte Carlo mean estimation.

Convert policy iteration to be model-free

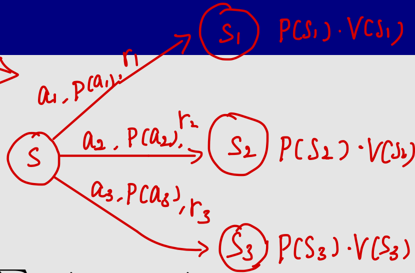
Policy iteration has two steps in each iteration:

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

The elementwise form of the **policy improvement step** is:

$$\begin{aligned} \pi_{k+1}(s) &= \arg\max_\pi \sum_a \pi(a|s) \underbrace{\left[\sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a) v_{\pi_k}(s')\right]}_{} \\ &= \arg\max_\pi \sum_a \pi(a|s) q_{\pi_k}(s,a), \quad s \in \mathcal{S} \end{aligned}$$

The key is to calculate $q_{\pi_k}(s,a)$!



所有的参数都知道，直接算

Two expressions of action value:

- **Expression 1 requires the model:**

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

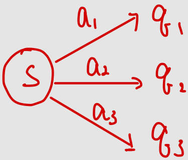
3个是 model 提供的

- **Expression 2 does not require the model:**

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

Idea to achieve model-free RL: We can use expression 2 to obtain $q_{\pi_k}(s,a)$ based on *data (samples or experiences)*!

用大数定理试出来
q的近似值



Convert policy iteration to be model-free

How to obtain $q_{\pi_k}(s,a)$ based on data?

- Starting from (s,a), following policy π_k, generate an episode.
- The return of this episode is $g(s,a)$, which is a sample of G_t in

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

- Suppose we have a set of episodes and hence $\{g^{(j)}(s,a)\}$. Then,

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

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

" samples/experience

设 initial π_0 $\xrightarrow{\text{用大数定律}}$ 所有S的所有 action value $\xrightarrow{\max q}$ π_1 / V_{π_1}

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

- **Step 1: policy evaluation.** This step aims to estimate $q_{\pi_k}(s,a)$ for all (s,a). Specifically, for each (s,a), run sufficiently many episodes. The average of their returns, denoted as $q_k(s,a)$, is used to approximate $q_{\pi_k}(s,a)$.
  - The first step of the *policy iteration algorithm* calculates $q_{\pi_k}(s,a)$ by firstly solving the state values v_{π_k} from the Bellman equation. This requires the model.
  - The first step of the *MC Basic algorithm* is to directly estimate $q_k(s,a)$ from experiences samples. This does not require the model.
- **Step 2: policy improvement.** This step aims to solve $\pi_{k+1}(s) = \arg\max_\pi \sum_a \pi(a|s) q_k(s,a)$ for all $s \in \mathcal{S}$. The greedy optimal policy is $\pi_{k+1}(a_k^*|s) = 1$ where $a_k^* = \arg\max_a q_k(s,a)$.
  - This step is exactly the same as the second step of the policy iteration algorithm.

▷ Description of the algorithm:

Pseudocode: MC Basic algorithm (a model-free variant of policy iteration)

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

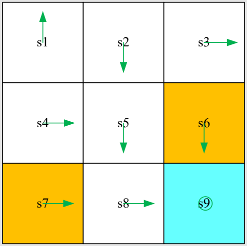
For the kth iteration ($k = 0, 1, 2, \dots$), do
    For every state $s \in \mathcal{S}$, do
        For every action $a \in \mathcal{A}(s)$, do
            Collect sufficiently many episodes starting from (s, a) following π_k
            Policy evaluation:
            $q_{\pi_k}(s, a) \approx q_k(s, a) =$ average return of all the episodes starting from (s, a)
        Policy improvement:
        $a_k^*(s) = \arg\max_a q_k(s, a)$
        $\pi_{k+1}(a|s) = 1$ if $a = a_k^*$, and $\pi_{k+1}(a|s) = 0$ otherwise

MC Basic是Policy iteration的变形

- 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.

