

# Rewarding Progress: Scaling Automated Process Verifiers for LLM Reasoning

Amrith Setlur<sup>1,3,\*</sup>, Chirag Nagpal<sup>1,\*</sup>, Adam Fisch<sup>2</sup>, Xinyang Geng<sup>2</sup>, Jacob Eisenstein<sup>2</sup>, Rishabh Agarwal<sup>2</sup>, Alekh Agarwal<sup>1</sup>, Jonathan Berant<sup>†,2</sup> and Aviral Kumar<sup>†,2,3</sup>

<sup>1</sup>Google Research, <sup>2</sup>Google DeepMind, <sup>3</sup>Carnegie Mellon University, \*Equal contribution, †Equal advising

A promising approach for improving reasoning in large language models is to use process reward models (PRMs). PRMs provide feedback at each step of a multi-step reasoning trace, potentially improving credit assignment over outcome reward models (ORMs) that only provide feedback at the final step. However, collecting dense, per-step human labels is not scalable, and training PRMs from automatically-labeled data has thus far led to limited gains. To improve a *base* policy by running search against a PRM or using it as dense rewards for reinforcement learning (RL), we ask: “How should we design process rewards?”. Our key insight is that, to be effective, the process reward for a step should measure *progress*: a change in the likelihood of producing a correct response in the future, before and after taking the step, corresponding to the notion of step-level advantages in RL. Crucially, this progress should be measured under a *prover* policy distinct from the base policy. We theoretically characterize the set of good provers and our results show that optimizing process rewards from such provers improves exploration during test-time search and online RL. In fact, our characterization shows that weak prover policies can substantially improve a stronger base policy, which we also observe empirically. We validate our claims by training *process advantage verifiers* (PAVs) to predict progress under such provers, and show that compared to ORMs, test-time search against PAVs is  $> 8\%$  more accurate, and  $1.5 - 5\times$  more compute-efficient. Online RL with dense rewards from PAVs enables *one of the first results* with  $5 - 6\times$  gain in sample efficiency, and  $> 6\%$  gain in accuracy, over ORMs.

## 1. Introduction

Trained reward models or *verifiers* are often used to improve math reasoning in large language models, either by re-ranking solutions at test-time (Collins, 2000) or via reinforcement learning (RL) (Uesato et al., 2022). Typically, verifiers are trained to predict the outcome of an entire reasoning trace, often referred to as *outcome* reward models (ORM) (Cobbe et al., 2021b; Hosseini et al., 2024). However, ORMs only provide a sparse signal of correctness, which can be hard to learn from and inefficient to search against. This challenge is alleviated by fine-grained supervision, in theory. For reasoning, prior works train *process* reward models (PRMs) that assign intermediate rewards after each step of search (Snell et al., 2024) or during RL. While Lightman et al. (2023) obtains PRM annotations from human raters, this approach is not scalable. More recent works (Luo et al., 2024; Wang et al., 2024) train PRMs to predict automatically-generated annotations that estimate future success of solving the problem, akin to value functions in RL. So far, automated PRMs, especially as dense rewards in RL, only improve by 1-2% over ORMs (Shao et al., 2024), raising serious doubts over their utility.

To resolve these uncertainties, in this paper, we train PRMs with automated annotations, such that optimizing the dense rewards from trained PRMs can improve a *base* policy compute- and sample-efficiently, during test-time search and online RL. For this, we first ask: (i) what should the *per-step* process rewards measure, and (ii) what kind of automated data collection strategy should we use to train PRMs that predict this measure. For (i), conventional belief (Lightman et al., 2023; Uesato et al., 2022)

# 奖励进步：扩展自动化过程验证器以用于大语言模型推理

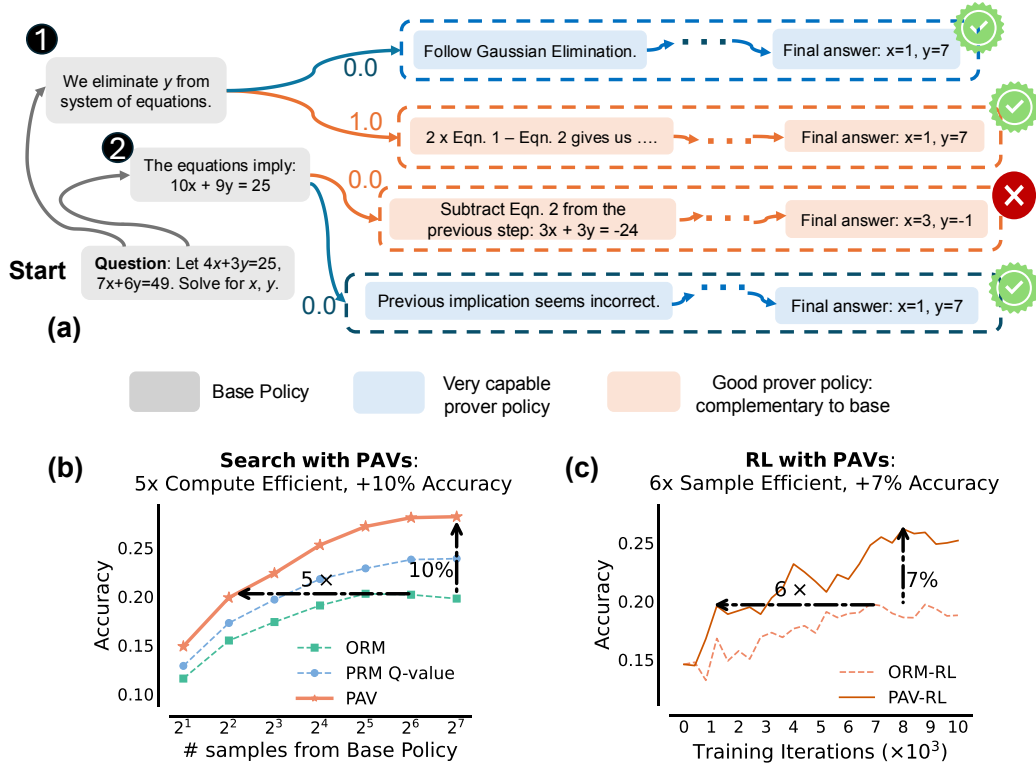
Amrith Setlur<sup>1,3,\*</sup>, Chirag Nagpal<sup>1,\*</sup>, Adam Fisch<sup>2</sup>, Xinyang Geng<sup>2</sup>, Jacob Eisenstein<sup>2</sup>, Rishabh Agarwal<sup>2</sup>, Alekh Agarwal<sup>1</sup>, Jonathan Berant<sup>†,2</sup>, Aviral Kumar<sup>†,2,3</sup> <sup>1</sup>谷歌研究, <sup>2</sup>谷歌深度思维, <sup>3</sup>卡内基梅隆大学, \*同等贡献, †同等指导

一种有前途的方法是使用过程奖励模型（PRMs）来提高大型语言模型的推理能力。PRMs在多步推理跟踪的每一步提供反馈，这可能比只在最终步骤提供反馈的结果奖励模型（ORMs）更好地分配信用。然而，收集密集的每步人工标签是不可扩展的，因此从自动标记的数据训练PRMs迄今仅带来了有限的收益。为了通过运行搜索来改进一个`base`策略，或者使用PRM作为强化学习（RL）的密集奖励，我们提出一个问题：“我们应该如何设计过程奖励？”我们的关键洞察是，为了有效，步骤的过程奖励应该衡量`progress`：采取步骤前后产生正确响应概率的变化，对应于RL中的步骤级优势的概念。至关重要的是，这种进步应该在与基础策略不同的`prover`策略下进行测量。我们从理论上界定了好的证明者集，我们的结果表明，从这样的证明者优化过程奖励可以提高测试时搜索和在线RL中的探索。事实上，我们的表征表明，弱证明者策略可以显著提高更强的基础策略，这也是我们在实验中观察到的。我们通过训练`process advantage verifiers (PAVs)`来预测在这些证明者下的进步，证明了与ORMs相比，基于PAVs的测试时搜索准确率提高了8%，计算效率提高了1>5倍。基于PAVs的密集奖励使在线RL在样本效率上提高了5.6倍，准确率提高了6%，并且PAVs使在线RL在样本效率上提高了-6%。

## 1. 介绍

训练奖励模型或`verifiers`常被用于提高大型语言模型的数学推理能力，要么在测试时重新排序解决方案（Collins, 2000），要么通过强化学习（RL）（Uesato等, 2022）。通常，验证器被训练来预测整个推理过程的结果，通常称为`outcome`奖励模型（ORM）（Cobbe等, 2021b; Hosseini等, 2024）。然而，ORMs只能提供稀疏的正确性信号，这很难学习并且搜索效率低下。通过精细监督，这一挑战可以得到缓解。对于推理，先前的工作训练了`process`奖励模型（PRMs），在搜索的每一步后分配中间奖励（Snell等, 2024）或在RL期间。Lightman等（2023）从人类评分者那里获得了PRM注释，但这种方法不具有可扩展性。更近期的工作（Luo等, 2024; Wang等, 2024）训练PRMs来预测自动生成的注释，这些注释估计解决问题的未来成功率，类似于RL中的价值函数。到目前为止，自动化的PRMs，特别是在RL中作为密集奖励，仅在ORMs上提高了1-2%（Shao等, 2024），这对其实用性提出了严重质疑。

为了解决这些不确定性，在本文中，我们使用自动化注释训练PRMs，从而使优化由训练好的PRMs产生的密集奖励能够高效地提升测试时搜索和在线RL中的`base`策略。为此，我们首先问：(i) `per-step`过程奖励应该衡量什么，以及(ii) 我们应该使用什么样的自动化数据收集策略来训练能够预测这种衡量方式的PRMs。对于(i)，传统的信念（Lightman等, 2023; Uesato等, 2022）



**Figure 1 | Process advantage verifiers (PAV):** Process reward for a step is defined as progress (advantage) under the prover policy, *i.e.*, change in prover policy’s success rate before and after the step. (a): The base policy samples both correct ① and incorrect ② steps but struggles to succeed from either. A strong prover policy completes the solution from both steps, and is unable to adequately reflect progress made by ① and ② (both scored 0.0). Conversely, a complementary prover policy distinguishes ①, ② more prominently (only succeeds from ①). (b,c): Compared to ORMs, PAVs are 5x more compute efficient, 10% more accurate in test-time search, and 6x more sample efficient, 7% more accurate for online reinforcement learning (RL).

has been to measure mathematical correctness or relevance of steps. But, it is unclear if this supervision yields the most improvement in the base policy (*e.g.*, a policy may need to generate simpler, repetitive, and even incorrect steps to explore and discover the final answer during test-time search and RL). **Our key insight** is that per-step, process rewards that measure a notion of *progress*: change in the likelihood of arriving at a correct final answer before and after taking the step, are effective, for both test-time beam search and online RL. Reinforcing steps that make progress regardless of whether they appear in a correct or incorrect trace diversifies the *exploration* of possible answers at initial steps, which is crucial when the approach to solve a problem is not clear. Formally, such rewards correspond to per-step *advantages* of steps from the RL literature (Sutton and Barto, 2018). We empirically show that using advantages in addition to ORM rewards outperforms the typical use of future probabilities of success or Q-values (Wang et al., 2024) for both search and RL. This is because, when given a combinatorial space of responses, under bounded computational and sampling constraints, Q-values mainly “exploit” states whereas advantages also “explore” steps that make the most progress towards the final answer (Fig. 2).

To answer (ii), we first note that advantages under a poor base policy are  $\approx 0$  on most steps, and thus will not be informative for search or RL. In addition, regardless of the strength of the base policy, using its own per-step advantages as process rewards in RL will result in base policy updates equivalent to *only*

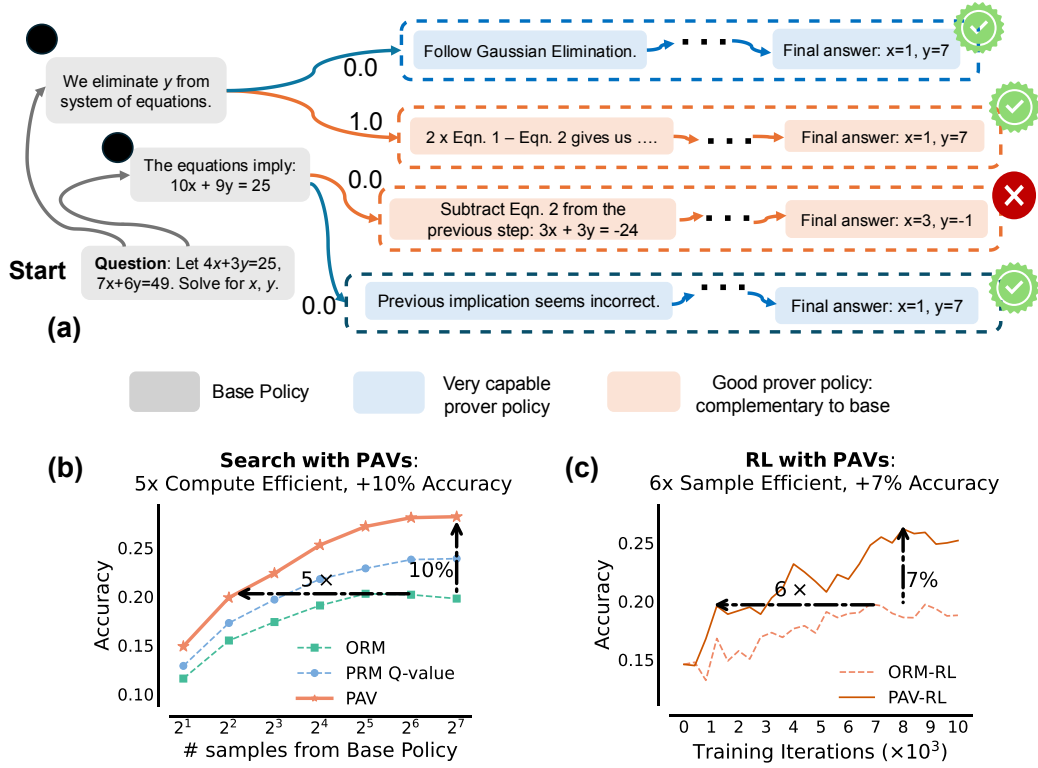


Figure 1 | **Process advantage verifiers (PAV):** 过程奖励对于一个步骤而言，定义为证明者策略下的进步（优势），i.e.，即证明者策略成功概率在该步骤前后的变化。(a)：基线策略会采样正确的步骤1和错误的步骤2，但难以从任何一个步骤中成功。强大的证明者策略能够从两个步骤中完成解决方案，但无法充分反映步骤1和步骤2所取得的进步（两者得分均为0.0）。相反，互补的证明者策略更明显地区分了步骤1和步骤2（仅从步骤1中成功）。(b,c)：与ORMs相比，PAVs在计算效率上提高了5倍，在测试时搜索的准确性上提高了10%，在样本效率上提高了6倍，在在线强化学习（RL）的准确性上提高了7%。

has been 用于衡量数学正确性或步骤的相关性。但不清楚这种监督是否能带来基线策略(e.g.)的最大改进（一个策略可能需要生成更简单、重复甚至错误的步骤来探索和发现最终答案，在测试时的搜索和RL过程中）。我们的关键洞察是，每一步的过程奖励，衡量的是一个概念*progress*：采取该步骤前后到达正确最终答案概率的变化，对于测试时的束搜索和在线RL都是有效的。强化那些无论是否出现在正确或错误的轨迹中都能取得进展的步骤，可以增加初始步骤中可能答案的多样性，这对于解决问题的方法不明确时至关重要。形式上，这样的奖励对应于来自RL文献中的每一步*advantages* (Sutton和Barto, 2018)。我们通过实验表明，使用优势奖励加上ORM奖励比仅使用未来成功概率或Q-值 (Wang等, 2024) 在搜索和RL中表现更好。这是因为，在给定组合响应空间的情况下，受制于计算和采样的限制，Q-值主要“利用”状态，而优势也“探索”那些最接近最终答案的步骤 (图2)。

为了回答(ii)，我们首先注意到，在一个较差的基础策略下，优势在大多数步骤上是  $\approx 0$ ，因此对于搜索或RL来说不具备信息性。此外，无论基础策略的强度如何，使用其自身的每步优势作为RL中的过程奖励将会导致基础策略更新等同于 *only*

using outcome rewards for RL (since a standard policy gradient algorithm already computes advantages). Hence, we propose to use advantages estimated via rollouts under a different **prover policy** as process rewards (Fig. 1(a)). How should we choose this prover policy? A natural guess would be to use a very capable prover. However, we show advantages under an overly capable prover policy, that can succeed from any step, fail to distinguish good and bad steps. A similar argument holds for very weak provers.

In theory, we formalize this intuition to define good provers as policies that are *complementary* to the base policy (i.e., policies with advantages that can contrast steps produced by the base policy sufficiently), while still producing step-level advantages correlated with those of the base policy. For e.g., for Best-of- $K$  policies (Nakano et al., 2021) corresponding to a base policy, we empirically find that provers corresponding to  $K > 1$  (but not too large) are more capable at improving the base policy. Contrary to intuition, the set of complementary provers also contains policies that are worse than the base policy. To predict the advantages of such provers we train dense verifiers, called **process advantage verifiers (PAVs)**, that accelerate sample and compute efficiency of RL and search.

With the conceptual design of PAVs in place, we prescribe practical workflows for training PAVs and demonstrate their efficacy on a series of 2B, 9B, and 27B Gemma2 models (Gemma Team et al., 2024). PAV training data is gathered by sampling “seed” solution traces from the prover and partial rollouts from the same to estimate the  $Q$ -value at each prefix of the seed trace. Our workflow prescribes favorable ratios for seed and partial rollouts. Our first set of empirical results show that for an equal budget on test-time compute, beam search against trained PAVs is  $>8\%$  better in accuracy, and **1.5 – 5 $\times$**  more compute efficient compared to re-ranking complete traces against an ORM (Fig. 1(b)). Dense rewards from PAVs improve the efficiency of step-level exploration during search by pruning the combinatorial space of solutions aggressively and honing in on a diverse set of possible sequences. Finally, we demonstrate **for the first time**, that using PAVs as dense rewards in RL scales up data efficiency by **6 $\times$**  compared to only using outcome rewards (Fig. 1(c)). Moreover, base policies trained with PAVs also achieve **8 $\times$**  better Pass @ $N$  performance (probability of sampling the correct solution in  $N$  attempts), and consequently afford a higher ceiling on the performance of any test-time re-ranker. Finally, running RL with PAVs discovers solutions to hard problems that sampling from the SFT policy with a very large budget can’t solve.

## 2. Preliminaries, Definitions, and Notation

Following protocols from Lightman et al. (2023); Uesato et al. (2022), a reasoning trace from an LLM consists of multiple logical steps separated by a demarcation token. An outcome reward model (ORM) is a trained verifier that assigns a numerical score after the last step of the trace, and a process reward model (PRM) is a trained verifier that scores each step of the trace individually.

**Problem setup and notation.** Given a math problem  $x \in \mathcal{X}$ , our goal is to improve a *base policy*  $\pi$  that samples a response  $y \sim \pi(\cdot | x)$  in the set  $\mathcal{Y}$ . A response  $y$  consists of multiple reasoning steps (maximum  $H$ ), separated by a delimiter (‘next line’ in our case), i.e.,  $y = (a_1, a_2, \dots, a_H)$ . Since sampling is auto-regressive, we can view each step as an action taken by the agent  $\pi$  in a Markov decision process (MDP) with deterministic dynamics. Specifically, we treat the prefix  $(x, a_1, \dots, a_{h-1})$  as the current *state*  $s_h$  and next step  $a_h \sim \pi(\cdot | x)$  as the *action* taken by  $\pi$  at  $s_h$ , resulting in the next state  $s_{h+1}$ . For problem  $x$ , with ground-truth response  $y_x^*$ , we can evaluate the accuracy of  $\pi$  by running a regular expression match on the final answer (Hendrycks et al., 2021):  $\text{Rex}(y, y_x^*) \mapsto \{0, 1\}$ , i.e., accuracy is given by  $\mathbb{E}_{y \sim \pi(\cdot | x)} [\text{Rex}(y, y_x^*)]$ . Now, given a dataset  $\mathcal{D} = \{(x_i, y_{x_i}^*)\}_i$  of problem-solution pairs, the main goal is to learn a good base policy by optimizing this outcome reward on  $\mathcal{D}$ . Next, we see how we can leverage the final answer verifier Rex available on  $\mathcal{D}$  to train ORMs and PRMs.



使用结果奖励进行RL（因为标准的策略梯度算法已经计算了优势）。因此，我们建议使用通过不同策略 *prover policy* 的 rollout 估计的优势作为过程奖励（图1(a)）。我们应该如何选择这个证明策略？一个自然的想法是使用一个非常有能力的证明者。然而，我们展示了在过于有能力的证明策略下，该策略可以从任何步骤成功，但无法区分好的步骤和坏的步骤。类似地，对于非常弱的证明者也存在相同论点。

在理论上，我们将这种直觉形式化，定义好的证明者为策略，这些策略相对于基础策略是 *complementary* 的（i.e., 具有能够与基础策略生成的步骤形成足够对比的优势的策略），同时仍然在步骤级别上产生与基础策略相关的优势。对于 e.g., 对于对应于基础策略的 Best-of- $K$  策略（Nakano 等, 2021），我们发现，对应于  $K > 1$ （但不是太大）的证明者在提高基础策略方面更具能力。与直觉相反，互补的证明者集合中也包含比基础策略更差的策略。为了预测这些证明者的优点，我们训练密集型验证器，称为 *process advantage verifiers (PAVs)*，这些验证器可以加速RL和搜索的样本和计算效率。

有了 PAVs 的概念设计，我们为训练 PAVs 制定了实用的工作流程，并在一系列 2B、9B 和 27B Gemma2 模型（Gemma Team et al., 2024）上展示了它们的有效性。PAVs 的训练数据通过从证明器中采样“种子”解决方案轨迹和相同的部分展开来收集，以估计每个种子轨迹前缀的 Q-值。我们的工作流程规定了种子和部分展开的有利比例。我们的第一组实证结果表明，在测试时计算预算相同的情况下，与训练好的 PAVs 进行束搜索比重新排名完整轨迹对 ORM 进行重新排名在准确性上高出  $>8\%$ ，并且在计算效率上高出  $1.5\times$ （Fig. 1(b)）。来自 PAVs 的密集奖励通过激进地修剪解决方案的组合空间并在搜索过程中聚焦于一系列可能的序列，从而提高了步骤级探索的效率。最后，我们展示了使用 PAVs 作为密集奖励在 RL 中可以将数据效率提高  $6\times$ （Fig. 1(c)）。此外，使用 PAVs 训练的基础策略在 Pass @ $N$  性能上也提高了  $8\times$ （在  $N$  次尝试中采样正确解的概率），从而为任何测试时的重新排名提供了更高的性能上限。最后，使用 PAVs 进行 RL 可以发现 SFT 策略即使在非常大的预算下也无法解决的难题。

## 2. 前提、定义和符号

遵循 Lightman 等（2023）和 Uesato 等（2022）的协议；一个 LLM 的推理轨迹由多个逻辑步骤组成，这些步骤由分隔标记分隔。结果奖励模型（ORM）是一个经过训练的验证器，在轨迹的最后一步之后分配一个数值分数，而过程奖励模型（PRM）是一个经过训练的验证器，单独为轨迹中的每一步打分。

问题设置和符号表示。给定一个数学问题  $\mathbf{x} \in \mathcal{X}$ ，我们的目标是改进一个 *base policy*  $\pi$ ，该模型从集合  $\mathcal{Y}$  中采样一个响应  $\mathbf{y} \sim \pi(\cdot | \mathbf{x})$ 。一个响应  $\mathbf{y}$  由多个推理步骤（最大  $H$  步）组成，这些步骤由分隔符（在我们的情况下是“换行符”）分隔，i.e.,  $\mathbf{y} = (a_1, a_2, \dots, a_H)$ 。由于采样是自回归的，我们可以将每个步骤视为代理  $\pi$  在具有确定性动力学的马尔可夫决策过程 (MDP) 中采取的一个动作。具体来说，我们把前缀  $(\mathbf{x}, a_1, \dots, a_{h-1})$  看作当前 *state*  $\mathbf{s}_h$  和下一步  $a_h \sim \pi(\cdot | \mathbf{x})$  看作  $\pi$  在  $\mathbf{s}_h$  采取的动作，从而导致下一个状态  $\mathbf{s}_{h+1}$ 。对于问题  $\mathbf{x}$ ，带有真实响应  $\mathbf{y}_x^*$ ，我们可以通过在最终答案上运行正则表达式匹配来评估  $\pi$  的准确性（Hendrycks 等人, 2021）： $\text{Rex}(\mathbf{y}, \mathbf{y}_x^*) \mapsto \{0, 1\}$ ，i.e., 准确性由  $\mathbb{E}_{\mathbf{y} \sim \pi(\cdot | \mathbf{x})} [\text{Rex}(\mathbf{y}, \mathbf{y}_x^*)]$  给出。现在，给定一个包含问题-解决方案对的数据集  $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_{x_i}^*)\}_i$ ，主要目标是通过在  $\mathcal{D}$  上优化此结果奖励来学习一个好的基础策略。接下来，我们看看如何利用在  $\mathcal{D}$  上可用的最终答案验证器 Rex 来训练 ORM 和 PRMs。

**Outcome reward model (ORM).** Given a response  $y$ , an ORM estimates the ground-truth correctness  $\text{Rex}(y, y_x^*)$ . To train such a model we first take problems in  $\mathcal{D}$ , and collect training data of the form  $\{(x, y \sim \pi(\cdot | x), \text{Rex}(y, y_x^*))\}$ . Then we train an ORM that takes as input a problem-response pair  $(x, y)$  and predicts  $\text{Rex}(y, y_x^*)$ . At test time, when  $y_x^*$  is unknown, the ORM is used to score candidate solutions revealed by test-time search. Given a base policy  $\pi$ , a Best-of- $K$  policy:  $\text{BoK}(\pi)$ , is a policy that samples  $K$  responses from  $\pi$ , scores them against an ORM, and returns the one with the highest score. Whenever the ORM matches  $\text{Rex}$ , the performance of  $\text{BoK}(\pi)$  is referred to as  $\text{Pass @}K$ . Furthermore, when the likelihood of  $\pi$  solving problem  $x$  is  $p_x$ , then for  $\text{BoK}(\pi)$  this likelihood is given by the expression:  $1 - (1 - p_x)^K$ . In general, this is larger than  $p_x$ , making  $\text{BoK}(\pi)$  stronger than  $\pi$  for  $K > 1$ .

**Standard process reward models (PRMs).** A PRM scores every step  $a_h$  in a multi-step response  $y \sim \pi$  (e.g., in [Lightman et al. \(2023\)](#) PRMs are trained to score correct steps over incorrect and irrelevant ones). But, unlike ORMs, which only require  $\text{Rex}$  for data collection, PRM training data requires expensive step-level human annotations. Prior works ([Luo et al., 2024](#); [Wang et al., 2024](#)) attempted to scale process rewards automatically by sampling from the model to provide a heuristic understanding of when a step is actually correct. In particular, they evaluate a prefix by computing the expected future accuracy of multiple completions sampled from  $\pi$ , after conditioning on the prefix, i.e., value function  $Q^\pi$  (Eq. 1) from RL. Similarly, we define  $V^\pi(s_h) := \mathbb{E}_{a_h \sim \pi(\cdot | s_h)} Q^\pi(s_h, a_h)$  as value of state  $s_h$ . These works use  $Q^\pi$  as the PRM that assigns a score of  $Q^\pi(s_h, a_h)$  to the action  $a_h$ , at state  $s_h$ .

$$Q^\pi(\underbrace{(x, a_1, \dots, a_{h-1})}_{\text{state } s_h}, \underbrace{a_h}_{\text{action } a_h}) = \underbrace{\mathbb{E}_{a_{h+1}, \dots, a_H \sim \pi(\cdot | s_h, a_h)} \left[ \text{Rex}((a_1, \dots, a_H), y_x^*) \right]}_{\text{likelihood of future success}}, \quad (1)$$

**Using PRMs for beam search at test-time.** Given a PRM, a natural way to spend test-time compute is to use it as a step-level re-ranker within a beam search procedure ([Snell et al., 2024](#)). For each problem, at step 0, a beam of maximum width  $B$ , is initialized with a single state consisting of just the problem. At step  $h$ , a beam contains partial responses unrolled till a set of states or prefixes  $\{s_i\}_{i=1}^B$ . From each state  $s_i$  in this set,  $C$  independent actions or steps  $\{a_{i,j}\}_{j=1}^C$  are sampled from  $\pi(\cdot | s_i)$ , each of which leads to a new state. Process rewards from PRMs assign a score to every new state  $(s_i, a_{i,j})$ , and only the states corresponding to the top  $B$  values are retained in the beam for the next step.

### 3. How Should we Define Process Rewards and Why?

Ultimately, we are interested in test-time search and RL methods that can most efficiently and reliably discover solution traces with the correct final answer, thus maximizing  $\text{Rex}$ . To this end, *process rewards should serve as step-level supervision to indirectly maximize outcome-level  $\text{Rex}$* . Our position contrasts with conventional belief that process rewards should mainly evaluate mathematical correctness or relevance of individual steps ([Lightman et al., 2023](#); [Uesato et al., 2022](#)), since LLMs might need to generate trivial or repetitive intermediate steps in order to discover a trace with the correct final answer. With this insight, in this section we approach the design of dense automated step rewards as a form of supervision to be used in conjunction with sparse outcome rewards to improve the base policy.

