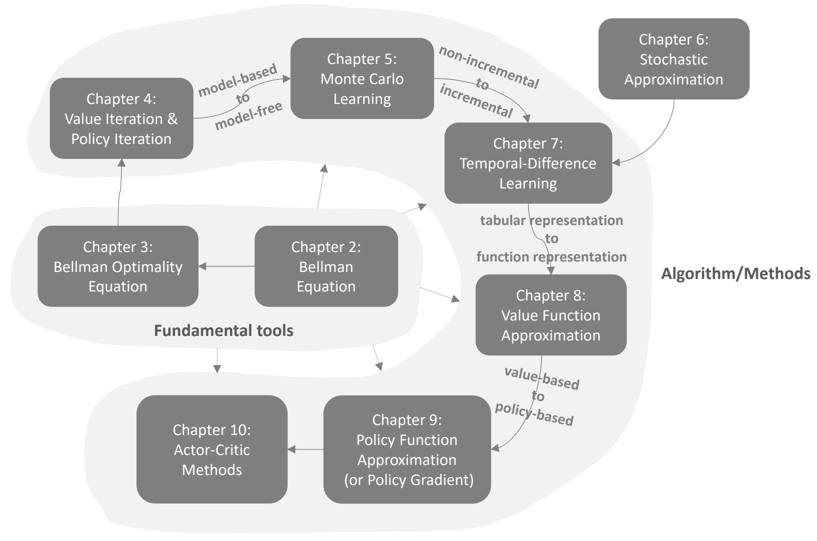
Lecture 10: Actor-Critic Methods

Shiyu Zhao

Introduction



Introduction

Actor-critic methods are still policy gradient methods.

- They emphasize the structure that incorporates the policy gradient and value-based methods.

What are "actor" and "critic"?

- Here, "actor" refers to policy update. It is called *actor* is because the policies will be applied to take actions.
- Here, "critic" refers to policy evaluation or value estimation. It is called *critic* because it criticizes the policy by evaluating it.

Outline

Outline

The simplest actor-critic

Revisit the idea of policy gradient introduced in the last lecture.

1) A scalar metric $J(\theta)$, which can be $\bar{v}_\pi$ or $\bar{r}_\pi$.

2) The gradient-ascent algorithm maximizing $J(\theta)$ is

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

3) The stochastic gradient-ascent algorithm is

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

We can see "actor" and "critic" from this algorithm:

- This algorithm corresponds to actor!
- The algorithm estimating $q_t(s,a)$ corresponds to critic!

The simplest actor-critic

How to get $q_t(s_t, a_t)$?

So far, we have studied two ways to estimate action values:

- **Monte Carlo learning:** If MC is used, the corresponding algorithm is called REINFORCE or Monte Carlo policy gradient.
  - We introduced in the last lecture.
- **Temporal-difference learning:** If TD is used, such kind of algorithms are usually called actor-critic.
  - We will introduce in this lecture.

The simplest actor-critic

The simplest actor-critic algorithm (QAC)

Aim: Search for an optimal policy by maximizing $J(\theta)$.

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 $\pi(a|s_{t+1}, \theta_t)$.

Critic (value update):

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

Actor (policy update):

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

The simplest actor-critic

Remarks:

- The critic corresponds to "SARSA+value function approximation".
- The actor corresponds to the policy update algorithm.
- The algorithm is on-policy (why is PG on-policy?).
  - Since the policy is stochastic, no need to use techniques like ε-greedy.
- This particular actor-citric algorithm is sometimes referred to as Q Actor-Critic (QAC).
- Though simple, this algorithm reveals the core idea of actor-critic methods. It can be extended to generate many other algorithms as shown later.

Outline

Introduction

Next, we extend QAC to advantage actor-critic (A2C)

- The core idea is to introduce a baseline to reduce variance.

Outline

Baseline invariance

Property: the policy gradient is invariant to an additional baseline:

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{S\sim\eta,A\sim\pi}\Big[\nabla_\theta \ln \pi(A|S,\theta_t) q_\pi(S,A)\Big] \\ &= \mathbb{E}_{S\sim\eta,A\sim\pi}\Big[\nabla_\theta \ln \pi(A|S,\theta_t)(q_\pi(S,A) - b(S))\Big]\end{aligned}$$

Here, the additional baseline $b(S)$ is a scalar function of S.

Next, we answer two questions:

- Why is it valid?
- Why is it useful?

Baseline invariance

First, why is it valid?

That is because

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

The details:

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

Baseline invariance

Second, why is the baseline useful?

