GRPO 教材

从数学预备、PPO 背景到公式推导与手算例题

主题：Group Relative Policy Optimization

面向对象：希望系统理解 GRPO 的学习者与研究者

本教材的写法遵循三个原则：
1. 先把需要的数学工具全部铺平，再讲 GRPO。
2. 所有关键公式都给出符号解释，并尽量把“为什么这样写”展开。
3. 对原始论文、后续 DeepSeek-R1 叙述、以及实践实现之间的差异，明确区分“直观说法”和“严格说法”。

版本：1.0
日期：2026-05-10

目录

前言

GRPO（Group Relative Policy Optimization）最早由 DeepSeekMath 提出，用于在大语言模型的强化学习阶段替代传统 PPO 中昂贵的 value/critic 估计，同时保留 PPO 风格的稳定更新机制 [1]。后来 DeepSeek-R1/R1-Zero 继续采用 GRPO 作为核心推理强化学习框架 [4]。如果只看一句口号，GRPO 常被概括为：

> 在同一个 prompt 下采样多条回答，用组内相对奖励构造优势（advantage），再用 PPO 风格的 clipping 与参考模型 KL 约束来更新策略。

但这句话虽然抓住了直觉，却不足以支撑真正的理解。要把 GRPO 学明白，至少要回答下面几类问题：

1. 为什么 LLM 的 RL 阶段会自然走到 PPO 这条路线？
2. PPO 里的 critic/value model 到底在做什么？
3. 为什么可以（至少在工程上）不用 critic，而改用同题多样本的组内比较？
4. 原始 DeepSeekMath 的 GRPO 目标函数究竟长什么样？
5. 为什么很多后续实现、教程、代码里的写法又和原论文不完全一样？

本教材按照“数学预备 → RL/PPO 背景 → GRPO 原式 → 推导 → 手算 → 局限与变体”的顺序展开。

第 1 章　数学预备：读 GRPO 前必须会的工具

1.1　符号总表

符号	含义
q	一个问题、prompt、或训练样本输入。
o	一个完整输出（completion），也可理解为一条回答轨迹。
o_t	输出序列在第 t 个位置的 token。
$o_{<t}$	第 t 个 token 之前的前缀。
π_θ	参数为 θ 的当前策略模型（当前语言模型）。
$\pi_{\theta_{\text{old}}}$	采样当前训练 batch 时所使用的旧策略。
π_{ref}	冻结的参考策略，常是初始 SFT 模型或上一个冻结检查点。
$R(q, o)$	对完整输出 o 的奖励。
r_i	第 i 条采样回答的原始奖励，通常记为 $r_i = R(q, o_i)$。
$\hat{A}_{i,t}$	第 i 条输出在第 t 个 token 处的 advantage。
G	同一个 prompt 下采样的输出个数（group size）。
$\lvert o_i \rvert$	第 i 条输出的 token 长度。
ϵ	PPO/GRPO 中 clipping 的上下界超参数。
β	KL 正则项的系数。
μ	原始 GRPO 伪代码中，同一轮生成后对同一批样本进行的优化迭代次数。
$\mathbb{E}[\cdot]$	期望。
$\text{Var}[\cdot]$	方差。
$\text{Std}[\cdot]$	标准差。
$\text{D}_{\text{KL}}(p\Vert q)$	Kullback–Leibler 散度。

1.2　随机变量、期望、方差与标准化

GRPO 的核心计算中最常出现的三个量是：均值、方差、标准差。

设一组数为 $x_1, x_2, \ldots, x_n$，则：

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1.1}$$

称为均值。

$$\mathrm{Var}(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{1.2}$$

称为方差。

$$\mathrm{Std}(x) = \sqrt{\mathrm{Var}(x)} \tag{1.3}$$

称为标准差。
如果我们把每个值减去均值再除以标准差，就得到标准化后的值：

$$z_i = \frac{x_i - \mu}{\mathrm{Std}(x)} \tag{1.4}$$

这个操作有两个直接效果：

(a) 让整体中心移到 0（因为标准化后的均值为 0）；

(b) 让尺度变得可比（标准差归一到 1）。

GRPO 中正是把同一个 prompt 下多条回答的奖励做了类似标准化。

手算例题

设四条回答的奖励分别为

$$0.2, \quad 0.8, \quad 0.5, \quad -0.1.$$

先算均值：

$$\mu = \frac{0.2 + 0.8 + 0.5 - 0.1}{4} = \frac{1.4}{4} = 0.35.$$

再算离均差：

$$0.2-0.35 = -0.15, \qquad 0.8-0.35 = 0.45, \qquad 0.5-0.35 = 0.15, \qquad -0.1-0.35 = -0.45.$$

平方后得到：

$$0.0225, \quad 0.2025, \quad 0.0225, \quad 0.2025.$$

于是方差：

$$\mathrm{Var} = \frac{0.0225 + 0.2025 + 0.0225 + 0.2025}{4} = \frac{0.45}{4} = 0.1125.$$

标准差：

$$\mathrm{Std} = \sqrt{0.1125} \approx 0.3354.$$

标准化结果：

$$\frac{0.2-0.35}{0.3354}\approx -0.4472,\quad \frac{0.8-0.35}{0.3354}\approx 1.3416,$$
$$\frac{0.5-0.35}{0.3354}\approx 0.4472,\quad \frac{-0.1-0.35}{0.3354}\approx -1.3416.$$

这组标准化数就是典型的 GRPO 风格“组内相对优势”。

1.3　条件概率与语言模型的序列分解

语言模型不是一次性“拍脑袋”生成整句答案，而是一个 token 一个 token 地生成。因此一个完整输出 $o=(o_1,o_2,\dots,o_T)$ 的概率不是一个原子量，而是一个条件概率连乘：

$$\pi_\theta(o|q)=\prod_{t=1}^{T}\pi_\theta(o_t\,|\,q,o_{<t}). \tag{1.5}$$

这是理解 PPO/GRPO 在 LLM 中如何落地的第一步。
上式的意思是：

1. 先在 prompt q 条件下生成第一个 token o_1；
2. 再在 (q,o_1) 条件下生成 o_2；
3. 再在 (q,o_1,o_2) 条件下生成 o_3；
4. 如此递推，直到输出结束。

对数形式更常用，因为乘法会转成求和：

$$\log\pi_\theta(o|q)=\sum_{t=1}^{T}\log\pi_\theta(o_t\,|\,q,o_{<t}). \tag{1.6}$$

这非常重要，因为 policy gradient 最终会作用到每一个 token 的对数概率上。

手算例题

若某条回答只有 3 个 token，并且

$$\pi_\theta(o_1|q)=0.4,\quad \pi_\theta(o_2|q,o_1)=0.5,\quad \pi_\theta(o_3|q,o_1,o_2)=0.2,$$

