Training LM's to Solve Complex Math Problems

CSCI 699 Mini-Project

Qilin Ye

May 9, 2023

Recently, the field of language models has seen a major surge in research popularities, accompanied with rapid expansion of model sizes and the emergence of various novel training techniques. State-of-the-art models are able to achieve impressive performance on diverse tasks, they perform poorly on sophisticated mathematical problems, for example competition-level problems or math at or beyond college level. Even the most recent GPT-4 only manages to achieve median standing in AMC 12 and lower quartile in AMC 10 [2]. The complicated process involved in correctly solving a nontrivial mathematical question makes it challenging for models to achieve high accuracy. The model needs to correctly parse and understand the question, relate the problem to appropriate facts and theorems, generate a sequence of interconnected intermediate steps, and finally, output a simple comprehensible output or the entire process for proof-related questions.

We summarize current results and also propose a few potential ideas that help to increase a models accuracy in answering complex mathematical questions. In particular we focus on transferring a math question into a program synthesis problem as well as training the model to

understand and make use of theorems and other common tricks in competition-level problem solving.

Firstly, transferring a math question into a program synthesis problem and training the model to understand and make use of theorems and other common tricks in problem-solving could significantly improve performance. Drori et al [1] have shown that program synthesis, combined with few-shot learning, can achieve human-level performance on introductory college-level math courses, surpassing the performance of GPT-3 and 4 by a large margin. When given a question "compute X," they prompt the synthesizer to "write a program to compute X" instead, after which they execute the code using well-known libraries and obtain the result.

Complicated math problems often consist of multiple cases. When presented with one such question, the model can subdivide the problem into separate phases before feeding each case into a synthesizer. Theoretically, one would expect the model to display increased performance when the tasks are broken down into smaller, simpler sub-tasks. For example, when asked to compute certain properties of a random variable, it is good practice to discuss the discrete and the continuous case separately. As for another example, consider the minimization of an objective f over a closed and bounded domain D, a classic question in multivariable calculus. We can subdivide the problem into minimizing f over the interior and boundary of f, respectively, where we find the former by setting gradients to zero and the latter via Lagrange multiplier.

When it comes to theorem proving, Polu and Sutskever [3] proposed an automated prover and proof assistant, GPT-f, based on the Metamath formal system. Other formal systems and provers are also available. This paper [4] provides a detailed lists of current efforts in theorem proving models.

Unfortunately, when run on various benchmarks consisting of problems from contests ranging from AMC to IMO, these provers still have major rooms for improvement. Nevertheless, they may

be used to assist in solving simpler proof-based problems, especially algebraic ones, which, with little concern of logical loopholes which tend to occur in more involved proofs, are easier to verify, and likely to only rely on various algebraic identities, which a model can pick up quickly.

In addition, when tackling contest problems, we may train our models on commonly used tricks such as *stars-n-bars* in combinatorics, DP in recursion, aforementioned various algebraic identities in (elementary) algebra, and various inequalities and bounds in probability theory. Apparently, the question of "exactly how" needs more research, but if models are to perform better, I might venture to say that they need to capture these essential shortcuts, just like humans do.

To sum up, while SoTA language models achieve impressive performance on diverse tasks, there is still a long way to go for mathematical problem solving. Above are some proposed potential strategies to increase their accuracy. Certainly, further research is needed to fully determine whether these strategies are viable, and if yes, how to implement them. Once these models are mature enough, they may become helpful assistance to both education and research.

References

[1] Iddo Drori et al. "A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level." In: PNAS 119.32 (2022), e2123433119. DOI: 10.1073/pnas.2123433119. URL: https://www.pnas.org/doi/abs/10.1073/pnas.2123433119.

[2] OpenAI. GPT-4 Technical Report. 2023. arXiv: 2303.08774 [cs.CL].

[3] Stanislas Polu and Ilya Sutskever. Generative Language Modeling for Automated Theorem Proving. 2020. arXiv: 2009.03393 [cs.LG].

[4] Yuhuai Wu et al. INT: An Inequality Benchmark for Evaluating Generalization in Theorem Proving. 2021. arXiv: 2007.02924 [cs.AI].