Chapter 5

Monte Carlo Methods

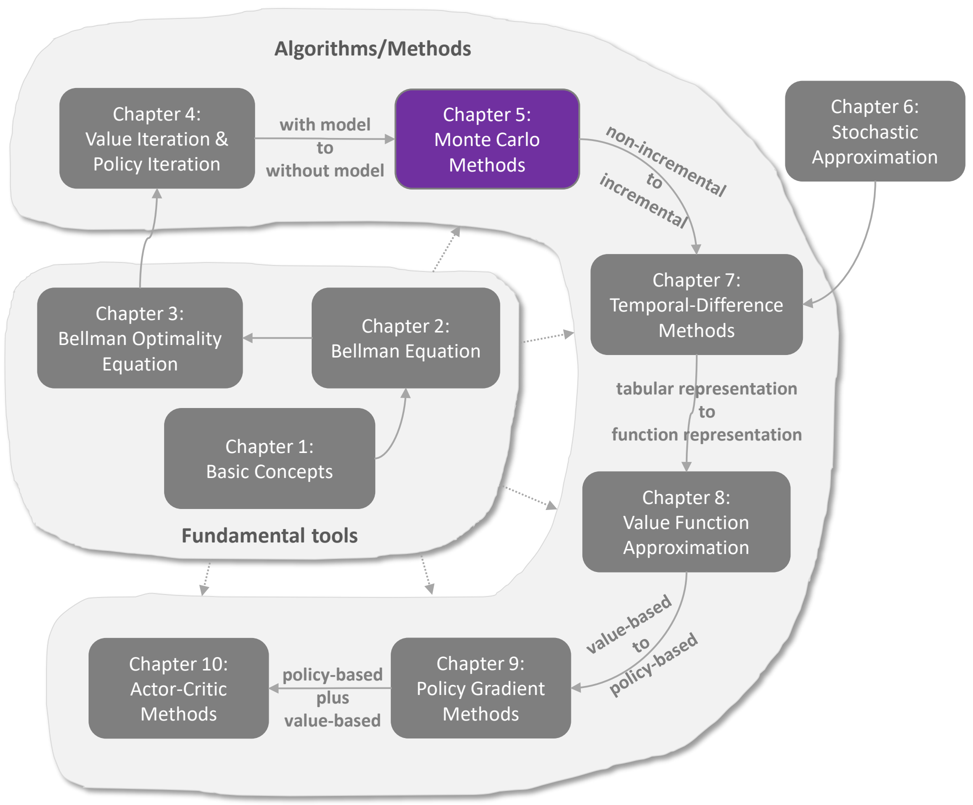

Algorithms/Methods
Chapter 4: Value Iteration & Policy Iteration
with model to without model
Chapter 5: Monte Carlo Methods
non-incremental to incremental
Chapter 6: Stochastic Approximation
Chapter 7: Temporal-Difference Methods
Chapter 3: Bellman Optimality Equation
Chapter 2: Bellman Equation
Chapter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

Figure 5.1 Where we are in this book.

In the previous chapter, we introduced algorithms that can find optimal policies based on the system model. In this chapter, we start introducing *model-free* reinforcement learning algorithms that do not presume system models.

While this is the first time we introduce model-free algorithms in this book, we must fill a knowledge gap: how can we find optimal policies without models? The philosophy is simple: If we do not have a model, we must have some data. If we do not have data, we must have a model. If we have neither, then we are not able to find optimal policies. The concept "data" in reinforcement learning usually refers to the agent's interaction experiences with the environment.

To demonstrate how to learn from data rather than model, we start this chapter by introducing the *mean estimation* problem, where the expected value of a random variable is estimated from some samples. Understanding this problem is crucial for understanding the fundamental idea of *learning from data*.

