OVERFITTING IN BAYESIAN OPTIMIZATION: AN EMPIRICAL STUDY AND EARLY-STOPPING SOLUTION

Anastasia Makarova[1], **Huibin Shen**[2*], **Valerio Perrone**[2], **Aaron Klein**[2], **Jean Baptiste Faddoul**[2], **Andreas Krause**[1], **Matthias Seeger**[2], **Cedric Archambeau**[2]
[1] *ETH Zürich, Zürich, Switzerland* [2] *Amazon Web Services, Berlin, Germany*
[1] {anmakaro,krausea}@inf.ethz.ch
[2] {huibishe,vperrone,kleiaaro,faddoul,matthis,cedrica}@amazon.com

ABSTRACT

Bayesian Optimization (BO) is a successful methodology to tune the hyperparameters of machine learning models. The user defines a metric of interest, such as the validation error, and BO finds the optimal hyperparameters that minimize it. However, the metric improvements on the validation set may not translate to improvements on the test set, especially when tuning models trained on small datasets. While cross-validation can mitigate this, it comes with an increased computational cost. In this paper, we carry out the first systematic investigation of overfitting in BO and demonstrate that this issue is a serious, yet often overlooked concern in practice. We propose the first problem-adaptive and interpretable criterion to early stop BO, reducing overfitting while mitigating the cost of cross-validation. Experimental results on real-world hyperparameter optimization tasks show that our approach can substantially reduce compute time with little to no loss of test accuracy, demonstrating a practical advantage over existing techniques.

1 INTRODUCTION

The performance of machine learning algorithms crucially depends on their hyperparameters. Tuning hyperparameters is usually a tedious and expensive process. Bayesian optimization (BO) is a popular approach to optimize gradient-free functions, and has recently gained traction in HPO by obtaining state-of-the-art results in tuning many modern machine learning models Chen et al. (2018); Snoek et al. (2012); Melis et al. (2018). Despite the wide usage of BO for HPO, to the best of our knowledge, its potential for overfitting has not been studied. As we show in Section A, overfitting is indeed occurring, and exhibits different characteristics than classical overfitting in training machine learning algorithms. On the one hand, we can not mitigate overfitting by directly adding a regularization term due to the gradient-free nature of BO. On the other hand, classical early stopping in deep learning training Prechelt (1996); Li et al. (2020) cannot be directly applied due to the explorative and global nature of BO. Finally, while cross-validation is a common technique to mitigate overfitting, it comes with high computational cost, and it is unclear how to effectively use it with BO.

In this work, we propose a termination rule for BO building on cross-validation that is problem-adaptive and easy to incorporate into standard BO framework. Our main contributions are as follows:

- We empirically study overfitting in BO, being the first to our knowledge to address it.
- We propose a simple yet powerful stopping criterion that is problem adaptive and interpretable. The method exploits existing BO components, thus, is easy to use in practice.
- Our experiments show that our method matches or outperforms other early stopping baselines by a large margin in test accuracy while maintaining a valuable speed-up in compute-time.

We present our empirical study of overfitting in BO in Appendix A, and introduce our stopping criterion in Section 2. In Section 3, we evaluate it on real-world hyperparameter optimization problems and compare it with the other baselines. We also provide further insight by discussing the related work and challenges for BO early stopping in Appendix B and Section 4.

*Correspondence to: Huibin Shen <huibishe@amazon.com>

2 REGRET BASED STOPPING

In this section we propose our novel regret-based stopping criterion for BO, which employs cross-validation. Assume we have a learning algorithm h_γ defined by its hyperparameters $\gamma \in \Gamma$ and parametrised by a weight vector $\mathbf{w}$: $h_\gamma(.;\mathbf{w})$. Let $\mathcal{D} = \{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^n$ be the collected dataset drawn from unknown data distribution $(\mathbf{x}, \mathrm{y}) \sim P_\mathcal{D}$. The goal of HPO is to find the best hyperparametrs optimizing the expected loss $\mathbb{E}_{\mathbf{x},\mathrm{y}\sim P_\mathcal{D}} \ell(\mathrm{y}, h_\gamma(\mathbf{x}, \mathbf{w}))$. In practice, the data distribution is unknown, and an empirical estimate is used instead. The data $\mathcal{D}$ is split into $\mathcal{D}_{tr}$ and $\mathcal{D}_{val}$, used for training and validation. Formally, the optimization problem over hyperparameters and weights is as follows:

$$
\begin{aligned}
f(\gamma; \mathbf{w}, \mathcal{D}) &= \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}_i, y_i \in \mathcal{D}} \ell(\mathrm{y}_i, h_\gamma(\mathbf{x}_i, \mathbf{w})) \\
\mathbf{w}^*(\gamma) &= \arg\min_{\mathbf{w} \in W} f(\gamma; \mathbf{w}, \mathcal{D}_{tr}), \\
\gamma^* &= \arg\min_{\gamma \in \Gamma} f(\gamma; \mathbf{w}^*(\gamma), \mathcal{D}_{val}).
\end{aligned}
$$

