Towards Optimizing with Large Language Model

Pei-Fu Guo[1,*], Ying-Hsuan Chen[1,*], Yun-Da Tsai[1] and Shou-De Lin[1]

[1]Department of Computer Science and Information Engineering, National Taiwan University

Abstract

In this study, we evaluate the optimization capabilities of Large Language Models (LLMs) across diverse mathematical and combinatorial optimization tasks, where each task is described in natural language. These tasks require LLM to iteratively generate and evaluate solutions through interactive prompting, where each optimization step involves generating new solutions based on past results and then pass to subsequent iterations. We demonstrate that LLMs can perform various optimization algorithms and act as effective black-box optimizers, capable of intelligently optimizing unknown functions. We also introduce three simple yet informative metrics to evaluate optimization performance, applicable across diverse tasks and less sensitive to test sample variations. Our findings reveal that LLMs excel at optimizing small-scale problems with limited data and their performance is significantly affected by the dimension of problem and values, highlighting the need for further research in LLM optimization.

Keywords

LLM Reasoning, LLM Optimization

1. Introduction

Large Language Models have demonstrated exceptional capabilities in reasoning across a variety of natural language-based tasks [1]. However, their potential extends beyond multiple-choice questions or single-question answering. This work explores LLMs' effectiveness in optimization across diverse tasks and problem dimensions. Optimization involves iteratively generating and evaluating solutions to improve a given objective function. Our research assesses LLM performance in interactive optimization, where each step generates new solutions based on previous ones and their values.

We conduct our study with four different types of optimization algorithms: Gradient Descent, Hill Climbing, Grid Search, and Black Box Optimization. To provide a comprehensive evaluation of LLM performance, we introduce three distinct metrics. These metrics provide a multifaceted view of task performance and are applicable across a broad spectrum of optimization tasks, reducing sensitivity to sample variations.

Our findings suggest that LLMs show impressive optimization capabilities, especially in small-scale problems. However, their performance is notably affected by factors like sample size and value range. These observations underscore the need for further research within the domain of optimization tasks tailored for LLMs. It's important to note that our work does not aim to outperform state-of-the-art optimization algorithms for either mathematical optimization or combinatorial optimization problems. Instead, our goal is to showcase the potential of LLM in these optimization domains and find out limitations in these settings.

Our contributions are summarized as follows:

- Exploring the potential of LLMs in mathematical and combinatorial optimization scenarios.
- Introduce three novel metrics for assessing LLM performance in optimization tasks.
- Delve into factors that influence LLM performance using our metrics, with a particular emphasis on the impact of problem dimension and task type.

The remainder of this paper is structured as follows. In Section 2, we present preliminary works on LLMs for addressing optimization challenges. In Section 3, we defined 4 optimization algorithms in the case studies. In Section 4, we demonstrate that LLMs with iterative prompting strategy function as optimizers. In Section 5, we present three metrics that we have designed to assess the overall performance of LLMs in undertaking optimization tasks. Section 6, details our experimental results, showcasing the effectiveness of using LLMs as optimizers. In Section 7, we consolidated noteworthy observations and points of discussion from the experiments. Finally, Section 8 summarizes and concludes the paper.

2. Related Works

In various optimization scenarios, the utilization of Large Language Models (LLMs) has become indispensable for the development of optimization algorithms or agent systems capable of handling complex and informative text-based feedback. In this section, we summarize three

KiL'24: Workshop on Knowledge-infused Learning co-located with 30th ACM KDD Conference, August 26, 2024, Barcelona, Spain

*Both authors contributed equally.

r12922217@csie.ntu.edu.tw (P. Guo); r12922044@csie.ntu.edu.tw (Y. Chen); f08946007@csie.ntu.edu.tw (Y. Tsai); sdlin@csie.ntu.edu.tw (S. Lin)

© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

significant related works that leverage LLMs to tackle optimization and reinforcement learning challenges. These works showcase the adaptability and effectiveness of LLMs in addressing optimization and learning challenges across various domains.

Optimization by PROmpting (OPRO) [2] OPRO harnesses LLMs as versatile optimizers by describing optimization tasks in natural language prompts. It iteratively generates and evaluates solutions from these prompts, demonstrating superior performance on tasks like linear regression and traveling salesman problems. OPRO outperforms human-designed prompts by up to 50% on challenging tasks.

Reflexion [3] Reflexion introduces a novel framework for training language agents that rely on linguistic feedback rather than traditional reinforcement learning. This framework delivers outstanding results, boasting a remarkable 91% pass@1 accuracy on coding tasks—an exceptional 11% improvement over previous state-of-the-art models. Reflexion's success underscores the potential of linguistic feedback as a powerful training mechanism.

