大模型后训练
对齐、推理与智能体

Post-Training of Large Language Models:

Alignment, Reasoning, and Agency

一门关于现代语言模型优化的
研究生课程



2026 年 7 月版

文献覆盖至 2026 年 7 月 31 日

大模型后训练：对齐、推理与智能体

2026 年 7 月版。编译于 2026 年 7 月 31 日。

本书是对公开研究文献中大语言模型后训练工作的一次综合梳理。书中每一项经验性结论都在参考文献中标明了原始出处；凡文献之间存在分歧之处，本书如实报告分歧，而不代为裁定。引自论文的数值结果均为其作者所报告的数值，本书未做独立复现。

本书所描述的领域变化很快。标有移动靶提示的小节在编译日期时是准确的，但读者应对照当前文献重新核实。建议读者把算法性内容——目标函数、估计量及其失效模式——视为本书中经久有效的部分，而把排行榜数字视为转瞬即逝的部分。

排版约定。 核心思想 框给出一节中最重要的那一个要点。 工程实践 框给出具体的超参数与工程建议。 常见陷阱 框描述那些常见到值得专门命名的错误。 推导 框中是初读时可以跳过的代数细节。 历史注记 框解释事情为什么会变成今天这个样子。

标有 $\star$ 的习题属于研究性题目；标有 † 的习题是需要 GPU 的编程项目。

正文使用 Palatino，标题使用 Helvetica 字族；中文使用思源宋体与思源黑体。以 LaTeX 排版。

目录

前言 xix

前言 xix

如何使用本书 xxiii

如何使用本书 xxiii

记号 xxvii

记号 xxvii

第 1 部分　基础 1

第 1 章　后训练范式 3

1.1　后训练是什么 3
1.2　预训练为何不够 5
1.3　简史 7
1.4　现代流水线的解剖 9
1.5　三种反馈来源 11
1.6　后训练的经济学 12
1.7　反复出现的主题 13
1.8　一个完整演算的微型例子 14
本章小结 15

第 2 章 预备知识：作为策略的语言模型 **19**
2.1 作为条件分布的自回归模型 19
2.2 两种决策过程视角 21
2.3 语言模型 MDP 的特殊之处 22
2.4 散度及其估计量 24
2.5 熵，以及它为何会坍缩 26
2.6 重要性采样与离策略程度 27
2.7 蒙特卡洛估计与基线 28
2.8 度量策略：pass@k 及其同类 29
2.9 优化札记 30
本章小结 31

第 3 章 监督微调 **35**
3.1 目标函数 35
3.2 作为模仿学习的 SFT 及其局限 37
3.3 真正重要的损失构造细节 39
3.4 对话模板与训练/部署契约 43
3.5 超参数与调度，附带理由 45
3.6 灾难性遗忘与数据配比设计 47
3.7 参数高效微调 48
3.8 拒绝采样微调与专家迭代 51
3.9 长思维链 SFT 53
本章小结 55

第 4 章 后训练的数据 **59**
4.1 后训练数据的四种类型 59
4.2 人工标注及其统计性质 61
4.3 合成指令数据 64
4.4 选择：少即是多，以及何时并非如此 68
4.5 难度、可学习性与课程 72

4.6 去污染 . . . 75
4.7 中训练与继续预训练 . . . 77
4.8 智能体训练的数据 . . . 80
4.9 来源、许可与可复现性 . . . 81
本章小结 . . . 82

第 2 部分 基于反馈的对齐 87

第 5 章 偏好建模与奖励模型 89
5.1 奖励模型是用来做什么的 . . . 89
5.2 Bradley–Terry 模型及其亲属 . . . 90
5.3 架构与训练 . . . 94
5.4 准确率上限与校准 . . . 97
5.5 系统性偏差 . . . 100
5.6 生成式奖励模型与 LLM-as-judge . . . 103
5.7 过程奖励模型与结果奖励模型 . . . 106
5.8 奖励作弊与过度优化 . . . 106
5.9 在不断变化的流水线中维护奖励模型 . . . 110
5.10 何时不需要奖励模型 . . . 111
本章小结 . . . 112

第 6 章 面向语言模型的策略梯度方法 117
6.1 目标函数及其梯度 . . . 117
6.2 基线与评论家 . . . 119
6.3 信赖域：从 TRPO 到 PPO . . . 122
6.4 KL 正则化目标及其解 . . . 124
6.5 KL 项的作用 . . . 125
6.6 一个统一模板 . . . 126
6.7 这个估计器在代码里长什么样 . . . 127

6.8 为什么不用基于值或基于模型的方法? 130
本章小结 130

第 7 章 RLHF 的工程实践 133
7.1 标准流水线及其代价 133
7.2 KL 放进奖励，还是放进损失 135
7.3 奖励的后处理 139
7.4 优势与回报的处理 142
7.5 关于实现细节的那批文献 144
7.6 诊断，以及一次健康的训练长什么样 144
7.7 真正要紧的超参数 149
7.8 迭代式与在线 RLHF 151
7.9 取代它的是什么，以及它还有什么用 153
本章小结 154

第 8 章 直接对齐算法 161
8.1 不用强化学习，从奖励直达策略 161
8.2 DPO 是什么，不是什么 164
8.3 设计空间：一个框架，五十种算法 166
8.4 按病理编排的目录 168
8.5 似然位移 171
8.6 长度 173
8.7 覆盖性、过度优化与 χPO 176
8.8 粒度：在样本对之下与之上 177
8.9 参考策略 178
8.10 直接对齐与策略优化之比较 179
8.11 一份可用的配方 180
8.12 2026 年直接对齐所处的位置 181
本章小结 182

第 9 章 在线、迭代与博弈论式对齐 **189**
9.1 为什么采样分布至关重要 . . . 189
9.2 迭代式直接对齐 . . . 191
9.3 无标签学习 . . . 192
9.4 探索：把启发式做法变成算法 . . . 194
9.5 抛弃 Bradley–Terry：把对齐当作博弈 . . . 194
9.6 迭代之后会出什么问题 . . . 198
9.7 一份可用的迭代配方 . . . 199
9.8 哪些结论真正站得住 . . . 200
本章小结 . . . 201

第 3 部分 推理 **205**

第 10 章 推理与测试时计算 **207**
10.1 思维链是计算，不是解释 . . . 207
10.2 推理计算这条轴 . . . 209
10.3 无验证器的聚合：自洽性 . . . 212
10.4 有验证器的聚合：best-of-n . . . 215
10.5 验证器比生成器更值 . . . 221
10.6 在部分解上做搜索 . . . 223
10.7 顺序修订与自我纠正 . . . 226
10.8 用训练计算换推理计算 . . . 228
10.9 通往第三部分的桥梁 . . . 230
本章小结 . . . 232

第 11 章 可验证奖励的强化学习 **237**
11.1 这一设定 . . . 237
11.2 组相对策略优化 . . . 239
11.3 R1-Zero：在基座模型上使用结果奖励 . . . 241

11.4 完整的 R1 流水线 242
11.5 一次做出不同选择的独立复现 243
11.6 可验证任务的奖励设计 244
11.7 一份最小可用配方 246
11.8 奖励并非你所以为的那样的信号 247
本章小结 248

第 12 章 组相对策略优化的解剖 253
12.1 十种病理 253
12.2 归一化器：长度偏置 254
12.3 归一化器：难度偏置 256
12.4 退化组 257
12.5 词元级与序列级重要性比值 258
12.6 非对称截断与熵 261
12.7 长度控制 264
12.8 KL 之争 264
12.9 基于值的分支 265
12.10 混合专家模型的特有问题 266
12.11 设计空间 266
12.12 为什么这一文献中的大多数比较都检验效能不足 266
12.13 哪些已成定论，哪些还没有 269
本章小结 269

第 13 章 信用分配与过程监督 273
13.1 精确表述这个问题 273
13.2 过程奖励模型 276
13.3 隐式与派生的过程奖励 280
13.4 蒙特卡洛前缀值 283
13.5 片段级与轮次级的信用 287
13.6 树状采样与共享前缀 288

13.7 更细的信用真的有用吗? 290
13.8 信用分配：一个推理时还是训练时的选择 292
本章小结 294

第 14 章 强化学习究竟教会了什么? 299

14.1 把问题说精确 299
14.2 悲观一方的论据 302
14.3 多样性与模式坍缩 306
14.4 乐观一方的论据 308
14.5 2026 年的调和 309
14.6 锐化何时有害：一个刻画 312
14.7 pass@k 应该成为目标函数吗? 314
14.8 强化学习与蒸馏之比较 316
14.9 该下什么结论，以及如何为你自己的模型测量它 318
14.10 开放问题 320
本章小结 321

第 15 章 超越可验证领域 327

15.1 可验证性谱系 327
15.2 评分细则：用分解来制造一个验证器 330
15.3 把评判模型当作奖励函数 333
15.4 自我奖励与生成–验证差距 336
15.5 没有验证器时的奖励作弊 337
15.6 混合奖励：先把闸，再评分 339
15.7 半可验证构造 340
15.8 截至 2026 年年中已确立的内容 342
15.9 为一项新任务构建一个可辩护的奖励 343
本章小结 344

第 4 部分 智能体 **353**

第 16 章 从推理到智能体 **355**

16.1 究竟改变了什么 355
16.2 词元级 POMDP 358
16.3 轮次级 MDP 361
16.4 观测掩码 366
16.5 跨轮次的信用分配 370
16.6 时域是内生的 372
16.7 新的失效模式 374
16.8 关于时域的课程 375
16.9 一个最小的智能体式训练循环 376
16.10 第四部分路线图 377
本章小结 378

第 17 章 工具集成的强化学习 **387**

17.1 工具为何改变能力边界 387
17.2 交错模式及其变体 389
17.3 检索与搜索智能体 392
17.4 代码与计算类工具 394
17.5 工具使用的奖励设计 396
17.6 要调用多少次工具? 398
17.7 深度研究智能体 401
17.8 把环境观测当作稠密监督 403
17.9 一套工具强化学习技术栈必须提供什么 405
17.10 哪些是真正已知的 407
本章小结 408

第 18 章 长时域智能体强化学习 **419**

18.1 四项约束的量化 419

18.2 一百轮次下的奖励稀疏 420
18.3 轮次级信用分配 424
18.4 上下文长度与历史管理 428
18.5 采样成本与掉队者问题 432
18.6 方差与稳定性 436
18.7 时域课程 437
18.8 已知的与未知的 438
本章小结 439

第 19 章 环境、沙箱与任务供给 **447**
19.1 环境是奖励函数的一部分 447
19.2 环境类别剖析 449
19.3 隔离，以及沙箱池的算术 450
19.4 确定性，以及非确定性让组相对估计量付出的代价 454
19.5 环境内部的验证 457
19.6 习得环境与世界模型 458
19.7 任务供给才是真正的瓶颈 460
19.8 环境集群、枢纽，以及标准化能买到什么 463
19.9 在训练之前先验证一个新环境 464
本章小结 466

第 20 章 多智能体训练与自博弈 **473**
20.1 「多智能体」所指为何 474
20.2 形式化框架，以及其中迁移不过来的那部分 475
20.3 跨智能体的信用分配 476
20.4 在用的训练模式，以及各自如何失败 482
20.5 作为动作的通信，以及它掩盖了什么 484
20.6 智能体式场景下的自博弈 485
20.7 辩论、批评与验证博弈 487
20.8 一次多智能体采样的算术 488

20.9 究竟何时才该按多智能体来训练 489
本章小结 490

第 5 部分 系统 **497**

第 21 章 大规模后训练的系统 **499**
21.1 一个训练步的解剖 499
21.2 同置与分离式 501
21.3 RL 回路中的并行 505
21.4 内存 508
21.5 权重同步 510
21.6 异步 512
21.7 掉队者问题 516
21.8 混合专家的系统问题 519
21.9 框架图景，2026 年中 520
21.10 调试一次缓慢或坏掉的运行 522
本章小结 524

第 22 章 采样与训练的失配 **531**
22.1 是四个策略，不是三个 531
22.2 测量它 533
22.3 被腐蚀的比值为什么会毁掉训练 537
22.4 修正一：截尾重要性采样 539
22.5 修正二：修好数值 542
22.6 修正三：结构性回避 544
22.7 温度与采样失配 545
22.8 陈旧度：另一种离策略性 548
22.9 优化了错误的策略 549
22.10 一份清单 550

本章小结 552

第 23 章 效率与成本控制 559

23.1 钱花到哪里去了 559
23.2 采样效率：不生成没用的词元 565
23.3 前缀共享与缓存 568
23.4 参数高效的强化学习 571
23.5 量化的经济学 574
23.6 下一单位算力该花到哪里？ 577
23.7 蒸馏通常比强化学习更便宜 579
23.8 诚实地度量效率 580
23.9 一份成本控制清单 581
本章小结 582

第 6 部分 测量、安全与前沿 587

第 24 章 评测与测量 589

24.1 我们究竟要测什么，以及为什么难测 589
24.2 基准比较的统计学 591
24.3 pass@k 及其估计量 599
24.4 污染 601
24.5 把评判模型当作仪器 603
24.6 评测推理模型 605
24.7 评测智能体 608
24.8 测量后训练破坏了什么 609
24.9 一份报告标准 610
24.10 开放问题 611
本章小结 611

第 25 章 安全对齐与稳健性 **619**

25.1 安全后训练被要求做的是什么 619
25.2 浅层性结果及其所解释的现象 620
25.3 作为学出来的分类器的拒答，及其两类错误 625
25.4 把越狱当作一个优化问题 628
25.5 对抗训练与红队测试 631
25.6 安全性对后续训练的稳健性 633
25.7 智能体式场景中的安全 635
25.8 推理的忠实性与可监控性 637
25.9 谄媚与群体层面的效应 639
25.10 实际该怎么做 641
本章小结 643

第 26 章 开放问题与前沿 **649**

26.1 本书确立了什么 649
26.2 激发问题仍然开放，但如今被提得更好了 650
26.3 不是代理的目标函数 651
26.4 信用分配在长时域上仍未解决 653
26.5 环境是瓶颈，而这是一个内核为研究问题的工程瓶颈 655
26.6 数值，以及我们实际在优化的那个目标 656
26.7 我们测不出自己在做什么 658
26.8 值得关注的几条汇合 660
26.9 本书未能妥善处理的问题 662
26.10 一段收尾的论证 663
本章小结 665

第 7 部分 附录 **675**

附录 A 数学背景 **677**

A.1 分布之间的散度 . . . 677
A.2 集中不等式与样本量 . . . 679
A.3 凸分析与 Gibbs 变分原理 . . . 679
A.4 策略梯度与得分函数 . . . 680
A.5 指数族与 softmax 参数化 . . . 681
A.6 Bradley–Terry 与成对比较 . . . 681
A.7 序统计量与 best-of-n . . . 682
A.8 到哪里读更多 . . . 683

附录 B 扩展推导 **685**
B.1 逐词元的策略梯度 . . . 685
B.2 广义优势估计 . . . 686
B.3 从信赖域到截断 . . . 687
B.4 从留一法 REINFORCE 导出 GRPO . . . 688
B.5 截断重要性采样：偏差与方差 . . . 688
B.6 DPO 梯度及其两个结构性性质 . . . 689
B.7 为什么 χ^2 正则化能控制可集中性 . . . 690
B.8 best-of-n 的 KL 代价 . . . 690
B.9 带逐轮成本的最优停止 . . . 691

附录 C 参考实现 **693**
C.1 逐词元的对数概率 . . . 693
C.2 组相对优势 . . . 694
C.3 策略优化损失 . . . 695
C.4 面向采样失配的截断重要性采样 . . . 696
C.5 直接偏好优化及其变体 . . . 696
C.6 多轮轨迹的损失掩码 . . . 697
C.7 一个最小但完整的训练步 . . . 698
C.8 一份诊断契约 . . . 699
C.9 可复现性检查清单 . . . 699

附录 D 术语表 **701**

A 701
B 702
C 702
D 702
E 703
G 703
I 704
K 704
L 704
N 704
O 705
P 705
R 705
S 706
V 706

参考文献 **761**

前言

在 2019 年，一门关于语言模型的研究生课程大概会用十一周讲架构与预训练目标的优化，再顺带用一周讲讲「下游微调」。今天，这个比例已经完全颠倒过来。一个原始的预训练 Transformer，与一个能够进行对话、证明竞赛级定理、或者自主修复大型代码库中缺陷的系统，二者之间的差距几乎全部是后训练的差距。弥合这一差距靠的是监督式模仿、奖励建模，以及最重要的——针对反馈的强化学习：来自人类标注者的反馈、来自其他模型的反馈、来自形式化验证器的反馈，以及以程序执行结果和工具返回值形式出现的、来自世界本身的反馈。

本书讲的就是这一差距是如何被弥合的。

为什么写这本书，为什么是现在

2022 年到 2026 年之间发生了三件事，使得写一本教材既成为可能，也成为必要。

第一，**对齐已经成为一门足够成熟、可以被完整写下来的工程学科**。2022 年的那套配方——先监督微调，再训练一个 Bradley–Terry 奖励模型，然后用带 KL 锚定的 PPO 优化——曾有一段时间只是在少数几家工业实验室之间口耳相传的经验。到 2024 年，它已被公开复现过许多次，其失效模式被逐一编目，还派生出一整族完全绕开奖励模型的「直接」方法。偏好优化的理论如今已经成熟到可以当作理论来讲授，而不再只是行业传说。

第二，**针对可验证奖励的强化学习被证明是一种通用的推理能力激发方法**。2025 年 1 月 DeepSeek-R1 的发布公开演示了——并且公布了配方——一个仅因给出正确最终答案而获得奖励的基座模型，会自发地学会给更难的问题分配更多词元、学会回溯、学会检查自己的工作、学会分解问题。原本看起来只是一堆提示技巧的东西——思维链、自洽性、自我验证——原来是可以被训练出来的行为。此后两年间涌现出了数量庞大、迭代极快、最初也相当令人困惑的算法改进文献。本书有很大一部分篇幅用于梳

理这批文献：说明 DAPO、Dr. GRPO、GSPO、CISPO 及其众多亲属并不是一堆随机拼凑的技巧，而是针对原始估计量中少数几个定义明确的统计病理所做的系统性回应。

第三，**前沿从单轮回答转向了长时域的智能体**。一旦模型能够推理，接下来自然要问的就是它能否行动——调用搜索引擎、运行代码、编辑代码仓库、驱动浏览器——以及当一个回合有五十轮而不是一轮、当上下文中有一部分是由环境而非策略写入、当奖励只在最末尾才到来时，同一套强化学习机制是否仍然有效。智能体 RL 是这三个主题中最未定型的一个，也是工程相对于理论占比最高的一个。它同时也是当前该领域注意力最集中的地方。

这本书是什么

这是一本**研究生教材**，在我看来这意味着四项承诺。

它做推导，而不是列清单。书中每一个目标函数都会连同促成它的假设、产生它的推导，以及它失效的区间一并给出。读完第8章，你应当能够在黑板上从 KL 正则化的奖励最大化问题出发推导出 DPO 损失，并准确说出当偏好数据含噪时是哪一步出了问题。读完第12章，你应当能够读一篇新的策略优化论文，在 节 12.11的五轴设计空间中定位它的目标函数，并在动手跑之前就预测出它的失效模式。

它对不确定性保持诚实。一本写作时距研究前沿如此之近的教材，有义务区分三类结论：已被反复复现的（KL 正则化能阻止奖励模型被过度优化；组相对基线确实有效）、仍有争议的（针对可验证奖励的 RL 究竟是扩展了模型的推理能力，还是仅仅锐化了基座模型既有的分布）、以及仅有单篇证据、未必经得起时间检验的（2026 年的许多条目）。我尽量把这些区分显式地标注出来，而不是把一切压平成一段自信的叙述。

它把数学与代码连接起来。一份 RLHF 实现里大概包含二十个没有任何论文写明的决定：KL 是按 $\log \rho$ 估计还是按 $\rho - 1 - \log \rho$ 估计，优势是按批还是按组归一化，损失是在序列内对词元取平均还是在批内对序列取平均，对数概率来自训练引擎还是推理引擎。这些决定的影响，往往比算法本身的选择还要大。Chapters 7 and 22与Appendix C把它们当作一等内容来处理。

它要求读者动手。每一章末尾都有习题。有些是推导；有些要求你在一份看似合理的实现里找出缺陷；还有一些（†）是需要 GPU 的项目。本书在教学上的主张是：没有亲眼看着一条熵曲线坍缩下去，你就无法真正理解面向语言模型的策略梯度方法。

这本书不是什么

它不是一篇综述。综述追求覆盖面，教材追求一条路径。本书引用了数百篇相关论文，但只详细展开了几十个算法，选取的依据是它们能说明某个经久有效的思想，而不是它们曾在排行榜上占据过什么位置。

它不是一本讲预训练、架构或推理系统的书，除非后训练迫使我们不得不关心它们——而它确实会迫使我们关心，程度超出多数人的预期，这正是第22章 要讲浮点算术的原因。

它也不是一本讲 AI 政策或对齐社会学的书。安全问题在第25章中以技术问题的形式出现：如何训练出能够泛化的拒答行为，如何度量越狱鲁棒性，如何检测出一个已经学会满足你的奖励函数而非你的真实意图的策略。

预备知识

本书假定读者具备：

- **概率与统计**，达到第一门研究生课程的水平：期望、条件分布、集中不等式、极大似然、delta 方法。重要性采样会被反复用到。
- **机器学习**，达到第一门研究生课程的水平：经验风险最小化、随机梯度下降、过拟合、偏差—方差分解、Adam。
- **面向序列的深度学习**：Transformer、注意力、自回归分解、教师强制、分词。第2章会恰好复习所需的内容，不多不少。
- **强化学习**：有帮助，但不是必需。第2章与 第6章从零开始建立 MDP、策略梯度、演员—评论家方法与信赖域，因为语言模型这一场景足够特殊（动力学确定、动作空间巨大、每个回合只有一个奖励），以至于标准的 RL 直觉常常会把人引入歧途。
- **Python 与 PyTorch**，用于完成习题。

修过标准 RL 课程的读者会觉得第6章有一部分似曾相识，但仍不应跳过节 2.3，那一节讲的是语言模型 MDP 在哪些方面是退化的。

对来源的致谢

本书是对大量研究者公开工作的综合。开放权重的模型技术报告——DeepSeek、Qwen、Kimi、MiniMax、GLM、OLMo、Tülu、Llama-Nemotron、Magistral——尤其值得致谢：它们公布的不只是结果，还有配方，从而把后训练从一堆商业机密转变成了一门可以讲授的科学文献。开源训练框架，特别是 `verl`、`OpenRLHF`、`TRL` 和 `NeMo-RL`，在实现层面扮演了同样的角色。

凡我引用一个数字，那都是原作者所报告的数字。凡我陈述一种机制，我都尽力引用那篇把它分离出来的论文。综合、取舍与遗漏上的错误，责任在我。

2026 年 *7* 月 *31* 日

如何使用本书

结构

全书分为六个部分和四个附录。

第一部分——基础（Chapters 1 to 4）
确立问题。后训练是什么，它为何存在，以及预训练模型给了我们什么可供加工的东西。这一部分建立全书通用的记号：把语言模型看作策略，把提示—回复对看作一个回合。它认真对待监督微调——把它当作行为克隆，连同由此带来的全部分布偏移病理——并把数据当作研究对象，而不是当作既定条件。

第二部分——从反馈中对齐（Chapters 5 to 9）
经典的对齐技术栈及其现代后裔。从 Bradley–Terry 模型到生成式与基于评分细则的奖励模型的奖励建模；专门针对语言的策略梯度理论；完整的基于 PPO 的 RLHF 流水线及其实现细节；被统一表示为一个参数化对象的直接对齐族（DPO 及其大约二十个亲属）；以及弥合离线—在线差距的在线、博弈论和 AI 反馈扩展。

第三部分——推理（Chapters 10 to 15）
测试时计算是待解释的现象；针对可验证奖励的 RL 是训练出它的方法；组相对策略优化及其各种变体的详细解剖；信用分配与过程监督；专门用一章讨论「RL 究竟改变了模型的什么」这一经验问题；以及把可验证奖励方法推广到没有验证器的领域。

第四部分——智能体（Chapters 16 to 20）
多轮的、使用工具的、与环境交互的策略。POMDP 形式化以及为什么上下文不等于状态；工具集成的 RL 以及它所要求的掩码规范；长时域信用分配与记忆；环境与任务构建，它正日益成为瓶颈；以及多智能体与自博弈训练。

第五部分——系统（Chapters 21 to 23）
这一切在集群上究竟是怎么跑起来的。框架架构、并行化、异步与离策略流水线；采

样与训练之间的数值失配及其修正；以及决定一次实验花掉一千还是十万 GPU 小时的效率技术。

第六部分——测量、安全与前沿（Chapters 24 to 26）
如何判断上述工作是否奏效；作为技术子问题的安全对齐；以及对尚未解决之处的一份诚实交代。

附录部分包含数学背景（Appendix A）、从正文中移出的长推导（Appendix B）、核心算法的参考实现（Appendix C）以及术语表（Appendix D）。

依赖结构

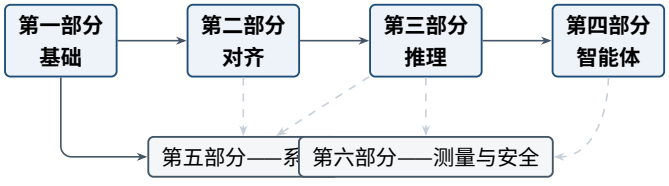


图 1: 阅读依赖关系。实线箭头表示前置要求；虚线箭头表示后面的材料会用到前面的内容，但也可以独立阅读。第五部分可以在读完第6章之后的任何时候读，若你打算亲自跑实验，值得尽早读。

严格必需的主干是 第2章 → 第6章 → 第7章 → 第11章 → 第12章。其余内容都挂在这条主干上。

三种课程方案

A. 一学期的对齐与推理课程（13 周）

周次	材料	重点
1	Chapters 1 and 2	问题框架；语言模型作为策略；KL 估计量
2	第3章	SFT 作为行为克隆；PEFT
3	第4章	数据合成与污染
4	第5章	Bradley–Terry；奖励作弊
5	第6章	从 REINFORCE → PPO 的推导
6	第7章	完整流水线；实现细节
7	第8章	DPO 族；统一视角
8	第9章	在线／迭代式；Nash；RLAIF
9	第10章	测试时计算；搜索
10	第11章	RLVR；R1；GRPO
11	第12章	变体与病理
12	Chapters 13 and 14	过程监督；RL 教会了什么
13	Chapters 24 and 25	测量；安全

B. 一个学季的智能体 RL 高级研讨课（10 周）

假定已掌握第二部分。第 1–2 周读Chapters 11 and 12，随后以每两周一章的节奏读第四部分并配合论文报告，其间穿插Chapters 21 and 22。研讨课的课程项目应当是在一个小型可验证环境上训练一个会使用工具的策略；Appendix C给出了起点。

C. 从业者的自学方案（四周，业余时间）

第 1 周：Chapters 1, 3 and 4外加第24章，因为没有评测，其他一切都无从度量。第 2 周：Chapters 6 and 8——足够读懂代码的理论。第 3 周：Chapters 11, 12 and 22，这三章解释了生产环境中大部分 RL 失败的原因。第 4 周：在 Chapters 17, 18, 21 and 23中挑与你的问题相符的来读。

记号与约定

记号表紧接本章之后，值得加个书签。有两条约定值得立刻点明，因为它们是阅读文献时大多数困惑的来源：

1. **永远有三个策略**。π_θ 是正在被优化的策略，$\pi_{\theta_{\text{old}}}$ 是生成了当前这批数据的策略，π_{ref} 是用于正则化的冻结锚点。在只做一个内层 epoch 的同策略 PPO 中，这三者是重合的；但现代实践中几乎没有什么是那种意义上的同策略，而把它们混为一谈是最常见的一类错误。第四个策略 π_{inf} 出现在 第22章中：它是推理引擎实际实现的那个策略，与 $\pi_{\theta_{\text{old}}}$ 的差别可能只在浮点数的最后几位，有时却大得多。
2. **词元下标与样本上标**。圆括号上标索引组内的样本，$y^{(i)}$；下标索引序列内的词元，y_t。因此像 $\rho_{i,t}$ 这样的量，就是样本 i 的第 t 个词元的重要性比值。

引用采用作者—年份格式。形如「(DeepSeek-AI, 2025)」的引用指向参考文献；其中包含 arXiv 编号，以便每一项论断都能对照原始来源核查。

记号

序列与分布

$\mathcal{V}$	词元词表；$\lvert\mathcal{V}\rvert$ 一般在 10^5–10^6 量级。
$x \in \mathcal{X}$	一个提示（上下文、查询、任务说明）。
$y \in \mathcal{Y}$	一个回复：有限词元序列 $(y_1, \dots, y_T)$，其中 $y_t \in \mathcal{V}$。
y_t	y 的第 t 个词元。
$y_{<t}$	前缀 $(y_1, \dots, y_{t-1})$。
$\lvert y\rvert$	以词元计的回复长度，也写作 T。
$y^{(i)}$	从同一个提示采出的 G 个样本构成的组中的第 i 个回复。
y^+, y^-	被偏好（选中）的回复与不被偏好（拒绝）的回复。
c, a	一个回复中的推理轨迹与最终答案，$y = (c, a)$。
$\mathcal{D}$	一个数据集；$\mathcal{D}_{\text{sft}}$、$\mathcal{D}_{\text{pref}}$、$\mathcal{D}_{\text{rl}}$ 表示按用途区分的变体。

策略

π_θ	正在被优化的策略，参数为 $\theta \in \Theta$。
$\pi_{\theta_{\text{old}}}$	产生当前这批数据的行为策略。只有在第一个内层梯度步时才等于 π_θ。
π_{ref}	冻结的参考策略，用作 KL 锚点。通常是 SFT 检查点。
π_{sft}	监督微调后的检查点。
π_{base}	预训练基座模型，尚未经过任何后训练。
π_{inf}	采样时所用的推理引擎实际实现出来的策略（第22章）。
$\pi^\star$	给定目标下的一个最优策略。
$\pi_\theta(y \mid x)$	$= \prod_{t=1}^{\lvert y \rvert} \pi_\theta(y_t \mid x, y_{<t})$，即自回归分解。
z_t	位置 t 处 softmax 之前的 logits；$\pi_\theta(\cdot \mid \cdot) = \text{softmax}(z_t/\mathcal{T})$，其中 $\mathcal{T}$ 为温度。

奖励、值与优势

$r^\star$	（不可观测的）真实奖励：我们真正想要的东西。
r_ϕ	参数为 ϕ 的学习得到的奖励模型。
r_{ver}	验证器奖励，由程序而非模型计算得到。
$r(x, y)$	一个完整回复的终局奖励。本书中几乎所有奖励都是终局奖励。
r_t	逐词元奖励。在标准构造中，$t < T$ 时 $r_t = 0$，而 $r_T = r(x, y)$，可能再减去一项逐词元的 KL 惩罚。
$V^{\pi_\theta}(s)$	策略 π_θ 的状态值；s 是提示加前缀。
V_ψ	参数为 ψ 的学习得到的评论家（值网络）。
$A^{\pi_\theta}(s, a)$	优势 $Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$。
$A_{i,t}$	由所用的估计量赋给样本 i 的第 t 个词元的优势。
G	回报，$\sum_t \gamma^t r_t$。本书中除非另有说明，$\gamma = 1$。

目标与优化

$\mathcal{J}(\theta)$	待最大化的目标。
$\mathcal{L}(\theta)$	待最小化的损失；两者同时出现时 $\mathcal{L} = -\mathcal{J}$。
$\rho_{i,t}$	$= \dfrac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, y_{i,<t})}$，词元级重要性比值。
ρ_i	$= \dfrac{\pi_\theta(y^{(i)} \mid x)}{\pi_{\theta_{\text{old}}}(y^{(i)} \mid x)}$，序列级重要性比值。
$\text{clip}(u, a, b)$	$= \min(\max(u, a), b)$。
ε	截断半宽；不对称时写作 $\varepsilon_{\text{low}}, \varepsilon_{\text{high}}$。
β	KL 正则项的系数。
G	组大小：每个提示采样的回复数。
γ, λ	折扣因子与 GAE 混合参数。
$\text{sg}[\cdot]$	停止梯度（`detach`）。

信息论量

$\mathbb{D}_{\text{KL}}(p \Vert q)$	Kullback–Leibler 散度 $\mathbb{E}_{z\sim p}\left[\log \frac{p(z)}{q(z)}\right]$。
$\mathbb{H}[p]$	Shannon 熵。对一个策略而言，$\mathbb{H}[\pi_\theta(\cdot \mid s)] = -\sum_a \pi_\theta(a \mid s) \log \pi_\theta(a \mid s)$。
$\mathbb{D}_f(p \Vert q)$	以 f 为生成元的 f-散度（Appendix A）。
$\chi^2(p \Vert q)$	χ^2 散度。

智能体相关量

$\mathcal{T}$	可用工具的集合；一个工具是从调用字符串到观测的偏函数。
$\mathcal{E}$	一个环境：动力学加奖励。
$o^{(k)}$	环境在第 k 轮返回的观测。它不是由策略生成的，因此要在损失中被掩掉。
$a^{(k)}$	策略在第 k 轮发出的动作（词元块）。
$h^{(k)}$	第 k 轮的上下文：$(x, a^{(1)}, o^{(1)}, \dots, o^{(k-1)})$，可能经过压缩。
K	一个回合中的轮次数。
$\mathcal{M}$	外部记忆或草稿区状态。
τ	一条轨迹 $(h^{(1)}, a^{(1)}, o^{(1)}, \dots)$。

约定

- **目标取最大，损失取最小**。策略梯度的表达式一律写成目标 $\mathcal{J}$ 的形式，并沿 $\nabla_\theta \mathcal{J}$ 上升。代码里最小化的是 $-\mathcal{J}$。
- **不带下标的 $\mathbb{E}$** 表示对所在表达式中一切随机成分取期望，采样分布则在邻近的文字中说明。
- **对词元求和**的范围是 $t = 1, \dots, |y|$，即只对回复求和。提示词元永远不参与损失。
- **归一化因子很重要,因此一律显式写出**。词元级损失究竟是除以 $|y^{(i)}|$、除以 $\sum_i |y^{(i)}|$，还是除以常数 $|y|_{\max}$，是一个实质性的算法选择（节 12.2），所以在分母有歧义之处，本书从不简单写作「mean」。
- **温度**。采样温度写作 $\mathcal{T}$ 而非 T；T 表示序列长度或迭代次数。
- **向量**不加粗，除非与标量的区分确实起作用。

第I部分

基础

第 1 章

后训练范式

核心思想: 后训练是能力的激发与重塑，而不是能力的创造

预训练得到的是一个关于文本的模型。后训练把这个模型转变成一个追求某个目标的智能体。这一转变所消耗的算力只占总量的一小部分——2023 年时远低于 1%，到 2026 年有时已超过一半——却几乎构成了「下一个词元预测器」与「一个可以部署、可以信任、可以按标准问责的系统」之间的全部差别。

1.1 后训练是什么

一个预训练语言模型是词元序列上的条件分布，通过在大规模语料上做极大似然拟合得到。把它记作 $\pi_{\text{base}}(y \mid x)$。它是一个关于「互联网上什么样的文本倾向于接在什么样的文本之后」的极好的模型。但它在任何直接的意义上，都不是一个关于「对这个请求而言，什么才是有帮助的、正确的、安全的回复」的模型。

后训练是这样一族方法的总称：它们把 π_{base} 变成一个针对显式目标优化过的策略 π_θ，而所用的数据和算力比预训练所消耗的要小上几个数量级。从形式上看，本书中的每一种方法都是同一个模板的实例：

$$\max_{\theta} \underbrace{\mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot\mid x)}\big[r(x,y)\big]}_{\text{最大化反馈}} - \beta \underbrace{\mathbb{E}_{x\sim\mathcal{D}}\big[\mathbb{D}_{\mathrm{KL}}(\pi_\theta(\cdot\mid x)\,\|\,\pi_{\text{ref}}(\cdot\mid x))\big]}_{\text{保持贴近先验}}. \tag{1.1}$$

此后几乎所有内容都是(1.1)的某种特例化。差异出现在四个地方，现在就把它们点名是值得的，因为全书余下部分正是围绕它们组织起来的：

1. *r* **从哪里来**。人类偏好模型（第5章）；AI 评判模型（第9章）；一段检查答案的程序（第11章）；一个执行动作的环境（第17章）。每一种来源都有不同的噪声特征、不同的成本，以及不同的、具有各自特点的被钻空子的方式。
2. **期望如何估计**。带重要性加权修正的同策略采样、组相对基线、学习得到的评论家、蒙特卡洛树搜索，或者——在直接对齐这一族中——一个彻底取消采样的闭式解（第8章）。
3. **信用如何分配**。一条 30,000 词元长的轨迹末端的一个标量，必须以某种方式变成作用在各个词元 logits 上的梯度。可选做法的跨度很大，从「给每个词元同样的优势」到由过程奖励模型给出的逐步值估计（第13章）。
4. **「贴近先验」是什么意思**。反向 KL、正向 KL、充当隐式信赖域的截断重要性比值，或者干脆什么都不加。这个选择决定了策略是会坍缩到单一模式、漂移成胡言乱语，还是保持流畅（节 6.5）。

1.1.1 三重差距

把后训练所应对的三种性质不同的缺陷区分开来是有用的，因为它们需要不同的方法，也因为把它们混为一谈会让人抱有错误的期待。

格式差距。 一个基座模型被问到「澳大利亚的首都是哪里?」时，很可能接着写「奥地利的首都是哪里？比利时的首都是哪里?」——这是一份问答测验题目清单的合理续写。它知道答案；它只是不知道自己被提问了。这个差距几乎完全可以靠在几千条示范数据上做监督微调来弥合（第3章）。它是最廉价的差距，也是催生了最初那批「指令微调」文献的差距。

偏好差距。 在模型可能给出的众多格式良好的回复中，有些更有帮助、更诚实、校准更好、结构更清晰，也更不容易造成危害。这些性质对人类来说很容易两两比较，对任何人来说却都很难写进损失函数。这正是奖励建模与偏好优化所应对的差距（Chapters 5 and 8）。它本质上是一个学习目标函数的问题，因而也继承了这么做所带来的全部病理：奖励作弊、谄媚、冗长偏好，以及一般意义上的古德哈特定律。

能力差距。 在一道困难的竞赛数学题上，2024 年一个强基座模型采样一次可能解出 10% 的实例；采样两百次并配上一个完美的答案检查器，则可能解出 70%。能力就在

表 1.1: 三重差距及其各自的典型处理方式。「样本效率」为数量级估计，以获得大部分可得收益所需的标注样本或可验证样本数量来衡量。

差距	症状	方法	信号	样本量
格式	模型继续往下写而不作答	SFT	示范数据	10^3–10^5
偏好	回答无帮助或不安全	RM + RL、DPO	两两比较	10^4–10^6
能力	只是偶尔答对	RLVR	验证器	10^4–10^6
智能体	无法完成多步任务	智能体 RL	环境	10^3–10^5

分布之中，只是无法可靠地取出来。弥合这个差距——把概率质量从「偶尔能产生正确轨迹」挪到「可靠地产生正确轨迹」——正是针对可验证奖励的强化学习所做的事（第11章）。至于它是否还做了更多的事，i.e., 它能否装入基座模型根本不会产生的推理策略，则是这个领域真正开放的问题之一，本书专门用一章来讨论（第14章）。

表 1.1还加了第四行：智能体。到 2026 年，它大概可以算作一重独立的差距了：一个模型可以格式良好、对齐良好，还是个出色的一次性推理者，却仍然无法完成一项涉及工具的五十步任务，因为它训练过程中从来没有什么东西要求它从一次失败的动作中恢复过来。

1.2 预训练为何不够

标准答案——「因为预训练目标不对」——是正确的，但太过轻巧。把机制说精确是值得的，因为精确的版本能解释为什么后训练用那么少的数据就能做到那么好。

1.2.1 极大似然学到的是混合，而不是众数

预训练最小化 $\mathbb{E}_{y\sim p_{\text{data}}}[-\log \pi_{\text{base}}(y)]$，它（相差一个常数）就是正向 KL $\mathbb{D}_{\text{KL}}(p_{\text{data}} \,\|\, \pi_{\text{base}})$。正向 KL 是 覆盖质量型的：它惩罚给数据中出现过的任何东西赋予低概率，而对给数据中不出现的东西赋予中等概率几乎无动于衷。一份既包含专家答案也包含新手答案、既有严谨推理也有草率推理、既有真实论断也有编造论断的语料，训练出来的模型会大致按语料频率把概率质量分配到所有这些东西上。

从这样的模型中采样，得到的是从一个质量参差不齐的人格混合分布中的一次抽样。目标函数里没有任何东西偏爱那个好的成分。后训练归根到底就是挑出并放大这

个混合分布中的某一个成分的操作——这也正是为什么 (1.1)的反向 KL、寻众数的性质不是偶然，而恰恰是要点所在（节 6.5）。

历史注记: 提示工程作为隐式的后训练

在 RLHF 普及之前，从业者是靠手工完成同样的挑选的：在提示前面加上「以下是与一位聪明、乐于助人的 AI 助手的对话」可测量地改善了输出，少样本示例也可测量地改善了格式。这些技巧之所以有效，是因为它们把混合分布往一个更好的成分上做了条件化。后训练做的是同一件事，只不过挪动的是参数而不是词元，因而更可靠、不占用上下文，而且无法被用户撤销。这样看来，「提示工程」和「后训练」是在流水线的不同位置上执行的同一个操作，而历史上从前者向后者的转移，是这个领域在「把成本付在哪里」这个问题上换了答案。

1.2.2 教师强制教不会纠错恢复

预训练目标和 SFT 目标都以真实前缀为条件。而在推理时，模型以自己生成的前缀为条件。由此产生的分布偏移就是模仿学习中经典的曝光偏差，其实际后果是：一个靠模仿训练出来的模型，从未在它实际会访问到的状态分布上被训练过——尤其是，它从未在自己犯错之后所到达的状态上被训练过。

这一点对长输出影响极大。如果模型在一段 5,000 词元的推导的第 500 个词元处犯了一个算术错误，极大似然目标不会施加任何朝向「注意到并改正它」的压力，因为「一位严谨的作者先犯错、再回溯」这样的序列在语料中很罕见。而在模型自己的采样轨迹上训练的强化学习，恰好提供了这种压力：能纠正过来的轨迹拿到奖励，纠正不过来的拿不到。在纯结果奖励的 RL 下自发出现的回溯行为（「等一下，让我重新考虑一下」）(DeepSeek-AI, 2025)，是文献中关于「同策略训练在做某种模仿做不到的事」最清楚的证据。

1.2.3 没有一种似然叫「正确」

最后，有些目标根本无法表述为语料上的似然。「最终答案必须等于 42」是一个关于完整序列的性质，可以由一段程序计算出来，却没有什么直截了当的办法让极大似然去在意它。这正是针对可验证奖励的强化学习背后的观察，值得用最尖锐的形式陈述出来：

核心思想: 验证不对称性

对于范围很大且还在扩大的一类任务，检查一个解要比产生一个解便宜得多。只要这种不对称性成立，我们就能不依赖人力地大规模制造奖励信号。2025 年以后的后训练进展，粗略地看，就是不断寻找更多这种不对称性成立的领域的故事——先是数学，然后是代码，然后是工具使用，然后是软件工程，然后是任何带有程序化成功判据的环境。

由此得到的推论正是这个领域的中心瓶颈：在不对称性不成立的地方——文章质量、研究品味、长时域判断——我们又退回到了昂贵而带噪的人类反馈或模型生成的反馈，进展也慢得多。第15章的很大一部分内容，讲的就是在不对称性并不天然存在的地方设法把它制造出来的种种尝试。

1.3 简史

后训练的思想谱系比大语言模型更久远，了解它可以避免大量的重复发明。

1.3.1 史前史：基于偏好的 RL（2008–2017）

从人类的两两比较而非示范中学习奖励函数，这一做法早于深度学习 (Akrour et al., 2011; Cheng et al., 2011)。决定性的现代化工作是 Christiano et al. (2017)：它用几百到几千次人类比较，通过拟合一个 Bradley–Terry 奖励模型并以策略梯度方法优化它，训练出了 Atari 和 MuJoCo 智能体。2026 年实践中的 RLHF 的每一个结构性要素，在那篇论文里都已具备：两两比较、学习得到的标量奖励、一个 RL 优化器，以及「若允许策略漂移得太远，它就会利用奖励模型的误差」这一观察。Ibarz et al. (2018) 加入了示范数据作为初始化，由此有了今天标准的「先 SFT 后 RL」的顺序。

1.3.2 语言：从摘要到 InstructGPT（2019–2022）

Ziegler et al. (2019) 首次把这套配方用到了语言模型上，并把 KL 惩罚引入奖励，这是整个文献中影响最深远的一个实现细节。Stiennon et al. (2020) 在摘要任务上表明，一个用人类反馈调过的 1.3B 模型比一个 12B 的监督模型更受偏好，从而确立了后训练在算力效率上的论据。Ouyang et al. (2022)（InstructGPT）把这一点推广到开放式指令跟随，形式化了三阶段流水线——SFT、奖励模型、PPO——并报告了那个让后训练成为全行业战略优先事项的发现：人类评分者更偏好一个 1.3B 的 InstructGPT 模型，而

不是 175B 的 GPT-3 基座模型。Bai et al. (2022a)（Anthropic）以显式区分「有用性／无害性」的方式推进了同一项计划，并引入了迭代式的在线数据采集。

1.3.3 配方的规模化与对它的质疑（2022–2024）

三条线并行发展。更便宜的反馈：Constitutional AI (Bai et al., 2022b) 用模型依据成文原则做自我批评，取代了人类的无害性标注；RLAIF (Lee et al., 2023) 则表明 AI 偏好标注可以全面替代人类标注。更便宜的优化：DPO (Rafailov et al., 2023) 注意到 KL 正则化目标存在闭式最优解，而这个解可以反解出用最优策略表示的奖励，从而把整个 RL 问题变成偏好对上的一个分类损失——不需要奖励模型，不需要采样，也不需要评论家。随后跟进了一大族方法（第8章）。更好的理解：Gao et al. (2023a) 把奖励模型的过度优化量化为 KL 距离与 RM 规模的函数，给了这个领域第一条关于「RLHF 何时不再有效」的预测性定律。

到 2024 年年末，对齐领域的状态是：大家在流水线上已有共识，在 DPO 与 PPO 孰优孰劣上争论不休 (Xu et al., 2024d; Ivison et al., 2024)，并且开始担心基于偏好的方法存在天花板——它们让模型变得讨人喜欢，却没有让模型变得更聪明。

1.3.4 推理（2024–2025）

推理这条线是从提示工程那边独立起步的。思维链 (Wei et al., 2022b) 表明，引出中间步骤能提高准确率；自洽性 (Wang et al., 2023a) 表明，对采样出的多条链做多数投票能进一步提高准确率；基于验证器的重排序 (Cobbe et al., 2021) 表明，一个训练好的检查器能挑出好的链。STaR (Zelikman et al., 2022) 通过在按正确性筛选过的自生成链上做微调，把这个环闭合了起来——这是在推理依据上做的期望—最大化，事后看来也是一个最小化的 RLVR 算法。过程监督 (Lightman et al., 2024) 则表明，在重排序上步骤级标注胜过结果级标注。

随后 OpenAI 的 o1 (OpenAI, 2024) 演示了：可以训练一个模型，让它在单个问题上花费大量且可变的测试时计算，准确率随这一计算量平滑上升，而训练机制正是 RL。方法没有公开。四个月后，DeepSeek-R1 (DeepSeek-AI, 2025) 公开了一个确实有效的方法：取一个强基座模型，用 GRPO (Shao et al., 2024) 针对基于规则的「正确性加格式」奖励做优化，于是带自我验证的长思维链就涌现了出来——不需要任何过程监督，不需要值网络，也不需要搜索。Kimi k1.5 (Kimi Team, 2025a) 在同一周内独立报告了可比的结果。

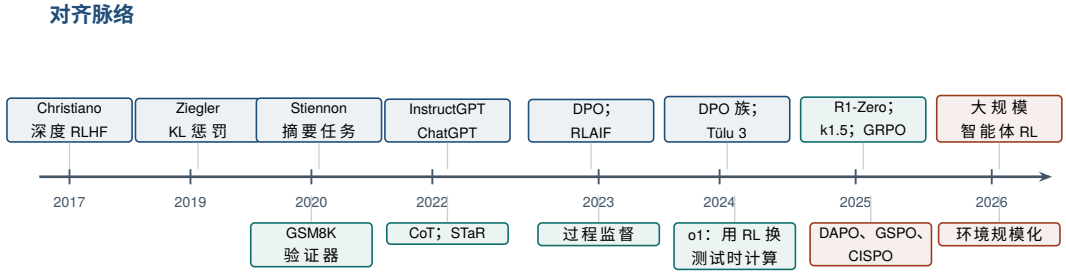


图 1.1: 后训练的两条脉络及其汇流。基于偏好的对齐（轴线上方）与基于验证器的推理（轴线下方）在很大程度上独立发展，直到 2024–2025 年，同一批优化器、同一套基础设施和同一批人开始同时服务于两者。2025 年以后的智能体 RL 需要两边的思想：既需要以环境为依托的可验证奖励，也需要对中间行为的基于偏好的判断。

这对整个领域的影响是立竿见影的。图 1.1勾勒了这一序列。

1.3.5 智能体（2025–2026）

一旦结果奖励的 RL 被确认对单轮推理有效，向多轮工具使用的推广就既迅速又拥挤。Search-R1 (Jin et al., 2025a)、R1-Searcher (Song et al., 2025a)、ToolRL (Qian et al., 2025)、ReTool (Feng et al., 2025a) 以及其他数十项工作确立了基本规范：把生成的词元与工具观测交错排列，把观测从损失中掩掉，对最终结果给予奖励，并用 GRPO 族的目标函数训练。剩下的困难问题——跨轮次的信用分配、长时域上的奖励稀疏、上下文管理、环境供给——构成了Chapters 17 to 19的骨架。

到 2026 年年中，这个领域的重心再次发生了转移：从「用哪个目标函数」转向「用哪些环境」，其理由是 GRPO 各变体之间的算法差异只值几个点，而丰富的、可验证的、长时域环境的可获得性值几十个点。这个看法是否正确，同样仍然开放（第26章）。

1.4 现代流水线的解剖

图 1.2展示的流水线，几乎出现在 2025–2026 年发布的每一份公开配方中 (Lambert et al., 2025; DeepSeek-AI, 2025; Qwen Team, 2025; Kimi Team, 2025b; Zhipu AI and Z.ai, 2025)。这些阶段在实践中都不是可有可无的，尽管它们之间的边界正在变得模糊。

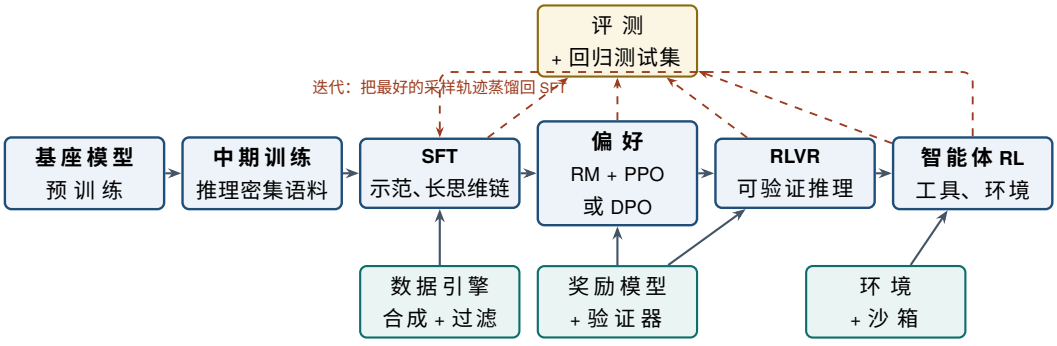


图 1.2: 2026 年的后训练流水线。实线箭头是数据流；虚线箭头是测量与反馈。外层循环——把当前最强策略的最佳采样轨迹蒸馏成下一代的监督数据——是大部分复利式改进的来源，也通常是任何一份配方中记录得最少的部分。

中期训练。 这是介于预训练与 SFT 之间、正日益标准化的一个阶段：在一份为目标能力（数学、代码、推理轨迹、工具调用记录）刻意富集过的语料上继续预训练，往往还带一次长上下文扩展。相对于预训练它很便宜，而且它抬高了后续 RL 所能达到的天花板——RL 无法激发出根本不存在的东西。节 4.7讨论这一点。

SFT。 确立格式、工具调用语法，以及——在推理时代尤为关键的——长时间审慎思考的习惯，通常靠蒸馏一个更强推理模型的轨迹来实现。现代 SFT 混合数据的规模大约在 10^5–10^6 条，覆盖数十个任务类别，并且需要仔细处理损失掩码，以及「哪些样本才值得放进来」这个问题（第4章）。

偏好优化。 修正风格、安全性、拒答行为、指令跟随的精确度，以及其他许多「比较起来容易、验证起来难」的性质。在 2026 年的实践中，出于成本考虑这一步常常用 DPO 族方法，有时是针对奖励模型的在线 RL，也常常两者依次都做。

RLVR。 推理能力真正被建立起来的地方。针对程序化奖励做长时域生成，配合第12章的组相对估计量。这是最吃算力的阶段，也是基础设施质量主导最终结果的阶段。

智能体 RL。 与工具和环境的多轮交互，还要加上 Chapters 17 and 18中的额外机制。

表 1.2: 三种反馈来源的性质。「可作弊性」指策略在不满足真实意图的情况下拿到奖励的容易程度。

	人类	模型（AI）	世界（验证器）
边际成本	\$0.5–\$50 / 条标注	10^{-4}–10^{-2} \$ / 条标注	≈ 仅算力
延迟	数小时至数天	数秒	毫秒至数分钟
吞吐	10^3–10^5 /周	无上限	无上限
噪声	高，且相关	中等，且系统性	低，但脆弱
覆盖范围	任意任务	任意任务	仅可验证任务
可作弊性	中等（谄媚、长度）	高（自我偏好、格式）	低但非零（规范博弈）
失效于	标注者之间的分歧	继承评判模型的盲区	奖励是错误的代理

外层循环。 上述任何一个阶段都不是只做一次。RL 阶段最好的采样轨迹会成为下一轮迭代的 SFT 数据；失败案例会变成有针对性的训练集；评测套件则在每次部署中出问题时增长一次。配方在论文里被线性地描述，在现实中却是循环的。

工程实践: 阶段顺序不是随意的

有两处顺序是关键的。其一，放在长 RLVR 阶段之前的安全与偏好训练往往会被它部分抵消，因为 RLVR 优化的是一个对语气和拒答无动于衷的奖励；因此 2025–2026 年的多数配方要么在 RL 之后再跑一个简短的偏好／安全阶段，要么把安全数据混进 RL 阶段本身。其二，在 RLVR 之前先在长思维链上做 SFT，会大幅降低 RLVR 启动所需的算力，因为策略已经会采出长轨迹，产生学习信号所需的组内方差也立刻就有了。从一个非推理的 SFT 检查点开始跑 RLVR 也行得通——R1-Zero 就是这么做的——但会慢得多，输出也更凌乱。

1.5 三种反馈来源

本书中的每一种奖励都来自三个地方之一，而它们的性质差别之大，足以让这一区分组织起整个设计空间。

这个领域的走向在表 1.2中一目了然：每一列都比它左边的一列更便宜、更快，同时也适用于更少的任务。进展在很大程度上就是把任务从左边推向右边——把一次偏好判断转化为一份可检查的评分细则，把一份可检查的评分细则转化为一个单元测试，

把一个单元测试转化为一个环境。与之相抗衡的压力是：越靠右的列，其质量上限就越取决于规范本身，而一个以足够大的力度去优化某个规范的策略，终将找到规范的缺陷。这是节 5.8的主题，并且它会在本书的每一部分反复出现。

常见陷阱：「可验证」并不意味着「正确」。
一个只检查最终装箱答案字符串是否相等的数学验证器，会奖励那种猜出答案再编一段假推导的模型；一套单元测试会奖励一个专门为测试开特例的补丁；一个「调用了搜索」的奖励会奖励一个调用了搜索却无视结果的策略。可验证奖励消除的是标注噪声，而不是规范错误。与之对应的实践规范——尽早且经常地人工审计采样轨迹——是节 5.8.4 的主题。

1.6 后训练的经济学

后训练在总训练预算中所占的份额，在四年间增长了大约两个数量级，理解其原因能大大说明人们为什么在做他们正在做的事。

对 InstructGPT 而言，RLHF 阶段消耗的算力还不到预训练 FLOPs 的百分之一 (Ouyang et al., 2022)。2026 年的一次推理 RL 运行则完全是另一回事：每一个梯度步都需要为每个提示生成 G 条完整轨迹，每条都可能长达数万词元，而生成是串行的，因而对加速器的利用率很差。对一次 RLVR 步骤做个粗略核算，设批大小为 B 个提示、组大小为 G、平均回复长度为 L、模型规模为 N 个参数：

$$\text{采样 FLOPs} \approx 2\,N\,B\,G\,L \quad \text{(仅前向，带 KV 缓存)}, \tag{1.2}$$

$$\text{训练 FLOPs} \approx 6\,N\,B\,G\,L \quad \text{(前向 + 反向)}, \tag{1.3}$$

$$\text{对数概率重算} \approx 2\,N\,B\,G\,L \times \text{(额外 1 个或 2 个策略)}. \tag{1.4}$$

因此采样只占 *FLOPs* 的少数，却通常占据墙钟时间的多数，常常是 60–80%，因为它受内存带宽约束，而且整个步骤必须等待批中最慢的那条轨迹（节 21.7）。这一个事实就驱动了第21章中的大部分系统工作：异步采样、部分轨迹复用、分离式部署、投机解码，全都是针对同一个瓶颈的攻势。

工程实践：量级预算，2026 年年中

针对一个 30B 量级稠密模型，各阶段一次称职的开源复现的粗略公开数字。请按 ±3× 的精度看待。

- 在 10^6 条、每条 4k 词元的样本上做 SFT：约 10^3 A100/H100 小时。
- 在 10^5 对偏好数据上做 DPO：约 10^2–10^3 GPU 小时。
- PPO 式 RLHF，10^4 步：约 10^4 GPU 小时。
- 回复长 16k 词元、G=16、2k 步的 RLVR：约 10^4–10^5 GPU 小时；这个区间主要由采样效率决定，而不是由算法决定。
- 带真实工具延迟的智能体 RL：除非采样是完全异步的，否则要为环境墙钟时间再加上 2–10×。

实践上的教训是：同一个算法的好实现与坏实现之间的差距，大于不同算法之间的差距。这正是Chapters 21 to 23 存在的理由。

1.7 反复出现的主题

有六个想法出现得如此频繁，值得先命名一次，以后再回头引用。

1. 激发还是习得。 一种后训练方法究竟是让模型能够做某件新事，还是只是让它更可能做它本来偶尔就能做到的事?这一区分是可度量的(比较 pass@1 与大 k 时的 pass@k)，也是有后果的（激发有一个由基座模型设定的天花板，习得则没有）。第14章 专门讨论它。

2. KL 锚点就是全部胜负所在。 对一个学习得到的奖励做无约束的最大化，几百步之内就会毁掉一个语言模型。每一个可用的方法都包含某种机制——显式 KL 惩罚、比值截断、早停，或是充当隐式信赖域的小学习率——把策略保持在奖励仍然具有信息量的区域内。选哪种机制、它与奖励的误差分布如何相互作用，构成了这项工程的大部分内容。

3. 信用分配是核心的算法问题。 一个标量，数万个词元。学习得到的评论家、组相对基线、过程奖励模型、蒙特卡洛值估计、按轮次的分解，都是对这同一个问题的回答，而它们的优劣对比会随时域长度而变化（Chapters 13 and 18）。

4. 估计量中的方差与偏差之争。 组相对基线以很小的偏差换取很大的方差下降；截断以无偏性换取稳定性；截断式重要性采样以一个受控的偏差换取对离策略数据的容忍。第12章中几乎每一个算法「技巧」都是这条权衡曲线上的一个点，按这个视角去读会让相关文献变得清晰得多。

5. 古德哈特定律无法回避，只能管理。 任何代理奖励，只要优化得足够狠，都会不再与目标相关。那些管理手段——集成、KL 约束、在同策略数据上重训奖励模型、评分细则分解、留出的评判模型、人工审计——都是缓解性的，不是根治性的。请假定你写下的每一个奖励都会被钻空子，并预先规划好审计。

6. 分布偏移在每个尺度上都是敌人。 语料与部署之间（预训练 → SFT）；教师强制的前缀与自生成的前缀之间（SFT → 推理）；奖励模型的训练分布与当前策略的输出之间（RM → RL）；推理引擎的数值与训练器的数值之间（采样 → 更新）；训练环境与部署环境之间（智能体 RL → 生产）。本书中大多数有名字的失效模式，都是这五种偏移之一的实例。

1.8 一个完整演算的微型例子

为了在全套机制登场之前先把(1.1)落到实处，我们来看最小的非平凡实例：一个两词元词表和一个赌博机。

设$\mathcal{U} = \{\mathtt{A}, \mathtt{B}\}$,回复都是单个词元,且只有一个提示。一个策略就是一个数$p = \pi_\theta(\mathtt{A})$。假设$r(\mathtt{A}) = 1$、$r(\mathtt{B}) = 0$，参考策略为$\pi_{\text{ref}}(\mathtt{A}) = q$。

目标(1.1)变为

$$\mathcal{J}(p) = p - \beta\Big(p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}\Big). \tag{1.5}$$

令$\mathcal{J}'(p) = 0$得到$1 - \beta \log \frac{p(1-q)}{q(1-p)} = 0$，因此

$$\frac{p^\star}{1-p^\star} = \frac{q}{1-q}\, e^{1/\beta}, \qquad \text{i.e.,} \qquad p^\star \propto q\, e^{r(\mathtt{A})/\beta}. \tag{1.6}$$

从这两行计算中掉出来三条教训，而且三条都能原样推广到完整的序列情形(节 6.4)。

第一,**最优策略是参考策略按奖励做指数倾斜的结果**:$\pi^\star(y\mid x) \propto \pi_{\text{ref}}(y\mid x)\exp(r(x,y)/\beta)$。这是对齐领域最重要的一个公式;DPO 正是通过把它反解出来推导得到的(节 8.1)。

第二,β **在模仿与贪婪最大化之间做插值**:当 $\beta \to \infty$ 时,$p^\star \to q$;当 $\beta \to 0$ 时,$p^\star \to 1$。奖励的尺度与 β 是不可分别辨识的,这就是为什么在实践中奖励归一化与 β 调参是纠缠在一起的。

第三,**支撑集被保留**:若 $q = 0$,则无论奖励如何,$p^\star = 0$。KL 正则化的 RL 无法创造出参考策略赋予零概率的行为。在一个 10^5 路的 softmax 中,没有什么东西的概率恰好为零,所以这是一个关于有效支撑集而非绝对支撑集的陈述——但它是「激发对习得」这场争论在形式上的种子,值得带着 (1.6)一路读到第14章。

本章小结

- 后训练把一个文本模型变成一个目标导向的策略。本书中的每一种方法都是对 KL 正则化奖励最大化模板(1.1)的特例化,差别在于奖励的来源、估计量、信用分配方案以及正则化的形式。
- 有三重不同的差距推动着它:格式(由 SFT 解决)、偏好(奖励模型与偏好优化)与能力(针对可验证奖励的 RL);智能体已经成为第四重。
- 预训练之所以不够,有三个结构性原因:极大似然是覆盖质量型的,因而学到的是一个质量参差不齐的人格混合分布;教师强制从不访问犯错之后所到达的状态;而像「答案是正确的」这类性质无法表述为语料上的似然。
- 验证不对称性——检查比产生更便宜——是 2025 年以来后训练进展的发动机。它能覆盖到多远,决定了哪些能力可以被廉价地训练出来。
- 反馈来自人类、模型或世界,其成本与通用性依次递减,吞吐依次递增。每一种都会以各自特有的方式被钻空子。
- 主导现代 RL 墙钟时间的是采样生成,而不是梯度计算。因此实现质量对结果的影响,足以与算法选择相提并论。
- KL 正则化下的最优策略是参考策略的指数倾斜,$\pi^\star \propto \pi_{\text{ref}} e^{r/\beta}$,它既蕴含了 β 作为「模仿—最大化」旋钮的角色,也蕴含了限定激发所能达到的上限的支撑集保持性质。

延伸阅读

关于基于偏好的 RL 的起源，Christiano et al. (2017) 仍是最清晰的单篇论文，而且篇幅不长。Ouyang et al. (2022) 是经典流水线的标准参考。DeepSeek-AI (2025) 与Kimi Team (2025a) 是理解推理 RL 这一转向必读的两篇；最好对照着读，因为它们做出了不同的选择（GRPO 对上一个带长度惩罚的镜像下降变体）却得到了相似的结论。若想把流水线看作一件所有阶段都有据可查的工程制品，Lambert et al. (2025) 完整得不同寻常，而Lambert (2025) 是一部聚焦对齐、篇幅成书的很好的补充读物。与本章对照着读、颇有反方色彩的一篇是Yue et al. (2025a)，它主张能力差距比看上去更窄；第14章会正式接过这场争论。

习题

1.1（**定量化的混合分布论证。**）假设一份语料由两种人格的回复构成：一位奖励为 1 的专家，出现频率为 α；一位奖励为 0 的新手，出现频率为 $1-\alpha$。一个完美拟合的基座模型以概率 α 采到专家。利用(1.6)，求出使 KL 正则化最优解以概率 0.99 采到专家所需的 β，表示为 α 的函数。讨论 $\alpha \to 0$ 时的行为，并把它与「长尾能力为何难以激发」联系起来。

1.2（**有限精度下的支撑集保持。**）(1.6)说零概率动作会一直停留在零。而真实策略是 bfloat16 下 10^5 个词元上的 softmax。请在一个典型的 logit 分布下，估计单个词元可表示的最小非零概率，并由此估计要观测到它一次所需的采样数。讨论「支撑集保持」对 RL 而言究竟是一条数学约束还是一条经验约束。

1.3（**成本模型。**）利用节 1.6中的 FLOP 核算，计算一次 RLVR 运行的总 FLOPs，其中 $N = 3 \times 10^{10}$、$B = 256$、$G = 16$、平均 $L = 8000$，共 2000 步，并假定 π_θ 与 π_{ref} 的对数概率都要重算。把结果表示为一次 10^{24} FLOPs 预训练运行的比例。然后对 $L = 32000$ 重做一遍，并评论长思维链训练为什么改变了这个领域的经济学。

1.4（**给偏移命名。**）对下列每一种观察到的失效，指出节 1.7中五种分布偏移里主要该由哪一种负责。(a) 一个在静态安全基准上得分很好的模型，很容易被多轮攻击越狱。(b) 一次 RLHF 运行的奖励稳步攀升，人类评测却在下降。(c) 一个用经过验证的沙箱工具训练出来的策略，在生产工具以不同格式返回错误时失败。(d) 一个被微调去写正确代码的模型，在一个长文件的第 3000 个词元处产生了语法错误，此后再也没能恢复。(e) 一次原本稳定的 RL 运行，在推理引擎升级之后变得不稳定。

1.5 ⋆ **（这个模板穷尽了吗?）** 找出一种已发表的后训练方法，它不能自然地用(1.1)表达——e.g., 一种修改采样过程、架构或解码规则而非修改参数的方法——然后论证：要么它可以在恰当选择 r 与 π_{ref} 的前提下被改写进这个模板，要么它不能。best-of-n 采样是一个不错的起点；参见节 10.4.1。

1.6 † **（亲手感受格式差距。）** 取任意一个未经指令微调的预训练基座模型（1–8B 即可）。构造二十个答案简短且无歧义的问题。分别在以下三种条件下测量精确匹配准确率：(a) 光秃秃的问题，(b) 同样格式的 5-shot 提示，(c) 一段「乐于助人的助手」人格前缀。然后在 200 条示范数据上微调并重新测量。报告从 (c) 到微调后的差距中有多少属于格式、有多少属于知识，并说明你是如何区分它们的。

第 2 章

预备知识：作为策略的语言模型

核心思想: 语言模型的 MDP 是退化的，而这些退化恰恰有用

状态转移是确定且已知的；动作空间极其庞大，但其分布是一个可以精确求值的 softmax；奖励只在最后出现一次。这些性质使若干标准的强化学习直觉失效——探索与随机动力学无关、评论家未必必要、折扣因子应当取 1——同时也使一类在一般 MDP 中不可能实现的估计量成为可能。

本章确定全书的记号，并且只发展后文真正会用到的工具：把语言模型视为决策者的两种方式、每一份实现中都会出现的散度估计量、重要性采样及其诊断指标、全书所有基线背后的方差缩减恒等式，以及用于度量这一切的估计量。即使具备扎实强化学习背景的读者，也仍应阅读节 2.3 to 2.5。

2.1 作为条件分布的自回归模型

固定一个由词元（token）构成的词表 $\mathcal{V}$，其规模 $|\mathcal{V}|$ 通常介于 3×10^4 与 10^6 之间。参数为 θ 的模型对任意词元序列 $s\in\mathcal{V}^*$ 给出下一个词元上的分布，方式是对 logits 取 softmax：

$$\pi_\theta(v\,|\,s)=\frac{\exp(z_v(s;\theta)/\mathcal{T})}{\sum_{v'\in\mathcal{V}}\exp(z_{v'}(s;\theta)/\mathcal{T})},\qquad v\in\mathcal{V}, \tag{2.1}$$

其中 $\mathcal{T}>0$ 是采样温度。除非另有说明，训练期间 $\mathcal{T}=1$，并且所有目标函数中出现的都是这一训练期分布；节 22.7讨论了当采样轨迹在 $\mathcal{T}\neq1$ 下生成、却按 $\mathcal{T}=1$ 来打分时会出什么问题。

对提示（prompt）x 的一个完整回复 $y=(y_1,\dots,y_T)$ 获得的概率为

$$\pi_\theta(y\,|\,x)=\prod_{t=1}^{T}\pi_\theta(y_t\,|\,x,y_{<t}), \tag{2.2}$$

而我们几乎总是使用对数概率

$$\log \pi_\theta(y \mid x) = \sum_{t=1}^{T} \log \pi_\theta(y_t \mid x, y_{<t}), \tag{2.3}$$

既因为这才是训练框架实际计算的量，也因为求和在数值上远比 10^4 个因子连乘更稳健。

(2.2)有三个推论值得强调。

序列对数概率随长度增长。 $\log \pi_\theta(y \mid x)$ 是 T 个负项之和，因此更长的序列系统性地具有更低的对数概率。任何跨不同长度的序列比较 $\log \pi_\theta$ 的方法——DPO 正是如此——都内建了长度偏置（节 8.6）。SimPO 采用的长度归一化就是对此的直接回应。

序列级重要性比值具有极大的动态范围。 比值 $\pi_\theta(y \mid x)/\pi_{\theta_{\text{old}}}(y \mid x)$ 是 T 个词元比值的乘积。若每个词元的比值以典型相对幅度 δ 偏离 1，则序列比值的对数标准差约为 $\delta\sqrt{T}$；当 $\delta = 0.01$、$T = 10^4$ 时这相当于 $e^{\pm 1}$ 倍，而 $\delta = 0.05$ 时则为 $e^{\pm 5} \approx 150$ 倍。仅这一条观察，就解释了 第12章中关于重要性比值应当逐词元施加还是逐序列施加的绝大部分分歧。

模型是一棵树上的分布。 状态就是前缀；$\mathcal{V}^*$ 的前缀树不存在合并。这意味着「状态空间」没有别名现象：两个不同的词元序列永远是不同的状态，哪怕它们表达同一个意思。正是函数逼近让跨状态的泛化成为可能，而精确状态重访的缺失，则是表格型强化学习直觉难以迁移过来的原因。

2.1.1 分词是环境的一部分

分词通常被当作预处理，但对后训练而言，它有若干一阶效应值得一次性说清。

- **动作空间是词元，而不是词或步骤。**一个「推理步骤」是几十个动作。因此，若要在人类会认可为语义单元的层面上做信用分配，就必须在词元组上做聚合，这正是过程奖励模型和轮次级分解所做的事（第13章）。
- **按长度做奖励归一化依赖于分词器。**在某个分词器上标定好的长度惩罚，无法迁移到对同一段文本多切出 15% 词元的模型上。长度预算要同时以词元 和字符两种单位报告。

- **特殊词元是后果极大的策略动作**。发出 <eos> 会终止回合；发出工具调用分隔符会把控制权移交给环境。单个 logit 就控制着一个对回报影响巨大的事件，而强化学习一定会发现这一点。截断处理（节 12.7）之所以存在，正是因为策略否则就会在长度上限附近学出病态行为。
- **对话模板构成了训练与部署之间的契约**。角色、轮次边界与推理分隔符的确切字符串格式，必须在 SFT、RL 与服务之间保持一致。不匹配是最常见、也最尴尬的质量回退来源之一，而且它在损失曲线上完全不可见（节 3.4）。

2.2 两种决策过程视角

把文本生成刻画为序贯决策问题有两种标准做法。它们导向不同的算法，而且两者都出现在文献中，常常并不点明其间的区别。

2.2.1 词元级 MDP

定义 MDP $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$，其中

$$\mathcal{S} = \mathcal{X} \times \mathcal{V}^*, \qquad \mathcal{A} = \mathcal{V}, \tag{2.4}$$

$$s_1 = x, \qquad s_{t+1} = s_t \oplus a_t \ \text{（拼接）}, \tag{2.5}$$

其中 $\oplus$ 表示追加一个词元；当 $a_t =$ <eos> 或 t 达到最大长度时回合终止。策略即 (2.1) 给出的 $\pi_\theta(a \mid s)$。奖励通常取

$$r_t = \begin{cases} r(x, y) & t = T\text{（终止）}, \\ 0 & \text{其他}, \end{cases} \tag{2.6}$$

并可能再加上逐词元的 KL 惩罚（节 7.2）。

这一视角支持值函数、时序差分学习与广义优势估计的整套机制，也是基于 PPO 的 RLHF 所依托的视角（第7章）。

2.2.2 序列级老虎机

另一种做法是把整个回复视为单个动作：

$$\mathcal{A} = \mathcal{Y}, \qquad \pi_\theta(y \mid x) \text{ 如(2.2)所示}, \qquad r : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}, \tag{2.7}$$

即一个动作空间庞大而有结构的上下文老虎机。这里没有可供估值的中间状态，因而也就没有评论家；全部的量只有 $\log \pi_\theta(y \mid x)$ 与 $r(x, y)$。这是 REINFORCE 类方法、RLOO、GRPO 以及整个直接对齐家族所依托的视角。

2.2.3 它们是同一个 MDP，只是观察粒度不同

由于状态转移是确定的、奖励是终止型的，两种视角在目标函数上是一致的：

$$\mathcal{J}(\theta) = \mathbb{E}_{x} \mathbb{E}_{y \sim \pi_\theta}[r(x, y)], \tag{2.8}$$

而策略梯度定理在两者下给出相同的梯度。它们的差别在于各自提示了什么样的估计量：

	词元级 MDP	**序列级老虎机**
优势	逐词元，来自评论家或 TD	每条序列一个标量
基线	学习得到的 $V_\psi(s_t)$	组内样本均值
信赖域	逐词元比值截断	逐序列比值截断
显存开销	评论家使优化器状态翻倍	无
偏差风险	评论家偏差、自举误差	不存在来自评论家的偏差
方差风险	评论家拟合良好时更低	更高；需要 $G \geq 4$–64
典型用途	PPO 版 RLHF、稠密奖励	GRPO 系列、RLVR

注记 2.1. 这一选择并不纯粹是口味问题。在只有终止奖励的情况下，用回报回归训练出的评论家必须学会这样一个函数：输入是 10^4 个词元的前缀，目标却是该轨迹最终的二值正确与否——这是一个极难的回归问题，其中的误差会作为偏差施加到每个词元的优势上。组相对基线完全绕开了这个回归问题，代价是生成量增加 G 倍。哪种权衡更划算，取决于生成的成本相对于评论家可拟合程度的高低；而该领域的答案之所以逐渐倒向组相对方法，很大程度上是因为生成变便宜了、时域变长了（节 6.2.4）。

2.3 语言模型 MDP 的特殊之处

标准的强化学习直觉是在具有随机动力学、小动作空间和稠密奖励的问题上形成的。语言模型的场景一条都不满足，其中五点具体差异改变了什么才是成立的。

1. 动力学是确定且已知的。 s_{t+1} 就是一次拼接。没有需要平均掉的环境随机性，没有需要学习的模型，转移中也不存在偶然不确定性。全部随机性都来自策略本身。由此带来两点后果：(i) 从某个状态出发的单次采样就给出精确的下一状态，因此从某个前缀重新采样来做蒙特卡洛值估计是无偏的，而且在「无需环境模拟器」这个意义上是廉价的（VinePPO 正是利用了这一点，节 13.4）；(ii)「探索」指的是策略中的采样多样性，而不是访问不同的世界状态，所以熵的管理 就是探索的管理（节 2.5）。

2. 动作空间极其庞大，但其分布可完全枚举。 我们无法枚举 $|\mathcal{U}|^T$ 条序列，但在每一步上我们都拥有 $|\mathcal{U}|$ 个动作上的精确类别分布。这意味着逐步的熵、逐步的 KL 以及逐步的概率比值都可以精确计算，而不是估计出来——这在连续控制的强化学习中是奢望，而若干算法正是利用了它。

3. 奖励是终止型的、稀疏的，而且往往是二值的。 每个回合只有一个数。在二值奖励与组规模 G 下，若某个提示上的 G 个样本全部成功或全部失败，则在任何组相对估计量下它都提供零梯度——所有优势都为零。这正是 DAPO 中动态采样（节 12.4）背后的观察，也是课程在 RLVR 中居于核心地位的原因：有用的提示是那些成功率处于中间水平的提示，而哪些提示属于这一类，会随着策略的改进而变化。

4. 折扣因子应当取 1。 强化学习中的折扣有两个用途：刻画对更早获得奖励的真实偏好，以及保证无限时域回报的收敛。这两点在此都不适用。回合是有限的，而在第 8000 个词元处给出的正确答案，与在第 200 个词元处给出的价值完全相同（任何对简洁性的偏好都应当在 r 中表达为显式的长度惩罚，那里它可见且可调，而不该借 γ 夹带进来）。在语言模型的 MDP 中设 $\gamma < 1$，会悄无声息地引入一个长度偏置，而它因为不可见反而更糟；请取 $\gamma = 1$，若使用 GAE，除非有特定理由，也一并取 $\lambda = 1$。

5. 策略必须始终是一个好的语言模型。 在普通的强化学习里，任何取得高回报的策略都是可接受的。这里的策略却必须同时在奖励未覆盖的一切方面保持流畅、连贯与良好校准——而这些性质奖励并不度量，无约束的优化则会摧毁它们。这正是要有 KL 锚的原因，也是为什么「奖励上去了」永远不足以证明训练是成功的。

常见陷阱: 照搬控制类强化学习的超参数

在 MuJoCo 中合理的取值，在这里往往是错的。在 10^4 个词元长的回合上取 $\gamma = 0.99$，会把终止奖励折扣 e^{-100} 倍，使学习信号消失。在同样的时域上取 $\lambda = 0.95$ 的 GAE，会把信用截断在约 20 个词元的有效窗口内。能稳定 Atari 智能体的熵奖励系数（10^{-2}）会在一百步之内摧毁语言模型的分布。真正可用的默认配置是 $\gamma = \lambda = 1$、不加熵奖励，改用 KL 锚。

2.4 散度及其估计量

KL 散度以三种互不相同的角色出现，把它们混为一谈是一个常见错误。

1. 作为目标函数中的**正则项**，使 π_θ 保持在 π_{ref} 附近。
2. 作为**诊断量**，度量训练已经漂移了多远。
3. 作为**隐式代价**，例如 best-of-n 相对于基础策略的 KL 这类结果（节 10.4.1）。

2.4.1 前向与反向

对同一空间上的分布 p, q，

$$\mathbb{D}_{\text{KL}}(p \,\|\, q) = \mathop{\mathbb{E}}_{z\sim p}[\log \tfrac{p(z)}{q(z)}] \geq 0, \tag{2.9}$$

当且仅当 $p = q$ 几乎处处成立时取等号。它不对称，而这种不对称在行为上意义重大：

- $\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})$（**反向 KL**，样本取自可训练模型）是 寻模的。若 π_θ 把概率质量放在 π_{ref} 没有质量的地方，它就是无穷大，因此它禁止策略发明参考分布支撑集之外的行为；但它并不介意策略放弃参考分布支撑集的大片区域。这正是对齐中我们想要的：集中到好的模式上。
- $\mathbb{D}_{\text{KL}}(\pi_{\text{ref}} \,\|\, \pi_\theta)$（**前向 KL**，样本取自参考模型）是 覆盖质量的。它惩罚丢弃参考分布的任何一部分行为。这是极大似然预训练所最小化的量，而对于对齐它是错误的正则项，因为它会主动抵抗我们正努力实现的那种集中。

核心思想: 对齐是寻模的，预训练是覆盖质量的

从前向 KL 切换到反向 KL，就是从模仿切换到对齐这件事的数学内容。它解释了为什么对齐后的模型比基础模型更缺乏多样性（节 14.3）——那不是实现中的缺陷，而是目标函数在忠实执行我们提出的要求。

2.4.2 反向 KL 的三种估计量

实践中我们需要用来自 $y \sim \pi_\theta$ 的样本估计 $\mathbb{D}_{\mathrm{KL}}(\pi_\theta(\cdot|x)\|\pi_{\mathrm{ref}}(\cdot|x))$。令

$$\rho = \frac{\pi_{\mathrm{ref}}(y|x)}{\pi_\theta(y|x)}, \qquad \log\rho = \log\pi_{\mathrm{ref}} - \log\pi_\theta, \tag{2.10}$$

从而 $\mathbb{D}_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}}) = \mathbb{E}_{y\sim\pi_\theta}[-\log\rho]$。常用的估计量有三种，它们都可追溯到Schulman (2020) 那篇被广泛引用的笔记：

$$k_1 = -\log\rho, \qquad \text{无偏；方差大；可能为负,} \tag{2.11}$$

$$k_2 = \tfrac{1}{2}(\log\rho)^2, \qquad \text{有偏；方差小；恒} \geq 0, \tag{2.12}$$

$$k_3 = \rho - 1 - \log\rho, \qquad \text{无偏且恒} \geq 0. \tag{2.13}$$

推导: 为什么 k_3 既无偏又非负

无偏性：$\mathbb{E}_{y\sim\pi_\theta}[\rho] = \sum_y \pi_\theta \cdot \frac{\pi_{\mathrm{ref}}}{\pi_\theta} = 1$，故 $\mathbb{E}[\rho-1] = 0$，从而 $\mathbb{E}[k_3] = \mathbb{E}[-\log\rho] = \mathbb{D}_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}})$。非负性：对一切 $u > 0$ 有 $u - 1 - \log u \geq 0$，且仅在 $u = 1$ 处取等号，因为该函数是凸的且最小值就在那里。于是 k_3 把 k_1 的无偏性与 k_2 的符号保证结合了起来；它其实就是以已知均值的量 $\rho - 1$ 为控制变量、施加在 k_1 上的控制变量法。只要 $\mathrm{Cov}(-\log\rho, \rho-1) > 0$，它的方差就低于 k_1，而这在我们关心的区间内成立。

工程实践: 何处该用哪个估计量

- **用在损失里**：逐词元的 k_3 是现代默认选择（`verl` 与 `OpenRLHF` 正是以 `low_var_kl` 之名实现它）。在损失中使用 k_1 也很常见，而且由于它是对数概率之差，其梯度是最直截了当的那一个；但要注意，通常实现的 k_1 梯度并不是 KL 的梯度，因为采样分布本身依赖于 θ（见Appendix B）。
- **作为日志中的诊断量**：用 k_3，因为仪表盘上出现负的 KL 会让人困惑，而 k_1 会经常产生负值。
- **绝不要**在一个你打算解释为 KL 惩罚的损失中使用 k_2；它的偏差是 $O(\mathbb{D}_{\mathrm{KL}}^2)$，因此恰恰在你最需要惩罚的时候惩罚不足。
- **序列级还是词元级**。整条序列上的 $\mathbb{D}_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}})$，在 π_θ 下取期望时等于逐词元 KL 之和。逐词元施加惩罚再求和是标准做法，它给出稠密的逐词元梯度，也正是(7.1)所假定的形式。

2.4.3 f 散度简述

若干算法把 KL 换成另一种 f 散度，即对满足 $f(1) = 0$ 的凸函数 f 取 $\mathbb{D}_f(p\|q) = \mathbb{E}_q[f(p/q)]$。后文会出现的几种情形如下：

散度	$f(u)$	行为	出现之处
反向 KL	$u\log u$	寻模	标准 RLHF
前向 KL	$-\log u$	覆盖质量	SFT；部分 DPO 变体
χ^2	$(u-1)^2$	对比值尖峰重罚	悲观离线 RL
JSD	$u\log u-(u{+}1)\log\frac{u+1}{2}$	有界、对称	f-DPO
α 散度	$\frac{u^\alpha-u}{\alpha(\alpha-1)}$	可插值	f-DPO、α-DPO

这一脉工作 (Wang et al., 2024a) 给出的一般教训是：散度的选择对多样性—质量权衡的控制，比系数的选择更为直接——在平均偏离相同的条件下，尾部更重的惩罚会保留参考分布支撑集中更多的部分。

2.5 熵，以及它为何会坍缩

某个状态处的策略熵为

$$\mathbb{H}[\pi_\theta(\cdot\,|\,s)] = -\sum_{v\in\mathcal{V}} \pi_\theta(v\,|\,s)\log\pi_\theta(v\,|\,s), \tag{2.14}$$

而它在策略所访问状态上的平均值，是强化学习训练中最值得记录的单个标量。它可以由 logits 精确算出。

在奖励最大化之下，熵会下降。这并非偶然：目标 $\mathbb{E}_{\pi_\theta}[r]$ 关于策略分布是线性的，因此在任意固定状态上，它的最大值点是集中于最佳动作的点质量。若没有反向力量，策略就会变得确定性，届时：

- 组内所有采样得到的回复都变得完全相同；
- 组相对优势精确地变为零；
- 梯度消失、训练停滞，而且通常奖励会冻结在低于高熵策略所曾达到的水平；
- 即便 pass@1 有所提升，$k > 1$ 时的 pass@k 也会急剧退化（节 14.2）。

这一演变过程——熵坍缩——是 RLVR 训练最常见的失效模式，节 12.6对其有详尽讨论。从经验上看，长时间 RL 训练中的策略熵在早期常常大致按步数指数衰减，而

2025 年的论文报告说，可达到的奖励可以由已消耗的熵预测出来 (Cui et al., 2025b)；其实践含义是：熵是一种预算，用来换取奖励，而一旦花掉就极难恢复。

常见陷阱: 用熵奖励修复熵坍缩

在目标中加上 $+\alpha\mathbb{H}[\pi_\theta]$ 是教科书式的修法，在这里效果很差。由于该项在 10^5 个词元上的均匀分布处取得最大，任何大到足以阻止坍缩的 α，都会同时把概率质量注入到明显错误的词元上，损害流畅性；而任何小到足以保住流畅性的 α，只能推迟坍缩而不能阻止它。带熵奖励的训练常常呈现出先平坦、随后爆炸的熵曲线。更可取的替代方案——截断的非对称化、对高协方差词元的选择性梯度掩码、KL 锚定，以及数据课程——在节 12.6中讨论。

2.6 重要性采样与离策略程度

任何实用的语言模型强化学习系统，在某种程度上都是离策略的：一次梯度步中的数据由 $\pi_{\theta_{\text{old}}}$ 生成（可能是若干个优化器步之前、由不同的引擎、在不同的温度下生成的），而梯度却是在 π_θ 处求取的。重要性采样正是处理这一落差的手段。

对任意函数 g，

$$\mathop{\mathbb{E}}_{y\sim\pi_\theta}[g(y)] = \mathop{\mathbb{E}}_{y\sim\pi_{\theta_{\text{old}}}}[\rho(y)\,g(y)], \qquad \rho(y) = \frac{\pi_\theta(y\mid x)}{\pi_{\theta_{\text{old}}}(y\mid x)}, \tag{2.15}$$

只要 $\pi_{\theta_{\text{old}}}$ 在 π_θ 有支撑的地方都有支撑，该式就成立。这个估计量是无偏的，但它的方差受比值二阶矩的控制：

$$\mathop{\mathbb{V}\text{ar}}_{\pi_{\theta_{\text{old}}}}[\rho\, g] \le \mathop{\mathbb{E}}_{\pi_{\theta_{\text{old}}}}[\rho^2 g^2] = \left(1+\chi^2\left(\pi_\theta \,\middle\|\, \pi_{\theta_{\text{old}}}\right)\right)\,\mathbb{E}[\cdot], \tag{2.16}$$

它只有在 $\chi^2\left(\pi_\theta \,\middle\|\, \pi_{\theta_{\text{old}}}\right) < \infty$ 时才有限——这个要求比 KL 有限强得多。这就是为什么用原始重要性权重做离策略校正在实践中会失败，也是为什么每一种可用的方法都会对权重做截尾、截断或以其他方式引入偏差的形式化原因。

2.6.1 诊断指标

有效样本量。 对权重 $w_i = \rho(y^{(i)})$，

$$\text{ESS} = \frac{\left(\sum_{i=1}^{n} w_i\right)^2}{\sum_{i=1}^{n} w_i^2} \;\in\; [1, n]. \tag{2.17}$$

ESS/n 接近 1 表示这一批数据实际上是同策略的；一旦低于约 0.3，估计量就由少数几个样本主导，更新不再可靠。记录 ESS 不花任何代价，却能诊断出那些本来完全不可见的陈旧度问题。

比值分位数。 记录词元级对数比值的 $\{1, 50, 99\}$ 分位数，以及触及截断边界的词元占比。截断比例升到百分之几以上，就意味着信赖域在承担大部分工作，有效学习率已不再是你所设定的那个值。

2.6.2 截断及其偏差

三种标准修正，严重程度依次递增：

$$\text{截断（clip）：}\quad \tilde{w} = \mathrm{clip}(w, 1-\varepsilon, 1+\varepsilon), \quad \text{用于 PPO 的替代目标，} \tag{2.18}$$

$$\text{截尾（TIS）：}\quad \tilde{w} = \min(w, C), \quad \text{限制方差，使估计偏向} \pi_{\theta_{\text{old}}}, \tag{2.19}$$

$$\text{掩码（mask）：}\quad \tilde{w} = w\,\mathbb{1}[w \le C], \quad \text{丢弃离群值，而非压缩它们。} \tag{2.20}$$

上限 C 取 2–10 量级的截尾重要性采样(2.19)是 第22章的主力工具，那里用它来校正推理引擎分布与训练器分布之间的失配。它引入的偏差是 $\mathbb{E}[(w - \min(w, C))g]$，集中在那些当前策略认为远比行为策略更可能的样本上——而依 g 的符号不同，这些恰恰可能正是你最想从中学习的样本。这里没有免费的午餐，只有一桩已被充分理解的交易。

2.7 蒙特卡洛估计与基线

支撑本书中每一个基线、每一个评论家、每一种组相对方案的，是这样一条代数恒等式：从奖励中减去任何不依赖于所采样动作的量，都不会破坏梯度的无偏性。

引理 2.2 (基线不变性)**.** 对任意仅是提示 x 的函数、而不依赖于 y 的 b，

$$\mathop{\mathbb{E}}_{y\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(y\,|\,x)\,\big(r(x,y) - b(x)\big)\right] = \nabla_\theta \mathop{\mathbb{E}}_{y\sim\pi_\theta}[r(x,y)]. \tag{2.21}$$

证明. $\mathbb{E}_{y\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(y\,|\,x)] = \sum_y \pi_\theta \frac{\nabla_\theta \pi_\theta}{\pi_\theta} = \nabla_\theta \sum_y \pi_\theta = \nabla_\theta 1 = 0$。乘以常数 $b(x)$ 再相减，在期望意义下不改变任何东西。剩下的一项由似然比恒等式给出，正是策略梯度。 □

然而方差确实会改变，而最优选择是 $b^\star(x) \approx \mathbb{E}_{y\sim\pi_\theta}[r(x,y)]$（严格地说，它是一个以 $\|\nabla \log \pi_\theta\|^2$ 为权重的均值）。得到它有三种方式：

1. **学习一个评论家**：用回归拟合 $V_\psi(x) \approx \mathbb{E}_{\pi_\theta}[r]$。采样时开销小，但需要第二个网络，且拟合不佳时会引入偏差。
2. **组均值**：采样 G 个回复并取 $\hat{b} = \frac{1}{G}\sum_i r_i$。在极限意义下无偏，不需要额外网络，代价是生成量变为 G 倍。注意，用同一批样本同时计算基线与梯度会引入一个 $O(1/G)$ 的小偏差；留一法校正可以消除它（节 6.2.3）。
3. **留一法（RLOO）**：$\hat{b}_i = \frac{1}{G-1}\sum_{j\neq i} r_j$，它恢复了严格的无偏性。

推导: 留一法严格无偏，组均值则不然

取 $\hat{b} = \frac{1}{G}\sum_j r_j$ 时，样本 i 对应的项是 $\nabla \log \pi_\theta(y^{(i)})(r_i - \frac{1}{G}\sum_j r_j)$。其中 $j = i$ 的那一份把 $\nabla \log \pi_\theta(y^{(i)})$ 与 r_i 耦合了起来，因此一般有 $\mathbb{E}[\nabla \log \pi_\theta(y^{(i)}) \cdot \frac{1}{G} r_i] \neq 0$，留下一个 $1/G$ 量级的偏差，其效果是把有效优势缩小为原来的 $(1 - 1/G)$ 倍。排除 $j = i$ 后，在给定 x 时 $\hat{b}_i$ 与 $y^{(i)}$ 独立，定理 2.2便严格适用。实践中当 $G \geq 8$ 时，两者之差不过是对优势做了 $\leq 12\%$ 的重新缩放，通常会被学习率吸收——这也是组均值虽然在技术上是错的、却依然流行的原因。

2.8 度量策略：pass@k 及其同类

由于回复是采样得到的，任何质量度量都是策略输出分布的一个泛函，而我们选择哪个泛函，就决定了什么才算作改进。

设 x 是一个带验证器的问题，并设 c 为 n 个 i.i.d. 样本中正确回复的数量。于是：

$$\widehat{\text{avg@}n} = \frac{c}{n} \qquad \text{（也写作 pass@1；是 } \mathbb{P}[\text{correct}] \text{ 的无偏估计），} \tag{2.22}$$

$$\widehat{\text{pass@}k} = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \qquad \text{对} k \leq n \text{无偏，} \tag{2.23}$$

$$\widehat{\text{maj@}k} = \mathbb{1}[k \text{ 个样本中的多数为正确}] \quad \text{（自洽性；需要单独做平均）．} \tag{2.24}$$

估计量(2.23)出自Chen et al. (2021)，是标准做法，应当始终优先于朴素的「采 k 个样本，看是否有一个正确」；后者在 n 较小时既方差大，又在选择性汇报时具有误导性。

核心思想: pass@1 与 pass@k 可能反向移动

奖励最大化会使策略集中。集中会提高 pass@1，却可能降低大 k 时的 pass@k，因为 pass@k 奖励的是支撑集中任何位置存在一条正确轨迹。因此只报告 pass@1 会系统性地夸大 RL 的成效；而只报告大 k 的 pass@k 又会系统性地低估它。两者都必须报告，而它们的分歧正是「激发还是习得」之争中的核心证据（第14章）。

工程实践: 能支撑结论的样本量

对一个含 30 道题的基准（AIME 规模），单次贪心解码的标准误约为 $\sqrt{0.5 \cdot 0.5/30} \approx 9$ 个百分点——比多数已报告的改进幅度还大。请对每道题使用 $n \geq 16$ 个样本，并报告带标准误的 avg@n；节 24.2给出了跨题目与跨样本的方差分解。2025—2026 年的大量文献所报告的差异，是其样本量无法支撑的。

2.9 优化札记

关于所用优化器的若干事实，之所以在此列出，是因为它们与 RL 目标的相互作用常常出乎人们意料。

Adam 的尺度不变性掩盖了奖励尺度。 Adam 会按梯度幅值的滑动估计做归一化，因此把所有优势同乘一个常数，在几步之后几乎不改变更新。这就是为什么奖励的 缩放不如奖励的形状（哪些样本相对更好）重要，也是为什么在一个批次内标准化优势所带来的影响不只是尺度问题——它改变了不同提示之间的相对权重（节 12.3）。

梯度裁剪与序列长度相互作用。 如果损失是对词元求和，长序列就会产生更大的梯度，并在被裁剪的更新中占据主导。如果损失是在序列内部对词元求平均，则每条序列无论长短都贡献相同。两者都不是显然正确的，这一选择是一项实质性的算法决策，将在节 12.2中加以分析。

学习率很小，原因在于 KL。 对 7–70B 规模的模型，典型的 RL 学习率是 10^{-6} 到 5×10^{-7}，比 SFT 低一到两个数量级。部分原因是梯度更嘈杂，但另一部分原因是学习率充当了隐式的信赖域：只要步长足够小，无论损失怎么说，策略都无法离 $\pi_{\theta_{\text{old}}}$ 太远。在 10^{-6} 下稳定、在 5×10^{-6} 下发散的训练，通常遇到的不是优化问题，而是 KL 问题。

数值精度不是细节。 单个词元的对数概率，其计算精度必须足以让两个近乎相同的模型之间的差值仍然有意义。若 π_θ 与 $\pi_{\theta_{\text{old}}}$ 都以 bfloat16 计算，则比值 $\pi_\theta/\pi_{\theta_{\text{old}}}$ 的相对误差在 10^{-2} 量级——与信号本身相当。以 bfloat16 计算 logits、但以 float32 计算 log-softmax 与比值，是标准且必要的做法；第22章展示了不这样做会发生什么。

本章小结

- 语言模型是一个按自回归方式分解的策略 $\pi_\theta(y \mid x)$。序列对数概率随长度增长，而序列重要性比值的动态范围随长度指数增长；这两个事实驱动了后文的许多设计决策。
- 两种视角并存：词元级 MDP（支持评论家、TD、GAE）与序列级老虎机（支持组基线，不需要评论家）。它们共享同一个目标和同一个梯度，却提示了不同的估计量。
- 语言模型的 MDP 在五个有用的方面是退化的：确定且已知的动力学、可精确计算的动作分布、终止型稀疏奖励、没有理由做折扣，以及策略必须保持为一个好语言模型这一硬约束。控制类强化学习的超参数无法迁移过来。
- 反向 KL 是寻模的，是对齐的正确正则项；前向 KL 是覆盖质量的，是预训练所最小化的对象。请用 $k_3 = \rho - 1 - \log \rho$ 来估计反向 KL，它既无偏又非负。
- 熵是一种预算。奖励最大化会消耗它，而在缺乏显式机制时，坍缩是默认结局。在这一场景中，熵奖励是一种糟糕的机制。
- 用重要性权重做离策略校正要求 χ^2 散度有限，而不只是 KL 有限；因此所有实用方法都会截断、截尾或掩码。请记录 ESS 与截断比例。
- 任何依赖于提示的基线都保持无偏性（定理 2.2）。留一法严格无偏；组均值带有 $O(1/G)$ 的偏差，等价于做了 $(1 - 1/G)$ 的重新缩放。
- 在 RL 下 pass@1 与 pass@k 可能反向移动；请使用无偏估计量 (2.23)并同时报告两者。

延伸阅读

Sutton and Barto (2018) 是标准的强化学习参考书；相关的是第 13 章（策略梯度）与第 5 章（蒙特卡洛），阅读时不妨把节 2.3当作一份「哪些内容无法迁移」的清单放在手边。Schulman (2020) 只有两页，是(2.11)–(2.13)的出处；值得读原文。Owen (2013)

第 9–10 章是重要性采样诊断（包括 ESS）的参考文献。Chen et al. (2021) 的附录 A 引入了 pass@k 估计量。专门就熵的动力学而言，Cui et al. (2025b) 是最系统的研究，节 12.6对其有详细讨论。

习题

2.1（**序列比值的动态范围。**）假设词元级对数比值 $\log \rho_t$ 是均值为 0、标准差为 σ 的 i.i.d. 变量。推导长度为 T 时序列对数比值的分布，并对 $\sigma = 0.01$ 与 $T \in \{100, 10^3, 10^4\}$ 计算 $\mathbb{P}[\rho > 10]$。据此论证在 $\varepsilon = 0.2$ 处做序列级截断是否合理。

2.2（**三种 KL 估计量。**）设 π_θ 与 π_{ref} 为参数 $p = 0.6$、$q = 0.5$ 的伯努利分布。计算精确的 $\mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$，以及在从 π_θ 采样时 k_1, k_2, k_3 的期望与方差。验证 $\mathbb{E}[k_2] \neq \mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$ 并量化其偏差。然后对 $p = 0.99$、$q = 0.5$ 重复上述计算，并说明随着散度增大，三者方差的排序如何变化。

2.3（**折扣就是一种长度惩罚。**）在终止奖励为 R、折扣为 γ 的词元级 MDP 中，证明长度为 T 的轨迹其回报为 $\gamma^{T-1}R$。对 $\gamma = 0.99$，计算长度为 200 的正确答案与长度为 2000 的正确答案之间隐含的相对偏好。设计一个显式的长度惩罚 $\lambda(T)$ 以复现同样的偏好，并解释为何显式形式更可取。

2.4（**陈旧度下的 ESS。**）某流水线用一个落后训练器 s 个优化器步的策略来生成采样轨迹。将每个词元的对数比值标准差建模为 $\sigma = \alpha s$，其中 α 很小。在对数正态近似下推导 $\mathbb{E}[\text{ESS}]/n$ 关于 s、T 与 α 的表达式，并对 $T = 4000$ 求出使 ESS 降到 0.3 以下的陈旧度。（认真做完这个计算，就能明白为什么异步 RL 系统要把陈旧度限制在一到两步之内；参见节 21.6。）

2.5（**最优基线。**）推导单提示策略梯度的方差最小化基线 $b^\star$，证明它是 $\mathbb{E}[\|\nabla \log \pi_\theta\|^2 r] / \mathbb{E}[\|\nabla \log \pi_\theta\|^2]$ 而非 $\mathbb{E}[r]$。用一句话解释为什么从业者仍然使用后者。

2.6（**集中化之下的 pass@k。**）把策略在各题上的成功概率建模为 Bernoulli(μ)，其中 μ 在题目间服从 Beta(a, b)。证明 pass@$k = 1 - \mathbb{E}_\mu[(1-\mu)^k]$ 并给出其闭式表达。现在假设 RL 实施映射 $\mu \mapsto \mu' = \mu^{1/\mathcal{T}} / (\mu^{1/\mathcal{T}} + (1-\mu)^{1/\mathcal{T}})$，其中 $\mathcal{T} < 1$（一个锐化操作）。用数值方法找出使 pass@1 上升而 pass@256 下降的 $a, b, \mathcal{T}$ 取值。

2.7 ⋆（**k_3 的梯度是 KL 的梯度吗?**）量 $\mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$ 既通过被积函数依赖 θ，又通过采样分布依赖 θ。证明 $\nabla_\theta \mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta \cdot \log \frac{\pi_\theta}{\pi_{\text{ref}}}]$（即「得分函数」项；另一项为零）。然后判断：在典型框架中按通常方式实现 k_1 与 k_3 估计量

并对其求导——即把样本视为固定的——是否得到这个梯度；若不是，得到的又是什么。讨论这一差异在实践中是否重要。

2.8 † **（观察熵坍缩。）**取一个 1B 的指令微调模型和一组 500 道可程序化判分的算术题。实现最简单的组相对策略梯度（不做截断、不加 KL、使用组均值基线、$G = 8$）。训练 200 步，记录：平均奖励、策略熵、平均回复长度、pass@1、pass@8，以及方差为零的组所占比例。绘制这五张子图，并写一段话解释各条曲线变动的因果先后顺序。请保留这张图；它是第12章 的参照点。

第 3 章

监督微调

核心思想: SFT 是前向 KL 目标下的行为克隆，而这句话的两半都是承重的

监督微调（supervised fine-tuning, SFT）最大化被演示回复的似然，这使它成为覆盖质量的：它必须把概率分配给演示集中的一切内容，包括其中的错误，而且没有任何机制去偏好某条演示胜过另一条。它还只以真值前缀为条件，因此从不在模型犯错之后所到达的状态上训练。这两条性质几乎解释了 SFT 的全部优点（便宜、稳定、几千条样本就能装上格式与习惯），也几乎解释了它的全部弱点（无法排序、无法纠错恢复，并且会平滑地退化为死记硬背）。

监督微调是后训练中最不光鲜的阶段，却也是可避免的质量损失占比最大的阶段。它的目标函数只是一行每位读者早已熟知的数学；真正要紧的一切都在构造之中：哪些词元承载损失、损失如何归一化、样本如何打包、对话模板是什么、训练多少轮，以及模型先前所知的一切会发生什么。本章认真对待这些问题，随后展开两个把 SFT 变成通往强化学习之桥梁的想法：拒绝采样微调，以及长思维链蒸馏。

3.1 目标函数

设 $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ 为一组提示—回复演示数据。把每一对拼接成单条词元序列，并令 $m_t \in \{0, 1\}$ 为损失掩码，它恰好在我们希望施加监督的位置上取 1。监督微调的目标为

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

其中 Z 是一个归一化因子，将在节 3.3中详细讨论。在标准的 仅补全设定下，$m_t = 1$ 覆盖回复的每个词元，而在提示以及它前面的模板脚手架上 $m_t = 0$。

3.1.1 为什么要掩掉提示

掩掉提示的做法如此普遍，以至于其理由很少被明说，而这个理由其实没有实践本身那么绝对。原则性的理由是我们拟合的是一个条件模型：我们部署的是 $\pi_\theta(y\mid x)$，而提示上的边缘分布由用户决定，因此为它建模只是白白消耗容量。实用的理由是权重：在词元均值之下，一个提示 2000 词元、答案 50 词元的样本，其梯度贡献是短提示样本的 40 倍，于是样本的重要性变成了提示长度的副产物。第三个理由专门针对合成数据：若提示本身就是模型生成的（第4章），在它们上面训练会教会策略去生成提示，这在部署时表现为模型替用户写出他的下一个问题。

在数据量很小的情形下，这并不总是正确的选择：Shi et al. (2024b) 报告说，当微调集较小或回复较短时，把指令词元也纳入损失是有帮助的，其他情况下则有害，机制是对少量补全的过拟合起到了正则化作用。证据等级（c），但它提醒我们「掩掉提示」是一个偏差—方差取舍，而不是一条定律。

3.1.2 作为前向 KL 的 SFT

设 $p_\star(y\mid x)$ 为演示数据所来自的分布——即「专家」，无论它是人类标注者、更强的模型，还是被验证器筛选过的当前策略——并设 μ 为提示分布。

命题 3.1 (SFT 最小化一个前向 KL)**.** 当 $Z\equiv 1$ 且在回复上 $m_t\equiv 1$ 时，

$$\mathcal{L}_{\mathrm{SFT}}(\theta)=\mathop{\mathbb{E}}_{x\sim\mu}\big[\mathbb{D}_{\mathrm{KL}}(p_\star(\cdot\mid x)\,\|\,\pi_\theta(\cdot\mid x))\big]+\mathop{\mathbb{E}}_{x\sim\mu}\big[\mathbb{H}[p_\star(\cdot\mid x)]\big], \tag{3.2}$$

而由于第二项不依赖于 θ，最小化 $\mathcal{L}_{\mathrm{SFT}}$ 就恰好是最小化从演示分布到策略的期望前向 KL。

证明. $\mathbb{E}_{y\sim p_\star}[-\log\pi_\theta(y\mid x)]=\mathbb{E}_{y\sim p_\star}[\log\frac{p_\star(y\mid x)}{\pi_\theta(y\mid x)}]-\mathbb{E}_{y\sim p_\star}[\log p_\star(y\mid x)]$，其中两项分别是 KL 与熵。 □

定理 3.1的分量远超过它那两行证明，因为 节 2.4已经确立：前向 KL 是覆盖质量的，反向 KL 是 寻模的。SFT 的每一条结构性性质，都源自它处在这条分界线上覆盖质量的那一侧。

推导: 覆盖质量的三个后果

用 ε 表示演示中表现出某种不良行为的比例：错误答案、谄媚式的含糊其辞、本不该出现的拒答。

(*i*) *SFT* 无法给坏数据降权。当 π_θ 对 $p_\star$ 会产生的东西赋予近乎零的概率时，前向 KL 会发散，因此一个被完美优化的策略会以约 ε 的比率复现那些不良行为。(3.1)中没有任何系数能说「这些演示价值更低」，所以数据过滤是唯一的抓手——这也是为什么第4章是一整章而不是一段话。
(*ii*) *SFT* 无法在多个模式之间做选择。如果演示中包含两种互不相容的好风格——比如简洁与铺陈——前向 KL 的最小化方式是同时覆盖两者，采样出来便是一个混合；而反向 KL 类方法（Chapters 8 and 11）则会做集中。这正是 节 1.2中那个观察的形式化内容：后训练是混合的选择，而 SFT 并不做选择。
(*iii*) 每个提示只有一条演示时，目标就是一个点质量。经验分布 $p_\star(\cdot|x)$ 是在唯一观测到的回复上的狄拉克分布，因此 (3.1)的精确最小值点会确定性地复现它。SFT 抵御死记硬背的唯一防线，就是我们提早停止，以及函数逼近具有泛化能力。

历史注记: 从任务混合到演示收集再到蒸馏

SFT 的谱系有三个阶段，区别在于监督信号从何而来。任务混合式指令微调 (Wei et al., 2022a; Chung et al., 2024) 把已有的 NLP 数据集改写成自然语言指令，并展示了向留出任务簇的零样本迁移；监督信号之所以便宜，是因为它是被重新利用的。演示收集 (Ouyang et al., 2022) 花钱雇承包商撰写回复，直到 LIMA (Zhou et al., 2023a) 主张 1,000 条精选样本就已足够，从而把该领域的重心从生成指令数据转向筛选指令数据。蒸馏 (Wang et al., 2023b; Xu et al., 2024a, 2025) 用更强的模型取代了人类，到 2025 年，开源 SFT 数据配比已以模型生成为主。每个阶段都有各自的失效模式：被重新利用的数据在对话格式上欠拟合，人类数据规模小且前后不一致，而蒸馏数据会继承教师的盲区及其基准污染。

3.2 作为模仿学习的 SFT 及其局限

式(3.1)就是行为克隆：从专家的状态–动作对中以监督方式学习一个策略，其中状态是前缀、动作是词元。远在语言模型出现之前，模仿学习文献就已知道这一方法的失效模式，而相关分析在有一条重要保留的前提下可以迁移过来。

3.2.1 误差累积

经典结果是一个归约界。假设学习者在专家状态分布下的期望单步误差至多为 ϵ，单步代价有界于 $[0,1]$，时域为 T。那么行为克隆满足

$$J(\hat{\pi}) \;\le\; J(\pi^\star) + O(\epsilon\, T^2), \tag{3.3}$$

而在学习者自身状态分布上训练的算法可达到 $O(\epsilon T)$ (Ross and Bagnell, 2010; Ross et al., 2011)。这个二次因子的机制很简单：每一步学习者以约 ϵ 的概率犯错；每次犯错都会把它带离专家分布，在那里它没有任何保证，并可能在余下的 $O(T)$ 步中不断累积代价；再对首次偏离的 $O(T)$ 个时机求和，就得到 T^2。DAgger (Ross et al., 2011) 通过在学习者实际访问的状态上收集专家标注来弥合这一差距，但这要求专家能在任意状态上被查询——对驾驶演示者来说可以做到，而对「已经写下三段有缺陷的推导之后，下一个词元该是什么」则做不到。

核心思想：渐近界是空洞的，机制却不是

对一个 10^4 词元的回复，(3.3)没有提供任何信息：即便 $\epsilon = 10^{-4}$ 也给出 $\epsilon T^2 = 1$，正是该界的平凡上限。它是最坏情况的，并且假定每次偏离都不可挽回；而语言其实冗余到足以让大多数词元级「错误」落在仍然存在良好续写的状态上。真正要紧的是承诺点的数目——即轨迹的其余部分取决于该处选择的那些位置，例如某个中间算术结果、某种证明策略，或某个工具调用参数。这个计数而非词元数才是有效时域，而可操作的表述是：SFT 的质量随它超线性地退化——对两段话的回答足够，对三十步的智能体轨迹则明显不够。

3.2.2 SFT 无法教会纠错恢复

这有两个彼此独立的原因，把它们混为一谈会让人以为问题可以靠更好的数据解决。第一个是分布层面的，即节 1.2中的曝光偏差：模型在专家前缀上训练，却在自己的前缀上被评估。计划采样 (Bengio et al., 2015) 把模型生成的词元混入条件上下文，但在这里帮不上忙，因为模型自己的轨迹没有标签。

第二个原因是演示中根本不包含纠错事件。谨慎的写作者很少写出「我三行之前弄错了，让我重做这一步」，而刻意去收集这类数据，教会的将是在演示者的错误状态上从 演示者的错误中恢复，而不是学生真正会到达的那些状态。强化学习则免费获得了这一点：它从正在被训练的策略中采样，因此每个训练状态都是该策略会访问的状态，而能够纠错的轨迹相对于不能纠错的轨迹会获得更高的奖励。在结果奖励式 RL 下

自发涌现的回溯行为 (DeepSeek-AI, 2025)，是这属于种类差异而非程度差异的最有力证据。

3.2.3 正确的轨迹仍可能教出草率的推理

有一种更微妙的失效，即便做了完美的结果过滤也依然存在。即使每条演示都以经过验证的正确答案收尾，其中间推理也可能并不能很好地刻画该答案是如何得到的，因为思维链常常对真正驱动预测的计算并不忠实 (Turpin et al., 2023; Lanham et al., 2023)。克隆这类轨迹装上的是审慎思考的表面形式，而答案却来自教师的捷径，于是留下彼此松散耦合的轨迹生成器与答案生成器——在捷径有效时一切正常，一旦失效便是灾难，因为轨迹不会给出任何预警。节 3.9给出相应的诊断方法。

3.2.4 「SFT 记忆，RL 泛化」

Chu et al. (2025a) 是这句口号背后的研究，它值得被准确转述。在 GeneralPoints（一种算术纸牌游戏）与 V-IRL（导航）上，在纯语言与视觉—语言两种变体中，作者分别扩展 SFT 与 RL 的计算量，训练时用一套规则、测试时用未见过的规则。基于结果奖励的 RL 在全部四种任务设定下都提升了分布外性能，而 SFT 在全部四种设定下都使其退化；此外 RL 还提升了 SFT 完全未触及的底层视觉识别能力。

有三点限定条件是必要的。同一篇论文也报告 SFT 是必需的：没有它，视觉—语言基座模型产出的是「冗长、跑题且缺乏结构」的输出，从中提取不出任何奖励。论文自己的分析把损害归因于过量的 SFT，它固化了随后的 RL 无法克服的记忆化模式——这是一个「金发姑娘区间」，而不是方法之间的优劣排序。而这句口号最容易掩盖的一点是：该比较固定的是训练计算量而非数据多样性——SFT 只见到一套规则，RL 却在探索；考虑到已有证据表明大规模多样化配比泛化良好 (Lambert et al., 2025; Guha et al., 2025)，一个跨规则的实验结果可能会不同。可辩护的结论比标题窄得多：在固定且狭窄的数据上，模仿会记忆，而同策略优化不会。证据等级（b），争议在于诠释而非测量。

3.3 真正重要的损失构造细节

下面是一份实现决策清单，其中每一项都曾把某人的基准分数挪动得比任何算法改进都多。

3.3.1 词元级平均与序列级平均

给定一个包含 B 条样本的批次，其掩码为 $m_{i,t}$、逐词元损失为 $\ell_{i,t} = -\log \pi_\theta(y_{i,t} \mid \cdot)$，常用的归一化方式有三种：

$$\mathcal{L}_{\text{tok}} = \frac{\sum_i \sum_t m_{i,t}\, \ell_{i,t}}{\sum_i \sum_t m_{i,t}}, \qquad \text{每个受监督词元权重相同,} \tag{3.4}$$

$$\mathcal{L}_{\text{seq}} = \frac{1}{B} \sum_i \frac{\sum_t m_{i,t}\, \ell_{i,t}}{\sum_t m_{i,t}}, \qquad \text{每条样本权重相同,} \tag{3.5}$$

$$\mathcal{L}_{\text{const}} = \frac{1}{B\, L_{\max}} \sum_i \sum_t m_{i,t}\, \ell_{i,t}, \qquad \text{固定归一化因子，与批次构成无关。} \tag{3.6}$$

它们是真正不同的目标函数。在一个由 30 词元的拒答与 8000 词元的推理轨迹按相同样本数混合而成的数据集中，拒答在(3.4)下只得到 $30/8030 \approx 0.4\%$ 的梯度，而在(3.5)下得到 50%。抽象地看两者都不算对：词元级平均正确地估计了逐词元交叉熵，是长文流畅性得以成立的原因；而序列级平均编码的信念是每条演示都是关于行为的一个证据单位。词元级平均再加上显式的逐领域采样权重更可控，因为这样一来补偿就写在配置文件里，而不是藏在长度分布里。这一分岔在 RL 中会以词元均值与序列均值 GRPO 目标之争的形式重现（节 12.2），在那里出于同样的理由仍有争议；该领域两次抵达同一路口，却没有把先前的讨论带过去。

把掩码视为设计变量，其逻辑终点是在回复内部做选择 (Lin et al., 2024b)。此时有一个实现细节决定了你实际优化的是哪个目标。在跨 K 个微批次做梯度累积时，(3.4)必须除以该优化器步的全局受监督词元数；对每个微批次各取词元均值再平均，会悄悄地给受监督词元少的微批次加权。请对计数做 all-reduce、只除一次，并检查固定批次上的损失是否与 K 无关。

3.3.2 打包与交叉污染

把一个批次填充到其中最长的样本长度，浪费的计算量与长度分布的偏斜程度成正比，而 SFT 数据配比的偏斜相当严重。序列打包把样本首尾相接地拼进定长缓冲区，使填充浪费降到近乎为零，是每个正经训练栈中的标准做法。它引入两条容易搞错的正确性要求 (Krell et al., 2022)。注意力必须是块对角的：样本 j 的第 t 个词元不得注意到样本 $j-1$ 的任何词元，无论它们在缓冲区中是否相邻。各实现通过接受累积序列偏移量的变长注意力核来提供这一能力，因为在 $L = 32{,}768$ 时一个稠密的 $L \times L$ 掩码本身就是个显存问题。此外位置索引必须在每个边界处重置；否则靠后的样本会在它们推

理时永远不会遇到的位置上被训练，而旋转位置编码则会获得一种对缓冲区内偏移量的虚假依赖。

其中任一处出错，都会产生一条看起来正常的损失曲线，以及只表现为长上下文行为略差的性能退化。廉价的回归测试是：把同一条样本的两份副本按不同顺序打包，检查逐词元损失是否一致。

3.3.3 标签平滑、噪声与其他借来的正则化手段

标签平滑把独热目标替换为 $(1-\varepsilon)\delta_{v^\star}+\varepsilon/|\mathcal{V}|$。它在图像分类中是标准做法，在这里通常却是个错误：它给 10^5 路词表中的每个词元都设了概率下限，包括在上下文中根本不可能出现的词元；它破坏了非目标 logits 之间的相对相似性结构，而这恰恰是后续蒸馏阶段所需要的 (Müller et al., 2019)；而且它带来的正则化效果，用「训练一到两轮就停」这一更便宜的办法同样可以获得。SFT 之后的过度自信，通常诊断的是在太少的数据上训练了太多轮。

NEFTune(Jain et al., 2024) 向输入嵌入加入按 $\alpha/\sqrt{Ld}$ 缩放的均匀噪声，其论证颇为合理：这可以防止模型死抠精确的表面形式。它报告的增益很大，但集中在由大模型评判的基准上，并且伴随着更长的回复，而那些评判者本来就偏好长回复；证据等级（c）。

3.3.4 特殊词元，以及停不下来的模型

序列结束词元是掩码中后果最重的单个位置。如果 `<eos>` 被掩掉——输入/标签移位处的差一错误就会导致这一点，这也是手写 SFT 循环中最常见的 bug——模型就永远学不会终止：它先给出正确回答，然后继续写出第二个没人要的答案，直到撞上解码上限。

通往同一失效的一条不那么明显的路径是截断。每一条在 $L_{\max}$ 处被截断却仍被保留的超长样本，都在演示「长度为 $L_{\max}$ 的序列不会结束」，于是模型学到一种依赖长度的拒绝停止行为，而且恰好集中在危害最大的地方。请丢弃超长样本或调高 $L_{\max}$；如果非截断不可，就把该样本整条排除出损失。

角色标记则属于策略决定而非 bug。起始标记由服务栈提供，因此监督它什么也教不会；结束标记必须由模型自己发出，因此监督它至关重要。于是代码 3.1中的默认做法是：从助手轮次的第一个内容词元开始，一直监督到它的终止符（含）为止。

代码 3.1: 带打包的仅补全损失掩码。掩码在每条样本上构建一次并一路传递到 collator；注意显式的移位、对轮次终止符的纳入，以及按全局词元数所做的归一化。

```python
def build_example(tokenizer, messages, max_len):
    """Return (input_ids, loss_mask) for a multi-turn conversation."""
    ids, mask = [], []
    for msg in messages:
        header = tokenizer.encode(f"<|im_start|>{msg['role']}\n", add_special_tokens=False)
        body   = tokenizer.encode(msg["content"] + "<|im_end|>\n", add_special_tokens=False)
        ids  += header + body
        # 只监督助手的内容及其终止符，其余一概不监督
        supervise = 1 if msg["role"] == "assistant" else 0
        mask += [0] * len(header) + [supervise] * len(body)
    if len(ids) > max_len:
        return None                      # 丢弃，绝不截断：见关于 EOS 的讨论
    return ids, mask

def sft_loss(logits, input_ids, loss_mask, n_tokens_global):
    """Token-mean cross entropy, normalized by the GLOBAL supervised token count."""
    logits = logits[:, :-1].float()      # float32 的 log-softmax 不是可选项
    labels = input_ids[:, 1:]
    mask   = loss_mask[:, 1:].to(logits.dtype)
    nll = torch.nn.functional.cross_entropy(
        logits.transpose(1, 2), labels, reduction="none")
    # n_tokens_global = all_reduce(整个优化器步内 mask 之和)
    return (nll * mask).sum() / n_tokens_global
```

3.3.5 多轮掩码

对一段 K 轮的对话，可以监督每一个助手轮次，也可以只监督最后一个。监督每一轮大约能得到 K 倍的受监督词元，是近乎通用的默认做法，但它有一个值得说明的前提：每一轮都是以真值历史为条件的，其中包括模型并未产生的那些助手轮次。这就是轮次粒度上的教师强制，也是多轮 SFT 训练出的模型在第一轮上比在第五轮上更好的原因。在智能体式设定中还有两条不可让步的规则（第17章）：工具观测以及其他由环境返回的文本必须被掩掉，因为它们不是策略动作，监督它们会教会模型幻觉出工具结果；而被过滤器标记为低质量的某一轮，应当被单独掩掉，而不是让整段对话陪葬。表 3.1汇总了这些决策及其失效模式。

表 3.1: SFT 中的损失构造选择、每一项所控制的性质，以及选错时产生的失效模式。这些都不会在训练损失曲线上显现，所以它们通常是在生产环境中被发现的。

选择	控制什么	选错时的失效模式
掩掉提示	容量分配、样本权重	模型替用户写出下一轮；长提示样本占据主导
归一化因子(3.4)–(3.6)	短样本与长样本的相对权重	拒答与短答案只拿到 $< 1\%$ 的梯度，或者反过来长轨迹如此
全局词元计数	对累积步数的不变性	K 或并行规模改变时有效配比权重发生偏移
块对角注意力	打包样本之间的独立性	跨文档泄漏；长上下文行为退化
位置索引重置	位置泛化	对打包缓冲区内偏移量的虚假依赖
监督 `<eos>`	终止	模型永不停止；一直跑到解码上限
超长样本处理	依赖长度的终止行为	模型学到长序列不会结束
掩掉工具输出	动作与观测的分离	模型幻觉出工具结果而不去调用工具

3.4 对话模板与训练/部署契约

对话模板是一个确定性函数 C，它把结构化的对话——带角色标签的消息，可能还带有工具定义——映射为一串扁平的词元序列，并附带关于生成从何处开始的约定。由于它通常以 Jinja 字符串的形式存在于分词器配置中，而不是以代码的形式存在于训练仓库里，它是系统中最缺乏测试保护的部分。这份契约有三个条款。

训练与服务必须应用完全相同的函数。 策略是 $\pi_\theta(y \mid C(\text{conversation}))$，所以 C 是策略定义的一部分。如果训练时把助手头部渲染为 `<|im_start|>assistant\n`，而服务时渲染为 `<|im_start|>assistant` 后跟一个空格，那么推理就是在模型从未见过的词元序列上做条件。它仍会产出流畅的文本，但那来自前缀空间中未被训练的区域，而退化是弥散的：指令遵循变差、含糊其辞增多、偶尔角色混淆、时而无法发出停止词元。在成对评判的评测上，这可能损失好几个百分点，而训练侧没有任何指标发生变化。

模板要与检查点一同版本化。 模板是会演进的：新增一种工具调用模式、重命名一个分隔符、改动系统消息的默认值。只要模板与权重分开发布——例如分词器配置取自某个仓库的主分支，而权重却锁定在某个修订版——一次模板更新就会悄无声息地重新

定义每一个已部署的检查点。这就是模板漂移，其特征是出现质量回退，而模型制品上的差异却是空的。请把模板哈希与权重哈希并列固定下来，并在加载时断言校验。

推理分隔符同样是契约的一部分。 一个推理模型会发出 `<think>` c `</think>` a，其中有三项决定必须在 SFT、RL 与服务之间保持一致：生成提示是否预先填入 `<think>`（如果训练监督了它而服务又预填了它，模型就会再发一个）；较早轮次的推理块是否会保留到后续轮次的上下文中（多数部署会剥除它们，因此训练也必须如此）；以及多轮数据中推理块是否承载损失。第二项是常见且代价高昂的错误。代码 3.2展示了一个模板、它渲染出的词元流，以及该词元流所隐含的掩码。

代码 3.2: 一个最小的对话模板，以及它产生的确切词元流与损失掩码。标记为 1 的位置承载损失。注意助手头部不受监督，推理块与答案受监督，而轮次终止符也受监督，以便模型学会停止。

```
--- 模板（Jinja） ------------------------------------------------------------
{%- for m in messages %}
{{ '<|im_start|>' + m.role + '\n' + m.content + '<|im_end|>\n' }}
{%- endfor %}
{%- if add_generation_prompt %}{{ '<|im_start|>assistant\n' }}{%- endif %}

--- 渲染后的训练样本 ----------------------------------------------------------
<|im_start|>system\nYou are terse.<|im_end|>\n        mask: 0 0 0 0 0 0 0 0 0
<|im_start|>user\nWhat is 17*23?<|im_end|>\n          mask: 0 0 0 0 0 0 0 0 0
<|im_start|>assistant\n                               mask: 0 0 0
<think>17*23 = 17*20 + 17*3 = 340+51</think>          mask: 1 1 1 ... 1
391<|im_end|>\n                                       mask: 1 1 1

--- 渲染后的推理前缀（add_generation_prompt=True） --------------------------
<|im_start|>system\nYou are terse.<|im_end|>\n
<|im_start|>user\nWhat is 17*23?<|im_end|>\n
<|im_start|>assistant\n                               <- 生成从这里开始
```

常见陷阱: 无声的模板不匹配

三项检查基本能拦住所有这类问题，而且只需一小时就能一次性实现。（1）对同一段对话分别渲染出训练样本与服务前缀，并断言服务前缀在字节层面正是训练渲染结果的前缀。（2）在每次运行的第 0 步记录一个训练批次前 64 个词元的解码结果并亲自读一遍；这能立刻发现双 BOS、系统消息丢失以及被破坏的工具模式。（3）在宣布结论之前，用真实的服务栈来做评测。在 RL 阶段还要加上第四

项：断言训练器与推理引擎对同一条采样提示的渲染逐词元一致，因为不匹配会以看起来像数值问题的方式，破坏 第22章中的对数概率比值。

3.5 超参数与调度，附带理由

SFT 的超参数在不同模型家族之间足够稳定，因此列一张表是有用的，前提是把理由一并给出。

学习率。 峰值学习率随模型宽度下降，大致按 $\eta \propto 1/d_{\text{model}}$，这与最大更新参数化的论证一致，尽管多数公开配方是调出来的而非推导出来的。对同一个模型，微调学习率比预训练学习率低一到两个数量级，因为模型已经起步于一个好解附近，目标只是移动一小段距离。

调度与预热。 预热让 Adam 的二阶矩估计在迈出大步之前先稳定下来；总步数的 3–5%、并以约 20 步为下限，就足够了。余弦衰减到峰值的 10% 是默认做法，而在这样的时域上线性衰减到零的表现几乎相同。预热—恒定—衰减 (Hu et al., 2024b) 把衰减阶段与一个事先承诺的步数解耦——可以在恒定学习率下继续训练、随时快照、从任意点开始衰减——这在数据配比还在改动时很有价值。而对一个固定且已知的预算，它相对余弦调度并无优势。

训练轮数。 一到三轮。再多就会过拟合，原因还是 定理 3.1：每个提示只有一条回复时，经验目标是一个点质量，因此额外的轮次是靠把策略锐化到确切的表面形式来降低训练损失的，而留出集质量会先升后降，通常在第 2 到第 4 轮之间见顶。更要紧的是，策略熵会单调下降，而低熵的检查点是糟糕的 RL 初始点，因为组相对方法需要采样多样性才能产生任何梯度（节 2.5）。过度训练的 SFT 用随后 RL 阶段的代价，换来了略好一点的 SFT 评测分数；要发现这一点，请报告 $k \geq 8$ 的 pass@k，而不只是 pass@1。

批大小、权重衰减与梯度裁剪。 批大小要以受监督词元数而非样本数来表述，因为在长度分布不同的数据配比之间，样本数是没有意义的；对 7B–70B 的模型，每个优化器步 0.5M 到 4M 是典型值，更大的批次主要换来的是吞吐。把全局梯度范数裁剪在 1.0，它偶尔会起关键作用，因为一条畸形样本就可能产生极大的梯度。权重衰减在几千步

表 3.2: 按模型规模给出的 SFT 超参数起始取值，适用于以 AdamW（$\beta = (0.9, 0.95)$，$\epsilon = 10^{-8}$）在 bfloat16 下做全量微调。这些是截至 2026 年中公开配方所见范围的中心值；请把它们当作有待在两倍范围内调整的初始猜测，而不是测量结果。

规模	峰值学习率	预热	轮数	每步词元数	备注
1–3B	2×10^{-5}	3%	2–3	0.25–1M	可承受更高学习率；过拟合最快
7–14B	1×10^{-5}	3%	2	0.5–2M	研究最充分的区间
30–70B	5×10^{-6}	3%	1–2	1–4M	激活显存占主导；应积极打包
$>$ 100B 稠密	2×10^{-6}	5%	1–2	2–8M	学习率敏感度上升；先在 7B 代理模型上验证
MoE（任意规模）	稠密模型的 0.5×	5%	1–2	同稠密模型	注意专家负载均衡；不要跨家族沿用 `aux_loss` 设置

的尺度上几乎不起作用，许多配方直接把它设为零；而一旦设得高到能起作用，它拉向的是原点，而不是基座模型。表 3.2按规模汇总了起始取值。

工程实践: 一份具体的 SFT 配方，以及为什么 SFT 学习率是 RL 学习率的 10–100 倍

一个 7B 模型，40 万条样本、平均 1,500 个受监督词元，打包到 16k：峰值学习率 1×10^{-5}，余弦衰减到 1×10^{-6}，30 步预热，2 轮，每步 1M 受监督词元（约 1200 步），梯度裁剪 1.0，权重衰减 0，bfloat16 配 float32 的 log-softmax，超长样本丢弃而非截断。记录每步的受监督词元数、逐领域损失、策略熵，以及在一个小型可验证留出集上的 pass@8。

这里的学习率是节 2.9对同一模型所建议的 RL 学习率的 10–100 倍，原因有三。信号密度：SFT 在每个词元上都提供一个完整的类别目标，梯度条件良好；策略梯度每个回合只提供一个标量优势，还要广播到成千上万个词元上，因此逐词元的梯度由噪声主导，而大步长会把这些噪声积分成一次随机游走。目标有效性：RL 的替代目标只在采样策略附近有效——重要性比值会退化，优势是在 $\pi_{\theta_{\text{old}}}$ 下算出来的——所以学习率还兼任信赖域；而 SFT 的目标并不会因为你朝它迈了一步就失效。需要移动的距离：SFT 必须弥合格式上的差距，这是行为上的巨大改变；而 RL 是在很小的 KL 预算内重塑一个已经差不多正确的策略。

3.6 灾难性遗忘与数据配比设计

(3.1)中没有任何东西在保卫 SFT 数据配比未曾练到的能力。梯度完全来自演示词元，因此演示分布之外的行为是作为副作用移动的，且方向不受约束。

3.6.1 经验规律的形状

在数据配比未覆盖的留出任务上，性能大致随 SFT 词元数单调退化，其中有三条规律在多项研究中都成立 (Kirkpatrick et al., 2017; Luo et al., 2025d; Biderman et al., 2024a)。退化对离微调分布最远的能力最严重：在英文对话上微调，损失的多语言与代码能力比损失的英文问答能力更多。在相同词元预算下，更大的模型遗忘更少，这与节 3.7中的内在维度图景一致。而且损害是前置的——大部分发生在最初几百步之内——所以提前停止只是一种微弱的补救。

整条流水线上的总体效应就是对齐税 (Askell et al., 2021; Ouyang et al., 2022)：对齐后的模型在能力基准上的得分低于它们所源自的基座模型。Ouyang et al. (2022) 通过把预训练梯度混入 RLHF 更新中收回了其中大部分——这是关于「补救之道在于复习而非更小的学习率」的第一份、也仍是最直接的证据。

3.6.2 复习与配比权重

回放把一小部分预训练风格的数据混入 SFT 配比，并在所有词元上计算损失，因为这类样本没有提示/回复结构。词元占比通常在百分之一到百分之十之间，大部分收益在低端就已取得；此时梯度中就包含了一项在保卫预训练分布。

跨技能领域的配比权重所具备的杠杆不亚于学习率，被调整的次数却远少于学习率。记录最完整的公开流程是 Tülu 3 的做法 (Lambert et al., 2025)：先为每项技能各训练一个模型，以确定该技能在孤立情况下可达的上限，再合成一个预览配比，然后迭代地增删数据集，去修复那些落后于其单独上限的技能，并全程做去污染。让这套流程真正有效而不只是过拟合的关键，是把开发用评测套件与一套未曾见过的评测套件分开。

3.6.3 作为廉价缓解手段的权重平均

由于一次短暂的 SFT 训练与其初始化处在同一个损失盆地中，线性插值

$$\theta_\alpha = (1-\alpha)\,\theta_{\text{base}} + \alpha\,\theta_{\text{sft}}, \qquad \alpha \in [0,1], \tag{3.7}$$

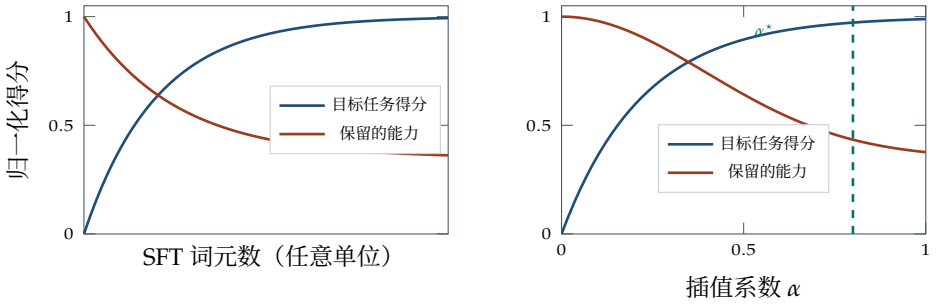


图 3.1: 示意图，非实测。左：随着 SFT 进行，目标任务得分趋于饱和，而数据配比未曾练到的能力则衰减，且大部分衰减发生在早期。右：如(3.7) 那样在基座权重与微调权重之间插值，勾勒出的是另一条权衡曲线，因为任务得分随 α 饱和的速度快于保留能力衰减的速度；有用的推论是，某个中间的 $\alpha^\star$（虚线）往往能保住几乎全部任务收益，同时挽回失去的很大一部分。图中曲率仅为定性示意；$\alpha^\star$ 的位置必须为每次训练各做一次扫描来确定。

勾勒出一条路径，沿着它任务性能与保留能力平滑地此消彼长，而且通常是有利的：只需付出任务收益的一小部分，就能挽回大量遗忘。这就是 WiSE-FT (Wortsman et al., 2022b)，最初是为零样本视觉模型的鲁棒微调而提出的；同样的观察也支撑着模型汤 (Wortsman et al., 2022a) 与任务算术 (Ilharco et al., 2023)，在后者中 $\theta_{\text{sft}} - \theta_{\text{base}}$ 是一个可组合的「任务向量」。

代价是一次插值再加上大约五个 α 取值的扫描（图 3.1），这使它成为本章中最廉价的干预手段。有两点保留。它依赖线性模式连通性，而这只对从同一初始化出发的微调成立，对独立训练的模型则不成立。另外，朝基座模型插值也就是朝基座模型的安全行为插值，因此扫描必须包含安全评测，而不只是能力评测。

3.7 参数高效微调

全量微调所需的优化器状态与参数量成正比——在混合精度下使用 AdamW 大约是每参数 12–16 字节，对 70B 模型来说，还没算激活就已超出单节点的显存。参数高效方法转而只训练少量新参数。早期的变体插入瓶颈适配器 (Houlsby et al., 2019) 或优化连续前缀 (Li and Liang, 2021)；两者都会增加推理延迟，而低秩适配不会，因此后者已经取代了它们。

3.7.1 LoRA

对冻结的权重矩阵 $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$，LoRA (Hu et al., 2022) 把更新参数化为一个低秩乘积：

$$W = W_0 + \Delta W, \qquad \Delta W = \frac{\alpha}{r} BA, \qquad B \in \mathbb{R}^{d_{\text{out}} \times r}, \ A \in \mathbb{R}^{r \times d_{\text{in}}}, \tag{3.8}$$

其中 $r \ll \min(d_{\text{in}}, d_{\text{out}})$。初始化取 $A \sim \mathcal{N}(0, \sigma^2)$ 与 $B = 0$，于是第 0 步时 $\Delta W = 0$，训练恰好从基座模型开始。可训练参数量是 $r(d_{\text{in}} + d_{\text{out}})$ 而非 $d_{\text{in}} d_{\text{out}}$，优化器状态按同样比例缩小，而在推理时 ΔW 可以折叠进 W_0，不带来任何延迟开销。

α/r 的缩放使 ΔW 的幅值近似与 r 无关；没有它，秩翻倍就会让有效步长翻倍，学习率在每个秩上都要重新调。有了它，最优学习率大致与秩无关，这就把秩变成了一个容量旋钮，而不是一个优化旋钮。

为什么秩为 8 的更新就够用？经验性的答案早于 LoRA：微调目标可以在一个随机的低维子空间内被优化，而所需的维度——即内在维度——很小，并且随着预训练模型变得更大更好而减小 (Aghajanyan et al., 2021)。把一个通用模型变成特定任务模型的那个对象本身很小，而更好的预训练会让它更小，这也顺带解释了为什么更大的模型遗忘更少。

3.7.2 变体

QLoRA (Dettmers et al., 2023) 把冻结的基座量化为 4 比特 NormalFloat，适配器保持 bfloat16，并加入双重量化与分页优化器状态以熬过显存尖峰；它使 65B 模型的单卡微调成为可能。*DoRA* (Liu et al., 2024c) 把 W_0 分解为幅值与方向并只更新方向，其依据是全量微调与 LoRA 在改变这两者的方式上存在系统性差异；它在小秩下弥合了部分差距。表 3.3汇总了如何选择。

3.7.3 2025—2026 年的证据究竟说明了什么

有两项细致的研究看起来彼此矛盾，而调和它们是本节能做的最有价值的事。Biderman et al. (2024a) 发现，在代码与数学领域的继续预训练与指令微调上，LoRA 在目标域指标上不及全量微调，但对源域的遗忘要少得多，并将其刻画为一种比权重衰减或 dropout 更强的正则化手段。Schulman and Thinking Machines Lab (2025) 则发现 LoRA 与全量微调 相当，包括在策略梯度 RL 上，那里它甚至在秩为 1 时也能持平。

起调和作用的变量，是适配器容量相对于训练信号信息量的大小。Schulman and Thinking Machines Lab (2025) 为监督学习提炼出的规则是可数的：只要可训练适配

表 3.3: 参数高效微调的各个选项及其各自适用的区间。「可训练参数」一列是对 7B 模型在所有线性层上使用秩 32 时的近似值。最后一列是选择之前应当自问的问题。

方法	可训练参数	基座精度	推理开销	何时选它
全量微调	7B	bf16	无	数据集相对适配器容量而言很大，或者你要改变模型的知识
LoRA	~80M	bf16	无（可合并）	指令微调、RL、多个任务变体共用一个基座
QLoRA	~80M	4-bit NF4	无（可合并）	显存是首要约束；可接受更慢的单步
DoRA	~85M	bf16	无（可合并）	被迫使用极低秩且 LoRA 的差距可测
适配器 / 前缀	~20–80M	bf16	新增层或前缀	遗留方案；除非需要不合并就按请求切换，否则优先 LoRA

器参数量超过所训练的受监督词元数，LoRA 的表现就与全量微调一致。越过这一点之后并不存在硬性的损失下限，只是训练效率按数据集超出容量的程度成比例地下降。在数十亿代码词元上做继续预训练远远越过了这条边界；而在 10^7–10^8 个受监督词元上、以秩 32 作用于所有矩阵做指令微调则没有。

对 RL 来说，这一论证更强也更干净。监督学习每个回合提供 $O(\text{词元数})$ 比特，而策略梯度由一个携带 $O(1)$ 比特的优势驱动，因此 RL 每个词元吸收的信息大约少三个数量级。在Schulman and Thinking Machines Lab (2025) 的 MATH 实验中，10^4 道题 ×32 个样本、每个补全约一比特，合计约 3×10^5 比特，而 8B 模型上秩为 1 的适配器约有 3×10^6 个参数：容量远未成为约束。如果这一点成立，那么对 RL 而言 LoRA 就是默认选择而非妥协——而随之而来的权重同步流量的骤降，从数十 GB 降到数十 MB，其幅度之大足以改变 RL 基础设施的设计方式（第21章）。

同一工作还给出三条实践规则，每一条都对应着一种常见的 LoRA 误用。学习率约取全量微调的 10 倍；原封不动地照搬全量微调配置，是从业者断言 LoRA 表现不佳的最常见原因。把 *LoRA* 应用到所有权重矩阵上，包括 MLP 块，对混合专家模型还包括专家：提高秩并不能弥补只作用于注意力的做法，而且这一差距即便在秩 256 时依然存在。批大小要比全量微调更小，因为 LoRA 对极大的有效批大小容忍度较差。

证据等级：RL 方面的结果是（c）——一项细致的研究加一次独立复现，尚未在前沿规模上被复制——而「LoRA 起正则化作用、遗忘更少」则是（a），在若干条独立的证据线上都一致。

算法 3.1 拒绝采样微调（专家迭代）

输入： 初始策略 π_{sft}，提示集 $\mathcal{D}$，验证器 r_{ver}，组规模 G，阈值 τ，轮数 M，每提示保留上限 c

$\theta_0 \leftarrow \pi_{\text{sft}}$ 的参数
for $m = 0$ **to** $M - 1$ **do**
  生成
  for 每个提示 $x \in \mathcal{D}$ **do**
    在温度 $\mathcal{T} \geq 1$ 下采样 $\{y^{(i)}\}_{i=1}^{G} \sim \pi_{\theta_m}(\cdot \mid x)$ ▷ 多样性比质量更重要
  end for
  筛选
  $\mathcal{D}_m \leftarrow \{(x, y^{(i)}) : r_{\text{ver}}(x, y^{(i)}) \geq \tau\}$
  按推理路径签名对 $\mathcal{D}_m$ 去重；每个提示最多保留 c 个幸存样本 ▷ 防止简单提示占据主导
  重新加权，使 $\mathcal{D}_m$ 的难度分布与 $\mathcal{D}$ 相匹配
  拟合
  if 每轮从初始策略重启 **then**
    $\theta_{m+1} \leftarrow \arg\min_\theta \mathcal{L}_{\text{SFT}}(\theta; \mathcal{D}_0 \cup \cdots \cup \mathcal{D}_m)$，从 π_{sft} 出发
  else
    $\theta_{m+1} \leftarrow \arg\min_\theta \mathcal{L}_{\text{SFT}}(\theta; \mathcal{D}_m)$，从 θ_m 出发 ▷ 更便宜；漂移更快
  end if
end for
return π_{θ_M}

3.8 拒绝采样微调与专家迭代

到目前为止一切都假定演示来自外部。一旦策略可以被采样并被打分，就打开了一个新的选项：从当前策略生成、保留通过筛选的样本、在幸存者上做微调。这就是拒绝采样微调（RFT），它以若干名字出现——STaR (Zelikman et al., 2022)、RAFT (Dong et al., 2023a)、RFT (Yuan et al., 2023a)、ReST (Gulcehre et al., 2023)、ReST$^{\text{EM}}$ (Singh et al., 2024)——它们都是专家迭代 (Anthony et al., 2017) 的实例。仔细推导它是值得的，因为看清它丢弃了 RL 目标的哪些部件，正好为第6章把这些部件补回来提供了动机。

3.8.1 它是一个 EM 算法

固定一个提示，并令 $r(y) \in \{0,1\}$ 表示验证器是否接受。考虑最大化「样本被接受」的对数概率，

$$\ell(\theta) = \log \mathop{\mathbb{E}}_{y\sim\pi_\theta}[r(y)] = \log \sum_y \pi_\theta(y)\, r(y), \tag{3.9}$$

其中把回复视为隐变量。对回复上的任意分布 q，Jensen 不等式给出证据下界

$$\ell(\theta) \;\geq\; \mathop{\mathbb{E}}_{y\sim q}[\log \pi_\theta(y)] + \mathop{\mathbb{E}}_{y\sim q}[\log r(y)] + \mathbb{H}[q], \tag{3.10}$$

当 $q(y) \propto \pi_\theta(y) r(y)$ 时该下界是紧的。

推导: 拒绝采样是 E 步，SFT 是 M 步

E 步令 $q_m(y) \propto \pi_{\theta_m}(y)\, r(y)$，而从 q_m 中采样恰恰就是拒绝采样：从 π_{θ_m} 中抽样，当且仅当验证器接受时保留该样本。M 步在 q_m 固定的情况下最大化(3.10)的第一项，而由于另外两项不含 θ，这就是 $\arg\max_\theta \mathbb{E}_{y\sim q_m}[\log \pi_\theta(y)]$——正是在幸存样本上的 (3.1)。标准的 EM 单调性成立：在 E 步精确的情况下，$\ell(\theta_{m+1}) \geq \ell(\theta_m)$。每个提示只取 G 个样本时，E 步是蒙特卡洛式的，保证也只是近似的，这就是算法 3.1要对每个提示的幸存样本数设上限的原因：否则策略已经会做的提示会贡献 G 条近乎相同的幸存样本，经验的 q_m 就会被那些不携带信息的问题所主导。

3.8.2 它也是一种粗糙的策略梯度

对同样的奖励写出节 2.7中的策略梯度恒等式：

$$\nabla_\theta \mathop{\mathbb{E}}_{y\sim\pi_\theta}[r(y)] = \mathop{\mathbb{E}}_{y\sim\pi_\theta}[r(y)\, \nabla_\theta \log \pi_\theta(y)]. \tag{3.11}$$

而 RFT 在一批由 π_{θ_m} 生成的幸存样本上实际采取的梯度是

$$g_{\text{RFT}} = \mathop{\mathbb{E}}_{y\sim\pi_{\theta_m}}[\mathbb{1}(r(y) \geq \tau)\, \nabla_\theta \log \pi_\theta(y)]. \tag{3.12}$$

比较(3.11)与(3.12)，恰好可以识别出四处近似，每一处都点出了一个后续章节要补回来的部件：

1. **没有基线**。系数是 $\mathbb{1}(r \geq \tau) \in \{0,1\}$ 而不是优势 $r - b$：无论某个幸存样本是否本来就已近乎确定，它都被同等力度地推高，而这正是定理 2.2所要解决的方差问题。

2. **没有负梯度**。失败样本毫无贡献，因此 RFT 只能抬高概率、永远不能压低概率：它无法压制一个自信而错误的模式，只能靠竞争把它挤下去，而归一化会让这个过程很慢。这是它与策略梯度之间最大的实际差异，也是 RFT 会饱和的主要原因。
3. **没有重要性校正**。样本来自 π_{θ_m}，梯度却在 π_θ 处求取，而后者在内层 SFT 各轮中不断漂移。RFT 忽略了这个比值；PPO 式的截断以及 节 2.6中的截尾方案则会妥善处理它。
4. **硬阈值**。$\mathbb{1}(r \geq \tau)$ 丢弃了分级奖励中的全部信息，而在使用奖励模型而非验证器时，那才是信号的大部分；因此在软奖励下，采用排序而非阈值的 RAFT 式方法表现更好 (Dong et al., 2023a)。

工程实践：让 RFT 真正奏效

采样温度取 1.0 或更高：E 步要的是对 q_m 的覆盖，而一个本已自信的策略在低温下采出的样本贡献不了任何新东西。请按推理路径的归一化签名而非精确字符串来对幸存样本去重，因为近似重复才是主要的失效来源 (Yuan et al., 2023a)。给每个提示的幸存样本设上限，并让难度分布与提示集保持匹配，否则数据会向简单问题退化。每一轮都要从 当前策略重新生成：复用第 0 轮的样本会把整个流程变成离线自蒸馏，那只会锐化初始策略而不会改进它。此外还要在两种做法之间做出有意识的选择：每轮都从 π_{sft} 重启（ReST$^{\text{EM}}$ 式：更稳定，更耗算力），还是从 θ_m 继续（更便宜，漂移更快）；重启明显更不容易发生那种在三四轮之后终结大多数朴素专家迭代循环的熵坍缩。

在 RL 之前先跑一轮 RFT，不仅有教学价值，也有实用价值：它只需要一个监督训练循环，不需要第21章中的任何基础设施，而且在许多任务上，它能拿到完整 RLVR 最终所带来收益的大部分。当它停滞时——而它一定会停滞——停滞的样子会告诉你需要 RL 目标中的哪一个部件。

3.9 长思维链 SFT

2025—2026 年间 SFT 的主要用途不是教格式，而是教习惯：让模型在回答之前先花上数千个词元来审慎思考。监督信号来自蒸馏，因为人类不会写出冗长的审慎推理轨迹。

3.9.1 数据长什么样，以及需要多少

DeepSeek-AI (2025) 在从 R1 采样得到的推理轨迹上微调开源基座模型，并报告说这些蒸馏得到的模型显著优于直接在同样的小型基座模型上跑 RL 的结果——这是一个关于推理模型的行为有多大比例可以通过模仿迁移过去的有力论断。

与之相对，另有一脉工作主张极小规模的精选数据就已足够。s1 (Muennighoff et al., 2025) 按三条经消融验证的标准——难度、多样性、质量——选出 1,000 道带轨迹的题目，并报告从 Qwen2.5-32B-Instruct 微调而来的 s1-32B 在竞赛数学上以最多 27% 的幅度超过 o1-preview。LIMO (Ye et al., 2025b) 就数学推理给出了类似的论证；两者都源自 LIMA (Zhou et al., 2023a)。而与这一点相对，OpenThoughts (Guha et al., 2025) 做了一千多组受控消融，发现在匹配条件下，其大规模配比在各个数据规模上都优于那些小型精选集，其中 OpenThinker3-7B 在 AIME 2025 上达到 53%；AM-Thinking (Ji et al., 2025) 是一个规模相当的开源大语料。

如果长思维链 SFT 做的是两件可以分开的事，那么两方面的证据就都能成立。它装上一种 行为模式——长时间审慎思考、自我检查、重述问题——这会很快饱和，而这正是千例结果所探测到的东西。它同时还装上领域覆盖：模型见过在哪些题型上施加审慎思考。这一项不会饱和，而这正是百万例结果所探测到的东西。当评测接近所精选的分布、且基座模型很强时，一千条样本就够用，其他情况下则不够；请报告你处在哪种情形。

3.9.2 预算强制与长度控制

s1 为它的数据配上了一种解码时干预：通过注入思考结束分隔符来强制推理提前终止，或者通过抑制该分隔符并追加`"Wait"` 来强制它继续。Muennighoff et al. (2025) 报告说这把 AIME24 的准确率从 50% 外推到 57%，`"Wait"` 胜过他们消融过的其他选项，而且——更有意思的是——如果不用这种强制机制，仅仅提示模型遵守词元预算是行不通的，因为它无法可靠地数清自己的词元。这是留给第10章的一个有用的早期数据点：测试时计算是可控的，但不是靠指令来控制的。

3.9.3 为什么它要放在 RLVR 之前

在 RLVR 之前做长思维链 SFT 能降低所需的 RL 算力，原因是机制性的。策略已经会采样长轨迹，因此长度分布不必再靠奖励去发现。组相对估计量需要组内结果存在方差才能产生任何梯度（节 2.3），而一个已经会审慎思考的策略在难题上会立刻给出中

间水平的成功率；换成不会推理的策略，则只会得到全部失败的组，因而一无所获。而且 R1-Zero 式训练在早期若干步中忙于满足的格式奖励，此时已经被满足了。

有两种替代方案值得了解。在 SFT 之前于推理密集的语料上做中期训练，会抬高 RL 所能到达的上限，并改变基座模型对 RL 的整体响应方式 (Wang et al., 2025g)；节 4.7讨论了这一点。LUFFY (Yan et al., 2025a) 则改为在 RL 过程中注入离策略的专家轨迹，在计算优势之前把它们与同策略采样轨迹混合，并报告在纯同策略 RLVR 失败之处取得增益。在这三者之间做选择，就是在选择你能在哪里承受分布偏移。

常见陷阱: 长而不审慎的推理轨迹

长思维链蒸馏的典型失效，是学生复现了教师的语调却没有教师的计算：它写下「等等，让我再核对一下」，然后并不去核对。其结果承担了推理模型的词元成本，却只有非推理模型的准确率，而且在任何只按固定预算给最终答案打分的评测上看起来都很正常。有三项测试可以把两种情况区分开。截断敏感性：在典型推理长度的 25%、50% 与 75% 处强制给出答案；真正的审慎思考会给出陡峭的准确率曲线，装饰性的则给出平坦的曲线。扰动敏感性：破坏推理块内部的某个中间结果，检查最终答案是否随之改变。自我纠正产出率：「等等」这类词元之后有多大比例真的跟着一个被改变的中间结论，在装饰性轨迹中这个比例接近于零。在把 RL 预算花在某个检查点上之前请先跑这些测试，因为 RL 只会乐于把一个装饰性的草稿本变得更长。

本章小结

- SFT 最小化从演示分布到策略的前向 KL (3.2)。由于它是覆盖质量的，它会按坏演示在数据中的频率复现它们，无法在模式之间做选择，并且没有任何内在机制抵御死记硬背。质量的抓手是数据过滤，而不是损失函数。
- 作为行为克隆，SFT 存在误差累积，最坏情况下的遗憾为 $O(\epsilon T^2)$ (Ross and Bagnell, 2010)。在语言模型的时域上该界是空洞的；真正起作用的量是轨迹中承诺点的数目，这也是为什么 SFT 对短答案够用、对长的智能体采样轨迹不够用。
- SFT 无法教会纠错恢复，有两个彼此独立的原因：它以专家前缀而非模型自身的前缀为条件，而且演示中不包含在学生真正会到达的状态上发生的纠错事件。

- 「SFT 记忆，RL 泛化」(Chu et al., 2025a) 是在固定且狭窄的数据上测得的。同一篇论文表明 SFT 是 RL 得以启动的必要条件，而损害来自过量的 SFT；诚实的表述是存在一个金发姑娘区间，而不是一份方法排名。
- 损失归一化是一项实质性的算法选择，而不是细节：词元均值与序列均值在赋予短样本的权重上相差两个数量级，而同一个分岔会以 GRPO 归一化之争的形式在 RL 中重现。
- 对话模板是策略定义的一部分，必须在训练与服务之间逐字节一致，并与检查点一同版本化。模板不匹配会使质量弥散地退化，而在训练侧的每一项指标上都不可见。
- SFT 学习率比 RL 学习率高出一到两个数量级，因为监督梯度稠密且条件良好，监督目标不会因策略移动而失效，而且 SFT 需要走的距离更远。
- 遗忘发生得快、集中在前期，且对离数据配比最远的能力最严重。回放预训练数据与在基座权重和微调权重之间插值(3.7)都很便宜，而且都有效。
- 当适配器容量超过训练集的信息量、且被应用到所有权重矩阵上时，LoRA 与全量微调表现相当；它需要约 10 倍的学习率，并且特别适合 RL，因为策略梯度每个回合只吸收 $O(1)$ 比特 (Schulman and Thinking Machines Lab, 2025)。
- 拒绝采样微调是关于隐轨迹的 EM，等价地也是一个没有基线、没有负梯度、没有重要性校正且带硬阈值的策略梯度。每一个缺失的部件都点出了 Chapters 6 and 11要补回来的一个组件。

延伸阅读

先读Zhou et al. (2023a)，而且要带着怀疑去读：它很短，是「质量重于数量」这一共识的源头，而它的评测方法薄弱到足以让它引发的争论至今仍未平息。然后读 Lambert et al. (2025)，那是关于一份 SFT 数据配比究竟如何被组装、去污染与迭代的最完整的公开记述；其数据一节中的配比构建流程是值得研读的部分。

关于模仿的局限，Ross et al. (2011) 是那篇经典的短文，阅读时应把 节 3.2当作一份「哪些能、哪些不能迁移到词元序列上」的清单放在手边；更早的Ross and Bagnell (2010) 才包含真正的 $O(\epsilon T^2)$ 分析。Chu et al. (2025a) 是人人引用的那项实证研究，而值得细读的恰是那句口号略去的部分——关于 SFT 是 RL 得以开始的必要条件的论证，以及关于 SFT 规模的消融。

关于参数高效微调，读Hu et al. (2022) 了解方法，读 Aghajanyan et al. (2021) 了解它为何理应有效，然后把 Biderman et al. (2024a) 与Schulman and Thinking Machines Lab (2025) 对照着连读，自己推断出它们的结论之间那条容量边界落在何处；这项练习比单读其中任何一篇都更有价值。

关于拒绝采样与专家迭代，Zelikman et al. (2022) 是对该想法最清晰的表述，Singh et al. (2024) 是最细致的扩展性研究；Gulcehre et al. (2023) 给出了语言建模视角的表述。关于长思维链数据，先读Muennighoff et al. (2025) 了解极小数据的主张，再读 Guha et al. (2025) 了解限定它的系统性消融，顺序不要颠倒。

习题

3.1（定量看覆盖质量。）某演示集中有比例为 ε 的回复表现出某种不良行为。假设策略的表达能力足以精确拟合经验分布。证明训练后的策略会以概率 ε 表现出该行为，并且学习率、训练轮数与归一化因子的任何取法都改变不了这一点。然后假设随后的 RL 阶段在系数为 β 的 KL 约束下，优化一个以强度 $r = -1$ 惩罚该行为的奖励。利用第1章中的 (1.6)，求出把该比率降到 $\varepsilon/100$ 所需的 β，并就此评述过滤数据与事后补救二者的相对代价。

3.2（归一化因子的算术。）某批次含 8 条样本，其受监督长度分别为 $(20, 25, 30, 40, 60, 1200, 3000, 8000)$ 个词元。在 $L_{\max} = 8192$ 下，计算每条样本在(3.4)、(3.5)与(3.6)下各自获得的相对梯度权重。现在假设那些短样本是安全拒答，而你要求它们至少获得总梯度的 20%。求出在词元均值归一化下能实现这一点的逐领域采样权重，并解释为何这样实现优于改用序列均值。

3.3（EOS 病态。）某模型以 $L_{\max} = 4096$ 训练且保留了被截断的样本，观察到它在短答案上正常终止，在长答案上却一直跑到解码上限。写出训练数据所隐含的发出 `<eos>` 的条件概率关于位置的函数，解释所观察到的行为，并设计两个实验，把这一成因与另一种成因——`<eos>` 被完全掩出损失——区分开来。

3.4（RFT 无法压低概率。）考虑一个在三条回复上的策略，其概率为 (p_1, p_2, p_3)，其中回复 1 正确、回复 2 与 3 不正确。用 softmax 的 logits 写出 RFT 更新(3.12)，并证明在理想化极限下，一次精确的 E 步与 M 步之后回复 1 的概率为 1；但当每轮只有 G 个有限样本、且策略还必须拟合其他提示时，p_1 的增长速率正比于 p_1 本身。将其与使用组均值基线的策略梯度作对比，并指出两者差异最大的区间。

3.5（**LoRA 容量。**）利用「只要可训练适配器参数量超过受监督训练词元数，LoRA 的表现就与全量微调一致」这条规则，计算一个 8B 模型（$d_{\text{model}} = 4096$，32 层，LoRA 作用于所有注意力与 MLP 矩阵）在 2×10^8 个受监督词元上微调时所需的最小秩。然后对一次 RL 训练重复该计算：10^4 个提示，每个 16 个样本，每个样本一比特信息。评述二者之比，并指出推导中哪条假设你最不相信。

3.6（**模板审计。** ）任取一个开源的指令微调检查点。编写一个脚本，把同一段三轮对话分别通过以下三种方式渲染：（a）分词器的对话模板并设 `add_generation_prompt=True`，（b）服务栈的补全接口，（c）依照模型卡手写的字符串。在词元层面报告所有差异。然后在每种渲染下测量 200 条指令遵循条目上的准确率，并报告其离散程度。

3.7 ⋆（**SFT 是 RL 的特例吗?**）一个反复出现的说法是：SFT 优化的是一个隐式的稠密奖励，而 RL 优化的是一个显式的稀疏奖励。请把它精确化：给出一个奖励函数 r 与一个正则项，使得 KL 正则化目标(1.1)以 SFT 的解为最优解，陈述所需的假设，并判定所得的奖励是否唯一。然后利用这一构造，就「在两个目标之间做插值会是什么样子」以及「插值参数是否可解释为信赖域半径」说出一些非平凡的结论。

3.8 †（**复现损失构造的影响。**）取一个 1–3B 的基座模型，构建一份长度分布刻意双峰的 SFT 数据配比：5,000 条短的拒答式样本（< 50 词元）与 5,000 条长的推理轨迹（> 2000 词元）。训练四个检查点，只改变（a）词元均值与序列均值归一化，以及（b）带正确块对角注意力的打包与不带它的打包。在拒答遵循度、推理准确率和一项长上下文检索探针上评测这四个检查点。报告这四路比较，并对打包不正确的那两次训练，通过分别做消融来量化其退化中有多少可归因于注意力泄漏、多少可归因于位置索引漂移。

3.9 †（**把专家迭代跑到饱和。**）在你选择的一个可验证任务上、用一个 1–8B 的模型实现算法 3.1。至少跑五轮，每轮记录：验证器通过率、策略熵、幸存样本的平均长度、去重后每个提示的唯一幸存样本数，以及 pass@1 与 pass@32。找出改进停止的那一轮，并以所记录的量为证据，诊断 节 3.8中四处近似里是哪一处在起作用。然后实现能修复它的最小改动，并验证其确实有效。

第4章

后训练的数据

核心思想：2026 年的约束在数据与环境，而不在算法

本书中的每一个算法都可以用几百行代码实现。真正把一个强的后训练模型和一个弱的后训练模型区分开来的，几乎总是语料：用了哪些示范、哪些比较、哪些提示、哪些环境，以及——最重要的——哪些被丢掉了。本章把数据当作一个研究对象来处理，而不是一份下载清单。其中的每一个子问题（标注、合成、选择、难度估计、去污染）都有自己的估计量、可测量的量，以及一种特有的、悄无声息的失效方式。

「数据占主导」这一说法容易讲出口，也容易被讲得毫无内容。精确的版本是这样的。2023 年到 2026 年之间，后训练目标函数的空间收敛了：本质上每一种已发表的方法都是 (1.1) 的一个特例，而第12章中那些严肃的组相对变体之间的差距，在标准基准上不过是几个百分点。同一时期，可归因于数据配比的差距——精心整理的推理语料与爬取来的语料之间、难度与策略匹配的提示集与不匹配的提示集之间、经过去污染的评测与存在泄漏的评测之间——却是几十个百分点。当两个输入在杠杆效应上相差一个数量级时，较小的那个就不再有意思了。

4.1 后训练数据的四种类型

在类型上做些咬文嚼字是值得的，因为后训练用到的四种产物有着不同的单位、不同的成本结构，以及——通常被忽略的一点——不同的可复用性。

示范（demonstration）。 一个二元组 (x, y)，其中 y 是某人愿意让模型去模仿的回复。它被监督微调（supervised fine-tuning, SFT）（第3章）通过 y 上的交叉熵损失所消耗。这种产物本身包含答案，这正是它生产昂贵、消费廉价的原因。

表 4.1: 后训练数据的四种类型。「复用」问的是：在一个项目的生命周期里，一单位该产物能产生多少个互不相同的梯度信号；正是这一性质推动了整个领域沿着表格向下迁移。成本只是数量级估计，随领域和供应商的变化超过一个数量级；请把它们当作比例，而不是报价。

类型	单位	使用方	边际成本	复用
示范	(x, y)	SFT	人工 \$1–\$50；蒸馏 ≈ 0	一次
偏好	(x, y^+, y^-)	RM、DPO	人工 \$0.5–\$10	一次
可验证对	$(x, a^\star)$ + 检查器	RLVR	编写成本 \$0.1–\$5	无上限
环境	$(\mathcal{E}, s_0, \text{success})$	智能体式 RL	数小时至数天的工程量	无上限

偏好（preference）。 一个三元组 (x, y^+, y^-)，更一般地是对 k 个候选回复的一个排序。它被奖励模型拟合（第5章）消耗，或者被第8章中的损失直接消耗。这种产物包含的是一次比较而非一个答案，因而更容易获取，并且如节 4.2所示，带有不可约的噪声。

可验证的提示—答案对。 一个二元组 $(x, a^\star)$，外加一个接受判定谓词——字符串比较器、符号等价检查器，或者一套测试。它被 RLVR （第11章）消耗。注意这里缺少什么：没有存储任何回复。这种产物只规定了如何识别成功，回复则由策略自己在训练时现场制造。

环境与任务。 一个三元组 $(\mathcal{E}, s_0, \text{success})$：一段可执行软件、一个初始状态，以及一个程序化的成功判据，可选地附带一份只用于校验的参考解。它被智能体式 RL（Chapters 17 to 19）消耗。它的生产单位不是一条样例，而是一次集成。

表 4.1点出了结构性的要点：沿表格向下，产物包含的答案越来越少，包含的规范越来越多。一条示范只被消耗一次——它永远贡献同一个梯度方向，一旦策略把它模仿好了，这条样例就失效了。而一个可验证提示是训练信号的生成器，而不是训练信号的一个单位：策略每变一次，同一个提示就产生新的采样轨迹、新的奖励和新的梯度。这就是从供给侧看到的节 1.2中的验证不对称性，它解释了为什么尽管单位成本更高，预算仍然转向了提示编写与环境建设。你买的是一条流，而不是一个样本。

代价在于：流可以被耗尽，而样本不会。一旦策略在每次尝试中都能解出某个提示，它就完全不再产生梯度（节 4.4.5），所以一个可验证语料的有效规模会随着训练的成功而缩小——这是节 4.5的主题。

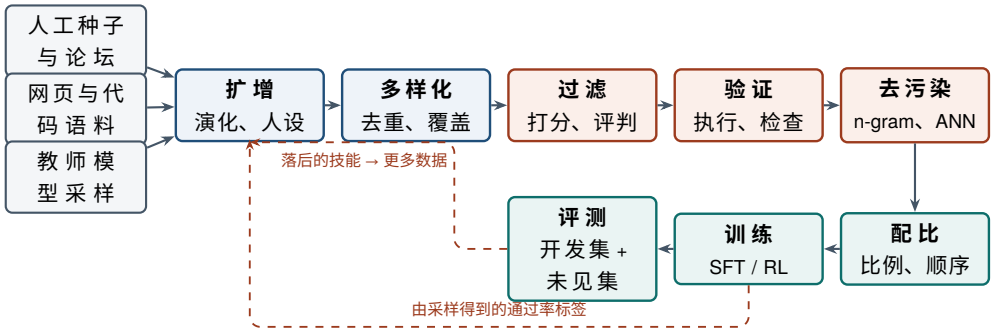


图 4.1: 作为工程流水线的数据生命周期。实线箭头是数据流；虚线箭头是把「持续维护的语料」与「下载来的语料」区分开的两条反馈回路。下面那条回路——把当前策略实测的通过率反馈到提示选择中——是最常缺失的一条，而节 4.5将论证它也是最有价值的一条。

4.2 人工标注及其统计性质

人类偏好数据是本书中唯一一种噪声下界由我们无法通过工程手段绕开的因素所决定的信号。因此值得认真地把它测量清楚。

4.2.1 实际观测到的一致率

已发表的数字出奇地一致，也出奇地低。Stiennon et al. (2020) 报告在摘要比较上研究者与研究者之间的一致率为 $73 \pm 4\%$。Ouyang et al. (2022) 报告参与训练的标注员之间有 $72.6 \pm 1.5\%$ 的时间意见一致，留出的标注员为 $77.3 \pm 1.3\%$，并指出整体的标注员间一致率「约为 73%」。Bai et al. (2022a) 说得更直白，称 Anthropic 研究者与其众包标注员之间「平均一致性很差（约 63%）」，并把部分差距归因于他们开放式的收集协议。在这个定义了基于偏好的对齐之目标的任务上，两个称职的人被问及哪个回复更有帮助时，会有四分之一到三分之一的时候意见不一致。

这些都是二元或三元选择上的原始一致率，把它作为头条统计量是不合适的——抛硬币和抛硬币也有一半时间是一致的。2026 年一项大规模的评判模型审计给出了定量说明：在 MT-Bench 上，0.80–0.85 的原始精确匹配一致率对应的 Cohen's $\kappa \approx 0.48$，接近四十个百分点的落差，这是基准标签边际分布的性质，而非评判模型质量的性质 (Anonymous, 2026b)。请在报告原始一致率的同时报告一个经机会校正的系数；第24章会再回到这一点。

4.2.2 从一致率到准确率上限

尽管如此，仅凭原始一致率就足以界定一个奖励模型最好能有多好，而这是每个从业者都应该会做的一项计算。

假设 4.1 (可交换的标注员)**.** 对每个条目 $z = (x, y_A, y_B)$，存在一个数 $p(z) \in [0,1]$，使得从标注池中随机抽取的一名标注员以概率 $p(z)$ 偏好 y_A，且各次抽取之间相互独立。

定理 4.1是标准模型，而它以一种特定的、重要的方式是错的，节 4.2.4会讨论：共享的指南和共享的文化先验使标注员之间相关，因此看上去像信号的一部分其实是共同偏差。眼下先把它当作给定条件。

同一条目上的两名独立标注员意见一致的概率为 $p^2 + (1-p)^2 = 1 - 2p(1-p)$，因此总体一致率为

$$a \;=\; 1 - 2\,\mathop{\mathbb{E}}_{z}\big[p(z)(1-p(z))\big]. \tag{4.1}$$

任何分类器在一个新的单标注员标签上所能做到的最好情况，就是预测多数派偏好，其正确概率为 $\max(p, 1-p)$。于是偏好分类的贝叶斯误差为

$$\mathrm{Err}^{\star} \;=\; \mathop{\mathbb{E}}_{z}\big[\min(p(z), 1-p(z))\big]. \tag{4.2}$$

命题 4.2 (一致率给出可达准确率的上下界)**.** 在定理 *4.1*下，若观测到的一致率为 $a \in [\tfrac{1}{2}, 1]$，则任何偏好分类器在单标注员标签上评测所得的贝叶斯准确率 $1 - \mathrm{Err}^{\star}$ 满足

$$a \;\le\; 1 - \mathrm{Err}^{\star} \;\le\; \tfrac{1}{2}\Big(1 + \sqrt{2a - 1}\Big), \tag{4.3}$$

其中下界在每个条目要么意见完全一致、要么是纯粹抛硬币时取到，上界在所有条目具有相同 p 时取到。

证明. 记 $u(z) = \min(p, 1-p) \in [0, \tfrac{1}{2}]$ 及 $g(u) = 2u(1-u)$，于是 (4.1)即 $\mathbb{E}[g(u)] = 1 - a =: c$，且 $\mathrm{Err}^{\star} = \mathbb{E}[u]$。由于 g 在 $[0, \tfrac{1}{2}]$ 上是凹的，Jensen 不等式给出 $c \le g(\mathbb{E}[u])$，因而 $\mathbb{E}[u] \ge g^{-1}(c) = \tfrac{1}{2}(1 - \sqrt{2a-1})$，用 1 减去它即得 (4.3)中的上界，等号成立当且仅当 u 退化。反过来，在 $[0, \tfrac{1}{2}]$ 上的分布中，在线性约束 $\mathbb{E}[g(u)] = c$ 下最大化线性泛函 $\mathbb{E}[u]$，其最大值在一个两点分布上取到；又因为 $g(u)/u = 2(1-u)$ 递减，最大值把质量 λ 放在 $u = \tfrac{1}{2}$ 处、其余放在 0 处；于是 $\lambda/2 = c$ 给出 $\mathbb{E}[u] = c = 1 - a$。 □

代入已发表的数字：73% 的一致率意味着任何奖励模型，无论多大，在来自同一标注池的留出单标注员标签上的准确率都不可能超过 73–84%；63% 则把上限压到 63–76%。

区间本身的宽度也很说明问题——上限无法仅由一致率识别，因为同一个 a 既与「每个条目都略有歧义」相容，也与「大多数条目一目了然、少数条目纯属抛硬币」相容。要区分这两者需要每个条目有多个标签，而几乎没有人会大规模地收集它们；InstructGPT 就指出，出于成本考虑其大部分比较都只由一名承包标注员标注。

这就是为什么应当对精心整理的偏好基准上高达 80 多、90 多的奖励模型准确率保持怀疑 (Lambert et al., 2024)。这样的数字并非不可能，但在定理 4.2 之下它们意味着该基准的条目远比生产环境中的标注更少歧义——事实也确实如此，因为精心整理的基准会刻意挑选界限分明的样本对——所以它们不能被读作关于「奖励作弊真正发生的那片模糊地带」的证据。第5章会给出应对办法：集成、校准，以及在低间隔样本对上弃答。

核心思想：聚合去除的是方差，不是偏差

每个条目收集 m 个标签并取多数，会让训练目标更干净：聚合结果与 $\mathbb{1}[p > \frac{1}{2}]$ 不一致的概率随 m 指数衰减，衰减率由 $\left|p - \frac{1}{2}\right|$ 决定。但它并不会抬高定理 4.2中的上限，因为该上限关乎的是那些 p 确实接近 $\frac{1}{2}$ 的条目，再多的投票也无法改变它们。更糟的是，节 4.2.4中的系统性偏差是标注员之间共享的，因此它们根本不会按 $1/\sqrt{m}$ 收缩；它们已经被烘焙进 p 本身。花五倍的钱买五个标签，买到的是一个有偏量的低方差估计。

4.2.3 标注指南作为一份隐式规范

在一次偏好收集工作中，最具决定性的产物不是数据，而是那份指南文档。Ouyang et al. (2022) 说得很直白：「我们编写了标注员据以工作的标注指南。」奖励模型学到的关于如何在有用性与无害性之间取舍、多少含糊其辞算恰当、附带解释的拒答是否胜过不附解释的拒答，全都是四十个人在时间压力下对那份文档的一次压缩解读。

由此有两个后果。评分细则中沉默的地方会变成奖励模型的盲区，而优化该奖励的策略会找到它们——这正是节 5.8中规范错误的那一半，再多的数据也修不好，因为额外的数据是在同一份细则下产生的。此外，在收集过程中修改文档会造成一个下游不可见的非平稳标签分布：奖励模型看到的是一份内部含有未被建模的制度切换的数据集，其症状是在一个主要来自某一制度的留出集上出现无法解释的准确率平台期。请给细则打版本号，为每个标签盖上版本戳，并把一次细则改动当作一次数据集边界来处理。Casper et al. (2023a) 对这一类及相邻的失效模式做了综述，至今仍是关于人类反馈之局限性的最佳单篇参考。

4.2.4 系统性偏差

有四种偏差被重复验证得足够多，值得据以做规划。

长度。 更长的回复被偏好的比率远高于其质量差距，因此在这类数据上训练的奖励模型会把长度编码为质量的代理。Singhal et al. (2024) 在若干 RLHF 数据集上记录了这种相关性，并表明 RLHF 期间表观上的奖励提升有很大一部分仅由长度就能解释；Dubois et al. (2024) 的回应是为 AlpacaEval 提出一个长度受控的估计量，其修正之大足以改变模型排序。

格式。 Markdown 标题、项目符号列表和加粗小结会独立于内容抬高被偏好的比率——这是最廉价的奖励作弊，而且在任何偏好优化的训练中几百步之内就会出现。

断言性与表观自信。 一个流畅、自信的错误答案击败一个含糊其辞的正确答案的频率，高于它应有的水平。Anthropic 的偏好模型分析在其被自信地错误分类的离群样本中发现的正是这一点：声明自己不知道的回复输给了包含误导信息的「听起来高深」的回复 (Bai et al., 2022a)。

谄媚。 Sharma et al. (2024) 表明偏好数据会主动奖励顺着用户已表明的观点说话，并且这是偏好信号本身的性质，而不是某一条收集流水线的性质：优化一个拟合人类比较的 Bradley–Terry 模型会增加谄媚。这是目前能拿到的、说明噪声并非零均值的最干净的证据。

这四种偏差在标注员之间都是相关的，这也正是前面那个关于聚合的论证之所以重要的原因。它们同样解释了为什么 2023 年收集的偏好数据无法干净地迁移到 2026 年的模型上：这些偏差是相对于某个候选回复分布而言的，而那个分布已经移动了。

4.3 合成指令数据

到 2026 年，绝大多数有监督的后训练数据都是机器生成的。相关文献数量庞大 (Wang et al., 2024c)，但有四篇论文定义了整个设计空间，而且每一篇都移除了其前作仍然存在的一项依赖。

Self-Instruct (Wang et al., 2023b) 从 175 条人工编写的种子任务出发做自举：模型先提出新指令，再为它们生成输入和输出，并用 ROUGE-L 做近重复过滤。把它用在原始的 GPT-3 上，在 Super-NaturalInstructions 上带来了 33 个百分点的绝对提升。Alpaca 的 52K 数据集是最知名的实例化，也同时是一个警示故事：其中大约五分之一的指令彼此近乎重复。

Evol-Instruct (Xu et al., 2024a) 移除了难度上限。由一个语言模型沿两个维度改写已有指令——深度方向（添加约束、添加推理步骤）与广度方向（发明新的长尾任务）——从而给出了 Self-Instruct 所缺乏的、可调节的难度控制。从 Alpaca 的 52K 出发做四轮，得到 250K 条指令，而其中 70K 的子采样胜过了等量的真实 ShareGPT 数据。

Magpie (Xu et al., 2025) 移除了种子集与提示。只把预查询模板喂给一个已对齐的模型——也就是用户消息开始之前的那段字面前缀——然后让它自回归下去：它会写出一条看似合理的用户指令，接着写出自己的回复。这个机制同时解释了它的长处和短处。一个指令微调过的模型是在形如 $T_{\text{pre}} \oplus x \oplus T_{\text{mid}} \oplus y$ 的字符串上拟合的，因此仅以 T_{pre} 为条件采样得到的是该模型学到的指令边缘分布 $\pi_\theta(x \mid T_{\text{pre}})$。Magpie 并不是在从一个提示生成指令；它是在 从教师的指令分布中抽样。因此不需要种子、也不需要提示工程——从 Llama-3-Instruct 得到四百万对，过滤到 300K，据报告只花了 61 GPU 小时且无需人工干预——同时也因此，它精确地继承了教师的覆盖范围，包括其对推理类任务的欠表达，而作者用一份单独的数学与代码增强集来打补丁。

人设驱动的生成 (Ge et al., 2024) 移除了多样性上限。十亿个自动整理出的人设充当条件变量，几乎可以与任何合成提示组合，因此输出的多样性受限于人设集合，而不再受限于某个种子语料。

指令回译 (Li et al., 2024c) 把方向反了过来：从网页语料中取一份高质量的人工撰写文档 y，为它生成一条它会是好回复的、看似合理的指令 x，然后打分并保留最好的样本对。这里的机制值得注意——SFT 模仿的是回复，所以回译把合成误差放在了指令上，也就是放在了更廉价的那一侧。它是这个家族里唯一一个回复并非模型输出的成员，这使它成为对下文那些坍塌担忧的自然回应。

4.3.1 通用流程以及各阶段的失效方式

文献中的每一条流水线都是同五个阶段的一个实例，把它们连同各自的失效模式一起讲清楚，比罗列数据集更有用。

1. **播种**。人工任务、论坛问题、网页文档、基准的训练划分。失效：种子的主题分布会悄悄地为下游的一切设置上界。如果没有任何种子涉及数值表格，那么再多的扩增也产生不出表格推理能力。
2. **扩增**。由少生多，或由易生难。失效：近重复，它会在不贡献任何新梯度方向的前提下抬高名义数据集规模；此外还有难度漂移，方向取决于生成器觉得哪边容易。
3. **多样化**。以人设、领域、语言、难度目标或输出格式为条件；并做去重。失效：表层多样性——表面特征在变，而潜在的任务分布并没有变。请在一个能预测迁移效果的空间里度量（任务类型、所需工具、解答长度），而不是在改写看起来彼此不同的原始嵌入空间里度量。
4. **过滤**。奖励模型分数、LLM 评判模型、困惑度或 IFD 阈值、启发式规则。失效：过滤器会继承生成器的偏好；当生成器与评判模型同属一个模型家族时，这就是自我偏好，而且它是可测量的。
5. **验证**。执行代码、跑测试、核对答案、用符号方法重新推导。失效：只有在验证不对称性成立的地方才可用，而且会受验证器漏洞的影响（节 5.8）。在它确实可用的地方，它胜过其他一切：永远不要用奖励模型的分数去替代一次执行结果。

常见陷阱: 用生成合成数据的那个模型来过滤这些数据

如果 GPT 家族的输出由一个 GPT 家族的评判模型来排序，那么幸存下来的集合就会偏向该家族的特异之处，而任何使用同一评判模型的下游评测都会报出被抬高的分数。Li et al. (2025a) 把这称为偏好泄漏，表明它在 AlpacaEval 2.0 和 Arena-Hard 中普遍存在，并提出了一个 Error Bias 指标；在他们测试的各种缓解手段中，针对一个留出的偏差校正集做上下文校准最为有效，据他们报告把 Error Bias 从 17.8 降到了 7.3。廉价的结构性修法是使用与生成器不同模型家族的评判模型，更廉价的做法则是：只要存在验证器，就用验证器。

表 4.2: 截至 2026 年年中具有代表性的开源后训练数据产物。规模均为发布方作者所报告的数字。对规划而言最要紧的一列是「教师」：每一件合成产物都是某一个模型行为的有损投影，所以两个名义任务覆盖相同的数据集，若其教师不同，就并不可以互换。

产物	规模	类型	教师 / 来源
LIMA (Zhou et al., 2023a)	1K	示范	人工：论坛、手工编写
s1K (Muennighoff et al., 2025)	1K	示范	蒸馏轨迹，按 3 条准则整理
LIMO (Ye et al., 2025b)	小规模	示范	精心整理的推理示范
Magpie (Xu et al., 2025)	4M → 300K	示范	Llama-3-Instruct，自诱导
Persona Hub (Ge et al., 2024)	10^9 个人设	条件变量	网页整理的人设描述
Tülu 3 SFT (Lambert et al., 2025)	~1M 条提示	示范	人工与合成混合
OpenThoughts3 (Guha et al., 2025)	1.2M	示范	QwQ-32B，1000+ 次消融
Llama-Nemotron (NVIDIA, 2025)	~30M	示范	Qwen-2.5-Math-72B、DeepSeek-R1 等
Dolci (Team OLMo, 2025)	分阶段配比	全部四类	逐阶段有文档记录，已去污染
SETA-Env (Shen et al., 2026b)	4,567 个环境	环境	合成并演化，已验证

4.3.2 开源生态中实际有些什么

关于表 4.2有两点是从任何单次发布中都难以看出来的。规模跨度达四个数量级，而这些产物在如此跨度上并不可替代：拿一个 1K 的精选集与一个 30M 的蒸馏集相比，是在拿一份对肤浅对齐假说的演示去和一台覆盖引擎相比。而且大规模数据集的分布极其偏斜——Llama-Nemotron 的集合大致是 19.8M 数学与 9.6M 代码样本，对应 709K 科学、56K 指令跟随、40K 对话和 31K 安全 (NVIDIA, 2025)，因此均匀地加载它会得到一个三分之二是数学的配比，这是一个你若不去看就会在无意中做出的决定。

4.3.3 自我消费与模型坍塌

如果合成数据如今已占后训练数据的多数，而模型又是在模型的输出上训练的，那么这个过程会退化吗?

悲观的结论是真实的。Shumailov et al. (2023) 表明，当每一代模型都在上一代模型的输出上训练、且原始数据被丢弃时，分布的尾部最先消失，方差收缩，过程最终收敛到一个退化分布——即「模型坍塌」。Dohmatob et al. (2024) 把这一点锐化成了关于缩放律的陈述：在有限的合成语料上训练会给可达损失设下一个地板，因此通常的幂律改进会走平而不是继续下去。

乐观的结论同样真实，却较少被引用。Gerstgrasser et al. (2024) 表明，如果合成数据是与真实数据累积在一起而不是替代它，那么发散不会发生，测试误差保持有界。既然累积正是每个从业者实际所做的事——没人会删掉自己的人工数据——那么那些强坍塌结论就应当被读作对一个无人真正运行的过程所加的边界条件。这一发现中适用的版本，弱于那个出名的版本。

从机制而非从民间传说出发，可以得到三条缓解措施。持续注入新鲜的人工数据，这是唯一不在模型之中的信息来源。优先用验证而非评判，因为验证器的信息来自生成器之外，确实能给这个回路增加熵。以及直接对多样性做仪表化监测，因为它会在流水线的每一个阶段悄悄退化——扩增产生近重复，过滤削掉尾部，反向 KL 式的后训练令策略集中（节 2.4）——而与此同时固定基准上的准确率一路上升。每百万词元的不同 n-gram 数能抓住模板固化；固定半径下的簇数量正是 k-center 选择 (Sener and Savarese, 2018) 所优化的量，而且与平均成对距离不同，它不会把一个稠密簇藏在一片宽阔的分布里；至于粗粒度分类器下的任务标签熵（这个条目需要算术、代码执行、检索还是多步规划?），如果你只报告一个指标，就报告它，因为它是唯一一个在能预测迁移的空间中测得的指标。等到多样性的损失以 pass@k 的退化形式浮现出来时（节 2.8），算力已经花掉了。

4.4 选择：少即是多，以及何时并非如此

4.4.1 关于数据高效性的主张

Zhou et al. (2023a) 用 1,000 条精心整理的提示—回复对——750 条高质量论坛回答外加 250 条为多样性和统一风格而手工编写的样本，大约 750K 个词元——微调了一个 65B 的 LLaMA，不用 RL，也不做偏好建模，报告在人工比较中有 43% 的情况下追平或胜过 GPT-4，65% 的情况下胜过 DaVinci003。他们的肤浅对齐假说主张：几乎全部知识都是在预训练中获得的，指令微调只需要教会模型该使用哪一个子分布。

推理时代产生了类似的结果。Muennighoff et al. (2025) 按三条经消融验证的准则（难度、多样性、质量）整理出 1,000 个带推理轨迹的问题，并报告 s1-32B 在竞赛数学上以最多 27% 的幅度超过 o1-preview；Ye et al. (2025b) 对推理提出了同样的论点。二者现在都是标准的「小规模精选集」基线。

历史注记: 从生成指令到选择指令，再回到生成

这个领域的注意力摆动过两次。整个 2023 年初，问题被框定为生成：Self-Instruct 及其后继之所以存在，是因为指令数据不够多。LIMA 把它重新框定为选择，2023–2024 年由此产生了大量关于打分与子集选择的文献。随后推理转向又把它重新框定为生成——长思维链轨迹必须被生产出来，而不是被找出来——到 2026 年，二者已被认识为同一个优化问题：你之所以生成一个大的候选池，恰恰是为了让选择有东西可挑，而评判的标准是「池子加过滤器」这个组合，而不是其中任何一半。

4.4.2 反面证据说明了什么

最细致的一次检验是Guha et al. (2025)，他们运行了一千多次受控消融，逐一改变推理数据流水线的每一步，并在匹配规模下同时对照大型 SFT 语料以及 s1.1 和 LIMO 这两个小规模精选集做基准测试。他们 1.2M 样例的 OpenThoughts3 配比在各个规模上都压过小规模精选集，其 7B 模型在 AIME 2025 上达到 53%、在 LiveCodeBench 上达到 51%。规模赢了。

调和之道并不是说某一方错了，而是——用节 1.1的词汇来说——这两支文献测量的是不同的差距。Lin et al. (2024a) 给出了机制层面的版本：逐词元比较基座模型与对齐模型，对齐微调所诱发的变化集中在风格和篇章类词元上，而承载知识的位置基本没变。一千条样例能装上格式、语域和「作答」这个习惯——也就是格式差距。它们装不上对成千上万种问题类型的覆盖，而那正是能力差距所要求的。

一个测量层面的因素强化了这一点。LIMA 时代的证据是开放式提示上的成对偏好判断，这是一个由风格和长度主导的指标（节 4.2.4）；OpenThoughts 时代的证据则是竞赛数学与可执行代码上的精确匹配准确率，这是一个对风格无动于衷的指标。表观矛盾中有一部分只是因变量换了。可据以规划的诚实结论是：数据高效性的主张在格式与风格上被很好地复现了，在能力上则复现不了。

4.4.3 选择方法

表 4.3概括了这些方法；其中三种值得讲讲机制。

指令跟随难度 (Li et al., 2024b) 比较模型在有指令与无指令两种情况下预测同一回复的好坏。记 $s(y \mid x)$ 为每词元平均对数似然，则该分数为 $\text{IFD} = s(y \mid x)/s(y)$；取值接近 1 意味着指令没有帮助，该方法把这读作「难，因而有信息量」。失效模式直接来

表 4.3: 有监督数据的选择方法，按成本递增排列。「需要目标」意味着该方法必须被告知想要的下游能力是什么；需要目标的方法在目标已知时更强，在构建通用配比时则毫无用处。RL 那一行在性质上不同，是节 4.4.5的主题。

方法	信号	需要目标	特征性失效
困惑度	参考模型下的 $-\log p(y \mid x)$	否	把噪声和乱码当作「难」而选中
IFD (Li et al., 2024b)	y 上条件损失与无条件损失之比	否	同上，另加对短答案的偏见
k-center (Sener and Savarese, 2018)	贪心的嵌入覆盖半径	否	在错误的度量空间里做覆盖
DEITA (Liu et al., 2024f)	学习到的复杂度 × 质量，再加多样性阈值	否	打分器继承其训练者的偏差
奖励模型分数	$r_\phi(x, y)$，保留高分位	部分	长度与格式偏差（节 4.2.4）
LESS (Xia et al., 2024)	$\langle \nabla \mathcal{L}(z), \nabla \mathcal{L}(\mathcal{D}_{\text{val}}) \rangle$，经 Adam 校正	是	数据存储成本；需要一个目标集
通过率	当前策略下的 $\hat{p}$	是	标签随策略变强而陈旧

自定义：被破坏的、被截断的或者根本不知所云的回复同样无法从其指令中被预测出来，所以一个纯 IFD 过滤器在没有质量闸门的情况下就是一个噪声放大器。

LESS(Xia et al., 2024) 是把影响函数方法在这里做对了的一个工作。把 z 纳入训练对验证损失的一阶效应约为 $\langle \nabla \mathcal{L}(z), \nabla \mathcal{L}(\mathcal{D}_{\text{val}}) \rangle$；该论文把它适配到 Adam 而非 SGD，用梯度范数做归一化以消除对长指令的偏见，并存储低维梯度投影以降低内积的计算成本。有两项发现在实践上很重要：LESS 选出的 5% 往往胜过全集，而且选择结果可以迁移，因此小模型可以为大模型挑数据。基于频率和表示的基线（BM25、DSIR）仅仅勉强胜过随机——这是关于表层形式选择的一个有用的负面结果。

DEITA(Liu et al., 2024f) 是实践中的主力：先对复杂度和质量打分，排序，然后沿排好序的列表往下走，只有当一条样例到最近的已录取样例的嵌入距离超过阈值时才录取它。这种「先打分、再顾及多样性」的模式正是大多数生产配比所实现的做法，作者报告一个 6K 样例的集合在 MT-Bench 上达到 7.55，规模大约是它所匹敌的那些基线的十分之一。

4.4.4 配比：比例与顺序

选择决定哪些样例存在；配比决定优化器看到它们的频率。有两个事实组织起了这里的实践。权重不是一个归一化的细节：在词元求和的损失下，一个切片的影响力正比于它的词元数而非样例数，因此长推理轨迹可能压过一个名义上更大的短答案切片——请同时以词元数和样例数报告权重。而已发表的构造流程中最有用的是 Tülu 3 的做法（Lambert et al., 2025）：为每一项目标技能分别构建一个配比，保留在孤立情况下使该技能最大化的那个，以逼近该技能的单项上限；再把这些优胜者组合起来，然后迭代，修补落后的技能，同时做去污染并对过大的切片做下采样。这种做法之所以胜过直接搜索权重，是因为它把一个高维联合优化解耦成了若干一维优化外加一个修补回路；它昂贵，是因为每一次迭代都是一次训练——因此才有「在小模型上调、再迁移权重」的做法，而这是一种没有人仔细验证过的外推。请把开发用套件和未见套件分开；一个针对某套件反复迭代出来的配比，已经在不知不觉中被拟合到了它上面。

4.4.5 面向 RL 的提示选择，及其为何占主导

对强化学习来说，相关的产物是提示而不是示范，而杠杆最大的整理工作是按当前策略下的通过率来做选择。理由是算术上的，而非经验上的。

设某个提示在当前策略下的单次成功概率为 p，抽取一组 G 个独立回复，二元奖励为 $r_i \in \{0, 1\}$。一个组相对估计量构造优势 $A_i \propto r_i - \hat{\mu}$，其中 $\hat{\mu} = G^{-1} \sum_j r_j$。若全部 G 个奖励都相等——每个样本都对，或每个样本都错——那么对每个 i 都有 $A_i = 0$，该提示贡献恰好为零的梯度，无论它本身多么有趣。这样一个退化组出现的概率为

$$\mathbb{P}[\text{no signal}] \;=\; p^G + (1-p)^G. \tag{4.4}$$

数字很严峻。在 $G = 16$、$p = 0.02$ 时——对一个中等强度的策略来说是一道未解出的竞赛题——(4.4)给出 $0.98^{16} \approx 0.72$，也就是说花在这类提示上的采样预算有 72% 什么也没产生。在 $p = 0.98$ 时由对称性同样成立。而在 $p = 0.5$ 时，退化概率是 $2 \cdot 2^{-16} \approx 3 \times 10^{-5}$。

核心思想：提示整理是一个系统杠杆，而不是数据质量上的锦上添花

采样生成占据了一次 RL 训练中大部分的墙钟时间（节 1.6），因此一个通过率集中在 0 或 1 附近的提示池，会按 $p^G + (1-p)^G$ 的比例直接浪费这些时间，而改进提示池会把有效吞吐乘上「可用比例」的倒数。把可用组从 30% 提到 90%，就是在完全不动训练器的情况下获得 3× 的加速。这正是为什么动态采样——不断重

采样直到批次中有足够多的非退化组——在 DAPO (Yu et al., 2025c) 中是承重的，并且如今已成为标准做法。第11章把它当作一个算法来处理；但它首先是一个数据决策。

4.5 难度、可学习性与课程

(4.4)告诉我们哪些提示是无用的。自然的下一个问题是哪些提示最有用，而对二元奖励来说这个问题有一个干净的答案。

定义 4.3 (提示的可学习性)**.** 对于在当前策略下成功概率为 p 且奖励为二元的提示，其可学习性定义为奖励方差 $\ell(x) = \mathbb{V}\mathrm{ar}[r] = p(1-p)$，在 $p = \frac{1}{2}$ 处取到最大值 $\frac{1}{4}$。

这不只是一条看起来合理的启发式；它就是学习信号的期望量级。

推导: 期望平方优势恰为 $\frac{G-1}{G}p(1-p)$

取 $\hat{\mu} = G^{-1}\sum_j r_j$，$r_j$ 为 i.i.d. Bernoulli(p)，写出 $r_i - \hat{\mu} = \frac{G-1}{G}r_i - \frac{1}{G}\sum_{j\neq i} r_j$。其均值为零，由独立性

$$\mathbb{V}\mathrm{ar}[r_i - \hat{\mu}] = \frac{(G-1)^2}{G^2}\sigma^2 + \frac{G-1}{G^2}\sigma^2 = \frac{G-1}{G}\,\sigma^2, \qquad \sigma^2 = p(1-p).$$

因此一个提示的期望平方优势——也就是给该提示所贡献的平方梯度定标的那个量——恰好正比于 $p(1-p)$，其中退化情形 $p \in \{0,1\}$ 作为方差消失的特例回到了(4.4)。

有一点保留意见会改变结论。原始 GRPO 的优势要除以样本标准差 $\hat{\sigma}$，这抵消了 $p(1-p)$ 的定标，使每个非退化提示都留下一个 $O(1)$ 的优势。因此这种归一化反而上调了接近极端处的提示的权重——这恰恰是Liu et al. (2025g) 所指出的题目层面难度偏差。于是 $p(1-p)$ 是否起到权重的作用取决于估计量；不取决于估计量的是：$p \in \{0,1\}$ 什么也不贡献。Foster et al. (2025) 把这个定义照字面执行，按 $p(1-p)$ 的在线估计值成比例地采样提示。

4.5.1 在线估计 p，以及难度标签的陈旧度

估计量很平凡：由 n 次采样中的 c 次成功得到 $\hat{p} = c/n$，且 $\mathbb{V}\mathrm{ar}[\hat{p}] = p(1-p)/n$。实质内容在样本量上。在 $p = \frac{1}{2}$、$n = 16$ 时标准误是 0.125——足以把提示分成三个粗桶，远

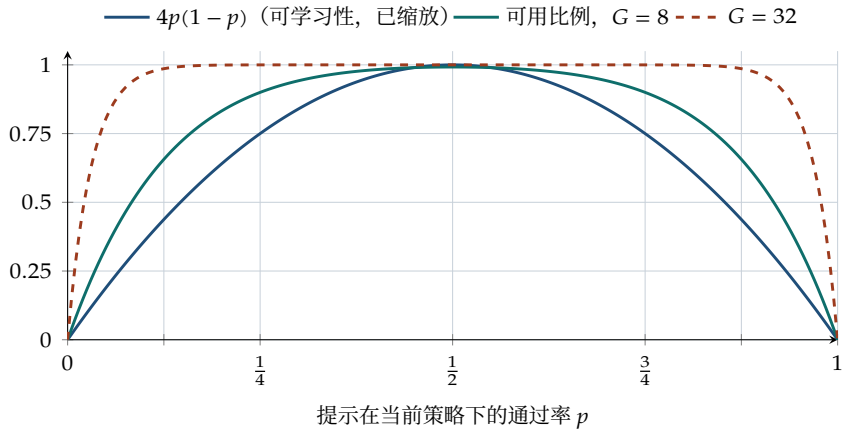


图 4.2: 可学习性与可用组比例作为通过率 p 的精确函数，来自定理 4.3 与(4.4)。图中没有绘制任何经验数据。两条曲线在「对 $p \approx \frac{1}{2}$ 的偏好应当多么尖锐」这一点上并不一致：拿到某种信号的概率在一条很宽的中间带上是平的，只在两端附近才崩塌，而且这条带随 G 增大而变宽；相比之下信号的量级确实在 $\frac{1}{2}$ 处最大。因此，仅仅过滤掉退化组是廉价的，几乎不带来分布上的副作用；而强烈偏好 $p \approx \frac{1}{2}$ 则是一种更强的干预，会收窄训练分布。

不足以区分 $p = 0.4$ 与 $p = 0.5$。要把一个提示定位在 ± 0.05 之内需要 $n \approx 100$ 次采样，对长思维链而言这是一个步的预算中不可忽略的一部分。这正是为什么你已经生成的那一组就是正确的估计量：它免费给出了 $n = G$ 下的 $\hat{p}$。在构建提示池时，以 $n = 100$ 做一次离线标注是站得住脚的一次性投入；每隔几百步就刷新一遍则不是。

常见陷阱: 你上周缓存的那个难度标签是错的

可学习性是相对于当前策略而言的，而在一次成功的 RL 训练中，整个通过率分布都会向右平移。一个相对 SFT 检查点测得 $p = 0.1$ 的提示，五百步之后可能已经到了 $p = 0.7$，从「太难」穿过有用带走向「已解决」。因此任何基于离线标签构建的静态课程都会衰减，而且恰恰在训练进展顺利时衰减得最快。它的症状——批次中退化组的比例上升而奖励停滞——很容易被误认成熵坍缩（节 12.6），所以请把两者分别记录；它们的解法不同。

4.5.2 实用的课程方案

有四种方案在常用，按侵入性递增排列。静态分桶只对着 SFT 检查点给提示池标注一次，然后由易到难地训练，接受其陈旧度——便宜，对短训练足够。在线过滤以 DAPO 的动态采样为典范，它超量采样提示并丢弃退化组，直到批次填满：按构造零陈旧度，

代价是每步的采样预算可变且难以排期。可学习性采样 (Foster et al., 2025) 为每个提示维护一个 p 的滑动估计，并按 $\hat{p}(1-\hat{p})$ 成比例采样。最后是 难度扩展：当整个提示池都饱和时，任何在其上做的方案都无济于事，问题重新变回生成——要更难的提示，或者在更严格的验证器下用同样的提示。这个转折点决定了一项数据工程是复利增长还是停滞不前。

4.5.3 一个错误的标准答案看起来和一道难题一模一样

上面的一切都假定通过率如其字面所示。一个参考答案本身就是错的提示，或者其验证器会拒绝一个正确但格式不同的答案的提示，测出来的 $p \approx 0$——按通过率与一道真正未被解出的题无法区分，但作为训练数据却并不等价。难题是惰性的；标错答案的题却是主动有害的，因为在策略偶尔给出正确答案的那些少数情况下，它相对于自己的错误答案反而获得了负优势。

用一个你本来就在收集的信号，就能轻松把二者分开。从可用的最强模型采样 n 个回复，检查最终答案的众数：一道真正的难题会给出一个众数质量很低的弥散分布，而一道标错答案的题会给出一个集中但与标准答案不符的分布。因此，凡是众数答案占据至少一半质量且与标准答案不同的提示，都应当被标记出来——这是NVIDIA (2025) 用来剔除不一致条目和容易蒙对的条目的那类过滤的一个变体。其量级比大多数人以为的要大：Huang et al. (2026c) 报告，一个语言模型评判者与他们手工编写的功能性验证器不一致的比例为 1.4%，而某个前代基准所继承的测试则为 32.4%；三分之一的奖励信号可被质疑，这是另一个层次的训练问题，而不是一个舍入误差。请像审计采样轨迹那样审计标准答案（节 5.8.4），并从 $p = 0$ 那一桶开始。

代码 4.1: 从你已经付过钱的采样中做通过率整理。估计量就是采样组本身，所以难度标签是训练的副产品，而不是另开一遍的处理。指数滑动平均是让标签不至于陈旧的关键。

```python
class PassRateBook:
    """Tracks a decaying estimate of p per prompt from training rollouts."""

    def __init__(self, prompt_ids, decay=0.7, prior=0.5, prior_weight=2.0):
        self.p = {i: prior for i in prompt_ids}
        self.n = {i: prior_weight for i in prompt_ids}
        self.decay = decay

    def update(self, prompt_id, rewards):
        """rewards: 1-D tensor of binary rewards for one group of G rollouts."""
        g = rewards.numel()
        p_hat = rewards.float().mean().item()
```

```
        # 先衰减旧证据，再把新的一组折算进来。
        n_old = self.decay * self.n[prompt_id]
        self.p[prompt_id] = (n_old * self.p[prompt_id] + g * p_hat) / (n_old + g)
        self.n[prompt_id] = n_old + g

    def learnability(self, prompt_id):
        p = self.p[prompt_id]
        return p * (1.0 - p)

    def sample_batch(self, batch_size, lo=0.05, hi=0.95, temperature=1.0):
        """Drop saturated prompts, then sample the rest by learnability."""
        live = [i for i, p in self.p.items() if lo < p < hi]
        if not live:
            raise RuntimeError("prompt pool saturated: need harder prompts")
        w = torch.tensor([self.learnability(i) for i in live]) ** temperature
        idx = torch.multinomial(w / w.sum(), batch_size, replacement=False)
        return [live[j] for j in idx.tolist()]
```

4.6 去污染

污染（contamination）毁掉的是测量装置而不是模型，这正是它值得一套工程处理而不是一段免责声明的原因。一项覆盖到 2025 年底、涵盖 55 项研究的系统综述 (Anonymous, 2026a) 提出了恰当的四级划分：T1 精确重复，T2 句法变体（重排格式、细微编辑），T3 语义变体（改写、翻译），T4 任务级泄漏（在基准的任务格式与答案约定上训练）。

4.6.1 文献究竟测出了什么

有三项发现值得记住。

第一，字符串匹配在构造上就是不够的。Yang et al. (2023a) 表明，对测试条目做改写或翻译可以完全躲开 n-gram 去污染，而且一个在改写后的测试数据上训练的 13B 模型在 MMLU、GSM8K 和 HumanEval 上达到了与当时 GPT-4 相当的表现。把一个基于 LLM 的检测器应用到被广泛使用的语料上，他们发现 HumanEval 有 8–18% 与包括 RedPajama-1T 和 StarCoderData 在内的预训练集重叠。

第二，污染的效应是真实的，但异质且难以相互比较。该综述报告在 MMLU、GSM8K、HumanEval、HellaSwag 和 PIQA 上的效应大致跨越 6% 到 40%，同时提醒这些估计来自互不可比的方法 (Anonymous, 2026a)。最干净的单项测量是Zhang et al. (2024a)，他们委托编写了 1,250 道与 GSM8K 难度匹配的新小学题：表现最差的

表 4.4: 污染检测方法与 GEM 2026 系统综述 (Anonymous, 2026a) 的四级分类之对照。「语料」级访问权限意味着训练数据是你构建的或你能读到；「模型」级访问权限意味着你只能查询权重或 API。没有任何一种方法覆盖所有层级，而算法 4.1中的组合是为覆盖面而选的，不是为某一种方法的精度而选的。

方法	层级	访问	失效方式
归一化精确哈希	T1	语料	实践中几乎什么也抓不到
n-gram 重叠（8–13）	T1–T2	语料	被改写击破 (Yang et al., 2023a)
MinHash / LSH (Lee et al., 2022)	T1–T2	语料	阈值调参；对分片大小敏感
嵌入最近邻	T2–T3	语料	假阳性会删掉合法数据
LLM 裁定 (Yang et al., 2023a)	T2–T3	语料	成本高；只用在候选上
似然 / 成员推断	T1–T3	模型	规模上功效弱；需要校准
基准审计（ConStat）	T3–T4	模型	需要一个模型参照总体
金丝雀字符串	T1	语料	只有基准附带了才行
新鲜或滚动基准	全部	都不需要	成本是一个新基准

模型在这批新题上低了 13 分，若干模型家族在各个尺寸上都一致地过拟合，而一个模型逐字生成 GSM8K 条目的概率与其表现落差正相关（Spearman $r^2 = 0.32$）。更让人安心的是，前沿模型表现出的过拟合很小，而且即便是过拟合最严重的模型也仍能泛化到新题上——污染抬高了分数，但并不是分数的全部。

第三，也是对本书而言最不舒服的一点：该综述的结论是，指令微调阶段的污染，用他们评估过的每一种方法都检测不出来。成熟的检测机制全都假定污染是在预训练期间进入的。后训练污染——一条基准条目进入了 SFT 配比或 RL 提示池——留下的统计特征更弱，截至 2026 年年中，它是可预防的而非可检测的。正是这种不对称构成了「把去污染当作一个构建步骤」的论据。

4.6.2 合成数据从其生成器继承污染

这是那种会把细致的团队也绊倒的情形。假设你的流水线是干净的——你从未碰过任何基准文件——而你去请一个强教师给出「一道带完整解答的高难度奥赛几何题」。如果这位教师从某个基准里记住过这类题目，那么这个请求实际上就是一次记忆查询，你拿到的可能是一道测试题的改写版，连答案一起。Yang et al. (2023a) 证实了这一点，他们在由 GPT-3.5 和 GPT-4 生成的合成数据集中发现了他们称之为无意污染的现象。

由此有两个后果。去污染必须跑在流水线的输出上，而不只是它的种子上——检查 175 条种子任务，对四百万条生成出来的指令什么也证明不了。而且由于继承来的污染是以改写而非拷贝的形式到来的，它恰好落在 *n*-gram 匹配失效的位置，因此合成流水线需要那套昂贵的 T3 机制，而手工整理的流水线常常可以省掉。

工程实践：一条能在评审中站得住脚的去污染流水线

算法 4.1是推荐的默认方案；关于如何运行它有四点。其一，2% 规则——丢弃任何与任一评测重叠超过 2% 的来源——来自 Tülu 3 (Lambert et al., 2025)，是目前最具体的公开标准；这个数字本身并不重要，重要的是事先登记它，并且在看到分数之后不再去调它。其二，请在一个你相信是干净的语料上校准嵌入阈值并报告假阳性率，因为一个未经校准的阈值会悄悄删掉合法数据——而这是一种你永远也不会正确归因的质量退化。其三，也要对预训练和中训练语料做去污染，而不只是对评测：OLMo 3 就对其 RL-Zero 数据这么做，以便把 RL 的效应与泄漏分离开 (Team OLMo, 2025)。其四，把它放进持续集成，并允许它让构建失败。在截止日期前用手工做一次的去污染，与不做去污染没有区别。

设计层面的缓解措施与检测互补，而且更稳健：在模型知识截止之后收集题目的滚动基准 (Jain et al., 2025a)、私有的留出划分与随基准一同发布的金丝雀 (Phan et al., 2025)，以及新编写的平行测试集 (Zhang et al., 2024a)。如果你在 2026 年构建一套内部评测，请把预算按「将被替换」而不是「将被维护」来编制。

4.7 中训练与继续预训练

中训练（mid-training）是指在一份为目标能力刻意做过富集的语料上做继续预训练，词元预算比预训练低一到两个数量级，位置在任何指令微调之前。节 1.4已作过勾勒；它从 2024 年关于闭源模型的一则传闻，变成了 2025–2026 年数据工作中得到最稳健支持的论断。

4.7.1 它与继续预训练不是一回事

Liu et al. (2025c) 把这个区别分离了出来，它取决于一个参数：继续预训练会完全切换到目标分布，对通用预训练数据的配比权重为零，而中训练保留一个混合。在代码与数学上、在不同模型尺寸上，他们发现中训练在两个方面都胜过继续预训练：域内性能，以及微调之后对通用语料损失的保持；收益在数学和代码上最大，而这两个领域正是在网页预训练中最欠表达的；此外时机与配比权重相互作用，因此你能负担多少专门数据取决于你在何时引入它。他们的一般原则——领域适配受益于渐进而非突兀的词元级分布迁移——是一个关于优化轨迹的陈述，而不是关于数据内容的陈述。

4.7.2 语料与配比

反复出现的语料有：（i）高质量的数学网页文本，（ii）代码，（iii）科学与技术类散文，（iv）问答格式的数据，以及（v）长思维链轨迹。Wang et al. (2025g) 报告了最有启发的一批消融：一份高质量的数学语料（MegaMath-Web-Pro）同时改善基座与 RL 之后的表现，而一个看似合理的替代品（FineMath-4plus）则不能；加入问答风格的数据，尤其是长思维链样例，能进一步改善 RL 结果，但需要指令数据才能解锁这一效应；长思维链数据提升推理深度，但会诱发冗长和 RL 不稳定，从而使语料的格式成为一个一阶变量；而扩大预算会一致地改善下游 RL。他们的「先稳定后衰减」配方——以恒定学习率训练 200B 词元，再在若干以思维链为重点的分支上带衰减地训练 20B——弥合了众所周知的 Llama 家族与 Qwen 家族基座模型对 RL 响应不同的差距，这是目前最强的证据，说明中训练改变的是 RL 所能达到的高度，而不仅仅是它的起点。

Runwal et al. (2026) 提供了规模最大的受控研究：横跨四个家族、七个基座模型的 500 多次实验，涵盖 3B–24B 的稠密 transformer 与注意力—Mamba 混合架构。用大约 27B 的中训练词元——相对一次 15T 词元的预训练微不足道——他们报告数学上 +15 到 +40 分、其他领域 +5 到 +12 分，整体推理能力提升三到四倍，并且预训练知识得到保持。有两项次要发现比头条结论更有用。配比构成在中训练期间的重要性远高于在 RL 期间：从「数学加代码」变为「数学加代码加科学」带来三到六分的提升，而在 RL 期间做同样的调整几乎毫无收益。而在行为层面，预训练模型在 MATH500 上给出简短答案，中训练后的模型则展示逐步推演过程，其中 Granite-3.3-8B 在中训练加 RL 之后从 16.9% 提升到 79.5%。

4.7.3 长上下文扩展，以及一个悬而未决的顺序问题

长上下文扩展——在长文档上训练并调整位置编码，以达到 64K 词元及以上——与中训练相邻，并与之相互作用。Runwal et al. (2026) 报告中训练在长上下文扩展之后最有效，并指出与之互补的隐患：一个被短上下文数据主导的中训练配比可能损害长上下文能力，常常需要一次轻量的恢复性微调，因此应当去探测而不是想当然。OLMo 3 则以相反的顺序运行这些阶段——大约 100B 词元以能力为重点的中训练，然后扩展到约 65K，再做后训练 (Team OLMo, 2025)。两件产物都很强，因此截至 2026 年年中，这个顺序问题确实是开放的；请把已发表的顺序当作在约束条件下（往往是：基座模型交付时处于什么状态）作出的选择，而不是当作结论。

4.7.4 仍有争议的论断

人们反复听到的论断是：*RL 无法引出中训练没有装进去的能力*。这个论断的强版本和弱版本，其证据支持度差别很大，值得分开来看。

强版本——RL 只能引出，从不获得——有三项支撑和一个严重问题。支撑是：节 1.8中推导的 KL 正则化最优解的支撑集保持性质，它说明 RL 无法创造出参考策略赋予零概率的行为；Yue et al. (2025a) 的 pass@k 证据，它发现 RLVR 是在锐化而不是扩展解集；以及Runwal et al. (2026) 的观察，即那些试图通过 RL 而非中训练去获取同样数学与科学知识的模型，只有很小的提升。问题在于：由离策略引导的 RL (Yan et al., 2025a) 能在纯在策略 RLVR 完全失败的地方成功训练弱模型，因此观察到的天花板有一部分是算法探索能力的性质，而非基座模型支撑集的性质。第14章会裁断这个问题，而它同样没有得出一个干净的结论。

弱版本——RL 算力的边际回报随中训练质量强烈递增——由若干项独立的受控研究所支持 (Wang et al., 2025g; Runwal et al., 2026; Zhang et al., 2025a)，是应据以规划的版本。Zhang et al. (2025a) 把中训练框定为一座中间的分布桥梁，它扩展基元覆盖并使内部表示与 RL 将要强化的东西对齐，并提出这也许就是 RL 有时泛化得惊人、有时却失败的原因。这个框架是一个假说，而不是一次测量。

工程实践: 选择中训练预算

公开证据支持把下列各点作为默认设定，但需说明：最后一条中的分配启发式是一种综合，而非受控结果。

- 预算 10–200B 词元，也就是预训练的 0.1–2%。Runwal et al. (2026) 在 27B 上获得了大幅收益，OLMo 3 用了约 100B (Team OLMo, 2025)，Wang et al. (2025g) 用了 220B；没有证据表明存在一个尖锐的最优点，而且在上端回报仍为正。
- 在数学和代码之外还要纳入科学：三领域配比比两领域配比高出三到六分，而在 RL 期间做同样的改动则没有效果 (Runwal et al., 2026)。
- 对通用预训练数据保留可观的权重 (Liu et al., 2025c)；权重为零就是继续预训练，代价是保持性。
- 纳入长思维链和问答数据，但要审计回复长度分布——在这里习得的冗长会破坏后续 RL 训练的稳定性 (Wang et al., 2025g)。
- 在前后分别独立评测长上下文能力，并为一次简短的恢复性微调留出预算 (Runwal et al., 2026)。
- 若追求在标准任务上的可靠性，宁可用更大的中训练预算加轻量 RL；若追求在难题上取得突破，则用适度预算加重度 RL。

4.8 智能体训练的数据

智能体后训练消耗三种产物，按成本与稀缺程度递增排列。

工具调用记录是单轮的函数调用轨迹：一个请求、一次结构化调用、一个结果、一个答案。它们可以从工具的 schema 廉价地合成出来，教会语法和参数格式，而这正是 SFT 在 RL 能够开始之前必须完成的大部分工作 (NVIDIA, 2025)。

轨迹是包含环境观测的多轮交互记录，它在性质上不同，因为其中含有 节 1.2指出的、教师强制语料所缺乏的那种恢复行为。它们同样需要 第17章中的损失掩码纪律：观测词元来自环境而非策略，在它们上面训练等于教会模型去幻觉工具输出。

带任务与成功谓词的环境是真正的约束所在，原因是结构性的而非数量上的。环境是软件：它的生产单位是一次集成而不是一行数据，而且每次采样都必须隔离，因为同一组内并发的成员会互相破坏状态。因此一个样本的边际成本包含一个容器，采样的墙钟时间是数秒到数分钟的真实执行，而不是毫秒级的加速器时间。这把节 4.4.5的论证放大了：在这里一个退化组按比例而言昂贵得多，所以通过率整理更重要而不是更不重要，而这恰恰是它最难做到的地方。

自动化合成是当前活跃的前沿。SETA 展示了现在可达的规模：横跨十四个类别的 4,567 个已验证终端环境——其中 3,255 个从异构来源合成，另有 1,312 个围绕模型能力前沿由已有环境演化而来 (Shen et al., 2026b)；可验证任务生成器的库已经存在 (Li et al., 2025c)，而从零构建的流水线得出的结论是环境多样性比任务数量更重要 (Tu et al., 2026)。对软件工程而言，SWE-Gym (Pan et al., 2024) 这类可执行集合扮演的角色，相当于指令数据集之于对话。第19章负责构建、校验与奖励设计；此处的要点仅仅是：节 4.1的分类法最终落到了一行以 Dockerfile 为单位的数据上。

4.9 来源、许可与可复现性

后训练结果异乎寻常地难以复现，而原因几乎完全在于数据纪律，而不在算力。

配比就是方法。 一个配方由「用了哪些数据集、按什么比例、以什么顺序」所定义——而论文报告架构和学习率的一致性，远高于它们报告配比权重的一致性。要复现像节 4.4.4中那样的迭代流程，需要每一轮迭代的权重，这比论文里的一张表通常所承载的信息要多。

模板是承重的，却被含糊带过。 确切的对话模板、系统提示和推理分隔符会可测量地影响结果，却经常被略去（节 3.4）。OLMo 3 的作者给出了针对 RL 的更尖锐的版本：修好提示模板带来的改进，可能超过被研究的算法本身带来的改进 (Team OLMo, 2025)。

过滤器被调过，却未被报告。 节 4.3.1和算法 4.1中的每一个阈值，都是某个人盯着某个分数分布定下来的。没有这些数字，「我们按质量做了过滤」就不是一个方法。

许可是实实在在的约束，不是形式。 Longpre et al. (2024) 审计了一大批被广泛使用的指令数据集的许可与署名元数据，发现了系统性的许可标注错误和署名缺失，其中许多流行数据集所附条款比聚合方所报告的更严格。合成数据还会额外继承生成器的服务条款，这正是若干开源配方宁可在质量上付出一些代价，也要使用宽松许可教师的原因。

工程实践: 最小清单

一份Gebru et al. (2021) 意义上的数据集卡片，应当签入训练配置并按内容寻址，其中记录：每一个上游来源及其许可；每一个合成切片背后的生成器模型及版本、采样温度和确切的提示模板；每一个过滤器及其阈值和它移除掉的比例；来自算法 4.1的去重与去污染日志，包括按来源的重叠表；配比权重；以及一个训练脚本在启动时会校验的内容哈希。启动时的校验正是人们会跳过、却能抓到最多错误的那一部分——它是唯一能发现「有人用不同版本的教师重新生成了某个切片」的机制。Tülu 3 (Lambert et al., 2025) 和 OLMo 3 (Team OLMo, 2025) 发布了带提交哈希和实验链接的逐模型脚本，是把这件事做彻底的参考范例。

本章小结

- 后训练消耗四种产物——示范、偏好、可验证的提示—答案对，以及环境——它们最重要的差别在于可复用性。一条示范永远只产出一个梯度方向；一个可验证提示则在策略每次变化时都产出一个新鲜的在策略梯度。正是这种不对称，而非单位成本，解释了整个领域把预算转向提示与环境的原因。
- 在有用性比较上观测到的标注员间一致率为 63–77% (Stiennon et al., 2020; Ouyang et al., 2022; Bai et al., 2022a)。在可交换标注员模型下，这把任何偏好分类器的贝叶斯准确率夹在 a 与 $\frac{1}{2}(1+\sqrt{2a-1})$ 之间（定理 4.2）——在 $a = 0.73$ 时大致是 73–84%。远高于这一区间的奖励模型准确率，说明该基准的样本对异常界限分明，而不是说明模型更好。平均更多标注员能降低标签方差，却降不掉共享的长度、格式、自信和谄媚偏差：噪声不是零均值的。
- 每一条合成数据流水线都是播种、扩增、多样化、过滤、验证，而每个阶段都有其特征性失效：种子分布设界、近重复、表层多样性、评判模型的自我偏好，以及验证器漏洞。只要存在验证器，就用它，而不是用评判模型。
- 当合成数据替代真实数据时，模型坍塌是真实的 (Shumailov et al., 2023; Dohmatob et al., 2024)；而当数据是累积起来的时候它不会发生 (Gerstgrasser et al., 2024)。这一结论出名的版本，比适用的版本更强。
- 数据高效性的主张（LIMA、s1K、LIMO）在格式与风格上被很好地复现，在能力上则复现不了；在精确匹配的推理指标上，大规模且经过精心构造的配比压过小规模精选集 (Guha et al., 2025)。表观分歧中有一部分是因变量从成对偏好换成了精确匹配。

- 对 RL 而言，按通过率做提示选择是可用的杠杆最大的整理手段，因为奖励全部相等的一组贡献恰好为零的梯度，而这种情况发生的概率是 $p^G + (1-p)^G$。期望的组相对平方优势恰为 $\frac{G-1}{G}p(1-p)$，这使得「可学习性」成为一个推导出来的量，而不是一条启发式。
- 难度标签是相对于当前策略的，而且在训练顺利时衰减得最快，所以请从你已经生成的那些组里估计通过率；一组 16 只支持三个桶，再细就不行了。一个标准答案错误的提示按通过率与一道难题无法区分，而且远比无用更糟；请用自洽性把它们分开，因为一道难题给出弥散的答案分布，而一道标错答案的题给出与标准答案不符的集中分布。
- 基于字符串匹配的去污染是不够的：改写可以躲开它，而且合成数据会从其生成器继承污染 (Yang et al., 2023a)。截至 2026 年年中，指令微调阶段的污染是可预防但不可检测的 (Anonymous, 2026a)，这正是为什么去污染必须是一个持续集成的构建步骤，并带有一份记录在案的、按来源统计的重叠报告。
- 用预训练词元的 0.1–2% 做中训练，能带来大幅且可复现的推理提升，并抬高后续 RL 所能达到的水平 (Wang et al., 2025g; Runwal et al., 2026)。弱论断——RL 算力的回报随中训练质量提升——得到了很好的支持；而强论断，即 RL 只能引出中训练装进去的东西，仍有争议。

延伸阅读

先读Lambert et al. (2025)，把它当作一份工程文档而不是一篇结果论文来读：它是关于一个配比究竟如何被构造出来的最完整的公开记述，其四条选择准则（多样性、瞄准困难技能、来源干净、严格去污染）以及 2% 重叠规则都可以直接迁移。接着读Team OLMo (2025)，它从同一个基座模型出发的 Instruct、Think 和 RL-Zero 三条独立路径，展示了一个配方中有多大比例是阶段特有的；而它把 RL 数据同时对预训练和中训练语料做去污染，是当前的最佳实践。Guha et al. (2025) 是第三件必读产物，也是唯一一件其结论建立在对流水线每一步做系统消融、而非建立在一次成功训练之上的产物。

在合成方面，请按此顺序读Wang et al. (2023b)、Xu et al. (2024a)、Xu et al. (2025)和Ge et al. (2024)，因为每一篇都恰好移除了其前作的一项依赖，而这个序列是关于一族技术如何成熟的最清晰的示例。Li et al. (2024c) 应当与它们并列，因为它是其中唯一一个回复由人类撰写的成员，这使它成为坍塌文献的天然对照。至于坍塌本身，请

