GRPO：Group Relative Policy Optimization

知行者
大模型持续炼丹

DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

Zhihong Shao[1,2*†], Peiyi Wang[1,3*†], Qihao Zhu[1,3*†], Runxin Xu[1], Junxiao Song[1]
Xiao Bi[1], Haowei Zhang[1], Mingchuan Zhang[1], Y.K. Li[1], Y. Wu[1], Daya Guo[1*]
[1]DeepSeek-AI, [2]Tsinghua University, [3]Peking University
{zhihongshao,wangpeiyi,zhuqh,guoday}@deepseek.com
https://github.com/deepseek-ai/DeepSeek-Math

paper: arxiv.org/pdf/2402.0330...

这里快速介绍一下 deepseek 提出来的这个GRPO 的算法原理。 暂时不对论文通篇进行讲解了。

GRPO的核心思想是通过**组内相对奖励**来估计基线（baseline），从而避免使用额外的价值函数模型（critic model）。传统的PPO算法需要训练一个价值函数来估计优势函数（advantage function），而GRPO通过从同一问题的多个输出中计算平均奖励来替代这一过程，显著减少了内存和计算资源的消耗。

1. 框架图

首先看一下PPO 与GRPO 的比较图。 对PPO 算法不熟悉的话，可以查看前一篇文章： 知行者：PPO: Proximal Policy Optimization Algorithms

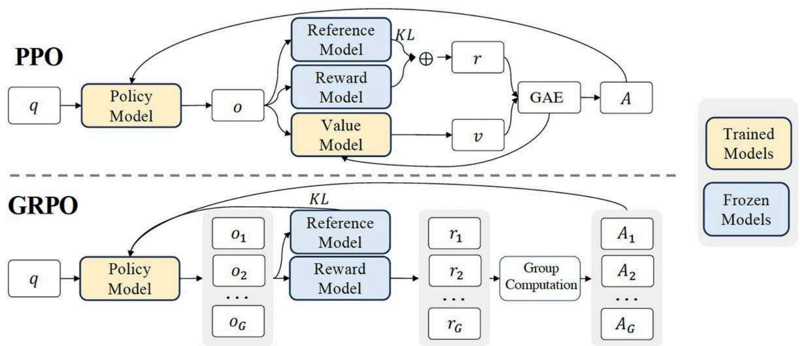


Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

从图上可以看出，GRPO 与PPO 的主要区别有：

- GRPO 省略了 value function model.
- GRPO reward 计算，改成了一个q 生成多个r, 然后reward 打分。
- PPO 优势函数计算时，KL 是包含在GAE内部的。 GRPO 直接挪到了外面，同时修改了计算方法。

2.1 PPO 复习

直接上原文，首先是PPO 的目标函数⁺：

这个比较熟悉，策略概率比与优势函数的乘积。 同时做了clip限制了参数更新范围。

4.1.1. *From PPO to GRPO*

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is an actor-critic RL algorithm that is widely used in the RL fine-tuning stage of LLMs (Ouyang et al., 2022). In particular, it optimizes LLMs by maximizing the following surrogate objective:

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

where π_θ and $\pi_{\theta_{old}}$ are the current and old policy models, and q, o are questions and outputs sampled from the question dataset and the old policy $\pi_{\theta_{old}}$, respectively. ε is a clipping-related hyper-parameter introduced in PPO for stabilizing training. A_t is the advantage, which is computed by applying Generalized Advantage Estimation (GAE) (Schulman et al., 2015), based

[6] https://github.com/InternLM/InternLM-Math

公式2 是PPO 中优势函数的计算。 在reward 打分上，加一个per-token 的KL 散度⁺惩罚。

on the rewards $\{r_{\geq t}\}$ and a learned value function V_ψ. Thus, in PPO, a value function needs to be trained alongside the policy model and to mitigate over-optimization of the reward model, the standard approach is to add a per-token KL penalty from a reference model in the reward at each token (Ouyang et al., 2022), i.e.,

$$r_t = r_\varphi(q, o_{\leq t}) - \beta \log \frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{ref}(o_t|q, o_{<t})}, \quad (2)$$

where r_φ is the reward model, π_{ref} is the reference model, which is usually the initial SFT model, and β is the coefficient of the KL penalty.

