

Optimization applied to LLM



HyperParameter Optimization to LLM Fine-Tuning


N. Davouse



Summary

1. Introduction
2. Review of Related Works
3. Problem Definition
4. Methodology
5. Experiments
6. Conclusion

01 Introduction



Large Language Models

Summary

- ▶ State-of-the-art of Natural Language Processing (NLP) problems
- ▶ Architecture : Transformers[16] block, mixed with classical layers (MLP, Conv)
- ▶ 2 phases of training : pre-training and fine-tuning

Self Attention

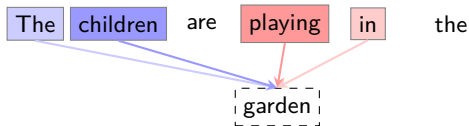


Figure: Scaled dot product attention



Fine Tuning

Parameters Efficient Fine-Tuning (PEFT)

Set of methods aims to reduce the computation cost of fine-tuning. Can change the structure like the 2 following, or just reduce the cost like Quantization (reduce the precision of calculus). These methods are often hyperparameter-dependent.

Low Rank Adaptation (LoRA)[7]

Use of low rank matrices of the weights matrices, which will be the only ones trained, to reduce the cost of gradient computations.

Adapter Layer

Add layer inside the model, and train only these. One con is to add inference for predicting.

Fine-Tuning workflow

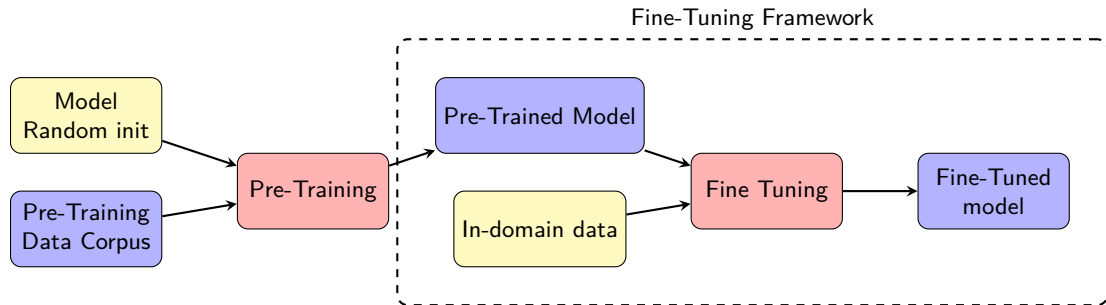


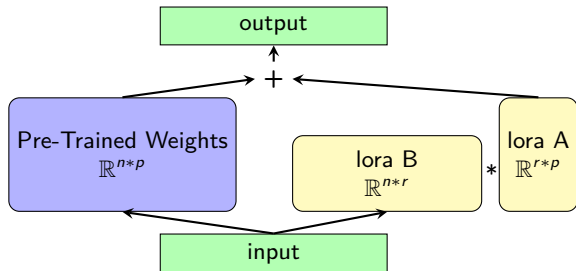
Figure: Pre-training and Fine-tuning generic workflow



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.



LoRA hyperparameters

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

Figure: LoRA Decomposition

02

Review of Related Works





Prompt Engineering

Prompt : process of interacting with an artificial intelligence (AI) system by providing specific instructions or queries to achieve a desired outcome.

Example with article [5], when a second LLM is used to modify the prompt.

Pros

Don't need to deal with architecture,
weights : act like the LLM is a generating
blackbox

Cons

Low impact, locate this work as the
end-user, not so much usable



LLM applied to Optimization

Multiples articles show the use of LLM to develop or code optimization algorithms, in particular Evolutionnary Algorithm. One intersting impact is to popularize the development of optimization algorithm.

Pros

Extend the fields of Meta-Heuristics, with new kinds of operators.

Cons

Low impact, don't achieve remarkable performance.



