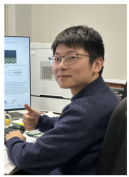
Demystifying LLM Reasoning through U-statistics Theory

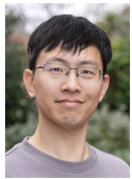
Chengchun Shi, Associate Professor of Data Science, LSE



Hongyi Zhou
Tsinghua

Kai Ye
LSE

Erhan Xu
LSE

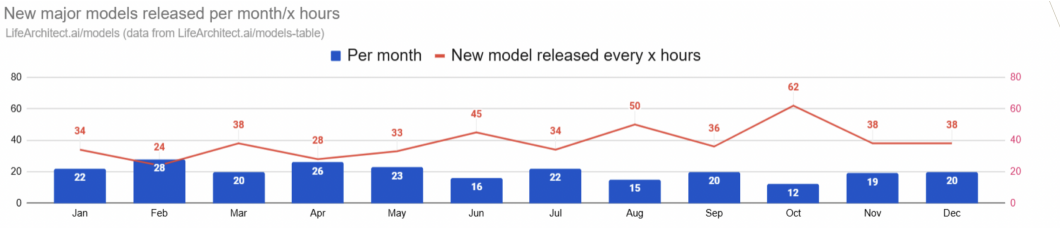


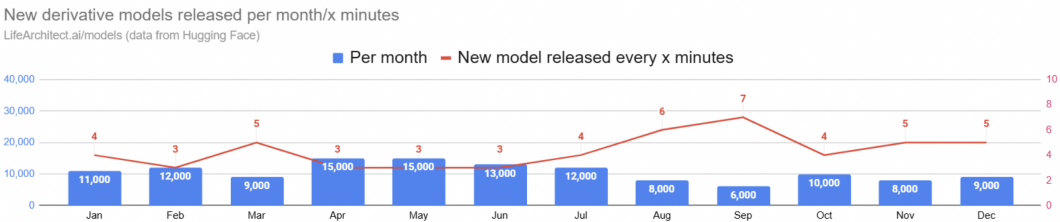
Jin Zhu
Birmingham

Ying Yang
Tsinghua

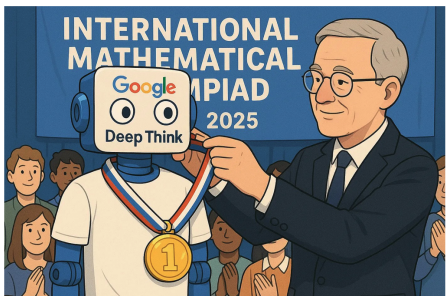
Shijin Gong
USTC

Number of LLMs Per Month (2024)





The Year of 2025



nature

Explore content | About the journal | Publish with us | Subscribe

nature › editorials › article

EDITORIAL | 17 September 2025

Bring us your LLMs: why peer review is good for AI models

Deepseek's R1 model has been peer reviewed. Others should follow the firm's example.

Group Relative Policy Optimization (GRPO)

	Benchmark (Metric)	Claude-3.5-Sonnet-1022	GPT-4o 0513	DeepSeek V3	OpenAI o1-mini	OpenAI o1-1217	DeepSeek R1
	Architecture	-	-	MoE	-	-	MoE
	# Activated Params	-	-	37B	-	-	37B
	# Total Params	-	-	671B	-	-	671B
English	MMLU (EM)	88.3	87.2	88.5	85.2	**91.8**	90.8
	MMLU-Redux (EM)	88.9	88.0	89.1	86.7	-	**92.9**
	MMLU-Pro (EM)	78.0	72.6	75.9	80.3	-	**84.0**
	DROP (3-shot F1)	88.3	83.7	91.6	83.9	90.2	**92.2**
	IF-Eval (Prompt Strict)	**86.5**	84.3	86.1	84.8	-	83.3
	GPQA Diamond (Pass@1)	65.0	49.9	59.1	60.0	**75.7**	71.5
	SimpleQA (Correct)	28.4	38.2	24.9	7.0	**47.0**	30.1
	FRAMES (Acc.)	72.5	80.5	73.3	76.9	-	**82.5**
	AlpacaEval2.0 (LC-winrate)	52.0	51.1	70.0	57.8	-	**87.6**
	ArenaHard (GPT-4-1106)	85.2	80.4	85.5	92.0	-	92.3
Code	LiveCodeBench (Pass@1-COT)	38.9	32.9	36.2	53.8	63.4	**65.9**
	Codeforces (Percentile)	20.3	23.6	58.7	93.4	96.6	96.3
	Codeforces (Rating)	717	759	1134	1820	2061	2029
	SWE Verified (Resolved)	**50.8**	38.8	42.0	41.6	48.9	49.2
	Aider-Polyglot (Acc.)	45.3	16.0	49.6	32.9	**61.7**	53.3
Math	AIME 2024 (Pass@1)	16.0	9.3	39.2	63.6	79.2	79.8
	MATH-500 (Pass@1)	78.3	74.6	90.2	90.0	96.4	97.3
	CNMO 2024 (Pass@1)	13.1	10.8	43.2	67.6	-	**78.8**

