

强化学习概览

Reinforcement Learning: A Quick Tour

卢奇

luqi.code@gmail.com

强化学习基础

基于价值的方法

基于策略的方法

Model-Based 与 Multi-Agent

LLM 与强化学习

总结

强化学习基础

强化学习：核心框架

核心思想：Agent 通过与环境交互，学习最大化累积奖励的策略

Markov Decision Process (MDP):

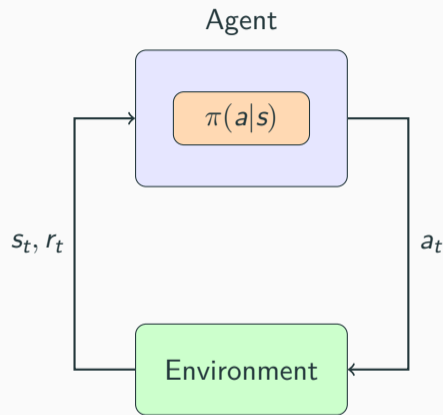
- Markov 性质:

$$P(s_{t+1}, r_t | s_t, a_t) = P(s_{t+1}, r_t | s_{1:t}, a_{1:t})$$

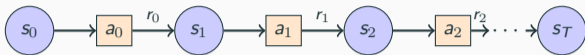
- 给定 (s_t, a_t) ，下一状态和奖励的分布与历史无关

MDP 五元组 $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$:

- \mathcal{S} : 状态空间 \mathcal{A} : 动作空间
- $P(s' | s, a)$: 转移概率 $R(s, a)$: 奖励函数
- $\gamma \in [0, 1]$: 折扣因子 (权衡即时和长期奖励)



交互过程与轨迹 (Trajectory)



轨迹: $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_T)$

Episodic: 有终止 s_T (游戏结束)

Continuing: $T \rightarrow \infty$ (无终止状态)

轨迹概率: $p(\tau|\pi) = p(s_0) \prod_{t=0}^{T-1} \pi(a_t|s_t) P(s_{t+1}|s_t, a_t)$

回报 (Return / Reward-to-go): $G_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$

目标: $\pi^* = \operatorname{argmax}_{\pi} J(\pi)$, 其中

$$J(\pi) = \mathbb{E}_{\tau \sim \pi}[G_0] = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^T \gamma^t r_t \right]$$

核心符号

s_t 状态 (state)

a_t 动作 (action)

r_t 奖励 (即时反馈)

G_t 回报 (累积奖励)

$\pi(a|s)$ 策略 (policy)

价值函数

$V^{\pi}(s)$ $\mathbb{E}[G_t|s_t=s]$

状态价值

$Q^{\pi}(s, a)$ $\mathbb{E}[G_t|s_t=s, a_t=a]$

动作价值

策略与价值函数：怎么评价一条策略？

策略 (**Policy**): Agent 在每个状态下选择动作的规则, RL 目标是**找到最优策略 π^***

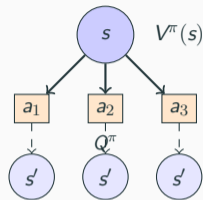
状态价值函数: 从状态 s 出发, 在策略 π 下的期望回报

$$V^\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

动作价值函数: 在 s 采取 a , 之后遵循 π 的期望回报

$$Q^\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

优势函数: $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$
衡量动作相对于平均水平的好坏



$V^\pi(s)$: 状态 s 下的期望回报

$Q^\pi(s, a)$: 在 s 执行 a 的期望回报

基于价值的方法

价值函数的递推关系 (Bellman 方程)

Bellman Expectation (策略 π 下)

$$V^\pi(s) = \mathbb{E}_\pi[r_t + \gamma V^\pi(S_{t+1}) \mid S_t = s]$$

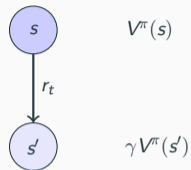
$$Q^\pi(s, a) = \mathbb{E}_\pi[r_t + \gamma Q^\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$$

Bellman Optimality (最优策略 π^*)

$$V^*(s) = \max_a \mathbb{E}[r_t + \gamma V^*(S_{t+1}) \mid S_t = s, A_t = a]$$

$$Q^*(s, a) = \mathbb{E}[r_t + \gamma \max_{a'} Q^*(S_{t+1}, a') \mid S_t = s, A_t = a]$$

最优价值 = 选最好的动作后能获得的期望回报
后面 Q-Learning / DQN 就是基于 Q^* 这个式子



直觉:

今天的价值 =
这一步奖励 r_t +
折扣后的明天 $\gamma V^\pi(s')$

如何用样本估计价值函数：MC vs TD

Monte Carlo (MC)

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$

- 用完整回报 $G_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$
- 必须等 episode 结束（对 episodic 任务）
- 无偏：因为是直接对目标进行优化，期望是一样，但方差大：因为 mc 是针对单条轨迹的，同一个策略不同轨迹天然回报差异就比较大
- 不 bootstrap（目标完全来自真实采样回报）

Temporal Difference (TD)

$$V(S_t) \leftarrow V(S_t) + \alpha(r_t + \gamma V(S_{t+1}) - V(S_t))$$

- 用单步估计 $r_t + \gamma V(S_{t+1})$
- 每步都能更新（可在线学习）
- 有偏（目标中用了 $V(S_{t+1})$ 的估计），但方差小：V 是一个相对比较平滑的函数
- Bootstrap：用自己的估计去更新估计

Q-Learning vs SARSA: Off-policy 与 On-policy

核心思想：不再评估某条给定策略 Q^π ，而是以 Bellman Optimality 为目标，**直接逼近最优 Q^***

Q-Learning (Off-policy)

$$Q(s_t, a_t) \leftarrow Q + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q)$$