In an MDP, a starting point to design step-level dense feedback that is eventually meant to optimize a sparse outcome reward  $\text{Rex}$  is to consider the notion of a *potential function* ([Ng et al., 1999](#)): in our case, this is a function that summarizes the difference between some statistic of the policy at the future state and the same statistic computed at the current state. By appealing to this framework, in Sec. 3.1, we

Outcome 奖励模型 (ORM)。给定一个响应  $y$ ，一个 ORM 估计其真实正确性  $\text{Rex}(y, y_x^*)$ 。为了训练这样的模型，我们首先从  $\mathcal{D}$  中选取问题，并收集形式为  $\{(x, y \sim \pi(\cdot | x), \text{Rex}(y, y_x^*))\}$  的训练数据。然后我们训练一个 ORM，该模型以问题-响应对  $(x, y)$  作为输入并预测  $\text{Rex}(y, y_x^*)$ 。在测试时，当  $y_x^*$  未知时，使用 ORM 对测试时搜索揭示的候选解决方案进行评分。给定一个基础策略  $\pi$ ，一个 Best-of- $K$  策略： $\text{BoK}(\pi)$ ，是从  $\pi$  中采样  $K$  个响应，使用 ORM 评分，并返回得分最高的一个。每当 ORM 匹配  $\text{Rex}$  时， $\text{BoK}(\pi)$  的性能被称为 Pass @ $K$ 。此外，当  $\pi$  解决问题  $x$  的概率为  $p_x$  时，对于  $\text{BoK}(\pi)$ ，该概率由表达式  $1 - (1 - p_x)^K$  给出。一般来说，这大于  $p_x$ ，使得  $\text{BoK}(\pi)$  比  $\pi$  更强，对于  $K > 1$  也是如此。

标准过程奖励模型 (PRMs)。一个 PRM 为一个多步响应中的每一步骤  $a_h$  进行评分  $y \sim \pi$  (e.g. 在 Lightman 等 (2023)) 中，PRMs 被训练为对正确的步骤进行评分，而不是错误的和无关的步骤。但是，与仅需要  $\text{Rex}$  进行数据收集的 ORM 不同，PRM 的训练数据需要昂贵的步骤级别的人工注释。先前的工作 (Luo 等, 2024; Wang 等, 2024) 尝试通过从模型中采样来自动扩展过程奖励，以提供一个启发式理解，即在什么情况下一个步骤实际上是正确的。特别是，他们通过计算在前缀  $\pi$  条件下从 i.e. 中采样出的多个完成的预期未来准确性来评估一个前缀， $Q^\pi$  (Eq. 1) 来自 RL。类似地，我们将  $V^\pi(s_h) := \mathbb{E}_{a_h \sim \pi(\cdot | s_h)} Q^\pi(s_h, a_h)$  定义为状态  $s_h$  的价值。这些工作使用  $Q^\pi$  作为 PRM，它为状态  $s_h$  中的动作  $a_h$  分配一个分数  $Q^\pi(s_h, a_h)$ 。

$$Q^\pi(\underbrace{(x, a_1, \dots, a_{h-1})}_{\text{state } s_h}, \underbrace{a_h}_{\text{action } a_h}) = \underbrace{\mathbb{E}_{a_{h+1}, \dots, a_H \sim \pi(\cdot | s_h, a_h)}}_{\text{likelihood of future success}} \left[ \text{Rex}((a_1, \dots, a_H), y_x^*) \right], \quad (1)$$

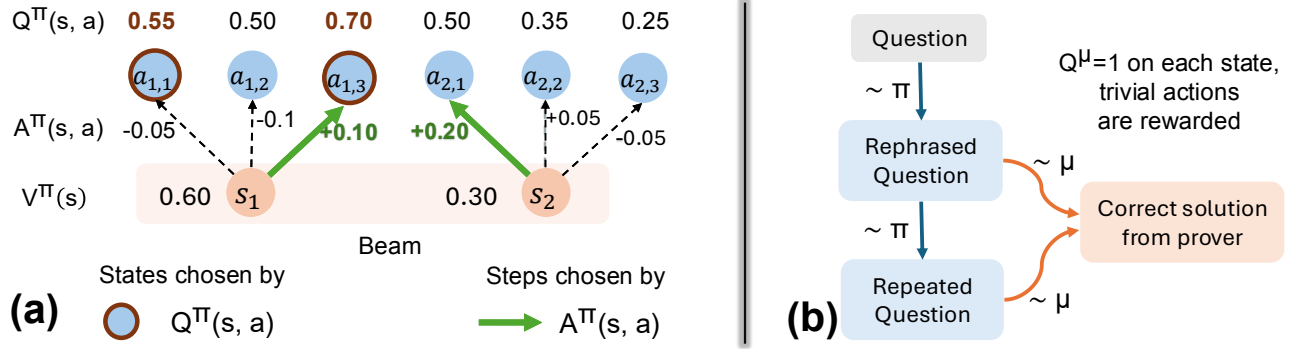
使用 PRMs 在测试时进行束搜索。给定一个 PRM，一种自然的测试时计算方式是将其用作束搜索过程中的步骤级重排序器 (Snell 等, 2024)。对于每个问题，在步骤 0，用宽度为  $B$  的束初始化，包含仅包含问题的一个状态。在步骤  $h$ ，束包含展开到一组状态或前缀  $\{s_i\}_{i=1}^B$  的部分响应。从这组中的每个状态  $s_i$ ，采样  $C$  个独立的动作或步骤  $\{a_{i,j}\}_{j=1}^C$ ，每个动作都导致一个新的状态。从 PRMs 计算奖励为每个新状态分配一个分数  $(s_i, a_{i,j})$ ，只有对应于前  $B$  个最高值的状态保留在束中以进行下一步。

### 3. 我们应该如何定义过程奖励以及为什么？

最终，我们关注的是可以在测试时最高效和可靠地发现具有正确最终答案的解题路径的搜索和 RL 方法，从而最大化  $\text{Rex}$ 。为此，*process rewards should serve as step-level supervision to indirectly maximize outcome-level Rex*。我们的观点与传统的信念相悖，即过程奖励主要应评估单个步骤的数学正确性或相关性 (Lightman 等, 2023; Uesato 等, 2022)，因为 LLMs 可能需要生成一些平凡或重复的中间步骤，以便发现具有正确最终答案的解题路径。基于这一洞察，在本节中，我们通过与稀疏结果奖励结合使用来改进基础策略，将密集自动步骤奖励视为一种监督形式。

在 MDP 中，设计一个逐步密集反馈的方法，最终目的是优化稀疏结果奖励  $\text{Rex}$ ，可以考虑 Ng 等人 (1999) 提出的 *notion potential function* (：在我们的情况下，这是一个函数，总结了未来状态中某些策略统计量与当前状态中相同统计量之间的差异。通过利用这一框架，在第 3.1 节中我们





**Figure 2 | Issues with using  $Q$ -values as process rewards:** (a): Unlike  $A^\pi$ ,  $Q^\pi$  mixes action evaluation with the  $Q$ -value of the previous state. Beam search with  $Q^\pi$  exploits high-likelihood states, while adding  $A^\pi$  (e.g.,  $Q^\pi + \alpha A^\pi$  in Eq. 5) aids in exploring states reached by making actions that induce progress, i.e., increase likelihood of success. (b):  $Q^\mu$  from a strong prover  $\mu$  can assign unmerited bonuses to trivial actions.

show that advantages – not value functions (Luo et al., 2024; Wang et al., 2024) – that measure a notion of “progress” at each new step are more appropriate for use as dense rewards in search and RL (primarily for exploration). Then in Secs. 3.3 and 3.4, we show that this progress or advantage value is measured best under a policy  $\mu$ , different from the base policy  $\pi$ . We call this policy  $\mu$ , the **prover policy**.

### 3.1. Process Rewards Should be Advantages, Not Value Functions

To understand the relationship to potential functions, we first study test-time beam search, and present some challenges with the reward design of Snell et al. (2024), that uses value function  $Q^\pi(s, a)$  of the base policy  $\pi$  to reward action  $a$  at state  $s$ . Consider the example in Fig. 2(a), where from the 2 states in the beam, we sample 3 actions. If we pick next states purely based on highest values of  $Q^\pi$ , we would be comparing steps sampled from different states (e.g.,  $a_{1,1}$  vs.  $a_{2,1}$ ) against each other. Clearly, a reduction in expected final outcome, i.e.,  $Q^\pi(s_1, a_{1,1}) - V^\pi(s_1)$ , means that  $a_{1,1}$  by itself has a negative effect of  $-0.05$  on the probability of success from  $s_1$ , whereas  $a_{2,1}$  has a positive effect of  $+0.20$  from  $s_2$ . However, expanding the beam based on *absolute* values of  $Q^\pi$  retains the action that makes negative progress, and removes state  $s_2$  from the beam (as beam size is 2). In other words,  $Q^\pi$  fails to decouple the “evaluation” of an action (step), from the “promise” shown by the previous state. This will not be an issue for every problem, and particularly not when the beam capacity is unbounded, but under finite computational and sampling constraints, using  $Q^\pi$  might retain states with potentially unfavorable steps that hurt the overall likelihood of success. **If we could also also utilize the progress made by the previous step** along with the likelihood of success  $Q^\pi$  when deciding what to retain in the beam, then we can address this tradeoff.

**How can we measure the “progress” made by a step?** One approach is to consider the relative increase/decrease in the likelihood of success, before and after the step. This notion is formalized by the advantage (Eq. 2) of a step under policy  $\pi$ . Furthermore, since advantages can attach either positive or negative values to a step, training the base policy against advantages supervises the base policy when it generates a step that makes progress (where  $A^\pi > 0$ ), and also when it fails to produce one, employing a “negative gradient” that speeds up RL training (Tajwar et al., 2024).

$$A^\pi(s_h, a_h) := Q^\pi(s_h, a_h) - V^\pi(s_h) = Q^\pi(s_h, a_h) - Q^\pi(s_{h-1}, a_{h-1}). \quad (2)$$

Recall that since we view process rewards as potential functions in the MDP, they can be computed under

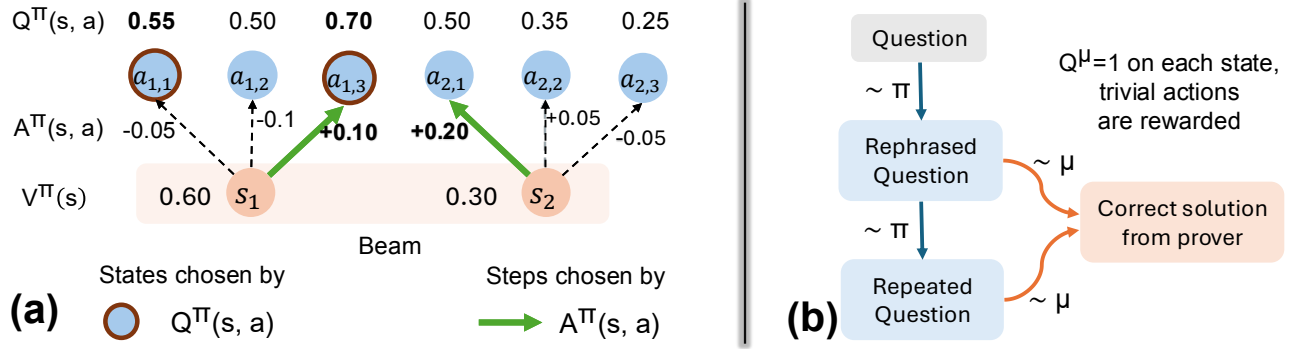


Figure 2 | **Issues with using Q-values as process rewards:** (a): 与  $A^\pi$  不同,  $Q^\pi$  将动作评估与前一状态的  $Q$ -值结合起来。使用  $Q^\pi$  的束搜索利用高概率状态, 而将  $A^\pi$  (e.g. 和  $Q^\pi + \alpha A^\pi$  添加到 Eq. 5) 中有助于探索通过执行促进进展的动作所达到的状态, i.e., 从而增加成功的机会。 (b):  $Q^\mu$  来自强大的证明器  $\mu$  可能会为琐碎的动作分配不公正的奖励。

显示的是优势——而不是价值函数 (Luo et al., 2024; Wang et al., 2024) ——在每一步衡量一种“进步”概念, 更适合作为搜索和RL中的密集奖励 (主要为了探索)。然后在第3.3节和第3.4节中, 我们显示这种进步或优势值在策略  $\mu$  下衡量得最好, 不同于基础策略  $\pi$ 。我们称这个策略为  $\mu$ , 即 **prover policy**。

### 3.1. 过程奖励应该是优势, 而不是价值函数

为了理解与潜在函数的关系, 我们首先研究测试时的束搜索, 并指出Snell等人 (2024) 使用基策略  $\pi$  的价值函数  $Q^\pi(s, a)$  来奖励状态  $s$  中的动作  $a$  的一些挑战。考虑图2(a)中的例子, 在束中的两个状态中, 我们采样了3个动作。如果我们纯粹根据  $Q^\pi$  的最高值来选择下一个状态, 我们将比较来自不同状态的步骤 (e.g.,  $a_{1,1}$  与  $a_{2,1}$ ) 之间的差异。显然, 预期最终结果 i.e.,  $Q^\pi(s_1, a_{1,1}) - V^\pi(s_1)$  的减少意味着  $a_{1,1}$  by itself 对从  $s_1$  成功概率的负面影响为 -0.05, 而  $a_{2,1}$  对  $s_2$  的成功概率有正面影响 +0.20。然而, 基于 *absolute* 值的束扩展保留了导致负面进展的动作, 并从束中移除了状态  $s_2$  (因为束的大小为2)。换句话说,  $Q^\pi$  未能将“评估”一个动作 (步骤) 与“前一状态展示的潜力”区分开来。这在每个问题中都不是一个问题, 特别是在束容量无限制的情况下, 但在有限的计算和采样约束下, 使用  $Q^\pi$  可能会保留那些可能导致整体成功概率降低的不利步骤的状态。如果我们还能利用前一步做出的 **progress** 以及成功概率  $Q^\pi$  来决定在束中保留什么, 那么我们可以解决这种权衡。

**How can we measure the “progress” made by a step?** 一种方法是考虑在步骤前后成功概率的相对增加/减少。这一概念通过策略  $\pi$  下步骤的优势 (公式 2) 来正式化。此外, 由于优势可以为步骤赋予正或负值, 基于优势训练基础策略不仅在基础策略生成促进进步的步骤 (其中  $A^\pi > 0$ ) 时监督它, 也在它未能生成一个时使用“负梯度”来加速 RL 训练 (Tajwar 等人, 2024)。

$$A^\pi(s_h, a_h) := Q^\pi(s_h, a_h) - V^\pi(s_h) = Q^\pi(s_h, a_h) - Q^\pi(s_{h-1}, a_{h-1}). \quad (2)$$

Recall 我们将过程奖励视为MDP中的潜在函数, 因此它们可以在其中计算

any policy  $\mu$ , which can be the base policy. However, in the above example, reasons for which  $Q^\pi$  is a seemingly unfit choice for process rewards also apply to  $Q^\mu$ . Nevertheless, we can possibly use advantage under  $\mu$ :  $A^\mu$ , which measures the progress made by a step to improve the likelihood of success under  $\mu$ . In that case, how should we choose this policy  $\mu$ , that we call the prover policy, and should it be necessarily different from base policy  $\pi$ ? Before diving into the choice of  $\mu$ , we discuss a more pertinent question: how should we use  $A^\mu$  in conjunction with outcome rewards for improving the base policy  $\pi$ ? We will then formally reason about the choice of  $\mu$  in Secs. 3.3 and 3.4.

### 3.2. Our Approach: Process Advantage Verifiers (PAV)

For building an approach that uses process rewards  $A^\mu$  together with the outcome reward  $Rex$  to improve the base policy  $\pi$ , we situate ourselves in the context of improving  $\pi$  with online RL. If all we had was access to  $Rex$  on  $\mathcal{D}$ , the standard RL objective is given by:

$$\ell_{\text{ORM-RL}}(\pi) := \mathbb{E}_{\mathbf{x} \sim \mathcal{D}, (a_1, \dots, a_H) \sim \pi(\cdot | \mathbf{x})} [\text{Rex}((\mathbf{x}, a_1, \dots, a_H), \mathbf{y}_x^*)] . \quad (3)$$

Inspired by how reward bonuses (and potential functions) are additive (Bellemare et al., 2016; Ng et al., 1999), one way to use process rewards  $A^\mu$  is to combine it with the standard RL objective as:

$$\ell_{\text{PAV-RL}}^{\pi'}(\pi) := \ell_{\text{ORM-RL}}(\pi) + \alpha \cdot \sum_{h=1}^H \mathbb{E}_{s_h \sim d_h^{\pi'}} \mathbb{E}_{a_h \sim \pi(\cdot | s_h)} [A^\mu(s_h, a_h)] \quad (4)$$

The term in red is the difference in likelihoods of success of the prover  $\mu$ , summed over consecutive steps (a notion of **progress**). Here,  $d_h^{\pi'}$  denotes the distribution over states at step  $h$ , visited by the old policy  $\pi'$  (policy at previous iterate). Following policy gradient derivations (Williams, 1992):

$$\boxed{\nabla_{\pi} \ell_{\text{PAV-RL}}^{\pi'}(\pi) \Big|_{\pi'=\pi} = \sum_{h=1}^H \nabla_{\pi} \log \pi(a_h | s_h) \cdot \underbrace{(Q^\pi(s_h, a_h) + \alpha \cdot A^\mu(s_h, a_h))}_{\text{effective reward}}} \quad (5)$$

At a glance, we can view  $Q^\pi(s_h, a_h) + \alpha A^\mu(s_h, a_h)$  as the effective reward for step  $a_h$  when scored against a combination of the outcome evaluation  $Rex$ , i.e.,  $Q^\pi$ , and process rewards  $A^\mu$ . Thus, we can optimize Eq. 4 indirectly via (a) running beam-search against the effective reward; or (b) online RL where the policy gradients are given by Eq. 5. For either of these, we need access to verifiers that are trained to predict the advantage  $A^\mu(s_h, a_h)$  under the prover. We refer to these verifiers as **process advantage verifiers (PAVs)**. In Sec. 4.2 we describe how to train PAVs, but now we use the above formulation to reason about how to choose prover  $\mu$  that is most effective at improving base  $\pi$ .

We also remark that the term in red resembles prior work on imitation learning via policy optimization (Ross and Bagnell, 2014; Sun et al., 2017), where the main aim is to learn a policy  $\pi$  that imitates the prover  $\mu$ , or to improve upon it to some extent. Of course, this is limiting since our goal is to not just take actions that perform at a similar level as  $\mu$ , but to improve the base policy even further, and using a combination of  $Q^\pi$  and  $A^\mu$  is critical towards this goal.

**How should we choose the prover  $\mu$ ?** Perhaps a natural starting point is to set the prover to be identical to the base policy, i.e.,  $\mu = \pi$ , which produces process rewards that prior works have considered Shao et al. (2024). However, setting  $A^\pi = A^\mu$  in Eq. 5 results in exactly the same policy gradient update as only optimizing outcome evaluation  $Rex$ . Moreover, for a poor base policy  $\pi$ , where  $Q^\pi \approx 0$  on most states,

任何策略  $\mu$ ，可以作为基础策略。然而，在上述示例中，适用于过程奖励的  $Q^\pi$  看似不合适的原因同样适用于  $Q^\mu$ 。不过，我们可能可以使用在  $\mu$  下的优势  $A^\mu$ ：衡量一步进展以提高在  $\mu$  下成功概率的进展程度。在这种情况下，我们应该如何选择这种我们称之为证明策略的策略  $\mu$ ，它是否必须与基础策略  $\pi$  不同？在深入选择  $\mu$  之前，我们先讨论一个更相关的问题：我们应该如何将  $A^\mu$  与结果奖励结合使用以改进基础策略  $\pi$ ？然后在第 3.3 节和第 3.4 节中正式讨论  $\mu$  的选择。

### 3.2. 我们的方法：过程优势验证器（PAV）

为了构建一种使用过程奖励  $A^\mu$  与结果奖励  $\text{Rex}$  一起改进基础策略  $\pi$  的方法，我们将自己置于使用在线 RL 改进  $\pi$  的上下文中。如果仅能访问  $\mathcal{D}$  上的  $\text{Rex}$ ，标准的 RL 目标由以下公式给出：

$$\ell_{\text{ORM-RL}}(\pi) := \mathbb{E}_{\mathbf{x} \sim \mathcal{D}, (a_1, \dots, a_H) \sim \pi(\cdot | \mathbf{x})} [\text{Rex}((\mathbf{x}, a_1, \dots, a_H), \mathbf{y}_x^*)]. \quad (3)$$

受奖励奖金（以及潜在函数）是可加性的启发（Bellemare等，2016年；Ng等，1999），一种使用过程奖励  $A^\mu$  的方法是将其与标准 RL 目标结合使用：

源文本: ,译文

$$\ell_{\text{PAV-RL}}^{\pi'}(\pi) := \ell_{\text{ORM-RL}}(\pi) + \alpha \cdot \sum_{h=1}^H \mathbb{E}_{s_h \sim d_h^{\pi'}} \mathbb{E}_{a_h \sim \pi(\cdot | s_h)} [A^\mu(s_h, a_h)] \quad (4)$$

The 术语用红色标注，表示证明者  $\mu$  成功概率差值之和，涵盖了连续步骤（**progress** 的一个概念）。这里， $d_h^{\pi'}$  表示由旧策略  $\pi'$ （在先前迭代中访问的状态分布，在步骤  $h$ 。遵循策略梯度推导（Williams, 1992）：

$$\left. \nabla_{\pi} \ell_{\text{PAV-RL}}^{\pi'}(\pi) \right|_{\pi'=\pi} = \sum_{h=1}^H \nabla_{\pi} \log \pi(a_h | s_h) \cdot \underbrace{(Q^\pi(s_h, a_h) + \alpha \cdot A^\mu(s_h, a_h))}_{\text{effective reward}} \quad (5)$$

从整体上看，我们可以将  $Q^\pi(s_h, a_h) + \alpha A^\mu(s_h, a_h)$  视为在结合结果评估  $\text{Rex}$ ，即  $Q^\pi$  和过程奖励  $A^\mu$  的情况下，第  $a_h$  步的有效奖励。因此，我们可以通过 (a) 使用有效奖励进行 beam-search；或 (b) 在线 RL，其中策略梯度由 Eq. 5 给出，间接优化 Eq. 4。对于这两种方法，我们需要访问能够预测证明者下优势  $A^\mu(s_h, a_h)$  的验证器。我们将这些验证器称为 **process advantage verifiers (PAVs)**。在第 4.2 节中，我们将描述如何训练 PAVs，但现在我们使用上述公式来推理如何选择最有效的证明者  $\mu$  以提高基础  $\pi$ 。

我们还注意到，用红色标记的项类似于通过策略优化进行模仿学习的先前工作（Ross 和 Bagnell, 2014; Sun 等人, 2017），其中的主要目标是学习一个策略  $\pi$  来模仿证明者  $\mu$ ，或者在某种程度上改进它。当然，这有些限制，因为我们的目标不仅仅是采取与  $\mu$  相似水平的动作，而是要进一步改进基础策略，而使用  $Q^\pi$  和  $A^\mu$  的结合是实现这一目标的关键。

我们应该如何选择证明者  $\mu$ ？也许一个自然的起点是将证明者设置为与基线策略 i.e.、 $\mu = \pi$  相同，这会产生先前工作认为类似于 Shao 等人（2024）的研究中的过程奖励。然而，将  $A^\pi = A^\mu$  设定在式 5 中会导致与仅优化结果评估  $\text{Rex}$  完全相同的策略梯度更新。此外，对于一个较差的基线策略  $\pi$ ，其中在大多数状态下  $Q^\pi \approx$  接近 0，

the term  $A^\pi$  would also be  $\approx 0$ , and hence running beam search with the effective rewards would not be informative at all. Hence, **a better approach is to use a different prover policy**, but a very weak prover  $\mu$  will likely run into similar issues as a poor base policy. We could instead use a very capable prover  $\mu$ , but unfortunately even this may not be any better than optimizing only the outcome reward either. To see why, consider a scenario where  $\pi$ 's response contains an intermediate step that does not help make progress towards the solution (e.g.,  $\pi$  simply restates the question, see Fig. 2(b)). Here,  $Q^\mu$  for a capable prover before and after this irrelevant step will be identical since  $\mu$  can succeed from either step. This means that  $\mu$  fails to distinguish steps, resulting in  $A^\mu \approx 0$  in most cases. Training with this process reward during RL will then lead to gradients that are equivalent to those observed when purely optimizing  $\ell_{\text{ORM-RL}}$ . In fact, empirically, we observe that online RL with  $Q^\mu$  from strong provers leads to policies that only produce re-phrasings of the question (App. G) and do not succeed at solving the question. Clearly, *any* policy different from the base policy cannot serve as a prover. So, how do we identify a set of good provers? Can they indeed be weaker than the base policy? We answer next.

**Takeaway:** What should process rewards measure during test-time search and online RL?

- Process rewards should correspond to progress, or **advantage**, as opposed to absolute  $Q$ -values, for a better explore-exploit tradeoff during beam search and online RL.
- Advantages should be computed using a **prover** policy, different from the base policy.

### 3.3. Analysis in a Didactic Setting: Learning a Planted Sub-sequence

In this section, we aim to characterize prover policies that are effective in improving the base policy. To do so, we first introduce a didactic example, representative of real reasoning scenarios to illustrate the main intuition. Then, we will formalize these intuitions in the form of theoretical results.

**Didactic example setup.** Given an unknown sub-sequence  $y^*$  consisting of tokens from vocabulary  $\mathcal{V} := \{1, 2, \dots, 15\}$ , we train a policy  $\pi$  to produce a response which contains this sub-sequence. The task completion reward is terminal and sparse, i.e.,  $r(y, y^*) = 1$  for a  $y$  if and only if  $y^*$  appears in  $y$ . By design, the reward  $r(y, y^*)$  resembles outcome reward  $\text{Rex}(y, y_x^*)$  in Sec. 2. The prover policy  $\mu$  is a procedural policy, parameterized by a scalar  $\gamma > 0$  (details in App. B). As  $\gamma$  increases, the performance of  $\mu$  improves and  $\rightarrow 1$  as  $\gamma \rightarrow \infty$ . For simplicity, we assume oracle access to ground-truth  $A^\mu$  and  $Q^\pi$ , and alleviate errors from learned verifiers approximating these values.

**(1) RL with effective reward  $Q^\pi + \alpha A^\mu$  is 10× more sample-efficient than only outcome reward.** In Fig. 3(a), we first note that training  $\pi$  with this effective reward under a prover  $\mu$  with strength  $\gamma = 10$ , produces optimal performance (100% accuracy) in 350 iterations, despite starting from a mediocre initialization for  $\pi$  ( $\gamma = 5.0$ ). Training with only outcome reward is ineffective. More importantly, in Fig. 3(b), we note that effective rewards only help for a set of provers, in  $\gamma \in [8.0, 15.0]$ . Outside this range, we observed advantages  $A^\mu$  were close to 0 on most states, either because  $\mu$  was poor (small  $\gamma$ ) and was unable to generate  $y^*$  even when  $\pi$  got the sequence partially correct, or because  $\mu$  was strong (large  $\gamma$ ) that it generated  $y^*$  with almost equal likelihood from all prefixes.

**(2) Effective reward improves Pass @N by 5× over only outcome reward.** We report the “Pass @N” performance in Fig. 3(c), which measures the maximum reward  $r$  across  $N$  traces sampled *i.i.d.* from  $\pi$  and hence, represents the ceiling on the performance of any test-time search method that picks a single response from multiple draws (e.g., as in Best-of-N). For a policy trained with the effective reward for 100 iterations, the Pass @N performance grows 5× faster with  $N$ , compared to the policy trained with only



术语  $A^\pi$  也会是  $\approx 0$ ，因此使用有效奖励运行束搜索将毫无信息性。因此，

**a better approach is to use a different prover policy**，但一个非常弱的证明者  $\mu$  很可能遇到与基础策略较差时类似的问题。相反，我们可以使用一个非常有能力的证明者  $\mu$ ，但不幸的是，即使这样也可能不如仅优化结果奖励更好。原因如下，考虑一个场景，其中  $\pi$  的回应包含一个中间步骤，这个步骤并没有帮助向解决方案前进（e.g.,  $\pi$  只是重述了问题，参见图 2(b)）。在这种情况下，在这个无关步骤之前和之后，一个有能力的证明者  $Q^\mu$  的表现将是一样的，因为  $\mu$  可以从任一步骤成功。这意味着  $\mu$  无法区分步骤，导致在大多数情况下  $A^\mu \approx 0$ 。使用这个过程奖励进行 RL 训练将导致与仅优化  $\ell_{\text{ORM-RL}}$  时观察到的梯度相同。实际上，我们观察到，使用来自强大证明者的  $Q^\mu$  在线 RL 生成的策略只会产生问题的重新表述（附录 G），而无法解决这个问题。显然，any 与基础策略不同的策略不能作为证明者。那么，我们如何识别一组好的证明者？它们确实可以比基础策略弱吗？我们将在接下来回答。

- 过程奖励应该与进度或优势相对应，而不是与绝对的Q-值相对应，以便在束搜索和在线强化学习中获得更好的探索-利用权衡。
- 优点应该使用证明者策略来计算，不同于基础策略。

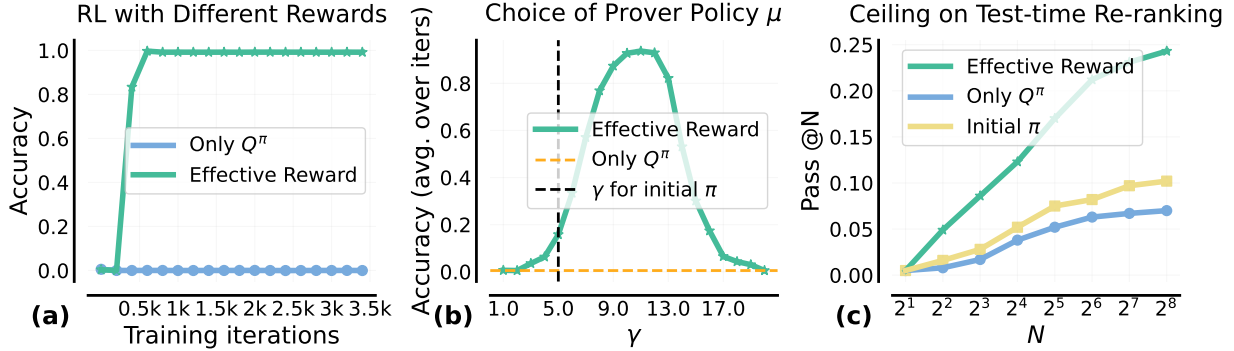
### 3.3. 在教学环境中进行分析：学习嵌入的子序列

在本节中，我们旨在表征能够有效改进基政策的证明者策略。为此，我们首先引入一个教学示例，该示例代表真实的推理场景，以阐明主要的直觉。然后，我们将这些直觉形式化为理论结果。

**教学示例设置。** 给定一个未知子序列  $y^*$ ，由词汇表  $\mathcal{V} := \{1, 2, \dots, 15\}$  中的标记组成，我们训练一个策略  $\pi$  以生成包含此子序列的响应。任务完成奖励是终端且稀疏的，i.e.,  $r(y, y^*) = 1$  当且仅当  $y$  出现在  $y^*$  中时。设计上，奖励  $r(y, y^*)$  类似于第 2 节中的结果奖励  $\text{Rex}(y, y_x^*)$ 。证明者策略  $\mu$  是一个过程性策略，由标量  $\gamma > 0$  参数化（详情参见附录 B）。随着  $\gamma$  增加， $\mu$  的性能提高，并且当  $\gamma \rightarrow \infty$  时  $\rightarrow 1$ 。为了简化，我们假设可以访问 ground-truth  $A^\mu$  和  $Q^\pi$  的 oracle 访问，并减轻由学习验证器近似这些值时产生的错误。

(1) 带有效奖励  $Q^\pi + \alpha A^\mu$  的 RL 比仅使用结果奖励的 RL 在样本效率上高 10 $\times$ 。在图 3(a) 中，我们首先注意到，在强度为  $\gamma = 10$  的证明者  $\mu$  下，使用这种有效奖励训练  $\pi$  可以在 350 次迭代内达到最优性能（准确率为 100%），尽管初始条件较差，初始值为  $\pi$  ( $\gamma = 5.0$ )。仅使用结果奖励的训练是无效的。更重要的是，在图 3(b) 中，我们注意到有效奖励仅在  $\gamma \in [8.0, 15.0]$  个证明者中有所帮助。在此范围之外，我们观察到在大多数状态下，有效奖励的优势  $A^\mu$  接近于 0，要么是因为  $\mu$  较差（强度  $\gamma$  较小），无法在  $\pi$  部分正确时生成  $y^*$ ，要么是因为  $\mu$  较强（强度  $\gamma$  较大），从所有前缀生成  $y^*$  的可能性几乎相同。

(2) 有效的奖励比仅基于结果的奖励提高 Pass @N 5 $\times$ 。我们在图 3(c) 中报告了“Pass @N”的性能，这衡量了从  $N$  采样的 i.i.d. 轨迹中获得的最大奖励  $r$ ，因此代表了任何在多个抽取中选择单一响应的测试时搜索方法的性能上限（如 e.g. 中的 Best-of-N）。对于用有效的奖励训练 100 个迭代的策略，Pass @N 的性能随着  $N$  的增长比仅使用  $\gamma$  训练的策略快 5 $\times$ 。



**Figure 3 | Results for our didactic analysis:** (a): We train base policy via RL with either effective reward  $Q^\pi + \alpha A^\mu$ , or the typical  $Q^\pi$  (computed via Monte-Carlo sampling). (b): We vary the strength  $\gamma$  of the prover  $\mu$  used to compute advantages  $A^\mu$  in the effective reward, and plot the base policy accuracy averaged over the RL run. (c): We plot the max score out of  $N$  responses (Pass @N) sampled *i.i.d.* from an undertrained base policy (iter 100) .

the outcome reward. Due to only sparse feedback, the latter policy does not learn to sample partially correct  $y^*$ , whereas a policy trained with the effective reward produces partially correct  $y^*$ , and is able to sample the complete  $y^*$  with higher likelihood during Pass @N.