如果MC Basic用大数定律先估计 state value，但因为没有model（转移概率和回报），还得用大数定律求q再求π.

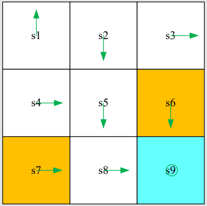
Illustrative example 1: step by step



Task:

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

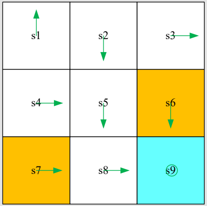
Illustrative example 1: step by step



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

Illustrative example 1: step by step



▷ Due to space limitation, we only show $q_{\pi_k}(s_1, a)$

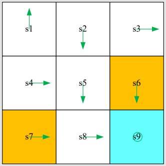
▷ **Step 1 - policy evaluation:**

- Since the current policy is deterministic, one episode would be sufficient to get the action value! Otherwise, if the policy or model is stochastic, many episodes are required!

因为policy和environment都是deterministic，每次采样都相同，所以只需采样一次
若有stochastic，则用大数定律

Illustrative example 1: step by step

每次运行model返回:
① 当前state
② 选择的action
③ reward
④ 下一个state



action value是return

$q_\pi(s,a) = E_\pi[G_t \mid s,a]$

- Starting from (s_1, a_1), the episode is $s_1 \xrightarrow{a_1} s_1 \xrightarrow{a_1} s_1 \xrightarrow{a_1} \ldots$. Hence, the action value is

这是跑一次的return

$$q_{\pi_0}(s_1, a_1) = -1 + \gamma(-1) + \gamma^2(-1) + \ldots$$

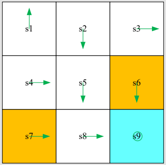
- Starting from (s_1, a_2), the episode is $s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_3} \ldots$. Hence, the action value is

$$q_{\pi_0}(s_1, a_2) = 0 + \gamma 0 + \gamma^2 0 + \gamma^3(1) + \gamma^4(1) + \ldots$$

- Starting from (s_1, a_3), the episode is $s_1 \xrightarrow{a_3} s_4 \xrightarrow{a_2} s_5 \xrightarrow{a_3} \ldots$. Hence, the action value is

$$q_{\pi_0}(s_1, a_3) = 0 + \gamma 0 + \gamma^2 0 + \gamma^3(1) + \gamma^4(1) + \ldots$$

Illustrative example 1: step by step

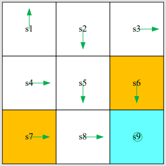


- Starting from (s_1, a_4), the episode is $s_1 \xrightarrow{a_4} s_1 \xrightarrow{a_1} s_1 \xrightarrow{a_1} \dots$. Hence, the action value is

$$q_{\pi_0}(s_1, a_4) = -1 + \gamma(-1) + \gamma^2(-1) + \dots$$

- Starting from (s_1, a_5), the episode is $s_1 \xrightarrow{a_5} s_1 \xrightarrow{a_1} s_1 \xrightarrow{a_1} \dots$. Hence, the action value is

$$q_{\pi_0}(s_1, a_5) = 0 + \gamma(-1) + \gamma^2(-1) + \dots$$



▷ **Step 2 - policy improvement:**

- By observing the action values, we see that

$$q_{\pi_0}(s_1, a_2) = q_{\pi_0}(s_1, a_3)$$

  are the *maximum*.

- As a result, the policy can be improved as

$$\pi_1(a_2|s_1) = 1 \text{ or } \pi_1(a_3|s_1) = 1.$$

  In either way, the new policy for s_1 becomes optimal.

One iteration is sufficient for this simple example!