Auto DNN

Sub-domains

- ▶ **HyperParameter Optimization(HPO)** : Automatically define best hyper-parameters, from training to inference
- ▶ **Neural Architecture Search(NAS)** : Define the best architecture, from scratch or from pruning an existing one

Metrics

- ▶ Performance metrics : Accuracy, Latency
- ▶ Ressource metrics : inference time, memory usage, energy consumption

Summary

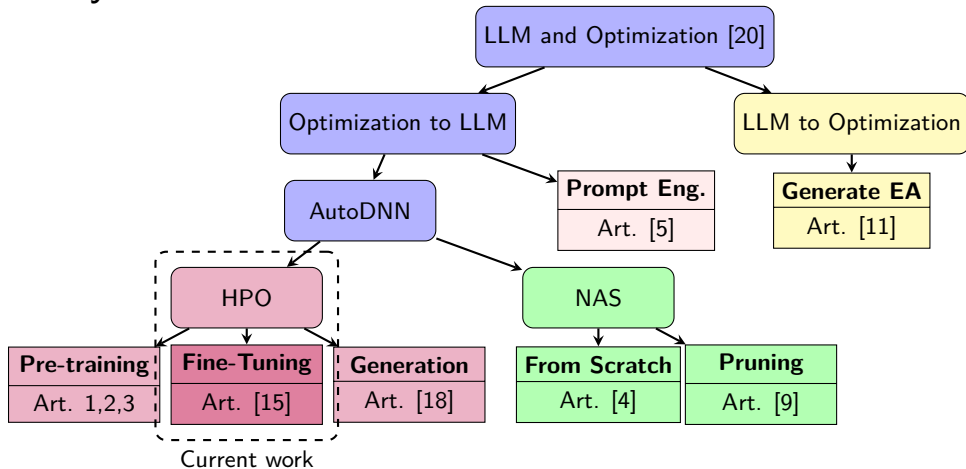


Figure: Summary of links between LLM and Optimization

03

Problem Definition





Problem Definition

Problem Formulation

The HPO problem can be defined as

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

Litterature often define theses problems and their solution with 3 stages : search space, search strategy, performance evaluation strategy.

Search Space

\mathcal{H} is the search space, i.e. the set of all values the solution tuple η can take. This stage includes method to handle the mixed-variables aspect of the problems.

Search Strategy

With η_i all the tested solutions, the search strategy is the method used to define the next solution η_{next} to evaluate.

Performance Evaluation Strategy

\mathcal{F} represent the objective function, and many can be chosen according to a problem. Also includes method like multi-fidelity that affect the fidelity of the evaluation.



Search Space

Hyper-parameters

| Name | Lower Bound | Upper Bound | Type | Steps | Field |
|--------------------|-------------|-------------|-------|-------|-----------|
| Learning Rate | $\exp(-10)$ | $\exp(-1)$ | float | exp | Optimizer |
| Weight Decay | $\exp(-5)$ | $\exp(1)$ | float | exp | Optimizer |
| LoRA Rank | 2 | 500 | int | int | model |
| LoRA alpha (scale) | 1 | 64 | float | cont | model |
| LoRA Dropout | 0 | 1 | float | cont | model |
| grad_batches | 1 | 32 | int | int | Trainer |

Mixed-variables handling

- ▶ Exponential steps : variables defined as a float between $\log(U_{bound})$ and $\log(L_{bound})$, the exponential is applied when it's used for evaluation.
- ▶ Integers variables : variables are relaxed during the generation of the solution, and then round when evaluating.



Search Strategy I

The search strategy of an optimization problem can be seen as a balance between the exploration, i.e. going to unexplored regions, and exploitation, i.e. going close to promising areas. Here are the fields of optimization to tackle HPO problems.



Exploratory Method

Grid Search and Random Search

These 2 methods are the simplest way possible of exploring the search space, without exploiting the acquired knowledge. Useful to assess the pertinence of the optimization algorithm.