- 目标用 $\max_{a'}$ ：直接逼近 Q^*
- 行为策略和目标策略**可以不同**
- 更激进，样本效率高

SARSA (On-policy)

$$Q(s_t, a_t) \leftarrow Q + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q)$$

- 目标用实际采样的 a_{t+1} ：学习 Q^π
- 行为策略和目标策略**必须相同**
- 更保守，考虑探索代价

ϵ -greedy 探索：以 $1-\epsilon$ 概率选 $\arg \max_a Q$ ，以 ϵ 概率随机（其他：UCB、Softmax、Boltzmann）

后续：DQN = Q-Learning + 神经网络 + Experience Replay + Target Network

DQN: 用神经网络逼近 Q^*

问题：表格 Q-Learning 无法处理大状态空间（如图像输入、连续状态），也难以泛化到没见过的状态

解决：用神经网络 $Q(s, a; \theta)$ 逼近 Q^* ，最小化 TD 误差

$$\mathcal{L}(\theta) = \mathbb{E} \left[\underbrace{\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-) - Q(s_t, a_t; \theta) \right)^2}_{\text{TD target}} \right]$$

Experience Replay

- 将 (s_t, a_t, r_t, s_{t+1}) 存入 buffer
- 随机采样 mini-batch 训练
- 打破样本相关性，提高数据效率

Target Network

- 用 θ^- （旧参数）计算 target
- 定期更新 $\theta^- \leftarrow \theta$
- 稳定训练，避免目标“互相追着跑”

DQN 变体：Double DQN（解耦选动作与估值）、Dueling DQN（分离 V 和 Q ）、Rainbow...

基于策略的方法

Value-Based 的局限与 Policy-Based 的动机

Value-Based 的局限

- **连续动作困难**: $\max_a Q(s, a)$ 需要枚举所有动作, 连续动作空间无法直接处理
- **函数逼近不稳定**: Deadly Triad——函数逼近 + Bootstrapping + Off-policy 同时使用时容易发散
- **目标不直接**: 通过最小化 TD 误差来逼近 Bellman 方程的解, 而非期望回报 $J(\pi)$
- **只能学确定性策略**: $\arg \max$ 输出确定动作, 但在随机/部分可观测环境中, 随机策略更优

Policy-Based: 直接优化策略参数

用参数化策略 $\pi_\theta(a|s)$ (通常是神经网络), 直接最大化期望回报:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[G_0]$$

策略输出形式:

- 离散动作: $\text{softmax} \rightarrow \text{Categorical}$ 分布
- 连续动作: 输出 $\mu, \sigma \rightarrow \text{Gaussian}$ 分布

核心问题: 如何计算 $\nabla_\theta J(\theta)$?

Policy Gradient 定理推导

目标：最大化期望回报 $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[G_0]$ ，其中 $G_0 = \sum_{t=0}^T \gamma^t r_t$ (Return)

推导：记 $R(\tau) = G_0$ ，轨迹概率 $p(\tau|\theta) = p(s_0) \prod_t \pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t)$

$$\begin{aligned}\nabla_\theta J(\theta) &= \nabla_\theta \int p(\tau|\theta) R(\tau) d\tau = \int p(\tau|\theta) \nabla_\theta \log p(\tau|\theta) R(\tau) d\tau \\ &= \mathbb{E}_{\tau \sim \pi_\theta} [\nabla_\theta \log p(\tau|\theta) \cdot R(\tau)] \quad (\text{Log-derivative trick})\end{aligned}$$

关键： $\nabla_\theta \log p(\tau|\theta) = \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)$ (环境动力学 P 与 θ 无关!)

Policy Gradient Theorem:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot G_t \right]$$

其中 reward-to-go $G_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$ (从 t 时刻开始的折扣回报)

REINFORCE：蒙特卡洛策略梯度

REINFORCE 算法：用采样轨迹估计策略梯度

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) \cdot G_t^{(i)}$$

为什么是无偏的？

- G_t 是真实回报的采样
- 期望 $\mathbb{E}[G_t | s_t, a_t] = Q^{\pi}(s_t, a_t)$
- 采样均值 \rightarrow 期望（大数定律）
- 不依赖任何函数逼近

为什么方差大？

- G_t 累积了整条轨迹的随机性
- 环境随机 + 策略随机 \rightarrow 方差叠加
- 轨迹越长，方差越大
- 奖励稀疏时， G_t 变化剧烈

直觉： G_t 包含了很多与当前动作 a_t 无关的噪声（未来的随机事件），但都被算进了梯度

下一步：如何降低方差？ \rightarrow Baseline / Advantage / Actor-Critic

降低方差：Baseline

问题：REINFORCE 方差太大，能否在不改变期望的情况下降低方差？

Baseline 技巧：减去任意只依赖于 s 、**不依赖动作 a** 的 baseline $b(s)$

$$\nabla_{\theta} J(\theta) = \mathbb{E} \left[\sum_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \cdot (G_t - b(s_t)) \right]$$

为什么可以减？ $\mathbb{E}_{a \sim \pi} [\nabla_{\theta} \log \pi_{\theta}(a | s) \cdot b(s)] = b(s) \cdot \underbrace{\nabla_{\theta} \sum_a \pi_{\theta}(a | s)}_{=1} = 0$

最优 baseline：在不改变期望的前提下， $b(s) = V^{\pi}(s)$ 可证明使方差最小

回顾： $V^{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s]$ ，即从状态 s 出发的期望回报

直觉： G_t 包含“基线”（状态本身好坏），减去 $V^{\pi}(s)$ 后只剩动作带来的增益

Advantage Function

当 $b(s) = V^\pi(s)$ 时, $G_t - V^\pi(s_t)$ 的期望是什么? 在给定 (s, a) 条件下:

$$\mathbb{E}_\pi[G_t - V^\pi(s) \mid S_t = s, A_t = a] = Q^\pi(s, a) - V^\pi(s)$$

