0001年1月1日 3 分钟阅读

DeepSeek R1 的技术流程

DeepSeek R1 的技术流程可总结为以下范式： 1.DeepSeek R1-Zero 的生成： 基于 DeepSeek V3-Base 模型，通过强化学习（RL），直接训练出 DeepSeek R1-Zero 模型。该阶段不进行监督微调 (SFT)，旨在探索模型自主发展推理能力的潜力。 2.推理链可读性增强：

> 冷启动数据微调： 采用高质量的冷启动数据（包括人工专家撰写和模型生成并经过筛选的高质量、符合格式规范的推理数据）对 R1-Zero 进行监督微调。
>
> 以推理为中心的强化学习： 以微调后的模型为基础，进一步进行强化学习，从而提升推理链的可读性。 3.通用能力和安全性提升：
>
> 全领域监督微调： 通过拒绝采样 (Rejection Sampling) 筛选高质量数据，并结合全领域数据进行监督微调，提升模型的通用能力。
>
> 全领域强化学习： 在全领域任务上进行强化学习训练：
>
> 推理任务：采用规则奖励。
>
> 通用任务 (如聊天)：进行偏好建模。
>
> 通过上述措施，在提升模型通用能力的同时，增强其安全性。

DeepSeek R1 Zero 的强化训练过程

DeepSeek R1 Zero 是完全从基础模型（DeepSeek V3)开始构建，完全依赖强化学习，而不使用人类专家标注的监督微调（SFT）。在训练过程中随着训练步骤的增加，模型也是逐渐展现出长文本推理以及长链修复的能力。随着推理路径的逐步增长，模型来表现出自我反思的能力，能够发现并修复之前的错误。DeepSeek R1-Zero 通过 直接在基础模型上应用强化学习，并设计 基于规则的奖励函数，实现了在没有监督数据的情况下发展强大的推理能力 DeepSeek R1 Zero 的强化训练过程中，设计了奖励机制，以优化模型的推理能力。具体来说，奖励机制的设计主要集中在以下几个方面： 没有使用监督微调 (SFT)：DeepSeek R1-Zero 直接应用强化学习 (RL) 到基础模型，而没有依赖于监督微调作为初步步骤。这种方法允许模型探索解决复杂问题的思维链 (CoT)，从而发展

目录

文章信息

字数

阅读时间

发布时间

DeepSeek R1-Zero。 奖励函数：DeepSeek R1 采用 基于规则的奖励系统，而不是神经奖励模型，以避免奖励 “黑客行为” 和过度的计算成本。主要的奖励函数包括：

> 准确性奖励：确保模型生成在事实上正确且可验证的响应。这对于具有确定性结果的任务（如数学和编码）特别有用。DeepSeek R1 在奖励建模中，采用基于规则的奖励，直接利用程序进行判断正误的奖励信号 [110, see earlier turn].
>
> 格式奖励： 显式地规劝模型的输出过程必须包含思考的过程，也就是利用一个 sinking token 将思考的过程圈起来 [57, see earlier turn]。

GRPO 算法： DeepSeek R1 的强化学习管线以 GRPO 为中心，GRPO 提供了一种轻量级但功能强大的优化机制。其关键创新包括移除评论家模型，从而显著减少了内存开销；通过基于群组的优势估计来稳定策略更新；以及与基于 PPO 的方法相比，在保持强大性能的同时实现高效训练。 自进化过程：DeepSeek R1-Zero 的自进化过程展示了强化学习如何驱动模型自主提高其推理能力。 测试时计算：为了提高 DeepSeek R1-Zero 的性能，可以采用多数投票的方式 [45, see earlier turn]。对每个问题采样多个回答，并选择出现频率最高的答案作为最终结果 [45, see earlier turn]。 总体而言，DeepSeek R1-Zero 通过 直接在基础模型上应用强化学习，并设计基于规则的奖励函数，实现了在没有监督数据的情况下发展强大的推理能力。 需要注意的是这部分奖励建模并没有采用先前我们经常讨论的比如说过程奖励模型 PRM 甚至没有采用奖励模型。这里边的主要考量是基于神经网络的奖励模型都有可能遭受奖励攻陷的问题，一旦发生奖励攻陷模型就可能陷入局部最优解，而重新训练奖励模型需要大量的计算资源可能会复杂化整个流程。

预训练阶段的扩展律：其实也就是在预训练模型上，计算量数据和参数量成一个类似于正比的关系，也就是算力等于 6 倍的参数量乘上数据量。因此在大模型时代发展的初期，囤卡提升预训练的算力和模型参数变成了主要目标。

