RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

Hyperparameter Optimization of LLM Fine-Tuning

Bayesian and Partition-based approaches

N. Davouse, E-G. Talbi,
Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France

Summary

01
Introduction

Large Language Models

Summary

- State-of-the-art of Natural Language Processing (NLP) problems
- Architecture : Transformers block, mixed with classical layers (MLP, Conv)
- Huge size : Billions of parameters (1B to 405B for Llama 3)
- 2 phases of training : pre-training and **fine-tuning**

Self Attention

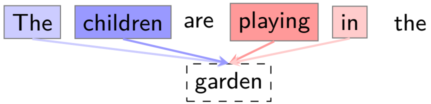


Figure: Self Attention mecanism illustration

Self attention is the key of LLM, used to compute the context of each token.

Fine-Tuning

Following a first phase of pre-training, Fine-tuning is used to correct behavior or add in-domain data to a model, with limited ressources.

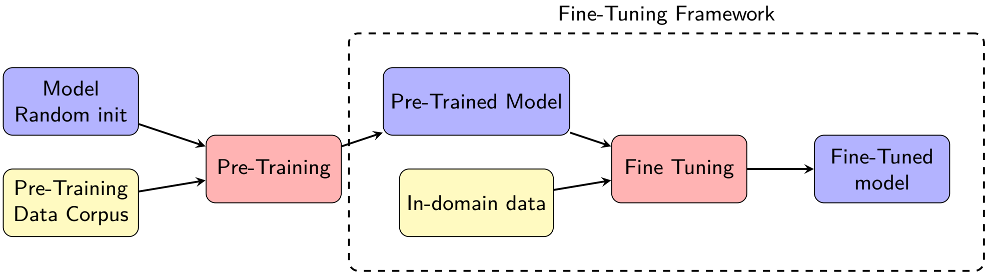


Figure: Pre-training and Fine-tuning generic workflow

Parameters Efficient Fine-Tuning (PEFT)

Set of methods aims to reduce the computation cost of fine-tuning. 2 main approaches : *Additive* and **reparametrization**.

Reparametrization

Use lower-cost proxy as trainable weights, and merge at the end.

Addivitive

Add part of the model, often linear layer, to train these. One con is to add inference to generation.

Quantization

To reduce further the cost of computing during the training, quantization can also be used. This can be combined with either of precedent approaches.

Low Rank Adaptation (LoRA)

Principle

Merging Fine-tuning layers with pre-trained ones can be written as $W = W_0 + \Delta W$, with W_0 the pre-trained weights and ΔW the fine-tuned ones. With LoRA, $W = W_0 + \frac{\alpha}{r} B.A$

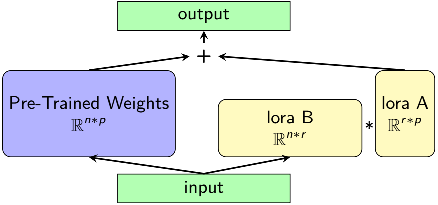


Figure: LoRA Decomposition

LoRA hyperparameters

- rank r : the common dimension between A and B.
- alpha α : apply a weighting between fine-tuning and pre-trained weights

Hyperparameter Optimization (HPO)

