Statistica Sinica Preprint No: SS-2023-0310	
Title	Bayesian Optimization with Pareto-Principled Training for Economical Hyperparameter Optimization
Manuscript ID	SS-2023-0310
URL	http://www.stat.sinica.edu.tw/statistica/
DOI	10.5705/ss.202023.0310
Complete List of Authors	Yang Yang, Ke Deng and Yu Zhu
Corresponding Authors	Ke Deng
E-mails	kdeng@tsinghua.edu.cn

Statistica Sinica

Bayesian Optimization with Pareto-Principled Training for Economical Hyperparameter Optimization

Yang Yang[1], Ke Deng[2] and Yu Zhu[3]

[1]*Nankai University,* [2]*Tsinghua University and* [3]*Purdue University*

Abstract: The specification of hyperparameters plays a critical role in determining the practical performance of a machine learning method. Hyperparameter Optimization (HPO), i.e., the searching for optimal specification of hyperparameters, however, often faces critical computational challenges due to the vast searching space and the high computational cost on model training under a given hyperparameter specification. In this paper, we propose BOPT-HPO, a systematic approach for efficient HPO by leveraging Bayesian optimization with Pareto-principled training, based on the observation that the training procedure of a machine learning method under a given hyperparameter specification often follows the Pareto principle (the 80/20 rule) that about 80% of the total improvement in the objective function is achieved in 20% of the training time. By introducing two levels of training corresponding to the Pareto principle, i.e., the eighty-percent training (ET) and the complete training (CT), and establishing a joint surrogate model for CT runs and ET runs, BOPT-HPO reduces the computational cost of HPO significantly under the framework of Bayesian optimization with multi-fidelity measurements. A wide range of experimental studies confirm that the proposed approach achieves economical HPO for various machine learning models, including support vector machines, feed-forward neural networks, and convolutional neural networks.

Corresponding authors: Ke Deng, Department of Statistics and Data Science, Tsinghua University, Beijing 100084, China. E-mail: kdeng@tsinghua.edu.cn. Yu Zhu, Department of Statistics, Purdue University, West Lafayette, IN 47907, USA. E-mail: yuzhu@purdue.edu.

Key words and phrases: Automated artificial intelligence; Black-box function optimization; Computer experiments; Multi-fidelity modelling; Truncated Gaussian process.

1. Introduction

Machine Learning (ML) models, such as Support Vector Machines (SVMs) and Deep Neural Networks (DNNs), have become a driving force for modern data-science technologies, leading to applications not only in artificial intelligence, such as computer vision (Krizhevsky et al., 2012) and natural language processing (Goldberg, 2017), but also in intelligent manufacturing, such as material design (Batra et al., 2021) and 3D printing (Zhu et al., 2021).

In addition to the usual model parameters, a ML model typically involves hyperparameters that define the model's structure (e.g., the number of hidden layers of a DNN model) or control the learning process (e.g., the learning rate). Let $\mathcal{M}$ denote an ML model with d hyperparameters. A specific setting of the d hyperparameters is referred to as a *hyperparameter configuration*. The collection of all possible hyperparameter configurations is called the hyperparameter configuration space and denoted as $\mathcal{X} \subseteq \mathbb{R}^d$. Given a hyperparameter configuration $\mathbf{x} \in \mathcal{X}$, a setting of the usual model parameters is denoted as $\boldsymbol{\psi}_{\mathbf{x}} \in \mathbb{R}^{d_{\mathbf{x}}}$, where $d_{\mathbf{x}}$ represents for the dimensionality of $\boldsymbol{\psi}_{\mathbf{x}}$. Let $\Psi_{\mathbf{x}}$ denote the collection or space of all possible settings of the model parameters.

