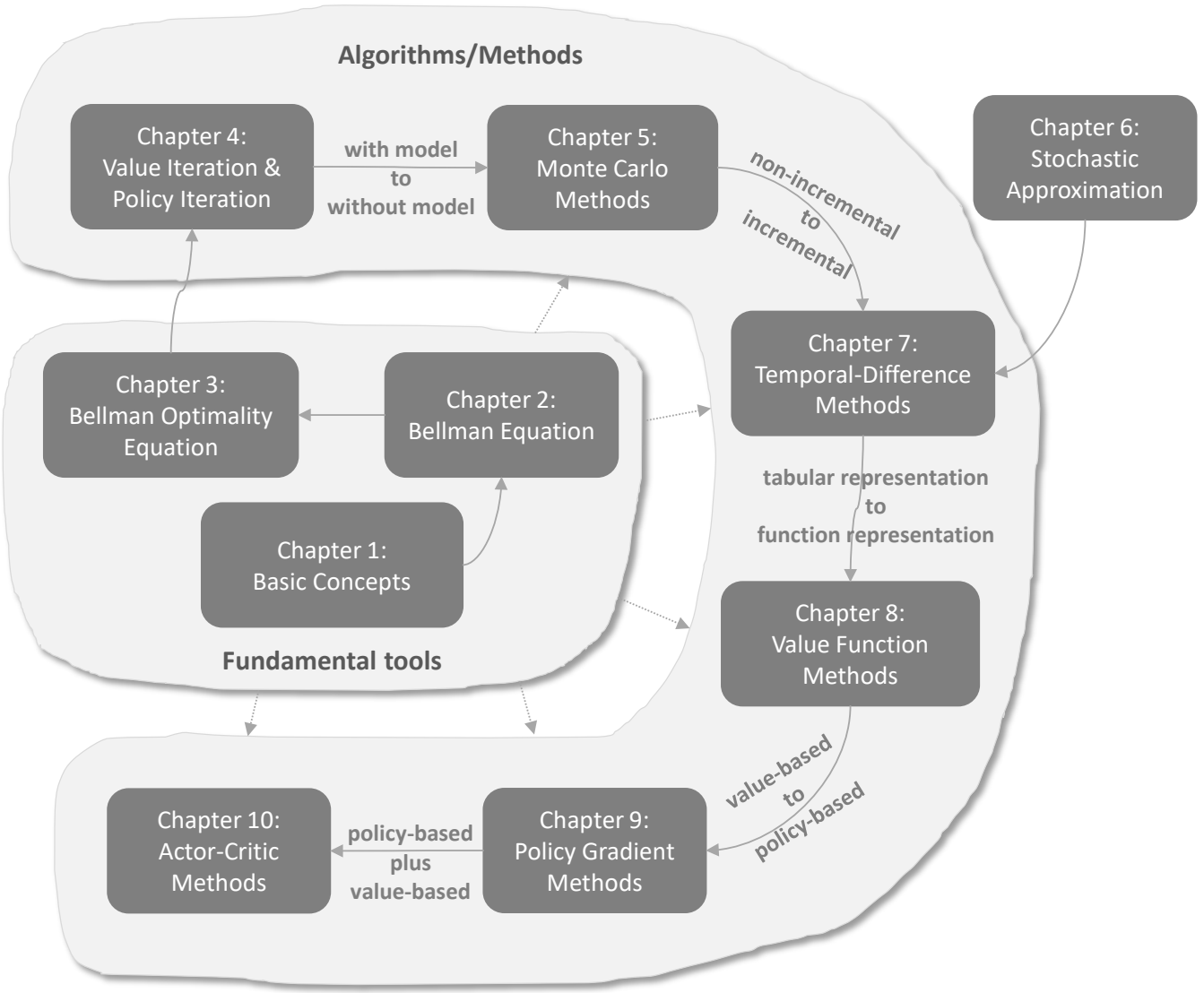
Lecture 5: Monte Carlo Methods

Shiyu Zhao

Department of Artificial Intelligence

Westlake University

Outline


Algorithms/Methods
Chapter 4: Value Iteration & Policy Iteration
with model to without model
Chapter 5: Monte Carlo Methods
non-incremental to incremental
Chapter 6: Stochastic Approximation
Chapter 7: Temporal-Difference Methods
tabular representation to function representation
Chapter 8: Value Function Methods
Chapter 3: Bellman Optimality Equation
Chapter 2: Bellman Equation
Chapter 1: Basic Concepts
Fundamental tools
value-based to policy-based
Chapter 10: Actor-Critic Methods
policy-based plus value-based
Chapter 9: Policy Gradient Methods

Outline

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

▷ 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

▷ **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!**

Motivating example: Monte Carlo estimation

▷ **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!**

Motivating example: Monte Carlo estimation

▷ **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

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

  This is the idea of Monte Carlo estimation!

Motivating example: Monte Carlo estimation

▷ **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

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

  This is the idea of Monte Carlo estimation!

Motivating example: Monte Carlo estimation

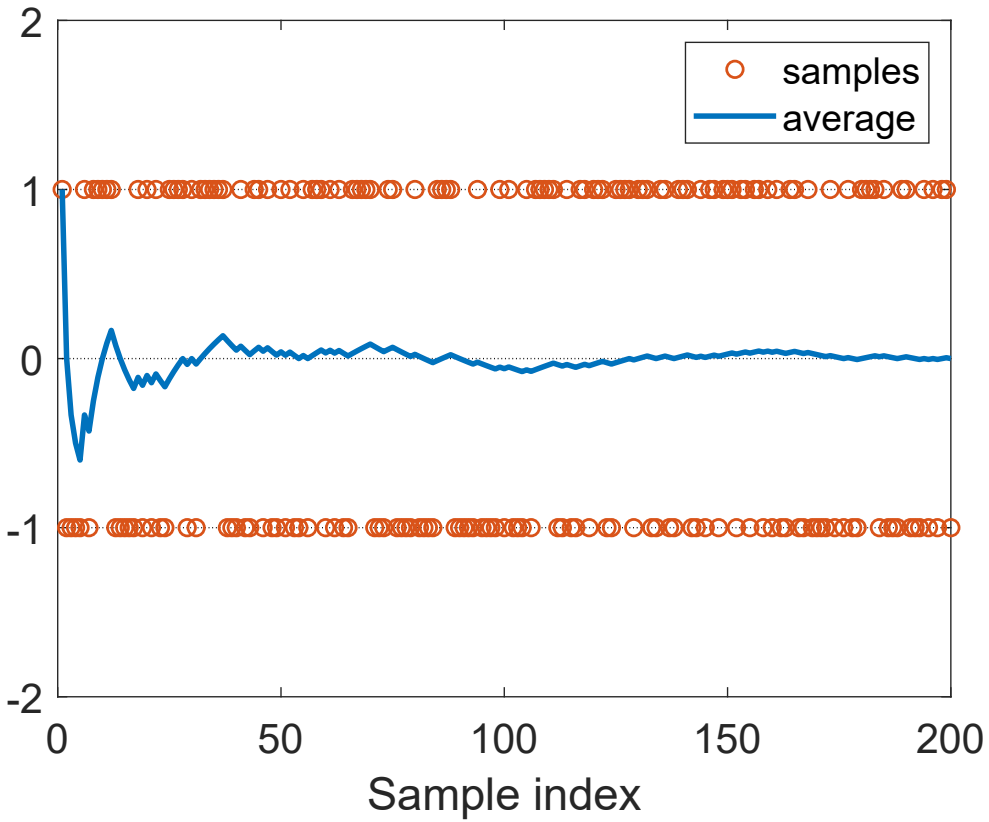
▷ **Question:** Is the Monte Carlo estimation accurate?