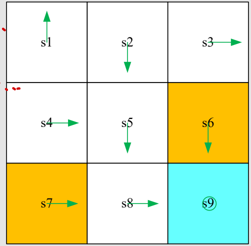
$$q(s_3, a_1) = -1 + \gamma(-1) + \gamma^2(-1) + \dots = -1 \times \frac{1}{1-\gamma} = -\frac{1}{0.1} = -10$$

$$q(s_3, a_2) = -1 + \gamma(-1) + \gamma^2(-1) + \dots = -10$$

$$q(s_3, a_3) = -1 + \gamma + \gamma^2 + \gamma^3 + \dots = -1 + \frac{\gamma}{1-\gamma} = -1 + \frac{0.9}{0.1} = 8$$

$$q(s_3, a_4) = 0 + 0 + 0 + \gamma^3 + \gamma^4 + \dots = \frac{\gamma^3}{1-\gamma} = \frac{0.9^3}{0.1} = 7.29$$

$$q(s_3, a_5) = 0 + \gamma(-1) + \gamma^2(-1) + \dots = -\frac{\gamma}{1-\gamma} = -9$$



Exercise: now update the policy for s_3 using MC Basic!

Outline

Use data more efficiently

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.

Use data more efficiently

MC Basic：用这部分的return值，
作为 $q_\pi(s_1,a_2)$

▷ Consider a grid-world example, following a policy π, we can get an episode such as

$$s_1 \xrightarrow{a_2} \boxed{s_2 \xrightarrow{a_4} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots}$$

▷ **Visit:** every time a state-action pair appears in the episode, it is called a *visit* of that state-action pair.

▷ Methods to use the data: **Initial-visit method**

- Just calculate the return and approximate $q_\pi(s_1, a_2)$.
- This is what the MC Basic algorithm does.
- Disadvantage: Not fully utilize the data. 没充分利用 episode

Use data more efficiently

▷ The episode also visits other state-action pairs.

$$s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_4} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots \quad \text{[original episode]}$$
$$s_2 \xrightarrow{a_4} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots \quad \text{[episode starting from } (s_2, a_4)\text{]}$$
$$s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots \quad \text{[episode starting from } (s_1, a_2)\text{]}$$
$$s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots \quad \text{[episode starting from } (s_2, a_3)\text{]}$$
$$s_5 \xrightarrow{a_1} \dots \quad \text{[episode starting from } (s_5, a_1)\text{]}$$

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 每个episode，每个state的expected return值只更新一次
- every-visit method 每次出现return，都更新该state的expected return

Update value estimate more efficiently

方法一：同Basic，把 $q(s_1,a_1)$，$q(s_1,a_2)$，$q(s_1,a_3)$都算出，再更新policy.

▷ 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.

方法二：

Update value estimate more efficiently

▷ Will the second method cause problems?

- One may say that the return of a single episode cannot accurately approximate the corresponding action value.
- In fact, we have done that in the truncated policy iteration algorithm introduced in the last chapter!

▷ **Generalized policy iteration:**

GPI (不需要把 action value 和 state value 精确)

- Not a specific algorithm.
- It refers to the general idea or framework of switching between policy-evaluation and policy-improvement processes.

相互交替进行.

- Many RL algorithms fall into this framework.

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

Algorithm: MC Exploring Starts (an efficient variant of MC Basic)

Initialization: Initial policy $\pi_0(a|s)$ and initial value $q(s,a)$ for all (s,a). Returns$(s,a) = 0$ and Num$(s,a) = 0$ for all (s,a).
Goal: Search for an optimal policy.

