Let's Reinforce Step by Step

Sarah Pan
MIT Primes
Phillips Academy
`span24@andover.edu`

Vladislav Lialin
University of Massachusetts Lowell
`vlialin@cs.uml.edu`

Sherin Muckatira
University of Massachusetts Lowell
`smuckati@cs.uml.edu`

Anna Rumshisky
University of Massachusetts Lowell
`arum@cs.uml.edu`

Abstract

While recent advances have boosted LM proficiency in linguistic benchmarks, LMs consistently struggle to reason correctly on complex tasks like mathematics. We turn to Reinforcement Learning from Human Feedback (RLHF) as a method with which to shape model reasoning processes. In particular, we explore two reward schemes, outcome-supervised reward models (ORMs) and process-supervised reward models (PRMs), to optimize for logical reasoning. Our results show that the fine-grained reward provided by PRM-based methods enhances accuracy on simple mathematical reasoning (GSM8K) while, unexpectedly, reducing performance in complex tasks (MATH). Furthermore, we show the critical role reward aggregation functions play in model performance. Providing promising avenues for future research, our study underscores the need for further exploration into fine-grained reward modeling for more reliable language models.

1   Introduction

While the linguistic ability of language models (LMs) as measured by NLP benchmarks has increased in recent years, [Brown et al., 2020, Touvron et al., 2023] improving their proficiency in complex reasoning tasks like mathematics remains a challenge [Yao et al., 2023a]. When prompted with a math problem, LMs will often generate solution steps that seem plausible but are neither correct nor follow logically from previous steps. We believe that the process used to train LMs is partially responsible for these *hallucinatory*, or in other words fallacious, responses [Maynez et al., 2020, Uesato et al., 2022, Lightman et al., 2023, Yang et al., 2023].

"Chain of Thought" (CoT) methods, which aim to instill step-wise reasoning at generation-time, increase mathematical performance [Wei et al., 2023]. Simultaneously, CoT prompting does not guarantee a logical response – little incentive is placed on logic or factual correctness in the original language modeling objective. Reinforcement Learning from Human Feedback (RLHF) [Ziegler et al., 2020], however, allows us to specify the criteria we optimize for, which in this case is logical reasoning. We explore the two reward schemes that have emerged recently – outcome-supervised reward models (ORMs) and process-supervised reward models (PRMs) [Uesato et al., 2022]. While Lightman et al. [2023] show that PRMs can identify correct solutions better than ORMs, they did not update a generator model. It is also important to note that our method differs from that of Yang et al. [2023], Wu et al. [2023] as we seek not to increase performance directly but rather to improve the logical stepping stones upon which a final answer is generated.

In lieu of using reward models as verifiers [Cobbe et al., 2021, Lightman et al., 2023], we combine CoT and RLHF to guide the internal reasoning process of the LM. We show that accuracy on
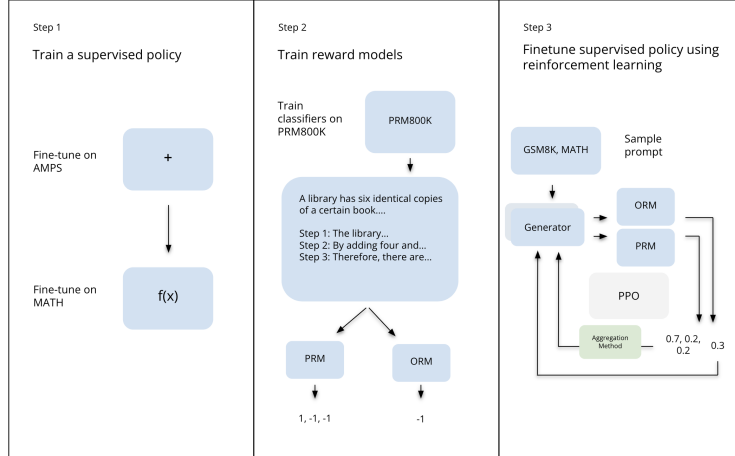
Figure 1: A visual guide to our method, which follows InstructGPT [Ouyang et al., 2022].

mathematical benchmarks MATH and GSM8K [Hendrycks et al., 2021, Cobbe et al., 2021], improves after policy optimization using ORMs and PRMs. Though PRM-based methods lead to decreases in MATH performance, they notably also increase performance on GSM8K by 33% (relative). Using outcome-supervised reward models does not improve performance on GSM8K but increases accuracy on MATH by 18%.