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

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.



Motivating example: Monte Carlo estimation

Theorem (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,

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

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

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.

Motivating example: Monte Carlo estimation

▷ 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!

Motivating example: Monte Carlo estimation

▷ 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!

Motivating example: Monte Carlo estimation

▷ 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:

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

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

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

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:

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

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

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:

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

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

Convert policy iteration to be model-free

Two expressions of action value:

- **Expression 1 requires the model:**

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

- **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)*!

Convert policy iteration to be model-free

Two expressions of action value:

- **Expression 1 requires the model:**

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

- **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)*!

Convert policy iteration to be model-free

Two expressions of action value:

- **Expression 1 requires the model:**

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

- **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)*!

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.

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.

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.

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.

The MC Basic algorithm

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.

The MC Basic algorithm

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.

The MC Basic algorithm

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.

The MC Basic 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

The MC Basic algorithm

- 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.

The MC Basic algorithm

- 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.

The MC Basic algorithm

- 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.

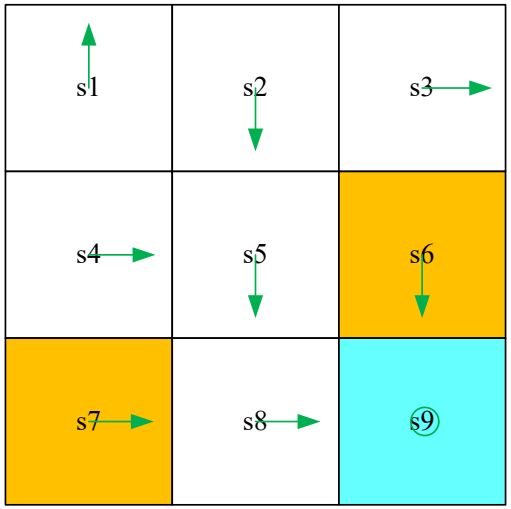
The MC Basic algorithm

- 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.

The MC Basic algorithm

- 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.

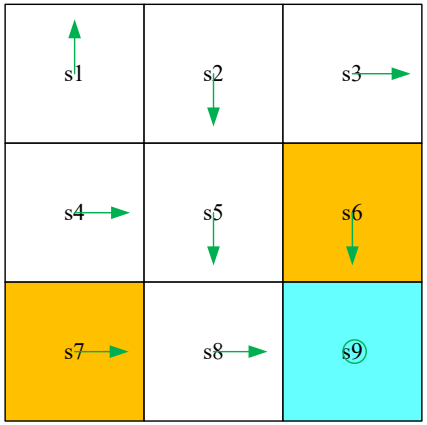
Illustrative example 1: step by step



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

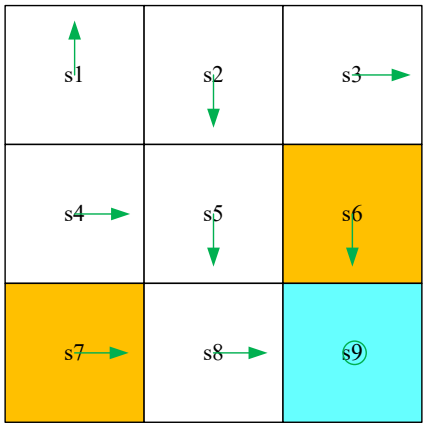
- Step 1 - policy evaluation: calculate $q_{\pi_k}(s, a)$

  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

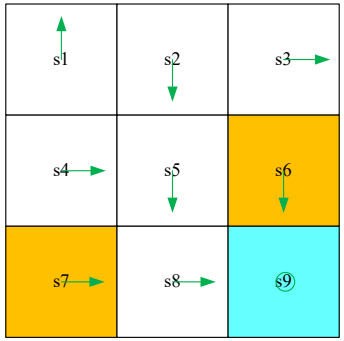


▷ Due to space limitation, we only show $q_{\pi_k}(s_1, a)$

▷ **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!

Illustrative example 1: step by step



- 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

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

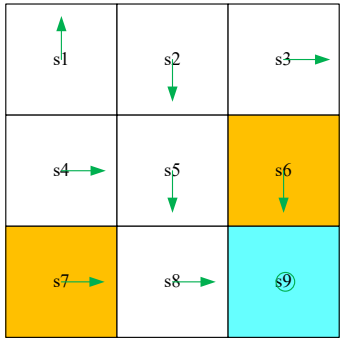
- Starting from (s_1, a_2), the episode is $s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_3} \ldots$. Hence, the action value is

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

- 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_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

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

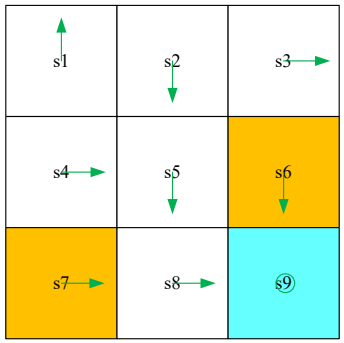
- Starting from (s_1, a_2), the episode is $s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_3} \ldots$. Hence, the action value is

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

- 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



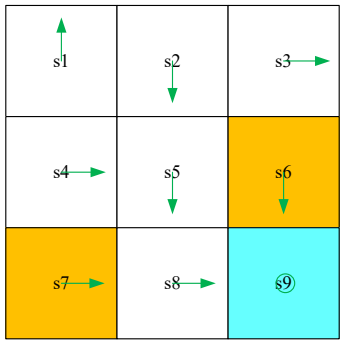
- Starting from (s_1, a_4), the episode is $s_1 \xrightarrow{a_4} s_1 \xrightarrow{a_1} s_1 \xrightarrow{a_1} \ldots$. Hence, the action value is

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

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

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

Illustrative example 1: step by step



▷ **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*.

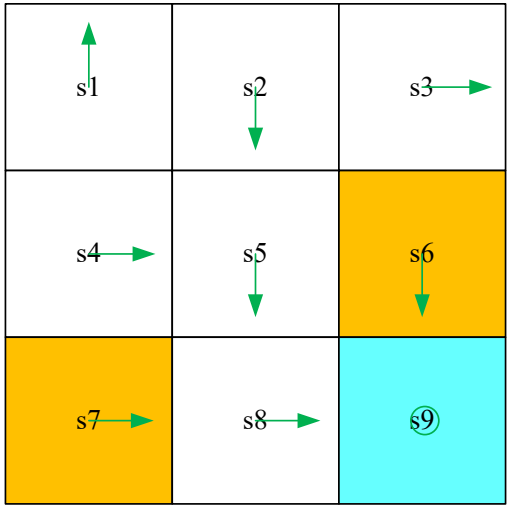
- As a result, the policy can be improved as

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

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

One iteration is sufficient for this simple example!

Illustrative example 1: step by step



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

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

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

▷ **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.

Use data more efficiently

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

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

▷ **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.

Use data more efficiently

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

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

▷ **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.

