Chapter 10

Actor-Critic Methods

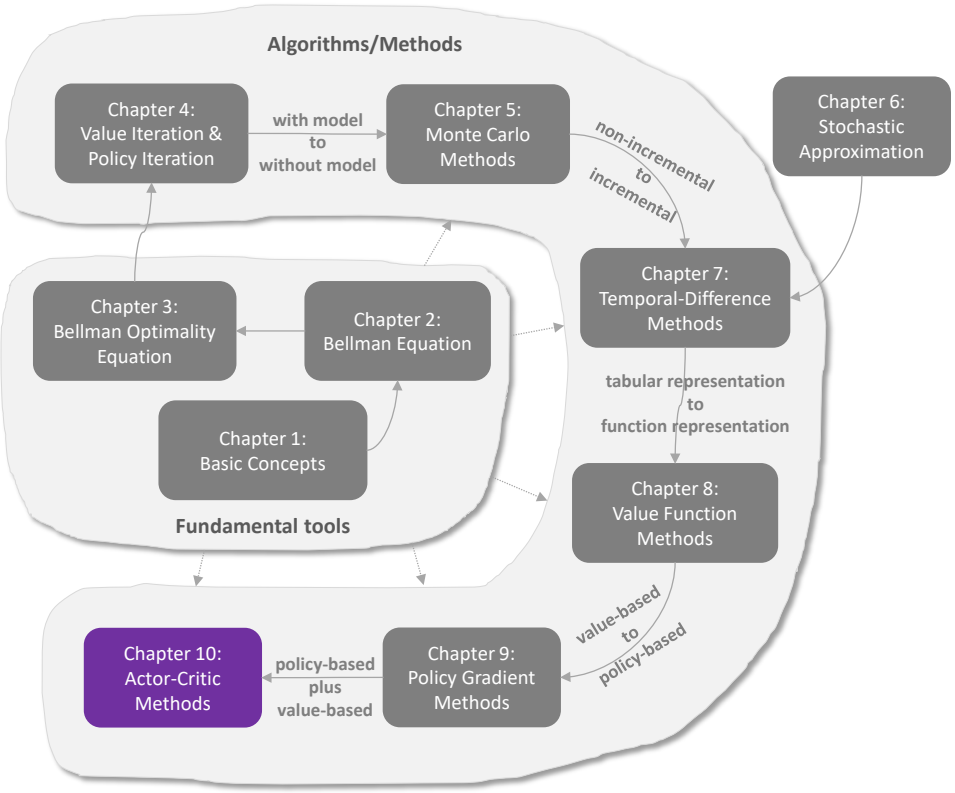

Figure 10.1: Where we are in this book.

This chapter introduces actor-critic methods. From one point of view, "actor-critic" refers to a structure that incorporates both policy-based and value-based methods. Here, an "actor" refers to a policy update step. The reason that it is called an actor is that the actions are taken by following the policy. Here, an "critic" refers to a value update step. It is called a critic because it criticizes the actor by evaluating its corresponding values. From another point of view, actor-critic methods are still policy gradient algorithms. They can be obtained by extending the policy gradient algorithm introduced in Chapter 9. It is important for the reader to well understand the contents of Chapters 8 and 9 before studying this chapter.

10.1 The simplest actor-critic algorithm (QAC)

This section introduces the simplest actor-critic algorithm. This algorithm can be easily obtained by extending the policy gradient algorithm in (9.32).

Recall that the idea of the policy gradient method is to search for an optimal policy by maximizing a scalar metric $J(\theta)$. The gradient-ascent algorithm for maximizing $J(\theta)$ is

$$\begin{aligned}\theta_{t+1} &= \theta_t + \alpha \nabla_\theta J(\theta_t) \\ &= \theta_t + \alpha \mathbb{E}_{S\sim\eta, A\sim\pi}\Big[\nabla_\theta \ln \pi(A|S,\theta_t) q_\pi(S,A)\Big],\end{aligned} \tag{10.1}$$

where η is a distribution of the states (see Theorem 9.1 for more information). Since the true gradient is unknown, we can use a stochastic gradient to approximate it:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \ln \pi(a_t|s_t,\theta_t) q_t(s_t,a_t). \tag{10.2}$$

This is the algorithm given in (9.32).

Equation (10.2) is important because it clearly shows how policy-based and value-based methods can be combined. On the one hand, it is a *policy-based* algorithm since it directly updates the policy parameter. On the other hand, this equation requires knowing $q_t(s_t,a_t)$, which is an estimate of the action value $q_\pi(s_t,a_t)$. As a result, another *value-based* algorithm is required to generate $q_t(s_t,a_t)$. So far, we have studied two ways to estimate action values in this book. The first is based on Monte Carlo learning and the second is temporal-difference (TD) learning.

- ◇ If $q_t(s_t,a_t)$ is estimated by Monte Carlo learning, the corresponding algorithm is called *REINFORCE* or *Monte Carlo policy gradient*, which has already been introduced in Chapter 9.
- ◇ If $q_t(s_t,a_t)$ is estimated by TD learning, the corresponding algorithms are usually called *actor-critic*. Therefore, actor-critic methods can be obtained by incorporating TD-based value estimation into policy gradient methods.

The procedure of the simplest actor-critic algorithm is summarized in Algorithm 10.1. The *critic* corresponds to the value update step via the Sarsa algorithm presented in (8.35). The action values are represented by a parameterized function $q(s,a,w)$. The *actor* corresponds to the policy update step in (10.2). This actor-citric algorithm is sometimes called *Q actor-critic* (QAC). Although it is simple, QAC reveals the core idea of actor-critic methods. It can be extended to generate many advanced ones as shown in the rest of this chapter.

Algorithm 10.1: The simplest actor-critic algorithm (QAC)

Initialization: A policy function $\pi(a|s,\theta_0)$ where θ_0 is the initial parameter. A value function $q(s,a,w_0)$ where w_0 is the initial parameter. $\alpha_w, \alpha_\theta > 0$.
Goal: Learn an optimal policy to maximize $J(\theta)$.

At time step t in each episode, do

Generate a_t following $\pi(a|s_t,\theta_t)$, observe r_{t+1}, s_{t+1}, and then generate a_{t+1} following $\pi(a|s_{t+1},\theta_t)$.

Actor (policy update):

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_\theta \ln \pi(a_t|s_t,\theta_t) q(s_t,a_t,w_t)$$

Critic (value update):

$$w_{t+1} = w_t + \alpha_w \big[r_{t+1} + \gamma q(s_{t+1},a_{t+1},w_t) - q(s_t,a_t,w_t)\big] \nabla_w q(s_t,a_t,w_t)$$

10.2 Advantage actor-critic (A2C)

We now introduce the algorithm of *advantage actor-critic*. The core idea of this algorithm is to introduce a baseline to reduce estimation variance.

10.2.1 Baseline invariance

One interesting property of the policy gradient is that it is invariant to an additional *baseline*. That is

$$\mathbb{E}_{S\sim\eta, A\sim\pi}\Big[\nabla_\theta \ln \pi(A|S,\theta_t) q_\pi(S,A)\Big] = \mathbb{E}_{S\sim\eta, A\sim\pi}\Big[\nabla_\theta \ln \pi(A|S,\theta_t)(q_\pi(S,A) - b(S))\Big], \tag{10.3}$$

where the additional baseline $b(S)$ is a scalar function of S. We next answer two questions about the baseline.

◇ First, why is (10.3) valid?

