MDPO: Multi-Granularity Direct Preference Optimization for Mathematical Reasoning

Authors: Yunze Lin, Yunze Lin

Date: 2025-06-10T18:28:59+00:00

Abstract

Mathematical reasoning presents a significant challenge for Large Language Models (LLMs) as it requires ensuring the correctness of each reasoning step. Researchers have been strengthening the mathematical reasoning abilities of LLMs through supervised fine-tuning, but due to the inability to suppress incorrect outputs, illusions can easily arise. Recently, Direct Preference Optimization (DPO) has been widely adopted for aligning human intent by using preference data to prevent LLMs from generating incorrect outputs. However, it has shown limited benefits in long-chain mathematical reasoning, mainly because DPO struggles to effectively capture the differences between accepted and rejected answers from preferences in long-chain data. The inconsistency between DPO training and LLMs' generation metrics also affects the effectiveness of suppressing incorrect outputs. We propose the Multi-Granularity Direct Preference Optimization (MDPO) method, optimizing the mathematical reasoning of LLMs at three granularities: Solution2Solution, Inference2Inference, and Step2Step. Solution2Solution focuses on the correctness of entire long-chain reasoning; Inference2Inference concentrates on logical reasoning between steps; Step2Step corrects computational errors in steps, enhancing the computational capabilities of LLMs. Additionally, we unify the training objectives of the three granularities to align with the generation metrics. We conducted experiments on the open-source models Qwen2 and Llama3, achieving improvements of 1.7% and 0.9% on the GSM8K dataset, and 2.3% and 1.2% on the MATH dataset, outperforming DPO and other DPO variant methods. Furthermore, we also provide a pipeline for constructing MDPO training data that is simple and does not require manual annotation costs.

Full Text

Preamble

MDPO: Multi-Granularity Direct Preference Optimization for Mathematical Reasoning

Yunze Lin, School of Artificial Intelligence
Beijing University of Posts and Telecommunications
Beijing
linyunze@bupt.edu.cn

Abstract

Mathematical reasoning presents a significant challenge for Large Language Models (LLMs) as it requires ensuring the correctness of each reasoning step. While researchers have strengthened LLMs' mathematical reasoning abilities through supervised fine-tuning, these models often suffer from hallucinations and performance saturation due to their inability to suppress incorrect outputs. Recently, Direct Preference Optimization (DPO) has gained traction for aligning models with human intent by using preference data to prevent generation of incorrect outputs. However, DPO shows limited benefits in long-chain mathematical reasoning, primarily because it struggles to effectively capture differences between accepted and rejected answers in lengthy reasoning chains. Moreover, the inconsistency between DPO' s training objective and LLMs' generation metrics further undermines its effectiveness at suppressing incorrect outputs.

We propose Multi-Granularity Direct Preference Optimization (MDPO), which optimizes mathematical reasoning at three granularities: Solution-to-Solution, Inference-to-Inference, and Step-to-Step. Solution-to-Solution focuses on the correctness of entire long-chain reasoning; Inference-to-Inference concentrates on logical reasoning between steps; and Step-to-Step corrects computational errors within steps, thereby enhancing LLMs' computational capabilities. Additionally, we unify the training objectives across all three granularities to align with generation metrics. Experiments on open-source models Qwen2 and Llama3 demonstrate improvements of 1.7% and 0.9% on GSM8K, and 2.3% and 1.2% on MATH, outperforming DPO and other DPO variants. Furthermore, we provide a simple pipeline for constructing MDPO training data that requires no manual annotation.

Introduction

Mathematical reasoning is considered a critical long-chain reasoning capability for large language models (LLMs). This task is particularly challenging because it typically requires extensive chains of thought involving numerous reasoning steps, where any single error can lead to an incorrect final result. Many studies have utilized additional math word problem (MWP) data to conduct supervised fine-tuning (SFT) of LLMs to improve their mathematical reasoning abilities

[?, ?, ?]. However, models often experience hallucinations during fine-tuning, leading to performance saturation [?]. On one hand, SFT struggles to provide fine-grained supervision signals; on the other hand, it cannot effectively suppress the probability of undesirable outputs, making models more prone to errors in long-chain reasoning. Therefore, developing methods that provide fine-grained supervision while suppressing incorrect outputs is crucial.

