策略梯度：强化学习(PPO/GRPO 等)之根基

——本文最新版、更多原创见我 Github (1.7K Star)：LLM-RL-Visualized

强化学习（RL）是当今大模型、具身智能等领域的热门技术方向。

策略梯度，正是众多主流 RL 算法的**理论根基**（例如，被 ChatGPT、DeepSeek 等广泛使用的 PPO 和 GRPO 算法都是基于策略梯度）。然而，策略梯度也是 RL 算法体系里较为抽象的一部分内容，在各种博客和书籍中存在不少表述冲突和混淆误解。

我结合 Sutton 的书、UC Berkeley 的 CS285 等内容对相关公式**重新推导，并绘制了原理图**，使得推导过程、算法原理更为清晰。多数内容也摘自我的书《大模型算法：强化学习、微调与对齐》，相关概念可进一步查看我的 Github 开源内容，或相关书籍资料（地址见页末）。

P.S. 为适配知乎、公众号等平台的阅读体验，本文采用窄版排版。

1.1 策略梯度的发展、演化

策略梯度（Policy Gradient）算法的思想最早可追溯至 Ronald J. Williams 于 1992 年发表的 REINFORCE 算法[5]，之后，RL 之父 Sutton 等人则在此基础上进一步提出了更为严谨且系统化的策略梯度定理（Policy Gradient Theorem）[4]，为策略梯度算法建立了理论基础并提供了清晰的数学推导。

之后，该领域得到长足的发展，衍生出众多基于策略梯度的算法。OpenAI 先后提出 TRPO、PPO 等算法被广泛应用，DeepSeek 提出更为易用的 GRPO。在其他领域，也诞生了 DPG、DDPG、TD3、SAC 等策略梯度算法。

1.2 策略梯度的简介

策略（Policy，π）：决定智能体在给定状态下如何执行动作的模型（例如，ChatGPT 模型、自动驾驶汽车中运行的智驾模型），是智能体

行为的核心。由于字母 π 的发音与“Policy”一词的前缀相近，因此常用 π 来表示策略。现如今，通常基于深度神经网络来实现策略，表示为 $\pi_\theta(a|s)$，其中 θ 表示策略模型的所有参数，策略模型 $\pi_\theta(a|s)$ 本质上可以看作一个**概率质量函数**，体现了智能体在状态 s 下执行动作 a 的概率。基于这些动作概率进行采样，即可选取要执行的动作 a。在训练过程中，利用梯度下降方法调整策略模型的参数 θ，使得策略 $\pi_\theta(a|s)$ 的动作概率分布逐步趋于最优。

策略梯度（Policy Gradient）方法通过直接优化策略以最大化回报（累积奖励），衍生出多种强化学习算法，可统称为**基于策略**（Policy-Based）的方法。与基于价值（Value-Based）的方法（例如 DQN、Q-learning 等）不同，策略梯度算法直接对策略函数进行参数化，并通过梯度下降的方式优化策略参数。

图 0.1 对比了基于价值的算法与基于策略的算法：基于价值的模型输出每个动作的价值，因此需要通过额外的动作选择策略（例如 softmax、Top-K、Ɛ-贪婪策略等）来决定具体执行哪个动作；而基于策略的模型则直接输出所有动作的概率分布，执行动作时只需根据这些概率进行采样即可。