The performance measurement (e.g., the validation loss) of $\mathcal{M}$, which is denoted as y in this article, depends on both its hyperparameter configuration $\mathbf{x}$ and its model parameter setting $\boldsymbol{\psi}_{\mathbf{x}}$ under $\mathbf{x}$, and can be expressed as $y(\mathbf{x}, \boldsymbol{\psi}_{\mathbf{x}})$. Under a particular hyperparameter configuration $\mathbf{x}$, we search the model parameter space $\Psi_{\mathbf{x}}$ for the optimal parameter setting $\boldsymbol{\psi}^*_{\mathbf{x}} = \arg\min_{\boldsymbol{\psi}_{\mathbf{x}} \in \Psi_{\mathbf{x}}} y(\mathbf{x}, \boldsymbol{\psi}_{\mathbf{x}})$, leading to $y(\mathbf{x}) = y(\mathbf{x}, \boldsymbol{\psi}^*_{\mathbf{x}})$, which is the best performance measurement of $\mathcal{M}$ under hyperparameter configuration $\mathbf{x}$. Apparently, $y(\mathbf{x})$ plays a crucial role in evaluating the effectiveness of $\mathbf{x}$ and serves as the objective function to guide *HyperParameter Optimization* (HPO), which is aimed to search $\mathcal{X}$ for the best hyperparameter configuration denoted as $\mathbf{x}^*$, that is, to find $\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathcal{X}} y(\mathbf{x})$.

Because the functional form of $y(\mathbf{x})$ is typically unknown, HPO is essentially a black-box function optimization problem, which faces critical challenges in most ML tasks. First, the extra high dimensionality of the model parameter setting $\boldsymbol{\psi}_{\mathbf{x}}$ makes it computationally expensive to find the optimal setting $\boldsymbol{\psi}^*_{\mathbf{x}}$ for each given $\mathbf{x} \in \mathcal{X}$, which further makes the evaluation of $y(\mathbf{x})$ a costly operation. Second, the complexity (e.g., size and dimensionality) of the hyperparameter configuration space $\mathcal{X}$ poses further difficulties in searching for the optimal configuration $\mathbf{x}^*$ in practice. Due to these challenges in HPO, practitioners were forced to either rely on the brute-force searching approach (Hinton, 2012; LeCun et al., 2012), which is feasible only for simple models, or use heuristic approaches (Bergstra et al., 2011; Li et al., 2018), which are unstable and sub-optimal.

There exists an urgent demand of developing a systematic approach to perform efficient and automatic HPO for general ML models and tasks. In this article, we meet the demand by making efforts on two primary directions for improving the efficiency of HPO: finding ways to reduce the computational cost for function evaluation of $y(\mathbf{x})$, and designing smart searching strategies to improve the efficiency to explore the vast configuration space $\mathcal{X}$. Integrating these efforts under the framework of Bayesian optimization, we come up with a highly efficient method called Bayesian Optimization with Pareto-Principled Training for Hyperparameter Optimization (BOPT-HPO) in a wide collection of ML methods, including support vector machines, feed-forward neural networks, and convolutional neural networks.

To reduce the computational cost for evaluating $y(\mathbf{x})$, we propose performing the evaluation with two different levels of fidelity via Pareto-principled training. The key idea comes from the observation that the training procedure of a ML method under a given hyperparameter configuration often follows the Pareto principle (also known as the 80/20 rule, Sanders (1987)): the fast-improving period, during which the objective function improves quickly over time, brings 80% of the total improvement in the objective function in about 20% of the total training time; whereas the slow-improving period, during which the objective function improves at a much slower pace, consumes 80% of the training time to achieve the rest 20% improvement. These facts suggest that we can actually perform two levels of training under a given configuration $\mathbf{x}$ to mitigate the computational burden in HPO: the *complete training* (CT) that keeps optimizing the model parameter $\boldsymbol{\psi}_{\mathbf{x}}$ until a pre-given convergence criterion is met, resulting in well optimized model parameters $\boldsymbol{\psi}_{\mathbf{x}}^{c}$ and the corresponding *accurate performance measurement* $y_c(\mathbf{x}) = y(\mathbf{x}, \boldsymbol{\psi}_{\mathbf{x}}^{c})$; and, the *eighty-percent training* (ET) that is the early-stopped version of CT immediately after the fast-improving period ends, resulting in insufficiently optimized model parameters $\boldsymbol{\psi}_{\mathbf{x}}^{e}$ and the corresponding *approximate performance measurement* $y_e(\mathbf{x}) = y(\mathbf{x}, \boldsymbol{\psi}_{\mathbf{x}}^{e})$. Figure 1 illustrates the idea of Pareto-principled training by highlighting the endpoints of ET and CT in a typical training procedure of a ML method under a given hyperparameter configuration. In practice, it is not necessary to strictly adhere the 80/20 rule to claim the ET stage. Instead, we can identify the end point of ET by detecting the elbow point of the loss curve as shown in Figure 1 via some stopping criteria (Prechelt, 2012).