那么整条序列的概率是

$$\pi_\theta(o|q)=0.4\times 0.5\times 0.2=0.04.$$

其对数概率为

$$\log\pi_\theta(o|q)=\log 0.4+\log 0.5+\log 0.2.$$

所以，只要我们让这条回答的总 logprob 上升，本质上就是在提高它在模型中的整

体生成概率。

1.4 梯度与 log-derivative trick

强化学习里一个极其关键的小技巧是：

$$\nabla_\theta \pi_\theta(x) = \pi_\theta(x) \nabla_\theta \log \pi_\theta(x). \tag{1.7}$$

证明非常短：

$$\log \pi_\theta(x) \quad \Longrightarrow \quad \nabla_\theta \log \pi_\theta(x) = \frac{\nabla_\theta \pi_\theta(x)}{\pi_\theta(x)}.$$

两边同乘 $\pi_\theta(x)$ 即得式 (1.7)。

为什么这个式子如此重要？因为强化学习的目标通常长成

$$\sum_x \pi_\theta(x) f(x)$$

或

$$\mathbb{E}_{x\sim\pi_\theta}[f(x)],$$

对它求梯度时，最难处理的地方就在概率分布本身依赖参数 θ。有了 (1.7)，很多求导都能改写成“奖励 $\times$ 对数概率梯度”的形式。

1.5 KL 散度

KL 散度衡量两个分布的差异。对离散分布 p, q 而言：

$$\mathrm{D}_{\mathrm{KL}}(p\|q) = \sum_x p(x) \log \frac{p(x)}{q(x)}. \tag{1.8}$$

它不是严格对称的，因此一般

$$\mathrm{D}_{\mathrm{KL}}(p\|q) \neq \mathrm{D}_{\mathrm{KL}}(q\|p).$$

在 PPO/GRPO 里，KL 的角色通常是“别偏得太远”。也就是说，新的策略模型可以变好，但不应该一口气偏离参考模型太多，否则容易出现奖励投机（reward hacking）、语言退化或不稳定训练。

DeepSeekMath 中采用了如下单点近似/估计形式 [1]：

$$\mathrm{D}_{\mathrm{KL}}[\pi_\theta\|\pi_{\mathrm{ref}}] \approx \frac{\pi_{\mathrm{ref}}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})} - \log \frac{\pi_{\mathrm{ref}}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})} - 1. \tag{1.9}$$

严格说法

令

$$x = \frac{\pi_{\text{ref}}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})},$$

定义

$$g(x) = x - \log x - 1, \qquad x > 0.$$

则

$$g'(x) = 1 - \frac{1}{x}, \qquad g''(x) = \frac{1}{x^2} > 0.$$

因此 g 是凸函数，并且在 $g'(x) = 0$ 即 $x = 1$ 时达到最小值。又因为

$$g(1) = 1 - \log 1 - 1 = 0,$$

故对所有 $x > 0$，都有

$$x - \log x - 1 \geq 0.$$

这解释了为什么式 (1.9) 产生的是一个非负惩罚量。

1.6　重要性采样与概率比值

PPO 与 GRPO 中最醒目的符号之一就是概率比值：

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}. \tag{1.10}$$

它来自重要性采样（importance sampling）。

假设我们原本想计算

$$\mathbb{E}_{x\sim p}[f(x)] = \sum_x p(x) f(x),$$

但实际上手上的样本来自另一个分布 q。只要 $q(x) > 0$ 的地方 $p(x)$ 也定义良好，就可以写成

$$\mathbb{E}_{x\sim p}[f(x)] = \sum_x q(x) \frac{p(x)}{q(x)} f(x) = \mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)} f(x)\right]. \tag{1.11}$$

PPO/GRPO 正是用旧策略 $\pi_{\theta_{\text{old}}}$ 采样，再用比值

$$\frac{\pi_\theta}{\pi_{\theta_{\text{old}}}}$$

把目标换算回当前策略对应的方向。

容易混淆的点

在 LLM 场景里，完整序列的概率比值其实是所有 token 条件概率比值的连乘：

$$\frac{\pi_\theta(o|q)}{\pi_{\theta_{\text{old}}}(o|q)} = \prod_{t=1}^{|o|} \frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\theta_{\text{old}}}(o_t|q, o_{<t})}.$$

但直接在整条序列级别做这个比值，方差通常很大，因此原始 PPO/GRPO 都更偏向在 token 级别使用 ratio。

1.7 一个最小可用心智模型

学完本章，你至少应该熟悉下面 5 个数学部件：

1. 语言模型输出概率的自回归分解；
2. 对数概率是逐 token 求和；
3. log-derivative trick；
4. 组内标准化；
5. 概率比值与 KL 正则。

这 5 个部件拼起来，几乎就是整个 GRPO 的骨架。

第 2 章　从 RLHF 到 PPO：GRPO 的背景

2.1　把语言模型看成策略

在强化学习语言里，策略（policy）就是“给定状态时，行动的概率分布”。在 LLM 场景中，最自然的映射是：

状态 s_t	当前 prompt 与已有前缀，即 $(q, o_{<t})$
动作 a_t	下一个 token o_t
策略 $\pi_\theta(a_t\|s_t)$	模型在当前位置输出某 token 的概率
轨迹 τ	一整条输出 $o = (o_1, \ldots, o_T)$
奖励 $R(q, o)$	对完整回答的评分

于是，一个最基本的训练目标就是提升期望奖励：

$$J(\theta) = \mathbb{E}_{q\sim P(Q),\, o\sim\pi_\theta(\cdot|q)}[R(q, o)]. \tag{2.1}$$

2.2　REINFORCE：最原始的策略梯度

为了理解 PPO 和 GRPO，最好先从最朴素的策略梯度开始。固定一个问题 q，定义

$$J_q(\theta) = \sum_o \pi_\theta(o|q) R(q, o). \tag{2.2}$$

对 θ 求梯度：

$$\begin{aligned}
\nabla_\theta J_q(\theta) &= \nabla_\theta \sum_o \pi_\theta(o|q) R(q, o) && (2.3)\\
&= \sum_o R(q, o) \nabla_\theta \pi_\theta(o|q) && (2.4)\\
&= \sum_o R(q, o) \pi_\theta(o|q) \nabla_\theta \log \pi_\theta(o|q) && (2.5)\\
&= \mathbb{E}_{o\sim\pi_\theta(\cdot|q)}\big[R(q, o) \nabla_\theta \log \pi_\theta(o|q)\big]. && (2.6)
\end{aligned}$$

再对问题分布 $P(Q)$ 取期望，就得到：

$$\nabla_\theta J(\theta) = \mathbb{E}_{q\sim P(Q),\, o\sim\pi_\theta(\cdot|q)}\big[R(q, o) \nabla_\theta \log \pi_\theta(o|q)\big]. \tag{2.7}$$

利用式 (1.6)，还可以继续拆成 token 级别：

$$\nabla_\theta \log \pi_\theta(o|q) = \sum_{t=1}^{|o|} \nabla_\theta \log \pi_\theta(o_t|q, o_{<t}). \tag{2.8}$$

