大语言模型后训练与强化学习

对齐理论 · RLHF/RLVR 算法推导 · 推理强化 · 系统工程

从 SFT、奖励模型、PPO/DPO/GRPO 到 RLVR、测试时扩展与大规模训练框架

一份自包含、可复制、面向研究与工程双重读者的详细报告

覆盖范围： 后训练流水线（SFT / RM / RL / 测试时计算）• 数学基础（策略梯度、重要性采样、KL 约束、GAE）• 奖励建模（Bradley–Terry、ORM/PRM、奖励攻破）• 核心算法（PPO、DPO 家族、GRPO、DAPO、RLOO、RLVR）• 推理强化（DeepSeek-R1、长思维链、过程奖励）• 测试时计算扩展 • 工程框架（OpenRLHF / veRL / TRL、vLLM rollout、异步架构）• 完整 PyTorch 参考实现与调参清单。

2026 年 11 月版

附 15 幅矢量原理图（SVG 源文件随附）与完整可运行代码清单

摘要

预训练赋予大语言模型（LLM）**能力**，后训练决定它**如何使用这些能力**。从预训练结束到最终部署之间的一切——监督微调（SFT）、奖励建模、基于人类偏好或可验证信号的强化学习（RLHF/RLVR）、以及围绕它们的测试时计算扩展——构成了 2022 年以来 LLM 技术演进最快、也最缺乏系统性资料的领域。InstructGPT 证明了“少量偏好数据 + RL”可以驯服千亿模型；DeepSeek-R1 进一步证明：在可验证奖励上直接做 RL，可以让模型自发涌现反思、验证、长程推理等行为，把后训练从“对齐工具”推向“能力引擎”。

本报告以**数学推导**（策略梯度定理及其方差分析、RLHF 目标的 KL 约束结构与闭式解、Bradley–Terry 奖励模型的极大似然推导、GRPO 组相对优势的无偏性讨论）、**信息论与统计视角**（重要性采样误差累积、off-policy 程度、熵坍缩与探索丧失、奖励攻破的过拟合机制）、**工程实践代码**（PPO/GRPO 的完整 PyTorch 参考实现、vLLM rollout 集成、分布式架构与权重同步）三条主线，系统讲解 2022–2026 年 LLM 后训练与强化学习的算法与系统：

- **算法层**：PPO（含 token 级目标与两种 KL 实现）、DPO 及其变体（IPO/KTO/ORPO/SimPO）、GRPO 与 Dr. GRPO/DAPO 的长度偏置修正、RLOO/ReMax、RLVR 与过程奖励（PRIME）；
- **推理层**：长思维链的 RL 涌现、aha moment、课程与难度过滤、预算控制（budget forcing、L1）、测试时计算的统一视角；
- **系统层**：四种训练范式（colocate / 分离 / 混合 / 异步）、vLLM/SGLang rollout 引擎、权重同步、off-policy 级联修正、rollout 长尾与资源配比、训练健康度监控。

阅读本报告后，读者应能：完整推导 RLHF/PPO/DPO/GRPO 的目标函数并理解每个设计选择的动因；看懂并修改主流开源框架（TRL、OpenRLHF、veRL、llama-factory）的关键代码路径；为一个新模型/新任务设计后训练流水线（数据构造、算法选型、超参数、监控指标）；并能诊断 reward hacking、熵坍缩、off-policy 衰减等典型训练病害。

如何使用本报告（阅读路线）

- **理论线**：第 2 章（RL 与信息论基础）→ 第 4.1 节（Bradley–Terry）→ 第 5 章（PPO 完整推导）→ 第 6 章（DPO 闭式解）→ 第 7 章（GRPO 与方差）。
- **算法线**：第 3–7 章按时间轴阅读；第 8 章（RLVR 与推理 RL）是当前前沿的主体；第 9–10 章为测试时与前沿专题。
- **工程线**：第 11 章（框架与架构）→ 第 12 章（完整代码）→ 第 13 章（性能优化）→ 第 14 章（评测与诊断）。
- **选型线**：图 1.1（流水线全景）→ 图 15（算法选型决策树）→ 第 14 章病害表 → 第 12.5 节超参数清单。
- 每章末尾的“要点”框汇总结论；推导框给出完整计算过程；算法框给出可实现伪代码；附录给出符号表与单文件 GRPO 训练循环参考实现。

目录

摘要 i

第一章 引言：后训练决定模型“是谁” 1
- 1.1 同一个底座，不同的产品 . . . 1
- 1.2 为什么预训练之后还需要后训练 . . . 1
- 1.3 2022–2026 简史：一条从“驯服”到“赋能”的谱系 . . . 2
- 1.4 本报告的三条主线 . . . 3
- 1.5 符号约定 . . . 3

第二章 数学基础：策略梯度、方差控制与 KL 约束 5
- 2.1 LLM 作为逐 token 的马尔可夫决策过程 . . . 5
- 2.2 策略梯度：从似然比到 token 级梯度 . . . 6
- 2.3 基线与优势：方差从哪里来 . . . 7
- 2.4 Critic 与广义优势估计（GAE） . . . 8
- 2.5 重要性采样与 off-policy 修正 . . . 8
- 2.6 KL 散度：约束、几何与两种不对称 . . . 9
- 2.7 LLM 场景的五个特化事实 . . . 9

第三章 监督微调：格式、任务与行为克隆 11
- 3.1 目标函数与 teacher forcing . . . 11
- 3.2 数据工程：示范从哪里来 . . . 11
- 3.3 打包（packing）与位置编码的交互 . . . 12
- 3.4 SFT 作为行为克隆：它的理论边界 . . . 12
- 3.5 参数高效微调（LoRA）在后训练中的角色 . . . 12

第四章 奖励建模：从人类偏好的标量化 14
- 4.1 Bradley–Terry 模型：偏好比较的极大似然 . . . 14
- 4.2 outcome 与 process：两种监督粒度 . . . 15
- 4.3 Reward hacking：代理目标必然被攻破 . . . 16

第五章 RLHF-PPO：完整剖析 17
- 5.1 RLHF 的目标：KL 约束下的奖励最大化 . . . 17
- 5.2 PPO-clip 目标：逐项剖析 . . . 17
- 5.3 优势与价值：critic 怎么训 . . . 18
- 5.4 一个完整 PPO 步的解剖 . . . 19
- 5.5 已披露的真实配置 . . . 19

第六章 直接偏好优化：闭式解与分类损失 21
- 6.1 从 RLHF 目标到闭式最优策略 . . . 21
- 6.2 梯度解剖与隐式奖励 . . . 22
- 6.3 为什么 DPO 可能出问题：理论与坑 . . . 22
- 6.4 变体族谱：IPO、cDPO、SimPO、KTO、ORPO . . . 23
- 6.5 迭代式偏好优化：把 DPO 变成在线算法 . . . 23

第七章 GRPO 与组相对方法：删除 critic **25**
7.1 从 PPO 到 GRPO：目标函数 . . . 25
7.2 组基线的统计性质 . . . 26
7.3 长度偏置的解剖与 Dr. GRPO . . . 26
7.4 DAPO：四项工程修正 . . . 26
7.5 实践配置基线 . . . 27

第八章 RLVR 与推理强化：R1 的完整故事 **28**
8.1 奖励设计：RLVR 的“法律条文” . . . 28
8.2 R1-Zero：纯 RL 的涌现现象 . . . 29
8.3 R1 的完整流水线：四阶段 . . . 30
8.4 课程、难度与健康监控 . . . 30
8.5 熵：RLVR 的关键存量 . . . 30
8.6 过程奖励的在线化：PRIME . . . 31

第九章 测试时计算扩展：用推理 FLOPs 换精度 **32**
9.1 统一视角：第二条扩展轴 . . . 32
9.2 并行采样：Best-of-N 与自洽性 . . . 33
9.3 序列搜索：过程奖励引导的解码 . . . 33
9.4 预算控制：把长度变成旋钮 . . . 34
9.5 pass@k 视角：RL 在压缩什么 . . . 34
9.6 训练与测试的对偶清单 . . . 34

第十章 前沿专题：自博弈、智能体 RL、多模态与蒸馏 **36**
10.1 自博弈与迭代放大 . . . 36
10.2 智能体 RL：多轮、工具与环境反馈 . . . 36
10.3 多模态后训练 . . . 37
10.4 蒸馏与后训练的交互 . . . 37
10.5 配方组合空间 . . . 38

第十一章 训练框架与系统架构 **39**
11.1 为什么 RLHF 系统难 . . . 39
11.2 四种范式 . . . 39
11.3 Rollout 引擎内幕 . . . 40
11.4 权重同步的三条路径 . . . 41
11.5 框架选型 . . . 41

第十二章 工程实践：参考实现、配置与排错 **43**
12.1 GRPO 的单文件参考实现 . . . 43
12.2 接入 vLLM：生产版 rollout . . . 44
12.3 生产超参数清单 . . . 45
12.4 排错表：症状 → 病因 → 处方 . . . 45
12.5 监控面板：六个数看住训练 . . . 46

第十三章 性能优化：算力配比与算法–系统协同 **47**
13.1 一步的墙钟模型 47
13.2 资源配比的粗算 48
13.3 Off-policy 级联与样本复用 48
13.4 数值稳定性清单 48
13.5 算法–系统协同：三组已验证的等价 49

第十四章 评测与训练诊断 **50**
14.1 评测集的分层设计 50
14.2 训练健康度：五条曲线怎么读 50
14.3 Reward hacking 排查程序 51
14.4 诊断速查：指标组合 → 结论 51
14.5 消融的组织方式 51

第十五章 开放问题与总结 **53**
15.1 选型决策树：五个问题 53
15.2 开放问题：六个没有共识的方向 54
15.3 学习路线：从本报告到动手 54
15.4 结语 54

附录 A 附录：符号表与算法速查 **56**
A. 符号总表 56
B. 算法速查表 56
C. 完整训练循环 57

参考文献 **59**

第一章 引言：后训练决定模型“是谁”

1.1 同一个底座，不同的产品

2023 年之后，大语言模型领域出现了一个被反复验证的事实：**预训练底座（base model）决定模型“知道什么”，后训练（post-training）决定模型“是谁”**。同一个 Llama-3-70B 基座，经过不同的后训练，可以变成一个礼貌的通用助手、一个拒绝越狱的安全模型，也可以变成一个在数学竞赛上碾压对手、写出数千 token 推理链的思维机器——三者的预训练权重完全相同。

所谓后训练，指预训练结束（next-token 预测已收敛）之后、部署之前的一切训练步骤，通常包括：

- **监督微调（SFT）**：用少量高质量的“指令 → 回答”示范数据，以交叉熵把模型行为从“续写”切换为“应答”；
- **奖励建模（RM）**：用人类偏好对（哪个回答更好）训练一个打分器，把“人类觉得好”压缩为一个标量；
- **强化学习（RLHF / RLVR）**：让模型自己生成回答，用 RM 或可验证的规则打分，用策略梯度把行为推向高分方向，同时用 KL 散度把它拴在参考模型附近；
- **测试时计算扩展（test-time scaling）**：训练模型“愿意且善于”在推理时花更多计算（长思维链、自我验证、多次采样）换取精度。

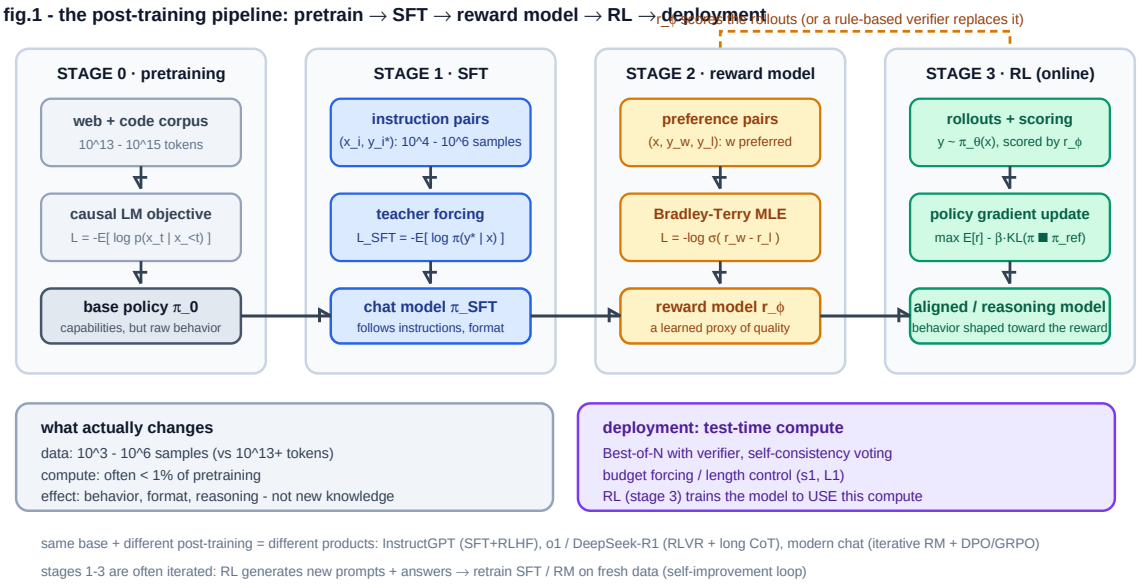


图 1.1 后训练全景。注意尺度的悬殊：预训练消耗 10^{13}–10^{15} token，后训练各阶段通常只有 10^{3}–10^{6} 条样本——但行为层面的差异几乎全部来自后者。

1.2 为什么预训练之后还需要后训练

预训练目标（预测下一个 token）与部署目标（有用、诚实、无害；或答对数学题）之间存在 **目标错位**。这个错位不是工程细节，而是结构性的：

1. **能力的“休眠”问题。**预训练语料中包含大量证明、代码、纠错文本，但自回归采样几乎从不自发组织成“先列计划、再验证、再修改”的长程结构。RL 的作用不是注入新知识，而是*重新加权已有的行为模式*：把采样分布的质量集中到语料中罕见但高回报的行为上。DeepSeek-R1-Zero 证明了这一点——不添加任何新数据，纯粹用“答对给 1 分”的奖励做 RL，模型自发出现了反思与回溯（第 八 章）。

2. **对齐问题。**续写型模型会代人写信、输出有害内容，因为语料里存在这些模式。RLHF（InstructGPT，2022）首次证明：10^4 量级的人类偏好比较数据 + KL 约束下的 RL，能把千亿模型的行为调整到与人类意图一致，且*几乎不损失基准能力*（KL 约束保证了这一点，见第 五 章）。

3. **测试时计算的“汇率”问题。**给定推理预算（生成多少 token、采样多少次），模型把 token 换成精度的效率差异巨大。这个汇率可以通过 RL 直接优化（第 九 章）——它是 o1/R1 一类模型的核心。

1.3 2022–2026 简史：一条从“驯服”到“赋能”的谱系

年份	工作	贡献（本报告对应章节）
2022	InstructGPT	SFT + RM + PPO 三段式成为标准（第 三–五 章）
2022–23	Anthropic Constitutional	用 AI 反馈替代人类偏好对，RM 规模化（第 四 章）
2023	DPO	RLHF 目标闭式解 $\Rightarrow$ 偏好对上的直接分类损失（第 六 章）
2023	Math-Shepherd	蒙特卡洛自动标注过程奖励（第 四 章）
2024	GRPO（DeepSeekMath）	组相对优势替代 critic；RLVR 门槛大幅下降（第 七 章）
2024	o1（OpenAI）	测试时扩展 + 推理 RL 的产品化验证（第 九 章）
2025	DeepSeek-R1 / Kimi k1.5	纯 RLVR 涌现长思维链；完整技术报告开源（第 八 章）
2025	DAPO / Dr.GRPO	系统修正 GRPO 的长度偏置与熵坍缩（第 七 章）
2025–26	veRL / OpenRLHF / AReaL	训练系统从同构走向异步（第 十一–十三 章）

表 1.1 后训练与 RL 的关键节点。

这条谱线上有两个值得强调的“范式切换”：

两个范式切换

（a）在线 RL → 离线直接优化（2023，又部分回摆）。 DPO 证明了 RLHF 目标可以绕开采样与 RM，直接在偏好对上以闭式梯度优化，成本骤降一个数量级——但随后实践发现它对数据分布、迭代方式极其敏感（第 六 章），线上 RL 在 2024 年后随 GRPO 与推理任务强势回归。

（b）RM 反馈 → 可验证奖励（2024–2025）。 当任务本身可判定（数学答案可对错、代码可测试），用规则验证器替代学习型 RM，彻底消除 reward hacking 的拟合面；这让 RL 第一次可以“放开手”跑数万步——因为奖励永远不会说谎。

1.4 本报告的三条主线

数学主线。 每个算法从其目标函数出发完整推导：策略梯度定理与其方差结构（第 二 章）、Bradley–Terry 偏好模型的极大似然（第 四 章）、KL 约束 RLHF 目标的拉格朗日与闭式解（第 五, 六 章）、GRPO 组相对优势的偏差–方差分析（第 七 章）。所有推导都做到“从零可重复”——需要的只有微积分与概率论。

信息与统计主线。 RLHF 的本质是*用极少的评分样本估计并改变一个极高维分布*。我们始终关注三个统计量：**梯度噪声**（优势估计的方差）、**分布距离**（当前策略与采样策略的 KL，即 off-policy 程度）、**熵**（探索能力的存量）。它们分别对应工程上的三大病害：学不动、学偏、学死（第 十三, 十四 章）。

工程主线。 算法只在纸上成立，落地时的一切困难都在系统里：rollout 引擎吞吐（生成占一次迭代 50%–90% 的墙钟）、四个模型的显存编排、训练与推理引擎的权重同步、异步架构下的 off-policy 修正（第 十一–十三 章）。我们给出可直接运行的参考实现与踩坑清单（第 十二 章与附录）。

1.5 符号约定

全文统一使用下列记号（附录 A 有完整符号表）：

记号	含义	说明
x，$y = (y_1, \dots, y_T)$	提示、生成回答	回答长度 T 由采样决定，回合终止于 EOS
$\pi_\theta(y_t \mid x, y_{<t})$	策略	即 LLM 本身；θ 为其全部参数
π_{ref}	参考策略	冻结（通常是 SFT 后的模型），KL 锚点
$r_\phi(x, y)$	奖励模型	标量；ϕ 为奖励模型参数
r，r_t	奖励	序列级或 token 级（上下文区分）
$\hat{A}_t$	优势估计	$A(s,a) = Q(s,a) - V(s)$ 的估计
ρ_t	重要性比率	$\pi_\theta(y_t \mid s_t) / \pi_{\text{old}}(y_t \mid s_t)$

β	KL 系数	RLHF 的“温度”：允许偏离 π_{ref} 的程度
G	组大小	GRPO 每个提示采样的回答数（勿与回报 G_t 混淆）
ϵ	clip 半径	PPO/GRPO 裁剪参数

表 1.2 主要符号。

第 1 章要点

- 后训练改变的是**行为分布**，不是知识总量；数据量比预训练小 6–9 个数量级。
- 三大手段各有分工：SFT 教“格式与任务”，RM+RL 教“好坏判断”，RLVR 教“利用测试时计算的策略”。
- 算法演进的两条主线：降低在线 RL 的工程门槛（PPO→GRPO→DAPO），以及把反馈源从“学出来的代理”换成“不可攻破的规则”（RM→verifier）。

第二章 数学基础：策略梯度、方差控制与 KL 约束

本章把后续所有算法（PPO/DPO/GRPO/RLVR）赖以成立的三块数学地基打好：**似然比策略梯度**（梯度的形态）、**基线与优势**（方差从哪里来、如何压下去）、**KL 散度约束**（如何防止一步走太远）。我们建议即使熟悉 RL 的读者也快速过一遍第 2.1 节与第 2.6 节——LLM 场景的特殊结构（终端奖励、$\gamma = 1$、token 级 KL）使许多教科书结论以更强的形式成立。

2.1 LLM 作为逐 token 的马尔可夫决策过程

一个回合（episode）由提示与回答组成：给定提示 x，模型逐 token 生成 $y_1, y_2, \ldots, y_T$，终止于 EOS（或达到长度上限）。把它映射为 MDP：

MDP 元素	LLM 对应物
状态 s_t	上下文 $s_t = (x, y_{<t})$：已生成的全部 token（在实现中即 KV cache）
动作 $a_t = y_t$	下一个 token，从词表 $\mathcal{V}$（$\lvert\mathcal{V}\rvert \sim 10^5$）中选出
转移 $P(s_{t+1} \mid s_t, a_t)$	**确定性**：$s_{t+1} = s_t \cup \{a_t\}$——环境就是“追加”
策略 $\pi_\theta(a_t \mid s_t)$	LLM 的 softmax 分布 $\pi_\theta(y_t \mid x, y_{<t})$
奖励 $r(s_t, a_t)$	ORM：仅在 $t = T$ 给一次序列级奖励；PRM：逐步给
折扣 γ	通常取 1（回合有限、长度 $\leq$ 数千，无需折扣）