Use data more efficiently

▷ The episode also visits other state-action pairs.

$$s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_4} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots \quad \text{[original episode]}$$

Can be used to 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

Use data more efficiently

▷ The episode also visits other state-action pairs.

$$s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_4} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots \quad \text{[original episode]}$$

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

Can be used to 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

Use data more efficiently

▷ The episode also visits other state-action pairs.

$$s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_4} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots \quad \text{[original episode]}$$

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

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

Can be used to 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

Use data more efficiently

▷ The episode also visits other state-action pairs.

$$s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_4} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots \quad \text{[original episode]}$$

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

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

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

Can be used to 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

Use data more efficiently

▷ The episode also visits other state-action pairs.

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

Can be used to 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

Use data more efficiently

▷ The episode also visits other state-action pairs.

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

Can be used to 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

Use data more efficiently

▷ The episode also visits other state-action pairs.

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

Can be used to 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 value estimate 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.

Update value estimate 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.

Update value estimate 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.

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:**

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

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:**

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

MC Exploring Starts

▷ 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

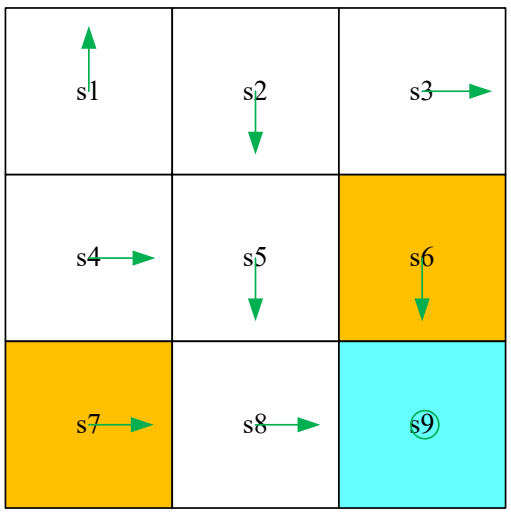
Episode generation: Select a starting state-action pair (s_0, a_0) and ensure that all pairs can be possibly selected (this is the exploring-starts condition). Following the current policy, generate an episode of length T: $s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T$.
Initialization for each episode: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

▷ **What is exploring starts?**

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



MC Exploring Starts

▷ **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.

MC Exploring Starts

▷ **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.

MC Exploring Starts

▷ **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

▷ 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.

Soft policies

▷ 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.

Soft policies

▷ 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 an ϵ-greedy policy?

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

  where $\epsilon \in [0, 1]$ and $|\mathcal{A}(s)|$ is the number of actions for s.
- Example: if $\epsilon = 0.2$, then

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

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

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

ϵ-greedy policies

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

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

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

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

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

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

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

ϵ-greedy policies

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

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

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

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

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

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

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

ϵ-greedy policies

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

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

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

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

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

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

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

ϵ-greedy policies

ϵ-greedy policies can balance exploitation and exploration.

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

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

More exploitation but less exploration.

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

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

More exploration but less exploitation.

ϵ-greedy policies

ϵ-greedy policies can balance exploitation and exploration.

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

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

More exploitation but less exploration.

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

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

More exploration but less exploitation.

ϵ-greedy policies

ϵ-greedy policies can balance exploitation and exploration.

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

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

More exploitation but less exploration.

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

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

More exploration but less exploitation.

ϵ-greedy policies

ϵ-greedy policies can balance exploitation and exploration.

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

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

More exploitation but less exploration.

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

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

More exploration but less exploitation.

ϵ-greedy policies

ϵ-greedy policies can balance exploitation and exploration.

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

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

More exploitation but less exploration.

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

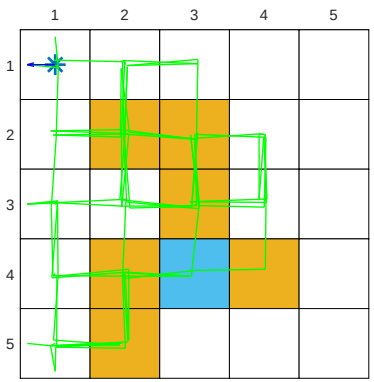
$$\pi(a|s) = \begin{cases} 1 - \frac{\epsilon}{|\mathcal{A}(s)|}(|\mathcal{A}(s)| - 1) = \frac{1}{|\mathcal{A}(s)|}, & \text{for the greedy action,} \\ \frac{\epsilon}{|\mathcal{A}(s)|} = \frac{1}{|\mathcal{A}(s)|}, & \text{for the other } |\mathcal{A}(s)| - 1 \text{ actions.} \end{cases}$$

More exploration but less exploitation.

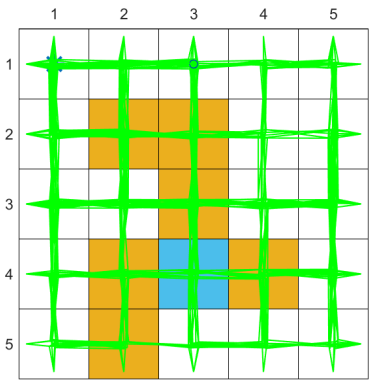
Examples to demonstrate the exploration ability

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

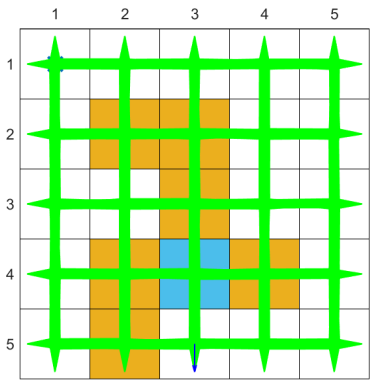
When $\epsilon = 1$, the policy (uniform distribution) has the strongest exploration ability.



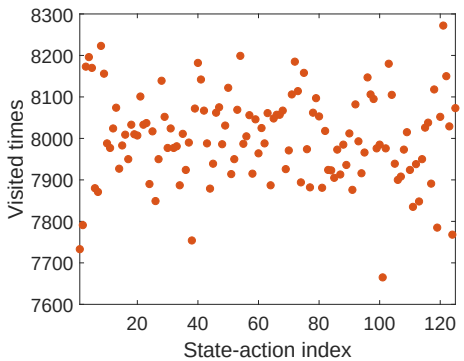
(a) 100 steps



(b) 1000 steps



(c) 10000 steps



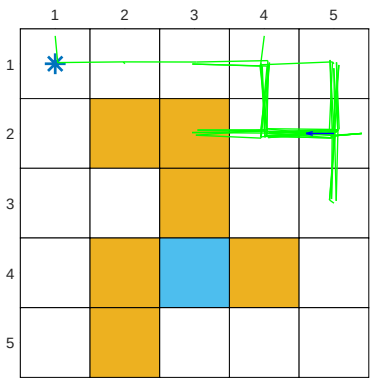
(d)

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

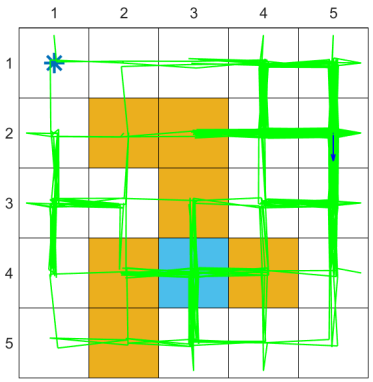
Examples to demonstrate the exploration ability

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

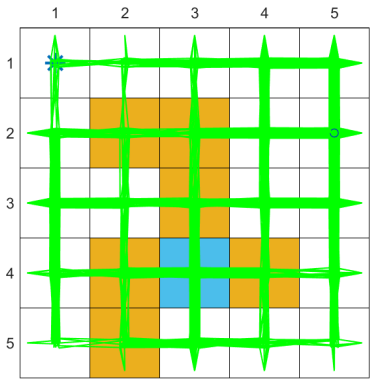
When ϵ is small, the exploration ability of the policy is also small.