代回去：

$$\nabla_\theta J(\theta) = \mathbb{E}\left[R(q,o) \sum_{t=1}^{|o|} \nabla_\theta \log \pi_\theta(o_t|q, o_{<t}) \right]. \tag{2.9}$$

这说明：如果一条完整回答拿到了高奖励，那么它的每个 token 的对数概率都会被整体推高；如果奖励低，则整体被压低。

2.3 为什么要引入 baseline

式 (2.7) 虽然正确，但方差通常很大。直觉上，如果奖励本身波动很剧烈，那么梯度估计就很吵。最常见的降方差方法是减去一个 baseline：

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\big(R(q,o) - b(q)\big) \nabla_\theta \log \pi_\theta(o|q) \right]. \tag{2.10}$$

其中 $b(q)$ 只依赖问题 q，不依赖当前采样出的具体输出 o。为什么这种减法不改变期望？因为：

$$\begin{aligned}
\mathbb{E}_{o\sim\pi_\theta(\cdot|q)}\big[b(q) \nabla_\theta \log \pi_\theta(o|q) \big] &= b(q) \sum_o \pi_\theta(o|q) \nabla_\theta \log \pi_\theta(o|q) && (2.11)\\
&= b(q) \sum_o \nabla_\theta \pi_\theta(o|q) && (2.12)\\
&= b(q) \nabla_\theta \sum_o \pi_\theta(o|q) && (2.13)\\
&= b(q) \nabla_\theta 1 = 0. && (2.14)
\end{aligned}$$

所以，

$$\mathbb{E}\left[\big(R - b(q)\big) \nabla_\theta \log \pi_\theta \right] = \mathbb{E}\left[R \nabla_\theta \log \pi_\theta \right]. \tag{2.15}$$

这就是 baseline 的基本理论来源。

容易混淆的点

这里要特别注意：标准 baseline 定理要求 baseline 对当前采样动作是“条件独立”的，即只依赖状态、不依赖动作。而 GRPO 里常说的“用组均值作 baseline”更多是一个**直观说法**。严格地讲，原始 GRPO 是直接在同题多样本组上定义的相对目标函数，而不是简单把 PPO 的 $V(s)$ 机械替换成组均值后仍自动享受同样的无偏结论。后文会专门解释这一点。

2.4 advantage、value 与 critic

在 actor–critic 方法中，常把

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) \tag{2.16}$$

称为 advantage。它表示“当前动作比这个状态下的平均水平好多少”。
其中：

- $Q(s_t, a_t)$：在状态 s_t 执行动作 a_t 后，未来回报的期望；
- $V(s_t)$：只看状态 s_t 的平均回报；
- $A(s_t, a_t)$：动作相对状态平均水平的超额价值。

PPO 在一般强化学习中通常借助 value model（critic）来估计 $V(s_t)$，并通过 GAE 等方法构造 $\hat{A}_t$[2]。在 LLM 的 RLHF 场景里，这意味着训练时经常至少会同时维护：

策略模型（policy）+ 参考模型（reference）+ 奖励模型（reward model）+ 价值模型（value/critic）。

问题在于：对于长链式思维(long CoT)的大模型，这个 value/critic 代价很高。DeepSeek-Math 提出 GRPO 的直接动机之一就是：**能否去掉 critic，仍然得到有效的相对更新信号？**[1]

2.5 PPO：为什么 RLHF 常用它

PPO 的核心思想是：允许多步更新，但不让策略一次改动太猛 [2]。它的经典 surrogate objective 可以写为：

$$J_{\text{PPO}}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right], \tag{2.17}$$

其中

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}. \tag{2.18}$$

2.5.1 为什么要 clip

如果没有 clip，目标就像

$$r_t(\theta)\hat{A}_t.$$

当 $\hat{A}_t > 0$ 时，我们希望 $r_t(\theta)$ 变大，也就是提高这类动作的概率；当 $\hat{A}_t < 0$ 时，希望 $r_t(\theta)$ 变小。

问题在于，纯粹最大化 $r_t(\theta)\hat{A}_t$ 可能导致策略突然跳得非常远。于是 PPO 把它改成“原值”和“裁剪值”二者中的较小者（对最大化问题而言）：

$$\min\left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right).$$

2.5.2 正 advantage 时的分段理解

当 $\hat{A}_t > 0$ 时：

- 如果 $r_t(\theta) \le 1+\epsilon$，目标基本沿着 $r_t(\theta)\hat{A}_t$ 增长；
- 如果 $r_t(\theta) > 1+\epsilon$，则被截成 $(1+\epsilon)\hat{A}_t$，继续增大比值不再带来额外收益。

也就是说：**好动作可以被鼓励，但不能被鼓励得过头**。

2.5.3 负 advantage 时的分段理解

当 $\hat{A}_t < 0$ 时，情况反过来：

- 我们希望 $r_t(\theta)$ 变小，从而压低坏动作的概率；
- 但如果压得过猛，clip 会阻止其进一步“极端化”。

因此，PPO 的本质并不是“无条件追求奖励最大”，而是“在旧策略附近做一个受控的改进”。

2.6 PPO 在 LLM-RLHF 中为什么会显得重

在 InstructGPT 这类 RLHF 工作流里，PPO 被用于语言模型后训练 [3]。当它被直接搬到长 CoT 场景时，会遇到至少三类问题：

1. critic/value model 本身成本高；
2. 奖励通常是序列末端给出，而不是每个 token 天然都有精确信号；
3. 长回答会让价值估计、credit assignment 和训练显存都更困难。

GRPO 正是沿着这个痛点切入的。

第 3 章　GRPO：原始动机、核心公式与论文图示

3.1　DeepSeekMath 给出的出发点

DeepSeekMath 在第 4.1 节明确把 GRPO 描述为 PPO 的变体：它去掉了 critic/value model，不再显式学习价值函数，而是从同一个问题下采样出来的一组回答分数中估计相对优势 [1]。

