DeepSeek-R1 训练流程

目录

DeepSeek 发展历程

DeepseekV1: DenseNet + SFT + DPO(RLHF的显式解，推理任务不是最优）

DeepSeekV2: MoE + MLA + SFT + RL

DeepSeekV3: MoE + auxiliary loss free + MLA + MTP + SFT + RL

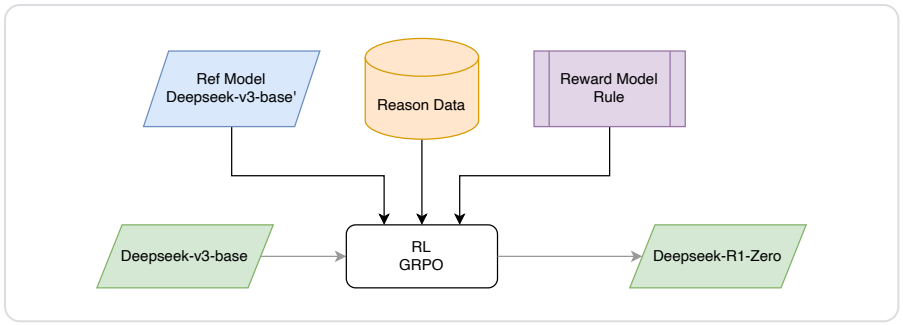
DeepSeek-R1-zero + DeepSeek-R1

训练流程

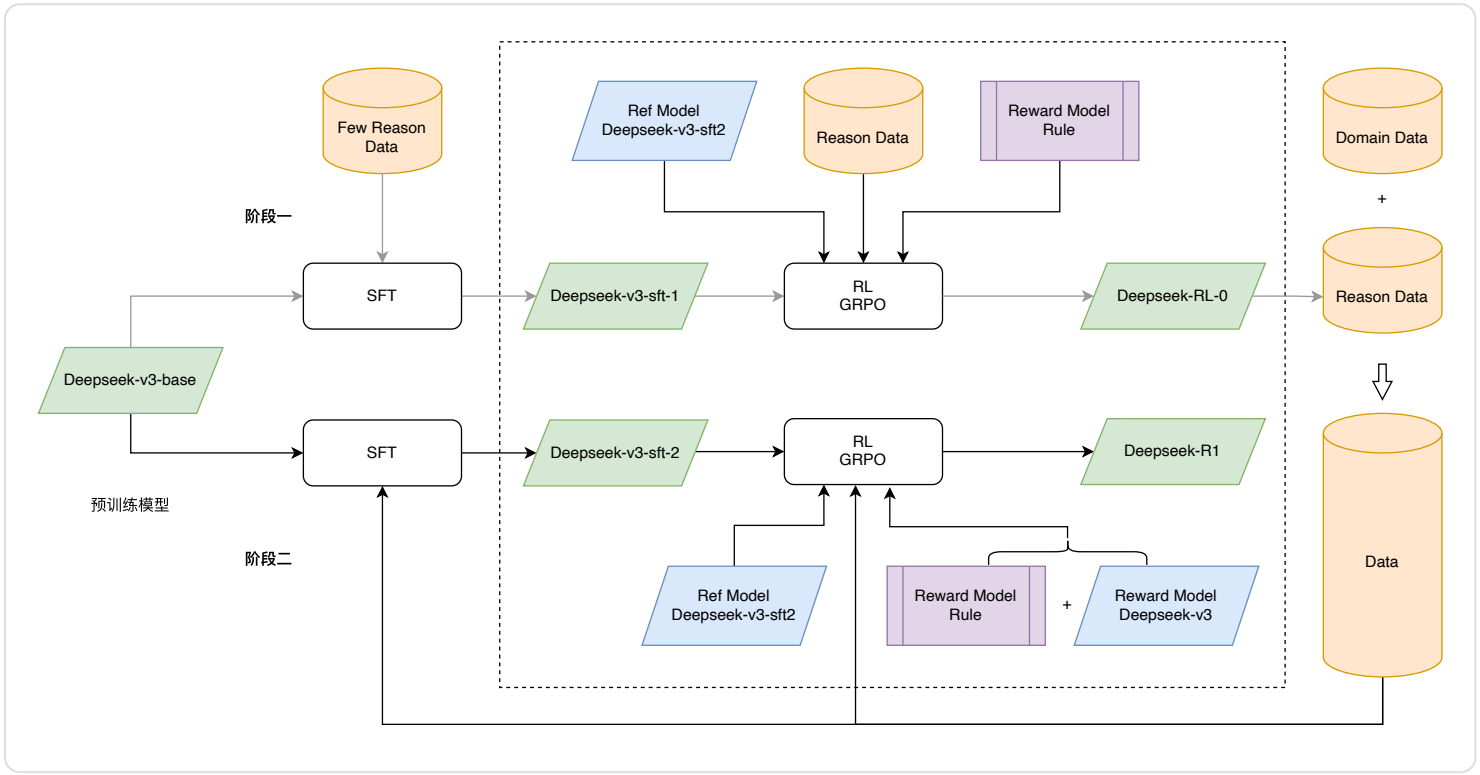
Deepseek-R1-zero：完全不使用监督数据，比如推理问题对应的答案无需人工标注，直接通过 reward model/rule 反馈。