定义 **Advantage Function**: $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$

三个价值函数对比:

- $V^\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$
在 s 按策略 π 的期望回报
- $Q^\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$
在 s 选定动作 a 后的期望回报
- $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$
动作 a 比“平均”好多少

Policy Gradient with Advantage:

$$\mathbb{E}[\nabla_\theta \log \pi_\theta \cdot A^\pi]$$

\hat{A}_t 的不同估计方式:

- MC: $G_t - V(s_t)$ (无偏, 高方差)
- TD: $r_t + \gamma V(s_{t+1}) - V(s_t)$ (有偏, 低方差)
- GAE: 介于两者之间

核心思想：同时学习 Actor（策略 π_θ ）和 Critic（价值函数 \hat{V}_ϕ ）

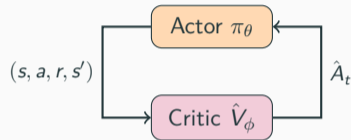
Actor（策略网络）：

- 输出动作分布 $\pi_\theta(a|s)$
- 用 Policy Gradient 更新：
$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta \cdot \hat{A}_t$$

Critic（价值网络）：

- 估计 $\hat{V}_\phi(s) \approx V^\pi(s)$
- 回归到 return 或 TD target：
$$L_V(\phi) = \mathbb{E}[(G_t - V_\phi(s_t))^2] \text{ 或 } \mathbb{E}[\delta_t^2]$$

训练循环：



为什么需要 **Critic**？

- 提供 $\hat{V}(s)$ 计算 \hat{A}_t
- 比纯 MC (G_t) 方差更小
- 可每步更新，不用等 episode 结束

GAE: Generalized Advantage Estimation

问题：MC 估计 $\hat{A}_t = G_t - \hat{V}(s_t)$ (相对无偏，方差大)，TD 估计 $\hat{A}_t = \delta_t$ (方差小，但有偏)

GAE 思路： 用 $\lambda \in [0, 1]$ 插值，定义 TD 残差 $\delta_t = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)$

$$\hat{A}_t^{\text{GAE}} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} = \delta_t + \gamma\lambda\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \dots$$

λ 的作用：

- $\lambda = 0$ ：单步 TD， $\hat{A}_t = \delta_t$ (偏差大，方差小)
- $\lambda = 1$ ：等价 MC， $\hat{A}_t = G_t - \hat{V}(s_t)$ (相对无偏，方差大)
- $\lambda \in (0, 1)$ ：bias-variance tradeoff，常用 $\lambda = 0.95$

直觉：

- δ_t 是单步 advantage 估计
- GAE 把多步 δ 加权求和
- $(\gamma\lambda)^l$ 让远处 δ 权重指数衰减
- 类似 TD(λ) 的思想

PPO、TRPO 等算法都使用 GAE

重要性采样：提高样本效率

问题：Policy Gradient 是 on-policy 的——每次更新 θ 后，旧数据就“过期”了，样本效率低

重要性采样 (Importance Sampling)：用旧策略 π_{old} 的样本，估计新策略 π_θ 下的期望

推导（为什么不改变期望）：

$$\begin{aligned}\mathbb{E}_{a \sim \pi_\theta}[f(a)] &= \sum_a \pi_\theta(a) f(a) = \sum_a \pi_{\text{old}}(a) \cdot \frac{\pi_\theta(a)}{\pi_{\text{old}}(a)} f(a) \\ &= \mathbb{E}_{a \sim \pi_{\text{old}}} \left[\frac{\pi_\theta(a)}{\pi_{\text{old}}(a)} f(a) \right]\end{aligned}$$

应用到 **Policy Gradient**：记重要性权重 $\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_{\text{old}}} \left[\rho_t(\theta) \nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t \right]$$

PPO: Proximal Policy Optimization

问题：重要性采样后， $\rho_t(\theta)$ 偏离 1 太多会导致方差爆炸、策略崩溃

解决思路：限制策略更新幅度，让 ρ_t 不要偏离 1 太远

TRPO: KL 散度约束

$$\max_{\theta} \mathbb{E}[\rho_t \hat{A}_t]$$

$$\text{s.t. } \text{KL}(\pi_{\text{old}} \parallel \pi_{\theta}) \leq \delta$$

理论优美，但需二阶优化，实现复杂

PPO-Clip: 截断重要性权重

$$L^{\text{CLIP}} = \mathbb{E}[\min(\rho_t \hat{A}_t, \text{clip}(\rho_t, 1 \pm \epsilon) \hat{A}_t)]$$

简单高效，一阶优化，常用 $\epsilon = 0.2$

PPO-KL: KL 惩罚项

$$L = \mathbb{E}[\rho_t \hat{A}_t] - \beta \cdot \text{KL}$$

自适应调整 β

Model-Based 与 Multi-Agent

Model-Based RL & Multi-Agent RL

Model-Based RL: 利用/学习环境模型

核心：得到 $\hat{P}(s'|s, a)$, $\hat{R}(s, a)$ (已知规则或学习)，在模型中规划

两种用法：

- **Background Planning:** 模型生成数据训练
- **Decision-time Planning:** 向前搜索 (MCTS)

优势：样本效率高，可在“想象”中学习

劣势：模型不准 \rightarrow model bias

Multi-Agent RL: 多智能体博弈

与单智能体的区别：

- 环境包含其他 agent (非稳态)
- 需要考虑对手/队友策略变化
- 博弈论常用 Nash 均衡 描述稳定解

两种设定：

- 合作：最大化团队收益
- 竞争/零和：一方获益 = 另一方损失

Self-Play: 与自己（历史版本）博弈，不断提升

AlphaZero 可以看作：Model-Based (已知棋规 + MCTS) + Multi-Agent (自我对弈) 19/47

AlphaGo (2016, 击败李世石):