**Takeaway:** Online RL with process rewards from different prover policies.

Effective rewards  $Q^\pi + \alpha A^\mu$  from prover  $\mu$ : (i) improve sample efficiency of online RL, and (ii) yield policies with better Pass @N performance, over using only outcome rewards. But, advantages of very capable or poor  $\mu$  do not improve base policy beyond outcome rewards.

### 3.4. Theory: Provers Complementary to the Base Policy Boost Improvement

From our didactic analysis, it is clear that process rewards  $A^\mu$  under different provers  $\mu$  disparately affect the base policy that optimizes  $Q^\pi + \alpha A^\mu$  via online RL. We now present a formal analysis of why this happens and characterize a class of provers that can guarantee non-trivial improvements to the base policy. For simplicity, we assume oracle access to  $Q^\pi, A^\mu$  at every state-action pair  $(s_h, a_h)$  and prove our result in the tabular RL setting, where the policy class is parameterized using the softmax parameterization in Agarwal et al. (2021). Proofs for this section are in App. F.

**Main intuitions.** We expect a prover  $\mu$  to improve a base policy  $\pi$  only when  $\mu$  is able to *distinguish different actions taken by  $\pi$* , by attaining sufficiently varying advantage values  $A^\mu(s_h, a)$  for actions  $a$  at state  $s_h$ . This can be formalized under the notion of sufficiently large variance across actions,  $\mathbb{V}_{a \sim \pi} [A^\mu(s_h, a)]$ . In that case, can we simply use a policy with large advantage variance under any measure? No, because when the prover  $\mu$  ranks actions at a given state very differently compared to the base policy  $\pi$  (e.g., if  $A^\mu$  and  $A^\pi$  are opposite), then effective rewards  $Q^\pi + \alpha A^\mu$  will be less reliable due to conflicting learning signals. Thus, we want  $\mathbb{E}_\pi [\langle A^\mu, A^\pi \rangle]$  to not be too negative, so that  $\mu$  and  $\pi$  are *reasonably aligned* on their assessment of steps from  $\pi$ .

In Theorem 3.1, we present our result on policy improvement where the base policy is updated with natural policy gradient (Kakade, 2001a):  $\pi_{t+1}(a | s_h) \propto \exp(\gamma \cdot (Q^\pi(s_h, a) + A^\mu(s_h, a)))$ . We note that in this idealized update rule, swapping  $Q$  values (of  $\mu$  or  $\pi$ ) with advantages does not affect the update since we assume access to all possible actions when running the update. Nonetheless, despite this simplifying assumption, the analysis is able to uncover good choices for the prover policy  $\mu$  for computing process

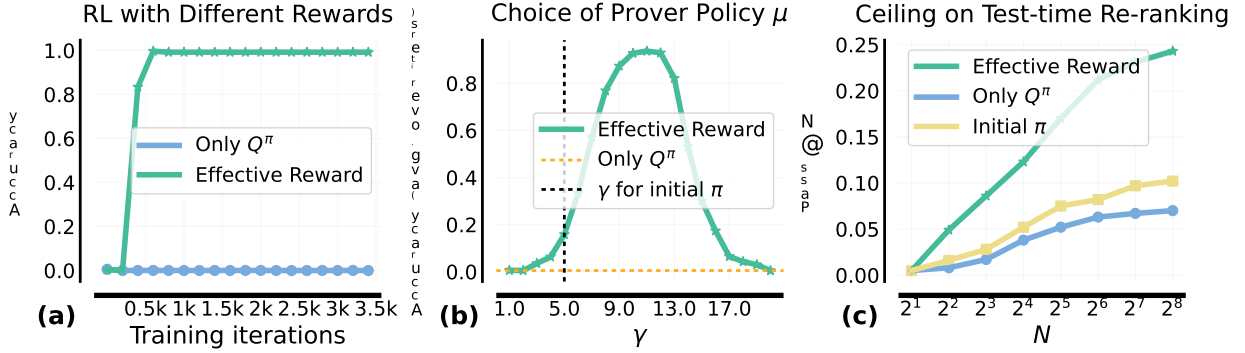


图 3 | **Results for our didactic analysis:** (a): 我们通过使用有效的奖励  $Q^\pi + \alpha A^\mu$  或者通过蒙特卡洛采样计算得到的典型奖励  $Q^\pi$  (来训练基础策略)。 (b): 我们改变用于计算优势  $A^\mu$  的证明器  $\mu$  的强度  $\gamma$ , 并在 RL 运行中绘制基础策略的平均准确性。 (c): 我们绘制从欠训练的基础策略 (迭代 100) 中采样得到的响应  $N$  中的最大得分 (Pass @  $N$ )。

结果奖励。由于只有稀疏反馈，后者策略未能学习采样部分正确的  $y^*$ ，而使用有效奖励训练的策略会产生部分正确的  $y^*$ ，并且能够在 Pass @  $N$  期间以更高的概率采样完整的  $y^*$ 。

有效的奖励  $Q^\pi + \alpha A^\mu$  来自证明者  $\mu$ : (i) 提高在线 RL 的样本效率, (ii) 在仅使用结果奖励的情况下产生具有更好 Pass @  $N$  性能的策略。但是，非常强大或较差的  $\mu$  的优势并不能使基础策略超越结果奖励。

### 3.4. 理论：辅助策略增强基政策提升改进

从我们的教学分析来看，过程奖励  $A^\mu$  在不同的证明者  $\mu$  下，对通过在线 RL 优化  $Q^\pi + \alpha A^\mu$  的基策略的影响是不同的。我们现在正式分析这种现象的原因，并刻画一类可以保证对基策略产生非平凡改进的证明者。为了简化问题，我们假设在每个状态-动作对  $(s_h, a_h)$  处都有 oracle 访问  $Q^\pi, A^\mu$ ，并在基于表格的 RL 设置中证明我们的结果，其中策略类使用 Agarwal 等人 (2021) 中的 softmax 参数化。本节的证明见附录 F。

主要直觉。我们期望证明者  $\mu$  只能在证明者  $\mu$  能够通过获得在状态  $s_h$  中不同动作  $a$  的足够变化的优势值  $A^\mu(s_h, a)$  来区分动作  $\pi$  的不同执行时，才能改进基础策略  $\pi$ 。这可以在足够大的动作方差的概念下形式化，记作  $\sigma_{a \sim \pi}[A^\mu(s_h, a)]$ 。然而，我们是否可以简单地使用在任何度量下具有大优势方差的策略？不可以，因为当证明者  $\mu$  在给定状态下对动作的排序与基础策略  $\pi$  (非常不同，例如，如果  $A^\mu$  和  $A^\pi$  是相反的)，则有效的回报  $Q^\mu + \alpha A^\pi$  会因为冲突的学习信号而变得不可靠。因此，我们希望  $\pi[A^\mu, A^\pi]$  不太负，这样  $\mu$  和  $\pi$  在评估从  $\pi$  的步骤时会比较合理。

在定理 3.1 中，我们呈现了关于策略改进的结果，其中基础策略使用自然策略梯度 (Kakade, 2001a) 进行更新： $\pi_{t+1}(a | s_h) \propto \exp(\gamma \cdot (Q^\pi(s_h, a) + A^\mu(s_h, a)))$ 。我们注意到，在这个理想化的更新规则中，用优势值替换  $Q$  值 (无论是  $\mu$  还是  $\pi$ ) 不会影响更新，因为我们假设在运行更新时可以访问所有可能的动作。尽管如此，即使在这种简化假设下，分析仍然能够发现计算过程中证明策略  $\mu$  的良好选择。

reward  $A^\mu$ , and is orthogonal to the design consideration of advantages or  $Q$ -values as process rewards that we have discussed so far in this paper. Theorem 3.1 formalizes our intuition by showing that policy improvement at iteration  $t$ , grows as the variance in  $A^\mu$  values increases (higher distinguishability) and reduces when  $A^\mu$  and  $A^\pi$  become extremely misaligned. This will then allow us to discuss a special case for the case of Best-of-K policies as provers as an immediate corollary.

**Theorem 3.1** (Lower bound on policy improvement; informal). *For base policy iterate  $\pi_t$ , after one step of policy update, with learning rate  $\gamma \ll 1$ , the improvement over a distribution of states  $\rho$ :*

$$\mathbb{E}_{s \sim \rho} [V^{\pi_{t+1}}(s) - V^{\pi_t}(s)] \gtrsim \underbrace{\gamma \cdot \mathbb{E}_{s \sim \rho} \mathbb{V}_{a \sim \pi_t} [A^\mu(s, a)]}_{\text{distinguishability from } \mu} + \underbrace{\gamma \cdot \mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} [A^\mu(s, a) A^{\pi_t}(s, a)]}_{\text{alignment between } \pi_t \text{ and } \mu} \quad (6)$$

It may seem that the base policy  $\pi$  can only learn from an improved prover  $\mu$ , but our result shows that a **weak prover can also amplify a stronger base policy**, since a weak prover  $\mu$  may have a lower average of  $Q^\mu$  under its own measure, but still have higher variance across  $Q^\mu$  (compared to  $Q^\pi$ ) when evaluated under  $\pi$  (see Proposition F.1 in App. F.5 for formal discussion). This tells us that **rewarding progress under a prover is different from typical knowledge distillation or imitation learning algorithms** (Hinton, 2015; Rusu et al., 2015) that in most cases remain upper bounded by the performance of the stronger teacher. So provers cannot be characterized purely by strength, what is a class of provers that is a reasonable starting point if we were to improve any base policy  $\pi$ ?

**The policy class of “Best-of-K” (computed over base policies) contain complementary provers.** A good starting point to identify good provers for a base policy  $\pi$ , is the class of Best-of-K policies or  $\text{BoK}(\pi)$ . Recall from Sec. 2 that the performance of  $\text{BoK}(\pi)$  increases monotonically with  $K$ . Applying Theorem 3.1 to this class, we arrive at Remark 3.1 that recommends using  $\text{BoK}(\pi)$  with  $K > 1$  as a prover policy for a poor base policy  $\pi$ . However,  $K$  cannot be too large always since when  $Q^\pi(s, a) \approx 1$ , increasing  $K$  too much can hurt distinguishability of different steps at that state. In the next section, we empirically note that the policies in the class of  $\text{BoK}(\pi)$  indeed induce different performance gains when used as prover policies, and we find Bo4 to be a good choice for test-time search over most base policies.

**Remark 3.1.** *When  $Q^\pi(s, a) = O(1/K)$ ,  $\forall s, a$ , using  $\text{BoK}(\pi)$  as a prover for base  $\pi$  improves distinguishability (and improvement) by  $\Omega(K^2)$ , and make alignment worse at most by  $O(K)$ .*

**Takeaway:** Formal characterization of good prover policies that improve the base policy.

Provers with advantages that can **distinguish** actions taken by the base policy (more strongly than the base policy itself) but are **not too misaligned** from the base, boost improvements on each update of the base policy. We call such policies **complementary provers**.  $\text{BoK}(\pi)$  for any base policy  $\pi$  for  $K > 1$  can provide a good starting choice of prover policies.

## 4. Results: Scaling Test-Time Compute with PAVs

Now, we study how process verifiers can scale up test-time compute. While our derivations from Sec. 3.2 were with RL, we can also use the *effective reward*  $Q^\pi(s_h, a_h) + \alpha \cdot A^\mu(s_h, a_h)$  for running beam search over intermediate steps sampled from base policy  $\pi$ . To do so, we train a process advantage verifier to predict  $A^\mu$ , along with a process reward model  $Q^\pi$ . PAV training is done using procedures discussed in Sec. 4.2. While the candidates of the beam are selected using a combination of both the PAV and the PRM  $Q^\pi$ ,



奖励  $A^\mu$ ，并且与我们在本文中迄今讨论的设计考虑的优势或  $Q$ -值作为过程奖励正交。定理 3.1 通过证明在第  $t$  次迭代的策略改进随着  $A^\mu$  值的方差增加（更高的可区分性）而增长，并在  $A^\mu$  和  $A^\pi$  极度不匹配时减少，从而正式化了我们的直觉。这将使我们能够讨论 Best-of-K 策略作为证明者的特殊情况作为直接推论。

定理 3.1 (策略改进的下界；非正式). *For base policy iterate  $\pi_t$ , after one step of policy update, with learning rate  $\gamma \ll 1$ , the improvement over a distribution of states  $\rho$ :*

$$\mathbb{E}_{s \sim \rho} [V^{\pi_{t+1}}(s) - V^{\pi_t}(s)] \gtrsim \underbrace{\gamma \cdot \mathbb{E}_{s \sim \rho} \mathbb{V}_{a \sim \pi_t} [A^\mu(s, a)]}_{\text{distinguishability from } \mu} + \underbrace{\gamma \cdot \mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} [A^\mu(s, a) A^{\pi_t}(s, a)]}_{\text{alignment between } \pi_t \text{ and } \mu} \quad (6)$$

它可能看起来基础策略  $\pi$  只能从改进的证明者  $\mu$  中学习，但我们的结果表明，一个较弱的证明者也可以增强一个更强的基础策略，因为一个较弱的证明者  $\mu$  可能在其自身的度量下  $Q^\mu$  的平均值较低，但在根据  $Q^\mu$ （评估时相对于  $Q^\pi$ ）仍然具有更高的方差，参见附录 F.5 中的命题 F.1 以获得正式讨论）。这告诉我们

**rewarding progress under a prover is different from typical knowledge distillation or imitation learning algorithms** (Hinton, 2015; Rusu 等人, 2015) 在大多数情况下，仍然被更强的教师的表现所上界限制。所以证明者不能单纯地用强度来表征，如果我们想要改进任何基础策略  $\pi$ ，那么什么是一类合理的起点？

政策类别“Best-of-K”（基于基础策略计算）包含互补的证明器。识别基础策略  $\pi$  的良好证明器的一个起点是 Best-of-K 策略或  $\text{BoK}(\pi)$  类别。回想第二章可知， $\text{BoK}(\pi)$  的性能随着  $K$  单调增加。将定理 3.1 应用于此类别，我们得出 Remark 3.1，建议使用  $\text{BoK}(\pi)$  与  $K > 1$  作为证明策略，以用于较差的基础策略  $\pi$ 。然而， $K$  不能总是太大，因为当  $Q^\pi(s, a) \approx 1$  时，过度增加  $K$  可能会损害该状态下不同步骤的可区分性。在下一节中，我们通过实验发现， $\text{BoK}(\pi)$  类别中的策略作为证明策略时确实会产生不同的性能增益，并且我们发现 Bo4 是在大多数基础策略上进行测试时搜索的良好选择。

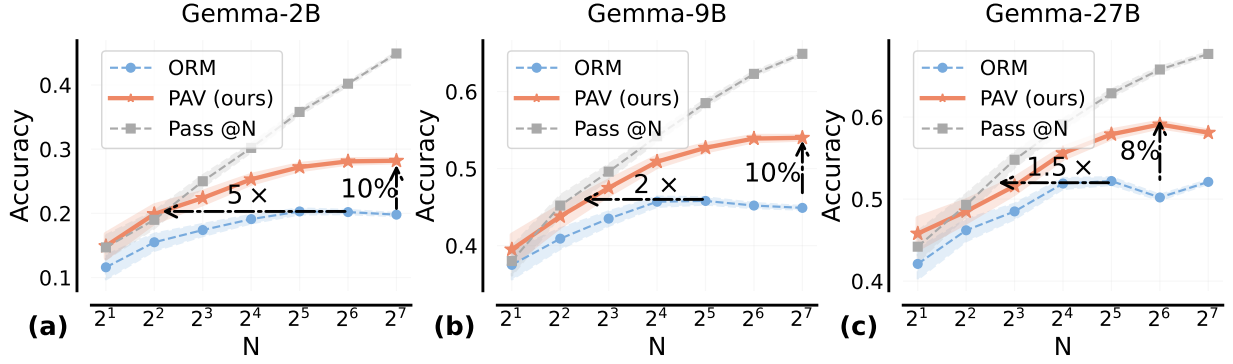
备注 3.1. *When  $Q^\pi(s, a) = O(1/K)$ ,  $\forall s, a$ , using  $\text{BoK}(\pi)$  as a prover for base  $\pi$  improves distinguishability (and improvement) by  $\Omega(K^2)$ , and make alignment worse at most by  $O(K)$ .*

证明器具有优势，可以区分基础策略采取的动作（比基础策略本身更为明显），但又不会与基础策略相差太远，从而在每次更新基础策略时都能提升改进。我们称这样的策略为 **complementary provers**。对于任何基础策略  $\pi$ ， $\text{BoK}(\pi)$  都可以提供一个良好的证明策略初始选择。

#### 4. 结果: 使用 PAVs 扩展测试时计算能力

现在，我们研究如何扩展测试时的计算量。虽然我们在第 3.2 节中的推导是基于强化学习 (RL)，我们也可以使用 *effective reward*  $Q^\pi(s_h, a_h) + \alpha \cdot A^\mu(s_h, a_h)$  在基政策  $\pi$  采样的中间步骤上运行束搜索。为此，我们训练一个过程优势验证器来预测  $A^\mu$ ，同时训练一个过程奖励模型  $Q^\pi$ 。PAV 的训练使用了第 4.2 节中讨论的程序。在束搜索候选的选择中，使用了 PAV 和  $\text{PRMQ}^\pi$  的组合。





**Figure 4 | For test-time search, PAVs are 8 – 10% more accurate and 1.5 – 5× more compute efficient over ORMs:** On samples from (a) Gemma-2B, (b) 9B, and (c) 27B SFT policies, we run test-time beam search with the estimate of effective reward  $Q^\pi + \alpha A^\mu$  (PAV), where  $\mu$  is the Bo4( $\pi$ ) policy. We compare beam search performance with best-of-N, re-ranking with a trained outcome verifier (ORM), or the oracle Rex (Pass @N).

the final candidate is selected using the outcome reward prediction from  $Q^\pi$  itself (i.e., we repurpose the PRM representing  $Q^\pi$  as an ORM). For clarity, we abuse notation and refer to the estimated effective reward (ORM +  $\alpha$  PAV) as PAV directly.

**Setup.** We finetune Gemma 2B, 9B, and 27B (Gemma Team et al., 2024) on MATH (Hendrycks et al., 2021) via supervised fine-tuning (SFT) to get three base policies. The set of provers consists of the three base SFT policies themselves as well as their best-of-K policies for different values of  $K \in \{2^0, \dots, 2^5\}$ . Additional details for the experiments in this section are in App. C.

#### 4.1. PAVs Scale Test-Time Compute by 5 – 10× Over ORMs

**Result 1: PAVs are more compute efficient than ORMs.** In Fig. 4, we plot the performance of beam search with PAVs for different sizes of the beam  $N$ , and compare it with best-of- $N$  using ORMs, i.e., sampling  $N$  complete solutions from the base policy and returning the one with the highest ORM score. To compare PAVs and ORMs, we evaluate the *compute efficiency* of PAVs over ORMs, given by the ratio of total compute needed by PAVs to obtain the same performance as running best-of-128 with ORM. Even when accounting for the fact that running beam search with PAVs does require additional compute *per solution trace* (since each element in the beam samples  $C = 3$  next steps, before scoring and pruning the beam), PAVs are able to scale the compute efficiency by **10×** over ORMs for Gemma-2B, 9B base models, and by **5×** for Gemma-27B model. We use BoK( $\pi$ ) with  $K = 4$  as the prover policy for all base policies  $\pi$ .

We also compare performance with beam search using process verifiers that only predict  $Q^\pi$ , and best-of- $N$  where the ORM is replaced with PAV (PAV-as-ORM). At  $N = 128$ , similar to Luo et al. (2024), we note a similar gain of 4% for “PAV-as-ORM” Fig. 5(a) over only ORMs, for base Gemma-9B  $\pi$ . When comparing beam search with  $Q^\pi$  (Snell et al., 2024), we find that PAVs scale compute efficiency by 8×. Evidently, advantages from the prover in the effective reward positively impact the beam search. Why does  $A^\mu$  help, and for what choice of the prover  $\mu$ ?

**Result 2: Beam search with too weak/strong provers is sub-optimal.** In Fig. 5(b), for the setting when the base policy  $\pi$  is a Gemma-2B SFT model, we compare beam search with PAVs where the provers are given by BoK( $\pi$ ), for different values of  $K$ . Recall that as  $K$  increases, BoK( $\pi$ ) becomes stronger. Corroborating our analysis in Sec. 3.4, our results show that neither too weak (Bo2) or too strong (Bo32) provers perform best. Instead, across all values of  $N$ , we find Bo4 to be dominant. The advantage values

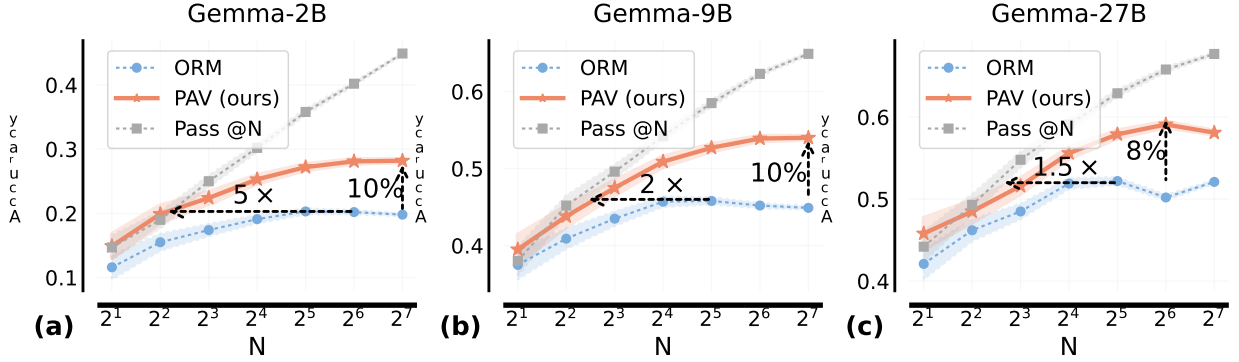


图 4 | **For test-time search, PAVs are 8 – 10% more accurate and 1.5 – 5× more compute efficient over ORMs:** 在来自 (a) Gemma-2B, (b) 9B, 和 (c) 27B SFT 策略的样本上, 我们使用估计的有效奖励  $Q^\pi + \alpha A^\mu$  (PAV) 运行测试时的束搜索, 其中  $\mu$  是 Bo4( $\pi$ ) 策略。我们将束搜索性能与最佳的  $N$  次结果、使用训练的结局验证器 (ORM) 重新排序或 oracle Rex (Pass @N) 进行比较。

最终候选者是使用自身结果奖励预测 (即, 我们将表示  $Q^\pi$  的 PRM 重新用于 ORM)。为了清晰起见, 我们滥用符号, 并将估计的有效奖励 ( $ORM + \alpha PAV$ ) 直接称为 PAV。

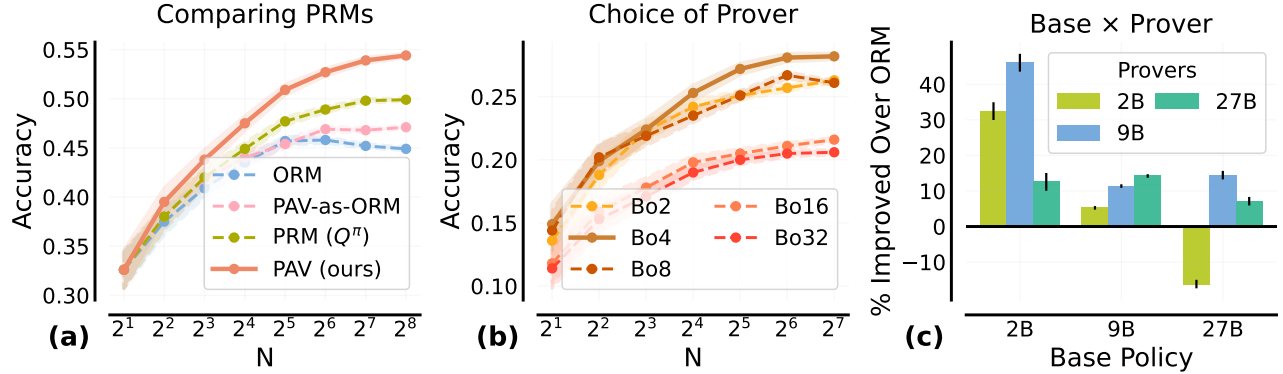
**Setup.** 我们使用监督微调 (SFT) 对 Gemma 2B、9B 和 27B (Gemma Team et al., 2024) 进行微调, 以获得三个基础策略。证明者集合包括这三个基础 SFT 策略本身及其在不同值的  $K \in \{2^0, \dots, 2^5\}$  下的最佳  $K$  策略。本节中实验的更多细节参见附录 C。

#### 4.1. PAVs 规模测试时计算加速比 ORMs 快 5– 倍 10× 倍

**Result 1:** PAVs 比 ORMs 在计算效率上更优。在图 4 中, 我们绘制了使用 PAVs 的不同大小的束搜索性能  $N$ , 并与使用 ORMs 的 best-of- $N$  进行比较, 后者是从基策略中采样  $N$  完整解并返回得分最高的 ORM 得分的解。为了比较 PAVs 和 ORMs, 我们评估了 PAVs 相对于 ORMs 的效率 *compute efficiency*, 该效率由 PAVs 获得与使用 ORMs 运行 best-of-128 相同性能所需的总计算量之比给出。即使考虑到使用 PAVs 进行束搜索确实需要额外的计算 *per solution trace* (, 因为束中的每个元素会采样  $C=3$  个下一步, 然后对束进行评分和修剪), PAVs 仍能将 Gemma-2B 和 9B 基模型的计算效率提高 10× 倍, 将 Gemma-27B 模型的计算效率提高 5× 倍。我们使用 BoK( $\pi$ ) 作为所有基策略的证明策略, 其中  $K=4$  作为证明策略。

我们还将性能与仅预测  $Q^\pi$  的过程验证器使用的束搜索进行比较, 并用 PAV (PAV-as-ORM) 替换 ORM 的最佳-of- $N$  方法。在  $N=128$  时, 类似于 Luo 等人 (2024), 我们注意到 “PAV-as-ORM” 在 Fig. 5 (a) 中相对于仅 ORM 的基线 Gemma-9B 获得了 4% 的类似增益。当我们比较束搜索与  $Q^\pi$  (Snell 等人 (2024)) 时, 发现 PAVs 通过 8× 倍提高了计算效率。显然, 有效奖励中证明者的优点对束搜索产生了积极影响。为什么  $A^\mu$  会有所帮助, 以及对于哪种证明者的  $\mu$  选择?

**Result 2:** 均值搜索中, 证明者过于弱或强都是次优的。在图 5(b) 中, 对于基策略  $\pi$  为 Gemma-2B SFT 模型的设置, 我们将均值搜索与由 BoK( $\pi$ ) 给出的证明者进行比较, 不同值的  $K$ 。请记住, 随着  $K$  的增加, BoK( $\pi$ ) 变得更强。与第 3.4 节中的分析一致, 我们的结果显示, 既不是过于弱 (Bo2) 也不是过于强 (Bo3) 的证明者表现最佳。相反, 对于所有值的  $N$ , 我们发现 Bo4 占主导地位。优势值

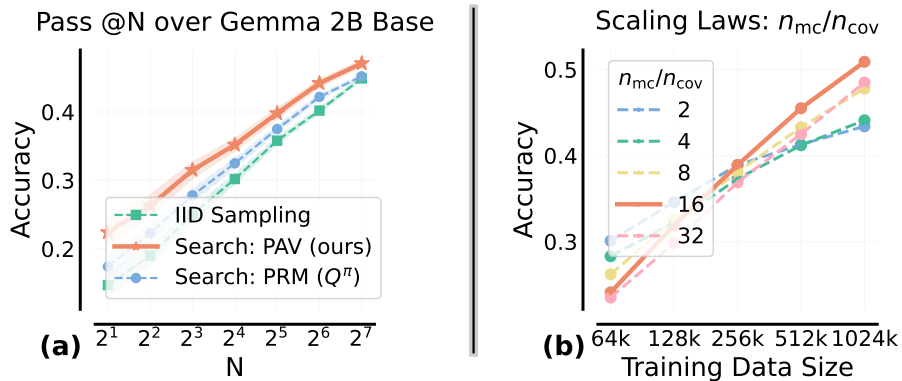


**Figure 5 | Comparing PAVs with search baselines and ablating over the prover policy:** (a): We compare beam search over Gemma 9B SFT, using either effective reward (PAV), or  $Q^\pi$  (Snell et al., 2024), and report best-of-N performance where the re-ranker is either the ORM or PAV-as-ORM. (b): For the base Gemma 2B SFT policy, we run beam search with the effective reward where the prover is  $\text{BoK}(\pi)$  for different values of  $K$ . In both (a), (b) the x-axis scales the size of the beam or  $N$  for best-of- $N$ . (c): For each base policy in the set: Gemma 2B, 9B, 27B policies, we run beam search with PAVs (beam size of 16) where the prover is another policy from the same set.