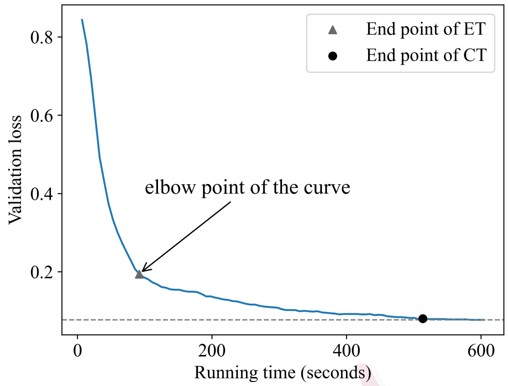


Figure 1: A typical training procedure of a ML method under a given hyperparameter configuration containing an ET and a CT period.

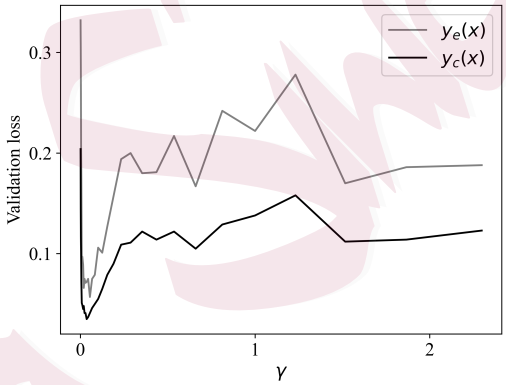


Figure 2: The validation loss of ET runs and CT runs against the kernel hyperparameter γ in a support vector machine model.

To explore the vast configuration space $\mathcal{X}$ efficiently, we rely on *Bayesian optimization* (BO), a classic framework for black-box function optimization, to design efficient exploration strategies. Different from brute-force or heuristic searching, which explores the configuration space $\mathcal{X}$ without clear guidance, BO explores $\mathcal{X}$ much more efficiently via sequential iterations between a fitting stage, which approximates the unknown black-box response function $y(\mathbf{x})$ with a computationally convenient surrogate model based on the available observations, and an exploration stage, which collects more observations under the guidance of the established surrogate model, until a plausible solution is reached (Jones et al., 1998; Shahriari et al., 2016). In the literature, BO-based HPO methods, such as TPE (Bergstra et al., 2011), SMAC

(Hutter et al., 2011), GP-BO (Snoek et al., 2012) and BOHB (Falkner et al., 2018), have demonstrated good potentials on improving HPO efficiency with various implementations of the BO framework. In this work, we will further extend these efforts in a more general setting.

Apparently, for any configuration $\mathbf{x} \in \mathcal{X}$, $y_e(\mathbf{x})$ obtained from an ET run serves as a less expensive approximation of $y_c(\mathbf{x})$ from the corresponding CT run. In an HPO problem, the CT and ET runs result in performance measurements of two levels of fidelity: level-1 fidelity and level-2 fidelity, respectively, leading to a scenario that is very similar to Multi-Fidelity Optimization (MFO) problem in both computer experiments and machine learning. When considering these fidelity levels as distinct information sources, our work also aligns with Multi-Information Source Optimization (MISO) in machine learning. In machine learning, researchers either utilize only the highest fidelity measurements or information sources to build surrogate model with other fidelity measurements completely ignored (Falkner et al., 2018), or model the multi-fidelity measurements separately or implicitly via the Multi-outputs Gaussian Process (MGP) (Kandasamy et al., 2016; Poloczek et al., 2017; Candelieri and Archetti, 2021). While in computer experiments, relationships between successive fidelity measurements are often hierarchical and modeled using some linkage functions (Kennedy and O'Hagan, 2000; Picheny et al., 2013; Gramacy, 2020), indicating that a group of low-fidelity measurements can often be calibrated by a small number of judiciously chosen high-fidelity measurements. This suggests a more computationally efficient strategy can be established for HPO by integrating observations from both CT and ET runs.