Then, we introduce three algorithms based on Monte Carlo (MC) methods. These algorithms can learn optimal policies from experience samples. The first and simplest algorithm is called MC Basic, which can be readily obtained by modifying the policy iteration algorithm introduced in the last chapter. Understanding this algorithm is important for grasping the fundamental idea of MC-based reinforcement learning. By extending this algorithm, we further introduce another two algorithms that are more complicated but more efficient.

5.1 Motivating example: Mean estimation

We next introduce the *mean estimation* problem to demonstrate how to learn from data rather than model. We will see that mean estimation can be achieved based on *Monte Carlo* methods, which refer to a broad class of techniques that use stochastic samples to solve estimation problems. Readers may wonder why we care about the mean estimation problem. It is simply because state and action values are both defined as the means of returns. Estimating a state or action value is actually a mean estimation problem.

Consider a random variable X that can take values from a finite set of real numbers denoted as $\mathcal{X}$. Suppose that our task is to calculate the mean or expected value of X: $\mathbb{E}[X]$. The following two approaches can be used to calculate $\mathbb{E}[X]$.

◇ The first approach is *model-based*. Here, the model refers to the probability distribution of X. If the model is known, then the mean can be directly calculated based on the definition of the expected value:

$$\mathbb{E}[X] = \sum_{x\in\mathcal{X}} p(x)x.$$

In this book, we use the terms *expected value*, *mean*, and *average* interchangeably.

◇ The second approach is *model-free*. When the probability distribution (i.e., the model) of X is unknown, suppose that we have some samples $\{x_1, x_2, \cdots, x_n\}$ of X. Then, the mean can be approximated as

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

When n is small, this approximation may not be accurate. However, as n increases, the approximation becomes increasingly accurate. When $n \to \infty$, we have $\bar{x} \to \mathbb{E}[X]$. The *law of large numbers* guarantees this: the average of a large number of samples is close to the expected value. The law of large numbers is introduced in Box 5.1.

The following example illustrates the two approaches described above. Consider a coin-flipping game. Let random variable X denote which side is showing when the coin lands. X has two possible values: $X = 1$ when the head is indicating, and $X = -1$ when the tail is showing. Suppose that the true probability distribution (i.e., the model) of X is

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

If the probability distribution is known in advance, we can directly calculate the mean as

$$\mathbb{E}[X] = 0.5 \cdot 1 + 0.5 \cdot (-1) = 0.$$

If the probability distribution is unknown, we can flip the coin many times and record the sampling results $\{x_i\}_{i=1}^n$. By calculating the average of the samples, we can obtain an estimate of the mean. As shown in Figure 5.2, the estimated mean becomes increasingly accurate as the number of samples increases.

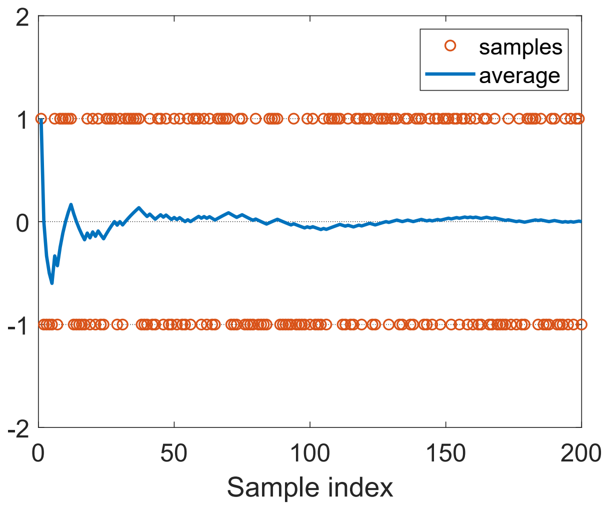


Figure 5.2 An example for demonstrating the law of large numbers. Here, the samples are drawn from $\{+1, -1\}$ following a uniform distribution. The average of the samples gradually converges to zero, which is the true expected value, as the number of samples increases.

