A Summary of BOHB

Yibo Pi, Qinye Li, Jiachen Liu

Problem and Motivation:

The main motivation behind this paper is to improve the hyperparameter optimization process, especially for complicated machine learning tasks and high-dimensional search space. This paper proposes a new hyperparameter optimization algorithm BOHB, combines the benefits of both bayesian optimization and bandit-based methods, in order to achieve the best of both strong anytime performance and fast convergence to optimal configurations.

Hypothesis

Bayesian Optimization (BO) uses an acquisition function $a : X \rightarrow R$ based on the current model that trades off exploration and exploitation. Based on the model and the acquisition function, it iterates the following three steps: (1) select the point that maximizes the acquisition function , (2) evaluate the objective function , and (3) augment the data and refit the model. But due to the long training times of state-of-the-art models, vanilla Bayesian hyperparameter optimization is typically computationally infeasible.

Hyperband (HB) is a bandit strategy that dynamically allocates resources to a set of random configurations and uses successive halving to stop poorly performing configurations and identify the best out of n randomly sampled configurations. However, bandit-based configuration evaluation approaches based on random search lack guidance and do not converge to the best configurations as quickly.

Solution

This paper proposes BOHB which combines Bayesian optimization (BO) and Hyperband (HB). They designed BOHB to satisfy all the desiderata:

1. Strong Anytime Performance
2. Strong Final Performance. Having guidance on finding the best configurations in a large space requires.
3. Effective Use of Parallel Resources. Leveraging large parallel resources (e.g., compute clusters or cloud computing) effectively.
4. Scalability. Practical modern HPO methods must be able to easily handle problems ranging from just a few to many dozens of hyperparameters.
5. Robustness & Flexibility. Different hyperparameter optimization problems and different types of hyperparameters (such as binary, categorical, integer, and continuous) should be handled.
6. Simplicity. Simple to be verified and reimplemented in different frameworks.
7. Computational efficiency.

BOHB relies on HB to determine how many configurations to evaluate with which budget, but it replaces the random selection of configurations at the beginning of each HB iteration by a model-based search. Once the desired number of configurations for the iteration is reached, the standard successive halving procedure is carried out using these configurations. We keep track of the performance of all function evaluations of configurations on all budgets to use as a basis. BOHB follows HB's way of choosing the budgets and continues to use SH, but replaces the random sampling by a BO component to guide the search. BOHB constructs a model and uses BO to select a new configuration, based on the configurations evaluated so far.

Limitations

In the paper, the author mentions that there are other orthogonal methods that could be used to improve BO, such as meta learning , active ensembling to combine models found during the optimization, and using multiple fidelities.

In the evaluation, BOHB has higher time to accuracy performance with the increase of workers, i.e. parallelism, but they didn't evaluate beet resource to accuracy performance (goodput) , where more workers means both faster search speed and waste resources.

Discussion

1. Baysian optimization may suffer from low efficiency in high-dimensional search space, where BO may need lots of prior data to get good estimates on search space. Maybe we could replace it with some other model-based method to improve the search guidance.
2. The evaluation between random search may not be fair because they only focus on wall-clock time but not controlling the resources they are using.

Summary of ASHA

Yibo Pi (yibo), Qinye Li (qinyeli), Jiachen Liu (amberljc)

Problem and Motivation

The main motivation behind this paper is to develop mature hyperparameter optimization functionality in distributed computing settings, for example production environments. This paper introduces a simple and robust hyperparameter optimization algorithm called ASHA built on top of SHA, which exploits parallelism and aggressive early-stopping. It also provides systems level solutions to improve the effectiveness of ASHA that are applicable to existing systems.

Hypothesis

In modern days, the following four trends call for support for massively parallel hyperparameter settings.

1. High-dimensional search space - Models are becoming increasingly complex.
2. Increasing training times - This is onerous since evaluating each candidate configuration requires training a model.
3. Rise of parallel computing - we could leverage distributed computational resources to speed hyperparameter tuning process.
4. Productionization of ML - ML is moving from R&D to production, and so ML infrastructure must mature accordingly.

Solution Overview

This paper introduces Asynchronous Successive Halving Algorithm(ASHA), which is built on top of Successive Halving Algorithm (SHA).
SHA is a hyperparameter tuning algorithm taking an adaptive configuration evaluation approach. SHA allocates a small budget to each configuration,

evaluates all configurations and keeps the top 1/η, increases the budget per configuration by a factor of η, and repeats until the maximum per-configuration budget is reached.
This paper proposes ASHA, which leverages asynchrony to mitigate stragglers and maximize parallelism. Instead of waiting for a rung to complete before proceeding to the next rung, it promotes configuration to the next rung whenever possible. When no promotions are possible, ASHA simply adds a configuration to the base rung, so that more configurations can be promoted to the upper rungs.
ASHA is able to remove the bottleneck associated with synchronous promotion by incurring a small number of incorrect promotions, but the erroneous promotion is expected to vanish as the number of configurations grow.

Limitations and Possible Improvements

In evaluation, ASHA has higher time to accuracy performance, but they didn’t evaluate resource to accuracy performance (goodput) , where simply launching more evaluating jobs to utilize idle workers may cause resource waste.

Summary of Class Discussion

1. In addition to machine learning tasks, hyperparameter tuning could also apply in different domains, like building auto-tune systems.