- **Policy Network**: 人类棋谱监督预训练
- **Policy Gradient**: 自我对弈强化
- **Value Network**: 预测胜率
- **MCTS**: 结合 p_θ, v_ϕ 搜索

MCTS 四步: Selection (UCB) \rightarrow Expansion (p_θ) \rightarrow Evaluation (v_ϕ) \rightarrow Backup

核心循环: MCTS 改进策略 \rightarrow 网络学习搜索结果 \rightarrow 更强网络 \rightarrow 自我对弈生成无限数据

AlphaZero (改进):

- **不需要人类棋谱**: 从零自我对弈
- **统一网络**: 同时输出 p_θ, v_ϕ
- **更简洁**: 去掉 rollout

训练循环:

1. 网络 + MCTS 自我对弈
2. 记录 $(s, \pi_{\text{MCTS}}, z)$
3. $p_\theta \rightarrow \pi_{\text{MCTS}}, v_\phi \rightarrow z$
4. 重复, 越来越强

LLM 与强化学习

LLM 中的 RL：如何建模？

核心问题：如何把“LLM 对齐”建模成 RL 问题？

RL 视角下的 LLM：

- **State** s ：prompt + 已生成的 token 序列
- **Action** a ：下一个 token（词表 $|\mathcal{V}|$ ）
- **Policy** $\pi_{\theta}(a|s)$ ：LLM 本身
- **Trajectory** τ ：完整的生成序列
- **Reward** r ：只在序列结束时给出

特点：

- 动作空间巨大（词表 $\sim 100k$ ）
- **稀疏奖励**：只有最后一步有 reward
- Episode = 一次完整生成

RLHF 优化目标：

Reward Model：学习人类偏好

- 人类标注偏好对： $y_w \succ y_l$

目标函数：

$$\max_{\theta} \mathbb{E}_{y \sim \pi_{\theta}} [r_{\phi}(x, y)] - \beta \text{KL}(\pi_{\theta} \| \pi_{\text{ref}})$$

- $r_{\phi}(x, y)$ ：Reward Model 打分
- $\text{KL}(\pi_{\theta} \| \pi_{\text{ref}})$ ：约束项
- π_{ref} ：SFT 后的初始模型

KL 项可以稳定训练，防止模型“hack”

reward model

Stage 1: Supervised Fine-Tuning (SFT)

- 用高质量对话数据微调预训练模型
- 得到参考模型 π_{ref}

Stage 2: Reward Model 训练

- 收集人类偏好数据 (x, y_w, y_l)
- Bradley-Terry pairwise loss:

$$L(\phi) = -\mathbb{E}[\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))]$$

Stage 3: 用 PPO 微调策略

- 用 r_ϕ 提供奖励信号
- 优化“高奖励 + 小 KL”目标
- 使用 Advantage / GAE + PPO-Clip

PPO 更新步骤:

1. 用 $\pi_{\theta_{\text{old}}}$ 生成回复 y
2. 计算序列总奖励:

$$R = r_\phi(x, y) - \beta \log \frac{\pi_{\theta_{\text{old}}}(y|x)}{\pi_{\text{ref}}(y|x)}$$

3. 用 GAE 计算 \hat{A}_t
4. 计算比率 $\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$

PPO-Clip 目标:

$$L^{\text{CLIP}} = \mathbb{E}[\min(\rho_t \hat{A}_t, \text{clip}(\rho_t, 1 \pm \epsilon) \hat{A}_t)]$$

总 loss = 策略 + 值函数 + 熵正则

DPO: 绕过 Reward Model 和 PPO

DPO Loss:

$$L_{\text{DPO}}(\theta) = -\mathbb{E}_{(x, y_w, y_l)} \left[\log \sigma \left(\beta \left[\log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right] \right) \right]$$

RLHF + PPO 的问题:

- 需要维护 π_{θ} 、 V_{ψ} 、 r_{ϕ} 三个模型
- 在线采样成本高（大模型生成贵）
- PPO 超参敏感，实现不稳定

DPO 的思路:

- 直接在偏好数据 (x, y_w, y_l) 上优化
- 不需要 Reward Model 和 PPO
- 离线训练，形式像监督学习

推导思路（详见附录 A）:

1. KL 正则 RL 目标引入配分函数 $Z(x)$
2. 最优策略: $\pi^*(y|x) \propto \pi_{\text{ref}}(y|x) e^{r(x,y)/\beta}$
3. 反解: $r^* = \beta \log \frac{\pi^*}{\pi_{\text{ref}}} + \beta \log Z$
4. 代入 Bradley-Terry, $Z(x)$ 消掉
5. 最大似然优化 \rightarrow DPO Loss

直观理解:

- 提高 y_w 相对 π_{ref} 的 log-prob
- 压低 y_l 相对 π_{ref} 的 log-prob

GRPO: Group Relative Policy Optimization

动机: PPO 太重, DPO 又不够用

PPO: 需要 Critic 网络, 显存开销大

DPO:

- 完全 offline, 没有探索机制
- Pairwise 信号粗糙, 不知道好多少
- 难任务 (数学/代码) 提升有限

GRPO 做法:

1. 对 prompt x , 采样一组 $\{y_1, \dots, y_G\}$
2. 计算组内奖励 $\{R_1, \dots, R_G\}$
3. 组内标准化: $\hat{A}_i = \frac{R_i - \bar{R}}{\text{Std}(R)}$
4. PPO-Clip 更新

核心: 用组内相对奖励代替 Critic

目标函数:

$$L_{\text{GRPO}} = \mathbb{E} \left[\frac{1}{G} \sum_{i=1}^G \sum_t \min \left(\rho_{i,t} \hat{A}_i, \text{clip}(\rho_{i,t}) \hat{A}_i \right) \right]$$

$$\text{其中 } \rho_{i,t} = \frac{\pi_{\theta}(a_{i,t} | s_{i,t})}{\pi_{\theta_{\text{old}}}(a_{i,t} | s_{i,t})}$$