2.2 GRPO的优化

下面是GRPO 的改进。 **论文认为value function model 占用了额外的显存和计算资源**。因此提出以下的改进方法。

去除value function , reward 直接对单个q生成的response进行打分，归一化后，作为替代的优势函数。

同时将KL散度抑制，移到了优势函数计算的外面。 KL 散度的计算也进行了改进，可以见公式4. 为了保证KL散度⁺为正值。

As the value function employed in PPO is typically another model of comparable size as the policy model, it brings a substantial memory and computational burden. Additionally, during RL training, the value function is treated as a baseline in the calculation of the advantage for variance reduction. While in the LLM context, usually only the last token is assigned a reward score by the reward model, which may complicate the training of a value function that is accurate at each token. To address this, as shown in Figure 4, we propose Group Relative Policy Optimization (GRPO), which obviates the need for additional value function approximation as in PPO, and instead uses the average reward of multiple sampled outputs, produced in response to the same question, as the baseline. More specifically, for each question q, GRPO samples a group of outputs $\{o_1, o_2, \cdots, o_G\}$ from the old policy $\pi_{\theta_{old}}$ and then optimizes the policy model by maximizing the following objective:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)]$$
$$\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min \left[\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})} \hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}, 1-\varepsilon, 1+\varepsilon \right) \hat{A}_{i,t} \right] - \beta \mathbb{D}_{KL}\left[\pi_\theta || \pi_{ref}\right] \right\}, \quad (3)$$

where ε and β are hyper-parameters, and $\hat{A}_{i,t}$ is the advantage calculated based on relative rewards of the outputs inside each group only, which will be detailed in the following subsections. The group relative way that GRPO leverages to calculate the advantages, aligns well with the comparative nature of rewards models, as reward models are typically trained on datasets of comparisons between outputs on the same question. Also note that, instead of adding KL penalty in the reward, GRPO regularizes by directly adding the KL divergence between the trained policy and the reference policy to the loss, avoiding complicating the calculation of $\hat{A}_{i,t}$.

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

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

which is guaranteed to be positive.

下图是基于group reward 计算优势函数的，归一化公式：

4.1.2. *Outcome Supervision RL with GRPO*

Formally, for each question q, a group of outputs $\{o_1, o_2, \cdots, o_G\}$ are sampled from the old policy model $\pi_{\theta_{old}}$. A reward model is then used to score the outputs, yielding G rewards $\mathbf{r} = \{r_1, r_2, \cdots, r_G\}$ correspondingly. Subsequently, these rewards are normalized by subtracting the group average and dividing by the group standard deviation. Outcome supervision provides the normalized reward at the end of each output o_i and sets the advantages $\hat{A}_{i,t}$ of all tokens in the output as the normalized reward, i.e., $\hat{A}_{i,t} = \widetilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$, and then optimizes the policy by maximizing the objective defined in equation (3).

下面是GRPO 的计算伪代码⁺：

Algorithm 1 Iterative Group Relative Policy Optimization

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

```
policy model π_θ ← π_θinit
for iteration = 1, ..., I do
    reference model π_ref ← π_θ
    for step = 1, ..., M do
        Sample a batch D_b from D
        Update the old policy model π_θold ← π_θ
        Sample G outputs {o_i}_{i=1}^G ~ π_θold(· | q) for each question q ∈ D_b
        Compute rewards {r_i}_{i=1}^G for each sampled output o_i by running r_φ
        Compute Â_{i,t} for the t-th token of o_i through group relative advantage estimation.
        for GRPO iteration = 1, ..., μ do
            Update the policy model π_θ by maximizing the GRPO objective (Equation 21)
    Update r_φ through continuous training using a replay mechanism.
```

Output π_θ

GRPO的计算流程包括：

1. 采样一组输出并计算每个输出的奖励。
2. 对组内奖励进行归一化处理。
3. 使用归一化后的奖励计算优势函数。
4. 通过最大化目标函数更新策略模型。
5. 迭代训练，逐步优化策略模型。