For each episode, do

Episode generation: Select a starting state-action pair (s_0, a_0) and ensure that all pairs can be possibly selected (this is the exploring-starts condition). Following the current policy, generate an episode of length T: $s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T$.
Initialization for each episode: $g \leftarrow 0$
For each step of the episode, $t = T-1, T-2, \ldots, 0$, do

$g \leftarrow \gamma g + r_{t+1}$
Returns$(s_t, a_t) \leftarrow$ Returns$(s_t, a_t) + g$
Num$(s_t, a_t) \leftarrow$ Num$(s_t, a_t) + 1$
Policy evaluation:
$q(s_t, a_t) \leftarrow$ Returns(s_t, a_t)/Num(s_t, a_t)
Policy improvement:
$\pi(a|s_t) = 1$ if $a = \arg\max_a q(s_t, a)$ and $\pi(a|s_t) = 0$ otherwise

MC Exploring Starts: 设 $T=3$

$$(s_0,a_0) \xrightarrow{r_1} (s_1,a_1) \xrightarrow{r_2} (s_2,a_2) \xrightarrow{r_3} \dots$$

g_3

$g_2 = r_2 + \gamma g_3$

$g_1 = r_1 + \gamma g_2$

For each episode, do:

① 随机选择一个 start pair (s_0,a_0)，保证公平(所有 pair都可能为 start pair)

② 由当前 policy，生成一个 episode：$(s_0,a_0,r_1,s_1,a_1,r_2,\dots,s_{T-1},a_{T-1},r_T)$

这时跑出每个pair对应的reward

③ 从后往前计算：$t=T-1,T-2,T-3,\dots,0$

- 计算当前 (s_t,a_t) 的 return：$g = r_{t+1} + \gamma g$
- $\text{Return}(s_t,a_t) \leftarrow \text{Return}(s_t,a_t) + g$
- $\text{Num}(s_t,a_t) \leftarrow \text{Num}(s_t,a_t) + 1$
- Policy evaluation：$q(s_t,a_t) = \dfrac{\text{Return}}{\text{Num}}$
- Policy improvement

MC Exploring Starts

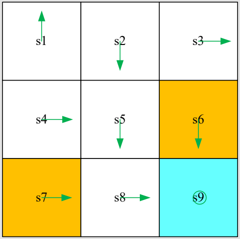
▷ **What is exploring starts?** 每个 (s, a) pair 都能被访问

- Exploring starts means we need to generate sufficiently many episodes $\underbrace{\textit{starting}}_{\text{starts}}$ from $\underbrace{\textit{every}}_{\text{exploring}}$ state-action pair.

为了

每个 (s, a) pair 都有以其为起始点的 episode

- For example, we need episodes starting from $\{(s_1, a_j)\}_{j=1}^5$, $\{(s_2, a_j)\}_{j=1}^5$, $\ldots$, $\{(s_9, a_j)\}_{j=1}^5$.
  Otherwise, if there are no episodes starting from (s_i, a_j), then (s_i, a_j) may not be visited by any episodes, and hence $q_\pi(s_i, a_j)$ cannot be estimated.



- Both MC Basic and MC Exploring Starts need this assumption.

▷ **Why do we need to consider exploring starts?**

- In theory, only if every action value for every state is well explored, can we select the optimal actions correctly.
  Otherwise, if an action is not explored, this action may happen to be the optimal one and hence be missed.
- In practice, exploring starts is difficult to achieve. For many applications, especially those involving physical interactions with environments, it is difficult to collect episodes starting from every state-action pair.

Can we remove the requirement of exploring starts? We next show that we can do that by using soft policies.

Outline

Soft policies

policy
确定性 deterministic : greedy
不确定性 stochastic : soft (ε-greedy)

▷ What is a soft policy?

- A policy is *soft* if the probability to take any action is positive.
  - Deterministic policy: for example, greedy policy
  - Stochastic policy: for example, soft policy

▷ Why introducing soft policies?

- With a soft policy, a few episodes that are sufficiently long can visit every state-action pair.
- 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 be removed.

ε-greedy policies

- What soft policies will we use? Answer: ε-greedy policies
- What is the ε-greedy policy?