Tricks:

- *Clip-Higher*: 上界更宽松
- *Dynamic Sampling*: 过滤全对/全错 prompt
- Token 级 PG loss 与 overlong reward shaping

适用场景

- GRPO: 中短序列、需快速上线, 显存敏感 (无 Critic)
- GSPO: 长 CoT / MoE, 序列级 IS 更稳定
- KL in reward (k_1): 保守、安全优先; KL in loss (k_3/k_2): 高效探索

常用超参参考

- Clip ϵ : 0.1-0.2 (长序列/偏离大时适当放宽)
- Group size G : 4-8; 过小噪声大, 过大耗显存
- KL 系数 β/λ : 从 0.05-0.2 网格,

稳定性 Tricks

- 组内标准化 advantage, 过滤全对/全错样本
- 长序列: 长度归一化 IS (GSPO) 或直接 clip IS 权重 (CISPO)
- MoE: 保持 routing (keep routing/replay); 推理与训练引擎保持一致
- 训练中监控: KL、拒答率、长度分布、win-rate, 偏离大就增大 KL 或减小步长

经验值因任务/模型而异, 建议先小步长网格搜索再放大 batch

PRM: 过程监督与 Verifier-Guided RL

PRM 基础 (Let's Verify Step by Step, OpenAI 2023)

- 数据: CoT \rightarrow step \rightarrow 每步标好/坏 (PRM800K)
- 训练: $(x, y_{\leq t}, y_{t+1}) \rightarrow$ 正确性分数
- 推理: 多条 CoT 按累积分数 rerank
- RL: $r_t = s_t - s_{t-1}$, 终止加成功信号

PRM vs ORM

- ORM: 只看最终结果, 信号稀疏
- PRM: 每步打分, dense reward
- 长链推理收敛更快、更稳定

核心: 把稀疏终局 reward 变成“步步有分”, 缓解长 CoT 的高方差问题

落地要点

- Verifier 要小且快, 支持批量
- 早停: 得分持续下降就截断
- 数据闭环: 高分进 replay, 低分作负例
- 监控 reward hacking: 提高 KL

代表: rStar-Math (Microsoft 2025)

- Process Preference Model + MCTS
- 小模型在 AIME 达 53% (8/15)

KL 散度的三种估计: k_1 / k_2 / k_3

问题: $D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) = \mathbb{E}[-\log r]$ 无法精确算, 需 MC 估计 $r = \pi_{\text{ref}} / \pi_\theta$, 详见附录 B

三种 Estimator:

k1: $k_1 = -\log r$ 无偏, 高方差

- 放进 reward: $r_t^{\text{RL}} = r_t^{\text{RM}} - \beta k_1$

k2: $k_2 = \frac{1}{2}(\log r)^2$ 有偏, 梯度正确

- 梯度等价于真 KL, 更平滑, 适合做 loss

k3: $k_3 = (r - 1) - \log r$ 无偏, 低方差

- $\mathbb{E}[r-1]=0$ 作 control variate
- 做 loss 时梯度是一阶近似; r 大时需 clip

PPO 中的两种用法:

KL in Reward (经典 RLHF):

- token reward 减 βk_1 , 再 PPO-Clip

KL as Loss (GRPO):

- 总 loss: $L = -L_{\text{RL}} + \lambda \mathbb{E}[k_3]$

实践建议:

- 理论严谨: k_1 -in-reward / k_2 -as-loss
- GRPO/R1: k_3 -as-loss + clip
- 新趋势 (RF++): 推荐 k_2

Long CoT RL: GSPO、CISPO 与 Kimi k1.5

GSPO (Qwen: Group Sequence PO)

- 问题: GRPO token 级 IS + clip, 长序列 / MoE 易崩溃
- 序列级 **IS**: $s_i(\theta) = \left(\frac{\pi_\theta(y_i|x)}{\pi_{\text{old}}(y_i|x)} \right)^{1/|y_i|}$
- 长度归一化后再 clip, 所有 token 共用同一个 s_i

CISPO (MiniMax: Clipped IS-weight PO)

- 继承 GRPO 组内标准化, 回到 token 级 REINFORCE
- Clip **IS** 权重 (非 loss):
 $\hat{r}_{i,t} = \text{clip}(r_{i,t}, 1 \pm \epsilon)$
- 去掉 KL 惩罚 + 动态采样 + 长度惩罚

Kimi k1.5: 长 CoT RL + Long2Short

长 CoT RL 配方:

- 128k 上下文直接 RL, Mirror Descent 式更新
- Trick: 部分 rollout、异步 train/infer、重复检测 + 早停、长度惩罚

Long2Short RL (长到短蒸馏):

- 长 CoT 模型作 teacher, 训练短 CoT student
- 正确性 + token 数奖励, 鼓励“又对又短”

1. Unbiased KL Estimate

- 问题: K3 在 off-policy (从 π_{old} 采样) 下有偏
- 用 IS 比率修正:
$$\hat{D}_{KL} = \frac{\pi_{\theta}}{\pi_{old}} \cdot \left(\frac{\pi_{ref}}{\pi_{\theta}} - \log \frac{\pi_{ref}}{\pi_{\theta}} - 1 \right)$$