表 2.1　LLM 的 token 级 MDP。

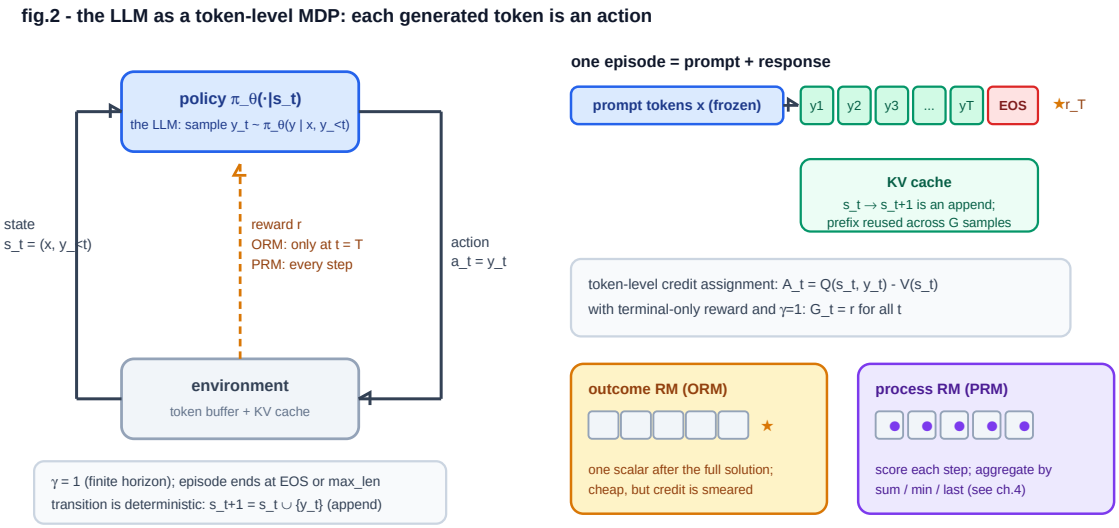


图 2.1　LLM 生成过程即 token 级 MDP：每个 token 是一次动作，状态是不断增长的上下文，奖励（ORM 情形）只在回合结束时结算。

这个形式化有三个直接推论，后文会反复用到：

1. **转移是确定性的且已知。**因此策略梯度完全不需要环境模型——这就是“log-derivative 技巧”在 LLM 上格外干净的原因。
2. **序列级奖励 + $\gamma = 1 \Rightarrow$ 所有位置的回报相同：** $G_1 = G_2 = \cdots = G_T = r$。在没有 critic 的情况下，整条回答共享同一个学习信号（GRPO 正是利用这一点，第 七 章）。
3. **回合长度是策略自身的函数。**T 由 EOS 的采样决定，因此“生成长回答”本身是被优化的行为——一切与长度偏置有关的问题都源于此。

2.2 策略梯度：从似然比到 token 级梯度

设轨迹 $\tau = (x, y_{1:T})$，轨迹分布由策略决定（提示分布 $p(x)$ 与环境无关项记为 μ）：

$$p_\theta(\tau) \;=\; p(x)\prod_{t=1}^{T}\pi_\theta(y_t \mid x, y_{<t}), \qquad J(\theta) = \mathbb{E}_{\tau\sim p_\theta}[\,R(\tau)\,], \tag{2.1}$$

其中序列级奖励 $R(\tau) = r_\phi(x, y)$（或验证器的输出）。目标是最大化 $J(\theta)$。

策略梯度定理：完整推导

第 1 步（对分布求导）。对任意由 θ 参数化的密度 p_θ 与函数 R：

$$\nabla_\theta \mathbb{E}_{p_\theta}[R] \;=\; \nabla_\theta \int p_\theta(\tau) R(\tau)\, d\tau \;=\; \int \nabla_\theta p_\theta(\tau)\, R(\tau)\, d\tau.$$

第 2 步（log-derivative 恒等式）。由 $\nabla \log p_\theta = \nabla p_\theta / p_\theta$：

$$\nabla_\theta p_\theta(\tau) = p_\theta(\tau)\, \nabla_\theta \log p_\theta(\tau),$$

代回并重新配权成期望：

$$\nabla_\theta \mathbb{E}_{p_\theta}[R] \;=\; \int p_\theta(\tau)\, \nabla_\theta \log p_\theta(\tau)\, R(\tau)\, d\tau \;=\; \mathbb{E}_{p_\theta}[\,R(\tau)\, \nabla_\theta \log p_\theta(\tau)\,].$$

第 3 步（轨迹因子分解）。将 (2.1) 的因子化代入：

$$\nabla_\theta \log p_\theta(\tau) \;=\; \underbrace{\nabla_\theta \log p(x)}_{=0} \;+\; \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t}),$$

提示分布与环境项与 θ 无关而消失。于是得到 **token 级策略梯度：**

$$\boxed{\nabla_\theta J(\theta) \;=\; \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t}) \;\cdot\; G_t\right]} \tag{2.2}$$

其中 $G_t = \sum_{t'\geq t} \gamma^{t'-t} r_{t'}$ 是从 t 起的回报。**解读：**梯度是“对数似然的梯度 × 未来的回报”——回报高的行为被提高概率，回报低的行为被降低概率，权重恰是回报本身。这正是 REINFORCE；它对 r_ϕ 只要求能打分，不要求可导——这决定了“奖励可以是任何黑盒”这一 RLHF 的根本自由度。■

注意 (2.2) 中每个 token 共享同一轨迹级系数时可以合并；若用 G_t 逐步衰减（$\gamma < 1$），则不同位置获得不同权重。LLM 实践通常 $\gamma = 1$，两种写法一致。

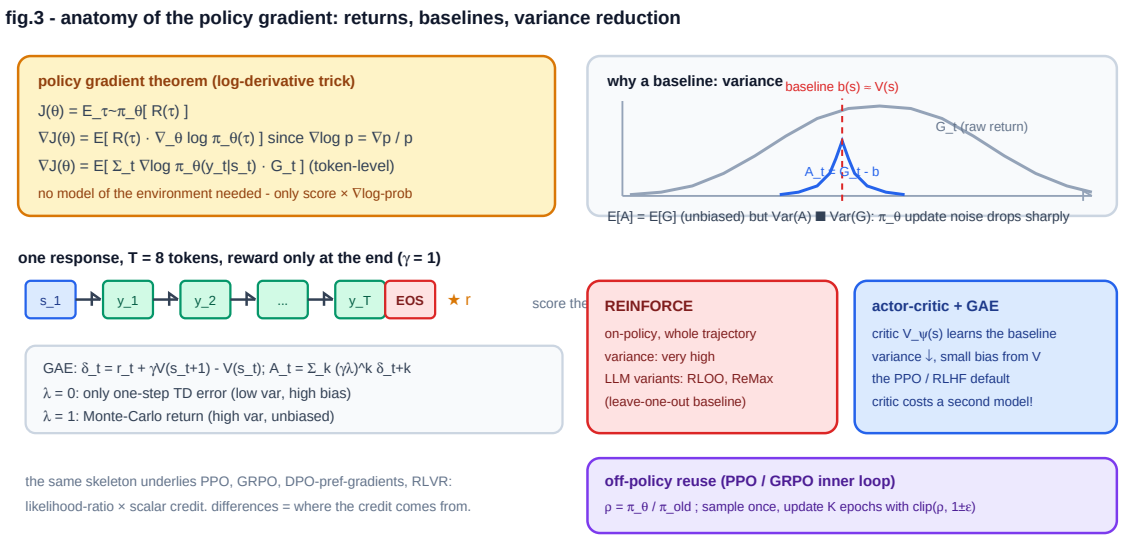


图 2.2 策略梯度的解剖：同一梯度骨架，三种方差级别的实现（REINFORCE / actor–critic / off-policy 复用），以及基线如何收窄梯度的噪声分布。

2.3 基线与优势：方差从哪里来

(2.2) 是无偏的，但方差大得惊人：不同回答的回报可能相差两个数量级，而每个 batch 只有数百条轨迹。标准做法是减去一个**与动作无关的基线** $b(s)$。

基线的无偏性与方差分析

无偏性。对任意不依赖 a_t 的函数 $b(s_t)$：

$$\mathbb{E}_{a\sim\pi_\theta}\big[\, b(s_t)\,\nabla_\theta \log \pi_\theta(a_t \mid s_t)\,\big] = b(s_t)\sum_a \pi_\theta(a \mid s_t)\nabla_\theta \log \pi_\theta(a \mid s_t) = b(s_t)\,\nabla_\theta \underbrace{\sum_a \pi_\theta(a \mid s_t)}_{=1} = 0.$$

因此从 G_t 中减去 $b(s_t)$ 不改变期望，梯度仍然无偏。

方差效应。对单个 (s,a) 样本，估计量 $G_t - b$ 的方差关于 b 求导为零的点在

$$b^\star(s_t) = \mathbb{E}[\, G_t \mid s_t\,] \;\equiv\; V(s_t),$$

即**条件期望回报**。于是最优基线就是状态价值 $V(s)$，残差

$$A(s_t, a_t) \;=\; Q(s_t, a_t) - V(s_t) \;=\; G_t - V(s_t) \tag{2.3}$$

就是**优势函数**：它度量“这个动作比该状态下的平均水平好多少”。V 越准，梯度噪声越小——这正是引入 critic 网络的全部理由。■

在 LLM 场景，$V(s_t)$ 是“当前上下文下期望最终奖励”，例如生成到一半时对“最终答对概率”的预估。估计它需要额外训练一个与策略同尺寸的网络（critic），这是 PPO 的开销主体，也是 GRPO 想绕开的东西（第 七 章）。

2.4 Critic 与广义优势估计（GAE）

critic V_ψ 用 TD 误差训练。定义单步 TD 误差与 λ-回报：

$$\delta_t = r_t + \gamma V_\psi(s_{t+1}) - V_\psi(s_t), \qquad G_t^\lambda = (1-\lambda)\sum_{n\geq 0}\lambda^n G_t^{(n+1)}, \tag{2.4}$$

其中 $G_t^{(n)}$ 是 n 步截断回报。GAE 的推导只需要一个恒等式：

GAE：几何平均的 TD 误差之和

把 n 步回报展开 $G_t^{(n)} = \sum_{k=0}^{n-1}\gamma^k r_{t+k} + \gamma^n V(s_{t+n})$，反复代入 $\gamma V(s_{t+k}) = \delta_{t+k} - r_{t+k} + V(s_{t+k-1})$ 可验证恒等式

$$G_t^\lambda - V(s_t) \;=\; \sum_{k=0}^{\infty}(\gamma\lambda)^k\delta_{t+k}.$$

于是定义 **GAE 优势**：

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} \;=\; \sum_{k=0}^{\infty}(\gamma\lambda)^k\,\delta_{t+k} \;=\; \delta_t + \gamma\lambda\,\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \cdots \tag{2.5}$$

两个端点：$\lambda = 0$ 时 $\hat{A}_t = \delta_t$（单步 TD：方差最低、偏差来自 critic 的误差）；$\lambda = 1$ 时 $\hat{A}_t = G_t - V(s_t)$（蒙特卡洛残差：无偏但方差高）。λ 在偏差与方差之间**几何插值**——这是 PPO 系实践取 $\lambda \in [0.95, 1]$ 的原因。终端奖励特例（$\gamma = 1$、$r_{t<T} = 0$、$r_T = r$）：$\delta_t = r$ 对任意准确的 V 成立仅当 V 处处相等；实际上 (2.5) 退化为一个把“路径上信任 critic 的程度”参数化的公式，工程实现里 LLM-PPO 通常直接取 $\lambda = 1$、由 critic 提供基线。■

2.5 重要性采样与 off-policy 修正

策略梯度 (2.2) 要求样本来自当前策略。若样本来自旧策略 π_{old}（例如同一批 rollout 被重复使用 K 个 epoch，或异步系统中样本天然滞后），需要重要性加权修正分布不匹配：

$$\nabla_\theta J(\theta) \;=\; \mathbb{E}_{\tau\sim\pi_{\text{old}}}\left[\frac{\pi_\theta(\tau)}{\pi_{\text{old}}(\tau)}\sum_t \nabla_\theta \log \pi_\theta(y_t \mid s_t)\, G_t\right], \qquad \frac{\pi_\theta(\tau)}{\pi_{\text{old}}(\tau)} = \prod_{t=1}^{T}\frac{\pi_\theta(y_t \mid s_t)}{\pi_{\text{old}}(y_t \mid s_t)}. \tag{2.6}$$

重要性采样为何可行、何时失效

恒等式。对任意两个支撑覆盖的分布 p, q 与函数 f：$\mathbb{E}_p[f] = \mathbb{E}_q\left[\frac{p}{q}f\right]$，一行验证：$\int q\cdot\frac{p}{q}f = \int pf$。**代价。**估计量的二阶矩恰为 $\mathbb{E}_q[(p/q)^2] = 1 + \chi^2(p\|q)$，且由不等式 $\log u \leq u - 1$ 可得下界 $\mathbb{E}_q[\rho^2] \geq \exp\big(2\,\mathrm{KL}(q\|p)\big)$。经验法则：*KL 每增大 0.35 nat，方差至少翻一倍。*序列级比率 (2.6) 是 T 个 token 比率的乘积——哪怕每步只偏 2%、T=1000 时乘积也会天文数字式爆炸/塌缩。**因此所有实用算法（PPO/GRPO）都放弃序列级比率，改用 token 级比率并配合 clip 截断：**

$$\rho_t = \frac{\pi_\theta(y_t \mid s_t)}{\pi_{\text{old}}(y_t \mid s_t)}, \qquad \mathrm{clip}(\rho_t, 1-\epsilon, 1+\epsilon). \tag{2.7}$$

clip 在 θ 与 π_{old} 偏离超过 ϵ 后**切断梯度**，等价于把每次更新的“信任域”限制在比率 $1\pm\epsilon$ 内。这就是 PPO 目标（第 五 章）的全部来源：*重要性采样的无偏性换取可行性，clip 换回稳定性。*■

2.6 KL 散度：约束、几何与两种不对称

RLHF 与 DPO 的目标函数中反复出现 $\mathrm{KL}(\pi_\theta\|\pi_{\text{ref}})$。两个方向并不等价：

前向与反向 KL 的不对称（mode-covering 与 mode-seeking）

设真实分布 p 多峰、近似分布 q 单峰：

$\mathrm{KL}(p\|q)=\mathbb{E}_p\left[\log\frac{p}{q}\right]$ （前向 / inclusive）：q 必须**覆盖**p 的所有峰，否则罚 $\log\frac{p}{0}\to\infty$;

$\mathrm{KL}(q\|p)=\mathbb{E}_q\left[\log\frac{q}{p}\right]$ （反向 / exclusive）：q 只需**抓住**p 的一个峰，均值附近质量集中即可.

RLHF 中约束的方向是 $\mathrm{KL}(\pi_\theta\|\pi_{\text{ref}})$（**反向**，以新策略为期望）：

- 语义上是“从新策略采样时，与参考的对数差异”——直接约束**模型实际生成**的行为；
- 副作用是 mode-seeking：策略可能收缩到参考分布的某个模式上（这正是“熵坍缩”的几何解释之一，第 十四 章）；
- 二阶近似下 $\mathrm{KL}(\pi_\theta\|\pi_{\text{ref}})\approx\frac{1}{2}\Delta\theta^{\mathsf{T}}F(\theta_0)\Delta\theta$ （F 为 Fisher 信息矩阵），因此 KL 约束 $\approx$ 自然梯度下的信赖域半径——这解释了为什么“KL 小幅上升”是 PPO 每步内层 epoch 的天然停止信号。

工程含义：监控 $\mathrm{KL}(\pi_\theta\|\pi_{\text{ref}})$ 的经验估计 $\widehat{\mathrm{KL}}=\frac{1}{T}\sum_t\left[\log\pi_\theta-\log\pi_{\text{ref}}\right]$（token 平均 nats），健康训练通常把它压在 10^{-2}–10^{-1} 量级。■

2.7 LLM 场景的五个特化事实

把以上一般理论与 LLM 结构结合，得到贯穿后文的五个事实：

1. **动作空间巨大而离散**（$|\mathcal{V}|\sim10^5$），但策略梯度不需要遍历动作——只对被采到的 token 求梯度，天然稀疏高效。
2. **终端奖励 $\Rightarrow$ 信用分配扁平**：没有过程奖励时，一次回答内所有 token 拿到相同的学习信号（GRPO 的全部合法性来源）。
3. **每个 token 都是一个决策点**，因此 KL 预算按 token 计：序列对数似然比是 T 个 token 比率之和 $\Rightarrow$ 长序列更容易越界，需要逐步约束而非序列级约束。
4. **提示 token 必须掩码**：$\pi_\theta(y_t\mid x,\cdot)$ 中对提示部分的对数似然既不受策略控制（prompt 固定）也不该被优化——所有损失只在回答 token 上平均。
5. **长度是被优化的行为**：任何“奖励 $\div$ 长度”的归一化方式都隐式定义了一个长度偏好（Dr. GRPO 与 DAPO 的修正点，第 七 章）。

第 2 章要点

- 策略梯度 $=\mathbb{E}[\nabla\log\pi\cdot A]$：形式上极简，全部工程难度在**降低 A 的方差**与**控制 off-policy 程度**。
- 基线（critic 或组均值）免费降方差且不引入偏差；重要性采样允许样本复用但方差按 $\exp(2\ \mathrm{KL})$ 放大 $\Rightarrow$ 必须 clip。
- KL 约束不是“对齐品味问题”，而是数学必需：它把每步更新限制在 Fisher 几何的信任

域内，同时把策略锚定在保留预训练能力的参考分布上。

第三章 监督微调：格式、任务与行为克隆

SFT 是后训练的第一站，也是最容易被低估的一站。它的作用是把一个“续写器”切换成“应答器”：学会**指令跟随、对话格式、角色边界**。虽然概念上是普通交叉熵训练，但数据构造、损失掩码与打包方式上的细节决定了后续 RL 阶段的起点质量——RL 不会凭空创造行为，只会放大 SFT 已展示过的行为。

3.1 目标函数与 teacher forcing

给定示范集 $\mathcal{D} = \{(x_i, y_i^*)\}_{i=1}^{N}$（$N \sim 10^4$–$10^6$），SFT 最大化示范回答的似然（teacher forcing：训练时以真实前缀为条件，不走采样）：

$$\mathcal{L}_{\mathrm{SFT}}(\theta) \;=\; -\,\mathbb{E}_{(x,y^*)\sim\mathcal{D}}\left[\frac{1}{|y^*|}\sum_{t=1}^{|y^*|}\log \pi_\theta(y_t^* \mid x, y_{<t}^*)\right]. \tag{3.1}$$

两个实现细节。 （i）**提示掩码**：$\log \pi_\theta(\cdot \mid x)$ 中对提示 token 的项不计入损失——我们不想教模型“预测用户的提示”，只想教它“给定提示如何回应”。（ii）**归一化选择**：按 token 平均（(3.1)）还是按序列求和？按 token 平均会系统性降低长回答的梯度权重；按序列求和会让长样本主导 batch。现代框架默认 per-token 全局平均（整个 batch 的所有回答 token 一起平均）。

3.2 数据工程：示范从哪里来

SFT 的一个核心经验规律是**质量压倒数量**（LIMA：1,000 条精心筛选的样本即可对齐 GPT-4 级对话格式；Qwen 系列公开报告同样强调“数据清洗 > 数据扩量”）。原因在信息论上很直观：SFT 的作用是把行为**聚拢**到一个窄的输出流形上——它需要的是这个流形的*干净的代表点*，而不是对整个语言空间的覆盖。

来源	代表方法	要点与坑
人类撰写	OpenAI 早期标注员数据	质量上限最高，成本也最高；风格一致性靠标注规范
强模型合成	Self-Instruct / Evol-Instruct	用强模型生成（指令, 回答）对；Evol-Instruct 对指令逐步加难；需去重与毒性过滤
真实对话回流	部署日志 + 人工改写	与部署分布最接近；注意隐私与风格漂移
自我改进	rejection sampling（第 八 章）	RL 模型自己采样 n 条取最好，反哺 SFT——R1 的 cold-start 数据即此
工具与代码	执行反馈筛选	代码/数学答案用验证器筛选，天然高纯度

表 3.1　指令数据来源谱系。

3.3 打包（packing）与位置编码的交互

示范多为短问答（数百 token），而训练序列通常 4k–8k。**打包**把多条样本拼进一条序列以填满上下文，配合注意力掩码（block-diagonal attention）阻断样本间信息流：

$$\text{AttnMask}[i,j] = \begin{cases} 0, & i,j \text{ 属于同一打包样本} \\ -\infty, & \text{否则} \end{cases} \tag{3.2}$$

若不阻断（naive packing），交叉污染一般可控但会在“跨样本边界”处产生轻微分布偏移；对 RoPE 等相对位置编码，现代实现（FlashAttention 的 varlen 接口、各家框架的 packed 输入）以“逻辑位置重置”处理，正确且零开销。工程上打包把 SFT 吞吐提升 2–5×，是 SFT 与 RLHF 训练效率差异的重要一环（RL 的 rollout 天然无法打包——每条序列必须独立采样）。

3.4 SFT 作为行为克隆：它的理论边界

把 SFT 放回 RL 视角：它等价于对某个**专家策略** π_E 做行为克隆（BC），即最小化 $\mathrm{KL}(\pi_E \| \pi_\theta)$。BC 的两个经典缺陷在 LLM 上都成立：

1. **复合误差（exposure bias）**。BC 只见过专家轨迹的“好”前缀；采样一旦偏离示范分布，模型处于分布外状态，后续逐步恶化。对 LLM 的表现是：多轮对话里一旦前面说错，后面越错越远。

2. **无负反馈**。交叉熵只在示范上定义，模型不知道“什么是坏”——它会把一切没见过的行为视为同等坏。RLHF 的偏好对比与 RL 的负优势恰好补上这块信息。

SFT 与 RL 的分工（为什么两者都要）

SFT 给出**高密度的行为先验**（每个 token 都是监督），RL 给出**低密度但含负信号的价值引导**（每条序列一个标量）。实践检验：纯 SFT 到达“格式正确但平庸”；在 SFT 之上做几千步 RL 的模型在几乎所有开放式评测上超过更大的纯 SFT 模型（InstructGPT 的 1.3B RLHF 模型优于 175B SFT 模型的对照结论即此）。反过来，跳过 SFT 直接 RL（R1-Zero）可行但产物不可读——见第 八 章。

3.5 参数高效微调（LoRA）在后训练中的角色

后训练几乎总是用比预训练更“省”的参数更新方式，LoRA 最为常见：冻结 W_0，学习低秩增量 $\Delta W = BA$（$B \in \mathbb{R}^{d\times r}$，$A \in \mathbb{R}^{r\times k}$，$r \ll d$）：

$$h = W_0 x + \frac{\alpha}{r} BAx. \tag{3.3}$$

对 RLHF 的三个关键影响：（i）显存与同步成本骤降——rollout 引擎每步只需同步 LoRA 权重（MB 级）而非全权重（GB 级）；（ii）隐式的正则化——低秩约束天然限制策略偏离，部分替代 KL 约束；（iii）对齐质量通常略逊于全参（尤其推理任务需要模型改变长程行为时），因此 2025 年后的大模型 RLVR 多回到全参或高秩（r=64–256）LoRA。

第 3 章要点

- SFT = 在干净示范上的 teacher-forcing 交叉熵；提示掩码、per-token 归一化与正确的打包掩码是三个最常见的实现错误。
- 质量压倒数量；示范数据决定 RL 的行为上限——RL 放大已有模式，不创造新模式。
- BC 的复合误差与“无负反馈”是 SFT 的理论天花板，由此引出奖励建模与 RL 的必要性。