优势：仅使用预训练模型 + 推理数据 + RL 即可在【推理任务上】达到闭源模型的指标水平

不足：语言的一致性不足，文本可读性差，各类语言交叉



Deepseek-R1：通过推理数据和常规数据，经过SFT+RL训练，文本输出合理同时推理能力强的模型



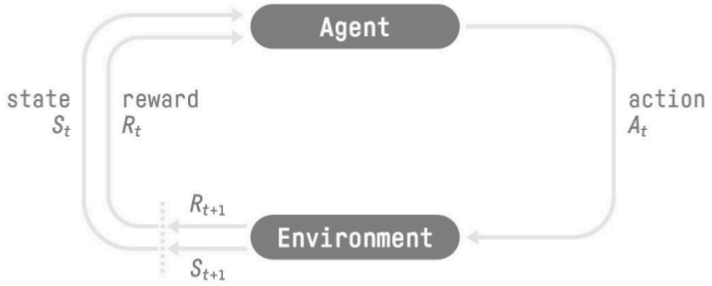
各阶段数据生成：

	构成和生成方式	量级
Few Reason Data	1. 使用长 COT 作为 few-shot prompt，要求生产详细解答 2. 将 DeepSeek-R1-Zero 输出格式化 3. **人工检验修正**	k级别
Reason Data		Large scale well-defined problems with clear solutions (> 600k ?)
New Reason Data	1. Reason Data I: Rejected Sampling: 用一个简单分布抽样复杂分布，生成 rule-based reward 的数据 2. Reason Data II: DeepSeek V3 判断好坏的数据，使用 deepseek v3 生成 reward	600k
Domain Data	Other Domain: DeepSeek v3 生产 COT	200k

策略函数（policy）- Transformer 模型

强化学习是智能体通过采取动作与环境进行交互，获取奖惩及后续观测，进而学习**动作策略** π 的框架。

具体来说， $P(a_t \mid s_t)$ 表示在状态 s_t 下采取动作 a_t 的条件概率，它是由策略函数 π 决定。



详细说明

s_t

- s_t 表示在时间步 t 时的状态（state）。
- 状态是环境对智能体的当前描述，例如在游戏中可能是角色的位置、速度等信息。

a_t

- a_t 表示在时间步 t 时智能体采取的动作（action）。
- 动作是智能体在给定状态下可以执行的操作，例如在游戏中可能是“向左移动”或“跳跃”。

$\pi(a_t \mid s_t)$

- $\pi(a_t \mid s_t)$ 是策略函数（policy），表示在状态 s_t 下选择动作 a_t 的概率。
- 如果是确定性策略，$\pi(a_t \mid s_t)$ 会直接输出一个确定的动作；如果是随机策略，它会输出一个动作的概率分布。

$R(\tau)$ 奖惩以及折扣：

对于一个动作有即时反馈 r，则对于一个动作轨迹的累计奖惩可以按如下表示：

$$R(\tau) = r_{t+1} + r_{t+2} + ...$$

但是对于轨迹来说，我们更关心眼前的奖惩，所以对于未来的反馈会增加一个折扣，最终的折扣累计奖惩形式如下：

$$R(\tau) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ...$$

奖惩的设置非常重要，因为这是智能体学习的唯一反馈来源，这个也是强化学习要重点设计的。

举例说明

假设我们有一个简单的游戏环境：

- 状态 s_t ：角色的位置。
- 动作 a_t ：可以执行的动作是“向左”或“向右”。
- 策略 $\pi(a_t \mid s_t)$ ：在某个位置 s_t 下，策略可能以 70% 的概率选择“向左”，以 30% 的概率选择“向右”。