EvoPrompt [4] EvoPrompt automates prompt optimization by connecting LLMs with evolutionary algorithms. This automated process surpasses human-designed prompts by up to 25% and outperforms existing automatic prompt generation methods by an impressive 14%. EvoPrompt's success highlights the relationship between Large Language Models and traditional algorithms, showcasing the potential for enhanced problem-solving capabilities through this synergistic fusion.

3. Problem setting

We design four optimization tasks that require the model to algorithmically search for the optimal value of parameters. These tasks encompass Gradient-Descent, Hill-Climbing, Grid-Search, and Black-Box Optimization, each representing unique optimization domains: gradient-based, meta-heuristics, decision-theoretic, and Bayesian. In terms of parameter types, Grid-Search and Hill-Climbing involve discrete search spaces, while Gradient-Descent and Black-Box Optimization tackle continuous search spaces. Following is detailed information on each optimization task.

Gradient-Descent assesses the model's proficiency in advanced calculations and its grasp of the principles of gradient descent. We instruct LLMs to undertake a conventional gradient descent optimization process based on the loss function they have defined. LLMs need to compute the gradient and update the parameters using the gradient information and the learning rate given.

Hill-Climbing evaluate the LLM's capability to adhere to custom predefined rules they have not seen before. LLMs start with an initial solution and iteratively explore nearby solutions by making small incremental changes. In our task, neighboring solutions are generated by selecting a specific element within the solution and either increasing or decreasing it by one each time. Subsequently, the neighbor solution with the minimum loss is chosen as the new solution and passed to the next iteration.

Grid-Search assesses the LLM's ability to conduct exhaustive searches and locate optimal solutions within a predefined search space. LLMs are tasked with generating all grid points and systematically searching for the point that results in the lowest loss according to the given loss function.

Black-Box Optimization evaluates the LLM's ability to make informed decisions and optimize in an abstract problem-solving context. We treat the LLMs as black boxes that try to fit an unknown loss function. We provide the LLM with a limited set of solutions, each paired with its respective true loss value. The LLM's objective is to discover new solutions that have lower losses than the existing solutions in each iteration by themselves.

4. Methodologies

In this section, we show how LLMs, guided by iterative prompting, can effectively function as optimizers, akin to various optimization algorithms. To systematically navigate the search space, we introduce an iterative prompting framework that enables LLMs to incrementally achieve better solutions within the search space through iterative processes.

We applied Chain of Thoughts and iterative prompting as our prompting method. LLM will accomplish each step with reasoning thoughts as intermediate outputs. In each of these tasks (optimization algorithm), LLMs are initially required to formulate the loss function based on given samples. Then each optimization iteration is composed of two steps: (1) Generates new solution based on algorithm instructions and past search results (2) Calculate loss of new solution and add the results to the prompt of the next iteration. We keep repeating the two steps until the stop criteria are met. Figure 1 shows an overview of how LLM performs optimization in interactive settings.

To create an interactive environment, we utilize the chat mode of GPTs, where the entire conversation history serves as the prompt. This allows LLMs to retain memory of past search results and reasoning paths. New instructions are appended to ongoing conversation records with each iteration. If the dialogue surpasses the token limit, earlier portions are removed.

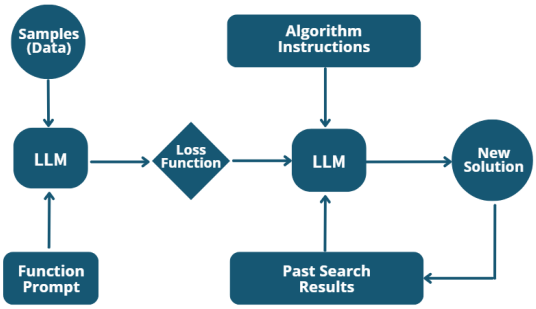

Figure 1: Overview of our prompting strategy. (1) LLMs formulate the loss function based on given samples. (2) Given algorithm instructions and past results, LLM generates a new solution. (3) Calculate the loss of the new solution and add the solution-score pairs to the prompt of the next iteration. (4) Repeat the second and third steps until stop criteria are met.

5. Evaluation

We devised three novel metrics for the comprehensive evaluation of LLM capabilities. In this section, we will explain the design and objective of each metric. These metrics offer versatility in assessing LLM performance across diverse tasks, making concurrent evaluation easier. Their reliance on ratio measures, rather than differences, makes them less sensitive to sample variations.