Since then, 100+ xPO variants have been proposed

Rollout Selection & Bias Correction:

- **SRPO** (Zhang et al., 2025c): history resampling
- **DAPO** (Yu et al., 2025): dynamic sampling
- **Dr.GRPO** (Liu et al., 2025): mitigates length bias
- **OPO** (Hao et al., 2025): optimal baseline to reduce gradient variance

Reward Shaping & Advantage Estimation:

- **EMPO** (Zhang et al., 2025b): fully unsupervised / minimizes predictive entropy
- **AAPO** (Xiong et al., 2025a): advantage momentum
- **BNPO** (Xiao et al., 2025): adaptively normalizes rewards via Beta distribution
- **SEED-GRPO** (Chen et al., 2025): ： uses semantic entropy to scale updates by uncertainty
- **GRPO-lead** (Zhang & Zuo, 2025): length-dependent accuracy, explicit penalties, difficulty-aware reweighting

Efficiency-Driven Methods:

- **CPPO** (Lin et al., 2025): pruning low-advantage completions
- **S-GRPO** (Dai et al., 2025b): early exit to cut redundancy
- **Ada-GRPO** (Wu et al., 2025): adaptive reasoning formats
- **GVPO** (Zhang et al., 2025a): analytical KL-constrained weighting
- **GRPO-λ** (Dai et al., 2025a): ddynamic switch between length-penalized vs length-agnostic rewards to avoid collapse

Gap between theory and practice

Q1 *Why is GRPO so effective?*

Q2 *What is the rationale for using the group mean to approximate the value function?*

Q3 *Can we provide finite-sample or asymptotic analyses regarding its convergence?*