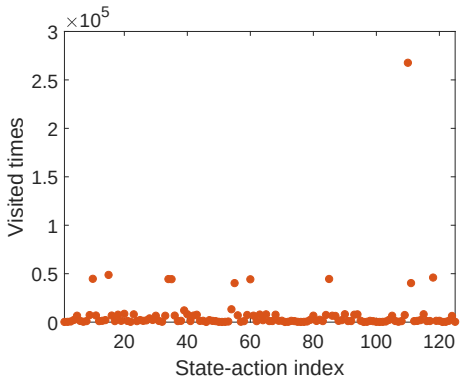
(a) 100 steps



(b) 1000 steps



(c) 10000 steps



(d)

MC ϵ-Greedy algorithm

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

MC ϵ-Greedy algorithm

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

MC ϵ-Greedy algorithm

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

MC ϵ-Greedy algorithm

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

Now, the policy improvement step is changed to solve

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

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

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

Summary:

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

MC ϵ-Greedy algorithm

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

Now, the policy improvement step is changed to solve

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

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

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

Summary:

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

MC ϵ-Greedy algorithm

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

Now, the policy improvement step is changed to solve

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

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

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

Summary:

- MC ϵ-Greedy is the *same* as that of MC Exploring Starts *except* that the former uses ϵ-greedy policies.
- MC ϵ-Greedy does not require the exploring starts condition, 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

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

Example

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

Example

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

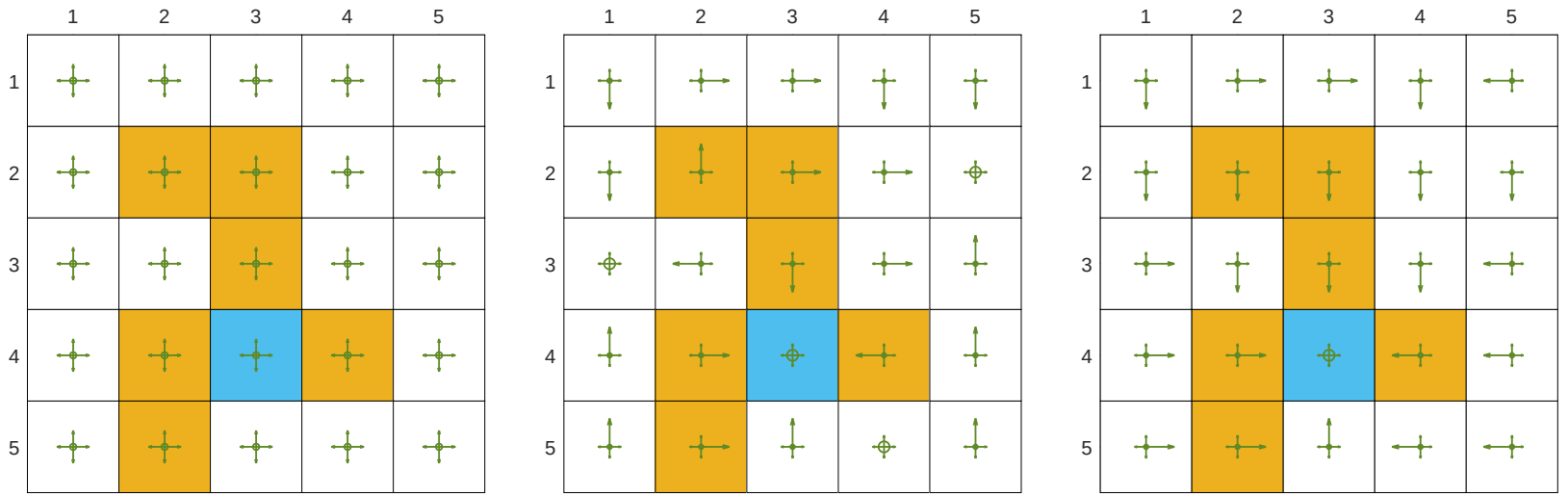
Example

▷ 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 within the set Π_ϵ of all ϵ-greedy policies.
  - Then, ϵ should not be too large. We can also use a decaying ϵ.

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

Optimality vs exploration

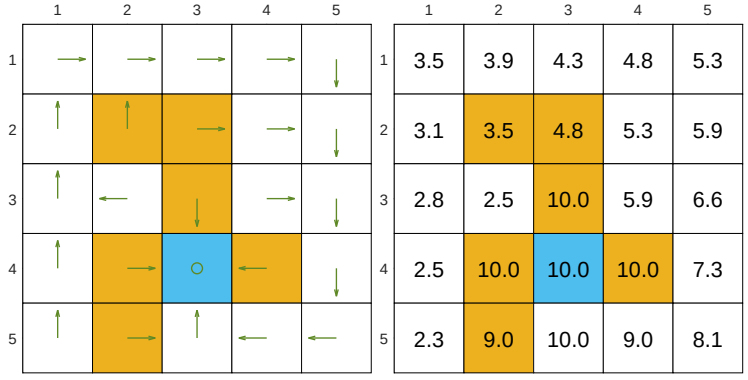
▷ Compared to greedy policies,

- 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 within the set Π_ϵ of all ϵ-greedy policies.
  - Then, ϵ should not be too large. We can also use a decaying ϵ.