Recently, Direct Preference Optimization (DPO) has emerged as an effective alignment method that uses preference data triplets (x, y_w, y_l) to increase the probability of human-preferred answers y_w while decreasing the probability of rejected answers y_l [?]. While DPO proves effective in casual chat benchmarks, it struggles with long-chain mathematical reasoning. Models trained with DPO perform poorly at distinguishing correct from incorrect mathematical solutions, often failing to accurately identify errors in incorrect solutions. For instance, when a model generates initially correct steps before making an error, DPO may inadvertently lower the probability of those correct earlier steps. This indicates that DPO cannot accurately pinpoint detailed errors in incorrect solutions, thereby hindering reasoning improvement.

Moreover, as highlighted in [?], during DPO training, satisfying the reward ranking $r(x, y_w) > r(x, y_l)$ does not necessarily imply satisfying the likelihood ranking $p_\theta(y_w|x) > p_\theta(y_l|x)$. In fact, only about 50% of triplets meet this condition, stemming from the inconsistency between fine-tuning objectives and downstream task requirements. In mathematics education, effective teachers clearly identify whether errors stem from derivation mistakes or calculation errors, provide correct solutions, and encourage reflection. Drawing inspiration from this pedagogical approach, we propose Multi-granularity mathematical Direct Preference Optimization (MDPO) that provides models with supervision signals ranging from coarse to fine, while unifying fine-tuning representation with final reasoning tasks to enhance both reasoning and computational abilities.

Specifically, LLMs consider the entire chain of reasoning for solving MWPs as a solution composed of multiple reasoning steps, as shown in Fig.~1 (left), where solution $= \text{step}_1, \ldots, \text{step}_n$. We define the generation from step_k to step_{k+1} as one inference. As shown in Fig.~1 (right), our method performs preference optimization at three granularities: Solution-to-Solution (Sol2Sol), Inference-to-Inference (Infer2Infer), and Step-to-Step. Sol2Sol provides complete reasoning chains as supervision signals, consistent with DPO, offering coarse-grained supervision. Infer2Infer locates unreliable inferences inf_{lose} and corrects the reasoning process to obtain inf_{win}, providing fine-grained supervision. Step-to-Step locates steps with computational errors, $\text{step}_{\text{lose}}$, and corrects them to obtain step_{win}, improving computational capabilities. These objectives are uniformly defined as: given problem x and previous k steps $\text{step}_{0\sim k}$, continue reasoning until generating the answer. This transformation of mathematical reasoning into a text completion task ensures consistency between fine-tuning and downstream objectives.

We conducted experiments using two popular open-source models, Qwen2 [?] and Llama3 [?], on the GSM8K [?] and MATH [?] datasets. With Qwen2-7B-Instruct, we achieved accuracy improvements of 1.7% and 0.9% on GSM8K, and 2.3% and 1.2% on MATH, surpassing DPO [?] and its variant Step-DPO [?], demonstrating MDPO' s potential. Additionally, we provide a method for automatically constructing multi-granularity preference data pairs without manual annotation.

2.1 Mathematical Reasoning

With increasing pre-training scale, LLMs have demonstrated strong reasoning abilities. [?] proposed Chain-of-Thought prompting, which enables LLMs to output step-by-step reasoning processes through carefully designed examples, improving both accuracy and confidence in solving math word problems. Subsequent researchers attempted to further enhance mathematical reasoning by strengthening Chain-of-Thought [?, ?, ?]. However, these efforts remain hindered by hallucinations and computational limitations inherent to LLMs.

[?] introduced Program-aided Language models (PAL), which transform math word problem solving into Python code generation tasks, ensuring correctness of intermediate calculations. However, this approach requires relatively strong code generation capabilities; otherwise, it may introduce additional noise during code generation. Additionally, many researchers have utilized data augmentation methods to construct higher-quality MWP data for supervised fine-tuning of LLMs, yielding significant improvements in reasoning abilities [?, ?, ?, ?, ?]. Yet the hallucination issue remains unresolved, partly because supervised learning only increases the probability of generating correct answers without suppressing incorrect outputs. Therefore, ensuring both the generation of correct answers and suppression of undesirable outputs is essential for improving LLM performance.

