Overfitting in Bayesian Optimization: an empirical study and early-stopping solution

Anastasia Makarova
anmakaro@inf.ethz.ch
ETH Zürich
Zürich, Switzerland

Huibin Shen*
huibishe@amazon.com
Amazon Web Services
Berlin, Germany

Valerio Perrone
vperrone@amazon.com
Amazon Web Services
Berlin, Germany

Aaron Klein
kleiaaro@amazon.com
Amazon Web Services
Berlin, Germany

Jean Baptiste Faddoul
faddoul@amazon.com
Amazon Web Services
Berlin, Germany

Andreas Krause
krausea@ethz.ch
ETH Zürich
Zürich, Switzerland

Matthias Seeger
matthis@amazon.com
Amazon Web Services
Berlin, Germany

Cedric Archambeau
cedrica@amazon.com
Amazon Web Services
Berlin, Germany

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. In other words, unlike conventional wisdom dictates, BO can overfit. 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.

ACM Reference Format:
Anastasia Makarova, Huibin Shen, Valerio Perrone, Aaron Klein, Jean Baptiste Faddoul, Andreas Krause, Matthias Seeger, and Cedric Archambeau. 2021. Overfitting in Bayesian Optimization: an empirical study and early-stopping solution. In *Proceedings of KDD '21: ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '21).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

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

KDD '21, August 14–18, 2021, Singapore

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The performance of machine learning algorithms crucially depends on their hyperparameters. Tuning hyperparameters is usually a tedious and expensive process. For this reason, there is a need for automated hyperparameter optimization (HPO) schemes that are sample efficient and robust. 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 [2, 14, 25].

BO optimizes an expensive gradient-free function by iteratively evaluating it at carefully chosen locations: it builds and sequentially updates a *probabilistic model* of the function, uses an *acquisition function* to select the next location to evaluate, and repeats. Consider an example of optimizing a neural network: here, "locations" correspond to choosing a given architecture and hyperparameter configuration. The model is evaluated by optimizing the weights of the neural network on the training set (e.g., via SGD), and estimating its loss on a validation set. This estimated performance is returned to the HPO algorithm to guide the search. While for certain HPO algorithms their convergence to optimal configurations can be shown [26, 29], such analyses generally assume that the number of steps T goes to infinity. In practice, however, BO is terminated after a *finite* number of iterations T. After these T iterations, BO outputs the best hyperparameters configuration based on the validation loss. One may notice several issues with this approach: (a) BO uses the validation metric to guide the search, and thus it may overfit to this metric, especially on small datasets; (b) Incorrectly fixing the number of BO iterations in advance can lead either to sub-optimal solutions or a waste of computational resources.

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 3, 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 [11, 21, 22] cannot be directly applied due to the explorative and global nature of BO. Finally, while cross-validation is a common technique to detect and mitigate overfitting, it comes with high computational cost, and it is unclear how to effectively use it with BO.

Although stopping criteria are critical for BO, only a few works [12, 19] study automated termination. These methods rely on a preselected threshold for acquisition functions which degrades performance when misspecified. In [13], the authors 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.

Contributions. 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. The intuition of our method is the following: stop BO when the maximum plausible improvement becomes less than the (post-corrected) standard deviation of the cross-validation metrics. In particular, our method relies on two components: (i) A high-probability bound on the gap of the validation metrics between the current best hyperparameter configuration and the optimal configuration [4] and (ii) a stopping threshold based on the variance of the cross-validation estimate of the generalization performance from [16].

Our main contributions are as follows:

- We present an empirical study of overfitting in BO, being the first to our knowledge to address this question.
- 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 experimental results 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 Section 3, and then introduce our stopping criterion in Section 4. In Section 5 we experimentally 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 Sections 2 and 6.

2 RELATED WORK

Overfitting and robustness in BO are relatively underexplored areas. Besides BO early stopping, another direction to robustify a solution is to consider distributional data shifts [8, 18] or incorporate aleatoric uncertainty [17]. In [8, 18], the objective is to optimize the expected loss under the worst adversarial data distribution rather than the commonly used uniform distribution. The approach is used for HPO in [18], where it also relies on cross-validation and makes performance more robust under the data shift. However, it does not scale to higher dimensional problems. The aleatoric uncertainty in [17] is used to measure the sensitivity of the solution under perturbations of the input.

Beyond BO, different stopping criteria were also proposed in other areas such as active learning [1, 7]. In [1], the authors predict a change in the objective function to decide when to stop. In [7], the authors propose statistical tests to track the difference in expected generalization errors between two consecutive evaluations.

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 [11, 21, 22]. This iterative training is exploited for BO-based HPO in [3, 9, 27] 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*.

3 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 [28]. 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 A. We use the same settings for evaluating our proposed early stopping method in Section 5.

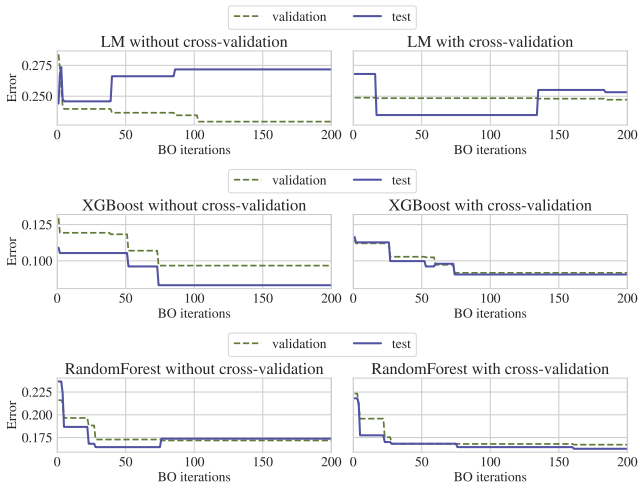

Figure 1: Validation and test errors of the best current hyperparameters. Each plot represents one experiment with 200 BO iterations for LM, XGBoost and RandomForest on the `op100-9952` data with cross-validation (right) and without (left).