Equation (10.3) holds if and only if

$$\mathbb{E}_{S\sim\eta, A\sim\pi}\Big[\nabla_\theta \ln \pi(A|S,\theta_t) b(S)\Big] = 0.$$

This equation is valid because

$$\begin{aligned}\mathbb{E}_{S\sim\eta,A\sim\pi}\Big[\nabla_\theta \ln\pi(A|S,\theta_t)b(S)\Big] &= \sum_{s\in\mathcal{S}}\eta(s)\sum_{a\in\mathcal{A}}\pi(a|s,\theta_t)\nabla_\theta\ln\pi(a|s,\theta_t)b(s)\\ &= \sum_{s\in\mathcal{S}}\eta(s)\sum_{a\in\mathcal{A}}\nabla_\theta\pi(a|s,\theta_t)b(s)\\ &= \sum_{s\in\mathcal{S}}\eta(s)b(s)\sum_{a\in\mathcal{A}}\nabla_\theta\pi(a|s,\theta_t)\\ &= \sum_{s\in\mathcal{S}}\eta(s)b(s)\nabla_\theta\sum_{a\in\mathcal{A}}\pi(a|s,\theta_t)\\ &= \sum_{s\in\mathcal{S}}\eta(s)b(s)\nabla_\theta 1 = 0.\end{aligned}$$

◇ Second, why is the baseline useful?

The baseline is useful because it can reduce the approximation variance when we use samples to approximate the true gradient. In particular, let

$$X(S,A) \doteq \nabla_\theta \ln\pi(A|S,\theta_t)[q_\pi(S,A) - b(S)]. \tag{10.4}$$

Then, the true gradient is $\mathbb{E}[X(S,A)]$. Since we need to use a stochastic sample x to approximate $\mathbb{E}[X]$, it would be favorable if the variance $\text{var}(X)$ is small. For example, if $\text{var}(X)$ is close to zero, then any sample x can accurately approximate $\mathbb{E}[X]$. On the contrary, if $\text{var}(X)$ is large, the value of a sample may be far from $\mathbb{E}[X]$.

Although $\mathbb{E}[X]$ is invariant to the baseline, the variance $\text{var}(X)$ is *not*. Our goal is to design a good baseline to minimize $\text{var}(X)$. In the algorithms of REINFORCE and QAC, we set $b = 0$, which is not guaranteed to be a good baseline.

In fact, the optimal baseline that minimizes $\text{var}(X)$ is

$$b^*(s) = \frac{\mathbb{E}_{A\sim\pi}\big[\|\nabla_\theta\ln\pi(A|s,\theta_t)\|^2 q_\pi(s,A)\big]}{\mathbb{E}_{A\sim\pi}\big[\|\nabla_\theta\ln\pi(A|s,\theta_t)\|^2\big]}, \quad s\in\mathcal{S}. \tag{10.5}$$

The proof is given in Box 10.1.

Although the baseline in (10.5) is optimal, it is too complex to be useful in practice. If the weight $\|\nabla_\theta\ln\pi(A|s,\theta_t)\|^2$ is removed from (10.5), we can obtain a suboptimal baseline that has a concise expression:

$$b^\dagger(s) = \mathbb{E}_{A\sim\pi}[q_\pi(s,A)] = v_\pi(s), \quad s\in\mathcal{S}.$$

Interestingly, this suboptimal baseline is the state value.

Box 10.1: Showing that $b^*(s)$ in (10.5) is the optimal baseline

Let $\bar{x} \doteq \mathbb{E}[X]$, which is invariant for any $b(s)$. If X is a vector, its variance is a matrix. It is common to select the trace of $\text{var}(X)$ as a scalar objective function for optimization:

$$\begin{aligned}\text{tr}[\text{var}(X)] &= \text{tr}\mathbb{E}[(X-\bar{x})(X-\bar{x})^T] \\ &= \text{tr}\mathbb{E}[XX^T - \bar{x}X^T - X\bar{x}^T + \bar{x}\bar{x}^T] \\ &= \mathbb{E}[X^TX - X^T\bar{x} - \bar{x}^TX + \bar{x}^T\bar{x}] \\ &= \mathbb{E}[X^TX] - \bar{x}^T\bar{x}. \end{aligned} \tag{10.6}$$

When deriving the above equation, we use the trace property $\text{tr}(AB) = \text{tr}(BA)$ for any squared matrices A, B with appropriate dimensions. Since $\bar{x}$ is invariant, equation (10.6) suggests that we only need to minimize $\mathbb{E}[X^TX]$. With X defined in (10.4), we have

$$\begin{aligned}\mathbb{E}[X^TX] &= \mathbb{E}\left[(\nabla_\theta \ln\pi)^T(\nabla_\theta \ln\pi)(q_\pi(S,A) - b(S))^2\right] \\ &= \mathbb{E}\left[\|\nabla_\theta \ln\pi\|^2(q_\pi(S,A) - b(S))^2\right],\end{aligned}$$

where $\pi(A|S,\theta)$ is written as π for short. Since $S \sim \eta$ and $A \sim \pi$, the above equation can be rewritten as

$$\mathbb{E}[X^TX] = \sum_{s\in\mathcal{S}} \eta(s)\mathbb{E}_{A\sim\pi}\left[\|\nabla_\theta \ln\pi\|^2(q_\pi(s,A) - b(s))^2\right].$$

To ensure $\nabla_b\mathbb{E}[X^TX] = 0$, $b(s)$ for any $s \in \mathcal{S}$ should satisfy

$$\mathbb{E}_{A\sim\pi}\left[\|\nabla_\theta \ln\pi\|^2(b(s) - q_\pi(s,A))\right] = 0, \qquad s \in \mathcal{S}.$$

The above equation can be easily solved to obtain the optimal baseline:

$$b^*(s) = \frac{\mathbb{E}_{A\sim\pi}[\|\nabla_\theta \ln\pi\|^2 q_\pi(s,A)]}{\mathbb{E}_{A\sim\pi}[\|\nabla_\theta \ln\pi\|^2]}, \qquad s \in \mathcal{S}.$$

More discussions on optimal baselines in policy gradient methods can be found in [69, 70].

10.2.2 Algorithm description

When $b(s) = v_\pi(s)$, the gradient-ascent algorithm in (10.1) becomes

$$\begin{aligned}\theta_{t+1} &= \theta_t + \alpha \mathbb{E}\Big[\nabla_\theta \ln \pi(A|S,\theta_t)[q_\pi(S,A) - v_\pi(S)]\Big] \\ &\doteq \theta_t + \alpha \mathbb{E}\Big[\nabla_\theta \ln \pi(A|S,\theta_t)\delta_\pi(S,A)\Big]. \end{aligned} \tag{10.7}$$

Here,

$$\delta_\pi(S,A) \doteq q_\pi(S,A) - v_\pi(S)$$

is called the *advantage function*, which reflects the advantage of one action over the others. More specifically, note that $v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s) q_\pi(s,a)$ is the mean of the action values. If $\delta_\pi(s,a) > 0$, it means that the corresponding action has a greater value than the mean value.

The stochastic version of (10.7) is

$$\begin{aligned}\theta_{t+1} &= \theta_t + \alpha \nabla_\theta \ln \pi(a_t|s_t,\theta_t)[q_t(s_t,a_t) - v_t(s_t)] \\ &= \theta_t + \alpha \nabla_\theta \ln \pi(a_t|s_t,\theta_t)\delta_t(s_t,a_t), \end{aligned} \tag{10.8}$$

where s_t, a_t are samples of S, A at time t. Here, $q_t(s_t,a_t)$ and $v_t(s_t)$ are approximations of $q_{\pi(\theta_t)}(s_t,a_t)$ and $v_{\pi(\theta_t)}(s_t)$, respectively. The algorithm in (10.8) updates the policy based on the *relative value* of q_t with respect to v_t rather than the *absolute value* of q_t. This is intuitively reasonable because, when we attempt to select an action at a state, we only care about which action has the greatest value *relative* to the others.