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

Optimality

▷ **Examples: 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

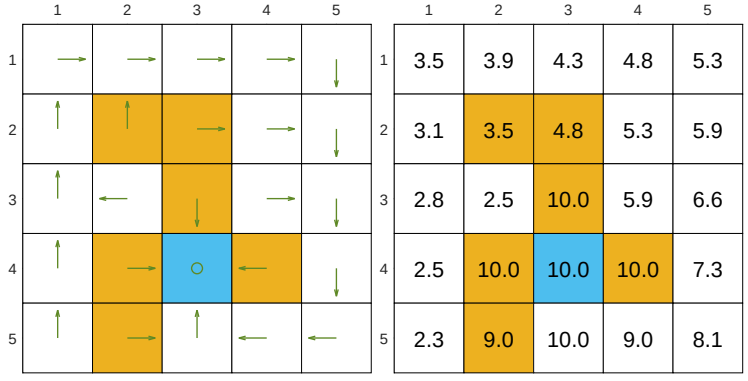
$\epsilon = 0$

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

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

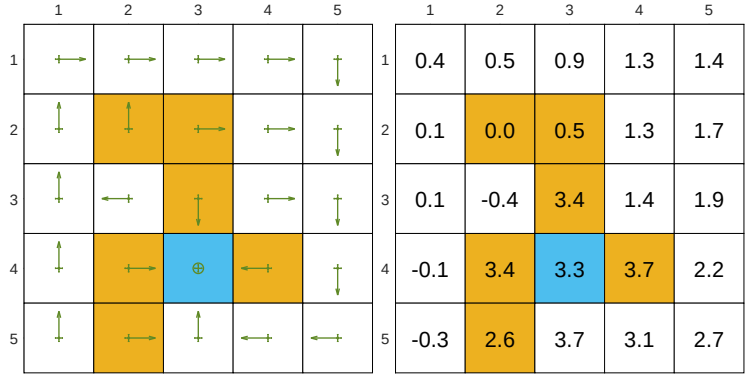
Optimality

▷ **Examples: 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

$\epsilon = 0$



	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

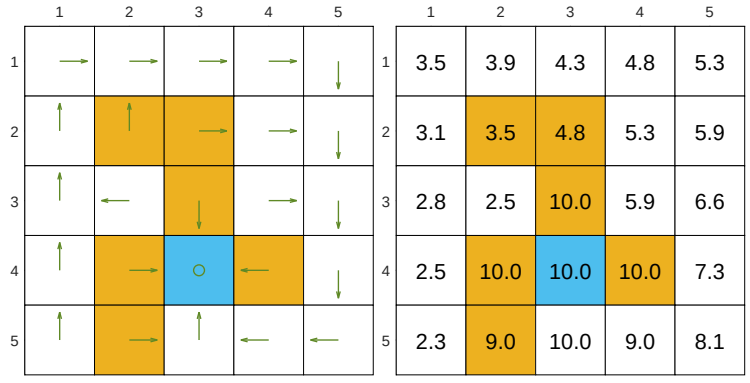
$\epsilon = 0.1$

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

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

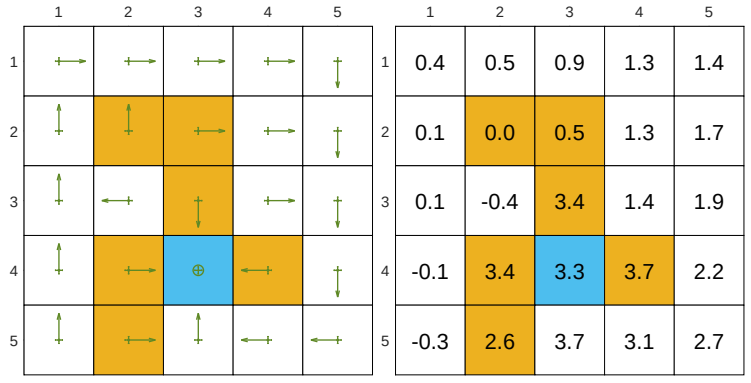
Optimality

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



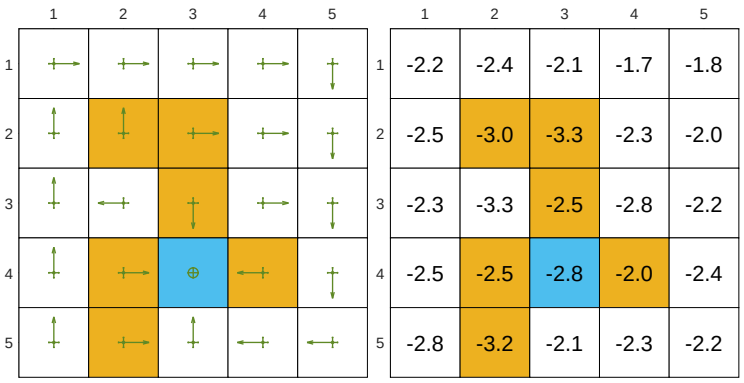
	1	2	3	4	5
1	3.5	3.9	4.3	4.8	5.3
2	3.1	3.5	4.8	5.3	5.9
3	2.8	2.5	10.0	5.9	6.6
4	2.5	10.0	10.0	10.0	7.3
5	2.3	9.0	10.0	9.0	8.1

$\epsilon = 0$



	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

$\epsilon = 0.1$



	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

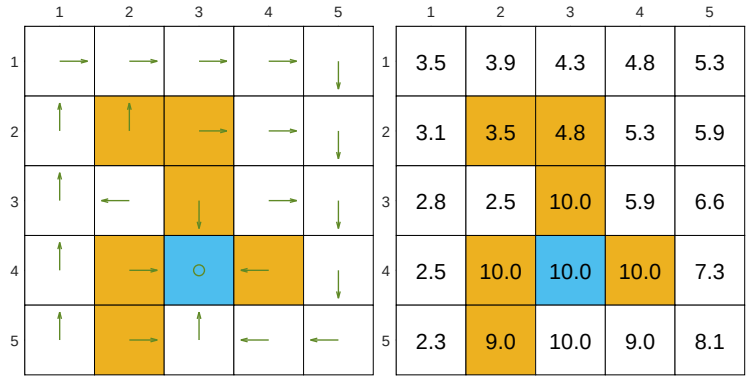
$\epsilon = 0.2$

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

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

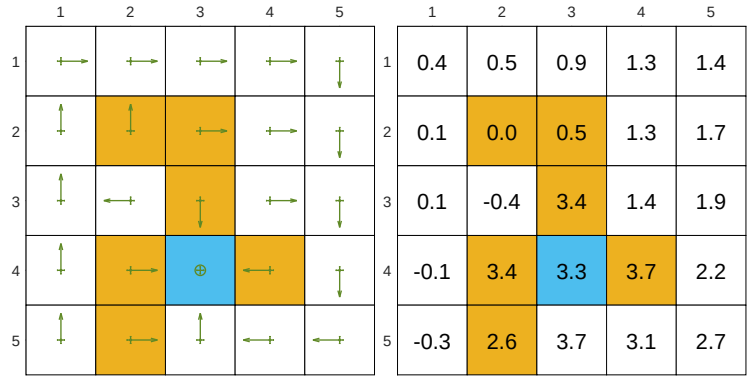
Optimality

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



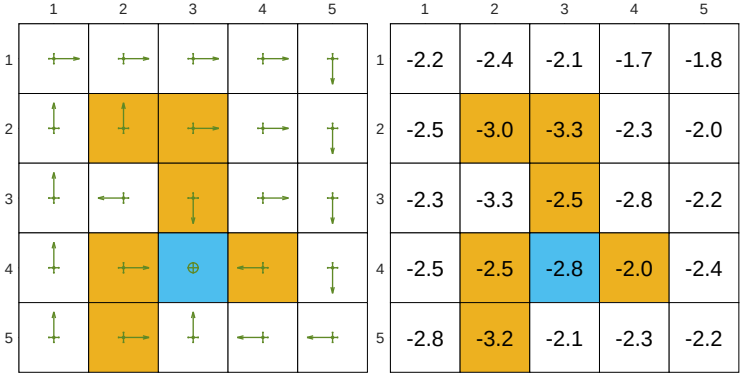
	1	2	3	4	5
1	3.5	3.9	4.3	4.8	5.3
2	3.1	3.5	4.8	5.3	5.9
3	2.8	2.5	10.0	5.9	6.6
4	2.5	10.0	10.0	10.0	7.3
5	2.3	9.0	10.0	9.0	8.1

$\epsilon = 0$



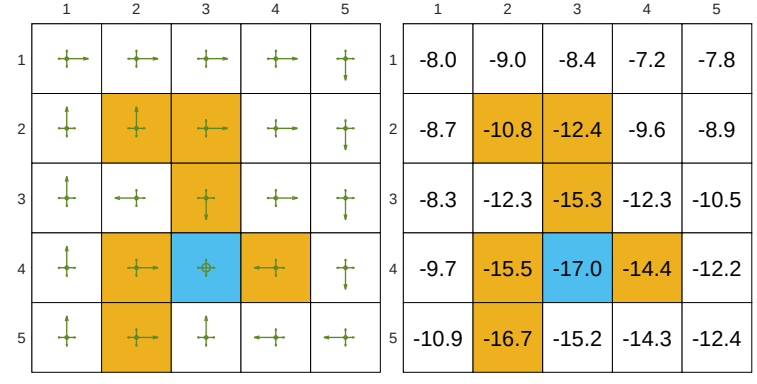
	1	2	3	4	5
1	0.4	0.5	0.9	1.3	1.4
2	0.1	0.0	0.5	1.3	1.7
3	0.1	-0.4	3.4	1.4	1.9
4	-0.1	3.4	3.3	3.7	2.2
5	-0.3	2.6	3.7	3.1	2.7