It is worth mentioning that the samples used for mean estimation must be *independent and identically distributed* (i.i.d. or iid). Otherwise, if the sampling values correlate, estimating the expected value correctly may be impossible. An extreme case is that all the sampling values are the same as the first one, whatever the first one is. In this case, the average of the samples is always equal to the first sample, no matter how many samples we use.

Box 5.1: Law of large numbers

Suppose that $\{x_i\}_{i=1}^n$ are some i.i.d. samples for a random variable X. Let $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$ be the average of the samples. Then,

$$\mathbb{E}[\bar{x}] = \mathbb{E}[X],$$
$$\text{var}[\bar{x}] = \frac{1}{n}\text{var}[X].$$

The two equations above indicate that $\bar{x}$ is an unbiased estimate of $\mathbb{E}[X]$, and its variance decreases to zero as n increases to infinity.

The proof is given below.

First, $\mathbb{E}[\bar{x}] = \mathbb{E}\left[\sum_{i=1}^n x_i/n\right] = \sum_{i=1}^n \mathbb{E}[x_i]/n = \mathbb{E}[X]$, where the last equability is due to the fact that the samples are *identically distributed* (i.e., $\mathbb{E}[x_i] = \mathbb{E}[X]$).

Second, $\text{var}(\bar{x}) = \text{var}\left[\sum_{i=1}^n x_i/n\right] = \sum_{i=1}^n \text{var}[x_i]/n^2 = (n \cdot \text{var}[X])/n^2 = \text{var}[X]/n$, where the second equality is due to the fact that the samples are *independent*, and the third equability is a result of the samples being *identically distributed* (i.e., $\text{var}[x_i] = \text{var}[X]$).

5.2 MC Basic: The simplest MC-based algorithm

This section introduces the first and the simplest MC-based reinforcement learning algorithm. This algorithm is obtained by replacing the *model-based policy evaluation step* in the policy iteration algorithm introduced in Section 4.2 with a *model-free MC estimation step*.

5.2.1 Converting policy iteration to be model-free

There are two steps in every iteration of the policy iteration algorithm (see Section 4.2). The first step is *policy evaluation*, which aims to compute v_{π_k} by solving $v_{\pi_k} = r_{\pi_k} + \gamma P_{\pi_k} v_{\pi_k}$. The second step is *policy improvement*, which aims to compute the greedy policy $\pi_{k+1} = \arg\max_\pi (r_\pi + \gamma P_\pi v_{\pi_k})$. The elementwise form of the policy improvement step is

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

It must be noted that the action values lie at the *core* of these two steps. Specifically, in the first step, the state values are calculated for the purpose of calculating the action values. In the second step, the new policy is generated based on the calculated action values. Let us reconsider how we can calculate the action values. There are two available approaches.

◇ The first approach is a *model-based* approach. This is the approach adopted by the policy iteration algorithm. In particular, we can first calculate the state value v_{π_k} by solving the Bellman equation. Then, we can calculate the action values by using

$$q_{\pi_k}(s,a) = \sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v_{\pi_k}(s'). \tag{5.1}$$

This approach requires the system model $\{p(r|s,a), p(s'|s,a)\}$ to be known.

◇ The second approach is a *model-free* approach. Recall that the definition of an action value is

$$\begin{aligned} q_{\pi_k}(s,a) &= \mathbb{E}[G_t|S_t=s,A_t=a] \\ &= \mathbb{E}[R_{t+1}+\gamma R_{t+2}+\gamma^2 R_{t+3}+\cdots|S_t=s,A_t=a], \end{aligned}$$

which is the expected return obtained when starting from (s,a). Since $q_{\pi_k}(s,a)$ is an expectation, it can be estimated by MC methods as demonstrated in Section 5.1. To do that, starting from (s,a), the agent can interact with the environment by following policy π_k and then obtain a certain number of episodes. Suppose that there are n episodes and that the return of the ith episode is $g_{\pi_k}^{(i)}(s,a)$. Then, $q_{\pi_k}(s,a)$ can be approximated as

$$q_{\pi_k}(s,a) = \mathbb{E}[G_t|S_t=s,A_t=a] \approx \frac{1}{n}\sum_{i=1}^{n} g_{\pi_k}^{(i)}(s,a). \tag{5.2}$$

We already know that if the number of episodes n is sufficiently large, the approximation will be sufficiently accurate according to the law of large numbers.