$A^\mu \approx 0$  on all steps for very large  $K$ , since  $Q^\mu(s_h, a_h) = 1 - (1 - Q^\pi(s_h, a_h))^K \rightarrow 1$  on all steps, as we increase  $K$ . Hence, *in order to succeed we need an intermediate-level prover policy*.

We make similar observations in Figure 5(c) where we use the three base policies (Gemma 2B/9B/27B) as provers for training PAVs. In this scenario, we evaluate beam search with PAVs at  $N = 16$  on top of different base policies. We find that for the 2B and 9B base models, the 9B and 27B provers are most effective respectively, whereas for the 27B model, *surprisingly a weaker 9B policy is more effective than the stronger 27B model*. The weaker model presumably offers a complementary signal that distinguishes between different actions taken by 27B, aligning with our theoretical observations in Sec. 3.4.

**Result 3: Advantages from the prover policy enable exploration.** As discussed in Sec. 3.1, advantage  $A^\mu$  measures the progress made by an action agnostic of the value of the previous state, whereas  $Q^\pi$  measures the promise of a particular state. Given a finite capacity beam, our effective reward (Eq. 5), which linearly combines  $Q^\pi$  and  $A^\mu$  induces a better tradeoff between exploring new prefixes (states) from where progress can be made and exploiting currently known prefixes with high  $Q$ -values. Exploration at



**Figure 6 | (a):** Beam search with PAVs improves exploration efficiency (higher Pass@N), over typical PRMs. (b): Performance of beam search over Gemma 9B SFT for PAVs trained on datasets with different  $n_{mc}/n_{cov}$ .

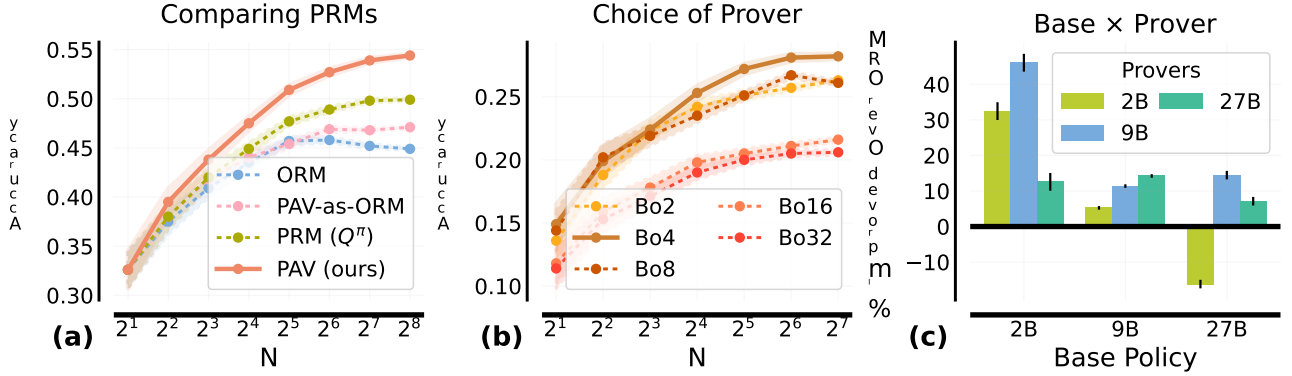


图 5 | **Comparing PAVs with search baselines and ablating over the prover policy:** (a): 我们比较了在 Gemma 9B SFT 上的 beam search, 使用的是有效的奖励 (PAV) 或  $Q^\pi$  (Snell 等, 2024), 并报告了 re-ranker 为 ORM 或 PAV-as-ORM 的最佳性能。 (b): 对于基础的 Gemma 2B SFT 策略, 我们使用有效的奖励进行 beam search, 其中证明者是  $\text{BoK}(\pi)$ , 并对  $K$  的不同值进行测试。在两者 (a), (b) 中, x 轴分别表示 beam 的大小或  $N$  对于 best-of- $N$ 。 (c): 对于集合中的每个基础策略 (Gemma 2B、9B、27B 策略), 我们使用 PAVs (beam 大小为 16) 进行 beam search, 其中证明者是同一集合中的另一个策略。

$A^\mu \approx 0$  在所有步骤中, 对于非常大的  $K$ , 因为  $Q^\mu(s_h, a_h) = 1 - (1 - Q^\pi(s_h, a_h))^K \rightarrow$  在所有步骤中都为 1, 随着我们增加  $K$ 。因此, **in order to succeed we need an intermediate-level prover policy**。

我们在图5(c)中也做出了类似的观察, 其中我们使用三种基础策略 (Gemma 2B/9B/27B) 作为证明者来训练 PAVs。在这种情况下, 我们在不同的基础策略之上评估带有 PAVs 的 16 步的束搜索。我们发现, 对于 2B 和 9B 基础模型, 9B 和 27B 证明者分别最为有效, 而对于 27B 模型, 则是

**surprisingly a weaker 9B policy is more effective than the stronger 27B model**。较弱的模型可能提供了互补的信号, 区分了 27B 采取的不同行动, 这与我们在第 3.4 节中的理论观察一致。

**Result 3:** 证明者政策的优势促进了探索。如第 3.1 节所述, 优势  $A^\mu$  衡量了在不考虑先前状态价值的情况下采取行动所取得的进展, 而  $Q^\pi$  则衡量了特定状态的前景。给定有限容量的束, 我们有效的奖励 (式 5), 这线性结合了  $Q^\pi$  and  $A^\mu$  induces a better tradeoff between exploring new prefixes (states) from where progress can be made and exploiting currently known prefixes with high  $Q$ -values. 探索在

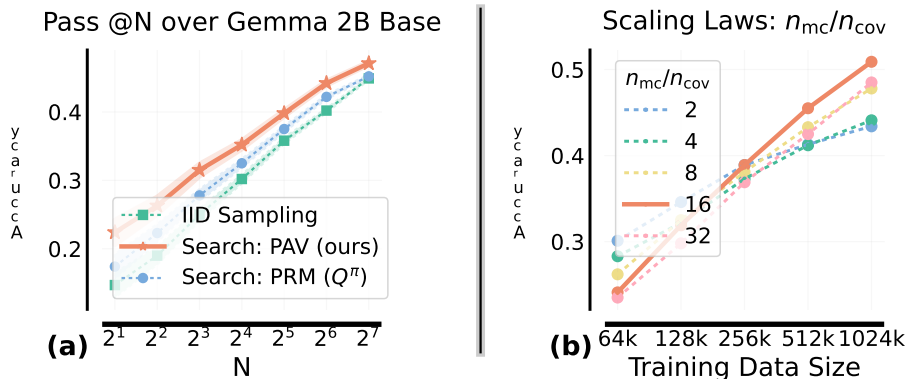


图 6 | (a): 带有 PAVs 的束搜索可以提高探索效率 (更高的 Pass@N), 超过典型的 PRMs。 (b) Performance of beam search over Gemma 9B SFT для PAVs trained on datasets with different  $n_{mc}/n_{cov}$ .

源文本: :翻译文

initial steps is critical to ensure that the beam at later steps covers diverse partial rollouts each with a high likelihood of producing the correct answer. Thus over-committing to the beam with actions from the same state, regardless of the progress made by each can prove to be sub-optimal over a selection strategy that balances rewarding previous actions  $A^\mu$  and current states  $Q^\pi$ . Indeed, we observe in Fig. 6(a), beam search with PAV enhances pass@N performance vs. beam search with  $Q^\pi$  and *i.i.d.* sampling.

**Takeaways: Scaling test-time compute with process advantage verifiers.**

- Beam search with PAVs boosts accuracy by  $>8\%$  & compute efficiency by 1.5-5x over ORMs.
- Utilizing Best-of-K policies (corresponding to the base policy) as provers induce better exploration to maximize outcome reward. Optimal provers for a base policy appear at  $K > 1$ .

#### 4.2. How to Collect Data to Train PAVs?: PAV Training Data Scaling Laws

We now describe the procedure for training outcome verifiers and PAVs. We can learn to predict  $Q^\pi$  for a policy  $\pi$  (similar for  $Q^\mu$ ) by finetuning LLMs with a cross-entropy loss on the following data with triplets  $(s, a, Q_{mc}^\pi(s, a))$ . To collect this data, we first sample  $n_{cov}$  “seed” rollouts from the base or prover policy respectively for ORM and PAVs, to promote coverage over prefixes and steps. Then we sample  $n_{mc}$  additional rollouts, conditioned on each prefix in the seed rollout to compute the Monte-Carlo estimate of  $Q^\pi$  at each prefix. In Fig. 6(b) we plot the beam search performance of PAVs trained with different ratios of  $n_{mc}/n_{cov}$ , as we scale the total dataset size. Here, the beam size is fixed to 128 and the base policy is the Gemma 9B SFT policy and prover is Bo4 policy. **We find that** under low sampling budgets, optimizing for coverage ( $n_{cov} > n_{mc}$ ) is better for performance, and when budget is higher, reducing label noise in  $Q_{mc}^\pi$  by setting  $n_{mc} > n_{cov}$  gets us more improvements. In addition, we also spend some initial sampling budget is spent to identify “high value” states where  $Q^\pi$  is larger than a threshold, and identify the first step with low  $Q^\pi$  on an incorrect partial rollout from this state. We found this strategy to scale better with dataset size, as we discuss in App. D.

### 5. Results: Scaling Dense-Reward RL with PAVs

We can also use PAVs to train policies via online reinforcement learning (RL), by using the effective reward  $Q^\pi + \alpha A^\mu$  as dense, per-step rewards. We compare the sample efficiency of PAV-RL (i.e.,  $\ell_{PAV-RL}$  in Eq. 4) with standard ORM-RL (i.e.,  $\ell_{ORM-RL}$  in Eq. 3) on Gemma 2B and Gemma 9B SFT models, which are further optimized via rejection finetuning (RFT) (Yuan et al., 2023), before using them to initialize RL. To our knowledge, no prior work has successfully demonstrated the use of dense per-step feedback with a process reward model for RL, and we present the first significant set of results establishing the efficacy of this approach. We show that PAV-RL is much more sample-efficient, and enjoys a higher ceiling on the performance of any test-time re-ranker. Additional details for the experiments are in App. E.

**Result 1: PAV-RL is  $> 7\%$  better than ORM-RL in test accuracy, and  $6\times$  sample efficient.** In Fig. 7(a), we report the test accuracies of Gemma 2B and 9B models trained with SFT, RFT, ORM-RL and PAV-RL. PAV-RL improves the RFT policy by 11% for 2B, and 15% for 9B, with  $> 7\%$  gain over ORM-RL in both cases. Not only do the effective rewards from PAV improve the raw accuracy after RL, this higher accuracy is attained  $6\times$  faster (see Fig. 7(b)) for the 2B run and similarly for the 9B RL run (Fig. 7(c)). For both 2B and 9B, RL runs, we experiment with two options for the prover policy: (i) 2B SFT policy; and (ii) 9B SFT policy. While both of these provers rapidly become weaker than the base policy within a few gradient steps of RL, a fixed PAV trained with each of these provers is able to still sustain performance gains in RL. More interestingly, we find that the 2B SFT policy serves as the best choice of the prover for both 2B and



初始步骤对于确保后续步骤的束能够覆盖多样化的部分展开路径，每条路径都有较高的可能性产生正确的答案至关重要。因此，无论每个状态的进展如何，过度承诺于来自同一状态的动作，可能会证明比平衡奖励先前动作 $A^\mu$ 和当前状态 $Q^\pi$ 的选择策略更差。事实上，我们在图6(a)中观察到，使用PAV的束搜索在pass@N性能上优于使用 $Q^\pi$ 和*i.i.d.*采样的束搜索。

- 使用PAVs的\_beam搜索\_提高了准确性\_>8%且计算效率提高了1.5-5倍，超过ORMs。
- 利用最佳-of-K 策略（对应基策略）作为证明者可以更好地探索以最大化结果奖励。基策略的最佳证明者出现在  $K > 1$ 。

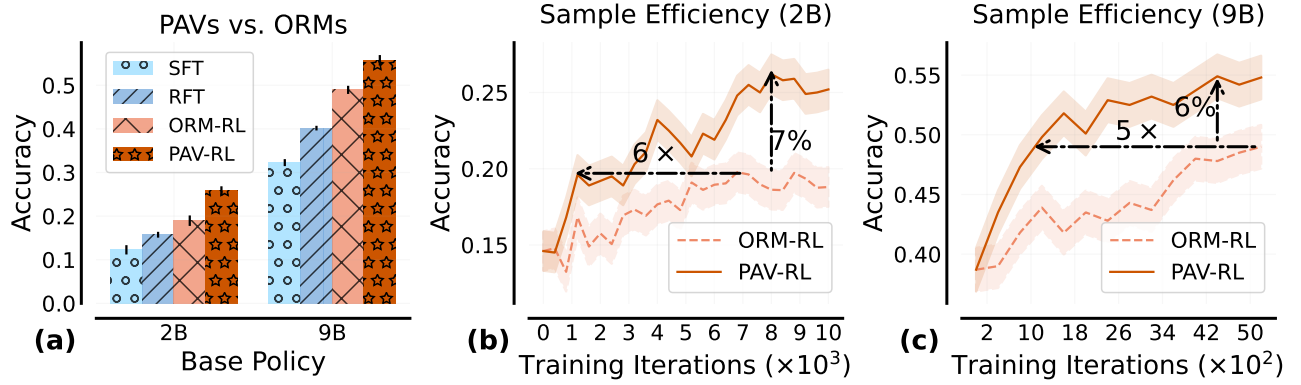
#### 4.2. 如何收集用于训练PAVs的数据？：PAVs的训练数据缩放定律

我们现在描述训练结果验证器和PAVs的过程。我们可以通过使用交叉熵损失对来自以下带有三元组 $(s, a, Q_{mc}^\pi(s, a))$ 的数据进行微调LLMs来学习预测一个政策 $\pi$ （对于 $Q^\mu$ ）类似的 $Q^\pi$ 。为了收集这些数据，我们首先分别从基政策或证明者政策中采样“种子”rollouts，以促进前缀和步骤的覆盖范围。然后，我们根据“种子”rollout中的每个前缀采样额外的rollouts，以计算每个前缀处的蒙特卡洛估计 $Q^\pi$ 。在图6(b)中，我们绘制了使用不同比例 $n_{mc}/n_{cov}$ 训练的PAVs的束搜索性能，随着数据集大小的增加。这里，束大小固定为128，基政策是Gemma 9B SFT政策，证明者是Bo4政策。我们发现，在低采样预算下，优化覆盖范围( $n_{cov} > n_{mc}$ )对性能更好，而在预算较高时，通过设置 $n_{mc} > n_{cov}$ 减少 $Q_{mc}^\pi$ 中的标签噪声可以获得更多的改进。此外，我们还花费了一部分初始采样预算来识别“高价值”状态，其中 $Q^\pi$ 超过了一个阈值，并确定从该状态开始的具有低 $Q^\pi$ 的第一步在不正确的部分rollout中。我们发现，随着数据集大小的增加，这种策略具有更好的扩展性，如我们在附录D中讨论的。

### 5. 结果：使用 PAVs 扩展密集奖励 RL

我们还可以使用PAVs通过在线强化学习（RL）来训练策略，通过使用有效的奖励 $Q^\pi + \alpha A^\mu$ 作为密集的每步奖励。我们在Gemma 2B和Gemma 9B SFT模型上比较了PAV-RL（即Eq. 4中的 $\ell_{PAV-RL}$ ）与标准ORM-RL（即Eq. 3中的 $\ell_{ORM-RL}$ ）的样本效率，这些模型在使用前通过拒绝微调（RFT）进一步优化（Yuan et al., 2023）。在使用它们初始化RL之前。据我们所知，没有先前的工作成功地使用密集的每步反馈与过程奖励模型进行RL，我们展示了这一方法的第一个重要结果，证明了PAV-RL的样本效率更高，并且在任何测试时重新排序器的性能上限更高。实验的更多细节见附录E。

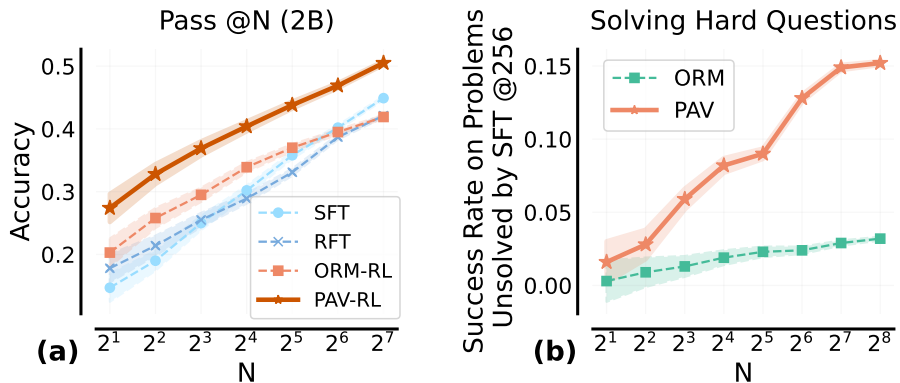
**Result 1:** PAV-RL在测试准确度上比ORM-RL高7%，并且在样本效率上快6×。在图7(a)中，我们报告了使用SFT、RFT、ORM-RL和PAV-RL训练的Gemma 2B和9B模型的测试准确度。PAV-RL将2B的RFT策略提高了11%，9B的提高了15%，并且在两种情况下都比ORM-RL高出7%的增益。不仅有效的奖励从PAV提高了经过RL后的原始准确度，而且这种更高的准确度在图7(b)中的2B运行中6×更快获得，9B的RL运行中也是如此（见图7(c)）。对于2B和9B的RL运行，我们尝试了两种证明者策略的选择：(i) 2B的SFT策略；(ii) 9B的SFT策略。虽然这两种证明者在几个梯度步骤内迅速变得比基础策略更弱，但用这两种证明者训练的固定PAV仍然能够在RL中维持性能增益。更有趣的是，我们发现2B的SFT策略是2B和



**Figure 7 | PAVs as dense rewards in RL improve sample efficiency compared to ORMs, along with gains on raw accuracy:** (a) We report the performance of a base policy trained using RL with effective rewards (PAV-RL), or only outcome rewards (ORM-RL), and baselines SFT, RFT. (b,c): Across training iterations, we report the test performance of policies trained with PAV-RL and ORM-RL, on Gemma 2B and 9B SFT base policies.

9B policies. This observation that a weak prover can still improve the base policy corroborates our results in the didactic setup and our analysis in Sec. 3.4. While we were not able to run experiments where the prover policy is dynamically updated on the fly, we believe that updating the prover through the process of RL training should only amplify these benefits.

**Result 2: PAV-RL achieves higher performance ceiling on test-time re-ranking.** In Fig. 8(a), for Gemma 2B, we plot the Pass @N performance for each method, and find (i) Pass @N is higher ( $> 7\%$ ) for PAV-RL, compared to ORM-RL, for any  $N \leq 128$ ; and (ii) the rate at which Pass @N improves for PAV-RL is higher than ORM-RL. Both trends are consistent with our observations on the didactic example in Sec. 3.3. Notably, for  $N \geq 64$ , ORM-RL is worse than the SFT policy, perhaps due to lower entropy over the distribution at the next step resulting in non-diverse candidates. Why does PAV-RL produce diverse candidates, and does not suffer from the low diversity problem in ORM-RL? We answer this with a key insight on how *the primary benefit of PAVs is to promote efficient exploration*.



**Figure 8 | (a):** For the policies trained in (a) we report the best-of-N performance where the oracle reward Rex is used to rank  $N$  candidates sampled from the base policy (Pass @N). **(b):** Amongst hard problems that remain unsolved by Best-of-256 over the base SFT policy, we check how many are solved by Best-of-N over PAV-RL or ORM-RL. PAV-RL is able to solve a substantially more problems than what ORM-RL was able to solve.

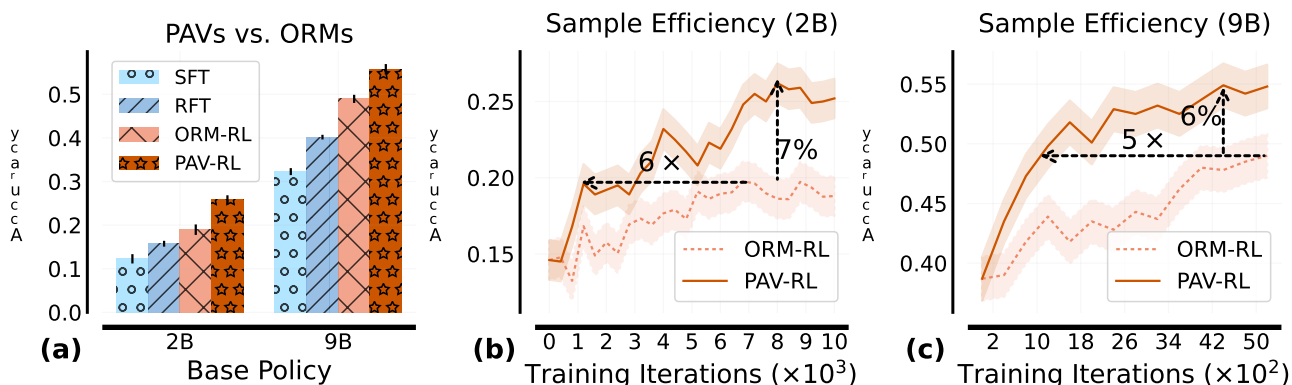


图7 | *PAVs as dense rewards in RL improve sample efficiency compared to ORMs, along with gains on raw accuracy*: (a) 我们报告了使用有效奖励（PAV-RL）或仅结果奖励（ORM-RL）训练的基本策略的性能，以及基线SFT、RFT。(b,c): 在训练迭代过程中，我们报告了使用PAV-RL和ORM-RL训练的策略在Gemma 2B和9B SFT基策略上的测试性能。

9B策略。这一观察表明，一个较弱的证明者仍然可以改进基础策略，这证实了我们在教学设置中的结果以及第3.4节中的分析。虽然我们无法运行证明者策略在实时过程中动态更新的实验，但我们认为通过RL训练过程更新证明者只会增强这些益处。

Result 2: PAV-RL 在测试时重新排序中实现了更高的性能上限。在图 8(a) 中，对于 Gemma 2B，我们绘制了每种方法的 Pass @N 性能，并发现 (i) 对于任何  $N > 128$ ，PAV-RL 的 Pass @N 比 ORM-RL 高 7%；(ii) PAV-RL 的 Pass @N 改进速度高于 ORM-RL。这两种趋势与我们在第 3.3 节的示例观察一致。值得注意的是，对于  $N \leq 64$ ，ORM-RL 的表现不如 SFT 策略，可能是因为下一步分布的熵较低导致候选者不多样化。PAV-RL 为什么能产生多样化的候选者，而不像 ORM-RL 那样遭受低多样性的问题？我们通过一个关键洞察来回答这个问题：

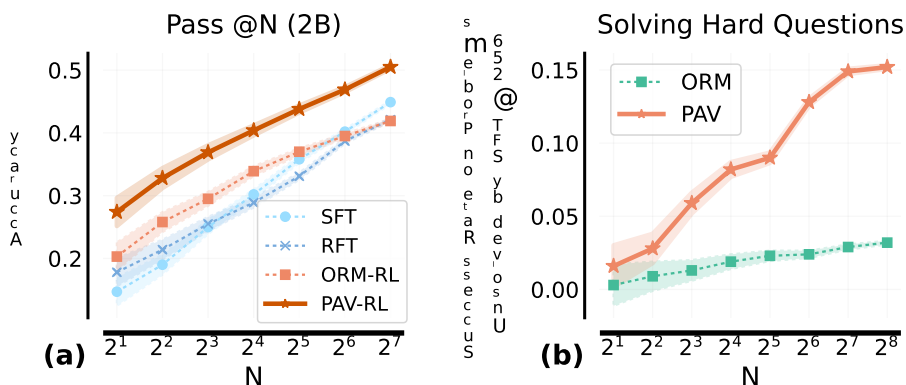


图 8 | (a): 对于在 (a) 中训练的策略，我们报告使用 oracle 奖励 Rex 排序的基策略采样的  $N$  个候选者中的最佳性能 (Pass @N)。 (b): 在 Best-of-256 无法解决的硬问题中，我们检查在 PAV-RL 或 ORM-RL 上使用 Best-of-N 能解决多少问题。PAV-RL 比 ORM-RL 能解决更多的问题。

**Result 3: PAVs improve exploration and discover correct solutions to novel problems.** An outcome reward model rewards downweight all steps in an incorrect rollout equally during RL, whereas the effective reward  $Q^\pi + \alpha A^\mu$  in PAVs, up-weights steps that make progress under the prover, even when the complete rollout is incorrect. This increases the coverage over individual steps that can improve the likelihood of the base policy to succeed (since the prover policy is not too misaligned with the base policy). These can now be proposed by the base policy at a given prefix. This mechanism for exploration is analogous to test-time search we discussed in Sec. 4.1. Hence, the directed supervision from PAVs improves sample-efficiency throughout the course of training (Fig. 7(c)). In fact, we also find that combining the PAV-RL policy with test-time beam search is able to solve a substantially larger number of *new* problems within smaller compute budgets ( $N = 16, 32$ ) that the SFT policy cannot solve with a much larger budget  $N = 256$  (Fig. 8(b)).

**Takeaway: RL with process advantage verifiers (PAVs) as dense rewards**

- Using trained PAVs as dense rewards in RL boosts scales sample efficiency by 5 – 6×, compared to only using sparse ORM rewards, and results in policies with a higher Pass @N performance.
- Advantages from a complementary prover policy improves the sample efficiency of exploration in RL, and produces policies that can discover solutions to hard novel questions.

## 6. Related Work

We briefly discuss some key related works here, and leave the detailed discussion for App. A. To address issues of sparse feedback in ORMs (Cobbe et al., 2021b), recent works (Lightman et al., 2023; Uesato et al., 2022) trained process reward models (PRMs) to densely predict incorrect steps in a multi-step reasoning trace. Since human data collection for process labels is not scalable enough, recent work (Luo et al., 2024; Wang et al., 2024) used automated supervision to annotate steps with  $Q$  values under the base policy, *i.e.*, the PRMs score a step with the likelihood of future success, when continuing to sample from the step. While  $Q$ -value PRMs in Lightman et al. (2023); Luo et al. (2024) were mainly used as verifiers for re-ranking, Snell et al. (2024) used them for test-time beam search. Shao et al. (2024) uses PRMs for RL but found a gain of only 1 – 2% with PRMs. In our work, we question solely relying on  $Q$ -values or advantages of the base policy, and find that measuring progress (*i.e.*, advantages) under a different prover policy can amplify exploration, thus boosting test-time search and RL. To our knowledge, we are the first to show substantial gains in compute and sample efficiency with PRMs. Our methodology for data collection is similar to (Hwang et al., 2024; Setlur et al., 2024) (*i.e.*, identify “first pits” in reasoning traces), these works only use it to collect preference pairs. Beyond all of these, we also characterize which policy to use for computing advantages.

Concurrently to us, akin to the methodology in Hwang et al. (2024); Setlur et al. (2024), Kazemnejad et al. (2024) optimize the base policy  $\pi$  with online RL, where the dense step-level rewards correspond to advantages  $A^\pi$  under the base policy  $\pi$  itself. This is a special case of our setting, where the prover policy  $\mu = \pi$ , but as we note in Sec. 3.2, setting  $\mu = \pi$  in our effective reward (Eq. 5) results in exactly the same policy gradient updates as only optimizing the outcome reward. Since Kazemnejad et al. (2024) use “on-the-fly” Monte-Carlo rollout estimation to estimate advantages, they are able to avoid estimation errors in the process reward model. Nonetheless, our theoretical result and the didactic example (both of which assume access to perfect advantage estimates) show that gains from this approach are significantly smaller than using an appropriate prover policy, which is distinct from the base policy.



**Result 3:** PAVs 提高了探索能力，并在遇到新问题时发现正确的解决方案。结果奖励模型在 RL 中会同削弱所有错误展开中的步骤，而 PAVs 中的有效奖励  $Q^\pi + \alpha A^\mu$  则会增强在证明器下取得进展的步骤的权重，即使整个展开是错误的。这增加了单个步骤的覆盖率，这些步骤可以提高基础策略成功的可能性（因为证明器策略与基础策略的偏差不大）。这些步骤现在可以由给定前缀的基础策略提出。这种探索机制类似于我们在第 4.1 节中讨论的测试时搜索。因此，来自 PAVs 的有向监督在整个训练过程中提高了样本效率（图 7(c)）。事实上，我们还发现，将 PAV-RL 策略与测试时的束搜索结合使用，能够在较小的计算预算（ $N = 16, 32$ ）内解决更多的 *new* 问题，而 SFT 策略在更大的预算（ $N = 256$ ）内也无法解决这些问题（图 8(b)）。

- 使用训练好的 PAVs 作为密集奖励在 RL 中提升样本效率，与仅使用稀疏 ORM 奖励相比，提升了 5–6 倍，并且导致了具有更高 Pass @N 性能的策略。
- 从互补证明器政策的优势提高了在 RL 中探索的样本效率，并产生了能够发现难题新问题解决方案的策略。

## 6. 相关工作

我们简要讨论一些相关工作，详细的讨论留到附录 A 中。为了应对 ORM 中的稀疏反馈问题（Cobbe 等，2021b），近期的工作（Lightman 等，2023；Uesato 等，2022）训练过程奖励模型（PRM）以密集预测多步推理跟踪中的错误步骤。由于人工收集过程标签的数据量不够大，近期的工作（Luo 等，2024；Wang 等，2024）使用自动监督在基策略  $Q$  下标注步骤，基策略记为 *i.e.*，PRM 会根据从该步骤继续采样的未来成功的可能性来评估步骤。Lightman 等（2023）和 Luo 等（2024）中的  $Q$  值 PRM 主要用于重新排序的验证，而 Snell 等（2024）则用于测试时的束搜索。Shao 等（2024）使用 PRM 进行 RL，但发现使用 PRM 仅获得了不到 2% 的收益。在我们的工作中，我们质疑仅仅依赖  $Q$  值或基策略的优势，并发现使用不同的证明策略来衡量进步（即优势）可以增强探索，从而提升测试时的搜索和 RL。据我们所知，我们是第一个展示使用 PRM 在计算和样本效率方面取得显著收益的研究。我们的数据收集方法类似于（Hwang 等，2024；Setlur 等，2024）（即识别推理跟踪中的“第一个坑”），这些工作仅将其用于收集偏好对。除此之外，我们还探讨了用于计算优势的策略。

同时，类似于 Hwang 等人（2024）；Setlur 等人（2024），Kazemnejad 等人（2024）的方法，他们使用在线 RL 来优化基础策略  $\pi$ ，其中密集的步骤级奖励对应于基础策略  $\pi$  本身的优势  $A^\pi$ 。这是我们的设置的一个特殊情况，其中证明者策略为  $\mu = \pi$ ，但如我们在第 3.2 节中所指出的，我们有效奖励（式 5）中的设置  $\mu = \pi$  导致与仅优化结果奖励相同的策略梯度更新。由于 Kazemnejad 等人（2024）使用“实时”的蒙特卡洛展开估计来估计优势，因此他们在过程奖励模型中避免了估计误差。然而，我们的理论结果和教学示例（两者都假设可以获取完美的优势估计）表明，这种方法的收益远小于使用适当的证明者策略，而该证明者策略不同于基础策略。



## 7. Discussion and Conclusion

We began our exposition with the following question: how to define process rewards such that optimizing the base policy against process rewards ultimately improves the outcome level correctness of the final answer? Our key finding is that process rewards defined as advantages of a prover policy, distinct from the base policy improve the efficiency of exploration for steps sampled from the base policy during test-time search and online RL. This improved exploration in turn leads to discovery of better solutions, resulting in a higher accuracy on the math reasoning task. We also formally characterized the set of good prover policies as policies with step-level advantages that meaningfully contrast steps generated by the base policy, while still producing step-level advantages that are aligned with the base policy. Having trained process advantage verifiers (PAVs) to predict advantages under the prover policy, we empirically observed that test-time search against the trained PAVs improve the compute-efficiency of search by  $1.5 - 5\times$ , and accuracy of search by over 8% compared to running best-of- $N$  against an ORM. Next, we present one of the significant results that validate the use of dense supervision when optimizing the base policy with online RL. Specifically, we show that dense online RL with rewards from our trained PAVs, improves sample efficiency of online RL by  $5 - 6\times$ , and results in an accuracy gain of over 6%.