Objectives

- Better performance than manual tuning
- Ease popularization of the Fine Tuning

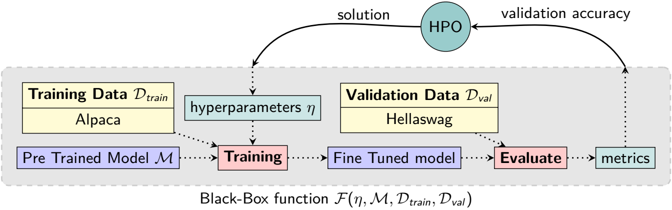


Figure: HPO workflow

Related Works

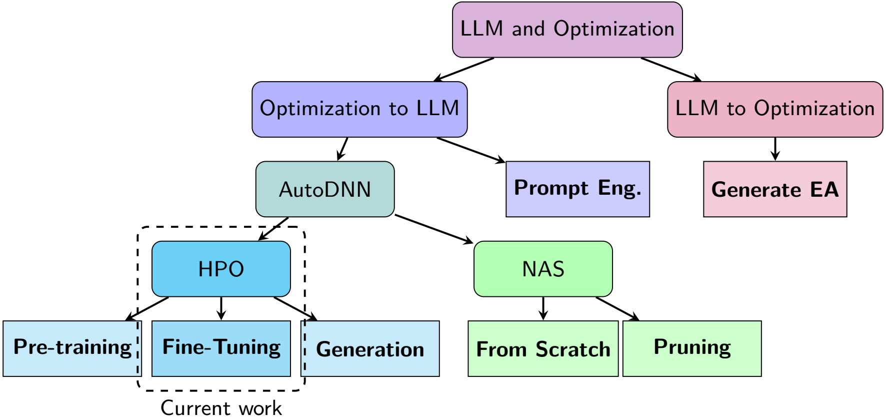


Figure: Summary of links between LLM and Optimization

02
Problem Definition

Problem Definition

Problem Formulation

The HPO problem can be defined as

$$\eta^* \in \arg\min_{\eta \in \mathcal{H}} \mathcal{F}(\eta) \qquad (1)$$

This function can be characterized as an **expensive, mixed-variable, noisy, blackbox** function.

3 phases of an optimization problem

- **Search Space** $\mathcal{H}$: all variables and how to handle them
- **Search Strategy** $\arg\min$: how to search for the minimum of the function
- **Performance Evaluation Strategy** $\mathcal{F}(.)$: how to evaluate a given solution

Search Space

Hyperparameters

Hyperparameters	Optimization range		Type	Conversion
	Lower Bound	Upper Bound		
Learning Rate	-10	-1	log.	$f(x) = 10^x$
LoRA Rank	1	64	int.	$f(x) = \text{round}(x)$
LoRA scale (α)	1	64	int.	$f(x) = \text{round}(x)$
LoRA Dropout	0	0.5	cont.	$f(x) = x$
Weight Decay	-3	-1	log.	$f(x) = 10^x$

Table: Summary of Hyperparameter Search Space

- Conversion and naming convention is taken from LitGPT framework.
- Variable conversion for handling mixed-variables with continuous algorithms
- No *A-priori* knowledge on hyperparameters importance

Search Strategy

Algorithms for LLM HPO are *Global Optimization* algorithms. Can be classified as :

- **Exploratory**(GS, Random Search, LHS) : sample the search space
  no exploitation, give a lower bound
- **Metaheuristics** (Genetic Algorithm, ILS, PSO) : bio-inspired heuristics
  evaluation greedy, cannot be used for expensive function
- **Surrogate-Model based Optimization** (Bayesian Optimization with Gaussian Process, TS) :
  Use a surrogate to enhance exploitation - innate sequential nature, strong exploitation
- **Partition-Based Optimization**(FDA, SOO, DiRect) : partition the search space
  massively parallel, slow convergence

Performance Evaluation Strategy

Evaluation context

In this part, there are many options, like the number of epochs (if not an hyperparameters), the precision of the model, the datasets of training or evaluation.

Objective function

2 ways to evaluate LLM Fine-Tuning :

- **Loss (validation/test)** : dataset and model dependant, difficult to compare to other models.
- **Accuracy on Benchmark dataset (GLUE, MMLU)** : can be used to compare to other models throughout the training.

Complementary approaches

- Multi-fidelity : reduce the cost and the reliability of early evaluations. (ex : BOHB algorithm)

03
Design and Implementation

Evaluate the solution

Use LitGPT framework with it's CLI to perform an evaluation of a solution. All models and datasets are taken from HuggingFace Hub.