GRPO通过组内相对奖励估计基线，避免了传统PPO中价值函数⁺的使用，显著减少了训练资源消耗，同时提升了模型在数学推理⁺等复杂任务中的表现。

3. GRPO 计算总结

GRPO 计算流程

1. 初始化

- **策略模型**：初始化策略模型 π_θ，通常基于预训练的语言模型或经过监督微调（SFT）的模型。
- **奖励模型**：初始化奖励模型 r_ϕ，用于对模型的输出进行评分。
- **参考模型**：设置参考模型 π_{ref}，通常是初始的SFT模型，用于计算KL散度以防止策略模型过度偏离初始模型。

---

2. 采样输出

对于每个问题 q：

1. 从当前策略模型 $\pi_{\theta_{old}}$ 中采样一组输出 $\{o_1, o_2, \ldots, o_G\}$，其中 G 是组的大小（例如 $G = 64$）。

3. 归一化奖励

1. 计算组内奖励的平均值：

$$\text{mean}(\mathbf{r}) = \frac{1}{G}\sum_{i=1}^{G} r_i$$

2. 计算组内奖励的标准差：

$$\text{std}(\mathbf{r}) = \sqrt{\frac{1}{G}\sum_{i=1}^{G}(r_i - \text{mean}(\mathbf{r}))^2}$$

3. 对每个输出 o_i 的奖励 r_i 进行归一化处理：

$$\tilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$$

---

4. 计算优势函数

- 对于每个输出 o_i，其每个时间步 t 的优势函数 $\hat{A}_{i,t}$ 被设置为归一化后的奖励 $\tilde{r}_i$：

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

- 如果使用**过程监督**（Process Supervision），则优势函数会基于每个推理步骤的奖励进行计算。

- 这里的公式缺了一部分，不要在意。可以查看查看原文截图。

5. 优化策略模型

通过最大化以下目标函数来更新策略模型 π_θ：

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(\min\left[\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}\hat{A}_{i,t}, \text{cli}\right.\right.$$

其中：

- ϵ 是裁剪参数，用于限制策略更新的幅度。
- β 是KL散度的系数，用于防止策略模型过度偏离参考模型。
- $D_{KL}[\pi_\theta||\pi_{ref}]$ 是策略模型与参考模型之间的KL散度。

---

6. 迭代训练

- **更新奖励模型**：在每一轮训练后，根据新的采样输出更新奖励模型 r_ϕ。
- **更新策略模型**：使用更新后的奖励模型继续优化策略模型 π_θ。
- **迭代**：重复上述过程，直到策略模型收敛或达到预定的训练轮数。

GRPO 的确节约了显存和计算资源。 但是是否真的提升复杂任务能力保留疑问。

编辑于 2025-01-30 13:48 · IP 属地北京

大语言模型

理性发言，友善互动

31 条评论 默认 最新

zhao
可以这样理解么：就是不要AC框架了，索性直接回到了类似REINFORCE的原始的policy' gradient算法，但是R(tau)不是一次采样而是多次采样的平均值。
01-27 · 美国 回复 9

py交易
最近有个算法就叫reinforce++。估计对于语言类任务并不需要考虑太强的中间判据？
02-01 · 新加坡 回复 1

py交易
可能当每一步action都有well representation之后，critic的作用就不明显了。
02-01 · 新加坡 回复 1

展开其他 1 条回复 ›

zhao
我理解其实还是这个最本源的intuition：LM场景下本来就没有高质量的step (token) reward，而PPO的Advantage计算其实是高度依赖step reward的，所以PPO天然的就不适用LM场景，只不过先驱巨头这样用了大家都follow。现在人们意识到直接用高质量的trajectory-end reward反而效果更好，不该用PPO，至少不该用未加修改的PPO。PPO的Advantage是来自于step reward以及Critic（Critic就是ValueModel(state)）。GRPO修改了PPO里面的Advantage，从他们的图中能看出来它抛弃了ValueModel，抛了ValueModel还怎么计算Advantage？它索性用了end-trajectory-reward作为每一步的Advantage。某种意义上这个只是名字叫Advantage，其实已经不是传统意义上的Advantage了，而更像是更本源的REINFORCE，即：这个loss的意义是找出最佳action使得获取的reward最大化 @py交易
02-03 · 美国 回复 8