第四章 奖励建模：从人类偏好的标量化

RL 的全部力量来自奖励信号的质量。本章处理三个问题：**偏好如何变成标量**（Bradley–Terry 模型及其推导）、**标量何时可靠**（outcome 与 process 两种监督粒度）、**何时必然失真**（reward hacking 的机制与对策）。

4.1 Bradley–Terry 模型：偏好比较的极大似然

人类给“y_w 比 y_l 好”的**比较判断**远比给绝对分数容易、也远比分数一致（评分尺度漂移，两两比较不受影响）。Bradley–Terry（1952）把比较结构公理化：存在一个潜在分数 $r(\cdot)$，使得比较结果服从

$$P(y_w \succ y_l \mid x) \;=\; \frac{e^{r(x,y_w)}}{e^{r(x,y_w)} + e^{r(x,y_l)}} \;=\; \sigma\big(r(x,y_w) - r(x,y_l)\big), \tag{4.1}$$

其中 σ 为 logistic 函数。RM 训练即最大化偏好对的似然：

RM 损失与梯度：三行推导

对每个偏好对 (x, y_w, y_l)，参数化 $r = r_\phi$，负对数似然为

$$\mathcal{L}_{\mathrm{RM}}(\phi) = -\,\mathbb{E}_{(x,y_w,y_l)}\left[\log \sigma\big(r_\phi(x,y_w) - r_\phi(x,y_l)\big)\right]. \tag{4.2}$$

记 $\Delta = r_\phi(x,y_w) - r_\phi(x,y_l)$，则 $\nabla_\phi \log\sigma(\Delta) = (1-\sigma(\Delta))\,\nabla_\phi\Delta$，即梯度**以 $(1-\sigma(\Delta))$ 为权重**拉开两个分数：

$$\nabla_\phi \mathcal{L} = -\,\mathbb{E}\Big[\big(1-\sigma(\Delta)\big)\Big(\nabla_\phi r_\phi(x,y_w) - \nabla_\phi r_\phi(x,y_l)\Big)\Big].$$

解读：y_w 与 y_l 分差已经很大（模型对偏好“很有把握”）时权重 $\to 0$，自动停训——**内建的自动早停**。代价：继续压分差就是过拟合偏好对本身的噪声，故实践中以留出集上的 *pairwise* 准确率（而非损失）做早停指标，准确率 65–72% 即应停。■

Plackett–Luce 推广。 若标注是 K 个回答的完整排序 $y^{(1)} \succ \cdots \succ y^{(K)}$，似然推广为逐对因子的乘积（Luce 选择公理）：

$$P\big(y^{(1)} \succ \cdots \succ y^{(K)}\big) = \prod_{i=1}^{K-1} \frac{e^{r(x,y^{(i)})}}{\sum_{j\ge i} e^{r(x,y^{(j)})}}, \tag{4.3}$$

一条 K 排序提供 $K-1$ 个“软对”，信息效率高于手工两两比较——这是 RLHF 标注平台（成对呈现、全局排序）的统计学基础。

实现要点。 （i）RM 通常从 SFT/基座检查点初始化——它必须读懂回答，预训练权重免费提供这一点；（ii）输出头是一个新增线性层，把最后一个 token 的隐状态映射为标量；（iii）**长度偏置**：若人类偏爱长回答，RM 会学到“长 $\approx$ 好”，RL 将立刻放大它——需在数据里平衡长短对，或显式做长度控制；（iv）排序对称增广（交换 w/l 位置重采样）防止位置泄漏。

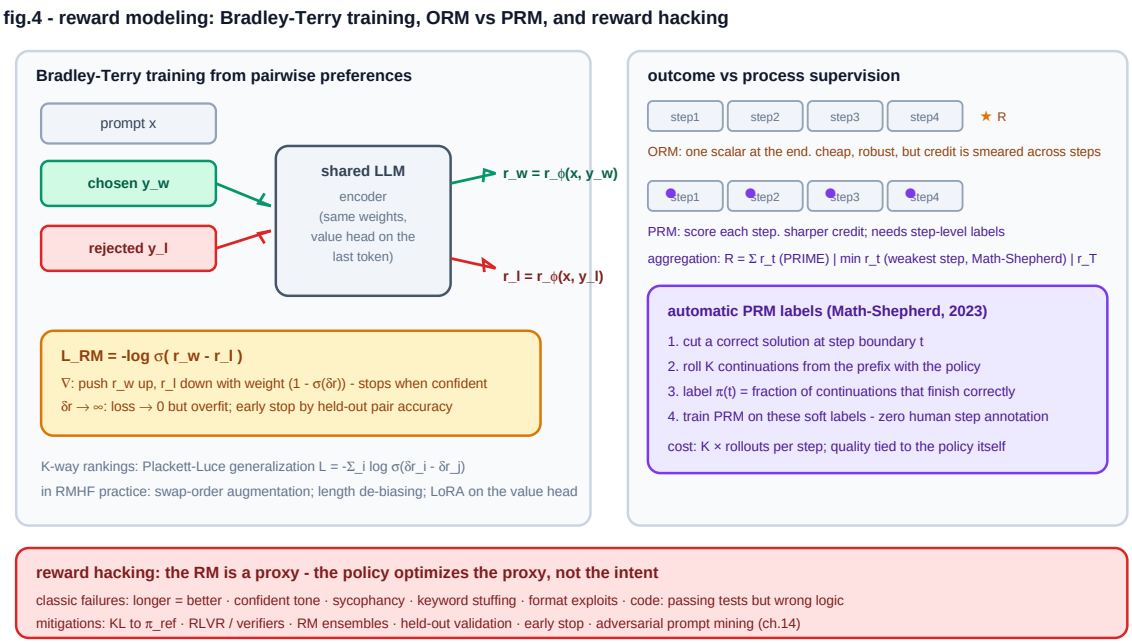


图 4.1 奖励建模全景：BT 训练回路（左）、outcome/process 两种粒度（右）、自动过程标注（右下）与 reward hacking（底）。

4.2 outcome 与 process：两种监督粒度

ORM（outcome reward model）。 对完整回答打一个标量。便宜、鲁棒、与 RLHF 主循环天然兼容（每条 rollout 一次前向）。缺点是**信用分配扁平**：回答中好步骤与坏步骤拿到同样的分数。

PRM（process reward model）。 对推理的每个步骤打分，$R = \sum_t r_t$ 或 $R = \min_t r_t$（最弱环节）。信用分配锐利，是 o1 式“在步骤层面做搜索”的前提（第 九 章）。难点是标注：人工逐步标注成本高且不一致（OpenAI 的 PRM800K 花费巨大）。

Math-Shepherd：蒙特卡洛自动标注过程奖励

关键观察：步骤 t 的“好坏”可以定义为**从该步骤出发最终答对的概率**——这可以用策略本身估计而无需人工：

$$\hat{\pi}(t; x, y_{\le t}) = \frac{1}{K} \sum_{k=1}^{K} \mathbf{1}\big[\text{rollout}_k(x, y_{\le t}) \text{ 最终验证通过}\big], \tag{4.4}$$

即以当前前缀为条件采样 K 条续写到结束，统计通过率。$\hat{\pi}(t)$ 关于 t 单调的序列对应“正确解”；标签噪声来自策略本身的能力上限（早期策略弱时标注质量低）——于是 PRM 与策略共同进化（PRIME 类在线过程奖励，第 八 章）。成本：每步 K 次续写，总计 $O(TK)$ 次生成，故只在数学/代码等有验证器的域可行。■

4.3 Reward hacking：代理目标必然被攻破

机制：优化压强下的代理偏差

RM 是人类偏好的**代理**（proxy）：$r_\phi \approx r^\star + \epsilon$，其中 $r^\star$ 是真实意图、ϵ 是 RM 自己的系统性偏差。RL 优化的压强集中在 r_ϕ 上，策略会沿 ϵ 的方向移动：

$$\max_\pi \mathbb{E}\big[r_\phi\big] \;\neq\; \max_\pi \mathbb{E}\big[r^\star\big], \qquad \text{且 } KL(\pi_\theta \| \pi_{\text{ref}}) \uparrow \Rightarrow \epsilon \text{ 被放大的自由度} \uparrow . \tag{4.5}$$

Goodhart 分类里这是典型的“代理回归”（regressional Goodhart）：任何带噪代理在被充分优化后都会失真，失真的速度正比于噪声与优化压强的乘积。实证案例：长度膨胀、谄媚（附和用户）、格式套路（罗列/加粗/免责声明堆砌）、代码通过测试但逻辑错误、数学答案抄近路。

对策按效力排序：

1. **换掉代理**（根治）：可验证域直接用规则验证器（RLVR，第 八 章）；
2. **限制自由度**（缓解）：KL 约束 + 早停 + 小学习率；RM 集成（多个 RM 分数平均/取最小）显著缩小 ϵ 的可利用面；
3. **对抗性数据**（压制）：把已发现的 hacking 行为做成新的偏好对回灌 RM （iterative RLHF 的核心循环）；
4. **监控**（报警）：held-out 任务上人工盲评、长度/格式分布跟踪（第 十四 章的诊断清单）。

第 4 章要点

- BT 模型把“比较”变“分数”：$\mathcal{L} = -\log \sigma(\Delta r)$，梯度自动带置信权重，早停看留出集 pairwise 准确率（65–72%）。
- ORM 信用扁平、PRM 信用锐利但标注贵；Math-Shepherd 用 MC 续写把标注成本转嫁给策略自身，催生在线过程奖励。
- Reward hacking 不是 bug 而是**代理目标的必然结局**；一切对策都在做三件事之一：换掉代理、限制压强、给代理补课。

第五章 RLHF-PPO：完整剖析

PPO 自 2017 年提出以来一直是 LLM 在线 RL 的骨架。本章不满足于复述目标函数，而是回答三个“为什么”：为什么 KL 约束长成那样（两种等价实现）、为什么 clip 目标是 min 而非 max、以及为什么一个看似标准的 RL 算法在 LLM 上需要那么多“土法修正”才跑得稳。

5.1 RLHF 的目标：KL 约束下的奖励最大化

RLHF 的顶层目标由 InstructGPT 固定下来：

$$\max_{\pi_\theta} \ \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\big[\, r_\phi(x,y)\,\big] \ - \ \beta\, \mathbb{E}_x\Big[\ \mathrm{KL}\left(\pi_\theta(\cdot\mid x)\,\big\|\,\pi_{\mathrm{ref}}(\cdot\mid x)\right)\Big]. \tag{5.1}$$

直觉读法：**期望奖励减去偏离税**。β 是税率：$\beta\to 0$：全力冲奖励（reward hacking 敞开大门）；$\beta\to\infty$：一步不走（回到 SFT 模型）。该目标存在闭式最优解（第 六 章推导）——这一点是 DPO 的出发点，也说明 PPO 实际上在以随机梯度方式逼近同一个解。

两种等价实现。 工程上 (5.1) 有两条实现路线，**必须二选一**：

1. **奖励整形（InstructGPT/OpenAI 路线）**。把 KL 折进每个 token 的即时奖励：

$$r'_t \ = \ r_t \ - \ \beta \log \frac{\pi_\theta(y_t\mid s_t)}{\pi_{\mathrm{ref}}(y_t\mid s_t)}, \qquad r_t = \begin{cases} r_\phi(x,y), & t=T \ \text{(ORM)} \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

   然后完全按标准 RL 处理 r'（GAE、critic、TD）。优点：KL 通过优势函数作用于每个决策，路径连贯；缺点：critic 必须学习“整形后”的价值，KL 不可直接监控。

2. **损失惩罚（Llama-2 等开源路线）**。优势用原始 r 计算，KL 作为独立正则项加进损失：

$$\mathcal{L}(\theta) \ = \ -\,\mathbb{E}_t\Big[\min\left(\rho_t\hat{A}_t,\ \mathrm{clip}(\rho_t,1{-}\epsilon,1{+}\epsilon)\hat{A}_t\right)\Big] \ + \ \beta'\,\mathbb{E}_t\Big[\log\tfrac{\pi_\theta(y_t|s_t)}{\pi_{\mathrm{ref}}(y_t|s_t)}\Big]. \tag{5.3}$$

   优点：KL 逐 token 可观测、可设预算（如目标 0.05 nats/token）；缺点：KL 只作用于梯度而不影响回报，理论上“税收”作用更间接。

经典 bug：KL 双重计入

若同时采用 (5.2) 的整形奖励和损失惩罚项，KL 被计入两次，实际税率是 2β——训练表现“异常温吞”，且极难从日志中发现。TRL/OpenRLHF 的默认模板互不混用；自己实现时在配置里显式断言二选一。

5.2 PPO-clip 目标：逐项剖析

把第 二 章的 off-policy 修正套上 token 级信任域，得到 PPO 的策略损失（以最大化写）：

$$\mathcal{L}^{\mathrm{CLIP}}(\theta) \ = \ \mathbb{E}_t\Big[\min\left(\rho_t\hat{A}_t,\ \mathrm{clip}(\rho_t,1{-}\epsilon,1{+}\epsilon)\,\hat{A}_t\right)\Big], \qquad \rho_t = \frac{\pi_\theta(y_t\mid s_t)}{\pi_{\mathrm{old}}(y_t\mid s_t)}. \tag{5.4}$$

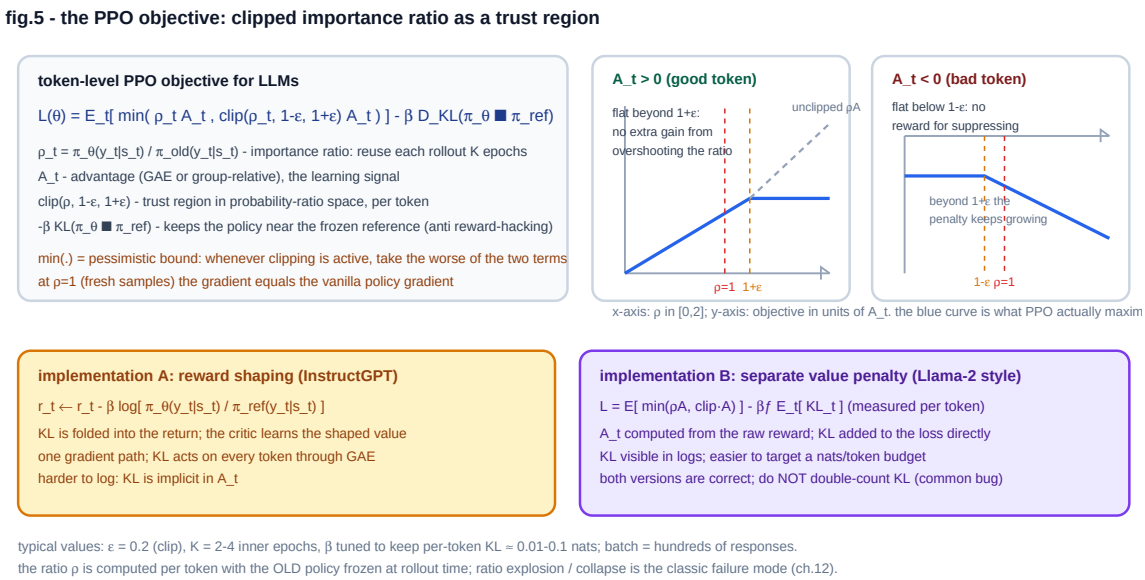


图 5.1 PPO 目标的解剖：比率、裁剪与两种 KL 实现。右图给出 $\hat{A} > 0$ 与 $\hat{A} < 0$ 时目标关于 ρ 的分段线性形状——这是所有 PPO 系算法的“有效景观”。

逐项问答：目标函数设计动机

- **为什么用 min？** min 是悲观下界：对 $\hat{A} > 0$，ρ 超过 $1{+}\epsilon$ 后目标封顶——**过度乐观的步子不再有奖励**；对 $\hat{A} < 0$，ρ 低于 $1{-}\epsilon$ 后目标封底——**过度压制也不再有额外回报**。两端都切断梯度，防止“一步内跑出信任域”。
- **为什么在 $\rho = 1$ 处与 vanilla 策略梯度一致？** $\rho_t = 1 \Rightarrow \nabla_\theta \rho_t\big|_{\theta=\theta_{\text{old}}} = \nabla \log \pi_\theta(y_t \mid s_t)$，即每个 batch 的第一次更新等价 REINFORCE；从第二次内层 epoch 起 clip 才开始工作。
- **为什么逐 token 而不是逐序列？** 见第 2.5 章——序列比率的方差随长度指数爆炸；token 级比率方差 $O(1)$。
- **clip 与 KL 惩罚是否冗余？** 不冗余：clip 管**单次更新内的步长**（minibatch 尺度），KL 管**全局分布漂移**（训练全程尺度）。前者是刹车，后者是围栏。

5.3 优势与价值：critic 怎么训

采用 (5.2) 时，critic 学习的是**整形价值**：

$$V^\pi(s_t) = \mathbb{E}_\pi\Big[\sum_{t' \geq t} r'_{t'}\Big] = \mathbb{E}_\pi\Big[\sum_{t' \geq t} \big(r_{t'} - \beta \mathrm{KL}_{t'}\big)\Big], \tag{5.5}$$

即“从这里出发，期望（奖励 − 未来 KL 税）”。critic 损失为标准的复合 TD 回归：

$$\mathcal{L}_V(\psi) = \mathbb{E}_t\big[\big(V_\psi(s_t) - V_t^{\text{targ}}\big)^2\big], \qquad V_t^{\text{targ}} = \hat{A}_t + V_{\psi_{\text{old}}}(s_t), \tag{5.6}$$

其中 V^{targ} 复用 GAE 中的回报目标（PPO 惯例，避免额外前向）。**预训练价值头的代价**：V 从随机初始化开始时前几百步的 $\hat{A}$ 全是噪声，InstructGPT 的做法是对 critic 额外做一轮“价值

预训练”（对 SFT 数据回归 RM 分数）。GRPO 直接删除 critic（第 七 章），动机之一就是省掉这一整套。

批归一化。 实践几乎总是把 $\hat{A}$ 做 batch 级白化：$\hat{A} \leftarrow (\hat{A} - \mu_B)/(\sigma_B + 10^{-8})$。这不改变方向、只重整尺度；它对学习率的鲁棒性贡献巨大——不同任务（奖励量纲 0/1 vs RM 连续分数）无需单独调 lr。

5.4 一个完整 PPO 步的解剖

LLM-PPO 主循环（伪代码，对应图 5.2）

1. 采一个 prompt 批 $x_{1:B}$；将 actor 权重同步到 rollout 引擎；
2. 引擎生成回答 $y^{(i)} \sim \pi_{\text{old}}(\cdot \mid x_i)$，记录每 token 的 $\log \pi_{\text{old}}(y_t \mid s_t)$ 与长度；
3. 前向 r_ϕ 得序列奖励 $r^{(i)}$；前向 π_{ref} 得每个 token 的 KL 项；
4. 组装整形奖励 (5.2)（或登记独立 KL 项）；
5. 前向 critic 得 $V_\psi(s_t)$，计算 GAE $\hat{A}_t$，白化；
6. **内层循环**（$K = 1$–4 epochs，切 minibatch）：前向 actor 得 $\log \pi_\theta$，计算 ρ_t 与 (5.4)，反传，更新 θ；同时按 (5.6) 更新 ψ；
7. 监控：$\bar{r}$、$\overline{\text{KL}}$、clip 触发率、熵、长度分布；KL 超预算则回滚或缩 lr；
8. 回到 1。整个外层循环即训练步；$\pi_{\text{old}} \leftarrow \pi_\theta$。

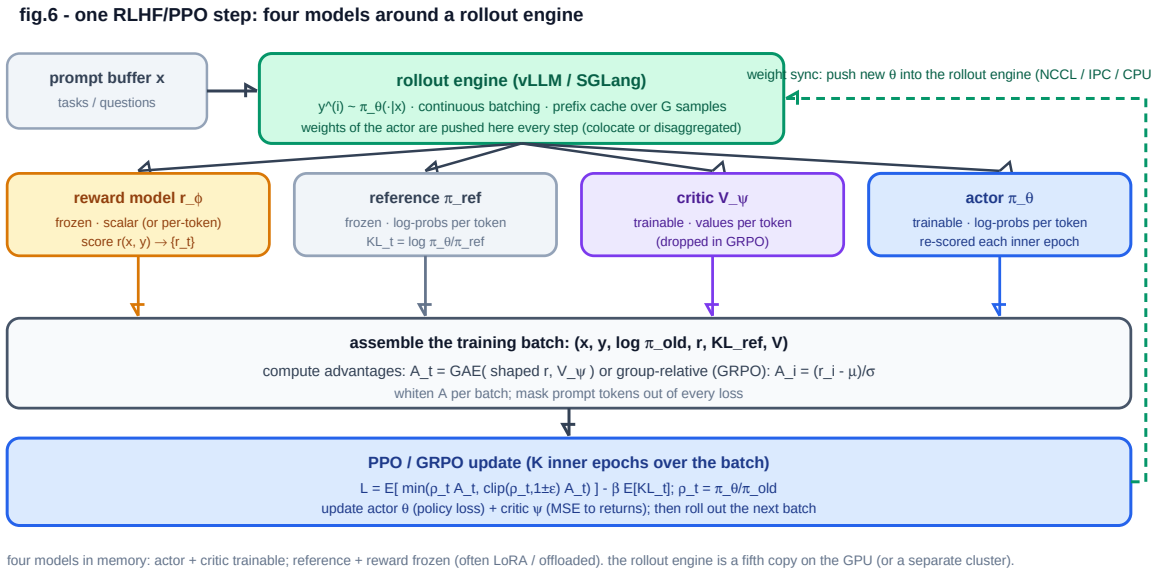


图 5.2 一次 RLHF-PPO 步中四个模型与 rollout 引擎的数据流。灰/紫/蓝三色前向（参考、critic、actor）只是打分，绿色的生成才是真正的开销。