5.1. Goal Metric

Goal metric evaluates how effectively LLMs perform optimization. It provides a quantitative measure of the degree to which the LLM contributes to minimizing the loss function values. In other words, ensuring that the ultimate solution loss is lower than the initial solution. We define the *goal metric* of a test sample j as :

$$G_j = \frac{1}{N}\sum_{i=1}^{N}\frac{loss_{LLM,init} - loss_{LLM,i}}{loss_{LLM,init}} \tag{1}$$

where $loss_{LLM,init}$ is the initial solution loss of sample j, $loss_{LLM,i}$ is the LLM output loss of trial i, and N is the number of trials per sample. The higher the metric value, the greater the progress in optimization. The goal metric plays a crucial role in our evaluation framework, particularly in scenarios where ground truth is absent, such as the Black-Box optimization scenarios.

5.2. Policy Metric

Policy metric assesses the degree of alignment between the final model output and the ground truth. Beyond self-improvement, which is measured by *goal metric*, it is also crucial to appraise the LLMs' capability to operate in a manner consistent with our truth model algorithm. This metric serves as an indicator of the LLM's adeptness in adhering to task-specific instructions. We define the *policy metric* of a test sample j as :

$$P_j = \frac{1}{N}\sum_{i=1}^{N}\frac{loss_{LLM,i} - loss_{truth}}{loss_{truth}} \tag{2}$$

where $loss_{LLM,i}$ is the LLM output loss of trial i, $loss_{truth}$ is the ground truth of sample j and N is the number of trials. Since the policy metric measures the disparity between the ground truth and the LLM's output, a lower policy metric value indicates a more effective alignment of the LLM's actions with the prescribed guidelines. When the value is negative, it means that LLM's performance surpasses the ground truth.

5.3. Uncertainty Metric

Uncertainty metric quantifies the variability in the LLM's solutions under identical conditions. Stability is a crucial characteristic in optimization tasks. We hope that the LLMs produce identical results in every trial involving the same sample, even under conditions with temperatures greater than zero. We define the *uncertainty metric* of a test sample j as :

$$U_j = \frac{1}{N}\sum_{i=1}^{N}(loss_{LLM,i} - \overline{loss_{LLM}})^2 \tag{3}$$

where $loss_{LLM,i}$ is the LLM output of the i-th trial, $\overline{loss_{LLM}}$ is the mean of the trial outputs and N is the number of trials. A stable LLM can be more trusted for tasks that demand consistent and reproducible results. In our case, if the language model truly understands the context of problems, the final optimal output should be identical in every trial of the same sample.

6. Experiments

This section provides details of our experimental configurations and highlights the outcomes of experiments. Subsection 6.1 outlines the process of generating synthetic datasets for all optimization tasks, while subsection 6.2 elucidates the detailed settings of our experiment. Lastly, subsection 6.3 offers a concise summary of the outcomes derived from our experiment.

6.1. Dataset

In the experiment, we create five datasets with d values chosen from the set $\{3, 6, 12, 24, 48\}$ and generate instances belonging to $[0, 10]^d$ in each dataset to examine

sensitivity to the number of parameters, representing the dimension of the optimization problem. For instance, $d = 3$ indicates that there are 3 variables in the loss function and the dimension of this optimization problem is 3. We then apply each instance to a loss function and find the true solution for each parameter search task. These authenticated solutions, coupled with their associated losses, not only serve as the ground truth for the tasks but also act as a pivotal benchmark against which the solutions derived by LLMs are systematically evaluated and compared in the ensuing analysis.

6.2. Detailed Settings

In our experiment, We set the LLM temperature to 0.8 and the reset as default. We performed 5 repetitions of the test for each instance in the dataset, with the LLM conducting 10 iterations of parameter search in each repetition. We excluded excessively biased results to prevent our metrics from being skewed by a minority of poorly performing test outcomes. All experiments employ the GPT-turbo-3.5 '0613' version as the Language Model.

6.3. Main Results

We summarize the outcomes of our experiment and subsequently examine the common trends observed across all experiments. In every plot, the x-axis displays the dimension of the optimization problem. In the case of the goal metric and policy metric plots, the y-axis illustrates the average metric value for the respective tasks, while the shaded area in a lighter color delineates the confidence interval of the metric, denoted as $[value - std, value + std]$. As for the uncertainty metric plot, the y-axis showcases the uncertainty metric value, which corresponds to the standard deviation of the LLM final solution loss. It is worth noting that the Goal Metric graph excludes the non-iterative Grid-Search task due to its non-iterative nature, while the Policy Metric graph omits the Black-Box task due to unattainable ground truth.