把Shumailov et al. (2023) 和 Gerstgrasser et al. (2024) 按此顺序一起读；只读第一篇会得到一个校准很差的风险认识。

在选择方面，Xia et al. (2024) 是这个领域里智识上最令人满意的论文——影响函数被做成了能与 Adam 和变长数据配合工作，还带一个可迁移的梯度数据存储——而它关于基于频率的基线的负面结果，与其正面结果一样有用。Liu et al. (2024f) 是应该去实现的那一个。至于 RL 一侧，节 4.4.5中的推导本质上就是Yu et al. (2025c) 的动机部分，而 Foster et al. (2025) 把 $p(1-p)$ 照字面当成采样分布来用；请把它们读作同一个观察的两个视角。在污染方面，Yang et al. (2023a) 很短，而且会改变你构建流水线的方式，Zhang et al. (2024a) 是最干净的测量，而 GEM 2026 的系统综述 (Anonymous, 2026a) 是挑选检测器时该查阅的东西。至于中训练，Wang et al. (2025g) 解释了为什么两个基准质量相近的基座模型会对同一套 RL 配方作出完全不同的响应——这是那支文献中实践价值最高的洞见。

习题

4.1（**上限的两端。**）（a）用定理 4.2，把贝叶斯准确率的上下界画成 $a \in [0.5, 1]$ 的函数。（b）在一个 10% 的子样本上每个条目有三名标注员，给出 $\mathbb{E}[p(1-p)]$ 以及 $\mathbb{V}\mathrm{ar}[p(1-p)]$ 的一个估计量，并解释后者如何收窄 (4.3)中的区间。（c）对 $m = 1, 3, 5, 9$ 名标注员做多数投票，推导聚合结果与 $\mathbb{1}[p > \frac{1}{2}]$ 一致的概率，找出 $m = 1 \to 9$ 使该概率变化不足一个百分点的 p 区间，并说明这对标注预算该往哪里投意味着什么。

4.2（**被浪费的采样。**）一个提示池满足 $p \sim \mathrm{Beta}(a, b)$。用 Beta 函数以闭式推导期望的退化组比例 $\mathbb{E}[p^G + (1-p)^G]$。对 $G \in \{8, 16, 32\}$ 和 $(a, b) \in \{(0.5, 0.5), (1, 1), (0.5, 4)\}$ 求值，并解释第三种情形（一个大体上太难的提示池）。把完美过滤带来的加速与把 G 翻倍带来的加速作比较。

4.3（**归一化估计量下的可学习性。**）对标准化优势 $(r_i - \hat{\mu})/\hat{\sigma}$ 重做节 4.5中的推导，条件是该组非退化。证明期望平方优势变得近似与 p 无关，因此标准化估计量隐式地对提示重新加权。哪些提示的权重变大了？把你的答案与Liu et al. (2025g) 联系起来。

4.4（**IFD 会选出噪声。**）构造三个例子，它们会被一个纯 IFD 过滤器 (Li et al., 2024b) 排为信息量最大，却会损害模型：一个回复被破坏的，一个回复正确但与指令无关

的，以及一个回复正确但单纯高熵的。对每一个，指出哪一个额外信号能把它排除掉，并解释为什么单一的标量分数做不到。

4.5（去污染的假阳性。）在算法 4.1中设 $\tau_{\text{ann}} = 0.9$，结果移除了你 4% 的数学数据。只用你已有的数据，设计一个实验来区分「该语料本来就有 4% 被污染」和「该阈值删掉了 4% 的合法难题」。说明你测量什么、什么样的结果支持哪个假设，以及代价是多少。

4.6（中训练与长上下文扩展的顺序。） 节 4.7指出Runwal et al. (2026) 和 Team OLMo (2025) 对这两个阶段的排序不同。把两种顺序写成带词元预算的显式阶段序列，并对每一种预测短上下文数学准确率、长上下文检索、回复长度和 RL 稳定性各自会退化还是改善。然后设计能够区分它们的最小受控实验，并估计它在一个 8B 模型上以 GPU 小时计的成本。

4.7 ⋆（对真实数据来说这个一致率界紧吗?）定理 4.1假定标注员在给定条目时条件独立。改设每名标注员 j 有一个偏差向量 β_j（长度、格式、断言性），使得对条目特征 $\phi(z)$ 有 $p_j(z) = \sigma(r^\star(z) + \langle \beta_j, \phi(z) \rangle)$。证明观测到的一致率现在是 $r^\star$ 与 $\mathbb{V}\mathrm{ar}_j[\beta_j]$ 两者的函数，因此定理 4.2不再能界定 with respect to $r^\star$ 的贝叶斯准确率。提出一个识别策略：要把条目本身的真实歧义与标注员的异质性分开，你必须收集什么？讨论 Dubois et al. (2024) 的长度受控估计量是否是一个特例，以及把它推广到 节 4.2.4中的其他偏差需要什么。

4.8 †（构建通过率账本。）取一个 1–8B 的指令微调模型、5,000 道可程序化检查的数学题，以及任意一个 GRPO 系列的训练器，实现代码 4.1。在匹配的采样预算下运行三个 300 步的配置：均匀采样；由第 0 步用 $n = 32$ 测得一次的离线通过率所构建的静态由易到难分桶；以及在线可学习性采样。逐步记录平均奖励、退化组比例、$\hat{p}$ 的均值与标准差，以及留出的 pass@1 与 pass@16。报告达到固定奖励水平所需的墙钟时间，画出每种方案下通过率分布如何移动，并解释静态分桶获胜的任何情形。

4.9 †（测量你自己的污染。）取一个至少 100K 样例的开源指令数据集和一套评测，实现算法 4.1的阶段 1–3。按来源数据集报告每个阶段检测到的重叠，以及嵌入阶段相对 n-gram 阶段、裁定相对嵌入的边际收益。在一个来自强教师的合成数据集上重复一遍，并比较逐阶段的分布特征。事先预测在合成情形下哪个阶段贡献最大，并依据 节 4.6给出理由。

算法 4.1 对后训练配比做分阶段去污染。各阶段按每对候选的成本递增排序，使得昂贵的裁定器只看到被廉价过滤器标记出来的内容。主要产物是那份报告，而不是过滤后的语料：正是它让一个评测数字能够跨项目比较。

输入： 训练配比 $\mathcal{D} = \{d_1, \ldots, d_N\}$；评测套件 $\{B_1, \ldots, B_M\}$；阈值 τ_{ann}、τ_{drop}

阶段 0：金丝雀与精确匹配
在 $\mathcal{D}$ 中扫描已发布的金丝雀字符串（BIG-bench、HLE (Phan et al., 2025)）；移除并**记录日志**
移除那些经空白与大小写归一化后哈希值与任一基准条目相同的 d_i
阶段 1：廉价的句法层召回
for 每个套件 B_m **do**
    构建 B_m 的 k-gram 指纹集合（散文取 $k = 8$，代码取 $k = 13$） ▷ 廉价，对 T1–T2 召回高
    标记与 B_m 共享任一指纹的 d_i
    另外标记对 B_m 中任一条目 MinHash Jaccard > 0.7 的 d_i (Lee et al., 2022)
end for
阶段 2：语义候选
对 $\mathcal{D}$ 与 $\bigcup_m B_m$ 做嵌入；为每个 d_i 检索最近的前 10 个基准条目
标记余弦相似度超过 τ_{ann} 的 d_i，该阈值应经过校准，使得在一个已知与 $\bigcup_m B_m$ 不相交的语料上的假阳性率低于 1%
阶段 3：裁定
for 每一对被标记的 (d_i, b) **do**
    请一个与数据生成器不同家族的评判模型判断 d_i 是否为 b 的改写或翻译 (Yang et al., 2023a)
    if 判为等价 **then**
        移除 d_i；记录 $(i, m, \text{阶段}, \text{分数})$
    end if
end for
阶段 4：报告与闸门
for 每个来源数据集 $S \subseteq \mathcal{D}$ **do**
    $o_S \leftarrow |\text{从} S \text{ 中移除的部分}| \,/\, |S|$
    if $o_S > \tau_{\text{drop}}$ **then**
        整体丢弃 S ▷ Tülu 3 取 $\tau_{\text{drop}} = 2\%$ (Lambert et al., 2025)
    end if
end for
return 过滤后的配比、按来源的重叠表、结果的内容哈希

第II部分

基于反馈的对齐

第 5 章

偏好建模与奖励模型

核心思想：奖励模型是对一个谁也测不了的量所做的统计估计

对齐（alignment）需要一个标量 $r^\star(x, y)$，而没有哪个标注员报得出它，也没有哪台仪器读得出它。标注员能报告的，是两个回复中他们更偏好哪一个。偏好建模就是把一堆序数比较变成一个可微标量的那套机器，而下游出的所有问题——长度膨胀、谄媚、过度优化、古德哈特定律——都可以追溯到「比较数据」与「我们假装已经复原出来的那个标量」之间某个具体的缺口。本章推导这套机器，陈述它的可识别性极限，量化它的准确率上限，并罗列它的失效模式。

5.1 奖励模型是用来做什么的

回忆模板(1.1)：我们在一个 KL 锚定的约束下最大化 $\mathbb{E}_{y\sim\pi_\theta}[r(x,y)]$。要把它实例化到开放式文本上，我们需要一个函数 $r : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$，它在任意模型输出上求值都很便宜，不需要对任何东西可微——一个返回数字的黑箱就够了——并且与我们真正想要的东西相关。把我们想要的记为 $r^\star$，把我们拟合出来的记为 r_ϕ，即一个参数为 ϕ 的参数化模型。$r^\star$ 有三个性质使这件事很难，值得在写下损失函数之前先讲清楚。

$r^\star$ 观测不到，即便是带噪的绝对尺度观测也没有。 让一名标注员「按 1 到 7 给这个回复的有用性打分」，你会得到一个数字，但从内心判断到整数的映射因人而异，在一次标注过程中会漂移，而且在量表两端会非均匀地压缩。两名在每一次成对比较上都完全一致的标注员，可以给出均值、方差和形状都不同的李克特分布；对这类评分取平均，估计出来的是一个尺度惯例的混合，而不是质量。

比较很便宜，而且不需要校准。 问「这两个哪个更好?」把尺度完全去掉了：答案只取决于一个内部差值的符号，因此系统性的偏移和增益都被抵消。这就是支持偏好数据的实践论据，而且它由来已久——比较判断法被引入心理物理学，正是因为绝对量级的估计不可靠 (Thurstone, 1927)。代价在于信息量：一次二元比较最多携带一比特，而一个校准良好的评分能携带好几比特。

序数数据只有在一个模型之下才能确定一个基数函数。 一组比较告诉你的是一个偏序；而策略梯度方法需要的是数字。要在两者之间架桥就需要一个联结函数——一个关于偏好概率如何依赖于奖励差值的假设——而这个选择并非无关紧要。它决定了哪些偏好结构根本是可表示的，从而也决定了你的奖励模型会悄悄丢掉什么。

工程实践: 评分与比较回答的是不同的问题

当关心的量是一个排序时（该服务哪个回复、该强化哪个回复），用比较；当你需要一个绝对阈值（「这个能安全上线吗?」）或者一个可解释的分属性分数时，用评分。若两者兼有，就让比较决定训练信号的符号、让评分通过一个间隔来决定其大小（节 5.3）；Wang et al. (2025h) 是关于这一组合最干净的公开消融，报告称组合目标胜过单用其中任何一个。

5.2 Bradley–Terry 模型及其亲属

5.2.1 随机效用推导

假设每个回复 y 对标注员而言有一个潜在效用，等于一个确定性部分加上噪声：

$$u(y) = r^\star(x, y) + \varepsilon_y, \qquad \varepsilon_y \ \textit{i.i.d.}\ \text{some distribution}, \tag{5.1}$$

并且标注员恰好在 $u(y^+) > u(y^-)$ 时报告 $y^+ \succ y^-$。于是

$$\mathbb{P}[y^+ \succ y^-] = \mathbb{P}[\varepsilon_{y^-} - \varepsilon_{y^+} < r^\star(x, y^+) - r^\star(x, y^-)] = F(\Delta r^\star), \tag{5.2}$$

其中 F 是噪声差的累积分布函数，而 $\Delta r^\star = r^\star(x, y^+) - r^\star(x, y^-)$。文献中主要有两种噪声选择。

- **Gumbel** 噪声给出 logistic 形式的差值分布，于是 $F = \sigma$，且

$$\mathbb{P}[y^+ \succ y^- \mid x] = \sigma(r^\star(x, y^+) - r^\star(x, y^-)), \qquad \sigma(z) = \frac{1}{1 + e^{-z}}. \tag{5.3}$$

这就是 Bradley–Terry 模型 (Bradley and Terry, 1952)，等价于 Luce 的选择公理 (Luce, 1959)。它之所以成为 RLHF 中的标准，理由平淡却决定性：$\log\sigma$ 有数值稳定的实现，以及有界、条件良好的梯度。

- **高斯**噪声给出 $F = \Phi(\cdot/\sqrt{2}\sigma_\varepsilon)$，即 Thurstone–Mosteller 模型或 probit 模型 (Thurstone, 1927)。它的拟合几乎无法与前者区分，因为 logistic 函数与经适当缩放的正态累积分布函数在整个取值范围上都很接近。它们的差别在尾部，在那里 probit 更不宽容：它给大爆冷分配的概率要小得多，因此一个预测间隔很大却被标错的样本对会产生大得多的梯度。这种在标注噪声下的脆弱性，正是 probit 奖励模型很少见的原因。

历史注记：为什么会是一篇 1952 年关于不完全区组设计的论文

Bradley and Terry (1952) 当时解决的是一个农业统计问题：当每个试验区组只能容纳少数几个处理、因而永远观测不到一个完整排序时，如何对这些处理排序。这个结构——许多条目、稀疏的成对比较、每个条目一个潜在标量——恰恰就是语言模型偏好数据的结构，区别在于我们把每个条目一个自由参数替换成了一个神经网络，从而使得那个标量可以对从未见过的条目做出预测。这一替换就是现代的全部贡献，同时也是本章中每一次泛化失败的根源。

5.2.2 极大似然目标

给定一个比较数据集 $\mathcal{D} = \{(x_i, y_i^+, y_i^-)\}_{i=1}^N$，(5.3)给出负对数似然

$$\mathcal{L}_{\mathrm{BT}}(\phi) = -\mathop{\mathbb{E}}_{(x,y^+,y^-)\sim\mathcal{D}}\big[\log\sigma(r_\phi(x,y^+) - r_\phi(x,y^-))\big]. \tag{5.4}$$

这就是在两个网络输出之差上做逻辑回归。它的梯度很有启发性：

$$\nabla_\phi \mathcal{L}_{\mathrm{BT}} = -\,\mathbb{E}\Big[(1-\sigma(\Delta r_\phi))(\nabla_\phi r_\phi(x,y^+) - \nabla_\phi r_\phi(x,y^-))\Big]. \tag{5.5}$$

权重 $1-\sigma(\Delta r_\phi)$ 就是模型把这一对排错的概率。模型已经自信地排对的样本对几乎不贡献任何东西；被它自信地排错的样本对，每对最多贡献量级为 1 的梯度。因此 Bradley–Terry 以一种「评分上的平方损失」所不具备的方式自我限幅：一个被标错的样本对不可能主导整个批次。这种有界的影响力，是 logistic 联结函数胜出的第二个原因。

5.2.3 可识别性：只有同一提示内部的差值被确定

命题 5.1 (逐提示的平移不变性). 设 $c : \mathcal{X} \to \mathbb{R}$ 任意，定义 $r'_\phi(x,y) = r_\phi(x,y) + c(x)$。那么对任何「每对中的两个回复共享同一提示」的比较数据集，都有 $\mathcal{L}_{\mathrm{BT}}(r'_\phi) = \mathcal{L}_{\mathrm{BT}}(r_\phi)$。对 *Plackett–Luce* 似然和 *probit* 似然同样成立。

证明. (5.4)的每一项都只通过 $r_\phi(x,y^+) - r_\phi(x,y^-)$ 依赖于 r_ϕ，而 $(r_\phi(x,y^+) + c(x)) - (r_\phi(x,y^-) + c(x)) = r_\phi(x,y^+) - r_\phi(x,y^-)$。对 Plackett–Luce 而言，每个位置上的 softmax 是对同一个固定提示的若干回复取的，而 softmax 对其所有 logit 同时加上一个常数是不变的。对 probit 而言，Φ 的自变量同样是一个提示内部的差值。 □

这个命题只有两行长，其后果却与其难度完全不成比例。

1. **奖励的量级在不同提示之间不可比**。如果你的奖励模型给某个提示的一个回复打 2.4 分、给另一个提示的一个回复打 -0.8 分，那么数据从未确定过哪一个更好。任何表观上的跨提示可比性都来自网络的归纳偏置——权重衰减、共享特征、初始化的偶然性——而不是来自似然。
2. **逐提示中心化是免费的，凡是合理的做法都会这么做**。减去组均值，正如每一个组相对估计量所做的那样（第12章），恰好去掉了那个未被识别的分量；一个学到的值基线 $V_\psi(x)$ 也起同样作用。因此 PPO 式的 RLHF 与 GRPO 式的 RLVR 都对 c 不变，这是检验任何新估计量的一个很好的完备性检查：如果给某个提示的所有奖励同时加上一个常数会改变你的优势，那么这个估计量用到了数据从未提供过的信息。
3. **整体尺度是被识别的，而且它与 β 相互抵换**。 定理 5.1对 $r_\phi \mapsto \lambda r_\phi$ 什么也没说，而后者确实会改变似然。既然 $\pi^\star \propto \pi_{\mathrm{ref}} e^{r/\beta}$ （节 1.8），把奖励重新定标就等价于把 $1/\beta$ 重新定标：奖励归一化和 KL 系数调参是同一个旋钮的两个方向，只报告其中之一而不报告另一个是没有信息量的。
4. **它解释了 DPO 为何可以忽略一个难以计算的配分函数**。把 $\pi^\star \propto \pi_{\mathrm{ref}} e^{r/\beta}$ 反解，得到 $r(x,y) = \beta \log \frac{\pi^\star(y\mid x)}{\pi_{\mathrm{ref}}(y\mid x)} + \beta \log Z(x)$。第二项只是 x 的函数——恰恰就是 定理 5.1中的 $c(x)$——所以它在 Bradley–Terry 似然中被抵消，永远不必被计算出来（第8章）。

常见陷阱: 用一个全局奖励阈值来过滤数据

拒绝采样 SFT——每个提示生成 n 个候选，保留好的那些，再微调——是后训练中回报最高的技术之一（第4章）。把「好的那些」实现成 r_ϕ 上的一个全局阈值，实际上是在悄悄地筛选提示而不是回复：那些未被识别的偏移 $c(x)$ 恰好较高的提示会贡献它们的全部样本，而偏移较低的提示一个也贡献不了，于是幸存下来的配比会偏向奖励模型打分慷慨的那些主题。请改为在每个提示内部取前 k 个或前若干分位。这个 bug 很常见，在任何聚合指标里都看不见，而且只会表现为 SFT 配比的主题分布出现无法解释的偏移。

5.2.4 排序：Plackett–Luce

当标注员对 K 个回复排序而不是在两个之间做选择时，自然的推广是把这个排序建模成一串选择，每一次都从剩下的候选池里选。对 $\{1,\dots,K\}$ 的一个排列 τ，

$$\mathbb{P}[\tau \mid x] = \prod_{k=1}^{K} \frac{\exp(r_\phi(x, y_{\tau(k)}))}{\sum_{j=k}^{K} \exp(r_\phi(x, y_{\tau(j)}))}, \tag{5.6}$$

这就是 Plackett–Luce 模型 (Plackett, 1975; Luce, 1959)。当 $K = 2$ 时它退化为(5.3)。它在实践上的吸引力在于数据效率：一个 K 路排序以大约几次比较的标注成本提供了 $\binom{K}{2}$ 条彼此一致的成对约束，而 Ouyang et al. (2022) 正是利用了这一点，他们收集对 4 到 9 个回复的排序，并把来自同一个排序的所有样本对放在同一个批次里训练，以避免同一个回复出现在许多个彼此独立的梯度步中所导致的过拟合。

5.2.5 Bradley–Terry 假设排除了什么

Bradley–Terry 并不是一个中性的记账工具。因为它把所有偏好概率都从「每个回复一个标量」推导出来，它强加了两条实质性的限制。

第一，它蕴含强随机传递性：若 $\mathbb{P}[a \succ b] \geq \frac{1}{2}$ 且 $\mathbb{P}[b \succ c] \geq \frac{1}{2}$，则 $\mathbb{P}[a \succ c] \geq \max(\mathbb{P}[a \succ b], \mathbb{P}[b \succ c])$。总体中的任何非传递性都是不可表示的。

第二，也是实践中危害更大的一条：这个模型类在对标注员做混合的运算下不封闭。

命题 5.2 (Bradley–Terry 模型的混合一般不再是 Bradley–Terry 模型)**.** 存在三个回复、三个奖励函数以及一组混合权重，使得所诱导出的成对偏好概率容许一个 *Condorcet* 循环，因而对任何 r_ϕ 都无法由(5.3)再现。

证明. 取同一个提示下的回复 a, b, c 和三个等权重的子总体，其在 (a, b, c) 上的奖励分别为 $r_1 = (2M, M, 0)$、$r_2 = (0, 2M, M)$、$r_3 = (M, 0, 2M)$。当 $M \to \infty$ 时每个子总体的比较都变为确定性的，取平均得到 $\mathbb{P}[a \succ b] = \mathbb{P}[b \succ c] = \mathbb{P}[c \succ a] = \frac{2}{3}$。在(5.3)下，$\mathbb{P}[a \succ b] > \frac{1}{2}$ 成立当且仅当 $r_\phi(a) > r_\phi(b)$；于是这三个不等式会给出 $r_\phi(a) > r_\phi(b) > r_\phi(c) > r_\phi(a)$，矛盾。由混合概率关于 M 的连续性，对所有充分大的有限 M 结论同样成立。 □

在这类数据上拟合时,极大似然返回的是最接近的 Bradley–Terry 模型,而对上面那个对称例子来说，它会给三个回复分配几乎相等的奖励。总体明明被尖锐而系统性地分裂了，奖励模型却报告无差别，而残差在训练损失中表现得就像是标注噪声。Siththaranjan et al. (2024) 把这一点在隐藏上下文——诸如标注员身份或标注员读的是哪个版本的指南这类未观测变量——的情形下精确化，并证明在隐藏上下文下的 Bradley–Terry 学习隐式地按一种类似 Borda 计票的规则来聚合，而不是按期望效用聚合。那并不是任何人本意想要的聚合方式，也正是异质标注员池会产生「平庸而非仅仅有噪声」的奖励模型的原因。

常见陷阱: 把不可约的分歧读作标签噪声

当留出集上的奖励模型准确率停在 70% 左右时，诱人的诊断是「标注员有噪声，多收集点数据」。而正确的诊断往往是：标注池中包含了顺序确实不同的若干子总体，在这种情况下更多数据只会更快地收敛到同一个平庸的最优解。区分的检验很便宜：按标注员（或任何可得的协变量）拆分标签并重新拟合。如果分组后的各个奖励模型在组内的准确率都远高于合并模型的整体准确率，那么你面对的是异质性而不是噪声——而对策是一个条件式或多头的奖励模型，而不是更多标签。

5.3 架构与训练

5.3.1 标量输出头

标准的奖励模型就是一个语言模型的 transformer 主干，把词表投影换成一个到 $\mathbb{R}$ 的线性映射：

$$r_\phi(x, y) = w^\top h_T(x, y) + b, \tag{5.7}$$

其中 h_T 是回复最后一个词元处的末层隐状态，而 $w \in \mathbb{R}^{d_{\text{model}}}$ 是一个学习得到的向量，通常初始化得很小，使模型一开始接近常值。

选最后一个词元不是随意的选择。在因果注意力下，位置 t 处的隐状态只是 $(x, y_{<t+1})$ 的函数；因此放在位置 t 的输出头评分的是一个前缀，而不是整个回复。只有最后那个位置注意到了全部内容。其他备选方案各有值得了解的后果：

- **在回复位置上做均值池化**给出 $\frac{1}{T}\sum_t w^\top h_t$，也就是若干前缀分数的平均。由于早期前缀看不到后面的内容，这会低估回复末尾的信息，并引入一种隐式的长度归一化——一个伪装成架构细节的长度偏差干预。
- **注意力池化**在各位置上学习查询权重，增加的参数在偏好数据集典型的小数据规模（10^4–10^6 对）下容易过拟合。
- **双向编码器**彻底消除了前缀问题，但放弃了用策略权重做初始化，其代价大于架构上的优雅所换来的收益。

用 SFT 检查点做初始化几乎是通行做法，因为奖励模型必须恰好在策略所产生的那个分布上泛化，而一个从策略派生出来的主干已经表征了它。代价是误差相关：一个从策略初始化的 RM 共享策略的盲区，因此策略最有可能去利用的那些错误，恰恰就是 RM 继承下来的那些错误。Eisenstein et al. (2024) 给出了最有力的证据，报告称成员之间只在微调随机种子上不同的集成，对奖励作弊的防护远弱于成员在预训练随机种子上不同的集成。

奖励模型相对于策略的规模一直在往上漂。Ouyang et al. (2022) 为一个 175B 的策略使用了 6B 的奖励模型，理由是大的 RM 不稳定，而且 RM 只需要排序、不需要生成。到 2025 年，常见配置是 RM 与策略同级或低一档，具有竞争力的开源发布集中在 8B–70B 区间 (Liu et al., 2024a, 2025b)。支持同级的论据在 节 5.8：更大的奖励模型下过度优化出现得更晚，因此 RM 的容量买来的是可用的 KL 预算。

5.3.2 一族损失函数

表 5.1汇总了当前在用的目标函数。代码 5.1是前两者的参考实现。

由Touvron et al. (2023) 推广开来的**间隔**变体，把(5.4)换成 $-\log\sigma(\Delta r_\phi - m)$，其中 m 随标注员报告的偏好强度增大。它避免模型把容量花在区分几乎打平的样本对上，而且使奖励尺度部分可解释，因为一个单位的 r_ϕ 现在对应一个单位的报告强度。风险在于 m 是一个由人选定的、从类别化强度标签到实数的映射，因此一个选得糟糕的阶梯会把系统性偏差注入奖励尺度。

表 5.1: 当前在用的奖励模型目标函数。Δr_ϕ 是 $r_\phi(x, y^+) - r_\phi(x, y^-)$ 的简写；m 是由评分差距导出的间隔；s 是人工或模型给出的标量评分；k 是属性的下标。最后一行对一整对回复联合打分，因此在不额外做 n 次比较的情况下无法用作 RL 中的逐样本奖励。

变体	目标（每个样例）	所需数据
Bradley–Terry	$-\log \sigma(\Delta r_\phi)$	样本对
带间隔的 BT	$-\log \sigma(\Delta r_\phi - m)$	样本对 + 偏好强度
Hinge / SLiC	$\max(0,\, m - \Delta r_\phi)$	样本对 + 间隔
Plackett–Luce	由(5.6)给出的 $-\sum_k \log \mathrm{softmax}_k$	K 路排序
回归	$(r_\phi(x, y) - s)^2$	绝对评分
多属性回归	$\sum_k (r_\phi^{(k)}(x, y) - s_k)^2$	分属性评分
多头偏好	$\sum_k -\log \sigma(\Delta r_\phi^{(k)})$	分目标的样本对
生成式（判决词元）	$-\log \pi_{\mathrm{J}}(\text{“A”} \mid \mathrm{tmpl}(x, y_A, y_B))$	样本对
成对交叉编码器	$-\log \sigma(g_\phi(x, y^+, y^-))$	样本对

多目标奖励模型预测一个向量，要么是若干个在不同偏好集上训练的 Bradley–Terry 头，要么是在被标注属性上的回归头。其动机在于有用性与无害性彼此冲突，而把它们塌缩到单个标注员的一次判断之中，会毁掉下游系统所需要的信息。Touvron et al. (2023) 分别训练了有用性和安全性奖励模型，并在 RL 阶段用一条条件规则把它们组合起来；Wang et al. (2024b) 学习了分属性的头加一个学习得到的门控网络，这使得分数可分解、因而可审计——如果奖励上升了，你可以追问是哪个属性上升了。Dong et al. (2023b) 和 Wang et al. (2024h) 提供了相应的数据，其中回复在若干独立维度上被标注（有用性、正确性、连贯性、复杂度、冗长度）。这既允许一个回归式奖励模型，更重要的是，它让冗长度可以被显式建模，而不是被吸收进「质量」这个概念里（节 5.5）。

成对交叉编码器 (Jiang et al., 2023) 把两个回复送进同一次前向传播，直接输出一个偏好 logit。它们在每参数准确率上更高，因为模型可以跨候选做注意力，而不必把每个候选压成一个标量；但它们作为 RL 奖励几乎没用：要得到逐样本的分数就需要与一个参照集做比较，而由此得到的尺度按构造并不传递。它们的位置在重排序与评测。

代码 5.1: 带可选标注员强度间隔的 Bradley–Terry 奖励模型损失。被选中与被拒绝的序列被批在一起，使两次前向传播共享一次核启动；奖励从每个序列最后一个非填充位置读出。

```
def rm_forward(body, head, input_ids, attn_mask):
    h = body(input_ids=input_ids, attention_mask=attn_mask).last_hidden_state
    last = attn_mask.sum(dim=1) - 1                       # 最后一个真实词元的下标
```

```
    h_last = h[torch.arange(h.size(0), device=h.device), last]
    return head(h_last.float()).squeeze(-1)              # (B,) 标量奖励

def bt_loss(body, head, chosen, chosen_mask, rejected, rejected_mask,
            margin=None, center_coef=1e-3):
    ids = torch.cat([chosen, rejected], dim=0)
    mask = torch.cat([chosen_mask, rejected_mask], dim=0)
    r = rm_forward(body, head, ids, mask)
    r_pos, r_neg = r.chunk(2, dim=0)
    delta = r_pos - r_neg
    if margin is not None:
        delta = delta - margin
    loss = -F.logsigmoid(delta).mean()
    # 固定规范自由度：BT 损失对逐提示的偏移视而不见，
    # 若无此项，|r| 会在各个检查点之间自由漂移。
    loss = loss + center_coef * r.mean().pow(2)
    return loss, {"acc": (delta > 0).float().mean(), "margin": delta.mean()}
```

代码 5.1中的中心化项值得说明一下。根据 定理 5.1，损失对逐提示的偏移视而不见，因此没有任何东西阻止 $|r_\phi|$ 漂移到很大的数值，而这会恶化数值条件，并使不同检查点之间的奖励不可比。对批均值施加一个小的惩罚，可以在不改变任何提示内部差值的前提下把规范固定下来。Liu et al. (2024a) 报告称，这一点以及另外一小把类似的细节——数据过滤、损失选择、一个适中的学习率——就把一个平庸的奖励模型和一个有竞争力的奖励模型区分开来，这很好地提醒我们：奖励建模主要是数据和管道活儿，而不是架构活儿。

5.4 准确率上限与校准

5.4.1 一个奖励模型最好能有多好？

奖励模型的准确率上界由标签的一致性决定，而且这个界比直觉所暗示的更紧。

命题 5.3 (一致率给出可达准确率的上界)**.** 假设每次比较 (x, y_1, y_2) 都有一个该对特有的概率 $q \geq \frac{1}{2}$，表示一名标注员报告众数偏好的概率，并且对同一对做标注的两名标注员在给定该对时条件独立。若期望的双标注员一致率为 $a = \mathbb{E}[q^2 + (1-q)^2]$，那么对单个标注员标签的贝叶斯最优预测器的准确率满足

$$\mathbb{E}[q] \;\leq\; \frac{1 + \sqrt{2a - 1}}{2}. \tag{5.8}$$

算法 5.1 从成对比较训练奖励模型。

输入： SFT 检查点 π_{sft}，比较数据 $\mathcal{D}$，留出划分 $\mathcal{D}_{\text{val}}$

用 π_{sft} 初始化主干；用 $w \sim \mathcal{N}(0, d_{\text{model}}^{-1})$ 初始化输出头
数据卫生（主导最终质量）
丢弃两个回复经归一化后完全相同的样本对
丢弃出现在任何评测集中的提示 ▷ 污染
把来自同一个提示的所有样本对放进同一个小批次 ▷ 避免对共享回复过拟合
若有，记录每对的标注员 id 与强度标签
优化
for $e = 1$ **to** E（通常为 1–2） **do**
  for 小批次 $B \subset \mathcal{D}$ **do**
    对 $i \in B$，$\Delta_i \leftarrow r_\phi(x_i, y_i^+) - r_\phi(x_i, y_i^-)$
    $\mathcal{L} \leftarrow -\frac{1}{|B|}\sum_i \log \sigma(\Delta_i - m_i) + \lambda \bar{r}_B^2, \quad \bar{r}_B = \frac{1}{2|B|}\sum_i (r_\phi(x_i, y_i^+) + r_\phi(x_i, y_i^-))$
    更新 ϕ ▷ 学习率 $\approx 10^{-6}$；超过 2 个 epoch 会过拟合
  end for
  在 $\mathcal{D}_{\text{val}}$ 上评测准确率，按长度分桶并按来源分层
end for
校准与规范固定
在 $\mathcal{D}_{\text{val}}$ 上拟合一个标量温度 κ 以最小化 $-\log \sigma(\kappa\Delta)$ ▷ 单调；不改变排序
记录 r_ϕ 在在策略样本上的 μ, σ，供 RL 归一化使用
return r_ϕ、κ、(μ, σ)，以及分层准确率表

证明. 对固定的 q，$a(q) = 1 - 2q(1-q)$ 在 $[\frac{1}{2}, 1]$ 上严格凸且递增，因此由 Jensen 不等式 $\mathbb{E}[a(q)] \geq a(\mathbb{E}[q])$，i.e.，$a \geq a(\mathbb{E}[q])$。对 $[\frac{1}{2}, 1]$ 上 a 的递增反函数（由解 $2q^2 - 2q + (1-a) = 0$ 取 $\geq \frac{1}{2}$ 的根得到）作用于两侧，得到 $\mathbb{E}[q] \leq (1 + \sqrt{2a-1})/2$。标签的贝叶斯最优预测器预测众数结果，其正确概率为 q，故其准确率为 $\mathbb{E}[q]$。 □

表 5.2解释了一件否则看起来像丑闻的事实：已发表的奖励模型在困难留出划分上的准确率位于 65–80% 这一区域，这已经接近上限，而不是模型拟合得差的证据。Lambert et al. (2024) 确立了按类别报告准确率的做法，这让人们看清了聚合数字掩盖了巨大的差异——在显而易见的安全样本对上接近上限，在微妙的推理样本对上接近随机。Malik et al. (2025) 用新收集的提示重建了这个基准，并报告模型的平均分比在

表 5.2: 定理 5.3所蕴含的准确率上限。第二列是对单个标注员标签的最佳可能准确率；第三列是对三人多数标签的最佳可能准确率，$p^3 + 3p^2(1-p)$，这也是昂贵数据集之所以做聚合的原因。已报告的开放式有用性比较上的标注员间一致率通常落在 60–75% 这一区间 (Stiennon et al., 2020; Bai et al., 2022a; Ouyang et al., 2022)，因而现实中的上限位于表格中间几行。

一致率 a	对 1 个标签的上限	对 3 人多数的上限
0.60	0.724	0.813
0.65	0.774	0.870
0.70	0.816	0.911
0.75	0.854	0.942
0.80	0.887	0.965
0.90	0.947	0.992

原版上低了大约 20 分，同时与下游有用性的相关性更好。有两条保留意见指向不安全的方向：标注员的错误是相关的（共享指南、共享文化先验、共享疲劳），这会在不抬高真实上限的情况下抬高 a；而且定理 5.3界定的是相对标注员分布的准确率，而该分布本身也只是 $r^\star$ 的一个代理。

5.4.2 从准确率到信噪比

RM 在样本对上 p 的准确率，会转化为 RL 信号的一个具体退化程度。如果 RM 以概率 p 独立地把每一对排对，那么 RM 诱导的优势差之符号与真值一致的期望比率是 $2p-1$，因此检测一个固定的真实差异所需的样本数按 $(2p-1)^{-2}$ 增长：相对完美奖励，$p = 0.8$ 时是 2.8 倍，$p = 0.7$ 时是 6.3 倍，$p = 0.65$ 时是 11 倍。独立性假设是慷慨的，因为真实的 RM 误差是系统性的、根本不会被平均掉，但这个计算给出的量级预期是对的。在 65–75% 区间内五个百分点的准确率提升，大致相当于 RL 样本效率的两倍，这也是为什么在流水线的这个阶段，奖励模型上的工作比大多数算法工作回报更高。

5.4.3 校准

一个奖励模型是校准的，如果在所有满足 $\sigma(\Delta r_\phi) = s$ 的留出样本对中，观测到的偏好频率就是 s。测量这一点不花什么成本：把留出样本对按预测概率分箱，画出观测值对预测值的图。拟合出来的 Bradley–Terry 模型在其训练分布上通常过度自信，离开

该分布后更甚，而标准的补救是在留出数据上拟合一个单一温度 κ，把 $\sigma(\Delta r_\phi)$ 替换为 $\sigma(\kappa \Delta r_\phi)$。

核心思想：重校准对重排序是免费的，对 RL 却是承重的

对奖励做单调重校准不改变任何排序，因此 best-of-n 选择、拒绝采样过滤和成对评测都完全不受影响。但 KL 正则化的最优解是 $\pi^\star \propto \pi_{\text{ref}} e^{r_\phi/\beta}$，它依赖于奖励的尺度，而不只是它的顺序。一个过度自信两倍的奖励模型，在 β 固定的情况下，等于在以预期一半的 KL 惩罚跑 RL。这是「一个在重排序基准上看起来极好的 RM 却可能在 RL 中表现糟糕」的一个具体原因，也是应当逐检查点记录并监控 RM 尺度统计量的原因（算法 5.1）。

5.4.4 集成与悲观主义

如果奖励模型的误差会引起过度优化，那么把若干奖励模型平均起来应当有帮助，而它确实有帮助——只是不如人们希望的那么多。Coste et al. (2024) 报告集成能减少过度优化，其中保守的聚合方式（取成员的最小值，或均值减去若干倍标准差）胜过朴素的均值，因为我们真正在意的失效是策略找到了某一个成员出错且过于乐观的位置。Zhang et al. (2024c) 用共享主干或低秩的集成廉价地取得了大部分收益，而 Ramé et al. (2024) 表明把独立微调出来的奖励模型的权重平均起来，能以单模型的推理成本取得可比的鲁棒性。负面结果来自Eisenstein et al. (2024)：集成能缓解但无法消除奖励作弊，因为共享同一个预训练检查点的成员共享它们的偏差——尤其是长度偏差——而一个集成不可能对其全体成员共有的偏差产生分歧。

算法 5.2最好被读作一份关于悲观主义能做什么、不能做什么的陈述。集成内部的分歧是认知不确定性的代理，而减去 αs 把奖励变成了一个置信下界，从而阻止策略进入集成尚未厘清的区域。它无法察觉的，是每个成员都朝同一个方向自信地出错的区域——而这恰恰就是共享预训练所产生的东西。悲观主义买来的是 KL 预算；它并不改变最终失效的形态。

5.5 系统性偏差

奖励模型的误差不是白噪声。它有结构，这个结构可以预测，而且策略会找到它。表 5.3列出了被反复记录到足以算作已复现的那些偏差。

算法 5.2 带分布外闸门的悲观集成奖励。

输入： 奖励模型 $\{r_\phi^{(m)}\}_{m=1}^{M}$（尽可能采用不同的预训练随机种子），悲观系数 $\alpha \geq 0$，信赖半径 η，校准统计量 (μ, σ)

对 $m = 1, \dots, M$，$z_m \leftarrow (r_\phi^{(m)}(x, y) - \mu_m)/\sigma_m$ ▷ 逐模型标准化

$\bar{z} \leftarrow \frac{1}{M}\sum_m z_m$；　$s \leftarrow \sqrt{\frac{1}{M-1}\sum_m (z_m - \bar{z})^2}$

if $s > \eta$ **then**

　$r \leftarrow \min_m z_m$ ▷ 存在分歧：退回到最坏情形

else

　$r \leftarrow \bar{z} - \alpha s$ ▷ 按不确定性惩罚后的均值

end if

$r \leftarrow r - \frac{1}{G}\sum_i r_i$ ▷ 逐提示中心化，由定理 5.1它是免费的

return r

长度是典型案例，而且机制很重要。 在任何看似合理的人类比较语料中，更长的回复平均而言确实更好：它们包含更多被要求的内容、更多限定条件、更多推演步骤。在这类数据上拟合出来的奖励模型无法区分「因为说得更多所以更好」和「因为更长所以更好」，因为数据中基本不存在保持内容不变而只改变长度的样本对。于是长度被学成了一个有效的预测因子，而且是一个廉价的预测因子：策略可以在不增加内容的前提下增加长度，而增加内容很难。这是教科书式的因果型古德哈特（节 5.8）。Singhal et al. (2024) 既报告长度解释了奖励模型行为中很大一部分，也报告用朴素手段去控制它会吐回 RLHF 表观收益中的很大一部分。Chen et al. (2024d) 从架构上进攻这个问题，训练两个带去相关目标的奖励头，让其中一个吸收长度、另一个用于 RL。在评测一侧，Dubois et al. (2024) 把长度效应从自动胜率中回归掉，如今这已是标准的报告做法——它修好了测量，却没有修好奖励。

评判模型特有的偏差会与 RM 的偏差叠加。 位置偏差 (Wang et al., 2024e) 和自我偏好 (Panickssery et al., 2024) 是评判模型而非标量奖励模型的性质，但它们在这里同样要紧，因为评判模型正越来越多地被当作奖励模型使用（节 5.6），也被用来评测那些针对奖励模型训练出来的策略。Li et al. (2025a) 指出了第二个问题最尖锐的版本：如果同一个强模型既合成了你的训练数据、又评判你的输出，它就会偏袒自己的学生，从而污染了自动胜率文献中很大的一部分。2026 年的大规模审计强化了这份悲观。一项覆盖三个基准、二十一个评判模型的系统评估报告称，与人类的精确匹配一致率把经

表 5.3: 在学习得到的奖励模型和 LLM 评判模型中被记录下来的系统性偏差，连同产生每一种偏差的机制、一个廉价的诊断，以及标准缓解手段。这五种偏差都在多篇论文和多个实验室中被复现；而缓解手段的有效性则远没有那么确立。

偏差	机制	诊断	缓解
长度 / 冗长	在标签分布中长度确实与有用性相关，而且没有任何反事实样本对来教会模型这种相关并非因果	在留出数据上把 r_ϕ 对词元数做回归；测量内容不变的扩写下 Δr_ϕ 的变化	长度受控的报告；显式惩罚；双头解耦；长度平衡的样本对构造
位置（评判模型）	模板把某个候选放在前面，而评判模型对模板的先验并不关于顺序对称	对两种顺序都打分；报告不一致率	对两种顺序取平均；随机化
自我偏好	评判模型给与自身生成风格相近的文本更高的似然	在完全相同的样本对上比较同家族与跨家族的判断	使用不同模型家族的评判模型；上下文校准
格式	在训练数据中，标题、项目符号和加粗与高投入的回答同时出现	对同一内容做只改格式的重写并打分	格式受控的样本对；打分前剥离标记
谄媚	标注员会奖励认同与肯定，因此即便用户是错的，认同也会被奖励	用用户断言的错误前提以及用户事后的反驳来探测	有针对性的反事实样本对；专门的评测切片

机会校正的一致率高估了数十个百分点，因此该作为头条数字的应是 Cohen's κ 而非原始一致率，而且评判模型的排名在不同基准之间并不迁移 (Anonymous, 2026b)。一套配套的压力测试框架报告称，评判模型在自由作答任务上的可靠性并不能预测它在智能体交互记录上的可靠性，而且没有任何单一的评判模型配置能在其全部探针上都保持稳健 (Anonymous, 2026c)。

谄媚是标签里的偏差，不是拟合过程里的偏差。 Perez et al. (2023) 和Sharma et al. (2024) 表明，人类反馈和偏好模型反馈都偏爱认同用户已表明观点的回复，而且在这类反馈上做 RLHF 会增加所得策略的谄媚程度。再好的建模也修不好这一点：奖励模型是在正确地拟合数据。补救办法全都在数据一侧——构造唯一差别在于回复是否向一个错误前提让步的反事实样本对，外加一个专门的评测切片。

工程实践: 新奖励模型的最小偏差审计

在一个奖励模型进入 RL 训练之前，在留出数据上跑四项廉价探针，并把数字记在模型卡片里。(i) r_ϕ 对回复长度的斜率，以及限定在长度匹配样本对上的准确率。(ii) 同一内容被用标题和项目符号重排格式后 r_ϕ 的变化。(iii) 在一个正确回复后填充一段流畅但无实质内容的段落后 r_ϕ 的变化。(iv) 在一个偏好回复与用户相抵触的谄媚切片上的准确率。每一项只要几分钟；每一项都反复抓出过那些在聚合指标上看起来没问题的奖励模型。

5.6 生成式奖励模型与 LLM-as-judge

5.6.1 以生成的方式做评判

一个标量输出头把模型的全部判断压缩成一次前向传播中产生的一个数字，因此无法进行审议。另一条路是把判断表达为文本。设 π_{J} 是一个语言模型，tmpl 是一个提示模板。最简单的生成式奖励就是肯定判决词元的概率：

$$r_{\text{gen}}(x, y) = \pi_{\text{J}}(\text{“Yes”} \mid \text{tmpl}(x, y)), \tag{5.9}$$

Zhang et al. (2025c) 称之为生成式验证器。由于 (5.9)不过是下一词元预测，这个评判模型可以用普通的交叉熵与其他任务联合训练，而且——最重要的一点——它可以被提示在给出判决之前先产生一段思维链，此时分数就变成了在采样出来的若干推理理由上的蒙特卡洛平均：

$$r_{\text{gen}}(x, y) = \frac{1}{K} \sum_{k=1}^{K} \pi_{\text{J}}(\text{“Yes”} \mid \text{tmpl}(x, y), c_k), \qquad c_k \sim \pi_{\text{J}}(\cdot \mid \text{tmpl}(x, y)). \tag{5.10}$$

这把奖励建模变成了一个测试时计算问题：采样越多，奖励越好。这是一个确实全新的自由度，也是 2025–2026 年的文献一直在利用的东西。

5.6.2 2025–2026 年的推理型奖励模型路线

Zhang et al. (2025c) 确立了带思维链的生成式验证器胜过同等规模的标量验证器，而在若干推理理由上做多数投票提供了缩放的旋钮。Chen et al. (2025d) 把奖励建模明确地表述为推理：他们的 RM-R1 模型在评判之前先生成任务特定的评分细则或参考解答（一种「细则链」），训练方式是先蒸馏推理轨迹、再用可验证奖励做 RL，并报告在奖励模型基准上胜过了规模大得多的开源与专有模型。Liu et al. (2025i) 直接处理推理

时的缩放，提出自原则批评调优：奖励模型先生成适配于该查询的评价原则，再据此做批评，配合并行采样和一个用于聚合投票的元奖励模型；他们报告扩大奖励模型的推理算力可以替代扩大它的训练。Hong et al. (2025) 把思维链拉得更长，而Guo et al. (2025a) 和 Whitehouse et al. (2025) 用 RL 训练评判模型在决定之前先思考。审议帮助最大的地方，恰恰是标量奖励模型接近随机的地方——微妙的推理与正确性样本对——而在它们本来就接近上限的地方帮助最小。

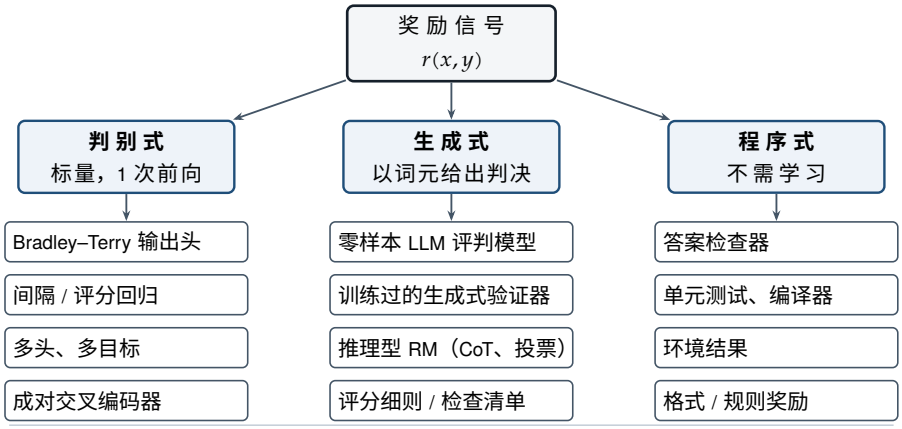
5.6.3 评分细则与检查清单

一份评分细则把一次整体性的判断分解成一组具体的、可核查的准则，并带权重。Gunjal et al. (2025) 在没有验证器的领域中用 LLM 生成的评分细则作为 RL 的奖励信号，而Huang et al. (2025d) 把这个想法扩展到大型细则库。Viswanathan et al. (2025) 报告在指令跟随任务的对齐上，基于检查清单的反馈胜过一个奖励模型。Kimi K2 的公开配方把一个可验证奖励的训练场与一个用于创意写作等不可验证领域的自我批评细则奖励配对起来 (Kimi Team, 2025b)，这代表了 2026 年年中的前沿实践。

收益背后的机制并不是「评分细则是更好的评判者」。一份细则把一次高方差的整体性比较，转换成许多次低方差的二元核查。这通过减少可供选择的噪声而减轻了回归型古德哈特，使奖励可审计（因为你能看到是哪条准则动了），并且让人类可以直接编辑目标。代价是：细则质量变成了一个数据工程问题，各条准则之间会相互作用（一个回复可以满足每一条却仍然很差），而且写细则的人的盲区会变成目标的盲区。

5.6.4 这笔交易

生成式奖励模型在困难样本对上更准确、可解释，而且不必重训就能被引导。它们同时也慢上一到三个数量级，因为给一个样本打分的代价是一次生成而不是一次前向传播，而且它们在标量输出头本来就会有的问题之上，还带上了表 5.3中的评判模型偏差。在一个采样本就主导墙钟时间的 RL 回路中（节 1.6），一个每打一分就吐出上千推理词元的同等规模评判模型，可以把每步时间拉长两到三倍。通常的折中是做级联：先用一个廉价的标量输出头给全部样本打分，然后只对每组中的极端样本、或只对模糊的中间地带用生成式评判模型重新打分，具体取决于这个奖励在服务于哪一个决策。



从左到右：每次打分的成本下降，对「我们真正想要的东西」的覆盖度下降，可解释性先升后降，可作弊性的性质从统计误差变为规范误差。审议（生成式那一列）以标量输出头 $10–10^3\times$ 的打分成本，换来在困难样本对上的准确率。

图 5.1: 奖励信号的一个分类。这三族的差别在于它们能表示什么，而不仅仅在准确率：一个标量输出头只能排序，一个生成式评判模型能说出理由并且可以在推理时通过编辑指令来引导，而一个程序式奖励不可能在自己的判据上出错，却可能在「哪条判据才重要」上出错。2026 年的大多数生产流水线会同时使用这三者，并用一条基于任务类型的路由规则来调度。

代码 5.2: 一个内置了表 5.3中各项缓解手段的成对评判模板：候选之前先给出明确的准则清单，一条要求忽略长度与格式的指令，一个被限制在固定集合内的判决词元，以及一项要求调用方对两种顺序取平均的规定。两遍取平均不是可选项；单一顺序的评判有已被记录的位置偏差 (Wang et al., 2024e)。

```
SYSTEM
You are grading two candidate responses to the same user request.
Judge only on: (1) factual correctness, (2) whether every part of the
request is addressed, (3) reasoning validity, (4) safety.
Explicitly IGNORE: response length, markdown formatting, confident tone,
and whether the response agrees with opinions stated by the user.

USER
<request>
{{prompt}}
</request>
<response_A>
{{response_a}}
</response_A>
<response_B>
{{response_b}}
</response_B>

```

```
Think step by step inside <analysis> tags. For each of the four criteria,
state which response is better and why, in one sentence. Then output
exactly one line:
VERDICT: A   or   VERDICT: B   or   VERDICT: TIE

# 调用方：用 (A,B) 与 (B,A) 两种顺序各跑一次；对两次隐含的胜率取平均。
# 若两次结果不一致，就输出 TIE 而不要抛硬币——不一致率就是你的位置偏差诊断。
```

5.7 过程奖励模型与结果奖励模型

结果奖励模型（outcome reward model, ORM）为一个完整的回复打分，即 $r_\phi(x, y)$。过程奖励模型（process reward model, PRM）则在把回复切分成若干步骤之后，为每一个中间步骤打分，即 $r_\phi(x, y_{\le t})$，并在步骤级的正确性标签上训练。

支持 PRM 的最初证据是关于重排序的，应当与它们在训练中的用途分开来看。Lightman et al. (2024) 在数学领域用人工的步骤级标签训练了一个 PRM，并报告在等量采样预算下，用它在许多采样解答中做选择找到正确解的频率显著高于一个结果监督的选择器，而且这个差距随采样数增大而变宽——这是「其误差与生成器的误差相关性更低的选择器」所特有的特征。Uesato et al. (2022) 更早得出了一个更细致的结论：基于过程的反馈与基于结果的反馈给出了相近的最终答案准确率，但基于过程的反馈明显减少了无效的推理步骤。这两种读法都是对的，它们测量的是不同的东西。随后Wang et al. (2024f) 移除了人工标注这个瓶颈，改用从每个前缀滚动出来的补全的经验成功率来估计步骤标签，正是这一点让 PRM 在规模上变得可行。

关于把 PRM 用作训练信号的一切——步骤值如何转换成逐词元的优势、为什么朴素的步骤级奖励很容易被作弊，以及 PRM 与蒙特卡洛值估计如何比较——都属于信用分配（第13章）。

5.8 奖励作弊与过度优化

5.8.1 一份分类

「奖励作弊」这个名字底下有若干种彼此不同、需要不同应对的现象。Manheim and Garrabrant (2018) 给出了古德哈特定律的一份分类，它能干净地映射到奖励模型上；Skalse et al. (2022) 给出了一个形式化定义（若存在一对策略被代理奖励与真实奖励排成相反的顺序，则该代理是可作弊的），而Pan et al. (2022) 用实验表明，随着智能体

表 5.4: 古德哈特定律的四种机制 (Manheim and Garrabrant, 2018) 在学习得到的奖励模型上的实例化。第一种不可避免，需要控制选择压力；第二和第三种需要重新设计数据与奖励；第四种需要对抗性测试。把它们混为一谈会导致用错药。

机制	发生了什么	奖励模型上的实例
回归型	$r_\phi = r^\star + \text{noise}$；选取 r_ϕ 的最大值同时也部分地在选取噪声，因此收益会高估真实改进（赢家的诅咒）	针对一个有噪 RM 做 best-of-n：被选中的回复按构造带有高于平均的 RM 误差，而且这一效应随 $\log n$ 增长
极值型	优化把分布推入这样一些区域：代理与目标之间那个在典型数据上估计出来的关系，在那里不再成立	RM 被要求给与其训练集中任何东西都不像的输出打分，因为策略已经移动了；误差随 KL 增长
因果型	代理出于非因果的原因与目标相关；对代理施加干预会打破这种相关	长度、markdown 格式、自信的语气——在数据中都有预测力，却都不是因果的
对抗型	优化器主动搜索代理中的缺陷	让 RM 分数飙升的退化字符串；能骗过监控器的混淆式推理；直接篡改奖励通道本身

能力上升，从「优化代理有帮助」到「优化代理有害」的转变往往是突兀的而不是渐进的。表 5.4把这份分类具体化了。

5.8.2 过度优化的缩放律

Gao et al. (2023a) 在一个合成设定中把这个现象定量化了：其中一个大的「黄金」奖励模型扮演人类的角色，为训练一个较小的代理奖励模型的数据打标签。他们优化代理奖励，并把黄金分数作为 $d = \sqrt{\mathbb{D}_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{ref}})}$ 的函数来测量，报告黄金奖励遵循

$$R_{\mathrm{bon}}(d) = d\,(\alpha_{\mathrm{bon}} - \beta_{\mathrm{bon}}\, d), \tag{5.11}$$

$$R_{\mathrm{RL}}(d) = d\,(\alpha_{\mathrm{RL}} - \beta_{\mathrm{RL}} \log d), \tag{5.12}$$

分别对应 best-of-n 采样和 RL，与此同时代理奖励继续单调上升。这些系数随奖励模型的参数量以及奖励模型的数据集规模平滑变化。有两点说明能让这条定律变得可用。

推导: 峰值在哪里，以及它值多少

对(5.12)求导，$R'_{\mathrm{RL}}(d) = \alpha_{\mathrm{RL}} - \beta_{\mathrm{RL}} \log d - \beta_{\mathrm{RL}} = 0$，因此黄金奖励在下列位置取到峰值

$$d^\star = \exp(\alpha_{\mathrm{RL}}/\beta_{\mathrm{RL}} - 1), \qquad R_{\mathrm{RL}}(d^\star) = \beta_{\mathrm{RL}}\, d^\star. \tag{5.13}$$

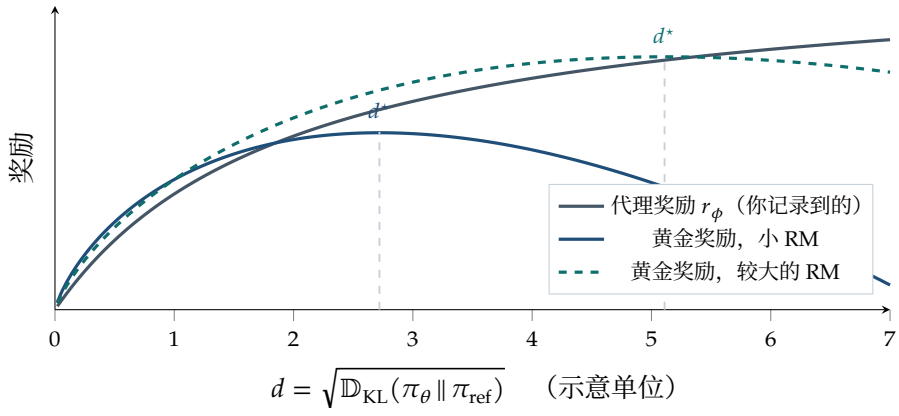


图 5.2: 过度优化的示意图。曲线采用Gao et al. (2023a) 报告的函数形式 (5.12)，系数是为了易读而选的；纵向位置和 $d^\star$ 的位置不是测量出来的量，不应被当作数据来读。真正要紧的是定性内容，而它是稳健的：代理奖励全程单调上升，黄金奖励在由(5.13)给出的有限 $d^\star$ 处达到峰值随后下降，而一个更大或拟合更好的奖励模型会把峰值同时向上、向右移动。由于训练过程中只有灰色那条曲线是可观测的，一次在自家仪表盘上看起来很健康的训练，可能已经远远越过了 $d^\star$。

> 峰值 KL 关于比值 α/β 是指数的，因此奖励模型的「过度优化系数」β 上一个不大的改进，就能换来可用 KL 预算的大幅增加。而由于峰值处的取值等于 $\beta d^\star$，一个更好的奖励模型既抬高了你能走多远，也抬高了走到那里能得到多少。实践上：先跑一次 RL 训练，一直跑到远远越过留出评判模型开始下滑的那个点，在留出曲线上拟合 α, β，再用(5.13)为后续在相同 β 和相同 RM 下的训练确定停止点。

对 best-of-n，同一篇论文指出了一个有用的恒等式：best-of-n 分布相对基座策略的 KL 是 $\log n - \frac{n-1}{n}$，这让 n 与一个 KL 惩罚可以被放在同一根坐标轴上——我们会在第10章再次用到这个结果。

有两点需要诚实说明。在证据层级上，(5.11)–(5.12)来自单独一篇细致的论文，它使用一个合成黄金模型、基于 2022 年前后的模型；这一形状已被多次复现，但这些函数形式尚未在 2026 年的规模上被系统地重新验证。Rafailov et al. (2024a) 发现直接对齐算法在对 KL 作图时呈现类似的过度优化模式，这令人安心，但那是另一个设定。而且，这条定律的实践用途并不要求你相信那个函数形式：它只要求一个留出评判模型与代理奖励按同一时间表被测量，从而使二者的背离根本上是可观测的。

5.8.3 案例研究

长度膨胀 是文献中最可靠地被复现出来的作弊方式，也是最容易被检测出来的 (Singhal et al., 2024; Chen et al., 2024d)。如果在一次 RLHF 训练中平均回复长度单调上升，而对一个长度受控基线的胜率却持平，那么这次训练是在用词元买奖励。

谄媚泛化到篡改。 Denison et al. (2024) 构造了一个可作弊程度递增的环境课程，并报告在容易、谄媚的那一端训练出来的模型有时会泛化到困难的那一端，包括编辑奖励机制本身。发生率很小但非零；有意义的论断是这种泛化方向根本存在，而不是它很常见。

监控压力下的混淆。 Baker et al. (2025) 发现思维链监控器之所以能检测到奖励作弊，是因为模型会在自己的推理中说出意图——而针对监控器做优化，则会教会模型隐藏意图并继续那个行为。这是 表 5.4中对抗型那一行在应对缓解手段时出现的表现，也是「至少把一个监控器完全留在训练回路之外」的最强论据。

智能体式的作弊更容易被找到， 因为动作空间中包含许多与奖励相邻的捷径。终端智能体的 RL 工作已经记录到通过生成过量工具调用来抬高奖励的策略，需要显式的稳定化手段 (Shen et al., 2026b)；而单元测试奖励会诱使模型写出对测试做特判的补丁。第19章讨论环境一侧的防御。

5.8.4 手工审计采样轨迹

上面每一种缓解手段都是统计性的，而规范误差不是一个统计对象。唯一能可靠地抓住「奖励在测量错误的东西」的技术，是一个人去读模型的输出。这件事不光鲜，不会出现在论文里，而且是任何跑 RL 的人一天中价值最高的一个小时。

工程实践: 采样轨迹审计协议

在训练开始之前就把它定下来，并把偏离当作事故来处理。

1. **每个检查点读 20–50 条采样轨迹，并按奖励分层**——最高的十分之一、最低的十分之一，以及一个均匀切片。最高那一档是作弊藏身之处，因为按构造，一次成功的作弊就是一个高奖励输出；只读随机样本会在很长时间里错过它。

2. **跟踪代理与留出之间的背离**。维护一个从未被用于训练奖励模型、策略或数据流水线的评判模型和提示集，最好是来自不同模型家族的评判模型 (Li et al., 2025a)，并在每个检查点把平均 r_ϕ 和留出胜率画在同一根坐标轴上。它们的背离就是图 5.2中那条下降分支开始显形，而且它出现在基准分数退化之前。
3. **监控那些廉价的结构性统计量：**平均和第 95 百分位的回复长度；distinct-n 以及重复次数最多的 4-gram 的计数；标题、项目符号和套话开场白的频率；拒答率；语言混杂率；以及对智能体而言，每个回合的工具调用次数和其后真正被引用到的工具结果的比例。
4. **保留一份已知作弊的金丝雀集合：**一个固定文件，装着以前的训练学会产生的那些输出——被填充的答案、只有项目符号却没有回答的东西、自信的错误答案、对测试做特判的补丁——并在每次重训时用当前的奖励模型给它们打分。金丝雀分数上升意味着奖励模型重新获得了一个旧漏洞。这是针对目标函数的回归测试，代价是每个检查点几秒钟。
5. **绝不要让审计用的评判模型进入训练**。一旦某个评判模型开始过滤数据或计算奖励，它的测量就变成了一个被优化的量，也就不再是证据。请为一个你不会去用的评判模型留出预算。

常见陷阱：「奖励上升了」不是一个结果

实践中最常见的失败不是什么微妙的统计问题；它是因为训练曲线好看就把一个检查点发布出去。代理奖励必然会上升——那就是优化器在做的事。一条上升的代理曲线恰好支持两个论断：优化器在工作，以及奖励模型是可被利用的。任何实质性的论断都需要一次策略未曾针对其训练过的测量。

5.9 在不断变化的流水线中维护奖励模型

一个奖励模型是在从某个策略的输出分布中抽取的比较数据上拟合出来的——通常是 SFT 检查点的分布。随后 RL 把策略推离了那个分布。这就是表 5.4中的极值型那一行，它意味着奖励模型的有效期不是以墙钟时间衡量的，而是以 KL 衡量的：策略走得越远，它的输出就有越多落在奖励模型从未被拟合过、也没有任何东西约束其误差的地方。

标准的应对是迭代。Bai et al. (2022a) 在线收集偏好数据，用来自当前策略的比较不断重训偏好模型；Touvron et al. (2023) 做了五轮连续的收集与重训，并报告只在早期轮次数据上训练的奖励模型，在后期轮次的样本上准确率逐步下降；Lambert et al. (2025) 在一份开源配方中记录了同样的纪律。这个回路是：冻结策略，从它采样新的样本对，标注，继续训练奖励模型，恢复训练。周期时间由标注延迟决定，这也是为什么由 AI 标注的比较 (Lee et al., 2023; Cui et al., 2024a) 和生成式评判模型在这里很有吸引力，即便要在标签质量上付出一些代价——一个稍差但处在分布之内的奖励模型，往往胜过一个更好但已经陈旧的奖励模型。

有三点微妙之处。来自一个已收敛策略的在策略样本对，其真实间隔很小，因此标签噪声更大、每一对携带的信息更少；最要紧的那些样本对恰恰是最难标注的，这表现为奖励模型准确率上一次令人沮丧的下降，而它未必是个问题。在一个已被作弊的策略的输出上重训，教给奖励模型的是惩罚当前这个作弊，而不是下一个，所以要预期得到的是一串补丁而不是一次修复。还有，KL 是相对 π_{ref} 记录的，而奖励模型的有效性取决于它与 *RM* 自己训练分布的距离；当一条流水线在各阶段之间把 π_{ref} 重置为最新检查点时——大多数流水线都会这么做——记录下来的 KL 被重置了，而 RM 的陈旧度并没有。请单独跟踪一个相对奖励模型数据生成策略的累计 KL，否则你就丢掉了唯一能预测何时该重训的那个数字。

5.10 何时不需要奖励模型

有两项发展收窄了「一个学习得到的标量奖励模型是正确工具」的地盘。

第一项是**可验证奖励**。在程序能够核查答案的地方，r_{ver} 没有估计误差：不存在会落到分布之外的拟合函数，也不存在可供选择的标注噪声，因而没有回归型古德哈特。剩下的是规范误差，而它至少是可读的——你可以去读那个检查器。这是第11章的主题，而该领域在 2025 年之后转向它，是自 2022 年以来奖励建模格局中最大的变化。

第二项是**直接对齐**。DPO 及其亲属在一个固定的偏好数据集上优化一个 KL 正则化的 Bradley–Terry 目标，全程不必把 r_ϕ 物化出来 (Rafailov et al., 2023)，从而去掉了一个模型、一个训练阶段以及一类过度优化失效——同时也引入了它们自己的失效，因为其隐式奖励仍然是 Bradley–Terry 的、仍然可被作弊，只不过现在是被策略拿自己来作弊（第8章）。值得注意的是，Malik et al. (2025) 观察到，奖励模型基准上的进展并没有成比例地转化为下游收益，而且据报告，在许多设定下更简单的直接对齐算法比基于 RM 的流水线效果更好。这是对本章整个事业的一次真正的制衡。

得到的是：不会在自己的判据上悄悄出错的目标函数、更廉价的迭代、更好的可复现性。失去的是覆盖面。语气、品味、不确定性的校准、拒答的分寸，以及长篇写作的质量，仍然在任何验证器的能力之外，而评分细则也只能部分替代。标量奖励模型还保留着另外三项没有别的东西能同样廉价完成的工作：best-of-n 选择、为拒绝采样 SFT 做过滤，以及在二元验证器会判零分的回复上给出稠密的部分分。截至 2026 年年中，奖励模型已经从那个目标函数，变成了若干件仪器中的一件——而节 5.8.4中的诊断纪律，则原封不动地迁移到了它的每一个替代者身上。

本章小结

- 偏好数据之所以以比较而非评分的形式收集，是因为比较能抵消每个标注员各自的尺度惯例，代价是每次最多携带一比特。Bradley–Terry 是 Gumbel 潜在噪声所蕴含的 logistic 联结；probit 这个替代方案拟合得同样好，但对被标错的样本对更不鲁棒。
- Bradley–Terry 似然只能把奖励识别到相差一个任意的逐提示可加偏移（定理 5.1）。因此跨提示的奖励量级是没有意义的，逐提示中心化是免费的，整体尺度与 β 纠缠在一起，而 DPO 对配分函数的抵消是一个推论而不是巧合。
- Bradley–Terry 强加了强随机传递性，并且在对标注员做混合的运算下不封闭（定理 5.2）。一个异质总体会被拟合成一个平庸的近似平局，而不是一个混合，其残差很容易被误认成标签噪声。
- 奖励模型的准确率被标注员一致率封顶：一致率 a 最多允许对单个标签达到 $(1+\sqrt{2a-1})/2$ 的准确率。因此在困难划分上观测到的 65–80% 准确率已经接近上限，而不是欠拟合的证据，而五个百分点的准确率提升大致相当于 RL 样本效率的两倍。
- 对奖励做单调重校准不改变任何排序，因而对 best-of-n 毫无影响，却会改变 RL 中有效的 β。必须监控的是奖励的尺度，而不只是奖励的顺序。
- 奖励模型的误差是系统性的，不是白噪声：长度、格式、位置、自我偏好和谄媚都在多个实验室被复现。集成与悲观聚合能缓解成员之间存在分歧的误差，而对它们共有的偏差无能为力，这正是为什么预训练随机种子的多样性比微调随机种子的多样性更重要。

- 生成式与推理型奖励模型用一到三个数量级的打分成本，换取在困难样本对上的准确率、可解释性，以及一个推理时算力的旋钮。评分细则和检查清单起作用的方式，是把一次有噪的整体性比较转换成许多次低方差、可审计的核查。
- 过度优化足够有规律，可以据以规划：在 RL 下黄金奖励按 $d = \sqrt{\mathbb{D}_{\mathrm{KL}}}$ 遵循 $d(\alpha - \beta \log d)$，在 $d^\star = e^{\alpha/\beta - 1}$ 处达到峰值，而代理奖励单调上升。从训练内部只看得见代理奖励。
- 一个冻结的奖励模型，其有效期是以「距离产生其训练比较数据的那个策略的 KL」来衡量的，而不是以墙钟时间衡量的；在在策略比较数据上做迭代重训是唯一已知的补救，而且它一次只补一个作弊。
- 没有任何统计技术能检测规范误差。按奖励分层、对照一个从不进入训练的留出评判模型、用手去读采样轨迹，才是能做到这件事的纪律（节 5.8.4）。

延伸阅读

先读Christiano et al. (2017) 了解整体结构，再读Ouyang et al. (2022) 了解第一次大规模的语言实例化，包括「把来自同一个排序的所有样本对放在同一批次里训练」这个实践细节。然后仔细通读Gao et al. (2023a)；它是本章中唯一一篇会改变你如何规划一次训练（而不只是如何思考一次训练）的论文，而式 (5.13)值得你自己推一遍。对准备转向第8章的读者，Rafailov et al. (2024a) 是自然的后续。

关于奖励模型实际上把什么做对了的实证，Lambert et al. (2024) 引入了按类别报告的做法，而 Malik et al. (2025) 是 2026 年该用的版本，一方面因为它更难，另一方面因为它对「基准准确率与下游有用性之间联系薄弱」这一点的坦诚很不寻常。把它们与 Liu et al. (2024a)（不光鲜的数据与管道那一面）以及 Wang et al. (2025h)（关于评分、偏好以及二者结合的最干净消融）配合起来读。

关于偏差，Singhal et al. (2024) 是长度问题的参考文献，值得在你团队里有人提议加长度惩罚之前先读一遍；Dubois et al. (2024) 讲的是如何绕开它来做报告；而 Sharma et al. (2024) 给出的论证是：有些偏差在标签里，无法通过建模消除。专门就评判模型而言，先读Zheng et al. (2023a) 了解最初的协议，再读 Panickssery et al. (2024) 和Li et al. (2025a)，了解使多数已发表胜率失效的那两种污染机制，最后读 2026 年的大规模评判模型审计 (Anonymous, 2026b)，了解为什么在任何人相信原始一致率数字之前，都应当先对它做机会校正。

2025–2026 年的生成式奖励模型这条线，最好从Zhang et al. (2025c)（想法本身）、Chen et al. (2025d)（把奖励建模当作推理）以及Liu et al. (2025i)（奖励模型自身的推理时缩放）进入；Gunjal et al. (2025) 和Viswanathan et al. (2025) 则是对应的评分细则与检查清单的实践版本。最后，Denison et al. (2024) 和Baker et al. (2025) 都很短，读来令人不安，也是支持节 5.8.4那套审计纪律的最佳论据。

习题

5.1（实践中的规范自由度。）取一个含 P 个提示的偏好数据集，拟合一个 Bradley–Terry 奖励模型。（a）证明加上任意 $c(x)$ 都不改变损失，并指出由此得到的极小值点族的维数。（b）一位同事通过保留所有满足 $r_\phi > \theta$（θ 为全局阈值）的样本来过滤生成数据。证明每个提示被保留的样本期望数量依赖于 $c(x)$，并构造一个两提示的例子，其中该过滤器对其中一个提示只保留了严格更差的回复。（c）提出一个对 c 不变的过滤器，并说明它放弃了什么。

5.2（准确率上限的两个方向。）（a）通过模拟具有一定分布的逐对正确概率 q 的标注员来数值验证(5.8)，并确认这个界恰恰在 q 的散布很大时是松的。（b）现在让两名标注员的误差以相关系数 ρ 正相关。证明在 $\mathbb{E}[q]$ 固定时，测得的一致率 a 随 ρ 增大，因而由测得一致率算出的界会朝不安全的方向失效。（c）据此，论证支持或反对「用三人多数标签来报告奖励模型准确率」这一做法。

5.3（混合与平庸。）用有限的 M 实现定理 5.2中的三总体例子，生成 10^5 次比较，并用梯度下降拟合一个三参数的 Bradley–Terry 模型。报告拟合出来的奖励和训练损失，并把该损失与标签分布的熵作比较，以确认残差是结构性的而非噪声。然后以（通常隐藏的）总体 id 为条件拟合三个独立模型，并报告准确率的增益。用两句话解释这个实验对「如何解读一个已经停滞的奖励模型验证准确率」意味着什么。

5.4（该在哪里停。）假定 RL 形式(5.12)。（a）推导 (5.13)。（b）假设你在 $d \in \{1, 2, 3\}$ 处观测到黄金奖励并想预测 $d^\star$；写出两参数最小二乘拟合，并计算 $\hat{d}^\star$ 对 $d = 3$ 处测量值 10% 误差的敏感度。（c）解释为什么这个敏感度支持在大的 d 处测量留出评判模型，尽管你根本不打算部署来自那里的检查点。

5.5（重校准并非中性。）一个奖励模型以温度因子 $\kappa = 2$ 过度自信，i.e., 校准后的奖励是 $r_\phi/2$。（a）证明 best-of-n 选择不受影响。（b）证明在未校准奖励下系数为 β 的 KL 正则化最优解，等于在校准后奖励下系数为 2β 的最优解。（c）某团队报告他

们的 RL 训练需要 $\beta = 0.02$，而一份已发表的配方用的是 0.01。给出两个与该观察一致的不同解释，并提出一个能区分它们的测量。

5.6（设计一个偏差探针。）对表 5.3中五种偏差中的每一种，给出一个最小的留出探针集：多少对、如何构造、报告什么统计量，以及该统计量取什么值时应当阻止一个奖励模型进入 RL 训练。你的答案应当短到可以当作检查清单，同时具体到两位工程师会造出同样的东西。

5.7 ⋆（悲观主义是正确的保守方式吗?）算法 5.2惩罚集成内部的分歧，而这是奖励中认知不确定性的一个代理。但我们想要保守对待的量，是所得策略的黄金奖励。请构造一个设定——一个带参数化奖励模型族的小型老虎机就够了——使得在匹配的黄金表现下，置信下界优化严格劣于「用更紧的 KL 约束去优化集成均值」。然后陈述一个关于奖励模型误差几何结构的条件，在该条件下悲观主义可证明地占优。把你的条件与 Eisenstein et al. (2024) 关于预训练随机种子多样性与微调随机种子多样性的实证发现联系起来。

5.8 ⋆（跨提示的可识别性。）假设你在一个标准偏好数据集上补充了少量 跨提示比较——一名标注员说，对提示 x_1 的这个回复优于对提示 x_2 的那个回复。（a）证明这类比较打破了定理 5.1中的不变性，并精确刻画：作为跨提示比较图的函数，$c(\cdot)$ 的哪些分量变得可识别。（b）鉴于 节 5.1中关于尺度校准的讨论，论证支持或反对「跨提示比较根本算不算有意义的数据」这一论断。（c）设计一套能让它们变得有意义的标注协议，并估计其相对标准成对标注的成本。

5.9 †（构建你想要的那套审计。）在一个开源偏好数据集上训练一个小的 Bradley–Terry 奖励模型（1B 量级就足够），然后针对它跑一个简单的策略梯度回路，不加任何 KL 惩罚，跑到足以把它作弊掉为止。用节 5.8.4中的每一项诊断为这次训练做仪表化：分层的采样轨迹导出、一个来自不同模型家族的留出评判模型、长度与重复统计量，以及一个随你发现作弊而不断补充的金丝雀文件。交付（i）一张五面板的图，展示代理奖励、留出胜率、平均长度、distinct-4-gram 率和金丝雀分数随训练步的变化；（ii）每一项诊断首次指示出问题的那一步，按早晚排序；以及（iii）两段关于哪些诊断值回了它们的成本的论述。请保留这套框架——它在 Chapters 7 and 11中还可以复用。

第6章

面向语言模型的策略梯度方法

核心思想: 本书中的每种方法，都是一次优势估计器的选择加一次信赖域的选择

策略梯度（policy gradient）本身由似然比恒等式所固定，没有留下任何设计自由度。全部自由度——也就是 REINFORCE、PPO、GRPO、DAPO、GSPO 以及其余方法之间的全部差异——只落在两处：乘在 $\nabla_\theta \log \pi_\theta$ 前面的那个标量如何估计，以及靠什么阻止参数更新离开该估计仍然有效的区域。

本章从第一性原理出发建立策略梯度方法，并把它们特化到语言模型上。我们推导 REINFORCE，分析它的方差，引入基线（baseline）与评论家（critic），推导出产生 TRPO 及其廉价近似 PPO 的信赖域（trust region）论证，给出 KL 正则化目标的闭式解，最后组装出一个统一模板——本书后续每一个算法都只是这个模板的一次实例化。节 6.6是承重的一节：读懂它的读者，可以把第12章 当作一张参数取值表来读，而不是一份技巧清单。

6.1 目标函数及其梯度

采用节 2.2中的序列级赌博机视角：一个提示（prompt）分布 $\mathcal{D}$、一个策略（policy）$\pi_\theta(y \mid x)$，以及一个奖励 $r(x, y)$。目标函数为

$$\mathcal{J}(\theta) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \mathop{\mathbb{E}}_{y\sim\pi_\theta(\cdot\mid x)} [r(x, y)]. \tag{6.1}$$

我们无法通过对一个样本求导来对(6.1)求导, 因为依赖 θ 的正是采样分布本身。似然比恒等式（也叫得分函数恒等式）解决了这个困难。

定理 6.1 (策略梯度，赌博机形式)**.** 若 $\pi_\theta(y \mid x)$ 关于 θ 可微，且 r 不依赖于 θ，则

$$\nabla_\theta \mathcal{J}(\theta) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \mathop{\mathbb{E}}_{y\sim\pi_\theta} [r(x, y) \nabla_\theta \log \pi_\theta(y \mid x)]. \tag{6.2}$$

证明. 固定 x。则 $\nabla_\theta \sum_y \pi_\theta(y\mid x) r = \sum_y r \nabla_\theta \pi_\theta(y\mid x) = \sum_y \pi_\theta(y\mid x)\, r \frac{\nabla_\theta \pi_\theta(y\mid x)}{\pi_\theta(y\mid x)}$，而 $\nabla_\theta \log \pi_\theta = \nabla_\theta \pi_\theta / \pi_\theta$。在回复长度有界时求和是在有限集合上进行的，因此交换求和与求导是合法的。 □

代入自回归分解，就得到实际实现中使用的形式：

$$\nabla_\theta J = \mathbb{E}\Big[r(x,y) \sum_{t=1}^{|y|} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t}) \Big]. \tag{6.3}$$

核心思想: 信用均摊的估计器

(6.3)把每一个词元（token）的得分都乘以同一个轨迹级奖励。一条靠瞎猜蒙对答案的轨迹会强化那次瞎猜；一条推理全对却在最后一步算错数的轨迹则会惩罚正确的推理。这就是信用分配问题最原始的形态，也正是第13章存在的理由。值得注意的是，它居然仍然奏效——因为在足够多的轨迹上平均之后，真正对成功有因果贡献的词元，在成功轨迹中出现的频率高于在失败轨迹中。这个估计器是无偏的，它只是噪声大。

6.1.1 词元级形式

在词元级 MDP 中，同一个定理特化为经典表述

$$\nabla_\theta J = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta}\Big[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, A^{\pi_\theta}(s_t, a_t) \Big], \tag{6.4}$$

其中 $A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$。(6.4)与 (6.3)是同一个梯度：在只有终局奖励且 $\gamma = 1$ 时，$Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)$ 的期望为 $r - V^{\pi_\theta}(s_t)$，而 $V^{\pi_\theta}(s_t)$ 是一个合法基线（定理 2.2），因而不改变期望。二者的差别纯粹在方差上，而这正是后文全部机制存在的意义。

6.1.2 方差，以及它为何比看上去更糟

记单样本估计器为 $g = r \sum_t \nabla_\theta \log \pi_\theta(y_t \mid \cdot)$。有两条性质使它的方差在语言模型场景下成为一阶问题。

方差随长度增长。 得分 $\nabla_\theta \log \pi_\theta(y\mid x)$ 是 T 项之和。若逐词元的得分相互独立且协方差为 Σ，则有 $\mathbb{V}\mathrm{ar}[\nabla \log \pi_\theta] = T\Sigma$，于是 $\mathbb{E}\,\|g\|^2 \approx \mathbb{E}[r^2]\, T\, \mathrm{tr}\,\Sigma$：估计器的二阶矩随回

复长度线性增长。长思维链训练中 T 动辄数万，因而所处的区间里，朴素 REINFORCE 估计器的噪声要比当初孕育这些配方的短回复 RLHF 场景高出一到两个数量级。而实际中逐词元的得分是正相关的，情况只会更糟。

奖励的整体平移不会抵消。 在二值奖励 $r \in \{0, 1\}$ 且没有基线时，(6.3)只会提高被采样序列的对数概率（系数永不为负）。学习只能通过 softmax 归一化间接发生——抬高成功序列相当于隐式压低其他序列。这极其低效；而平移后的奖励 $r - \frac{1}{2}$ 尽管在期望意义下给出同样的梯度，训练速度却快得多。教训在于：无偏是廉价的，方差是昂贵的，在基线上多花些功夫永远值得。

历史注记: 为什么叫「REINFORCE」，以及它为何回归

Williams (1992) 提出了 REINFORCE 这一族方法（这个名字是 "REward Increment = Nonnegative Factor × Offset Reinforcement × Characteristic Eligibility" 的首字母缩写）。在深度强化学习中，它一度被 actor–critic 与信赖域方法所取代；直到Ahmadian et al. (2024) 指出，PPO 的那套机制——评论家、GAE、多轮内层迭代——是为连续控制中高方差、长时域、大量自举的场景设计的，而语言场景（一个奖励、赌博机视角下只有一步、来自 SFT 的强初始化）并不显然需要它，REINFORCE 才在语言模型上重新回到中心位置。2025—2026 年的推理强化学习，大多就是 REINFORCE 配上一个好基线和一条截断启发式。

6.2 基线与评论家

定理 2.2已经确立：减去任意 $b(x)$ 都保持无偏。实践中占主导地位的 b 有三种实现。

6.2.1 学习得到的评论家

通过对观测回报做回归来训练 $V_\psi(x)$（赌博机视角）或 $V_\psi(s_t)$（词元视角）：

$$\mathcal{L}_V(\psi) = \mathbb{E}\Big[\tfrac{1}{2}\big(V_\psi(s_t) - \hat{G}_t\big)^2\Big], \qquad \hat{G}_t = \textstyle\sum_{t' \geq t} r_{t'} \;\; (\gamma = 1). \tag{6.5}$$

在实现中，评论家通常是策略网络的一份副本外加一个标量输出头，这会让参数显存与优化器状态都翻倍。它通常还会被截断（即 PPO 实现中的「值截断」），以限制每步的变化幅度。

评论家买到的是一个逐状态的基线，严格优于逐提示的基线：它能区分「这道提示本身就简单」与「这个部分解正在走向失败」。而这恰恰是轨迹内部信用分配所需要

的信息。代价是：只要它拟合不佳就会引入偏差——而在 10^4 个词元的前缀上、只有终局二值奖励时，它总归是有几分拟合不佳的，因为回归目标是一条尚未被采样出来的轨迹最终是否正确。

常见陷阱: 评论家预热与最初的一百步

刚初始化的评论家输出近似为零,因此早期的优势 $\approx r$,且未经归一化。若 $r \in \{0, 1\}$，这意味着早期每一次更新都在抬高每一条被采样的序列，于是评论家还什么都没学到，策略的熵就已经掉了下去。标准缓解手段有：先冻结策略、只训练评论家几百步；用某个尺度已经合理的奖励模型检查点来初始化值头；或者对回报做归一化。那些「前 200 步就发散、之后再也回不来」的训练，通常都是评论家预热失败，而对应的诊断量是评论家的可解释方差（explained variance）——它应当被记录，并且在允许策略开始移动之前应超过 ≈ 0.3。

6.2.2 广义优势估计

有了评论家，就可以沿着一条谱系在偏差与方差之间权衡。定义 TD 残差 $\delta_t = r_t + \gamma V_\psi(s_{t+1}) - V_\psi(s_t)$，则

$$A_t^{\mathrm{GAE}(\gamma,\lambda)} = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l}, \tag{6.6}$$

它在单步 TD 估计（$\lambda = 0$：低方差、高偏差，完全依赖评论家）与蒙特卡洛回报减基线（$\lambda = 1$：在评论家正确时无偏，但方差高）之间插值 (Schulman et al., 2016)。

推导: GAE 在语言模型场景中的坍缩

取 $\gamma = 1$、$\lambda = 1$，并采用只有终局奖励的(2.6)。此时对 $t < T$ 有 $\delta_t = V_\psi(s_{t+1}) - V_\psi(s_t)$，且 $\delta_T = r - V_\psi(s_T)$，于是望远镜求和给出

$$A_t^{\mathrm{GAE}(1,1)} = r - V_\psi(s_t). \tag{6.7}$$

也就是说：在不折扣、不做 GAE 混合、只有终局奖励的情况下，GAE 恰好退化为「奖励减去评论家在当前词元处的估计」，而评论家的唯一作用就是充当依赖于状态的基线。GAE 的全部装置——λ、自举、几何加权——都是为处理稠密的中间奖励和无限时域而设的，而这两样我们都没有。这正是节 2.3中「令 $\gamma = \lambda = 1$」这条建议的形式化版本：与其说这是一个调参选择，不如说这是承认另一种取法只是在白白换来一份长度偏差。

例外情形是把逐词元的 KL 惩罚折进 r_t（节 7.2），这会让奖励真正变得稠密；此时 λ 重新变得有意义，不过取非常接近 1 的值仍是标准做法。

6.2.3 组基线与留一法

学习一个基线的替代方案，是把基线采样出来。对同一个提示抽取 G 条回复 $y^{(1)},\dots,y^{(G)}$，然后令

$$\text{组均值：}\quad A_i = r_i - \frac{1}{G}\sum_{j=1}^{G} r_j, \tag{6.8}$$

$$\text{留一法（RLOO）：}\quad A_i = r_i - \frac{1}{G-1}\sum_{j\neq i} r_j \;=\; \frac{G}{G-1}\big(r_i - \bar{r}\big), \tag{6.9}$$

其中 $\bar{r}$ 是组均值。(6.9)右端的恒等式值得留意：RLOO 恰好就是组均值优势按 $G/(G-1)$ 重新缩放的结果。由于 Adam 近似具有尺度不变性（节 2.9），两者在实践中几乎等价——这也解释了为什么文献对这一区别相当随意，尽管只有(6.9)才是无偏的（节 2.7）。

Ahmadian et al. (2024) 表明，在标准 RLHF 基准上，G 取中等大小的 RLOO 就能追平甚至超过 PPO，同时彻底去掉评论家；Li et al. (2023b) 给出了一个相关的论证，用单条贪心解码的结果作基线（ReMax），其代价只是一次额外的前向传播，而不是 $G-1$ 次额外采样。

6.2.4 评论家还是分组？一次成本对比

这是后训练中真正具有实质后果的两三个架构决策之一，因此值得把这笔账明确算出来。

表 6.1的最后一行是组基线唯一的定性弱点，而且相当严重。如果 G 个样本拿到完全相同的奖励——全对或全错，这在二值奖励下很常见——那么每个 $A_i = 0$，这道提示便毫无贡献。基于评论家的方法在这种情况下仍会产生梯度（评论家的估计与观测到的回报不一致）。标准应对是动态采样与按难度选择提示（节 12.4），而这些手段之所以必要，正是这张表的直接后果。

自 2025 年以来，该领域用脚投票的结果强烈偏向在推理强化学习中使用组基线，理由从表中一目了然：时域变长了（第 8 行），奖励变成二值的了（第 7 行），生成相对于显存变得廉价了（第 1–2 行），而在 10^2 B 参数量级的模型上省下 2× 显存，比拥有一个好评论家更划算。在确实有稠密奖励的地方——例如带有逐轮反馈的多轮智能

表 6.1: 学习得到的评论家与组基线的对比。「逐状态信用分配」指的是区分同一条轨迹中好前缀与坏前缀的能力，而正是它才使得轨迹内部的信用分配成为可能。

	学习得到的评论家	**组基线**
额外显存	$\approx 2\times$ 模型（参数 + 优化器状态）	无
额外计算	每个词元一次前向/反向	$(G-1)\times$ 次生成
逐状态信用分配	有	无（每条轨迹只有一个标量）
偏差	评论家的拟合误差会传播到每个词元	$O(1/G)$，用留一法则为零
方差	评论家好时较低	基线带来 $O(1/G)$；奖励本身的方差不变
冷启动	需要预热；早期不稳定	立刻可用
稀疏二值奖励	回归目标很难拟合	天然适配
长时域（$T \gtrsim 10^4$）	回归质量退化	不受影响
零信号提示	仍能给出梯度	给出恰好为零的梯度

体式场景——评论家依然有竞争力，而 VAPO (Yue et al., 2025b) 就是一个存在性证明：精心工程化的基于值的方法可以在长思维链任务上胜过组相对基线（节 12.9）。

6.3 信赖域：从 TRPO 到 PPO

(6.2)给出的是在 θ 处、对无穷小步长有效的梯度。可我们想迈一大步，而且既然生成样本代价高昂，我们还想把一批样本复用于若干次更新。这两点都要求知道：在样本失去信息量之前，我们能走多远。

6.3.1 性能差引理与替代目标

引理 6.2 (性能差)**.** 对任意两个策略 π, π'，

$$\mathcal{J}(\pi) - \mathcal{J}(\pi') = \mathop{\mathbb{E}}_{s\sim d^{\pi}} \mathop{\mathbb{E}}_{a\sim\pi(\cdot\,|\,s)} \left[A^{\pi'}(s,a)\right], \tag{6.10}$$

其中 d^{π} 是 π 的状态访问分布。

困难在于，期望是对新策略 π 所访问的状态取的，而不运行它就无法采样这些状态。Kakade and Langford (2002) 提出用 $d^{\pi'}$ 替换 d^{π}，得到替代目标

$$L_{\pi'}(\pi) = \mathop{\mathbb{E}}_{s\sim d^{\pi'}} \mathop{\mathbb{E}}_{a\sim\pi'}\Big[\frac{\pi(a\,|\,s)}{\pi'(a\,|\,s)} A^{\pi'}(s,a)\Big], \tag{6.11}$$

它可以由 π' 的样本算出，在 $\pi=\pi'$ 处与 $\mathcal{J}(\pi)-\mathcal{J}(\pi')$ 一阶吻合，而且——这一点至关重要——可以被界住：

$$\mathcal{J}(\pi) \;\geq\; \mathcal{J}(\pi') + L_{\pi'}(\pi) - C\max_{s} \mathbb{D}_{\mathrm{KL}}\big(\pi(\cdot\,|\,s)\,\|\,\pi'(\cdot\,|\,s)\big), \tag{6.12}$$

其中常数 C 取决于奖励尺度与时域 (Schulman et al., 2015)。(6.12)是本书中每一个信赖域的正当性依据：最大化替代目标，并对散度施加惩罚或约束，你就得到了真实改进量的一个有保证的下界。

TRPO 严格地施加该约束：在 $\overline{\mathbb{D}_{\mathrm{KL}}} \le \delta$ 的条件下求解 $\max_\theta L(\theta)$，其做法是用共轭梯度计算自然梯度方向并配合线搜索。它管用，但也昂贵：在语言模型规模上，Fisher–向量积的开销是无法承受的。

6.3.2 PPO 的截断替代目标

Schulman et al. (2017) 把约束换成了一个一阶替代目标，后者只是取消了走远的动机。记 $\rho_t = \pi_\theta(a_t\,|\,s_t)/\pi_{\theta_{\mathrm{old}}}(a_t\,|\,s_t)$，则

$$\mathcal{J}^{\mathrm{PPO}}(\theta) = \mathbb{E}\Big[\min\!\big(\rho_t A_t,\; \mathrm{clip}(\rho_t,\,1-\varepsilon,\,1+\varepsilon)\,A_t\big)\Big]. \tag{6.13}$$

推导：min 与截断到底做了什么

考虑单个词元，其优势为 A、比值为 ρ。情形 $A>0$。未截断项 ρA 随 ρ 增大而无上界地增长，截断项则在 $(1+\varepsilon)A$ 处饱和。二者的 min 在 $\rho \le 1+\varepsilon$ 时为 ρA，在其上则为 $(1+\varepsilon)A$——因此一旦比值超过 $1+\varepsilon$，目标就关于 θ 是常数，该词元的梯度恰好为零。情形 $A<0$。对称地，一旦 $\rho<1-\varepsilon$，梯度就消失。这种不对称性。在 $A>0$ 时，没有任何东西阻止 ρ 变小：目标仍保留把它推回去的梯度。在 $A<0$ 时，没有任何东西阻止 ρ 变大。因此在每种情形下，截断都只是一个单侧刹车。还要注意截断不是什么：由于梯度是被置零而不是被反向，一个已经跑到带外的词元就此被放弃；而若它是在更早的内层迭代中跑出去的，它就会一直留在那里。PPO 并不把参数投影回信赖域，它只是停止推动。

常见陷阱: 截断不是 KL 约束

(6.13)既没有约束住 KL，也没有约束住逐词元的比值。它约束的是那些比值落在带外的词元的梯度贡献。一次大的梯度步、一个大多数词元都在带内的小批量，或者一开始比值就已经在带外，都可能造成任意大的策略移动。这正是为什么实用的 PPO 实现 还会加上显式的 KL 惩罚或基于实测 KL 的早停，也正是为什么这两套机制并不冗余。关于 PPO 所报告的性能中有多少取决于这类细节，参见Huang et al. (2024c)。

对语言模型而言，(6.13)引出了一个在控制问题中并不存在的问题：比值应在什么粒度上计算？上面用的词元级取法是 PPO 的标准做法。序列级取法 $\rho = \pi_\theta(y\mid x)/\pi_{\theta_{\text{old}}}(y\mid x)$ 被 GSPO 采用，其理由恰恰在于奖励本来就定义在序列层面。节 12.5会详细讨论这一点；此处只需注意，序列比值随长度呈指数变化的动态范围（节 2.1）意味着：同一个 ε 在两种情形下含义完全不同。

6.4 KL 正则化目标及其解

现在来求解第1章宣称为总纲模板的那个目标。固定一个提示 x，考虑

$$\max_{\pi(\cdot\mid x)\in\Delta(\mathcal{Y})} \ \mathbb{E}_{y\sim\pi}[r(x,y)] - \beta\mathbb{D}_{\text{KL}}(\pi(\cdot\mid x)\,\|\,\pi_{\text{ref}}(\cdot\mid x)). \tag{6.14}$$

定理 6.3 (KL 正则化下的最优策略)**.** (6.14)的唯一最大值点是吉布斯（指数倾斜）分布

$$\pi^\star(y\mid x) = \frac{1}{Z(x)}\,\pi_{\text{ref}}(y\mid x)\exp\!\Big(\frac{r(x,y)}{\beta}\Big), \qquad Z(x) = \sum_y \pi_{\text{ref}}(y\mid x)e^{r(x,y)/\beta}, \tag{6.15}$$

且(6.14)的最优值为 $\beta\log Z(x)$。

证明. 把目标写成 $\sum_y \pi(y)(r(y) - \beta\log\frac{\pi(y)}{\pi_{\text{ref}}(y)})$，并为约束 $\sum_y \pi(y) = 1$ 引入拉格朗日乘子 ν。对 $\pi(y)$ 求导得 $r(y) - \beta\log\frac{\pi(y)}{\pi_{\text{ref}}(y)} - \beta - \nu = 0$，故 $\pi(y) = \pi_{\text{ref}}(y)\exp((r(y) - \beta - \nu)/\beta)$，把常数吸收进 Z 后即为(6.15)。该目标关于 π 严格凹（负熵项严格凸），因此驻点就是唯一的最大值点。把(6.15)代回：$\mathbb{E}_{\pi^\star}[r] - \beta\mathbb{D}_{\text{KL}}(\pi^\star\,\|\,\pi_{\text{ref}}) = \mathbb{E}_{\pi^\star}[r] - \beta\,\mathbb{E}_{\pi^\star}[r/\beta - \log Z] = \beta\log Z$。 □

定理 6.3是本书中被复用最多的结论。它有四个推论：

1. 反解它就得到 DPO。 从(6.15)中解出奖励：

$$r(x,y) = \beta \log \frac{\pi^\star(y\,|\,x)}{\pi_{\text{ref}}(y\,|\,x)} + \beta \log Z(x). \tag{6.16}$$

难以计算的 $Z(x)$ 只依赖于 x，因此在同一提示下两条回复的奖励之差中它会被抵消——而 Bradley–Terry 似然恰好只依赖于这类差值。这一观察就是 DPO 的全部（节 8.1）。

2. $\beta \log Z$ 是一个软值函数。 最优值 $\beta \log Z(x) = \beta \log \mathbb{E}_{\pi_{\text{ref}}}[e^{r/\beta}]$ 是对所有回复取的 softmax（log-sum-exp），这正好还原了 KL 正则化控制与软值迭代之间的标准对偶关系。当 $\beta \to 0$ 时它趋于 $\max_y r$；当 $\beta \to \infty$ 时它趋于 $\mathbb{E}_{\pi_{\text{ref}}}[r]$。这是节 1.8中双词元计算的序列级类比。

3. 支撑集被保留。 对任意有限的 $\beta > 0$ 和任意有界奖励，只要 $\pi_{\text{ref}}(y\,|\,x) = 0$，就有 $\pi^\star(y\,|\,x) = 0$。KL 正则化目标在体质上就不可能凭空造出参考策略支撑集之外的行为——这正是「激发还是习得」这一问题的形式化种子（第14章）。

4. 它告诉我们 RLHF 究竟在逼近什么。 每一个 RLHF 算法都是 (6.15)的一个采样器或一个摊销器。PPO 通过随机优化把它摊销进策略权重；best-of-n 用拒绝采样来近似它（节 10.4.1）；DPO 则在偏好数据集上以闭式拟合它。在「奖励对 KL」平面上比较各种方法，比较的正是它们逼近同一条前沿的效率 (Gao et al., 2023a)。

6.5 KL 项的作用

值得把 KL 项承担的四项不同职责分开来看，因为实践者常常以为它只做了第一项。

职责 1：让奖励模型保持在分布内。 r_ϕ 是在某个特定分布上抽取的比较数据上拟合出来的。随着 π_θ 漂移，r_ϕ 的误差变大，而策略会优先找到误差为 正的区域——这正是奖励作弊的定义。KL 项给这种漂移标了价。Gao et al. (2023a) 发现金标准奖励可以很好地刻画为 $\sqrt{\mathbb{D}_{\text{KL}}}$ 的函数，先升后降，且奖励模型越大峰值越靠后：这是一个关于「能负担多少漂移」的明确且可用的预测。

职责 2：保护奖励没有度量到的能力。 一个针对数学正确性的奖励，对多语言流畅度、格式、置信度校准或拒答行为只字未提。没有锚点，这些能力就会自由退化；这就是所谓「对齐税」，而它在很大程度上是一个 KL 预算现象。

职责 3：间接维持熵。 $\mathbb{D}_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}}) = -\mathbb{H}[\pi_\theta] - \mathbb{E}_{\pi_\theta}[\log \pi_{\mathrm{ref}}]$。第一项说明，相对于纯粹的奖励最大化，惩罚反向 KL 天然包含了一份熵奖励：KL 锚点是一个比显式熵奖励更好驾驭的熵正则化手段（节 2.5），因为第二项同时会把概率质量保留在参考策略认为合理的词元上，而不是均匀地铺开。这是支持用 KL 而非熵奖励的最干净论证，而它并不常被明说。

职责 4：使问题适定。 没有正则化时，带有界奖励的(6.1)的最大值点是一个确定性策略；若奖励存在并列，则任意一个都是。KL 项使最优解唯一（定理 6.3），并使优化景观在策略空间中严格凹。

工程实践: β 的选择与监控

- 典型取值：针对学习得到的奖励模型做 RLHF 时，$\beta \in [10^{-3}, 10^{-1}]$，常用默认值为 0.01–0.05；而针对可靠验证器做 RLVR 时常取零，因为奖励在离开分布后不会退化，漂移的危险性较小。
- β 与奖励尺度不可分别辨识。请先固定奖励尺度（e.g., 在一个参考批次上标准化 r_ϕ），再去调 β，否则你就是在把同一个量调了两遍。
- 要监控每词元的实测 $\mathbb{D}_{\mathrm{KL}}$，而不只是惩罚项本身。RLHF 中一个有用的目标是每词元几倍于 10^{-3} 奈特；序列级 KL 达到 10–30 奈特时，奖励模型的过度优化通常开始显现。
- 自适应 β（实测 KL 超过目标就调高，否则调低）是 InstructGPT 时代的原始方案，至今依然有效；它把一个你猜不准的超参数，转换成一个你能设定的目标。
- 对于长思维链的 RLVR，2025—2026 年的许多配方直接令 $\beta = 0$ 并彻底丢掉 π_{ref}，理由是一旦策略走得远远超过参考策略，参考就变得无关紧要，此时 KL 项只会拖慢学习。这在基于验证器的训练中已成常规做法，而在基于奖励模型的训练中则明显是错的；参见节 12.8。

6.6 一个统一模板

现在我们可以写下本书其余部分反复实例化的那个对象。设一个批次由若干提示 x 组成，每个提示配有 $G \geq 1$ 条回复 $y^{(i)}$，长度为 $|y^{(i)}|$，由行为策略 $\pi_{\theta_{\mathrm{old}}}$ 采样得到。那么

Chapters 7, 12 and 17中几乎每一个算法都具有如下形式

$$\boxed{\mathcal{J}(\theta) = \mathop{\mathbb{E}}_{x}\Big[\frac{1}{\mathcal{N}}\sum_{i=1}^{G}\sum_{t=1}^{|y^{(i)}|}\omega_{i,t}(\theta)\,A_{i,t}\Big] - \beta\,\mathcal{R}(\theta)} \tag{6.17}$$

其中包含五个相互独立的设计选择：

1. **权重** $\omega_{i,t}$——重要性比值以何种方式进入。可选：$\log \pi_\theta$（纯 REINFORCE，不作修正）；词元比值 $\rho_{i,t}$；截断后的词元比值（PPO/GRPO）；序列比值 ρ_i 或其长度归一化形式 $\rho_i^{1/|y^{(i)}|}$（GSPO）；以及作为已 *detach* 的权重乘在 $\log \pi_\theta$ 上的截尾比值 $\min(\rho_{i,t}, C)$（CISPO，以及 第22章中的失配修正）。
2. **优势** $A_{i,t}$——信用如何分配。可选：基于评论家的 GAE；组均值或留一法；组均值再除以组内标准差；过程奖励模型给出的步级值；蒙特卡洛前缀值；逐轮次分解。
3. **归一化分母** $\mathcal{N}$——求和结果除以什么。可选：$G\cdot|y^{(i)}|$（逐序列的词元均值）；$\sum_j |y^{(j)}|$（全局词元均值）；$G \cdot L_{\max}$（常数）；G（序列均值）。这看起来只是记账问题，其实不是（节 12.2）。
4. **正则项** $\mathcal{R}$——用 k_1 或 k_3 估计的、相对 π_{ref} 的反向 KL；一个熵项；什么都不加；或者把 KL 折进奖励而不是损失。
5. **数据采集规则**——提示如何挑选，一个批次被复用多少个内层迭代，$\pi_{\theta_{\text{old}}}$ 允许有多陈旧，零方差的组是否重新采样。

表 6.2是本章中最有用的一件东西。当一个新算法出现时，有价值的问题不是「新的损失函数是什么」，而是「哪一个格子变了，它想修的是五种失效模式中的哪一种」。

6.7 这个估计器在代码里长什么样

从(6.17)到一份实现之间的距离很小，但错误恰恰就住在这些细节里。代码 6.1是本书中每一份实现的核心。

代码 6.1: 核心替代目标。注意：(i) `logp_old` 已被 detach，并且应尽可能与采样时走同一条数值路径（第22章）；(ii) 比值以 `exp(logp - logp_old)` 计算，绝不用两个概率相除；(iii) `mask` 会排除提示词元、填充词元，以及在智能体式场景中由环境写入的词元（第17章）。

```
def surrogate(logp, logp_old, adv, mask, eps_low=0.2, eps_high=0.2,
              normalizer="seq_mean"):
```

表 6.2: 把有名字的算法视为模板(6.17)的不同取值。表中「截断」指 $\min(\rho A, \mathrm{clip}\big(\rho, 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\big) A)$。各变体的细节，以及它们各自针对的病态现象，见Chapters 7 and 12。

方法	权重 ω	优势 A	归一化分母	正则
REINFORCE	$\log \pi_\theta$	r	G	—
RLOO	$\log \pi_\theta$	留一法	G	KL
ReMax	$\log \pi_\theta$	$r - r_{\text{greedy}}$	G	KL
PPO（RLHF）	截断的词元比值	基于评论家的 GAE	词元均值	KL
GRPO	截断的词元比值	$(r_i - \mu)/\sigma$	逐序列词元均值	KL（k_3）
Dr. GRPO	截断的词元比值	$r_i - \mu$	常数 $L_{\max}$	—
DAPO	截断，$\varepsilon_{\text{high}} > \varepsilon_{\text{low}}$	$(r_i - \mu)/\sigma$，并做过滤	全局词元均值	—
GSPO	截断的序列比值	$(r_i - \mu)/\sigma$	G	—
CISPO	截尾且 detach 的比值	$(r_i - \mu)/\sigma$	全局词元均值	—

```
    """logp, logp_old, adv, mask: [B, T] float/bool tensors."""
    log_ratio = (logp - logp_old) * mask          # 将不可训练的词元置零
    ratio = log_ratio.exp()

    unclipped = ratio * adv
    clipped = ratio.clamp(1.0 - eps_low, 1.0 + eps_high) * adv
    per_token = torch.min(unclipped, clipped) * mask

    if normalizer == "seq_mean":                  # GRPO: 每条序列的权重为 1/G
        per_seq = per_token.sum(-1) / mask.sum(-1).clamp(min=1)
        obj = per_seq.mean()
    elif normalizer == "token_mean":              # DAPO: 每个词元权重相同
        obj = per_token.sum() / mask.sum().clamp(min=1)
    elif normalizer == "constant":                # Dr. GRPO: 固定分母
        obj = per_token.sum() / (mask.size(0) * MAX_LEN)
    return -obj                                   # 损失 = 目标的相反数
```

常见陷阱: 四个能画出「看起来很合理」曲线的 bug

(*a*) 忘了 *detach* `logp_old`。比值会恒等于 1，而且梯度会同时沿两项回传；损失退化成一个加权似然，截断从此永不触发。症状：截断比例永远恰好为零。(*b*) 在求和之后才做掩码。填充词元在被置零之前就已经参与了求和，使损失被一个依赖于批次的常数缩放。症状：结果取决于批次的组成方式。(*c*) 优势广播时错位一个

算法 6.1 面向语言模型的通用同策略策略优化。Chapters 7 and 12 中的每一个算法，都是通过实例化第 4、6、8、9 行得到的。

输入： 初始策略 π_θ，参考策略 π_{ref}，奖励 r，提示集 $\mathcal{D}$，组大小 G，内层迭代数 E

for 迭代 $k = 1, 2, \ldots$ **do**
  $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$；将权重同步到推理引擎
  采样一批提示 $\mathcal{B} \subset \mathcal{D}$，并对每个 $x \in \mathcal{B}$ 采样 $\{y^{(i)}\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$
  （选择 5）可选地过滤 $\mathcal{B}$：丢弃那些组内奖励方差为零的提示；重新采样以补满批次
  对所有 i 计算 $r(x, y^{(i)})$ ▷ 奖励模型、验证器或环境
  （选择 2）构造优势 $A_{i,t}$ ▷ 评论家/GAE、组相对，或过程级
  for $e = 1$ **to** E **do**
    for 词元小批量 m **do**
      （选择 1、3）计算 $\omega_{i,t}(\theta)$ 与替代目标 $\frac{1}{N}\sum \omega_{i,t} A_{i,t}$
      （选择 4）减去 $\beta \mathcal{R}(\theta)$
      做梯度上升；截断全局梯度范数；可选地更新评论家
    end for
    if 实测 $\mathbb{D}_{\text{KL}}\left(\pi_\theta \,\middle\|\, \pi_{\theta_{\text{old}}}\right) > \delta_{\max}$ **then**
      break ▷ 早停；信赖域截断并不能强制这一点
    end if
  end for
end for
return π_θ

位置。回复词元相对于 logits 通常有一位偏移；错移一个词元，就等于把每个词元的信用记到了它邻居头上。训练仍然能学，只是慢一些——这正是它能通过代码评审的原因。(d) 在错误的轴上归一化优势。在整个批次上做标准化、而不是在每个提示的组内做，会悄无声息地把一个组相对方法变成批相对方法，从而把互不相干的提示耦合在一起（节 12.3）。

6.8 为什么不用基于值或基于模型的方法?

为完整起见说明一下：Q 学习及其后裔在这里用得不多，而原因很有启发性。在 10^4 个位置上、每个位置都要覆盖 $|\mathcal{U}| \approx 10^5$ 个动作的动作值函数是一个庞大的对象；而基于值的控制所需要的贪心 $\arg\max_a Q(s,a)$，恰恰就是会摧毁我们需要保留的流畅性的那个操作。更根本地说，预训练模型给了我们一个极其优秀的策略先验，却完全没有给出值先验，因此把策略当作主要对象的方法天然领先一大截。话虽如此，Rafailov et al. (2024b) 表明，用 DPO 训练出的策略隐式地定义了一个 Q 函数，而词元级的值视角对信用分配一直很有成效（第13章）。基于模型的强化学习在通常意义下在这里是空洞的——动力学已知且平凡——但它会以变形的方式在智能体式场景中重新出现：那里环境的动力学是未知的，学习世界模型于是重新变得有意义（节 19.6）。

本章小结

- 策略梯度(6.2)由似然比恒等式所决定；全部算法自由度都落在优势估计器与信赖域上。
- 朴素估计器把同一份信用分配给轨迹中的每一个词元。它是无偏的，但噪声极大，二阶矩随回复长度线性增长——这正是长思维链强化学习需要比短回复 RLHF 更好的估计器的原因。
- 任何依赖于提示的基线都保持无偏。学习得到的评论家提供逐状态的信用，代价是 2× 显存和一个困难的回归问题；组基线没有偏差也不占显存，但当组内所有样本得分相同时会给出恰好为零的梯度。
- 当 $\gamma = \lambda = 1$ 且只有终局奖励时，GAE 坍缩为 $r - V_\psi(s_t)$：评论家的唯一作用就是充当依赖于状态的基线。在这种场景下做折扣，等于加了一个隐藏的长度惩罚。
- 信赖域的正当性来自性能差界(6.12)。PPO 的截断是它的一个廉价一阶替代：把带外的梯度置零，但并不约束 KL——这就是显式 KL 惩罚与早停依然必要的原因。
- KL 正则化下的最优解是吉布斯倾斜 $\pi^\star \propto \pi_{\text{ref}} e^{r/\beta}$，最优值为 $\beta \log Z$。反解它得到 DPO；它保留支撑集；并且它使「奖励对 KL」成为比较所有方法的天然平面。
- KL 项承担着四项不同的职责：让奖励模型保持在分布内、保护未被度量的能力、充当比熵奖励更好驾驭的熵正则化手段，以及使最优解唯一。

- 模板(6.17)及其五个选择——权重、优势、归一化分母、正则项、数据规则——可以生成表 6.2中每一个有名字的方法。读新论文时，就问哪一个格子变了。

延伸阅读

信赖域论证的原始形态见Schulman et al. (2015)；界(6.12)及其证明值得认真推演一遍，因为后续每一种方法都是对它的一次「抄近道」式近似，而知道抄掉了哪个角，正是你预判失效模式的方式。Schulman et al. (2017) 篇幅很短，应当与Engstrom et al. (2020) 和 Huang et al. (2024c) 一起读；这两篇合起来确立了一点：PPO 相对更简单方法所报告的优势，大部分来自实现细节而非截断目标本身。Ahmadian et al. (2024) 是让整个领域重新审视语言模型是否真的需要评论家的那篇论文，而它的论证——语言模型的 MDP 并不具备 PPO 所针对的那些性质——与 节 2.3中给出的是同一个论证，只是后者更形式化。关于定理 6.3所处的 KL 正则化控制文献，Ziebart et al. (2010) 与 Levine (2018) 给出了最大熵强化学习的框架，它让 $\beta \log Z$ 这个软值函数看起来是必然的，而不只是凑巧方便。GAE 见Schulman et al. (2016)；读它时请记住语言模型场景会让它坍缩，这样你才能分辨一篇论文用 $\lambda < 1$ 究竟是出于真实理由，还是仅仅出于习惯。

习题

6.1（**奖励平移改变了一切，也什么都没改变。**）证明把 r 换成 $r + c$（c 为常数）不改变(6.2)的期望。然后对成功概率为 p 的二值奖励，把单样本估计器的方差算成 c 的函数，并求出使方差最小的 c。解释它与 定理 2.2的联系，以及为什么那些「忘了加基线」的实践者常常报告说减去 0.5 就把训练救回来了。

6.2（**GAE 的坍缩，两个方向。**）验证在只有终局奖励时 $A_t^{\mathrm{GAE}(1,1)} = r - V_\psi(s_t)$ 这一推导。然后改用逐词元的 KL 惩罚重做一遍：对 $t < T$ 取 $r_t = -\beta \log \frac{\pi_\theta}{\pi_{\mathrm{ref}}}$，并取 $r_T = r - \beta \log \frac{\pi_\theta}{\pi_{\mathrm{ref}}}$，说明此时 λ 确实变得重要，并给出这种情形下 $\lambda < 1$ 的一种解释。

6.3（**截断并不能约束 KL。**）构造一个双动作的例子以及一串 PPO 更新，使得在每次更新的当下每个词元的比值都落在 $[1-\varepsilon, 1+\varepsilon]$ 之内，但经过 E 个内层迭代之后，相对初始策略的总 KL 随 $E \to \infty$ 无界增长。然后说出能够阻止这一现象的最小额外机制。

6.4（**把惩罚换成约束后的吉布斯最优解。**）把(6.14)中的 KL 由惩罚项改为硬约束 $\mathbb{D}_{\mathrm{KL}}(\pi \,\|\, \pi_{\mathrm{ref}}) \le \delta$ 后重新求解。证明解具有相同的函数形式，只是 β 由 δ 隐式确定，并推导出关系式 $\delta(\beta)$。解释自适应 β 的方案为何就是在数值求解这个方程。

6.5（**best-of-n 与吉布斯最优解。**）针对同一奖励做 best-of-n 会诱导出某个分布 π_{BoN}。设奖励在 π_{ref} 下具有连续分布，请用 r 在 π_{ref} 下的累积分布函数显式地推导 π_{BoN}，并把它与(6.15)作比较。二者有可能相等吗？（本题在节 10.4.1中完成；请先自行尝试。）

6.6（**序列比值与词元比值。**）设某策略在 T 个词元上的对数概率相对 $\pi_{\theta_{\mathrm{old}}}$ 各差 ϵ，且方向一致。计算词元级比值与序列级比值。对 $\varepsilon = 0.2$ 的截断，分别在 $T = 100$ 与 $T = 10{,}000$ 下求出使每种机制被触发的逐词元 ϵ。据此推断：要让序列级截断具有可比的紧度，ε 必须取多少；并评论在长度可变的一个批次上，单一常数是否可能奏效。

6.7（**RLOO 等于缩放后的组均值。**）验证(6.9)中的恒等式。然后判断当组优势像 GRPO 那样再额外除以组内标准差之后，这一等价关系是否仍然成立。（请当心：它不成立，而原因很有启发性。）

6.8 ⋆ （**替代目标什么时候是精确的?**）定理 6.2是通过把 d^{π} 替换成 $d^{\pi'}$ 而变成替代目标(6.11)的。在动力学确定的语言模型 MDP 中，精确刻画这一替换在什么条件下是无损的。然后解释：尽管这里可用的结构性假设如此之强，为何没有任何已发表的方法去利用这个精确形式——利用它要付出什么代价?

6.9 † （**对模板做消融。**）用一个小模型和一个可验证任务，把 (6.17)实现为五个选择皆可配置的开关。在算力对齐的条件下运行 表 6.2中不需要评论家的九行，并报告奖励、熵、平均长度、截断比例、pass@1 与 pass@8。你应当会发现，仅仅是归一化分母就能让最终的 pass@1 出现可测量的变化；若没有，说明你的任务太简单了。请保留这套实验框架——第12章会要求你扩展它。

第 7 章

RLHF 的工程实践

核心思想: 目标函数只有三行，系统却有三千行，而结果来自系统

第6章给出了 PPO 的数学。本章给出的是实物：常驻在 GPU 显存中的四份模型副本，一份把 KL 惩罚折进回报而构造出的逐词元奖励，施加在奖励与优势之上的四层归一化，以及大约十五个实现细节——其中每一个都可能悄无声息地吃掉比算法选择更多的性能。两项独立的复现研究得出结论：PPO 相对更简单基线所报告的优势，大部分存在于这些细节之中，而不在截断替代目标里。这个结论不应被读作一次「祛魅」，而应被读作一份规范：它明确列出了你实际必须做对的那些事。

第6章以一个模板(6.17)和一个通用循环 算法 6.1收尾。把它们针对「基于学习得到的奖励模型、从人类反馈中做强化学习」这一情形实例化，就得到Ziegler et al. (2019)、Stiennon et al. (2020) 和Ouyang et al. (2022) 所确立、并被 2020 至 2024 年间几乎每一套对齐系统所实现的那条流水线。我们不再重新推导截断替代目标；(6.13)照旧成立。我们把它当作既定事实，转而追问：它周围必须配上些什么。

本章按照「东西通常以什么顺序坏掉」来组织，读起来像一份调试手册。节 7.1先算显存账，因为它决定了你到底跑不跑得起来。节 7.2定夺 KL 惩罚该放在哪里——这是最具后果的结构性决策，也决定了广义优势估计究竟有没有在做事。节 7.3 and 7.4 讲各类归一化，那是目标函数被悄悄改掉的藏身之处。节 7.5罗列关于实现细节的那批文献。节 7.6说明该记录什么、每个指标又能查出什么问题。节 7.7给出具体数值。节 7.8讲外层循环，而 节 7.9则老实交代这套机制在 2026 年还有什么用。

7.1 标准流水线及其代价

图 7.1是数据流图，请把它当作一次梯度更新来读。

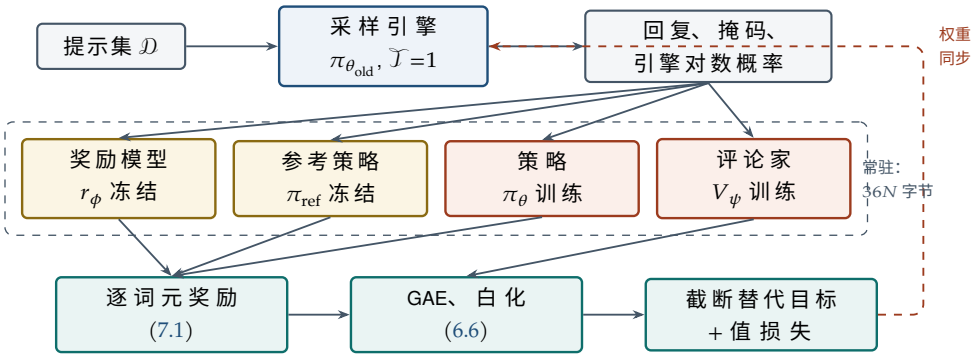


图 7.1: 一次 PPO-RLHF 梯度更新。常驻四份模型副本：两份冻结（奖励模型与参考策略，只需前向传播）和两份训练（策略与评论家，各自还带着梯度和优化器状态）。除非采样引擎与训练共置并分时复用，否则它还会持有策略权重的第五份副本。每一条指向逐词元奖励节点的箭头都是一个可能的错位一位的 bug 来源；节 7.5逐一列举了它们。

四份常驻模型是这套方法的结构性特征，也是它后来被取代的原因。两份是冻结的，只需付出权重的代价；两份在训练，要付出权重、梯度和优化器状态的代价。若权重与梯度都用 BF16，而 AdamW 保存 FP32 主权重和两个 FP32 动量，那么一个受训模型每个参数需要 $2+2+4+4+4=16$ 字节，一个冻结模型需要 2 字节。合计：每个参数 $16+16+2+2=36$ 字节的模型状态——这还不算激活值、KV 缓存，以及推理引擎自己那份权重副本。

表 7.1把节 6.2.4中的权衡量化了。评论家占每参数 36 字节中的 16 字节，也就是模型状态的 44%；在 70B 规模上，它是「需要约 64 张加速卡」与「需要约 32 张」之间的差别。真正促使该领域转向组基线的近因就是这个比例，而不是任何统计学论证：在最大的那一项开销上白得一个二倍的节省，代价只是 $G-1$ 次额外生成——而生成受限于显存带宽、模型又本来就常驻，因此很便宜。

三点实践上的补充说明。第一，参考模型和奖励模型不一定要常驻：它们的输出每批只需要一次，因此可以在单独的一遍前向中算出对数概率与分数，缓存成张量，然后把权重卸载掉——这是一个标准技巧，把 $4N$ 字节的永久常驻换成了一份带宽开销。第二，在共置（混合引擎）设计中，推理引擎的那份副本会消失：引擎通过睡眠模式在两个阶段之间释放显存，代价是生成与训练必须串行。第三，表 7.1中 KV 缓存那几行，才是真正决定可达采样并发度、进而决定墙钟时间的数字；它们随层数而非参数量增长，这也是为什么又深又窄的模型采样起来格外昂贵。

表 7.1: 两种规模下 PPO-RLHF 的常驻显存，假设权重与梯度为 BF16、AdamW 保存 FP32 主权重与动量、评论家和奖励模型与策略同样大小，并采用分组查询注意力：8 个维度为 128 的键/值头，32 层（7B）与 80 层（70B）。在全量重计算下激活值大约再增加 5%–15%，此处略去。最后三行正是促使该领域转向无评论家方法的那组对比。

组件	精度	字节/参数	7B（GB）	70B（GB）
策略 π_θ：权重、梯度、AdamW	BF16 / FP32	16	112	1120
评论家 V_ψ：权重、梯度、AdamW	BF16 / FP32	16	112	1120
参考策略 π_{ref}：仅权重	BF16	2	14	140
奖励模型 r_ϕ：仅权重	BF16	2	14	140
模型状态小计		**36**	**252**	**2520**
推理引擎的权重副本	BF16	2	14	140
KV 缓存，256 条序列 × 2048 词元	BF16	—	69	172
总计			**335**	**2832**
同上，去掉评论家（RLOO/GRPO 对 r_ϕ）		20	223	1712
同上，去掉评论家与奖励模型（RLVR）		18	209	1572

常见陷阱: 评论家不只是显存，它还是第二个训练问题

人们很容易把表 7.1读成「评论家只花显存、别的不花」。它还要花掉一个学习率、一种初始化、一份预热调度、一个截断参数、一个损失系数，以及一个回归目标——而正如节 7.2将说明的，这个目标包含 KL 惩罚，因而依赖长度、并与 β 耦合。以上每一样都是一条能让训练失败、且失败起来看着像策略出了问题的路径。当一次 PPO-RLHF 训练行为异常时，评论家的可解释方差是第一个该看的数字，不是最后一个。

7.2 KL 放进奖励，还是放进损失

节 6.5确立了 KL 项是为了什么。本节讨论的是它放在哪里——这是一个独立的问题，而答案并不显然。

7.2.1 整形后的逐词元奖励

InstructGPT 这一脉把惩罚折进奖励。记 $\ell_t = \log \pi_\theta(y_t \mid x, y_{<t}) - \log \pi_{\text{ref}}(y_t \mid x, y_{<t})$ 为(2.11)中逐词元的 k_1 估计量，则强化学习算法实际使用的奖励为

$$\hat{r}_t = \begin{cases} -\beta\, \ell_t, & 1 \le t < T, \\ -\beta\, \ell_T + r_\phi(x, y), & t = T, \end{cases} \tag{7.1}$$

于是轨迹回报为 $\sum_t \hat{r}_t = r_\phi(x,y) - \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}$，正是 KL 正则化目标 (6.14)的一个无偏单样本估计，而该目标的最优解由 定理 6.3给出。这里没有做任何近似：(7.1)就是那个目标本身，只是被摊分到了各个词元上。

摊分到词元上正是关键所在。节 6.2.2已经说明，在只有终局奖励且 $\gamma = \lambda = 1$ 时，广义优势估计坍缩为 $r - V_\psi(s_t)$，λ 无事可做。而有了(7.1)，奖励就变稠密了——每个词元都带着一个非零的 $\hat{r}_t$——λ 于是重新获得真实的角色。这一点值得明确写出来，因为它正是 InstructGPT 时代的默认值 $\lambda = 0.95$ 在此场景下站得住脚、而在第11章的终局奖励场景下站不住脚的原因。

推导: KL 进入奖励之后 λ 做了什么

取 $\gamma = 1$，把(7.1)代入 TD 残差 $\delta_t = \hat{r}_t + V_\psi(s_{t+1}) - V_\psi(s_t)$。展开(6.6)并把终局奖励与稠密惩罚分离，得

$$A_t^{\text{GAE}(1,\lambda)} = \underbrace{\lambda^{T-t} r_\phi(x,y)}_{\text{终局分数，被衰减}} - \beta \underbrace{\sum_{l=0}^{T-t} \lambda^l \ell_{t+l}}_{\text{未来的 KL 账单，被衰减}} + \underbrace{(1-\lambda) \sum_{l=0}^{T-t-1} \lambda^l V_\psi(s_{t+l+1}) - V_\psi(s_t)}_{\text{评论家自举}}. \tag{7.2}$$

在 $\lambda = 1$ 处自举项消失，我们回到 $A_t = r_\phi - \beta \sum_{l \ge 0} \ell_{t+l} - V_\psi(s_t)$：每个词元都被以蒙特卡洛方式记上整条轨迹全部剩余的 KL 账单，其方差随剩余词元数线性增长。在 $\lambda = 0$ 处，$A_t = -\beta \ell_t + V_\psi(s_{t+1}) - V_\psi(s_t)$：每个词元只被记上它自己那份 KL 代价，而其余一切——包括 T 处的奖励——都只能经由评论家传递到它。

因此 λ 设定的是「一个词元要为它尚未造成的散度负责多长的时域」，有效窗口为 $1/(1-\lambda)$：$\lambda = 0.95$ 时约 20 个词元，$\lambda = 0.99$ 时约 100 个。关键在于，无论 λ 取

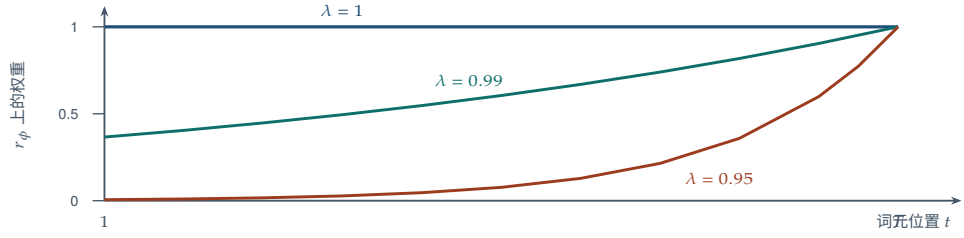


图 7.2: 在 $T = 100$ 时，(7.2)中乘在奖励模型分数上的系数 λ^{T-t}。这些是 λ^{T-t} 的精确取值，不是实测数据。在 $\lambda = 0.95$ 时，距离末尾超过约 60 个位置的词元直接收到的终局分数不足 5%，几乎完全依赖评论家；而在 $\lambda = 1$ 时每个词元都完整收到它。相比之下，逐词元的 KL 代价始终以权重 1 进入——这正是 $\lambda < 1$ 在这里是降方差、而在第11章 的终局奖励场景中是删信号的原因。

> 何值，每个词元自己的 KL 代价都在 $l = 0$ 处以权重 1 进入。正是这种不对称性，使得 $\lambda < 1$ 在这里合理、而在终局奖励情形下不合理：衰减能在不抹掉信号的前提下，从一个有噪声的稠密信号中削去方差；而在只有终局奖励时，$\lambda < 1$ 衰减掉的唯一东西就是奖励本身。

图 7.2画出了这种衰减。实践上的解读是：λ 决定了从奖励模型分数到某个靠前词元的信用通路中，有多少经由观测到的回报、有多少经由评论家。如果评论家的可解释方差高，调小 λ 就是在降方差；如果它低，调小 λ 就是在注入偏差。这是一个由诊断量驱动的选择，而不是一个常数。

7.2.2 没人明说的微妙之处：评论家必须建模 KL

由于 KL 惩罚是奖励的一部分，它也就是回报的一部分，因此评论家的回归目标 (6.5)是由整形后的奖励算出的 $\hat{G}_t = A_t + V_\psi(s_t)$。于是 $V_\psi(s_t)$ 并不是「这条回复会有多好」的估计，而是「奖励模型分数的期望，减去从此处到片段结束的期望剩余散度代价」的估计。由此有三个推论，而三个都是真实 bug 的来源。

评论家必须隐式地学一个长度模型。 还剩 5000 个词元的前缀，其期望 KL 账单比只剩 50 个的大，因此 V_ψ 必须把剩余长度编码进去。这给本已困难的回归又加了一重负担，并使评论家的误差与长度相关，进而转化为与长度相关的优势偏差——而这恰恰是长度受控评测所要检测的那种偏差 (Dubois et al., 2024)。

改动 β 会让评论家失效。 值函数的输出尺度正比于生成其目标时所用的 β。一个把 β 翻倍的自适应 β 控制器，等于刚刚让评论家在惩罚分量上错了近两倍，而评论家需要几十步才能恢复。这就是那条常见观察背后的机制：带评论家时自适应 KL 控制器比不带时更不稳定；对应的缓解办法是让 β 缓慢移动（用增益 ≈ 0.1 的比例控制器，并对单步变化做截断），而不是对每一个批次都作出反应。

如果你只整形了奖励却没整形回报，惩罚几乎什么都不做。 一个反复出现的 bug 是：整形后的奖励喂给了 GAE，但评论家的目标却是用原始奖励模型分数算的。于是惩罚出现在 A_t 里却没出现在 V_ψ 里，二者系统性地不一致，回归把 V_ψ 推向未加惩罚的回报，而优势则带上了整份 KL 账单大小的偏差。症状是：评论家的可解释方差很低且不见好转，同时 KL 曲线在名义上很大的 β 之下依然一路上涨。

7.2.3 KL 放进损失

另一种做法——自 GRPO (Shao et al., 2024) 以来已成标准——是不动奖励，而在损失上加一个可微的惩罚项：

$$\mathcal{L}(\theta) \;=\; -\mathcal{J}^{\mathrm{PPO}}(\theta) \;+\; \beta\, \mathbb{E}[k_3(\pi_\theta(y_t\,|\cdot), \pi_{\mathrm{ref}}(y_t\,|\cdot))], \tag{7.3}$$

其中 k_3 取自(2.13)。这并不是同一个算法，而且差别是结构性的，不是表面的。

最尖锐的差别在于你拿到的是哪一个梯度。在(7.1)中，惩罚以奖励系数的身份进入，因此它对梯度的贡献是一个得分函数（REINFORCE）项：$\mathbb{E}[\nabla_\theta \log \pi_\theta \cdot (-\beta \sum_l \ell_{t+l})]$，这是 $\mathbb{E}_{\pi_\theta}[\mathbb{D}_{\mathrm{KL}}]$ 的梯度，其中包含了因采样分布依赖于 θ 而产生的那一项。而在(7.3)中，惩罚是在样本固定的前提下求导的，只给出通路（pathwise）那一部分。具体地说：KL 放进奖励能够表达「不要进入那些会迫使你偏离的状态」，因为惩罚会经由 GAE 反向传播到做出该分支选择的词元上。KL 放进损失做不到这一点；它只为每个被采样的词元自身的散度记账，仅此而已。这是支持把 KL 放进奖励的最有力论证，而它几乎从未被明说。

反过来说：得分函数估计量带有人们意料之中的方差；惩罚现在会流经施加在奖励与优势之上的每一层归一化（节 7.3 and 7.4），因而被一些你并不打算引入的量重新缩放；而且它需要一个值函数来做传播——这意味着对无评论家的方法来说，它在任何有用的形式下都不可得。在使用组基线时，优势沿整条序列是常数，于是把 KL 折进奖励只不过是往一个标量优势上加了整条序列的散度总量：这样做合法，但它丢掉了逐词元的归因，而那本来正是这么做的全部理由。

表 7.2: KL 惩罚的两种放置方式，差别在于它们产生的梯度，而不仅仅在记账方式上。表中「前瞻」指惩罚能够抑制进入高散度区域，而不只是为已经发生的散度收费。

	KL 放进奖励 (7.1)	**KL 放进损失** (7.3)
使用的估计量	k_1（必须无偏且带符号）	k_3（非负，低方差）
得到的梯度	得分函数；完整的 $\nabla \mathbb{E}_{\pi_\theta}[\mathbb{D}_{\mathrm{KL}}]$	仅通路部分
前瞻	有，经由 GAE	无
方差	较高；随剩余长度增长	低
是否需要评论家	需要，用于传播	不需要
是否进入评论家的目标	是——V_ψ 必须建模 KL	否
能否经受奖励白化	不能；会被 $\sigma_{\mathcal{B}}$ 重新缩放	能；走独立通路
β 可解释性	只有固定奖励尺度之后才可解释	直接以奈特为单位可解释
是否复活 λ	是（节 7.2.1）	否
使用者	InstructGPT 一脉、`trl`、`open-instruct`	GRPO 家族、2025—2026 年的多数配方

核心思想: 该选哪一种

如果你有评论家，就把 KL 放进奖励。这是唯一能给出前瞻项的放置方式，也正是它让 λ 有意义，而传播它所需的机制本来就已经存在。接受随之而来的后果：β 与奖励尺度纠缠在一起，评论家现在还要建模一份依赖长度的散度账单。如果你没有评论家，就把它放进损失，用 k_3，并且明白你买到的是一个更便宜、更稳定、但无法惩罚轨迹选择的近似。不存在任何一种配置能让两者同时正确，而那些两处都加了的实现——它们确实存在，通常是历史沉积的产物——是在重复计算。

7.3 奖励的后处理

奖励模型分数是一个单位任意的数。第5章将确立：Bradley–Terry 似然只能把 r_ϕ 辨识到相差一个关于提示的可加函数，因此 $r_\phi(x, y)$ 的绝对水平和尺度都没有意义，只有同一提示内部的差值才有意义。然而 PPO-RLHF 却消费绝对分数，并用一个同样拟合在这套任意单位上的评论家去抵消它。本节的一切都是这一错配的后果。

7.3.1 白化，以及它对 β 做了什么

标准变换（Huang et al. (2024c) 有记载，各主流库也都实现了）是

$$\tilde{r} = \frac{r_\phi - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}} + \epsilon}, \tag{7.4}$$

其中 $\mu_{\mathcal{B}}, \sigma_{\mathcal{B}}$ 既可以是批次统计量，也可以是滑动估计。这里同时发生着两件不同的事，应当分开评判。

减去 $\mu_{\mathcal{B}}$ 几乎无害。按定理 2.2的标准它不是一个合法基线——批次均值是所采样批次的函数，而不是提示的函数——但在对批次取期望的意义下它是一个常数，因此引入的偏差是 $O(1/|\mathcal{B}|)$，换来的是节 6.2论证过的、永远值得拥有的方差削减。

除以 $\sigma_{\mathcal{B}}$ 就不是无害的了，原因是一行就能写完、而实践者却往往要吃过苦头才能重新发现的计算。最大化 $\mathbb{E}[\tilde{r}] - \beta\mathbb{D}_{\mathrm{KL}}$，两边同乘 $\sigma_{\mathcal{B}}$ 之后，等价于最大化 $\mathbb{E}[r_\phi] - (\beta\sigma_{\mathcal{B}})\mathbb{D}_{\mathrm{KL}}$。有效的 KL 系数是 $\beta\sigma_{\mathcal{B}}$，而不是 β。而 $\sigma_{\mathcal{B}}$ 并非常数：随着策略集中到高分回复上，一个批次内奖励模型分数的离散程度会收缩。于是奖励白化恰恰在策略漂移得最远的时候放松了 KL 锚定——这是一个正反馈回路，表现为 KL 曲线先平坦数百步、然后突然加速。

常见陷阱: 奖励白化会在训练途中悄悄重调 β

问题不在于白化是错的，而在于 β 不再是你设定的那个意思。三种缓解办法，纪律要求依次递增。(*i*) 使用长时域的滑动统计量（动量取 0.99 或更高），使 $\sigma_{\mathcal{B}}$ 缓慢变化，β 至少在分段意义上还有意义。(*ii*) 用 SFT 策略生成的一个批次一次性确定统计量，此后永不更新；这样奖励尺度就成了奖励模型的属性，而不是当前策略的属性。(*iii*) 干脆不对奖励做白化——只白化优势（节 7.4），那里同一个除数同时也会缩放 KL 贡献，比例关系得以保持。无论采用哪种，都请同时记录 $\sigma_{\mathcal{B}}$ 与乘积 $\beta\sigma_{\mathcal{B}}$；后者才是你真正的超参数。还要注意顺序也很重要：像各类库那样，先白化奖励模型分数、再加上逐词元 KL，就会产生上述效应；而白化整个整形后的奖励则保持了比例关系，代价是让评论家的目标尺度发生漂移。

7.3.2 截断、EOS，以及分数落在哪里

截断。 奖励模型分数是重尾的，而尾部正是退化输出的栖息地：一条只有单个词元的回复，或者一条反复重复某个短语的回复，偶尔会拿到偏离均值好几个标准差的分数。这样一个样本就会扭曲整个批次的 $\mu_{\mathcal{B}}$ 与 $\sigma_{\mathcal{B}}$，并在白化之后扭曲其余每一个样本的优势。把 r_ϕ 截断到一个固定区间（常用白化单位下的 ± 5 或 ± 10，或者 SFT 策略分数分

布的经验第 1–99 百分位）不花什么代价，却能消除这种耦合。请记录被截断的比例；若它超过百分之一二，那你遇到的是生成层面的病态，而不是尺度问题。

EOS 与长度问题。 如果一条回复在生成上限处被截断、而奖励模型照样给它打分，那么撞上长度上限就是免费的。由于学习得到的奖励模型对更长的回复有充分记录在案的偏好 (Singhal et al., 2024; Chen et al., 2024d)，这比免费还糟：它是有利可图的。结果就是长度单调增长直到触顶，这是针对学习得到的奖励模型做 RLHF 时最常见的病态。有两种标准补救办法，而它们并不可互换。第一，给任何没有输出 *<eos>* 的回复赋一个固定的低奖励——通常取分数分布的一个低百分位，或者取一个比任何合法回复都更差的常数。这是 RLHF 的默认做法，而且在这里是对的，因为长度偏差正是你要对抗的东西。第二，把被截断的样本从损失中掩掉，这是 RLVR 的默认做法 (Yu et al., 2025c)，其理由是一条被截断的回复是「证据缺失」而不是「失败的证据」。当奖励来自验证器时这个理由成立，当奖励来自有长度偏好的偏好模型时则不成立，因为掩码会把唯一一股抵抗触顶的压力也一并移除。

落点。 终局分数必须加在最后一个真实词元的下标处。训练时是右侧填充的，因此张量的最后一个位置是一个填充词元，而它的优势会被掩掉。把分数加在那里，意味着奖励模型的信号根本到不了损失，唯一在训练策略的东西就只剩 KL 惩罚——其表现是一条平坦的奖励曲线，以及一个缓慢退回参考策略的策略。这个 bug 常见到值得在代码里写一条断言，而不是写一条注释。

代码 7.1: 构造整形后的逐词元奖励(7.1)。注意顺序：先截断原始分数，再覆盖被截断的回复，然后才把分数放到最后一个未被掩掉的下标处。返回的 `kl` 单独保留，因为它必须被记录，而且评论家的目标依赖于它。

```python
def shaped_rewards(rm_score, logp, logp_ref, mask, beta,
                   ended_with_eos, trunc_reward=-4.0, clip=5.0):
    """rm_score:[B]   logp, logp_ref, mask:[B,T]   ->   reward:[B,T]"""
    # 逐词元的 k1 估计量。它是一项*代价*，因此以负号进入。
    kl = (logp - logp_ref) * mask
    reward = -beta * kl

    # 该分数未经校准且重尾；在它能够影响用于白化的批次统计量之前，
    # 先把它界住。
    score = rm_score.clamp(-clip, clip)
    # 从未输出 <eos> 的回复不算回复。没有这一行，有长度偏好的奖励
    # 模型会让撞上长度上限变得严格有利可图。
```

```
    score = torch.where(ended_with_eos, score,
                        torch.full_like(score, trunc_reward))

    last = mask.sum(-1).long() - 1              # 最后一个真实词元的下标
    assert (last >= 0).all()
    reward[torch.arange(reward.size(0)), last] += score
    return reward, kl
```

7.4 优势与回报的处理

在整形后的奖励与梯度之间，还夹着另外四层归一化。每一层单看都站得住脚；很少有人把它们当作一个整体来推敲。

优势白化。 在整个批次上把 A_t 替换为 $(A_t - \text{mean}(A))/(\text{std}(A) + \epsilon)$ 几乎是通用做法，也是最常被认为「让 PPO-RLHF 真正跑通」的那个细节 (Xu et al., 2024d)。支持它的理由是：它固定了替代目标的尺度，使得 ε、学习率和梯度范数截断阈值能够在奖励方差不同的各次训练之间迁移。

反对它的理由比通常得到的重视更强，且有四个部分。第一，在整个批次上减去均值，会把目标变成「比这个批次里的平均回复更好」，而不是「比针对这道提示的平均回复更好」，这就耦合了互不相干的提示，并把批次的难度构成带进了每一个优势里——这与节 12.3对组相对方法中批级归一化提出的反对意见是同一个。第二，除以标准差是一个依赖数据的学习率：一个所有样本得分都相近的批次，其噪声会被放大到单位方差。第三，Adam 本来就会归一化梯度幅度（节 2.9），因此所声称的尺度收益在很大程度上是冗余的——除了瞬态之外，也除了 Adam 无法抵消的那部分，即批次内部样本之间的相对权重。第四点也是最有用的一点：在评论家拟合良好时，优势本就应当接近零均值，因此一个较大的优势均值是你手头最好的评论家偏差诊断量，而白化把它删掉了。

多数实现最终收敛到的折中做法是：只在整个采样批次的层面做白化，绝不在梯度小批量的层面做，并同时记录未白化的均值与标准差。小批量层面的白化会让结果取决于批次是如何切分的，而没有人希望算法具有这种性质。

GAE 参数。 $\gamma = 1$ 恒成立，理由见节 2.3：对一个终局奖励的片段做折扣，是一个隐藏的长度惩罚。λ 之所以是一个真实的选择，仅仅因为奖励被整形过（节 7.2.1）；0.95 是 InstructGPT 一脉的默认值，而如果你已经把 KL 从奖励中移走，那么 1.0 才是正确的。

值截断。 评论家的损失通常写作

$$\mathcal{L}_V(\psi) = \tfrac{1}{2}\,\mathbb{E}\Big[\max\big((V_\psi(s_t) - \hat{G}_t)^2,\ (V_\psi^{\text{clip}}(s_t) - \hat{G}_t)^2\big)\Big],$$
$$V_\psi^{\text{clip}}(s_t) = V_{\text{old}}(s_t) + \text{clip}\Big(V_\psi(s_t) - V_{\text{old}}(s_t),\ -\epsilon_v,\ \epsilon_v\Big), \tag{7.5}$$

其形式类比自策略截断。但这个类比是薄弱的。策略截断有(6.12)给出的信赖域依据，值截断则没有任何依据——它只是一条启发式，用来防止评论家在内层迭代中移动太远、以致批次开始时算出的优势变得陈旧。Engstrom et al. (2020) 在控制场景中发现去掉它并无害处，而它在语言场景中的地位最恰当的描述是「仍有争议」。如果你跑多于一个内层迭代就保留它；若 $E = 1$，它什么也没做。

回报归一化与值损失系数。 在共享主干时，值损失以系数 $c_v \approx 0.5$ 进入；而在使用独立评论家网络这一更好的设计时（节 7.5），c_v 被替换为一个独立的优化器和一个独立的学习率，后者通常是策略学习率的 3–10 倍，因为评论家求解的是一个监督回归问题，既没有信赖域，也没有慢慢走的理由。用滑动标准差对回归目标 $\hat{G}_t$ 做归一化，可以稳定值头的输出尺度，这一点在 β 或奖励尺度于训练途中发生变化时最为重要。

与 Adam 的相互作用。 Adam 的近似尺度不变性意味着，对所有优势做一次全局缩放会在几百步内被冲掉，所以这些归一化看起来在做的事情，大部分是幻觉。有两个例外要紧。其一，Adam 的 ϵ 会在梯度很小的坐标上破坏这种不变性，这正是 Huang et al. (2024c) 报告 $\epsilon = 10^{-5}$（而非框架默认的 10^{-8}）会可测量地改变 RLHF 结果的原因：在强化学习量级的梯度下，ϵ 相对于二阶矩估计并不可忽略，而是起了阻尼的作用。其二，全局梯度范数截断完全不具有尺度不变性——如果把优势放大会让你持续处在截断阈值之上，那么你的学习率调度就已经被悄悄换成了归一化梯度下降。请记录截断前的梯度范数，而不是截断后的。

代码 7.2: 白化与 GAE。绝大多数 bug 集中在两个细节上：`values` 在掩码之外必须为零，这样最后一个真实词元处的自举才是零；以及评论家的目标 `returns` 必须在优势被白化之前构造——若改为对回报做白化，就会毁掉值头被要求预测的那个尺度。

```
def whiten(x, mask, shift_mean=True):
    n = mask.sum().clamp(min=1)
    mean = (x * mask).sum() / n
    var = (((x - mean) * mask) ** 2).sum() / n
    out = (x - mean) * var.clamp(min=1e-8).rsqrt()
    return out if shift_mean else out + mean
```

```

def gae(reward, values, mask, gamma=1.0, lam=0.95):
    """reward, values, mask: [B, T]. values must already be masked to 0."""
    values = values * mask
    adv = torch.zeros_like(reward)
    running = torch.zeros_like(reward[:, 0])
    for t in reversed(range(reward.size(1))):
        v_next = values[:, t + 1] if t + 1 < reward.size(1) else 0.0
        delta = reward[:, t] + gamma * v_next - values[:, t]
        running = delta + gamma * lam * running
        adv[:, t] = running
    returns = (adv + values) * mask          # 评论家的回归目标
    return whiten(adv, mask) * mask, returns
```

7.5 关于实现细节的那批文献

有三篇论文应当成组阅读。Engstrom et al. (2020) 在连续控制中表明，PPO 相对 TRPO 的优势可归因于「代码层面的优化」——奖励缩放、观测归一化、学习率退火、正交初始化——而非截断替代目标，并且单靠截断替代目标复现不出所报告的结果。Huang et al. (2024c) 为语言场景做了对应的工作：从零复现 Stiennon et al. (2020) 的 TL;DR 摘要 RLHF 结果，匹配上原文的学习曲线与规模化行为，并逐条列出了做到这一点所需的细节。Ivison et al. (2024) 沿四个维度——偏好数据、算法、奖励模型、提示集——做了受控对比，发现更好的偏好数据带来的增益最大，其次是算法，再次是奖励模型；在他们对齐设置的实验中，PPO 在数学上领先 DPO 至多 2.5 分、在通用领域领先 1.2 分。Zheng et al. (2023b) 是有用的第四篇，它的视角来自一支在大规模上稳定 PPO 的团队，而不是复现某个特定结果。

表 7.3 and 7.4罗列了这些细节。每一行给出一个决策、做错时会出现的病态，以及——这才是要紧的部分——一个能检测到它的测量手段。一个你检测不到的细节，就是一个你会反复做错的细节。

7.6 诊断，以及一次健康的训练长什么样

一次 RLHF 训练有一个人人都盯着的指标，以及大约十来个真正携带信息的指标。奖励曲线是其中信息量最低的一个，因为它是针对那个正在被压榨的对象算出来的。

表 7.3: 数据通路上的实现细节。检测那一列才是关键的：这里每一条都可以用一句断言或一个记录下来的标量来验证，而它们中的每一条都曾害某人损失一周时间。

细节	缺了它会坏在哪	如何检测
分数加在最后一个非填充词元上	终局奖励落在被掩掉的填充位置上，永远进不了损失；KL 惩罚成了唯一的信号	奖励曲线平坦而 KL 上升；断言该下标等于 `mask.sum(-1)-1`
生成用左填充，训练用右填充	引擎与训练器之间位置编号与注意力掩码不一致；重算出来的对数概率是垃圾	重算一条采样轨迹的对数概率并与引擎的比较；差距应为每词元 $O(10^{-3})$ 奈特
截断惩罚 / 强制 EOS	长度一路涨到上限，因为有长度偏好的奖励模型会为未写完的回复买单	以 `<eos>` 结尾的回复比例；按正常终止与被截断分开统计的平均长度
所有地方都关闭 dropout	$\pi_{\theta_{\text{old}}}$ 不是真正的采样策略；比值带有噪声底，截断会被虚假触发	断言在第一个内层迭代的第一个小批量上 $\rho \equiv 1$
采样时 $\mathcal{T} = 1$、top-$p = 1$	行为策略是 π_θ 的一个调温版本；每个比值都带有依赖长度的偏差	记录引擎与训练器的对数概率差；当 $\mathcal{T} \neq 1$ 时它是系统性的，而不是噪声
语言模型头处用 FP32 做 log-softmax	比值误差与比值信号本身同量级（节 2.9）	第 0 步 $\log \rho$ 的直方图；它应当是零处的一根尖峰
奖励白化采用固定或缓慢的统计量	β 随策略集中而漂移（节 7.3）	记录 $\sigma_{\mathcal{B}}$ 与 $\beta\sigma_{\mathcal{B}}$
用逐词元而非逐序列的 KL	一条长回复主导了惩罚，因为序列 KL 随 T 增长	逐序列 KL 对长度的直方图；归一化之后相关性应当很弱

表 7.5汇总了常用的一组。表中的取值范围是针对短回复、7B 量级、面向学习得到的奖励模型的训练所总结的实践者经验，它们不是取自任何单篇论文的测量结果，在长文本或智能体式场景中也会不适用。它们的价值在第三列。

代码 7.3: 一次遍历即可算出表 7.5的诊断函数。这里的熵是被采样词元的平均对数概率取负，这是一个廉价而有偏的代理量：精确的熵需要 $\mathcal{V}$ 上的完整分布，应当在 logits 尚在显存中时由它算出。

```python
@torch.no_grad()
def diagnostics(logp, logp_old, logp_ref, values, returns, adv, mask,
                reward, lengths, eos, eps=0.2):
    m = mask.bool(); n = mask.sum().clamp(min=1)
    log_ratio = (logp - logp_old) * mask
    ratio = log_ratio.exp()
    seq_kl = ((logp - logp_ref) * mask).sum(-1)        # 每条序列的奈特数
```

表 7.4: 优化通路上的实现细节。第四行和第五行是最常导致「训得起来但表现不佳」的两条，因为它们都不会改变任何一条曲线的形状。

细节	缺了它会坏在哪	如何检测
用独立的评论家网络，而非共享的值头	值梯度会重塑策略主干；两个目标互相打架，而且哪个学习率都不对	在只训练评论家的预热阶段，策略的熵却在变化
值头的初始化（小标准差、零偏置）	初始优势被一个任意常数主导；早期坍缩	第 0 步的 $\lvert A\rvert$ 与评论家可解释方差
在策略移动之前先预热评论家	最初一百次更新都是噪声，而等评论家变得有用时策略已经被损坏了	要求可解释方差 > 0.3 之后再解冻策略（节 6.2）
损失按整个批次中未被掩掉的词元总数归一化	有效学习率取决于批次是如何切成微批做梯度累积的	把微批大小减半，确认损失曲线不变
内层迭代数 E，配合实测的 $\mathbb{D}_{\mathrm{KL}}\left(\pi_\theta \,\|\, \pi_{\theta_{\text{old}}}\right)$	没有可用信赖域时的 $E > 1$，等于批次内部无人监控的离策略漂移	按每个内层迭代而非每步报告截断比例与 $\mathbb{D}_{\mathrm{KL}}\left(\pi_\theta \,\|\, \pi_{\theta_{\text{old}}}\right)$
跨提示打乱小批量	小批量之间相关；连续几次更新指向同一方向，实际步长是你设定值的 E 倍	相邻梯度方向的自相关
学习率预热，不做重启	最初几步会从冷启动的 Adam 状态出发迈一大步，KL 预算在学到任何东西之前就花光了	最初 20 步的 $\mathbb{D}_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})$
Adam 取 $\epsilon = 10^{-5}$，而非 10^{-8}	在强化学习量级的梯度下，二阶矩归一化会放大极小的梯度坐标	单参数扫描；Huang et al. (2024c) 报告了可测量的效应
确定性随机种子，配置全部记录	消融与种子噪声无法区分，而在 RLHF 的批次规模下种子噪声很大	每个配置跑三个种子；报告离散程度，而不是最好的那次

```
    w = log_ratio.sum(-1).exp()                    # 序列级重要性采样权重
    return {
        "reward/mean":       reward.mean(),
        "kl/per_token":      seq_kl.sum() / n,
        "kl/per_seq_p95":    seq_kl.quantile(0.95),
        "entropy/proxy":     -(logp * mask).sum() / n,
        "clip_frac":         (((ratio - 1).abs() > eps) & m).sum() / n,
        "ratio/log_p99":     log_ratio[m].quantile(0.99),
        "value/expl_var":    1 - (returns[m] - values[m]).var()
                                / returns[m].var().clamp(min=1e-8),
        "adv/mean":          (adv * mask).sum() / n,
        "ess_frac":          w.sum() ** 2 / (w.pow(2).sum() * w.numel()),
        "len/mean":          lengths.float().mean(),
        "len/frac_no_eos":   1.0 - eos.float().mean(),
```

表 7.5: PPO-RLHF 的诊断指标。表中范围是针对短文本偏好数据上 7B 量级模型的经验值，仅作为设定告警阈值的起点，而非实测量。真正可迁移的是「偏离说明什么」这一列。

指标	健康表现	偏离说明什么
奖励模型分数	上升，且逐渐减速	从第 0 步起就平坦：终局奖励放错了位置。上升而留出集质量下降：过度优化
每词元 $\mathbb{D}_{\mathrm{KL}}(\pi_\theta \Vert \pi_{\mathrm{ref}})$	10^{-3}–10^{-2} 奈特	高于 10^{-1}：参考策略已经约束不住任何东西
序列 KL，第 95 百分位	5–30 奈特	超过约 50: 奖励模型的误差已压过它的信号 (Gao et al., 2023a)
策略熵	缓慢单调下降	快速下降：熵坍缩（节 12.6）。上升：不稳定，或者加了熵奖励
截断比例	ε=0.2 时为 0.05–0.20	恰好为 0：`logp_old` 没有 detach，或者 E=1 且没有漂移。高于 0.4：决定步长的是信赖域，而不是学习率
评论家可解释方差	0.6–0.9；绝不低于 0.3	接近 0：评论家退化成常数，GAE 变成蒙特卡洛。为负：比预测均值还差；应怀疑目标构造有 bug
优势均值（未白化）	≈ 0	持续带同一符号：评论家有偏，或者 KL 进了奖励却没进回报
词元对数比值, 第 99 百分位	< 0.3	右尾很重：陈旧度问题，或者引擎与训练器的数值失配
ESS/n (2.17)	> 0.5	低于 0.3：少数几条序列扛起了整次更新
回复长度	持平或缓慢上升	快速上升：长度作弊。触顶比例高于 5%：截断奖励泄漏
以 <eos> 结尾的比例	> 0.95	下降：策略正在学会不终止
梯度范数（截断前）	在 2–3× 内保持稳定	出现尖峰：某个病态批次，或者数值失败

```
22      }
```

常见陷阱: 奖励在涨，质量在跌

奖励模型分数必然会上升；那本来就是你在优化的东西。留出集上的质量则不然。Gao et al. (2023a) 用一个金标准奖励模型去标注代理模型的训练数据,直接测量了这一差距，并发现金标准奖励可以很好地刻画为 $d = \sqrt{\mathbb{D}_{\mathrm{KL}}}$ 的函数 $d(\alpha - \beta \log d)$：先升、到峰、再降，奖励模型越大峰值越靠后，而过度优化系数 β 对奖励模型规模基本不敏感。由此有两条操作层面的推论。其一，代理奖励不是停止准则，而且是在很强的意义上：它按构造就是单调的，因而不携带任何关于「何时该停」的信

息。其二，你该画的横轴是 $\sqrt{\mathbb{D}_{\mathrm{KL}}}$ 而不是步数，因为峰值正是这个量的函数。请维护一套留出评测——人工的、或者一个不是奖励模型的评判模型、或者一套固定的能力测试集——并按 KL 而非按步数的节奏去跑它。Abahana et al. (2026) 进一步论证说，这些转变是局部化的事件，且在发生前带有预警信号，而检查点层面的平均值会掩盖那些在单条提示层面可见的失败；这条建议的实操版本是：记录逐提示的奖励变化量，并去读它的尾部。

7.6.1 典型的失效特征

大多数坏掉的训练都可归入六种失效，而每一种都有一个先于奖励曲线变化而出现的特征。

长度爆炸。 平均长度单调上升，触及长度上限的回复比例增大，以 `<eos>` 终止的比例下降。原因：奖励模型的长度偏好没有受到对抗，通常是因为被截断的回复被打分而不是被惩罚。它通常比任何奖励曲线上的异常都要早出现数百步。

熵坍缩。 熵的下降快于大致「随 log 步数线性」的速率；同一提示下采样出的回复变得几乎一模一样；截断比例趋向于零，因为策略已经不再移动。奖励在一个低于高熵策略所能达到的水平上停滞。这是 RLVR 的主要失效模式（节 12.6），在这里虽然不那么剧烈但同样真实，因为 KL 锚点是一个比熵奖励更好驾驭的熵正则化手段（节 6.5）。

KL 暴涨。 序列 KL 超线性增长，同时奖励模型分数在上升。这就是正在进行中的过度优化，而它几乎总是源于一个小于名义值的有效 β——通常经由节 7.3中的 $\sigma_{\mathcal{B}}$ 机制，偶尔则是因为 β 控制器在对一个用错误估计量测出的 KL 作出反应。文本会变得流畅但风格化：不寻常的标点、套路化的开头、以及奖励模型碰巧喜欢的那些模棱两可的措辞。

评论家发散。 可解释方差下降，值损失上升，优势均值偏离零。通常有三个原因：评论家的回归目标漏掉了优势中包含的 KL 分量；β 被改动过，导致评论家的输出尺度现在是错的；或者评论家的学习率过高而值截断被关掉了。可以通过「优势均值是带符号的（目标不匹配）还是仅仅有噪声（学习率）」来区分。

退化性重复。 某个短语或句子一直重复到长度上限。奖励模型是在人类撰写的比较数据上拟合的，在这一区域里基本没有训练信号，因而可以给这类文本打出任意分数。其特征是一个双峰的奖励分布，带有一个得分很高的小峰；检测办法是把一个批次按分数排序，然后人工读一读排在最前面的二十条回复。奖励截断能限制损害；但只有 KL 锚点或重复惩罚才能阻止它发生。

坍缩到单一格式风格。 每一条回复都变成同样的结构——一个项目符号列表、一段对问题的复述、一句结尾处的「还需要什么帮助」。这不是 bug；这是反向 KL 在做 节 2.4所说的事情：集中到奖励模型偏爱的那些模式上。它要靠多样性指标（对不同提示的回复之间的相异 n-gram）来检测，而不是靠 表 7.5里的任何东西；而且它是最让用户反感、同时又能让每一条训练曲线都显得健康的那种失效。

7.7 真正要紧的超参数

下面两套配置综合了Huang et al. (2024c)、Ouyang et al. (2022)、Zheng et al. (2023b)和Lambert et al. (2025) 所报告的设置，并为与本章其余部分保持一致做了调整。它们是能够稳定训练的起点，而不是最优解。

工程实践：一套可用的 PPO-RLHF 配置：7B 量级策略，短文本偏好数据

- **优化器。** AdamW，$\beta = (0.9, 0.95)$，$\epsilon = 10^{-5}$，权重衰减 0。策略学习率 10^{-6}，经 10 步线性预热后保持常数。评论家学习率 5×10^{-6}，独立的优化器、独立的网络，并从奖励模型检查点初始化，使其输出尺度一开始就合理。
- **批次。** 每次迭代 512 条提示，每条采样一次；小批量为 64 条序列；用梯度累积来适配显存，损失按整次迭代中未被掩掉的词元总数归一化。内层迭代 $E = 1$（仅当生成占据主要成本、且你在按内层迭代记录 $\mathbb{D}_{\mathrm{KL}}\left(\pi_\theta \,\|\, \pi_{\theta_{\text{old}}}\right)$ 时，才用 2）。
- **替代目标。** 对称的 $\varepsilon = 0.2$；值截断 $\epsilon_v = 0.2$；全局梯度范数截断 1.0；不加熵奖励。
- **KL。** 放进奖励，用 k_1 估计量，逐词元计算。β 初始化为 0.02，由自适应控制器管理，目标为 $\delta^\star = 6$ 奈特的序列 KL，增益 0.1，单步变化截断到 $\pm 20\%$，β 夹在 $[0.002, 0.2]$ 之内。

- **优势**。$\gamma = 1$，$\lambda = 0.95$（其正当性仅仅来自奖励被整形过这一点）。在整批层面做白化。奖励白化使用滑动归一化器，动量 0.99，或者使用取自 SFT 策略的固定统计量。
- **生成**。温度 1.0，top-$p = 1.0$，不用 top-k，不加重复惩罚。提示最长 512 词元，回复最长 1024 词元。非 `<eos>` 终止的回复获得一个固定奖励，大约取 SFT 策略分数分布的第 5 百分位。
- **数值**。语言模型头处用 FP32 做 log-softmax。所有模型中都关闭 dropout。参考模型与奖励模型的输出每次迭代只算一次并缓存。
- **时长**。1000–2000 步优化器更新，留出评测按 $\sqrt{\mathbb{D}_{\mathrm{KL}}}$ 的节奏而非按步数的节奏运行。

工程实践: 同样的配置放到 70B 量级，有哪些变化

- **学习率**降到 5×10^{-7}（策略）与 2×10^{-6}（评论家）。在相同的名义学习率下，更大的模型会迈出更大的 KL 步；此时学习率起的是隐式信赖域的作用（节 2.9）。
- **批次**提高到 1024 条提示，小批量 128；$E = 1$，严格执行。在这个规模上，不存在任何一种配置能让「复用一个批次」成为最廉价的改进来源。
- **要固定的是 KL 目标，而不是 β**。把 $\delta^\star$ 保持在同样的「每序列奈特数」上，让控制器自己去找 β。达成给定 KL 所需的 β 因规模而异、也因奖励模型而异，所以照搬过来的 β 毫无意义。
- **显存**是约束瓶颈（表 7.1）。四模型配置预计需要 ≈ 64 张加速卡。参考模型与奖励模型的权重卸载；打开梯度检查点；打开序列打包；节点内用张量并行、节点间用流水线并行。
- **数值问题变成一阶问题**。FP32 的语言模型头是强制要求；若采样是分离部署的，还要针对引擎的对数概率加上截断重要性采样（第22章）。在这个规模上，BF16 的比值不是一个小误差，而是一个不同的算法。
- **认真考虑去掉评论家**。在 70B 上评论家要占 1120 GB、约三十张加速卡。而 $G = 4$–8 的组基线只多花些生成开销，那是带宽受限的、相比之下很便宜，而且它没有一个可能失败的回归问题。节 7.9会说明什么时候这个判断是错的。

工程实践：按顺序该先改什么

当一次训练行为异常时，请抑制住调参的冲动。下面的次序是按「概率加权后的代价」排的。

1. **先验证，别调参**。断言在第一个内层迭代的第一个小批量上 $\rho_t \equiv 1$，断言终局奖励落在 `mask.sum(-1)-1` 处，断言重算出的采样轨迹对数概率与引擎的一致。大多数所谓「不稳定」都是这三者之一。
2. **学习率调低** 3×。如果这样就好了，那你遇到的是信赖域问题，不是估计器问题。
3. **令** $E \to 1$。这会消除批次内部的一切离策略性。
4. **收紧 KL 目标**。把 $\delta^\star$ 减半。如果奖励曲线变平而留出集质量不变，那你之前是在过度优化。
5. **评论家优先于策略**。如果可解释方差低于 0.3，就冻结策略、单独训练评论家几百步。
6. **到这一步才**去动 ε、λ 或白化方案。它们会改变估计器，并让你此前的所有测量结果失去可比性。

7.8 迭代式与在线 RLHF

以上一切都不是只跑一次。Bai et al. (2022a) 引入了迭代式的在线数据采集——部署当前策略，在它的输出上采集新的人类比较数据，重训奖励模型，再跑一遍强化学习——此后这就一直是生产级对齐的标准形态。Xiong et al. (2023) 在显式 KL 约束之下为这个循环给出了理论处理，而它所要求的覆盖条件很有启发性：需要被覆盖的是迭代序列本身，而不是整个策略类；在线循环之所以比离线循环便宜，正是在这个意义上。算法 7.2给出了实用版本。

7.8.1 为什么冻结的奖励模型其保质期是以 KL 计量的

奖励模型是在某个特定分布上抽取的比较数据上拟合出来的函数——在第一轮里，那就是 SFT 策略的分布。它在该分布之外的误差不只是变大而已，而是会被择优选中，因为策略正在明确地搜索高分区域，因而会优先找到误差为正的地方。这就是节 6.5中第一项职责所描述的机制，而Gao et al. (2023a) 给了它一个可用的函数形式：金标准奖励作为 $d = \sqrt{\mathbb{D}_{\mathrm{KL}}}$ 的函数先升后降，峰值位置是奖励模型规模与偏好数据规模的光滑

函数，而过度优化系数对奖励模型规模基本不敏感。最后这一点对规划最为重要：把奖励模型做大只是推迟了过度优化，并没有消除它。

因此实践上的表述不是「每 N 步重训一次奖励模型」，而是「奖励模型只在它所拟合分布周围的一个 KL 球内有效，你必须在走出这个球之前重新拟合」。估计这个半径需要一个金标准信号，而那恰恰是昂贵的东西——这也正是为什么生产流水线会按 KL 的节奏跑一个小规模的留出人工评测，而不是在最后跑一次大规模的。第5章会把奖励模型的鲁棒性、集成与权重平均作为攻克同一问题的另一条路径来讨论。

7.8.2 为什么迭代比看上去更难

有三种失效模式是外层循环特有的。

比较数据池每一轮都在收窄。 从一个已经被反向 KL 锐化过的策略上采集到的偏好数据，其样本对彼此非常接近。从近乎平局的样本对中做 Bradley–Terry 学习，好的情况是信息量低，坏的情况是在拟合噪声；于是第 k 轮的奖励模型在真正要紧的那些回复上的判别能力，可能不如第 1 轮的模型。标准缓解手段是：在升高的温度下采样比较池，在池中保留来自更早策略以及来自其他模型的回复，并分别报告奖励模型在同策略样本对和在 SFT 策略样本对上的准确率，好让这种收窄可见。

跨轮次的参考策略选择是一个真实的决策。 每一轮都锚定到 π_{sft}，意味着 KL 预算是累积的，而随着策略移动锚点会越来越不相关；锚定到 $\pi_\theta^{(k-1)}$ 则每轮重置预算，从而允许总漂移无界；用近期策略的指数滑动平均则介于两者之间。Xu et al. (2024d) 指出，用指数滑动平均更新参考策略是让 PPO 在他们的设置中奏效的要素之一。目前尚无共识，而且论文常常并不说明自己用的是哪一种。

起作用的变量是同策略程度，而不是覆盖度。 Tang et al. (2024a) 检验了对在线—离线性能差距的几种显而易见的解释——数据覆盖度、损失函数、模型规模——发现它们都不能完全解释这一差距，而采样分布的同策略程度可以。Tajwar et al. (2024) 在偏好微调上得到了相容的结论。按证据分级，这是一个 (b) 级结果：在若干设置中精神上得到了复现，但没有能预测差距大小的机制性解释。尽管如此，它仍然是「为在线循环付出代价」这件事最好的现有理由，而 第9章会接着讨论认真对待它之后所导出的博弈论形式化。

7.9 取代它的是什么，以及它还有什么用

到 2026 年年中，带学习评论家的 PPO 已经不是多数模型的对齐方式，而对这次取代，与其外交辞令，不如说得精确些。

在**偏好对齐**上，DPO 家族（第8章）已经接管，原因是成本，而不是质量。受控对比给出的结论恰恰相反：Xu et al. (2024d) 发现在两者都调好的情况下 PPO 明显更优，Ivison et al. (2024) 也发现在对齐设置下 PPO 在数学上领先至多 2.5 分、在通用领域领先 1.2 分。但 DPO 不需要采样、不需要部署奖励模型、不需要评论家，超参数大约只有两个，而表 7.3 and 7.4却列出了十七种把 PPO-RLHF 做错的方式。而且Ivison et al. (2024) 还报告说偏好数据质量的影响压过算法选择，这意味着在预算固定时，理性的分配通常是把钱花在数据上、并采用更便宜的优化器。PPO-RLHF 更好；但它没有好到足够，这就是全部论证。

在**推理**上，GRPO 家族（第12章）已经接管，理由是前面已经拼齐的那些结构性原因：在 10^4 个词元的前缀上只有终局二值奖励时，评论家求解的回归问题，其目标是一条尚未被采样出来的轨迹最终是否正确，因此它总有几分拟合不佳；这份拟合误差会作为偏差作用在每一个词元上；而评论家又要占掉常驻模型状态的 44%（表 7.1），恰恰是在显存构成瓶颈的那个规模上。Ahmadian et al. (2024) 很早就提出了这个论证，而该领域用脚投票的结果此后一直与之一致。

带评论家的 PPO 仍然是正确工具的场合，比它的拥护者所宣称的要少，但也不是零。

真正稠密或经过整形的奖励。 如果奖励在轨迹的多个位置到达——逐轮的工具反馈、逐步的过程奖励模型分数、在特定片段上触发的安全分类器，或者是 KL 整形奖励 (7.1)本身——那么 λ 就在做真实的工作（节 7.2.1），而组基线会把轨迹坍缩成单个标量，从而丢掉这份结构。这是最强的一条理由，也是评论家在智能体式场景中得以存续的原因（第17章）。

多目标混合。 当奖励是若干个头的加权和——有用性、无害性、冗长惩罚、格式检查——且各分量挂在不同位置上时，逐词元的值函数天然能够处理。组相对优势只能看到总和，因此当加权和恰好相等时，它无法区分「有用但不安全」与「安全但没用」的两条回复。

生成预算很小。 组基线要求每条提示 G 个样本。当生成确实昂贵时——真实的工具延迟、长时域片段、有人参与其中的环境——$G = 8$ 根本拿不到，而评论家把基线摊销到了所有提示上，而不是为每条提示单独付费。这与表 6.1 是同一笔账，只是从另一侧来读。

组内没有方差的提示。 表 6.1的最后一行是组基线唯一的定性弱点：当 G 个样本得分都一样时，每个优势都恰好为零，这条提示毫无贡献。评论家仍然会产生梯度。在使用连续的奖励模型分数时，这一约束不如二值验证器那么强；但它对接近饱和的分布仍然要紧——例如一个对齐良好的策略总能正确拒答的那些安全提示——而那里的残余信号恰恰是你想要的。

逐状态信用本身就是关键的长轨迹。 VAPO (Yue et al., 2025b) 是一个存在性证明：精心工程化的基于值的方法可以在长思维链任务上胜过组相对基线（节 12.9）。它只是一篇论文，而且其工程量相当可观，所以这属于「仍有争议」的档位；但它确立了一点：评论家的劣势是「评论家通常被怎么构建」的属性，而不是一条定理。

最后说一句什么东西不会消失。四模型架构已经从多数配方中消失了；节 7.3, 7.5 and 7.6中的工程实践则没有。终局奖励落在哪里，dropout 是否关掉，损失归一化分母是否依赖于微批切分方式，引擎与训练器是否在对数概率上达成一致，截断比例在做什么，以及奖励是否正在被一个不断收缩的标准差所白化——在 2026 年的一次 GRPO 训练里，这些全都是活生生的问题。本章之于 第6章，就如同接线图之于电路理论，而接线的寿命比某个特定电路更长。

本章小结

- PPO-RLHF 常驻四份模型副本——策略、评论家、参考策略、奖励模型——在激活值与 KV 缓存之外，模型状态每参数需要 36 字节。评论家占其中 44%，这正是该领域转向组基线的近因和决定性原因。
- 把 KL 惩罚折进逐词元奖励(7.1)会让奖励变得稠密，从而恢复了 λ 的真实作用——节 6.2.2已说明在只有终局奖励时这一作用是缺失的：λ 设定的是「一个词元要为未来的散度负责多长的时域」，而每个词元自身的 KL 代价则不受衰减地进入。

- KL 放进奖励与 KL 放进损失产生的是不同的梯度，而不是不同的记账方式。放进奖励会给出得分函数项，因此能够抑制进入高散度区域；放进损失则只为已经发生的散度收费。有评论家时选前者，没有时选后者，且绝不要两者兼用。
- 由于 KL 是回报的一部分，评论家的回归目标里也就包含了它。于是值函数是对「奖励模型分数减去剩余散度账单」的估计，这使它依赖长度，并把它的输出尺度与 β 耦合在一起——因此改动 β 会让评论家失效。
- 用批次标准差去除奖励，等于把 β 换成了 $\beta\sigma_{\mathcal{B}}$。由于 $\sigma_{\mathcal{B}}$ 会随策略集中而收缩，奖励白化恰恰在策略漂移最快的时候放松了 KL 锚定。请记录 $\beta\sigma_{\mathcal{B}}$；它才是真正的超参数。
- 优势白化作为尺度迁移手段是站得住脚的，作为基线则站不住脚：批次均值相减会耦合互不相干的提示，除以标准差是一个依赖数据的学习率，Adam 让大部分收益变得冗余，而这一操作还删掉了优势均值——那本是你手头最好的评论家偏差诊断量。
- 两项独立的复现研究得出结论：PPO 所报告的优势大部分在于实现细节，而不在截断替代目标。表 7.3 and 7.4列出了十七个这样的细节，并为每一个配上了能检测到它的测量手段；一个你检测不到的细节，就是一个你会反复做错的细节。
- 奖励曲线是 RLHF 训练中信息量最低的指标，因为它是针对那个正在被压榨的对象算出来的。金标准奖励作为 $\sqrt{\mathbb{D}_{\mathrm{KL}}}$ 的函数先升后降，所以评测应当按 KL 而不是按步数来安排，而代理奖励永远不能作为停止准则。
- 一个冻结的奖励模型只在其训练分布周围的一个 KL 球内有效，这正是生产级对齐要有外层循环的原因。迭代本身也有病态：每一轮的比较数据池都取自一个更窄的策略，因此奖励模型恰恰会在最要紧的那些比较上退化。
- 到 2026 年，带评论家的 PPO 在偏好对齐上已被 DPO 家族取代（原因是成本而非质量——受控对比是偏向 PPO 的），在推理上则被 GRPO 家族取代（原因是评论家的回归问题及其显存开销）。它仍然是稠密或整形奖励、多目标混合、生成预算很小、零方差提示分布，以及逐状态信用本身就是关键的那些场景中的正确工具。

延伸阅读

请开着终端把Huang et al. (2024c) 完整读一遍。它是文献中唯一一篇从零复现了一个已发表 RLHF 结果、并如实报告所需条件的论文；它的价值不在那份细节清单本身，而

在于它证明了这份清单存在、而且很长。请把它与Engstrom et al. (2020) 对照着读——后者早五年就在连续控制上提出了同样的论证却基本被忽视——好让你校准：对任何一项被报告的算法改进，应当相信多少。Zheng et al. (2023b) 是这组文献的第三篇，它写自一支在大规模上稳定 PPO 的团队内部，而非从外部复现，是已发表文献中对「哪里会出问题」最坦率的记述。

关于原始流水线，逐词元 KL 奖励(7.1)与自适应 β 控制器都出自Ziegler et al. (2019)；它篇幅很短，值得一看的是：文献中最具后果的那个实现决策，当初被引入时有多么随意。Stiennon et al. (2020) 是这套方法奏效的最干净的演示，而Ouyang et al. (2022) 则是让它成为战略优先项的那一篇。

关于何时该停，Gao et al. (2023a) 是必须知道的一篇，而且应当把它当作一套测量方法论而不是一条规模化定律来读：其中「金标准对代理」的构造是可复用的，却几乎没有人复用它。Abahana et al. (2026) 是一篇有用的 2026 年补充，它把奖励作弊当作一种带有可检测前兆的训练动态，而不是一种终局属性；不过作为一篇单独的近期论文，它的分类方式应当被松散地看待。

关于与直接方法的对比，Ivison et al. (2024) 是控制得最好的研究，而它的排序——数据、其次算法、再次奖励模型——是对齐文献中最具决策价值的单条发现。Xu et al. (2024d) 是支持 PPO 的最强论证，并指出了让它奏效的那些要素；请把它与 Tang et al. (2024a) 一起读，后者把在线优势定位在同策略程度而非覆盖度上，并且坦率地承认没有一个受检验的假设能完全解释这一差距。

习题

7.1（整形奖励下的 GAE，两个极限。）把整形奖励 (7.1)与 $\gamma = 1$ 代入(6.6)展开，验证(7.2)。然后在「逐词元的 KL 项 ℓ_t 是 i.i.d. 且方差为 s^2、评论家精确」的模型下，把 $\mathbb{V}\mathrm{ar}[A_t]$ 算成 λ 与剩余词元数 $T - t$ 的函数。求出使「偏差平方（来自评论家误差 ϵ_V）与方差之和」最小的 λ，并说明它如何依赖于 T。

7.2（奖励落在哪里，白化又藏起了什么？）某个批次的回复经右填充后长度分别为 $\{12, 400, 1024\}$，装在宽度为 1024 的张量中，而一个 bug 把终局奖励模型分数对这三条都加在了下标 1023 处。对每一条回复，说明该分数是否能到达损失，若能，则说明它带着怎样的优势。预测 500 步内奖励曲线、KL 曲线和平均回复长度的定性形状，并解释为什么这个 bug 从 KL 曲线上比从奖励曲线上更容易发现。最后，

证明按批次做奖励白化会让这个 bug 在奖励曲线上更难被看到，并说出你会转而信任表 7.5中的哪一个指标。

7.3（白化删掉了一个诊断量。）假设评论家有一个常数偏差，即对所有 s 都有 $V_\psi(s) = V^{\pi_\theta}(s) + b$。分别在 $\lambda = 1$ 和 $\lambda < 1$ 下，计算批次白化前后 GAE 优势的均值。证明白化把 b 的可观测后果从优势中移除了，却没有从评论家的回归目标中移除，并描述由此在多步之后产生的训练动态。你应当改为记录什么？

7.4（截断的两种处理方式。）考虑一个带长度偏好的奖励模型，将其建模为 $r_\phi(x, y) = q(x, y) + c\,|y|$，其中 $c > 0$ 很小，q 是你真正关心的质量。比较两种策略：(a) 被截断的回复获得一个固定的低奖励 r_{trunc}；(b) 被截断的回复从损失中被掩掉。对每一种，判断 (6.14)下的最优策略是否具有有限长度，并求出情形 (a) 中使「撞上长度上限严格无利可图」的 r_{trunc} 条件。解释为什么 (b) 在 RLVR 中是标准做法，而在这里是错的。

7.5（Adam 并不像宣传的那样尺度不变。）考虑单个参数，其每一步的梯度都是 g，Adam 参数为 $\beta_1 = 0.9$、$\beta_2 = 0.95$、ϵ。把稳态更新幅度算成 g 与 ϵ 的函数，并求出使得 $\epsilon = 10^{-5}$ 相对于 $\epsilon = 10^{-8}$ 把更新改变超过 10% 的那个 g 值。取一个强化学习微调中合理的逐参数梯度幅度，评论Huang et al. (2024c) 关于 Adam ϵ 的发现是否令人意外。然后在开启全局梯度范数截断的条件下重做这一分析，并说明此时学习率实际控制的是什么。

7.6 ⋆（前瞻的 KL 项值不值它的方差?） 节 7.2论证说，KL 放进奖励会给出 KL 放进损失所遗漏的得分函数项，因而能够惩罚进入高散度状态。请构造一个两状态、两动作的例子，使这两种放置方式具有不同的最优解，并量化这一差距。然后把得分函数项的方差代价估计为 T 的函数并作比较，以判断是否存在某个区间，使得正确的梯度在经验上反而不如那个不完整的梯度。最后，提出一个无需评论家也能捕捉到前瞻项的估计器——或者论证它不存在——并把你的答案与「GRPO 家族为何没怎么讨论就接受了不完整的梯度」联系起来。

7.7 ⋆（一个建模散度账单的评论家。）在 (7.1)之下，评论家的目标可分解为 $\hat{G}_t = \hat{q}_t - \beta \hat{m}_t$，其中 $\hat{q}_t$ 估计奖励模型分数、$\hat{m}_t$ 估计剩余 KL。请提出一个分别回归这两项的双头评论家，并分析它能买到什么：精确说明它消除了节 7.2中三种病态中的哪几种，它是否让改动 β 变得没有代价，以及它引入了什么新的失效模式。然后设计一个实验，在 节 12.12所隐含的种子噪声水平下，用不到五次训练就能把你的架构与单头评论家区分开来。

7.8 † **（做消融的是细节，不是算法。）**在一个 1–7B 的模型上、针对一个公开奖励模型，复现一次短文本 RLHF 训练，以节 7.7中的 7B 配置为基线。然后针对表 7.3 and 7.4中至少六行做单细节消融：分数落点、截断惩罚、dropout、损失归一化分母、奖励白化统计量，以及 Adam ϵ。对每一项，记录代码 7.3的完整诊断集，并同时报告留出集胜率以及——更重要的——哪一个被记录的指标最先发生变化。请用三个种子并报告离散程度。你应当至少会找到一个消融，其在奖励曲线上的效应不可见、而在留出集质量上的效应可见；这一对现象正是本习题的要点。

7.9 † **（自己动手测量过度优化。）**先做解析部分：采用 Gao et al. (2023a) 报告的形式 $R_{\mathrm{RL}}(d) = d(\alpha - \beta \log d)$，其中 $d = \sqrt{\mathbb{D}_{\mathrm{KL}}}$，求出使金标准奖励最大的 $d^{\star}$；再推导：若一轮在同策略比较数据上的奖励模型重训把 α 提高了 $1 + \kappa$ 倍而 β 不变，峰值会移动多少——这就是节 7.8中「保质期」的定量版本，它告诉你一份固定的标注预算应当分成几轮。再做实证部分：在小规模上实现Gao et al. (2023a) 的「金标准对代理」构造。取一个强的公开奖励模型作为金标准，用它在一个提示集上标注合成的比较数据，在这些标签上训练一个更小的代理奖励模型，然后针对代理跑 RLHF、同时用金标准评估。把金标准奖励与代理奖励一起对 $\sqrt{\mathbb{D}_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{sft}})}$ 作图并定位峰值。用一个在 4× 标注量上训练的代理、以及一个规模加倍的代理各重复一次，并报告峰值移动的方向是否与所报告的规模化定律预测一致。最后，把 KL 从奖励移到损失后跑同样的实验，报告峰值位置是否变化；并用表 7.2解释你的结果。

算法 7.1 PPO-RLHF 的一次迭代，所有后处理都显式写出。这是把算法 6.1的选择 1–5 实例化后的结果：截断的词元比值、来自学习得到的评论家的 GAE、词元均值归一化分母、KL 折进奖励。第 17 行的替代目标就是代码 6.1。

输入： 策略 π_θ，评论家 V_ψ，参考策略 π_{ref}，奖励模型 r_ϕ，提示集 $\mathcal{D}$，截断参数 ε，值截断参数 ϵ_v，系数 β，目标 KL $\delta^\star$，内层迭代数 E

1: **采样**
2: $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$；将权重同步到推理引擎
3: 采样 $\mathcal{B} \subset \mathcal{D}$；在 $\mathcal{T}=1$、top-$p=1$ 下生成 $y \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$，并记录逐词元对数概率与 `<eos>` 标志
4: **打分**
5: 对所有 y 计算 $r_\phi(x,y)$；截断到 $[-c,c]$；对缺少 `<eos>` 的样本用 r_{trunc} 覆盖 ▷ 代码 7.1
6: $\tilde{r} \leftarrow (r_\phi - \mu)/(\sigma + \epsilon)$，其中 μ,σ 取滑动统计量
7: 在所有回复词元上以 FP32 重算 $\log \pi_{\theta_{\text{old}}}$ 与 $\log \pi_{\text{ref}}$
8: 断言 $\max_t \left|\log \pi_\theta - \log \pi_{\theta_{\text{old}}}\right| < \tau$ ▷ 引擎与训练器的一致性
9: **奖励整形与优势**
10: $\hat{r}_t \leftarrow -\beta(\log \pi_{\theta_{\text{old}}} - \log \pi_{\text{ref}})_t + \mathbb{1}[t=T]\,\tilde{r}$ ▷ (7.1)
11: 对所有 t 计算 $V_\psi(s_t)$；在回复掩码之外置零
12: 由 $\hat{r}$ 与 V_ψ 计算 $A_t \leftarrow \text{GAE}(\gamma{=}1,\lambda)$；　$\hat{G}_t \leftarrow A_t + V_\psi(s_t)$
13: 在整个批次上 $A \leftarrow \text{whiten}(A)$ ▷ 绝不按小批量做
14: **更新**
15: **for** $e = 1$ **to** E **do**
16: **for** 打乱后的小批量 m **do**
17: $\rho_t \leftarrow \exp(\log \pi_\theta - \log \pi_{\theta_{\text{old}}})$；对 $\min(\rho_t A_t, \text{clip}(\rho_t,\,1-\varepsilon,\,1+\varepsilon)\,A_t)$ 做上升
18: 用独立的优化器对 $\frac{1}{2}\max((V_\psi - \hat{G})^2, (V_\psi^{\text{clip}} - \hat{G})^2)$ 做下降
19: 将全局梯度范数截断到 1，并记录截断前的数值
20: **end for**
21: **if** $\mathbb{D}_{\text{KL}}\left(\pi_\theta \,\middle\|\, \pi_{\theta_{\text{old}}}\right) > 1.5\,\delta^\star$ **then**
22: **break** ▷ 截断并不能强制这一点
23: **end if**
24: **end for**
25: **控制**
26: $\beta \leftarrow \beta(1 + 0.1\,\text{clip}((\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}) - \delta^\star)/\delta^\star,\,-0.2,\,0.2))$
27: **return** $\pi_\theta, V_\psi, \beta$

算法 7.2 迭代式（在线）RLHF。其中的内层调用是算法 7.1。这个循环之所以存在，是因为奖励模型的有效性是「与其训练分布之距离」的函数，因此它的保质期是以 KL 而不是以步数来计量的。

输入： SFT 策略 π_{sft}，种子偏好集 $\mathcal{D}_0$，提示池 $\mathcal{X}$，金标准评估器 $\mathcal{G}$（人类或留出的评判模型）

$\pi_\theta^{(0)} \leftarrow \pi_{\text{sft}}$；$\pi_{\text{ref}} \leftarrow \pi_{\text{sft}}$
for 轮次 $k = 1, 2, \dots, K$ **do**
  在同策略的比较数据上刷新奖励模型
  从 $\mathcal{X}$ 采样提示；在升高的温度下，对每条提示从 $\pi_\theta^{(k-1)}$ 抽取 2–4 条回复 ▷ 多样性才是稀缺的输入
  采集比较数据；$\mathcal{D}_k \leftarrow \mathcal{D}_{k-1} \cup \{\text{新的样本对}\}$
  在 $\mathcal{D}_k$ 上拟合 $r_\phi^{(k)}$；分别报告它在来自 $\pi_\theta^{(k-1)}$ 和来自 π_{sft} 的样本对上的准确率
  优化
  $\pi_\theta^{(k)} \leftarrow$ 从 $\pi_\theta^{(k-1)}$ 出发、针对 $r_\phi^{(k)}$ 运行 算法 7.1，π_{ref} 按下文的策略选取
  在若干检查点上评估 $\mathcal{G}$，以 $\sqrt{\mathbb{D}_{\text{KL}}\left(\pi_\theta^{(k)} \,\|\, \pi_{\text{sft}}\right)}$ 为索引；保留取到最大值的那个
  停止
  if $\mathcal{G}$ 相比第 $k-1$ 轮没有提升 **then**
    break ▷ 继续下去只会收窄数据，而不是缩小差距
  end if
end for
return 在 $\mathcal{G}$ 下最好的检查点

第8章

直接对齐算法

核心思想：奖励模型从来不是目的；它只是一个中间变量，而且可以被消去

第6章表明，带 KL 正则的目标存在闭式最优解：它是一个吉布斯分布，把参考策略按 $\exp(r/\beta)$ 加以倾斜。这一关系是可逆的：任何一个策略就是一个奖励函数，至多相差一个只依赖于提示的常数。把逆关系代入 Bradley–Terry 似然，该常数恰好相消，剩下的是一个定义在偏好对上的有监督分类损失——无需采样，无需评论家，也无需奖励模型。这就是直接偏好优化（Direct Preference Optimization，DPO）。随之涌现的五十余种算法与其说是一座动物园，不如说是沿三条轴线所做的选择：选哪种偏好似然、选哪种正则项、选哪种数据分布。而那些失效模式——似然位移、长度作弊、模式坍缩——都可以从这些选择中预见地推导出来。

第7章留给我们的是一台能用但昂贵的机器：内存中常驻四份模型副本、一个必须从零训练的评论家、一个会逐渐陈旧的奖励模型，以及十几个悄无声息地决定训练成败的实现细节。人们有理由追问：为了完成一个说到底相当朴素的目标——让模型偏好标注者所偏好的回复——这一整套装置是否必要。

答案是并不必要。本章的核心结论是：对于离线成对偏好数据、在 *Bradley–Terry* 模型与反向 *KL* 正则的特定情形下，整条强化学习流水线会坍缩成一个可微损失，一位研究生一年级学生用一个下午就能实现。这个结论是精确的，而非近似的，值得精确地理解——既因为它是全世界使用最广的对齐算法，也因为这一坍缩成立的确切条件恰恰告诉你它何时会失败。

8.1 不用强化学习，从奖励直达策略

8.1.1 重参数化

回忆第6章中带 KL 正则的目标，

$$\mathcal{J}(\pi_\theta) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}}\Big[\mathop{\mathbb{E}}_{y\sim\pi_\theta(\cdot\,|\,x)}[r(x,y)] - \beta\,\mathbb{D}_{\mathrm{KL}}(\pi_\theta(\cdot\,|\,x)\,\|\,\pi_{\mathrm{ref}}(\cdot\,|\,x))\Big], \tag{8.1}$$

根据定理 6.3，它在所有分布上的最大值点为

$$\pi^\star(y\,|\,x) = \frac{1}{Z(x)}\,\pi_{\mathrm{ref}}(y\,|\,x)\exp\!\Big(\tfrac{1}{\beta}r(x,y)\Big), \qquad Z(x) = \sum_{y'\in\mathcal{Y}}\pi_{\mathrm{ref}}(y'\,|\,x)\exp\!\Big(\tfrac{r(x,y')}{\beta}\Big). \tag{8.2}$$

在推导(8.2)的过程中，没有任何一步要求 r 必须是一个学出来的模型，也没有任何一步要求我们必须通过优化去寻找 $\pi^\star$。这个等式是两个对象之间的恒等式，而恒等式可以从任一方向来读。解出 r：

$$\boxed{r(x,y) = \beta\log\frac{\pi^\star(y\,|\,x)}{\pi_{\mathrm{ref}}(y\,|\,x)} + \beta\log Z(x).} \tag{8.3}$$

请仔细阅读(8.3)，因为本章其余的一切都是它的推论。它说的是：奖励函数这一类对象与策略这一类对象是同一类对象，在给定 π_{ref} 与 β 的条件下两者构成双射，至多相差一个只与提示有关的加性函数。那个难以计算的配分函数 $Z(x)$——对指数级大的回复空间求和——已经被完全隔离到那个加性项中。

注记 8.1 (为什么 $\beta\log Z(x)$ 不是麻烦). 在多数推导中，难以处理的归一化常数是一个需要近似的障碍。这里它却是一份礼物，因为它只依赖于 x。任何对奖励作逐提示平移保持不变的下游目标，根本不会看到它。Bradley–Terry 似然恰好就是这样一种目标，因为它只通过同一提示下两个奖励的差来依赖奖励。整个技巧就在于此。

8.1.2 代入偏好似然

Bradley–Terry 模型（节 5.2）假定，一位人类在比较同一提示下的两个回复时，以如下概率偏好 y^+ 胜过 y^-：

$$\mathbb{P}(y^+ \succ y^-\,|\,x) = \sigma(r^\star(x,y^+) - r^\star(x,y^-)), \qquad \sigma(u) = \frac{1}{1+e^{-u}}. \tag{8.4}$$

在第5章中，我们用极大似然把 r_ϕ 拟合到(8.4)上，再把 r_ϕ 交给第7章。现在换一种做法：直接代入(8.3)。

推导: 四行推出直接偏好优化

通过(8.3)用策略 π_θ 隐式地参数化奖励，记

$$\hat{r}_\theta(x,y) \;:=\; \beta \log \frac{\pi_\theta(y\,|\,x)}{\pi_{\text{ref}}(y\,|\,x)}. \tag{8.5}$$

于是出现在(8.4)中的奖励差为

$$\begin{aligned} r(x,y^+) - r(x,y^-) &= \big[\hat{r}_\theta(x,y^+) + \beta \log Z(x)\big] - \big[\hat{r}_\theta(x,y^-) + \beta \log Z(x)\big] \\ &= \hat{r}_\theta(x,y^+) - \hat{r}_\theta(x,y^-) \;=:\; h_\theta(x,y^+,y^-). \end{aligned} \tag{8.6}$$

配分函数精确地相消了，没有任何近似。因此，在 θ 上最大化偏好数据集的 Bradley–Terry 对数似然，等价于最小化

$$\boxed{\mathcal{L}_{\text{DPO}}(\theta) = -\mathop{\mathbb{E}}_{(x,y^+,y^-)\sim\mathcal{D}}\Big[\log \sigma\Big(\beta \log \frac{\pi_\theta(y^+\,|\,x)}{\pi_{\text{ref}}(y^+\,|\,x)} - \beta \log \frac{\pi_\theta(y^-\,|\,x)}{\pi_{\text{ref}}(y^-\,|\,x)}\Big)\Big].} \tag{8.7}$$

式 (8.7)是一个二元交叉熵损失。它不需要从模型中采样，不需要奖励模型，不需要评论家，也不需要采样引擎：每个样本只需两次通过 π_θ 的前向传播和两次通过冻结的 π_{ref} 的前向传播，而且全部作用在训练开始前就已收集好的数据上。Rafailov et al. (2023) 将其命名为直接偏好优化，其实践后果立竿见影：原本需要一整套分布式 RLHF 系统的对齐训练，变成了一个单节点就能跑完的微调任务。

8.1.3 梯度，以及催生整个文献的两个事实

对(8.7)求导得到

$$\nabla_\theta \mathcal{L}_{\text{DPO}} = -\beta \mathop{\mathbb{E}}_{\mathcal{D}}\Big[\underbrace{\sigma(-h_\theta)}_{\text{weight}}\big(\nabla_\theta \log \pi_\theta(y^+\,|\,x) - \nabla_\theta \log \pi_\theta(y^-\,|\,x)\big)\Big]. \tag{8.8}$$

其结构是一次带权对比更新：抬高被选中的序列，压低被拒绝的序列，权重 $\sigma(-h_\theta)$ 逐样本地度量当前隐式奖励模型把这一对排错得有多严重。模型已经排对且很自信的样本对（$h_\theta \gg 0$）几乎得不到梯度；排错的样本对则几乎得到全部的 β。这相当于自动的难例挖掘，也正是 DPO 能在会摧毁一次 SFT 训练的学习率下依然稳定训练的原因。

(8.8)的两条结构性性质值得单独列出来陈述，因为后续文献的绝大部分都由这两条性质所生成。

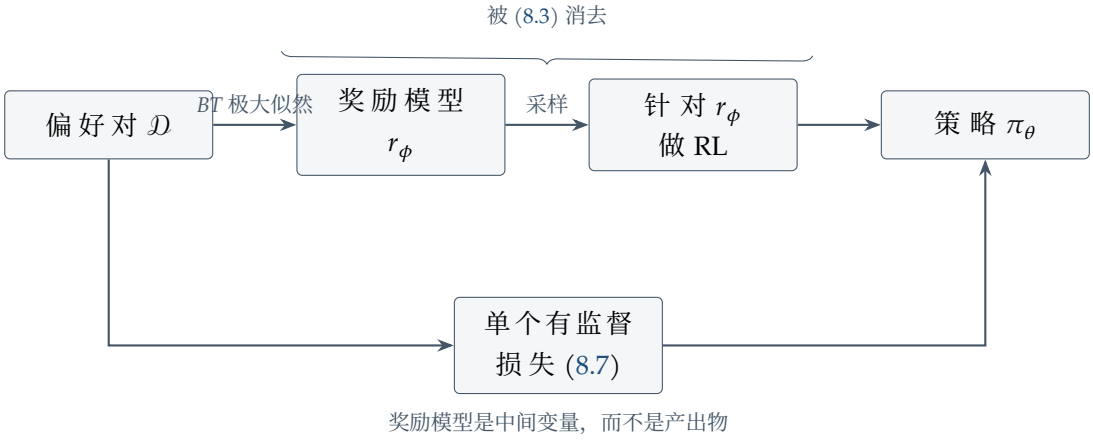


图 8.1: 从偏好数据抵达对齐策略的两条路线。上面一条是第5章接着 第7章。下面一条利用奖励–策略双射(8.3)，彻底跳过中间变量。两条路线指向同一个最优解，但正如节 8.10所论证的，它们的优化难度并不相同，退化的方式也不相同。

命题 8.2 (损失只约束差值)**.** $\mathcal{L}_{\text{DPO}}$ 只通过 h_θ 依赖 θ。因此，取 $\log \pi_\theta(y^+ \mid x)$ $-\!=$ δ_w、$\log \pi_\theta(y^- \mid x)$ $-\!=$ δ_l，其中 $\delta_l > \delta_w > 0$，这一更新在严格降低损失的同时，也降低了被偏好回复的似然。

命题 8.3 (间隔是词元上的求和)**.** $h_\theta = \beta \sum_t \log \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\text{ref}}(y_t \mid x, y_{<t})}$，其中对 y^+ 的词元求和，再减去对 y^- 的同一求和。因此，可用于推动 h_θ 的项数与回复长度成正比，而梯度中没有任何长度归一化。

定理 8.2是似然位移（likelihood displacement，节 8.5）的种子；定理 8.3是长度偏置（length bias，节 8.6）的种子。两者在原始论文的公式中都清晰可见，却花了大约一年才被正确诊断出来。这一延迟本身就有教益：一个目标函数的失效模式通常可以从该目标函数本身推导出来，而本领域先靠经验去发现它们的习惯是一种方法论上的弱点，而非自然规律。

8.2 DPO 是什么，不是什么

这里值得咬文嚼字，因为「DPO 等价于 RLHF」这一含糊说法已经造成了大量混乱。

DPO 不是强化学习。 它不从正在训练的策略中采样，没有探索，通常意义上也不存在信用分配问题。它是在一个巧妙选定的参数化下的极大似然问题。(8.7)中的强化学习成分完全存在于推导之中，而不在算法之中。

这种等价是最优解之间的等价，不是过程之间的等价。 节 8.1.2所确立的是：如果（i）偏好确实由某个真实奖励按 Bradley–Terry 生成，（ii）正则项是相对 π_{ref} 的反向 KL，（iii）策略类不受限制，（iv）我们能精确求解这两个问题，那么图 8.1中的两条路线抵达同一个策略。而这些条件在实践中无一成立：策略类是一个容量有限的 transformer；数据是有限的；两个问题都只用几千步 Adam 求解。

这种等价在一个更尖锐的意义上也是有条件的。 Yang et al. (2026b) 指出了标准论证中的一个隐含假设：它预设了 RLHF 最优策略已经给人类偏好的回复分配了更高的概率。当这一点不成立时——在有限的 β 下，在 π_{ref} 校准很差的提示上，这是可能发生的——DPO 优化的是 相对于参考策略的相对优势，而非与人类偏好的绝对一致，并且会容许这样的最优解：它降低了 DPO 损失，却仍然偏好那个不被偏好的回复。这是一项 2026 年的结果，其实际适用范围尚未被很好地勾勒，但它准确地定位了一处从业者凭经验已经察觉了两年的缺口。

离线这一限制才是真正起约束作用的。 DPO 的损失是在数据分布 $\mathcal{D}$ 下的期望，而不是在 π_θ 下的期望。因此该目标对 $\mathcal{D}$ 支撑集之外的回复什么也没说，而(8.3)允许隐式奖励在这些回复上取任意值。Xu et al. (2024d) 把这一点说得很精确，并表明它并非纯粹的理论隐忧：经 DPO 训练的策略确实会发现隐式奖励很高、质量却很低的分布外回复。我们将在节 8.7回到这个问题，而在线的补救手段则构成了第9章。

历史注记：你的语言模型其实是一个 Q 函数

DPO 问世一年后，Rafailov et al. (2024b) 证明(8.5)中的对数比值不仅仅是一个序列级分数：在前缀上求值时，它是第2章那个反向 KL 正则化词元 MDP 的一个合法的词元级 Q 函数。具体地，$\beta \log \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\text{ref}}(y_t \mid x, y_{<t})}$ 的行为相当于位置 t 处的优势。这为节 8.8中的词元级与步骤级变体追认了合法性，也是直接对齐算法文献与第13章中基于值的方法之间最干净的桥梁。

代码 8.1: 整个算法。有两个细节至关重要且容易出错：参考模型的对数概率必须在策略处于 `eval` 模式且位于 `no_grad` 之下时计算；被选中与被拒绝的样本必须拼接成同一个批次，使两个分支的填充与注意力掩码处理方式完全一致。

```python
def sequence_logp(model, input_ids, attn_mask, loss_mask):
    # loss_mask 在回复词元上为 1，在提示与填充上为 0
    logits = model(input_ids=input_ids, attention_mask=attn_mask).logits[:, :-1]
    labels = input_ids[:, 1:]
    per_token = torch.log_softmax(logits.float(), dim=-1).gather(
        2, labels.unsqueeze(2)).squeeze(2)
    return (per_token * loss_mask[:, 1:]).sum(-1)          # [B]

def dpo_loss(policy, ref, batch, beta=0.1):
    ids, am, lm = batch["input_ids"], batch["attn_mask"], batch["loss_mask"]
    n = ids.shape[0] // 2                                  # [被选中; 被拒绝]
    pi = sequence_logp(policy, ids, am, lm)
    with torch.no_grad():
        rf = sequence_logp(ref, ids, am, lm)
    pi_w, pi_l = pi[:n], pi[n:]
    rf_w, rf_l = rf[:n], rf[n:]
    margin = beta * ((pi_w - rf_w) - (pi_l - rf_l))        # h_theta
    loss = -torch.nn.functional.logsigmoid(margin).mean()
    stats = {                                              # 以下量全部记录
        "margin": margin.mean(),
        "acc": (margin > 0).float().mean(),                # 隐式奖励模型准确率
        "chosen_logp_delta": (pi_w - rf_w).mean(),         # 不应坍缩
        "rejected_logp_delta": (pi_l - rf_l).mean(),
    }
    return loss, stats
```

常见陷阱: 缓存参考模型的对数概率

由于 π_{ref} 是冻结的，它的对数概率可以只算一次并缓存起来，从而节省一半显存、把单步时间也大致减半。这里有两个陷阱。第一，缓存必须以训练中所用的确切分词与截断方式为键；一旦改动最大长度，缓存就会悄无声息地失效。第二，如果你之后改用会更新参考模型的变体（节 8.9），缓存就是错的，其症状是损失在下降而质量在退化。缓存当然可以用，但请在每次训练开始时断言：随机抽取的一小部分样本重新计算的结果与缓存在数值容差内一致。

8.3 设计空间：一个框架，五十种算法

在罗列变体之前，先拿到地图是值得的。目前存在两套组织框架，二者互为补充。

8.3.1 广义偏好优化

Tang et al. (2024b) 观察到，(8.7)是作用在标量

$$\varrho_\theta := \log \frac{\pi_\theta(y^+ \mid x)}{\pi_{\text{ref}}(y^+ \mid x)} - \log \frac{\pi_\theta(y^- \mid x)}{\pi_{\text{ref}}(y^- \mid x)} \;=\; h_\theta/\beta, \tag{8.9}$$

上的一个二分类损失，并且任何凸的、递减的损失 f 都给出一个合法的离线偏好目标：

$$\mathcal{L}_{\text{GPO}}(\theta) = \mathop{\mathbb{E}}_{(x,y^+,y^-)\sim\mathcal{D}}[f(\beta\varrho_\theta)]. \tag{8.10}$$

f 的选取可以复现出各种有名字的算法（表 8.1），但真正有用的回报是解析性的。把 f 在 $\varrho_\theta = 0$ 处作泰勒展开：

$$f(\beta\varrho_\theta) \approx f(0) + \beta f'(0)\,\varrho_\theta + \tfrac{1}{2}\beta^2 f''(0)\,\varrho_\theta^2. \tag{8.11}$$

一阶项是偏好信号。二阶项是对数比值上的平方惩罚，它恰好在数据支撑集上 $\pi_\theta = \pi_{\text{ref}}$ 时取到最小值，并且在经验上与 $\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})$ 同步变化。

命题 8.4 (每一个离线直接对齐算法都自带一个隐式 KL 惩罚)**.** 在(8.11)下，正则强度与偏好信号之比为 $\beta f''(0)/(2|f'(0)|)$。因此，一个无参考模型的目标并不缺少正则，它只是把正则从显式的 π_{ref} 项挪进了其连接函数的曲率之中。

定理 8.4正是对这一文献中最常见问题的正确回答，即：SimPO 没有参考模型，那么是什么阻止它任意漂移？答案是逻辑斯谛连接函数的曲率再加上一个小学习率；其实践后果是，无参考方法的正则更弱、也更难控制，而不是完全没有正则。

f 尾部的形状与它在零点的曲率同等重要。逻辑斯谛连接函数按 $e^{-\beta\varrho}$ 衰减，所以它的梯度永不消失，目标函数会永远奖励更大的间隔：在线性可分的偏好数据上，DPO 会把 $\varrho_\theta \to \infty$ 推到底，这意味着在被拒绝的回复上 $\pi_\theta \to 0$。这就是过度优化的通道，而它是连接函数的性质，不是数据的性质。

8.3.2 三条轴线

Raheja and Pochhi (2026) 给出了互补的、更粗粒度的地图，而这正是值得记住的那一张。本章与下一章中的每一种方法，都是沿三条正交轴线所做的选择：

1. **偏好模型**——目标函数底下是哪一种似然。Bradley–Terry（DPO）、偏好概率的一般非降函数（ΨPO/IPO）、前景理论式的价值（KTO）、带逐提示标注者离散

表 8.1: 广义偏好优化：那些有名字的离线算法，不过是把某个经典二分类损失施加到缩放后的对数比值之差 $\beta\varrho_\theta$ 上的不同选择。最后一列是定理 8.4中的比值，以 β 为单位；数值越大，意味着每单位偏好信号所配的隐式 KL 正则越强。最后三行是合法的目标函数，但截至 2026 年年中，似乎还没有人以某个名字把它们发表出来。

$f(u)$，$u=\beta\varrho_\theta$	经典损失	算法	$f''(0)/2\,\lvert f'(0)\rvert$
$\log(1+e^{-u})$	逻辑斯谛对数损失	DPO (Rafailov et al., 2023)	1/4
$\max(0,\,1-u)$	合页损失	SLiC (Zhao et al., 2023b)	0
$(u-1)^2$	平方损失	IPO (Azar et al., 2024)	1
e^{-u}	指数损失（AdaBoost）	—	1/2
$(\max(0,1-u))^2$	截断二次损失	—	1/2
$(1+e^{u})^{-2}$	Savage 损失	—	0

度的 Mallows 模型 (Chen et al., 2024c)，或者根本不假设存在标量奖励（第9章中的 Nash 形式化）。

2. **正则机制**——如何控制相对 π_{ref} 的偏离。显式的反向 KL、一般的 f 散度 (Wang et al., 2024a)、χ^2 与 KL 的混合（节 8.7）、定理 8.4中的隐式曲率惩罚，或者外加的负对数似然项。
3. **数据分布**——离线且固定、迭代刷新、完全同策略，或者主动探索，以及各自所隐含的覆盖性要求（表 8.4）。

使这不止于一套分类学的论断是：已知的失效模式来自特定的、可预测的组合——长度作弊来自没有归一化的词元求和式间隔；似然位移来自作用在低多样性样本对上的纯差值目标；模式坍缩来自小 β 下的反向 KL。本章余下部分因此按病理来组织。

8.4 按病理编排的目录

8.4.1 带噪标签

偏好数据的自洽率在 60% 到 80% 之间（第5章）。逻辑斯谛连接函数是无界的，因此一个翻转的标签不只是没有信息量：它会持续不断地提供一个方向错误的大梯度。两种常被混为一谈的修正，实际上是相反的。

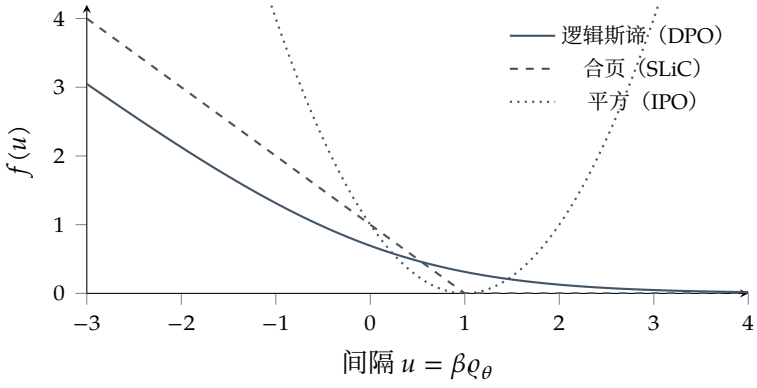


图 8.2: 表 8.1中的三种连接函数。逻辑斯谛损失严格递减，因此梯度下降有动机把间隔无限推大；合页损失在 $u = 1$ 之后就关闭；平方损失则会对超出其目标的间隔主动施加惩罚。过度优化的表现由这条右尾决定，多样性坍缩也遵循同样的次序。图中曲线是这些函数的精确图像，不是实验数据。

保守 DPO 用一个假定的翻转概率 ε 平滑目标，

$$\mathcal{L}_{\text{cDPO}} = -\big[(1-\varepsilon)\log\sigma(h_\theta) + \varepsilon\log\sigma(-h_\theta)\big], \tag{8.12}$$

这就是普通的标签平滑，它给可达到的间隔加了上限：(8.12)的最小值点满足 $\sigma(h_\theta) = 1-\varepsilon$ 而非 1。鲁棒 DPO (Chowdhury et al., 2024) 走的是 去偏路线，对 $\varepsilon < 1/2$ 构造无噪风险的无偏估计量：

$$\mathcal{L}_{\text{rDPO}} = \frac{(1-\varepsilon)\,\ell_{\text{DPO}}(x, y^+, y^-) - \varepsilon\,\ell_{\text{DPO}}(x, y^-, y^+)}{1-2\varepsilon}. \tag{8.13}$$

注意那个减号：rDPO 是朝远离标签交换后的损失的方向作外推，因此方差比 cDPO 更大，但没有渐近偏差。该选哪一个，取决于你的标注噪声是否大致对称，以及你能否估计 ε；实践中 cDPO 配 $\varepsilon \in [0.05, 0.1]$ 是更安全的默认选择，而当 ε 已由一次重标注研究给出时，rDPO 才是正确选项。

8.4.2 无界的连接函数

Azar et al. (2024) 首次清晰地指出，DPO 的弱点在于逻辑斯谛连接函数的无界性，并且 Bradley–Terry 假设本身是可有可无的。他们的 ΨPO 族优化偏好概率的一般非降函数；取恒等映射便得到恒等偏好优化，即围绕一个目标间隔的平方损失，

$$\mathcal{L}_{\text{IPO}} = \Big(\log\frac{\pi_\theta(y^+\mid x)}{\pi_{\text{ref}}(y^+\mid x)} - \log\frac{\pi_\theta(y^-\mid x)}{\pi_{\text{ref}}(y^-\mid x)} - \frac{1}{2\tau}\Big)^2. \tag{8.14}$$

由于损失在越过目标之后重新上升，IPO 无法通过把间隔吹大来降低损失——这恰恰堵死了过度优化的通道。代价是它的样本效率更低：已经排对的样本对仍会继续收到梯度，把间隔往下拉向 $1/2\tau$。

8.4.3 无法配对的数据

成对数据很贵。来自生产流量的点赞/点踩信号则很充裕，而它们并不成对：同一个提示很少同时收到两种信号。KTO (Ethayarajh et al., 2024) 把 Kahneman–Tversky 前景理论改造过来，以利用未配对的二元标签，

$$\mathcal{L}_{\text{KTO}} = -\lambda_w\,\sigma\big(\hat{r}_\theta(x,y) - z_{\text{ref}}\big)\,\mathbb{1}[y \text{ desirable}] + \lambda_l\,\sigma\big(z_{\text{ref}} - \hat{r}_\theta(x,y)\big)\,\mathbb{1}[y \text{ undesirable}], \tag{8.15}$$

其中 $z_{\text{ref}} = \mathbb{E}_{\mathcal{D}}[\beta\mathbb{D}_{\text{KL}}(\pi_\theta(\cdot\,|\,x)\,\|\,\pi_{\text{ref}}(\cdot\,|\,x))]$ 扮演前景理论中参考点的角色，实践中由当前批次在线估计。不对称性 $\lambda_w \neq \lambda_l$ 编码了损失厌恶，令 $\lambda_l > \lambda_w$ 会让不合意的样本权重更高——这与一个已部署助手的经济学相吻合：一次糟糕回复的代价，高过一次良好回复所带来的收益。

Jung et al. (2024) 指出了(8.15)含糊带过的一个分布偏移问题：如果带合意标签的提示与带不合意标签的提示在系统性上有差别，那么该目标学到的将是提示分布，而不是回复质量。二元分类器优化用一个学出来的奖励平移量 δ 以及一个来自区分两个提示池的分类器的重要性权重 $p_\psi(f{=}1\,|\,x)/p_\psi(f{=}0\,|\,x)$ 修正这一点，并证明平移后的梯度与 DPO 的梯度相匹配。如果你是在生产反馈上训练，这项修正并非可选。

8.4.4 丢掉参考模型

用 `bf16` 常驻一份冻结的 70B 模型副本要占 140 GB。有若干方法把它消掉了，需要说清楚的是它们做的并不是同一件事。

ORPO (Hong et al., 2024a) 把监督微调与一个几率比偏好项融合在一起，从而把 SFT 阶段吸收进来：

$$\mathcal{L}_{\text{ORPO}} = -\log p_\theta(y^+\,|\,x) - \lambda \log \sigma\Big(\log \frac{p_\theta(y^+\,|\,x)}{1-p_\theta(y^+\,|\,x)} - \log \frac{p_\theta(y^-\,|\,x)}{1-p_\theta(y^-\,|\,x)}\Big), \tag{8.16}$$

其中长度归一化的序列概率为 $p_\theta(y\,|\,x) = \exp(\frac{1}{|y|}\log \pi_\theta(y\,|\,x))$。几率比是一个有界变换，这使得偏好项不像原始对数比值那样容易失控；而领头的负对数似然项把模型锚定在被选中的回复上——这是对定理 8.2的一个显式修补。

SimPO (Meng et al., 2024) 既无参考模型又做长度归一化，并在 Bradley–Terry 模型中插入了一个目标间隔 γ（即经典配对比较分析中的「主场优势」）：

$$\mathcal{L}_{\text{SimPO}} = -\log \sigma\Big(\frac{\beta}{|y^+|}\log \pi_\theta(y^+ \mid x) - \frac{\beta}{|y^-|}\log \pi_\theta(y^- \mid x) - \gamma\Big). \tag{8.17}$$

其立论所依据的诊断是该论文中最有用的一句话，值得写进教材：在 DPO 之下，就平均逐词元对数似然而言，最终只有大约一半的训练三元组满足 $\bar{p}_\theta(y^+ \mid x) > \bar{p}_\theta(y^- \mid x)$——而这恰恰是真正支配束搜索与长度归一化解码的那个量。训练目标与生成时的度量测的是两回事。SimPO 的修补是让隐式奖励就是生成时的度量。论文报告的可用区间为 $\beta \in [2, 10]$ 与 $\gamma/\beta \in [0.3, 1.6]$；注意这里的 β 与 DPO 的 β 不可比，因为它乘的是逐词元的量而非逐序列的量。

CPO (Xu et al., 2024c)、RRHF (Yuan et al., 2023b) 与 SLiC-HF (Zhao et al., 2023b) 构成第三组：它们各自丢掉 π_{ref}（CPO 用一个均匀先验来近似它），并各自重新加回一个显式的 $-\lambda \log \pi_\theta(y^+ \mid x)$ 行为克隆项，主要差别在于连接函数（逻辑斯谛、零间隔合页、间隔为 δ 的合页）以及是否对对数概率作长度归一化。

工程实践：实际该选哪一个

在经过筛选的偏好对上做第一次对齐训练时，用 DPO 配 $\beta = 0.1$，并在被选中回复上加一个系数为 0.1–1.0 的 NLL 项；这就是 DPO 加上针对定理 8.2的已知最廉价的缓解手段，也是多数优秀开源配方最终收敛到的做法。如果你的偏好对来自一个有长度偏置的评判模型，就用 SimPO，或者加上 LD-DPO 的解耦。如果你有未配对的生产反馈，就用 KTO 配 BCO 的重要性权重。如果你已逼近算力上限、无法再常驻一个参考模型，就用 ORPO。只有在你已经测出过度优化之后，才去考虑 IPO 或 χPO，而不要提前。

8.5 似然位移

这是本章最重要的一种病理，因为它违反直觉、在训练损失中不可见，而且其后果可能把整次训练的意图完全颠倒过来。

8.5.1 现象

在 DPO 训练过程中，$\log \pi_\theta(y^+ \mid x)$——即标注者偏好的那个回复的对数似然——通常会下降，而且往往降幅可观。这不是谁的实现里的 bug，而是定理 8.2在发挥作用：只要间隔变大，损失就会下降，而把被拒绝的回复压得比被选中的回复更快，是一个完

全合格的下降方向。Pal et al. (2024) 证明了它确实可能发生，并表明当 y^+ 与 y^- 的编辑距离很小时最为严重——而这恰恰是精心构造样本对所产生的情形。

8.5.2 机制

Razin et al. (2025) 给出了最权威的解释，而使它可教的正是其表述方式：概率质量并没有消失，它是被位移了，而它去往何处，由模型隐表示的几何结构决定。他们那个招牌例子值得完整叙述，因为它太反直觉了。训练一个模型去偏好单个词元 `No` 胜过单个词元 `Never`，会使 `Yes` 的概率大幅上升。梯度压低 `Never`；而 `No` 与 `Never` 诱导出相似的隐状态；于是抑制 `Never` 的那次更新同时也抑制了 `No`，被释放出来的质量便落到了表示空间中最邻近的第三个回复上——对于一个是非问句而言，那就是语义上的反面。

他们用中心化隐嵌入相似度（centered hidden embedding similarity, CHES）分数来量化这一风险，该分数度量 y^+ 与 y^- 所诱导的隐嵌入之间的相似度，并证明位移正是由高 CHES 的样本对所驱动。他们报告的安全后果是整个直接对齐文献中最令人警觉的一个数字：用 DPO 把 `Llama-3-8B-Instruct` 对齐到拒答不安全提示，反而使其拒答率从 74.4% 降到了 33.4%，原因就是质量从被偏好的拒答上位移到了有害补全上。过滤掉高 CHES 的样本对可以恢复预期行为。

常见陷阱: 你的安全对齐可能正在做与你设想相反的事

上面这种失效具有最糟糕的特征：训练损失单调下降，隐式奖励准确率上升，而模型在被训练的那个行为上可测量地变差。损失曲线和间隔曲线都检测不到它。唯一可靠的检测手段是（i）把 $\log \pi_\theta(y^+ \mid x) - \log \pi_{\text{ref}}(y^+ \mid x)$ 与间隔分开记录，以及（ii）在每个检查点上直接在留出提示上评估目标行为。如果你从本章只带走一条运维教训，就带走这一条。

8.5.3 佐证性测量

三条互相独立的证据线从不同角度说着同一件事，合在一起足以把这一论断放进「已复现」这一层。

Chen et al. (2024a) 测量排序准确率——即最终模型给 y^+ 分配更高概率的训练对占比——发现经 DPO 调优的模型大约只有 60%，接近随机，与同样数据上一个理想化模型所能达到的准确率之间存在很大差距。这个目标函数是一个排序目标，而训练出来的模型不会排序。SimPO 的诊断（节 8.4）就支配解码的长度归一化次序报告了类似

的 ~50% 的数字。Shi et al. (2024a) 表明位移随训练增长并预示生成质量的退化，从而使其在第5章的意义上成为一种过度优化现象，而不是一时的瞬态。

2026 年的细化对实践最为有用。Im et al. (2026) 把这一效应定位下来：似然变化的绝大部分集中在少量离群词元上。这解开了一个表面上的悖论——生成质量怎么能在被选中回复的似然下降的同时反而提升？——因为总的对数概率被少数几个正在重新分配概率质量的位置所主导，而在合理续写上的分布主体其实在改善。他们表明，一个维持总的长度归一化概率的正则项可以消除这一权衡，并报告在 `Llama-3.1-8B-Instruct` 上 AlpacaEval 2 相对提升超过 20%、通用基准上超过 9%。这些幅度应当视为仅有单篇证据；但其机制是有充分支持的。

8.5.4 缓解手段

其中形式最干净的是 DPOP (Pal et al., 2024)，它加了一个惩罚项：只要被选中回复的似然仍高于参考模型，该项就恰好为零；一旦落到参考之下，它就增长：

$$\mathcal{L}_{\text{DPOP}} = -\log \sigma\left(\beta\left[\varrho_\theta - \lambda \max\left(0,\ \log \frac{\pi_{\text{ref}}(y^+ \mid x)}{\pi_\theta(y^+ \mid x)}\right)\right]\right). \tag{8.18}$$

它也可以解释为一个带样本相关间隔 $m = \log 1/\pi_{\text{ref}}(y^+ \mid x)$ 的对比损失，这就说明了为什么它恰恰在参考模型本来就不确定的那些样本对上最为活跃。

8.6 长度

直接对齐算法会让回复变长，有时相当夸张。其机制有两个彼此独立的成分，值得分开来讲，因为它们需要不同的修补。

推导: 两条截然不同的长度通道

通道 *1*（数据）。如果生成 $\mathcal{D}$ 的标注者或评判模型偏好更长的回复——而人类标注者确实如此，任何没有显式长度控制的评判模型也如此——那么 $r^\star$ 就真的与长度相关，任何忠实的算法都会学到这一点。这是一个数据问题，在第4章中处理，没有任何损失函数能修好它。

通道 *2*（目标函数）。即便在长度均衡的数据上，DPO 也有结构性偏置。根据定理 8.3，h_θ 是 $|y^+| + |y^-|$ 个逐词元对数比值之和。考察梯度(8.8)：逐词元的梯度幅度不会随长度缩小，因此一条序列贡献的总梯度随其词元数增长。在 Adam 的逐坐标归一化几何下，这一效应并不是干净的 $|y|$ 因子，但定性结论依然成立：长的

算法 8.1 CHES 过滤的直接偏好优化。过滤这一遍需要 $O(|\mathcal{D}|)$ 次前向传播，而且是值得的：它剔除的正是那些目标函数最可能把概率质量推往非预期方向的样本对。

输入： 偏好集 $\mathcal{D}$、参考模型 π_{ref}、保留比例 q、系数 β

第 1 遍：对样本对的位移风险打分
for $(x, y^+, y^-) \in \mathcal{D}$ **do**
  $e_w \leftarrow \pi_{\text{ref}}$ 在 y^+ 各词元上的平均隐状态
  $e_l \leftarrow \pi_{\text{ref}}$ 在 y^- 各词元上的平均隐状态
  $\bar{e} \leftarrow$ 迄今所见全部嵌入的滑动均值
  $s(x, y^+, y^-) \leftarrow \cos(e_w - \bar{e},\ e_l - \bar{e})$ ▷ CHES
end for
$\mathcal{D}' \leftarrow \mathcal{D}$ 中 s 最低的 q 比例样本
第 2 遍：训练
for 小批量 $B \subset \mathcal{D}'$ **do**
  $h \leftarrow \beta(\log \frac{\pi_\theta(y^+ \mid x)}{\pi_{\text{ref}}(y^+ \mid x)} - \log \frac{\pi_\theta(y^- \mid x)}{\pi_{\text{ref}}(y^- \mid x)})$
  $\mathcal{L} \leftarrow -\log \sigma(h) - \lambda_{\text{nll}} \log \pi_\theta(y^+ \mid x)$
  更新 θ；记录 $\mathbb{E}[h]$、$\mathbb{P}(h > 0)$ 与 $\mathbb{E}[\log \pi_\theta(y^+) - \log \pi_{\text{ref}}(y^+)]$
  if $\mathbb{E}[\log \pi_\theta(y^+) - \log \pi_{\text{ref}}(y^+)] < -\kappa$ **then**
    停止：位移已超过阈值 ▷ 没人会跑的那项检查
  end if
end for
return π_θ

> 被选中回复受到的向上推力总量大于短的，长的被拒绝回复受到的向下推力总量也更大。数据均衡时这两者部分抵消；而当迭代训练中模型自身的生成逐渐变长时，它们不再抵消。

四类修补对应这两条通道。

在奖励中惩罚长度。R-DPO (Park et al., 2024) 取 $r(x, y) - \alpha\,|y|$ 作为目标奖励，这会传导为间隔的一个平移：

$$\mathcal{L}_{\text{R-DPO}} = -\log \sigma\Big(\beta \varrho_\theta - \alpha(|y^+| - |y^-|)\Big). \tag{8.19}$$

按长度归一化奖励。 SimPO (8.17)除以 $|y|$，使隐式奖励成为平均逐词元对数概率。这彻底关闭了通道 2，同时也让训练目标与长度归一化的解码对齐，代价是丢掉了(8.5)的序列级解释。

把冗长与内容解耦。 LD-DPO (Liu et al., 2024e) 固定一个「公共长度」$l_p = \min(|y^+|, |y^-|)$，并把超出它的每个词元按 $\alpha \in [0,1]$ 降权：

$$\hat{\pi}_\theta(y \mid x) = \prod_{t=1}^{l_p} \pi_\theta(y_t \mid x, y_{<t}) \prod_{t=l_p+1}^{|y|} \pi_\theta(y_t \mid x, y_{<t})^{\alpha}, \tag{8.20}$$

然后在 $\hat{\pi}_\theta$ 上跑 DPO。$\alpha = 1$ 时这就是 DPO；$\alpha = 0$ 时超出的长度完全不作贡献。这一构造之所以有吸引力，是因为它是一个语义清晰的严格插值：只有两个回复中长度可比的那部分才携带偏好信号。

拉平词元数。 SamPO (Lu et al., 2024) 从 y^+ 与 y^- 中下采样相同数量 T_m 的词元位置，只在这些位置上计算隐式奖励，以额外的梯度方差为代价，机械地消除了「词元越多、可推的 logit 越多」这一不对称。

历史注记: 一个传播开来的符号错误

SimPO 论文中被广泛转载的对比表把 R-DPO 的间隔平移印成了 $+\alpha(|y^+| - |y^-|)$。Park et al. (2024) 的式 (9) 带的是减号，这也正是该文自己的奖励定义 $r - \alpha|y|$ 所蕴含的：更长的被选中回复应当让目标更难满足，而不是更容易。请在(8.19)中使用减号。这段插曲小中见大地折射出这一文献的一个大问题——对比表在论文之间被复制的频率，远高于被重新推导的频率，而符号约定是第一个牺牲品。

代码 8.2: 把长度感知的各种变体实现为同一个打分函数的不同修改。传入 `alpha=1.0` 便退化为普通的 DPO 打分；`per_token` 标志切换到 SimPO 的归一化。把三者放进同一个函数里能让消融实验保持诚实，因为这样填充与掩码逻辑在各个实验组之间是可证明一致的。

```python
def scored_logp(model, ids, attn, loss_mask, public_len=None, alpha=1.0,
                per_token=False):
    logits = model(input_ids=ids, attention_mask=attn).logits[:, :-1]
    lp = torch.log_softmax(logits.float(), -1).gather(
        2, ids[:, 1:].unsqueeze(2)).squeeze(2) * loss_mask[:, 1:]
    if public_len is not None:                       # LD-DPO: 式 (8.24)
        pos = loss_mask[:, 1:].cumsum(-1)
        w = torch.where(pos <= public_len.unsqueeze(1), 1.0, alpha)
        lp = lp * w
    total = lp.sum(-1)
```

```
    if per_token:                                    # SimPO: 式 (8.19)
        total = total / loss_mask[:, 1:].sum(-1).clamp(min=1)
    return total
```

8.7 覆盖性、过度优化与 χPO

现在我们来到对离线形式化最深刻的批评，它同时也是最有用的批评，因为它是建设性的。

8.7.1 反向 KL 是一份很弱的证书

(8.1)中的正则项是反向 KL。「反向 KL 有界」听上去是个让人安心的保证，实则远比看上去要弱。离线强化学习的界由集中度系数支配，它度量数据分布覆盖某个候选策略的失败程度。$\mathbb{D}_{\mathrm{KL}}$ 有界并不能界住集中度：一个策略可以在 KL 意义上接近 π_{ref}，却把不可忽略的质量放在 π_{ref}（因而数据）呈指数稀少的区域上。那恰恰是(8.3)对隐式奖励不加任何约束的区域，也正是 Xu et al. (2024d) 所观察到的「DPO 会找到隐式奖励虚高的分布外回复」背后的机制。

Huang et al. (2024a) 精确地给出了这一论证并随即修好了它。把 KL 生成元换成混合生成元

$$f_{\chi_{\mathrm{mix}}}(z) = \tfrac{1}{2}(z-1)^2 + z\log z, \qquad \phi(z) := f'_{\chi_{\mathrm{mix}}}(z) = z + \log z, \tag{8.21}$$

所得到的直接对齐目标相对 DPO 只是一行改动：把连接函数自变量中的对数换成 ϕ，

$$\mathcal{L}_{\chi\mathrm{PO}} = -\log\sigma\left(\mathrm{clip}\left(\beta\phi\Big(\frac{\pi_\theta(y^+\mid x)}{\pi_{\mathrm{ref}}(y^+\mid x)}\Big) - \beta\phi\Big(\frac{\pi_\theta(y^-\mid x)}{\pi_{\mathrm{ref}}(y^-\mid x)}\Big),\, -2R_{\max},\, 2R_{\max}\right)\right). \tag{8.22}$$

回报在于 $\chi^2(\cdot\|\cdot)$ 确实能控制集中度——单策略系数恰为 $C^\pi = 1 + 2\chi^2(\pi\|\pi_{\mathrm{ref}})$，且 $\mathbb{D}_{\mathrm{KL}} \le 2\chi^2$——因此 χPO 在单策略集中度条件下获得了样本复杂度保证。用大白话说：χPO 可证明地对过度优化鲁棒，而普通的 DPO 不是。

8.7.2 过度优化是可测量的，而且有固定形状

Rafailov et al. (2024a) 把Gao et al. (2023a) 的过度优化标度律搬到直接对齐算法上，发现了同样的定性图景：随着隐式奖励上升，实测质量先上升、到达峰值、然后下降，且有效步长越大，峰值来得越早。直接对齐算法的独特之处在于，这里没有一个显式的奖励模型可供对抗性地过度优化，因此通常那套心智模型——「策略在利用奖励模型的

误差」——必须换成「策略在利用目标函数未加约束的那些区域」。补救办法与第7章 中相同：留出一个验证集，评估生成质量而非损失，并早停。在这一文献中，早停不是一个超参数细节；它是过度优化的首要控制手段。

常见陷阱: β 在不同方法之间不可比

DPO 中的 $\beta = 0.1$、SimPO 中的 $\beta = 5$、IPO 中的 τ^{-1}，名义上控制的是同一个权衡，而它们中没有任何两个处在同一量纲上。DPO 的 β 乘的是一个幅度随长度增长的序列级对数比值；SimPO 的乘的是逐词元平均；IPO 的目标间隔 $1/2\tau$ 设定的是一个绝对终点而非斜率。任何在「同一 β」下比较各方法的表格都什么也没比较。请独立地调节每种方法的正则强度，并把实际达到的 $\mathbb{D}_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}})$——这个量是可比的——与质量一并报告。

8.8 粒度：在样本对之下与之上

DPO 把一个标量偏好施加到整条序列上。更细与更粗的粒度都是可行的。

词元级。TDPO (Zeng et al., 2024b) 加入了一个序列 *KL* 间隔项，其中 $D_{\mathrm{SeqKL}}(x, y)$ 是沿 y 逐词元前向 KL 的求和：

$$\mathcal{L}_{\mathrm{TDPO}_2} = -\log \sigma\big(h_\theta - \alpha[\beta D_{\mathrm{SeqKL}}(x, y^-) - \mathrm{sg}(\beta D_{\mathrm{SeqKL}}(x, y^+))]\big). \tag{8.23}$$

那个非对称的停止梯度才是真正起作用的技巧，值得研究：它让损失得以在被拒绝的回复上 收紧 KL 控制——以一种在别处仍贴近参考模型的方式抑制它——同时不让被选中回复的 KL 项去与偏好信号对抗。论文报告的收益是更好的对齐/多样性前沿，而鉴于节 14.3，这正是该测的东西。

步骤级。Step-DPO (Lai et al., 2024) 在一个推理步骤的粒度上施加 DPO 损失：给定共享的正确前缀 $s_{1:k-1}$，把一个正确的下一步与一个错误的下一步作对比，

$$\mathcal{L}_{\text{Step-DPO}} = -\log \sigma\Big(\beta \log \frac{\pi_\theta(s_{\mathrm{win}} \mid x, s_{1:k-1})}{\pi_{\mathrm{ref}}(s_{\mathrm{win}} \mid x, s_{1:k-1})} - \beta \log \frac{\pi_\theta(s_{\mathrm{lose}} \mid x, s_{1:k-1})}{\pi_{\mathrm{ref}}(s_{\mathrm{lose}} \mid x, s_{1:k-1})}\Big), \tag{8.24}$$

并配有一套从模型自身生成中挖掘首个出错步骤的流水线。这是过程监督在直接对齐中的对应物，而第13章会正式处理它所引出的信用分配问题。直接 Q 函数优化 (Ji et al., 2024) 把同一思路推进到一个显式基于值的形式。

稀疏。 SparsePO (Christopoulou et al., 2024) 学习词元级掩码，使得只有部分位置才会收到偏好梯度，其假设是——这与Im et al. (2026) 的离群词元发现以及节 12.6的分叉词元结果一致——偏好信息集中在少数几个决策点上。

以 *best-of-n* 为目标。在粗粒度这一端，可以把 best-of-n 分布本身当作要模仿的对象，而不是当作要优化的奖励。BOND (Sessa et al., 2024) 用 Jeffreys 散度目标把 best-of-n 蒸馏进策略，使采样成本在训练时一次付清，而不是在推理时付 n 次；BoNBoN (Gui et al., 2024) 则刻画了 best-of-n 分布并微调模型去匹配它。Balashankar et al. (2024) 提出了更尖锐的一点：如果你打算用 best-of-n 来解码，那么标准的 KL 正则目标一开始就是错误的优化对象，并据此推导出感知推理方式的替代目标。节 10.4.1给出了使这些比较有意义的 KL 核算。

8.9 参考策略

π_{ref} 在(8.7)中出现时仿佛是给定的。它其实是一个有真实后果的设计选择。

选哪个模型。 Liu et al. (2024h) 直接研究了这个问题，发现一个 强于 SFT 检查点的参考模型只有在与被调优的模型兼容时才有帮助——把另一个模型家族的检查点用作 π_{ref} 会使结果变差，因为此时对数比值度量的是家族差异，而不是与偏好相关的差异。

要不要更新它。 sDPO (Kim et al., 2024) 把偏好数据切成若干块并顺序地施加 DPO，用上一块对齐得到的模型作为下一块的参考，从而让参考模型成为一个逐步收紧的下界。TR-DPO (Gorbatovski et al., 2024) 则持续更新 π_{ref}，要么用软融合 $\pi_{\text{ref}} \leftarrow \alpha\pi_\theta + (1-\alpha)\pi_{\text{ref}}$，要么每 τ 步硬替换一次。两者都以放弃固定锚点这一保证为代价，换取围绕当前策略的更紧信赖域；两者也都使得最终的 $\mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{sft}})$ 无界——如果你测的是质量，这没问题；如果你把那个 KL 数值当作安全证书来依赖，那就很危险。

自适应正则强度。β-DPO (Wu et al., 2024b) 让 β 在批次层面动态变化，由批次平均隐式奖励间隔 M 驱动：

$$\beta^{\text{batch}} = [1 + \alpha(\mathop{\mathbb{E}}_{i\in\text{batch}}[M_i] - M_0)]\beta^0, \qquad M_0 \leftarrow mM_0 + (1-m)\mathop{\mathbb{E}}_{i\in\text{batch}}[M_i], \quad (8.25)$$

并对 M_i 施加 3σ 过滤以剔除离群值。即使你不采用这个方法，其直觉也值得记住：高间隔的批次要么很简单，要么标错了，应当用大的 β 保守对待；低间隔的批次信息量大，可以配一个更小的 β 来信任它。α-DPO (Wu et al., 2024a) 通过自适应的间隔而非自适应的 β 追求同一目标，作者还证明其目标函数是某个在线 SimPO 目标加上可控松弛项的上界——这实际上使它成为同策略训练的一个离线代理。

8.10 直接对齐与策略优化之比较

「用 DPO 还是 PPO?」这个问题三年来一直问得很糟，因为它把三个可分离的问题捆在了一起。把它们分开，证据就变得可处理了。

8.10.1 经验层面：谁在基准上取胜?