2 Method

Our overall framework resembles that of Ouyang et al. [2022]. Shown in Figure 1, the stages are supervised training, reward model training, and reinforcement learning with the learned reward model.

When performing RLHF, we look at the effects of various PRM reward aggregation schemes. We note that our exploration of aggregation methods is not exhaustive but provides insight into particularly promising methods. We hypothesize that the more fine-grained feedback provided by the PRM provides better signals than that from an ORM. When aggregated effectively, we expect that the PRM improves policy performance on mathematical benchmarks.

2.1 Supervised Finetuning and Reward Model Training

We first fine-tune the generator model on multi-step math solutions. Next, we train our reward models. To train the outcome-supervised reward model, an n-token long question and step-by-step solution are passed in as context. Only the logits corresponding to the last token are taken into account. [1] The label denotes whether that solution contains an incorrect reasoning step.

$$\mathcal{L}_i = \begin{cases} -\frac{1}{|S|} y_i \cdot \log p_i & i \in S \\ 0 & i \notin S \end{cases} \quad (1)$$

The process-supervised reward model is trained on a similar classification objective, but as shown in Equation (1), there exists a set S that contains indices corresponding to the last token in a reasoning step. All other tokens are masked when calculating the loss.

2.2 Reward Aggregation Methods

Using the ORM and PRM trained in phase two, we fine-tune our generator model using PPO [Schulman et al., 2017]. Our study delves into five distinct reward delivery paradigms: vanilla ORM,

---

[1]This is similar to GPT-1 [Radford and Narasimhan, 2018], where the hidden state of the last token is used as a representation vector for the entire text.

PRM-Avg, PRM-Prod, PRM-Max, and PRM-Min. For all the hyperparameters we use, look to Appendix A.

Vanilla ORM For the vanilla ORM method, we denote the probability of there being a misstep as r_{ORM}. While all other tokens receive an estimate of the KL divergence from the original policy as a reward, the last token in the response additionally receives r_{ORM}. This is shown in Equation (2) for an n-token long response.

$$r_t = \begin{cases} t \neq n & \text{KL-div}_t \\ t = n & \text{KL-div}_t + r_{\text{ORM}} \end{cases} \tag{2}$$

Aggregated PRMs The general equation for reward aggregation is shown by Equation (3). The PRM-Avg method uses $\texttt{aggregate} = \frac{1}{n}\sum_{i=0}^{n}$ for an n-token long sequence. Similarly, the PRM-Prod method uses $\texttt{aggregate} = \prod_{i=0}^{n}$.

The PRM-Max approach uses $\texttt{aggregate} = \max$, where the largest per-step reward is used for the entire generation. This method explores the idea that one decisively good reasoning step might be all that is needed to produce a correct final answer. Conversely, PRM-Min uses $\texttt{aggregate} = \min$ and penalizes the model according to the worst reasoning step.

$$r_t = \begin{cases} t \neq n & \text{KL-div}_t \\ t = n & \text{KL-div}_t + \texttt{aggregate}(\text{PRM}) \end{cases} \tag{3}$$

3 Experiments

3.1 Experimental Setup

We fine-tune OPT-1.3B [Zhang et al., 2022] as our generator model. We first train on the Auxiliary Mathematics Problems and Solutions (AMPS) dataset and MATH [Hendrycks et al., 2021] for one and ten epochs, respectively. Like Lewkowycz et al. [2022], we find that supervised fine-tuning is necessary to produce meaningful responses. We evaluate this model as the SFT Base model.