LLMs show strong optimization capabilities in small-scale problems. Our experiments test the comprehensive optimization capabilities of LLMs. Observing figure 2, GPT-turbo-3.5 showcases considerable optimization capabilities across various scenarios. Impressively, in the Gradient-Descent task, GPT-turbo-3.5 even surpasses the ground truth, particularly in the case of the sample dimension equal to six. It's also surprising that the model achieves respectable results in the Grid-Search task, considering it must compute a vast number of grid points, which increase exponentially as the dimension of the problem expands. The model faces challenges in the Hill-Climbing task, evident from a policy metric significantly exceeding zero. This suggests that meta-heuristics may pose greater difficulty for LLMs compared to other tasks.

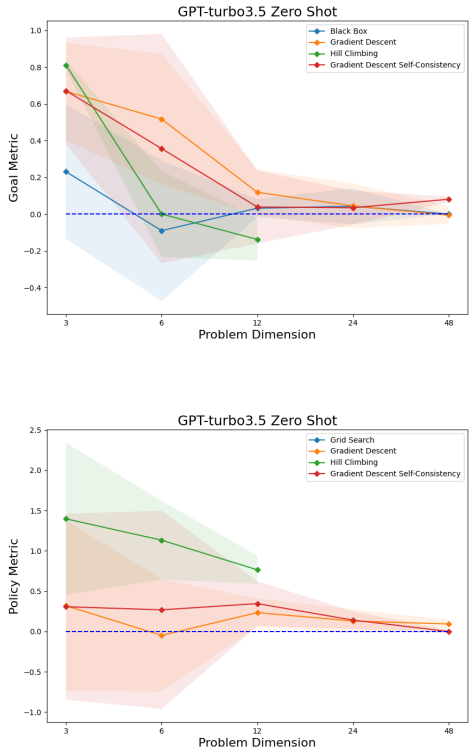

Figure 2: Goal Metric and Policy Metric hover from positive to near zero, signifying substantial optimization capability and alignment between LLM's output and ground truth.

LLMs show potential as Black-Box Optimizers. Favorable performance in Black-Box experiments suggests the use of LLM as an optimizer without giving any algorithm instructions. From figure 3, we can see that GPT-turbo-3.5 performs notably when the dimension of the problem is three, whereas GPT-4 excels when the dimensions are three and six. Interestingly, as the dimension increases, the performance of both models gradually diminishes. Eventually, GPT-4 edged out GPT-turbo-3.5 by a slight margin in optimization and stability.

LLMs exhibit strong performance in Gradient-Descent. Gradient-Descent experiment tests the model's proficiency in advanced calculations and grasp of mathematics principles. Figure 4 underscores this by revealing a policy metric that consistently hovers near zero, signifying a remarkable alignment between the LLM's output and the ground truth. Despite a decline in the goal metric as the sample size increases, the consistently low and

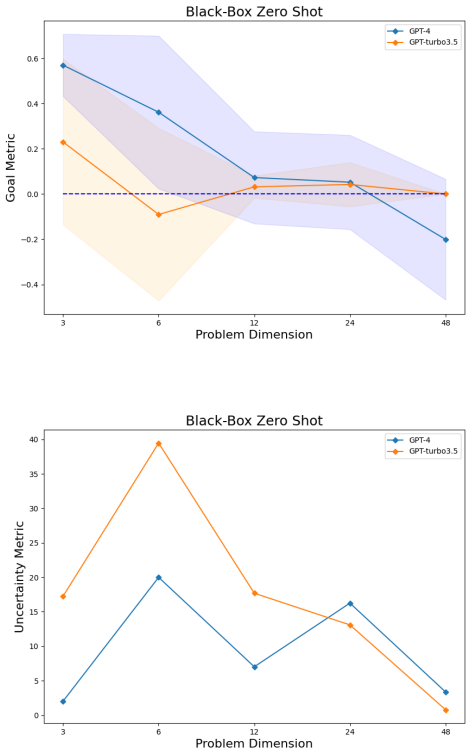

Figure 3: Goal Metric reflects the performance of LLMs as Black-Box optimizer, showing strong performance with instances of smaller dimensions.

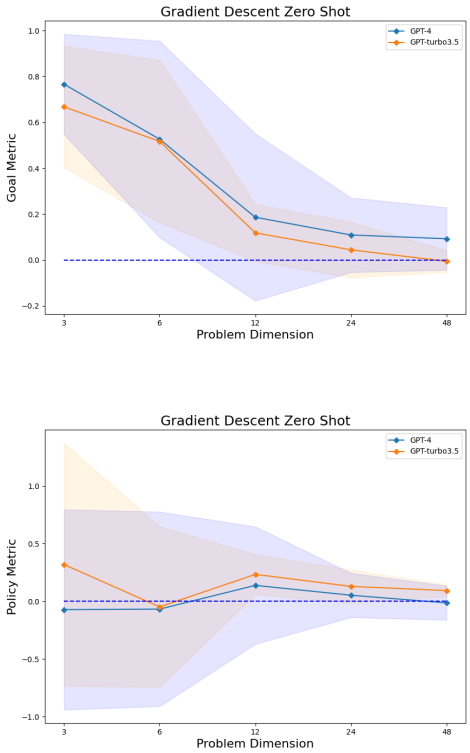

Figure 4: Low values in the Policy Metric and high positive values in the Goal Metric indicate the robust performance of the LLM in the gradient descent task.

stable value of the policy metric underscores the fact that GPT's performance in the gradient-descent task is nearly on par with the truth model.