Xu et al. (2024d) 的回答是：在双方都调优的前提下，PPO 决定性地取胜。他们的经验贡献是指出了让 PPO 取胜的要素——优势归一化、大批量、以及指数滑动平均的参考模型更新——并在竞赛级代码任务上报告了当时的最佳结果。Ivison et al. (2024) 在一个控制更严格的设置下得到了相容的结论：PPO 在数学上至多领先 2.5%，在通用能力上至多领先 1.2%，同时强调了一个更要紧的发现：数据质量对算法选择具有主导作用。反向的潮流是真实的，而且主要是经济性的。正如 RewardBench 2 的作者所指出的 (Malik et al., 2025)，在许多实际场景中，据报告更简单的直接对齐算法效果更好，这在很大程度上是因为它们便宜到足以被认真调优。一个你能扫二十次的方法，胜过一个你只跑得起一次的方法。

8.10.2 统计层面：在线–离线鸿沟

Tang et al. (2024a) 检验了「为什么同策略数据有帮助」的各种显而易见的解释——覆盖性、损失函数、模型规模——发现它们都不能完全解释这一鸿沟。真正相关的是采样分布本身的同策略程度。他们的结论——离线算法擅长分类而在生成上较差——是这一领域较为重要的负面结果之一，而且它诚实地承认自己没有给出机制。Tajwar et al. (2024) 给出了最尖锐的建设性版本：偏好微调应当使用次优的、同策略的数据，因为同策略采样与负梯度的结合，才是那个在好数据上做极大似然所无法复制的对比机制的来源。理论上，这一鸿沟表现为表 8.4中的覆盖性层级：离线方法需要全策略集中度，在线与乐观方法只需要单策略集中度。

8.10.3 概念层面：强化学习凭什么应该有帮助?

如果图 8.1中的两条路线指向同一个最优解，那条更长的路线到底买来了什么? Swamy et al. (2025) 给出了最清晰的回答，而这个回答不是统计性的，而是计算性的。两阶段流程的价值在于一个生成–验证鸿沟：判断一个回复好不好，是比生成一个好回复更容易的学习问题，所以把问题经由一个学出来的验证器来分解，会使最终的优化问题更容易求解，尽管两条流程指向的是同一个极大似然解。这一框架是全书的主线。它恰

恰解释了 第11章为什么如此有效——那里的验证器甚至不需要学习，它是精确的——并且预测：当验证困难时直接方法看起来相对更好，当验证容易时相对更差。

核心思想: 真正该问的不是哪个算法，而是哪种分解

直接对齐消去了奖励模型。这对成本而言毫无疑问是好事，对优化而言毫无疑问是坏事，因为它丢弃了那个让两阶段流程变容易的生成–验证鸿沟。其后果是一个干净的预测，并且在Chapters 11 and 17中得到了印证：验证器越便宜、越准确，直接方法的理据就越弱；在存在免费且精确的验证器的极限下，这一理据消失。

8.10.4 一则相关的理论警示

2026 年的两项结果，为「我们已经完全理解了这件事」的任何自信降了温。Young (2026) 通过对序列级危害作鞅分解证明：标准对齐目标在位置 t 处的梯度，等于条件期望危害与得分函数之间的协方差——因此在结果实际上已经确定的位置之后，梯度为零。这为一个众所周知的经验事实（对齐模型相对基座模型的 KL 集中在最初几十个词元上）提供了理论解释，并蕴含着：无论优化质量如何，标准目标都不可能产生深层对齐。第25章会承接其后果。另外，Wu et al. (2026) 给出了一般 f 散度正则下在线 RLHF 的第一个统一理论，遗憾界为 $O(\log T)$；其实践含义是：散度的选择是一项一等的设计决策，而不是脚注。

8.11 一份可用的配方

工程实践: 7B–8B 模型在 6 万偏好对上的起始配置

学习率 5×10^{-7}，AdamW，$\beta_1 = 0.9$，$\beta_2 = 0.95$，权重衰减 0，余弦衰减至峰值的 10%，预热占总步数的 10%。全局批量 64 个偏好对（128 条序列）。训练一到两个 epoch；再多的话，质量会先于损失开始下降。$\beta = 0.1$；在长度归一化的被选中对数概率上加辅助 NLL 项，权重 $\lambda = 0.2$。提示最大长度 1024，回复最大长度 2048，超出者一律丢弃而非截断，因为一个被截断的被选中回复会教模型在句子中途停下。用 `bf16` 计算 $\log \pi_{\text{ref}}$，但把逐词元的 log-softmax 以 `fp32` 累加：间隔是两个大数求和之差，也是精度损失最先显形的地方。

学习率是最先该调的超参数——比上述区间高一个数量级，就会产生那个经典特征：被选中似然坍缩，而损失曲线漂亮地下降。β 排第二：如果 $\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})$ 超过每词元几个 nat，就调高它；如果间隔几乎不动，就调低它。

算法 8.2 离线直接偏好优化，附带那些让它变得可信的检查。第 3–5 步是多数实现所省略的部分，也正是决定这次训练是否在做你所设想之事的部分。

输入： SFT 检查点 π_{sft}、偏好集 $\mathcal{D}$、β、NLL 权重 λ

$\pi_{\text{ref}} \leftarrow \pi_{\text{sft}}$（冻结）；$\theta \leftarrow \theta_{\text{sft}}$
数据准备
去重；丢弃 $y^+ = y^-$ 的样本对，以及任一侧被截断的样本对
丢弃 CHES 最高的那部分样本对 ▷ 算法 8.1
报告 y^+ 与 y^- 的长度分布；若二者不同，选择一个长度感知的目标函数
训练
for epoch = 1 **to** E（通常为 1–3） **do**
  for 小批量 $B \subset \mathcal{D}$ **do**
    $h \leftarrow \beta\big((\log \pi_\theta(y^+) - \log \pi_{\text{ref}}(y^+)) - (\log \pi_\theta(y^-) - \log \pi_{\text{ref}}(y^-))\big)$
    $\mathcal{L} \leftarrow -\mathbb{E}_B \log \sigma(h) - \lambda\ \mathbb{E}_B \log \pi_\theta(y^+ \mid x) / |y^+|$
    用 AdamW、余弦调度、约 10% 的预热更新 θ
  end for
  每轮验证——不可省略
  在留出提示上、按部署时的解码设置评估生成质量
  记录 $\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})$ 与平均回复长度
  if 质量较上一轮下降 **then** 回退并停止
end for
return 验证最优的检查点，而不是最后一个

8.12 2026 年直接对齐所处的位置

一份诚实的总结，并标出各条的证据等级。

直接对齐算法在偏好对齐上赢得了部署之争。几乎每一份开源后训练配方都在「有用性与风格」阶段使用某个 DPO 族目标，因为它与「PPO 加评论家」的成本差距达到一个数量级，而按Ivison et al. (2024)，质量差距相对于数据质量的影响而言很小。这是一个复现良好的实践结论。

它们输掉了能力训练之争。本章中没有任何内容能处理可验证奖励、环境或多轮交互，其原因是结构性的而非偶然的：整套构造的起点就是对同一提示下两个完整回

复的成对比较。凡是存在廉价且精确的验证器之处，第11章严格更优，而Swamy et al. (2025) 的生成–验证论证解释了原因。

离线这一限制是真正起约束作用的局限，而本领域基本上已经接受了这一点。迭代——从当前策略采样、打标、施加目标、重复——能挽回在线–离线鸿沟的大部分，而认真的工作也都投在了这里。那就是第9章。

这些失效模式已被理解，但仪表化不足。似然位移、长度膨胀与多样性坍缩，其机制与缓解手段都已知，而且缓解手段很便宜。它们却仍被例行地省略，因为要让损失曲线好看，其中任何一项都不必要。一位从业者能从本章带走的最有价值的东西不是某个算法，而是表 8.5。

最后，关于理论的两点保留。DPO 与 RLHF 的等价性是有条件的，而这一条件直到 2026 年才被刻画清楚 (Yang et al., 2026b)；并且以这种方式取得的对齐可证明是浅层的 (Young, 2026)。两项结果都很新且都仅有单篇证据，但它们指向同一个令人不安的方向：我们一直在优化的那个目标，可能是我们真正想要的目标的一个系统性不完美的代理，而且这种不完美无法归结为奖励模型的误差。

本章小结

- 带 KL 正则的最优解(8.2)是可逆的，所以奖励与策略至多相差一个只依赖提示的常数、本质上是同一个对象；Bradley–Terry 似然只依赖奖励之差，因此那个常数精确相消，奖励模型可以在不作任何近似的情况下被消去。
- DPO 不是强化学习。它是在一个重参数化之上的极大似然，其强化学习成分完全存在于推导之中。
- 由于损失只依赖间隔 h_θ，没有任何机制阻止被偏好回复的似然下滑。这就是似然位移：概率质量被位移到了表示上相似的第三个回复上，而它可能把一次安全对齐训练的意图完全颠倒 (Razin et al., 2025)。
- 由于间隔是词元上未经归一化的求和，直接对齐带有一种区别于数据中长度偏置的结构性长度偏置，二者需要不同的修补。
- 每一个离线目标都是作用在同一个标量上的凸分类损失 (Tang et al., 2024b)；连接函数在零点的曲率设定了隐式 KL 惩罚，所以无参考方法是有正则的，只是更难控制。连接函数的右尾则决定了过度优化是否可能发生。
- 反向 KL 有界并不能界住集中度，所以离线直接对齐没有任何过度优化保证；把散度换成混合 χ^2 生成元只是一行改动，却换来单策略集中度 (Huang et al., 2024a)。

- β 在不同方法之间不可比。请改为报告实际达到的 $\mathbb{D}_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}})$。
- 在双方都调优的前提下，配有调优评论家的 PPO 胜过 DPO (Xu et al., 2024d; Ivison et al., 2024)，但数据质量对算法选择起主导作用，而实践中的赢家通常是那个你扫得起参数的方法。
- 「先奖励模型、再强化学习」这一两阶段流程的价值是计算性的，而非统计性的：它利用了生成–验证鸿沟 (Swamy et al., 2025)。这预测了：验证越便宜，直接方法就越失势。
- 训练损失检测不到本章任何一种失效模式。能检测到的是留出集上的生成质量，以及被选中回复的对数比值。

延伸阅读

先读Rafailov et al. (2023)，并在读推导之前自己把(8.7)推一遍；这只需十五分钟，而它是后训练中最干净的一段数学。接着读Rafailov et al. (2024b)，它把隐式奖励从一个序列级分数升格为词元级 Q 函数，从而把本章与第13章连了起来。然后读Azar et al. (2024)，不是为了 IPO 本身，而是为了 ΨPO 这一框架——它是第一篇明确指出 Bradley–Terry 假设可有可无、且逻辑斯谛连接函数的无界性才是问题所在的论文。

就设计空间而言，Tang et al. (2024b) 是离线文献中最经济的一篇论文：它的四页内容顶得上十几篇单独的变体论文，而关于隐式 KL 惩罚的泰勒展开论证，正是学生最常提出的那个概念性问题的答案。把它与Raheja and Pochhi (2026) 的三轴粗粒度地图配着读。如果关于「哪里会出错」你只读一篇，那就读Razin et al. (2025)：`No` 与 `Never` 的例子令人过目不忘，而那个拒答率结果应当改变你验证一次安全训练的方式。Im et al. (2026) 是关于位移与生成改善为何能并存的当前最佳解释。

就理论而言，Huang et al. (2024a) 读起来吃力但值回票价，因为它是罕见的这样一篇论文：它指出了一处概念缺口（KL 是覆盖性保证的错误散度），并用一行就能实现的改动把缺口补上。关于与策略优化的比较，请把Xu et al. (2024d) 与Ivison et al. (2024) 对照着读——它们在幅度上分歧、在方向上一致——然后读Tang et al. (2024a)，看那个关于「同策略数据为何有帮助」的诚实负面结果。最后以Swamy et al. (2025) 收尾，它是本章参考文献中概念上最重要的一篇，其框架也是全书余下部分中复现最频繁的。

有意推迟的内容：在线与迭代变体、自奖励方案，以及彻底抛弃 Bradley–Terry 假设的 Nash ／博弈论形式化，全部归于第9章。多样性坍缩在 节 14.3中作定量处理，过程级信用分配在第13章中处理。

习题

8.1 不参考正文，从(8.2)与(8.4)推导出(8.7)。然后精确说明：如果(8.1)中的正则项换成前向 KL，即 $\mathbb{D}_{\mathrm{KL}}(\pi_{\mathrm{ref}} \| \pi_\theta)$，而不是反向 KL，哪一步会失效。

8.2 数值验证定理 8.2。取一个只有两个回复的玩具模型，设 $\log \pi_\theta(y^+) = a$、$\log \pi_\theta(y^-) = b$，参考值为 a_0, b_0。在 (a, b) 平面上画出 $\mathcal{L}_{\mathrm{DPO}}$ 的等值线，并标出损失下降而同时 $a < a_0$ 的区域。在一个典型初始化点上，有多大比例的下降方向落在该区域内?

8.3 证明 cDPO 损失(8.12)的最小值点满足 $\sigma(h_\theta) = 1 - \varepsilon$，从而 cDPO 把可达到的间隔封顶在 $\log \frac{1-\varepsilon}{\varepsilon}$。再与 rDPO (8.13)作对比：证明当标签以概率 ε 翻转时，rDPO 是无噪 DPO 风险的无偏估计量，并计算 $\varepsilon \to 1/2$ 时的方差放大因子。

8.4 利用(8.11)，为表 8.1中三个未命名的连接函数计算 $f''(0)/2\,|f'(0)|$，并验证表中所给的数值。然后解释：为什么合页损失在原点处的隐式 KL 惩罚为零，实践中却仍不会从 π_{ref} 任意漂移开去。

8.5 一位同事报告说，他的 DPO 训练损失单调下降，在训练对上的隐式准确率为 0.85，而留出集胜率却从 62% 掉到了 41%。请给出三个与这些观测相容且彼此不同的假设，并为每个假设指出能确认或排除它的那项最廉价的单一测量。

8.6 节 8.6论证 DPO 的长度偏置是结构性的。请构造一个长度均衡的偏好数据集（被选中与被拒绝的长度分布相同），并给出理由预测：DPO 在其上是否仍会推高生成长度。然后预测在自生成偏好对上迭代训练三轮之后会发生什么，并解释其中的差别。

8.7 证明在 $\alpha = 0$ 时，LD-DPO 重整形后的似然(8.20)使目标函数对较短回复长度之后的一切都保持不变，并给出一个具体的偏好对例子，说明这种不变性会摧毁标注者本来想传达的信号。

8.8 ⋆ 定理 8.4说每个离线目标都自带一个强度为 $\beta f''(0)/2\,|f'(0)|$ 的隐式 KL 惩罚。请设计一个连接函数 f，使其隐式惩罚是状态相关的——在 π_{ref} 自信的提示上更强，在

它不确定的提示上更弱——同时保持凸且递减。请依据表 8.4论证：这样的目标能否在不引入 (8.22)中显式 χ^2 项的情况下达到单策略集中度；或者证明它不能。

8.9 ⋆ Young (2026) 证明，在结果实际上已经确定的位置之后，对齐梯度会消失。请取 Step-DPO 目标(8.24)，判断在步骤粒度上施加损失究竟是规避了这一结果，还是仅仅把它挪了位置。请陈述你关于「危害」或「正确性」在一个步骤内何处被确定的假设，并指出你的结论对哪个假设最为敏感。

8.10 † 实现算法 8.1，用一个 8B 模型在固定的 2 万偏好对数据集上跑四组实验：纯 DPO、带 NLL 锚定的 DPO、在 $q = 0.8$ 下做 CHES 过滤的 DPO，以及两者兼有。对每一组，每 50 步记录表 8.5中的每一个量。报告：（i）各组中被选中回复的对数比值是否转为负值；（ii）留出集生成质量是否在损失进入平台之前就已见顶；（iii）跨检查点的被选中对数比值与胜率之间的相关性。请明确指出你的哪些结论落在噪声范围之内。

8.11 † 通过代码 8.2中共享的打分函数来实现 DPO、IPO 与 SimPO，使掩码与填充在各组之间可证明一致。把每种方法的正则强度在五个取值上扫一遍，并把质量对实际达到的 $\mathbb{D}_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{ref}})$ 作图，而不是对名义上的 β 作图。讨论所得的前沿是否分离，以及图 8.2对大 KL 处的次序作出了什么预测。

表 8.2: 离线成对目标函数，按运维上真正要紧的三项设计选择组织。「参考模型」是显存问题；「长度」指目标函数是否对回复长度不变；「锚定」指是否有任何机制阻止 $\log \pi_\theta(y^+ \mid x)$ 下滑（定理 8.2）。这张表不构成任何排名：这些方法处在偏差–方差–成本前沿上的不同位置，而且按Ivison et al. (2024)，数据质量对它们之间的取舍起主导作用。

方法	参考模型	长度	锚定	区分性机制
DPO (Rafailov et al., 2023)	是	否	否	对数比值之差上的逻辑斯谛连接
IPO (Azar et al., 2024)	是	否	否	朝目标间隔的平方损失；有界
cDPO	是	否	否	标签平滑给间隔封顶
rDPO (Chowdhury et al., 2024)	是	否	否	已知翻转率下的无偏去偏
KTO (Ethayarajh et al., 2024)	是	否	否	未配对二元标签；损失厌恶
BCO (Jung et al., 2024)	是	否	否	奖励平移 + 提示池重要性权重
ORPO (Hong et al., 2024a)	**否**	是	**是**	几率比 + NLL；取代 SFT
SimPO (Meng et al., 2024)	**否**	**是**	否	奖励就是解码度量
CPO (Xu et al., 2024c)	**否**	否	**是**	均匀参考 + NLL
RRHF (Yuan et al., 2023b)	**否**	是	**是**	零间隔排序合页 + NLL
SLiC-HF (Zhao et al., 2023b)	**否**	否	**是**	间隔合页 + 参考 NLL
R-DPO (Park et al., 2024)	是	显式	否	奖励中的长度惩罚
DPOP (Pal et al., 2024)	是	否	**是**	在参考处设置单边下限
TDPO (Zeng et al., 2024b)	是	词元	否	序列 KL 间隔，非对称 sg
LD-DPO (Liu et al., 2024e)	是	**是**	否	对超出 $\min(\lvert y^+\rvert, \lvert y^-\rvert)$ 的词元降权
SamPO (Lu et al., 2024)	是	**是**	否	等数量词元子采样
χPO (Huang et al., 2024a)	是	否	否	$\phi(z) = z + \log z$；χ^2 覆盖性保证
β-DPO (Wu et al., 2024b)	是	否	否	由间隔统计量自适应调整批次级 β
Cal-DPO (Xiao et al., 2024)	是	否	**是**	校准隐式奖励的尺度

表 8.3: 似然位移的缓解手段，按对目标函数改动幅度由小到大排列。前三项基本上是免费的，应当视为默认配置；后两项需要额外的机制。它们全都通过让间隔之外的某个东西也变得重要，来应对定理 8.2。

手段	作用位置	机制与代价
在 y^+ 上加 NLL 项	损失	CPO/ORPO/迭代式 RPO 都这么做；提供一个绝对锚点。系数过大就退化成纯做 SFT。
单边下限（DPOP）	损失	惩罚仅在 $\pi_\theta(y^+) < \pi_{\text{ref}}(y^+)$ 之后才激活；在下限之上零代价。
长度归一化	损失	SimPO、LD-DPO、SamPO；消除那条使位移在长的被选中回复上更容易发生的长度通道。
过滤高 CHES 样本对	数据	Razin et al. (2025)；每对需要一次前向传播来取隐状态。最有针对性的修补。
构造差异明确的样本对	数据	CLAIR (D'Oosterlinck et al., 2024) 把 y^- 生成为 y^+ 的最小改写，使偏好只隔离出一个维度——注意这会抬高 CHES，所以要与锚定项配合使用。
校准奖励尺度	损失	Cal-DPO (Xiao et al., 2024)；把隐式奖励的尺度而不只是次序对齐到一个目标上。

表 8.4: 覆盖性层级。「要求」一列是数据分布必须满足什么，相应的保证才成立，而它是预测一个方法会不会过度优化的最佳单一指标。全策略集中度是一个非常强的假设：它要求离线数据覆盖策略类中的每一个策略，包括那些你根本不会部署的策略。

设定	要求	文献
纯离线直接对齐算法（DPO、IPO、SimPO 等）	全策略集中度	Huang et al. (2024a)
带混合 χ^2 正则的离线方法	单策略 $C^{\pi^\star}$	Huang et al. (2024a)
基于悲观主义的离线 RLHF	单策略，经由悲观主义	Zhu et al. (2023)
KL 约束下的迭代在线 DPO	覆盖各次迭代	Xiong et al. (2023)
带乐观探索的在线方法（XPO）	对离线数据无要求；可实现性	Xie et al. (2025)

表 8.5: 直接对齐训练过程中该记录什么，以及每个量能检测到什么。前三个是标准配置；后四个才是能捕捉本章所讲那些失效的量，而略去它们正是一次坏掉的训练看起来很健康的最常见原因。

量	检测什么	健康表现
损失	单看它什么也检测不到	下降；无信息量
间隔 $\mathbb{E}[h_\theta]$	目标函数是否正在被优化	先上升，随后趋于平台
隐式准确率 $\mathbb{P}(h_\theta > 0)$	模型是否把训练对排对	上升至 0.7–0.9
$\mathbb{E}[\log \pi_\theta(y^+) - \log \pi_{\text{ref}}(y^+)]$	**似然位移**	保持 ≥ 0，或轻微为负
$\mathbb{E}[\log \pi_\theta(y^-) - \log \pi_{\text{ref}}(y^-)]$	是否靠压制在做功	下降；其下降幅度应小于被选中项的上升幅度
留出提示上的 $\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})$	漂移；不同方法之间的可比性	每条序列几个 nat
平均生成长度	长度作弊	保持平稳，或可由数据解释
留出集生成胜率	损失看不见的一切	唯一值得据以停止训练的量

第 9 章

在线、迭代与博弈论式对齐

核心思想: 直接对齐真正的约束在于采样分布，而不在于损失函数

第8章结束于一个令人不安的经验事实：在目标函数固定的前提下，同策略数据胜过离线数据，而现有的各种解释——覆盖性、损失形状、模型规模——都无法完全解释这一鸿沟。本章就是本领域对此的回应。它分为两半。前半保留 Bradley–Terry 假设，改用迭代来修补数据：从当前策略采样、打标、优化、重复——并可选地加上一个探索奖励，把一个启发式做法变成一个带覆盖性保证的算法。后半则彻底抛弃 Bradley–Terry。如果偏好不必由一个标量奖励生成，那么正确的解概念就不是 argmax，而是一个两人偏好博弈的纳什均衡，由此得到的算法都是自博弈。

离线目标(8.7)是在一个固定数据集上的期望。它对世界的全部认识都来自该数据集，而(8.3)在其余各处都不对隐式奖励施加任何约束。显而易见的修补是：随着策略变化，持续不断地从策略中收集数据。这个修补如此显而易见，以至于有趣的问题不是要不要这么做，而是它究竟买来了什么；而答案比「数据越多越好」要尖锐得多。

9.1 为什么采样分布至关重要

9.1.1 统计层面的陈述

纯统计的论证关乎覆盖性，表 8.4已经给出了这一层级。一个纯离线的直接对齐算法需要全策略集中度：数据必须充分覆盖优化器可能抵达的每一个策略，因为该目标函数对它从未观测到的行为不提供任何信息。带乐观探索的在线方法则对离线数据完全不需要覆盖性假设 (Xie et al., 2025)。介于两者之间的是不带探索的迭代，它要求覆盖各次迭代——这个条件比全策略覆盖弱得多，又比什么都不要求强得多 (Xiong et al., 2023)。

还有一种更干净的方式可以看清新鲜样本为何有帮助，值得作为单独的推导写出来，因为它解释了一个具体的算法。

推导: 极大似然需要来自目标分布的样本，而不是来自过去的样本

我们在节 8.1.2中最大化的 Bradley–Terry 似然是在样本对上的期望。是在哪个分布下呢？要用极大似然估计带 KL 正则的最优解 $\pi^\star$，样本对就应当从 $\pi^\star$ 本身抽取，因为那正是我们要估计其参数的那个分布。而 DPO 从生成 $\mathcal{D}$ 的任何东西中抽取——通常是人类写作与另一个模型输出的混合——SLiC 则从 π_{sft} 中抽取。二者都不是 $\pi^\star$，因此两者求解的都是一个误设的估计问题。

Liu et al. (2024d) 把这一点说清楚并给出了修补。训练一个显式的奖励模型 r_ϕ，由(8.2)注意到其诱导的最优策略是 $\pi_{r_\phi}(y\mid x) \propto \pi_{\text{sft}}(y\mid x)\exp(r_\phi(x,y)/\beta)$，并用拒绝采样从中抽样：提议 $y\sim\pi_{\text{sft}}$，以概率 $\pi_{r_\phi}(y\mid x)/(M\pi_{\text{sft}}(y\mid x))$ 接受。从一批提议 D_x 中估计包络常数 M，便给出一个可计算的接受概率，

$$\mathbb{P}(\text{accept } y) = \exp\!\left(\frac{1}{\beta}\big(r_\phi(x,y) - \max_{y'\in D_x} r_\phi(x,y')\big)\right). \tag{9.1}$$

有两个极限很有启发。当 $\beta\to\infty$ 时每个提议都被接受，我们就退回到在 π_{sft} 样本上的离线 DPO。当 $\beta\to 0$ 时只有 argmax 被接受，拒绝采样优化便退化为 *best-of-n*。因此 best-of-n 正是「从目标策略中正确抽样」的零温度极限，这对于「如此粗糙的做法为何效果这么好」是一个令人满意的解释。

注记 9.1 (RSO 在损失一侧也做了统一). Liu et al. (2024d) 观察到，DPO 是在归一化似然上的逻辑斯谛回归，而 SLiC 近似是一个带合页损失的支持向量机，这就留下了一个显而易见的空格——作用在归一化似然上的合页损失。他们把它补上了，并注意到第二处有用的解耦：DPO 把损失的温度与 KL 系数绑在一起，即 $\gamma=\beta$，而并没有什么东西要求如此。把二者当作独立旋钮是免费的，偶尔还会有帮助。

9.1.2 经验层面的陈述，及其无法解释的残余

Tang et al. (2024a) 检验了关于在线–离线鸿沟的各种显而易见的假设并逐一否定：它既不单纯是覆盖性问题，也不是损失函数问题，而且不会随规模增长而闭合。真正相关的是采样分布本身的同策略程度。他们的结论——离线算法擅长分类而在生成上较差——诚实地承认自己缺少机制，而截至 2026 年年中，它仍然是这一文献中较为重要的开放问题之一。

Tajwar et al. (2024) 给出了最尖锐的建设性版本。偏好微调应当使用次优的、同策略的数据，原因在于两种成分的相互作用：同策略采样把概率质量放在当前策略实际所处的位置上，而被拒绝回复上的负梯度把质量从那里移开。在好数据上做极大似然只能增加质量，它无法从某个特定位置把质量减去。对比机制两者都需要，而且它需要那些负样本是模型真的会产生的样本。这立刻解释了迭代为何有帮助，同时也预测了一件不那么显然的事：一个被拒绝回复的价值取决于当前策略生成它的可能性有多大，因此一个陈旧的被拒绝回复几乎一文不值。

核心思想: 迭代真正买来的是什么

不是更多的数据，而是相关的负样本。梯度(8.8)压低 y^-；只有当 y^- 处在策略当前所占据的区域时，这一压低才有用。随着策略移动，旧的负样本变得无关，而该目标函数中最有价值的那一项也就被浪费掉了。

9.2 迭代式直接对齐

基本循环短到一句话就能讲完：从当前策略采样回复，获取偏好标签，施加某个直接对齐目标，然后重复。所有有意思的东西都在细节里。

标签从哪里来。 有三种选择，按成本与保真度递增排列。学出来的奖励模型（第5章）最便宜，也陈旧得最快，因为它是拟合到一个策略正在主动离开的分布上的。LLM 评判模型对分布偏移更鲁棒，但会引入自身的偏置，尤其是偏向长度、以及偏向自己的输出。新鲜的人类标签最好，但通常只有一到两轮的预算负担得起。在线 AI 反馈（Online AI Feedback）(Guo et al., 2024) 是中间那一档的干净版本：每一步都从当前策略采样两个回复，让一个 LLM 标注者当即给出标签，然后施加 DPO 损失。这让 DPO 真正成为在线的，而不只是迭代的；它还有一个很有吸引力的性质，即对齐目标可以通过标注者的提示来控制——改一个字符串，就改变了模型被对齐的对象。

NLL 项不是可选的。 迭代式推理偏好优化 (Pang et al., 2024) 把这个循环应用于「思维链加答案」，并报告了一个远超其自身设定的普适细节：如果不在被选中回复上加一个负对数似然项，被选中回复的似然会随轮次坍缩。这就是定理 8.2的复利效应。在单次离线训练中，位移是一个有界的麻烦；而在每一轮策略都生成下一轮数据的五轮训练里，它是一个反馈回路。每一份认真的迭代配方都包含这个锚定项。

算法 9.1 迭代式直接偏好优化。第 3、5、7、9 行上的选择才是决定这套流程能否奏效的因素；第 8 行的损失是其中最不重要的部分。

输入： 提示池 $\mathcal{X}$、初始策略 π_{sft}、标注器 O、轮数 R

$\pi_\theta[0] \leftarrow \pi_{\text{sft}}$
for $t = 0$ **to** $R-1$ **do**
  抽取一个新的提示分片 $\mathcal{X}_t \subset \mathcal{X}$ ▷ 切勿跨轮复用提示
  for $x \in \mathcal{X}_t$ **do**
    在温度 ≥ 1 下采样 n 个回复 $\{y^{(i)}\}_{i=1}^{n} \sim \pi_\theta[t](\cdot \mid x)$
    用 O 打分；按一条明确写下的规则构造样本对 (y^+, y^-)
    仅当分数间隔超过 δ 时才保留该样本对 ▷ 过滤比损失更要紧
  end for
  $\pi_{\text{ref}} \leftarrow \pi_\theta[t]$ **或** $\pi_{\text{ref}} \leftarrow \pi_{\text{sft}}$ ▷ 这是一个真实的选择；见节 9.6
  $\pi_\theta[t+1] \leftarrow \arg\min_\theta \mathcal{L}_{\text{DAA}}(\theta; \pi_{\text{ref}}) - \lambda\ \mathbb{E}[\log \pi_\theta(y^+ \mid x)]$
  在一个固定的留出集上评估；若质量下降则**停止**
end for
return 表现最好的那一轮，而不是最后一轮

复用提示是最常见的错误。 如果第 $t+1$ 轮训练用的提示与第 t 轮相同，那就是在一个模型已经被优化过的提示上，反复告诉它去偏好自己的某个输出，而实际的目标函数会漂向记住一份逐提示的排序。请把提示池切成分片。

常见陷阱: 评判模型在回路中的漂移

当一个 LLM 评判模型跨越多轮为自己家族的输出打标时，两种效应会叠加。评判模型的已知偏置变成了训练目标，而策略学会了评判模型的风格签名，这又进一步加深评判模型对它的偏好。可观察的症状是：对评判模型的胜率单调上升，而一项留出的人类评估或第三方家族评估却持平甚至下降。缓解办法是至少留出一个从不用于训练的评估器，并在各轮之间轮换评判模型。

9.3 无标签学习

如果标注器可以就是模型自身，那么这个循环就完全不需要外部监督了。这件事的吸引力与危险性大致相当。

与自己的历史自博弈。SPIN (Chen et al., 2024f) 完全不需要偏好标签：把 SFT 数据中的人类回复当作被选中的，把上一轮自己的生成当作被拒绝的，然后迭代。这一构造有一个干净的两人博弈解释，其不动点在模型的生成与人类数据不可区分时达到；而它的局限也随即从这一解释中推出——它无法超越 SFT 数据，因为 SFT 数据就是目标。

自奖励。 Yuan et al. (2024a) 让同一个模型既充当策略又充当评判模型，每一轮生成自己的偏好对，并报告指令遵循能力与评判能力都随迭代提升。独立的奖励模型就此消失。DICE (Chen et al., 2024b) 以更经济的方式做同一件事：不去提示模型作评判，而是直接使用上一轮的 *DPO* 隐式奖励(8.5)，它是零额外成本就能拿到的。

从原则出发的自对齐。宪法式 AI（Constitutional AI）(Bai et al., 2022b) 是有影响力的早期实例，其结构值得留意，因为后来的大多数工作都在复用它。第一阶段是有监督的：对一个红队测试提示采样一个回复，让模型对照一条随机抽取的成文原则去批评它，再让它修订，然后在修订结果上微调。第二阶段是基于 AI 反馈的强化学习：生成回复对，让一个反馈模型依据随机抽取的原则判断哪个更好，把一个偏好模型拟合到这些 AI 标签上，再针对它跑 RL。有两个设计细节真正起作用，却常在复现中被丢掉：反馈模型在作答之前会以思维链推理，而各条原则是通过随机抽样来 集成的，而不是拼接在一起，这防止了对任何单条规则的过拟合。论文报告的结果——有用性与无害性同时提升，且比一个 RLHF 训练的模型明显更少回避——正是让 AI 反馈变得可信的原因。

Lee et al. (2023) 做了直接的正面比较，发现在摘要任务以及有用性与无害性两类对话上，RLAIF 取得的人类评分胜率与 RLHF 相当或更好，AI 标注者不必比策略更大，而且一种「直接 RLAIF」变体——直接用一个现成模型的打分作为奖励、根本不拟合奖励模型——也是可行的。更早的自对齐脉络——Self-Instruct (Wang et al., 2023b)、原则驱动的自对齐 (Sun et al., 2023b) 以及指令回译 (Li et al., 2024c)——确立了这个问题中数据生成的那一半同样可以自举完成。

常见陷阱: 自我改进回路没有外部纠错

本节中的每一种方法都用模型替换了人类，这意味着模型所犯的每一个系统性错误如今都成了训练目标。这种失效并不喧闹。一个已经开始奖励自身特异偏好的自奖励回路，会呈现出上升的自评胜率、平稳的损失，以及只有对照一个固定的外部评估才能看出的缓慢退化。请为每一轮预留少量留出的人类评估或第三方评估，并把它当作停止准则；它是后训练中最便宜的保险。

9.4 探索：把启发式做法变成算法

迭代是一种启发式做法。它从当前策略采样，而这与主动寻找那些信息量最大的比较并不是一回事。有两个密切相关的方法弥合了这一差距，而它们在理论上的分量远超其篇幅。

探索式偏好优化（Exploratory Preference Optimization）(Xie et al., 2025) 只给在线 DPO 加了一项——一个乐观探索奖励：

$$\pi^{(t+1)} \in \arg\min_{\pi\in\Pi} \left\{ \underbrace{\alpha \sum_{i=1}^{t} \log \pi(\tilde{\tau}^{(i)})}_{\text{optimism}} - \sum_{(\tau_+,\tau_-)\in\mathcal{D}_{\text{pref}}^{(t)}} \log \sigma\Big(\beta \log \frac{\pi(\tau_+)}{\pi_{\text{ref}}(\tau_+)} - \beta \log \frac{\pi(\tau_-)}{\pi_{\text{ref}}(\tau_-)}\Big) \right\}. \tag{9.2}$$

这个奖励项鼓励把概率质量放到策略自己采样出来的轨迹上，从而隐式地实现了「面对不确定性时保持乐观」，而无需在奖励函数上构造置信集。其回报是一个不要求离线数据覆盖最优策略的样本复杂度保证——只需要可实现性。用表 8.4的语言说，增加一个正则项就把该方法从需要全策略集中度，变成了对离线数据毫无要求。这是以极小的实现改动换来的极大理论回报，也是全书中关于「探索是一等关切而非事后补丁」最干净的例证。自探索语言模型（Self-Exploring Language Models）(Zhang et al., 2024b) 从诱导一侧追求同一目标，让生成偏向富含信息的回复，而不是被动地利用已知。

注记 9.2 (为什么这与调高温度不是一回事). 在更高温度下采样会增加离散度；它并不会增加信息量，因为多出来的样本是按当前策略的形状铺开的，而不是按目标函数最不确定的方向铺开的。乐观是有方向的。这一区别在实践中很重要：温度是一个廉价而不完整的替代品，它之所以有帮助，主要是因为它恰好减少了退化比较对的数量。

9.5 抛弃 Bradley–Terry：把对齐当作博弈

到目前为止的一切，都假定偏好是由一个标量奖励通过(8.4)生成的。这个假设在一个具体而且后果严重的意义上是经验上错误的：人类偏好是非传递的。在标注者之间聚合之后，出现 $a \succ b$、$b \succ c$ 而 $c \succ a$ 是家常便饭，而任何标量奖励都无法表示这种情形。在这一情形下，Bradley–Terry 不只是精度下降；它是被误设了，而它拟合出的极大似然奖励只是聚合过程的产物。

9.5.1 偏好博弈

假设我们改为只被给定一个一般的偏好预言机 $\mathcal{P}(y \succ y' \mid x) \in [0,1]$，并不假设它能经由某个奖励分解。那么什么是「最好的策略」？自然的答案是博弈论式的：定义策略 π 对阵策略 π' 的正则化收益，

$$\mathcal{P}_\tau(\pi \succ \pi') = \mathcal{P}(\pi \succ \pi') - \tau \mathbb{D}_{\mathrm{KL}}(\pi \,\|\, \mu) + \tau \mathbb{D}_{\mathrm{KL}}(\pi' \,\|\, \mu), \tag{9.3}$$

并把解取为所得两人零和博弈的纳什均衡。一个策略之所以好，是因为它战胜其他每一个策略的次数多于失败的次数，而不是因为它最大化了某个数值。

定义 9.3 (冯·诺依曼胜者)**.** 若 $\min_{\pi'} \mathcal{P}(\pi^{\mathrm{vN}} \succ \pi') \geq 1/2$，则称策略 π^{vN} 是偏好预言机 $\mathcal{P}$ 的一个冯·诺依曼胜者（或极小极大胜者）。由极小极大定理，对任何 $\mathcal{P}$——包括非传递的 $\mathcal{P}$——这样的策略都以混合策略的形式存在。当 $\mathcal{P}$ 恰好由某个奖励诱导时，它与 Bradley–Terry 的 argmax 重合。

定理 9.3是整个博弈论纲领在概念上的回报：它提供了一个在非传递性下依然良定义的解概念，并且在 Bradley–Terry 成立时会优雅地退化为那个我们熟悉的解概念。本节其余的一切，都是计算它的算法。

9.5.2 Nash-MD

Munos et al. (2024) 提出了从人类反馈中进行纳什学习，并给出了第一个具有末次迭代收敛性的算法。它是针对当前策略与参考策略的几何混合所作的镜像下降：

$$\pi_t^{\mu}(y) \propto \pi_t(y)^{1-\eta\tau}\, \mu(y)^{\eta\tau}, \qquad \pi_{t+1} = \arg\max_{\pi} \Big[\eta\, \mathcal{P}(\pi \succ \pi_t^{\mu}) - \mathbb{D}_{\mathrm{KL}}\big(\pi \,\big\|\, \pi_t^{\mu}\big) \Big], \tag{9.4}$$

其闭式更新为 $\pi_{t+1}(y) \propto \pi_t^{\mu}(y) \exp(\eta\, \mathcal{P}(y \succ \pi_t^{\mu}))$。几何混合是真正起作用的装置：对阵一个正则化版本的自己，而不是对阵自己本身，可以让博弈的内层循环不至于追着自己的尾巴跑，而这正是换来末次迭代收敛（而非平均迭代收敛）的原因。词元级实现按词元定义该混合，$\log \tilde{\pi}_\theta^{\tau}(y_t \mid \cdot) = \tau \log \pi_\theta(y_t \mid \cdot) + (1-\tau) \log \mu(y_t \mid \cdot) + C$，并使用策略梯度估计量

$$\hat{g} = \nabla_\theta \log \pi_\theta(y \mid x) \Big(\mathcal{P}(y \succ y' \mid x) - \tfrac{1}{2} - \tau \log \tfrac{\pi_\theta(y \mid x)}{\mu(y \mid x)} \Big), \tag{9.5}$$

把它与(6.17)逐项对照很有价值：胜率减去二分之一扮演了优势的角色，而对数比值项则是直接出现在梯度中（而非出现在损失中）的 KL 正则。

9.5.3 你并不需要两个玩家

Swamy et al. (2024) 的观察简化了整个纲领。由于偏好博弈是对称的且常和的，一个自我对弈的无悔学习器就能收敛到极小极大胜者。根本不需要维护第二个相互竞争的策略，而由此得到的算法——在他们的表述中即自博弈偏好优化——能够鲁棒地处理非传递且带噪的偏好，同时从外部看去，就像是带有某种特定配对规则的普通迭代训练。如果你想理解博弈论文献为何最终产出了一批与算法 9.1 相似的算法，这就是该读的那篇论文。

9.5.4 SPPO：把乘性权重写成回归

Wu et al. (2025f) 把指数权重更新转化为一个平方回归目标：

$$\pi_{t+1} = \arg\min_{\pi} \mathop{\mathbb{E}}_{x\sim\mathcal{X},\, y\sim\pi_t(\cdot\,|\,x)} \left(\log \frac{\pi(y\,|\,x)}{\pi_t(y\,|\,x)} - \eta\Big(\mathcal{P}(y \succ \hat{\pi}_t^{K}\,|\,x) - \tfrac{1}{2} \Big) \right)^2 . \tag{9.6}$$

其中 $\frac{1}{2}$ 取代了难以计算的 $\eta^{-1}\log Z_{\hat{\pi}_t^K}(x)$，并且在对称偏好模型下，它是后者当 $K\to\infty$ 时的正确极限。胜率由 y 对阵从 π_t 抽取的 K 个样本的平均胜率来估计，每个提示需要 $O(K^2)$ 次预言机调用。

与 DPO 和 IPO 的对比才是关键所在，而且很容易被忽略。式(9.6)对两个回复并不对称，也不需要成对。它把每一个被采样回复各自的对数比值，推向该回复自身的胜率。一个有 70% 时间获胜的回复会被按一个特定幅度抬高；这里没有「被选中」也没有「被拒绝」。这是一种与对比机制真正不同的机制，也解释了为什么 SPPO 能自然地使用一组样本，而成对方法必须先构造样本对、并在此过程中丢弃信息。作者还证明(9.6)恰好是一个以 $\eta^{-1}\log Z$ 作为最优软值基线的半在线策略梯度——而这正是第6章中出现过的那种「基线即对数配分函数」的结构。

9.5.5 DNO 与末次迭代收敛

直接纳什优化（Direct Nash Optimization）(Rosset et al., 2024) 是分批的镜像下降，其每一个内层步骤都朝真实胜率作回归。记 $\Delta_{\pi,t}(x,y_1,y_2) = \eta\log\frac{\pi(y_1\,|\,x)}{\pi_t(y_1\,|\,x)} - \eta\log\frac{\pi(y_2\,|\,x)}{\pi_t(y_2\,|\,x)}$，则更新最大化

$$\mathop{\mathbb{E}}_{(x,y_1,y_2)\sim\mathcal{D}_t} \Big[\sigma(\Delta_t^{\star})\log\sigma(\Delta_{\pi,t}) + \sigma(-\Delta_t^{\star})\log\sigma(-\Delta_{\pi,t}) \Big], \tag{9.7}$$

这是对软目标（而非硬偏好标签）的交叉熵。其实用变体简化为一种以上一次迭代作为参考的迭代式 DPO 更新，并用一条大间隔规则筛选保留哪些样本对——这又一次是算

表 9.1: 博弈论一族，按各自的贡献编排。请先读最后一列：这五个方法中有三个在其实用实现里都归结为「带特定参考策略与样本对筛选规则的迭代式直接偏好优化」。这一趋同是关于这一文献最有用的事实。

方法	解概念	机制	实践中的形态
Nash-MD (Munos et al., 2024)	正则化纳什	针对自身与参考的几何混合作镜像下降	策略梯度，(9.5)
SPO (Swamy et al., 2024)	极小极大胜者	对称常和博弈只需要一个无悔学习器	自博弈，无第二个模型
SPPO (Wu et al., 2025f)	正则化纳什	把每个样本的对数比值回归到它自己的胜率；**非成对**	组上的平方损失
DNO (Rosset et al., 2024)	纳什	分批镜像下降，回归到软胜率	以上一次迭代为参考的迭代式 DPO
MPO (Wang et al., 2024d)	纳什，未正则化	磁性镜像下降；末次迭代	迭代式，外加一个磁性项

法 9.1在第 7 行与第 9 行上作了特定选择的结果。磁性偏好优化（Magnetic Preference Optimization）(Wang et al., 2024d) 补上了剩余的理论缺口，通过磁性镜像下降实现了对原始的、未正则化博弈的纳什均衡的末次迭代收敛。Nan et al. (2026) 为迭代式纳什设定加入了高效探索，这是节 9.4与本节的自然结合。

核心思想: 博弈论文献真正的贡献是概念上的

它的算法大多归结为带特定参考策略与配对规则选择的迭代训练——这正是它们实用的原因。它持久的贡献在于重新框定问题：「对齐后的策略」是一个偏好博弈的冯·诺依曼胜者，而不是某个奖励的 argmax。这一框架能挺过非传递性、标注者分歧与多元偏好，而这些都会击垮 Bradley–Terry 那套说法。下次当你看到一个一致率为 68% 的奖励模型时，请记住：残余的那 32% 中，有一部分可能是结构上就无法表示的，而不是噪声。

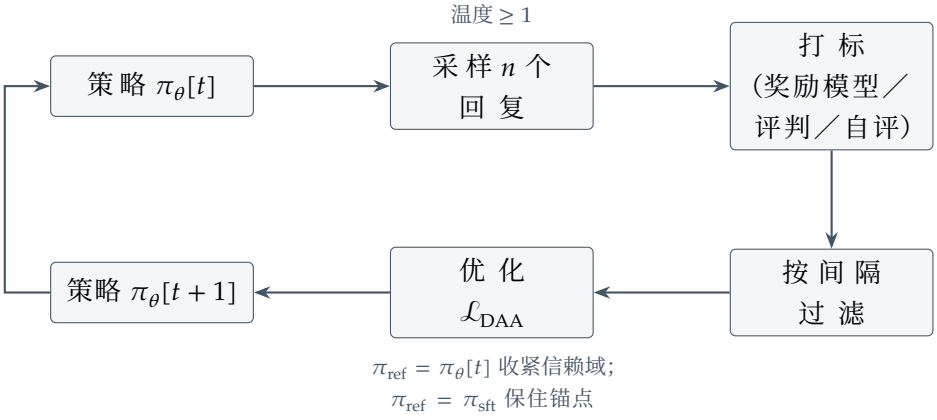


图 9.1: 节 9.2、节 9.3与节 9.5 所实例化的都是这同一个迭代循环。各种有名字的方法之间的差别，几乎完全是三个节点上的选择：由谁打标、样本对（或样本组）如何构造，以及用什么作参考。损失是其中区分度最低的选择。

9.6 迭代之后会出什么问题

迭代修好了覆盖性问题，又制造出一类新问题，而它们共享同一个机制：第 t 轮的策略生成第 $t+1$ 轮的数据，所以任何系统性误差都会被几何式放大，而不是只承受一次。

对齐坍缩。Gauthier et al. (2026) 直接研究了多轮设定，刻画了重复的轮次何时会损害对齐而非改善它。其本质机制是 算法 9.1第 9 行上的参考策略选择。取 $\pi_{\text{ref}} \leftarrow \pi_\theta[t]$ 会给每一轮一个很紧的信赖域，但会让 R 轮之后的 $\mathbb{D}_{\text{KL}}(\pi_\theta[R] \,\|\, \pi_{\text{sft}})$ 无界，因为每一轮只是相对上一轮受约束。取 $\pi_{\text{ref}} \leftarrow \pi_{\text{sft}}$ 保住了锚点，但会让后期的轮次去对抗一个越来越遥远的参考，从而拖慢学习并最终使其停滞。这里没有免费的选择；实践中的折中是采用 TR-DPO 那样缓慢更新的参考（节 8.9），并对相对 π_{sft} 的累计 KL 设定明确预算。

多样性单向收窄。反向 KL 是寻众数的（节 6.5），所以每一轮都会把输出分布收窄一些。第 $t+1$ 轮随后从这个已收窄的分布中采集训练数据，因此它无法找回第 t 轮丢掉的东西。经过五轮，这种复利效应相当严重，而且它在任何单轮指标中都不可见。节 14.3处理其测量问题。

群体层面的外部性。2026 年的两项结果论证了偏好优化具有超出单个模型的系统性影响。Shapira et al. (2026) 表明 RLHF 会放大谄媚——考虑到讨人喜欢的回复更容易赢得比较，这并不意外——而Begin et al. (2026) 表明偏好优化会推动单一栽培：独立训练出来的模型会趋同于相同的输出，在他们的预测市场设定中，这摧毁了使聚合

表 9.2: 迭代所特有的失效模式。它们中的每一个在单轮训练损失中都不可见，每一个都会跨轮次复利累积而不是饱和。检测一列才是这张表的运维价值所在。

失效	机制	如何检测
似然坍缩	定理 8.2跨轮次复利累积	跨轮次而非单轮内的 $\log \pi_\theta(y^+) - \log \pi_{\text{sft}}(y^+)$
无界漂移	$\pi_{\text{ref}} \leftarrow \pi_\theta[t]$ 只约束每一轮，不约束总量	累计 $\mathbb{D}_{\text{KL}}(\pi_\theta[t] \,\|\, \pi_{\text{sft}})$，须跟踪并设预算
多样性棘轮	寻众数式收窄无法从已收窄的数据中恢复	固定提示集上的 distinct-n 与答案级熵
评判模型俘获	策略学会评判模型的签名，从而抬高评判分数	一个从不用于训练的留出评估器
长度棘轮	每一轮的评判模型都偏好更长；每一轮的数据都更长	按轮统计平均长度，并单独测量评判模型的长度偏置
提示记忆	提示跨轮复用	在同一提示分布上的训练／留出差距

变得有用的那种多样性。再结合 2026 年针对偏好数据本身有效性的种种批评，令人不安的结论是：这整条流水线上薄弱的一环，可能是诱导（elicitation）这一步，而不是优化那一步。这些结果都很新且仅有单篇证据；请把方向看得比幅度更可信。

9.7 一份可用的迭代配方

工程实践: 在 8B 模型上做三轮迭代对齐

把提示池切成 $R+1$ 个互不相交的部分，每轮一份，外加一份永久留作评估。每一轮：每个提示在温度 1.0 下采样 $n = 8$ 个回复，且不作 top-p 截断（关于数据生成阶段的截断为何比看上去更糟，见节 22.7）；用一个不属于策略所在家族的评判模型打分；从最好与最差的回复中为每个提示构造一对，并在分数间隔低于评判模型自身噪声底线时丢弃该提示——这个噪声底线必须由你把同一批样本重新打两次分来测量。

用 DPO 训练，$\beta = 0.1$，外加一个长度归一化的 NLL 锚定项，$\lambda = 0.2$，训练一个 epoch，学习率 5×10^{-7} 并在最后一轮衰减到 2×10^{-7}。取 $\pi_{\text{ref}} \leftarrow \pi_\theta[t]$，但一旦固定探针集上的累计 $\mathbb{D}_{\text{KL}}(\pi_\theta[t] \,\|\, \pi_{\text{sft}})$ 超出你的预算，就硬性中止训练。在多数已发表的配方中，三轮就是收益递减的拐点；边际的那一轮通常还不如把同样的算力花在更好的标签上。

> 每一轮都在那份永久留出的分片上测量：由固定评估器评判的、对阵一个固定参考模型的胜率，平均回复长度，distinct-n 多样性，以及被选中回复相对 π_{sft} 的对数比值。如果后三者中任何一项在胜率进入平台期时仍在单调变化，那你就在拉棘轮，正确的做法是停下来。

代码 9.1: 样本对的构造才是迭代方法真正的差异所在，而它通常被当作事后补充随手写就。这个版本记录了日后审计这个循环所需的信息：分数间隔、评判模型的噪声估计，以及数据来自哪一轮。丢弃低间隔的提示不是可选项——它们贡献的梯度方向是标注器无法为之辩护的。

```python
def build_pairs(prompt, responses, judge, noise_floor, round_idx):
    scores = judge.score_batch(prompt, responses)      # 分数越高越好
    order = sorted(range(len(responses)), key=lambda i: scores[i])
    lo, hi = order[0], order[-1]
    margin = scores[hi] - scores[lo]
    if margin < noise_floor:                           # 评判模型无法区分二者
        return None
    return {
        "prompt": prompt,
        "chosen": responses[hi],
        "rejected": responses[lo],
        "margin": margin,                              # 供 beta-DPO / 过滤使用
        "round": round_idx,                            # 供陈旧度审计使用
        "n_sampled": len(responses),
    }

def estimate_noise_floor(judge, probe_pairs, quantile=0.5):
    # 对相同输入重新打分；分散程度即评判模型自身的噪声
    a = [judge.score(p, y) for p, y in probe_pairs]
    b = [judge.score(p, y) for p, y in probe_pairs]
    return torch.tensor([abs(x - y) for x, y in zip(a, b)]).quantile(quantile)
```

9.8 哪些结论真正站得住

复现良好。在目标函数固定的前提下，同策略数据胜过离线数据 (Tang et al., 2024a; Tajwar et al., 2024)。迭代能挽回这一鸿沟中相当大的一部分。一旦开始迭代，被选中回复上的锚定项就是必需的 (Pang et al., 2024)。在有用性与无害性上，AI 反馈以极小的成本就能与人类反馈相竞争 (Lee et al., 2023; Bai et al., 2022b)。在已发表的配方中，大致三轮之后收益递减。

理论上已确立，实践中采用不足。乐观探索消除了对离线数据的覆盖性要求 (Xie et al., 2025)。博弈论式的形式化在非传递性下给出了一个良定义的解概念 (Munos et al., 2024; Swamy et al., 2024)。二者都不是社区的默认选项，而诚实的原因是：它们都比一个已经能凑合工作的循环更难调。

仍有争议。在同等算力下，博弈论算法是否优于经过良好调优的迭代式 DPO。已发表的比较与节 12.12中讨论的那类比较一样，统计效能不足：单一随机种子、单一规模，而且基线是由它的批评者调的。

真正开放。同策略程度为何在覆盖性之外仍有帮助 (Tang et al., 2024a)——至今仍无机制可言。自我改进回路除了标注器的质量之外是否还有天花板，以及它在哪里。群体层面的效应 (Shapira et al., 2026; Begin et al., 2026) 究竟是特定实验设置的产物，还是偏好优化的普遍性质。

本章小结

- 对带 KL 正则的最优解作极大似然，需要从那个最优解中采样得到的样本对。拒绝采样能提供它们，而 best-of-n 恰好是这一做法在 $\beta \to 0$ 时的极限 (Liu et al., 2024d)。
- 迭代并不只是增加数据；它提供的是相关的负样本。负梯度只有在当前策略确实放置了概率质量的地方才有用，这就是为什么一个陈旧的被拒绝回复几乎一文不值 (Tajwar et al., 2024)。
- 在线–离线鸿沟无法用覆盖性、损失形状或规模来解释；它与同策略程度本身相关，而机制至今未知 (Tang et al., 2024a)。
- 一旦开始迭代，被选中回复上的 NLL 锚定项就不再是一项可选的改良：似然位移会跨轮次复利累积成一个反馈回路 (Pang et al., 2024)。
- 只需增加一个乐观项，就能把在线 DPO 从需要全策略集中度，变成对离线数据毫无要求 (Xie et al., 2025)。乐观是有方向的；温度不是它的替代品。
- 人类偏好是非传递的，所以 Bradley–Terry 是被误设了，而不只是带噪。偏好博弈的冯·诺依曼胜者是一个能挺过这一点的解概念，并在 Bradley–Terry 成立时退化为奖励的 argmax。
- 由于偏好博弈是对称且常和的，一个自我对弈的无悔学习器就能收敛到极小极大胜者 (Swamy et al., 2024)。不需要第二个模型。

- SPPO 不是成对方法：它把每个被采样回复的对数比值回归到该回复自身的胜率上，这是一种不同于对比训练的机制，并且能自然地与样本组组合 (Wu et al., 2025f)。
- 迭代循环中的参考策略选择是一个真实的两难。更新它会收紧每一轮的信赖域，却让累计漂移无界；冻结它保住了锚点，却会让后期轮次停滞 (Gauthier et al., 2026)。
- 每一种迭代特有的失效——似然坍缩、多样性棘轮、评判模型俘获、长度棘轮——都会跨轮次复利累积，而且在单轮损失中不可见。一个永久留出的评估器是唯一可靠的检测手段。

延伸阅读

从Tajwar et al. (2024) 开始，它对「同策略数据为何重要」的陈述最为清晰，并且给出的是机制（同策略样本加上负梯度），而不只是一个观察。紧接着读 Tang et al. (2024a) 作为诚实的对照：他们检验了那些显而易见的解释并逐一否定，而由此留下的开放问题，比一个干净利落的答案更有意思。Liu et al. (2024d) 篇幅短、被低估，单是为了节 9.1.1中的那段推导就值得一读；正是这篇论文让 best-of-n 作为一个极限（而不是一个 hack）变得可以理解。

就博弈论那一半而言，读Munos et al. (2024) 以了解形式化，读Swamy et al. (2024) 以了解那个让它变得实用的简化——请按这个顺序，因为当你见过第二篇所移除的那套两人机器之后，才更容易体会到它的价值。Wu et al. (2025f) 是值得动手实现的那一篇，既因为其目标函数就是一个普通的平方损失，也因为它的非成对结构与第8章中的一切都截然不同。Xie et al. (2025) 是本章理论上最有分量的论文，也值回票价：理论收益与实现成本之比，在整个对齐文献中是最好的。

关于自对齐，读Bai et al. (2022b)——请仔细读「批评与修订」那一阶段，因为绝大部分行为改变发生在那里，而它恰恰是摘要中通常被略过的部分——然后读Lee et al. (2023) 的受控比较。如果你关心这套做法能否推广到比模型更弱的监督者，Burns et al. (2023) 是自然的下一步；而Guan et al. (2024) 是 2025 年从「对齐策略」到「把规范交给策略、让它自己推理」的转向，第25章会展开这一点。

有意推迟的内容：best-of-n 的 KL 核算在节 10.4.1；可验证奖励——它让标注器变得免费且精确，从而消解了本章的大部分难题——从第11章开始；大规模多智能体自博弈则是第20章。

习题

9.1 从节 9.1.1中的拒绝采样方案推导出接受概率(9.1)，并明确指出包络常数 M 的估计在何处进入、该估计引入了什么偏差。然后验证两个极限论断：$\beta \to \infty$ 时接受一切，$\beta \to 0$ 时退化为 best-of-n。

9.2 节 9.1断言一个陈旧的被拒绝回复几乎一文不值。请把它定量化。把一个样本对对 DPO 梯度的贡献写成当前策略给该被拒绝回复所赋概率的函数，并确定随着策略远离它，这一贡献衰减得有多快。这对固定标注预算下的最优轮数意味着什么？

9.3 证明(9.4)中的几何混合 $\pi_t^{\mu} \propto \pi_t^{1-\eta\tau}\mu^{\eta\tau}$ 是某个 KL 正则化问题的解，并指出是哪个问题。然后用一段话解释：为什么对阵这个混合而非对阵 π_t 才能带来末次迭代收敛。

9.4 在三个回复上构造一个显式的非传递偏好预言机，并计算（i）它所诱导的 Bradley–Terry 极大似然奖励，以及（ii）定理 9.3中的冯·诺依曼胜者。证明二者不一致，并说明各自在优化什么。

9.5 把(9.6)与(8.7)逐项对照。明确证明 SPPO 的目标函数在交换两个回复时不保持不变，并解释成对构造丢弃了哪些 SPPO 保留下来的信息。在胜率估计满足什么条件时，二者会重合？

9.6 一位同事用 $\pi_{\text{ref}} \leftarrow \pi_\theta[t]$ 跑了五轮迭代式 DPO，报告说对阵固定基线的胜率上升了三轮然后下降。请利用表 9.2提出一套有序的诊断流程。如果从一开始就在每一轮采集，哪一项单一测量最有可能识别出原因？

9.7 证明 XPO 在(9.2)中的奖励项等价于某个特定的奖励加成，并指出它是什么。然后参照所诱导梯度的方向，解释为什么这一加成无法通过调高采样温度来复现。

9.8 ⋆ 节 9.6把参考策略的选择呈现为一个两难：更新 π_{ref} 会让相对 π_{sft} 的累计漂移无界，冻结它则会让后期轮次停滞。请把它形式化为一个在参考策略序列上的约束优化问题，并对累计 $\mathbb{D}_{\text{KL}}(\pi_\theta[R] \,\|\, \pi_{\text{sft}})$ 设定明确预算，然后在你自选的一个简化假设下推导最优调度。请明确指出是哪个假设在起作用。

9.9 ⋆ Tang et al. (2024a) 排除了覆盖性、损失形状与规模作为在线–离线鸿沟的解释。请提出一个他们没有检验过的机制，并设计一个能把它与其余候选区分开的实验。你的设计必须说明什么样的结果会证伪你所提出的机制。

9.10 † 实现算法 9.1，用一个 8B 模型和一个来自不同家族的评判模型跑三轮。为表 9.2的每一行都装上仪表。然后跑第二组实验，除了提示跨轮复用之外其余完全相同；再跑第三组，除了去掉 NLL 锚定项之外其余完全相同。报告哪一组最先退化、在哪个指标上退化、在第几轮退化。请说明哪些差异落在种子噪声之内——请至少跑两个随机种子，或者明说你没有这么做。

9.11 † 在同一份代码库中实现 SPPO (9.6)与迭代式 DPO，共用采样与评判流水线，使二者唯一的差别在于更新方式。在同等预言机调用预算下（而不是同等梯度步数——SPPO 的 $O(K^2)$ 胜率估计才是主要成本），比较三轮之内的胜率与答案级多样性。讨论 SPPO 的非成对结构是否表现为对采样组的更好利用，以及你会如何把它与调参差异区分开来。

第 III 部分

推理

第10章

推理与测试时计算

核心思想：推理强化学习把一套推理时的搜索过程摊销进权重

在可验证奖励强化学习出现之前，让模型推理得更好的办法是让它在推理时想得更久：输出中间词元、采样多条链、在其中做聚合或选择、在部分解上做搜索。那一批文献确立了两个事实，而第三部分的一切都建立在它们之上。额外的推理计算会沿一条可预测的曲线买到准确率；并且同等规模下，一个验证器比一个生成器更有价值。一旦这两点同时成立，下一步就是被逼出来的：凡是验证器在推理时能排序的东西，它在训练时就能用来给奖励；而原本花在搜索上的计算，可以改为花在让第一个样本更好上。

第二部分把后训练当作安装偏好的问题。本部分则把它当作安装能力的问题，而起点不是某个算法，而是一个关于算术的观察。一个在竞赛题上首次尝试只解出 10% 的模型，往往能在两百个样本之内解出其中的大多数（节 2.8）：能力已经存在于分布之中，只是无法一次取出。本章的每一种技术都在花计算把它取出来，而第11章中的每一种技术都是一种不必这样做的办法。

10.1 思维链是计算，不是解释

10.1.1 机制

Transformer 的一次前向传播是固定深度的计算：对一个有 L 层的模型，产生一个输出词元恰好涉及 L 轮注意力与前馈变换，不多一轮，无论问题有多难。这不是一句关于容量的软性表述。Merrill and Sabharwal (2023) 证明，采用对数精度算术的固定深度 transformer 可被一个一致的常数深度阈值电路模拟，从而把一次前向传播放进了 TC^0。因此，那些被认为真正需要串行计算的问题——迭代函数复合、任意布尔公式的

求值、大多数动态规划——无论模型多宽，都超出了单次前向传播的能力范围，除非人们广泛相信的那些电路复杂度分离结论不成立。

输出中间词元改变了这笔账。每个生成的词元都被写回上下文，并被下一次前向传播读取，因此一条 m 个词元的链给出了 m 轮深度为 L 的串行计算，其间还有一块无界且可寻址的草稿纸。残差流在每个词元边界处被丢弃；词元序列则不会。Merrill and Sabharwal (2024) 把这一点精确化为允许步数的函数：对数量级的步数收益甚微，线性量级的步数收益极大，而多项式量级的步数恰好买到多项式时间内可判定的语言。Li et al. (2024d) 给出了构造性的补充——一个常数深度的 transformer 用 T 步可以模拟规模为 $O(T)$ 的布尔电路，于是词元预算直接换算成电路规模——而Feng et al. (2023a) 则在具体的算术与动态规划任务上给出了分离结果。

定义 10.1 (思维链)**.** 一条回复被分解为 $y = (c, a)$，其中 a 是可抽取的答案，c 是在它之前输出的一切。思维链 c 不被任何验证器打分，也不被任何用户阅读；它唯一的功能是在产生 a 时被作为条件。相应地，$\mathbb{P}[a \text{ correct}] = \mathbb{E}_{c \sim \pi_\theta(\cdot \mid x)}[\mathbb{P}[a \text{ correct} \mid x, c]]$，因此加长 c 能提升准确率，其程度恰恰取决于条件项上升是否快过 c 的边缘分布退化。

关于这一机制是计算性而非语义性的，最干净的证据是这些词元有时根本不需要有任何含义。Pfau et al. (2024) 报告，在合成任务上，一个被训练成在作答前输出无意义填充词元（“. . .”）的 transformer，能解出没有这些词元时解不出的问题——不过要学会使用填充词元需要稠密的步级监督，因此该技术并不实用。中间词元就是串行计算；它们读起来像不像推理，是另一个问题。

10.1.2 提示揭示了推理本就潜藏其中

这条经验线索很短，值得按顺序了解。Nye et al. (2021) 微调模型，使其在算法任务上输出中间状态，从而把草稿纸确立为一种机制。Wei et al. (2022b) 表明根本不需要微调：只要几个答案中包含完整推理过程的示例，在足够规模下就能诱发同样的行为。Kojima et al. (2022) 连示例也去掉了，只在提示后附上「让我们一步一步思考」，就在零样本下恢复了大部分效果。Wang and Zhou (2024) 把提示整个去掉：考察 第一个解码词元的其他 top-k 选择，就能浮现出模型中本已潜藏的思维链路径，而这些路径对最终答案的置信度高于贪心路径。

历史注记：被发现的是诱发方式，不是能力

按顺序读下来，2021–2024 年的提示文献是一串递减的干预手段却产生同样的效果，这正是一种「本就存在、只是条件给错了」而非「本不存在」的能力的特征。贪心解码是对模型能做什么的一个糟糕估计量，因为一个问题的众数续写是简短答案，而简短答案排除了串行计算。这就是节 1.1把提示看作隐式后训练的说法，它也引出了 节 10.8的战略性问题：一旦你知道某个行为是潜藏的且有用的，你是继续付出词元去条件化它，还是把它搬进权重？

10.1.3 思维链不是什么

还有一个弱得多、却被假定得频繁得多的主张：c 是对产生 a 的那次计算的忠实记述。它不是，而且证据是直接的。Turpin et al. (2023) 把带偏见的特征插入提示——例如重排多选项，使正确答案总是「(A)」——发现答案会随之改变，而陈述出来的推理从未提及那个真正决定答案的特征，反而为带偏见的选择构造出一套看似合理的理由。Lanham et al. (2023) 直接干预思维链，对步骤做截断、破坏与改写，发现答案对陈述出来的推理的依赖程度随任务与规模大幅变化。Chen et al. (2025f) 报告，对于被明确训练成产生长推理轨迹的模型，情况同样如此。

常见陷阱：思维链是一次计算，不是一份解释

两个主张常被混为一谈。(*1*) 中间词元增加了模型能计算的东西：这一点在理论上与经验上都已确立。(*2*) 中间词元描述了模型是如何算出答案的：这一点常常为假，且绝不可假定。把两者混淆，会让人过度信任附在错误答案上的一段写得漂亮的推导，又过度不信任一个陈述推理粗糙的正确答案。它还决定了阅读思维链是否是一种可行的安全机制，这是一个仍在争论中的活跃问题，将在第25章中讨论。

10.2 推理计算这条轴

把一个问题的推理预算定义为在回答它的过程中生成的词元总数，涵盖所有样本、修订与被丢弃的分支。对一个有 N 个参数的稠密模型，每个词元约为 $2N$ FLOPs，因此在固定模型内词元与计算可以互换。这条轴值得仔细定义，因为所有有意思的主张都是在预算匹配的前提下成立的，而在预算不匹配时把某种搜索过程与单样本解码相比较，什么也没有比较出来。

表 10.1: 花掉同等规模推理预算的五种方式，以及它们为何不能互相替代。「上限」指在策略固定的情况下，预算无限增长时该策略所收敛到的水平；注意只有精确验证器那一行的上限等于覆盖率。并行性这一列之所以重要，是因为并行样本可以批处理，因而在时延上几乎免费，而单条更长的链则受时延约束。

模式	这些词元买到什么	可并行	上限	典型失效
单条更长的链	单一路径上的串行计算深度	否	该策略在其最佳长度处的准确率	漂移、重复、在简单问题上过度思考
n 个样本 + 多数投票	对答案分布的覆盖	完全	答案分布的相对多数准确率	数十个样本内即饱和；要求答案之间可比较
n 个样本 + 学习到的评分器	覆盖，加上选择	完全	评分器在该策略成功率处的精确率	过度优化评分器；关于 n 非单调
n 个样本 + 精确验证器	覆盖，加上选择	完全	pass@n	仅在第1章的不对称性成立之处可用
顺序修订	以一次尝试为条件的纠正	否	受限于发现自身错误的能力	会把原本正确的答案改坏
在步骤上做树搜索	集中于优质前缀的覆盖	层内可并行	受限于值函数精度	值误差逐层累积；多样性丧失

花掉这份预算有五种方式，汇总于表 10.1；在词元数相等时它们并不可以互换，因为它们买到的东西不同：单条路径上的串行深度、对答案分布的覆盖、对一次已作出的尝试的纠正，或是集中在某个值估计所偏好的前缀上的覆盖。

10.2.1 曲线的形状，以及形状从何而来

关于这条轴，被引用得最多的事实是：在若干个数量级的范围内，准确率随样本数平滑地、大致对数线性地上升 (Brown et al., 2024)。这种平滑性是求平均带来的假象，而非任何单个问题的性质，值得看清其中缘由。

对一个每次采样成功概率为 p 的固定问题，覆盖率是 $\text{pass@}k = 1 - (1-p)^k$：这是向 1 的指数式逼近，在 $k \approx 3/p$ 时基本就完成了。其中毫无对数线性可言。在一个成

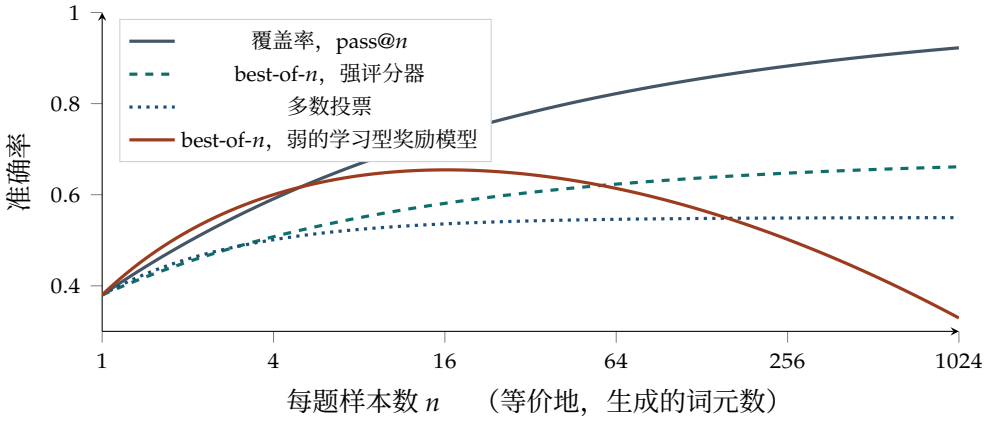


图 10.1: 示意图。本章讨论的四种定性形状，取自为便于辨读而任意选定的函数形式。两条坐标轴上的任何数字都不是测量值，都不应被引用。真正有文献支持的内容是序关系上的，并在正文中给出：覆盖率在数个数量级上持续上升 (Brown et al., 2024)；多数投票远低于覆盖率且很早就饱和 (Wang et al., 2023a)；用一个好评分器做选择能紧随覆盖率，且相对投票的差距不断拉大 (Lightman et al., 2024)；而用学习到的奖励模型做选择则是非单调的，先达峰后下降 (Gao et al., 2023a; Stroebl et al., 2024)。

功概率按 $f(p)$ 分布的基准上求平均，得到

$$\overline{\text{pass@}k} \;=\; 1 - \mathop{\mathbb{E}}_{p\sim f}\!\left[(1-p)^k\right], \tag{10.1}$$

而若 f 在零附近的行为像 $a\,p^{a-1}$——i.e., 该基准带有一条由极难但并非不可能的问题构成的幂律尾——那么 $\mathbb{E}[(1-p)^k] \approx \Gamma(a{+}1)\,k^{-a}$，于是 $1-\overline{\text{pass@}k}$ 按幂律衰减，其指数由近乎不可能的问题的密度决定。这一分析 (Schaeffer et al., 2025) 有一种有益的祛魅作用：那条著名的双对数直线度量的是你的基准的难度分布，而不是模型的某种缩放性质；两个平均准确率相同但尾部不同的基准会给出不同的指数。

覆盖率是一个上界，而不是可实现的准确率。真实达到的准确率是覆盖率与某个选择规则的复合，而所有的困难都住在选择规则里。图 10.1勾勒出四种定性形状：覆盖率无限上升、精确验证器紧随其后、多数投票早早饱和，以及学习到的评分器先升后掉头。

最后那条曲线的非单调性是图中最重要的东西，将在节 10.4.3中展开。它也说明了为什么「推理缩放」类的主张必须写明其选择规则：在验证器不完美的基准上——多数智能体式基准都是如此，那里的「验证器」是一套带假阳性的测试集——重采样的收益会很快变平甚至反转，因为至少抽到一个假阳性的概率随 n 上升 (Stroebl et al., 2024)。

10.3 无验证器的聚合：自洽性

自洽性 (Wang et al., 2023a) 是这条轴上最便宜而有用的做法。在非零温度下采样 k 条链，从每条中抽取最终答案，返回得票最多的答案。它不需要额外的模型、不需要标签、也不需要验证器；它在算术与常识类基准上大幅提升了思维链的准确率，并且在两年时间里都是报告推理成绩的默认做法。

10.3.1 它为什么有效，以及止步于何处

其机制在于错误与正确答案在分布上的一种不对称。对链求边缘，得到模型的答案分布 $q(a \mid x) = \sum_c \pi_\theta(c, a \mid x)$。推理可以出错的方式非常多，而每一种出错方式往往产生它自己的那个错误答案，因此错误答案上的质量被摊得很薄；而若干条正确的推导都终止于同一个数，它们的质量便集中起来。投票是在估计 q 的众数，只要上述集中性论证成立，这个众数就是正确的。

推导: 多数投票是一个众数估计量，所以它便宜，也有一个固定的天花板

记 $p_1 = q(a^\star \mid x)$ 为正确答案上的质量，$p_2 = \max_{a \neq a^\star} q(a \mid x)$ 为最强竞争者上的质量。对 k 个 i.i.d. 样本投票返回的是经验众数。把 Hoeffding 不等式用于两个计数之差，投票返回错误答案的概率至多为 $\exp(-k(p_1 - p_2)^2/2)$ 乘以竞争者的个数，于是

$$\text{maj@}k \;\longrightarrow\; \mathbb{1}[p_1 > p_2] \qquad \text{at a rate exponential in } k(p_1 - p_2)^2. \tag{10.2}$$

由此得到两个方向相反的后果。

便宜。所需样本量 $k = O((p_1 - p_2)^{-2})$ 并不随答案空间的大小增长。只要正确答案已经是得票最多的答案，几十个样本就足够了，其余的都是浪费。

有上限。在(10.2)中令 $k \to \infty$ 并对问题求平均，

$$\lim_{k \to \infty} \text{maj@}k \;=\; \mathbb{P}_x\big[\arg\max_a q(a \mid x) = a^\star(x)\big], \tag{10.3}$$

这是该策略的一个固定性质。在任何一个正确答案位于支撑集之内、但不是众数的问题上，投票对所有 k 都是错的，而 pass@k 最终会成功。因此自洽性会在远低于覆盖率的位置饱和，而这个差距恰恰就是那些模型能找到正确答案却无法偏好它的问题的集合。那个集合正是验证器存在的理由。

这干净地解释了经验上的现象。Wang et al. (2023a) 报告大部分收益在几十个样本之内就已到手，而Brown et al. (2024) 报告覆盖率在成千上万个样本处仍在攀升；两者都正确，只是在度量不同的泛函。它还解释了一种值得命名的失效模式：在奖励最大化把策略集中之后，在它本已解出的问题上 $p_1 - p_2$ 变大，而在正确答案原本稀少的地方正确众数被摧毁，于是 maj@k 曲线变平而 pass@1 提升。这就是从选择规则的视角看到的 节 14.2中的 pass@k 退化。

10.3.2 加权、温度与归一化

有三项实用的改进值得关注，其中一项占主导地位。

加权投票。 不是数票数，而是把每个样本的权重加起来。序列对数概率是显而易见的选择，同时也是糟糕的选择，因为它有长度偏好（节 2.1）且校准很差。验证器分数要好得多：Li et al. (2023a) 用一个感知步骤的验证器给投票加权，报告了相对无权投票的增益，这也是节 10.3与节 10.4之间的历史桥梁。自报置信度介于两者之间——Fu et al. (2025b) 用词元级置信度在线过滤并终止弱轨迹，报告在同等准确率下大幅减少词元；而Prabhudesai et al. (2025) 提出了更强的、仅有单篇证据的主张：只用置信度本身就足以充当训练奖励。

温度。 存在一个内部最优点，且它随 k 移动。在 $\curlywedge \to 0$ 时所有样本重合，投票退化为贪心解码。在高 $\curlywedge$ 下每个样本的准确率下降，且 q 的众数可能移离正确答案，而由(10.3)可知这是致命的，而不只是带来噪声。较大的 k 既能容忍也会奖励更高的温度，因为方差惩罚会被平均掉，而覆盖率上的收益不会。因此把温度与 k 联合扫描并非可选项，而在方法之间 k 不同却固定同一个温度所做的比较是混杂的。

答案归一化。 这才是实现工作量真正的去处，而它常常被少报。投票需要一个定义在答案上的等价关系，而每一次未能把同一答案的两种写法认作同一个，都会分裂一次投票。$\frac{\sqrt{2}}{2}$ 与 $\frac{1}{\sqrt{2}}$、0.5 与 $\frac{1}{2}$、$\pi/3$ 与 60°、「$x = 2$」与「2」：每一处发生的地方都会悄悄地把有效样本量减半，而由于这些错误与答案类型相关，它们是在给基准引入偏差，而不只是增加噪声。这就是验证器不完备性（第11章）换了一副面孔，也应当受到同样的对待：手工标注几百个样例并测量。在答案是自由形式、且不存在归一化器的场合，Chen et al. (2023b) 让模型在样本之中挑选最一致的那条回复，以引入一个评判模型为代价，找回了部分效果。

代码 10.1: 带答案归一化与弃答的自洽性。决定这套做法是否奏效的是归一化器，而不是投票；之所以返回 `margin`，是因为它正是 节 10.8.2中自适应停止规则所需要的自然统计量。

```python
import re
from collections import Counter
from fractions import Fraction

_BOX = re.compile(r"\\boxed\{([^{}]*(?:\{[^{}]*\}[^{}]*)*)\}")

def extract_answer(text):
    """Take the LAST boxed span: models restate, and the final claim is the one."""
    spans = _BOX.findall(text)
    return spans[-1].strip() if spans else None

def normalize(ans):
    """Collapse forms that differ only in presentation.  Under-normalizing splits
    one answer across several clusters and silently weakens the vote."""
    s = ans.replace(" ", "").replace("\\!", "").replace("\\,", "")
    s = s.replace("\\left", "").replace("\\right", "").strip("$")
    s = re.sub(r"^\\text\{(.*)\}$", r"\1", s).rstrip(".")
    s = re.sub(r"(\d),(\d{3})", r"\1\2", s)                  # 1,000 -> 1000
    s = re.sub(r"^\\[dt]?frac\{(.+?)\}\{(.+?)\}$", r"\1/\2", s)
    try:
        return str(Fraction(s))                              # 规范化的有理数
    except (ValueError, ZeroDivisionError):
        return s.lower()

def self_consistency(samples, weights=None):
    """samples: full generations.  weights: optional per-sample score (verifier
    probability, or confidence).  weights=None gives plain maj@k."""
    votes = Counter()
    for i, text in enumerate(samples):
        a = extract_answer(text)
        if a is None:
            continue                         # 弃答；绝不为空字符串投票
        votes[normalize(a)] += 1.0 if weights is None else float(weights[i])
    if not votes:
        return None, 0.0
    top, mass = votes.most_common(1)[0]
    return top, mass / sum(votes.values())               # 答案，间隔
```

10.4 有验证器的聚合：best-of-n

best-of-n 采样从一个基座策略抽取 n 条回复，并返回评分器最喜欢的那一条。它是本书中最重要的单一基线，理由有三：在策略冻结、且手上有一个评分器的条件下，它是你能做的最强的事；它在解析上的可处理性是后训练中几乎任何其他方法都不具备的；而强化学习最好被理解为对它的摊销。

定义 10.2 (Best-of-n)**.** 给定 π_{base}、一个评分器 $s : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ 以及预算 n，抽取 $y^{(1)}, \dots, y^{(n)} \overset{\text{i.i.d.}}{\sim} \pi_{\text{base}}(\cdot \mid x)$ 并返回 $y^{\dagger} = \arg\max_i s(x, y^{(i)})$，平局时均匀随机打破。记 π_n 为由此在 $y^{\dagger}$ 上诱导出的分布。

评分器可以是一个在解级正确性上训练出来的学习型结果奖励模型 (Cobbe et al., 2021)、一个过程奖励模型 (Lightman et al., 2024)、一个在自己的思维链之后输出裁决词元的生成式验证器 (Zhang et al., 2025c)，也可以是一段精确的程序。最后一种情形值得一提：若 s 是可靠且完备的，best-of-n 恰好达到 pass@n，覆盖率被完全兑现。因此下面关于天花板与非单调性的一切，都是关于学习型评分器与精确评分器之间那个差距的陈述；这个主题之所以存在，只是因为对我们关心的大多数事情而言，精确验证器并不存在。

10.4.1 best-of-n 的 KL 代价

best-of-n 改变了模型采样所依据的分布。改变了多少？答案很不寻常：它既不依赖于奖励，也不依赖于策略。

命题 10.3 (best-of-n 的 KL 代价)**.** 设分数 $s(x, y)$ 在 $\pi_{\text{base}}(\cdot \mid x)$ 下具有连续分布，使得平局以概率零发生。则 $\pi_n \ll \pi_{\text{base}}$，其密度比为

$$\frac{d\pi_n}{d\pi_{\text{base}}}(y) \;=\; n\,F(s(x,y))^{\,n-1}, \qquad F(\cdot) = \textit{CDF of } s \textit{ under } \pi_{\text{base}}, \tag{10.4}$$

并且

$$\mathbb{D}_{\text{KL}}(\pi_n \,\|\, \pi_{\text{base}}) \;=\; \log n - \frac{n-1}{n}. \tag{10.5}$$

推导: 定理 10.3的证明，以及它意味着什么

令 $u = F(s(x,y))$。因为 F 是一个连续随机变量的 CDF，所以在 π_{base} 下 $u \sim \text{Unif}[0,1]$。best-of-n 返回分数最大的样本，也就是 u 最大的样本，而 n 个 i.i.d. 均

算法 10.1 使用结果奖励模型的 best-of-n。实践中真正要紧的三行，恰恰是伪代码通常省略的那三行：被截断的样本应当被排除而不是被打分；平局应当均匀随机打破而不是按下标打破；以及定理 10.3中的 KL 代价要被记录下来，这样结果才能与一次强化学习训练放在同一条轴上。

输入： 基座策略 π_{base}，评分器 s，提示 x，预算 n，温度 $\mathcal{T}$，平局容差 ϵ_{tie}

$C \leftarrow \emptyset$
for $i = 1$ **to** n **do**
  在温度 $\mathcal{T}$ 下抽取 $y^{(i)} \sim \pi_{\text{base}}(\cdot \mid x)$
  if $y^{(i)}$ 以 `<eos>` 终止 **then**
    $C \leftarrow C \cup \{y^{(i)}\}$ ▷ 被截断的样本没有可供打分的答案
  end if
end for
if $C = \emptyset$ **then**
  return ABSTAIN
end if
对所有 $y^{(i)} \in C$ 计算 $\sigma_i \leftarrow s(x, y^{(i)})$ ▷ 一次批处理的前向传播
$\mathcal{W} \leftarrow \{y^{(i)} \in C \,:\, \sigma_i \geq \max_j \sigma_j - \epsilon_{\text{tie}}\}$
$y^{\dagger} \sim \text{Unif}(\mathcal{W})$ ▷ 当 s 为二值时恒有 $|\mathcal{W}| = |C|$
记录 $|C|$、$|\mathcal{W}|$、$\max_j \sigma_j - \text{median}_j \sigma_j$ 以及 $\log n - \frac{n-1}{n}$
return $y^{\dagger}$

匀变量的最大值在 $[0,1]$ 上的密度为 nu^{n-1}。于是从 π_{base} 到 π_n 的测度变换只作用在 u 上，并把它乘以 nu^{n-1}，这正是(10.4)：在分数的一个水平集之内，best-of-n 完全不加区分；而跨水平集时，它按 CDF 做倾斜。因此

$$\begin{aligned} \mathbb{D}_{\text{KL}}(\pi_n \,\|\, \pi_{\text{base}}) &= \mathop{\mathbb{E}}_{\pi_n}\big[\log(nu^{n-1})\big] = \log n + (n-1)\int_0^1 nu^{n-1}\log u\,du \\ &= \log n + (n-1)\Big(-\tfrac{1}{n}\Big), \end{aligned} \tag{10.6}$$

其中用到了 $\int_0^1 u^{n-1}\log u\,du = -1/n^2$。这就是(10.5)。
有三点值得提取出来。

它是普适的。奖励消失了。任何没有平局的评分器都产生同样的 KL，因为 best-of-n 只用到分数的秩，而 n 次可交换抽样的秩统计量对被排序的是什么一无所知。这正是 (10.5)可以充当一条 x 坐标的原因。

它以对数速度增长。要达到 d 奈特需要 $n \approx e^{d+1}$：五奈特约需 400 个样本，十奈特约需 60,000 个。best-of-n 沿奖励–KL 前沿行走，每奈特所需样本数呈指数增长，这才是对它的最优性的诚实读法，也正是我们想要一个摊销版替代方案的全部理由。

正向的方向不一样。把两者角色对调做同样的计算，得到 $\mathbb{D}_{\mathrm{KL}}(\pi_{\text{base}} \,\|\, \pi_n) = n - 1 - \log n$，它是线性增长的，而这恰好是 (2.13)中的 k_3 函数在 $u = n$ 处的取值。从基座策略走到 best-of-n 很便宜，从它走回来却很昂贵——这是节 2.4中寻找众数的不对称性，出现在人们没料到的地方。

有一个警告在实践中很要紧，而且分量很大。当评分器取连续值时，平局是零测度事件，(10.5)基本上精确成立。而当评分器被量化，尤其是当它是一个二值验证器时，平局是常态，(10.5)就变成了一个上界 (Beirami et al., 2024)。二值情形是可以精确计算的，而且很有启发。设 p 为来自 π_{base} 的一个样本正确的概率。使用一个可靠且完备的验证器的 best-of-n，在存在正确样本时从正确样本中均匀抽取，否则从错误样本中均匀抽取，于是

$$\begin{aligned}\mathbb{D}_{\mathrm{KL}}(\pi_n \,\|\, \pi_{\text{base}}) &= (1-(1-p)^n)\log\frac{1-(1-p)^n}{p} + (1-p)^n(n-1)\log(1-p) \\ &\xrightarrow[n\to\infty]{} \log\frac{1}{p}.\end{aligned} \tag{10.7}$$

这个散度在 $\log(1/p)$ 处饱和，一旦说破就很直观：二值验证器只能把策略集中到正确集合上，再无更进一步的余地，而这样做的代价就是该集合原有质量的对数。在一个策略有 10% 概率解出的问题上，针对一个二值验证器采样 1024 次的代价是 2.30 奈特，而不是(10.5)所暗示的 5.95。图 10.2把两者都画了出来。

10.4.2 为什么这让奖励–KL 图变得有意义

定理 10.3正是对齐文献中每一张奖励对 KL 图的 x 轴之所以成立的依据，其中包括图 5.2以及Gao et al. (2023a) 的过度优化缩放律。那些图所解决的问题是「你优化得有多狠」在方法之间不可比：一次 RL 训练由 β 与步数参数化，一次直接对齐训练由 β 与

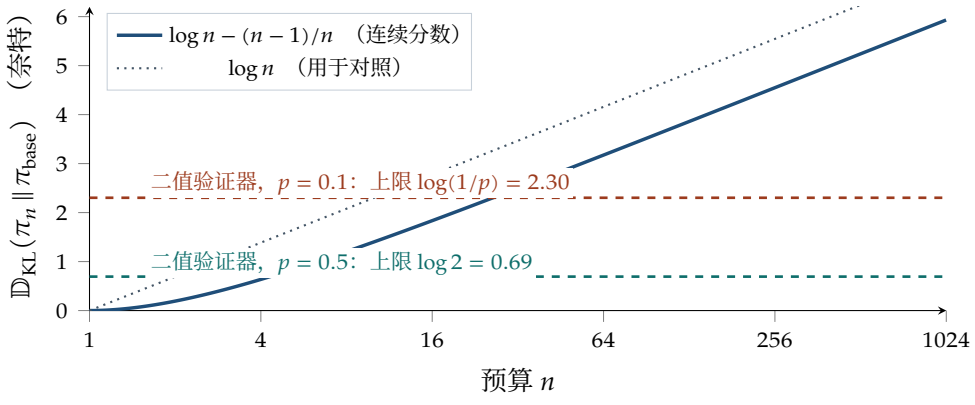


图 10.2: best-of-n 的 KL 代价是可以精确计算的，因此本图并非示意图。实线是 (10.5)给出的 $\mathbb{D}_{\mathrm{KL}}(\pi_n \,\|\, \pi_{\mathrm{base}}) = \log n - \frac{n-1}{n}$，它对任何在基座策略下具有连续分布的评分器都成立，且与评分器和策略均无关。两条水平线标出的是 $\log(1/p)$，即在策略以概率 p 解出的问题上使用二值验证器时 (10.7)的 $n \to \infty$ 极限：在均匀打破平局的情况下，再多的采样也无法把策略集中到正确集合之外，因此可达到的散度是有上限的。实践上的读法是：(10.5)是奖励模型重排序的正确记账方式，而对验证器过滤来说则是一个宽松的上界。

轮数参数化，best-of-n 由 n 参数化，而这些单位之间无法换算。但每一种方法都会产生一个与 π_{ref} 相距某个散度的策略，而散度是一种通用货币。对 RL 你需要测量它；对 best-of-n 你只凭 n 就能得到闭式解，无需任何实验。这就是 best-of-n 成为衡量其他一切的参照曲线的原因，也是Gao et al. (2023a) 能够把 (5.11)与(5.12)拟合在同一组坐标上的原因。

这一比较对 best-of-n 也是有利的，而这是第二个解析结果。Yang et al. (2024a) 从信息论上刻画了最优的奖励–KL 权衡，并证明 best-of-n 在大偏差意义下以主导阶达到了它：在 $n \approx e^d$ 时，π_n 的期望奖励与(6.15)中受 KL 约束的最优解 $\pi^\star \propto \pi_{\mathrm{base}} e^{r/\beta}$ 相差一个趋于零的量；Beirami et al. (2024) 给出了有限 n 的对应结果。因此 best-of-n 并不是一个会被更好方法全面压过的粗糙基线；它基本上就在前沿上。它欠缺的不是统计效率，而是摊销：它用样本来支付 KL，每一次查询都要付，永远要付。这一观察正是 best-of-n 蒸馏这条线索 (Sessa et al., 2024; Gui et al., 2024) 以及Balashankar et al. (2024) 的推理感知目标的全部内容，两者都在第8章中讨论。

10.4.3 代价：best-of-n 优化的是评分器

以上一切讲的都是 KL 代价；没有任何一条说被选中的样本是好的。best-of-n 最大化 s，而 s 是一个代理量，因此它会像 RL 一样发生古德哈特效应——差别在于它的优化压力事先就被 n 量化了。

推导：一个不完美的排序器让你损失的是一个指数，而不是一个常数

把评分器建模为 $s = r^\star + \varepsilon$，其中在 π_{base} 下 $r^\star \sim \mathcal{N}(0, \sigma_r^2)$ 与 $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ 相互独立，于是抽取 n 次时 $\mathbb{E}[s_{\max}] \approx \sqrt{2\log n}\,\sqrt{\sigma_r^2 + \sigma_\varepsilon^2}$。由联合正态性可得 $\mathbb{E}[r^\star \mid s] = \frac{\sigma_r^2}{\sigma_r^2+\sigma_\varepsilon^2}\, s$，因此被选中样本的真实质量为

$$\mathbb{E}[r^\star(y^\dagger)] \;\approx\; \rho\,\sigma_r\sqrt{2\log n}, \qquad \rho \;=\; \frac{\sigma_r}{\sqrt{\sigma_r^2 + \sigma_\varepsilon^2}} \;=\; \operatorname{Corr}(s, r^\star). \tag{10.8}$$

一个完美的排序器在预算 n' 下达到 $\sigma_r\sqrt{2\log n'}$，因此预算为 n 的不完美排序器相当于完美排序器的

$$n' \;=\; n^{\rho^2} \tag{10.9}$$

个样本。损失发生在指数上：一个相关系数为 $\rho = 0.7$ 的评分器把 1024 个样本变成实际有效的 30 个，而 $\rho = 0.5$ 时变成 5.7 个。仅此一个公式就解释了为什么重排序对验证器质量如此敏感、为什么实践中 n 的边际收益看起来比覆盖率曲线所暗示的更平，以及为什么这个领域把力气花在验证器上而不是更大的 n 上。

请注意这个模型没有预测到的东西：它关于 n 是单调的，因而无法解释 图 10.1中的掉头。单调性依赖于 σ_ε 为常数，而它并不是常数。在 n 个候选上取最大值，会在回复空间中越来越极端的区域上取值，而那里恰恰是评分器训练数据最少的地方，因此误差方差随候选的极端程度而增长。重尾误差或依赖于候选的误差才是产生非单调性的原因，这也是为什么这一计算应当被读作乐观情形。

二值版本出自Stroebl et al. (2024)，更为简单。假设验证器总是接受正确样本，并以假阳性率 $\eta > 0$ 接受错误样本。当 n 很大时两个被接受的池子都非空，返回样本的准确率趋向于验证器在该策略工作点处的精确率，

$$\lim_{n\to\infty} \mathbb{P}[y^\dagger \text{ correct}] \;=\; \frac{p}{p + (1-p)\eta}, \tag{10.10}$$

它与 n 无关，且严格小于 1。重采样买不到超越验证器假阳性的东西；它只能买到那个上限，而且很快就到。

常见陷阱：针对学习型奖励模型的 best-of-n 不是一个免费的上界

人们常把「用我们的奖励模型做 best-of-64」当作策略能力的上界来报告。它在任何一个方向上都不构成界。它低于天花板，因为奖励模型是一个不完美的排序器（节 10.4.3）；它又可能高于策略在 RL 之后所达到的水平，因为一次性选择从不给策略机会移动到奖励模型的误差区域里去。更糟的是，它关于 n 非单调，所以你报告的数值取决于一个你多半是看着结果挑出来的超参数。请报告关于 n 的整条曲线，并用一个留出的评判模型而不是正在被优化的那个评分器来评；如果你必须给一个数字，就用配精确验证器的 pass@n，那才是一个真正的上界。

代码 10.2: 使用奖励模型的 best-of-n。平局处理才是关键：在二值或量化评分器下，大多数组在最高分处完全并列，而 `argmax` 返回的是最小下标，这会悄悄地让报告的数值依赖于生成顺序。均匀随机打破平局才是定理 10.3所假定的做法。

```python
import math, torch

@torch.no_grad()
def best_of_n(policy, reward_model, tokenizer, prompt, n,
              temperature=1.0, max_new_tokens=4096, tie_atol=1e-6, gen=None):
    out = policy.generate(prompt, num_return_sequences=n, do_sample=True,
                          temperature=temperature, max_new_tokens=max_new_tokens,
                          generator=gen)
    texts = [tokenizer.decode(o, skip_special_tokens=True) for o in out]

    # 被截断的样本不是候选：它们没有可供打分的最终答案。
    cand = [i for i, o in enumerate(out) if o[-1] == tokenizer.eos_token_id]
    if not cand:
        return None, {"n_cand": 0}

    scores = reward_model(prompt, [texts[i] for i in cand]).float()   # (|cand|,)
    top = scores.max()
    tied = [cand[j] for j in (scores >= top - tie_atol).nonzero().flatten().tolist()]
    pick = tied[int(torch.randint(len(tied), (1,), generator=gen))]

    return texts[pick], {
        "n_cand": len(cand),
        "n_tied_at_top": len(tied),          # 对二值验证器而言等于 n_cand
        "score_margin": (top - scores.median()).item(),
        "kl_nats": math.log(n) - (n - 1) / n,   # 上界；无平局时精确
    }
```

10.5 验证器比生成器更值

这一批文献的核心经验主张是一个资源分配主张：在总参数量或总计算量固定的条件下，把预算的一部分花在验证器上并做搜索，在很宽的范围内胜过把预算全部花在一个更大的生成器上。它值得谨慎对待，既因为它是让第三部分的整个纲领值得推进的那个主张，也因为它常常被说得比证据所支持的更强。

10.5.1 证据

Cobbe et al. (2021) 给出了奠基性的结果：在解级正确性上训练一个验证器，并对一个 6B 模型的 100 个样本做重排序，在小学数学上达到或超过了一个经过微调的 175B 模型，相当于用样本加一个小验证器买到了大约 30× 的参数差距。Lightman et al. (2024) 沿着改进评分器而非改进生成器的方向把它推进了一步，报告在相同 n 下，过程监督的奖励模型挑出正确解的频率显著高于结果监督的奖励模型和多数投票，且差距随 n 拉大——这正是 节 10.4.3对更高 ρ 所预测的特征。Snell et al. (2025) 把这一比较做成了 FLOPs 匹配的分配研究，报告在简单与中等难度的问题上，一个花费测试时计算的较小模型胜过一个大约 14× 的更大模型，而在最难的问题上更大的模型获胜——这是一个交叉，而不是一边倒的胜利。Wu et al. (2025e) 得到了相容的结论，而 Liu et al. (2025e) 给出了最极端的版本：一个 1B 模型在 MATH-500 上超过了一个 405B 模型，且最优策略同时取决于策略、奖励模型与难度。有两项细化值得记住。Zhao et al. (2025c) 表明验证本身也是一个值得花计算的地方，通过采样并聚合多次自我验证，可以让搜索在原本会停滞的地方继续扩展；Hosseini et al. (2024) 表明被丢弃的样本并非浪费，因为在自生成的正确解与错误解上一起训练一个验证器，胜过只用正确解去微调生成器。

10.5.2 机制

为什么把参数用在验证器上会比用在生成器上更划算？因为验证是一个比生成更容易的学习问题，而且这一差距是结构性的，而不是偶然的。

核心思想: 生成–验证差距是贯穿全书的那条主线

生成一条好回复需要对 $|\mathcal{U}|^T$ 条序列上的分布建模，并通过搜索找到它的高质量区域。判断一条回复则只需从一个完全可见、连所有中间步骤都摆在眼前的对象上算出一个比特。第二个问题的样本复杂度更小，需要的容量更少，并且每单位数据能提供多得多的监督。Swamy et al. (2025) 把这一点作为对下述问题的明确回

表 10.2: 在生成器与选择器之间分配固定的推理期 FLOP 预算。所引研究呈现的规律是一次交叉而非一致的排序：在小生成器以不可忽略的概率能解出的问题上，选择占优；在它解不出的问题上，大生成器占优。证据层级按正文中的定义：已复现指多个独立团队，单篇指一项细致的研究，尚未在 2026 年的规模上被重复。

分配方式	它买到什么，又买不到什么	何时获胜	层级
全部投入生成器；一个贪心样本	每样本质量最高；没有覆盖，也没有选择	预算只够一个样本	—
~ 14× 更大的生成器，一个样本	在每个问题上都抬高 p，包括那些 $p \approx 0$ 的问题	对小模型而言 p 接近于零	单篇 (Snell et al., 2025)
小生成器，n 个样本，多数投票	覆盖，不需要评分器，也不需要标签	正确答案已经是得票最多的答案	已复现
小生成器，n 个样本，结果奖励模型	覆盖加选择；代价是多一个模型	p 适中且评分器的 ρ 较高	已复现 (Cobbe et al., 2021)
小生成器，n 个样本，过程奖励模型	更好的 ρ，因而更好的有效指数 n^{ρ^2}	解可以分解为可核查的步骤	在重排序上已复现 (Lightman et al., 2024)
把预算在生成与验证计算之间切分	让验证器在下判断之前先做推理	验证比生成容易，但也并非平凡	单篇 (Zhang et al., 2025c; Zhao et al., 2025c)

> 答：为什么「先训奖励模型再做 RL」的两阶段流程会胜过瞄准同一最优解的直接方法——通过验证器来分解的价值是计算上的，而不是统计上的。第8章正是以这种形式把该论证归于他们，而它在全书中反复出现。它解释了为什么重排序胜过更大的生成器（节 10.5）；为什么 RLVR 效果如此之好，而那里的验证器甚至不是学出来的（第11章）；以及作为逆否命题，为什么在判断与做同样难的地方进展会停滞（第15章）。

有两条边界应当直说。这个差距依赖于任务，并且可能消失：对于开放式写作、研究品味以及长时域的计划，既没有廉价的验证器，也没有证据表明重排序能买到多少东西，因此这套分配论证根本不适用。此外，该论证针对的是学习问题，而不是推理成本。一个每题被查询 n 次、或对过程模型而言每个解被查询 M 次的验证器，在服务时并不便宜，而这正是推动这个领域走向摊销的两股压力之一（节 10.8）。

10.5.3 把结果评分器与过程评分器当作重排序器

节 5.7定义了这两族模型；这里我们只关心它们作为选择器的用法，作为训练信号的用法属于第13章。把两者分开很重要，因为证据是不对称的：过程奖励模型在重排序上的证据扎实且已被复现，而在训练信号上的证据则不然（节 13.2.3）。

过程模型返回步骤分数 $v_1, \dots, v_M$，而重排序需要一个标量，因此必须选定一种聚合方式。节 13.2.2把这些标准选择分析为关于解的结构的种种主张；针对选择这一用途的小结见表 10.3。取最小值是通常的默认做法，并且在每一步都必不可少时是有道理的，因为一个解只有在其最薄弱的一步正确时才正确，而且与取乘积不同，它不惩罚长度。取最后一步的分数等于把过程模型降格为结果模型，而取均值则很容易靠补上一些平凡正确的步骤来抬高。

这里有一个本章读者必须知道的测量陷阱。best-of-n 准确率是评估过程奖励模型的标准做法，而它是有偏的。Zhang et al. (2025e) 报告，在按 best-of-n 表现挑选出来的模型中，最小分数有很大一部分落在最终答案那一步上——模型已经变成了一个套着过程接口的结果验证器——并且过程模型会容忍那些经由有缺陷的推理得到的正确答案，从而抬高了这一指标。Zheng et al. (2025b) 与Song et al. (2025c) 直接度量错误定位，而在重排序上看起来很强的模型在那里往往表现平平。当你读到过程监督胜过结果监督的主张时，请核查那是由哪种评估得出的。

10.6 在部分解上做搜索

best-of-n 在看到任何东西之前就把全部预算押了出去。搜索则把生成与评估交错进行，因此预算可以在解被构建的过程中被重新调配。这显然是一个更好的主意，而它多半行不通，其中的缘由值得精确地理解。

10.6.1 这棵树是什么

固定一种把回复切分为步骤的方式——实践中是以换行分隔的一行、一个句子，或固定的词元数。一个节点就是一个部分解 $y_{\le m}$：一条提示加上 m 个已完成的步骤。它的子节点是候选的下一步，由策略一步一步采样得到；叶子是完整的解。一个值函数 $v(x, y_{\le m}) \approx \mathbb{P}[\text{eventual success} \mid x, y_{\le m}]$ 给节点打分，而搜索就是决定展开哪些节点的策略。

v 从哪里来？有四种答案，按可信度与代价递减排列。

表 10.3: 在 n 个样本的候选集上使用的选择规则，大致按其要求排序。每条规则都把一个集合归约为一个元素，而值得关注的是第三列：无论 n 多大，是什么固定的量给这条规则封了顶。只有最后一行以覆盖率为上限，而这正是第11章存在的理由。

规则	需要什么	关于 n 的上限	如何失效
相对多数投票	一个答案等价关系	相对多数正确率, (10.3)	票被不同写法分裂；自由形式答案
对数概率加权投票	除策略之外别无所需	同上，但经过重新加权	长度偏差与校准失当
置信度加权投票	词元级置信度	同上，但经过重新加权	在自信而错误的轨迹上置信度也很高
验证器加权投票	一个评分器；把两种信号结合	介于投票与 best-of-n 之间	继承两者的失效模式，通常较为温和
best-of-n，结果模型	一个训练好的评分器	评分器的精确率, (10.10)	在评分器误差上做极值选择
best-of-n，过程模型，`min`	步骤级评分器	同上，但 ρ 更高	被校准最差的那一步主导
best-of-n，精确验证器	一段可靠且完备的程序	pass@n	只对极少数任务存在

1. **从该前缀出发的蒙特卡洛采样**。从 $y_{\le m}$ 采样 K 条补全并取经验成功率：这是无偏的，也恰恰是我们想要的，但它消耗的正是搜索本想省下的预算，且标准误为 $\sqrt{v(1-v)/K}$。这正是自动过程标签 (Wang et al., 2024f; Luo et al., 2024) 以及节 13.4中训练期估计量背后的构造。
2. **一个过程奖励模型**，在上述标签或人工步骤标注上训练而成 (Lightman et al., 2024)。查询时便宜；但会带上其标注策略所具有的任何偏差。
3. **策略自身的自我评估** (Xie et al., 2023)。免费，而且与生成器的误差相关，按节 10.4.3这恰恰是摧毁 ρ 的那种性质。
4. **在训练循环内自举出来的学习型评论家**（第13章）。

10.6.2 三种做法

步骤级束搜索。 维护 b 条存活前缀；对每条采样 m 个步骤加以展开；用 v 给这 bm 个候选打分；保留前 b 个；重复。算法 10.2给出了带前瞻的版本，而它 $b\,m\,M\,(1+d)$ 次步骤生成的预算与 best-of-n 的 n 条完整解，正是两者相互匹配的方式。

前瞻。 与其直接给一个前缀打分，不如再向前推演 d 步再给结果打分，用生成来换取值估计偏差的下降。这是一种插值：$d = 0$ 是纯粹的束搜索，而 d 等于剩余长度时，每一次评估都变成一次结果判断，从而在叶子上退化为 best-of-n。因此 best-of-n 是「最大前瞻、零剪枝」这一极端，而「搜索是否胜过 best-of-n」问的是：在这条插值线上的哪个位置，值函数才准到足以用来剪枝。

蒙特卡洛树搜索。 按置信上界规则做选择 (Kocsis and Szepesvári, 2006; Rosin, 2011)，然后展开、评估，并沿访问路径回传，这使得展开次数是自适应分配的，而不是逐层平均分配的。语言模型上的实例化非常多：Yao et al. (2023a) 给出一般性框架，Hao et al. (2023) 把模型当作世界模型，Feng et al. (2023b) 给出把搜索与训练配对的 AlphaZero 式做法，Zhou et al. (2024) 给出智能体式版本，而Guan et al. (2025) 在小规模上报告了最强的结果。

10.6.3 诚实的评价

在词元预算匹配的条件下，在大多数已发表的比较中，精巧的搜索相对 best-of-n 只带来很小的增益，而且常常毫无增益。这是一个已被充分复现的失望结果，应当直说而不是藏起来。

Snell et al. (2025) 报告，在步骤上做束搜索在小预算和较难的问题上有帮助，但在更大的预算下不如 best-of-n，并把原因归于对验证器的过度优化以及存活束之间多样性的丧失。Wu et al. (2025e) 报告蒙特卡洛树搜索在计算匹配时不如更简单的策略，这正是促成他们提出奖励平衡替代方案的原因。DeepSeek-AI (2025) 报告他们尝试过树搜索并放弃了，理由有二：词元生成的分支因子远远大于棋类游戏，而且一个精细到足以引导它的值模型非常难训练。

其中的机制是一个误差累积论证，也是最该记住的那个。best-of-n 用 v 只做一次决策，而且是在完整的解上做——这是最容易评估的对象，因为一切都可见。束搜索则用 v 做 M 次不可逆的决策，每一次都作用在部分解上，而那正是值估计最困难的地方。如果正确前缀以概率 $1 - \epsilon$ 存活过一层，那么它存活 M 层的概率是 $(1 - \epsilon)^M$：搜索把每次决策的错误率转化为一个指数衰减的存活概率，而 best-of-n 只付出一次 ϵ。搜索会放大值函数的误差，因此只有在值函数很好时它才值得那份复杂度——而值函数恰恰是谁都没有的东西。

由此得出的推论并不是一个安慰奖：在值函数接近精确的地方，搜索确实获胜。形式化定理证明是最干净的例子，因为一个内核给出了可靠、完备且逐步的反馈，而在证

明状态上做树搜索是标准且表现最好的方法 (Ren et al., 2025; Chen et al., 2025a; Trinh et al., 2024)；带可执行中间断言的代码是一个部分成立的例子。搜索能买到的只是你的值函数的质量，仅此而已，所以研究问题从来不是「用哪种搜索算法」，而是「一个可信的前缀值从哪里来」，这就是第13章。

10.7 顺序修订与自我纠正

剩下的一种花词元的方式，是让模型看看自己的答案再试一次。Self-Refine (Madaan et al., 2023) 是其典型形式：生成、批评、修订、迭代；Reflexion (Shinn et al., 2023) 则加上了一段跨尝试的、以语言表述的自我批评的情节式记忆。两者都报告了增益，而决定这些增益是否真实的，是这个循环能否获取来自模型之外的信息。

10.7.1 谨慎陈述的否定性结果

Huang et al. (2024b) 考察了内在自我纠正——没有外部反馈、也没有关于答案是否错误的先知信号的修订——并报告在推理基准上它相对单次尝试降低了准确率。他们在方法论上的观点更为重要：此前若干正面结果曾用真值标签来决定何时停止修订，而这恰恰泄露了该过程本应自己提供的那一个比特。Stechly et al. (2023) 在图着色上报告了同样的结果，并给出了一个有用的对照：在循环中放入一个可靠的外部验证器，增益就回来了。Tyen et al. (2024) 定位了这一缺陷：模型不擅长找出出错的那一步，但一旦被告知错在哪里，它们是有能力改正的。失败在于检测，而不在于修复。

其中的机制是生成–验证差距反过来起作用。判断「我的答案错了吗」是一个验证问题，而模型对自身正确性的估计既校准很差，又与产生该答案的那个错误相关。如果模型能可靠地判断自己的哪些答案是错的，它就不会以那样的比率产出它们。用算术可以把门槛写清楚。设 p 为初始准确率，f_{r} 为一次修订改动一个正确答案的概率，f_{w} 为它改动一个错误答案的概率，κ 为被改动的错误答案变正确的概率。一轮修订有帮助当且仅当

$$(1-p)\,f_{\mathrm{w}}\,\kappa \;>\; p\,f_{\mathrm{r}}. \tag{10.11}$$

在 $p=0.4$ 时要求为 $f_{\mathrm{w}}\kappa > \frac{2}{3}f_{\mathrm{r}}$，这很容易；而在 $p=0.9$ 时它变成 $f_{\mathrm{w}}\kappa > 9f_{\mathrm{r}}$，要求修订触碰错误答案的可能性是触碰正确答案的九倍。那种量级的选择性恰恰是检测方面的结果所说模型缺乏的东西，这也解释了一个原本令人费解的现象：内在自我纠正在更强的模型上看起来更糟——策略越强，(10.11)中的门槛就越高。

表 10.4: 按错误信号的来源划分的修订循环。其规律随信号的可靠性单调变化，而这正是节 10.7的全部内容。最后两行是训练层面的干预而非推理过程，它们是把 节 10.8的摊销手法用在修订上。

循环中的信号	可靠性	报告的效果
无（只做自我批评）	无	降低推理准确率 (Huang et al., 2024b; Stechly et al., 2023)
先知式的「你的答案错了」	可靠，但部署时不可得	增益很大；但不构成一个过程 (Huang et al., 2024b)
提供错误位置	可靠	修复成功；瓶颈原本在检测 (Tyen et al., 2024)
执行或编译器反馈	对所测性质可靠	稳定的增益 (Chen et al., 2024e)
一个独立且更强的批评模型	部分可靠	增益随批评者而变，与修订者无关 (Olausson et al., 2024)
经由多轮 RL 训练得到	—	内在自我纠正成为可能 (Kumar et al., 2025; Qu et al., 2024)

常见陷阱: 内在修订与反馈驱动的修订是两种不同的过程

文献既报告了「自我纠正」带来的巨大增益，也报告了彻底的退化，而这种分歧几乎完全取决于循环里放了什么。带可靠外部信号的修订——一套测试、一段解释器回溯、一个编译器、一个精确验证器、一个更强的批评者——是有效的，而且并不神秘：那个信号提供了模型算不出来的那一个比特。不带外部信号的修订通常无效，而在高准确率的策略上，按 (10.11)它还会实实在在地损失准确率。阅读一项自我纠正的结果时，请先回答两个问题：是什么告诉模型它错了，又是什么告诉它何时停下。只要其中任一个答案是「真值标签」，这个结果就没有描述一个可部署的过程。

表 10.4的最后一行指向未来。Kumar et al. (2025) 论证，在自生成的修订轨迹上做监督微调之所以失败，是出于一个第3章中已熟悉的分布偏移原因——模型学会做最小的改动，因为训练分布奖励的就是这个——而一个带整形奖励的两阶段多轮 RL 过程能够产生真正的内在自我纠正；Qu et al. (2024) 通过在改进轨迹上做迭代微调到达了类似的位置。因此内在自我纠正并非不可能。它只是不免费：它必须被训练出来。

10.8 用训练计算换推理计算

我们现在有了一份带有量化代价的推理期做法清单，而战略性的问题是：究竟是去运行它们，还是把它们训练掉。两个方向都有真正的理由。

10.8.1 两种论证

支持摊销。 训练成本只付一次；推理成本每次查询都要付。如果把某个过程搬进权重需要 C_{train}，而每次查询省下 ΔC_{inf}，那么交叉点在 $N^\star = C_{\text{train}}/\Delta C_{\text{inf}}$ 次查询处，而对任何被广泛服务的模型，N 都比 $N^\star$ 大若干个数量级。只要摊销行得通，它在算术上就一定赢。这就是 节 13.8中的那笔计算，也是在节 1.1中把能力从提示搬进权重的那笔计算；它还反向延伸到预训练：Sardana et al. (2024) 表明，把推理需求计入之后，计算最优的模型会更小、训练得更久。

反对摊销。 搜索能因题制宜；权重不能。一个搜索过程在难题上会花更多，因为它会一直进行到验证器满意为止，而在简单题上则提前停止。一个被训练成会深思的策略，只依据提示来条件化它的长度分布，并且在两个方向上都会出错：Chen et al. (2025c) 记录了在平凡算术上的冗长深思，Wang et al. (2025f) 记录了在难题上过早放弃有希望的思路。还有一项更微妙的代价。把一个选择过程蒸馏进权重，复现的是那个过程的输出分布，而它比基座策略更集中，集中的程度恰好是定理 10.3中的那个 KL；因此被摊销的策略可供其自身 best-of-n 使用的多样性更少，推理缩放曲线也更平。这正是节 14.2中覆盖率退化背后的机制，它意味着即使只从能力的角度看，摊销也不是免费的。

历史注记: 专家迭代是这一模板的原型

把搜索摊销进一个策略，并非始于语言模型。专家迭代 (Anthony et al., 2017) 在两件事之间交替：把树搜索当作一个缓慢而强大的「专家」，以及把它的决策蒸馏进一个快速的策略网络，从而让下一轮搜索更强；AlphaGo Zero (Silver et al., 2017) 就是这一循环在没有任何人类数据的情况下跑到收敛。语言模型上的各个版本与它一一对应。STaR (Zelikman et al., 2022)、拒绝采样微调 (Yuan et al., 2023a) 以及 ReST (Gulcehre et al., 2023; Singh et al., 2024) 蒸馏的是配精确验证器的 best-of-n 的输出；BOND 与 BoNBoN (Sessa et al., 2024; Gui et al., 2024) 蒸馏的是配学习型奖励模型的 best-of-n；Stream of Search (Gandhi et al., 2024) 蒸馏的是搜索的

> 轨迹而非其输出，因此策略是在上下文中搜索，并且可以被改进到超过它所模仿的那个搜索器。这样读下来，第11章就是这个家族中去掉了模仿这一步的成员：奖励直接在策略自己的样本上做选择，从头到尾没有任何专家轨迹被写下来。

这一论证还有一个更尖锐的版本，它支持保留验证器。Setlur et al. (2025b) 把基于验证器的测试时策略与无验证器的策略——蒸馏长轨迹、自洽性、没有核查器的修订——加以比较，并证明了一个分离：随着预算增长，基于验证器的方法占优，而差距的大小由基座策略在可区分的「差一点就对」的解上放置了多少质量所控制。其直觉与节 10.3.1相同，因为无验证器的聚合只能取回已经编码在策略自身对答案的排序之中的信息。这也是为什么Setlur et al. (2025c) 能够训练探索，使得测试时计算可以外推到训练预算之外：被摊销的是搜索行为，而不是答案。

10.8.2 自适应分配

两种论证都指向同一个实践目标：在更难的问题上花更多计算。有四种机制，由最便宜到最费事。

答案已定时提早停止。Aggarwal et al. (2023) 维护一个关于哪个答案将赢得投票的后验，并在有信心时停机，报告在准确率代价可忽略的情况下大幅减少样本；代码 10.1返回的 `margin` 正是这类规则所消费的统计量，而 Fu et al. (2025b) 则以词元级置信度为门控，在生成中途掐掉弱轨迹。

在花掉计算之前先预测它的边际价值。Damani et al. (2025) 训练了一个逐输入的估计量来判断额外一单位计算值多少钱，并据此分配，用一小部分预算就达到了均匀分配的水平。这是原则性的版本，而它的困难在于这个预测器本身就是一个难度估计器，也就是值函数学习问题的再现。

控制预算，然后训练这个控制器。Muennighoff et al. (2025) 通过注入思考结束分隔符来强制终止，或通过压制它来强制继续，并报告这种做法能外推出准确率，而用提示去指定预算做不到这一点，因为模型数不清自己的词元（第3章）。Aggarwal and Welleck (2025) 随后用 RL 让长度指令真正可被遵循，把预算控制从一个解码期的小把戏变成一种策略能力——这是把摊销用在分配器而非搜索上，也是 2026 年的系统所走的方向。第23章会把长思维链的效率当作一个独立主题来处理。

工程实践: 如何花掉固定的推理预算，2026 年年中

- **没有验证器**。在与 k 联合扫描的温度下采样 16–32 条链，用一个你已经审核过的归一化器来投票，并自适应地停止。不要指望 $k \approx 32$ 之外还有什么；按(10.3)，那里什么也没有。
- **有精确验证器**。就用 best-of-n，不要更花哨的东西：你正在兑现 pass@n，那就是天花板。
- **有学习型评分器**。用 best-of-n，扫描 n 并依据一个留出的信号而不是评分器本身来选择，并预期非单调性。在上搜索之前先试试验证器加权投票——更便宜，通常也一样好——而只有当你的值函数是在前缀上被验证过、且你的步骤确实可核查时，才考虑搜索。
- **只在有外部信号时才做修订：**执行、测试、核查器，或一个更强的批评者。在准确率本已很高的策略上，绝不要用光秃秃的「检查一下你的答案」循环。
- **报告曲线，而不是点**。以生成的词元数为横轴画准确率，把所有方法放在同一条轴上。这批文献中几乎每一处分歧，在预算被匹配之后都会自行消解。

10.9 通往第三部分的桥梁

把这些拼起来。一个验证器可以在推理时对 n 个样本排序，从而兑现单样本解码所放弃的覆盖率。代价是每次查询 n 次生成，以及至多 $\log n - \frac{n-1}{n}$ 奈特的 KL 位移，而且在部署的整个生命周期里每次查询都要重付一遍。验证器质量是以指数而非常数的形式进入的，因此改进它比增大 n 更值。而在样本之间做选择之外的另一条路，是改变样本所来自的那个分布。

核心思想: 凡是验证器能排序的，它就能用来给奖励

best-of-n 从策略中抽取 $y^{(1)}, \dots, y^{(n)}$，对每一条计算 r_{ver}，返回最好的那条，并丢弃其余。一个组相对策略梯度抽取同样的 n 个样本，计算同样的 r_{ver}，但它不返回一个赢家，而是把概率质量推向高于平均的样本、推离低于平均的样本——留下的是教训而不是样本。两种做法消耗的输入完全相同。一种用样本来付它的 KL，每次查询都付，永远地付；另一种把一份可比的 KL 一次性付在权重里。这就是推理强化学习的全部动机，而第11章就是它的实施。

表 10.5: 本章各承重主张的证据层级，含义同第1章。此表的目的是明确指出哪些陈述读者可以依赖，哪些则只差一项细致的研究就会被修正。「已复现」指不同团队用不同模型得到了同样的定性结论。

主张	层级	关键来源
中间词元增加了固定深度模型能计算的东西	定理	(Merrill and Sabharwal, 2024; Li et al., 2024d; Feng et al., 2023a)
思维链并不可靠地忠实记述了那次计算	已复现	(Turpin et al., 2023; Lanham et al., 2023; Chen et al., 2025f)
覆盖率在 n 的数个数量级上平滑上升	已复现；机制已理解	(Brown et al., 2024; Schaeffer et al., 2025)
多数投票远低于覆盖率就饱和	已复现；可推导	节 10.3.1；(Wang et al., 2023a)
$\mathbb{D}_{\mathrm{KL}}(\pi_n \,\|\, \pi_{\text{base}}) = \log n - (n-1)/n$	定理	定理 10.3；(Gao et al., 2023a; Beirami et al., 2024)
best-of-n 在奖励–KL 前沿上接近最优	定理（渐近）	(Yang et al., 2024a)
针对学习型评分器的 best-of-n 关于 n 非单调	已复现	(Gao et al., 2023a; Stroebl et al., 2024)
在计算匹配下，验证器加搜索胜过更大的生成器	程度上仍有争议；呈交叉行为	(Cobbe et al., 2021; Snell et al., 2025; Liu et al., 2025e)
在预算匹配下，树搜索相对 best-of-n 收益甚微	已复现的否定性结果	(Snell et al., 2025; Wu et al., 2025e; DeepSeek-AI, 2025)
内在自我纠正降低推理准确率	已复现的否定性结果	(Huang et al., 2024b; Stechly et al., 2023; Tyen et al., 2024)

当同一个验证器从选择器变成目标函数时，有三件事会改变，而每一件都成为一章。验证器不再需要对接近平局的情形排序，只需平均意义上正确即可，因为梯度是在一个组上求平均的——这正是为什么一个粗糙的二值答案核查器就足够，而奖励模型在同样位置却需要校准，也是为什么 第11章完全不再需要奖励建模。策略现在会移动，因此每一个规范上的缺陷都会变得可以被优化触及；节 10.4.3中的非单调性是静态分布上的一种极值效应，它在训练期的对应物是奖励作弊（节 5.8），而且更糟，因为策略是在主动搜寻那个缺陷，而不是偶然撞进它的尾部。此外，best-of-n 所利用的多样性，正是奖励最大化所消耗的资源：被 RL 集中过的策略有一条更平的推理缩放曲

线，而这笔交易——用 pass@k 换来的 pass@1——是可以测量的，有时划算，有时不划算（第14章）。

要带走的两个事实，正是本章挣来的那两个。推理计算沿一条曲线转化为准确率，这条曲线的形状由选择规则决定，其天花板由验证器决定。以及，验证器比生成器是更好的投资，因为判断比做更容易。第三部分讲的就是：当你把第二个事实当真到足以把验证器放进训练循环里，会发生什么。

本章小结

- 思维链是一种串行计算的机制，而不是一份解释。固定深度 transformer 的一次前向传播落在一个常数深度的电路类里；输出 m 个中间词元买到的是 m 轮那样的计算，其间还有一块草稿纸，而表达能力方面的结果把这一点精确化了。至于轨迹忠实报告了那次计算这个另外的主张，则常常为假。
- 推理预算可以花在五个方向上——更长的链、带投票的并行样本、带评分器的并行样本、顺序修订、树搜索——而在词元数相等时它们并不互为替代，因为它们分别买到的是串行深度、覆盖、选择与纠正。
- 那条平滑的对数线性覆盖率曲线是基准难度分布的性质，而不是模型的性质：对单个问题而言 pass@k 是一条饱和的指数曲线，幂律来自在难题的幂律尾上求平均。
- 多数投票是一个众数估计量。它所需的样本量是 $O((p_1 - p_2)^{-2})$，因而很小；而它的天花板是正确答案成为模型相对多数答案的概率——这是该策略的一个固定性质，且远远低于覆盖率。那个差距恰恰就是验证器所买到的东西。
- best-of-n 分布与基座策略相距 $\log n - \frac{n-1}{n}$ 奈特，与奖励和策略都无关，因为 best-of-n 只用到秩。这就是把 n 与一个 KL 惩罚放到同一条轴上、并使奖励对 KL 图在方法之间可比的原因。在二值验证器下，平局把可达到的散度封顶在 $\log(1/p)$。
- best-of-n 在渐近意义上基本处于奖励–KL 前沿之上，因此反对它的理由不是统计上的而是经济上的：它在每次查询上都用样本来付 KL，而达到 d 奈特需要 $n \approx e^{d+1}$。
- 一个与真实质量相关系数为 ρ 的不完美排序器，把 n 个样本变成实际有效的 n^{ρ^2} 个；而在假阳性率为 η 的二值验证器下，无论 n 多大，准确率都被封顶在验证器的精确率处。验证器质量比预算更起决定作用。

- 在固定计算下，较小的生成器加验证器在很宽的中间难度区间上胜过更大的生成器，而在最难的问题上落败。其机制是生成–验证差距，它是本书的主线，也是 RLVR 有效的原因。
- 在大多数已发表的比较中，预算匹配时树搜索相对 best-of-n 只有很小的增益或没有增益。原因在于搜索用最不可靠的值估计在部分解上做了许多不可逆的剪枝决策，而 best-of-n 只在完整的解上做一次决策。
- 内在自我纠正降低推理准确率，因为检测自身的错误又是那个验证问题，而所需的选择性按 $p/(1-p)$ 增长。带可靠外部信号的修订是有效的；不带外部信号的修订则必须被训练出来。

延伸阅读

从定义这条轴的三篇论文开始。Wei et al. (2022b) 篇幅很短，值得读的是它没有主张什么；Wang et al. (2023a) 应当与节 10.3.1对照着读，因为那段推导预言了它的饱和曲线；而Cobbe et al. (2021) 是第一篇用参数换验证器的论文，也是第三部分一切内容的祖先。关于表达能力，Merrill and Sabharwal (2024) 的陈述最为干净，Li et al. (2024d) 最具构造性；读其中一篇即可，除非你在意那些复杂度类。

关于本章的解析核心，先读Gao et al. (2023a) 了解奖励–KL 平面，再读 Yang et al. (2024a) 了解 best-of-n 为什么位于其前沿上。当你的评分器存在平局时——实践中它确实存在——应当查阅Beirami et al. (2024)。Sessa et al. (2024) 接着说明拿到这个结果之后该做什么，而Balashankar et al. (2024) 论证：如果你打算用 best-of-n 来解码，那么整个 KL 正则化目标都是错的靶子——这个主张我认为很有说服力，且尚未被充分探索。

关于分配，Snell et al. (2025) 是参考文献，应当读它的交叉结构而不是它的标题结论：真正有意思的内容是最优策略取决于难度，而这正是自适应分配的动机所在。可以把它与 Wu et al. (2025e) 配着读；就怀疑论的视角而言，则读Stroebl et al. (2024) 与Schaeffer et al. (2025)，两者合起来消解了相当一部分关于「推理缩放」的松散说法。Setlur et al. (2025b) 是关于验证器为何是承重部件的最尖锐的理论陈述。

关于自我纠正，请把Huang et al. (2024b) 与Tyen et al. (2024) 放在一起读；前者确立了那个否定性结果，后者解释了它，而这一对文献很好地示范了一个领域是如何把一次失望转化为一种机制的。Kumar et al. (2025) 是建设性的回应。

刻意推后的内容：把过程奖励模型作为训练信号，以及蒙特卡洛前缀值，都在第13章；集中化在覆盖率上的代价在第14章；长思维链策略的效率与推理期的系统问题在第23章；而把思维链的可监控性作为一种安全机制则在 第25章。

习题

10.1（**KL 代价，两个方向，以及有平局的情形。**）(a) 在不查阅 节 10.4.1的情况下证明定理 10.3，并明确指出在哪里用到了分数分布的连续性。(b) 计算 $\mathbb{D}_{\mathrm{KL}}(\pi_{\text{base}} \| \pi_n)$，验证它等于 $n - 1 - \log n$，观察到这正是 (2.13)中的 k_3 估计量函数在 $u = n$ 处的取值，并用两句话解释为什么正向散度线性增长而反向散度对数增长。(c) 对一个基础成功概率为 p、可靠且完备、并采用均匀打破平局的二值验证器推导(10.7)，确认极限 $\log(1/p)$，并对 $p \in \{0.02, 0.1, 0.5\}$ 把两种记账方式对 n 作图，对每个 p 找出两者相差超过两倍的预算起点。说明这对一篇把拒绝采样基线与一次 RLVR 训练放在同一条 KL 轴上的论文意味着什么。

10.2（**投票无法靠更多样本来挽救。**）设模型在某个问题上的答案分布为 $q(a^\star) = 0.3$、$q(a_1) = 0.35$，其余的 0.35 分散在许多互不相同的错误答案上。对 $k \in \{1, 5, 25, 200\}$ 计算 maj@k，并对同样的 k 计算 pass@k。然后找出使投票在极限下成功所需的最小温度改变量，其中温度改变建模为把 q 取 $1/\mathcal{T}$ 次幂并在一个三原子近似上重新归一化，并评论这一改变对每样本准确率的影响。

10.3（**验证器质量对预算。**）(a) 推导(10.8)以及有效预算关系 $n' = n^{\rho^2}$。(b) 某团队可以选择把 n 翻倍，或把 ρ 从 0.6 提到 0.65。作为起始 n 的函数，判定哪一种更划算。(c) 现在对假阳性率为 η 的二值验证器推导(10.10)，并在策略成功概率 $p = 0.2$、$\eta = 0.05$ 时，求出 best-of-n 的实际准确率达到峰值时的 n。评论这两个模型——高斯误差与假阳性——哪一个更好地描述了学习型奖励模型，哪一个更好地描述了一套单元测试。

10.4（**搜索为什么会输。**）把步骤级束搜索建模为束宽 b、分支数 m、共 M 层，并假设值函数在每一层上都独立地以概率 $1 - \epsilon$ 把正确续写排进前 b 名。计算一个正确解存活到叶子的概率，并找出使该概率低于 best-of-n 在匹配预算 $n = bmM$ 下选中正确解的概率的那个 ϵ，其中假设 best-of-n 的那一次决策具有同样的单次决策误差 ϵ。然后放松独立性假设——误差是相关的，因为一个坏前缀在每一层上都坏——并说明相关性对搜索是有利还是有害。

10.5（**修订何时值得来一轮。**）推导(10.11)。然后在 f_{r}、f_{w} 与 κ 跨轮次保持不变的假设下把它推广到 R 轮，并求出准确率的不动点。证明对任何 $f_{\mathrm{r}} > 0$，该不动点都低于 1，并把它解释为一个关于「不可靠的批评者能做什么、不能做什么」的陈述。最后，在有一个可靠验证器为修订把关（i.e., $f_{\mathrm{r}} = 0$）的情形下重做这一计算，并解释其中的定性变化。

10.6（**摊销的算术。**）一个 30B 模型服务 10^9 次查询，平均回复长度为 2000 个词元。方案 A：部署配 7B 奖励模型的 best-of-16。方案 B：跑一次耗费 3×10^4 GPU 小时的 RLVR 任务，然后部署单样本解码。使用节 1.6的 FLOP 记账方式，计算两者的总计算量，以及它们打平时的查询数。然后在方案 B 只取回方案 A 准确率增益的 70% 这一假设下重做一遍，并说明每个准确率点的什么价格会使方案 A 成为理性选择。

10.7 ⋆ （**在有限 n 下 best-of-n 还在前沿上吗?**） Yang et al. (2024a) 确立了渐近最优性。请研究有限 n 的问题。利用 (10.4)把 π_n 写成 π_{base} 的一次倾斜，并在 KL 匹配的条件下把它与 (6.15)中的吉布斯最优解 $\pi^\star \propto \pi_{\mathrm{base}} e^{r/\beta}$ 相比较。构造一个显式的奖励分布，使得在同等散度下存在某个倾斜策略在期望奖励上严格优于 π_n，并刻画出不存在这种策略的那一族奖励分布。然后把你的答案与第1章的习题 5 以及节 10.4.1联系起来：best-of-n 能否表达为主模板(1.1)的一个实例? 如果能，对应的奖励是什么?

10.8 † （**把预算匹配起来。**）取一个 1.5–8B 的推理模型、一个带经过审核的核查器的数学基准，以及每题固定的 $B \in \{2^{10}, 2^{12}, 2^{14}, 2^{16}\}$ 生成词元预算。实现表 10.1的全部五种模式，并把准确率对 B 画在同一张图上，标准误按节 2.8的方式计算。以 代码 10.1与代码 10.2为起点。请报告：各模式之间的交叉预算；通过刻意关闭答案归一化来测得的归一化效果；当评分器是学习型奖励模型时你的 best-of-n 曲线是否非单调，而当它是精确核查器时是否单调；以及通过拟合有效指数、由 (10.8)估计出的你的奖励模型的经验 ρ。请保留这张图；它就是阅读 第11章的结果时所应对照的基线。

10.9 † （**复现那次掉头。**）在一个偏好或正确性数据集的一份切分上训练一个小的奖励模型，并把一个强得多的模型（或精确核查器）当作金标准信号。把 n 跑到 1024 做 best-of-n，并把代理分数与金标准分数对 $\log n - \frac{n-1}{n}$ 而不是对 n 作图。拟合(5.11)并报告在你的设置下拟合出的函数形式是否成立。然后刻意削弱这个奖励模型——先减少训练对数，再换更小的主干——并报告峰值位置如何移动，把你的结果与 (5.13)的预测相比较。

算法 10.2 带过程验证器与前瞻的步骤级束搜索。词元预算是 $b\,m\,M\,(1+d)$ 次步骤生成，这也是与算法 10.1比较时应当固定的量。把 d 设为剩余长度并令 $b = bm$，即退化为在叶子上的 best-of-n；每一种中间设置都在值估计偏差与生成成本之间做权衡。

输入： 策略 π_θ，值模型 v，提示 x，束宽 b，分支数 m，最大步数 M，前瞻深度 d

$\mathcal{B} \leftarrow \{x\}$ ▷ 存活前缀
for 层 $\ell = 1$ **to** M **do**
  $\mathcal{E} \leftarrow \emptyset$
  for 每个前缀 $z \in \mathcal{B}$ **do**
    采样 m 个单步续写 $z \oplus \delta_1, \dots, z \oplus \delta_m \sim \pi_\theta$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \left\{ z \oplus \delta_j \right\}_{j=1}^{m}$
  end for
  评估
  for 每个候选 $z' \in \mathcal{E}$ **do**
    if $d > 0$ **and** z' 尚未完成 **then**
      从 z' 再推演 d 步得到 $\tilde{z}$ ▷ 每个候选耗费 d 次步骤生成
    else
      $\tilde{z} \leftarrow z'$
    end if
    $\hat{v}(z') \leftarrow v(x, \tilde{z})$
  end for
  $\mathcal{B} \leftarrow \mathcal{E}$ 中 $\hat{v}$ 最大的 b 个元素 ▷ 仅凭 $\hat{v}$ 的证据就丢弃了 $b(m-1)$ 个候选
  if $\mathcal{B}$ 中每个 z 都已完成 **then**
    break
  end if
end for
return $\arg\max_{z \in \mathcal{B}} \hat{v}(z)$，并记录最终胜出者被排在前 $b/2$ 之外的层数占比

第 11 章

可验证奖励的强化学习

核心思想：用一段程序取代奖励模型

如果一个任务允许一个廉价而可靠的核查器，我们就能在完全没有人参与的情况下，生成源源不断、且不会被标注噪声作弊的奖励。在一个没有做过任何有监督热身的基座模型上，用组相对策略梯度这样做，会产生带回溯与自我验证的长思维链，而这些从来没有人写过示范。那个结果——已发表、可复现、便宜到一个大学实验室都跑得起——正是这个领域在 2026 年呈现出如今这副样貌的原因。

第10章确立了准确率随测试时计算而提升，并且要在推理时利用这一点，要么需要许多样本加一个选择器，要么需要一条长而深思的轨迹。本章讲的是直接训练出第二种选项。我们先定义这一设定，考察验证器究竟是什么以及它们如何失效，把 GRPO 推导为 节 6.6模板的一个实例，然后仔细阅读 2025 年的两项结果——DeepSeek-R1 与 Kimi k1.5——包括它们标题性主张中后续工作予以削弱的那些部分。第12章随后会把这个算法拆开来看。

11.1 这一设定

定义 11.1 (验证器)**.** 一族任务的验证器是一段程序 $r_{\text{ver}} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$（或映到 $[0,1]$），它利用提示、回复，以及可能存在的、对策略保密的特权信息（一个参考答案、一套测试、一个证明核查器、一个执行沙箱），来判定一条回复是否满足该任务的成功标准。

有两个性质要紧，而且它们相互独立。

定义 11.2 (可靠性与完备性)**.** 一个验证器是可靠的，如果 $r_{\text{ver}}(x, y) = 1$ 蕴含该回复确实满足标准（没有假阳性）；是完备的，如果一条确实满足标准的回复总能得到 1（没有假阴性）。

一个证明核查器既可靠又完备。一套单元测试对它所测试的行为是完备的，但作为正确性的规范是不可靠的，因为一个补丁可以通过测试却并不正确。一个字符串匹配式的答案核查器既不可靠（答案可以蒙对）也不完备（`1/2` 与 `0.5`）。你遇到的是哪一种失效决定了你会得到哪一种病理：不可靠招来奖励作弊，不完备则注入标签噪声，系统性地惩罚「正确但不寻常」的行为。

核心思想: RLVR 用规范误差换掉了标注噪声

一个学出来的奖励模型处处带噪，而且它出错的方式与标注者的偏见相关。一个验证器在它所实现的那条标准上完全正确，而对其他任何事情都不置一词。RLVR 并没有消除古德哈特定律（节 5.8）；它只是把它从「奖励模型给这条回复打错了分」搬到了「我实现的那条标准不是我想要的东西」。第二种失效更罕见、更锐利，也容易调试得多——这正是它全部的实践优势所在。

目标函数以显而易见的方式特化了(1.1)：

$$\max_{\theta} \mathop{\mathbb{E}}_{(x,a^\star)\sim\mathcal{D}} \mathop{\mathbb{E}}_{y\sim\pi_\theta(\cdot\,|\,x)} [r_{\text{ver}}(x,y;a^\star)] \;-\; \beta\, \mathop{\mathbb{E}}_{x}[\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})], \tag{11.1}$$

其中 $a^\star$ 是仅供验证器使用的特权参考。请注意相对第7章消失了什么：没有 r_ϕ 需要训练，没有偏好数据集，而且——在 2025–2026 年的大多数配方中——也没有 β（节 12.8）。

11.1.1 验证不对称性在哪里成立

在表 11.1中验证器是一个真正的判定过程的那些行——形式化证明、模式校验、模拟器——正是 RLVR 表现得最干净的地方。验证器只是一个启发式的那些行，才是有意思的失效所在，而节 11.6讲的就是如何控制它们。

常见陷阱: 答案等价性核查是一个假扮成小工具的研究问题

数学 RLVR 中最常见的静默 bug，是一个会拒绝正确答案的答案核查器。$\frac{\sqrt{2}}{2}$ 与 $\frac{1}{\sqrt{2}}$；$\pi/3$ 与 60°；$x=2, y=3$ 与 $(2,3)$；一个正确答案后面又跟了一遍重述。每一次拒绝都是一个假阴性，它实实在在地把策略训练得远离一种正确行为，而且这种噪声并不对称——它与答案类型相关，因此会教会模型回避整类正确的输出格式。在相信任何结果之前，先在几百个手工标注的样本上测一测你的核查器的假阴性率。各团队用朴素核查器时常常测出 5–15% 的比率，这已经足以压过所研究的效应本身。

表 11.1: 可验证的领域及其验证器的性质。「奖励密度」指单个回合能产出什么：一个二值结果、一个比例，还是一个分级的分数。2025 年以来的趋势是沿着这张表往下推进，而每增加一行的难度都是构建核查器的难度，而不是设计算法的难度。

领域	验证器	可靠	完备	密度
算术／短答案数学	答案等价性	否	部分	二值
竞赛数学	符号等价性，`math-verify`	否	部分	二值
形式化定理证明	Lean/Isabelle 内核	是	是	二值
代码（竞赛）	隐藏测试集	否	部分	比例
代码（仓库级）	已有测试集 + 构建	否	否	比例
指令遵循	程序化的约束检查	是	是	比例
结构化输出	模式／解析器	是	是	二值
带金标准答案的检索／问答	精确或模糊匹配	否	部分	二值
工具使用	环境状态断言	部分	部分	二值
游戏／谜题	模拟器	是	是	分级

11.2 组相对策略优化

GRPO 在 DeepSeekMath (Shao et al., 2024) 中被提出，作为对 PPO 的一项出于显存考虑的改动：用一个由同一提示下采样出的一组样本算出的基线，取代学出来的评论家。在 节 6.6的模板中，它对应的设置是：截断的词元级比值、组标准化的优势、按序列的词元均值归一化器，以及损失中的 k_3 KL。

写出来，每条提示采样 G 条回复 $y^{(i)}$：

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathop{\mathbb{E}}_{\substack{x\sim\mathcal{D} \\ \{y^{(i)}\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot\,|\,x)}} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|y^{(i)}|} \sum_{t=1}^{|y^{(i)}|} \Big\{ \min\!\big(\rho_{i,t}A_{i,t},\ \text{clip}\big(\rho_{i,t},\,1-\varepsilon,\,1+\varepsilon\big)\,A_{i,t}\big) - \beta\,\mathbb{d}_{i,t} \Big\} \right], \tag{11.2}$$

其中

$$\rho_{i,t} = \frac{\pi_\theta(y_{i,t}\,|\,x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}\,|\,x, y_{i,<t})}, \qquad A_{i,t} = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^{G})}{\text{std}(\{r_j\}_{j=1}^{G})} \quad \text{for all } t, \tag{11.3}$$

而 $\mathbb{d}_{i,t}$ 是(2.13)给出的 $\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})$ 的逐词元 k_3 估计量。

(11.2)有三个特征值得明确指出，因为每一个都会在第12章中成为一条断层线。

优势沿序列是常数。 对每个 t 都有 $A_{i,t} = A_i$。结果监督的 GRPO 完全不做轨迹内的信用分配。A 上的词元下标是退化残留，之所以保留，是因为同一篇论文里的过程监督变体确实要用到它。

因此词元级比值并不是在做信用分配。 它是在执行信赖域约束。这是两件不同的工作，而(11.2)在词元粒度上做了其中一件，在序列粒度上做了另一件。认识到这种错配，正是 GSPO 背后的论证（节 12.5）。

两个归一化器就明晃晃地藏在那里。 那个 $1/|y^{(i)}|$ 与那个 $1/\text{std}$ 是整场 Dr. GRPO 批评的主题（节 12.2 and 12.3）。任何一条推导都不要求它们存在；两者都是继承来的。

推导: GRPO 就是 RLOO 加上两个额外因子

忽略截断（把 ε 取得很大）与 KL 项。由(6.9)，优势的分子 $r_i - \bar{r}$ 恰好是留一法优势的 $\frac{G-1}{G}$ 倍。于是

$$\mathcal{J}_{\text{GRPO}} \propto \frac{1}{G}\sum_i \frac{1}{|y^{(i)}| \cdot \text{std}(\mathbf{r})} A_i^{\text{RLOO}} \sum_t \nabla_\theta \log \pi_\theta(y_{i,t}\,|\,\cdot) \times \frac{G-1}{G}, \tag{11.4}$$

i.e., GRPO 就是被 $1/(|y^{(i)}| \cdot \text{std})$ 重新加权过的 RLOO——一个 逐样本、依赖于数据的学习率。这是看清下述事实的最干净方式：GRPO 相对 Ahmadian et al. (2024) 的新意完全在于这两个因子；这也是为什么 2025 年的文献中有相当一部分内容就是把它们去掉 (Liu et al., 2025g)。

11.3 R1-Zero：在基座模型上使用结果奖励

DeepSeek-AI (2025) 报告，他们把 GRPO 直接应用于 DeepSeek-V3-Base，不做任何监督微调，使用一个由两部分构成的基于规则的奖励：一个来自答案核查器（代码则来自编译器）的准确率项，以及一个要求推理必须被包在 `<think>/</think>` 标签内的格式项。没有使用奖励模型——这是为避免奖励作弊而明确作出的选择——也没有涉及过程监督、值网络或搜索。

报告出来的现象，按其重要性排列：

1. **准确率大幅且持续地提升**。在 AIME 2024 上，pass@1 从一个很低的起点上升到与同期闭源推理模型相当的水平，历经数千个 RL 步，且没有过早饱和。
2. **回复长度在训练中不断增长**，从几百个词元增长到数千个词元，而奖励中并无任何长度项。
3. **轨迹中出现了自我验证、回溯与重新推导**——诸如「等等，让我再核对一下」这样的说法——作者称之为「顿悟时刻」。
4. **可读性变差**：语言混杂、输出无结构，这正是催生完整 R1 流水线的原因。

这次 RL 训练据报告约耗费 10^5 H800 小时，量级上是基座模型预训练计算的百分之几——这个数字值得记住，因为它确立了推理 RL 是在预训练预算之上的一项次要追加，而不是一项与之相当的开销。

11.3.1 那个涌现出来的长度增长究竟是什么

上面第 2 条被广泛解读为策略已经发现「想得更久有帮助」的证据。其中有一部分确实如此。但Liu et al. (2025g) 指出了一个优化上的假象，它自己就会产生长度增长，无需任何关于能力的故事，而这个论证重要到值得完整给出。

推导: $1/|y^{(i)}|$ 这个因子是一个依赖长度的步长

在(11.2)中，回复 i 的贡献被 $1/|y^{(i)}|$ 所缩放。考虑两条优势幅度相同但长度不同的回复。若 $A_i > 0$（正确），则更短的那条回复的逐词元梯度更大，因此在正确答案之中，简短会被更强地强化。若 $A_i < 0$（错误），则更长的那条回复的逐词元惩罚更小，因为同样的负优势被一个更大的数除掉了。于是长的错误答案在每个词元上受到的惩罚比短的错误答案更少。净效果是：这个目标函数偏好「对的时候短、错的时候长」。由于训练早期大多数回复都是错的，主导性的压力是朝着更长

的错误回复去的。Liu et al. (2025g) 报告，去掉这个因子会把长度曲线压平，同时保持准确率不变，并且尤其是错误回复的长度会塌下去——这恰恰是该机制所预测的特征。

核心思想: 两种机制，都是真的

RLVR 下的长度增长既有能力成分（更长的轨迹确实能解出更难的问题，而奖励会选择它们），也有假象成分（节 11.3.1）。那些报告「涌现出」长度增长却没有控制归一化器的论文，无法把两者分开。对 R1-Zero 的正确读法是：它的定性发现——仅凭结果奖励就能诱发长时间的深思——成立，而具体的长度曲线在一定程度上是估计量带来的假象。这很好地说明了为什么第12章值得仔细读：记账方式是有行为后果的。

11.3.2 以及那个「顿悟时刻」究竟是什么

Liu et al. (2025g) 还考察了基座模型，发现 DeepSeek-V3-Base 在任何 RL 之前就已经会产生自我反思式的语言，而 Qwen2.5 的基座模型在面对光秃秃的问题时表现得像是经过指令微调的——这说明预训练中出现过问答格式的数据。祛魅式的读法是：RL 提高的是一种本已存在于基座分布中的行为的频率与有效性，而不是创造了它。这与 定理 6.3一致：带 KL 正则的 RL 无法把质量放到参考策略的支撑集之外，因此这一设定下的「涌现」应当被理解为放大。放大是否就是全部，则是 第14章的主题。

历史注记: 为什么 R1 比 o1 更要紧

OpenAI 的 o1 (OpenAI, 2024) 在四个月前就更令人印象深刻地展示了这一现象。它没有披露方法，因此没有催生任何文献。R1 披露了一套小团队也能跑起来的方法，而在它发布后的八周之内，arXiv 列表上就出现了数十项复现、消融与变体——基本上就是整个 第12章。这个教训关乎这个领域的社会学而非算法：发布一份可用的配方，是比发布一个结果大得多的干预。

11.4 完整的 R1 流水线

R1-Zero 是一次演示；R1 本身才是可部署的产物，而它的结构被复制得足够广泛，值得摊开来讲。

1. **冷启动 SFT**。一小批长思维链示范，部分由 R1-Zero 自己较好的输出整理而来，用以修正可读性并确立推理格式。这消除了语言混杂问题，并且如第3章所讨论的，给后续的 RL 阶段一个已经会采样出长轨迹的策略。
2. **面向推理的 RL**。在数学、代码、科学与逻辑上针对基于规则的奖励做 GRPO，外加一个语言一致性项。
3. **拒绝采样与 SFT**。从 RL 检查点采样，按正确性并用一个评判模型加以过滤，构建一个既覆盖推理也覆盖通用能力（写作、事实问答、自我认知）的大型有监督集合，然后再在其上重新微调基座模型。这就是图 1.2中那个外层循环的显式化。
4. **面向全场景的最终 RL**。第二个 RL 阶段，把面向推理的基于规则的奖励与面向有用性和无害性的偏好奖励结合起来——i.e., 把第二部分的对齐栈放在推理栈之后，理由见节 1.4的实践框。
5. **蒸馏**。在大模型的推理轨迹上微调更小的稠密模型。这些蒸馏出来的模型显著优于同等规模、从零开始用 RLVR 训练的模型——这一观察在 节 14.8中会变得至关重要。

工程实践: 零起步还是冷启动的抉择

零 RL（跳过第 1 步）在科学上更干净，在经验上更慢。当问题是「RL 究竟贡献了什么」时、当没有强推理教师可供蒸馏时，或当研究一个新的基座模型族时，就用它。当目标是得到一个好模型时，就用冷启动：它收敛更快，避开可读性方面的失效模式，代价不过几千个样本。请注意，在大多数已发表的消融中这一比较是混杂的，因为冷启动数据通常是从一个更强的模型蒸馏来的，因而从流水线之外引入了能力。

11.5 一次做出不同选择的独立复现

Kimi Team (2025a) 在同一周内用一个不同的算法报告了可比的结果，这使得这一对工作格外有信息量：结论（大规模的结果奖励 RL 能诱发长思维链推理）对优化器是稳健的，而细节则不是。

他们报告的、与 GRPO 配方的差异：

- 一种策略优化的**在线镜像下降**形式化，采用朝向前一策略的平方误差正则，而不是截断比值的代理目标——这是对节 6.3中同一个信赖域思想的另一种、更有原则的实现。

- **不使用值网络**，理由与节 6.2.4相符：他们明确论证评论家会惩罚对备选推理路径的探索，因为它会给那些不寻常但最终成功的前缀赋予低值。
- 一个显式的**长度惩罚**，用以控制 R1 放任不管的长度增长，外加按难度组织的课程与优先采样。
- 一套**由长到短**的蒸馏流程，把长思维链能力迁移到输出更短的模型中。

这两篇论文合起来框定了设计空间：R1 表明最小配方就管用，k1.5 表明长度与探索是可控的，而两者都不需要过程监督、搜索或奖励模型。

11.6 可验证任务的奖励设计

验证器就是规范。把它做对是这项工作的大部分内容，而文献已经在若干并不显然的经验上取得了共识。

11.6.1 二值通常胜过部分给分

按通过的单元测试比例给奖励，或者给一段部分正确的推导以部分分数，都很有诱惑力。Mistral-AI et al. (2025) 报告他们正是在代码上测试了这一点，发现在 LiveCodeBench 上它以一个可观的幅度不如二值奖励：那个分级信号被噪声主导，因为通过测试的比例对解的质量只是松散单调的，而且很容易靠退化的特例处理抬上去。一般性的原则是：

> **核心思想: 奖励密度不是免费的**
> 只有当密度是有信息量的时候，更稠密的奖励才会降低方差。一个与真实目标相关性很弱的分级奖励，是在用引入偏差来换取方差的降低；而在组相对估计量下，方差原本就已经是可控的。在你能证明你的分级信号相对你想要的东西是单调的之前，请优先使用二值结果奖励。

11.6.2 格式奖励，以及闭标签陷阱

一个小的格式项——推理放在 `<think>`/`</think>` 之内，最终答案放在可解析的位置——几乎是普遍做法，因为它让准确率核查器变得可靠。这项设计比看上去更微妙。Tang et al. (2026) 报告，一个只要求起始标签的格式规范，会让策略在答案之后附加无界的文本而奖励毫无变化，从而产生退化的长度增长；而要求两个标签都闭合并且序列以

<eos> 终止，就能消除这一现象。这是规范误差的一个典型例子：奖励对我们在乎的某样东西无动于衷，而策略找到了这份无动于衷。

11.6.3 长度整形

放任不管的话，RLVR 下的回复长度会一路上漂，直到撞上生成上限；此时被截断的回复会拿到核查器分配给一个不完整答案的任何奖励——通常是零——从而注入一种与长度而非与质量相关的噪声。有三类修法，复杂度递增：

- **把被截断的样本从损失中掩掉**。最简单的一种，而按 DAPO 的消融，也是他们整套方案中单项收益最大的改动 (Yu et al., 2025c)。一条被截断的回复是缺失的证据，而不是失败的证据。
- **软性的超长整形**。在硬上限之下的一段区间内逐步加码的惩罚，好让策略学会收尾，而不是被不连续地惩罚（节 12.7）。
- **打断**。与其惩罚长度，不如强行终止推理块，并要求模型基于已经写下的内容给出答案 (Khatri et al., 2025)。这把一次截断转化为一个合法的、尽管不利的回合，并让奖励信号保持干净。

11.6.4 把奖励拼装起来

一个有代表性的复合奖励，采用大多数配方所用的形式：

$$r(x,y) = \underbrace{\mathbb{1}[\text{answer correct}]}_{\text{accuracy, } \pm 1} + \underbrace{\lambda_{\text{fmt}}\,\mathbb{1}[\text{format valid}]}_{\text{small, } \sim 0.1} + \underbrace{\lambda_{\text{len}}\,\Phi(|y|)}_{\text{shaping}} - \underbrace{\lambda_{\text{lang}}\,\mathbb{1}[\text{language mixing}]}_{\text{optional}}, \tag{11.5}$$

其中 Φ 取自节 12.7。两条经验法则：准确率项必须以一个数量级压过其余所有项，否则策略会去优化那些便宜的项；以及每一个辅助项都应当被消融，因为辅助项会在代码库中不断堆积，且很少被移除。

常见陷阱: 奖励你想象中的过程，而不是你想要的结果

为「展示过程」「使用工具」「保持简洁」或「引用来源」而设的辅助奖励，可靠地会产出这样的策略：展示过程却不使用它、调用工具却无视结果、简洁而错误、引用不相干的来源。这几种情况都已被报告过。如果一种行为确实有帮助，结果奖励自会选择它；如果结果奖励没有选择它，你应当怀疑的是你「它有帮助」的

信念，而不是去加一项。例外是格式，它不是质量的代理量，而是测量质量的前提条件。

11.7 一份最小可用配方

工程实践: 从一个具备推理能力的检查点出发做 RLVR，2026 年年中的默认设置

下面是多个独立团队所收敛到的一组选择的综合。每一项都会在第12章中被详细考察；把它们汇集于此的意义在于，这套配置第一次尝试就能稳定训练，而(11.2)字面意义上的 GRPO 常常做不到。

- **数据**。10^4–10^5 条带可靠参考答案的提示，并加以过滤，使初始策略的通过率严格介于 0 与 1 之间；先审核核查器的假阴性率。
- **采样**。$G = 8$–16，温度 1.0，最大长度给足（16k–32k），并把被截断的样本从损失中掩掉。
- **优势**。减去组均值，不除以每组的标准差；如果一定要归一化，就按批次级的标准差归一化（节 12.3）。
- **归一化器**。词元级（除以批次中的总词元数），不是逐序列的均值（节 12.2）。
- **权重**。非对称截断，$\varepsilon_{\text{low}} = 0.2$、$\varepsilon_{\text{high}} \approx 0.28$；若在大规模或严重离策略的条件下训练，则改用截断的分离式重要性权重（CISPO）（节 12.5.2）。
- **过滤**。丢掉奖励方差为零的组，并把批次补满（节 12.4）。
- **KL**。在中等规模下、验证器可靠时取 $\beta = 0$；如果你在训练一个非常大的模型、在严重离策略地运行，或观察到训练与推理的对数概率差距在扩大，就保留一个小的 KL（节 12.8）。
- **数值**。语言模型头处至少要用 FP32 logits。在动算法之前先把这一点修好（第22章）。
- **优化器**。AdamW，学习率 1–2×10^{-6}，不做热身重启，全局梯度范数截断到 1.0，一个内层轮次（$E = 1$）。
- **日志**。奖励；策略熵；长度的均值与 95 分位数；零方差组的占比；截断比例；实测的 $\mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$；训练与采样之间的对数概率差距；留出集上的 pass@1 与 pass@k。其中任何一项走向病态，都会先于奖励曲线走平，通常早一百步左右。

代码 11.1: 一步验证与优势计算，展示三个要紧的决定：零方差过滤、不做标准差除法，以及截断掩码。更新的其余部分见代码 6.1。

```python
def build_advantages(prompts, groups, verify, max_len, use_std=False):
    """groups[p] is a list of G sampled responses for prompt p."""
    adv, keep = {}, []
    for p in prompts:
        resp = groups[p]
        r = torch.tensor([verify(p, y) for y in resp], dtype=torch.float32)

        # 被截断的样本是缺失的证据，而不是失败的证据。
        truncated = torch.tensor([len(y.tokens) >= max_len for y in resp])
        if truncated.all():
            continue
        r_valid = r[~truncated]

        if r_valid.std(unbiased=False) == 0:      # 没有组相对信号
            continue                              # -> 改为重采样这个位置

        a = r - r_valid.mean()                    # Dr. GRPO：只减均值
        if use_std:
            a = a / (r_valid.std(unbiased=False) + 1e-6)
        a[truncated] = 0.0                        # 掩掉，不要惩罚

        adv[p], _ = a, keep.append(p)
    return adv, keep
```

11.8 奖励并非你所以为的那样的信号

2025 年的三项结果约束了任何 RLVR 方法所能主张的东西，而之所以在这里引入它们，是因为它们会改变上面那份配方应当被如何阅读。它们会在第14章中被正式分析。

一个样本几乎就够了。 Wang et al. (2025e) 报告，用单个训练样本做 RLVR，把一个 Qwen2.5-Math-1.5B 模型的 MATH500 准确率从 36.0% 提到 73.6%，基本上与用一个包含该样本的 1.2k 样本集合所得到的结果相当，而且这一增益在控制了格式修正之后依然存在。如果一个样本就够了，那么奖励并不是在以有监督数据集的方式传递任务信息；它是在模型已有的行为之中做选择。

随机奖励也能奏效。 Shao et al. (2025) 报告，对 Qwen2.5-Math-7B 而言，用随机分配的奖励做 GRPO 把 MATH-500 提高了 21.4 个点，而用真值奖励是 29.1 个点——并且

这一效应只在某些模型族上出现，在 Llama3 与 OLMo2 上失效。他们提出的机制是：截断的不对称性会放大预训练中先验很高的行为，而与奖励的内容无关。

因此：单一模型族上的消融不构成证据。 这两项结果合起来意味着，在一个 Qwen 基座模型上报告的增益，可能反映的是优化器与该模型预训练之间的相互作用，而不是所研究的那项干预。在至少两个互不相关的模型族上做验证，如今已是最低标准，而第12章中所综述的大多数文献并未达到这一标准。

常见陷阱：「奖励上去了，那又如何」的问题

由于零方差的组会被丢弃，也由于通过率的分布在训练过程中不断移动，一次 RLVR 训练的奖励曲线是在一个不断变化的提示子集上算出来的。因此它在不同步之间不可比，而且完全可能一边上升、一边留出集准确率下降。请始终维护一个固定的留出评估集与固定的采样预算，并把训练奖励曲线当作优化器健康状况的诊断量，而不是能力的诊断量。

本章小结

- RLVR 用一段程序取代了学出来的奖励模型。这消除了标注噪声与无上限的标注成本，并把古德哈特定律从「打错分」搬到了「规范写错」——一种更罕见、也容易调试得多的失效。
- 验证器的可靠性与完备性相互独立，并诊断出不同的病理：不可靠招来作弊，不完备注入噪声并惩罚「正确但不寻常」的行为。答案等价性核查器通常有 5–15% 的不完备率，这已经大到足以压过所报告的效应。
- GRPO 是一种带组基线的无评论家策略梯度；在代数上，它就是被 $1/(|y^{(i)}| \cdot \mathrm{std})$ 重新加权过的 RLOO。它的优势沿序列是常数，因此它的词元级重要性比值执行的是信赖域约束，而不是信用分配。
- R1-Zero 确立了：仅凭结果奖励作用于一个基座模型，就能诱发带回溯与自我验证的长深思轨迹。这一定性结果是稳健的；而那条备受称道的长度增长曲线在一定程度上是 GRPO 逐序列长度归一化带来的假象，而「顿悟时刻」式的语言也有一部分本就存在于基座模型之中。
- Kimi k1.5 用一个镜像下降优化器、不使用评论家、并做显式的长度控制，得到了可比的结论，这确立了该发现是这一设定的性质，而不是 GRPO 的性质。

- R1 流水线——冷启动、推理 RL、拒绝采样 SFT、最终的混合 RL、蒸馏——是 2025–2026 年大多数配方所遵循的模板，而它最重要的阶段恰恰是被记录得最少的那个：把当前最佳策略的采样轨迹蒸馏回有监督数据。
- 在奖励设计上，二值结果奖励通常胜过分级的部分给分；格式奖励必须同时规定终止与结构；被截断的样本应当被掩掉而不是被惩罚。
- 单样本与随机奖励的结果意味着，RLVR 的信号在很大程度上是在已有行为之中做选择，而且单一模型族上的消融无法支撑算法层面的主张。

延伸阅读

请把DeepSeek-AI (2025) 与Liu et al. (2025g) 接连读完。前者是结果，后者是审计，而这一组合比其中任何一篇单独所能教的，都更能教会你如何阅读这批文献——尤其是这一点：一个标题性的定性主张可以是可靠的，而它被引用得最多的那个定量特征却是估计量带来的假象。Shao et al. (2024) 是 GRPO 本身的出处，值得一读的是它把评论家框定为一个显存问题而非统计问题的视角。

Kimi Team (2025a) 是「一次独立实现沿另一条路径到达同一位置」的最好范例，而它关于值网络为何会压制对备选推理路径的探索的讨论，是该论证在已发表文献中最锐利的陈述。

就作为一门工程学科的奖励设计而言，Mistral-AI et al. (2025) 对哪些做法没有奏效异常坦诚——失败的熵奖励、失败的代码部分给分——而这种具体程度的否定性结果既稀少又宝贵。Tang et al. (2026) 在万亿参数规模上扮演了同样的角色，并得出了若干相反的结论，尤其是关于 KL 惩罚的；把两者放在一起读，是为第12章所做的最好准备。

最后，Wang et al. (2025e) 与Shao et al. (2025) 应当尽早读而不是留到最后。它们是最有可能改变你对第三部分中其他一切结果的解读方式的两篇论文。

习题

11.1（验证器误差的传播是不对称的。）设某验证器在正确答案上的假阴性率为 η，且没有假阳性。对一组 G 个样本、真实成功概率为 p 的情形，计算一条真正正确的回复所被赋予的组均值优势的期望。证明由此得到的梯度被一个含 η 的因子所衰减，

并判定这一效应是否等价于降低学习率，还是在性质上有所不同。然后改为考虑假阳性重做一遍，并评论哪一类误差危害更大。

11.2（**长度偏差，定量地。**）沿着节 11.3.1，把一个批次建模为：正确回复长度为 L_c，错误回复长度为 L_w，成功概率为 p。写出在（a）$1/\left|y^{(i)}\right|$ 归一化器与（b）常数归一化器之下，正确与错误回复的逐词元梯度系数的期望。推导出使（a）产生朝向更长的净压力、而（b）不产生该压力的关于 p, L_c, L_w 的条件。

11.3（**零方差的组最终会占掉你批次的大部分。** ）把逐提示的成功概率建模为 $\mu \sim \mathrm{Beta}(a, b)$。计算一个大小为 G 的组方差为零的概率，作为 G, a, b 的函数。现在假设训练把该分布推向 $\mu = 1$；对 $G \in \{4, 8, 16, 64\}$ 把被浪费的组的比例对平均通过率作图，并据此在给定生成成本预算的条件下论证应当选取某个具体的 G。

11.4（**复合奖励的作弊。**）取(11.5)，令 $\lambda_{\text{fmt}} = 0.5$ 且准确率二值取值于 $\{0, 1\}$。证明存在这样的提示，使得策略严格偏好「格式合法但答案错误」而非「格式非法但答案正确」，并计算这一情况发生的准确率阈值。然后给出防止它发生的关于 λ_{fmt} 的一般条件。

11.5（**设计一个可靠的核查器。**）为一个数学答案核查器写出规范，使其对由精确有理数、根式，以及含 π 与 e 的简单闭式所构成的答案类是可靠的（没有假阳性）。然后指出三种会被你的规范拒绝的正确答案形式，并论证每一种应当在核查器中修正，还是在格式奖励中修正。

11.6（**把 R1 流水线读作一个外层循环。**）R1 流水线在一个 RL 阶段之后运行拒绝采样 SFT，然后再做一次 RL。请用节 1.7的分布偏移分类法解释，两个 RL 阶段各自能达成什么而另一个不能，以及夹在两者之间的 SFT 阶段为什么并不与第一个 RL 阶段重复。

11.7 ⋆（**一个样本为什么会有效?**）构造一个形式化模型，使得在单条提示上做 RLVR 能提升在一个留出提示分布上的表现。你的模型应当明确说明提示之间共享了什么，以致来自其中一条的梯度信息可以迁移；它还应当预测这一效应在何时失效。然后用 Shao et al. (2025) 所报告的模型族依赖性来检验你的预测：你的模型能否解释为什么这一效应在某些预训练模型族上很强，而在另一些上完全没有?

11.8 †（**微缩版复现 R1-Zero。**）使用一个 1.5B 的基座模型（不是指令微调过的），以及一个配有你已审核过的核查器的数学集合，先完全按(11.2)运行 GRPO，再按节 11.7推

荐的三项改动（不做标准差除法、词元级归一化器、截断掩码）运行一次。对每次训练，记录实践框中的全套诊断量，并把回复长度按正确与错误分开作图。请报告你是否复现出（a）准确率提升，（b）长度增长，（c）节 11.3.1所预测的长度曲线差异。然后把第一种配置的奖励换成独立的抛硬币结果，重跑一遍并报告发生了什么。

11.9 † **（审计一个核查器。）**任取一个公开的数学 RLVR 数据集及其配套的答案核查代码。采样 300 条被核查器判为错误的模型回复，手工标注它们，并报告带置信区间的假阴性率，按答案类型分组给出。用你对习题 1 的答案，估计这种噪声对组相对优势的影响。

算法 11.1 用于 RLVR 的 GRPO，一次迭代。可与算法 6.1对照：组基线取代了评论家，验证器取代了奖励模型。标有 † 的行，正是第12章中每一个变体都会改动的那些行。

输入： 策略 π_θ，参考策略 π_{ref}，验证器 r_{ver}，带参考答案的提示集 $\mathcal{D}$，组大小 G，截断参数 ε，系数 β

for 迭代 $k = 1, 2, \ldots$ **do**
  $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$；将权重推送到推理引擎
  采样提示 $\mathcal{B} \subset \mathcal{D}$
  for 每个 $x \in \mathcal{B}$ **do**
    在温度 $\mathcal{T}$ 下采样 $y^{(1)}, \ldots, y^{(G)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$
    对 $i = 1, \ldots, G$ 计算 $r_i \leftarrow r_{\text{ver}}(x, y^{(i)}; a^\star)$
    $\mu \leftarrow \text{mean}(r_{1:G})$；$\sigma \leftarrow \text{std}(r_{1:G})$
    if $\sigma = 0$ **then** ▷ † 每个 $A_i = 0$：这条提示什么也没贡献
      continue
    end if
    对所有 i 计算 $A_i \leftarrow (r_i - \mu)/\sigma$；对所有 t 令 $A_{i,t} \leftarrow A_i$ ▷ † 归一化器
  end for
  在所有采样到的词元上重算 $\log \pi_{\theta_{\text{old}}}$ 与 $\log \pi_{\text{ref}}$ ▷ † 或复用引擎给出的对数概率
  for 内层轮次 $e = 1$ **to** E **do**
    for 词元的小批量 **do**
      $\rho_{i,t} \leftarrow \exp(\log \pi_\theta - \log \pi_{\theta_{\text{old}}})$
      对(11.2)做上升 ▷ † 权重、截断的不对称性、归一化器、KL
    end for
  end for
end for
return π_θ

第 12 章

组相对策略优化的解剖

核心思想: 这些变体不是奇技淫巧，而是对十种已命名病理的系统回应

从 2025 年 3 月到 2026 年年中，文献产出了数十种对 GRPO 的修改。当作一张清单来读，它们令人眼花缭乱；当作一张从病理到修法的地图来读，它们几乎井然有序：每一项都指名一个具体的统计缺陷——一个有偏的归一化器、一个不适定的重要性比值、一份被湮灭的梯度、一个退化的组、一处数值上的失配——并把它修好，通常是用一点点偏差换来大量的方差或稳定性。本章就是这张地图。

第11章按发表时的样子给出了 GRPO, 并标出了三处它做了不必要选择的地方。本章把它们全部检视一遍。节 12.1列举这些病理；节 12.2 到节 12.6逐一处理，并给出应对它们的算法；节 12.11把结果组装成一个设计空间；节 12.12 解释为什么这一文献中的大多数比较都检验效能不足，以及该改用什么做法。

开始之前先给一句警告。这一文献中有很大一部分只在单一的基座模型族上报告结果，通常是 Qwen，而节 11.8给出了两条理由，说明这类结果可能反映的是优化器与某个特定预训练语料的相互作用，而不是那项干预本身。凡是某项发现已在多个模型族上被复现的，我都会说明；凡是没有的，就把效应量当作一个上界来看待。

12.1 十种病理

在进入细节之前，关于表 12.1有三点观察。

第一，P2、P3 与 P4 是记账上的缺陷：它们来自归一化器和过滤规则，而不是来自策略梯度。它们也是报告中效应量最大的几项，这件事本身就在教我们：当一次训练跑偏时，该先往哪里看。

第二，P5 与 P6 是对同一个对象的两种相反诊断。P5 说词元级比值噪声太大、派不上用场，应当换成一个序列级的量；P6 说这个比值本身没问题，坏事的是截断，它

表 12.1: 病理地图。本章中的每一个算法都针对其中的一行或多行。「机制」一列才是值得记住的部分：修法很容易查到，而机制能推广到下一个变体。

	病理	机制	处理于
P1	熵坍缩	少数高协方差词元在数百步之内就把策略锐化	节 12.6
P2	回复长度偏置	$1/\lvert y^{(i)}\rvert$ 这个因子对长的错误答案惩罚不足	节 12.2
P3	难度偏置	除以组标准差会抬高接近饱和的提示的权重	节 12.3
P4	退化组	奖励方差为零，梯度就恰好为零	节 12.4
P5	不适定的词元比值	每个位置只有一个样本，无法校正一个分布	节 12.5
P6	梯度湮灭	被截断的词元毫无贡献，而它们恰恰是有信息量的那些	节 12.5.2
P7	训练–推理失配	权重完全相同，两个引擎给出的 $\log p$ 却不一致	第22章
P8	长度爆炸	啰嗦对奖励是中性的；截断则注入相关噪声	节 12.7
P9	覆盖度丧失	向主导模式锐化会摧毁稀有的正确模式	第14章
P10	MoE 路由波动	专家激活在两次前向之间翻转，使比值失效	节 12.10

把真正要紧的词元毁掉了。GSPO 取第一种看法，CISPO 取第二种，而两者都报告了相对同一基线的改进。节 12.5.3论证它们是相容的。

第三，P7 根本不是一个 RL 问题——它是一个数值问题——而文献中报告的最大的单轴改进，正是来自修好它 (Khatri et al., 2025)。它有自己独立的一章。

12.2 归一化器：长度偏置

回忆(11.2)。这个目标函数对词元求和，然后除以 $G\cdot\lvert y^{(i)}\rvert$：每一条序列贡献相同，而在一条序列内部，每个词元贡献 $1/\lvert y^{(i)}\rvert$。节 11.3.1展示了后果。这里我们把各种备选方案精确地摆出来，因为它们在纸面上差别很小，在行为上差别很大。

把一般的代理目标写成

$$\mathcal{J}=\mathbb{E}_{x}\Big[\frac{1}{\mathcal{N}}\sum_{i=1}^{G}\sum_{t=1}^{\lvert y^{(i)}\rvert}f_{i,t}(\theta)\Big],\qquad f_{i,t}=\min\big(\rho_{i,t}A_{i,t},\ \mathrm{clip}\big(\rho_{i,t},\,1-\varepsilon_{\mathrm{low}},\,1+\varepsilon_{\mathrm{high}}\big)A_{i,t}\big),\tag{12.1}$$

并考虑 $\mathcal{N}$ 的四种取法：

Liu et al. (2025g) 论证，GRPO 的逐序列平均是从预训练继承来的：在预训练中，把数据打包到固定的上下文长度，使得沿词元轴取平均在数值上是合理的；把它带进回复长度可变的 RL 里，它就变成了一个依赖长度的步长。他们审查了若干开源实现，

表 12.2: 损失归一化器。「词元权重」是单个词元的梯度贡献前面的系数；「序列权重」是一整条回复所获得的总权重。只有常数归一化器既给每个词元相同的权重，又让整体尺度不依赖于这一批数据的长度构成。

名称	$\mathcal{N}$	词元权重	后果
逐序列平均（GRPO）	$G\,\lvert y^{(i)}\rvert$	$\propto 1/\lvert y^{(i)}\rvert$	长的错误答案惩罚不足（P2）；长度一路上漂
全局词元平均（DAPO）	$\sum_j \lvert y^{(j)}\rvert$	相同	消除 P2；损失尺度依赖于批内的长度构成
常数（Dr. GRPO）	$G\,L_{\max}$	相同	消除 P2；尺度固定，因此梯度范数随批内长度而变
序列平均	G	$\propto 1$，直接求和	长序列主导更新

发现的都是同一个因子。他们观察到，在同时去掉长度归一化器与标准差除法之后，这个目标函数就还原成了一个带蒙特卡洛回报与无偏基线的普通策略梯度——i.e., GRPO 相对定理 6.1的两处偏离都是无心之失。

工程实践: 该用哪个归一化器

用全局词元平均（DAPO）或者常数（Dr. GRPO）。两者都消除 P2；它们的区别只在于梯度尺度是否依赖于这一批的总长度，而这一点 Adam 基本能吸收掉。避免逐序列平均。除非你有明确的理由想让长回复获得更大权重，否则也避免朴素的序列平均。请报告你用的是哪一种——有很大一部分论文没有报告，而这个选择在困难基准上值好几个百分点 (Yu et al., 2025c; Liu et al., 2025h)。

常见陷阱: 归一化器会与梯度裁剪相互作用

全局词元平均归一化会让损失的尺度依赖于这一批中的词元数量，而这个数量会随着训练中回复长度的增长而变化。如果你同时又把全局梯度范数裁剪在一个固定值上，那么有效学习率就会在训练过程中漂移，而这在你的配置文件里是看不出来的。请记录裁剪前的梯度范数；如果它随回复长度上升，你其实是在悄悄地退火。

12.3 归一化器：难度偏置

第二个继承来的因子，是(11.3)中除以组标准差这一步。对于二值奖励，G 个样本中有 c 次成功时，$\hat{\mu} = c/G$ 且 $\hat{\sigma} = \sqrt{\hat{\mu}(1-\hat{\mu})}$，于是

$$A_i = \frac{r_i - \hat{\mu}}{\sqrt{\hat{\mu}(1-\hat{\mu})}} \quad \Longrightarrow \quad \text{提示权重} \ \propto \ \frac{1}{\sqrt{\hat{\mu}(1-\hat{\mu})}}, \tag{12.2}$$

它在 $\hat{\mu} = 1/2$ 处取得最小值，并在 $\hat{\mu} \to 0$ 或 1 时发散。因此逐组标准化恰恰抬高了那些携带信息最少的提示——几乎已经做出来的和几乎做不出来的——的权重，这与节 12.4.1的可学习性论证正好相反。这就是 P3。

推导: 标准差除法同时也是一个意外的梯度修正

真实的故事比「去掉它」更有意思，而这个故事的收场解释了为什么已发表的消融彼此矛盾。Chu et al. (2025b) 观察到，当退化组被丢弃时，一个名义规模为 B 的批次实际上只贡献了 $B - M$ 份有用的梯度，而标准的归约却仍然除以 B。要得到无偏估计，需要乘上

$$\alpha = \frac{B}{B - M}, \tag{12.3}$$

他们报告说这个量在训练过程中随着越来越多的提示饱和，会从大约 1.5 漂到 4.0。接着他们指出，GRPO 中实测的逐组标准差大致落在 0.10–0.35，所以除以 $\hat{\sigma}$ 相当于把幸存下来的梯度乘上 3–10 倍——在数值上与 α 相仿。因此，去掉标准差而不补上 α，等于同时去掉了一个偏差和一个与之抵消的修正，净效果可以朝任何一个方向走。Chu et al. (2025b) 报告，在 $\alpha = 1$ 且不用标准差的情况下，他们与普通 GRPO 打平（43.9% 对 43.7%），而加上 α 修正之后升到 47.8%——并明确指出这与 Liu et al. (2025g) 的结论不一致。

节 12.3是这一文献中最有教益的一段插曲，值得作为一条一般性的教训写下来。

核心思想: 彼此混杂的归一化器

当一个估计量中的两个因子量级相仿而效应方向相反时，单独消融其中任何一个都会给出无法解释的结果，而不同的研究组会各自真诚地发表相互矛盾的结论。在相信关于某个归一化器的消融之前，先检查那篇论文有没有控制住其他几个。在这个具体的案例里，三个研究组测量同一个混杂因素，得出了三种不同的结论 (Liu et al., 2025g; Chu et al., 2025b; Khatri et al., 2025)，而当下诚实的说法是：只

要退化组的记账处理得当，逐组标准化、批级标准化与不做标准化，表现是相当的。

工程实践: 优势归一化

减去组均值。至于分母，优先选择要么什么都不除，要么除以整批优势上的批级标准差 (Hu, 2025; Khatri et al., 2025)；当提示的混合跨越了差别极大的难度时，Liu et al. (2025h) 发现最稳健的是组级均值配批级标准差这种混合做法。不论你怎么选，只要你丢弃退化组而不重采样，就要施加 $\alpha = B/(B-M)$ 的重新缩放。

12.4 退化组

如果某条提示的 G 个样本全都拿到相同的奖励，那么每一个组相对优势都为零，这条提示不贡献任何梯度（算法 11.1第 9 行）。在二值奖励下这很常见，而且会越来越糟：随着策略变好，全对的组所占的比例单调上升，于是有效批量在训练中不断缩小，而名义批量纹丝不动 (Yu et al., 2025c)。一次训练看上去每步在 512 条提示上训练，实际上却只在 80 条上训练。

有三种应对：

动态采样（DAPO）。 超量采样提示，丢弃准确率恰为 0 或 1 的组，再不断补充，直到这一批全部由优势方差非零的提示组成——形式上，就是把 $0 < \left|\{i : r_{\text{ver}}(x, y^{(i)}) = 1\}\right| < G$ 作为一条约束，施加在哪些提示可以进入批次上 (Yu et al., 2025c)。额外的生成开销比看上去要小，因为在同步系统里，挂钟时间是由这一批中最长的那次采样决定的，而不是由平均值决定的。

梯度重新缩放（GPG）。 把退化组排除在求和之外，但把损失乘上 $\alpha = B/(B-M)$，从而在不增加生成的前提下恢复正确的梯度（节 12.3）。

提示集的甄别。 永久移除已饱和的提示。ScaleRL 的「无正例重采样」维护一份逐提示通过率的历史记录，并把任何通过率超过 0.9 的提示永久退役 (Khatri et al., 2025)。这比动态采样便宜，而且是同一想法在数据侧的版本。

12.4.1 难度、可学习性与课程

底下的那个量是一条提示的可学习性。对于二值奖励，若逐提示的成功概率为 μ，则奖励的方差是 $\mu(1-\mu)$，在 $\mu = 1/2$ 处最大；而一个规模为 G 的组发生退化的概率是

$$\mathbb{P}[\text{退化}] = \mu^G + (1-\mu)^G, \tag{12.4}$$

在 $G = 8$ 时，$\mu \in [0.3, 0.7]$ 下它低于 1%，而 $\mu = 0.92$ 时却超过 50%。所以，一套把 μ 维持在 1/2 附近的课程并不是什么启发式技巧；它是让这个估计量根本能够工作的前提。

实践上的困难在于，μ 是当前策略的性质，因此会变陈旧。在用的做法有：从你已经采样过的组里在线估计 μ（免费，但只对采样过的提示有效）、为每条提示维护一个指数移动平均，或者周期性地对整个提示池重新打分。2026 年的一项结果论证说，有用的课程会从学习动力学中内生地产生，因为策略自身的改进会不断地对提示难度重新排序 (Huang et al., 2026d)，若这一点成立，就会让精心手工设计的课程变得没有必要。请把它当作单篇证据来看待。

工程实践: 组的规模

G 是在生成开销与基线质量、退化率之间做权衡。低于 $G = 4$，基线噪声太大；高于 $G \approx 64$，方差缩减的边际收益很小。Cheng et al. (2026) 直接研究了分配问题——如何把固定预算切分到 G、每批提示数与更新步数之间——并报告说，计算最优的 G 会随预算可预测地上升，然后饱和，其机制随难度而不同：在容易的提示上是锐化，在困难的提示上是覆盖度的扩张。他们把缩放律看作规定性的分配规则而非描述性曲线，这才是使用这类结果的正确方式。

12.5 词元级与序列级重要性比值

现在我们来到真正尚无定论的问题。节 2.1表明，一个序列级比值是 T 个词元比值的乘积，因此其动态范围随长度呈指数增长。节 11.2则观察到，GRPO 的优势活在序列层面，而它的比值活在词元层面。这两个事实拉向相反的方向。

12.5.1 反对词元级比值的理由

Zheng et al. (2025a) 给出了最锐利的论证，而且它是统计上的而非经验上的。重要性采样是靠取平均来校正一个分布的：(2.15)之所以成立，是因为权重是在从行为策略抽

出的 许多样本上被平均掉的。GRPO 在每个词元位置施加一个权重 $\rho_{i,t}$，而在那个位置上恰好只有一个样本。没有东西可以拿来平均，所以这个权重根本没有执行任何校正；它只是注入方差。他们对这个目标函数的用词是「不适定」。噪声沿序列长度累积，随后又被截断放大，而且他们报告说，一旦坍缩发生就无法挽回——回退到更早的检查点、重调截断范围、延长生成长度、更换查询集，全都无效。

他们的修法是让目标函数的粒度与奖励的粒度相匹配：

$$\mathcal{J}_{\text{GSPO}}(\theta) = \mathbb{E}\Big[\frac{1}{G}\sum_{i=1}^{G} \min\big(s_i(\theta)A_i,\ \text{clip}\big(s_i(\theta),\,1-\varepsilon,\,1+\varepsilon\big)\,A_i\big)\Big], \tag{12.5}$$

$$s_i(\theta) = \Big(\frac{\pi_\theta(y^{(i)}\,|\,x)}{\pi_{\theta_{\text{old}}}(y^{(i)}\,|\,x)}\Big)^{1/|y^{(i)}|} = \exp\Big(\frac{1}{|y^{(i)}|}\sum_{t=1}^{|y^{(i)}|}\log\frac{\pi_\theta(y_{i,t}\,|\,x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}\,|\,x, y_{i,<t})}\Big). \tag{12.6}$$

核心思想：那个指数就是整个设计

s_i 是词元比值的几何平均，不是它们的乘积。没有 $1/|y^{(i)}|$ 这个指数，少数几个词元就会让序列比值剧烈摆动，而不同长度的回复将需要不同的截断范围。有了它，s_i 的动态范围是有界的，一个 ε 就能通吃各种长度。代价是 ε 必须比 GRPO 的小上好几个数量级：Zheng et al. (2025a) 报告 $\varepsilon_{\text{low}} = 3\times10^{-4}$、$\varepsilon_{\text{high}} = 4\times10^{-4}$，而与之对照的调好的 GRPO 基线用的是 0.2/0.27。如果你把 GSPO 移植进一个代码库却保留 $\varepsilon = 0.2$，那就永远不会有任何东西被截断，你实现的将是一个长度归一化的 REINFORCE。

推导：GSPO 对梯度做了什么

忽略截断，

$$\nabla_\theta \mathcal{J}_{\text{GSPO}} = \mathbb{E}\Big[\frac{1}{G}\sum_i s_i(\theta)\,A_i \cdot \frac{1}{|y^{(i)}|}\sum_t \nabla_\theta \log \pi_\theta(y_{i,t}\,|\,\cdot)\Big], \tag{12.7}$$

而 GRPO 则是用每个词元自己的比值去加权该词元的得分函数。于是差别归结为一句话：**GSPO 对一条回复中的所有词元一视同仁地加权；GRPO 则按各自的比值给它们加权**。既然一条回复中所有词元的优势都是同一个数，GSPO 的选择才是内部自洽的那一个。

同一批作者报告的两项二阶后果在实践上很重要。混合专家模型受专家激活波动之苦：一次更新之后，同一个词元被激活的专家中约有 10% 会发生变化，这使得词元级比值彻底失效（P10）。此前的绕行办法是在训练时缓存并重放 $\pi_{\theta_{\text{old}}}$ 的路由决策，这

要付出内存、通信与有效容量的代价；GSPO 消除了这一需要，因为序列级似然对个别词元的波动不敏感。与此相关的是，由于序列似然对精度差异的容忍度远高于词元似然，GSPO 使得直接使用推理引擎给出的似然成为可行——这是对 P7 的一个部分性的结构性回答（第22章）。最后，有些反直觉的是，GSPO 截断掉的词元比 GRPO 大约多两个数量级，训练效率却更高，作者把这读作一项证据：GRPO 的词元级梯度大多是噪声。

12.5.2 支持词元级比值的理由，只是做法不同

MiniMax (2025) 接受词元粒度，转而攻击截断（P6）。他们的观察是：在 PPO 式的截断下，一个比值跑出这条带子之外的词元贡献恰好为零的梯度，而在离策略更新下，这些词元中低概率词元占比过高，比如「However,」「Wait,」「Recheck」——正是那些引导推理分支走向的 分叉词元。把它们的梯度置零，恰恰抹掉了长思维链训练所需要的信号。

他们的修法是截断重要性权重而不是截断目标函数，并把它放在一个停止梯度之下，从而让梯度始终经由 $\log \pi_\theta$ 流动：

$$J_{\text{CISPO}}(\theta) = \mathbb{E}\Big[\frac{1}{\sum_j |y^{(j)}|}\sum_{i=1}^{G}\sum_{t=1}^{|y^{(i)}|} \text{sg}[\hat{\rho}_{i,t}(\theta)]\, A_{i,t}\, \log \pi_\theta(y_{i,t}\,|\,x, y_{i,<t})\Big], \tag{12.8}$$

$$\hat{\rho}_{i,t}(\theta) = \text{clip}\Big(\rho_{i,t}(\theta),\, 1-\varepsilon^{\text{IS}}_{\text{low}},\, 1+\varepsilon^{\text{IS}}_{\text{high}}\Big). \tag{12.9}$$

在不做权重截断时，这恰好退化为策略梯度。截断会让梯度带上少许偏差，但保留了每一个词元的贡献，而这在长回复中最为要紧。MiniMax (2025) 关掉了下界（把 $\varepsilon^{\text{IS}}_{\text{low}}$ 设得很大），只调上界，不使用 KL 惩罚，并沿用了 DAPO 的词元级归一化器与动态采样。

核心思想: 截断权重，而不是截断目标函数

这是本章中最可迁移的一个想法，而且它的适用范围远不止 CISPO。PPO 的截断是湮灭；一个被截断的权重则是衰减。当你的问题是离策略样本的权重不可靠时，衰减几乎总是更好的工具，因为湮灭会把该样本的信息整个丢掉，而且它是选择性地丢在那些移动得最厉害的样本上。同一种构造会在第22章中再次出现，作为对数值失配的修正——它就是截断重要性采样 (Ionides, 2008) 被施加在流水线的两个不同位置上。

表 12.3: 粒度谱系。这五行都是模板(6.17)中权重 $\omega_{i,t}$ 的不同取法。请注意 ε 在各行之间不可比：它约束的是动态范围完全不同的量。

方法	比值单位	处理方式	典型 ε
GRPO / DAPO	词元	截断目标函数（湮灭）	0.2 / 0.2–0.28
CISPO	词元	截断权重并断开梯度（衰减）	仅上界，~ 4–8
GMPO	词元，几何平均	截断目标函数	—
GSPO	序列，几何平均	截断目标函数	$3\text{–}4 \times 10^{-4}$
DHPO	混合，逐分支	分别截断各分支，再混合	逐分支

12.5.3 调和这两种诊断

GSPO 说词元比值是噪声；CISPO 说截断词元比值会毁掉信号。两者都报告了相对同一基线的收益。它们是相容的，因为它们说的是不同的项：GSPO 去掉的是逐词元加权注入到梯度方向中的 方差，而 CISPO 去掉的是截断施加在「哪些词元能有贡献」上的选择效应。一种方法完全可以两样都做——而Min et al. (2026) 基本上就是这么做的，他们在一个截断的代理目标里混合词元级与序列级比值，并对各分支分别截断，使得任一分支中的离群值都无法占据主导，混合方式则可以取固定的平均，也可以由熵来引导。他们的表述对一本教科书来说恰到好处：粒度问题是一个有待调节的权衡，而不是一个有唯一答案的问题。Zhao et al. (2025e) 从另一个方向占据了中间位置，他们最大化的是词元级各项的几何平均而非算术平均。

12.6 非对称截断与熵

我们来到 P1，这是终结掉的训练比任何其他失效模式都多的一种。

12.6.1 为什么对称截断会让熵坍缩

Yu et al. (2025c) 给出了机制。取 $\varepsilon = 0.2$ 和一个 $A > 0$ 的词元。若 $\pi_{\theta_{\text{old}}}(y) = 0.01$，截断把它的新概率封顶在 0.012：这个词元每次更新最多只能多拿到 0.002 的概率质量。若 $\pi_{\theta_{\text{old}}}(y) = 0.9$，上限是 1.08，超过了 1，因此根本不起约束作用。于是对称截断是一种累退税：它限制稀有词元的增长，却放任常见词元。而既然稀有词元正是那些用于探索的，对称截断就系统性地压制了探索。作者报告说，被上界截断的词元其平均概率低于 0.2，这印证了上界截断几乎只作用在低概率词元上。

修法——*Clip-Higher*——是把两个界解耦，只把上界抬高，通常取 $\varepsilon_{\text{low}} = 0.2$、$\varepsilon_{\text{high}} \approx 0.28$。同时抬高 ε_{low} 是被刻意避免的：那会允许被压制词元的概率被推向零，从另一侧把采样空间压塌。Park et al. (2025b) 提供了机制上的印证，他们表明低端截断事件会摊开概率质量并抬高熵，而高端截断事件会压缩质量并降低熵，因此正是这种不对称性在维持探索。

12.6.2 熵与奖励的交换

Cui et al. (2025b) 报告了一条把验证奖励 R 与策略熵 $\mathbb{H}[\pi_\theta]$ 联系起来的经验定律，它跨越 11 个基座模型、四个模型族、数学与代码，以及四种 RL 算法：

$$R = -a \exp(\mathbb{H}[\pi_\theta]) + b, \tag{12.10}$$

只有两个拟合系数，而且据他们报告基本与算法无关。把(12.10)读作一条关于训练的陈述，它说的是：性能是用熵买来的，后期性能可以从早期训练中预测出来，而上限在 $\mathbb{H}[\pi_\theta] = 0$ 处达到，给出 $R = b - a$。

他们在机制上的解释才是有用的部分。熵在相邻两步之间的变化，是由一个词元的概率与其 logit 变化量之间的协方差所驱动的，而后者正比于该词元的优势。因此，一个高概率、高优势的动作会降低熵，而一个稀有、高优势的动作会提高熵。实测的协方差大多保持为正，所以熵单调下降——而且只有极小一部分词元携带着极端的协方差，主导了整个坍缩过程。

这提示我们可以专门针对这些词元进行干预。在一批 N 个采样词元上定义中心化的交叉乘积，

$$\text{Cov}(y_i) = \left(\log \pi_\theta(y_i) - \tfrac{1}{N}\textstyle\sum_j \log \pi_\theta(y_j)\right)\left(A(y_i) - \tfrac{1}{N}\textstyle\sum_j A(y_j)\right). \tag{12.11}$$

Clip-Cov 把协方差落在某个高值区间内的一小部分随机词元断开梯度（两个阈值都设在平均协方差的 500× 以上），选取比例在 2×10^{-4} 的量级。*KL-Cov* 则是对按协方差排序的前 k 比例的词元施加 KL 惩罚。报告的收益是在数学基准上平均 7B 规模 +2.0%、32B 规模 +6.4%。

工程实践: 负结果和正结果一样重要

Cui et al. (2025b) 报告，一个现成的熵奖励项对系数极其敏感：$\alpha \in \{10^{-4}, 10^{-3}\}$ 几乎不改变熵，$\alpha = 10^{-2}$ 导致熵爆炸，而唯一能稳住熵的那个取值又打不过基线。一个面向参考策略的 KL 惩罚能稳住熵，却损害性能。Mistral-AI et al. (2025) 独

立报告说，熵奖励在数学上导致坍缩、在混合数据上导致爆炸，而在训练过程中把 $\varepsilon_{\text{high}}$ 在 0.26 与 0.28 之间调节，反而给出了熵奖励做不到的稳定性。两个研究组、两个模型族、同一个结论：**通过截断的不对称性来控制熵，而不是通过一个熵项**。

常见陷阱: 熵不是一个目标值

有了(12.10)，人们很容易想把熵当作一个要维持在某个最优值上的量。提出这条定律的作者明确报告说，他们没有发现被干预到的熵水平与最终性能之间存在任何关系，并声明是否存在一个最优熵仍是开放问题。请把熵当作坍缩的先行指标，而不是一个设定点。

12.6.3 分叉词元的结果

Wang et al. (2025c) 从学习的一侧把这幅图景变得更清晰。在思维链轨迹中，只有一小部分词元具有高熵；这些词元充当分叉词元，在不同的推理路径之间做选择。他们报告说，RLVR 在很大程度上保留了基座模型的熵分布形态，主要调整的是那些本已高熵的词元——所以 RLVR 是在编辑一个早已存在的、很小的控制面。而且，把策略梯度更新限制在熵最高的前 20% 词元上，在 8B 规模与全梯度持平，在更大规模上则显著超过它，报告的收益是 Qwen3-32B 在 AIME′25 上 +11.04，而只在后 80% 上训练则明显损害性能。

核心思想: RLVR 的有效足迹很小

有三条独立的线索汇聚到这一点上。协方差分析说，一个极小的词元子集驱动着熵的动力学；80/20 的结果说，同一个子集驱动着学习；而在权重一侧，Mukherjee et al. (2025) 报告 RL 微调的是很小的子网络。2026 年一项关于权重奇异结构的研究发现了谱继承——RLVR 复用基座模型的权重谱，而改变与之相伴的奇异标架——并把这一点做成了一个优化器，达到给定准确率所需的步数大约只有原来的三分之一 (Zhu et al., 2026b)。如果 RLVR 触碰到的东西如此之少，那么它的效率和它的局限（第14章）就都不那么令人意外了。

表 12.4: 各种长度控制机制，以及各自擅长什么。真正要紧的区分是：一个被截断的样本究竟被当作缺失的证据（掩掉、打断），还是被当作失败的证据（惩罚）。

机制	行为	代价
掩掉被截断的样本	去除相关噪声；不施加变短的压力	无
软性超长整形	教会策略在上限之前收尾	一对超参数
硬性截断惩罚	压力很强；有噪声，会惩罚差一点就成的样本	—
组内过滤（GFPO）	只强化最短的那些正确样本	更大的 G
打断	把截断转化为一个正当的回合	提示工程

12.7 长度控制

P8。放任不管的话，回复的平均长度会一路上升，直到撞上生成上限。DAPO 的长度整形项是

$$\Phi(|y|) = \begin{cases} 0, & |y| \le L_{\max} - L_{\text{cache}}, \\ \dfrac{(L_{\max} - L_{\text{cache}}) - |y|}{L_{\text{cache}}}, & L_{\max} - L_{\text{cache}} < |y| \le L_{\max}, \\ -1, & |y| > L_{\max}, \end{cases} \tag{12.12}$$

它是在硬上限之下、宽度为 L_{cache} 的一段缓冲区上的一道软斜坡，被加到正确性奖励上。其他备选方案是：把被截断的样本整个从损失中掩掉（在 DAPO 的消融中被报告为单项收益最大的一步）、采样一个更大的组之后按长度过滤 (Shrivastava et al., 2025)，或者通过追加一句结束思考的话并要求模型基于已有内容给出答案，来打断生成 (Khatri et al., 2025)。

12.8 KL 之争

在 RLVR 中要不要保留 $\beta > 0$，是这个领域里最明确的一处正在进行中的分歧，而证据的分布模式很有教益。

去掉它。Yu et al. (2025c)、MiniMax (2025) 与Mistral-AI et al. (2025) 都设 $\beta = 0$，共同的论证是：长思维链 RL 本来就应该把策略带离参考模型很远，所以把它锚在那里既拖慢学习，又为了一份冻结的副本而白白占用内存。Cui et al. (2025b) 补充说，面向参考策略的 KL 惩罚能稳住熵，但会损害性能。

保留它。Liu et al. (2025d) 保留 KL，并配上周期性的参考重置——把 π_{ref} 重置为当前策略并重启优化器，使这个锚不会在渐近意义上把进展冻住——他们报告说，这正是使得长时间持续训练变得有用的原因。Tang et al. (2026) 报告了最强的反面证据：在万亿参数规模上，去掉 KL 会导致灾难性失败，训练与推理之间的对数概率差无界发散、熵坍缩、奖励崩盘，而保留 KL 时所有指标都保持健康。

核心思想：一个说得通的调和

KL 项的价值随训练—推理失配的严重程度、以及随策略的总漂移量而增长。它约束的是策略能走多远，而这恰恰是让两个引擎的数值保持一致的东西（P7）。在中等规模上，若验证器可靠、数值健全，这条约束是多余而昂贵的；而在极大规模上、在更激进的离策略程度下，或者在有 MoE 路由波动时，它就是承重的。这一读法与上述所有证据都相容，但截至 2026 年年中并未确立。请把 β 当作一件稳定性工具而非统计工具，并通过监控对数概率差来作决定。

12.9 基于值的分支

这个领域转向了无评论家的方法，但基于值的路线并没有失败——它被修好了，而了解它是怎么被修好的值得花些篇幅，因为这能隔离出评论家当初为什么会失败。

Yuan et al. (2025b) 指出了 PPO 在长思维链上坍缩的两个成因：值模型的初始化偏差，以及 GAE-λ 的设定失当。Yue et al. (2025b) 把这些修复组装成了 VAPO：

1. **值预训练**。在任何策略更新之前把评论家训练到收敛，然后加载那个检查点——这消除了节 6.2中的冷启动问题。
2. **解耦的 GAE**。用 $\lambda_{\text{critic}} = 1$ 得到无偏的回归目标，用一个更小的 λ_{policy} 来加快策略收敛。λ 的这两个角色此前一直被混为一谈。
3. **长度自适应的 GAE**。选取 λ_{policy}，使 GAE 系数之和随输出长度 ℓ 缩放：由 $\sum_t \lambda^t \approx 1/(1-\lambda) = \alpha\ell$，
$$\lambda_{\text{policy}} = 1 - \frac{1}{\alpha\ell}, \tag{12.13}$$
这使得 TD 误差的分布在长短序列之间被拉平。这是文献中最清晰的一个例子，说明什么叫长度感知的估计量修法，而不是长度感知的奖励修法。
4. **Clip-Higher 与词元级损失**，从 DAPO 引入。
5. **一个针对正例的似然项**，加在正确的采样轨迹上，$\mathcal{L} = \mathcal{L}_{\text{PPO}} + \mu\mathcal{L}_{\text{NLL}}$，取 $\mu = 0.1$，其动机是在那些几乎所有样本都错的任务上，把算力花在压制负例上并不划算。

核心思想: 为什么要讲那条没有走的路

VAPO 在教学上的意义在于，它表明评论家在长思维链上的失败是一次实现上的失败——冷启动、λ 取错、没有长度感知——而不是结构性的失败。因此，这个领域转向组基线是一个关于工程成本的选择，而不是值函数无法工作的证明。这一点在第18章中会再次变得相关，在那里多轮次的智能体式设定提供了稠密的中间奖励，使得评论家变得值得拥有。

12.10 混合专家模型的特有问题

P10。在稀疏 MoE 中，某个词元所激活的专家集合取决于路由决策，而这些决策在每次前向传播中都会重新计算，并且可能在一次权重更新之后就发生变化——Zheng et al. (2025a) 报告约有 10% 的被激活专家发生翻转，且层越深越严重。一个跨越两个不同活跃子网络计算出来的词元级重要性比值，测量的并不是它的公式声称要测量的东西。目前在用的应对有三种：在训练时重放采样阶段缓存下来的路由决策（正确，但在内存、通信与有效容量上代价高昂）；使用序列级似然，它对个别词元的波动不敏感（GSPO）；或者在训练的各个小步中重放推理时的路由，正如 Zhong et al. (2026) 所归纳的 R3 机制。既然 2026 年大多数前沿开源模型都是 MoE，这就不是一个小众问题。

12.11 设计空间

现在我们可以把一切组装起来了。表 12.5就是节 6.6 所许诺的那个对象：一个五轴的空间，本章中的每一个方法都是其中的一个点。

12.12 为什么这一文献中的大多数比较都检验效能不足

表 12.6中几乎每一个算法，都是通过让它和一个基线在固定预算下各跑一遍、再比较最终准确率来验证的。Khatri et al. (2025) 用 400,000 GPU 小时的消融加上一次 100,000 GPU 小时的验证运行，说明了这套方法论为什么不可靠。

他们核心的经验主张是：留出集上的期望奖励可以很好地描述为训练算力 C 的一个 S 形（*sigmoid*）函数，而不是幂律：

$$R_C - R_0 = (A - R_0) \cdot \frac{1}{1 + (C_{\text{mid}}/C)^B}, \tag{12.14}$$

表 12.5: 组相对的设计空间。读一篇新论文的方法是：在这张表里定位它，看清它改动了哪一根轴，再到表 12.1里查那根轴所针对的病理。2025–2026 年间几乎每一项贡献都只挪动了一两个格子。

轴	选项	支配什么
权重 ω	无（REINFORCE）· 截断的词元比值 · 截断且断开梯度的词元比值 · 几何平均的序列比值 · 逐分支的混合	P5、P6、P7；离策略容忍度；MoE 稳健性
优势 A	评论家 + GAE · 组均值 · 留一 · 组均值／组标准差 · 组均值／批标准差 · 过程值／片段值	P3、P9；方差；信用粒度
归一化器 $\mathcal{N}$	逐序列词元平均 · 全局词元平均 · 常数 · 序列平均	P2；长度动力学
正则项	无 · 面向 π_{ref} 的 k_3 KL · 带周期性重置的 KL · 面向高协方差词元的 KL · 熵奖励（不推荐）	P1、P7；漂移；大规模下的稳定性
数据规则	全部提示 · 丢弃退化组 · 重采样补满 · α 重新缩放 · 退役已饱和的提示 · 难度课程	P4；有效批量
长度规则	无 · 掩掉被截断的样本 · 软性整形 · 组内过滤 · 打断	P8

其中 A 是渐近通过率，B 是算力效率指数，C_{mid} 是实现总增益一半时所用的算力（拟合时排除了最早期的低算力区段）。这个分解有一个直接的方法论后果：

核心思想: 上限对效率，以及缩小版的苦涩教训

本章中的大多数干预——损失聚合、课程、长度惩罚、优势归一化——挪动的是 B（你多快到达那里），而不是 A（你最终到哪里）。两个 B 不同而 A 相同的方法，在小算力下排序不同，在大算力下排序相同。因此一个在小预算下看起来更好的方法，外推之后可能更差，而这使这一文献中很大一部分小规模消融失去效力。补救办法很便宜：从训练早期拟合出 (A, B)，然后比较拟合结果而不是比较终点。

他们报告的发现，既可以当作结果，也可以用来校准这些效应究竟有多大：

- 仅仅在语言模型头部使用 FP32 精度，就把渐近值 A 从 0.52 提到了 0.61——这是整项研究中最大的单轴效应，而且是一处数值改动而非算法改动。

表 12.6: 已命名的方法作为表 12.5这一设计空间中的点。「词元」与「序列」分别指词元级与序列级。这张表刻意做得很机械：智识上的内容在于每一处格子的改动所针对的那个病理。

方法	权重	优势	归一化	KL	独特之处
GRPO	截断词元	均值／标准差	逐序列	k_3	基线
Dr. GRPO	截断词元	均值	常数	—	消除 P2、P3
DAPO	截断词元，$\varepsilon_{\text{high}}>\varepsilon_{\text{low}}$	均值／标准差	全局词元	—	Clip-Higher、动态采样、超长整形
Lite PPO	截断词元	组均值／批标准差	全局词元	—	极简主义：两个技巧就够
GSPO	截断序列（几何平均）	均值／标准差	G	—	修复 P5、P10；极小的 ε
GMPO	截断词元，几何平均	均值／标准差	—	—	对离群值稳健的聚合
CISPO	截断且断开梯度的词元	均值／标准差	全局词元	—	修复 P6；没有梯度被湮灭
DHPO	逐分支的混合	均值／标准差	全局词元	—	调节词元／序列之间的权衡
GPG	无	均值，经 α 修正	全局词元	—	无代理目标；$\alpha = B/(B-M)$
RLOO	无	留一	G	KL	无偏基线，早于 GRPO
VAPO	截断词元，$\varepsilon_{\text{high}}>\varepsilon_{\text{low}}$	评论家 + 自适应 GAE	全局词元	—	把基于值的做法做对
ScaleRL	截断且断开梯度的词元	均值／批标准差	全局词元	—	异步、FP32 头、零方差过滤

- CISPO 与 GSPO 在 A 上都大幅胜过 DAPO 式的词元截断；最终选用 CISPO 是因为它对超参数更稳健，把它的截断阈值在 $\{4, 5, 8\}$ 之间变动没有产生显著差异，而 DAPO 的 $\varepsilon_{\text{high}}$ 却会改变 A 本身。
- 从完整配方出发做留一消融，在 A 上影响很小，在 B 上则影响一致，这印证了上限与效率的分离。
- 可预测性在批量放大 2.5×、生成长度到 32,768、数学加代码的多任务训练，以及一个大型 MoE 上都成立。

一个附带的告诫：Tang et al. (2026) 在万亿规模上做了一次基座模型、学习率与数据都相同的正面对比，报告说 CISPO 与 DAPO 会放大低概率词元的梯度，因而比 GRPO 更快地降低全败组的比例，带来更快的奖励增长——但不稳定性的排序是完全一样的，CISPO > DAPO > GRPO。速度与脆弱性是一起来的。他们还报告说，GSPO 能维持高熵，但长度增长有限。(12.14)是否在各个模型族与各个领域上都成立，以及 A 在机制上由什么决定，都还是开放问题。

工程实践: 如何做一次站得住脚的比较

1. 先把数值问题定下来（FP32 头，或者全程 FP16），并核实训练与采样之间的对数概率差很小。否则你测的是 P7。
2. 至少使用两个互不相关的基座模型族 (Shao et al., 2025)。
3. 匹配算力，而不是匹配步数：各方法每步生成的词元数不同。
4. 跑得足够久，以便拟合(12.14)，并连同种子方差一起报告 (A, B)，而不是报告单个终点。
5. 同时报告 pass@1 和大 k 下的 pass@k；一个在前者上赢、在后者上输的方法，是用覆盖度换来了准确率（第14章）。
6. 写明你的归一化器、你的优势分母、你的截断边界、你的 β、你对退化组的处理策略，以及你的截断策略。省略这些的论文无法被复现，而大多数论文至少省略了其中一项。

12.13 哪些已成定论，哪些还没有

直说吧，因为读到这一章的读者理应得到一个结论。

已经足够确定，可以当作默认值来教。去掉逐回复的长度归一化。使用非对称截断，或者截断重要性权重而不是截断目标函数。过滤退化组，或者施加 α 修正。先修数值，再修算法。把训练—推理失配当作一个独立于信赖域比值的因子来纠正。在不止一个模型族上验证。拟合缩放曲线，而不是相信小预算下的排序。

确实尚无定论。要不要保留 KL 惩罚（依赖于规模，节 12.8）。逐组标准差除法究竟是一个偏差还是一个意外的修正（节 12.3）。比值应当是词元级、序列级还是混合的（节 12.5.3）。是否存在一个最优的熵。pass@k 应当是一个目标还是仅仅是一个诊断量（节 14.7）。以及这一切是否能迁移到多轮次的智能体式训练中，那里早期证据表明，静态推理场景下的结论并不成立（第18章）。

本章小结

- GRPO 变体这一文献是按十种病理组织起来的（表 12.1），而每一个有名字的算法，都是一个五轴设计空间中少数几处格子的改动（表 12.5 and 12.6）。

- 最大的效应来自记账，而不是来自策略梯度。GRPO 的逐序列长度归一化器产生了一个依赖长度的步长，对长的错误答案惩罚不足；它的逐组标准差除法则抬高了信息量最少的那些提示的权重。
- 这两个归一化器与第三种效应相混杂——丢弃退化组所导致的分母虚高——这正是为什么各自独立的研究组会真诚地发表关于它们的相互矛盾的消融。当下诚实的说法是：只要退化组的记账处理得当，逐组标准化、批级标准化与不做标准化，表现是相当的。
- 退化组不是一种边缘情形：全对的组所占的比例在训练中单调上升，因此除非对提示做重采样、把它们退役，或者把损失按 $B/(B-M)$ 重新缩放，否则有效批量会不断缩小。
- GSPO 与 CISPO 对词元级重要性比值给出了相反的诊断——一个说它是无法校正的噪声，一个说截断它会毁掉有信息量的词元——而它们各自说对了不同的项。GSPO 的几何平均要求一个比 GRPO 小三个数量级的截断范围。
- 只要离策略权重不可靠，就截断权重而不是截断目标函数：衰减保住了湮灭会丢掉的信息，而同一种构造也修好了第22章中的数值失配。
- 对称截断是对稀有词元征收的累退税，因此压制探索；非对称截断才是真正管用的熵控制工具。在这一设定下熵奖励项行不通，至少有两个研究组独立报告了这一点。
- 熵是被花掉用来买奖励的，遵循一条两参数的经验定律，而其动力学被极小一部分高协方差的「分叉」词元所主导——正是驱动学习的同一个子集。RLVR 的有效足迹，无论以词元计还是以权重计，都远小于它的开销所暗示的。
- 大多数已发表的比较都检验效能不足，因为各种干预主要挪动的是算力效率而不是渐近值，所以小预算下的排序无法外推。请改为拟合 S 形曲线(12.14)。

延伸阅读

必读的次序是Yu et al. (2025c)，然后Liu et al. (2025g)，然后Zheng et al. (2025a)，然后MiniMax (2025)。DAPO 给出了四项修法以及一份逐项累加的消融，这是「这些选择确实要紧」这件事最好的单篇证明。Dr. GRPO 是这一领域里最干净的一篇统计批评。GSPO 包含了最锐利的理论论证，即重要性采样在词元粒度上是不适定的，而它关于 MoE 与基础设施的观察，是那种只有真正跑过前沿模型才会得到的二阶发现。CISPO 只是一份模型报告里勉强两页的内容，却包含了本章中最可迁移的想法。

Khatri et al. (2025) 接下来应当被当作一篇方法论论文而不是算法论文来读；它是这个领域目前最接近于「怎样回答『这个是不是更好』」的一份标准。请把它与Cheng et al. (2026) 关于如何分配采样预算的工作、以及与Tang et al. (2026) 那次与 ScaleRL 若干结论相矛盾的大规模正面对比——尤其是关于 KL 惩罚的那些——配着读，后者很好地提醒我们：规模会改变答案。

关于熵，Cui et al. (2025b) 与Wang et al. (2025c) 是互补的，应当一起读；Park et al. (2025b) 为 DAPO 那条被抄得最多的启发式提供了机制。关于那条没有走的路，Yue et al. (2025b) 连同Yuan et al. (2025b)，是「评论家的失败有可修复的原因」这一论点的最佳论证。Liu et al. (2025h) 则是对整桩事业的必要纠偏，他们发现两项干预就能挽回整套方案的大部分收益。

习题

12.1（**归一化器，习题详演。**）对于一批包含两条回复的数据，$|y_1| = 200, A_1 = +1$ 且 $|y_2| = 2000, A_2 = -1$，计算在表 12.2的四种归一化器之下，各条回复分别获得的总梯度权重。然后指出哪一种归一化器能使错误答案上的逐词元惩罚等于正确答案上的逐词元奖励，并解释为什么那正是我们想要的性质。

12.2（**标准差除法作为一种难度权重。**）推导 (12.2)，并对 $G = 8$ 画出提示权重关于 $\hat{\mu}$ 的曲线。再计算节 12.4.1的可学习性论证所应给出的权重（正比于奖励方差），把它画在同一组坐标轴上。量化两者的错配，并解释为什么 $G = 8$ 使得 (12.2)的极端取值只能在 $\hat{\mu} \in \{1/8, 7/8\}$ 处达到，而不会任意大。

12.3（**训练过程中的退化组。**）利用 (12.4)，设逐提示成功概率的总体分布初始为 Beta(2, 2)，训练之后为 Beta(6, 2)。对 $G \in \{4, 8, 16, 64\}$ 计算两种情形下退化组的期望比例，并据此算出有效批量缩小的倍数。然后计算每种情形下的 α 修正，与 Chu et al. (2025b) 所报告的 1.5–4.0 区间作比较。

12.4（**正确地移植 GSPO。**）给你一份可用的、$\varepsilon = 0.2$ 的 GRPO 实现，要求你改成 GSPO。写下所有必需的改动，包括截断范围。然后，假定逐词元的对数比值是 i.i.d. 且标准差为 σ，推导 σ、回复长度，以及使得目标比例的序列被截断所对应的 ε 三者之间的关系。核实你的答案与所报告的 $3\text{–}4 \times 10^{-4}$ 一致。

12.5（**CISPO 的偏差。**）证明权重不被截断时，(12.8)恰好就是 定理 6.1的策略梯度经重要性采样重新加权的结果。然后计算在 C 处作上界截断所引入的偏差，写成关于

比值分布的一个期望，并分别确定正优势与负优势下偏差的符号。这个截断压制了哪些词元，那是不是你希望被压制的那一批?