直观理解

PPO 的直觉是：

$$\text{当前回答好不好} \approx \text{奖励} - \text{critic 给出的平均水平}.$$

GRPO 的直觉是：

$$\text{当前回答好不好} \approx \text{它在同题多条回答里排第几}.$$

也就是说，GRPO 不再问“它绝对值多高”，而更多问“它相对同题其它样本是更好还是更差”。

3.2 论文中的 PPO–GRPO 对照图

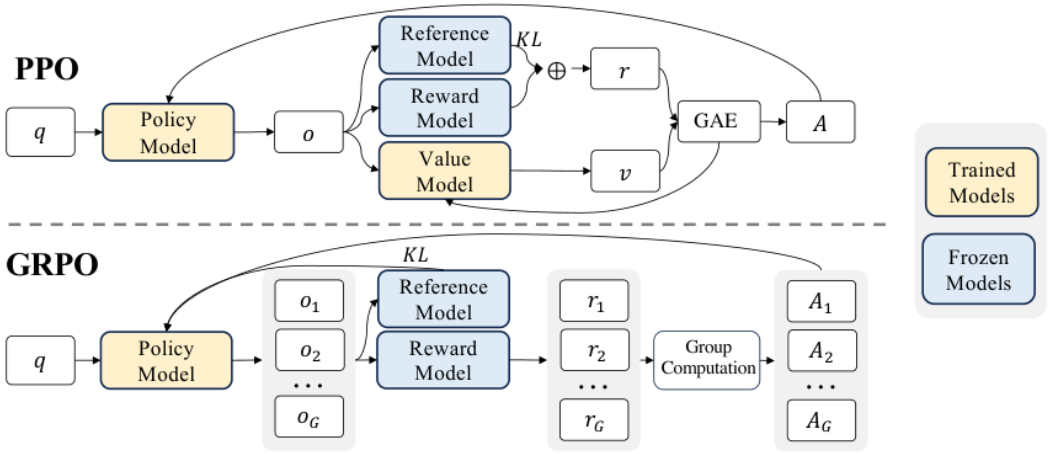

Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

on the rewards $\{r_{>t}\}$ and a learned value function V_{ψ}. Thus, in PPO, a value function needs to

图 3.1: DeepSeekMath 中对 PPO 与 GRPO 的结构对照。图中可以直接看出：GRPO 去掉了 value model，改为对同题多样本的奖励做 group computation，从而得到相对优势信号。图像裁剪自 Shao 等人的 DeepSeekMath 论文第 13 页 [1]。

这张图非常值得细读。它传达了 4 个关键信息：

1. PPO 需要 value model；GRPO 不需要。
2. PPO 一般采样单条输出并结合 value 估计 advantage；GRPO 对同一个问题采样一组输出。
3. GRPO 的奖励信号不是孤立使用，而是做组内相对计算。
4. 参考模型与奖励模型依然保留，因为它们分别承担“防止偏太远”和“给出训练方向”的作用。

3.3 原始 GRPO 目标函数

按照 DeepSeekMath 的写法，GRPO 的目标函数可以表述为 [1]：

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{\substack{q\sim P(Q),\\ \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(\min\left(r_{i,t}(\theta)\hat{A}_{i,t},\right.\right.\right. \\ \left.\left.\left. \text{clip}(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{i,t}\right) - \beta\, \mathrm{D}_{\text{KL}}[\pi_\theta\|\pi_{\text{ref}}]\right)\right], \tag{3.1}$$

其中

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})}. \tag{3.2}$$

现在我们逐项解释式 (3.1)：

1. $\{o_i\}_{i=1}^{G}$：同一个 prompt 下采样出的 G 条回答；
2. $\frac{1}{G}\sum_{i=1}^{G}$：对组中样本平均；
3. $\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}$：对同一回答内部的 token 平均（原始 GRPO 的一种长度归一方式）；
4. $\min(\cdots)$：继承 PPO 的 clipped surrogate；
5. $\hat{A}_{i,t}$：核心，表示第 i 条输出在第 t 个 token 的组相对 advantage；
6. $-\beta\, \mathrm{D}_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}]$：让当前策略别偏离参考策略太远。

3.4 Outcome supervision：最常见、也最容易理解的 GRPO

在 DeepSeekMath 的 outcome supervision 设定中，每条完整输出只拿到一个最终分数 r_i，然后在组内做标准化：

$$\mu_g = \frac{1}{G}\sum_{i=1}^{G} r_i, \qquad \sigma_g = \sqrt{\frac{1}{G}\sum_{i=1}^{G}(r_i - \mu_g)^2}. \tag{3.3}$$

再定义标准化奖励（也就是组相对 advantage 的最常见形式）：

$$\tilde{r}_i = \frac{r_i - \mu_g}{\sigma_g}. \tag{3.4}$$

随后，对该输出中的所有 token，统一赋值：

$$\hat{A}_{i,t} = \tilde{r}_i, \qquad t = 1, \ldots, |o_i|. \tag{3.5}$$

这意味着：在 outcome supervision 下，一条回答内部所有 token 分享同一个组相对奖励信号。

直观理解

如果第 i 条回答在同题组里明显更好，那么 $\tilde{r}_i > 0$，其所有 token 的 logprob 整体都被推高。
如果第 i 条回答明显更差，那么 $\tilde{r}_i < 0$，其所有 token 的 logprob 整体都被压低。

3.5 Process supervision：更细粒度的 credit assignment

DeepSeekMath 还讨论了 process supervision[1]。其想法是：不是只在回答结尾给一个总分，而是在若干“推理步骤结束位置”打分。设第 i 条输出有 K_i 个步骤，其步骤终止 token 的索引为

$$\text{index}_i(1), \ldots, \text{index}_i(K_i),$$

对应奖励记为

$$r_i^{\text{index}(1)}, \ldots, r_i^{\text{index}(K_i)}.$$

把组里所有步骤奖励汇总成集合 $\mathbf{R}$，先做标准化：

$$\tilde{r}_i^{\text{index}(j)} = \frac{r_i^{\text{index}(j)} - \text{mean}(\mathbf{R})}{\text{std}(\mathbf{R})}. \tag{3.6}$$

随后，某个 token 的 advantage 取为它“未来还能拿到的所有步骤奖励之和”：

$$\hat{A}_{i,t} = \sum_{\text{index}(j) \geq t} \tilde{r}_i^{\text{index}(j)}. \tag{3.7}$$

这比 outcome supervision 更细，因为它允许模型区分“哪个推理步骤对最终成功更有贡献”。

3.6 KL 正则为何直接加在 loss 里

DeepSeekMath 特别指出，GRPO 不是把 KL 罚项塞进 reward 再去算 advantage，而是直接把 KL 加进目标函数 [1]。这有一个好处：

不会把 advantage 的计算再搅复杂。

如果把 KL 混进 reward，那么组内标准化前后的量会变得更绕，因为 reward 中既有任务正确性，也有分布约束。而把 KL 直接放在 loss 中，就能把“学什么”和“别偏太远”这两件事更清楚地分开。

3.7 DeepSeek-R1 中的更简洁表述

DeepSeek-R1 的方法部分给出了更简洁的 GRPO 叙述：对每个问题 q 采样一组输出 $\{o_1, \ldots, o_G\}$，对每条输出给奖励 r_i，并用

$$A_i = \frac{r_i - \text{mean}(r_1, \ldots, r_G)}{\text{std}(r_1, \ldots, r_G)} \tag{3.8}$$

构造组相对优势 [4]。这与 DeepSeekMath 的 outcome supervision 直觉一致。不同之处主要在于：DeepSeek-R1 的叙述更像是“序列级摘要写法”，而 DeepSeekMath 原式是更贴近实现的 token 级 surrogate objective。

容易混淆的点

所以，后续很多博客或教程写成

$$A_i = \frac{r_i - \bar{r}}{\sigma_r}$$

并没有错，但它表达的是 **GRPO 的核心直觉**；真正对应原始论文实现的，是 token 级 ratio、token 级 surrogate，以及 outcome/process supervision 两种 $\hat{A}_{i,t}$ 的构造方式。

第 4 章　从 PPO 到 GRPO：一步一步推出来

4.1　第一步：从 PPO 的 critic 视角出发

在 PPO 里，一个常见的直觉写法是

$$\hat{A}_t \approx \hat{Q}_t - V_\psi(s_t), \tag{4.1}$$

其中 V_ψ 是 critic/value model。它回答的问题是：

“在当前状态下，不管你选什么动作，平均大概能拿到多大回报?”