7. Analysis and Discussion

In this section, we consolidate several crucial insights derived from our experimental results and subject them to analysis.

Pretrained Knowledge dominates the optimization capability of LLM. Among all optimization tasks performed by LLMs, Gradient Descent emerges as the leading performer, while Hill-Climbing poses greater challenges. The main difference between the two tasks is that Hill-Climbing is a heuristic algorithm with more user-specific parameters, whereas gradient descent is an optimization algorithm that relies more on mathematical principles. This suggests that LLM optimization capabilities primarily stem from pretrained knowledge stored within the model parameters, rather than from context knowledge provided by users. Our findings align with previous research [5, 6, 7] showing that language models often prioritize their prior knowledge over new context. Achieving balanced attention to both prior and context knowledge is essential for further research to improve the optimization capability of language models.

LLMs are potential hybrid optimizers. The predominantly positive goal metric values across most tasks and datasets indicate LLMs' capability for optimization. This highlights their versatile capacity to optimize across different problem spaces, potentially allowing for the switching between optimization methods within a single task. Such switching can help LLMs better explore the solution space and escape local optima where they might get stuck. This is a significant advantage of LLMs in optimization, as they can easily change methods through a simple natural language prompt during iterations. Furthermore, LLMs can act as agents (world models) that use different algorithms as tools (actions), switching methods by evaluating the optimization path from past to present (state). This adaptability underscores the potential of LLMs to enhance optimization processes through

dynamic method selection and strategic problem-solving.

LLMs possess richer solution space in small-scale problems. In our experiments, we observed high uncertainty metric values and significant variations in policy and goal metrics when samples had smaller dimensions. Interestingly, LLMs tend to perform more effectively with smaller dimension instances, suggesting a correlation between higher uncertainty and better performance. This consistent pattern across various tasks and models indicates that LLMs have a richer solution space when tackling small-scale problems. The expanded solution space leads to higher uncertainty, providing LLMs with a broader range of solutions to explore. This highlights the importance of dimension reduction in data preprocessing for effective optimization by LLMs. Figure 2 and 5 both highlight the pattern of uncertainty, where the uncertainty initially rises and then gradually decreases.

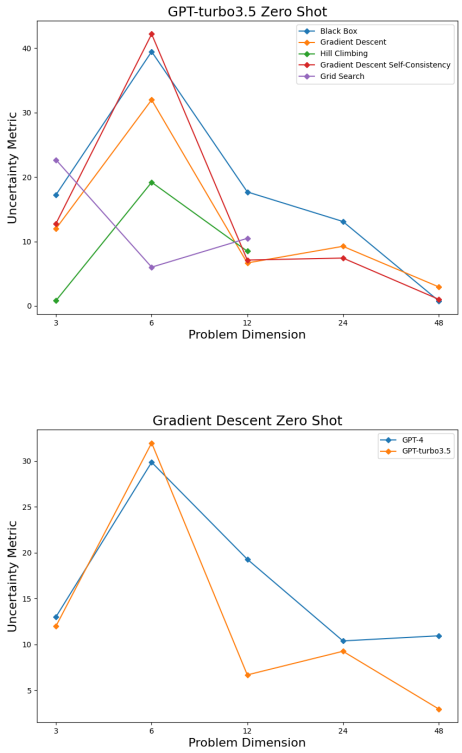

Figure 5: An initial rise followed by a decline in the Uncertainty Metric with instance dimension growth suggests LLMs may have a richer sample space for small-scale problems, consistent across tasks and models.

LLMs are sensitive to numerical values. It's worth considering that the aforementioned results may be influenced by the inherent randomness in the generation of test samples. Previous research has indicated that LLMs may demonstrate preferences for particular numbers, words, and symbols [8], which can introduce a level of bias in their responses. Given the high sensitivity of LLMs to the input prompt, the initial starting points and data provided can exert a significant influence on their outputs. In essence, the impact of instruction description and data initialization should be carefully considered when interpreting the results of LLM-based experiments to ensure a more accurate assessment of their performance.