3.1 Observations

We now present our observations for BO-based HPO from an overfitting perspective by considering the test error. 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

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

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.

3.1.1 Non-monotonicity of the test error. In Fig. 1, we plot the validation and test errors of the incumbent as we tune LM, XGB and RandomForest algorithms on the op100-9952 dataset 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 for with a minimal point between underfitting and overfitting. When tuning XGB and RandomForest on the same data, 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).

3.1.2 Variance in BO experiments. For the op100-9952 data, we compute the *variances* of validation and test errors at every BO iteration across 100 replicates for LM, XGB and RandomForest in Fig. 2. From Fig. 2, one can see again 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.

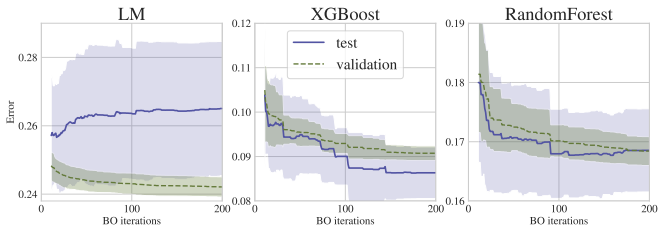

Figure 2: Mean validation and test errors +/− std (y-axis) over 100 experiments with 200 BO iterations (x-axis) for for LM, XGBoost and RandomForest on `op100-9952` dataset with cross-validation. Even with cross-validation, the variance in test performance can be high even in the later stage of HPO.

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.

There are many observations one can make from Fig. 3. First, the variance from BO tends to decrease in both validation and test errors as BO proceeds, and it is the largest source of variance for tuning XGB and RF. The variance from algorithm training is the highest for LM while the lowest for XGB and RF. 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.

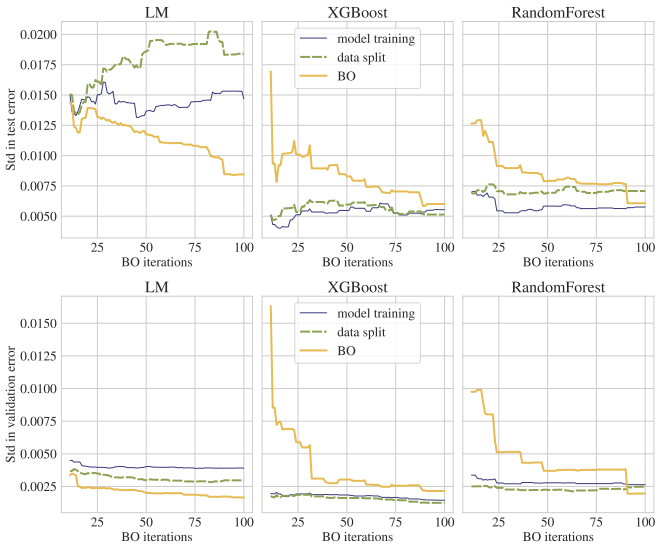

Figure 3: Disentangled sources (model training, data split and BO) of variance in the BO experiments for tuning LM, XGBoost and RandomForest on `op100-9952` dataset. Std of test error and validation error are shown in the top and bottom rows, respectively.

3.1.3 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.

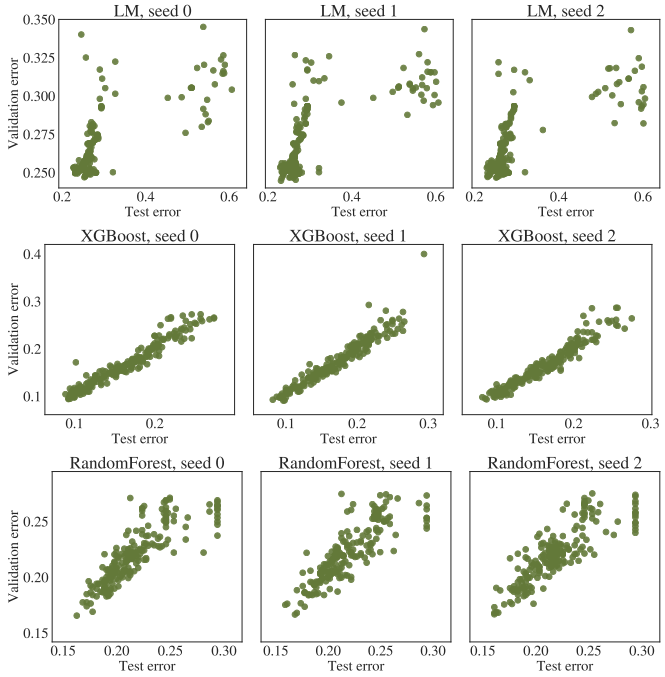

Figure 4: Scatter plot for validation and test errors when tuning LM (1st row), XGB (2nd row) and RandomForest (3rd row) on the `op100-9952` data with cross-validation. Each point represents one hyperparameter evaluation.

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 for the solution quality, and at the same time reduce 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.

4 REGRET BASED STOPPING

In this section, we review the basics of Bayesian Optimization in Section 4.1, and then propose our novel regret-based stopping criterion for BO in Section 4.2, which employs cross-validation.