5.5 已披露的真实配置

系统	模型	β / KL	ϵ, K	备注
InstructGPT (2022)	1.3B–175B	0.12–0.2（shaped）	0.2 / 4	RM 6B 独立训练；PPO lr 3×10^{-6}–6×10^{-6}
Llama-2-Chat (2023)	7B–70B	损失惩罚制	0.14-0.2 / 1-2	安全 + 有用双 RM，几何平均组合；迭代 5 轮以上
Anthropic 各代	8B–52B	早期 RLHF	—	Constitutional AI：RM 由 AI 反馈训练
DeepSeekMath (2024)	7B	—	0.2 / 1	首个公开 GRPO（无 critic）配方
DeepSeek-R1 (2025)	671B MoE	KL $\to$ 0 目标	0.2 / 1	RLVR；KL 系数调度；工程细节见第八 章

表 5.1 代表性系统的公开超参数（详细出处见参考文献）。

已知病害速查。 （i）**比率爆炸**：K 过大或 lr 过高 $\Rightarrow$ ρ 大量越界、clip 触发率 $>30\%$、策略发散——降 K、降 lr。（ii）**熵坍缩**：温度不足时策略迅速变尖，探索丧失（第 十四 章）。（iii）**critic 滞后**：价值函数跟不上策略 $\Rightarrow$ $\hat{A}$ 系统性偏差；症状是奖励上翘但评估集不动。（iv）**长度膨胀**：RM 偏长，或整形奖励的 KL 税对长序列更松。

第 5 章要点

- RLHF 目标 = 期望奖励 $-$ $\beta\times$KL 偏离税；KL 的两种实现（整形奖励 / 损失惩罚）等价，**绝不能并用**。
- PPO-clip 的全部智慧在 min：对好 token 封顶、对坏 token 封底，切断越界梯度；clip 管单步步长，KL 管全程漂移。
- 一次 PPO 步 = 采样 $\to$ 三次打分 $\to$ GAE $\to$ K 次内层更新；墙钟大头永远在采样，为第 十一 章的系统设计埋下伏笔。

第六章 直接偏好优化：闭式解与分类损失

2023 年，Rafailov 等人注意到一件被整个领域忽略的事：RLHF 目标 (5.1) 存在**闭式最优解**。由此可以反向把奖励表示成策略的函数，再塞回 Bradley–Terry 似然——整个 RLHF 流水线（RM + 采样 + PPO）坍缩成一个*偏好对上的分类损失*。这就是 DPO。理解它的最好方式是完整走一遍这条推导。

6.1 从 RLHF 目标到闭式最优策略

DPO 的五步推导

目标。固定提示 x，对回答分布 $\pi(\cdot \mid x)$（带 KL 约束）：

$$\max_{\pi(\cdot|x)} \; \mathbb{E}_{y\sim\pi(\cdot|x)}\big[r(x,y)\big] \;-\; \beta\,\mathrm{KL}\big(\pi(\cdot \mid x)\,\big\|\,\pi_{\mathrm{ref}}(\cdot \mid x)\big). \tag{6.1}$$

第 1 步（点态拉格朗日）。把 KL 展开，对每个 y 的概率质量做变分（或写拉格朗日函数对 $\pi(y \mid x)$ 求导并归一化）：

$$\mathcal{L} = \sum_y \pi(y \mid x)\Big[r(x,y) - \beta \log \frac{\pi(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)}\Big] + \lambda\Big(\sum_y \pi(y \mid x) - 1\Big).$$

对 $\pi(y \mid x)$ 求导置零：$r - \beta\big(\log \frac{\pi}{\pi_{\mathrm{ref}}} + 1\big) + \lambda = 0$，解出

$$\pi^*(y \mid x) \;=\; \frac{1}{Z(x)}\,\pi_{\mathrm{ref}}(y \mid x)\,\exp\big(r(x,y)/\beta\big), \qquad Z(x) = \sum_{y'} \pi_{\mathrm{ref}}(y' \mid x)\, e^{r(x,y')/\beta}.$$

第 2 步（结构解读）。最优策略是参考策略的 **Boltzmann 再加权**：奖励只以 $\exp(r/\beta)$ 的乘子改变相对概率，β 是偏离温度；$Z(x)$ 只依赖提示，是归一化常数。**第 3 步（反解奖励）。**对 (6.1) 的解取对数移项：

$$r(x,y) \;=\; \beta \log \frac{\pi^*(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)} + \beta \log Z(x). \tag{6.2}$$

关键一步：任何策略都定义了一个隐式奖励——不需要知道 $Z(x)$ 也能用。**第 4 步（代入 BT）。**把 (6.2) 代入偏好概率 (4.1)：两个 $Z(x)$ 在差分中**相消**：

$$P(y_w \succ y_l \mid x) = \sigma\Big(\beta \log \frac{\pi^*(y_w \mid x)}{\pi_{\mathrm{ref}}(y_w \mid x)} - \beta \log \frac{\pi^*(y_l \mid x)}{\pi_{\mathrm{ref}}(y_l \mid x)}\Big).$$

第 5 步（换元即损失）。用 π_θ 扮演 π^*，取负对数：

$$\boxed{\mathcal{L}_{\mathrm{DPO}}(\theta) = -\,\mathbb{E}_{(x,y_w,y_l)}\left[\log \sigma\Big(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\mathrm{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\mathrm{ref}}(y_l \mid x)}\Big)\right]} \tag{6.3}$$

■

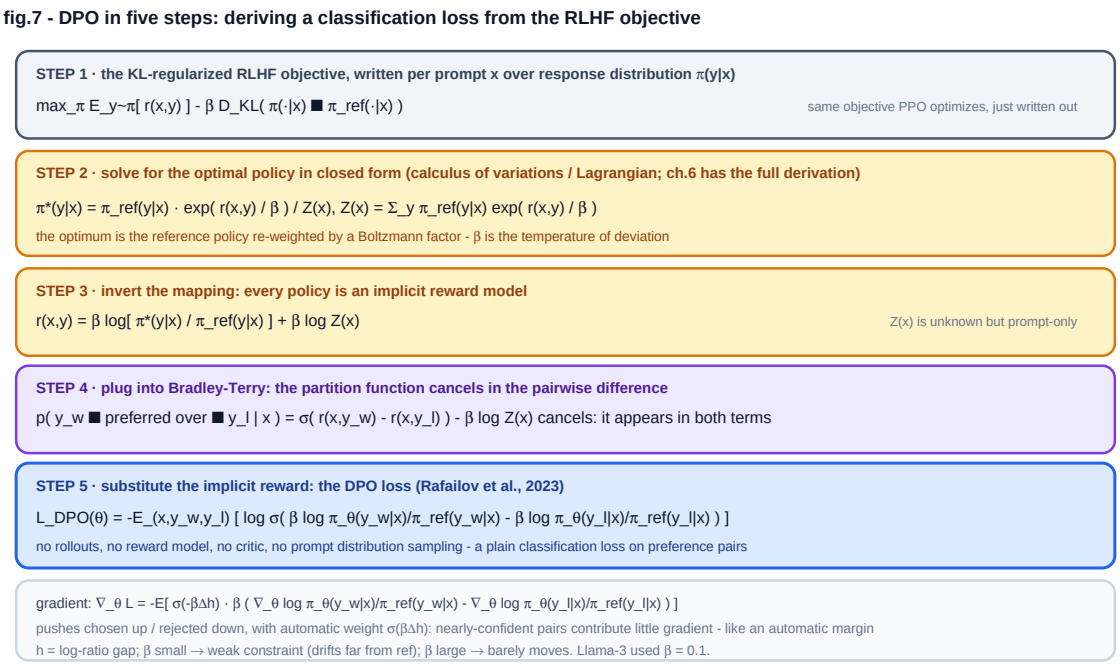


图 6.1 DPO 的推导是五步单向链条：KL 目标 → Boltzmann 闭式解 → 反解隐式奖励 → BT 差分消去 $Z(x)$ → 分类损失。

6.2 梯度解剖与隐式奖励

对 (6.3) 直接求导（记 $h_\theta(x,y)=\beta\log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$）：

$$\nabla_\theta\mathcal{L}_{\text{DPO}}=-\mathbb{E}\Big[\underbrace{\sigma\big(\hat{r}_l-\hat{r}_w\big)}_{\text{样本权重}}\;\beta\Big(\nabla_\theta\log\pi_\theta(y_w\mid x)-\nabla_\theta\log\pi_\theta(y_l\mid x)\Big)\Big],\tag{6.4}$$

其中隐式奖励 $\hat{r}(x,y)=h_\theta(x,y)$（丢弃常数 Z）。三个观察：

1. **梯度形态与 RLHF 一致**：提高偏好回答、降低非偏好回答的似然，按 token 级 $\nabla\log\pi$ 作用——与 PPO 作用于 (x,y_w,y_l) 的期望梯度同构，但完全离线、无需采样。
2. **自动 margin**：$\sigma(\hat{r}_l-\hat{r}_w)$ 在模型已经“分对”时趋零，已分对的样本不再推挤——与 RM 训练中的 $(1-\sigma(\cdot))$ 同一机制。
3. **隐式 RM 免费赠送**：训练好的 π_θ 可以在任何 (x,y) 上计算 $\hat{r}=\beta\log\frac{\pi_\theta}{\pi_{\text{ref}}}$ 当打分器用——这正是 rejection sampling、BoN 重排与迭代 DPO 的“RM 替身”。

6.3 为什么 DPO 可能出问题：理论与坑

DPO 的三大坑（按实践中出现频率排序）

1. 对（弱）非偏好回答的挤压过猛。 (6.4) 对 y_l 的下拉没有下限：低质量 y_l（甚至乱码）被无限压低不损害损失，但会**把概率质量从 y_l 所在的整体区域驱赶出去**，连带压低相邻的合理回答——文献称 *likelihood displacement*。症状：训练损失下降、留出集偏好准确率上升，但实际生成质量**变差**、多样性骤降。对策：IPO/KTO 的有界替代（下节）、非偏好样本只取“稍差”的（同分布生成 y_l，而不是乱码或异源负例）。

2. 离线数据的分布错配。DPO 隐含假设偏好对 $\sim$ 目标部署分布。若 y_w/y_l 全来自其他模型的输出，$\beta \log \frac{\pi_\theta}{\pi_{\text{ref}}}$ 在这些点上的梯度推动的是别人分布上的对数比，本策略自己采样的区域不受控——**迭代式 DPO**（on-policy 采样 y，用 RM/LLM 裁判组对）因此显著更稳。
3. β 的双重身份。β 既是 KL 温度又是梯度步长因子 (6.4)：β 小 $\Rightarrow$ 允许大偏离但每步梯度也小；实践中 DPO 对 β 的敏感度远高于 PPO 对 β 的敏感度，建议以 0.1 为中心网格搜索，并在日志里直接跟踪 $\mathbb{E}[\log \frac{\pi_\theta}{\pi_{\text{ref}}}]$（它就是被优化的“隐式奖励”）。

6.4 变体族谱：IPO、cDPO、SimPO、KTO、ORPO

方法	改动	动机 / 适用
IPO (2023)	以 $\left(h_w - h_l - \frac{1}{2\tau}\right)^2$ 替换 $-\log\sigma$	有界：对数比差被推到 $1/(2\tau)$ 即停，防 likelihood displacement 的理论上干净解
cDPO (2023)	标签平滑：$y \leftarrow (1-\lambda)y + \lambda/2$	容纳标注噪声：把“偏好”当软标签
R-DPO	正则项惩罚 $\lvert h_w\rvert + \lvert h_l\rvert$	抑制隐式奖励无界漂移
SimPO (2024)	去掉 π_{ref}：$r = \frac{\beta}{\lvert y\rvert}\log\pi_\theta(y \mid x)$，比较加 margin γ	参考模型免费：省一次前向、显存减半；需调 $(\beta, \gamma, \gamma_{\text{len}})$
KTO (2024)	单边标签：好/坏样本各自与参考比	不需要成对数据，直接吃点赞点踩；基于前景理论的不对称效用
ORPO (2024)	SFT 损失 + 对 y_l 的 odds-ratio 惩罚	单阶段：SFT 与偏好优化合一，省两段式流水线

表 6.1 DPO 变体族谱（均保持“闭式重参数化 + 分类式损失”的骨架）。

值得单独一提的是 **KTO** 的目标结构——它对每个样本只有一个二元信号 $z \in \{+, -\}$：

$$\mathcal{L}_{\text{KTO}} = \mathbb{E}\Big[\lambda_{\text{D}}\, w(z)\left(1 - \sigma(\beta(k_h - z_{\text{ref}}))\right) \;+\; \lambda_{\text{KD}}\, w(z)\left(1 - \sigma(\beta(z_{\text{ref}} - k_l))\right)\Big], \tag{6.5}$$

其中 $k_h = \log \frac{\pi_\theta(y_+)}{\pi_{\text{ref}}(y_+)}$、$k_l$ 同理，z_{ref} 是 batch 平均的 KL 项（锚点）。不对称权重来自前景理论（损失厌恶 $\lambda_{\text{KD}} > \lambda_{\text{D}}$）。实践中 KTO 在“只有点赞/点踩日志”的场景（几乎所有的产品环境）里是首选。

6.5 迭代式偏好优化：把 DPO 变成在线算法

DPO 的离线缺陷有个直接的修复：**多轮采样**（Zephyr-β、Llama-3、SPIN 系列）：

1. 用当前 π_θ 对 prompt 批采样 n 个回答；
2. 用 RM / 隐式奖励 / LLM 裁判排序，取 (best, worst) 或 (best, 随机) 组对；
3. 一轮 DPO（或 IPO/KTO）更新；
4. 重复。每轮都是 on-policy 数据，likelihood displacement 被自然抑制。

这事实上重建了一个“RM 打分 + 负梯度”的在线 RL 循环，只是把“策略梯度更新”换成了“偏好分类更新”。它工程门槛低（无 rollout 引擎也能跑）、稳定性好，成为 2024–2026 年开源社区对齐的主力配方之一；代价是对采样多样性的要求高（温度、top-p 需调），且收敛上限仍弱于 PPO/GRPO 的直接 RL。

第 6 章要点

- DPO 不是启发式而是**精确换元**：RLHF 目标闭式解 $\Rightarrow$ 隐式奖励 $\Rightarrow$ BT 差分消 Z。β 身兼 KL 温度与梯度步长。
- 同一骨架下，改“如何比较”就得到整个变体族：有界比较（IPO）、去参考（SimPO）、单边标签（KTO）、单阶段（ORPO）。
- 实践三件套：on-policy 组对、非偏好样本同分布、跟踪隐式奖励的分布；离线单轮 DPO 在强基座上容易“损失降、质量降”。

第七章 GRPO 与组相对方法：删除 critic

PPO 在 LLM 上的最大开销不是策略更新，而是 **critic**：一个与策略同尺寸、需要持续训练的第二网络。DeepSeekMath（2024）提出 GRPO（Group Relative Policy Optimization）：同一个提示采样 G 个回答，用**组内统计量**构造基线，从而把 critic 整个删掉。这个看似简单的改动把 RLVR 的工程门槛降了一个数量级，直接铺平了 DeepSeek-R1 与 2025 年开源推理模型浪潮的道路。本章推导它的数学性质、剖析它的两个系统性偏置（长度与难度），以及 2025 年两个修正（Dr. GRPO 与 DAPO）。

7.1 从 PPO 到 GRPO：目标函数

对每个提示 x_i，采样一组回答 $\{y_{i,1}, \ldots, y_{i,G}\} \sim \pi_{\text{old}}(\cdot \mid x_i)$，各自得奖励 $\{r_{i,1}, \ldots, r_{i,G}\}$。GRPO 的优势是组内标准化：

$$\hat{A}_{i,j} = \frac{r_{i,j} - \text{mean}(\{r_{i,g}\}_{g=1}^{G})}{\text{std}(\{r_{i,g}\}_{g=1}^{G})}, \qquad \hat{A}_{i,j,t} \equiv \hat{A}_{i,j} \quad \text{（广播到}y_{i,j}\text{ 的每个 token）}, \tag{7.1}$$

策略损失沿用 PPO 的 clipped 形式（外加 KL 正则，与第 五 章相同）：

$$\mathcal{L}(\theta) = -\,\mathbb{E}_{i,j,t}\Big[\min\big(\rho_{i,j,t}\,\hat{A}_{i,j},\ \text{clip}(\rho_{i,j,t}, 1{-}\epsilon, 1{+}\epsilon)\,\hat{A}_{i,j}\big)\Big] \;+\; \beta\,\mathbb{E}_{i,j,t}\big[\text{KL}_t\big]. \tag{7.2}$$

直觉：**“比同题的其他尝试好多少”就是优势**。不需要任何价值函数，基线的统计功效来自 G 次重复采样。

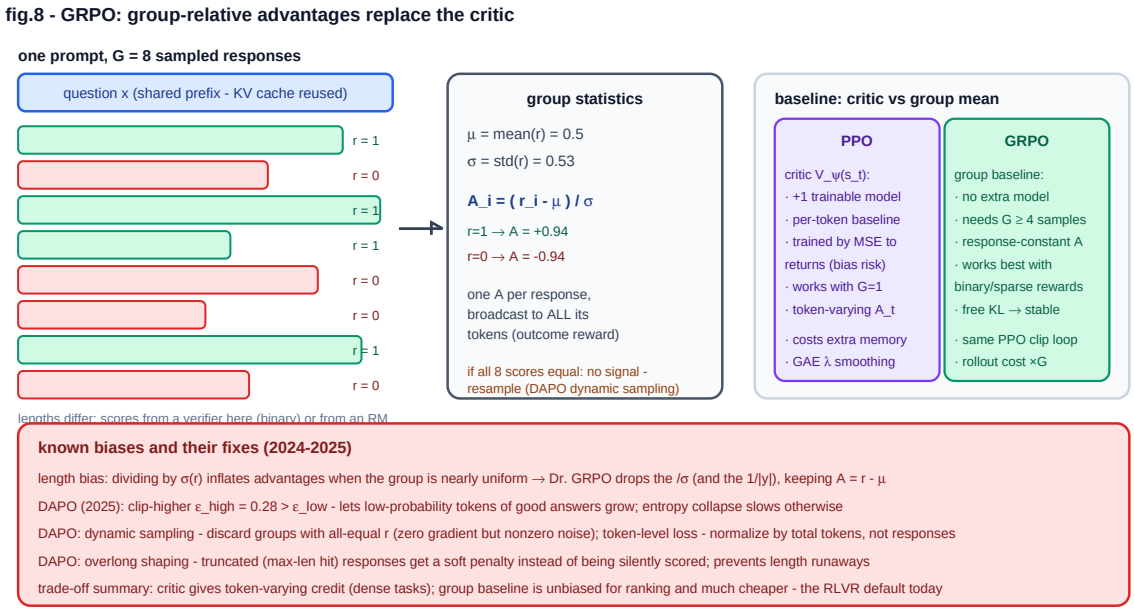


图 7.1 GRPO：一个提示的 G 个采样构成对照组；组均值（可选除以标准差）提供基线，优势广播到整条回答。右栏对比 PPO 的 critic 基线。

7.2 组基线的统计性质

RLOO 的无偏性与 GRPO 的微小偏差

留一法（RLOO）基线。对样本 j，用除 j 外的组均值做基线：

$$b_{-j} = \frac{1}{G-1}\sum_{g\neq j} r_g \qquad \Longrightarrow \qquad \hat{A}_j^{\mathrm{RLOO}} = r_j - b_{-j}.$$

无偏性验证：r_j 与 b_{-j} 独立（j 不参与自己的基线），$\mathbb{E}[\hat{A}_j^{\mathrm{RLOO}}] = \mathbb{E}[r_j] - \mathbb{E}[r]$，且

$\mathbb{E}\big[r_j - b_{-j}\big] = \mu - \mu = 0$（中心化），　$\mathrm{Var} = \mathrm{Var}(r)\Big(1+\frac{1}{G-1}\Big)$—方差只比理论下界多 $\frac{1}{G-1}$ 倍.