The fundamental idea of MC-based reinforcement learning is to use a model-free method for estimating action values, as shown in (5.2), to replace the model-based method in the policy iteration algorithm.

5.2.2 The MC Basic algorithm

We are now ready to present the first MC-based reinforcement learning algorithm. Starting from an initial policy π_0, the kth iteration $(k = 0, 1, 2, \cdots)$ of the algorithm contains two steps.

- *Step* 1: *Policy evaluation.* This step is used to estimate $q_{\pi_k}(s,a)$ for all (s,a). Specifically, for every (s,a), we collect sufficiently many episodes and use the average of the returns, denoted as $q_k(s,a)$, to approximate $q_{\pi_k}(s,a)$.
- *Step* 2: *Policy improvement.* This step solves $\pi_{k+1}(s) = \arg\max_\pi \sum_a \pi(a|s)q_k(s,a)$ 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 is the simplest MC-based reinforcement learning algorithm, which is called *MC Basic* in this book. The pseudocode of the MC Basic algorithm is given in Algorithm 5.1. As can be seen, it is very similar to the policy iteration algorithm. The only difference is that it calculates action values directly from experience samples, whereas policy iteration calculates state values first and then calculates the action values based on the system model. It should be noted that the model-free algorithm directly estimates action values. Otherwise, if it estimates state values instead, we still need to calculate action values from these state values using the system model, as shown in (5.1).

Algorithm 5.1: MC Basic (a model-free variant of policy iteration)

```
Initialization: The initial guess is $\pi_0$.
Goal: Search for an optimal policy.

For the $k$th iteration ($k = 0, 1, 2, \cdots$), do
    For every state $s \in \mathcal{S}$, do
        For every action $a \in \mathcal{A}(s)$, do
            Collect sufficiently many episodes starting from $(s,a)$ by following $\pi_k$
            Policy evaluation:
            $q_{\pi_k}(s,a) \approx q_k(s,a) =$ the 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
```

Since policy iteration is convergent, MC Basic is also convergent given sufficient samples. For every (s,a), suppose that there are sufficiently many episodes starting from (s,a). Then, the average of the returns of these episodes can accurately approximate the action value of (s,a). In practice, we usually do not have sufficient episodes for every (s,a). As a result, the approximation of the action values may not be accurate. Nevertheless, usually the algorithm can still work. This is similar to the truncated policy iteration algorithm, where the action values are neither accurately calculated.

Finally, MC Basic is too simple to be practical due to its low sample efficiency. The reason why we introduce this algorithm is to let readers grasp the core idea of MC-based reinforcement learning. It is essential to understand this algorithm well before studying more complex algorithms introduced later in this chapter. We will see that more complex and sample-efficient algorithms can be readily obtained by extending the MC Basic algorithm.

5.2.3 Illustrative examples

A simple example: A step-by-step implementation

We next use an example to demonstrate the implementation details of the MC Basic algorithm. The reward settings are $r_{\text{boundary}} = r_{\text{forbidden}} = -1$ and $r_{\text{target}} = 1$. The discount rate is $\gamma = 0.9$. The initial policy π_0 is shown in Figure 5.3. This initial policy is not optimal for s_1 or s_3.