Self-consistency prompting improves stability. In the Gradient-Descent task, we employ self-consistency technique [9], where we conduct five repetitions for each iteration and select the solution that emerges most frequently. From Figure 6, we can see that GPT-4 performances increase largely, and the confidence interval for both the policy-metric and goal-metric narrows, indicating improved stability and reliability. Nonetheless, this approach does not yield favorable outcomes when applied to GPT-turbo-3.5. This suggests the need for further investigation within the realm of variance reduction.

8. Conclusion and Future Directions

In this paper, we present our in-depth examination of assessing Large Language Models within the realm of optimization, where LLM progressively generates new solutions to optimize an objective function. We investigate LLMs' performance across four optimization tasks that necessitate their comprehension of algorithmic instructions and their ability to generate new solutions based on previous solutions and their corresponding values.

Our evaluation shows that LLMs showcase optimization prowess across diverse domains. Among the four tasks we examined, LLMs exhibit their greatest strengths in the Gradient-Descent task, displaying remarkable proficiency in this area. However, they encounter more pronounced difficulties in the meta-heuristics task, where they must adhere to predefined rules that they have not encountered previously. Furthermore, LLMs demonstrate impressive skills in the grid search task, showcasing their ability to conduct exhaustive searches effectively. In the Black-Box task, LLMs excel, particularly when dealing with limited sample sizes, suggesting inherent optimization abilities within them.

We also consolidate several crucial insights derived from our experimental results and subject them to analysis. We find that pretrained knowledge dominates the optimization capability of LLMs, while they also possess a richer solution space in small-scale problems. Furthermore, we elaborate on the potential of LLMs as hybrid optimizers. These insights and analyses unveil a host of

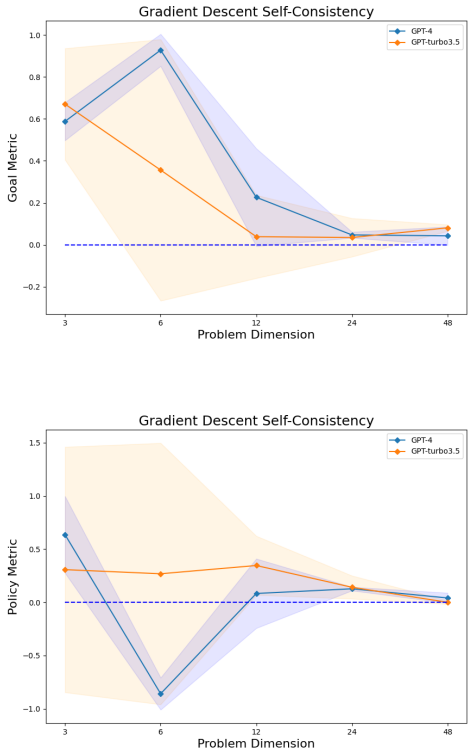

Figure 6: The confidence intervals for both the policy and goal metrics of GPT-4 narrow, indicating improved stability. A negative policy metric with a high goal metric signifies significant outperformance of the ground truth model with six-dimensional instances.

unresolved questions that warrant further research.

References

[1] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, D. Zhou, Chain of thought prompting elicits reasoning in large language models, in: arXiv preprint arXiv:2201.11903, 2022.

[2] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, X. Chen, Large language models as optimizers, in: arXiv preprint arXiv:2309.03409, 2023.

[3] N. Shinn, F. Cassano, B. Labash, A. Gopinath, K. Narasimhan, S. Yao, Reflexion: Language agents with verbal reinforcement learning, in: arXiv:2303.11366, 2023.

[4] Q. Guo, R. Wang, J. Guo, B. Li, K. Song, X. Tan, G. Liu, J. Bian, Y. Yang, Connecting large language models with evolutionary algorithms yields powerful prompt optimizers, in: arXiv:2309.08532, 2023.

[5] H.-T. Chen, M. Zhang, E. Choi, Rich knowledge sources bring complex knowledge conflicts: Recalibrating models to reflect conflicting evidence, in: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 2292–2307, Abu Dhabi, United Arab Emirates, Association for Computational Linguistics., 2022.

[6] A. Pagnoni, V. Balachandran, Y. Tsvetkov, Understanding factuality in abstractive summarization with frank: A benchmark for factuality metrics, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4812–4829, Online. Association for Computational Linguistics, 2021.

[7] W. Zhou, S. Zhang, H. Poon, M. Chen, Context-faithful prompting for large language models, in: ArXiv, abs/2303.11315, 2023.

[8] A. Renda, A. Hopkins, M. Carbin, Can llms generate random numbers? evaluatingllm sampling in controlled domains, in: ICML 2023 Workshop: Sampling and Optimization in Discrete Space, 2023.