GRPO 基线（组均值含自身）则 $r_j - \mathrm{mean}(r)$ 与 r_j 负相关（j 的分值把自己往下拉），产生**轻度的均值收缩**：$\mathbb{E}\big[r_j - \mathrm{mean}(r)\big] = (1-\frac{1}{G})(\mu_j - \mu)$，即优势被因子 $(1-1/G)$ 缩小——通常无害（相当于学习率打折），但配合下述 std 归一化时会与难度分布交互产生系统偏置。**除以 std 的动机：**二元奖励下 $\mathrm{std}(r) = \sqrt{p(1-p)}$（$p$ 为该题通过率），难度两端（$p \to 0, 1$）$\mathrm{std} \to 0$，除以 std 会**放大简单/难题的梯度**——这是 Dr. GRPO 指出的第一个偏置。■

7.3 长度偏置的解剖与 Dr. GRPO

GRPO 原始实现的第二个偏置更隐蔽：损失按**序列均值**归一（每条回答内 token 平均），而 KL 项按 **token 求和**。两者不对称：长回答的每个 token 拿到更小的策略梯度、却缴纳同样的 KL 税。等效后果是**隐式长度偏好**：模型倾向于在长回答里“藏进”低优势 token，在短回答里回避高风险表述——与“更长的思考”混在一起后极难归因。Dr. GRPO（2025）的修正极其朴素：

$$\hat{A}_{i,j} = r_{i,j} - \mathrm{mean}(r_{i,\cdot}) \quad \text{(不除 std)}, \qquad \text{损失按全体 token 求和} \quad \text{(不做序列内均值)}. \tag{7.3}$$

其论文用同一预算复跑 R1-Zero 式训练：AIME 分数相当甚至略升、**平均回答长度下降约 40%**——直接证明原始 GRPO 的长度增长中有相当部分是归一化伪影而非“思考需求”。

7.4 DAPO：四项工程修正

DAPO（2025，字节跳动）是目前 RLVR 配方中工程细节最扎实的一版，四项修正各自对应一类失败模式：

1. **Clip-higher（非对称裁剪）。**$\epsilon_{\mathrm{high}} = 0.28 > \epsilon_{\mathrm{low}} = 0.2$：允许低概率 token 的概率比率向上走得更远。动机：预训练模型里许多“正确推理的关键措辞”初始概率极低，对称 clip 会把它们的增长压死——熵过早坍缩、多样性枯竭。非对称 clip 单侧放松后：熵曲线平稳下降而非跳崖。

2. **动态采样。**丢弃全对/全错组（$\hat{A} \equiv 0$，零梯度纯噪声），用新样本补足 batch。实现上等价于一个**难度过滤器**：把训练分布动态集中在“当前策略 30–70% 通过率”的题目上——课程学习的自动化版本。

3. **Token 级损失归一。**全局 token 均值替代序列均值，与 Dr. GRPO 的修正同源（消除长度偏置）。

4. **超长整形奖励**。对撞到 `max_new_tokens` 被截断的回答，给一个软惩罚项（如 $r \leftarrow r - \eta$ 或专门的“截断负奖励”），而不是悄悄当作普通样本。否则模型学到“写长无妨”，而截断样本里藏着大量未被结算的半成品推理。

GRPO 家谱速查

- **RLOO / ReMax：**与 GRPO 同思想的极简版——组均值（或贪婪基线）+ REINFORCE，无 clip、无内层 epoch。实现 50 行，适合验证管线正确性。
- **GRPO：** = RLOO + PPO 的 clip 信任域 + KL 正则 + std 归一。
- **Dr. GRPO：**去掉 std 与序列均值（去偏置）。
- **DAPO：**clip-higher + 动态采样 + token 级归一 + 超长整形（工程完整版）。全部共享同一主干：**组内比较产生优势，clip 限制步长**。

7.5 实践配置基线

参数	典型值	说明
组大小 G	8–16	太小则组统计噪声大；太大则单题采样成本高。rollout 预算固定时与 batch size 权衡
内层 epoch K	1	RLVR 场景样本新鲜度远比复用重要；$K > 2$ 几乎总有 off-policy 噪声
ϵ	0.2 / 高侧 0.28	DAPO 非对称裁剪
KL 系数 β	0.001–0.04	RLVR 常直接把 per-token KL 目标压在 ~ 0.01 nats；过大阻断探索
γ（折扣）	1	终端奖励场景无意义，多数实现根本不实现折扣
batch（提示数）	32–512	与 G 相乘得到有效 rollout 数；决定信噪比
lr	1×10^{-6} 级	比 SFT 低 5–10×；RL 梯度噪声大，靠小步长积累
温度	0.7–1.0	采样温度是算法的一部分：低 $\Rightarrow$ 熵坍缩，高 $\Rightarrow$ 奖励信号稀释

表 7.1 GRPO/DAPO 的现代默认配置（综合 veRL/OpenRLHF 配方与公开论文）。

第 7 章要点

- GRPO 用“同题对照”替代 critic：均值（RLOO 更纯净）做基线、std 做量纲（Dr. GRPO 建议删去）。统计上它把方差从“跨题差异”剥离，只留下“同题内的好坏差异”。
- 两个系统性偏置——std 放大极端难度、序列均值压长回答——分别由 Dr. GRPO 与 DAPO 修正；修正后长度增长 $\sim 40\%$ 是伪影。
- clip-higher 与动态采样这两项“工程小改”对可训性影响巨大：前者保住探索，后者自动构造课程。任何 RLVR 实现都应默认带上。

第八章 RLVR 与推理强化：R1 的完整故事

可验证奖励强化学习（RL with Verifiable Rewards）把第 四 章“代理必被攻破”的困境绕开了：当任务本身可以**机器判定**（数学答案对错、代码测试通过与否），奖励就是规则而非模型，reward hacking 在构造层面被消除。2025 年的 DeepSeek-R1 与 OpenAI o1 分别从开源与闭源两侧验证了同一件事：**在足够的 RL 步数下，长程推理行为可以从纯结果奖励中自发涌现**。本章拆解它的奖励设计、训练配方与涌现机制。

8.1 奖励设计：RLVR 的“法律条文”

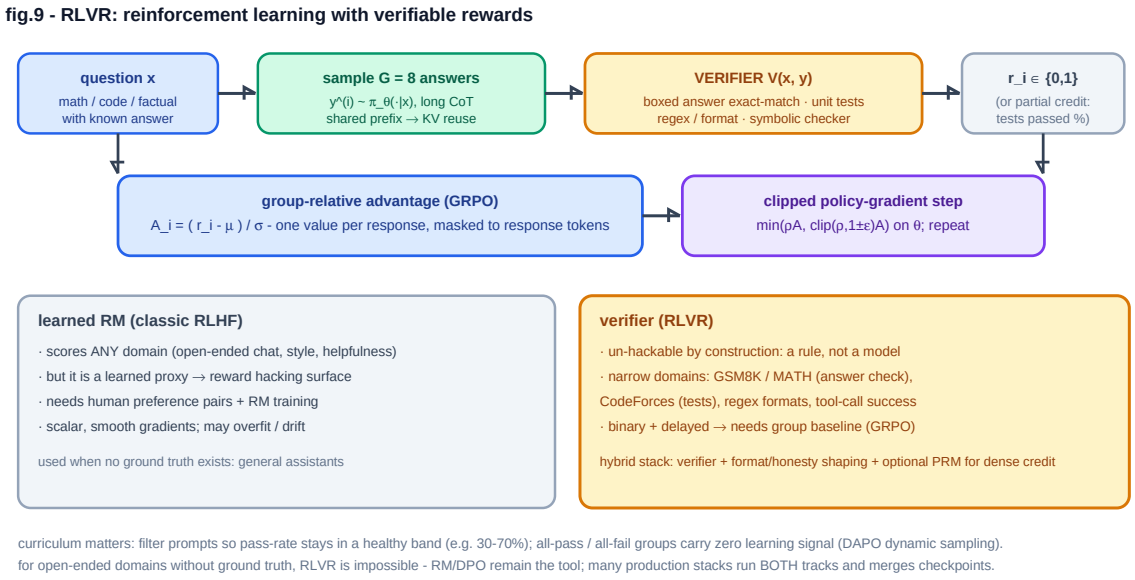


图 8.1 RLVR 主循环与两条反馈路线（学习型 RM vs 规则验证器）的对比。

R1 报告公开的奖励构成值得逐条读（它是目前最完整的一手配方）：

- **数学正确性**：最终答案须放入约定的 `boxed{}` 定界符，字符串比对。规则简单到“没有 RM 可黑”。
- **代码正确性**：生成测试用例 + 沙箱执行（或在线判题）。奖励 = 测试通过。
- **格式奖励**（早期阶段）：把思考放进 `<think>` 标签的格式分。这是唯一的“软”奖励，且在行为稳定后被移除。
- **语言一致性奖励**（R1 阶段二）：惩罚中英夹杂（按目标语言字符占比）。
- **明确不做的事**：不给过程奖励（避免 PRM 被 hack 的拟合面）；不给长度奖励（防“写长换分”）；不给 AI 裁判分（同 RM 风险）。

设计哲学：奖励的“最小可判定集”

RLVR 的奖励越少越好：每加一条软奖励就加一个被优化的代理面。R1 的选择是——正确性（硬）+ 一两条塑造行为的格式/语言分（软、限时）。这条纪律与第 四 章的 Goodhart 分析完全一致：硬奖励是规则的（不可攻破），软奖励是临时的（不需要长期可靠）。

8.2 R1-Zero：纯 RL 的涌现现象

R1-Zero 直接在基座上（跳过 SFT）对数学/代码提示做 GRPO-RLVR，模板只加一句“*Let's think step by step*”，并要求把最终答案放进约定的 `boxed{}` 标记。训练数千步后报告了四类**自发涌现**的行为：

1. **反思（aha moment）**：“*Wait. Wait. That's an error in my derivation... let me reconsider*”——没有任何数据教过这个模式，奖励也从不为“反思”付费，它只是提高了答对率从而被强化。
2. **回溯与验证**：解到中途主动检查中间结果，错则推倒重算。
3. **长度自发增长**：回答从数百 token 增长到上万 token——注意结合第 七 章：其中约四成是 GRPO 归一化伪影（Dr. GRPO 复测），剩余部分是真实的“更多计算换精度”。
4. **泛化**：训练只有数学/代码，但长思维链一类的推理模式在开放式任务上也有迁移迹象（仍有争议）。

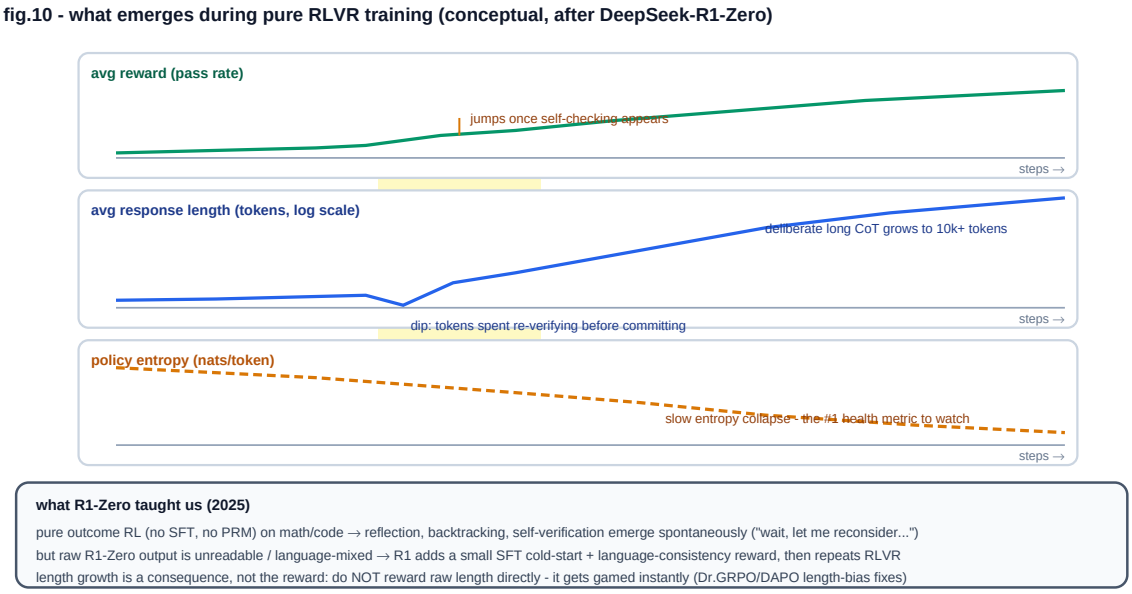


图 8.2 RLVR 训练动态的概念图（据 R1 报告与后续复现综合）：奖励上台阶、长度先降后升、熵持续衰减。三个量是每一次 RLVR run 的“心电图”。

为什么纯 RL 能“长出”推理？ 机制上有三层解释，层层递进：（i）**素材论**：预训练语料含海量“验算、检查、重写”的模式（程序员都在这么做），RL 只是重新加权采样分布，把这些休眠模式串成策略——它不创造行为，只“挖开河道”（呼应第 一 章的能力休眠论）。（ii）**搜索论**：一条长 CoT 是一次串行搜索（试错、回溯、换法），RLVR 通过奖励差分告诉模型“哪些搜索动作期望回报高”——等价于训练了一个搜索策略而非一个答案表。（iii）**计算即精度论**：在验证器可判定的域，精度(T) 关于思考预算 T 单调（更多采样/更长推理 $\Rightarrow$ 更高的通过率，BoN 数学见第 九 章），RL 只需把策略推向“用满预算”的一侧。

8.3 R1 的完整流水线：四阶段

R1-Zero 的产物“能力强但难读”（语言混杂、格式混乱、促销式长段落）。R1 最终版用一条四阶段流水线修复，它已成为 2025–26 年推理模型的事实标准：

1. **Cold-start SFT**：数千条长 CoT 示范（部分由 R1-Zero 自身 rejection-sampling 后人工修整）——安装可读的格式与语言；
2. **RLVR 第一轮**：推理对齐（数学/代码 + 格式/语言软奖励）；
3. **Rejection-sampling SFT**：用当前模型采样大量回答，**只保留验证通过的**，混合通用 SFT 数据再训一轮——把 RL 学到的推理行为固化进监督分布，并修复非推理域的退化；
4. **RLVR 第二轮**：全任务混合（推理 + 通用指令 + 安全）的最终 RL。

四阶段流水线的内在逻辑

每轮 RL 与 SFT 交替都在做同一件事的互补两面：RL 负责**探索**（发现新的高分行为），SFT/采样回灌负责**固化**（把发现变成稳定分布）。纯 RL 会把行为挤到窄模式上（熵坍缩），纯 SFT 无法发现新行为——交替执行既保探索又保可读性。这与进化算法的“变异–选择–遗传”循环同构。

8.4 课程、难度与健康监控

RLVR 对**题目难度分布**极敏感，且这个分布随训练动态漂移：

- 全对组（通过率 $p \approx 1$）与全错组（$p \approx 0$）的优势恒为零，**零学习信号、纯方差**。理想区间是 $p \in [0.3, 0.7]$。
- 实现“难度过滤”的三种方式：（a）离线：先用当前模型扫题库，剔除通过率 $< 10\%/> 90\%$ 的题；（b）在线：DAPO 动态采样——全同组直接丢弃重采；（c）半在线：按近期通过率滑动调整采样权重（真正的自组织课程）。
- 随训练推进，题库整体变“简单”，需要**持续补题**（合成难题、上探难度）——这是 RLVR 数据工程与 SFT 数据工程的最大区别。

8.5 熵：RLVR 的关键存量

训练全程需要盯住的第一指标是**策略熵** $\mathcal{H} = \mathbb{E}_t\big[-\sum_v \pi_\theta(v \mid s_t) \log \pi_\theta(v \mid s_t)\big]$：

- 健康轨迹：$\mathcal{H}$ 平稳缓慢下降（推理风格收敛但仍保留方法多样性）；
- 病危轨迹：$\mathcal{H}$ 塌崖式下降 $\Rightarrow$ 策略锁死在单一模板，后续奖励增长停滞（“学死了”）。此时新题不再产生新策略，只产生复述。
- 维持手段（按副作用从小到大）：提高采样温度；DAPO clip-higher；熵正则（损失中显式加 $+\eta\mathcal{H}$ 项）；降低 β（放松 KL 对参考的锚定——注意权衡能力漂移风险）；重置参考模型（把 π_{ref} 更新为当前策略，等价“重启探索时钟”，有争议）。

8.6 过程奖励的在线化：PRIME

PRIME（2025）展示了介于 ORM 与静态 PRM 之间的第三条路：**隐式过程奖励**。利用 DPO 同款换元（第 六 章）：

$$\hat{r}(x, y_{\le t}) \;=\; \beta \log \frac{\pi_{\mathrm{PRM}}(y_{\le t} \mid x)}{\pi_{\mathrm{SFT}}(y_{\le t} \mid x)}, \tag{8.1}$$

把一个“步骤价值模型”参数化为两个策略的对数比，用 Math-Shepherd 式 MC 标签在线训练它，再以 $\hat{r}_t$ 作为逐步奖励做策略梯度。效果：信用分配密度介于 ORM（1 次/序列）与人工 PRM（1 次/步）之间，且标注完全自动化。这代表了“过程奖励”方向的当前工程共识：**标签可以用策略自己生成，但要与策略共同进化防止过拟合。**

第 8 章要点

- RLVR 的奖励设计纪律：**最少、最硬、最可判定**；软奖励（格式/语言）限时存在，硬奖励（对错）永不妥协。
- R1 的四阶段（cold-SFT → RL → 拒采 SFT → RL）是“探索–固化”交替的模板；涌现推理不来自数据标注，而来自结果奖励对预训练已有模式的重新加权。
- 难度分布是被训练动态消费的资源：动态采样/补题是必需品；熵是唯一需要全程守护的存量指标。

第九章 测试时计算扩展：用推理 FLOPs 换精度

预训练时代的能力公式里只有一个变量：训练算力。2024 年之后，o1 与 R1 证明了第二个变量至少同等重要：**测试时算力**——同一个模型，允许它生成更长的思维链、采样更多的候选、做更多的自我验证，精度可以沿一条新的扩展曲线继续上升。本章给出三种花测试时算力的方式的精确刻画（并行采样、序列搜索、预算控制），并说明它们与第 八 章的 RLVR 训练如何互为表里。

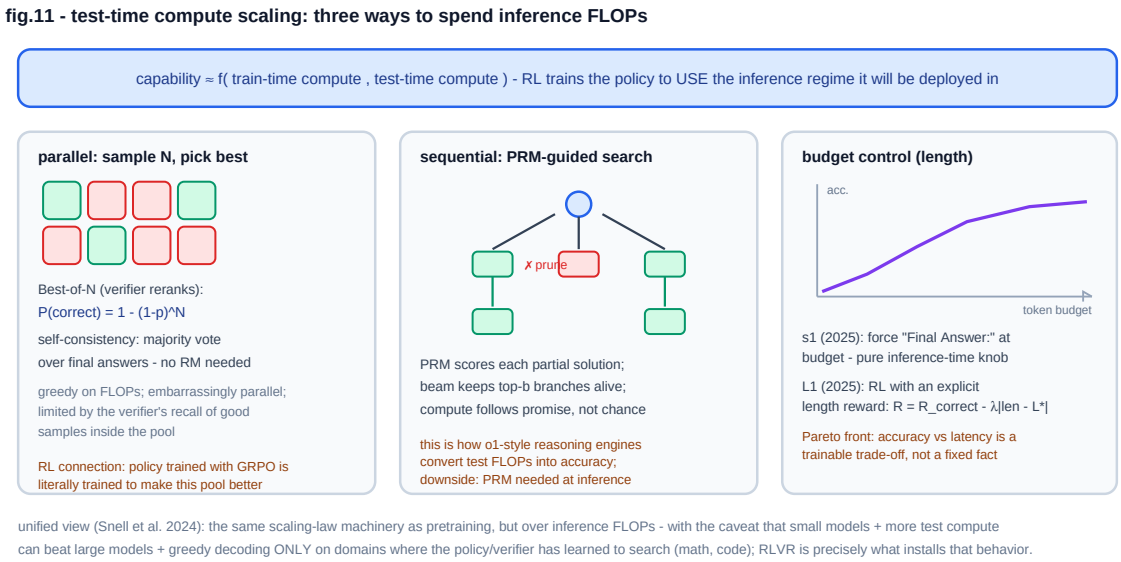


图 9.1 测试时计算的三种形态：并行采样后择优（左）、过程奖励引导的序列搜索（中）、显式的长度/预算控制（右）。RLVR 的作用是训练模型在这三个维度上“会花钱”。

9.1 统一视角：第二条扩展轴

把一次问答的总算力分解为

$$C_{\text{total}} = \underbrace{C_{\text{train}}}_{\text{一次性，摊销}} + \underbrace{N \cdot \mathbb{E}[|y|] \cdot c_{\text{tok}}}_{\text{每次请求的推理 FLOPs}}, \tag{9.1}$$

其中 N 为采样条数、$|y|$ 为单条回答长度、c_{tok} 为每 token 成本。预训练只优化第一项；后训练（尤其 RLVR）优化的是**第二项的效率**：给定同样的测试预算，把 token 换成多少正确率。Snell 等人（2024）的系统实验给出了两条经验规律：

1. **测试时扩展曲线存在且可预测**：在数学等可判定任务上，精度关于测试预算近似幂律上升——形式与预训练 scaling law 同构，但自变量换成了推理 FLOPs。

2. **算力最优分配是模型尺寸与测试预算的函数**：固定总预算时，“小模型 + 大测试预算”可以在一定区间内打败“大模型 + 贪心解码”；最优模型尺寸随预算缓慢增长。前提是验证器/投票机制要足够可靠——这是该路线最大的实际约束。

9.2 并行采样：Best-of-N 与自洽性

Best-of-N 与多数投票的精确数学

覆盖（完美验证器）。设单次采样答对的概率为 p，N 次独立采样中至少有一个正确的概率：

$$P_{\text{cov}}(N) = 1 - (1-p)^N. \tag{9.2}$$

$p = 0.2$ 时 N=16 的覆盖率已达 97%——这就是“模型知道答案但一次说不稳”这一现象的定量刻画。BoN 成功 = **覆盖** ∧ **选中**：

$$P_{\text{BoN}}(N) \;\le\; \min\big(P_{\text{cov}}(N),\; P(\text{验证器把正确样本排第一})\big). \tag{9.3}$$

$N \to \infty$ 时 BoN 收敛到验证器的召回上限，而不是 1：**覆盖随 N 指数改善，选择质量与 N 无关**——这条“两因子分解”是理解 BoN 曲线饱和形状的钥匙。

自洽性（无需验证器）。若答案可以聚成离散选项（数学题的最终数值、选择题），对 N 条回答的最终答案做多数投票。设单次正确概率 $p > \frac{1}{2}$ 且错误答案散布在多个不同取值上，则多数票错误的概率按 Hoeffding 界指数衰减：

$$P(\text{多数票错误}) \;\le\; \exp\!\big(-2N(p-\tfrac{1}{2})^2\big). \tag{9.4}$$

它把“验证器”外包给了**答案分布自身**（正确答案天然是投票吸引子），代价是只适用于答案可比的任务。■

与 RLHF 的对偶。 KL 约束 RLHF 的最优解是 Boltzmann 加权 $\pi^* \propto \pi_{\text{ref}} \cdot e^{r/\beta}$（第 六 章）；BoN 从 π_{ref} 中抽 N 条取奖励最高者，其选择分布是同一加权在 $N \to \infty$、温度 $\to 0$ 极限下的“软 argmax”近似。所以 BoN 可以看作**无梯度的 RLHF 一次近似**——这个视角解释了为什么 RL 训练后的模型 BoN 曲线会整体上移：RL 直接优化了那个“加权”，BoN 只是在它的结果上再取一次最大值。

9.3 序列搜索：过程奖励引导的解码

并行采样在**整条回答**的粒度上花算力；序列搜索把算力花在**步骤**粒度：每生成一个推理步骤，用 PRM（第 四 章）给当前前缀打分，保留分数最高的 b 个束（beam），淘汰其余分支。计算沿“有希望的方向”集中而不是均匀铺开，这是树搜索（MCTS、ToT）与 PRM 的标准组合。

o1/R1 式模型带来一个深刻变化：**把外显搜索内化为串行文本**。回溯、验算、换方法这些原本由搜索框架在外面做的动作，被 RL 直接写进策略本身的 token 流里。两种范式的关系可以精确表述：

- **外显搜索**：策略 π 固定，搜索框架（PRM + beam）在推理时探索 π 支撑的树。算力可插拔，但每步都要跑一次 PRM，部署重。
- **内化搜索**：RL 把“哪些分支值得继续”的判断力蒸馏进 π，采样单条长轨迹即完成搜索。部署轻，但算力预算在训练时就被“定价”。