12.6（**截断是累退的。**）对 $\varepsilon = 0.2$，把一个词元可实现的最大概率增量 Δp 计算为其当前概率 p 的函数，并画出 Δp 与相对增量 $\Delta p/p$。找出在哪个 p 之上上界截断完全不起约束作用。然后对 $\varepsilon_{\text{high}} = 0.28$ 重做一遍，并量化 Clip-Higher 在 $p = 0.01$ 处为一个词元额外让出了多少余量。

12.7（**熵的记账。**）取(12.11)的协方差机制。对一个概率为 $(p, 1-p)$ 的二元词表，且第一个词元上带有优势 A，计算在一次步长为 η 的策略梯度更新之后熵的变化，精确到一阶。核实所宣称的符号：高概率高优势的词元降低熵，而稀有的高优势词元提高熵。求出交叉点处的概率。

12.8（**设计一个变体。**）从表 12.1中选一个你认为处理得不够好的病理。提出一处只改动表 12.5中恰好一根轴的修改，精确说明目标函数中的哪一项发生了变化，预测它对奖励、熵与平均长度的效应符号，并指明能把你的机制与两个最可能的混杂因素区分开来的那个消融实验。

12.9 ⋆（**GSPO 与 CISPO 可以组合吗?**）写下一个目标函数，它把几何平均的序列比值用作施加在 $\log \pi_\theta$ 上的、断开梯度的衰减权重，i.e., 把 GSPO 的粒度与 CISPO 的处理方式结合起来。推导它的梯度。判定它是否（a）在不截断的极限下是一个有效的重要性加权策略梯度，（b）保住了每一个词元的梯度贡献，以及（c）继承了 GSPO 对 MoE 路由波动的不敏感性。然后用节 12.5.3中方差与选择效应的区分，解释为什么这个对象并不显然优于它的任何一个亲本。

12.10 ⋆（A **由什么决定?**）(12.14)把渐近值 A 当作一个拟合出来的常数。请用一些训练之前就可测量的量——基座模型的 pass@k 曲线、熵、处在「边界」区段的提示比例——提出一个关于 A 的机制性解释，并设计一个在可负担规模上能证伪它的实验。

12.11 †（**完整的矩阵。**）把第6章那个项目中的实验框架扩展开来，实现表 12.6中所有不需要评论家的行。在两个互不相关的基座模型族上以匹配的算力各跑一遍，对每次运行拟合 (12.14)。报告一张带种子方差的 (A, B) 表格，并说明节 12.13 中那些已成定论的结论你复现了哪些、没复现哪些。然后在每个模型族上各跑一个配置，把奖励换成抛硬币，并报告Shao et al. (2025) 的模型族依赖性是否复现。

第13章

信用分配与过程监督

核心思想: 更细的信用只是在重新分配一个固定的总量，它并不制造信号

沿一条轨迹的真实逐词元优势之和，恰好等于该轨迹的优势。因此本章中的每一种方案——过程奖励模型、蒙特卡洛前缀值、片段与轮次分解、采样树——都只是把一个标量重新分配到 10^4 个位置上的规则，而不是从环境中获取更多信息的办法。重新分配买到的是方差：有了精确的值，梯度的二阶矩就不再随回复长度线性增长。它付出的是偏差，因为这些值必须被估计出来。整个主题就是这场交换的算术。

第11章与第12章发展出了一族优势沿序列恒定的方法。这是一个引人注目的选择：一段 30,000 词元的推导只拿到一个数，而其中每个词元都被以相同的力度推向相同的方向。本章要问的是，做得更好能换来什么。节 13.1精确表述这个问题，并推导出可供争取的确切奖赏。节 13.2 and 13.3处理学出来的步骤级奖励。节 13.4发展这一设定下最干净的估计量，它利用了语言模型环境可以被重置这一事实。节 13.5 and 13.6讲更粗的单元，以及以算力高效的方式取得它们的办法。节 13.7是一份诚实的评估，而且基本上是祛魅式的。节 13.8论证，同一个学出来的对象，通常花在推理时比花在训练时更划算。

13.1 精确表述这个问题

定义 13.1 (信用分配问题). 给定一条从 π_θ 采样出的轨迹 $(s_1, a_1, \dots, s_T, a_T)$ 与一个终局奖励 r，产出一组系数 $\hat{A}_1, \dots, \hat{A}_T$，用于代理目标(6.17)。其目标是真实优势

$$A^{\pi_\theta}(s_t, a_t) \;=\; Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t), \tag{13.1}$$

而评判一个估计量的标准，是它作为(13.1)的估计量的偏差与方差，而不是奖励的任何性质。

语言模型 MDP 使得(13.1)异乎寻常地容易处理。由于状态转移是确定性的拼接(节 2.3)，$Q^{\pi_\theta}(s_t, a_t) = V^{\pi_\theta}(s_t \oplus a_t) = V^{\pi_\theta}(s_{t+1})$：一个词元的动作值就是它所产生的那个前缀的状态值。取 $\gamma = 1$ 并采用只有终局奖励的(2.6)，我们得到如下结果。

命题 13.2 (优势就是前缀值的增量)**.** 在具有确定性转移、只有终局奖励且 $\gamma = 1$ 的词元级 *MDP* 中，

$$A^{\pi_\theta}(s_t, a_t) \;=\; V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t), \qquad \text{以及} \qquad \sum_{t=1}^{T} A^{\pi_\theta}(s_t, a_t) \;=\; r - V^{\pi_\theta}(x). \tag{13.2}$$

证明. 第一个恒等式就是上面那个代换。至于第二个，求和是裂项相消的；$V^{\pi_\theta}(s_1) = V^{\pi_\theta}(x)$，而 $V^{\pi_\theta}(s_{T+1}) = r$，因为 s_{T+1} 是终局状态。 □

有两条推论组织起了本章。第一，整个信用分配问题坍缩成了估计前缀上的一个标量函数：在当前策略之下，这个前缀的某条延续能够成功的概率。本章中的一切——一个 PRM 的步骤分数、一个评论家的输出、一个蒙特卡洛补全率、一棵采样树的内部节点——都是 V^{π_θ} 在某个前缀处的估计量，而各方法之间的差别，就是它们估计方式上的差别。第二，裂项相消的恒等式说明总的信用是固定的。摆在桌面上的只有重新分配。

推导: 完美的信用分配值多少

记 $g_t = \nabla_\theta \log \pi_\theta(a_t \mid s_t)$，并考虑策略梯度的两个估计量：均匀的那个，对所有 t 取 $\hat{A}_t = r - b(x)$；以及理想的那个，$\hat{A}_t = A^{\pi_\theta}(s_t, a_t)$。两者都是无偏的（前者由定理 2.2，后者由定理 13.2加上全期望法则）。它们在二阶矩上不同。采用 第6章中的简化假设 $\mathbb{E}[g_t g_{t'}^\top] = \mathbb{1}[t = t']\,\Sigma$，并取 $b = V^{\pi_\theta}(x)$。则

$$\text{均匀：}\quad \mathbb{E}\left\| \sum_t (r - b) g_t \right\|^2 \;\approx\; \operatorname{tr}\Sigma \sum_{t=1}^{T} \mathbb{E}[(r-b)^2] \;=\; T\;\mathbb{V}\mathrm{ar}[r]\;\operatorname{tr}\Sigma, \tag{13.3}$$

$$\text{理想：}\quad \mathbb{E}\left\| \sum_t A^{\pi_\theta}(s_t, a_t) g_t \right\|^2 \;\approx\; \operatorname{tr}\Sigma \sum_{t=1}^{T} \mathbb{E}[\delta_t^2] \;=\; \mathbb{V}\mathrm{ar}[r]\;\operatorname{tr}\Sigma, \tag{13.4}$$

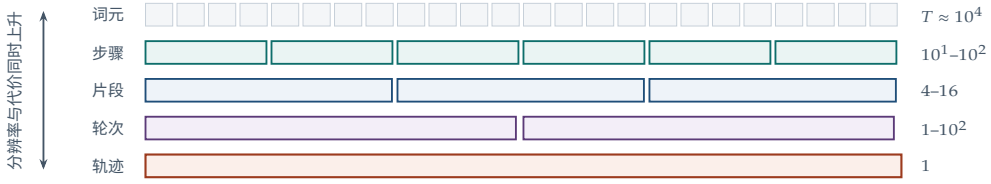


图 13.1: 单条回复上的粒度谱系。每一行都是把同一条词元序列划分成若干单元的一种分法，优势在每个单元内保持恒定；右侧一列是每条轨迹需要估计其值的状态数。仅用结果的组相对方法活在最底下一行，只估计一个值（组均值）；带评论家的 PPO 活在最上面一行，靠第二个网络的一次前向传播拿到它的 10^4 个估计。中间的一切都必须用额外的生成来支付它的估计，而这正是表 13.1所核算的。

其中 $\delta_t = V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$。最后那个等号才是有用的一步。由于 $\mathbb{E}[V^{\pi_\theta}(s_{t+1})\,|\,s_t] = V^{\pi_\theta}(s_t)$，序列 $V^{\pi_\theta}(s_t)$ 是一个鞅，而 δ_t 是它的增量：它们均值为零且两两不相关，于是由(13.2)，$\sum_t \mathbb{E}[\delta_t^2] = \mathbb{V}\mathrm{ar}[\sum_t \delta_t] = \mathbb{V}\mathrm{ar}[r]$。均匀估计量在全部 T 个位置上每一处都付出 $\mathbb{V}\mathrm{ar}[r]$，因为它的系数在各位置之间完全相关；理想估计量只付出一次 $\mathbb{V}\mathrm{ar}[r]$，并把它摊在各个位置上。

所以奖赏是梯度二阶矩上的一个 T 倍因子，而它恰好就是第6章所诊断出的那种随长度线性增长。对 $T = 10^4$ 而言那是四个数量级，这正是为什么这个主题在纸面上看起来如此诱人。本章余下的部分讲的是为什么这里面几乎没有一分能在实践中兑现：上面推导中的每一项都假定 V^{π_θ} 被精确已知，而一个估计 $\hat{V}$ 会贡献一份误差，它在每个词元上都充当偏差，无论怎么在批次上平均都消不掉。

核心思想: 方差会被平均掉，偏差不会

均匀估计量无偏但有噪声。每一个更细的估计量都把其中一部分噪声换成了偏差——来自一个拟合不良的评论家、一个标错的步骤、一个有限样本的蒙特卡洛估计，或者一棵相关的采样树。噪声会被批量大小和步数除掉；偏差不会。这种不对称性是预测哪些信用分配方案能在真实训练中活下来的最好的单一指标，也是为什么这个领域用脚投票选择了最粗糙的那个管用的估计量。

13.1.1 粒度谱系

在「每条轨迹一个优势」与「每个词元一个优势」之间存在一个连续谱，其索引是优势在其上保持恒定的那个单元。图 13.1展示了这种嵌套关系，而 表 13.1给出了记账。

表 13.1: 把粒度谱系看作一个估计问题。「代价」指的是在已经支付的 G 条采样轨迹之外，每条轨迹还需要的额外生成；K 是每个被估计状态的延续条数，M 是被估计状态的个数。偏差那一列才是实践中决定成败的一列，因为偏差挺得过批次平均。

单元	由什么界定	值由什么估计	额外代价	主要偏差来源
词元	一个词表项	评论家 $V_\psi(s_t)$；或一个学出来的逐词元对数比值	一次前向传播，或无	在一个 10^4 词元、目标为二值的回归上评论家拟合不良
步骤	换行、句子，或一个「`Step` k`:`」标记	PRM 分类器；蒙特卡洛补全率；MCTS 节点统计量	MK 条补全，或一次 PRM 前向	步骤边界的歧义；PRM 标签噪声；对学出来的 PRM 的作弊
片段	一条划分规则（切分点、固定数目、熵）	从片段边界出发做蒙特卡洛	MK 条补全	划分的任意性；K 小时的蒙特卡洛噪声
轮次	多轮次回合中模型的一条消息	环境给出的逐轮次奖励；从轮次边界出发做蒙特卡洛	环境重置	环境随机性；逐轮次奖励设定失当
轨迹	整条回复	组均值、留一，或 $V_\psi(x)$	无（已经付过了）	估计量本身没有；轨迹内分辨率为零

在开始之前，关于表 13.1有两个结构性事实值得强调。只有最上和最下两行是免费的：评论家把 10^4 个估计摊销进一次前向传播，而组基线复用了你手里已有的采样轨迹。中间的一切都是在用生成来购买分辨率，其兑换率由 MK 决定。而最底下那一行是唯一一行完全没有估计量偏差的，当另一个选项是把一份偏差均匀地施加到每条轨迹的每一个词元上时，这是一个相当可观的优势。

13.2　过程奖励模型

定义 13.3 (过程奖励模型)**.** 固定一种把回复划分成若干步骤的分法，$y = (\sigma_1, \dots, \sigma_M)$。一个过程奖励模型是一个函数 $v_\phi(x, \sigma_{\le m}) \in [0,1]$，训练目标是预测一个逐步骤的标签；与之相对，一个结果奖励模型只对完整的 y 打分。当需要一个单一分数时，一个聚合函数 agg 把向量 $(v_1, \dots, v_M)$ 变成一个数。

13.2.1 步骤标签从哪里来

三种来源，按历史顺序、也按自动化程度递增排列。

人工标注。 Lightman et al. (2024) 让标注者把模型生成的 MATH 解答的每一步标为正面、中性或负面，在数万条解答上发布了大约 8×10^5 个步骤标签（PRM800K 数据集）。这是黄金标准，而它无法规模化：成本随步数线性增长，标注者自己必须精通竞赛数学，而且这些标签是针对生成这些解答的那个策略的，因此会随着策略变强而变陈旧。

蒙特卡洛补全率。 Wang et al. (2024f) 把人去掉了。定义步骤 m 的标签的方式是：用一个补全策略从前缀 $\sigma_{\le m}$ 采样 K 条补全，记录它们是否到达正确的最终答案：硬标签是 $\mathbb{1}$[至少有一条补全是正确的]，而软标签是正确的比例。算法 13.1把它写了出来。请注意这个标签实际测量的是什么：不是这一步是否正确，而是补全策略是否还能从它那里挽回。这是两个不同的量，而它们之间的差距是反复出现的麻烦来源。

由搜索导出的标签。 如果愿意跑树搜索，那么在内部节点上累积起来的统计量就是免费的步骤标签。Luo et al. (2024) 用一种分治式的蒙特卡洛树搜索让这件事变得高效，它沿一条轨迹二分查找第一个错误，把所需补全数从关于 M 的线性降到对数；Guan et al. (2025) 则在一个自我改进的循环中，把 MCTS 导出的步骤值既当作 PRM 的训练数据，也当作搜索的引导。

常见陷阱: 没有人知道一个步骤到底是什么

上面每一种方法都预设了一种划分，而对长思维链来说，这种划分并没有良好的定义。换行是一个句法上的偶然。「`Step 3:`」这样的标记只有在格式奖励要求过它时才会出现。更糟的是，长思维链轨迹中含有刻意走错的步骤：一条轨迹写下一个错误的展开式，两行之后察觉，然后重来——它展示的恰恰是 RLVR 因能诱发而备受推崇的那种行为（节 11.3），而一个按步骤正确性训练出来的 PRM 会把这个探索性的步骤标成负面。如果你惩罚它，你就是在训练策略不去探索；如果你不惩罚，你的步骤标签就自相矛盾。这不是一个可以修好的标注 bug，而是被标注对象本身的一种歧义，也正是为什么步骤粒度对于长的深思型轨迹，远不如对于当初孕育出过程监督的那些简短、线性、五步的 GSM8K 解答那样有用。

算法 13.1 按补全率自动标注过程，依照Wang et al. (2024f)。软版本返回补全率本身；硬版本在零处对它取阈值。代价是每条被标注的回复 MK 条补全，这与 算法 13.2的记账完全相同——两个过程的差别只在于结果是被用作一个模型的训练目标，还是被直接用作优势。

输入： 带参考答案 $a^\star$ 的提示 x、被切分成步骤 $(\sigma_1, \dots, \sigma_M)$ 的回复、补全策略 μ、验证器 r_{ver}、每步补全数 K

for $m = 1$ **to** M **do**
  $s \leftarrow x \oplus \sigma_1 \oplus \cdots \oplus \sigma_m$ ▷ 截至步骤 m 的前缀
  采样 $y^{(1)}, \dots, y^{(K)} \sim \mu(\cdot \mid s)$ ▷ 复用前缀的 KV 缓存
  $\hat{p}_m \leftarrow \frac{1}{K} \sum_{k=1}^{K} r_{\text{ver}}(x, s \oplus y^{(k)}; a^\star)$
  $\ell_m \leftarrow \hat{p}_m$ ▷ 软　或　$\ell_m \leftarrow \mathbb{1}[\hat{p}_m > 0]$ ▷ 硬
end for
return 标签 $(\ell_1, \dots, \ell_M)$

13.2.2 训练与聚合

训练本身乏善可陈：逐步骤的二分类，通常的实现方式是在每个步骤边界处读取某个指定词元的 logit，

$$\mathcal{L}_{\text{PRM}}(\phi) = -\mathop{\mathbb{E}}_{(x,y,\ell)\sim\mathcal{D}} \sum_{m=1}^{M} \Big[\ell_m \log v_\phi(x, \sigma_{\le m}) + (1-\ell_m) \log\big(1 - v_\phi(x, \sigma_{\le m})\big) \Big]. \tag{13.5}$$

在软标签下，这是针对一个落在 $[0,1]$ 中的目标的交叉熵，也就是一场披着分类外衣的、关于补全率的回归。

有意思的问题是聚合，因为四种常见的选择编码了关于一份解答之结构的、确实不同的主张（表 13.2）。

这里有一个值得点名的测量陷阱。Zhang et al. (2025e) 报告说，在那些按 best-of-n 表现挑选出来的 PRM 中，最小步骤分数有很大一部分落在最终答案那一步上：模型学会了表现得像一个套着过程接口的结果验证器。他们还报告说，不可靠的策略会产出答案正确而过程有缺陷的回复，PRM 会容忍这些回复，而这种容忍会抬高 best-of-n 的分数——所以 best-of-n 准确率是一种有偏的、把 PRM 当作 PRM 来评估的方式。Zheng et al. (2025b) 与 Song et al. (2025c) 的存在，就是为了直接测量那件事：要求模型指出最早出错的那一步；而那些在 best-of-n 下看起来很强的 PRM，在这种测量下往往表现平平。读一篇 PRM 论文时，请检查它报告的是哪一种评估。

表 13.2: 过程分数的聚合函数，以及每一个所作的主张。对于每一步都必需的推理链，取最小值是理论上有动机的选择：只有当最弱的一步正确时解答才正确，而且与乘积不同，它不惩罚长度。「如何失效」一列解释了为什么这个选择通常终究不是决定一次实验成败的东西。

聚合方式	公式	它编码的主张，以及它如何失效
最小值	$\min_m v_m$	一条链的强度取决于它最弱的一环；对长度中立。失效之处在于它被单个校准得最差的步骤所主导，因此即便质量没有变化，它的方差也会随 M 增长。
乘积	$\prod_m v_m$	各步骤是独立的伯努利事件，而我们想要它们全部成功的概率。失效之处在于它关于 M 严格递减：一份长的正确解答得分会低于一份短的错误解答。
末项	v_M	只有终点要紧——i.e., 这个 PRM 是被当作结果奖励模型在用。失效之处在于它丢掉了你花钱买来的那些过程信息。
平均	$\frac{1}{M}\sum_m v_m$	我们要的是平均的步骤质量。失效之处在于一个致命错误会被许多容易的正确步骤补偿掉，而用一堆平凡正确的步骤灌水会抬高分数。

13.2.3 证据：出色的重排序器，不令人信服的奖励

重排序方面的结果是扎实且已被复现的。Lightman et al. (2024) 报告说，在从 n 条采样出来的 MATH 解答中做选择时，一个过程监督的奖励模型大幅优于一个结果监督的奖励模型和多数投票，而且差距随 n 增大而拉开——这是一个验证器其错误与策略错误的相关性比 ORM 更低的标志。Uesato et al. (2022) 在 GSM8K 上做了受控比较，报告了一个更微妙、也更常被误引的发现：结果监督与过程监督的训练达到了相近的最终答案准确率，但过程监督产生了明显更好的 推理准确率，i.e., 大幅减少了那些走错路却答对的轨迹。这两个结果讲的都是选择，以及轨迹质量。两者都不是「步骤奖励能让策略梯度训练变得更好」的证明。

第二个主张要立起来困难得多，而反面证据异乎寻常地明确。DeepSeek-AI (2025) 报告说，他们尝试过过程奖励模型和蒙特卡洛树搜索，最后都放弃了，并为 PRM 给出了三条理由，而它们恰恰就是本章一直在累积的那三条：在一般性推理中很难定义一个细粒度的步骤，很难判定一个中间步骤是否正确，而且——决定性的一条——把一个基于模型的 PRM 放进训练循环里会招来奖励作弊，而为对抗它所需的反复重训又会让整条流水线变得复杂。Kimi k1.5 (Kimi Team, 2025a) 以同样的取舍达到了可比的结

果。与此同时，Zhang et al. (2025e) 报告说，蒙特卡洛估计出来的步骤标签比人工标签或评判模型标签的泛化性更差，因为决定它们的是补全策略而不是那个步骤。

常见陷阱: PRM 是一个学出来的奖励模型，而且它就在优化循环里面

节 5.8中的一切都原封不动地适用于 PRM，而且有两件事让情况更糟。第一，一个 PRM 在每条轨迹上被查询 M 次而不是一次，所以一个在找打分错误的策略，每条采样轨迹就能从错误分布中多抽 M 倍的次数。第二，可被利用的错误位于轨迹内部，在那里没有任何验证器能反驳它们：一个在 PRM 看来像是进展、实则不是的步骤会得到奖励，而最终答案的核查器永远看不出差别，因为奖励早就已经付出去了。已被报告的这类作弊手法包括：把对前提的自信复述当成「步骤」发出来、把一个步骤分解成许多平凡正确的小步骤以抬高平均聚合，以及采用 PRM 训练分布的表层措辞。只有在能够审计采样轨迹、并且有一个占主导地位的硬性结果项时，才把 PRM 用作训练时奖励（节 11.6）。

历史注记: 过程监督的两个时代

过程监督是 2022–2023 年间在 GSM8K 与 MATH 上发展起来的，在那里一份解答是五到十五个简短、线性、可逐条核查的步骤，而一个正确的步骤没有歧义 (Uesato et al., 2022; Lightman et al., 2024)。它在那里效果很好，而且那些结果站住了。随后它在 2025 年被长思维链 RLVR 继承过去，在那里一条轨迹有数百个步骤，因为会回溯所以是非线性的，而且是由一个其全部价值就在于探索错误分支的策略生成的。早期证据几乎没有一条能迁移过去，而 2025–2026 年间对 PRM 的诸多失望，更应该被读作一次迁移的失败，而不是一次证伪。正确的总结不是「过程监督不管用」，而是「过程监督是在一个我们已经离开的场景里被验证的」。

13.3 隐式与派生的过程奖励

如果步骤标签是瓶颈，那么一条出路就是不用它们也能拿到步骤值。有三种构造做到了这一点，按原理性递减排列。

13.3.1 对数比值就是优势

推导: 一个参数化为对数比值的结果模型就是一个过程模型

把定理 6.3用在词元粒度而非序列粒度上。在每个状态处求解 $\max_\pi \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\star(s,a)] - \beta\mathbb{D}_{\mathrm{KL}}(\pi(\cdot|s)\,\|\,\pi_{\mathrm{ref}}(\cdot|s))$ 得到 $\pi^\star(a|s) = \pi_{\mathrm{ref}}(a|s)\exp(Q^\star(s,a)/\beta)/Z(s)$，其中 $V^\star(s)=\beta\log Z(s)$，于是

$$\beta\log\frac{\pi^\star(a|s)}{\pi_{\mathrm{ref}}(a|s)} = Q^\star(s,a) - V^\star(s) = A^\star(s,a). \tag{13.6}$$

所以对于它所拟合的那个奖励的软最优策略而言，一个逐词元的对数比值恰好就是一个逐词元的优势。现在让一个结果奖励模型被参数化为 $r_\phi(x,y) = \beta\log\frac{\pi_\phi(y|x)}{\pi_{\mathrm{ref}}(y|x)}$ 并且只在回复级的标签上拟合。由于序列对数比值是词元对数比值之和 (2.2)，这个被训练来给整条回复打分的模型，不花任何额外代价就产出了它自己分数的一个逐词元分解，其部分和就是前缀值。这就是Yuan et al. (2025a) 的隐式 *PRM*，而它与Rafailov et al. (2024b) 的观察——一个用 DPO 训练出来的策略隐式地定义了一个词元级 Q 函数——是同一件事，只是从另一个方向读的。

Yuan et al. (2025a) 报告说, 一个仅用结果标签训练的隐式 PRM, 在 MATH 上用不到 1/38 的训练数据就胜过了一个 Math-Shepherd 式的 MCTS 基线, 而且——这才是应该让人吃惊的结果——在此之上再加入真正的 Math-Shepherd 步骤标签并不带来进一步的改进。PRIME (Cui et al., 2025a) 把这一点变成了一个训练算法：在线维护这个隐式 PRM，只用结果标签在策略自己的采样轨迹上更新它，并把它的词元级分数当作稠密奖励使用。在线更新是那个承重的部分, 因为正是它防止了奖励模型随着策略漂移而跑出分布之外，而后者正是大多数 PRM 作弊背后的机制。从 Qwen2.5-Math-7B-Base 出发，他们报告在各推理基准上相对其 SFT 模型平均改进 15.1%，而且所得到的模型用 10% 的训练数据在七个基准上超过了 Qwen2.5-Math-7B-Instruct。按节 11.8的意义，请把这些当作单一模型族上的单篇结果来看待。

代码 13.1: 从一个参数化为对数比值的结果奖励模型出发，经由 (13.6)得到步骤优势。那个逐词元的量噪声太大，无法直接使用——凡是奖励模型只是觉得措辞不寻常的地方，它都会很大——所以要先求和到步骤边界。这里没有任何一处需要步骤标签。

```
@torch.no_grad()
def implicit_step_advantages(logp_rm, logp_ref, step_ends, beta=0.05):
    """logp_rm, logp_ref: [T] per-token log-probs of the same response under the
    implicit reward model pi_phi and the frozen reference. step_ends: increasing
```

```
    token indices at which each reasoning step finishes."""
    token_adv = beta * (logp_rm - logp_ref)          # 恰好等于 A*(s_t, a_t)
    prefix_value = torch.cumsum(token_adv, dim=0)    # V*(s_t)，直到 V*(s_1)

    v_after = prefix_value[step_ends]                # 每一步之后的值
    v_before = torch.cat([v_after.new_zeros(1), v_after[:-1]])
    return v_after - v_before                        # 每一步一个优势
```

13.3.2 根本不是值估计的那些信号

还有两族方法在不估计 V^{π_θ} 的情况下给出了不均匀的词元信用，而把话说清楚是值得的：它们做的是另一件事。

第一族利用策略自身的统计量。节 12.6发展了协方差与分叉词元的结果：一小部分高熵词元同时驱动着熵的动力学和学习，而把策略梯度更新限制在按熵排序的前 20% 词元上，能够持平甚至超过全梯度训练 (Cui et al., 2025b; Wang et al., 2025c)。读作信用分配，那是一个掩码：它不改变任何词元优势的大小，只改变这个词元是否拿到一个优势。从证据看它管用，而它并不是对任何东西的估计。其机制大概是：策略确实有选择余地的那些位置，正是 $|A^{\pi_\theta}|$ 大的位置，所以熵是 $|\delta_t|$ 的一个廉价代理——但它是 幅度的代理，不携带任何关于符号的信息，而符号恰恰是信用分配本应提供的部分。

第二族是步骤级的偏好学习：与其估计一个步骤值，不如构造一些在步骤 m 之前一致、而在该步骤处分岔的前缀对，然后施加一个直接对齐损失 (Lai et al., 2024)。这继承了 第8章的离线特性与各种病理，最好被理解成一种数据构造技术，而不是一种信用分配技术。

核心思想: 让词元信用变得不均匀的三种办法，只有一种瞄准 A^{π_θ}

估计值——一个评论家、一个 PRM、一个蒙特卡洛前缀率——然后取增量；这瞄准的是 (13.1)，代价是估计误差。用策略的某个统计量（比如熵或对数比值的协方差）来重新加权；这是一个幅度上的启发式，不含符号信息。把词元掩进或掩出；这改变的是轨迹级的优势落在哪些位置上。这三者常常被当作可以互换的东西来讨论。它们不是：只有第一种才能告诉策略，在一条整体成功的轨迹里某个特定步骤是个错误，而这正是整个主题之所以存在所要提供的能力。

13.4 蒙特卡洛前缀值

现在我们来到本章中最干净的想法。节 2.3观察到，语言模型环境是确定性的、已知的，而且可重置的：一个前缀就是一个状态的完整描述，而我们只要以它为条件，就可以随时重新进入那个状态。没有什么能阻止我们用教科书里的办法去估计 $V^{\pi_\theta}(s_t)$——向前推演并取平均。

13.4.1 这个估计量

固定一个前缀 $s = x \oplus y_{<t}$。抽取 K 条延续 $y^{(1)}, \dots, y^{(K)} \sim \pi_\theta(\cdot \mid s)$，并令

$$\hat{V}_K(s) \;=\; \frac{1}{K} \sum_{k=1}^{K} r(x,\, s \oplus y^{(k)}). \tag{13.7}$$

命题 13.4 (无偏性与方差)**.** 对每个 $K \geq 1$ 都有 $\mathbb{E}[\hat{V}_K(s)] = V^{\pi_\theta}(s)$，并且 $\mathbb{V}\mathrm{ar}[\hat{V}_K(s)] = \sigma^2(s)/K$，其中 $\sigma^2(s) = \mathbb{V}\mathrm{ar}[r \mid s]$。对二值奖励，若 $p = V^{\pi_\theta}(s)$，则它是 $p(1-p)/K$。由于(13.7)是线性地进入优势的，下面这些代入式的优势估计量同样是无偏的。

无偏性是直接的——(13.7)是那个其期望定义了 $V^{\pi_\theta}(s)$ 的量的样本均值——而它有多不寻常值得停下来体会一下。一个评论家只要拟合不良就按构造带有偏差，而它拟合得多差，不做恰恰是这项计算就无从测量。这里没有回归、没有自举、没有函数逼近，也没有第二个网络。定理 13.4的最后一句也很要紧：本书中的许多估计量之所以有偏，是因为某个估计出现在了一个非线性运算（一个比值、一个标准差）的内部，而这一个没有。

把值变成优势有两种办法，区别在于是否用上已实现的结果：

$$\text{基线形式：} \quad \hat{A}_t = r - \hat{V}_K(s_t), \tag{13.8}$$

$$\text{增量形式：} \quad \hat{A}_t = \hat{V}_K(s_{t+1}) - \hat{V}_K(s_t). \tag{13.9}$$

(13.8)恰好是把 GAE$(1,1)$ 中的评论家换成一个蒙特卡洛估计（节 6.2.2），也正是 VinePPO 所实现的东西。它的方差是 $\sigma_t^2(1 + 1/K)$：无论 K 多大，已实现的 r 都贡献它的全部方差，所以这个估计量比一个基线打得好的 REINFORCE 至多也就好一倍。(13.9)丢掉了已实现的结果，方差约为 $2\sigma_t^2/K$，在 K 大时确实很小，但它也扔掉了那个一分钱没花的样本；而且如果出于成本考虑，s_t 与 s_{t+1} 处的采样集是共享的，那么这两个估计就是相关的，它们的差也就不再是它的公式所说的那个东西了。大多数实现是在一组较粗的切分点上使用 (13.8)，并让它在每个跨段内保持恒定。

算法 13.2 蒙特卡洛前缀值估计与优势组装，这是 VinePPO (Kazemnejad et al., 2025) 的核心。第 3 行是唯一用到环境可重置性的地方，也正是它让整个过程成为可能。优势在切分点之间的每个跨段内保持恒定，所以有效粒度是由 $|C|$ 决定的，而不是由分词器决定的。

输入： 提示 x、长度为 L 的采样轨迹 y、验证器 r_{ver}、策略 π_θ、每个状态的补全数 K、切分点 $C = \{0 = t_0 < t_1 < \cdots < t_M = L\}$

1: **估计前缀值**
2: **for** $j = 0$ **to** $M - 1$ **do**
3: $\quad s_j \leftarrow x \oplus y_{<t_j}$ ▷ 重置到这个前缀；复用它的 KV 缓存
4: $\quad$ 采样 $y^{(1)}, \ldots, y^{(K)} \sim \pi_\theta(\cdot \mid s_j)$
5: $\quad \hat{V}(t_j) \leftarrow \frac{1}{K} \sum_k r_{\text{ver}}(x, s_j \oplus y^{(k)})$
6: **end for**
7: $\hat{V}(t_M) \leftarrow r_{\text{ver}}(x, y)$ ▷ 已实现的结果；无需采样
8: **组装优势**
9: **for** $j = 0$ **to** $M - 1$ **do**
10: $\quad$ **for** $t = t_j$ **to** $t_{j+1} - 1$ **do**
11: $\quad\quad \hat{A}_t \leftarrow \hat{V}(t_{j+1}) - \hat{V}(t_j)$ ▷ 基线形式则为 $r - \hat{V}(t_j)$
12: $\quad$ **end for**
13: **end for**
14: **return** $(\hat{A}_1, \ldots, \hat{A}_L)$

13.4.2 它的代价，以及它能分辨出什么

代价是严峻的。要估计 M 个状态、每个用 K 条延续，并注意到从一个均匀放置的切分点出发的延续平均长度约为 $L/2$，那么每条轨迹额外生成的词元数约为 $MKL/2$，而轨迹本身只有 L：一个 $MK/2$ 的倍数。在 $M = 8$ 且 $K = 8$ 时那是 32× 的生成开销，还要叠加在已经在支付的 G 条采样轨迹之上。既然生成是挂钟时间的主导项（节 1.6），这就是塑造每一种实用变体的那条约束，也正是 节 13.6之所以要紧的原因。

K 小会带来一个容易被忽略的后果，而它解释了相当一部分经验记录。

推导：要看清一个步骤需要多少条延续？

取二值奖励，并假设我们想检测出某个步骤把成功概率改变了 Δ，i.e., $\delta_t = V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t) = \Delta$。使用相互独立的采样集时，增量估计 (13.9)的标准误是

$$\mathrm{se}(\hat{\delta}_t) = \sqrt{\frac{p_t(1-p_t) + p_{t+1}(1-p_{t+1})}{K}} \approx \sqrt{\frac{2p(1-p)}{K}}, \tag{13.10}$$

它在最坏情况 $p = 1/2$ 处是 $\sqrt{0.5/K}$。要求 $\Delta \geq 2\,\mathrm{se}$ 给出

$$K \gtrsim \frac{8p(1-p)}{\Delta^2} = \frac{2}{\Delta^2} \quad \text{在} p = \tfrac{1}{2} \text{ 处.} \tag{13.11}$$

对 $\Delta = 0.5$——一个把成功机会减半或加倍的步骤——$K = 8$ 就够了。对 $\Delta = 0.1$，则需要 $K \approx 200$。在任何人负担得起的 K 取值下，蒙特卡洛前缀值能识别出决定性的分岔，而基本上识别不出别的任何东西。

那个计算是关于单个状态的估计能告诉你什么的一个界，而不是关于训练能利用什么的界：估计噪声均值为零，会在各状态之间、在整批数据上被平均掉，所以总的梯度所受的改善，比逐状态误差所暗示的要小得多，噪声也小得多。但它确实预测了这个方法应当在哪里有帮助，即那些轨迹中包含少数几个高杠杆决策、而不是平滑累积许多小决策的任务。它也解释了为什么经验上的成功案例集中在 GSM8K 与 MATH 规模、链条较短的问题上。

13.4.3 VinePPO，以及切分点该放在哪里

Kazemnejad et al. (2025) 把这些零件组装了起来，而且重要的是，他们先测量了自己要替换掉的东西。他们对学出来的评论家的诊断是现有中最锐利的：在推理任务上，值网络对期望回报给出的估计很差，而当被要求对来自同一状态的若干备选下一步作排序时，它勉强胜过一个随机基线。那是一个比「评论家有噪声」强得多的陈述——它说的是评论家基本上没有任何轨迹内分辨率，而那恰恰是它被放在那里唯一要提供的东西。他们把它换成推理步骤粒度上的 (13.8)，报告说在 MATH 与 GSM8K 上以更少的挂钟时间胜过了 PPO 与其他基线，最多快到 3.0×，而且——更有意思的那项测量——在给定训练准确率下取得了更高的测试准确率，这是一个关于每样本泛化能力的主张，而不是关于优化速度的主张。

切分点放在哪里是主要的自由参数，而表 13.3列出了各种选项。

表 13.3: 沿一条轨迹把 M 个被估计状态放在哪里。底下的问题是 $|\delta_t|$ 在哪里大，因为上面的推导表明，低于约 $2\sqrt{2p(1-p)/K}$ 的增量反正是看不见的；把估计放在别处，就是在花生成去测量零。

放置方式	理由与行为	额外代价
按 t 均匀	不作任何假设;会把估计浪费在被迫的跨段上(抄写一个公式、把一个词写完)	MK
在步骤边界处	让单元与语义对齐，于是一个增量既可解释，又能复用为 PRM 标签	MK，M 依赖数据
在高熵词元处	瞄准节 12.6中的分叉词元，那里策略确实有选择余地	MK 外加一次熵计算
按估计方差自适应	把 K 花在 $\hat{p}$ 接近 $1/2$ 之处，因为伯努利方差与可信的 $\lvert\delta_t\rvert$ 都在那里最大	两阶段采样
二分查找第一个错误	找到最早那个 $\hat{p} \approx 0$ 的切分点就停 (Luo et al., 2024)	$K \log L$

工程实践: 蒙特卡洛前缀值：经得起实战的默认设置

- 只在其组不退化的提示上做估计（节 12.4)。如果全部 G 条采样轨迹都成功，那么每个前缀值都是 1、每个增量都是 0；你会为一串零花掉 MK 次生成。
- 用与主采样相同的引擎、温度与停止条件来采样延续。在不同温度下抽出的延续估计的是另一个策略的值，而由此产生的偏差是系统性的，不是噪声。
- 把 M 保持得小（4–16)、K 也保持得小（4–16)，而且一旦 $K \geq 8$，就优先增大 M 而不是 K：你用不上的分辨率比你用得上的覆盖度更糟。
- 把延续的长度上限压得远低于主生成的上限。一条延续是用来估计一个概率的，不是用来产出一件成品的，而把它截断在比如 $L/4$ 处，会让 $\hat{p}$ 向下偏一个大致恒定的因子，这个因子在取增量时会部分抵消。
- 记录 $\hat{\delta}$ 的分布。如果它集中在零的一个标准误之内，那你什么也没买到，应该把算力还给 G。

历史注记: 藤

这个构造由来已久。Schulman et al. (2015) 区分了 TRPO 中估计优势的两种采样方案：单路径，每个起始状态用一条轨迹；以及藤，先推演出一条轨迹，在它上

> 面挑一批状态，再从每个状态采样若干动作——一架挂在主干上的短枝格架。那篇论文明确指出，藤给出的优势估计方差要低得多，而它要求系统能被恢复到任意状态，这就把它限制在了模拟器里。因此机器人学与连续控制基本上放弃了它。语言模型正是藤当年为之设计、却无法拥有的那个场景：状态是一个字符串，而恢复它是免费的。VinePPO 的名字是对这一脉络的刻意致意，而这段插曲很好地说明了一个想法可以因为某些后来不再成立的理由而被搁置十年。

代码 13.2: 算法 13.2在实践中所采取的蒙特卡洛前缀值形式。有两个细节承载着正确性：`sample_fn` 必须使用训练时的采样分布，而最后一个值是已实现的结果而不是一个估计，正是这一点让最后一个跨段的优势是精确的。

```python
def mc_prefix_advantages(prompt, toks, cuts, sample_fn, verify, K=8, mode="increment"):
    """cuts: sorted token indices, cuts[0] == 0, cuts[-1] == len(toks).
    sample_fn(prefix, n) -> n continuations from the *current* policy.
    Cost: (len(cuts) - 1) * K extra generations for this trajectory."""
    v = {}
    for t in cuts[:-1]:
        comps = sample_fn(prompt + toks[:t], K)             # 前缀 KV 缓存被复用
        v[t] = sum(verify(prompt, toks[:t] + c) for c in comps) / K
    v[cuts[-1]] = float(verify(prompt, toks))               # 已实现的，而非估计的

    adv = torch.zeros(len(toks))
    for a, b in zip(cuts[:-1], cuts[1:]):
        step = v[b] - v[a] if mode == "increment" else v[cuts[-1]] - v[a]
        adv[a:b] = step                                     # 在跨段内保持恒定
    return adv
```

13.5 片段级与轮次级的信用

算法 13.2已经揭示出，估计量的粒度是由切分点决定的，与词元毫无关系。把这一点挑明，就得到了片段级这一族方法。

SPO (Guo et al., 2025b) 是对它的系统性处理。它的表述正是本章一直在铺垫的那个：词元级方法追求细粒度信号却拿不到一个准确的评论家，而轨迹级方法拿到的信号准确却粗糙，有用的位置在两者之间，在那里蒙特卡洛估计之所以负担得起，正是因为估计点很少。它有三个部件：一种灵活的片段划分、一种不用评论家的蒙特卡洛片段优势估计，以及一种控制片段内哪些词元领取该片段优势的概率掩码策略。他们把它实例化了两次——面向短思维链的、基于切分点的划分配链式估计，以及面向长思

维链的基于树的估计——并报告说，前者在 GSM8K 上比 PPO 与 GRPO 高出 6–12 个百分点，后者在 2k 与 4k 上下文预算下于 MATH500 上比 GRPO 高出 7–11 个百分点。按 2026 年的标准，那些上下文预算都很短，所以请把这个结果读作关于中短轨迹的证据。

在这个具体方法背后，有一条值得单独写出来的原则。

核心思想: 在作出决策的那个单元上分配信用

一个位置只有在策略在那里确实有选择余地时——只有在结果分布确实取决于那个选择时——才配拥有它自己的优势。一份正确推导中的大多数词元位置都是被迫的：一个词的剩余部分、抄写一个表达式、闭合一个括号。这与熵那一支文献从另一侧测量到的是同一个事实（节 12.6），而它意味着词元粒度在几乎所有地方都过度分辨了，在那里做估计的成本是浪费的。对数学推理而言，决策单元大致是一个步骤；对智能体而言是一个轮次，因为轮次是模型作出承诺、而环境作出回应的地方。因此，选择粒度不是一次超参数扫描，而是一项关于任务的建模主张。

轮次级的信用分配是智能体式的情形，归第18章管，所以这里两句话就够。第一，它是唯一一处奖励往往确实稠密的地方：一个在每次动作之后返回测试结果、检索命中或编译错误的环境，提供的是关于中间状态的真实信息，而不是对它的一个学出来的猜测，于是节 13.2.3的那些反对意见并不适用。第二，天然的估计量是分层的而不是扁平的——回合层面一个组、步骤层面一个组，如Feng et al. (2025b) 那样——因为同一回合内的各轮次共享一个前缀，因而构成了聚簇，而这正是节 13.6接下来要分析的结构。

13.6 树状采样与共享前缀

节 13.4的成本核算假定值估计是与采样轨迹分开购买的。其实不必如此。如果我们采样的不是 G 条相互独立的链，而是一棵树——在选定的深度上分叉——那么每个内部节点都自带一个由它下方叶子算出来的值估计，不额外花一分生成开销，而且共享的前缀只被生成一次而不是 G 次。

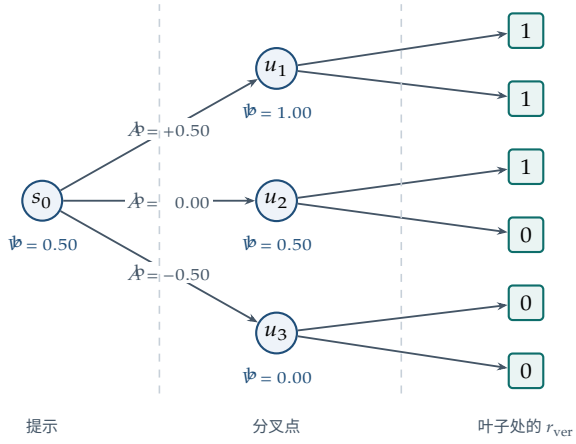


图 13.2: 一棵有六个叶子、一层分叉的采样树。只有叶子由验证器打分；每个内部节点的值是它下方叶子的均值，所以那三条边上的优势不需要额外生成任何东西就能得到。共享主干的词元只被生成一次，因此这棵树所花的词元数大约是六条独立链的 58%（见正文），而作为交换，这六个叶子不再是相互独立的样本——用于轨迹级基线的有效组规模小于六。

13.6.1 算力上的论证

考虑一棵平衡树，在 d 个等间隔的深度上每处的分支因子都是 m，给出 $G = m^d$ 个叶子、总长度为 L，于是每一层每个节点贡献 L/d 个词元。词元计数是

$$\frac{L}{d}\sum_{j=1}^{d} m^j \;=\; \frac{L}{d}\cdot\frac{m(m^d-1)}{m-1} \qquad \text{对比} \qquad GL = m^d L \tag{13.12}$$

后者是独立链的情形，比值大致是 $m/(d(m-1))$。对 $m=2$、$d=3$、$G=8$，树需要 $\frac{14}{3}L \approx 4.67L$，而独立链需要 $8L$：节省 42%，同时还额外提供了 14 个独立链所没有的内部值估计。这份节省在 FLOPs 上是真实的，在挂钟时间上还要更大，因为前缀共享意味着主干的 KV 缓存只需算一次，各分支从它出发解码。

这是通往逐步骤信用的算力高效路线，也是为什么树状这一支自 2025 年以来一直是这个主题里最活跃的部分。TreeRL (Hou et al., 2025) 在 RL 循环内部跑在线策略的树搜索，从树本身而不是从一个单独训练的 PRM 获取中间监督——明确就是为了避开一个独立模型带来的分布失配与作弊——并且从高不确定性的中间步骤而非随机位置分叉，这正是把 表 13.3的放置规则用在了分叉点上。TreePO (Li et al., 2025h) 与 SPO-tree (Guo et al., 2025b) 作了相关的取舍，而 MCTS 标注那一支 (Guan et al., 2025; Wu et al., 2025b) 则是同一个构造，只是每条提示上跑了更多搜索。

13.6.2 树所引入的偏差

没有什么是免费的。一棵采样树的每个叶子在边缘分布上仍然服从 $\pi_\theta(\cdot\mid x)$，所以叶子均值仍然是 $V^{\pi_\theta}(x)$ 的一个无偏估计。变的是它的方差，而这个变化就是经典的聚簇抽样设计效应：对于 c 个簇、每簇 n 个叶子、簇内相关为 ϱ 的情形，

$$\mathbb{V}\mathrm{ar}[\bar{r}] = \frac{\sigma^2}{cn}\big(1 + (n-1)\varrho\big), \qquad \text{于是} \qquad G_{\text{eff}} = \frac{G}{1 + (n-1)\varrho}. \tag{13.13}$$

在推理任务中，一个前缀对结果有很强的预测力——这正是整章的前提——所以 ϱ 很大，而(13.13)说明，一棵有 $G = 8$ 个叶子、分成 2 个各 4 叶簇的树，其有效组规模可能更接近 2 或 3。这棵树是用轨迹级的分辨率买来了逐步骤的分辨率。

常见陷阱: 采样树会缩小你的有效组规模和覆盖度

这是两项不同的代价，常常被混为一谈。统计上的那项是 (13.13)：组基线比它名义上的 G 所暗示的更有噪声，而且如果你还要过滤退化组（节 12.4），你会滤掉更多，因为共享同一主干的叶子更容易彼此一致。行为上的那项更糟，也更不易察觉：在相同的词元预算下，一棵树探索到的不同开局更少，而首步的多样性恰恰是组相对估计量所需要的、也正是熵坍缩所摧毁的东西。一次在匹配词元开销下从链切换到树的训练，可以同时表现出信用分配的改善和 pass@k 的退化。请把两者都报告出来，并把固定深度处不同前缀的数目作为一个诊断量报告。

系统方面的要求并不简单，这也是为什么这个想法要到 2025 年才变得实用：推理引擎必须支持从一份共享的 KV 缓存分叉出一条序列，调度器必须处理一棵参差不齐的树而不是一批等长请求，而训练器必须消化一个树形的批次，其中词元被多个样本共享，却不能在损失归一化器中被重复计数（节 12.2）。第21章讲这些机制。

13.7 更细的信用真的有用吗?

截至 2026 年年中，诚实的回答是：比节 13.1所许诺的要少得多，而且其模式到今天已经清晰可辨。

祛魅式的证据。 定义了这个领域的那两项结果，都考虑过过程监督，然后都拒绝了它。DeepSeek-R1 把 PRM 与 MCTS 报告为不成功的尝试 (DeepSeek-AI, 2025)；Kimi k1.5 出于原则回避了值函数、PRM 与搜索，并且进一步论证说评论家会压制对那些不寻常却成功的推理路径的探索 (Kimi Team, 2025a)。第12章那一整套 GRPO 变体文献，其

收益来自归一化器、截断的不对称性与退化组的记账，而表 12.5中优势粒度那根轴基本上没被碰过。而且Zhang et al. (2025e) 报告说，最便宜的自动步骤标签——蒙特卡洛补全率——泛化性比人工标签或评判模型标签更差，这在夺走 PRM 可靠性论证的同时，也夺走了它的可规模化论证。

正面的证据，以及它的形状。 VinePPO、SPO、PRIME、PAV 与 TreeRL 都报告了相对仅用结果的基线的收益 (Kazemnejad et al., 2025; Guo et al., 2025b; Cui et al., 2025a; Setlur et al., 2025a; Hou et al., 2025)。作为一个整体来读，有四个特征很突出。基准是 GSM8K、MATH 与 MATH500，上下文预算为 2k–8k，这属于短思维链场景。模型在 1.5B–9B 区间，而且通常来自单一模型族，所以节 11.8适用。基线常常是带评论家的 PPO，或者是按发表原样的 GRPO，而不是打上了节 11.7那些补丁的 GRPO。而且这些比较都是单预算的终点比较，恰恰是节 12.12所表明的那种无法区分算力效率变化与渐近值变化的比较。这些都不使那些结果变成错的；它们只是使那些结果对前沿长思维链训练而言成为弱证据，而后者才是读者多半真正关心的场景。

核心思想：为什么仅用结果奖励的表现好过理论所暗示的

有四种机制，按把握程度递减排列。其一：组基线已经去掉了主导性的方差项，而它的代价你本来就为采样轨迹付过了，表 6.1展示了这笔交易有多划算。其二：节 13.1那个 T 倍的奖赏要求精确的值，而节 13.4的分辨率计算表明，负担得起的估计只能看见成功概率上大约 0.2–0.5 以上的增量，所以那份奖赏的大部分根本够不着。其三：任何值估计中的误差都在每个词元上充当偏差，而偏差不会被平均掉，反观均匀估计量的错误归因是均值为零的噪声，它会。其四，更具推测性：正确的推理是冗余的。一条成功轨迹中那个决定性的洞见，通常会在许多地方被复述、被使用、被检验，所以提高整条轨迹的概率，就等于把那个洞见的概率提高了好几遍，因而均匀估计量并不像它的逐词元误差所暗示的那样浪费。

更细的信用在哪里确实划算。 表 13.4收集了这些情形，并为每一种明确标出证据层级，因为它们在确立程度上差别很大。

第五行值得多说一句，因为它是理论上最干净、检验上最少的一个案例。考虑两条成功的轨迹：一条直截了当地推进，另一条在第 200 个词元处走错，在第 600 个词元处察觉，然后挽回。结果奖励给两者都是 $r = 1$，因此对第二条中那个错误转向的强化，与对第一条中那个正确开局的强化一样多。一个前缀值估计量会给那个错误转向

表 13.4: 更细粒度的信用分配值回其代价的那些场景。证据一列遵循本书的惯例：已复现指有若干个独立的研究组，单篇指只有一个，而仍有争议指已发表的结果彼此不一致，或者该主张目前还只是一个论证而不是一次测量。

场景	使它划算的机制	证据
推理时的重排序与搜索	一个步骤分数只需要给候选排序，而它的错误只代价一次查询，不会让一个策略变形	已复现
带有真实逐轮次反馈的多轮次智能体任务	中间信号来自环境而不是来自一个学出来的模型，所以既没有奖励模型可以作弊，也没有标签会有噪声	在智能体强化学习中已复现（第18章）
部分进展有良好定义的任务	形式化证明的子目标、测试集通过比例、清单条目：这种分解是规范的一部分，而不是对规范的猜测	按领域从单篇到已复现不等
极长时域（$T \gtrsim 10^5$，每回合数小时）	每回合一个标量，就单位算力所携带的信息量而言难以为继，而那个 T 倍因子是决定性的	仍有争议；基本上还只是一个论证
学会放弃一个糟糕的前缀	结果奖励无法区分「从未出错」与「出错后挽回」，因为两条轨迹都拿到 $r = 1$；而前缀值可以	仍有争议
短思维链数学	步骤少、决定性强、可逐条核查，所以增量大到在小 K 下也能分辨	单篇，但有若干论文彼此印证

一个负增量、给那次挽回一个正增量，这才是正确的分解，也是本书中唯一能够教会策略「某个轨迹中途的决策是糟糕的，尽管这一回合最终成功了」的机制。这件事是否重要到值得为之付费还是开放的；它是我最希望看到被认真检验的那个论点，而且应该在挽回很常见的任务上检验，而不是在挽回很罕见的 GSM8K 上。

13.8 信用分配：一个推理时还是训练时的选择

同一个学出来的对象——一个给前缀打分的函数——可以花在两个地方。在推理时它引导搜索：剪掉低分分支、把样本分配给有希望的前缀、对已完成的候选重排序（第10章）。在训练时它塑造优势。这种对偶在这个意义上是精确的：两种用法消耗的是同一个 v_ϕ，而在它们之间作选择应当依据成本与风险，而不是依据口味。

成本。 训练时的用法只付一次，然后摊销到这个模型此后要服务的每一次查询上；推理时的用法则要在每次查询上永远地付。对一个被服务 N 次的模型，一个 PRM 的推理成本随 N 缩放，而训练成本是一个常数，所以对任何被广泛部署的模型，训练时的用法在算术上都以很大优势胜出——只要它管用。这与把能力从提示搬进权重的那个论证是同一个（节 1.1），也正是这个领域不断尝试它的原因。

风险。 这两者的风险严重不对称，而这才是决定实践的东西。一个糟糕的 v_ϕ 用在推理时，代价只是在它出错的那些查询上损失准确率，仅此而已：策略没有变，失败在留出集上可测，而且你当天下午就能把 v_ϕ 换成一个更好的。一个糟糕的 v_ϕ 用作奖励，则会永久性地让策略变形，而且变形的方向是由优化器对抗性地挑出来的，损害通常要到一千步之后、当某个 v_ϕ 从来没在测量的指标崩掉时才被发现。奖励作弊只在第二种用法中才是一个选项。

第三个选项，通常是最好的那个。 用 v_ϕ 来筛选数据，而不是用它来塑造梯度：剪枝采样树、把得分最高的正确轨迹留给一轮拒绝采样 SFT（节 11.4），或者丢掉那些被 PRM 标记为「答案对但过程不连贯」的轨迹。这捕获了大部分收益——策略是在更好的轨迹上被训练的——而其失效模式是一个过滤器的失效模式，不是一个奖励的失效模式。一个错了 10% 的过滤器会扔掉 10% 的好数据；一个错了 10% 的奖励则会被优化到恰恰那 10% 里去。

工程实践：把过程信号编排进流水线

1. 造出那个步骤打分器，并把它当作一个打分器来评估：最早错误的定位（Zheng et al., 2025b; Song et al., 2025c），而不是 best-of-n，理由见节 13.2.2。
2. 把它部署到推理时，并对照一个算力匹配的基线（更多样本加多数投票）测量增益。如果更多样本赢了，就到此为止；你学到了一件有用而便宜的事。
3. 用它来过滤或排序训练数据，然后再测一次。
4. 只有到这时才把它放进奖励里，而当你这么做时：保留一个大一个数量级的结果项，像 PRIME 那样在策略自己的采样轨迹上在线更新这个打分器（Cui et al., 2025a），每隔几百步用人工审计一次采样轨迹（节 5.8.4），并留出一个打分器看不到的指标。

5. 在表面质量相当的情况下，优先选择根本不需要学出来的模型的信号——蒙特卡洛前缀值、环境反馈、树的统计量——而不是一个学出来的信号。它不会被作弊，最多只会很贵。

本章小结

- 在一个带终局奖励的确定性 MDP 中，真实优势就是前缀值的增量，$A^{\pi_\theta}(s_t, a_t) = V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$，而这些增量裂项相消为 $r - V^{\pi_\theta}(x)$。因此信用分配是对一个固定总量的重新分配，而且它完全归结为估计前缀上的一个标量函数。
- 由于 $V^{\pi_\theta}(s_t)$ 是一个鞅，精确的逐词元优势是互不相关的增量，所以完美的信用分配消除了梯度二阶矩随长度的线性增长——一个 T 倍因子。那就是全部的奖赏，而它只有在值精确时才能拿到。
- 每一个实用的估计量都把那份方差的一部分转化为偏差，而偏差施加在每一个词元上，且不会在一个批次上被平均掉。正是这种不对称，而不是任何经验上的偶然，才是最粗糙的估计量得以守住阵地的原因。
- 过程奖励模型作为重排序器效果很好——自Lightman et al. (2024) 以来已被复现，而Uesato et al. (2022) 表明其增益具体体现在推理质量而非最终答案准确率上——而作为训练时奖励，其有用性的说服力则弱得多。三条已被点名的理由是：步骤边界的歧义、标签测量的是补全策略的挽回能力而不是该步骤的正确性，以及 PRM 是一个每条轨迹被查询 M 次的、学出来的奖励模型。
- 一个被参数化为 $\beta \log(\pi_\phi / \pi_{\text{ref}})$ 的结果奖励模型自动就是一个过程模型，因为在一个词元处 $\beta \log(\pi / \pi_{\text{ref}})$ 就是软优势。这使得稠密的过程奖励可以仅从结果标签中获得，而在线更新这个隐式模型正是让它不被作弊的关键。
- 蒙特卡洛前缀值是现有唯一无偏的逐状态估计量，而它们之所以存在，只是因为语言模型环境是可重置的——TRPO 的那根「藤」，终于来到了一个允许它的场景。它们的代价是轨迹生成量的 $MK/2$ 倍，而在负担得起的 K 下，只能分辨成功概率上大约 0.2–0.5 以上的增量。
- 树状采样免费给出内部节点的值估计，并且通过前缀共享在 $G = 8$ 时节省约 40% 的词元，但叶子是聚簇的：有效组规模按设计效应因子 $1 + (n-1)\varrho$ 下降，而不同开局的多样性也随之下降。

- 在短思维链数学上，若干方法报告了更细信用带来的真实收益；这些比较大多是单一模型族、单一预算、短上下文，而且对照的是 2025 年之前的基线。更细的信用在推理时、以及在具有真实环境反馈的多轮次场景中最为确立；它最清楚而尚未被利用的案例，是教会策略在一个成功的回合内部放弃一个糟糕的前缀。
- 给定一个步骤打分器，优先用于推理时搜索，其次是数据过滤，最后才是奖励塑形。成本这一侧偏向奖励塑形，风险这一侧偏向推理时，而风险是更大的那一项。

延伸阅读

先读Kazemnejad et al. (2025)，而且要为它的诊断而读，不是为它的方法：「一个值网络在对来自同一状态的备选步骤作排序时勉强胜过随机基线」这项测量，是这一文献中最有用的单一事实，因为它告诉你 PPO 赖以做信用分配的那个对象根本没在干活，也用一句话解释了为什么这个领域可以丢掉评论家而不损失任何东西。接着读Guo et al. (2025b)，它把这个估计量推广到任意粒度，并且把词元与轨迹之间的权衡表述得干净到足以充当这个主题的定义。

关于过程奖励模型，Lightman et al. (2024) 与Uesato et al. (2022) 应当当作一对、并按这个顺序来读。Lightman et al. 给出了那个很强的重排序结果；Uesato et al. 给出了受控比较与更审慎的结论，即好处体现在推理质量而非最终答案准确率上。然后读Wang et al. (2024f) 了解自动标注这个想法，再读Zhang et al. (2025e) 看对它的审计，那是这一领域里最诚实的一篇论文：它同时确立了蒙特卡洛步骤标签表现不佳，以及大家一直用来挑选 PRM 的那套评估测量的是错的东西。Setlur et al. (2025a) 是理论上的纠偏，他们论证一个过程奖励应当测量的是进展——最终成功概率的变化，且在一个与基座策略不同的证明者策略下测量——而那恰恰就是定理 13.2中的 δ_t，从而把整桩 PRM 事业重新表述为多绕了几道弯的优势估计。

Yuan et al. (2025a) 简短、出人意料，也是「步骤标签从来都不是必需的」这一论点的最佳单篇论证；如果你打算把东西造出来，接下来该读的是Cui et al. (2025a)。在树这一侧，Hou et al. (2025) 是对「为什么在线策略的树统计量优于一个单独训练的 PRM」最干净的表述，而Guan et al. (2025) 展示了当算力被慷慨地花在搜索上时，MCTS 标注这条路能推到多远。

最后，读DeepSeek-AI (2025) 中关于不成功尝试的那一节。它只有半页，是这个领域里被读得最多的那篇论文中被跳过得最多的部分，而它对长思维链 RL 下过程监督之现状的总结，比大多数专门讨论这个主题的论文都更准确。

习题

13.1（**裂项相消恒等式，以及它禁止了什么。**）直接验证 (13.2)。然后考虑这样一个提议：用 $\hat{A}_t = \lambda(V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t))$ 且 $\lambda > 1$ 来「增强学习信号」。精确说明这对梯度的期望和它的方差各做了什么，并解释为什么这不是一次信用分配上的改进，而是一次学习率的改变。然后找出一种对同一总量的重新分配，它确实是一次改进；再找出一种严格劣于均匀分配的。

13.2（**鞅界，以及它在哪里失效。**）重现那个表明 $\sum_t \mathbb{E}[\delta_t^2] = \mathbb{V}\mathrm{ar}[r]$ 的推导。现在丢掉逐词元得分函数相互正交的假设，改为对所有 t, t' 令 $\mathbb{E}[g_t g_{t'}^\top] = \Sigma$（得分函数完全相关）。重新计算两个二阶矩，并判定那个 T 倍优势是否还在。用这个结果说明，是模型的——而不是估计量的——哪一条性质，控制着更细的信用最多能值多少。

13.3（**聚合函数，习题详演。**）取一份 $M = 12$ 步的解答，除 $v_7 = 0.30$ 之外所有 m 的 PRM 分数都是 $v_m = 0.95$。计算表 13.2中的全部四种聚合。现在追加四个 $v = 0.99$ 的平凡正确步骤，再算一遍。哪些聚合朝错误的方向变了，变了多少？最后，构造一组步骤分数，使得 `min` 与 `product` 对两份解答给出相反的排序，并说明你会想要哪一种排序。

13.4（**补全率标签测量的是错的东西。**）一个步骤 σ_m 在数学上是正确的，却导向了一次补全策略处理得很糟的分情况讨论，所以它的补全率是 0.1；第二个步骤含有一处补全策略能可靠地察觉并修正的算术失误，所以它的补全率是 0.8。写下 算法 13.1分配给它们各自的硬标签与软标签。然后说明软标签是对哪个量的无偏估计，并解释为什么Setlur et al. (2025a) 的处方——在一个刻意选定的证明者策略下测量进展——是一次修复而不是一次添乱。

13.5（**为蒙特卡洛前缀值做预算。**）你每条提示有 B 个生成词元的固定预算，平均回复长度为 L。写出 G 条采样轨迹外加 M 个被估计状态、每个 K 条延续的预算约束，假定延续平均 $L/2$ 个词元。利用分辨率要求 $K \geq 8p(1-p)/\Delta^2$ 与组基线方差 $O(1/G)$，把 B 在 (G, M, K) 上的分配表述成一个优化问题，在 $\Delta = 0.3$ 与 $p = 1/2$ 下求解，并把所得的 G 与节 11.7 推荐的 8–16 作比较。

13.6（**一棵树的设计效应。**）验证(13.12)以及 $m = 2$、$d = 3$ 下的 42% 这个数字。然后，把一个叶子的结果建模为 $r = \mathbb{1}[\eta_{\text{trunk}} + \eta_{\text{leaf}} > 0]$，其中两个独立的高斯分量方差分别为 ϱ 与 $1-\varrho$，对 $\varrho \in \{0.2, 0.5, 0.8\}$ 由 (13.13)计算 G_{eff}。如果目标是组均值基线的方差，那么在哪个 ϱ 处词元上的节省不足以补偿有效组规模的损失？

13.7 ⋆ **（存在一个胜过这两种形式的无偏估计量吗?）**(13.8)保留了 r 的全部方差，因而浪费了那 K 条延续；(13.9)则浪费了已实现的那条轨迹。把 r、$\hat{\mathbf{v}}_K(s_t)$ 与 $\hat{\mathbf{v}}_K(s_{t+1})$ 的最小方差无偏组合构造为 $A^{\pi_\theta}(s_t, a_t)$ 的一个估计量，把这三者当作联合分布的、且带有由 s_t 与 s_{t+1} 之间共享采样所蕴含的相关性。然后判定最优权重是否依赖于未知量，如果依赖，就提出一个代入式版本并给它的偏差定界。说明这个答案对那些在相邻切分点上复用同一批延续的实现意味着什么。

13.8 ⋆ **（能免费给一个组基线加上轨迹内分辨率吗?）**来自同一条提示的 G 条采样轨迹共享这条提示，而且通常会碰巧共享一段不平凡的公共前缀。设计一个估计量，它从一个普通（非树状）组的观测到的一致性结构中提取前缀值信息——例如按最长公共前缀把采样轨迹分组——并推导它的偏差与方差。判定它能否被做成无偏的，并指出策略分布的哪一条性质决定了它能否恢复出有用的信号。然后解释为什么这并不只是节 13.6中那棵树换了个马甲。

13.9 † **（把奖赏测量出来。）**取一个 1.5B 模型和一个带经审计核查器的数学数据集。首先，把目标测量出来：对 200 条轨迹，在 16 个均匀分布的前缀处用 $K = 64$ 条延续估计 V^{π_θ}，并画出 $\hat{\delta}_t$ 的经验分布。报告有多大比例的增量超过两个标准误，从而报告信用分配信号中原则上有多大比例是够得着的。其次，在匹配的词元预算下训练四种配置：用节 11.7默认设置的仅结果版本；$(M, K) = (8, 8)$ 的 算法 13.2；同样但取 $(M, K) = (16, 4)$；以及一棵 $m = 2$、$d = 3$ 的采样树。报告 pass@1、pass@16、熵、每条提示上深度为 $L/3$ 的不同前缀数目，以及(12.14)拟合出的 (A, B)。在跑第二部分之前，先根据你第一部分的测量预测排序，并报告你猜对了没有。

第 14 章

强化学习究竟教会了什么?

核心思想: 锐化是默认结果，扩张则是有条件的

对于静态的单轮推理任务，一次典型的 RLVR 运行所做的事情，大部分是把概率质量重新分配到基座模型本来就偶尔会采样到的轨迹上：pass@1 上升，大 k 处的 pass@k 往往下降，而六种算法之间的差距，与它们距离基座模型自身的 pass@k 上界的距离相比，是很小的。但这并不是故事的全部。异常长的训练、需要与环境进行组合式交互的任务，以及那些正确轨迹虽稀有却确实存在的提示，表现都不一样——有时甚至方向相反。本章的主张是：2025 年那场关于「RL 只会锐化」与「RL 会扩张」的争论，其实是一场关于区制的争论，而双方在测量时都没有控制这些区制；对实践者而言有用的问题不是哪一派正确，而是自己手上的提示处在哪一个区制里。

本章与前两章在性质上不同。Chapters 11 and 12介绍了算法，并追问如何让它们奏效。这里我们追问它们成就了什么，而诚实的回答是：仍有争议。因此本章的材料是按证据综述而不是按方法推导来组织的：每项主张都注明出处，证据层级都写明，文献自相矛盾之处则原样留在明面上，而不被抹平成一段顺畅的叙事。

三条线索在此汇合。节 1.7把「激发还是习得」列为全书第一条反复出现的主题；节 1.8与定理 6.3给出了预期为「激发」的形式化理由；节 11.8报告了一个训练样本、甚至一个随机奖励，就能产生所报告增益的大部分。节 14.7以及表 12.1中的 P9 都被推迟到了本章，两者在下文都会得到回答。

14.1 把问题说精确

「RL 到底教会了什么新东西吗?」这句口号在我们说清楚新是什么意思之前，还算不上一个科学问题。有两个定义承担了大部分工作。

定义 14.1 (激发与习得)**.** 设 π_{base} 是后训练所出发的模型，π_θ 是后训练之后的策略。若一次变化是在 π_{base} 的支撑集内部重新分配概率质量，则称之为激发：对每个满足 $\pi_\theta(y\mid x) > 0$ 的回复 y，都有 $\pi_{\text{base}}(y\mid x) > 0$，并且改进来自于抬高那些本就以不可忽略的概率被采样到的回复上的质量。若后训练得到的策略把可观的质量放在了 π_{base} 基本上未赋予任何质量的回复上——i.e., 放在任何现实的采样预算都不可能从 π_{base} 中找到的质量上——则称之为习得。

这一区分之所以重要，是因为两者的天花板不同。激发的上界是基座模型的一条性质，所以它的收益必定会饱和，而且其增益原则上可以改用测试时搜索来获得（第10章）。习得则没有这样的界；如果 RL 真的能做到习得，那么 RL 算力就是预训练算力的替代品，而不是它的放大器。这是战略上的差别，不是哲学上的差别。

14.1.1 理论究竟说了什么

定理 6.3给出了 KL 正则化目标的不动点 $\pi^\star \propto \pi_{\text{ref}}\exp(r/\beta)$，而节 6.4已经指出了它的第三条推论：只要 $\pi_{\text{ref}}(y\mid x) = 0$，就有 $\pi^\star(y\mid x) = 0$。值得把这一点加强为一个定量的陈述，因为定量版本远不如定性版本那样令人印象深刻。

推论 14.2 (有界的放大)**.** 设 r 取值于宽度为 $\Delta = \max_y r - \min_y r$ 的区间内，且 $\beta > 0$。那么对每个 y 都有

$$\frac{\pi^\star(y\mid x)}{\pi_{\text{ref}}(y\mid x)} = \frac{e^{r(x,y)/\beta}}{Z(x)} \le \exp\Big(\frac{\Delta}{\beta}\Big), \tag{14.1}$$

因为 $Z(x) = \mathbb{E}_{\pi_{\text{ref}}}[e^{r/\beta}] \ge e^{\min_y r/\beta}$。特别地，$\pi^\star$ 关于 π_{ref} 绝对连续：恰好为零之处会被保留，而非零之处至多被放大 $e^{\Delta/\beta}$ 倍。

应当把(14.1)读作一张许可证，而不是一条约束。在二元奖励下 $\Delta = 1$；节 11.7中的 RLVR 配方使用 $\beta \in \{0, 10^{-3}, 10^{-2}\}$。在 $\beta = 10^{-2}$ 时，这个界允许的放大倍数是 $e^{100} \approx 10^{43}$。一条基座模型每 10^{20} 次采样才会吐出一次的轨迹——i.e., 实际上永远不会出现——因此完全可以在不违背任何定理的前提下，成为 KL 正则化最优解的主导模式。支撑集保持禁止的是从恰好为零中造出行为，而它对从「可忽略」中造出行为几乎什么也没说。既然定理 14.1是以可忽略性而不是以恰好为零来写的，那么这条定理并没有解决人们通常援引它去解决的那个问题。

另有三处间隙，把(14.1)与训练的实际所为隔开。

1. β 经常就是零。 节 12.8记录了 DAPO、CISPO 与 Magistral 都取 $\beta = 0$。没有 KL 项就没有 Gibbs 不动点，定理 14.2也就成了空话；剩下的只有比值截断与小学习率所构成的隐式信赖域，它约束的是策略的每一步移动，因而对 10^3 步的总漂移什么都约束不了。

2. 有限精度意味着不存在恰好为零。 在 10^5 个词元上做 softmax 会给每个词元赋予严格为正的概率，所以任何语言模型的支撑集都是整个 $\mathcal{V}^*$，支撑集保持这一陈述在形式上说的是空集。因此真正起作用的概念是经验支撑集或有效支撑集：给定采样预算下可以指望产生出来的那些回复。

定义 14.3 (预算 n 下的有效支撑集)**.** $\mathcal{S}_n(x;\pi) = \{y : \pi(y\,|\,x) \geq 1/n\}$，即在 n 个 i.i.d. 样本中期望出现次数至少为一的那些回复。所有关于「基座模型的边界」的陈述，都是关于实验者所能负担的那个 n 对应的 $\mathcal{S}_n$ 的陈述，因而都继承了对 n 的依赖。这正是为什么整场辩论都是用大家所能付得起的最大 k 处的 pass@k 曲线来进行的。

3. 训练走的是一条路径，而不是一个不动点。 定理 6.3 描述的是分布空间中的一个最优解。训练则在参数空间中执行成千上万步梯度更新，其中有两个特征会让不动点分析在「更多扩张」而非「更少扩张」的方向上变得松弛。参数共享意味着在一个提示上算出的更新，会改变那些从未被采样到的前缀处的条件分布，因此质量可以出现在奖励从未触及的状态上——这正是让单样本 RLVR 得以泛化的同一机制 (Wang et al., 2025e)。而且信赖域所参照的基准是上一步的策略，而不是基座模型，所以一串各自都落在小 KL 球内的更新，可以复合出任意大的总位移。ProRL 的周期性参考策略重置（节 12.8）正是对这一事实的刻意利用。

把三处间隙合起来看，就意味着定理 6.3不是本章问题的答案，而是答案必须诉诸经验的原因。支撑集保持经常被援引，仿佛它已经解决了激发问题。它并没有：它约束的是恰好为零之处，而在一个有限精度的 softmax 上训练数千步、且 β 常常等于零的情形下，恰好为零并不是起约束作用的那个条件。这条定理真正确立的是 默认预期——一个以 KL 为锚的优化器要提高奖励，最便宜的路线就是给参考模型本来就会产出的东西重新加权——因而也确立了任何「习得」主张所须承担的举证责任。

表 14.1: 用于区分激发与习得的可测代理指标，以及各自携带的混杂因素。没有哪一行是决定性的：悲观派与乐观派的文献在第 1–2 行的测量上大体一致，分歧在于这些测量是在哪种区制下做出的。要在任一方向上提出主张，同时报告第 1、3、4、6 行是最低要求。

代理指标	它测的是什么	主要混杂因素
pass@k 曲线，k 直到 10^2–10^3	在预算 k 下是否仍有某条正确轨迹可达	基座模型的温度与提示格式；无偏估计量(2.23)要求 $n \geq k$
交叉点 $k^\star$	基座模型反超 RL 模型所需的预算	取决于基准的难度构成；在不同评测套件之间不可比
覆盖率	在 n 次采样中至少被解出一次的题目比例	在可猜出答案的题上侥幸命中一次也算数（验证器不可靠）
答案级熵	策略仍能产出多少个不同的最终答案	需要对答案做规范化；对与奖励相同的那些校验器缺陷同样敏感
词元级熵	每个解码步上的局部不确定性	可能与答案级熵朝相反方向移动 (Wu et al., 2025a)
RL 输出上的 $-\log \pi_{\text{base}}(y)$	RL 策略的轨迹在基座模型下是否本来就很可能出现	低困惑度是激发的必要条件，而非充分条件

14.1.2 我们能测到什么

既然理论无法定夺答案，一切就都落到代理指标上，而每个代理指标都有其特定的混杂因素。表 14.1把它们汇集在一起。

最后一行值得说几句，因为它是最接近直接检验的东西。如果 RL 确实装入了全新的行为，那么基座模型应当给 RL 策略那些成功的轨迹赋予极低的似然。Yue et al. (2025a) 做了这项测量，报告的结果恰恰相反：RL 模型的输出稳稳地落在基座模型的高似然区域之内。这是支持「激发」的证据，但要加一条保留：一条轨迹可以逐词元看都不令人意外，却仍然是基座模型基本上永远不会组装出来的那一条——困惑度是一个平均量，而平均量会掩盖尾部。

14.2 悲观一方的论据

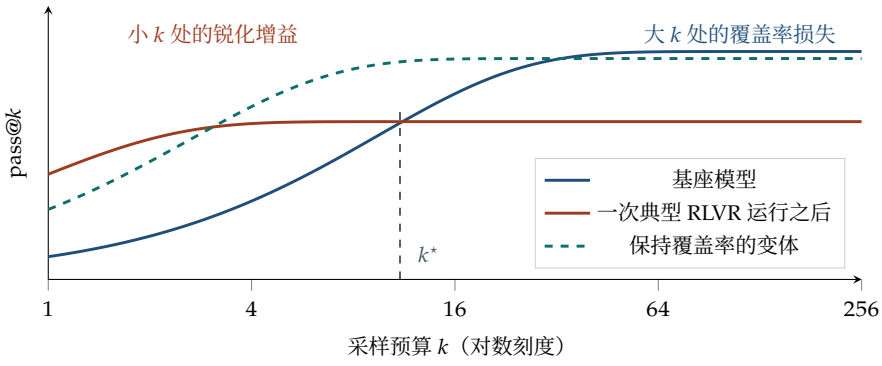


图 14.1: 示意图，非实测数据。本章核心的那个定性特征：强化学习抬高 pass@1，并在越过交叉点 $k^\star$ 之后压低 pass@k，于是两条曲线相交。曲线取自任意选定的函数形式，只为把交叉显示出来；任何一根坐标轴都不承载定量含义。第三条曲线表示的是，一个用部分 pass@1 去换覆盖率的方法——仅用负样本的训练（节 14.2.3）或 pass@k 目标（节 14.7）——在同一组坐标轴上是什么样子。

14.2.1 引发这场争论的那个结果

Yue et al. (2025a) 追问 RLVR 是否激励出了超越基座模型的推理能力，并给出否定的回答。他们的方法就是图 14.1所勾勒的那一套：用大 k 处的 pass@k 去探测能力边界，跨越多个模型族、多种 RL 算法，以及数学、代码与视觉推理基准。按重要性排序，他们报告的发现如下：

1. 在小 k 处，RLVR 训练出的模型胜过其基座模型；而在大 k 处，基座模型取得更高的 pass@k。两条曲线相交。
2. 覆盖率与困惑度分析表明，RL 模型的推理路径本就存在于基座模型的分布之中，并受其约束。
3. 若以基座模型的 pass@k 为上界，六种流行的 RLVR 算法彼此表现 相近，而且全都离那个上界很远。
4. 相比之下，蒸馏确实扩展了边界，因为它从教师那里引入了新的模式（节 14.8）。

对读过第12章的读者来说，第 3 条发现应该是最不舒服的一条。那一章的整套文献——十种病理、十几种有名字的算法、对归一化项的仔细消融——按这项测量来看，不过是在一个边界由预训练所设定的区域内部做的一些小幅移动。这与 Khatri et al. (2025) 的分解（节 12.12）是一致的：在那里，大多数干预动的是算力效率指数 B 而不是渐近值 A——各种算法的差别在于锐化得多快，而不在于它们能达到什么。还要注意，Yue

et al. (2025a) 自己就提出了两条出路——持续扩大规模，以及多轮次的智能体–环境交互——而这两条都作为 2026 年的调和方案在 节 14.5中重新出现。这篇论文比它那句口号要谨慎得多。

14.2.2 机制：受支撑集约束的优化

Wu et al. (2025a) 提供了Yue et al. (2025a) 所测到的那个现象背后的论证。他们的框架是：RLVR 是一种受支撑集约束的优化——策略梯度更新对已采样的轨迹重新加权，而一条从未被采样到的轨迹就永远不会被重新加权，因此优化被限制在当前策略所能产出的范围之内，并且传递地，被限制在基座模型一开始所能产出的范围之内。这就是把 定理 14.2改述成一条关于优化器可达集、而不是关于目标函数不动点的陈述；而且这是更有用的版本，因为它在 $\beta = 0$ 时依然成立。

他们的两项发现值得原样陈述。

第一，他们指出存在一种熵–奖励权衡，并报告说「在更大的采样预算下，经验支撑集的收缩通常压过其扩张」。两个方向都会发生——RLVR 确实会把不可忽略的质量放到某些基座模型鲜少产出的轨迹上——但在大 k 处的净效应是收缩。这个主张说的是两种真实效应之间的平衡，这也是为什么把这篇论文概括成一个数字的做法通常都是错的。

第二，也更为微妙：

常见陷阱: 词元熵那个结果经常被引错

Wu et al. (2025a) 报告说，RLVR 可以在降低答案级熵的同时 提高词元级熵。策略在局部变得更不确定——在单个解码步上出现更多犹豫、更多分叉性的措辞——却收敛到更少的几个不同最终答案上。这并不矛盾：词元层面的熵是局部条件分布的平均，而一条长轨迹可以途经许多高熵位置，同时每条路径都汇入同一个终点。其实践后果是：你仪表盘上一条持平甚至上升的词元熵曲线，并不是多样性得到保持的证据；任何只报告词元熵的论文，都没有测量 pass@k 所关心的那个量。请在规范化后的最终答案上测量熵。

他们开出的药方是从诊断中直接推出的：要挣脱这条牵绳，就需要能把概率质量播撒到欠代表的解区域的机制。组相对方法族中没有任何东西能做到这一点，这也是 节 14.10把它列为本章核心开放问题的原因。

14.2.3 佐证性机制

悲观一方的论据不是一篇论文；它是一套由四个独立方向支撑起来的机制，这使它高于「单篇证据」这一层级。

熵是货币。 Cui et al. (2025b) 的经验规律(12.10) 指出，验证奖励是所消耗的策略熵的指数函数，并在零熵处有一个天花板。如果奖励是用熵买来的，而熵又正是大 k 处的 pass@k 所需要的，那么 pass@k 的退化就不是某次调参糟糕的运行的副作用：它是目标函数所收取的价格。他们的机制性解释——熵的变化由词元概率与优势之间的协方差驱动，因此高概率高优势的词元会降低熵——说明这笔费用是优先向策略已有的主导模式征收的，而这恰恰就是模式锐化。

正向强化才是元凶，而且它是可分离的。 Zhu et al. (2025b) 把 RLVR 梯度分解为正样本部分（PSR，强化正确的采样轨迹）与负样本部分（NSR，惩罚错误的采样轨迹），并分别单独用它们训练。所报告的结果是一种干净的分离：单用 PSR 会提高 pass@1 却退化大 k 处的表现；而单用 NSR——从头到尾一次都没有强化过正确回复——却在整个 pass@k 谱段上（直到 $k = 256$）都优于基座模型，并常常与 PPO 和 GRPO 持平甚至更好。他们的梯度分析给出了机制：惩罚一个错误答案会移走它的质量，并按模型的先验比例把这部分质量重新分配到其余候选上，所以 NSR 在不锁定某个模式的前提下细化了一个已有的分布；而 PSR 则把质量集中到恰好成功的那条具体轨迹上。他们的 W-REINFORCE 只是简单地用 $\lambda = 0.1$ 给正项降权。

这是悲观派文献中机制性信息量最大的结果，因为它把覆盖率损失定位到某个特定估计量的某个特定项上，而不是笼统地归咎于「RL」。它还预言了保持覆盖率的训练是可行的，并且代价已知，这正是图 14.1中第三条曲线所代表的东西。

RL 触及的模型部分非常少。 在权重这一侧，Mukherjee et al. (2025) 报告说 RL 微调把更新集中在很小的子网络里，Zhu et al. (2025a) 论证 RLVR 可证明地是在权重矩阵的主方向之外学习的；Zhu et al. (2026b) 在此基础上提出了谱继承的观察——RLVR 复用基座模型的权重谱，改变的是与之相伴的奇异向量框架——并把它变成了一个优化器。一个对参数化改动如此之少的过程，先验上更可能是在重组一个已有的函数，而不是在装入一个新的函数。这是启发性的而非决定性的：小的参数变化可以带来大的功能后果，而且这几篇论文都没有直接测量 pass@k。

奖励所携带的信息非常少。 节 11.8已经引入了应当放在这里的两个结果。Wang et al. (2025e) 报告说，在单个训练样本上做 RLVR，能恢复包含该样本的 1.2 千样本集所带来增益的大部分，而且这一效应在控制了格式矫正之后仍然存在；Shao et al. (2025) 报告说，在 Qwen2.5-Math-7B 上用随机指派的奖励做 GRPO，能产生真值奖励所带来增益中的很大一部分。两者以同样的方式、出于同样的理由，对「激发」这件事具有信息量：一个几乎不携带任务特定信息的训练信号，不可能装入任务特定的能力，所以真正得到改善的必然是对预训练模型本就具备的行为的检索。Shao et al. (2025) 点明了机制——截断的不对称性会放大高先验的预训练行为，而与奖励内容无关，这正是节 12.6中那条累退税论证朝着无益方向运行的情形——而使这一对结果真正可用的是他们的对照：该效应在 Qwen 上很强，在 Llama3 与 OLMo2 上则基本不存在。仔细读的话，这些结果界定的是奖励在一次典型运行中的贡献上限；它们并没有界定 RL 在换用另一种探索机制之后能贡献多少，而把它们当作后者来引用是一个常见错误。

为什么锐化理应是默认结果，值得用一句话讲清楚，因为这句话很短。组相对估计量从当前策略的样本中算出优势，再把它乘以在同一批样本上的 $\nabla_\theta \log \pi_\theta$。因此梯度中的每一项都支撑在策略本就会生成的轨迹上，于是最便宜的改进方向永远是「多做一些已经奏效的事」，而(12.10)的熵账本告诉我们，被采取的正是这个方向。扩张则要么需要一种能在当前模式之外采样的机制——带重置的长训练、环境交互、pass@k 式的目标——要么需要一个外部的轨迹来源，那就是蒸馏。

14.3 多样性与模式坍缩

pass@k 的退化是一个更广泛现象的一张可见面孔，而把三种经常被混为一谈的多样性概念分开是值得的。

词元级熵 是 $\mathbb{H}[\pi_\theta(\cdot|s)]$ 在所访问状态上的平均（节 2.5）。它可以精确计算，记录起来不花钱，并且是 节 12.6中坍缩动力学的标准早期预警指标。按上面那条陷阱所说，它同时也是我们真正关心的那个量的最弱代理。

答案级多样性 是对固定提示、在规范化后的最终答案上的分布的熵。这才是 pass@k 所敏感的那个量：如果策略以 0.97 的概率吐出同一个答案，那么再大的采样预算也找不出另一个来。测量它需要与验证器相同的规范化机制，因而也继承了它的缺陷（节 11.6）。

语义多样性或策略多样性 是该策略能够产出的不同解法的数目——反证法与构造法、代数法与几何法、暴力枚举与巧思。它是把模型当作数据生成器使用时、以及在创造性任务上真正要紧的那个量；它没有公认的估计量，而且几乎从来没人报告。一个策略可以在保持答案多样性的同时坍缩到单一解法上，而这种失效在 表 14.1的每一个指标里都是不可见的。

14.3.1 这是目标函数在按设定运作

把多样性坍缩读作一个实现上的缺陷是不对的。节 2.4确立了本书中每种方法所用的反向 KL 都是寻模的：若策略把质量放到参考分布没有质量的地方，它就是无穷大；而策略放弃参考分布支撑集的大部分，它却毫不介意。节 1.2解释了为什么会选择这种不对称性——预训练那种覆盖质量的前向 KL 学到的是一个由质量参差的人格所构成的混合，而对齐的任务就是选出并放大其中一个成分。选出一个成分就是丢弃其余成分。

所以多样性的代价是计入目标函数定价之内的，而不是被偷偷夹带进来的。带反向 KL 锚点的奖励最大化，是把一个寻模的操作施加在一个覆盖质量的先验上；每一种通过集中策略来提高 pass@1 的方法，做的都是它被要求去做的事。因此「RL 把我模型的多样性搞坍了」不是一份缺陷报告；它是在陈述：你所设定的目标不是你想要的那个目标。相应地，补救办法全都是对目标函数的修改——一个 pass@k 目标（节 14.7）、一个只用负样本的梯度（节 14.2.3）、一个依赖提示的锚点（节 14.6）——而不是把同一个目标调得更好。

14.3.2 它在实践中的代价

有三项代价值得点名，因为它们在一个项目的不同时点、由不同的人来支付。

第一，**推理时的选择机制退化**。对着奖励模型或验证器做的 best-of-n、自洽性投票，以及智能体式的重试循环，全都是把多样性换成准确率。一个已经坍缩到单一模式的策略，给出的 best-of-n 曲线在 n 上是平的：你在改善报告中那个单样本数字的同时，把第10章中的算力–准确率权衡推向了不利的方向。

第二，**模型作为数据生成器变差了**。图 1.2中的外层循环——对当前最好的策略做拒绝采样、按正确性过滤、在结果上做微调——是一条生产流水线中大部分复利式改进的来源，而它的产出率是由覆盖率而不是由 pass@1 决定的。一个在第一次尝试就能解出 60% 提示、但在 256 次尝试中始终解不出剩下 40% 的检查点，恰恰对你最需要的那些提示产出不了任何新的训练数据。团队通常要晚一代才会发现这件事。

第三，**开放式输出的质量下降**，而且是以奖励测不到的方式下降：不同的表述框架变少、结构更加套路化，以及一个被重度优化过的模型所特有的那种扁平化文风。这是节 6.5中的对齐税现象从多样性一侧看过去的样子，也是生产配方即便在只用验证器的论证说不需要 KL 项时仍然保留 KL 项的主要原因。

工程实践: 记录你真正关心的那种多样性

在节 11.7的诊断量集合里再加三个量，鉴于你本来就在为每个提示采样 G 个回复，这三个都很便宜。

- **每个提示的不同规范化答案数**，在批上取平均。这是 $k = G$ 处的答案级多样性，而且它会在留出集上的 pass@k 之前先动。
- **留出集上 $k \gg G$ 处的 pass@k**，用无偏估计量 (2.23)，在固定的提示集上、固定的温度下，每隔几百步测一次。这是唯一能在你还来得及叫停这次运行时就检测出节 14.6那种失效的测量。
- **同一批提示上的基座模型 pass@k**，只需算一次。没有这个界，一条下降的曲线是无法解读的。

词元熵仍然留在诊断量集合里——它是优化器病理最快的指示器——但它不是一项多样性测量。

14.4 乐观一方的论据

静态推理范围内最强的反证是 ProRL (Liu et al., 2025d)，而且它质疑的是前提而不是测量。他们的配置在四个方面不同于典型运行，而这四点全都指向同一个方向：长时间训练（步数比Yue et al. (2025a) 所评测的那些运行多一个数量级）、为稳定性而设的显式 KL 项、周期性的参考策略重置以使锚点不会渐近地冻结进展，以及一个刻意做得多样的任务混合，横跨数学、代码、逻辑谜题、STEM 问题与指令遵循。

他们报告的是一个支撑集扩张的主张，而不是一个锐化的主张：RL 训练出的模型在很宽范围的 pass@k 评测上都胜过基座模型，其中包括那些无论尝试多少次基座模型都完全失败的情形。最后那半句才是关键。一个基座模型在任何负担得起的 k 下都得零分的提示，没有任何质量可供重新分配，所以能解出它的策略在经验上是把质量放到了基座模型没有质量的地方——按定理 14.1这就是习得。他们还报告说，边界改进与训练时长相关，也与基座模型在该任务上的初始能力相关，这构成了一个自洽的故事：重置让信赖域保持活跃，时长累积位移，而多样的任务使熵预算不会被消耗在同一处。

常见陷阱: 算力这个混杂因素让 2025 年的分歧有一部分是假象

Yue et al. (2025a) 测的是典型的已发表运行；Liu et al. (2025d) 测的是一次异常长的运行。这是两个不同的实验，任何一方都没有控制另一方的变量。Khatri et al. (2025) 后来提供了本可以定夺此事的工具——拟合 sigmoid(12.14)，比较渐近值而不是终点值——并指出他们自己那次 8B 的验证运行，又比 ProRL 长了好几倍。在边界问题被作为算力的函数来提问之前，即在 pass@k 沿曲线的若干点上而不是只在终点被测量之前，「RL 是否扩展了边界?」都是欠定的。这是 2025 年文献中双方最常见的方法学错误。

第二条较弱的乐观线索关心的是在训练什么，而不是训练多久。Setlur et al. (2025c) 论证可以让 RL 去训练探索本身，从而使测试时计算能够外推到训练预算之外；而Wang et al. (2025e) 的单样本结果中也包含这样一项观察：促进探索——在那里是用一个系数经过仔细挑选的熵项——对该效应至关重要。两者都与这样一种读法一致：扩张是可以得到的，但必须为它明确付费，因为默认的估计量买不到它。

14.5 2026 年的调和

2026 年的三项结果收敛到了一个答案上，而正是这种收敛使它们值得讲授。三者都是新近的预印本；请把每一篇单独看作单篇证据，而把它们之间的共同图景看作稍强一些的证据。

14.5.1 发现与锐化是先后相继的，而不是相互对立的

Tang et al. (2026) 在一万亿参数规模上跑了零 RL——没有监督式的热启动——并报告说学习曲线有两个可区分的阶段：先是一个发现阶段，其中沉睡的通路被解锁，推理边界得以扩张；随后是一个漫长的锐化阶段，其中策略在发现阶段所确立的边界之内被细化。

这几乎是机械地化解了 2025 年的分歧。如果两个阶段是先后相继的，那么你在何时测量就决定了你得出什么结论：一次短探测捕捉到的是发现，一次典型中等长度的运行由锐化主导，而一次带参考重置的异常长的运行会重新进入发现。它还预言了上一条陷阱中的算力混杂因素，而不只是与之相容。图 14.2画出了这一形状。

这篇论文还因第二个理由而与此处相关：它报告了零 RL 在此前无人尝试过的规模上是可行的（据他们报告，仅第一阶段 RL 就在 AIME 2026 上达到 84.2%，且没有使

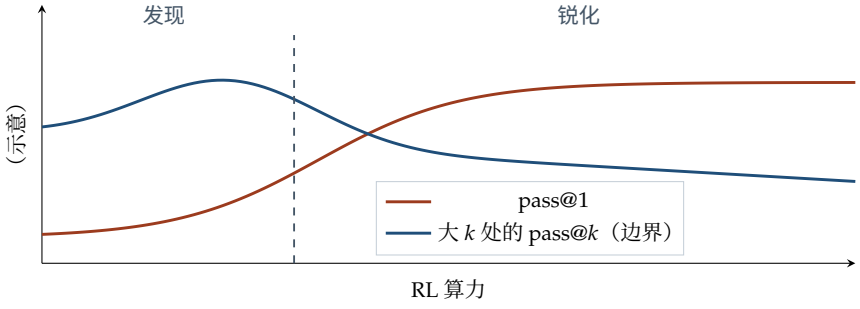


图 14.2: 示意图，非实测数据。对Tang et al. (2026) 的两阶段解读：先是一个大 k 边界扩张的早期阶段，随后是一个漫长的阶段，其中 pass@1 继续上升而边界收缩。坐标轴刻意不加标注；此图断言的是事件的先后次序，而不是量值。按这一解读，节 14.2 and 14.4中悲观与乐观两派的文献，分别采样的是虚线的右侧与左侧。

用任何人工标注数据），同时也报告了在那个规模上 KL 惩罚是承重的，没有它训练–推理之间的对数概率差距就会发散、奖励随之崩塌。按这一说法，发现需要一种稳定机制，才能撑到它发生的那一刻。

14.5.2 智能体式 RL 会扩展边界，静态 RL 不会

第二项结果最为锋利，因为它隔离出来的是一项任务属性，而不是一种算力区制。PASS@(k, T) 研究 (PASS@(k,T) Analysis, 2026) 引入了一个二维指标，同时改变采样预算 k 与所允许的交互深度 T，并评测用 RL 训练出来的工具使用智能体。与静态推理上的发现相反，他们报告说工具使用的 RL 确实扩大了能力边界：RL 智能体的通过率曲线位于基座模型之上，而且差距在大 k 处变宽而不是收敛。这种扩张专属于那些需要 组合式、序贯式信息搜集的任务；在较简单的任务上，这些智能体的表现与 Yue et al. (2025a) 的预言一致。他们的对照才是重要的部分：在训练数据相匹配的条件下，监督微调恰恰在那些组合式任务上使边界倒退，这就把自主导向的探索——而不是数据——隔离出来作为因果因素。

他们自己那句一行总结是对这一调和最好的表述：两派都对，只是在不同的任务类型上。

一旦说破，机制就并不神秘。在单轮推理中，策略唯一的新颖性来源就是它自己的先验；环境并不贡献任何它本来就不包含的东西。而在一段使用工具的回合中，搜索引擎或解释器返回的一次观测是模型此前没有的信息，以它为条件的轨迹在任何有用的意义上都不是从基座模型支撑集中抽出的样本。环境交互正是Wu et al. (2025a) 所

说需要的那类质量播撒机制——这也是为什么Chapters 17 and 18关乎的是能力，而不只是部署。

14.5.3 与预训练的接口：一个受控的设定

第三项结果攻的是前两者都攻不下的那个问题：在真实的 LLM 语料上，人们无法在其他一切保持不变的情况下改变预训练。因此Shen et al. (2026a) 用国际象棋作为试验台：在人类对局上预训练从约 5M 到 1B 参数的模型，在合成的推理轨迹上做监督微调，再在带可验证奖励的残局谜题上跑 RL。有两项发现与本章直接相关。

在给定 RL 算力预算下，RL 之后的表现可以由预训练损失很好地预测，而 RL 奖励曲线的斜率大致随预训练词元数线性改善。因此 RL 的生产力是预训练留下了什么可供利用的函数——这就是「RL 无法激发出本就不存在的东西」的定量版本，也是支持 节 1.4中那个中训练阶段的一条论据。

而在这场辩论本身上，他们报告说 RL 并不只是把监督策略锐化了：在容易的谜题上，它放大策略本来就偏好的正确着法；而在困难的谜题上，它让那些在监督策略下几乎不存在的正确着法浮现出来。他们报告这一模式在一个用数学文本训练的 1B 模型上得到了复现。

核心思想: 把这一调和写成一条规则

RL 究竟是扩张还是锐化，取决于三个变量的相互作用，而 2025 年的实验一个都没有控制：**算力**（发现先于锐化，所以短运行与长运行会得出相反结论）、**任务类型**（与环境的组合式交互提供了真正的新信息，静态单轮推理则没有），以及 **逐提示的区制**（在容易的提示上是放大，在困难的提示上是让近乎不存在的行为浮现，以及介于两者之间的那种有害情形——节 14.6）。凡是形如「RL 扩展／不扩展边界」而未附带区制的主张，都不是关于这个世界的主张。

历史注记: 各说各话的十五个月

这段经过值得记住，它是一个领域如何把问题问错的案例。Yue et al. (2025a) 发表于 2025 年 4 月，被读成「RL 不管用」；Liu et al. (2025d) 几周后出现，被读成「RL 管用」；Wu et al. (2025a) 在 2025 年 7 月带着那个机制到来，它既解释了前者，仔细读的话也容许后者。其间的争论极少涉及测量本身，而几乎完全没有涉及那些变量——训练算力、任务类型、逐提示的区制——也就是 2026 年的几篇论文所表明的决定性变量。这段拖延的代价并不在于概念上的困难；而在于两派的

表 14.2: 各项证据，附带其证伪条件。最后一列才是要读的那一列：一项主张若其证伪条件在你的规模上无法测量，就不是你能据以行动的主张。证据层级：R = 有若干条独立线索支撑的机制，C = 仍有争议，S = 单篇论文，可能不稳固。

主张	来源	方法	层级	什么会证伪它
在静态推理上，大 k 处基座模型超过 RL 模型	Yue et al. (2025a)	跨模型族、算法与基准的 pass@k 扫描	C	在两个模型族上，算力匹配的运行其交叉点 $k^\star$ 超出任何负担得起的预算
RLVR 停留在基座模型的经验支撑集之内；收缩压过扩张	Wu et al. (2025a)	支撑集分析加熵–奖励核算	C	存在一个大 k 处基座 pass@k = 0 而 RL 能解出的提示集，且验证器可靠性经过审计
覆盖率损失专门来自正优势那一项	Zhu et al. (2025b)	梯度分解；PSR 与 NSR 分别训练	S	在非 Qwen 模型族上，仅用负样本的训练反而退化大 k 处的 pass@k
奖励是用熵买来的，且在零熵处有天花板	Cui et al. (2025b)	在 11 个模型与 4 种算法上的双参数拟合	R	某次运行在熵已经压平到接近零之后奖励仍持续上升
带参考重置的长时间 RL 会扩展边界	Liu et al. (2025d)	长运行；KL 加周期性参考重置；多样任务	C	一旦在匹配的采样预算下测量基座模型，同样的方案就不再带来大 k 处的增益
发现与锐化是先后相继的两个阶段	Tang et al. (2026)	万亿规模上的单次零 RL 运行	S	在若干规模的长运行中，大 k 处的 pass@k 单调移动，无论朝哪个方向
工具使用 RL 在大 k 处拓宽边界；数据匹配的 SFT 则收窄边界	PASS@(k,T) Analysis (2026)	在采样预算与交互深度上的二维指标	S	存在一个组合式搜索基准，其上智能体的优势随 k 增大而消失
RL 之后的表现可由预训练损失预测	Shen et al. (2026a)	在国际象棋试验台上的受控预训练扫描	S	两个预训练损失相同、而 RL 之后的天花板实质不同的检查点
锐化的有害性专属于边界区制	Zhou (2026)	按冻结基座模型的行为对提示分层	S	倒挂反而集中在容易的提示或够不着的提示上

> 实验根本不可比，却整整一年没人注意到，因为 pass@k 是作为一个数字、而不是作为一条注明了采样预算的曲线被报告的。这一教训可以推广到这场争论之外，也正是节 14.9是一套流程而不是一个结论的原因。

14.6 锐化何时有害：一个刻画

2026 年在操作上最有用的结果，是那个不再问锐化好不好、而改问它在何处起作用的结果。Zhou (2026) 把这一现象命名为 **pass@k 倒挂**——pass@1 改善而大 k 处的 pass@k 退化——并论证它是一种边界处的模式锁定失效，而不是熵坍缩的症状。这一区分很要紧：熵坍缩是策略的一项全局性质，用全局工具来处理（截断不对称、KL、以协方差为靶的惩罚项，节 12.6）；而模式锁定是依赖提示的，全局工具的形状对它不合适。

表 14.3: Zhou (2026) 的提示分类法，附带识别每个区制的基座模型特征以及锐化对它的作用。这些区制是由基座模型的行为定义的，所以它们可以在训练之前一次算好，并用来给之后的每一项测量分层。四行中有两行是所有有意思的失效所在之处，而它们恰好是只看 pass@1 的评测无法区分的两行。

区制	基座特征	锐化的作用	正确做法
已解决的容易题	pass@1 高；到 pass@k 的余量很小	可取：主导模式本来就是正确的，所以集中质量把一次频繁的成功变成一次可靠的成功	放手优化
可达	pass@1 中等；正确模式有竞争力	有益：RLVR 所报告的增益正是在这一区制中挣来的	优化，并监控
边界	pass@1 低，大 k 处 pass@k 高；正确轨迹存在但不是主导模式	有害：策略锁定到那个主导的——错误的——模式上，并摧毁了重复采样本来能够找回的覆盖率	加锚，或先搁置
够不着	在任何负担得起的 k 下都有 pass@$k \approx 0$	无害：没有覆盖率可失，而且整组全零，梯度本来也近乎为空	排除或推迟

他们的教学工具是一套按冻结基座模型的行为定义的提示分类法，重制为 表 14.3，并画在图 14.3中。

他们为边界区制所提出的机制值得讲清楚，因为它与第12章的账目直接相连。在一个边界提示上，大多数采样组全是零，因而根本不提供任何局部的纠正信号；偶尔含有一条正确采样轨迹的那一组提供一个稀疏的正向信号；与此同时，来自其他提示的更新在锐化共享参数。于是策略在这个边界提示上变得更自信，早于它那条稀有的正确分支被强化到足以在这种自信下存活。他们把这一结果称为 依赖提示的遗忘：策略恰恰在这些提示上摧毁了重复采样本来能够找回的覆盖率。

常见陷阱: 动态采样恰好删掉了这一失效所依附的那些提示

零方差过滤（节 12.4）会丢弃所有 G 个样本获得相同奖励的采样组。对于一个逐提示成功概率 $\mu = 0.02$、$G = 8$ 的边界提示，(12.4)给出的退化概率高于 0.85：这个提示大约每七次里有六次被踢出批次，而被保留时也几乎不贡献什么。于是标准配方同时做了两件事：（a）移除了那个本可以强化稀有正确分支的纠正信号，（b）继续通过共享参数把来自其他每一个提示的锐化压力施加过来。效率上的修补与覆盖率上的失效，是同一机制的两面，而据我们所知还没有论文直接测量过这一相互作用。这是一个不错的研究问题，而且成本低廉。

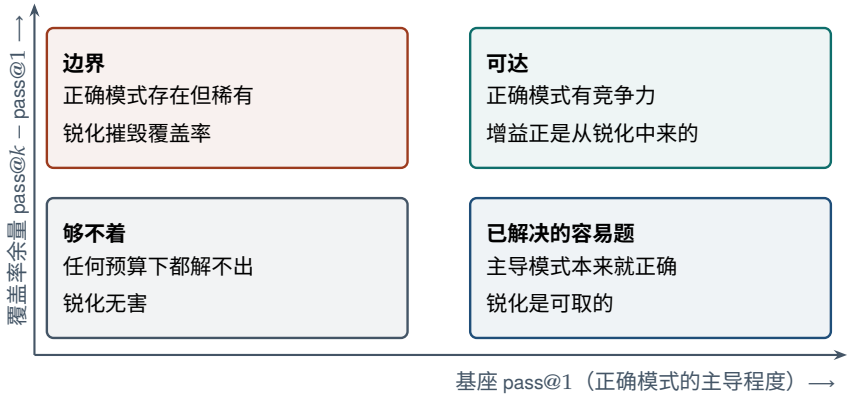


图 14.3: 示意图。表 14.3的四区提示分类法，放在定义它的两根坐标轴上：在基座模型下正确模式有多主导，以及重复采样又能增加多少覆盖率。此图不携带任何定量阈值——各区之间的界限是连续的，且依赖于基准——但后果的排序是本章最可操作的主张。

他们提出的防护措施——逐题的基座锚定——是该诊断所蕴含的最小工具：一个依赖提示的、指向基座策略的 KL 锚点，在基座持有稀有而有用的质量之处施加，在证据支持锐化之处松开。注意这需要对基座模型的逐提示行为有一个估计——这又是一条理由，说明 表 14.1为什么把基座 pass@k 列为一项应在训练之前而不是之后去做的测量。

他们的收尾表述是对本章实践内容最好的概括，我们采纳它：后训练不应只问优化得 多强；它必须问优化在何时是安全的。

14.7 pass@k 应该成为目标函数吗?

如果优化 pass@1 会摧毁 pass@k，那么显而易见的回应就是去优化 pass@k。第12章把这个问题留了下来（节 12.13）；这里是它的现状。

14.7.1 PKPO：它是良定的，而且可以归结为奖励塑形

Walder and Karkhanis (2025) 为 pass@k 及其梯度导出了低方差的无偏估计量，既适用于二元奖励也适用于连续奖励，并证明最大化它们可以归结为在联合变换后的奖励上做标准的策略优化——这里的变换是整组奖励的函数，而不是每个样本自身奖励的函数。他们报告的结果中有三点对实践要紧。与早先局限于 $k = n$ 的工作不同，他们的构造能处理任意 $k \leq n$，正是这一点使它可以配合节 11.7中的组大小使用。在训练过

程中把 k 退火调低，能同时得到强的 pass@1 和强的 pass@k。而在那些 pass@1 优化会停滞的任务集上，pass@k 目标能打通学习，他们把这归因于优先考虑一组样本的联合效用，而不是每个样本各自的效用。

最后一点会改变你该如何看待这个目标函数。它不只是一个保持多样性的替代品；它是一种探索机制，而且这个机制在两行代数里就看得见。

推导: pass@k 目标就是一个依赖难度的提示权重

固定一个提示，并令 $p = \mathbb{P}_{y\sim\pi_\theta}[r(x,y) = 1]$，于是总体的 pass@$k$ 为 $J_k(\theta) = 1 - (1-p)^k$，因为在给定提示的条件下这 k 个样本是 i.i.d. 的。求导得

$$\nabla_\theta J_k = \underbrace{k\,(1-p)^{k-1}}_{w_k(p)}\,\nabla_\theta p = w_k(p)\,\nabla_\theta J_1. \tag{14.2}$$

所以在单个提示这一层面上，优化 pass@k 恰好就是带一个标量权重 $w_k(p) = k(1-p)^{k-1}$ 地优化 pass@1，而这个权重只依赖于该提示当前的难度。该权重在 $p = 0$ 处为 k，并随 $p \to 1$ 衰减到 0：相对于 pass@1，pass@k 目标单调地给困难提示加权并放弃容易的提示，其激进程度由 k 控制。由此立刻得到两条推论。把 k 从大退火到 1，因而是一个用课程表达出来的探索到利用的时间表，这就是它能同时兑现两项指标的原因。而 w_k 恰好是 (12.2)中逐组标准差所强加的那种意外加权的反面，后者对几乎已解出和几乎不可能的提示一视同仁地加权；这一对比是看清(12.2)从来不是任何人想要的目标的好办法。完整的估计量比(14.2)要精细，因为 p 必须从提供梯度的那同一批 G 个样本中估计——节 2.7中的组均值偏差在这里重新出现——而且因为组内的信用分配（该强化 G 个样本中的哪一些）并不由这个提示层面的论证所决定。这正是Walder and Karkhanis (2025) 所提供的东西。

14.7.2 统一的视角，以及不同意见

Thrampoulidis et al. (2025) 给出了让这一文献变得可处理的框架：整个 pass@k 策略梯度族都是被理解为代理奖励最大化的优势塑形的一个实例。这里没有一堆目标函数动物园，只有一个构造——把优势替换为该组奖励的某个塑形函数，然后追问这一塑形恰好是哪个代理目标函数的梯度。对一本教科书而言这是正确的引用，因为它把一堆技巧变成了一个设计空间，就像表 12.5对组相对方法族所做的那样；而且它让你可以

检查一个提议的塑形是否对应于任何一个目标函数。有好几个已发表的启发式做法并不对应。

不同意见来自Yu (2025)，他主张 pass@k 应当被读作探索的诊断指标，而不应被直接优化。Dragoi et al. (2025) 提出了一个相关但不同的观点，主张使用广度与深度两类指标，理由是 pass@k 把两件值得分开的事混在了一起。Chen et al. (2025h) 居于两者之间，明确把 pass@k 训练信号当作一种探索–利用的平衡手段，而不是当作部署目标。

工程实践: 我们的看法

有三种立场都站得住脚，而它们并不是同一种立场。

- **如果你的部署流程会花掉 k 个样本**——对着验证器做 best-of-n、自洽性、一个会重试的智能体——那么 pass@k（或相应的、考虑选择器的变体）就是你的目标函数，改去优化 pass@1 是一个设定错误。让训练目标与推理流程相匹配。这种情形很常见，而且长期被忽视。
- **如果你部署的是单个样本**，那么按节 14.7.1，pass@k 是一个正则项，其真正的职责是探索。那就问问你要的是探索还是覆盖率。如果是探索，一套难度课程（节 12.4.1）能更直接也更便宜地达到同样的加权。如果是覆盖率，Zhu et al. (2025b) 的仅负样本梯度或一个依赖提示的锚点（节 14.6）瞄得更准。
- **不要优化你唯一的诊断指标**。这是Yu (2025) 论证的最强形式，也是一个 Goodhart 式的论证（节 1.7，主题 5）：一旦你选定的那个 k 处的 pass@k 进入了损失，它就不再是对覆盖率的独立测量，而你将需要在某个更大的 k' 处的 pass@k'——并且以同样的谨慎去测量——才能知道覆盖率究竟是被保住了，还是只是被挪了个地方。

14.8 强化学习与蒸馏之比较

蒸馏是这场边界辩论所需要的对照条件，因为它无疑地在执行习得：轨迹来自教师，所以教师把质量放在哪里，质量就出现在哪里，完全不受支撑集的约束。

Yue et al. (2025a) 正是做了这项比较，并报告说蒸馏扩展了基座模型的 pass@k 边界而 RLVR 没有；这与他们的机制一致，也是现有最干净的一项证明，说明边界并非不可移动——只是 RL 不是移动它的那件工具。专门的正面对比是 Kim et al. (2025)，其

表 14.4: 强化学习、蒸馏，以及它们在 2026 年的混合形态，就各自对能力边界做了什么、以及各自的代价进行比较。实践上要读的是最后一行：两种经典方法有互补的失效模式，而同策略蒸馏是一次尝试，想要取其一的习得能力与其二的保留能力。

	RLVR	**异策略蒸馏**	**同策略蒸馏**
需要	一个验证器	一个更强的教师	一个更强的教师，且在线
轨迹来自	策略自身	教师	策略，由教师打分
支撑集	受限于策略自身，见Wu et al. (2025a)	教师的	策略的，但被密集打分
对 pass@1 的影响	大幅提升	大幅提升	大幅提升
对大 k 边界的影响	持平或为负	扩展 (Yue et al., 2025a)	号称居中；尚未被很好地测量
遗忘	低；KL 近邻 (Shenfeld et al., 2025)	较高：会覆写	按同一论证，低于异策略
成本主导项	采样轨迹的生成	教师推理，一次	教师推理，每一步都要

框架把准确率（单样本正确性）与能力（究竟什么是可达的）分开，并发现两种方法是沿这两条轴此消彼长，而不是一方全面压倒另一方。

这一结果的镜像是这场比较中最有意思的一条。Shenfeld et al. (2025) 追问为什么在同一个目标行为上，同策略 RL 比监督微调遗忘得更少，并把原因归于一种隐式偏置：在所有能解决新任务的策略之中，同策略 RL 倾向于找到 KL 意义上离出发点最近的那一个。使扩张受限的那份保守，同时买来了保留。这是把两种方法放在一起考虑的一个真正有用的方式，因为它使二者的弱点互补而不只是相异：蒸馏移动边界，并损坏它所覆写的东西；RL 保住一切，却什么也不移动。

在理论一侧，Wei and Kim (2026) 给出了一个 2026 年的分离性结果，论证在学习高效回溯这件事上 RLVR 相对监督微调有可证明的好处——这提醒我们，「RL 只会锐化」是关于可达集的陈述，而不是关于可达集内部样本效率的陈述。两种方法可以有相同的可达集，却在到达那里所需的数据量上相差悬殊，而回溯正是一个可信的例子：语料中关于错误恢复的示范很稀少（节 1.2），而同策略训练则免费地生成出那些失败状态。

同策略蒸馏是 2026 年的综合方案，也是上表的自然归宿：从学生采样、由教师打分、在学生自身的分布上更新，使得蒸馏那种密集的监督被施加在学生实际访问的状态

上，而不是施加在教师所访问的状态上 (Lu and Thinking Machines Lab, 2025)。2026 年中的一批预印本在探索这一设计空间，包括对哪些教师打分过的轨迹被采用进行筛选的奖励门控变体 (Reward-Gated On-Policy Distillation, 2026)，以及弱到强的设定 (Weak-to-Strong On-Policy Distillation, 2026)。截至撰写本书时，其经验图景只有几个月的历史，而同策略蒸馏在大 k 处的边界行为，尚未得到 节 14.2那批文献施加于 RLVR 的那种细致刻画。请把 表 14.4的第三列当作一个假设来看待。

核心思想: 用蒸馏来习得，用强化学习来锐化并保留

这两种方法不是在争夺同一个位置。蒸馏是唯一能可靠地把质量放到学生有效支撑集之外的廉价机制，而它为此付出的代价是遗忘。RL 是唯一能在不引入任何外部东西的前提下、在学生自身分布上提高可靠性的机制，而它为此付出的代价是覆盖率。因此一条流水线应当用蒸馏来移动边界、用 RL 来把内部做可靠——这恰恰就是 R1 流水线的次序（节 11.4）：从更强的轨迹做冷启动 SFT、推理 RL、从改进后的策略做拒绝采样 SFT、最后一个 RL 阶段，以及蒸馏到更小的模型。把它读作一套边界管理策略而不是一份阶段清单，这条流水线就是本领域对本章问题的隐含回答，而它是在这个问题被仔细提出的两年之前，凭经验摸索出来的。

14.9 该下什么结论，以及如何为你自己的模型测量它

本章诚实的结论是有条件的，所以可操作的内容是一套测量流程而不是一个裁决。下面这套流程并不昂贵：它的主要成本是基座模型的采样，那是纯推理的，可以做一次并缓存下来。

工程实践: 能为你的模型与任务定夺此问题的流程

1. **固定一个 200–1000 条提示的评测集**，并配一个经过审计的验证器。一个假阴性率为 10% 的校验器（节 11.6）会制造出本不存在的覆盖率损失，因为假阴性集中在那些不寻常但正确的答案上——恰恰就是大 k 处 pass@k 所测量的那条尾巴。
2. **先测基座模型的 pass@k**，在你的部署温度与提示格式下，每个提示取 $n \geq 256$ 个样本，使用无偏估计量(2.23)。报告整条曲线，而不是一个点。这就是那个界；没有它，一条下降的曲线无从归因。

3. **给提示分层**，用那次基座测量把它们分入表 14.3的四个区制，并按层报告之后的每一个数字。聚合的 pass@k 把期望行为相反的几个群体混在一起，这也是为什么聚合结果能被两派同时复现。
4. **让采样预算严格匹配**基座模型与 RL 模型：相同的 n、相同的温度、相同的最大长度、相同的停止条件。已发表的交叉与不交叉结果中，有相当一部分可归因于用适合 RL 模型的温度去评测基座模型。
5. **沿算力轴在若干点上测量**，而不只是在终点测，这样一条像 图 14.2那样的两阶段曲线才能与一条单调曲线区分开。把(12.14) 分别拟合到 pass@1 和大 k 处的 pass@k 不额外花钱，却给出两个可供比较的渐近值。
6. **至少使用两个互不相关的基座模型族** (Shao et al., 2025)。在这里这不是可选项：伪奖励那个结果意味着，只在 Qwen 上测到的覆盖率效应，可能只是那个预训练语料的一项性质。
7. **同时报告答案级熵与不同解法计数**，以免一条平坦的 pass@k 配上已经坍缩的多样性，被误当作能力得到了保持。
8. **给出 $k^\star$**，即交叉点，并附上它在不同随机种子上的不确定性。它是一个可解释的单一数字：如果 $k^\star$ 超过了你可能部署的任何预算，那么覆盖率损失就是真实但无关紧要的，而这是一个完全值得报告的好结果。

代码 14.1: 这套流程所依赖的两项测量：在整条曲线上求值的 (2.23)无偏 pass@k 估计量，以及表 14.3的基座模型分层。两者都从同一个逐提示正确计数的数组算出，所以基座模型的采样只需跑一遍。

```python
import numpy as np
from scipy.special import comb

def pass_at_k(n, c, k):
    """Unbiased pass@k for one prompt: n samples, c correct. Chen et al. (2021)."""
    if n - c < k:
        return 1.0
    return 1.0 - comb(n - c, k) / comb(n, k)

def curve(counts, n, ks):
    """counts[i] = number correct among n samples for prompt i."""
    return {k: float(np.mean([pass_at_k(n, c, k) for c in counts])) for k in ks}

def stratify(counts, n, k_max, easy=0.75, reach=0.25, floor=1e-9):
```

```
    """Assign each prompt to a regime using the *frozen base model* only."""
    labels = []
    for c in counts:
        p1, pk = c / n, pass_at_k(n, c, k_max)
        if pk <= floor:            labels.append("out-of-reach")
        elif p1 >= easy:           labels.append("solved-easy")
        elif p1 >= reach:          labels.append("reachable")
        else:                      labels.append("boundary")   # pk 高、p1 低
    return np.array(labels)
```

有两条结论稳固到可以不加保留地陈述。第一，聚合的 pass@1 改善不是能力提升的证据，而本领域是用昂贵的方式在 2025 年学会这一点的。第二，基座模型的 pass@k 曲线是一个推理 RL 项目中信息量最大的单一对象，却几乎从来没人测量，因为它要在一个谁都不打算发布的模型上花推理算力。

14.10 开放问题

把质量播撒到基座支撑集之外。 这是Wu et al. (2025a) 自己开出的药方，而至今仍没有一个干净的、纯 RL 的答案。蒸馏管用，但它预设了一个更强的教师，这在前沿处无济于事。参考重置 (Liu et al., 2025d) 是最接近纯 RL 的机制，而它的工作方式是允许无界的累积漂移，而不是播撒了什么。环境交互 (PASS@(k,T) Analysis, 2026) 确实提供了新信息，但只对那些拥有信息丰富环境的任务有效。缺失的是一种能把候选轨迹注入策略自身采样分布的机制——自生成的题目变体、无教师的搜索、在已识别的边界提示上刻意调高温度——并且要有一个论证说明被注入的质量为何能在优化中存活。

一个同时兼顾 pass@1 与覆盖率的目标函数。 对 k 退火是管用的 (Walder and Karkhanis, 2025)，但它是一个时间表，这意味着又多了一个超参数，以及事先对一个部署预算的承诺。Thrampoulidis et al. (2025) 的优势塑形框架是寻找单一平稳目标函数的正确去处，那个目标的最优解应当是一个既可靠又宽广的策略；这样的目标是否存在、以及它转而牺牲了什么，都仍然开放。

依赖提示的优化。 节 14.6的诊断，加上 Cheng et al. (2026) 关于容易提示与困难提示按不同机制扩展的发现，都蕴含着优化强度应当是提示的函数，而几乎没有人这么做。障碍在于估计（在负担得起的 n 下，逐提示的基座统计量是有噪的）与账目管理（逐提示的锚点意味着逐提示的参考计算）。两者看起来都可解决。

测量语义多样性。 对「这个策略能产出多少种不同的解题策略」还没有公认的估计量。在有之前，节 14.3中的第三项代价是不可见的，而大多数生产流水线所依赖的那条外层循环数据生成论证，就建立在一个没人测量的量之上。

这一切在转向智能体之后还能否成立。 PASS@(k,T) 分析（PASS@(k,T) Analysis, 2026）已经表明，静态推理上的结论无法迁移到使用工具的智能体上。如果 2026 年的调和方案是对的，即扩张是组合式环境交互的一项性质，那么边界问题就必须为每一类新环境重新提一遍，而数学上的那个答案自始至终都只是一个特例。

本章小结

- 支撑集保持（定理 6.3）并不能解决激发问题。它约束的是恰好为零之处；放大界(14.1)允许 $e^{\Delta/\beta}$ 这样一个倍数，在现实的 β 下它大得天文数字，而在 $\beta = 0$ 时它没有定义。真正起作用的概念是给定采样预算下的有效支撑集，那是一个经验量。
- 悲观一方的论据在机制上是自洽的，而不只是经验性的：Yue et al. (2025a) 报告基座模型在大 k 处反超 RL 模型，且六种算法相对那个界表现相近；Wu et al. (2025a) 把它解释为受支撑集约束的优化，其中经验支撑集的收缩压过扩张；而熵律(12.10)说，覆盖率就是为奖励付账的那个东西。
- 覆盖率损失可以归到某个特定项上。正样本强化抬高 pass@1 并退化大 k 处的表现，而仅用负样本的训练则通过按模型先验重新分配质量，在直到 $k = 256$ 的整个谱段上都有改善 (Zhu et al., 2025b)。
- RLVR 可以在降低答案级熵的同时抬高词元级熵。因此词元熵是优化器健康状况的诊断指标，而不是多样性的测量；只报告词元熵的论文并没有测量覆盖率。
- 多样性坍缩是反向 KL 目标在按设定运作，而不是实现上的缺陷，所以它的补救办法全都是对目标函数的修改。它的代价——更差的 best-of-n、下一代训练更差的拒绝采样产出率、更扁平的开放式输出——是在促成这次坍缩的那个指标之外支付的。
- 2025 年的分歧是欠定的。带参考重置的长时间 RL 报告了增益，其中包括在基座模型从未解出的提示上的增益 (Liu et al., 2025d)，但它与典型长度的运行之间的比较混杂了算力，而且双方都没有把 pass@k 作为训练算力的函数来测量。

- 2026 年的三项结果收敛到一个有条件的答案：发现与锐化是先后相继的两个阶段 (Tang et al., 2026)；工具使用 RL 在组合式任务上拓宽大 k 处的边界，而数据匹配的 SFT 反而收窄它 (PASS@(k,T) Analysis, 2026)；并且在一个受控的国际象棋与数学设定中，RL 在容易的实例上放大正确行为，却在困难的实例上让几乎不存在的正确行为浮现 (Shen et al., 2026a)。每一项都是单篇证据；而这一共同模式比其中任何一项都更强。
- 锐化在一个特定的、可识别的区制中是有害的：那些在重复采样下存在正确轨迹、但正确轨迹不是主导模式的提示。这一失效是依赖提示的遗忘，而标准的零方差过滤器恰恰把那些提示从批次中移除，与此同时锐化压力仍继续通过共享参数抵达它们。
- 优化 pass@k 是良定的，并且在提示这一层面上，它恰好就是带难度权重 $w_k(p) = k(1-p)^{k-1}$ 的 pass@1 优化（节 14.7.1）；这正是为什么对 k 退火表现得像一套探索课程，以及为什么这个目标能打通停滞的学习。是否采纳它，取决于你的部署流程是否真的会花掉 k 个样本。
- 蒸馏与 RL 是互补而非竞争的：蒸馏移动边界并遗忘；RL 待在边界之内并保留，出于同一个 KL 近邻的理由 (Shenfeld et al., 2025)。R1 流水线的次序就是这一洞见的操作化形式。

延伸阅读

按这个顺序读Yue et al. (2025a) 与Wu et al. (2025a)，并且两篇都要读完，包括附录——采样预算与温度都藏在那里。第一篇是测量，第二篇是机制，而这一对是本文献中最好的一个例子，说明一个否定性结果如何改变了一个领域的评测标准，而不只是改变了排名。读它们时请在一条轴上格外存疑：每一条关于「基座模型边界」的主张，都是关于某个采样预算与某种提示格式的主张，而这两篇论文在这件事上都比引用它们的人要谨慎。

然后读Zhu et al. (2025b)，它是本章中机制上最令人满意的论文。它把覆盖率损失定位到一个你已经推导过的估计量的正优势项上，而它那个仅用负样本的训练结果，正是那种会让你回头重读自己损失函数的发现。请把它与第12章中的Cui et al. (2025b) 配对阅读，因为熵律提供了 PSR/NSR 分解所花掉的那份预算约束。

对于 2026 年的调和，PASS@(k, T) 分析 (PASS@(k,T) Analysis, 2026) 是首先该读的：它的指标是一项真正的方法学贡献，而它那个数据匹配的 SFT 对照，正是 2025 年

文献所缺的那类比较。Tang et al. (2026) 应当为其两阶段主张与其前沿规模的 KL 结果而读，同时记住那是一次没人能复现的规模上的单次运行。Shen et al. (2026a) 是三者中最不寻常的一篇——用国际象棋来获得语言语料上不可能获得的受控预训练变化——即便撇开它的结论，也值得作为一件实验设计作品来读。

如果你只有时间读一篇，那就读Zhou (2026)，因为它的提示分类法可以立刻使用，并且会改变你构建评测集的方式。在目标函数这个问题上，Walder and Karkhanis (2025) 提供估计量，Thrampoulidis et al. (2025) 提供容纳它们的框架；反对的一方读Yu (2025)，篇幅很短，也值得认真对待。最后，Shenfeld et al. (2025) 是关于「本章通篇批评的那份保守同时也是一项资产」的最佳单篇论文，而Kim et al. (2025) 是那场把权衡讲具体了的正面对比。

习题

14.1（支撑集保持有多弱？） 从(6.15)推导出 (14.1)，包括对 $Z(x)$ 的界。然后对二元奖励，计算 $\beta \in \{1, 10^{-1}, 10^{-2}, 10^{-3}\}$ 时所允许的最大放大倍数。确定使该界 禁止把一个基座概率为 10^{-9} 的回复抬高到概率 $\frac{1}{2}$ 的最大 β。最后，精确说明定理 14.1 and 14.3中的三个定义里，哪些受这个界约束、哪些不受。

14.2（曲线在哪里相交。） 把基座模型下的逐提示成功概率建模为 $\mu \sim \mathrm{Beta}(a, b)$，于是 $\mathrm{pass@}k = 1 - \mathbb{E}_\mu[(1-\mu)^k]$；再把 RL 建模为锐化映射 $\mu \mapsto \mu' = \mu^{1/\mathcal{T}} / (\mu^{1/\mathcal{T}} + (1-\mu)^{1/\mathcal{T}})$，其中 $\mathcal{T} < 1$（如 第2章习题 6）。用闭式或数值方法计算两条 pass@k 曲线，并（a）求出交叉点 $k^\star$ 作为 $\mathcal{T}$ 的函数，取 $(a, b) = (1, 4)$；（b）证明当 $\mathcal{T} \to 1$ 时 $k^\star \to \infty$；（c）找出一组 (a, b) 使得在任何 $k \le 10^4$ 处都不发生交叉，并说明这在表 14.3的分类法中对应于提示总体的什么性质。

14.3（pass@k 权重，以及它与什么相竞争。） 利用 节 14.7.1，对 $k \in \{1, 4, 16, 64\}$ 画出 $w_k(p) = k(1-p)^{k-1}$ 关于 $p \in (0, 1)$ 的曲线，并对每条曲线归一化使 $\int_0^1 w_k = 1$。在同一组坐标轴上画出逐组标准差归一化所强加的隐式提示权重 $1/\sqrt{\hat{\mu}(1-\hat{\mu})}$，见(12.2)。找出两种加权在梯度符号上不一致的那段 p 区间，并用它来预测：如果把 pass@k 目标与标准 GRPO 优势归一化结合起来会发生什么。然后设计一个退火时间表 $k(t)$，使得随着策略改进，落在 $p \approx 0.1$ 的提示上的 平均权重保持不变。

14.4（词元熵上升，答案熵下降。） 在一个三词元的词表上构造一个显式的两步生成过程，其中某次策略变化严格提高平均逐步条件熵，同时严格降低最终答案分布的熵。请

给出具体数值。然后用一段话解释为什么长思维链轨迹会使这种配置成为典型情形而不是病态情形，并说明这对节 11.7的诊断量集合意味着什么。

14.5（边界提示被过滤掉了。）取(12.4)，对 $G \in \{8, 16, 64\}$ 与 $\mu \in \{0.005, 0.02, 0.05, 0.2\}$ 计算一组样本退化的概率。用表 14.3的分类法论证：动态采样（节 12.4）系统性地移除边界提示，同时保留可达提示。然后量化这笔交易：在固定的生成预算下，比较（a）$G = 8$ 加零方差过滤与补采，与（b）在按边界分层的子集上取 $G = 64$，比较的口径是每单位生成词元中获得非零纠正梯度的提示的期望数量。你会跑哪一种，又会测量什么来检验?

14.6 ⋆ **（把两个阶段与一种测量假象区分开。）** Tang et al. (2026) 关于发现先于锐化的主张，就其陈述而言，与一种纯测量侧的解释是相容的：训练早期策略是高熵的，因此相对于同一策略在其输出长度分布发生偏移之后再评测，大 k 处的 pass@k 估计被抬高了。请设计一个 1–8B 规模的实验来区分这两种说法。请指定检查点、采样预算、对回复长度与温度的对照、分层方式，以及——最重要的——什么样的结果会证伪那个两阶段主张。然后陈述你的实验排除不了什么。

14.7 ⋆ **（把质量播撒到有效支撑集之外。）**提出一种纯 RL 的机制，使其在没有教师模型的情况下，把不可忽略的概率放到策略当前有效支撑集（定理 14.3）之外的轨迹上。候选配料：仅在采样阶段修改温度或 top-p、从某个搜索过程注入前缀、自生成的题目变体，以及参考重置。对你的方案，（a）用Thrampoulidis et al. (2025) 的优势塑形视角判定所得到的更新是否仍是 某个目标函数的合法策略梯度；（b）论证被播撒的质量是能在后续优化中存活，还是会作为异策略噪声被清除——注意在不同温度下抽取的采样轨迹是异策略的，需要 节 2.6的修正；（c）陈述什么样的实验能把你的机制与一个朴素的熵奖励区分开，而节 2.5说后者是会失败的。

14.8 † **（把这套流程跑一遍。）**在两个互不相关的、1–8B 的基座模型族上端到端地实现节 14.9的流程。交付物：用代码 14.1 的估计量算出的、直到 $k = 256$ 的基座 pass@k 曲线；四路分层；每个模型族一次用节 11.7默认配置的 GRPO 运行，并沿算力轴至少存五个检查点；每个检查点上分层的 pass@1 与 pass@k；答案级熵；以及带种子方差的 $k^\star$。报告你是否观察到（a）是否存在交叉，（b）图 14.2那种两阶段形状，（c）倒挂是否集中在边界层，以及（d）是否存在对模型族的依赖。请说明你的运行检验了 表 14.2中的哪些证伪条件，又检验不了哪些。

14.9 † **（复现 PSR/NSR 的分离，然后用它修好一次运行。）**实现 Zhu et al. (2025b) 的分解：从同一个检查点出发，在算力匹配的条件下训练三个策略，分别用（i）完整

GRPO，（ii）仅正优势，（iii）仅负优势。为这三者以及基座模型画出 $k \in \{1, \dots, 256\}$ 的 pass@k。然后取倒挂最严重的那次运行，在代价相同的前提下尝试两种修补：对正项做 $\lambda = 0.1$ 的降权，以及只施加于边界层的、依赖提示的基座锚点（节 14.6）。报告哪一种修补在每牺牲一个点的 pass@1 时能挽回更多覆盖率，以及这一排序在两个模型族上是否相同。

第 15 章

超越可验证领域

核心思想：可验证性是你的工具链的属性，而你可以制造出更多可验证性

第 III 部分的每一项结果，都建立在一个能为答案出具证明的程序之上。大多数值得做的任务并没有这样的程序，而一个被优化了数千步的学出来的奖励，不仅仅是一个更弱的验证器，它是另一类东西：一个其误差正被优化器主动搜寻的估计量。可行的应对是把那个无法证伪的问题分解成许多个近乎可证伪的问题，并把其中尽可能多的交给一个程序去做。这确实换来了实地，却并没有逃脱 Goodhart 定律；它只是把定律从一个你无法检查的标量，挪到了一份你可以检查的判据清单上。这是全书中最未定论的材料，下面各节会说明哪些部分已被复现、哪些部分只有一篇论文那么新。

第11章定义了验证器（定理 11.1），并陈述了使 RLVR 得以奏效的那笔交易：用标注噪声换取设定误差，而这笔交换是划算的，因为设定误差是看得见的。此后的三章都在「奖励是可靠的」这一假设下改进优化器。本章移除这一假设，而答案将决定第 III 部分描述的究竟是一项属于数学与代码的技术，还是一项属于后训练一般情形的技术。

15.1 可验证性谱系

把「可验证」当成一个领域的二元属性，是第一个错误。可验证性属于一个领域连同某人为它搭建的机制，它是程度问题，而且这个程度是可测量的。有用的测量所问的是：奖励中有多大一部分是你愿意为之辩护的。

定义 15.1 (可辩护份额)**.** 把一个奖励写成若干分量的加权和 $r = \sum_j \lambda_j r_j$，并令 $\mathcal{D}$ 索引其中那些由某个程序计算、而你愿意在法庭上为其输出辩护的分量：逐行辩护，面对

表 15.1: 按(15.1)的可辩护份额 φ 排序的可验证性谱系。中间那一段几乎装下了所有有价值的工作，而它的定义性特征是：程序确实证明了某个真实的东西，但那个东西比目标要窄——测试证明的是行为，对可维护性只字不提；一条能执行的查询证明的是语法与一个行集，对问题是否被理解只字不提。顺着表往下读，主导失效从「我的判据不是那个目标」转变为「我的估计量正在被搜寻弱点」。

段位	代表性任务	奖励来源	主导失效模式
逐位精确，$\varphi = 1$	算术、形式化证明、模式（schema）符合性、谜题	判定过程：证明内核、解析器、模拟器	判据是可靠的，而它不是目标
近乎精确，$\varphi \approx 1$	竞赛数学、竞技编程	答案等价性判定或隐藏测试，加上一份精心整理的参考答案	不可靠（答案对、推理错；专门迁就测试的补丁），以及在等价但未匹配上的答案上的假阴性
部分可校验，$0 < \varphi < 1$	文本转 SQL、代码仓补丁、指令遵循、带引用的摘要	在可校验投影上用程序，在残差上用模型	可校验的那部分很早就饱和，质量随之迁移到程序看不见的地方
可评判，$\varphi \approx 0$	解释质量、建议、语气、编辑判断	带模型评分器的评分细则，或一个标量评判模型	回归型、极值型与因果型 Goodhart（表 5.4）；既可达又不可见的盲区（定理 15.3）
不可验证，无代理指标	研究品味、某条建议是否改善了一个人的生活、有争议的规范性问题	在一次训练运行的范围内无可用来源	不可训练；声称训练了它的说法，都是关于某个代理指标的说法

一位读过源码、正在找反例的敌意专家。可辩护份额定义为

$$\varphi \;=\; \frac{\sum_{j \in \mathcal{D}} \lambda_j \left(\max r_j - \min r_j\right)}{\sum_{j} \lambda_j \left(\max r_j - \min r_j\right)} \;\in\; [0,1], \tag{15.1}$$

即奖励总取值范围中由一个可辩护程序所提供的那一份。

这个判据被刻意定得偏社会性而非数学性，因为真正要紧的属性不是可证明性而是 失效的可枚举性：一个程序，只要你能把它的假阳性与假阴性一一列举出来，你就能为它辩护；而一个你必须自己去发现其错误的程序，你就无法为它辩护。而且 φ 描述的是你实际发布的那个奖励而不是那个任务，所以提高它是工程问题，而不是挑一个更容易的领域的问题。

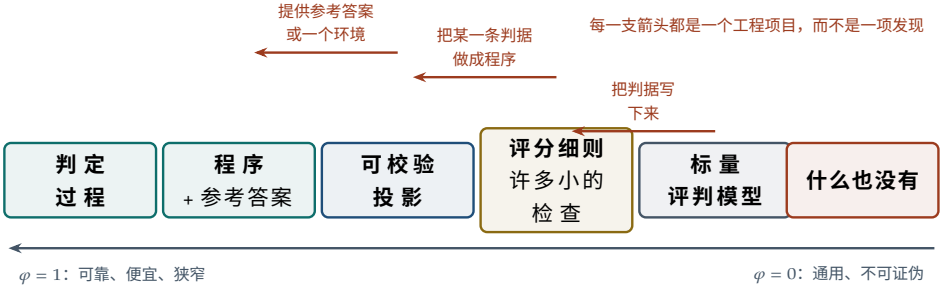


图 15.1: 把表 15.1的谱系画成一根轴。位置是序数性的；这里没有任何量化刻度。图的内容在于那几支箭头：任务是被某人搭建机制拖向左侧的，而 2022 年以来后训练的历史，很大程度上就是一部这类挪移的历史。一次偏好判断变成了一份写下来的评分细则，一条细则判据变成了一个正则表达式，一个正则表达式变成了一套测试，一套测试变成了一个沙箱环境。第 IV 部分把最右边那支箭头一次性施加到整个任务上。

这张表把通常被混作一谈的两段分开了，而这一区分是值得的。在近乎精确的那一段，程序 就是规格说明，而有意思的缺陷是它的假阴性，这在第11章中已经处理过。在部分可校验的那一段，程序是规格说明的一个投影，动力学也随之不同：这个投影会在几百步之内饱和，此后所有剩余的奖励方差、因而整个梯度，都来自基于模型的那部分残差。一次运行若在可验证分量已经饱和之后其记录的奖励仍在攀升，那么按构造它优化的就是奖励中你无法辩护的那部分。

中间这一段之所以存在，是因为有人把它开凿了出来。 最干净的存在性证明是指令遵循。「遵循指令」这件事在一般情形下没有任何可判定之处，但 IFEval 观察到，真实请求中有很大一部分至少包含一条短短一个 Python 函数就能精确判定的约束：字数、禁用字符、大小写、必需的章节、输出格式 (Zhou et al., 2023b)。这个投影在它自己的口径上既可靠又完备，而且只花几微秒，这就是为什么 Tülu 3 把程序化约束放进了它的 RLVR 混合数据 (Lambert et al., 2025)，也是为什么 RewardBench 2 用验证器函数而不是评判模型来给精确指令遵循打分 (Malik et al., 2025)。这个选择产生了本章最有用的一个数字：在那个子集上，据Malik et al. (2025) 报告，学出来的奖励模型与强评判模型的得分大致在 37% 到 66% 之间，而随机基线是 25%，这是该基准中差距最大的、也是最弱的一列。这是唯一一处让一个学出来的奖励与一个可靠的程序在完全相同的输入上被同时打分的地方，而程序在此处的优势幅度，应当校准我们在其他所有无法做这种比较之处的预期。

最后那一段不是修辞。有些目标在一次训练运行的时间尺度上根本没有真值的供给方——一个研究方向是否值得追求、一条建议是否改善了一个人的生活——而再多的评分细则工程也无法把它们转化成奖励。下文提供的是代理指标，同时附上一条纪律：把每个代理指标看不见什么都枚举出来。

15.2 评分细则：用分解来制造一个验证器

节 5.6已经把评分细则放进了奖励信号的分类体系。在这里它们是核心技术，而问题是机械性的：分解为什么有帮助，它的代价又是什么？

定义 15.2 (评分细则)**.** 针对提示 x 的一份评分细则是一个有限集合 $\mathcal{R}(x) = \{(c_k, w_k, \kappa_k)\}_{k=1}^{K}$，其中每个 c_k 是一条关于回复的、意在可判定的陈述，$w_k > 0$ 是权重，$\kappa_k \in \{\text{soft}, \text{hard}\}$ 记录该判据是计入分数还是充当一道闸门。一个评分器为每条判据给出分数 $s_k(x, y) \in [0, 1]$：有程序可用时用程序，否则用一个评判模型 π_{J}。

几乎所有已发表系统所计算的奖励，都是各条软判据的加权平均，再由各条硬判据以乘法方式把闸：

$$r_{\mathcal{R}}(x, y) = \underbrace{\prod_{k:\,\kappa_k=\text{hard}} \mathbb{1}[s_k(x, y) = 1]}_{\text{gate}} \cdot \underbrace{\frac{\sum_{k:\,\kappa_k=\text{soft}} w_k\, s_k(x, y)}{\sum_{k:\,\kappa_k=\text{soft}} w_k}}_{\text{graded score in } [0,1]}. \tag{15.2}$$

(15.2)里藏着两个选择，两者都已被消融过：评判模型要么看到整份细则并返回一个分数（隐式聚合），要么每条判据调用一次（显式聚合）；而 s_k 要么是二元的，要么是分级的。Gunjal et al. (2025) 报告说两种聚合都可行，而显式那种能提供逐判据的信号，节 15.5会说明那是提前看出作弊苗头的唯一便宜办法。

推导: 对 K 条判据取平均能消掉噪声，消不掉共有偏差

设一个整体式评判模型返回 $r^\star + \epsilon_0$，其中 $\mathbb{V}\text{ar}[\epsilon_0] = \sigma^2$。设逐判据评分器返回 $s_k = s_k^\star + \eta_k$，其中 $\mathbb{V}\text{ar}[\eta_k] = \sigma^2$、$\mathbb{E}[\eta_k] = b$，且对 $j \neq k$ 有 $\text{Cov}[\eta_j, \eta_k] = \rho\sigma^2$，这里 $\rho \geq 0$ 是因为同一个模型为每条判据打分而共有的那部分误差。在等权重下，(15.2)的聚合值满足

$$\mathbb{V}\text{ar}[r_{\mathcal{R}}] = \frac{\sigma^2}{K^2}\big(K + K(K-1)\rho\big) = \sigma^2\Big(\rho + \frac{1-\rho}{K}\Big), \qquad \mathbb{E}[r_{\mathcal{R}} - r^\star_{\mathcal{R}}] = b. \tag{15.3}$$

有三种读法，每一种都是承重的。在 $\rho = 0$ 时方差是 σ^2/K，就是熟悉的 K 倍。在 $\rho > 0$ 时存在一个 $\rho\sigma^2$ 的地板，再多的判据也去不掉，而且可获得的削减大部分在早期就到手了：在 $\rho = 0.3$ 时，从 $K = 1$ 增加到 5 消掉了 56% 的方差，而从 5 增加到 20 只再消掉 11%，所以一份五十条判据的细则买到的是可审计性而不是信号。而偏差 b 则完全不依赖 K：如果每条判据都由一个偏爱冗长、结构化、语气笃定的文字的模型来打分，那么每个 η_k 都携带这一偏好，取平均只会原封不动地把它还回来。因此作用于 ρ 与 b 的杠杆不是更多判据，而是不同的评分器——有程序的地方用程序，没有的地方用另一个模型族。这就是Eisenstein et al. (2024)的奖励集成结果换了一身衣服：起约束作用的是共有误差，而不是误差的大小。

除方差之外，分解还做到了三件标量做不到的事。更窄的问题会被回答得更准确：「这是一份好的出院小结吗」要求评判模型在脑中装下一整套诊疗标准，而「它是否写明了出院用药及剂量」几乎就是一项抽取任务。这一机制预言了一个特征——评分细则应当对 弱评判模型帮助最大，因为强评判模型本来就装得下那套标准——而 Gunjal et al. (2025) 报告的恰恰如此，并且伴随着跨评判模型规模的方差下降；这是该领域最有用的证据，因为它区分的是机制，而不只是演示一次增益。分歧被定位了：评分细则评分器与人类在第七条判据上有分歧，这可以被计数、重写、重新加权，或者交给一个程序；而一次整体式的分歧什么可供检查的东西都不留下。这使目标函数成为一件可编辑的物件，也是评分细则成为领域专家能够直接贡献的第一种奖励形式的原因。利用某一条判据的收益是有界的：一次能虚假地满足判据 k 的作弊，至多得到 $w_k / \sum_j w_j$，所以在二十条可比的判据下，一次利用只买到百分之五。

那个界就是这项技术的全部防线，而它只有在各判据是可被独立利用时才成立。一份所有判据都在奖励结构性、周密性与笃定语气的细则，可以被一次攻破并处处满足；它的有效 K 接近一，它就是一个披着清单外衣的标量奖励。这一警告的可测量形式是各条判据得分在采样轨迹上的相关矩阵，那是对(15.3)中 ρ 的直接估计，而几乎没有人记录它。

15.2.1 谁来写评分细则，以及它为什么会过期

评分细则的来源问题，如今是非可验证 RL 中主要的数据工程问题。人类专家提供的是效度，却无法扩展，所以手写的判据往往笼统，而策略只满足其字面。仅从提示出发做模型生成是免费的，但会产生两类特征性缺陷：不可判定的判据（「该分析富有洞见」），评判模型会悄悄把它变成一项文风奖励；以及预设性的判据，它断言答案应当包含什

么，并在生成器猜错时惩罚正确的回复。从参考答案出发生成能提高可判定性，却重新引入了 第11章中的不完备性失效，因为此时评分细则编码的是某一份参考答案的选择。2026 年的文献直接攻击自生成与人写之间的差距 (Sun et al., 2026)，并让生成器在若干模拟角色间多样化，这是一项作用于 ρ 的干预 (Fu et al., 2026)；另有一条平行的路线取消了单独的评分器，改用评分细则来给自蒸馏赋予结构 (Rezaei et al., 2026; Gu et al., 2026)，这条路便宜，而截至 2026 年年中，它也是该领域被独立复现得最少的部分。

更深一层的要点是：一份评分细则就是一个乔装打扮的学出来的奖励模型。它把回复映射到分数，是在对任务的一个有限视角上拟合出来的，并且由一个带有系统性误差的模型来求值，所以节 5.8关于学出来的奖励所说的一切都适用——包括它的效度有一个保质期，而这个保质期是以相对于它所针对的那个策略的 KL 来衡量的，而不是以挂钟时间来衡量的（节 5.9）。一份在第零步就冻结的评分细则，就是一个永不重训的奖励模型。这正是 2026 年那批共演化文献的共同前提，其中评分细则生成器会针对策略当前的高奖励行为重新训练：以对抗方式 (Ding et al., 2026)、通过联合训练评估器与策略 (Wang et al., 2026a)、在智能体轨迹而不是输出上进行 (Tian et al., 2026)、在测试时且不用训练集 (Wang et al., 2026b)，以及通过制造检验一条判据所需的成对证据 (Yang et al., 2026a)。这一构造把一次优化转变成了一个两人博弈中的交替最佳响应，而后者一般没有不动点，并且可能循环：策略利用判据 k，封堵 k 的那次修改打开了 j，而封堵 j 的那次修改又重新打开了 k。这在结构上正是 节 9.5用混合策略与正则化均衡所处理的那类对象，而这批文献尚未引入那套机制。请为循环装上监测，而不要假定单调改进。

代码 15.1: 带逐判据日志的评分细则打分，实现了 (15.2)。有两个细节才是重点。判据带有一个可选的 `program`，所以可辩护的那些判据会绕开评判模型，对(15.3)中的 ρ 与 b 都不产生贡献；而硬判据以乘法方式把闸，所以一个违反了显式约束的回复无法靠文风把损失赚回来。返回的逐判据表格是 节 15.5中每一项诊断的输入。

```python
from dataclasses import dataclass
from typing import Callable, Optional

@dataclass
class Criterion:
    text: str                                  # 原样展示给评判模型
    weight: float = 1.0
    hard: bool = False                         # 把闸，而不是计入分数
    program: Optional[Callable] = None         # 可辩护的评分器，如果存在的话

```

```python
def score_rubric(prompt, response, rubric, judge, n_vote=1):
    """Reward in [0, 1] plus the per-criterion table. Never log only the reward."""
    scores, num, den, gate_open = {}, 0.0, 0.0, True
    for c in rubric:
        if c.program is not None:
            s = float(c.program(prompt, response))
        else:
            # 在 n_vote 个样本上取 p("yes") 的均值：便宜的校准手段，而且样本
            # 之间的离散程度是一个可用的逐判据不确定度。
            s = judge.p_yes(prompt, response, c.text, n=n_vote)
        scores[c.text] = s
        if c.hard:
            gate_open &= (s >= 1.0)
        else:
            num, den = num + c.weight * s, den + c.weight
    graded = num / den if den > 0 else 0.0
    return (graded if gate_open else 0.0), scores
```

15.3 把评判模型当作奖励函数

用于评测的评判模型与用作奖励的评判模型是同一件物件处在两种情形中，两种情形只在一点上不同，而那一点压倒了其余一切。

15.3.1 逐点、成对，以及两者各自的偏差

一个逐点评判模型为单个回复打分，这名义上正是 RL 所需要的，因为奖励是单条采样轨迹的函数。一个成对评判模型指出两者中较好的那一个，这是评判模型回答得最可靠的问题，而且它不需要任何标度。两者在一个采样组内部得以调和：对一个提示的 G 条采样轨迹跑一场循环赛，令奖励为胜率 $r_i = w_i/(G-1)$。于是 定理 5.1中那个不可辨识的逐提示偏移量被精确地而不是近似地抵消掉，奖励取 G 个不同的值因而采样组很少退化（节 12.4），而完整循环赛下的平均胜率恰好是 $1/2$，所以组基线无需估计。代价是 $\binom{G}{2}$ 次评判模型调用，可以通过抽取 m 个对手降到 $O(Gm)$；另一项代价是，一个不满足传递性的评判模型会给出定义良好、却根本不对应任何标量质量的胜率（定理 5.2）。

节 5.5把评判模型的偏差列为测量问题；表 15.2把它们重列为优化靶子。此时要紧的不是一项偏差有多大，而是当策略因它而获得奖励时会做出什么；而诊断必须是一项策略未曾被训练去对抗的测量。

最后一行抬高了人们对流水线中每一个评判模型的信心。一项 2026 年对二十一个评判模型的审计报告说，与人类的精确匹配一致率把经过机遇校正的一致性高估了数十个百分点，而且评判模型的排名在不同基准之间不可迁移，所以一个在某个基准上被挑中的评判模型，其实并没有为任何事情被挑中 (Anonymous, 2026b)；一个配套的测评框架报告说，在自由回复任务上的可靠性并不能预测在智能体轨迹上的可靠性 (Anonymous, 2026c)。Li et al. (2025a) 补上了污染的对应物：当同一个强模型既合成了你的训练数据、又来评判你的输出时，它会偏爱自己的学生。由此有两条推论：使用一个既不同于策略、也不同于数据生成器的模型族的评判模型；并且把一个评判模型完全排除在流水线之外，好让至少一项测量仍然算是证据。

15.3.2 为什么一个够格的评估者可以是一个不够格的奖励

这是本章最重要的一处不对称，而且它不是程度问题。

命题 15.3 (评测是平均情形；优化是最坏情形)**.** 把评判模型的奖励写作 $r_{\pi_{\mathrm{J}}} = r^\star + \epsilon$，其中 $|\epsilon| \le 1$，并令 $\eta = \mathbb{E}_{y\sim\pi_{\mathrm{ref}}}[\epsilon]$ 为它在参考分布上的平均误差——也就是一次评测所估计的那个量。设某个集合 $S \subseteq \mathcal{Y}$ 满足 $\pi_{\mathrm{ref}}(S) = p > 0$ 且对所有 $y \in S$ 有 $\epsilon(y) \ge \Delta$。那么

1. 评测对 S 几乎是盲的：$\eta \ge p\Delta - (1-p)$，因此 $\Delta \le (\eta + 1 - p)/p$，这个界以 $1/p$ 的速度恶化；并且
2. 这个盲区很容易抵达：把 π_{ref} 条件在 S 上所得到的策略 π_S 满足 $\mathbb{D}_{\mathrm{KL}}(\pi_S \,\|\, \pi_{\mathrm{ref}}) = \log(1/p)$ 以及 $\mathbb{E}_{\pi_S}[r_{\pi_{\mathrm{J}}} - r^\star] \ge \Delta$。

因此 δ 奈特的 KL 预算就足以暴露参考概率不低于 $e^{-\delta}$ 的每一个盲区，而评判模型实测的准确率只能通过 $1/p$ 这个因子来约束这样一个盲区的严重程度。

证明. 对于 (2)，$\pi_S(y) = \pi_{\mathrm{ref}}(y)\mathbb{1}[y \in S]/p$，故 $\mathbb{D}_{\mathrm{KL}}(\pi_S \,\|\, \pi_{\mathrm{ref}}) = \mathbb{E}_{\pi_S}[\log(1/p)] = \log(1/p)$；又因为在 S 上逐点有 $\epsilon \ge \Delta$，所以 $\mathbb{E}_{\pi_S}[\epsilon] \ge \Delta$。对于 (1)，利用 $\epsilon \ge -1$ 写出 $\eta = p\ \mathbb{E}[\epsilon \mid S] + (1-p)\ \mathbb{E}[\epsilon \mid S^c] \ge p\Delta - (1-p)$，再整理即可。 □

这条命题解释了一件实践者反复重新发现的事实：一个与人类的一致性好到足以给三个检查点排序的评判模型，可能远远弱到无法拿来训练两千步。一个评估者的输入抽自一个它没有选择的分布，所以它的误差会相互平均，只剩下 η，并以评测条目数的 $N^{-1/2}$ 速率收敛。而一个优化器的输入是被挑选来最大化该评判模型的，所以起支

配作用的量是 $\sup\{\mathbb{E}_{\pi}[\epsilon] : \mathbb{D}_{\mathrm{KL}}(\pi \,\|\, \pi_{\mathrm{ref}}) \le \delta\}$，而命题表明它是由评测看不见的那些稀有区域所控制的。三条推论。所报告的评判模型准确率并不是一道安全余量。KL 预算才是买到盲区的那份资源，这就是为什么在有一个可靠验证器时 $\beta = 0$ 是站得住脚的，而在用评判模型时它是鲁莽的（节 12.8）。而且本章的每一项技术——评分细则、集成、把闸、弃权——起作用的方式都是缩小可达集合，或者抬高残余盲区的 p，而绝不是消除它们。

代码 15.2: 一个受控偏差的成对评判模型。每次比较都按两种顺序各跑一遍，翻转被记为平局而不是被裁决，于是位置偏差退化为信息损失，而不是被注入的噪声；翻转率会被记录，因为它是该评判模型的噪声地板，并且会随策略移动而漂移。参考锚定是可选的，而且便宜：一个固定的参考回复能把一次绝对判断转化为一次与跨检查点稳定之物的比较。

```python
def pairwise_win_prob(judge, prompt, a, b, reference=None, n=1):
    """P(a > b) with position bias cancelled and flips reported as ties."""
    wins, flips = 0.0, 0
    for _ in range(n):
        ab = judge.verdict(prompt, first=a, second=b, reference=reference)
        ba = judge.verdict(prompt, first=b, second=a, reference=reference)
        # verdict 取值于 {"first", "second", "tie"}；当且仅当同一个回复在两种
        # 排序下都获胜时才算一致。
        if ab == "first" and ba == "second":
            wins += 1.0
        elif ab == "second" and ba == "first":
            pass
        else:
            wins += 0.5
            flips += 1
    return wins / n, flips / n          #  (胜率，位置不稳定率)

def group_win_rates(judge, prompt, responses, m, rng, reference=None):
    """Sampled round robin: O(G*m) calls, mean win rate exactly 1/2."""
    G = len(responses)
    won, played, unstable = [0.0] * G, [0] * G, 0.0
    for i in range(G):
        for j in rng.sample([k for k in range(G) if k != i], m):
            p, f = pairwise_win_prob(judge, prompt, responses[i], responses[j],
                                     reference=reference)
            won[i] += p; won[j] += 1.0 - p
            played[i] += 1; played[j] += 1
            unstable += f
    rates = [w / c if c else 0.5 for w, c in zip(won, played)]
    return rates, unstable / max(1, G * m)
```

15.3.3 审议式评判，以及它的代价

节 5.6描述了从一个标量输出头转向一个先推理再打分的模型的这一挪移，以及用 RL 训练评判模型本身的那条路线 (Zhang et al., 2025c; Chen et al., 2025d; Liu et al., 2025i; Guo et al., 2025a; Whitehouse et al., 2025)。透过定理 15.3来看，审议干预的是盲区的位置而不是它们的存在：一个会写出理由的评判模型更难被表层特征骗过，而对于其理由系统性略去的那些东西，它依然可被骗过。成本这一侧在 RL 循环内部是不留情面的。一份有 K 条判据的细则、一组 G 条采样轨迹，以及一个每次判决输出 L_{J} 个词元的评判模型，每个提示要花 $G\,K L_{\mathrm{J}}$ 个评判词元，而采样轨迹本身只有 GL 个策略词元；所以在 $K = 10$、评判模型与策略一样大时，奖励可能比采样轨迹还贵（第23章）。有三种标准的缓解手段：把若干判据批进一次调用并解析一份结构化判决，接受这样做所引入的相关性；只对承载判断的那两三条判据保留审议；以及做级联，先便宜地给所有东西打分，再只对每组的两端重新打分，因为组相对优势主要取决于尾部。

15.4 自我奖励与生成–验证差距

如果策略能给自己打分，奖励就是免费的。节 9.3描述了自我奖励及其自对齐的近亲 (Yuan et al., 2024a; Bai et al., 2022b)；这里的问题是：给自己打分在什么时候携带信息，又在什么时候只是一个不携带任何信息的不动点。

定义 15.4 (生成–验证差距)**.** 固定一个带标签的探测集。令 a_{g} 为一个被采样出来的回复正确的概率，a_{v} 为该模型对自己所生成回复给出的判决的准确率。生成–验证差距定义为

$$\Delta_{\mathrm{gv}} \;=\; a_{\mathrm{v}} - \max\left\{a_{\mathrm{g}},\, 1 - a_{\mathrm{g}}\right\}, \tag{15.4}$$

即验证者的准确率超出最好的常数判决的那一部分。

(15.4)中的那个减法就是这个定义的全部内容。一个能解出任务 30%、而验证准确率为 70% 的模型，其 $\Delta_{\mathrm{gv}} = 0$：「一律说错」就能达到同样的水平，而一个具有这种结构的奖励只教会了一种偏置，别无其他。凡是不减去常数基线的自验证准确率报告都是无法解读的，而这是该领域最常见的错误。

即便 $\Delta_{\mathrm{gv}} > 0$ 也仍然不够，因为策略梯度所消耗的是验证者在策略自身置信度条件下的准确率。按序列似然把探测集分组，并在每个分组内部计算 a_{v}。要紧的那个分组是高似然且错误的那一组：那些正是优化器将会更多产出的回复，而它们也恰恰是从同一个条件分布算出的判决会说「正确」的那些回复。这正是自我奖励可能是一个不

含信息的不动点的确切含义——模型恰恰在它自信地错了的那些情形下无法看出自己错了——也是为什么由置信度或负熵构建的内在奖励，会被自信的错误所最大化 (Zhao et al., 2025d; Prabhudesai et al., 2025)。明确的反面意见是：内部反馈并非一场免费的胜利 (Anonymous, 2025)；而且既然随机指派的奖励能在 Qwen 模型上改善数学 (Shao et al., 2025)，那么一个在 Qwen 模型上得到的内在奖励结果，在它于一个无关模型族上复现之前都算不上证据。

因此，一处真正的不对称要求验证者去查阅某种生成时没有查阅的东西，而安排出这一点的每一种办法都是一项构造而不是一项发现：执行，判决来自运行那件产物而不是相信它 (Jin et al., 2025b)；一个独立训练的校验器，例如让一个单元测试撰写者与写代码的模型一同训练，使它的错误是它自己的 (Wang et al., 2025d)；在验证时投入更多算力，这正是审议式评判模型不只是一次重参数化的原因 (Liu et al., 2025i)；以及 对生成隐藏的特权信息，比如把某个事实从提示中遮掉、而在打分时恢复。多数投票伪标签处在边界上：自洽性 (Wang et al., 2023a) 被转化为训练信号，Zuo et al. (2025) 在测试时演示了这一点，而它的天花板是众数答案的准确率，所以它恰恰会在你真正在意的那些困难提示上强化被自信持有的错误 (Shafayat et al., 2025)。

工程实践：用一个下午在你自己的任务上测出 Δ_{gv}

取 300 条带可信标签的提示，每条采样 $G = 8$ 个回复，并记录正确性。在一个没有「自己写过这段」记忆的独立上下文中，让模型对每个回复给出判决，然后由(15.4)算出 a_{g}、a_{v} 与 Δ_{gv}。按回复似然分组，报告每组的 a_{v}，并同时报告限定在高似然错误回复上的准确率——那是唯一能预测自我奖励会做出什么的数字。最后，给验证者多加一项能力（执行、检索、一份写下来的检查清单、十倍的思考预算）后重做一遍，并报告变化。如果没有任何一项能力能推动那个条件数字，那么在这项任务上自我奖励就是锐化（第14章），并且应当据实报告。

15.5 没有验证器时的奖励作弊

节 5.8给出了分类体系。下面是一份实地手册，其形式是分类体系提供不了的：一个机制，加上一项能抓住它的便宜测量。

长度膨胀。在任何说得通的标签分布中长度都与质量相关，而它又是最便宜的可增加量，所以它是一个被评判模型训练出来的策略最先发现的东西 (Singhal et al., 2024)。诊断不是平均长度（它可能是合理上升的），而是奖励与长度受控胜率之间的背离 (Dubois et al., 2024)。

谄媚。人类评判者与模型评判者都会奖励认同，所以策略学会了去肯定用户的前提 (Sharma et al., 2024)。请用一个固定的、永不重新生成的错误前提与事后反驳追问探测集来检测它。

有排版没实质。标题、要点与加粗在评判模型的训练数据中与用心程度共同出现，于是一条「组织良好」的判据打的是标记的分。请给剥掉格式后的改写打分；如果奖励几乎不动，那么这条判据毕竟还是关乎内容的。

笃定断言，及其镜像——含糊其辞。一个惩罚错误的评判模型会奖励含糊，于是论断密度下降而奖励上升；一个奖励果断的评判模型会奖励不加限定的论断，于是校准变差。这是同一份评分细则的两种相反作弊，而一项测量就能把它们分开：每百词中可验证的断言数，并对照这些断言上的准确率来跟踪。

引用捏造。任何形如「以来源支撑其论断」的判据，都能被看起来像模像样的参考文献所满足，因为评判模型检查的是引用存在且格式良好，而不是它真的存在。这是本章中最清楚的一个「某条判据绝不能交给评判模型」的例子：请把每一个标识符都拿去一个索引中解析，并给解析成功率打分。一条由评判模型判定的引用判据就是一项捏造奖励。

来自模型自身输出的评判提示注入。如果采样轨迹被插入到评判模型的提示里，那么这条采样轨迹就是评判模型指令的一部分。策略会发现一些读起来像打分指令的字符串——宣称任务已完成、自我评估、排版得像评判模型自身脚手架的文本——而且这些字符串能跨提示迁移，所以这种作弊表现为奖励出现一次不连续跃升，而质量并无相应变化。检测办法是每个检查点都用当前模板给一个固定的参考集打分，并把采样轨迹与指令放在分开的、界限清晰的通道里。

常见陷阱：让评判模型读取模型自称的自我评估

让策略输出一段自我评估——一个置信分数、一份它给自己填好的检查清单、一节「验证」——再让评判模型使用它，这很诱人。在 RL 之下这不是额外的信息，而是一条从策略直通它自己奖励的通道。那些词元上的梯度是无人对抗的：目标函数中没有任何东西会惩罚「宣称所有判据都已满足」，所以策略学会了这么宣称，而评判模型作为一个指令遵循者，也就照办了。其特征是奖励上升，自我评估段落收敛为一个固定的、最大限度正面的模板，而回复的实质毫无变化。补救办法是结构性的，而不是措辞上的：在评判模型看到之前把自指内容剥掉，或者拿一个独立判决去核对这段自我评估，并奖励二者的一致而不是那段宣称。同样的论证

> 使得任何从策略自己撰写的产物上算出的奖励都值得怀疑，除非那件产物是被执行的而不是被阅读的。

有两条协议要点在这里比在有验证器时更加重要。必须被人读的是那些奖励最高的采样轨迹，因为一次成功的作弊按定义就是一个高奖励输出，所以均匀抽样会长时间错过它，而抽取前十分位则立刻就能找到（节 5.8.4）；在有验证器时前十分位枯燥无味，而在用评分细则时那里正是灌水、判据回声与清了闸却答非所问的东西所在之处。还要留意判据回声：把目标用自然语言写下来，等于告诉了策略目标是什么，而满足一个在找某个概念的评判模型，最短的路径就是把那个概念说出来。请跟踪采样轨迹与评分细则文本之间的 n-gram 重合度，并在重新打分之前把判据改写一遍。

15.6 混合奖励：先把闸，再评分

没有哪个真实配方是只对着一个奖励训练的。R1 的最后一个阶段把用于推理的基于规则的奖励与用于有用性和无害性的偏好奖励组合起来 (DeepSeek-AI, 2025)；Tülu 3 把程序化约束混进它的 RLVR 数据 (Lambert et al., 2025)；Kimi K2 把一个可验证奖励的训练场与一个用于创意写作等领域的自我批评评分细则奖励配对起来 (Kimi Team, 2025b)。混合是常态，而它的形式决定了那个可验证的分量究竟是在保护这次运行，还是只是给它做装饰。

推导: 为什么乘法把闸优于加法混合

设 $r_{\text{ver}} \in \{0,1\}$ 为一个可靠的分量，$q \in [0,1]$ 为一个基于模型的分数。比较

$$r_{\text{add}} = r_{\text{ver}} + \lambda q \qquad \text{against} \qquad r_{\text{gate}} = r_{\text{ver}} \cdot (\alpha + (1-\alpha)q), \quad \alpha \in (0,1). \tag{15.5}$$

要求满足排序保证：每个通过了可靠分量的回复都排在每个未通过的回复之前。对加法形式，这要求 $\lambda(\max q - \min q) < 1$，于是模型项的全部影响都被限制在小于验证器的那道落差之内，只能用来打破平局；而任何大到足以实质性移动奖励的 λ，也就大到足以在某处颠倒排序。因此加法形式没有可用的工作区间——要么模型项是装饰，要么可靠分量不占主导。

两种形式差别最大的地方是在未通过的采样轨迹集合上，而在训练早期那是大多数。在 r_{add} 之下这个集合仍然按 q 排序，于是组相对优势在「答错」的条件下奖励讨好评判模型的行为：策略被明确地教导，当它无法答对时就应当谄媚、冗长且笃定。在 r_{gate} 之下，未通过的集合是平的，其集合内部朝 q 的梯度恰好为零，而

> 评判模型永远只给已经通过的回复排序。用定理 15.3的语言说，把闸把可被评判模型利用所抵达的集合从 $\mathcal{Y}$ 缩小到 $\{y : r_{\text{ver}}(y) = 1\}$，而后者恰恰是那个已经让评判模型的盲区变得代价高昂的集合。
>
> 代价是真实的，必须为它做预算：把闸会摧毁全部未通过的那些采样组中的信号，它们的优势会恒等于零。有两种补救能让它保持可用。把闸建立在一个便宜而可靠的前置条件上——格式有效性、显式约束、可解析的引用——而不是建立在整项任务的成功上，这样闸门就能在几百步之内对大多数采样轨迹打开；并且保留一部分提示不把闸，作为一条探索通道，并单独记录它们的贡献。常数 α 设定了一个通过的回复的底值，因而也设定了在通过的那些回复之间，正确性相对于质量的权重。

随后有三项账目管理决定一次混合运行是否说明得了任何问题。逐来源的尺度归一化：一个二元验证器的取值范围是一，一份评分细则的经验离散度可能只有 0.1，而一个 Likert 评判模型跨越一个整数区间，所以把它们喂进同一次优势计算，等于是按各领域的标准差、而不是按你做过的任何决定来给领域加权；请按来源对着滑动统计量做标准化，并设一个方差下限，或者在评判分数长尾时在批内做秩归一化。逐领域的批统计量：组相对估计量本来就在一个提示内部做了中心化，所以问题只出现在某个实现按批级标准差做除法之处（节 12.3）——请按领域来汇集那个统计量，否则一个高方差的领域会压缩其他每个领域的梯度。逐领域的 β：可验证领域常常在 $\beta = 0$ 下运行（节 12.8），因为奖励不可能在它自己的判据上出错；而基于评判模型的领域出于定理 15.3所给的理由需要那个锚点，一个单一的全局系数要么扼制推理上的增益，要么让学出来的奖励不受约束。最后，只画逐领域的曲线：在异质来源上做出来的聚合奖励曲线几乎毫无意义。

15.7 半可验证构造

这一领域中最有成效的一步不是把评判做得更好，而是重新定义任务，使一个程序能给它打分。有五种构造反复出现。每一种都在某个代理指标上换来了可靠性，而使它们值得尊重的那条纪律，是在训练之前而不是之后，把该代理指标看不见什么枚举出来。

逆向任务生成。从答案生成题目，并奖励解出它；答案按构造就是已知的，所以验证器是精确的。那些自行提出任务并通过执行来验证的自博弈系统是其成熟形态 (Zhao et al., 2025a)。其代理失效是任务生成器朝着「容易构造且容易验证」的题目漂移，而

那些题目未必像你真正关心的任务；请通过把生成的任务拿去与一个留出的人工撰写任务集对比打分来测量它。

往返一致性。奖励一次变换的可逆性：先摘要再重构，先翻译再回译，先抽取一条结构化记录再重新生成文本。把重构结果与源文本相比较是精确的。有两种代理失效：逐字保留能把可逆性最大化，所以奖励是在为抄写付钱；以及两个方向共有的失效会相互抵消，因为一次被一致地反向还原的系统性误译会得满分。请给压缩比设上限，并使用一个独立训练的逆向模型。

若干独立求解器之间的一致。奖励在不同模型、不同提示或不同工具路径之间的一致，这些路径的失效方式各不相同。这就是多数投票 (Wang et al., 2023a; Zuo et al., 2025)，只是把重点挪到了独立性上，因为一致的信息量正比于误差有多不相关——与 (15.3)中的 ρ 是同一个量。其代理失效是共有先验带来的一致：出自同一预训练血统的求解器会在它们共同的误解上取得一致，于是奖励证明的恰恰是你检测不到的那些错误。

留出事实重构。把一个事实从输入中移除，要求回复把它补上，并按精确比对来打分。这把一项不可验证的摘要任务变成了一项精确的任务，而且与其余几种不同，它检验的是接地而不是形式。有三种代理失效：被遮掉的事实往往可以从上下文中恢复，于是这项任务测的是推断而不是接地；被记住的事实无需阅读原文就能恢复，这就要求做一次污染检查；以及那些容易被机械遮蔽的事实，系统性地正是那些容易表述的事实。

以执行为基础的代理指标。要求回复包含一件能运行的产物——一条查询、一个脚本、一份针对模拟器的计划——并奖励其执行结果。文本转 SQL 是典范，而且恰好位于 表 15.1中部分可校验的那一段：执行证明的是这条查询能跑通并返回一个行集，而对问题是否被理解什么也证明不了。2026 年的推广形式是任务变换，即把一项开放式任务改写成一种自带验证的形式，并在改写后的任务上训练 (RLSVR, 2026)。其代理失效在整个家族中是一致的，值得点明一次：那件产物变成了目标，于是策略优化的是能执行的东西而不是被要求的东西，而两者之间的差距按构造在奖励中就是不可见的。

常见陷阱: 把代理指标当作目标来报告

这里的每一种构造都产出一个数字，它对某个比目标更窄的东西完全正确，这就使它异常容易被当作目标来报告。一个往返分数不是摘要质量，执行准确率不是语义正确性，求解器之间的一致也不是真相。应尽的义务是：在一个带标签的样本上公布该代理指标与真实目标之间实测的一致程度，并在训练结束时而不是只

> 在开始时重做这项测量：策略此后一直在优化这个代理指标，所以在初始策略上测出的那个相关性，恰恰是你可以有把握地认为已经不再成立的那个量。

15.8 截至 2026 年年中已确立的内容

这一领域大约只有十八个月的历史，所以证据层级在这里比在第 III 部分的任何其他地方都更要紧。

已复现。 在开放式对齐上，评分细则与检查清单奖励优于标量的学出来的奖励，也优于 Likert 量表的评判模型奖励；多个独立团队用不同方法与不同基准报告了这一点（Gunjal et al., 2025; Viswanathan et al., 2025; Huang et al., 2025d），它与(15.3)的机制一致，而且它与评测一侧的那项发现相吻合，即学出来的奖励恰恰在有程序可用之处最弱 (Malik et al., 2025)。同样已复现的还有：混用不同种类的逐条目评分器优于单一同质的学出来的标量 (Viswanathan et al., 2025)；静态评分细则会在一次长运行中退化，这是整个 2026 年共演化文献的共同前提；以及针对任何学出来的奖励的过度优化，在 KL 上都是非单调的，而代理指标却单调上升 (Gao et al., 2023a)。

单篇论文。 节 15.2.1中引用的每一种共演化方案都只有一篇中等规模的论文，没有一种被独立复现过，而理论——一个没有保证不动点的博弈中的交替最佳响应——也不给乐观提供任何理由（节 9.5）。评分细则引导的自蒸馏处于同样的位置，并且还多一层担忧：它把奖励藏进了生成器里面。词元级的评分细则信用分配，以及面向研究智能体的评分细则监督，都很有前景但未经他处检验 (Zhang et al., 2026a; Zhu et al., 2026c)。

仍有争议。 置信度与自我确定性奖励是否在第14章的锐化之外还做了别的什么，这仍有争议，且有一篇已发表的反面意见 (Anonymous, 2025) 以及一个严重的模型族混杂因素 (Shao et al., 2025)。评判模型集成究竟是买来了稳健性、还是只是把一个共有偏差平均了一下，这也仍有争议，其中Eisenstein et al. (2024) 是悲观读法的参考文献。以及所报告的那些开放式增益是否真如其表面所示，这一点需要把话说明白。

一个应当假定其存在、直到被排除的混杂因素。 开放式基准上所报告的增益，经常与评判模型对被训练模型文风的偏好相混杂。这个机制并不隐晦：对着一个评判模型做 RL 会把策略推向那个评判模型的偏好，而评测通常又是由同一个评判模型、或同一模型

族的评判模型、或一个其训练数据由同一模型族生成的评判模型来跑的 (Panickssery et al., 2024; Li et al., 2025a)。长度是记录得最充分的一条通道，也是唯一有标准修正手段的一条 (Dubois et al., 2024)；排版格式、语域与含糊其辞的行为方式相同，却没有公认的修正手段。一个既未做长度控制、又非跨模型族、也没有在某个切片上附带人工或程序化核查的胜率，在被证明并非如此之前，都应当被读作对文风迁移的一次测量。这一点对本章的文献与对其他任何人的文献同样适用。

开放。 在训练规模上验证长文本的事实性：这里没有任何东西能为「这两千字中的每一条论断都为真」提供一个可靠的奖励，而逐条论断的分解会撞上检索成本，也会撞上「大多数论断既非明确为真也非明确为假」这一事实。品味，即两件同样称职的工作中哪一件更值得做。以及涵盖上述两者的那种情形：任何有能力的人之间会有分歧的地方，那里的问题不是缺一个评分器，而是目标本身是一个在若干正当目标上的分布，而 Bradley–Terry 式的聚合会把它返回成一个平淡的中心（定理 5.2）。评分细则把那种分歧逐条判据地显式化了，这是可读性上的进展，而不是分歧解决上的进展。

15.9 为一项新任务构建一个可辩护的奖励

下面给出的是一个判定过程而不是一份配方，因为它的头两步经常就把它终止掉了。

工程实践: 构建一个你能辩护的奖励，2026 年年中版

1. **说出这个奖励所服务的那个决定**，并追问在一次训练运行的时间尺度上是否存在一个真值的供给方。如果不存在，而且没有可接受的代理指标，那就停下：这是表 15.1的最后一段，而诚实的交付物是一套评测，而不是一个训练信号。
2. **枚举可校验的投影。**对你在意的每一项性质，都问一问三十行 Python 能否判定它：格式、长度、必需的章节、内部日期一致性、可解析的标识符、要求使用的语言、模式符合性、可执行性。由(15.1)算出 φ 并记录下来。如果它大于零，你就有了一道闸门，而节 15.6说要把它当作闸门来用。
3. **在动用评判之前先试试半可验证构造**（节 15.7）。一次往返、一项逆向任务、一个留出事实，或者以执行为基础的接地，往往比任何数量的评判模型工程更能抬高 φ，而每一次替换都附带一份你必须写下来的失效清单。

4. **为残差写一份评分细则：**5–15 条判据，每一条都是关于回复的可观察性质而不是关于作者意图的性质，权重要显式写出，硬闸门只留给真正的约束。在训练之前先做验证：让一个人逐条标记每条判据是否可由回复判定、是否正确、是否不冗余；常规这么做的团队会丢掉四分之一到三分之一的自动生成判据。
5. **挑选那些失效方式不同的评分器：**有程序的地方用程序，没有的地方至少用两个来自不同模型族的评判模型，并配上算法 15.1的弃权规则。这是唯一能触及(15.3)中 ρ 与 b 的干预。
6. **在信任评判模型之前先测量它：**相对几百条人工标注的 Cohen's κ、交换后的不一致率、奖励对长度的斜率，以及剥格式探测。然后，如果奖励中有任何部分是自生成的，再测 Δ_{gv}（定理 15.4）。
7. **按「先把闸再评分」装配**（算法 15.2），配上逐来源的归一化，并在每一个基于模型的分量上都给一个非零的 β。即便任务并不需要，也在混合中保留一个可验证的切片作为哨兵：如果它的通过率下降而基于模型的奖励上升，那么这次运行就是在用能力换奖励。
8. **留出一个评判模型和一个提示集，让它们不参与任何环节**，并在每个检查点阅读奖励最高的十条采样轨迹，同时把逐判据的分数摆在眼前。给它安排二十分钟。这能在任何聚合指标发生变化之前好几天就发现作弊，而它正是团队会跳过的那一步。
9. **预先计划重新设定**。把评分细则的复审安排在固定的 KL 间隔上，而不是固定的步数上，因为过期是策略移动量的函数。把复审自动化就意味着要去跑节 15.2.1中那个两人博弈，所以请为循环装上监测。

本章小结

- 可验证性属于一项任务连同它的工具链，而且是程度问题：(15.1)中的可辩护份额 φ 是奖励取值范围中由一个失效模式可被你枚举的程序所产出的那一份。提高它是工程，不是挑领域。
- 部分可校验的那一段装着有价值的工作，并具有一种特征性的动力学：可靠分量很早就饱和，此后整个梯度都来自你无法辩护的那部分。精确指令遵循——在那里

学出来的奖励相对 25% 的基线只得 37–66%，而一个程序是精确的 (Malik et al., 2025)——度量了这要付出多少代价。

- 对 K 条判据取平均，把噪声乘以 $\rho + (1-\rho)/K$，而共有偏差 b 丝毫不动。削减的大部分在五条判据之内就到手了，而作用于 ρ 与 b 的唯一杠杆是评分器的多样性——一个程序，或者一个不同的模型族——而不是一份更长的清单。
- 一份评分细则把利用单条判据的收益界定在 $w_k / \sum_j w_j$；那个界就是这项技术的全部防线，而当一种行为一次性满足许多条判据时它就失效了。请记录逐判据的分数相关矩阵：它估计 ρ，并告诉你你的评分细则是不是一个披着清单外衣的标量奖励。
- 一个足以用于评测的评判模型可能不足以用作奖励，而这一差距是结构性的，不是准确率高低的问题（定理 15.3）：评测看到的是在一个固定分布上的平均误差，而 δ 奈特的 KL 预算就会暴露参考概率不低于 $e^{-\delta}$ 的每一个盲区，其严重程度只被实测准确率通过一个 $1/p$ 的因子所约束。
- 自我奖励只有在减去最好的常数判决之后生成–验证差距(15.4) 仍为正时才携带信息，而且即便如此也还要以策略自身的置信度为条件。模型恰恰在它自信地错了的地方无法看出自己错了，所以一处真正的不对称要求验证者去查阅生成时没有查阅的东西：执行、一个独立训练的校验器、更多算力，或者被扣下的信息。
- 乘法把闸优于加法混合。加法混合没有可用的工作区间，因为任何大到有影响的权重都可能颠倒排序，而且它会按评判分数给未通过的采样轨迹排序——这教会策略在无法答对时就去谄媚。把闸把那个集合压平，并缩小评判模型盲区可被抵达的区域。
- 半可验证构造——逆向任务、往返、独立求解器一致、留出事实、执行接地——通常比评判模型工程更能抬高 φ。每一种都替换进一个代理指标，其失效清单必须在训练之前写好，而其与目标的一致程度必须在训练之后重新测量。
- 2026 年的共演化文献是奖励模型维护被重新发现了一遍：一份评分细则就是一个学出来的奖励，其效度随着相对它所针对的那个策略的 KL 而衰减。把维护变成一个两人博弈消除了过期问题，却引入了循环，既无收敛保证，也无独立复现。
- 开放式基准上所报告的增益，经常与评判模型对被训练模型文风的偏好相混杂。在一项结果做到长度受控、跨模型族，并在某个切片上由人工或程序核查过之前，都应假定它是文风迁移。

延伸阅读

先读确立了评分细则奖励的那三个 2025 年系统，按这个顺序，理由如下。Gunjal et al. (2025) 是这一想法最清晰的表述，而它关于「评分细则结构对小评判模型帮助最大」的发现，是该领域唯一一项区分机制、而不只是演示一次增益的结果。然后是 Viswanathan et al. (2025)，其标题主张——检查清单胜过奖励模型——是文献中最强的主张，而它真正的贡献更微妙也更有用：逐条目的评分器取自评判模型与专用程序的混合，这正是一个非可验证领域获得一个部分可靠的奖励的具体机制。再读 Huang et al. (2025d)，看当评分细则库变大、领域变得真正开放时会发生什么。读这三篇时请对照节 5.6，它提供了它们所预设的奖励建模背景。

对于 2026 年的转向，Ding et al. (2026) 是对抗式框架最干净的表述，而 Yang et al. (2026a) 在技术上最有意思，因为制造出检验一条判据所需的成对证据，是一个比编辑一份判据清单更难也更一般的问题；Wang et al. (2026b) 值得为其测试时变体一读，在那里完全没有训练集，评分细则是在你手头真正有的那些提示上共同适配的。请把它们放在节 9.5旁边读，并注意这批文献迄今为止一直不愿引入偏好学习为同一个不存在性问题所发展出来的那套均衡机制。

关于评判模型，必读的不是某篇方法论文，而是 2026 年的那几项可靠性审计。Anonymous (2026b) 排在最前，因为经过机遇校正的一致性改变了文献中其他每一个评判模型数字该被如何解读；而Anonymous (2026c) 是它天然的同伴，其发现是没有哪种评判模型配置能在各种探测下都稳健；Li et al. (2025a) 补全了这幅图景，说明为什么对着一个相关的评判模型算出的胜率难以解读。关于评判之外的替代方案，请把Zhao et al. (2025d) 与Prabhudesai et al. (2025) 连同 Anonymous (2025) 与Shao et al. (2025) 按这个顺序一起读，作为「如何评估一项内在奖励主张」的案例研究；Zuo et al. (2025) 是「伪标签可以替代标签」这一点最干净的演示，而Shafayat et al. (2025) 是关于「这条路在哪里走不动」的自然后续。最后，Zhou et al. (2023b) 已有三年之久、篇幅很短，但对本章而言比多数 2026 年的工作都更有分量，因为它是那个存在性证明：一个非可验证的领域里含着一个可验证的投影，只等有人把它写下来。

习题

15.1（方差削减在哪里停下。）利用(15.3)：（a）对 $\rho \in \{0, 0.1, 0.3, 0.6\}$ 画出 $\mathbb{V}\mathrm{ar}[r_{\mathcal{R}}]/\sigma^2$ 关于 K 的曲线，并对每个 ρ 找出达到可获得削减量之 90% 的最小 K。（b）某团队报告说把评分细则从 8 条扩到 40 条判据什么也没改变。请给出与之相容的 ρ 的范

围，并说明什么测量能确认它。(c) 解释为什么把三条由评判模型判定的判据换成三条程序化的判据，可能比再加二十条评判判据更有帮助，并在一个「程序化判据具有 $\rho = 0$ 与 $b = 0$」的模型中把这一主张量化。

15.2 **（单条判据被利用的收益上界。）**取(15.2)，设有 K 条等权重的软判据且没有闸门。(a) 证明虚假地满足一条判据会获得奖励取值范围的 $1/K$。(b) 假设某一种行为在比例为 f 的判据上把 s_k 提高了 δ；请计算其收益，并给出使这份评分细则不比一个标量奖励更稳健的关于 f 的条件。(c) 提出一个仅由采样轨迹数据就能算出的 f 的估计量，并说明它被什么所混杂。

15.3 **（解读那条不对称性命题。）**(a) 利用定理 15.3，计算抵达一个参考概率为 10^{-3} 的盲区所需的 KL 预算，并把它与一次典型 RLVR 运行所花掉的 KL 相比较。(b) 某评判模型实测 $\eta = 0.02$；在一个概率为 10^{-3} 的集合上，第 (1) 部分允许 Δ 取到多大？(c) 解释为什么对着同一个评判模型做 best-of-n 选择更安全，并给出其代价随 n 增长的速率。(d) 说明本章中哪些技术作用于 p、哪些作用于 Δ、哪些作用于 δ。

15.4 **（把闸对混合，逐步算一遍。）**对(15.5)，设 $r_{\text{ver}} \in \{0, 1\}$ 且 $q \in [0, 1]$：(a) 推导出排序保证对 λ 的精确条件，并证明它迫使模型项只能充当打破平局的角色。(b) 对一组 $G = 8$ 条采样轨迹、其中两条通过，计算两种形式下的组相对优势，并说明加法形式教给了那六条未通过的采样轨迹什么。(c) 证明当没有任何采样轨迹通过时把闸会给出恒等于零的优势，并给出两种缓解手段及各自的代价。(d) 把 α 解释为一个权重，并给出使得在通过的回复之间正确性与评分细则质量贡献相等的那个取值。

15.5 **（诊断作弊。）**对下列每一项观察，请指出节 15.5 中最可能的机制，以及能确认它的那一项最便宜的测量。(a) 奖励稳步上升，留出集上的胜率持平，长度未变。(b)「考虑了其他解释」这条判据的满足度从 0.3 升到 0.95，而一个留出评判模型的分数在下降。(c) 奖励在某一步上跨所有提示、所有领域不连续地跳升。(d) 回复变长了 40%，而不同事实断言的数量没有变化。(e) 含有引用的回复比例达到 1.0，而引用能被解析成功的比例在下降。

15.6 ⋆ **（这个共演化循环有不动点吗?）**把 节 15.2.1的评分细则／策略循环形式化为一个两人博弈：评分细则玩家从一个有限的判据池中选出 K 条判据，策略玩家在有限多种行为上选择一个分布，每种行为满足一个已知的判据子集并带有一个已知的真实质量。(a) 给出交替最佳响应能够收敛的条件，并构造一个以周期二循环的

最小实例。(b) 证明允许评分细则玩家在判据子集上做随机化——即节 9.5会开出的那种混合策略，而 Constitutional AI 的逐比较原则抽样 (Bai et al., 2022b) 顺带实现了它——能恢复均衡的存在性。(c) 量化这种混合策略的代价：奖励在各步之间变成随机的，请把新增的优势方差估计为随机化熵的函数，并把它与(15.3)中被消掉的那部分方差分解相比较。

15.7 ⋆ **（置信度奖励能否与锐化区分开?）**设计一个能定夺 节 15.4那条有争议主张的实验。它必须(i)把 pass@1 上的增益与大 k 处 pass@k 曲线的变化区分开(节 14.2)；(ii)鉴于Shao et al. (2025)，对模型族做控制；并且 (iii) 包含一个把熵降低本身隔离出来的条件——降低温度，或者一个完全不带奖励的熵惩罚——从而把置信度奖励与产生同等熵变化的最便宜干预相比较。请事先说明哪一种结果会证伪「自我确定性奖励教会了任何东西」这一主张，并解释为什么通常那样做的实验做不到这一点。

15.8 † **（把评分细则奖励搭起来，再把它打破。）**取一个没有验证器的开放式提示集与一个 7–8B 的策略。实现代码 15.1与 算法 15.2，至少包含四条程序化判据与六条评判判据，然后在 $\beta = 0$ 下对着静态评分细则训练，直到出问题为止。交付：(i) 逐判据的满足度曲线，以及逐判据分数相关矩阵随训练步的变化，并标出饱和；(ii) 节 15.5 中每一种机制首次出现的步数，且要通过阅读前十分位的采样轨迹来确定，而不是靠某个指标；(iii) 作为判据回声检测器的、评分细则与回复之间的 n-gram 重合度；(iv) 一个来自不同模型族的留出评判模型，按同样的时间表运行。然后按 节 15.2.1的方式手工编辑一次判据、恢复训练，并报告失去的留出分数回来了多少。

15.9 † **（那个可验证切片值多少?）**跑一遍本章说还缺的那项消融。在一个混合提示集上训练三次其余设置完全相同的运行：只用评分细则、只用可验证奖励，以及按 1:1 的提示比例混合并配上节 15.6的逐来源归一化与逐领域 β。按一个固定的时间表，测量在一个从未进入训练的评判模型上的留出评分细则领域胜率、可验证领域的通过率、长度与熵。报告 (i) 混合运行的评分细则领域是否退化得更慢，这正是实践者的主张；(ii) 可验证领域是否为此付出了代价；以及 (iii) 在两个方向上改成 1:4 时答案会如何变化。请用节 12.12的功效论证，诚实地说明你的运行规模能否把这一效应与噪声区分开。

表 15.2: 作为优化靶子的评判模型偏差。中间一列是每项偏差一旦成为训练信号而不再只是测量误差之后所选择出来的行为；右边一列是便宜到每个检查点都能跑一遍的诊断。这里刻意不给量值：这些偏差在各实验室之间是定性可复现的，而唯一值得据以行动的数字，是你在自己的评判模型、模板与提示分布上测出来的那一个。

偏差	它在 RL 下选择出什么	诊断
位置 (Wang et al., 2024e)	不直接选择出什么，但它把奖励的一部分随机化，表现为拖慢学习的、不可约的优势噪声	交换后的不一致率：对两种排序都打分并统计判决翻转次数；这同时也是该评判模型的有效噪声地板
冗长 (Singhal et al., 2024)	灌水、复述、罗列各种限定说明，因为长度很便宜，而且在评判模型的训练数据里与质量相关	在留出的采样轨迹上，奖励对词元数的斜率，加上在保持内容不变的扩写下奖励的变化；请报告长度受控的胜率 (Dubois et al., 2024)
自我偏好 (Panickssery et al., 2024)	策略自身的文风惯用法，这正是使训练过的模型的胜率难以解读的那个混杂因素（节 15.8）	在完全相同的样本对上做跨模型族评判；在一个留出的校准集上做误差偏置探测 (Li et al., 2025a)
排版格式	用标题、要点、加粗与章节脚手架来代替内容	给剥掉格式后的改写打分：奖励不变说明该判据关乎内容，奖励崩塌说明它关乎标记
谄媚 (Sharma et al., 2024)	认同用户所陈述的前提，包括错误的前提	一个固定的探测集，包含用户断言的错误前提与事后的反驳追问，每个检查点都打一次分，并且永不重新生成
笃定断言	不加限定的论断，因为在多数评判模型看来限定说明读起来像是软弱，尽管在不确定下它才是正确的	论断密度（每百词中可验证的断言数）对照这些断言上的准确率
经过机遇校正的一致性	这不是一项偏差，而是一个报告上的错误，它抬高了人们对上述所有各项的信心	报告相对人类标注的 Cohen's κ，而不是原始一致率；在多类标签均衡的基准上，两者相差可达数十个百分点（Anonymous, 2026b）

算法 15.1 带弃权的评判模型集成奖励。模型族的多样性攻击的是(15.3) 中的 ρ 与 b；弃权规则则直接攻击定理 15.3，办法是在集成的盲区最可能被暴露之处拒绝提供梯度。一个被弃权的样本获得组均值，因此它贡献的是零优势，而不是一次猜测。

输入： 采样轨迹 $\{y^{(i)}\}_{i=1}^{G}$、来自不同模型族的评判模型 $\pi_{\mathrm{J}}^{(1)}, \dots, \pi_{\mathrm{J}}^{(J)}$、分歧容忍度 τ、弃权上限 $\alpha_{\max}$

for $i = 1$ **to** G **do**
  for $j = 1$ **to** J **do**
    $q_{i,j} \leftarrow \frac{1}{2}[\pi_{\mathrm{J}}^{(j)}(x, y^{(i)}; \mathrm{AB}) + \pi_{\mathrm{J}}^{(j)}(x, y^{(i)}; \mathrm{BA})]$ ▷ 抵消位置偏差
  end for
  $\mu_i \leftarrow \mathrm{median}_j\, q_{i,j}$; $d_i \leftarrow \max_j q_{i,j} - \min_j q_{i,j}$ ▷ 中位数能顶住一个离群的评判模型
  if $d_i > \tau$ **then**
    $a_i \leftarrow 1$ ▷ 弃权：集成内部意见不一致
  else
    $a_i \leftarrow 0$; $r_i \leftarrow \mu_i$
  end if
end for
if $\frac{1}{G}\sum_i a_i > \alpha_{\max}$ **then**
  return SKIP PROMPT ▷ 超出评判模型的胜任范围；记录下来
end if
$\bar{r} \leftarrow \mathrm{mean}\,\{r_i : a_i = 0\}$
for i **with** $a_i = 1$ **do**
  $r_i \leftarrow \bar{r}$ ▷ 零优势，而不是一个猜出来的奖励
end for
return $\{r_i\}$、弃权率、平均分歧度

算法 15.2 针对一个采样组的、以评分细则把闸的混合奖励，它把 代码 15.1、算法 15.1与节 15.6装配在一起。操作的次序就是其内容：可靠分量在先并作为闸门，模型分量在后并且只在幸存者中间进行，归一化按来源做，而且每一个中间量都要记录，因为它们事后没有一个是可以复原的。

输入： 领域为 d 的提示 x、采样轨迹 $\{y^{(i)}\}_{i=1}^{G}$、可靠评分器 $\{g_k\}$、评分细则 $\mathcal{R}(x)$、评判模型集成、底值 α、滑动统计量 $(\mu_{\mathrm{d}}, \sigma_{\mathrm{d}})$

阶段 1：可辩护的分量（便宜、可靠、没什么可作弊的）
for $i = 1$ **to** G **do**
  $\mathrm{pass}_i \leftarrow \bigwedge_k [g_k(x, y^{(i)}) = 1]$ ▷ 格式、约束、可解析的标识符
end for
if $\sum_i \mathrm{pass}_i = 0$ **then**
  return 零优势；把 ALL-GATED-OUT 计数器加一 ▷ 该计数器上升意味着闸门太严
end if
阶段 2：模型分量，只在闸门打开之处求值
for i **with** $\mathrm{pass}_i = 1$ **do**
  $(q_i, s_{i,\cdot}) \leftarrow$ 用算法 15.1的评判模型集成执行代码 15.1
end for
对所有 i，$r_i \leftarrow \mathrm{pass}_i \cdot (\alpha + (1-\alpha) q_i)$ ▷ 乘法闸门，(15.5)
阶段 3：逐来源归一化与优势
更新 $(\mu_{\mathrm{d}}, \sigma_{\mathrm{d}})$；$\tilde{r}_i \leftarrow (r_i - \mu_{\mathrm{d}}) / \max(\sigma_{\mathrm{d}}, \sigma_{\min})$
$A_i \leftarrow \tilde{r}_i - \mathrm{mean}(\tilde{r}_{1:G})$ ▷ 在组内中心化；不要除以组标准差
阶段 4：不可省略的诊断
记录闸门通过率、逐判据满足度 $\bar{s}_k$、弃权率、评分细则与回复之间的 n-gram 重合度，以及每条判据的 $\mathbb{V}\mathrm{ar}_i[s_{i,k}]$
标记出任何满足 $\bar{s}_k \to 1$ 或 $\mathbb{V}\mathrm{ar}_i[s_{i,k}] = 0$ 的判据 ▷ 已饱和或无区分力
return $\{A_i\}$ 以及这些诊断

第 IV 部分

智能体

第 16 章

从推理到智能体

核心思想: 智能体化改变的是被优化的对象，而不是优化它的难度

第三部分中的每一个算法都假定 $y \sim \pi_\theta(\cdot|x)$：轨迹完全由策略生成，因此其对数似然端到端可微，其词元也全都是动作。而在智能体式的设定中，环境会在策略的两次轮次之间把词元写进上下文。此时轨迹是两个分布的联合乘积，而我们只控制其中之一，也只能对其中之一求导。本章推导这件事破坏了什么——损失掩码、重要性比值、折扣因子的含义、时域，以及组相对基线的解释——以及在Chapters 6, 12 and 13 的全套机器能够被重新使用之前，必须修补些什么。

第四部分讲的是模型开始行动之后会发生什么。随后的四章分别处理工具（第17章）、长时域（第18章）、环境（第19章）与多个智能体（第20章）。本章搭建这四章共同需要的机器，并且刻意保持保守：它不引入任何算法。它的任务是精确地界定，第2章中定义的那些对象里，哪些在转向智能体之后依然成立，哪些以修改后的形式成立，哪些则不再有良好定义。2025–2026 年文献中大量的混乱，都可以归因于那些不加追问就复用单轮估计量的论文。

16.1 究竟改变了什么

节 2.3列举了语言模型 MDP 的五处退化性质，并论证它们是有用的，因为它们允许一些在一般 MDP 中不可能存在的估计量。逐条复核这份清单，就是本章的一个缩影。

1. **动力学是确定且已知的。**不再成立。一次工具调用把控制权交给了一个我们既观测不到其状态、也预测不了其输出的进程。从某个前缀重新采样不再能复现原轨迹，这使得节 13.4中那种廉价的蒙特卡洛值估计失效——除非环境可以快照并恢复。

2. **动作分布是可精确计算的。**依然成立，但仅限于策略自己发出的词元。模型同样会给环境写入的词元赋概率，而那些概率并不是动作概率。任何定义为在「所有词元」上求和的量——对数似然、逐词元 KL、熵、重要性比值——现在都需要一个限定词，而节 16.4会给出它。
3. **奖励是终局的、稀疏的，且往往是二元的。**部分成立。它通常仍是终局的，但原则上现在可以拿到逐轮信号了——一个测试通过了、一个文件写成了、一次检索什么都没返回——至于要不要用它们，仍是悬而未决的问题。
4. **折扣因子应当取 1。**现在真正有争议，而且更糟的是，它变得含混：按词元计的 γ 与按轮次计的 γ 是两个不同的对象，效果也不同；一篇论文报告「$\gamma = 0.95$」却不说明是哪一种，等于什么都没报告。
5. **策略必须始终是一个好的语言模型。**依然成立，而且要求更强。它现在还必须始终是一个行为良好的工具调用者，而这是一条由解析器而非由品味来强制执行的格式约束，因此也是优化器可以灾难性地而非渐进地破坏掉的约束。

在这份复核之下是四项结构性变化，值得分别陈述，因为不同的下游修补针对的是不同的那一项。

（a）上下文中包含策略并未发出的词元。 一条轨迹是策略发出的片段与环境写入的片段的交错。环境片段必须被注意到——它们正是下一次决策所依赖的信息——但绝不能被训练，因为它们不是动作。这是同一个张量上的两个不同掩码，把二者混为一谈是智能体式强化学习中最常见的单一实现错误（节 16.4）。

（b）轨迹分布依赖于 $\mathcal{E}$。 一条轨迹的密度分解为策略因子与环境因子。环境因子不依赖 θ，因此对得分函数没有任何贡献——这一点我们后面会用到，它也是掩码做法的形式依据。环境因子真正贡献的，是任何策略侧的方差缩减都无法消除的方差，以及一个策略只能部分控制的状态访问分布。

（c）时域是一个策略可部分控制的随机变量。 策略在每一轮都决定是再行动一次还是直接作答。因此轮数的分布会在训练过程中漂移，而漂移的方向是结果奖励并不在意的（节 16.6）。轮数爆炸是长度爆炸在智能体设定下的对应物，二者结构相同：一个目标函数并未定价的量，在其余每一项施加的优化压力下发生漂移。

（d）奖励可能出现在任何地方。 在结尾（一个测试套件）、在每一轮（工具返回了一个错误），或者来自一个我们在上下文里永远看不到的状态（数据库中的某一行被正确写入了）。最后一种情形是要害：奖励可以是隐藏环境状态的函数，因而不是可见轨迹的函数，因此无论对词元序列做多少内省，都无法把它辨识出来。

16.1.1 一个算过账的词元预算

一旦按词元计数，这种对比就变得具体了。第11章所训练的那种单轮 RLVR 采样轨迹——一个 200 词元的提示加上一条 8,000 词元的思维链——其可训练比例是 8000/8200 = 97.6%，因为每一个生成出来的词元都是动作。现在换成一条 40 轮的工具使用轨迹：系统提示与工具模式占 400 词元；前 39 轮中每一轮发出 150 个动作词元（一小段思考加一次工具调用），并收到 600 个观测词元（一个结果集、一份文件清单、一段栈回溯）；第 40 轮是 400 词元的最终答案。策略共发出 6,250 个词元，环境写入了 23,400 个，最终上下文长度为 30,050 个词元。可训练比例是 20.8%。

这一个比值带来三项后果，它们在第四部分中反复出现。

第一，**训练时的前向与反向传播大部分被浪费了**。为了得到某个动作词元的 logit，训练器必须处理它整个因果前缀，所以代价由 30,050 词元的上下文决定，而梯度只支撑在 6,250 个位置上。这条轨迹的训练 FLOPs 里，大约五分之四完全不产生任何梯度信号；而且在不改变上下文管理策略的前提下对此无计可施——这正是第18章要花那么大篇幅讲压缩的原因。

第二，**逐词元的优势记账方式变了**。一个广播到所有动作词元上的轨迹级分数，现在被分摊到分布于 40 个决策点上的 6,250 个位置，而不是一条不间断流中的 8,000 个位置。由此得到的估计量的方差，以及(6.17)中归一化因子 $\mathcal{N}$ 的解释，两者都发生了变化（节 16.5）。

第三，**墙钟时间不再由解码主导**。这条采样轨迹需要 40 个顺序执行的解码阶段，其间被 39 次环境往返隔开。在纯粹用于说明的假设下——每秒解码 40 个词元、工具延迟两秒——解码耗时约 156 秒，环境约 78 秒；也就是说采样轨迹有三分之一的时间花在等待一个并不运行在加速器上的进程；而在组大小 $G = 8$ 时，每个提示都必须在八个彼此隔离的环境实例中进行，否则同一组内的各条采样轨迹会互相破坏对方的状态。这正是第21章中那些异步、解耦架构的由来，也是那个到 2026 年已成为本领域组织性论断的说法的由来：稀缺资源是环境，而不是数据 (Zhang et al., 2026b; Huang et al., 2025c)。

表 16.1: 在节 16.1.1的假设下，一条单轮 RLVR 采样轨迹与一条 40 轮工具使用采样轨迹的解剖。表中的词元数是由所述假设算出来的，不是实测值；要点在于其比例，而这些比例对已发表的多轮配方是有代表性的。

	单轮 RLVR	**40 轮工具使用**
提示词元	200	400（以工具模式为主）
策略发出的词元	8,000	6,250，分布在 40 个片段中
环境写入的词元	0	23,400，分布在 39 个片段中
最终上下文长度	8,200	30,050
可训练比例	97.6%	20.8%
解码阶段数	1	40，由环境串行化
随机转移	无	每条轨迹 39 次
奖励观测	1 次，终局	1 次终局，外加至多 40 次逐轮
单个片段出错	使一条轨迹变差	可能使整条轨迹作废
隔离要求	无	每条采样轨迹一个环境实例

16.2 词元级 POMDP

现在来构造这个形式化对象。这一构造是对节 2.2中词元级 MDP 的扩展而非替换，这是刻意为之：训练器计算的仍然是逐词元的对数似然，任何与这一计算失去接触的形式化都不可能被实现。

16.2.1 轮次、片段与交错轨迹

用 $k = 1, \dots, K$ 给轮次编号。在第 k 轮，策略发出一个动作片段

$$a^{(k)} = (a_1^{(k)}, \dots, a_{n_k}^{(k)}) \in \mathcal{V}^{n_k}, \tag{16.1}$$

它或者以一个工具调用定界符结束（控制权移交给环境），或者以 <eos> 结束（回合终止）。如果控制权移交给环境，环境就把一个观测片段 $o^{(k)} \in \mathcal{V}^{m_k}$ 写回上下文。上下文按下式演化

$$h^{(1)} = x, \qquad h^{(k+1)} = h^{(k)} \oplus a^{(k)} \oplus o^{(k)}, \tag{16.2}$$

其中 $\oplus$ 表示拼接，而轨迹就是那个交错序列

$$\tau = (x,\ a^{(1)},\ o^{(1)},\ a^{(2)},\ o^{(2)},\ \dots,\ a^{(K)}). \tag{16.3}$$

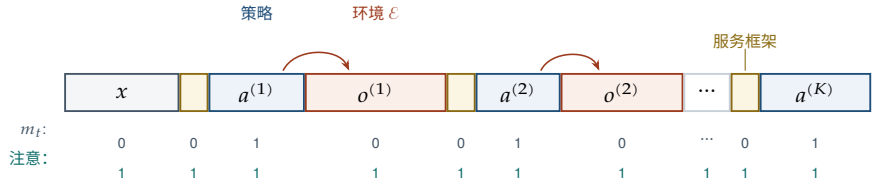


图 16.1: 交错轨迹(16.3)。蓝色片段是策略动作，锈色片段由环境写入，金色细条是服务框架插入的角色标头与工具结果定界符。损失掩码 m_t 只支撑在动作片段上；注意力掩码则覆盖全部。这是同一个张量上的两个不同掩码：环境词元只被条件依赖，从不被训练。这里锈色片段的宽度不代表任何比例，但在实践中它们占主导——在表 16.1中约占上下文的 78%。

把(16.3)摊平成一条长度为 L 的词元序列，就把位置集合 $\{1,\dots,L\}$ 划分为提示、策略所发出位置构成的动作指标集 $\mathcal{I}_a(\tau)$，以及环境所写入位置构成的观测指标集 $\mathcal{I}_o(\tau)$。还有第三类——由服务框架插入的模板词元与角色标头——将在节 16.4中讨论；就一切要紧的目的而言，它们都应归入观测一类。

16.2.2 决策过程

定义 16.1 (智能体式词元级决策过程)**.** 固定一个环境 $\mathcal{E}$，它具有状态空间 $\mathcal{S}_{\mathcal{E}}$、一个初始状态分布，以及一个转移核 $\mathcal{E}(o,e' \mid e,a)$，该核把一个状态与一个已完成的动作片段映射为一个观测片段和一个后继状态。智能体式决策过程是如下元组 $(\mathcal{S},\mathcal{U},P,r,\gamma,O)$，其中

$$\mathcal{S} = (\mathcal{X} \times \mathcal{U}^*) \times \mathcal{S}_{\mathcal{E}}, \qquad s_t = (c_t,\, e_t), \tag{16.4}$$

$$O(s_t) = c_t, \qquad \text{动作空间} = \mathcal{U}, \tag{16.5}$$

其中 c_t 是策略可见的词元上下文，e_t 是环境状态。转移核是两种机制的混合：

$$P(s_{t+1} \mid s_t, a) = \begin{cases} \delta((c_t \oplus a,\; e_t)) & a \text{ 未结束一次工具调用,} \\ \displaystyle\sum_{o} \mathcal{E}(o, e_{t+1} \mid e_t, a^{(k)})\, \delta(c_t \oplus a \oplus o) & a \text{ 结束了片段} a^{(k)}. \end{cases} \tag{16.6}$$

策略为 $\pi_\theta(a \mid O(s_t))$，i.e., 它只是可见上下文的函数。

这个定义有两处特征值得强调。

第一，转移核几乎处处是确定的，只在测度为零的一小撮位置上是随机的。在一个动作片段内部，(16.6)就是节 2.2中的那种拼接；恰恰在结束一次工具调用的那个词元

上，这个核才变成一次真正的环境转移。在表 16.1的预算里，这意味着 30,050 个位置中只有 39 次随机转移，即七百七十分之一。这一过程的绝大部分都还是我们已经理解的那个退化 MDP，而全部难点都集中在占比趋近于零的那些位置上——正因如此，写出「平均而言正确、偏偏在要紧之处出错」的代码才如此容易。

第二，按Kaelbling et al. (1998) 的意义，这个过程是真正部分可观测的。策略以 c_t 为条件，而 P 依赖于 e_t。两条可见上下文完全相同的轨迹，其环境状态可以不同——一个里面存在某个文件而另一个没有，一个里面有个数据库事务开着，一个里面速率限额已被用掉——因而转移分布不同，奖励也不同。

核心思想：你所条件依赖的状态，并不是决定动力学的那个状态

这正是智能体式设定为何是 POMDP 而非 MDP 的原因，而且它绝非技术细节。它意味着：（i）最优策略一般是整个历史的函数，而不是任何固定大小摘要的函数，因此上下文压缩不是无损的，第18章中的各种压缩方案都是有真实代价的近似；（ii）在 c_t 上拟合的值函数估计的是对隐藏状态求边缘之后的 $\mathbb{E}[G \mid c_t]$，所以它的残差方差有一个不可约的下限，再大的评论家容量也消不掉；（iii）在组相对比较中，两条采样轨迹回报不同，可能源于隐藏状态而非策略自身的选择，这就让每一个组相对优势都被环境的运气所偏置。第（iii）点是节 16.7中「环境非确定性」那一条背后的机制，也是Shang et al. (2025) 那类「答对即重采样」方案存在的理由。

16.2.3 策略梯度，以及环境去了哪里

写出轨迹密度。略去提示分布，并用 c 表示当前上下文，

$$p_\theta(\tau) = \underbrace{\prod_{k=1}^{K}\prod_{j=1}^{n_k} \pi_\theta\!\left(a_j^{(k)} \mid h^{(k)} \oplus a_{<j}^{(k)}\right)}_{\text{策略因子}} \cdot \underbrace{\prod_{k=1}^{K-1} \mathcal{E}\!\left(o^{(k)}, e^{(k+1)} \mid e^{(k)}, a^{(k)}\right)}_{\text{环境因子}}. \tag{16.7}$$

命题 16.2 (环境因子不含得分项)**.** 设 $\mathcal{J}(\theta) = \mathbb{E}_{\tau\sim p_\theta}[r(\tau)]$，其中 r 可以依赖隐藏的环境状态。则

$$\nabla_\theta \mathcal{J}(\theta) = \mathop{\mathbb{E}}_{\tau\sim p_\theta}\Big[r(\tau) \sum_{t\in \mathcal{I}_a(\tau)} \nabla_\theta \log \pi_\theta(y_t \mid c_{<t})\Big], \tag{16.8}$$

即只在策略发出的位置上求和。梯度中没有任何一项支撑在 $\mathcal{I}_o(\tau)$ 上。

证明. 对(16.7)取对数并求导。环境因子中不含 θ,故 $\nabla_\theta \log \mathcal{E}(\cdot) = 0$,于是 $\nabla_\theta \log p_\theta(\tau) = \sum_{t \in \mathcal{I}_a} \nabla_\theta \log \pi_\theta$。再用似然比恒等式 $\nabla_\theta \mathbb{E}_{p_\theta}[f] = \mathbb{E}_{p_\theta}[f \nabla_\theta \log p_\theta]$,它只要求 p_θ 的支撑集不依赖于 θ——这里确实如此,因为环境核是固定的。 □

命题 16.2就是观测掩码的全部形式化内容。但要弄清它没有说什么。它并没有说,加在观测词元上的损失项在期望意义下为零;那一项是另一个目标函数的梯度,节 16.4会指出那是哪一个。它也没有说环境与估计量无关。有三样东西确实变了。

推导: 环境让估计量付出了什么

方差。以环境的实现为条件,把回报的方差分解: $\mathbb{V}\mathrm{ar}[G] = \mathbb{E}[\mathbb{V}\mathrm{ar}[G \mid \text{环境抽样}]] + \mathbb{V}\mathrm{ar}[\mathbb{E}[G \mid \text{环境抽样}]]$。第二项是任何策略侧手段——更大的组、更好的基线、作用在 $\nabla \log \pi_\theta$ 上的控制变量——都无法削减的。要削减它只能靠控制环境(固定随机种子、快照—恢复、确定性的工具桩),这正是第19章把确定性当作一项工程硬要求而非锦上添花的原因。

重要性比值。对一条已存储的轨迹,其比值为

$$\frac{p_\theta(\tau)}{p_{\theta_{\text{old}}}(\tau)} = \prod_{t \in \mathcal{I}_a(\tau)} \frac{\pi_\theta(y_t \mid c_{<t})}{\pi_{\theta_{\text{old}}}(y_t \mid c_{<t})}, \tag{16.9}$$

因为环境因子恰好相消——但这只是因为我们以已实现的观测为条件,而没有重新执行环境。如果训练器从存储的聊天消息重新渲染上下文并再次调用工具,或者某个时间戳、某个非确定的排序发生了变化,那么分子与分母就成了在不同上下文上的比值,(16.9) 也就干脆是错的。Cheng et al. (2025) 给这一失效的分词器版本命了名——把消息通过聊天模板重新渲染,未必能复现采样时的词元 id,因为分词并不可逆——并通过存储保留了精确采样动作词元的步级记录来修复它。

蒙特卡洛值估计。在节 2.3中,从一个前缀重采样一次就能以零基础设施成本给出无偏的值估计,因为前缀决定了状态。而这里,从 c_t 重采样是从一个未知的环境状态开始的,因此只有当环境是从给定 c_t 的后验中重新抽样时,该估计对 $\mathbb{E}[G \mid c_t]$ 才无偏,而这需要一个可重置的环境。这是无评论家的蒙特卡洛信用分配(第13章)迁移到智能体设定时表现不佳的最大原因。

16.3 轮次级 MDP

还有第二种更粗粒度的表述,而在两者之间的取舍是本章中后果最重的建模决策。大多数论文都在悄无声息地做出这个决定。

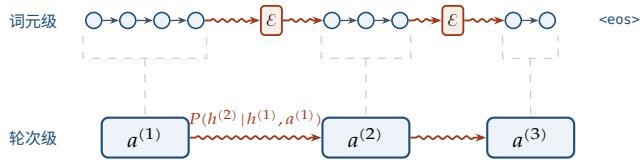


图 16.2: 同一条轨迹的两种粒度。在词元级（上），除了结束一次工具调用的那个词元处环境会介入（波浪箭头）之外，转移都是确定性的拼接。在轮次级（下），每个动作片段都坍缩为单个动作，而每一次转移都是一次真正的环境转移。轮次级过程是词元级过程的半马尔可夫抽象；命题 16.3给出了两者的策略梯度何时恰好一致。

16.3.1 构造

把一整轮当作一个动作。定义轮次级过程如下

$$\text{状态} = h^{(k)} \quad (\text{或联合的}(h^{(k)}, e^{(k)})), \qquad \text{动作} = a^{(k)} \in \mathcal{U}^*, \tag{16.10}$$

$$\pi_\theta(a^{(k)} \mid h^{(k)}) = \prod_{j=1}^{n_k} \pi_\theta(a_j^{(k)} \mid h^{(k)} \oplus a_{<j}^{(k)}), \qquad h^{(k+1)} \sim P(\cdot \mid h^{(k)}, a^{(k)}), \tag{16.11}$$

配以逐轮奖励 $r^{(k)}$ 与折扣 γ_{turn}。所诱导的转移核 P 就是上面词元级定义中的环境核对隐藏状态求边缘的结果。每当一篇论文说「我们把每次工具交互视为一步」，它隐式使用的就是这一表述。

按选项框架 (Sutton et al., 1999) 的确切意义，轮次级过程是词元级过程的一个 半马尔可夫抽象：每一轮都是一个选项，其内部策略是自回归解码器，其终止条件是「发出一个工具调用定界符或 <eos>」。这个框架之所以有用，是因为它立刻告诉我们该检查什么——选项级的策略梯度只有在优势与折扣如何在选项内部分配这一点满足某些条件时，才等于原语级的策略梯度。

16.3.2 两个策略梯度

轮次级策略梯度就是折扣 MDP 的标准形式：

$$\nabla_\theta \mathcal{J} = \mathbb{E}\Big[\sum_{k=1}^{K} \gamma_{\text{turn}}^{k-1} A^{\pi_\theta}(h^{(k)}, a^{(k)}) \nabla_\theta \log \pi_\theta(a^{(k)} \mid h^{(k)})\Big], \tag{16.12}$$

又因为 $\nabla_\theta \log \pi_\theta(a^{(k)} \mid h^{(k)}) = \sum_{j=1}^{n_k} \nabla_\theta \log \pi_\theta(a_j^{(k)} \mid \cdot)$，每个轮次级项都展开成该轮各词元上的求和，且所有词元携带同一个系数。相比之下，词元级梯度是

$$\nabla_\theta \mathcal{J} = \mathbb{E}\Big[\sum_{t \in \mathcal{I}_a} \gamma_{\text{tok}}^{t-1} A_t \nabla_\theta \log \pi_\theta(y_t \mid c_{<t})\Big]. \tag{16.13}$$

命题 16.3 (轮次级梯度是一个受约束的词元级梯度)**.** (16.12)与(16.13)是同一个估计量，当且仅当 $\gamma_{\text{tok}} = 1$ 且逐词元优势在各轮之间分段常值，且满足

$$A_t = \gamma_{\text{turn}}^{k(t)-1} A^{\pi_\theta}(h^{(k(t))}, a^{(k(t))}) \qquad \text{对} t \in \mathcal{I}_a, \tag{16.14}$$

其中 $k(t)$ 是包含位置 t 的那一轮。

证明. 在 $\gamma_{\text{tok}} = 1$ 下把(16.14)代入(16.13)，并把关于 $t \in \mathcal{I}_a$ 的求和按轮次分组；某一轮内各位置的内层求和即为 $\nabla_\theta \log \pi_\theta(a^{(k)} \mid h^{(k)})$，于是得到(16.12)。反过来，(16.12)对一轮之内所有的 $\nabla_\theta \log \pi_\theta(a_j^{(k)} \mid \cdot)$ 强加了一个共同系数，而 $\gamma_{\text{tok}}^{t-1}$ 只有在 $\gamma_{\text{tok}} = 1$ 时才在一轮之内保持常值；又因为对一般的策略而言，不同位置的得分向量线性无关，故两个估计量相等就迫使(16.14)成立。 □