Then, we train two 300M-parameter DeBERTav3[2] reward models as a sequence-level classifier and a reasoning-step-level classifier, respectively. Both of these models are trained on the PRM800K dataset [Lightman et al., 2023], which consists of prompts from MATH [Cobbe et al., 2021], step-by-step responses generated by a model, and step-level labels from human labelers on whether each step is correct.

Following Yao et al. [2023b], we initialize the value function from a reward model with the same tokenizer. We use an OPT-1.3B [Zhang et al., 2022] model trained on the ORM objective. We choose to train our critic with the ORM objective to avoid the unnecessary complexities of using a PRM. During RLHF, we utilize prompts from both MATH and GSM8K. This provides the generator with greater exposure to math with varying levels of difficulty. Lastly, because PRM800K contains examples from the MATH test set, we filter the overlap to arrive at approximately five hundred questions. We evaluate on the GSM8K test set as is.

3.2 Results

Our main results are presented in Figure 2. We see that providing feedback on LM reasoning process through RLHF is a promising method with which to improve mathematical performance as judged by the accuracy of the final answer (up to 17% on MATH 33% on GSM8K, relative).

We find a strong dependence of accuracy on the reward aggregation method. The best performance on GSM8K results from the use of the PRM-Max method, and the best score on MATH comes from the vanilla ORM method. Though PRM-based methods (excluding PRM-Avg and PRM-Min) increase performance on GSM8K, they consistently decrease performance on MATH. The significant

---

[2]DeBERTav3 accuracy on the reward modeling dataset was much better than OPT-350M/1.3B in all our experiments.
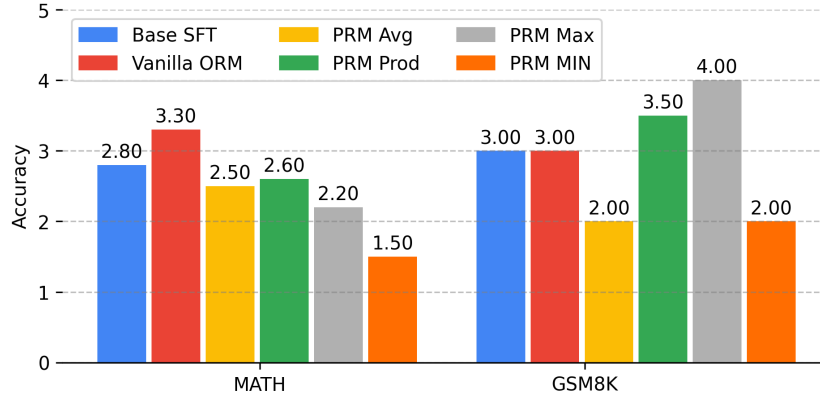
Figure 2: Results on MATH and GSM8K. PRM-based methods degrade performance on MATH but improve accuracy on GSM8K. ORM-based methods improve accuracy on MATH but decrease that on GSM8K.

decrease in performance after using PRM-Avg and PRM-Min points to the importance of choosing an appropriate aggregation method when using a process-supervised reward model.

The increase in MATH performance with the outcome-supervised reward model may stem from its holistic training, enabling better handling of complex problems, while its lack of exposure to simpler reasoning hinders decisive signaling for GSM8K problems. In contrast, the process-supervised reward model, also trained on PRM800K, prioritizes correctness on the step level, resulting in improved GSM8K performance but a decline in MATH performance. This hints that the fine-grained knowledge possessed by a PRM may be more akin to "real math" and generalize better onto different levels of complexity. Simultaneously, this implies potential limitations in the PRM's ability to effectively understand and reward complex mathematics. Lightman et al. [2023] use PRMs that are presumably much larger than ours and thus may be better at understanding complex patterns. This study suggests the need for deeper exploration into the trade-offs between ORM and PRM-based methods, especially in terms of varying model scales.

