CentraleSupélec

université PARIS-SACLAY

Automotatic hyper-parameters optimization for deep learning algorithms

Quentin Goulas - Lucas Hocquette

ICE 24/25

Quentin Goulas

L'objectif de ce rapport est de présenter l'objet d'étude pour le projet d'apprentissage profond associé au cours de la mention ICE du même nom. Ce projet consiste à étudier l'optimisation des hyper-paramètres des algorithmes d'apprentissage profond.

1 Hyper-paramètres classiques des réseaux de neurones

Les hyper-paramètres des réseaux de neurones sont des paramètres qui ne sont pas appris par le modèle, mais qui influent sur la qualité de l'apprentissage. Les principaux hyper-paramètres sont les suivants :

- conception du modèle : nombre de couches cachées et nombre de neurones par couche, type de couche (dense, convolutionnelle, récurrente, etc.), fonction d'activation (sigmoïde, tanh, ReLU, etc.).
- prétraitement : normalisation, augmentation des données.
- algorithme d'optimisation : type d'optimiseur, taux d'apprentissage, taille du lot, nombre d'époques d'entraînement.
- régularisation : taux de régularisation L1/L2, dropout.
- fonction de perte : entropie croisée, erreur quadratique moyenne, etc.

2 Optimisation des hyper-paramètres

Voici quelques méthodes couramment utilisées pour optimiser les hyper-paramètres :

- Baby sitting : essayer différents hyper-paramètres, entraîner le modèle, évaluer la performance, ajuster les hyper-paramètres, répéter. Cela peut être très coûteux en temps et en ressources.
- Grid search : essayer toutes les combinaisons possibles d'hyper-paramètres. Cela peut être très coûteux en temps et en ressources.
- Random search : essayer des combinaisons d'hyper-paramètres aléatoires. Cela peut être plus efficace que la recherche en grille.
- Gradient-based optimization : utiliser des méthodes d'optimisation basées sur le gradient pour optimiser les hyper-paramètres. Cela peut être très coûteux en temps et en ressources.
- Bayesian optimization : utiliser des modèles probabilistes pour optimiser les hyper-paramètres. Cela peut être plus efficace que les méthodes basées sur le gradient.
- Bayesian Optimization and Hyperband hyperparameter tuning (BOHB) [1]
- Particle Swarm Optimization (PSO) et Genetic Algorithm (GA)

Pour des algorithmes de deep learning, PSO (Particle Swarm Optimization) est le meilleur choix d'après [3]. D'autres techniques d'optimisation comme GA (Genetic Algorithm), BO-TPE (Bayesian Optimization with Tree-structured Parzen Estimator) et SMAC (Sequential Model-based Algorithm Configuration) peuvent aussi être utilisés.

Avec ce projet, nous proposons d'implémenter les méthodes au minimum PSO et GA, optionnellement BOHB pour optimiser le nombre et l'épaisseur des couches denses pour un modèle de reconnaissance d'images.

Nous pouvons utiliser le jeu de données CIFAR-10 pour évaluer la performance des différents algorithmes d'optimisation [2]. Ce jeu de donnée nous met à disposition 60000 images de 32×32 pixels, munies d'étiquettes correspondant à 10 types d'images différentes (avion, voiture, ...).

Les métriques que nous surveillerons sur cette comparaison seront de trois types : performance de l'algorithme après convergence (probabilité de reconnaissance de l'image), rapidité de la convergence (probabilité de reconnaissance du modèle à une itération t) et consommation de ressources de l'algorithme (nombre d'itérations nécessaires à la convergence de l'algorithme).

Références

[1] Sai Nikhilesh Kasturi. Bayesian optimization and hyperband (bohb) : Hyperparameter tuning with an example, 2023.

[2] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms, 2012.

[3] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms : Theory and practice. Neurocomputing, 415 :295–316, 2020.