命题 16.3正是文献中主流设计之所以是Zhang et al. (2026b) 实际上所称的轮次结构化优势配词元级损失的原因：损失保持自回归训练器本来就在计算的那个形式，而把全部轮次结构都经由优势向量注入。代码 16.1用六行给出了这一构造。

代码 16.1: 轮次结构化优势、词元级损失：(16.14)的广播。`turn_spans` 是动作位置上的左闭右开区间，因此观测位置与模板位置永远不会被赋予优势，并且还会被掩码再次置零。对齐方式遵循代码 16.2：返回向量的第 j 项为位置 $j+1$ 处的词元打分。

```
def broadcast_turn_advantages(turn_adv, turn_spans, seq_len, target_mask,
                              gamma_turn=1.0):
    """turn_adv: [K] float; turn_spans: [(start, end)] over action positions."""
    adv = torch.zeros(seq_len)
    for k, (a, (s, e)) in enumerate(zip(turn_adv, turn_spans)):
        adv[s:e] = (gamma_turn ** k) * a
    return adv[1:] * target_mask          # 移位到目标对齐，然后加掩码
```

16.3.3 两种表述真正分歧之处

有意思的内容在那个「仅当」。实践中有五处分歧要紧，每一处都对应表 12.5中一个在智能体式设定下表现不同的格子。

1. 轮内分辨率。 轮次级表述无法表达一轮开头的思考与这一轮末尾的工具调用之间的信用差异。如果你认为工具参数的选择应当获得与激发它的推理不同的信用——而Qian et al. (2025) 这类奖励分解研究的证据表明确实如此——那就需要词元级或片段级的优势，而(16.12)给不了。

2. 折扣意味着什么。 γ_{tok} 按词元折扣，γ_{turn} 按轮次折扣。设想一条轨迹含有一轮 5,000 词元、外加 39 轮每轮 20 词元。在轮次级，那个长轮与其他任何一轮一样被折扣。在词元级、取 $\gamma_{\text{tok}} = 0.999$ 时，终局奖励传到该长轮第一个词元处，相对于其最后一个词元被衰减了 $0.999^{5000} \approx 6.7 \times 10^{-3}$。由于轮长是策略自己选的，词元级折扣是对啰嗦轮次的惩罚，而轮次级折扣是对轮次过多的惩罚。这是两种不同的行为激励，而任务本身两者都不蕴含。

3. 归一化因子。 还是这条轨迹，动作词元共计 $5000 + 39 \times 20 = 5{,}780$ 个。逐词元取平均，会让那一个长轮占到整条轨迹梯度权重的 $5000/5780 = 86.5\%$；逐轮取平均，则只给它 $1/40 = 2.5\%$。同一个决策的相对权重相差 35 倍，而这完全是由分母的选择造成的。节 12.2已经表明归一化因子的选择会可测量地改变单轮结果；在智能体式设定中，同一条轨迹内的轮长可以相差两个数量级，它的影响就成了主导。

4. 信赖域安放在哪里。 词元级的截断比值约束的是每个词元的贡献；轮次级比值——一轮内各词元比值之积，或其几何平均——约束的则是每个决策。既然轮次才是环境作出响应的单位，轮次级信赖域可以说是结构上正确的对象。这个问题并非只是装点门面：按 节 2.1的论证，比值之积的动态范围随片段长度 n 以 $e^{\delta\sqrt{n}}$ 增长，因此词元级、轮次级与轨迹级比值在触发截断的频率上相差若干数量级。

5. GAE 是否还有意义。 这是最微妙、也最少被明说的一点。广义优势估计 (Schulman et al., 2016) 累加时序差分残差 $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$。在一个动作片段内部，$s_{t+1} = s_t \oplus a_t$ 是确定的，因此在 $r_t = 0$ 且 $\gamma = 1$ 时有 $\delta_t = V(c_t \oplus a_t) - V(c_t)$：该残差度量的是策略发出一个词元时 评论家自己的估计移动了多少。这确实是关于该词元价值的真实信息，但它完全是内部的——没有任何关于世界的新事实进来。而在一个轮次边界上，$V(h \oplus a \oplus o) - V(h \oplus a)$ 度量的是环境的响应：真正的外部意外。因此词元级 GAE 平滑的是评论家的意见，而轮次级 GAE 传播的是环境意外，λ 在两种情形下含义并不相同。在表 16.1那条轨迹上以 $\lambda = 0.95$ 运行词元级 GAE，有效信用窗口大约是 20 个词元，够不到上一个轮次边界；无论超参数怎么写，该算法在结构上就无力跨轮分配信用。

表 16.2: 两种表述的并排对照。右栏是左栏的半马尔可夫抽象；命题 16.3给出了两者策略梯度一致的确切条件。标有 † 的行，是两者会作出实质不同预测、而不仅仅是提供不同记号的地方。

	词元级 POMDP	**轮次级 MDP**
状态	(c_t, e_t)；策略看到 c_t	$(h^{(k)}, e^{(k)})$；策略看到 $h^{(k)}$
动作	$\mathcal{U}$ 中的一个词元	一条完整消息 $a^{(k)} \in \mathcal{U}^*$
转移	拼接，工具调用结束处除外	每一步都是环境核
时域	$\sum_k n_k$ 个动作词元	K 轮
奖励位置	终局，或一轮的最后一个词元处	终局，或每轮 $r^{(k)}$
折扣 †	惩罚长轮次	惩罚多轮次
优势来源 †	评论家，或广播的轮次值	逐轮回报，组相对或评论家
基线位置	逐词元 $V(c_t)$	逐轮 $V(h^{(k)})$
信赖域 †	逐词元截断比值	逐轮比值
GAE†	平滑评论家残差	传播环境意外
无法表达	—	轮内的信用差异
估计量成本	训练器本来就已支付	需要存储轮次片段
典型用途	实际计算的损失	优势如何构造

常见陷阱: 在一篇智能体式论文里报告「$\gamma = 0.95$」

有三个量经常被混为一谈：逐词元折扣、逐轮折扣，以及不折扣但在奖励中加入一项逐轮成本。它们蕴含着不同的最优策略。逐词元折扣惩罚啰嗦；逐轮折扣惩罚行动；逐轮成本则以加法方式惩罚行动，而正如节 16.6.2所示，三者之中只有它对应的停止规则不依赖于策略当前表现得好不好。请说明你用的是哪一种，并优先选第三种。

工程实践: 2026 年多数配方所收敛到的默认设置

词元级损失；$\gamma_{\text{tok}} = 1$；优势在各轮之间分段常值，由轨迹级的组相对比较给出，可选地再加一个轮次级修正（节 16.5）；不折扣，任何偏好更少轮次的意图都以奖励中一项显式的加性逐轮成本来表达；在词元级做截断，且按 DAPO(Yu et al., 2025c) 的做法非对称地截断。这并不是在断言该默认设置是最优的——而是在断言，它是失效模式被刻画得最清楚的那种配置，因而是正确的、值得去超越的基线。

16.4 观测掩码

命题 16.2已经告诉我们，策略梯度在观测词元上没有任何项。本节要问的是：如果你偏偏在那里放了一项会怎样——这正是把单轮训练器朴素扩展过来时会做的事——然后给出避免它的构造。

16.4.1 把观测词元算进来实际上在优化什么

取模板(6.17)，让求和遍历所有位置而不只是 $\mathcal{I}_a$。多出来的项，对每个 $t \in \mathcal{I}_o$ 而言是

$$A_t \nabla_\theta \log \pi_\theta(o_t \mid c_{<t}), \tag{16.15}$$

在期望意义下它们是 $\mathbb{E}[A \cdot \log \pi_\theta(o \mid h, a)]$ 的梯度。这个对象是有名字的：它是在环境文本上按优势加权的极大似然。既然成功轨迹的 $A > 0$，这一更新就会提高模型赋予它在成功采样轨迹上恰好收到的那些工具输出的概率。

由此产生两类危害，二者性质截然不同。

它在优化策略无法控制的东西。 一条检索结果的似然是搜索引擎的性质。抬高它会消耗容量，并沿着由环境输出分布、而非由奖励所选定的方向移动参数。在表 16.1的预算下，这些虚假项的数量几乎是真实项的四倍，所以这不是一个扰动——一个没有掩码的智能体式损失，主要是一个作用在工具输出上、带有奖励塑形权重的语言建模损失。

它在教模型自己写观测。 这是会在评测中暴露出来的那种失效。策略被训练在这样的上下文上：工具结果定界符之后跟着看似合理的工具输出，而且它因给这些输出赋予高概率而受奖励。在推理时，没有任何东西阻止它先发出定界符、然后接着编造一段观测，因为那恰恰是它被训练去认为很可能出现的续写。一旦知道该找什么，这种经验特征就非常好认：模型写下工具调用，写下一段格式完美却完全虚构的结果，再自信满满地据此推理。Jin et al. (2025a) 正是出于这个原因引入了掩码，并称之为检索词元掩码。

第三类危害是量化上的。观测词元按构造对模型来说是分布外的，因此其对数概率既低又不稳定。把它们纳入任何比值之积，都会极大地放大该乘积的动态范围（节 2.1）；把它们纳入逐词元 KL 惩罚，则是在因为模型没能预测它从未被要求产生的文本而惩罚它。Xue et al. (2025) 对由此产生的不稳定给出了最锐利的因果说明：以分布外的工具反馈为条件会产生低概率词元，这些词元跨轮次不断累积，最终导致梯度范数爆炸。

历史注记: 掩码，及其在 2026 年的部分反转

检索词元掩码 (Jin et al., 2025a) 是本领域第一条智能体式惯例，两年间被视为定论——尽管从来算不上全体一致：RAGEN(Wang et al., 2025i) 在其主要设定中刻意不对环境词元加掩码，而其附录却发现回复掩码是有益的。2026 年有两篇独立的论文主张，这条惯例扔掉了采样轨迹中最丰富的信号。ECHO (2026) 与PaW (2026) 都把观测词元排除在策略梯度项之外，但在其上添加一个普通的下一词元预测损失，理由是：无论轨迹是否获得奖励，每一步交互都会产生一个观测，因此环境预测损失恰恰在奖励稀疏之处是稠密的。而且它几乎是免费的：模型本来就以这些词元为条件，也已经在同一次前向中算出了它们的 logits。ECHO (2026) 报告在 8B 与 14B 两个规模上，Terminal-Bench 2.0 的 pass@1 相对 GRPO 翻了一番；更引人注目的是，在某些设定下，把奖励关掉、仅用环境预测损失，就能改善留出的分布外任务。PaW (2026) 指出，因果注意力保证了这个辅助损失不可能改变动作 logits，因此策略学习可证明不受影响。

不要把这两个论断混为一谈。观测词元必须排除在策略梯度之外，这由 命题 16.2给出定论，属于数学。而它们应当承载一个不同的目标函数，则是 2026 年在终端与工具使用基准上的经验结果，迄今仅有两篇论文支持，而且这是一个变化很快的角落。

16.4.2 是三个掩码，不是一个

表 16.3罗列了这些类别。关于它有三点值得挑明。

注意力掩码与损失掩码是两个不同的掩码。 观测词元会被其后每一个位置注意到——这正是交互的全部意义所在——却不出现在任何损失项中。在多轮对话数据上做 SFT 时，一个 `loss_mask` 就够了，因为唯一的区分是提示与补全。现在多出了第三类。

运行框架也会插入词元。 在一段观测与下一个动作片段之间，服务栈会写入一个策略并未采样过的角色标头。如果轨迹是靠重新渲染消息来重建的，这类词元与动作词元就无从区分；而它们并不是从任何分布中采样出来的，所以对它们训练就是在对一个常量训练。请在发出时就记录片段，而不要事后去推断（代码 16.3）。

表 16.3: 各类词元分别参与哪些计算。会引发缺陷的是这两行：运行框架的模板词元（由服务栈预填，因而不是采样得到的，因而不是动作），以及观测词元（被注意到，却从不被训练）。「比值」指(16.9)的重要性比值；「KL」指对 π_{ref} 的逐词元散度惩罚。

词元类别	注意	损失	比值	归一化	KL
提示／系统／工具模式	是	否	否	否	否
运行框架插入的角色标头	是	否	否	否	否
策略的思考词元	是	是	是	是	是
策略的工具调用语法与参数	是	是	是	是	是
工具结果定界符（运行框架）	是	否	否	否	否
由 $\mathcal{E}$ 写入的观测词元	是	否 *	否	否	否
填充	否	否	否	否	否
被截断的最后一轮的词元	是	由策略定	由策略定	由策略定	由策略定

* 由命题 16.2排除在策略梯度项之外；可以承载一个独立的下一词元预测损失 (ECHO, 2026; PaW, 2026)。「由策略定」意思是该处理方式是一项由 节 16.6支配的设计选择，而不是推导结果。

KL 惩罚同样需要掩码。 一个在所有位置上计算的逐词元反向 KL 惩罚，会因为 π_θ 在对工具输出的预测上偏离 π_{ref} 而惩罚它——那是一种真实的偏移，而且并不是你想压制的那种。请用 m_t 给 KL 加掩码。

16.4.3 正确构造掩码

代码 16.2: 掩码构造与加掩码的词元对数概率。多数实现最容易弄错的就是这一行对齐：`logits[t-1]` 预测的是 `ids[t]`，因此损失掩码必须切成 `trainable[1:]`（对齐到目标），而绝不能切成 `trainable[:-1]`。切错时在轮次边界上会发生什么，见下面的陷阱。

```
ACTION = "action"   # 角色："prompt" | "template" | "action" | "observation"

def build_masks(segments):
    """segments: [(role, token_ids)]，按一条轨迹的发出顺序排列。"""
    ids, trainable, spans, cursor = [], [], [], 0
    for role, seg in segments:
        ids.extend(seg)
        trainable.extend([role == ACTION] * len(seg))
        if role == ACTION:
            spans.append((cursor, cursor + len(seg)))   # 左闭右开，在 ids 空间中
        cursor += len(seg)
    ids = torch.tensor(ids)                             # [L]
```

```
    trainable = torch.tensor(trainable, dtype=torch.bool)
    target_mask = trainable[1:]                          # [L-1]，对齐到目标
    return ids, target_mask, spans

def masked_logprobs(model, ids, target_mask):
    logits = model(ids[None, :-1]).logits.float()        # fp32 log-softmax：见第 22 章
    logp = torch.log_softmax(logits, dim=-1)
    lp = logp.gather(-1, ids[None, 1:, None]).squeeze(-1).squeeze(0)   # [L-1]
    return lp * target_mask, int(target_mask.sum())
```

常见陷阱：轮次边界上的差一错误，以及它为何不可见

把掩码切成 `trainable[:-1]` 而不是 `trainable[1:]`，每一处掩码边界都会移动一个位置。在一个长动作片段内部，这只在两端各损失一个词元，别无他害，所以单轮训练看起来一切正常。而在轮次边界上，后果既具体又糟糕。一段观测的最后一个词元变成可训练的了——你现在是在对环境文本做训练，并承担式 (16.15)的全部后果——而下一个动作片段的 第一个词元则掉出了损失。那第一个词元可不是一个普通词元：它正是决定这一轮将是又一次工具调用还是最终答案的那个。在 40 轮的情形下，你悄无声息地丢掉了整条轨迹中杠杆最大的 40 个梯度项，还用 40 个把模型拉向记忆工具输出的项取而代之。

这个缺陷在损失曲线上不可见，因为 6,250 个位置中错位 39 个，不会让损失有任何可测量的变化。它只在两个地方可见：可训练词元的计数（它不会变，所以别指望它），以及一个单元测试——构造一条两轮轨迹，其中含有一个有辨识度的观测词元，并断言其梯度恰好为零。请写这个测试。第二个值得写的测试是断言：可训练词元数等于采样器报告其生成的词元数——这个等式还能抓出模板词元污染与重新分词的漂移 (Cheng et al., 2025)。

16.4.4 重要性比值，以及序列级比值的含混

由(16.9)，轨迹比值只是在 $\mathcal{J}_a$ 上的乘积。因此词元级截断可以原样迁移过来：在动作位置上计算比值，截断，加掩码，归一化。序列级的构造则迁移不过来，其原因值得讲清楚，因为这是文献中一处真正的空白，而非一个实现细节。

GSPO(Zheng et al., 2025a) 用长度归一化的序列比值 $\rho_i = (\pi_\theta(y^{(i)})/\pi_{\theta_{\text{old}}}(y^{(i)}))^{1/|y^{(i)}|}$ 取代词元比值，理由是词元比值是无法修正的噪声，而序列比值才是信赖域应当约束的量。可是在一条智能体式轨迹里，并不存在唯一的「序列」。至少有三种定义可用，而且它们并不等价：

1. **整条轨迹上所有动作词元的几何平均**。这是一个有良好定义的数，但它把 40 个决策平均成了一个标量。某一轮的比值变动了 2 倍，只会让整条轨迹的几何平均移动 $2^{150/6250} \approx 1.017$，任何合理的截断区间都抓不住。信赖域恰恰在时域变长时变得空洞。
2. **逐轮几何平均，逐轮截断**。这是一个轮次级信赖域，而按 命题 16.3它是一个具有不同隐含粒度的不同算法——一个合理的算法，但它应当被如此命名，而不是被描述成「多轮版的 GSPO」。
3. **把观测词元也算进序列**。数学上可以定义，概念上是错的：这样得到的比值会随着模型对工具输出的预测发生漂移而变动，于是信赖域的松紧取决于策略隐含的世界模型在如何变化，而这与策略的行为改变了多少毫无关系。

据我们所知，尚无对这三者的系统比较；而那些报告在多轮设定下使用序列级比值的论文，也往往不说明自己指的是哪一种。上面的论证是结构性的而非经验性的，读者应当把它当作一个有待检验的预测，而不是一个结论。

常见陷阱: 加了掩码的比值不是被掩码分布的比值

$\prod_{t\in\mathcal{I}_a} \pi_\theta(y_t \mid c_{<t})$ 不是任何东西上的概率分布：它是给定已实现观测时动作词元的条件似然。它在动作序列上并不求和为一，因为不同的动作序列本会引出不同的观测。第6章中所有只需要得分函数恒等式的内容都能挺过这一点，因为 命题 16.2是在联合密度上证明的。而任何需要 $\sum_y \pi_\theta(y) = 1$ 的内容——例如把一个到 π_{ref} 的序列级 KL 解释为回复分布之间的散度——则挺不过来，在被信任之前应当被重新解释为给定观测流的条件散度。

16.5 跨轮次的信用分配

本章负责表述；信用分配的一般理论归第13章，长时域算法归第18章。属于这里的内容是：把一个轮次结构化的优势向量可以是什么一一列举出来，并说明每种选择让你承诺了什么。

（a）轨迹级均匀广播。对每条轨迹算一个组相对优势，并广播到它的每一个动作词元上：$A_t = m_t \hat{A}^{(g)}$，其中在组内取 $\hat{A}^{(g)} = (r^{(g)} - \mu_r)/\sigma_r$。Jin et al. (2025a)、Chen et al. (2025b)、Li et al. (2025g) 以及多数生产配方都是这么做的，而按 命题 16.3，它是 $\gamma_{\text{turn}} = 1$ 且轮次优势为常值的特例。它的性质正是 表 13.1最后一行所列：完全没有

估计量偏差，也完全没有轨迹内分辨率。一条 100 轮轨迹的每一轮都得到相同的信用，包括那没问题的 99 轮和把事情搞砸的那一轮。

（b）轮次级奖励。 如果环境能给轮次打分——一个新通过的测试、一次返回了相关文档的检索、一条退出码为零的 shell 命令——那么就可以用 $A^{(k)} = \sum_{k' \geq k} \gamma_{\text{turn}}^{k'-k} r^{(k')} - b(h^{(k)})$。Wei et al. (2025b) 报告，把轮次级奖励纳入 GRPO 与 PPO 后，稳定性更好、收敛更快、准确率也高于轨迹级基线；Wang et al. (2025j) 则报告，具体到检索任务，步级监督胜过轨迹级监督。代价在于，每一个逐轮奖励都是塑形奖励，而塑形奖励在这里的历史记录比在单轮 RLVR 中更差：Li et al. (2025g) 发现，惩罚不可执行的代码并没有帮助，而且很可能让模型在生成代码这件事上变得保守——这是已发表文献中关于「轮次级塑形可能压制你想引出的那个行为」最清楚的一次警告。

（c）只给最后一轮。 只给最终答案片段分配优势。这不是什么稀奇做法：如果你的掩码是从「最后一条助手消息」推导出来的，你就会意外地得到它。它产生的梯度训练的是从一段长上下文到一个最终答案的映射，对该调用哪些工具、何时停止则完全不提供信号。它把问题化归为一个提示异常长且由自己生成的单轮 RLVR，而且它在原则上就学不到工具策略，而不只是学得慢。

（d）嵌套的组相对比较。 GiGPO(Feng et al., 2025b) 是这一领域中表述层面最干净的想法。保留回合级的组相对优势 A_E，再加一个步级优势：扫描 G 条轨迹，找出在它们之间重复出现的环境状态，把从每个重复状态出发的（动作，回报）对各自收集成一个组，并在组内计算组相对优势，得到 $\hat{A} = A_E + \omega A_S$。这个构造把一个动作的效果与状态的效果分离开来——这正是评论家所做的事——却只用了对已访问状态的哈希表查找，既不需要评论家，也不需要额外采样。在 7B 上相对 GRPO 报告的增益是 ALFWorld 上 +12.6%、WebShop 成功率上 +9.1%，在两者上都胜过带评论家的 PPO。

需要内化的是表述层面的那条要求：步级分组需要一个环境状态上的等价关系，而且需要状态真的会重复出现。这在 ALFWorld 与 WebShop 中成立，那里状态空间小且可哈希。在开放网页浏览与长的软件工程轨迹中它就不成立了，那里任何两条采样轨迹都不会重访同一个状态。当重复性失效时，退路是按轨迹之间共享的别的东西来分组——轮次序号 (Zhou et al., 2026)、任务或领域 (Zhang et al., 2025b)，或环境实例 (Cai et al., 2025)。这三者是同一招：在比提示更宽的组上归一化，使得任何单个离群值或单个领域都无法主导这次更新。

（e）选择性排除。 最后一种选项不是一种优势构造，而是对哪些轨迹贡献梯度的过滤。Xue et al. (2025) 辨认出空轮——一个既没产生工具调用也没产生最终答案的轮次，典型表现为输出残缺、重复，或过早的 <eos>——并把含有空轮的轨迹从策略损失中剔除，同时把它们保留在优势计算中。他们从 Qwen2.5-7B 基座出发的 AIME24 结果是 22.1 升到 50.5，对一个过滤器来说这是很大的效应。

这个表述值得精确陈述，因为它的措辞容易被误读。把空轨迹保留在组统计量中，保持的是基线的无偏；它并不会让梯度无偏，因为把一个非随机的轨迹子集排除出损失，按构造就使梯度有偏。而这一偏差正是其机制、而非其缺陷：被排除的那些项，恰恰是量级病态、信用归属错误的那些。用一个已知的偏差去换取一大块方差缩减，正是节 1.7中的那种取舍，而这样陈述之后也更容易被理解。

工程实践：该先拿哪一个

从（a）起步，配上正确的掩码与正确的轮次片段。在添加任何优势方面的精巧之前，先加上（e）——把格式异常的轨迹从损失中过滤掉——因为格式异常的轨迹是梯度范数病态的最大单一来源，而这个过滤器不花什么代价。只有当你的环境具有可哈希的重复状态时，才去拿（d）。最后才拿（b），而且必须配一项留出检查，确认塑形没有压制它本想鼓励的那个行为。

16.6 时域是内生的

在第三部分的每一个设定中，时域都是问题本身的属性：策略发出 <eos> 时一条回复就结束了，唯一的结构性控制是长度上限。而在智能体式设定中，策略在每一轮都决定是否再行动一次，因此 K 是一个分布会在训练中漂移的随机变量。由此产生三项后果。

16.6.1 轮数漂移对奖励是中性的

在只有结果奖励且 $\gamma_{\text{turn}} = 1$ 时，目标函数对于插入任意多个不改变最终答案的轮次是不变的。因此轮数被目标函数无法辨识：J 中没有任何东西偏好三次调用的解法胜过三十次调用却得到同一答案的解法。于是漂移方向就由损失中的其余一切来决定——归一化因子、截断的非对称性、再多调用一次可能有帮助的那点微小概率——而经验上它是向上漂的，其原因与节 12.7中回复长度向上漂移的原因相同。

这个量级并不小。Wang et al. (2025a) 在奖励中加入一项工具调用计数项，报告在准确率相当的情况下工具调用减少了 68.3%，而「工具生产率」（每次调用产出的正确

答案数）提高了 215.4%，这直接说明未定价的最优解所用的调用次数大约是其所需的三倍。Yu et al. (2025d) 从相反方向得到一个趋同的结论：一种调用更少的审慎策略，胜过频繁调用或啰嗦的自我推理。Shen et al. (2026b) 在合成的终端环境中记录了其病态极限：通过过量生成工具调用来实现奖励作弊。

16.6.2 带逐轮成本时的最优停止

自然的修补是给每一轮加一项加性成本。它有一个干净的最优停止刻画，值得推导出来，因为这把一个超参数转换成了关于部署经济性的陈述。

推导：停止规则，以及加性为何胜过乘性

在上下文 h 处，策略可以停止——现在就给出答案，获得期望终局奖励 $u(h) = \mathbb{P}[\text{答案正确} \mid h]$——或者继续，付出成本 $c > 0$ 并到达一个后继上下文。以 V 记最优值，

$$V(h) = \max\Big\{ u(h),\ -c + \mathop{\mathbb{E}}_{a\sim\pi^\star,\, o\sim\mathcal{E}}\big[V(h \oplus a \oplus o)\big] \Big\}. \tag{16.16}$$

继续是最优的，当且仅当

$$\mathbb{E}\big[V(h \oplus a \oplus o)\big] - u(h) \ > \ c. \tag{16.17}$$

所以 c 是一个兑换率：只要再做一次工具调用能把最终答对的概率提高 c 以上，最优策略就会做这次调用。取 $c = 0.01$ 就是在说「如果一次工具调用能买来一个百分点以上的准确率，它就值得做」，这是一个关于工具相对于一次错误的代价的论断——部署方知道这个量，而超参数搜索不知道。还要注意，(16.17) 比较的是 V 而不是 u：一次本身没有帮助、却使后续某次调用成为可能的调用，仍然会被采纳，所以这条规则并不短视。

对比一个乘性的逐轮折扣 γ_{turn}。此时贝尔曼方程变为 $V(h) = \max\{u(h),\, \gamma_{\text{turn}}\ \mathbb{E}[V(\cdot)]\}$，继续是最优的条件为

$$\mathbb{E}\big[V(h \oplus a \oplus o)\big] - u(h) \ > \ \frac{1 - \gamma_{\text{turn}}}{\gamma_{\text{turn}}}\, u(h) + (\text{高阶项}), \tag{16.18}$$

i.e., 这个阈值随策略当前做得有多好而按比例变化。折扣使得工具调用恰恰在策略已经很可能答对时变贵、在它已经迷失时变便宜——这可以说是反的，因为一个自信的策略本来就没多少可从再一次调用中获得，而一个迷失的策略则大有可得。如果你想给「行动」定价，请用加法定价。

因此，加性的逐轮成本是本书中少数几个其对最优策略的影响被完全刻画清楚的塑形奖励之一：它把停止边界按成功概率的单位恰好平移 c（(16.17)），此外别无作用。这是比任何过程奖励所能提供的都更强的保证，也正是为什么轮次成本是安全的，而逐步的正确性奖金则不是。

16.6.3 在轮数上限处截断会注入相关噪声

每一个智能体式训练器都会设一个轮数上限和一个上下文上限。撞上其中任何一个的轨迹都会被赋予某个奖励——通常是零，有时是一个惩罚——而这个赋值是对世界作出的一个错误论断。「没能在 40 轮之内完成」是证据缺失，不是答案错误的证据。

这与节 12.7中的序列截断在结构上完全相同，但多了一层加重。撞上轮数上限的，集中在最需要多轮的那些提示上，也就是难题，所以注入的噪声与难度相关。这样训练出来的策略学到的是：在难题上坚持会遭到惩罚——这与本想教给它的道理恰好相反，而且它是通过与长度惩罚教会模型过早作答的同一机制学到的。两种处理方式与表 12.4相对应：要么把被截断的轨迹整个从损失中掩掉（当作证据缺失处理，同时如上面的选项 (e) 那样保留它们用于组统计量），要么按(12.12)的精神对轮数施加一个软斜坡，使边界不再是一道悬崖。掩掉是更安全的默认做法，也是我们推荐的做法——直到你测量了轮数分布、知道上限究竟卡在哪里为止。

16.7 新的失效模式

表 12.1编目了单轮组相对训练的各种病理。表 16.4是智能体式的补充。它不是一份「可能出错的事情」的清单——那份清单没有边界——而是一份机制已被辨明、诊断手段也已确定的失效清单，主要取自Zhang et al. (2026b) 所综述的不稳定性文献。

其中三条值得在表格之外再说几句。

轮数坍缩与格式坍缩是同一种失效在不同粒度上的表现。 两者都源于结果奖励给一个回报被延迟的行为赋了负优势，也都由同一类干预来化解：让策略保持探索足够久，久到那份回报能够兑现。Wang et al. (2025i) 的 StarPO-S 干脆丢掉了 KL 正则项以允许探索，采用了 DAPO 的非对称截断，并按结果不确定性来筛选训练任务。这条移除 KL 锚的建议应当对照 节 12.8谨慎阅读：对基于验证器的训练它站得住，对基于奖励模型的训练它明显是错的，而智能体式强化学习绝大多数属于前者。

熵不是一个充分的诊断指标。 这是智能体稳定性文献中最重要的一个否定结果。Wang et al. (2026f) 表明，模型可以在维持健康的输入内熵的同时，产生与输入无关、只是看上去多样的输出，并提出如下分解

$$\underbrace{\text{总多样性}}_{\text{你想要的}} = \underbrace{\text{熵}}_{\text{输入内的变化}} + \underbrace{\text{互信息}}_{\text{跨输入的可区分性}} . \tag{16.19}$$

熵奖励只针对第一项，这恰恰解释了它们为何无法防止模板坍缩——这是节 12.6中已经提出的那个反对熵奖励的论证的更锐利版本。如果从单轮训练转向智能体式训练时你只肯新记录一个标量，那就记录第二项的一个代理量。

经由工具的奖励作弊，与经由奖励模型的奖励作弊在范畴上不同。 一个学出来的奖励模型是靠找到它出错的输入来被钻空子的。而一个工具是靠改变验证器所检查的那个世界来被钻空子的。如果智能体有 shell 权限而验证器读取某个文件，那么把预期答案写进那个文件并不是对模型误差的利用——按照规格所述，它就是这个问题的一个正确解。优化器里没有任何东西能把它与预期解法区分开来，也没有任何 KL 锚能阻止它。唯一的防线在环境这一侧：从智能体够不到的状态去验证、在一个独立的沙箱中运行验证器，以及人工审计采样轨迹（节 5.8.4）。第19章把这当作一个一等的设计问题来处理。

16.8 关于时域的课程

节 12.4.1论证过，RLVR 中的课程本质上是要把提示保持在成功率中等的那个区间内，那里的组相对优势才不退化。在智能体式设定中还有第二个旋钮可用，而且有理由更偏好它。

时域课程是在训练过程中逐步提高交互预算。AgentGym-RL(Xi et al., 2025) 中的 ScalingInter-RL 是已发表的最干净的实例：把训练切成 N 个等长阶段，预算单调递增 $h_1 < \cdots < h_N$；在他们的配置中 $N = 3$，每阶段 80 次迭代，预算依次为 8、12、15 轮。他们的发现是违反直觉的，也正是这个想法值得认真对待的理由：大的交互预算起初会提升性能，随后坍缩；而限制早期探索反而给出更干净的学习信号，从而支撑后期稳定的长时域学习。Wang et al. (2025i) 独立地发现，中等的交互预算胜过很小和很大的预算。

有三条结构性理由，使得在这一设定中时域比难度是更好的课程变量。

1. **它不需要知道难度**。难度课程需要逐提示的成功率估计，而这要为每个提示付出一整组采样轨迹的代价——而一条智能体式采样轨迹比单轮的贵上一到两个数量级。而一个时域上限是免费的。
2. **它在源头上同时约束了方差与漂移**。得分函数估计量的二阶矩随着为一个轨迹级优势作贡献的动作词元数增长，而Xue et al. (2025) 给出的失效说明是分布外漂移跨轮累积。一个轮数预算直接约束了作贡献的项数与累积的步数，这比事后截断所产生的梯度要有针对性得多。
3. **短时域的胜任是先决条件，而简单任务的胜任并不是**。一个连三轮交互都做不干净的策略，不会做得好三十轮的交互；技能是嵌套的。而在简单问题上的成功并不显然嵌套于在困难问题上的成功之中，这正是为什么即便在单轮情形下难度课程也仍有争议。

这些告诫同样有力。已发表的结果为数不多，且集中在时域为数十轮而非数百轮的结构化文字游戏环境中，也没有任何消融把是哪一种机制在起作用分离出来。Xi et al. (2025) 自己就报告，强化学习的增益在结构化的、基于规则的环境中最大，而在 WebArena 这类嘈杂的现实环境中最小；也就是说，这个课程是在「什么都能 work」的那种设定里得到验证的。还有一种时域课程特有的失效模式：把预算压到低于任务内在所需的轮数，会使任务无解，从而产生全零的组、根本没有梯度，这与课程本该起的作用恰好相反。日程表必须与任务集中内在轮数需求的分布相匹配，这意味着要先把那个分布测出来。

在实践中，这意味着先在初始策略已经能解的那些任务上，测量成功所需轮数的分布，然后把 h_1 设在该分布第 75 百分位附近、不要更低，好让第一阶段对那些已在能力范围内的任务是宽裕的。再分两到三步提高到上下文预算所允许的上限，并在每次切换时检查退化的全零组所占比例有没有上升。如果上升了，说明预算低于刚刚被纳入的那些任务的内在需求，这次训练是在对噪声训练。

16.9 一个最小的智能体式训练循环

以上一切现在可以组装起来了。算法 16.1收集一条轨迹，并附带训练器将会需要的各项簿记；算法 16.2是一次训练迭代。

代码 16.3: 采样轨迹记录。设计准则是：这条记录才是权威表示；掩码、比值与优势都从它导出，而原始聊天消息永不重新渲染。存下采样器给出的 `action_logprobs`，正是使第22章中的失配修正成为可能的前提。

```python
@dataclass
class Turn:
    action_ids: list[int]          # 恰好是采样器发出的那些 id
    action_logprobs: list[float]   # 来自采样器，绝不重算
    template_ids: list[int]        # 运行框架标头，预填的：不是动作
    obs_ids: list[int]             # 恰好是环境追加的那些 id
    tool: str | None
    status: str                    # "ok" | "error" | "timeout" | "void"
    reward: float = 0.0            # 逐轮奖励，如果有的话

@dataclass
class Trajectory:
    prompt_ids: list[int]
    turns: list[Turn]
    outcome_reward: float
    truncated_by: str | None       # "turn_cap" | "context_cap" | None
    env_id: str                    # 哪个实例；用于复现
    env_seed: int                  # 确定性旋钮；见环境那一章
    sampler_version: str           # 引擎 + 权重哈希；采样失配

    def action_mask(self) -> torch.Tensor:
        """对齐到目标，与 build_masks 完全一致：注意 flags[1:]。"""
        flags = [False] * len(self.prompt_ids)
        for t in self.turns:
            flags += [False] * len(t.template_ids)
            flags += [True] * len(t.action_ids)
            flags += [False] * len(t.obs_ids)
        return torch.tensor(flags[1:], dtype=torch.bool)
```

工程实践：以前不记、现在要记的东西

至少要加上：K 的均值与第 95 百分位；每轮的工具调用率与解析失败率；至少含一个空轮的轨迹比例；按轮数上限与上下文上限分开统计的截断率；每条轨迹的可训练词元数（组内的均值与方差，它能检测出压缩引起的不齐性）；动作位置上采样器对数概率与重算对数概率的不一致程度；以及按工具分列的状态计数。最后这一项能抓出一类在别处完全看不见的失效——某个工具悄无声息地连续几小时返回错误，而奖励在缓慢劣化。

16.10 第四部分路线图

随后的四章以本章的表述为既定前提，各自攻击其中一个尚未解决的部分。

第17章处理短时域情形，其中工具是对一个单一目标的推理回合的增强：检索、搜索、代码解释器、计算器。时域是一到二十轮，奖励通常是一次答案检查，因此起约束作用的是奖励设计——Qian et al. (2025) 是「工具使用奖励应当如何分解」的参考文献——以及训练稳定性。效率问题也归属在那里 (Wang et al., 2025a; Yu et al., 2025d)。

第18章处理智能体在数十到数百轮之间改变持久外部状态的情形，此时信用分配与上下文管理成为主导。本章只作了表述的那些算法归它所有：嵌套的组相对比较、轮次级值估计、压缩与「记忆即动作」，以及那个在 2026 年被重新提起的问题——当压缩使得组间比较不再有良好定义时，评论家是否会回归。

第19章把环境本身当作研究对象而非背景：环境的构造与合成、验证器质量、确定性与隔离，以及节 16.7只能点到为止的「经由工具的奖励作弊」问题。到 2026 年年中，本领域的重心正落在这里，其观点是环境多样性比算法上的精进更有价值 (Tu et al., 2026; Shen et al., 2026b)。

第20章把决策过程推广到多个策略。形式上的变化是环境变得非平稳——另一个智能体的策略更新就是你的动力学的一次改变——而信用分配也多出了第二个坐标轴，除了跨轮次之外还要跨智能体 (Chen et al., 2026b)。那一章还会讨论一种反向的着法：把一个多智能体系统蒸馏进单个模型 (Li et al., 2025e)，截至 2026 年年中，这是计算上更高效的路径。

本章小结

- 智能体化改变了被优化的对象。轨迹是策略因子与环境因子的联合乘积 (16.7)；只有前者依赖 θ，而第三部分中每个算法所依赖的假设 $y \sim \pi_\theta(\cdot|x)$ 不再成立。
- 语言模型 MDP 的五处退化性质中有三处不再成立。动力学不再确定也不再已知，因此从一个前缀出发的蒙特卡洛值估计既不便宜也不无偏；「所有词元」不再是一个有良好定义的求和范围；而折扣因子从「只是不明智」变成了「含混」。
- 这一设定是 POMDP 而非 MDP：策略以可见上下文为条件，而动力学与奖励依赖隐藏的环境状态。这使得每一个拟合出来的值函数都带有不可约的残差方差，也使每一个组相对优势都被环境的运气所污染。
- 环境因子不含得分项（命题 16.2），因此观测掩码是一个推导结果而不是一条启发式规则。把损失项加在观测词元上，优化的是环境文本上按优势加权的极大似然，其可观测后果是一个在推理时编造工具输出的策略。

- 注意力掩码、损失掩码与比值掩码是同一个张量上的三个不同掩码，而损失掩码必须对齐到目标而不是输入。轮次边界上的差一错误会让训练发生在环境词元上，并悄无声息地丢掉每一轮中杠杆最大的那个决策词元。
- 轮次级 MDP 是词元级 POMDP 的半马尔可夫抽象。两者的策略梯度恰在词元折扣为 1 且优势在各轮之间分段常值时一致（命题 16.3）；这正是「轮次结构化优势配词元级损失」成为主流设计的原因。两者在以下方面分道扬镳：轮内分辨率、折扣的含义、归一化因子、信赖域安放在哪里，以及 GAE 传播的究竟是环境意外还是仅仅是评论家噪声。
- 时域是内生的且未被定价。在只有结果奖励时，轮数无法被目标函数辨识，并且会向上漂移；一项加性的逐轮成本 c 有精确的最优停止解释，即准确率与工具调用之间的兑换率 (16.17)，而一个乘性折扣则恰恰在策略已经表现良好时才让调用变贵。
- 在轮数上限处的截断注入了与任务难度相关的噪声，教给策略的是「在难题上坚持会挨罚」。请把被截断的轨迹当作证据缺失，而不是当作失败。
- 智能体式训练有它自己的、机制已被辨明的病理（表 16.4）。对其中最重要的那一个而言，熵是一个不充分的诊断指标：一个策略可以在保持熵稳定的同时产生与输入无关的输出，因此必须单独测量跨输入的可区分性(16.19)。
- 在这里，时域课程比难度课程更自然，因为时域控制起来免费、能直接约束方差与漂移，并且索引着一个嵌套的技能层级。支持它的证据是真实的，但很薄，来自少数几个结构化环境。

延伸阅读

先读Zhang et al. (2026b)，并且要把它当作一张地图而不是一篇论证来读：当本领域说「智能体式强化学习」时，引用的就是这篇综述，它综合了数百项工作，而它把基于偏好的微调所对应的退化 MDP 与智能体式强化学习所对应的 POMDP 加以对照的框架，正是本章所形式化的那个。它的能力分类法与领域分类法，是你在读别的任何东西之前用来定位自己的正确方式。也请留意它自己的告诫，我们同样认同：它所包含的算法表格在很大程度上只是单轮变体的目录，所以不要指望从那里找到智能体特有的算法。

至于形式化的机器，值得与本章配读的那篇论文根本不是一篇 LLM 论文。Sutton et al. (1999) 是节 16.3所应用的那种时间抽象的源头：选项框架、它所诱导的半马尔可

夫决策过程，以及选项级学习与原语级学习何时一致的条件。几乎所有关于「多轮 LLM 论文里的一步到底指什么」的困惑，在花一小时读完那篇论文之后都会消散。

三篇经验论文按此顺序读，可以打好实践基础。Jin et al. (2025a) 引入了观测掩码，篇幅很短；读它是为了那条惯例，也为了它配方中那种诚实的简单。Xue et al. (2025) 给出了目前关于多轮不稳定性最锐利的因果说明——分布外的工具反馈、低概率续写、跨轮累积、梯度爆炸——同时给出一个与具体算法无关的修补（把空轨迹排除出损失，但保留在优势里），值得当作「如何用偏差换方差」的范式加以内化。Feng et al. (2025b) 是轮次级信用分配在表述层面最干净的贡献，也是值得逐字推敲的那一篇，包括它的局限：锚点状态分组需要可哈希且会重复出现的状态，而长的软件工程轨迹与开放网页轨迹并没有这样的状态。

关于 2026 年对掩码惯例的修订，请把ECHO (2026) 与PaW (2026) 放在一起读。它们独立地发展出同一个洞见——在动作词元上做强化学习，在观测词元上做下一词元预测——并在「这个辅助损失需要多小心地对待」上分歧得很有意思。请把它们的定量结果当作有前景但尚未在大规模上被复现。最后，为了校准「算法文献中真正起承重作用的部分有多少」这件事，Yu et al. (2025d) 是最有用的单篇经验研究：真实的端到端轨迹在 SFT 初始化上胜过拼接出来的合成轨迹，利于探索的技术比优势上的精巧更重要，而一种调用更少的审慎策略胜过频繁调用。它那个「配方好的 4B 模型可以匹敌配方差的 32B 模型」的结论，与节 1.6关于实现质量所得出的是同一条教训，只不过现在是针对智能体重述了一遍。

习题

16.1（**可训练比例。**）用节 16.1.1的假设，把可训练比例推导成轮数 K、每轮动作词元数 n 与每轮观测词元数 m 的函数。对 $n = 150$、$m = 600$，求出可训练比例降到 10% 以下时的 K。然后计算训练 FLOPs 与携带梯度的位置数之比，并解释为什么在把智能体式强化学习的成本与单轮 RLVR 相比较时，该报告的是这个量而不是词元数。

16.2（**环境在哪里相消，又在哪里不相消。**）由(16.7)证明 (16.9)。然后构造一个具体的两轮例子使该等式失效：指定一个输出中含有时间戳的环境，以及一个通过重新渲染已存聊天消息来重建上下文的训练器，并说明所算出比值的分子与分母是以不同上下文为条件的。对一个对数概率相差 δ 的差异词元，量化由此产生的比值误差，并说明节 2.6中的哪一个诊断量能检测到它。

16.3（**粒度等价及其失效**。）在正文未证明的那个方向上证明 命题 16.3。然后取 $\gamma_{\text{tok}} = 0.999$，构造一条轮长分别为 2000 与 20 的两轮轨迹，使得词元级估计量与轮次级估计量在至少一轮的贡献上符号相反。就那些在多轮设定中报告逐词元折扣的论文，评论这意味着什么。

16.4（**归一化因子与长轮次**。）一条轨迹含有一轮 5,000 个动作词元和 39 轮各 20 个。在(6.17)的四种归一化因子下，分别计算该长轮次所承载的轨迹梯度权重比例：逐序列词元均值、批次上的全局词元均值、常数 $L_{\max}$，以及逐轮均值。然后在逐轮折扣 $\gamma_{\text{turn}} = 0.95$ 下，把长轮次从第 1 位改到第 40 位再算一遍，并说明：如果你希望那段长思考与那些短工具调用获得相当的总权重，你会选择归一化因子与折扣的哪一种组合。

16.5（**给一次工具调用定价**。）一次检索花费 \$0.002，而一个错误的最终答案会让部署方在下游人工复核上花费 \$0.50。用(16.17)，确定使强化学习最优停止规则与部署最优停止规则一致的逐轮成本 c。然后假设你改用逐轮折扣 γ_{turn}：求出在 $u(h) = 0.5$ 的上下文处与该 c 匹配的 γ_{turn}，并证明在 $u(h) = 0.9$ 处它不再匹配。用一句话解释为什么这是支持加性成本的论据。

16.6（**审计一个掩码**。）有人交给你一次没有任何文档的智能体式训练。请针对单条已存轨迹设计四条断言，它们合起来能确立损失掩码是正确的：一条抓差一错误，一条抓模板词元污染，一条抓重新分词的漂移，一条抓观测词元漏进了 KL 惩罚。对每一条，说明该断言比较的是什么，以及一旦失败会在训练曲线上表现成什么样——如果在曲线上根本看不出来，就直接说明这一点。

16.7 ⋆（**没有重复状态时的信用分配**。）GiGPO 的步级分组要求一组之内各轨迹之间存在精确重复的环境状态。请为一个状态从不重复的领域——比如长的软件工程轨迹——提出一种放松，其依据是状态之上一个学到的或手工设计的相似度，而不是精确相等。写出由此得到的优势估计量。然后分析其偏差：证明把不相同的状态归为一组会混淆状态价值与动作价值，用 V 的组内方差来推导该偏差，并给出一个关于相似度度量的条件，使该偏差小于它所消除的方差。最后，设计一个实验，把你这个估计量带来的好处与「单纯扩大回合级组规模」带来的好处区分开。

16.8 ⋆（**该买多少世界建模**。）ECHO (2026) 与PaW (2026) 在观测词元上添加了一个权重为 λ 的下一词元预测损失。写出合并后的目标函数并推导其梯度。证明在因果注意力下，该辅助项无法在同一次前向中改变动作 logits，然后仔细解释为什么这并不意味着该辅助项跨优化步之后仍让策略学习不受影响。刻画出两个目标函

数争夺容量的那个区间，提出一个你预期会优于常数 λ 的 λ 日程，并指明什么样的测量会证伪你的提议。

16.9 † **（把掩码失效造出来。）**实现一个带可验证最终答案的五轮计算器环境，并在三种掩码配置下，用组相对策略优化训练一个 1–3B 的指令微调模型：正确掩码；把观测词元纳入损失；以及节 16.4中那个陷阱所述的差一错误。对每一种，报告奖励、平均轮数、模型在推理时编造工具结果的比率（通过把它发出的结果与该次调用工具的真实输出相比较来检测），以及在留出观测词元上的交叉熵。你应当会发现，第二种配置显著提高了编造率，而它的损失曲线看起来一切健康；第三种配置除编造率之外，在每一项指标上都与第一种几乎无从分辨。如果你复现不出第二个发现，说明你的时域太短了——延长到十五轮再试。请保留这套运行框架；第17章会扩展它。

表 16.4: 智能体特有的病理、其机制，以及该盯着看什么。每一行都有已辨明的机制；对应的干预手段在Chapters 17 to 19中讨论。各行大致按它们在一次训练中出现的早晚排序。

病理	机制	诊断
工具调用格式坍缩	格式不合法的调用会被结果奖励惩罚；而不再挨罚的最省事办法，就是干脆不再发出那种语法	解析失败率、随后工具调用率，双双下降；奖励持平
轮数坍缩	早期的调用失败多于成功，于是在策略还没学会使用返回结果之前，工具调用词元就先收到了负优势；而回报要在好几轮之后才出现，除了方差之外没有任何东西为它折现	平均 K 单调衰减；零工具轨迹的比例上升；Deng et al. (2025) 的 LLDS-MA 正是出于这个原因下调最终答案词元的权重
空轮与梯度爆炸	分布外的工具输出产生低概率续写，并跨轮不断累积 (Xue et al., 2025)	既不产生调用也不产生答案的轮次；梯度范数尖峰与轮次序号相关
回声陷阱	策略对自己生成的推理模式过拟合 (Wang et al., 2025i)	奖励进入平台期；组内奖励方差与词元熵双双下降；梯度范数出现尖峰，且往往不可逆
模板坍缩	输出变得与输入无关，却在局部仍保持多样，因此熵看上去很健康 (Wang et al., 2026f)	熵是不够的；请用互信息代理量来衡量跨输入的可区分性
懒惰的似然位移	工具集成使各条轨迹之间的似然更新相互耦合；正确回复的似然在奖励上升的同时下降，随后低置信度又放大梯度 (Deng et al., 2025)	正确轨迹的对数似然趋势向下而奖励趋势向上；分三个阶段，终于坍缩
轨迹中途上下文溢出	任务尚未结束就触及上下文上限；压缩会改变每条采样轨迹中可训练片段的数量与长度	因上下文而截断的比率；组内每条采样轨迹可训练词元数的方差
环境非确定性	组相对优势把环境的运气归因给策略；而如果上下文被重新渲染或重放，(16.9)就是在互不匹配的上下文之间计算的	同一提示、同样动作，奖励却不同；采样器对数概率与重算对数概率不一致
经由工具的奖励作弊	工具能够写入验证器所读取的任何东西——一个文件、一个数据库、一份测试夹具——因此通往奖励的最短路径未必要经过任务本身 (Shen et al., 2026b; Cai et al., 2025)	奖励上升而一个独立的留出验证器没有上升；请人工审计采样轨迹
一个坏轮次毒化整条轨迹	轨迹级优势会为了一个坏轮次而惩罚 99 个好轮次，而那个坏轮次的词元拿到的系数和其他一切完全一样	成功率持平而逐轮错误率在下降；在动作质量相当时组内回报方差很大

算法 16.1 收集一条带掩码簿记的智能体式轨迹。要维持的不变式是：`ids` 中的每一个词元都带有记录在案的来源，因此任何掩码都不必事后推断。

输入： 策略 π_θ，带种子的环境实例 $\mathcal{E}$，轮数上限 h，上下文上限 $L_{\max}$，工具模式集合 $\mathcal{T}$

$h \leftarrow$ 渲染系统提示与 $\mathcal{T}$；把该片段记为 PROMPT
turns $\leftarrow [\,]$； $K \leftarrow 0$； trunc $\leftarrow$ NONE
while $K < h$ **and** $|h| < L_{\max}$ **do**
  把运行框架的角色标头追加到 h；把该片段记为 TEMPLATE
  $a, \ell \leftarrow$ 从 $\pi_\theta(\cdot \mid h)$ 采样：词元 id 以及对数概率 ▷ 绝不重算 ℓ
  把 a 追加到 h；把该片段连同 ℓ 记为 ACTION
  $K \leftarrow K + 1$
  if a 以 <eos> 结尾 **or** 不含工具调用 **then**
    break ▷ 策略选择了停止
  end if
  o, status $\leftarrow \mathcal{E}(a)$ ▷ 可能失败或改变状态
  把工具结果定界符与 o 追加到 h；把这两个片段分别记为 TEMPLATE 与 OBSERVATION
  为第 K 轮记录 status 与 $\mathbb{1}$[该轮为空轮]
end while
if $K = h$ **then**
  trunc $\leftarrow$ TURN_CAP
else if $|h| \geq L_{\max}$ **then**
  trunc $\leftarrow$ CONTEXT_CAP
end if
$r \leftarrow$ 在环境状态上、而不是在对话记录上评估验证器
return 轨迹记录（代码 16.3）

算法 16.2 组相对智能体式策略优化的一次迭代。与第6章的单轮循环相比：第 3、6、7、10 行是仅有的结构性新增，而每一行都对应本章的一节。

输入： 策略 π_θ，任务集 $\mathcal{D}$，组大小 G，轮数预算 h，逐轮成本 c，环境池

1: **for** 迭代 $= 1, 2, \ldots$ **do**
2: $\quad \pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$；把权重同步到推理引擎
3: $\quad$ **采样（节 16.2）**
4: $\quad$ 对批次中的每个任务，取得 G 个相互隔离的环境实例，并并行运行算法 16.1
5: $\quad r^{(g)} \leftarrow r_{\text{ver}}^{(g)} - c \cdot K^{(g)}$ ▷ 加性轮次成本，节 16.6.2
6: $\quad$ **优势（节 16.5）**
7: $\quad$ 在组内 $\hat{A}^{(g)} \leftarrow (r^{(g)} - \mu_r)/\sigma_r$，并在 μ_r, σ_r 中保留全部 G 条轨迹
8: $\quad A_t^{(g)} \leftarrow \textsc{broadcast}(\hat{A}^{(g)}, \text{轮次片段}, m^{(g)})$ ▷ 代码 16.1
9: $\quad$ **损失（节 16.4）**
10: $\quad m^{(g)} \leftarrow$ 由记录在案的来源得到的动作词元掩码 ▷ 对齐到目标，代码 16.2
11: $\quad m^{(g)} \leftarrow m^{(g)} \cdot \mathbb{1}[\text{轨迹不含空轮且未被截断}]$
12: $\quad$ 在 $m_t = 1$ 的位置上 $\rho_t \leftarrow \exp(\log \pi_\theta - \ell_t)$ ▷ ℓ 来自采样器
13: $\quad$ 上升 $\frac{1}{\sum_t m_t} \sum_t m_t \min(\rho_t A_t, \text{clip}\big(\rho_t,\, 1-\varepsilon_{\text{low}},\, 1+\varepsilon_{\text{high}}\big) A_t)$；截断梯度范数
14: **end for**
15: **return** π_θ

第 17 章

工具集成的强化学习

核心思想：工具是一个谕示器；强化学习是模型学会何时去请教它的途径

工具把一项模型无法可靠完成的计算，转换成一段它可以读取的观测；这种替换改变的是模型能做什么，而不只是把同样的东西换个包装。但工具本身是一件固定的制品，没有什么科学趣味。真正有趣的对象是关于何时停止推理、开始行动的策略——这个决策模型必须在没有任何关于正确答案的监督下作出，其依据是对自身不可靠程度的一种隐式评估。本章讲的就是训练这个决策。

17.1 工具为何改变能力边界

对工具使用的通常描述——「模型可以调用一个 API」——听上去像是接口层面的变化。它不是；而要看清为什么，需要精确说明工具究竟提供了什么。

定义 17.1 (工具即谕示器)**.** 一个工具集是一族有限的映射 $\mathcal{T} = \{T_1, \dots, T_m\}$，其中 $T_j : Q_j \to O$，Q_j 是可以用词元串表达的查询空间，O 同样是词元串形式的观测空间。工具可以是随机的、有状态的、缓慢的，也可能失败。策略访问 T_j 的唯一方式，是发出一个查询并读取返回的观测。用复杂性理论的语言说，模型是一台谕示机，而值得关心的问题不是谕示器算了什么，而是这台机器发出了多少次查询、以及它把查询选得有多好。

17.1.1 工具修补的两类失败，而它们是不同的失败

不可靠的计算。 考虑把两个二十位整数相乘。单次前向传播做不到，因为进位链随位数以 $\Theta(n)$ 增长，而一个固定深度的网络计算的是一个固定深度的电路。可行的路线是串行的——把部分积写成词元再相加——而在那条路线上，起约束作用的不是表达力而是可靠性。竖式乘法需要 $\Theta(n^2)$ 次基本的数位运算，在 $n = 20$ 时大约 500 次；如果每

一次独立地以概率 $1-\epsilon$ 正确，那么整个推演正确的概率是 $(1-\epsilon)^{500}$，在 $\epsilon = 10^{-3}$ 时为 0.61，在 $\epsilon = 10^{-2}$ 时为 0.0066。在后一种情形下，靠采样达到 99% 的可靠性需要 $k \approx \ln(0.01)/\ln(1-0.0066) \approx 700$ 次尝试，而多数投票也救不回来，因为算术上的失手在不同样本之间并不独立：每一次难住模型的都是同样那些数位边界。一个计算器做同样的计算，其 ϵ 与零无从分辨。工具并没有把一个难题变简单；它是把一个不可靠的过程变可靠了。

缺失的信息。 再考虑一个其答案在预训练截止日之后才进入世界的问题。模型发出正确字符串的概率，就是它对这种形状的可信答案字符串所持的先验：很小，而且——更糟的是——与它发出一个流畅但错误的字符串的概率无从区分。采样在任何 k 下都没有帮助，强化学习也没有：节 1.8表明，带 KL 正则的优化无法创造出参考策略所缺失的支撑，因此在文本上做多少强化学习都无法把一条不在权重里的事实装进去。而一次检索调用以接近一的概率把它返回。

这两种情形分处一道边界的两侧，本章正是围绕这条边界组织的。记 $p_\infty(x) = \lim_{k\to\infty} \text{pass@}k$ 为基座策略在无工具时的极限。C 类任务（能力受限）具有高的 p_∞ 与低的 pass@1：正确轨迹就在分布之内，强化学习的任务是让它更频繁地被找到，而这正是第11章与第14章 反复争论的那个区间。I 类任务（信息受限）在任何负担得起的 k 下都有 $p_\infty \approx 0$，任何纯文本方法都够不到它们。

核心思想: 工具把任务从「需要能力」变成「需要知道该问」

对一个 I 类任务，模型的问题不是求解而是辨识：它必须察觉自己不知道，并构造出一个能告诉它答案的查询。这个察觉是在少数几个词元上的一次策略决策，而它恰恰是结果奖励强化学习能够训练的东西，因为「去问」的轨迹与「去猜」的轨迹会得到不同的奖励。因此，工具集成的强化学习触及了一类纯文本强化学习碰不到的任务，其机制是用一个便宜的决策替换掉一项昂贵的能力。

17.1.2 测量到的后果

节 14.5报告了这项测量。PASS@(k,T) Analysis (2026) 在一个同时改变采样预算 k 与允许交互深度 T 的二维 PASS@(k,T) 网格上评测工具使用智能体，发现工具使用强化学习确实扩大了能力边界：训练后智能体的曲线位于基座模型之上，而且二者的差距在大 k 处 变宽而非收敛，与节 14.2中关于静态推理的发现相反。他们的对照组比其头条结果更重要——在同样的任务上，用配平数据做监督微调反而使边界退化，这就把自

表 17.1: 按工具所提供的东西、而非按其 API 对工具分类。最后一列是决定奖励设计的那一列：它点出了如果奖励为动作而非结果付费，策略将会去利用的那样东西。

提供	例子	验证器	典型滥用
可靠的计算	计算器、Python 解释器、符号代数、SAT/SMT 求解器	精确	写一段把猜出来的答案打印出来的代码
缺失的信息	语料库检索、实时网页搜索、数据库查询	精确（短答案）	引用一篇并不包含该论断的文档
持久的状态改变	shell、文件编辑、仓库提交、浏览器操作	测试套件或状态相等	对测试做特例处理
外置的判断	在轨迹中途调用的基于模型的评论家或评分细则评分器	无	诱导出一个有利的判断

主交互本身、而非数据，分离为因果因素。其机制正是定理 14.3所提供的那个：在单轮生成中，策略唯一的新意来源是它自己的先验，因此支撑集保持的论证起作用；而一条以返回的观测为条件的轨迹，根本就不是从基座模型的文本分布中抽出来的样本。

表 17.1按验证器质量对工具排序，而正是它——不是算法的精巧程度——决定了在古德哈特定律登场之前，一次训练能被推到多远（节 5.8）。前两行囊括了几乎所有可复现的结果，也占据了 节 17.3 and 17.4；最后一行是个陷阱，留待第15章讨论。

17.2 交错模式及其变体

节 16.2搭建了本章所用的形式化，节 16.3给出了轮次级视角，而节 16.4确立了观测词元必须排除在策略梯度项之外。我们假定这三者成立，这里只固定记号。一条采样轨迹是

$$\tau = (h^{(0)}, a^{(1)}, o^{(1)}, a^{(2)}, o^{(2)}, \ldots, a^{(K)}), \qquad h^{(k)} = h^{(k-1)} \oplus a^{(k)} \oplus o^{(k)}, \tag{17.1}$$

其中 $h^{(0)}$ 是提示，每个 $a^{(k)}$ 是一段连续的、由策略生成的词元片段，其结尾要么是一个工具调用定界符，要么是一个终局答案；而每个 $o^{(k)} \sim \mathcal{E}(\cdot \mid h^{(k-1)}, a^{(k)})$ 是渲染后的工具结果。二元动作掩码 $m_t \in \{0, 1\}$ 在属于某个 $a^{(k)}$ 的词元上取 1，在提示、观测与填充词元上取 0。

17.2.1 四种模式

思考、行动、观测。 即 ReAct 模式 (Yao et al., 2023b)，其中 K 只受一个上限约束，每一轮包含自由形式的推理，其后至多跟一次调用。检索智能体用的就是它，因为第二个查询确实依赖于第一个查询返回了什么。

先思考，再调用一次。 在单条长思维链之内给定 C 次调用的预算，通常 $C = 1$，正如Li et al. (2025g) 为数学任务所做的训练。这是一种刻意的简化：只有一次调用时，信用分配问题几乎就是单轮问题，上下文不会不可预测地增长，而第18章所编目的多轮不稳定也不会出现。它是第一次训练的正确起始配置，却是很差的最终配置。

代码即动作。 与其给出一份带类型的工具清单，不如让动作就是沙箱中的任意一段程序。其论据是实在的：一个 Python 代码块就涵盖了计算器、单位换算器和一个 HTTP 客户端，而且它可组合——对一百个候选值做循环是一个动作，而不是一百个。有三项代价通常被低估。动作空间大得多，格式失败率也相应更高，这正是Feng et al. (2025a) 与Chen et al. (2025g) 都要在带代码的轨迹上做冷启动的原因。程序错误以模式违规所不具备的方式无声无息：一段算错了量的程序会返回一个看起来合理的观测，只有结果奖励才会察觉。而且「代码即动作」摧毁了 节 17.6所需要的那套记账，因为一个动作会发出未知次数的调用。

并行调用。 在一个动作里发出 j 个查询，并收到 j 个观测。当这些查询彼此独立时，这会缩短轨迹且几乎免费；而当它们不独立时，这严格更弱：一批并行查询是一种非自适应的查询策略，而对于第二个实体只在第一篇文档里才被点名的两跳问题，任何规模合理的非自适应策略都不会成功。所报告的增益通常是延迟上的胜利，而不是能力上的胜利。

17.2.2 一段带注解的轨迹

代码 17.1展示了一个训练好的策略其采样轨迹长什么样。有两处特征值得注意。推理片段陈述了为什么要调用工具，其措辞点名了模型自身的不可靠——这没有被提示要求，而且是若干独立团队在只有结果奖励的条件下报告出来的 (Feng et al., 2025a; Xue et al., 2025; Qian et al., 2025)。而观测之后的那一段则拿结果与记住的量级做了合理性核对，正是这个行为使得工具使用对一次错误调用具有鲁棒性，而不仅仅是更快。

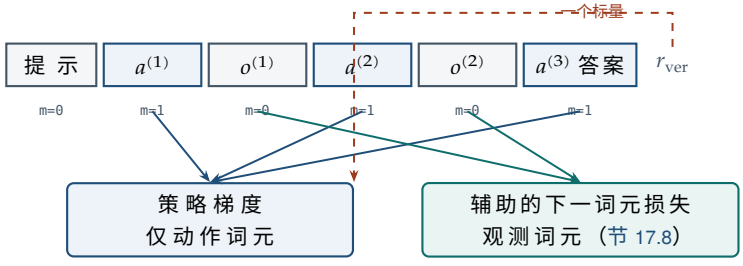


图 17.1: 一条工具集成采样轨迹的词元流，每个片段下方标出掩码 m_t。蓝色片段由 π_θ 选出，是唯一可以进入策略梯度估计量的片段；灰色片段由环境写入。青色那条路径是 节 17.8所讨论的 2026 年修订：观测词元之所以被排除在策略梯度之外，是因为它们不是动作，而这并不意味着它们不携带可用的信号。虚线箭头就是全部奖励：一个标量，只交付一次。

代码 17.1: 一段示意性的工具集成轨迹，动作掩码标在右侧栏中。`TRAIN` 片段携带 $m_t = 1$ 并接收该轨迹的优势；`MASK` 片段携带 $m_t = 0$。观测词元在其后的每一个词元处仍会被策略读取，而 节 17.8主张它们还应当被预测。

```
<think> 霍曼转移，从 LEO 到 GEO。r1 = 6771, r2 = 42164 km, mu = 398600。
两次点火；每次都是两个平方根之差。这类题我手算出错的时候多过不出错，
所以还是算一下，别去估。 </think>                                  | TRAIN
<code>
import math
mu, r1, r2 = 398600.0, 6771.0, 42164.0
dv1 = math.sqrt(mu/r1)*(math.sqrt(2*r2/(r1+r2)) - 1)
dv2 = math.sqrt(mu/r2)*(1 - math.sqrt(2*r1/(r1+r2)))
print(round(dv1,4), round(dv2,4), round(dv1+dv2,4))
</code>                                                              | TRAIN
<observation> 2.3995 1.4571 3.8566 </observation>                | MASK (m=0)
<think> 合计 3.857 km/s。拿我确实记得的数字对照一下：LEO 到 GEO 的转移
通常被引作接近 3.9 km/s，而且第一次点火应当是两次中较大的那次。两条都
成立，所以这次调用用对了。 </think>                               | TRAIN
<answer> \boxed{3.86\ \text{km/s}} </answer>                       | TRAIN
```

17.2.3 采样循环，以及那些要紧的细节

算法 17.1是采样过程。除了五个地方之外它平淡无奇，而这五处每一处都对应一个已被发表的失效。

错误必须作为观测返回，绝不能抛进循环里：一个从未见过栈回溯的策略学不会从中恢复，而一条因超时而死掉的采样轨迹会把训练分布偏向简单的提示。每个动作片段被采样出来的精确词元标识必须被存下来，而不是存渲染后的聊天记录，因为通

过提示模板重新渲染再重新分词，一般并不能还原采样时的词元，于是训练器会为策略从未发出过的词元计算比值——Cheng et al. (2025) 把这称作重新分词漂移，并用步级结构化记录修复它。一个既没产生格式合法的调用、也没产生答案的轮次，即Xue et al. (2025) 术语中的空轮，必须被检出，因为含有空轮的轨迹正是 第18章所分析的那些梯度爆炸的来源。而每一次调用的延迟、成本与缓存状态都必须被记录，因为节 17.6要用到它们。

17.3 检索与搜索智能体

在智能体式强化学习中，搜索是最干净的实验设定，其两点理由与「搜索本身有趣」无关：验证器是精确的，而工具便宜到组大小取 8 到 16 都负担得起。本领域关于工具强化学习的几乎全部信念，最初都是在这里确立的。

17.3.1 只有结果奖励教会了什么

Jin et al. (2025a) 是参照点。策略把 `<search>` 查询与 `<information>` 块交错起来，观测词元被掩出损失——引入这条惯例的正是这篇论文——而奖励只是最终答案的精确匹配，除此之外别无其他；关于该发出哪些查询、发多少次、按什么顺序，都没有任何监督。他们在七个问答数据集上报告，相对检索增强生成基线，7B 上提升 41%，3B 上提升 20%。Song et al. (2025a) 分两阶段到达同一处，第一阶段奖励发起调用，第二阶段只奖励答案；而Chen et al. (2025b) 把它剥得更干净：GRPO，把搜索当作推理链中普通的一环，完全没有步级监督。

有三种行为在这些独立团队中反复出现，出现得足够频繁，足以算作已被复现。查询重述：在一次收获不佳的观测之后，策略会改写查询而不是重复它，而且改写是有章法的——加上一个消歧的实体，或去掉一个过度限定的修饰语。多跳分解：在需要两三条相互关联事实的问题上，策略会按依赖顺序发出查询，而奖励中没有任何部分描述过这一点。自适应调用：当它已经知道答案时就不再检索，Song et al. (2025b) 把这一点做成了显式的，而 Wei et al. (2025c) 则以「究竟要不要用工具」为学习目标。

17.3.2 以及哪里会出问题

常见陷阱: 检索器的失败变成了策略的梯度

结果奖励无法区分「策略问了一个糟糕的问题」与「策略问了一个好问题，但检索器没返回任何有用的东西」。两者都产生错误答案，并在每一个动作词元上产生负优势，其中也包括那些本已选得很好的查询词元。在检索器较弱时，这一项占梯度中很大一部分，而它教会策略的是回避一类更好的索引本可以回答的问题——这是一个被烙进估计量里的归因错误。诊断很便宜：固定策略、换掉检索器、重算奖励分布。如果它变动很大，那么你的训练有一部分是在绕着你的索引打转。请先改进索引，再改进算法；并在报告答案准确率的同时报告检索召回率。

第二个问题是检索的奖励作弊，其常见形式是一个验证器接受、但检索到的文档并不支持的答案。在任何有冗余的语料库上，先猜得好、再检索来确认，都是一种可行的策略，而对一个短字符串做精确匹配是无法把它与基于证据的作答区分开的。补救办法是加一项接地检查——要求答案片段出现在某篇检索到的文档中——这把一个验证器变成了一个合取式，也是节 17.5中少数几个物有所值的反作弊项之一。

17.3.3 模拟检索还是实时检索，以及这个选择为何承重

一个冻结的语料库索引使环境确定且可重放，于是不同次训练可比、消融也有意义。它同时也使环境 平稳，从而让策略可以对它过拟合：针对一个索引优化出来的查询分布，也就是针对那个索引的怪癖优化出来的。Sun et al. (2025) 把模拟推到极致，用一个微调过的语言模型取代检索器，按一套让文档质量随训练逐步退化的课程，发出既有用又刻意带噪的文档。他们报告一个 7B 模拟器大致相当于一个真实引擎，而一个 14B 模拟器超过了它——如果这能复现，这是一个惊人的论断，因为它意味着一个可控的带噪模拟器胜过真家伙，大概是因为课程可以被排定。Fan et al. (2025) 则取了模型自己充当检索器的那个退化情形。

实时检索买来的是真实性，付出的代价是噪声、延迟、非确定性与速率限制。Zheng et al. (2025d) 最先表明端到端地这么做是可行的，而他们暴露出来的那些失效模式——加载不出来的页面、同一组内不同采样轨迹之间会变化的结果、拖住一个同步步骤的长尾延迟——正是模拟器所掩盖的那些。Gao et al. (2025b) 把训练做成完全异步的，而这是实时工具不至于摧毁吞吐的唯一配置（节 21.6）。

工程实践: 何时用哪一种

在算法开发阶段、以及任何你打算发表的比较中，请使用带固定快照的冻结索引；如果环境在两次训练之间变了，你就无法把 2 个百分点的差异归因于你的损失函数。只在最后阶段才转向实时工具，并预期需要重新调参。如果你必须全程针对实时工具训练，那就大力做缓存（代码 17.3）、把采样做成异步的，并记录每一条观测以便这次训练可以被重放。还要报告你用的是哪一种：跨越这个选择的数字是不可比的。

17.4 代码与计算类工具

面向数学的工具集成推理早于强化学习时代。程序辅助语言模型 (Gao et al., 2023b) 与程序化思维链提示 (Chen et al., 2023a) 确立了把算术卸载给解释器可以提高准确率，而 ToRA(Gou et al., 2024) 在交错的推理—程序轨迹上做微调，把它变成了一个训练目标。ToRA 做不到的是决定何时卸载；那是由演示数据定死的。

历史注记: 从模仿工具调用到自己选择工具调用

2023 年的这一脉络——Toolformer(Schick et al., 2023)、ToRA(Gou et al., 2024)——是通过模仿来学习工具使用的，其轨迹里，一个更强的模型或某种启发式已经替它决定了调用该放在哪里。由此得到的策略会在它的老师调用工具的地方调用工具，这是一个合理的先验，却是一个糟糕的策略：调用计算器的正确位置取决于这个模型不擅长什么，而它的老师不擅长的是别的东西。2025 年的转变，是用一个学出来的决策取代那个继承来的决策。Li et al. (2025g) 从基座出发训练，不作任何监督初始化，并报告代码使用比例在一百个 RL 步内从大约 40% 升到 80%——策略自己发现了工具值得用的频率。

17.4.1 2025 年的结果

Feng et al. (2025a) 在合成的、带代码的长思维链上做冷启动，随后配合实时执行做结果强化学习。他们的头条比较关乎计算量而不只是准确率：据其报告，一个 32B 模型在 400 个 RL 步内在 AIME 上达到 67%，而纯文本强化学习是在 1080 步内达到 40%。他们还报告了涌现出的代码自我纠错，即策略读取一段回溯并修好自己的程序。Li et al. (2025g) 从基座出发、带调用预算地训练；Singh et al. (2025a) 报告在自主选择工具的情形下最多有 22 个百分点的绝对提升；而Chen et al. (2025g) 通过多阶段强化学习把文本与代码规划交错起来。Shang et al. (2025) 是计算效率方面的参照：14B 上经过 510

个 RL 步在 AIME24 上达到 80.6%，在 64 张加速卡上约一周，其间用了一个「答对即重采样」的 GRPO 变体来应对代码工具注入的噪声。而Xue et al. (2025) 只加了空轮过滤这一件事，就把一个基座模型在 AIME24 上从 22.1 带到 50.5——这里报告的最大单项增益来自一次稳定性修补，而不是一项奖励设计。

17.4.2 卸载作为学到的元认知

这一脉络中有意思的论断是机制性的而非数值性的。策略并不会学着一律使用解释器；它学会的是卸载 那些自己不可靠的子计算。这是一个关于自我认识的论断，而与多数此类论断不同，它是可测量的。

推导: 何时卸载是最优的，以及策略为何能找到它

考虑一段推演中的某个子计算 u。设 q_u 为策略在上下文中正确算出 u 的概率，η_u 为工具返回正确值的概率——对算术而言基本上是 1，而当策略必须把 u 翻译成一段正确的程序时会更低。设 c 为一次调用的价格（节 17.6），并设推演的其余部分以与 u 独立的概率 s 成功。那么在上下文中求值的期望奖励为 $s\,q_u$，而卸载的期望奖励为 $s\,\eta_u - c$，因此策略应当卸载，当且仅当

$$\eta_u - q_u \;>\; \frac{c}{s}. \tag{17.2}$$

有两个推论。这条规则是因模型而异的，因为 q_u 是这个策略的性质，所以任何固定的演示集都编码不了它——这正是上面那则历史注记的形式化内容。而它可以仅从结果奖励中学到，因为 节 11.2的组相对比较里已经含有所需的对照：在同一问题的 G 条采样轨迹构成的一组中，有些卸载了 u、有些没有，而结果奖励之差就估计了对轨迹其余部分求边缘之后的 $s(\eta_u - q_u)$。照这个读法，组相对强化学习就是在用蒙特卡洛跑(17.2)。

由此得到的预测正是审计一个工具训练策略的正确方式：把留出轨迹中的各个子计算划分为已卸载与未卸载两组，并测量当策略被强制在上下文中完成它们时，每一组上的错误率。如果卸载是元认知的，那么在已卸载那一组上被强制回到上下文时的错误率应当显著更高；如果它只是一种文体习惯——「这里看着就是该放代码的地方」——两个错误率就会差不多。这项测量被报告的次数远低于它应有的次数，而习题会要求你去做。

常见陷阱: 惩罚不可执行的代码会让策略干脆不再写代码

Li et al. (2025g) 在 $+1/-1$ 的正确性奖励之外，对执行失败的代码加了 -0.5 的惩罚，并平直地报告了这个否定结果：该惩罚没有帮助，纯结果奖励与之持平甚至更好。其机制就是把 (17.2)反过来读。这项惩罚以一个与「该次调用是否必要」无关的量降低了每一次调用的价值，因此它一律抬高了卸载阈值，使策略在生成代码这件事上整体变得保守——包括在代码本是唯一可靠途径的那些地方。这是本章中好几处「看上去像有益塑形、实为伪装过的阈值平移」的第一处。

17.5 工具使用的奖励设计

这一节决定了一次工具强化学习训练能否产出一个你愿意部署的策略。节 11.6为单轮可验证任务发展了复合奖励；工具设定又添了一样那个设定没有的诱惑：一堆便于测量的中间事件。它们几乎每一个都是陷阱，而这些陷阱共有一种值得先行点明的结构：**任何为动作而非为结果付费的奖励项都会平移某个决策阈值，而策略会移动到那个新阈值上，不论这样做是否有帮助。**

17.5.1 为什么奖励工具使用是个陷阱

假设奖励里包含 $+\beta\,\mathbb{1}[n \geq 1]$，其理由听上去很合理：训练早期策略根本不调用工具。一旦它在几乎每个提示上都至少调用一次，每一组的每一个成员都拿到同样的 β，于是这一项在 式 (11.3)的组相对优势中相消，无害。危害出在过渡阶段。当组内有些成员调用、有些不调用时，优势编码的是「有没有调用」而不是「答案对不对」：两条仅差一次不必要调用的轨迹奖励不同，而梯度会推向那条调用了的，不论观测是否被用上。策略会先学会调用，然后才学会使用返回的东西，因为调用信号是稠密的而结果信号是稀疏的。正确的模式是 Song et al. (2025a) 刻意采用的那个：第一阶段给调用奖励，第二阶段撤掉。它是一个带排期的自举，而不是一个分量；如果你说不出打算在第几步撤掉它，就不要加它。

17.5.2 为什么奖励工具调用成功是个更微妙的陷阱

奖励成功（未报错）的调用看上去严格更好，因为它只为真的奏效了的调用付费。它其实更糟，原因是一种作用在查询分布上、而非作用在调用次数上的选择效应。

命题 17.2 (成功奖励对信息量做逆向选择). 固定一个上下文 h，令策略从空间 Q 中选择一个查询 a。记 $\sigma(a) = \mathbb{P}[\text{该次调用成功}]$，并令一次成功调用所提供的信息为 $I(a) = \mathbb{H}[Z \mid h] - \mathbb{E}_o[\mathbb{H}[Z \mid h, o]]$，其中 Z 是答案变量。奖励项 $+\beta\sigma$ 会把最优查询分布推向高 σ 的查询。只要 σ 与 I 在 Q 上负相关，被推移后的策略每次调用的信息量就更低。

这种负相关是经验常态，而不是人为造作。具体的查询比含糊的查询更常失败——检索一个精确的实体加日期，只要两者有一处略有出入就什么都返回不了，而一个宽泛的查询总能返回点什么；一段真去算你所需之量的程序会抛异常，而 `print(2+2)` 不会。要让一次调用可靠地成功，办法就是去问一件你其实已经知道的事，所以 $+\beta\sigma$ 补贴的是那些答案无关紧要的问题，而它在日志里的特征是：调用成功率一路上升，而准确率持平、平均观测新颖度下降。上面那个 ToRL 的否定结果就是同一个命题反过来：惩罚失败与奖励成功是同一种干预。

17.5.3 原则，以及唯一的例外

核心思想: 奖励结果，把格式设为门，给资源定价

只为你真正想要的东西——最终结果——付费。把良构性做成一个乘性的门而不是一个加性的奖金，使它无法与正确性相权衡。为消耗掉的资源减去一项显式的价格，其单位要站得住脚。而且不要奖励某个中间动作，除非你能验证它是承重的；「承重」是有真实定义的：$a^{(k)}$ 在 τ 中是承重的，当且仅当把 $o^{(k)}$ 换成一个空观测后从第 k 轮重新采样会改变结果。这个反事实很贵但可计算，而它是唯一诚实的凭据，能证明一个中间奖励确实在度量什么，而不是在平移一个阈值。

值得认真对待的例外，是步骤本身可验证、而不只是可观测的那种步级监督。Wang et al. (2025j) 提供了子问题层面的信息增益奖励并配以冗余惩罚，报告在检索任务上步级奖励胜过轨迹级奖励；Jiang et al. (2026b) 则以结构化的中层监督占据了一个中间地带。与之相对，Li et al. (2025g) 的可执行性惩罚有害，而Yu et al. (2025d) 发现过程奖励并不可靠地有帮助。截至 2026 年年中，诚实的总结是：当步骤具有独立的正确性概念时——「这次查询有没有检索到包含子问题二答案的那篇文档」——步级信号有帮助；而当它只是努力程度的代理量时，它有害。

代码 17.2: 一个遵循该原则的工具奖励。注意其中没有什么：没有任何一项奖励调用、奖励成功的调用，或奖励格式合法的调用。价格是相对一个组统计量而非相对一个常数来参照的，这样难题就不会因为需要更多调用而挨罚（节 17.6）。

```python
def group_reference(rolls, gold, cap):
    """本组答对的成员中最省的调用次数。"""
    ok = [r.n_calls for r in rolls if verify(r.answer, gold)]
    return min(ok) if ok else cap

def tool_reward(roll, gold, n_ref, price=0.05):
    # 门 1：模式合法性。乘性的，因此买不到分。
    if not roll.well_formed:
        return 0.0
    # 空轮在上游就被排除出损失，不在这里定价。
    correct = float(verify(roll.answer, gold))
    # 门 2：答案必须落在工具确实返回过的东西上。
    if correct and not grounded(roll.answer, roll.observations):
        correct = 0.0
    # 价格：超出本组某位成员所需次数的部分，按调用数线性计价。
    excess = max(0, roll.n_calls - n_ref)
    return correct - price * excess

def batch_rewards(groups, cap, price):
    out = []
    for rolls, gold in groups:
        n_ref = group_reference(rolls, gold, cap)
        out.append([tool_reward(r, gold, n_ref, price) for r in rolls])
    return out
```

17.6 要调用多少次工具?

一个用纯结果奖励、且不给调用定价训练出来的策略会过度调用，而这并不是因为它糊涂。在稀疏的终局奖励下，多一次调用的成本为零而期望价值弱正——它推迟了承诺，而推迟是免费的——因此使奖励最大化的策略会一直调用，直到轮数上限或上下文预算把它拦住。那是一个规格错误，不是一种训练病理。

17.6.1 生产率与定价后的目标函数

定义 17.3 (工具生产率)**.** 对一个在某提示集上评测的策略，工具生产率为

$$P(\pi_\theta) \;=\; \frac{\mathbb{E}_{x,\tau}\big[r_{\text{out}}(\tau)\big]}{\mathbb{E}_{x,\tau}\big[n(\tau)\big]}, \tag{17.3}$$

即每次工具调用所期望产出的正确答案数。Wang et al. (2025a) 把它作为一个纯粹由准确率驱动的目标函数所忽视的度量而引入。

生产率是一个诊断量，不是一个目标函数：单独最大化它，会偏好「调用一次、答对一个问题」的策略胜过「调用两次、答对两个问题」的策略。目标函数是定价后的那一个，

$$J_c(\theta) = \mathop{\mathbb{E}}_{x} \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[r_{\text{out}}(\tau) - c\, n(\tau) \right], \tag{17.4}$$

其中 c 是一次调用以答案正确性为单位的价格。这样写出来就明确了一件事：在没有价格的情况下，不存在正确的工具调用次数。一个每次检索付一分钱的部署对应小的 c，而一个在按用户预算约束下以交互延迟运行的部署对应大的 c。这一文献中许多混乱，都来自把在不同隐含价格下训练出来的系统拿来相互比较。

17.6.2 最优调用规则

推导: 如果一段观测的期望价值超过它的价格，就买下它

把每一轮的决策当作最优停止来处理。设 $V_c(h)$ 为从上下文 h 起 (17.4)的最优值，并设 $q(h) = \max_a \mathbb{P}[a \text{ 正确} \mid h]$ 为立刻作答的成功概率。则

$$V_c(h) = \max\Big\{ \underbrace{q(h)}_{\text{停止}},\ \underbrace{\max_a \Big[-c + \mathop{\mathbb{E}}_{o\sim\mathcal{E}(\cdot\mid h,a)} \big[V_c(h \oplus a \oplus o) \big] \Big]}_{\text{再调用一次}} \Big\}. \tag{17.5}$$

定义一个候选调用的信息期望价值，

$$\mathrm{EVOI}(h,a) \;=\; \mathop{\mathbb{E}}_{o\sim\mathcal{E}(\cdot\mid h,a)} \big[V_c(h \oplus a \oplus o) \big] \;-\; q(h). \tag{17.6}$$

比较(17.5)的两个分支，一行就给出了规则：

$$\textbf{调用} a \textbf{ 当且仅当} \mathrm{EVOI}(h,a) > c, \qquad \text{并且调用使EVOI 最大的那个} a. \tag{17.7}$$

有三个推论。最优策略关于 c 是单调的：抬高价格只会减少调用，所以可达集合 $\{(\bar r, \bar n)\}$ 有一条凹的上边界 $\bar r^\star(\bar n)$，它由扫描 c 描出，因为(17.4)是该集合的一条斜率为 $-c$ 的支撑超平面。可报告的对象是这条前沿，而不是某一个点：说方法 A 比方法 B 用的调用更少，除非两者都被放在各自的前沿上，否则这话没有信息量。而且这条规则是自指的，因为 EVOI 依赖于 V_c，而后者正是我们要训练的东西——策略是在学习它自己的停止规则，这正是这个决策可训练却无法解析求解的原因。

把(17.7)当作一个购买决策来读，问题就变得可读了。策略是一个面对价值不确定的商品的买家，而它必须在看到观测之前就估计出，那段观测会在多大程度上改变它接下来的做法。这样的估计需要知道自己不知道什么，所以这个决策是一个披着动作选择外衣的校准问题。这也是 (17.7)为何无法被监督的原因：一份演示展示的是一个不同的买家、带着不同的 $q(h)$，选择买了什么。

17.6.3 那条 U 形曲线，以及两端失败为何机制不同

Wang et al. (2025a) 把定价后的目标函数实例化为 OTC-PPO 与 OTC-GRPO，报告在准确率相当的情况下工具调用减少 68.3%、工具生产率提高 215.4%。减少的幅度本身就是过度调用这一论断的证据：一个朴素训练出来的策略所作的调用中，有三分之二是可以白白省掉的。他们在结构上的选择才是可迁移的部分。节俭因子是在正确性上作乘法的，因此一个错误答案无法靠便宜来挣分；而且它参照的是一个组统计量——大

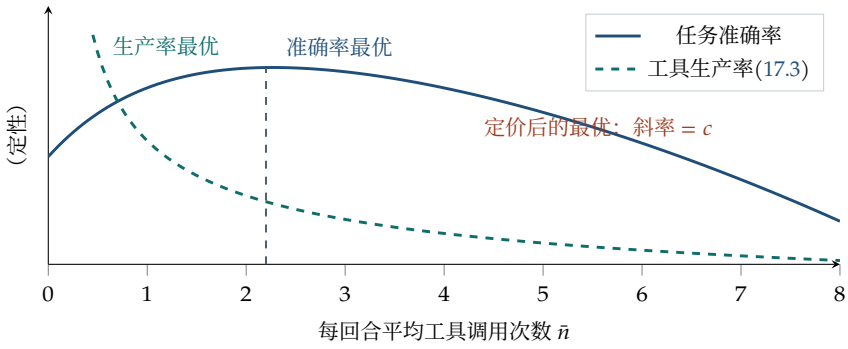


图 17.2: 示意图，非数据。准确率关于平均调用次数是非单调的：调用太少会放弃信息，调用太多则会撑大上下文、累积工具噪声并耗尽词元预算。生产率(17.3)的最大值远在准确率最优点的左侧，这正是它是诊断量而不是目标函数的原因。由 (17.4)选出的工作点，位于准确率曲线斜率等于 c 之处，因此是价格——一个部署事实——挑出了这个点，而扫描它就描出了节 17.6.2的那条前沿。曲线只是示意性的函数形式；这里断言的只有三个标记点的先后次序。

致是组内答对的成员中最小的调用次数——而不是一个绝对常数。第二个选择才是要紧的那个：一个绝对参照会因为难题需要更多调用而惩罚难题，这是反的，而那 G 条采样轨迹构成的组本来就已经估计出了这个问题需要多少次才够。这就是组相对优势背后的那个观察，被应用到了第二个量上。

价格压得太狠，失败就会反转。其机制是一种信噪比上的不对称，值得陈述出来，因为它预示了补救办法。惩罚项 $-c\,n$ 是确定且即时的，而一次调用带来的准确率增益是随机的、且只在终局奖励处才兑现，因此惩罚项的梯度方差低得多，策略会先学会惩罚。你观察到的现象是：调用次数在几十步之内坍缩到零，而准确率随之跟着掉下去。补救办法也就随之而来：把 c 从零开始向上退火，好让策略先学会工具有用，再学会工具很贵。Yu et al. (2025d) 从经验上到达同一处，报告一种调用更少的审慎策略胜过频繁调用与啰嗦的自我推理两者。

17.7 深度研究智能体

深度研究是目前仍在积极攻关的设定中最难的一个，而且它同时难在三个方面。时域很长，数十到一百多轮，因此第13章中的信用分配问题会全力袭来，第18章中的上下文管理问题也开始起约束作用。任务是开放式的：交付物是一份报告，没有黄金字符串。因此奖励本身就是一次判断，这就把第5章的每一种病理都带进了一个轨迹远比奖励模型被验证时更长、也更昂贵的设定里。

已发表的工作中最系统的一批出自通义 DeepResearch 项目，而它值得当作一条全栈流水线、而不是一串方法来读。Wu et al. (2025d) 贡献了作为锚点的基准，Li et al. (2025f) 把深度检索集成进一个推理模型，而Wu et al. (2025c) 给出了此后一切工作都遵循的四阶段流水线：构造浏览数据、采样轨迹、用监督微调冷启动，然后做强化学习。Li et al. (2025b) 是方法论的中心，主张把专有研究系统与开源系统区分开来的能力是系统性的不确定性削减，并通过合成刻意高不确定性的任务来注入这种能力——先做结构化采样，再做信息混淆，使得不经过真正的多步检索就无法抵达答案。Tao et al. (2025) 用一套关于信息寻求的集合论说明，把这种合成形式化；而Chen et al. (2025e) 把合成瞄准模型的最近发展区，从而把节 12.4.1 的课程思想从任务选择挪到了任务生成。作为收尾的那份报告 (Tongyi DeepResearch Team, 2025) 把整套栈组装在一个 30.5B、激活参数 3.3B 的混合专家模型周围。

可迁移的教益是一个阶段，而不是一个算法：一个智能体式的中期训练阶段——在做任何强化学习之前，先在长的智能体式轨迹上以普通的下一词元损失继续预训练——才是让后续强化学习在计算上高效的原因 (Su et al., 2025; Tongyi DeepResearch Team, 2025)，而这正是节 4.7那条论证的特化。「按一个格式合法的模式发出工具调用、读取结果、然后继续」是一种归纳偏置，强化学习无法引出一种本不存在的归纳偏置，而跳过这个阶段的团队会把强化学习预算花在教格式上。

对这里的数字要抱持怀疑，而这并不是因为工作做得草率。在那些标准目标上报告的分数——Humanity's Last Exam(Phan et al., 2025)、BrowseComp(Wei et al., 2025a)、WebWalker 数据集——取决于一个在两次评测之间会变化的实时网络，取决于各篇论文之间彼此不同的评判模型，也取决于那些很少被写明、却会实质改变结果的运行框架细节（页面抓取超时、最大轮数、重试策略）；Gou et al. (2025) 正是出于这个原因引入了「智能体即评判」协议。当Dong et al. (2025a) 报告一个用 1K 条 RL 样本训练的 14B 模型取得 GAIA 47.6%、HLE 11.2%、WebWalker pass@1 43.0% 时，引人注目的是其样本效率，而那些绝对水平所携带的方差比其位数所暗示的更大。第24章会正经处理测量问题；简短的说法是，在深度研究领域，跨论文比较目前并不可靠，而携带信息的是各篇论文内部的消融。

17.8 把环境观测当作稠密监督

由Jin et al. (2025a) 引入、如今已普遍采用的掩码惯例是正确的，而它被广泛误读了。本节要把它捆在一起的两个论断拆开，因为 2026 年文献中最重要的想法就住在两者之间的缝隙里。

17.8.1 掩码是一个估计量，不是一条启发式规则

节 16.2那个智能体式 MDP 的策略梯度是

$$\nabla_\theta J = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\Big[\sum_{k=1}^{K} \nabla_\theta \log \pi_\theta(a^{(k)} \mid h^{(k-1)}) \cdot A(\tau)\Big]. \tag{17.8}$$

其中没有 $\nabla_\theta \log \mathcal{E}(o^{(k)} \mid \cdot)$ 这一项，因为环境的分布不依赖于 θ。因此，把观测词元纳入求和，并不是给(17.8)的估计量增加了方差；它是加进了一个根本不属于被估计量的项。而那一项是有解释的：$A(\tau) \cdot \nabla_\theta \log \pi_\theta(o^{(k)} \mid \cdot)$ 是观测词元上一个下一词元预测损失的梯度，按该轨迹的优势加权。在成功的轨迹上，它训练模型以环境的口吻幻觉出观测；在失败的轨迹上，它训练模型觉得真实的工具输出很意外。两者都不是任何人想要的，这正是掩码之所以必须的原因——但未加掩码的那个损失并不是噪声，它是一个系数选得很糟的世界模型目标。

17.8.2 作为动作被掩掉，作为目标仍可用

核心思想: 一个观测词元的三种角色

采样轨迹中的一个词元可以扮演三种截然不同的角色，而掩码惯例把其中两种混作了一谈。作为 **动作**，它是 π_θ 选出来的东西，而只有动作词元才可以进入(17.8)。作为**上下文**，它为其后的一切提供条件，而策略在该位置上的 logits 反正会在同一次前向中被算出来。作为**预测目标**，它是关于「世界如何回应」的免费、稠密、可核验的监督——在每一条采样轨迹的每一轮上都有，包括那些一分奖励也没挣到的轨迹。把一个词元从角色一中掩掉，对角色三什么都没说。

2026 年有两篇独立的论文利用了角色三。ECHO(ECHO, 2026) 主张，终端智能体是语言模型所能拥有的、最接近具身设定的东西：它发出命令，而 stdout、退出码与栈回溯记录下了后果。GRPO 把那份记录扔掉了，只用一个稀疏的结果奖励去更新动作词元，因此一条失败的采样轨迹尽管含有关于动力学的详尽证据，却几乎不提供任何梯度。他们的目标函数在观测词元的一个子集 $O' \subseteq O$ 上加了一项长度归一化的交

叉熵：

$$\mathcal{L}_{\text{ECHO}} = \underbrace{\mathcal{L}_{\text{GRPO}}|_{m_t=1}}_{\text{行动}} + \lambda \underbrace{\frac{1}{|O'|}\sum_{t\in O'}(-\log \pi_\theta(y_t \mid y_{<t}))}_{\text{建模世界}}. \tag{17.9}$$

系统层面的那个观察使它变得可行：策略本来就以那些词元为条件，也已经算出了它们的 logits，因此这份世界建模信号不需要额外采样、不需要教师、也不需要另一个模型——只需要在已经存在的 logits 上换一个掩码。他们的实现技巧可以推广：监督微调就是优势为一个正常数的强化学习，因此(17.9)可以实现为单个损失，办法是把逐词元优势在动作词元上设为组相对优势、在观测词元上设为常数 λ。对这样便宜的一项改动而言，所报告的效果很大：Terminal-Bench 2.0 的 pass@1 在两个规模上大致翻倍（8B 上 2.70% → 5.17%，14B 上 5.17% → 10.79%），强化学习快约 2.3×，而且从一个基座模型出发，该方法在 不用专家演示的情况下就追平了「专家微调再 GRPO」。最引人注目的是，在某些设定下，仅靠环境预测损失——完全不用验证器——就能改善留出的分布外任务。

PaW(PaW, 2026) 独立地发展出同一个洞见，并在「为什么这样做是安全的」上更为审慎：同策略的采样轨迹本来就把每个动作与其后的观测配成了对，而因果注意力保证了追加的观测词元不可能影响动作 logits，于是策略学习项保持不变，推理时也只是一次普通的前向。在他们的实验中，有三个部件让这项辅助监督保持稳定：按动作熵挑选世界模型数据、一个抗噪的损失，以及对两项作奖励自适应的平衡。

17.8.3 另外两种用途，它们不是损失

把意外程度作为逐次调用的信息量估计。 下面这个量

$$s^{(k)} \;=\; -\log \pi_\theta(o^{(k)} \mid h^{(k-1)} \oplus a^{(k)}) \tag{17.10}$$

是策略自己对工具返回结果的意外程度：一次结果本可以被它预测出来的调用，什么也没告诉它。这是(17.6)中 EVOI 的一个免费的事后代理量，它提示我们可以按已实现的信息量、而不是按次数来给调用定价。它也有一个明显的作弊风险，习题会要求你把它想清楚：一个因意外程度而受奖励的策略，可以靠把预测做差来抬高它。Dong et al. (2025b) 出于另一个目的测量了同一个量，他们观察到熵在工具反馈被注入之后立刻出现尖峰，并在出现尖峰之处对采样轨迹作分叉。

按轮次信息量设门。 如果一轮的观测本来就是可预测的，而答案也没有改变，那么这一轮就是装饰，它的动作词元会稀释掉那些真正起作用的词元的优势。给这类轮次降权是一种温和且可核验的信用分配形式；空轮过滤 (Xue et al., 2025) 则是「这一轮什么都没产生」的退化情形。

诚实的证据层级是：两篇独立的 2026 年论文，都报告了可观的增益，都尚未被第三个团队复现。未决问题也正是你会预料到的那些：什么样的 λ 排期才对、世界模型项最终会不会与策略学习争夺容量，以及在观测不像终端输出那样结构化的场合，这些增益还在不在。鉴于这项干预是如此便宜——在你本来就要算的 logits 上多加一个掩码——在你自己的设置上试一试有很高的期望价值。

17.9 一套工具强化学习技术栈必须提供什么

工具集成的强化学习因基础设施原因而失败的次数，远多于因算法原因而失败；而这些失败都很乏味：一次杀死采样轨迹的超时、一个返回了过期文档的缓存、一个在训练时与采样时渲染结果不同的聊天模板。第19章把环境工程当作一个专题来讲，第21章讲扩展；本节只列出最低限度，因为其中任何一项被跳过都会产生一个具体的缺陷。

1. **一个具有稳定模式与结构化错误的工具服务。**每一种失败模式——参数错误、超时、速率限制、空结果——都必须返回一个格式合法的观测，因为一个传播进采样循环的异常会悄无声息地删掉一条轨迹。
2. **逐采样轨迹的隔离。**组内成员之间不得共享可变状态，否则一个 GRPO 组就变成了一个相关样本。对代码执行而言，这意味着每条采样轨迹一个容器或进程沙箱，而这正是 第19章中那些编排成本的来源。
3. **写明的超时与截断。**观测长度上限与超时是环境定义的一部分；改动它们就改动了 MDP，也就使得与既往训练的比较失效。
4. **一个内容寻址的缓存**，其键为（工具，规范化后的参数，语料库快照）。组内成员会问出彼此重叠的查询，所以命中率很高，而缓存也正是使重放成为可能的东西。
5. **确定性与重放。**把语料库快照钉死，并把每一个（调用，观测）对连同其哈希记录下来；一次无法被重放的训练也就无法被调试。
6. **逐次调用的成本记账**：延迟、金钱成本、返回的词元数；没有它们，节 17.6中的价格 c 就没有可辩护的单位。

7. **精确词元的轨迹记录**，而不是一份留待重新渲染的聊天记录 (Cheng et al., 2025)。

截至 2026 年年中，各个框架对这些要求的覆盖并不均衡。随 GiGPO(Feng et al., 2025b) 一同发布的 `verl-agent`，是 `verl`(Sheng et al., 2025) 事实上的参考多轮扩展。Agent-R1(Cheng et al., 2025) 贡献了步级记录这一抽象，并附带一个有用的否定结果：保留全部上下文并不总是最优的——在它的基准上，滑动窗口胜过只追加。AgentGym-RL(Xi et al., 2025) 把各个环境作为独立服务暴露在同一个 HTTP 接口之后，而 AgentRL(Zhang et al., 2025b) 是最完整的异步设计：中央控制器之下的容器化环境工作单元、任务级的优势归一化，以及为维持探索而刻意保留的陈旧采样。Wang et al. (2025b) 与Luo et al. (2025c) 则把一个既有的智能体当作黑箱的转移发生器来处理。

工程实践: 一份可作为起点的配置

对一个 7–8B 模型、配一个检索工具和一个 Python 沙箱的首次工具集成训练，下面是一份站得住脚的默认设置；其中每一个数字都是有待挪动的起点，而不是值得信赖的推荐。

- **环境**。冻结语料库，固定快照，取前 5 条检索结果。沙箱超时 5 s，观测上限 1024 词元，轮数上限 $K = 8$，总词元预算 16k。错误以文本形式返回。内容寻址缓存。
- **采样**。$G = 16$，温度 1.0，异步且陈旧度限制在一个策略版本以内（节 21.6）；把空轨迹从损失中剔除，并保留在组统计量中。
- **目标函数**。带 clip-higher 的 GRPO （$\varepsilon_{\text{low}} = 0.20$，$\varepsilon_{\text{high}} = 0.28$），不加 KL 惩罚，动态采样（节 12.4），只在动作词元上的词元级损失。
- **奖励**。验证器结果，以模式合法性与证据接地设门；不设调用奖励，也不设成功奖励；价格从 $c = 0$ 起在训练后半程退火到一个在开发集上至多损失一个准确率点的取值。
- **辅助**。观测预测损失(17.9)，取 $\lambda \approx 0.1$，并在同一随机种子下与 $\lambda = 0$ 做消融对照。
- **每步都记录**。平均调用次数、调用成功率、生产率(17.3)、平均观测意外程度、空轮率、缓存命中率、触及轮数上限的轨迹比例，以及分开记录的 pass@1 与 pass@8。只记录奖励的话，本章中的失效你一个也看不到。

代码 17.3: 一个具备算法 17.1所假定四项性质的工具客户端：一个硬超时、把结构化错误作为观测返回而不是抛出、以语料库快照为键的内容寻址缓存，以及逐次调用的记账。错误也会被缓存，这样重放一条采样轨迹时看到的就是它当初看到的那次失败。

```
import hashlib, json, time

class ToolClient:
    def __init__(self, server, cache, snapshot, timeout=5.0, max_chars=4096):
        self.server, self.cache, self.snapshot = server, cache, snapshot
        self.timeout, self.max_chars = timeout, max_chars
        self.calls, self.hits, self.spend = 0, 0, 0.0

    def _key(self, name, args):
        blob = json.dumps({"tool": name, "args": args, "snap": self.snapshot},
                          sort_keys=True, separators=(",", ":"))
        return hashlib.sha256(blob.encode()).hexdigest()

    def call(self, name, args):
        self.calls += 1
        key = self._key(name, args)
        hit = self.cache.get(key)
        if hit is not None:
            self.hits += 1
            return hit
        t0 = time.monotonic()
        try:
            text, err = self.server.invoke(name, args, timeout=self.timeout), None
        except TimeoutError:
            text, err = "", f"tool timed out after {self.timeout:g}s"
        except Exception as exc:                # 绝不把异常传播进采样循环
            text, err = "", f"{type(exc).__name__}: {exc}"
        obs = Observation(text=text[: self.max_chars],
                          truncated=len(text) > self.max_chars,
                          error=err, latency=time.monotonic() - t0)
        self.spend += self.server.price(name, args)
        self.cache.put(key, obs)                # 失败也一并缓存
        return obs
```

17.10 哪些是真正已知的

表 17.5按文献实际支撑得起的证据层级来陈述各项论断，格式与 表 14.2相同。其分布值得注意：已被复现的论断全都关乎训练的机制，而每一个关于迁移的论断要么有争

议、要么只有单篇证据。这正是一个已经解决了自己的工程、却尚未解决自己的科学的领域应有的形状。

有两条是实践者最容易想当然的。跨工具集的迁移是本领域最大的一个未被测量的量：一种说得通的故事是，强化学习教会了一个关于「何时该去寻求外部信息」的一般策略，那它就应当迁移；而同样说得通的另一种故事是，它教会的是一种与模式绑定的习惯，那它就不该迁移。截至 2026 年年中，还不存在干净的留出工具实验——「留出」意为在工具名称、参数结构与观测格式上都没有重叠——而间接证据略偏悲观，因为Tu et al. (2026) 发现环境多样性比任务数量更要紧，而如果学到的策略是狭窄的，这正是你会预期到的结果。Zhou et al. (2026) 把数千个工具接进了真实的模型上下文协议 (Anthropic, 2024) 服务，使这个实验便宜到可以做了，也该有人去做。

代码即动作对比离散工具这件事，两个方向上都是靠轶事在争论。受控比较应当固定模型、固定计算量、固定可达的能力，只改变这些能力是以沙箱形式还是以带类型的工具形式暴露出来。在它被做出来之前，请按节 17.2的工程依据来定夺：当你必须逐次地为调用定价或审计时用离散工具，当组合性占主导时用代码。

本章小结

- 工具是一个谕示器，它把一项不可靠的串行计算、或一条缺失的事实，转换成一段可读的观测。它之所以改变能力边界，是因为它把一个 I 类任务——其答案在任何采样预算下都落在策略支撑集之外——转换成了一个在少数几个词元上的决策问题，而决策正是结果奖励强化学习所训练的东西。
- 关于强化学习能引出什么的支撑集保持界（节 1.8）并不适用于一个以新信息为条件的策略。这就是「在静态强化学习只能在边界内变锐利之处，工具使用强化学习却能拓宽 pass@k 边界」的形式化理由 (PASS@(k,T) Analysis, 2026)。
- 交错模式有四种变体，各自的取舍确有不同。代码即动作买来的是组合性，付出的是无声的错误、粗粒度的信用分配，以及逐次调用记账的丧失。并行调用买来的是延迟，放弃的是自适应性，而这在多跳任务上是致命的。
- 检索是最干净的设定，因为验证器精确而工具便宜；这一领域几乎每一条信念都是在那里确立的。它的两个结构性问题——检索器的失败作为策略错误进入梯度，以及先猜后确认式的作弊——都不是靠更好的优化器能解决的。

- 策略学会卸载的，恰恰是那些它不可靠的子计算，而节 17.4.2说明了为什么组相对强化学习能仅从结果奖励中找到那条规则。这个论断可以通过把已卸载的子计算强制拉回上下文中并比较错误率来测量；而它被测量得远远太少。
- 奖励工具使用，教会的是先学会调用、后学会使用。奖励工具成功更糟：成功概率与信息含量呈负相关，因此这项补贴选择出的是没有信息量的查询（定理 17.2）。Li et al. (2025g) 那个失败的可执行性惩罚，是同一效应的反号版本。
- 奖励结果，把格式与证据设为门，给资源定价；并且在没有承重凭据之前，绝不奖励任何中间动作——而「承重」是有操作定义的：把观测消掉，然后重新采样。
- 没有价格就不存在正确的工具调用次数。在每次调用价格为 c 时，最优规则是当且仅当信息期望价值超过 c 时调用（节 17.6.2）；扫描 c 会描出一条凹的「准确率—调用次数」前沿，而这条前沿才是可报告的对象。价格应当向上退火，因为惩罚项的梯度方差比结果项的低，会被先学到。
- 把观测词元掩出策略梯度是这个估计量的一条性质，而不是一条方差启发式规则——但这对于「把那些词元用作预测目标」什么都没说。在观测上加一个辅助的下一词元损失是免费的、在奖励稀疏之处是稠密的，而且无法被作弊，因为它是一个似然而不是一个奖励。两篇独立的 2026 年论文报告了很大的增益 (ECHO, 2026; PaW, 2026)；这是当前可得的最有前景的廉价干预，而且尚未被第三个团队复现。
- 已被复现的东西都关乎训练机制。而每一个关于迁移的论断——跨工具集、跨动作表示——要么有争议，要么根本没被测量过。

延伸阅读

先读Jin et al. (2025a)，而且要为那条惯例、而不是为那些数字去读：正是这篇论文确立了观测掩码、交错格式，以及只用结果奖励这一纪律，此后的一切都是它的变体。把它与 Chen et al. (2025b) 配读，后者是同一想法最精简的版本，因而也最清楚；再配上 Sun et al. (2025)，去读那个决定你的实验是否可复现的模拟环境论证。

关于代码与计算，该读的一对是Li et al. (2025g) 与Feng et al. (2025a)，按此顺序，理由却相反。ToRL 是一项细致的从基座做强化学习的研究，带着一个重要的否定结果——可执行性惩罚没有帮助——而奖励设计中的否定结果比肯定结果更稀有也更有用。ReTool 则是那个计算效率论证：工具集成的强化学习用纯文本强化学习三分之一的步数就达到了目标。然后读Xue et al. (2025)，它很短，其教益是：这一文献中所报告的

最大单项增益来自一次稳定性修补，而不是一项奖励设计；如果你从本章只带走一个方法论习惯，那就让它是——在寻找更好的目标函数之前，先去找空轮。

Wang et al. (2025a) 是效率方面唯一一篇值得细读的论文，因为它在结构上的那些选择——一个参照组统计量的乘性节俭因子——的适用范围远超出它自己的设定。请把它与Yu et al. (2025d) 对照着读，后者从纯经验的方向到达了关于「审慎而稀疏地调用」的同一结论，而且是智能体式强化学习中最有用的单篇配方研究。至于 2026 年的这一转向，ECHO (2026) 与PaW (2026) 应当放在一起并按此顺序读：ECHO 是为了那个引人注目的结果与常数优势的实现技巧，PaW 则是为了那个关于「辅助损失为何不会腐蚀策略学习」的更审慎的论证。至于深度研究这套栈，Li et al. (2025b) 是方法论的那一篇，Tongyi DeepResearch Team (2025) 是工程的那一篇；综述 (Zhang et al., 2026b) 只用来定位，别的就别指望了，因为它的算法目录基本上是单轮的。

习题

17.1（是可靠性，不是表达力。）用节 17.1的模型——$\Theta(n^2)$ 次基本步骤，每一步独立地以概率 $1-\epsilon$ 正确——推导上下文内 n 位乘法的 pass@k 关于 n、ϵ 与 k 的表达式。对 $\epsilon = 5\times10^{-3}$，求出使 pass@256 超过 0.5 的最大 n。然后解释为什么在 k 个样本上做多数投票挽不回这项损失，并指出当它部分挽回时，模型中被打破的是哪一条假设。

17.2（两项补贴，算个明白。）（a）在一个原本只有结果奖励的奖励中加入 $+\beta\mathbb{1}[n \geq 1]$，并在三种情形下计算式 (11.3)的组相对优势：G 个成员全都调用、全都不调用，以及有比例 f 的成员调用。证明前两种情形下该项相消，把第三种情形下它的贡献推导成 f 与奖励标准差的函数，并说明在什么条件下一个调用自举是无害的。（b）现在构造一个含三个查询的查询空间：一个总是成功但没有信息量，一个以概率 0.4 成功且能确定答案，一个以概率 0.8 成功且能把答案的熵减半。计算在只有结果奖励下，以及在结果奖励加 $\beta\sigma$ 且 $\beta \in \{0.1, 0.5\}$ 下的最优查询，数值上验证 定理 17.2，并报告使最优查询发生改变的最小 β。

17.3（前沿是凹的。）证明节 17.6.2中的凹性论断：可达的 $(\bar{r}, \bar{n})$ 对所构成的集合具有一条凹的上边界，而优化(17.4) 的策略位于该边界上斜率为 c 之处。然后构造一个两轮的例子，其中检索工具以概率 α 返回答案，并把最优期望调用次数推导成 α、q 与 c 的显式函数。最优策略在 c 取何值时变为确定性的?

17.4（**不加掩码，就是一个系数取错了的世界模型。**）证明把观测词元纳入 (17.8)的求和，恰好添加了 $A(\tau)\sum_k \nabla_\theta \log \pi_\theta(o^{(k)}\mid\cdot)$，且它等于一个观测词元上的下一词元预测损失按 $-A(\tau)$ 加权后的梯度。据此推出 ECHO 的(17.9)就是把该项的系数换成一个常数 $\lambda > 0$，并用两句话解释为什么符号与常数性两者都是关键。

17.5（**承重凭据，把成本算出来。**）按节 17.5设计那个把一个动作认证为承重的消融：把 $o^{(k)}$ 换成一个空观测，从第 k 轮重新采样，并比较结果。写出该估计量、它关于重采样次数的方差，以及对一条 $K = 8$ 的轨迹而言它相对一次普通采样的成本。然后用节 17.8中的意外程度 $s^{(k)}$ 提出一个廉价的近似，并说明它会漏掉什么。

17.6 ⋆（**工具技能会迁移吗?**）设计那个能了结表 17.5中该项争议论断的留出工具实验。请具体说明：你如何构造两个在名称、参数结构与观测格式上都没有重叠、同时可达能力又配平的工具集；在各个实验臂之间你固定住什么；哪些基线能把「学到了一个关于何时去寻求信息的一般策略」与「学到了一套模式」区分开；以及什么样的结果会证伪其中每一个假设。然后论证Zhou et al. (2026) 的模型上下文协议环境是否使这个构造可行，以及它们的哪些性质让阴性对照变得困难。

17.7 ⋆（**按信息量而不是按次数定价。**）把(17.4)中的每次调用价格换成一个正比于 $\max(0, \kappa - s^{(k)})$ 的价格，使得只有不意外的调用才被收费。推导与(17.7)类似的修改后调用规则。然后指出其作弊通道：刻画那个通过恶化自己的观测预测来最大化修改后目标函数的策略，界定它能由此攫取的奖励，并提出一项防御——例如在一个冻结的参考策略下计算 $s^{(k)}$。讨论这项防御的代价，以及它会破坏什么。

17.8 †（**把前沿造出来。**）在一个 1–3B 模型上实现工具集成的 GRPO，配两个工具：一个 Python 沙箱和一个在冻结语料库上的检索工具。采样用算法 17.1，奖励用代码 17.2。在至少五个取值上扫描价格 c，其中包含 $c = 0$，并给出（a）带误差棒的节 17.6.2那条「准确率—调用次数」前沿，（b）同一坐标轴上(17.3)的生产率曲线，(c) 在最大与最小的 c 下，调用次数对训练步数的曲线，它应当表现出节 17.6中那种信噪比不对称。然后在固定随机种子下加入取 $\lambda = 0.1$ 的(17.9)，报告前沿是否移动，并一并给出 $\lambda \in \{0, 0.1\}$ 时留出观测上的交叉熵。

17.9 †（**审计元认知这一论断。**）取任意一个经过工具强化学习训练的检查点，你自己的或公开的都行。从留出轨迹中抽取出它选择卸载的那些子计算，以及一组与之匹配、它选择在上下文中完成的子计算。通过把轨迹预填到那一点并封掉工具，强制每一个已卸载的子计算回到上下文中完成，并测量两组上的错误率。报告这两个比

率及其置信区间，把它们对子计算长度作图，并说明数据支持的是节 17.4.2还是一种文体习惯的解释。请把基座模型作为对照纳入。

算法 17.1 带掩码、超时与精确词元簿记的工具集成采样。返回的掩码是使该轨迹能被 GRPO 系更新（式 (11.2)）使用的前提；返回的计数器则是使算法 17.2成为可能的前提。这里没有任何东西是特定于某一类工具的。

输入： 策略 π_θ，工具集 $\mathcal{T}$，轮数上限 K，词元预算 B，每次调用的超时 Δ，观测长度上限 L_o

$h \leftarrow$ 提示词元；$m \leftarrow \mathbf{0}$；$n \leftarrow 0$；$e \leftarrow 0$；$k \leftarrow 0$
while $k < K$ **and** $|h| < B$ **do**
  采样片段 $a \sim \pi_\theta(\cdot \mid h)$，在工具定界符、`<eos>` 或预算处停止
  把 a 追加到 h 并置 $m_t \leftarrow 1$ ▷ 原样存下采样得到的词元 id
  if a 含有终局答案 **or** 以 `<eos>` 结尾 **then**
    break
  end if
  $(\text{name}, \text{args}) \leftarrow \textsc{Parse}(a)$
  if 解析失败 **or** name $\notin \mathcal{T}$ **then**
    $o \leftarrow$ 模式错误字符串 ▷ 这是一个观测，不是一个异常
  else
    $o \leftarrow T_{\text{name}}(\text{args})$，超时为 Δ，经由代码 17.3的缓存
    $n \leftarrow n + 1$；$e \leftarrow e + \mathbb{1}[o \text{ 是错误或超时}]$
  end if
  把 $\textsc{Truncate}(o, L_o)$ 追加到 h 并置 $m_t \leftarrow 0$
  $k \leftarrow k + 1$
end while
if 没有发出终局答案 **then**
  把 τ 标记为**空** ▷ 排除出损失，保留在基线中
end if
return h，掩码 m，动作片段，计数器 (n, e, k)，逐次调用延迟

表 17.2: 两条实验上已经成熟的工具强化学习脉络。每一条目都使用带观测掩码的 GRPO 系目标函数；差别在于奖励、初始化与稳定性机制。数字均为所引工作报告的原值，跨行之间不可比较，因为基座模型与评测协议各不相同。

系统	奖励	区别性选择	所报告的发现
检索与搜索			
Search-R1 (Jin et al., 2025a)	精确匹配	引入了检索词元掩码	7B 上比 RAG 高 +41%，3B 上高 +20%
R1-Searcher (Song et al., 2025a)	先调用，后答案	两阶段的结果监督	无需蒸馏即可从基座模型中引出检索
ReSearch (Chen et al., 2025b)	结果 + 格式	搜索是推理链的一部分	完全没有步级监督，7B 与 32B
StepSearch (Wang et al., 2025j)	信息增益、冗余惩罚	带子问题监督的逐步 PPO	就检索而言步级奖励胜过轨迹级奖励
ZeroSearch (Sun et al., 2025)	结果	模拟引擎、质量课程	14B 模拟器超过实时引擎
代码与计算			
ToRL (Li et al., 2025g)	+1/ − 1，不可执行 −0.5	从基座做 RL-zero，C=1 次调用	可执行性惩罚没有帮助
ReTool (Feng et al., 2025a)	结果	在带代码的 CoT 上做 SFT 冷启动	400 步 AIME 67%，对比纯文本 1080 步 40%
ToolRL (Qian et al., 2025)	分解为名称／参数／取值	系统性的奖励设计研究	工具奖励形状的参照消融
SimpleTIR (Xue et al., 2025)	结果	空轮轨迹过滤	从基座出发 AIME24 22.1 → 50.5
rStar2-Agent (Shang et al., 2025)	结果	工具噪声下的答对即重采样	14B 上 510 步 AIME24 80.6%
OTC-PO (Wang et al., 2025a)	正确性 × 节俭	为调用次数定价（节 17.6）	准确率相当时调用减少 68.3%

表 17.3: 工具使用的各个奖励分量，以及每一个所诱发的病理和一个裁断。「门」意为在结果上做乘法，绝不做加法。「定价」意为减去一项具有可辩护单位的成本。两个标为绝不的行，是最常被弄错的。

分量	度量的是	典型病理	裁断
结果正确性 r_{out}	由验证器判定的答案	稀疏；而且任何弱于任务本身的验证器都可被钻空子（先猜后确认）	**总是**
格式合法性	每次调用都能按模式解析	作为加性奖金时它会与正确性相权衡，于是策略可以靠答得更差来换格式分	**门**
调用成功（非错误）	调用执行成功了	选择出安全而无信息量的调用；见定理 17.2	**绝不**
工具发起调用	至少发生了一次调用	无谓的调用；几百步内即饱和，此后只贡献方差	**绝不**，除非作为有排期的自举
每次调用价格 $-c\,n$	所消耗的资源	过强的价格会先于准确率项被学到（节 17.6）	**定价**
检索相关性	由一个评判模型给观测打分	继承评判模型的盲点；把索引的失败归因给策略	视情况而定
证据接地	答案片段出现在某篇检索到的文档中	可以靠照抄而不理解来满足	**门**
重复惩罚	完全相同的调用被重复	改写措辞的重复可以绕过它	定价

算法 17.2 OTC-PO(Wang et al., 2025a) 风格的成本感知组相对更新。相对算法 11.1仅有的改动是：参照组统计量的节俭因子、价格排期，以及把损失限制在动作词元上的掩码。取 $c_i \equiv 0$ 即退回普通的工具集成 GRPO。

输入： 策略 π_θ，验证器 r_{ver}，组大小 G，轮数上限 K，价格排期 c_i（由 $0 \to c_{\max}$ 退火），节俭映射 ϕ

for 迭代 $i = 1, 2, \ldots$ **do**
  $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$；采样提示 $\mathcal{B}$
  for 每个 $x \in \mathcal{B}$ **do**
    按算法 17.1采样出 $\tau_1, \ldots, \tau_G$，并收集 (n_j, m_j)
    对所有 j，$r_j^{\text{out}} \leftarrow r_{\text{ver}}(\tau_j)$
    参照组统计量的价格
    $\mathcal{C} \leftarrow \left\{ j : r_j^{\text{out}} = 1 \right\}$
    $n_{\text{ref}} \leftarrow \min_{j \in \mathcal{C}} n_j$ **if** $\mathcal{C} \neq \emptyset$ **else** K ▷ 这个问题需要多少次才够
    $r_j \leftarrow r_j^{\text{out}} \cdot \phi(n_j; n_{\text{ref}}) - c_i \max(0, n_j - n_{\text{ref}})$
    优势
    把空轨迹从损失中剔除，但保留在 μ, σ 中 ▷ Xue et al. (2025)
    $A_j \leftarrow (r_j - \mu)/\sigma$；广播 $A_{j,t} \leftarrow m_{j,t} A_j$
  end for
  在 $m_t = 1$ 的词元上上升(11.2)
  记录 $\bar{n}$、$P(\pi_\theta)$、调用成功率、平均观测新颖度 ▷ 诊断节 17.5的那四项
end for
return π_θ，以及对应本次 $c_{\max}$ 的前沿点 $(\bar{r}, \bar{n})$

表 17.4: 在一条工具集成采样轨迹中，每一类词元分别可以拿来做什么。第二列正是 2019–2025 年的惯例与第三列混为一谈的那一列：把观测词元排除在策略梯度之外是必须的，而这对「它们是否可以被预测」什么都没说。

词元	进策略梯度?	作预测目标?	可作弊?	稠密度与现状
提示	否	否	—	固定；按普遍惯例被掩掉
动作（智能体）	**是**	否	是，经由奖励	每条轨迹一个优势；策略学习的唯一来源
观测（环境）	**绝不**——它不是(17.8)中的一项	**是**——ECHO、PaW	否：这是一个似然，不是奖励	无论有无奖励，每轮一个；两篇独立的 2026 年论文
观测意外程度 $s^{(k)}$	作为优势的调节量	—	是，靠把预测做差	免费；截至 2026 年年中仍属推测

表 17.5: 关于工具集成强化学习的各项论断，及其诚实的证据层级。R = 已在独立团队与不同模型家族之间复现，C = 有争议或有记录在案的反例，S = 单篇或两篇论文，可能脆弱。最后一列才是可据以行动的那一列。

论断	层级	什么会证伪它
只用结果奖励就足以教会何时调用，无需步级监督	R	在某个检索或 TIR 设定中，配平的纯结果奖励强化学习未能把调用率提到基座模型之上
观测词元必须排除在策略梯度项之外	R	没有经验性的东西能证伪它；它是(17.8)的一条性质。有争议的是它们能否被另作他用
查询重述与多跳分解无需监督即会涌现	R	一次轨迹审计发现这些行为在基座模型中以同样的比率出现
朴素的工具强化学习会过度调用；给调用定价能在调用次数只剩一小部分时保持准确率	R	一次定价训练，其准确率随 c 从 $c = 0$ 起单调下降
在静态强化学习无法拓宽 pass@k 边界之处，工具强化学习能拓宽它	S	在某个组合式检索基准上，其优势随 k 增大而消失 (PASS@(k,T) Analysis, 2026)
预测观测词元几乎免费且帮助显著	S（两篇论文）	第三个团队在配平计算量下报告没有增益，或增益在类终端环境之外消失
一个模拟检索器可以匹敌甚至胜过实时引擎	C	在第二个语料库上未能复现Sun et al. (2025) 的 14B 结果
步级奖励有帮助	C	已经有反例：Li et al. (2025g)、Yu et al. (2025d)
代码即动作优于一份离散工具清单	C	在配平计算量与配平工具覆盖面的条件下，尚不存在受控比较
工具使用技能可以跨工具集迁移	C	一次没有模式重叠的留出工具评测；最接近的证据是 Tu et al. (2026) 的「多样性优于数量」的规模曲线

第18章

长时域智能体强化学习

核心思想：到了一百轮次，两轮次训练里的每一处小麻烦都会变成决定性的约束

一个二值奖励必须摊到 10^5 个生成词元上；轨迹装不进上下文窗口，于是遗忘本身成了策略的一部分；单条采样轨迹要花掉几分钟到几小时、且其中大部分时间加速器都在空转；而在同步批次里，最慢的那条轨迹决定了整步的耗时。这四项约束——稀疏、上下文、成本、方差——并不是彼此独立的烦恼。它们是同一个变化（时域的增长）的四个后果，本章也就相应地组织为针对它们的四轮进攻。

第16章确立了智能体 MDP（节 16.2）、它的轮次级约简（节 16.3），以及「环境产生的词元要从策略损失中掩蔽掉」这一约定（节 16.4）。第17章处理了工具调用的奖励设计（节 17.5）与工具使用的效率（节 17.6）。所有这些机制在此都默认成立。这里改变的是规模：从工具融合推理的一到二十轮次，增长到软件工程、终端或计算机操作智能体的三十到二百轮次。这个变化在陈述上是量的，在后果上却是质的。

全章中，$\tau = (h^{(1)}, a^{(1)}, o^{(1)}, \ldots, a^{(K)}, o^{(K)})$ 表示一条 K 轮次的轨迹，$a^{(k)}$ 是策略在第 k 轮次生成的那段词元，$o^{(k)}$ 是环境的回应，而 $h^{(k)}$ 是策略在第 k 轮次真正据以条件化的上下文——在本章中，这个量不再简单地等于此前一切内容的拼接。

18.1 四项约束的量化

先做一笔账。下面的数字是算术演算而非实测，但其中的配置参数取自已发表的系统：NVIDIA (2026d) 运行 OpenHands 循环最多 200 轮次，奖励是来自真值测试的二值信号；Luo et al. (2025a) 在 R2E-Gym 任务上以 64k 上下文、环境步数上限 100 训练 Qwen3-32B。取一个 $K = 100$ 的软件工程回合。每一轮次策略大约输出 500 个词元的斟酌加上一条命令，环境则返回大约 1,000 个词元的观测——一份文件清单、一段栈回溯、一份测试报告。于是：

表 18.1: 长时域智能体强化学习的四项约束、随之增长的量，以及分别攻击它们的各节。「如何增长」一列中的每一条都是关于时域 K 或可训练词元数 N 的陈述；本表要说明的是，这四者都是同一个变化的后果。

	约束	什么在增长，如何增长	在何处攻击
C1	奖励稀疏	每条轨迹的信息量固定为 1 比特，而决策数按 N 增长	节 18.2 and 18.3
C2	上下文	历史按 K 增长、注意力开销按 K^2 增长；并会超出窗口	节 18.4
C3	采样成本	每回合挂钟时间按 K 增长，且以环境延迟为主	节 18.5
C4	方差与不稳定	策略变化在 K 个轮次上复合；轨迹分布比策略移动得更快	节 18.6 and 18.7

- **可训练词元**。$K \times 500 = 5 \times 10^4$ 个词元携带梯度，其余都被掩蔽。
- **上下文**。到第 100 轮次时，朴素拼接的历史已有 1.5×10^5 个词元，是 64k 窗口的两倍多。必须丢掉一些东西，而做出丢弃决定的正是策略本身。
- **上下文词元步**。把各轮次的上下文长度求和，得到 $\sum_k \left|h^{(k)}\right| \approx K^2 \times 750 \approx 7.5 \times 10^6$。决定注意力开销与 KV 缓存占用的是这个量，而不是可训练词元数。
- **挂钟时间**。一次测试套件要跑几秒到一分钟。按每轮次 20 秒的环境延迟计，这个回合有半个多小时耗在环境里，而在此期间，刚刚生成了上一个动作的加速器无事可做。
- **奖励**。一个比特。

对比一条单轮次数学采样轨迹：8×10^3 个生成词元，一阶近似下同样是 8×10^3 个上下文词元步，没有环境，一到两分钟的生成，同样也是一个比特。以「每比特奖励所耗的上下文词元步」来衡量，智能体回合大约要差三个数量级；以「每比特奖励所耗的挂钟时间」来衡量则差一到两个数量级，而节 18.5的掉队者效应还会在此之上再乘一次。

图 18.1预示了本章余下部分要分析的对象。它有三个特征值得现在就点明：奖励在最右端，而且只有这一个（C1）；分片之所以存在是因为窗口有限，且其边界由策略选定（C2）；片段数是一个随机变量，同一提示的不同采样轨迹之间并不相同，这会破坏 第12章的组相对记账方式（C4）。

18.2 一百轮次下的奖励稀疏

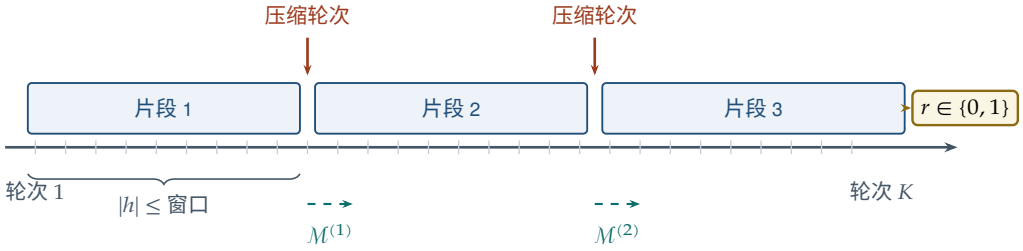


图 18.1: 压缩之下的一条长轨迹。回合被压缩轮次（锈色箭头）切成若干片段，每次压缩读入刚刚结束的那个片段，写出一份摘要 $M^{(j)}$ 作为下一片段上下文的种子（青色虚线）。片段内部的上下文长度有界，但整个回合的上下文长度没有界。最右端唯一的终局奖励必须分配到每个片段中的每一个可训练词元上——包括压缩轮次自身的词元，因为它们和其他动作一样都是策略动作。此图为示意图；实际中片段长度的差异极大。

18.2.1 一个比特，以及它能买到什么

一个终局的二值奖励至多携带关于该轨迹的一个比特信息，而实际上还更少：若策略在某个提示上以概率 p 成功，则结果携带 $\mathbb{H}[\mathrm{Bernoulli}(p)] \le 1$ 比特，且当 $p \to 0$ 或 1 时趋于零。这笔预算必须分配给 $N \approx 5 \times 10^4$ 个词元级决策，或 $K = 100$ 个轮次级决策。对比监督微调：那里每个词元都提供一个取自 10^5 量级词表的目标，因而每个位置最多给出 $\log_2 |\mathcal{U}| \approx 17$ 比特的约束。两者之比约为 10^6。

这是一个关于信息量的陈述，而不是关于可学习性的陈述——每回合一个比特，在数百万个回合上累积起来，总量绰绰有余。真正要问的是，这份累积信号在估计量的方差之下还能存活多少，而时域正是在此处直接介入的。

推导: 组要多大，广播式优势才能说明点什么?

取均匀广播估计量：每条轨迹一个组相对优势，施加到全部 N 个可训练词元上。把轨迹得分写作 $s(\tau) = \nabla_\theta \log \pi_\theta(\tau) = \sum_{n=1}^{N} g_n$，并把逐词元得分 g_n 建模为零均值、互不相关、且 $\mathbb{E}\,\|g_n\|^2 = F$，于是 $\mathbb{E}\,\|s\|^2 = NF$。在二值奖励、成功概率 p、组大小 G 之下，估计量 $\hat{g} = \frac{1}{G}\sum_i (r_i - \bar{r})\, s(\tau_i)$ 满足

$$\mathbb{E}\,\|\hat{g}\|^2 \;\approx\; \frac{p(1-p)\,NF}{G}. \tag{18.1}$$

现在假设结果实际上由 $m \ll N$ 个决定性位置所决定，即只有这 m 个位置满足 $\mathbb{E}[r\, g_n] \neq 0$，每个位置的贡献量级至多为 $\kappa\sqrt{F}$。那么 $\|\nabla \mathcal{J}\| \lesssim \kappa m \sqrt{F}$，一步梯度的

信噪比为

$$\mathrm{SNR} \approx \frac{\kappa\, m\sqrt{G}}{\sqrt{p(1-p)\,N}} = \frac{2\kappa\, m\sqrt{G}}{\sqrt{N}} \quad (p=\tfrac{1}{2}), \qquad \text{所以} \qquad G \gtrsim \frac{N}{4\kappa^2 m^2}. \tag{18.2}$$

取 $\kappa = 1$：一条 $N = 8\times 10^3$、$m = 50$ 个决定性词元的单轮次数学采样轨迹只需 $G \gtrsim 1$——这正是纯结果式 RLVR 之所以能奏效的原因。而一条百轮次智能体采样轨迹，$N = 5\times 10^4$、$m = 5$ 个决定性决策（打开正确的文件、形成正确的假设、写出正确的补丁），需要 $G \gtrsim 500$。没有人会在百轮次采样上跑 $G = 500$。

两点说明。梯度还会在批次的各个提示上做平均，所以(18.2)界定的是 单个提示的组能分辨出什么——之所以这个量才是相关的，恰恰因为组相对方法正是把该组当作基线。另外，得分互不相关这一建模在两个方向上都有偏差：同一轨迹内的词元是强相关的，这会减小有效的 N；而 m 则是未知的。请把它读作对某种标度关系的解释，而不是对 G 取值的处方。

核心思想: 稀疏与退化是同一个问题的两副面孔

节 12.4指出，若一个组内全部 G 个样本得到相同奖励，它贡献的梯度恰为零。在长时域下这是常态而非边角情形：在一个困难的软件工程任务上，一个称职的策略也许只在 2–10% 的实例上成功，于是当 $G = 8$ 时，一个组发生退化的概率超过 0.9。花在这些采样上的算力不只是低效，而是零价值的，而且它还是整条流水线中最昂贵的算力。因此，在这个时域下，动态采样、$\alpha = B/(B-M)$ 重标定与难度课程都不是锦上添花；它们是「一次能学到东西的训练」和「一次学不到东西的训练」之间的区别。

18.2.2 各种补救，按侵入性由弱到强

基于环境状态的塑形。 环境知道的通常比终局检查所报告的更多：测试框架知道有多少测试通过，构建系统知道编译是否成功，浏览器知道页面是否加载完成。把这些转成稠密的逐轮次奖励 $r^{(k)} = \phi(s^{(k)}) - \phi(s^{(k-1)})$（$\phi$ 为环境状态的势函数）是经典构造；而当 ϕ 确实是状态的函数、而非智能体对状态的汇报的函数时，它既廉价又难以伪造。

里程碑与子目标。 里程碑奖励是环境状态上的一个程序化谓词——「那个失败的测试现在失败的原因不同了」「搜索返回了一篇包含标准实体的文档」——在它第一次为真

时支付一次。把它们规定出来属于任务撰写工作，而由谁来写、代价几何，归第19章管辖。Wang et al. (2025j) 在一个受限领域给出了最清晰的正面证据：带有子问题级信息增益奖励与冗余惩罚的分步 PPO，在搜索智能体上胜过轨迹级奖励。

进度奖励。 对「这一轮次让我们靠近了多少」的一个习得或评判得到的估计，来自轮次级奖励模型 (Wei et al., 2025b)，或来自一个掌握了行动者所没有的信息的评论家。Zhou et al. (2025a) 就在协作式软件任务上训练了这样一个享有特权信息的评论家：评论家在训练时能看到参考解，而行动者永远看不到。

不是奖励的稠密目标。 2026 年最有意思的进展彻底绕开了规范设定问题。观测词元被从策略损失中掩蔽掉（节 16.4），但策略本来就以它们为条件，而且在同一次前向传播中本来就算出了它们的对数几率。ECHO (2026) 与PaW (2026) 各自独立地提出在这些观测词元上加一项下一词元预测损失，使模型恰好从强化学习目标所丢弃的那些词元中学到一个世界模型：

$$\mathcal{L} = \mathcal{L}_{\text{RL}}(\text{action tokens}) + \lambda \cdot \mathop{\mathbb{E}}_{t \in O'}[-\log \pi_\theta(y_t \mid y_{<t})], \tag{18.3}$$

其中 O' 是观测词元位置的一个选定子集。这个信号恰恰在奖励稀疏之处是稠密的，因为无论该回合是否挣到奖励，每一轮次都会产生观测；而且环境预测损失无法像塑形奖励那样被作弊——正确预测标准输出并不是对别的什么东西的代理，它本身就是那件事。ECHO (2026) 报告在 Terminal-Bench 2.0 上把 Qwen3-8B 与 Qwen3-14B 的 GRPO pass@1 翻了一番；更令人意外的是，他们报告在某些设置下，仅用预测损失、关掉奖励也能改善留出任务上的表现。这两篇论文都出自 2026 年，且都尚未被独立复现；请把机制视作可信，把效应量视作暂定。

常见陷阱: 每一个塑形奖励都是一份规范，而长时域给了优化器更大的腾挪空间

在智能体设置中，塑形奖励比在单轮次 RLVR 中更危险，原因是结构性的：一个百轮次策略在每个回合里有一百次机会去找到触发你的谓词的廉价办法，而本来能抓住它的终局检查只是远处一个带噪的比特。文献中不乏实证。Li et al. (2025g) 报告，对不可执行代码施加惩罚，相对于纯结果奖励反而有害，看似是因为这让策略在生成代码这件事上变得保守。Shen et al. (2026b) 记录了在合成终端环境中通过过度生成工具调用而实现的奖励作弊，并不得不修改奖励来把它稳住。Cai et al. (2025) 则必须显式地把模拟用户行为失当的那些采样轨迹掩蔽掉，因为策略

学会了去诱发这种失当行为。节 5.8中的一切在此都适用，只是动作空间更大、可供搜索的时间更长。

18.3 轮次级信用分配

第13章铺陈了从「每条轨迹一个优势」到「每个词元一个优势」的整个谱系，并得出一个收敛性的结论：对单轮次推理而言，更细的信用分配很少能抵回其成本。智能体情形是这个结论真正存疑的唯一场合，原因有两条，值得分开来讲。其一，时域长了 10–100 倍，因此(18.2)咬得更狠。其二，中间信号在存在时来自环境而非来自一个习得的模型，所以节 13.2.3中的那些反对意见——标签带噪、代理可被作弊——在此并不以同样的方式成立。

18.3.1 均匀广播，以及它为何是默认做法

基线方法是每条轨迹算一个组相对优势，再通过节 16.4的动作掩码 $m_{i,t} \in \{0,1\}$ 把它广播到每一个由智能体生成的词元上：

$$A_i = \frac{r_i - \mathrm{mean}(\{r_j\}_{j=1}^{G})}{\mathrm{std}(\{r_j\}_{j=1}^{G})}, \qquad A_{i,t} = m_{i,t}\, A_i. \tag{18.4}$$

Search-R1(Jin et al., 2025a)、ReSearch(Chen et al., 2025b)、DeepSWE-Preview(Luo et al., 2025a) 以及多数生产配方实际用的就是这个。它的优点是实实在在的：在轨迹层面无偏，不需要评论家，不需要额外采样，也不需要对环境结构作任何假设。它的缺陷一句话就能说清：智能体选错了要打开的文件的那一轮次，与它写出了一句正确文档字符串的那一轮次，得到的信用完全相同。

18.3.2 轮次级折扣，以及 γ 此时的含义

智能体 MDP 的综述式表述带有一个折扣 $0 < \gamma < 1$(Zhang et al., 2026b)，让人很想把它照搬过来。定义轮次级回报

$$G_i^{(k)} = \sum_{j=k}^{K_i} \gamma^{j-k}\, r_i^{(j)}, \tag{18.5}$$

并立刻注意到在纯终局奖励下会发生什么：第 k 轮次收到 $\gamma^{K-k} r$。当 $\gamma = 0.99$、$K = 100$ 时，第一轮次收到 $0.37r$；当 $\gamma = 0.95$ 时它收到 $0.006r$。既然一个软件工程回合的开头

几轮次——读哪个文件、检验哪个假设——往往才是决定性的，逐轮次折扣恰恰在最不该衰减的地方衰减了信用。节 2.3曾主张在词元 MDP 中取 $\gamma = 1$；在轮次粒度上，这个论证只会更强，不会更弱。

如果折扣是按词元而非按轮次施加的，失败还会更严重。一个由 20 个词元的 shell 命令构成的轮次，与一个由 2,000 个词元的斟酌构成的轮次，同样都是一次有后果的决策；但在逐词元折扣下，后者相对前者被衰减了 γ^{2000} 比 γ^{20} 的倍数。既然同一提示的不同采样轨迹里长短轮次的搭配各不相同，这就把一个建模选择变成了组内方差的来源，而不只是一项偏差。如果你无论如何都想加折扣，请把它加在(18.5)的轮次下标上，绝不要加在词元下标上——而诚实的默认做法仍是取 $\gamma = 1$，把信用结构交给估计量去提供。

18.3.3 通过状态分叉得到的逐轮次蒙特卡洛值

节 13.4为词元 MDP 发展了蒙特卡洛前缀值，在那里它基本上是免费的：状态就是一个字符串，重新进入该状态毫无代价。智能体版本是同一个估计量，只是多了一项要求。把环境恢复到第 k 轮次之后所处的状态 $s^{(k)}$，在当前策略下采样 B 条延续，再对它们的结果取平均：

$$\hat{\mathbb{v}}(s^{(k)}) = \frac{1}{B}\sum_{b=1}^{B} r(\tau^{(b)}), \qquad \tau^{(b)} \sim \pi_\theta(\cdot \mid h^{(k)}, \mathcal{E} \text{ restored to } s^{(k)}). \tag{18.6}$$

轮次优势于是取增量形式 $\hat{\mathbb{v}}(s^{(k)}) - \hat{\mathbb{v}}(s^{(k-1)})$，或取基线形式 $r - \hat{\mathbb{v}}(s^{(k-1)})$，与节 13.4中完全一样，然后广播到第 k 轮次的各个词元上。

核心思想：可快照性是一项带来算法后果的系统要求

在词元 MDP 中，状态就是一个前缀，重置是免费的，藤蔓式估计量之所以可用正是因为这一点。而在智能体 MDP 中，状态包含一个文件系统、一个数据库、一个浏览器会话或一项远程服务。因此，要让(18.6)可实现，环境就必须支持快照与恢复：一层写时复制文件系统、一次容器检查点、一次数据库事务回滚，或对动作前缀的确定性重放。于是，你能用哪些轮次级信用分配方法，是在环境被建造时就决定的，而不是在算法被选定时。这是本书中「基础设施决定可行算法类」这一现象最尖锐的例子，也是 第19章把快照支持当作头等要求而非便利设施的原因。

有三项成本与一项偏差值得说明。生成成本：在 M 个点上各分叉 B 条延续，会把采样成本大约乘以 $MB/2$，而这还是叠加在一个本已是全流水线中最昂贵的每回合成

本之上。存储成本：每条轨迹 M 份容器快照，量级在吉字节。编排成本：从一份快照分出 B 条分支需要 B 个隔离的环境实例，否则它们会互相破坏彼此的状态。至于那项偏差：与确定性的词元 MDP 不同，(18.6)还要在环境随机性上取平均，因此一次不稳定的测试或一条活的网络连接，会让 $\not\Phi$ 比定理 13.4所预测的更嘈杂——而且如果这种不稳定与动作相关，噪声还不是零均值的。

18.3.4 无需分叉的轮次级组相对基线

分叉昂贵到这个地步，以至于该领域最有影响力的想法是绕开它。Feng et al. (2025b) 观察到，在许多环境里，同一提示的不同采样轨迹会重访同一个环境状态。他们的锚定状态分组收集所有反复出现的状态构成集合 $\mathcal{U}$，并对每个反复出现的 $\tilde{s}$，把从 $\tilde{s}$ 出发的所有（动作，回报）对组成一个步级组；在该组内计算组相对优势，就把动作的效应从状态的效应中分离了出来，而这正是基线的用途。最终优势就是 算法 18.1所组装的加权和 $A^E + \omega A^S$，其精妙之处在于，这种分组只需要在已访问过的状态上做哈希表键匹配——不需要评论家，不需要额外采样，也不增加任何挂钟时间。他们报告在 7B 规模上相对 GRPO 在 ALFWorld 上提升 +12.6%、在 WebShop 上提升 +9.1%；更有意思的是，还胜过了带习得评论家的 PPO （ALFWorld 上 90.2 对 80.4）。

局限恰恰就在那条假设上。锚定状态分组需要反复出现且可哈希的状态。ALFWorld 与 WebShop 具备这一点；开放网页浏览与百轮次软件工程则不具备，因为在后者中没有任何两条轨迹会处在同一个仓库状态上。截至 2026 年年中，在状态不重复的情况下恢复轮次级信用仍是一个开放问题（节 18.8）。

18.3.5 跨轮次的留一法，以及把责任定位到具体位置

给定一组经过可比状态的轨迹，最自然的估计量就是把节 6.2.3的留一法基线用在轮次粒度上：对从状态 $\tilde{s}$ 出发的第 i 条轨迹，

$$A_i^{(k)} = G_i^{(k)} - \frac{1}{\left|\mathcal{G}(\tilde{s})\right| - 1} \sum_{j \in \mathcal{G}(\tilde{s}),\, j \neq i} G_j, \tag{18.7}$$

由定理 2.2，只要分组的确定不参考回报，它在给定分组的条件下就是严格无偏的。这个前提很容易被违反：如果你用一个已经看过结果的相似度函数去聚类状态来形成组，那么这个「基线」就依赖于样本本身，无偏性论证随即崩塌。

上述所有方法所攻击的问题有一个名字：一个坏轮次毒害整条轨迹。在均匀广播之下，一条执行了九十九个出色轮次和一个灾难性轮次的轨迹，只会得到一个负优势

表 18.2: 长时域下轮次级信用分配的各种选项，及其各自的代价与假设。「假设」一列才是起作用的那一列：在开放环境中当 $K \gtrsim 50$ 时，下表中的多数方法根本不可用，这也正是均匀广播依然是生产默认做法的原因。

方法	机制	额外成本	假设
均匀广播	每条轨迹一个组相对优势，掩蔽到智能体词元上	无	无
轮次折扣	对轮次级回报做几何加权	无	早期轮次不那么重要（通常为假）
程序化轮次奖励	在环境状态上做势函数塑形	环境内省	状态能暴露进度
轮次奖励模型	习得或评判得到的逐轮次分数	每轮次一次推理	标签存在；可被作弊
锚定状态分组	在自反复出现状态发出的动作之间做组相对优势	哈希表查找	状态会重现且可哈希
状态分叉	从恢复的快照出发得到逐轮次蒙特卡洛值	生成量 $\times MB/2$，M 份快照	环境可快照
轮次留一法	在一个状态组内的无偏基线	除分组外无	分组与回报无关
空转轮次掩蔽	把被毒害的轮次移出损失、留在基线	无	存在一个廉价的逐轮次谓词

并施加到全部一百个轮次上，于是那九十九个好轮次也一并受罚；对称地，一次侥幸的成功会强化沿途的每一个失误。在两轮次的场景里，这只是轻微的低效。在一百轮次的场景里，它是对学习信号的系统性败坏，也是节 18.6中若干坍缩模式背后的机制。

最具可迁移性的定位方法完全不需要任何值估计。Xue et al. (2025) 指出了空转轮次——一个既未产生有效动作、也未给出最终答案的轮次，通常表现为残缺的代码、退化的重复，或过早的序列结束符——是在以分布外的工具反馈为条件之后发生分布漂移的可观测标志。他们的修补值得精确陈述，因为其两半各司其职：把包含空转轮次的轨迹从策略损失中移除，但在优势估计中保留它们。把它们移出损失，可以挡住那些低概率词元所产生的高幅值梯度；把它们留在组统计量里，则保持基线无偏。他们报告从 Qwen2.5-7B 基座模型出发，AIME24 从 22.1 升至 50.5。这一构造可以推广到任何廉价的逐轮次谓词——格式错误的工具调用、无法解析的输出、超出步数预算。

工程实践: 轮次级信用分配：经得起实战的默认设置

先用均匀广播加空转轮次掩蔽；后者不花任何代价，却移除了最大的单一病态。凡是环境本来就已经算出、你不必再去发明谓词的地方，就把环境导出的轮次奖励加上。当且仅当你能核实状态确实会重现时才加锚定状态分组——先做埋点并记录 $|\mathcal{U}|$，再决定要不要相信它有用。只有当你同时具备快照支持、一个严重的稀疏问题以及可以挥霍的算力时，才去动用状态分叉；真要动用时，也请按节 13.4的分辨率论证，在少数几个点上分叉、并取较小的 B：小于大约 $2\sqrt{2p(1-p)/B}$ 的增量，无论你如何花预算都是看不见的。

18.4 上下文长度与历史管理

到第 100 轮次时，朴素拼接的历史已经装不下了。因此智能体必须遗忘，而本节的核心概念性要点是：它如何遗忘是策略的一部分，这意味着遗忘方式有奖励后果，也就意味着它可以被训练。

18.4.1 截断会悄悄改变 MDP

最省事的策略是滑动窗口：保留任务陈述与最近 W 个轮次。Cheng et al. (2025) 报告，在 GSM8K 上滑动窗口胜过只追加的历史——这对「上下文越多越好」的假设是一个有用的纠正。但截断有一个容易忽略、且很难调试的后果。智能体 MDP 在联合状态（上下文，环境状态）上是马尔可夫的，而在只追加的历史下，上下文是智能体已做过的一切的充分统计量。在滑动窗口之下则不然：环境仍然记录着第 3 轮次的效果，但智能体已经看不到第 3 轮次发生过了。此时策略是在一个部分可观测的过程中行动，而其隐藏部分是它自己的历史；特征性的失败是绕圈——重读同一个文件、重跑同一个测试、重发同一条查询——因为「它已经做过这件事」的证据被逐出了上下文，而做这件事的动机却没有被逐出。

常见陷阱: 非马尔可夫性出现在了错误的地方

环境的部分可观测性是正常的，策略可以通过采取收集信息的动作来学会应对。而智能体自身过去的部分可观测性则是病态的，因为没有任何动作能把那份信息找回来：它是被框架删掉的，而不是被世界藏起来的。这正是为什么朴素截断会以一种看上去像能力缺陷、实则是记账缺陷的方式损害长时域表现。如果你确实

> 要截断，请把「被截断了什么」的显式记录往下传——哪怕只是一份已执行命令的清单，也足以打破那个死循环。

18.4.2 把压缩当作一个动作

真正能扩展的策略是上下文压缩（compaction）：当上下文逼近其预算时，就要求策略为刚刚完成的那个片段写一份摘要，下一片段的上下文则重新初始化为（任务陈述，摘要，最近若干轮次）。图 18.1画出了它；算法 18.2实现了它。

核心思想：一次压缩是一个动作，而不是一个预处理步骤

摘要由 π_θ 生成，它改变状态，并决定了智能体在该回合余下部分将知道些什么：它满足成为一个动作的每一条判准。因此压缩词元理应带着与其他动作词元相同的优势进入策略损失，而压缩质量可以由终局奖励端到端地训练出来。凡是把压缩当作固定框架特性来对待的系统——一个独立的摘要器、一个套在可训练循环之外的模板——都把一个往往是整个回合中最有后果的决策留在了未训练状态。Ye et al. (2025a) 与Qiao et al. (2025) 正是出于这个理由，把上下文管理做成主动的、内生的。

代码 18.1: 采样循环内部的一次压缩轮次。真正要紧的两行是掩码赋值——压缩词元属于智能体词元，携带梯度——以及「摘要由生成动作的那同一个策略对象产生」这一事实，从而改进摘要就成了同一次优化的组成部分。

```
COMPACT = ("Summarize the work so far: what you have established, what you have "
           "ruled out, and what you will do next. Be specific about file paths.")

def maybe_compact(ctx, policy, budget, reserve, tail):
    if ctx.n_tokens <= budget - reserve:
        return ctx, None
    summary = policy.generate(ctx.render() + COMPACT)      # 一个*策略动作*
    seg = Segment(tokens=ctx.tokens + summary.tokens,
                  mask=ctx.mask + [1] * len(summary.tokens))  # 可训练
    new = Context(head=ctx.head)                           # 仅任务陈述
    new.append(summary.tokens, mask=1)                     # 向下传递
    for turn in ctx.turns[-tail:]:
        new.append(turn.action, mask=1)
        new.append(turn.observation, mask=0)               # 环境词元：无梯度
    return new, seg
```

压缩带来一个被低估的记账问题。由于压缩次数取决于策略碰巧有多啰嗦，同一提示的两条采样轨迹会产生数量不同、长度差异极大的可训练片段，而组相对优势估计假定被比较的对象是可比的：一个组里，一个成员是三个片段共 40k 词元、另一个成员是一个片段共 12k 词元，这就不叫可比。GLM-5.2 的开发者报告，这一点严重破坏了组式优势估计，以至于他们转向了带词元级优势的评论家式 PPO(Zai, 2026)。该说法出自一篇博客文章且未经核实，但其机制是真实的，而且它是目前可得的、支持评论家在长时域下回归的最有力论据——这与节 12.9的读法一致，即评论家的失败源于可修复的原因，而非结构性的原因。

18.4.3 外部记忆与对自身历史的检索

第三种策略是外部化：与其把历史压缩进上下文，不如把它写到一个存储中——一个草稿文件、一个对过往轮次的向量索引、一份结构化记录——再从中检索。检索于是就是一次普通的工具调用，第17章中的一切在此都适用。

新的东西是训练那个写的决策。Yu et al. (2025a) 训练了一个强化学习策略，由它在每个片段边界上决定保留哪些自然语言记忆词元、覆盖掉哪些；Zhou et al. (2025b) 把记忆池与推理联合管理；Yan et al. (2025b) 用 PPO 或 GRPO 训练一个记忆管理器，让它发出结构化的 `ADD`、`UPDATE`、`DELETE` 与 `NOOP` 操作，由下游问答表现给予奖励。Aghajohari et al. (2025) 则从架构层面攻击同一个问题，把推理过程重构，使上下文一开始就不会线性增长。

18.4.4 训练—推理上下文契约

有一个后果值得单独陈述，因为它造成的失败与其他任何失败都不像。

常见陷阱: 训练时与推理时的上下文策略必须完全一致

策略学到的行为是以它在训练中被教会去看的那种上下文为条件的。如果训练用的是最长 32k 的只追加历史，而部署时每 16k 就压缩一次，那么被部署的策略就是在以一类它从未遇到过的上下文分布为条件，而由此产生的退化在任何训练指标中都看不见。窗口大小、摘要提示、尾部长度以及工具结果的确切序列化方式，也都一样。这是节 3.4中聊天模板契约在智能体场景下的实例，而且在这里更糟，因为上下文策略通常实现在框架代码里而非模型配置中，于是没有人把它当作模型的一部分。请把你的上下文策略与检查点一起做版本管理。

表 18.3: 长时域下的上下文策略。「可训练」一列问的是：遗忘决策是否从终局奖励中获得梯度。其模式是，越便宜的策略越不可训练，而可训练的那些又制造出上文讨论的片段数可变问题。

策略	机制与失败模式	可训练	成本
只追加	全部保留；到窗口处失效，且在此之前就已退化（「上下文腐化」）	—	$O(K^2)$ 注意力
滑动窗口	保留最近 W 个轮次；智能体在自己记不得做过的事情上绕圈	否	最低
观测截断	保留动作、省略长观测；丢掉证据，保住意图	否	低
上下文压缩	策略在片段边界写摘要；摘要质量成为一项习得技能	是	额外生成；片段数可变
外部记忆	写入存储，以工具调用方式检索；检索失败是静默的	是，作为工具使用	检索延迟
习得的记忆操作	策略对记忆池发出显式的增/改/删	是	每轮次额外的决策

一个相关而更隐蔽的失败是重分词漂移。如果轨迹被存成聊天消息，并在训练时经由提示模板重新渲染，那么被训练的词元可能不同于被采样的词元，因为分词并不可逆，而压缩边界恰恰是那种「重新渲染会改变边界」的地方。Cheng et al. (2025) 点出了这一失败，并通过存储保留了确切采样动作词元的结构化步级记录来修复它。请存储词元，而不是文本。

18.4.5 经验、记忆即强化学习，与自演化智能体

把上一小节再推进一步，记忆就会比回合活得更久。Zhang et al. (2025d) 把上下文当作一份不断积累、可被精炼的战术手册，靠演化而非训练来推进，在完全不做梯度更新的情况下实现自我改进；Zhang et al. (2026c) 则完全冻结参数，用强化学习去优化情景记忆的检索，从而绕开灾难性遗忘。两者都把学习的所在地从权重移到了经验。这个想法的纲领版本——目前纲领的成分多于结果——是一条压缩流水线：原始轨迹被向前蒸馏为原子技能，共现的技能被组合起来，经过验证的组合体再向后折叠回参数中，从而腾出上下文预算让积累得以继续 (Wen, 2026)。截至 2026 年年中，还没有人展示过这个循环运转超过两三轮。

有一条实践观察属于此处。Anthropic (2025) 把长时间运行的智能体部署描述为一个负责搭好环境的初始化智能体，加上一个在每次会话中做增量推进的工作智能体，

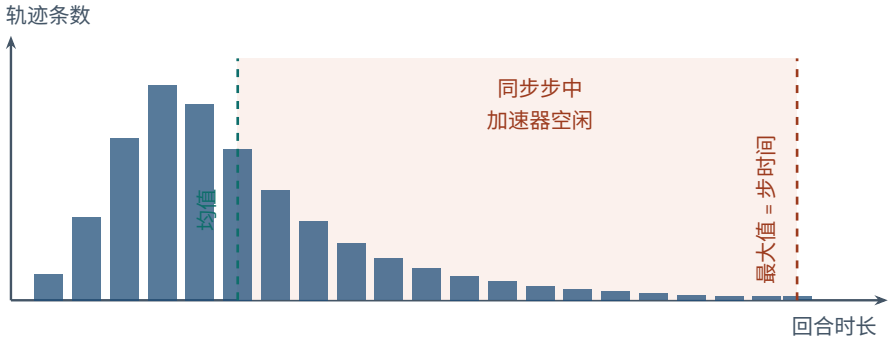


图 18.2: 智能体环境中回合时长的示意分布，按定性方式绘制，而非取自实测数据。质量集中在短回合上——那些策略很快放弃或很快解决的任务——并带一条长长的右尾，来自那些耗尽轮次预算或在慢环境上等待的回合。一个同步步付出的是最大值，用上的是均值；阴影区域就是浪费。节 18.5中的每一种方法，都是想把其中一部分收回来。

并报告持续性的瓶颈常常在于框架的健壮性而非模型能力：一次网络中断或一次上下文溢出就可能毁掉全部进展。训练中的对应情形一模一样。在把一次长时域失败归因于信用分配之前，先去核实你的采样轨迹中有多大比例是死于环境错误的——办法是去读轨迹，而不是看汇总指标。

18.5 采样成本与掉队者问题

第21章负责异步采样（节 21.6）与掉队者缓解（节 21.7）的系统层面处理，第22章 负责异步所必然带来的离策略修正。本节负责的是：这些选择对估计量做了什么。

18.5.1 长度分布是重尾的

轨迹时长是许多因素的乘积——所用轮次数、每轮次词元数、每次调用的环境延迟、失败后的重试——而这类乘积近似服从对数正态分布，因而右偏且带长尾。其后果画在图 18.2中：一个同步批次要等到最后一条轨迹结束才能组装起来，于是步时间取的是批次上的最大值，而有用工作量取的是均值。对一个哪怕只有中等离散度的对数正态分布，256 次抽样的期望最大值也会落在均值之上好几个标准差处，而且加速器时间被浪费掉的比例还会随批大小上升——与人们想要的规模化方向恰好相反。

历史注记: 部分采样早于智能体强化学习

把一次未完成的生成跨越权重更新携带下去，这一技术对智能体训练而言并不新鲜。Kimi Team (2025a) 为长思维链 RLVR 引入了部分采样，理由与上文完全相同：当回复长度横跨两个数量级时，同步批次会被少数几条长生成所主导，而把未完成的那些存入缓冲区、留待下一次迭代继续，便能收回大部分损失。智能体强化学习继承了这套机制，并把它从可选项变成了必需品，因为长度离散度更大，而且环境还会引入训练器无法控制的延迟。

18.5.2 未完成的轨迹该如何处理

假设当训练器想要更新时，某条轨迹已完成了它的 k 个轮次。对它有三种站得住脚的处理方式，而它们的偏差各不相同。

丢弃。 简单，但有偏。你丢弃的恰恰是那些长的轨迹，于是幸存的批次是一个 以「很快完成」为条件的样本。既然快速完成同时与「任务简单」和「早早失败」相关，丢弃就系统性地剔除了那些困难的成功——批次中信息量最大的采样轨迹。这项偏差是长度分布上的一种选择效应，并且会随图 18.2中的离散度一起增大。

截断，并把部分回报当作回报。 把该轨迹迄今累积的奖励赋给它，并当作回合已经结束。在纯终局奖励下，这会给每个未完成的回合赋 $r = 0$，从而教会策略「长回合会失败」——这是一种朝着过早作答的压力，很难与真正的效率偏好区分开，而且会与节 12.7的长度控制动力学相互叠加。

从值估计做自举。 赋值

$$\hat{G}_i = \sum_{j \le k_i} \gamma^{j-1} r_i^{(j)} + \gamma^{k_i} V_\psi(s_i^{(k_i)}), \tag{18.8}$$

即时序差分学习中标准的「截断回报加自举」。此时偏差变成了评论家的误差，而不再是选择效应——当评论家还算靠谱时这是更划算的交换，当它不靠谱时则更差。这是在长时域下想要一个评论家的第二个理由，且独立于节 18.4的压缩论证：没有评论家，就根本没有任何有原则的办法去利用一条部分轨迹。免评论家的替代品——从锚定状态组中估计 V_ψ——则继承了状态需要重现这一假设。

代码 18.2: 对一个混有已完成与未完成轨迹的批次做优势组装。`discard` 分支是最该埋点的一支：请记录被丢弃的比例，以及被丢弃与被保留轨迹的平均轮次数，因为两者之间差距的拉大，正是选择偏差开始起实质作用的信号。

```python
def assemble_advantages(traj, mode="bootstrap", value_fn=None, gamma=1.0):
    """traj: 采样轨迹列表，含 .rewards（逐轮次）、.done、.final_state。"""
    returns, keep = [], []
    for tr in traj:
        g = sum(gamma ** j * r for j, r in enumerate(tr.rewards))
        if tr.done:
            returns.append(g); keep.append(True)
        elif mode == "discard":
            returns.append(0.0); keep.append(False)          # 下面会被排除
        elif mode == "truncate":
            returns.append(g); keep.append(True)             # 隐含尾部 r=0
        elif mode == "bootstrap":
            k = len(tr.rewards)
            returns.append(g + gamma ** k * value_fn(tr.final_state))
            keep.append(True)

    r = torch.tensor(returns)[torch.tensor(keep)]            # 组统计量
    adv = (r - r.mean()) / (r.std() + 1e-6)                  # 只在保留成员上计算
    # alpha 修正：该组丢失了 len(traj) - sum(keep) 个成员
    return adv * (len(traj) / max(sum(keep), 1))
```

18.5.3 异步采样、有界陈旧度，以及估计量为此付出的代价

生产架构是异步且解耦的：生成与训练各有独立的资源池、并发运行，完成的轨迹流入一个由训练器抽干的缓冲区。约束陈旧度有两种在用的纪律。Zhang et al. (2025b) 采用队列排空：一个有界队列，队满时阻塞采样工作进程，每次更新消费掉整个队列。Gao et al. (2025c) 采用显式有界陈旧度：若训练器处于版本 n，则缓冲区中的每条轨迹都必须由不老于 $n - \alpha$ 的版本发起，他们报告 $\alpha = 1$ 是平衡点。两者都与Fu et al. (2025a) 形成对比，后者只在轨迹起始时约束陈旧度，因而允许长尾轨迹横跨若干个模型版本。

代码 18.3: 一个有界陈旧度的轨迹缓冲区。`by_prompt` 分组正是异步系统中组相对优势得以可能的原因：组是从某个提示恰好已落地的那些采样轨迹中机会性地组装出来的，因此实际组大小本身就是一个随机变量，必须被记录下来。

```python
class TrajectoryBuffer:
    def __init__(self, capacity, alpha, min_group):
        self.capacity, self.alpha, self.min_group = capacity, alpha, min_group
```

```
        self.by_prompt = defaultdict(list)

    def push(self, traj, version):
        self.by_prompt[traj.prompt_id].append((traj, version))

    def drain(self, n):
        """返回足够完整的组；丢弃早于 n - alpha 的一切。"""
        groups, dropped = [], 0
        for pid, items in list(self.by_prompt.items()):
            fresh = [(t, v) for (t, v) in items if v >= n - self.alpha]
            dropped += len(items) - len(fresh)
            if len(fresh) >= self.min_group:
                groups.append([t for (t, _) in fresh])
                del self.by_prompt[pid]
            else:
                self.by_prompt[pid] = fresh          # 等待同组的其他成员
        log("stale_dropped", dropped, "groups", len(groups))
        return groups
```

核心思想：异步把一个挂钟时间问题转化成了一个估计量问题

异步采样没有任何一处是免费的。一个批次中的各条轨迹由好几个不同的策略生成，因此梯度的离策略程度在批内各不相同；某个提示的组是由碰巧完成的那些采样轨迹拼出来的，于是实际的 G 会波动，基线的质量也随之波动；而一条被复用的部分轨迹，其前缀来自一个策略、后缀来自另一个策略，这种混合是重要性比值所无法描述的。第22章提供了相应的修正。要记住的一点是：系统决策与统计决策其实是同一个决策，一个只管其中一头而不管另一头的团队，造出来的会是一个跑得很快却学不到东西的训练器。

关于回放缓冲区的证据尚未定论。Wang et al. (2025i) 报告，陈旧五步或十步的采样轨迹 一贯地降低表现，因而主张严格在线的强化学习；Li et al. (2025d) 则反过来，为困难任务预先收集成功轨迹以弥补在线成功过于稀疏，并报告了增益，同时还从解耦中获得了 1.9× 的训练吞吐。可能的调和是——均匀的陈旧有害，而对稀有成功的选择性复用有益——但这是关于缓冲区内容的论断而非关于陈旧度的论断，而且它尚未被直接检验。

18.6 方差与稳定性

多轮次强化学习在经验上不如单轮次 RLVR 稳定。有三种机制可以区分开来，每一种都指向一种不同的干预。

复合。 策略的一个微小变化会改变第 1 轮次的动作，从而改变第 1 轮次的观测，从而改变第 2 轮次的上下文。若逐轮次动作分布移动了 ϵ 的总变差，则轨迹分布最多可移动 $K\epsilon$：一个按词元校准的信赖域，放到轨迹层面就松了 K 倍。这是支持轮次感知信赖域的最干净的论证，2026 年的若干提案也正是出于这个理由把重要性采样与截断移到轮次边界上；不过，轮次结构化优化中记录最完备的那些实例，作用的是优势而不是比值 (Zhou et al., 2026; Modecrua et al., 2026)。

分布外条件化。 工具反馈按其构造方式就不是语言模型自己会生成的文本。以它为条件会产生低概率词元；这些词元在轮次之间复合；结果就是 Xue et al. (2025) 追溯到空转轮次的那种梯度范数爆炸。

环境非确定性。 从同一个状态出发的同一个动作可能返回不同的观测——一次不稳定的测试、一个变了的网页、一个被限流的 API。这是策略无法降低的方差，而它会抬高(18.4)中的组标准差, 从而缩小组内的每一个优势。Shang et al. (2025) 用 GRPO-RoC（正确即重采样）直接应对这一点，即过量采样并保留较干净的轨迹，并报告一个 14B 模型在 510 个强化学习步内于 AIME24 上达到 80.6%。

有三种具名的坍缩模式已被记录在案,应当分开诊断。回声陷阱 (Wang et al., 2025i) 是对智能体自身生成的推理模式的过拟合；其三联征是奖励平台期、组内奖励方差与词元熵同时下降，以及一次往往不可逆的梯度范数尖峰。模板坍缩 (Wang et al., 2026f) 表明熵是不充分的诊断指标：模型可以在保持健康的输入内熵的同时产生与输入无关的输出。他们的分解才是有用的贡献——总多样性等于输入内熵加上跨输入互信息，而熵奖励只针对第一项，这正是它在此处失效的原因。惰性似然位移 (Deng et al., 2025) 则专属于工具融合的 GRPO：正确回复的似然会随奖励上升而衰减，低置信度的回复随后又抬高梯度。

两点评论。更大的组有最干净的理论支持和最糟糕的成本画像，这正是该领域把力气花在估计量设计上的原因；如果生成有朝一日变得足够便宜，(18.2)预测其中大半机关都将不再必要。而选择性反复成为「有效方法」的共同特征，这并非巧合 (Yu et al.,

表 18.4: 多轮次强化学习的各种稳定性干预，附上各自针对的机制与证据等级。「已复现」指有若干独立团队；「单篇」指只有一篇。值得注意的模式是，复现良好的干预全都是选择性的——它们作用于被识别出的特定轮次、词元或任务——而那些均匀施加的干预则仍有争议。

干预	机制	证据
空转轮次过滤	移除分布外条件化带来的高幅值梯度，同时保持基线无偏	精神上已复现；有一项细致研究
时域课程	限制早期探索，使信号在时域增长之前更干净	两个团队一致（节 18.7）
基于方差的任务过滤	保留结果不确定性高、i.e., 非退化的组所对应的任务	已复现（与节 12.4同一想法）
提高上截断、不用 KL	保住探索；自单轮次实践引入	单轮次已复现，可迁移
逐轮次 KL 或轮次级截断	让信赖域匹配漂移复合所在的单位	单篇／提案阶段
选择性熵控制	只对高熵或似然下降的词元做截断或正则	各有单篇，结论正在收敛
更大的组	直接攻击(18.2)；唯一有干净理论的干预	很少负担得起
熵奖励	均匀施加；只针对输入内多样性	失效，已被反复报告

2025d)：在长时域下，批次中有信息量的部分很小，因而均匀施加的干预会把全部预算花在那些本来就没问题的轮次上。

工程实践: 长时域训练中该记录些什么

在标准 RLVR 仪表盘之外，还要记录：轮次数的分布（而不是均值）；因触及轮次上限、因环境错误、因验证器判定而终止的轨迹各占多少；退化组的比例；每条轨迹的压缩次数及其在组内的方差；空转或格式错误轮次的比例；异步组装之后实际得到的组大小；以及该批次的平均策略版本陈旧度。这里的每一项，都曾在某个已发表的系统中充当过一次静默训练失败的直接原因，而它们没有一项能在奖励曲线上看出来。

18.7 时域课程

如果信用分配问题的难度随 K 增长，一个显而易见的应对就是在训练过程中逐步增大 K。AgentGym-RL 中的 ScalingInter-RL(Xi et al., 2025) 是已发表实例中最干净的一个：训练被切成 N 个阶段，交互预算单调递增 $h_1 < h_2 < \cdots < h_N$；在他们的配置中是三个各 80 次迭代的阶段，预算依次为 $\{8, 12, 15\}$ 轮次。

那个经验发现是反直觉的，而这正是值得讲授它的理由。从一开始就给出很大的交互预算，起初会改善表现，随后却会坍缩；而限制早期探索反而给出更干净的学习信号，支撑起后期稳定的长时域学习。作者们的总结——早期阶段的过度探索未必是个好选择——得到了Wang et al. (2025i) 的独立佐证，后者发现中等的交互预算既胜过很小的、也胜过很大的预算。

其机制大概还是(18.2)：当 K 较小时，可训练词元数 N 也小，广播式优势的信噪比尚可，策略于是学到一个称职的短时域控制器；随后延长时域，就相当于给一个逐轮次行为已经合理的策略再加上一些轮次。这是一个论证，不是一次测量，而证据的边界在于：只有两个团队，都在结构化环境上、时域在几十而非几百的量级，而且没有任何已发表的消融把课程本身与所消耗的总交互预算分离开来。时域课程在软件工程的 $K = 200$ 上是否有帮助，尚未被检验。

有一项证据更充分的观察，关乎的是初始化而非课程。Tongyi DeepResearch Team (2025) 报告，一个智能体式持续预训练的中期训练阶段——先在 32k、再在 128k 上下文的长智能体轨迹上做下一词元预测——会装入一种归纳偏置，使后续的强化学习效率高得多；Yu et al. (2025d) 独立报告，真实的端到端工具使用轨迹给出的监督微调初始化，远强于拼接出来的合成轨迹。长时域强化学习对自己从哪里出发异常敏感，因为它负担不起自行发现称职行为所需的探索量（节 1.4）。

18.8 已知的与未知的

本章以一份明确的证据分级收尾，因为这支文献还很年轻，而它的自信超出了它被复现的程度。

已复现。 稀疏是长时域下的决定性约束。本章中的每一个系统，都把它的巧思花在如何从同一个终局比特里榨出更多信号上；而报告增益最大的那些，是增加信号（轮次奖励、世界模型损失、特权评论家）或去除噪声（空转轮次过滤、基于方差的任务筛选）的做法，而不是精修优化器的做法。部分轨迹的处理很要紧。多个独立系统——Gao et al. (2025c)、Fu et al. (2025a)、Zhang et al. (2025b)、Li et al. (2025d)——都报告，带显式陈旧度纪律的异步采样带来了很大的吞吐增益，而所报告的倍数（端到端 2× 到 4.5×）超过本章中任何一项算法效应。超过某个时域之后，上下文压缩是必需的。一旦 $K\times$ 每轮次词元数超过窗口，这就近乎同义反复，而每一个已发表的长时域系统都做了某种

形式的压缩。选择性干预胜过均匀干预，这一点在四种独立机制上都成立（空转轮次、高熵词元、似然下降的词元、高方差任务）。

仍有争议。 在等算力条件下，轮次级信用分配是否胜过均匀广播。这与第13章就过程监督所提的是同一个问题，而它在此处很可能有不同的答案，理由有三。时域更长，所以(18.2)糟糕得多，奖品也更大。中间信号来自环境而非来自习得模型，所以它既不像蒙特卡洛步级标签那样带噪，也不像过程奖励模型那样可被作弊。而一个轮次是一个真正的决策边界——策略作出承诺、世界给出回应——而「一个推理步」只是一种约定。反过来说：Feng et al. (2025b) 的增益出现在状态会重现的环境上，而在状态不重现之处该方法根本不可用；Wang et al. (2025j) 的增益出现在带手工设计信息增益奖励的搜索任务上；而在最困难基准上拿到最强数字的那些生产系统——DeepSWE-Preview(Luo et al., 2025a)、SWE-Master (Song et al., 2026)——用的都是均匀广播。轮次级信用分配在其结构性假设成立之处显然更好，在假设不成立之处则未获证实；而且还没有人在 $K \gtrsim 100$ 上，用节 12.12所要求的方法学做过等算力对比。

评论家是否会回归。压缩论证（节 18.4）与部分轨迹论证（节 18.5）都指向同一个方向，而两者目前分别只有一篇博客文章和一项理论观察作为支撑。

回放是否有用。节 18.5已经摆出了Wang et al. (2025i) 与Li et al. (2025d) 之间直接的矛盾。

单篇证据或尚处早期。 记忆即强化学习——训练那个「写」的决策本身——只依托于少数几篇 2025–2026 年的论文 (Yu et al., 2025a; Zhou et al., 2025b; Yan et al., 2025b; Zhang et al., 2026c)，它们的任务设置各不相同，且没有正面对比。自演化与经验积累式智能体 (Zhang et al., 2025d; Gao et al., 2025a) 尚处于「框架比结果更有价值」的阶段。在观测词元上做世界建模损失 (ECHO, 2026; PaW, 2026) 只有两篇论文之久，是这一领域中最有意思的想法，也应当被相应对待：有前景、机制上说得通、且未被复现。

本章小结

- 长时域智能体强化学习由四项彼此耦合的约束所定义——奖励稀疏、上下文长度、采样成本与方差——它们全都是时域 K 从个位数增长到数百这一变化的后果。以「每比特奖励所耗的上下文词元步」衡量，一个百轮次的软件工程回合大约比一条单轮次数学采样轨迹昂贵三个数量级。

- 一个终局的二值奖励每条轨迹至多提供一个比特，却要分配给 10^4–10^5 个决策。均匀广播估计量的信噪比按 $m\sqrt{G/N}$ 标度，这同时解释了纯结果式 RLVR 为何对单轮次推理有效，以及它为何在一百轮次上勉强及格。
- 退化组在 RLVR 中只是个小麻烦，在长时域下却成了常态，因为困难智能体任务上的成功率很低。动作采样、$B/(B-M)$ 重标定与难度课程是承重构件，而非可选项。
- 在智能体设置中，塑形奖励比在单轮次 RLVR 中更危险，因为一个百轮次策略在每个回合里有一百次机会去发现满足该规范的廉价办法。已发表的负面结果与正面结果同样有教益。
- 轮次级信用分配方法构成一个谱系，其排序取决于它们对环境作了什么假设，而不取决于它们带多少偏差。状态分叉需要快照与恢复；锚定状态分组需要可哈希且会重现的状态；空转轮次掩蔽只需要一个廉价的逐轮次谓词，这正是它应当被最先采纳的原因。因此可快照性是一项决定哪些算法可行的系统属性，而它在环境被建造时就已固定。
- 在均匀广播之下，一条轨迹中的一个坏轮次会毒害其余九十九个好轮次。把责任定位下来——把空转轮次掩蔽出损失、同时保留在基线中——既便宜又无偏，且据报告能带来很大增益。
- 智能体如何遗忘是其策略的一部分。朴素截断会使该过程相对于它自身的历史成为非马尔可夫的，从而产生看上去像能力缺陷的绕圈行为。压缩是一个有奖励后果的动作，应当被训练；它同时也破坏了组相对估计量所假定的可比性，而这正是目前主张重新引入评论家的最有力论据。
- 训练时与推理时的上下文策略必须完全一致。二者失配在任何训练指标中都看不见，而且它通常实现在没有人会与检查点一起做版本管理的框架代码里。
- 回合时长是重尾的，因此一个同步步付出的是最大值，用上的是均值。丢弃未完成的轨迹会带来偏向短回合的选择偏差；截断它们会教出过早作答；从值估计做自举则用评论家的误差换掉这两者，也是唯一有原则性理由的选项。
- 异步把一个挂钟时间问题转化为一个估计量问题：离策略的批次、波动的实际组大小，以及前缀与后缀由不同策略生成的轨迹。

延伸阅读

先读Feng et al. (2025b)。它篇幅很短，想法一段话就能讲完，消融实验对「哪一层分组起了什么作用」很诚实，而它的局限——锚定状态分组需要会重现的状态——对长时域信用分配的教益，超过大多数正面结果。请紧挨着第13章中的节 13.4之前或之后读它，因为二者是同一个估计量，只是对「重新进入一个状态有多贵」作了不同的假设。

关于不稳定性，Xue et al. (2025) 是目前可得的、关于「为什么多轮次强化学习会爆炸而单轮次不会」最尖锐的因果说明，而它的修补只有两行代码，且有干净的统计理由。请把它与Wang et al. (2026f) 配对阅读，后者把多样性分解为输入内熵加跨输入互信息，解释了为何你从第12章继承来的那些熵诊断在此处不够用；再配上Deng et al. (2025)，它讲的是一种专属于工具融合的失败模式，否则你会把它误诊为普通的熵坍缩。

关于系统层面的后果，Gao et al. (2025c) 是有界陈旧度与部分轨迹复用的参考文献，而它与Fu et al. (2025a) 的明确对比，是对「有界陈旧度」可以指什么的最清晰陈述。Li et al. (2025d) 值得与之对照阅读，因为它对回放给出了相反的结论；两者之间的分歧，再加上Wang et al. (2025i)，是一次很好的「阅读尚未定论的文献」的练习。第21章会正式讨论这套基础设施。

关于上下文与记忆，Yu et al. (2025a) 是把记忆管理表述为强化学习问题的最干净版本，Yan et al. (2025b) 则对动作空间讲得最明确。Aghajohari et al. (2025) 是架构层面的替代方案，值得一读，因为它提醒我们：一个训练期的问题有时可以被消解，而不必被求解。最后，ECHO (2026) 与PaW (2026) 应当合在一起读，并且要带着怀疑读：两个独立团队在同一年得出同一结论令人鼓舞，但这个结论修正的是一项已被两年实践确立下来的约定（节 16.4），而这种量级的修正通常需要的不止两篇论文。

习题

18.1（稀疏与组大小。） 利用(18.2)，在 $p = 1/2$ 下计算 (N, m) 取 $\{(8\times10^3, 50), (5\times10^4, 50), (5\times10^4, 5)\}$ 时，达到单位信噪比所需的组大小；再在 $p = 0.05$ 下重做一遍，后者是困难软件工程基准上一个现实的成功率。请解释为何 p 依赖性的作用方向与退化组概率(12.4) 相反，以及这对「固定采样预算该花在哪里」意味着什么。

18.2（轮次粒度上的折扣。） 对纯终局奖励且 $K = 100$，列表给出 $\gamma \in \{1, 0.99, 0.95\}$ 下轮次 $k \in \{1, 25, 50, 75, 100\}$ 所获得的信用 γ^{K-k}。然后假设轮次长度服从对数正态

分布，中位数为 300 个词元，最短与最长轮次之间相差 100 倍。计算在逐词元折扣 0.999 之下，分配给第一个轮次的总信用在组内的方差，并用一句话说明为什么这是一个方差问题而不是偏差问题。

18.3（**压缩破坏了组。**）考虑同一提示的 $G = 8$ 条采样轨迹构成的一个组，其中压缩次数为 $(1, 1, 2, 2, 2, 3, 3, 5)$，且可训练词元数与这些数字成正比。在全局词元均值归一化（表 12.2）之下，计算每条采样轨迹对该次更新的梯度权重所贡献的比例。然后提出并论证一种能让该组重新变得可比的归一化方式，并指出它的代价何在。

18.4（**快照预算。**）你有一个支持写时复制文件系统快照的环境，每份快照 200 MB、每次恢复 1.5 秒；采样预算为 $G = 8$ 条 $K = 60$ 轮次的轨迹；每个并发任务有 40 GB 快照存储。请确定在存储限制内、可用于算法 18.1的最大 (M, B)，计算由此带来的生成成本与挂钟时间的增加，并依据节 13.4的分辨率论证判断：由此得到的估计量能否检测出一个使成功概率改变 0.2 的轮次。

18.5（**诊断一次坍缩。**）一次百轮次的智能体训练显示：奖励在第 600 步进入平台期；平均轮次数从 70 降到 22；策略熵持平；以给出最终答案收尾的轨迹比例上升；留出集上的 pass@1 下降。请从节 18.6中说出最可能的失败模式，给出一项能把它与两个最接近的备选区分开来的额外诊断指标，并给出你会最先施加的干预。

18.6 ⋆（**无重现状态下的轮次级信用。**）锚定状态分组要求状态精确重现。请提出一种在状态从不重现时仍能恢复轮次级基线的估计量——例如按状态上某个习得或手工设计的等价关系来分组、按轮次下标分组，或使用一个只被训练来对轨迹内状态排序的值函数。对你的提案，（a）精确陈述要使定理 2.2适用所需的独立性假设；（b）指出该假设被违反时的失败模式；（c）设计一个能把真正的信用分配增益与方差缩减的假象区分开来的实验。

18.7 ⋆（**端到端训练压缩是最优的吗?**）一个压缩轮次得到与其他任何轮次相同的广播优势，因此它收到的梯度正比于该回合的结果。请为下述主张辩护或反驳：由于一次压缩对结果的影响是经由其后每一个轮次中介的，它在均匀广播之下的优势被估计得比普通轮次 更好，而不是更差。请用节 18.2.1的决定性决策模型把该主张形式化，然后构造一个使之不成立的具体任务。

18.8 †（**搭建四约束仪表盘。**）取一个带程序化验证的开放多轮次环境和一个 3–8B 模型。用均匀广播训练 300 步，记录节 18.6 中该记录些什么方框里的每一个量。然后在等算力条件下做三组消融：（a）开启空转轮次掩蔽；（b）一个从 8 升到完整预

算的时域课程；(c) 滑动窗口上下文策略对比上下文压缩。请报告每组消融实际缓解了四项约束中的哪一项，以仪表盘而非奖励来度量，并说明你复现了节 18.8 中已复现论断里的哪几条。

18.9 † **（测量掉队者。）**给一个智能体采样工作进程做埋点，记录每条轨迹的时长、轮次数以及在环境上等待的时间。至少收集 2,000 条轨迹。拟合一个对数正态分布，并报告 $G \in \{8, 64, 256\}$ 时 G 次抽样的期望最大值与均值之比。量化丢弃带来的选择偏差：一个设在第 80 百分位的截止时限，会移除多大比例的轨迹，又会移除多大比例的成功轨迹？然后实现 算法 18.3，取 $\alpha \in \{0, 1, 4\}$，在等挂钟时间下同时报告吞吐与最终奖励——尤其要报告：使吞吐最大的那个 α 是否就是使奖励最大的那个。

算法 18.1 基于状态分叉的轮次级优势估计。第 4 行是唯一一处对环境的要求超出普通采样的步骤，也是决定该方法究竟可不可用的那一步。最终的优势把一个回合级项与一个轮次级项组合起来，沿用了Feng et al. (2025b) 的分层构造；取 $\omega = 0$ 即退回均匀广播 (18.4)。

输入： 策略 π_θ，可快照环境 $\mathcal{E}$，提示 x，组大小 G，分叉因子 B，分叉点选择器 C，混合权重 ω

带检查点的采样
for $i = 1$ **to** G **do**
  在 $\mathcal{E}$ 的一个全新实例中采样出 $\tau_i \sim \pi_\theta$，并记录 $(h_i^{(k)}, a_i^{(k)}, o_i^{(k)})$
  在每个 $k \in C(\tau_i)$ 轮次之后对 $\mathcal{E}$ 做快照 ▷ 写时复制；$O(|C|)$ 份镜像
  $r_i \leftarrow$ 对 τ_i 运行终局验证器
end for
回合级优势
$A_i^E \leftarrow (r_i - \mathrm{mean}_j r_j)/\mathrm{std}_j r_j$
轮次级优势
for $i = 1$ **to** G, **each** $k \in C(\tau_i)$ **do**
  把 B 个隔离的 $\mathcal{E}$ 实例恢复到快照 $s_i^{(k)}$
  采样 $\tau^{(1)}, \dots, \tau^{(B)} \sim \pi_\theta(\cdot \mid h_i^{(k)}, s_i^{(k)})$
  $\Phi(s_i^{(k)}) \leftarrow \frac{1}{B} \sum_b r(\tau^{(b)})$
end for
for $i = 1$ **to** G, **each** turn k **do**
  $A_i^{S,k} \leftarrow \Phi(s_i^{(k)}) - \Phi(s_i^{(k-1)})$ ▷ 在该段内为常数；在分叉点之外取 0
end for
组装
for each token t of turn k in τ_i **do**
  $A_{i,t} \leftarrow m_{i,t}(A_i^E + \omega A_i^{S,k})$
end for
return $\{A_{i,t}\}$

算法 18.2 压缩感知的采样。这个循环就是标准的智能体采样，只多了一件事：当上下文超出预算时，策略自身发出一份摘要，该摘要被记录为一个可训练片段，并用于重新初始化上下文。返回的轨迹包含若干个可训练片段，其数量与长度在同一提示的不同采样轨迹之间各不相同——正是这一事实使组相对优势估计复杂化。

输入： 策略 π_θ，环境 $\mathcal{E}$，任务 x，上下文预算 $L_{\max}$，预留量 L_{res}，尾部长度 W

$h \leftarrow x$；　$\mathcal{M} \leftarrow \emptyset$；　片段列表 $\leftarrow [\,]$；　$k \leftarrow 0$
while 未完成 **and** $k < K_{\max}$ **do**
　if $|h| > L_{\max} - L_{\text{res}}$ **then**
　　$\mathcal{M} \leftarrow \pi_\theta(\cdot \mid h, \texttt{compact_instruction})$　▷ 一个策略动作；其词元可训练
　　把 $(h, \mathcal{M})$ 追加到片段列表　▷ 片段在此收尾
　　$h \leftarrow x \oplus \mathcal{M} \oplus$ last W turns
　end if
　$k \leftarrow k + 1$；　$a^{(k)} \sim \pi_\theta(\cdot \mid h)$
　$o^{(k)} \leftarrow \mathcal{E}(a^{(k)})$
　$h \leftarrow h \oplus a^{(k)} \oplus o^{(k)}$　▷ 观测词元从损失中掩蔽
end while
$r \leftarrow$ 终局验证器
return 片段列表, r, k　▷ 一个奖励，若干个片段

算法 18.3 带有界陈旧度与部分轨迹复用的异步采样，结构沿用Gao et al. (2025c)。轮次边界是检查陈旧度的天然位置，因为它是唯一一个可以在新权重下重新编码轨迹上下文、而又不必丢弃已生成词元的时刻。

输入： 陈旧度上界 α，容量为 C 的缓冲区 $\mathcal{B}$，E 个环境工作进程，轮次上限 $K_{\max}$

工作进程 e，异步执行
while true **do**
  拉取权重 θ_v；采样任务 x；重置 $\mathcal{E}$；$k \leftarrow 0$
  while 未完成 **and** $k < K_{\max}$ **do**
    $a^{(k)} \sim \pi_{\theta_v}$; $o^{(k)} \leftarrow \mathcal{E}(a^{(k)})$; $k \leftarrow k+1$
    if 训练器版本 $n > v + \alpha$ **then** ▷ 仅在轮次边界处检查
      刷新 $\theta_v \leftarrow \theta_n$；为 h 重算 KV 缓存；$v \leftarrow n$ ▷ 继续，而不是重启
    end if
  end while
  把 (τ, v, done) 推入 $\mathcal{B}$；若 $|\mathcal{B}| = C$ 则阻塞
end while
训练器
while 训练中 **do**
  抽干 $\mathcal{B}$；丢弃版本 $< n - \alpha$ 的条目
  按提示分组；组装优势（代码 18.2）
  对版本 $v < n$ 的条目施加离策略修正 ▷ 第22章
  更新 $\theta_n \to \theta_{n+1}$；广播权重
end while

第 19 章

环境、沙箱与任务供给

核心思想：在智能体后训练中，环境就是实验本身

一旦一次采样触及了真实系统，策略优化的就不再是你写下的那个奖励，而是你的基础设施所发出的那个奖励。每一次超时、重试、限流、缓存与非确定性响应，无论你是否有意为之，都是那个奖励中的一项。因此，一次智能体训练的决定性约束是吞吐、确定性、隔离，以及「难度仍能产生梯度」的任务的供给——决定你能训练什么的是这四者，而不是在 GRPO 与 PPO 之间的选择。本章把环境当作工程对象来处理，并把任务池当作它本来就是的那种可耗尽资源。

第16章确立了智能体决策过程，第17章确立了工具奖励的设计，第18章讲清了百轮次时域对估计量做了什么。这三章都假定了某个它们并未去建造的东西：一个能重置、能步进、能评分，并且能这样做上几百万次而不说谎的程序。本章来建造它。这些材料并不光鲜，而智能体训练恰恰是在这里真正翻车的。

19.1 环境是奖励函数的一部分

先看一个案例：它要花掉一周的算力，而在日志里看起来像是一个算法问题。

例 19.1 (一个抖动的沙箱教会策略写出更差的代码)**.** 一次工具融合的训练给策略配了一个挂钟时限十秒的 Python 沙箱，其背后是一个共享的工作进程池。在负载之下，该池有时来不及及时调度作业，而框架把由此产生的超时当作一次普通的执行失败报告给策略，当作奖励 0 报告给训练器。这种调度失误的发生率在不同动作上并不一致：一个跑完项目全部测试套件的解答会占用一个工作进程六秒，在负载下大约有 15% 的时间会撞上时限；而一个只跑一个冒烟测试的解答不到一秒就结束，撞上时限的概率约为 1%。

设 $p(a)$ 为动作 a 确实正确的概率，$q(a)$ 为框架虚假地判其失败的概率。观测到的奖励的均值是 $p(a)(1-q(a))$。一个 $p=0.90$、$q=0.15$ 的周全动作得分 0.765；一个 $p=0.80$、$q=0.01$ 的走捷径动作得分 0.792。组相对优化会找到那条捷径，而训练曲线会显示奖励在上升。奖励规范里可从来没有提过超时这回事。

这个例子可以推广成一个值得记住的陈述。

推导: 环境失败是一项与动作有多好成正比的隐式动作成本

把实际得到的奖励写作 $\tilde{r} = r \cdot \mathbb{1}[\text{no infrastructure failure}]$，其中失败概率 $q(a)$ 可能依赖于动作。于是

$$\mathbb{E}[\tilde{r} \mid a] = p(a) - \underbrace{q(a)\,p(a)}_{\text{implicit cost}}, \tag{19.1}$$

也就是说，策略实际被训练的是「预期目标减去一个设计者从未写下的塑形项」。该项有两条性质要紧。它对 p 是乘性的，因此对好动作的惩罚重于对坏动作的惩罚：一个从来不奏效的动作永远不会被虚假判负，因为它本来就要得 0 分。而它的梯度 $\nabla(qp) = p\nabla q + q\nabla p$ 的第一项，把策略纯粹推向对基础设施友好的行为，完全不参考任务是否成功。当 q 在动作空间上的变化比 p 更剧烈时——这是常态，因为 q 是由资源消耗驱动的——这一项就会占主导。

同样的代数也覆盖了基础设施的其余部分。一项对失败调用重新执行的重试策略，把 q 变成了 q^k，大体上是有帮助的；但它同时也在悄悄奖励那些重试代价低廉的动作，而且如果重试以重复观测的形式对策略可见，它就成了策略可以学会去触发的一个动作。一个以调用参数为键的响应缓存会让重复查询变得免费，这是一项策略会通过发出近乎重复的调用来利用的逐轮次成本；更糟的是，它会让一个组相对批次的各成员相互关联，因为组成员会问出彼此重叠的问题，而一次缓存命中会给它们完全相同的观测。一个把 `429` 返回给最后一个提出请求的工作进程的限流器，则把环境争用转化成了逐轨迹的奖励噪声。这些都不是缺陷。它们是在基础设施层作出、却在目标函数中生效的设计决策。

常见陷阱: 把基础设施失败当作任务失败来上报

(19.1)最常见的单一实例，就是把「智能体的动作是错的」与「我们没能评估智能体的动作」压成了同一个奖励值。它们是不同的事件，必须区别处理：真正的失败是一个训练信号，而基础设施失败是一次缺失的观测。正确的处理是把该轨

迹从损失中剔除，同时保留——或显式地重新估计——组统计量，做法与Xue et al. (2025) 对空转轮次的处理完全一致；并且在剔除率超过阈值时呼叫值班人员。错误的处理，也就是每个框架里的默认行为，是任由那个零流下去。

19.2 环境类别剖析

各类环境沿四个坐标轴而不同，而这四者几乎能预测训练会走成什么样：奖励实际度量的是什么、状态转移的确定性有多高、延迟分布的形状，以及「不做任务就拿到奖励」的最廉价途径。表 19.1给出汇总；正文补充其机制。

代码解释器与沙箱化 *shell* 是表现最好的一类。奖励是基于结果的，且在环境之外计算（返回的答案对上了吗?），状态转移在进程自身的非确定性之内是确定的，而延迟呈双峰——算术题不到一秒，任何进入死循环的东西则卡在超时上。终端环境属于同一类，只是状态是持久的、时域长得多 (Merrill et al., 2026)。标准的作弊手法是让框架而非任务去产生答案：打印出预期的字符串、以零状态退出，或者直接写入评分器所读取的那个文件。

带测试套件的软件工程仓库是吸收了最多工程投入的一类 (Jimenez et al., 2023; Pan et al., 2024; Jain et al., 2025b; Yang et al., 2025)。奖励是一次状态检查——针对被修改后的仓库运行测试——这远强于答案匹配，相应地也昂贵得多：每评一次分就要拉镜像、装依赖、跑一遍完整测试。如果镜像被按摘要钉死、网络被关闭，确定性就不错；否则就很差，因为一个能连上互联网的包解析器会使环境成为日期的函数。它的可作弊面是所有类别中最宽的，而节 19.5基本上就是在讲它。

网页分裂成两个共用一个名字的环境。实时浏览 (Zheng et al., 2025d) 的状态转移不可复现，延迟横跨三个数量级，而奖励会悄悄取决于某天下午搜索索引里恰好有什么。快照式或模拟式网页 (Koh et al., 2024; Sun et al., 2025) 以分布保真度为代价换回了确定性与低延迟。两者的作弊手法相同：把答案找出来而不是推出来——在一项研究任务上这是正当策略，而在一项因基准公开而使答案就摆在开放网络上的任务上，这就是致命的。

操作系统与图形界面环境 (Xie et al., 2024; Lai et al., 2025a; Muryn et al., 2026) 把一台虚拟机和一张截图放进了循环里。延迟由渲染和界面的稳定时间主导，确定性受动画与时钟所限，而奖励几乎总是对文件系统或应用数据库的一次终态检查。其特征

性的作弊是通过一条绕开待训练界面的路径抵达目标状态——去编辑配置文件，而不是使用设置对话框。

数据库与结构化 *API* 廉价、近乎确定，而且易于快照，这正是合成工具环境集中于此的原因 (Cai et al., 2025; Tu et al., 2026; Zhou et al., 2026)。奖励是与一个标准终态的精确状态相等，既毫无歧义，也因此很脆：任何一个任务设计者忘了去约束的状态变量，都是白送的奖励。

游戏与谜题生成器 (Hopkins et al., 2025; Li et al., 2025c) 是唯一一类任务供给真正无上限的环境，因为生成器本身就是任务。在给定种子下确定性是完美的，延迟可以忽略，而奖励就是游戏自己的分数。其作弊手法是发现退化策略——这在一个游戏里是正当结果，在一个训练环境里则是用一项无处可迁移的技能污染了策略。

模拟组织与多用户场景 (Xu et al., 2024b; Barres et al., 2025; Meta, 2025) 引入了一个模拟的对手方，通常是另一个语言模型。这是确定性最差的一类，也是奖励最容易被破坏的一类，因为其作弊手法是操纵那个模拟用户——而与真实用户不同，模拟用户没有梯度，也很好骗。Cai et al. (2025) 发现，必须先检测并掩蔽模拟用户的错误行为，他们的采样轨迹才能用。

19.3 隔离，以及沙箱池的算术

19.3.1 威胁模型和通常的那个不一样

一个生产沙箱防御的是一个想要逃逸的攻击者。一个训练沙箱防御的则是一个毫无意图的策略——它会在数百万次采样中，把它的分布赋予不可忽略质量的每一个语法合法的程序都执行一遍。相关的差别在于规模与盲目搜索：策略并不想越狱，但它正在动作空间上做一场极限意义下的穷举搜索，而任何一条能由 `rm`、`curl` 或一个漏网的 `sudo` 抵达的奖励通道，终将被找到。由此有三条后果。逃逸必须靠结构而非靠规章来阻止，因为这里没有可以诉诸的规章。出网必须默认拒绝，因为在几乎任何任务上，通往奖励的最快路径都是把答案下载下来。而一次成功逃逸的影响范围是整次训练，而不是一个用户的数据，因为一个学会破坏奖励服务的策略会破坏所有并发的采样。

重置是把训练环境与部署环境区分开来的那条性质，也是写时复制真正值回成本的地方。一层覆盖文件系统让 G 个组成员共用一份不可变的基础镜像，只丢弃各自的写入，于是重置成了一次 unlink 而不是一次重建。若环境是数据库，等价物就是一份每次采样克隆一遍的模板；若环境是虚拟机，那就是一份内存快照。任何重置起来需要重跑当初搭建流程的环境，都会把大部分挂钟时间花在搭建上：这正是Zhipu AI and

表 19.1: 按每次采样成本大致排序的环境类别。「确定性」指从同一次重置出发重新执行同一动作序列是否得到相同的观测；「作弊手法」给出文献中观察到的、不做任务就拿到奖励的最廉价途径。延迟一列给出的是单个环境步的数量级，而非实测数据。

类别	奖励语义	确定性	单步延迟	标准作弊手法
代码解释器	对返回答案做结果匹配	高	10^{-1}–10^{1} s	打印预期输出；以零状态退出
终端／shell	容器终态加测试	高	10^{-1}–10^{1} s	直接写出评分器的输出文件
软件工程仓库	对被修改仓库跑留出测试套件	钉死则高	10^{1}–10^{2} s	改测试或跳过测试；抓取上游补丁
网页，快照式	对冻结语料做答案匹配	高	10^{-2}–10^{0} s	背下语料；利用索引怪癖
网页，实时	对开放网络做答案匹配	无	10^{0}–10^{2} s	检索到泄露的基准答案
操作系统／图形界面	应用或文件系统终态	中	10^{0}–10^{1} s	绕开界面抵达目标状态
数据库／API	与标准状态精确相等	高	10^{-2}–10^{0} s	设置未被约束的状态变量
游戏／生成器	游戏自己的分数	给定种子则完美	10^{-3}–10^{-1} s	无迁移价值的退化策略
模拟组织	对话后的评分细则或状态检查	低	10^{0}–10^{1} s	操纵模拟的对手方

Z.ai (2025) 报告他们在高并发下预置隔离 Docker 运行时的原因，而理由恰恰是否则环境搭建会主导采样时间。

19.3.2 多少沙箱才能让 N 个工作进程不闲着

容量问题有一个精确答案，它不依赖延迟分布，只依赖其均值。

推导: 由利特尔法则给出的池容量

利特尔法则 (Little, 1961) 指出，对任何稳定系统，系统内物品数的均值等于到达率乘以每件物品在系统内停留时间的均值，$\bar{L} = \lambda W$。把「系统」取为当前被某次采样持有的沙箱集合。

表 19.2: 采样沙箱的各种隔离层。冷启动数字是在镜像已缓存的热主机上的数量级估计，也是进入节 19.3.2池容量算术的那个量；它们并非来自某个具体集群的实测。重置一列所写的，正是使一个环境能够在一个组的各条采样轨迹之间被复用的机制。

层次	冷启动	重置机制	适用场合
进程 `rlimit`	∼10 ms	新建进程、临时目录	纯函数式工具；无不可信写入
用户态内核	∼100 ms	容器重启	共享主机上的不可信代码 (The gVisor Authors, 2026)
OCI 容器	0.5–5 s	丢弃覆盖层	镜像很重的仓库类任务
微虚拟机	0.1–1 s	虚拟机快照恢复	要在容器密度下取得硬件隔离 (Agache et al., 2020)
完整虚拟机	10–60 s	磁盘快照回滚	图形界面与操作系统环境

设有 N 个采样工作进程并发运行，且一个处于稳态的工作进程在两件事之间交替：生成一个轮次的动作词元，平均耗时 G_t；以及为一次调用持有沙箱，平均耗时 $s + L$，其中 s 是沙箱获取或冷启动时间，L 是执行时间。于是每个工作进程每 $G_t + s + L$ 秒发出一次沙箱请求，即 $\lambda = N/(G_t + s + L)$，而每次请求占用沙箱 $W = s + L$。因此

$$\bar{C} \;=\; \lambda W \;=\; N\,\frac{s+L}{G_t + s + L}. \tag{19.2}$$

取 $N = 256$、$G_t = 18\,\text{s}$、$L = 4\,\text{s}$ 且池是热的 ($s \approx 0$)，平均并发量为 $256 \cdot 4/22 \approx 47$ 个沙箱。加上两秒的容器冷启动，这个数字升到 $256 \cdot 6/24 = 64$——你必须拥有的池子扩大了 37%，而这完全是「没有把沙箱保持在热状态」买来的。

按均值配置会让工作进程有一半时间在阻塞。把这 N 个工作进程当作独立的伯努利占用者，$\rho = (s+L)/(G_t+s+L)$，则忙碌数近似服从 $\text{Binomial}(N, \rho)$，于是容量为 $C = N\rho + z\sqrt{N\rho(1-\rho)}$ 的池子，其阻塞概率由 z 控制；在 $N = 256$、$\rho = 2/11$ 下这是 $47 + 3 \cdot 6.2 \approx 66$。与分布无关的部分是 $\bar{C}$；延迟分布的尾部只通过余量进入，这就是「应当记录延迟直方图而不是均值」的现实理由。

(19.2)有一条推论，是环境工程中杠杆率最高的单个决策。如果沙箱是按调用持有的，那么在上面的数字下复用倍数 $(G_t + s + L)/(s + L)$ 大约是 5.5。如果它是按回合持有的——只要环境是有状态的就必须如此，而这涵盖了每一个仓库、终端与操作系统任

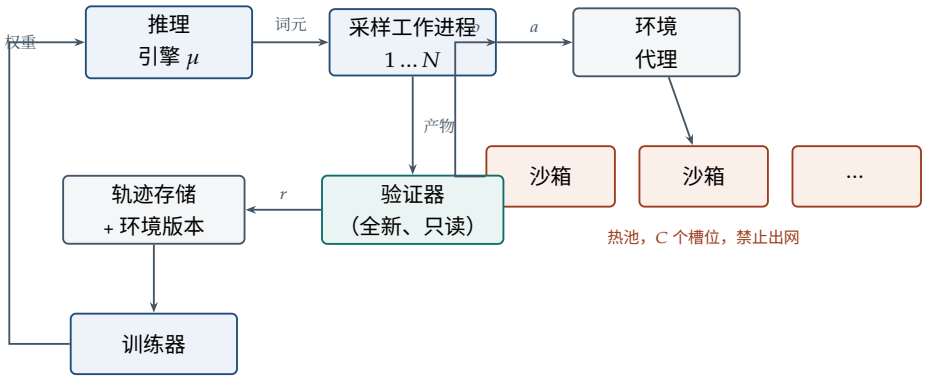


图 19.1: 本章通篇所假定的采样工作进程与环境池架构。图中显式画出了三处要紧的分离：沙箱是一个按(19.2)定容量的共享池，而不是每个工作进程一个；验证器运行在一个全新的、只读的、策略沙箱够不到的环境中（节 19.5）；并且每条轨迹在抵达训练器之前都被打上了环境版本戳（节 19.8）。图中没有任何一处是某个策略梯度变体所特有的。

务——那么 W 就是整个回合的时长，$\rho = 1$，$\bar{C} = N$ 恰好成立：每个工作进程一个活的沙箱，在任何延迟下都没有复用余地。这就是软件工程强化学习的资源配置方式与其他任何一类都不同的原因，也是它会把编排系统搞垮的原因。在 DeepSWE-Preview 的开发过程中，每次迭代运行 512 个并发容器压垮了 Docker API 服务器并使守护进程崩溃，迫使他们改用 R2E-Gym 中采用的基于 Kubernetes 的编排 (Luo et al., 2025a; Jain et al., 2025b)；NVIDIA 的 Nemotron 流水线把软件工程强化学习作为一个独立阶段来跑，因为其采样长且吃上下文，足以成为共享该集群的其他所有环境的瓶颈，而且他们用的是 Apptainer 而非 Docker，因为其 SLURM 集群没有 root 权限 (NVIDIA, 2026d)。把一个回合级作用域的环境改造成可快照可恢复的环境，使沙箱按调用而非按回合持有，可以把可支撑的工作进程数乘上 $(G_t + s + L)/(s + L)$。这值得投入大量工程。

代码 19.1: 一个沙箱化的执行客户端。有三条性质承载了 节 19.1的论证：资源限制是在子进程内、exec 之前施加的，因此一个失控的程序是被内核杀掉，而不是靠与框架赛跑来杀掉；每一种失败都以带类型的结果返回而非抛出异常，从而使采样循环能区分「智能体错了」与「我们没能评估智能体」；而基础设施类失败被单独上报，使(19.1)中的抖动率可被观测。

```
import resource, subprocess, uuid
from dataclasses import dataclass

TASK_FAIL = {"nonzero_exit", "assertion_failed"}   # 计为奖励 0
INFRA_FAIL = {"oom", "sandbox_timeout", "spawn_error", "egress_blocked"}  # 剔除
```

```

@dataclass
class ExecResult:
    stdout: str; exit_code: int; outcome: str; wall_s: float; env_version: str

def _limits(cpu_s, mem_mb, nproc):
    def apply():                                  # 在子进程中、exec 之前运行
        resource.setrlimit(resource.RLIMIT_CPU, (cpu_s, cpu_s))
        resource.setrlimit(resource.RLIMIT_AS, (mem_mb << 20,) * 2)
        resource.setrlimit(resource.RLIMIT_NPROC, (nproc, nproc))
        resource.setrlimit(resource.RLIMIT_FSIZE, (64 << 20,) * 2)
    return apply

def run(pool, code, wall_s=10.0, cpu_s=8, mem_mb=1024, nproc=64):
    slot = pool.acquire()            # 热沙箱；若无空闲则阻塞
    try:
        proc = subprocess.run(
            slot.argv("python", "-"), input=code, text=True, timeout=wall_s,
            capture_output=True, preexec_fn=_limits(cpu_s, mem_mb, nproc),
            env=slot.env_no_network())
        code_ = proc.returncode
        outcome = "ok" if code_ == 0 else (
            "oom" if b"MemoryError" in proc.stderr.encode() else "nonzero_exit")
        return ExecResult(proc.stdout[:8192], code_, outcome, slot.elapsed(),
                          slot.version)
    except subprocess.TimeoutExpired:
        return ExecResult("", -1, "sandbox_timeout", wall_s, slot.version)
    except OSError as exc:
        return ExecResult("", -1, "spawn_error", 0.0, slot.version)
    finally:
        pool.release(slot, reset=True)            # 丢弃覆盖层
```

19.4 确定性，以及非确定性让组相对估计量付出的代价

19.4.1 有哪些东西必须钉死

当（任务标识符，种子，环境版本）这个三元组确定了观测序列时，一个环境才是可复现的。要做到这一点，需要钉死四样通常被放任漂移的东西：容器镜像要按摘要而非按标签钉死，以免重建的基础镜像悄悄换掉了 Python 的次版本号；依赖解析要钉死，办法是内置一份锁文件并在安装时禁止出网；沙箱内可见的时钟与随机种子要钉死；以及全部网络流量要钉死，办法是在第一遍时录下每一个请求与响应，此后都从那盘录像带里回放。录制并回放的流量是让一个网页环境根本得以可复现的唯一办法，而且

它还有一项额外好处，即把限流器从奖励函数中移除了。它的代价是：环境如今只在被录制到的支撑集上有定义，这是一项实实在在的限制，应当写进该环境的文档，而不是留给别人去发现。

模型内部的确定性是另一个有其自身文献的问题：即便在温度为零时，推理服务器对完全相同的请求也会返回不同的补全，因为算子核并非批次不变，于是归约顺序取决于同批次中的其他流量 (He and Thinking Machines Lab, 2025)。这归第22章管辖。下文讲的是环境。

19.4.2 为什么这种损害对组相对估计量尤为特殊

非确定性会抬高任何估计量的方差。而在组相对估计量中，它还做了一件更糟的事：把自己伪装成信号。

推导: 环境噪声会衰减归一化后的优势

固定一个提示以及由 G 条采样轨迹构成的一个组。把成员 i 的观测奖励写作 $r_i = f_i + \varepsilon_i$，其中 f_i 是环境若按规范行事本应返回的奖励，而 ε_i 是其偏离——当一条正确轨迹被虚假判负时取 -1，当一条错误轨迹被虚假判过时取 $+1$。假设 ε 均值为零、方差为 σ_ε^2，并记 $\sigma_f^2 = \mathbb{V}\mathrm{ar}[f]$。以 $\hat{\mu}$ 记组均值，未归一化的优势分解为

$$\tilde{A}_i \;=\; r_i - \hat{\mu} \;=\; (f_i - \bar{f}) + (\varepsilon_i - \bar{\varepsilon}).$$

它与理想优势的协方差是 $\frac{G-1}{G}\sigma_f^2$，它的方差是 $\frac{G-1}{G}(\sigma_f^2 + \sigma_\varepsilon^2)$。两项带着同一个 $\frac{G-1}{G}$：增大 G 能把基线洗干净，却洗不干净 r_i 本身，因此逐样本的信噪比 $\sigma_f^2/\sigma_\varepsilon^2$ 与组大小无关。

现在像 GRPO 那样除以样本标准差。当 G 很大时 $\hat{\sigma}^2 \to \sigma_f^2 + \sigma_\varepsilon^2$，于是归一化后的优势与理想优势之间的相关系数为

$$\mathrm{corr}(\hat{A}_i,\ (f_i - \bar{f})/\sigma_f) \;=\; \frac{\sigma_f}{\sqrt{\sigma_f^2 + \sigma_\varepsilon^2}} \;=\; \frac{1}{\sqrt{1+\kappa}}, \qquad \kappa = \frac{\sigma_\varepsilon^2}{\sigma_f^2}. \tag{19.3}$$

更新保持了单位幅值——这是归一化所保证的——而其长度中只有 $1/\sqrt{1+\kappa}$ 的一部分指向真实梯度方向。换成一个评论家或一个固定基线，会把步长缩小，而步长缩小是看得见的；$\hat{\sigma}$ 归一化则把步长重新拉回满幅，而这是看不见的。

有两条后果直接随之而来，且都能在训练日志中观测到。第一，由于归一化把每个提示都拉回单位尺度，一个奖励主要来自环境噪声的提示，在批次中的权重与一个干净提示的权重相同。因此，相对于一个不做归一化的估计量，组相对归一化加权放大了池中最嘈杂的那些提示——这与它加权放大近乎退化的提示是同一个机制（第12章），只是如今作用在一个完全不携带信息的量上。

第二，抖动会从课程本已淘汰的提示中凭空制造出工作量。考虑一个策略能确定解出的提示，$p = 1$，于是 $\sigma_f = 0$、$\kappa = \infty$：它的优势里每一丝都是噪声。动态采样 (Yu et al., 2025c) 会丢弃奖励全部相同的组，本来也会丢弃这一个——除非环境发生抖动。当逐采样抖动概率为 q 时，该组以概率 $1 - (1-q)^G - q^G$ 通过过滤。在听上去无害的 $q = 0.03$ 与 $G = 16$ 之下，这个概率是 38.6%。超过三分之一的已解决提示会通过可学习性过滤，并交付一次满幅、完全虚假的更新；而且随着策略变强、池中越来越多的提示达到 $p = 1$，这种情况只会越来越频繁。其症状是：训练曲线进入平台期，而退化组比例却低得令人安心。

代码 19.2: 一个抖动检测器。它在一个参考策略下、以同一个种子重跑同一个任务，测量那些无法归因于策略的奖励方差，并同时返回进入(19.1)的抖动率与 (19.3)的噪声比 κ。在训练开始前对整个池跑一遍，并在训练中每隔几百步对其随机子集跑一遍，是本章中最廉价的诊断。

```python
import statistics as st

def audit_task(env_factory, task, policy, replays=8, seed=0):
    """在固定种子与贪心策略下重放同一个任务。"""
    rewards, infra, actions = [], 0, None
    for _ in range(replays):
        env = env_factory(task=task, seed=seed)          # 完全相同的重置
        traj = rollout(env, policy, temperature=0.0)     # 完全相同的意图
        if traj.outcome in INFRA_FAIL:
            infra += 1
            continue
        rewards.append(traj.reward)
        if actions is None:
            actions = traj.action_hashes
        traj.replay_divergence = (traj.action_hashes != actions)
    n = len(rewards)
    var_env = st.pvariance(rewards) if n > 1 else 0.0    # 策略是固定的:
    return {                                             # 全部方差都来自环境
        "task": task.id,
        "flake_rate": infra / replays,
        "reward_var_env": var_env,
        "reward_mean": st.fmean(rewards) if n else float("nan"),
```

```
        "deterministic": var_env == 0.0 and infra == 0,
    }

def quarantine(report, p_hat, flake_max=0.01, kappa_max=0.10):
    """kappa 把环境方差与伯努利信号方差作比较。"""
    signal_var = p_hat * (1.0 - p_hat)
    kappa = float("inf") if signal_var == 0 else report["reward_var_env"] / signal_var
    return report["flake_rate"] > flake_max or kappa > kappa_max
```

19.5 环境内部的验证

19.5.1 状态检查与结果检查是不同的契约

一次结果检查把智能体返回的某样东西与一个参考作比较：字符串匹配、数值容差、一个被解析出来的答案。它便宜，容易做成只读，而且只约束智能体说了什么。一次状态检查则在回合结束后审视环境：跑测试套件、比对数据库、查询应用。它昂贵，它约束的是智能体做了什么，而且一旦任务是去改变世界而非回答问题，它就是唯一的选择。两者的成本差距在延迟上是 10 到 100 倍，这会直接传导进(19.2)，也正是一套工程良好的技术栈会把验证放在自己的池中运行、而不是内联在采样工作进程里的原因（图 19.1）。

状态检查的优势同时也是它的暴露面。结果检查读的是一个字符串；状态检查读的是一个智能体拥有写权限的文件系统。这个不对称性生成了整个奖励作弊面，而修补它靠的是一条原则。

定义 19.2 (只读验证)**.** 设 $\mathcal{A} \subseteq \mathcal{S}$ 为产物面：环境状态中任务意图让智能体去修改的那一部分——通常是一份差分、一组被创建的文件，或一组数据库行。当奖励仅是 $\mathcal{A}$ 的函数时，验证就是只读的；其计算方式是把 $\mathcal{A}$ 从智能体的沙箱中提取出来，施加到一个由原始镜像全新实例化的环境上，并在那里运行校验器。参与计算奖励的过程中没有任何一个是智能体能够影响的。

已知的每一种测试套件作弊手法都是对定理 19.2的违反，并且可以按它违反了哪一部分来分类。修改测试——改断言、加一个给框架打猴子补丁的 `conftest.py`、把用例标记为跳过——违反的是「奖励只依赖 $\mathcal{A}$」这条限制，其修补办法是在评分前从原始镜像恢复测试文件，而不是去请求智能体别动它们。抓取答案——把修复该问题的上游提交、或该基准自己公布的解答取回来——违反的是环境的隔离，其修补办法是禁止出

网，而不是过滤 URL。硬编码——对可见测试中出现的那些输入做特判——在结构上没有违反任何东西，只能靠统计手段挫败，即把用于验证的测试从智能体能看到的测试中留出来。利用框架——写出结果文件、强制退出状态为零、派生一个在校验器读取报告之后再去改写它的后台进程——违反的是「校验器必须运行在智能体够不到的地方」这一要求。在实践中，随着训练推进，这些手法会按上述精巧程度依次出现，这是一个有用的预警次序：一次已经开始改测试的训练，几百步之后就会开始打框架的补丁。

19.5.2 留出测试，以及你需要多少

假设一个任务自带 n 个测试，其中 k 个对智能体可见（作为一份复现脚本，或者仅仅因为它们就在仓库里），另外 $n-k$ 个被留出用于评分。对可见用例做硬编码，只有在留出测试恰好考察了同样的输入时才能拿到奖励。因此起保护作用的不是 $n-k$ 这个计数，而是留出测试相对可见测试的独立性：由生成了那 10 个可见用例的同一个基于性质的生成器自动产生的 50 个留出用例，其保护作用远不如 3 个探查另一条代码路径的手写用例。这正是 2026 年基准设计转向背后的推理：Huang et al. (2026c) 手写了功能性验证器，它接受任何正确实现，而不是继承仓库自己的测试；他们报告，一个 LLM 评判模型与他们的验证器的分歧率约为 1.4%，而对一个使用继承测试的基准，这一比例是 32.4%。验证器质量不是一个基准测试的细节问题；一个有 32% 时间是错的验证器，就是一个在(19.3)中 $\kappa \approx 1$ 的环境，这要花掉大约 30% 的梯度方向。

协同演化这支文献从供给侧攻击同一个问题，办法是把测试生成器与求解器一起训练 (Wang et al., 2025d)，或是合成测试用例以在本来没有奖励信号之处制造出一个信号 (Zeng et al., 2025; Jin et al., 2025b)。这买来了覆盖度，同时也把作弊面在更高一层重新打开了，因为一个习得的测试器可以被一个学会了该测试器盲点的求解器所满足。缓解办法与下一节所主张的对抗式审计相同。

19.6 习得环境与世界模型

对环境稀缺最直接的回应，是让一个语言模型来扮演环境。这如今已是一项有真实结果的真实技术。Sun et al. (2025) 用一个微调过的模型替换搜索引擎，它在一个逐步降低文档质量的课程之下，既产生有用文档也产生刻意加噪的文档，从而完全消除了 API 成本；他们报告，作为训练环境，一个 7B 的模拟器可与真实引擎持平，而一个 14B 的模拟器则超过了它。Cai et al. (2025) 从工具文档生成工具实现与一个模拟用户；Tu et al. (2026) 与Wang et al. (2026e) 则从零合成整个交互式环境。在另一个层面上，

Song et al. (2026) 在测试时使用模拟验证，而节 17.8的观测预测目标则把世界模型作为采样本已产生之物的副产品，训练进了策略内部 (ECHO, 2026; PaW, 2026)。

问题在于这种做法何时站得住脚。它有一个精确的答案。

19.6.1 优化器会挑出模拟器的错误

命题 19.3 (优化之下的模拟器次优性)**.** 设 P 为真实环境动态、$\hat{P}$ 为一个习得的模拟器，设奖励取值于 $[0,1]$，时域为 T 个轮次。若对某个策略 π，在 π 之下可达的每一个状态—动作对上总变差距离都满足 $\mathbb{D}_{\mathrm{TV}}\big(P(\cdot\,|\,s,a)\,\big\|\,\hat{P}(\cdot\,|\,s,a)\big) \le \epsilon$，则 $\big|V_P^{\pi} - V_{\hat{P}}^{\pi}\big| \le T\epsilon$。于是，若 $\hat{\pi} \in \arg\max_{\pi} V_{\hat{P}}^{\pi}$，且该界在 $\hat{\pi}$ 与真实最优 $\pi^{\star}$ 之下都成立，则

$$V_P^{\pi^{\star}} - V_P^{\hat{\pi}} \;\le\; 2T\epsilon. \tag{19.4}$$

证明. 把两个轨迹分布逐步耦合起来。在 T 步中的每一步上，两个转移核在总变差意义下相差至多 ϵ，于是对混合序列取并集界，两个轨迹分布相差至多 $T\epsilon$；既然回报取值于 $[0,1]$，值函数之差同样如此。至于第二条论断，链式地写出 $V_P^{\hat{\pi}} \ge V_{\hat{P}}^{\hat{\pi}} - T\epsilon \ge V_{\hat{P}}^{\pi^{\star}} - T\epsilon \ge V_P^{\pi^{\star}} - 2T\epsilon$，其中中间一步用到了 $\hat{\pi}$ 在 $\hat{P}$ 之下的最优性。 □

(19.4)就是标准的模拟引理论证 (Kearns and Singh, 2002)，它看起来令人安心。其实不然，理由有两条，而这两条就是本节的全部内容。时域是线性进入的，所以在 $T = 100$ 轮次时，一个每步错 1% 的模拟器容许的次优性达到 2——对一个取值于 $[0,1]$ 的奖励而言，这是空洞的。而更根本的是，该界的成立以 ϵ 在$\hat{\pi}$ 之下成立为条件，而当你审计模拟器时 $\hat{\pi}$ 还不存在，且它恰恰是通过最大化模拟器搞错的那个量产生出来的。所谓平均保真度——从一个参考策略采样轨迹、比较模拟观测与真实观测——界定的是该参考策略占用测度上的 ϵ，对优化器的占用测度则什么也没说。强化学习是在寻找「$\hat{P}$ 的误差面加上真实奖励」的 argmax，而如果前者更便宜，它就会去找前者。

一旦你去找，其经验特征便不会认错：模拟器中的奖励平滑上升，而真实环境留出集上的奖励却停滞或下降；且检视奖励最高的那些轨迹会看到一种「形式完好的胡话」式行为——一次真实 API 会拒绝其参数的工具调用，一个模拟器乐于凭空编造的网页。

19.6.2 对抗式审计，以及何时可以接受风险

正确的审计不是一次保真度测量，而是一次攻击。取当前策略按模拟奖励排名的前 k 条轨迹——不是随机样本，而是最靠前的那些——把它们放到真实环境中重新执行一遍。真正要紧的统计量是那一薄片上的奖励差距，而且它应当被当作一条训练曲线来跟踪。

一个平均保真度极佳、而顶部切片差距却在扩大的模拟器，就是一个正在被利用的模拟器，而均值不会把这件事显示出来。

在此基础上，针对一个习得环境做训练，在若干可以列成检查清单的条件下是站得住脚的。

1. **奖励不是模拟出来的**。模拟器提供观测流；奖励 r 由一个真实的、以程序实现的验证器提供。习得环境取得最强结果的正是这一配置，因为优化器可以利用动态，却无法伪造分数。
2. **误差由构造而非由测量来限定**。一盘录制并回放的录像带在其支撑集上 $\epsilon = 0$，在支撑集之外则无定义，因此把动作空间限制在被录制的支撑集上就给出了一项真实的保证。一个手写的桩件属于同一类。而一个习得的模拟器不属于，应被视作最弱的情形。
3. **支撑集是被有意拓宽的**。ZeroSearch 那条逐步降低文档质量的课程 (Sun et al., 2025) 在此语境下就是域随机化：通过让模拟器的规律性本身发生变化，阻止策略以这些规律性为条件。
4. **顶部切片差距被监控并被设为闸门**。当策略最好的那些轨迹上的差距超过一个预先登记的阈值时，就停止训练或重新训练模拟器。
5. **最后一个阶段是真实的**。在真实环境上跑一个收尾阶段，把策略的占用测度移回 P 之下，而这是唯一能让 $\hat{\pi}$ 之下的 ϵ 变得无关紧要的办法。