BO aims to find the global maximizer γ^* by leveraging a probabilistic function model, e.g., *Gaussian process (GP)*, and an acquisition function. The GP posterior about value $f(\gamma)$ is defined by posterior mean $\mu_t(\gamma)$ and posterior variance $\sigma_t^2(\gamma)$. Convergence of BO can be quantified by the *simple regret*:

$$
R_T := f(\gamma_t^*) - f(\gamma^*).
$$

where γ^* are the optimal hyperparameters. It defines the sub-optimality in function value. However, the optimum $f(\gamma^*)$ is rarely known, and R_T can not be computed in practice. In the following, we propose a stopping criterion for BO which relies on two building blocks: an upper bound on the simple regret and an adaptive threshold that is based on the sample variance obtained via cross-validation.

2.1 UPPER BOUND FOR SIMPLE REGRET

We firstly use the following simple bounds for $f(\gamma_t^*)$ and $f(\gamma^*)$:

$$
f(\gamma_t^*) := \min_{\gamma \in \mathfrak{D}_t} f(\gamma) \leq \min_{\gamma \in \mathfrak{D}_t} \mathrm{ucb}(\gamma | \mathfrak{D}_t), \tag{1}
$$

$$
f(\gamma^*) \geq \min_{\gamma \in \Gamma} \mathrm{lcb}(\gamma | \mathfrak{D}_t), \tag{2}
$$

where $\mathrm{ucb}(\gamma|\mathfrak{D}_t) = \mu_t(\gamma) + \sqrt{\beta_t}\sigma_t(\gamma)$, $\mathrm{lcb}(\gamma|\mathfrak{D}_t) = \mu_t(\gamma) - \sqrt{\beta_t}\sigma_t(\gamma)$, and β_t are appropriate constants for the confidence bound to hold and are studied in Srinivas et al. (2010). This results into:

$$
f(\gamma_t^*) - f(\gamma^*) \leq \min_{\gamma \in \mathfrak{D}_t} \mathrm{ucb}(\gamma|\mathfrak{D}_t) - \min_{\gamma \in \Gamma} \mathrm{lcb}(\gamma|\mathfrak{D}_t) := \hat{R}_t. \tag{3}
$$

This upper bound $\hat{R}_t$ is also used for BO with unknown search space in Ha et al. (2019). Loosely speaking, they have shown with high probability, this bound will shrink to a very small value after enough BO iterations under certain conditions. For more details on the theoretical aspects, we refer readers to Theorem 5.1 in Ha et al. (2019).

2.2 STOPPING THRESHOLD

Given the validation metrics from different splits, besides mean, one can also compute variance of these metrics. Let us use s_{cv}^2 to denote this sample variance. We are interested in the variance of the cross-validation estimate of the generalization performance. A simple post-correction technique to estimate it is proposed by Nadeau & Bengio (2003) and is as follows:

$$
s^2 = \left(\frac{1}{K} + \frac{|\mathcal{D}_{val}|}{|\mathcal{D}_{tr}|}\right) s_{cv}^2, \tag{4}
$$

where $|\mathcal{D}_{tr}|$ and $|\mathcal{D}_{val}|$ are sizes of the training and the validation sets in K-fold cross-validation. We use 10-fold cross-validation, thus, the post correction constant on the variance s_{cv}^2 is ≈ 0.21.

In BO, we have s^2 for every $\gamma \in \mathfrak{D}_t$, and for the stopping threshold we need to decide on using an average estimate of $\overline{s^2}$ or a specific $s^2(\gamma)$ for some γ. Our ablation study on the correlation between the sample variance and its mean performance shows that sample variance indeed depends on the hyperparameter configuration, thus we propose to use only the variance of the incumbent $s^2(\gamma_t^*)$.

Now we are ready to introduce our stopping criterion. Given $\hat{R}_t$ as the upper bound of the distance to the optimal function value at iteration t and s as the standard deviation of the generalization error estimate for the current incumbent, we terminate BO if the following condition is met:

$$\hat{R}_t < s(\gamma_t^*). \tag{5}$$

The stopping condition has the following interpretation: Once the maximum plausible improvement becomes less than the standard deviation of the generalization error estimate, further evaluations will not reliably lead to an improvement in the generalization error. The variance-based threshold is problem specific and adapts to a particular algorithm and data.

Computation complexity. The stopping rule reuses existing BO components. The computation time for the upper-bound Eq. (3) is negligible compared to fitting the GP (at most 10% for the fitting time in our experiments for 200 evaluations).