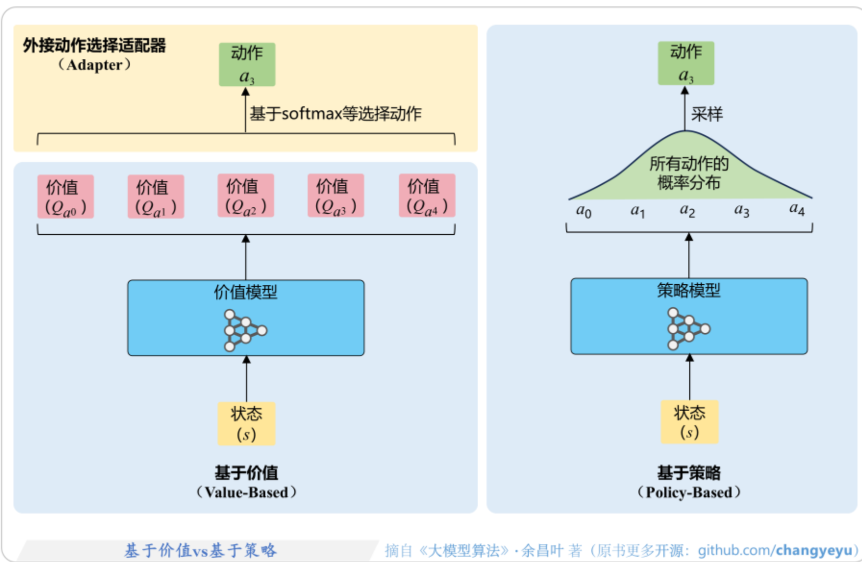

图 0.1 基于价值 vs 基于策略

1.3 策略梯度定理

策略梯度定理（Policy Gradient Theorem）提供了计算策略参数梯度的公式，这使得可以通过梯度下降的方法来优化策略[3]。使用字母 $\boldsymbol{J}$ 表示优化目标（objective），定义预期回报为**优化目标**：

$$J(\theta)=\sum_{\tau}\underbrace{p(\tau;\theta)}_{\text{轨迹概率}}\cdot\underbrace{R(\tau)}_{\text{轨迹回报}}=\underbrace{E_{\tau\sim p(\tau;\theta)}[R(\tau)]}_{\text{所有轨迹回报的均值}} \tag{0.1}$$

其中：

（1）τ 是一条轨迹，如 $\{s_0,a_0,r_1,s_1,a_1,\cdots,r_{T-1},s_{T-1},a_{T-1},r_T,s_T\}$ 。

（2）$R(\tau)$是轨迹 τ 的回报，$R(\tau)=r_1+\gamma r_2+\cdots+\gamma^{T-1}r_T=\sum_{t=0}^{T-1}\gamma^t\cdot r_{t+1}$ 。

（3）$p(\tau;\theta)$是轨迹 τ 的发生概率，与策略模型的参数 θ 有关，因为策略影响了动作的选择和轨迹的形成。

假设以 $\pi_\theta(a\,|\,s)$ 表示策略模型，忽略状态 s_0 的初始概率，可以计算出**轨迹 $\boldsymbol{\tau}$ 的概率**：

$$\begin{aligned}p(\tau;\theta)&=p(\{s_0,a_0,r_1,s_1,a_1,\cdots,r_{T-1},s_{T-1},a_{T-1},r_T,s_T\};\theta)\\&=\prod_{t=0}^{T-1}\underbrace{\pi_\theta(a_t\,|\,s_t)}_{\text{动作选择概率}}\cdot\underbrace{P(s_{t+1}\,|\,s_t,a_t)}_{\text{状态转移概率}}\end{aligned} \tag{0.2}$$

其中:

（1）$\pi_\theta(a_t\,|\,s_t)$ 是策略模型 π 在状态 s_t 下选择动作 a_t 的概率。

（2）$P(s_{t+1}\,|\,s_t,a_t)$ 是在状态 s_t 执行动作 a_t 后转移到状态 s_{t+1} 的概率。

（3）$\prod$ 是从起始时间步 t=0 到时间步 t=T−1 的所有动作选择概率和状态转移概率的乘积（注意：终止状态是 s_T）。

为了最大化预期回报，需要通过逐步训练并调整参数 θ，以最大化目标 $J(\theta)$。对目标函数 $J(\theta)$关于参数 θ 求**梯度**：

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta\Big(\sum_\tau p(\tau;\theta)\cdot R(\tau)\Big) \\
&= \sum_\tau R(\tau)\cdot \nabla_\theta p(\tau;\theta) \\
&= \sum_\tau R(\tau)\cdot p(\tau;\theta)\cdot \frac{\nabla_\theta p(\tau;\theta)}{p(\tau;\theta)} \\
&= \sum_\tau R(\tau)\cdot p(\tau;\theta)\cdot \nabla_\theta \log p(\tau;\theta) \\
&= \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[R(\tau)\cdot \nabla_\theta \log p(\tau;\theta)\right]
\end{aligned} \tag{0.3}
$$

