

Optimization des Hyperparamètres appliquée au Fine Tuning de LLM

Basé sur l'article : *Bayesian and Partition-Based Optimization for Hyperparameter Optimization of LLM Fine-Tuning*

Nathan Davouse

Sommaire

1. Introduction

2. Design et Implémentation

3. Résultats et Analysis

4. Conclusion

Large Language Models

Point clés

- ▶ Etat de l'art pour le traitement de langage naturel.
- ▶ Réseaux de Neurones avec une architecture basé sur le transformer¹ (annexe 1)
- ▶ Taille : entre 1 et 405 Milliards de neurones

¹Vaswani et al, Attention is all you need,2017

Auto-attention

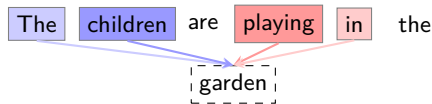


Figure: Illustration du mécanisme d'auto-attention

L'auto-attention est la clé du LLM, en permettant de comprendre le contexte

Sommaire

1. Introduction

2. Design et Implémentation

3. Résultats et Analysis

4. Conclusion

Strategie de Recherche : Optimisation Bayésienne par Process Gaussien

Principe

Utiliser un substitut moins cher à optimiser pour explorer l'espace de recherche

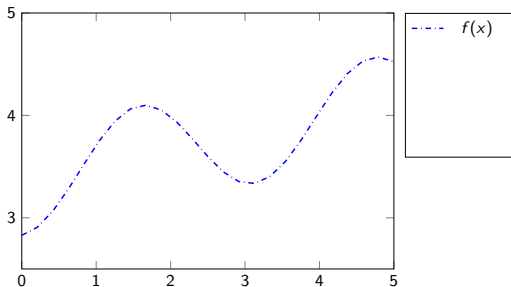


Figure: Exemple d'un surrogate sur une fonction en 1D

Strategie de Recherche : Optimisation Bayésienne par Process Gaussien

Principe

Utiliser un substitut moins cher à optimiser pour explorer l'espace de recherche

Algorithme

- Echantillon de n Points (LHS)

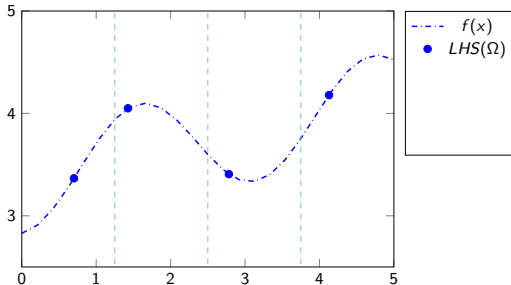


Figure: Exemple d'un surrogate sur une fonction en 1D

Stratégie de Recherche : Bayesian Multi-Scale Optimistic Optimization (BaMSOO)

Décomposition suivant SOO, mais utilisant des Process Gaussien pour éviter les évaluations non prometteuses.

Evaluation par BaMSOO

- ▶ If $UCB(x) > f^+$:
 - $g(x) = f(x)$ real evaluation
- ▶ Else :
 - $g(x) = LCB(x)$ use LCB to replace $f(x)$

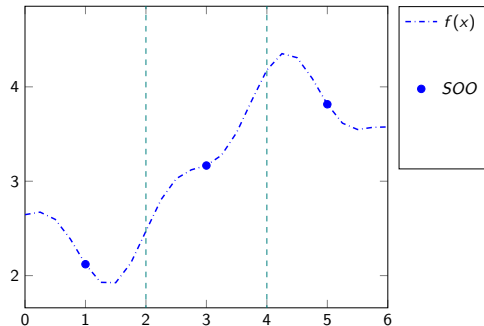


Figure: Illustration de l'Algorithme BaMSOO

Stratégie de Recherche : Bayesian Multi-Scale Optimistic Optimization (BaMSOO)

Décomposition suivant SOO, mais utilisant des Process Gaussien pour éviter les évaluations non prometteuses.

Evaluation par BaMSOO

- If $UCB(x) > f^+$:
 - $g(x) = f(x)$ real evaluation
- Else :
 - $g(x) = LCB(x)$ use LCB to replace $f(x)$

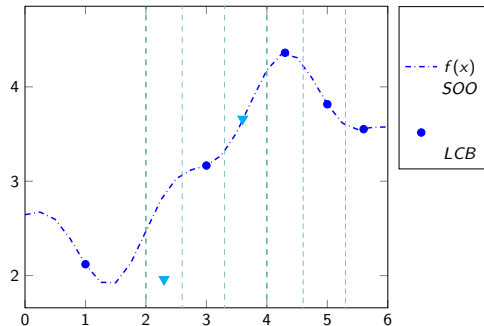


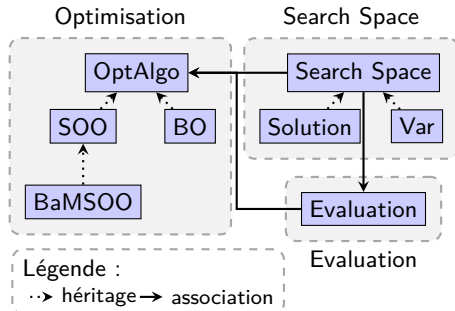
Figure: Illustration de l'Algorithme BaMSOO

Stratégie d'Evaluation de Solutions

Implémentation

- Fine Tuning
 - modèle : LLaMa-3.2-1B
 -
- Evaluation
 - librairie lm_eval
 - Evaluation par la précision sur des jeu de données Benchmark : Hellaswag et MMLU

- ▶ Programmation Orienté Object en Python
- ▶ Travail de documentation : *readme*, indication de type...
- ▶ Objectif : permettre le réusage
- ▶ Utilisable en ligne de commande pour Grid5000
- ▶ Intégralement open-source²