3 EXPERIMENTS

We study how the speed-up gained from the early stopping affects the final test performance. To this end, we firstly compare our method to the setting with the default number of iterations, and then evaluate the existing stopping criteria, such as Nguyen et al. (2017b); Lorenz et al. (2016). We present experimental results on tuning 3 common models, Linear Model trained with SGD (LM), Random Forest (RF) and XGBoost (XGB), on 19 small datasets (less than 10k instances) with 10-fold cross-validation. We describe the setup in Appendix C and the figures are in Appendix D.

Metrics. Given BO budget T, we compare the test error when early stopping is triggered y_{es} to the test error y_T. We compute *relative test error change* as RYC $= \frac{y_T - y_{es}}{\max(y_T, y_{es})}$. RYC allows aggregating the results over different algorithms and datasets as RYC $\in [-1, 1]$, and can be interpreted as follows: a positive RYC represents an improvement in the test error when applying early stopping, while a negative RYC indicates the opposite. Similarly, let the total training time for a predefined budget T be t_T and the total training time when early stopping is triggered be t_{es}. Then the *relative time change* (RTC) is defined as RTC $= \frac{t_T - t_{es}}{t_T}$. A positive RTC, where RTC $\in [0, 1]$, indicates a reduction in total training time.

3.1 COMPARING TO DEFAULT BUDGET

We firstly study our stopping criterion for all datasets and algorithms under the predefined BO budget $T = 200$ and visualize the corresponding RYC and RTC scores in Fig. 5. One can see that in the experiments, where our early stopping is triggered, many RYC scores are non-negative, showing that our method was able to either improve or match the default test error. However, there are a few cases where our method leads to worse test errors and thus negative RYC scores.

We further demonstrate the effectiveness of our early stopping criterion under different BO budgets $T = \{50, 100, 150, 200\}$ and show how much we can improve over the default setting for T. The resulting distributions of RTC and RYC in Fig. 6 demonstrate that our method is effective under all budgets: stopping does not harm the solution on average as RYC scores are concentrated around 0 while the speed up is noticeable especially for large budgets.

3.2 COMPARING TO NÄIVE CONVERGENCE TEST

We compare our method with a näive convergence test controlled by a parameter i: BO is stopped once the *best* observed validation metric remains unchanged for i consecutive iterations. This method mimics the early stopping during algorithm training with two notable differences: First, we only track the validation metrics of the incumbent instead of the suggested hyperparameters at every iteration because the later may underperform due to the exploration nature in BO. Second, defining a threshold is not necessary as the incumbent may stay the same for many iterations and then suddenly change.

This convergence condition heavily relies on a predefined i and optimal i is different across experiments. We consider values commonly used in practice, such as $i = \{10, 30, 50\}$. The resulting distributions for RYC and RTC are presented in Fig. 7. Our adaptive stopping condition not only outperforms by the best average RYC score, but also results in the smallest variance thus delivering a more robust solution. Moreover, for some cases, it outperforms the baselines by a large margin, e.g., for XGBoost, it improves from RYC$= -0.016$ ($i = 50$) to RYC$= -0.002$ and, for random forest, it improves from RYC$= -0.011$ to RYC$= -0.003$ (3.6 times better) while being 1.3 times slower than the convergence check with $i = 50$.

3.3 Comparing to other stopping criteria

Finally, we study two existing conditions for terminating BO, both relying on a predefined threshold that has to be tuned. The first one terminates BO once the value of the Expected Improvement (EI) acquisition function drops below the threshold Nguyen et al. (2017b). The second one uses a mixed approach and defines the termination threshold for the Probability of Improvement (PI) over the incumbent while still using EI as the acquisition function Lorenz et al. (2016). By relying on EI and PI, these stopping criteria inherit their exploration-exploitation trade-off. However, these approaches are not problem adaptive as they rely on a fixed threshold.

We follow the recommendations from Nguyen et al. (2017b); Lorenz et al. (2016) and firstly consider several values for each of the thresholds: for EI based stopping we use $\{10^{-9}, 10^{-13}, 10^{-17}\}$, and for PI based stopping we use $\{10^{-5}, 10^{-9}, 10^{-13}\}$. Empirically, we observe that lower thresholds lead to worse RYC-RTC trade-off: it decreases the average RTC score only by around 5% while increasing the average RYC scores only by around 0.5%. This highlights the challenge of setting the threshold properly for each experiment. As a result, we report only the results of using 10^{-17} for EI based stopping and 10^{-13} for PI based stopping. Fig. 8 illustrates the corresponding distribution of RTC and RYC scores for our method and these two baselines.

The EI and PI based stopping criteria behave similarly in terms of both RTC and RYC scores. The methods tend to stop BO much earlier than our method, thus leading to significant speed up as shown in the left of Fig. 8. However, and not surprisingly, such aggressive early stopping leads to worse test performance on average, as shown in the right of Fig. 8. Moreover, the variance of the test performance for the baselines is larger, which is in contrast to the robustness provided by our method.