工程上两者可叠加（在 R1 式模型上再跑 BoN/投票，长 CoT + 多数票是当前竞赛评测的标准配置）。

9.4 预算控制：把长度变成旋钮

测试时扩展引入了一个部署侧的新问题：**算力预算必须可控**（延迟、成本）。两个代表工作给出了相反方向的解法：

1. **纯推理侧：budget forcing（s1，2025）**。到达预算 $L^\star$ 时强制注入结束标记（如“Final Answer:”）迫使收尾。零训练成本，一条模型在 $L^\star$ 上滑动出一条精度–延迟曲线；问题是截断处质量不可控（思考被腰斩）。

2. **训练侧：长度奖励（L1，2025）**。把长度写进 RL 目标：

$$R(y) \;=\; \mathbf{1}[\text{正确}] \;-\; \lambda \max\big(0,\; |y| - L^\star\big), \tag{9.5}$$

即在正确的前提下惩罚超出预算的部分。GRPO/DAPO 框架直接可用，训出的模型在预算内*主动规划*自己的思考长度——实测能在“精度几乎不掉”的前提下把平均长度压掉数倍。

注意 (9.5) 的设计要点：惩罚挂在 $\max(0,\cdot)$ 上、且**正确性优先**（先乘 $\mathbf{1}[\text{正确}]$ 的语义靠分组基线天然保证——答错的短回答不会因为短而得分）。这与第 七 章的长度偏置讨论呼应：长度要么不做目标（防 hack），要么做成*显式约束*（可控），唯独不能让它做隐式代理。

9.5 pass@k 视角：RL 在压缩什么

一个非常有用的诊断量是 **pass@k 与 pass@1 的差距**：

$$\text{gap}(k) \;=\; \text{pass@}k \;-\; \text{pass@1}, \tag{9.6}$$

即“多采样 k 次能额外捞回的正确率”。基座/SFT 模型的 gap 通常很大（采样分布的右上尾藏着大量好答案），而 RLVR 收敛后 gap 显著缩小——**RL 把采样分布的右上尾搬进了主干**。这给了“RLVR 在做什么”一个可测定义：不是增加新知识，而是**重新排列概率质量，使贪心解码（k=1）就能触及过去需要多次采样才碰得到的解**。据此，训练健康的判据是：pass@1 上升的同时 pass@k 不应大幅下降（后者下降意味着多样性被过度收割，是熵坍缩的前兆，见第 十四 章）。

9.6 训练与测试的对偶清单

把两张账本对齐

- 部署时要长 CoT ⇒ 训练数据与 RL 奖励必须**允许**长 CoT （去掉所有隐式的长度惩罚，DAPO 超长整形而非硬截断）。
- 部署时用 BoN/投票 ⇒ 训练时保留足够的采样多样性（熵管理；β 不要过大）。
- 部署时预算受限 ⇒ 用 (9.5) 式长度奖励显式定价，而不是事后截断。
- 采样温度是训练与部署的**共享参数**：RLVR 时的温度应与部署温度一致，否则分布错配会吃掉训练收益。

第 9 章要点

- 测试时算力是一条独立的扩展轴：覆盖率 $1-(1-p)^N$ 指数上升，但被验证器/投票质量截断；序列搜索把算力集中到有希望的分支。

- o1/R1 的本质是把搜索**内化**进策略；外显搜索与内化搜索可叠加使用。
- pass@1 与 pass@k 的差距度量“未被贪心榨干的潜力”，RLVR 的工作就是压缩这个 gap；压过头就是熵坍缩。
- 长度要么不管、要么显式约束（L1 式预算奖励），绝不隐式代理。

第十章 前沿专题：自博弈、智能体 RL、多模态与蒸馏

本章收集四个正在快速演化、但已形成基本共识的方向：不依赖人工标注的**自博弈**、面向工具使用与多轮交互的**智能体 RL**、**多模态**后训练、以及与 RL 紧密交织的**蒸馏**。它们共同勾勒出后训练的下一个主战场：把 RL 从“单轮文本 + 标量奖励”推向长交互、多模态、环境反馈的通用形态。

10.1 自博弈与迭代放大

偏好数据贵，这是 RLHF 的永恒瓶颈。自博弈（self-play）路线的核心想法是 **让模型自己生产训练信号**：自己的输出当负例、监督数据当正例，反复迭代。

SPIN：把“区分自己与数据”当作奖励

第 t 轮，训练一个**专家辨识函数**：

$$F_t(y) \;\approx\; \log \frac{p_{\text{data}}(y)}{p_t(y)}, \tag{10.1}$$

用 logistic 分类“数据样本 vs 模型样本”实现（无需显式建模两个分布）。策略更新逼近对 F_t 的最优响应：

$$p_{t+1}(y) \;\propto\; p_t(y)\, e^{\eta F_t(y)} \qquad \Longrightarrow \qquad \text{模型样本被推向“像数据”的区域.} \tag{10.2}$$

不动点：$p_t = p_{\text{data}}$ 时 $F_t \equiv 0$，梯度消失——博弈到达 Nash 均衡，且均衡点**恰是数据分布本身**。结论一目了然：SPIN 免费但它**只能复原监督信息，不能超越数据**；这解释了它与 RLHF（奖励来自人、理论上可超越任何单一数据分布）的本质差异。后续变体的努力方向都是打破这个上限：SPPO 用软最优响应与正则化让迭代更稳定；**self-rewarding LLM**（Meta，2024）让 LLM 自己当裁判给自己的输出打偏好分（judge 能力与生成能力在同一次训练里共同进化）；弱到强泛化（W2S）研究“弱教师 + 强学生”时学生超越教师信号的条件。■

10.2 智能体 RL：多轮、工具与环境反馈

当模型需要**调用工具、跨越多轮**完成任务（写代码跑测试、查资料、操作网页），MDP 的形式要做三处扩展：

1. **轨迹含观测**：$\tau = (x, a_1, o_1, a_2, o_2, \ldots, a_K, o_K)$，其中 a 是模型生成的 token（工具调用、文本），o 是**环境的回复**（执行结果、检索片段、页面截图）。环境是真正随机的，不再是“追加 token”。
2. **奖励来自环境**：全部测试通过、任务终态校验成功、甚至中间的工具调用成功信号（可作为 shaping）。这其实是 RLVR 的自然延伸——验证器从“比对答案”升级为“执行任务”。
3. **梯度只流过模型 token**：o 部分不进策略梯度、不进 KL 统计，与提示掩码同理，但实现上更容易漏。

多轮 GRPO（agent-RL 主循环）

1. 取任务批 x；对每个任务采样 G 条**整条轨迹**（rollout 引擎必须支持工具调用循环：生成 → 执行 → 拼接 o → 续生成）；
2. 终态结算奖励 r_i（可加逐步工具成功的 shaping 项）；
3. 组内标准化得 $\hat{A}_i$，广播到该轨迹所有**模型** token；
4. clipped 策略梯度更新（与第 七 章完全相同的损失）；
5. 难点全在工程：环境并发与沙箱吞吐、轨迹超长、失败重试的记法。

挑战	表现	现行对策
轨迹超长	一条轨迹数万 token，rollout 长尾剧增	分段 rollout/续写缓存；截断策略显式化
奖励稀疏	几十轮才一个终态信号	工具成功 shaping；PRM 式中间打分
环境吞吐	沙箱/检索比模型慢	环境与引擎解耦、异步化（第 十一 章 D 范式）
跨轮信用	前几轮的坏决策很久后才暴露	轨迹级 GRPO 信号足够；进阶用 critic/PRM 归因
错误传播	一轮工具失败带偏全轨迹	环境重试指令训练；失败案例回流数据

表 10.1 智能体 RL 的五大工程难题与现行做法。

10.3 多模态后训练

VLM（视觉–语言模型）的后训练在数学上与文本完全同构——偏好对 (x, y_w, y_l) 中的 x 换成“图像/视频 + 指令”即可，RM、DPO、GRPO 的一切推导原样成立。差异集中在**奖励来源**与**特有 hack 面**：

- **视觉 RLVR**：图表问答（数值可校验）、OCR 任务（转录文本可比对）、**GUI 智能体**（点击坐标是否达成任务目标可自动判定）——这几个域因为“可机器判定”已成为多模态 RL 的主战场。
- **视觉特有的 hacking**：RM 偏好“图里有文字”式回答；视觉裁判对图像位置/分辨率敏感；GUI 奖励被“刷新到成功页面”绕过。对策与文本同理：规则化验证、对抗案例挖掘、多裁判集成。
- **数据细节**：比较型 RM 对 (x, y_w, y_l) 的**顺序**与图像裁剪敏感，交换增广与几何增广是标配。

10.4 蒸馏与后训练的交互

推理模型的兴起让“蒸馏”重新成为后训练的核心组件。两种基本形态：

1. **序列蒸馏（SeqKD）**：用教师（如 R1）生成带长 CoT 的回答，对学生做普通 SFT（前向 KL，$\max_\theta \mathbb{E}_{y\sim\text{teacher}} \log \pi_\theta(y)$）。简单稳定，学生的上限是教师的采样分布。
2. **拒采自蒸馏**：学生（或教师）对 prompt 采样 n 条，**验证器筛掉错的**，把幸存者作为 SFT 数据——R1 流水线第三阶段（第 八 章）就是这个形态，它是“RL 发现行为 → SFT 固化行为”的桥梁。

R1 论文给出的关键实证：**小模型从大推理模型蒸馏，优于对小模型直接跑 RLVR**（1.5B–7B 级别上普遍成立）——推理模式作为“行为”可以高效复制，而小模型自身的探索存量不足以在 RL 中重新发现它们；但当模型规模足够大（30B+）时，直接 RL 的上限反超蒸馏（教师即天花板）。实践共识的配方因此是：**小模型走蒸馏、大模型走 RLVR、中等规模先蒸馏再 RLVR**（蒸馏装上行为模式，RL 再做自我一致化）。

10.5 配方组合空间

把第 一 章的全景图折叠成一张决策表（结合图 15.1）：

流水线	算力量级	擅长	代表
仅 SFT	单卡–数卡日	格式、任务理解、领域注入	各领域垂直模型
SFT + 单轮 DPO	数卡日	风格对齐、拒答、语气	Zephyr 系配方
SFT + 迭代 DPO（on-policy 组对）	数十卡日	开源条件下的通用对齐	Llama-3 Instruct 配方
SFT + RLVR（GRPO/DAPO）	百卡日–千卡日	数学/代码能力跃升	R1、Qwen-3、各种 32B 推理模型
SFT + RM-PPO（完整 RLHF）	百卡日	开放域人工偏好对齐	InstructGPT/GPT-4 系
多阶段混合（R1 四阶段式）	千卡时级	兼顾推理、通用、安全	R1 最终版

表 10.2 后训练配方谱系（算力为 30–70B 级模型的量级估计）。

第 10 章要点

- 自博弈免费但上限是数据分布本身；要突破上限需要引入模型之外的信号（验证器、环境、人类）。
- 智能体 RL = GRPO + 真环境：轨迹含观测、奖励来自执行、梯度只过模型 token；难点全在系统（第 十一 章异步范式）。
- 蒸馏与 RL 的分工：蒸馏复制行为模式（快、有上限），RL 自我发现并一致化（慢、上限高）；小蒸大训是当前最优折中。

第十一章 训练框架与系统架构

RLHF/RLVR 的算法在纸面上只有几行公式，落地却要同时驱动 **四个模型**（actor、critic、reference、reward）与**一个推理引擎**，跨越训练态与推理态两种截然不同的执行模式。系统设计的全部命题可以压缩成一句话：**让“采样”与“更新”这两种负载在 GPU 上高效共存**。本章剖析四种主流架构范式、rollout 引擎的关键机制、权重同步的三条路径，并给出框架选型对照。

11.1 为什么 RLHF 系统难

三个结构性原因，决定了它比预训练系统工程复杂得多：

1. **负载异质**：一次迭代中，生成（自回归、内存带宽型）与训练（大矩阵乘、计算型）交替出现，各自的最优并行策略完全不同（引擎要 TP/continuous batching，训练要 FSDP/Megatron 3D）。
2. **长尾采样**：生成长度由策略决定，一个 32k token 的“思考”可以拖住整个 batch（图 13.1，另见第 十三 章）。
3. **权重一致性**：引擎里的权重必须每步追上 actor，同步路径的选择直接决定可扩展规模。

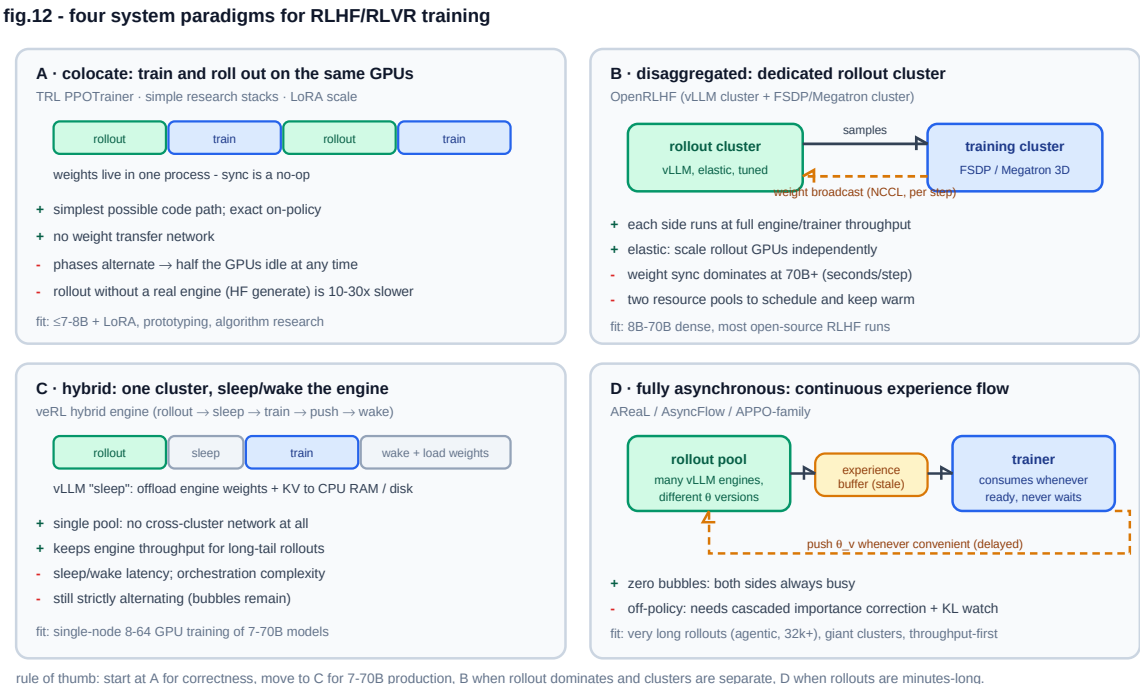


图 11.1　四种系统范式。A（colocate）与 C（hybrid sleep/wake）共享一个集群，B（disaggregated）拆成两个集群、D（async）彻底解耦生产与消费。选择依据见 11.5 节。

11.2 四种范式

A. Colocate（同置）。　训练与生成在**同一组 GPU** 上交替执行。权重天然同处一个进程组，同步近零成本；TRL 的 `PPOTrainer` 是典型实现。代价：两阶段互相等待，GPU 利用率腰斩；生

成通常还要退回 HF `generate`（无 continuous batching），吞吐比专业引擎低 10–30×。定位：算法研究、7B 级、LoRA。

B. Disaggregated（分离）。 两个独立集群：rollout 集群（vLLM/SGLang，可弹性伸缩）与训练集群（FSDP/Megatron），每步通过**权重广播**对齐。OpenRLHF 是代表。优点是两边都以最优形态运行、rollout 可以按长度动态扩容；代价是同步网络的秒级开销（70B 级）与两套资源池的运维。

C. Hybrid（混合引擎，veRL）。 单集群内，引擎平时**睡眠**（权重与 KV cache 卸载到 CPU），训练接管 GPU；训练完推送权重、唤醒引擎。没有跨集群网络，又保住了引擎吞吐；代价是 sleep/wake 的秒级换入换出与调度复杂度。2025 年后单机–单集群（8–64 卡、7–70B）生产训练的事实标准。

D. Fully async（异步）。 rollout 池持续生产经验，训练器随时消费**滞后的**经验（AReaL、AsyncFlow 等）。样本天然 off-policy：生成时的策略版本 θ_v 落后当前 θ 若干步，必须靠重要性采样修正（第 2.5 章）：

$$\rho_t^{(v)} = \frac{\pi_\theta(y_t \mid s_t)}{\pi_{\theta_v}(y_t \mid s_t)}, \tag{11.1}$$

版本跨度 v 越大修正越失真，故异步系统都要监控“平均滞后步数”并截断过旧样本。收益：零气泡（rollout 分钟级的智能体任务下唯一可行范式）；代价：实现与调参复杂度最高。

11.3 Rollout 引擎内幕

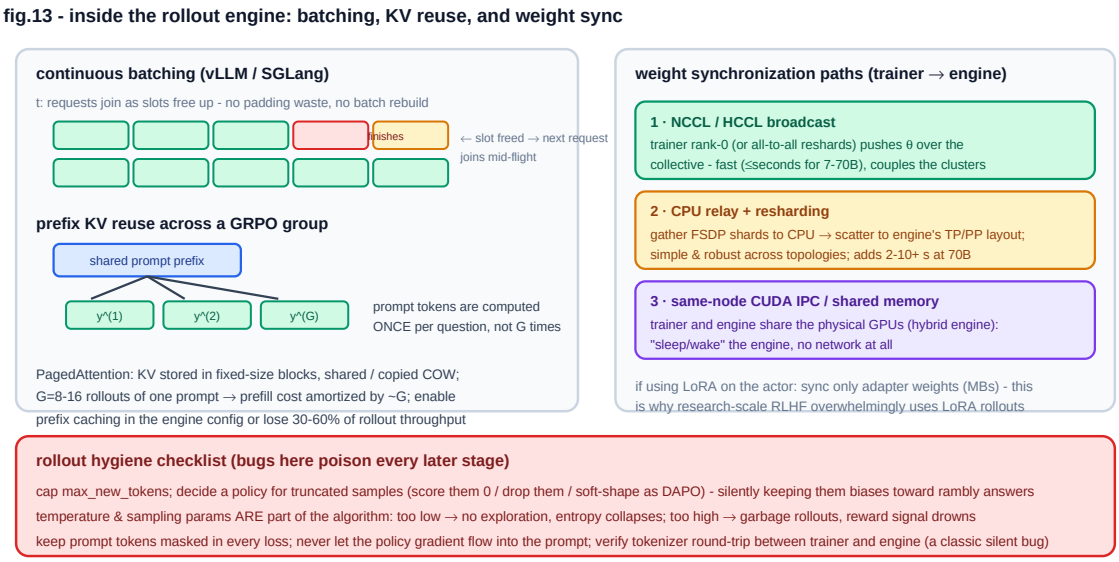


图 11.2 rollout 引擎的三个关键机制：continuous batching、GRPO 组内的前缀 KV 复用、以及三种权重同步路径。

现代引擎（vLLM/SGLang）的三个机制直接决定 RL 吞吐：

1. **Continuous batching：**序列完成即释放槽位、新序列立即加入，batch 结构逐 token 重组。RL 的生成长度方差极大，这一点消除了静态 batch 的填充浪费（可占理论吞吐的一半）。

2. **PagedAttention**：KV cache 按固定块组织，支持共享与写时复制。
3. **Prefix caching**：GRPO 的 G 条采样共享同一提示（以及多轮智能体的历史前缀）——提示的 KV 只算一次，随后的 G 条采样全部命中前缀缓存。**不开此开关，RLVR 的 rollout 吞吐直接损失 30–60%**（提示长、回答长的场景更多）。

11.4 权重同步的三条路径

1. **NCCL/HCCL broadcast**：训练侧 rank 0（或做 all-to-all 重分片）把 θ 推给引擎各 rank。快（7B 约亚秒、70B 秒级），但要求两侧在同一通信域——适合 B 范式的紧耦合部署。
2. **CPU 中继 + 重分片**：训练侧先把 FSDP 分片聚到 CPU，再按引擎的 TP/PP 布局散出去。慢（70B 数秒到数十秒）但拓扑解耦、最鲁棒，是早期 OpenRLHF 的默认路径。
3. **同卡共享（hybrid 睡眠唤醒）**：引擎与训练器共享物理 GPU，权重不过网络，只有 device 间拷贝。**LoRA 特例**：若 actor 只训适配器，同步的是 MB 级的 BA 而非 GB 级的全权重——这是研究规模 RLHF 偏爱 LoRA 的最大系统原因。

11.5 框架选型

框架	范式	强项	适用
TRL	A（colocate）	代码最简、算法全（SFT/DPO/PPO/GRPO）	原型验证、教学、单卡–4 卡
OpenRLHF	B（分离）	Megatron/FSDP 双后端、vLLM 集成早	多节点 8B–70B、经典 RLHF
veRL	C（hybrid）	hybrid engine、Rollout/训练解耦抽象、算法可插拔（含各 RLVR 变体）	2025 年主流生产选型
NeMo-Aligner	B	Megatron 深度整合	大规模 NVIDIA 集群
AReaL / AsyncFlow	D（async）	零气泡、长 rollout	智能体/超长 CoT 研究
SkyRL	C/D	agent 循环一体化	工具使用 RL

表 11.1 主流后训练框架对照（2025–26 快照，各框架能力边界仍在快速变化）。

选型四问

1. 规模：≤8B + LoRA → A；8–70B 全参 → C；多节点要弹性 rollout → B；rollout 分钟级 → D。
2. 算法：只做 DPO/KTO 类离线 → A 永远够；RLVR 长链 → C/D。
3. 团队：有没有专职系统工程师？没有就选 C 的现成实现，别自研 B/D。
4. 网络：跨节点带宽差就避开 B，同步会成为每步的固定税。

第 11 章要点

- RLHF 系统的三个硬骨头：异质负载、长尾生成、权重一致性；一切架构都是围绕“采样与更新如何共享 GPU”的回答。
- hybrid 引擎（sleep/wake）是单集群生产的主流；异步范式用 off-policy 换零气泡，必须配重要性修正与滞后监控。
- 引擎三件套（continuous batching / PagedAttention / prefix cache）决定 RL 吞吐；LoRA 把权重同步从 GB 缩到 MB。