Q4 *What is the optimal group size?*

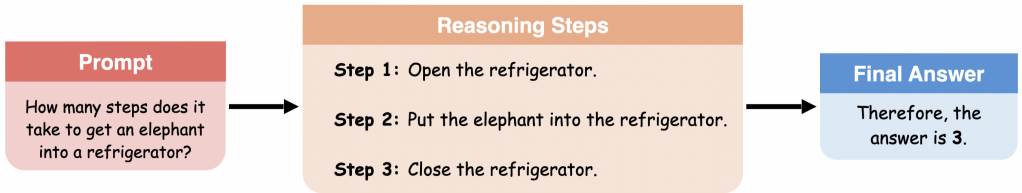
Background

1. **Reinforcement Learning**
2. **LLM Reasoning**
3. **U-statistics**

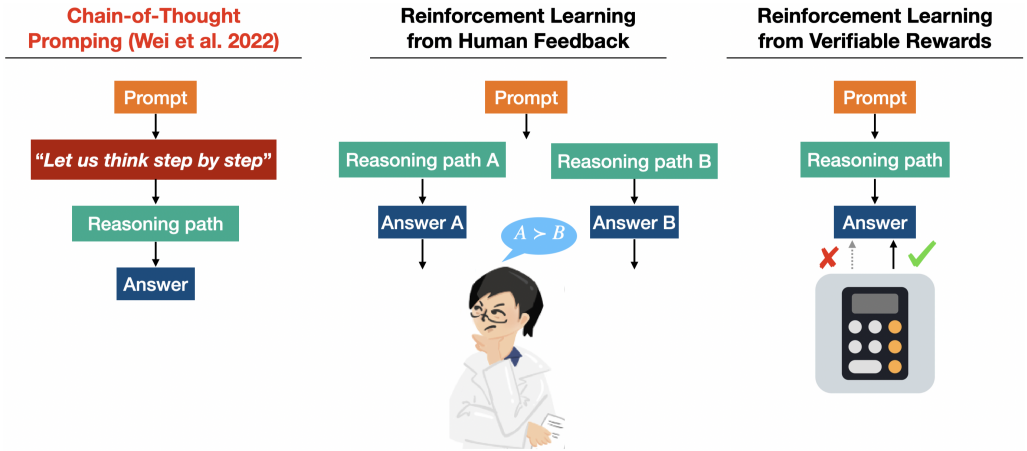
Background

1. Reinforcement Learning
2. LLM Reasoning
3. U-statistics

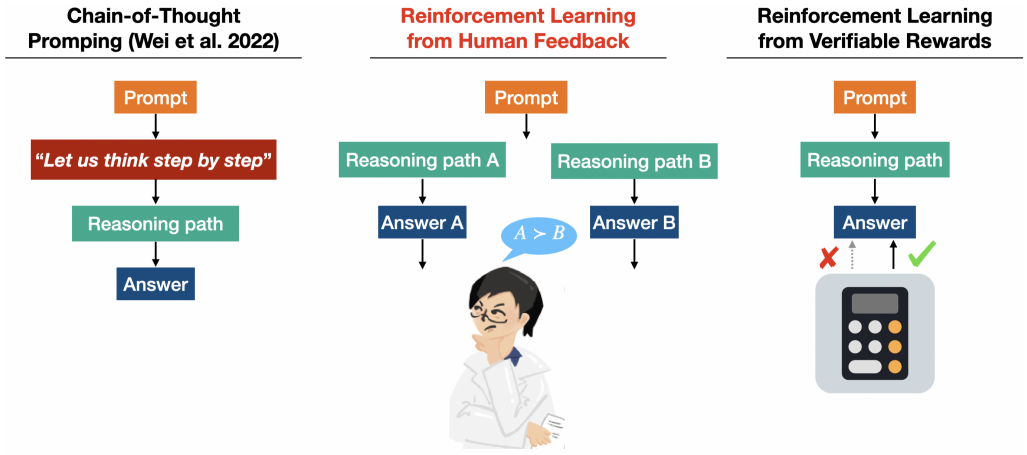
LLM Reasoning



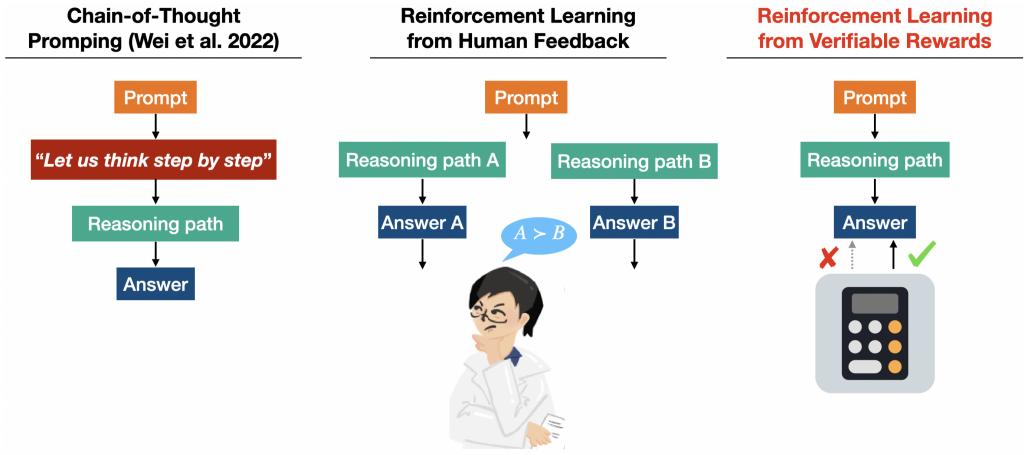
Approach I: Chain-of-Thought Prompting



Approach II: RLHF



Approach III: RLVR



RLVR (Lambert et al., 2024)

Motivation

Problem with RLHF:

- Human preferences are subjective
- Reward model can be inaccurate
- Hard to verify correctness

RLVR Solution:

- Use objective, verifiable rewards
- Automatic verification
- Ground truth available
- Scalable and reliable

Key Principle

Instead of "Is this good?" ask "Is this correct?"