Training

- Model : Llama-3.2-1B
- dataset : Alpaca
- 1 epochs of training
- Fully Sharded Data Parallelism (FSDP) as distributed strategy

Evaluating

Based on lm_eval library

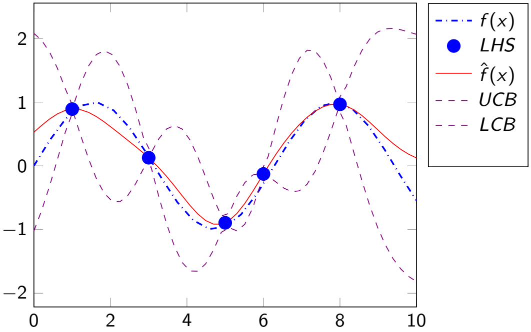
- validation dataset : Hellaswag
- testing dataset : MMLU

SMBO : Bayesian-Optimization based on Gaussian-Process (BO-GP)

Principe :

Iterate these two steps over budget :

1. Build a surrogate of the objective function
2. Optimize the surrogate to find the most promising point to evaluate

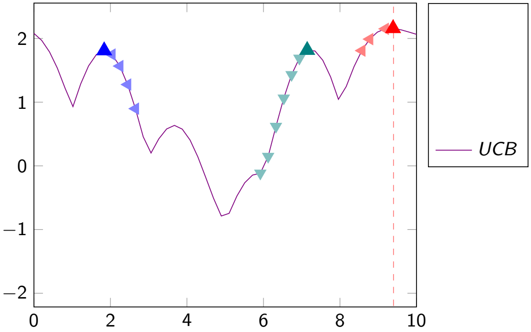


SMBO : Bayesian-Optimization based on Gaussian-Process (BO-GP)

Principe :

Iterate these two steps over budget :

1. Build a surrogate of the objective function
2. Optimize the surrogate to find the most promising point to evaluate



PBO : Simultaneous Optimistic Optimization(SOO)

Principe :

- K-inary partition of the space
- Evaluate the center of each partition
- Expand a maximum of one node by iteration / by depth

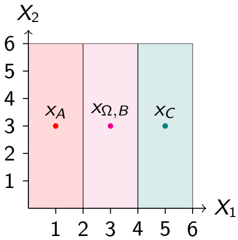


Figure: SOO Partition

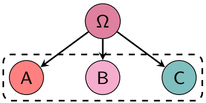


Figure: SOO Tree

PBO : Simultaneous Optimistic Optimization(SOO)

Principe :

- K-inary partition of the space
- Evaluate the center of each partition
- Expand a maximum of one node by iteration / by depth

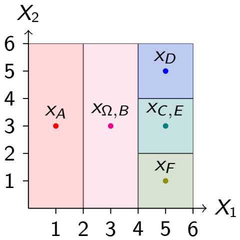


Figure: SOO Partition

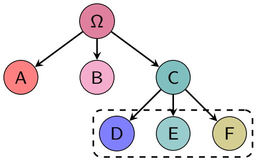


Figure: SOO Tree

PBO : Simultaneous Optimistic Optimization(SOO)

Principe :

- K-inary partition of the space
- Evaluate the center of each partition
- Expand a maximum of one node by iteration / by depth

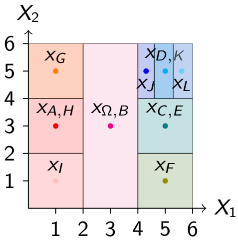


Figure: SOO Partition

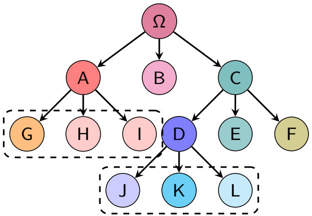


Figure: SOO Tree

Hybridization : Bayesian Multi-Scale Optimistic Optimization (BaMSOO)