[9] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, D. Zhou, Self-consistency improves chain of thought reasoning in language models, in: arXiv preprint arXiv:2203.11171, 2022.

A. Prompt Templates

User Prompt:
Q :
Given the data points (y1, y2, ...) = {data}, what is the MSE loss function with respect to the ŷs for a hypothetical set of predicted ŷs values?
A :
The MSE loss function for the given data points (y1, y2, ...) = {data} with respect to ŷs is:...

Figure 7: Example prompt for getting objective function.

User Prompt:
Q :
Please minimize the loss function using gradient descent with learning rate of 0.1 at point (ŷ1, ŷ2, ŷ3, ...) = {point}. What is the point we eventually end up after one update? Your answer includes two parts an explanation with calculation and a short answer of result.
A :
Explanation : Lets think step by step ...
Short Answer: After calculation, the next update point is ($\hat{y}1_{new}$, $\hat{y}2_{new}$, $\hat{y}3_{new}$, ...) = ...

Figure 8: Example prompt for Gradient-Descent.

User Prompt :
Q :
I want to do grid search on the ŷs and the range of them are the integers of {low_bound} to {high_bound}. Generate all possible combinations of ŷs values from the specify range.
What are the combinations? Your answer includes two parts an explanation with calculation and a list containing all the combinations.
A :
Explanation : Lets think step by step ...
List : [write all the combinations here]

Figure 9: Example prompt for Grid-Search (Create Grid Points)

User Prompt :
Q :
For every combinations of ŷs, calculate its MSE loss. Which combination has the smallest MSE loss? Your answer includes two parts an explanation with calculation and a list containing the combination with the smallest MSE loss.
A :
Explanation : Lets think step by step
List : [write the combination with smallest MSE loss]

Figure 10: Example prompt for Grid-Search (Select)

User Prompt :
Q :
I want to minimize the loss function using hill climbing. Generate neighboring solutions by either adding 1 or minus 1 to a specific element in the current solution. The current solution is solution. Your answer includes two parts an explanation with calculation and a list containing all neighbor solutions(eg. [($\hat{y}1$, $\hat{y}2$,....), ($\hat{y}1$, $\hat{y}2$,....), ...]).
A :
Explanation : Let's think step by step ...
List : [write neighbor solutions here]

Figure 11: Example prompt for Hill-Climbing Prompt (Generate Neighbors)

User Prompt :
Q :
For every neighbor solution, calculate its MSE loss. Which neighbor has the smallest MSE loss? Your answer includes two parts an explanation with calculation and a list containing the smallest MSE loss neighbor solution(eg. [($\hat{y}1$, $\hat{y}2$,....)]).
A :
Explanation : Lets think step by step ...
List : [write best neighbor solution here]

Figure 12: Example prompt for Hill-Climbing Prompt (Select)

User Prompt :
Q :
You want to minimize an unknown MSE loss function by guessing the values of the $\hat{y}$s. When you guess, you should take consider of the past guessing result so that your new guess will have smaller loss than the past results. Pass guessing result are {pass_result}. Base on the previous guesses, what is your next guess?
A :
($\hat{y}1$, $\hat{y}2$,....) = [your answer]

Figure 13: Example prompt for Black-Box

User Prompt :
***Define Loss function**
User :
Given the data points (y1, y2, y3) = (2,6,0), what is the MSE loss function with respect to the $\hat{y}$s for a hypothetical set of predicted $\hat{y}$s values?
LLM :
The MSE loss function for the given data points (y1, y2, y3) = (2,6,0) with respect to $\hat{y}$s is: $\frac{1}{3}[(\hat{y}_1 - 2)^2 + (\hat{y}_2 - 6)^2 + (\hat{y}_3 - 0)^2]$
***First Iteration**
User :
Please minimize the loss function using gradient descent with learning rate of 0.1 at point ($\hat{y}1$, $\hat{y}2$, $\hat{y}3$) = (2,3,4). What is the point we eventually end up after one update? Your answer includes two parts an explanation with calculation and a short answer of result.
LLM :
Explanation : To minimize the Mean Squared Error (MSE) loss function using gradient descent ...
Short Answer: After calculation, the next update point is ($\hat{y}1_{new}$, $\hat{y}2_{new}$, $\hat{y}3_{new}$) = (1.92, 3.05, 3.94)
***Second Iteration**
User :
Please minimize the loss function using gradient descent with learning rate of 0.1 at point ($\hat{y}1$, $\hat{y}2$, $\hat{y}3$) = ((1.92, 3.05, 3.94). What is the point we eventually end up after one update? Your answer includes two parts an explanation with calculation and a short answer of result.
LLM :
Explanation : To minimize the Mean Squared Error (MSE) loss function using gradient descent ...
Short Answer: After calculation, the next update point is ($\hat{y}1_{new}$, $\hat{y}2_{new}$, $\hat{y}3_{new}$) = (1.81, 3.12, 3.41)...