$\epsilon = 0.1$



	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

$\epsilon = 0.2$



	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

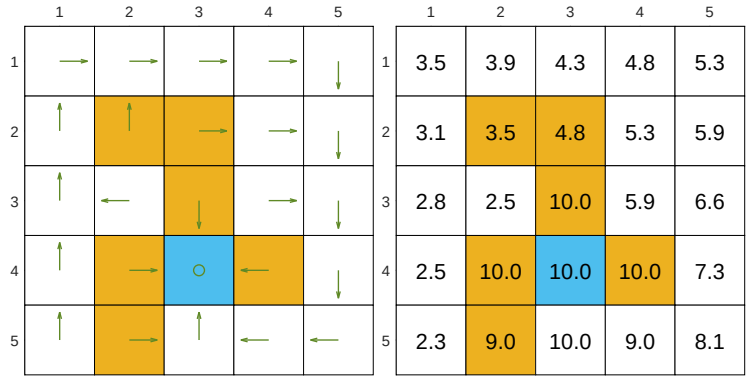
$\epsilon = 0.5$

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

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

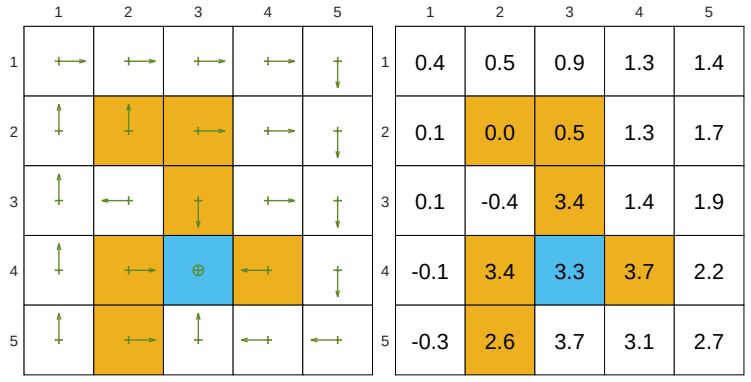
Optimality

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



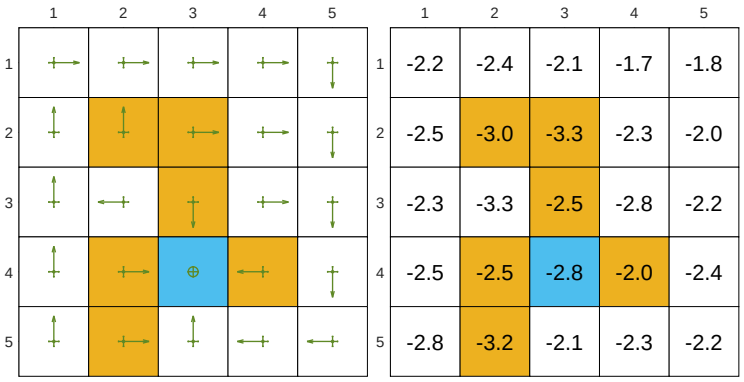
	1	2	3	4	5
1	3.5	3.9	4.3	4.8	5.3
2	3.1	3.5	4.8	5.3	5.9
3	2.8	2.5	10.0	5.9	6.6
4	2.5	10.0	10.0	10.0	7.3
5	2.3	9.0	10.0	9.0	8.1

$\epsilon = 0$



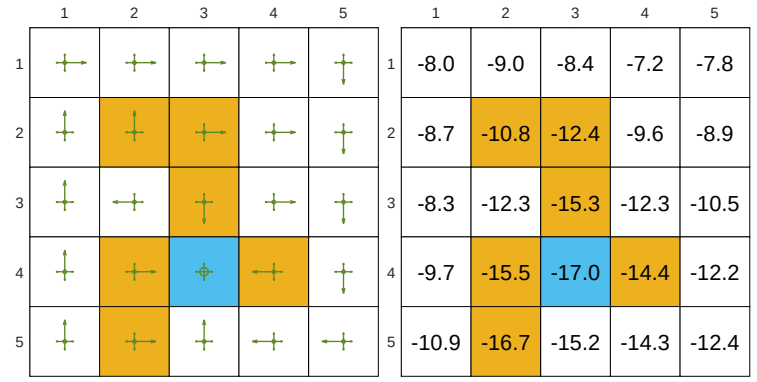
	1	2	3	4	5
1	0.4	0.5	0.9	1.3	1.4
2	0.1	0.0	0.5	1.3	1.7
3	0.1	-0.4	3.4	1.4	1.9
4	-0.1	3.4	3.3	3.7	2.2
5	-0.3	2.6	3.7	3.1	2.7

$\epsilon = 0.1$



	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

$\epsilon = 0.2$



	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

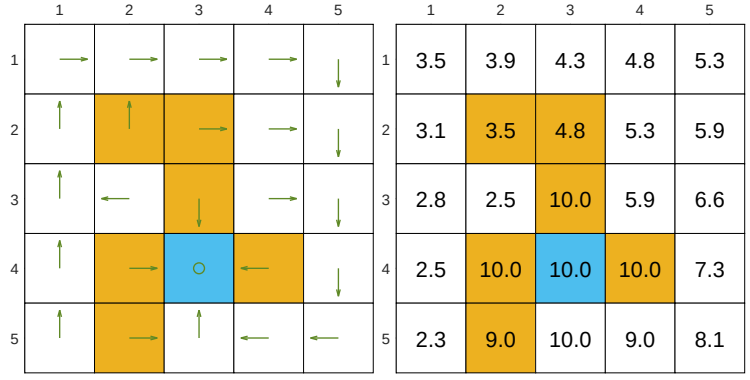
$\epsilon = 0.5$

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

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

Consistency

▷ **Examples: Find the optimal ϵ-greedy policies and their state values.**



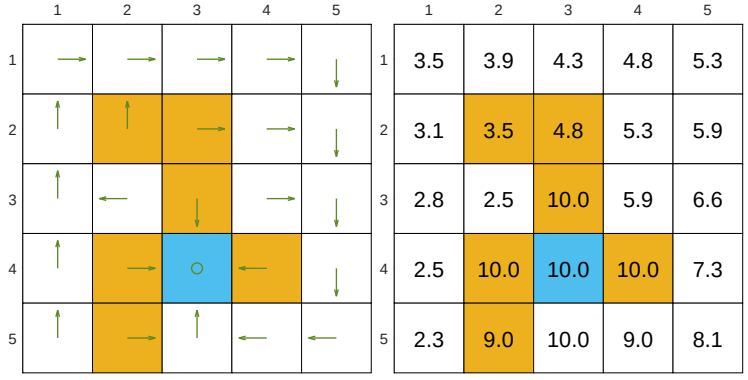
	1	2	3	4	5
1	3.5	3.9	4.3	4.8	5.3
2	3.1	3.5	4.8	5.3	5.9
3	2.8	2.5	10.0	5.9	6.6
4	2.5	10.0	10.0	10.0	7.3
5	2.3	9.0	10.0	9.0	8.1