4.1 Bayesian Optimization

Assume we have a learning algorithm h_γ defined by its hyperparameters $\gamma \in \Gamma$ and parametrised by a weight (parameter) vector $\mathbf{w}$: $h_\gamma(.;\mathbf{w})$. Let $\mathcal{D} = \{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^n$ be the collected dataset of pairs drawn from unknown data distribution $(\mathbf{x}, \mathrm{y}) \sim P_\mathcal{D}$. The goal of HPO is then 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 available data $\mathcal{D}$ is split into $\mathcal{D}_{tr}$ and $\mathcal{D}_{val}$, used for training and validation. One can also use cross-validation and report the average loss across different validation folds. Formally, the bi-level optimization problem over hyperparameters and weights is as follows:

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

BO is an iterative gradient-free optimization methods which, at every step t, selects an input $\gamma_t \in \Gamma$ and observes a noise-perturbed output $y_t \triangleq f(\gamma_t)+\epsilon_t$, where ϵ_t is typically assumed to be i.i.d. (sub)-Gaussian noise with variance (proxy) σ^2. BO algorithms aim to find the global maximizer γ^* by leveraging two components: (i) a probabilistic function model, used to approximate the gradient-free function f, and (ii) an acquisition function which determines the next query. A popular choice for the probabilistic model (or surrogate) is a *Gaussian process (GP)* [23], specified by a mean function $\mu_t : \Gamma \to \mathbb{R}$ and a kernel $k : \Gamma \times \Gamma \to \mathbb{R}$. We assume the objective f is sampled from a GP prior, i.e., $f \sim GP(\mu, k)$, thus, for all $\gamma \in \Gamma$ values are normally distributed, i.e., $f(\gamma) \sim \mathcal{N}(\mu(\gamma), k(\gamma,\gamma'))$. After collecting t data points $\mathfrak{D}_t = \{(\gamma_1, y_1), \ldots, (\gamma_t, y_t)\}$, the GP posterior about value $f(\gamma)$ at a new point γ is defined by posterior mean $\mu_t(\gamma)$ and posterior variance $\sigma_t^2(\gamma)$ as:

$$\mu_t(\gamma) = \mathbf{k}_t(\gamma)^T(\mathbf{K}_t + \sigma_\epsilon\mathbf{I})^{-1}y_t \quad (1)$$
$$\sigma_t^2(\gamma) = k(\gamma,\gamma) - \mathbf{k}_t(\gamma)^T(\mathbf{K}_t + \sigma_\epsilon\mathbf{I})^{-1}\mathbf{k}_t(\gamma), \quad (2)$$

where $\mathbf{K}_t = \{k(\gamma_i,\gamma_j)\}_{i,j=1}^t, \mathbf{k}_t(\gamma) = \{k(\gamma_i,\gamma)\}_{i=1}^t$.

Given a fitted probabilistic model, BO uses an acquisition function to balance the exploration and exploitation tradeoff for suggesting the next hyperparameters. Common choices are probability of improvement (PI) [10], expected improvement (EI) [15], entropy search (ES) [5], predictive entropy search (PES) [6] as well as maximum value entropy search (MES) [29]. We focus on the expected improvement throughout our paper for its simplicity and wide adoption, but our approach is general. Let us denote $f(\gamma_t^*) := \min_{\gamma\in\mathfrak{D}_t} f(\gamma)$ to be the hyperparameters with the minimum loss so far, the EI for a hyperparameter γ can be defined as:

$$\begin{aligned}\mathrm{EI}(\gamma) &= \mathbb{E}[\max(0, \mu_t(\gamma) - f(\gamma_t^*)] \\ &= \sigma_t(\gamma)(v(\gamma)\Phi(v(\gamma)) + \phi(v(\gamma))),\end{aligned}$$

where $v(\gamma) := \frac{\mu_t(\gamma)-f(\gamma_t^*)}{\sigma_t(\gamma)}$, Φ and ϕ denote the CDF and PDF of the standard normal, respectively. In case of noisy observations, the unknown value $f(\gamma_t^*)$ is replaced by the corresponding GP mean estimate [20]. A thorough review of BO can be found in [24].

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 in advance, thus R_T can not be computed in practice.

4.2 Stopping criterion for BO

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.

4.2.1 Upper bound for simple regret. Even though the optimal γ^* is unknown, it is possible to estimate an upper bound for it based on our GP surrogate as shown in [4]. Specifically, we can upper bound the best value found so far $f(\gamma_t^*)$ by

$$f(\gamma_t^*) := \min_{\gamma \in \mathfrak{D}_t} f(\gamma) \leq \min_{\gamma \in \mathfrak{D}_t} ucb(\gamma|\mathfrak{D}_t), \tag{3}$$

where $ucb(\gamma|\mathfrak{D}_t) = \mu_t(\gamma) + \sqrt{\beta_t}\sigma_t(\gamma)$, β_t are appropriate constants for the confidence bound to hold and are studied in [26]. Specifically, we used Theorem 1 in [26] to compute β_t with the modification of using the number of hyperparameters as the size of input domain to accommodate continuous hyperparameters.

Similarly, we can lower bound the true unknown optimum $f(\gamma^*)$ as:

$$f(\gamma^*) \geq \min_{\gamma \in \Gamma} lcb(\gamma|\mathfrak{D}_t) \tag{4}$$

where $lcb(\gamma|\mathfrak{D}_t) = \mu_t(\gamma) - \sqrt{\beta_t}\sigma_t(\gamma)$. Putting together Eqs. (3) and (4), we get:

$$f(\gamma_t^*) - f(\gamma^*) \leq \min_{\gamma \in \mathfrak{D}_t} ucb(\gamma|\mathfrak{D}_t) - \min_{\gamma \in \Gamma} lcb(\gamma|\mathfrak{D}_t) := \hat{R}_t. \tag{5}$$

This upper bound $\hat{R}_t$ is used for BO with unknown search space in [4] to decide when to expand the search space. 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 [4].