$q_\pi(s, a^*)$

$$\pi(a|s) = \begin{cases} 1 - \frac{\varepsilon}{|\mathcal{A}(s)|}(|\mathcal{A}(s)| - 1), & \text{for the greedy action, 最大action value} \\ \frac{\varepsilon}{|\mathcal{A}(s)|}, & \text{for the other } |\mathcal{A}(s)| - 1 \text{ actions.} \end{cases}$$

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

- Example: if $\epsilon = 0.2$, then

$$\frac{\varepsilon}{|\mathcal{A}(s)|} = \frac{0.2}{5} = 0.04, \qquad 1 - \frac{\varepsilon}{|\mathcal{A}(s)|}(|\mathcal{A}(s)| - 1) = 1 - 0.04 \times 4 = 0.84$$

- The chance to choose the greedy action is always greater than other actions, because

$$1 - \frac{\varepsilon}{|\mathcal{A}(s)|}(|\mathcal{A}(s)| - 1) = 1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}(s)|} \geq \frac{\varepsilon}{|\mathcal{A}(s)|}$$

最大的概率给 greedy action.

利用 探索

ε-greedy policies can balance exploitation and exploration.

- When $\varepsilon \to 0$, it becomes greedy!

$$\pi(a|s) = \begin{cases} 1 - \frac{\varepsilon}{|\mathcal{A}(s)|}(|\mathcal{A}(s)| - 1) = 1, & \text{for the greedy action,} \\ \frac{\varepsilon}{|\mathcal{A}(s)|} = 0, & \text{for the other } |\mathcal{A}(s)| - 1 \text{ actions.} \end{cases}$$

More exploitation but less exploration.

- When $\varepsilon \to 1$, it becomes a uniform distribution.

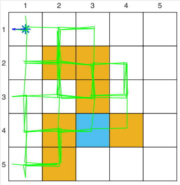
$$\pi(a|s) = \begin{cases} 1 - \frac{\varepsilon}{|\mathcal{A}(s)|}(|\mathcal{A}(s)| - 1) = \frac{1}{|\mathcal{A}(s)|}, & \text{for the greedy action,} \\ \frac{\varepsilon}{|\mathcal{A}(s)|} = \frac{1}{|\mathcal{A}(s)|}, & \text{for the other } |\mathcal{A}(s)| - 1 \text{ actions.} \end{cases}$$

More exploration but less exploitation.

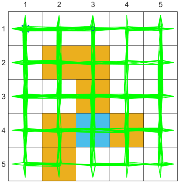
▷ **Can a single episode visit all state-action pairs?**

When $\varepsilon = 1$, the policy (uniform distribution) has the strongest exploration ability.

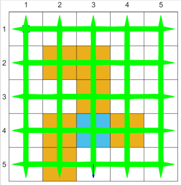
一个state中，所有action的概率相同



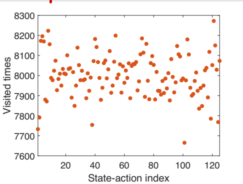
(a) 100 steps



(b) 1000 steps



(c) 10000 steps



(d)

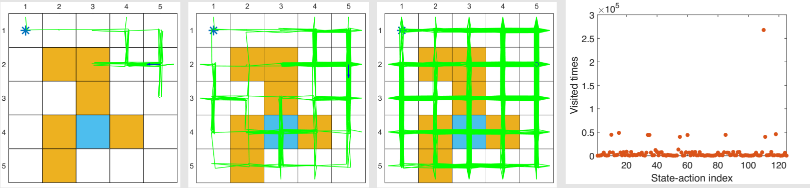
Click here to play a video (the video is only on my computer)

Examples to demonstrate the exploration ability

▷ **Can a single episode visit all state-action pairs?**

但所有的 state action pair都会访问到.

When ε is small, the exploration ability of the policy is also small.



(a) 100 steps (b) 1000 steps (c) 10000 steps (d)

▷ **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

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

体现是 deterministic

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

和之前不一样了

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

greedy → 只有0和1

ε-greedy → 概率.

Summary:

- 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.