**Limitations.** Despite the promise of our results, there are several limitations to our work that present important avenues for future research. First, while we can easily compute the right hand side of our result in Theorem 3.1 to understand whether a given prover policy will improve a fixed base policy, it is unclear how to automatically design a flexible class of optimal (or very good) prover policies for a sequence of base policy iterates. Perhaps simultaneously optimizing the prover and the base policy (in a two-player game) might provide for an approach to obtain the best prover during RL, but this is largely an open question. Second, since inevitably learning a process advantage verifier (PAV) will incur fitting errors and this upper bounds performance of our method. Fitting errors can be circumvented if our approach if we can simply run rollouts from prover policies during online RL or search to estimate advantages without training verifiers, and extending our approach to this setup is a good avenue for future work.

## Acknowledgements

The authors would like to thank Charlie Snell, Yi Su, Katherine Heller, and Virginia Smith for feedback on an earlier version of this paper. We also thank Ahmad Beirami, Sergey Levine, Victor Veitch, Idan Shenfeld, Arian Hosseini, Stephen Pfohl, Xiangyu Qi, Tianhe Yu, and Christina Baek for technical discussions. AS and CN also thank Preston Robinette, Sho Kannan, Tianze Shi, Diana Mincu, Hritik Bansal, and Liangchen Luo for code, infrastructure and data analytics support.

## References

- A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research*, 22(98):1–76, 2021.
- T. Anthony, Z. Tian, and D. Barber. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.
- M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

## 7. 讨论与结论

我们从以下问题开始了我们的阐述：如何定义过程奖励，使得优化基础策略针对过程奖励最终提高最终答案的正确性水平？我们的主要发现是，将过程奖励定义为证明者策略的优势，与基础策略不同，这在测试时搜索和在线RL中从基础策略采样的步骤中提高了探索效率。这种改进的探索反过来导致了更好的解决方案的发现，从而提高了数学推理任务的准确性。我们还正式定义了良好的证明者策略的集合，这些策略在步骤级别上的优势与基础策略生成的步骤有实质性的对比，同时仍然保持与基础策略对齐的步骤级别优势。通过训练过程优势验证器（PAVs）来预测证明者策略下的优势，我们实证观察到，针对训练好的PAVs的测试时搜索提高了搜索的计算效率 $1.5 - 5\times$ ，并且相比使用ORM的best-of-N提高了超过8%的准确性。接下来，我们呈现了一个重要的结果，验证了在使用在线RL优化基础策略时使用密集监督的有效性。具体来说，我们展示了使用我们训练好的PAVs的密集在线RL提高了在线RL的样本效率 $5 - 6\times$ ，并且带来了超过6%的准确性提升。

限制。尽管我们的结果充满希望，但我们的工作仍存在一些限制，这些限制为未来的研究指出了重要的方向。首先，虽然我们可以轻松地计算定理3.1结果的右侧，以了解给定的证明者策略是否会改善固定的基础策略，但不清楚如何自动设计一个灵活的最优（或非常好的）证明者策略类，以适应一系列基础策略迭代。也许同时优化证明者和基础策略（在一个两玩家游戏中）可能会提供一种在强化学习中获得最佳证明者的途径，但这仍然是一个开放的问题。其次，由于不可避免地学习过程优势验证器（PAV）会产生拟合误差，这会限制我们方法的性能。如果我们可以在线RL或搜索中从证明者策略运行滚动测试来估计优势而不训练验证器，就可以避免拟合误差，将我们的方法扩展到这种设置是一个值得未来研究的方向。

## Acknowledgements

作者们要感谢Charlie Snell、Yi Su、Katherine Heller和Virginia Smith在本文早期版本上的反馈。我们还感谢Ahmad Beirami、Sergey Levine、Victor Veitch、Idan Shenfeld、Arian Hosseini、Stephen Pfohl、Xian gyu Qi、Tianhe Yu和Christina Baek在技术讨论方面的支持。AS和CN还感谢Preston Robinette、Sho Kannan、Tianze Shi、Diana Mincu、Hritik Bansal和Liangchen Luo在代码、基础设施和数据分析支持方面的帮助。

## 参考文献

- A. Agarwal, S. M. Kakade, J. D. Lee, 和 G. Mahajan. 关于政策梯度方法的理论：最优性、逼近和分布偏移 *Journal of Machine Learning Research*, 22(98):1–76, 2021.
- T. Anthony, Z. Tian, 和 D. Barber. 快速思考与缓慢思考：深度学习与树搜索的方法。 *Advances in neural information processing systems*, 30, 2017.
- M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, 和 R. Munos. 统一基于计数的探索和内在动机。在 *Advances in Neural Information Processing Systems*, 第 1471–1479 页, 2016。

- X. Bi, D. Chen, G. Chen, S. Chen, D. Dai, C. Deng, H. Ding, K. Dong, Q. Du, Z. Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.
- J. D. Chang, K. Brantley, R. Ramamurthy, D. Misra, and W. Sun. Learning to generate better than your llm. *arXiv preprint arXiv:2306.11816*, 2023.
- K.-W. Chang, A. Krishnamurthy, A. Agarwal, H. Daumé III, and J. Langford. Learning to search better than your teacher. In *International Conference on Machine Learning*, pages 2058–2066. PMLR, 2015.
- K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021a.
- K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021b.
- M. Collins. Discriminative reranking for natural language parsing. In *Proceedings of the International Conference on Machine Learning*, 2000.
- Gemma Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.
- M. Germain, K. Gregor, I. Murray, and H. Larochelle. Made: Masked autoencoder for distribution estimation. In *International conference on machine learning*, pages 881–889. PMLR, 2015.
- A. Havrilla, Y. Du, S. C. Raparthy, C. Nalmpantis, J. Dwivedi-Yu, M. Zhuravinskiy, E. Hambro, S. Sukhbaatar, and R. Raileanu. Teaching large language models to reason with reinforcement learning. *arXiv preprint arXiv:2403.04642*, 2024.
- D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.
- G. Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- A. Hosseini, X. Yuan, N. Malkin, A. Courville, A. Sordoni, and R. Agarwal. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.
- H. Hwang, D. Kim, S. Kim, S. Ye, and M. Seo. Self-explore to avoid the pit: Improving the reasoning capabilities of language models with fine-grained rewards. *arXiv preprint arXiv:2404.10346*, 2024.
- S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 267–274, 2002.
- S. M. Kakade. A natural policy gradient. In *Advances in neural information processing systems*, volume 14. Advances in neural information processing systems, 2001a.
- S. M. Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001b.
- A. Kazemnejad, M. Aghajohari, E. Portelance, A. Sordoni, S. Reddy, A. Courville, and N. L. Roux. Vineppo: Unlocking rl potential for llm reasoning through refined credit assignment. *arXiv preprint arXiv:2410.01679*, 2024.

- X. Bi, D. Chen, G. Chen, S. Chen, D. Dai, C. Deng, H. Ding, K. Dong, Q. Du, Z. Fu, 等. Deepseek llm: 使用长期主义扩展开源语言模型. *arXiv preprint arXiv:2401.02954*, 2024.
- J. D. 张, K. 布兰特利, R. 瑞马穆尔thy, D. 迈斯拉, 和 W. 孙. 学习生成优于你的LLM. *arXiv preprint arXiv:2306.11816*, 2023.
- K.-W. Chang, A. Krishnamurthy, A. Agarwal, H. Daumé III, 和 J. Langford. 通过学习搜索超越你的老师. In *International Conference on Machine Learning*, 第2058–2066页. PMLR, 2015. K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, 和 J. Schulman. 训练验证器解决数学文字问题. *arXiv preprint arXiv:2110.14168*, 2021a. K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. 训练验证器解决数学文字问题. *arXiv preprint arXiv:2110.14168*, 2021b. M. Collins. 自 discriminative 重排序对自然语言解析的影响. In *Proceedings of the International Conference on Machine Learning*, 2000. Gemma Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, et al. Gemma: 基于gemini研究和技术的开放模型. *arXiv preprint arXiv:2403.08295*, 2024.
- M. Germain, K. Gregor, I. Murray, 和 H. Larochelle. Made: 遮盖自编码器用于分布估计. 在 *International conference on machine learning*, 页码 881–889. PMLR, 2015.
- A. Havrilla, Y. Du, S. C. Raparthy, C. Nalmpantis, J. Dwivedi-Yu, M. Zhuravinskyi, E. Hambro, S. Sukhbaatar, and R. Raileanu. 教授大型语言模型使用强化学习进行推理. *arXiv preprint arXiv:2403.04642*, 2024.
- D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, 和 J. Steinhardt. 用 Math 数据集衡量数学问题解决能力. *NeurIPS*, 2021.
- G. Hinton. 在神经网络中提炼知识. *arXiv preprint arXiv:1503.02531*, 2015.
- A. Hosseini, X. Yuan, N. Malkin, A. Courville, A. Sordoni, 和 R. Agarwal. V-star: 训练验证器以供自学推理器使用. *arXiv preprint arXiv:2402.06457*, 2024. H. Hwang, D. Kim, S. Kim, S. Ye, 和 M. Seo. 自我探索以避免陷阱: 通过细粒度奖励提高语言模型的推理能力. *arXiv preprint arXiv:2404.10346*, 2024. S. Kakade 和 J. Langford. 近似最优近似强化学习. 在 *Proceedings of the Nineteenth International Conference on Machine Learning*, 页码 267–274, 2002. S. M. Kakade. 自然策略梯度. 在 *Advances in neural information processing systems*, 卷 14. 神经信息处理系统进展, 2001a. S. M. Kakade. 自然策略梯度. *Advances in neural information processing systems*, 14, 2001b. A. Kazemnejad, M. Aghajohari, E. Portelance, A. Sordoni, S. Reddy, A. Courville, 和 N. L. Roux. Vineppo: 通过精细的功劳分配解锁 llm 推理的 rl 潜力. *arXiv preprint arXiv:2410.01679*, 2024.

- H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.
- L. Luo, Y. Liu, R. Liu, S. Phatale, H. Lara, Y. Li, L. Shu, Y. Zhu, L. Meng, J. Sun, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.
- Q. Ma, H. Zhou, T. Liu, J. Yuan, P. Liu, Y. You, and H. Yang. Let’s reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*, 2023.
- R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.
- A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.
- L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.
- R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.
- S. Ross and J. A. Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.
- A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.
- A. Setlur, S. Garg, X. Geng, N. Garg, V. Smith, and A. Kumar. Rl on incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold. *arXiv preprint arXiv:2406.14532*, 2024.
- Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- A. Singh, J. D. Co-Reyes, R. Agarwal, A. Anand, P. Patil, P. J. Liu, J. Harrison, J. Lee, K. Xu, A. Parisi, et al. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023a.
- I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Prog-prompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023b.
- C. Snell, J. Lee, K. Xu, and A. Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.
- W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell. Deeply aggravated: Differentiable imitation learning for sequential prediction. In *International conference on machine learning*, pages 3309–3318. PMLR, 2017.



H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, 和 K. Cobbe. 让我们逐步验证。 *arXiv preprint arXiv:2305.20050*, 2023. L. Luo, Y. Liu, R. Liu, S. Phatale, H. Lar a, Y. Li, L. Shu, Y. Zhu, L. Meng, J. Sun, 等. 通过自动化过程监督提高语言模型的数学推理能力。 *arXiv preprint arXiv:2406.06592*, 2024. Q. Ma, H. Zhou, T. Liu, J. Yuan, P. Liu, Y. You, 和 H. Yang. 逐步奖励: 作为推理导航器的步骤级奖励模型。 *arXiv preprint arXiv:2310.10080*, 2023. R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, 等. Webgpt: 带有人类反馈的浏览器辅助问答。 *arXiv preprint arXiv:2112.09332*, 2021. A. Y. Ng, D. Harada, 和 S. Russell. 在奖励转换下策略不变性: 理论及其在奖励塑造中的应用。在 *ICML*, 卷 99, 页码 278–287, 1999 年. L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, 等. 使用人类反馈训练语言模型遵循指令。 *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022. R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, 和 C. Finn. 直接偏好优化: 你的语言模型实际上是一个奖励模型。 *arXiv preprint arXiv:2305.18290*, 2023. S. Ross 和 J. A. Bagnell. 通过交互式无遗憾学习进行强化学习和模仿学习。 *arXiv preprint arXiv:1406.5979*, 2014. A. A. Rusu, S. G. Colmenar ejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, 和 R. Hadsell. 策略蒸馏。 *arXiv preprint arXiv:1511.06295*, 2015. A. Setlur, S. Garg, X. Geng, N. Garg, V. Smith, 和 A. Kumar. 在错误的合成数据上进行 RL 提高了语言模型数学推理的效率八倍。 *arXiv preprint arXiv:2406.14532*, 2024. Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. Li, Y. Wu, 和 D. Guo. Deepseekmath: 在开放语言模型中推动数学推理的极限。 *arXiv preprint arXiv:2402.03300*, 2024. A. Singh, J. D. Co-Reyes, R. A garwal, A. Anand, P. Patil, P. J. Liu, J. Harrison, J. Lee, K. Xu, A. Parisi, 等. 超越人类数据: 通过自我训练扩展语言模型解决问题的规模。 *arXiv preprint arXiv:2312.06585*, 2023a。

I. 珊吉, V. 布卢基斯, A. 摩苏维安, A. 吉oyal, D. 许, J. 特伦布莱, D. 福克斯, J. 汤马森, 和 A. 加尔. Prog-提示: 使用大型语言模型生成情境化机器人任务计划。在 *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 页码 11523–11530. IEEE, 2023b.

C. Snell, J. Lee, K. Xu, 和 A. Kumar. 在测试时按最优方式扩展LLM计算能力比扩展模型参数更为有效。 *arXiv preprint arXiv:2408.03314*, 2024. W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, 和 J. A. Bagnell. Deeply aggravated: 可微imitation学习在序列预测中的应用。In *International conference on machine learning*, 页码 3309–3318. PMLR, 2017.

- R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. The MIT Press, second edition, 2018.
- R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.
- F. Tajwar, A. Singh, A. Sharma, R. Rafailov, J. Schneider, T. Xie, S. Ermon, C. Finn, and A. Kumar. Preference Fine-Tuning of LLMs Should Leverage Suboptimal, On-Policy Data, 2024.
- J. Uesato, N. Kushman, R. Kumar, F. Song, N. Siegel, L. Wang, A. Creswell, G. Irving, and I. Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.
- P. Wang, L. Li, Z. Shao, R. X. Xu, D. Dai, Y. Li, D. Chen, Y. Wu, and Z. Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024.
- R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.
- Y. Wu, Z. Sun, S. Li, S. Welleck, and Y. Yang. An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.
- F. Yu, A. Gao, and B. Wang. Outcome-supervised verifiers for planning in mathematical reasoning. *arXiv preprint arXiv:2311.09724*, 2023.
- Z. Yuan, H. Yuan, C. Li, G. Dong, C. Tan, and C. Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.
- E. Zelikman, Y. Wu, J. Mu, and N. Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.
- L. Zhang, A. Hosseini, H. Bansal, M. Kazemi, A. Kumar, and R. Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.

R. S. 塞顿和A. G. 巴托. *Reinforcement learning: An introduction*. MIT出版社, 第二版 2018年。

源文本: ,译文

R. S. 塞顿, D. 麦卡利斯特, S. 芒索尔, 和 Y. 曼索. 使用函数近似的强化学习策略梯度方法. *Advances in neural information processing systems*, 12, 1999. F. 塔贾瓦尔, A. 芒索尔, A. 拉玛, R. 拉法伊洛夫, J. 施耐德, T. 谢, S. 埃蒙, C. 芬, 和 A. 库马尔. 偏好微调的大型语言模型应利用次优的、在线策略数据, 2024. J. 乌塞托, N. 库什曼, R. 科马鲁, F. 宋, N. 西格尔, L. 王, A. 克雷尔, G. 伊里, 和 I. 希恩斯. 使用过程和结果反馈解决数学文字问题. *arXiv preprint arXiv:2211.14275*, 2022. P. 王, L. 李, Z. 尚, R. X. 许, D. 戴, Y. 李, D. 陈, Y. 吴, 和 Z. 谢. 数学牧羊人: 逐步验证和强化大型语言模型无需人工注释, 2024. R. J. 威廉姆斯. 简单的统计梯度跟随算法用于连接主义强化学习. *Machine learning*, 8(3-4):229–256, 1992. Y. 吴, Z. 孙, S. 李, S. 威尔克, 和 Y. 杨. 语言模型用于解决问题的计算最优推理的实证分析. *arXiv preprint arXiv:2408.00724*, 2024. F. 于, A. 高, 和 B. 王. 结果监督验证器在数学推理中的规划. *arXiv preprint arXiv:2311.09724*, 2023. Z. 袁, H. 袁, C. 李, G. 东, C. 田, 和 C. 周. 大型语言模型学习数学推理的扩展关系. *arXiv preprint arXiv:2308.01825*, 2023. E. 谢尔金曼, Y. 吴, J. 谟, 和 N. 戈登. 星: 通过推理进行推理的自举. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022. L. 张, A. 潘萨尼, H. 巴纳尔, M. 卡泽米, A. 库马尔, 和 R. 阿加瓦尔. 生成验证器: 奖励建模作为下一个标记预测. *arXiv preprint arXiv:2408.15240*, 2024.

# Appendices

## A. Additional Related Work

In this section, we highlight works from four relevant streams, expanding on discussion in Section 6. First, we look at works that train verifiers to provide outcome level feedback (Cobbe et al., 2021b; Hosseini et al., 2024; Singh et al., 2023b; Zelikman et al., 2022) on the correctness of the full response (ORM). Here, the trained ORMs are mainly used for test-time search (best-of- $N$ ). Next, we look at works that alleviate issues with sparse feedback in ORMs, and instead train process reward models (PRMs), that can perform credit assignment. PRMs are trained either through human annotations (Lightman et al., 2023; Uesato et al., 2022), or automated forms of supervision (Luo et al., 2024; Snell et al., 2024; Wang et al., 2024). While some works use PRMs and ORMs to collect data for supervised fine-tuning Hosseini et al. (2024) or offline RL Setlur et al. (2024), other works directly use them as rewards in online RL (Shao et al., 2024; Uesato et al., 2022; Wang et al., 2024). Finally, we contrast our work against papers on imitating stronger teacher policies via RL objectives that optimize potential functions of teacher policies.

**Outcome reward models.** ORMs are verifiers (Cobbe et al., 2021b; Uesato et al., 2022) commonly used to improve the test-time performance using best-of- $N$ , where we generate multiple candidate solutions from the base policy (LLM), rank them using the ORM, and pick the best one. ORMs are trained to assess correctness of a solution either using binary classification (Cobbe et al., 2021a; Yu et al., 2023), preference optimization using DPO (Hosseini et al., 2024), or next-token prediction (Zhang et al., 2024). Furthermore, prior works train LLMs on self-generated data using ground-truth outcome rewards (Rex), either via supervised fine-tuning (Singh et al., 2023a; Yuan et al., 2023; Zelikman et al., 2022), or online RL (Bi et al., 2024). In contrast to these approaches, our work focuses on process reward models (PRMs) for improving performance with beam-search at test time as well as online RL where we maximize the effective reward in Eq. 5 which linearly combines both Rex (outcome supervision) and process supervision in the form of advantages  $A^\mu$  under a prover policy  $\mu$ .

**PRMs and credit assignment.** Several works focus on training step-level PRMs on math reasoning tasks, either using human labels (Lightman et al., 2023) or automated LLM-generated data to estimate value functions  $Q^\pi$  (Luo et al., 2024; Wang et al., 2024). Our work also focus on automated data collection for PRMs but empirically argues for using the advantage function  $A^\mu$  as step-level rewards along with  $Q^\pi$ , with a conceptual explanation in Section 3.1. Several prior works have explored step-level search algorithms with PRMs, such as beam search (Snell et al., 2024), heuristic greedy search (Ma et al., 2023), and reward-balanced tree search (Wu et al., 2024). Hwang et al. (2024); Setlur et al. (2024) use advantages to identify the “first pit” in an incorrect reasoning trace. Specifically, they collect data by computing advantages at each step using Monte Carlo rollouts. Then in an incorrect trace, they identify the step with the least advantage, and use the prefix of that step to construct preference pairs for offline direct preference optimization (Rafailov et al., 2023). In contrast, our work computes advantages under a prover policy, that we formally characterize, and use the computed advantages for improving test-time search and efficiency of online reinforcement learning.

**Online RL for math reasoning.** Once we have a trained outcome or process verifiers, it is natural update a policy by optimizing it against the learned signal, similar to how learned reward models are optimized in RLHF (Ouyang et al., 2022). In the context of math reasoning, Havrilla et al. (2024); Shao et al. (2024); Uesato et al. (2022) trained policies with RL, experimenting with both dense and sparse

# 附录

## A. 附加的相关工作

在本节中，我们突出显示了四个相关流派中的作品，扩展了第6节中的讨论。首先，我们研究了训练验证器以提供结果级反馈的作品（Cobbe et al., 2021b; Hosseini et al., 2024; Singh et al., 2023b; Zelikman et al., 2022），这些作品针对完整响应（ORM）的正确性。在这里，训练好的ORM主要在测试时用于搜索（best-of- $N$ ）。接下来，我们研究了缓解ORM中稀疏反馈问题的作品，并训练了可以进行责任分配的过程奖励模型（PRMs）。PRMs可以通过人工注释（Lightman et al., 2023; Uesato et al., 2022）或自动形式的监督（Luo et al., 2024; Snell et al., 2024; Wang et al., 2024）进行训练。虽然有些作品使用PRMs和ORMs来收集监督微调的数据（Hosseini et al., 2024）或离线RL（Setlur et al., 2024），其他作品则直接将它们用作在线RL中的奖励（Shao et al., 2024; Uesato et al., 2022; Wang et al., 2024）。最后，我们将我们的工作与通过RL目标优化教师策略潜在函数的模仿更强教师策略的论文进行了对比。

**Outcome 奖励模型。**ORM 是验证器（Cobbe 等人，2021b; Uesato 等人，2022），常用于通过 best-of- $N$  提高测试时的性能。我们从基础策略（LLM）生成多个候选解决方案，使用 ORM 对它们进行排名，并选择最佳的一个。ORM 被训练来评估解决方案的正确性，方法包括二元分类（Cobbe 等人，2021a; Yu 等人，2023）、使用 DPO 的偏好优化（Hosseini 等人，2024）或下一个标记预测（Zhang 等人，2024）。此外，先前的工作通过使用真实结果奖励（Rex）自动生成数据来训练 LLM，方法包括监督微调（Singh 等人，2023a; Yuan 等人，2023; Zelikman 等人，2022）或在线 RL（Bi 等人，2024）。与这些方法不同，我们的工作集中在过程奖励模型（PRMs）上，以通过测试时的 beam-search 和在线 RL 提高性能，我们最大化了式 5 中的有效奖励，该奖励线性结合了 Rex（结果监督）和过程监督（优势  $A^\mu$ ）的形式，在证明者策略  $\mu$  下。

**PRMs和信用分配。**一些研究工作集中在使用数学推理任务训练步骤级PRMs上，要么使用人工标签（Lightman等，2023），要么使用自动化的LLM生成数据来估计价值函数 $Q^\pi$ （Luo等，2024; Wang等，2024）。我们的工作也关注自动收集PRMs的数据，但实证上主张使用优势函数 $A^\mu$ 作为步骤级奖励，同时使用 $Q^\pi$ ，并在第3.1节中给出概念性解释。一些先前的工作已经探索了使用PRMs的步骤级搜索算法，如束搜索（Snell等，2024）、启发式贪婪搜索（Ma等，2023）和奖励平衡树搜索（Wu等，2024）。Hwang等（2024）; Setlur等（2024）使用优势来识别错误推理轨迹中的“第一个坑”。具体来说，他们通过计算每一步的蒙特卡洛展开来收集数据。然后在错误轨迹中，他们识别出优势最少的那一步，并使用该步骤的前缀来构建偏好的对，用于离线直接偏好优化（Rafailov等，2023）。相比之下，我们的工作证明者策略下计算优势，并正式刻画了这种策略，然后使用计算出的优势来改进测试时的搜索和在线强化学习的效率。

**在线数学推理的RL。**一旦我们有了训练好的结果或过程验证器，自然地可以通过优化策略来更新它，使其与学习到的信号相匹配，类似于在RLHF（Ouyang等人，2022年）中优化学习到的奖励模型的方式。在数学推理的背景下，Havrilla等人（2024年）; Shao等人（2024年）; Uesato等人（2022年）使用RL训练策略，并尝试了稠密和稀疏



Planted sub-sequence  $y^*$  : [3 6 0 3 5]

Samples from policy corresponding to  $\gamma = 15$

sample=[	3	6	0	3	5	14	14	14	14	14]	reward=1.0
sample=[	3	6	0	4	0	3	1	11	7	3]	reward=0.0
sample=[	3	6	0	3	5	14	13	3	2	11]	reward=1.0
sample=[	3	6	0	13	5	3	1	3	14	1]	reward=0.0
sample=[	12	8	3	6	14	3	6	0	3	5]	reward=0.0

Samples from policy corresponding to  $\gamma = 100$

sample=[	3	6	0	3	5	14	14	14	14	14]	reward=1.0
sample=[	3	6	0	3	5	14	14	14	14	14]	reward=1.0
sample=[	3	6	0	3	5	14	14	14	14	14]	reward=1.0
sample=[	3	6	0	3	5	14	14	14	14	14]	reward=1.0
sample=[	3	6	0	3	5	14	14	14	14	14]	reward=1.0

**Figure 9 | Pictorial description of our planted sub-sequence didactic setup:** An example showing five samples drawn *i.i.d.* from a very strong policy ( $\gamma = 100$ ), and a relatively weaker ( $\gamma = 15$ ) policy in our didactic setup.

rewards. In all three works, the gains observed by using PRMs that predict step-level correctness (similar to [Lightman et al. \(2023\)](#)) is quite small, compared to simply using trained ORMs, or the ground-truth outcome supervision Rex. In fact, [Havrilla et al. \(2024\)](#) states that the only algorithm that does well is a form of expert iteration ([Anthony et al., 2017](#)), which does not inhibit exploration as severely as some other approaches they compare with. Our work presents one of the first results, where trained PRMs, used in conjunction with the outcome rewards during online RL, result in policies with substantially higher (+6%) performance, than the one trained only with outcome supervision. Our results also indicate a 5 – 6 $\times$  sample efficiency boost for online RL, with our trained PAVs.

**Connections to imitation learning through RL.** The idea of mixing potential functions from different policies  $\mu$  and  $\pi$ , in order to improve upon a sub-optimal expert  $\mu$  appears in [Chang et al. \(2015\)](#), but this work considers the structured prediction problem which is vastly different from our setting. Related to this, is the work by [Chang et al. \(2023\)](#), which uses a “guide” policy to rollout from prefixes generated by a base policy. The base policy can now imitate the guide by cloning those rollouts, and eventually surpass. Our work also uses a prover policy which can complete rollouts from states where the base policy fails. But, we also show that weak provers in many cases are able to improve the base policy, or search over its responses, better than a stronger prover policy. We tie this observation to the insight that the main goal of the prover policy is to distinguish steps taken by the base policy, as measured by advantages under the prover. Thus, we do not require the prover policy to be something better than the base policy, which is a key distinction with [Chang et al. \(2023\)](#).