Suitable for:

- **Code**: Run unit tests ✓
- **Math**: Check answer ✓
- **Logic**: Verify proof ✓

$$R_{\text{RLVR}}(x, y) = \begin{cases} +1 & \text{if correct} \\ 0 & \text{if wrong} \end{cases}$$

A simple policy gradient algorithm

- **Initialization**: θ arbitrary
- **For** $i = 1, 2, \cdots, n$ **do**:

  Sample an prompt X

  Generate a completion Y (reasoning trace + response) using policy π_θ

  Obtain the verifiable reward $Z = r(X, Y)$:

$$\theta \leftarrow \theta + \eta \nabla_\theta \log(\pi_\theta(Y|X))Z$$

end for

return θ

Variance reduction using a baseline

- **Initialization**: θ arbitrary
- **For** $i = 1, 2, \cdots, n$ **do**:

  Sample an prompt X

  Generate a completion Y using policy π_θ

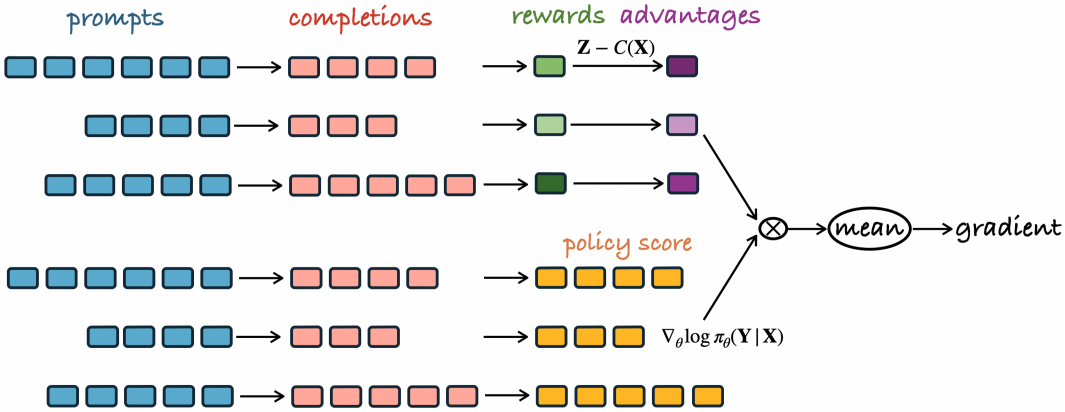
  Obtain the verifiable reward $Z = r(X, Y)$:

$$\theta \leftarrow \theta + \eta \nabla_\theta \log(\pi_\theta(Y|X))[Z - C(X)]$$

  end for

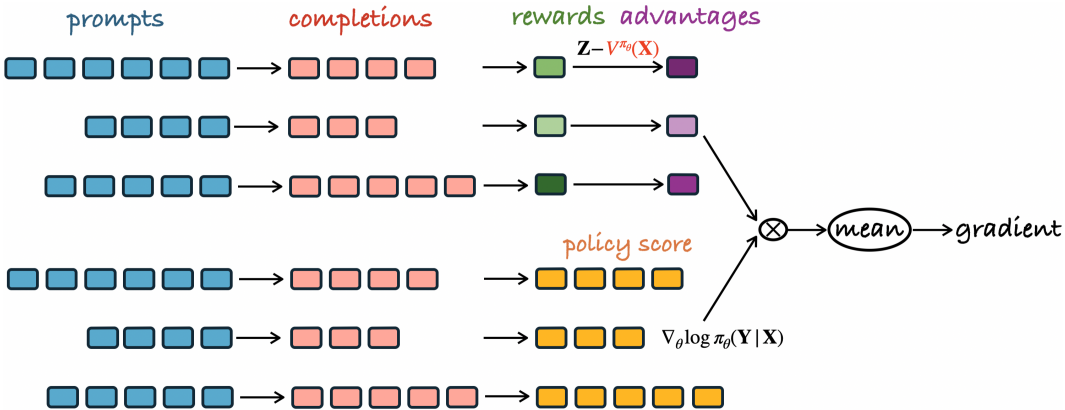
  return θ

A minibatch version



$$\theta \leftarrow \theta + \eta \text{ gradient}$$

Optimal choice of baseline



$$\theta \leftarrow \theta + \eta \text{ gradient}$$

Challenge: estimating and maintaining a value network is extremely computationally intensive in reasoning tasks!