其中，$\dfrac{\nabla_\theta p(\tau;\theta)}{p(\tau;\theta)} = \nabla_\theta \log p(\tau;\theta)$，由导数链式法则 $\nabla(\log f(x)) = \dfrac{1}{f(x)}\cdot \nabla f(x)$ 可得。即得到**策略梯度公式的形式 1**：

$$
\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[R(\tau)\cdot \nabla_\theta \log p(\tau;\theta)\right] \tag{0.4}
$$

将式（0.2）代入式（0.4），可得

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[R(\tau)\cdot \nabla_\theta \log p(\tau;\theta)\right] \\
&= \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[R(\tau)\cdot \nabla_\theta\left(\sum_{t=0}^{T-1}\log \pi_\theta(a_t\mid s_t) + \sum_{t=0}^{T-1}\log \underbrace{P(s_{t+1}\mid s_t,a_t)}_{\text{与}\theta\text{无关，求导为0}}\right)\right] \\
&= \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[R(\tau)\cdot \nabla_\theta\left(\sum_{t=0}^{T-1}\log \pi_\theta(a_t\mid s_t)\right)\right] \\
&= \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t\mid s_t)\cdot R(\tau)\right]
\end{aligned} \tag{0.5}
$$

即得到**策略梯度公式的形式 2**，被称为**策略梯度定理**（Policy Gradient Theorem）：

$$
\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[\underbrace{\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t\mid s_t)}_{\text{动作偏好梯度}}\cdot \underset{\text{轨迹回报}}{R(\tau)}\right] \tag{0.6}
$$

1.4 策略梯度的计算

如图 0.2 所示，在实际计算时，假设通过实际运行收集到 N 条轨迹（记为 $\tau^{(1)}, \tau^{(2)}, \tau^{(3)}, \dots, \tau^{(N)}$），第 i 条轨迹

$\tau^{(i)}=\left(s_0^{(i)},a_0^{(i)},r_1^{(i)},s_1^{(i)},a_1^{(i)},r_2^{(i)},\ldots,a_{T-1}^{(i)},r_T^{(i)},s_T^{(i)}\right)$，则基于式（0.6），可以求得 $\nabla_\theta J(\theta)$：

$$\nabla_\theta J(\theta)\approx\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=0}^{T^{(i)}-1}\nabla_\theta\log\pi_\theta(a_t^{(i)}\mid s_t^{(i)})\cdot R(\tau^{(i)})\right)\tag{0.7}$$