2.2 RLHF

Reinforcement Learning from Human Feedback (RLHF) is commonly used to train LLMs to align with human values and intentions, including suppressing harmful information and hallucinations [?, ?, ?, ?]. Recent studies have explored using reinforcement learning to reduce hallucinations in mathematical reasoning, involving training a reward model and then optimizing the policy model to maximize this reward. [?] proposed the Process Reward Model (PRM) to evaluate the quality of each reasoning step, thereby enhancing mathematical reasoning abilities. However, these methods require high-quality manually annotated data to train the reward model and involve complex, laborious training processes.

Recently, researchers have explored simpler offline algorithms, notably Direct Preference Optimization (DPO) [?]. DPO learns policy models directly from preference data by parameterizing the reward function in RLHF, eliminating

the need for an explicit reward model. This simple, stable method has inspired various variants [**?**, **?**, **?**, **?**]. One variant, Step-DPO, introduces step-level supervised signals to help LLMs accurately locate errors, similar to our approach [**?**]. However, its implicit reward is constructed from the logarithm of the likelihood ratio between responses from the current policy model and the SFT model, which does not directly align with generation metrics, resulting in poor performance. Our method focuses on advancing LLMs' computational capabilities. [**?**] proposed the SimPO method, which directly uses average log-likelihood as a reward for preference learning, aligning training with reasoning and making it simple and efficient. Inspired by this research, we propose MDPO for mathematical long-chain reasoning, providing LLMs with multi-level supervised signals.

3 Background: Simple Preference Optimization (SimPO)

SimPO is a popular preference optimization method that requires neither a reward model nor a reference model. It addresses the discrepancy between the reward optimized during training and the generation metrics used during inference, significantly outperforming DPO and other variants. The algorithm's core is aligning the reward function in the preference optimization objective with the generation metric. Specifically, during generation, a policy model π_θ generates a sequence that approximates the maximization of average log-likelihood, defined as:

$$p_\theta(y|x) = \log \pi_\theta(y|x) = \sum_i \log \pi_\theta(y_i|x, y_{<i})$$

SimPO directly uses p_θ from equation (1) as the reward, aligning it with the likelihood metric used for guided generation, while introducing a length normalization term to prevent the model from generating longer but lower-quality sequences:

$$r_{\text{SimPO}}(x, y) = \frac{\log \pi_\theta(y|x)}{|y|} = \frac{\sum_i \log \pi_\theta(y_i|x, y_{<i})}{|y|}$$

where β is a constant controlling the magnitude of reward difference. In addition, SimPO introduces a target reward margin term $\gamma > 0$ to ensure that the reward $r(x, y_w)$ of a winning response exceeds the reward $r(x, y_l)$ of a losing response by at least γ:

$$p(y_w \succ y_l|x) = \sigma(r(x, y_w) - r(x, y_l) - \gamma)$$

The loss function is represented as:

$$\mathcal{L}_{\text{SimPO}}(\pi_\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \log \sigma \left(\frac{\log \pi_\theta(y_w|x)}{|y_w|} - \frac{\log \pi_\theta(y_l|x)}{|y_l|} - \gamma \right)$$

4 Method

[Figure 1: see original paper]

Figure 1: (Left) Given a math word problem, which when fully reasoned through the solution chain can be broken down into a series of solutions = Step_1, ···, Step_n, we define the generation from Step_k to Step_{k+1} as one inference step. (Right) MDPO provides LLMs with three granularities of supervision signals: Solution-to-Solution (Sol2Sol), Inference-to-Inference (Infer2Infer), and Step-to-Step, optimizing the model using preference data. Sol2Sol constructs preferences for complete inference chains; Infer2Infer identifies faulty inferences in the chain and corrects them, while Step-to-Step focuses on identifying and correcting computational errors at a specific step in the inference process.

4.1 MDPO

Although DPO has proven effective in chat benchmarks, it yields only marginal improvements for long-chain reasoning tasks such as mathematical problem solving. This limitation arises because rejecting an entire undesirable answer in DPO may discard preceding correct reasoning steps, introducing significant noise that negatively impacts training. Analogous to how teachers correct students by pinpointing specific errors rather than dismissing entire answers, our proposed MDPO provides more detailed supervision across three granularities: Solution-to-Solution (Sol2Sol), Inference-to-Inference (Infer2Infer), and Step-to-Step.

Sol2Sol. In solving mathematical word problems, the entire reasoning path from scratch to the final answer is referred to as a Solution, which aligns with DPO' s format. Although rejecting the entire undesirable answer can have negative consequences, we retain this approach to ensure the model can generate complete reasoning chains.