If $q_t(s_t,a_t)$ and $v_t(s_t)$ are estimated by Monte Carlo learning, the algorithm in (10.8) is called *REINFORCE with a baseline*. If $q_t(s_t,a_t)$ and $v_t(s_t)$ are estimated by TD learning, the algorithm is usually called *advantage actor-critic* (A2C). The implementation of A2C is summarized in Algorithm 10.2. It should be noted that the advantage function in this implementation is approximated by the TD error:

$$q_t(s_t,a_t) - v_t(s_t) \approx r_{t+1} + \gamma v_t(s_{t+1}) - v_t(s_t).$$

This approximation is reasonable because

$$q_\pi(s_t,a_t) - v_\pi(s_t) = \mathbb{E}\Big[R_{t+1} + \gamma v_\pi(S_{t+1}) - v_\pi(S_t)|S_t = s_t, A_t = a_t\Big],$$

which is valid due to the definition of $q_\pi(s_t,a_t)$. One merit of using the TD error is that we only need to use a single neural network to represent $v_\pi(s)$. Otherwise, if $\delta_t = q_t(s_t,a_t) - v_t(s_t)$, we need to maintain two networks to represent $v_\pi(s)$ and $q_\pi(s,a)$, respectively. When we use the TD error, the algorithm may also be called *TD actor-critic*. In addition, it is notable that the policy $\pi(\theta_t)$ is stochastic and hence exploratory. Therefore, it can be directly used to generate experience samples without relying on

Algorithm 10.2: Advantage actor-critic (A2C) or TD actor-critic

Initialization: A policy function $\pi(a|s, \theta_0)$ where θ_0 is the initial parameter. A value function $v(s, w_0)$ where w_0 is the initial parameter. $\alpha_w, \alpha_\theta > 0$.
Goal: Learn an optimal policy to maximize $J(\theta)$.

At time step t in each episode, do
Generate a_t following $\pi(a|s_t, \theta_t)$ and then observe r_{t+1}, s_{t+1}.
Advantage (TD error):
$\delta_t = r_{t+1} + \gamma v(s_{t+1}, w_t) - v(s_t, w_t)$
Actor (policy update):
$\theta_{t+1} = \theta_t + \alpha_\theta \delta_t \nabla_\theta \ln \pi(a_t|s_t, \theta_t)$
Critic (value update):
$w_{t+1} = w_t + \alpha_w \delta_t \nabla_w v(s_t, w_t)$

techniques such as ε-greedy. There are some variants of A2C such as *asynchronous advantage actor-critic* (A3C). Interested readers may check [71, 72].

10.3 Off-policy actor-critic

The policy gradient methods that we have studied so far, including REINFORCE, QAC, and A2C, are all *on-policy*. The reason for this can be seen from the expression of the true gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}_{S\sim\eta, A\sim\pi}\Big[\nabla_\theta \ln \pi(A|S, \theta_t)(q_\pi(S, A) - v_\pi(S))\Big].$$

To use samples to approximate this true gradient, we must generate the action samples by following $\pi(\theta)$. Hence, $\pi(\theta)$ is the behavior policy. Since $\pi(\theta)$ is also the target policy that we aim to improve, the policy gradient methods are on-policy.

In the case that we already have some samples generated by a given behavior policy, the policy gradient methods can still be applied to utilize these samples. To do that, we can employ a technique called *importance sampling*. It is worth mentioning that the importance sampling technique is not restricted to the field of reinforcement learning. It is a general technique for estimating expected values defined over one probability distribution using some samples drawn from another distribution.

10.3.1 Importance sampling

We next introduce the importance sampling technique. Consider a random variable $X \in \mathcal{X}$. Suppose that $p_0(X)$ is a probability distribution. Our goal is to estimate $\mathbb{E}_{X\sim p_0}[X]$. Suppose that we have some i.i.d. samples $\{x_i\}_{i=1}^n$.

- ◇ First, if the samples $\{x_i\}_{i=1}^n$ are generated by following p_0, then the average value $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$ can be used to approximate $\mathbb{E}_{X\sim p_0}[X]$ because $\bar{x}$ is an unbiased estimate of $\mathbb{E}_{X\sim p_0}[X]$ and the estimation variance converges to zero as $n \to \infty$ (see the law of large numbers in Box 5.1 for more information).
- ◇ Second, consider a new scenario where the samples $\{x_i\}_{i=1}^n$ are *not* generated by p_0. Instead, they are generated by another distribution p_1. Can we still use these samples to approximate $\mathbb{E}_{X\sim p_0}[X]$? The answer is yes. However, we can no longer use $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$ to approximate $\mathbb{E}_{X\sim p_0}[X]$ since $\bar{x} \approx \mathbb{E}_{X\sim p_1}[X]$ rather than $\mathbb{E}_{X\sim p_0}[X]$.

In the second scenario, $\mathbb{E}_{X\sim p_0}[X]$ can be approximated based on the *importance sampling* technique. In particular, $\mathbb{E}_{X\sim p_0}[X]$ satisfies

$$\mathbb{E}_{X\sim p_0}[X] = \sum_{x\in\mathcal{X}} p_0(x)x = \sum_{x\in\mathcal{X}} p_1(x)\underbrace{\frac{p_0(x)}{p_1(x)}}_{f(x)}x = \mathbb{E}_{X\sim p_1}[f(X)]. \tag{10.9}$$

Thus, estimating $\mathbb{E}_{X\sim p_0}[X]$ becomes the problem of estimating $\mathbb{E}_{X\sim p_1}[f(X)]$. Let

$$\bar{f} \doteq \frac{1}{n}\sum_{i=1}^n f(x_i).$$

Since $\bar{f}$ can effectively approximate $\mathbb{E}_{X\sim p_1}[f(X)]$, it then follows from (10.9) that

$$\mathbb{E}_{X\sim p_0}[X] = \mathbb{E}_{X\sim p_1}[f(X)] \approx \bar{f} = \frac{1}{n}\sum_{i=1}^n f(x_i) = \frac{1}{n}\sum_{i=1}^n \underbrace{\frac{p_0(x_i)}{p_1(x_i)}}_{\substack{\text{importance}\\ \text{weight}}} x_i. \tag{10.10}$$

Equation (10.10) suggests that $\mathbb{E}_{X\sim p_0}[X]$ can be approximated by a weighted average of x_i. Here, $\frac{p_0(x_i)}{p_1(x_i)}$ is called the *importance weight*. When $p_1 = p_0$, the importance weight is 1 and $\bar{f}$ becomes $\bar{x}$. When $p_0(x_i) \geq p_1(x_i)$, x_i can be sampled more frequently by p_0 but less frequently by p_1. In this case, the importance weight, which is greater than one, emphasizes the importance of this sample.

Some readers may ask the following question: while $p_0(x)$ is required in (10.10), why do we not directly calculate $\mathbb{E}_{X\sim p_0}[X]$ using its definition $\mathbb{E}_{X\sim p_0}[X] = \sum_{x\in\mathcal{X}} p_0(x)x$? The answer is as follows. To use the definition, we need to know either the *analytical* expression of p_0 or the value of $p_0(x)$ for *every* $x \in \mathcal{X}$. However, it is difficult to obtain the *analytical* expression of p_0 when the distribution is represented by, for example, a neural network. It is also difficult to obtain the value of $p_0(x)$ for *every* $x \in \mathcal{X}$ when $\mathcal{X}$ is large. By contrast, (10.10) merely requires the values of $p_0(x_i)$ for some samples and is much easier to implement in practice.