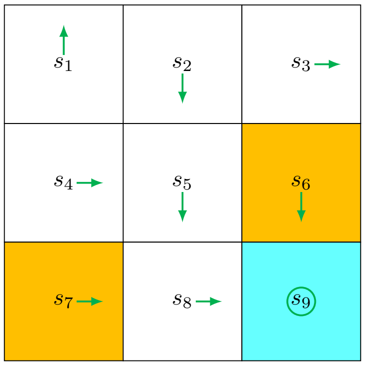


Figure 5.3 An example for illustrating the MC Basic algorithm.

While all the action values should be calculated, we merely present those of s_1 due to space limitations. At s_1, there are five possible actions. For each action, we need to collect many episodes that are sufficiently long to approximate the action value effectively. However, since this example is deterministic regarding the policy and model, running multiple times would generate the same trajectory. As a result, the estimation of each action value merely requires a single episode.

Following π_0, we can obtain the following episodes by respectively starting from (s_1, a_1), (s_1, a_2), $\cdots$, (s_1, a_5).

◇ Starting from (s_1, a_1), the episode is $s_1 \xrightarrow{a_1} s_1 \xrightarrow{a_1} s_1 \xrightarrow{a_1} \cdots$. The action value equals the discounted return of the episode:

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

◇ Starting from (s_1, a_2), the episode is $s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_3} \cdots$. The action value equals the discounted return of the episode:

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

◇ Starting from (s_1, a_3), the episode is $s_1 \xrightarrow{a_3} s_4 \xrightarrow{a_2} s_5 \xrightarrow{a_3} \cdots$. The action value equals the discounted return of the episode:

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

◇ Starting from (s_1, a_4), the episode is $s_1 \xrightarrow{a_4} s_1 \xrightarrow{a_1} s_1 \xrightarrow{a_1} \cdots$. The action value equals the discounted return of the episode:

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

◇ Starting from (s_1, a_5), the episode is $s_1 \xrightarrow{a_5} s_1 \xrightarrow{a_1} s_1 \xrightarrow{a_1} \cdots$. The action value equals the discounted return of the episode:

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

By comparing the five action values, we see that

$$q_{\pi_0}(s_1, a_2) = q_{\pi_0}(s_1, a_3) = \frac{\gamma^3}{1-\gamma} > 0.$$

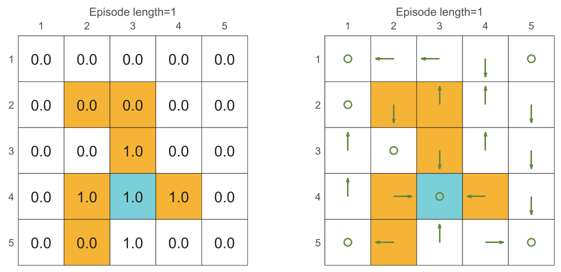
are the maximum values. As a result, the new policy can be obtained as

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

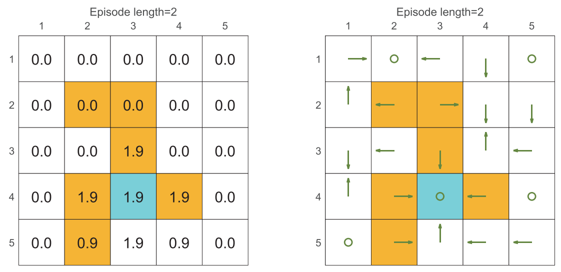
It is intuitive that the improved policy, which takes either a_2 or a_3 at s_1, is optimal. Therefore, we can successfully obtain an optimal policy by using merely one iteration for this simple example. In this simple example, the initial policy is already optimal for all the states except s_1 and s_3. Therefore, the policy can become optimal after merely a single iteration. When the policy is nonoptimal for other states, more iterations are needed.