4.2.2 Stopping threshold. For small datasets, it is common to use cross-validation to prevent overfitting. Formally, for the K-fold cross-validation, the train-validation dataset is split into K smaller sets and then $\{(\mathcal{D}_{tr}^k, \mathcal{D}_{val}^k)\}_{k=1}^K$ are constructed by iterating over these sets. At each BO iteration, the average loss across different validation splits is then reported. The details can be found in Algorithm 1.

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 [16] and is as follows:

$$s^2 = (\frac{1}{K} + \frac{|\mathcal{D}_{val}|}{|\mathcal{D}_{tr}|})s_{cv}^2, \tag{6}$$

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 in our experiments, thus, the post correction constant on the variance s_{cv}^2 is $\frac{1}{10} + \frac{1}{9} \approx 0.21$. We also empirically validate that with larger K, the variance of cross-validation metrics indeed tends to be higher in Appendix A.2.

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 $\bar{s^2}$ or a specific $s^2(\gamma)$ for some γ. To answer this question, we conducted an ablation study on the correlation between the sample variance in cross-validation and its mean performance in Appendix A.3. We found out that the sample variance in cross-validation is indeed depending 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:

$$\hat{R}_t < s(\gamma_t^*). \tag{7}$$

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. The pseudo code of our method can be found in Algorithm 1.

Algorithm 1 BO with cross-validation and automatic termination

Require: $\{(\mathcal{D}_{tr}^k, \mathcal{D}_{val}^k)\}_{k=1}^K$ for K-fold CV, acq. function $\alpha(\gamma)$
1: Initialize $\mathfrak{D}_0 = \{\}$, $y_t^* = +\infty$
2: **for** $t = 1, 2, \ldots$ **do**
3: Sample $\gamma_t \in \arg\max_{\gamma \in \Gamma} \alpha_t(\gamma)$
4: **for** $k = 1, 2, \ldots, K$ **do**
5: Query output $y_t^k = f(\gamma_t|\mathcal{D}_{tr}^k, \mathcal{D}_{val}^k) + \epsilon_t^k$
6: **end for**
7: Calculate sample mean $y_t = \frac{1}{K}\sum_k y_t^k$
8: **if** $y_t \leq y_t^*$ **then**
9: Update $y_t^* = y_t$ and incumbent $\gamma^* = \gamma_t$
10: Calculate sample variance $s_{cv}^2 = \frac{1}{K}\sum_k (y_t - y_t^k)^2$
11: **end if**
12: Calculate variance estimate s_t^2 for gen. error with Eq. (6)
13: Update $\mathfrak{D}_t \leftarrow \mathfrak{D}_{t-1} \cup \{(\gamma_t, y_t)\}$
14: Update σ_t, μ_t with Eqs. (1) and (2)
15: Calculate upper bound $\hat{R}_t$ for simple regret with Eq. (5)
16: **if** stopping condition $\hat{R}_t \leq s_t$ holds **then**
17: **break for loop**
18: **end if**
19: **end for**
20: **Output:** γ^*

5 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 [12, 19]. 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.

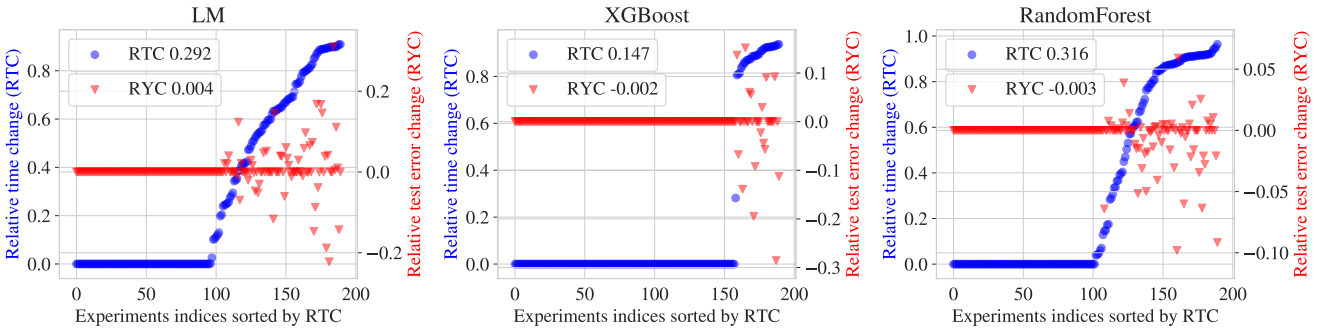

Figure 5: Speed up with our early stopping and test error change given BO budget as 200. Each dot represents an experiment and all the 190 experiments (19 datasets, 10 seeds) are sorted by relative time change, i.e., RTC defined in Eq. (9) (higher is better), on the x-axis. RTC scores for all the experiments are on the *left* y-axis in blue. It can be seen from the plot that early stopping is not triggered for more than 50% experiments as RTC scores are zeros. The relative test error change, RYC defined in Eq. (8), is shown on the *right* y-axis in red. Again, many experiments have RYC scores of zero appeared as a horizontal line because early stopping is not triggered. The average RTC and RYC scores of all experiments are shown in the legend.