Background

1. Reinforcement Learning
2. LLM Reasoning
3. **U-statistics**

Second-order U-statistics

- $\{\boldsymbol{X_i}\}_{\boldsymbol{i=1}}^{\boldsymbol{n}}$ denotes a sequence of i.i.d. random vectors
- Given a symmetric kernel function $\boldsymbol{h}$
- A second-order U-statistic is defined as

$$\boldsymbol{U} = \frac{\boldsymbol{1}}{\boldsymbol{n(n-1)}} \sum_{\boldsymbol{i \neq j}} \boldsymbol{h}(\boldsymbol{X_i}, \boldsymbol{X_j}).$$

- Hoeffding decomposition:

$$\boldsymbol{U} = \boldsymbol{h_0} + \underbrace{\frac{\boldsymbol{2}}{\boldsymbol{n}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} [\boldsymbol{h_1}(\boldsymbol{X_i}) - \boldsymbol{h_0}]}_{\text{first-order term } O_p(\boldsymbol{n}^{-1/2})} + \underbrace{\frac{\boldsymbol{1}}{\boldsymbol{n(n-1)}} \sum_{\boldsymbol{i \neq j}} [\boldsymbol{h}(\boldsymbol{X_i}, \boldsymbol{X_j}) - \boldsymbol{h_1}(\boldsymbol{X_i}) - \boldsymbol{h_1}(\boldsymbol{X_j}) + \boldsymbol{h_0}]}_{\text{second-order term } O_p(\boldsymbol{n}^{-1})},$$

where $\boldsymbol{h_0} = \mathbb{E}[\boldsymbol{h}(\boldsymbol{X_1}, \boldsymbol{X_2})]$ is the expectation of the kernel and $\boldsymbol{h_1}(\boldsymbol{x}) = \mathbb{E}[\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{X_2})]$ denotes the first-order projection.

Bring it all together: GRPO

Motivation

Problem with PPO:

- a separate value model/critic increases GPU memory.
- a poorly learned value model makes advantages noisy, destabilizing updates.

GRPO Solution:

- No value network needed
- Uses group-based advantages

Key Idea

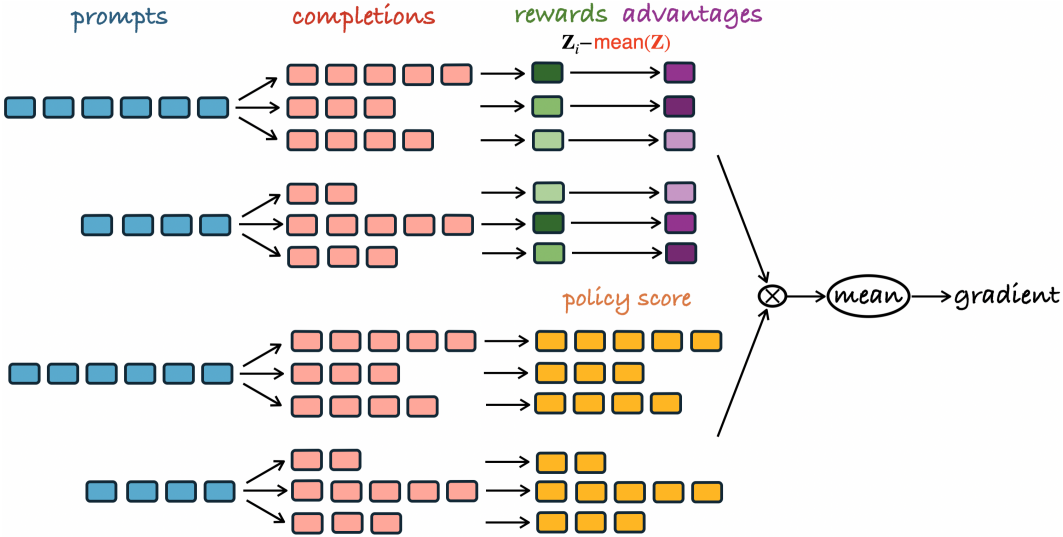
Instead of comparing to a baseline **by model**, compare responses **within a group**:

1. Generate **multiple responses** for same prompt
2. Score all responses with reward model
3. Use **group statistics** for advantage
4. Update policy based on relative quality

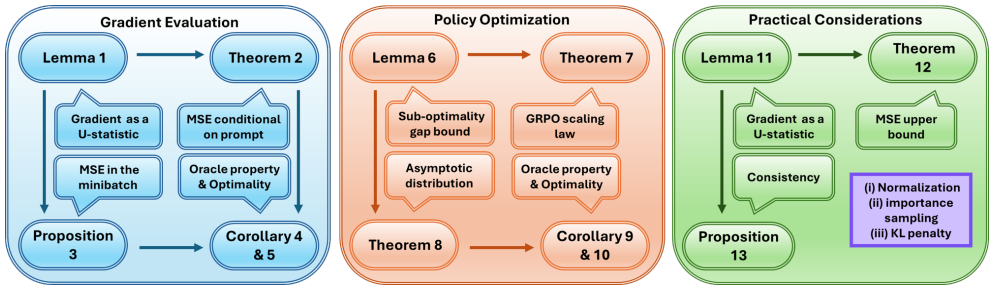
$$\text{Advantage}_i = \mathbf{Z}_i - \text{mean}(\mathbf{Z})$$

No separate value network needed!

GRPO-type algorithm



Demystify GRPO: Our results



Demystify GRPO (Cont'd)

Q1 *Why is GRPO so effective?*

Q2 *What is the rationale for using the group mean to approximate the value function?*

Q3 *Can we provide finite-sample or asymptotic analyses regarding its convergence?*

Q4 *What is the optimal group size?*

GRPO's gradient is a U-statistic

- $\boldsymbol{G}$ = no. of sampled completions per prompt.
- GRPO-type gradient:

$$\widehat{g}_{\mathrm{GRPO}}(\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{G-1}\sum_{g=1}^{G} \underbrace{\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{Y}^{(g)} \mid \boldsymbol{x})}_{\text{policy score}} \underbrace{[\boldsymbol{Z}^{(g)} - \mathbf{mean}(\boldsymbol{Z})]}_{\text{advantage}}.$$

Lemma

$\widehat{g}_{\mathrm{GRPO}}(\boldsymbol{x};\boldsymbol{\theta})$ *can be written as a second-order U-statistic.*

According to Hoeffding decomposition

$$\widehat{g}_{\mathrm{GRPO}}(\boldsymbol{x};\boldsymbol{\theta}) = \text{true gradient at } \boldsymbol{x} + \underbrace{\widehat{g}_{\mathrm{oracle}}(\boldsymbol{x};\boldsymbol{\theta}) - \mathbb{E}[\widehat{g}_{\mathrm{oracle}}(\boldsymbol{x};\boldsymbol{\theta})]}_{\textit{first-order term } O_p(\boldsymbol{G}^{-1/2})} + \underbrace{\text{second-order term}}_{O_p(\boldsymbol{G}^{-1})},$$

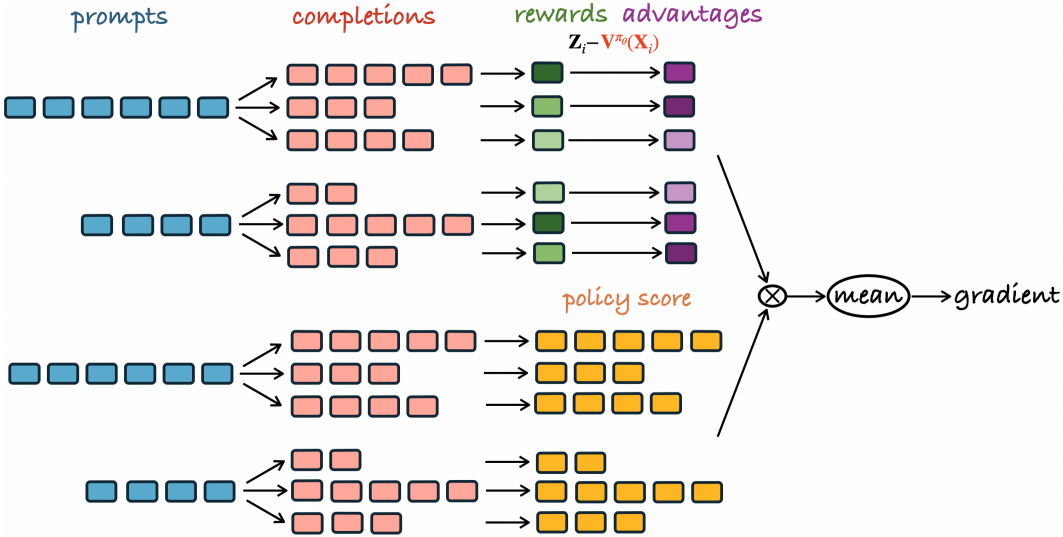
the **oracle** algorithm is a hypothetical algorithm with access to oracle value function.

