

HyperParameter Optimization of LLM Fine Tuning




Partition and Acquisition based approaches

N. Davouse



Sommaire

01 Introduction





Large Language Models

Summary

- ▶ State-of-the-art of Natural Language Processing (NLP) problems
- ▶ Architecture :
Transformers[NIPS2017'3f5ee243] block,
mixed with classical layers (MLP, Conv)

Self Attention cell

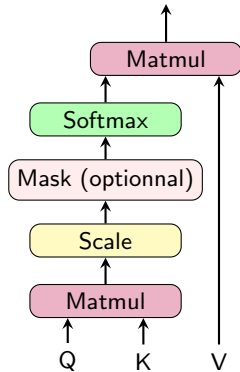
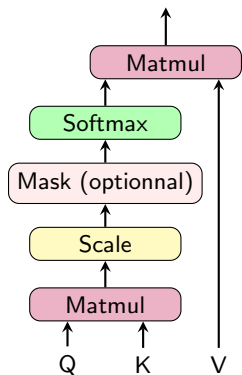


Figure: Scaled dot product attention

Scaled Dot-Product Attention



Multi-Head Attention

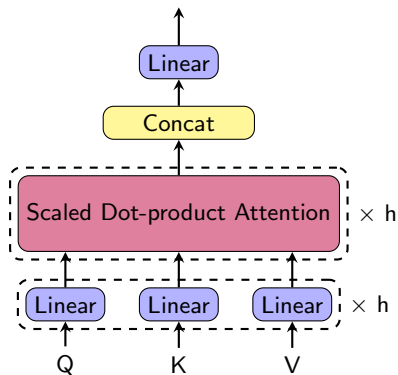


Figure: MHA

self attention

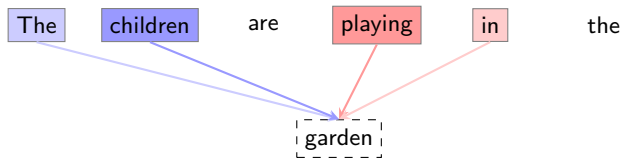


Figure: self attention mechanism



Pre-training and Fine-tuning

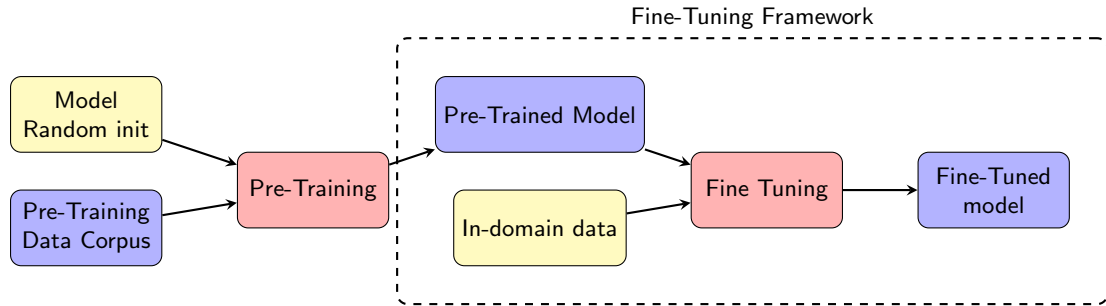


Figure: Pre-training and Fine-tuning generic workflow



Fine Tuning

Instruction Tuning

fine-tunes models to follow specific user instructions effectively. Consists of using specific fine-tuning datasets (Alpaca [alpaca], Databrick's Dolly[DatabricksBlog2023DollyV2])

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)[hu2021loralowerrankadaptationlarge]

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

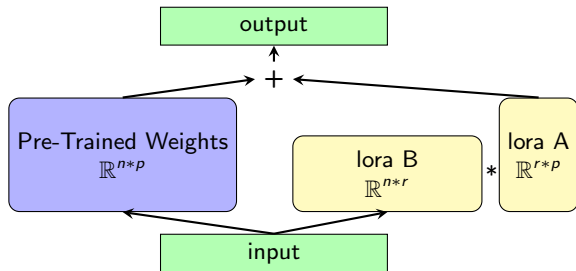
Add layer inside the model, and train only these. One con is to add inference for predicting.