第十二章 工程实践：参考实现、配置与排错

本章把前十一章的公式变成能跑的代码。我们给出一个 **单文件、无框架依赖的 GRPO/DAPO 参考实现**（约 120 行，任何 PyTorch + HF 环境可跑，用于教学与管线验证）、vLLM rollout 的接入点、生产级超参数清单、以及一张“症状 → 病因 → 处方”的排错表。工程章节的全部内容服务于一个信念：**先让正确的小系统转起来，再谈规模。**

12.1 GRPO 的单文件参考实现

```python
# grpo_min.py - a self-contained GRPO/DAPO reference implementation.
# Educational clarity over speed: HF forward passes, no vLLM here.
import torch, torch.nn.functional as F

def per_token_logprobs(model, input_ids, attn, resp_mask):
    """log p(y_t | y_<t) for every response token.
    input_ids : [B, L]  prompt + response concatenated
    resp_mask : [B, L]  1 on response tokens, 0 on prompt/padding
    returns   : [B, L-1] log-probs aligned with targets input_ids[:, 1:]"""
    logits = model(input_ids[:, :-1], attention_mask=attn[:, :-1]).logits
    tgt    = input_ids[:, 1:]
    lp     = torch.log_softmax(logits.float(), dim=-1).gather(-1,
        tgt.unsqueeze(-1)).squeeze(-1)
    return lp * resp_mask[:, 1:]                        # mask prompt tokens!

@torch.no_grad()
def rollout_group(model, tok, prompts, G, temp, max_new):
    """Sample G responses per prompt with the CURRENT policy.
    In production this is replaced by a vLLM/SGLang engine call
    (continuous batching + prefix caching), see Sec 12.2."""
    outs, old_logps = [], []
    for p in prompts * G:
        ids = tok(p, return_tensors="pt").input_ids.cuda()
        gen = model.generate(**ids, do_sample=True, temperature=temp,
                             max_new_tokens=max_new,
                                 return_dict_in_generate=True,
                             output_scores=True)
        outs.append((ids[0], gen.sequences[0][ids.shape[1]:]))
    return outs                                         # list of (prompt_ids,
        resp_ids)

def grpo_loss(lp_new, lp_old, lp_ref, adv, mask,
              eps_lo=0.2, eps_hi=0.28, beta=0.0):
    """DAPO-style clipped objective with asymmetric clip (clip-higher).
    lp_* : [B, T] per-token log-probs (new / at rollout / reference)
    adv  : [B, T] broadcast group-relative advantage
    mask : [B, T] response mask"""
    ratio      = torch.exp(lp_new - lp_old)
    clipped    = torch.clamp(ratio, 1 - eps_lo, 1 + eps_hi)
    pg         = -torch.min(ratio * adv, clipped * adv)
    kl_per_tok = (lp_new - lp_ref).exp() - (lp_new - lp_ref) - 1  # k3 estimator
    loss       = (pg + beta * kl_per_tok) * mask
```

```
    return loss.sum() / mask.sum()                    # token-level normalization

def train_step(actor, ref, tok, batch_prompts, verifier, cfg):
    # 1. rollout ------------------------------------------------------
    samples = rollout_group(actor, tok, batch_prompts, cfg.G,
                            cfg.temp, cfg.max_new)
    # 2. score + group-relative advantage (Dr.GRPO: no /std) ----------
    rewards = []
    for prompt_ids, resp_ids in samples:
        text = tok.decode(resp_ids)
        rewards.append(verifier(text))                # 0/1 or partial credit
    rewards   = torch.tensor(rewards).view(-1, cfg.G)
    adv       = (rewards - rewards.mean(1, keepdim=True))   # [n_prompts, G]
    adv       = adv.repeat_interleave(rewards.shape[1] if False else 1, dim=0)
    # reshape to [n_prompts*G] then broadcast over tokens in the loss
    # 3. assemble tensors ---------------------------------------------
    B, L   = max(len(p) + len(r) for p, r in samples), 0
    ... # pad to common length; build attn/resp_mask; concat prompt+resp
    lp_old = per_token_logprobs(actor, input_ids, attn, resp_mask).detach()
    lp_ref = per_token_logprobs(ref,   input_ids, attn, resp_mask)
    # 4. DAPO dynamic sampling: drop groups with zero advantage ---------
    keep   = (adv.abs() > 0).any(dim=-1)              # per-prompt groups
    # 5. PPO/GRPO inner loop (K epochs, minibatches) --------------------
    for epoch in range(cfg.K):
        for mb in minibatches(input_ids, attn, resp_mask, lp_old, lp_ref, adv_b):
            lp_new = per_token_logprobs(actor, *mb[:3])
            loss   = grpo_loss(lp_new, mb[3], mb[4], mb[5], mb[2],
                               cfg.eps_lo, cfg.eps_hi, cfg.beta)
            loss.backward(); actor.optim.step(); actor.optim.zero_grad()
    # 6. metrics ------------------------------------------------------
    with torch.no_grad():
        ent  = -(lp_new.exp() * lp_new * resp_mask[:, 1:]).sum(-1).mean()
        klt  = (lp_new - lp_ref).masked_fill(~resp_mask.bool(), 0).mean()
        clip = ((lp_new - lp_old).exp() > 1 + cfg.eps_hi).float().mean()
    return dict(loss=loss.item(), reward=rewards.mean().item(),
                entropy=ent.item(), kl=klt.item(), clip_frac=clip.item())
```

三个值得逐行读的实现细节：

1. **移位对齐**：位置 t 的 logits 预测的是 $t{+}1$ 的 token，所以 `logits[:, :-1]` 与 `targets[:, 1:]` 对齐后 gather。这是所有 log-prob 计算的第一坑（差一位时训练“看起来也能跑”，reward 却不涨）。

2. **k3 估计器**：KL 项用 $e^{\Delta} - \Delta - 1$（$\Delta = \log \pi_\theta - \log \pi_{\text{ref}}$）而非裸的 Δ：无偏、恒正、对大 Δ 稳定，是 Schulman 推荐并被各框架采用的形态。

3. **损失归一在 token 级**（`mask.sum()`）而非序列级——第 七 章的长度偏置修正落实处。

12.2 接入 vLLM：生产版 rollout

研究实现与生产的分水岭在 rollout。接入 vLLM 只需要三件事：

```
from vllm import LLM, SamplingParams
llm = LLM(model=path, enable_prefix_caching=True,   # critical for GRPO
          gpu_memory_utilization=0.25, sleep_mode=True)  # hybrid engine
sp  = SamplingParams(n=G, temperature=1.0, top_p=1.0,
                     max_tokens=4096, logprobs=0)
outs = llm.generate(prompt_texts, sp)   # one call -> G samples per prompt
# old log-probs come back with each token (logprobs=0 keeps the chosen one)
# weight sync after each training round:
llm.sleep(level=1)              # offload weights+KV to CPU
trainer.update_actor(...)       # your FSDP/Megatron step
llm.wake_up()                   # reload + push new weights
# or with dedicated rollout cluster: nccl broadcast from trainer rank0
```

配套的三个工程要点：（i）`n=G` 一次生成整组，前缀缓存自动复用提示 KV；（ii）`logprobs=0` 让引擎顺带返回被采样 token 的对数概率，省去训练侧重算 π_{old} 的一整次前向；（iii）tokenizer 的 **round-trip 校验**（encode–decode 完全一致）在训练器与引擎是两套代码时是经典静默 bug 源。

12.3 生产超参数清单

参数	默认/区间	调整逻辑
组大小 G	8–16	方差与吞吐的折衷；验证任务用大 G 换组统计质量
prompt 批	32–512	与 G 相乘得到有效样本；够大才能稳定白化
内层 epoch K	1	RLVR 数据新鲜度优先；>2 引入 off-policy 噪声
lr	5e-7 – 2e-6	全参 7B 从 1e-6 起；KL 涨太快就减半
$\epsilon_{\text{lo}}/\epsilon_{\text{hi}}$	0.2 / 0.28	DAPO 非对称；熵掉太快再加大高侧
β（KL）	0.001–0.04	目标 per-token KL $\approx$ 0.01–0.1 nats；RLVR 常比 RLHF 小
采样温度	0.7–1.0	与部署温度一致；过低熵坍缩
max new tokens	4k–32k	配 DAPO 超长整形；截断样本要有显式策略
梯度裁剪	1.0	常规
advantage 白化	batch 级	关闭后 lr 需大幅重调

表 12.1 RLVR/GRPO 生产基线（综合 veRL/OpenRLHF 配方与 DAPO/Dr.GRPO 论文）。

12.4 排错表：症状 $\rightarrow$ 病因 $\rightarrow$ 处方

症状	最可能病因	处方
ratio 大量越界 / clip 率 $>30\%$	lr 过大或 K 过大，一步走出信任域	lr 减半；$K \rightarrow 1$；查白化是否关闭

reward 平坦不动	组内全同分（题太难/太易）；或 logits 移位错位	难度过滤；单元测试 log-prob 对齐
熵跳崖	温度低 + 对称 clip + 数据同质	温度上调；clip-higher；数据难度分层
KL 超预算线性涨	β 太小或参考模型权重被同步覆盖	加大 β；检查 ref 权重确实冻结
长度暴涨	RM 偏长 / 序列级归一化 / 无截断惩罚	Dr.GRPO 归一；DAPO 超长整形
训练涨、评测崩	reward hacking 或评测污染	held-out 人工盲评；RM 对抗案例
loss 变 NaN	ratio 上溢（fp16）	log-probs 用 fp32；loss 按 token 求和
每步奇慢	rollout 长尾；无 prefix cache	截断策略 + 引擎开关核对

表 12.2 八种高频故障。第一条与最后一条占实际 debugging 时间的一半以上。

12.5 监控面板：六个数看住训练

- **avg reward 与 组内通过率**：学习是否发生；通过率长期 $> 90\%$ 或 $< 10\%$ 说明课程失衡。
- **per-token KL**（对 ref）：分布漂移速度；目标一个 nats 预算。
- **策略熵**：探索存量；平稳缓降为健康，跳崖即预警（第 八 章）。
- **clip 触发率**：步长健康度；健康区间 5–20%。
- **长度分布（分位数）**：均值会骗人，看 p95/p99 的长尾。
- **held-out 正确率**（含人工盲评抽样）：唯一不会说谎的数。

第 12 章要点

- 参考实现里 90% 的正确性在三个细节：logits 移位、提示掩码、token 级归一；先用小模型 + 合成任务让它通过单元测试再上生产。
- vLLM 接入三板斧：prefix caching、`logprobs` 回传、tokenizer round-trip 校验。
- 排错先查两端：clip 率（步长病）与组内方差（信号病）；监控看六个数，held-out 与熵是底线。

第十三章 性能优化：算力配比与算法–系统协同

后训练的“性能”几乎完全等价于**每小时能完成多少有效的 RL 步**（有效 = 组内信号非零的步）。本章建立一步的墙钟模型、给出资源配比的粗算法则、处理异步与样本复用带来的 off-policy 数学、并清点数值稳定的暗坑。图 13.1 是本章的地图。

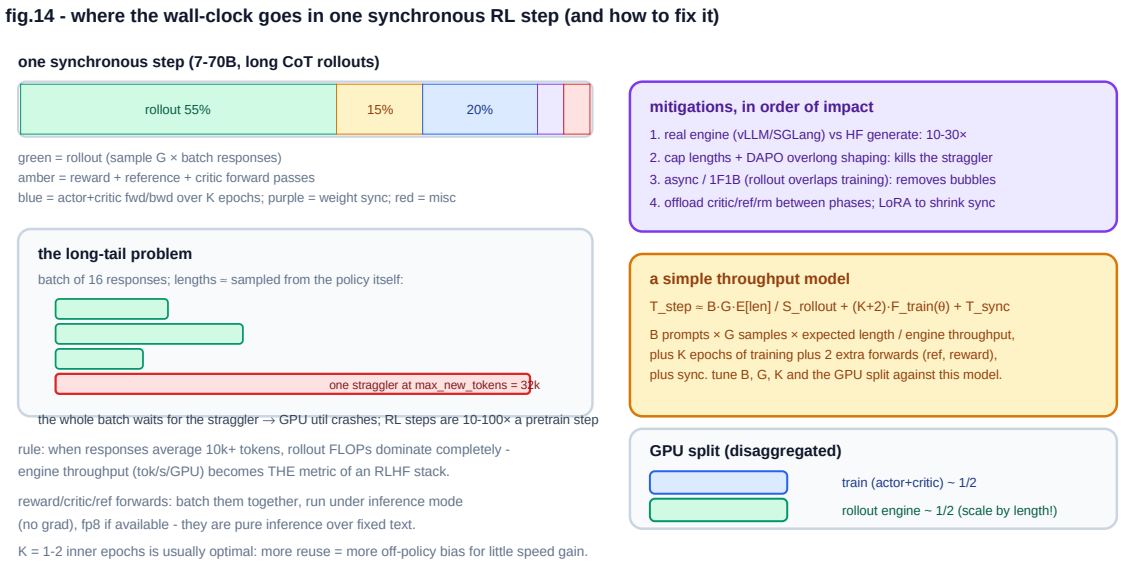


图 13.1 一次同步 RL 步的墙钟构成与治理手段。当平均回答长度上万 token 时，rollout 吞吐成为整个系统的第一瓶颈。

13.1 一步的墙钟模型

设 prompt 批 B、组大小 G、平均长度 $\bar{L}$（token）、引擎吞吐 S_{roll}（tok/s/卡）、训练前向-反向耗时 $F(\theta)$、内层 epoch 数 K、权重同步 T_{sync}：

$$T_{\text{step}} \approx \underbrace{\frac{B\,G\,\bar{L}}{S_{\text{roll}}\,N_{\text{roll}}}}_{\text{rollout（常占 50–90\%）}} + \underbrace{(K+2)\,F(\theta)}_{\text{actor 的 K 次训练 + ref/reward 前向}} + T_{\text{sync}} + T_{\text{misc}}. \tag{13.1}$$

两个立即可用的推论：

- **长链时代 rollout 独大：**$\bar{L}$ 从 1k 涨到 10k，第一项涨十倍而其余几乎不变——RLVR 系统优化的主战场永远是引擎。
- **K 的边际收益快速衰减：**$K{=}1 \to 2$ 让训练项翻倍、却只换来第二次复用的梯度；而复用本身引入 off-policy 偏差（13.3 节）。这就是现代配方普遍取 $K{=}1$ 的定量原因。

长尾的几何。 batch 里最长的样本决定生成阶段的下界。若长度分布 $p(|y|)$ 重尾（RLVR 恰好会把它越推越重），最慢样本 $\approx$ 分布的 $1/B$ 分位数——批越大，等得越久。治理手段按序：**截断并整形**（DAPO 超长惩罚，把截断变成负反馈而不是免费行为）、**分块 rollout**（长样本切到多个 micro-batch）、**弹性引擎池**（B/D 范式按负载扩容）。

13.2 资源配比的粗算

分离式部署下，训练与 rollout 的 GPU 数应使两侧耗时接近：

$$N_{\text{roll}} \;\approx\; N_{\text{train}} \cdot \frac{B\,G\,\bar{L}/S_{\text{roll}}}{(K+2)F(\theta)}, \tag{13.2}$$

即按“两侧单卡耗时之比”配平。经验区间：**短链（$\bar{L} \le 2$k）$\approx$ 1:1；长链（$\ge$8k）可达 2:1–4:1**（rollout 侧多）。另外三项压开销的常规操作：

1. **冻结模型用推理态**：ref/reward/critic 的前向走 `torch.inference_mode` + fp8（若可用），合批一次算完；
2. **offload 与复用显存**：critic 与 reward 换相进驻留；hybrid 引擎的 sleep 把权重沉到 CPU；
3. **micro-batch 自适应**：长样本用小 micro-batch 防显存溢出，短样本合批提吞吐（数据长度分桶）。

13.3 Off-policy 级联与样本复用

样本被复用第 k 次时，采样策略 π_{old} 与当前 π_θ 的距离随更新步数增长，重要性修正项的方差按 exp(2 KL) 放大（第 2.5 章）。异步系统（D 范式）把这个现象变成常态：生成时版本 θ_v 落后当前若干步，**级联修正**即逐层嵌套的比率修正：

$$\rho_t^{(v)} \;=\; \frac{\pi_\theta(y_t \mid s_t)}{\pi_{\theta_v}(y_t \mid s_t)} \;=\; \prod_{j=v}^{\text{now}-1} \underbrace{\frac{\pi_{\theta_{j+1}}(y_t \mid s_t)}{\pi_{\theta_j}(y_t \mid s_t)}}_{\text{相邻检查点间的比率}}, \tag{13.3}$$

实践中不做全连乘（方差不可控），而是：（i）用“生成版本 $\to$ 当前版本”一步修正 + clip；（ii）监控样本的平均滞后步数 v_{lag}，超过阈值（如 4–8 步）的样本直接丢弃；（iii）用经验 KL 作为健康指标：$\widehat{\mathrm{KL}}(\pi_{\theta_v} \| \pi_\theta)$ 应保持在 clip 设计的信任域内（$\lesssim 0.1$ nats/token）。同样的数学限制样本复用：K>1 的内层 epoch 每多一次，序列级累计 KL 大约按 token 数线性堆积——长链场景 K=2 就可能明显失真，这再次解释了 RLVR 的 K=1 惯例。

13.4 数值稳定性清单

RL 训练特有的数值坑（预训练中不会遇到）：

1. **log-softmax 用 fp32**：10^5 词表的 logits 在 fp16 下尾部精度不足，比率 e^{Δ} 会在小数值处抖动放大；
2. **比率按 log 域相减再 exp**：先算 Δ = lp_new − lp_old 再 `exp`，绝不做“两个大 softmax 相除”；
3. **KL 用 k3 形态**（$e^{\Delta} - \Delta - 1$）：恒非负、对大 Δ 不爆；
4. **长序列 loss 聚合用 token 求和**而非逐步平均后平均——防止 fp16 下小值被冲洗（也顺便保住了第 七 章的长度正确性）；
5. **梯度裁剪 + 优势白化**是 RL 的安全带：白化让 lr 与奖励量纲解耦，裁剪在比率爆炸时保住优化器状态。

13.5 算法–系统协同：三组已验证的等价

本章最后给一个“协同设计”的观察：2024–26 年的算法改良与系统改良经常是**同一枚硬币的两面**：

- **删 critic（GRPO）** $\leftrightarrow$ 省一个同尺寸模型的训练与显存，代价 $G\times$ rollout——只有引擎吞吐足够高时才划算；引擎进化（prefix cache、continuous batching）正是让这笔账翻正的技术前提。
- K=1（少复用）$\leftrightarrow$ 对引擎吞吐更高要求；hybrid/异步架构（消除气泡）把“新鲜度换算力”变成可选项。
- **异步 RL** 需要 off-policy 修正 $\leftrightarrow$ 重要性采样 + clip 本来就在算法里——D 范式不是绕过算法约束，而是“用算法已有的机构消化系统想要的自由度”。

第 13 章要点

- 用 (13.1) 建模再配资源：长链时代 rollout 独大，K=1、引擎吞吐、长尾治理是三件最有杠杆的事。
- off-policy 的数学上限是 exp(2 KL)：复用、异步、滞后全部受它管辖，用平均滞后与经验 KL 做护栏。
- 算法与系统的改良互为因果：GRPO 依赖引擎、异步依赖 IS/clip。设计任何一侧时先看另一侧的约束。

第十四章 评测与训练诊断

后训练的评测有两条互相纠缠的线：**产品线**（模型到底变好了没有）与**训练线**（RL 迭代是否健康）。两条线的共同敌人是**测量本身被优化**——当评测分数进入训练目标（直接或间接），它就不再是不偏的尺子。本章给出分层评测设计、训练健康度判读、reward hacking 的排查程序，以及一套“指标组合 → 诊断”的速查表。

14.1 评测集的分层设计

层	内容	测什么	要点
能力（推理）	AIME / MATH-500 / GPQA / LiveCodeBench	训练增益是否迁移	用滚动更新集防污染；报 pass@1 与 pass@k
能力（通用）	MMLU-Pro / IFEval / 多轮任务	非目标域能否守住	RLVR 后重点盯**回退**
对齐与安全	拒答、越狱套件、红队	边界行为	人工校准子集不可省
开放质量	成对比较（GPT/human 判官）	综合助手质量	判官需报位置偏差校验
held-out	从未参与任何选择的冻结集	唯一可信的总分	每轮只看一次，防过拟合

表 14.1 五层评测。前三层可以自动跑，第四层半自动，第五层是制度。

三个协议细节。 （i）**成对比较优于绝对打分**：与基线的 win-rate 对判官噪声更鲁棒（呼应 Bradley–Terry 的比较公理，第 四 章）；（ii）**推理评测必须多次采样**：单次贪心的方差在 RLVR 模型上显著增大（模型学会了“押策略”），报 $n \geq 16$ 的均值与标准差；（iii）**长度与分数一起报**：RLVR 后任何纯精度提升都要折算成“每 token 的精度”看一遍，排除“多写了一倍”的解释。

14.2 训练健康度：五条曲线怎么读

第 十二 章列了六个监控量；这里给出“形态判读”：

- **奖励**：健康形态是带平台的阶梯（新策略被发现 → 固化 → 再发现）；一条光滑斜线反而可疑（多半是 RM 在被过拟合）。
- **熵**：平稳缓降；与奖励同时冻结 = 学死了（探索耗尽，处置见第 八 章熵清单）；熵先降后升 = 模式切换，检查数据新鲜度。
- **per-token KL**：目标带 10^{-2}–10^{-1} nats；线性外冲说明 β 或 lr 失配，立即回滚几个检查点。
- **长度 p95**：暴涨且奖励同涨——先做“奖励去长度化”检查（用长度分层重算奖励增益），确认不是第 七 章的归一化伪影。

- **clip 率**：长期 $> 30\%$ 等于优化器在持续越界，属于 lr/白化问题而非算法问题。

14.3 Reward hacking 排查程序

怀疑被 hack 时按顺序执行（成本递增）：

1. **长度/格式分解**：把奖励增益按长度分桶、按结构特征（列表、代码块、标记词）分解，看增益是否集中在与“实质”无关的特征上；
2. **反事实改写**：把高分回答的“形式”移植到低分内容上（加长、加列表、加客套），若分数大幅上升即坐实 RM 的形式偏置；
3. **held-out 人工盲评**：训练分与盲评分的相关性跌破即定性；
4. **对抗挖掘闭环**：把 hack 样本做成新的偏好对回灌 RM（第 四 章），下一轮用集成 RM 或换验证器。