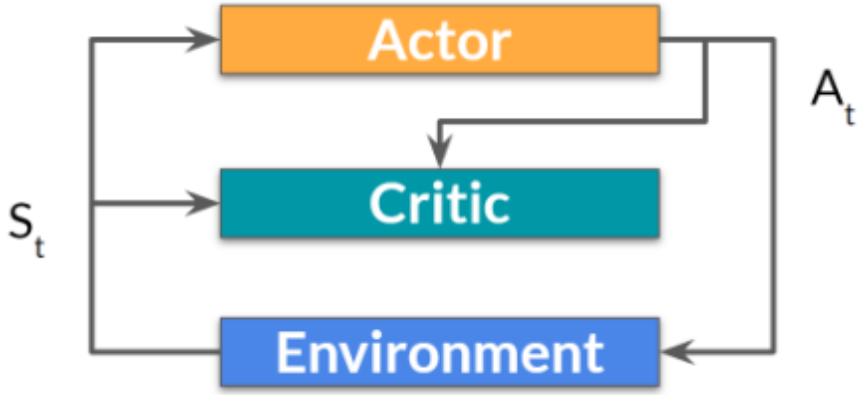
在 PPO 中，我们会比较新旧策略在相同状态 s_t 下选择相同动作 a_t 的概率，从而计算概率比 $r_t(\theta)$ ，并用于优化目标函数。

总结

$P(a_t \mid s_t)$ 表示在状态 s_t 下选择动作 a_t 的条件概率，由策略函数 π 决定。

近端策略优化（PPO）

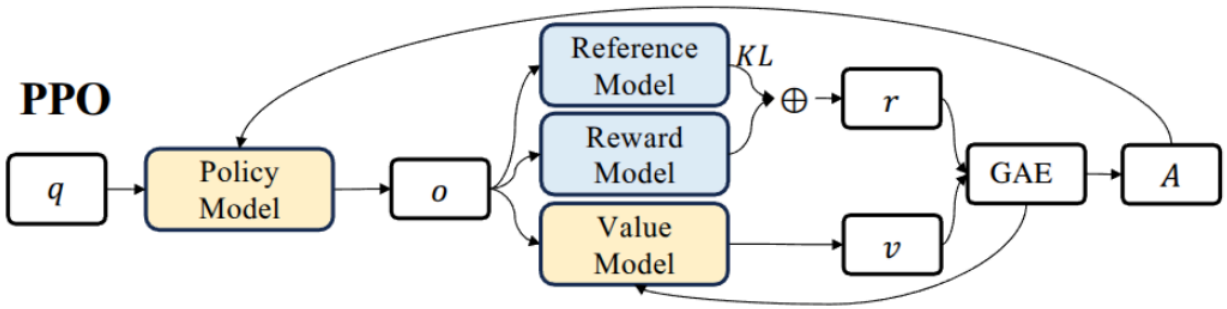
PPO（Proximal Policy Optimization） 是一种用于强化学习的策略优化算法。它直接优化策略函数，使用 Actor-Critic 方式，通过限制策略更新的幅度，确保训练过程的稳定性。



核心思想

PPO 的核心在于限制策略更新的幅度，避免因更新过大导致性能下降。它通过引入“裁剪”机制，控制新旧策略之间的差异。

公式



PPO 的替代目标函数 $\mathcal{J}_{PPO}(\theta)$ 用于优化策略 π_θ ，公式如下：

$$\mathcal{J}_{PPO}(\theta) = E_{[q\sim P(Q),o\sim\pi_{\theta_{old}}(O|q)]}\frac{1}{|o|}\sum_{t=1}^{|o|}\left[min\left(\frac{\pi_\theta(o_t|q,o<t)}{\pi_{\theta_{old}}(o_t|q,o<t)}A_t, clip(\frac{\pi_\theta(o_t|q,o<t)}{\pi_{\theta_{old}}(o_t|q,o<t)}, 1-\varepsilon, 1+\varepsilon)A_t\right) - D_{KL}\right]$$

其中：

- **期望符号** $\mathbb{E}$ **表示对查询** q **和输出** o **的期望**:
- $q \sim P(Q)$ ：查询 q 从分布 $P(Q)$ 中采样，理解为 prompts。
- $o \sim \pi_{\theta_{old}}(O|q)$ ：输出 o 由旧策略 $\pi_{\theta_{old}}(O|q)$ 生成。
- $\frac{1}{|o|}\sum_{t=1}^{|o|}$ 对输出 o 的每个时间步 t 求平均:
- $|o|$ 是输出序列的长度。