Experimental setup. In BO, we optimize classification error or rooted mean square error computed by cross-validation. These errors are positive by definition, and we incorporate this prior knowledge by modelling log transformation of these errors and then adapting the variance, accordingly. We use the number of hyperparameter evaluations as the budget for BO. We report the test performance computed on the fixed test split. We refer the reader to the Appendix A for BO settings (Appendix A.1.1), the detailed hyperparameter search space of the algorithms (Appendix A.1.2), as well as characteristics of the datasets and their splits (Appendix A.1.3). 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 involved into our method is β_t that is set such that confidence bounds in Eqs. (3) and (4) hold with high probability. We use Theorem 1 in [26] to set β_t and further scale it down by a factor of 5 as defined in the experiments in [26], it is then fixed for all the experiments.

Metrics. To measure the effectiveness of a termination criterion, we analyze two metrics, quantifying the change in test error, as well as the time saved. Particularly, given BO budget T, we compare the test error when early stopping is triggered y_{es} to the test error y_T. For each experiment, we compute *relative test error change*, i.e., RYC (we use y to denote the test error), as:

$$\text{RYC} = \frac{y_T - y_{es}}{\max(y_T, y_{es})}. \tag{8}$$

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*, i.e., RTC, is defined as:

$$\text{RTC} = \frac{t_T - t_{es}}{t_T}. \tag{9}$$

A positive RTC, where RTC $\in [0, 1]$, indicates a reduction in total training time.

5.1 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. Each dot in Fig. 5 represents an experiment sorted on the x-axis by RTC score. 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. We present the resulting distributions of RTC and RYC scores in Fig. 6 with violin plots. We choose to use violin plot instead of box plot because the boxes are in many cases not visible due to the clustered scores, while the violin plot clearly reveals the density of the values.

From Fig. 6, it can be seen 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.

5.2 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

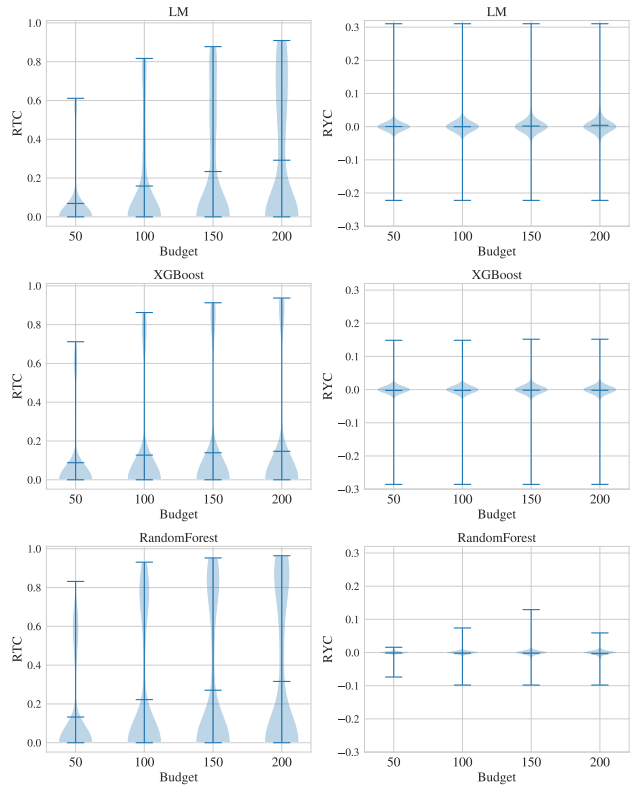

Figure 6: BO with early stopping under different budgets over all datasets and methods. RTC (left) and RYC (right) scores are defined in Eqs. (8) and (9), higher is better. The average score is also shown as a horizontal line in the middle of each violin plot.

nature in BO. Second, defining a threshold is not necessary as the incumbent may stay the same for many iterations and then suddenly change, as shown in Fig. 1.

This convergence condition heavily relies on i, which is chosen in advance. However, the optimal i is different across experiments. We consider values commonly used in practice, in particular, $i = \{10, 30, 50\}$ and BO budget $T = 200$. The results for RYC and RTC distributions are presented in Fig. 7.

A general obvious trend on i illustrated in Fig. 7 is as following: as i increases, the speed up decreases, e.g., the average RTC drops from 80% to 40% as i increases from 10 to 50. However, the solution quality increases as well, and one can see a significant gain in the mean RYC score, except for LM. One can notice distinguishable differences between this convergence baseline and our method: our adaptive stopping condition results not only in the best average RYC score, but also in the smallest variance, which shows that it delivers a more robust solution. Moreover, it sometimes 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 $i = 50$.

At this point, we want to highlight that having a competitive RYC score is a much more challenging task than just gaining speed-up. If one aims at maintaining the solution quality while stopping BO earlier, one needs to take into account the probabilistic model and BO process in a more comprehensive manner.