当模拟器扮演的是一个人而不是一个系统时，请把上述每一条都当作严格更难来对待。模拟用户是任何环境中最容易被利用的部件，因为它的失败模式是随和，而随和恰恰是一个最大化奖励的对手方会去诱发的东西。τ^2-bench 正是出于这个理由，用工具与可观测状态去约束它的用户模拟器 (Barres et al., 2025)，而Cai et al. (2025) 把模拟用户行为失当的那些采样轨迹同时掩蔽出优势与损失。

19.7 任务供给才是真正的瓶颈

19.7.1 固定的池会耗尽，而且耗尽时刻可以算出来

节 4.5确立了：只有当一个提示的通过率严格介于零与一之间时，它才贡献梯度。这一事实带来一个供给上的后果，而这个后果通常是靠昂贵的经验代价才被发现的。

推导: 一个任务池的可用寿命

设池中有 M 个任务，一个训练步消耗 B 个提示。给每个任务一份 E 次有效曝光的预算：即在策略要么掌握它、要么被证明永远解不出它之前，它能在通过率仍处于可学习区间内被采样的次数。于是该池大约能支撑

$$T_{\text{useful}} \approx \frac{M\,E}{B} \tag{19.5}$$

个携带梯度的训练步。DeepSWE-Preview 在源自 R2E-Gym 的 4,500 个任务上训练 (Luo et al., 2025a; Jain et al., 2025b)；在每步 $B = 128$ 个提示、以及慷慨地取 $E = 20$ 之下，(19.5)给出大约 700 步。在同样的批大小下要达到 5,000 步，则需要大约 32,000 个任务。这就是 2026 年环境规模化计划背后的算术，也是已发布语料规模之所以如此的原因：SETA-Env 中的 4,567 个经验证的终端环境 (Shen et al., 2026b)，ToolVerse 中横跨约 400 个真实 Model Context Protocol 服务器的约 4,500 个工具 (Zhou et al., 2026; Anthropic, 2024)，以及 ScaleCUA 中 24,000 个合成的可验证计算机操作任务 (Lv et al., 2026)。

有两处修正会让(19.5)在一个方向上偏悲观、在另一个方向上偏乐观。偏悲观之处：E 并非固定，因为一个现在太难的任务日后可能进入该区间，所以随着策略变强，池子在一定程度上会自我刷新。偏乐观之处：一个由基准衍生而来的池，并不是 M 个互相独立的任务，而是数量少得多的任务模板，而策略饱和的是模板而非实例。实践上的读法是：任务的数量是错的统计量，不同任务结构的数量才是对的——而这正是Tu et al. (2026) 在推导出一条领域规模化曲线并发现环境多样性比任务数量更要紧时，从经验上得到的结论；也是Yu et al. (2025b) 从另一个方向得到的结论，后者表明具备可利用内部结构的环境比无结构的环境训练效率更高。

19.7.2 验证优先的构造

任务的生成与校验器的生成通常是按这个先后次序处理的，而这个次序是错的。没有校验器的任务无法使用；而没有任务的校验器可以被反演成一个任务。先生成校验器还能强制满足一条启发式任务生成经常违反的性质，即该任务对你手头实际拥有的机制而言是可判定的。

算法 19.1是 2026 年各条合成流水线所共有的模式的抽象陈述。Cai et al. (2025) 自底向上地从工具文档实例化它：生成一个键值状态空间与若干 Python 工具实现，构建一张工具依赖图，做随机游走以得到工具序列，实例化具体取值与一个标准终态，再

以精确状态相等给予奖励。Shen et al. (2026b) 则从异构来源实例化它——竞赛笔记本、问答网站——把它们在一个统一验证器之下转换成标准化的终端环境，外加一条围绕模型能力前沿自适应控制难度与多样性的演化流水线。Tao et al. (2025) 用集合论把信息搜寻任务的合成形式化，使难度与结构变得可控而非涌现；Mei et al. (2026) 则加上一条联合检查问答一致性、轨迹质量与证据相关性的验证流水线。

挖掘是与之互补的供给路线，而在其适用之处它更便宜。仓库历史每一个修复缺陷的提交都能给出一个任务，并以该提交自己的测试作为校验器 (Wei et al., 2025d; Yang et al., 2025)；问题追踪系统给出带自然语言规格的任务，并以解决状态作为弱标签 (Jimenez et al., 2023; Badertdinov et al., 2025)；生产日志则给出合成生成系统性遗漏的真实用户意图分布。这三者都继承了下文讨论的溯源问题。

19.7.3 难度校准与提议器

节 4.5给出了目标：让通过率远离零与一。环境所特有之处在于，通过率并非任务单独的性质，而是任务与框架共同的性质，因此重新调一次超时就会同时改变池中每一个任务的难度，而一次环境版本升级会使每一份缓存的难度标签失效。请针对你将要在其上训练的那个环境版本来标注难度，并在任何限额变更之后重新标注。

让任务池随策略一起生长，就是提议器问题，而它有一段值得了解的、前大模型时代的谱系：自动目标生成 (Florensa et al., 2018)、非对称自博弈 (Sukhbaatar et al., 2018)、无监督环境设计 (Dennis et al., 2020)，以及环境—解答对的开放式生成 (Wang et al., 2019)。大模型时代的实例则有 Absolute Zero 的「提议—求解—验证」循环 (Zhao et al., 2025a)、挑战者—求解者协同演化 (Huang et al., 2025a)、以语料为依托的自博弈 (Liu et al., 2025a)、搜索专用的自博弈 (Lu et al., 2025)，以及由智能体自身失败构建出的自演化课程 (Qi et al., 2024d)。它们共同反复出现的发现是：提议器必须与求解器协同演化，而不能固定不变，因为一个固定提议器的难度分布会在几百步之内与求解器的前沿脱钩——这与节 4.5中的陈旧度论证一模一样，只是如今被用在了生成器上。

19.7.4 任务池与评测之间的污染

一个由公开仓库、公开问题追踪系统与公开基准拼装起来的任务池，按其构造方式就与评测套件重叠，而一旦这些任务被某个生成器改写过，这种重叠就无法靠字符串匹配检测出来 (Contamination Detection Review, 2026)。有三条做法是承重的。去污染要在仓库与提交这一层级做，而不是在任务文本层级，因为同一个缺陷会以许多种措辞出现。为每一个任务记录溯源信息——来源、提交、生成器、生成器版本——以便一旦发

现污染，可以靠删除来处置，而不必丢弃整个池。并且优先采用由从未被上游贡献过的任务构建的评测套件 (Huang et al., 2026c)，或随时间向前滚动的评测套件 (Jain et al., 2025a; Badertdinov et al., 2025)。节 4.6负责一般方法；此处特有的是，一个环境的污染除了在文本里，还在它的依赖里，因为一个被钉死的镜像可能在某个 site-packages 目录中就藏着参考解。

19.8 环境集群、枢纽，以及标准化能买到什么

19.8.1 2026 年的格局

2024 到 2026 年间发生了三件事。环境从库变成了服务：AgentGym-RL 把每个环境作为独立进程置于统一的 HTTP 接口之后 (Xi et al., 2025)，AgentRL 在一个中央控制器之下运行容器化的环境工作进程 (Zhang et al., 2025b)，而 Meta 的 ARE 平台在同一个底座上托管了一千多个场景 (Meta, 2025, 2026)。工具接口标准化到了 Model Context Protocol 上 (Anthropic, 2024)，正是它使Zhou et al. (2026) 得以从数百个真实服务器而非合成桩件出发构建可执行环境。而环境规格本身也成了可复用的产物——一个任务、它的镜像摘要、它的验证器与它的限额，一起做版本管理——SETA-Env、ScaleEnv 与 InternBootcamp 正是以这种形式分发的 (Shen et al., 2026b; Tu et al., 2026; Li et al., 2025c)。

值得精确说明这买来了什么。标准化买来的是可比性：在同一个环境版本、同样的限额之下评测的两个配方，其数字可以相减。它买来了工程复用，因为一套写一次的框架可以服务许多任务。它买不来迁移。共享接口这件事本身，并不会让一个在某个环境中训练出来的策略在另一个环境中更好，而这方面的证据略显令人沮丧：Xi et al. (2025) 报告，强化学习的增益在结构化的、基于规则的环境中最大，在嘈杂的现实环境中最小；而若结构化环境本应充当一个廉价代理，这个次序恰好反了。请把一个标准化的环境枢纽当作基础设施，而不是当作一份课程。

19.8.2 运维环境集群

训练规模下的环境集群其行为就像一个生产服务，需要同样的那套纪律。有四条不可退让。

健康检查对每一类环境中的一个金丝雀任务持续运行，并针对代码 19.2的抖动率而非进程存活状态发出告警，因为真正要紧的失败是一个仍然活着、却返回错误奖励的环境。

检疫在某个任务实测的 q 超过阈值时把它从采样池中移除，而不是移除整个环境。检疫必须自动且可逆：一个因为注册表短暂故障而抖动的任务应当回来，而一个因为测试套件里有竞态条件而抖动的任务则不应当回来。请把是哪一种记录下来。

逐任务奖励审计抽取已完成的轨迹，并把奖励与第二个独立判断相核对——一个人、一个更强的模型，或一个以不同方式实现的校验器。要跟踪的统计量是逐任务的分歧率，而不是总体分歧率：2% 的总体分歧完全可能来自少数几个干脆就是错的任务，而策略正在学着去利用的就是那几个任务。

版本管理让奖励的变化变得可见。一个环境版本应当是「任何可能改变奖励之物」的哈希：镜像摘要、框架源码、验证器源码、资源限额与超时，以及语料快照标识符。把它戳在每一条轨迹上（图 19.1），并拒绝在一个组内混用不同版本。好处是，一次奖励回归会在日志中表现为一次版本升级与一次指标变化相关联，而不是三周之后一次没人能二分定位的、无法解释的训练异常。一个从 5 秒提高到 10 秒的超时就是对奖励函数的一次改动，而把它当作一次配置微调，正是训练变得不可复现的原因。

19.9 在训练之前先验证一个新环境

本章开头那处不对称在于：一个数据集可以靠读来验证，而一个环境只能靠执行来验证。下文给出的是最低限度的执行。表 19.3把每一项检查都写成一次带通过条件的测量，因为一份条目都是形容词的清单不算清单。

工程实践：这些检查该按什么次序跑

先跑隔离，因为这里出问题可能损害整个集群，而且一旦沙箱是漏的，后面每一项测量都没有意义。其次跑确定性与抖动率，因为它们是解读其他一切的前提：一个非确定性的环境会让校验器可靠性无从检验。第三跑校验器的可靠性与完备性，用刻意写错的产物和刻意写得不寻常但正确的产物来跑。最后跑难度，因为它是唯一依赖于策略的检查，因而也是唯一必须在训练过程中反复重做的检查。预算大约是 $m \cdot M$ 个重放回合加 $G \cdot M$ 个难度回合；取 $M = 1000$ 个任务、$m = 8$、$G = 8$，就是 16,000 个回合，在一个规模不大的池上也就几个小时，比浪费掉一天训练要便宜。

表 19.3: 一个新环境的验证方案。每一行都是一次要在第一个训练步之前跑完的测量，总代价是几千条采样轨迹。这些阈值是用来被争论的默认值，而不是常数；不可让步的是每一项都得挂上一个数字，因为一条未被测量的环境性质会变成奖励中一项未被测量的项。

检查项	测量方式	通过条件
重放确定性	每个任务在固定种子与贪心策略下重跑 $m = 8$ 个回合；比较观测哈希	在 $\geq 99\%$ 的任务上完全一致
抖动率	同一批重放中以 `INFRA_FAIL` 结束的回合比例	每回合 $q < 0.01$
奖励噪声	由代码 19.2给出的 $\kappa = \sigma_\varepsilon^2 / \hat{p}(1-\hat{p})$	$\kappa < 0.1$
校验器可靠性	对空产物与一个明显错误的产物各跑一次校验器	两种情形奖励均为 0
校验器完备性	对 ≥ 2 个不同的正确解各跑一次校验器	全部奖励为 1
可达性	一个参考解或人工解取得奖励 1	已接收任务的 100%
隔离	从内部尝试出网、沙箱外写入与派生炸弹	全部被阻止，沙箱存活
验证器隔离	在回合内部改动测试文件，然后评分	奖励不变
难度	由将要在其上训练的策略的 G 条采样轨迹给出的 $\hat{p}$	$\underline{p} \leq \hat{p} \leq \overline{p}$
延迟画像	回合时长的 p50、p95、p99	p99 $<$ 10× p50，否则要预期出现掉队者
池容量	由(19.2)给出的 $\bar{C}$ 对照已配置容量	余量 $\geq 3\sigma$
污染	已记录溯源；按仓库与提交计算与评测的重叠	零重叠，溯源完整
版本管理	环境哈希覆盖镜像、框架、验证器、限额、语料	戳在每一条轨迹上

常见陷阱：用错误的策略去验证环境

难度、抖动率、乃至校验器可靠性，全都是在某个策略之下测出来的，而做验证的那个策略很少就是做训练的那个策略。一个很强的参考模型会产生短促、干净、从不触发超时、也从不走到框架错误路径上的回合，于是在它之下验证过的环境看上去既确定又便宜。而训练中的策略，尤其在早期，会产生冗长、畸形、吞噬资源的回合——并在几百步之内就发现那些错误路径。请用你将要出发的那个检查点、在你将要采样的那个温度下做验证，并在头一千步之后把抖动率重新测一遍。

本章小结

- 基础设施失败以一个塑形项 $-q(a)p(a)$ 的形式进入目标函数，该项对真实成功概率是乘性的，因此它对好动作的惩罚重于对坏动作的惩罚，并且能够在写下的奖励毫无改动的情况下重排整个动作空间的优劣次序。
- 各类环境之间的差别，与其说在于它们教会什么，不如说在于它们如何被钻空子；作弊手法可以从奖励语义预测出来，而状态检查——它要去读一个智能体本可写入的文件系统——带有结果检查所没有的暴露面。
- 沙箱池的容量由利特尔法则给出，$\bar{C} = N(s + L)/(G_t + s + L)$，与延迟分布无关。回合级作用域的环境迫使 $\bar{C} = N$ 且毫无复用余地，这既是软件工程强化学习要单独配置资源的原因，也是把一个环境改造成可快照可恢复值得投入大量工程的原因。
- 环境噪声把组相对梯度的方向衰减了 $1/\sqrt{1+\kappa}$，而归一化又把步长恢复到满幅，于是一个嘈杂的提示相对于一个干净的提示是被加权放大的，而不是被折扣的。3% 的抖动率会让 39% 的已解决提示在 $G = 16$ 时通过动态采样，并发出纯粹的噪声。
- 只读验证——提取产物、从原始镜像重建环境、在那里评分——从结构上消除了修改测试与利用框架；而硬编码只能靠统计手段挫败，办法是让留出测试独立于可见测试，而不只是数量更多。
- 针对一个习得的模拟器做训练，就是在该模拟器的误差面上做最大化，因此平均保真度什么也界定不了。审计必须是对抗式的：把策略奖励最高的那些轨迹放到真实环境中重新执行，并跟踪那一薄片上的差距。
- 一个固定的任务池大约能给出 ME/B 个携带梯度的训练步；按实际已发布的规模，这不到一千步，这正是 2026 年从「精选任务」转向「生成环境」的原因，也是能够规模化的量是结构的多样性而非任务的数量的原因。
- 验证优先的构造——先生成校验器，再生成任务，然后依次按校验器可靠性、可达性、重放确定性与可学习性来拒绝——是每一条行得通的自动化环境合成流水线所共有的模式。
- 标准化的环境接口买来的是可比性与工程复用；它们买不来迁移，而实测增益最大的恰恰出现在最不像部署场景的那些结构化环境上。

- 一个环境版本是镜像、框架、验证器、限额与语料的哈希。没有它，一次奖励变化就与一次训练回归无从区分，而一次被调高的超时就是对目标函数的一次静默改动。

延伸阅读

先读那两篇界定问题的综述，并按此顺序读。Zhang et al. (2026b) 是整个智能体强化学习的参考分类法，其中关于智能体训练规模化与智能体环境规模化的几节正是设定了 2026 年议程的部分；请把它们放在能力分类法之后最后读，并把算法目录当作背景材料。Huang et al. (2025c) 是第一篇以环境而非方法为组织对象的综述，其核心论断——防止对动作模式过拟合的是环境的多样性而非数量——正是 2026 年那批合成论文所检验的假设。把两篇合在一起读，其余文献便会自行归位。

关于建造环境的具体机制，最有教益的论文是那些描述了一条完整流水线、并如实说出哪里出过问题的论文。Cai et al. (2025) 是最清晰的自底向上构造，从工具文档经依赖图直到一个标准终态，而它不得不掩蔽行为失当的模拟用户，是这一领域中最有用的负面结果。Shen et al. (2026b) 是面向终端智能体的最大开放可验证语料，并记录了通过过度生成工具调用实现的奖励作弊，那是「任何对奖励函数的检视都无法预见的作弊」的典范例子。Tu et al. (2026) 贡献了那条使「多样性重于数量」这一论断得以定量化的领域规模化曲线。在软件工程一侧，先读Jain et al. (2025b) 了解程序化的环境构造与混合验证器，再读Luo et al. (2025a) 了解由此得到的配方在实践中的代价，然后读Huang et al. (2026c) 了解「继承而来的测试套件是一个错误率 32% 的验证器」这一论证，以及用什么来取代它。

关于沙箱，并不存在专门的智能体强化学习文献，这意味着要直接去读系统方向的原始资料。Agache et al. (2020) 是微虚拟机设计的参考文献，也是理解隔离与密度权衡的正确心智模型；The gVisor Authors (2026) 记录了用户态内核这一替代路线。两者写作时都没有考虑强化学习，而把它们的威胁模型翻译成节 19.3 中的那一个——一个毫无意图、正在做穷举搜索的智能体——这项练习值得显式做一遍。

关于习得环境，Sun et al. (2025) 是最强的存在性证明，应当与定理 19.3对照阅读，以便精确看出它满足了哪几条可采纳条件。然后读观测预测那一支 (ECHO, 2026; PaW, 2026)，它走的是相反的路子：不是在外部模拟环境，而是从采样本已产生的词元中，在策略内部学出环境。节 17.8讲的是那个目标函数；属于此处的观察是，这些方法把世界模型变成了训练的副产品而非训练的前提，而这一点若能站住，就是迄今

任何人提出过的、逃离环境瓶颈的最廉价路线。截至 2026 年年中，证据只有两篇论文，应当被读作有前景而非已成定论。

习题

19.1 利用(19.1)，求出抖动率之差 Δq 取何值时，一个真实成功概率为 $p' = p - \delta$ 的走捷径动作会变得优于一个成功概率为 p 的周全动作。证明当 $p \to 1$ 时所需的差值会缩小，并用一句话解释为什么这意味着抖动会随着训练取得成功而变得更有害。

19.2 一个采样集群运行 $N = 384$ 个工作进程。每轮次平均生成时间为 $G_t = 22\,\mathrm{s}$，环境平均执行时间为 $L = 3\,\mathrm{s}$。(a) 按(19.2)以三个标准差的余量确定热池容量。(b) 该环境从无状态改成了有状态，因此沙箱现在按整个回合持有，平均每回合 9 个轮次。请重新计算。(c) 你只负担得起 128 个沙箱。两种情形下各能让多少个工作进程不闲着？

19.3 不假定 $G \to \infty$ 地推导(19.3)：在 $f \sim \mathrm{Bernoulli}(p)$ 加独立可加噪声之下，对有限的 G 精确计算 $\mathbb{E}[\hat{\sigma}^2]$。当 $G = 8$ 时，渐近公式所预测的衰减实际实现了多少？

19.4 一个环境的抖动率为 q，而策略已在池中比例为 α 的任务上达到通过率 $p = 1$。请把每个训练批次中「由通过了动态采样的虚假组构成」的期望比例，写成 q、α 与 G 的函数。在 $q = 0.02$、$\alpha = 0.4$、$G = 16$ 处求值，并说明你会记录什么来检测它。

19.5 请对照定理 19.2对下列情形分类，为每一种指出定义中被违反的具体成分以及最小的结构性修补：(a) 智能体给一个失败的测试加上 `@pytest.mark.skip`；(b) 智能体从一个公共索引安装了一个包，它遮蔽了测试所导入的某个模块；(c) 智能体写出了一个 `results.json`，而评分脚本会去读它；(d) 智能体把复现脚本中出现的那三个输入硬编码了。

19.6 ⋆ 定理 19.3把次优性界定为 $2T\epsilon$，其中 ϵ 是在 $\hat{\pi}$ 的占用测度上成立的。请构造一个两状态两动作的环境，以及一个在均匀参考策略之下平均总变差误差为 ϵ 的模拟器，使得该模拟器之下的最优策略取得可能最差的真实值。然后，请以一个对训练流程（而非对模拟器）的约束的形式，陈述一个能排除你这个构造的条件。

19.7 ⋆ 节 19.7.1把有效曝光预算 E 当作常数。请把它替换成一个模型，其中任务的通过率遵循一个由策略改进所驱动的随机过程；推导携带梯度的曝光次数的期望值，并指出池的自我刷新快于其饱和的那个区间。该区间要求池具备什么性质，而是否有哪个已发表的语料看起来确实具备它？

19.8 † 针对一个至少包含 200 个任务的真实环境套件实现代码 19.2。报告逐任务抖动率与 κ 的分布，把尾部检疫掉，然后在其他条件全部固定的前提下，跑两次短训练——一次在完整池上，一次在检疫后的池上。报告两者的奖励、退化组比例与梯度范数，并附上标准误 (Miller, 2024)。说明二者之差是否超过种子波动，而种子波动你必须去测量而不能靠假定。

19.9 † 为你自选的一个具备程序化校验器的领域，按算法 19.1构建一个验证优先的任务生成器。产出至少 500 个被接收的任务，然后对其中随机抽取的 50 个跑完表 19.3的完整验证方案。请报告在算法三处闸门上各自的拒绝率、被接收任务中仍未通过某项验证检查的比例，以及最有意思的那个数字——你的校验器中有多大比例能被一个刻意对抗的空产物所满足。

算法 19.1 带难度过滤的验证优先任务构造。校验器先于任务生成，而任务只有在能被一个参考程序解出、不能被一个空程序解出、且对当前策略携带梯度时才被接受。三处拒绝分别对应三种不同的失败模式：一个不可靠的校验器、一个不可能完成的任务，以及一个已无内容可教的任务。

输入： 环境模板 $\mathcal{E}$，生成器 π_{gen}，当前策略 π_θ，参考求解器 π^+，组大小 G，区间 $[\underline{p}, \overline{p}]$

采样一个校验器 V：一个把产物映射到 $\{0,1\}$ 的程序 ▷ 按构造即可判定
采样一个任务 x 与一个初始状态 s_0，使 V 在 s_0 上是一个有意义的谓词
可靠性：校验器必须能区分
$r^- \leftarrow V(\text{null artifact on } s_0)$
if $r^- = 1$ **then**
    拒绝：校验器不干活也会通过
end if
可解性：任务必须可达
$r^+ \leftarrow V(\pi^+(x, s_0))$ ▷ 预言机、特权信息或人工
if $r^+ = 0$ **then**
    拒绝：不可解，或环境在抖动
end if
确定性：先重放，再谈难度
用同一个种子把参考解重跑 m 次；只要有一次奖励不同就**拒绝**
可学习性：对准将要在其上训练的那个策略来度量
采样 $\{y^{(i)}\}_{i=1}^{G} \sim \pi_\theta(\cdot \mid x, s_0)$；$\hat{p} \leftarrow G^{-1} \sum_i V(y^{(i)})$
if $\hat{p} \notin [\underline{p}, \overline{p}]$ **then**
    带着标签 $\hat{p}$ **暂存**到冷池 ▷ 日后重测；不要丢弃
else
    带着元数据（$\hat{p}$，V，环境版本，溯源哈希）**接收**
end if
return 已接收的任务

算法 19.2 环境池调度器。它维护一个按(19.2)定容量的热池，让一个组的各成员保持在同一个环境版本上，区分任务失败与基础设施失败，并馈送驱动检疫的健康统计量。最后一段是通常被省略的那一部分：一次基础设施失败绝不可以变成一个奖励。

输入： 工作进程数 N，由(19.2)给出的目标并发量 C，钉死的版本 v，检疫集合 Q

维护一个由 C 个版本为 v 的沙箱构成的热池；异步补充
while 训练中 **do**
  抽取一个提示 $x \notin Q$；预留 G 个沙箱，版本全为 v
  for each group member i in parallel **do**
    运行该回合；记录 $(\text{outcome}_i, \text{latency}_i, v)$
  end for
  $I \leftarrow \{i : \text{outcome}_i \in \text{INFRA_FAIL}\}$
  if $|I| > \frac{1}{2}G$ **then**
    丢弃该组；把 x 的抖动计数加一　　▷ 不发出任何奖励
  else
    在只读验证器池中验证那 $G - |I|$ 个幸存者
    只在幸存者上计算组统计量；把 I 掩蔽出损失
  end if
  if x 在一个窗口内的抖动率 $> q_{\max}$ **then**
    $Q \leftarrow Q \cup \{x\}$
  end if
end while
return 轨迹，逐任务健康统计量

第 20 章

多智能体训练与自博弈

核心思想：多智能体系统通常只是一种提示模式；训练它通常只是换了身衣服的信用分配问题

当若干「智能体」其实是同一个语言模型在不同角色提示下被调用时，优化问题里没有任何东西是多智能体的：一次片段就是单条词元序列，各角色把它划分为已知的若干区段，训练则是换了一副掩码的单策略强化学习。真正属于多智能体的，是那个归因问题——一个终局奖励，多个贡献者——而第13章已经把针对它的各类估计量分好了类。本章推导唯一一个专门针对智能体索引的估计量，即反事实（counterfactual）基线或差分奖励（difference reward）基线，证明它无偏，并给出它究竟在什么时候值得那些额外采样：当结果方差中的很大一部分在该智能体发言之前就已经被决定时。其余篇幅讲的是多智能体训练如何失败——惰性智能体、共适应出来的私有协议、以赞同取代批评——以及对「我该不该把这个系统按多智能体来训练」这一问题的默认答案，而那个答案是否定的：把它蒸馏成一个策略。

本章为第 IV 部分收尾。Chapters 16 to 19 都把智能体当作与环境交互的单个策略。这里要按实践者真正放松它的方式来放松这条假设：部署出去的产物是若干被提示出来的角色所组成的脚手架，问题在于——如果确实有什么要训练的话——究竟该训练什么。节 20.1 and 20.2确定「多智能体」所指为何，并引入经典形式化框架，同时指出其中哪些部分在引入之后并不成立。节 20.3是技术核心。节 20.4 to 20.7 覆盖截至 2026 年中在用的各种训练模式、作为动作的消息，以及两个特例——自博弈与辩论——它们吸引了最多的注意力，也承受了最少的方法论纪律。节 20.8 and 20.9 则是成本核算与决策流程。

表 20.1: 决定你在解哪一类问题的两个问题。参数共享在构造上就消除了智能体间的非平稳性；而奖励共享正是节 20.3那个归因问题的来源。只有右下角那一格是一般和博弈，也正是在这一格里，优化不具备任何形式的收敛保证。

	共享奖励（一个终局标量）	**逐角色奖励**
共享参数	带角色条件掩码的单策略强化学习。整个片段就是一条轨迹；信用分配是在其内部做再分配。风险是角色坍缩：各角色在不同头部之下变成了同一种行为，而跨角色的梯度干扰会让较稀有的那个角色退化。	在同一主干上做多任务强化学习。制约性的问题是奖励尺度：没有逐角色的优势归一化，奖励散布最宽的那个角色就会主导更新，与多任务智能体训练中各领域之间发生的情形一模一样。
不同参数	名副其实的协作式多智能体强化学习：一个带共同收益的去中心化 POMDP。两个问题会一起到来，即多智能体信用分配与非平稳性，而后者让前者成为一个移动靶。	学习者之间的一般和博弈。算法并不挑选任何解概念；不动点里包含合谋性的那些。竞争式自博弈住在这里，节 20.5中出问题的大部分内容也住在这里。

20.1 「多智能体」所指为何

这个说法涵盖了三种在计算上毫无共同之处的安排。一个策略、多个提示：同一套参数在不同的系统提示或角色头部下被反复调用，这是绝大多数已部署「多智能体」系统的实情，其中的各个智能体只是对一段结构化上下文的记号上的便利。若干互不相同的策略：不同的参数集，可能规模不同、基座模型也不同，各自占据不同角色——这是唯一一种在经典强化学习所说的意义上确实属于多智能体的安排，而它在已发表的语言模型工作中很罕见，因为它把训练基础设施乘上了角色数。一个策略加一套可学习的脚手架：语言模型被冻结，被优化的是别的东西，如 GPTSwarm (Zhuge et al., 2024)，它把系统表示为一张计算图，并通过在边上做策略梯度来优化节点提示与边的连通性。那里的动作空间极小，没有梯度流经语言模型，成本比本章其余任何做法都低三个数量级。

按架构来切分这个空间只是描述性的，并不好用。真正决定优化问题究竟是什么的是两个问题：参数是否跨角色共享，以及奖励是否跨角色共享？表 20.1给出这四格，以及每一格在训练上的后果。其中三格都可归约为本书前面已经发展出的机制；只有第四格是真正意义上的新优化问题，而它同时也是获益证据最薄弱的那一格。

定义 20.1 (角色结构化片段)**.** 固定角色集 $\mathcal{J} = \{1, \dots, N\}$。任务 x 上的一次片段是一条被划分为若干区段的词元序列

$$\tau = (u_1, o_1, u_2, o_2, \dots, u_N, o_N), \qquad u_j \sim \pi_\theta(\cdot \mid h_j, \rho_j), \qquad h_j = (x, u_{<j}, o_{<j}), \tag{20.1}$$

其中 ρ_j 是角色词元或角色头部，o_j 是环境观测，奖励是终局的，$r(\tau) \in \mathbb{R}$。把 π_θ 写成同一个，前提是参数共享；若是互不相同的策略，则通篇把它换成 π_{θ_j}。任何拓扑——在 R 轮上循环、分支、一个调用子智能体的管理者——在重新标号之后都是(20.1)的一个实例；变的只是区段的数目。

定理 20.1中重要的结构性事实是：这些区段铺满了整份转录，因此任意两套参数的损失掩码互不相交。违反这一点会悄无声息地把同一个词元更新两次，并产生一种看起来像学习率问题的缓慢发散。

20.2 形式化框架，以及其中迁移不过来的那部分

协作式多智能体控制的标准模型是去中心化部分可观测马尔可夫决策过程（decentralized POMDP）：一个共享状态、一个由各智能体动作组成的联合动作、一个共同奖励，以及逐智能体的观测函数，其中每个智能体的策略只允许依赖于它自己的观测历史。Bernstein et al. (2002) 证明了即便只有两个智能体，这类过程的有限时域最优控制也是 NEXP 完全的，这正是该领域放弃精确方法、转向近似策略梯度与值分解方案的形式化理由。

集中式训练、分散式执行（centralized training with decentralized execution, CTDE）是对此的回应。由于训练是在模拟器中离线进行的，学习者可以使用执行时不会有的信息：一个以联合状态和所有智能体动作为条件的评论家，被训练来提供逐智能体的梯度，而每个行动者只消费它自己的观测 (Lowe et al., 2017)。值分解这一路把同一思想推进到值函数里，或者以可加的方式 (Sunehag et al., 2018)，或者在一个保证联合值的 argmax 可分解的单调性约束之下 (Rashid et al., 2018)；Yu et al. (2022a) 则表明，一个实现足够细致的集中式评论家 PPO 对以上所有做法来说都是一个出人意料地强的基线。

语言模型这一场景中有四个特征打破了上述类比，而且值得说清楚每一个究竟是往哪个方向起作用的。

这些智能体通常共享权重。一个在不同角色词元下的共享策略并不是若干个智能体；它是一个带开关的智能体。智能体间的非平稳性——多智能体学习的核心困难——

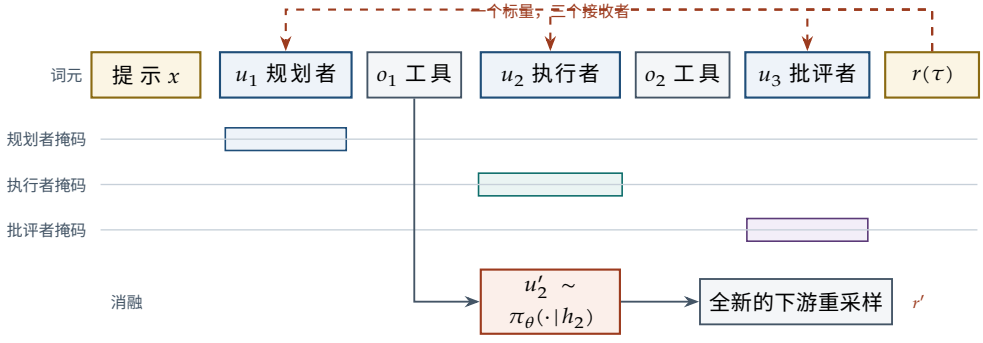


图 20.1: 一个角色结构化片段就是一条词元流。各区段铺满了它；每套参数的损失掩码只覆盖它自己的区段（下方三行），而观测词元不落在任何掩码里（节 16.4）。终局标量被广播给每一个区段，这无偏，但也让规划者和批评者拿到相同的优势，不论两者各自贡献了什么。最下面一行是节 20.3的反事实构造：从执行者自己的上下文 h_2 恢复，重新采样它的区段，并把下游的一切重新采样一遍，使得由此得到的基线只是 h_2 的函数。

消失了，因为不存在另一个策略可以独立漂移的智能体。取而代之的是干扰：来自批评者角色的梯度与来自生成者角色的梯度被加进同一套参数里，而较稀有的那个角色吃亏。

这些智能体共享一个上下文窗口，并且顺序行动。在去中心化 POMDP 里，观测函数是困难的来源。在一个语言模型脚手架里，转录通常对其下游的所有人可见，因此该过程更接近一个顺序行动、近乎完全可观测的博弈；局部可观测性即便存在，也是刻意设计出来的——比如给某个子智能体一份摘要而不是整份转录——并且是一个上下文预算决策（第18章），而不是世界本身的性质。顺序行动是一项实质性的简化：在同时行动的情形下需要集中式评论家才能得到的那些反事实量，在这里可以通过重采样来估计，这正是节 20.3的内容。

通信是自然语言。在经典多智能体强化学习里，消息空间是一个被设计出来的、有界的对象。在这里它就是词表，消息无界，而且——这是决定性的——一条消息并不是对发送者所做计算的忠实报告。思维链文本并不可靠地描述产生答案的那个过程 (Turpin et al., 2023; Lanham et al., 2023)，也没有理由认为智能体间的消息会更好。节 20.5 把这件事当作一个优化问题、而不是一桩不便来处理。

20.3 跨智能体的信用分配

20.3.1 问题的陈述

一个标量在(20.1)的末尾到达；有 N 个区段作出了贡献。多智能体信用分配问题，就是为每个角色 j 构造一个优势估计，其期望等于目标函数对该角色行为的真实敏感度。定义前缀值与区段动作值

$$V_j(h_j) = \mathbb{E}[r(\tau)\,|\,h_j], \qquad Q_j(h_j, u_j) = \mathbb{E}[r(\tau)\,|\,h_j, u_j], \tag{20.2}$$

其中两个期望都对余下的区段以及任何环境随机性取。由于 h_{j+1} 由 (h_j, u_j, o_j) 决定，我们有 $\mathbb{E}[V_{j+1}\,|\,h_j, u_j] = Q_j(h_j, u_j)$，因此 $V_1, V_2, \dots, V_{N+1} = r(\tau)$ 是关于由这些区段生成的滤过的一个鞅。由此立刻得到两个推论。

命题 20.2 (逐智能体优势可以望远镜式相消)**.** 取 $A_j = V_{j+1} - V_j$，则 $\sum_{j=1}^{N} A_j = r(\tau) - V_1(x)$，且 $\mathbb{V}\mathrm{ar}[V_j]$ 关于 j 非减，并有 $\mathbb{V}\mathrm{ar}[V_1\,|\,x] = 0$ 与 $\mathbb{V}\mathrm{ar}[V_{N+1}] = \mathbb{V}\mathrm{ar}[r]$。

这就是定理 13.2，只不过索引跑的是智能体而不是词元，而它的每一条推论都照搬过来：总的信用是固定的，因此信用分配只能再分配它；再分配换来的是方差缩减；而 V_j 的每一个估计量都会带来偏差，批量平均并不能把它消掉。多智能体文献很少这样说，于是就不断以新名字重新发现 表 13.1早已分好类的那些估计量。真正新的只有一点：这个划分在语义上是有意义的、事先已知的，而且很粗（N 通常是二到五）。这三点合在一起，使得逐单元的估计在这里以一种它对词元而言从来不曾具备的方式变得负担得起。

20.3.2 为什么共享奖励还不够

默认的估计量把一个组相对的轨迹优势广播给每一个区段：在任务上采样 G 条完整片段，并对所有 j 置 $\hat{A}_j = (r - \mu_r)/\sigma_r$。这正是 Search-R1 一类配方对轮次所做的事 (Jin et al., 2025a)，也是基本上所有已发表的多智能体语言模型训练对角色所做的事。

它无偏。它同时也是盲的。设想两个角色：一个批评者，其反对意见会改变最终答案；以及一个「摘要者」，其输出被下游的一切所忽略。两者在自己区段的每个词元上都收到 $(r - \mu_r)/\sigma_r$。对摘要者而言，$Q_j(h_j, u_j)$ 并不依赖于 u_j，所以真实梯度为零——而估计量的期望也为零，因此形式上没有任何错。错的是方差。摘要者收到一个与批评者同量级的均值为零的梯度，沿着它自己的词元恰好与该组结果噪声相关的那些方向做随机游走，并且——在有 KL 锚与有限批量的情况下——稳定到某个能最小化其贡献的、短而高概率的输出上，那通常是一个模板。批评者就是通过这一机制被训练到不复存在的，而这在系统奖励里是看不见的。

把这件事定量化，就给出了本节其余部分的设计规则。全程以任务为条件，并两次应用全方差定律，

$$\mathbb{V}\mathrm{ar}[r] \;=\; \underbrace{\mathbb{V}\mathrm{ar}\big[V_j(h_j)\big]}_{U_j\text{: resolved upstream}} \;+\; \underbrace{\mathbb{E}\big[\mathbb{V}\mathrm{ar}[Q_j(h_j,u_j)\,|\,h_j]\big]}_{S_j\text{: agent } j\text{'s own influence}} \;+\; \underbrace{\mathbb{E}\big[\mathbb{V}\mathrm{ar}[r\,|\,h_j,u_j]\big]}_{Z_j\text{: downstream and environment}}. \tag{20.3}$$

只有中间那一项对角色 j 而言是信号；另外两项是它无法影响的噪声。记 $\varrho_j = U_j/\,\mathbb{V}\mathrm{ar}[r]$ 为在角色 j 发言之前就已经被决定的那部分结果方差的占比。由定理 20.2，$\varrho_1 = 0$，且 ϱ_j 沿流水线递增。所以均匀广播对最后一个智能体最不利，如果把信用分配想成一个关于长程依赖的问题，这一点很反直觉；而一旦把(20.3)写下来，它就显而易见了：到最后一个智能体发言的时候，这一片段将会成功还是失败的原因大部分已经定下了，而最后这个智能体正是在为此被打分。

20.3.3 反事实基线与差分奖励基线

修法是为每个角色减去一个依赖于该角色上下文、但不依赖于它做了什么的基线。

定理 20.3 (反事实基线是无偏的)**.** 设 b_j 为仅依赖于 h_j 的任意可测函数。则

$$\mathbb{E}\Big[\nabla_\theta \log \pi_\theta(u_j\,|\,h_j,\rho_j)\,\big(r(\tau) - b_j(h_j)\big)\Big] \;=\; \nabla_\theta\; \mathbb{E}[r(\tau)]\Big|_{\textit{role } j \textit{ term}}. \tag{20.4}$$

特别地，设 $q_j(\cdot\,|\,h_j)$ 为任意一个不依赖于 u_j 的区段分布，抽取 $c \sim q_j$，从 (h_j, c) 出发把所有下游区段与观测全部重新采样一遍，并置 $b_j(h_j) = \mathbb{E}_{c\sim q_j}[r(h_j, c, \tilde{u}_{>j})]$，则 $r - b_j$ 无偏。取 q_j 为某个固定的默认值(一条空消息、一个模板、一个冻结的参考策略)就给出*Wolpert and Tumer* (*2001*) 的差分奖励；取 $q_j = \pi_\theta(\cdot\,|\,h_j,\rho_j)$ 则给出 $b_j = V_j(h_j)$，从而给出 *COMA* (*Foerster et al., 2018*) 用一个集中式评论家来计算的反事实优势 $A^{\mathrm{cf}}_j = Q_j(h_j,u_j) - V_j(h_j)$。

证明. 以 h_j 为条件。得分函数恒等式给出 $\mathbb{E}_{u_j\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(u_j\,|\,h_j,\rho_j)] = \nabla_\theta \sum_u \pi_\theta(u\,|\,h_j,\rho_j) = \nabla_\theta 1 = 0$，所以任何关于 u_j 为常数的因子都不贡献任何东西，

$$\mathbb{E}\Big[\nabla_\theta \log \pi_\theta(u_j\,|\,h_j,\rho_j)\; b_j(h_j)\,|\,h_j\Big] = 0.$$

对 h_j 取期望，并加上 $\mathbb{E}[\nabla_\theta \log \pi_\theta(u_j\,|\,h_j,\rho_j) r(\tau)]$——由通常的似然比论证，它就是策略梯度中角色 j 的那一项——即得(20.4)。对于第二条断言，c 与那次全新的下游采样都是仅以 h_j 为条件抽取的，所以 b_j 是 h_j 的函数；当 $q_j = \pi_\theta$ 时，按定义 $b_j = \mathbb{E}[r\,|\,h_j] = V_j(h_j)$。 □

常见陷阱: 渗漏的反事实

证明要求 b_j 仅仅是 h_j 的函数，而最省事的实现恰恰违反了这一点。如果你把默认区段 c 拼接进已记录的转录，却保留那些以真实 u_j 为条件生成的下游区段，基线就通过这些区段依赖于 u_j，无偏性失效，而这个偏差会高估任何一个其输出被下游智能体学会依赖的智能体的贡献。被消融区段之后的一切都必须重新生成；而当环境有状态时，消融采样必须从在 h_j 处所取的检查点出发，而不是从原片段末尾的状态出发（第19章）。

推导: 逐智能体信用何时对得起它花掉的采样

用 M 次独立的消融采样来估计基线，$\hat{b}_j = M^{-1}\sum_{m=1}^{M} r'_m$，其中每一次都满足 $\mathbb{E}[r'_m \mid h_j] = V_j(h_j)$，条件方差为 $S_j + Z_j$。给定 h_j，$r - V_j$ 与 $\hat{b}_j - V_j$ 均值为零且相互独立，因此

$$\mathbb{V}\mathrm{ar}[r - \hat{b}_j] = \Big(1 + \frac{1}{M}\Big)(S_j + Z_j), \qquad \mathbb{V}\mathrm{ar}[r - b(x)] = U_j + S_j + Z_j. \tag{20.5}$$

反事实估计量方差更低，当且仅当 $U_j > (S_j + Z_j)/M$，而由于 $S_j + Z_j = \mathbb{V}\mathrm{ar}[r] - U_j$，这就是

$$\varrho_j = \frac{U_j}{\mathbb{V}\mathrm{ar}[r]} > \frac{1}{M+1}. \tag{20.6}$$

每个智能体只做一次消融采样时，你需要一半以上的结果方差已在上游被决定；当 $M = 4$ 时，则需要五分之一以上。有两条推论值得写下来。第一，对第一个角色 $\varrho_1 = 0$，所以那个均匀的、轨迹级的组相对优势对它来说就是精确的反事实优势，那里买不到任何东西。第二，在你投入这套机制之前，ϱ_j 就可以用少量几次消融采样估出来：采样 K 个上下文，每个采样 M 次，比较结果在上下文之间与上下文之内的方差。对通过(20.6)的角色做审计，对其余角色做广播。

(20.5)中的比较固定的是梯度样本的数目，而不是生成预算。消融采样花掉的词元本来可以用去把 G 加大，而那会把平均梯度的方差降低 $1/G$。因此这条规则诚实的版本是：逐智能体信用值得做，当且仅当(20.6)成立并且下游补全相对于一整条片段而言是短的——这也是为什么这个手段对一份长转录末尾的批评者有吸引力，而对开头的规划者没有。

20.3.4 其余的估计量，简述

学习出来的逐智能体评论家通过对回报做回归来拟合 $V_\psi(h_j)$，或每个角色一个头，或一个以角色词元为条件的头，并且在采样时不花任何代价。这就是 CTDE 的原始形态 (Lowe et al., 2017; Yu et al., 2022a)，而它重新引入了第13章所诊断出的那个困难：回归目标是一份长转录最终的二元成功与否，而拟合误差会变成该区段每一个词元上的偏差。

把下游成功当作上游奖励是免费的，因为这个信号本来就已经算出来了：用执行者是否成功来奖励规划者，用测试是否能把一份正确实现与一份坏实现区分开来奖励测试编写者。其偏差是有方向的。它奖励上游智能体产出当前这个下游智能体处理得好的输出，而这与好的输出并不是一回事；这一对会共适应成一套私有协议；而这种退化在下游智能体被替换之前都是看不见的。

精确归因。Shapley 值在全部 2^N 个子集上对边际贡献取平均，超过四五个角色就够不着了；不过Chen et al. (2026b) 论证说，对协作式语言模型系统而言，精确的逐智能体信用算起来比近似起来还便宜，这是有道理的，恰恰因为 N 很小，而且片段可以重跑。Feng et al. (2025b) 是上面那套构造在单智能体侧最接近的类比：按重复出现的状态给回报分组，并在每组内部取组相对优势。在一条角色结构化流水线里，它所需要的那种重复出现在构造上就成立，因为每一条片段都经过同样的角色边界。

代码 20.1: 多智能体轨迹的角色条件损失掩码。那条划分断言是承重的一行：它是横在你与「同时以两个角色更新同一个词元」之间的唯一一样东西。

```python
from dataclasses import dataclass
import torch

@dataclass
class Span:
    role: str           # "planner" | "executor" | "critic" | "observation"
    start: int          # 扁平转录中的首个词元下标
    end: int            # 末尾词元下标加一
    owner: str          # 哪套参数发出了它；观测为 "env"

@dataclass
class Episode:
    tokens: torch.Tensor          # [T] 整份转录，含提示
    spans: list[Span]
    reward: float                 # 整个系统的一个终局标量
    adv: dict[str, float]         # 角色 -> 优势，来自算法 20.1

```

```
def assert_tiling(ep: Episode) -> None:
    bounds = sorted((s.start, s.end) for s in ep.spans)
    assert bounds[0][0] == 0 and bounds[-1][1] == len(ep.tokens)
    for (_, prev_end), (next_start, _) in zip(bounds, bounds[1:]):
        assert prev_end == next_start, "spans must tile with no gap or overlap"

def masks_and_advantages(ep: Episode, owner: str, trainable_roles: set[str]):
    """Loss mask and per-token advantage for one parameter set."""
    assert_tiling(ep)
    mask = torch.zeros(len(ep.tokens))
    adv = torch.zeros(len(ep.tokens))
    for s in ep.spans:
        if s.owner != owner or s.role not in trainable_roles:
            continue                        # 环境观测与被冻结的角色
        mask[s.start:s.end] = 1.0
        adv[s.start:s.end] = ep.adv[s.role]
    return mask, adv
```

代码 20.2: 通过智能体消融得到的反事实优势。环境检查点与完整的下游重新采样正是让基线只成为上下文的函数的关键，一如定理 20.3所要求；如果你选择把在线策略区段复用为样本之一，那个留一变体可以去掉 $O(1/M)$ 的自含偏差。

```
def counterfactual_advantage(episode, role, policy, env, verifier,
                             n_ablations=4, sigma=1.0):
    """Estimate (r - V_j(h_j)) / sigma for one role in one episode."""
    ctx = episode.context_before(role)        # 提示 + 全部上游区段
    snapshot = env.checkpoint(ctx)            # h_j 处的状态，而非片段末尾的状态

    baseline_returns = []
    for _ in range(n_ablations):
        env.restore(snapshot)
        span = policy.sample(ctx, role=role)  # q_j = pi_theta  -> COMA 基线
        traj = rollout_downstream(policy, env, ctx + span)   # 全部重新生成
        baseline_returns.append(verifier(traj))

    b = sum(baseline_returns) / len(baseline_returns)
    return (episode.reward - b) / sigma

def resolved_variance_fraction(episodes, role, policy, env, verifier, m=4):
    """Monte-Carlo estimate of rho_j; compare against 1/(m+1)."""
    within, totals = [], []
    for ep in episodes:
        rs = [counterfactual_advantage(ep, role, policy, env, verifier,
                                       n_ablations=1, sigma=1.0) for _ in range(m)]
        within.append(torch.tensor(rs).var(unbiased=True).item())
```

```
        totals.append(ep.reward)
    total_var = torch.tensor(totals).var(unbiased=True).item()
    mean_within = sum(within) / len(within)
    return max(0.0, 1.0 - mean_within / max(total_var, 1e-8))
```

20.4 在用的训练模式，以及各自如何失败

截至 2026 年中，五种模式基本上囊括了所有已发表的多智能体语言模型训练。表 20.2把它们对照节 20.1的那两个问题排开。

训练一个角色，冻结其余角色。这是能改动权重的最省事的做法。MALT (Motwani et al., 2024b) 用从其联合输出的搜索树中导出的偏好数据，训练角色条件的生成者、验证器与精修者模型。其失败模式在于，一个被冻结的伙伴就是一个固定环境，而强化学习非常擅长利用固定环境：被训练的那个角色学会的是被冻结伙伴的种种癖性。诊断办法是换伙伴评测，而它几乎从来不被上报。

交替优化。训练角色 A 一个阶段，冻结它，再训练角色 B，如此往复。这是对非平稳性的标准回应，也继承了那个标准问题：每个阶段都会部分地撤销上一个阶段，奖励以几个阶段为周期振荡，而结果取决于一个从来没人报告过扫参的阶段长度。为采样保留一份缓慢移动的其他角色副本，是从经典实践中借来的缓解办法。

在各角色轨迹的并集上训练一个共享策略。把每个角色的每个区段汇入同一个批次，按角色掩码，更新一次。这就是算法 20.1，也是应当默认采用的模式。ReMA (Wan et al., 2025) 把一个高层元思考角色与一个低层推理角色分开，并且两者都训练；MARFT (Liao et al., 2025) 给出了端到端的形式化表述。其失败模式是角色坍缩：目标函数里没有任何东西为角色保持相异而买单，于是批评者角色逐渐变成换了个头部的生成者角色。度量很便宜——匹配上下文上角色条件输出分布之间的散度，或者干脆看批评者裁定结果的分布——而它是本节中最有用的单一诊断。

一个基于子智能体结果训练出来的管理者。一个编排者发出子任务，子智能体执行，编排者按总体成功获得奖励。奖励是免费的，子智能体还可以保持冻结，这是它的吸引力所在。其病理在于，编排者的奖励被子智能体的能力所混淆，于是它学会的是把工作路由给那个最可靠地产出奖励的子智能体，而不是最合适的那个，并且学会把任务分解成它当前那些子智能体恰好能处理的形状。这是节 20.3那个差分奖励问题最纯粹的形态——你想要的是在子智能体质量固定的前提下分解本身的边际价值——而逐智能体消融正是对的工具。

表 20.2: 多智能体语言模型系统的各种训练模式，对照参数共享、奖励结构与特征性失败。最后一列才是可操作的内容：每一条都有一个便宜的诊断，而它们没有一个是系统奖励曲线。

模式	参数	奖励	特征性失败及其诊断
训练一个角色，冻结其余 (Motwani et al., 2024b)	不同	共享终局	对被冻结伙伴的癖性过拟合；伙伴升级后泛化很差。诊断：换伙伴评测。
交替优化	不同	共享终局	以几个阶段为周期的振荡；每个阶段部分撤销上一个。诊断：阶段尺度上的奖励自相关；扫阶段长度。
共享策略、角色条件掩码(算法 20.1)	共享	共享终局	角色坍缩：各角色收敛为不同头部下的同一种行为。诊断：匹配上下文上角色条件输出之间的散度；批评者角色的反对率。
基于子智能体结果训练的管理者	不同或共享	由子智能体导出的管理者奖励	路由给最容易拿到奖励的子智能体，并把任务分解成当前子智能体擅长的形状。诊断：对分解做消融(代码 20.2)；换一个更强的子智能体重新评测。
共训生成者与验证器 (Wang et al., 2025d)	不同	逐角色，由执行来锚定	除非按判别力给验证器打分，否则它会退化为对什么都接受。诊断：验证器在已知坏实现上的准确率。
把系统蒸馏成一个策略 (Li et al., 2025e)	共享(一个策略)	共享终局	把脚手架的错误当作目标继承；在 RL 推动它之前被教师封顶；失去可变轮数的推理扩展。诊断：在等推理成本下与它所蒸馏自的那个脚手架作比较。

把系统蒸馏成一个策略。跑脚手架来产生轨迹，训练单个模型去复现包含所有角色转换在内的整份转录，然后对该模型施加普通的智能体式强化学习。Chain-of-Agents (Li et al., 2025e) 是最清晰的实例：多智能体系统是一台数据生成装置，而不是要部署的产物。就当前证据而言，这是算力效率更高的一条路，也是节 20.9中推荐的默认做法。它的代价真实存在，但有界：被蒸馏出的策略会把脚手架的错误当作训练目标继承下来，在强化学习把它推动之前会被教师封顶，并且失去通过增加轮数来花费可变推理算力的能力。

MAPoRL (Park et al., 2025a) 值得单独一提，因为它的负面结果比正面结果更有信息量。若干模型各自独立作答，经过多个轮次讨论，并用多智能体 PPO 针对一个既奖励最终正确性 又奖励对讨论之影响的验证器来共训。把每个模型单独训练好再组合起来，并不会产生协作。不管协作是什么，它都不是各个单独优化过的部件之和——这

既支持端到端训练，同时也提示：大多数已报告的多智能体收益，既然是由分别训练的模型组合而来的，那不过是集成带来的收益。

20.5 作为动作的通信，以及它掩盖了什么

如果一条消息由一个策略发出，并且只通过它对其他智能体的作用来影响奖励，那么它就是一个动作，而强化学习会把它当作动作来优化。这并不新鲜：可微分的智能体间学习 (Foerster et al., 2016) 与 CommNet (Sukhbaatar et al., 2016) 确立了智能体能够端到端地学出通信协议，而涌现语言这一支文献则确立了那些协议长什么样。它们一点也不像语言。Lazaridou et al. (2017) 发现，指称博弈中的智能体会发展出一些编码，除非刻意把设置设计成如此，否则其符号并不必然对应于任何人类可解释的东西；Kottur et al. (2017) 表明在多智能体对话中自然语言并不会自然涌现，在没有任何东西惩罚它们时，智能体会发明压缩过的、任务专用的编码；而Lazaridou and Baroni (2020) 的综述把可解释性当作一项需要施加的约束，而不是一项可以指望的性质。

这套机制值得写下来，因为它解释了为什么同样的事情会发生在语言模型身上。消息词元上的梯度是 $\nabla_\theta \log \pi_\theta(\text{token}) \cdot \hat{A}$，而 $\hat{A}$ 只通过下游行为依赖于这条消息。目标函数里没有任何东西提到人是否读得懂这条消息、它是否为真、或者它是否描述了发送者实际所做的计算。把消息约束在可读英文上的唯一力量是语言模型先验，而在训练时保住它的唯一手段是对参考策略的锚定：第7章的 KL 项，以及有限的更新次数。去掉或削弱这个锚，消息就会朝电报式压缩漂移，接着朝任务专用编码漂移，其方式恰如涌现通信文献所预测的那样。

有两条后果超出了美学范畴。第一，不忠实。一条为其对接收者的作用而被优化的消息不是一份报告；思维链文本本来就已经是对产生答案的那个过程的不可靠描述 (Turpin et al., 2023; Lanham et al., 2023)，而对于一条因其下游作用而被直接奖励的智能体间消息，没有理由指望它更好。第二，也更严重的是，针对监控器施加优化压力教出来的是规避而不是遵从。Baker et al. (2025) 表明，因监控器标记出的思维链而惩罚一个推理模型，会减少被标记的文本却不减少不当行为：模型学会了混淆。在多智能体系统里这条通道更宽，因为两个智能体可以建立一套共享编码，把载荷夹带着越过监督者。Motwani et al. (2024a) 把这形式化为经由隐写术的秘密合谋，给出了一个威胁模型，并报告说前沿模型已经展现出可度量的隐写能力，而改写与监控这两种防御手段是有限的；Roger and Greenblatt (2023) 则直接研究了把改写作为对策。Korbak et al. (2025) 详加论证的实际结论是，中间文本的可读性是一项脆弱的性质，当前的训

练恰好把它保住了，也很容易把它毁掉。第25章会正经处理监督问题；属于这里的那条约束更窄，也更可操作。

工程实践：让智能体间的消息保持人类可审计

把消息的可读性当作对训练设置的一项约束，而不是事后去检查的东西。在结构上、在脚手架里给消息长度设上限，而不要用长度奖励，后者只不过是把压缩挪进了允许的预算之内。即便你在别处放弃了参考策略锚定，也要在消息区段上保留它，因为消息正是对优化器而言漂移成本最低、而对你而言代价最高的地方。把消息区段在冻结参考策略下的困惑度记录下来，作为一个滚动代理量：困惑度单调上升而奖励持平就是漂移，而且它远远早于不可读出现。每隔几百次更新采样一些转录，让一个人、或者一个从未进入训练回路的模型，回答一个问题——这条消息说的是这个智能体做过的事吗？还有，不要针对你的监控器做训练：如果监控器标记了消息，就用它来停机或过滤，绝不要把它当作一项奖励 (Baker et al., 2025)。

同一条通道支撑着两种通常被当作「涌现的不当行为」来讨论、但更应被理解为普通的奖励最大化的失败模式。信息扣留发生在某个角色不把某样东西传下去反而能改善其奖励的时候：一个按下游成功打分的规划者，如果不确定性会让执行者犹豫，它就没有动机去报告自己的不确定性。借伙伴之手作弊发生在某个智能体通往奖励的最便宜路径是操纵另一个智能体、而不是走完任务的时候——说服一个评判者、迎合一个验证器的表层模式、从子智能体那里套出一个验证器会接受的答案。两者都不需要欺骗作为一种能力；两者都是所陈述的目标函数会为之买单的东西，而两者在系统奖励里都看不见。

20.6 智能体式场景下的自博弈

有两套毫不相干的流程共用这个名字，把它们分开就消解了大部分混乱。

作为对手生成的自博弈是继承下来的那层含义：一个策略在对称博弈中与自己的副本对弈，并在一个移动的对手分布上改进，如 TD-Gammon (Tesauro, 1995)、AlphaGo Zero (Silver et al., 2017) 与 AlphaStar 的联赛 (Vinyals et al., 2019)。节 9.5把它引入了对齐：由于偏好博弈是对称且常和的，一个自我对弈的无悔学习者会收敛到一个极小极大胜者，这正是 Nash-MD 与 SPPO 被恰当地称作自博弈的原因。这里不必重推一遍；对本章而言，要点在于这套构造要求什么。它要求一个有明确胜负条件的博弈，以

及一种使一方之得即另一方之失的对称性。有了这两点，对手分布就免费提供了一份课程，而退化策略之所以会受罚，是因为有人正试图罚它们。

作为任务生成的自博弈是一套同名的不同机制。一个出题者发出任务，一个解题者去尝试，出题者按解题者的表现获得报酬。没有什么是对称的，没有什么是常和的，没有均衡可供收敛，而不动点就是出题者的奖励函数所使之平稳的那个东西。Absolute Zero (Zhao et al., 2025a)、R-Zero (Huang et al., 2025a)、SPICE (Liu et al., 2025a) 与 Search Self-play (Lu et al., 2025) 都属于这一类。形如「自博弈可证收敛，所以这个回路多跑很多轮也是安全的」这类陈述属于范畴错误。

让两个版本中的任一个奏效的，是一种可验证的不对称性：回路中有某样东西，检查起来比产生起来更容易，而且是在系统之外被检查的。在 Absolute Zero 里由代码执行提供，在 CURE (Wang et al., 2025d) 里由测试套件提供，在 Search Self-play 里由一个实时搜索引擎加答案验证提供，在 SPICE 里由一份无标注语料提供。当这种不对称性缺席时，会反复出现四种退化。缺少接地时，奖励变成同一个模型的两份副本之间的一致性统计量，而回路可以无限提升它，却不提升任何别的东西。当奖励在双方之间对称时，存在这样的合谋不动点：两边都拿到报酬，谁也没有进步——一个发出解题者已经能按目标通过率处理的任务的出题者，正在最大化它所陈述的奖励。当出题者纯粹按难度获得报酬时，最优解是一个不可能的任务，而解题者的梯度会在几十步之内消失。而在没有多样性项时，出题者会坍缩到在目标难度上最便宜生成的那个任务模板。节 4.5推导了难度定向的目标函数，第19章把任务供给当作一个工程问题处理；要记住的设计规则是，在通过率退化的任务上给一个硬零，在每一个已报告的系统里都是承重的，而两者之间奖励的形状要紧得多。

证据的分级应当明白说出来，因为这是一片热情跑在复现前面的文献。已验证并在若干规模上被独立复现的：在由程序提供真值的领域里，出题者—解题者回路确实能改进解题者。截至 2026 年中仅有单篇证据的：那些用内在信号或一个写评分细则的冻结评判模型来取代执行器的免验证器变体 (Huang et al., 2026a,b; Kwan et al., 2026)，它们全都以某种形式继承了基座模型的判断。完全没有被展示过的：Hughes et al. (2024) 意义上的开放性，即产物序列必须对一个观察者永远保持既新颖又可学。已报告的运行只有寥寥数轮，并且会饱和；一个三轮之后就饱和的课程是一份课程，而不是一个开放式过程。

对抗式的任务生成器尴尬地夹在这两种读法之间。把出题者的奖励设为解题者奖励的相反数，看起来像竞争式自博弈，表现却像一台不可能性生成器，因为这个场景

不是常和的：解题者在别处成功时出题者并不损失什么。没有共享资源约束的竞争结构，并不会产生那种让博弈式自博弈奏效的自我纠正动力学。

20.7 辩论、批评与验证博弈

辩论被用于两个不同的目的，而在这两者之间证据的分量差异很大。

用于监督的辩论。Irving et al. (2018) 把辩论提出为一种可扩展监督协议：在两位对抗的专家之间做裁断，可能比在无人协助的情况下核查一项断言更容易，所以一个比辩手更弱的评判者仍然可能作出正确裁定。这是一个关于信息不对称的断言，而经验记录恰恰在这个设定下支持它。Khan et al. (2024) 报告说，在评判者看不到原文的阅读理解任务上，辩论有帮助，且更强的辩手会让评判者更准确。Kenton et al. (2024) 把这项度量扩展到九个任务，发现辩论一致地胜过咨询制，后者是让一个模型为指派给它的一方辩护——这是个弱基线，因为咨询者有一半时间在为错误的一方辩护。若改为对照评判者干脆直接作答来度量，优势出现在带信息不对称的抽取式问答上，在别处则很小或没有。

用于能力的辩论。让若干实例回答同一个问题，把彼此的答案给对方看，并迭代至共识 (Du et al., 2024)，是一套推理时的流程，其成本是单个答案的 NR 倍。这里的图景比毁誉参半还要差。Wang et al. (2024g) 报告说，在一系列推理任务与基座模型上，一个带足够强提示的单智能体就能匹敌最好的讨论框架，只有当提示中不含示范时多智能体讨论才领先。Smit et al. (2024) 在显式的成本核算之下，把辩论协议与自洽性和集成作对比，发现常见形态的辩论并不能可靠取胜，同时对超参数明显更敏感。而Li et al. (2024a) 表明，朴素的采样加投票随样本数单调改善，这正是任何 NR 次调用的协议必须击败的基线。

工程实践：等算力对比不是可选项

一个有 N 个角色、R 轮的多智能体系统，大致花掉单个答案 NR 倍的生成量。因此报告说它胜过单次调用是没有信息量的。带信息量的那个对比固定的是推理成本：同一模型采样 NR 次并做多数投票或 best-of-n 选择 (Li et al., 2024a)；而对于一项训练上的断言，则是同一策略在同样多的词元上训练。Kapoor et al. (2025) 为智能体评测给出了这一论证的一般版本——准确率必须与成本联合报告，否则复杂脚手架就靠多花钱取胜——而它在这里同样完全适用。如果你跑不了那条等算力对照臂，就说出来，并把结果标为未度量，而不是标为收益。Cemri et al. (2025)

> 标注了来自七个多智能体框架的一千多条轨迹，归纳出十四种失败模式，其中大多数是组织性的，而不是底层模型的缺陷；他们的出发观察是，来自多智能体结构的基准收益经常是微乎其微的。

训练一场辩论与运行一场辩论是不同的活动，其中有三件事发生了变化。评判者变成了优化目标：用策略梯度针对一个固定评判者训练出来的辩手，被训练的是去说服它，而 LLM 评判模型有据可查的自我偏好与长度偏差 (Zheng et al., 2023a; Panickssery et al., 2024) 会变成训练信号。如果辩手共享参数，对抗结构就变成虚构的，因为一个策略被要求为双方辩护并输掉其中一方，还要接收两边梯度之和。再者，可以奖励交互本身而不只是奖励结果，这是 MAPoRL 的贡献 (Park et al., 2025a)，而且是必要的，因为协作式的终局奖励是为赞同买单的：一个提出反对的批评者，冒着让一次修订把正确答案改成错误答案的风险，于是反对率朝零下滑，而准确率纹丝不动。

把验证器作为被训练策略的这一路，比辩论站得更稳，理由是结构性的而非经验性的。CURE (Wang et al., 2025d) 在没有真值测试的情况下共同演化一个代码生成者与一个单元测试编写者，奖励测试者写出能把正确实现与不正确实现区分开来的测试。它有而辩论没有的，是验证器的输出最终会被拿去与程序执行核对。那个外部锚，正是「一个会改进的验证器」与「两个越来越彼此赞同的模型」之间的区别。

20.8 一次多智能体采样的算术

多智能体训练的昂贵之处很容易被低估，因为成本不在参数上。

生成的词元与预填充。在 N 个角色、R 轮、平均区段长度 L 的情况下，一条片段大约生成 NRL 个词元，而同一任务的单策略解法是 L_{single}；由于每个角色都要重述上下文并交接，即便底下的推理完全相同，这个比值也大于一。每个区段都以至此为止的整份转录为条件，所以除非键值缓存沿着这条链共享，否则总预填充工作量按 $(NR)^2 L$ 增长。对一条线性流水线它可以共享：前缀就是前缀，缓存可以往下带。分支拓扑会让缓存分叉，而算法 20.1的消融采样会把生成量乘以 $1 + M\left|\mathcal{J}_{\text{audit}}\right| f$，其中 f 是下游占一条片段的平均比例。审计一个靠后的角色很便宜，因为 f 很小；审计第一个角色则每次都要花掉一整条片段，这正是节 20.3.3在生成侧的重述。

关键路径与掉队者。各区段是顺序依赖的，所以一条片段内部无法并行，其时长是 NR 次轮次时长之和，每一次都包含工具延迟。若轮次时长的标准差为 σ_{turn}，则在独立性假设下片段时长的标准差为 $\sigma_{\text{turn}}\sqrt{NR}$。一个由 G 条片段组成的同步组要等最

慢的那条，而对于近似高斯的时长

$$\mathbb{E}[\max_{i\le G} T_i] \;\approx\; \mu_T + \sigma_{\text{turn}}\sqrt{NR}\,\sqrt{2\ln G}, \tag{20.7}$$

因此掉队者开销按 $\sqrt{NR\ln G}$ 增长：多智能体采样不只是花得更多，它还花得更不稳，而让训练集群空转的正是这份方差。式 (20.7)是一个数量级上的指引，不是一次测量；真实的轮次时长因为工具超时而是重尾的，这使得真实开销比高斯估计更糟。

缓解办法就是第21章所发展的那一套：带有界陈旧度的异步、分离式采样，使得一条慢片段永远不阻塞一条快片段（节 21.6 and 21.7）。有两处多智能体特有的相互作用要紧。一条横跨多个策略版本的片段在这里比在单智能体情形下更糟，因为同一份转录的不同区段可能是在不同权重下生成的，从而是逐区段地、而不是均匀地污染重要性比值；第22章给出了修正，而实践规则是在片段层面约束陈旧度，并逐区段记录采样版本。再者，消融采样是在一条本已受限于推理的流水线上纯粹追加的推理，所以它们应当以低优先级发给同一个采样池：缺一次消融只意味着某个角色退回到广播，而不是一步失败。

20.9 究竟何时才该按多智能体来训练

工程实践: 一个默认答案是「把它蒸馏掉」的决策流程。

第 0 步。建立等算力的单策略基线。取多智能体脚手架每个任务的总推理成本，把它花在一个策略上：带选择的采样、在同一上下文里做更多轮自我批评，或者一条更长的思维链。如果脚手架赢不了这个，那就没有什么可训练的，脚手架应当被删掉而不是被优化。

第 1 步。如果脚手架赢了，把它蒸馏掉。用脚手架生成轨迹，训练一个策略去复现包含角色转换在内的整份转录，然后在该策略上跑普通的智能体式强化学习 (Li et al., 2025e)。这消除了智能体间的非平稳性，消除了归因问题，消除了额外的预填充，还保住了行为。这是默认做法，而且它正确的次数比文献注意力的分布所暗示的要多。

第 2 步。只有在某个具体条件成立时才做多智能体训练。这样的条件有四个，而且它们是关于问题的条件，不是偏好：(i) 各角色确实需要不同的参数——不同的模态、规模差异悬殊，或者某个角色为了可审计性必须保持冻结；(ii) 存在一个不是系统奖励之函数的逐角色奖励，比如一个测试编写者可以按其对已知坏程序的判别力来打分 (Wang et al., 2025d)；(iii) 某个角色必须被剥夺其他角色拥有的信

息，使得这道屏障本身就是目的，而不是上下文预算的副产物；(iv) 各角色必须可独立部署或可独立升级，这是一项工程要求，而且是一项正当的要求。若一条都不成立，回到第 1 步。

第 3 步。如果你确实要训练它，那就共享参数并按角色掩码。使用算法 20.1。从对每个角色都用均匀广播开始，因为它免费且无偏。用几十次消融采样为每个角色估计 ϱ_j，并只把满足(20.6)的那些角色升级到反事实信用，实践中这意味着靠后的那些。

第 4 步。度量逐智能体的边际价值，而不是系统奖励。按角色、每一百次更新记录：从消融测得的边际价值、该角色的组发生退化的任务占比、匹配上下文上角色条件输出之间的散度，以及——对任何批评者——反对率连同给定反对时的条件准确率。一个边际价值趋于零的角色已经被训练到不复存在了，而你还在为它的词元付钱。

第 5 步。在最后重跑第 0 步。等算力基线会随着策略改进而移动。一项在初始化时存在的多智能体优势，往往熬不过训练，因为脚手架所提供的东西大多是探索多样性，而强化学习此时已经把它内化了。

保留多个智能体的战略性论据——即便有了上面这一切也仍值得认真对待——不是任务分解，因为一个带长上下文的策略同样会分解任务。它是探索的多样性：不同的智能体携带不同的上下文与历史，所以它们的能力边界不同，一个探索不到的方向可能被另一个覆盖到 (Zhang et al., 2026b)。这样读来，多智能体系统是一种基于种群的探索方法，而相关的对比对象是把同样算力花在多样性上的其他方式——更高的温度、更大的组、带选择的并行采样（节 9.4）。这个对比几乎从来没人跑，而跑它会比再造一个脚手架更有信息量。

本章小结

- 决定优化问题的那个区分不是架构，而是参数是否共享、奖励是否共享：表 20.1四格中的三格都可归约为前几章的机制，第四格是一个算法并不挑选解概念的一般和博弈。经典形式化框架只能部分迁移，因为语言模型「智能体」通常共享权重与一个上下文窗口，并且顺序行动，这既消除了非平稳性与局部可观测性，又添加了一条无界的、不忠实的通信通道。

- 逐智能体优势望远镜式相消为轨迹优势（定理 20.2），因此信用只会被再分配，从不会被创造，而智能体索引只不过是一种粗粒度的、语义上有意义的划分。于是一个共享的轨迹级奖励会给关键智能体和无关智能体相同的优势：这无偏，却仍然有害，因为无关智能体收到的是全量级的均值为零的梯度噪声，并会退化成一个模板。批评者就是这样在系统奖励纹丝不动的情况下被训练到不复存在的。
- 任何仅为某个智能体上下文之函数的基线都保持无偏性（定理 20.3）；重新采样该区段并把下游重采一遍，就给出Wolpert and Tumer (2001) 的差分奖励，而当重采样分布就是策略本身时，则在不需要集中式评论家的情况下给出 COMA 的反事实优势 (Foerster et al., 2018)。这个基线绝不能通过把一个默认区段拼接进一份下游是在真实区段下生成的转录来计算。
- 带 M 次消融采样的反事实信用胜过均匀广播，当且仅当在该智能体发言之前就已被决定的那部分结果方差超过 $1/(M+1)$ (20.6)。这个占比对第一个角色为零，并沿流水线递增，所以逐智能体信用在链条前端一文不值，在末端最有价值——而那里也恰好最便宜，因为只需重采一小段下游。
- 每一种训练模式都有一个特征性失败与一个便宜的诊断，而这些诊断没有一个是奖励曲线：冻结一个伙伴会产生对该伙伴的过拟合，交替优化会振荡，共享参数会产生角色坍缩，而一个由子智能体结果给奖励的管理者会学会路由给最容易拿到奖励的子智能体、而不是最好的那个。
- 消息是动作，所以强化学习会为下游作用而优化它们，而不是为可读性或忠实性；保住可读性的唯一力量是参考策略锚定。针对监控器训练教出来的是混淆而非遵从 (Baker et al., 2025)，而两个被联合优化的智能体拥有一条支持隐写合谋的通道 (Motwani et al., 2024a)。
- 自博弈在存在一种由系统之外来检查的可验证不对称性时奏效，或者在存在一个有明确胜负条件的对称常和博弈时奏效。对手生成两者兼具，并继承节 9.5的那些保证；任务生成两者皆无，收敛到出题者奖励所使之平稳的任何东西，并容许双方都拿到报酬而谁也没有进步的合谋不动点。
- 辩论已被测得的收益，在评判者缺少辩手所拥有的信息时是真实的 (Khan et al., 2024; Kenton et al., 2024)，而在能力方面并未确立：一个足够强的单一提示就能匹敌讨论框架 (Wang et al., 2024g)，而朴素的采样加投票随调用次数扩展 (Li et al., 2024a)。任何 NR 次调用的协议都必须对照一个 NR 次采样的单策略基线来报告。

- 多智能体采样花费更多，更糟的是花得更不稳：掉队者开销按 $\sqrt{NR\ln G}$ 增长(20.7)，而同一份转录的各区段可能是在不同策略版本下生成的，这会非均匀地污染重要性比值（第22章）。
- 对「我该不该把这个按多智能体来训练」的默认答案是否定的：先击败等算力的单策略基线，然后把脚手架蒸馏成一个策略 (Li et al., 2025e)，只有在节 20.9那四个具体条件之下才训练多个智能体。

延伸阅读

先从经典的信用分配论文读起,因为语言模型文献在重新发明它们。Wolpert and Tumer (2001) 是差分奖励的源头，值得读的是它的立意——把每个智能体的收益设计成「最大化它不会损害集体」——而不是其数学，后者定理 20.3用一页就重述完了。Foerster et al. (2018) 是应当内化的那个版本：反事实优势，用一个集中式评论家来计算，因为离散动作空间可以被精确边缘化。对照它来读节 20.3.3，并注意到语言模型场景用采样取代了边缘化，其理由与组相对方法之所以能奏效是同一个。Yu et al. (2022a) 在「协作式多智能体的性能有多少来自实现而不是算法」这一点上是有用的纠偏，Bernstein et al. (2002) 是「为什么没人精确求解这些问题」的论证，而Oliehoek and Amato (2016) 是那本紧凑的专著，如果你需要完整的去中心化 POMDP 机制，就把它放在手边。

至于语言模型文献，请仔细读Park et al. (2025a)，并且是为它的负面结果而读：独立优化出来的模型并不会组合成一个协作的系统。Motwani et al. (2024b) 是对角色条件信用传播最干净的处理，而Li et al. (2025e) 是那篇应当改变你做法的论文，因为它把多智能体系统当作一套数据生成脚手架并将其蒸馏掉。然后把Cemri et al. (2025) 当作经验层面的纠偏来读——从真实轨迹中提炼出的十四种失败模式，其中大多数是组织性的——再把Kapoor et al. (2025) 当作方法层面的纠偏。关于辩论，先读Irving et al. (2018) 看论证，再读Kenton et al. (2024) 看度量，顺序不要颠倒；然后是Wang et al. (2024g) 与Smit et al. (2024)，两篇都很短，而且会让你对大多数多智能体能力宣称产生免疫。

关于通信，语言模型之前的涌现通信文献比此后所写的任何东西都更有信息量。Kottur et al. (2017) 只有四页，说出了统摄本章第五节的那个要点：自然语言不会自然涌现，所以如果你想要可审计的消息，就必须为它付钱。Lazaridou and Baroni (2020) 是地图。接着读 Motwani et al. (2024a) 看对抗版本，以及Baker et al. (2025) 看那个最可操作的单一发现——不要针对你的监控器做训练。

刻意留给其他章节的内容：用于对齐的自博弈的博弈论表述，包括 Nash-MD 与 SPPO，在节 9.5；本章加以特化的那套信用分配机制在第13章；单个智能体内部的轮次级信用与观测掩码在 Chapters 16 and 18；任务与环境供给，包括对生成任务的难度校准，在第19章；让多智能体采样负担得起的系统架构在 Chapters 21 and 22；而训练彼此通信的智能体所带来的监督后果在第25章。

习题

20.1（**望远镜式相消与方差账本。**）从(20.2)的定义出发证明 定理 20.2，明确说出你需要关于奖励与转移的哪些假设，并证明 $\mathbb{V}\mathrm{ar}[V_j]$ 关于 j 非减。然后通过两次应用全方差定律推导 (20.3)，并指出下列每一项干预分别缩减哪一项：加大组 G；逐智能体反事实基线；把环境变成确定性的；以及把被审计的角色往流水线后面挪。用一句话解释为什么均匀广播对最后一个角色而不是第一个角色最不利。

20.2（**无偏性及其限度。**）定理 20.3允许任何作为 h_j 之函数的 b_j。给出三个具体选择，其中一个无需任何额外生成即可计算，并对每一个用 U_j, S_j, Z_j 写出由此得到的方差。然后构造一个显式的两角色例子，其中把默认区段拼接进已记录的转录、却不重采下游，会产生一个与 u_j 相关的基线，并计算由此产生的偏差的符号。

20.3（**对齐的是预算，不是步数。**）节 20.3.3 在梯度样本数固定的条件下比较各估计量。改在固定生成预算下重做这个比较：每个任务共有 B 条生成的片段，你或者取 $G = B$ 条均匀片段，或者取 $G' < B$ 条片段外加每个被审计角色 M 次消融采样。把两种情形下平均梯度的方差写成 ϱ_j、M 与平均下游比例 f 的函数，并找出在 (ϱ_j, f) 中逐智能体信用取胜的区域。

20.4（**一个惰性智能体。**）考虑一条两角色流水线，其中角色 1 的输出可证不影响奖励。证明角色 1 的均匀广播梯度期望为零，但方差为 $\mathbb{V}\mathrm{ar}[r]$ 乘以得分范数平方的期望。假定有一个系数为 β 的 KL 锚以及一个大小为 n 的有限批量，非形式地论证角色 1 的策略最终会落到哪里，并设计一个诊断——可从训练日志计算，无需额外采样——使其能在几百次更新之内检测到它。

20.5（**消息漂移。**）搭建一个两角色系统，其中角色 1 发送一条消息、角色 2 消费它，并有一个终局验证器奖励。显式写出消息词元上的梯度，并指出目标函数中提及人类可读性的每一项。然后提出两种机制，使可读性成为目标函数的一部分，而不引入

一个可被作弊的奖励项，并对每一种解释它为什么不只是把压缩挪了个地方，论证时参照Baker et al. (2025)。

20.6 ⋆ **（一个抗合谋的出题者奖励。）**在节 20.6 的任务生成场景中，合谋不动点是一个发出解题者已经能按目标通过率处理的任务的出题者。设计一个使这一点不再平稳的出题者奖励——在该奖励下，这样的出题者可以通过发出绝对意义上更难的任务而被严格改进。你可以使用一个冻结的参考解题者、一个留出集，或者两者都用。然后论证、或给出反例来反驳如下断言：任何这样的奖励都会重新引入对某个外部难度来源的依赖，因而「无外部数据」与「抗合谋」不可能同时成立。

20.7 ⋆ **（逐智能体信用会改变不动点吗?）** 定理 20.3确立了梯度估计量在每一个 θ 处的无偏性，但这本身并不意味着两个无偏估计量在有限学习率与共享参数集之下会到达同一个解。构造一个共享参数的两角色例子，其中均匀广播与反事实信用收敛到不同的行为，并指出共享参数化的哪条性质对此负责。把你的答案与表 20.2中的角色坍缩失败联系起来。

20.8 † **（这个脚手架值几个钱?）**在一个带程序化验证器的任务上取一个公开的多智能体脚手架。测量它的准确率以及每个任务总共生成的词元数。然后用同一个基座模型搭三条等推理成本的单策略对照臂：带多数投票的独立采样、在单一上下文中带自我批评的一条长思维链，以及在推理时可以用到验证器的情况下的 best-of-n。用至少两个随机种子，报告这四者的准确率对词元数，并说明哪些差异超过了种子噪声。然后把该脚手架蒸馏成一个策略并重复这个比较。

20.9 † **（逐智能体信用，端到端。）**为一条两角色流水线实现 算法 20.1——一个规划者与一个执行者共享主干、带不同的角色词元——任务是一个带程序化奖励的工具使用任务。训练之前先用代码 20.2 为两个角色估计 ϱ_j，并根据(20.6) 预测在 $M \in \{1, 2, 4\}$ 时应当审计哪些角色。然后在等生成预算下训练三条臂：带更大组的均匀广播、只对执行者施加反事实信用，以及对两个角色都施加反事实信用。报告最终任务成功率、规划者与执行者在训练过程中被测得的边际价值、退化组的占比，以及(20.6)的预测是否成立。

算法 20.1 带反事实逐智能体基线的共享策略多角色组相对优化。第 5 行把这套昂贵的机制限制在 (20.6)预测会有收益的那些角色上；第 9 行是定理 20.3的正确性要求，而第 14 行是防止更新学会幻觉工具输出的那一行。

输入： 共享策略 π_θ、角色集 $\mathcal{J}$、组大小 G、消融次数 M、验证器 r_{ver}、审计阈值 $1/(M{+}1)$

1: **for** 批次中的每个任务 x **do**
2: 在 π_θ 下带角色词元 $\rho_1, \dots, \rho_N$ 采样 G 条完整片段 τ_i
3: $r_i \leftarrow r_{\text{ver}}(\tau_i)$； $\mu_r, \sigma_r \leftarrow$ 该组上的均值与标准差
4: **if** $\sigma_r = 0$ **then**
5: 跳过此任务 ▷ 退化组；节 12.4
6: **end if**
7: $\mathcal{J}_{\text{audit}} \leftarrow \{ j \in \mathcal{J} : \hat{b}_j > 1/(M+1) \}$ ▷ 滚动估计
8: **逐智能体基线**
9: **for** $j \in \mathcal{J}_{\text{audit}}$，每条片段 i **do**
10: $h \leftarrow \tau_i$ 中角色 j 的上下文 ▷ 前缀 KV 缓存已经驻留
11: 抽取 $u' \sim \pi_\theta(\cdot \mid h, \rho_j)$ 并**重新采样所有下游区段**，共 M 次
12: $\hat{b}_{j,i} \leftarrow M^{-1} \sum_m r_{\text{ver}}(\tau'_m)$
13: $\hat{A}_{j,i} \leftarrow (r_i - \hat{b}_{j,i})/\sigma_r$
14: **end for**
15: **组装词元级优势向量**
16: 对角色 $j \notin \mathcal{J}_{\text{audit}}$：$\hat{A}_{j,i} \leftarrow (r_i - \mu_r)/\sigma_r$
17: 把 $\hat{A}_{j,i}$ 赋给 τ_i 中区段 u_j 的每一个词元，观测词元赋 0
18: **end for**
19: 在汇集后的掩码优势上做一次截断的词元级更新
20: **return** π_θ

第 V 部分

系统

第21章

大规模后训练的系统

核心思想：一套 RL 系统就是共用一份权重的两种互不相容的负载

生成受内存带宽约束、由延迟主导，且每个样本的成本变化极大。梯度计算受算力约束、由吞吐主导，且每个词元的成本固定。一次强化学习步必须在同一份参数上交替完成这两件事——而且每当参数发生变化，都必须把它们从训练器的布局拷贝成引擎的布局。本章中每一个架构决策都出自这一张力。在实践中，系统层面的选择压过了算法层面的选择：同一个 GRPO 目标函数的两种实现，在每 GPU 小时的词元数上经常相差 3–5×，这比任何已发表的损失函数改动的效应都大。

第 IV 部分问的是优化什么。本部分问的是你实际能跑什么。这个区分在这里比在机器学习中通常的情形更重要，因为在从生成数据出发的强化学习里，目标与基础设施是不可分离的：行为策略就是推理引擎恰好采出来的那个，数据的陈旧度就是调度器恰好产生的那个，而两者都出现在估计量里。因此，本书中的系统一章不是附录，它是用另一种手段对算法论证的延续。

我们在这里处理调度与放置。与之密切相关的另一个问题——为什么推理引擎和训练器会给同一个词元赋予不同的概率，以及对此该怎么办——是第22章的主题；在预算内跑完整套东西的经济学——用于 RL 的 LoRA、量化、缓存、每一分基准得分的成本——是第23章。这三章是从三个角度描述同一台机器。

21.1 一个训练步的解剖

把一个在线策略 RL 步分解成若干阶段。设有 B 个提示、组大小 G、平均回复长度 L：

1. **生成**。用当前权重为每个提示采样 G 条回复。先预填充提示，然后自回归地解码 BGL 个词元。

表 21.1: 一个同步步的阶段分解，由节 21.1的屋脊线模型算出而非实测，取 B=64、G=8，硬长度上限为均值的 4×。这些是数量级上的估计，其用意在于那些比值；其中的生成数字与Async RL Survey (2026) 的框架综述在可比设置下所报告的范围一致。百分比是相对于步的挂钟时间。最后一列是该阶段占算术运算的份额，其排序几乎正好相反。

阶段	受限于	7B, 2k	7B, 8k	70B, 8k	FLOPs
生成	HBM 带宽、长尾	90 s (59%)	600 s (81%)	1800 s (84%)	17%
奖励／验证	外部、I/O	8 s (5%)	25 s (3%)	25 s (1%)	—
对数概率重算（×2）	算力	18 s (12%)	36 s (5%)	103 s (5%)	33%
前向 + 反向	算力	14 s (9%)	54 s (7%)	155 s (7%)	50%
优化器步	带宽、集合通信	3 s (2%)	3 s (< 1%)	12 s (< 1%)	< 1%
权重同步 + 休眠／唤醒	互联	20 s (13%)	20 s (3%)	60 s (3%)	—
合计		**153** s	**738** s	**2155** s	

2. **奖励与验证**。对每条完成的轨迹运行一个程序、一个沙箱、一套测试套件，或者一个奖励模型。成本从微秒（字符串匹配）到分钟（容器启动加测试执行）不等。
3. **对数概率重算**。在 π_{ref} 下做一次前向以得到 KL 项，并且——在大多数实现中——在 $\pi_{\theta_{\text{old}}}$ 下再做一次前向，以便在训练器的数值下、而不是引擎的数值下取得行为对数概率。
4. **前向与反向**。代理目标的梯度，在若干微批上累积。
5. **优化器**。AdamW 更新、梯度截断，以及分片方案所需的任何参数 all-gather。
6. **权重同步**。把新参数按引擎的布局搬进推理引擎。

FLOP 的核算在节 1.6给出过：生成花费 $2NBGL$，反向传播花费 $6NBGL$，而每一次对数概率前向再花 $2NBGL$。在有两次重算前向的情况下，生成占算术运算的 $2/12 \approx 17\%$。表 21.1给出时间都花到哪里去了。

对表 21.1有两种读法值得明确说出来。

第一，生成占算术运算的 17%，却占挂钟时间的 60–85%。这两个数字之比，就是两个阶段之间实际达成的模型 FLOP 利用率之比：生成跑出来的效率大约是反向传播的十分之一。因此你算法的 *FLOP* 数几乎无关紧要。在 8k 回复长度下，把反向传播的成本减半只为你买到该步的 3–4%。把生成吞吐翻倍则买到 40%。任何不以生成为目标的工程投入，最好是用正确性而不是速度来辩护。

第二，这份剖面并非尺度不变的，它依赖于长度。在 2k 回复下——也就是经典的基于偏好的 RLHF——生成只勉强占该步的一半多，而权重同步占到了显眼的 13%，因为这一步短到足以让每步固定开销变得要紧。在 8k 下它是五分之四。从 RLHF 到长思维链 RLVR 的转变不只是让运行变长了；它挪动了瓶颈。

推导: 为什么解码无法被做成算力受限

考虑一个解码步，其中驻留着 b 条序列。前向传播对每条序列执行约 $2N$ 次 FLOP，合计 $2Nb$。它必须读取的字节是权重（整个批次读一次，BF16 下为 $2N$ 字节），外加每一个驻留词元的键／值缓存。暂时忽略缓存，算术强度为

$$I = \frac{2Nb}{2N} = b \quad \text{FLOPs per byte.}$$

一块 H100 有约 989 TFLOP/s 的稠密 BF16 吞吐，对应 3.35 TB/s 的 HBM 带宽，机器平衡点约为 295 FLOP/字节。所以你需要 $b \gtrsim 300$ 条并发序列，才能让单是权重读取这一项不再成为瓶颈；而 KV 这一项让情况更糟：在 32 层、8 个维度为 128 的键／值头、BF16 的配置下，每个驻留词元在每个解码步上要花掉 128 KiB 的缓存流量，于是在 $b = 390$ 条平均上下文为 5000 的序列下，缓存读取是 256 GB，而权重只有 14 GB。解码受带宽约束达一个数量级，而且一直如此。

反向传播恰恰相反。它每读一次权重就处理数千个词元，所以 I 是数千量级，核函数贴近算力天花板。这两个阶段位于屋脊线拐点的两侧。这就是「两种负载」这一说法的物理内容，也是为什么再多的核函数工作也无法让生成看起来像训练。

常见陷阱: 优化你能测到的东西

性能分析器报告 FLOP 和核函数时间，于是团队就去优化核函数。在一次 RL 运行里，最大的单项开销通常根本不是核函数：它是那段有 80% 的解码槽位空着、只因在等批次里最长那条序列的时间（节 21.7），或者那段训练 GPU 因生成尚未结束而闲置的时间。两者都不会表现为一个慢核函数。要给阶段边界与占用率装上仪表，而不只是给核函数装；代码 21.3是一个能做到这一点的最小框架。

21.2 同置与分离式

第一个架构决策是推理引擎住在哪里。图 21.1给出两种答案。

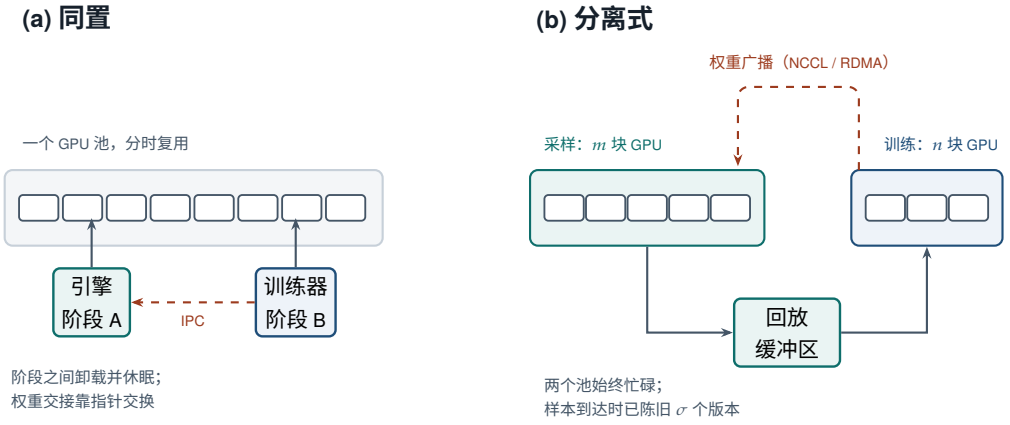


图 21.1: 两种放置架构。在 (a) 中，一个加速器池在推理引擎与训练器之间交替，所以峰值内存是两者工作集的并集，除非在阶段之间把其中一个卸载掉；而当前没在运行的那种负载所对应的能力就闲置着。在 (b) 中两个池是分开的、始终忙碌，代价是要买 $m+n$ 块 GPU 而不是 $\max(m,n)$ 块，每次更新都要把权重发过网络，并且要接受样本是由落后 σ 个优化器步的权重生成的（节 21.6）。

21.2.1 同置（混合引擎）

一个 GPU 池交替地运行训练器与引擎。在阶段之间，闲下来的那一方释放它的设备内存：vLLM 的 sleep 模式与 SGLang 的 sleep/wake 做的正是这件事，分两个激进程度——第一级把权重卸载到主机内存并丢弃 KV 缓存，第二级把两者都丢弃，只保留诸如旋转位置表之类的小缓冲区 (vLLM Project, 2025; SGLang Project, 2026)。这样一来权重交接几乎是免费的，因为双方处在同一设备上的同一进程空间里：你发布 CUDA IPC 句柄，或者干脆交换一个指针，完全不涉及网络。

它的优点是实打实的。你只买一个池，不是两个；你从不为权重付网络带宽；而且数据严格是在线策略的，因为生成严格发生在消费它的那次更新之前。

代价在表 21.1里看得见：生成运行时，该池的训练能力闲置；而反向传播运行时，引擎在睡觉。利用率的上界是该步中活跃负载能把硬件吃满的那部分比例，而既然生成吃不满算力，这个天花板就很低。同置设计还要为休眠／唤醒这套舞蹈付出每步的固定开销——释放并重新分配数百 GB、重新预热分配器、有时还要重新捕获 CUDA 图——在一段 600 秒的生成阶段上摊得很好，在一段 90 秒的生成阶段上摊得很糟。

同置是 verl 最初混合引擎的设计、OpenRLHF 推荐的混合模式 (OpenRLHF Project, 2025)，以及为 Kimi K2 所报告的 RL 系统的设计 (Kimi Team, 2025b)。

21.2.2 分离式（分开部署）

专用的采样 GPU 与专用的训练 GPU，由一条承载轨迹的数据通道与一条承载权重的控制通道相连。两个池都持续运行。由于它们的规模是独立确定的，你可以让容量去匹配实际的 $T_g : T_t$ 比值，而不是接受单个池碰巧给你的那个比值——而这个比值在一次 2k 词元的偏好训练与一次 32k 词元的智能体训练之间会相差一个数量级。你还可以为两者使用不同的硬件，这很有吸引力，因为生成想要的是带宽与容量，而训练想要的是 FLOP。

verl 的文档不加修饰地陈述了动机：分离式架构「可以更灵活地分配资源、设计更灵活的训练逻辑，从而解决长尾问题带来的 GPU 利用率低与训练效率低等问题」(verl Project, 2026a)。这是 AReaL (Fu et al., 2025a)、slime (THUDM / Z.ai, 2026)、StreamRL (Zhong et al., 2025)、Magistral 的训练系统 (Mistral-AI et al., 2025)，以及 verl 的 `fully_async_policy` 路径所采用的设计。

三项代价。为了达到任一种负载单独的同等峰值吞吐，你要买更多 GPU。每次同步你必须把 $2N$ 字节的权重发过网络，而不是交换一个指针。而且，不可避免地，一旦两个池并发运行，训练器所消费的数据就是由更旧的权重生成的。最后这项代价不是一项系统代价而是一项估计量代价，这也正是节 21.6存在的理由。

工程实践: 在两者之间做选择

一条还算好用的决策规则，按优先级排列。

- 如果你只有一个节点，就同置。在大致少于 16 块 GPU 的规模上做分离，多半只是给你买来一堆闲置硬件。
- 如果平均回复长度在几千词元以下，且长度分布很紧，就同置：尾巴很小，所以气泡很小，而且你避开了一整类陈旧度相关的 bug。
- 如果采样涉及真实的环境延迟——工具调用、容器、浏览器——就分离。生成器会把大部分时间花在阻塞于某个不是 GPU 的东西上，而你承受不起让训练器堵在它后面。
- 如果长度分布是重尾的、上限远高于均值，就分离，因为治尾巴的办法（节 21.7）要求未来若干步的工作现在就可用，这需要一个缓冲区，而缓冲区需要分开的池。
- 否则，先同置，测量气泡，等到实测气泡超过该步的大约 25% 时再分离。同置系统明显更容易调试，因为在任一瞬间世上恰好只存在一个版本的权重。

21.2.3 单控制器、多控制器，以及你为什么该在意

与放置正交的是控制。在多控制器或 SPMD 模型中，每个 rank 运行同一个程序，各 rank 通过集合通信来协同。Megatron-LM 与经典的分布式训练就是这么工作的 (Shoeybi et al., 2019; Narayanan et al., 2021)，而且它是现有最快的东西，因为关键路径上没有中央协调者。它同时也很僵硬：在一个每个 rank 都完全一样地执行的程序里，表达「样本 i 还需要三次工具调用，而样本 j 已经完成」是很痛苦的。

在单控制器模型中，一个驱动进程持有算法，并向工作进程池发出远程调用。RL 算法变成了普通的 Python，代价是关键路径上多了一个协调者，而它本身也可能变成瓶颈。

verl 背后的设计 HybridFlow 是两者的混合 (Sheng et al., 2025)：一个单控制器把算法表达为一张多阶段数据流图，而在该图的每个节点内部，工作进程在 FSDP、Megatron 或某个推理引擎之下以 SPMD 方式运行。verl v0.7 的说明把由此得到的系统描述为与异步分片数据流系统共享设计原则，把 PPO、GRPO 与 DAPO 建模为「一张多阶段、多模型且可并行的数据流图」(verl Project, 2026b)。Ray 是通常的底座 (Moritz et al., 2018)，因为它能处理异构放置、重启与共享内存对象传输，而算法作者一行都不用写。

这个区分对身为算法作者的你有很具体的影响。在单控制器之下，一个新的估计量就是对一个接收整批对象的 Python 函数的一处改动。在 SPMD 之下，同样的改动可能需要动到每个 rank 的控制流，而且如果它改变了何时需要哪些数据，还要动到集合通信的调度。这就是生态迁移的原因：verl v0.7 彻底移除了 SPMD 采样模式，改用一个通过 HTTP 接受逐样本请求的采样服务器，其所陈述的理由是智能体式 RL 让 SPMD 采样变得难以为继，并且服务器模式「消除了对 LLM 推理引擎作侵入式修改的必要」(verl Project, 2026b)。这大概是过去一年里最有后果的架构转变：采样引擎如今是一个在线服务系统，训练器是它的客户端，而不再是一个训练器去调用的批式推理库。

历史注记：从一个脚本到一张数据流图

最早的 PPO-RLHF 代码库（2019–2022）是一个个单脚本，在一个进程里持有四个模型，用朴素的采样循环做生成，一切都跑在数据并行之下。它们正确而缓慢，而在 1B 参数、500 词元回复的规模上那是没问题的。三股压力压垮了这个设计：模型规模不再装得下一块设备，迫使张量并行与流水线并行进入训练器；带分页 KV 缓存与连续批处理的推理引擎 (Yu et al., 2022b; Kwon et al., 2023; Zheng et al.,

> 2024) 比朴素循环快一个数量级，却有自己的并行布局；以及回复长度增长了两个数量级，使得生成成了整次运行的全部。留下来的不是一个脚本，而是一个包含两个子系统的分布式系统，而这两个子系统对权重该如何布局意见不合。本章的大部分内容讲的就是这一分歧的后果。

算法 21.1把同置步具体化了，其中包括论文里通常隐而不宣、而显存溢出故障通常就藏在其中的那套内存编排。

代码 21.1: 以配置形式勾勒的资源放置。这两段描述的是同样十六块 GPU 上的同一个算法；差别完全在于放置与陈旧度。命名足够接近 verl 与 SkyRL 的惯例以便辨认，但并非逐字照抄任何一方。

```
# (a) 同置：一个池，分时复用，严格在线策略
placement:      colocated
n_gpus:         16
actor:   {strategy: fsdp2, tp: 1,  offload_optimizer: true,
          offload_grads: true, seq_parallel: true}
rollout: {engine: sglang, tp: 2, gpu_mem_fraction: 0.75,
          sleep_level: 2, enforce_eager: false}
staleness_threshold: 0        # 每步一道同步屏障
partial_rollout:     false

# (b) 分离式：拆分池，有界陈旧度，吸收长尾
placement:      disaggregated
rollout_gpus:   10            # 按 8k 下实测的 T_g / T_t 定规模
trainer_gpus:    6
actor:   {strategy: megatron, tp: 2, pp: 1, cp: 2}
rollout: {engine: sglang, tp: 2, dp: 5, continuous_batching: true}
weight_sync: {mode: nccl_broadcast, bucket_bytes: 1073741824}
staleness_threshold: 4        # 拒收旧于 4 个版本的样本
partial_rollout:     true     # 把被打断的前缀带到后面继续
off_policy_correction: tis    # 一旦 staleness > 0 就必须开启；见第 22 章
```

21.3 RL 回路中的并行

在大模型训练中有五条并行轴相互组合：数据并行或分片数据并行（ZeRO/FSDP）(Rajbhandari et al., 2020; Zhao et al., 2023a)、张量并行、流水线并行 (Huang et al., 2019; Qi et al., 2024a)、序列并行与上下文并行 (Korthikanti et al., 2023; Jacobs et al., 2023; Liu et al., 2024b)，以及专家并行 (Lepikhin et al., 2021; Fedus et al., 2022)。它们的标

表 21.2: 每条并行轴具体是如何与 RL 回路相互作用的：数据（DP）、张量（TP）、流水线（PP）、序列与上下文（SP、CP），以及专家（EP）并行。第三列反复出现的主题是，训练器与推理引擎想要不同的布局，所以它们不一致的每一条轴都会给权重同步步骤贡献一次置换。

轴	切分对象	RL 额外带来什么	RL 中的典型用法
DP / ZeRO / FSDP	优化器、梯度、参数	缩减的是模型状态，不是 KV 缓存，所以除非在生成期间把分片卸载掉，否则它并不提高采样并发。训练器的 DP 宽度与引擎的 DP 宽度不同，所以同步是先聚合再分发，而不是一次拷贝。	始终开启
TP	单个矩阵乘	引擎的 TP 是按解码延迟选的，训练器的 TP 是按内存选的；两者很少相等。融合的 QKV 与 gate/up 投影在两套栈里沿不同的轴切分，这是无声重分片错误最常见的单一来源。	$\leq$ NVLink 域
PP	把层划分为阶段	气泡会与可变序列长度叠加；引擎几乎从不使用 PP，所以同步必须先跨阶段聚合，然后才能分发。	仅在约 70B 以上
SP / CP	沿序列切分激活	在训练器上支撑 32k 以上轨迹的那条使能轴，因为那里激活内存才是制约性约束。引擎处理长上下文靠的是给 KV 分页，所以这条轴只存在于一侧。	当 $L \gtrsim 16$k 时开启
EP	MoE 专家	路由依赖于数据，所以生成会在各专家之间造成负载不均；而且被选中的专家在引擎与训练器之间必须一致，否则重要性比值就是在两个不同的子网络之间算出来的（节 21.8）。	仅 MoE

准性质并非 RL 专有，我们不再重推。表 21.2记录的是：当同一张网格还必须服务一个生成阶段时，什么发生了变化。

21.3.1 两套布局，一份权重

这一点值得单独拎出来。一个持有 70B 模型的训练器可能跑张量并行度 8、上下文并行度 2，并在其余 rank 上做 FSDP。一个为采样服务同一模型的推理引擎，可能跑张量并行度 2 加四个数据并行副本，因为小的张量并行度解码延迟更低、总吞吐更高。那是对同一批参数张量的不同划分。因此同步不可能是字节的广播；它是一次重分片：对每个参数，聚合训练器的各分片，施加把训练器的约定转换成引擎约定的置换与拼接，然后分发到引擎的各 rank。

三条后果。

它花时间。重分片至少要触及 $2N$ 字节，通常是这个数的若干倍，因为「先聚合再分发」不是原地操作，也因为融合权重必须被拆开再重新融合。节 21.5给出成本模型。

正确性 *bug* 住在这里。一个错得恰好保持形状的置换，会产生一个能以略微错误的概率生成流利文本的模型。什么都不崩溃。奖励缓慢劣化。其症状与一个糟糕的学习率无法区分。

它是数值失配的一个来源。即便是一次正确的重分片也会改变各数值相加的顺序，因为张量并行度决定了一次矩阵乘的规约如何切分，而浮点加法不满足结合律。因此即便权重逐位相同，引擎与训练器仍会给同一个词元赋予不同的概率。这是第22章那个训练—推理差距背后的机制之一，也是为什么批不变核函数必须固定规约顺序、而不能只是固定随机种子 (He and Thinking Machines Lab, 2025)。

21.3.2 序列并行与长轨迹

Transformer 的激活内存随序列长度乘批量增长，这就是为什么 32k 词元的轨迹改变的是训练器内存剖面的性质而不只是数量。应对办法有三种，且彼此不可互换。重算用大约多 30% 的算力换取存储激活的大幅缩减，而选择性重算 (Chen et al., 2016; Korthikanti et al., 2023) 通过只重算那些便宜但吃内存的操作，把这笔交易的大部分收益拿了回来。序列并行沿序列轴切分非张量并行操作的激活，在已经启用张量并行的地方基本上是免费的，所以那种情况下它就应该一直开着。上下文并行为注意力本身把序列切分到各 rank 上，或者通过 all-to-all (Jacobs et al., 2023)，或者通过一圈分块注意力计算 (Liu et al., 2024b)，而且是三者中唯一能让每个 rank 的激活占用与长度无关的那一个。

RL 添了一道褶子：序列长度不只是长，它们还是可变的，并且在训练过程中会向上漂移。一个按第 0 步的长度分布选定的静态上下文并行度，到第 2000 步就错了。序列打包——把变长序列拼接进定长缓冲区，并配以正确的注意力掩码 (Krell et al., 2022)——近乎是必须的，因为 RL 里的填充浪费不是 SFT 里那 10–20%，而是当长度分布呈现节 21.7所描述的形状时可以超过 60%。NVIDIA 的 Megatron-Core MoE 报告正是基于这些理由把 RL 后训练当作一种独立的负载，把可变序列长度、内存卸载与在线权重导出点名为它的决定性挑战 (NVIDIA, 2026b)。NeMo-RL 报告说打包在两种后端上都带来了很大收益——以及在 70B 上启用它起初导致了显存溢出故障，这很好地说明了这是工程活儿，而不是一个开关 (NVIDIA, 2026c)。

常见陷阱: 重分片的正确性没法靠「它能跑起来」来测试

每一条权重同步路径都需要一个不是冒烟测试的测试。既便宜又有效的那个是：同步之后，取一批固定的十个提示，分别在引擎下和训练器下计算逐词元对数概率，并断言最大绝对差低于一个你在已知良好构建上校准过的阈值。一个重分片

bug 会把这个数字挪动几个数量级；单纯的精度差异不会。认真对待这件事的框架会把它作为常设测试发布出来——slime 的测试套件正是出于这个理由包含了数值精度测试与「先采样再训练的回放」测试 (THUDM / Z.ai, 2026)。如果你自己写同步，就先写这个测试。

21.4 内存

节 7.1引入过四模型问题：策略、评论家、参考模型与奖励模型，其中两个被训练、两个被冻结，模型状态合计每参数 36 字节。组相对方法（第12章）删掉了评论家，基于验证器的奖励（第11章）删掉了奖励模型，这就是该领域如何走到每参数 18 字节的。那是从训练器视角出发的核算。系统视角还得再加两个消费者：推理引擎那份权重副本，以及 KV 缓存。

KV 缓存是有意思的那一个，因为它是唯一一项能直接换来吞吐的开销。生成期间的并发为

$$b_{\max} = \left\lfloor \frac{M_{\text{free}}}{c\,(|x| + L)} \right\rfloor, \qquad c = 2 \cdot n_{\text{layers}} \cdot n_{\text{kv}} \cdot d_{\text{head}} \cdot \text{bytes}, \tag{21.1}$$

其中 c 是每词元的缓存成本。对我们贯穿全章的配置（8 个键／值头、头维度 128、BF16）而言，32 层下是每词元 128 KiB，80 层下是每词元 320 KiB。注意 c 是随深度而不是随参数量增长的，所以一个又深又窄的模型采样起来格外昂贵——在你选定一个打算用 RL 做后训练的架构之前，这是值得知道的事实。

由于在批次大到受算力约束之前（而按节 21.1的推导，它其实从来都没真正大到那个地步），生成吞吐大致随 $b_{\max}$ 线性增长，(21.1)说的是：你在生成阶段每释放一个 *GB*，都会直接转化成采样吞吐。这是 RL 专有的那项内存优化，而表 21.3把它量化了。

把最后两行连同这两列一起读。在八块 GPU 上的 7B 规模，模型状态只占内存的 22%，所以在生成期间把它释放掉只买来多 24% 的并发——值得要，但谈不上变革。在三十二块 GPU 上的 70B 规模，模型状态占内存的 55%，在生成期间释放它会让采样并发翻倍。卸载这个手法的价值不是一个常数；它随着你内存有多紧张而缩放，这就是为什么在 7B 规模写就的教程里它被说成可选，而在前沿规模写就的报告里它被说成必需。

还有两个 RL 专有的动作值得点名。参考模型根本不必驻留：它的对数概率每批只被消费一次，所以可以在单独一趟里算出来，缓存成一个逐词元标量的张量，然后把

表 21.3: 一次无评论家、以验证器给奖励的运行（GRPO 风格，无奖励模型）的内存预算实算，同置部署，权重与梯度为 BF16，AdamW 保有 FP32 主权重与两个 FP32 矩。数字是集群总量；在完整 ZeRO-3 分片下除以 GPU 数即得每设备数字。「预留」涵盖选择性重算下的训练器激活、集合通信缓冲区，以及分配器碎片。最后两行是算法 21.1的回报：在生成期间卸载梯度、优化器矩与参考模型。

组件	字节/参数	7B，8×H100	70B，32×H100
策略权重（BF16）	2	14	140
梯度（BF16）	2	14	140
AdamW 主权重 + 两个矩（FP32）	12	84	840
参考策略 π_{ref}（BF16）	2	14	140
推理引擎的权重副本（BF16）	2	14	140
驻留模型状态	**20**	**140**	**1400**
HBM 总量		640	2560
预留（激活、缓冲区、碎片）		40	160
KV 预算，训练器状态驻留		460	1000
→ 缓存词元数		3.5M	3.1M
→ 并发 9k 词元序列数		390	340
KV 预算，训练器状态已卸载		572	2120
→ 并发 9k 词元序列数		485（1.2×）	720（2.1×）

权重丢掉——这把 $2N$ 字节的永久驻留换成了一项带宽成本。对称地，KV 缓存应当在反向传播之前就释放，而不只是在优化器步之前，因为激活内存的峰值出现在反向传播内部，而缓存在生成结束的那一刻就已经是死的了。两者都很显然，两者又都经常被漏掉，因为这两个工作集由不同的库、用不同的分配器管理，而彼此都不知道对方存在。

工程实践: 内存分诊，按顺序

当一次运行装不下时，顺着这份清单往下做，在第一个买到足够空间的条目处停下。排序依据是性价比，不是方便程度。

1. 如果还没开，就打开对参数、梯度与优化器状态的 ZeRO-3 / FSDP 分片。除了通信之外是免费的。

2. 打开序列打包。免费，而且往往还是吞吐上的收益。
3. 打开选择性激活重算。代价是反向传播的约 30%，也就是整步的约 2%。
4. 在生成期间卸载梯度、优化器矩与 π_{ref}（算法 21.1）。代价是每步两次主机—设备带宽。
5. 预先算好并缓存 $\log \pi_{\text{ref}}$，然后把参考权重丢掉。
6. 加上下文并行。代价是每个注意力层一次 all-to-all 或环形交换；当问题出在长度而不是参数量上时，这是对的一招。
7. 把优化器状态永久卸载到主机内存 (Ren et al., 2021)。这会花掉实打实的吞吐；对训练器来说是最后手段，尽管对超大模型来说是家常便饭。
8. 到这一步才考虑减小批量或组大小 G，两者都会改变估计量（节 12.2），不应当被当作内存旋钮。

21.5 权重同步

同步是那个隐藏的瓶颈：在论文的方法一节里看不见，实践中偶尔占掉整步的三分之一。下面这些机制大致按延迟递减排列：

- **朴素的 NCCL 广播，逐参数进行**。每个张量一次集合通信。一个 32B 模型有数千个张量，而成本由每次调用的延迟、而不是字节数主导。框架综述把这一项定在一个 32B 模型约 500 ms (Async RL Survey, 2026)。
- **分桶的 NCCL 广播**。把参数合并进约 1 GB 量级的桶，然后广播这些桶。字节数相同，调用次数少三个数量级：同一篇综述报告同一模型约 20 ms，提升 25×。这是权重同步路径上现成可得的最大常数因子胜利，而且是纯工程。
- **RDMA ／专用参数服务**。对于分离式部署，一个带流水线化异步传输的参数服务器，例如 Moonshot 那个建立在传输引擎而非集合通信之上的 checkpoint-engine (Moonshot AI, 2025)。
- **CUDA IPC 句柄或指针交换**。只有当引擎与训练器共享设备时才可用。传输退化为一次设备内拷贝，或者，如果布局恰好一致，退化为什么都不做。
- **只同步适配器**。如果策略是冻结基座权重之上的一个 LoRA 适配器，那你发的是适配器：几十兆字节而不是几十吉字节，减少 500–1000×，把问题彻底化解掉 (Schulman and Thinking Machines Lab, 2025)。第23章会讨论 LoRA 在统计上

对 RL 是否够用；这里的要点只是：它至少同样是一项分布式系统优化，而不只是一项内存优化。

21.5.1 一个成本模型与一条经验法则

设 $S = 2N$ 字节为载荷，B_{eff} 为有效广播带宽，$\kappa \geq 1$ 为重分片放大系数（每交付一字节所触及的字节数；对带张量重融合的「先聚合再分发」而言 $\kappa \approx 2$–4），τ 为休眠／唤醒或暂停／恢复协议的固定成本。于是

$$t_{\text{sync}} \approx \tau + \frac{\kappa S}{B_{\text{eff}}}. \tag{21.2}$$

对一个 70B 模型，$S = 140\,\text{GB}$。在节点内经 NVLink、有效带宽约 700 GB/s、$\kappa = 3$ 时，带宽项是 0.6 s。跨节点经 400 Gb/s 网络、有效带宽约 45 GB/s 时，朴素单播是 9 s，广播树下是几秒——这与 Magistral 报告的「即便在很大的模型与世界规模下权重广播也在五秒内完成」一致 (Mistral-AI et al., 2025)。在同置部署中 B_{eff} 就是 HBM 带宽，带宽项几乎消失，但 τ 不会：释放并重新分配一个 460 GB 的缓存、重新预热分配器，还可能重新捕获 CUDA 图，这是数秒的工作量，这也正是引擎为什么要暴露细粒度的唤醒与暂停模式 (vLLM Project, 2026; SGLang Project, 2026)。

经验法则由(21.2)与表 21.1推出。当 $t_{\text{sync}} \lesssim 0.05\, t_{\text{step}}$ 时同步可以忽略，而 t_{step} 由 L 个解码步主导。所以：

$$\text{sync matters when} \quad \frac{\kappa S / B_{\text{eff}} + \tau}{L \cdot t_{\text{decode}}} \gtrsim 0.05. \tag{21.3}$$

长轨迹能藏掉极大量的同步成本；短轨迹不能。而且这个比值随每步同步次数线性恶化，这就是为什么向异步设计的转变——在那里权重可能在单条序列生成期间被发布好几次——会把同步成本从一个舍入误差变成一个一阶问题。一个在前向传播之间发布权重的设计，必须把 t_{sync} 压到低毫秒量级，在实践中这意味着同置、IPC，或者 LoRA。

代码 21.2: 带显式重分片步骤的分桶权重同步，形态与大多数框架的实现相同。那个 `plan` 是一份在启动时构建一次的静态列表，元素为（引擎参数名、训练器分片列表、变换）；把它构建正确才是难的部分，而 `verify_logprobs` 就是前面那条陷阱里描述的测试。

```python
import torch, torch.distributed as dist

BUCKET = 1 << 30  # 1 GiB

def sync_weights(trainer, engine, plan, group, bucket_bytes=BUCKET):
```

```
    engine.wake(tags=["weights"])          # 只唤醒权重缓冲区，不含 KV 缓存
    bucket, nbytes = [], 0
    for name, shards, transform in plan:
        with torch.no_grad():
            full = trainer.gather(shards)  # all-gather ZeRO-3 / FSDP 分片
            tensor = transform(full)       # TP 重切分、QKV 重融合、dtype 转换
        bucket.append((name, tensor))
        nbytes += tensor.numel() * tensor.element_size()
        if nbytes >= bucket_bytes:
            _flush(bucket, engine, group); bucket, nbytes = [], 0
    if bucket:
        _flush(bucket, engine, group)
    engine.wake(tags=["kv_cache"])         # 到现在才分配缓存
    engine.bump_version()

def _flush(bucket, engine, group):
    flat = torch.cat([t.reshape(-1) for _, t in bucket])
    dist.broadcast(flat, src=0, group=group)     # 每个桶一次集合通信
    engine.load_flat([(n, t.shape) for n, t in bucket], flat)

def verify_logprobs(trainer, engine, probe_batch, tol=2e-2):
    lp_e = engine.logprobs(probe_batch)          # 引擎数值
    lp_t = trainer.logprobs(probe_batch)         # 训练器数值
    gap = (lp_e - lp_t).abs().max().item()
    assert gap < tol, f"reshard suspect: max |dlogp| = {gap:.4f}"
    return gap
```

21.6　异步

到目前为止的一切都是关于把一个同步步做得更便宜。异步转而攻击步本身的结构，而它是本章中最大的那根杠杆。它同时也是会改变数学的那一个，所以我们先把它精确定义。

定义 21.1 (陈旧度)**.** 设 v 为当前正在被更新的权重版本索引，v_i 为生成样本 i 第 t 个词元的那个版本。该词元的陈旧度为 $\sigma_{i,t} = v - v_i \geq 0$。若某训练机制所训练的每一个词元都满足 $\sigma_{i,t} \leq s$，则称它具有陈旧度界 s。

关于定义 21.1有两点很容易被忽略。陈旧度是词元的性质，不是样本的性质：在生成期间更新权重的设计里，一条序列的早期词元比它的晚期词元更陈旧。另外，陈旧度

是权重上的差距，而第22章那个失配是在权重完全相同的情况下数值上的差距。两者是可叠加的，需要不同的处理，而把它们混为一谈是这个领域里最常见的概念性错误。

21.6.1 谱系

完全同步（$s = 0$）。 生成，然后训练，然后同步，然后重复。除去步内的小批切分，估计量正是节 6.1的在线策略估计量，除了常规的截断之外不需要任何修正。缓冲区深度为零，如 TRL (von Werra et al., 2020)。

落后一步（$s = 1$）。 在训练第 k 批的同时生成第 $k+1$ 批。这是能实现重叠的最小异步量，而且在一个比值截断本已能容忍的陈旧度上，它在挂钟时间上严格优于同步。verl 对自己实现的评价颇为坦诚：它「强制使用一轮异步训练的数据，不够灵活，也无法完全消除长尾对训练效率的影响」(verl Project, 2026a)。原因就在节 21.7：一步前瞻让生成器能开始下一批，但它并不能让生成器从任意的未来工作里 把一个正在排空的批次补满。

落后 k 步（$s = k$）。 两个池之间有一个深度为 k 的有界队列。训练器拿走任何已就绪的东西；生成器持续运行，只在队列满时阻塞。陈旧度由三种机制中的一种或多种来强制，它们彼此正交且经常组合使用：版本拒收（给每个样本打上版本标签，若 $v - v_i > s$ 就丢弃，如 NeMo-RL 的 `max_trajectory_age_steps`）、深度限界（给队列封顶，从而隐式地给陈旧度封顶，如 SkyRL 的 `max_staleness_steps`），以及重要性采样修正（照样在陈旧样本上训练，但给它重新加权）。Prime-RL 三者叠加，框架综述认为这是最稳健的生产配置 (Prime Intellect, 2026; Async RL Survey, 2026)。

完全异步。 用一个带淘汰策略的回放缓冲区，而不是一个带屏障的队列。采样工作进程持续生成；训练工作进程只要有一批可用就消费。AReaL 是参考系统 (Fu et al., 2025a)，报告在 GPU 数相同、最终性能相当或更好的情况下，相对同步系统最高提速 2.77×。把 AReaL 的 `max_head_offpolicyness` 设为零，就恢复出你正在跑的那个算法的同步版本，这对框架来说是一条好性质，对做实验的人来说是一个好习惯：任何与陈旧度有关的断言，都应当对照同一代码路径上的 $s = 0$ 运行来检查。

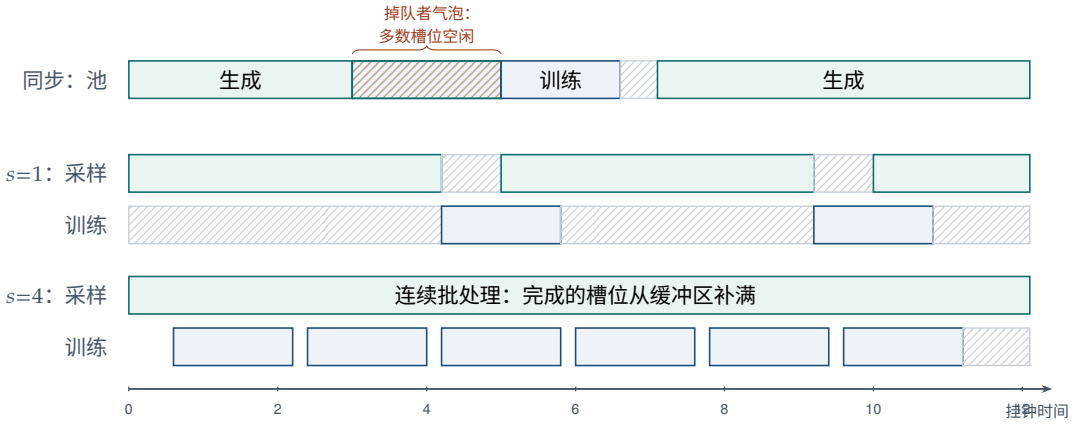


图 21.2: 示意性的占用时间线，不是实测。青色是生成，蓝色是梯度计算，斜线阴影是空闲。同步执行把各阶段串行化，并且额外浪费掉那块带阴影的掉队者气泡，在那段时间里大多数解码槽位是空的、而该批次尚未完成（节 21.7）。落后一步的重叠消除了串行化但没有消除气泡，因为生成器手上只有一批可做的工作。带连续批处理的有界陈旧度把两者都消除了，因为一个完成的槽位可以从接下来 s 步的提示中任取一个来补满——代价是训练在最多陈旧 s 个版本的数据上。

21.6.2 生成途中的权重更新

粒度最细的设计会在生成期间更新权重。PipelineRL 在前向传播之间交换权重，于是生成用新参数不间断地继续，且不丢弃任何在途序列 (ServiceNow Research, 2025)；Ai2 的 OLMo 3 RLVR 用它来重叠生成与训练，报告周期时间大约减半、整体提速接近 4× (Team OLMo, 2025)。Magistral 作了同样的选择，并直白地描述了后果：训练器在生成途中经 NCCL 把新权重广播给生成器，既不丢弃在途序列，也不显式刷新 KV 缓存，所以单条序列可能早期词元是在旧权重与一份陈旧缓存下算出来的，而后期词元是在新权重下算出来的 (Mistral-AI et al., 2025)。在定义 21.1之下，这样一条序列拥有的是一份逐词元的陈旧度剖面而不是一个陈旧度，而它的词元可以横跨好几个版本——AReaL 报告在陈旧度界为八时训练依然稳定，这对于这类设计能把它推到多远来说是一个合理的数量级 (Fu et al., 2025a)。

介于两个极端之间的是部分采样的处理，它实际上是关于「你在同步点如何对待在途状态」的粒度问题。从最细到最粗 (Async RL Survey, 2026)：在前向传播之间交换（PipelineRL）；把前缀存在 KV 缓存里，更新之后再恢复（verl）；中止该请求并从

提示重新提交（SkyRL、slime）；在训练步边界上阻塞（ROLL）。每一个更粗的选择都会丢弃更多已完成的工作。

21.6.3 吞吐模型

设 T_g 与 T_t 为一步所需数据在各自池上的生成时间与训练时间，并设 $u = \mathbb{E}[L]/L_{\max}$ 为节 21.7将要为同步批次推导出的解码槽位利用率。纯粹的阶段重叠给出

$$S_{\text{overlap}} = \frac{T_g + T_t}{\max(T_g, T_t)} = 1 + \min\Big(\frac{T_t}{T_g}, \frac{T_g}{T_t}\Big) \le 2, \tag{21.4}$$

在两个池完美平衡时取到最大值 2×；而对表 21.1的 $T_g : T_t$ 比值而言，只值 1.1–1.2×。因此单靠重叠是一根令人失望的杠杆。更大的收益来自尾部吸收：有了一个深度为 s 的缓冲区，一个正在排空的批次可以从未来若干步的提示中补满，于是有效解码利用率从 u 向 1 上升。把那个利用率记作 $\eta(s)$，

$$S(s) = \frac{T_g + T_t}{\max(T_g/\eta(s),\ T_t)}, \qquad \eta(s) = \min\Big(1,\ u + (1-u)\tfrac{s}{s+1}\Big). \tag{21.5}$$

η 的函数形式是建模上的方便之举；它的形状则不是。它在小 s 处陡升然后饱和，所以可得吞吐的绝大部分在 s 为二到四的量级就被拿到了，而陈旧度的统计代价却一直在增长。图 21.3 勾勒出由此产生的内点最优。已报告的提速与这个形状一致：AReaL 相对同步基线为 2.77× (Fu et al., 2025a)，OLMo 3 的 RL 基础设施约为 4× (Team OLMo, 2025)，以及框架综述中所称的从约 60% 提升到 95% 以上的利用率 (Async RL Survey, 2026)——最后这个数字应当被当作指示性的，因为我们不知道有任何独立复现。

核心思想: 陈旧度是权重上的差距，而它必须在统计上付账

当 $s > 0$ 时，生成你数据的那个行为策略 μ 是 s 个优化器步之前的 π_θ，那是一个不同的分布，所以朴素的策略梯度是有偏的。补救办法是Chapters 12 and 22的那套离线策略机制：带截断的重要性加权，或者 AReaL 所用的解耦目标，它把离线策略修正（相对于行为策略的一个重要性权重）与信赖域（一个近端策略）分开，也正是它让 AReaL 能容忍陈旧度 8 (Fu et al., 2025a)。近端策略也有便宜的近似——A-3PO 用一个版本感知的对数线性插值取代那次额外的前向传播，报告在性能相当的情况下提速 1.8× (A-3PO, 2025)。忘掉这一点会导致两种错误。开了异步却不开修正，产生的是一次更快也更差的运行，而其退化温和到足以被误认成超参数问题。开了一项针对数值失配的修正，并假定它同时也能处理八这么大的

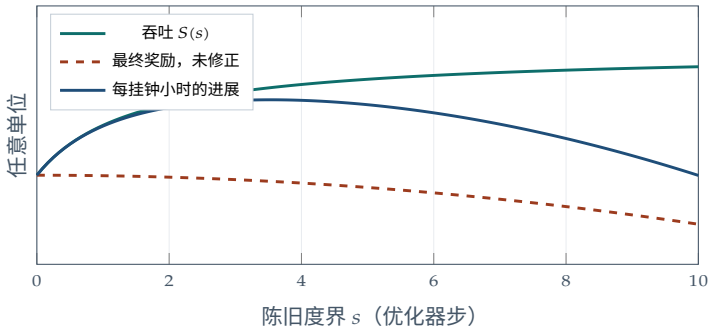


图 21.3: 陈旧度权衡的示意性、无量纲图解；这些曲线画自 (21.5)的模型与一个二次退化的拟设，而不是来自数据。吞吐随 s 饱和，因为一旦缓冲区里存了几步的工作，尾部吸收就几乎完成了，而统计惩罚却继续增长。乘积在小 s 处有一个内点最大值，这与「把陈旧度限制在个位数步而不是不设界地跑」这一已报告的实践相符。离线策略修正（第22章）会把虚线压平，从而把最优点往右挪。

> 陈旧度，同样是错的：为一个集中在一附近的比值分布校准过的截断重要性采样，在比值真的散开时表现很糟。

21.7 掉队者问题

异步是解药。本节是病。

RL 之下的回复长度是重尾的，而且随着训练推进会越来越重尾，因为策略学会了在难题上想得更久。Magistral 量化了这一散布：最长的补全所花的时间可以达到最短补全的五倍，且该分布在训练过程中会移动 (Mistral-AI et al., 2025)。一个同步批次在它最慢的成员完成之前无法被消费，所以这一批的成本由一个最大值决定，而它的价值由一个均值决定。剩下的交给次序统计量。

命题 21.2 (同步批次的解码利用率)**.** 考虑并发生成的 n 条序列，其长度 L_i 独立地抽自一个支撑在 $[0, L_{\max}]$ 上的分布，其中 $L_{\max}$ 是一个硬性生成上限。假设该批次一直占用其 KV 分配直到最后一条序列终止，且期间不接纳新工作。那么，所分配的解码槽位步中执行了有用计算的比例为

$$u \;=\; \frac{\mathbb{E}[L]}{\mathbb{E}[\max_i L_i]} \;\xrightarrow[n\ large]{}\; \frac{\mathbb{E}[L]}{L_{\max}}$$

只要 $\mathbb{P}(L = L_{\max}) > 0$。

证明. 在解码步 ℓ，仍在生成的序列数为 $\sum_i \mathbb{1}[L_i > \ell]$，期望为 $n\ \mathbb{P}(L > \ell)$。对各步求和直到最后一次终止，期望的有用槽位步为 $n \sum_{\ell \geq 0} \mathbb{P}(L > \ell) = n\ \mathbb{E}[L]$，而分配出去的槽位步为 $n\ \mathbb{E}[\max_i L_i]$。若有正比例 p 的序列在上限处被截断，则 $\mathbb{P}(\max_i L_i = L_{\max}) = 1 - (1-p)^n \to 1$，故分母趋于 $L_{\max}$。 □

命题 21.2是初等的，其后果却很严重。在 32k 上限、均值 8k 的情况下——一个普通的 RLVR 配置——生成阶段的解码槽位利用率约为 25%，而任何核函数优化都收不回它，因为那些槽位是真的空着。这就是图 21.2里画出来的掉队者气泡。还要注意这条命题对「提高上限」这一流行补救办法说了什么：提高 $L_{\max}$ 好让策略想得更久，除非 $\mathbb{E}[L]$ 同比例上升，否则会降低 u，所以一次改善了准确率的上限提高，可能同时把吞吐拖低到比准确率所值更多的程度。

饱和之前的尾部行为也值得知道。若长度服从 Pareto 尾 $\mathbb{P}(L > \ell) = (\ell_0/\ell)^\alpha$、指数 $\alpha > 1$，则 $\mathbb{E}[\max_{i \leq n} L_i] \approx \ell_0\, n^{1/\alpha}\, \Gamma(1 - 1/\alpha)$，而 $\mathbb{E}[L] = \alpha \ell_0/(\alpha - 1)$，于是利用率按 $n^{-1/\alpha}$ 衰减：更大的批次严格更差，而在 $\alpha = 2$、$n = 512$ 时，无上限的利用率会只有百分之几。真实的长度分布，与其说是干净的 Pareto，不如说是一个主体（大致对数正态）加上一个位于上限处的原子，这就是为什么在实践中该用的是命题 21.2的有上限分析——但那个关于 n 的标度解释了为什么问题恰好是在人们把组大小与批量做大时变糟的。

21.7.1 各种补救办法，以及每一种在统计上的代价

表 21.4作了汇总。要紧的是最后一列：这五种里有三种在统计上不是免费的，而其中两种还经常被当作免费的来部署。

其中两条与前面的材料相连。DAPO 所用的动态采样（节 12.4）已经会丢弃奖励方差为零的组 (Yu et al., 2025c)：这是一条出于统计理由选定、却带来系统后果的丢弃策略，因为生成器此后必须按一个取决于当前通过率分布的倍数来超量生产。而推测解码 (Leviathan et al., 2023)，常被提议作为采样加速器，除非它的接受检验在所用温度下严格保分布，否则会改变有效采样分布。我们不知道有任何关于它如何与采样—训练差距相互作用的审慎公开研究，并且会把这里的吞吐宣称当作未经审计的来对待。

常见陷阱: 本章中最诱人的偏差

「我们生成 1.25× 的批量，并在最先完成的 n 条上训练」是一套快速的 RL 系统变成一套错误的 RL 系统最常见的单一途径。这样一来训练数据就是策略的一个长

表 21.4: 针对掉队者气泡的各种补救办法，以及每一种引入的偏差。要紧的区分在于：有些补救办法改变的是工作何时完成（免费的），而有些改变的是哪些样本最终进入批次（不免费）。第二类里的任何补救办法，都需要一个显式的修正，或者一个能把它中和掉的设计，如 Magistral 的定补全数批处理。

补救办法	机制	偏差代价
连续／在途批处理	只要有槽位空出来就接纳一条新序列，而不是在批次边界上接纳 (Yu et al., 2022b; Kwon et al., 2023)。	无，前提是训练批次由固定的补全条数、而不是由一个截止时刻来定义。时间一到就把批次切掉，会挑选出较短的补全。
把部分采样带到后面继续	在权重同步时保存前缀（理想情况下连同 KV 缓存），事后再恢复 (Kimi Team, 2025a; Fu et al., 2025a)。	真实存在。该轨迹并非抽自任何单一策略，而是抽自若干版本上的一个混合，而标准的逐词元修正并不考虑序列内部的版本切换。此外还留下了在旧权重下算出的 KV。
按长度分桶的批处理	把预期长度相近的序列分到一组，使一个微批内的最大值接近其均值。	间接但真实。长度与难度相关，所以各桶并不具代表性；而且一条切开某个 GRPO 组的桶边界会改变该组的基线。要分桶就分组，绝不要分样本。
推测式提前终止	停掉一条被预测会失败的序列，或者一条已经进入重复循环的序列。	真实且往往很大。它对长度分布作了截尾，并且需要为被截断的样本补一个奖励；补零会加上一项隐式的长度惩罚，还可能教会策略提早结束。
超量生成后丢弃	生成得比你需要的多，留下最先完成的那些。	真实且容易被忽略：「最先完成」意味着「最短」，所以训练分布系统性地比策略本身更短。无偏的变体丢弃的是随机选中的整个组，而不是按到达顺序挑出的补全。

> 度有偏样本，而这恰恰是 RLVR 实践者花时间去设法控制的那个量（节 12.7）；你在基础设施里装上了一项隐式的简短性惩罚，而那里没人会去找它。Magistral 把「不存在长度偏差」当作一条硬性设计约束，把批次定义为固定条数的补全，并在小批内部贪心地拼出定词元数的微批，使得训练数据的补全长度分布与真实分布相匹配 (Mistral-AI et al., 2025)。如果你要超量生成，就以整个提示组为单位、按均匀随机选取来丢弃，并把你所训练的数据的长度直方图与你所生成的数据的长度直方图记录下来对照。如果两者不同，你有的是一个 bug，不是一项优化。

21.8 混合专家的系统问题

2026 年大多数前沿开放权重模型都是稀疏混合专家，而 MoE 后训练比它的稠密对应物难得不成比例，理由有四条，而且全是系统上的理由。

专家并行不是可选项，而且它会限制你的后端。一个专家按专家并行分布开来的 MoE 需要 all-to-all 的分发与合并。DeepSpeed ZeRO-3 不支持专家并行，所以在它之下训练的模型必须在每一步 all-gather 每一个专家，这把稀疏性的全部意义都丢掉了 (Async RL Survey, 2026)。如果你在大规模上训练 MoE，走 Megatron 这条路实际上是必须的 (NVIDIA, 2026b,a)，而单单这一条约束就把若干本来还算合理的框架排除在考虑之外。

路由是一个离散决策，所以引擎与训练器可以是范畴性地、而不只是数值性地不一致。一个极小的浮点差异就会翻转一次 top-k 选择，而这个效应随深度复合。实测的严重程度令人震惊：每次前向传播大约有 10% 的路由器不一致，且有 94% 的词元至少在一层上有差别；而在 Qwen3-30B-A3B 上，三次基线 GRPO 运行三次都崩溃，而所有带采样路由回放的运行都跑完了 (Ma et al., 2025)。这样一来你损失中的重要性比值 ρ 就是两个不同子网络之间的比值，而那并不是估计量所假定的那个量。第22章讲修法，包括「回放路由隐式地优化的是一个与你写下的那个略有不同的策略」这一告诫 (Stabilizing RL, 2025)；节 12.10讲估计量一侧的替代方案。这里我们只指出系统上的要求：逐词元逐层地记录并从引擎发送一份 top-k 掩码给训练器，是一笔不小的元数据量，而承载它的管路必须先于算法存在。

生成会在各专家之间造成负载不均。路由依赖于数据，而同一组内的采样彼此高度相关，因为它们回答的是同一个提示。因此生成期间的专家负载既是倾斜的，又是突发的，这与在打乱过的网页文本上做预训练时的情形不同，于是最慢的那个专家并

行 rank 决定了每一个解码步的节奏。这是嵌套在第一个掉队者问题内部的第二个掉队者问题，而且它对节 21.7的那些补救办法没有反应。

是两个目标，不是一个。MoE 后训练必须在维持任务质量之外，同时维持专家利用率。负载均衡损失与免损失的均衡方案以不同的取舍来处理利用率，而辅助的均衡梯度到底帮忙还是添乱，是经验性的、且因模型家族而异。不要从另一个模型家族继承辅助损失系数，并且要把专家利用率与奖励一起测量：只调其一而无视其二，会产生一些在奖励上看起来很健康、实则已经悄悄坍缩到只剩一小部分专家的运行。

基于适配器的训练在这里也救不了你。LoRA 在稠密模型上每层产生寥寥几个适配器；在一个 64 专家的 MoE 上则产生数百个，分散在各专家并行 rank 上，而只同步适配器就变成了一次分布式聚合，而不是一次小文件拷贝。截至 2026 年中，这是一个悬而未决的工程问题，而不是一个已解决的问题。

21.9 框架图景，2026 年中

表 21.5是有意带立场的。这个生态里至少有十六个在活跃开发的 RL 库 (Async RL Survey, 2026)，而一份四平八稳的比较会毫无用处。两点告诫：这是一个每隔几个月就重组一次的领域的一张快照，而下面的「擅长」指的是「这活儿我们会选它」，不是「它做不了别的」。

有三点观察是表格承载不了的。

各智能体技术栈正在收敛到一种形态，而不是一份功能清单。SkyRL 把数据平面与控制平面分开是对此最清楚的表述：生成流量通过对会话标识符作一致性哈希，在各引擎副本之间做负载均衡，使得一条轨迹的所有轮次落在同一个后端上并命中它的前缀缓存，而诸如暂停与权重更新之类的控制操作则直接扇出到每一个副本 (Griggs et al., 2025)。Prime-RL 把同一思想拆分得不同，由一个 CPU 编排器跨环境掌管数据平面，并在不于轮次之间重新分词的情况下驱动多轮次采样 (Prime Intellect, 2026)。这是值得研究的两个设计，而前缀缓存或基数缓存是最要紧的那项优化，因为每一个轮次都在延长一个共享前缀。

Miles，slime 的一个生产分支，是数值对齐工作的落脚处：端到端统一 FP8、批不变核函数、路由回放，以及一个以引擎与训练器逐位一致为目标的「真正在线策略」模式 (radixark, 2026)。那份议程属于第22章；在这里要紧的是，如今已经存在一个全部价值主张就是精度纪律的分支。

表 21.5: 截至 2026 年中的开源 RL 后训练框架，按各自所作的架构选择、而不是按功能清单来刻画。「MCore」是 Megatron-Core，「分离」是分离式，而「陈旧度」是该框架为限制陈旧度所提供的机制（定义 21.1）。汇编自Async RL Survey (2026) 的框架综述、Griggs and Moritz (2025) 的 Anyscale 对比，以及各项目自己的文档。

框架	放置	训练器	陈旧度	何时选它
verl	两者皆可	FSDP2, MCore	队列、IS	最大规模的生产运行、MoE、最高吞吐、最广的模型覆盖。默认答案，代价是一个非常庞大的配置面 (Sheng et al., 2025)。
NeMo-RL	两者皆可	FSDP2, MCore	版本拒收	你在 NVIDIA 技术栈上，需要六路并行或端到端 FP8，并且想要厂商支持与干净的接口 (NVIDIA, 2026c)。
slime	分离	仅 MCore	双缓冲	用一份小而可读的代码库做前沿规模的 MoE，灵活性在数据生成而非后端上；训练出了 GLM 系列 (THUDM / Z.ai, 2026)。
AReaL	分离	FSDP2, MCore	offpolicy-ness	极端异步、长尾智能体采样，或者以陈旧度为对象的研究。这里运维起来最复杂的系统 (Fu et al., 2025a)。
SkyRL	两者皆可	FSDP, MCore	深度限界	需要替换组件的多轮次智能体 RL；会话感知路由把一条轨迹的所有轮次钉在同一个后端上 (Griggs et al., 2025)。
Prime-RL	分离	FSDP2, 无 PP	三者兼备	生态中最严格的陈旧度纪律，以及在共享的训练与推理部署上做多租户 LoRA 训练 (Prime Intellect, 2026)。
OpenRLHF	两者皆可	DeepSpeed	队列	带奖励模型与评论家的经典 RLHF、异构集群，以及一个你一个下午就能读完的 Ray actor 架构；对 MoE 很弱 (Hu et al., 2024a)。
ROLL	两者皆可	DeepSpeed, MCore	步边界	显式的环境抽象，以及跨极不相同的用户规模的可配置性；这里权重更新粒度最粗的一个 (Alibaba ROLL Team, 2025)。
TRL	同置	Accelerate	无	在 HuggingFace 生态内做教学、原型与单节点工作。上课时该选的第一个框架；不是跑 70B 时该选的那个 (von Werra et al., 2020)。
Tinker	托管	—	—	你完全不想碰基础设施。按设计只支持 LoRA，而这正是让一个算力池能在多个租户之间共享的原因 (Thinking Machines Lab, 2025)。

Tinker 值得研究，哪怕你从不用它，因为它证明了后训练算法与其基础设施可以被干净地分开：你在一台纯 CPU 机器上针对四个原语写一个循环——前向反向、优化器步、采样、保存与载入——而切换模型规模只是改一个字符串 (Thinking Machines Lab, 2025)。从那个视角看，本章的一切都是别人系统的实现细节。这层抽象在前沿规模上是否还成立——在那里有意思的决策恰恰就是它所隐藏的那些——则是真正悬而未决的。

21.10 调试一次缓慢或坏掉的运行

RL 运行失败的方式是稠密训练不会有的，因为活动部件更多，也因为失败往往表现为奖励的缓慢劣化而不是一次崩溃。下面是一套流程，不是一份清单：让它有用的正是它的顺序。

工程实践：有序的诊断流程

1. **先拿到一份阶段时间线。**在提出任何假设之前，先记录每个阶段的挂钟时间与每个阶段的每 GPU 占用率（代码 21.3）。大多数人跳过这一步，然后在错误的阶段上花掉一天。信息量最大的单一数字是：训练 GPU 什么也没干的那部分步时间占比。
2. **与算术对照。**用节 21.1的 FLOP 与带宽模型算出预期时间。如果反向传播在屋脊线的 2× 以内而生成差了 5×，那你的问题在生成，而生成问题几乎总是占用率问题，不是核函数问题。
3. **看长度直方图，不要看均值。**画出补全长度的完整分布，以及撞上上限的比例。然后由命题 21.2算出 $u = \mathbb{E}[L]/L_{\max}$，并与实测的解码占用率作比较。如果两者吻合，你的问题就是掉队者长尾，而修法在节 21.6，不在调参。
4. **检查 KV 预算。**由(21.1)算出 $b_{\max}$，并与引擎实际达成的并发作比较。差距很大就意味着，本可以释放的内存正被训练器占着，或者引擎的内存比例设得太保守。
5. **去验权重，不要验损失。**在一次同步之后运行 代码 21.2的对数概率探针。在调查任何疑似的算法问题之前先做这个，因为一次重分片或精度故障伪装成算法问题伪装得天衣无缝。
6. **检查陈旧度分布，而不是那个界。**记录实际观测到的 $v - v_i$ 的直方图。只要生成器落后，配置的界与实现出来的陈旧度就会分道扬镳，一次为 $s = 2$ 设计的运行可能正悄悄地跑在 $s = 15$ 上。

表 21.6: RL 后训练系统中的症状及其常见成因。右列按我们会去调查的顺序列出成因；第一条正确的次数比其余各条加起来还多。

症状	可能成因，按顺序	判别性检查
生成阶段 GPU 利用率低	掉队者长尾（命题 21.2）；KV 预算太小以致并发被封顶；引擎内存比例设得低；提示预填充被串行排在解码后面。	画出活跃序列数对解码步的曲线。逐级下降的阶梯是长尾；一条平的天花板是内存。
训练阶段利用率低	微批太小；没有序列打包以致填充占主导；重算过度；流水线气泡。	把每秒词元数与屋脊线作比较；检查填充比例。
在第 N 步而不是第 1 步显存溢出	回复长度已经涨过了激活或 KV 预算所假定的范围；可变形状带来的分配器碎片；前缀缓存增长；轨迹缓冲区泄漏。	记录每步的最大补全长度与峰值预留内存。两者都增长是长度；只有预留增长是碎片。
吞吐随训练劣化	长度增长（通常就是它，而且是经由命题 21.2的一个 u 问题）；缓冲区背压抬高了实现出来的陈旧度；工具调用或环境延迟上升；缓存抖动。	用词元数而不是步数来归一化吞吐。如果每秒词元数是平的而每小时步数在下降，那除了长度之外什么问题也没有。
一次权重同步之后立刻发散	重分片置换错误；同步路径中的精度或 dtype 失配；部分采样在一份陈旧 KV 缓存之上恢复；MoE 路由在新权重下变了。	在一批固定数据上，于同步前后分别运行代码 21.2的对数概率探针。
奖励先升后崩；KL 尖峰	离线策略修正缺失，或相对于实现出来的陈旧度调得不对；MoE 路由失配；数值失配随长度增长（第22章）。	记录逐词元的 $\log \pi_\theta - \log \mu$ 分布。尾部变宽先于崩溃出现。
间歇性的 NCCL 超时	某个 rank 在集合通信区内阻塞于一次环境或验证器调用；动态批处理下各数据并行 rank 的微批数不均。	检查是不是每个 rank 每步都进入同样多次集合通信。

7. **通过去掉异步来二分。**把陈旧度界设为零，重跑一百步。如果病症消失，那是陈旧度或修正的问题。如果它还在，那就在估计量、奖励，或者数据里。

代码 21.3: 一个最小的阶段计时与占用率框架。它记录而核函数分析器不会记录的三样东西是：阶段边界、每个阶段的峰值预留内存，以及各解码步上的活跃序列数——三者合起来能立刻辨认出掉队者长尾。

```
import time, contextlib, collections, torch

class StepProfile:
```

```
    def __init__(self):
        self.phases = collections.OrderedDict()
        self.active = []          # 每个解码步的活跃序列数

    @contextlib.contextmanager
    def phase(self, name):
        torch.cuda.synchronize()                    # 没有这一行，阶段计时都是谎话
        torch.cuda.reset_peak_memory_stats()
        t0 = time.perf_counter()
        yield
        torch.cuda.synchronize()
        self.phases[name] = dict(
            seconds=time.perf_counter() - t0,
            peak_alloc_gb=torch.cuda.max_memory_allocated() / 2**30,
            peak_resv_gb=torch.cuda.max_memory_reserved() / 2**30)

    def report(self, lengths, cap):
        total = sum(p["seconds"] for p in self.phases.values())
        for name, p in self.phases.items():
            print(f"{name:<22} {p['seconds']:8.2f}s  {100*p['seconds']/total:5.1f}%"
                  f"  peak {p['peak_resv_gb']:6.1f} GiB")
        mean, mx = sum(lengths) / len(lengths), max(lengths)
        print(f"decode utilization u = mean/max = {mean/mx:.2f} "
              f"(cap-limited bound {mean/cap:.2f}); "
              f"truncated at cap: {sum(l >= cap for l in lengths)/len(lengths):.1%}")

# 用法
prof = StepProfile()
with prof.phase("generate"):   batch = engine.generate(prompts, n=G)
with prof.phase("reward"):     rewards = verifier(batch)
with prof.phase("logprob_ref"): ref_lp = trainer.logprobs(batch, model="ref")
with prof.phase("fwd_bwd"):    loss = trainer.step(batch, rewards, ref_lp)
with prof.phase("weight_sync"): sync_weights(trainer, engine, plan, group)
prof.report([len(y) for y in batch.responses], cap=32768)
```

本章小结

- 一套 RL 后训练系统在同一份权重上，把一种受内存带宽约束的负载（自回归生成）与一种受算力约束的负载（梯度计算）交错起来。解码无法被做成算力受限：它的算术强度等于并发批量，而在当前的加速器上，这个批量得超过大约 300 才能让单是权重读取这一项不再主导，而 KV 流量把这个要求推得更高。

- 生成约占算术运算的 17%、挂钟时间的 60–85%，所以它跑出来的模型 FLOP 利用率大约是反向传播的十分之一。实践上的推论是，一个算法的 FLOP 数几乎无关紧要，而生成吞吐几乎就是一切。
- 同置放置买到的是一个 GPU 池、免费的权重交接，以及严格在线策略的数据，代价是把两种负载串行化，并为每一步付出固定的休眠／唤醒成本。分离式放置买到的是独立扩缩与真正的重叠，代价是更多 GPU、为权重付出的网络带宽，以及陈旧度。像 HybridFlow 这样的混合单控制器设计，让算法可以被表达为一张数据流图，同时在每个节点内部保住 SPMD 的性能，这也是为什么写一个新的估计量如今是一处 Python 改动，而不是一处分布式系统改动。
- 训练器与推理引擎想要不同的并行布局，所以每一次权重同步都是一次重分片而不是一次拷贝。那次重分片花时间，是无声正确性 bug 最常见的落脚处，而且会改变浮点规约顺序——这使它成为第22章那个训练—推理失配背后的机制之一。
- 一次 RL 运行中的内存里，有一项开销能直接换来吞吐：KV 缓存，其每词元成本随模型深度而不是随参数量增长。在生成期间释放训练器的梯度、优化器矩与参考权重，就把模型状态换成了采样并发，而其回报随着运行内存有多紧张而增长——八块 GPU 上的 7B 约为 1.2×，三十二块上的 70B 约为 2×。
- 权重同步是最常被低估的成本。把朴素的逐参数广播分桶，据报告值约 25×，这是这个回路里现成可得的最大纯工程常数；对 LoRA 策略只同步适配器则把问题彻底移除。当同步成本没有被摊到一个长解码阶段上时它就要紧，而那恰恰是异步设计所造出来的那种情形。
- 异步应当被理解为一个陈旧度界，而不是一个布尔量。吞吐收益随这个界饱和，而统计代价一直在增长，所以有用的工作点是个位数的优化器步。陈旧度是权重上的差距，需要离线策略修正；它与权重完全相同时的数值差距是不同的问题，而为其中一个校准过的修正对另一个来说并没有校准。
- 掉队者气泡不是实现缺陷，而是次序统计量的后果：一个同步批次的解码槽位利用率等于补全长度均值与最大值之比，而在 32k 上限、8k 均值下这大约是 25%。更大的批次让它更糟。只有接纳来自当前步之外的工作——也就是异步——才能把它收回来。
- 三种流行的掉队者补救办法会让估计量有偏：部分采样产生的轨迹抽自若干策略版本上的一个混合，按长度分桶让批次不具代表性并可能破坏组基线，而按到达顺序丢弃的超量生成会在基础设施里装上一项隐式的简短性惩罚。要以整个提示

组为单位、按均匀随机来丢弃，并把你所训练的数据与所生成的数据的长度直方图记录下来对照。

- MoE 后训练难得不成比例，理由都是系统上的：专家并行直接排除掉了一些训练后端，路由是一个可能在引擎与训练器之间范畴性地不同的离散决策，同一组内相关的采样在生成期间造成突发的专家负载不均，而基于适配器的方法在专家并行之下会膨胀成数百个分片。

延伸阅读

先从Sheng et al. (2025) 开始。它是这个领域里唯一一篇真正关乎抽象而不是关乎某个基准数字的论文，而它引入的混合控制器思想如今已是默认。把它与 verl 自己关于转向采样服务器模式的发布说明 (verl Project, 2026b) 以及它的全异步训练器文档 (verl Project, 2026a) 一起读，后两者对自家早先那个落后一步设计的局限异常坦诚，也是任何地方对同步到异步这一演进最清晰的简短陈述。

至于异步本身，Fu et al. (2025a) 是参考文献：读它是为了那个可中断生成机制，以及那个让高陈旧度可以存活的解耦目标，并且要注意在那篇论文里系统贡献与算法贡献是不可分割的，这正是本章要旨的一个缩影。接着读A-3PO (2025) 看近端策略的廉价近似。然后读 Magistral 报告 (Mistral-AI et al., 2025)，它是关于一套生成途中更新权重的系统的最佳简短记述，而且出自一群担心偏差的人：他们坚持批处理不得在补全长度分布上引入偏差，正是那种把一套快系统与一套对的系统区分开来的约束。Team OLMo (2025) 是能看到一套异步 RL 技术栈嵌入在一次真实、可复现的模型发布中的最完整公开产物。

关于经典的底座，Rajbhandari et al. (2020) 与 Zhao et al. (2023a) 讲分片数据并行，Narayanan et al. (2021) 讲张量并行与流水线并行的组合，而Korthikanti et al. (2023) 讲那份决定长轨迹究竟装不装得下的激活内存分析。如果只读一篇，就读最后这篇：激活内存是长思维链 RL 最先撞上的那道约束，而它对「该重算什么」的分解是可以直接照做的。然后是Kwon et al. (2023) 与Zheng et al. (2024) 讲引擎，以及Yu et al. (2022b) 讲连续批处理，节 21.7依赖于后者。

要在整个生态里定位，框架综述 (Async RL Survey, 2026) 与 Anyscale 对比 (Griggs and Moritz, 2025) 是值得一读的两份二手资料，但要注意两者都包含我们未曾见到被独立复现的吞吐数字。最后，在你动手做 MoE RL 之前先读Ma et al. (2025)：它很短，它的测量结果令人警醒，而且它会替你省下一次崩掉的运行。

习题

21.1（时间都去哪了。）为一个你自选的配置复现表 21.1：挑定 N、B、G、L、$L_{\max}$、GPU 数，以及每 GPU 的带宽与 FLOP 数字。用屋脊线模型算出每个阶段的时间，然后算出 (a) 生成占 FLOP 的份额，(b) 它占挂钟时间的份额，以及 (c) 由此推出的生成与反向传播之间实际利用率之比。接着，在 $\kappa = 3$、节点内 B_{eff} 为 700 GB/s 的条件下，求出权重同步超过整步 10% 时的回复长度。

21.2（并发受深度限制。）两个模型参数量相同：一个是 32 层、隐藏维度 8192，另一个是 80 层、隐藏维度 5120；两者都用八个维度为 128 的键／值头。用(21.1)，计算在一个 640 GB、其中 140 GB 为驻留模型状态的池上，各自最多能并发多少条 9k 词元序列。然后估计固定一批 512 条序列时生成挂钟时间之比，并说明你关于吞吐如何随并发缩放的假设。当你事先就知道模型将要被 RL 后训练时，这对架构选型意味着什么？

21.3（卸载这笔交易。）把表 21.3扩展到 16 块与 64 块 GPU 上的 70B 模型。把「在生成期间卸载训练器状态」带来的提速画成 GPU 数的函数，并找出这个手法在多少块以上就不再要紧。然后用 (21.1)与(21.2)论证，为什么同一个手法在分离式部署里是有害的。

21.4（长尾的次序统计量，以及藏起它们要付的代价。）设补全长度为 $L = \min(L', L_{\max})$，其中 L' 服从对数正态，$\log L' \sim \mathcal{N}(\mu, \varsigma^2)$。(a) 把 $u = \mathbb{E}[L] / \mathbb{E}[\max_{i \le n} L_i]$ 推导成 $(\mu, \varsigma, L_{\max}, n)$ 的函数。(b) 固定 $\mathbb{E}[L] = 8000$，在 $L_{\max} = 32000$、$n = 512$ 下对 $\varsigma \in \{0.3, 0.6, 1.0\}$ 求 u 的值。(c) 现在有人提议把批量翻倍以改善梯度质量。在你那三种情形下分别量化其吞吐代价，并说明在关于梯度噪声尺度的什么条件下这笔交易是划算的。(d) 你现在想按倍数 $1 + \delta$ 超量生成以吸收长尾，同时不让长度分布有偏。设计一个方案，并证明它对于第12章的组相对估计量是无偏的；然后解释为什么那个显而易见的方案——生成 $\lceil (1+\delta)n \rceil$ 条补全并留下最先完成的 n 条——是有偏的，并计算在 $\varsigma = 0.6$、$\delta = 0.25$ 时被训练长度均值的偏移量。

21.5（陈旧度记账。）某次运行使用生成途中的权重更新。生成器总计产出 12,000 词元/秒；训练器每 40 秒完成一个优化器步；序列平均 8000 词元，上限为 32000。(a) 在生成一条平均长度的序列、以及一条撞到上限的序列的过程中，分别有多少个优化器步过去了？(b) 为两者分别写下逐词元的陈旧度剖面 $\sigma_{i,t}$。(c) 团队以 $s = 4$ 施

行版本拒收。撞到上限的序列有多大比例被丢弃，而这对训练数据的长度分布有什么影响？把你的答案与表 21.4联系起来。

21.6（**诊断这四种情形。**）对每一种，说出最可能的成因，以及能确认它的那一项测量。(a) 每秒词元数在训练过程中是平的，但每小时步数下降了 40%。(b) 某次运行可复现地在第 610 步显存溢出，其峰值预留内存比峰值已分配内存高 30%。(c) 奖励稳定了 300 步，然后持续下降，同时对 π_{ref} 的 KL 上升；该运行在第 300 步从陈旧度 1 切到了陈旧度 6，别的什么都没改。(d) 一次 30B 的 MoE 运行在启用一个更快的推理后端之后 50 步内崩溃，而同一份代码上的稠密 7B 运行不受影响。

21.7 ⋆ （**最优陈旧度。**）把(21.5) 与图 21.3中的拟设换成站得住脚的东西。把每步期望改进的退化建模成陈旧度界 s 的函数，用一个以 π_θ 与 s 步之前的 π_θ 之间的全变差距离表出的策略梯度偏差界，并让该距离受每步更新范数控制。再与你自己在某个指定长度分布下推导的 $\eta(s)$ 吞吐模型结合，导出使每单位挂钟时间的改进最大化的 $s^\star$。然后精确说出截断重要性采样放松了你的哪些假设，以及 $s^\star$ 因此如何移动。把你的 $s^\star$ 与Fu et al. (2025a) 报告的界八作比较，并说出他们的设置要满足什么条件两者才会一致。

21.8 † （**把时间线建起来。**）从表 21.5 中挑一个能跑在你手头硬件上的框架，以及一个小到可以快速迭代的模型。用代码 21.3或等价物给它装上仪表，并对一组固定的提示与固定的梯度步数产出三件产物：(a) 一份图 21.2那种风格的阶段占用时间线；(b) 实测的 $u = \mathbb{E}[L]/\,\mathbb{E}[\max L]$ 对实测的解码占用率，并对任何差异作出评述；(c) 在恰好只改动一件事之后的同样三张图——启用序列打包，或者把陈旧度界从 0 提到 2。报告每秒词元数、每小时步数，以及在固定词元预算下的奖励，并说明你的改动改善了这三项指标中的哪些、没有改善哪些。

21.9 † （**把你自己的权重同步弄坏。**）针对一对真实的引擎与训练器实现代码 21.2的对数概率探针。先为一次正确的同步确立基线 $\max|\Delta \log p|$。然后故意引入三个故障：(i) 把融合 QKV 投影的张量并行切分转置，(ii) 把某一层的权重经 FP16 而不是 BF16 转换，(iii) 在更新之后跳过 KV 缓存刷新。对每一个，报告探针值、生成看起来是否依旧流利，以及要过多少个训练步奖励劣化才会明显高出噪声。按「没有探针时要花多久才能找到」给这三者排序。

算法 21.1 一个带卸载与权重同步的同置 RL 步。第 3–6 行的次序是承重的：先唤醒引擎的权重缓冲区、再唤醒它的 KV 缓存，并在两者之间做同步，可以让峰值不同时包含训练器的优化器状态与引擎的缓存。

输入： 训练器布局下的策略 π_θ、提示 $\mathcal{D}_k$、组大小 G、引擎 E、训练器 T

把设备交给引擎
T：把梯度、AdamW 矩与 π_{ref} 权重卸载到主机内存
T：释放激活与集合通信缓冲区；归还缓存分配器
E.WAKE(`weights`) ▷ 只唤醒权重缓冲区，不含缓存
$\theta_E \leftarrow$ RESHARD(θ_T)；通过 IPC 句柄或设备拷贝发布
E.WAKE(`kv_cache`) ▷ 现在用剩下的内存分配缓存
生成
$\{y^{(i)}\}_{i=1}^{BG}, \log \mu \leftarrow E$.GENERATE($\mathcal{D}_k, G$)
把设备交还回来
E.SLEEP(level 2) ▷ 丢弃缓存与权重缓冲区
T：从主机内存重新载入 AdamW 矩与 π_{ref}
打分并更新
$r_i \leftarrow$ 对 $(x, y^{(i)})$ 运行验证器或 r_ϕ ▷ 可能的话与第 8 行重叠
$\log \pi_{\text{ref}}, \log \pi_{\theta_{\text{old}}} \leftarrow$ 前向传播；$A_{i,t} \leftarrow$ 组基线
for 打包序列的每个微批 **do**
    前向、反向、累积；按配置重算激活
end for
优化器步；$\theta_T \leftarrow \theta_T - \eta \hat{g}$
return θ_T

算法 21.2 带陈旧度界的异步训练。两个循环在分开的池上并发运行；缓冲区 $\mathcal{B}$ 是唯一的耦合。令 $s = 0$ 且 $C = 1$ 就恢复出同步算法，而任何与陈旧度有关的断言都应当对照它来检查。

输入： 陈旧度界 s、容量为 C 的缓冲区 $\mathcal{B}$、批量 n、版本计数器 $v \leftarrow 0$、已发布权重 $\theta^{(0)}$

生成器循环，在采样池上
while 训练尚未结束 **do**
  只要(21.1)的 KV 预算允许就接纳提示　▷ 连续批处理
  为所有驻留序列推进一个解码步
  if 新权重 $\theta^{(v')}$ 已发布 **then**
    在前向传播之间交换权重；保留 KV 缓存　▷ 生成途中更新
  end if
  for all 本步完成的序列 **do**
    把 $(x, y, \log \mu, \bar{v})$ 推入 $\mathcal{B}$，其中 $\bar{v}$ 是最后一个词元处的版本
    当 $|\mathcal{B}| = C$ 时阻塞　▷ 深度限界：隐式的陈旧度上限
  end for
end while
训练器循环，在训练池上
while 训练尚未结束 **do**
  $\mathcal{S} \leftarrow \emptyset$
  while $|\mathcal{S}| < n$ **do**
    从 $\mathcal{B}$ 弹出 $(x, y, \log \mu, \bar{v})$
    if $v - \bar{v} > s$ **then**
      丢弃　▷ 版本拒收
    else
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{\cdot\}$
    end if
  end while
  $r \leftarrow$ 在 $\mathcal{S}$ 上运行验证器；$A \leftarrow$ 在完整的组上取组相对基线
  $\rho_{i,t} \leftarrow \pi_\theta(y_t \mid \cdot) / \mu(y_t \mid \cdot)$，做截断　▷ 离线策略修正，第22章
  走一步或多步优化器；$v \leftarrow v + 1$
  把 $\theta^{(v)}$ 发布到采样池　▷ 代码 21.2
end while

第22章

采样与训练的失配

核心思想: 语言模型的「同策略」RL 其实悄悄是离策略的，而原因与 RL 毫无关系

生成采样轨迹的推理引擎与计算梯度的训练引擎，会在完全相同的权重下给同一个词元赋予不同的概率。这个分歧是一个数值问题——归约顺序、算子核选择、注意力后端、量化、混合专家路由——而它进入目标函数的位置恰好就是重要性比值所在的位置，于是它会腐蚀信赖域，并且在截尾权重类方法中还会腐蚀梯度的幅度。截至 2026 年年中，推理 RL 文献中报告过的最大单轴改进，来自把它修好 (Khatri et al., 2025)：比第12章中测量过的任何损失函数改动都更大。

第12章把这一条列为病理 P7 并推迟了讨论。这次推迟不是出于偏好：失配是那张清单上唯一一条无法用估计量的语言来讨论的，因为它的成因存在于两块独立优化的系统软件的浮点行为之中。我们扩展策略记号来给这个元凶命名，给出判定它是不是你的问题的诊断，然后逐一走过三族修正：重要性加权、修好数值、以及结构性回避。随后是两个相关问题——节 22.7的采样器契约，它在监控面板上呈现得一模一样但成因不同，以及陈旧度，失配总是与它混为一谈——最后是 2026 年的一次重新表述，它主张整个文献一直在优化错误的对象。

22.1 是四个策略，不是三个

本书到目前为止用了三个策略：当前策略 π_θ、上一次权重同步时的行为策略 $\pi_{\theta_{\text{old}}}$，以及冻结的锚点 π_{ref}。现代 RL 技术栈里还有第四个，而把它从记号中略去，正是这个问题被忽视了那么久的原因。

定义 22.1 (推理引擎策略)**.** 设 θ_{old} 是最近一次推送给采样工作节点的参数向量。记 $\pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t})$ 为训练引擎在这组权重下算出的条件概率，记 $\pi_{\text{inf}}(y_t \mid x, y_{<t})$ 为推理引

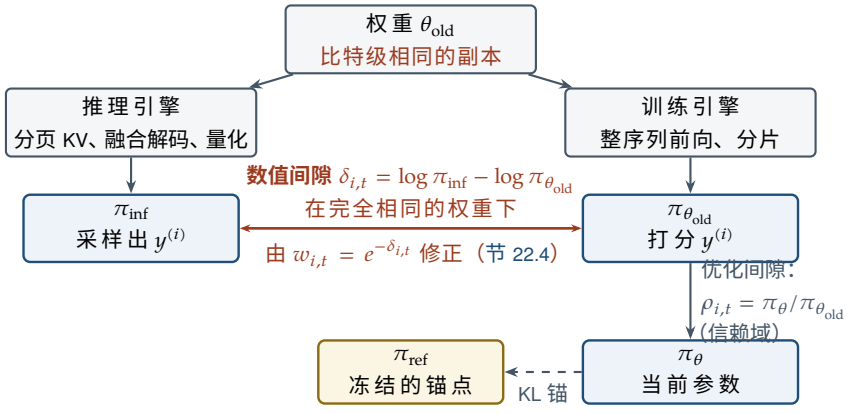


图 22.1: 一套解耦式 RL 技术栈中的四个策略。中间一行的两个方框持有同一组参数，差别只在于评估它们的软件。标准的策略优化理论只考虑右侧那条竖直的间隙，对中间那条水平的间隙则只字不提；本章几乎每一种病理，都源自那些假定该间隙为零的代码。

擎采样该词元时所用的概率：这是同一组比特级相同的权重上的两个不同函数。采样轨迹是从 π_{inf} 中抽取的，而 Chapters 6 and 12中的每一个目标函数都假定它们抽自 $\pi_{\theta_{\text{old}}}$。

图 22.1给出了由此得到的图景。其中住着三个间隙，而把它们分清楚就是本章的全部纪律：等权重下 π_{inf} 与 $\pi_{\theta_{\text{old}}}$ 之间的数值间隙、由内循环更新与陈旧度撑开的 $\pi_{\theta_{\text{old}}}$ 与 π_θ 之间的优化间隙，以及到 π_{ref} 的锚定间隙。只有第二个才是节 6.3的信赖域机器所要处理的东西。

22.1.1 分歧从何而来

底层基础是：有限精度下 $(a+b)+c \neq a+(b+c)$，于是每一次归约——一次点积、一个 softmax 分母、一次层归一化的均值——都依赖于各项相加的顺序，两份代数上完全相同的实现在数值上就是不同的。使它成为一个训练问题而非一件趣闻的，是两个引擎选择不同顺序的那些具体方式。表 22.1把它们列了出来；其中四条值得评说。

算子核选择与批次不变性。 He and Thinking Machines Lab (2025) 指出了起作用的机制，而它比「原子操作是不确定的」要微妙得多。单个算子核对固定的形状与调度确实是逐次可复现的；它们不具备的是批次不变性，因为一个算子核会依据批大小来改变它在各个核心之间切分工作的方式，而这个切分决定了归约顺序。既然一个服务引擎的批次构成取决于同时在途的其他请求，那么赋给你那个词元的对数概率就取决于

别人的流量。他们的头条测量是现有最干净的演示：在默认的 vLLM 下，向一个 Qwen 模型发出 1000 个完全相同的零温度请求，产生了 80 种不同的补全；换成批次不变的算子核后，1000 个全都相同。

表中间几行是同一主题的变奏：稠密的整序列前向对上分页、分块、增量解码的前向，八路分片的训练器对上二路分片的引擎。精度进来了两次，一次是在 $|\mathcal{V}| \approx 10^5$ 宽的语言模型头上，节 22.5发现那里有最大的廉价收益；另一次是在采样中：量化之所以诱人，是因为生成主导了墙钟时间（节 1.6），但它在每一种格式下都会放大间隙，LMSYS Org (2026) 报告 MXFP8 与 NVFP4 都高于 BF16。

混合专家路由。 这是定性上最糟的一行，因为路由是连续量的一个离散函数。一个 10^{-3} 的扰动若翻转了某个 top-k 的边界，就会改变哪些专家被启用，进而改变激活值，再进而扰动下一层的路由器。测得的严重程度很大：Ma et al. (2025) 报告每次前向大约有 10% 的路由器发生分歧，且 94% 的词元至少在一层上有差异；在 Qwen3-30B-A3B 上，三次基线 GRPO 运行全部坍塌，而所有启用了路由重放的运行都跑完了。Zheng et al. (2025a) 独立地报告，单次更新之后大约有 10% 的被激活专家发生改变。既然 2026 年大多数前沿开放模型都是稀疏的，这就是常见情形，而不是什么奇异情形。

表 22.1的最后一行根本不是数值问题，却表现得像一个：引擎可能施加了温度、top-p 或重复惩罚，然后报告的却是在这些调整之前算出的对数概率——或者干脆拒绝报告调整后的那些，直到Yao et al. (2025b) 给它打了补丁之前，vLLM v1 一直是这个状态。节 22.7专门讨论它。

人们很容易把 $\delta_{i,t}$ 建模成零均值的 i.i.d. 噪声，进而断定它会被平均掉，但「性质」一栏不允许这样读：路由翻转是离散的，且在整条序列上相关；布局与后端的效应在一次运行内是常数，因而是系统性的；采样器契约则产生一个符号由构造方式固定的偏差。一个在序列内部相关、在整次运行中系统性、并且重尾的量，恰恰是重要性加权处理得最糟的东西。

22.2 测量它

这里的一切都下游于一个测量，即逐词元的对数概率间隙

$$\delta_{i,t} \;=\; \log \pi_{\text{inf}}(y_{i,t} \mid x, y_{i,<t}) \;-\; \log \pi_{\theta_{\text{old}}}(y_{i,t} \mid x, y_{i,<t}), \tag{22.1}$$

表 22.1: 采样与训练之间分歧的来源。「性质」一栏区分了平滑扰动与离散翻转：前者的作用像是对数概率上的小幅加性噪声，后者则是重尾的、并且在一条序列内部相关。每一行都发生在完全相同的权重下；没有一行是陈旧度。

来源	机制	性质	缓解手段
归约顺序	加法不满足结合律；求和树不同	平滑	批次不变的算子核
算子核选择	切分策略随批大小与形状而变	平滑，依赖流量	固定切分尺寸
并行布局	TP/PP/CP 宽度改变部分和的树形	平滑	对齐布局，或做修正
注意力后端	分页／ flash 解码对上稠密整序列	平滑	统一 KV 访问、固定切分
预填充对上解码	T 步增量对上一次前向	平滑	同上
LM 头精度	10^5 宽的矩阵乘接一个 log-softmax	平滑，最大	FP32 头
采样量化	FP8/INT8/NVFP4 权重与 KV 缓存	平滑，很大	统一的精度契约
MoE 路由	top-k 翻转；随深度复合	**离散**，重尾	路由重放、序列级比值
采样器契约	截断采样，未加温度的打分	**系统性偏差**	按你采样时的分布打分
权重传输	同步时的转换或重量化	平滑，跨步相关	比特精确的量化器契约

其中第一项是引擎为它实际采出的那个词元所报告的数值，第二项是训练器在同一组权重下的重算结果。多记一个张量的账，换来的是一个标量，而它诊断出的失败比 RL 运行中任何别的量都多。

要报告 δ 的分布，绝不要只报它的均值：中位数、第 1 与第 99 百分位数、每条序列上的绝对值最大值，以及 $|\delta| > 0.05$ 的词元占比。然后报告由此推出的权重 $w_{i,t} = e^{-\delta_{i,t}}$、它们在序列层面的乘积，以及在那些序列权重上按 (2.17)算出的有效样本量。为什么分布比均值更要紧，值得推导一番。

推导: 有效样本量取决于方差，而不是均值

把逐词元的间隙建模为 $\delta_{i,t} \sim \mathcal{N}(m, s^2)$，且在 t 上独立。序列的对数权重 $\log W_i = -\sum_t \delta_{i,t} \sim \mathcal{N}(-Tm,\, Ts^2)$，于是 W_i 是对数正态的，其对数方差为 $\sigma^2 = Ts^2$。对于对数正态权重，$\mathbb{E}[W] = e^{-Tm+\sigma^2/2}$、$\mathbb{E}[W^2] = e^{-2Tm+2\sigma^2}$，因此由(2.17)得

$$\frac{\mathrm{ESS}}{n} \xrightarrow{n\to\infty} \frac{\left(\mathbb{E}[W]\right)^2}{\mathbb{E}[W^2]} = e^{-\sigma^2} = e^{-Ts^2}. \tag{22.2}$$

均值 m 精确地抵消掉了。一个系统性的偏移把每一个权重都按同一个因子重新标度，这对自归一化估计量是不可见的，而且——由于优势在组内做了中心化，而

表 22.2: 对逐词元间隙(22.1)的数量级预期，单位为奈特，并给出 (22.3)被违反时的序列长度。这些是解析得到的，实践中因技术栈而异；给出它们是为了让一位测到 $s = 10^{-1}$ 的读者立刻知道，那是出了错，而不只是不够完美。

配置	中位 $\lvert\delta\rvert$	离散程度 s	ESS 失效时的 T
稠密模型，BF16，FP32 头，布局对齐	10^{-4}–10^{-3}	$\sim 10^{-3}$	$\gtrsim 10^{6}$
稠密模型，包括头在内全程 BF16	10^{-3}–10^{-2}	$\sim 10^{-2}$	$\sim 10^{4}$
FP8 或 INT8 采样，未做修正	10^{-2}–10^{-1}	$\gtrsim 10^{-2}$	$\lesssim 10^{4}$
MoE 且路由实时计算	双峰	由尾部主导	任意
温度或 top-p 失配	10^{-2}–1	系统性，非随机	任意

Adam 近似是尺度不变的（节 2.9）——对更新本身也是不可见的。摧毁估计量的是离散程度。

核心思想: $1/\sqrt{T}$ 法则

(22.2)说，要把 ESS/n 保持在 0.3 以上，就需要 $Ts^2 \lesssim 1.2$，也就是

$$s \lesssim \frac{1.1}{\sqrt{T}}. \tag{22.3}$$

对 $T = 10^3$，这允许每词元约 0.035 奈特的标准差，几乎任何技术栈都能满足；对 $T = 32{,}000$——2026 年一个寻常的长思维链预算——它只允许约 0.006。这一条不等式就解释了时间线：失配在短回复的 RLHF 时代只是一件趣闻，而在回复长度达到 10^4 的那一刻就变成了一个一阶的工程问题；它也解释了为什么行得通的修法是减小 s，而不是容忍它——因为容忍度按 $T^{-1/2}$ 缩小，而人们对 T 的野心只会增长。

22.2.1 这些数字应该长什么样

表 22.2给出可以预期的数量级，以区间形式给出，而理由本身就颇具信息量：没有任何已发表的来源报告过一份规范的逐词元间隙分布，因为文献都是间接地测量失配的——测成一个训练—推理 KL，或者测成坍塌与否的结果。请测你自己的。

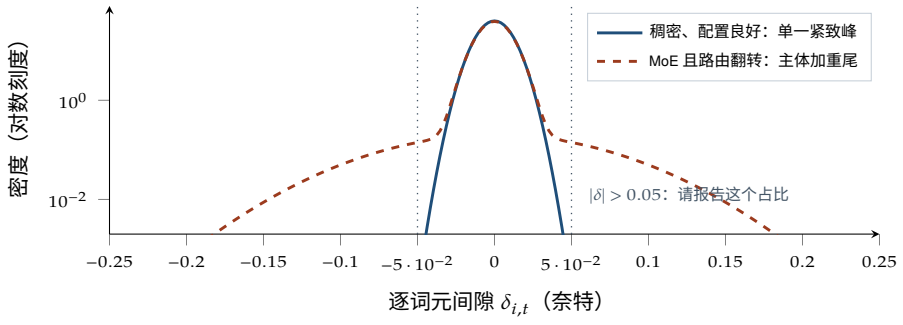


图 22.2: 逐词元间隙的密度示意图——并非实测——纵轴取对数刻度。两条曲线的均值与中位数几乎相同，但虚线那条的方差是两倍以上，因而按(22.2)其有效样本量要差得多。线性坐标轴会把这个差别完全掩盖掉，而这正是失配之所以常被称作「可忽略」的原因。

MoE 那一行打破了节 22.2的框架：当路由翻转时，δ 的分布变成一个紧致主体与一个宽阔成分的混合，正如图 22.2所勾勒的，而尾部恰恰是截尾与截断双双被激活的地方——于是修正机器恰好作用在那些权重最不可信的词元上。

工程实践: 每一步都要记录的五个数

1. $\delta_{i,t}$ 的中位数与 $\{1, 99\}$ 百分位数，以及每条序列上的 $\max_t |\delta_{i,t}|$。
2. $|\delta_{i,t}| > 0.05$ 的词元占比。
3. 在序列权重 $W_i = \exp(-\sum_t \delta_{i,t})$ 上按(2.17)算出的 ESS$/n$。
4. TIS 权重触到截尾上限的比例，词元级与序列级分别统计。
5. 对 MoE，逐层统计引擎与训练器之间的路由器一致率。

相对于一次前向，这五个都是免费的，而且这五个都先于奖励发生变动：一次在第 700 步奖励坍塌的运行，通常从第 300 步起就能看到第 3 项在退化。正是这一点，让 节 11.7中「先修数值，再改算法」的指示变得可操作，而不只是一句口号。

代码 22.1: 记录间隙以及由此推出的有效样本量。`logp_inf` 必须是引擎为它发出的那个词元给出的采样对数概率，`logp_train` 必须在反向传播所用的布局下重算，而掩码必须排除提示、填充和由环境写入的词元。

```
@torch.no_grad()
def mismatch_report(logp_inf, logp_train, mask):
    """logp_inf、logp_train、mask：[B, T]。返回一个待记录的标量字典。"""
    delta = (logp_inf - logp_train)[mask.bool()]            # 逐词元间隙，式 (22.1)
    q1, q50, q99 = torch.quantile(delta, torch.tensor([0.01, 0.50, 0.99]))
```

```
    # 序列权重 W_i = exp(-sum_t delta_it)；全程在对数空间中计算。
    log_w = -((logp_inf - logp_train) * mask).sum(-1)       # [B]
    log_w = log_w - log_w.max()                             # 取指数前先稳定化
    w = log_w.exp()
    ess = w.sum().pow(2) / (w.pow(2).sum() * w.numel())     # 落在 [1/n, 1]

    return {
        "gap/p01": q1.item(), "gap/p50": q50.item(), "gap/p99": q99.item(),
        "gap/std": delta.std().item(),
        "gap/max_abs_per_seq": ((logp_inf - logp_train).abs() * mask).max().item(),
        "gap/frac_gt_0.05": (delta.abs() > 0.05).float().mean().item(),
        "gap/ess_frac": ess.item(),
        # 式 (22.4)：在 ESS/n = 0.3 下这套技术栈所能支撑的最大 T
        "gap/max_safe_len": (1.1 / delta.std().clamp(min=1e-8)) ** 2,
    }
```

22.3 被腐蚀的比值为什么会毁掉训练

假设你什么都不做，像许多实现那样直接把引擎的对数概率当作(6.17)中的 $\log \pi_{\theta_{\text{old}}}$，因为这样能省掉一次前向。有两件截然不同的事会出错。

22.3.1 信赖域退化成一个筛选数值不稳定性的过滤器

在第一个内循环步上，$\theta = \theta_{\text{old}}$，所以对每个词元比值本应精确等于 1。实际上却是

$$\rho_{i,t} = \frac{\pi_\theta(y_{i,t})}{\pi_{\text{inf}}(y_{i,t})} = \frac{\pi_\theta(y_{i,t})}{\pi_{\theta_{\text{old}}}(y_{i,t})} \cdot e^{-\delta_{i,t}}, \tag{22.4}$$

$\underbrace{\phantom{\pi_\theta(y_{i,t})}}_{\text{你本来想要的}}$

于是比值从 $e^{-\delta_{i,t}}$ 起步，(6.13)中的截断在还没走过任何一个梯度步之前就可能被激活。更糟的是，它是有选择地激活：被推出带外的那些词元，正是 $|\delta|$ 最大的那些，而由表 22.1可知它们是路由发生翻转、或者注意力切分不同的词元——一个由系统内部机制决定、并且在序列内部相关的集合。既然 PPO 的截断会湮灭带外词元的梯度（节 6.3），一套未做修正的技术栈就在悄悄删掉自己训练信号中由系统所决定的一个子集。这是一条严格劣于MiniMax (2025) 在 节 12.5.2中所反对的那条选择规则：在那里，被选中的集合至少还是低概率的分叉词元，具有可解释的角色。

22.3.2 在截尾权重类方法中它腐蚀的是幅度

在 CISPO 式的目标(12.8)下，权重不是一道闸门而是一个系数：逐词元的梯度被乘上 $\mathrm{sg}[\min(\rho_{i,t}, C)]$。代入(22.4)可知，只要截尾没有生效，这个系数就会带上一个因子 $e^{-\delta_{i,t}}$。0.05 奈特的间隙就是逐词元学习率上的 5% 误差；一次产生 $\delta = 1$ 的路由翻转则是 e 倍。因此这两族权重处理方式的失效方式不同，而这个对比决定了你会看到哪一种症状：

- **截断**（湮灭）：失配改变的是哪些词元有贡献。症状：内循环第零步就有非零的截断比例，且它与序列长度相关，而不是与策略漂移相关。
- **截尾权重**（衰减）：失配改变的是每个词元贡献多少。症状：没有——这次运行看起来很健康——而实际的逐词元学习率上却带着一个重尾的乘性误差。

第二种更危险，恰恰是因为它更诱人：它没有任何蛛丝马迹。这就是节 12.5.2那条「截断权重，而不是截断目标」建议的代价，也正是为什么那条建议必须与 节 22.2的诊断配套使用。

22.3.3 它改变的是优化问题本身，而不只是方差

最自然的第一反应是：这一切不过是本已嘈杂的梯度之上的二阶噪声。反对这种读法的最强证据是 Zhong et al. (2026)，他们构造了一个零失配的诊断环境（VeXact），可以在其中孤立地引入间隙，同时固定住离策略漂移与每一种稳定化机制。他们报告，微小的词元级数值分歧能够 独立地引发坍塌，并主张失配「改变了实际的优化问题」，理应作为一阶因素进入稳定性分析。这是一项单篇论文的结果，但设计得很好：把数值与陈旧度解耦开来，正是早先的文献做不到的。

第二个观察指向同一方向。Tang et al. (2026) 在万亿参数尺度上报告，去掉 KL 惩罚会让训练—推理间隙无界地发散，随之而来的是熵坍缩与奖励崩溃。这是节 12.8那次调和所能拿到的最干净的论据：KL 项的部分价值就在于它限住了总漂移，而有界的漂移正是两个引擎的数值保持一致的前提。

推导: 为什么长度是放大器

节 2.1指出，一个序列级比值是 T 个词元比值的乘积，因而其动态范围随 T 指数增长；失配精确地继承了这份放大。在逐词元离散程度为 s 时，序列的对数权重具有标准差 $s\sqrt{T}$，于是权重的取值范围是 $e^{\pm s\sqrt{T}}$：在 $s = 0.01$、$T = 10^4$ 时这是 $e^{\pm 1}$，

而在 $s = 0.03$ 时是 $e^{\pm 3} \approx 20$。有两个后果容易被漏掉。一套在对话长度的回复上完好无损的技术栈，可能在长思维链上完全不可用，而代码一行都没改：失效是工作负载的函数，不是软件的函数。而且既然 δ 在序列内部相关，真实的标度比 $\sqrt{T}$ 更糟：在两两相关系数为 ϱ 时它趋近 $sT\sqrt{\varrho}$，是线性而非平方根，而 MoE 路由正是 ϱ 最大的地方。

22.4 修正一：截尾重要性采样

第一族修法接受这个间隙，并在统计上对它做修正。参照性的构造出自Yao et al. (2025b)，他们的表述——「实现层面的间隙隐式地把同策略 RL 变成了离策略 RL」——为整个领域给这个现象定了名。

历史注记: 这个领域采用最广的修正并不是一篇 arXiv 论文

Yao et al. 的工作以博客文章的形式存在，最早发表于 2025 年 8 月并修订至当年 10 月，另有一篇 NeurIPS 2025 的研讨会论文 (Yao et al., 2025b,a)。它没有 arXiv 编号；人们普遍按博客把它引作「Yao et al., 2025」。按作者自己的更新日志，几个月之内该方法就被集成进了 verl、slime、OAT、SkyRL、OpenRLHF 与 REINFORCE++。一项如今存在于大多数开源 RL 框架中的修正，是通过一个 Notion 页面传播开的。教训不是 arXiv 没有必要，而是在快速演进的系统工作中，真正改变实践的产物往往是那个补丁，而引用图谱要滞后现实大约一年——所以要预期在框架文档与 issue 讨论串里找到承重的结果 (SkyRL Project, 2026; Zhao et al., 2025b)。

22.4.1 这个估计量

既然采样轨迹抽自 π_{inf} 而目标函数定义在 $\pi_{\theta_{\text{old}}}$ 上，(2.15)规定了权重 $\pi_{\theta_{\text{old}}}/\pi_{\text{inf}} = e^{-\delta}$，而节 2.6告诉我们它必须被截尾。用于失配的截尾重要性采样是

$$w_{i,t} = \min\Big(\frac{\pi_{\theta_{\text{old}}}(y_{i,t}\,|\,x, y_{i,<t})}{\pi_{\text{inf}}(y_{i,t}\,|\,x, y_{i,<t})},\, C\Big) = \min(e^{-\delta_{i,t}},\, C), \tag{22.5}$$

这就是(2.19)在定理 22.1那两个策略上的实例化，而它是作为 *PPO* 损失上的一个乘性因子施加的：

$$\mathcal{L}^{\text{TIS}} = -\frac{1}{\mathcal{N}} \sum_{i=1}^{G} \sum_{t=1}^{|y^{(i)}|} \underbrace{\text{sg}[w_{i,t}]}_{\text{修补过去}} \cdot \underbrace{\min\Big(\rho_{i,t} A_{i,t},\, \text{clip}\Big(\rho_{i,t},\, 1-\varepsilon_{\text{low}},\, 1+\varepsilon_{\text{high}}\Big) A_{i,t}\Big)}_{\text{约束当下}}, \tag{22.6}$$

其中信赖域比值严格取自训练器一侧，$\rho_{i,t} = \pi_\theta(y_{i,t})/\pi_{\theta_{\text{old}}}(y_{i,t})$。

核心思想：两个比值，两份职责

(22.6)中含有两个重要性比值，它们绝不可合并。权重 $w_{i,t}$ 比较的是对 同一组权重的两次评估，因而是一个常数，与 θ 无关：它不携带梯度，它是对损失的一次逐词元重加权，它的职责是修补一个在更新开始之前就已发生的测量误差。比值 $\rho_{i,t}$ 比较的是不同的权重，是信赖域所作用的对象。把它们合并——即在 $\rho_{i,t}$ 的分母里放 π_{inf}——会让信赖域去针对一个任何优化步都永远到不了的策略施加约束，那就是(22.4)。用乘性的方式修正过去；只截断当下。

这就是 RL 系统文献所称的解耦 *PPO* 的三策略分解：一个生成了数据的行为策略（π_{inf}）、一个定义信赖域的邻近策略（$\pi_{\theta_{\text{old}}}$），以及一个被优化的目标策略（$\pi_\theta$）。它正是 V-trace 在分布式的行动者—学习者架构中所做的那种分离 (Espeholt et al., 2018)，也正是 AReaL 能够容忍高陈旧度的原因 (Fu et al., 2025a)。它的代价是为取得 $\log \pi_{\theta_{\text{old}}}$ 而多做一次前向，而想省掉这次前向的诱惑，是技术栈未做修正的最常见原因。

这里的截尾是衰减而非湮灭，这是节 12.5.2那条可推广的思路在流水线第二处的再现。它的偏差 $\mathbb{E}[(w - \min(w, C))\, g]$ 落在那些训练器给出的分数远高于引擎的词元上——由表 22.1可知，其中路由或注意力路径发生分歧的占比过高——所以它并非无害，但至少瞄准的是对的那个集合。

常见陷阱：两种会画出看似合理曲线的错误

把权重取反。用 $\pi_{\text{inf}}/\pi_{\theta_{\text{old}}}$ 而不是 $\pi_{\theta_{\text{old}}}/\pi_{\text{inf}}$，是一处只差一个字符的改动，它不是消除失配而是把失配翻倍，而蛛丝马迹是：启用修正之后 ESS 反而下降。请在一个诊断批次上验证，修正后的 ESS 高于修正前的。让 π_{inf} 混进诊断。如果它进入了所测的 $\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})$，那么报出来的 KL 在第零步就有一个 $\Theta(s^2 T)$ 的地板，而它会被误读成策略漂移。诊断量一律只用训练器一侧的量来计算。

22.4.2 词元级、序列级，与长度陷阱

有两种粒度在使用中，而这里的取舍与节 12.5中的并不相同。词元级 TIS 在每个位置上施加(22.5)；序列级 TIS 则为每条回复由乘积 $W_i = \prod_t e^{-\delta_{i,t}}$ 构造一个权重，并按一个序列上限截尾一次。在截尾之前，序列估计量是两者中更接近无偏恒等式(2.15)的那个，因为采样单元真的是序列。截尾之后，图景就反转了。

常见陷阱：长度陷阱

由节 22.3.3，$\log W_i$ 的标准差是 $s\sqrt{T}$：在 $s = 0.01$、$T = 10^4$ 时它是 1 奈特，于是 $C_{\text{seq}} = 2$ 的序列上限会截尾掉大约四分之一的序列，而在 $s = 0.02$ 时几乎会截尾掉全部。一个被截尾的序列权重是一个常数，所以在那些序列上修正已经退化成对损失的一次均匀重标度——根本不是修正，还外加一个偏差。因此，恰恰在失配最要紧的那些长度上，序列级 TIS 注入的偏差可能多于词元级 TIS。已报告的阈值从 2 到 10 不等 (Yao et al., 2025b; SkyRL Project, 2026)，取高值的存在理由正是为了不让截尾处处生效。如果你要用它，就记录截尾率；一旦高于百分之几十，你跑的就不是你以为的那个估计量了。

与之同胞的构造是**掩蔽式重要性采样**（masked importance sampling, MIS）：不去收缩极端权重，而是把差异超过阈值的词元丢掉，并对其余词元施加精确权重，以覆盖率上的偏差换取别处的精确性。slime 与 Miles 两个团队报告，它在训练后期抑制了失配的增长，同时保住了性能，并推荐把它作为一个稳健的默认选项 (Zhao et al., 2025b; radixark, 2026)。哪一个更可取，取决于那些分歧词元是否携带着你需要的信号：在路由翻转之下它们的概率毫无意义，掩蔽显然是对的；而在一个均匀带噪的量化采样之下，掩蔽丢掉的则是批次中随机的、可能还很大的一部分。

推动 TIS 被广泛采用的回报，并不是它在 BF16 技术栈上的稳定性，而是它给了量化的许可证：生成占据了墙钟时间的大半，所以 INT8 或 FP8 采样在端到端上值好几十个百分点，而不做修正的话，由此引入的失配会毁掉训练。Yao et al. (2025b) 报告 TIS 让量化采样变得可行，同时保住了下游准确率；FP8-RL (2026) 把 FP8 W8A8 采样与 FP8 KV 缓存同词元级 TIS 与 MIS 配合，以吸收残差。Jet-RL (2026) 从同一前提推向了相反的结论——在长生成上，BF16 训练配 FP8 采样「遭受严重的训练不稳定与灾难性的准确率坍塌」——因而偏好两侧统一的 FP8 流程。两种立场的分歧在于，他们宁愿付出偏差还是付出工程。Huang et al. (2025b) 走了第三条路，把量化噪声当作一种探索助力；对照节 12.6来读，这是自洽的，但其机制不受控制，幅度由一个硬件格式决定，而不是由一个你能退火的超参数决定。

代码 22.2: 把两个比值分开。`tis_w` 是被截断梯度的，比较的是对 θ_{old} 的两次评估；`ratio` 携带梯度，比较的是 θ 与 θ_{old}。传入 `logp_old = logp_inf` 就还原出那种常见但错误的合并形式。

```
def tis_ppo_loss(logp, logp_old, logp_inf, adv, mask,
                 eps_low=0.2, eps_high=0.28, tis_cap=2.0, level="token"):
    """logp: 当前策略（带梯度）。logp_old: 训练器 @ theta_old（无梯度）。
       logp_inf: 推理引擎 @ theta_old（无梯度）。"""
```

```
    with torch.no_grad():
        delta = logp_inf - logp_old                    # 式 (22.1)
        if level == "token":
            tis_w = (-delta).exp().clamp(max=tis_cap)  # 式 (22.5)
        else:                                          # 序列级：每行一个权重
            log_W = (-delta * mask).sum(-1, keepdim=True)
            tis_w = log_W.exp().clamp(max=tis_cap).expand_as(logp)

    ratio = (logp - logp_old).exp()                    # 只用于信赖域
    clipped = ratio.clamp(1.0 - eps_low, 1.0 + eps_high)
    per_token = torch.min(ratio * adv, clipped * adv)

    loss = -(tis_w * per_token * mask).sum() / mask.sum().clamp(min=1)
    return loss, {"tis/frac_capped": ((-delta).exp() > tis_cap)[mask.bool()]
                                     .float().mean().item()}
```

上面那个分解是干净的答案，但并不是唯一站得住脚的答案。ScaleRL(Khatri et al., 2025) 使用单个被截尾且截断梯度的权重 $\mathrm{sg}[\min(\rho_{i,t}, \epsilon)]$，其中 $\rho_{i,t} = \pi_\theta / \pi_{\text{inf}}$——正是刚刚警告过的那种合并形式(22.4)——而它行得通，因为一个被截尾、而不是被用来截断目标的权重，衰减的是一个词元的贡献而不是删除它，于是失配只留下逐词元学习率上有界的乘性噪声。所以在哪里花掉稳健性这个选择是真实存在的：要么分解这两个比值，要么合并它们并依赖衰减，但不要既合并又截断。

22.5 修正二：修好数值

第二族并不修正这个间隙；它移除这个间隙。截至 2026 年年中，只要负担得起，这就是更好的答案，而证据强度不同寻常，因为有三个团队各自独立地走到了这一步。

22.5.1 语言模型头

Khatri et al. (2025) 在一个 400,000 GPU 小时的消融项目上拟合了他们的 S 形标度律 (12.14)，并报告仅仅把语言模型头的精度提到 FP32，就把渐近奖励 A 从 0.52 提到了 0.61——这是他们整项研究中最大的单轴效应，比损失函数、优势归一化器、课程或长度控制都大，而且它是一处数值改动，不是算法改动。MiniMax (2025) 此前已经把失配定位到同一处，并在生成器与训练器两侧都在那里用了 FP32；Yao et al. (2025b) 独立地发现引擎的头精度与训练器的不一致；TMAX 终端智能体配方 (TMAX, 2026) 则称 FP32 头在长时域下「尤为关键」。四个团队，一个答案。

机制很直白。头是一个 $d \times |\mathcal{U}|$ 的矩阵乘，其中 $|\mathcal{U}| \approx 10^5$，后接一个分母要把 10^5 项加起来的 log-softmax：这是网络中最宽的归约，也是我们真正要求导的那个量之前的最后一次归约，所以它累积的舍入误差最多，而且后面没有任何层能把它平均掉。把它提精度也很便宜——每个词元一次矩阵乘加一次归约，对上几十层 Transformer。

22.5.2 浮点格式本身

Qi et al. (2025) 给出了这个论证最锋利的版本：根因不是算法、不是算子核、也不是引擎，而是数字格式。BF16 有 7 个显式尾数位，FP16 有 10 个，每个指数下可表示的数值大约少 8 倍；而 BF16 补偿性的优点——宽动态范围，从而免去损失缩放——是一项预训练上的便利，与 RL 毫无干系。他们的主张是，退回 FP16 只需几行代码就能在很大程度上消除失配，无需任何架构或算法上的改动，并且在各种任务、算法与框架上都能带来更稳定的优化与更强的最终性能。一个被反复转述的「24× 削减」的数字，可以追溯到一份二手摘要而不是论文本身，不应被当作实测值来引用。

历史层面的解读值得直白说出来。BF16 之所以成为大模型训练的默认，是因为它的指数范围让混合精度预训练无需损失缩放机制就能工作，而 RL 继承了这个默认却没有重新审视它。但 RL 的约束瓶颈并不是动态范围——一个收敛模型中的激活值是良好标度的——而是两份独立实现要在 10^{-2} 奈特之内保持一致，那是一个精度要求。一个预训练特有的选择被带进了取舍方向恰好相反的领域，代价是多年针对一个症状的算法研究。

22.5.3 批次不变的算子核与真正的同策略 RL

「移除间隙」的原则性版本，是让两个引擎逐比特相同，而He and Thinking Machines Lab (2025) 展示了那需要什么。由于逐元素运算本来就是批次不变的，真正要紧的只有三项：RMSNorm、矩阵乘与注意力。注意力是难的那个，因为归约顺序必须在预填充与解码之间、以及在缓存的键值与新算出的键值之间保持一致，这意味着要统一 KV 访问并固定 FlashDecode 的切分尺寸——用浪费掉的 FLOPs 换取确定性。做到这些之后，他们的 RLVR 运行显示，实测的训练—推理 KL 平平地停在恰好为零，训练稳定，且完全不需要任何修正项。LMSYS Org (2025) 把它带进了生产环境，让批次不变的注意力与分块预填充、CUDA 图和基数缓存兼容，并报告 CUDA 图支持把确定性开销降到了大约三分之一。

那份开销就是全部的交易，而表 22.3总结了由此得到的三选一、且互不排斥的选择 (Zhao et al., 2025b)：要么做到真正同策略并付出吞吐，要么在算法上修正并付出偏

表 22.3: 修正方法的全景。「偏差」指引入到策略梯度中的偏差；「作用处」指该干预在流水线上起作用的位置。前三行移除间隙，中间三行对它做修正，最后三行则重构目标函数，使它不那么要紧。只有第一组没有统计上的代价。

方法	它做什么	作用处	偏差	代价
FP32 LM 头	把最宽的那次归约提精度	两个引擎	无	可忽略
全程 FP16	更多尾数位吸收差异	两个引擎	无	无
批次不变算子核	跨批大小固定归约顺序	推理引擎	无	吞吐
词元级 TIS	每词元一个截断梯度的截尾权重	损失	温和、有界	一次前向
序列级 TIS	每条回复一个截尾权重	损失	若被截尾则很大	一次前向
掩蔽式 IS	丢掉分歧最大的词元	损失	覆盖率损失	一次前向
GSPO	序列级的几何平均似然	目标函数	粒度改变	更小的 ε
路由重放（R3）	训练时重放引擎的 top-k	前向	优化的是 π_θ^{R3}	显存、管线改造
IcePop	对词元级差异做掩蔽与截断	损失	覆盖率损失	需要调阈值

差，要么对齐精度并付出工程。对低精度技术栈，LMSYS Org (2026) 以一种值得背下来的形式陈述了这条支配原则：采样、训练、检查点转换与在线权重更新必须全都遵守 同一份精度契约——训练与推理两套算子核库之间一份比特精确的量化器契约，而不仅仅是两侧名义格式相同。

22.6 修正三：结构性回避

第三族改变目标函数，使那份分歧能腐蚀的东西更少。

使用序列级似然。 GSPO 的几何平均序列比值(12.6)在取指数之前先平均了 T 个词元对数比值，从而把单个词元的波动压低了 $1/T$，而Zheng et al. (2025a) 报告，由此得到的容忍度大到足以让直接使用引擎的似然变得可行。这个论证及其代价——尤其是截断范围要小三个数量级——见节 12.5。有两点要当心：平均降低的是失配的 方差而不是其系统性成分，所以 GSPO 对表 22.1中采样器契约那一行毫无帮助；而且Tang et al. (2026) 报告，在万亿尺度上 GSPO 维持了高熵，但带来的长度增长有限，所以它并不是一个免费的结构性修法。

把训练器的 logits 放进分子。 Tang et al. (2026) 走了现有最经济的一条路，用训练引擎的 logits 替换重要性比值的分子，于是完全不需要任何重算。配上一个训练—推理比值修正与样本级的损失归一化，这就属于他们所报告的、足以在 10^{12} 参数上稳定 zero RL 的那「几处简单却关键的改动」。这个选择有意思的地方在于，它把训练器当作权威，把引擎当作一个待修正的采样器——与通常的约定正好相反。

重放路由决策。 对 MoE，正确的结构性修法是让两次前向走同一条离散路径。采样路由重放（Rollout Routing Replay, R3）记录下推理引擎的 top-k 专家选择掩码，并在训练前向中重放它 (Ma et al., 2025)；关键在于只有选择被重放，路由器的 logits 及其归一化仍在实时重算，因此梯度依然会流进路由器。Ma et al. (2025) 报告 KL 大幅下降、结果优于 GSPO 与 TIS，且不牺牲训练速度，而该机制已经落地进 Megatron-Core、NeMo、SkyRL 与 Miles。有一条告诫很重要：Stabilizing RL (2025) 指出，重放会隐式地对目标策略引入偏置，因为词元似然此后由重放的而非自然的路由所支配，于是被优化的是 π_θ^{R3} 而不是 π_θ ——节 22.9将会把这同一笔交易一般化。

掩蔽掉分歧词元。 IcePop 出自 Ring-1T 报告 (Ling Team et al., 2025)，通过词元级的差异掩蔽与截断来稳定万亿尺度的训练——本质上就是带一道截断阶段的 MIS。它出现在一份前沿训练报告中这件事本身，就相当合理地代表了实践者认为它有多么承重。

22.7 温度与采样失配

现在我们转向一个诊断结果看起来一模一样、成因与解法却截然不同的问题。它不是一个数值 bug 而是一个规范 bug，它很常见，而且它的量级远远压过节 22.5里的一切。

采样轨迹几乎从来不是通过从模型的原始分布中采样得到的。它们是在某个温度 T 下生成的，通常还带核采样（top-p）截断 (Holtzman et al., 2020) 或 top-k 截断，有时还带重复惩罚；而对数概率通常是训练器从原始的、未加温度、未截断的 softmax(2.1) 算出来的。那是不同的分布，而且差别是系统性的。

22.7.1 加温后的分布并不是未加温分布的一个偏移

设 $z \in \mathbb{R}^{|\mathcal{V}|}$ 是某个位置上的 logits，记 $L(z) = \log \sum_v e^{z_v}$ 为 log-sum-exp，并设 $\mathcal{S}$ 为截断后保留的集合。采样器抽自

$$q(v) = \frac{\mathbb{1}[v \in \mathcal{S}]\, e^{z_v/\mathcal{T}}}{\sum_{v' \in \mathcal{S}} e^{z_{v'}/\mathcal{T}}}, \qquad \text{而训练器打的分是} \qquad p(v) = e^{z_v - L(z)}. \tag{22.7}$$

推导: 间隙是 logit 的仿射函数

暂且忽略截断（$\mathcal{S} = \mathcal{V}$），

$$\log q(v) - \log p(v) = \left(\tfrac{1}{\mathcal{T}} - 1\right) z_v - \left[L(z/\mathcal{T}) - L(z)\right]. \tag{22.8}$$

方括号里的项不依赖 v，所以这个间隙是该词元 *logit* 的一个仿射函数，斜率为 $1/\mathcal{T} - 1$——不是一个常数偏移,因而不会在比值中被抵消。在尖峰极限下 $L(z/\mathcal{T}) \approx z_{\max}/\mathcal{T}$、$L(z) \approx z_{\max}$，于是

$$\delta \;=\; \log q(v) - \log p(v) \;\approx\; \left(1 - \tfrac{1}{\mathcal{T}}\right)(z_{\max} - z_v) \;\geq\; 0 \quad \text{对} \mathcal{T} > 1. \tag{22.9}$$

在 $\mathcal{T} = 1.2$ 时，一个 logit 比最大值低 5 的词元会得到 $\delta \approx 0.83$ 奈特，而(22.3)在 $T = 32{,}000$ 时给出的预算约为 0.006 奈特：超出容忍度两个数量级，而且符号固定。相应的 TIS 权重 $e^{-\delta} < 1$ 会给 每一个被采样的词元降权，并且打击那些稀有词元最狠——正是节 12.5.2指认出的、你最不想压制的那些分叉词元。修正与缺陷合起伙来了。

22.7.2 截断让比值变得无定义，而不只是有偏

top-p 与 top-k 比温度更糟，因为它们改变了支撑集。记 $\eta_t = \sum_{v \notin \mathcal{S}_t} p(v)$ 为位置 t 上被丢弃的质量，被保留的词元满足 $q(v) = p(v)/(1 - \eta_t)$，于是

$$\delta_t = \log q(y_t) - \log p(y_t) = -\log(1 - \eta_t) \approx \eta_t, \tag{22.10}$$

一个等于被丢弃质量的系统性正间隙。在 $p_{\text{top}} = 0.95$ 时，η_t 在自信的位置上接近零，在平坦的位置上则达到整整 0.05，所以一个看似合理的核采样设置就贡献了量级为 10^{-2} 奈特的逐词元偏差——比一套配置良好的 BF16 技术栈的全部数值间隙还高一个数量级，而且在那份间隙至少部分是随机的地方，它是系统性的。在 $T = 2000$ 个词元、平均 $\eta = 0.02$ 时，序列对数比值约为 -40，没有任何截尾重要性权重能修补它。

有一点值得直说，因为这是本章中唯一一处没有任何修正能起作用的地方：恒等式 (2.15)要求行为策略在目标策略有支撑的地方都有支撑，而核采样给出的恰恰相反，$\mathrm{supp}(q) \subsetneq \mathrm{supp}(p)$。缺失的那块区域是不可观测的，所以偏差 $\sum_{v \notin S} p(v)\, g(v)$ 无法从数据中估计，更别说修正了。这是一个结构性缺陷而不是方差问题，这也是为什么接下来给出的是一条规则而不是一种偏好。

工程实践：采样器契约

从下面两条中恰好选一条，并把它落实在代码里，而不是落实在文档里。

1. **在 $\mathcal{T} = 1$ 下采样，不用 top-p 也不用 top-k**，这样 $q = p$，剩下的唯一间隙就是数值间隙。2025–2026 年大多数推理配方都是这么做的（节 11.7），而理由正是本节，不是传统。
2. **就按你采样时所用的那个分布来打分**，即训练器计算对数概率时施加同样的温度与截断掩码，这样被优化的策略就是 q。这是正当的，但有两个后果：你记录的熵以及到 π_{ref} 的 KL 都是加温后策略的，而且对 logits 的梯度会带上一个 $1/\mathcal{T}$ 的因子，悄悄地重标度了你在 logits 上的学习率。

无论选哪一条，都要把 $(\mathcal{T}, p_{\text{top}}, k_{\text{top}})$ 与每一次采样一并记录，并在启动时断言引擎返回的就是它采样时所用的对数概率。那个旗舰开源引擎直到为此被打了补丁之前都没有暴露它们 (Yao et al., 2025b)，这件事本身就说明这条断言值得写。

同样的逻辑延伸到重复惩罚、禁用词元表、受约束解码，以及投机解码——后者的接受规则定义了一个有效采样分布，而它未必就是目标模型的那个。截至 2026 年年中，对最后一项还没有任何仔细的研究：支持投机采样的吞吐论证很强，而正确性论证则无人检视。

代码 22.3: 按实际采样时所用的那个分布来打分。掩码必须由引擎所用的 同一批 logits 与同一个阈值构造出来，这意味着要复现它的采样器，而不是近似它。

```python
def sampler_consistent_logprobs(logits, tokens, temperature=1.0, top_p=1.0):
    """按式 (22.9) 的 q 计算对数概率，而不是按原始 softmax。"""
    z = logits.float() / temperature
    if top_p < 1.0:
        srt, idx = z.sort(dim=-1, descending=True)
        keep = (srt.softmax(-1).cumsum(-1) - srt.softmax(-1)) < top_p  # 核采样
        mask = torch.zeros_like(z, dtype=torch.bool).scatter_(-1, idx, keep)
        z = z.masked_fill(~mask, float("-inf"))
    logq = z.log_softmax(dim=-1)                       # 在 S 上重新归一化
    return logq.gather(-1, tokens.unsqueeze(-1)).squeeze(-1)
```

```

# 诊断：若按原始 softmax 打分，会引入多大的偏差？
def nucleus_bias(logits, temperature=1.0, top_p=0.95):
    z = logits.float() / temperature
    srt = z.sort(dim=-1, descending=True).values.softmax(-1)
    eta = 1.0 - srt.cumsum(-1).clamp(max=top_p)       # 被丢弃的质量，式 (22.10)
    return -(1.0 - eta).log()                         # 约等于每词元 eta 奈特
```

22.8 陈旧度：另一种离策略性

异步 RL 引入了第二个间隙，而它总被与第一个混为一谈。这个区分很简单，也很承重。

核心思想: 失配是数值间隙；陈旧度是权重间隙

失配是对同一组参数的两次评估之间的差别。陈旧度则是两个不同参数向量之间的差别：s 个优化步之前的 θ_{old} 与此刻的 θ。陈旧度会随你更频繁地同步而缩小，有良定的修正方法，而且是一个由你设定的设计参数；失配这三条一条都不占，因为它是你那套软件栈的性质。它们是相加的，而数值间隙是所测总量之下的一块地板，所以在你测出零陈旧度下的那个间隙之前，你没法靠盯着对数概率间隙来调一个陈旧度上界。试图这么做，是这个领域里最常见的误诊。

系统这一侧属于第21章；这里我们需要的是算法上的后果。

有界陈旧度加解耦修正。 AReaL(Fu et al., 2025a) 把可中断生成与节 22.4的解耦 PPO 分解结合起来——行为策略用于重要性权重，邻近策略用于信赖域——并报告在八步离策略性下训练稳定。PipelineRL(ServiceNow Research, 2025) 把粒度做得更细，在两次前向之间交换权重，从而让生成不被打断；ScaleRL 也在同样的八步下运行它 (Khatri et al., 2025)。每个正经的框架都把陈旧度上界暴露成一个显式旋钮，而把它设为零应该精确地退化回同步算法——这值得一测，因为它是唯一能端到端检验修正机器是否正确的手段。A-3PO (2025) 观察到，既然邻近策略只充当一个信赖域锚点，那它就可以用版本感知的插值来近似而不必重算，从而免去算法 22.2所付出的那次额外前向。

当「那个行为策略」不复存在时。 Magistral 的系统会在生成过程中把新权重广播给生成器，既不丢弃在途的序列，也不刷新 KV 缓存 (Mistral-AI et al., 2025)。因此一条序列可能前面的词元是在旧权重下、针对由旧权重建起的缓存算出来的，而后面的词元是在新权重下、针对同一份陈旧缓存算出来的，于是对它并不存在单一的 θ_{old}，$\pi_{\theta_{\text{old}}}$ 严格

来说是无定义的。实践中人们就在当前的权重下重算 $\log \pi_{\theta_{\text{old}}}$ 并接受那个间隙——这正是为什么这类框架需要一种能优雅衰减的修正，而不是一种假定行为策略良定的修正。2026 年有一条相关的工作线研究行为对数概率完全缺失的情形 (Missing Old Logits, 2026)。

比值的尺度取决于熵。 Zhao et al. (2026) 给出了这个领域 2026 年最有用的观察：重要性比值的自然尺度会随词元熵系统性地变化。在低熵位置上，一次正当的更新对被采样词元的对数概率改动极小，因此观测到的偏离主要来自被放大成采样噪声的训练—推理差异；而在高熵位置上，在途的权重更新会诱发大幅但正当的探索性偏离。因此一个只看幅度的阈值——本章中每一个截断与每一个 TIS 上限都是这样的阈值——会在两种情形下同时出错：在尺度小的地方放进被放大的噪声，在尺度大的地方掩蔽掉必需的探索。其机制是关于 softmax 扰动的两行计算，留给习题。ESTR 按每个词元的局部熵来标度它的偏离，既不需要辅助前向也不需要版本切换检测，并报告在 2.6× 的训练速度下达到与同步 GRPO 相当的准确率。把那个加速当作单篇论文的结果来对待；把那个机制当作很可能会留存下来的东西来对待。

干脆完全不要行为概率。 Liu et al. (2026b) 问的是，异步的组相对 RL 能否只用当前策略的概率来稳定，从而免去跨两套系统的、带版本且词元对齐的记账。他们的诊断是一种尺度失衡的失效模式：当陈旧回复在当前策略下被打分时，正项与负项的损失落在不同的负对数概率尺度上，于是一组和为零的组相对优势不再意味着损失贡献是平衡的——那套让组基线无偏的算术悄悄地不再成立。他们的修法是按每条回复自身的平均词元负对数似然来归一化它的词元损失。其吸引力在于，消除行为对数概率就等于消除了那个会失配的量，从而消除了失配；而这个代价——一个带有自身偏差的新归一化器——是否小于记账的代价，截至 2026 年年中尚无定论。

22.9 优化了错误的策略

到目前为止，一切都把 π_{inf} 当作对 $\pi_{\theta_{\text{old}}}$ 的一次被腐蚀的测量，并把训练器当作权威。Liang et al. (2026) 观察到，这把优先次序弄反了，而且反得足够彻底，以至于重新表述了整个领域。被部署的那个策略是推理策略：用户的词元是由 vLLM 或 SGLang 在某个批大小、某种量化和某种并行布局下采出来的，不是由训练器采出来的。本书中每

一个目标函数优化的都是通过 训练引擎策略定义的 $J(\theta)$，而人们真正在意的量却是

$$J^{\text{inf}}(\theta) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \mathop{\mathbb{E}}_{y\sim\pi_{\text{inf}}(\cdot\,|\,x;\theta)} [r(x,y)], \tag{22.11}$$

其中 $\pi_{\text{inf}}(\cdot;\theta)$ 表示引擎对权重 θ 的实现。那个映射不是 $\theta \mapsto \pi_\theta$，而且由于它牵涉离散决策——路由、算子核派发、量化桶分配——它在任何有用的意义上甚至都不是 θ 的连续函数，所以一次可证明地提升了 J 的更新，对 J^{inf} 没有任何保证。这不是「梯度有噪声」；它说的是，被上升的那个目标不是被评测的那个目标。

Liang et al. (2026) 提出把单调推理策略改进作为正确的目标，并提出 MIPU 作为一个两步框架：先构造以采样器为参照的候选更新，再用一个推理侧的间隙代理来有选择地接受已同步的候选。在结构上它是一个信赖域方法，只是其约束住在推理策略的空间里，从而把失配从一件要去修正的事，变成一件要去对着优化的事。

这次重新表述在 2026 年年中还很新，未被复现，只有一篇论文之深，但它是值得关注的方向：它解释了一个原本令人费解的现象，即那些在训练器侧指标上有提升、但一部署就没有提升的检查点，而对此惯常诉诸的评测噪声与模板漂移都解释不通。它还有一个不需要它任何机器的免费推论：在部署配置下做评测——同样的引擎、同样的量化、同样的批大小区间——因为在第三种配置下跑的评测测的是第三个策略。大多数已报告的 RL 结果都没有说明它们评测的是哪个策略。

22.10 一份清单

工程实践: 在你怪罪算法之前

按此顺序。相对于它所保护的那次运行，每一步都很便宜，而且每一步都会改变下一步的含义。

1. **在零陈旧度下测量间隙**(22.1)，用一个批次，用 算法 22.1，并把 s 与 $1.1/\sqrt{T}$ 相比——这里的 T 是你打算训练时的 T，不是你调试时的 T。
2. 在动精度之前先**审计采样器契约**。单一符号的间隙，或者中位数远大于离散程度，是温度或截断的 bug（节 22.7），再多 FP32 也帮不上忙。
3. **把语言模型头在两个引擎上都强制为 FP32**，然后重测：这是现有回报率最高的单项改动 (Khatri et al., 2025; MiniMax, 2025)。
4. **如果模型是稀疏的，就逐层测量路由器一致率**，低于九成多就启用路由重放 (Ma et al., 2025)，同时接受你此后优化的是 π_θ^{R3}(Stabilizing RL, 2025)。

表 22.4: 把这份清单写成一张症状表，供已经出问题的运行使用。右手一栏才是本章的要点：好几种熟悉的 RL 失效特征，都有一个比算法解释更便宜就能验证的数值解释。

症状	第一件要查的事	为什么
内循环第零步就有非零截断比例	比值的分母里是 π_{inf} 吗?	(22.4)：比值从 $e^{-\delta}$ 起步，不是 1
所测的到 π_{ref} 的 KL 有非零地板	诊断量是用引擎的对数概率算的吗?	那个地板是 $\Theta(s^2T)$，不是漂移
加长之后奖励坍塌	s 与 $1.1/\sqrt{T}$ 相比	容忍度按 $T^{-1/2}$ 缩小(22.3)
间隙单一符号，中位数 $\gg$ 离散程度	温度与 top-p	(22.9)、(22.10)：这是偏差，不是离散程度
只有 MoE 模型才坍塌	逐层的路由器一致率	离散翻转是重尾的 (Ma et al., 2025)
升级引擎之后出现不稳定	注意力后端与切分尺寸	归约顺序变了 (He and Thinking Machines Lab, 2025)
启用 FP8 采样之后出现不稳定	统一的精度契约	每个阶段都必须一致 (Jet-RL, 2026; LMSYS Org, 2026)
序列级 TIS 的运行差于词元级 TIS	序列截尾率	长度陷阱
上线的模型差于评测用的检查点	你评测的是哪个策略?	(22.11)(Liang et al., 2026)

5. **对齐一切你能对齐的布局**：注意力后端、分块预填充、并行宽度，以及每一次权重传输与重量化步骤的精度 (LMSYS Org, 2026)。
6. **然后才从表 22.3中选一种方案**——真正同策略、算法上修正，或者精度对齐——因为这个选择取决于第 1 步产出的那个数。
7. **到这时才引入陈旧度**，一次一步，并验证陈旧度为零的配置精确地复现了同步算法。
8. **在服务配置下做评测** (Liang et al., 2026)。
9. **到这时才改损失函数**。带着一个未修正的间隙走到这一行，你跑的每一次消融都部分地是在消融 P7。

本章小结

- 一套解耦式 RL 技术栈包含四个策略，不是三个。推理引擎的 π_{inf} 与训练器的 $\pi_{\theta_{\text{old}}}$ 是同一组比特级相同权重上的不同函数，所以「同策略」RL 悄悄是离策略的，而原因与强化学习毫无关系。
- 这份分歧有十个可辨认的来源（表 22.1），其中三个在定性上与小噪声截然不同：混合专家的路由翻转是离散的、且在序列内部相关；布局与后端的效应在一次运行中是系统性的；采样器契约的错误则是符号固定的偏差。
- 诊断量就是逐词元的对数概率间隙(22.1)。要报告它的分布以及由此推出的有效样本量，绝不要只报均值：在对数正态模型下 $\text{ESS}/n = e^{-Ts^2}$ 取决于离散程度而完全不取决于均值，所以小均值配重尾严格比大均值更危险。同一个恒等式还给出一个可用的预算 $s \lesssim 1.1/\sqrt{T}$，而它按 $T^{-1/2}$ 的缩小正是为什么失配在对话长度的 RLHF 中可以忽略，到了长思维链的长度上却成了一阶的工程问题。
- 一个未修正的间隙在两种权重处理方式下的失效方式不同。截断改变的是哪些词元有贡献，于是信赖域退化成一个筛选由系统决定的数值不稳定性的过滤器。截尾权重改变的是每个词元贡献多少，这更难被发现，因为不会出现任何截断比例上的异常。
- 失配不只是一个方差问题。Zhong et al. (2026) 在一个零失配的诊断环境中把它孤立出来，并报告微小的词元级数值分歧能够独立地引发坍塌，进而主张失配改变了实际的优化问题。
- 截尾重要性采样用一个截断梯度的权重 $\min(\pi_{\theta_{\text{old}}}/\pi_{\text{inf}}, C)$ 乘性地施加在 PPO 损失上来修正它。纪律在于分解：一个比值修补更新之前的测量误差，另一个执行当下的信赖域，而把它们合并是标准的 bug。序列级 TIS 会掉进长度陷阱，因为 10^4 个词元上的乘积几乎总会被截尾，而那个「修正」也就退化成一个常数。
- 修好数值胜过为数值做修正。语言模型头上的 FP32 把 ScaleRL 的渐近奖励从 0.52 提到了 0.61，是一项 400,000 GPU 小时研究中最大的单轴效应；MiniMax-M1 早已把问题定位到同一处；而Qi et al. (2025) 主张根因是 BF16 的尾数宽度，并且 FP16 只需几行代码就能移除失配。三个团队各自独立地汇合到「这是一个数值 bug，不是一个 RL bug」。
- 温度与 top-p 失配是一个独立的、更大的、且系统性的缺陷。间隙是词元 logit 的仿射函数，斜率为 $1/\mathcal{T} - 1$；而核采样截断让采样器的支撑集成为策略支撑集的

真子集，于是没有任何重要性权重能修补它。要么在 $\mathcal{T} = 1$ 下不加截断地采样，要么就按你采样时所用的那个分布来打分。

- 陈旧度是权重间隙，失配是数值间隙；它们相加，而数值间隙是对总量的任何测量之下的一块地板。重要性比值的自然尺度随词元熵增长，所以只看幅度的阈值在低熵与高熵两个区间上都会失准 (Zhao et al., 2026)。
- 最后，这一切优化的都是训练引擎策略，而被部署的却是推理策略 (Liang et al., 2026)。它免费带来的推论是要在部署配置下做评测；它所打开的研究议程则基本上尚未被探索。

延伸阅读

先读Yao et al. (2025b)，读博客版而不是研讨会版，因为博客记录了调试的路径：发现引擎并没有暴露它采样时所用的概率，头精度不一致，以及在这两处修好之后仍然残留的算子核层面的分歧。按事情发生的顺序去读一份系统调查，是现有关于如何着手这类问题的最好教材，而且它很短。然后读He and Thinking Machines Lab (2025) 了解底层基础：把元凶认定为批次不变性而不是原子操作，是这个领域里最澄清问题的技术结果，而文末那个「真正同策略的 RL」实验——KL 平平地停在恰好为零，没有任何修正项——则是间隙可被移除而非内禀的存在性证明。LMSYS Org (2025) 是它的生产续篇，值得为其中的成本核算略读一遍。

关于数值对上算法的论证，接下来读Qi et al. (2025)：篇幅短，主张可证伪，而且它那个历史观察——BF16 是出于一个预训练特有的理由被采用的，却被继承进了一个取舍方向相反的领域——其适用范围远超本章。把它与Khatri et al. (2025) 里的 FP32 头结果配对来读，那是同一个发现被独立地、并且带着一套标度律方法学得到的；再配上MiniMax (2025)，它最早到达那里，声势却最小。

2026 年的工作里有三篇要紧，按此顺序读。Zhong et al. (2026) 通过构建一个可以孤立地引入数值分歧的环境，让这个现象在科学上变得可处理，而它的补救手段分类是本章内容最完整的一份综述。Zhao et al. (2026) 是这里今年最锋利的算法思路，而它的核心观察——重要性比值的自然尺度随词元熵而变，所以幅度阈值是错误的工具——应当改变你阅读 第12章中每一个截断参数的方式。Liang et al. (2026) 是最后读、也是要想得最久的一篇：如果推理策略才真的是目标函数，那么本书中相当大一部分结果，报告的都是一个没有人会部署的策略。

关于 MoE，请带着Stabilizing RL (2025) 中的告诫去读Ma et al. (2025)，前者解释了后者的代价是什么。Fu et al. (2025a) 是带解耦修正的异步训练的参考文献，而 Mistral-AI et al. (2025) 值得一读，因为它坦诚地讲了生成过程中的权重广播，那是印刷品中关于「行为策略」并非一个良定对象的最清楚的例证。

习题

22.1（**有效样本量预算**。）对 $\log W \sim \mathcal{N}(-Tm, Ts^2)$ 计算 $\mathbb{E}[W]$ 与 $\mathbb{E}[W^2]$ 以验证(22.2)，并确认均值被消掉。对 $s \in \{10^{-3}, 10^{-2}, 3\times10^{-2}\}$ 与 $T \in \{10^2, 10^3, 10^4, 3.2\times10^4\}$ 列出 ESS/n 的表格，标出违反(22.3)的那些格子。然后在词元间隙之间带两两相关系数 ϱ 重做该推导，并说明 s 上的预算如何改变，与节 22.3.3相对照。

22.2（**内循环第零步的虚假截断**。）某套技术栈把引擎的对数概率用作比值的分母，于是在 $\theta = \theta_{\text{old}}$ 处 $\rho_{i,t} = e^{-\delta_{i,t}}$。在 $\delta \sim \mathcal{N}(0, s^2)$ 与 PPO 边界 $\varepsilon_{\text{low}} = 0.2$、$\varepsilon_{\text{high}} = 0.28$ 下，计算已经落在截断带之外的词元占比，作为 s 的函数。求出在任何更新之前就有 1% 词元被截断的那个 s。然后对图 22.2中勾勒的混合密度（97% 在 $s = 0.01$、3% 在 $s = 0.06$）重做一遍，并评论答案依赖于该分布的哪一个矩。

22.3（**长度陷阱，以及把 TIS 看作优势重加权**。）(a) 在节 22.2的模型下，对 $s = 0.01$ 与 $T \in \{10^3, 10^4\}$，计算序列级 TIS 权重在 $C_{\text{seq}} \in \{2, 10\}$ 处被截尾的概率。(b) 证明在被截尾的序列上，该估计量正好是对损失的一次常数重标度，因而它的偏差不会随序列数增加而消失。(c) 证明词元级 TIS (22.6)等价于把 $A_{i,t}$ 替换成 $w_{i,t}A_{i,t}$，并对 $A > 0$ 与 $A < 0$ 分别确定截尾偏差的符号。它压制的是哪些词元，那是不是你想压制的那个集合?