随着 OpenAI o1 的发布，也证明了在强化学习加持下后训练时代一个新的扩展律：随着模型在后训练阶段的训练时计算量和测试时计算量的提升，模型的性能特别是数学代码能力也会随之提升。那么在后训练扩展律下语言模型的训练时计算量多了一个新的变量，也就是在探索时语言模型推理产生的计算量。

DeepSeek 的成功也为我们带来了一些关键的启示：例如在传统的大语言模型训练中监督微调（SFT) 通常被认为是不可或缺的一环，其逻辑是先用大量人工标注的数据来让模型初步掌握某种能力或回答范式，再利用强化学习进一步优化模型的性能。 然而 DeepSeek 却打破了这一传统，他们选择直接将 RL 应用于基础模型，而没有经过任何形式的 SFT 训练。这种纯强化学习的方法之所以如此引人注目，是很大程度上因为它抛弃了对于大规模人工标注数据的依

赖。众所周知 SFT 是非常需要消耗大量的人力物力来构建和维护高质量的训练数据集，而 DeepSeek 的团队这种做法可以直接让模型在强化学习的环境中进行自我探索，通过与环境的互动，自主的去发现和学习解决复杂问题的能力，就好比一个初学者在没有老师的指导下通过不断的尝试和错误，来掌握一门新的技能。这种自主学习的方式，不仅节省了大量的标注成本，更重要的是它能让模型更加自由地探索解决问题的路径，而不是被预先设定的模式所束缚，这也使得模型最终具备了更加强大的泛化能力和适应能力。

但是 DeepSeek-R1 Zero也有对应的问题，比如说长推理过程可读性差，语言混合帮助性低。那么我们能否在 zero 的基础上，在兼顾推理性能的同时，提升模型的帮助性和安全性的。例如能不能产生一些比较清晰且直接的推理过程，并且能够泛化到通用能力任务上的模型。例如 R1；以及我们能否利用一些高质量的反思数据去做冷启动，从而加速强化学习的收敛或者帮助提升推理表现。那么围绕这两个研究问题，应运而生了 DeepSeek R1 这个模型。

DeepSeek-R1 Zero 的局限性

DeepSeek-R1 Zero虽然在强化学习中展现出强大的推理能力，但仍然进化到DeepSeek-R1，主要是因为R1 Zero存在一些先天缺陷，导致其在实际应用中受到限制。 以下是DeepSeek-R1 Zero的主要缺陷以及DeepSeek-R1如何改进这些缺陷的详细解释：

可读性差 (Poor Readability)：

问题：DeepSeek-R1 Zero生成的内容通常不适合阅读。其输出可能混合多种语言，或者缺乏Markdown格式来突出显示答案。

解决方案：DeepSeek-R1通过收集高质量的数据来微调DeepSeek-V3基础模型，作为强化学习的起点。在创建用于DeepSeek-R1的冷启动数据时，设计了一种可读的模式，包括在每个响应的末尾添加摘要，并过滤掉不适合阅读的响应。

语言混合 (Language Mixing)：

问题：DeepSeek-R1 Zero在处理非中文或英文的查询时，可能会使用英文进行推理和回答。

解决方案：DeepSeek-R1 优化了中文和英文的处理，并通过多阶段训练和少量冷启动数据来提高推理性能和可用性。

通用能力不足 (Lack of General Capability)：

问题：DeepSeek-R1 Zero在函数调用、多轮对话、复杂角色扮演和JSON输出等任务中的能力不如DeepSeek-V3。

解决方案：DeepSeek-R1在最后的监督微调 (SFT) 和强化学习 (RL) 训练阶段加入指令遵循数据，从而提高模型理解和遵循用户定义格式约束的能力。通过大规模强化学习 (RL) 来增强STEM相关问题的准确性，并在各种任务中展示了强大的泛化能力。

对提示词敏感 (Sensitive to Prompts):

问题: DeepSeek-R1 对提示词非常敏感，少量样本提示会持续降低其性能。

解决方案: 建议用户直接描述问题，并使用零样本设置指定输出格式，以获得最佳结果。

训练数据 (Cold Start Data)：

问题：DeepSeek-R1 Zero 没有任何监督微调 (SFT)，从DeepSeek-V3-Base直接应用强化学习 (RL)。

解决方案：DeepSeek-R1 通过在进行 RL 之前，在数千个高质量思维链 (CoT) 示例上进行训练，从而确保更结构化的学习轨迹。

奖励模型 (Reward Hacking)：

问题: 神经奖励模型容易受到奖励利用的影响，且需要昂贵的再训练。

解决方案: DeepSeek-R1 采用确定性的、基于规则的方法，保证更高的透明度、降低计算成本，并提供更稳定的训练动态。 总而言之，DeepSeek-R1通过引入冷启动数据、优化训练流程和改进奖励机制，克服了DeepSeek-R1 Zero的局限性，使其在可读性、语言一致性、通用能力和实际应用方面都得到了显著提升。