An illustrative example

We next present an example to demonstrate the importance sampling technique. Consider $X \in \mathcal{X} \doteq \{+1, -1\}$. Suppose that p_0 is a probability distribution satisfying

$$p_0(X = +1) = 0.5, \quad p_0(X = -1) = 0.5.$$

The expectation of X over p_0 is

$$\mathbb{E}_{X \sim p_0}[X] = (+1) \cdot 0.5 + (-1) \cdot 0.5 = 0.$$

Suppose that p_1 is another distribution satisfying

$$p_1(X = +1) = 0.8, \quad p_1(X = -1) = 0.2.$$

The expectation of X over p_1 is

$$\mathbb{E}_{X \sim p_1}[X] = (+1) \cdot 0.8 + (-1) \cdot 0.2 = 0.6.$$

Suppose that we have some samples $\{x_i\}$ drawn over p_1. Our goal is to estimate $\mathbb{E}_{X \sim p_0}[X]$ using these samples. As shown in Figure 10.2, there are more samples of $+1$ than -1. That is because $p_1(X = +1) = 0.8 > p_1(X = -1) = 0.2$. If we directly calculate the average value $\sum_{i=1}^{n} x_i / n$ of the samples, this value converges to $\mathbb{E}_{X \sim p_1}[X] = 0.6$ (see the dotted line in Figure 10.2). By contrast, if we calculate the weighted average value as in (10.10), this value can successfully converge to $\mathbb{E}_{X \sim p_0}[X] = 0$ (see the solid line in Figure 10.2).

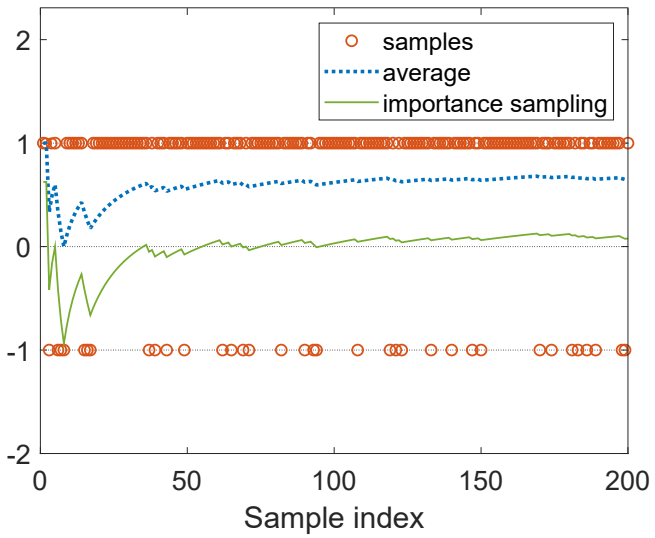

Figure 10.2: An example for demonstrating the importance sampling technique. Here, $X \in \{+1, -1\}$ and $p_0(X = +1) = p_0(X = -1) = 0.5$. The samples are generated according to p_1 where $p_1(X = +1) = 0.8$ and $p_1(X = -1) = 0.2$. The average of the samples converges to $E_{X \sim p_1}[X] = 0.6$, but the weighted average calculated by the importance sampling technique in (10.10) converges to $E_{X \sim p_0}[X] = 0$.

Finally, the distribution p_1, which is used to generate samples, must satisfy that $p_1(x) \neq 0$ when $p_0(x) \neq 0$. If $p_1(x) = 0$ while $p_0(x) \neq 0$, the estimation result may be problematic. For example, if

$$p_1(X = +1) = 1, \quad p_1(X = -1) = 0,$$

then the samples generated by p_1 are all positive: $\{x_i\} = \{+1, +1, \ldots, +1\}$. These samples cannot be used to correctly estimate $\mathbb{E}_{X\sim p_0}[X] = 0$ because

$$\frac{1}{n}\sum_{i=1}^{n}\frac{p_0(x_i)}{p_1(x_i)}x_i = \frac{1}{n}\sum_{i=1}^{n}\frac{p_0(+1)}{p_1(+1)}1 = \frac{1}{n}\sum_{i=1}^{n}\frac{0.5}{1}1 \equiv 0.5,$$

no matter how large n is.

10.3.2 The off-policy policy gradient theorem

With the importance sampling technique, we are ready to present the off-policy policy gradient theorem. Suppose that β is a behavior policy. Our goal is to use the samples generated by β to learn a target policy π that can maximize the following metric:

$$J(\theta) = \sum_{s\in\mathcal{S}} d_\beta(s) v_\pi(s) = \mathbb{E}_{S\sim d_\beta}[v_\pi(S)],$$

where d_β is the stationary distribution under policy β and v_π is the state value under policy π. The gradient of this metric is given in the following theorem.

Theorem 10.1 (Off-policy policy gradient theorem)**.** *In the discounted case where $\gamma \in (0, 1)$, the gradient of $J(\theta)$ is*

$$\nabla_\theta J(\theta) = \mathbb{E}_{S\sim\rho, A\sim\beta}\left[\underbrace{\frac{\pi(A|S,\theta)}{\beta(A|S)}}_{\substack{importance\\ weight}} \nabla_\theta \ln \pi(A|S,\theta) q_\pi(S, A)\right], \tag{10.11}$$

where the state distribution ρ is

$$\rho(s) \doteq \sum_{s'\in\mathcal{S}} d_\beta(s') \mathrm{Pr}_\pi(s|s'), \qquad s \in \mathcal{S},$$

where $\mathrm{Pr}_\pi(s|s') = \sum_{k=0}^{\infty}\gamma^k [P_\pi^k]_{s's} = [(I - \gamma P_\pi)^{-1}]_{s's}$ is the discounted total probability of transitioning from s' to s under policy π.

The gradient in (10.11) is similar to that in the on-policy case in Theorem 9.1, but there are two differences. The first difference is the importance weight. The second difference is that $A \sim \beta$ instead of $A \sim \pi$. Therefore, we can use the action samples

generated by following β to approximate the true gradient. The proof of the theorem is given in Box 10.2.

Box 10.2: Proof of Theorem 10.1

Since d_β is independent of θ, the gradient of $J(\theta)$ satisfies

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_{s\in\mathcal{S}} d_\beta(s) v_\pi(s) = \sum_{s\in\mathcal{S}} d_\beta(s) \nabla_\theta v_\pi(s). \tag{10.12}$$

According to Lemma 9.2, the expression of $\nabla_\theta v_\pi(s)$ is

$$\nabla_\theta v_\pi(s) = \sum_{s'\in\mathcal{S}} \Pr_\pi(s'|s) \sum_{a\in\mathcal{A}} \nabla_\theta \pi(a|s',\theta) q_\pi(s',a), \tag{10.13}$$

where $\Pr_\pi(s'|s) \doteq \sum_{k=0}^{\infty} \gamma^k [P_\pi^k]_{ss'} = [(I_n - \gamma P_\pi)^{-1}]_{ss'}$. Substituting (10.13) into (10.12) yields

$$\begin{aligned}
\nabla_\theta J(\theta) = \sum_{s\in\mathcal{S}} d_\beta(s) \nabla_\theta v_\pi(s) &= \sum_{s\in\mathcal{S}} d_\beta(s) \sum_{s'\in\mathcal{S}} \Pr_\pi(s'|s) \sum_{a\in\mathcal{A}} \nabla_\theta \pi(a|s',\theta) q_\pi(s',a) \\
&= \sum_{s'\in\mathcal{S}} \left(\sum_{s\in\mathcal{S}} d_\beta(s) \Pr_\pi(s'|s) \right) \sum_{a\in\mathcal{A}} \nabla_\theta \pi(a|s',\theta) q_\pi(s',a) \\
&\doteq \sum_{s'\in\mathcal{S}} \rho(s') \sum_{a\in\mathcal{A}} \nabla_\theta \pi(a|s',\theta) q_\pi(s',a) \\
&= \sum_{s\in\mathcal{S}} \rho(s) \sum_{a\in\mathcal{A}} \nabla_\theta \pi(a|s,\theta) q_\pi(s,a) \qquad \text{(change } s' \text{ to } s\text{)} \\
&= \mathbb{E}_{S\sim\rho} \left[\sum_{a\in\mathcal{A}} \nabla_\theta \pi(a|S,\theta) q_\pi(S,a) \right].
\end{aligned}$$