于是：

- 若某动作最终结果比平均水平好，则 $\hat{A}_t > 0$；
- 若比平均水平差，则 $\hat{A}_t < 0$。

4.2　第二步：把 “平均水平” 从 learned value 改成同题组内比较

假设我们暂时只考虑 outcome supervision：同一个问题 q 下，采样得到 G 条回答

$$o_1, \ldots, o_G,$$

对应奖励

$$r_1, \ldots, r_G.$$

那么最自然的组平均就是

$$\mu_g = \frac{1}{G} \sum_{i=1}^{G} r_i. \tag{4.2}$$

如果仅从直觉上看，我们会想把它类比成 “同一问题下的平均水平”，于是得到

$$r_i - \mu_g. \tag{4.3}$$

它的符号有明确含义：

- $r_i - \mu_g > 0$：第 i 条回答高于组平均；
- $r_i - \mu_g < 0$：低于组平均；
- $r_i - \mu_g = 0$：与组平均持平。

再进一步除以组标准差 σ_g，就得到尺度统一的形式：

$$\tilde{r}_i = \frac{r_i - \mu_g}{\sigma_g}. \tag{4.4}$$

4.3 第三步：把序列级相对分数传给每个 token

LLM 的策略更新最终要落到 token 对数概率上，因此 outcome supervision 下最简单的处理是：

$$\hat{A}_{i,t} = \tilde{r}_i.$$

也就是说，第 i 条输出中每个 token 的更新方向都共享同一个“整条回答相对好坏”的信号。于是 token 级 surrogate 就变成：

$$\min\Big(r_{i,t}(\theta)\tilde{r}_i,\ \mathrm{clip}(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\tilde{r}_i\Big). \tag{4.5}$$

4.4 第四步：为什么还需要 KL

如果只有组相对奖励，那么模型会倾向于快速把“高奖答案”的概率抬上去，把“低奖答案”的概率压下去。若没有任何约束，模型可能在短时间内：

1. 急剧偏离初始语言分布；
2. 过拟合某些奖励漏洞；
3. 出现语言质量退化。

因此引入参考策略 π_{ref} 和 KL 罚项，限制模型更新过快：

$$-\beta\, \mathrm{D}_{\mathrm{KL}}[\pi_\theta \| \pi_{\mathrm{ref}}]. \tag{4.6}$$

这里负号表示：KL 越大，目标函数越小；因此最大化目标时会主动压制过大的偏移。

4.5 第五步：重要的严格性提醒

严格说法

很多入门解释会说：“GRPO 就是把 PPO 的 $V(s_t)$ 换成组平均 reward。”这个说法**足够直观，但不够严格**。
更准确的表述是：**DeepSeekMath 直接定义了一个基于组相对奖励的 surrogate objective**。这个 objective 的优势项由同题多样本组构造出来，而不是先证明“组平均一定是一个标准无偏 baseline”，再从 PPO 一步推导得到。
换句话说，PPO → GRPO 的关系更像“设计思想上的继承”和“目标函数结构上的相似”，而不是“把 PPO 的一处符号替换一下，理论就原封不动继承过来”。

4.6 第六步：为什么组标准化是合理的

组标准化

$$\tilde{r}_i = \frac{r_i - \mu_g}{\sigma_g}$$

有两层作用：

1. **去中心化：**只保留相对好坏，不保留组的绝对难度基线；
2. **统一尺度：**不同问题的 reward 范围不同，标准化后更新幅度更可比。

例如：

- 简单题：可能四条回答都得高分；
- 难题：可能四条回答都很差。

若只看原始 reward，简单题会在训练中“声量更大”；而组标准化会让模型更多关注同题内部的优劣排序。

容易混淆的点

不过，后续一些工作也指出：标准差归一化本身可能带来额外偏置或难度缩放问题，因此后来的实现并不总是保留原始 GRPO 的标准化细节 [6, 5]。这属于“原始算法”和“后续工程变体”之间的重要区别。

第 5 章　把公式真正算一遍：四个手算例题

5.1　例题一：组内标准化优势怎么算

手算例题

设某个 prompt 下，采样了 $G = 4$ 条回答，奖励分别是：

$$r_1 = 0.2, \qquad r_2 = 0.8, \qquad r_3 = 0.5, \qquad r_4 = -0.1.$$

目标：计算每条回答的组相对优势。

5.1.1　第一步：求组均值

$$\mu_g = \frac{0.2 + 0.8 + 0.5 - 0.1}{4} = \frac{1.4}{4} = 0.35.$$

5.1.2　第二步：求离均差

$$r_1 - \mu_g = 0.2 - 0.35 = -0.15,$$
$$r_2 - \mu_g = 0.8 - 0.35 = 0.45,$$
$$r_3 - \mu_g = 0.5 - 0.35 = 0.15,$$
$$r_4 - \mu_g = -0.1 - 0.35 = -0.45.$$

5.1.3　第三步：求方差与标准差

先平方：

$$(-0.15)^2 = 0.0225, \qquad 0.45^2 = 0.2025, \qquad 0.15^2 = 0.0225, \qquad (-0.45)^2 = 0.2025.$$

方差：

$$\sigma_g^2 = \frac{0.0225 + 0.2025 + 0.0225 + 0.2025}{4} = \frac{0.45}{4} = 0.1125.$$

标准差：

$$\sigma_g = \sqrt{0.1125} \approx 0.3354.$$

5.1.4 第四步：求标准化后的相对优势

$$A_1 = \frac{0.2 - 0.35}{0.3354} \approx -0.4472,$$
$$A_2 = \frac{0.8 - 0.35}{0.3354} \approx 1.3416,$$
$$A_3 = \frac{0.5 - 0.35}{0.3354} \approx 0.4472,$$
$$A_4 = \frac{-0.1 - 0.35}{0.3354} \approx -1.3416.$$