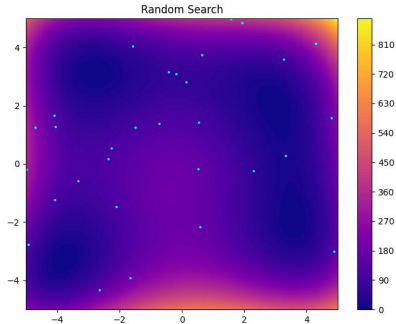


Figure: Random Search

Meta-heuristics

Meta-heuristics are mostly algorithms inspired by nature, divided in 2 approaches : population evolution or individual evolution. For the first one, methods like Genetic Algorithm (GA) allow the fitting of the population to a problem, with evolutionary operators like crossover, mutation or selection. For the second one, like Intensive Local Search (ILS), the methods iterate through the search space with only one solution by iteration.

These methods aren't fit to HPO problems applied to LLM, due of their needs of numerous evaluations, begin computationally prohibitive.



Partition Based Optimization

Partition Based Optimization

Sets of methods that apply partition to the search space, to favor (partition again) or penalize (discard region) these partitions. E.g.: DIRECT[8], Fractals[3], SOO[12]
Useful for parallelization abilities

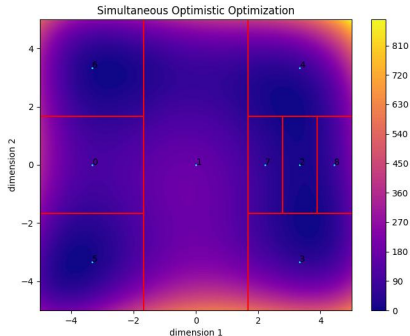


Figure: SOO

Surrogate-Model Based Optimization (SMBO)

Surrogate-Model Based Optimization (SMBO)

Methods based on creating a surrogate model of the objective function, using the knowledge from already evaluated solutions. The acquisition function, i.e., the function of the surrogate model, is used to balance exploration and exploitation.

E.g.: Bayesian Modeling, Gaussian Process (GP), Tree Parzen Estimator (TPE).

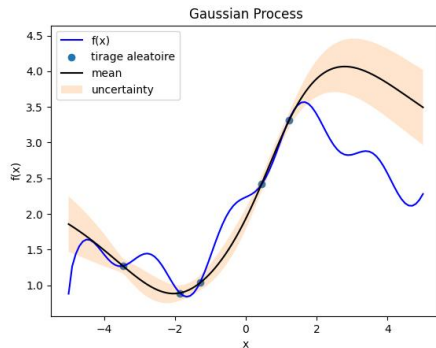


Figure: Gaussian Process



Partition and Surrogate-Model based optimization : the hybridation

Partition and Surrogate-Model based optimization : the hybridation

The partition based methods are by nature parallel, by the generation of a tree-search of possible solutions. The SMBO achieve to reduce the number of evaluation by using an acquisition function to discard or favor a possible solution. The hybridation would result in a parallelization-able Bayesian modeling, allowing to extract the best of the two fields.

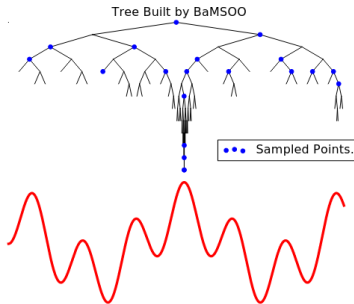


Figure: BaMSOO



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


For this problem, there are 2 ways to evaluate a solution :

- ▶ **Loss (validation or testing)** : the loss is computed through the training, and we can keep a small part of the datasets unused to use it the evaluate the model. Cons : dataset dependant, difficult to put in global context
- ▶ **Benchmark dataset (GLUE[17], MMLU[6])** : the accuracy on a literature benchmark dataset can be used to evaluate the training. It's interesting, since it's a good measure of generalization, since the model has not read this type of questions. Warning : the benchmark used during the optimization can't be used as a final testing.

Multi-fidelity approaches can be used to reduce the cost of evaluation in earlier steps. Algorithms like Bayesian Optimization and HyperBand (BOHB[2]) achieve cost-efficient optimization by reducing the part of the datasets in early stages.