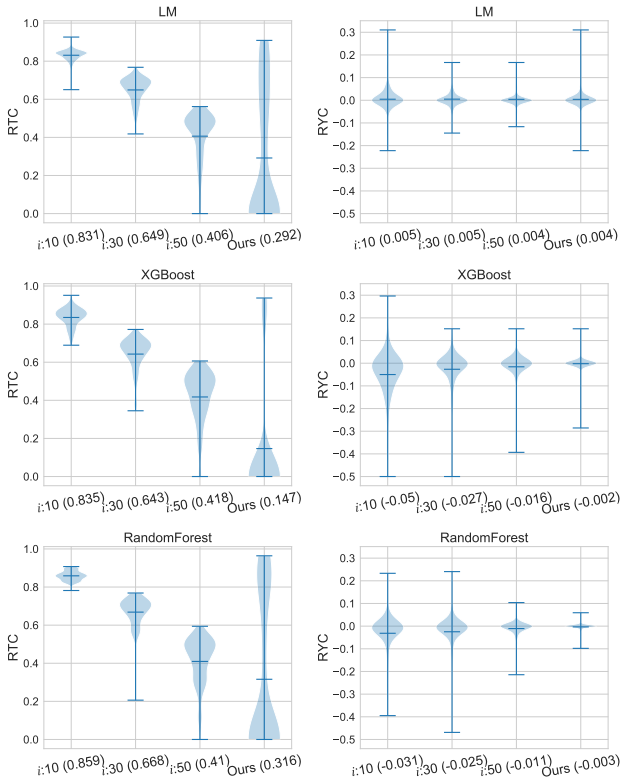

Figure 7: The RTC (left) and RYC (right) scores for our method as well as a convergence check baseline. We stop BO if the best validation error remains the same for more than i iterations, where $i = \{10, 30, 50\}$. The average score is also shown as a horizontal line in the middle of each violin, as well as in the parenthesis next to the method labels on the x-axis.

5.3 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 [19]. 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 [12]. By relying on EI and PI, these stopping criteria inherit their exploration-exploitation trade-off. However, these approaches are not problem adaptive:

they rely on a fixed threshold and do not take into account the variance obtained from cross-validation.

We set the BO budget to 200, and avoid termination during the first 20 iterations. We follow the recommendations from [12, 19] 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.

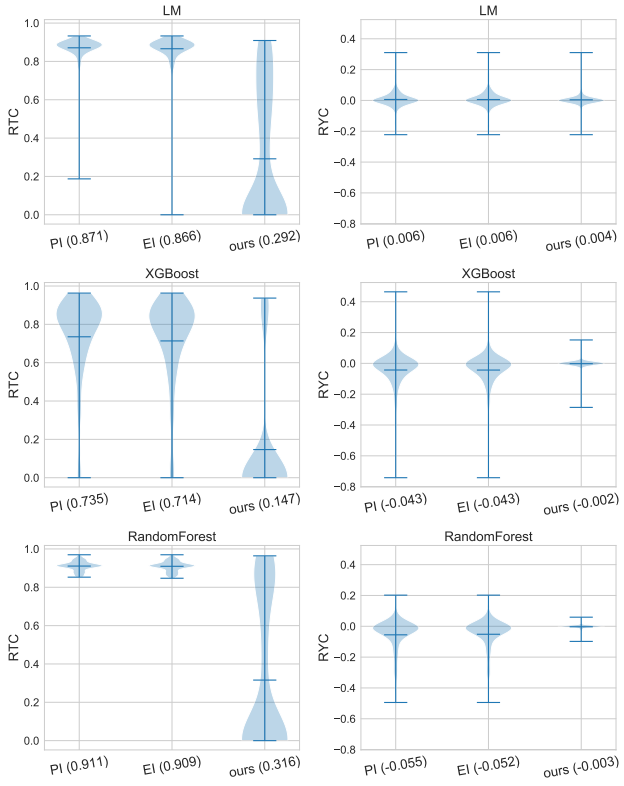

Figure 8: Violin plot for RTC (left) and RYC (right) scores for our method and two baselines based on EI and PI. The average score is also shown as a horizontal line in the middle of each violin, as well as in the parenthesis next to the method labels on the x-axis.

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.

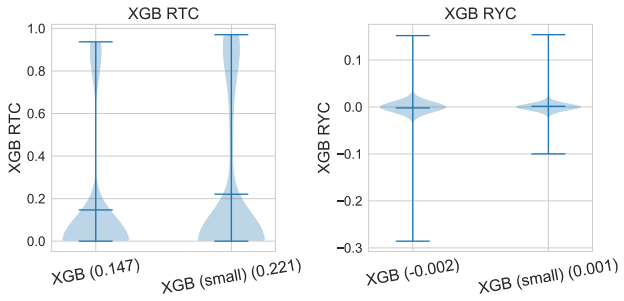

Figure 9: Violin plot for RTC (left) and RYC (right) scores for tuning XGBoost (`XGB`) with 9 HPs and 3 HPs (`XGB (small)`).

6 CONCLUSIONS

This work investigated the problem of overfitting in BO, focusing on the context of tuning the hyperparameters of machine learning models. We proposed a novel stopping criterion based on two theoretically inspired quantities: an upper bound on the suboptimality of the incumbent, and a cross-validation estimate for the variance of generalization performance. These ingredients make the proposed approach problem adaptive, resulting in a method that is very simple to implement, comes with no extra hyperparameters, and is agnostic to the specific BO method. In extensive 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. (7) relies on cross-validation, which can be computationally expensive. As the upper bound on the regret Eq. (5) has a clear interpretation, a promising alternative is to let users specify a threshold in Eq. (7) even without cross-validation.

REFERENCES

[1] Michael Altschuler and Michael Bloodgood. 2019. Stopping Active Learning Based on Predicted Change of F Measure for Text Classification. *2019 IEEE 13th International Conference on Semantic Computing (ICSC)* (Jan 2019). https://doi.org/10.1109/icosc.2019.8665646

[2] Yutian Chen, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver, and Nando de Freitas. 2018. Bayesian Optimization in AlphaGo. arXiv:1812.06855 [cs.LG]

[3] Zhongxiang Dai, H. Yu, K. H. Low, and Patrick Jaillet. 2019. Bayesian Optimization Meets Bayesian Optimal Stopping. In *ICML*.