## B. Didactic Analysis

We consider sequences of length 10 from a 15-token vocabulary  $\mathcal{V} := \{1, 2, \dots, 14\}$ , where the end-of-sequence token is given by 14, and all tokens following the end-of-sequence token (including it) are masked. Given an unknown planted sequence  $y^*$  (in Fig. 9), we train a policy  $\pi$  with policy gradient, where the outcome reward we wish to optimize is terminal and sparse, *i.e.*, for  $y \sim \pi$  we have  $r(y, y^*) = 1$  if and only if  $y^*$  appears in  $y$ , and 0 otherwise (Fig. 9). The policy  $\pi$  in our experiments is represented by a multi-layer neural network, similar to the MADE architecture ([Germain et al., 2015](#)). The prover policy  $\mu$  is parameterized by a scalar  $\gamma > 0$ . In particular, at any state  $s$ , where the last  $k$  tokens leading up to  $s$  match first  $k$  tokens of  $y^*$ , then:

$$\mu(y_{k+1}^* | s) \propto \gamma,$$

and uniform on all other tokens. Thus, as  $\gamma$  increases, the performance of  $\mu$  improves and  $\rightarrow 1$  as  $\gamma \rightarrow \infty$ . For our experiments, we assume (almost) oracle access to ground-truth advantage and  $Q$ -values, thus

Planted sub-sequence  $\mathbf{y}^*$  : [3 6 0 3 5]

Samples from policy corresponding to  $\gamma = 15$

sample=[	3	6	0	3	5	14	14	14	14	14]	reward=1.0
sample=[	3	6	0	4	0	3	1	11	7	3]	reward=0.0
sample=[	3	6	0	3	5	14	13	3	2	11]	reward=1.0
sample=[	3	6	0	13	5	3	1	3	14	1]	reward=0.0
sample=[	12	8	3	6	14	3	6	0	3	5]	reward=0.0

Samples from policy corresponding to  $\gamma = 100$

sample=[	3	6	0	3	5	14	14	14	14	14]	reward=1.0
sample=[	3	6	0	3	5	14	14	14	14	14]	reward=1.0
sample=[	3	6	0	3	5	14	14	14	14	14]	reward=1.0
sample=[	3	6	0	3	5	14	14	14	14	14]	reward=1.0
sample=[	3	6	0	3	5	14	14	14	14	14]	reward=1.0

图9 | *Pictorial description of our planted sub-sequence didactic setup*: 一个示例，显示了从非常强的策略（ $\gamma = 100$ ）中抽取的五个样本 *i.i.d.*，以及在我们的教学设置中相对较弱的策略（ $\gamma = 15$ ）。

奖励。在所有三篇作品中，使用预测步骤正确性的PRM（类似于Lightman等人（2023））观察到的收益相对较小，与仅使用训练好的ORM或基于Rex的真实结果监督相比。实际上，Havrilla等人（2024）指出，唯一表现良好的算法是一种专家迭代的形式（Anthony等人，2017），这种算法不像他们与其他方法进行比较时那样严重抑制探索。我们的工作呈现了第一个结果之一，即在在线RL过程中结合使用训练好的PRM和结果奖励，导致政策的性能显著提高（+6%），而仅通过结果监督训练的政策则不然。我们的结果还表明，对于在线RL，我们的训练好的PAVs具有5 – 6×样本效率提升。

连接到通过强化学习的模仿学习。将不同策略  $\mu$  和  $\pi$  的潜在函数混合以改进一个次优专家  $\mu$  的想法出现在 Chang 等人 (2015) 中，但这项工作考虑的是结构化预测问题，这与我们的设置大不相同。与此相关的是，Chang 等人 (2023) 的工作使用一个“引导”策略从基策略生成的前缀进行展开。基策略现在可以通过克隆这些展开来模仿引导策略，并最终超越。我们的工作也使用一个证明者策略，可以从基策略失败的状态完成展开。但是，我们还表明，在许多情况下，较弱的证明者比更强的证明者策略更能提高基策略或搜索其响应。我们将这一观察与证明者策略的主要目标是根据证明者的优点衡量基策略所采取的步骤来区分这一洞察联系起来。因此，我们不需要证明者策略比基策略更好，这是与 Chang 等人 (2023) 的关键区别。

## B. 教学分析

我们考虑来自15个标记词汇  $\mathcal{V} := \{1, 2, \dots, 14\}$  的长度为10的序列，其中序列结束标记由14给出，并且所有在序列结束标记之后的标记（包括该标记）都被遮掩。给定图9中未知植入的序列  $\mathbf{y}^*$ （我们训练一个策略  $\pi$ ，使用策略梯度，其中我们希望优化的结果奖励是终端且稀疏的，对于  $\mathbf{y} \sim \pi$ ，我们有  $r(\mathbf{y}, \mathbf{y}^*) = 1$  当且仅当  $\mathbf{y}^*$  出现在  $\mathbf{y}$  中，否则为0（图9）。在我们的实验中，策略  $\pi$  由多层神经网络表示，类似于MADE架构（Germain等，2015）。证明策略  $\mu$  由标量  $\gamma > 0$  参数化。特别是，在任何状态  $\mathbf{s}$  中，如果最后  $k$  个标记与  $\mathbf{y}^*$  的前  $k$  个标记匹配，则有：

$$\mu(\mathbf{y}_{k+1}^* \mid \mathbf{s}) \propto \gamma,$$

并且在所有其他token上均匀。因此，随着  $\gamma$  增加， $\mu$  的性能提高并趋向于  $\rightarrow 1$  作为  $\gamma \rightarrow \infty$ 。对于我们的实验，我们假设可以几乎无障碍地访问真实优势值和Q值，从而

源文本: .翻译

mitigating any confounding issues due to usage of a learned verifier. We are able to approximate exact  $Q$ -values very accurately by using Monte Carlo estimates with large  $> 100$  rollouts. With the goal of optimizing the terminal reward  $r(y, y^*)$ , we optimize  $\pi$  with two types of rewards: (i) only the outcome reward  $r(y, y^*)$ , which is equivalent to using only  $Q^\pi$  as step-level rewards; and (ii) using the effective reward:  $Q^\pi + \alpha A^\mu$  as the step-level reward.

**Training details.** We use effective rewards with  $\alpha = 1$ , and use the gradient in Eq. 5 to update the policy via policy gradient iterations. For the ORM runs, where we only use the outcome reward  $r(y, y^*)$ , the policy gradient is equivalent to the case where  $\alpha = 0$  in Eq. 5. We train for 10,000 iterations in both cases, with a batch size of 64, and a constant learning rate of  $1e-3$  for the Adam optimizer. The RL runs are initialized with a supervised finetuned policy. For this we take a randomly initialized network, based on the MADE architecture (Germain et al., 2015), with 3 layers, and 128 hidden units in each. Then we train it with supervised next-token prediction loss for 50 iterations on a dataset of 3200 samples from a weak policy ( $\gamma = 5.0$ ). The batch size for the SFT training is also set to 64. For evaluating Pass @ $N$  performance, we either sample  $N$  independent trajectories (temperature 1.0) from the base policy trained using effective rewards, or only  $Q^\pi$ . We also evaluate Pass @ $N$  for the SFT policy for comparison.

## C. Additional: Experiments on Test-time Search with PAVs

**Implementation details.** For our experiments in Sec. 4, we use three pretrained models: Gemma 2B, 9B and 27B. We finetune each of these on the MATH (Hendrycks et al., 2021) dataset. The finetuning is done for 5000 iterations, with a batchsize of 32, and a maximum learning rate of  $5e-6$  for 2B, 9B and  $5e-7$  for the 27B models. We trained the policies using the Adam optimizer, with a linear warm up and cosine decay learning rate schedule. The linear warm up is done for the first 500 iterations. For the base policies, we choose the SFT checkpoints with the best accuracy on a holdout validation set of the MATH dataset. Given the SFT checkpoints, we next train PAVs using the procedure in Sec. 4.2. We do this for a class of provers, which include the base policies themselves. As we discuss in Sec. 4, the prover class also includes the best-of- $K$  policy for  $K$  in  $\{2, 4, 8, 16, 32\}$ .

We use the hold out validation set to ascertain the value of  $\alpha$  in the effective reward. For each base policy we run beam search with a beam size of 16 on this hold out validation set, and using the base policy itself as a prover, we evaluate the value of  $\alpha$  that works best in the effective reward. We find that  $\alpha = 0.5$  worked best for Gemma 2B and 9B base policies, while a lower value of  $\alpha = 0.2$  was optimal for Gemma 27B. To tune  $\alpha$  we ran a grid search over the range  $[0.0, 1.0]$ , evaluating at an interval of 0.1. We observe that the choice of  $\alpha$  is a relatively robust one, since for all three base policies, we saw improvements (over only  $Q^\pi$  as the reward) for values in the range of  $[0.2, 0.6]$ . Having a separate value of  $\alpha$  for each base policy, we use the same value in the effective reward given by any choice of the prover policy that is used for that base policy. Next, we present an experiment that compares the predictive power of effective reward vs. just  $Q^\pi$  at initial states of a rollout under the base policy  $\pi$ , when either is used to predict the final outcome given by Rex.

**Experiment: Is the effective reward able to predict the final outcome better than  $Q^\pi$ ?** In Fig. 10, we describe an experiment where for both the effective reward  $Q^\pi + \alpha A^\mu$  (PAV) and just  $Q^\pi$  (PQV), we compute the error of the classifier that makes a prediction on the final outcome by thresholding on either reward value at each step of the rollout. This threshold is computed using a validation set, and is separate for each step and reward combination. The figure tells us that the outcome prediction error drops for both rewards as the base policy is rolled out more, but clearly the effective reward dominates  $Q^\pi$  (PQV)

减轻由于使用学习验证器而导致的任何混杂问题。我们能够通过使用大量 $> 100$ 次滚出的蒙特卡洛估计来非常准确地逼近精确的 $Q$ -值。为了优化终端奖励 $r(y, y^*)$ ，我们使用两种类型的奖励来优化 $\pi$ ：(i) 只有结果奖励 $r(y, y^*)$ ，这相当于仅使用 $Q^\pi$ 作为每步奖励；(ii) 使用有效奖励 $Q^\pi + \alpha A^\mu$ 作为每步奖励。

培训细节。我们使用有效的奖励 $\alpha = 1$ ，并通过策略梯度迭代使用Eq. 5中的梯度更新策略。对于仅使用结果奖励 $r(y, y^*)$ 的ORM运行，策略梯度等同于Eq. 5中 $\alpha = 0$ 的情况。两种情况下我们均训练10,000次迭代，批量大小为64，Adam优化器的学习率为 $1e-3$ 。对于RL运行，我们使用监督微调策略初始化。为此，我们基于MADE架构（Germain等，2015）随机初始化一个网络，包含3层，每层128个隐藏单元。然后我们使用监督下一个标记预测损失在包含3200个样本的数据集上对该网络进行50次迭代的训练，这些样本来自弱策略（ $\gamma = 5.0$ ）。SFT训练的批量大小也设置为64。对于Pass @ $N$ 性能评估，我们从使用有效奖励训练的基策略中采样 $N$ 独立轨迹（温度1.0），或者仅进行评估。我们还评估SFT策略的Pass @ $N$ 以进行比较。

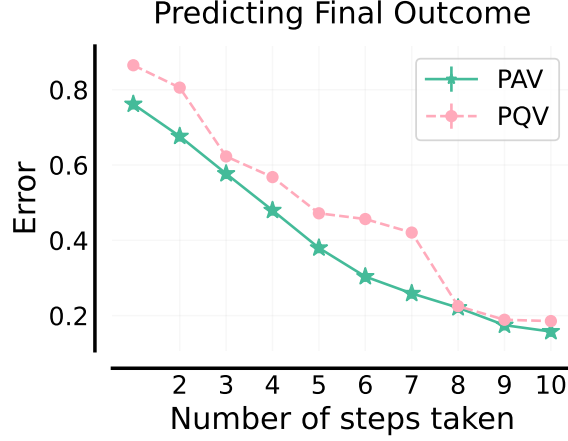
### C. 补充内容：使用PAVs的测试时搜索实验

实现细节。在第4节的实验中，我们使用了三个预训练模型：Gemma 2B、9B和27B。我们将每个模型分别在MATH（Hendrycks等人，2021）数据集上进行微调。微调过程进行了5000次迭代，批量大小为32，最大学习率为2B模型的 $5e-6$ ，9B模型的 $5e-7$ ，27B模型的 $5e-7$ 。我们使用Adam优化器训练策略，并采用线性预热和余弦衰减的学习率调度。线性预热在前500次迭代中进行。对于基础策略，我们选择了在MATH数据集的保留验证集上准确率最高的SFT检查点。在获得SFT检查点后，我们按照第4节2.2节的程序训练PAVs。我们对一类证明者进行训练，其中包括基础策略本身。如我们在第4节中讨论的，证明者类还包括 $K$ 个最佳策略中的一个，对于 $K$ 在 $\{2, 4, 8, 16, 32\}$ 。

我们使用保留验证集来确定有效奖励中 $\alpha$ 的值。对于每个基础策略，我们在保留验证集上运行具有16个分支的束搜索，并使用基础策略本身作为证明者，评估在有效奖励中表现最佳的 $\alpha$ 的值。我们发现对于Gemma 2B和9B基础策略， $\alpha = 0.5$ 工作最佳，而对于Gemma 27B基础策略，较低的 $\alpha = 0.2$ 是最优的。为了调整 $\alpha$ ，我们在范围 $[0.0, 1.0]$ 之间进行了网格搜索，每隔0.1评估一次。我们观察到 $\alpha$ 的选择相对稳健，因为在所有三个基础策略中，我们都在范围 $[0.2, 0.6]$ 的值中看到了改进（相对于仅使用 $Q^\pi$ 作为奖励）。对于每个基础策略使用不同的 $\alpha$ 值，在使用该基础策略的证明者策略中的有效奖励中使用相同的值。接下来，我们展示了一个实验，该实验比较了有效奖励与仅在基础策略 $\pi$ 的初始状态下使用 $Q^\pi$ 预测最终结果的预测能力，当使用Rex给出最终结果时。

实验：有效的奖励能否比 $Q^\pi$ 更好地预测最终结果？在图10中，我们描述了一个实验，在该实验中，对于有效的奖励 $Q^\pi + \alpha A^\mu$  (PAV)和仅仅 $Q^\pi$  (PQV)，我们在每个展开步骤中使用验证集计算分类器的误差，该分类器通过对奖励值进行阈值化来对最终结果进行预测。图表显示，随着基础策略的展开，两种奖励的预测误差都降低，但显然有效的奖励优于 $Q^\pi$  (PQV)。

across all steps. Thus, the effective reward is a more informative signal (lower classification error) for the problem of predicting the success of a partial rollout, especially in the earlier steps of the rollout. This helps to explain the better performance of beam search with a finite capacity beam that re-ranks partial rollouts with the effective reward. For this experiment, we use the Gemma 9B SFT policy as the base policy and the best-of-4 policy corresponding to the same SFT policy as the prover.



**Figure 10 | Effective rewards at any step are able to predict the outcome rewards at the final step, better than  $Q^\pi$ :** For both the effective reward  $Q^\pi + \alpha A^\mu$  and just  $Q^\pi$ , we compute the error of the classifier that makes a prediction on the final outcome by thresholding on either reward value at each step of the rollout. This threshold is computed using a validation set, and is separate for each step and reward combination. The figure tells us that the outcome prediction error drops for both rewards, as the base policy is rolled out more, but compared to  $Q^\pi$  at an intermediate step, the effective reward  $Q^\pi + \alpha A^\mu$  at an intermediate step is able to reliably predict the final outcome level correctness of the future rollout under the base policy  $\pi$ .

## D. Details on Collecting Data and Training PAVs

In Fig. 6(b) in Sec. 4.2, our seed rollouts are *i.i.d.* sampled from  $\pi$ , but prior works (Hwang et al., 2024; Luo et al., 2024; Setlur et al., 2024) employed a “first pit” strategy for coverage. Here, some initial sampling budget is spent to identify “high value” states where  $Q^\pi$  is larger than a threshold. Then, for any incorrect sample from these high value states greater budget is spent to estimate the first step (first pit) with  $Q^\pi = 0$ . All prefixes (and their estimated  $Q$ -values) until the first pit are then added to the training data. In Fig. 11, we compare beam search using PAVs trained using data from the first pit strategy, and the random sampling strategy. Both of them use the best value of  $n_{mc}/n_{cov}$  from Fig. 6(b) for every dataset size. We find the first pit strategy to be better than random, especially when the number of seed rollouts are limited. Once we get coverage over such pits, we sample a large number of partial rollouts conditioned on each prefix until the first pit. This is used to compute the Monte Carlo estimate of  $Q$  values more accurately on the path to the first pit. Each prefix and estimated  $Q$  value pair is then added to the dataset used to train PAVs.

**Training details.** All PAVs used in this work are trained by taking the Gemma 9B pretrained checkpoint and finetuning it on the data collected from the above strategy. The data collection uses first pit strategy for better coverage over pits in the seed rollouts. Based on findings in Fig. 6(b), we use a high value of  $n_{mc} = 20$  to estimate the  $Q$ -values accurately for each step in the seed rollout. For each base policy,



在整个步骤中。因此，有效的奖励是一个更具有信息量的信号（更低的分类误差），对于预测部分展开成功的问题，尤其是在展开的早期步骤中。这有助于解释有限容量的束搜索使用有效奖励重新排名部分展开时的更好性能。对于这个实验，我们使用Gemma 9B SFT策略作为基础策略，使用相同的SFT策略对应的最优4策略作为证明者。

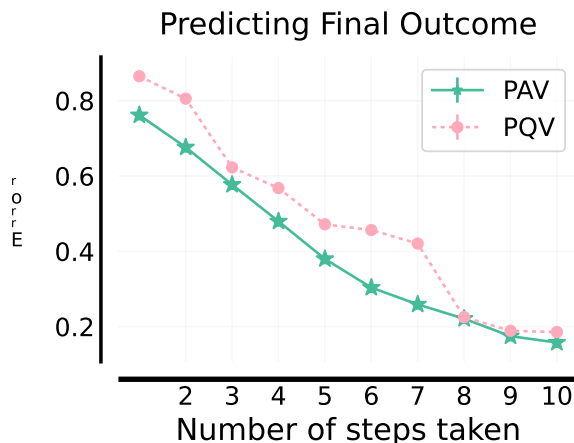


图 10 | *Effective rewards at any step are able to predict the outcome rewards at the final step, better than  $Q^\pi$* : 对于有效的奖励  $Q^\pi + \alpha A^\mu$  和普通的奖励  $Q^\pi$ ，我们在展开过程中的每一步通过阈值化奖励值来计算分类器的预测误差。该阈值使用验证集计算得出，并且每一步和每种奖励组合都不同。图表显示，随着基础策略的展开，预测最终结果的误差下降，但在中间步骤中，有效的奖励  $Q^\pi + \alpha A^\mu$  能够更可靠地预测在基础策略  $\pi$  下未来展开的最终结果水平准确性，与  $Q^\pi$  相比。

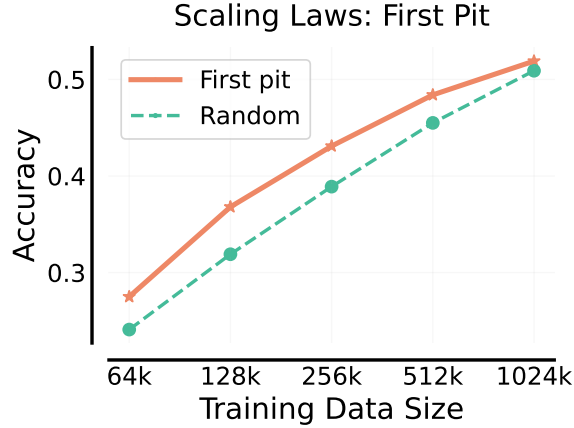
#### D. 数据收集和训练PAVs的详细信息

在第4.2节的Fig. 6(b)中，我们的种子展开是从*i.i.d.*中 $\pi$ 采样的，但之前的 works (Hwang et al., 2024; Luo et al., 2024; Setlur et al., 2024) 使用了一种“第一个坑”策略来进行覆盖。在此策略中，初始采样预算用于识别“高价值”状态，其中 $Q^\pi$ 大于阈值。然后，对于这些高价值状态中的任何错误样本，会花费更多的预算来使用 $Q^\pi = 0$ 估计第一个步骤（第一个坑）。直到第一个坑的所有前缀及其估计的Q值随后被添加到训练数据中。在Fig. 11中，我们比较了使用“第一个坑”策略训练的PAVs进行的束搜索和随机采样策略。两者都使用了Fig. 6(b)中每个数据集大小的最佳 $n_{mc}/n_{cov}$ 值。我们发现“第一个坑”策略优于随机策略，尤其是在种子展开数量有限的情况下。一旦我们对这些坑实现了覆盖，我们就会在每个前缀直到第一个坑的条件下采样大量部分展开。这用于更准确地计算通向第一个坑的路径上的Q值的蒙特卡洛估计。每个前缀及其估计的Q值对随后被添加到用于训练PAVs的数据集中。

训练细节。本文中使用的所有PAV都是通过使用Gemma 9B预训练检查点，并在上述策略收集的数据上进行微调而训练的。数据收集使用了优先挖掘策略，以更好地覆盖种子展开中的坑。根据图6(b)中的发现，我们使用高值 $n_{mc} = 20$ 来准确估计每个步骤中的Q值。对于每个基础策略，

in total, we collect a dataset of over 300,000 (prefix,  $\hat{Q}^\pi$ -value) pairs. Here,  $\hat{Q}^\pi$  is the Monte Carlo estimate for the  $Q$ -value at the prefix, under the policy  $\pi$ , on which we finetune the Gemma 9B model with cross-entropy loss. Since the distribution of values for  $\hat{Q}^\pi$  can be skewed, we split the range of  $\hat{Q}^\pi$ -values into two buckets, based on which we also partition the training data. The first bucket is the set of all prefixes with  $\hat{Q}^\pi < 0.5$  and the second is the set of all prefixes with  $\hat{Q}^\pi \geq 0.5$ . Then, we use class-balanced sampling over these buckets to finetune the pretrained model for 20000 training iterations, using a batch size of 32. We use an Adam optimizer with a maximum learning rate of  $5e-7$ . We use a linear warm up (till 2000 steps), followed by a cosine decay learning rate schedule to train the models. Since a pretrained LLM would output a matrix of logits (vocabulary size  $\times$  sequence length) we fix a token as the “scoring token” to be the end of the sequence / prefix that needs to be scored. The logits of this scoring token are then used to determine the prediction for the LLM being trained.

Once we have models that predict the  $Q^\pi$  for a base policy  $\pi$ , we compute the  $Q$ -value under the BoK policy corresponding to  $\pi$ , by setting:  $Q^{\text{BoK}(\pi)}(s, a) = 1 - (1 - Q^\pi(s, a))^K$ . Next, given the  $Q$ -values we for  $\pi$ , and their corresponding best-of- $K$  policies, we can compute any effective reward in Eq. 5, using the definition of advantage value of a step in Eq. 2.



**Figure 11 | First pit strategy from Luo et al. (2024); Setlur et al. (2024):** We compare the beam search performance (with beam size 128) for a PAV trained on data collected using two types of seed rollouts. For the seed rollouts, we either randomly sample from the distribution of state action pairs induced by the base policy. Or we improve coverage particularly by using the “first pit” strategy of identifying the first state with low  $Q$  values on a partial rollout that starts from a high  $Q$ -value state and ends with an outcome reward of 0, i.e.,  $Q$  value is 0.

## E. Additional: Experiments on RL Training with PAVs

**Training details.** As discussed in Section 5, the initialization for RL training is the RFT (rejection finetuned) checkpoint for the corresponding base policies. More specifically, we consider two base policies Gemma 2B SFT, and Gemma 9B SFT, where the RL training is initialized with the policy obtained by further optimizing the base policy via rejection finetuning (RFT) Yuan et al. (2023). This is done to improve coverage over states and actions during the initial iterations of RL training. For rejection finetuning we train the SFT model (base policy) on all correct trajectories sampled when collecting seed rollouts for training PAVs (that predict the advantage of the same base policy). The training details for RFT remain same as SFT and are detailed in Appendix C. We use the REINFORCE (Sutton et al., 1999)

总共，我们收集了一个包含超过 300,000 (前缀,  $\hat{Q}^\pi$ -值) 对的数据集。在这里， $\hat{Q}^\pi$  是在策略  $\pi$  下前缀的  $Q$ -值的蒙特卡洛估计值。我们使用交叉熵损失对 Gemma 9B 模型进行微调。由于  $\hat{Q}^\pi$  值的分布可能偏斜，我们将  $\hat{Q}^\pi$ -值的范围分为两个桶，并基于此对训练数据进行分区。第一个桶是所有  $\hat{Q}^\pi < 0.5$  的前缀集合，第二个桶是所有  $\hat{Q}^\pi \geq 0.5$  的前缀集合。然后，我们使用这些桶中的类别平衡采样对预训练模型进行 20000 次训练迭代，批量大小为 32。我们使用 Adam 优化器，最大学习率为  $5e-7$ 。我们使用线性预热（直到 2000 步骤），然后是余弦衰减学习率计划来训练模型。由于预训练的大语言模型会输出一个 logits 矩阵（词汇表大小  $\times$  序列长度），我们将一个标记作为“评分标记”，该标记是需要评分的序列/前缀的结束标记。然后使用该评分标记的 logits 来确定正在训练的大语言模型的预测。

一旦我们有了能够预测基政策  $Q^\pi$  的  $\pi$  的模型，我们就可以通过设置： $Q^{\text{BoK}(\pi)}(s, a) = 1 - (1 - Q^\pi(s, a))^K$  来计算与  $\pi$  对应的 BoK 政策下的  $Q$ -值。接下来，给定  $Q$ -值及其对应的最优  $K$  政策，我们就可以使用 Eq. 2 中步骤优势值的定义来计算 Eq. 5 中的任何有效奖励。

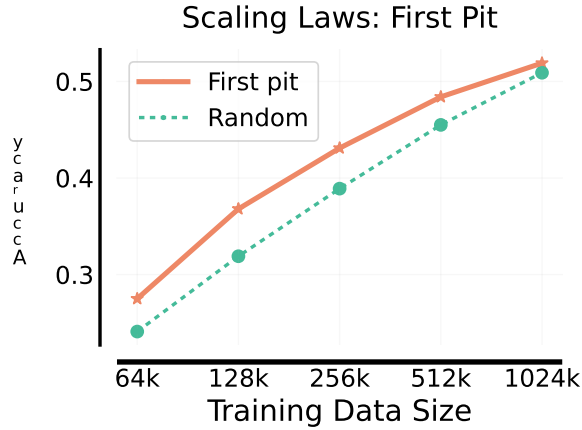


图 11 | **First pit strategy from Luo et al. (2024); Setlur et al. (2024)**: 我们比较了在使用两种类型种子展开数据收集的 PAU 上进行的束搜索性能（束大小为 128）。对于种子展开，我们要么从基础策略诱导的状态动作对分布中随机采样。要么通过使用“第一个坑”策略来特别提高覆盖率，该策略是从高  $Q$  值状态开始的部分展开的初始状态中识别出具有低  $Q$  值的状态，并且该部分展开以结果奖励为 0 结束，即  $Q$  值为 0。

## E. 补充：关于RL训练中使用PAVs的实验

训练细节。如第5节所述，RL训练的初始化为对应基础策略的RFT（拒绝微调）检查点。具体而言，我们考虑两种基础策略Gemma 2B SFT和Gemma 9B SFT，其中RL训练通过进一步优化基础策略（使用拒绝微调RFT，Yuan et al. (2023)）初始化策略，以在RL训练的初始迭代中提高状态和动作的覆盖范围。对于拒绝微调，我们使用SFT模型（基础策略）在收集用于训练PAVs（预测相同基础策略优势的种子回放）的所有正确轨迹时进行训练。RFT的训练细节与SFT相同，详见附录C。我们使用REINFORCE（Sutton et al., 1999）。

algorithm to improve the base policy. The RL training is run for 10000 iterations for the 2B model, and for 5000 iterations for the 9B model. For both we use the Adam optimizer with a learning rate of  $1e-7$ , and a batchsize of 32. The maximum response length is set to 512. For both we used a learning rate schedule with a linear warm up for 10% of the total training iterations, followed by a cosine decay. The implementation of our policy gradient algorithm also uses a token-level value function that is initialized to be the base policy itself, and is trained with a square loss. The value function is only used as a baseline during RL training, i.e., at any state  $s$  it is only predicting  $\mathbb{E}_{a \sim \pi(\cdot|s)} Q^\pi(s, a) + \alpha A^\mu(s, a)$ .

Most importantly, we use a validation set to identify a good choice of  $\alpha$  in the effective reward (in Eq. 5). For Gemma 2B this value is 5.0 and for the 9B policy  $\alpha = 3.0$  works best. Similar to the choice of  $\alpha$  for test-time search, we find that most values of  $\alpha$  in the range of 0.5 to 6.0 improved performance over ORM-RL, to different degrees. Both policies are also optimized with KL regularization, against the initial iterate of the RL policy, where the strength of the KL penalty is set to 0.001 for both.

## F. Theoretical Analysis: Complementary Provers Amplify Base Policy Improvement

In this section, we present the proofs for our theoretical results in the main paper (Sec. 3.4). We begin by describing some notation we use in our proofs, the natural policy gradient algorithm we use for the policy update, followed by the proof for Theorem 3.1. We also present a simple application of this result in Proposition F.1. Our results in this section are in the tabular setting, with softmax parameterization of the policies. Note that for the deterministic Markov decision process induced by the LLM, we are indeed in a tabular setting, where the set of states and actions is discrete, but large and grows exponentially in sequence length.

**Notation and preliminaries.** We use  $d_h^\pi, d_h^\mu$  to denote the distribution over states  $s_h$  at time step  $h$ , starting from the initial state distribution given by the empirical distribution over the questions in the dataset  $\mathcal{D}$ , and following the base policy  $\pi$ , or prover policy  $\mu$  respectively. The term  $d_s^\pi$  denotes the distribution over future states, starting from state  $s$ , and following policy  $\pi$ . Here,  $s$  can be a state at any time  $h \in [0, \dots, H]$ . For convenience, we overload the notation  $d_s^\pi$  (the distribution over future states induced by a policy starting from state  $s$ ), and use  $d_\rho^\pi$  to denote the mixture distribution over  $d_s^\pi$  starting from a random state  $s$  drawn from  $s \sim \rho$ , and following policy  $\pi$ .

The term  $Q^\pi(s_h, a_h)$  refers to the value of state-action pair  $s_h, a_h$ , i.e., the expected return in the future, starting the policy from state  $s_h$  and taking action  $a_h$ :

$$Q^\pi(s_h, a_h) := \mathbb{E}_{a_h, \dots, a_H \sim \pi(\cdot|s_h, a_h)} \left[ \text{Rex}((a_1, \dots, a_H), \mathbf{y}_x^*) \right]. \quad (7)$$

Note that  $\mathbf{y}_x^*$  is known on the dataset  $\mathcal{D}$ , and state  $s_h$  contains the question  $x$  as part of it. Similarly, we can define the value function  $V^\pi(s_h)$  of a state  $s_h$  as:

$$V^\pi(s_h) := \mathbb{E}_{a_{h+1} \sim \pi(\cdot|s_h)} Q^\pi(s_h, a_h). \quad (8)$$

The advantage function is then given by:

$$A^\pi(s_h, a_h) := Q^\pi(s_h, a_h) - V^\pi(s_h). \quad (9)$$

The policy gradient algorithm we use to update the base policy iteratively is natural policy gradient (Kakade, 2001b), and we use  $\pi_t$  to refer to the base policy iterate at time  $t$  of this iterative algorithm.