其**核心目标函数**为：

$$L^{CLIP}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right) - D_{KL}\right]$$

其中：

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ 是新旧策略的概率比。
- $\hat{A}_t$ 是优势函数，衡量动作的相对好坏。
- ϵ 是裁剪函数的参数，通常为 0.1 或 0.2。
- D_{KL} 为KL散度惩罚项。

步骤

1. **采样**：使用当前策略与环境交互，收集数据，在语言模型中，可以类比为生成补全（generating completions）。
2. **计算优势值**：基于收集的数据计算优势值函数 $\hat{A}_t$ 。
3. **优化目标函数**：通过**梯度上升**优化目标函数 $L^{CLIP}(\theta)$ 。
4. **更新策略**：重复上述步骤，直到策略收敛。

优点

- **稳定性**：通过裁剪机制，避免策略更新过大。
- **高效性**：相比 TRPO，PPO 实现更简单，计算效率更高。

Advantage（优势函数）

定义

Advantage函数用于衡量在某个状态（State）下，采取某个动作（Action）相对于平均表现的优劣程度。它的数学定义为：

$A(s,a) = Q(s,a) - V(s)$ ，

其中：

- Q(s, a) 是**动作值函数**，表示在状态 s 下采取动作 a 后，未来累积回报的期望。
- V(s) 是**状态值函数**，表示在状态 s 下，按照当前策略采取动作后，未来累积回报的期望。
- A(s, a) 是**优势函数**，表示在状态 s 下采取动作 a 比平均表现好多少（或差多少）。

作用

- Advantage函数用于指导策略更新：
  - 如果 A(s, a) > 0 ，说明动作 a 比平均表现更好，策略应该更倾向于选择这个动作；
  - 如果 A(s, a) < 0 ，说明动作 a 比平均表现更差，策略应该减少选择这个动作的概率。
- 在PPO等算法中，Advantage函数通常通过**GAE（Generalized Advantage Estimation）**来估计。

直观理解

Advantage函数就像一个“评分”，告诉模型某个动作在当前状态下是好还是坏，以及好（或坏）的程度，**依赖 Critical Model**。

效果：能降低样本方差，提升训练的稳定性。

如果没有 Critial Model 是否可以？可以。通过多次采样的方式，如 DeepSeek，**用时间换空间**。

函数比例 $r_t(\theta)$

在 PPO 中， $r_t(\theta)$ 是新策略 π_θ 和旧策略 $\pi_{\theta_{\text{old}}}$ 在状态 s_t 下选择动作 a_t 的概率比。

- If $r_t(\theta) > 1$, the **action** a_t **at state** s_t **相对于旧的策略，更倾向选择当前策略**
- If $r_t(\theta)$ is between 0 and 1, the **action 选择当前策略的倾向性偏低**

Clip 限制

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

用于策略的选择，同时基于 clip 控制更新的幅度。通过裁剪机制限制其变化范围，确保训练的稳定性。

KL Penalty（KL散度惩罚）

定义

KL Penalty是基于**KL散度（Kullback-Leibler Divergence）**的一种正则化手段。KL散度用于衡量两个概率分布之间的差异。在强化学习中，KL Penalty通常用于限制当前策略 π_θ 和参考策略 π_{ref} 之间概率分布的差异。其数学定义为： $\text{KL Penalty} = D_{\text{KL}}(\pi_{\text{ref}} || \pi_\theta)$ 其中：

- π_θ 是当前策略（由模型参数 θ 决定）。
- π_{ref} 是参考策略（通常是更新前的策略或某个基线策略）。
- D_{KL} 是KL散度，用于衡量两个策略之间的差异。

作用

- KL Penalty用于防止策略更新过大，确保当前策略不会偏离参考策略太远。这样可以避免训练过程中的不稳定现象（如策略崩溃）。
- 在PPO等算法中，KL Penalty通常被添加到目标函数中，作为正则化项。

直观理解

KL Penalty就像一个“约束”，告诉模型在更新策略时不要“步子迈得太大”，以免失去稳定性。

Advantage和KL Penalty的关系

- **Advantage** 用于指导策略更新，告诉模型哪些动作更好。

- **KL Penalty** 用于约束策略更新，防止策略变化过大。

- 在PPO等算法中，Advantage和KL Penalty共同作用，既鼓励模型选择更好的动作，又确保更新过程稳定可靠。

举例说明

假设我们训练一个机器人走迷宫：

- **Advantage**：机器人发现“向右转”比“向左转”更容易找到出口，于是Advantage函数会给“向右转”一个正的值，鼓励策略更倾向于选择“向右转”。