A comprehensive example: Episode length and sparse rewards

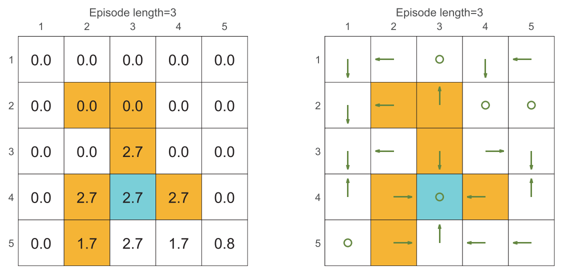
We next discuss some interesting properties of the MC Basic algorithm by examining a more comprehensive example. The example is a 5-by-5 grid world (Figure 5.4). The reward settings are $r_{\text{boundary}} = -1$, $r_{\text{forbidden}} = -10$, and $r_{\text{target}} = 1$. The discount rate is $\gamma = 0.9$.



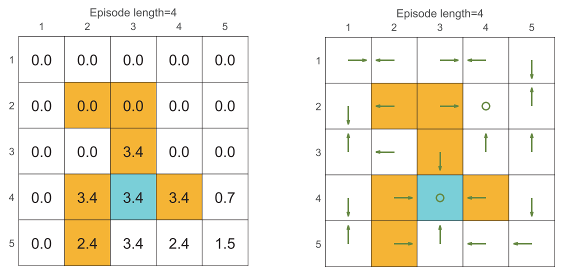
(a) Final value and policy with episode length=1



(b) Final value and policy with episode length=2

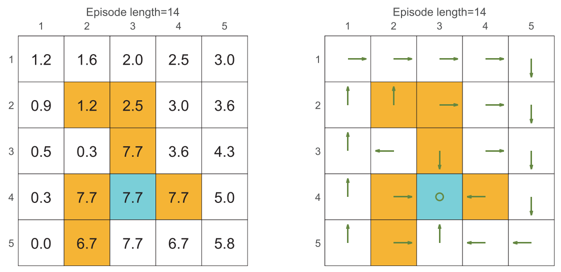


(c) Final value and policy with episode length=3

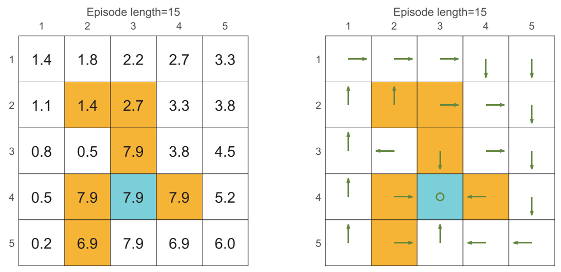


(d) Final value and policy with episode length=4

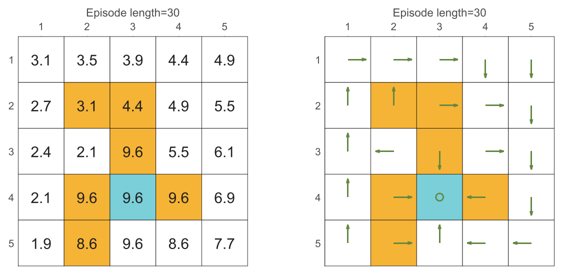
$\vdots$ $\vdots$ $\vdots$



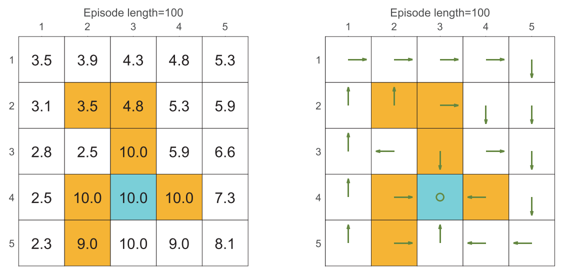
(e) Final value and policy with episode length=14



(f) Final value and policy with episode length=15



(g) Final value and policy with episode length=30



(h) Final value and policy with episode length=100

Figure 5.4 The policies and state values obtained by the MC Basic algorithm when given different episode lengths. Only if the length of each episode is sufficiently long, can the state values be accurately estimated.