- $\boldsymbol{B}$ denotes the number of prompts sampled per training iteration
- Consider the minibatch version $\widehat{g}_{\mathrm{GRPO}}(\boldsymbol{\theta}) = \frac{1}{B}\sum_{b=1}^{B} \widehat{g}_{\mathrm{GRPO}}(\boldsymbol{X}^{(b)};\boldsymbol{\theta})$

Theorem

$$MSE(\widehat{g}_{GRPO}(\boldsymbol{\theta})) = MSE(\widehat{g}_{oracle}(\boldsymbol{\theta})) + O\Big(\frac{\mathbb{E}\|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X})\|^2}{BG^2}\Big)$$

Oracle algorithm



Demystify GRPO (Cont'd)

Q1 *Why is GRPO so effective?*

Q2 *What is the rationale for using the group mean to approximate the value function?*

A2 Use of group mean provides a **first-order approximation** to the gradient of the **oracle algorithm** with access to the value function

Q3 *Can we provide finite-sample or asymptotic analyses regarding its convergence?*

Q4 *What is the optimal group size?*

Demystify GRPO (Cont'd)

Q1 *Why is GRPO so effective?*

Q2 *What is the rationale for using the group mean to approximate the value function?*

Q3 *Can we provide finite-sample or asymptotic analyses regarding its convergence?*

Q4 *What is the optimal group size?*

Demystify GRPO (Cont'd)

Corollary (Oracle property)

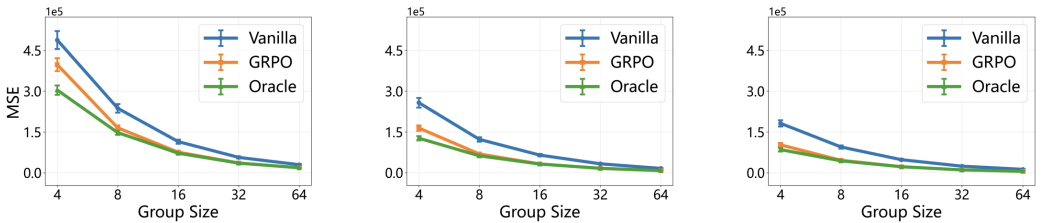
As $G \to \infty$, both (i) **MSE** *of the GRPO-type gradient and (ii)* **suboptimality gap** *of its induced policy are asymptotically equivalent to those of the* **oracle** *algorithm.*

Corollary (Optimality)

As $G \to \infty$, both (i) the MSE of the GRPO-type gradient and (ii) the suboptimality gap of its induced policy are asymptotically no larger than those of **any** *algorithm using a generic baseline $\boldsymbol{C}(\boldsymbol{X})$. In particular, the gradient MSE is strictly smaller than that of the* **vanilla** *algorithm with zero baseline, under mild conditions.*

In DeepSeekMath, $G = \mathbf{64}$, which is sufficiently large to approximate the asymptotics.

Experiment I: Gradient evaluation



(a) Base model *(b) Instruct model* *(c) ICL model*

Figure 4: MSEs of three policy gradient estimators (vanilla, GRPO-type, and oracle) under three model configurations (base, instruct and in-context learning (ICL)) for different group sizes. Error bars represent 95% confidence intervals of the empirically estimated MSEs.

Demystify GRPO (Cont'd)

Q1 *Why is GRPO so effective?*

A1 Because it achieves **oracle** performance without value network and is asymptotically **optimal** in both **MSE** and **suboptimality gap**.