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



Hyper-Parameter Optimization (HPO) I

Hyper-Parameters

In a Deep Learning Context, set of value that the algorithms need to run. Affect the model, the training or the prediction. The hyper-parameters, often called variables, are mixed (i.e. continuous and integers, no categoricals).

Optimization

Set of methods to find the best input value to maximize or minimize an objective function. The problem here can be defined as a black-box optimization (i.e. no analytic form), and an expensive one. An evaluation can take hours.

The Hyper-Parameter Optimization field can be defined as the use of Optimization method to hyper-parameter dependent problem, like Deep Learning ones.

Aims are diversives, from obtaining better results, putting humans out of the loop or ensuring reliability.



Hyper-Parameter Optimization (HPO) II

Generic workflow

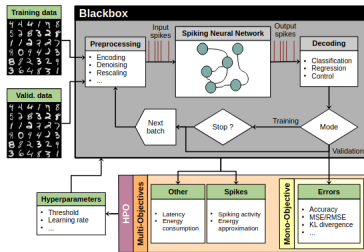


Figure 2.17: Usual workflow of HPO applied to SNNs

Figure: HPO Workflow

02

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.

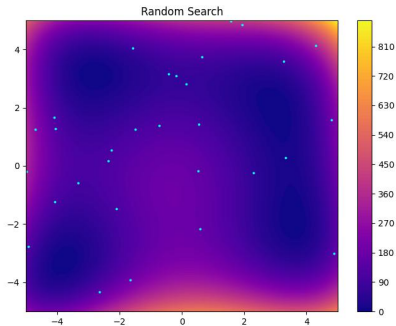


Figure: Random Search

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[jones1993direct],

Fractals[firmin:hal-04474444],

SOO[munos2011soo]

Useful for parallelization abilities

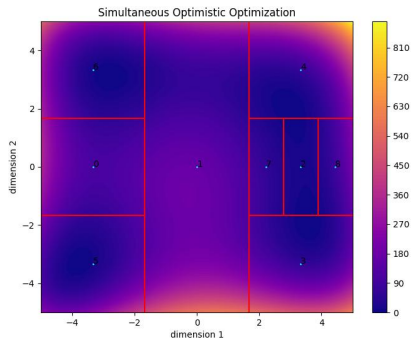


Figure: SOO

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).

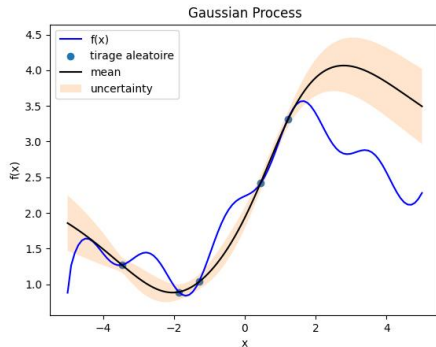


Figure: Gaussian Process



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.

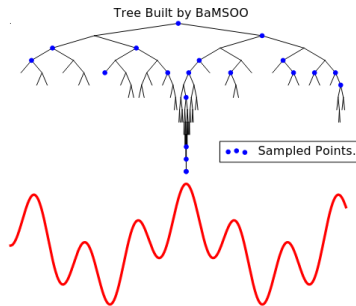


Figure: BaMSOO



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[wang2018glue], MMLU[hendrycks2021mmlu])** : 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[[DBLP:journals/corr/abs-1807-01774](#)]) achieve cost-efficient optimization by reducing the part of the datasets in early stages.