Bayes ‣ **超级大笨蛋**
不太认同，我还是认为prm是可行的路径。目前结果只能说grpo这样假设中间每个token都一样不比ppo差，但是ppo里的critic感觉对于过程建模的也不太靠谱，直接用kl散度跟下雪也没什么区别了。只从grpo可行也不一定能说prm相比orm没有优势
19 小时前 · 马来西亚 回复 喜欢

超级大笨蛋
完全赞同你的观点，我也认为 过程监督其实是AI的中间产物。只要探索路径足够多，根据结果也可以做出不错的成绩，ds挖了一个大坑，接下来很多人要去挖掘了哈哈哈
昨天 01:05 · 北京 回复 喜欢

flaneur
请问如果它没有 reward model 的话，最后是怎么打分的呢？
01-28 · 山东 回复 1

知行者 作者
有reward model。 看GRPO的结构图
01-30 · 北京 回复 2

陈仲
有 reward model，只是没有value model
昨天 11:19 · 新加坡 回复 1

HAN Bing
一直感觉语言模型用ppo就是脱裤子放屁。本身reward model都是估计，还要再用value model 估计advantage。其实只要模型参数够大，采样路径够大，就不用value model了。
02-01 · 四川 回复 2

超级大笨蛋
嗯 有点道理，但是实际上 value model的含义是对过程进行监督(个人理解)，critic value对每一个step状态 后续可以得到的奖励进行预估。实际上 确实在某种程度上可以帮助模型更好的收敛，但是 基于ds想法 我们可以不考虑这个过程是怎么来的 只要结果对就行，其实这样更符合RL 或者更符合AI的思路，路径那么多条 为什么我都要判断是否合理，拿结果来说事不更好么，而且 这样也能发现很多看似不合理 但是最终结果确不错的过程。explore
昨天 01:02 · 北京 回复 喜欢

一维无限深势阱
感谢作者分享！有个小问题，您回答中的GRPO计算流程step 5 "迭代训练，逐步优化策略模型和奖励模型"是不是应该为 “迭代训练，逐步优化策略模型”？
01-29 · 澳大利亚 回复 2

知行者 作者
是的，手误已修改。
01-30 · 北京 回复 1

py交易

01-25 · 新加坡 回复 1

py交易 ‣ **知行者**
进一步地，哪怕回复里面存在不属于被训练模型所生成的内容也能给出一个reward，并进行训练从而更新参数?
02-01 · 新加坡 回复 1

知行者 作者
是的
01-30 · 北京 回复 喜欢

展开其他 2 条回复 ›

不知道叫什么
感觉GRPO是专门为大模型的训练做的优化，目的就是要节省资源，那对于机器人的强化学习，模型基本都不大，感觉这样GRPO可能还不如PPO
01-30 · 山东 回复 1

知行者 作者
相对于PPO主要是为了节约资源。
01-30 · 北京 回复 2

整个人都震惊了1
GRPO训练过程中还要更新reward model，还挺有意思。。。
刚刚 · 浙江 回复 喜欢

小鱼儿at青岛
看伪代码，GRPO目标函数的iteration迭代，是说算出来的objective反复N次backward？中途不需要基于新的policy重算一下probs和advan吗
3 小时前 · 山东 回复 喜欢

LcRobin
可以理解为随着迭代，Reference Model = last step Policy Model吗?
5 小时前 · 浙江 回复 喜欢

LcRobin
θ = next step Policy Model ; θold = this step Policy Model ; Reference Model = last step Policy Model. 这样理解对吗
5 小时前 · 浙江 回复 喜欢

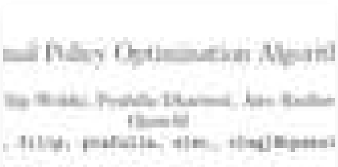
点击查看全部评论 ›

理性发言，友善互动

推荐阅读

监督&强化学习模型在金融市场的应用
Yushu

在线学习(MAB)与强化学习(RL)[8]：Policy Iteration和...
覃含章 发表于Optim...



【强化学习算法 4】PPO
张海抱

Trust Region Policy Optimization (TRPO) 论文...
一、介绍 本文作者首次证明了最小化一个确定的代理目标方程可以以一个较大的步长来提升策略。他们采用一系列的简化来逼近理论算法，从而产生了TRPO算法。文章提出了两种算法的实施方式：第...
陈子纮 发表于强化学习-...