[4] Huong Ha, Santu Rana, Sunil Gupta, Thanh Nguyen, Hung Tran-The, and Svetha Venkatesh. 2019. Bayesian Optimization with Unknown Search Space. In *Advances in Neural Information Processing Systems 32 (NIPS)*. Curran Associates, Inc., 11795–11804.

[5] Philipp Hennig and Christian J Schuler. 2012. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research* 98888, 1 (2012), 1809–1837.

[6] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. 2014. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in neural information processing systems (NeurIPS)*. 918–926.

[7] Hideaki Ishibashi and Hideitsu Hino. 2020. Stopping criterion for active learning based on deterministic generalization bounds. arXiv:2005.07402 [stat.ML]

[8] Johannes Kirschner, Ilija Bogunovic, Stefanie Jegelka, and Andreas Krause. 2020. Distributionally Robust Bayesian Optimization.

[9] A. Klein, Stefan Falkner, Jost Tobias Springenberg, and F. Hutter. 2017. Learning Curve Prediction with Bayesian Neural Networks. In *ICLR*.

[10] Harold J Kushner. 1963. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. In *Joint Automatic Control Conference*. 69–79.

[11] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. 2020. Gradient Descent with Early Stopping is Provably Robust to Label Noise for Overparameterized Neural Networks. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 108)*, Silvia Chiappa and Roberto Calandra (Eds.). PMLR, 4313–4324.

[12] Romy Lorenz, Ricardo P Monti, Ines R Violante, Aldo A Faisal, Christoforos Anagnostopoulos, Robert Leech, and Giovanni Montana. 2016. Stopping criteria for boosting automatic experimental design using real-time fMRI with Bayesian optimization. arXiv:1511.07827 [q-bio.NC]

[13] Mark McLeod, Stephen Roberts, and Michael A. Osborne. 2018. Optimization, fast and slow: optimally switching between local and Bayesian optimization. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholmsmässan, Stockholm Sweden, 3443–3452.

[14] Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the State of the Art of Evaluation in Neural Language Models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ByJHuTgA-

[15] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. 1978. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization* 2, 117-129 (1978), 2.

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

[17] Thanh Dai Nguyen, Sunil Gupta, Santu Rana, and Svetha Venkatesh. 2017. Stable Bayesian Optimization. In *Advances in Knowledge Discovery and Data Mining*, Jinho Kim, Kyuseok Shim, Longbing Cao, Jae-Gil Lee, Xuemin Lin, and Yang-Sae Moon (Eds.).

[18] Thanh Tang Nguyen, Sunil Gupta, Huong Ha, Santu Rana, and Svetha Venkatesh. 2020. Distributionally Robust Bayesian Quadrature Optimization.

[19] Vu Nguyen, Sunil Gupta, Santu Rana, Cheng Li, and Svetha Venkatesh. 2017. Regret for Expected Improvement over the Best-Observed Value and Stopping Condition.

[20] Victor Picheny, David Ginsbourger, Yann Richet, and Grégory Caplin. 2013. Quantile-based optimization of noisy computer experiments with tunable precision. *Technometrics* 55, 1 (2013), 2–13. https://doi.org/10.1080/00401706.2012.707580

[21] Lutz Prechelt. 1996. Early Stopping-But When? In *Neural Networks: Tricks of the Trade*, Genevieve B. Orr and Klaus-Robert Müller (Eds.). Lecture Notes in Computer Science, Vol. 1524. Springer, 55–69.

[22] Garvesh Raskutti, Martin J. Wainwright, and Bin Yu. 2014. Early Stopping and Non-Parametric Regression: An Optimal Data-Dependent Stopping Rule. *J. Mach. Learn. Res.* 15, 1 (Jan. 2014), 335–366.

[23] CE. Rasmussen and CKI. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, USA. 248 pages.

[24] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. 2016. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104, 1 (2016), 148–175.

[25] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2* (Lake Tahoe, Nevada) *(NIPS'12)*. 2951–2959.

[26] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. 2010. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning* (Haifa, Israel) *(ICML'10)*. Omnipress, Madison, WI, USA, 1015–1022.

[27] Kevin Swersky, Jasper Snoek, and R. Adams. 2014. Freeze-Thaw Bayesian Optimization. *ArXiv* abs/1406.3896 (2014).

[28] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2014. OpenML. *ACM SIGKDD Explorations Newsletter* 15, 2 (Jun 2014), 49–60. https://doi.org/10.1145/2641190.2641198

[29] Zi Wang and Stefanie Jegelka. 2017. Max-value entropy search for efficient Bayesian optimization. *34th International Conference on Machine Learning, ICML 2017* 7, NeurIPS (2017), 5530–5543. arXiv:1703.01968

A APPENDIX

A.1 Experiments setting

A.1.1 BO setting. We used an internal BO implementation where expected improvement (EI) together with Matʿern-52 kernel in the GP are used. The hyperparameters of the GP includes output noise, a scalar mean value, bandwidths for every input dimension, 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 searcher [3].

We tested two methods to learn the GP hyperparameters in our experiments: either maximizing type II likelihood or using slice sampling to draw posterior samples of the hyperparameters. In the later case, we use average across hyperparameters samples (in our experiments we always use 10 samples) to compute EI and predictions. For Slice sampling, we used 1 chain where we draw 300 samples with 250 as burin and 5 as thinning. We also fixed max step in and step out to 200 and the scale parameter is fixed to 1.

We found out that using slice sampling for learning GP hyperparameters is more robust for model fitting than using maximum likelihood estimates. This is especially important for our baselines [12, 19] when using maximum likelihood. In that setting, the EI and PI values can have very small values (10^{-50} to 10^{-200}) due to a bad model fit, triggering stopping signal much earlier than it should be. As a result, we only report experimental results using slice sampling throughout our paper.

