L4: Policy Gradient

Deep Reinforcement Learning
Tsinghua University, 2025 Spring

Contents

1 Math Prerequisites for L4

Gradient operation:

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix} \tag{1.1}$$

Calculation trick:

$$\nabla_\theta \log z = \frac{1}{z} \nabla_\theta z \tag{1.2}$$

Expectation:

$$\mathbb{E}_{x \sim p(x)}[f(x)] = \int_x p(x) f(x) \mathrm{d}x \tag{1.3}$$

2 Policy Gradient

Policy can be directly parameterized as:

$$\pi_\theta(s, a) = \mathbb{P}[a | s, \theta] \tag{2.1}$$

Here is a comparison among value-based, policy-based and actor-critic.

Value-based	**Policy-based**	**Actor-critic**
Learn Value Function Implicit policy (using Q and ϵ-greedy)	No Value Function Learn Policy	Learn Value Function Learn Policy

Table 1: Comparison of different RL approaches

2.1 Policy Objective Functions

Goal: given policy $\pi_\theta(s, a)$ with parameters, find best θ.

Measurement: the quality of a policy π_θ. We can use average reward per time-step.

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) R_s^a \tag{2.2}$$

Where $d^{\pi_\theta}(s)$ is stationary distribution of Markov chain for π_θ. (This distribution may not be quite important.) R_s^a is the immediate reward at state s when choosing action a.

2.2 Policy Ascending

Update θ:

$$\Delta\theta = \alpha \nabla_\theta J(\theta) \tag{2.3}$$

Where α is learning rate (stepsize), $\nabla_\theta J(\theta)$ is the policy gradient.

NOTE

Two ideas to do policy gradient ascent.

1. Make the world differentiable. (possible)
2. Deal with the scalar reward and find another way out to calculate the gradient.

At each step, we can do gradient ascent to maximize expected reward.
Single action:

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{a\sim p(a|s;\theta)}[r(a)] &= \nabla_\theta \sum_a p(a|s;\theta) r(a) \\
&= \sum_a r(a) \nabla_\theta p(a|s;\theta) \\
&= \sum_a r(a) p(a|s;\theta) \frac{\nabla_\theta p(a|s;\theta)}{p(a|s;\theta)} \\
&= \sum_a r(a) p(a|s;\theta) \nabla_\theta \log p(a|s;\theta) \\
&= \mathbb{E}_{a\sim p(a|s;\theta)}[r(a) \nabla_\theta \log p(a|s;\theta)]
\end{aligned} \tag{2.4}$$

If we want to maximize the reward of **a trajectory**, we can modify it.

Firstly, we calculate the probability of a particular trajectory:

$$p(\tau|\theta) = \mu(s_0) \cdot \prod_{t=0}^{T-1} [\pi(a_t|s_t;\theta) \cdot p(s_{t+1}, r_t|s_t, a_t)] \tag{2.5}$$

Where $\mu(s_0)$ is the initial state distribution.

$$\nabla_\theta \mathbb{E}_\tau[R(\tau)] = \mathbb{E}_\tau[R(\tau) \cdot \nabla_\theta \log p(\tau|\theta)] \tag{2.6}$$

Where $R(\tau)$ is the reward of a trajectory. And we can calculate $\log p(\tau|\theta)$ from (2.4):

$$\begin{aligned}
\log p(\tau|\theta) &= \log \mu(s_0) + \log \prod_{t=0}^{T-1} [\pi(a_t|s_t;\theta) \cdot p(s_{t+1}, r_t|s_t, a_t)] \\
&= \log \mu(s_0) + \sum_{t=0}^{T-1} \log[\pi(a_t|s_t;\theta) \cdot p(s_{t+1}, r_t|s_t, a_t)] \\
&= \log \mu(s_0) + \sum_{t=0}^{T-1} [\log \pi(a_t|s_t;\theta) + \log p(s_{t+1}, r_t|s_t, a_t)]
\end{aligned} \tag{2.7}$$