03

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

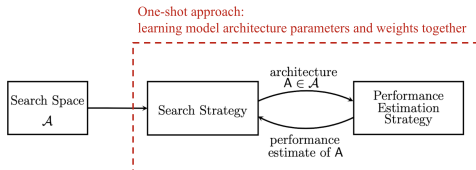


Figure: Optimization frame



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[alpaca], 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 = LLMDataClass()
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
```

04 Experiments





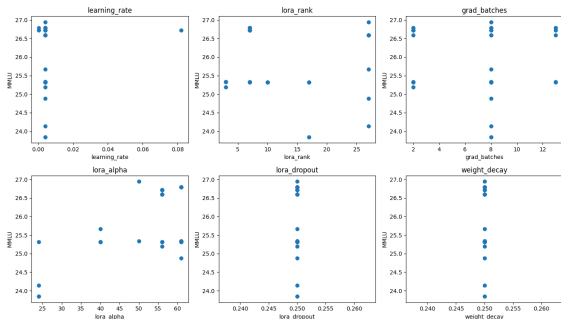
Experiment 1 : Proof of concept using DiRect with Zellij |

Experimental setup

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

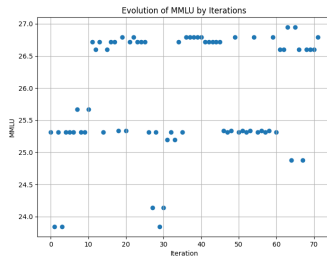
Fixed HP : 1 epochs by training

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

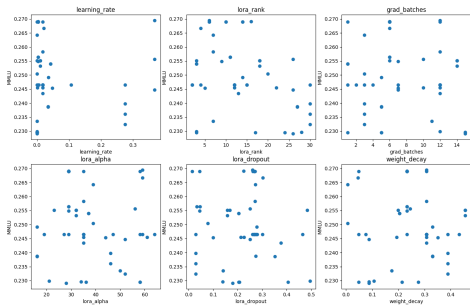
Gaussian Process parameter

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



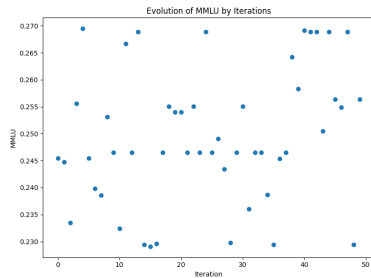
Experiment 2 : First step with Bayesian Optimization II

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

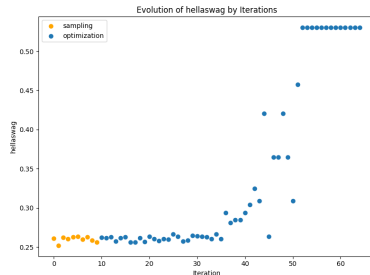
Hardware : single node with 4*A100 40G

Gaussian Process parameter

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

Convergence

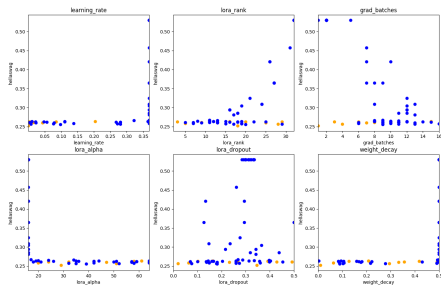
Best Hellaswag result : 53,02%



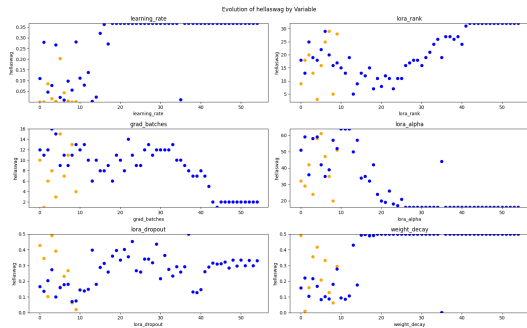


Experiment 3 : Bayesian Optimization with Latin Hypercube Sampling II

Variable one by one



Variable iteration



Results

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



Conclusion

end



Bibliography I

Thank You.