MC ε-Greedy algorithm

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

Initialization: Initial policy $\pi_0(a|s)$ and initial value $q(s,a)$ for all (s,a). Returns$(s,a) = 0$ and Num$(s,a) = 0$ for all (s,a). $\epsilon \in (0,1]$
Goal: Search for an optimal policy.

For each episode, do

Episode generation: Select a starting state-action pair (s_0, a_0) (the exploring starts condition is not required). Following the current policy, generate an episode of length T: $s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T$.

Initialization for each episode: $g \leftarrow 0$

For each step of the episode, $t = T-1, T-2, \ldots, 0$, do

$g \leftarrow \gamma g + r_{t+1}$

Returns$(s_t, a_t) \leftarrow$ Returns$(s_t, a_t) + g$

Num$(s_t, a_t) \leftarrow$ Num$(s_t, a_t) + 1$

Policy evaluation:

$q(s_t, a_t) \leftarrow$ Returns$(s_t, a_t)/$Num(s_t, a_t)

Policy improvement:

Let $a^* = \arg\max_a q(s_t, a)$ and

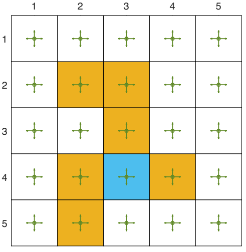
$$\pi(a|s_t) = \begin{cases} 1 - \frac{|\mathcal{A}(s_t)|-1}{|\mathcal{A}(s_t)|}\epsilon, & a = a^* \\ \frac{1}{|\mathcal{A}(s_t)|}\epsilon, & a \neq a^* \end{cases}$$

▷ Demonstrate the MC ε-Greedy algorithm.

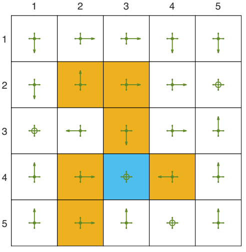
In every iteration, do the following:

- In the episode generation step, use the previous policy generates a single episode of 1 million steps!
- In the rest steps, use the single episode to update the policy.
- Two iterations can lead to the optimal ε-greedy policy.

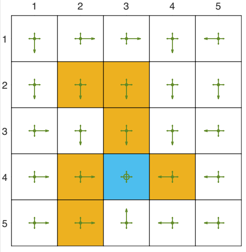
Here, $r_{\text{boundary}} = -1$, $r_{\text{forbidden}} = -10$, $r_{\text{target}} = 1$, $\gamma = 0.9$.



(a) Initial policy



(b) After the first iteration



(c) After the second iteration

Optimality vs exploration

▷ Compared to greedy policies,

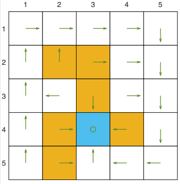
- The **advantage** of ε-greedy policies is that they have strong exploration ability when ε is large.
  - Then, the exploring starts condition is not required.
- The **disadvantage** is that ε-greedy polices are not optimal in general.
  - It is only optimal in the set Π_ε of all ε-greedy policies.
  - ε cannot be too large. We can also use a decaying ε.

ε越小，Π_ε越接近Π，策略越靠近最优

▷ Next, we use examples to demonstrate. The setup is $r_{\text{boundary}} = -1$, $r_{\text{forbidden}} = -10$, $r_{\text{target}} = 1$, $\gamma = 0.9$

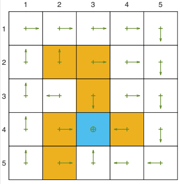
state value 变小, 因为有概率采取错误的 action.

▷ Given an ε-greedy policy, what is its state value?