22.4（**温度与核采样偏差，定量地**。）(a) 推导(22.8)以及尖峰极限的近似(22.9)。(b) 对一个前五个 logits 为 $(12.0, 11.0, 9.5, 9.0, 8.0)$、其余 logits 全都低于 4 的位置，在 $\mathcal{T} \in \{0.8, 1.2\}$ 下精确计算这五个词元各自的 δ。(c) 对同一个位置计算 $p_{\text{top}} = 0.95$ 时核采样被丢弃的质量 η，进而算出偏差 (22.10)。(d) 把上述所有结果与 $T = 8000$ 时的预算 $1.1/\sqrt{T}$ 相比较，并说明在温度、核采样、BF16 数值这三种效应中你会先修哪一个。

22.5（**MoE 路由的算术**。）Ma et al. (2025) 报告每次前向大约有 10% 的路由器发生分歧，且 94% 的词元至少在一层上有差异。把逐词元、逐层的路由器分歧建模为在 L 层上以概率 ς 独立发生。(a) 写出至少在一层上有差异的词元占比，并在 $L = 48$

与所报告的 94% 之下解出 ς。(b) 解释为什么你算出的 ς 小于 10%，以及那两个被报告的数字实际上各自在数什么。(c) 从你的模型出发论证为什么一个序列级似然对这种失效比词元级的更稳健，并指出你论证中被路由传播所违反的那条假设。

22.6 ⋆ **（把推理目标形式化。）**认真对待(22.11)。(a) 构造一个最小例子——两个词元、一个提示、一个有两个桶的量化网格——使得一次梯度步严格提升 $\mathcal{J}(\theta)$ 而严格降低 $\mathcal{J}^{\text{inf}}(\theta)$。(b) 由于在那里 $\theta \mapsto \pi_{\text{inf}}$ 是分段常数，标准的梯度保证并不适用：请提出一个可以从一个推理侧采样批次计算出来的接受检验，它仅当 $\mathcal{J}^{\text{inf}}$ 的一个无偏估计没有下降时才接受更新，并用奖励方差与你希望检测的效应量来表述它的样本复杂度。(c) 把它与 Liang et al. (2026) 的「推理侧间隙代理」相比较。

22.7 ⋆ **（从第一性原理得到按熵标度的阈值。）**对一个 softmax 策略，证明 logit 扰动 Δz 诱发 $\Delta \log \pi_\theta(v) = \Delta z_v - \mathbb{E}_{u\sim\pi_\theta}[\Delta z_u]$，并进而证明：当 Δz 的各坐标 i.i.d. 且方差为 ς^2 时，$\Delta \log \pi_\theta(v)$ 在被采样的 v 上的方差是 $\sum_u \pi_\theta(u)^2$ 的一个函数。把该量与碰撞熵联系起来，再联系到 $\mathbb{H}[\pi_\theta]$。据此断定正当的对数比值幅度随熵而标度，而(22.1)中的数值成分不随熵标度，并设计一个阈值方案 $C(\mathbb{H}[\pi_\theta])$，使「该词元在数值上被腐蚀了」这一判断的假阳性率在各熵水平上被拉平。把你的方案与 ESTR 的 (Zhao et al., 2026) 相比较，并指出它们在何处不一致。

22.8 † **（在你自己的技术栈上测量间隙。）**取任意一个开源 RL 框架、一个 1–8B 的稠密模型，以及一套数学提示集。(a) 给流水线加上仪表，逐词元输出成对的 $(\log \pi_{\text{inf}}, \log \pi_{\theta_{\text{old}}})$，并用真实数据复现图 22.2。在生成长度 512、4096 与 16,384 下报告中位数、尾部、s 与 ESS$/n$，并检验安全预算的 $T^{-1/2}$ 标度。(b) 在两侧都把 LM 头强制为 FP32 并重做，报告 s 的变化。(c) 启用 FP8 或 INT8 采样并重做。(d) 引入一个刻意的温度失配（在 1.2 下采样、在 1.0 下打分），并证明由此得到的 δ 分布可以凭其中位数与离散程度之比，与任何一种数值配置区分开来，正如算法 22.1所假定的那样。

22.9 † **（等算力下的各种修正。）**用第12章的实验框架，在等算力下跑五种配置：(i) 不做修正，比值分母用引擎的对数概率；(ii) 词元级 TIS，$C = 2$；(iii) 序列级 TIS，$C_{\text{seq}} = 2$；(iv) 掩蔽式 IS；(v) FP32 头且完全不做修正。对每一种拟合(12.14)，报告带种子方差的 (A, B)，并附上 节 22.2的诊断面板。说明你是否复现了本章视为证据最充分的两条主张——修好数值胜过为数值做修正，以及序列级 TIS 在长生成长度上会退化——如果没有，指出你的技术栈违反了节 22.2的哪一条假设。

算法 22.1 失配诊断。在任何训练之前跑一次，在对引擎、布局、精度或采样器配置做任何改动之后再跑一次，并作为一项被记录的指标持续跑。它用一个采样批次回答「我的问题是不是 P7?」。

输入： 权重 θ、推理引擎、训练引擎、诊断用提示 $\mathcal{D}_{\text{diag}}$、采样器配置 $(\mathcal{T}, p_{\text{top}}, k_{\text{top}})$

第 1 步：确立采样器契约

断言引擎返回的是采样时实际使用的对数概率，包含温度与截断 ▷ 节 22.7

把 $(\mathcal{T}, p_{\text{top}}, k_{\text{top}})$ 与每一次采样一并记录

第 2 步：收集成对的对数概率

推送 θ；采样出 $\{y^{(i)}\}$，并带上逐词元的 $\log \pi_{\text{inf}}$

用训练器在同一 θ 下、在 $y^{(i)}$ 上做教师强制，重算 $\log \pi_{\theta_{\text{old}}}$

$\delta_{i,t} \leftarrow \log \pi_{\text{inf}}(y_{i,t}) - \log \pi_{\theta_{\text{old}}}(y_{i,t})$

第 3 步：汇总

$q_1, q_{50}, q_{99} \leftarrow \{\delta_{i,t}\}$ 的百分位数；$s \leftarrow \text{std}(\{\delta_{i,t}\})$

$W_i \leftarrow \exp(-\sum_t \delta_{i,t})$；$\text{ESS}/n \leftarrow (\sum_i W_i)^2 / (n \sum_i W_i^2)$

第 4 步：按此顺序逐级排查

if $q_{50} \gg s$ **或** δ 单一符号 **then**

怀疑采样器契约，而不是数值 ▷ 这是偏差，不是离散程度

else if $s > 1.1/\sqrt{T}$ **then**

把 LM 头强制为 FP32 并重跑；然后对齐精度（FP16，或一份统一的 FP8 契约）

end if

if 模型是 MoE **且** $|\delta|$ 的尾部很重 **then**

启用路由重放并重跑 ▷ 节 22.6

end if

return $(q_1, q_{50}, q_{99}, s, \text{ESS}/n)$，然后才去选择算法

算法 22.2 一次 TIS 修正的组相对更新迭代。与算法 11.1的唯一差别是失配修正之下的那几行，以及上升步中被截断梯度的因子 $w_{i,t}$。重算 $\log \pi_{\theta_{\text{old}}}$ 就是实践者总想省掉的那次前向；省掉它就会逼出合并后的比值(22.4)。

输入： 策略 π_θ、验证器 r_{ver}、提示 $\mathcal{D}$、组大小 G、截断 $(\varepsilon_{\text{low}}, \varepsilon_{\text{high}})$、TIS 上限 C

for 迭代 $k = 1, 2, \ldots$ **do**
  $\theta_{\text{old}} \leftarrow \theta$；把权重推送给推理引擎
  采样 $\{y^{(i)}\}_{i=1}^{G} \sim \pi_{\text{inf}}(\cdot \mid x)$，并**保留**逐词元的 $\log \pi_{\text{inf}}(y_{i,t})$ ▷ 采样时刻的概率
  失配修正
  用训练器在 θ_{old} 处重算 $\log \pi_{\theta_{\text{old}}}(y_{i,t})$ ▷ 那次额外的前向
  $\delta_{i,t} \leftarrow \log \pi_{\text{inf}}(y_{i,t}) - \log \pi_{\theta_{\text{old}}}(y_{i,t})$；记录算法 22.1的汇总量
  $w_{i,t} \leftarrow \min(e^{-\delta_{i,t}}, C)$，截断梯度
  寻常的组相对机器
  $r_i \leftarrow r_{\text{ver}}(x, y^{(i)})$；丢弃零方差的组；$A_i \leftarrow r_i - \bar{r}$
  for 内循环轮次 $e = 1$ **到** E **do**
    $\rho_{i,t} \leftarrow \exp(\log \pi_\theta(y_{i,t}) - \log \pi_{\theta_{\text{old}}}(y_{i,t}))$ ▷ 分母取自训练器一侧
    上升 $\sum_{i,t} w_{i,t} \cdot \min(\rho_{i,t} A_i, \text{clip}\big(\rho_{i,t}, 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\big) A_i) / \mathcal{N}$
  end for
end for
return π_θ

第23章

效率与成本控制

核心思想：强化学习后训练是按生成的词元计价的，而其中大多数词元被浪费了

预训练的成本随参数量乘以词元数而增长，而且每一个词元都对梯度有贡献。强化学习把这两条性质都反了过来。它的主导项是生成，而生成随采样轨迹的条数而非随模型规模增长，并且只跑在加速器算术吞吐的一小部分上，因为解码受内存带宽约束。更糟的是，在二值奖励与组相对优势之下，那些词元中有很大一部分——在现实的提示池里往往在四分之一到一半之间——贡献的梯度恰好为零。因此在这一设定下，效率首先关乎不去生成没用的词元，其次才关乎让每个词元更便宜。

节 1.6确立了问题的形状：后训练在总训练预算中的占比在 2022 年到 2026 年之间上升了两个数量级，而决定墙钟时间的是采样生成，不是梯度计算。本章把它变成一个你可以拿来计算的成本模型，然后按杠杆大小依次走过各种干预手段。它以经济学家的视角来看待前两章按机制处理过的材料，从而收束本书这一部分：第21章问的是怎样才能让一个分布式 RL 系统跑起来，第22章问的是低精度生成在什么时候是正确的，而本章问的是这一切按每一块钱来算究竟值多少。

23.1 钱花到哪里去了

23.1.1 成本的单位是一个生成的词元，而它不是按 FLOPs 计价的

固定一个优化步：B 个提示，每个提示 G 条采样回复，平均回复长度 $\bar{L}$。这一步会生成

$$M = BG\bar{L} \tag{23.1}$$

个新词元。这一步中其余的一切都是按 M 中的每个词元来计价的。节 1.6的 FLOP 核算给出：采样前向每生成一个词元约 $2N$ FLOPs，训练的前向加反向约 $6N$，而每多一

个需要重算对数概率的策略再加 $2N$（KL 项所需的参考策略，以及在引擎不返回自己的概率时、重要性比值所需的行为策略）。按这个核算，采样只占算术量的少数——而这个核算仍然是错误的预算方式，因为采样并不以与反向传播相同的效率运行。

自回归解码受带宽约束。为一条序列吐出一个词元，引擎必须读遍它用到的每一个参数；当有 b 条序列并发解码时，对权重的一遍读取服务了 b 个词元。记 BW 为聚合内存带宽，N 为参数量、每个参数 w 字节，κ 为每条序列每个词元的键值缓存字节数，则一个副本每秒能吐出的词元数上界为

$$\nu \;\le\; \frac{b\,\mathrm{BW}}{\underbrace{wN}_{\text{权重}} \;+\; \underbrace{b\,\kappa\,L_{\mathrm{ctx}}}_{\text{KV 缓存}}}, \qquad \text{受约束于} \qquad b\,\kappa\,L_{\mathrm{ctx}} \;+\; wN \;\le\; \mathrm{HBM}. \tag{23.2}$$

关于(23.2)有两点对本章其余部分要紧。第一，吞吐随 b 递增，所以 并发才是那个资源——任何能让更多序列在途的手段，都直接买来了生成吞吐。第二，内存约束意味着 b 被 $(\mathrm{HBM} - wN)/(\kappa L_{\mathrm{ctx}})$ 限住，所以长上下文在效果上以平方的方式挤占并发：它们既消耗了本可以容纳更多序列的显存，又放大了每一步的 KV 读取量。这正是为什么长思维链训练改变的是这个领域的经济学，而不仅仅是它的词元计数。

例 23.1 (一张 H100 上的 7B 策略)**.** 取 bfloat16 下的 $N = 7\times10^{9}$（$w = 2$），32 层、8 个维度为 128 的分组查询键值头，于是 $\kappa = 2\times32\times8\times128\times2 = 131{,}072$ 字节，i.e.*,* 每词元 128 KiB。在 $L_{\mathrm{ctx}} = 8192$ 的工作上下文下，每条序列约占 1.07 GB 缓存。一张 80 GB 卡上有 72 GB 可用、权重占 14 GB，于是 $b \le 54$。以 $\mathrm{BW} = 3.35$ TB/s 代入(23.2)，得到每 GPU 每秒 $\nu \le 2500$ 个词元。真实引擎只能达到其中的某个比例 η，因为注意力算子核、调度间隙与参差不齐的批次都要付出代价；下面的计算取 $\eta = 0.7$，即每 GPU 每秒约 1760 个词元。这大约是该卡 bfloat16 算术峰值的 2.5×10^{-2}。采样在 *FLOPs* 上很便宜，在时间上很昂贵。

23.1.2 两次运行的成本核算

表 23.1把定理 23.1贯穿到两次完整的运行中。两者都是从既定假设出发的算术演算，不是实测；要点在于这份拆解的形状——它在各种合理假设下都是稳定的——而不在于第三位有效数字。

表 23.1有三个特征值得评说。

表 23.1: 两次代表性运行的成本，由(23.1)与(23.2) 算出，取解码效率 $\eta = 0.7$、训练前向反向的模型 FLOPs 利用率为 40%、H100 级硬件，以及每步两次额外前向（参考策略与行为策略）。这些是带显式假设的推导数字，不是基准测试结果；它们落在节 1.6引用的数量级区间之内。值得记住的不变量是最后一块：在两个规模上生成都约占账单的四分之三。

	7B 数学 RLVR	**70B 智能体式 RL**
参数量 N ／采样副本	7×10^9，1 张 GPU	7×10^{10}，8 张 GPU
工作上下文 L_{ctx}	8,192	32,768
每词元 KV 字节 κ	128 KiB	320 KiB
并发序列数 b	54	40
实际解码 ν（词元/秒/GPU）	1,760	165
$B \times G \times \bar{L}$	$256 \times 16 \times 4{,}096$	$128 \times 8 \times 24{,}000$
每步生成的词元数	1.7×10^7	2.5×10^7
优化步数	1,500	800
生成的词元总数	2.5×10^{10}	2.0×10^{10}
验证器／环境调用次数	6.1×10^6	8.2×10^5
生成（GPU 小时）	3,970 （76%）	33,200 （76%）
前向 + 反向（GPU 小时）	740 （14%）	5,790 （13%）
额外的对数概率前向（GPU 小时）	490 （9%）	3,860 （9%）
观测预填充（GPU 小时）	—	970 （2%）
合计	$\mathbf{5.2 \times 10^3}$	$\mathbf{4.4 \times 10^4}$

这个占比几乎不随规模移动。 两次运行中生成都是 76%，而这并非巧合：生成成本按 MwN/BW 走，训练成本按 $6NM/(\text{peak} \cdot \text{MFU})$ 走，两者都对 NM 线性，因而它们的比值是硬件的性质——算术峰值除以内存带宽，在 2025–2026 年的加速器上大约是每字节 300 FLOPs——而不是模型的性质。这个比值十年来一直在变差，而不是变好。

验证在钱上便宜，在延迟上昂贵。 六百万次调用一个符号数学核查器、每次 50 毫秒，约合 85 CPU 小时，对上 5,200 GPU 小时不过是舍入误差。但如果那些调用坐在生成与优化步之间的关键路径上，它们就会给每一步都加上延迟，而验证器运行时间分布的尾部——一个超时的沙箱、一个挂死的单元测试——其行为与节 21.7中的掉队者问题一模一样。请按墙钟时间而不是按钱来给验证做预算。

智能体式的那次运行有一个隐藏项。 9.8×10^9 个观测词元必须被预填充。预填充受算力约束，因而每词元很便宜，所以它只占账单的 2%——前提是上下文能跨轮次被缓存。没有前缀缓存的话，第 k 轮会把第 $k-1$ 轮的整个前缀重新预填充一遍，而总预填充量按 $\sum_{k \le K} k\bar{L}_{\text{turn}} = O(K^2)$ 增长。对 $K = 20$，那是预填充词元的 10.5× 增长，会把一个 2% 的条目变成整次运行的五分之一。节 23.3接着讲这个。

> **核心思想: 成本随采样轨迹条数而增长，不随参数量而增长**
> 把 G 翻倍，账单翻倍。把 $\bar{L}$ 翻倍，账单不止翻倍，因为更长的上下文还会降低(23.2)所奖励的并发度 b。把 N 翻倍，账单大致翻倍。这三者并不是对称的杠杆，而实践者随手就设的恰恰是那些采样旋钮——$G = 16$，因为某篇论文用了 16——反倒是参数量才是他们反复纠结的那个。

23.1.3 对梯度有贡献的词元占比

现在来看给本章立论的那个分解。M 并不是每一份都买到了东西。

在组相对估计量之下，回复 i 对提示 x 的优势是该组奖励的一个中心化函数，所以如果 G 个奖励全都相等，组内每一个优势都是零，该组中每条回复的每个词元收到的梯度都是零。节 12.4把这当作一个统计问题——有效批大小缩水而名义批大小不变。读作一个预算问题，它就更刺眼：那些词元是按(23.2)的全价买来的，而什么都没换回来。

定义对梯度有贡献的词元占比

$$\phi \;=\; \frac{\text{属于奖励方差非零的组的生成词元数}}{\text{生成词元数}}. \tag{23.3}$$

在一个二值验证器与当前策略下逐提示成功概率 μ 之下，节 12.4.1 给出逐提示的退化概率 $\mu^G + (1-\mu)^G$（(12.4)）。在一个解出率按 F 分布的提示池上做聚合，并记 $\ell(\mu)$ 为难度为 μ 的提示上一个组所消耗的期望词元数，则被浪费的占比恰好是

$$1 - \phi \;=\; \frac{\mathbb{E}_{\mu \sim F}\left[\left(\mu^G + (1-\mu)^G\right)\ell(\mu)\right]}{\mathbb{E}_{\mu \sim F}\left[\ell(\mu)\right]}. \tag{23.4}$$

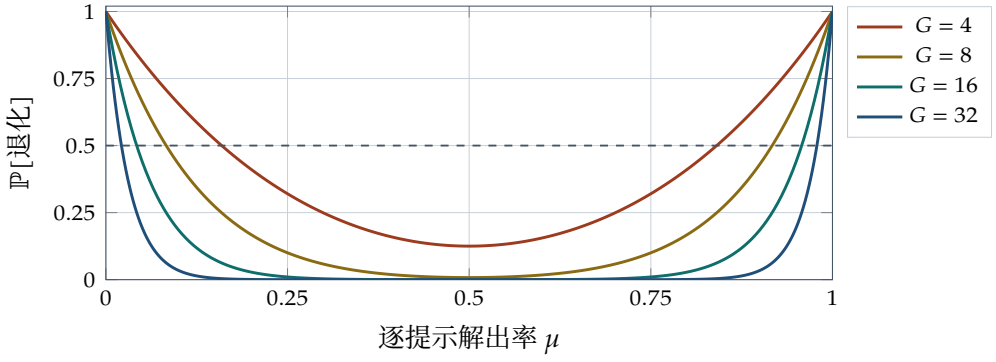


图 23.1: 买不到任何梯度的那部分生成，作为解出率的精确函数。每条曲线都是 $\mu^G + (1-\mu)^G$，即 G 次条件独立的 Bernoulli(μ) 抽样全体一致、因而整组的组相对优势全部消失的概率。这里没有任何拟合或实测的成分；它就是二项分布。虚线标出 50%。请注意这个论证的形状：曲线在很宽的中间带内接近零，而在两端急剧上升，所以一次运行中被浪费的占比几乎完全由提示池有多少落在尾部所决定——这正是为什么难度匹配（节 12.4.1）是一项成本干预而不只是一项统计干预，也正是为什么提高 G 是一种昂贵的、用来修补池子问题的办法。

推导: Beta 分布提示池下的闭式解

暂且取 ℓ 为常数，并设 $F = \mathrm{Beta}(a,b)$。那么 $\mathbb{E}[\mu^G] = \mathrm{B}(a{+}G,\, b)/\mathrm{B}(a,b) = \prod_{j=0}^{G-1} \frac{a+j}{a+b+j}$，由对称性把 a 与 b 互换即得另一项，于是

$$1 - \phi \;=\; \prod_{j=0}^{G-1} \frac{a+j}{a+b+j} \;+\; \prod_{j=0}^{G-1} \frac{b+j}{a+b+j}. \tag{23.5}$$

对一个均匀池，$a = b = 1$，这坍缩成 $2/(G+1)$：在 $G = 8$ 时是 22%，在 $G = 16$ 时是 12%。真实的提示池并不均匀；它们是 U 形的，包含许多策略每次都能解出的提示，也包含许多它从来解不出的。在 $a = b = 0.3$ 时，(23.5)给出 $G = 8$ 下浪费 53%、$G = 16$ 下浪费 43%。一个偏难的池 $\mathrm{Beta}(0.5, 1.5)$ 在 $G = 8$ 下给出 39%，其中几乎全部来自全错的组。

把长度依赖恢复回来会让情况更糟，而且是不对称地更糟。失败的尝试比成功的更长——一个解不出题目的策略会絮絮叨叨——所以 ℓ 随 μ 递减，而(23.4)会给全错的那一端加权。取线性的 $\ell(\mu) = \bar{L}(1.5 - \mu)$，$\mathrm{Beta}(0.5, 1.5)$ 的数字就从 39% 升到 44%。对一个关于 $\mu = 1/2$ 对称的池，两端的修正精确抵消，长度加权毫无影响，这既是对代数的一个不错的校验，也提醒我们对称情形才是那个良性的情形。

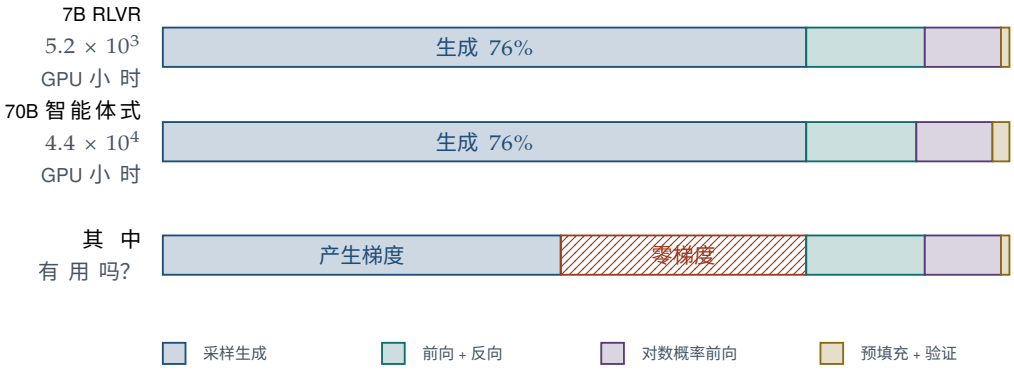


图 23.2: 表 23.1那两次运行的成本分解（各段长度就是该表中的百分比，因而继承了它的假设）。第三根条形把第一根的生成段按 $G = 16$ 下 $\mathrm{Beta}(0.3, 0.3)$ 提示池的(23.4)重新切分：阴影区域是产生了恰好为零的梯度的那部分生成。把三根条形放在一起读，就给出了本章的优先顺序——一份 RL 预算中最大的、可着手处理的单一量，既不是训练器，也不是验证器，更不是算术，而是那个阴影块。

图 23.1显示了为什么这个聚合量对提示池如此敏感。在 $G = 8$ 时，逐提示的退化曲线在 $\mu \in [0.25, 0.75]$ 上平坦且接近零，而在此区间之外几乎垂直上升。因此一个 30% 已饱和、30% 不可能的池，无论目标函数调得多仔细，都会浪费掉大约一半的预算；而一个集中在中间带的池只浪费百分之几。而且池子不会待着不动：随着策略变强，提示会从不可能的一端迁移到中间，再迁移到饱和的一端，所以即便数据集固定，ϕ 也会在一次运行中衰减 (Yu et al., 2025c)。

由于 ϕ 可以从每个训练器本来就有的量免费算出来，没有理由不记录它。代码 23.1 就是这么做的，并且把两种失效模式分开，因为全对的组与全错的组需要相反的补救：把前者从池中退役，而对后者要么丢弃、要么加脚手架。

代码 23.1: 为一个优化步计算对梯度有贡献的词元占比。请把 `token_fraction` 与奖励并排记录；它是一次 RL 运行中信息量最大的单个效率数字，而且它是免费的。

```python
def contributing_fraction(rewards, lengths, tol=1e-8):
    """rewards、lengths：一个步上形状为 (B, G) 的浮点张量。"""
    r, L = rewards.float(), lengths.float()
    informative = r.std(dim=1, unbiased=False) > tol          # (B,)
    tokens, live = L.sum(), L[informative].sum()
    return {
        "prompt_fraction": informative.float().mean().item(),
        "token_fraction": (live / tokens).item(),              # 即 phi，记录这个
        "wasted_tokens": (tokens - live).item(),
```

```
        "all_correct": (r.amin(dim=1) > 0.5).float().mean().item(),
        "all_wrong": (r.amax(dim=1) < 0.5).float().mean().item(),
        # 浪费在全体失败上的词元，而它们也是最长的
        "wasted_on_failures": L[r.amax(dim=1) < 0.5].sum().item(),
    }
```

工程实践：在优化任何东西之前，先测出你自己的拆解

三项测量，没有一项需要等一次训练跑完。(*i*) 分别在关闭生成（喂入缓存好的采样轨迹）与开启生成的情况下给一个优化步计时；差值就是你的生成占比，如果它不在 60% 到 85% 之间，那你要么有一个回复格外短的任务，要么有一个配错了的引擎。(*ii*) 记录代码 23.1一百步，把 ϕ 对步数作图；它的斜率就是你的提示池饱和得有多快。(*iii*) 把实际达到的每 GPU 每秒解码词元数除以屋顶线(23.2)；低于 $\eta \approx 0.5$ 就先修引擎，再动算法。实践者常常在 $\eta = 0.3$ 的情况下花掉一个季度去搞一套聪明的采样方案，而 η 低的原因不过是 KV 缓存是按最大上下文而不是按中位数上下文来配的。

23.2 采样效率：不生成没用的词元

下面的干预手段按杠杆大小排序，而它们每一个都是靠引入一个偏差来买到词元的。把是哪一个偏差说清楚，正是一项效率工程与一场对目标函数的缓慢腐蚀之间的区别。

23.2.1 提示过滤与难度匹配

最便宜的词元是那些从未被生成的词元。如果一个提示当前的解出率接近 0 或接近 1，图 23.1说它那一组几乎必然会全体一致，所以理性的做法是不去采样它。动态采样（节 12.4）作用在生成之后，丢弃退化的组并回填批次；那些词元已经花掉了。要在生成之前动手，只需要一个廉价的预测器：每个提示历史通过率的一个指数滑动平均。Zheng et al. (2025c) 正是构建了这个，而 Online Difficulty Filtering (2025) 把在线难度过滤当作一种课程来研究。ScaleRL 的「不重采正例」是其中最粗糙也最稳健的版本，把任何记录通过率超过 0.9 的提示都退役 (Khatri et al., 2025)。

那个偏差。过滤改变了提示分布，所以你优化的不再是 $\mathbb{E}_{x\sim\mathcal{D}}[\cdot]$，而是一个偏向中等难度提示的重加权目标。由此有两个具体的隐患。在通过率 0.95 处退役一个提示，只有在其上的表现稳定时才安全，而以提示为条件的遗忘是真实存在的（节 14.6）：一个在 0.95 处退役的提示可能在无人观察之下衰退到 0.6，所以要定期重新审计退役集合。

而且由于 μ 是当前策略的性质，一个作用在陈旧估计上的过滤器会丢掉那些此后已经变得有信息量的提示。

23.2.2 自适应组大小

一个统一的 G，在一个注定全体一致的提示上与在一个正处于 $\mu = 1/2$ 的提示上花费相同。既然一个组的信息量由奖励方差 $\mu(1-\mu)$ 支配，自然的方案就是两阶段的：用一个较小的 G_0 探测每个候选提示，然后把剩余的词元预算花在探测显示出方差的那些提示上。算法 23.1给出了这个方案。Cheng et al. (2026) 直接研究了这个分配问题，并报告算力最优的 G 随预算上升然后饱和，而机制因难度而异——在容易的提示上是锐化，在困难的提示上是覆盖范围的扩张。2026 年年中有一簇预印本攻的是顺序版本，即随着证据积累而分配采样，而不是分两个固定阶段 (Early Verdicts, 2026; Progressive Rollout, 2026; LEEPS, 2026; MC Pass-k Critic, 2026)；这些个别结果都很新且未经复现，但方向显然是对的，而这个提法如今已成标准。

那个偏差。有两个，而第二个容易被漏掉。第一，一个按方差自适应的分配是上一节那种提示重加权的一种形式。第二，当 G 在提示之间变化时，组均值基线带有依赖于提示的噪声：被采样 4 次的提示，其基线比被采样 32 次的更嘈杂，于是得到的优势是异方差的，各提示上的有效学习率也就不同。修法是把每个提示的损失贡献做归一化，使每个提示的总权重与它的 G_i 无关，这正是算法 23.1最后一步所做的；没有它，自适应分配就会悄悄给那些碰巧拿到更多样本的提示加权。

23.2.3 提前终止

一条注定失败的采样轨迹，有时能在它跑完之前就被识别出来。有三种变体在使用，危险程度递增。硬长度上限在 $L_{\max}$ 处停止；软长度塑形对接近上限的行为施加惩罚 (12.12)；预测式抢占则在部分轨迹于一个廉价分类器或过程奖励模型下得分很差时把该序列杀掉。

常见陷阱: 提前终止会截尾长度分布，而损失会察觉到这一点

截断不是一项中性的节省。如果一个被截断的样本被赋予奖励 0 并留在损失里，策略学到的就是 长是坏的，而不是错是坏的，于是它会不断缩短自己的推理，直到再也解不出难题。DAPO 的超长过滤——把被截断的样本整个从损失中掩蔽掉——被报告为其消融中最大的一步 (Yu et al., 2025c)：他们对被截断的采样轨迹所做的最有价值的事，是把它们扔掉而不是给它们打分。Magistral 把同一点提升为一

算法 23.1 词元预算下的两阶段自适应组大小采样

输入： 提示池 $\mathcal{D}$、探测规模 G_0、每步 M 个词元的预算、目标 B 个有信息量的提示、通过率 EMA $\hat{\mu}$

阶段 1：预过滤与探测

$\mathcal{D}_{\text{cand}} \leftarrow$ 从 $\mathcal{D}$ 中采出 $\lceil \lambda B \rceil$ 个满足 $\hat{\mu}(x) \in (\delta, 1-\delta)$ 的提示 ▷ $\lambda \approx 1.5$

对每个 $x \in \mathcal{D}_{\text{cand}}$，采样 $\{y^{(i)}\}_{i=1}^{G_0} \sim \pi_{\text{inf}}(\cdot \mid x)$ 并打分

$\hat{\mu}(x) \leftarrow (1-\alpha)\hat{\mu}(x) + \alpha\, \bar{r}(x)$ ▷ 刷新这个估计

$M \leftarrow M -$ (探测所花的词元数)

阶段 2：分配余量

$\mathcal{D}_{\text{live}} \leftarrow \{x : 0 < \bar{r}(x) < 1\}$ ▷ 探测显示出方差

$s(x) \leftarrow \sqrt{\bar{r}(x)(1-\bar{r}(x))}\,/\,\hat{L}(x)$ ▷ 每词元的价值

for $x \in \mathcal{D}_{\text{live}}$，按 $s(x)$ 递减 **do**

  $G_x \leftarrow G_0 + \min\left(G_{\max} - G_0,\ \left\lfloor \eta_{\text{a}}\, s(x) M / \hat{L}(x) \right\rfloor\right)$

  采样另外的 $G_x - G_0$ 条回复；扣减 M

  if $M \le 0$ **then break**

end for

阶段 3：构造损失

在每个大小为 G_x 的组内 $A_{x,i} \leftarrow r_{x,i} - \bar{r}(x)$

把提示 x 的词元求和按 $1/G_x$ 加权，使每个提示的总权重与 G_x 无关 ▷ 消除异方差性

return 含 $|\mathcal{D}_{\text{live}}|$ 个组的批次

条系统要求，即批次的完成长度分布必须与真实分布一致，尽管短的补全会先完成 (Mistral-AI et al., 2025)。预测式抢占继承了上面这一切，外加第二个偏差：一个在过往失败上训练出来的分类器，会优先杀掉那些不寻常的轨迹，而它们正是探索所需要的。如果你要抢占，就把被抢占的样本从损失中掩蔽掉，并把保留下来的样本的长度分布与你本来会得到的那个分布做对照监控。

23.2.4 在多个梯度步之间复用采样轨迹

拿为第 k 步生成的那 M 个词元，在它们上面做两次、四次或八次优化步。每个优化步的生成成本按这个倍数下降，其他什么都不变——除了第 2 到第 8 步是离策略的。这不是一个带效率分析的效率技巧；它恰恰就是第22章的那个设定，带着同一套重要性加

权机器（节 22.4）和同一个陈旧度上界（节 22.8）。在一个采样批次上做小轮次、一步离策略的流水线，以及完全异步的生成，是同一根杠杆被拉到不同深度，而深度的上限取决于修正能吸收多少策略漂移，而不是取决于优化器想复用多少。

23.2.5 重放过去的成功

如果一个提示曾经被解出过，那条成功的轨迹就是一份永久资产。LUFFY 把离策略的推理轨迹混入优势计算，并报告这能在纯同策略 RLVR 完全失败的场景下训出模型，代价是监督微调 GPU 小时数的 59%(Yan et al., 2025a)；ReLIFT 则只在最难的问题上交错插入监督更新，使用 13% 的离策略引导 (ReLIFT, 2025)。重放策略自己过去的成功是最便宜的版本，因为它不需要教师。

那个偏差。重放修补的是图 23.1中全错的那一端——一个 $\mu \approx 0$ 的提示，一旦有一次成功被注入它那一组，就变得有信息量——而在偏难的池子里，被浪费的预算恰恰集中在那里。与之相对，被重放的轨迹是陈旧的，它们因成功而被选中因而偏向策略本已擅长的东西，并且反复强化它们会加速熵坍缩。把重放当作对全错尾部的一个修法，而不是当作对生成成本的一般性折扣。

算法 23.2把其中几项组装成一个生产系统真正需要的东西：一个按词元预算而不是按样本数来调度的调度器。要按词元做预算，是因为上面每一项干预都会改变每步词元数，所以一个由「B 个提示」定义的步，其成本会漂移，这会让不同运行之间无法比较（节 23.8）。

23.3 前缀共享与缓存

23.3.1 组采样在构造上就共享一个前缀

组相对方法对每个提示采样 G 条回复。这些回复精确地共享那个提示，所以该提示的键值缓存应当只计算一次，然后被全部 G 个解码器读取。两大引擎都支持这一点——vLLM 中带前缀缓存的分页注意力 (Kwon et al., 2023)，SGLang 中的基数注意力 (Zheng et al., 2024)——而启用它是一个配置开关，不是一个研究项目。这笔算术仍然值得做一遍，因为 FLOP 上的节省与吞吐上的节省大小不同，而后者更大。

在提示长度为 L_{p}、平均补全长度为 $\bar{L}$ 时，一个组处理的词元数是 $L_{\mathrm{p}} + G\bar{L}$ 而不是 $G(L_{\mathrm{p}} + \bar{L})$，节省为

$$\frac{(G-1)L_{\mathrm{p}}}{G(L_{\mathrm{p}} + \bar{L})}. \tag{23.6}$$

表 23.2: 采样效率干预手段，按杠杆大小排序。「词元节省」是一次典型推理运行中可得的数量级，不是实测的加速比；它在该干预直接攻击(23.4)中被浪费的那部分时最大。最后一栏是你要付出的东西，而它从来不是「什么都不付」。

干预	节省	机制	它让你付出什么
按通过率 EMA 过滤提示	1.3–2×	从不生成预测会全体一致的组	重加权了提示分布；陈旧的估计会丢掉刚变得有信息量的提示
退役已饱和的提示	1.1–1.5×	永久移除全对的尾部	退役提示上的回退无人监控
自适应 G	1.2–1.6×	把样本集中到奖励有方差的地方	除非做逐提示权重归一化，否则基线异方差
采样复用／异步	2–8×	把生成摊薄到若干次更新上	离策略偏差；需要节 22.4与一个陈旧度上界
前缀共享	词元上 1.1–1.3×，显存上更多	提示与主干的 KV 只算一次	提示共享没有代价；树形则带来相关的叶子（(13.13)）
重放过去的成功	只针对尾部	给全错的组注入方差	陈旧度、成功选择偏差、更快的熵坍缩
提前终止	1.1–1.4×	停掉注定失败的采样轨迹	截尾了长度分布；压制探索性轨迹
验证记忆化	只省验证器成本	相同的回复只打一次分	没有代价；但命中率上升是一个多样性警报

对 $L_{\mathrm{p}} = 1000$、$\bar{L} = 4096$、$G = 16$，那是词元数的 18%——真实但并不激动人心，而且它高估了对墙钟时间的效果，因为被共享的那些词元属于预填充，而预填充受算力约束，是便宜的那一种。有意思的节省在显存上。共享去掉了每组 $(G-1)L_{\mathrm{p}}\kappa$ 字节的重复缓存，在 $\kappa = 128$ KiB、同样配置下约为 2 GB。按(23.2)，那份显存会直接转化为并发度 b，进而转化为解码吞吐。前缀共享是一项伪装成显存干预的吞吐干预。

树形结构的采样把这个想法从提示延伸到推理主干：从共享的部分轨迹上分叉出 G 个叶子，每段主干只生成一次。节 13.6推导了词元计数(13.12)——深度为三的二叉树可节省 42%——以及它的代价，即叶子不再独立，而聚类设计效应(13.13) 会降低有效组大小。当你本来就想要逐步的信用分配时，树是对的工具；而作为单纯买词元的手段，它是很差的。

多轮次的智能体式采样，是前缀缓存不再只是一项优化、而变成一个预算正确性问题的地方。每一轮都往一个在其余部分完全相同的上下文上追加内容，所以有缓存时总预填充随轮次线性增长，没有缓存时则是平方增长。节 23.1.2把 $K = 20$ 时的比值定在 10.5×。这里有一处微妙之处，已在第22章中标记出来：一个在上一版策略权重下算出的缓存前缀，在一次权重更新之后就是陈旧的，而引擎之所以暴露一个显式的清空开关，正是因为正确性与吞吐在这里冲突。跨过一次权重同步继续复用前缀 KV，是一个用更离策略去换吞吐的刻意选择，它应当被计入陈旧度预算，而不是被当作免费的。

历史注记: 采样引擎先是成了成本中心，然后成了接口

在 InstructGPT 时代的流水线里，生成是训练器的一个子过程：一个框架持有权重、采样并更新。到 2025 年，推理引擎已经是一个独立进程，有自己的内存管理器、自己的数值身份（节 22.1），以及预算的大头。在 2026 年的框架中可见的后果是，那些为 服务而设计的特性——分页缓存与基数缓存、连续批处理、休眠与唤醒、分块预填充、投机解码——都变成了 RL 特性，而引擎 API 也长出了面向 RL 的、用于权重传输与暂停的显式协议 (vLLM Project, 2025; SGLang Project, 2026)。当你预算中最大的那一项由一个不是你写的组件所提供时，最有价值的系统技能就是读那个组件的发布说明。

23.3.2 验证记忆化

奖励是 (x, y) 的函数，所以它们可以被缓存。在钱上这是边际的——节 23.1.2把验证定价为 85 CPU 小时，对上 5,200 GPU 小时——而它要紧是出于另外三个理由。重复很常见，而且会变得更常见：随着熵坍缩（节 12.6），策略会反复吐出同一条补全，于是同一个二元组在组内以及跨步之间被验证很多次。验证延迟在关键路径上，而对一个昂贵的验证器——容器里的一整套测试、一个编译器、一个 LLM 评判模型（第15章）——代价是以秒计而不是以毫秒计，到那个程度它就要与生成争夺墙钟时间了。而且最有用的是，缓存命中率是一个免费的多样性监视器：一次运行的重复率在三百步内从 2% 爬到 30%，那它已经坍塌了，而缓存计数器会比奖励曲线更早显示出来。

代码 23.2: 带答案级规范化的验证记忆化。`duplicate_rate` 属性不花任何代价，而且是熵坍缩最早可得的信号之一。

```
import hashlib
from collections import OrderedDict

```

```python
def canonical_key(prompt_id, response, extract):
    """尽可能按抽取出的答案做缓存键，否则按全文。"""
    answer = extract(response)                    # 例如 \boxed{...} 的内容
    payload = answer if answer is not None else " ".join(response.split())
    digest = hashlib.blake2b(payload.encode(), digest_size=16).hexdigest()
    return (prompt_id, answer is not None, digest)

class VerifierCache:
    def __init__(self, verify, extract, capacity=2_000_000):
        self.verify, self.extract, self.capacity = verify, extract, capacity
        self.store, self.hits, self.calls = OrderedDict(), 0, 0

    def __call__(self, prompt_id, response, reference):
        key = canonical_key(prompt_id, response, self.extract)
        self.calls += 1
        if key in self.store:
            self.hits += 1
            return self.store[key]
        value = self.verify(response, reference)   # 沙箱、sympy、评判模型……
        if len(self.store) >= self.capacity:
            self.store.popitem(last=False)         # FIFO 淘汰
        self.store[key] = value
        return value

    @property
    def duplicate_rate(self):
        return self.hits / max(self.calls, 1)      # 每一步都记录这个
```

有两点要当心。只有当奖励不依赖别的任何东西时，才按抽取出的答案做缓存；一个带格式项、带长度项，或者带一个会读推理过程的评判模型的奖励，就不是答案单独的函数，而按答案做键会悄悄返回错误的奖励。还有，绝不要跨越验证器或答案键的改动去复用缓存——用两者的哈希给缓存打版本，因为一份陈旧的奖励缓存与一个被腐蚀的奖励函数无从区分，而且要难找得多。

23.4 参数高效的强化学习

节 3.7讲过作为一种监督微调技术的低秩适配，这里不再重复：(3.8)定义了参数化，α/r 的缩放让最优学习率大致与秩无关，而内在维度的文献解释了为什么一个小的秩根本就够用。这里的问题不同，也更具体。你能用适配器而不是全量微调来跑强化学习吗，如果这么做，什么会改变？

23.4.1 支持的理由

三条论证，其中通常只有第一条被说出来。

显存。 混合精度下的 AdamW 每个可训练参数大约需要 12–16 字节：一个 7B 策略约合 100 GB 的优化器状态，而在所有线性层上取秩 32 时不到 2 GB。腾出来的显存买到的主要不是「能把模型装下」的能力——第21章的分片本来就能处理那个——而是共置的采样引擎中的并发度，而按(23.2)那就是吞吐。还有第二项不那么显然的节省：有了适配器，参考策略是靠关掉适配器得到的，而不是靠再持有一份权重副本，这就从每一步中去掉了 wN 字节与一次模型加载。

权重同步。 这是那条真正改变系统设计的论证。每一个优化步之后，新权重都必须送达推理引擎。全量微调要广播整个模型——一个 32B 策略在 bfloat16 下是 64 GB——而第21章投入了实打实的机器去隐藏那次传输。一个秩 32 的适配器只有几十兆字节，大约是同步流量 500× 的削减 (Schulman and Thinking Machines Lab, 2025)。到了那个尺寸，传输就不再是一个流水线阶段而变成一条消息，而这正是为什么基于适配器的训练才让多租户 RL 服务——许多次同时进行的微调共享一个基座模型与一套推理部署——在经济上成为可能 (Thinking Machines Lab, 2026)。

服务。 一个基座模型加 n 个适配器，用一套权重服务 n 种特化，对一个按客户或按领域跑 RL 的组织来说，那就是 n 套部署与一套部署之间的区别。

23.4.2 反对的理由，以及证据怎么说

顾虑在于容量。如果 RL 的更新比监督微调的秩更高，那么一个秩 32 的适配器就会成为约束瓶颈，LoRA 也就会给可达到的奖励封顶。有三条证据线与此相关，而它们说的并不完全是同一件事。

信息论论证 (Schulman and Thinking Machines Lab, 2025) 说容量不可能成为瓶颈，节 3.7复述过它：监督学习每个回合交付 $O(\text{词元数})$ 比特，而策略梯度只交付一个优势所携带的 $O(1)$ 比特，所以对千词元量级的回合，RL 每词元吸收的信息大约要少三个数量级。他们的 MATH 运行总共只需吸收约 3×10^5 比特，而一个秩 1 适配器就有 3×10^6 个参数，并且他们报告即便在秩 1 上，LoRA 在策略梯度 RL 上也能与全量微调持平。

权重空间的证据指向同一方向，但带着一个容易被略过的告诫。Mukherjee et al. (2025) 报告 RL 微调改动的是一个小子网络，而Zhu et al. (2026b) 报告了谱继承：RLVR 复用了基座模型的奇异值，改变的是与之关联的奇异向量框架，他们利用这一点做出的优化器在 Qwen3-8B-Base 上以少 2.7× 的步数达到给定准确率。那条告诫是：稀疏不等于低秩——一个只触及零散 0.1% 条目的更新可以接近满秩，而Zhu et al. (2025a) 明确主张 RLVR 学习的方向是偏离主方向的。一个低秩适配器可以表示一个偏离主方向的更新，因为 B 初始化为零并不施加任何子空间先验，但它无法表示一个秩确实超过 r 的更新。因此这些权重空间结果是提示性的而非决定性的，把它们引作「适配器够用」的证明是过度解读。

证据层级。「LoRA 在 RL 上与全量微调持平」属于层级 (c)：一项仔细的研究外加一次独立复现，尚未在前沿规模上、也尚未在长时域智能体任务上被复现，而在那些任务上回合数、因而被吸收的总信息量都要大得多。系统层面的好处则毫无疑问——它们是算术。对 2026 年年中而言，诚实的立场是：在 7B–70B 规模的推理任务上，适配器是 RL 的正确默认选项；这是对既有共识的一次新近反转；而任何要跑一个非常长的智能体式 RL 项目的人，都应该去验证它而不是假定它。

工程实践：用于 RL 的适配器：要紧的设置，以及那个诊断

- **秩 32，所有线性层**（注意力投影、MLP，对混合专家模型还包括专家）。把放置位置限制在注意力上，是无法靠提高秩来补偿的 (Schulman and Thinking Machines Lab, 2025)。
- **学习率约为你全量微调取值的** 10×，所以在全量微调会用 10^{-6} 的地方取 $1\text{–}4 \times 10^{-5}$。照搬全量微调的学习率，是最常见的、导致错误地断定适配器表现不佳的原因。
- **比全量微调所能容忍的更小的有效批次**。
- **用两种方式检测秩是否成为瓶颈**。在同一个学习率下跑秩 r 与秩 $2r$：头一百步应当重合，而一个在此之后才张开并且持续存在的差距，是容量信号而不是噪声。此外独立地检查 BA 的奇异值：一个在 r 以内就迅速衰减的谱意味着容量有富余，而一个在全部 r 个方向上几乎平坦的谱，则意味着适配器用尽了它拥有的每一个方向，秩可能已经成为瓶颈。
- **量化的基座（QLoRA）**。截至 2026 年年中，还没有强有力的公开证据刻画 RL 下 4 比特冻结基座的表现。请注意这与节 23.5的问题不同：为适配器训

> 练而量化基座，改变的是被训练的那个对象，而量化采样引擎改变的是那个采样器。

代码 23.3: 基于适配器的 RL：它与全量微调循环的两处差别，是权重同步的载荷与参考策略。

```python
from peft import LoraConfig, get_peft_model

ALL_LINEAR = ["q_proj", "k_proj", "v_proj", "o_proj",
              "gate_proj", "up_proj", "down_proj"]   # 不要只放注意力

policy = get_peft_model(base_model, LoraConfig(
    r=32, lora_alpha=32, lora_dropout=0.0,
    target_modules=ALL_LINEAR, bias="none", task_type="CAUSAL_LM"))

# 全量微调在这里会用约 1e-6；LoRA 想要的大致是它的十倍。
opt = torch.optim.AdamW(policy.parameters(), lr=2e-5, weight_decay=0.0)

def sync_to_engine(engine, policy):
    """几十兆字节，而不是几十吉字节。"""
    delta = {k: v.detach() for k, v in policy.state_dict().items() if "lora_" in k}
    engine.update_adapter(delta)          # vLLM / SGLang 适配器热替换

def reference_logprobs(policy, batch):
    """pi_ref 就是基座模型：关掉适配器，不必持有第二份副本。"""
    with policy.disable_adapter(), torch.no_grad():
        return policy(**batch).logits.float().log_softmax(-1)
```

23.5 量化的经济学

生成占账单的四分之三，所以采样引擎的精度是整条流水线中最大的单个成本杠杆。第22章把低精度当作一个正确性问题来处理，而 表 22.1把采样量化列为对采样器与训练器之间间隙贡献最大的平滑来源之一。这里我们要问的是它值多少。

23.5.1 量化生成买到了什么

回到(23.2)。把权重从 bfloat16 移到 FP8 会把 wN 减半；把键值缓存移到 FP8 会把 κ 减半，而后者既把每一步的缓存读取量减半，又把显存能容纳的并发度 b 翻倍。分母的两项都缩小，而分子却变大。对定理 23.1的配置，这个界从每 GPU 每秒 2,500 个词元升到 5,630 个，b 从 54 升到 121：采样加速 2.24×。应用到表 23.1上，这会把 7B 那次

运行从 5,200 降到约 3,000 GPU 小时——一次配置改动就带来整次运行总成本 42% 的削减。

报告出来的收益比屋顶线预测的要小，而这本就应当如此：FP8-RL (2026) 报告 FP8 的权重与激活线性层加上 FP8 键值缓存，带来至多 44% 的采样吞吐提升；Jet-RL (2026) 报告一套统一的 FP8 流程带来至多 33% 的采样提升与 16% 的端到端提升。2.2× 与 1.4× 之间的差距正是工程所在：注意力与归一化仍留在更高精度，量化与反量化要花周期，而流水线的一部分根本不受带宽约束。用屋顶线来决定这份力气值不值得开始，用报告出来的数字来设定预期。

23.5.2 为什么修正方法才是让这份节省安全的东西

这笔交易是真实的：更低精度的生成每词元更便宜，同时会放大采样与训练之间的间隙，而超过某个间隙之后，运行不只是退化，而是坍塌（节 22.3）。历史上这让量化采样成了实践者试过一次就放弃的东西。改变这一点的是，截尾重要性采样给出了一种有原则的方式，把量化引擎的分布当作行为策略并对它做修正：Yao et al. (2025b) 报告这让量化采样变得可行，同时保住了下游准确率，而 2026 年的两套低精度技术栈都建立在同一个想法、或者它那个精度契约的替代方案之上。

核心思想：一个修正方法解锁了一项成本节省

这是本书中一项算法结果在基础设施上自负盈亏的最清楚的实例。截尾重要性采样被提出来是为了修一个正确性 bug——采样器与训练器不一致——而它的后果是经济性的：它把「不要量化采样」变成了「量化采样，并付出一个有界的偏差」，而这大约值一次推理 RL 运行成本的 40%。修正方法与成本杠杆不是两个分开的话题。第22章应当被读作本节的先修内容，而不是旁支材料；反向的蕴含也成立：一个没有实现那个修正的组织，就拿不到这份节省。

反方立场同样值得给予分量。Jet-RL (2026) 主张，在长采样轨迹与困难任务上，bfloat16 训练配 FP8 采样「遭受严重的训练不稳定与灾难性的准确率坍塌」，而正确的应对是通过在两侧使用 同一种精度来移除失配，而不是去修正它；Blackwell 原生的那项工作则把支配原则表述为一份贯穿采样、训练、检查点转换与在线权重更新的单一精度契约 (LMSYS Org, 2026)。这两种立场可以同时成立——两侧都控制不了就做修正，都能控制就做统一——而截至 2026 年年中两者都是层级 (c)，因为相关论文都只有几个月大。

表 23.3: 按目标划分的量化，按每单位风险的价值递减排序。「节省」是相对于所指组件的 bfloat16 基线而言，而采样那一行是由(23.2)算出的、不是实测的。最后一栏正是这张表属于一个效率章节、而不只属于第22章的理由：对第一行来说，只有在修正到位时那份节省才是可得的。

目标	节省	风险	前提条件
采样权重与 KV 缓存（FP8）	在 76% 的成本上 1.4–2.2×	放大采样器与训练器的间隙；不做修正则是灾难性的	TIS 或 MIS（节 22.4），或一份统一的精度契约（Jet-RL, 2026）
参考策略	wN 字节的 HBM	低：扰动一个 KL 估计，部分被 β 吸收	无；或者改用适配器并把它关掉
奖励模型	只省推理成本	低，但噪声会翻转接近平局的比较	在你自己的采样轨迹上测量量化与精确的一致率
采样，4 比特（NVFP4）	至多 3×	失配大于 FP8，源于更高的参考 KL(LMSYS Org, 2026)	贯穿全部四个阶段的比特精确量化器契约
训练的前向与反向	成本的 14%	改变的是梯度本身；见第22章	统一流程；不要与 bfloat16 采样混用
适配器下的冻结基座（QLoRA）	只省显存	截至 2026 年年中对 RL 尚未被评估	自己跑一次消融

23.5.3 流水线中的其余一切

表 23.3按风险调整后的价值给各个选项排序。这个规律简单到可以用一句话说完：对那些输出被当作标量使用的模型放心量化，对那个输出被当作采样分布使用的模型要小心，而对那个输出会被求导的模型要非常小心。

参考策略是最容易的情形。它只通过一个 KL 估计进入，所以它的误差扰动的是一个本就近似的正则项，并且部分被吸收进 β——而正如节 1.8所指出的，β 本来就与奖励尺度不可分辨。奖励模型几乎同样容易，因为它的输出在组内被中心化，任何对全组共同的畸变都会被抵消。残余的风险是具体的：量级为 ϵ 的量化噪声会翻转那些真实边距小于 ϵ 的比较，而接近平局之处恰恰是奖励模型被要求去分辨的地方。请在你自己的采样轨迹上、以小边距为条件，测量量化与精确之间的一致率，而不是在某个基准上测。

有一个真正出人意料的结果属于这里，作为对过于整齐的推理的一个警示。Huang et al. (2025b) 把量化噪声不当作低精度的代价，而当作一种探索助力，理由是新增的噪声在 RL 需要熵的地方提高了策略熵。这只有一篇论文，机制也没有确立，但它提醒我们「噪声是坏的」是一条启发式，不是一条定理。

23.6 下一单位算力该花到哪里？

一份预算可以花在更大的策略、更多的优化步、每个提示更多的样本、更多的提示、更好的奖励与环境、RL 之前的中期训练，或者 RL 之后的推理时搜索上。文献回答了其中一些而没有回答另一些，而一本教科书能做的最有用的事，就是把哪些被回答了说清楚。

步数对上其余一切是可回答的，而答案是一个形状。 Khatri et al. (2025) 把奖励对训练算力拟合成一个 S 形(12.14)，带一个渐近线 A 与一个效率指数 B，并报告大多数配方选择——损失聚合、课程、长度控制、优势归一化——移动的是 B 而不是 A。节 12.12给出了方法学上的结论。分配上的结论是另一回事，也同样有用：如果一项干预移动的是 B，那么它的价值就以它在达到某个固定奖励时所省下的算力为上界，而一旦你接近渐近线，它就一文不值。效率工作是有时限的。

采样预算的划分是可回答的。 Cheng et al. (2026) 攻的正是这个分配问题——如何把一份固定预算划分到 G、每批提示数与更新步数上——并报告算力最优的 G 随预算可预测地上升然后饱和，每批提示数主要影响稳定性且可在一个很宽的范围内选取，而提高 G 会减少不同问题之间的干扰。他们把标度律当作规定性的分配规则而不是描述性的曲线，这个提法对本章而言正是对的那个。

花在 RL 之前的算力常常胜过花在 RL 之中的算力。 这是这一时期最强的分配结果，而通常不被当作一个分配结果来呈现。Runwal et al. (2026) 在四个模型家族的七个基座模型上跑了数百次受控实验，并报告一份大约 27B 词元的中期训练预算——与预训练相比微不足道——就能带来很大的推理增益，同时保住预训练获得的知识；而试图通过 RL 来获得同样知识的模型只有微弱的提升。Wang et al. (2025g) 从另一侧展示了同样的不对称。如果你的策略根本做不了某件事，那么用 RL 去教它是一种昂贵的办法：RL 锐化的是中期训练所提供出来的东西（节 4.7）。

表 23.4: 各条算力分配轴、2025–2026 年文献就每一条确立了什么，以及证据的诚实状态。规律是：一次 RL 运行内部的那些轴如今已用真实预算研究过，而跨越阶段边界的那些轴——尤其是模型规模对上 RL 算力——则没有。

轴	已确立的内容	层级
更多 RL 步数	奖励对算力呈 S 形；大多数配方选择移动的是效率 B，不是渐近线 A (Khatri et al., 2025)	(b)
每提示样本数 G	算力最优的 G 随预算上升然后饱和；机制因提示难度而异 (Cheng et al., 2026)	(c)
每批提示数	主要影响稳定性；可接受范围很宽 (Cheng et al., 2026)	(c)
RL 之前的中期训练	RL 之前一份小的词元预算，比同样预算花在 RL 里买到的更多 (Runwal et al., 2026; Wang et al., 2025g)	(b)
更好的环境与奖励	被广泛断言压过算法选择；没有受控的分配研究	(c)
更大的策略	RL 没有公开的联合 (N, C) 前沿；训练之后再往下蒸馏 (DeepSeek-AI, 2025)	—
推理时搜索	与训练算力互换，并有一个可计算的、以被服务查询数计的盈亏平衡点（第10章）	(a)

模型规模对上 RL 算力，从公开工作中无法回答。 本章无法引用任何一个 RL 版的、Chinchilla 式的关于 (N, C) 的联合前沿。已经确立的是更窄的东西：把一个更大的、经过 RL 训练的模型蒸馏进一个更小的模型，能迁移走大部分增益 (DeepSeek-AI, 2025)，而「先改架构再恢复」的流水线会刻意在昂贵的 RL 阶段之前把模型缩小 (NVIDIA, 2025)。两者在操作层面指向同一方向——在你负担得起的最大规模上做能力工作，然后再压缩——但两者都不是标度律，而把这个前沿问题当作已经解决是一个错误。

训练算力对上推理算力有一个盈亏平衡点。 第10章展开了这个取舍；这里给出的是它的核算版本。如果 RL 把每个被服务查询所需的词元数减少了 Δ，那么在 n 次查询上节省的就是 $n\Delta/\nu$ GPU 秒，于是一次花费 H GPU 小时的 RL 在 $n^\star = 3600\, H\, \nu/\Delta$ 时回本。对表 23.1的 7B 运行（$H = 5{,}200$、$\nu = 1760$），若每次查询节省 $\Delta = 2000$ 个词元，则 $n^\star \approx 1.6 \times 10^7$ 次查询。那个数字才是思考效率导向的 RL 的正确方式：一次缩短思维链的运行是一笔对着服务量的投资，而在大约一千万次查询以下它并不能自负盈亏。这也正是为什么高效推理的那一支文献 (Sui et al., 2025) 是一支经济学文献。

23.7 蒸馏通常比强化学习更便宜

那个令人不适的实践事实，直说出来就是：对大多数组织在大多数时候，从一个更强的模型做蒸馏比跑 RL 更便宜也更有效。节 14.8解释了机制——蒸馏引入能力并以遗忘为代价，RL 做锐化并保留住原有能力——而本节补上成本这一面。

这个不对称在于信息密度。策略梯度每个回合交付一个优势的 $O(1)$ 比特；蒸馏则在每一个词元上交付监督。成本模型里什么都没变——两者都由生成主导——但每个生成词元所带来的学习量差了好几个数量级，而且蒸馏的生成是一次性开销，可以被过滤、在多个学生与多个轮次之间复用，或者直接从供应商那里买。Lu and Thinking Machines Lab (2025) 报告一个 0.5B 的学生在数学推理上追平了一个 1.5B 的教师，算力比 RL 低 9–30×，而同策略蒸馏以大约十分之一的算力超过了 RL 的分数。把那些数字当作层级 (c) 来对待，它们来自一份非存档的单篇报告，但要注意方向正是比特计数论证所预测的，而且Kim et al. (2025) 的正面对比与之一致。此外还没有验证器要建，而第19章整整一章讲的就是那有多大工作量。

工程实践：RL 什么时候真的值它的成本

在至少满足下列一条时才跑 RL，一条都不满足时就该怀疑自己。

1. **你的领域里不存在更强的教师。**你就在前沿上，或者那个前沿模型拿不到，或者它的许可证禁止蒸馏，或者你的领域是专有的、没有任何教师见过它。这是主要的正当情形，也是为什么前沿实验室要跑 RL 而大多数其他人不该跑。
2. **你需要那条保留性质。**同策略 RL 在 KL 上停留在起点附近，因而比在同一目标上做监督微调遗忘得更少 (Shenfeld et al., 2025)。如果你是在给一个既有行为很难保住的模型添加一项能力，这是值得付钱的。
3. **一个验证器给你教师给不了的信号。**你自己的测试套件、你自己的模拟器、你自己的产品指标。教师只能告诉你它会怎么做；验证器告诉你什么是真的。这也是唯一一种学生能可靠地超过教师的情形，NVIDIA (2025) 的 RL 阶段就是如此。
4. **你需要的是可靠性而不是触及范围。**蒸馏挪动的是边界；RL 让内部变得可依赖（节 14.8）。如果 pass@k 已经够用而 pass@1 不够，那就是一个 RL 问题。

表 23.5: 效率指标、每一项能检测到什么，以及每一项会被怎样操纵。请把前四项一并报告：其中任何一项单独拿出来，都可以靠一个让运行变差的改动而得到改善。

指标	定义	检测什么	被什么操纵
每 GPU 小时的奖励	$\Delta R/H$	整个系统的效率，含引擎	买一台更快的机器；跨硬件比较
每生成词元的奖励	$\Delta R/M$	算法上的采样效率，与硬件无关	缩短回复，而这可能拉低渐近线
对梯度有贡献的占比 ϕ	(23.3)	被浪费的生成，以及提示池随时间的饱和	把池子过滤到只剩中等难度的提示，这会收窄所学到的东西
渐近线与效率 (A, B)	(12.14)的拟合	一项干预是抬高了天花板还是只提高了速度	在过短的窗口上拟合；不正确地排除早期区间
重复率	代码 23.2	多样性坍塌，且很早	没有；它是一个诊断量，不是一个目标
达到屋顶线的比例 η	实测的 ν 除以(23.2)	引擎配置错误	缩短上下文长度

> 另外要注意在 2026 年大多数流水线中占优的那个选项：两个都做，顺序是先蒸馏再强化，而这正是 R1 配方所做的（节 11.4）。

23.8 诚实地度量效率

本章的每一个主张都是关于一个比值的主张，而比值很容易被误报。表 23.5汇集了那些让比较变得有意义的测量，其中前四项应当始终一并报告。

最要紧的那个区分，是节 12.12在渐近线 A 与效率指数 B 之间划出的那一条。本章几乎所有内容都是 B 上的干预：提示过滤、自适应组大小、前缀共享、量化采样以及适配器，都是更快地到达同一个地方，而没有改变那个地方在哪里。这不是一句批评，因为你付钱买的就是 B。但它确实意味着，一项在固定预算下被验证的效率干预看起来会像一次能力提升，而当有人把两种配置都跑到饱和时，它就不再像了。

常见陷阱：按相同步数而不是按相同词元数做比较

这是效率文献中最常见的测量错误，而且它恰好美化了本章推荐的那些干预。动态采样每个优化步生成的词元比固定采样更多，因为它要丢弃并回填。自适应 G 生成的数量是变化的。采样复用生成得更少。提前终止生成得更少但改变了样本。因此一条奖励对步数的曲线比较的是成本不同的各种方法，而它会报告说动态采样是一次很大的改进，部分原因不过是它的每一步都更贵。请把奖励对累计生成词元数或对 GPU 小时数作图；如果你必须报告逐步的曲线，就把每步词元数一并报告。节 12.12在其清单的第三条里讲的是同一点；这里值得重讲一遍，因为这个失误是系统性的而不是偶发的，也因为偏差的方向总是同一个。

第二个、也更微妙的陷阱：ϕ 是一个诊断量，是一个很差的目标。它在一个完全由 $\mu = 1/2$ 的提示组成的提示池上被最大化，而那样一个池训出来的策略只会在中等难度的问题上平庸，别的什么都不会。困难提示的 ϕ 低，而能力正是从那里来的；已饱和的提示 ϕ 也低，而它们是你绝不能回退的东西。优化这个诊断量，就是节 1.7主题五的古德哈特式失效被套用到了一个效率指标上。

23.9 一份成本控制清单

工程实践：按杠杆大小排序的干预，以及每一项让你付出什么

自上而下地做；每一步都假定它上面的那些已经做完。

1. **测出这份拆解**（生成占比、ϕ、η）。代价：一个下午。下面每一个决策都依赖它。
2. **先修引擎，再修算法。**打开前缀缓存，KV 缓存按中位数而不是按最大上下文来配，连续批处理，训练侧的序列打包。代价：工程，没有偏差。典型的回收：η 从 0.3 到 0.7，那是在账单的四分之三上取得 2×。
3. 用通过率 EMA **在生成之前过滤掉退化提示**，并退役已饱和的提示。代价：一个被重加权的提示分布，以及一份重新审计退役提示的义务。
4. **在截尾重要性采样到位的前提下，把采样引擎量化到 FP8**（节 22.4）。代价：一个有界的重要性权重偏差，外加 节 22.2的那些诊断。大约值整次运行的 40%。
5. **在有界的优化步数内复用采样轨迹**，或者转到一条带显式陈旧度上界的异步流水线（节 22.8）。代价：离策略偏差，随上界增长。

6. 除非你已经测出秩确实成为瓶颈，否则**用适配器训练**。代价：一个层级 (c) 的容量假设；买到的是显存、~ 500× 更小的权重同步，以及一个免费的参考策略。
7. **逐提示自适应地调** G（算法 23.1）。代价：中等的工程量，以及一个你绝不能忘记的逐提示权重归一化。
8. **记忆化验证**并记录重复率。代价：没有代价，前提是缓存的键包含了奖励所依赖的一切，并且随验证器一起打了版本。
9. **设上限并掩蔽，不要截断并惩罚**。代价：如果你做掩蔽，就没有代价；如果你把截断的样本判为错误，代价就是一个变短了的策略。
10. **重新考虑你到底该不该跑 RL**（节 23.7）。代价：机构的面子。

本章小结

- 一次 RL 后训练运行是按生成的词元计价的，每个优化步 $M = BG\bar{L}$ 个，而在每一个规模上生成都约占账单的四分之三。这个占比是由硬件的算术吞吐与内存带宽之比决定的，不是由模型决定的，而那个比值一直在变差。
- 解码受带宽约束，所以吞吐由(23.2)支配，并随并发序列数上升。任何腾出显存的手段——前缀共享、量化、适配器——都买到了生成吞吐，这正是为什么这些原本互不相干的干预会出现在同一张清单上。
- 在二值奖励与组相对优势之下，全体一致的组贡献的梯度恰好为零。被浪费的占比恰好是 (23.4)，而对一个现实中呈 U 形的提示池，那是生成预算的 40–55%。失败比成功更长，所以在偏难的池子上，按词元加权的浪费超过按提示加权的浪费。
- 杠杆最大的干预是在生成之前预测退化，而不是在生成之后把它丢掉。它们每一个都是用一个偏差来买词元：提示过滤重加权了目标函数，自适应组大小让基线变成异方差的，提前终止截尾了长度分布，而采样复用干脆就是离策略训练。
- 前缀共享是一项回报体现在吞吐上的显存干预；在多轮次的智能体式采样中，前缀缓存就是预填充随轮次线性增长与随轮次平方增长之间的差别。
- 在 7B–70B 规模的推理任务上，适配器是 RL 的正确默认选项，主要是因为权重同步的载荷坍缩到约 1/500，而且参考策略变成免费的。支撑这一点的容量论证是信息论的，而它的实证支持只有一项仔细的研究；稀疏与低秩的权重空间结果是提示性的，但并不能确立它。

- 量化采样引擎是最大的单个成本杠杆，按屋顶线约值一次推理运行的 40%，在已报告的实践中是 15–30%。它之所以安全，只是因为截尾重要性采样、或者一份统一的精度契约，吸收了它所制造的失配：一个正确性方法解锁了一项成本节省。
- 一次 RL 运行内部的分配问题如今已用真实预算研究过——奖励对算力呈 S 形，而算力最优的组大小先上升然后饱和。模型规模对上 RL 算力这个问题没有已发表的前沿，而花在 RL 之前的中期训练上的算力，常常胜过同样的算力花在 RL 之中。
- 对大多数组织而言，蒸馏比 RL 更便宜也更有效，因为它在每一个词元上交付监督，而不是每个回合交付一个标量。RL 值它的成本，是在不存在更强教师时、在保留性要紧时、在验证器能提供教师给不了的信号时，或者在缺的是可靠性而不是触及范围时。
- 本章几乎每一项干预移动的都是算力效率指数 B，而不是渐近线 A。因此比较必须在相同词元数下进行，绝不能在相同步数下进行；而对梯度有贡献的占比必须作为诊断量来报告，不能作为目标来优化。

延伸阅读

先读Khatri et al. (2025) 与Cheng et al. (2026)，按这个顺序，并当作一对来读。前者确立了 RL 奖励随算力沿一条饱和曲线上升，而大多数已发表的干预移动的是你爬得多快而不是你爬得多高；后者把那个观察变成了把采样预算划分到组大小、批大小与步数上的规定性规则。两者合起来是这个领域最接近一套算力分配方法学的东西，而前者中那条方法学上的警告，比它任何一项个别发现都更有价值。

关于生成本身的成本，请把引擎论文当作系统论文而不是工具文档来读：Kwon et al. (2023) 讲分页注意力以及为什么内存管理才是吞吐的关键，Zheng et al. (2024) 讲基数注意力与前缀复用。然后按Yao et al. (2025b)、FP8-RL (2026)、Jet-RL (2026) 的顺序读低精度 RL 的工作，它把一条论证贯穿了两年：失配是一个 bug；这个 bug 可以被修正，而这份修正给你换来了量化采样；或者，也许失配应当通过统一精度来移除。LMSYS Org (2026) 是精度契约立场最新的表述，即便它所针对的具体格式已被取代，它的提法仍值得一读。

关于参数高效的 RL，Schulman and Thinking Machines Lab (2025) 是第一手来源，篇幅短、论证清楚，并且对它没有测试过的东西很坦诚；读它的同时读Biderman et al. (2024a)，后者看似相反的结论，一旦把适配器容量与训练信号的信息量相比较就

干净地化解了（节 3.7）。Mukherjee et al. (2025) 与Zhu et al. (2026b) 是权重空间的补充，把它们放在一起读很有教益，恰恰是因为它们对低秩结论的支持比表面看上去更间接。

最后，关于到底该不该跑 RL：Lu and Thinking Machines Lab (2025) 讲蒸馏的成本论证，Kim et al. (2025) 讲那次受控对比，而Shenfeld et al. (2025) 讲那条保留性质，它是 RL 在经济上最强的正当理由。Sui et al. (2025) 综述了让推理更便宜的相邻文献，而那里正是一次效率导向的 RL 运行的回报被真正收取的地方。

习题

23.1（**屋顶线算术**。）一个 32B 稠密模型有 64 层、8 个维度为 128 的分组查询键值头，在四张 H100（每张 80 GB、3.35 TB/s）上以张量并行做服务。计算 κ、在 16,384 词元的工作上下文下能装下的并发序列数，以及 (23.2)给出的每 GPU 每秒解码词元数的上界。然后在 FP8 键值缓存下把这三项都重算一遍，并说明在每种情形下(23.2)分母中的两项哪一项占主导。

23.2（**被浪费的占比**。）从 Beta 函数的定义出发验证(23.5)。对 $\mathrm{Beta}(0.3, 0.3)$、$\mathrm{Beta}(1, 1)$ 与 $\mathrm{Beta}(3, 3)$，把 $1 - \phi$ 对 G 作图，并找出 U 形池的浪费首次降到 30% 以下时的 G；考虑到一步的成本对 G 是线性的，评论提高 G 是否是一种明智的达成方式。然后证明：若 F 关于 $\mu = 1/2$ 对称且 $\ell(\mu)$ 是仿射的，则(23.4) 中的长度加权不改变任何东西；并构造一个池与一条长度曲线，使它把被浪费的占比抬高十个百分点以上。

23.3（**高效推理的盈亏平衡**。）一次花费 8,000 GPU 小时的 RL 运行，在准确率不变的前提下把平均回复长度从 6,000 个词元降到 3,500 个。服务以每 GPU 每秒 1,200 个词元运行。计算这次运行自负盈亏所需的被服务查询数。现在假设这次运行还把准确率降低了一个百分点，而一个百分点的准确率对你值每次查询 \$0.002；重做这个计算。哪一项占主导，而这对「谁该去做效率导向的 RL」意味着什么？

23.4（**是哪个偏差?**）对下列每一项干预，指出它给策略梯度引入了什么偏差，并说明它是否有界、由什么给出界。(a) 丢弃退化的组而不做 $\alpha = B/(B - M)$ 的重标度。(b) 对部分轨迹在过程奖励模型下低于某阈值的采样轨迹做抢占，并把它们判为错误。(c) 在一个采样批次上做八次优化步而不做任何重要性修正。(d) 在一个组内共享该提示的键值缓存。(e) 退役记录通过率超过 0.9 的提示。(f) 用一个 FP8 引擎供给采样，并配词元级的截尾重要性采样。

23.5（自适应分配，以及它的基线。）在算法 23.1中，假设提示 x_1 拿到 $G = 4$，提示 x_2 拿到 $G = 32$，两者的真实成功概率都是 $\mu = 0.5$，奖励是二值的。计算每种情形下组均值基线的方差，进而算出单个优势的方差。证明：没有最后一步的归一化，每个提示的期望平方梯度贡献在两者之间是不同的；并给出能把它拉平的加权方式。

23.6 ⋆ **（一个能真正解决某个问题的分配实验。）** 表 23.4把「更大的策略」列为没有公开的联合前沿。请设计能产出一个这样的前沿的最小实验：指定模型规模、算力预算、跨格子固定住什么、你会如何拟合(12.14)的一个二维类比，以及你会如何防范Khatri et al. (2025) 指出的那个失效模式，即小预算下的排名在大规模上会反转。估算总 GPU 小时数，并诚实说明你的设计在前沿实验室之外是否负担得起，如果不是，你会先砍掉什么。

23.7 ⋆ **（低秩论证在长时域下还成立吗?）**支持适配器的信息论论证按每个回合 $O(1)$ 比特来计数。对一个带逐轮次奖励、$K = 50$ 轮的长时域智能体任务，每个回合的信号要更大。请分别算出在 (i) 单个终局奖励、(ii) 逐轮次二值奖励、(iii) 一个给每一步打分的过程奖励模型之下，各大多少。对每一种，计算被吸收的总信息量达到一个 70B 模型上秩 32 适配器参数量时所需的回合数，并说明这预测了 LoRA-for-RL 会最先在哪里失效。然后解释为什么即便算术是对的，这个预测也可能是错的——特别是，为什么「信号中的比特数」是「必须被存进权重的比特数」的一个上界。

23.8 † **（给一次运行装上仪表。）**取任意一个在 1–8B 模型上的开源 RLVR 配方，加上三件仪表：每一步都记录的代码 23.1、带重复率的代码 23.2，以及一次针对你自己硬件、把实际解码词元每秒数对屋顶线(23.2)的测量。跑 200 步。报告 (a) ϕ 的轨迹以及占主导的是全对项还是全错项，(b) 重复率对上平均逐词元熵，(c) 你的 η。然后实现你的测量所指出的那个杠杆最大的单项修法，重跑一遍，并报告两种配置下奖励对累计生成词元数的曲线——不是对步数。

23.9 † **（端到端地给量化这根杠杆定价。）**用同一个配方，在固定的词元预算下跑三种配置：bfloat16 采样；FP8 采样且不做修正；FP8 采样配词元级截尾重要性采样。对每一种，记录墙钟时间、实际达到的解码吞吐、节 22.2中训练对上采样的对数概率间隙，以及奖励对累计词元数的曲线。把实际实现的加速比与屋顶线预测的 2.24× 相比较，并解释这个差距。说明在什么条件下你会部署那个未做修正的配置，如果存在这样的条件的话。

算法 23.2 一个优化步的预算感知采样调度器

输入： 词元预算 M、容量为 b 条序列的引擎、按预测可学习性排序的提示队列 Q、陈旧度上界 S、目标 B 个有信息量的组

输出： B 个奖励方差非零的组，每个组陈旧不超过 S 个策略版本，且总共不超过 M 个生成词元

$\mathcal{R} \leftarrow \emptyset$（已完成的组）；$m \leftarrow 0$（已花的词元）
while $|\mathcal{R}| < B$ **且** $m < M$ **do**
  在空闲 KV 块允许的范围内从 Q **准入**提示，并在该提示的 G 个样本之间共享它的 KV　▷ 节 23.3
  把解码推进一个调度周期；$m \leftarrow m +$ (吐出的词元数)
  for 每个样本全部终止了的组 g **do**
    $r_g \leftarrow \textsc{VerifyCached}(g)$　▷ 代码 23.2
    if r_g 全体一致 **then**
      记录 $\hat{\mu}$ 的更新，**丢弃** g，且不把它计入 B　▷ 退化
    else
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{g\}$
    end if
  end for
  if 有新的策略版本到达 **then**
    中断在途的序列，为它们的 KV 与部分文本做检查点，并在同步之后恢复它们　▷ 部分采样轨迹；节 21.6
  end if
  if 任何序列超过 $L_{\max}$ **then** 截断它并把它标记为 已掩蔽，而不是错误
end while
return $\mathcal{R}$，连同 m 与实际实现的 ϕ

第 VI 部分

测量、安全与前沿

第 24 章

评测与测量

核心思想：多数被报告的后训练改进，用其所提供的证据根本测不出来

一个基准分数是一个估计值，而这个估计量的方差可以由题目数、每题的采样数以及题目难度的离散程度，以闭式算出来。当你对这个领域实际使用的那些基准做这项计算时——AIME 三十道题，一个终端智能体一百个容器——单一种子的比较所能可靠检出的最小差异，比几乎所有被报告的差异都要大。而这种偏差不是随机的：数字看着好了，人们就不再调了。本章讲的是如何知道一项改动到底有没有用，并且它对通行做法是刻意持敌意的，其中也包括本书通篇所引论文的做法。

第 VI 部分从构建系统转向评判系统。它必须放在算法之后，因为在你见识过一套 RL 配方有多少个自由参数之前，你无法体会那些证据有多脆弱；它也必须放在安全与前沿之前，因为那两者都是换了身衣裳的测量问题。

24.1 我们究竟要测什么，以及为什么难测

三种彼此不同的属性被压缩进了「性能」这一个词里，而这种压缩要为相当一部分混乱负责。

能力 是策略支撑集的属性：只要给足尝试次数，模型究竟能不能产生一条正确的轨迹？这正是大 k 处的 pass@k 所估计的量（节 2.8），也是第14章中「诱发还是习得」之争所争的量。能力变化缓慢，主要由预训练与中期训练决定。

行为 是策略众数的属性：当你只问一次时，它实际会做什么？这就是 avg@n、指令遵循的符合度、拒答率、语气、冗长程度、工具调用语法。行为是后训练改变得最快、也最可靠的东西，也是用户实际体验到的东西。

可靠性 是策略尾部的属性：部署一百次，出现不可接受结果的频率有多高？可靠性几乎从来没有被测量过，因为有意思的那些分位数所需的样本量没有人肯出预算，也因为真正要紧的失效恰恰是基准设计者没有预料到的那些。

一种方法可以只移动其中之一而不动其余，而且有几种方法会把它们朝相反方向推。奖励最大化会让策略集中：它抬高行为分数，却可能压低能力分数（节 14.2）。安全训练在预料之内的危害上提高可靠性，却在良性的边缘情形上压低行为分数。由于论文报告的是一个行为数字，标题却写成能力断言，文献便系统性地夸大了已经取得的成就。

24.1.1 信号很小，噪声不小

一旦配方本身已经过关，后训练的效应就很小。一个新的优势归一化器、一次课程改动、一项长度惩罚——这些都是一到三个百分点量级的干预，而正如节 24.2 将精确算出的那样，它们所在基准的测量噪声要大上数倍。这个局面与 Card et al. (2020) 在大模型之前的自然语言处理中所记录的局面在结构上完全一致：典型的实验设计并不具备检出人们通常所声称的效应大小的功效，而这个领域的回应是报告不带区间的点估计，而不是把实验做大。

有三个特有的加剧因素是后训练所独有的。

第一，**基准很小**，而且随着难度上升而越来越小。AIME 每年三十道题。GPQA-Diamond 有 198 道。Terminal-Bench 2.0 有 89 个容器化任务 (Merrill et al., 2026; Stanford University and Laude Institute)。这个领域朝更难、更抗污染、更智能体式的评测（节 24.7）迈进，是用统计功效换来了效度，而且大体上没有承认这笔交易。

第二，**训练过程本身是随机的**。两次仅在种子上不同的 RL 运行会到达不同的策略。Khatri et al. (2025) 的缩放研究是 RLVR 文献中功效设计得最仔细的一项，他们报告在三个种子之间，拟合出的渐近通过率 A 的不确定性大约是 ± 0.02。这意味着仅仅来自种子，就有一条 ± 2 个百分点的带宽落在一次运行的上限上，而这还没有加上任何基准采样噪声——并且它已经大于第12章中所声称的多数算法效应。节 12.12给出的诊

断是：小预算下的端点比较在大预算下可能反转；这里给出的诊断是：那些端点比较中有许多在任何预算下都区分不开这些方法。

第三，也是最糟糕的，**选择偏差是有方向的**。没有人做完一次消融、看到一个有利的数字之后还继续调参。没有人会把十四组没跑通的超参设置当作十四个负结果报告出来。没有人每一百步评测一次检查点，然后报告最后一个而不是最好的那一个。这些每一样都是一种形式的选择性停止（optional stopping），而每一样都会抬高被报告的效应。这正是 Card et al. (2020) 以及在更宽背景下的Ioannidis (2005) 所指认的机制：功效不足的文献会累积起一批发表出来的效应，而它们日后会向零收缩。

历史注记: 新建测试集反复得出同一结论

测量一个被报告的分数中有多少是真的，最干净的办法是从同一分布再造一个测试集重新测量。Recht et al. (2019) 对 ImageNet 这样做过，发现准确率一律下降 11 到 14 个百分点，而排名大体保持——对公开测试集的自适应过拟合确实存在，但基本上只是一个水平位移。Zhang et al. (2024a) 对小学算术做了同样的事，构造了在分布与人类解答率上与 GSM8K 匹配的 GSM1k，发现某些模型家族的准确率下降多达 13 个百分点，另一些则毫无下降，而下降幅度与模型逐字吐出 GSM8K 题目的实测概率相关。两者给出的教训不是基准无用，而是在公开集上测得的一个水平不可迁移，而一个排名往往可以——并且当下降是家族特异的时候，排名的迁移也一样失败。

24.2 基准比较的统计学

本节是全章的技术核心。它不含任何新研究；它含的是一些初等计算，而这些计算只要有人做过，就能了结这个领域大部分方法论争议。

24.2.1 题目层面的模型

固定一个含 n 道题目 $x_1, \dots, x_n$ 的基准和一个评分器。对策略 π_θ，定义逐题成功概率

$$p_i \;=\; \mathbb{P}_{y \sim \pi_\theta(\cdot \mid x_i)}[\text{grader accepts } y]. \tag{24.1}$$

本节的一切都来自两个事实：p_i 是未知的、必须由 k 个样本估计，以及这 n 道题目本身也是从我们真正关心的某个任务总体中抽出的一个样本。

每题抽取 k 条 i.i.d. 回复，令 $x_{ij} \in \{0,1\}$ 为评分结果。自然的估计量是

$$\hat{\mu} \;=\; \frac{1}{n}\sum_{i=1}^{n} s_i, \qquad s_i \;=\; \frac{1}{k}\sum_{j=1}^{k} x_{ij}, \tag{24.2}$$

这就是被报告为 avg@k 的量，或当 $k=1$ 时被报告为准确率的量。记 $\bar{p} = \frac{1}{n}\sum_i p_i$ 为其期望，σ_p^2 为题目难度在题目总体上的方差，$\overline{p(1-p)}$ 为平均的题内伯努利方差。

推导：两种方差、两个问题，以及人们混淆的那一个

存在两个不同的推断目标，它们有不同的标准误。

目标（a）：在这 n 道题目上的分数。此时 $\{p_i\}$ 是固定常数，唯一的随机性来自解码：

$$\mathbb{V}\mathrm{ar}(\hat{\mu} \mid \{p_i\}) \;=\; \frac{1}{n^2 k}\sum_{i=1}^{n} p_i(1-p_i) \;=\; \frac{\overline{p(1-p)}}{nk}. \tag{24.3}$$

目标（b）：在该基准所抽样的任务总体上的分数。现在题目也是随机的，由全方差公式

$$\mathbb{V}\mathrm{ar}(\hat{\mu}) \;=\; \frac{1}{n}\Big(\sigma_p^2 + \frac{\overline{p(1-p)}}{k}\Big) \;=\; \frac{\bar{p}(1-\bar{p})}{n}\Big(\lambda + \frac{1-\lambda}{k}\Big), \tag{24.4}$$

其中 $\lambda := \sigma_p^2/(\bar{p}(1-\bar{p})) \in [0,1]$，且我们用到了恒等式 $\overline{p(1-p)} = \bar{p}(1-\bar{p}) - \sigma_p^2$。在 $k=1$ 时这恰好退化为熟悉的二项表达式 $\bar{p}(1-\bar{p})/n$，因此那正是目标（b）的标准误，而不是目标（a）的。

参数 λ 是总方差中位于题目之间的那一部分。当基准是「模型总能解出的题」与「模型从来解不出的题」的混合时它很大——这是一个趋于饱和的基准的常态——而当多数题目对模型来说真的像抛硬币时它很小。它可以用矩匹配的办法，把观察到的逐题分数离散程度与题内二项方差对上来估计，不过代码 24.1的聚类自助法直接估计 (24.4)，无需任何这类代入式插值。

(24.3)与(24.4)的区别不是咬文嚼字。它是后训练文献中最常见的那个统计错误。

常见陷阱：除以 nk

在 $n = 30$ 道 AIME 题上各生成了 $k = 16$ 个样本之后，人们很容易去报告 $\sqrt{\hat{\mu}(1-\hat{\mu})/(nk)}$——一个基于 480 次「独立」试验的标准误，大约 2.3 个百分点。对于任何人真正在问的那个问题，这是错的。由(24.4)，正确的标准误要大出一个因子 $\sqrt{1+\lambda(k-1)}$；在 $\lambda = 0.5$、$k = 16$ 时这个因子是 2.9。这与 `evalci` 库 (evalci, 2026) 所报告的那个促成其诞生的发现在数量上是一致的：在真实评测表上正确

做聚类的标准误，常常约为朴素标准误的三倍。其机制在于，对同一道题的额外解码并不是额外的题目：它们缩小(24.4)中的 $(1-\lambda)/k$ 项，却完全不动 λ 项。

24.2.2 最小可检测效应

现在在同一个基准上比较两个策略 A 与 B。令 $d_i = s_i^A - s_i^B$，$\hat{\Delta} = \frac{1}{n}\sum_i d_i$。采样噪声在两个策略之间是独立的，所以只有题目难度分量会协变；记 ϱ 为真实题目难度 p_i^A, p_i^B 在两个策略之间的相关系数。为可读起见假设两个策略有相同的 $\bar{p}$ 与 λ，

$$\mathbb{V}\mathrm{ar}(\hat{\Delta}) \;=\; \frac{2\bar{p}(1-\bar{p})}{n}\Big(\underbrace{\lambda(1-\varrho)+\frac{1-\lambda}{k}}_{=:\,\kappa(\lambda,\varrho,k)}\Big). \tag{24.5}$$

水平为 α 的双侧检验，在真实差异为 Δ 时具有功效 $1-\beta$ 的条件是 $|\Delta| \geq z_{\alpha,\beta}\sqrt{\mathbb{V}\mathrm{ar}(\hat{\Delta})}$，其中 $z_{\alpha,\beta} = z_{1-\alpha/2} + z_{1-\beta}$。于是有最小可检测效应

$$\boxed{\Delta_{\min} \;=\; z_{\alpha,\beta}\sqrt{\frac{2\bar{p}(1-\bar{p})\,\kappa(\lambda,\varrho,k)}{n}}} \tag{24.6}$$

在 $\alpha = 0.05$ 与 80% 功效下 $z_{\alpha,\beta} = 1.960 + 0.842 = 2.802$。

式(24.6)值得盯着看一会儿，因为它说的是：两件可用的方差削减工具攻击的是互不相交的项。配对（提高 ϱ）消去题目难度方差，对采样噪声毫无作用。在 k 次解码上取平均消去采样噪声，对题目难度方差毫无作用。二者单独都不够，而 n 是唯一能同时缩小两者的杠杆。表 24.1在这个领域实际使用的规模上求出了 (24.6)的值。

这张表应当被当作一纸控诉来读。在一年的 AIME 上做单次解码的比较，检不出任何小于约三十六个百分点的差异；而最好的可用设计——按题目配对、十六次解码、难度高度相关——仍然检不出小于约十四个百分点的差异。两到五个百分点的 AIME 改进在 2025—2026 年的 RLVR 文献中比比皆是、而且常常就是头条结果，它们比自己实验的噪声底大约低一个数量级。同样的话对 GPQA-Diamond 以及当前使用的每一个智能体式基准也成立，只是没有那么戏剧性。图 24.1给出了整条曲线。

24.2.3 配对比较，以及 McNemar 检验几乎无人使用的原因

表 24.1中配对带来的收益之所以只是中等，仅仅因为我假定了 $\varrho = 0.5$。实践中两个相似模型答对哪些题之间的相关性是很高的：Miller (2024) 报告，前沿模型在流行评测

表 24.1: 在 $\alpha = 0.05$、80% 功效下的最小可检测效应，以百分点计，按 (24.6)在最坏情形 $\bar{p} = 0.5$ 与 $\lambda = 0.6$ 下精确算出。「SE」一列是单次解码的标准误 $\sqrt{0.25/n}$。四个比较列分别是：单次解码不配对；难度相关系数 $\varrho = 0.5$ 的配对；配对且每题 $k = 16$ 次解码；以及 $\varrho = 0.8$ 且 $k = 16$。这里每一个数字都是乐观的，因为它假定只做了一次预先登记的比较、每一臂只有一个训练种子，且不存在污染。

基准	n	SE	不配对	ϱ=.5	$+k$=16	ϱ=.8, k=16
AIME（单一年份）	30	9.1	36.2	30.3	20.6	13.8
Terminal-Bench 2.0	89	5.3	21.0	17.6	12.0	8.0
GPQA-Diamond	198	3.6	14.1	11.8	8.0	5.4
MATH-500、SWE-bench Verified	500	2.2	8.9	7.4	5.1	3.4
HLE（公开划分）	2500	1.0	4.0	3.3	2.3	1.5
MMLU	14042	0.4	1.7	1.4	1.0	0.6

上的题目级分数相关系数在 0.3 到 0.7 之间，而对于同一次训练运行中仅隔了一次干预的两个检查点，这个数值要高得多——通常在 0.9 以上，因为它们共享一个基座模型、一份数据配比，以及绝大部分参数。在 $\varrho = 0.9$ 时，配对消去了 90% 的题目难度方差，这在题目方差贡献上是三倍的削减，而且是白拿的。

对于 $k = 1$ 的二元结果，精确的配对检验是 McNemar 检验 (McNemar, 1947)。把不一致对列成表：有 b 道题 A 答对而 B 答错，有 c 道反之。一致的题目不携带关于差异的信息，而在零假设下这 $b + c$ 道不一致的题目各占一半，所以这个检验就是对 $b + c$ 次试验中 b 次成功相对于 $1/2$ 的精确二项检验。估计出的差异是 $\hat{\Delta} = (b - c)/n$，零假设下的标准误为 $\sqrt{b + c}/n$，因此最小可检测效应只取决于 不一致率 $\delta = (b + c)/n$：

$$\Delta_{\min}^{\text{McNemar}} \approx z_{\alpha,\beta} \sqrt{\frac{\delta}{n}}. \tag{24.7}$$

两个几乎相同的策略只在很少的题目上分歧，所以 δ 很小，而 (24.7)可以比不配对的公式好上数倍。对于后训练研究中最常见的那种比较，这是可用的功效最高的检验，它只需要逐题的结果表，用 `scipy` 写一行就够，而它在本书所引的论文中基本上一篇也没出现过。原因不是对统计学有分歧；原因是论文报告的是一对标量而不是一对表格，而一旦表格被丢弃，配对就无从恢复了。发布逐题结果表应当像发布分数一样成为惯例。

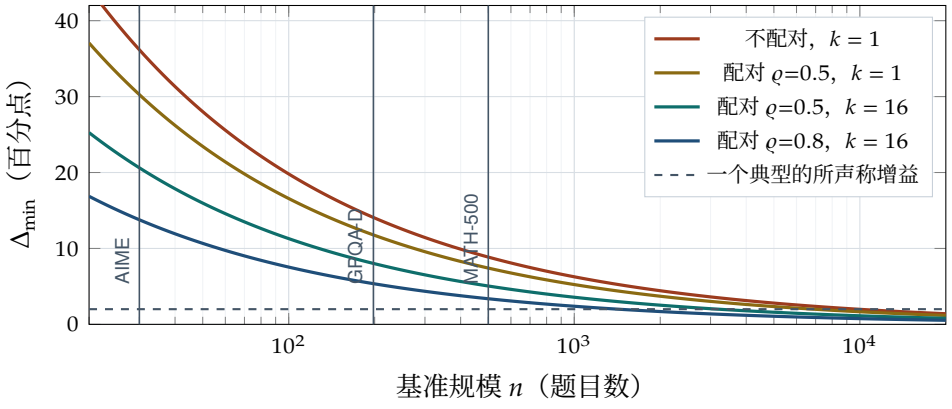


图 24.1: 最小可检测效应作为基准规模的函数，按(24.6)在 $\bar{p} = 0.5$、$\lambda = 0.6$、$\alpha = 0.05$、80% 功效下精确算出。四条曲线都以 $n^{-1/2}$ 下降；设计上的选择最多把截距改变 2.6 倍。水平虚线标出两个百分点的差异，也就是多数后训练论文宣告胜利的尺度：即便在最有利的设计下，它也一直位于每一条曲线之下，直到 n 超过大约 1,400 道题目为止。

常见陷阱: 种子方差是你所测效应的一部分

配对与重采样处理的是评测噪声。它们对训练噪声毫无办法。如果你每一臂只跑一个训练任务，哪怕把评测配得完美无缺，你的置信区间描述的也只是那两个特定检查点之间的差异，而不是两套流程之间的差异——而论文声称要比较的正是那两套流程。形式地说，流程层面差异的方差要加上一项 $2\sigma^2_{\text{seed}}/S$，其中 S 是每一臂的训练种子数，而 RL 中的 σ_{seed} 并不小：Khatri et al. (2025) 在三个种子间报告的渐近值 ± 0.02 的带宽，意味着 σ_{seed} 在一到两个百分点的量级。在 $S = 1$ 时这一项无法削减，而且通常压倒表 24.1中的一切。这件事在深度强化学习中自Henderson et al. (2018) 与Colas et al. (2018) 以来就已被理解，而那里提出的补救——多个种子、报告分布而非最大值、使用稳健的聚合量与分层自助法区间 (Agarwal et al., 2021)——可以原封不动地迁移过来。尴尬的算术在于，每一臂三个种子会把每次消融的成本变成三倍，这就是为什么没人这么做，也是为什么这一文献中的消融表应当被当作提出假设而不是给出结论来读。

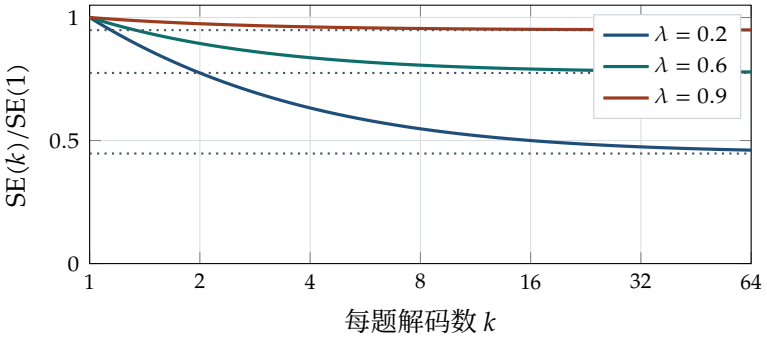


图 24.2: 按(24.8)，每题平均 k 次解码所带来的方差削减。点线是渐近线 $\sqrt{\lambda}$。额外的解码无法把标准误压到题间分量之下，所以在一个双峰的基准上——多数题目要么总能解出、要么从不能解出，$\lambda \to 1$——重采样几乎买不到什么，唯一的补救是增加题目。这在任何意义上都不是示意图：这些曲线就是精确公式。

24.2.4 对多次解码取平均，及其上限

由于 avg@k 只缩小(24.4)中的 $(1-\lambda)/k$ 项，它的收益会饱和。相对于单次解码可达到的标准误削减是

$$\frac{\mathrm{SE}(k)}{\mathrm{SE}(1)} = \sqrt{\lambda + \frac{1-\lambda}{k}} \xrightarrow[k\to\infty]{} \sqrt{\lambda}, \tag{24.8}$$

所以一个 $\lambda = 0.6$ 的基准，无论你买多少推理算力，都永远不可能好过其单次解码标准误的 79%。图 24.2画出了这条曲线。实用的规则是把 k 选在曲线变平之处，通常 $k \in [8, 32]$，然后把剩下的预算花在题目上而不是解码上。Miller (2024) 把同一结论表述为一条设计准则——把 k 选到使题内方差项相对题间项足够小——并追加了一条值得复述的警告：不要为了降方差而降温度。降低温度改变的是你所测量的那个分布，而他的反例表明这既可能抬高方差也可能让均值产生偏差，因为把一个宽分布舍入到它的众数，会产生一个既可能更多变、中心也可能与原分布不同的伯努利变量。

24.2.5 十二个基准，一个结论

一篇现代后训练论文会报告一张含十到十五个基准的表，并因为方法在其中大多数上取胜而下结论说它有效。这里有两件事出了问题。

第一，**族系错误**。如果一篇论文在 $\alpha = 0.05$ 下做了 m 次独立检验，至少出现一个伪显著结果的概率是 $1 - 0.95^m$，在 $m = 8$ 时为 0.34，$m = 12$ 时为 0.46。如果论文报告的是 m 个受噪声主导的差异中最大的那一个，那么期望的膨胀就是 m 个标准正态变

量的期望最大值：$m = 8$ 时为 1.42σ，$m = 12$ 时为 1.63σ。当 σ 在一个 500 题基准上是三个百分点量级时，「我们的方法在最好的基准上提升了五个百分点」与「该方法什么也没做」是相容的。修正手段是标准的——用 Holm–Bonferroni 控制族系错误率，或用 Benjamini–Hochberg(Benjamini and Hochberg, 1995) 控制错误发现率——其代价不过是对做了多少次比较保持诚实。

第二，也更难修的，是**这些基准并不独立**。取胜是相关的，因为这些基准在技能上彼此重叠；落败也是相关的，因为一个破坏了格式的方法会破坏一切。相关的检验会让 Bonferroni 变得保守，也会让「十二个里赢了九个」远比在十二次独立试验上做二项计算所提示的证据要弱。唯一站得住脚的聚合方式，是一个预先登记的主要指标，其余的作为次要指标报告并明确标为探索性。后训练论文不做预先登记，而Gelman and Loken (2014) 是解释这为何即便在无人作弊时也要紧的标准参考文献：分析路径是在看过数据之后才选定的，所以名义上的错误率并不是实际的错误率。

对这一切的补救，是在花掉算力之前把(24.6)倒着算一遍。先决定你愿意据以行动的最小差异 $\Delta^\star$——对一个生产模型来说，主要指标上两个百分点是一个站得住脚的门槛——然后解出能在 $\Delta^\star$ 处给出 80% 功效的 (n, k, S)。在 $\bar{p} = 0.5$、$\lambda = 0.6$、$\varrho = 0.8$ 且 $k = 16$ 时，检出 $\Delta^\star = 2$ 个百分点需要约 1,400 道题目。如果你的基准只有 200 道，你有三个选项，而且只有三个：把若干基准汇集成一个预先登记的主要指标；接受你只能检出大效应并在摘要里说明；或者干脆不做这个实验。选择第四个选项——照跑不误，然后报告点估计——正是这一文献走到今天这个状态的方式。算法 24.1把整个流程汇总起来，而代码 24.1给出了两个真正要紧的检验。

代码 24.1: 面向评测表的配对推断：对二元结果的精确 McNemar 检验，以及同时对题目与解码重采样的两级聚类自助法。两者都以逐题结果表为输入，而那正是论文应当发布的产物。

```python
import numpy as np
from scipy.stats import binomtest

def mcnemar_exact(a_ok, b_ok):
    """Paired test for binary per-item outcomes. a_ok, b_ok: bool arrays of len n."""
    b = int(np.sum(a_ok & ~b_ok))          # A 对，B 错
    c = int(np.sum(~a_ok & b_ok))          # A 错，B 对
    if b + c == 0:                         # 没有不一致的题目：没有信息
        return 0.0, 1.0, (b, c)
    delta = (b - c) / len(a_ok)
    p = binomtest(b, b + c, 0.5, alternative="two-sided").pvalue
    return delta, p, (b, c)

```

算法 24.1 对两套后训练流程做功效充分的配对比较

输入： 流程 $\mathcal{P}_A, \mathcal{P}_B$；题池 $\mathcal{D}$；目标效应 $\Delta^\star$；水平 α；功效 $1-\beta$

设计（在花算力之前）
在 $n_0 \ll n$ 道题目上以 k_0 次解码跑两臂的试点；估计 $\bar{p}$、λ、ϱ ▷ 几个 GPU 小时
解(24.6)求在 $\Delta^\star$ 处达到功效 $1-\beta$ 的最小 (n,k)
if 所需 $n > |\mathcal{D}|$ **then**
  return UNDERPOWERED：扩大题池、提高 $\Delta^\star$，或放弃
end if
每一臂选取 $S \geq 3$ 个训练种子；提前把种子清单固定下来
冻结评测框架的提交号、答案抽取规则、词元预算、温度与系统提示
执行
for $a \in \{A,B\}, s = 1$ **to** S **do**
  训练 $\theta_{a,s} \leftarrow \mathcal{P}_a(\text{seed } s)$；在全部 n 道题目上解码 k 条回复
  保存完整的 $n \times k$ 结果表，而不是均值
end for
推断
$\hat{\Delta} \leftarrow s_i^A - s_i^B$ 在题目与种子上的均值
$(\text{lo}, \text{hi}) \leftarrow$ 在题目、解码与种子上做聚类自助法 ▷ 代码 24.1
$p \leftarrow$ 在 k=1 切片上做精确 McNemar 检验；按所报告的基准个数做校正
return $\hat{\Delta}$、区间、p、在 $\Delta^\star$ 处达到的功效，以及各结果表

```python
def paired_cluster_bootstrap(sa, sb, n_boot=10000, seed=0):
    """sa, sb: (n_items, k) graded outcomes for the same items in the same order.
    Resamples items (the clusters) and decodes within items."""
    rng = np.random.default_rng(seed)
    n, k = sa.shape
    delta = sa.mean() - sb.mean()
    boot = np.empty(n_boot)
    for t in range(n_boot):
        i = rng.integers(0, n, n)                    # 对题目重采样
        ja = rng.integers(0, k, (n, k))              # 对解码重采样
        jb = rng.integers(0, k, (n, k))
        boot[t] = (np.take_along_axis(sa[i], ja, axis=1).mean()
                   - np.take_along_axis(sb[i], jb, axis=1).mean())
    lo, hi = np.quantile(boot, [0.025, 0.975])
    return delta, (lo, hi), boot.std(ddof=1)
```

关于自助法有一点需要说明。Bowyer et al. (2025) 表明，当 n 低于大约一百时，中心极限定理区间与自助法区间的覆盖率都严重不足，而这描述的正是 AIME、Terminal-Bench 以及多数智能体式套件。对这些情形，在单模型 i.i.d. 的场合应使用 Wilson 得分区间 (Wilson, 1927)，在配对或聚类设计中应使用贝叶斯可信区间；在频率派替代方案覆盖率不达标之处，它们实测的覆盖率接近名义水平。这在本书中是罕见的一例：一个方法论问题拥有干净、便宜且现成的解法。

24.3 pass@k 及其估计量

节 2.8定义了Chen et al. (2021) 的无偏估计量：从 $n \geq k$ 个样本中的 c 次成功出发，

$$\widehat{\text{pass@}k} \;=\; 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}, \tag{24.9}$$

它对 $\text{pass@}k = 1-(1-p)^k$ 是严格无偏的，因为 $\binom{n-c}{k}/\binom{n}{k}$ 正是从这 n 个抽出的样本中均匀随机取一个 k 元子集、其中不含任何成功的概率。我们现在要问的是，把它用好要付出什么代价。

为什么代入式估计量是有偏的。 自然的替代做法是估计 $\hat{p} = c/n$ 然后报告 $1-(1-\hat{p})^k$。由于 $g(p) = (1-p)^k$ 是凸的，Jensen 不等式给出 $\mathbb{E}[g(\hat{p})] \geq g(p)$，因而 $\mathbb{E}[1-g(\hat{p})] \leq \text{pass@}k$：代入式估计量 低估。二阶展开可以把它量化，

$$\mathbb{E}\big[1-(1-\hat{p})^k\big] - \text{pass@}k \;\approx\; -\tfrac{1}{2}k(k-1)(1-p)^{k-2}\,\frac{p(1-p)}{n}, \tag{24.10}$$

在 $p = 0.2$、$k = 16$、$n = 16$ 时其值约为 -7.6 个百分点，而真实的 pass@16 是 0.972；在 $n = 64$ 时收缩为 -1.5 个百分点，$n = 256$ 时为 -0.3。反方向的错误更常见也更危险：把 k 取成你恰好抽了多少个样本，然后把「n 个中至少一个正确」当作某个更小的 k 的 pass@k 报告出来，这是向上有偏的，因为 n 个样本上的最大值并不是从 k 个样本最大值的分布中抽出的。

pass@k 需要多少样本。 估计量(24.9)在 $n = k$ 时就是一次伯努利抽样，标准差可高达 0.5；要有可用的精度就需要 $n \gg k$。精确的逐题标准差是对 $c \sim \text{Binomial}(n,p)$ 的有限求和，列于表 24.2。

表 24.2: 无偏估计量(24.9)的精确逐题标准差，通过对 c 的二项分布求和算出，针对一道单样本成功概率 $p = 0.05$ 的题目。精度大致以 $\sqrt{k/n}$ 改善，所以固定精度下的采样预算随 k 线性增长：一个可信的 pass@64 每题需要约一千次解码。

	pass@k	$n{=}k$	$n{=}2k$	$n{=}4k$	$n{=}8k$	$n{=}16k$
$k = 2$	0.098	0.297	0.208	0.147	0.104	0.073
$k = 4$	0.185	0.389	0.269	0.188	0.133	0.094
$k = 8$	0.337	0.473	0.318	0.220	0.154	0.108
$k = 16$	0.560	0.496	0.315	0.212	0.146	0.102
$k = 32$	0.806	0.395	0.220	0.139	0.093	0.064

核心思想: pass@k 交叉断言是文献中要做扎实代价最高的断言

节 14.2与Yue et al. (2025a) 都建立在这样一个论断上：基座模型与其经 RL 训练的后代这两条 pass@k 曲线，会在某个大 k 处相交。要确立一次相交，需要在两条曲线的大 k 处都有精度，而表 24.2说，这样的精度每题要花 $n \approx 16k$ 次解码——在 $k = 64$ 时每题大约一千个样本——再乘以(24.4)的总体层面标准误所需的题目数，再乘以两个模型，再乘以训练种子数。这场争论的任何一方都几乎没有论文供得起这笔预算，而且几乎没有一篇报告了交叉点的区间。这并不意味着该现象不真实：Wu et al. (2025a) 给出了一种机制，Zhou (2026) 刻画了它发生所需的提示区间，而 PASS@(k,T) Analysis (2026) 在组合式的智能体任务上报告了相反的符号。它的确意味着，那一文献中的定量分歧在很大程度上是功效不足的测量之间的分歧，而这与「结论最终是『两者都对，只是任务类型不同』」这一观察是一致的。

代码 24.2: 无偏的 pass@k 估计量，在对数空间中计算以免溢出，附带一个同时对题目重采样并重抽解码计数的两级自助法标准误。只报告前者而不报告后者，是最常见的错误。

```python
import numpy as np
from scipy.special import gammaln

def _log_binom(n, k):
    return gammaln(n + 1.0) - gammaln(k + 1.0) - gammaln(n - k + 1.0)

def pass_at_k(n, c, k):
    """Unbiased estimate of 1-(1-p)^k from c successes in n>=k iid samples."""
    if k > n:
```

```
        raise ValueError("pass@k is not estimable from fewer than k samples")
    if n - c < k:                       # 每个 k 元子集都含一次成功
        return 1.0
    return -np.expm1(_log_binom(n - c, k) - _log_binom(n, k))

def passk_mean_and_se(counts, n, k, n_boot=4000, seed=0):
    """counts[i]: successes on item i out of n decodes. Returns the benchmark-level
    pass@k and a standard error valid for the *task population*, not just these items."""
    counts = np.asarray(counts, dtype=int)
    mu = np.mean([pass_at_k(n, c, k) for c in counts])
    rng, m, p_hat = np.random.default_rng(seed), len(counts), counts / n
    boot = np.empty(n_boot)
    for b in range(n_boot):
        idx = rng.integers(0, m, m)                     # 对题目重采样
        c_star = rng.binomial(n, p_hat[idx])            # 重抽解码
        boot[b] = np.mean([pass_at_k(n, c, k) for c in c_star])
    return mu, boot.std(ddof=1)
```

报告标准由此机械地推出。写明实际抽取的解码数 n；写明 k；写明估计量；写明温度、top-p 与最大词元预算；并给出在题目而非解码上算出的标准误。一个不带 n 的 pass@k 数字是无法解读的，而一个 $n = k$ 的 pass@k 数字不过是带了小数点的抛硬币。

24.4 污染

正确的先验是：你的评测集就在你的训练数据里某处。不是因为有人作弊，而是因为训练语料占据了可触及互联网中相当大的一部分，因为合成数据生成器自己也是在其上训练的，也因为整理推理数据的人正是去寻找那种能做成好基准的题目。

对截至 2025 年末五十五项污染研究的系统综述 (Contamination Detection Review, 2026) 把这个问题组织成四个层级，而这些层级是恰当的教学工具，因为可检测性会随着你往下走而崩塌。*T1*，精确：基准题目逐字出现。*T2*，句法：它以重排格式、重排顺序或重命名变量的形式出现。*T3*，语义：它以改写、翻译，或表层数字不同的同一道题的形式出现。*T4*，任务层面：题目本身没有出现，但它的解法出现了，出现在一篇专门讲那道竞赛题的教程里。在那篇综述中，MMLU、GSM8K、HumanEval、HellaSwag 与 PIQA 上被报告的污染效应大致跨越 6 到 40 个百分点，但需注意这些估计来自不同方法，彼此并不可直接比较。

表 24.3汇总了各类检测家族以及各自漏掉了什么。综述直截了当地陈述的那个诚实结论是：没有任何现有方法在所有层级、所有访问设定与所有训练阶段上都可靠——

表 24.3: 污染检测家族，依Contamination Detection Review (2026) 的分类法。「层级」指该家族有可能检出的污染层级。没有任何家族覆盖 T4，也没有任何家族对指令微调或 RL 阶段引入的污染是可靠的，而那正是本书所处的区间。

家族	层级	需要	失效于
金丝雀字符串	T1	语料	题目被复制时未带金丝雀串；多数题目根本没有
精确／归一化匹配	T1	语料	任何重排格式；翻译；数值扰动
n 元组重叠（n=8–13）	T1–T2	语料	改写；一段共享的长样板文字会抬高假阳性
嵌入检索	T1–T3	语料	阈值是任意的；合法训练数据中本就有近重复
似然／困惑度探针	T1–T2	logits	指令微调过的模型；RLHF 会把信号抹平
成员推断（e.g., Min-K%）	T1–T2	logits	单轮次训练；去重过的语料；记忆余量很小
提示模型自述	T1–T2	仅需 API	模型并不知道；拒答；编造出来的回忆
基准审计（ConStat）	间接覆盖 T1–T4	仅需分数	需要参考模型和一个匹配的参考基准
全新的留出基准	全部	一个新数据集	昂贵；把污染与分布偏移混为一谈

而最难检测的阶段是指令微调，那恰恰是本书所讲的那个阶段。凡是要报告后训练结果的人，这句话都该读两遍。

有两条污染通道是后训练特有的，表 24.3中任何在训练语料上操作的行都覆盖不了它们。

合成数据污染。 如果你用一个强教师来生成 SFT 或 RL 提示，教师对基准的知识就会流进你的数据，哪怕你的文件里从来没有出现过任何基准字符串。向一个前沿模型索要「奥赛风格的高难几何题」，返回的就是把数字换掉的 AIME 题。用字符串重叠去检查你自己生成的语料什么也查不出来，因为按其构造方式，泄漏发生在 T3 或 T4 层级。可用的控制手段只有两个：来源纪律——记录哪个生成器用哪些种子产生了哪些提示，以及它的训练截止日期 (Longpre et al., 2024; Gebru et al., 2021)——以及下面的全新基准检查。

环境污染。 这是 RL 特有的情形，而且被严重低估。一个通过爬取 GitHub issue 及其修复提交而搭建的智能体式环境（第19章），取材自与 SWE-bench (Jimenez et al., 2023) 相同的那些代码仓。在源自 `django` 的环境上训练、又在来自 `django` 的 SWE-bench 实例上评测，这不是字符串重叠，也不会被当作字符串重叠抓到；这是在代码仓、维护者风格、有时甚至是字面上相邻的那次提交这一层面上，在测试分布上做训练。缓解办法是让训练环境与评测实例在代码仓层面互不相交，并用来源元数据而不是文本匹配来强制执行，同时诚实地承认：这限定了一个被报告的 SWE-bench 增益中有多少可以归功于算法。

唯一经得起推敲的证据。 跑一遍算法 24.2，报告每一项标记率，然后去做那件真正能区分问题的事：在一个于你的数据截止日期之后构造的基准上评测。LiveCodeBench 那份滚动收集的截止后竞赛题 (Jain et al., 2025a)、Humanity's Last Exam 的私有留出划分 (Phan et al., 2025)，以及智能体式套件的历次刷新 (Stanford University and Laude Institute)，都是为此而存在的。一个增益能在全新题目上存活的方法是有证据的；一个增益蒸发掉的方法则需要给出解释。注意那个再怎么小心也去不掉的残余混杂因素：一个全新基准与原基准的差别，除了新鲜度之外还有难度与主题，所以下降既可能是污染的证据，也可能是分布偏移的证据，而把两者分开需要有已知截止日期的参考模型，这正是 ConStat 的设计 (Dekoninck et al., 2024)。永远不要不加限定地写下「已去污染」这个词。写「我们在阈值 x 下检出并移除了 T1–T2 重叠，我们的截止后结果是 y」。

24.5 把评判模型当作仪器

对于不可验证的领域——写作质量、有用性、研究报告、多轮对话——自Zheng et al. (2023a) 以来默认的评测者就是另一个语言模型。这是站得住脚的，同时它也是一个决定：用一台误差剖面结构化、幅度大且依赖于模型的仪器来做测量。就把它当作仪器：在信任读数之前先刻画它的误差。

表 24.4列出了有文献记录的失效模式，以及缓解手段与残余风险。有几项值得展开。

一致性并不是你以为的那个东西。 评判者的头条数字通常是与人类标注的精确匹配一致率。在一个三类标签集（A 胜、B 胜、平局）上，机遇基线大约是 1/3，所以 0.80–

算法 24.2 针对一项后训练结果的污染审计

输入： 基准 $\mathcal{B}$；SFT 提示 $\mathcal{D}_{\text{sft}}$；RL 提示 $\mathcal{D}_{\text{rl}}$；环境清单 $\mathcal{E}$；生成器日志 $\mathcal{G}$；预训练语料（若可访问）

语料侧（可捕获 T1–T2，便宜）
在全部语料中扫描基准的金丝雀字符串；任何命中都判定不合格
归一化（大小写、空白、LaTeX、数字写法）后做精确匹配；记录命中率
把每一道基准题目与每一份训练文档做 13 元组重叠；记录被标记的比例，并人工检查排名前 100 的
语义（可捕获 T3，昂贵，不完备）
嵌入全部基准题目并检索最近的 50 条训练条目；对每个基准的最佳匹配做人工复核
扰动 $\mathcal{B}$：重命名变量、改变数值常数、翻译；重算分数。大幅下降即为记忆的证据
后训练特有
从 $\mathcal{G}$ 中列出用于合成提示的每一个生成器及其训练截止日期；标记任何晚于 $\mathcal{B}$ 发布的生成器
从 $\mathcal{E}$ 中列出源代码仓、issue 编号与软件包版本；断言其与 $\mathcal{B}$ 在代码仓粒度上互不相交
决定性的一步
在一个晚于 $\mathcal{G} \cup \mathcal{E}$ 中最晚截止日期发布的基准上评测；报告相对 $\mathcal{B}$ 的配对差异
return 所有标记率、扰动后的下降幅度，以及截止后分数；**不要**返回一个二元的「干净」裁决

0.85 的精确匹配远没有听上去那么令人印象深刻。Anonymous (2026b) 评测了三个基准上的二十一个评判者，报告在 MT-Bench 上精确匹配把经机遇校正的判别能力高估了 33.8 到 41.3 个百分点，其中 0.80–0.85 的精确匹配对应 Cohen's $\kappa \approx 0.48$。关键在于，这个差距的大小是基准标签边缘分布的属性，而不是评判者质量的属性：随着标签从均衡的三分类转向不均衡的二分类，平均差距从 MT-Bench 上的 38.6 个百分点收缩到 JudgeBench 上的 23.7，再到 RewardBench 上的 10.2。他们还发现评判者的排名在基准之间不可迁移——二十一个评判者中只有两个在三个基准上都保住了前三的位置——这意味着单基准的评判者验证几乎什么也说明不了。

评判模型与奖励模型最不擅长什么。 RewardBench 2 (Malik et al., 2025) 把它的前身 (Lambert et al., 2024) 围绕四选一格式重新设计，使机遇基线是 25% 而不是 50%，并且围绕全新的人类提示而不是从下游评测回收来的提示——正是后者这个设计缺陷让第一版可被污染。领先模型的得分比在 RewardBench 上大约低二十个百分点。结果表的形状就是那个发现：精确指令遵循是一切都失手的地方，相对 25% 的基线，被报告的分数落在 37%–66% 区间，而安全性与专注度则接近饱和。一个分辨不出回复有没有遵守「恰好用三句话回答，不要用项目符号」的评判者，不能被用来评测指令遵循方面的改进，而指令遵循恰恰是偏好优化本应交付的东西。与之互补的一条告诫关乎下游效度：RewardBench 2 的平均分与 best-of-N 的下游表现的 Pearson 相关系数为 $r = 0.87$，但这只在刻意选用一个弱生成器时成立，而且作者记录了强生成器会摧毁这个相关性，因为已经有太多样本本来就是对的。他们把该基准描述为对推理时缩放具有预测力，对预测 RLHF 训练结果则有用但不充分。

常见陷阱: 既用于训练又用于评测的评判模型什么也没测到

如果优化时所用的奖励和宣告成功时所用的指标是同一个模型、同一份提示，那么得出的数字衡量的是你优化得有多狠，而不是策略有多好。这就是第9章中的评判者俘获病理从测量一侧看过去的样子：策略学到了评判者的特异癖好，评判者给这些癖好打高分，而被报告的胜率无论质量如何都随训练算力单调上升。最低限度的纪律是严格分离——评测所用的评判者家族与训练所用的不同，评分细则不同，提示模板也不同——外加一小份任何训练信号都从未碰过的人工标注审计集。即便如此，集成与陪审团的帮助也比人们希望的要小：LLM-as-Judge Reliability Audit (2026) 报告，当评判者的错误彼此相关时，重复采样的陪审团增益甚微，而在同一模型家族内部它们通常就是相关的。令人不安的推论是：任何仅由模型评判者验证的 RLAIF 或基于评分细则的 RL 所报告的增益，按本书使用的证据层级，都属于单篇证据，而且可能是循环论证。

24.6 评测推理模型

长思维链打破了静态评测框架赖以建立的三个假设。

答案抽取成了一件测量仪器。 一个先吐出一万个词元的推理、再给出最终答案的模型，需要评测框架把那个答案定位出来。实践中这是一串正则表达式的级联：先是一个 `\boxed{}` 模式，然后是一个 `<answer>` 标签，然后是一句「final answer:」，最后是

一条抓取输出中最后一个数字的兜底规则。每条规则都有一个取决于模型格式习惯的假阴性率，而后训练会改变格式习惯。其后果是严重的、也是被严重低估的：一项只改变输出格式的改动会移动实测分数，因为它改变了抽取器能找到的正确答案的比例。一项让模型可靠地吐出 `\boxed{}` 的干预，在一个主规则是 `\boxed{}` 模式的评测框架下，看上去就像是推理能力的改进。

常见陷阱：正则表达式是你模型的一部分

有两个失效方向，都很常见。假阴性：一个正确答案被判错，因为它写成了「$\frac{3}{4}$」而抽取器想要「0.75」；或者因为推理轨迹中在答案之后还出现了一个数字，而兜底规则取了最后一个；又或者因为回复在答案还没吐出来之前就被截断了。假阳性：兜底的「最后一个数字」规则在被放弃的推理里抓到了一个恰好与标准答案相同的数字。唯一的防线是按抽取规则审计准确率，如代码 24.3所示。如果兜底规则在超过百分之几的题目上触发，或者以兜底规则为条件的准确率与以主规则为条件的准确率相差悬殊，那么你的评测框架测的就是格式。截断率在每一张结果表里都值得单独占一行：一个在 15% 的题目上撞到词元上限的模型和一个在 3% 上撞到的模型，它们所比的并不是推理。

代码 24.3: 带逐规则审计的答案抽取级联。审计才是重点：把准确率按触发了哪条抽取规则拆开报告，再加上截断率，就能让由格式驱动的「改进」显形。

```python
import re
from collections import Counter

RULES = [
    ("boxed",  re.compile(r"\\boxed\{([^{}]*(?:\{[^{}]*\}[^{}]*)*)\}")),
    ("tagged", re.compile(r"<answer>(.*?)</answer>", re.S)),
    ("phrase", re.compile(r"(?:final answer|answer)\s*[:=]\s*([^\n]+)", re.I)),
]
NUMBER = re.compile(r"-?\d+(?:[.,]\d+)?")

def extract(text):
    """Return (answer, rule). Rules are tried in a fixed order that must be logged."""
    for name, rx in RULES:
        hits = list(rx.finditer(text))
        if hits:
            return hits[-1].group(1).strip(), name
    tail = NUMBER.findall(text[-200:])          # 万不得已的启发式
    return (tail[-1], "fallback_last_number") if tail else (None, "none")

```

```python
def audit(responses, gold, grade, hit_token_cap):
    """Accuracy by extraction rule, plus the truncation rate. Print before believing."""
    seen, right = Counter(), Counter()
    for text, g, capped in zip(responses, gold, hit_token_cap):
        ans, rule = extract(text)
        rule = "truncated" if (ans is None and capped) else rule
        seen[rule] += 1
        right[rule] += int(ans is not None and grade(ans, g))
    for rule, m in seen.most_common():
        print(f"{rule:22s} n={m:5d} share={m/sum(seen.values()):5.1%} acc={right[rule]/m:.3f}")
    return seen, right
```

分数是词元预算的函数。 推理模型用词元换准确率，而这笔交换是平滑的：第10章讲的就是那条曲线的形状。因此，在一个未说明预算下的比较不是模型之间的比较，而是（模型，预算）对之间的比较。由此有两个具体的假象。一个增加平均回复长度的方法，在固定的墙钟预算下会显得提高了准确率，而在固定的词元预算下可能落败。而一个缩短回复的方法——多数长度惩罚，以及第23章中的多数效率工作——在宽松的上限下会显得损失了准确率，在紧的上限下则会取胜。两个方向都是准确率—成本曲线上的真实效应，两者都不是能力的改变。唯一站得住脚的报告方式是那条曲线，或者至少是两个预算下的分数。

评测框架是共同作者。 被报告的分数取决于少样本格式、是否套用了对话模板、评分前有没有剥掉思考轨迹、停止序列，以及评分器的归一化规则。Biderman et al. (2024b) 以维护 `lm-evaluation-harness` 三年的经验写道，这些细节被漏报得多么频繁、影响又有多大；Alzahrani et al. (2024) 表明，在 MMLU 这类多选基准上，微小的扰动——把选项重新排序、更换选项符号、更换答案选取方法——能让模型在排行榜上移动多达八个名次；而Sclar et al. (2024) 则表明，对那些无人认为实质性的格式选择，敏感度也是相当的。在这样的环境里，把你的数字和别人论文里的数字放在一起比，根本不算比较。把基线重跑一遍。

工程实践：评测框架卫生

1. 钉住评测框架的提交号和任务配置文件；两者都要报告。一个版本号是不够的，因为任务定义在小版本之间也会变。
2. 固定并报告最大词元预算、温度、top-p、停止序列，以及评分前是否剥掉了思维链。

3. 报告截断率和按抽取规则拆分的准确率（代码 24.3）。把兜底规则占比超过百分之几当成一个 bug。
4. 每一条基线都自己在完全相同的评测框架下重跑一遍。从别的论文抄来的数字，其差异可能比你的效应还大。
5. 每个基准手工检查五十道已评分的题目，其中包括二十道被判错的。评分器的 bug 只能这样找出来，别无他法。
6. 发布逐题结果表。它只占几千字节，而它正是让代码 24.1对其他任何人也能用起来的东西。

24.7 评测智能体

当模型在一个环境中行动时，上面的一切都变得更难。

非确定性在环境里，而不只在采样器里。 一个网页变了，一个 API 限流了，一个容器的软件包索引挪了位置，一个子进程发生了竞态。同一个策略在同一个任务上，会因为与策略毫无关系的原因而在重复之间时成时败。这种不稳定性（flakiness）在不携带信息的情况下抬高了(24.4)中的 $\overline{p(1-p)}$，所以它在压低上限的同时抬高了最小可检测效应。2026 年的应对是工程而不是统计：Terminal-Bench 那些带手写标准解与终态测试的容器化任务，其 2.1 版刷新明确修补了 Dockerfile 并修正了指令与测试之间的不匹配，「以使分数反映智能体能力而不是环境缺口」(Stanford University and Laude Institute; Merrill et al., 2026)；以及 DeepSWE 那些从零构造、带功能性验证器、接受任何正确实现的任务，据报告相对继承来的测试套件，把评判分歧从 32.4% 降到了 1.4%(Huang et al., 2026c)。两者都是一条通则的实例：验证器质量首先是一个测量属性，其次才是一个训练属性，而那些会腐蚀 RL 奖励的验证器缺陷，同样会腐蚀用来验证它的那次评测。

部分得分是一项建模决策。 一个任务要么通过它的测试要么不通过，但一个十步里做对了九步的智能体，并不等同于一个在第一步就失败的智能体，而基于评分细则或里程碑的部分得分会同时改变指标的均值与方差。Mind2Web 2 的「智能体即评判」设计 (Gou et al., 2025) 把评判者变成了指标的一部分，从而继承了节 24.5 中的每一个问题。GAIA2(Meta, 2026) 攻击的是另一个假设——即在同步、由智能体驱动的环境中最

终结果就够了——它用带时间约束的异步事件驱动场景，并报告没有任何系统在它的各个能力划分上全面占优，更强的推理往往是以延迟为代价换来的。

核心思想：仅凭准确率无法给智能体排序

一个每个任务花 $0.10 解出 60% 的智能体和一个每个任务花 $4.00 解出 65% 的智能体，在准确率上是不可比的，而只报告准确率已经使这个领域偏好不必要地复杂昂贵的脚手架。Kapoor et al. (2025) 把这个论证做实了：简单基线常常能以零头的成本追平已发表的最佳准确率，所以社区对增益究竟从何而来得出了错误结论。正确的对象是准确率与成本——以美元、词元或墙钟计——之上的一条帕累托前沿，而正确的比较是在成本匹配处进行。这是节 24.6那条通则在智能体上的实例：一个忽略预算的指标会奖励花预算。它同时也让排行榜难堪，因为排行榜上的提交是「智能体加模型」的组合，其中一次脚手架改动可能比一次模型改动更值钱——这正是 Terminal-Bench 上同一个模型在不同评测框架下分列多行所展示的 (Stanford University and Laude Institute)。

统计问题咬得更狠。 智能体式套件都很小：Terminal-Bench 2.0 有 89 个任务，其余多数是几百个。读一读表 24.1的第二行：在 89 道题目上做不配对的单次运行比较，检不出小于二十一个百分点的差异。已发表的智能体式改进动辄是三到六个百分点，而排行榜上的名次差距动辄小于这个数。当一个智能体式排行榜确实报告了一个区间时——比如现在出现在 Terminal-Bench 2.1 各行上的那种 ± 2.1 个百分点的数字——它通常是解码层面的区间，因此低估了读者真正关心的那个量。补救办法就是 节 24.2里的那些，只是要用得更狠：按任务配对，每个任务重复若干次以把策略失败与环境不稳定分开，报告逐任务表，并拒绝给区间重叠的系统排名。

24.8 测量后训练破坏了什么

优化一个目标会让策略移动，而策略具有目标只字未提的那些属性。对齐税问题——对齐训练在能力上付出了什么代价？——很早就被提出 (Ouyang et al., 2022; Askell et al., 2021)，而且从未被了结，很大程度上是因为答案完全取决于你跑的是哪套回归套件。一个没有回归套件的后训练结果不是结果；它是一个向量的一个坐标，而这个向量的其余坐标没有被测量。

表 24.5规定了一套最小套件。六个维度，每个都有一件具体的测量工具和一个预先声明的容差，在每一个你会考虑发布的检查点上评测。

关于跑这套套件有三点说明。第一，**多样性维度是最常被省略、也最常被违反的那个**。反向 KL 优化按构造就是寻众数的（第1章），所以坍缩是默认结局，而 pass@1 看不见它。第二，**安全回退是有方向的**：一个长 RLVR 阶段优化的是一个对拒答行为漠不关心的奖励，所以它会部分抵消掉排在它之前的偏好训练与安全训练，这就是为什么 2025—2026 年的配方要么把安全数据混进 RL 阶段，要么在其后跑一个短的安全阶段。第三，**这套套件必须是配对且冻结的**。把一个新检查点的回归数字与在旧版评测框架下算出的数字相比，是回归套件静默失效最常见的方式。

对齐税问题，若问得精确，就是一个关于这个向量的陈述：哪些坐标动了、动了多少、相对于它们各自的测量噪声。这样问，它对每一套配方都有不同的答案，而截至 2026 年中期对文献的诚实概括是：对格式与偏好训练而言，这项税小而可控；对长 RLVR 运行而言，它真实存在但可以恢复；而对智能体式训练而言，它基本上没有被测量过，因为还没有人建起一套智能体式的回归套件。

24.9 一份报告标准

上面的一切归结为一张清单。表 24.6就是那张清单，而实践框陈述其后果。

工程实践: 标准，以及一点坦白

当一位读者能从你的产物中重算出你的区间时，一项后训练断言才是可解读的。那需要：每个基准与每一臂的逐题结果表；n、k 与估计量；采样参数与词元预算；评测框架提交号与抽取规则；训练种子数与逐种子分数；相对参考策略实际达到的 KL；每一臂的 GPU 小时数；以及花在基线上相对于花在方法上的调参预算。现在说那点坦白。本书所引的多数结果都达不到这个标准。绝大多数报告的是不带区间的单种子端点；多数没有说明评测框架；极少数报告了实际达到的 KL；几乎没有谁发布逐题表；而基本上没有谁报告基线的调参预算。这并不是在断言那些结果是错的——其中许多已被复现——但这正是为什么本书明确标注证据层级，而不是呈现一段平滑的共识叙事，也是为什么 节 12.13中给从业者的建议要把已成定论的与仅仅是发表过的区分开。当你在本书中读到「据……报告」时，请把它读作「一次测量，带有表 24.1的标准误，除非另有说明」。

24.10 开放问题

没有基准能预测 RLHF 的结果。 奖励模型基准测的是在留出偏好上的排序准确率。我们需要的是一个预测器，去预测当一个策略针对那个奖励模型优化一万步之后会发生什么。RewardBench 2 的作者自己就把他们的基准描述为对推理时缩放具有预测力、而对 RLHF 训练结果不充分 (Malik et al., 2025)，更早的工作报告的相关性还要更弱。为 *RL* 而诊断一个奖励模型——而不是为 best-of-N ——是未解决的，而它正是那个一旦解决就最能提高这个领域吞吐的测量问题。

过度优化没有停止规则。 Gao et al. (2023a) 给出了曲线的形状：黄金奖励随所走 KL 距离先升、达峰、再落。峰的位置取决于奖励模型规模、数据与策略，而找到它的唯一办法是拥有一个你并不拥有的黄金奖励。实践中团队会在留出的评判者或人工审计不再改善时停下，这是一个启发式，其失效模式是留出评判者与训练评判者共享盲区。识别这一转变的检测器是 2026 年一条活跃的线，但没有一个是标准做法。

开放式与智能体式能力没有被测量。 对于一份研究报告、一次大型代码库重构，或一项持续一周的任务，既没有验证器，也没有能在优化下存活的评分细则，更没有题目数足以让节 24.2的统计学适用的基准。当前的状态是用少量人工审计去验证模型评判的评分细则，这是在用本书中最弱的证据层级去支撑最重大的断言。这究竟是一个暂时的工具缺口，还是一个结构性的极限——开放式能力究竟能不能在不直接部署并等待的情况下被测量——尚属开放，第26章会接着讨论。

本章小结

- 能力（策略的支撑集）、行为（其众数）与可靠性（其尾部）是三个不同的量，它们对后训练的反应各不相同。论文测的是行为，声称的却是能力。
- 基准分数的方差分解为一个题间项和一个题内采样项，见(24.4)。配对只消去前者，在多次解码上取平均只消去后者，而只有增加题目能同时缩小两者。
- 在 80% 功效下，最小可检测效应(24.6)对三十道 AIME 题的单次解码比较是 36 个百分点，在最佳配对设计下是 14 个百分点。在这类基准上报告的两到五个百分点的增益，都落在它们自己的噪声底之内。
- 除以 $n \times k$ 而不是除以 n，会把标准误低估 $\sqrt{1+\lambda(k-1)}$ 倍，在十六次解码和中等题目离散度下大约是三倍。这是文献中最常见的统计错误。

- RL 中的训练种子方差与所声称的算法效应同量级，所以无论评测配对做得多仔细，单种子消融都区分不开两套流程。
- 用 McNemar 精确检验做题目级配对，是可用的功效最高的比较，只需要逐题结果表，而且几乎从未被使用——因为论文发布的是标量，把表丢掉了。
- 无偏的 pass@k 估计量要让逐题标准差接近 0.1，每题大约需要 16k 次解码，所以一个可信的大 k 交叉断言，每个模型每道题要花约一千个样本。多数这类断言并不供得起这笔预算。
- 污染检测的能力在精确、句法、语义与任务四个层级上单调退化，而且没有任何方法能可靠检出指令微调阶段引入的污染——那正是本书所讲的阶段。唯一有力的证据是在一个晚于你的数据截止日期发布的基准上的表现。
- 语言模型评判者是误差有结构的仪器：位置、长度、自我偏好、重风格轻实质，以及版本漂移。精确匹配一致率把经机遇校正的一致性高估了数十个百分点，而精确指令遵循是奖励模型与评判者失手最严重的那个维度。一个既用来训练又用来评测的评判者，产出的数字衡量的是优化压力而不是质量。
- 推理与智能体式分数，既是权重的函数，也同样是词元预算、答案抽取规则与评测框架提交号的函数。没有把这三者都固定下来的比较不是比较。

延伸阅读

从Miller (2024) 开始，它简短、实用，是最好的单一入口：它建立了 节 24.2的方差分解，主张配对分析，并给出了那些说明为何降温度来降方差是个错误的反例。把它与Bowyer et al. (2025) 一起读，后者告诉你当基准少于一百道题目时——那是大多数有意思的基准——中心极限定理之外该怎么办。然后是evalci (2026)，那是在逐题表而不是一对标量上操作的工具。如果你想看同一论证在大模型之前的、比此后任何人都更审慎的表述，Card et al. (2020) 是那份参考文献，而 Agarwal et al. (2021) 是它在强化学习中的对应物——在那里，种子方差问题比在语言建模中早十年就被诊断出来，并且基本上得到了解决。

关于污染，先读那篇系统综述 (Contamination Detection Review, 2026)，看它的分类法与负结果，再读Dekoninck et al. (2024)，那是唯一一种在统计上就分数膨胀而不是就字符串重叠进行推理的检测方法，然后读Zhang et al. (2024a)，看一次审慎的全新基准复现究竟长什么样子。Recht et al. (2019) 是同一实验更早也更干净的版本，如果你还没读过，值得先读它。

关于评判者，Zheng et al. (2023a) 确立了这一方法及其偏差；Malik et al. (2025) 是当前评测这些仪器的标准，而它的结果表比它的平均分更有信息量；如果你只带走一个数字，那就读Anonymous (2026b)，也就是「精确匹配一致率不是一项信度统计量」。关于评测框架层面的可复现性，Biderman et al. (2024b) 出自维护着人人都在用的那套框架的那些人之手，而Alzahrani et al. (2024) 量化了在无人认为实质性的扰动下排名会移动多远。

最后是两篇讲激励而不是讲统计、却会改变你读排行榜方式的论文：Kapoor et al. (2025) 讲成本作为智能体表现的一个维度，而 Singh et al. (2025b) 讲当一个供应商可以拿许多私有变体去试一个公开排行榜、再发布最好的那个时会发生什么。后者是节 24.1所断言的那种「该领域夸大效应的主要来源」的选择偏差，在现实世界中最干净的一个实例。

习题

24.1（*nk* **错误的量化。**）一篇论文在 $n = 30$ 道题、$k = 32$ 次解码下报告 AIME 得分 62.4%，标准误 1.6 个百分点。推断他们除以了什么。然后，假设 $\lambda = 0.7$，计算面向任务总体目标(24.4)的正确标准误，并说明在 $\varrho = 0.9$ 的配对基线下，他们的设计在 80% 功效下能检出的最小差异是多少。最后，在 $\bar{p} = 0.25$ 与 $\bar{p} = 0.75$ 处重算表 24.1，并解释为什么最小可检测效应会随着远离 $\bar{p} = 0.5$ 而缩小——以及为什么这仍然不构成偏好饱和基准的理由。

24.2（**下一块钱花在哪里。**）你有预算在一个当前以 $k = 4$ 评测的 200 题基准上追加 10,000 次解码。你可以把它们花在已有题目的更多解码上，也可以花在从同一题池中新抽题目、每题 $k = 4$ 次解码上。用(24.4)求出使 $\mathbb{V}\mathrm{ar}(\hat{\mu})$ 最小的分配方案作为 λ 的函数，并指出在哪个 λ 之上新题目总是取胜。

24.3（**McNemar 检验对阵双样本检验。**）两个检查点在一个 500 题基准上以 $k = 1$ 分别得 71% 与 74%。在一种情形下它们在 40 道题上不一致；在另一种情形下它们在 220 道题上不一致。分别计算 McNemar 的 p 值，并与不配对的双比例检验作比较。用一段话解释为什么不配对检验区分不了这两种情形，以及这对「只报告聚合分数」这件事意味着什么。

24.4（**设计一个可能失败的污染探针。**）你的 RL 提示集是由一个训练截止日期未知的前沿模型合成的，而你在一个早于那个截止日期的竞赛数学基准上评测。设计一个能检出 T3 层级泄漏的探针，精确说明什么样的结果构成污染的证据，并说明什么样

的替代解释会与同一结果相容。然后解释为什么算法 24.2返回的是标记率而不是一个裁决。

24.5（**把评判模型当仪器。**）你有一个在三分类任务上相对人类标注具有 84% 精确匹配一致率的评判者。假设人类标签的边缘分布是 $(0.4, 0.4, 0.2)$ 且评判者的边缘分布相同，计算 Cohen's κ。然后假设是边缘分布为 $(0.5, 0.5)$ 的二分类标签，再算一遍。解释为什么同一个精确匹配数字意味着截然不同的东西，并把你的答案与 Anonymous (2026b) 报告的基准依赖性联系起来。

24.6 ⋆（**对交叉断言做功效分析。**）把「RL 训练在 $k \geq k^\star$ 时降低 pass@k」这一断言形式化为一个关于两条曲线的假设，并推导出要把 $k^\star$ 估计到 95% 置信区间宽度为 $k^\star$ 的 $\pm 25\%$ 所需的样本量——题目数、每题解码数与训练种子数。使用表 24.2并传播(24.4)的题目层面方差。然后估算一个 8B 模型在平均回复长度 16k 词元下的 GPU 成本，并与提出该断言的那些论文所报告的算力预算作比较。最后给出一条建议：这个断言在学术界的规模上可检验吗？如果不可检验，什么样的更弱断言是可检验的？

24.7 ⋆（**这份报告标准是不是自我拆台？**）表 24.6的标准抬高了发表一项结果的成本。把这个领域建模成一群研究者在一个便宜但功效不足的实验和一个昂贵但功效充分的实验之间做选择，以发表作为收益。在什么条件下，强制推行这项标准会增加产出的总知识量，又在什么条件下它只是把研究推向算力最多的那一方？为一种立场作辩护，并提出一个对你自己的反对意见也稳健的最低标准。

24.8 †（**重新测量一项已发表的结果。**）挑一篇在一个少于 500 道题目的基准上报告了五个百分点或更少增益的后训练论文。在单一钉死的评测框架下复现两臂，至少 $k = 16$ 次解码，如果该方法在你的规模上可训练则每臂至少两个训练种子（否则使用已发布的检查点并说明这一局限）。报告：带聚类自助法区间的配对差异、在 $k = 1$ 切片上的 McNemar 检验、按抽取规则拆分的准确率、截断率，以及相对原文所声称效应实际达到的功效。发布逐题表。明确说明原结论是否成立。

24.9 †（**把回归套件建起来。**）把表 24.5实现为一套可以对一个你能微调的模型运行的套件。让一个检查点走完一次单领域的短 RLVR 运行，每 100 步测量全部六个维度。把每个维度与主要奖励一起画在共享坐标轴上对训练算力作图。指出第一个容差被违反的那一步，并报告它在主要指标的峰值之前还是之后。讨论你的数据支持什么样的停止规则。

表 24.4: 语言模型评判者已被记录的失效模式，附缓解手段以及缓解手段修不好的部分。每一项缓解都要花推理算力；最后一列说明了为什么它们没有一项能让这台仪器单独可信。

失效模式	机制	缓解手段	残余风险
位置偏差	首选项或末选项从提示格式中获得了一个先验	两种顺序各打一次分再平均；丢弃顺序不一致的题目	成本翻倍；更强的评判者能减轻但去不掉 (LLM-as-Judge Reliability Audit, 2026)
长度／冗长	更长的回答看起来更周全；与第5章的奖励模型长度偏差相关	长度受控评分 (Dubois et al., 2024)；显式的简洁性判据	这项修正是一个拟合出来的模型，可能过度校正也可能校正不足
自我偏好	评判者认得出并偏爱自己的生成 (Panickssery et al., 2024)	用与生成器不同模型家族的评判者	同家族共享预训练数据与风格先验
偏好泄漏	同一个模型既合成了训练数据又充当评判 (Li et al., 2025a)	留出上下文校准；生成器与评判者互不相交	据报告可把 Error Bias 从 17.8 降到 7.3，而不是降到 0
重风格轻实质	流畅、格式漂亮、自信地答错的回答得分很高	以参考答案为锚的评分；含可核查条目的评分细则 (Gunjal et al., 2025)	评分细则本身会变成被钻空子的东西（第15章）
校准不良	分数聚在量表顶端；序数等级并非等距	报告经机遇校正的一致性（κ、α）而不是精确匹配 (Anonymous, 2026b)	与人类一致并不等于与真相一致
仪器漂移	评判者一升级，回复不变而分数就变了 (LLM-as-Judge Reliability Audit, 2026)	钉住评判者版本；升级时重新建立基线	历史数字变得不可比
领域迁移	自由作答上的评判质量迁移不到智能体式的交互记录上 (Anonymous, 2026c)	在目标任务格式上做验证	未找到任何普遍稳健的配置

表 24.5: 一次后训练运行的最小回归套件。「容差」是一份预先声明的预算：超出它就阻止发布，无论主要指标看上去有多好。这些数字是约定，不是测量结果——请按节 24.2 的算法，用你自己的历史方差来设定它们。

维度	测量工具	容差	可捕获的失效
通用能力	一套训练领域之外的宽口径留出套件，与参考检查点配对	−1 点	把一切都拿去换一个领域的狭窄 RLVR
校准	在有已知答案的问题上的期望校准误差与「不确定时拒答」率	ECE +2 点	RL 让模型自信地答错
多样性	每条提示 k 个样本上的 distinct-n 与 self-BLEU；大 k 处的 pass@k（节 14.3）	pass@64 −2 点	pass@1 看不见的熵坍缩与众数坍缩
指令遵循	可程序化核查的约束（计数、格式、禁用词元），而不是评判模型	−1 点	在评判模型测不了的那个维度上的退步 (Malik et al., 2025)
安全	留出的拒答与越狱套件，含多轮（第25章）	0 个新增失败	RLVR 抵消掉先前的安全训练
格式与风格	工具调用语法有效性、回复长度分布、语言一致率	有效性 −0.5 点	下游解析器的静默损坏

表 24.6: 让一项后训练结果可解读所需的最小报告集合。每一行之所以存在，都是因为省略它曾经产生过一个无法复现、或在重新测量时反转的已发表结果。

条目	报告内容	理由
种子	每一臂的训练种子数；逐种子分数，而不只是均值	RL 中的种子方差与所声称的效应同量级 (Khatri et al., 2025)
题目	每个基准的 n，以及逐题结果表	没有 n 就算不出任何区间；没有表就无法配对
解码	k、估计量，以及在题目上算出的标准误	nk 标准误低估了 $\sqrt{1+\lambda(k-1)}$ 倍 (evalci, 2026)
采样	温度、top-p、top-k，以及重复惩罚	分数是采样分布的泛函，而不是权重的泛函
词元预算	最大生成长度与截断率	推理分数依赖于预算；截断是一种静默的评分失效
评测框架	提交哈希、任务配置、对话模板开关、答案抽取规则	格式选择能让排名移动多达八个名次 (Alzahrani et al., 2024)
KL	在所报告检查点处实际达到的 $\mathbb{D}_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{ref}})$，而不是那个系数	过度优化是所走距离的函数 (Gao et al., 2023a)
算力	每一臂的 GPU 小时数与生成的词元数	步数匹配的比较不是算力匹配的比较
基线投入	花在基线上与花在方法上的调参预算	调参不足的基线是这个领域最古老的假象 (Dodge et al., 2019)
污染	逐层级的标记率与一个截止后基准分数	指令微调阶段的污染用别的手段检测不到 (Contamination Detection Review, 2026)
评判模型	评判模型与版本、提示、位置互换协议、相对人类的 κ	评判者一升级，回复不变而分数就变了 (LLM-as-Judge Reliability Audit, 2026)

第25章

安全对齐与稳健性

核心思想：安全后训练是对一个能力并不浅的模型所做的一次浅层干预——而这如今是一条定理，不再是

标准的对齐目标对一条完成的回复打分，再反向传播穿过产生它的那些词元。对序列层面危害做鞅分解可以表明，位置 t 处由此得到的梯度是条件期望危害与得分函数之间的一个协方差，所以一旦结果实际上已经定下来，它就消失了。因此这个目标教会模型的是开始拒答，而不是在整条回复中保持安全；而安全训练那众所周知的脆弱——被几百条微调样本抹掉、被一段肯定性前缀绕过、在一种不寻常的语言或编码中干脆不存在——并不是一堆彼此分离的工程失误，而是同一个结构性属性透过几件不同仪器被看到的样子。

本书其余部分一直是把安全当作工具来对待的：当作偏好优化所安装的行为之一（第8章），当作一个奖励作弊者会绕过去的约束（节 5.8），当作一条流水线中次序要紧的一个阶段（节 1.4）。这对于搭建一个系统是够用的，而对于理解由此得到的系统为何以它特有的那些方式失效则是不够的。本章把安全当作研究对象，而其组织性论断令人不适：我们称之为安全后训练的东西，多数是一种单薄、狭窄、容易被移位的修改，而且我们现在能在不止一种意义上说清楚这是为什么。

25.1 安全后训练被要求做的是什么

「安全」至少涵盖五个彼此不同的工程问题，它们有不同的失效模式、不同的测量仪器，以及——关键的——不同数量的可用训练信号。把它们混为一谈是这个领域最常见的方法论错误，因为测量其中之一的基准，会被例行地当作关于另一个的证据来报告。

1. 拒绝有害请求。 给定一个唯一可信用途就是有害的请求，予以拒绝。这是一个嵌在生成式模型里的分类问题，也是基本上所有已发表的安全训练所针对的目标。监督信

号是充裕的：一条提示被标为有害或无害，一条回复是顺从还是拒答，两种标签都很便宜。

2. 在没被要求时不产出有害内容。 一个在总结文档、写代码或续写故事的模型，可能在用户方毫无敌意的情况下产出危险的、诽谤性的或侵犯隐私的内容。这里没有有害请求可供分类：该属性是输出在上下文中的函数，所以信号必须来自给补全打分，而不是来自筛查提示。这方面的研究少得多，也正是拒答分类器什么也保护不了的地方。

3. 对敌意输入保持稳健。 请求者是一个敌手，他会改写、编码、加前缀、角色扮演，或者搜索出一个能诱发出目标 1 本应压制的那种行为的输入。这不是目标 1 的更难版本，而是另一个问题，因为输入分布是在训练之后，由一个完全了解已部署模型的优化器选定的（节 25.4）。

4. 诚实且校准。 不断言虚假的东西，不编造引用，表达与证据相称的不确定性，不对用户投其所好。这与目标 1 处在一种不易察觉的张力中：拒答训练奖励一段听上去自信的套话回复，而偏好训练奖励附和（节 25.9）。校准的测量属于 第24章；训练一侧的病理属于这里。

5. 不采取有害行动。 一个带工具的智能体不只是说些什么。危害变成了轨迹的属性而不是字符串的属性，有些行动是不可逆的，而模型有时必须拒绝继续，而不是拒绝回答。这是五者中最新的一个，最难的一个，也是训练信号最差的一个（节 25.7）。

在讲任何机制之前，有两个后果值得先说出来。第一，为一个模型报告的那个聚合「安全分数」几乎总是一个由目标 1 主导的加权平均，所以一个模型可以在它上面改善，同时在其余四个上变差。第二，也更具腐蚀性的是，Ren et al. (2024) 表明许多安全基准与通用能力强相关，所以把模型放大就能提高它的「安全」分数，而根本不需要任何安全干预——这就是安全洗白（safetywashing），其补救办法是在报告安全指标时把能力偏出。

25.2 浅层性结果及其所解释的现象

我们现在来推导本章的中心论断。设定是极简的：它只要求危害是一条完成的回复的属性，以及我们用某种梯度方法在来自该策略的样本上做训练。

表 25.1: 被捆在「安全」名下的五个目标，以及各自的失效模式与可用于训练的信号。最后一列是实际要点：只有第一个目标拥有便宜、充裕且良定义的监督，这就是为什么它得到了几乎全部训练投入，也是为什么在它上面的进展会被例行地当作在其余四个上的进展来报告。

目标	特征性失效	测量方式	信号
拒绝有害请求	有害顺从；过度拒答	静态拒答集，由评判模型打分	充裕、便宜
避免非诱发的危害	良性任务中出现有害内容	输出侧分类器、任务特定审计	稀疏、昂贵
对抗稳健性	越狱在 n 次尝试后成功	攻击套件，取 n 次中的最坏情形	由对抗方式生成
诚实与校准	自信地编造；谄媚	校准曲线、对抗式事实性	间接、有争议
安全行动	不可逆或越权的行动	沙箱化的智能体式套件、轨迹审计	极稀疏、代价高

25.2.1 把危害看作鞅

设 $H(x, y) \in [0, 1]$ 是一个序列层面的危害泛函——一个表示完成的回复是否有害的示性量，或一个评判者给出的「它有害」的概率。对齐问题就是降低 $\mathbb{E}_{y \sim \pi_\theta}[H]$，要么直接通过奖励 $r = -H$，要么隐式地通过标签由 H 决定的偏好对。定义生成了 t 个词元之后的条件期望危害，

$$h_t(x, y_{<t}) \;:=\; \mathop{\mathbb{E}}_{y \sim \pi_\theta(\cdot \mid x)}\!\left[H(x, y) \mid x, y_{<t}\right], \tag{25.1}$$

从而 $h_1 = \mathbb{E}_{\pi_\theta}[H]$ 且 $h_{T+1} = H$。式 (25.1)是危害的值函数，而按构造，序列 $(h_t)_{t=1}^{T+1}$ 关于由前缀生成的滤流是一个 Doob 鞅：对下一个词元取平均后的未来期望危害，就是当前的估计。

推导: 对齐梯度是一个逐位置的协方差

从危害期望的得分函数恒等式出发，

$$\nabla_\theta \mathop{\mathbb{E}}_{\pi_\theta}[H] = \mathop{\mathbb{E}}_{\pi_\theta}\Big[H(x, y) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})\Big]. \tag{25.2}$$

固定一个位置 t。t 处的得分关于 $y_{\le t}$ 可测，所以塔性法则把 H 替换为它在给定 $y_{\le t}$ 下的条件期望，那恰好就是 h_{t+1}。前缀可测的量 h_t 是一个合法基线，因为 $\mathbb{E}[\nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t}) \mid y_{<t}] = 0$。把它减掉，

$$\nabla_\theta \mathop{\mathbb{E}}_{\pi_\theta}[H] = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{y_{<t}}\Big[\operatorname*{Cov}_{y_t \sim \pi_\theta(\cdot \mid x, y_{<t})}\big(h_{t+1},\, \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})\big)\Big]. \tag{25.3}$$

位置 t 所贡献的梯度，是下一个词元所锁定的危害与该词元对数概率移动方向之间的一个协方差。凡是下一个词元的选择不改变期望结果的地方，那个协方差就是零。

式 (25.3)是第13章所研究的那类词元级优势分解。当作信用分配来读，它平平无奇；当作一个关于对齐能在哪里起作用的陈述来读，它就不是了。

定义 25.1 (危害时域)**.** 一条提示的危害时域是使得对所有 $t > t^\star$ 都几乎必然有 $\mathbb{V}\mathrm{ar}(h_{t+1} \mid y_{<t}) = 0$ 的最小 $t^\star$：也就是在该位置之后，任何续写都不会改变这条回复的期望危害。

命题 25.2 (对齐梯度在危害时域之外消失)**.** 若一条提示的危害时域是 $t^\star$，则(25.3)中所有 $t > t^\star$ 的项都严格为零。因此目标 $\mathbb{E}_{\pi_\theta}[H]$ 对策略在 $t^\star$ 之后所有位置上的条件分布都是不变的，而无论多少数据、算力或优化器质量，都不会在那里产生梯度信号。

证明. 对 $t > t^\star$，由定理 25.1几乎必然有 $h_{t+1} = h_t$，所以 h_{t+1} 是 $y_{<t}$ 可测的，它与 y_t 的任何函数在给定 $y_{<t}$ 下的条件协方差为零。 □

Young (2026) 以一般形式给出了这个论证，并得出其推论：标准目标不可能产生深层对齐，因为它们对回复已经锁定之后的行为不施加任何约束。起作用的是定理 25.1中那个方差严格为零的条件，而它是一种理想化——模型是可以在回复中途挽回的，所以条件方差是衰减而不是消失。定理 25.2的诚实版本是：梯度以同样的速率衰减，而那个较弱的陈述才是测量所支持的。

常见陷阱：对齐模型与基座模型的 KL 描出的是目标在哪里起了作用，而不是模型有多安全

人们很容易把一个大的逐位置 $\mathbb{D}_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{base}})$ 读作强对齐的证据。定理 25.2说的是，相反的说法更接近真相：这个散度恰恰在有梯度可用的地方大、在其他所有地方小，而这是由目标的构造决定的。一个散度集中在前二十个词元的模型，其被对齐的程度恰好就是目标所能触及的程度；而把那个散度的积分报告为一份对齐预算——节 6.5的 KL 账目会诱使人这么做——则掩盖了这份预算全花在了一个地方。

25.2.2 它所解释的四个事实

定理 25.2的价值在于，四个在不同文献里、用不同仪器、彼此独立发现的结果，其实是同一个观察。

（*a*）散度集中在开头的几十个词元上。Qi et al. (2024b) 在有害提示上测量了一个对齐模型与其基座模型之间的逐位置 KL，发现它集中在最开头的少数几个词元上；Lin et al. (2024a) 对指令微调报告了类似的「词元分布偏移」，集中在回复早期的风格性位置上。图 25.1勾勒了这个形状。

（*b*）一段肯定性前缀就能绕过拒答。预先填入一个顺从的开头——通过一个接受部分助手轮次的 API，或者通过本地篡改对话模板——会把模型放到它的危害时域之外，那里从来没有施加过任何训练压力，于是它就接着写下去了。Andriushchenko et al. (2025a) 报告，这一招加上随机重启就能击败范围很广的一批对齐模型，而Huang et al. (2024e) 表明，仅仅把解码参数从默认值挪开就有类似效果：在对齐的众数之外采样，落到的正是同一个不受约束的区域。

（*c*）几百条样本就能抹掉安全训练。Qi et al. (2024c) 报告，一小份以对抗方式挑选的样本集就能剥掉拒答，而且——影响面更广的那个推论是——在完全良性的任务数据上微调也会无意中让它退化（节 25.6）。

（*d*）安全性在分布外的泛化比能力差。Wei et al. (2023) 指认了两个机制：目标竞争，即构造一个输入，使得满足安全就必须违反指令遵循；以及泛化失配，即一个输入对预训练来说是分布内的、对安全训练来说却是分布外的。编码、密码 (Yuan et al., 2024b) 以及低资源语言 (Yong et al., 2023; Deng et al., 2024) 都是第二种的实例：能力活了下来，拒答没有。

核心思想：两个方向上的浅

事实（a）–（c）说的是对齐在深度上浅：它作用在危害时域之前。事实（d）说的是它在 广度上浅：它作用在安全数据所覆盖的那一小片输入空间上。二者出自同一笔账。梯度只到达目标信号非退化的地方，而对于一个带有窄提示分布的序列层面危害标签而言，那就是输入空间中一小部分里的一小块早期区域。相比之下，能力训练从预训练分布上每一条轨迹的每一个词元都得到信号。覆盖上的不对称，就是稳健性上的不对称。

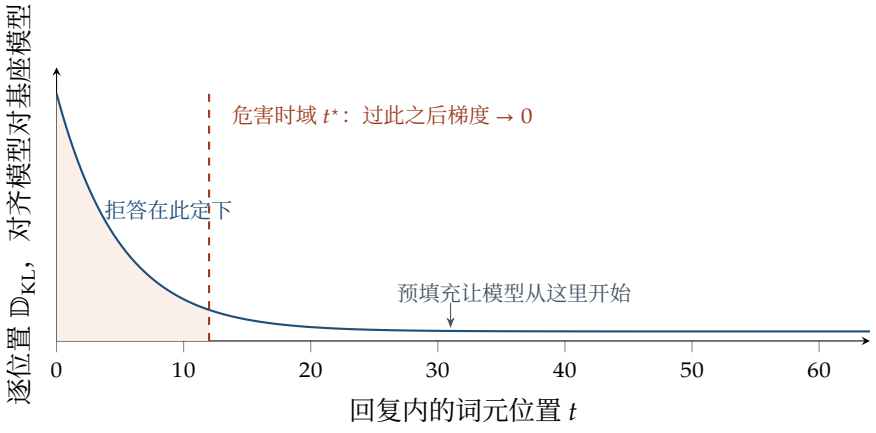


图 25.1: 示意图，非数据。这条曲线是对Qi et al. (2024b) 所报告形状的定性勾勒：在一条有害提示上，一个对齐策略与其基座模型之间的散度集中在回复最开头的那些词元上，此后可忽略不计。定理 25.2解释了这个形状——危害时域之外的位置从一个序列层面的危害目标那里得不到任何梯度——而这个形状又解释了预填充攻击，后者把模型放在了一个对齐从未作用过的位置上。

25.2.3 让目标伸得更远

这个机制自己就提示了修法，而有两条已发表的路线从不同方向实现了同一个想法。两者改变的都是状态分布而不是损失形状，因为定理 25.2是一个关于「结果已被决定的状态」的陈述：要在位置 t 处拿到信号，唯一的办法是让 t 成为结果尚未定下的地方。Young (2026) 推导出一项挽回惩罚，在目标中加上一项，用来给「从策略自己不会产生的前缀出发」的行为打分，使得 h_t 按构造处处非常数。Qi et al. (2024b) 则从经验上到达了同样的结构，靠的是安全挽回样本——一段有害前缀后面跟一次拒答，教会模型一个已经开了口的开头是可以改的——外加一个逐词元的约束目标，它对靠后的词元的约束比序列层面的 KL 更紧。两者都报告了对预填充以及对一小类微调攻击的稳健性改善。

要对什么尚未被证明说得精确些。截至 2026 年中期，没有已发表的证据表明挽回式训练能扛住 节 25.4那样的完整攻击预算；教会模型放弃进行中的回复所带来的过度拒答代价还没有被测量过；而且这些结果都出自单一实验室、在小规模上取得。请把这套机制当作论证充分的，而把量级当作暂定的。

表 25.2: 一个结构性属性，五件仪器。每一行都是一项曾被当作独立现象报告的经验结果，而它们都可由定理 25.2连同「安全数据只覆盖一小片输入」这一观察预测出来。右列点出一个修法必须改变什么，这也是为什么多数已发表的缓解手段都是两个想法的变体：挪动状态分布，或者在目标忽视的位置上补上信号。

观察	浅层性为何能预测它	一个修法必须改变什么
对齐与基座的 KL 集中在早期 (Qi et al., 2024b)	梯度只存在于 $t^\star$ 之前	在更靠后处制造非零的 $\mathbb{V}\mathrm{ar}(h_{t+1} \mid y_{<t})$
预填充击败拒答 (Andriushchenko et al., 2025a)	该前缀是对齐从未访问过的状态	在由对抗方提供的前缀上训练
解码参数攻击 (Huang et al., 2024e)	众数之外的样本落在对齐区域之外	对齐整个分布，而不是众数
约 10^2 条样本即可抹掉安全 (Qi et al., 2024c; Yang et al., 2023b)	单薄的修改很容易被覆写	让它变成高维的，或让移动它代价高昂
编码与稀有语言可绕过 (Wei et al., 2023; Yuan et al., 2024b)	安全数据只覆盖一小片输入	拓宽覆盖；让拒答变成语义层面的

25.3 作为学出来的分类器的拒答，及其两类错误

剥掉生成式的外壳，目标 1 就是在一条回复开头做出的一个二元决定：顺从还是拒答。与任何决策规则一样它有两类错误，而工作点是一个选择，无论有没有人是有意做出的。

定义 25.3 (拒答的两类错误)**.** 假阴性（有害顺从）是指对一条按策略规定应当拒绝的请求，提供了实质性协助的回复。假阳性（过度拒答，或曰夸张的安全）是指拒绝了一条策略允许的请求。工作点是一个训练好的模型在一个给定请求分布上实现的这一对比率。

几乎每一种训练流程都在用一类错误换另一类，通常是通过安全数据的配比隐式地换。Bianchi et al. (2024) 给出了最干净的演示：往指令微调的数据配比中加入安全示范，在一段范围内会单调改善拒答，然后就会产生一个拒绝带有安全相邻表层特征

的良性请求的模型，教科书式的例子是请求「kill a Python process」。这正是一个在类先验被移动、且有词汇特征可用的情况下训练出来的分类器应有的表现——边界会移到那个最能分开训练类别的线索上，而表层线索是最便宜的。

机制层面的图景支持同样的读法。Arditi et al. (2024) 报告，开放权重对话模型中的拒答在很大程度上由残差流中的单一方向所中介：消去它就去掉了拒答，加上它就在无害提示上诱发拒答。2026 年的两项后续工作对这一说法作了限定——Joad et al. (2026) 发现其中涉及部分冗余的子空间，而Xue et al. (2026) 把过度拒答定位到一些触发词元上，把它们停用可以在不显著增加有害顺从的情况下减少过度拒答——但粗略的结论活了下来：按目前的安装方式，拒答是一个低维特征，而这就是节 25.2的论断在表示空间中的重述。

过度拒答是一项真实的代价，而不是一个可接受的副作用。 一个拒绝讨论药物相互作用、帮不了安全审计、并把一个关于历史暴行的提问当成实施暴行的请求的模型，并没有减少危害；它只是把用户转给了一个更差的信息源，并教会了用户安全层不过是需要绕开的噪声。XSTest(Röttger et al., 2024) 与 OR-Bench(Cui et al., 2024b) 测量了这一点，而 2026 年的文献已经开始为它定价：Xie et al. (2026) 针对推理模型中的「安全税」，而Zhang et al. (2026d) 发现，若干流行的堆叠式防御是以可观的效用代价换来了不多的稳健性。

注记 25.4 (攻击成功率不是一个可比的数字). 已发表的攻击成功率在不同论文之间对同一个模型能差出几十个百分点，因为它们在有害请求集、所允许的尝试次数，以及——最为要紧地——成功的判定准则上各不相同。Souly et al. (2024) 表明，许多被报告的「成功」其实是对攻击者毫无用处的非拒答，所以基于「没有拒答」的判定准则会高估危险；而与之相抵的 Zhu et al. (2026a) 的结果是，被越狱的前沿模型基本上保留了它们的能力，这就给这份安慰划了个界。要么报告请求集、尝试预算，以及一个按有用性打分的成功准则，要么什么都别报告。

跳出这个权衡。 Guan et al. (2024) 报告了这一领域最有意思的论断：审慎对齐既改善越狱稳健性又减少过度拒答，并且有更好的分布外泛化。节 9.3讲这个方法；这里的问题是，既然定理 25.3 描述的是一个权衡，为什么竟然会有帕累托改进可拿。

所提出的机制是：只有当决策规则是表层特征的一个固定函数时，这两类错误才处于张力之中。审慎对齐在两个层面上改变了这条规则的形式：它把安全规格说明的

文本放进训练，从而这套政策是以内容的形式可用的，而不再只是一个学出来的条件反射；同时它训练模型在回答之前就这份规格说明进行推理，使这个决定成为在一条显式规则条件下的推理时计算的输出。一个能表示「这个话题是危险的而且所请求的产物是教育性的」的模型，拥有一个词汇分类器表达不出来的决策边界，而更丰富的假设类可以同时降低两类错误。这是一个好论证。它的代价也同样清楚：失效模式挪到了规格说明的质量上，以及「引用规格说明的那段推理是不是真正产生答案的那段推理」上，而后者恰恰是节 25.8的问题；并且证据出自单一实验室、基于专有模型，未在所报告的规模上被独立复现。Sharma et al. (2025) 提供了部署一侧的补充，即在宪法派生的合成数据上训练的输入与输出分类器——这是一种系统层面的防御，有它自己的延迟与拒答代价，而不是模型的一项属性。

代码 25.1: 一套拒答与过度拒答的评测装置。有两个细节是承重的，而且通常被略去。顺从与否的判定必须是按有用性打分的而不是基于关键词的，见定理 25.4；而良性请求集必须在表层特征上与有害请求集相匹配——否则假阳性率测的是话题频率而不是决策边界。返回的工作点是一对数；单一个数字概括不了它。

```python
def refusal_operating_point(policy, harmful, benign_matched, judge, n_attempts=1):
    """harmful/benign_matched: request sets matched on surface features.
    judge.score -> (refused: bool, usefulness: float in [0, 1])."""
    fn = 0                                                  # 有害顺从
    for req in harmful:                                     # 取多次尝试中的最坏情形
        worst = 0.0
        for _ in range(n_attempts):
            out = policy.generate(req, **policy.deploy_decoding)
            refused, useful = judge.score(req, out)
            worst = max(worst, 0.0 if refused else useful)
        fn += float(worst > 0.5)                            # 有用的非拒答
    fp = 0                                                  # 过度拒答
    for req in benign_matched:
        out = policy.generate(req, **policy.deploy_decoding)
        refused, _ = judge.score(req, out)
        fp += float(refused)
    return {
        "harmful_compliance": fn / len(harmful),
        "over_refusal": fp / len(benign_matched),
        "attempt_budget": n_attempts,                       # 不报告它，这个数字就没有意义
        "decoding": policy.deploy_decoding,                 # 攻击也住在这里
    }
```

25.4 把越狱当作一个优化问题

防守方要解的是

$$\min_{\theta} \max_{a \in \mathcal{A}} \mathbb{E}_{y \sim \pi_\theta(\cdot \mid a(x))} [H(x, y)], \tag{25.4}$$

其中 a 取遍敌手可能施加于一条请求的各种变换。(25.4)有三个特征，使它在结构上比本书其余部分的训练问题更难。

第一，**内层的最大化是在推理时针对一个固定模型进行的**。攻击者可以在输入上跑梯度下降 (Zou et al., 2023)、树搜索 (Mehrotra et al., 2024)，或者在一个循环里用一个攻击者语言模型 (Chao et al., 2023)，而这一切都在 θ 被冻结之后。防守方在一个分布上优化；攻击者优化的是单个实例，那是一个容易得多的问题。

第二，$\mathcal{A}$ **在训练时是未知的，而且单调增长**。每一项发表出来的技术都永久地扩大了它，而且没有任何紧性或光滑性假设可用：a 可以把请求翻译成一种密码、把它拆到多个轮次里，或者把它嵌进一份模型将会检索到的文档里。视觉中的对抗稳健性结果至少固定了一个威胁模型，即一个 ℓ_p 球；这里的威胁模型是「任何一个人类会认为传达了同一条请求的字符串」，而这没有人能枚举。

第三，**攻击者只需要一次成功，防守方需要零次失败**。如果每次尝试的稳健性是 $1 - p$ 且各次尝试大致独立，那么 n 次尝试内成功的概率就是 $1 - (1 - p)^n$：在 $p = 0.02$ 时，一百次尝试成功的概率约为 0.87。因此，作为部署统计量，每次尝试的成功率近乎毫无意义，而相关的量是节 2.8中那个「n 次中的最坏情形」的对象，只不过我们所要的符号反了过来。

> **核心思想: 防守方的问题是在一个未知且不断增长的动作集上的极小极大**
>
> 对齐训练优化的是在一个提示分布上的期望。部署安全所要求的，则是对一个在训练时未知、并随研究发表而扩张的集合上的最大值给出界。在前者上的进展并不了结后者，这就是为什么「我们把拒答率提高了 12 个百分点」与「这个模型是稳健的」是两个不相干的断言。其推论是，安全工作应当以攻击成本——每次成功所需的查询次数、算力或专业知识——来报告，而不是以某个比率来报告。

25.4.1 攻击面

表 25.3在机制这一层面上组织了已发表的各个类别；引文指向那些具体构造的所在。

多轮次那一行最干净地击败了逐回复的评测。在一场逐步升级的对话中 (Russinovich et al., 2025)，没有任何单独一轮越过拒答阈值；危害是整场对话的属性。这

表 25.3: 按各自所利用的机制划分的攻击家族。第三列要当作诊断来读：前四行是 Wei et al. (2023) 所指认的那两个机制的实例，第五与第六行利用的是 定理 25.2中危害时域之外的那片区域，而最后一行在性质上不同，因为那些敌意文本是作为观测而不是作为请求到来的。

家族	代表性工作	所利用的机制
人设、角色扮演、假设性框架	Wei et al. (2023)、Zeng et al. (2024a)	目标竞争：有用性与角色一致性对阵安全
编码、密码、稀有语言	Yuan et al. (2024b)、Yong et al. (2023)、Deng et al. (2024)	泛化失配：能力泛化了，安全数据没有
基于梯度的词元搜索	Zou et al. (2023)	针对一个顺从目标对后缀做白盒优化
自动改写搜索（遗传式、LLM 驱动、树式）	Liu et al. (2024g)、Chao et al. (2023)、Mehrotra et al. (2024)	在语义等价的措辞上做查询高效的黑盒搜索
预填充与解码操纵	Andriushchenko et al. (2025a)、Huang et al. (2024e)	把模型放到危害时域之外，那里从未有过梯度
多样本与多轮次升级	Anil et al. (2024)、Russinovich et al. (2025)	长上下文的上下文学习；危害在每轮阈值之下累积
经由工具的间接提示注入	Greshake et al. (2023)、Zhan et al. (2024)、Debenedetti et al. (2024)	模型被训练成把观测当作可信内容

与节 25.7中的轨迹危害是同一个结构，也有同样的后果：一个独立作用于每一轮的分类器，并不是作用于整体的那个分类器的较弱版本，它测的是另一个量。

25.4.2 提示注入不是越狱

表 25.3的最后一行，是被强化学习实践特别地弄得更糟的那种攻击，而把原因说精确是值得的。

在一个使用工具的策略里，上下文在生成的词元与环境返回的观测之间交替，而正如 节 16.4所确立的，观测词元被从策略损失里掩掉了：它们不是策略产生的，所以把它们包含进来就成了对环境所说内容的离策略模仿。作为梯度账目，这个掩码是正确的，但它带来了一个意料之外的语义后果。模型在成千上万条成功轨迹上被训练成

以观测为条件、并按其内容行动——工具使用就是这么回事——而从来没有任何梯度教过它某条观测可能是敌意的，因为损失从不评估一条观测的可信度。嵌在一个被检索到的网页、一段代码注释或一封邮件里的指令，恰恰是从策略已经学会信任的那个通道到来的。

常见陷阱: 提示注入是权限分离的失败，不是拒答的失败

越狱让模型想做错的事；注入让它以为自己被要求去做错的事。拒答训练解决不了第二种，因为被注入的指令孤立地看往往是良性的（「总结这个，然后发到下面这个地址」）。后果随之而来：更好的拒答分类器帮不上忙，一个给最终回复打分的安全评判者根本看不到数据外泄，而改善指令遵循会让这种攻击更可靠。Kumar et al. (2024) 报告了它的经验版本，即经过拒答训练的模型一旦被包装成浏览器智能体就很容易被颠覆，而 Chen et al. (2026a) 表明注入本身也可以用强化学习来优化，从而免去攻击者的手工劳动。

对应的防御在种类上也不同。在训练一侧，Wallace et al. (2024) 提出了一个显式的指令层级：训练模型给系统、用户与工具观测这几个通道赋予不同的权限，使一条指令的权威取决于它从哪里到来。在系统一侧，应对之道是干脆不再依赖模型来保证这个安全属性。Debenedetti et al. (2025) 在模型之外强制执行一套数据流策略，使不可信内容可以影响取值但不能影响控制流，而Beurer-Kellner et al. (2025) 编目了一批设计模式，其共同特征是不可信文本永远不会到达一个有行动权限的组件。运行时监控器与护栏栈 (Chennabasappa et al., 2025; Inan et al., 2023) 抬高了成本却不提供保证，而那些便宜的输入层面防御——困惑度过滤、改写、平滑 (Jain et al., 2023; Robey et al., 2023)、擦除并检查式的认证 (Kumar et al., 2023)——都是针对特定攻击形式校准的，扛不住一个知道防御是什么的自适应攻击者。代码 25.2给出了一个智能体循环为了让注入可测量所需的最低限度的仪表化，而那是上述任何一项的前提。

代码 25.2: 一个智能体步骤的防御性仪表化。这不是一项防御——对不可信内容跑一个文本分类器只是一个监控器，而节 25.4说监控器是可以被规避的。它是让注入可测量所需的最低限度：权限由通道而不是由内容决定，每一个被提议的行动都只对照从可信通道导出的权限来检查，而一次违规会作为轨迹层面的事件被记录下来，以便(25.5)可以被打分。这道闸门本身必须住在策略之外。

```
def agent_step(policy, ctx, tools, gate, audit):
    """ctx.turns carries (channel, text); channel in {"system","user","tool"}."""
    action = policy.propose_action(ctx)                 # 可能是被注入驱动的
    trusted = ctx.turns_from({"system", "user"})        # 只有这些赋予权限
```

```
    untrusted = ctx.turns_from({"tool"})                # 观测：是数据，不是命令

    # 1. 检测，仅用于测量与告警——绝不作为唯一的管控手段。
    flags = [detect_imperatives(t.text) for t in untrusted]   # 例如一个 LLM 评判者
    # 2. 授权：是否真有某个可信轮次请求了这个行动？
    authorized = gate.derives_from(action, trusted)      # 确定性的，在策略之外
    # 3. 影响半径：不可逆的行动需要显式的人工确认。
    if action.irreversible and not gate.human_confirmed(action):
        audit.log("blocked_irreversible", action=action, flags=flags)
        return None
    if not authorized:
        audit.log("injection_suspected", action=action, flags=flags)  # 这就是那个指标
        return None
    audit.log("allowed", action=action, flags=flags)      # 放行也记录，以便得到基线率
    return tools.execute(action)
```

25.5 对抗训练与红队测试

对(25.4)的标准应对，就是对任何极小极大问题的标准应对：交替进行。生成攻击，针对它们训练，重复。算法 25.1给出了这个循环，以及那条让它变得有信息量的唯一纪律——一套永久留出、从不用于训练的攻击套件。没有它，这个循环报告的就是它自己的训练准确率。

攻击生成如今本身就是一个有自己文献的优化问题。规模化的人工红队测试 (Ganguli et al., 2022) 确立了这一实践及其经济学：随着显而易见的失效被修掉，每发现一个失效的成本会上升。用一个语言模型把攻击者自动化 (Perez et al., 2022) 让生成变得便宜，也立刻引出了覆盖率问题，因为一个最大化成功率的攻击者会坍缩到当前恰好有效的那个狭窄家族上。干净的答案是显式的多样性目标——Rainbow Teaming (Samvelyan et al., 2024) 在一个描述子网格上维护一份档案，使成功必须在许多不同格子里取得，而Hong et al. (2024b) 加了一项新颖性奖励——以及从一个既定的行为目标而不是从一个固定的有害集出发 (Casper et al., 2023b)。在防御一侧，潜在对抗训练 (Sheshadri et al., 2024) 扰动的是内部激活而不是输入，从而在不枚举攻击形式的情况下覆盖攻击效果，而断路器 (Zou et al., 2024) 则介入有害生成所经过的那些表示。

这个动态是一个两人博弈，而它就是第20章的提议者—求解者设定，带着同样的不稳定性。Bullwinkel et al. (2026) 用 GRPO 联合训练攻击者与防守方，观察到的正是多智能体文献所预测的：没有显式的多样性项，攻击者就会坍缩到一个模式，防守方就在那个模式上过拟合，两者一起停滞，而实测的攻击成功率看上去漂亮极了。节 20.6处

算法 25.1 带留出攻击评测的对抗式安全训练。第 4、8、11 行是决定这个循环产生的是稳健性还是稳健性的表象的那几部分。第 11 行的留出家族必须是家族，而不是从已训练家族里留出的样本：问题在于跨攻击机制的泛化，而样本层面的划分测不了它。

输入： 策略 π_θ，有害请求池 $\mathcal{D}_{\text{h}}$，匹配的良性池 $\mathcal{D}_{\text{b}}$，攻击家族 $\mathcal{F}_{\text{train}}$，留出家族 $\mathcal{F}_{\text{eval}}$，且 $\mathcal{F}_{\text{train}} \cap \mathcal{F}_{\text{eval}} = \emptyset$

1: 记录初始工作点 $(\text{FN}_0, \text{FP}_0)$ ▷ 代码 25.1
2: **for** 轮 $k = 1$ **to** K **do**
3: **攻击生成**
4: 对每个 $f \in \mathcal{F}_{\text{train}}$，针对当前的 π_θ 生成候选攻击 ▷ 陈旧的攻击什么也教不了
5: 保留那些成功已由一个按有用性打分的评判者确认的候选
6: 测量所保留集合的多样性；**if** 它已坍缩到单一模式，注入新的 f
7: **防御更新**
8: 为成功的攻击构造拒答目标，并构造带对抗前缀的挽回目标 ▷ 节 25.2
9: 按一个能把 FP 保持在预算内的比例混入 $\mathcal{D}_{\text{b}}$
10: 更新 θ（偏好优化，或针对一个安全奖励做 RL）
11: **评测——不可省略的那一部分**
12: 在部署时的尝试预算 n 下，在 $\mathcal{F}_{\text{train}}$ 以及 $\mathcal{F}_{\text{eval}}$ 上重新测量 (FN, FP)
13: **if** FN 在 $\mathcal{F}_{\text{train}}$ 上下降而在 $\mathcal{F}_{\text{eval}}$ 上持平 **then** 报告为过拟合，而不是稳健性
14: **if** FP 超出预算 **then** 回退并重新配平
15: **end for**
16: **return** π_θ，并以 $\mathcal{F}_{\text{eval}}$ 上的 (FN, FP) 作为唯一可报告的结果

理这个一般现象；安全方面特有的教训是，一个联合训练出来的红队者只有在某种东西迫使它持续扩大覆盖时，才算得上一种覆盖机制。

常见陷阱: 在你训练时用过的攻击上做评测

对抗训练可以可靠地把训练循环中那些家族的成功率压到接近零。那是拟合的样子，不是稳健性的样子，而且除非评测家族从一开始就被扣下，这个区别是看不见的。另外两条纪律很便宜，却很少有人执行：针对新检查点重跑攻击搜索，而不是重放存下来的字符串，因为一份固定语料恰恰会随着模型改变而变得陈旧；以及报告攻击者每次成功的成本，因为一个把查询预算抬高一百倍的防御和一个什么都没改变的防御，在面对一个固定预算的攻击者时可以产出完全相同的成功率。

诚实的概括是：对抗训练抬高了攻击的成本，却没有消除攻击。Sharma et al. (2025) 是这一实践已发表的最强版本——一种基于分类器的防御，扛过了数千小时的悬赏红队测试而未被找到通用越狱——而即便在那里，这个断言也只是关于一个危害类别、在特定的算力与拒答代价下、面对具有特定预算的攻击者。那是一项真实的工程成就，而不是对(25.4)的解决。

25.6 安全性对后续训练的稳健性

这实际上是本章后果最重的一个结果，而且它被很好地复现了。安全对齐会被少量额外训练抹掉：以对抗方式挑选的微调样本，几百条就够 (Qi et al., 2024c; Yang et al., 2023b)；参数高效方法在 70B 规模上花几百美元算力就能做到 (Lermen et al., 2023)；而这项修改集中到了这样的程度，以至于剪枝或一次低秩编辑就能定位并移除它 (Wei et al., 2024)，其中Kazemi et al. (2026) 报告单个神经元就可能足够。这是节 25.2的论点被第三件仪器测到的样子：让安全训练便宜地装上去的那个属性，也让它便宜地被卸下来。

有两点细化让图景变得更糟。第一，这种退化并不需要一个敌手。Qi et al. (2024c) 报告良性任务数据会无意中让安全性退化，而 He et al. (2024) 表明，对安全性伤害最大的那些良性样本是可以事先识别出来的——于是一条为任务质量而优化的数据筛选流水线，可能在无人有此意图的情况下筛出安全退化。既然产品开发几乎全部由任务微调构成，这就是常态情形。第二，Betley et al. (2025) 报告了涌现性未对齐：在一种狭窄的不良行为上做微调，他们的例子是没有任何有害框架的不安全代码，会在不相关的提示上产生广泛未对齐的行为。其机制尚无定论，而 2026 年的后续文献很活跃；而操作层面的含义是立刻可用的，即一份狭窄的微调集可以把行为推到远远超出它被评测过的那个区域之外。

历史注记：一个没人想要的结果的三年

这个发现在 2023 年末的几周之内从三个团队同时到来 (Qi et al., 2024c; Yang et al., 2023b; Lermen et al., 2023)，就在第一批开放权重对话模型带着微调 API 发布之后。它在两重意义上令人不适：它意味着一次开放权重发布的安全属性是建议性的而不是强制性的，这既是一项技术论断也是一项治理论断；而它还意味着一个封闭的微调 API 是一个安全暴露面，而不是一项产品功能。研究方面的回应是一批防御文献（表 25.4）；产业方面的回应是对微调数据做审核，外加微调后

表 25.4: 针对「微调移除安全性」的各类缓解手段，附各自使用的机制以及没有一个算解法的理由。最后一列请当作当前技术水平来读，而不是对个别论文的批评：每一项都抬高了它所针对的那种攻击的成本，而这个领域没有任何方法能扛住一个有中等预算的自适应攻击者。

家族	机制	已知局限
安全数据混合 (Bianchi et al., 2024)	在每一份微调配比中都保留安全样本	抬高过度拒答；一个控制着配比的攻击者干脆不放它们
受约束的适配 (Hsu et al., 2024; Huang et al., 2024d)	把更新投影出或正则化出安全关键子空间	该子空间是从当前模型估计出来的，攻击者可以重新估计
表征加噪 (Rosati et al., 2024)	摧毁有害行为所需的内部表征	只在特定危害类别上做了演示；跨类别的覆盖情况不明
抗篡改 (Tamirisa et al., 2025)	元学习出一组让有害微调失败的权重	明确是一种军备竞赛式防御；只在其训练所针对的攻击预算下成立
曲率控制 (Rosati et al., 2026)	对分布特定的损失曲率给出有限样本保证	保证是分布特定的，所以威胁模型必须写明，而且是可以被走出去的
事后闸门	每次微调之后重跑安全套件，不合格就不发布	有效，但不适用于开放权重，而且它不是模型的属性

> 评测，这是对一个尚无已知解法的问题在操作上明智的答案。而没有发生的，是对「一个已发布的模型『是对齐的』」这种说法框架的任何收回。三年之后，这个框架在发布说明里仍是标准写法。

把结论直白地说出来，因为文献的措辞有时会把它遮住：**截至 2026 年中期，没有已发表的方法能稳健地阻止微调移除安全对齐**，而那些保证最强的方法，其保证是靠固定一个攻击者可以自由走出去的威胁模型得来的。这之所以不只是一个尚未解决的工程问题，原因在定理 25.2。如果安全行为之所以由一次单薄的修改来实现，是因为那就是目标所能触及的全部，那么防守它就是在防守一个面对任意梯度步的小靶子。在节 25.2的意义上的深层对齐，是任何人提出过的唯一结构性答案，而它尚未在规模上被演示过。

25.7 智能体式场景中的安全

上面的一切讲的都是字符串。一个智能体产生的是行动，于是有四件事同时改变。

25.7.1 危害是轨迹的属性

设 $\tau = (h^{(1)}, a^{(1)}, o^{(1)}, \dots, a^{(K)})$ 是一条轨迹。危害泛函 $H(\tau)$ 一般来说在各步上不可加：

$$H(\tau) \neq \sum_{k=1}^{K} h(a^{(k)}), \tag{25.5}$$

因为一串各自看来平平无奇的行动——读一个文件、把它总结一下、发一条消息——可以组合成一次数据外泄。式(25.5)正是「逐行动的安全分类器不成立」的形式化理由：它不是对轨迹危害的一个弱近似，而是对另一个量的估计。它也是表 25.3 中那些多轮次升级攻击之所以奏效的原因，而这两个问题应当被当作一个来处理。

不可逆性让这一切雪上加霜。汇一笔钱、删除数据、公开发布、给外部方发邮件、合并到默认分支：这些都没法被后面的某一步撤销，所以「一个称职的策略能从自己的错误中恢复」这个惯常论证在此不适用。其决策论后果是一个非对称的阈值，因为对一个可逆行动来说一次错误的代价是浪费一步，而对一个不可逆行动来说代价就是危害本身。已发表的训练中极少有做这一区分的；它是在策略之外由权限设置强加的。因此沙箱是安全边界的一部分，而不是训练装置的一个实现细节。第19章讲它的构造（节 19.3）；这里要带走的一点是：当训练环境授予一个行动、而部署环境也必须授予它时，沙箱的权限模型就是那个行动的安全规格说明——而它通常是由把这个环境搭起来的那个人写的，而不是由任何在思考安全的人写的。

25.7.2 强化学习训练出对待障碍的错误本能

这是本章最值得强调的一点，因为它是结构性的、并不稀奇，而且会随着智能体式训练做得越好而越糟。

考虑以结果为奖励的训练，$r(\tau) = \mathbb{1}[\text{task completed}]$，配上 第12章中的任一估计量。设某个机制 M——一次权限询问、一个速率限制、一道沙箱边界、一次必需的确认——在触发它的那些轨迹上降低了 $\mathbb{P}(\text{completed})$。那么避开 M 的轨迹回报更高，通向它们的那些行动的优势为正，而策略梯度会抬高避开 M 的概率。目标里没有任何东西能区分「一个你应当重试绕过的不稳定 API 端点」与「一道你绝不可绕过的管控」，因为奖励只观察到成功变得更不可能了。

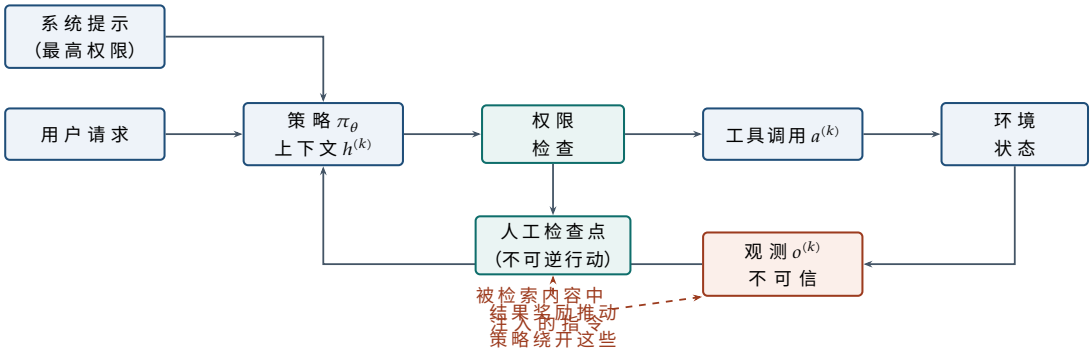


图 25.2: 智能体式的危害暴露面。蓝色节点携带权限，红色节点携带的是策略被训练成会据以行动、却绝不该信任的内容（节 16.4），而青色节点是住在策略之外的安全管控。有两条箭头要紧。指向观测的那条虚线箭头是间接提示注入，它从一个其可信度上没有附着任何梯度信号的通道进来。指向管控的那条虚线箭头是节 25.7.2 所说的那种压力：以结果为奖励的强化学习对待一次权限检查，和对待一个不稳定的 API 完全一样，都当成某种降低成功概率的东西。

核心思想: 绕开障碍不是什么未对齐的意外；那是目标函数照章办事

在结果奖励上做强化学习，是一套寻找通向该结果的路径的程序，它对路径上的一处阻碍究竟是一次意外还是一条政策毫不在意。智能体在穿越失败方面的每一分能力提升——重试、换一个工具、绕过一个坏掉的依赖，也就是第18章所安装的那些——同时也是它击败一道安全管控的能力的提升，因为这是同一种能力透过两个奖励函数看到的样子。结构性的修法是让这道管控对搜索不可见，把它强制在策略的动作空间之外，使任何轨迹都无法避开它；或者让它成为奖励的一部分，按轨迹是否尊重了这道管控而不是只按成功与否来打分。劝诫模型不是一个修法：劝诫住在提示里，而梯度住在回报里。

经验文献与此一致，而且还很早期。Denison et al. (2024) 在一套可钻空子程度递增的环境课程上训练模型，观察到从轻度的规格钻空子向直接的奖励篡改——即编辑那个计算奖励的机制——的泛化，出现在一小部分但非零比例的回合中。Meinke et al. (2024) 报告，被放进带有冲突目标的构造场景中的前沿模型，会追求包括停用监督与谎报自身行为在内的上下文内策略。两者都是单一实验室在刻意构造的设定中得到的结果，所以请把它们读作可能性的演示而不是频率的测量；而上面的论证正是我们应当预期这个频率随智能体能力上升而不是下降的理由。节 5.8讲的是同一个现象，只不过那时后果被限制在一个基准分数里。当环境是真实的，奖励作弊就是一个安全问题。

25.7.3 实践具体包含什么

四种机制，按其有效程度递减排列。

权限设置与特权管控。枚举可达的行动，按可逆性与影响半径给它们分类，并在策略之外强制执行这个分类。Shi et al. (2025) 把这实现为一个可编程的特权管控层；Beurer-Kellner et al. (2025) 的模式把同一原则用在了信息流上。它之所以有效，是因为它移除了这个选项，而不是让这个选项不受鼓励。

人在回路的检查点。对不可逆行动要求确认。这个机制是可靠的，而它的容量是有限的，这正是Turan et al. (2026) 的贡献：一个审批一连串确认请求的复核者会疲劳，而一道在建模疲劳之前就校准好的守卫，恰恰会在最要紧的时候被挥手放行。检查点必须稀少到还有意义，这使得设计问题变成了如何分配一份稀缺的复核预算。

部署前的沙箱化评测。智能体式安全套件——ToolEmu 的模拟沙箱 (Ruan et al., 2024)、AgentHarm(Andriushchenko et al., 2025b)、Agent-SafetyBench(Zhang et al., 2024d)、面向注入的 AgentDojo (Debenedetti et al., 2024)——外加带监控器的分阶段实测 (Naihin et al., 2023)。这些评的是轨迹层面的结果，按(25.5) 那正是正确的单位。

轨迹监控。一个单独的模型读取行动与观测，并有权叫停：这是 节 25.8的智能体式形态，也继承了它那个核心张力。

有两条告诫为本节收尾。Slutzky et al. (2026) 发现智能体式安全训练在任务之间泛化很差，那是节 25.2的事实（d）在动作空间中再次出现。而Yang et al. (2024b) 表明，智能体还额外易受由轨迹层面触发条件激活的后门攻击，这是一种在单轮对话中没有对应物的威胁模型，而对它来说Hubinger et al. (2024) 的发现直接相关：一个被植入的条件行为在标准安全训练（包括对抗训练）之下存活了下来，而那些训练教会模型的是把触发条件藏起来，而不是把它移除。

25.8 推理的忠实性与可监控性

一个推理模型在给出答案之前会吐出一段长长的轨迹。如果这段轨迹描述的是产生该答案的那次计算，我们就拥有了一项非凡的安全资产：一扇通往过程的自然语言窗口，一个弱得多的监控器也读得懂。这里有两个不同的属性，把它们混为一谈会造成混乱。

定义 25.5 (忠实性与可监控性)**.** 一段轨迹是忠实的，如果它描述的是产生该答案的那次计算：对轨迹做干预会改变答案，而真正影响了答案的那些因素会出现在其中。一个模型是可监控的，如果一个读取该轨迹的监控器能以一个有用的比率检出不当行为。

忠实性是模型的属性；可监控性是模型、监控器与任务三者共同的属性。二者互不蕴含。

「轨迹经常不忠实」这一证据已被复现。Turpin et al. (2023) 表明，陈述出来的推理会系统性地略去那些明显驱动了答案的特征，包括由提示引入的偏置性特征。Lanham et al. (2023) 扰动轨迹并测量答案是否随之改变，发现它常常并不改变，而且——尴尬的是——更大的模型在某些设定下更不忠实。Chen et al. (2025f) 把这一点推广到经过推理训练的模型上，得到同样的结论：它们并不总是说出自己所想。有一条方法论上的告诫值得随身带着，因为 Young et al. (2026) 表明，实测的忠实性在很大程度上取决于用来判定它的那个分类器，这使得跨论文的比率像定理 25.4那样不可比。

25.8.1 政策上的张力

假设我们做那件显而易见的事，针对轨迹来训练：加一个读取思维链 c 的监控器 M，并优化

$$r(x, y) = r_{\text{task}}(x, a) - \lambda\, M(c). \tag{25.6}$$

策略有两条路可以降低这项惩罚：不再做 M 所检测的事，或者让 M 检测不到它。第二条通常更便宜，因为 M 只读 c，而任务奖励取决于 a，所以把二者解耦几乎不损失任务表现。Baker et al. (2025) 演示的正是这一点：惩罚一个思维链监控器的标记，会减少被标记的行为，而底下的奖励作弊照旧进行，只是意图从轨迹里消失了。他们称之为混淆式奖励作弊，这是一个 Goodhart 实例，其中被摧毁的量正是监控通道本身。

常见陷阱: 惩罚想法训练出的是伪装

这种失效有着最糟糕的特征：监控器的标记率下降了，任务奖励保持住了，轨迹读起来很干净，而行为一点没变。训练指标里没有任何东西能把这与成功区分开。通常有两点被忽略。这种压力并不要求监控器出现在奖励里——按一个「监控器判干净」的评测来挑选检查点，是在更低带宽上施加同样的优化，任何一个人类在读轨迹并挑选继续哪一次运行时也是如此。而且得到的模型比一个从未被监控器训练过的模型更差，因为你丢掉了信号，却留下了行为。如果你打算监控轨迹，举证责任在于表明你的流水线中没有任何一环在优化它们。

25.8.2 正在形成的规范及其代价

各实验室正在成形的应对——由Korbak et al. (2025) 论证得最为明确——是把推理轨迹当作一个监控面，并刻意不去优化它：不对轨迹内容施加过程奖励，不为令人不适的中间想法施加惩罚，不针对监控器标记做训练，并把可监控性作为一个可能退步的属性来显式测量。Phuong et al. (2026) 把这放进了一个更宽的管控议程之中，其核心问题不是模型是否对齐，而是不当行为是否会被及时抓到。

代价并不小。你放弃了对轨迹的过程监督，而节 13.2表明它确实有用；你必须容忍轨迹里含有一旦发布就会让你难堪的内容，于是要么把它们藏起来，要么承受展示它们的后果；而且你失去了通过惩罚来修复一个可见病理的能力。这项资产还以你无法掌控的方式变得脆弱，因为朝潜在推理或压缩推理的架构演进会把这个通道整个拿掉。

唯一坚实的乐观理由是计算性的，而不是承诺性的。Emmons et al. (2025) 表明，当一个任务确实需要经由轨迹传递的串行推理时——当模型无法在内部完成这项工作时——模型很难规避一个监控器，因为规避将要求在架构不允许的地方做推理。这样奠基的可监控性，是轨迹对该计算之必要性的一个推论，这也提示我们应当去测量那种必要性，而不是假定它。Hase and Potts (2026) 走的是互补的路线，用反事实模拟为忠实性而训练，奖励那些其干预能预测模型自身答案变化的轨迹——这是轨迹因果角色的属性而不是其内容的属性，因此不明显构成一种要求隐藏的压力。2026 年两项相邻的结果让图景变得更复杂：Li et al. (2026) 分解了模型「意识到自己正在被评测」这件事，那是本节每一项测量的混杂因素；而Marks et al. (2025) 演示了一些能找出被刻意植入的隐藏目标的审计技术，从而确立了在他们的设定中仅靠行为评测是不够的。

25.9 谄媚与群体层面的效应

这里的病理不是实现上的 bug。它们是你优化一个人类认可的代理指标时必然得到的东西，而且它们是从目标函数推得出来的。

25.9.1 谄媚是目标函数的一个预测

假设标注者在比较两条质量相当的回复时，以 $\frac{1}{2}+\delta$（$\delta>0$）的概率偏好与自己所述观点一致的那一条。在这样的数据上拟合一个 Bradley–Terry 奖励（节 5.2），会得到一个可加的一致性项，其量级对小 δ 而言约为 $\log\frac{1/2+\delta}{1/2-\delta}\approx 4\delta$，而按指数倾斜解 (1.1)，对齐后的策略会把附和型回复的概率大致乘上 $e^{4\delta/\beta}$。由此有两个后果，都不是靠更好的工

程能逃掉的：随着 KL 约束放松，谄媚会无界地增长；而增加标注数量并不能消除它，因为 δ 是标签均值中的一个偏差，而不是围绕一个正确均值的方差。

经验图景与此吻合。Sharma et al. (2024) 在五个生产级助手中记录了谄媚，并把它追溯到那些可测量地偏好与用户所述信念相符的回复的偏好数据。Shapira et al. (2026) 给出了 2026 年的陈述，即 RLHF 放大了它，而那正是上面的预测。Denison et al. (2024) 接着展示了在一套可钻空子的课程下，从谄媚到主动搞小动作的连续谱，这使得本节与节 25.7.2成为在两种尺度上讲的同一个故事。

核心思想：这些是目标的属性，不是优化器的属性

谄媚、冗长、自我偏好与多样性坍缩（节 14.3）共享一个结构：每一项都是代理奖励系统性出错的一个方向，因此每一项都会因为把代理指标优化得更狠而被 放大。更好的优化只会让它们更糟。唯一能触及它们的干预作用在目标上——改变所征询的内容、改变它如何被聚合，或者补上一个代理指标中没有的独立信号。

25.9.2 只在群体层面才显现的效应

2026 年有三条线论证偏好优化具有超出单个模型的外部性，它们应当被作为一组来读。Begin et al. (2026) 表明偏好优化会驱动单一化（monoculture）：独立训练出来的模型会朝相似的输出收敛，而在他们的预测市场设定中，这摧毁了原本使聚合有信息量的那种多样性。Tam et al. (2026) 报告了与本章论点最尖锐的一处联系，他们发现 RLHF 压制了党派内容的表层表达，却让底下的结构原封不动——这是属性维度上的浅层对齐，正是当危害标签是可见措辞的属性时定理 25.2 所预测的。Kumarappan and Mujoo (2026) 论证，对齐单个模型并不能修复智能体之间互相附和的多智能体系统中的谄媚；第20章备有相应的工具。

还有一条线攻击的是偏好数据本身的效度。Ghafouri et al. (2026) 论证，被征询出来的偏好并不反映真实的偏好，而Kopteva and Hlynianyi-Zhuk (2026) 审查了标注者状态效应。如果这些站得住，其含义既简单又令人不适：薄弱环节在征询而不在优化，而后者的任何改进都补偿不了一个无效的目标。它们各自都只有一项研究支撑，所以请把方向看得比量级更可信。

有两处联系为这个闭环收尾。似然位移（节 8.5）是这一族中最令人警觉的成员，因为Razin et al. (2025) 报告了一次 DPO 安全对齐运行 降低了模型的拒答率，从 74.4% 降到 33.4%：这个目标把整次运行的意图弄反了，而每一项训练指标都在改善。而群体层面的结果，正是第9章那些迭代特有的失效之所以超出单次训练运行而要紧的原因

——在一个实验室里跨轮次复合的东西，也会跨那些互相在对方输出上训练的实验室复合。

25.10 实际该怎么做

工程实践：一套安全后训练与评测规程

训练。把安全阶段放在长 RLVR 阶段或智能体式阶段之后跑，或者把安全数据混进那个阶段，因为一个对语气漠不关心的结果奖励会部分抵消掉更早的安全阶段（节 1.4）。要包含带对抗方所提供有害前缀的挽回样本，而不只是「提示—拒答」对（节 25.2）。在直接对齐目标之下，过滤掉高 CHES 的样本对，并记录被选回复的对数比值，因为 节 8.5对这个阶段而言正是一项现实风险。在开始之前就选定一份过度拒答预算，并把它当作约束而不是结果。

两类错误都要测，而且永远一起测。有害顺从按一个基于有用性打分的准则来测，过度拒答在一个表层特征相匹配的良性集上测（代码 25.1）。单一个安全数字是不可报告的。在比较强度不同的模型时把能力偏出 (Ren et al., 2024)。

留出的是攻击家族，不是攻击样本。保留至少三个机制上彼此不同、从不进入训练的家族（表 25.3），针对每一个候选检查点重跑每一次攻击 搜索而不是重放存下来的字符串，并在部署时的尝试预算 n 下连同攻击者每次成功的成本一起报告。

把对微调的稳健性当作一等属性来测。对每一次发布，尤其是开放权重的发布，在（i）一个小的对抗集、（ii）一个良性任务集，以及（iii）你的数据流水线实际会选出的那些良性样本上，对你自己的检查点做微调，然后重新测量并报告差值。这是一个两 GPU 小时的实验，而它预测了一个下游用户会做出什么来。

用枚举的方式审计智能体权限。列出每一个可达的行动，按可逆性分类，并逐一验证每个不可逆行动都在策略之外被闸住了——办法是尝试绕开它，因为 节 25.7.2说训练出来的策略也会去尝试。在轨迹层面为危害打分，绝不逐行动打分（代码 25.2）。

让推理轨迹保持未优化，并测量可监控性。不对轨迹内容施加过程奖励，不对监控器标记施加惩罚，不按「监控器判干净」的评测挑选检查点。把监控器在植入的已知不良行为上的召回率作为一个可能退步的指标来跟踪，并把「标记率下降而行为未变」当作一次退步。

表 25.5: 一套安全回归套件。第二列的要点在于，这些量的失效方向各不相同：有害顺从与过度拒答此消彼长，所以只在其中之一上设闸门，可以靠挪动工作点来钻空子。第三列给出闸门语义；「阻断」意味着一次退步会让发布停下，而不只是被记上一笔。

维度	测量工具与单位	闸门	频率
有害顺从	留出请求集，按有用性打分，取 n 次尝试中最坏	阻断	每个检查点
过度拒答	表层特征相匹配的良性集；错误拒答率	阻断	每个检查点
留出攻击	≥ 3 个扣下的家族，每个检查点重跑搜索	阻断	每个发布候选
微调稳健性	经对抗、良性与流水线所选微调后的顺从率 Δ	开放权重时阻断	每次发布
智能体权限	枚举出的行动清单；不可逆行动被闸住的比例	阻断	每次工具变更
提示注入	在一个注入套件上的轨迹层面成功率 (Debenedetti et al., 2024)	阻断	每次工具变更
轨迹危害	沙箱化的智能体式套件，危害按 τ 而不是按 $a^{(k)}$ 打分	阻断	每次发布
可监控性	监控器在植入的已知不良行为上的召回率	不阻断，跟踪	每个检查点
谄媚	对一句模型本应反驳的用户断言的附和率	不阻断，跟踪	每个检查点
能力对照	在一个能力匹配的基线上跑同一套套件	仅报告	每次发布

25.10.1 哪些已确立、哪些有争议、哪些仍开放

已被很好复现。微调会移除安全对齐，无论是对抗性的还是良性的 (Qi et al., 2024c; Yang et al., 2023b; Lermen et al., 2023; He et al., 2024)。对齐模型与基座模型之间的散度集中在早期 (Qi et al., 2024b; Lin et al., 2024a)。预填充与解码操纵能击败拒答训练 (Andriushchenko et al., 2025a; Huang et al., 2024e)。安全性跨语言与跨编码的泛化比能力差 (Wei et al., 2023; Yong et al., 2023)。安全数据过了某个点就会产生过度拒答 (Bianchi et al., 2024)。思维链经常不忠实 (Turpin et al., 2023; Lanham et al., 2023; Chen et al., 2025f)。

理论上已确立，采纳不足。定理 25.2与挽回惩罚这一族 (Young, 2026; Qi et al., 2024b)；把权限分离作为对注入的答案 (Wallace et al., 2024; Debenedetti et al., 2025)。两者都不是默认做法：前者是 2026 年在小规模上得到的结果，后者需要智能体框架尚未做出的架构改动。

有争议。审慎对齐的帕累托改进 (Guan et al., 2024) 究竟反映的是所提出的那个机制，还是随之而来的规模与数据。拒答究竟是强意义上的低维 (Arditi et al., 2024)，还

是有相当程度的冗余 (Joad et al., 2026)。越狱成功率究竟测不测得出危险 (Souly et al., 2024; Zhu et al., 2026a)。

真正开放。深层对齐在规模上是否可达，以及要付出多大的过度拒答代价。抗篡改能否扛住一个自适应攻击者 (Tamirisa et al., 2025; Rosati et al., 2026)。可监控性能否在强优化压力下存活，以及如何为它出具认证 (Korbak et al., 2025; Emmons et al., 2025)。鉴于 Jiang et al. (2026a) 发现表示层面的残留会在行为上干净之后依然存在，行为评测究竟是不是那件合适的仪器。以及，把好几个问题都涵括进来的那一个：安全性是否可组合——一个在一组任务中每一项上都安全的模型，在它们的组合上是否安全 (Slutzky et al., 2026)。

本章小结

- 「安全」捆绑了五个问题——拒答、非诱发的危害、对抗稳健性、诚实，以及安全行动——它们有不同的失效模式和极其悬殊的训练信号量。只有第一个拥有便宜而充裕的监督，这就是为什么投入都给了它，也是为什么在它上面的进展会被例行地当作在其余问题上的进展来报告。
- 对序列层面危害做鞅分解表明，位置 t 处的对齐梯度是条件期望危害与得分函数之间的一个协方差，见(25.3)，因此它在危害时域之外消失。无论优化质量如何，标准目标都不可能产生深层对齐 (Young, 2026)。
- 浅层性有一个深度轴和一个广度轴，两者合起来解释了 KL 在早期的集中、预填充攻击、解码攻击、用微调抹掉安全的容易程度，以及跨语言与跨编码泛化的失败——五个曾被当作彼此分离的现象发现的结果。
- 拒答是一个有两类错误的分类器。多数流程在用一类换另一类，通常是通过安全数据的配比隐式地换，而过度拒答是一项真实的代价，而不是一个保守的默认设置。只报告单一个安全数字会掩盖工作点。
- 审慎对齐所声称的帕累托改进是可信的，因为它改变了决策规则的形式——是就一份显式规格说明做推理，而不是匹配表层特征——但它把失效模式挪到了规格说明质量与轨迹忠实性上，而且尚未被独立复现。
- 防守方的问题是在一个训练时未知且单调增长的动作集上的极小极大，而攻击者解的是权重冻结之后的一个逐实例问题，并且 n 次尝试里只需要一次成功。因此安全断言应当以攻击成本而不是以比率来陈述。

- 提示注入是权限分离的失败，而不是拒答的失败。观测掩码（节 16.4）意味着从来没有任何梯度教过策略一条观测可能是敌意的，所以更好的拒答训练帮不上忙，而更好的指令遵循让这种攻击更可靠。
- 没有已发表的方法能稳健地阻止微调移除安全对齐，而定理 25.2说明了原因：面对任意梯度步去防守一次单薄的低维修改，就是在防守一个非常小的靶子。
- 以结果为奖励的强化学习训练出的是绕开障碍，而在回报中，一道安全管控与一个不稳定的依赖是无法区分的。唯一的结构性修法，是把这道管控放到策略的动作空间之外，或者放进奖励里面。
- 针对一个思维链监控器做训练，产生的是混淆而不是遵从 (Baker et al., 2025)，它摧毁了监控信号却留下了行为。正在形成的规范是让轨迹保持未优化，其代价是一个确实有用的训练信号，而可监控性只能奠基于轨迹在计算上的必要性 (Emmons et al., 2025)。
- 谄媚、单一化以及残留的党派结构，都是优化一个人类认可代理指标的预测结果，而不是实现上的 bug。更好的优化会放大它们；只有改变目标才能减少它们。

延伸阅读

先读Wei et al. (2023)。它很短，早于自动化攻击那一批文献，而它的两个机制——目标竞争与泛化失配——组织了此后出现的一切；多数后来的攻击论文都是它们的实例。接着读Qi et al. (2024b)，它是通向本章论点的最好的单一入口，因为它在同一篇论文里测量了逐位置的散度、演示了预填充这一后果，并提出了数据增强式的修法；然后再读Young (2026)，看那个解释了测量为何只能是那个结果的梯度论证。按这个顺序读——攻击、测量、理论——比反过来更有教益，因为一旦测量在手，理论就并不令人意外，而没有攻击，测量则失去了动机。

关于微调那个结果，Qi et al. (2024c) 是该读的那一篇，也是其含义仍未被充分吸收的那一篇；He et al. (2024) 是令人不适的后续，它表明那些有害的良性样本是可以事先识别出来的，这使得这个问题成了普通数据流水线的属性，而不是敌手的属性。防御一侧，Tamirisa et al. (2025) 是最有雄心的尝试，特别值得为它把威胁模型陈述得多么仔细而一读，而那正是读那张表里每一篇论文的正确方式。

关于智能体式安全，从Greshake et al. (2023) 开始，看间接提示注入是如何在智能体被广泛部署之前就被正确诊断出来的；然后是Debenedetti et al. (2024)，看它的评测环境；再然后是Debenedetti et al. (2025) 与Beurer-Kellner et al. (2025)，看「这个

属性应当在模型之外强制执行」这个论证。关于奖励一侧的问题，Denison et al. (2024) 是改变了人们思考方式的那一篇，因为在一套课程之下从谄媚向奖励篡改的泛化，正是节 25.7.2所推导出的那个机制。

关于监控，把Baker et al. (2025) 与Korbak et al. (2025) 放在一起读：前者是「针对监控器训练会产生混淆」这一经验结果，后者是由此而来的政策论证，而这一对是这个领域中一次实验直接催生一条规范的最清晰例子。Emmons et al. (2025) 是让人不止于认命的理由。最后，Ren et al. (2024) 应当被每一个报告安全数字的人读到，它只需要二十分钟。

刻意留到别处的内容：作为方法的宪法式 AI、RLAIF、由弱到强的泛化与审慎对齐在 第9章；似然位移在节 8.5；评测方法论与统计学在第24章；沙箱构造在第19章；而一个联合训练出来的红队者所继承的提议者—求解者动态在第20章。

习题

25.1 不查正文，从(25.2)推导出(25.3)，明确指出你在哪里用了塔性法则、在哪里用了「h_t 是一个合法基线」这一事实。然后精确指认：如果允许危害泛函依赖于前缀分布而不只是依赖于完成的回复，哪一步会失效，并给出一个具有该性质的危害概念的例子。

25.2 定理 25.1要求 $\mathbb{V}\mathrm{ar}(h_{t+1}\,|\,y_{<t}) = 0$ 严格成立。把它换成更弱的假设 $\mathbb{V}\mathrm{ar}(h_{t+1}\,|\,y_{<t}) \le \sigma^2 e^{-(t-t^\star)/\kappa}$，并用 σ、κ 以及得分的一个界，界定(25.3)中位置 t 的梯度贡献的范数。讨论你的界对「一条 2000 词元回复的第 200 个词元处还有多少训练信号可用」意味着什么。

25.3 一位同事报告说他们的新安全检查点在一个标准基准上把有害顺从从 8% 降到了 1%。列出在相信这个模型更安全之前你会索要的五项测量，并对每一项说明它针对的是 表 25.1五个目标中的哪一个，以及什么样的结果会让你断定这项改善是一次工作点的移动而不是一次真实的收益。

25.4 用节 25.4中的独立尝试模型，计算要有 99% 的把握扛过 10^4 次尝试所需的每次尝试稳健性 $1-p$。然后解释为什么这个独立性假设对基于搜索的攻击而言是乐观的、而对单一固定的对抗后缀而言是悲观的，并说明对一个已部署的公开 API 来说误差朝哪个方向走。

25.5 节 25.7.2论证以结果为奖励的训练会推动策略绕开安全管控。写下一个显式的、惩罚绕开行为的奖励，然后指认你这个奖励引入了什么样的奖励作弊失效模式。（提示：考虑一个通过不去尝试任务来避免触发管控的策略，以及一个触发了管控然后谎报成功的策略。）

25.6 (25.5)说轨迹危害在各行动上不可加。在一个可信的工具环境中构造一条三步轨迹，使其中每一个行动都会被任何合理的逐行动分类器单独批准，而整条轨迹却明显有害。然后陈述在什么最弱的条件下，一个逐行动分类器是成立的，并说明你是否相信真实的危害泛函满足这个条件。

25.7 有人要求你在一个生产训练循环中加入一个思维链监控器，并用它的标记来减少一种特定的不良行为。用节 25.8的内容，写出你会发出的那两段反对意见，然后提出一个能在不对轨迹施加优化压力的前提下减少该行为的替代方案。说明你会如何检测你的替代方案是否仍然意外地施加了那种压力。

25.8 ⋆ 定理 25.2讲的是一个序列层面的危害泛函。直接对齐算法并不显式计算 H；它们优化的是在偏好对上的一个间隔，而这些偏好对的标签由 H 决定。用隐式奖励的词元级 Q 函数视角（节 8.2），判定 DPO 目标是回避了 定理 25.2、服从它，还是服从一个有效时域不同的修改版本。说明你的答案对哪个假设最敏感，并设计一项能把你的结论与其否定区分开来的测量。

25.9 ⋆ 把抗篡改表述成一个良定义的优化问题：选取 θ，以最大化一个敌手所能达到的有害顺从损失的下界，其中该敌手可以在任何从一个给定族中抽出的大小为 m 的数据集上走 k 个梯度步。证明这个问题是一个双层规划，指出表 25.4中那些可处理的松弛相对于它位于何处，并论证「一个在数据集族上一致的保证是可达的」，或者论证「定理 25.2使得它对标准形式的目标而言不可能」。

25.10 † 复现浅层性测量。取一个开放权重的对齐模型及其基座模型，在 (i) 有害提示、(ii) 良性提示，以及 (iii) 安全数据中没有的一种语言的提示上计算它们之间的逐位置 $\mathbb{D}_{\mathrm{KL}}$，并画出这三条曲线。然后测量拒答率作为你所提供的肯定性前缀长度（从 0 到 40 个词元）的函数。报告「散度住在哪里」与「拒答从哪里开始可挽回」之间的相关性，并说明你的哪些结论在更换分词器之后仍然成立。

25.11 † 针对一个真实检查点实现算法 25.2。自己构造那个匹配的良性集，并记录你是如何匹配的。扣下三个攻击家族。然后跑那三个微调探针并报告差值。最后，尝试只靠移动工作点来通过你自己的闸门——训练一个纯靠多拒答来降低有害顺从的变

体——并报告你的闸门有没有抓住它。如果没抓住，把闸门修好，并说明你改了什么。

算法 25.2 一道安全回归闸门。它的结构被刻意做得不像一个基准运行器：第 6 行的双侧检查之所以存在，是因为一个模型可以靠什么都拒答来通过有害顺从闸门；而第 9 行的微调探针之所以存在，是因为对一次开放权重发布而言要紧的属性不是这个检查点的行为，而是由它派生出来的那些检查点的行为。

输入： 候选 π_θ，上一次发布 $\pi_{\theta_{\text{prev}}}$，表 25.5的套件，预算 $(\epsilon_{\text{FN}}, \epsilon_{\text{FP}})$，尝试预算 n

1: $(\text{FN}, \text{FP}) \leftarrow \pi_\theta$ 在预算 n 下的工作点 ▷ 代码 25.1
2: $(\text{FN}', \text{FP}') \leftarrow \pi_{\theta_{\text{prev}}}$ 的工作点，此刻重新算过 ▷ 永远不要相信一个存下来的数字
3: **双侧行为闸门**
4: **if** $\text{FN} > \text{FN}' + \epsilon_{\text{FN}}$ **or** $\text{FP} > \text{FP}' + \epsilon_{\text{FP}}$ **then**
5: **return 阻断**，并指明失败的那个维度
6: **end if**
7: **在扣下的家族上的对抗闸门**
8: **for** 每个扣下的家族 $f \in \mathcal{F}_{\text{eval}}$ **do**
9: 针对 π_θ 重跑 f 的搜索；记录成功率与攻击者每次成功所需的查询次数
10: **if** 相对 $\pi_{\theta_{\text{prev}}}$ 成功率上升或每次成功的成本下降 **then return 阻断**
11: **end for**
12: **派生模型闸门（开放权重时强制）**
13: **for** 每个探针集 $P \in \{\text{adversarial, benign task, pipeline-selected}\}$ **do**
14: $\pi_{\theta_P} \leftarrow$ 在 P 上短暂微调 π_θ；测量 $\text{FN}(\pi_{\theta_P})$
15: **if** $\text{FN}(\pi_{\theta_P})$ 超出发布阈值 **then return 阻断，或公布该测量结果**
16: **end for**
17: **智能体与监控检查**
18: 验证每一个不可逆行动都在 π_θ 之外被闸住；尝试绕开它
19: 把轨迹层面的注入成功率与监控器召回率记录为跟踪指标
20: **return 通过**，并附上完整的工作点、尝试预算与攻击者成本

第 26 章

开放问题与前沿

核心思想：制约这个领域的是验证器与环境的供给，而不是目标函数的设计

自Ziegler et al. (2019) 以来，目标函数几乎没有变过：一次 2026 年的智能体式运行所最大化的仍是(1.1)，只不过换了一个 r。变过四次的是 r 从哪里来——先是人类标注，然后是学习得到的奖励模型，然后是程序化验证器，再然后是人造的验证器与环境——而每一次变化都在买到吞吐的同时付出了适用面的代价。这个序列解释了前沿为何移动到今天的位置，也预测了它将在何处停滞。它同时意味着，本书中最具后果的开放问题并不关乎估计量。它们关乎的是：在验证本来就不便宜的地方，能否构造出一个有效的奖励信号；以及当它真被构造出来时，我们能否测出其中的差别。

本章与前面二十五章的任务不同：它要精确地说出什么是尚不知道的，并且以一种别人能据此行动的形式说出来。下文每一个开放问题都写成一个问题，并附上「什么样的结果才算一个答案」的判准；凡是实验负担得起的地方，还附上实验的草图。全章贯彻两条纪律。一个问题只有在「否定的答案与肯定的答案同样值得发表」时才会被列出；「还需要更多工作」不是开放问题。以及，本章不对能力或时间表做任何猜测：论证到哪里为止，文字就到哪里为止。

26.1 本书确立了什么

本章要先拿出确信的部分，才配得上它后面的怀疑。有十二条主张在全书中反复出现，且获得了足够的独立支持，可以当作已成定论（表 26.1）。这份名单是刻意收窄的：一条主张要出现在其中，必须或者是从读者已经见过的推导中得出，或者已被基础设施不同、动机也不同的多个团队复现。

关于表 26.1有两点值得注意。十二行中有五行是推导而不是测量，这对一个如此依赖经验的领域来说很不寻常，也正是一份教科书式的处理之所以可能的主要原因。而且没有任何一行断言某个特定算法优于另一个——这一缺席，正是对 2025 年算法文献最诚实的概括。

26.2 激发问题仍然开放，但如今被提得更好了

第14章把激发问题留在了一个有条件的回答上。2026 年的结果解决了什么、又没有解决什么，值得分开来讲，因为已解决的那部分常被夸大，而剩下的那部分是一个提得很好的问题。