因此：

$$(A_1, A_2, A_3, A_4) \approx (-0.4472,\ 1.3416,\ 0.4472,\ -1.3416).$$

解释如下：

- 第 2 条回答最好，应该被最明显地增强；
- 第 4 条回答最差，应该被最明显地削弱；
- 第 1、3 条则分别是轻度负向与轻度正向。

若采用 outcome supervision，那么第 2 条回答中**所有 token** 都共享

$$\hat{A}_{2,t} = 1.3416.$$

5.2 例题二：一个 token 的 clipped surrogate 怎么算

现在拿上例中的第 2 条回答，设它有 3 个 token。我们只看其中 3 个位置的 ratio：

$$r_{2,1}(\theta) = 1.3, \qquad r_{2,2}(\theta) = 1.05, \qquad r_{2,3}(\theta) = 0.7.$$

并令

$$\hat{A}_{2,t} = 1.3416, \qquad \epsilon = 0.2.$$

于是 clip 的上下界是

$$1 - \epsilon = 0.8, \qquad 1 + \epsilon = 1.2.$$

5.2.1 第 1 个 token

原始项：

$$1.3 \times 1.3416 = 1.7441.$$

裁剪后 ratio：

$$\text{clip}(1.3, 0.8, 1.2) = 1.2.$$

裁剪项：

$$1.2 \times 1.3416 = 1.6099.$$

PPO/GRPO surrogate 取二者较小值：

$$\min(1.7441, 1.6099) = 1.6099.$$

5.2.2　第 2 个 token

原始项：

$$1.05 \times 1.3416 = 1.4087.$$

由于 1.05 在 $[0.8, 1.2]$ 内，裁剪后不变：

$$\text{clip}(1.05, 0.8, 1.2) = 1.05.$$

所以：

$$\min(1.4087, 1.4087) = 1.4087.$$

5.2.3　第 3 个 token

原始项：

$$0.7 \times 1.3416 = 0.9391.$$

裁剪后：

$$\text{clip}(0.7, 0.8, 1.2) = 0.8.$$

裁剪项：

$$0.8 \times 1.3416 = 1.0733.$$

因此：

$$\min(0.9391, 1.0733) = 0.9391.$$

5.2.4　汇总

三个 token 的 surrogate 分别是：

$$1.6099, \qquad 1.4087, \qquad 0.9391.$$

若简单平均，则该 3-token 片段的平均 surrogate 为

$$\frac{1.6099 + 1.4087 + 0.9391}{3} = \frac{3.9577}{3} \approx 1.3192.$$

直观理解

这个例子很有代表性：第 1 个 token 因为“涨得太多”而被 clip 截住；第 2 个 token 正常更新；第 3 个 token 虽然比值小于 1，但由于当前 advantage 为正，它的目标仍为正，只是推动力度较弱。

5.3 例题三：KL 惩罚项怎么算

假设某个 token 位置上，当前策略和参考策略对“实际被采样到的 token”的概率分别是

$$\pi_\theta = 0.30, \qquad \pi_{\text{ref}} = 0.24.$$

代入 DeepSeekMath 使用的形式：

$$x = \frac{\pi_{\text{ref}}}{\pi_\theta} = \frac{0.24}{0.30} = 0.8.$$

于是

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}[\pi_\theta \| \pi_{\text{ref}}] &\approx x - \log x - 1 && (5.1)\\ &= 0.8 - \log(0.8) - 1 && (5.2)\\ &\approx 0.8 - (-0.2231) - 1 && (5.3)\\ &= 0.0231. && (5.4) \end{aligned}$$

若 $\beta = 0.04$，则 KL 罚项贡献为

$$\beta \cdot \mathrm{D}_{\mathrm{KL}} = 0.04 \times 0.0231 \approx 0.000924.$$

这说明在这个具体位置上，KL 约束是一个比较小、但持续存在的“别偏太远”拉力。

5.4 例题四：process supervision 的 token 级优势

设一个 group 中所有步骤奖励（汇总到一起）是：

$$\mathbf{R} = [0.2,\ 0.9,\ -0.1,\ 0.4].$$

先算均值：

$$\mu_R = \frac{0.2 + 0.9 - 0.1 + 0.4}{4} = \frac{1.4}{4} = 0.35.$$

离均差为：

$$-0.15, \quad 0.55, \quad -0.45, \quad 0.05.$$

平方后：

$$0.0225, \quad 0.3025, \quad 0.2025, \quad 0.0025.$$

方差：

$$\frac{0.0225 + 0.3025 + 0.2025 + 0.0025}{4} = \frac{0.53}{4} = 0.1325.$$

标准差：

$$\mathrm{Std}(\mathbf{R}) = \sqrt{0.1325} \approx 0.3640.$$

于是标准化步骤奖励是：

$$\begin{aligned}
\frac{0.2 - 0.35}{0.3640} &\approx -0.4121, \\
\frac{0.9 - 0.35}{0.3640} &\approx 1.5110, \\
\frac{-0.1 - 0.35}{0.3640} &\approx -1.2362, \\
\frac{0.4 - 0.35}{0.3640} &\approx 0.1374.
\end{aligned}$$

现在设输出 o_1 有两个步骤终点，分别在 token 位置 5 和 11，对应的标准化步骤奖励是

$$-0.4121, \qquad 1.5110.$$

根据 process supervision 规则：

- 对于 token $t = 3$，未来还会遇到步骤 5 和 11，因此

$$\hat{A}_{1,3} = -0.4121 + 1.5110 = 1.0989.$$

- 对于 token $t = 7$，步骤 5 已经过了，只剩步骤 11，因此

$$\hat{A}_{1,7} = 1.5110.$$

- 对于 token $t = 12$，所有步骤都结束了，因此

$$\hat{A}_{1,12} = 0.$$

直观理解

这说明 process supervision 不是“整条回答所有 token 共用同一个分数”，而是：**离后续好步骤更近的 token，会保留更多正向信用；离后续坏步骤更近的 token，也会承受更多负向信用。**

5.5 一个特别简单但很有启发的边界情形：$G = 2$

若 group size $G = 2$，且奖励为

$$r_1 = 1, \qquad r_2 = 0,$$

那么组均值

$$\mu_g = \frac{1+0}{2} = 0.5,$$

标准差

$$\sigma_g = \sqrt{\frac{(1-0.5)^2 + (0-0.5)^2}{2}} = \sqrt{0.25} = 0.5.$$

所以

$$A_1 = \frac{1-0.5}{0.5} = 1, \qquad A_2 = \frac{0-0.5}{0.5} = -1.$$

这几乎就是“二选一偏好比较”的最小版本。也因此，很多人会把 GRPO 理解成一种**在线、组相对、带 clipping 的偏好优化**。

第 6 章　GRPO 一轮训练到底发生了什么

6.1　原始伪代码图

Algorithm 1 Iterative Group Relative Policy Optimization

Input initial policy model $\pi_{\theta_{\text{init}}}$; reward models r_φ; task prompts $\mathcal{D}$; hyperparameters ε, β, μ

policy model $\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$
for iteration = 1, . . . , I **do**
  reference model $\pi_{ref} \leftarrow \pi_\theta$
  for step = 1, . . . , M **do**
    Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
    Update the old policy model $\pi_{\theta_{old}} \leftarrow \pi_\theta$
    Sample G outputs $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(\cdot \mid q)$ for each question $q \in \mathcal{D}_b$
    Compute rewards $\{r_i\}_{i=1}^{G}$ for each sampled output o_i by running r_φ
    Compute $\hat{A}_{i,t}$ for the t-th token of o_i through group relative advantage estimation.
    for GRPO iteration = 1, . . . , μ **do**
      Update the policy model π_θ by maximizing the GRPO objective (Equation 21)
  Update r_φ through continuous training using a replay mechanism.