One can observe that XGBoost is not early stopped as frequently as the other two algorithms. We suspect that it is because XGBoost has 9 tuning hyperparameters (the others have 3) and it is commonly known that GP works well in a low dimensional setting. To validate this, we repeat the XGBoost tuning experiments but with only three hyperparameters (n_estimators, max_depth and learning_rate) and we denote this new tuning task as XGB (small). We then compare our early stopping results when tuning XGBoost with these two search spaces in Fig. 9. Indeed, when tuning XGBoost with only 3 hyperparameters (thus easier for GP to model), the average RTC score is improved by 50%. However, comparing to the speed up in tuning LM and RF, it is still relatively low.

4 Conclusion

We proposed a novel stopping criterion that is problem adaptive, simple to implement. It comes with no extra hyperparameters, and is agnostic to the specific BO method. In real-world experiments, we demonstrated that our method adapts successfully to the tuning task at hand. We found that our proposal is robust and consistently finds solutions that have lower variance than baselines.

This paper opens several venues for future work. First, while our method tends to improve the test error from 5 to 10 times compared to baselines, it can be slower on average. Future work could reduce the computational cost by making the stopping strategy less conservative. Second, the variance estimate in Eq. (5) relies on cross-validation, which can be computationally expensive. As the upper bound on the regret Eq. (3) has a clear interpretation, a promising alternative is to let users specify a threshold in Eq. (5) even without cross-validation.

REFERENCES

Yutian Chen, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver, and Nando de Freitas. Bayesian optimization in alphago, 2018.

Zhongxiang Dai, H. Yu, K. H. Low, and Patrick Jaillet. Bayesian optimization meets bayesian optimal stopping. In *ICML*, 2019.

Huong Ha, Santu Rana, Sunil Gupta, Thanh Nguyen, Hung Tran-The, and Svetha Venkatesh. Bayesian optimization with unknown search space. In *NeurIPS*. 2019.

Johannes Kirschner, Ilija Bogunovic, Stefanie Jegelka, and Andreas Krause. Distributionally robust bayesian optimization. 2020.

A. Klein, Stefan Falkner, Jost Tobias Springenberg, and F. Hutter. Learning curve prediction with bayesian neural networks. In *ICLR*, 2017.

Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In *AISTATS*, 2020.

Romy Lorenz, Ricardo P Monti, Ines R Violante, Aldo A Faisal, Christoforos Anagnostopoulos, Robert Leech, and Giovanni Montana. Stopping criteria for boosting automatic experimental design using real-time fmri with bayesian optimization, 2016.

Mark McLeod, Stephen Roberts, and Michael A. Osborne. Optimization, fast and slow: optimally switching between local and Bayesian optimization. In *ICML*, 2018.

Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. In *ICLR*, 2018. URL `https://openreview.net/forum?id=ByJHuTgA-`.

Claude Nadeau and Yoshua Bengio. Inference for the generalization error. *Machine learning*, 52(3): 239–281, 2003.

Thanh Dai Nguyen, Sunil Gupta, Santu Rana, and Svetha Venkatesh. Stable bayesian optimization. In Jinho Kim, Kyuseok Shim, Longbing Cao, Jae-Gil Lee, Xuemin Lin, and Yang-Sae Moon (eds.), *Advances in Knowledge Discovery and Data Mining*, 2017a.

Thanh Tang Nguyen, Sunil Gupta, Huong Ha, Santu Rana, and Svetha Venkatesh. Distributionally robust bayesian quadrature optimization, 2020.

Vu Nguyen, Sunil Gupta, Santu Rana, Cheng Li, and Svetha Venkatesh. Regret for expected improvement over the best-observed value and stopping condition. 11 2017b.

Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the Trade*. Springer, 1996.

Garvesh Raskutti, Martin J. Wainwright, and Bin Yu. Early stopping and non-parametric regression: An optimal data-dependent stopping rule. *J. Mach. Learn. Res.*, 2014.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *NeurIPS*, 2012.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th ICML*, ICML'10, pp. 1015–1022, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

Kevin Swersky, Jasper Snoek, and R. Adams. Freeze-thaw bayesian optimization. *ArXiv*, abs/1406.3896, 2014.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml. *ACM SIGKDD Explorations Newsletter*, 2014.

A OVERFITTING IN HPO

We empirically assess overfitting in BO-based HPO and outline its characteristics. We consider tuning three common algorithms, i.e., Linear Model trained with SGD (LM)[1], Random Forest (RF) and XGBoost (XGB), on 19 datasets from various sources, mostly from OpenML Vanschoren et al. (2014). We set 200 hyperparameter evaluations as the budget for BO and repeat each experiment with 10 seeds. Each combination of an algorithm, dataset and seed is referred to as an *experiment* throughout the paper. The detailed hyperparameter search space for the algorithms, the properties of the datasets and data splits, as well as the BO specification are listed in Appendix C.