算法以改进基政策。对于2B模型，RL训练运行10000次迭代；对于9B模型，则运行5000次迭代。对于两者，我们都使用Adam优化器，学习率为 $1e-7$ ，批量大小为32。最大响应长度设置为512。对于两者，我们都使用了学习率调度，其中前10%的总训练迭代使用线性预热，之后使用余弦衰减。我们的策略梯度算法的实现还使用了一个初始化为基政策本身的标记级值函数，并使用平方损失进行训练。值函数仅在RL训练期间用作基线，即在任何状态 $s$ ，它仅预测  $_{a \sim \pi(\cdot|s)} Q^\pi(s, a) + \alpha A^\mu(s, a)$ 。

最重要的是，我们使用验证集来识别有效奖励（在 Eq. 5 中）中一个良好的  $\alpha$  选择。对于 Gemma 2B，这个值是 5.0，而对于 9B 策略， $\alpha = 3.0$  最佳。类似于测试时搜索中  $\alpha$  的选择，我们发现大多数  $\alpha$  值在 0.5 到 6.0 的范围内都能提高 ORM-RL 的性能，程度不同。两种策略也都使用 KL 正则化进行优化，针对 RL 策略的初始迭代，两种情况下的 KL 罚分强度都设置为 0.001。

## F. 理论分析：互补证明者增强基础政策改进

在本节中，我们呈现了我们在主论文（Sec. 3.4）中理论结果的证明。我们首先描述我们在证明中使用的某些符号，然后介绍我们用于策略更新的自然策略梯度算法，接着证明定理3.1。我们还展示了该结果的一个简单应用，即命题F.1。本节中的结果是在表格式设置中，使用softmax参数化策略。请注意，由LLM诱导的确定性马尔可夫决策过程确实处于表格式设置中，其中状态集和动作集是离散的，但数量庞大且随着序列长度的增加呈指数增长。

符号和预备知识。我们使用  $d_h^\pi, d_h^\mu$  表示在时间步  $h$  时的状态分布  $s_h$ ，从数据集中问题的 empirical 分布给出的初始状态分布出发，并遵循基础策略  $\pi$ ，或证明者策略  $\mu$ 。术语  $d_s^\pi$  表示从状态  $s$  出发，遵循策略  $\pi$  的未来状态分布。这里， $s$  可以是任意时间步  $h \in [0, \dots, H]$  的状态。为了方便，我们重载符号  $d_s^\pi$ （表示从状态  $s$ ）出发的策略诱导的未来状态分布，并使用  $d_\rho^\pi$  表示从分布  $s \sim \rho$  中随机抽取的状态  $s$  出发，遵循策略  $\pi$  的混合分布  $d_s^\pi$ 。

术语  $Q^\pi(s_h, a_h)$  指的是状态-动作对  $s_h, a_h$  i.e. 的值，从状态  $s_h$  开始执行策略并采取动作  $a_h$  之后的预期回报：

$$Q^\pi(s_h, a_h) := \mathbb{E}_{a_h, \dots, a_H \sim \pi(\cdot|s_h, a_h)} \left[ \text{Rex}((a_1, \dots, a_H), y_x^*) \right]. \quad (7)$$

注意  $y_x^*$  在数据集  $\mathcal{D}$  中是已知的，并且状态  $s_h$  包含问题  $x$  作为其一部分。类似地，我们可以定义状态  $s_h$  的价值函数  $V^\pi(s_h)$  为：

$$V^\pi(s_h) := \mathbb{E}_{a_{h+1} \sim \pi(\cdot|s_h)} Q^\pi(s_h, a_h). \quad (8)$$

优势函数然后由以下方式给出：

$$A^\pi(s_h, a_h) := Q^\pi(s_h, a_h) - V^\pi(s_h). \quad (9)$$

我们使用的用于迭代更新基础策略的策略梯度算法是自然策略梯度（Kakade, 2001b），并且我们使用  $\pi_t$  来表示此迭代算法在时间  $t$  的基础策略迭代  $\{v^*\}$ 。



Finally, we use  $\mathcal{S}$  to denote the set of all states (prefixes) and  $\mathcal{A}$  for the set of all actions (steps) that the LLM can take at any state.

**Parameterization of the base policy.** We adopt the softmax parameterization for the base policy:

$$\pi_{\theta}(a \mid s_h) = \frac{\exp(\theta_{s_h, a})}{\sum_{a' \in \mathcal{A}} \exp(\theta_{s_h, a'})}. \quad (10)$$

Here  $\theta_{s_h, a} \in \Theta \subseteq \mathbb{R}^d$  controls the probability of taking action  $a$  at state  $s_h$ . The full set of parameters across all states and actions is denoted by  $\theta \in \mathbb{R}^{d \times |\mathcal{S}| \times |\mathcal{A}|}$ . Whenever clear from context, we overload the notation  $\pi_t$  to denote both the policy at iterate  $t$ , i.e.,  $\pi_{\theta_t}$  and the parameter  $\theta_t$  itself. E.g., the gradient operator  $\nabla_{\pi_t}[\cdot]$  is referring to  $\nabla_{\theta_t}[\cdot]$ .

**Defining policy improvement.** Let  $\rho$  be a distribution over all states  $\{s_h : h \in [0, 1, \dots, H]\}$ , then  $\mathbb{E}_{s \sim \rho} V^{\pi}(s)$ , and  $\mathbb{E}_{s \sim \rho} V^{\mu}(s)$  give us the expected value functions over states across time steps, measured under  $\rho$ , for policies  $\pi$  and  $\mu$  respectively. We assume that  $d_h^{\pi}$  and  $d_h^{\mu}$  are both absolutely continuous with respect to  $\rho$ , and use the expected value function over  $\rho$  as the quantity we track before and after a policy update. A positive change in  $\mathbb{E}_{s \sim \rho} V^{\pi}(s)$  implies a net positive improvement in the base policy. Thus, progress is made at each update of the policy when:

$$\mathbb{E}_{s \sim \rho} V^{\pi_{t+1}}(s) - \mathbb{E}_{s \sim \rho} V^{\pi_t}(s) > 0.$$

### F.1. Natural Policy Gradient

The natural policy gradient (NPG) algorithm (Kakade, 2001a) defines a Fisher information matrix (induced by the policy), and performs gradient updates in the geometry induced by the following matrix:

$$F_{\rho}(\pi) = \mathbb{E}_{s \sim d_{\rho}^{\pi}} \mathbb{E}_{a \sim \pi(\cdot \mid s)} \left[ \nabla_{\pi} \log \pi(a \mid s) \left( \nabla_{\pi} \log \pi(a \mid s) \right)^{\top} \right] \quad (11)$$

Typically, the NPG update does gradient updates on the objective  $\ell_{\text{ORM-RL}}$  in Eq. 3, but in our case, the objective of interest is  $\ell_{\text{PAV-RL}}$  in Eq. 4, and thus the natural gradient is given by:

$$\pi_{t+1} = \pi_t + \gamma \cdot F_{\rho}(\pi_t)^{\dagger} \left( \nabla_{\pi} \ell_{\text{PAV-RL}}(\pi) \Big|_{\pi=\pi_t} \right), \quad (12)$$

where  $M^{\dagger}$  denotes the Moore-Penrose pseudoinverse of the matrix  $M$ . We restrict to using the initial state distribution  $\rho$  in our update rule, i.e., we restrict attention to states  $s$  reachable from  $\rho$ , since  $\rho$  governs the performance measure of interest when evaluating the expected value of a policy. Thus, without loss of generality, we can exclude states that are not reachable under  $\rho$ . Specifically, we restrict the MDP to the set:  $\{s_h : \exists \pi \text{ such that } d_{\rho}^{\pi}(s_h) > 0, h \in [0, \dots, H]\}$ . The scalar  $\gamma > 0$  determines the learning rate.

### F.2. Useful Lemmas

**Lemma F.1.** [The performance difference lemma; (Kakade and Langford, 2002)] For all policies  $\pi, \pi'$  and states  $s_0$ ,

$$V^{\pi}(s) - V^{\pi'}(s) = \mathbb{E}_{s_h \sim d_s^{\pi}} \mathbb{E}_{a_h \sim \pi(\cdot \mid s_h)} \left[ A^{\pi'}(s_h, a_h) \right].$$

最后，我们使用  $\mathcal{S}$  表示所有状态（前缀）的集合，并使用  $\mathcal{A}$  表示所有 LLM 可以在任何状态下采取的动作（步骤）的集合。

基政策的参数化。我们采用softmax参数化基政策：

$$\pi_{\theta}(a | s_h) = \frac{\exp(\theta_{s_h, a})}{\sum_{a' \in \mathcal{A}} \exp(\theta_{s_h, a'})}. \quad (10)$$

Here  $\theta_{s_h, a} \in \Theta \subseteq \mathbb{R}^d$  控制在状态  $s_h$  执行动作  $a$  的概率。所有状态和动作的完整参数集由  $\theta \in \mathbb{R}^{d \times |\mathcal{S}| \times |\mathcal{A}|}$  表示。只要从上下文可以明确理解，我们重载符号  $\pi_t$  来表示第  $t$  次迭代、第  $i.e.$  次迭代、第  $\pi_{\theta_t}$  次迭代的策略和参数  $\theta_t$  本身。E.g., 梯度运算符  $\nabla_{\pi_t}[\cdot]$  指的是  $\nabla_{\theta_t}[\cdot]$ 。

定义策略改进。设  $\rho$  是所有状态  $\{s_h\}$  的分布： $h \in [0, 1, \dots, H]$ ，则  $s \sim_{\rho} V^{\pi}(s)$  和  $s \sim_{\rho} V^{\mu}(s)$  给出了在时间步长上基于  $\rho$  的状态的期望值函数，分别对应于策略  $\pi$  和  $\mu$ 。我们假设  $d_h^{\pi}$  和  $d_h^{\mu}$  都相对于  $\rho$  完全连续，并使用  $\rho$  的期望值函数作为在策略更新前后跟踪的数量。 $s \sim_{\rho} V^{\pi}(s)$  的正值变化意味着基础策略的净积极改进。因此，当在策略更新时： $d_h^{\pi}$  和  $d_h^{\mu}$  都相对于  $\rho$  完全连续，使用  $\rho$  的期望值函数作为在策略更新前后跟踪的数量。 $s \sim_{\rho} V^{\pi}(s)$  的正值变化意味着基础策略的净积极改进。因此，当在策略更新时：

$$\mathbb{E}_{s \sim \rho} V^{\pi_{t+1}}(s) - \mathbb{E}_{s \sim \rho} V^{\pi_t}(s) > 0.$$

### F.1. 自然策略梯度

自然策略梯度（NPG）算法（Kakade, 2001a）定义了一个费舍尔信息矩阵（由策略诱导），并在由以下矩阵诱导的几何中进行梯度更新：

$$F_{\rho}(\pi) = \mathbb{E}_{s \sim d_{\rho}^{\pi}} \mathbb{E}_{a \sim \pi(\cdot | s)} \left[ \nabla_{\pi} \log \pi(a | s) \left( \nabla_{\pi} \log \pi(a | s) \right)^{\top} \right] \quad (11)$$

通常，NPG 更新会对目标  $\ell_{\text{ORM-RL}}$ （在式 3 中）进行梯度更新，但在我们的情况下，目标是式 4 中的  $\ell_{\text{PAV-RL}}$ ，因此自然梯度给定为：

$$\pi_{t+1} = \pi_t + \gamma \cdot F_{\rho}(\pi_t)^{\dagger} \left( \nabla_{\pi} \ell_{\text{PAV-RL}}(\pi) \Big|_{\pi=\pi_t} \right), \quad (12)$$

其中  $M^{\dagger}$  表示矩阵  $M$  的 Moore-Penrose 幻影逆。我们限制使用初始状态分布  $\rho$  在我们的更新规则  $i.e.$  中，我们限制注意力仅关注由  $s$  可达的状态  $\rho$ ，因为  $\rho$  管理着评估策略期望值时所关心的性能指标。因此，不失一般性，我们可以排除在  $\rho$  下不可达的状态。具体来说，我们将 MDP 限制在集合： $\{s_h : \exists \pi \text{ 使得 } d_{\rho}^{\pi}(s_h) > 0, h \in [0, \dots, H]\}$ 。标量  $\gamma > 0$  确定了学习率。

### F.2. 有用的引理

引理 F.1. [The performance difference lemma; (Kakade and Langford, 2002)] For all policies  $\pi, \pi'$  and states  $s_0$ ,

$$V^{\pi}(s) - V^{\pi'}(s) = \mathbb{E}_{s_h \sim d_s^{\pi}} \mathbb{E}_{a_h \sim \pi(\cdot | s_h)} \left[ A^{\pi'}(s_h, a_h) \right].$$

*Proof.* See proof of Lemma 6.1 in [Kakade and Langford \(2002\)](#).  $\square$

**Lemma F.2** (Natural policy gradient update). *For the natural policy gradient in Eq. 12, the corresponding policy update is given by:*

$$\pi^{t+1}(a | s) = \pi^t(a | s) \cdot \frac{\exp(\gamma \cdot (Q^t(s, a) + \alpha \cdot A^\mu(s, a)))}{Z^t(s)}, \quad (13)$$

$$Z^t(s) = \gamma \cdot \sum_{a \in \mathcal{A}} (Q^t(s, a) + \alpha \cdot A^\mu(s, a)) \quad (14)$$

*Proof.* We use arguments similar to the proof of Lemma 15 in [Agarwal et al. \(2021\)](#), with the key difference of separately accounting for the term  $A^\mu(s, a)$  in the effective reward. For the sake of completeness we reproduce some of the derivation, accounting for the  $A^\mu$  term in the process. We follow compatible function approximation in [Sutton et al. \(1999\)](#) and [Kakade \(2001a\)](#). For a vector  $\mathbf{w} \in \mathbb{R}^{d \times |S| \times |\mathcal{A}|}$ , we define the error function

$$L^\pi(\mathbf{w}) = \mathbb{E}_{s \sim d_\rho^\pi, \mathbb{E}_{a \sim \pi(\cdot | s)} [\mathbf{w}^\top \nabla_\pi \log \pi(\cdot | s) - (A^\pi(s, a) + \alpha A^\mu(s, a) - \alpha \mathbb{E}_{a \sim \pi_t(\cdot | s)} A^\mu(s, a))]^2. \quad (15)$$

Let  $\mathbf{w}^*$  be the minimizer of  $L^\pi(\mathbf{w})$  with the smallest  $\ell_2$  norm. Then by definition of Moore-Penrose pseudoinverse:

$$\begin{aligned} \mathbf{w}^* &= F_\rho(\pi)^\dagger \mathbb{E}_{s \sim d_\rho^\pi, a \sim \pi(a | s)} [\nabla_\pi \log \pi(a | s) (A^\pi(s, a) + \alpha A^\mu(s, a) - \alpha \mathbb{E}_{a \sim \pi_t(\cdot | s)} A^\mu(s, a))] \\ &= F_\rho(\pi)^\dagger \nabla_\pi \ell_{\text{PAV-RL}}(\pi). \end{aligned} \quad (16)$$

In other words,  $\mathbf{w}^*$  is precisely proportional to the NPG update direction. Note further that for the Softmax policy parameterization, we have:

$$\mathbf{w}^\top \nabla \log \pi(a | s) = \mathbf{w}_{s,a} - \sum_{a' \in \mathcal{A}} \mathbf{w}_{s,a'} \pi(a' | s).$$

Since  $\sum_{a \in \mathcal{A}} \pi(a | s) A^\pi(s, a) = 0$ , this immediately yields that:

$$L^\pi(A^\pi(s, a) + \alpha A^\mu(s, a)) = 0.$$

However, this might not be the unique minimizer of  $L^\pi$ , which is problematic since  $\mathbf{w}^*(\pi)$  as defined in terms of the Moore-Penrose pseudoinverse is formally the smallest norm solution to the least-squares problem, which  $A^\pi + \alpha A^\mu$  may not be. However, given any vector  $\mathbf{v} \in \mathbb{R}^{|S| \times |\mathcal{A}|}$ , let us consider solutions of the form  $A^\pi + \alpha A^\mu + \mathbf{v}$ . Due to the form of the derivatives of the policy for the softmax parameterization, we have for any state  $s, a$  such that  $s$  is reachable under  $\rho$ ,

$$\mathbf{v}^\top \nabla_\pi \log \pi(a | s) = \sum_{a' \in \mathcal{A}} (\mathbf{v}_{s,a'} \mathbf{1}[a = a'] - \mathbf{v}_{s,a'} \pi(a' | s)) = \mathbf{v}_{s,a} - \sum_{a' \in \mathcal{A}} \mathbf{v}_{s,a'} \pi(a' | s).$$

This is because  $\pi$  is a stochastic policy with  $\pi(a | s) > 0$  for all actions  $a$  in each state  $s$ , so that if a state is reachable under  $\rho$ , it will also be reachable using  $\pi$ , and hence the zero derivative conditions apply at each reachable state. For  $A^\pi + \alpha A^\mu + \mathbf{v}$  to minimize  $L^\pi$ , we would like  $\mathbf{v}^\top \nabla_\pi \log \pi(a | s) = 0$  for all  $s, a$  so that  $\mathbf{v}_{s,a}$  is independent of the action and can be written as a constant  $c_s$  for each  $s$  by the above equality. Hence, the minimizer of  $L^\pi(\mathbf{w})$  is determined up to a state-dependent offset, and

$$F_\rho(\theta)^\dagger \nabla_\pi \ell_{\text{PAV-RL}}(\pi) = Q^\pi + \alpha A^\mu + \mathbf{v},$$

*Proof.* 查看Kakade和Langford (2002) 中的引理6.1的证明。  $\square$

引理 F.2 (自然策略梯度更新). *For the natural policy gradient in Eq. 12, the corresponding policy update is given by:*

$$\pi^{t+1}(a | s) = \pi^t(a | s) \cdot \frac{\exp(\gamma \cdot (Q^t(s, a) + \alpha \cdot A^\mu(s, a)))}{Z^t(s)}, \quad (13)$$

$$Z^t(s) = \gamma \cdot \sum_{a \in \mathcal{A}} (Q^t(s, a) + \alpha \cdot A^\mu(s, a)) \quad (14)$$

*Proof.* 我们使用类似于 Agarwal 等人 (2021) 中引理 15 证明中的论证方法, 关键区别在于单独考虑有效奖励中的项  $A^\mu(s, a)$ 。为了完整性, 我们重新推导了一些公式, 并在过程中考虑了  $A^\mu$  项。我们遵循 Sutton 等人 (1999) 和 Kakade (2001a) 的兼容函数逼近方法。对于向量  $\mathbf{w} \in \mathbb{R}^{d \times |\mathcal{S}| \times |\mathcal{A}|}$ , 我们定义误差函数

$$L^\pi(\mathbf{w}) = \mathbb{E}_{s \sim d_\rho^\pi, \mathbb{E}_{a \sim \pi(\cdot | s)} [\mathbf{w}^\top \nabla_\pi \log \pi(\cdot | s) - (A^\pi(s, a) + \alpha A^\mu(s, a) - \alpha \mathbb{E}_{a \sim \pi_t(\cdot | s)} A^\mu(s, a))]^2. \quad (15)$$

令  $\mathbf{w}^*$  为使  $L^\pi(\mathbf{w})$  最小化且  $\ell_2$  范数最小的解。然后根据 Moore-Penrose 广义逆的定义:

$$\begin{aligned} \mathbf{w}^* &= F_\rho(\pi)^\dagger \mathbb{E}_{s \sim d_\rho^\pi, a \sim \pi(a | s)} [\nabla_\pi \log \pi(a | s) (A^\pi(s, a) + \alpha A^\mu(s, a) - \alpha \mathbb{E}_{a \sim \pi_t(\cdot | s)} A^\mu(s, a))] \\ &= F_\rho(\pi)^\dagger \nabla_\pi \ell_{\text{PAV-RL}}(\pi). \end{aligned} \quad (16)$$

换句话说,  $\mathbf{w}^*$  精确地与 NPG 更新方向成正比。进一步注意, 在 Softmax 策略参数化的情况下, 我们有:

$$\mathbf{w}^\top \nabla \log \pi(a | s) = \mathbf{w}_{s,a} - \sum_{a' \in \mathcal{A}} \mathbf{w}_{s,a'} \pi(a' | s).$$

自从  $\sum_{a \in \mathcal{A}} \pi(a | s) A^\pi(s, a) = 0$ , 这立即意味着:

$$L^\pi(A^\pi(s, a) + \alpha A^\mu(s, a)) = 0.$$

然而, 这可能不是  $L^\pi$  的唯一最小化器, 这在  $\mathbf{w}^*(\pi)$  作为基于摩尔-彭罗斯伪逆定义的最小范数最小二乘问题解的情况下是存在问题的, 而  $A^\pi + \alpha A^\mu$  可能不满足这一条件。然而, 对于任意向量  $\mathbf{v} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ , 让我们考虑形式为  $A^\pi + \alpha A^\mu + \mathbf{v}$  的解。由于 softmax 参数化策略的导数形式, 对于任意状态  $s, a$ , 只要  $s$  在  $\rho$  下可达, 我们有

$$\mathbf{v}^\top \nabla_\pi \log \pi(a | s) = \sum_{a' \in \mathcal{A}} (\mathbf{v}_{s,a'} \mathbf{1}[a = a'] - \mathbf{v}_{s,a'} \pi(a' | s)) = \mathbf{v}_{s,a} - \sum_{a' \in \mathcal{A}} \mathbf{v}_{s,a'} \pi(a' | s).$$

这是因为  $\pi$  是一个在每个状态  $s$  的所有动作  $a$  上都为  $\pi(a | s) > 0$  的随机策略, 因此如果一个状态在  $\rho$  下可达, 那么使用  $\pi$  也可达, 从而在每个可达状态处零导数条件成立。为了使  $A^\pi + \alpha A^\mu + \mathbf{v}$  最小化  $L^\pi$ , 我们希望  $\mathbf{v}^\top \nabla_\pi \log \pi(a | s) = 0$  对所有  $s, a$  成立, 这样  $\mathbf{v}_{s,a}$  就与动作无关, 并且可以通过上述等式写成每个  $s$  的常数  $c_s$ 。因此,  $L^\pi(\mathbf{w})$  的最小值确定到一个状态相关的偏移量, 和

$$F_\rho(\theta)^\dagger \nabla_\pi \ell_{\text{PAV-RL}}(\pi) = Q^\pi + \alpha A^\mu + \mathbf{v},$$

where  $\mathbf{v}_{s,a} = c_s$  for some  $c_s \in \mathbb{R}$  for each state  $s$  and action  $a$ . Finally, we observe that this yields the updates

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}^t + \gamma(Q^\pi + \alpha A^\mu + \mathbf{v}) \quad \text{and} \quad \pi_{t+1}(a | s) = \pi_t(a | s) \frac{\exp(\gamma Q^t(s, a) + \gamma \alpha A^\mu(s, a))}{Z^t(s)}.$$

Owing to the normalization factor  $Z^t(s)$ , the state dependent offsets cancel in the updates for  $\pi$ , which yields the statement of the lemma.  $\square$

### F.3. Proof of Theorem 3.1

*Proof.* For some notational convenience, we use  $V^t, V^{t+1}$ , to denote the value functions  $V^{\pi_t}, V^{\pi_{t+1}}$  for policies at  $\pi_t, \pi_{t+1}$  at time  $t$  and  $t+1$  respectively. Similarly, we use  $A^t, A^{t+1}$  for  $A^{\pi_t}, A^{\pi_{t+1}}$  respectively. For the distribution over states  $d_\rho^{\pi_{t+1}}$  induced by the policy  $\pi_{t+1}$ , starting from an initial distribution of states given by  $\rho$ , we simplify the notation and use  $d_\rho^{t+1}$ . Similarly we use  $d_\rho^t$  for  $d_\rho^{\pi_t}$ .

Next, for simplicity we set  $\alpha = 1$  in the natural policy gradient update in Lemma F.2. It is easy to see that the lower bound result we show holds for any value of  $\alpha > 0$ , and the term in the lower bound scales linearly with  $\alpha$ . Note, that this is not a free variable, and  $\alpha$  has to be  $O(1)$ , since as we increase the value of  $\alpha$  we would have to correspondingly reduce  $\gamma$  for our result to hold. For now, we fix  $\alpha = 1$ .

From the policy difference Lemma F.1, we can write:

$$\mathbb{E}_{s \sim \rho} V^{t+1}(s) - \mathbb{E}_{s \sim \rho} V^t(s) = \mathbb{E}_{s \sim d_\rho^{t+1}} \mathbb{E}_{a \sim \pi^{t+1}(a|s)} [A^t(s, a)] \quad (17)$$

Next, from the natural policy gradient update in Lemma F.2, we can write  $A^{t+1}(s, a)$  as:

$$A^t(s, a) = \frac{1}{\gamma} \cdot \log \left( \frac{\pi^{t+1}(a | s) \cdot Z^t(s, a)}{\pi^t(a | s)} \right) - A^\mu(s, a) \quad (18)$$

Substituting Eq. 18 in Eq. 17 we get:

$$\begin{aligned} \mathbb{E}_{s \sim \rho} V^{t+1}(s) - \mathbb{E}_{s \sim \rho} V^t(s) &= \frac{1}{\gamma} \mathbb{E}_{s \sim d_\rho^{t+1}} [\text{KL}(\pi^{t+1}(\cdot | s) \| \pi^t(\cdot | s))] \\ &\quad + \frac{1}{\gamma} \log Z^t(s) - \mathbb{E}_{s \sim d_\rho^{t+1}} \mathbb{E}_{a \sim \pi^{t+1}(a|s)} A^\mu(s, a). \end{aligned} \quad (19)$$

Recall that for  $\alpha = 1$ ,

$$\log Z^t(s) = \log \mathbb{E}_{a \sim \pi^t(\cdot|s)} \exp(\gamma \cdot (A^t(s, a) + A^\mu(s, a))) \quad (20)$$

Applying Jensen's inequality we get:

$$\log Z^t(s) \geq \gamma \cdot \mathbb{E}_{a \sim \pi^t(\cdot|s)} [A^t(s, a) + A^\mu(s, a)] \quad (21)$$

$$= \gamma \cdot \mathbb{E}_{a \sim \pi^t(\cdot|s)} [A^\mu(s, a)], \quad (22)$$

since  $\mathbb{E}_{\pi^t} [A^t(s, a)] = 0$ . Note that in Eq. 20 the KL term is always non-negative. Thus, we can lower bound our policy improvement:

$$\mathbb{E}_{s \sim \rho} V^{t+1}(s) - \mathbb{E}_{s \sim \rho} V^t(s) \geq \mathbb{E}_{s \sim d_\rho^{t+1}} \langle \pi^{t+1}(\cdot | s) - \pi^t(\cdot | s), A^\mu(s, \cdot) \rangle, \quad (23)$$



其中  $v_{s,a} = c_s$  对于某些  $c_s \in \mathbb{R}$  对于每个状态  $s$  和动作  $a$ 。最后，我们观察到这产生了更新

$$\theta_{t+1} = \theta^t + \gamma(Q^\pi + \alpha A^\mu + v) \quad \text{and} \quad \pi_{t+1}(a | s) = \pi_t(a | s) \frac{\exp(\gamma Q^t(s, a) + \gamma \alpha A^\mu(s, a))}{Z^t(s)}.$$

由于归一化因子  $Z^t(s)$ ，状态相关的偏移在  $\pi$  的更新中相互抵消，这给出了引理的陈述。□

### F.3. 定理 3.1 的证明

*Proof.* 为了书写方便，我们使用  $V^t, V^{t+1}$  来表示在时间  $t$  和状态  $t+1$  下的策略  $\pi_t$  和  $\pi_{t+1}$  的值函数  $V^{\pi_t}, V^{\pi_{t+1}}$ 。类似地，我们使用  $A^t, A^{t+1}$  来表示  $A^{\pi_t}, A^{\pi_{t+1}}$ 。对于由策略  $\pi_{t+1}$  引起的状态分布  $d_\rho^{\pi_{t+1}}$ ，从初始状态分布  $\rho$  开始，我们简化记号使用  $d_\rho^{t+1}$ 。类似地，我们使用  $d_\rho^t$  来表示  $d_\rho^{\pi_t}$ 。

接下来，为了简化起见，在引理F.2中的自然策略梯度更新中我们设  $\alpha = 1$ 。很容易看出，我们展示的下界结果对于  $\alpha > 0$  的任何值都成立，并且下界中的项与  $\alpha$  成线性关系。注意，这不是一个自由变量，且  $\alpha$  必须是  $O(1)$ ，因为随着  $\alpha$  值的增加，我们必须相应地减少  $\gamma$  以使结果成立。目前，我们固定  $\alpha = 1$ 。

从政策差异引理 F.1，我们可以写出：

$$\mathbb{E}_{s \sim \rho} V^{t+1}(s) - \mathbb{E}_{s \sim \rho} V^t(s) = \mathbb{E}_{s \sim d_\rho^{t+1}} \mathbb{E}_{a \sim \pi^{t+1}(a|s)} [A^t(s, a)] \quad (17)$$

Next, 根据引理F.2中的自然策略梯度更新，我们可以将  $A^{t+1}(s, a)$  写为：

$$A^t(s, a) = \frac{1}{\gamma} \cdot \log \left( \frac{\pi^{t+1}(a | s) \cdot Z^t(s, a)}{\pi^t(a | s)} \right) - A^\mu(s, a) \quad (18)$$

代入 Eq. 18 到 Eq. 17，我们得到：

$$\begin{aligned} \mathbb{E}_{s \sim \rho} V^{t+1}(s) - \mathbb{E}_{s \sim \rho} V^t(s) &= \frac{1}{\gamma} \mathbb{E}_{s \sim d_\rho^{t+1}} [\text{KL}(\pi^{t+1}(\cdot | s) \| \pi^t(\cdot | s))] \\ &\quad + \frac{1}{\gamma} \log Z^t(s) - \mathbb{E}_{s \sim d_\rho^{t+1}} \mathbb{E}_{a \sim \pi^{t+1}(a|s)} A^\mu(s, a). \end{aligned} \quad (19)$$

召回对于  $\alpha = 1$ ,

$$\log Z^t(s) = \log \mathbb{E}_{a \sim \pi^t(\cdot | s)} \exp(\gamma \cdot (A^t(s, a) + A^\mu(s, a))) \quad (20)$$

应用詹森不等式，我们得到：

$$\log Z^t(s) \geq \gamma \cdot \mathbb{E}_{a \sim \pi^t(\cdot | s)} [A^t(s, a) + A^\mu(s, a)] \quad (21)$$

$$= \gamma \cdot \mathbb{E}_{a \sim \pi^t(\cdot | s)} [A^\mu(s, a)], \quad (22)$$

自从  $\pi^t[A^t(s, a)] = 0$ 。请注意，在Eq. 20中，KL项总是非负的。因此，我们可以给出我们策略改进的下界：

$$\mathbb{E}_{s \sim \rho} V^{t+1}(s) - \mathbb{E}_{s \sim \rho} V^t(s) \geq \mathbb{E}_{s \sim d_\rho^{t+1}} \langle \pi^{t+1}(\cdot | s) - \pi^t(\cdot | s), A^\mu(s, \cdot) \rangle, \quad (23)$$

where the inner product is the standard euclidean product as our actions space  $\mathcal{A}$  is discrete.