其中：

（1）$a_t^{(i)}$ 和 $s_t^{(i)}$ 分别表示轨迹 $\tau^{(i)}$ 中采集到的动作和状态。

（2）$R(\tau^{(i)})$ 是第 i 条轨迹 $\tau^{(i)}$ 对应的回报（累计折扣奖励）。

（3）$T^{(i)}$是第 i 条轨迹 $\tau^{(i)}$ 对应的终止时间步。

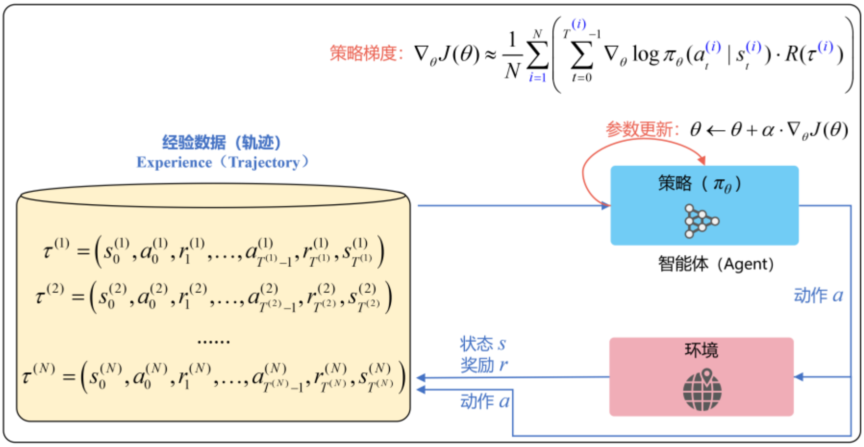

图 0.2 策略梯度原理

如果仅针对一条轨迹，则**一条轨迹的策略梯度**公式简化为

$$\nabla_\theta J(\theta)=\sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_t\mid s_t)\cdot R(\tau)\tag{0.8}$$

优化的目标是最大化 $J(\theta)$，即最大化回报，则需要采用梯度上升法（因为策略梯度 $\nabla_\theta J(\theta)$ 的方向是 $J(\theta)$增加最快的方向）。通过梯度上升更新策略模型的参数 θ：

$$\theta\leftarrow\theta+\alpha\cdot\nabla_\theta J(\theta)\tag{0.9}$$

其中，α 是学习率。随着策略模型参数 θ 逐步更新，策略模型 $\pi_\theta(a\mid s)$ 越来越接近最优策略，从而使得预期回报 $J(\theta)$不断提高。如果使用深度神经网络实现策略函数（策略模型），在训练过程中，诸如 PyTorch 等深度学习框架将自动计算梯度并更新参数 θ。

1.5 策略梯度与传统梯度有何不同?

与一般的梯度计算不同，在策略梯度下，动作偏好梯度 $\sum\nabla_\theta \log\pi_\theta(a_t \mid s_t)$ 还需要乘以回报 $R(\tau)$进行缩放，这意味着参数更新的幅度不仅取决于动作偏好梯度本身，还受到轨迹回报 $R(\tau)$的影响。假设轨迹 $\tau=\{s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3\}$，根据轨迹 τ 的回报 $R(\tau)$的正负不同，结合式（0.6），可知：

（1）**当 $R(\tau)>0$ 时**：则式（0.9）的更新方向与“动作偏好梯度”的方向**相同**，参数更新后，会增加在各状态下执行对应动作的概率，即**增加** $\pi_\theta(a_0 \mid s_0)$、$\pi_\theta(a_1 \mid s_1)$、$\pi_\theta(a_2 \mid s_2)$。

（2）**当 $R(\tau)<0$ 时**：则式（0.9）的更新方向与“动作偏好梯度”的方向**相反**，参数更新后，会减少在各状态下执行对应动作的概率，即**减少** $\pi_\theta(a_0 \mid s_0)$、$\pi_\theta(a_1 \mid s_1)$、$\pi_\theta(a_2 \mid s_2)$。

通过这种方式，模型参数能够朝着获得更高回报的方向进行优化，从而逐步提升策略模型的效果。这种结合回报缩放的策略梯度更新方式，使得模型不仅能够学习动作偏好的方向，还能够依据实际回报大小灵活调整优化幅度，从而提高训练效率和性能。

1.6 策略梯度的应用：REINFORCE 和 Actor-Critic

基于策略梯度定理，衍生出了两类经典算法：REINFORCE 和 Actor-Critic。

如式（0.6）所示，在计算策略梯度 $\nabla_\theta J(\theta)$ 时，需要先计算轨迹回报 $R(\tau)$，轨迹回报 $R(\tau)$的计算方法主要有两种：