- **KL Penalty**：为了防止策略突然变得只选择“向右转”而忽略其他可能性，KL Penalty会限制策略的变化幅度，确保策略更新是平滑的。

总结

- **Advantage（优势函数）**：衡量某个动作比平均表现好多少，用于指导策略更新。

- **KL Penalty（KL散度惩罚）**：限制策略更新的幅度，确保训练过程的稳定性。

群体相对策略优化（GRPO）

GRPO 是一种在线学习算法（online learning algorithm），这意味着它通过使用训练过程中由训练模型自身生成的数据来迭代改进。GRPO 的目标直觉是最大化生成补全（completions）的优势函数（advantage），同时确保模型保持在参考策略（reference policy）附近。

其目标函数为：

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\text{old}}(O|q)} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(r_{i,t}(\theta)\hat{A}_{i,t} - \beta D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}}) \right) \right]$$

为了理解 GRPO 的工作原理，可以将其分解为四个主要步骤：

1. 生成补全（Generating completions）
2. 计算优势值（Computing the advantage）
3. 估计KL散度（Estimating the KL divergence）
4. 计算损失（Computing the loss）

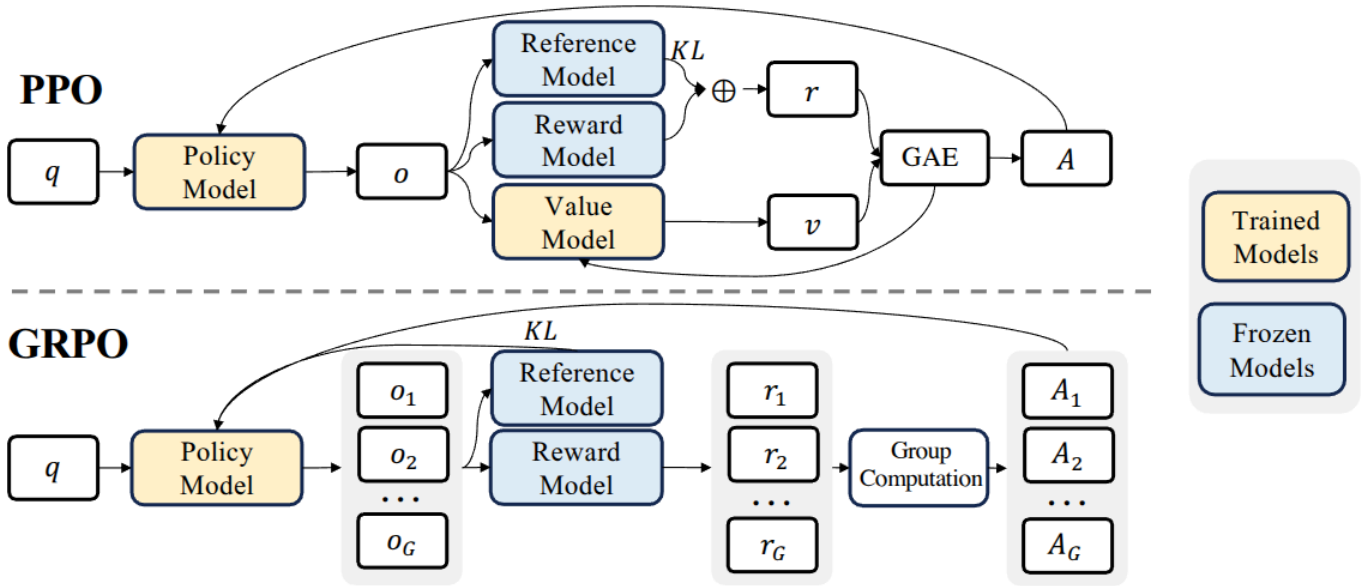


Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

1. 生成补全（Generating completions）

在每一个训练步骤中，我们从提示（prompts）中采样一个批次（batch），并为每个提示生成一组 G 个补全（completions）（记为 o_i ）。

2. 计算优势值（Computing the advantage）

对于每一个 G 序列，使用奖励模型（reward model）计算其奖励（reward）。为了与奖励模型的比较性质保持一致——通常奖励模型是基于同一问题的输出之间的比较数据集进行训练的——优势的计算反映了这些相对比较。其归一化公式如下：