Infer2Infer. Each solution can be decomposed into a sequence of reasoning steps, solution = $\text{step}_1, \ldots, \text{step}_n$, where step_i is the i-th reasoning step. We define the generation from step_k to step_{k+1} as an Inference. Given a prompt x and a series of initial correct reasoning steps $\text{step}_{1\sim k-1} = \text{step}_1, \ldots, \text{step}_{k-1}$, Infer2Infer provides fine-grained supervision signals for the generation process inf_{k-1} from step_{k-1} to step_k, maximizing the probability of generating the correct next reasoning step step_{win} while minimizing the probability of generating the incorrect reasoning step $\text{step}_{\text{lose}}$, thereby enhancing the model' s reasoning capabilities.

Step-to-Step. Since large models often face computational challenges during reasoning that lead to overall failure, we provide preference data for accepting and rejecting steps. Specifically, for a $\text{step}^k_{\text{lose}}$ with computational error, we construct the correct calculation step step_{win} to rectify the model' s computational errors and enhance its computational abilities.

4.2 Objective

To maintain consistency between fine-tuning and downstream tasks, we convert mathematical word problem reasoning into the following format: given a math word problem and the first k steps, the model must continue writing based on the problem and these steps to obtain the final answer. This format allows our multi-granularity optimization goals to align with the ultimate solving objectives. For Sol2Sol, this can be considered as providing the problem and the first 0 steps, requiring the model to generate all reasoning steps, which aligns with the final solving of the math word problem. We use "Let' s think step by step." as the 0-th step to guide the model in reasoning. For Infer2Infer, this can be viewed as providing the problem and the first $i-1$ steps, asking the model to generate the i-th step and subsequent reasoning steps. For Step-to-Step, likewise, this can be seen as providing the problem and the first $i-1$ steps, demanding the model to generate the i-th step and subsequent reasoning paths.

$$\mathcal{L}_{\text{MDPO}}(\pi_\theta) = -\mathbb{E}_{(x,s_{0\sim k-1},y_w,y_l)\sim\mathcal{D}} \log\sigma\left(\frac{\log\pi_\theta(y_w|(x,s_{0\sim k-1}))}{|y_w|} - \frac{\log\pi_\theta(y_l|(x,s_{0\sim k-1}))}{|y_l|} - \gamma\right)$$

where x represents the math word problem to be solved, $s_{0\sim k-1}$ represents the first k solving steps from step_0 to step_{k-1}, y_w represents a series of correct solving steps from step_k to step_n, and y_l represents a series of incorrect solving steps from step_k to step_n.

4.3 Data Construction

Sol2Sol. In the Sol2Sol component, we use LLMs to sequentially generate reasoning for questions and require the model to prepend "[Step i]" before each step. Subsequently, we sample k reasoning paths and verify the generated paths based on labels in the dataset, adopting complete reasoning paths with correct final answers as preferred responses and paths with incorrect final answers as rejected responses, consistent with simple DPO. Moreover, we select questions in the generated paths that have both correct and incorrect answers, as these are more challenging for the model and more effective at enhancing reasoning abilities.

Infer2Infer. In the Infer2Infer component, we utilize the erroneous reasoning paths selected from the aforementioned problems as our foundation. These reasoning paths are divided into steps and reorganized into multiple windows $W = (w_0, \ldots, w_i)$, where $w_i = (\text{step}_0, \text{step}_1, \text{step}_2, \ldots, \text{step}_i)$. Subsequently, we leverage LLMs to generate and sample k reasoning paths. For each w_i, we define its error rate as the number of erroneous reasoning paths divided by the total number of paths. Based on this, we define an unreliable step as a step where $\text{error}(w_i) > \text{error}(w_{i-1})$. We consider that step_i increases the reasoning error rate and negatively impacts the final reasoning process. As a result, the transition from step_{i-1} to step_i is regarded as an unreliable inference

inf_{lose}. Next, we continue generating using w_{i-1}, sampling the reliable reasoning path with the final correct answer as inf_{win}, to construct preference data pairs $(x||w_{i-1}, \text{inf}_{\text{win}}, \text{inf}_{\text{lose}})$, where $||$ denotes concatenation.