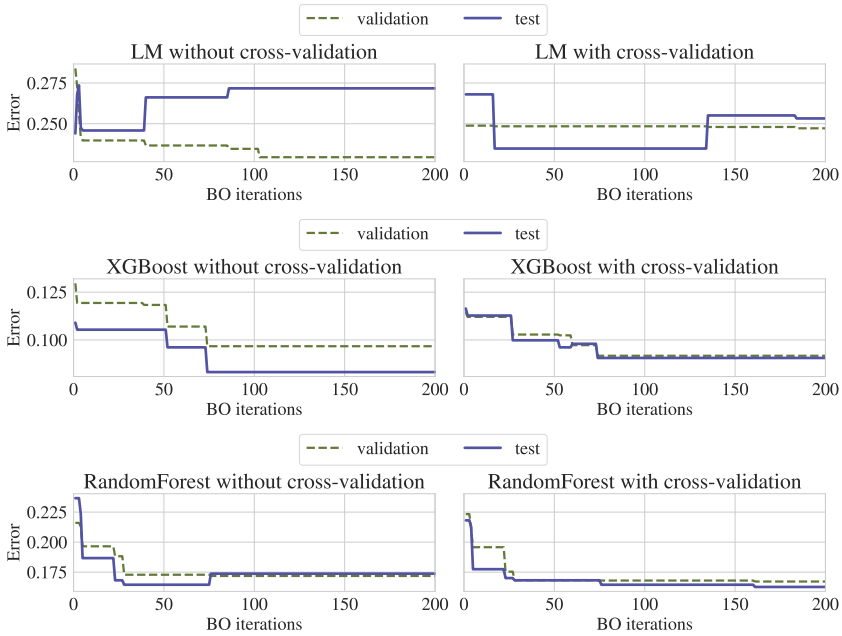

Figure 1: Validation and test errors of the best current hyperparameters. Each plot represents one experiment with 200 BO iterations for LM, XGBoost and RandomForest on the `op100-9952` data with cross-validation (right) and without (left).

During BO, we maintain an *incumbent*, i.e., the hyperparameters with the best validation error found so far. While the validation error of the incumbent is non-increasing by definition, the *test error* corresponding to the incumbent may reveal a different picture. In the following, we use the BO results on one particular dataset to demonstrate interesting observations. Unless emphasised, the observations generalize to other settings.

Non-monotonicity of the test error. In Fig. 1, we plot the validation and test errors of the incumbent with and without cross-validation. While the validation error is indeed decreasing, the test error behaves non-monotonically. This behavior contrasts with the "textbook" setting with a minimal point between underfitting and overfitting. For XGB and RandomForest, less overfitting is observed, indicating that some algorithms are more robust to their hyperparameters than others.

Cross-validation is the *de facto* method to mitigate overfitting and we indeed observe an improvement in the test errors overall when using cross-validation estimates in the HPO procedure. However, cross-validation does not solve the overfitting problem, as we show in Fig. 1. Even with cross-validation, the test errors can still increase (as the experiments for LM show).

Variance in BO experiments. For `op100-9952` data, we compute the *variances* of validation and test errors at every BO iteration across 100 replicates in Fig. 2. From Fig. 2, one can see that the validation errors converge and the test errors are increasing on average for LM. The test error variance is much higher than the validation error when tuning on this dataset.

[1]It is implemented with SGDClassifier (logloss) and SGDRegressor in Scikit-learn.

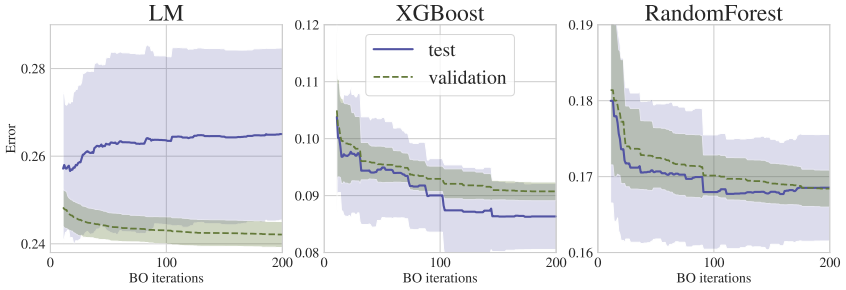

Figure 2: Mean validation and test errors $+/-$ std (y-axis) over 100 experiments with 200 BO iterations (x-axis) for LM, XGBoost and RandomForest on `op100-9952` dataset with cross-validation. Even with cross-validation, the variance in test performance can be high even in the later stage.