Output π_θ

And different from the KL penalty term used in (2), we estimate the KL divergence with the following unbiased estimator (Schulman, 2020):

$$\mathbb{D}_{KL}\left[\pi_\theta || \pi_{ref}\right] = \frac{\pi_{ref}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} - \log \frac{\pi_{ref}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} - 1, \tag{4}$$

图 6.1: DeepSeekMath 中的 Iterative GRPO 伪代码与 KL 估计公式。图像裁剪自 Shao 等人的论文第 14 页 [1]。

6.2　把伪代码翻译成人话

原始 Algorithm 1 可以翻译成下面的训练循环：

1. 初始化策略模型 π_θ；
2. 进入外层迭代；
3. 冻结/拷贝一个参考模型 π_{ref}；

4. 抽一个 prompt batch；
5. 令当前策略拷贝为旧策略 $\pi_{\theta_{\text{old}}}$；
6. 对 batch 中每个问题，从旧策略采样 G 条回答；
7. 对这些回答计算 reward；
8. 根据组内 reward 计算每条回答的 advantage；
9. 用 GRPO 目标函数更新策略；
10. （如果有奖励模型）可继续用 replay 数据更新奖励模型。

6.3 GRPO 的训练对象不是“答案文本”，而是策略分布

这是很多初学者最容易忽略的一点。GRPO 并不是把高分答案“抄下来背会”，而是在做分布更新：

$$\pi_{\theta_{\text{old}}} \longrightarrow \pi_{\theta}.$$

更准确地说，它在做的是：

- 提高产生高优势回答的 token 概率；
- 降低产生低优势回答的 token 概率；
- 同时保持新分布不要偏离参考分布太远。

6.4 为什么 reasoning 任务特别适合 GRPO

DeepSeek-R1 的方法部分说明，它们在 reasoning-oriented 数据上大量使用 rule-based reward，对数学、代码和逻辑任务提供准确反馈 [4]。这类任务特别适合 GRPO，原因在于：

1. 同题多样本天然可比较；
2. 奖励往往可验证（正确/错误、测试通过/失败、格式满足/不满足）；
3. 相对奖励比绝对打分更稳定；
4. reasoning 过程本来就有“尝试–验证–修正”的搜索特征。

6.5　Outcome supervision 与 rule-based reward 的组合

在很多数学/代码任务里，一个直接可用的奖励设计是：

$$r_i = r_i^{\text{correct}} + r_i^{\text{format}} + r_i^{\text{aux}}. \tag{6.1}$$

例如：

- r_i^{correct}：答案是否正确；
- r_i^{format}：是否满足指定输出格式；
- r_i^{aux}：是否通过单元测试、是否调用了工具、是否遵守模板等。

然后把这些原始 reward 再放进组内标准化。DeepSeek-R1 明确说明：对 DeepSeek-R1-Zero，主要使用 rule-based rewards 来给数学、代码和逻辑推理数据提供准确反馈；对 DeepSeek-R1，则在 general data 上还加入了 model-based rewards[4]。

第 7 章　GRPO 到底解决了什么问题

7.1　问题一：去掉 critic/value model

DeepSeekMath 提出 GRPO 的直接收益，就是**不再需要额外 value model**，从而显著减少训练资源 [1]。

如果把传统 PPO-RLHF 粗略写成：

$$\text{Policy} + \text{Reference} + \text{Reward} + \text{Value/Critic},$$

那么 GRPO 更像：

$$\text{Policy} + \text{Reference} + \text{Reward/Verifier}.$$

在长 CoT 场景里，这个差别很有工程价值。

7.2　问题二：更自然地利用同题比较信号

奖励模型、偏好比较数据、verifier 打分，这些对象很多时候本来就是“比较式”的。DeepSeekMath 也指出，GRPO 的 group-relative 计算与奖励模型的比较性数据分布是相契合的 [1]。

换句话说，GRPO 很适合回答这样的问题：

同一个 prompt 下，这几条回答里谁更应该被强化？

而不一定非要回答：

这条回答在全局意义上的绝对价值是多少？

7.3　问题三：把 reasoning 当作可探索的策略行为

DeepSeek-R1 报告了在纯 RL 或以 RL 为核心的训练中，模型会逐渐出现更长的思考时间、自我反思与策略调整等行为 [4]。这表明 reasoning 不只是“背答案”，更像是一种在奖励驱动下逐步形成的搜索策略。

GRPO 恰好适合这种训练范式，因为它允许模型：

1. 对同一题尝试多种轨迹；

2. 通过 verifier/reward 找到相对更优轨迹；
3. 逐步把概率质量向优轨迹迁移。

7.4　问题四：在 PPO 风格稳定性与 critic-free 之间折中

GRPO 不是简单地退回到原始 REINFORCE。它仍保留了两类重要的稳定器：

1. clipping；
2. KL regularization.

因此你可以把 GRPO 看成：

> “保留 PPO 的稳定更新骨架，但把 advantage 的来源从 learned value 改成 group-relative reward。”

第 8 章　GRPO 的局限、边界条件与后续变体

8.1　局限一：如果组内奖励没有差异，就几乎没有训练信号

若某个问题下所有回答奖励都一样：

$$r_1 = r_2 = \cdots = r_G,$$

那么有

$$\mu_g = r_i, \qquad \sigma_g = 0.$$

这意味着标准化会失效。在工程实现中，通常会做下面某种处理：

- 给分母加一个很小的 ε；
- 或者直接把该组 advantage 置零；
- 或者跳过这个 group。

这说明：**GRPO 依赖可区分的奖励**。

8.2　局限二：group size 太小，估计会不稳定

如果 G 很小，那么组均值与组标准差本身就很 noisy。典型后果包括：

1. 某些回答只是偶然高分，却被过度强化；
2. 组统计量波动太大，导致 advantage 尺度不稳；
3. 难题、易题之间的更新节奏被 sampling noise 放大。

所以，group size 是一个非常重要的超参数。

8.3　局限三：原始 GRPO 的长度归一方式会带来争议

原始 DeepSeekMath 目标中有一项

$$\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}(\cdots),$$

也就是先按样本内部 token 平均，再对样本平均。后续工作指出，这种 sample-level loss reduction 在 long-CoT 场景下可能带来长度相关问题。