Step-to-Step. In the Step-to-Step component, we also use the selected problems mentioned above and add new problems involving complex calculations, constructed by replacing original problems with more complex numbers. Subsequently, we use LLMs for reasoning, sample the reasoning paths, and segment the steps. Using prompts, we employ GPT-4 to search for the first step $\text{step}_{\text{lose}}$ where a calculation error occurred, correct it to obtain step_{win}, and continue generating the final answers. Through answer verification, we ensure that the LLMs made correct modifications. We construct preference data $(x||\text{step}_{0\sim k-1}, \text{step}_{\text{win}}, \text{step}_{\text{lose}})$.

5 Experiments

Network Architecture. Our experiments employed two popular open-source models, Qwen2 [**?**] and Llama3 [**?**]. Due to computational resource limitations, we utilized smaller-scale models: Qwen2-7B-Instruct and Meta-Llama-3-8B-Instruct. Our choice to employ instruction-tuned models rather than base models for direct training stems from common practice in reinforcement learning pipelines, where supervised fine-tuning typically serves as a crucial warm-start initialization phase before RL-based optimization.

Datasets. For evaluation, we used the widely adopted GSM8K [**?**] and MATH [**?**] datasets, with accuracy serving as the evaluation metric. The MATH test set contains 5,000 mathematical problems spanning 5 difficulty levels and 7 subjects, including algebra, counting and probability, geometry, intermediate algebra, number theory, prealgebra, and precalculus. Problems in GSM8K are generally easier than those in MATH. Additionally, we use the GSM-HARD [**?**] dataset to examine MDPO' s improvement in computational capabilities. Our training dataset is based on the training sets of GSM8K and MATH, constructed following the method described in Section 4.3, comprising a total of 30,000 preference data pairs.

Implementation Details. We perform MDPO on the models for 8 epochs with a global batch size of 128 and a learning rate of 5e-7. The hyperparameter β is set to 0.4. We use the AdamW optimizer with a cosine learning rate scheduler, with the warmup ratio set to 0.1.

6.1 Main Results

The results of MDPO on Qwen2-7B-Instruct and Llama3-8B-Instruct are shown in Table 1. MDPO demonstrates significant improvements for both models, with Qwen2-7B-Instruct achieving a 1.7% accuracy increase on GSM8K and a 2.3% increase on MATH. Similarly, Llama3-8B-Instruct with MDPO achieved a 0.9% accuracy increase on GSM8K and a 1.2% increase on MATH. These results demonstrate that MDPO can effectively enhance LLMs' ability to solve MWPs.

[Figure 2: see original paper]

Figure 2: MDPO has different training objectives from DPO and Step-DPO, primarily in reward formulation and margin setting. On GSM8K and MATH tasks, MDPO' s consistency between training and downstream objectives has led to performance improvements, mainly reflected in the increased generation probability of accepted answers. The Win Rate indicates the proportion where $p_\theta(y_w|x) > p_\theta(y_l|x)$.

6.2 Comparison with Other Methods

We compared MDPO with DPO and Step-DPO methods on GSM8K and MATH datasets, with results shown in Table 2. The benefits of DPO are limited and significantly smaller than those of MDPO. Furthermore, MDPO outperforms Step-DPO on both datasets, with greater enhancement on MATH. This is because MDPO not only optimizes reasoning ability but also provides supervisory signals for computational power, making it more effective on complex datasets.

Due to computational resource limitations, we were only able to conduct experiments on small-scale LLMs. However, based on research on DPO and Step-DPO, methods proven effective on small-scale LLMs generally yield greater improvements on large-scale LLMs.

We also conducted experiments using the original SimPO method. Empirical results demonstrate that SimPO achieves consistent performance improvements over standard DPO across both datasets, attributable to its more direct reward signals. However, compared with our MDPO approach, SimPO exhibits inferior performance on both datasets, primarily due to its lack of fine-grained, multi-level supervision signals.

6.3 Ablation Study

We conducted ablation experiments on MDPO using Qwen2-7B-Instruct as the base model on the GSM8K dataset. We sequentially added three granularities of preference data for fine-tuning, with results shown in Table 3. Compared to the base model, fine-tuning at each granularity improved final performance. Notably, the Infer2Infer component contributed most to performance improvement, as it provides finer supervisory signals than Sol2Sol, making a greater contribution to enhancing reasoning abilities. The Step-to-Step component aims to enhance computational abilities. Since operations in GSM8K are relatively simple, Step-to-Step' s contribution may not be fully reflected. Additional experiments are described in Section 6.4.