Substitute into (2.5) and take derive of a trajectory return:

$$\begin{aligned}
\nabla_\theta \mathbb{E}_\tau[R(\tau)] &= \mathbb{E}_\tau[R(\tau) \cdot \nabla_\theta \log p(\tau|\theta)] \\
&= \mathbb{E}_\tau[R(\tau) \cdot \nabla_\theta [\log \mu(s_0) + \sum_{t=0}^{T-1} [\log \pi(a_t|s_t;\theta) + \log p(s_{t+1}, r_t|s_t, a_t)]]] \\
&= \mathbb{E}_\tau[R(\tau) \cdot \nabla_\theta \sum_{t=0}^{T-1} \log \pi(a_t|s_t;\theta)]
\end{aligned} \tag{2.8}$$

NOTE

Here we calculate the expected return of **complete trajectories**. (from $t = 0$ to $T - 1$)

2.3 REINFORCE

The **REINFORCE algorithm** was the first policy gradient method. It is based on Monte-Carlo method (unbiased estimation). The algorithm works as below.

1. Sample $\{\tau^i\}$ from $\pi_\theta(a_t|s_t)$. (run the policy)
2. $\nabla_\theta J(\theta) = \sum_i (\sum_t \nabla_\theta \log \pi_\theta(a_t^i|s_t^i))(\sum_t r(s_t^i, a_t^i))$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

However, MC method causes high variance, so there is an improved version of REINFORCE.

2.3.1 REINFORCE with Baseline

$$\nabla_\theta \mathbb{E}_\tau[R(\tau)] = \mathbb{E}_\tau[(R(\tau) - B(s)) \cdot \nabla_\theta \sum_{t=0}^{T-1} \log \pi(a_t|s_t;\theta)] \tag{2.9}$$

NOTE

Why can we introduce baseline $B(s)$?

1. Baseline will not change the expectation of the gradient (unbiased). $B(s)$ is irrelevant to action choice, thus

$$\begin{aligned}\mathbb{E}[\nabla_\theta \log \pi_\theta(a_t \mid s_t) b(s_t)] &= b(s_t)\mathbb{E}[\nabla_\theta \log \pi_\theta(a_t|s_t)] \\ &= b(s_t) \sum_a \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \\ &= b(s_t) \sum_a \nabla_\theta \pi_\theta(a_t|s_t) \\ &= b(s_t) \nabla_\theta \sum_a \pi_\theta(a_t|s_t) \text{ (linear property)} \\ &= 0\end{aligned} \tag{2.10}$$

2. Baseline will reduce variance. $R(\tau)$ introduces high variance, and proper baseline will reduce the noise in gradient estimation process. (Ideal baseline is $\frac{\mathbb{E}[g(\tau)^2 r(\tau)]}{\mathbb{E}[g(\tau)^2]}$)

reference: The Role of Baselines in Policy Gradient Optimization, instead, the analysis reveals that the primary effect of the value baseline is to reduce the aggressiveness of the updates rather than variance.

2.3.2 Causality

Here is another way to reduce variance. Causality tells us that policy at time t_0 should not affect the reward at time t when $t \leq t_0$. The modified gradient function is:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} (\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i))(\sum_{t'=t}^{T} r(s_{t'}^i, a_{t'}^i)) \tag{2.11}$$

Sometimes we also write it as,

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i))\hat{Q}_{i,t}^{\pi} \tag{2.12}$$

where $\hat{Q}_{i,t}^{\pi} = \sum_{t'=t}^{T} r(s_{t'}^i, a_{t'}^i)$ means "reward-to-go".

2.4 Parameterize Policy

We can use neural networks to parameterize our policy.