COLM 2025 的《Understanding R1-Zero-Like Training: A Critical Perspective》指出，GRPO 存在一种优化偏置，会在训练中人为推长回答长度，尤其是错误回答的长度，并提出 Dr. GRPO 通过移除长度项与标准差项来修正这一偏置 [6]。

DAPO 进一步指出，原始 GRPO 的 sample-level loss 在长 CoT 场景中会让长回答中的单个 token 对整体 loss 的贡献过低，进而导致不健康的熵和回答长度增长，因此提出 token-level policy gradient loss 作为改进 [7]。

严格说法

这并不意味着“原始 GRPO 错了”，而是意味着：**原始 GRPO 是一个非常重要的起点，但在更大规模、更长 CoT、更多工程约束下，社区已经发展出一批修正版和工程增强版**。

8.4　局限四：std 标准化也不是永远无争议

原始 GRPO 使用

$$\frac{r_i-\mu_g}{\sigma_g}.$$

后续一些研究和实现认为，标准差归一化会额外改变不同问题之间的更新尺度，从而引入 question-level difficulty bias 或其它不希望的缩放效应 [6, 5]。

Hugging Face 的 TRL 官方文档也明确提示：相较于原始论文，现代实现往往允许关闭或修改 std-based scaling，并且默认损失形式、KL 配置等也可能与原论文不同 [5]。

8.5　局限五：GRPO 仍然是在线 RL，数据成本并没有消失

GRPO 去掉的是 critic，而不是在线采样本身。它依然通常需要：

1. 用当前或旧策略不断生成新样本；
2. 对样本做奖励计算；
3. 多轮迭代更新。

因此，与纯离线的 DPO 类方法相比，GRPO 依然具有明显的在线 RL 成本。

第 9 章　与 PPO、DPO、RFT 的关系

9.1　一张对比表

方法	核心优化对象	是否需要在线采样	是否依赖 critic/value	更典型的使用场景
PPO	概率比值 + clipped surrogate + critic advantage	常需要	通常需要	通用 RLHF、经典 actor–critic 场景
GRPO	组相对奖励 + PPO 风格 clipping + （可选）KL	常需要	不需要	数学、代码、逻辑等可验证 reasoning
DPO	偏好对 chosen/rejected 的离线对比目标	不一定	不需要	离线偏好对齐、偏好数据充足时
RFT	先筛高质量结果，再监督学习模仿	不一定	不需要	有明确筛选规则、想做高质量蒸馏时

9.2　一个有用的统一视角

可以把几种方法放在“训练信号从哪里来”的坐标系里理解：

1. PPO：critic 估计“平均水平”，reward 决定方向；
2. GRPO：同题组内比较给出相对方向；
3. DPO：离线偏好对 (chosen, rejected) 直接规定方向；
4. RFT：先筛再学，用监督学习逼近高质量输出分布。

9.3　GRPO 最适合什么任务

最适合 GRPO 的任务通常有三个特征：

1. 能验证：有答案、有测试、有格式约束；

2. 能比较：同题多样本之间好坏明显；

3. 允许在线探索：模型多试几次是有价值的。

因此数学、代码、逻辑推理、结构化输出，是 GRPO 特别舒适的场景。

相反，如果任务是纯审美、纯开放式写作、极其主观的长期偏好，GRPO 当然不是不能用，但 reward 设计会难很多。

第 10 章　把全书压缩成一页：GRPO 的最小理解骨架

如果你读完整本教材后，只想记住最重要的内容，那么请记住下面这一页。

一、语言模型是策略

$$\pi_\theta(o|q) = \prod_{t=1}^{|o|} \pi_\theta(o_t|q, o_{<t}).$$

二、最基本 RL 目标是最大化期望奖励

$$J(\theta) = \mathbb{E}_{q,o\sim\pi_\theta}[R(q,o)].$$

三、PPO 用 clipped ratio 稳定更新

$$\min\Big(r_t(\theta)\hat{A}_t,\ \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big).$$

四、GRPO 的关键变化：用组相对优势取代 critic

对同一个问题，采样

$$o_1, \ldots, o_G,$$

打分得到

$$r_1, \ldots, r_G.$$

计算组均值与组标准差：

$$\mu_g = \frac{1}{G}\sum_i r_i, \qquad \sigma_g = \sqrt{\frac{1}{G}\sum_i (r_i - \mu_g)^2}.$$

再得到标准化优势：

$$A_i = \frac{r_i - \mu_g}{\sigma_g}.$$

在 outcome supervision 下，

$$\hat{A}_{i,t} = A_i.$$

五、原始 GRPO 仍然保留 PPO 风格的稳定器

$$J_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\Big(\min(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{i,t}) - \beta\, \mathrm{D}_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}]\Big)\right].$$

六、它解决的问题

1. 去掉 critic，降低 LLM-RL 的资源成本；
2. 更自然地利用同题多样本比较信号；
3. 对数学、代码、逻辑等可验证推理任务特别有效；
4. 在 critic-free 的同时保留 PPO 风格的稳定更新。

七、它的边界

1. 奖励必须有区分度；
2. group size 太小会不稳定；
3. 长回答与标准化细节会带来偏置问题；
4. 实际工程中常用的是“GRPO 家族变体”，未必与原论文完全一致。

结语

如果你想把 GRPO 彻底吃透，建议按下面顺序复习：

1. 先独立推一遍 REINFORCE 和 baseline；
2. 再手推 PPO 的 clipped surrogate；
3. 然后把 DeepSeekMath 的 Figure 4 和 Algorithm 1 对着本教材再读一遍；
4. 最后自己手算一个 4-sample group 的 GRPO 更新。

当你能不看书直接写出下面这 4 句时，说明你已经真正抓到 GRPO 的骨架了：

(a) LLM 是一个自回归策略；

(b) PPO 用 ratio + clipping 稳定策略更新；

(c) GRPO 通过同题多样本奖励构造组相对 advantage；

(d) GRPO 的本质是 critic-free 的、组比较式的 PPO 风格 RL。

参考文献

参考文献

[1] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300, 2024. https://arxiv.org/abs/2402.03300.

[2] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov. Proximal Policy Optimization Algorithms. arXiv:1707.06347, 2017. https://arxiv.org/abs/1707.06347.

[3] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, et al. Training Language Models to Follow Instructions with Human Feedback. arXiv:2203.02155, 2022. https://arxiv.org/abs/2203.02155.

[4] Daya Guo et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948, 2025. https://arxiv.org/abs/2501.12948.

[5] Hugging Face. TRL Documentation: GRPO Trainer. accessed 2026-05-10. https://huggingface.co/docs/trl/grpo_trainer.

[6] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, Min Lin. Understanding R1-Zero-Like Training: A Critical Perspective. COLM 2025 / arXiv:2503.20783, 2025. https://arxiv.org/abs/2503.20783.

[7] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. arXiv:2503.14476, 2025. https://arxiv.org/abs/2503.14476.