（1）**蒙特卡洛方法求解 $R(\tau)$（REINFORCE 方法）**：原理如图 0.2 所示，直接基于实际运行，采集多条完整的轨迹，这些轨迹均是一个完整的回合（episode），计算每条轨迹的实际回报 $R(\tau)$，并按照式（0.7）进行梯度计算和参数更新，这种方法被称为 **REINFORCE**（名称源于 REward Increment = Nonnegative Factor × Offset Reinforcement × Characteristic Eligibility）[5]。REINFORCE 算法属于同策略方法，这意味着在学习过程中，算法必须使用当前正在评估和优化的策略来收集经验。该方法具有无偏性，但由于依赖完整轨迹，方差较大。实际使用

时，可引入基线（baseline）来减小方差。

（2）**使用神经网络估计 $R(\tau)$（Actor-Critic 架构）**：通过引入一个价值模型（Critic），使用深度神经网络近似状态价值函数 $V_\pi(s)$或动作价值函数 $Q_\pi(s, a)$，以估计从当前状态（或状态-动作对）开始的期望回报，这种方法被称为 Actor-Critic 架构的方法。其中，Actor 更新策略参数 θ，Critic 评估策略 π_θ 的价值。通过使用 Critic，可以减少回报估计的方差，提高训练效率。

1.7 AC、A2C、Advantage（优势）

Actor-Critic 架构（**AC 架构**），即演员-评委架构，是一种应用极为广泛的强化学习架构，知名的 PPO、DPG、DDPG、TD3、A2C、A3C、SAC 等算法均基于 Actor-Critic 架构。

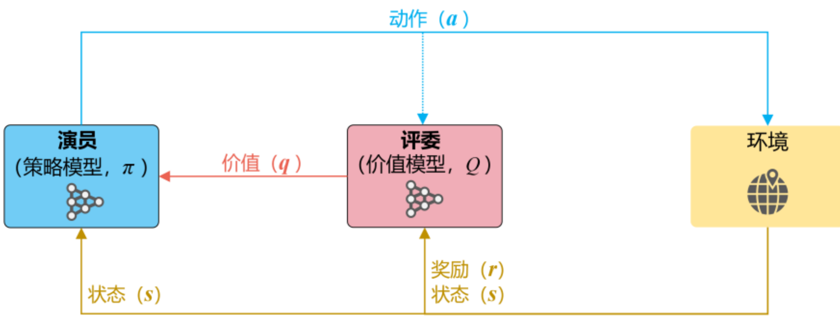

图 0.3 Actor-Critic 架构原理

如上图所示，该架构基于两个角色的模型：

（1）**Actor**（演员）：对应于策略模型 π，负责选择动作，直接输出策略 $\pi(a\,|\,s)$，即在给定状态 s 下选择动作 a 的概率分布。

（2）**Critic**（评委）：对应于价值模型 Q，评估 Actor 执行的动作的好坏，这可以协助 Actor 逐步优化策略模型的参数。

策略梯度 $\nabla_\theta J(\theta)$ 的求解依赖于轨迹回报 $R(\tau)$，可以通过 REINFORCE 方法计算 $R(\tau)$。而 REINFORCE 方法基于蒙特卡洛方法，具有较大的方差。

与 REINFORCE 方法不同，另一种方法是直接使用一个深度神经网络来估计回报 $R(\tau)$。可以借鉴基于价值的算法，例如 DQN，使用一个价值模型来近似动作价值函数，以估计从当前状态（或状态-动作对）开始的回报。则式（0.6）变为

$$\nabla_\theta J(\theta)=\mathbb{E}_{\tau\sim p(\tau;\theta)}\left[\underbrace{\sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_t\mid s_t)}_{\text{动作偏好梯度}}\cdot\underbrace{Q_w(s_t,a_t)}_{\text{价值}}\right] \tag{0.10}$$

这就是**基础版本的 Actor-Critic 架构**方法。其中，$\nabla_\theta J(\theta)$ 表示策略梯度，π_θ 表示策略模型（Actor, 演员），θ 为模型参数；Q_w 表示价值模型（Critic, 评委），w 为模型参数。