Effects of dataset mixing To verify our strategy of mixing GSM8K and MATH training sets in stage threes, we use either dataset with the PRM-Prod method and observe significant decreases in model ability. When only training on MATH, a 0.5% accuracy on GSM8K and a 2.6% on MATH is observed. Similarly, only training on GSM8K produces an accuracy of 1.5% on MATH and 1.5% on GSM8K. When compared with our results in Figure 2, it is clear that training on both datasets is beneficial for performance.

4   Conclusion

We have shown that the use of reinforcement learning with outcome-supervised and process-supervised reward can increase performance on complex reasoning tasks like mathematics. We observe up to 33% accuracy increase on GSM8K and a 18% increase on MATH. Additionally, we find a strong performance dependence on the reward aggregation function for PRM-based methods. For example, maximum-aggregated PRM lead to the best result on GSM8K dataset but resulted in worse-than-baseline performance on MATH. Moreover, the failure of the average-aggregated PRM highlights the significance of choosing the right aggregation function. In future work, we plan to explore the use of a non-aggregated step-by-step or, in general, fine-grained rewards directly in the reinforcement learning objective. Additionally, as Lightman et al. [2023] found model size to be a crucial factor for step-by-step rewards to be beneficial, we plan to scale up our experiments to larger models and datasets. We hope that our study encourages more research in fine-grained reward modeling – especially in relation to the effectiveness of various aggregation methods – and brings us closer to more reliable language models.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

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, 2021.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization, 2020.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A. Smith, Mari Ostendorf, and Hannaneh Hajishirzi. Fine-grained human feedback gives better rewards for language model training, 2023.

Shentao Yang, Shujian Zhang, Congying Xia, Yihao Feng, Caiming Xiong, and Mingyuan Zhou. Preference-grounded token-level guidance for language model fine-tuning, 2023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023a.

Zhewei Yao, Reza Yazdani Aminabadi, Olatunji Ruwase, Samyam Rajbhandari, Xiaoxia Wu, Ammar Ahmad Awan, Jeff Rasley, Minjia Zhang, Conglong Li, Connor Holmes, Zhongzhu Zhou, Michael Wyatt, Molly Smith, Lev Kurilenko, Heyang Qin, Masahiro Tanaka, Shuai Che, Shuaiwen Leon Song, and Yuxiong He. Deepspeed-chat: Easy, fast and affordable rlhf training of chatgpt-like models at all scales, 2023b.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068, 2022. URL `https://api.semanticscholar.org/CorpusID:248496292`.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2020.

Appendix

A   Training Details

For all experiments, a weight decay of 0.1 and cosine learning rate scheduler is used. We initialize the generator model from the pre-trained checkpoint of OPT-1.3b. We use a $lr = 6e^{-5}$ to train on AMPS for one epoch using batch size of 152. We further fine-tune this model on MATH for ten epochs using the same batch size.

For the PRM-Avg result, RLHF using prompts from both MATH and GSM8K was performed using batch size 144 and $lr = 1e^{-4}$ for both actor and critic models. Similar hyperparameters were used for PRM-Prod with a batch size of 126 and $lr = 1e^{-4}$ for both actor and critic models. The PRM-Max result was achieved using batch size of 160 and $lr = 1e^{-4}$ for the actor and $lr = 5e^{-5}$ for the critic. Lastly PRM-Min used batch size 144, $lr = 1e^{-4}$ for the actor, and $lr = 5e^{-5}$ for the critic.

During RLHF, we found the coefficient for KL-divergence to have a nontrivial role in model stability. After some experimentation we found that kl coeff $= 0.2$ provided optimal results for our setup. Further, we clip our rewards at 0.7, and use 0.2 as our clip range when calculating the loss for both the actor and critic model. Following Ouyang et al. [2022], we set $\lambda = 0.95$ and $\gamma = 1.0$ in the generalized advantage estimate.