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$$

这种方法赋予了该方法其名称：**群体相对策略优化（Group Relative Policy Optimization, GRPO）**

GRPO通过优化PPO算法，解决了计算优势值时需要同时依赖奖励模型（reward model）和价值模型（value model）的问题，成功移除了value model（价值模型），显著降低了推理时的内存占用和时间开销。**Advantage（优势值）**的核心价值在于为模型输出提供更精准的评估，不仅衡量答案的绝对质量，还通过相对比较（与其他回答的对比）来更全面地定位其优劣。

Reward Model 的来源：

1. 基于规则的：针对 reasoning 数据，通过编译器、执行结果等方式给出 结果 reward，不是 过程 reward；文本一致的 reward；人工设计的 rule-based reward；
2. 基于模型的：通过 Deepseek-v3 训练得到的偏好模型

3. 估计KL散度（Estimating the KL divergence）

在实际算法实现中，直接计算KL散度可能会面临一些挑战：

- **计算复杂度高**：KL散度的定义涉及对两个概率分布的对数比值的期望计算。对于复杂的策略分布，直接计算KL散度可能需要大量的计算资源；
- **数值稳定性**：在实际计算中，直接计算KL散度可能会遇到数值不稳定的问题，尤其是当两个策略的概率分布非常接近时，对数比值可能会趋近于零或无穷大。近似器可以通过引入一些数值稳定性的技巧（如截断或平滑）来避免这些问题；

- **在线学习**：在强化学习中，策略通常需要在每一步或每几步更新一次。如果每次更新都需要精确计算KL散度，可能会导致训练过程变得非常缓慢。近似器可以快速估计KL散度，从而支持在线学习和实时更新。

Schulman et al. (2020) 提出的近似器可以根据当前策略和参考策略的差异动态调整估计的精度，从而在保证计算效率的同时，尽可能减少估计误差，其定义如下：

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

这个近似器的核心思想是通过对当前策略和参考策略的概率比值的简单变换来估计KL散度。具体来说：

- **第一项**： $\frac{\pi_{\mathrm{ref}}(o_{i,t} \mid q, o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}$ 是参考策略与当前策略的概率比值。
- **第二项**： $\log \frac{\pi_{\mathrm{ref}}(o_{i,t} \mid q, o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}$ 是对数概率比值。
- **第三项**：-1 是一个常数项，用于调整近似器的偏差。

这个近似器的优势在于它只需要计算当前策略和参考策略的概率比值，而不需要直接计算KL散度的积分或期望。因此，它可以在保证一定精度的同时，显著降低计算复杂度。

近似器的直观理解

这个近似器的设计灵感来自于泰勒展开。KL散度可以看作是两个分布之间的某种“距离”，而这个近似器通过一阶或二阶近似来估计这个距离。具体来说：

- 当 π_θ 和 π_{ref} 非常接近时， $\frac{\pi_{\mathrm{ref}}}{\pi_\theta} \approx 1$ ，此时 $\log \frac{\pi_{\mathrm{ref}}}{\pi_\theta} \approx 0$ ，近似器的值趋近于零，符合KL散度的性质。
- 当 π_θ 和 π_{ref} 差异较大时，近似器会给出一个较大的正值，反映出两个分布之间的差异。

Note: π_{ref} 的来源一般是初始的 SFT 模型。

4. 计算损失（Computing the loss）

这一步的目标是最大化优势，同时确保模型保持在参考策略附近。因此，损失定义如下：

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left[\frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{[\pi_\theta(o_{i,t} \mid q, o_{i,<t})]_{\text{no grad}}}\hat{A}_{i,t} - \beta\mathbb{D}_{\text{KL}}\left[\pi_\theta \| \pi_{\text{ref}}\right]\right]$$

其中第一项表示缩放后的优势，第二项通过KL散度惩罚与参考策略的偏离。

在原始论文中，该公式被推广为在每次生成后通过利用**裁剪替代目标（clipped surrogate objective）**进行多次更新：

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left[\min\left(\frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}\hat{A}_{i,t},\ \text{clip}\left(\frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}, 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,t}\right) - \beta\mathbb{D}_{\text{KL}}\left[\pi_\theta \| \pi_{\text{ref}}\right]\right]$$

其中 $\text{clip}(\cdot, 1-\epsilon, 1+\epsilon)$ 通过将策略比率限制在 $1-\epsilon$ 和 $1+\epsilon$ 之间，确保更新不会过度偏离参考策略。

参考链接

Deepseek 强化学习训练流程：https://zhuanlan.zhihu.com/p/19969128139

DeepSeek 强化学习算法：https://www.zhihu.com/question/10766825126/answer/88583863333