2. Off-Policy Sequence Masking

- 负优势 $\hat{A}_i < 0$ 且 π_{θ} 与 π_{old} 偏离过大 \rightarrow mask
- 避免在“坏序列”上浪费梯度

3. Keep Routing (MoE 专用)

- 保持采样时的 expert routing 路径
- 防止 routing 变化导致训练不稳定

4. Keep Sampling Mask

- 保持 top-p/top-k 的 truncation mask
- 训练时只在采样时可选的 action 上计算

核心思想: 让 off-policy 训练尽量“接近” on-policy 的行为

On-Policy Distillation: 高效的后训练范式

动机: Off-policy 蒸馏的分布偏移

- SFT on 教师数据 → **compounding errors**
- 学生推理时状态 \neq 训练时见过的
- 长序列推理中问题更严重

三种范式对比

- Off-policy SFT: 密集监督, 但分布偏移
- On-policy RL: 分布匹配, 但奖励稀疏
- **On-Policy Distill**: 两者优势结合

核心: 学生采样 + 教师逐 token 打分

Reverse KL 损失 (mode-seeking)

$$L = D_{\text{KL}}(\pi_{\theta} \parallel \pi_T) = \mathbb{E}_{\pi_{\theta}} \left[\log \frac{\pi_{\theta}}{\pi_T} \right]$$

KDRL: KD + RL

$$\mathcal{J} = \underbrace{\mathcal{J}_{\text{GRPO}}}_{\text{奖励}} - \beta \cdot \underbrace{D_{\text{KL}}^{k_2}(\pi_{\theta} \parallel \pi_T)}_{\text{教师监督}}$$

效率 (Qwen3)

- 相比 RL: 速度 **7-10x**, 总效率 **50-100x**
- GPU 小时仅需 RL 的 $\sim 1/10$

核心问题：Reward 是 sequence-level，
但 REINFORCE/GRPO 是 token-level，合理吗？

关键洞察：Token-level 目标是 \mathcal{J}^{seq} 的一阶近似

- 设 $\frac{\pi_{\theta}(y_t)}{\mu_{\theta_{old}}(y_t)} = 1 + \delta_t$
- $\prod_t (1 + \delta_t) \approx 1 + \sum_t \delta_t$
- $\Rightarrow \nabla \mathcal{J}^{seq} \approx \nabla \mathcal{J}^{token}$

成立条件： $\pi_{\theta} \approx \mu_{\theta_{old}}$

$$\frac{\pi_{\theta}}{\mu_{\theta_{old}}} = \underbrace{\frac{\pi_{\theta_{old}}}{\mu_{\theta_{old}}}}_{\text{train-infer}} \times \underbrace{\frac{\pi_{\theta}}{\pi_{\theta_{old}}}}_{\text{staleness}}$$

两个关键条件：

▪ Training-Inference Discrepancy

训练/推理引擎数值不一致（kernel、精度、MoE routing）

▪ Policy Staleness

Off-policy: rollout policy \neq 当前 policy

稳定 RL 的统一解释：

- **IS correction**：一阶近似的固有组成
- **Clipping**：限制 policy staleness
- **Routing Replay**：MoE 路由一致性

总结：问题的根源与解决方向

核心矛盾：想优化序列级 reward，但用的是 token-level surrogate + heuristic

四个根源问题：

1. 目标错位

Token loss $\neq \mathcal{J}^{seq}$ ，一阶近似条件苛刻

2. 高方差

长 CoT + 稀疏 reward \rightarrow 梯度噪声大

3. 系统不一致

Train/infer 差异、off-policy、MoE 路由抖动

4. Reward 不完备

只看答案 \rightarrow reward hacking

解决方向：

► 修正目标函数

对齐 \mathcal{J}^{seq} (GSPO)、无偏 KL (V3.2)

► 降低方差

Group advantage (GRPO)、动态采样 (DAPO)

► 系统层面修正

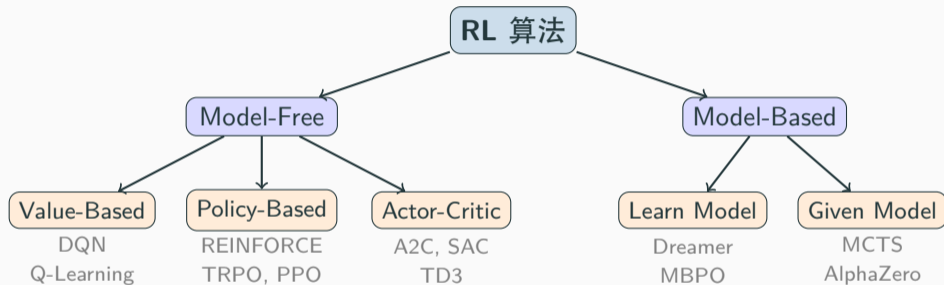
IS、Clipping、Routing Replay、Seq Mask

► 改进 Reward

Reward shaping、数据 curriculum

GRPO/GSPO/DAPO/V3.2 切入点不同，但都在解决这些根源问题

总结



LLM 时代: RLHF (PPO) → DPO (离线) → GRPO (无 Critic) → GSPO/CISPO (Long CoT)

Thanks!

附录 A: DPO 详细推导 (1/5) ——RLHF 目标与变换

Step 1: RLHF 目标函数

$$\max_{\pi} \mathbb{E}_{x \sim D, y \sim \pi} [r(x, y)] - \beta \text{D}_{\text{KL}}[\pi(y|x) \parallel \pi_{\text{ref}}(y|x)]$$

Step 2: 展开 KL 散度, 转为 min 问题

$$\begin{aligned} J(\theta) &= \max_{\pi} \mathbb{E}_{x, y \sim \pi} \left[r(x, y) - \beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} \right] \\ &= \min_{\pi} \mathbb{E}_{x, y \sim \pi} \left[\beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - r(x, y) \right] \\ &= \min_{\pi} \mathbb{E}_{x, y \sim \pi} \left[\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta} r(x, y) \right] \end{aligned}$$