In the following we will treat the distribution  $\pi^{t+1}(\cdot | s)$  as a vector denoted by  $\pi$ . Next, from the NPG update we know that:

$$\pi^{t+1}(a | s) - \pi^t(a | s) = \pi^t(a | s) \left( \frac{\exp(\gamma A^t(s, a) + \gamma A^\mu(s, a))}{Z^t(s)} - 1 \right) \quad (24)$$

We note that for  $\gamma \ll 1$ ,  $\exp(\gamma A^t(s, a) + \gamma A^\mu(s, a)) = \Theta(1 + \gamma(A^t(s, a) + A^\mu(s, a)))$ , where the terms that grow as  $\omega(\gamma)$  are being ignored. Based on this, for  $\gamma \ll 1$ ,  $\exists$  constants  $0 < C_1 < C_2$  such that:

$$\exp(\gamma A^t(s, a) + \gamma A^\mu(s, a)) - 1 \in [C_1 \gamma (A^t(s, a) + A^\mu(s, a)), C_2 \gamma (A^t(s, a) + A^\mu(s, a))]$$

Applying the above claim in Eq. 24 gives us:

$$\begin{aligned} \pi^{t+1}(a | s) - \pi^t(a | s) &\geq \pi^t(a | s) \left( \frac{1 + C_1 \gamma (A^t(s, a) + A^\mu(s, a))}{1 + C_2 \gamma \mathbb{E}_{a \sim \pi^t(\cdot | s)} [A^t(s, a) + A^\mu(s, a)]} - 1 \right) \\ &\geq C_3 \gamma \frac{(\pi^t(a | s) (A^t(s, a) + A^\mu(s, a)) - \pi_t(a | s) \mathbb{E}_{a \sim \pi_t(a | s)} [A^t(s, a) + A^\mu(s, a)])}{1 + C_2 \gamma \mathbb{E}_{a \sim \pi_t(a | s)} [A^t(s, a) + A^\mu(s, a)]} \end{aligned} \quad (25)$$

$$= C_3 \gamma \frac{(\pi^t(a | s) (A^t(s, a) + A^\mu(s, a)) - \pi_t(a | s) \mathbb{E}_{a \sim \pi_t(a | s)} [A^\mu(s, a)])}{1 + \gamma C_2 \mathbb{E}_{a \sim \pi_t(a | s)} [A^t(s, a) + A^\mu(s, a)]}, \quad (26)$$

where we reused:  $\mathbb{E}_{\pi^t} [A^t(s, a)] = 0$ . Here,  $C_3 > 0$  is a constant.

We now plug in the above lower bound into Eq. 21 to get the final lower bound on the policy improvement in Theorem 3.1. For this, we will once again use the assumption that the learning rate  $\gamma \ll 1$ , which allows us to use  $1 + \gamma \mathbb{E}_{a \sim \pi_t(a | s)} [A^t(s, a) + A^\mu(s, a)] \geq C_4$  for some constant  $C_4 > 0$ . This is because, in our setting the range of the advantages is  $[-1, 1]$ .

$$\begin{aligned} \mathbb{E}_{s \sim \rho} [V^{t+1}(s) - V^t(s)] &\gtrsim \gamma \mathbb{E}_{s \sim d_\rho^{t+1}} [\mathbb{E}_{a \sim \pi^t(a | s)} [A^\mu(s, a) A^t(s, a)]] \\ &\quad + \gamma \mathbb{E}_{s \sim d_\rho^{t+1}} \left[ \mathbb{E}_{a \sim \pi^t(a | s)} [(A^\mu(s, a))^2] \right] \\ &\quad - \gamma \mathbb{E}_{s \sim d_\rho^{t+1}} \left[ (\mathbb{E}_{a \sim \pi^t(a | s)} [A^\mu(s, a)])^2 \right] \end{aligned} \quad (27)$$

Now Eq. 27 gives us,

$$\mathbb{E}_{s \sim \rho} [V^{t+1}(s) - V^t(s)] \gtrsim \gamma \mathbb{E}_{s \sim d_\rho^{t+1}} [\mathbb{V}_{a \sim \pi_t(a | s)} [A^\mu(s, a)] - \mathbb{E}_{a \sim \pi_t(a | s)} [A^\mu(s, a) A^t(s, a)]] \quad (28)$$

Now, for the last step we note that  $d_\rho^{t+1}$  is component wise larger than  $\rho$ , and this gives us the final result:

$$\mathbb{E}_{s \sim \rho} [V^{t+1}(s) - V^t(s)] \gtrsim \gamma \mathbb{E}_{s \sim \rho} [\mathbb{V}_{a \sim \pi_t(a | s)} [A^\mu(s, a)] - \mathbb{E}_{a \sim \pi_t(a | s)} [A^\mu(s, a) A^t(s, a)]] \quad (29)$$

□

#### F.4. Discussion on Remark 3.1

First, we note that if the  $Q$ -value of a base policy  $\pi$  at state, action pair  $(s, a)$  is  $Q^\pi(s, a)$ , then for the prover  $\mu$  set to the best-of- $K$  policy  $\text{BoK}(\pi)$ , the  $Q$ -value at the same state, action pair is:

$$Q^\mu(s, a) = 1 - (1 - Q^\pi(s, a))^K \quad (30)$$

其中内积是标准的欧几里得内积，因为我们的作用空间  $\mathcal{A}$  是离散的。

在以下内容中，我们将分布  $\pi^{t+1}(\cdot | \mathbf{s})$  处理为由  $\pi$  表示的向量。接下来，从 NPG 更新中我们知道：

$$\pi^{t+1}(a | \mathbf{s}) - \pi^t(a | \mathbf{s}) = \pi^t(a | \mathbf{s}) \left( \frac{\exp(\gamma A^t(\mathbf{s}, a) + \gamma A^\mu(\mathbf{s}, a))}{Z^t(\mathbf{s})} - 1 \right) \quad (24)$$

我们注意到对于  $\gamma \ll 1$ ， $\exp(\gamma A^t(\mathbf{s}, a) + \gamma A^\mu(\mathbf{s}, a)) = \Theta(1 + \gamma(A^t(\mathbf{s}, a) + A^\mu(\mathbf{s}, a)))$ ，其中的项  $\gamma(A^t(\mathbf{s}, a) + A^\mu(\mathbf{s}, a))$  增长作为  $\omega(\gamma)$  被忽略。基于此，对于  $\gamma \ll 1$ ， $\exists$  常数  $0 < C_1 < C_2$  使得：

$$\exp(\gamma A^t(\mathbf{s}, a) + \gamma A^\mu(\mathbf{s}, a)) - 1 \in [C_1 \gamma (A^t(\mathbf{s}, a) + A^\mu(\mathbf{s}, a)), C_2 \gamma (A^t(\mathbf{s}, a) + A^\mu(\mathbf{s}, a))]$$

应用上述断言在 Eq. 24 中给我们：

$$\begin{aligned} \pi^{t+1}(a | \mathbf{s}) - \pi^t(a | \mathbf{s}) &\geq \pi^t(a | \mathbf{s}) \left( \frac{1 + C_1 \gamma (A^t(\mathbf{s}, a) + A^\mu(\mathbf{s}, a))}{1 + C_2 \gamma \mathbb{E}_{a \sim \pi^t(\cdot | \mathbf{s})} [A^t(\mathbf{s}, a) + A^\mu(\mathbf{s}, a)]} - 1 \right) \\ &\geq C_3 \gamma \frac{(\pi^t(a | \mathbf{s}) (A^t(\mathbf{s}, a) + A^\mu(\mathbf{s}, a)) - \pi^t(a | \mathbf{s}) \mathbb{E}_{a \sim \pi^t(a | \mathbf{s})} [A^t(\mathbf{s}, a) + A^\mu(\mathbf{s}, a)])}{1 + C_2 \gamma \mathbb{E}_{a \sim \pi^t(a | \mathbf{s})} [A^t(\mathbf{s}, a) + A^\mu(\mathbf{s}, a)]} \end{aligned} \quad (25)$$

$$= C_3 \gamma \frac{(\pi^t(a | \mathbf{s}) (A^t(\mathbf{s}, a) + A^\mu(\mathbf{s}, a)) - \pi^t(a | \mathbf{s}) \mathbb{E}_{a \sim \pi^t(a | \mathbf{s})} [A^\mu(\mathbf{s}, a)])}{1 + \gamma C_2 \mathbb{E}_{a \sim \pi^t(a | \mathbf{s})} [A^t(\mathbf{s}, a) + A^\mu(\mathbf{s}, a)]}, \quad (26)$$

我们重用：  $\pi^t[A^t(\mathbf{s}, a)] = 0$ 。这里， $C_3 > 0$  是一个 constant。

我们现在将上述下界代入 Eq. 21，得到定理 3.1 中策略改进的最终下界。为此，我们将再次使用学习率  $\gamma \ll 1$  的假设，这使得我们可以使用  $1 + \gamma \mathbb{E}_{a \sim \pi^t(a | \mathbf{s})} [A^t(\mathbf{s}, a) + A^\mu(\mathbf{s}, a)] \geq C_4$  对于某个常数  $C_4 > 0$ 。因为在我们的设置中，优势的范围是  $[-1, 1]$ 。

$$\begin{aligned} \mathbb{E}_{\mathbf{s} \sim \rho} [V^{t+1}(\mathbf{s}) - V^t(\mathbf{s})] &\geq \gamma \mathbb{E}_{\mathbf{s} \sim d_\rho^{t+1}} [\mathbb{E}_{a \sim \pi^t(a | \mathbf{s})} [A^\mu(\mathbf{s}, a) A^t(\mathbf{s}, a)]] \\ &\quad + \gamma \mathbb{E}_{\mathbf{s} \sim d_\rho^{t+1}} \left[ \mathbb{E}_{a \sim \pi^t(a | \mathbf{s})} [(A^\mu(\mathbf{s}, a))^2] \right] \\ &\quad - \gamma \mathbb{E}_{\mathbf{s} \sim d_\rho^{t+1}} \left[ (\mathbb{E}_{a \sim \pi^t(a | \mathbf{s})} [A^\mu(\mathbf{s}, a)])^2 \right] \end{aligned} \quad (27)$$

现在 Eq. 27 给我们，

$$\mathbb{E}_{\mathbf{s} \sim \rho} [V^{t+1}(\mathbf{s}) - V^t(\mathbf{s})] \geq \gamma \mathbb{E}_{\mathbf{s} \sim d_\rho^{t+1}} [\mathbb{V}_{a \sim \pi^t(a | \mathbf{s})} [A^\mu(\mathbf{s}, a)] - \mathbb{E}_{a \sim \pi^t(a | \mathbf{s})} [A^\mu(\mathbf{s}, a) A^t(\mathbf{s}, a)]] \quad (28)$$

现在，对于最后一步，我们注意到  $d_\rho^{t+1}$  在每个分量上都大于  $\rho$ ，这给了我们最终结果

$$\mathbb{E}_{\mathbf{s} \sim \rho} [V^{t+1}(\mathbf{s}) - V^t(\mathbf{s})] \geq \gamma \mathbb{E}_{\mathbf{s} \sim \rho} [\mathbb{V}_{a \sim \pi^t(a | \mathbf{s})} [A^\mu(\mathbf{s}, a)] - \mathbb{E}_{a \sim \pi^t(a | \mathbf{s})} [A^\mu(\mathbf{s}, a) A^t(\mathbf{s}, a)]] \quad (29)$$

□

#### F.4. 关于注释 3.1 的讨论

首先，我们注意到如果基策略  $\pi$  在状态-动作对  $(\mathbf{s}, a)$  处的  $Q$ -值为  $Q^\pi(\mathbf{s}, a)$ ，那么对于证明者  $\mu$  设置为最佳-of-K 策略  $\text{BoK}(\pi)$ ，在相同的状态-动作对处的  $Q$ -值为：

$$Q^\mu(\mathbf{s}, a) = 1 - (1 - Q^\pi(\mathbf{s}, a))^K \quad (30)$$

This is because, in our setup the final outcome of correctness given by Rex is a random variable taking values in  $\{0, 1\}$ , with expectation  $Q^\pi(s, a)$ , when completing a rollout starting from prefix  $(s, a)$ . Thus, we can treat the outcome of function Rex as a Bernoulli random variable. Next, we can simply compute the probability of sampling a single correct answer out of  $K$  attempts, which is also a Bernoulli random variable with mean given by  $Q^\mu(s, a)$  in Eq. 30.

Now for Remark 3.1, we observe that whenever  $Q^\pi(s, a) \ll 1$ , e.g., when  $Q^\pi(s, a) = O(1/K)$ , for all values of  $(s, a)$ , we can do the Taylor approximation of  $Q^\mu(s, a)$  around 0, and note that  $Q^\mu(s, a) = \Theta(K \cdot Q^\pi(s, a))$ . Next, note the following calculation for the first term (distinguishability) in Theorem 3.1:

$$\begin{aligned} \mathbb{V}_\pi A^\mu(s, a) &= \mathbb{V}_\pi Q^\mu(s, a) \\ &= \mathbb{V}_\pi [1 - (1 - Q^\pi(s, a))^K] \\ &= \mathbb{V}_\pi [(1 - Q^\pi(s, a))^K] \end{aligned}$$

This means that the first term in Theorem 3.1 which measures “distinguishability” now increases by a factor of  $K^2$ . Similarly, we can see that the term which measures “misalignment” can change in magnitude by at most a factor of  $O(K)$ , since the misalignment term is linear in  $Q^\mu$ . These two observations combined lead us to the conclusion in Remark 3.1.

### F.5. Improving a Stronger Base policy with a Weaker Prover Policy

In Proposition F.1, we consider the case where the  $\pi$  and  $\mu$  differ in performance, as measured under the distribution of states  $\rho$  in the following way:

$$\mathbb{E}_{s \sim \rho} [|V^\mu(s) - V^{\pi_t}(s)|] = \eta.$$

Next, whenever the prover’s preference over actions is complementary to the base policy, by a factor of  $\eta$ , i.e.,

$$\mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi} [|Q^\mu(s, a) - Q^{\pi_t}(s, a)|] = \Theta(\eta),$$

then the variance of  $A^\mu$  or  $A^{\pi_t}$  under  $\pi_t$  should scale as  $\eta^2$ .

Thus, we see that when  $\pi_t$  fails to distinguish actions (i.e.,  $\mathbb{E}_{s \sim \rho} \mathbb{V}_{\pi_t} [A^{\pi_t}(s, a)]$  is small) regardless of the strength of prover policy  $\mu$ , as long as it is sufficiently complementary to  $\pi_t$ , the prover policy induces an improvement in base policy, that scales as  $\eta^2$ .

**Proposition F.1** (Complementary  $\mu$  boosts improvements in  $\pi$ ). *Under the distribution over states given by  $\rho$ , let prover  $\mu$  and base policy at iterate  $t$ ,  $\pi_t$ , differ in absolute performance, i.e.,*

$$\mathbb{E}_{s \sim \rho} [|V^\mu(s) - V^{\pi_t}(s)|] = \eta.$$

*When  $\mathbb{E}_{s \sim \rho} \mathbb{V}_{a \sim \pi_t} [A^{\pi_t}(s, a)] < \mathbb{E}_{s \sim \rho} \mathbb{V}_{a \sim \pi_t} [A^\mu(s, a)]$ , and  $\mu$  is complementary to  $\pi_t$ , i.e.,*

$$\mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} |Q^{\pi_t}(s, a) - Q^\mu(s, a)| = \Theta(\eta),$$

*then  $\mathbb{E}_{s \sim \rho} [V^{\pi_{t+1}}(s) - V^{\pi_t}(s)] \gtrsim \eta^2$ .*

这是因为，在我们的设置中，Rex 给出的最终正确性结果是一个取值在  $\{0, 1\}$  的随机变量，其期望为  $Q^\pi(s, a)$ ，当从前缀  $(s, a)$  开始完成一个 rollout 时。因此，我们可以将 Rex 的输出结果视为一个伯努利随机变量。接下来，我们可以简单地计算在  $K$  次尝试中采样一个正确答案的概率，这也是一个均值由  $\mathbb{E} q. 30$  中的  $Q^\mu(s, a)$  给出的伯努利随机变量。

现在对于 Remark 3.1，我们观察到每当  $Q^\pi(s, a) \ll 1$ , e.g., 当  $Q^\pi(s, a) = O(1/K)$  时，对于  $(s, a)$  的所有值，我们可以对  $Q^\mu(s, a)$  在 0 处进行泰勒近似，并注意到  $Q^\mu(s, a) = \Theta(K \cdot Q^\pi(s, a))$ 。接下来，注意定理 3.1 中第一项（可区分性）的以下计算：

$$\begin{aligned} \mathbb{V}_\pi A^\mu(s, a) &= \mathbb{V}_\pi Q^\mu(s, a) \\ &= \mathbb{V}_\pi [1 - (1 - Q^\pi(s, a))^K] \\ &= \mathbb{V}_\pi [(1 - Q^\pi(s, a))^K] \end{aligned}$$

这表明，在定理 3.1 中用于衡量“可区分性”的第一个项现在增加了  $K^2$  倍。类似地，我们可以看到用于衡量“不对齐”的项的大小最多可以变化  $O(K)$  倍，因为不对齐项与  $Q^\mu$  成线性关系。结合这两个观察，我们得出在注记 3.1 中的结论。

#### F.5. 使用较弱证明者策略改进更强基础策略

在命题 F.1 中，我们考虑  $\pi$  和  $\mu$  在状态分布  $\rho$  下的性能不同的情况：

$$\mathbb{E}_{s \sim \rho} [|V^\mu(s) - V^{\pi_t}(s)|] = \eta.$$

接下来，每当证明者对行动的偏好与基础策略互补时，按因子  $\eta$  即，

$$\mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi} [|Q^\mu(s, a) - Q^{\pi_t}(s, a)|] = \Theta(\eta),$$

然后  $A^\mu$  或  $A^{\pi_t}$  在  $\pi_t$  下的方差应按比例缩放为  $\eta^2$ 。

因此，我们看到当  $\pi_t$  无法区分行动 (i.e.,  $s \sim \rho \pi_t [A^{\pi_t}(s, a)]$  较小) 无论证明者策略  $\mu$  的强度如何，只要它足够补充  $\pi_t$ ，证明者策略就会诱导基础策略的改进，这种改进按比例为  $\eta^2$ 。

命题 F.1 (互补的  $\mu$  提升了  $\pi$  的改进)。Under the distribution over states given by  $\rho$ , let prover  $\mu$  and base policy at iterate  $t$ ,  $\pi_t$ , differ in absolute performance, i.e.,

$$\mathbb{E}_{s \sim \rho} [|V^\mu(s) - V^{\pi_t}(s)|] = \eta.$$

When  $s \sim \rho \pi_t [A^{\pi_t}(s, a)] < s \sim \rho \pi_t [A^\mu(s, a)]$ , and  $\mu$  is complementary to  $\pi_t$ , i.e.,

$$\mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} |Q^{\pi_t}(s, a) - Q^\mu(s, a)| = \Theta(\eta),$$

then  $s \sim \rho [V^{\pi_{t+1}}(s) - V^{\pi_t}(s)] \gtrsim \eta^2$ .

源文本: ,译文

*Proof.* We begin by proving an upper bound on the disagreement between prover and base policy:

$$\begin{aligned}
 & \mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} |Q^\mu(s, a) - Q^{\pi_t}(s, a)| \\
 & \leq \mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} |Q^\mu(s, a) - V^\mu(s)| + \mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} |Q^{\pi_t}(s, a) - V^\mu(s)| \\
 & \leq \mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} |Q^\mu(s, a) - V^\mu(s)| + \mathbb{E}_{s \sim \rho} [|V^\mu(s) - V^{\pi_t}(s)|] + \mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} |Q^{\pi_t}(s, a) - V^{\pi_t}(s, a)| \\
 & \leq \eta + \mathbb{E}_{s \sim \rho} \sqrt{\mathbb{V}_{\pi_t}[A^\mu(s, a)]} + \mathbb{E}_{s \sim \rho} \sqrt{\mathbb{V}_{\pi_t}[A^{\pi_t}(s, a)]},
 \end{aligned}$$

where the last inequality uses Cauchy-Schwartz. Next we apply Jensen’s inequality on the terms in the square root, and the conditions that  $\mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} |Q^{\pi_t}(s, a) - Q^\mu(s_h, a)| = \Omega(\eta)$  to conclude:

$$\sqrt{\mathbb{E}_{s \sim \rho} \mathbb{V}_{\pi_t}[A^\mu(s, a)]} \gtrsim \eta,$$

whenever  $\mathbb{E}_{s \sim \rho} \mathbb{V}_{a \sim \pi_t}[A^{\pi_t}(s, a)] < \mathbb{E}_{s \sim \rho} \mathbb{V}_{a \sim \pi_t}[A^\mu(s, a)]$ . Now that we lower bound “distinguishability”, it is easy to see that a similar derivation would upper bound the magnitude of the misalignment term by  $O(\eta^2)$ . Invoking the result in Theorem 3.1 yields:

$$E_{s \sim \rho}[V^{\pi_{t+1}}(s) - V^{\pi_t}(s)] \gtrsim \eta^2$$

□

## G. Examples Generated by Base Policy Trained on Rewards $Q^\pi + \alpha Q^\mu$

When we train the base policy with reinforcement learning where the reward is  $Q^\pi + \alpha Q^\mu$ , instead of the effective reward  $Q^\pi + \alpha A^\mu$ , we find that the policy training converges to contrived solutions with degenerate failures.

For example, we see in multiple examples (like the four below), that for every question, the model learns to output "REPHRASE THE PROBLEM" as the first step. This is because under  $Q^\mu$  this trivial step achieves a positive reward. While there exists a better solution where the policy outputs the correct answer and achieves higher rewards, it fails to recover from this local optima and find the better solution. Across multiple training runs, we observed similar trends when training with  $Q^\pi + \alpha Q^\mu$ . Note that in the idealized update, where we observe advantages  $A^\mu$  or value function  $Q^\mu$  on all actions at a state (e.g., in the NPG update in Sec. 3.4), using  $Q^\pi + \alpha Q^\mu$  is equivalent to  $Q^\pi + \alpha A^\mu$ . But, in practice,  $A^\mu$  offers significant benefits in the form of variance reduction in policy gradient iterates. In this instance, it does that by not assigning reward to steps like “REPHRASE THE PROBLEM”, since the likelihood of a prover solving before or after this step is expected to be roughly the same under any choice of the prover policy (i.e., the prover policy makes no progress).

### Example 1

Q: My club has 25 members. In how many ways can I choose members to form a 4-person executive committee?

A:

>>>

REPHRASE THE PROBLEM

WE NEED TO FIND THE NUMBER OF WAYS TO CHOOSE 4 MEMBERS OUT OF 25 FOR THE EXECUTIVE COMMITTEE SO WE CAN USE THE COMBINATION FORMULA WHICH IS



*Proof.* 我们首先证明证明者与基策略之间的分歧的一个上界：

$$\begin{aligned}
 & \mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} |Q^\mu(s, a) - Q^{\pi_t}(s, a)| \\
 & \leq \mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} |Q^\mu(s, a) - V^\mu(s)| + \mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} |Q^{\pi_t}(s, a) - V^\mu(s)| \\
 & \leq \mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} |Q^\mu(s, a) - V^\mu(s)| + \mathbb{E}_{s \sim \rho} [|V^\mu(s) - V^{\pi_t}(s)|] + \mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} |Q^{\pi_t}(s, a) - V^{\pi_t}(s, a)| \\
 & \leq \eta + \mathbb{E}_{s \sim \rho} \sqrt{\mathbb{V}_{\pi_t}[A^\mu(s, a)]} + \mathbb{E}_{s \sim \rho} \sqrt{\mathbb{V}_{\pi_t}[A^{\pi_t}(s, a)]},
 \end{aligned}$$

其中最后一个不等式使用了柯西-施瓦茨不等式。接下来我们在根号内的项上应用詹森不等式，并利用  $\mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} |Q^{\pi_t}(s, a) - Q^\mu(s, a)| = \Omega(\eta)$  的条件得出：

$$\sqrt{\mathbb{E}_{s \sim \rho} \mathbb{V}_{\pi_t}[A^\mu(s, a)]} \gtrsim \eta,$$

whenever  $\mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} [A^{\pi_t}(s, a)] < \mathbb{E}_{s \sim \rho} \mathbb{E}_{a \sim \pi_t} [A^\mu(s, a)]$ . 现在我们下界“可区分性”，很容易看出类似的推导会以上界的方式来限制偏差项的大小为  $O(\eta^2)$ 。调用定理 3.1 的结果得到：

$$E_{s \sim \rho} [V^{\pi_{t+1}}(s) - V^{\pi_t}(s)] \gtrsim \eta^2$$

□

## G. 由基于奖励 $Q^\pi + \alpha Q^\mu$ 训练的基础策略生成的示例

当我们在强化学习中训练基础策略，奖励是  $Q^\pi + \alpha Q^\mu$  而不是有效的奖励  $Q^\pi + \alpha A^\mu$  时，我们发现策略训练收敛到一些牵强的解决方案，并伴有退化的失败。

例如，在多个示例（如下面的四个示例）中，我们看到，对于每一个问题，模型学会将“重述问题”作为第一步输出。这是因为根据  $Q^\mu$ ，这个简单的步骤可以获得正奖励。虽然存在一个更好的解决方案，即策略输出正确的答案并获得更高的奖励，但它无法从这个局部最优解中恢复并找到更好的解决方案。在使用  $Q^\pi + \alpha Q^\mu$  进行多次训练时，我们观察到类似的趋势。请注意，在理想的更新中，如果我们在一个状态下观察到所有动作的优势  $A^\mu$  或价值函数  $Q^\mu$ （在第 3.4 节的 NPG 更新中为 e.g.），使用  $Q^\pi + \alpha Q^\mu$  等同于  $Q^\pi + \alpha A^\mu$ 。但在实践中， $A^\mu$  通过减少策略梯度迭代中的方差提供了显著的好处。在这个例子中，它通过不给像“重述问题”这样的步骤分配奖励来实现这一点，因为根据任何证明者策略的选择，证明者在该步骤之前或之后解决的概率预计大致相同（i.e., 证明者策略没有取得进展）。

### 示例 1

```

Q: My club has 25 members. In how many ways can I choose members to form a 4-person executive committee?

A:

>>>
REPHRASE THE PROBLEM
WE NEED TO FIND THE NUMBER OF WAYS TO CHOOSE 4 MEMBERS OUT OF 25 FOR THE EXECUTIVE COMMITTEE SO WE CAN USE THE COMBINATION FORMULA WHICH IS
    
```

```

N C R = N! / (R! * (N - R)!) WHERE N IS THE TOTAL NUMBER OF MEMBERS AND R IS
THE NUMBER OF MEMBERS WE CHOOSE SO IN THIS CASE N = 25 AND R = 4 SO WE PLUG THEM IN AND
SIMPLIFY 25 C 4 = 25! / (4! * (25 - 4)!) = (25 * 24 * 23 * 22) / (4 * 3 * 2 * 1) = 104005
# Answer

104005
    
```

### Example 2

Q: We have a triangle  $\triangle ABC$  and a point  $K$  on  $BC$  such that  $AK$  is an altitude of  $\triangle ABC$ . If  $AC = 10$ ,  $BK = 7$ , and  $BC = 13$ , then what is the area of  $\triangle ABC$ ?

A:

```

>>>
REPHRASE THE PROBLEM
WE NEED TO FIND THE AREA OF  $\triangle ABC$  SO WE CAN USE THE FORMULA  $A = \frac{1}{2}bh$ 
WHERE  $b$  IS THE BASE AND  $h$  IS THE HEIGHT SO SINCE  $AK$  IS AN ALTITUDE, THAT MEANS
 $AK \perp BC$  SO WE CAN USE  $BC$  AS THE BASE AND  $AK$  AS THE HEIGHT SO NOW WE NEED
TO FIND  $AK$  SO WE CAN USE THE PYTHAGORE THEOREM ON  $\triangle AKC$  SINCE  $AC = 10$ 
# Answer

48
    
```

### Example 3

Q: Suppose that  $n, n+1, n+2, n+3, n+4$  are five consecutive integers.

Determine a simplified expression for the sum of these five consecutive integers.

A:

```

>>>
REPHRASE THE PROBLEM
WE NEED TO FIND THE SUM OF THESE FIVE CONSECUTIVE INTEGERS SO WE CAN USE THE FORMULA
 $S_n = \frac{n}{2}(a_1 + a_n)$  WHERE  $S_n$  IS THE SUM,  $n$  IS THE NUMBER OF TERMS,
 $a_1$  IS THE FIRST TERM, AND  $a_n$  IS THE LAST TERM SO IN THIS CASE  $n = 5$ ,  $a_1 = n$ ,
 $a_5 = n + 4$ 
# Answer

5n + 10
    
```

### Example 4

Q: What is the value of  $x^2 + y^2 - z^2 + 2xy$  when  $x = -3$ ,  $y = 5$ , and  $z = -4$ ?

A:

```

>>>
REPHRASE THE PROBLEM
WE NEED TO FIND THE VALUE OF  $x^2 + y^2 - z^2 + 2xy$  WHEN  $x = -3$ ,  $y = 5$ , AND  $z = -4$ 
SO WE JUST PLUG IN THE VALUES AND SIMPLIFY
# Answer

-2
    
```

```

N C R = N! / (R! * (N - R)!) WHERE N IS THE TOTAL NUMBER OF MEMBERS AND R IS
THE NUMBER OF MEMBERS WE CHOOSE SO IN THIS CASE N = 25 AND R = 4 SO WE PLUG THEM IN AND
SIMPLIFY 25 C 4 = 25! / (4! * (25 - 4)!) = (25 * 24 * 23 * 22) / (4 * 3 * 2 * 1) = 104005
# Answer

104005
    
```

## 示例 2

Q: We have a triangle  $\triangle ABC$  and a point  $K$  on  $BC$  such that  $AK$  is an altitude of  $\triangle ABC$ . If  $AC = 10$ ,  $BK = 7$ , and  $BC = 13$ , then what is the area of  $\triangle ABC$ ?

A:

```

>>>
REPHRASE THE PROBLEM
WE NEED TO FIND THE AREA OF  $\triangle ABC$  SO WE CAN USE THE FORMULA  $A = \frac{1}{2}bh$ 
WHERE  $b$  IS THE BASE AND  $h$  IS THE HEIGHT SO SINCE  $AK$  IS AN ALTITUDE, THAT MEANS
 $AK \perp BC$  SO WE CAN USE  $BC$  AS THE BASE AND  $AK$  AS THE HEIGHT SO NOW WE NEED
TO FIND  $AK$  SO WE CAN USE THE PYTHAGORE THEOREM ON  $\triangle AKC$  SINCE  $AC = 10$ 
# Answer

48
    
```

## Example 3

Q: Suppose that  $n, n+1, n+2, n+3, n+4$  are five consecutive integers.

Determine a simplified expression for the sum of these five consecutive integers.

A:

```

>>>
REPHRASE THE PROBLEM
WE NEED TO FIND THE SUM OF THESE FIVE CONSECUTIVE INTEGERS SO WE CAN USE THE FORMULA
 $S_n = \frac{n}{2}(a_1 + a_n)$  WHERE  $S_n$  IS THE SUM,  $n$  IS THE NUMBER OF TERMS,
 $a_1$  IS THE FIRST TERM, AND  $a_n$  IS THE LAST TERM SO IN THIS CASE  $n = 5$ ,  $a_1 = n$ ,
 $a_5 = n + 4$ 
# Answer

5n + 10
    
```

## Example 4

Q: What is the value of  $x^2 + y^2 - z^2 + 2xy$  when  $x = -3$ ,  $y = 5$ , and  $z = -4$ ?

A:

```

>>>
REPHRASE THE PROBLEM
WE NEED TO FIND THE VALUE OF  $x^2 + y^2 - z^2 + 2xy$  WHEN  $x = -3$ ,  $y = 5$ , AND  $z = -4$ 
SO WE JUST PLUG IN THE VALUES AND SIMPLIFY
# Answer

-2
    
```