The gradient is $\nabla_\theta J(\theta) = \mathbb{E}[X]$ where

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

We have

- $\mathbb{E}[X]$ is invariant to $b(S)$.
- $\mathrm{var}(X)$ is NOT invariant to $b(S)$.
  - Why? Because $\mathrm{tr}[\mathrm{var}(X)] = \mathbb{E}[X^T X] - \bar{x}^T \bar{x}$ and

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

Imagine b is huge (e.g., 1 millon)

Our goal: Select an optimal baseline b to minimize $\mathrm{var}(X)$

- Benefit: when we use a random sample to approximate $\mathbb{E}[X]$, the estimation variance would also be small.

In the algorithms of REINFORCE and QAC,

- There is no baseline.
- Or, we can say $b = 0$, which is not guaranteed to be a good baseline.

Baseline invariance

- The optimal baseline that can minimize $\mathrm{var}(X)$ is, for any $s \in \mathcal{S}$,

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

  See the proof in my book.

- Although this baseline is optimal, it is complex.

- We can remove the weight $\|\nabla_\theta \ln \pi(A|s,\theta_t)\|^2$ and select the suboptimal baseline:

$$b(s) = \mathbb{E}_{A\sim\pi}[q_\pi(s,A)] = v_\pi(s)$$

  which is the state value of s!

Outline

The algorithm of advantage actor-critic

When $b(s) = v_\pi(s)$,

- the gradient-ascent algorithm is

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

  where

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

  is called the advantage function (why called advantage?).

- the stochastic version of this algorithm is

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

The algorithm of advantage actor-critic

Moreover, the algorithm can be reexpressed as

$$
\begin{aligned}
\theta_{t+1} &= \theta_t + \alpha \nabla_\theta \ln \pi(a_t|s_t,\theta_t)\delta_t(s_t,a_t) \\
&= \theta_t + \alpha \frac{\nabla_\theta \pi(a_t|s_t,\theta_t)}{\pi(a_t|s_t,\theta_t)} \delta_t(s_t,a_t) \\
&= \theta_t + \alpha \underbrace{\left(\frac{\delta_t(s_t,a_t)}{\pi(a_t|s_t,\theta_t)} \right)}_{\text{step size}} \nabla_\theta \pi(a_t|s_t,\theta_t)
\end{aligned}
$$

- The step size is proportional to the relative value δ_t rather than the absolute value q_t, which is more reasonable.
- It can still well balance exploration and exploitation.

The algorithm of advantage actor-critic

Furthermore, the advantage function is approximated by the TD error:

$$\delta_t = q_t(s_t, a_t) - v_t(s_t) \to r_{t+1} + \gamma v_t(s_{t+1}) - v_t(s_t)$$

- This approximation is reasonable because

$$\mathbb{E}[q_\pi(S, A) - v_\pi(S)|S = s_t, A = a_t] = \mathbb{E}\Big[R + \gamma v_\pi(S') - v_\pi(S)|S = s_t, A = a_t\Big]$$

- Benefit: only need one network to approximate $v_\pi(s)$ rather than two networks for $q_\pi(s, a)$ and $v_\pi(s)$.

The algorithm of advantage actor-critic

Advantage actor-critic (A2C) or TD actor-critic

Aim: Search for an optimal policy by maximizing $J(\theta)$.

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}.

TD error (advantage function):

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

Critic (value update):

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

Actor (policy update):

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

It is on-policy. Since the policy $\pi(\theta_t)$ is stochastic, no need to use techniques like ε-greedy.

Outline

- Policy gradient is on-policy.
  - Why? because the gradient is $\nabla_\theta J(\theta) = \mathbb{E}_{S\sim\eta, A\sim\pi}[*]$
- Can we convert it to off-policy?
  - Yes, by importance sampling
  - The importance sampling technique is not limited to AC, but also to any algorithm that aims to estimate an expectation.

Outline

Illustrative examples

Consider a random variable $X \in \mathcal{X} = \{+1, -1\}$.

If the probability distribution of X is p_0:

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

then the expectation of X is

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

Question: how to estimate $\mathbb{E}[X]$ by using some samples $\{x_i\}$?

Illustrative examples

Case 1 (we are already familiar):

- The samples $\{x_i\}$ are generated according to p_0:

$$\mathbb{E}[x_i] = \mathbb{E}[X], \quad \mathrm{var}[x_i] = \mathrm{var}[X]$$

Then, the average value can converge to the expectation:

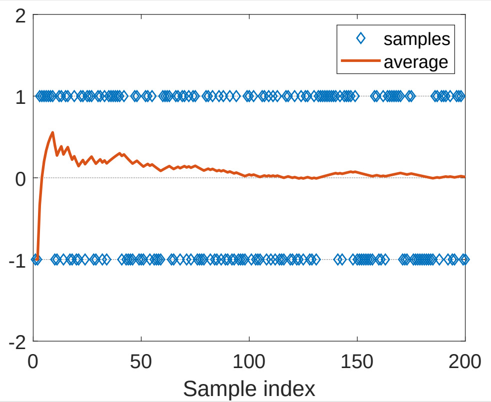
$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \to \mathbb{E}[X], \quad \text{as } n \to \infty$$

because

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

See my book for details (Law of large numbers).