目标: 找到一个策略 π , 既能获得高 reward, 又不偏离 π_{ref} 太远

附录 A: DPO 详细推导 (2/5) —— 引入配分函数

Step 3: 引入配分函数 $Z(x)$

目标函数中加减 $\log Z(x)$ (不影响优化):

$$\begin{aligned} J(\theta) &= \min_{\pi} \mathbb{E}_{x, y \sim \pi} \left[\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta} r(x, y) + \log Z(x) - \log Z(x) \right] \\ &= \min_{\pi} \mathbb{E}_{x, y \sim \pi} \left[\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \log \exp \left(\frac{1}{\beta} r(x, y) \right) - \log Z(x) + \log Z(x) \right] \\ &= \min_{\pi} \mathbb{E}_{x, y \sim \pi} \left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp \left(\frac{1}{\beta} r(x, y) \right)} - \log Z(x) \right] \end{aligned}$$

定义配分函数 (为了让分母是合法概率分布):

$$Z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp \left(\frac{1}{\beta} r(x, y) \right)$$

附录 A: DPO 详细推导 (3/5) ——最优策略

Step 4: 定义最优策略 π^*

通过在目标中加减 $\log Z(x)$, 目标改写为:

$$J(\theta) = \min_{\pi} \mathbb{E}_x [\mathcal{D}_{\text{KL}}(\pi(\cdot|x) \parallel \pi^*(\cdot|x)) - \log Z(x)]$$

其中 $\pi^*(y|x) \propto \pi_{\text{ref}}(y|x) \exp(r(x, y)/\beta)$ 是使 KL 最小 (为 0) 的分布, 即最优策略的闭式解:

$$\pi^*(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

由于 $Z(x)$ 不依赖 π , 最优解为 $\pi = \pi^*$

附录 A: DPO 详细推导 (4/5) ——反解 reward

Step 5: 反解 reward function

从 π^* 的定义式反解出 $r(x, y)$:

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

Step 6: Bradley-Terry 偏好模型

$$p(y_w \succ y_l | x) = \frac{\exp(r(x, y_w))}{\exp(r(x, y_w)) + \exp(r(x, y_l))} = \sigma(r(x, y_w) - r(x, y_l))$$

附录 A: DPO 详细推导 (5/5) —— DPO Loss

Step 7: 代入 reward, $Z(x)$ 消掉

$$\begin{aligned} r(x, y_w) - r(x, y_l) &= \beta \log \frac{\pi^*(y_w|x)}{\pi_{\text{ref}}(y_w|x)} + \cancel{\beta \log Z(x)} - \beta \log \frac{\pi^*(y_l|x)}{\pi_{\text{ref}}(y_l|x)} - \cancel{\beta \log Z(x)} \\ &= \beta \left[\log \frac{\pi^*(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi^*(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right] \end{aligned}$$

Step 8: 最大似然优化 \rightarrow 希望 $p(y_w \succ y_l|x)$ 越大越好

$$L_{\text{DPO}}(\theta) = -\mathbb{E}_{(x, y_w, y_l)} \left[\log \sigma \left(\beta \left[\log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right] \right) \right]$$

附录 B: KL Estimator 推导 (1/3) ——问题设定与 k1

目标: 估计 $D_{\text{KL}}(q\|p) = \mathbb{E}_{x\sim q} \left[\log \frac{q(x)}{p(x)} \right]$, 其中 $q = \pi_\theta$, $p = \pi_{\text{ref}}$

记号 (Schulman): $r(x) = \frac{p(x)}{q(x)} = \frac{\pi_{\text{ref}}}{\pi_\theta}$, 则 $\log \frac{q}{p} = -\log r$

k1: 直接 Monte Carlo 估计

$$k_1(x) = -\log r(x) = \log \pi_\theta - \log \pi_{\text{ref}}$$

无偏性: $\mathbb{E}_{x\sim q}[k_1] = \mathbb{E}_{x\sim q}[-\log r] = \mathbb{E}_{x\sim q} \left[\log \frac{q}{p} \right] = D_{\text{KL}}(q\|p) \checkmark$

性质:

- 无偏, 但方差大 (π_θ 与 π_{ref} 差异大时尤其明显)
- 用法: 放进 reward, $r_t^{\text{RL}} = r_t^{\text{RM}} - \beta \cdot k_1$

附录 B: KL Estimator 推导 (2/3) —— k_2 的来源与梯度等价性

k_2 : 平形式

$$k_2(x) = \frac{1}{2}(\log r)^2 = \frac{1}{2} \left(\log \frac{\pi_{\text{ref}}}{\pi_{\theta}} \right)^2$$

数值有偏: $\mathbb{E}[k_2] = \mathbb{E}[\frac{1}{2}(\log r)^2] \neq D_{\text{KL}}$ (除非 $\pi_{\theta} = \pi_{\text{ref}}$)

但梯度等价! 对 $L_{\text{KL}} = \mathbb{E}_{y \sim \pi_{\theta}}[k_2]$ 求梯度:

$$\begin{aligned} \nabla_{\theta} L_{\text{KL}} &= \nabla_{\theta} \mathbb{E}_{y \sim \pi_{\theta}} \left[\frac{1}{2}(\log r)^2 \right] \\ &= \mathbb{E}_{y \sim \pi_{\theta}} \left[\frac{1}{2}(\log r)^2 \nabla_{\theta} \log \pi_{\theta} + \log r \cdot \nabla_{\theta} \log \pi_{\theta} \right] \\ &= \mathbb{E}_{y \sim \pi_{\theta}} [\log r \cdot \nabla_{\theta} \log \pi_{\theta}] \quad (\text{第一项期望为 } 0) \end{aligned}$$

这与 $\nabla_{\theta} D_{\text{KL}}$ 相同! \Rightarrow 数值有偏, 梯度正确, 且更平滑

附录 B: KL Estimator 推导 (3/3) —— k3 的 Control Variate 构造

k3: 加入 control variate 降方差

$$\text{观察: } \mathbb{E}_{x \sim q}[r(x)] = \mathbb{E}_{x \sim q} \left[\frac{p(x)}{q(x)} \right] = \int q(x) \frac{p(x)}{q(x)} dx = \int p(x) dx = 1$$