$$\epsilon = 0$$

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

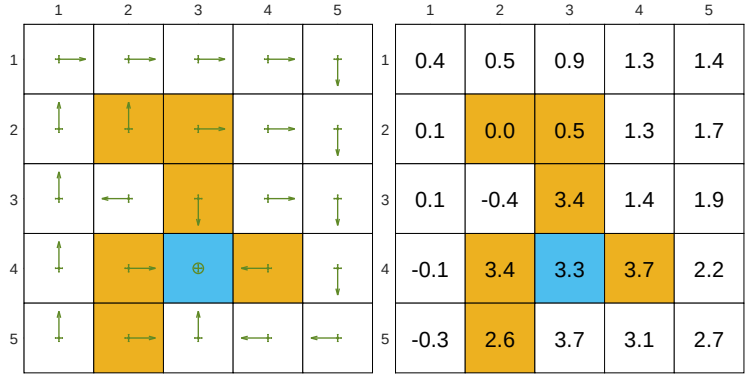
Consistency

▷ **Examples: 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

$\epsilon = 0$



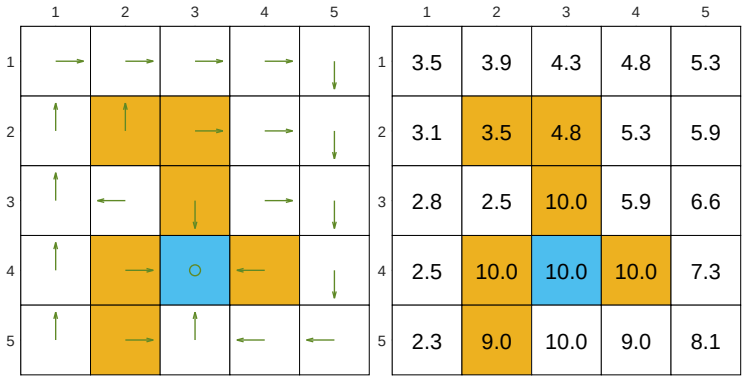
	1	2	3	4	5
1	0.4	0.5	0.9	1.3	1.4
2	0.1	0.0	0.5	1.3	1.7
3	0.1	-0.4	3.4	1.4	1.9
4	-0.1	3.4	3.3	3.7	2.2
5	-0.3	2.6	3.7	3.1	2.7

$\epsilon = 0.1$

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

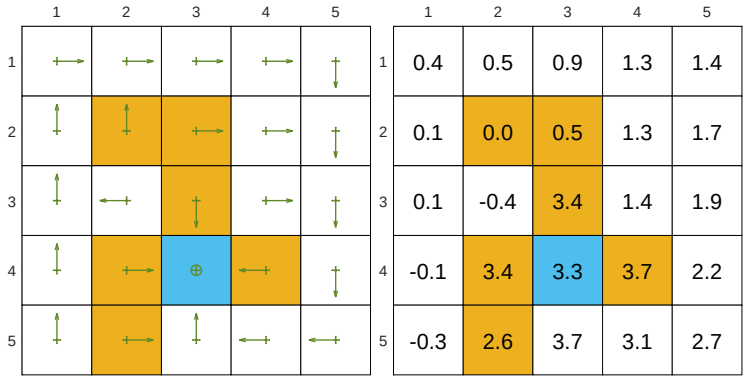
Consistency

▷ **Examples: Find the optimal ϵ-greedy policies and their state values.**



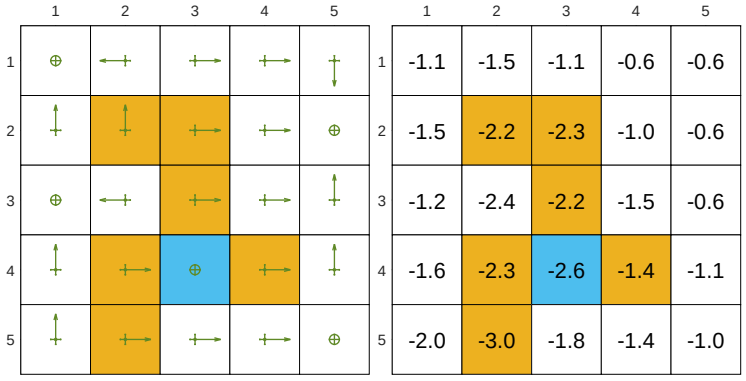
	1	2	3	4	5
1	3.5	3.9	4.3	4.8	5.3
2	3.1	3.5	4.8	5.3	5.9
3	2.8	2.5	10.0	5.9	6.6
4	2.5	10.0	10.0	10.0	7.3
5	2.3	9.0	10.0	9.0	8.1

$\epsilon = 0$



	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

$\epsilon = 0.1$



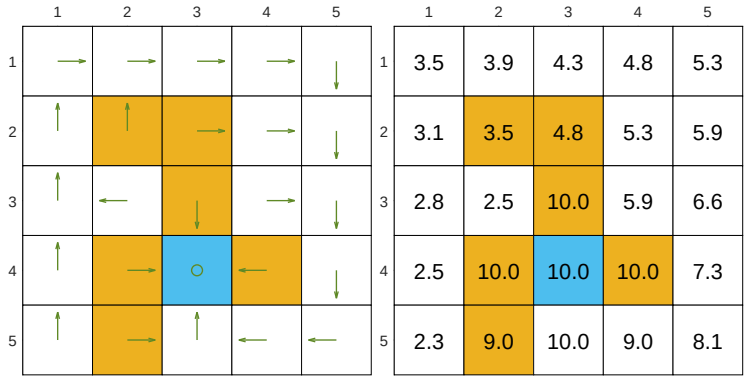
	1	2	3	4	5
1	-1.1	-1.5	-1.1	-0.6	-0.6
2	-1.5	-2.2	-2.3	-1.0	-0.6
3	-1.2	-2.4	-2.2	-1.5	-0.6
4	-1.6	-2.3	-2.6	-1.4	-1.1
5	-2.0	-3.0	-1.8	-1.4	-1.0

$\epsilon = 0.2$

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

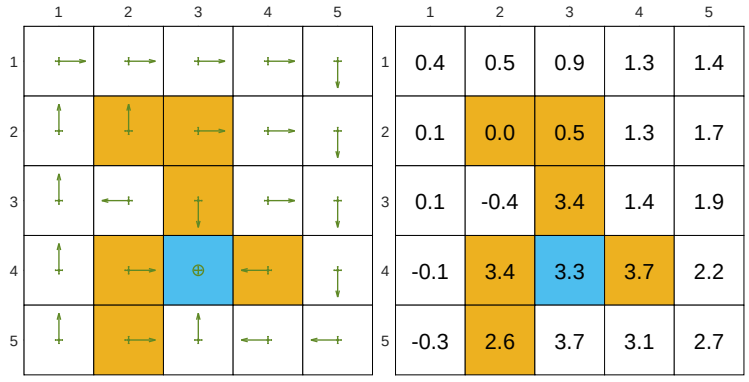
Consistency

▷ **Examples: Find the optimal ϵ-greedy policies and their state values.**



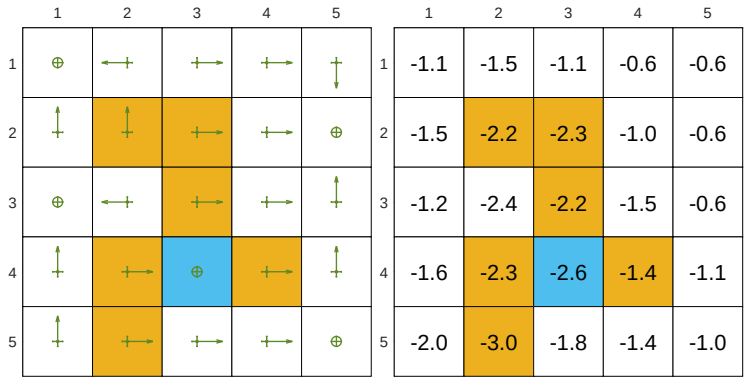
	1	2	3	4	5
1	3.5	3.9	4.3	4.8	5.3
2	3.1	3.5	4.8	5.3	5.9
3	2.8	2.5	10.0	5.9	6.6
4	2.5	10.0	10.0	10.0	7.3
5	2.3	9.0	10.0	9.0	8.1

