Continual Hyperband

Jasmin Brandt
University of Paderborn
Paderborn, Germany

Marcel Wever
University of Munich (LMU)
Munich, Germany

Dimitrios Iliadis
Ghent University
Ghent, Belgium

Viktor Bengs
University of Munich (LMU)
Munich, Germany

Eyke Hüllermeier
University of Munich (LMU)
Munich, Germany

***Abstract*—Hyperband is a state-of-the-art method for automatically optimizing the hyperparameters of a machine learning algorithm. However, if the maximal budget used in Hyperband is chosen too small, the budget needs to be increased manually post hoc and another run of Hyperband must be started from scratch, leading to a loss of already observed information and a lot of wasted budget in the first run of Hyperband. We propose in this paper an extension of Hyperband, called Continual Hyperband, that overcomes the above problem and moreover, comes with a qualitatively similar theoretical guarantee as for the original version of Hyperband.**

I. Introduction

The performance, complexity and learning speed of a machine learning algorithm usually highly depends on the choice of its hyperparameters. For this reason, it is important to select suitable hyperparameters for a given learning task. However, manually searching for these parameters is a time-consuming or even infeasible task for the user, which emerged in the research field of automated hyperparameter optimization (HPO). In general, HPO methods aim to find an optimal hyperparameter configuration for a learning algorithm as efficiently as possible, that generalizes well to new and unseen data points.

II. Hyperparameter Optimization

Given a target algorithm $\mathcal{A}$, hyperparameter optimization is the problem of finding a set of optimal hyperparameters for $\mathcal{A}$. Hyperparameters are all parameters of $\mathcal{A}$ that should be set by a user and usually their values control the learning process of the target algorithm. Let $\mathcal{X}$ be the space of all valid hyperparameter configurations for $\mathcal{A}$, then we define a sequence of loss functions $\ell_k : \mathcal{X} \to [0,1], x \mapsto \ell_k(x)$ that represent the validation error of $\mathcal{A}$ trained with hyperparameter configuration x on k resource units. If we denote $\ell_* = \lim_{k \to R} \ell_k$ for a fixed maximal size of iterations R and $\nu_* = \inf_{x \in \mathcal{X}} \ell_*(x)$, the goal of an HPO algorithm is to identify a hyperparameter configuration $x^* \in \text{argmin}_{x \in \mathcal{X}} \ell_*(x) - \nu_*$.
Even if HPO can be considered as a black-box optimization problem and thus can, in theory, be solved by various methods, most of them are impractical through the costly evaluation of a hyperparameter configuration. In the literature, one distinguished between model-free and model-based methods. The latter ones work by learning a surrogate model of the optimization surface to sample more promising candidates. Typical model-free methods are grid search and random search, but in particular the latter is usually too inefficient in the HPO setting. For a more thorough overview, see Bischl et al. (2021).

III. Hyperband

While usually random search methods are considered as too inefficient for an HPO problem, Hyperband (Li et al. (2018)) overcomes this problem by combining random search with a multi-fidelity candidate evaluation routine, called Successive Halving (SH, Jamieson and Talwalkar (2016)). SH iteratively allocates the available budget B to a set of hyperparameter configurations and discards the worse half in each step based on their performance for the current budget. This process is repeated until only one configuration remains which is then returned by the algorithm. Hyperband calls SH multiple times, each time with a different size of the set of hyperparameter configurations n, but with a fixed overall budget B. Due to this, it covers different tradeoffs between considering many different configurations n and gaining more reliable information about the performance of the configurations by giving them longer training time B/n.
For most learning algorithms it is hard to define a suitable maximal budget R for a single run of a configuration beforehand. To overcome this problem, Jamieson and Talwalkar (2016) also proposed an infinite horizon setting, where Hyperband is run repeatedly with an increasing maximal budget R for each run of a configuration. However, each new run of Hyperband is started from scratch and previously evaluated configurations are completely discarded. In other words, the budget of the previous run is wasted and the observed information about the configurations is lost.

IV. Continual Hyperband (C-HB)

For avoiding this severe loss of information, we propose an extension of Hyperband, called Continual Hyperband (Algorithm 1, Brandt et al. (2023)). The idea is again to iteratively enlarge the maximal budget R by a factor η. This results in a larger pool size $s_{\max}$ in comparison to the last run of C-HB (see line 4 in Algorithm 1), while the budget per configuration remains the same in the first iterations of the Continual Successive Halving (C-SH, Algorithm 2) subroutine. Thus, we only need to fill the newly available slots in the pool of configurations (see line 4 in Algorithm 2) and only need to collect performance information for them, because we can compare this information with the already observed information from promoted configurations in the previous run of C-HB. To be as efficient and resource-saving as possible, we do not revoke any previous decisions and keep all configurations in the subsequent iterations of C-SH that were also kept in the previous run of C-HB. So we only need to fill up the top-k configurations for the next iteration of C-SH by the number of available slots in the next iteration minus the number of already promoted configurations in the previous run of C-HB (see line 7 in Algorithm 2). All parts of the algorithms that are different from the original versions of Hyperband and SH are marked in blue in Algorithm 1 and 2.