Figure: Samples and $\bar{x} \to \mathbb{E}[X]$



Illustrative examples

Case 2 (a new case that we want to study):

- The samples $\{x_i\}$ are generated according to another distribution p_1:

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

The expectation is

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

If we use the average of the samples, then without suprising

$$\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_i \to \mathbb{E}_{X\sim p_1}[X]=0.6\neq \mathbb{E}_{X\sim p_0}[X]$$

Illustrative examples

Question: Can we use $\{x_i\} \sim p_1$ to estimate $\mathbb{E}_{X\sim p_0}[X]$?

- **Why to do that?**

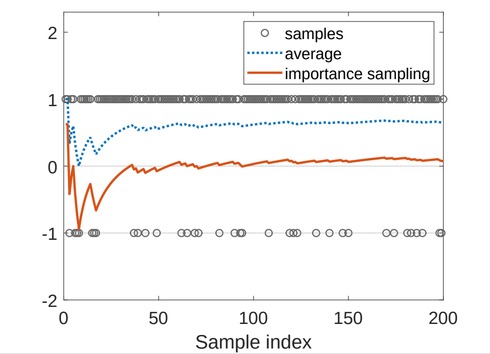
  We may want to estimate $\mathbb{E}_{A\sim\pi}[*]$ where π is the *target policy* based on the samples of a *behavior policy* β.

- **How to do that?**
  - We can't achieve that if directly using $\bar{x}$:

    $$\bar{x} \to \mathbb{E}_{X\sim p_1}[X] = 0.6 \neq \mathbb{E}_{X\sim p_0}[X]$$

  - We can achieve that by using the importance sampling technique.

Figure: Samples and $\bar{x} \to \mathbb{E}_{X\sim p1}[X]$ (the dotted line)



Outline

Importance sampling

Note that

$$\mathbb{E}_{X\sim p_0}[X]=\sum_x p_0(x)x=\sum_x p_1(x)\underbrace{\frac{p_0(x)}{p_1(x)}x}_{f(x)}=\mathbb{E}_{X\sim p_1}[f(X)]$$

- Thus, we can estimate $\mathbb{E}_{X\sim p_1}[f(X)]$ in order to estimate $\mathbb{E}_{X\sim p_0}[X]$.
- How to estimate $\mathbb{E}_{X\sim p_1}[f(X)]$? Easy. Let

$$\bar{f}\doteq\frac{1}{n}\sum_{i=1}^{n}f(x_i),\qquad \text{where } x_i\sim p_1$$

Then,

$$\mathbb{E}_{X\sim p_1}[\bar{f}]=\mathbb{E}_{X\sim p_1}[f(X)]$$
$$\mathrm{var}_{X\sim p_1}[\bar{f}]=\frac{1}{n}\mathrm{var}_{X\sim p_1}[f(X)]$$

Importance sampling

Therefore, $\bar{f}$ is a good approximation for $\mathbb{E}_{X\sim p_1}[f(X)] = \mathbb{E}_{X\sim p_0}[X]$

$$\mathbb{E}_{X\sim p_0}[X] \approx \bar{f} = \frac{1}{n}\sum_{i=1}^{n} f(x_i) = \frac{1}{n}\sum_{i=1}^{n} \frac{p_0(x_i)}{p_1(x_i)} x_i$$

- $\frac{p_0(x_i)}{p_1(x_i)}$ is called the *importance weight*.
  - If $p_1(x_i) = p_0(x_i)$, the importance weight is one and $\bar{f}$ becomes $\bar{x}$.
  - If $p_0(x_i) \geq p_1(x_i)$, x_i can be more often sampled by p_0 than p_1. The importance weight (> 1) can emphasize the importance of this sample.

Importance sampling

You may ask: While $\bar{f} = \frac{1}{n}\sum_{i=1}^{n} \frac{p_0(x_i)}{p_1(x_i)} x_i$ requires $p_0(x)$, if I know $p_0(x)$, why not directly calculate the expectation?

Answer: It is applicable to the case where it is easy to calculate $p_0(x)$ given an x, but difficult to calculate the expectation.

- For example, continuous case, complex expression of p_0, or no expression of p_0 (e.g., p_0 represented by a neural network).