DeepSeek R1 的技术 pipeline

总的来说 DeepSeek R1 的技术 pipeline 可以被总结为这么一套范式。 首先第一基于 DeepSeek v3-base 产生了 DeepSeek R1 Zero 这个模型， 第一阶段是我们希望先增强 R1 zero 的推理链的可读性，在这一阶段我们会利用一些冷启动的数据，这些数据里边可能是包含了人类专家和模型所撰写的高质量的语言，符合语言格式的这样一些反思数据。 然后我们再以推理为中心的强化学习去进一步的去进行微调，从而获得一个相对推理链可读性更强的一个中间模型； 那么更进一步我们采用传统 RLHF 中的一些技术，比如说通过拒绝采样和全领域的监督微调以及在全领域的任务上进行强化学习的训练，比如对于推理任务我们可以使用规则奖励，而对于一些通用比如说聊天任务我们进行偏好建模，从而来在第二阶段去提升模型的通用能力和安全性，最终获得了 DeepSeek R1 这样一个模型。

GRPO

GRPO（Group Relative Policy Optimization，群组相对策略优化）是一种用于训练AI模型的强化学习算法，尤其在DeepSeek R1等模型中发挥着关键作用。它主要用于提升模型的推理能力，其核心思想是在训练过程中，通过比较同一问题的多个答案，来判断哪个答案更好，从而优化模型的策略。 下面用更简单的语言来描述GRPO的工作原理： 1.提出问题，生成多个答案：

首先，给模型提出一个问题（例如一道数学题）。

然后，模型尝试给出多个不同的答案。这些答案可能有些是正确的，有些是错误的，也可能有些是部分正确的。 2.评估答案，计算相对奖励：

接着，对每个答案进行评估，判断其质量。这个评估不一定需要非常精确，而是相对的。

GRPO会计算每个答案在同一组答案中的相对好坏程度，也就是计算一个“相对奖励”。例如，如果一个答案比其他答案更接近正确答案，那么它的相对奖励就更高。 3.优化策略，提升模型能力：

最后，GRPO利用这些相对奖励来调整模型的策略，使其更倾向于生成高质量的答案。

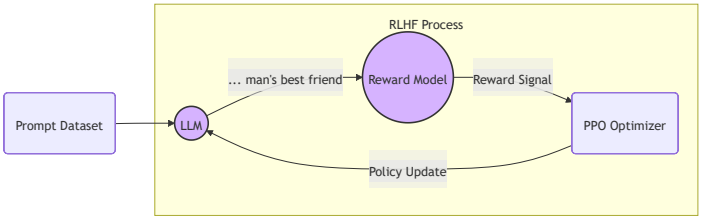
这个过程就像老师给学生批改作业，不是简单地判断对错，而是比较不同学生的解题思路，鼓励更优秀的解法。 GRPO的关键特点和优点：

不需要额外的“评论家”模型：传统的强化学习算法通常需要一个额外的“评论家”模型来评估答案的质量。GRPO通过比较同一问题的多个答案来计算相对奖励，从而避免了对“评论家”模型的需求，降低了计算成本。

稳定训练过程：GRPO通过对奖励进行归一化处理，可以稳定训练过程，避免模型过度优化或陷入局部最优解。

提升推理能力：GRPO鼓励模型探索不同的解题思路，并通过比较来学习，从而提升模型的推理能力，使其能够更好地解决复杂问题。 总而言之，GRPO就像一个**“群组讨论”机制**，让模型通过比较和学习，不断提升解决问题的能力。这种方法降低了计算成本，稳定了训练过程，并有效提升了模型的推理能力。

PPO算法详解：原理与工作流程解析



PPO（Proximal Policy Optimization，近端策略优化）是一种在强化学习（RL）中广泛使用的优化算法，尤其在像RLHF（Reinforcement Learning from Human Feedback，基于人类反馈的强化学习）这样的场景中，PPO因其高效性和稳定性而被大量采用。以下是对PPO工作原理的详细解释：