Sommaire

1. Introduction

2. Design et Implémentation

3. Résultats et Analysis

4. Conclusion

Résultats des 3 algorithms

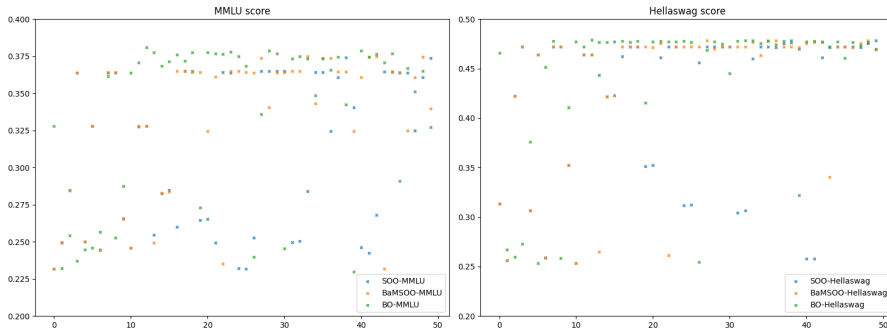


Figure: Résumé des résultats par sampling. Détails par algorithme dans les annexes 4, 5 et 6

Analyse

Jeu de données	Borne Inf. ¹	Borne Sup. ²	BO-GP	SOO	BaMSOO
Hellaswag (validation)	47.90	41.5	47.91	47.84	47.84
MMLU (testing)	37.61	49.3	38.11	37.42	37.50

Table: Bornes et meilleurs résultats sur les 2 jeu de données

1 : expérience avec LHS; 2 : Fine tuning par Meta

Points clés

- ▶ Borne Sup. sur Hellaswag non pertinente
- ▶ Seul BO arrive au dessus de LHS
- ▶ BaMSOO n'améliore que peu SOO
- ▶ principe de BaMSOO fonctionnel (visible annexe 6)
- ▶ Espace de solution n'évolue que peu, le retravailler pour mesurer pleinement la performance des algorithmes

Sommaire

4. Conclusion

Résultats du stage

Une conclusion

Apprentissage

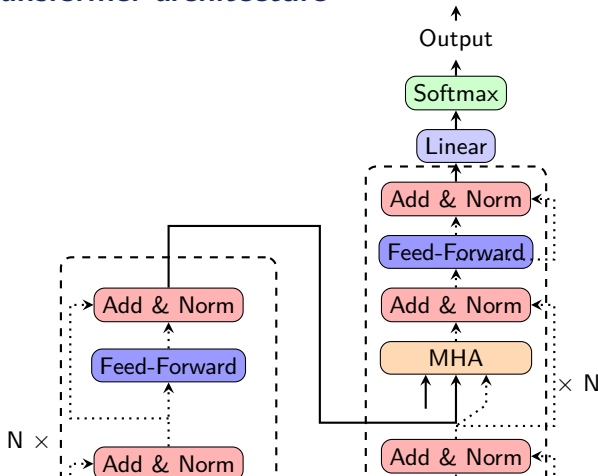
Une conclusion

Poursuite du projet professionnel

Une conclusion

Merci.

Annexe 1 : Transformer architecture



Annexe 2 : Multi-Head Attention

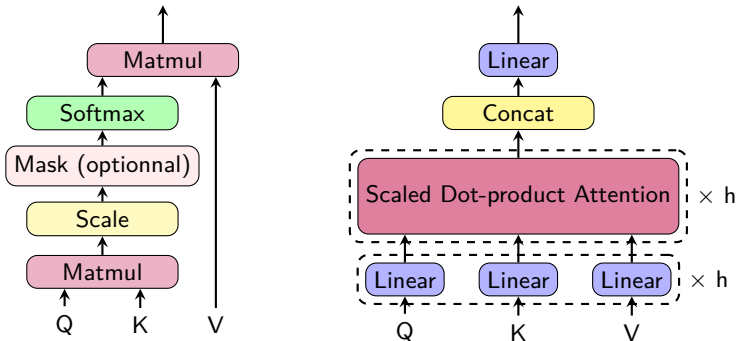


Figure: Illustration du mécanisme d'auto-attention : A droite le mécanisme complet, à gauche le *Scaled Dot-product Attention*

Annexe 3 : Low Rank Adaptation (LoRA)

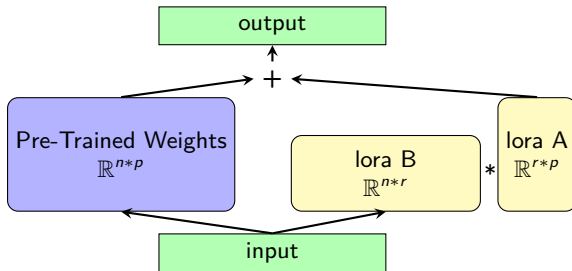


Figure: Illustration de l'application du Low Rank Adaptation (LoRA)

Annexe 5 : Résultats pour S00

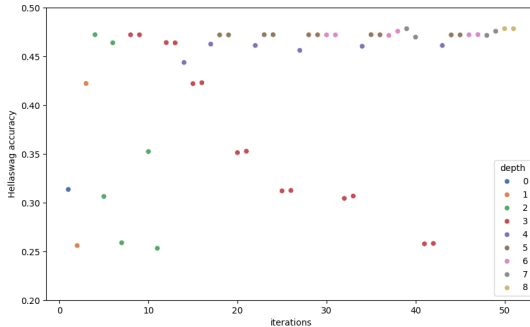


Figure: Evolution des score lors de l'expérience SOO