6.4 Computation

In MDPO, we introduced Step-to-Step to enhance the model' s computational capabilities. We used GSM-HARD and MATH datasets to validate its effectiveness. In GSM-HARD, numbers in the GSM8K test set were replaced with

more complex digits, increasing computational difficulty. The MATH dataset itself involves complex operations including fractions. We conducted experiments with Qwen2-7B-Instruct and Step-to-Step data, with results shown in Table 4. Models fine-tuned using Step-to-Step data show significant improvements (+3.4 and +1.7 on GSM-HARD and MATH, respectively). Compared to DPO and Step-DPO methods, our approach shows clear advantages when dealing with computationally complex datasets, demonstrating that incorporating fine-grained supervision signals for computational validation is essential.

6.5 Training Objective

In DPO and Step-DPO, the training objective does not align with the downstream task objective, resulting in a higher probability of LLMs outputting rejected answers compared to accepted answers, contradicting the original intention of fine-tuning. We conducted experiments on the test set, and Fig.~2 shows the proportion of cases where LLMs output an accepted answer with higher probability after training with different methods. There is little difference between Step-DPO and DPO, as Step-DPO, while providing more detailed supervisory signals through modified optimization objectives, still fails to align training objectives with downstream tasks. In contrast, our MDPO method aligns the reward function with the generation metric and unifies fine-tuning with the final downstream task, thereby enhancing LLMs' responsiveness to rewards. Experimental results demonstrate that MDPO significantly increases the probability of LLMs outputting accepted answers, effectively reducing the occurrence of rejected answers.

7 Conclusion

We proposed Multi-granularity Direct Preference Optimization (MDPO), which provides three granularities of supervision signals for LLMs: Solution-to-Solution, Inference-to-Inference, and Step-to-Step. The method optimizes models using preference data. Solution-to-Solution ensures consistency between downstream tasks and fine-tuning, while Inference-to-Inference focuses on providing detailed problem-solving guidance for LLMs, accurately identifying logical errors in the reasoning process to improve long-chain reasoning abilities. Step-to-Step is dedicated to pinpointing computational errors in LLMs' inference process, enhancing foundational computational capabilities through data preferences.

Our method has been validated on two popular open-source LLMs, Qwen2 and Llama3, showing significant improvements across multiple mathematical reasoning datasets and generally outperforming DPO and Step-DPO methods. Specifically, on the MATH dataset, MDPO achieved a 2.3% improvement, surpassing the widely recognized Step-DPO method by 1.4%. Furthermore, experiments on the challenging GSM-HARD and MATH datasets demonstrate the effectiveness of Step-to-Step in enhancing LLMs' computational capabilities. Additionally,

our experiments on target consistency showcase the necessity of aligning training objectives with downstream tasks.

Due to computational resource limitations, our experiments have been conducted only on 7B-sized models. However, based on trends reported by other scholars [**?**, **?**, **?**], methods proven effective on small-scale models often lead to greater improvements when scaled to larger models. In the future, we plan to enrich our experimental results by testing on a wider range of large-scale models.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. arXiv preprint arXiv:2309.16609, 2023.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. arXiv preprint arXiv:2402.01306, 2024.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. arXiv preprint arXiv:2103.03874, 2021.

Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model. arXiv preprint arXiv:2403.07691, 2024.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. arXiv preprint arXiv:2406.18629, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe.

Let' s verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Zimu Lu, Aojun Zhou, Houxing Ren, Ke Wang, Weikang Shi, Junting Pan, Mingjie Zhan, and Hongsheng Li. Mathgenie: Generating synthetic data with question back-translation for enhancing mathematical reasoning of llms. arXiv preprint arXiv:2402.16352, 2024.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, et al. Improve mathematical reasoning in language models by automated process supervision. arXiv preprint arXiv:2406.06592, 2024.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math. arXiv preprint arXiv:2402.14830, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. Mathscale: Scaling instruction tuning for mathematical reasoning. arXiv preprint arXiv:2403.02884, 2024.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *Advances in Neural Information Processing Systems*, 37:34737–34774, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023.

Zengzhi Wang, Rui Xia, and Pengfei Liu. Generative ai for math: Part i–

mathpile: A billion-token-scale pretraining corpus for math. arXiv preprint arXiv:2312.17120, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. arXiv preprint arXiv:2210.03493, 2022.

Note: Figure translations are in progress. See original paper for figures.