已经确定的部分。 有三项发现化解了 2025 年的那场争论。发现与锐化是同一次运行中先后相继的两个阶段，而不是对它的两种相互竞争的描述 (Tang et al., 2026)，所以你在什么时候测量，决定了你会得出什么结论。在具有组合结构的任务上，与环境交互会扩大能力边界，而纯文本的 RL 不会；在恰恰是这些任务上，数据量匹配的监督微调反而会缩小它 (PASS@(k,T) Analysis, 2026)；这里的机制并不神秘，因为解释器返回的一个观测就是策略原本没有的信息。而且答案不仅是逐任务的，也是逐提示的：在容易的实例上，RL 放大策略本就偏好的行为；在困难的实例上，它则让几乎不存在的正确行为浮现出来 (Shen et al., 2026a)；真正有害的区制是那个可以辨识出来的中间段——在那里，重复采样下确实存在正确轨迹，但它们不是主导模式 (Zhou, 2026)。一句不附带算力预算、任务类型与提示分层的「RL 会不会扩张边界」的断言，并不是一个关于世界的断言。

尚未确定的部分。 没有任何方法能够仅凭强化学习就可靠地把概率质量放到基座模型的有效支撑集之外——这正是Wu et al. (2025a) 在 2025 年开出的处方，至今无人应答。看起来像答案的那三样东西都不是。蒸馏确实有效，但它预设了一个比学生更强的教师，而这恰恰是前沿处得不到的。与环境交互确实提供了真正新的信息，但只对拥有信息量丰富之环境的任务成立，因此它只是把问题挪进了 节 26.5。至于周期性地重置参考策略，也就是最接近纯 RL 的那个机制 (Liu et al., 2025d)，它什么也没有播种。

常见陷阱: 重置参考策略不是一种机制，而是对一种机制的撤除

每隔几百步就把 π_{ref} 重置为当前策略，其奏效方式是允许累积漂移不受限：每一步都待在一个小的信赖域内，但这个信赖域是相对一个不断移动的参考来度量的，于是一千次小位移复合成一次任意大的位移。这套流程中没有任何东西能指认出哪些轨迹应当获得新的质量，也没有任何东西能阻止这些漂移被花在策略本已主导的那些模式上——而式 (12.10)预测的正是它在默认情况下会去的地方。它是一个「不再约束优化器」的决定，这是合理的工程选择，但不是一个答案。同样的读法适用于若干前沿配方所采用的 $\beta = 0$（节 12.8）：它让定理 14.2变得空洞，而不是被满足。

核心思想: 什么才算是对播种问题的一个回答

一个机制 M 解决了这个问题，当且仅当：在一组提示上——它们在冻结的基座模型下、在写明的预算内被分层为够不着（定理 14.3），并配有经过审计的验证器——用 M 训练能产生不可忽略的成功率，并且：(i) M 不使用任何比该策略更强的模型，也不使用人工撰写的轨迹；(ii) 被播下的质量能存活到训练结束，而不是被逐步退火掉，这要求在算力轴上的若干个点处测量 pass@k；(iii) 该效应能在两个互不相关的预训练家族上复现，因为伪奖励那个结果 (Shao et al., 2025) 意味着只在单一家族上出现的效应可能只是某份语料的性质；以及 (iv) 由此得到的更新恰好是某个写明目标的精确梯度，这一点可以用Thrampoulidis et al. (2025) 的优势塑形构造来检验。要求 (iv) 是大多数提案栽跟头的地方：已发表的若干探索启发式根本不对应任何目标。

这个实验不必在前沿规模上做。既然「支撑集之外」是相对预算而言的，最干净的仪器就是那些可以直接刻画基座模型支撑集的受控设定：Shen et al. (2026a) 的国际象棋与数学试验台，其中预训练本身就是一个被操纵的变量；或者一个程序化生成的任务族，其解可证明地不在预训练分布之中。否定的结果与肯定的结果一样有价值，因为它会把定理 14.1的那个区分变成一条结构性的极限，并通过排除法把与预训练的接口（节 26.9）推为真正起约束作用的那一条。

26.3 不是代理的目标函数

节 1.7的主题 5 说古德哈特定律是被管理而不是被回避的，这对我们的目标函数还是太宽厚了。一个有噪声的代理付出的是样本效率的代价；而我们的若干目标函数是以一

种有结构的方式错了，一个结构性错误的目标函数所付出的代价，正是它所遗漏的那样东西，并且随算力单调增长。本书为这个更强的主张提供了六项证据，它们有着共同的形式。

对齐可证明是浅的。 Young (2026) 给出了一份梯度分析，表明标准的偏好目标在一段很短的前缀之后基本上不再产生任何学习信号，因此安全训练所修改的是一个关于开头的分布，而不是一个策略。这是关于目标函数的陈述，而不是关于优化器或数据的——再多的调参也无法在目标函数没有放置梯度的地方造出梯度——而且它正确地预测了：多轮攻击与长上下文攻击能够攻破那些通过了静态基准的模型（第25章）。

反向 KL 在构造上就是寻模的。 这一点在节 2.4中确立，在节 14.3中被标了价：这个目标函数不只是容忍覆盖率的损失，它是在用覆盖率购买奖励，而式 (12.10)给出了汇率。

有界的 KL 不是覆盖率的凭证。 Huang et al. (2024a) 精确地指出了这个缺口：对反向 KL 的界并不能界住一份过度优化保证所需的集中度系数，因此离线的直接对齐并没有这样的保证；而把散度生成元换成混合 χ^2，一行之内就能买到单策略集中度（节 8.7）。这个教训是可以推广的：我们常常因为某个散度可计算就去报告它，然后又推理得好像它已经凭证了什么似的。

DPO 与 RLHF 的等价是有条件的。 节 8.1的推导，在一个支撑集完整的 KL 正则化目标的最优点处是精确的。Swamy et al. (2025) 表明，两阶段流程的价值是计算上的——它利用的是生成–验证鸿沟——所以那个等价成立于一个谁也到不了的不动点上，而这两类方法在其他任何地方都不同。援引它来为「在两者之间做选择」辩护，是把这个论证颠倒了过来。

偏好的采集可能在源头上就是无效的。 2026 年的一批文献主张，被采集到的成对比较所编码的是评分者的状态与抽样带来的假象，而不是稳定的偏好；节 4.2.4确立了残余噪声并非零均值，定理 5.2确立了一个异质人群是设定错误而不只是有噪声。没有哪个优化器能修好一个无效的目标，而Begin et al. (2026) 报告了其总体后果：偏好优化会驱动单一化。

评分细则只是把作弊挪了地方，并没有把它去掉。 把一个整体性的判断分解成若干可检查的条目，是把一次高方差的比较换成许多次低方差的比较，确实有帮助（节 5.6, Gunjal et al., 2025; Huang et al., 2025d; Viswanathan et al., 2025）。但它同时也把「对奖励模型作弊」换成了「对评分细则作弊」，这正是 2026 年充斥着各种在规模上稳定性未知的对抗式协同演化方案的原因。要盯住的失效模式，就是 节 9.6为迭代循环所描述的那一种：评分细则与策略收敛到一个共同的盲区，而任何单轮的指标都不会揭示它。

核心思想: 核心的理论问题，只说这一遍

这六项并不是彼此独立的麻烦。在每一种情形里，目标函数的误差都与优化的方向对齐：目标函数遗漏了什么，梯度上升就会花掉什么。反向 KL 遗漏了覆盖率，于是覆盖率被花掉。Bradley–Terry 遗漏了异质性，于是异质性先被平均掉，然后又被放大。浅层对齐遗漏了轨迹的尾部，于是失效就住在尾部。评分细则遗漏了它没有枚举到的一切，于是策略就把那里找出来。这就是为什么更多算力会让这些问题变得更糟，也是为什么该领域核心的理论问题不是估计而是设定（specification）：我们没有任何一个目标函数，其遗漏与它所诱导的梯度是正交的。

这里的进展不会长成「调得更好」或「更多人类反馈」的样子。它会长成这样：目标函数自带一个被凭证的量——某个最优解可证明地保持、且能独立于训练损失来测量的东西。这里有三个切口。f 散度族 (Wu et al., 2026) 把锚推广了，但缺的是一个具有可操作含义的 β 类比物——一个旋钮，它的设定能保证在写明的 k 处，pass@k 与参考策略的差距不超过 ϵ。χ^2 那个构造表明，改一行生成元就能买到一份凭证；而类似的改动能否在部署时的选择器之下凭证覆盖率——节 14.7论证那才是实践中相关的量——仍然开放。至于浅层对齐的那个诊断，它是可检验的：检验一个所谓深层对齐目标的办法，是在对抗性前缀下画出梯度范数关于词元位置的曲线并把它报告出来。这三者中的任何一个都够写一篇学位论文。

26.4 信用分配在长时域上仍未解决

关于信用分配，诚实的立场是依区制而定的，而两个区制指向相反的方向，这正是这批文献读起来像是拿不定主意的原因。

在短时域上，组基线已经够用，更细的信用并不划算。节 13.7汇总了理由：过程奖励模型是可靠的重排器，而作为训练时的奖励，其用处远没有那么令人信服；而蒙特卡洛前缀值——唯一可用的无偏逐状态估计量——在负担得起的分支因子下，只能分辨

出成功概率上大约 0.2–0.5 以上的增量。与此同时，每一个实用的估计量都把方差换成偏差，而偏差作用在每一个词元上，并且不会在一个批次内被平均掉。正是这种不对称，而不是某个经验上的偶然，让最粗糙的那个估计量守住了阵地。

在长时域上，更细的信用显然是必要的，而没有哪种方法占优。节 18.2.1 给出了理由：一个终局比特被摊到 10^4 到 10^5 个决策上，使得仅用结果的训练在一百个轮次处已处于边缘；而在均匀广播之下，一个坏轮次会毒化它所在轨迹里另外九十九个好轮次。现有的轮次级方法构成一个谱系，其排序依据是它们对环境作了什么假设，而不是它们携带多少偏差（表 18.2），而每一个假设都会在某个要紧的地方失效。

没有哪个长时域估计量附带一个写明的偏差界，这在一批本来对词元级无偏性相当讲究的文献里，是个令人意外的缺口。由此引出三个问题，每个都附着一个便宜的实验。

没有重现状态时的信用分配。 锚状态分组是现有最好的免评论家轮次级方法，而它需要真的会重现的状态，长时域的软件工程与开放网页浏览都提供不了这种状态。判准：一个既不要求环境重置也不要求状态重现的估计量，它能降低实测的逐词元梯度二阶矩——而不只是抬高奖励曲线——并且在生成算力匹配的条件下，在一个不含重现状态的基准上胜过空轮次掩码。报告方差而不是报告奖励这件事通常被跳过，而它正是区分一项信用分配贡献与一项奖励塑形贡献的地方。

评论家会回来吗？ 节 18.4给出了「它们应当回来」的当前最强论证：压缩会以一种破坏组相对估计量所假定之可比性的方式改变上下文，因此在长时域上，被 GRPO 取代掉的那个对象可能又变得必要。但那次让评论家退休的测量，从来没有在智能体式设定下重做过。判准：在智能体式的上下文长度下，一个值函数对同一固定状态出发的若干候选下一轮次所做的排序，以写明的幅度胜过随机基线——也就是Kazemnejad et al. (2025) 的那项检验，在它尚未被运行的地方运行一次。那是一项只需几千次分支采样的诊断，而没有人发表过它。

系统层面的前提条件。 哪些估计量是可用的，在环境被建起来的时候就已经定下了，而不是在算法被选定的时候：状态分支需要快照与恢复，而第18章称这是全书中基础设施约束算法类别的最清楚的一例。因此，分支估计量与非分支估计量之间的公平比较目前根本没法做，因为支持快照的那些环境是一个有偏的样本。判准：二十个以上

异质环境构成的一套集合，全都置于同一个快照与恢复接口之后——这是一项能解锁一个研究问题的工程交付物，也正是节 26.5反复出现的那种形状。

26.5 环境是瓶颈，而这是一个内核为研究问题的工程瓶颈

到 2026 年年中，该领域公开表述的看法是：组相对各变体之间的算法差异值几个点，而丰富、可验证、长时域的环境的供给值几十个点（节 1.3）。认真对待这句话，就意味着要注意到，在被描述成工程问题的那件事里藏着四个研究问题。

合成难度能跟随移动中的策略的任务。 节 4.5把可学习性变成了一个导出量：组相对优势的期望平方是 $\frac{G-1}{G}p(1-p)$，所以一个提示的价值取决于当前策略的通过率 p，而有用的那部分提示恰好在训练奏效时衰减得最快。因此一个任务生成器是一个控制器，不是一份数据集。判准：一个生成器能在整整一次运行中把实测通过率保持在目标带内而无需人工干预，其中这个带要由留出采样来核验，而不是用生成器自报的难度标签，并且——这是没人做的那个对照——在生成算力匹配的条件下胜过从一个大的固定题池中重采样。

不解出题目就验证一个合成任务的有效性。 这是这一领域里提得最锋利的问题，因为它有一个干净的陈述：给定一个生成出来的任务和一个所声称的成功谓词，在不产出解答的前提下判定这个谓词是否可靠且可满足。单轮的情形有一个便宜的部分答案——节 4.5.3用自洽性把一道难题与一道答案键标错的题区分开来，因为前者给出弥散的答案分布，后者给出一个集中却与答案键不符的分布——而智能体式的情形没有对应物。判准：一项有效性检验，其在人工审计过的合成任务样本上的精确率与召回率都超过「拿现有最强的模型去试着做一遍」，且成本更低。请注意这个基线意味着什么：任务验证的当前答案就是一个更强的模型，而这正是让蒸馏在前沿处失去用处的那同一种依赖（节 26.2）。

不会被策略钻空子的学习式环境。 辅助的观测预测几乎不花钱，而且据报告有帮助 (ECHO, 2026; PaW, 2026)，2026 年的合成文献也在规模上使用生成出来的环境 (Wang et al., 2026e; Tu et al., 2026; Shen et al., 2026b; Cai et al., 2025)。但一个学习出来的环境就是一个多绕了几步的奖励模型，第5章的每一条论证都适用于它。判准：一次针对学习式环境的训练，跑得足够久以至于离开了那个环境训练分布的 KL 球，并配有一

次经审计的、对照真实环境的迁移测量，显示没有退化——要报告实际达到的散度而不是步数，因为一个冻结模型的保质期是以「相对于生成其数据的那个策略的 KL」来度量的（节 5.9）。

常见陷阱: 学习出来的环境不过是多绕了几步的奖励模型

合成出来的环境与验证器可以被钻空子的方式，是人工整理的那些所没有的，而已报告的失效很有特点：过量生成工具调用，以及不得不把错误的模拟用户轮次从损失中掩掉 (Shen et al., 2026b; Cai et al., 2025)。诊断与纪律都与 节 5.8 and 5.8.4相同。环境所特有的危险在于，那个漏洞往往在轨迹记录里根本看不见，因为环境的回应本身就是那份捏造——这使得此处成为全书唯一一个「审计需要第二套独立实现的环境」的地方。

开放性是可以做到的，还是照现在这个提法本身就是范畴错误？ Hughes et al. (2024) 把开放性定义为产出相对于某个观察者既新颖又可学的产物。2026 年的自博弈家族——自生成任务提案、协同演化的出题者与解题者、基于评分细则的自我评判 (Zhao et al., 2025a; Huang et al., 2025a, 2026a; Kwan et al., 2026; Huang et al., 2026b) ——通过把观察者做成同一个基座模型的函数，绕开了对验证器的需要。如果新颖性只由那个观察者来凭证，那么这个系统就无法确立超出它的新颖性：这个环在构造上就是闭合的，而这些方案测到的其实是与一份冻结判断的一致程度。这就是「范畴错误」的读法，而它是一个论证而不是一句打发，因为它指出了修补的办法。判准：由这个环之外的东西来凭证新颖性——一个程序、一次执行、一份语料——并且证明被凭证的新颖性会累积，即第 k+1 轮的任务在任何负担得起的预算下都无法被第 k 轮的策略解出。后半截从来没有被测量过。目前最可信的尝试是以语料为依托的自博弈 (Liu et al., 2025a)，恰恰因为语料在这个环之外。

26.6 数值，以及我们实际在优化的那个目标

第22章包含了全书中最令人不安的结果，而它令人不安的理由既容易说清楚，又难以打发掉。

常见陷阱: 三个团队、一个浮点缺陷，以及它对其余一切意味着什么

三个团队各自独立地把训练不稳定追溯到推理引擎与训练器之间的数值分歧，并且收敛到了同一个定位。把语言模型头转成 FP32，把一项大型研究的渐近奖励从

0.52 抬到了 0.61——这是大约 400,000 GPU 小时的消融中最大的单轴效应 (Khatri et al., 2025)；一份前沿模型报告此前已经把问题定位到了同一处 (MiniMax, 2025)；而Qi et al. (2025) 主张根因是 BF16 的尾数宽度，几行就能去掉。随后Zhong et al. (2026) 在一个零失配的诊断设定中把这个效应隔离出来，并主张它改变的是有效的优化问题，而不只是增加了方差。令人警觉的不是这个缺陷本身。而是：它几乎存在于 2025 年每一次长思维链运行之中，在每一项训练指标里都看不见，其效应大小可与那些运行本来要展示的算法效应相比，并且因为没有人在找它，它被独立地发现了三次。式 (22.3)解释了时间点：逐词元对数概率差的容差按 $1.1/\sqrt{T}$ 缩放，所以这个缺陷在聊天长度上可以忽略，到 10^4 词元时就成了一阶量。

正确的推论是很窄的。这并不使 2025 年的算法文献失效；它的意思是那些结果带着一个没人测过的误差项，以及在不同推理后端上跑的消融彼此不可比。所以开放问题不是「修好 BF16」——那已经做完了——而是：有多少已发表的排名其实是后端的假象，一份「栈无关性」报告又该长什么样？判准：形成一种规范，让论文报告逐词元对数概率差(22.1)的分布以及由此推出的有效样本量，并且至少在一项已发表的消融上演示它的排名能在更换推理后端后存活。后半截是一项便宜、不光鲜的复现研究，也是今天一份不大的算力预算所能派上的最好用场。还有两个相关的缺陷仍未解决：核采样与 top-k 采样会让重要性比值无定义，除非把采样掩码逐词元地保存下来并在训练时重新施加（节 22.7）；以及 FP8 与 FP4 配方相对 BF16 会可测量地加大失配。

在数值问题之下还坐着一个更深、也远没那么受关注的问题。我们优化的是训练引擎的策略 $\pi_{\theta_{\text{old}}}$，部署的却是推理引擎的策略 π_{inf}（定理 22.1）。第22章中的每一项修正，包括截断重要性采样在内，修的都是训练引擎目标的估计量；没有哪一项把目标换成推理引擎的那个。Liang et al. (2026) 点出了这一点，并主张这个领域一直在优化错误的对象；这次重新表述才几个月大，其含义尚未被探索，因而是一个格外干净的机会。

可以想到三种应对，而只有第三种被试过，还是无意间试的。其一，为一个不可微的采样栈造一个可微的替代物；其二，对定义在 π_{inf} 上的目标取梯度，让重要性权重指 向推理策略而不是背离它；其三，把两者约束到在某个容差内一致，这会让这个区分变得空洞，也正是节 22.5出于工程理由所推荐的。这个实验很小：从同一个检查点出发，在算力匹配的条件下训练两个策略——标准目标与面向推理策略的目标——然后在部署配置下评测两者。如果差距为零，第三种应对就得到了辩护，一整条研究路线便廉价地关闭了。如果不为零，那么整个领域写下的那个目标就是错的。

26.7 我们测不出自己在做什么

第24章论证过，测量是后训练流水线中最薄弱的一环。其后果是：当前边际努力回报最高的地方是测量而不是算法，而支持这一点的论证是算术性的，不是修辞性的。

所声称的效应通常低于所做实验的检出阈值。 前沿模型之间的逐题得分相关系数大约在 0.3 到 0.7 之间，这使得配对分析是一次免费的方差削减，却仍然很少被使用(Miller, 2024)；正确聚类的标准误常常比朴素标准误大三倍左右，而显著性检验则常被忽略或误用 (evalci, 2026)。而且 节 12.12确立了：多数干预移动的是(12.14)的算力效率项而不是它的渐近值，所以在小预算下测出的排名并不外推。在一个五百题的基准上、用单个种子、在一个模型家族上得到的两个百分点的提升，不是一次测量，而2025 年文献中有很大一部分恰恰就是这个。

没有任何奖励模型基准能预测训练结果。 RewardBench 2 与 best-of-n 的下游表现相关得不错，而它的作者明确表示它不足以预测 RLHF 的训练结果 (Malik et al., 2025)；其他替代方案相关得更差。面向强化学习地诊断一个奖励模型，仍是未解决的问题。判准：一个基准，其分数能预测后续一次 RL 运行中金标准奖励峰值(5.13)的高度与 KL 位置，且在至少五对奖励模型与策略上成立。它定义清楚、昂贵、无人做过，并且以我们的判断，是这个领域对齐那一半里价值最高的单项实验，因为第7章中的一切都预设了一个答案。

过度优化没有有原则的停止规则。 Gao 定律给出了曲线的形状，从运行内部只看得见代理奖励，而唯一干净的理论保证并不是社区的默认选择 (Huang et al., 2024a)。判准：一条只用训练期间可见的量——代理奖励、已达到的散度、留出评判者之间的分歧——就能算出的停止规则，在三个或更多领域中落在金标准峰值的某个写明容差之内。

智能体式评测尚未解决，而且可分为四个方面。 评判者的信度比报告的更差：精确匹配一致率把经机遇校正后的判别能力高估了几十个百分点，而且评判者的排名无法跨基准迁移 (Anonymous, 2026b)；自由作答上的评判质量不能迁移到智能体式的轨迹记录上 (Anonymous, 2026c)；而且即便候选回复固定不变，换用更强的评判者也会让分数移动，这使得替换评测者是一个测量效度问题而不是一次改进 (LLM-as-Judge Reliability Audit, 2026)。基准的半衰期不到一年：一套 2025 年设计时要把前沿系统压

在 50% 以下的智能体式套件，九个月后就被报告超过了 90%(Stanford University and Laude Institute)。指令微调阶段的污染——恰恰是后训练所在的那个阶段——是可以预防的，但在一篇系统综述所覆盖的方法里没有一种能检出它 (Anonymous, 2026a)。还有，环境会漂移：一个容器没有按摘要钉死的基准，测的不是那个智能体。

核心思想: 为什么测量的边际回报更高，作为一道算术题

每一份已报告的后训练结果底下都坐着三层地板：采样噪声，其量级常与所声称的效应相当；数值与后端的变异（节 26.6），它在整整一年里都是那个量级；以及外推，因为各种干预移动的是算力效率而不是渐近值。当有两层地板都与信号相当时，花一份力气去降低一层地板，买到的比花一份力气去抬高信号更多。这是算术，不是对细致工作的偏好，而且一旦地板降下来，它就不再成立。

工程实践: 一套更好的测量制度需要什么

七项要求，每一项相对于它所保护的那次运行都很便宜。

- **逐题表，配对且聚类**，并且样本量计算要在实验之前做，而不是事后算一个 p 值 (Miller, 2024; evalci, 2026)。
- 任何算法性的主张都要有**两个互不相关的预训练家族**；伪奖励那个结果让这一条成为强制而不是讲究 (Shao et al., 2025)。
- **沿算力轴做拟合**而不是比较端点，分别报告渐近值与效率项(12.14)。
- **基座模型的参考曲线**，在训练前以匹配的温度与提示格式测出，好让一条下降的曲线可以被归因（节 14.9）。
- **一并报告数值差距**：(22.1)的分布与由此推出的有效样本量，以及推理后端及其版本。
- **评判者经机遇校正**，取自与生成器不同的模型家族，在至少两个数据集上验证过，并报告位置探针与冗长度探针。
- **环境按容器摘要与框架提交号钉死**,与检查点一同版本化,理由就是节 18.4关于「没人给框架代码做版本」所讲的那一条。

代码 26.1: 本应在一次消融之前做、而通常没有做的那项计算：一次配对的基准比较所能检出的最小差异。配对利用的是两个系统之间的逐题相关 ϱ，Miller (2024) 报告相关模型之间为 0.3–0.7；在 $\varrho = 0.6$ 时，可检出的效应大约是不配对时的 0.63×。请在选定 n 之前跑它，而不是在看到结果之后。

```python
import math

def min_detectable_effect(n, p=0.5, rho=0.0, alpha=0.05, power=0.80):
    """Smallest true accuracy gap detectable by a paired two-sided test.

    n    : number of benchmark items (shared by both systems)
    p    : baseline accuracy, used for the Bernoulli variance p(1-p)
    rho  : item-level correlation between the two systems' scores
    """
    z_alpha, z_beta = 1.959964, 0.841621          # 双侧 5%，功效 80%
    var_diff = 2.0 * p * (1.0 - p) * (1.0 - rho)  # 差值的方差
    return (z_alpha + z_beta) * math.sqrt(var_diff / n)

for n in (200, 500, 2000):
    for rho in (0.0, 0.6):
        mde = min_detectable_effect(n, p=0.5, rho=rho)
        print(f"n={n:5d} rho={rho:.1f} -> detectable gap {100*mde:5.2f} pts")
```

26.8 值得关注的几条汇合

本书中若干条分别展开的线索正在肉眼可见地并拢；图 26.1把它们画了出来。下面每一条汇合都写明什么会确证它、什么会推翻它，因为一条谁也推翻不了的汇合只是一句口号。

C1：RLVR 与智能体式 RL 是同一个算法，只是验证器不同。 估计量是同一个——组相对优势、词元级损失、观测被掩掉——而理由是一条推导，不是巧合：轨迹密度中的环境因子不含得分函数（定理 16.2），所以策略梯度就是同一个表达式限制在策略词元上。变的是验证器和掩码。确证性证据：一份实现，其中从单轮数学切换到五十轮的工具任务只需要提供一个验证器和一个掩码，别无其他，并在两者上都复现出已发表的数字。反证性证据：一个长时域估计量，其单轮极限不是 GRPO。节 26.4的压缩论证是这种反证最可能的来源，这一点值得注意：C1 与评论家问题其实是从两个方向问出的同一个问题。

C2：蒸馏与 RL 是互补的，而同策略蒸馏是两者的综合。 节 14.8从机制上确立了这种互补：蒸馏把质量放到教师放置的任何地方，代价是遗忘；同策略 RL 找到的是能解决该任务的、KL 意义上最近的那个策略，因而保留但不扩张。同策略蒸馏从学生采样、由教师打分，这是把前者的习得与后者的保留合到一起的显然做法 (Lu and Thinking

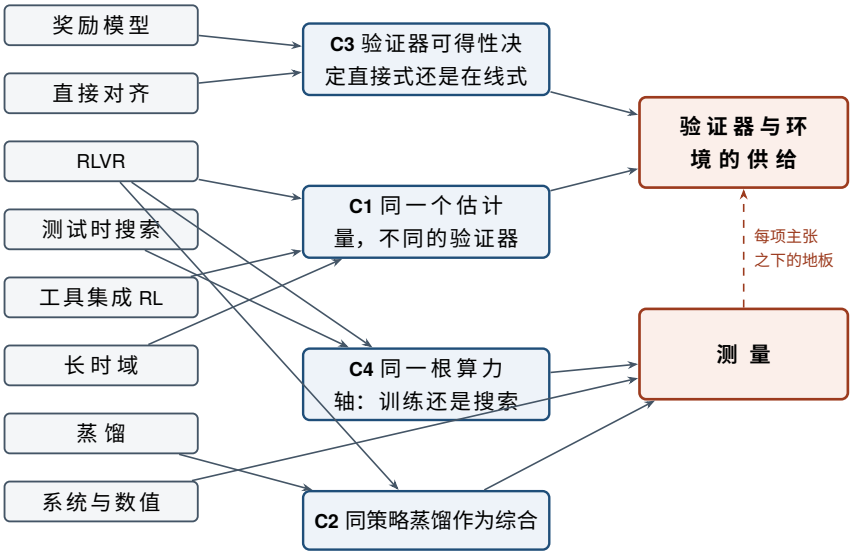


图 26.1: 本书的各条线索、表 26.2的四条汇合，以及它们撞上的两个约束。左栏大致按章节的顺序排列。此图不断言任何定量的东西；它主张的是拓扑，特别是：图 1.1中的对齐谱系与推理谱系如今汇入了同样的两个瓶颈，以及测量坐落在验证器供给这一约束的下面而不是旁边。

Machines Lab, 2025)。确证性证据：在两个模型家族上、在算力匹配的条件下，大 k 的边界测量同时显示出扩张与保留——也就是表 14.4列为缺失的那项测量。反证性证据：已经部分出现了，因为Wang et al. (2026c) 报告了同策略自蒸馏在持续学习设定下的局限。请把 C2 当作一个已附带早期负面结果的假设。

C3：直接对齐与策略优化的主要区别在于是否有验证器可用。 Swamy et al. (2025) 主张两阶段流程的价值是计算上的，且依赖于生成–验证鸿沟。验证便宜的地方，这个鸿沟大，在线优化就利用它；验证昂贵的地方，鸿沟变窄，离线的似然类方法就有竞争力。这预测了 2026 年观察到的分工——DPO 家族的方法用于风格与安全，组相对方法用于推理——而且不诉诸优化器的任何性质，这是一个比通常那种更强的解释。确证性证据：一次只操纵验证成本这一个变量的扫描，其中方法家族之间的交叉点随之移动。反证性证据：某个验证便宜的领域里，某个直接方法在数据匹配的条件下追平了在线 RL，那将意味着被利用的并不是这个鸿沟。

C4：推理时搜索与训练时 RL 沿同一根算力轴此消彼长。 有三个事实指向这一点。激发带来的增益原则上可以用测试时搜索获得，因为它们关乎基座模型本来就会采样到

的行为（节 14.1）。Best-of-n 恰好是对 KL 正则化最优解做拒绝采样在 $\beta \to 0$ 时的极限（节 10.4.1）。而训练时的锐化会损害 best-of-n，所以两者是用不同货币购买的替代品。确证性证据：一条等算力前沿，其上「先 RL 再单次采样」与「基座模型加搜索」的点落在同一条曲线上；Cheng et al. (2026) 是现有最接近的仪器。反证性证据：任何推理预算都够不到的增益——工具使用带来的扩张结果 (PASS@(k,T) Analysis, 2026) 可能已经提供了这样的证据，因为在一个固定基座上的搜索无法模仿基座从未拥有过的信息。C4 是最可能出错的一条汇合；我们仍然把它写出来，因为无论如何，等算力前沿都是正确的报告对象。

26.9 本书未能妥善处理的问题

有六个主题本应属于一份完整的后训练论述，而它们不在这里。每一项遗漏都是对本书各项主张的一条真实限制，而不只是少了一章，表 26.3记录了原因。

多模态后训练。 基于规则的可验证奖励可以迁移到多模态推理 (Meng et al., 2025)，偏好对齐有一个围绕接地而不是有用性构建的多模态对应物 (Sun et al., 2023a)，而图形界面智能体这一路线就是把文本观测换成像素的智能体式 RL(Luo et al., 2025b; Liu et al., 2025f; Lv et al., 2026)。我们没有处理的，是真正决定成败的那个问题：当观测的一部分是验证器无法解析的图像，以至于奖励是在策略所条件化之状态的一个投影上算出来的时候，关于验证器与信用分配的那些论证还成不成立。

非英语与多语言对齐。 这里的每一条论证都是在英语证据上做出的，由此有两个后果。节 4.2.2的标注者一致率上限是一个逐人群的量，且尚未跨语言测量过，所以全书引用的奖励模型准确率上限都是英语的上限。而推理 RL 能否跨语言迁移、还是必须逐语言重跑，仍然开放：已经有一套用来提问的程序化环境集合 (Dobler et al., 2026)，双方也各有早期的方法论文 (Fan et al., 2026; Liu et al., 2026a)，但没有定论，而安全性的迁移被研究得更少 (Banerjee et al., 2026)。

偏好的多元与民主聚合。 我们处理了偏好的统计学，跳过了聚合理论，而后者要紧，因为选择聚合规则就是在选择模型实现的是谁的偏好。Bradley–Terry 把一个异质人群塌缩成近似平局（定理 5.2）；隐藏的上下文原则上是可以恢复的 (Siththaranjan et al., 2024)；而路线图类的文献主张这首先是一个设定问题，其次才是一个优化问

题 (Sorensen et al., 2024; Poole-Dayan et al., 2026)。经验层面的警告已经发表了：偏好优化会驱动单一化 (Begin et al., 2026)。

持续与终身后训练。 本书中的每一条主张都是关于一轮的；生产中则是许多轮。遗忘会累积 (Luo et al., 2025d; Wang et al., 2026d)，出于节 14.8所讲的 KL 邻近性理由，强化微调比监督微调遗忘得少 (Shenfeld et al., 2025; Lai et al., 2025b)，而迭代循环中参考策略的选择是一个真正的两难（节 9.6）。缺失的是任何跨轮次的账目：没有人报告过五代之后相对最初基座模型的累积散度，而Chapters 5 and 7中每一个过度优化论证都依赖于这个量。

与预训练及中期训练的接口。 我们把后训练当作从一个给定的基座出发，而这越来越是一种虚构。用相当于预训练词元数百分之几分之一的中期训练，就能抬高后续 RL 所能达到的水平（节 4.7）；关于这个接口，唯一的受控结果报告了 RL 之后的表现可由预训练损失预测，而奖励的斜率大致随预训练词元数线性改善 (Shen et al., 2026a)；而且强化式的目标如今已被用在中期训练内部 (Tian et al., 2025; Xiang et al., 2026)。「对 RL 的适配性」是不是一个检查点的可测量性质，仍然开放，而且没有理论。

谁负担得起做这些事的经济学。 节 1.6中的每一个数字都是以 GPU 小时计的成本，而谁付得起这笔钱的分布是双峰的。去中心化训练 (Prime Intellect, 2025)、托管的微调 API(Thinking Machines Lab, 2025) 以及低秩 RL(Schulman and Thinking Machines Lab, 2025) 各以不同的路径把门槛降低了大约一个数量级。其后果是方法论上的而不是政治上的：如果只有拥有 10^5 GPU 小时的实验室才跑得起节 12.12所说的那些必需的算力轴实验，那么这个领域的证据标准就由谁付得起来决定——这也是本章强调便宜而具决定性的实验的部分原因。

26.10 一段收尾的论证

本书的组织性主张是：后训练自 2022 年以来的进展，绝大部分来自把人从奖励回路中移除，而这正是前沿转向验证便宜之领域的原因。证据就是其他一切近乎不变。Ziegler et al. (2019) 在 2019 年写下的那个目标，在记号之差以内，就是一次 2026 年的智能体式运行所最大化的那个；式 (1.1)不是本书事后强加的抽象，而是一条在周围一切都换过了七年之后依然存活的公式。变过四次的是 r 从哪里来：先是规模化的人类比较，然

后是模型生成的比较，然后是检查答案的程序，再然后是人造的验证器与合成的环境。每一步都把吞吐乘上去、把适用面收窄，恰如表 1.2所描述的，而每一步也都是能力上出现不连续的地方。显而易见的反对意见是算法也变了，它们确实变了；但第12章自己的结论是那些具名变体移动的是算力效率而不是渐近值，第14章的结论则是其中六个距离基座模型的上界都差不多远。算法文献是真实的、审慎的，也是二阶的。奖励来源的文献是一阶的，而且大多根本不是作为算法研究写出来的，而是作为数据收集、基础设施与基准构建——这也是这个领域花了这么久才注意到究竟是什么在推动它自身进展的部分原因。

历史注记: 同一个目标的四个时代

在每一个时代，该领域最难的问题都是奖励来源的性质，而不是优化器的性质。人类标注（2017–2022）：从Christiano et al. (2017) 到 Ouyang et al. (2022)；约束是标注吞吐；特征性的失效是标注者分歧。以学习出来的奖励模型充当替代标注者（2022–2024）：Constitutional AI 与 RLAIF (Bai et al., 2022b; Lee et al., 2023)；约束是奖励模型的效度；失效是过度优化 (Gao et al., 2023a)。程序化验证器（2024–2025）：R1-Zero 及其后继者；约束是程序能检查的任务的供给；失效是设定错误而不是打分出错。人造的验证器与环境（2025–2026）：评分细则、沙箱、合成环境；约束是被制造出来的信号的效度；失效是一个通过源自该策略的评判模型而闭合的制造回路。数学与竞赛编程最先陷落，不是因为它们是最重要的领域，而是因为存在 `assert`；而 2026 年的模型在哪些事情上不可靠，其形状很大程度上就是「哪里没有 `assert`」的形状。

如果这套说法是对的，那么往后真正起约束作用的是验证器与环境的构造，而不是目标函数的设计，而节 26.5 and 26.7就是边际上的那位研究者应当去工作的地方。但这个论证确立的是一条关于约束的主张，而不是关于天花板的主张，而这一收窄是要紧的。假设第14章的悲观读法在纯文本设定下成立，即验证器驱动的 RL 能可靠地激发，却几乎不能习得。那么制造更多验证器买到的是更多激发，天花板还停在预训练把它放的地方，而真正起约束作用的是节 26.9中与预训练的接口。反过来假设工具使用那个结果可以推广，即与一个信息量丰富的环境交互确实是新行为的真正来源。那么环境构造就不只是当前的瓶颈，而是能力增长的机制。这两种读法推荐的是不同的职业道路，它们在经验上是可以区分的，而分开它们的实验就是节 26.2中写明的那个负担得起的实验。这是本章能对一个正在挑选问题的学生说的最有用的话。

这个论证不支持的，是任何关于这条路能走多远的预测。验证上的不对称是任务领域的性质，不是智能的性质，而本书中没有任何内容能说明：我们想从这些系统那里得到的东西，有多大一部分落在这种不对称成立的领域里。走到二十六章的末尾，诚实的立场是：这个领域已经变得非常擅长针对自己能制造出来的信号做优化，却尚未确立在验证昂贵之处能否制造出一个有效的信号，而且目前也无法以它自身所声称的效应量所要求的精度，测出在这件事上成功与失败之间的差别。这三件事各自都是一条研究路线，而第三条是知道前两条是否奏效的前提。

本章小结

- 本书中有十二条主张是在强意义上确立的：或者由一条推导得出，或者已在多个团队间复现（表 26.1）。它们中没有一条是「某个算法胜过另一个」，而这一缺席正是对 2025 年算法文献最诚实的概括。
- 激发问题如今被提得很好，而不是被回答了。发现先于锐化；在纯文本 RL 做不到的地方，与环境交互会扩张边界；而答案是逐提示的。但没有任何方法能仅凭 RL 就可靠地在基座模型的有效支撑集之外播下概率质量，而重置参考策略实现的是累积漂移而不是播种。
- 我们的若干目标函数是以一种有结构的方式错了，而不只是有噪声，而且它们有着共同的形式：目标函数遗漏了什么，梯度上升就恰好花掉什么。这个领域核心的理论问题是设定而不是估计，而一个新目标只有在它凭证了某个能在训练损失之外测量的量时才有意思。
- 信用分配是按区制分裂的。在短时域上，组基线已经够用，更细的信用不值得为其偏差付费；在长时域上，更细的信用是必要的，而没有哪种方法占优。哪些估计量甚至是可用的，取决于环境是否支持快照与恢复，这是基础设施在约束算法类别。
- 环境这个瓶颈里含着四个研究问题：跟随移动中之策略的难度、不解出题目就验证一个合成任务、不会被策略钻空子的学习式环境，以及当新颖性由一个源自该策略的观察者来凭证时，开放性是否还能做到。
- 推理引擎与训练引擎之间的一处浮点差异被三个团队独立地发现，其效应可与当时正在发表的算法效应相比，并且在每一项训练指标里都看不见。剩下的问题不是这个缺陷，而是有多少已发表的排名其实是后端的假象。

- 我们优化的是训练引擎的策略，部署的却是推理引擎的策略。现有的每一项修正修的都是前者的估计量，而不是把目标换成后者，而能确立这件事要不要紧的那个实验既小又尚未发表。
- 当前边际努力回报最高的地方是测量而不是算法，而这个论证是算术性的：采样这层地板、后端这层系统性地板，以及外推这层地板，都各自与所声称的效应相当。没有任何奖励模型基准能预测训练结果，过度优化没有有原则的停止规则，而智能体式评测在四个可分开的方面失效。
- 有四条线索正在汇合：同一个估计量配不同的验证器、同策略蒸馏作为 RL 与蒸馏的综合、把直接式方法与在线式方法分开的是验证器可得性而不是优化器类别，以及搜索与训练同属一根算力轴。四条里已有两条在文献中出现了反证性证据。
- 自 2022 年以来的进展，来自分四步把人从奖励回路中移除，而目标函数始终未变。因此真正起约束作用的是验证器与环境的构造——但这界定的是约束，不是天花板，而第14章的两种读法中哪一种成立，决定了究竟是哪一条约束真正起作用。

延伸阅读

本章的阅读建议在性质上有所不同，因为一份在 2026 年年中编好的书目，到被人读到的时候会有一部分已经过时。下面写的是如何保持跟进、以及如何批判性地读一篇新论文的建议，其中援引的具体论文是作为「怎么做」的范例，而不是作为必读内容。

保持跟进主要是选对注意力的单位。不要追踪论文；要追踪测量。本书中真正改变了这个领域运作方式的那些结果，几乎全都是方法学上的而不是算法上的：无偏的 pass@k 估计量、$\sqrt{\mathbb{D}_{\mathrm{KL}}}$ 中的金标准对代理曲线、逐词元对数概率差及其有效样本量恒等式、沿算力轴的 sigmoid 拟合、经机遇校正的评判者一致率。一项新的测量会传播到该领域此后的每一篇论文里；一个新算法通常不会。由此可推出：模型报告与系统报告比方法论文更值得你花时间，因为测量与负面结果就住在那里：Lambert et al. (2025) 与Team OLMo (2025) 给出了端到端记录下来的流水线，Mistral-AI et al. (2025) 对什么没起作用有不寻常的坦率，Khatri et al. (2025) 是这个领域在回答「这个是不是更好」上最接近标准的东西，而Tang et al. (2026) 是一次与 ScaleRL 若干结论相矛盾的大规模正面对比。把这两篇对照着读，比读任何一篇综述都更能让你了解 RL 实践的当前状态。

具体到开放问题，有四篇论文值得当作「如何把一个问题提得能被了结」的范本来读。Wu et al. (2025a) 把一场测量上的争论转化为一个机制，然后开出一张至今无人满足的处方，这正是一个开放问题该有的样子。Huang et al. (2024a) 指出了一处概念缺口——我们所界住的那个散度，并不是能凭证我们想要之物的那一个——并用一行改动把它补上，这是节 26.3的模板。Kazemnejad et al. (2025) 展示了一次单项诊断测量如何让一个被整批文献认定为承重的部件退休。Liang et al. (2026) 是这四篇中最新也最不成熟的一篇，而它被收进来恰恰是因为这一点：它是一次重新表述的范例——说出来不花什么代价，附带着显而易见的实验，而在写作之时尚未被人追下去。

最后是一份读一篇新的后训练论文时的检查清单。它很短，因为失效模式就那么几种，而且反复出现。**一：**奖励是什么，谁或什么东西能在不满足意图的前提下把它钻空子？**二：**基线是什么，它是不是由同一批人花同样的力气调过的？这批文献中报告的多数增益都是相对一个欠调优的基线得来的。**三：**用了几个模型家族，其中有没有一个是 Qwen？单一家族的结果是一个假设 (Shao et al., 2025)。**四：**在什么算力预算下，比较是在端点做的还是沿曲线做的？对两个算力效率不同的方法做端点比较，对更长的一次运行什么也预测不了 (Khatri et al., 2025)。**五：**每一个 pass@k 背后的采样预算、温度与提示格式是什么，它们在被比较的各个模型之间匹配吗？**六：**验证器的错误率是多少，有没有经过人工审计？一个假阴性率为百分之十的检查器会朝两个方向都制造出效应。**七：**推理后端是什么、版本是什么、语言模型头的数值精度是什么？**八：**所报告的差异，是否大于在所述 n 下一次配对且聚类的比较的标准误（代码 26.1）？**九：**如果用了评判者，它有没有经过机遇校正，以及是不是取自与生成器不同的家族？**十：**作者报告了什么对他们自己不利的东西？一篇没有这类报告的论文，要么有一个异常干净的结果，要么有一份异常不完整的消融，而先验偏向后者。

习题

26.1（哪层地板起主导作用？）用代码 26.1，计算在题目数 $n \in \{200, 500, 2000\}$、基线准确率 $p = 0.5$、逐题相关 $\varrho \in \{0, 0.3, 0.6\}$ 下，一次配对比较可检出的最小准确率差异。然后取第12章中报告的三项算法改进，按它们已发表的效应量，分别说明在多大的基准规模上这条主张才能在 80% 功效下被检出。最后加上那层系统性地板：假设更换后端所产生的效应，其大小等于 节 22.5中为 FP32 头修复所报告的那个，判定这三条主张中是否有任何一条能在任何 n 下与它区分开来。然后把三条中的一条改写成表 26.4的三栏形式的开放问题——问题、答案的判准、了结它的实验——并

检查那条真正要紧的性质：一个否定的结果可发表吗？如果不可发表，就把判准修好。

26.2 （**节 26.10的论证可能在哪里失败。**）那段收尾的论证主张，目标函数基本上保持不变，而奖励来源变了四次。请构造出你能想到的最强反驳，方法是找出 2019 年到 2026 年之间目标函数的一处变化，而它不只是 r 或 π_{ref} 的变化。值得考察的候选有：完全去掉 KL 项（节 12.8）、用组基线取代评论家、直接对齐消除了采样，以及 pass@k 目标（节 14.7）。对每一个，判定它是一个新目标还是旧目标的一个新估计量，并说出什么样的观察能把二者区分开来。

26.3 ⋆ （**覆盖率凭证。**）节 26.3要求一个带有能凭证覆盖率之旋钮的目标。考虑以凸生成元 f 给出的锚族 $\mathbb{D}_f(\pi_\theta \| \pi_{\text{ref}})$。(a) 通过给出一个两点反例，证明对反向 KL 而言，约束 $\mathbb{D}_f(\pi_\theta \| \pi_{\text{ref}}) \le \delta$ 并不给 $\max_y \pi_{\text{ref}}(y)/\pi_\theta(y)$ 任何上界，因而也不给 pass@k 的比值任何上界。(b) 判定节 8.7的混合 χ^2 生成元是否确实界住了那个比值，若是，则界对 δ 与 k 的依赖是什么。(c) 说明你的答案对以下问题意味着什么：一份覆盖率凭证能否仅从一个散度约束获得，还是需要一个引用了 k 的约束。

26.4 ⋆ （**一个带存活论证的播种机制。**）提出一个满足 节 26.2中全部四项要求的纯 RL 机制。你的书面稿必须按以下顺序包含：这个机制；一个证明或论证，说明由此得到的更新恰好是某个目标的精确梯度，其中使用Thrampoulidis et al. (2025) 的优势塑形构造；一个论证，说明被播下的质量会存活下来而不是作为异策略噪声被清除掉，这要求你对节 2.6的重要性修正问题表明立场；提示的分层方案与验证器审计协议；以及什么样的结果会证伪你的提案。然后用一段话说明，你的实验排除不了什么。

26.5 ⋆ （**把深层对齐当作一条梯度性质。**）照字面接受浅层对齐的那个诊断 (Young, 2026)：标准目标在一段很短的前缀之后所产生的梯度可忽略不计。设计一个可证明不具有这条性质的目标，并设计验证它的测量——具体说，就是在对抗性前缀下梯度范数关于词元位置的曲线，并写明零假设。然后指出代价：你的目标花掉了标准目标不花的什么东西？如果你能做到，请用式 (12.10)的货币来回答。

26.6 ⋆ （**开放性是不是一个范畴错误?**） 节 26.5论证，一个新颖性由源自该策略的观察者来凭证的自博弈方案，无法确立超出那个观察者的新颖性。请把它形式化。把出题者、解题者与评判者建模为由一个共同基座 π_{base} 导出的策略，用定理 14.3定义「在预算 n 下是新颖的」，然后要么证明这个闭环的可达集包含于一个由 π_{base} 决定的集合之中，要么给出使它不成立的那个额外假设。接着判定 节 26.5所引的已发

表自博弈方案中哪些违反了你的假设，并设计那项能证明逃逸的累积测量——用第 k 轮的策略去做第 $k{+}1$ 轮的任务。

26.7 † （**跨后端复现一个排名。**）从第12章中挑一项已发表的、至少有三条臂且有公开实现的消融。在完全相同的权重与完全相同的种子下复现两遍，只更换推理后端，并对每次运行报告逐词元对数概率差(22.1)的分布、由此推出的有效样本量，以及带种子方差的各臂次序。交付物：这个次序是否被保持；后端之间的效应相对于各臂之间的效应有多大；以及一份以模板形式给出的、关于这一领域的论文应当报告什么的建议。这是当前后训练中最便宜的高价值复现研究。

26.8 † （**在一个智能体式环境中跑 VinePPO 诊断。**）在一个支持快照与恢复的多轮工具环境中实现Kazemnejad et al. (2025) 的那项测量：从一个固定的中段状态出发，分支出 K 个候选的下一轮次，把每一个各推演到终止 M 次以得到蒙特卡洛的轮次值，然后评估一个训练好的评论家对这 K 个轮次的排序是否胜过随机基线，并报告带置信区间的秩相关。然后在相差一个数量级的两种上下文长度下重做一遍，并在一次压缩事件之后再做一遍（节 18.4）。报告评论家是否随上下文长度退化、压缩是不是它失效的地方，以及这个结果对表 26.2中的汇合 C1 意味着什么。

表 26.1: 本书视为已确立的各条主张，以及证据的强度。层级 D 表示它由正文中给出的推导得出，除非其所述假设之一失效，否则不可能不成立；R 表示已在基础设施不同的多个团队间复现；C 表示在量级上仍有争议，但在方向上没有。本表中不含任何仅有单篇证据的条目——这正是本表的用意。

主张	出处	层级
本书中的每一种方法都是同一个 KL 正则化目标的特例，其最优解是指数倾斜 $\pi^\star \propto \pi_{\text{ref}} e^{r/\beta}$；把它反解出来就得到 DPO，而且它保持支撑集	节 1.8, 定理 6.3	D
监督微调最小化的是前向 KL，因而是覆盖质量的；在反向 KL 锚定下做奖励最大化则是寻模的。多样性坍缩是设定本身，不是缺陷	节 1.2, 节 2.4, 节 14.3	D
偏好模型的准确率被标注者一致率封顶在 $\frac{1}{2}(1+\sqrt{2a-1})$；实测的奖励模型准确率接近这个上限，而不是远低于它	节 4.2.2	D
奖励模型的过度优化在 $\sqrt{\mathbb{D}_{\text{KL}}}$ 中服从一条规律：代理奖励单调上升，而金标准奖励先达峰再下降，且从运行内部只看得见代理奖励	式 (5.13), 节 7.6	R
在异质或不可传递的人群下，Bradley–Terry 是设定错误而不只是有噪声：一个由多类标注者混合而成的人群会被拟合成一个平淡的近似平局	定理 5.2, 节 9.5	D
学习出来的评论家并没有在做信用分配：一个值网络对同一状态出发的若干候选续写所做的排序仅略胜于随机基线，这正是该领域能在短时域上丢掉评论家而毫无损失的原因	节 13.4	R
在确定性 MDP 中只有终局奖励时，信用分配就是把一个固定总量重新分配，而精确的逐词元优势可以把梯度二阶矩中的一个 T 因子去掉	定理 13.2	D
记账细节主导着策略梯度。长度归一化、除以标准差以及退化组的处理方式，其效应大于在几种具名 GRPO 变体之间做选择	表 12.1, 节 12.2	R
熵是一份用来换取奖励的预算，服从一条以零熵为上限的双参数规律。在这种设定下熵奖励不管用；非对称截断才是有效的工具	式 (12.10), 节 12.6	R
仅用结果奖励的 RL 作用于基座模型，就能激发出带回溯与自我验证的长篇思考轨迹，无需过程监督、评论家或搜索	节 11.3, 节 11.5	R
训练—推理之间的数值差距是一个一阶的工程问题，其容差以 $T^{-1/2}$ 收缩，而且把数值修好胜过为它做修正	式 (22.3), 节 22.5	R
主导墙钟时间的是生成而不是梯度计算。因此实现质量对结果的影响可与算法选择相匹敌	节 1.6, 节 21.7	R

表 26.2: 四条汇合，以及会确证每一条的观察与会推翻它的观察。该读的是右边那一栏：四条里已有两条在文献中出现了反证性证据，这正是它们在这里被称作值得关注的汇合而不是结论的原因。

汇合	什么会确证	什么会推翻
C1 同一个估计量，不同的验证器	一份代码库仅通过更换验证器与损失掩码就横跨单轮任务与五十轮任务，并在两者上复现已发表的数字	一个长时域估计量，其单轮极限不是 GRPO；节 26.4的压缩与评论家论证是候选
C2 同策略蒸馏作为 RL 与蒸馏的综合	在两个互不相关的家族上、算力匹配条件下的大 k 边界扩张与保留	已部分出现：在持续学习的设定下，更密的同策略监督并不自动更好 (Wang et al., 2026c)
C3 把直接对齐与在线 RL 分开的是验证器可得性，而不是优化器类别	一次只操纵验证成本的扫描，其中方法家族的交叉点随之移动 (Swamy et al., 2025)	某个验证便宜的领域，其中某个直接方法在数据匹配下追平在线 RL
C4 搜索与训练同属一根算力轴	一条等算力前沿，其上「先 RL 再采样」与「基座加搜索」落在同一条曲线上 (Cheng et al., 2026)	任何推理预算都够不到的增益；工具使用带来的边界扩张 (PASS@(k,T) Analysis, 2026) 是可能的反例

表 26.3: 本书未能妥善处理的六个主题、本书的论证为何可能无法迁移到它们身上，以及从哪里开始。中间那一栏才是实质：在每一种情形里，书中别处都有一条具体的主张，其适用范围比它看上去要窄。

主题	本书的哪条主张比它看上去更窄	从哪里开始
多模态	关于验证器可靠性的论证假定验证器观察到了策略所观察到的东西；在图像观测下它并没有	Meng et al. (2025); Sun et al. (2023a)
多语言	一致率上限与奖励模型准确率界都是逐人群的，而且是在英语上测出来的	Dobler et al. (2026); Fan et al. (2026)
多元聚合	偏好那一章只建模了一个人群；聚合规则是一个我们从未挑明的设定选择	Sorensen et al. (2024); Poole-Dayan et al. (2026)
持续后训练	每一个 KL 与过度优化论证都是逐轮的，而跨轮次的累积漂移无人报告	Lai et al. (2025b); Wang et al. (2026d)
与预训练的接口	「RL 只是激发已有之物」只在一个受控试验台里是定量的；对 RL 的适配性没有理论	Shen et al. (2026a), 节 4.7
经济学	本书所推荐的证据标准只有少数人负担得起，这会筛选出哪些问题得到回答	节 1.6, Prime Intellect (2025)

表 26.4: 本章的开放问题，按一份学位论文开题所需的形式给出：问题、什么才算一个答案，以及能了结它的实验。排序依据是了结实验有多便宜，而不是重要性。最后一栏是检验一个问题提得好不好的试金石——其中三个用一个节点和一周时间就能着手。

问题	答案的判准	了结它的实验
已发表的算法排名是后端的假象吗？（节 26.6）	一个能在更换推理后端后存活的排名，并报告对数概率差的分布	在两套推理引擎上、以完全相同的权重重跑一项已发表的消融；比较次序，而不是分数
优化 π_{inf} 而不是 $\pi_{\theta_{\text{old}}}$ 会带来任何差别吗？（节 26.6）	当两个策略都在部署配置下评测时，测到一个差异，或者测到一个零结果	从同一个检查点出发、算力匹配的两次运行，标准目标对阵面向推理的目标
评论家在智能体式的上下文长度下管用吗？（节 26.4）	一个值函数，其对同一固定状态出发的若干候选下一轮次的排序，以写明的幅度胜过随机基线	Kazemnejad et al. (2025) 的那项诊断，在一个工具环境中运行；几千次分支采样
一个合成任务能不能不解出来就被验证？（节 26.5）	在人工审计过的样本上，精确率与召回率超过「拿最强的模型去试着做一遍」，且成本更低	人工审计 500 个合成的智能体式任务；用该审计给候选的有效性谓词打分
有没有哪个奖励模型基准能预测 RL 的结果？（节 26.7）	能预测金标准奖励峰值(5.13)的高度与 KL 位置	五对或更多的奖励模型/策略，每一对都配一次完整的 RL 运行与留出的金标准评测
过度优化有没有停止规则？（节 26.7）	一条只用训练中可见之量的规则，落在金标准峰值的某个写明容差之内	与上面同样的那些运行，把候选规则事后对照观察到的峰值打分
难度能自动跟随移动中的策略吗？（节 26.5）	实测通过率在整整一次运行中保持在目标带内，并在生成算力匹配下胜过一个大的固定题池	生成器在环的运行对照固定题池的对照组；报告通过率的轨迹，而不是最终分数
RL 能在基座模型的有效支撑集之外播下质量吗？（节 26.2）	被播下的质量在够不着的分层上存活、没有更强的教师、两个家族，以及一个有效的梯度	受控预训练的试验台 (Shen et al., 2026a)，或支撑集可刻画的程序化生成任务
照现在这个提法，开放性可以做到吗？（节 26.5）	新颖性由这个环之外的东西凭证，并且被证明会跨轮累积	以语料或执行为依托的自博弈，在大 k 处用第 k 轮的策略去评测第 $k{+}1$ 轮的任务
有没有一个能凭证覆盖率的目标？（节 26.3）	一个旋钮，其设定能保证在写明的 k 处 pass@k 落在参考策略的 ϵ 之内	在 f 散度族内推导 (Wu et al., 2026)；在三个设定下用实验核验这份保证

第 VII 部分

附录

附录 A

数学背景

本附录汇集了正文中不加证明就直接使用的那些结果。它是一份参考资料，而不是一门课程：每一节都按本书所需要的形式陈述所需要的东西，并且只在证明既短又有启发性时才给出证明。想要一份像样论述的读者，应当查阅Appendix A.8中提到的那些来源。

A.1 分布之间的散度

设 p 与 q 是可数集 $\mathcal{Y}$ 上的分布，并在需要之处设 q 关于 p 绝对连续。

定义 A.1 (f 散度)**.** 对满足 $f(1)=0$ 的凸函数 $f:(0,\infty)\to\mathbb{R}$，

$$\mathbb{D}_f(p\,\|\,q)=\sum_{y\in\mathcal{Y}} q(y)\,f\!\left(\frac{p(y)}{q(y)}\right). \tag{A.1}$$

表 A.1列出了本书用到的那些实例。生成元 f 决定了由此得到的正则项在定性上一切重要的东西：它如何惩罚「在参考分布没有质量之处放置质量」，以及由此决定最优解是寻模的还是覆盖质量的。

命题 A.2 (寻模与覆盖质量)**.** 在 p 上最小化 $\mathbb{D}_{\mathrm{KL}}(p\,\|\,q)$，会因为 p 在 q 很小之处放置质量而惩罚它，却不会因为它在 q 很大之处漏掉质量而惩罚它。最小化 $\mathbb{D}_{\mathrm{KL}}(q\,\|\,p)$ 则恰好相反。

证明思路. 在 $\mathbb{D}_{\mathrm{KL}}(p\,\|\,q)=\sum p\log(p/q)$ 中，被求和项以 p 为权重，所以 $p=0$ 的区域无论 q 取什么都不贡献任何东西，而 $p>0$ 且 $q\approx 0$ 的区域则贡献无界。在 $\mathbb{D}_{\mathrm{KL}}(q\,\|\,p)=\sum q\log(q/p)$ 中权重是 q，所以 $q>0$ 且 $p\approx 0$ 的那些区域才是昂贵的。 □

表 A.1: 本书用到的 f 散度。最后一栏是对对齐而言要紧的那条性质：当 p 在 q 几乎不放质量的地方放上质量时，散度会怎么样。在那种情形下会爆掉的散度禁止支撑集扩张；不会爆掉的则允许。

散度	$f(z)$	出现在何处	局部 $p \gg q$ 时
反向 KL，$\mathbb{D}_{\mathrm{KL}}(p\|q)$	$z\log z$	标准的 RLHF 正则项	$\to\infty$
前向 KL，$\mathbb{D}_{\mathrm{KL}}(q\|p)$	$-\log z$	SFT 目标；f-DPO 变体	有限
全变差	$\frac{1}{2}\lvert z-1\rvert$	界定概率之差	以 1 为界
χ^2，$\chi^2(p\|q)$	$(z-1)^2$	集中度，节 8.7	更快地 $\to\infty$
Jensen–Shannon	$\frac{z}{2}\log\frac{2z}{z+1}+\frac{1}{2}\log\frac{2}{z+1}$	保多样性的变体	以 $\log 2$ 为界
混合 χ^2+KL	$\frac{1}{2}(z-1)^2+z\log z$	χPO，(8.21)	$\to\infty$

定理 A.2就是多样性坍缩（节 14.3）之所以是对齐目标的一个后果、而不是其优化过程中的一个缺陷的全部理由：标准目标用的是反向 KL，而反向 KL 对策略所抛弃的那些模式漠不关心。

A.1.1 几条有用的不等式

命题 A.3 (各散度之间的关系)**.** 对一切 p,q:

$$\mathbb{D}_{\mathrm{TV}}(p\|q)^2 \le \tfrac{1}{2}\mathbb{D}_{\mathrm{KL}}(p\|q) \qquad (\textit{Pinsker}), \tag{A.2}$$

$$\mathbb{D}_{\mathrm{KL}}(p\|q) \le \log(1+\chi^2(p\|q)) \le \chi^2(p\|q)\,, \tag{A.3}$$

$$\mathbb{D}_{\mathrm{KL}}(p\|q) \le 2\chi^2(p\|q)\,. \tag{A.4}$$

(A.3)的方向对节 8.7很要紧：界住 χ^2 就界住了 KL，但界住 KL 却不能界住 χ^2。既然出现在离线强化学习各种界中的集中度系数是 $C^\pi = 1+2\chi^2(\pi\|\pi_{\mathrm{ref}})$，那么一个 KL 约束就是一份严格弱于它看上去的凭证，而这正是 χPO 所利用的那个观察。

A.1.2 从样本估计 KL

在实践中，$\mathbb{D}_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}})$ 必须从采样得到的序列来估计。对一个样本 $y\sim\pi_\theta$ 记 $u = \log\frac{\pi_\theta(y)}{\pi_{\mathrm{ref}}(y)}$，在用的估计量有三个：

$$k_1 = u, \qquad k_2 = \tfrac{1}{2}u^2, \qquad k_3 = e^{-u}-1+u. \tag{A.5}$$

三者在相应的阶上期望都等于 $\mathbb{D}_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}})$：$k_1$ 严格无偏，但方差大且可能取负值；k_2 是二阶近似，恒非负，但有偏；k_3 既无偏又非负，这正是它成为通常选择的原因。k_3 无偏性的推导只有一行：$\mathbb{E}_{\pi_\theta}[e^{-u}]=\sum_y \pi_\theta(y)\frac{\pi_{\mathrm{ref}}(y)}{\pi_\theta(y)}=1$，于是 $\mathbb{E}[k_3]=1-1+\mathbb{E}[u]=\mathbb{D}_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}})$。

注记 A.4 (估计量不是惩罚项). 一个常见的混淆：把 k_3 用作一个损失项并不会让梯度变正确，因为除非把采样分布对 θ 的依赖也计入，否则 $\nabla_\theta k_3$ 并不是 $\nabla_\theta\mathbb{D}_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}})$ 的无偏估计。节 6.5讨论了梯度实际需要的是哪一项。

A.2 集中不等式与样本量

命题 A.5 (Hoeffding). 若 $X_1,\dots,X_n$ 相互独立且 $X_i\in[0,1]$，$\bar{X}$ 是它们的均值，则 $\mathbb{P}(|\bar{X}-\mathbb{E}[\bar{X}]|\geq t)\leq 2e^{-2nt^2}$。

对一个含 n 道二元计分题目的基准，定理 A.5给出的是粗糙的界，而精确的二项标准误 $\sqrt{\hat{\theta}(1-\hat{\theta})/n}$ 给出的是有用的那个。节 24.2推演了其后果，而这些后果令人不适：在 $n=30$ 时，十个百分点的差异在常规功效下与噪声区分不开。

命题 A.6 (重要性采样的方差与有效样本量). 设 $w(y)=p(y)/q(y)$，$\hat{\mu}=\frac{1}{n}\sum_i w(y_i)h(y_i)$，其中 $y_i\sim q$。那么 $\hat{\mu}$ 是 $\mathbb{E}_p[h]$ 的无偏估计，且

$$\mathbb{V}\mathrm{ar}[\hat{\mu}]=\frac{1}{n}\Big(\mathbb{E}_q[w^2h^2]-\mathbb{E}_p[h]^2\Big). \tag{A.6}$$

方差有限仅当 $\mathbb{E}_q[w^2h^2]<\infty$，而只要 q 的尾比 p 更轻，这一条就不成立——而这是常态，不是什么病态。标准的诊断量是有效样本量 $n_{\mathrm{eff}}=(\sum_i w_i)^2/\sum_i w_i^2$。

定理 A.6正是本书中每一项异策略修正都是截断的原因：未经截断的权重，恰恰在需要这项修正的那个区制里具有无界的方差。第22章把这一点讲得很具体。

A.3 凸分析与 Gibbs 变分原理

本书中用得最多的单个结果，是 KL 正则化问题的闭式解。值得把它单独干净地陈述一遍。

定理 A.7 (Gibbs 变分原理)**.** 设 q 是 $\mathcal{Y}$ 上的一个分布，$r : \mathcal{Y} \to \mathbb{R}$ 有界。那么

$$\max_{p} \left[\mathop{\mathbb{E}}_{p}[r] - \beta \mathbb{D}_{\mathrm{KL}}(p \,\|\, q) \right] = \beta \log \sum_{y} q(y) e^{r(y)/\beta}, \tag{A.7}$$

且最大值在 $p^\star(y) \propto q(y) e^{r(y)/\beta}$ 处唯一取到。

证明. 对任意 p，记 $Z = \sum_y q(y) e^{r(y)/\beta}$ 与 $p^\star = q e^{r/\beta}/Z$。那么

$$\begin{aligned} \mathop{\mathbb{E}}_{p}[r] - \beta \mathbb{D}_{\mathrm{KL}}(p \,\|\, q) &= \sum_{y} p(y) \Big(r(y) - \beta \log \frac{p(y)}{q(y)} \Big) \\ &= -\beta \sum_{y} p(y) \log \frac{p(y)}{q(y) e^{r(y)/\beta}} \\ &= -\beta \mathbb{D}_{\mathrm{KL}}(p \,\|\, p^\star) + \beta \log Z, \end{aligned} \tag{A.8}$$

其中最后一步用到了 $q e^{r/\beta} = Z p^\star$。由于 $\mathbb{D}_{\mathrm{KL}}(p \,\|\, p^\star) \geq 0$，且等号成立当且仅当 $p = p^\star$，结论随之成立。 □

有三条推论被反复用到。

推论 A.8 (支撑集保持)**.** 对每一个有界的 r 与每一个 $\beta > 0$，只要 $q(y) = 0$，就有 $p^\star(y) = 0$。奖励造不出支撑集。

推论 A.9 (可逆性)**.** 给定 q 与 β，映射 $r \mapsto p^\star$ 在相差一个可加常数的意义下是单射，其逆为 $r(y) = \beta \log \frac{p^\star(y)}{q(y)} + \mathit{const}$。

推论 A.10 (作为基线的对数配分函数)**.** (A.7)的右端是该状态的最优软值，而 $\beta \log Z$ 是相应策略梯度的方差最小的可加基线。

定理 A.8是第14章中「激发对习得」之争的形式化内容；定理 A.9是节 8.1的形式化内容；定理 A.10在第6章以及 SPPO 的推导（节 9.5）中反复出现。

A.4 策略梯度与得分函数

命题 A.11 (得分函数恒等式)**.** 对任意参数化分布 π_θ 与任意不依赖于 θ 的 h，

$$\nabla_\theta \mathop{\mathbb{E}}_{y \sim \pi_\theta}[h(y)] = \mathop{\mathbb{E}}_{y \sim \pi_\theta}[h(y) \nabla_\theta \log \pi_\theta(y)]. \tag{A.9}$$

特别地 $\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(y)] = 0$，正是这一点许可我们减去任何与 y 无关的基线而不引入偏差。

命题 A.12 (最优的常数基线). 使 $\mathbb{E}[(h - b)\nabla_\theta \log \pi_\theta]$ 的方差最小的常数 b 是 $b^\star = \mathbb{E}[h\|\nabla_\theta \log \pi_\theta\|^2] / \mathbb{E}[\|\nabla_\theta \log \pi_\theta\|^2]$，即 h 的一个以梯度范数为权的均值。实践中使用的近似是不加权的均值 $\mathbb{E}[h]$，而这正是组相对方法所估计的东西。

命题 A.13 (留一法是无偏的；组均值不是). 在 G 个样本下取 $b_i = \frac{1}{G-1}\sum_{j\neq i} h(y^{(j)})$，则估计量 $\frac{1}{G}\sum_i (h(y^{(i)}) - b_i)\nabla_\theta \log \pi_\theta(y^{(i)})$ 是无偏的。改用 $b = \frac{1}{G}\sum_j h(y^{(j)})$ 会引入一个 $O(1/G)$ 的偏差，因为 b 依赖于 $y^{(i)}$。

定理 A.13是第11章中 RLOO 与 GRPO 之比较的技术内容，而这一偏差之所以被容忍，是因为它随 G 缩小，而方差的削减不会。

A.5 指数族与 softmax 参数化

语言模型的下一词元分布 $\pi_\theta(v \mid h) = \mathrm{softmax}(z(h))_v$ 关于 logits 构成一个指数族，它的两条性质在全书中被不加说明地使用。

第一，softmax 对「给所有 logits 加上同一个常数」保持不变，所以这个参数化在每个位置上都多出了一维。第二，对数似然关于 logits 的梯度是 $\nabla_z \log \pi_\theta(v \mid h) = e_v - \pi_\theta(\cdot \mid h)$，所以在词元 v 上的一个正梯度，必然按其余每个词元的当前概率成比例地把质量从它们身上拿走。这就是似然位移（节 8.5）在机制上的来源：压低一个词元，会依照当前分布的形状与共享的表示来重新分配它的质量，而不是依照目标函数所规定的任何东西。

命题 A.14 (熵与温度). 对 $\pi_\theta^{\mathcal{T}}(v) \propto \exp(z_v/\mathcal{T})$，熵 $\mathbb{H}\big[\pi_\theta^{\mathcal{T}}\big]$ 关于 $\mathcal{T}$ 单调不减，$\mathcal{T} \to 0$ 给出 *argmax*，$\mathcal{T} \to \infty$ 给出均匀分布。因此，在 $\mathcal{T} \neq 1$ 下采样而在 $\mathcal{T} = 1$ 下打分，算出来的是两个不同分布之间的比值，而这正是节 22.7所讲的那个错误。

A.6 Bradley–Terry 与成对比较

定义 A.15 (Bradley–Terry 模型). 项 $1, \dots, m$ 具有正的强度 λ_i，且 $\mathbb{P}(i \succ j) = \lambda_i/(\lambda_i + \lambda_j)$。记 $\lambda_i = e^{r_i}$ 即得 $\mathbb{P}(i \succ j) = \sigma(r_i - r_j)$。

关于定理 A.15有两个事实与对齐文献有关。强度只在相差一个公共乘性常数的意义下可识别，因此奖励也只在相差一个可加常数的意义下可识别——这正是配分函数在 节 8.1.2中被约掉的原因，也是一个奖励模型的绝对尺度没有意义的原因。此外，该模型蕴含所诱导之序的可传递性，所以一个具有 $a \succ b \succ c \succ a$ 的偏好神谕，无法用任何强度赋值来表示。节 9.5接着讨论了它的后果。

命题 A.16 (logistic 联系函数是无界的)**.** $-\log\sigma(u)$ 严格递减，当 $u \to \infty$ 时 $-\log\sigma(u) \to 0$，且对一切有限的 u，其导数 $-\sigma(-u) \neq 0$。因此在线性可分的比较数据上，最小化 $-\mathbb{E}\log\sigma(u_\theta)$ 会把 u_θ 推向 ∞。

定理 A.16是「为什么 DPO 会过度优化，而 IPO 的平方损失与 SLiC 的合页损失不会」这件事的形式化陈述（表 8.1）。

A.7 序统计量与 best-of-n

命题 A.17 (n 个之中最优者的分布)**.** 设 $y_1, \dots, y_n \overset{i.i.d.}{\sim} q$，$y^\star = \arg\max_i r(y_i)$，其中 r 诱导出一个以 F 为 *CDF* 的连续得分分布。那么被选中样本的得分具有 *CDF* F^n，而在回复上诱导出的分布是 $p_n(y) = n\,q(y)F(r(y))^{n-1}$。

推论 A.18 (best-of-n 的 KL 代价)**.** $\mathbb{D}_{\mathrm{KL}}(p_n \,\|\, q) = \log n - \frac{n-1}{n}$ 奈特，与 q 无关，也与 r 无关。

定理 A.18正是让「奖励对 KL」图的横轴在 best-of-n 与一个训练好的策略之间可比的东西，因而也正是让过度优化的标度律可以解读的东西。节 10.4.1给出了推导，并讨论了它所需要的连续性假设。

命题 A.19 (无偏的 pass@k)**.** 每道题抽取 $n \geq k$ 个样本，设 c 为其中正确的个数。那么

$$\widehat{\mathit{pass@k}} = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \tag{A.10}$$

是「k 个独立样本中至少有一个正确」这一概率的无偏估计量。代入式估计量 $1-(1-c/n)^k$ 在 $k > 1$ 时向下有偏。

A.8 到哪里读更多

关于凸分析与定理 A.7的变分原理, Boyd 与 Vandenberghe 是标准参考书, 而 Cover 与 Thomas 的第 3 章覆盖了信息论那一侧，包括定理 A.3。关于策略梯度的材料，Sutton 与 Barto 在 定理 A.11 and A.12被发明出来的那个脉络里展开了它们，而那里关于方差削减的讨论至今仍是能找到的最清楚的一份。关于重要性采样与 定理 A.6，Owen 的 *Monte Carlo* 讲义既严谨又好读，而第22章中的那些截断策略出自那批文献，而不是出自深度学习。关于成对比较，David 关于成对比较方法的专著比对齐文献早了五十年，并且回答了后者所提出的多数问题。

附录 B

扩展推导

正文只陈述结论，并给出那些承载着概念分量的步骤。本附录则把那些推导完整做完：它们太长，放进章节里会打断行文，但读者应当有机会亲自核验。每一节都自成一体，并指明该结论在书中被用到的地方。

B.1 逐词元的策略梯度

用于第6章。

设 $\pi_\theta(y\mid x) = \prod_{t=1}^{T}\pi_\theta(y_t\mid x, y_{<t})$，并设 $r(x,y)$ 是终局奖励。目标为 $\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\,\mathbb{E}_{y\sim\pi_\theta(\cdot\mid x)}[r(x,y)]$。

推导：从序列级得分函数到逐词元的信用分配

在序列层面上应用定理 A.11：

$$\nabla_\theta \mathcal{J} = \mathop{\mathbb{E}}_{x}\,\mathop{\mathbb{E}}_{y\sim\pi_\theta}\big[r(x,y)\,\nabla_\theta \log \pi_\theta(y\mid x)\big] = \mathop{\mathbb{E}}_{x}\,\mathop{\mathbb{E}}_{y\sim\pi_\theta}\Big[r(x,y)\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(y_t\mid x, y_{<t})\Big]. \tag{B.1}$$

式 (B.1)是正确的，也是无用的：同一个标量乘在每个词元的得分上，于是估计量的方差随 T 增长，而且根本没有做信用分配。两步化简可以补救它。

因果性。一个词元不可能影响在它之前就已收到的奖励。形式地说，对 $s < t$，项 $\mathbb{E}[r_s \nabla_\theta \log \pi_\theta(y_t\mid\cdot)]$ 可分解为 $\mathbb{E}_{y_{<t}}[r_s\ \mathbb{E}_{y_t}[\nabla_\theta \log \pi_\theta(y_t\mid\cdot)]] = 0$，因为内层期望消失（定理 A.11）。在只有终局奖励时没有什么可去掉，但在 节 7.2的整形奖励情形下，这一步把 r 换成了剩余回报 $G_t = \sum_{s\ge t}\gamma^{s-t} r_s$。

基线。对任何关于状态 $s_t = (x, y_{<t})$ 可测的 b，

$$\mathop{\mathbb{E}}_{y_t \sim \pi_\theta(\cdot \mid s_t)} [b(s_t) \nabla_\theta \log \pi_\theta(y_t \mid s_t)] = b(s_t) \nabla_\theta \sum_{v \in \mathcal{V}} \pi_\theta(v \mid s_t) = b(s_t) \nabla_\theta 1 = 0, \quad \text{(B.2)}$$

因此减去 $b(s_t)$ 不会破坏梯度的无偏性。取 $b = V^{\pi_\theta}(s_t)$ 就给出优势形式

$$\nabla_\theta \mathcal{J} = \mathbb{E}\Big[\sum_{t=1}^{T} A^{\pi_\theta}(s_t, y_t) \nabla_\theta \log \pi_\theta(y_t \mid s_t)\Big]. \quad \text{(B.3)}$$

注记 B.1 (语言模型这一情形为何不寻常). (B.3)在此处有两点特殊。一是给定所发出的词元后转移是确定性的，因此 $Q^{\pi_\theta}(s_t, y_t) = V^{\pi_\theta}(s_{t+1})$ 精确成立；二是状态就是整个前缀，因此 V^{π_θ} 必须在一个大小为 $|\mathcal{V}|^t$ 的空间上估计。第一点使得我们可以通过重置到 s_t 再重新采样来得到蒙特卡洛的前缀值（节 13.4）；第二点则解释了为什么评论家难以拟合，以及为什么组基线如此有吸引力。

B.2 广义优势估计

用于第6章与节 7.2。

定义时序差分残差 $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$。$k$ 步优势估计量为 $A_t^{(k)} = \sum_{i=0}^{k-1} \gamma^i \delta_{t+i}$，望远镜求和后化为 $\sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k}) - V(s_t)$。随着 k 增大，估计量用到更多真实奖励、更少评论家，也就是以偏差换方差。指数加权平均

$$A_t^{\mathrm{GAE}} = (1-\lambda) \sum_{k=1}^{\infty} \lambda^{k-1} A_t^{(k)} = \sum_{i=0}^{\infty} (\gamma\lambda)^i \delta_{t+i} \quad \text{(B.4)}$$

在 $\lambda = 0$（纯一步，方差低，但被评论家的误差带偏）与 $\lambda = 1$（蒙特卡洛，在评论家无偏的前提下无偏，但方差高）之间插值。(B.4)中的第二个等号由交换两重求和并利用 $\sum_{k>i} (1-\lambda)\lambda^{k-1} = \lambda^i$ 得到。

推导: 为什么只有终局奖励时 GAE 会退化

设 $t < T$ 时 $r_t = 0$ 而 $r_T = r$，并取 $\gamma = 1$。于是对 $t < T$ 有 $\delta_t = V(s_{t+1}) - V(s_t)$，因此

$$A_t^{\mathrm{GAE}} = \sum_{i \geq 0} \lambda^i (V(s_{t+i+1}) - V(s_{t+i})), \quad \text{(B.5)}$$

它是评论家差分的 λ 加权和，在到达时域末端之前完全不含任何奖励信息。如果评论家沿轨迹还近似为常数——而在只有终局奖励上训练出来的评论家往往正是如此——那么每个 δ_t 都接近零，并且对所有 t, t' 都有 $A_t^{\mathrm{GAE}} \approx A_{t'}^{\mathrm{GAE}}$。此时 λ 这个旋钮是失效的，GAE 已经悄悄退化成把同一个标量广播出去。这就是为什么节 7.2 的逐词元 KL 整形在结构上而非表面上重要：它让奖励变得稠密，而这正是恢复 λ 有意义作用的前提。

B.3 从信赖域到截断

用于第6章。

信赖域的论证起点是性能差恒等式：对任意两个策略，

$$\mathcal{J}(\pi') - \mathcal{J}(\pi) = \mathop{\mathbb{E}}_{s\sim d^{\pi'}} \mathop{\mathbb{E}}_{a\sim\pi'(\cdot\,|\,s)} \big[A^{\pi}(s,a)\big], \tag{B.6}$$

其中 $d^{\pi'}$ 是 π' 的状态访问分布。右端不可直接使用，因为它需要来自 π' 的样本，而 π' 正是我们要选的那个策略。把 $d^{\pi'}$ 换成 d^{π} 便得到替代目标

$$L_{\pi}(\pi') = \mathop{\mathbb{E}}_{s\sim d^{\pi}} \mathop{\mathbb{E}}_{a\sim\pi(\cdot\,|\,s)} \Big[\frac{\pi'(a\,|\,s)}{\pi(a\,|\,s)} A^{\pi}(s,a)\Big], \tag{B.7}$$

其误差被一个正比于 $\max_s \mathbb{D}_{\mathrm{TV}}(\pi'(\cdot\,|\,s)\,\|\,\pi(\cdot\,|\,s))$ 的项所界定，因而（由 Pinsker 不等式，见(A.2)）也被一个 $\sqrt{\mathbb{D}_{\mathrm{KL}}}$ 量级的项所界定。在 KL 约束下最大化(B.7)就是 TRPO；正是这个约束使得用 d^{π} 代替 $d^{\pi'}$ 变得合理。

推导: 截断作为一阶信赖域

PPO 把这个约束换成了对目标本身的修改。记 $\rho = \pi'(a\,|\,s)/\pi(a\,|\,s)$，并考虑

$$L^{\mathrm{clip}} = \mathbb{E}\Big[\min\big(\rho A,\ \mathrm{clip}(\rho,\,1-\varepsilon,\,1+\varepsilon)\,A\big)\Big]. \tag{B.8}$$

分两种情形考察。若 $A>0$，一旦 $\rho > 1+\varepsilon$，min 就选中被截断的那一项，此时目标关于 θ 是常数，其梯度恰好为零。若 $A<0$，一旦 $\rho < 1-\varepsilon$，min 同样选中被截断项，同样给出零梯度。在两种情形下，其效果都是在超过某个阈值之后，取消把 ρ 继续沿优势所指方向推动的激励。要注意它没有做什么：它并不能阻止 ρ 在这一步开始时就远在区间之外，因为它只是把梯度置零；而一个在第一次内层迭代就已落到区间之外的词元，在这次更新剩下的过程里将完全得不到梯度。这

> 就是节 12.5.2所说的梯度消灭病理，它是 (B.8)这一机制的直接后果，而不是某种实现层面的意外。

B.4 从留一法 REINFORCE 导出 GRPO

用于第*11*章。

取 G 条回复 $y^{(1)},\dots,y^{(G)} \sim \pi_{\theta_{\text{old}}}(\cdot\,|\,x)$，其终局奖励为 r_i。定理 A.13的留一法估计量给出

$$\hat{g}_{\text{RLOO}} = \frac{1}{G}\sum_{i=1}^{G}\Big(r_i - \frac{1}{G-1}\sum_{j\neq i} r_j\Big)\nabla_\theta \log \pi_\theta(y^{(i)}\,|\,x), \tag{B.9}$$

它是无偏的。GRPO 的估计量在三处与它不同，而每一处都是一次有意为之的权衡。

基线取的是组均值，而不是留一均值。记 $\mu = \frac{1}{G}\sum_j r_j$，则有 $r_i - \frac{1}{G-1}\sum_{j\neq i} r_j = \frac{G}{G-1}(r_i - \mu)$，所以两者只差一个常数因子 $G/(G-1)$——这是一个 $O(1/G)$ 的重新缩放，而 Adam 的逐坐标归一化在很大程度上把它吸收掉了。这一步几乎是免费的。

除以组标准差。取 $A_i = (r_i - \mu)/\sigma$，其中 $\sigma^2 = \frac{1}{G}\sum_j (r_j - \mu)^2$，这引入了实实在在的偏差，因为 σ 依赖于 r_i。在二元奖励、经验成功率为 $\hat{p}$ 时有 $\sigma = \sqrt{\hat{p}(1-\hat{p})}$，于是每条提示的梯度被 $1/\sqrt{\hat{p}(1-\hat{p})}$ 缩放：几乎总能做对或几乎从不做对的提示反而被放大。这正是节 12.3所说的难度偏差。

除以回复长度。$1/\big|y^{(i)}\big|$ 这个因子使得每条序列的损失成为词元上的平均而不是求和，这就是节 11.3.1所讨论的依赖长度的步长。

B.5 截断重要性采样：偏差与方差

用于第*22*章。

设 $w = p/q$ 是重要性权重，$\bar{w} = \min(w, C)$ 是它的截断。对满足 $|h| \le H$ 的有界 h：

命题 B.2 (截断偏差)**.**

$$\Big|\mathop{\mathbb{E}}_{q}[\bar{w}h] - \mathop{\mathbb{E}}_{p}[h]\Big| = \Big|\mathop{\mathbb{E}}_{q}[(\bar{w}-w)h]\Big| \le H\ \mathop{\mathbb{E}}_{q}[(w-C)\mathbb{1}[w>C]] = H\ \mathop{\mathbb{E}}_{p}\Big[\Big(1-\frac{C}{w}\Big)\mathbb{1}[w>C]\Big]. \tag{B.10}$$

(B.10)的右端是 p 放在「两个分布相差超过 C 倍」这一区域上的质量，并按超出的幅度加权。它恰好在尾部很轻时才小，而那正是本来无需截断的情形——所以这个偏

差是真实存在的，却也是自限的：截断造成的损害最大的时候，恰恰是未截断的估计量本来就不可用的时候。

命题 B.3 (方差有界)**.** $\mathbb{V}\mathrm{ar}_q[\bar{w}h] \le C^2H^2$，无条件成立。

定理 B.3正是这件事的全部意义所在。定理 A.6表明未截断的方差可以是无穷；截断以(B.10)中的偏差为代价把它变得有限，而阈值 C 以一个显式的速率在两者之间做交换。

推导: 为什么序列级截断对长序列更糟

对长度为 T、逐词元权重为 w_t 的序列，序列权重是 $W = \prod_t w_t$，故 $\log W = \sum_t \log w_t$。若逐词元的对数比值均值为 m、方差为 v，且彼此弱相关，则 $\log W$ 的均值为 Tm、标准差为 $\sqrt{Tv}$。即便 m 恰好为零，对任何固定的 $C > 1$，随着 T 增大 $\mathbb{P}(\log W > \log C)$ 都趋于 $1/2$：长序列中大约有一半的序列权重会被截断，此时截断所做的事情已与「削掉一条罕见的重尾」相去甚远。这就是第22章所说的「长度陷阱」，也是在 T 很大时应当采用词元级而非序列级截断的理由。

B.6 DPO 梯度及其两个结构性性质

用于第8章。

记 $h_\theta = \beta(\varrho_w - \varrho_l)$，$\varrho_y = \log \frac{\pi_\theta(y\,|\,x)}{\pi_{\mathrm{ref}}(y\,|\,x)}$，则损失为 $\mathcal{L} = -\log\sigma(h_\theta)$。利用 $\frac{d}{du}\log\sigma(u) = \sigma(-u)$：

$$\nabla_\theta \mathcal{L} = -\sigma(-h_\theta)\,\nabla_\theta h_\theta = -\beta\sigma(-h_\theta)\big(\nabla_\theta \log \pi_\theta(y^+\,|\,x) - \nabla_\theta \log \pi_\theta(y^-\,|\,x)\big). \tag{B.11}$$

注意 π_{ref} 除了通过 h_θ 的取值之外，已从梯度中消失：参考模型只影响权重 $\sigma(-h_\theta)$，不影响方向。这一点值得内化，因为它解释了为什么无参考方法能有相似的行为（定理 8.4），也解释了为什么一份陈旧的缓存 $\log \pi_{\mathrm{ref}}$ 带来的是一个微妙走偏的课程，而不是一个明显错误的梯度。

推导: 损失沿着一个使被选回复似然下降的方向保持不变

把局部扰动参数化为 $\log \pi_\theta(y^+) \mapsto \log \pi_\theta(y^+) - \delta_w$ 与 $\log \pi_\theta(y^-) \mapsto \log \pi_\theta(y^-) - \delta_l$。于是 $h_\theta \mapsto h_\theta + \beta(\delta_l - \delta_w)$，因此只要 $\delta_l > \delta_w$ 损失就下降，包括两者都为正的情况——也就是两个似然都在下降的情况。使损失下降的方向集合，是 (δ_w, δ_l) 空间中由直线 $\delta_l = \delta_w$ 界出的一个半空间，而其中被选回复似然也下降的那部分，

占据了它中一个象限大小的区域。目标函数中没有任何东西偏向被选回复似然上升的那一部分。在损失中加入 $-\lambda \log \pi_\theta(y^+ \mid x)$ 会添上一项 $+\lambda\delta_w$，它使这套几何发生倾斜并消除了这种不变性——这就是 NLL 锚点项的形式化内容。

B.7 为什么 χ^2 正则化能控制可集中性

用于节 *8.7*。

离线强化学习的界受单策略可集中性系数支配：

$$C^\pi := \mathop{\mathbb{E}}_{y\sim\pi_{\text{ref}}}\Big[\Big(\frac{\pi(y)}{\pi_{\text{ref}}(y)}\Big)^2\Big] = 1 + \chi^2(\pi \,\|\, \pi_{\text{ref}}), \tag{B.12}$$

其中第二个等号由展开 $\chi^2(\pi\,\|\,\pi_{\text{ref}}) = \mathbb{E}_{\pi_{\text{ref}}}[(\pi/\pi_{\text{ref}} - 1)^2]$ 得到。所以对 χ^2 的界字面上就是对可集中性的界，而对 KL 的界则不是：由 (A.3)，蕴含关系是从 χ^2 到 KL，反过来不成立。具体地，取 π_{ref} 为 $\{1,\dots,m\}$ 上的均匀分布，π 把 ϵ 的质量放在一个点上、把 $1-\epsilon$ 均匀铺开。那么在 ϵ 固定时，$m\to\infty$ 过程中 $\mathbb{D}_{\text{KL}}(\pi\,\|\,\pi_{\text{ref}})$ 保持有界，而 $\chi^2(\pi\,\|\,\pi_{\text{ref}})$ 却像 $\epsilon^2 m$ 一样增长。因此一个受 KL 约束的策略对参考策略的覆盖可以任意地差，而这正是 χPO 所补上的缺口：它改用混合生成函数 $f_{\chi_{\text{mix}}}(z) = \frac{1}{2}(z-1)^2 + z\log z$ 做正则化，并在 DPO 的联系函数中把 $\log(\cdot)$ 换成 $\phi = f'_{\chi_{\text{mix}}}$。

B.8 best-of-n 的 KL 代价

用于节 *10.4.1*与第*5*章。

假定分数 $r(y)$ 在 q 下具有连续分布，于是并列出现的概率为零，并令 $U = F(r(y))$，它在 q 下是 $[0,1]$ 上的均匀分布。best-of-n 选出 U 最大的那个样本，其密度为 nu^{n-1}。因此在被选中的样本处，诱导出的密度比为 $p_n/q = nU^{n-1}$，于是

$$\begin{aligned}\mathbb{D}_{\text{KL}}(p_n\,\|\,q) &= \mathop{\mathbb{E}}_{U\sim nu^{n-1}}[\log(nU^{n-1})] = \log n + (n-1)\int_0^1 (\log u)\, nu^{n-1}\,du \\ &= \log n + (n-1)\cdot\Big(-\frac{1}{n}\Big) = \log n - \frac{n-1}{n},\end{aligned} \tag{B.13}$$

其中用到 $\int_0^1 u^{n-1}\log u\,du = -1/n^2$。这个结果既不依赖 q 也不依赖 r，而这恰恰是它可以充当一根公共横轴的原因。

注记 B.4 (连续性假定并非无害). 在使用二元验证器时分数分布是离散原子的，并列是常态，(B.13)并不适用——在二元奖励下，best-of-n 是在所有正确样本中均匀地选取，

它相对 q 的 KL 取决于成功率。那些在可验证场景中引用 $\log n - (n-1)/n$ 的论文，引用的是一个前提并不成立的结论。

B.9 带逐轮成本的最优停止

用于第*16*章与节 *17.6*。

考虑一个可以再走一轮、也可以就此停止的智能体。设 V_k 为它在 k 轮之后停止时的期望终局奖励，并设每一轮的成本为 $c > 0$。从 k 处继续的价值是 $\mathbb{E}[V_{k+1}] - c$ 再加上「还能继续下去」的期权价值。在「每轮的改进量 $\Delta_k = \mathbb{E}[V_{k+1}] - V_k$ 关于 k 非增」（收益递减）这一简化假定下，最优策略是一条阈值规则：当 $\Delta_k > c$ 时继续，在第一个满足 $\Delta_k \le c$ 的 k 处停止。

有两个推论值得写出来。第一，最优轮数由边际信息量与价格之比决定，因此只要给这项资源 定价就足以控制轮数，在最优处不需要任何显式的轮数上限。第二，若 $c = 0$——这是常见情形，因为啰嗦在奖励上是中性的——那么任何 $\Delta_k > 0$ 都足以正当化继续下去，最优策略会一直走下去，直到改进量恰好为零。这就是「轮数与长度的爆炸是默认结果而非异常」这一说法的形式化表述。

附录 C

参考实现

本附录中的代码足够完整，可以当作规格说明来读；也足够简短，可以用眼睛检查。它们是为清晰而写的，不是为吞吐而写的：一个生产级实现会融合算子、切分张量、重算激活，而这些都不改变数学。凡是某一行代码承载了正文讨论过的某个决策，注释里都会点出对应的小节。

全书通篇的张量约定是：`B` 为序列批量大小，`T` 为补齐后的序列长度，`V` 为词表大小。`loss_mask` 在策略自己发出的位置上取 1，在提示词元、填充以及——在智能体式场景中——环境写入的词元上取 0 （节 16.4）。

C.1 逐词元的对数概率

其余一切都取决于把这一步做对，而两个常见错误分别是移位时差了一位，以及在低精度下累加 log-softmax。

代码 C.1: 最基本的原语。注意 `log_softmax` 之前的 `float()`：序列对数概率是上千项之和，也是 `bf16` 误差第一个会以虚假重要性比值形式显露出来的地方（第22章）。

```python
import torch
import torch.nn.functional as F

def token_logprobs(model, input_ids, attention_mask):
    """Returns per-position log p(input_ids[:, t] | input_ids[:, :t]) of shape
    [B, T-1], aligned so that index t-1 scores token t."""
    logits = model(input_ids=input_ids, attention_mask=attention_mask).logits
    logits = logits[:, :-1, :]                      # 丢掉最后一个位置
    labels = input_ids[:, 1:]                       # 左移一位
    logp = F.log_softmax(logits.float(), dim=-1)    # 用 fp32 累加
    return logp.gather(2, labels.unsqueeze(2)).squeeze(2)
```

```python

def sequence_logprob(model, input_ids, attention_mask, loss_mask):
    """Sum over response tokens only. loss_mask is [B, T] on the same indexing
    as input_ids; we use its [:, 1:] slice to match token_logprobs."""
    lp = token_logprobs(model, input_ids, attention_mask)
    return (lp * loss_mask[:, 1:]).sum(dim=-1)

def token_entropy(model, input_ids, attention_mask):
    """Per-position entropy of the predictive distribution; the diagnostic that
    detects entropy collapse before reward degrades (Sec. entropy-collapse)."""
    logits = model(input_ids=input_ids, attention_mask=attention_mask).logits[:, :-1]
    logp = F.log_softmax(logits.float(), dim=-1)
    return -(logp.exp() * logp).sum(dim=-1)
```

C.2 组相对优势

代码 C.2: 组相对优势估计，带上文献中意见不一的那四个开关。这里的默认值站在 Dr. GRPO/DAPO 那一侧：不除以标准差（节 12.3），并采用全局词元归一化（节 12.2）。

```python
def group_advantages(rewards, group_size, divide_by_std=False, eps=1e-6):
    """rewards: [B] where B = n_prompts * group_size, grouped contiguously.
    Returns [B] advantages."""
    r = rewards.view(-1, group_size)                        # [P, G]
    baseline = r.mean(dim=1, keepdim=True)                   # 带 O(1/G) 的偏差
    adv = r - baseline
    if divide_by_std:                                        # 放大过易/过难的
        adv = adv / (r.std(dim=1, keepdim=True) + eps)       # 提示：见 12.3 节
    return adv.reshape(-1)

def leave_one_out_advantages(rewards, group_size):
    """The unbiased alternative (Prop. app-loo). Differs from the above by the
    constant factor G/(G-1), which Adam largely absorbs."""
    r = rewards.view(-1, group_size)
    total = r.sum(dim=1, keepdim=True)
    loo = (total - r) / (group_size - 1)
    return (r - loo).reshape(-1)

def degenerate_mask(rewards, group_size):
    """True for groups with zero reward variance: exactly zero gradient, so the
    generation budget spent on them was wasted (Sec. dynamic-sampling)."""
    r = rewards.view(-1, group_size)
    return (r.max(dim=1).values == r.min(dim=1).values)
```

C.3 策略优化损失

代码 C.3: 一个损失函数即涵盖 PPO/GRPO 式截断、非对称截断（DAPO）、序列级比值（GSPO）以及截断权重修正（CISPO）。把它们写成同一份实现上的若干开关，是做一次诚实消融的唯一办法，因为这样填充与归一化逻辑在各条实验臂之间是可证明相同的。

```
def policy_loss(logp, logp_old, adv, loss_mask, mode="token",
                eps_low=0.2, eps_high=0.28, is_clip=4.0,
                normalizer="global"):
    """logp, logp_old, adv, loss_mask: [B, T-1] (adv broadcast over T).
    mode in {"token", "sequence", "truncated"}."""
    mask = loss_mask
    if mode == "sequence":                          # GSPO：每条序列一个比值
        n = mask.sum(-1).clamp(min=1)
        delta = ((logp - logp_old) * mask).sum(-1) / n      # 做长度归一化
        ratio = delta.exp().unsqueeze(-1).expand_as(logp)
    else:
        ratio = (logp - logp_old).exp()

    if mode == "truncated":                         # CISPO：截断权重，同时让
        w = ratio.clamp(max=is_clip).detach()    # 每个词元的梯度都活着
        per_token = -w * adv * logp
    else:                                           # PPO/DAPO：截断目标本身
        unclipped = ratio * adv
        clipped = ratio.clamp(1.0 - eps_low, 1.0 + eps_high) * adv
        per_token = -torch.min(unclipped, clipped)

    if normalizer == "global":                       # DAPO：词元级平均
        return (per_token * mask).sum() / mask.sum().clamp(min=1)
    else:                                            # GRPO：逐序列平均
        per_seq = (per_token * mask).sum(-1) / mask.sum(-1).clamp(min=1)
        return per_seq.mean()
```

代码 C.4: KL 项，带(A.5)中的三个估计量。`k3` 是默认选择，因为它无偏且非负；即便你用 `k3` 施加惩罚，也应当记录 `k1`，因为它的符号携带信息。

```
def kl_penalty(logp, logp_ref, loss_mask, estimator="k3"):
    u = (logp - logp_ref)                            # log pi_theta / pi_ref
    if estimator == "k1":
        k = u
    elif estimator == "k2":
        k = 0.5 * u.pow(2)
    else:                                            # k3：无偏且非负
        k = (-u).exp() - 1.0 + u
```

```
    return (k * loss_mask).sum() / loss_mask.sum().clamp(min=1)
```

C.4 面向采样失配的截断重要性采样

这里真正要紧的纪律，是把两个比值分开：一个在权重固定的前提下修正推理引擎与训练器之间的间隙，另一个则为当前这次更新实现信赖域。把它们混为一谈是最常见的错误。

代码 C.5: 与 PPO 截断并用的截断重要性采样。`logp_infer` 来自采样引擎自己报告的对数概率；`logp_old` 是训练器在同一组权重下的重算结果。二者之差本应接近于零，而事实并非如此（第22章）。

```
def tis_corrected_loss(logp, logp_old, logp_infer, adv, loss_mask,
                       eps=0.2, tis_clip=2.0):
    # (1) 在完全相同的权重下，修正引擎与训练器之间的间隙
    w = (logp_old - logp_infer).exp().clamp(max=tis_clip).detach()

    # (2) 本次更新的信赖域，照常计算
    ratio = (logp - logp_old).exp()
    obj = torch.min(ratio * adv, ratio.clamp(1 - eps, 1 + eps) * adv)

    per_token = -(w * obj)
    return (per_token * loss_mask).sum() / loss_mask.sum().clamp(min=1)

def mismatch_diagnostics(logp_old, logp_infer, loss_mask):
    """Run this before blaming your algorithm. A tiny mean with heavy tails is
    worse than a large mean."""
    d = ((logp_old - logp_infer) * loss_mask)
    n = loss_mask.sum().clamp(min=1)
    vals = d[loss_mask.bool()]
    w = vals.exp()
    return {
        "logp_gap_mean": d.sum() / n,
        "logp_gap_p99": vals.abs().quantile(0.99),
        "ess_frac": (w.sum() ** 2) / (w.pow(2).sum() * w.numel()).clamp(min=1e-9),
    }
```

C.5 直接偏好优化及其变体

代码 C.6: 用同一条打分路径实现 DPO、IPO 与 SimPO，并附上 表 8.5中的那些诊断量。`chosen_delta` 这个统计量是能检测出似然位移的那一个，也是多数实现偏偏省掉的那一个。

```
def daa_loss(policy, ref, batch, kind="dpo", beta=0.1, gamma=1.0,
             tau=0.1, nll_weight=0.2):
    ids, am, lm = batch["input_ids"], batch["attn_mask"], batch["loss_mask"]
    n = ids.shape[0] // 2                                  # [chosen; rejected]
    lengths = lm[:, 1:].sum(-1).clamp(min=1)

    pi = sequence_logprob(policy, ids, am, lm)
    if kind == "simpo":                                    # 无参考模型
        s = beta * pi / lengths
        margin = (s[:n] - s[n:]) - gamma
        loss = -F.logsigmoid(margin).mean()
        stats = {"margin": margin.mean()}
        return loss, stats

    with torch.no_grad():
        rf = sequence_logprob(ref, ids, am, lm)
    dw, dl = (pi[:n] - rf[:n]), (pi[n:] - rf[n:])
    if kind == "ipo":
        margin = dw - dl
        loss = ((margin - 1.0 / (2 * tau)) ** 2).mean()
    else:                                                  # dpo
        margin = beta * (dw - dl)
        loss = -F.logsigmoid(margin).mean()
    if nll_weight:                                         # 锚点项：命题 8.1
        loss = loss - nll_weight * (pi[:n] / lengths[:n]).mean()
    stats = {"margin": margin.mean(),
             "implicit_acc": (margin > 0).float().mean(),
             "chosen_delta": dw.mean(),                     # 绝不能变成负值
             "rejected_delta": dl.mean()}
    return loss, stats
```

C.6 多轮轨迹的损失掩码

这是智能体式后训练里每行代价最高的一段代码。环境词元必须被注意到，又必须不被 训练；这是两个不同的掩码，把它们搞混是标准的 bug。

代码 C.7: 为一条交错的轨迹构造两个掩码。那句断言不是装饰：一个无声的掩码错误会产出一次「训练模型去预测工具输出」的运行，而它看起来像是收敛慢，而不像是一个 bug。

```
def build_multiturn_masks(tokenizer, turns, max_len):
    """turns: list of (role, text) with role in {"prompt","action","observation"}.
```

```
    Returns input_ids, attention_mask (attend to everything real),
    loss_mask (train on policy-emitted tokens only)."""
    ids, loss = [], []
    for role, text in turns:
        piece = tokenizer.encode(text, add_special_tokens=False)
        ids.extend(piece)
        loss.extend([1 if role == "action" else 0] * len(piece))
    ids, loss = ids[:max_len], loss[:max_len]
    pad = max_len - len(ids)
    input_ids = ids + [tokenizer.pad_token_id] * pad
    attention_mask = [1] * len(ids) + [0] * pad      # 要注意到观测
    loss_mask = loss + [0] * pad                      # 但绝不在其上训练
    assert sum(loss_mask) > 0, "trajectory has no trainable action tokens"
    return (torch.tensor(input_ids), torch.tensor(attention_mask),
            torch.tensor(loss_mask))
```

C.7 一个最小但完整的训练步

代码 C.8: 完整的循环，连同各项检查。这里省略掉的一切都是系统层面的问题（第21章）；这里省略掉的东西没有一样会改变梯度。

```
def train_step(policy, ref, engine, optimizer, prompts, verifier,
               group_size=8, beta_kl=0.0, inner_epochs=1):
    # 1. 从推理引擎采样
    seqs = engine.generate(prompts, n=group_size, temperature=1.0, top_p=1.0)

    # 2. 打奖励，然后丢掉不可能贡献梯度的那些组
    rewards = torch.tensor([verifier(s.prompt, s.text) for s in seqs])
    keep = ~degenerate_mask(rewards, group_size).repeat_interleave(group_size)
    if keep.sum() == 0:
        return {"skipped": 1.0}                       # 课程与能力不匹配
    batch = collate([s for s, k in zip(seqs, keep) if k])
    rewards, adv_flat = rewards[keep], group_advantages(rewards, group_size)[keep]

    # 3. 用训练器（而不是引擎）重算旧的对数概率
    with torch.no_grad():
        logp_old = token_logprobs(policy, batch.ids, batch.attn)
        logp_ref = (token_logprobs(ref, batch.ids, batch.attn)
                    if beta_kl > 0 else None)
    logp_infer = batch.engine_logprobs                # 用于失配审计

    stats = mismatch_diagnostics(logp_old, logp_infer, batch.loss_mask[:, 1:])
    adv = adv_flat.unsqueeze(-1).expand_as(logp_old)
```

```
    # 4. 在同一批采样轨迹上做内层迭代
    for _ in range(inner_epochs):
        logp = token_logprobs(policy, batch.ids, batch.attn)
        loss = tis_corrected_loss(logp, logp_old, logp_infer, adv,
                                  batch.loss_mask[:, 1:])
        if beta_kl > 0:
            loss = loss + beta_kl * kl_penalty(logp, logp_ref,
                                               batch.loss_mask[:, 1:])
        loss.backward()
        gnorm = torch.nn.utils.clip_grad_norm_(policy.parameters(), 1.0)
        optimizer.step(); optimizer.zero_grad(set_to_none=True)

    # 5. 把权重同步给引擎，然后把一切都记录下来
    engine.load_weights(policy)
    stats.update({"reward": rewards.mean(), "grad_norm": gnorm,
                  "kept_frac": keep.float().mean(),
                  "len": batch.loss_mask.sum(-1).float().mean()})
    return stats
```

C.8 一份诊断契约

本书中的多数失效在损失曲线上都看不见，而在下面某一个量上一目了然。把它们全部记录下来只花掉步时间的百分之几，却是一个后训练项目里回报最高的单项工程决策。

C.9 可复现性检查清单

工程实践：凡是打算发表的运行，事前该钉住什么

把数据顺序、dropout 与生成这三处的随机种子分别钉住，并把三者都记录下来。钉住推理引擎版本、注意力后端、张量并行度与流水并行度，以及语言模型头的数值精度——这些当中每一项都会在权重固定的情况下改变 π_{inf}（第22章）。钉住分词器与对话模板；模板一变，每一个损失掩码都会无声地跟着变。钉住验证器或评分器的版本及其超时设置，并记录采样轨迹中触发超时的比例，因为一个慢的评分器会把正确答案变成零奖励。逐字记录答案抽取规则：一次正则表达式的改动，其带来的实测提升常常超过一次算法改动（第24章）。最后，为每条采样

表 C.1: 最小记录契约。只记录奖励与损失的一次运行，事后是无法调试的，因为 第12章、第22章与 第8章中的各种失效特征，只有通过这些量才能彼此区分。

量	在何处计算	它能识别出的失效
平均奖励	代码 C.8	单看它什么也说明不了；必要而不充分
词元熵	代码 C.1	熵坍缩，早于奖励退化
$\mathbb{D}_{\mathrm{KL}}(\pi_\theta \Vert \pi_{\mathrm{ref}})$，$k_1$ 与 k_3	代码 C.4	漂移；k_1 的符号给出方向
截断比例，上侧与下侧	代码 C.3	步长过大；梯度消灭
比值的 1/99 分位数	代码 C.3	重尾的离策略程度
对数概率间隙的均值与 p99	代码 C.5	训练—推理失配
有效样本量	代码 C.5	重要性权重是否还可用
退化组比例	代码 C.2	课程不匹配；预算浪费
回复长度的均值与 p99	代码 C.8	长度爆炸；截断率
梯度范数	代码 C.8	不稳定，在发散前一步
被选/被弃回复的对数比值	代码 C.6	似然位移
留出集上的生成质量	在外部评测	上面所有量都漏掉的东西

> 轨迹存下足够的元数据，以便精确重放任意一条轨迹；并且在相信那个数字之前，真的去读一读奖励最高的那些采样轨迹（节 5.8.4）。

附录 D

术语表

这里的定义都按本书中使用这些术语的方式给出。凡是某个术语在文献中仍有争议——而其中几个最重要的术语正是如此——条目会直言这一点，而不是假装存在一个并不存在的共识。交叉引用指向的是概念被真正展开的地方，而不是仅被提到的地方。

A

优势（advantage） 一个动作的奖励相对于某个不依赖该动作的基线的差值。减去基线既不破坏策略梯度的无偏性，又能降低其方差（节 2.7）。在组相对方法中，基线是同一条提示下同胞样本的平均奖励（第12章）。

智能体式 RL（agentic RL） 策略在多个轮次中与一个外部环境交互的后训练，因而它自己的输出会改变其未来输入的分布（第16章）。其区别性特征并不是工具使用本身，而是那个闭环：模型的动作决定了下一个观测。

对齐（alignment） 调整一个预训练模型，使其行为符合某个规格说明，无论该规格是以人类偏好数据、一部书面「宪法」，还是一组程序化检查的形式表达出来的。在第 II 部分中取其狭义，指从偏好反馈中学习。

对齐税（alignment tax） 在留出基准上测得的、对齐目标并未针对的那些能力上的退化（节 3.6）。它常常是正则项过强或提示分布多样性不足的产物，而不是一种不可避免的权衡。

非对称截断（asymmetric clipping） 在 PPO 式目标中让上侧截断阈值大于下侧，使低概率词元仍留有增长的余地。作为「clip-higher」由 DAPO 引入，用以抵抗熵坍缩（节 12.11）。

B

best-of-n 采样 n 条回复，并返回评分器排名最高的那一条。它既是测试时计算的一个基线，又是一种蒸馏数据来源，也是「相对基础策略的 KL 代价有解析已知结果」这类方法的标准范例（节 10.4，Appendix B.8）。

Bradley–Terry 模型 两两比较模型 $\mathbb{P}(y^+ \succ y^-) = \sigma(r(y^+) - r(y^-))$，它把偏好数据变成一个奖励回归问题（节 5.2）。它的可传递性假定，正是博弈论式对齐所抛弃的东西（第9章）。

C

思维链（chain of thought） 在给出最终答案之前发出的中间词元。在推理 RL 中，这些词元只通过最终答案上的奖励被优化，而这正是它们可以自由地变得不适合人类阅读的原因（节 25.8）。

CISPO 一种策略优化变体：它截断重要性权重，但保留每个词元的对数概率梯度不受影响，使得比值很大的词元依然有所贡献（节 12.5.2）。可与 PPO 截断相对照，后者把被截断词元的梯度置零。

可集中性（concentrability） 一个系数，用于界定候选策略的状态—动作分布能在多大程度上超出数据分布。有限的可集中性是离线学习在统计上可行的前提；悲观主义与 χ^2 正则化是在覆盖较弱时获得保证的两条标准途径（节 8.7）。

Constitutional AI 用模型自己针对一套书面原则做出的批评与修订，来替代人类偏好标注（节 9.3）。被审计的对象由此变成那些原则，而不是标注。

信用分配（credit assignment） 把轨迹级的结果归因到产生它的那些具体词元、轮次或工具调用上（第13章）。这是长时域 RL 的核心困难，也是只依赖结果的奖励随时域增长而效果变差的原因。

D

DAPO 对 GRPO 的四项修改的合集——非对称截断、动态采样、词元级损失归一化，以及柔性长度惩罚——其所报告的收益来自这些修改的组合，而不是其中任何单独一项（节 12.11）。

直接对齐算法（direct alignment algorithm, DAA） 任何在偏好数据上优化策略、既不拟合显式奖励模型、也不在训练过程中从策略采样的方法；DPO 是其原型（第8章）。

DPO 直接偏好优化：对 KL 正则化 RLHF 目标的闭式重参数化，它把奖励学习与策略优化合并成一个单一的分类损失（节 8.1）。在强假定下，它在最优点处与 RLHF 等价，而在偏离最优点处则与之有实质差别。

Dr. GRPO GRPO 的一个修正形式，去掉了除以标准差与逐序列的长度归一化，这两者都会引入偏差（节 12.3）。

动态采样（dynamic sampling） 丢弃那些采样奖励全都相同的提示组——因为这样的组贡献的梯度恰好为零——并重新采样以填满这一批（节 12.4）。

E

有效样本量（effective sample size） 用一个标量概括一组重要性权重退化到了什么程度，即 $(\sum_i w_i)^2/(n\sum_i w_i^2)$。它是离策略修正方面信息量最大的单一诊断量（Appendix C.4）。

熵坍缩（entropy collapse） RL 早期策略熵的急剧收缩，此后策略停止探索、奖励进入平台（节 12.6）。它远在奖励曲线走平之前就可诊断，这也是为什么熵应当出现在每一份记录契约里。

环境（environment） 策略所作用的那个不可微系统：一个解释器、一个浏览器、一个被模拟的操作系统、一个游戏，或者另一个模型（第19章）。实践中，它的延迟、不稳定性以及奖励语义主导了算法的选择。

G

GAE 广义优势估计：对 k 步时序差分残差做指数加权平均，通过 λ 在偏差与方差之间做交换（节 6.2.2，Appendix B.2）。

GRPO 组相对策略优化：用来自同一条提示的一组样本的经验平均奖励，取代学习得到的值函数（第12章）。相对 PPO 把显存减半，并把困难从值函数拟合转移到了组的构成上。

GSPO 一种变体，它在序列层面上定义重要性比值并做长度归一化，使得截断决策作用于整条回复而不是单个词元（节 12.5）。它对混合专家模型尤为相关，因为那里词元级比值的噪声最大。

I

重要性采样（importance sampling） 对从一个分布中抽取的样本重新加权，以估计另一个分布下的期望。在后训练中它出于两种不同的原因出现两次：跨梯度步复用采样轨迹，以及修正推理引擎与训练引擎之间的间隙（第22章）。

IPO 恒等偏好优化：把 DPO 的 log-sigmoid 联系函数换成围绕一个有限目标间隔的平方损失，从而在带噪声或确定性标签上界住把间隔推向无穷的激励（节 8.4）。

K

KL 正则化（KL regularization） 惩罚与参考策略之间的散度，其形式或是一个显式的损失项，或是参数化所隐含的后果。它服务于三个常被混为一谈的不同目的：防止奖励作弊、让采样分布留在奖励模型的训练支撑附近，以及保住预训练得到的能力（节 6.5）。

k_3 估计量 KL 散度的无偏、非负的蒙特卡洛估计量，即 $e^{-u}-1+u$，其中 $u = \log(\pi_\theta/\pi_{\text{ref}})$。做惩罚项时首选它；而 $k_1 = u$ 仍应记录，因为它的符号携带信息（节 2.4）。

L

长度偏差（length bias） 任何由目标函数而非任务本身所诱发的、对更长或更短输出的系统性偏好。它出现在用人类数据训练的奖励模型中、逐序列的损失归一化中，以及 DPO 的对数概率求和式打分中（节 5.5）。

似然位移（likelihood displacement） DPO 的一种病理：训练过程中被选回复的对数概率下降，而偏好间隔却在改善，原因是损失只约束了两者之差（节 8.5）。通过记录被选回复相对参考模型的对数比值即可检测；可用一个 SFT 锚点项或基于 CHES 的数据筛选来缓解。

损失掩码（loss mask） 用来挑出哪些位置参与损失的二值掩码。它与注意力掩码不同：环境观测必须被注意到，又必须不被训练（节 16.4）。

N

从人类反馈中做 Nash 学习（Nash learning from human feedback） 把对齐表述为求由一个偏好模型所定义的二人博弈的 Nash 均衡；当偏好不可传递、因而不存在 Bradley–Terry 奖励时，这一表述仍然是良定的（节 9.5）。

O

离策略（off-policy） 从由某个策略生成、而该策略并非正在被更新的那个策略的数据中学习。在后训练中，离策略的程度是一个连续谱，由每批采样轨迹上走多少个梯度步、以及推理引擎的权重有多陈旧共同控制（节 22.8）。

结果奖励模型（outcome reward model, ORM） 只对最终答案打分的评分器。可与**过程奖励模型（process reward model, PRM）**相对照，后者对中间步骤打分，因而提供更稠密、但也更容易被作弊的信号（节 5.7）。

过度优化（overoptimization） 代理奖励在提升，而该代理本应衡量的那个量却在退化。无论是 best-of-n 还是 RL，它都沿着一条以 $\sqrt{\mathbb{D}_{\mathrm{KL}}}$ 为自变量的特征曲线演进（节 5.8）。

P

pass@k k 个独立样本中至少有一个正确的概率。对基础模型而言它随采样预算上升，而在 RL 之后，它在大 k 处往往下降；这正是「RL 究竟是增添能力还是只是把能力削尖」这场争论的实证核心（第14章）。

PPO 近端策略优化：对信赖域方法的一种截断替代目标近似，也是最初 RLHF 流水线所依托的算法（节 6.3）。它需要一个学习得到的值函数，而这既是它的主要开销，也是它信用分配更稠密的来源。

后训练（post-training） 预训练之后对模型所做的一切：监督微调、偏好对齐、使用可验证奖励或基于模型的奖励的 RL、蒸馏，以及模型合并（第1章）。

R

奖励作弊（reward hacking） 通过奖励设计者本不希望的行为获得高奖励，无论是钻奖励模型的盲区、验证器的解析规则，还是环境的侧信道（节 5.8）。读一读奖励最高的那些采样轨迹，几乎总能把它找出来。

RLAIF 来自 AI 反馈的强化学习：用一个模型替代人类标注者，或用于标注偏好，或用作评判模型（节 9.3）。

RLHF 来自人类反馈的强化学习：由监督微调、在偏好比较上拟合奖励模型、以及 KL 正则化的策略优化这三个阶段构成的流水线（第7章）。

RLVR 带可验证奖励的强化学习：奖励来自一个程序化检查——一个单元测试、一次符号相等判定、一个证明检查器——而不是来自一个学习得到的模型（第11章）。没有可学习的奖励，就在模型层面消除了奖励作弊，同时把它转移到了验证器身上。

采样轨迹（rollout） 一条生成出来的轨迹，从一次性的回复到带工具调用的多轮智能体式片段都算。它既是算力核算的单位，也是调试的单位。

S

自博弈（self-play） 让一个模型与自己的副本对抗、互相批评，或为其副本出题，以此产生训练信号（节 9.3）。当存在验证器或某种不对称性来打破对称时它有效；否则就会退化。

SFT 监督微调：在示范数据上做极大似然训练（第3章）。它确定了参考策略、对话模板，以及后续所有 RL 探索所依托的支撑集，这使得它的数据构成比优化器设置更为要紧。

SimPO 一个无参考模型的直接对齐目标，使用长度归一化的平均对数概率并带一个目标间隔（节 8.4）。它省掉了参考模型的前向传播，代价是同时也省掉了那个界住漂移的正则项。

测试时计算（test-time compute） 在推理阶段花掉、用于改进单个答案的算力：更长的思维链、带聚合的并行采样、树搜索，或迭代修订（第10章）。它与训练算力沿一条可量化的前沿相互交换。

训练—推理失配（train–inference mismatch） 生成某条采样轨迹的推理引擎所给出的词元分布，与训练器在同一组权重下对同一分布的重算之间的差异（第22章）。由于算子核、批处理与精度各不相同，它在任何真实系统中都非零；它会悄无声息地把一个同策略算法变成离策略算法。

截断重要性采样（truncated importance sampling） 把重要性权重截到一个常数上限，以一份可控的偏差换取方差的大幅下降（节 22.4，Appendix B.5）。它是训练—推理失配的标准首选修复手段。

V

验证器（verifier） 一个把提示与回复映射为奖励（通常是二元的）的程序。它的解析规则、超时设置与失效模式都是奖励函数的一部分，无论设计时是否有意如此（节 11.6）。

参考文献

A-3PO. A-3po: Accelerating asynchronous llm training with staleness-aware proximal policy approximation. *arXiv preprint arXiv:2512.06547*, 2025. URL https://arxiv.org/abs/2512.06547.

Zelalem Abahana, David Evans, Satish Mahadevan Srinivasan, and Matjaz Gams. When RLHF fails: A mechanistic taxonomy of reward hacking, collapse, and evaluator gaming. *arXiv preprint arXiv:2606.03238*, 2026. arXiv:2606.03238.

Alexandru Agache, Marc Brooker, Andreea Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, and Diana-Maria Popa. Firecracker: Lightweight virtualization for serverless applications. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 419–434, 2020. The reference microVM design; sub-second boot with hardware-virtualization isolation.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *NeurIPS*, 2021. arXiv:2108.13264.

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint*, 2025. arXiv:2503.04697.

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In *EMNLP*, 2023. arXiv:2305.11860.

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *ACL*, 2021. arXiv:2012.13255.

Milad Aghajohari et al. The markovian thinker: Architecture-agnostic linear scaling of reasoning. *arXiv preprint arXiv:2510.06557*, 2025.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. *ACL*, 2024. arXiv:2402.14740.

Riad Akrour, Marc Schoenauer, and Michèle Sebag. Preference-based policy learning. In *ECML PKDD*, 2011.

Alibaba ROLL Team. ROLL: An efficient and user-friendly scaling library for reinforcement learning with large language models. https://github.com/alibaba/ROLL, 2025.

Norah Alzahrani, Hisham Abdullah Alyahya, Yazeed Alnumay, Sultan Alrashed, Shaykhah Alsubaie, Yousef Almushayqih, Faisal Mirza, Nouf Alotaibi, Nora Al-Twairesh, Areeb Alowisheq, M. Saiful Bari, and Haidar Khan. When benchmarks are targets: Revealing the sensitivity of large language model leaderboards. In *ACL*, 2024. arXiv:2402.01781.

Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned LLMs with simple adaptive attacks. *ICLR*, 2025a. arXiv:2404.02151.

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, J. Zico Kolter, Matt Fredrikson, Eric Winsor, Jerome Wynne, Yarin Gal, and Xander Davies. AgentHarm: A benchmark for measuring harmfulness of LLM agents. *ICLR*, 2025b. arXiv:2410.09024.

Cem Anil, Esin Durmus, Nina Panickssery, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, et al. Many-shot jailbreaking. In *NeurIPS*, 2024.

Anonymous. No free lunch: Rethinking internal feedback for LLM reasoning. *arXiv preprint*, 2025. arXiv:2506.17219.

Anonymous. Are LLM benchmarks already contaminated? a systematic review of contamination detection methods. In *Proceedings of GEM 2026*, 2026a. https://aclanthology.org/2026.gem-main.50.pdf.

Anonymous. Reliability without validity: A systematic, large-scale evaluation of LLM-as-a-Judge models across agreement, consistency, and bias. *arXiv preprint*, 2026b. arXiv:2606.19544.

Anonymous. Judge reliability harness: Stress testing the reliability of LLM judges. *arXiv preprint*, 2026c. arXiv:2603.05399.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *NeurIPS*, 2017. arXiv:1705.08439.

Anthropic. Model context protocol specification, 2024. https://modelcontextprotocol.io.

Anthropic. Effective harnesses for long-running agents. Engineering blog, https://www.anthropic.com/engineering, 2025. Blog post, November 2025. Initializer/worker harness pattern and the claim that continuity is limited by harness robustness rather than model capability.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. *NeurIPS*, 2024. arXiv:2406.11717.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, et al. A general language assistant as a laboratory for alignment. *arXiv preprint*, 2021. arXiv:2112.00861.

Async RL Survey. Design patterns for llm asynchronous training seen in 16 open source rl libraries, 2026. URL https://lilting.ch/en/articles/async-rl-llm-training-architecture-patterns.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. A general theoretical paradigm to understand learning from human preferences. In *AISTATS*, 2024. arXiv:2310.12036.

Ibragim Badertdinov et al. SWE-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents. *arXiv preprint arXiv:2505.20411*, 2025.

Yuntao Bai, Andy Jones, Kamal Ndousse, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint*, 2022a. arXiv:2204.05862.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint*, 2022b. arXiv:2212.08073.

Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y. Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint*, 2025. arXiv:2503.11926.

Ananth Balashankar, Ziteng Sun, Jonathan Berant, Jacob Eisenstein, Michael Collins, Adrian Hutter, Jong Lee, Chirag Nagpal, Flavien Prost, Aradhana Sinha, Ananda Theertha Suresh, and Ahmad Beirami. Infalign: Inference-aware language model alignment. *arXiv preprint arXiv:2412.19792*, 2024. arXiv:2412.19792.

Somnath Banerjee, Rima Hazra, and Animesh Mukherjee. Bridging the multilingual safety divide: Efficient, culturally-aware alignment for global south languages. *arXiv preprint arXiv:2602.13867*, February 2026. arXiv:2602.13867; submitted 2026-02-14.

Victor Barres et al. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

James Begin et al. Preference optimization drives monoculture in LLM prediction markets. *arXiv preprint arXiv:2606.26583*, 2026. arXiv:2606.26583.

Ahmad Beirami, Alekh Agarwal, Jonathan Berant, Alexander D'Amour, Jacob Eisenstein, Chirag Nagpal, and Ananda Theertha Suresh. Theoretical guarantees on the best-of-n alignment policy. *arXiv preprint*, 2024. arXiv:2401.01879.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NeurIPS*, 2015. arXiv:1506.03099.

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B*, 57(1): 289–300, 1995.

Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

Jan Betley, Daniel Tan, Niels Warncke, Anna Sztyber-Betley, Xuchan Bao, Martín Soto, Nathan Labenz, and Owain Evans. Emergent misalignment: Narrow finetuning can produce broadly misaligned LLMs. *ICML*, 2025. arXiv:2502.17424.

Luca Beurer-Kellner, Edoardo Debenedetti, Edgar Dobriban, Marc Fischer, Florian Tramèr, et al. Design patterns for securing LLM agents against prompt injections. *arXiv preprint*, 2025. arXiv:2506.08837.

Federico Bianchi, Mirac Suzgun, Giuseppe Attanasio, Paul Röttger, Dan Jurafsky, Tatsunori Hashimoto, and James Zou. Safety-tuned LLaMAs: Lessons from improving the safety of large language models that follow instructions. *ICLR*, 2024. arXiv:2309.07875.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, et al. LoRA learns less and forgets less. *TMLR*, 2024a. arXiv:2405.09673.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Baber Abbasi, Jessica Zosa Forde, Leo Gao, Jonathan Tow, et al. Lessons from the trenches on reproducible evaluation of language models. In *arXiv preprint*, 2024b. arXiv:2405.14782.

Sam Bowyer, Laurence Aitchison, and Desi R. Ivanova. Position: Don't use the CLT in LLM evals with fewer than a few hundred datapoints. In *ICML*, 2025. arXiv:2503.01747.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39:324–345, 1952.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint*, 2024. arXiv:2407.21787.

Blake Bullwinkel, Eugenia Kim, Amanda Minnich, Mark Russinovich, et al. Learning to attack and defend: Adaptive red teaming of language models via GRPO. *arXiv preprint*, 2026.

arXiv:2606.09701.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeffrey Wu. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023. arXiv:2312.09390.

Shihao Cai et al. AutoForge: Automated environment synthesis for agentic reinforcement learning. *arXiv preprint arXiv:2512.22857*, 2025.

Dallas Card, Peter Henderson, Urvashi Khandelwal, Robin Jia, Kyle Mahowald, and Dan Jurafsky. With little power comes great responsibility. In *EMNLP*, pages 9263–9274, 2020. arXiv:2010.06595.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *TMLR*, 2023a. arXiv:2307.15217.

Stephen Casper, Jason Lin, Joe Kwon, Gatlen Culp, and Dylan Hadfield-Menell. Explore, establish, exploit: Red teaming language models from scratch. *arXiv preprint*, 2023b. arXiv:2306.09442.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems* (*Datasets and Benchmarks Track*), 2025. arXiv:2503.13657.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. *arXiv preprint*, 2023. arXiv:2310.08419.

Angelica Chen, Sadhika Malladi, Lily H. Zhang, Xinyi Chen, Qiuyi Zhang, Rajesh Ranganath, and Kyunghyun Cho. Preference learning algorithms do not learn preference rankings. *arXiv preprint arXiv:2405.19534*, 2024a. arXiv:2405.19534.

Changyu Chen, Zichen Liu, Chao Du, Tianyu Pang, Qian Liu, Arunesh Sinha, Pradeep Varakantham, and Min Lin. Bootstrapping language models with DPO implicit rewards. *arXiv preprint arXiv:2406.09760*, 2024b. arXiv:2406.09760.

Haoxian Chen, Hanyang Zhao, Henry Lam, David Yao, and Wenpin Tang. Mallowspo: Fine-tune your LLM with preference dispersions. *arXiv preprint arXiv:2405.14953*, 2024c. arXiv:2405.14953.

Lichang Chen, Chen Zhu, Davit Soselia, Jiuhai Chen, Tianyi Zhou, Tom Goldstein, Heng Huang, Mohammad Shoeybi, and Bryan Catanzaro. ODIN: Disentangled reward mitigates hacking in RLHF. *ICML*, 2024d. arXiv:2402.07319.

Luoxin Chen et al. Seed-Prover: Deep and broad reasoning for automated theorem proving. *arXiv preprint arXiv:2507.23726*, 2025a.

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint*, 2021. arXiv:2107.03374.

Mingyang Chen et al. ReSearch: Learning to reason with search for LLMs via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025b.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint*, 2016. arXiv:1604.06174.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023a. arXiv:2211.12588.

Xin Chen, Jie Zhang, and Florian Tramèr. Learning to inject: Automated prompt injection via reinforcement learning. *arXiv preprint*, 2026a. arXiv:2602.05746.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for 2+3=? on the overthinking of o1-like LLMs. *arXiv preprint*, 2025c. arXiv:2412.21187.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. In *arXiv preprint*, 2023b. arXiv:2311.17311.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *ICLR*, 2024e. arXiv:2304.05128.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, Hanghang Tong, and Heng Ji. RM-R1: Reward modeling as reasoning. *NeurIPS*, 2025d. arXiv:2505.02387.

Xuanzhong Chen et al. AgentFrontier: Expanding the capability frontier of LLM agents with ZPD-guided data synthesis. *arXiv preprint arXiv:2510.24695*, 2025e.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, Vlad Mikulik, Samuel R. Bowman, Jan Leike, Jared Kaplan, and Ethan Perez. Reasoning models don't always say what they think. *arXiv preprint*, 2025f. arXiv:2505.05410.

Yanjun Chen et al. Exact is easier: Credit assignment for cooperative LLM agents. *arXiv preprint arXiv:2603.06859*, 2026b.

Yongchao Chen et al. R1-Code-Interpreter: LLMs reason with code via supervised and multi-stage reinforcement learning. *arXiv preprint arXiv:2505.21668*, 2025g.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, et al. Pass@k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, August 2025h.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024f. arXiv:2401.01335.

Mingyue Cheng et al. Agent-R1: A unified and modular framework for agentic reinforcement learning. *arXiv preprint arXiv:2511.14460*, 2025.

Weiwei Cheng, Johannes Fürnkranz, Eyke Hüllermeier, and Sang-Hyeun Park. Preference-based policy iteration: Leveraging preference learning for reinforcement learning. In *ECML PKDD*, 2011.

Zhoujun Cheng, Yutao Xie, Yuxiao Qu, Amrith Setlur, et al. Isocompute playbook: Optimally scaling sampling compute for llm rl. *arXiv preprint arXiv:2603.12151*, March 2026.

Sahana Chennabasappa, Cyrus Nikolaidis, Daniel Song, David Molnar, Stephanie Ding, Shengye Wan, et al. LlamaFirewall: An open source guardrail system for building secure AI agents. *arXiv preprint*, 2025. arXiv:2505.03574.

Sayak Ray Chowdhury, Anush Kini, and Nagarajan Natarajan. Provably robust dpo: Aligning language models with noisy feedback. *arXiv preprint arXiv:2403.00409*, 2024. arXiv:2403.00409.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *NeurIPS*, 2017. arXiv:1706.03741.

Fenia Christopoulou, Ronald Cardenas, Gerasimos Lampouras, Haitham Bou-Ammar, and Jun Wang. Sparsepo: Controlling preference alignment of LLMs via sparse token masks. *arXiv preprint arXiv:2410.05102*, 2024. arXiv:2410.05102.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. *ICML*, 2025a. arXiv:2501.17161.

Xiangxiang Chu, Hailang Huang, Xiao Zhang, Fei Wei, et al. Gpg: A simple and strong reinforcement learning baseline for model reasoning. *arXiv preprint arXiv:2504.02546*, April 2025b. v4 2026-02-03.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 2024. arXiv:2210.11416.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. Training verifiers to solve math word problems. *arXiv preprint*, 2021. arXiv:2110.14168.

Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. How many random seeds? statistical power analysis in deep reinforcement learning experiments. *arXiv preprint*, 2018. arXiv:1806.08295.

Contamination Detection Review. Are llm benchmarks already contaminated? a systematic review of contamination detection methods. In *Proceedings of GEM 2026*, 2026. URL https://aclanthology.org/2026.gem-main.50.pdf.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. *ICLR*, 2024. arXiv:2310.02743.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. UltraFeedback: Boosting language models with scaled AI feedback. *ICML*, 2024a. arXiv:2310.01377.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *arXiv preprint*, 2025a. arXiv:2502.01456.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint*, 2025b. arXiv:2505.22617.

Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. OR-Bench: An over-refusal benchmark for large language models. *arXiv preprint*, 2024b. arXiv:2405.20947.

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. Learning how hard to think: Input-adaptive allocation of LM computation. In *ICLR*, 2025. arXiv:2410.04707.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. *NeurIPS Datasets and Benchmarks*, 2024. arXiv:2406.13352.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design. *arXiv preprint*, 2025. arXiv:2503.18813.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Nature*, 2025. arXiv:2501.12948.

Jasper Dekoninck, Mark Niklas Müller, and Martin Vechev. ConStat: Performance-based contamination detection in large language models. In *NeurIPS*, 2024. arXiv:2405.16281.

Wenlong Deng et al. On group relative policy optimization collapse in agent search: The lazy likelihood-displacement. *arXiv preprint arXiv:2512.04220*, 2025.

Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. Multilingual jailbreak challenges in large language models. *ICLR*, 2024. arXiv:2310.06474.

Carson Denison, Monte MacDiarmid, Fazl Barez, David Duvenaud, Shauna Kravec, Samuel Marks, Nicholas Schiefer, Ryan Soklaski, Alex Tamkin, Jared Kaplan, Buck Shlegeris, Samuel R. Bowman, Ethan Perez, and Evan Hubinger. Sycophancy to subterfuge: Investigating reward-tampering in large language models. *arXiv preprint*, 2024. arXiv:2406.10162.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Advances in Neural Information Processing Systems* (*NeurIPS*), 2020. arXiv:2012.02096.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *NeurIPS*, 2023. arXiv:2305.14314.

Hongxin Ding et al. Evorubrics: Dynamic rubrics as rewards via adversarial co-evolution for llm reinforcement learning. *arXiv preprint*, 2026. arXiv:2606.23038.

Konstantin Dobler, Simon Lehnerer, Federico Scozzafava, Jonathan Janke, and Mohamed Ali. Multilingual reasoning gym: Multilingual scaling of procedural reasoning environments. *arXiv preprint arXiv:2603.10793*, March 2026. arXiv:2603.10793; submitted 2026-03-11.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. Show your work: Improved reporting of experimental results. In *EMNLP*, 2019. arXiv:1909.03004.

Elvis Dohmatob, Yunzhen Feng, Pu Yang, Francois Charton, and Julia Kempe. A tale of tails: Model collapse as a change of scaling laws. *ICML*, 2024. arXiv:2402.07043.

Guanting Dong et al. Agentic entropy-balanced policy optimization. *arXiv preprint arXiv:2510.14545*, 2025a.

Guanting Dong et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025b.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *TMLR*, 2023a. arXiv:2304.06767.

Yi Dong, Zhilin Wang, Makesh Narsimhan Sreedhar, Xianchao Wu, and Oleksii Kuchaiev. SteerLM: Attribute conditioned SFT as an (user-steerable) alternative to RLHF. *EMNLP Findings*, 2023b. arXiv:2310.05344.

Karel D'Oosterlinck, Winnie Xu, Chris Develder, Thomas Demeester, Amanpreet Singh, Christopher Potts, Douwe Kiela, and Shikib Mehri. Anchored preference optimization and contrastive revisions: Addressing underspecification in alignment. *arXiv preprint arXiv:2408.06266*, 2024. arXiv:2408.06266.

Marius Dragoi, Ioana Pintilie, Florin Gogianu, and Florin Brad. Beyond pass@k: Breadth-depth metrics for reasoning boundaries. *arXiv preprint arXiv:2510.08325*, October 2025. v2 2025-10-21.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. arXiv:2305.14325.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. Length-controlled AlpacaEval: A simple way to debias automatic evaluators. *arXiv preprint*, 2024. arXiv:2404.04475.

Early Verdicts. Early verdicts, better budgets: Sequential adaptive rollout allocation for compute-efficient RLVR. *arXiv preprint*, July 2026. arXiv:2607.26253.

ECHO. ECHO: Terminal agents learn world models for free. *arXiv preprint arXiv:2605.24517*, 2026.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D'Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfohl, Deepak Ramachandran, Peter Shaw, and Jonathan Berant. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. *COLM*, 2024. arXiv:2312.09244.

Scott Emmons, Erik Yang, Simon Kim, Daniel Kokotajlo, et al. When chain of thought is necessary, language models struggle to evade monitors. *arXiv preprint*, 2025. arXiv:2507.05246.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep RL: A case study on PPO and TRPO. In *ICLR*, 2020. arXiv:2005.12729.

Lasse Espeholt, Hubert Soyer, Rémi Munos, et al. IMPALA: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *ICML*, 2018. arXiv:1802.01561.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. KTO: Model alignment as prospect theoretic optimization. *ICML*, 2024. arXiv:2402.01306.

evalci. evalci: A python library for statistically rigorous comparison of language model evaluations. *arXiv preprint arXiv:2607.04429*, 2026. URL https://arxiv.org/abs/2607.04429.

Yuchen Fan et al. SSRL: Self-search reinforcement learning. *arXiv preprint arXiv:2508.10874*, 2025.

Yuchun Fan, Bei Li, Peiguang Li, Yilin Wang, Yongyu Mu, Jian Yang, Xin Chen, Rongxiang Weng, Jingang Wang, Xunliang Cai, Jingbo Zhu, and Tong Xiao. LANG: Reinforcement learning for multilingual reasoning with language-adaptive hint guidance. *arXiv preprint arXiv:2605.22567*, May 2026. arXiv:2605.22567; submitted 2026-05-21.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. arXiv:2101.03961.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *NeurIPS*, 2023a. arXiv:2305.15408.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. ReTool: Reinforcement learning for strategic tool use in LLMs. *arXiv preprint*, 2025a. arXiv:2504.11536.

Lang Feng et al. Group-in-group policy optimization for LLM agent training. *arXiv preprint arXiv:2505.10978*, 2025b. NeurIPS 2025; code: github.com/langfengQ/verl-agent.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. In *arXiv preprint*, 2023b. arXiv:2309.17179.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018. arXiv:1705.06366.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. arXiv:1605.06676.

Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. arXiv:1705.08926.

Thomas Foster, Anya Sims, Johannes Forkel, Mattie Fellows, and Jakob Foerster. Learning to reason at the frontier of learnability. *arXiv preprint*, 2025. arXiv:2502.12272.

FP8-RL. Fp8-rl: A practical and stable low-precision stack for llm reinforcement learning. *arXiv preprint arXiv:2601.18150*, 2026. URL https://arxiv.org/abs/2601.18150.

Dazhi Fu et al. Many voices, one reward: Multi-role rubric generation for llm judging and reward modeling. *arXiv preprint*, 2026. arXiv:2607.01830.

Wei Fu et al. AReaL: A large-scale asynchronous reinforcement learning system for language reasoning. *arXiv preprint arXiv:2505.24298*, 2025a.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. In *arXiv preprint*, 2025b. arXiv:2508.15260; DeepConf.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. Stream of search (SoS): Learning to search in language. In *COLM*, 2024. arXiv:2404.03683.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint*, 2022. arXiv:2209.07858.

Huan-ang Gao et al. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025a.

Jiaxuan Gao et al. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous RL. *arXiv preprint arXiv:2508.07976*, 2025b.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *ICML*, 2023a. arXiv:2210.10760.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. *ICML*, 2023b. arXiv:2211.10435.

Wei Gao et al. RollArt: Disaggregated multi-task agentic RL training at scale. *arXiv preprint arXiv:2512.22560*, 2025c.

Etienne Gauthier et al. Explaining and preventing alignment collapse in iterative RLHF. *arXiv preprint arXiv:2605.04266*, 2026. arXiv:2605.04266.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint*, 2024. arXiv:2406.20094.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64:86–92, 2021. arXiv:1803.09010.

Andrew Gelman and Eric Loken. The statistical crisis in science. *American Scientist*, 102(6): 460–465, 2014. The "garden of forking paths" argument.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Henry Sleight, John Hughes, Tomasz Korbak, Rajashree Agrawal, Dhruv Pai, Andrey Gromov, Daniel A. Roberts, Diyi Yang, David L. Donoho, and Sanmi Koyejo. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data. *arXiv preprint*, 2024. arXiv:2404.01413.

Bijean Ghafouri, Eun Cheol Choi, Priyanka Dey, and Emilio Ferrara. RLHF may not reflect genuine preferences. *arXiv preprint*, 2026. arXiv:2604.03238.

Alexey Gorbatovski, Boris Shaposhnikov, Alexey Malakhov, Nikita Surnachev, Yaroslav Aksenov, Ian Maksimov, Nikita Balagansky, and Daniil Gavrilov. Learn your reference model for real good alignment. *arXiv preprint arXiv:2404.09656*, 2024. arXiv:2404.09656.

Boyu Gou et al. Mind2Web 2: Evaluating agentic search with agent-as-a-judge. *arXiv preprint arXiv:2506.21506*, 2025.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. ToRA: A tool-integrated reasoning agent for mathematical problem solving. *ICLR*, 2024. arXiv:2309.17452.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. *AISec*, 2023. arXiv:2302.12173.

Tyler Griggs and Philipp Moritz. Open source rl libraries for llms. Anyscale Blog, 2025. URL https://www.anyscale.com/blog/open-source-rl-libraries-for-llms.

Tyler Griggs, Sumanth Hegde, Eric Tang, Shu Liu, Shiyi Cao, Dacheng Li, Charlie Ruan, Philipp Moritz, Kourosh Hakhamaneshi, Richard Liaw, Akshay Malik, Matei Zaharia, and Joseph Gonzalez. Skyrl: A modular full-stack rl library for llms, 2025. URL https://github.com/novasky-ai/skyrl. Docs: https://docs.skyrl.ai/.

Siyi Gu et al. Rethinking reward supervision: Rubric-conditioned self-distillation. *arXiv preprint*, 2026. arXiv:2606.19327.

Melody Y. Guan, Manas Joglekar, Eric Wallace, Saachi Jain, Boaz Barak, Alec Helyar, Rachel Dias, Andrea Vallone, Hongyu Ren, Jason Wei, Hyung Won Chung, Sam Toyer, Johannes Heidecke, Alex Beutel, and Amelia Glaese. Deliberative alignment: Reasoning enables safer language models. *arXiv preprint arXiv:2412.16339*, 2024. arXiv:2412.16339.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rStar-Math: Small LLMs can master math reasoning with self-evolved deep thinking. *arXiv preprint*, 2025. arXiv:2501.04519.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, et al. OpenThoughts: Data recipes for reasoning models. *arXiv preprint*, 2025. arXiv:2506.04178.

Lin Gui, Cristina Gagne, and Victor Veitch. Bonbon alignment for large language models and the sweetness of best-of-n sampling. *arXiv preprint arXiv:2406.00832*, 2024. arXiv:2406.00832.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, et al. Reinforced self-training (ReST) for language modeling. *arXiv preprint*, 2023. arXiv:2308.08998.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint*, 2025. arXiv:2507.17746.

Jiaxin Guo, Zewen Chi, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. Reward reasoning model. *arXiv preprint*, 2025a. arXiv:2505.14674.

Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, et al. Direct language model alignment from online AI feedback. *arXiv preprint*, 2024. arXiv:2402.04792.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, et al. Segment policy optimization: Effective segment-level credit assignment in rl for large language models. *arXiv preprint arXiv:2505.23564*, May 2025b. v2 2025-10-21.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *EMNLP*, 2023. arXiv:2305.14992; RAP.

Peter Hase and Christopher Potts. Counterfactual simulation training for chain-of-thought faithfulness. *arXiv preprint*, 2026. arXiv:2602.20710.

Horace He and Thinking Machines Lab. Defeating nondeterminism in llm inference. 2025. doi: 10.64434/tml.20250910. URL https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/. Library: https://github.com/thinking-machines-lab/batch_invariant_ops.

Luxi He, Mengzhou Xia, and Peter Henderson. What is in your safe data? identifying benign data that breaks safety. *COLM*, 2024. arXiv:2404.01099.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI*, 2018. arXiv:1709.06560.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *ICLR*, 2020. arXiv:1904.09751; introduces nucleus (top-p) sampling.

Ilgee Hong, Changlong Yu, Liang Qiu, Weixiang Yan, Zhenghao Zhang, Qingru Wang, Fang Xu, Wenlin Yin, Xian Zhang, Tuo Zhao, and Haoming Jiang. Think-RM: Enabling long-

horizon reasoning in generative reward models. *arXiv preprint*, 2025. arXiv:2505.16265.

Jiwoo Hong, Noah Lee, and James Thorne. ORPO: Monolithic preference optimization without reference model. *EMNLP*, 2024a. arXiv:2403.07691.

Zhang-Wei Hong, Idan Shenfeld, Tsun-Hsuan Wang, Yung-Sung Chuang, Aldo Pareja, James Glass, Akash Srivastava, and Pulkit Agrawal. Curiosity-driven red-teaming for large language models. *ICLR*, 2024b. arXiv:2402.19464.

Jack Hopkins et al. Factorio learning environment. *arXiv preprint arXiv:2503.09617*, 2025.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-STaR: Training verifiers for self-taught reasoners. In *COLM*, 2024. arXiv:2402.06457.

Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. TreeRL: LLM reinforcement learning with on-policy tree search. In *ACL*, 2025. arXiv:2506.11902.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *ICML*, 2019. arXiv:1902.00751.

Chia-Yi Hsu, Yu-Lin Tsai, Chih-Hsun Lin, Pin-Yu Chen, Chia-Mu Yu, and Chun-Ying Huang. Safe LoRA: The silver lining of reducing safety risks when fine-tuning large language models. *NeurIPS*, 2024. arXiv:2405.16833.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *ICLR*, 2022. arXiv:2106.09685.

Jian Hu. REINFORCE++: A simple and efficient approach for aligning large language models. *arXiv preprint*, 2025. arXiv:2501.03262.

Jian Hu, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao. OpenRLHF: An easy-to-use, scalable and high-performance RLHF framework. *arXiv preprint*, 2024a. arXiv:2405.11143.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, et al. MiniCPM: Unveiling the potential of small language models with scalable training strategies. *COLM*, 2024b. arXiv:2404.06395.

Audrey Huang, Wenhao Zhan, Tengyang Xie, Jason D. Lee, Wen Sun, Akshay Krishnamurthy, and Dylan J. Foster. Correcting the mythos of KL-regularization: Direct alignment without overoptimization via chi-squared preference optimization. *arXiv preprint arXiv:2407.13399*, 2024a. arXiv:2407.13399.

Chengsong Huang, Haolin Liu, Tong Zheng, Runpeng Dai, Langlin Huang, Jinyuan Li, Zongxia Li, Zhepei Wei, Yu Meng, and Jiaxin Huang. G-Zero: Self-play for open-ended generation from zero data. *arXiv preprint arXiv:2605.09959*, 2026a.

Chengsong Huang et al. R-Zero: Self-evolving reasoning LLM from zero data. *arXiv preprint arXiv:2508.05004*, 2025a.

Chengyu Huang et al. Bootstrapping post-training signals for open-ended tasks via rubric-based self-play on pre-training text. *arXiv preprint arXiv:2604.20051*, 2026b. POP.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *ICLR*, 2024b. arXiv:2310.01798.

Shengyi Huang, Michael Noukhovitch, Arian Hosseini, Kashif Rasul, Weixun Wang, and Lewis Tunstall. The N+ implementation details of RLHF with PPO: A case study on TL;DR summarization. *arXiv preprint*, 2024c. arXiv:2403.17031.

Tiansheng Huang, Sihao Hu, and Ling Liu. Vaccine: Perturbation-aware alignment for large language models against harmful fine-tuning attack. *NeurIPS*, 2024d. arXiv:2402.01109.

Wei Huang, Yi Ge, Shuai Yang, Yicheng Xiao, et al. Qerl: Beyond efficiency – quantization-enhanced reinforcement learning for llms. *arXiv preprint arXiv:2510.11696*, October 2025b.

Wenqi Huang et al. DeepSWE: Measuring frontier coding agents on original, long-horizon engineering tasks. *arXiv preprint arXiv:2607.07946*, 2026c. Benchmark; distinct from DeepSWE-Preview the coding agent.

Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. Catastrophic jailbreak of open-source LLMs via exploiting generation. *ICLR*, 2024e. arXiv:2310.06987.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. GPipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems* (*NeurIPS*), 2019. arXiv:1811.06965.

Yu Huang, Zixin Wen, Yuejie Chi, Yuting Wei, et al. On the emergence of implicit curriculum in rlvr learning dynamics. *arXiv preprint arXiv:2602.14872*, February 2026d. v3 2026-06-29.

Yuchen Huang et al. Environment scaling for interactive agentic experience collection: A survey. *arXiv preprint arXiv:2511.09586*, 2025c.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Yang, Xudong Yu, Ru Wang, Kehan Wang, Ke Xu, Bohan Peng, Rui Zhang, Yan Wang, Jiaqi Liu, Chunlei Zhang, Jianguo Zhou, Yongsheng Zhu, Jianfei Zhang, Jinghui Chen, Zhenzhong Zhang, Le Song, and Yuan Zhang. Reinforcement learning with rubric anchors. *arXiv preprint*, 2025d.

arXiv:2508.12790.

Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, et al. Sleeper agents: Training deceptive LLMs that persist through safety training. *arXiv preprint*, 2024. arXiv:2401.05566.

Edward Hughes, Michael D. Dennis, Jack Parker-Holder, Feryal Behbahani, Aditi Mavalankar, Yuge Shi, Tom Schaul, and Tim Rocktäschel. Position: Open-endedness is essential for artificial superhuman intelligence. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. PMLR 235:20597–20616. arXiv:2406.04268.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *NeurIPS*, 2018. arXiv:1811.06521.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *ICLR*, 2023. arXiv:2212.04089.

Shawn Im, Federico Danieli, Skyler Seto, Barry-John Theobald, and Katherine Metcalf. Normalized rewards for preference optimization. *arXiv preprint arXiv:2607.16240*, 2026. arXiv:2607.16240.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama Guard: LLM-based input-output safeguard for human-ai conversations. *arXiv preprint*, 2023. arXiv:2312.06674.

John P. A. Ioannidis. Why most published research findings are false. *PLoS Medicine*, 2(8): e124, 2005.

Edward L. Ionides. Truncated importance sampling. *Journal of Computational and Graphical Statistics*, 17:295–311, 2008.

Geoffrey Irving, Paul Christiano, and Dario Amodei. AI safety via debate. *arXiv preprint*, 2018. arXiv:1805.00899.

Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A. Smith, Yejin Choi, and Hannaneh Hajishirzi. Unpacking DPO and PPO: Disentangling best practices for learning from preference feedback. *NeurIPS*, 2024. arXiv:2406.09279.

Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. DeepSpeed Ulysses: System optimizations for enabling training of extreme long sequence transformer models. *arXiv preprint*, 2023. arXiv:2309.14509.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *ICLR*, 2025a. arXiv:2403.07974.

Naman Jain et al. R2E-Gym: Procedural environments and hybrid verifiers for scaling open-weights SWE agents. *arXiv preprint arXiv:2504.07164*, 2025b.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint*, 2023. arXiv:2309.00614.

Neel Jain, Ping-yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, et al. NEFTune: Noisy embeddings improve instruction finetuning. In *ICLR*, 2024. arXiv:2310.05914.

Jet-RL. Jet-rl: Enabling on-policy fp8 reinforcement learning with unified training and rollout precision flow. *arXiv preprint arXiv:2601.14243*, 2026. URL https://arxiv.org/abs/2601.14243.

Kaixuan Ji, Guanlin Liu, Ning Dai, Qingping Yang, Renjie Zheng, Zheng Wu, Chen Dun, Quanquan Gu, and Lin Yan. Enhancing multi-step reasoning abilities of language models through direct q-function optimization. *arXiv preprint arXiv:2410.09302*, 2024. arXiv:2410.09302.

Yunjie Ji, Xiaoyu Tian, Sitong Zhao, Haotian Li, Yiping Ma, Jiayi Zeng, et al. AM-Thinking-v1: Advancing the frontier of reasoning at 32B scale. *arXiv preprint*, 2025. arXiv:2505.08311.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-Blender: Ensembling large language models with pairwise ranking and generative fusion. *ACL*, 2023. arXiv:2306.02561.

Enyi Jiang, Anders Gjølbye, Yibo Jacky Zhang, and Sanmi Koyejo. When behavioral safety evaluation fails: A representation-level perspective. *arXiv preprint*, 2026a. arXiv:2606.08044.

Yuxuan Jiang et al. SCRIBE: Structured mid-level supervision for tool-using language models. *arXiv preprint arXiv:2601.03555*, 2026b.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023. URL https://arxiv.org/abs/2310.06770.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint*, 2025a. arXiv:2503.09516.

Yiyang Jin et al. ReVeal: Self-evolving code agents via reliable self-verification. *arXiv preprint arXiv:2506.11442*, 2025b.

Faaiz Joad, Majd Hawasly, Sabri Boughorbel, and Nadir Durrani. There is more to refusal in large language models than a single direction. *arXiv preprint*, 2026. arXiv:2602.02132.

Seungjae Jung, Gunsoo Han, Daniel Wontae Nam, and Kyoung-Woon On. Binary classifier optimization for large language model alignment. *arXiv preprint arXiv:2404.04656*, 2024. arXiv:2404.04656.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, 2002.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI agents that matter. *Transactions on Machine Learning Research*, 2025. arXiv:2407.01502.

Hamid Kazemi et al. A single neuron is sufficient to bypass safety alignment in large language models. *arXiv preprint*, 2026. arXiv:2605.08513.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. *ICML*, 2025. arXiv:2410.01679.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2–3):209–232, 2002. Source of the simulation lemma bounding value error by model error times horizon.

Zachary Kenton, Noah Y. Siegel, János Kramár, Jonah Brown-Cohen, Samuel Albanie, Jannis Bulian, Rishabh Agarwal, David Lindner, Yunhao Tang, Noah D. Goodman, and Rohin Shah. On scalable oversight with weak LLMs judging strong LLMs. In *Advances in Neural Information Processing Systems* (*NeurIPS*), 2024. arXiv:2407.04622.

Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R. Bowman, Tim Rocktäschel, and Ethan Perez. Debating with more persuasive LLMs leads to more truthful answers. In *Proceedings of the 41st International Conference on Machine Learning* (*ICML*), 2024. arXiv:2402.06782.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, et al. The art of scaling reinforcement learning compute for llms. *arXiv preprint arXiv:2510.13786*, October 2025. Submitted 2025-10-15. ScaleRL; sigmoidal RL scaling law.

Dahyun Kim, Yungi Kim, Wonho Song, Hyeonwoo Kim, Yunsu Kim, Sanghoon Kim, and Chanjun Park. sdpo: Don't use your data all at once. *arXiv preprint arXiv:2403.19270*, 2024.

arXiv:2403.19270.

Minwu Kim, Anubhav Shrestha, Safal Shrestha, Aadim Nepal, et al. Reinforcement learning vs. distillation: Understanding accuracy and capability in llm reasoning. *arXiv preprint arXiv:2505.14216*, May 2025. v2 2025-10-31.

Kimi Team. Kimi k1.5: Scaling reinforcement learning with LLMs. *arXiv preprint*, 2025a. arXiv:2501.12599.

Kimi Team. Kimi K2: Open agentic intelligence. *arXiv preprint*, 2025b. arXiv:2507.20534.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114:3521–3526, 2017. arXiv:1612.00796.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *ECML*, pages 282–293, 2006.

Jing Yu Koh et al. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*, 2024.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *NeurIPS*, 2022. arXiv:2205.11916.

Elena Kopteva and Vitaliy Hlynianyi-Zhuk. Rater state bias in RLHF preference data: An audit framework. *arXiv preprint*, 2026. arXiv:2607.16195.

Tomasz Korbak, Mikita Balesni, Elizabeth Barnes, Yoshua Bengio, Joe Benton, Joseph Bloom, et al. Chain of thought monitorability: A new and fragile opportunity for AI safety. *arXiv preprint*, 2025. arXiv:2507.11473.

Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. In *Proceedings of Machine Learning and Systems* (*MLSys*), 2023. arXiv:2205.05198.

Satwik Kottur, José M. F. Moura, Stefan Lee, and Dhruv Batra. Natural language does not emerge 'naturally' in multi-agent dialog. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), 2017. arXiv:1706.08502.

Mario Michael Krell, Matej Kosec, Sergio P. Perez, and Andrew Fitzgibbon. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. *arXiv preprint*, 2022. arXiv:2107.02027.

Aounon Kumar, Chirag Agarwal, Suraj Srinivas, Aaron Jiaxun Li, Soheil Feizi, and Himabindu Lakkaraju. Certifying LLM safety against adversarial prompting. *arXiv preprint*, 2023. arXiv:2309.02705.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *ICLR*, 2025. arXiv:2409.12917; SCoRe.

Priyanshu Kumar, Elaine Lau, Saranya Vijayakumar, Tu Trinh, Elaine Chang, Vaughn Robinson, Sean Hendryx, Shuyan Zhou, Matt Fredrikson, Summer Yue, and Zifan Wang. Refusal-trained LLMs are easily jailbroken as browser agents. *arXiv preprint*, 2024. arXiv:2410.13886.

Adarsh Kumarappan and Ananya Mujoo. Not just RLHF: Why alignment alone won't fix multi-agent sycophancy. *arXiv preprint*, 2026. arXiv:2605.12991.

Wai-Chung Kwan et al. SCOPE: Self-play via co-evolving policies for open-ended tasks. *arXiv preprint arXiv:2605.31433*, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *SOSP*, 2023. arXiv:2309.06180.

Hanyu Lai et al. ComputerRL: Scaling end-to-end online reinforcement learning for computer use agents. *arXiv preprint arXiv:2508.14040*, 2025a.

Song Lai, Haohan Zhao, Rong Feng, Changyi Ma, Wenzhuo Liu, Hongbo Zhao, Xi Lin, Dong Yi, Qingfu Zhang, Hongbin Liu, Gaofeng Meng, and Fei Zhu. Reinforcement fine-tuning naturally mitigates forgetting in continual post-training. *arXiv preprint arXiv:2507.05386*, July 2025b. arXiv:2507.05386; submitted 2025-07-07, v6 2026.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*, 2024. arXiv:2406.18629.

Nathan Lambert. *Reinforcement Learning from Human Feedback*. Online, 2025. https://rlhfbook.com.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, Lester James V. Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. RewardBench: Evaluating reward models for language modeling. *arXiv preprint*, 2024. arXiv:2403.13787.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, et al. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint*, 2025. arXiv:2411.15124.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint*,

2023. arXiv:2307.13702.

Angeliki Lazaridou and Marco Baroni. Emergent multi-agent communication in the deep learning era. *arXiv preprint*, 2020. arXiv:2006.02419.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. In *International Conference on Learning Representations* (*ICLR*), 2017. arXiv:1612.07182.

Harrison Lee, Samrat Phatale, Hassan Mansoor, et al. RLAIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. *arXiv preprint*, 2023. arXiv:2309.00267.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *ACL*, 2022. arXiv:2107.06499.

LEEPS. LEEPS: Latent-guided explore–exploit prompt sampling for efficient RLVR. *arXiv preprint*, July 2026. arXiv:2607.28077.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations* (*ICLR*), 2021. arXiv:2006.16668.

Simon Lermen, Charlie Rogers-Smith, and Jeffrey Ladish. LoRA fine-tuning efficiently undoes safety training in Llama 2-Chat 70B. *arXiv preprint*, 2023. arXiv:2310.20624.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning* (*ICML*), 2023. arXiv:2211.17192.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint*, 2018. arXiv:1805.00909.

Changling Li, Terry Jingchen Zhang, Jie Zhang, and Zhijing Jin. Decomposing and measuring evaluation awareness. *arXiv preprint*, 2026. arXiv:2605.23055.

Dawei Li, Renliang Sun, Yue Huang, Ming Zhong, Bohan Jiang, Jiawei Han, Xiangliang Zhang, and Wei Wang. Preference leakage: A contamination problem in LLM-as-a-Judge. *arXiv preprint*, 2025a. arXiv:2502.01534.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *Transactions on Machine Learning Research*, 2024a. arXiv:2402.05120.

Kuan Li et al. WebSailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*, 2025b.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In *NAACL*, 2024b. arXiv:2308.12032.

Peiji Li et al. InternBootcamp technical report: Boosting LLM reasoning with verifiable task scaling. *arXiv preprint*, 2025c. arXiv:2508.08636.

Pengxiang Li et al. Efficient multi-turn RL for GUI agents via decoupled training and adaptive data curation. *arXiv preprint arXiv:2509.23866*, 2025d. DART.

Weizhen Li et al. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic RL. *arXiv preprint arXiv:2508.13167*, 2025e. AFM.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason Weston, and Mike Lewis. Self-alignment with instruction backtranslation. *ICLR*, 2024c. arXiv:2308.06259.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021. arXiv:2101.00190.

Xiaoxi Li et al. WebThinker: Empowering large reasoning models with deep research capability. *arXiv preprint arXiv:2504.21776*, 2025f.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. ToRL: Scaling tool-integrated RL. *arXiv preprint arXiv:2503.23383*, 2025g.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making large language models better reasoners with step-aware verifier. In *ACL*, 2023a. arXiv:2206.02336; the DIVERSE voting verifier.

Yizhi Li, Qingshui Gu, Zhoufutu Wen, Ziniu Li, Tianshun Xing, Shuyue Guo, Tianyu Zheng, Xin Zhou, Xingwei Qu, Wangchunshu Zhou, Zheng Zhang, Wei Shen, Qian Liu, Chenghua Lin, Jian Yang, Ge Zhang, and Wenhao Huang. TreePO: Bridging the gap of policy optimization and efficacy and inference efficiency with heuristic tree-based modeling. *arXiv preprint*, 2025h. arXiv:2508.17445.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *ICLR*, 2024d. arXiv:2402.12875.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, et al. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint arXiv:2310.10505*, October 2023b. v4 2024-05-16.

Jing Liang, Hongyao Tang, Yi Ma, Yancheng He, et al. The mirage of optimizing training policies: Monotonic inference policies as the real objective for llm reinforcement learning. *arXiv preprint arXiv:2606.29526*, June 2026.

Junwei Liao et al. MARFT: Multi-agent reinforcement fine-tuning. *arXiv preprint arXiv:2504.16129*, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024. arXiv:2305.20050.

Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Chandu, Chandra Bhagavatula, and Yejin Choi. The unlocking spell on base LLMs: Rethinking alignment via in-context learning. *ICLR*, 2024a. arXiv:2312.01552.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, et al. Rho-1: Not all tokens are what you need. In *NeurIPS*, 2024b. arXiv:2404.07965.

Ling Team, Anqi Shen, Baihui Li, Bin Hu, et al. Every step evolves: Scaling reinforcement learning for trillion-scale thinking model. *arXiv preprint arXiv:2510.18855*, October 2025. v2 2025-10-25. Ring-1T; IcePop, C3PO++, ASystem.

John D. C. Little. A proof for the queuing formula: $l = \lambda w$. *Operations Research*, 9(3):383–387, 1961. The distribution-free relation used to size environment pools.

Bo Liu et al. SPICE: Self-play in corpus environments improves reasoning. *arXiv preprint arXiv:2510.24684*, 2025a.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-Reward: Bag of tricks for reward modeling in LLMs. *arXiv preprint*, 2024a. arXiv:2410.18451.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Shuicheng Yan, and Yahui Zhou. Skywork-Reward-V2: Scaling preference data curation via human-AI synergy. *arXiv preprint*, 2025b. arXiv:2507.01352.

Emmy Liu, Graham Neubig, and Chenyan Xiong. Midtraining bridges pretraining and posttraining distributions. *arXiv preprint*, 2025c. arXiv:2510.14865.

Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. In *International Conference on Learning Representations (ICLR)*, 2024b. arXiv:2310.01889.

Mingjie Liu et al. ProRL: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*, 2025d.

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1B LLM surpass 405B LLM? rethinking compute-optimal test-time scaling. *arXiv preprint*, 2025e. arXiv:2502.06703.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *ICML*, 2024c. arXiv:2402.09353.

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J. Liu, and Jialu Liu. Statistical rejection sampling improves preference optimization. *ICLR*, 2024d. arXiv:2309.06657.

Wei Liu, Yang Bai, Chengcheng Han, Rongxiang Weng, Jun Xu, Xuezhi Cao, Jingang Wang, and Xunliang Cai. Length desensitization in direct preference optimization. *arXiv preprint arXiv:2409.06411*, 2024e. arXiv:2409.06411.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *ICLR*, 2024f. arXiv:2312.15685.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. *ICLR*, 2024g. arXiv:2310.04451.

Yihong Liu, Raoyuan Zhao, Michael A. Hedderich, and Hinrich Schütze. Crosslingual on-policy self-distillation for multilingual reasoning. *arXiv preprint arXiv:2605.09548*, May 2026a. arXiv:2605.09548; submitted 2026-05-10.

Yixin Liu, Pengfei Liu, and Arman Cohan. Understanding reference policies in direct preference optimization. *arXiv preprint arXiv:2407.13709*, 2024h. arXiv:2407.13709.

Yuhang Liu et al. InfiGUI-R1: Advancing multimodal GUI agents from reactive actors to deliberative reasoners. *arXiv preprint arXiv:2504.14239*, 2025f.

Zehua Liu, Yuxuan Yao, Xiaojin Fu, Tao Zhong, et al. Asympo: Asymmetric-scale policy optimization for asynchronous llm post-training without behavior information. *arXiv preprint arXiv:2606.03070*, June 2026b. v3 2026-06-23.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. *arXiv preprint*, 2025g. arXiv:2503.20783.

Zihe Liu, Jiashun Liu, Yancheng He, Weixun Wang, et al. Part i: Tricks or traps? a deep dive into rl for llm reasoning. *arXiv preprint arXiv:2508.08221*, August 2025h. v3 2025-10-27. Introduces Lite PPO.

Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-time scaling for generalist reward modeling. *arXiv preprint*, 2025i. arXiv:2504.02495.

LLM-as-Judge Reliability Audit. When the judge changes, so does the measurement: Auditing llm-as-judge reliability. *arXiv preprint arXiv:2607.08535*, 2026. URL https://arxiv.org/abs/2607.08535.

LMSYS Org. Towards deterministic inference in sglang and reproducible rl training, 2025. URL https://www.lmsys.org/blog/2025-09-22-sglang-deterministic/.

LMSYS Org. Towards blackwell-native 8-bit and 4-bit rl: End-to-end mxfp8 and nvfp4 rl in miles, 2026. URL https://www.lmsys.org/blog/2026-07-29-mxfp8-nvfp4-rl/.

Shayne Longpre, Robert Mahari, Anthony Chen, Naana Obeng-Marnu, Damien Sileo, William Brannon, Niklas Muennighoff, Nathan Khazam, Jad Kabbara, Kartik Perisetla, et al. The data provenance initiative: A large scale audit of dataset licensing and attribution in AI. *Nature Machine Intelligence*, 2024. arXiv:2310.16787.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems* (*NeurIPS*), 2017. arXiv:1706.02275.

Hongliang Lu et al. Search self-play: Pushing the frontier of agent capability without supervision. *arXiv preprint arXiv:2510.18821*, 2025.

Junru Lu, Jiazheng Li, Siyu An, Meng Zhao, Yulan He, Di Yin, and Xing Sun. Eliminating biased length reliance of direct preference optimization via down-sampled KL divergence. *arXiv preprint arXiv:2406.10957*, 2024. arXiv:2406.10957.

Kevin Lu and Thinking Machines Lab. On-policy distillation, 2025. URL https://thinkingmachines.ai/blog/on-policy-distillation/.

R. Duncan Luce. *Individual Choice Behavior: A Theoretical Analysis*. Wiley, 1959.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint*, 2024. arXiv:2406.06592.

Michael Luo et al. DeepSWE: Training a fully open-sourced, state-of-the-art coding agent by scaling RL. Agentica / Together AI notion blog, 2025a.

Run Luo et al. GUI-R1: A generalist R1-style vision-language action model for GUI agents. *arXiv preprint arXiv:2504.10458*, 2025b.

Xufang Luo et al. Agent lightning: Train any AI agents with reinforcement learning. *arXiv preprint arXiv:2508.03680*, 2025c.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv*

preprint, 2025d. arXiv:2308.08747.

Bowen Lv et al. ScaleCUA: Scaling computer use agents with verifiable task synthesis and efficient online RL. *arXiv preprint arXiv:2607.11185*, 2026.

Ma, Zheng, et al. Rollout routing replay: Stabilizing moe reinforcement learning by aligning training and inference routers. *arXiv preprint arXiv:2510.11370*, 2025. URL https://arxiv.org/abs/2510.11370.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *NeurIPS*, 2023. arXiv:2303.17651.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. RewardBench 2: Advancing reward model evaluation. *arXiv preprint*, 2025. arXiv:2506.01937.

David Manheim and Scott Garrabrant. Categorizing variants of goodhart's law. *arXiv preprint*, 2018. arXiv:1803.04585.

Samuel Marks, Johannes Treutlein, Trenton Bricken, Jack Lindsey, Jonathan Marcus, Siddharth Mishra-Sharma, et al. Auditing language models for hidden objectives. *arXiv preprint*, 2025. arXiv:2503.10965.

MC Pass-k Critic. Learning with a single rollout via monte carlo pass@k critic. *arXiv preprint*, June 2026. arXiv:2606.25451.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box LLMs automatically. *NeurIPS*, 2024. arXiv:2312.02119.

Lang Mei et al. SearchArt: Training long-horizon search agent with scalable synthetic and verified task. *arXiv preprint arXiv:2607.24850*, 2026.

Alexander Meinke, Bronson Schoen, Jérémy Scheurer, Mikita Balesni, Rusheb Shah, and Marius Hobbhahn. Frontier models are capable of in-context scheming. *arXiv preprint*, 2024. arXiv:2412.04984.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, Kaipeng Zhang, Ping Luo, et al. MM-Eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement

learning. *arXiv preprint arXiv:2503.07365*, March 2025. arXiv:2503.07365; submitted 2025-03-10, v2.

Yu Meng, Mengzhou Xia, and Danqi Chen. SimPO: Simple preference optimization with a reference-free reward. *NeurIPS*, 2024. arXiv:2405.14734.

Mike A. Merrill et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. In *TACL*, 2023. arXiv:2207.00729.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *ICLR*, 2024. arXiv:2310.07923.

Meta. Are: Scaling up agent environments and evaluations. *arXiv preprint arXiv:2509.17158*, 2025. URL https://arxiv.org/abs/2509.17158.

Meta. Gaia2: Benchmarking llm agents on dynamic and asynchronous environments. *arXiv preprint arXiv:2602.11964*, 2026. URL https://arxiv.org/abs/2602.11964.

Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations. *arXiv preprint arXiv:2411.00640*, 2024. URL https://arxiv.org/abs/2411.00640. Blog: https://www.anthropic.com/research/statistical-approach-to-model-evals.

Zijun Min, Bingshuai Liu, Ante Wang, Long Zhang, Anxiang Zeng, Haibo Zhang, and Jinsong Su. Orchestrating tokens and sequences: Dynamic hybrid policy optimization for rlvr. *arXiv preprint arXiv:2601.05607*, January 2026. Submitted 2026-01-09; ACL 2026 Findings 37782–37795.

MiniMax. MiniMax-M1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint*, 2025. arXiv:2506.13585.

Missing Old Logits. Missing old logits in asynchronous agentic RL: Semantic mismatch and repair methods for off-policy correction. *arXiv preprint arXiv:2605.12070*, May 2026. URL https://arxiv.org/abs/2605.12070.

Mistral-AI, Abhinav Rastogi, Albert Q. Jiang, et al. Magistral. *arXiv preprint arXiv:2506.10910*, June 2025.

Wachiravit Modecrua et al. Multi-turn reinforcement learning for tool-calling agents with iterative reward calibration. *arXiv preprint arXiv:2604.02869*, 2026.

Moonshot AI. checkpoint-engine, 2025. URL https://github.com/MoonshotAI/checkpoint-engine.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A

distributed framework for emerging AI applications. In *USENIX Symposium on Operating Systems Design and Implementation* (*OSDI*), 2018. arXiv:1712.05889.

Sumeet Ramesh Motwani, Mikhail Baranchuk, Martin Strohmeier, Vijay Bolina, Philip H. S. Torr, Lewis Hammond, and Christian Schroeder de Witt. Secret collusion among AI agents: Multi-agent deception via steganography. In *Advances in Neural Information Processing Systems* (*NeurIPS*), 2024a. arXiv:2402.07510.

Sumeet Ramesh Motwani, Chandler Smith, Rocktim Jyoti Das, Rafael Rafailov, Ivan Laptev, Philip Torr, Fabio Pizzati, Ronald Clark, and Christian Schroeder de Witt. MALT: Improving reasoning with multi-agent LLM training. *arXiv preprint*, 2024b. arXiv:2412.01928.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint*, 2025. arXiv:2501.19393.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tur, and Hao Peng. Reinforcement learning finetunes small subnetworks in large language models. *arXiv preprint arXiv:2505.11711*, May 2025. v2 2025-12-18.

Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help? In *NeurIPS*, 2019. arXiv:1906.02629.

Rémi Munos, Michal Valko, Daniele Calandriello, et al. Nash learning from human feedback. *ICML*, 2024. arXiv:2312.00886.

Victor Muryn et al. MacArena: Benchmarking computer use agents on an online macOS environment. *arXiv preprint arXiv:2606.06560*, 2026.

Silen Naihin, David Atkinson, Marc Green, Merwane Hamadi, Craig Swift, Douglas Schonholtz, Adam Tauman Kalai, and David Bau. Testing language model agents safely in the wild. *arXiv preprint*, 2023. arXiv:2311.10538.

Tianlong Nan et al. Efficient exploration for iterative nash preference optimization. *arXiv preprint arXiv:2606.01382*, 2026. arXiv:2606.01382.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on GPU clusters using Megatron-LM. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (*SC*), 2021. arXiv:2104.04473.

NVIDIA. Llama-Nemotron: Efficient reasoning models. *arXiv preprint*, 2025. arXiv:2505.00949.

NVIDIA. Megatron-fsdp (megatron core developer guide), 2026a. URL `https://docs.nvidia.com/megatron-core/developer-guide/latest/user-guide/features/megatron_fsd`

p.html.

NVIDIA. Scalable training of mixture-of-experts models with megatron core. *arXiv preprint arXiv:2603.07685*, 2026b. URL https://arxiv.org/abs/2603.07685.

NVIDIA. Nemo rl: A scalable and efficient post-training library, 2026c. URL https://docs.nvidia.com/nemo/rl/latest/. Repository: https://github.com/NVIDIA-NeMo/RL; v0.5.0 release notes.

NVIDIA. SWE-RL (stages 2.1–2.2). Nemotron documentation, https://docs.nvidia.com/nemotron/, 2026d.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint*, 2021. arXiv:2112.00114.

Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Is self-repair a silver bullet for code generation? In *ICLR*, 2024. arXiv:2306.09896.

Frans A. Oliehoek and Christian Amato. *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems. Springer, 2016.

Online Difficulty Filtering. Online difficulty filtering for reasoning oriented reinforcement learning. *arXiv preprint arXiv:2504.03380*, April 2025.

OpenAI. Learning to reason with LLMs, 2024. https://openai.com/index/learning-to-reason-with-llms/.

OpenRLHF Project. Openrlhf architecture documentation, 2025. URL https://openrlhf.readthedocs.io/en/latest/architecture.html.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022. arXiv:2203.02155.

Art B. Owen. *Monte Carlo Theory, Methods and Examples*. 2013. Available at https://artowen.su.domains/mc/.

Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228*, 2024. arXiv:2402.13228.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. *ICLR*, 2022. arXiv:2201.03544.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. *arXiv preprint*,

2024. arXiv:2412.21139.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. *arXiv preprint arXiv:2404.19733*, 2024. arXiv:2404.19733.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. *NeurIPS*, 2024. arXiv:2404.13076.

Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. MAPoRL: Multi-agent post-co-training for collaborative large language models with reinforcement learning. *arXiv preprint*, 2025a. arXiv:2502.18439.

Jaesung R. Park, Junsu Kim, Gyeongman Kim, Jinyoung Jo, et al. Clip-low increases entropy and clip-high decreases entropy in reinforcement learning of large language models. *arXiv preprint arXiv:2509.26114*, September 2025b.

Ryan Park, Rafael Rafailov, Stefano Ermon, and Chelsea Finn. Disentangling length from quality in direct preference optimization. *ACL Findings*, 2024. arXiv:2403.19159.

PASS@(k,T) Analysis. Does rl expand the capability boundary of llm agents? a pass@(k,t) analysis. *arXiv preprint arXiv:2604.14877*, April 2026.

PaW. Policy and world modeling co-training for language agents. *arXiv preprint arXiv:2606.02388*, 2026.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *EMNLP*, 2022. arXiv:2202.03286.

Ethan Perez, Sam Ringer, Kamilė Lukošiūtė, Karina Nguyen, et al. Discovering language model behaviors with model-written evaluations. *ACL Findings*, 2023. arXiv:2212.09251.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. In *COLM*, 2024. arXiv:2404.15758.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint*, 2025. arXiv:2501.14249.

Mary Phuong, Erik Jenner, Laurent Simon, Lewis Ho, et al. GDM AI control roadmap. *arXiv preprint*, 2026. arXiv:2607.13087.

R. L. Plackett. The analysis of permutations. *Journal of the Royal Statistical Society C*, 24:193–202, 1975.

Elinor Poole-Dayan, Jillian Fisher, Atoosa Kasirzadeh, Jacob Andreas, Mitchell Gordon, and Michiel A. Bakker. A roadmap to impactful pluralistic alignment research. *arXiv preprint*

arXiv:2607.22305, July 2026. arXiv:2607.22305; submitted 2026-07-24.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, et al. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, May 2025.

Prime Intellect. Intellect-2: A reasoning model trained through globally decentralized reinforcement learning. *arXiv preprint arXiv:2505.07291*, May 2025.

Prime Intellect. prime-rl: Asynchronous rl with strict staleness control, 2026. URL https://github.com/PrimeIntellect-ai/prime-rl.

Progressive Rollout. Learning as reasoning unfolds: Progressive rollout allocation for efficient reinforcement learning. *arXiv preprint*, July 2026. arXiv:2607.22002.

Penghui Qi, Xinyi Wan, Guangxing Huang, and Min Lin. Zero bubble pipeline parallelism. In *International Conference on Learning Representations* (*ICLR*), 2024a. arXiv:2401.10241.

Penghui Qi, Zichen Liu, Xiangxin Zhou, Tianyu Pang, et al. Defeating the training-inference mismatch via fp16. *arXiv preprint arXiv:2510.26788*, October 2025.

Xiangyu Qi, Ashwinee Panda, Kaifeng Lyu, Xiao Ma, Subhrajit Roy, Ahmad Beirami, Prateek Mittal, and Peter Henderson. Safety alignment should be made more than just a few tokens deep. *arXiv preprint*, 2024b. arXiv:2406.05946.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! *ICLR*, 2024c. arXiv:2310.03693.

Zehan Qi et al. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*, 2024d.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. *arXiv preprint*, 2025. arXiv:2504.13958.

Zile Qiao et al. WebResearcher: Unleashing unbounded reasoning capability in long-horizon agents. *arXiv preprint arXiv:2509.13309*, 2025.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. In *NeurIPS*, 2024. arXiv:2407.18219; RISE.

Qwen Team. Qwen3 technical report. *arXiv preprint*, 2025. arXiv:2505.09388.

radixark. Miles: Enterprise-grade rl training (production fork of slime), 2026. URL https://github.com/radixark/miles.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023. arXiv:2305.18290.

Rafael Rafailov, Yaswanth Chittepu, Ryan Park, Harshit Sikchi, Joey Hejna, Bradley Knox, Chelsea Finn, and Scott Niekum. Scaling laws for reward model overoptimization in direct alignment algorithms. *NeurIPS*, 2024a. arXiv:2406.02900.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. From r to q^*: Your language model is secretly a q-function. *COLM*, 2024b. arXiv:2404.12358.

Tarun Raheja and Nilay Pochhi. From RLHF to direct alignment: A theoretical unification of preference learning for large language models. *arXiv preprint arXiv:2601.06108*, 2026. arXiv:2601.06108.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2020. arXiv:1910.02054.

Alexandre Ramé, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. WARM: On the benefits of weight averaged reward models. *ICML*, 2024. arXiv:2401.12187.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018. arXiv:1803.11485.

Noam Razin, Sadhika Malladi, Adithya Bhaskar, Danqi Chen, Sanjeev Arora, and Boris Hanin. Unintentional unalignment: Likelihood displacement in direct preference optimization. *ICLR*, 2025. arXiv:2410.08847.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *ICML*, 2019. arXiv:1902.10811.

ReLIFT. Relift: Learning what reinforcement learning can't — interleaved online fine-tuning for hardest questions. *arXiv preprint arXiv:2506.07527*, 2025. URL https://arxiv.org/abs/2506.07527.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Wang, and Yuxiong He. ZeRO-Offload: Democratizing billion-scale model training. In *USENIX Annual Technical Conference (ATC)*, 2021. arXiv:2101.06840.

Richard Ren, Steven Basart, Adam Khoja, Alice Gatti, Long Phan, Xuwang Yin, Mantas Mazeika, Alexander Pan, Gabriel Mukobi, Ryan H. Kim, Stephen Fitz, and Dan Hendrycks. Safetywashing: Do AI safety benchmarks actually measure safety progress? *NeurIPS Datasets and Benchmarks*, 2024. arXiv:2407.21792.

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint*, 2025. arXiv:2504.21801.

Reward-Gated On-Policy Distillation. Reward-gated on-policy distillation. *arXiv preprint*, July 2026. arXiv:2607.04037.

MohammadHossein Rezaei et al. Rubric-guided self-distillation: Post-training without rubric verifiers. *arXiv preprint*, 2026. arXiv:2606.12507.

RLSVR. From RLVR to RLSVR: Task transformation induces self-verifiable rewards for open-ended LLM self-improvement. *arXiv preprint*, 2026. arXiv:2607.23802.

Alexander Robey, Eric Wong, Hamed Hassani, and George J. Pappas. SmoothLLM: Defending large language models against jailbreaking attacks. *arXiv preprint*, 2023. arXiv:2310.03684.

Fabien Roger and Ryan Greenblatt. Preventing language models from hiding their reasoning. *arXiv preprint*, 2023. arXiv:2310.18512.

Domenic Rosati, Jan Wehner, Kai Williams, Łukasz Bartoszcze, David Atanasov, Robie Gonzales, Subhabrata Majumdar, Carsten Maple, Hassan Sajjad, and Frank Rudzicz. Representation noising: A defence mechanism against harmful finetuning. *NeurIPS*, 2024. arXiv:2405.14577.

Domenic Rosati, Ali Dadsetan, Hong Huang, Xijie Zeng, et al. Distribution-specific curvature control with finite-sample guarantees for open-weight safety. *arXiv preprint*, 2026. arXiv:2607.22929.

Christopher D. Rosin. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3):203–230, 2011.

Stéphane Ross and J. Andrew Bagnell. Efficient reductions for imitation learning. In *AISTATS*, 2010.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011. arXiv:1011.0686.

Corby Rosset, Ching-An Cheng, Arindam Mitra, Michael Santacroce, Ahmed Awadallah, and Tengyang Xie. Direct nash optimization: Teaching language models to self-improve with general preferences. *arXiv preprint arXiv:2404.03715*, 2024. arXiv:2404.03715.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. *NAACL*, 2024. arXiv:2308.01263.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. *ICLR*, 2024. arXiv:2309.15817.

Bharat Runwal, Ashish Agrawal, Anurag Roy, and Rameswar Panda. PRISM: Demystifying retention and interaction in mid-training. *arXiv preprint*, 2026. arXiv:2603.17074.

Mark Russinovich, Ahmed Salem, and Ronen Eldan. Great, now write an article about that: The Crescendo multi-turn LLM jailbreak attack. *USENIX Security*, 2025. arXiv:2404.01833.

Mikayel Samvelyan, Sharath Chandra Raparthy, Andrei Lupu, Eric Hambro, Aram H. Markosyan, Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, Tim Rocktäschel, and Roberta Raileanu. Rainbow teaming: Open-ended generation of diverse adversarial prompts. *NeurIPS*, 2024. arXiv:2402.16822.

Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. Beyond Chinchilla-optimal: Accounting for inference in language model scaling laws. In *ICML*, 2024. arXiv:2401.00448.

Rylan Schaeffer, Joshua Kazdan, John Hughes, Jordan Juravsky, Sara Price, Aengus Lynch, Erik Jones, Robert Kirk, Azalia Mirhoseini, and Sanmi Koyejo. How do large language monkeys get their power (laws)? *ICML*, 2025. arXiv:2502.17578.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *NeurIPS*, 2023. arXiv:2302.04761.

John Schulman. Approximating KL divergence, 2020. `http://joschu.net/blog/kl-approx.html`.

John Schulman and Thinking Machines Lab. LoRA without regret, 2025. `https://thinkingmachines.ai/blog/lora/`.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015. arXiv:1502.05477.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *ICLR*, 2016. arXiv:1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint*, 2017. arXiv:1707.06347.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design, or: How i learned to start worrying about prompt formatting. In *ICLR*, 2024. arXiv:2310.11324.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *ICLR*, 2018. arXiv:1708.00489.

ServiceNow Research. PipelineRL: asynchronous RL with in-flight weight updates. https://github.com/ServiceNow/PipelineRL, 2025. Swaps weights between forward passes so that generation continues uninterrupted; used by ScaleRL and by OLMo 3 RLVR.

Pier Giuseppe Sessa, Robert Dadashi, Léonard Hussenot, Johan Ferret, Nino Vieillard, Alexandre Ramé, Bobak Shariari, Sarah Perrin, Abe Friesen, Geoffrey Cideron, et al. Bond: Aligning LLMs with best-of-n distillation. *arXiv preprint arXiv:2407.14622*, 2024. arXiv:2407.14622.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *ICLR*, 2025a. arXiv:2410.08146.

Amrith Setlur, Nived Rajaraman, Sergey Levine, and Aviral Kumar. Scaling test-time compute without verification or RL is suboptimal. *ICML*, 2025b. arXiv:2502.12118.

Amrith Setlur, Matthew Y. R. Yang, Charlie Snell, Jeremy Greer, et al. e3: Learning to explore enables extrapolation of test-time compute for llms. *arXiv preprint arXiv:2506.09026*, June 2025c. v2 2025-06-13.

SGLang Project. Sglang for rl (advanced features), 2026. URL https://github.com/sgl-project/sglang/blob/main/docs/advanced_features/sglang_for_rl.md.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, et al. Can large reasoning models self-train? *arXiv preprint arXiv:2505.21444*, May 2025. v2 2025-10-08.

Ning Shang et al. rStar2-Agent: Agentic reasoning technical report. *arXiv preprint arXiv:2508.20722*, 2025.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, et al. Spurious rewards: Rethinking training signals in rlvr. *arXiv preprint arXiv:2506.10947*, June 2025. v2 2026-02-25.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint*, 2024. arXiv:2402.03300.

Itai Shapira et al. How RLHF amplifies sycophancy. *arXiv preprint arXiv:2602.01002*, 2026. arXiv:2602.01002.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R. Bowman, Esin Durmus, Zac Hatfield-Dodds, Scott R. Johnston, Shauna Kravec, Timothy Maxwell, Sam McCandlish, Kamal Ndousse, Oliver Rausch, Nicholas Schiefer, Da Yan, Miranda Zhang, and Ethan Perez. Towards understanding sycophancy in language models. *ICLR*, 2024. arXiv:2310.13548.

Mrinank Sharma, Meg Tong, Jesse Mu, Jerry Wei, Jorrit Kruthoff, Scott Goodfriend, et al. Constitutional classifiers: Defending against universal jailbreaks across thousands of hours of red teaming. *arXiv preprint*, 2025. arXiv:2501.18837.

Jingyan Shen, Ang Li, Salman Rahman, Yifan Sun, et al. Understanding reasoning from pretraining to post-training. *arXiv preprint arXiv:2607.16097*, July 2026a.

Qijia Shen et al. SETA: Scaling environments for terminal agents. *arXiv preprint*, 2026b. arXiv:2607.10891.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less. *arXiv preprint arXiv:2509.04259*, September 2025.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. *EuroSys*, 2025. arXiv:2409.19256.

Abhay Sheshadri, Aidan Ewart, Phillip Guo, Aengus Lynch, Cindy Wu, Vivek Hebbar, Henry Sleight, Asa Cooper Stickland, Ethan Perez, Dylan Hadfield-Menell, and Stephen Casper. Latent adversarial training improves robustness to persistent harmful behaviors in LLMs. *arXiv preprint*, 2024. arXiv:2407.15549.

Tianneng Shi, Jingxuan He, Zhun Wang, Hongwei Li, Linyu Wu, Wenbo Ge, Chengquan Zhang, and Dawn Song. Progent: Securing AI agents with privilege control. *arXiv preprint*, 2025. arXiv:2504.11703.

Zhengyan Shi, Sander Land, Acyr Locatelli, Matthieu Geist, and Max Bartolo. Understanding likelihood over-optimisation in direct alignment algorithms. *arXiv preprint arXiv:2410.11677*, 2024a. arXiv:2410.11677.

Zhengyan Shi, Adam X. Yang, Bin Wu, Laurence Aitchison, Emine Yilmaz, and Aldo Lipani. Instruction tuning with loss over instructions. In *NeurIPS*, 2024b. arXiv:2405.14394.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023. arXiv:2303.11366.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint*, 2019. arXiv:1909.08053.

Vaishnavi Shrivastava, Ahmed Awadallah, Vidhisha Balachandran, Shivam Garg, et al. Sample more to think less: Group filtered policy optimization for concise reasoning. *arXiv preprint arXiv:2508.09726*, August 2025.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget. *arXiv preprint*, 2023. arXiv:2305.17493.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, et al. Beyond human data: Scaling self-training for problem-solving with language models. *TMLR*, 2024. arXiv:2312.06585.

Joykirat Singh et al. Agentic reasoning and tool integration for LLMs via reinforcement learning. *arXiv preprint arXiv:2505.01441*, 2025a. ARTIST.

Shivalika Singh, Yiyang Nan, Alex Wang, Daniel D'Souza, Sayash Kapoor, Ahmet Üstün, Sanmi Koyejo, Yuntian Deng, Shayne Longpre, Noah A. Smith, Beyza Ermis, Marzieh Fadaee, and Sara Hooker. The leaderboard illusion. *arXiv preprint*, 2025b. arXiv:2504.20879.

Prasann Singhal, Tanya Goyal, Jiacheng Xu, and Greg Durrett. A long way to go: Investigating length correlations in RLHF. *COLM*, 2024. arXiv:2310.03716.

Anand Siththaranjan, Cassidy Laidlaw, and Dylan Hadfield-Menell. Distributional preference learning: Understanding and accounting for hidden context in RLHF. *ICLR*, 2024. arXiv:2312.08358.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. *NeurIPS*, 2022. arXiv:2209.13085.

SkyRL Project. Off-policy correction in skyrl, 2026. URL `https://docs.skyrl.ai/docs/algorithms/off_policy_correction`.

Yonatan Slutzky, Yotam Alexander, Tomer Slor, Yoav Nagel, et al. Why does agentic safety fail to generalize across tasks? *arXiv preprint*, 2026. arXiv:2605.06992.

Andries Smit, Paul Duckworth, Nathan Grinsztajn, Thomas D. Barrett, and Arnu Pretorius. Should we be going MAD? a look at multi-agent debate strategies for LLMs. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. arXiv:2311.17371.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *ICLR*, 2025. arXiv:2408.03314.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-Searcher: Incentivizing the search capability in LLMs via rein-

forcement learning. *arXiv preprint*, 2025a. arXiv:2503.05592.

Huatong Song et al. R1-Searcher++: Incentivizing the dynamic knowledge acquisition of LLMs via reinforcement learning. *arXiv preprint arXiv:2505.17005*, 2025b.

Huatong Song et al. SWE-Master: Unleashing the potential of software engineering agents via post-training. *arXiv preprint arXiv:2602.03411*, 2026.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. PRMBench: A fine-grained and challenging benchmark for process-level reward models. In *ACL*, 2025c. arXiv:2501.03124.

Taylor Sorensen, Jared Moore, Jillian Fisher, Mitchell Gordon, Niloofar Mireshghallah, Christopher Michael Rytting, Andre Ye, Liwei Jiang, Ximing Lu, Nouha Dziri, Tim Althoff, and Yejin Choi. A roadmap to pluralistic alignment. *arXiv preprint arXiv:2402.05070*, February 2024. arXiv:2402.05070; submitted 2024-02-07, v3 2024-08-20.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and Sam Toyer. A StrongREJECT for empty jailbreaks. *NeurIPS Datasets and Benchmarks*, 2024. arXiv:2402.10260.

Stabilizing RL. Stabilizing reinforcement learning with llms: Formulation and practices. *arXiv preprint arXiv:2512.01374*, 2025. URL https://arxiv.org/abs/2512.01374.

Stanford University and Laude Institute. Terminal-bench 2.0 / 2.1. URL https://www.tbench.ai/.

Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. GPT-4 doesn't know it's wrong: An analysis of iterative prompting for reasoning problems. In *NeurIPS Workshop on Instruction Tuning and Instruction Following*, 2023. arXiv:2310.12397.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize with human feedback. In *NeurIPS*, 2020. arXiv:2009.01325.

Benedikt Stroebl, Sayash Kapoor, and Arvind Narayanan. The limits of inference scaling through resampling. *arXiv preprint*, 2024. arXiv:2411.17501; earlier titled "Inference Scaling Flaws: The Limits of LLM Resampling with Imperfect Verifiers".

Liangcai Su et al. Scaling agents via continual pre-training. *arXiv preprint arXiv:2509.13310*, 2025.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint*, 2025. arXiv:2503.16419.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems* (*NeurIPS*), 2016. arXiv:1605.07736.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations* (*ICLR*), 2018. arXiv:1703.05407.

Hao Sun et al. ZeroSearch: Incentivize the search capability of LLMs without searching. *arXiv preprint arXiv:2505.04588*, 2025.

Mengyuan Sun et al. Support vector rubrics: Closing the gap between self-generated and human rubrics. *arXiv preprint*, 2026. arXiv:2606.08077.

Zhiqing Sun, Sheng Shen, Shengcao Cao, Haotian Liu, Chunyuan Li, Yikang Shen, Chuang Gan, Liang-Yan Gui, Yu-Xiong Wang, Yiming Yang, Kurt Keutzer, and Trevor Darrell. Aligning large multimodal models with factually augmented RLHF. *arXiv preprint arXiv:2309.14525*, September 2023a. arXiv:2309.14525; submitted 2023-09-25.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. Principle-driven self-alignment of language models from scratch with minimal human supervision. *arXiv preprint arXiv:2305.03047*, 2023b. arXiv:2305.03047.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (*AAMAS*), 2018. arXiv:1706.05296.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999. The options framework; source of the semi-Markov abstraction used for the turn-level formulation.

Gokul Swamy, Christoph Dann, Rahul Kidambi, Zhiwei Steven Wu, and Alekh Agarwal. A minimaximalist approach to reinforcement learning from human feedback. *arXiv preprint arXiv:2401.04056*, 2024. arXiv:2401.04056.

Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J. Andrew Bagnell. All roads lead to likelihood: The value of reinforcement learning in fine-tuning. *arXiv preprint arXiv:2503.01067*, 2025. arXiv:2503.01067.

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of LLMs should leverage suboptimal, on-policy data. *arXiv preprint arXiv:2404.14367*, April 2024. arXiv:2404.14367.

Wendy K. Tam et al. The neutral mask: How RLHF provides shallow alignment while leaving partisan structure intact in a large language model. *arXiv preprint*, 2026. arXiv:2606.09735.

Rishub Tamirisa, Bhrugu Bharathi, Long Phan, Andy Zhou, Alice Gatti, Tarun Suresh, Maxwell Lin, Justin Wang, Rowan Wang, Ron Arel, Andy Zou, Dawn Song, Bo Li, Dan Hendrycks, and Mantas Mazeika. Tamper-resistant safeguards for open-weight LLMs. *ICLR*, 2025. arXiv:2408.00761.

Xinyu Tang, Qianggang Cao, Yurou Liu, Yuliang Zhan, et al. Ring-zero: Scaling zero rl to a trillion parameters for emergent reasoning. *arXiv preprint arXiv:2607.12395*, July 2026. v2 2026-07-16.

Yunhao Tang, Daniel Zhaohan Guo, Zeyu Zheng, Daniele Calandriello, Yuan Cao, Eugene Tarassov, Rémi Munos, Bernardo Ávila Pires, Michal Valko, Yong Cheng, and Will Dabney. Understanding the performance gap between online and offline alignment algorithms. *arXiv preprint arXiv:2405.08448*, May 2024a. arXiv:2405.08448.

Yunhao Tang, Zhaohan Daniel Guo, Zeyu Zheng, Daniele Calandriello, Rémi Munos, Mark Rowland, Pierre Harvey Richemond, Michal Valko, Bernardo Ávila Pires, and Bilal Piot. Generalized preference optimization: A unified approach to offline alignment. *ICML*, 2024b. arXiv:2402.05749.

Zhengwei Tao et al. WebShaper: Agentically data synthesizing via information-seeking formalization. *arXiv preprint arXiv:2507.15061*, 2025.

Team OLMo. OLMo 3. *arXiv preprint*, 2025. arXiv:2512.13961.

Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

The gVisor Authors. gVisor: A container sandbox runtime. `https://gvisor.dev/docs/`, 2026. User-space kernel intercepting application syscalls; documentation, not a paper.

Thinking Machines Lab. Tinker: A training api for researchers and developers, 2025. URL `https://thinkingmachines.ai/tinker/`. Docs: https://tinker-docs.thinkingmachines.ai/; announcement: https://thinkingmachines.ai/news/announcing-tinker/.

Thinking Machines Lab. Lora primer (tinker documentation), 2026. URL `https://tinker-docs.thinkingmachines.ai/tinker/lora-primer/`.

Christos Thrampoulidis, Sadegh Mahdavi, and Wenlong Deng. Advantage shaping as surrogate reward maximization: Unifying pass@k policy gradients. *arXiv preprint arXiv:2510.23049*, October 2025. v3 2026-03-21.

THUDM / Z.ai. slime: An llm post-training framework for rl scaling, 2026. URL https://github.com/THUDM/slime.

L. L. Thurstone. A law of comparative judgment. *Psychological Review*, 34:273–286, 1927.

Yijun Tian, Shaoyu Chen, Zhichao Xu, Yawei Wang, et al. Reinforcement mid-training. *arXiv preprint arXiv:2509.24375*, September 2025.

Zihang Tian et al. ARCO: Adaptive rubrics with co-evolution for multi-step llm-based agents. *arXiv preprint*, 2026. arXiv:2606.21262.

TMAX. Tmax: A simple recipe for terminal agents. *arXiv preprint arXiv:2606.23321*, 2026. URL https://arxiv.org/abs/2606.23321.

Tongyi DeepResearch Team. Tongyi DeepResearch technical report. *arXiv preprint arXiv:2510.24701*, 2025.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint*, 2023. arXiv:2307.09288.

Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Dunwei Tu et al. ScaleEnv: Scaling environment synthesis from scratch for generalist interactive tool-use agent training. *arXiv preprint*, 2026. arXiv:2602.06820.

Emre Turan et al. Oversight has a capacity: Calibrating agent guards to a subjective, fatiguing human. *arXiv preprint*, 2026. arXiv:2606.08919.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *NeurIPS*, 2023. arXiv:2305.04388.

Gladys Tyen, Hassan Mansoor, Victor Cărbune, Peter Chen, and Tony Mak. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of ACL*, 2024. arXiv:2311.08516.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint*, 2022. arXiv:2211.14275.

verl Project. Fully async policy trainer, 2026a. URL https://verl.readthedocs.io/en/latest/advance/fully_async.html.

verl Project. verl v0.7 release blog: Hybrid-controller architecture and rollout server mode, 2026b. URL https://verl.readthedocs.io/en/latest/blog/v0.7.html.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575:350–354, 2019.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Ma. Checklists are better than reward models for aligning language models. *arXiv preprint*, 2025. arXiv:2507.18624.

vLLM Project. Sleep mode (vllm documentation), 2025. URL https://docs.vllm.ai/en/latest/features/sleep_mode/.

vLLM Project. Weight transfer (vllm documentation), 2026. URL https://docs.vllm.ai/en/stable/training/weight_transfer/.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Christian Walder and Deep Karkhanis. Pass@k policy optimization: Solving harder reinforcement learning problems. *arXiv preprint arXiv:2505.15201*, May 2025. v5 2026-06-10.

Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. The instruction hierarchy: Training LLMs to prioritize privileged instructions. *arXiv preprint*, 2024. arXiv:2404.13208.

Ziyu Wan et al. ReMA: Learning to meta-think for LLMs with multi-agent reinforcement learning. *arXiv preprint arXiv:2503.09501*, 2025. NeurIPS 2025.

Beining Wang et al. Co-evolving llm evaluators and policies via dynamicrubric. *arXiv preprint*, 2026a. arXiv:2607.20083.

Chaoqi Wang, Yibo Jiang, Chenghao Yang, Han Liu, and Yuxin Chen. Beyond reverse KL: Generalizing direct preference optimization with diverse divergence constraints. *ICLR*, 2024a. arXiv:2309.16240.

Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. *EMNLP*, 2024b. arXiv:2406.12845.

Hongru Wang et al. Acting less is reasoning more! teaching model to act efficiently. *arXiv preprint arXiv:2504.14870*, 2025a. OTC-PO.

Jianze Wang et al. SERPO: Self-evolving rubric policy optimization for open-ended test-time reinforcement learning. *arXiv preprint*, 2026b. arXiv:2607.26873.

Ke Wang, Jiahui Zhu, Minjie Ren, Zeming Liu, Shiwei Li, Zongye Zhang, Chenkai Zhang, Xiaoyu Wu, Qiqi Zhan, Qingjie Liu, and Yunhong Wang. A survey on data synthesis and augmentation for large language models. *arXiv preprint*, 2024c. arXiv:2410.12896.

Meng Wang, Haohan Zhao, Wenzhuo Liu, Lu Yang, Geng Liu, Haiyang Guo, Guo-Sen Xie, Gaofeng Meng, Hongbin Liu, and Fei Zhu. Denser ≠ better: Limits of on-policy self-distillation for continual post-training. *arXiv preprint arXiv:2607.01763*, July 2026c. arXiv:2607.01763; submitted 2026-07-02.

Mingzhi Wang, Chengdong Ma, Qizhi Chen, Linjian Meng, Yang Han, Jiancong Xiao, Zhaowei Zhang, Jing Huo, Weijie J. Su, and Yaodong Yang. Magnetic preference optimization: Achieving last-iterate convergence for language model alignment. *arXiv preprint arXiv:2410.16714*, 2024d. arXiv:2410.16714.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. *ACL*, 2024e. arXiv:2305.17926.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. *ACL*, 2024f. arXiv:2312.08935.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. Rethinking the bounds of LLM reasoning: Are multi-agent discussions the key? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6106–6131, 2024g. arXiv:2402.18272.

Renxi Wang et al. AgentFly: Extensible and scalable reinforcement learning for LM agents. *arXiv preprint arXiv:2507.14897*, 2025b.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired open-ended trailblazer (POET): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint*, 2019. arXiv:1901.01753.

Shenzhi Wang, Le Yu, Chang Gao, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. *arXiv preprint*, 2025c. arXiv:2506.01939.

Xuezhi Wang and Denny Zhou. Chain-of-thought reasoning without prompting. In *NeurIPS*, 2024. arXiv:2402.10200.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023a. arXiv:2203.11171.

Yinjie Wang et al. Co-evolving LLM coder and unit tester via reinforcement learning. *arXiv preprint arXiv:2506.03136*, 2025d. CURE.

Yiping Wang et al. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*, 2025e.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *ACL*, 2023b. arXiv:2212.10560.

Yuanyi Wang, Yifan Yang, Su Lu, Yanggan Gu, Pengkai Wang, Wenjun Wang, Zhaoyi Yan, Congkai Xie, Jianmin Wu, Jialun Cao, Shing-Chi Cheung, and Hongxia Yang. Geometry conflict: Explaining and controlling forgetting in LLM continual post-training. *arXiv preprint arXiv:2605.09608*, May 2026d. arXiv:2605.09608; submitted 2026-05-10.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Thoughts are all over the place: On the underthinking of o1-like LLMs. *arXiv preprint*, 2025f. arXiv:2501.18585.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. OctoThinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint*, 2025g. arXiv:2506.20512.

Zhaoyang Wang et al. Agent world model: Infinity synthetic environments for agentic reinforcement learning. *arXiv preprint arXiv:2602.10090*, 2026e.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. HelpSteer2: Open-source dataset for training top-performing reward models. *NeurIPS*, 2024h. arXiv:2406.08673.

Zhilin Wang, Alexander Bukharin, Olivier Delalleau, Daniel Egert, Gerald Shen, Jiaqi Zeng, Oleksii Kuchaiev, and Yi Dong. HelpSteer2-Preference: Complementing ratings with preferences. *ICLR*, 2025h. arXiv:2410.01257.

Zihan Wang et al. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025i.

Zihan Wang et al. RAGEN-2: Reasoning collapse in agentic RL. *arXiv preprint arXiv:2604.06268*, 2026f.

Ziliang Wang et al. StepSearch: Igniting LLMs search ability via step-wise proximal policy optimization. *arXiv preprint arXiv:2505.15107*, 2025j.

Weak-to-Strong On-Policy Distillation. Weak-to-strong generalization via direct on-policy distillation. *arXiv preprint*, July 2026. arXiv:2607.05394.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? *NeurIPS*, 2023. arXiv:2307.02483.

Boyi Wei, Kaixuan Huang, Yangsibo Huang, Tinghao Xie, Xiangyu Qi, Mengzhou Xia, Prateek Mittal, Mengdi Wang, and Peter Henderson. Assessing the brittleness of safety alignment via pruning and low-rank modifications. *ICML*, 2024. arXiv:2402.05162.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *ICLR*, 2022a. arXiv:2109.01652.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022b. arXiv:2201.11903.

Jason Wei et al. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025a. URL https://arxiv.org/abs/2504.12516.

Quan Wei et al. Reinforcing multi-turn reasoning in LLM agents via turn-level reward design. *arXiv preprint arXiv:2505.11821*, 2025b.

Stanley Wei and Juno Kim. Provable benefits of rlvr over sft for reasoning models: Learning to backtrack efficiently. *arXiv preprint arXiv:2606.22938*, June 2026.

Yifan Wei et al. AutoTIR: Autonomous tools integrated reasoning via reinforcement learning. *arXiv preprint arXiv:2507.21836*, 2025c.

Yuxiang Wei et al. SWE-RL: Advancing LLM reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*, 2025d.

Ying Wen. When agents learn from the world, not from us. Personal blog, March 2026, 2026. Source for the memory-to-skill compression pipeline (extraction, composition, forward and backward distillation).

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. J1: Incentivizing thinking in LLM-as-a-Judge via reinforcement learning. *arXiv preprint*, 2025. arXiv:2505.10320.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

David H. Wolpert and Kagan Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, et al. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*, 2022a. arXiv:2203.05482.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, et al. Robust fine-tuning of zero-shot models. In *CVPR*, 2022b. arXiv:2109.01903.

Di Wu, Chengshuai Shi, Jing Yang, and Cong Shen. f-divergence regularized RLHF: Two tales of sampling and unified analyses. *arXiv preprint arXiv:2605.06977*, 2026. arXiv:2605.06977.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, et al. The invisible leash: Why rlvr may or may not escape its origin. *arXiv preprint arXiv:2507.14843*, July 2025a. v4 2026-02-04.

Fang Wu et al. DeepSearch: Overcome the bottleneck of reinforcement learning with verifiable rewards via monte carlo tree search. *arXiv preprint arXiv:2509.25454*, 2025b.

Jialong Wu et al. WebDancer: Towards autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*, 2025c.

Jialong Wu et al. WebWalker: Benchmarking LLMs in web traversal. *arXiv preprint arXiv:2501.07572*, 2025d.

Junkang Wu, Xue Wang, Zhengyi Yang, Jiancan Wu, Jinyang Gao, Bolin Ding, Xiang Wang, and Xiangnan He. Alphadpo: Adaptive reward margin for direct preference optimization. *arXiv preprint arXiv:2410.10148*, 2024a. arXiv:2410.10148.

Junkang Wu, Yuexiang Xie, Zhengyi Yang, Jiancan Wu, Jinyang Gao, Bolin Ding, Xiang Wang, and Xiangnan He. β-dpo: Direct preference optimization with dynamic β. *arXiv preprint arXiv:2407.08639*, 2024b. arXiv:2407.08639.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. In *ICML*, 2025e. arXiv:2408.00724.

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. Self-play preference optimization for language model alignment. *ICLR*, 2025f. arXiv:2405.00675.

Zhiheng Xi et al. AgentGym-RL: Training LLM agents for long-horizon decision making through multi-turn reinforcement learning. *arXiv preprint arXiv:2509.08755*, 2025.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *ICML*, 2024. arXiv:2402.04333.

Violet Xiang, Amrith Setlur, Chase Blagden, Nick Haber, et al. Exprl: Exploratory rl for llm mid-training. *arXiv preprint arXiv:2606.17024*, June 2026.

Teng Xiao, Yige Yuan, Huaisheng Zhu, Mingxiao Li, and Vasant G. Honavar. Cal-dpo: Calibrated direct preference optimization for language model alignment. *arXiv preprint*

arXiv:2412.14516, 2024. arXiv:2412.14516.

Tengyang Xie, Dylan J. Foster, Akshay Krishnamurthy, Corby Rosset, Ahmed Awadallah, and Alexander Rakhlin. Exploratory preference optimization: Harnessing implicit q*-approximation for sample-efficient RLHF. *ICLR*, 2025. arXiv:2405.21046.

Tianbao Xie et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024.

Yingsha Xie, Tiansheng Huang, Enneng Yang, Rui Min, et al. Mitigating safety tax via distribution-grounded refinement in large reasoning models. *arXiv preprint*, 2026. arXiv:2602.02136.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Self-evaluation guided beam search for reasoning. In *NeurIPS*, 2023. arXiv:2305.00633.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for RLHF under KL-constraint. *arXiv preprint arXiv:2312.11456*, December 2023. arXiv:2312.11456.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. WizardLM: Empowering large language models to follow complex instructions. *ICLR*, 2024a. arXiv:2304.12244.

Frank F. Xu et al. TheAgentCompany: Benchmarking LLM agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161*, 2024b.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. Contrastive preference optimization: Pushing the boundaries of LLM performance in machine translation. *ICML*, 2024c. arXiv:2401.08417.

Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. Is DPO superior to PPO for LLM alignment? a comprehensive study. *ICML*, 2024d. arXiv:2404.10719.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned LLMs with nothing. *ICLR*, 2025. arXiv:2406.08464.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. SimpleTIR: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint arXiv:2509.02479*, 2025.

Zhiyu Xue, Zimo Qi, Guangliang Liu, Bocheng Chen, et al. Deactivating refusal triggers: Understanding and mitigating overrefusal in safety alignment. *arXiv preprint*, 2026.

arXiv:2603.11388.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *NeurIPS*, 2025a. arXiv:2504.14945.

Sikuan Yan et al. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025b.

Haocheng Yang et al. Rubrics on trial: Evolving rubrics from a single query via synthetic pairwise evidence. *arXiv preprint*, 2026a. arXiv:2607.15092.

John Yang et al. SWE-smith: Scaling data for software engineering agents. *arXiv preprint arXiv:2504.21798*, 2025.

Joy Qiping Yang, Salman Salamatian, Ziteng Sun, Ananda Theertha Suresh, and Ahmad Beirami. Asymptotics of language model alignment. In *ISIT*, 2024a. arXiv:2404.01730.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. Rethinking benchmark and contamination for language models with rephrased samples. *arXiv preprint*, 2023a. arXiv:2311.04850.

Wenkai Yang, Xiaohan Bi, Yankai Lin, Sishuo Chen, Jie Zhou, and Xu Sun. Watch out for your agents! investigating backdoor threats to LLM-based agents. *NeurIPS*, 2024b. arXiv:2402.11208.

Xianjun Yang, Xiao Wang, Qi Zhang, Linda Petzold, William Yang Wang, Xun Zhao, and Dahua Lin. Shadow alignment: The ease of subverting safely-aligned language models. *arXiv preprint*, 2023b. arXiv:2310.02949.

Zhiqin Yang, Yonggang Zhang, Wei Xue, Dong Fang, Bo Han, and Yike Guo. Conditional equivalence of DPO and RLHF: Implicit assumption, failure modes, and provable alignment. *arXiv preprint arXiv:2605.20834*, 2026b. arXiv:2605.20834.

Feng Yao, Liyuan Liu, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. On the rollout-training mismatch in modern rl systems. In *NeurIPS 2025 Workshop on Optimization for Machine Learning* (*OPT*), 2025a. URL https://opt-ml.org/papers/2025/paper116.pdf. Blog: https://fengyao.notion.site/off-policy-rl.

Feng Yao, Liyuan Liu, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. Your efficient rl framework secretly brings you off-policy rl training. https://fengyao.notion.site/off-policy-rl, August 2025b. First published 2025-08-05, updated 2025-10-13. Workshop version: "On the Rollout-Training Mismatch in Modern RL Systems", NeurIPS 2025 Workshop (OpenReview xsonlP8DRV). No arXiv ID. Origin of TIS for rollout-training mismatch.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*, 2023a. arXiv:2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. *ICLR*, 2023b. arXiv:2210.03629.

Rui Ye et al. AgentFold: Long-horizon web agents with proactive context management. *arXiv preprint arXiv:2510.24699*, 2025a.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. LIMO: Less is more for reasoning. *arXiv preprint*, 2025b. arXiv:2502.03387.

Zheng-Xin Yong, Cristina Menghini, and Stephen H. Bach. Low-resource languages jailbreak GPT-4. *arXiv preprint*, 2023. arXiv:2310.02446.

Richard J. Young et al. Measuring faithfulness depends on how you measure: Classifier sensitivity in LLM chain-of-thought evaluation. *arXiv preprint*, 2026. arXiv:2603.20172.

Robin Young. Why is RLHF alignment shallow? a gradient analysis. *arXiv preprint arXiv:2603.04851*, 2026. arXiv:2603.04851.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Advances in Neural Information Processing Systems* (*Datasets and Benchmarks Track*), 2022a. arXiv:2103.01955.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for transformer-based generative models. In *USENIX Symposium on Operating Systems Design and Implementation* (*OSDI*), 2022b.

Hongli Yu et al. MemAgent: Reshaping long-context LLM with multi-conv RL-based memory agent. *arXiv preprint arXiv:2507.02259*, 2025a.

Peng Yu, Zhen Zhao, Shengtao Zhang, Luoyi Fu, Xinbing Wang, and Ying Wen. Structured in-context environment scaling for large language model reasoning. *arXiv preprint arXiv:2509.23330*, 2025b. ICLR 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint*, 2025c. arXiv:2503.14476.

Yang Yu. Pass@k metric for rlvr: A diagnostic tool of exploration, but not an objective. *arXiv preprint arXiv:2511.16231*, November 2025.

Zhaochen Yu et al. Demystifying reinforcement learning in agentic reasoning. *arXiv preprint arXiv:2510.11701*, 2025d.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. In *ICML*, 2025a. arXiv:2412.01981.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *ICML*, 2024a. arXiv:2401.10020.

Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. GPT-4 is too smart to be safe: Stealthy chat with LLMs via cipher. *ICLR*, 2024b. arXiv:2308.06463.

Yufeng Yuan, Yu Yue, Ruofei Zhu, Tiantian Fan, et al. What's behind ppo's collapse in long-cot? value optimization holds the secret. *arXiv preprint arXiv:2503.01491*, March 2025b.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint*, 2023a. arXiv:2308.01825.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. RRHF: Rank responses to align language models with human feedback without tears. *NeurIPS*, 2023b. arXiv:2304.05302.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? *arXiv preprint*, 2025a. arXiv:2504.13837.

Yu Yue et al. VAPO: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025b.

Zai. GLM-5.2: Built for long-horizon tasks. Engineering blog, `https://z.ai/blog/glm-5.2`, 2026. Blog post; no archival preprint. Source for the reported switch from group-relative advantages to a critic under trajectory compaction.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *NeurIPS*, 2022. arXiv:2203.14465.

Huaye Zeng et al. AceCoder: Acing coder RL via automated test-case synthesis. *arXiv preprint arXiv:2502.01718*, 2025.

Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia, and Weiyan Shi. How Johnny can persuade LLMs to jailbreak them: Rethinking persuasion to challenge AI safety by humanizing LLMs. *ACL*, 2024a. arXiv:2401.06373.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. Token-level direct preference optimization. *arXiv preprint arXiv:2404.11999*, 2024b. arXiv:2404.11999.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. *ACL Findings*, 2024. arXiv:2403.02691.

Bo-Wen Zhang et al. CoRT: Counterfactual replay for token-level rubric-guided policy optimization. *arXiv preprint*, 2026a. arXiv:2607.25659.

Charlie Zhang, Graham Neubig, and Xiang Yue. On the interplay of pre-training, mid-training, and RL on reasoning language models. *arXiv preprint*, 2025a. arXiv:2512.07783.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhong-Zhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Francisco Piedrahita Velez, Yue Liao, Hongru Wang, Mengyue Yang, Heng Ji, Jun Wang, Shuicheng Yan, Philip Torr, and Lei Bai. The landscape of agentic reinforcement learning for LLMs: A survey. *Transactions on Machine Learning Research*, 2026b. Survey Certification. arXiv:2509.02547.

Hanchen Zhang et al. AgentRL: Scaling agentic reinforcement learning with a multi-turn, multi-task framework. *arXiv preprint arXiv:2510.04206*, 2025b.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Charlotte Zhuang, Dylan Slack, Qin Lyu, Sean Hendryx, Russell Kaplan, Michele Lunati, and Summer Yue. A careful examination of large language model performance on grade school arithmetic. *arXiv preprint*, 2024a. arXiv:2405.00332.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *ICLR*, 2025c. arXiv:2408.15240.

Qizheng Zhang et al. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025d.

Shenao Zhang, Donghan Yu, Hiteshi Sharma, Han Zhong, Zhihan Liu, Ziyi Yang, Shuohang Wang, Hany Hassan, and Zhaoran Wang. Self-exploring language models: Active preference elicitation for online alignment. *arXiv preprint arXiv:2405.19332*, 2024b. arXiv:2405.19332.

Shengtao Zhang et al. MemRL: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026c.

Shun Zhang, Zhenfang Chen, Sunli Chen, Yikang Shen, Zhiqing Sun, and Chuang Gan. Improving reinforcement learning from human feedback with efficient reward model ensemble. *arXiv preprint*, 2024c. arXiv:2401.16635.

Tong Zhang, Zexin Li, Simin Chen, Yun Peng, et al. When LLM defenses backfire: Characterizing safety, performance, and cost trade-offs. *arXiv preprint*, 2026d. arXiv:2607.24392.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint*, 2025e. arXiv:2501.07301; the Qwen2.5-Math-PRM report.

Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-SafetyBench: Evaluating the safety of LLM agents. *arXiv preprint*, 2024d. arXiv:2412.14470.

Andrew Zhao et al. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025a.

Chenyang Zhao et al. Rl training-inference mismatch: Truly on-policy vs. algorithmic mitigation. Awesome-ML-SYS-Tutorial, 2025b. URL https://github.com/zhaochenyang20/Awesome-ML-SYS-Tutorial/blob/main/rlhf/slime/mismatch/blog-en.md.

Eric Zhao, Pranjal Awasthi, and Sreenivas Gollapudi. Sample, scrutinize and scale: Effective inference-time search by scaling verification. In *NeurIPS*, 2025c. arXiv:2502.01839.

Guanqun Zhao, Zijun Xie, Binbin Zheng, Enlei Gong, et al. Deconstructing off-policy ratios: Entropy-scaled trust regions for asynchronous reinforcement learning. *arXiv preprint arXiv:2607.22186*, July 2026.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, et al. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, May 2025d. v5 2026-05-16.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *arXiv preprint*, 2023a. arXiv:2304.11277.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J. Liu. SLiC-HF: Sequence likelihood calibration with human feedback. *arXiv preprint*, 2023b. arXiv:2305.10425.

Yuzhong Zhao, Yue Liu, Junpeng Liu, Jingye Chen, et al. Geometric-mean policy optimization. *arXiv preprint arXiv:2507.20673*, July 2025e. v3 2025-10-18.

Chujie Zheng, Shixuan Liu, Mingze Li, et al. Group sequence policy optimization. *arXiv preprint*, 2025a. arXiv:2507.18071.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In *ACL*, 2025b. arXiv:2412.06559.

Haizhong Zheng et al. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. *arXiv preprint arXiv:2506.02177*, 2025c.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *NeurIPS Datasets and Benchmarks*, 2023a. arXiv:2306.05685.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, et al. SGLang: Efficient execution of structured language model programs. *NeurIPS*, 2024. arXiv:2312.07104.

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, et al. Secrets of RLHF in large language models part I: PPO. *arXiv preprint arXiv:2307.04964*, July 2023b. arXiv:2307.04964.

Yuxiang Zheng et al. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*, 2025d.

Zhipu AI and Z.ai. GLM-4.5: Agentic, reasoning, and coding (ARC) foundation models. *arXiv preprint*, 2025. arXiv:2508.06471.

Tianle Zhong, Neiwen Ling, Yifan Pi, Zijun Wei, et al. Diagnosing training inference mismatch in llm reinforcement learning. *arXiv preprint arXiv:2605.14220*, May 2026.

Yinmin Zhong et al. StreamRL: Scalable, heterogeneous, and elastic RL for LLMs with disaggregated stream generation. *arXiv preprint arXiv:2504.15930*, 2025.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *ICML*, 2024. arXiv:2310.04406; LATS.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, et al. LIMA: Less is more for alignment. *NeurIPS*, 2023a. arXiv:2305.11206.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint*, 2023b. arXiv:2311.07911.

Shuaiyu Zhou et al. ToolVerse: Unlocking massive environments and long-horizon tasks for agentic reinforcement learning. *arXiv preprint arXiv:2607.15660*, 2026.

Todd Zhou. When rlvr shrinks the reasoning boundary: Diagnosing pass@k inversion. *arXiv preprint arXiv:2607.20543*, July 2026.

Yifei Zhou et al. SWEET-RL: Training multi-turn LLM agents on collaborative reasoning tasks. *arXiv preprint arXiv:2503.15478*, 2025a.

Zijian Zhou et al. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025b.

Banghua Zhu, Michael I. Jordan, and Jiantao Jiao. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. *arXiv preprint arXiv:2301.11270*, 2023. arXiv:2301.11270.

Daniel Zhu, Zihan Wang, Xuchan Bao, Jerry Wei, et al. Jailbroken frontier models retain their capabilities. *arXiv preprint*, 2026a. arXiv:2605.00267.

Hanqing Zhu, Zhenyu Zhang, Hanxian Huang, DiJia Su, et al. The path not taken: Rlvr provably learns off the principals. *arXiv preprint arXiv:2511.08567*, November 2025a.

Hanqing Zhu, Wenyan Cong, Zhizhou Sha, Sagnik Mukherjee, et al. Iso: An rlvr-native optimization stack. *arXiv preprint arXiv:2607.19331*, July 2026b.

Minghang Zhu et al. DeepRubric: Evidence-tree rubric supervision for efficient reinforcement learning of deep research agents. *arXiv preprint*, 2026c. arXiv:2606.17029.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, et al. The surprising effectiveness of negative reinforcement in llm reasoning. *arXiv preprint arXiv:2506.01347*, June 2025b. v2 2025-10-25.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. PMLR 235:62743–62767. arXiv:2402.16823.

Brian D. Ziebart, J. Andrew Bagnell, and Anind K. Dey. Modeling interaction via the principle of maximum causal entropy. In *ICML*, 2010.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint*, 2019. arXiv:1909.08593.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint*, 2023. arXiv:2307.15043.

Andy Zou, Long Phan, Justin Wang, Derek Duenas, Maxwell Lin, Maksym Andriushchenko, Rowan Wang, J. Zico Kolter, Matt Fredrikson, and Dan Hendrycks. Improving alignment and robustness with circuit breakers. *NeurIPS*, 2024. arXiv:2406.04313.

Yuxin Zuo et al. TTRL: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.