By using the importance sampling technique, the above equation can be further rewritten as

$$\begin{aligned}
\mathbb{E}_{S\sim\rho} \left[\sum_{a\in\mathcal{A}} \nabla_\theta \pi(a|S,\theta) q_\pi(S,a) \right] &= \mathbb{E}_{S\sim\rho} \left[\sum_{a\in\mathcal{A}} \beta(a|S) \frac{\pi(a|S,\theta)}{\beta(a|S)} \frac{\nabla_\theta \pi(a|S,\theta)}{\pi(a|S,\theta)} q_\pi(S,a) \right] \\
&= \mathbb{E}_{S\sim\rho} \left[\sum_{a\in\mathcal{A}} \beta(a|S) \frac{\pi(a|S,\theta)}{\beta(a|S)} \nabla_\theta \ln \pi(a|S,\theta) q_\pi(S,a) \right] \\
&= \mathbb{E}_{S\sim\rho, A\sim\beta} \left[\frac{\pi(A|S,\theta)}{\beta(A|S)} \nabla_\theta \ln \pi(A|S,\theta) q_\pi(S,A) \right].
\end{aligned}$$

The proof is complete. The above proof is similar to that of Theorem 9.1.

10.3.3 Algorithm description

Based on the off-policy policy gradient theorem, we are ready to present the off-policy actor-critic algorithm. Since the off-policy case is very similar to the on-policy case, we merely present some key steps.

First, the off-policy policy gradient is invariant to any additional baseline $b(s)$. In particular, we have

$$\nabla_\theta J(\theta) = \mathbb{E}_{S\sim\rho, A\sim\beta}\left[\frac{\pi(A|S,\theta)}{\beta(A|S)}\nabla_\theta \ln \pi(A|S,\theta)\big(q_\pi(S,A) - b(S)\big)\right],$$

because $\mathbb{E}\left[\frac{\pi(A|S,\theta)}{\beta(A|S)}\nabla_\theta \ln \pi(A|S,\theta)b(S)\right] = 0$. To reduce the estimation variance, we can select the baseline as $b(S) = v_\pi(S)$ and obtain

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\frac{\pi(A|S,\theta)}{\beta(A|S)}\nabla_\theta \ln \pi(A|S,\theta)\big(q_\pi(S,A) - v_\pi(S)\big)\right].$$

The corresponding stochastic gradient-ascent algorithm is

$$\theta_{t+1} = \theta_t + \alpha_\theta \frac{\pi(a_t|s_t,\theta_t)}{\beta(a_t|s_t)}\nabla_\theta \ln \pi(a_t|s_t,\theta_t)\big(q_t(s_t,a_t) - v_t(s_t)\big),$$

where $\alpha_\theta > 0$. Similar to the on-policy case, the advantage function $q_t(s,a) - v_t(s)$ can be replaced by the TD error. That is

$$q_t(s_t,a_t) - v_t(s_t) \approx r_{t+1} + \gamma v_t(s_{t+1}) - v_t(s_t) \doteq \delta_t(s_t,a_t).$$

Then, the algorithm becomes

$$\theta_{t+1} = \theta_t + \alpha_\theta \frac{\pi(a_t|s_t,\theta)}{\beta(a_t|s_t)}\nabla_\theta \ln \pi(a_t|s_t,\theta)\delta_t(s_t,a_t).$$

The implementation of the off-policy actor-critic algorithm is summarized in Algorithm 10.3. As can be seen, the algorithm is the same as the advantage actor-critic algorithm except that an additional importance weight is included in both the critic and the actor. It must be noted that, in addition to the actor, the critic is also converted from on-policy to off-policy by the importance sampling technique. In fact, importance sampling is a general technique that can be applied to both policy-based and value-based algorithms. Finally, Algorithm 10.3 can be extended in various ways to incorporate more techniques such as eligibility traces [73].

Algorithm 10.3: Off-policy actor-critic based on importance sampling

Initialization: A given behavior policy $\beta(a|s)$. A target policy $\pi(a|s,\theta_0)$ where θ_0 is the initial parameter. A value function $v(s,w_0)$ where w_0 is the initial parameter. $\alpha_w, \alpha_\theta > 0$.
Goal: Learn an optimal policy to maximize $J(\theta)$.

At time step t in each episode, do

Generate a_t following $\beta(s_t)$ and then observe r_{t+1}, s_{t+1}.

Advantage (TD error):

$$\delta_t = r_{t+1} + \gamma v(s_{t+1}, w_t) - v(s_t, w_t)$$

Actor (policy update):

$$\theta_{t+1} = \theta_t + \alpha_\theta \frac{\pi(a_t|s_t,\theta_t)}{\beta(a_t|s_t)} \delta_t \nabla_\theta \ln \pi(a_t|s_t,\theta_t)$$

Critic (value update):

$$w_{t+1} = w_t + \alpha_w \frac{\pi(a_t|s_t,\theta_t)}{\beta(a_t|s_t)} \delta_t \nabla_w v(s_t, w_t)$$

10.4 Deterministic actor-critic

Up to now, the policies used in the policy gradient methods are all *stochastic* since it is required that $\pi(a|s,\theta) > 0$ for every (s,a). This section shows that *deterministic* policies can also be used in policy gradient methods. Here, "deterministic" indicates that, for any state, a single action is given a probability of one and all the other actions are given probabilities of zero. It is important to study the deterministic case since it is naturally off-policy and can effectively handle continuous action spaces.

We have been using $\pi(a|s,\theta)$ to denote a general policy, which can be either stochastic or deterministic. In this section, we use

$$a = \mu(s,\theta)$$

to specifically denote a deterministic policy. Different from π which gives the probability of an action, μ directly gives the action since it is a mapping from $\mathcal{S}$ to $\mathcal{A}$. This deterministic policy can be represented by, for example, a neural network with s as its input, a as its output, and θ as its parameter. For the sake of simplicity, we often write $\mu(s,\theta)$ as $\mu(s)$ for short.

10.4.1 The deterministic policy gradient theorem

The policy gradient theorem introduced in the last chapter is only valid for stochastic policies. When we require the policy to be deterministic, a new policy gradient theorem must be derived.

Theorem 10.2 (Deterministic policy gradient theorem)**.** *The gradient of $J(\theta)$ is*

$$\begin{aligned}\nabla_\theta J(\theta) &= \sum_{s\in\mathcal{S}} \eta(s) \nabla_\theta \mu(s) \big(\nabla_a q_\mu(s,a)\big)|_{a=\mu(s)} \\ &= \mathbb{E}_{S\sim\eta}\left[\nabla_\theta \mu(S) \big(\nabla_a q_\mu(S,a)\big)|_{a=\mu(S)}\right], \end{aligned} \tag{10.14}$$

where η is a distribution of the states.

Theorem 10.2 is a summary of the results presented in Theorem 10.3 and Theorem 10.4 since the gradients in the two theorems have similar expressions. The specific expressions of $J(\theta)$ and η can be found in Theorems 10.3 and 10.4.