There are three sources of randomness in the BO experiments: (i) randomness in reshuffling the dataset and splitting the data into K folds (controllable by cross-validation seed), (ii) randomness in the BO procedure including the random initialization and optimization of the acquisition function (controllable by BO seed), (iii) randomness in the model training, e.g., from stochastic gradient descent or model parameter initialization (controllable by training seed).

We study the impact of randomness inherited from these three sources in Fig. 3 by designing the following experiments: To estimate the variance from cross-validation splits, we fix the BO seed and algorithm training seed, only allow dataset to be reshuffled, and repeat the BO experiments 10 times. Then we get one estimate of the variance from cross-validation for every BO iteration. To make the estimate more reliable, we then repeat this experiment for 10 different configurations of BO seed and algorithm training seed (as an outer-loop) to compute 10 estimates of the variances from cross-validation. In the end we report the mean estimate of the variances from cross-validation in Fig. 3 for every BO iteration. Similarly, we get 10 estimates of variance from BO (fixing cross-validation seed and algorithm training seed) and algorithm training (fixing cross-validation seed and BO seed) and report the mean of the variances from these two sources also in Fig. 3.

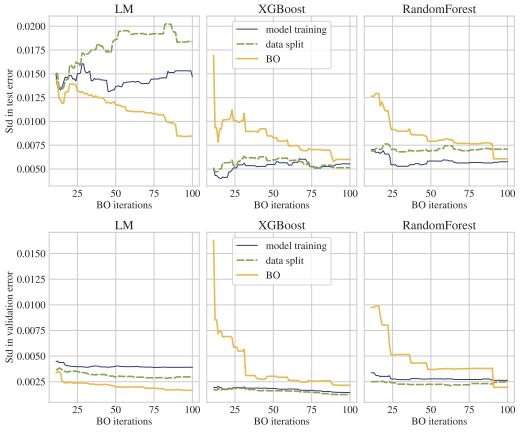

Figure 3: Disentangled sources (model training, data split and BO) of variance for `op100-9952`: std of test error (top row) and validation error (bottom row).

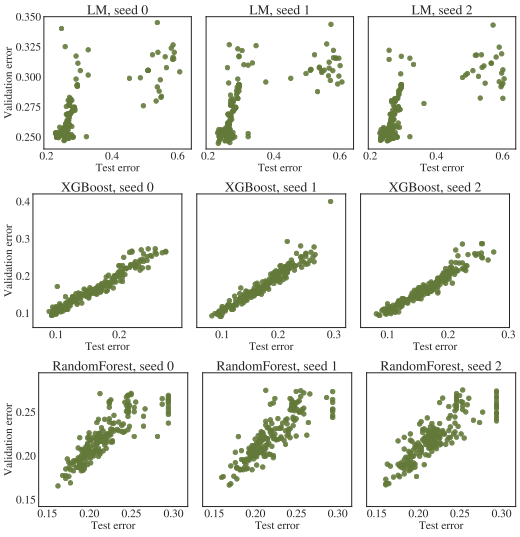

Figure 4: Scatter plot for validation and test errors on the `op100-9952` data with cross-validation. Each point is one hyperparameter evaluation.

There are many observations one can make from Fig. 3. First, the variance from BO tends to decrease in both validation and test errors, and it is the largest source of variance for tuning XGB and RF. The variance from algorithm training is the highest for LM. The variance from cross-validation data splits is usually on a similar scale as the variance from algorithm training, at least for XGB and RF.

Why does overfitting happen? As we have seen when tuning LM on the op100-9952 dataset, the test errors behave drastically different from validation errors, while for XGB and RF, less overfitting is happening. We conjecture that this is because the correlation between the validation and test errors of the hyperparameter configurations is weak. We illustrate this correlation in Fig. 4 where we plot the test and validation errors for all hyperparameters observed in the experiments.

From Fig. 4, we indeed observe a weaker correlation between the test and validation errors for LM. In practice, the correlation between the test and validation errors can be indeed weak, due to the small size of datasets or data shifts. However, we do not have access to the test set during BO, thus we do not know how good the correlation is beforehand. Fortunately, when using cross-validation, the reliability of the validation metrics can be estimated, and it serves as a key component of our stopping criterion.

In conclusion, we have shown that overfitting can indeed happen in BO-based HPO, with perhaps unusual characteristics compared to "classical" overfitting. Running BO longer does not necessarily lead to better generalization performance, thus some form of early stopping for BO may be beneficial both for the solution quality and the computational cost. The variance of tuning the same algorithm on the same dataset can be large; the differences among different algorithms and datasets can also vary. As a result, the early stopping method needs to be *adaptive and robust* to diverse scenarios.

B Related work

Overfitting and robustness in BO are relatively underexplored areas. One direction is to early stop BO and only a few works by Nguyen et al. (2017b); Lorenz et al. (2016) study automated termination. These methods rely on a preselected threshold for acquisition functions which degrades performance when misspecified. McLeod et al. (2018) combine local and Bayesian optimization by selecting from multiple acquisition functions at each iteration, defining an automatic stopping rule for the resulting algorithm. However, this approach still comes with a stopping tolerance hyperparameter, which significantly affects results and yet has to be manually set by the user.