04

Methodology





Global optimization frame

Diagram for the general principle

There are 2 important steps :

- ▶ Generate new solution : the optimization algorithm
- ▶ Evaluate the solution : application of the black-box function

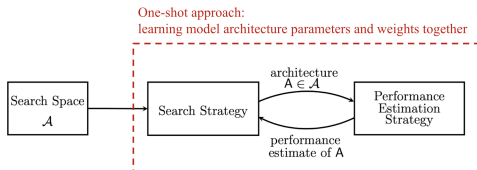


Figure: Optimization frame



Optimization : generate the new solution |

Frameworks

The optimisation part is mostly done with Zellij, an open-source framework made for hyper-parameter optimization, composed of method from metaheuristics to fractals one, passing by bayesian based one.

Some bayesian simulation are directly done with BoTorch.

Used optimization algorithms

Present SOO, BO, BaMSOO



Evaluate the solution I

Training frameworks

- ▶ Model : TinyLLama-1.B, lightweight LLama derived model. Imported with litgpt library.
- ▶ Training dataset : Alpaca[14], a 52k entries Instructions Tuning datasets
- ▶ Training loop : Pytorch Lightning framework, automating the training with class hooks

Algorithm 1 Training pseudocode

```
1: HP = load_hp()
2: data = LLMDDataClass()
3: trainer = TrainerClass(HP)
4: model = LLMMModelClass(HP)
5: trainer.fit(model, data)
6: merge_lora_weight
7: save(model)
```



Evaluate the solution II


Evaluation framework

- ▶ Evaluation framework:
lm_eval library
- ▶ Benchmark dataset MMLU:
imported from HuggingFace
hub

Algorithm 2 Evaluate pseudocode

```
1: model = load_model()
2: eval_dataset = datasets["mmlu"]
3: predictions = model.predict(eval_dataset.input)
4: accuracy = compare(predictions, eval_dataset.output)
5: return accuracy
```

05 Experiments





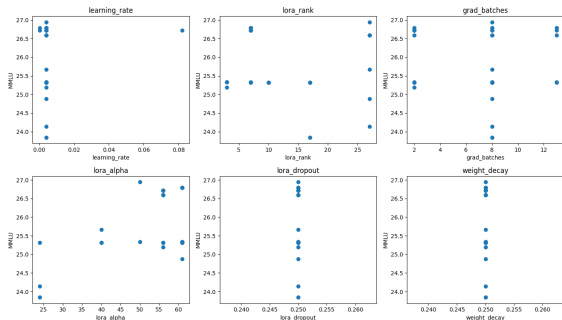
Experiment 1 : Proof of concept using DiRect with Zellij |

Experimental setup

23 hours of run : 72 complete iterations using direct implementation of Zellij.

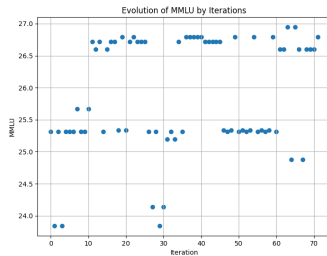
Fixed HP : 1 epochs by training

Variable one by one



Convergence

Best MMLU result : 26,94%



Experiment 2 : First step with Bayesian Optimization |

Experimental setup

50 complete iterations with BoTorch implementation

Hardware : single node with 4*A100 40G

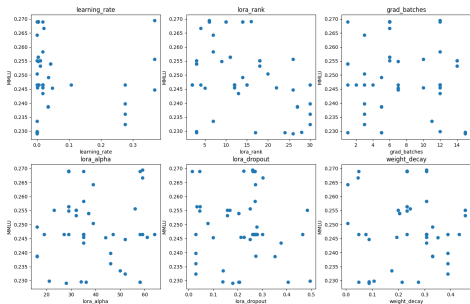
Gaussian Process parameter

- ▶ Acquisition function : Log EI
- ▶ model : MixedSingleTaskGP
- ▶ likelihood : ExactMarginalLogLikelihood