Importance sampling

Summary: if $\{x_i\} \sim p_1$,

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \rightarrow \mathbb{E}_{X\sim p_1}[X]$$

$$\bar{f} = \frac{1}{n}\sum_{i=1}^{n} \frac{p_0(x_i)}{p_1(x_i)} x_i \rightarrow \mathbb{E}_{X\sim p_0}[X]$$

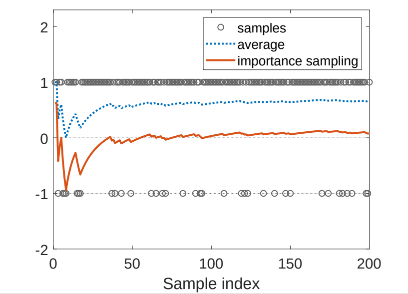


Figure: Blue dotted line: $\bar{x}$; Red solid line: $\bar{f}$

Outline

The theorem of off-policy policy gradient

Like the previous on-policy case, we need to derive the policy gradient in the off-policy case.

- Suppose β is the behavior policy that generates experience samples.
- Our aim is to use these samples to update a target policy π that can minimize the 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.

The theorem of off-policy policy gradient

Theorem (Off-policy policy gradient theorem)

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

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

where β is the behavior policy and ρ is a state distribution.

See the details and the proof in my book.

Outline

The algorithm of off-policy actor-critic

The off-policy policy gradient is also invariant to a baseline $b(s)$.

- In particular, we have

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

- 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 algorithm of off-policy actor-critic

The corresponding stochastic gradient-ascent algorithm is

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

Similar to the on-policy case,

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

Then, the algorithm becomes

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

and hence

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

The algorithm of off-policy actor-critic

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 vector. A value function $v(s,w_0)$ where w_0 is the initial parameter vector.

Aim: Search for an optimal policy by maximizing $J(\theta)$.

At time step t in each episode, do

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

TD error (advantage function):

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

Critic (value update):

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

Actor (policy update):

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

Outline

Introduction

Up to now, the policies used in the policy gradient methods are all stochastic since $\pi(a|s,\theta) > 0$ for every (s,a).

Can we use deterministic policies in the policy gradient methods?

- Benefit: it can handle continuous action.

Introduction

The ways to represent a policy:

- Up to now, a general policy is denoted as $\pi(a|s,\theta) \in [0,1]$, which can be either stochastic or deterministic.
- Now, the deterministic policy is specifically denoted as

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

  - μ is a mapping from $\mathcal{S}$ to $\mathcal{A}$.
  - μ can be represented by, for example, a neural network with the input as s, the output as a, and the parameter as θ.
  - We may write $\mu(s,\theta)$ in short as $\mu(s)$.

Outline

The theorem of deterministic policy gradient

- The policy gradient theorems introduced before are merely valid for stochastic policies.
- If the policy must be deterministic, we must derive a new policy gradient theorem.
- The ideas and procedures are similar.

The theorem of deterministic policy gradient

Consider the metric of average state value in the discounted case:

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

where $d_0(s)$ is a probability distribution satisfying $\sum_{s \in \mathcal{S}} d_0(s) = 1$.

- d_0 is selected to be independent of μ. The gradient in this case is easier to calculate.
- There are two special yet important cases of selecting d_0.
  - The first special case is that $d_0(s_0) = 1$ and $d_0(s \neq s_0) = 0$, where s_0 is a specific starting state of interest.
  - The second special case is that d_0 is the stationary distribution of a behavior policy that is different from the μ.

The theorem of deterministic policy gradient

Theorem (Deterministic policy gradient theorem in the discounted case)

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

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

Here, ρ_μ is a state distribution.

See more details and the proof in my book.

One important difference from the stochastic case:

- The gradient does not involve the distribution of the action A (why?).
- As a result, the deterministic policy gradient method is off-policy.

Outline

The algorithm of deterministic actor-critic

Based on the policy gradient, the gradient-ascent algorithm for maximizing $J(\theta)$ is:

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

The corresponding stochastic gradient-ascent algorithm is

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

Deterministic actor-critic algorithm

Initialization: A given behavior policy $\beta(a|s)$. A deterministic target policy $\mu(s,\theta_0)$ where θ_0 is the initial parameter vector. A value function $v(s,w_0)$ where w_0 is the initial parameter vector.

Aim: Search for an optimal policy by maximizing $J(\theta)$.

At time step t in each episode, do

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

TD error:

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

Critic (value update):

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

Actor (policy update):

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

The algorithm of deterministic actor-critic

Remarks:

- This is an off-policy implementation where the behavior policy β may be different from μ.
- β can also be replaced by μ+noise.
- How to select the function to represent $q(s, a, w)$?
  - Linear function: $q(s, a, w) = \phi^T(s, a)w$ where $\phi(s, a)$ is the feature vector. Details can be found in the DPG paper.
  - Neural networks: deep deterministic policy gradient (DDPG) method.

Summary

- The simplest actor-critic
- Advantage actor-critic
- Off-policy actor-critic
- Deterministic actor-critic

The end

This is the end of the course, but a start for your journey in the field of reinforcement learning!