Besides BO early stopping, another direction to robustify a solution is to consider distributional data shifts (Kirschner et al. (2020); Nguyen et al. (2020)) or incorporate aleatoric uncertainty (Nguyen et al. (2017a)). Kirschner et al. (2020); Nguyen et al. (2020) propose to optimize the expected loss under the worst adversarial data distribution rather than the commonly used uniform distribution. The aleatoric uncertainty in Nguyen et al. (2017a) is used to measure the sensitivity of the solution under perturbations of the input.

The term *early stopping* commonly refers to terminating the training loop of algorithms that are trained iteratively, such as neural networks optimized via SGD or XGBoost Prechelt (1996); Raskutti et al. (2014); Li et al. (2020). This iterative training is exploited for BO-based HPO in Klein et al. (2017); Dai et al. (2019); Swersky et al. (2014) as a way to save resources and prevent overfitting. Their notion of early stopping is different, and in a way complementary to the method proposed in our paper. Hence, we refer to our proposal as *BO early stopping*.

C Experiments detail

We optimize classification error or rooted mean square error computed by cross-validation. As these errors are positive, we incorporate this prior knowledge by modelling its log transformation. We use the number of hyperparameter evaluations as the BO budget. We apply early stopping only after the first 20 iterations, to ensure robust fit of the surrogate models both for our method and the baselines. The only hyperparameter of our method is β_t that is set such that confidence bounds in Eqs. (1) and (2) hold with high probability. We set β_t and scale it down by a factor of 5 according to Theorem 1 and experiments in Srinivas et al. (2010) .

BO settings. We used an internal BO implementation with EI, Matern-52 kernel and the following GP hyperparameters: output noise, a scalar mean value, bandwidths, 2 input warping parameters and a scalar covariance scale parameter. The closest open-source implementations are GPyOpt using input warped GP [2] or AutoGluon BayesOpt [3].

[2] `https://github.com/SheffieldML/GPyOpt`

[3] `https://github.com/awslabs/autogluon`

tasks	hyperparameter	search space	scale
LM	l1_ratio	$[10^{-7}, 1]$	log
	alpha	$[10^{-7}, 1]$	log
	eta0	$[10^{-5}, 1]$	log
XGBoost	n_estimators	$[2, 2^{9}]$	log
	learning_rate	$[10^{-6}, 1]$	log
	gamma	$[10^{-6}, 2^{6}]$	log
	min_child_weight	$[10^{-6}, 2^{5}]$	log
	max_depth	$[2, 2^{5}]$	log
	subsample	$[0.5, 1]$	linear
	colsample_bytree	$[0.3, 1]$	linear
	reg_lambda	$[10^{-6}, 2]$	log
	reg_alpha	$[10^{-6}, 2]$	log
RandomForest	n_estimators	$[1, 2^{8}]$	log
	min_samples_split	$[0.01, 0.5]$	log
	max_depth	$[1, 5]$	log

Table 1: Search spaces for algorithms.

dataset	problem_type	n_rows	n_cols	n_classes	source
openml14	classification	1999	76	10	openml
openml20	classification	1999	240	10	openml
tst-hate-crimes	classification	2024	43	63	data.gov
openml-9910	classification	3751	1776	2	openml
farmads	classification	4142	4	2	uci
openml-3892	classification	4229	1617	2	openml
sylvine	classification	5124	21	2	openml
op100-9952	classification	5404	5	2	openml
openml28	classification	5619	64	10	openml
philippine	classification	5832	309	2	data.gov
fabert	classification	8237	801	2	openml
openml32	classification	10991	16	10	openml
openml34538	regression	1744	43	-	openml
tst-census	regression	2000	44	-	data.gov
openml405	regression	4449	202	-	openml
tmdb-movie-metadata	regression	4809	22	-	kaggle
openml503	regression	6573	14	-	openml
openml558	regression	8191	32	-	openml
openml308	regression	8191	32	-	openml

Table 2: Datasets used in our experiments including their characteristics and sources.

Search space of 3 algorithms. Linear Model with SGD (LM), XGBoost (XGB) and RandomForest (RF) are based on scikit-learn implementations and their search spaces are listed in Table 1.

Datasets. The details about the datasets are listed in in Table 2. We randomly draw 20% as test set and for the rest, we use 10-fold cross validations for regression and 10-fold stratified cross validation for classification. For the experiments without cross-validation, we fix the validation set to one of the 10 folds, and the rest 9 folds are used for the training set.