The HPO problem with ET and CT runs forms a nested structure, which is a common design in computer experiments. However, our work has some unique features that usually do not appear in computer experiments. First, an ET run must be completed before a CT run, implying that an ET run is a sub-process of the corresponding CT run in terms of performance measurement collection. This naturally creates a nested structure between the explored configuration sets at the two fidelity levels, i.e., $\mathcal{X}_c \subseteq \mathcal{X}_e$, where $\mathcal{X}_e$ and $\mathcal{X}_c$ represent the configurations explored by ET and CT runs, respectively. Second, the performance measurement (e.g., the validation loss) from a CT run is typically smaller than the measurement from the corresponding ET run, leading to the constraint $y_c(\mathbf{x}) \leq y_e(\mathbf{x})$ for most $\mathbf{x} \in \mathcal{X}$. Third, the landscape of $y_c(\mathbf{x})$ can still be well characterized by the landscape of $y_e(\mathbf{x})$, even though $y_e(\mathbf{x})$ is a biased approximation of $y_c(\mathbf{x})$. Figure 2 visualizes the landscapes of $y_c(\mathbf{x})$ and $y_e(\mathbf{x})$ under different specifications of the kernel hyperparameter γ in a support vector machine, illustrating these features intuitively.

The existence of two different types of training runs, i.e., CT and ET runs, gives us more flexibility, but also introduces more complexity, in response surface fitting and exploration. Moreover, the nested structure between $\mathcal{X}_c$ and $\mathcal{X}_e$ and extra constraint between $y_c(\mathbf{x})$ and $y_e(\mathbf{x})$ make it non-trivial to establish an appropriate BO approach for this complicated setting. This study fills in these gaps with the BOPT-HPO method, which establishes a joint surrogate model for CT and ET runs at the fitting stage with full consideration of the involved constraints, and explores the vast hyperparameter space efficiently under the guidance of carefully designed acquisition functions for both CT and ET runs at the exploration stage. A wide range of experimental studies demonstrate the superiority of BOPT-HPO as an ideal solution to the

Statistica Sinica: Preprint
doi:10.5705/ss.202023.0310

HPO problem for ML methods.

The rest of the article is organized as follows. In Section 2, we briefly review the Bayesian optimization. And then we propose a linkage model to systematically integrate the ET and CT runs in Section 3. The BOPT-HPO method is further developed in Section 4. Section 5 illustrates the performances of the proposed method through experiments involving several synthetic functions and a number of different machine learning algorithms. Finally, we conclude the article with discussions in Section 6.

2. A Review for Black-Box Function Optimization via BO

In this section, we will briefly review the classic BO for black-box function optimization. Tracing back to Kushner (1964), BO is a classic approach for black-box function optimization, which fits and explores the unknown response surface in turn.

Given a group of observations $\mathcal{O} = \{(\mathbf{x}_i, y(\mathbf{x}_i))\}_{i=1}^{t}$, where $\{\mathbf{x}_i\}_{i=1}^{t}$ is the set of pre-explored configurations, BO approximates the unknown function $y(\mathbf{x})$ with a surrogate model $\hat{y}(\mathbf{x})$ based on $\mathcal{O}$ in the fitting stage. A popular surrogate model for $y(\mathbf{x})$ is the Gaussian process (GP): $y(\mathbf{x}) \sim \mathrm{GP}\big(\mu, \sigma^2 R_{\boldsymbol{\phi}}(\cdot,\cdot)\big)$, where μ is a constant mean function, σ^2 is the variance, and the correlation function $R_{\boldsymbol{\phi}}(\cdot,\cdot)$ is often specified to be the Gaussian kernel $R_{\boldsymbol{\phi}}(\mathbf{x},\mathbf{x}') = \prod_{m=1}^{d} \exp\left(-\phi_m (x_m - x'_m)^2\right)$ with $\boldsymbol{\phi} = (\phi_1, ..., \phi_d)$ as the length-scale parameters (Rasmussen, 2004). Let $\hat{\boldsymbol{\theta}} = (\hat{\mu}, \hat{\sigma}^2, \hat{\boldsymbol{\phi}})$ be the maximum likelihood estimates (MLE) of the GP parameters $\boldsymbol{\theta} = (\mu, \sigma^2, \boldsymbol{\phi})$. Direct application of the Bayes rule leads to the result that the posterior distribution of $y(\mathbf{x})$ given $\mathcal{O}$ and $\hat{\boldsymbol{\theta}}$ is still a Gaussian distribution for any $\mathbf{x} \in \mathcal{X}$:

$$y(\mathbf{x}) \mid \mathcal{O}, \hat{\boldsymbol{\theta}} \;\sim\; \mathcal{N}\big(\hat{y}(\mathbf{x}), \hat{s}^2(\mathbf{x})\big),$$
$$\hat{y}(\mathbf{x}) = \hat{\mu} + \hat{\mathbf{r}}^T\hat{\mathbf{R}}^{-1}(\mathbf{y} - \hat{\mu}\cdot\mathbf{1}_t), \qquad \hat{s}^2(\mathbf{x}) = \hat{\sigma}^2(1 - \hat{\mathbf{r}}^T\hat{\mathbf{R}}^{-1}\hat{\mathbf{r}}), \tag{2.1}$$

with $\hat{\mathbf{R}} = \{R_{\hat{\boldsymbol{\phi}}}(\mathbf{x}_i, \mathbf{x}_j)\}_{1\le i,j\le t}$ being the estimated correlation matrix of $\mathbf{y} = (y(\mathbf{x}_1), \ldots, y(\mathbf{x}_t))^T$, $\hat{\mathbf{r}}$ the correlation vector between $y(\mathbf{x})$ and $\mathbf{y}$, and $\mathbf{1}_t$ the t-dimensional column vector whose elements are all equal to 1. Apparently, the predictive distribution $\mathcal{N}\big(\hat{y}(\mathbf{x}), \hat{s}^2(\mathbf{x})\big)$ serves as a convenient surrogate model of $y(\mathbf{x})$ at the t-th iteration of the BO procedure.

In the exploration stage, BO searches the configuration space $\mathcal{X}$ for a new configuration $\mathbf{x}_{t+1}$ to explore by maximizing the *acquisition functions* based on the surrogate model (2.1). A variety of acquisition functions have been proposed, such as the Probability of Improvement (PI) (Jones, 2001), Expected Improvement (EI) (Jones, 2001; Yang et al., 2021), and Upper Confidence Bound (UCB) (Srinivas et al., 2010). Particularly, the UCB acquisition function $\varphi(\mathbf{x}) = -\hat{y}(\mathbf{x}) + \sqrt{\beta_t}\cdot\hat{s}(\mathbf{x})$, balancing exploration and exploitation, has been widely used in practice, with β_t suggested to be specified to $0.2d\log(2t)$ (Kandasamy, 2018). Given the acquisition function $\varphi(\mathbf{x})$, BO seeks the next configuration $\mathbf{x}_{t+1}$ to explore by solving the optimization problem: $\mathbf{x}_{t+1} = \mathrm{argmax}_{\mathbf{x}\in\mathcal{X}}\,\varphi(\mathbf{x})$, and update the observation set $\mathcal{O}$ by $\mathcal{O} = \mathcal{O}\cup\big\{(\mathbf{x}_{t+1}, y(\mathbf{x}_{t+1}))\big\}$ accordingly. The iteration stops till a plausible configuration to solve the BO optimization problem is reached.

3. Joint Surrogate Model for CT and ET Runs

Before presenting the BOPT-HPO method, we first briefly review the commonly used baseline surrogate model for multi-fidelity responses in literature in Section 3.1. Following that, we propose an improved surrogate model GP-TGP for CT and ET runs in Sections 3.2-3.4 to consider the additional constraint between CT and ET runs, which is not encountered in classic multi-fidelity computer experiments. This significantly improves the statistical efficiency of the fitting stage of BOPT-HPO, and such an extension is one of the major contributions of this study.