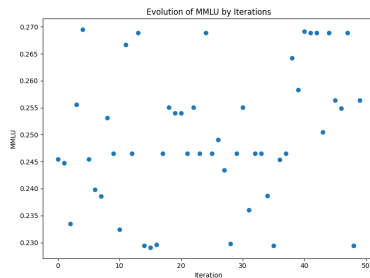
Experiment 2 : First step with Bayesian Optimization II

Variable one by one



Convergence

Best MMLU result : 26,94%



TO DO : look if there is convergence at some point



Experiment 3 : Bayesian Optimization with Latin Hypercube Sampling

Experimental setup

57 complete iterations with BoTorch implementation

Training : one epoch on alpaca cleaned dataset

Evaluation : Hellaswag dataset

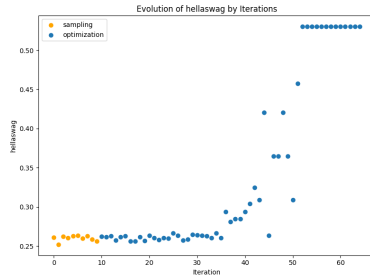
Hardware : single node with 4*A100 40G

Gaussian Process parameter

- ▶ Acquisition function : Log EI
- ▶ model : MixedSingleTaskGP
- ▶ likelihood : ExactMarginalLogLikelihood

Convergence

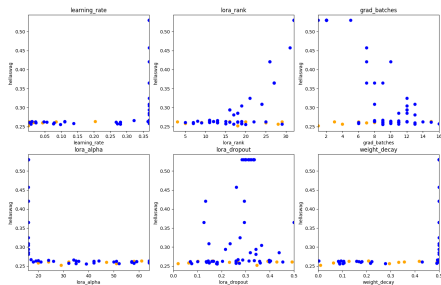
Best Hellaswag result : 53,02%



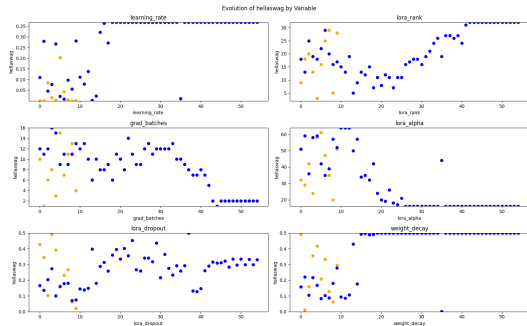


Experiment 3 : Bayesian Optimization with Latin Hypercube Sampling II

Variable one by one



Variable iteration



Results

Need to enlarge search space, since 5 hyperparameters are constrained by bounds.



Conclusion

end



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



Bibliography II

- [7] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL].
- [8] 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.
- [9] Aaron Klein et al. “Structural Pruning of Large Language Models via Neural Architecture Search”. en. In: (Oct. 2023).
- [10] Guillaume Lample, Hugo Touvron, Lucas Beatching, et al. *LLaMA 3: Open and Adaptable Foundation Models*. <https://github.com/meta-llama/llama3>. 2024.
- [11] Siyi Liu, Chen Gao, and Yong Li. *Large Language Model Agent for Hyper-Parameter Optimization*. arXiv:2402.01881 [cs]. Feb. 2024.
- [12] 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.
- [13] OpenAI. *GPT-4 Technical Report*. <https://openai.com/research/gpt-4>. 2023.



Bibliography III

- [14] Rohan Taori et al. *Stanford Alpaca: An Instruction-following LLaMA model*. https://github.com/tatsu-lab/stanford_alpaca. 2023.
- [15] Christophe Tribes et al. *Hyperparameter Optimization for Large Language Model Instruction-Tuning*. arXiv:2312.00949. Jan. 2024.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] 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.



Bibliography IV

- [20] Xingyu Wu et al. *Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap*. [arXiv:2401.10034](https://arxiv.org/abs/2401.10034). May 2024.

Thank You.