Q2 *What is the rationale for using the group mean to approximate the value function?*

Q3 *Can we provide finite-sample or asymptotic analyses regarding its convergence?*

Q4 *What is the optimal group size?*

Demystify GRPO (Cont'd)

Q1 *Why is GRPO so effective?*

Q2 *What is the rationale for using the group mean to approximate the value function?*

Q3 *Can we provide finite-sample or asymptotic analyses regarding its convergence?*

A3 Yes – this is exactly what we establish in the paper. In particular, we establish parameter consistency and derive the asymptotic distribution of the suboptimality gap (a mixture of χ^2 random variables) without requiring parameter identifiability.

Q4 *What is the optimal group size?*

Demystify GRPO (Cont'd)

Q1 *Why is GRPO so effective?*

Q2 *What is the rationale for using the group mean to approximate the value function?*

Q3 *Can we provide finite-sample or asymptotic analyses regarding its convergence?*

Q4 *What is the optimal group size?*

A4 We derive a **scaling law**

(i) delineates how GRPO's performance depends on the group size

(ii) identifies the optimal group size that maximizes its performance

GRPO scaling law

Theorem

GRPO's sub-optimality upper bound depend on (i) the number of sampled prompts B and (ii) the number of sampled completions G per prompt through the following:

$$\frac{c_1}{B} + \frac{c_2}{BG} + \frac{c_3}{BG^2},$$

where c_1, c_2, c_3 depend only on the **data generating process** *and* **geometry of the policy space**.
Under a fixed sampling budget $N = BG$ per iteration or a total sampling budget $\boldsymbol{N} = nBG$, the optimal group size G^* that minimizes the upper bound is:

$$G^* = \sqrt{\frac{c_3}{c_1}}.$$

GRPO scaling law

Theorem

GRPO's sub-optimality upper bound depend on (i) the number of sampled prompts B and (ii) the number of sampled completions G per prompt through the following:

$$\frac{c_1 \boldsymbol{G}}{N} + \frac{c_2}{N} + \frac{c_3}{N\boldsymbol{G}},$$

where c_1, c_2, c_3 depend only on the **data generating process** *and* **geometry of the policy space**.
Under a fixed sampling budget $N = BG$ per iteration or a total sampling budget $\boldsymbol{N} = nBG$, the optimal group size G^* that minimizes the upper bound is:

$$G^* = \sqrt{\frac{c_3}{c_1}}.$$

Universality of G^*

The optimal group size G^* is **universal**:

- independent of the **budget** N
- independent of the **number of iterations** n
- independent of the **learning rate schedule**

depending solely on the **data generating process** and **geometry of the policy space**.

Experiment II: Universality across training iterations

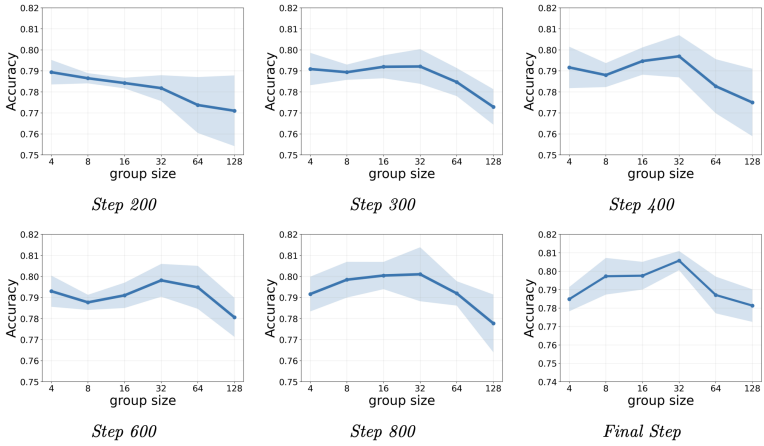


Figure 5: Test accuracy of GRPO-fine-tuned models at different training steps with a fixed sampling budget of $N = B \times G = 1024$ per prompt. Both training and evaluation are conducted on GSM8K. Each curve shows accuracy as a function of the group size $G \in \{4, 8, 16, 32, 64, 128\}$. Results are averaged over five independent runs, with shaded regions visualizing 95% confidence bands.

Thank You! ☺

Demystifying GRPO: Its Policy Gradient is a U-Statistic