基础版本的 Actor-Critic 架构存在高方差等问题。为了解决这些问题，**A2C**（Advantage Actor-Critic）方法在 Actor-Critic 的基础上引入基线（Baseline），并进一步构造了优势函数。具体步骤如下：

（1）**引入基线**：基线通常采用状态价值函数 $V(s)$，即在状态 s 下的预期回报。将基线设为 $V(s)$，相当于在每个状态 s 下设定了一个“平均水平”。

（2）**构造优势函数**：优势函数衡量在给定状态下，采取某一动作相对于“平均水平”的优劣，即某个动作 a 相对于特定状态下其他动作的相对优势。优势函数关注的是执行某个动作的相对优势，而非动作的绝对价值。优势函数的定义为

$$A(s_t,a_t)=Q(s_t,a_t)-V(s_t) \tag{0.11}$$

（3）**构建 A2C 方法的策略梯度**：使用优势函数 $A(s_t,a_t)$ 替换式（0.10）中的 $Q(s_t,a_t)$，可得 **A2C 方法中的策略梯度公式**：

$$\nabla_\theta J(\theta)=\mathbb{E}_{\tau\sim p(\tau;\theta)}\left[\underbrace{\sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_t\mid s_t)}_{\text{动作偏好梯度}}\cdot\underbrace{A(s_t,a_t)}_{\text{优势}}\right] \tag{0.12}$$

1.8 策略梯度算法一览

首先，我们简要回顾 TRPO、PPO 与 GRPO 算法。策略梯度、TRPO 、PPO 的演进过程如下。

图 0.4 策略梯度、TRPO、PPO 算法的演进

2015 年，John Schulman 等人在策略梯度方法的基础上，引入置信域和重要性采样技术，提出了 **TRPO**（Trust Region Policy Optimization，**置信域策略优化**）算法。TRPO 的核心思想是在最大化目标函数 $J(\theta)$的同时，限制新旧策略之间的差异。其优化目标可以表示为一个带约束的优化问题，**TRPO 的目标函数：**

$$J(\theta)=\mathbb{E}_t\left[\frac{\pi_\theta(a_t|\,s_t)}{\pi_{\theta_{\text{old}}}(a_t|\,s_t)}A_t^{\pi_{\theta_{\text{old}}}}\right]$$

随后在 2017 年，John Schulman 与 OpenAI 的其他研究人员进一步优化了 TRPO 算法，通过引入 KL 散度惩罚项和剪裁（Clip）机制，分别提出了**两种形式的 PPO 算法**——PPO-Penalty 和 PPO-Clip。其中，PPO-Clip 因其更优的效果而获得了更多关注和应用，因此通常所说的 PPO 即指 PPO-Clip。**PPO-Clip 的目标函数：**

$$J^{\text{PPO-Clip}}(\theta)=$$

$$\mathbb{E}_t\left[\min\left\{\frac{\pi_\theta(a_t\,|\,s_t)}{\pi_{\theta_{\text{old}}}(a_t\,|\,s_t)}\cdot A_t^{\pi_{\theta_{\text{old}}}},\ \text{clip}\left(\frac{\pi_\theta(a_t\,|\,s_t)}{\pi_{\theta_{\text{old}}}(a_t\,|\,s_t)},1-\varepsilon,1+\varepsilon\right)\cdot A_t^{\pi_{\theta_{\text{old}}}}\right\}\right]$$