Unlike the stochastic case, the gradient in the deterministic case shown in (10.14) does not involve the action random variable A. As a result, when we use samples to approximate the true gradient, it is not required to sample actions. Therefore, the deterministic policy gradient method is *off-policy*. In addition, some readers may wonder why $\big(\nabla_a q_\mu(S,a)\big)|_{a=\mu(S)}$ cannot be written as $\nabla_a q_\mu(S,\mu(S))$, which seems more concise. That is simply because, if we do that, it is unclear how $q_\mu(S,\mu(S))$ is a function of a. A concise yet less confusing expression may be $\nabla_a q_\mu(S, a=\mu(S))$.

In the rest of this subsection, we present the derivation details of Theorem 10.2. In particular, we derive the gradients of two common metrics: the first is the average value and the second is the average reward. Since these two metrics have been discussed in detail in Section 9.2, we sometimes use their properties without proof. For most readers, it is sufficient to be familiar with Theorem 10.2 without knowing its derivation details. Interested readers can selectively examine the details in the remainder of this section.

Metric 1: Average value

We first derive the gradient of the average value:

$$J(\theta) = \mathbb{E}[v_\mu(s)] = \sum_{s\in\mathcal{S}} d_0(s) v_\mu(s), \tag{10.15}$$

where d_0 is the probability distribution of the states. Here, d_0 is selected to be *independent* of μ for simplicity. There are two special yet important cases of selecting d_0. The first case is that $d_0(s_0) = 1$ and $d_0(s \neq s_0) = 0$, where s_0 is a specific state of interest. In this case, the policy aims to maximize the discounted return that can be obtained when starting from s_0. The second case is that d_0 is the distribution of a given behavior policy that is different from the target policy.

To calculate the gradient of $J(\theta)$, we need to first calculate the gradient of $v_\mu(s)$ for any $s \in \mathcal{S}$. Consider the discounted case where $\gamma \in (0,1)$.

Lemma 10.1 (Gradient of $v_\mu(s)$)**.** *In the discounted case, it holds for any $s \in \mathcal{S}$ that*

$$\nabla_\theta v_\mu(s) = \sum_{s' \in \mathcal{S}} \Pr_\mu(s'|s) \nabla_\theta \mu(s') \big(\nabla_a q_\mu(s', a)\big)|_{a=\mu(s')}, \tag{10.16}$$

where

$$\Pr_\mu(s'|s) \doteq \sum_{k=0}^{\infty} \gamma^k [P_\mu^k]_{ss'} = \big[(I - \gamma P_\mu)^{-1}\big]_{ss'}$$

is the discounted total probability of transitioning from s to s' under policy μ. Here, $[\cdot]_{ss'}$ denotes the entry in the sth row and s'th column of a matrix.

Box 10.3: Proof of Lemma 10.1

Since the policy is deterministic, we have

$$v_\mu(s) = q_\mu(s, \mu(s)).$$

Since both q_μ and μ are functions of θ, we have

$$\nabla_\theta v_\mu(s) = \nabla_\theta q_\mu(s, \mu(s)) = \big(\nabla_\theta q_\mu(s, a)\big)|_{a=\mu(s)} + \nabla_\theta \mu(s) \big(\nabla_a q_\mu(s, a)\big)|_{a=\mu(s)}. \tag{10.17}$$

By the definition of action values, for any given (s, a), we have

$$q_\mu(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) v_\mu(s'),$$

where $r(s, a) = \sum_r r p(r|s, a)$. Since $r(s, a)$ is independent of μ, we have

$$\nabla_\theta q_\mu(s, a) = 0 + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) \nabla_\theta v_\mu(s').$$

Substituting the above equation into (10.17) yields

$$\nabla_\theta v_\mu(s) = \gamma \sum_{s' \in \mathcal{S}} p(s'|s, \mu(s)) \nabla_\theta v_\mu(s') + \underbrace{\nabla_\theta \mu(s) \big(\nabla_a q_\mu(s, a)\big)|_{a=\mu(s)}}_{u(s)}, \quad s \in \mathcal{S}.$$

Since the above equation is valid for all $s \in \mathcal{S}$, we can combine these equations to obtain a matrix-vector form:

$$\underbrace{\begin{bmatrix} \vdots \\ \nabla_\theta v_\mu(s) \\ \vdots \end{bmatrix}}_{\nabla_\theta v_\mu \in \mathbb{R}^{mn}} = \underbrace{\begin{bmatrix} \vdots \\ u(s) \\ \vdots \end{bmatrix}}_{u \in \mathbb{R}^{mn}} + \gamma(P_\mu \otimes I_m) \underbrace{\begin{bmatrix} \vdots \\ \nabla_\theta v_\mu(s') \\ \vdots \end{bmatrix}}_{\nabla_\theta v_\mu \in \mathbb{R}^{mn}},$$

where $n = |\mathcal{S}|$, m is the dimensionality of θ, P_μ is the state transition matrix with $[P_\mu]_{ss'} = p(s'|s, \mu(s))$, and $\otimes$ is the Kronecker product. The above matrix-vector form can be written concisely as

$$\nabla_\theta v_\mu = u + \gamma(P_\mu \otimes I_m)\nabla_\theta v_\mu,$$

which is a linear equation of $\nabla_\theta v_\mu$. Then, $\nabla_\theta v_\mu$ can be solved as

$$\begin{aligned} \nabla_\theta v_\mu &= (I_{mn} - \gamma P_\mu \otimes I_m)^{-1} u \\ &= (I_n \otimes I_m - \gamma P_\mu \otimes I_m)^{-1} u \\ &= \left[(I_n - \gamma P_\mu)^{-1} \otimes I_m\right] u. \end{aligned} \tag{10.18}$$

The elementwise form of (10.18) is

$$\begin{aligned} \nabla_\theta v_\mu(s) &= \sum_{s' \in \mathcal{S}} \left[(I - \gamma P_\mu)^{-1}\right]_{ss'} u(s') \\ &= \sum_{s' \in \mathcal{S}} \left[(I - \gamma P_\mu)^{-1}\right]_{ss'} \left[\nabla_\theta \mu(s') \big(\nabla_a q_\mu(s', a)\big)|_{a=\mu(s')}\right]. \end{aligned} \tag{10.19}$$

The quantity $[(I - \gamma P_\mu)^{-1}]_{ss'}$ has a clear probabilistic interpretation. Since $(I - \gamma P_\mu)^{-1} = I + \gamma P_\mu + \gamma^2 P_\mu^2 + \cdots$, we have

$$\left[(I - \gamma P_\mu)^{-1}\right]_{ss'} = [I]_{ss'} + \gamma[P_\mu]_{ss'} + \gamma^2[P_\mu^2]_{ss'} + \cdots = \sum_{k=0}^{\infty} \gamma^k [P_\mu^k]_{ss'}.$$

Note that $[P_\mu^k]_{ss'}$ is the probability of transitioning from s to s' using exactly k steps (see Box 8.1 for more information). Therefore, $[(I - \gamma P_\mu)^{-1}]_{ss'}$ is the discounted total probability of transitioning from s to s' using any number of steps. By denoting $[(I - \gamma P_\mu)^{-1}]_{ss'} \doteq \Pr_\mu(s'|s)$, equation (10.19) leads to (10.16).

With the preparation in Lemma 10.1, we are ready to derive the gradient of $J(\theta)$.