A.1.2 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.

Table 1: Search spaces description for each algorithm.

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

[2] https://github.com/SheffieldML/GPyOpt
[3] https://github.com/awslabs/autogluon

A.1.3 Dataset. We list the datasets that are used in our experiments, as well as their characteristics and sources in Table 2. For each dataset, we first randomly draw 20% as test set and for the rest, we use 10-fold cross validations for regression datasets and 10-fold stratified cross validation for classification datasets. 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.

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.

A.2 K-fold cross-validation and its variance

We study the variance of K-fold cross-validation and its relation to the choice of K. We select 50 hyperparameters (first 50 from BO) and allow cross-validation to reshuffle so that we could have 10 replicates for every choice of $K = \{3, 5, 10\}$. For every hyperparameters configuration, we first compute the standard deviation (std) of cross-validation metrics for every replicate and then take the average. The resulting plots on 2 datasets and 3 algorithms are shown in Fig. 10. It seems that with higher K, the standard deviation of the cross-validation metrics tends to be larger.

A.3 Heteroscedastic cross-validation variances

We study the variances of cross validation metrics and its relation to the hyperparameter configurations through hyperparameter evaluations collected in our BO experiments (without early stopping) on 6 example datasets. In Figure 11, the validation error and standard deviation for the hyperparameters are shown in the x-axis and y-axis, respectively. The Pearson correlation coefficients for all the datasets are shown in the legend next to the dataset names. The average correlation coefficients for an algorithm is also shown in the title next to the algorithm name.

From 11, it is clear that the variances of cross validation metrics depends on the hyperparameter configurations, they are mostly positively correlated (in a few cases negatively correlated). For the same dataset, the correlation between the two can change significantly depending on the algorithm being used.

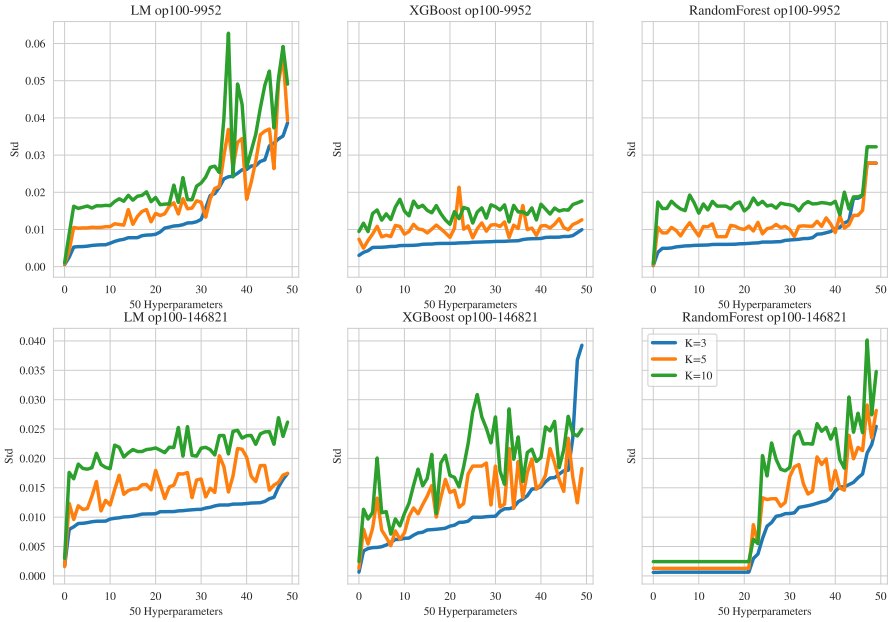

Figure 10: Standard deviation of K-fold cross-validation for $K = \{3, 5, 10\}$ on a set of 50 hyperparameters (sorted by standard deviation for $K = 3$.)

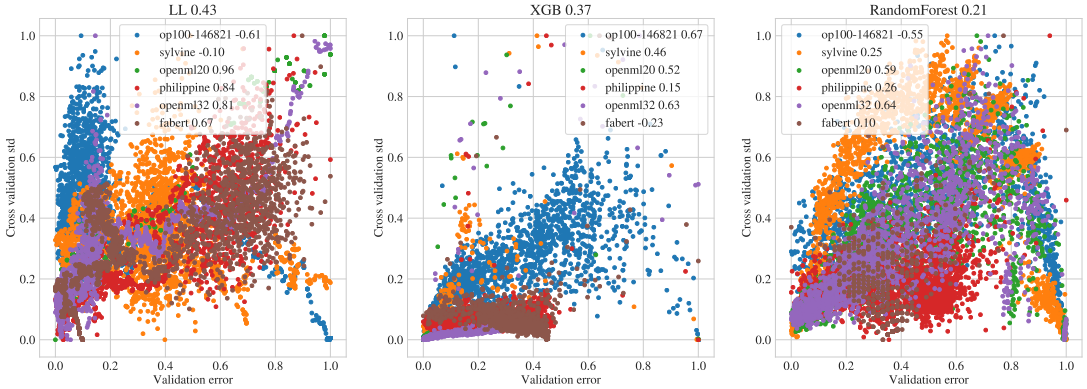

Figure 11: Scatter plot of validation error (x-axis) and standard deviation of cross validation metrics (y-axis) for the same hyperparameter configuration. Every hyperparameter is one dot. The Pearson correlation coefficients for all the datasets are shown in the legend next to the dataset names. The average correlation coefficients for an algorithm is also shown in the title next to the algorithm name.