Algorithm 1 Continual-Hyperband (C-HB)

```
1:  Input: max size $R$, $\eta \geq 2$, old max size $\tilde{R} \in \{0, R/\eta\}$, old configuration samples $((x_{s,k})_{k\in\{0,\dots,s\}})_{s\in\{0,\dots,\log_\eta(\tilde{R})\}}$ and losses $((\ell_{s,k})_{k\in\{0,\dots,s\}})_{s\in\{0,\dots,\log_\eta(\tilde{R})\}}$
2:  Initialize: $s_{max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{max}+1)R$
3:  if $\tilde{R} > 0$ then
4:      $\tilde{s}_{max} = \lfloor \log_\eta(\tilde{R}) \rfloor = s_{max} - 1$, $\tilde{B} = (\tilde{s}_{max}+1)\tilde{R}$
5:  end if
6:  for $s \in \{s_{max}, s_{max}-1, \dots, 0\}$ do
7:      $n_s = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil$, $r_s = R/\eta^s$
8:      if $\tilde{R} > 0$ and $s > 0$ then
9:          $\tilde{s} = s - 1$, $\tilde{n}_s = \lceil \frac{\tilde{B}}{\tilde{R}} \frac{\eta^{\tilde{s}}}{(\tilde{s}+1)} \rceil$, $\tilde{r}_s = \tilde{R}/\eta^{\tilde{s}} = r_s$
10:     else
11:         $\tilde{n}_s = 0$
12:     end if
13:     $S \leftarrow$ sample $n_s - \tilde{n}_s$ configurations
14:     C-SH($S, B, r_s, R, \eta, (x_{\tilde{s},k})_{k\in\{0,\dots,\tilde{s}\}}, (\ell_{\tilde{s},k})_{k\in\{0,\dots,\tilde{s}\}}$)
15: end for
16: Output: argmin $(\ell_{s,k})_{k,s}$
```

Algorithm 2 Continual-SuccessiveHalving (C-SH)

```
1:  Input: $S$ set of arms, budget $B$, $r$, max size $R$, $\eta$, $(x_k)_k$ old configurations, $(\ell_k)_k$ old losses
2:  Initialize: $S_0 \leftarrow S$, $\tilde{n} = |x_0|$, $n = |S_0| + |x_0|$
3:  for $k \in \{0, 1, \dots, s\}$ do
4:      $n_k = \lfloor n/\eta^k \rfloor - \lfloor \tilde{n}/\eta^k \rfloor$ , $r_k = r\eta^k$
5:      pull each arm in $S_k$ for $r_k$ times
6:      if $k \leq s - 1$ then
7:          $S_{k+1} \leftarrow$ keep best $\lfloor n/\eta^{k+1} \rfloor - \lfloor \tilde{n}/\eta^{k+1} \rfloor$ arms from $S_k \cup x_k \backslash x_{k+1}$
8:      else
9:          $S_{k+1} \leftarrow$ keep best $\lfloor n/\eta^{k+1} \rfloor$ arms from $S_k \cup x_k$
10:     end if
11: end for
12: Output: Remaining configuration
```

V. Theoretical Guarantee

Since an optimal configuration x^* as defined above does not always exist, we will focus in the following on near-optimal configurations. We call $\hat{x}$ an ϵ-optimal configuration iff $\nu_{\hat{x}} - \nu_{x^*} \leq \epsilon$. For a fixed proportion α of ϵ-optimal configurations in the configuration space, we can derive the following guarantee, the proof of which is given in Brandt et al. (2023).

Theorem 5.1: Let η, R, α and δ be fixed such that

$$\begin{aligned} R \geq \max\Big\{ & \lceil \log_{1-\alpha}(\delta) \rceil (\eta - 1) + 1, \\ & \eta\Big(\log_\eta(\log_\eta(R)) + 4 + \lfloor \log_\eta(R) \rfloor / 2 \\ & - \log_\eta\left((\lfloor \log_\eta(R) \rfloor + 1)! \right) / (\lfloor \log_\eta(R) \rfloor + 1) \Big) \bar{\gamma}^{-1} \Big\} \end{aligned}$$

for

$$\begin{aligned} \bar{\gamma}^{-1} := \max_{s=0,\dots,\lfloor \log_\eta(R) \rfloor} \max_{i=2,\dots,n_s} i\Big(1 \\ + \min\Big\{R, \gamma^{-1}\big(\max\{\epsilon/4, (\nu_i - \nu_1)/2\}\big)\Big\}\Big). \end{aligned}$$

Then C-HB finds an ϵ-optimal configuration with probability at least $1 - \delta$.

References

Bernd Bischl et al. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1484, 2021.

Jasmin Brandt, Marcel Wever, Dimitrios Iliadis, Viktor Bengs, and Eyke Hüllermeier. Iterative deepening hyperband. *arXiv preprint*, 2023.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *AISTATS*, volume 51 of *PMLR*, pages 240–248, 2016.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *JMLR*, 18(185):1–52, 2018.