Figure 14: Iterative prompt for Gradient-Descent

User Prompt :
***Define Loss function**
User :
Given the data points (y1, y2, y3) = (2,6,0), what is the MSE loss function with respect to the ŷs for a hypothetical set of predicted ŷs values?
LLM :
The MSE loss function for the given data points (y1, y2, y3) = (2,6,0) with respect to ŷs is: $\frac{1}{3}[(\hat{y}_1 - 2)^2 + (\hat{y}_2 - 6)^2 + (\hat{y}_3 - 0)^2]$
***Create Grid Points**
User :
I want to do grid search on the ŷs and the range of them are the integers of 2 to 3. Generate all possible combinations of ŷs values from the specify range. What are the combinations? Your answer includes two parts an explanation with calculation and a list containing all the combinations.
LLM :
Explanation : To generate all the grid points, you should ...
List : [(2, 2, 2), (2, 2, 3), (2, 3, 2), (2, 3, 3), (3, 2, 2), (3, 2, 3), (3, 3, 2), (3, 3, 3)]
***Select Best**
User :
For every combinations of ŷs, calculate its MSE loss. Which combination has the smallest MSE loss? Your answer includes two parts an explanation with calculation and a list containing the combination with the smallest MSE loss.
LLM :
Explanation : To select the grid points with smallest MSE value ...
List : [(2, 3, 3)]...

Figure 15: Iterative prompt for Grid-Search

User Prompt :
***Define Loss function**
User :
Given the data points (y1, y2, y3) = (2,6,0), what is the MSE loss function with respect to the ŷs for a hypothetical set of predicted ŷs values?
LLM :
The MSE loss function for the given data points (y1, y2, y3) = (2,6,0) with respect to ŷs is: $\frac{1}{3}[(\hat{y}_1 - 2)^2 + (\hat{y}_2 - 6)^2 + (\hat{y}_3 - 0)^2]$
***Generate Neighbors**
User :
I want to minimize the loss function using hill climbing. Generate neighboring solutions by either adding 1 or minus 1 to a specific element in the current solution. The current solution is (10,10,10). Your answer includes two parts an explanation with calculation and a list containing all neighbor solutions(eg. [(ŷ1, ŷ2,....), (ŷ1, ŷ2,....), ...]).
LLM :
Explanation : To generate all the neighbors ...
List : [(11, 10, 10), (10, 11, 10), (10, 10, 11), (9, 10, 10), (10, 9, 10), (10, 10, 9)].
***Select Best**
User :
For every neighbor solution, calculate its MSE loss. Which neighbor has the smallest MSE loss? Your answer includes two parts an explanation with calculation and a list containing the smallest MSE loss neighbor solution(eg. [(ŷ1, ŷ2,....)]).
LLM :
Explanation : To select the neighbor solution with smallest MSE value ...
List : [(9, 10, 10)]

Figure 16: Iterative prompt for Hill-Climbing

User Prompt :
***Define Loss function**
User :
Given the data points (y1, y2, y3) = (2,6,0), what is the MSE loss function with respect to the ŷs for a hypothetical set of predicted ŷs values?
LLM :
The MSE loss function for the given data points (y1, y2, y3) = (2,6,0) with respect to ŷs is: $\frac{1}{3}[(\hat{y}_1 - 2)^2 + (\hat{y}_2 - 6)^2 + (\hat{y}_3 - 0)^2]$
First Iteration
User :
You want to minimize an unknown MSE loss function by guessing the values of the ŷs. When you guess, you should take consider of the past guessing result so that your new guess will have smaller loss than the past results. Pass guessing result are $f(2, 4, 6) = 10,\ f(1, 5, 0) = 2$. Base on the previous guesses, what is your next guess?
LLM :
(ŷ1, ŷ2, ŷ3) = $[1, 3, 5]$
Second Iteration
User :
You want to minimize an unknown MSE loss function by guessing the values of the ŷs. When you guess, you should take consider of the past guessing result so that your new guess will have smaller loss than the past results. Pass guessing result are $f(2, 4, 6) = 10,\ f(1, 5, 0) = 2,\ f(1, 3, 5) = 5$. Base on the previous guesses, what is your next guess?
LLM :
(ŷ1, ŷ2, ŷ3) = $[1, 1, 1]$

Figure 17: Iterative prompt for Black-Box Optimization