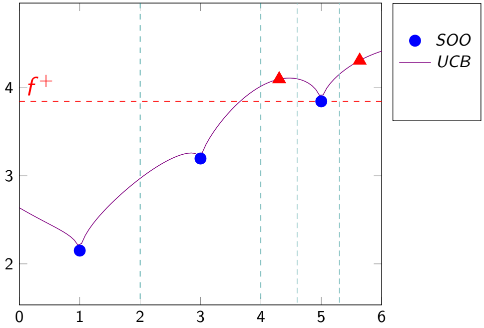
Principe :

- SOO partitionning
- Use a Gaussian process to enhance the scoring

Objective : prevent unpromising evaluations

BaMSOO Scoring $g(x)$:

- **If** $UCB(x) > f^+$: // x has potential to beat f^+
  - $g(x) = f(x)$ // score x using $f(x)$
- **Else** :
  - $g(x) = LCB(x)$ // score x using $LCB(x)$



Hybridization : Bayesian Multi-Scale Optimistic Optimization (BaMSOO)

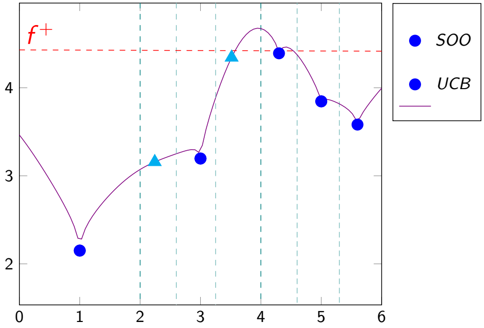
Principe :

- SOO partitionning
- Use a Gaussian process to enhance the scoring

Objective : prevent unpromising evaluations

BaMSOO Scoring $g(x)$:

- **If** $UCB(x) > f^+$: // x has potential to beat f^+
  - $g(x) = f(x)$ // score x using $f(x)$
- **Else** :
  - $g(x) = LCB(x)$ // score x using $LCB(x)$



Hybridization : Bayesian Multi-Scale Optimistic Optimization (BaMSOO)

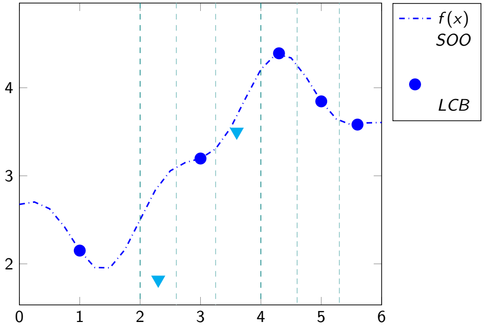
Principe :

- SOO partitionning
- Use a Gaussian process to enhance the scoring

Objective : prevent unpromising evaluations

BaMSOO Scoring $g(x)$:

- **If** $UCB(x) > f^+$: // x has potential to beat f^+
  - $g(x) = f(x)$ // score x using $f(x)$
- **Else** :
  - $g(x) = LCB(x)$ // score x using $LCB(x)$



04
Computation Experiments

Experimental Setup

Experimental testbed

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

Hardware and budget allocated

One evaluation on chuc cluster, using 4*A100 40G of VRAM GPU, is taking around 40 minutes. Each algorithms have a budget of 50 evaluations, including the 10 sampling evaluation of BO.

Sampling experiment : Latin Hypercube Sampling

Objective : Explore the space and define a lower bound for next experiments

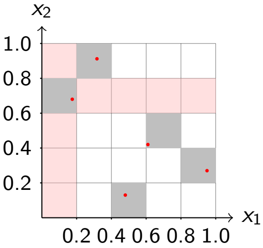


Figure: LHS illustration with $g = 5$ samples

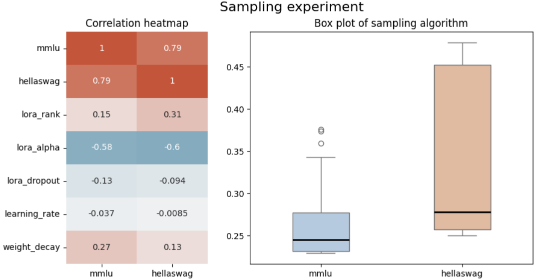


Figure: Results of LHS Experiment[5]