1. PPO的背景与目标

PPO是一种策略梯度方法（Policy Gradient Method），其核心目标是优化策略网络（Policy Network），使得智能体在与环境交互时能够获得更高的累积奖励。PPO的特点是通过限制新策略和旧策略之间的差异，避免策略更新过大，从而实现更稳定的训练。

在RLHF中，PPO通常用于优化大模型的行为，使其符合人类的偏好。通过人类反馈数据（如排序、评分等），PPO能够调整模型的输出策略，使其更符合人类的期望。

2. PPO的核心思想

PPO的核心思想是通过限制新策略（Policy）和旧策略（Old Policy）之间的差异，确保策略更新是渐进式的。这种限制可以通过两种方式实现：

> **裁剪（Clipping）**：限制策略分布的比值在一个固定范围内。
>
> **KL散度约束（KL Divergence Constraint）**：限制新旧策略之间的KL散度不超过某个阈值。

PPO的目标是最大化一个目标函数，同时确保策略更新不会过于激进。

3. PPO的目标函数

PPO的目标函数通常有两种形式：**裁剪形式**和**KL约束形式**。以下是裁剪形式的目标函数：

$$ L^{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \cdot A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \cdot A_t \right) \right] $$

关键部分解释

$r_t(\theta)$：新旧策略的概率比值，表示在状态 s_t 下，动作 a_t 在新策略 π_θ 和旧策略 $\pi_{\theta_{\text{old}}}$ 下的概率比值： $$ r_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)}

$$

A_t：优势函数（Advantage Function），表示在状态 s_t 下采取动作 a_t 相对于平均动作的价值增益。

$\text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)$：对概率比值进行裁剪，限制其范围在 $[1 - \epsilon, 1 + \epsilon]$，其中 ϵ 是一个超参数（通常设为0.1或0.2）。这种裁剪机制确保新策略不会偏离旧策略太远。

$\mathbb{E}_t$：对时间步 t 的期望。

通过这种裁剪机制，PPO确保了策略更新的幅度是可控的，从而避免了大范围的策略震荡。

4. PPO的工作流程

PPO的训练过程通常包括以下几个步骤：

(1) 收集数据

使用当前策略（旧策略）与环境交互，收集轨迹数据（状态、动作、奖励等）。

在RLHF中，这些数据可能还包括人类反馈的偏好信息。

(2) 计算优势函数

使用收集到的数据计算优势函数 A_t，通常通过广义优势估计（Generalized Advantage Estimation, GAE）来计算。

(3) 优化策略

使用PPO的目标函数（如裁剪形式）优化策略网络参数 θ。

通过梯度上升方法最大化目标函数，同时确保新策略和旧策略之间的差异不会过大。

(4) 重复迭代

重复上述步骤，直到策略收敛或达到预定的训练目标。

5. PPO的优点

稳定性：通过限制新旧策略的差异，PPO避免了传统策略梯度方法中常见的策略震荡问题。

高效性：PPO不需要额外的超参数（如熵正则化系数），训练过程更加高效。

灵活性：PPO可以与其他技术（如GAE、值函数优化等）结合使用，进一步提升性能。

> *上面写到"PPO不需要额外的超参数（如熵正则化系数），训练过程更加高效"。实际上，在实际使用时，往往仍然会设置一些常见的超参数，例如：*
>
> *裁剪系数ϵ（通常在0.1或0.2左右）；*
>
> *学习率(decoder)；*
>
> *若需要保持策略多样性，还会引入熵系数(entropy bonus coefficient)等。 因此，说"PPO不需要额外的超参数"容易让人产生误解：PPO仍有一系列需要调节的超参数，并不是"零超参数"或"仅仅一个裁剪范围"这么简单。只是在相对于像TRPO（Trust Region Policy Optimization）这种需要复杂约束优化的算法，PPO减少了一些实现上的复杂度和需要额外调参的环节，因此通常被认为更"易用"或"高效"。*

6. PPO在RLHF中的应用

在RLHF中，PPO的主要任务是优化模型的输出策略，使其更符合人类的偏好。具体流程包括：

数据收集：通过人类反馈（如排序、评分）生成奖励信号。

奖励建模：将人类反馈转化为奖励函数。

策略优化：使用PPO优化模型参数，使模型输出更符合奖励函数的期望。

通过这种方式，PPO能够帮助大模型逐步学习到符合人类偏好的行为模式。

总结

PPO是一种高效且稳定的策略优化算法，其核心思想是通过限制新旧策略的差异，确保策略更新是渐进式的。PPO在RLHF中被广泛应用，能够有效地优化大模型的行为，使其更符合人类的期望。

分享这篇文章