因此 $\mathbb{E}_{x \sim q}[r - 1] = 0$, 可作为 control variate 加入 k1 而不改变期望:

$$k_3(x) = \underbrace{-\log r}_{k_1} + \lambda(r - 1), \quad \text{取 } \lambda = 1$$

$$k_3(x) = (r - 1) - \log r = \frac{\pi_{\text{ref}}}{\pi_{\theta}} - 1 - \log \frac{\pi_{\text{ref}}}{\pi_{\theta}}$$

无偏性证明:

$$\mathbb{E}_{x \sim q}[k_3] = \underbrace{\mathbb{E}_q[r - 1]}_{=0} - \underbrace{\mathbb{E}_q[\log r]}_{=-D_{\text{KL}}(q||p)} = D_{\text{KL}}(q||p) \quad \checkmark$$

性质: 无偏 + 方差比 k1 小, 但做 loss 时梯度只是近似; r 大时会爆 \rightarrow **k3_clip**

Q&A 1: 为什么 RLHF 目标要加 KL 正则?

Q: 为什么 RLHF 写成 $\max_{\pi} \mathbb{E}_{y \sim \pi} [r(x, y)] - \beta \text{KL}(\pi \| \pi_{\text{ref}})$?

A: 本质是约束优化的拉格朗日形式:

$$\max_{\pi} \mathbb{E}[r(x, y)] \quad \text{s.t.} \quad \mathbb{E}_x[\text{KL}(\pi(\cdot|x) \| \pi_{\text{ref}}(\cdot|x))] \leq \epsilon$$

直接解约束问题麻烦, 用拉格朗日乘子 β 把约束搬到目标里。

直观解释:

- 第一项: 希望输出人类喜欢/高 reward 的内容
- 第二项: 不允许离参考模型太远, 防止 reward hacking / 语言崩坏

和最大熵 RL 的关系:

- 经典最大熵 RL: 惩罚 $\text{KL}(\pi \| \text{uniform})$ 或加 entropy
- RLHF: 惩罚 $\text{KL}(\pi \| \pi_{\text{ref}})$, 即相对参考模型的偏移

Q&A 2: 为什么长上下文下 token-level PPO/GRPO 容易崩?

Q: 为什么长序列 RL 训练 (如 Long CoT) 用 token-level IS 容易崩溃?

A: 关键是重要性比率的长度效应:

数学上: 序列级 IS ratio = $\prod_t \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}$, 等价于 $\exp(\sum_t \log \rho_t)$

- logprob 差在长序列上线性累积, exp 后变成指数级差异
- 某几个 token 的 logprob 差就能让整个 ratio 特别大/特别小

后果:

- Variance 巨大, 梯度由极少数样本、极少数 token 主导
- 对 MoE / 大模型尤其灾难: 少量极端更新打坏路由
- Clip 也救不了: 要么 clip 太多信息丢失, 要么 clip 不够仍然爆炸

Q&A 2 (续): 长序列 RL 的解决方案

GSPO (Qwen)

- IS ratio 搬到序列级:

$$\rho = \exp\left(\frac{\log \pi_{\theta}(y|x) - \log \pi_{\text{old}}(y|x)}{|y|}\right)$$

- 先长度归一化, 再 clip
- 同一序列所有 token 共用一个 ρ
- 直觉: reward 是序列级, IS 粒度对齐

CISPO (MiniMax)

- 仍用 token-level REINFORCE
- 对 IS 权重本身 clip, 而非 loss
- 去掉/减弱 KL + 长度惩罚
- 动态过滤过长/失败样本

Kimi k1.5 (真 · 128k RL)

工程重点:

- **Partial rollout**: 发现胡说就早停
- 重复检测: anti-degenerate 正则
- 长度惩罚: 控制输出长度

Long2Short:

- 长 CoT teacher 教 short student
- Reward = 正确性 + token 数惩罚
- 目标: 答案正确且尽量简洁

Q&A 3: 如何同时优化 Helpfulness / Harmlessness / Honesty?

Q: 多目标对齐怎么做?

方法一: 多头 RM + 加权

- RM 输出 (r_{help} , r_{safe} , r_{honest})
- 总 reward = $\sum_i w_i \cdot r_i$
- 权重通过 A/B test / 人工调优

方法二: 约束优化 / Lagrangian

- 目标: max helpfulness
- 约束: $\mathbb{E}[r_{\text{harm}}] \leq \epsilon$
- Reward shaping:
$$R = r_{\text{help}} - \lambda \cdot \max(0, r_{\text{harm}} - \epsilon)$$

方法三: Hard Constraint + RL

- 安全条款用 rule-based / classifier 直接拒绝
- RL 只优化“回复”的质量
- 训练与推理双重保险

实践中常见组合:

- 多头 RM 提供细粒度信号
- Lagrangian 处理硬约束
- 推理时再加安全过滤层

Q&A 4: 如何避免 Reward Hacking?

Q: 模型学会 reward hacking 怎么办?

问题: RM 是学出来的, 必然有 bias / 漏洞; 模型会找到高 reward 但人类不喜欢的 pattern

常见对策:

- **多源 Feedback**: 不同 annotator、不同模型投票, ensemble RM
- **Adversarial Eval / Red Teaming**: 主动找 RM 漏洞, 加入训练
- **保守 RM**: 训练成 lower bound (宁可漏标好样本, 也不放过坏样本)
- **强 KL 约束**: 限制策略偏离 π_{ref} 的程度, 减少 exploit 空间
- **双重保险**: 训练时 KL 约束 + 推理时安全过滤

本质: RM 是 proxy, 真正目标是“人类满意”; 多层防护 + 持续迭代是关键