随后在 2024 年，DeepSeek 团队基于 PPO 算法进行改进，提出 GRPO（Group Relative Policy Optimization, **群体相对策略优化**）算法，与 PPO 相比，GRPO 摒弃了显式的价值网络（Critic），转而使用组内相对奖励差值计算优势（Advantage），并且将 KL 散度作为正则项直接融入损失函数，更为直观简洁，显著降低了训练资源和实施难度。**GRPO 的目标函数：**

$$J^{\text{GRPO}}(\theta)=$$

$$\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|\,q)}{\pi_{\theta_{\text{old}}}(o_i|\,q)}\cdot A_i,\ \text{clip}\left(\frac{\pi_\theta(o_i|\,q)}{\pi_{\theta_{\text{old}}}(o_i|\,q)},1-\varepsilon,1+\varepsilon\right)\cdot A_i\right)-\beta\cdot\text{KL}\left(\pi_\theta\,\|\,\pi_{\text{ref}}\right)\right)\right]$$

可进一步简化上述几个函数的形式，定义**策略概率比**（Probability

Ratio）如下：

$$r_t(\theta) = \frac{\pi_\theta(a_t|\ s_t)}{\pi_{\theta_{\text{old}}}(a_t|\ s_t)}$$

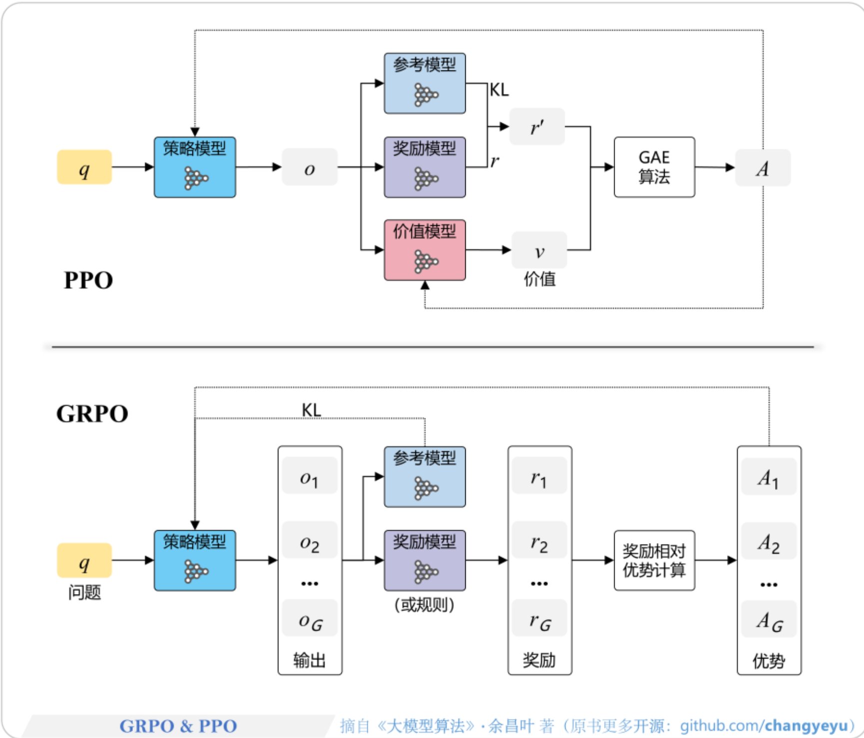

图 0.5 PPO 与 GRPO

通过以上 TRPO、PPO、GRPO 的目标函数，可能还是很难将它们与策略梯度关联起来。

笔者将逐一写出它们的梯度形式，以便于读者更直观地理解和对比。

根据以上各个目标函数 $J(\theta)$，对 $J(\theta)$ 求导，综合其他策略梯度算法的公式一并整理，得出策略**梯度定理、DPG、AC、A2C、SAC、TRPO、PPO、GRPO** 的梯度形式 $\nabla_\theta J(\theta)$ 如下所示：

$$\textbf{策略梯度定理}:\quad \nabla_\theta J(\theta)=\mathbb{E}\left[\sum_{t=0}^{T-1}\nabla_\theta \log\pi_\theta(a_t\mid s_t)\bullet R(\tau)\right]$$

$$\textbf{DPG}:\quad \nabla_\theta J(\theta)=\mathbb{E}_s\left[\nabla_\theta\pi_\theta(s)\bullet\nabla_a Q_w(s,a)\big|_{a=\pi_\theta(s)}\right]$$