2.4.1 Softmax Policy

Probability of action is proportional to exponentiated weight,

$$\pi_\theta(s,a) \propto e^{\phi(s,a)^\top\theta} \tag{2.13}$$

where $\phi(s,a)^\top$ is the feature, and θ is the weight vector. The actual policy can be calculated as:

$$\pi_\theta(s,a) = \frac{e^{\phi(s,a)^\top\theta}}{\sum_{a'} e^{\phi(s,a')^\top\theta}} \tag{2.14}$$

And the score function is:

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(s,a) &= \nabla_\theta[\phi(s,a)^\top\theta - \log\sum_{a'} e^{\phi(s,a')^\top\theta}] \\
&= \phi(s,a)^\top\theta - \frac{\sum_{a'} e^{\phi(s,a')^\top\theta}\phi(s,a')}{\sum_{a'} e^{\phi(s,a')^\top\theta}} \\
&= \phi(s,a)^\top\theta - \sum_{a'}\pi_\theta(s,a)\phi(s,a') \\
&= \phi(s,a)^\top\theta - \mathbb{E}_{a\sim\pi_\theta}\phi(s,a)
\end{aligned} \tag{2.15}$$

2.4.2 Gaussian Policy

We can also use Gaussian policy, where mean is the linear combination of state features $\mu(s) = \phi(s)^\top\theta$, and variance may be fixed σ^2 or can also be

parameterized, $a \sim \mathcal{N}(\mu(s), \sigma^2)$. This method is proper for continuous action space.

$$\pi_\theta(s, a) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(a - \mu(s))^2}{2\sigma^2}\right) \tag{2.16}$$

And the score function is:

$$\nabla_\theta \log \pi_\theta(s, a) = \frac{(a - \mu(s))\phi(s)}{\sigma^2} \tag{2.17}$$

2.5 On-policy Troubles

1. We cannot use other people's experience with on-policy PG.
2. The trajectory used in gradient calculation must be sampled from current policy rather than old or other policies.

2.6 Importance Sampling & Offline PG

A Monte-Carlo method for evaluating properties of a particular distribution, while only having samples generated from a different distribution than the distribution of interest. (wikipedia)

If we want to estimate the expectation of $f(x)$, where $x \sim p(x)$, we could use sampling method:

$$\begin{aligned} \mathbb{E}[f(x)] &= \int f(x)p(x)dx \approx \frac{1}{n}\sum_i f(x_i) \\ \mathbb{E}[f(x)] &= \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx \approx \frac{1}{n}\sum_i f(x_i)\frac{p(x_i)}{q(x_i)} \end{aligned} \tag{2.18}$$

Therefore we can use samples from another policy $\bar{p}(\tau)$:

$$J(\theta) = \mathbb{E}_{\tau\sim\bar{p}(\tau)}[\frac{p_\theta(\tau)}{\bar{p}(\tau)}r(\tau)] \tag{2.19}$$

This is the off-policy PG object. And we can simplify the formula:

$$\frac{p_\theta(\tau)}{\bar{p}(\tau)} = \frac{p(s_1)\prod_{t=1}^{T}\pi(a_t|s_t)p(s_{t+1}|s_t;a_t)}{p(s_1)\prod_{t=1}^{T}\bar{\pi}(a_t|s_t)p(s_{t+1}|s_t;a_t)} = \frac{\prod_{t=1}^{T}\pi(a_t|s_t)}{\prod_{t=1}^{T}\bar{\pi}(a_t|s_t)} \tag{2.20}$$

And we can calculate the gradient:

$$\begin{aligned}
\nabla_{\theta'}J(\theta') &= \mathbb{E}_{\tau\sim p_\theta(\tau)}[\frac{p_{\theta'}(\tau)}{p_\theta(\tau)}\nabla_{\theta'}\log\pi_{\theta'}(\tau)r(\tau)] \\
&= \mathbb{E}_{\tau\sim p_\theta(\tau)}[(\prod_{t=1}^{T}\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)})(\sum_{t=1}^{T}\nabla_{\theta'}\log\pi_{\theta'}(a_t|s_t))(\sum_{t=1}^{T}r(s_t,a_t))] \\
&= \mathbb{E}_{\tau\sim p_\theta(\tau)}[(\sum_{t=1}^{T}\nabla_{\theta'}\log\pi_{\theta'}(a_t|s_t))(\prod_{t'=1}^{t}\frac{\pi_{\theta'}(a_{t'}|s_{t'})}{\pi_\theta(a_{t'}|s_{t'})})(\sum_{t'=t}^{T}r(s_{t'},a_{t'}))]
\end{aligned} \tag{2.21}$$

Future action does not affect current weight. And only "reward to go" could be affected by current policy.