[5] *At left* : correlation between metrics and hyperparameters // At right : metrics distribution

Results : Bayesian Optimization

Score over time

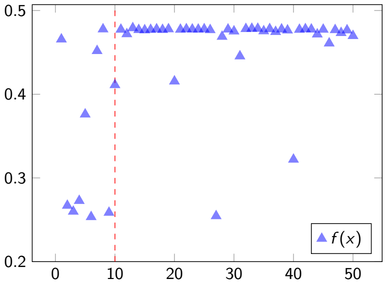


Figure: Results on validation dataset for BO-GP Algorithm

Results

Hellaswag(D_{val}) Best score : 47,91%

Behavior

- Fast convergence after sampling phase
- Few shots emphasing exploration with lower score

Results : SOO

Score over time

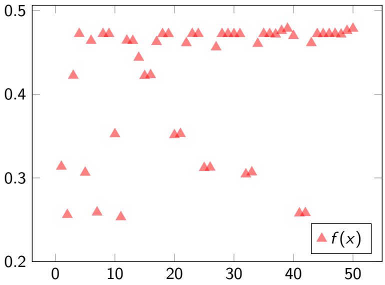


Figure: Results on validation dataset for SOO Algorithm

Results

Hellaswag(D_{val}) Best score : 47.84%

Behavior

- A lot of low score evaluation, due to one hyperparameter
- maximum depth of 8 (only 2 points with depth = 8)

Results : BaMSOO

Score over time

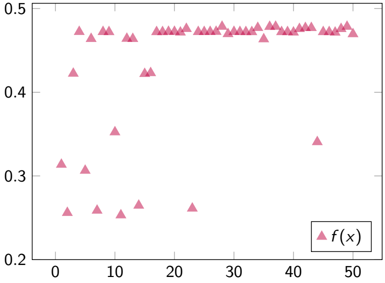


Figure: Results on validation dataset for BaMSOO Algorithm

Results

Hellaswag(D_{val}) Best score : 47.84%
Do not achieve to overperform SOO best score

Behavior

- Prevent SOO unpromising evaluation (16 approximated evaluations)
- maximum depth of 8 (8 points with depth = 8)

Comparison and analysis

Analysis

- Upper Bound on Hellaswag is irrelevant
- Only BO-GP beat LHS
- with more high-performing solution, BaMSOO overperform SOO on MMLU

Results

	Hellaswag	MMLU
Lower Bnd[1]	47.90	37.61
Upper Bnd[2]	*41.5*	49.3
BO-GP	**47.91**	**38.11**
SOO	47.84	37.42
BaMSOO	47.84	37.50

Table: Bound and best score for datasets (val and test)
1 : LHS results; 2 : Meta Fine tuning

Score over time on testing dataset

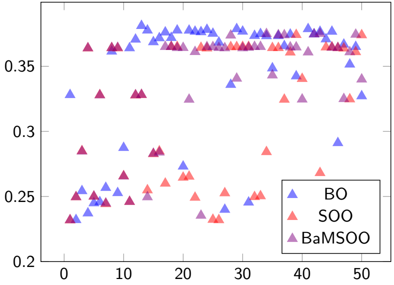


Figure: Results on testing dataset for the three algorithms

05
Conclusions and Perspectives

Conclusion

review

On a sequential comparaison, BO-GP algorithms is the most efficient between theses 3 algorithms, even considering the exploitation made by BamSOO algorithms. But this kind of performance needs to efficiently scale to be able to be usable with very expensive function, especially if the evaluation can't be distributed.
With it's acceleration using GP, BaMSOO keep most of the SOO abilities, in particular it's parallelism inate abilities, but achieve to be efficient with a smaller number of evaluation.
To be able to effectively compare theses approaches, it's necessary to look at higher dimensionnal problem.