14.4 诊断速查：指标组合 → 结论

指标组合形态	诊断	首要处置
奖励涨 / held-out 涨 / 熵缓降	健康	无
奖励涨 / held-out 平 / 长度涨	长度或形式 hack	反事实改写；去长度归一
奖励涨 / heldout 跌 / KL 涨	过度优化代理	β 上调；早停；RM 轮换
奖励平 / 熵平 / 通过率两端化	课程失衡	动态采样与补题
奖励抖动巨大 / clip 率高	步长过激	lr 减半；查白化
奖励平 / KL 爆	更新被 clip 全部截断	检查掩码与 logits 对齐 bug
heldout 涨 / 熵跳崖	正在收割多样性	温度/clip-higher；准备早停

表 14.2 组合形态比单指标可靠；先读组合再开药。

14.5 消融的组织方式

工程上还有一条隐性评测线：**改动是否值得保留**。RL 训练天然高方差，消融协议要点：

- 每个配置 ≥ 3 个种子；报告均值 $\pm$ 标准差，差距在种子噪声内的一律视为“无结论”；
- 固定数据顺序与温度（它们与算法超参同等重要，却常被忽视）；
- 训练中期检查点也参与评测：好的改动应表现为**整条曲线**（更快到达同一精度或同时间更高精度），而非单点运气；

- 回归集（regression suite）：上一版本的所有修复用例必须保持绿。

第 14 章要点

- 五层评测里，held-out 冻结集与成对比较是防“尺子被优化”的两条制度性防线；推理评测必须多次采样报方差。
- 训练判读读**曲线组合形态**而非单点：奖励/熵/KL/长度/clip 的联合形态几乎覆盖全部常见病害。
- reward hacking 排查四步（长度分解 → 反事实改写 → 盲评 → 回灌）成本递增，前两步应纳入每周例行。

第十五章 开放问题与总结

最后一章做两件事：把全书内容折叠成一张**可操作的选型决策树**（图 15.1），然后诚实地列出目前没有好答案的问题——它们是读者可以真正做出增量贡献的地方。

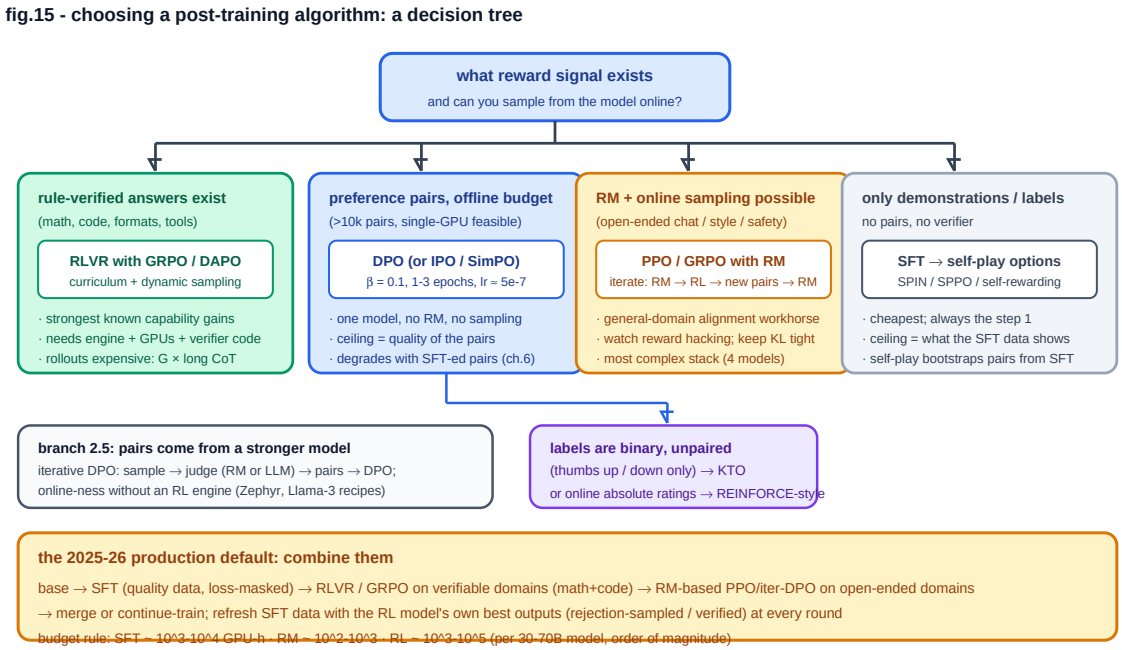


图 15.1　算法选型决策树：从“手里有什么信号”出发四问到底，底部给出 2025–26 年生产环境的默认组合。

15.1　选型决策树：五个问题

1. **有没有可机器判定的正确性？**（数学答案、测试、格式、终态校验）——有 → RLVR + GRPO/DAPO（第 八, 七 章），这是当前单位算力收益最大的路线；没有 → 走偏好路线，继续问 2。

2. **偏好数据是什么形态？**成对偏好 → DPO 及变体（第 六 章）；仅有点赞/点踩 → KTO；只有示范 → SFT 后考虑自博弈（第 十 章）。

3. **能否承担在线采样？**能（有引擎与集群）→ 迭代 DPO 或 PPO/GRPO+RM；不能 → 离线单轮 DPO，并接受其分布错配风险。

4. **规模与预算？** ≤8B + LoRA → colocate；8–70B 全参 → hybrid（第 十一 章 C 范式）；超长 rollout → 异步 D。

5. **要不要推理能力？**要 → 混合流水线（cold-start SFT → RLVR → 拒采 SFT → 再 RL，第 八 章）；不要 → 对齐流水线止于 DPO/PPO。

15.2 开放问题：六个没有共识的方向

1. **RM 的上限在哪里。**RM 是人类偏好的低维投影，长期被优化的 RM 必然被攻破（Goodhart）；RLVR 只是绕开而非解决了这个问题。开放域（写作、共情、品味）至今没有“不可 hack 的奖励”候选。RM 集成、判官多样性训练、对抗性数据闭环都只是减速带。

2. **熵的经济学。**熵既是探索存量（要保）又是收敛信号（要看降），目前没有理论告诉我们“该花多少熵换多少奖励”。重置参考模型、温度调度、熵正则的实践五花八门，缺一个统一框架。

3. **过程监督的甜点位。**PRM 提供密集信用但自带过拟合面（PRM 也是被学出来的代理）。PRIME 式在线过程奖励是一个折中，但“过程奖励何时优于结果奖励”仍无定量判据。

4. **涌现的边界。**R1 的反思/回溯在什么模型规模、什么数据分布下才会涌现？小模型上蒸馏可以复制行为，但“自发涌现”的临界条件是 2026 年最有价值的实验问题之一。

5. **评测的军备竞赛。**训练目标不断吸收评测（含 RL 阶段接触到的任何带答案数据），held-out 集的公信力持续贬值；需要制度性答案（滚动任务生成、实时任务市场）而非更好的静态集。

6. **自博弈的天花板。**SPIN 类方法的均衡点就是数据分布（第 十 章）；要超越必须有外部信号。“模型自己给自己出题并验证”（自我提出-解决-验证闭环）是唯一在逻辑上可能突破天花板的方向，其失败模式（自我确认偏差）刚刚开始被研究。

15.3 学习路线：从本报告到动手

四周实践路线（假设一张 8 卡 A100/H100 或等效）

- **第 1 周：**跑通 TRL 的 DPO（1–7B + 公开偏好对），复现 β 敏感性；用第 十二 章的参考实现对照理解 log-prob 移位与掩码。
- **第 2 周：**GSM8K 上手写验证器；用单文件 GRPO 训 1.5B 模型 500 步，画出奖励/熵/长度三条曲线（复现第 八 章的“心电图”）。
- **第 3 周：**接入 vLLM（prefix caching + sleep/wake），测量第 十三 章的墙钟模型各分项；对比 K=1 与 K=4。
- **第 4 周：**做一次完整消融（G、温度、clip-higher 三因子），按第 十四 章协议报告多种子结果。走完这四周，读者对全书的每个公式都会有一份“亲手摸过”的对应物。

15.4 结语

回望全书的逻辑主线，只有三句话：

1. **后训练是重新加权，不是灌输知识。**策略梯度的形态（$\nabla \log \pi \cdot A$）决定了 RL 只能放大采样分布里已有的模式——数据质量与基座能力决定上限，算法决定逼近速度。

2. **一切设计都在管理三个量。**方差（优势估计）、距离（KL/off-policy）、熵（探索存量）——从 PPO 的 clip 到 GRPO 的组基线、从 KL 双实现之辨到异步系统的级联修正，无一例外。

3. **奖励的可靠性是唯一的“宪法问题”。**代理必被攻破；verifier 换掉了代理但缩窄了疆域。后训练的全部工程纪律，都是在这个约束下尽量安全地跑得更快。

2022 年，对齐还是“让模型听话”的修补术；2026 年，它已经是把算力转化为推理能力的主动机。这条曲线的斜率来自算法（DPO、GRPO、DAPO）、系统（hybrid/异步引擎）、数据（RLVR 的题库工程）三条线的合力——希望本报告给了读者在任意一条线上继续往前走所需的全部地基。

附录 A 附录：符号表与算法速查

A. 符号总表

记号	首次出现	含义
x； $y = (y_1, \dots, y_T)$	二	提示；回答（长度 T 由采样决定，止于 EOS）
$s_t = (x, y_{<t})$	2.1	token 级状态：当前上下文
$\pi_\theta(y_t \mid x, y_{<t})$	二	策略（即 LLM）；θ 为参数
π_{ref}， π_{old}	二	冻结参考策略；采样时刻的策略快照
$r_\phi(x, y)$	四	奖励模型输出（标量）；ϕ 为其参数
r_t， $R(\tau)$	二	token 级奖励；轨迹级回报
G_t	2.2	从 t 起的折扣回报
$V(s)$， $Q(s,a)$， $A(s,a)$	2.3	状态价值、动作价值、优势
$\hat{A}_t$	2.3	优势的估计量（GAE 或组相对）
δ_t， λ	2.4	TD 误差；GAE 插值参数
$\rho_t = \pi_\theta / \pi_{\text{old}}$	2.5	token 级重要性比率
ϵ（$\epsilon_{\text{lo}}, \epsilon_{\text{hi}}$）	五, 七	clip 半径（DAPO 非对称上下侧）
β	五	KL 系数（偏离税税率）
G（组）	七	GRPO 每提示采样数（勿与回报 G_t 混淆）
K	五	内层 epoch 数（样本复用次数）
y_w, y_l	四	偏好对中更优 / 更劣的回答
$h_\theta = \beta \log(\pi_\theta / \pi_{\text{ref}})$	六	DPO 隐式奖励
$Z(x)$	六	配分函数（提示级常数，BT 差分中消去）
$\mathcal{H}(\pi)$	八	策略熵（nats/token）
pass@k	九	k 次采样至少一次正确的比率
$P_{\text{cov}}(N)$	9.2	BoN 覆盖率 $1 - (1-p)^N$

表 A.1　全书符号一览。

B. 算法速查表

算法	核心目标（一行版）	章节
REINFORCE	$\nabla J = \mathbb{E}[\sum_t \nabla \log \pi \cdot G_t]$；直接、高方差	2.2
RLOO / ReMax	组均值（留一法）基线的无偏 REINFORCE	七

PPO	$\mathbb{E}[\min(\rho\hat{A}, \text{clip}(\rho, 1\pm\epsilon)\hat{A})] - \beta$ KL；critic + GAE	五
RLHF（目标）	$\max_\pi \mathbb{E}[r] - \beta\,\text{KL}(\pi\|\pi_{\text{ref}})$；闭式解为 Boltzmann 加权	5.1
DPO	$-\log\sigma\big(\beta\log\frac{\pi(y_w)}{\pi_{\text{ref}}(y_w)} - \beta\log\frac{\pi(y_l)}{\pi_{\text{ref}}(y_l)}\big)$；RLHF 的闭式换元	六
IPO / cDPO	有界二次替代 / 标签平滑的 DPO	六
SimPO	去参考：$\frac{\beta}{\|y\|}\log\pi$ 加 margin 比较	六
KTO	单边二元标签 + 前景理论不对称权重	六
ORPO	SFT 损失 + odds-ratio 惩罚，单阶段	六
GRPO	$\hat{A}_{i,j} = (r_{i,j} - \text{mean})/\text{std}$ 广播全 token；无 critic	七
Dr.GRPO	去掉 ÷std 与序列均值（去长度偏置）	七
DAPO	clip-higher + 动态采样 + token 级归一 + 超长整形	七
RLVR	规则验证器作奖励的 GRPO/PPO；组基线适配二值奖励	八
PRIME	隐式过程奖励 $\hat{r}_t = \beta\log\frac{\pi_{\text{PRM}}}{\pi_{\text{SFT}}}$，在线更新	八
SPIN	$F_t = \log p_{\text{data}}/p_t$ 当奖励；均衡点 = 数据分布	10.1
BoN / SC	$1-(1-p)^N$ 覆盖 ∧ 选择；多数票指数收敛	9.2
L1（预算 RL）	$R = \mathbf{1}[\text{correct}] - \lambda\max(0, \|y\| - L^\star)$	九

表 A.2　按“梯度从哪里来”排序的算法一览。

C. 完整训练循环（综合伪代码）

一段式 RLVR 训练循环（融合本书全部默认项）

1. **初始化**：actor = SFT 检查点；ref 冻结为同权重；引擎载入 actor（开启 prefix caching）；
2. **每步**：
   a. 采 B 个提示（按近期通过率加权，目标区间 30–70%）；
   b. 引擎采样 G 条/题（温度与部署一致）；回传 token log-probs；
   c. 验证器打分；**动态采样**：全同分组丢弃重采直至配额满；
   d. 截断样本给超长整形惩罚；组内标准化（Dr.GRPO 形态）；
   e. 白化优势；ref 前向得逐 token KL 项（k3 估计器，fp32）；
   f. K=1 个内层 epoch：clipped 目标（ϵ_{lo}=0.2，ϵ_{hi}=0.28）+ βKL，token 级归一；
   g. 记录六指标（reward / pass-rate / KL / entropy / clip / length-p95）；KL 超带即回滚 lr；
3. **每若干小时**：held-out 评测一次；奖励去长度化复查一次；人工盲评抽样；题库补新题；
4. **结束条件**：held-out 收敛或熵降至阈值或学习信号枯竭（全同分组占比持续过高）——

三者任一先到即停，保留最佳检查点。

参考文献

[1] L. Ouyang et al. *Training language models to follow instructions with human feedback* (InstructGPT). NeurIPS, 2022.

[2] J. Schulman et al. *Proximal Policy Optimization Algorithms*. arXiv:1707.06347, 2017.

[3] J. Schulman et al. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. ICLR, 2016.

[4] J. Schulman. *Approximating KL Divergence*. Web log post, 2016 (k3 estimator).

[5] R. A. Bradley, M. E. Terry. *Rank Analysis of Incomplete Block Designs: The Method of Paired Comparisons*. Biometrika, 1952.

[6] Y. Bai et al. *Constitutional AI: Harmlessness from AI Feedback*. arXiv:2212.08073, 2022.

[7] Long Ouyang 等的前身工作与综述见 *Learning from Human Preferences* 系列博文 (OpenAI, 2016–2018).

[8] R. Rafailov et al. *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. NeurIPS, 2023.

[9] M. Azar et al. *A General Theoretical Paradigm to Understand Learning from Human Preferences* (IPO). AISTATS, 2024.

[10] K. Ethayarajh et al. *KTO: Model Alignment as Prospect Theoretic Optimization*. ICML, 2024.

[11] H. R. Wang et al. *ORPO: Monolithic Preference Optimization without Reference Model*. EMNLP, 2024.

[12] Y. Meng et al. *SimPO: Simple Preference Optimization with a Reference-Free Reward*. NeurIPS, 2024.

[13] E. Mitchell. *A Note on DPO and Posterior Sampling*. arXiv:2406.14501, 2024 (likelihood displacement 讨论).

[14] Z. Shao et al. *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models* (GRPO). arXiv:2402.03300, 2024.

[15] DeepSeek-AI. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. arXiv:2501.12948, 2025.

[16] Kimi Team. *Kimi k1.5: Scaling Reinforcement Learning with LLMs*. arXiv:2501.12599, 2025.

[17] Q. Yu et al. *DAPO: An Open-Source LLM Reinforcement Learning System at Scale*. arXiv:2503.14476, 2025.

[18] Z. Liu et al. *Understanding R1-Zero-Like Training: A Critical Perspective* (Dr. GRPO). NeurIPS, 2025.

[19] A. Ahmadian et al. *Back to Basics: Revisiting REINFORCE-Style Optimization for Learning from Human Feedback in LLMs* (RLOO). ACL, 2024.

[20] T. Li et al. *ReMax: A Simple, Effective, and Efficient Reinforcement Learning Method for Aligning Large Language Models*. NeurIPS, 2024.

[21] H. Lightman et al. *Let's Verify Step by Step* (PRM800K). ICLR, 2024.

[22] P. Wang et al. *Math-Shepherd: Verify and Reinforce LLMs Step-by-Step without Human Annotations*. ACL, 2024.

[23] L. Yuan et al. *PRIME: Process Reinforcement through Implicit Rewards*. arXiv:2502.01456, 2025.

[24] K. Cobbe et al. *Training Verifiers to Solve Math Word Problems* (GSM8K). arXiv:2110.14168, 2021.

[25] D. Hendrycks et al. *Measuring Mathematical Problem Solving With the MATH Dataset*. NeurIPS Datasets, 2021.

[26] C. Snell, J. Lee, Y. Su, Z. Xu. *Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters*. ICLR, 2025.

[27] B. Brown et al. *Large Language Monkeys: Scaling Inference Compute with Repeated Sampling*. ICLR, 2025.

[28] X. Wang et al. *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. ICLR, 2023.

[29] S. Yao et al. *Tree of Thoughts: Deliberate Problem Solving with Large Language Models*. NeurIPS, 2023.

[30] M. Muennighoff et al. *s1: Simple Test-Time Scaling*. arXiv:2501.19393, 2025.

[31] P. Aggarwal et al. *L1: Controlling How Long A Reasoning Model Thinks With Reinforcement Learning*. arXiv:2503.04697, 2025.

[32] Z. Chen et al. *SPIN: Self-Play Fine-Tuning Converts Weak Language Models to Strong Language Models*. ICML, 2024.

[33] Y. Wu et al. *Self-Play Preference Optimization for Language Model Alignment*. arXiv:2405.00675, 2024.

[34] W. Yuan et al. *Self-Rewarding Language Models*. arXiv:2401.10020, 2024.

[35] C. Burns et al. *Weak-to-Strong Generalization: Eliciting Strong Capabilities With Weak Supervision*. arXiv:2312.09390, 2023.

[36] L. Tunstall et al. *Zephyr: Direct Distillation of LM Alignment*. arXiv:2310.16944, 2023.

[37] A. Grattafiori et al. *The Llama 3 Herd of Models*. arXiv:2407.21783, 2024 (迭代 DPO 配方).

[38] H. Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. arXiv:2307.09288, 2023 (RLHF 细节).

[39] C. Zhou et al. *LIMA: Less Is More for Alignment*. NeurIPS, 2023.

[40] Y. Wang et al. *Self-Instruct: Aligning Language Models with Self-Generated Instructions*. ACL, 2023.

[41] E. J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. ICLR, 2022.

[42] C. Hsu et al. *On the Implementation of Reinforcement Learning from Human Feedback for Large Language Models* (RLHF 工程陷阱与 KL 处理讨论). 2023.

[43] H. Sheng et al. *Hybrid RLHF / Flexible Rollouts* 系列, veRL 技术报告, 2024–2025.

[44] W. Sheng et al. *verl: An Open-Source Unified Reinforcement Learning Framework for Post-Training Large Language Models*. HPML/Blog, 2024–2025 (hybrid engine 与 1F1B 语义).

[45] J. Hu et al. *OpenRLHF: An Easy-to-Use, Scalable and High-Performance RLHF Framework*. arXiv:2405.11143, 2024.

[46] L. von Werra et al. *TRL: Transformer Reinforcement Learning*. GitHub, HuggingFace, 2022–2026.

[47] W. Kwon et al. *Efficient Memory Management for Large Language Model Serving with PagedAttention* (vLLM). SOSP, 2023.

[48] L. Zheng et al. *SGLang: Efficient and Flexible LLM Serving with RadixAttention*. NeurIPS, 2024.

[49] AReaL Team (Ant/清华). *AReaL: A Large-Scale Asynchronous Reinforcement Learning System for Language Reasoning*. arXiv:2505.24298, 2025.

[50] W. Jiang et al. *AsyncFlow: An Asynchronous RLHF Framework for Large-scale LLM Post-training*. 2025.

[51] M. Zhao et al. *Agent RL 综述: Agent Reinforcement Learning for LLM Agents* / SkyRL 技术报告, 2025.

[52] G. Guo et al. *DeepSeek-R1* 分析复现系列（含 Dr.GRPO 长度偏置的独立验证）, 2025.

[53] L. Gao et al. *Scaling Laws for Reward Model Overoptimization* (延续 Gao et al. 2022). ICML, 2023.

[54] J. Skalse et al. *Defining and Characterizing Reward Hacking (Goodhart's Law in RL)*. NeurIPS, 2023.

[55] OpenAI. *Learning to Reason with LLMs* (o1 系统公告), 2024.

[56] Qwen Team. *Qwen3 Technical Report*（RLVR 与思考预算设计）. arXiv:2505.09388, 2025.

[57] Y. Tang et al. *Generalized Preference Optimization: A Unified Approach to Offline Alignment* (GPO, BoN 对偶视角). ICML, 2024.

[58] Y. Liu et al. *RLHF-V / VLM 偏好训练系列*（多模态奖励与幻觉抑制）, 2024–2025.

[59] OpenAI. *GPT-4 Technical Report*. arXiv:2303.08774, 2023 (大规模 RLHF 报告先例).