	1	2	3	4	5
1	3.5	3.9	4.3	4.8	5.3
2	3.1	3.5	4.8	5.3	5.9
3	2.8	2.5	10.0	5.9	6.6
4	2.5	10.0	10.0	10.0	7.3
5	2.3	9.0	10.0	9.0	8.1

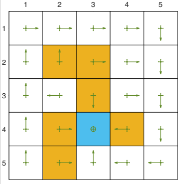
$\varepsilon = 0$



	1	2	3	4	5
1	0.4	0.5	0.9	1.3	1.4
2	0.1	0.0	0.5	1.3	1.7
3	0.1	-0.4	3.4	1.4	1.9
4	-0.1	3.4	3.3	3.7	2.2
5	-0.3	2.6	3.7	3.1	2.7

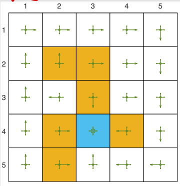
$\varepsilon = 0.1$

consistent
一致性



	1	2	3	4	5
1	-2.2	-2.4	-2.1	-1.7	-1.8
2	-2.5	-3.0	-3.3	-2.3	-2.0
3	-2.3	-3.3	-2.5	-2.8	-2.2
4	-2.5	-2.5	-2.8	-2.0	-2.4
5	-2.8	-3.2	-2.1	-2.3	-2.2

$\varepsilon = 0.2$



	1	2	3	4	5
1	-8.0	-9.0	-8.4	-7.2	-7.8
2	-8.7	-10.8	-12.4	-9.6	-8.9
3	-8.3	-12.3	-15.3	-12.3	-10.5
4	-9.7	-15.5	-17.0	-14.4	-12.2
5	-10.9	-16.7	-15.2	-14.3	-12.4

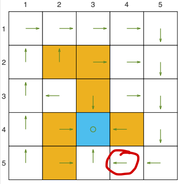
$\varepsilon = 0.5$

▷ When ε increases, the optimality of the policy becomes worse!

▷ Why is the state value of the target state negative?

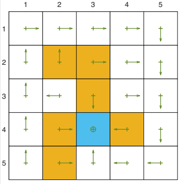
Consistency

▷ Find the optimal ε-greedy policies and their state values?



	1	2	3	4	5
1	3.5	3.9	4.3	4.8	5.3
2	3.1	3.5	4.8	5.3	5.9
3	2.8	2.5	10.0	5.9	6.6
4	2.5	10.0	10.0	10.0	7.3
5	2.3	9.0	10.0	9.0	8.1

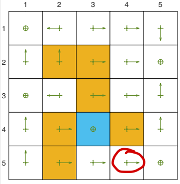
$\varepsilon = 0$



	1	2	3	4	5
1	0.4	0.5	0.9	1.3	1.4
2	0.1	0.0	0.5	1.3	1.7
3	0.1	-0.4	3.4	1.4	1.9
4	-0.1	3.4	3.3	3.7	2.2
5	-0.3	2.6	3.7	3.1	2.7

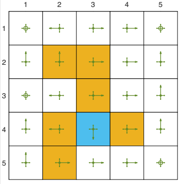
$\varepsilon = 0.1$

not consistent



	1	2	3	4	5
1	-1.1	-1.5	-1.1	-0.6	-0.6
2	-1.5	-2.2	-2.3	-1.0	-0.6
3	-1.2	-2.4	-2.2	-1.5	-0.6
4	-1.6	-2.3	-2.6	-1.4	-1.1
5	-2.0	-3.0	-1.8	-1.4	-1.0

$\varepsilon = 0.2$



	1	2	3	4	5
1	-4.3	-5.5	-4.5	-2.6	-2.3
2	-5.6	-7.7	-7.7	-4.1	-2.4
3	-5.5	-9.0	-8.0	-5.6	-2.8
4	-6.8	-8.9	-9.4	-5.5	-4.2
5	-7.9	-10.1	-6.7	-5.1	-3.7

$\varepsilon = 0.5$

▷ The optimal ε-greedy policies are not *consistent* with the greedy optimal one! Why is that? Consider the target for example.

Summary

Key points:

- Mean estimation by the Monte Carlo methods
- Three algorithms:
  - MC Basic
  - MC Exploring Starts
  - MC ε-Greedy
- Relationship among the three algorithms
- Optimality vs exploration of ε-greedy policies