$$\textbf{Actor-Critic}:\quad \nabla_\theta J(\theta)=\mathbb{E}\left[\sum_{t=0}^{T-1}\nabla_\theta \log\pi_\theta(a_t\mid s_t)\bullet Q_w(s_t,a_t)\right]$$

$$\textbf{A2C}:\quad \nabla_\theta J(\theta)=\mathbb{E}\left[\sum_{t=0}^{T-1}\nabla_\theta \log\pi_\theta(a_t\mid s_t)\bullet A(s_t,a_t)\right]$$

$$\textbf{SAC}:\quad \nabla_\theta J(\theta)=\mathbb{E}\left[\sum_{t=0}^{T-1}\nabla_\theta \log\pi_\theta(a_t\mid s_t)\bullet(Q_w(s_t,a_t)-V)\right]$$

$$\textbf{TRPO}:\quad \nabla_\theta J(\theta)=\mathbb{E}\left[\sum_{t=0}^{T-1}\nabla_\theta \log\pi_\theta(a_t\mid s_t)\, r_t(\theta)\bullet A_t^{\pi_{\text{old}}}\right]$$

$$\textbf{PPO-Clip}:\quad \nabla_\theta J(\theta)=\mathbb{E}\left[\sum_{t=0}^{T-1}\nabla_\theta \log\pi_\theta(a_t\mid s_t)\,\mathbf{1}_{\text{not-clip}}\, r_t(\theta)\bullet A_t\right]$$

$$\textbf{GRPO}:\quad \nabla_\theta J(\theta)=\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\nabla_\theta \log\pi_\theta(o_i\mid q)\,\mathbf{1}_{\text{not-clip}(i)}\, r_i(\theta)\bullet A_i\right]-\beta\nabla_\theta\mathrm{KL}\left(\pi_\theta(\cdot\mid q)\|\pi_{\text{ref}}(\cdot\mid q)\right)$$

其中，$\mathbf{1}_{\text{not-clip}}$ 指示“未被裁剪且会提升目标”的区域，被裁剪分支视作常数不回传梯度。可以表示如下：

$$\mathbf{1}_{\text{not-clip}}(r_t,A_t)=\begin{cases}1, & (A_t\ge 0 \text{ and } r_t(\theta)\le 1+\varepsilon)\ \text{ or }\ (A_t<0 \text{ and } r_t(\theta)\ge 1-\varepsilon),\\ 0, & \text{otherwise.}\end{cases}$$

至此，我们已梳理并推导了策略梯度及其主要衍生算法的原理。通过对最终梯度表达式的对比，可以清晰地看出：**策略梯度**系列算法拥有同样的思想和算法内核——**将“策略”视为可微参数模型，并沿着能提升期望回报的梯度方向直接更新参数。**

参考文献

[1] 《大模型算法：强化学习、微调与对齐》,余昌叶,电子工业出版社
[2] LLM-RL-Visualized :https://github.com/changyeyu/LLM-RL-Visualized
[3] Reinforcement Learning: An Introduction, 2nd Edition, Richard S. Sutton：http://incompleteideas.net/book/the-book-2nd.html
[4] Policy Gradient Theorem ： https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf
[5] Policy Gradient&REINFORCE： https://people.cs.umass.edu/~barto/courses/cs687/williams92simple.pdf
[6] DPG： https://proceedings.mlr.press/v32/silver14.pdf
[7] AC架构： http://www.derongliu.org/adp/adp-cdrom/Barto1983.pdf
[8] SAC： https://arxiv.org/pdf/1801.01290
[9] A3C： https://arxiv.org/pdf/1602.01783
[10] TRPO： https://arxiv.org/pdf/1502.05477
[11] PPO： https://arxiv.org/abs/1707.06347
[12] GRPO： https://arxiv.org/pdf/2402.03300