Theorem 10.3 (Deterministic policy gradient theorem in the discounted case)**.** *In the*

discounted case where $\gamma \in (0,1)$, the gradient of $J(\theta)$ in (10.15) is

$$\begin{aligned}\nabla_\theta J(\theta) &= \sum_{s\in\mathcal{S}} \rho_\mu(s) \nabla_\theta \mu(s) \big(\nabla_a q_\mu(s,a)\big)|_{a=\mu(s)} \\ &= \mathbb{E}_{S\sim\rho_\mu}\left[\nabla_\theta \mu(S) \big(\nabla_a q_\mu(S,a)\big)|_{a=\mu(S)}\right],\end{aligned}$$

where the state distribution ρ_μ is

$$\rho_\mu(s) = \sum_{s'\in\mathcal{S}} d_0(s') \Pr_\mu(s|s'), \qquad s \in \mathcal{S}.$$

Here, $\Pr_\mu(s|s') = \sum_{k=0}^{\infty} \gamma^k [P_\mu^k]_{s's} = [(I-\gamma P_\mu)^{-1}]_{s's}$ is the discounted total probability of transitioning from s' to s under policy μ.

Box 10.4: Proof of Theorem 10.3

Since d_0 is independent of μ, we have

$$\nabla_\theta J(\theta) = \sum_{s\in\mathcal{S}} d_0(s) \nabla_\theta v_\mu(s).$$

Substituting the expression of $\nabla_\theta v_\mu(s)$ given by Lemma 10.1 into the above equation yields

$$\begin{aligned}\nabla_\theta J(\theta) &= \sum_{s\in\mathcal{S}} d_0(s) \nabla_\theta v_\mu(s) \\ &= \sum_{s\in\mathcal{S}} d_0(s) \sum_{s'\in\mathcal{S}} \Pr_\mu(s'|s) \nabla_\theta \mu(s') \big(\nabla_a q_\mu(s',a)\big)|_{a=\mu(s')} \\ &= \sum_{s'\in\mathcal{S}} \left(\sum_{s\in\mathcal{S}} d_0(s) \Pr_\mu(s'|s) \right) \nabla_\theta \mu(s') \big(\nabla_a q_\mu(s',a)\big)|_{a=\mu(s')} \\ &\doteq \sum_{s'\in\mathcal{S}} \rho_\mu(s') \nabla_\theta \mu(s') \big(\nabla_a q_\mu(s',a)\big)|_{a=\mu(s')} \\ &= \sum_{s\in\mathcal{S}} \rho_\mu(s) \nabla_\theta \mu(s) \big(\nabla_a q_\mu(s,a)\big)|_{a=\mu(s)} \qquad \text{(change } s' \text{ to } s\text{)} \\ &= \mathbb{E}_{S\sim\rho_\mu}\left[\nabla_\theta \mu(S) \big(\nabla_a q_\mu(S,a)\big)|_{a=\mu(S)}\right].\end{aligned}$$

The proof is complete. The above proof is consistent with the proof of Theorem 1 in [74]. Here, we consider the case in which the states and actions are finite. When they are continuous, the proof is similar, but the summations should be replaced by integrals [74].

Metric 2: Average reward

We next derive the gradient of the average reward:

$$\begin{aligned} J(\theta) = \bar{r}_\mu &= \sum_{s \in \mathcal{S}} d_\mu(s) r_\mu(s) \\ &= \mathbb{E}_{S \sim d_\mu}[r_\mu(S)], \end{aligned} \tag{10.20}$$

where

$$r_\mu(s) = \mathbb{E}[R|s, a = \mu(s)] = \sum_r r p(r|s, a = \mu(s))$$

is the expectation of the immediate rewards. More information about this metric can be found in Section 9.2.

The gradient of $J(\theta)$ is given in the following theorem.

Theorem 10.4 (Deterministic policy gradient theorem in the undiscounted case)**.** *In the undiscounted case, the gradient of $J(\theta)$ in* (10.20) *is*

$$\begin{aligned} \nabla_\theta J(\theta) &= \sum_{s \in \mathcal{S}} d_\mu(s) \nabla_\theta \mu(s) \big(\nabla_a q_\mu(s, a)\big)|_{a=\mu(s)} \\ &= \mathbb{E}_{S \sim d_\mu} \left[\nabla_\theta \mu(S) \big(\nabla_a q_\mu(S, a)\big)|_{a=\mu(S)}\right], \end{aligned}$$

where d_μ is the stationary distribution of the states under policy μ.

Box 10.5: Proof of Theorem 10.4

Since the policy is deterministic, we have

$$v_\mu(s) = q_\mu(s, \mu(s)).$$

Since both q_μ and μ are functions of θ, we have

$$\nabla_\theta v_\mu(s) = \nabla_\theta q_\mu(s, \mu(s)) = \big(\nabla_\theta q_\mu(s, a)\big)|_{a=\mu(s)} + \nabla_\theta \mu(s) \big(\nabla_a q_\mu(s, a)\big)|_{a=\mu(s)}. \tag{10.21}$$

In the undiscounted case, it follows from the definition of action value (Section 9.3.2) that

$$\begin{aligned} q_\mu(s, a) &= \mathbb{E}[R_{t+1} - \bar{r}_\mu + v_\mu(S_{t+1})|s, a] \\ &= \sum_r p(r|s, a)(r - \bar{r}_\mu) + \sum_{s'} p(s'|s, a) v_\mu(s') \\ &= r(s, a) - \bar{r}_\mu + \sum_{s'} p(s'|s, a) v_\mu(s'). \end{aligned}$$

Since $r(s,a) = \sum_r rp(r|s,a)$ is independent of θ, we have

$$\nabla_\theta q_\mu(s,a) = 0 - \nabla_\theta \bar{r}_\mu + \sum_{s'} p(s'|s,a)\nabla_\theta v_\mu(s').$$

Substituting the above equation into (10.21) gives

$$\nabla_\theta v_\mu(s) = -\nabla_\theta \bar{r}_\mu + \sum_{s'} p(s'|s,\mu(s))\nabla_\theta v_\mu(s') + \underbrace{\nabla_\theta \mu(s)\big(\nabla_a q_\mu(s,a)\big)|_{a=\mu(s)}}_{u(s)}, \quad s \in \mathcal{S}.$$

While the above equation is valid for all $s \in \mathcal{S}$, we can combine these equations to obtain a matrix-vector form:

$$\underbrace{\begin{bmatrix} \vdots \\ \nabla_\theta v_\mu(s) \\ \vdots \end{bmatrix}}_{\nabla_\theta v_\mu \in \mathbb{R}^{mn}} = -\mathbf{1}_n \otimes \nabla_\theta \bar{r}_\mu + (P_\mu \otimes I_m) \underbrace{\begin{bmatrix} \vdots \\ \nabla_\theta v_\mu(s') \\ \vdots \end{bmatrix}}_{\nabla_\theta v_\mu \in \mathbb{R}^{mn}} + \underbrace{\begin{bmatrix} \vdots \\ u(s) \\ \vdots \end{bmatrix}}_{u \in \mathbb{R}^{mn}},$$

where $n = |\mathcal{S}|$, m is the dimension of θ, P_μ is the state transition matrix with $[P_\mu]_{ss'} = p(s'|s,\mu(s))$, and $\otimes$ is the Kronecker product. The above matrix-vector form can be written concisely as

$$\nabla_\theta v_\mu = u - \mathbf{1}_n \otimes \nabla_\theta \bar{r}_\mu + (P_\mu \otimes I_m)\nabla_\theta v_\mu,$$

and hence

$$\mathbf{1}_n \otimes \nabla_\theta \bar{r}_\mu = u + (P_\mu \otimes I_m)\nabla_\theta v_\mu - \nabla_\theta v_\mu. \tag{10.22}$$