$\epsilon = 0$



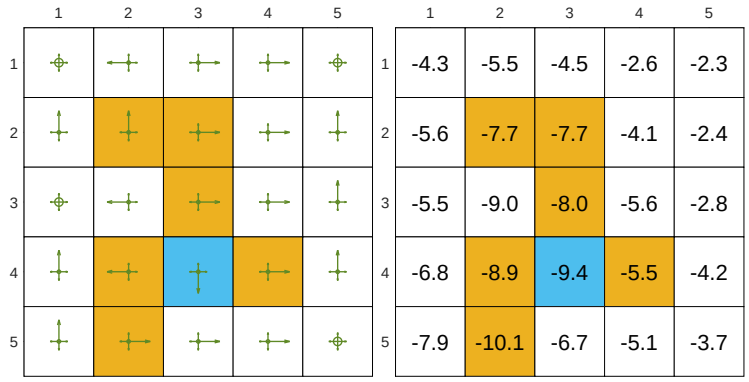
	1	2	3	4	5
1	0.4	0.5	0.9	1.3	1.4
2	0.1	0.0	0.5	1.3	1.7
3	0.1	-0.4	3.4	1.4	1.9
4	-0.1	3.4	3.3	3.7	2.2
5	-0.3	2.6	3.7	3.1	2.7

$\epsilon = 0.1$



	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

$\epsilon = 0.2$



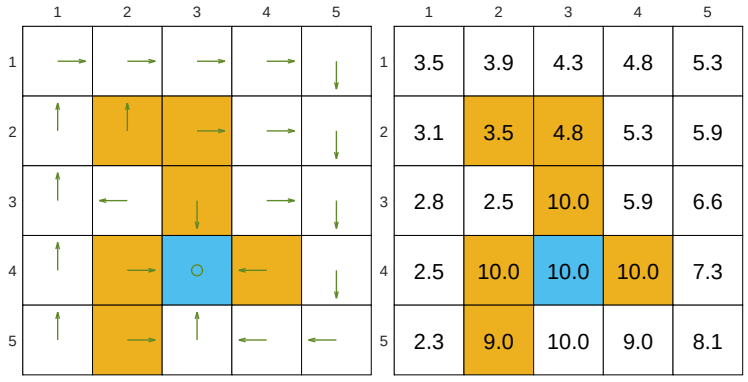
	1	2	3	4	5
1	-4.3	-5.5	-4.5	-2.6	-2.3
2	-5.6	-7.7	-7.7	-4.1	-2.4
3	-5.5	-9.0	-8.0	-5.6	-2.8
4	-6.8	-8.9	-9.4	-5.5	-4.2
5	-7.9	-10.1	-6.7	-5.1	-3.7

$\epsilon = 0.5$

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

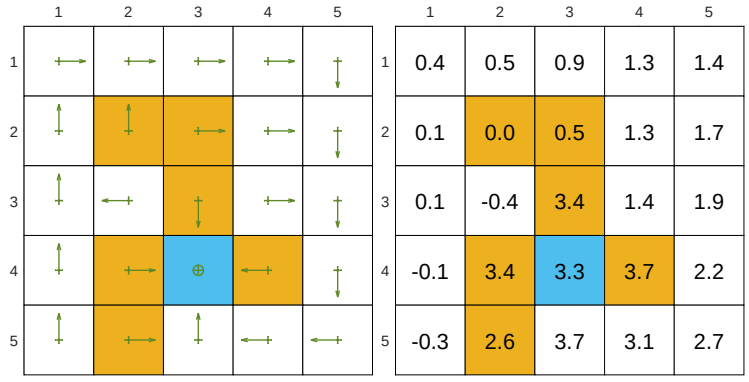
Consistency

▷ **Examples: Find the optimal ϵ-greedy policies and their state values.**



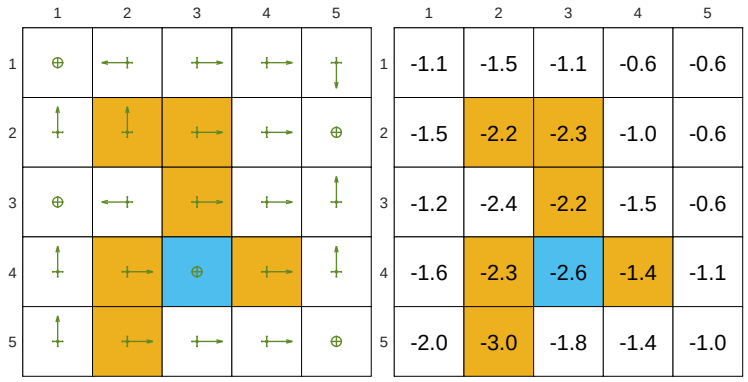
	1	2	3	4	5
1	3.5	3.9	4.3	4.8	5.3
2	3.1	3.5	4.8	5.3	5.9
3	2.8	2.5	10.0	5.9	6.6
4	2.5	10.0	10.0	10.0	7.3
5	2.3	9.0	10.0	9.0	8.1

$\epsilon = 0$



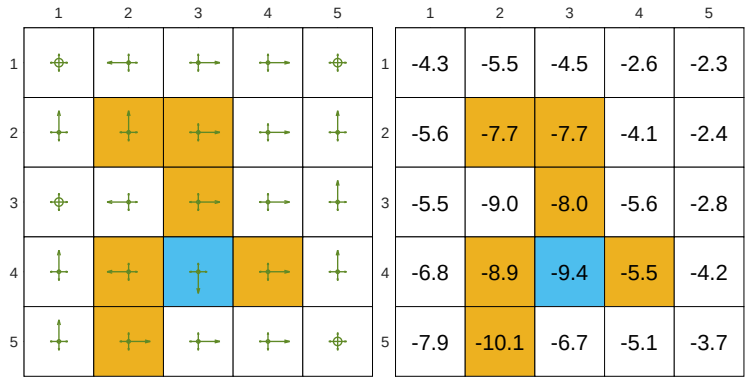
	1	2	3	4	5
1	0.4	0.5	0.9	1.3	1.4
2	0.1	0.0	0.5	1.3	1.7
3	0.1	-0.4	3.4	1.4	1.9
4	-0.1	3.4	3.3	3.7	2.2
5	-0.3	2.6	3.7	3.1	2.7

$\epsilon = 0.1$



	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

$\epsilon = 0.2$



	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

$\epsilon = 0.5$

▷ 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 ES (Exploring Starts)
  - MC ϵ-Greedy
- Relationship among the three algorithms
- Optimality vs exploration of ϵ-greedy policies