D SUPPORTING FIGURES

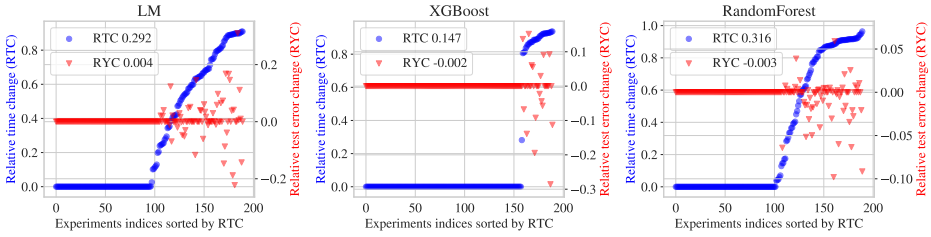

Figure 5: Speed up with our early stopping (RYC) and test error change with BO budget 200 (RTC). Each dot represents one of 190 experiments (19 datasets, 10 seeds) sorted by RTC, on the x-axis. RTC is shown on the *left* y-axis in blue, RYC is shown on the *right* y-axis in red, their averages are in the legend. Note that early stopping is not triggered for more 50% cases as RTC and RYC are zeros.

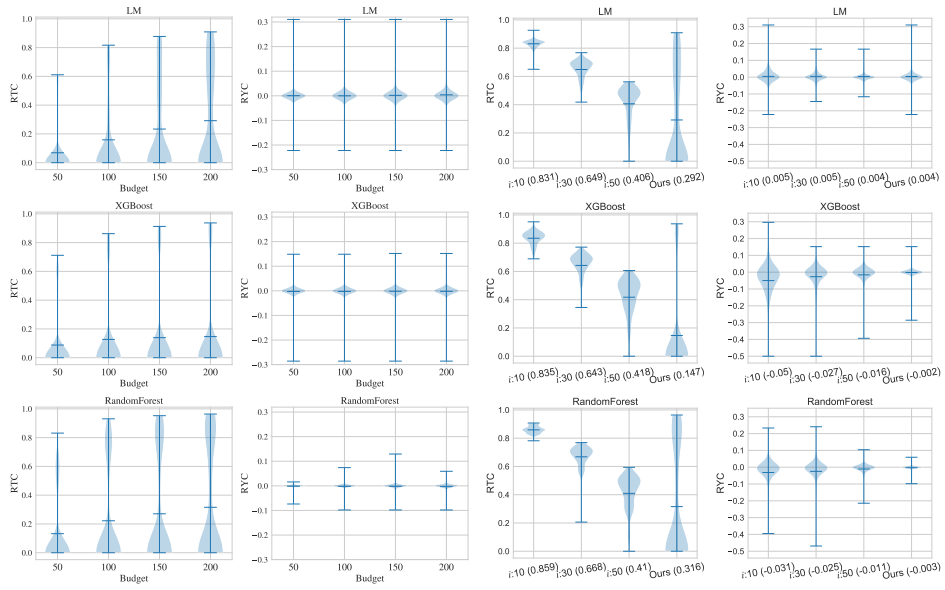

Figure 6: BO with early stopping under different budgets over all datasets and models.

Figure 7: Convergence check baseline. The average score is shown below the violins.

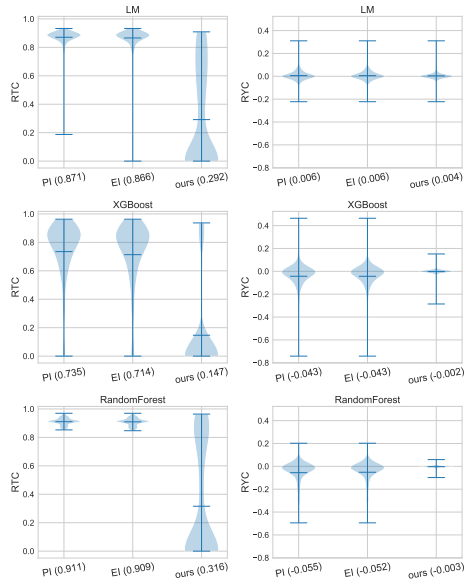

Figure 8: Violin plot for RTC (left) and RYC (right) scores for our method and two baselines based on EI and PI. The average score is also shown as a horizontal line in the middle of each violin, as well as in the parenthesis next to the method labels on the x-axis.

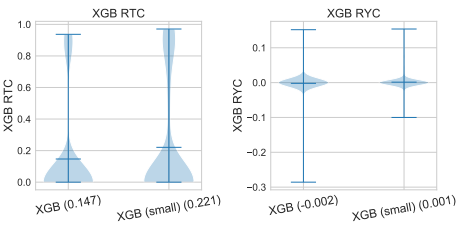

Figure 9: Violin plot for RTC (left) and RYC (right) scores for tuning XGBoost (`XGB`) with 9 HPs and 3 HPs (`XGB (small)`).