Since d_μ is the stationary distribution, we have $d_\mu^T P_\mu = d_\mu^T$. Multiplying $d_\mu^T \otimes I_m$ on both sides of (10.22) gives

$$\begin{aligned}(d_\mu^T \mathbf{1}_n) \otimes \nabla_\theta \bar{r}_\mu &= d_\mu^T \otimes I_m u + (d_\mu^T P_\mu) \otimes I_m \nabla_\theta v_\mu - d_\mu^T \otimes I_m \nabla_\theta v_\mu \\ &= d_\mu^T \otimes I_m u + d_\mu^T \otimes I_m \nabla_\theta v_\mu - d_\mu^T \otimes I_m \nabla_\theta v_\mu \\ &= d_\mu^T \otimes I_m u.\end{aligned}$$

Since $d_\mu^T \mathbf{1}_n = 1$, the above equations become

$$\begin{aligned}\nabla_\theta \bar{r}_\mu &= d_\mu^T \otimes I_m u \\ &= \sum_{s\in\mathcal{S}} d_\mu(s) u(s) \\ &= \sum_{s\in\mathcal{S}} d_\mu(s) \nabla_\theta \mu(s) \big(\nabla_a q_\mu(s,a)\big)|_{a=\mu(s)} \\ &= \mathbb{E}_{S\sim d_\mu}\left[\nabla_\theta \mu(S)\big(\nabla_a q_\mu(S,a)\big)|_{a=\mu(S)}\right].\end{aligned}$$

The proof is complete.

10.4.2 Algorithm description

Based on the gradient given in Theorem 10.2, we can apply the gradient-ascent algorithm to maximize $J(\theta)$:

$$\theta_{t+1} = \theta_t + \alpha_\theta \mathbb{E}_{S\sim\eta}\left[\nabla_\theta \mu(S)\big(\nabla_a q_\mu(S,a)\big)|_{a=\mu(S)}\right].$$

The corresponding stochastic gradient-ascent algorithm is

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_\theta \mu(s_t)\big(\nabla_a q_\mu(s_t,a)\big)|_{a=\mu(s_t)}.$$

The implementation is summarized in Algorithm 10.4. It should be noted that this algorithm is *off-policy* since the behavior policy β may be different from μ. First, the actor is off-policy. We already explained the reason when presenting Theorem 10.2. Second, the critic is also off-policy. Special attention must be paid to why the critic is off-policy but does not require the importance sampling technique. In particular, the experience sample required by the critic is $(s_t, a_t, r_{t+1}, s_{t+1}, \tilde{a}_{t+1})$, where $\tilde{a}_{t+1} = \mu(s_{t+1})$. The generation of this experience sample involves two policies. The first is the policy for generating a_t at s_t, and the second is the policy for generating $\tilde{a}_{t+1}$ at s_{t+1}. The first policy that generates a_t is the behavior policy since a_t is used to interact with the environment. The second policy must be μ because it is the policy that the critic aims to evaluate. Hence, μ is the target policy. It should be noted that $\tilde{a}_{t+1}$ is not used to interact with the environment in the next time step. Hence, μ is not the behavior policy. Therefore, the critic is off-policy.

How to select the function $q(s,a,w)$? The original research work [74] that proposed the deterministic policy gradient method adopted linear functions: $q(s,a,w) = \phi^T(s,a)w$ where $\phi(s,a)$ is the feature vector. It is currently popular to represent $q(s,a,w)$ using neural networks, as suggested in the deep deterministic policy gradient (DDPG) method [75].

Algorithm 10.4: Deterministic policy gradient or deterministic actor-critic

Initialization: A given behavior policy $\beta(a|s)$. A deterministic target policy $\mu(s, \theta_0)$ where θ_0 is the initial parameter. A value function $q(s, a, w_0)$ where w_0 is the initial parameter. $\alpha_w, \alpha_\theta > 0$.
Goal: Learn an optimal policy to maximize $J(\theta)$.

At time step t in each episode, do

Generate a_t following β and then observe r_{t+1}, s_{t+1}.

TD error:

$$\delta_t = r_{t+1} + \gamma q(s_{t+1}, \mu(s_{t+1}, \theta_t), w_t) - q(s_t, a_t, w_t)$$

Actor (policy update):

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_\theta \mu(s_t, \theta_t) \big(\nabla_a q(s_t, a, w_t)\big)|_{a=\mu(s_t)}$$

Critic (value update):

$$w_{t+1} = w_t + \alpha_w \delta_t \nabla_w q(s_t, a_t, w_t)$$

How to select the behavior policy β? It can be any exploratory policy. It can also be a stochastic policy obtained by adding noise to μ [75]. In this case, μ is also the behavior policy and hence this way is an on-policy implementation.

10.5 Summary

In this chapter, we introduced actor-critic methods. The contents are summarized as follows.

- ◇ Section 10.1 introduced the simplest actor-critic algorithm called QAC. This algorithm is similar to the policy gradient algorithm, REINFORCE, introduced in the last chapter. The only difference is that the q-value estimation in QAC relies on TD learning while REINFORCE relies on Monte Carlo estimation.
- ◇ Section 10.2 extended QAC to advantage actor-critic. It was shown that the policy gradient is invariant to any additional baseline. It was then shown that an optimal baseline could help reduce the estimation variance.
- ◇ Section 10.3 further extended the advantage actor-critic algorithm to the off-policy case. To do that, we introduced an important technique called importance sampling.
- ◇ Finally, while all the previously presented policy gradient algorithms rely on stochastic policies, we showed in Section 10.4 that the policy can be forced to be deterministic. The corresponding gradient was derived, and the deterministic policy gradient algorithm was introduced.

Policy gradient and actor-critic methods are widely used in modern reinforcement learning. There exist a large number of advanced algorithms in the literature such as SAC [76, 77], TRPO [78], PPO [79], and TD3 [80]. In addition, the single-agent case can

also be extended to the case of multi-agent reinforcement learning [81–85]. Experience samples can also be used to fit system models to achieve model-based reinforcement learning [15, 86, 87]. Distributional reinforcement learning provides a fundamentally different perspective from the conventional one [88, 89]. The relationships between reinforcement learning and control theory have been discussed in [90–95]. This book is not able to cover all these topics. Hopefully, the foundations laid by this book can help readers better study them in the future.

10.6 Q&A

◇ Q: What is the relationship between actor-critic and policy gradient methods?

A: Actor-critic methods are actually policy gradient methods. Sometimes, we use them interchangeably. It is required to estimate action values in any policy gradient algorithm. When the action values are estimated using temporal-difference learning with value function approximation, such a policy gradient algorithm is called actor-critic. The name "actor-critic" highlights its algorithmic structure that combines the components of policy update and value update. This structure is also the fundamental structure used in all reinforcement learning algorithms.

◇ Q: Why is it important to introduce additional baselines to actor-critic methods?

A: Since the policy gradient is invariant to any additional baseline, we can utilize the baseline to reduce estimation variance. The resulting algorithm is called advantage actor-critic.

◇ Q: Can importance sampling be used in value-based algorithms other than policy-based ones?

A: The answer is yes. That is because importance sampling is a general technique for estimating the expectation of a random variable over *one distribution* using some samples drawn from *another distribution*. The reason why this technique is useful in reinforcement learning is that the many problems in reinforcement learning are to estimate expectations. For example, in value-based methods, the action or state values are defined as expectations. In the policy gradient method, the true gradient is also an expectation. As a result, importance sampling can be applied in both value-based and policy-based algorithms. In fact, it has been applied in the value-based component of Algorithm 10.3.

◇ Q: Why is the deterministic policy gradient method off-policy?

A: The true gradient in the deterministic case does not involve the action random variable. As a result, when we use samples to approximate the true gradient, it is not required to sample actions and hence any policy can be used. Therefore, the deterministic policy gradient method is *off-policy*.