Perspective

- Expand search space : add dimensions (Adam momentum, precision, matrices to apply LoRA)
- use more training datasets
- make a distributed implementation

Bibliography I

[1] Mike Conover et al. *Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM*. 2023. URL: https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm (visited on 06/30/2023).

[2] Stefan Falkner, Aaron Klein, and Frank Hutter. "BOHB: Robust and Efficient Hyperparameter Optimization at Scale". In: *CoRR* abs/1807.01774 (2018). arXiv: 1807.01774.

[3] Thomas Firmin and El-Ghazali Talbi. "A fractal-based decomposition framework for continuous optimization". working paper or preprint. July 2022.

[4] Jiahui Gao et al. "AutoBERT-Zero: Evolving BERT Backbone from Scratch". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.10 (June 2022). Number: 10, pp. 10663–10671.

[5] Qingyan Guo et al. *Connecting Large Language Models with Evolutionary Algorithms Yields Powerful Prompt Optimizers*. arXiv:2309.08532 [cs]. Feb. 2024.

Bibliography II

[6] Rohan Taori and Ishaan Gulrajani and Tianyi Zhang and Yann Dubois and Xuechen Li and Carlos Guestrin and Percy Liang and Tatsunori B. Hashimoto. *Stanford Alpaca: An Instruction-following LLaMA model*. publisher : GitHub. Dec. 2024.

[7] Dan Hendrycks et al. “Measuring Massive Multitask Language Understanding”. In: *arXiv preprint arXiv:2009.03300* (2020).

[8] Dan Hendrycks et al. *Measuring Massive Multitask Language Understanding*. arXiv:2009.03300 [cs]. Jan. 2021.

[9] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL].

[10] Donald R. Jones, Cary D. Perttunen, and Bruce E. Stuckman. “Lipschitzian Optimization Without the Lipschitz Constant”. In: *Journal of Optimization Theory and Applications* 79.1 (1993), pp. 157–181.

[11] Aaron Klein et al. “Structural Pruning of Large Language Models via Neural Architecture Search”. en. In: (Oct. 2023).

Bibliography III

[12] Guillaume Lample, Hugo Touvron, Lucas Beeching, et al. *LLaMA 3: Open and Adaptable Foundation Models*. https://github.com/meta-llama/llama3. 2024.

[13] Siyi Liu, Chen Gao, and Yong Li. *Large Language Model Agent for Hyper-Parameter Optimization*. arXiv:2402.01881 [cs]. Feb. 2024.

[14] Rémi Munos. "Optimistic Optimization of a Deterministic Function without the Knowledge of its Smoothness". In: *Advances in Neural Information Processing Systems 24 (NeurIPS)*. 2011, pp. 783–791.

[15] OpenAI. *GPT-4 Technical Report*. https://openai.com/research/gpt-4. 2023.

[16] Rohan Taori et al. *Stanford Alpaca: An Instruction-following LLaMA model*. https://github.com/tatsu-lab/stanford_alpaca. 2023.

[17] Christophe Tribes et al. *Hyperparameter Optimization for Large Language Model Instruction-Tuning*. arXiv:2312.00949. Jan. 2024.

[18] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

Bibliography IV

[19] Alex Wang et al. "GLUE: A multi-task benchmark and analysis platform for natural language understanding". In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. 2018, pp. 353–355.

[20] Chi Wang, Xueqing Liu, and Ahmed Hassan Awadallah. "Cost-Effective Hyperparameter Optimization for Large Language Model Generation Inference". en. In: *Proceedings of the Second International Conference on Automated Machine Learning*. ISSN: 2640-3498. PMLR, Dec. 2023, pp. 21/1–17.

[21] Ziyu Wang et al. "Bayesian multi-scale optimistic optimization". In: *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics* 33 (2014), pp. 1005–1013.

[22] Xingyu Wu et al. *Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap*. arXiv:2401.10034. May 2024.

[23] Rowan Zellers et al. *HellaSwag: Can a Machine Really Finish Your Sentence?* arXiv:1905.07830 [cs]. May 2019.

Thank You.