3.1 The Baseline Model

Treating $y_c(\mathbf{x})$ and $y_e(\mathbf{x})$ as two responses at configuration $\mathbf{x}$ with different levels of fidelity, much effort has been made to model them jointly in the literature of computer experiments (Qian et al., 2006; Huang et al., 2006; Kuya et al., 2011; Goh et al., 2013; Le Gratiet and Cannamela, 2015). The earliest work in this area traced back to the *auto-regressive model* proposed by Kennedy and O'Hagan (2000), which models the responses at the lower fidelity level $y_{k-1}(\mathbf{x})$ and the higher fidelity level $y_k(\mathbf{x})$ as follows.

$$\begin{cases} y_1(\mathbf{x}) = & \delta_1(\mathbf{x}), \\ y_k(\mathbf{x}) = & \rho_{k-1} \cdot y_{k-1}(\mathbf{x}) + \delta_k(\mathbf{x}), \quad k = 2, \dots, K, \\ \delta_k(\mathbf{x}) \sim & \mathrm{GP}\big(\mu_k, \sigma_k^2 R_{\phi_k}(\cdot,\cdot)\big), \quad k = 1, \dots, K, \end{cases} \tag{3.1}$$

where K is the number of fidelity levels, $\{\rho_k\}_{k=1}^{K-1}$ are unknown regression coefficients, and $\{\delta_k(\mathbf{x})\}_{k=1}^{K}$ are K independent GPs. Picheny et al. (2013), Tuo et al. (2014) and Stroh et al. (2022) further extended the discrete fidelity levels in (3.1) to continuous fidelity levels.

When only two fidelity levels of responses are considered, i.e., $K = 2$, the auto-regressive model (3.1) degenerates to the Double-GP (DGP) model (Forrester et al., 2007) below.

$$\begin{cases} y_c(\mathbf{x}) = & \rho \cdot y_e(\mathbf{x}) + \delta(\mathbf{x}), \\ y_e(\mathbf{x}) \sim & \mathrm{GP}\big(\mu_e, \sigma_e^2 R_{\boldsymbol{\phi}_e}(\cdot,\cdot)\big), \\ \delta(\mathbf{x}) \sim & \mathrm{GP}\big(\mu_\delta, \sigma_\delta^2 R_{\boldsymbol{\phi}_\delta}(\cdot,\cdot)\big), \end{cases} \tag{3.2}$$

where the unknown parameters $(\mu_e, \sigma_e^2, \boldsymbol{\phi}_e)$ involved in $y_e(\mathbf{x})$ and $(\mu_\delta, \sigma_\delta^2, \boldsymbol{\phi}_\delta)$ involved in $\delta(\mathbf{x})$ are the means, variances and parameters of the correlation functions, respectively. Qian and Wu (2008) generalized the above DGP model by replacing the constant regression coefficient ρ with a stochastic process $\rho(\mathbf{x})$ modeled by an additional GP, and adding a random measurement error to $y_c(\mathbf{x})$. In this study, we take the basic model in (3.2) as the baseline surrogate model.

3.2 The Improved Model

The baseline surrogate model in (3.2) and its extensions do not consider the additional constraint between $y_c(\mathbf{x})$ and $y_e(\mathbf{x})$. In this study, we fill in this gap by replacing the second GP for the bias adjustment function $\delta(\mathbf{x})$ in the DGP model by a Truncated GP (TGP), resulting in a GP-TGP model defined below.

$$\begin{cases} y_c(\mathbf{x}) = & \rho \cdot y_e(\mathbf{x}) + \delta(\mathbf{x}), \\ y_e(\mathbf{x}) \sim & \mathrm{GP}\big(\mu_e, \sigma_e^2 R_{\boldsymbol{\phi}_e}(\cdot,\cdot)\big), \\ \delta(\mathbf{x}) \sim & \mathrm{TGP}\big(\mu_\delta, \sigma_\delta^2 R_{\boldsymbol{\phi}_\delta}(\cdot,\cdot);\ b_1, b_2\big), \end{cases} \tag{3.3}$$

where b_1 and b_2 are pre-determined lower and upper bounds of the TGP satisfying $b_1 \leq b_2$. Please note that the specification of (b_1, b_2) is straightforward in most ML tasks. For example, if $y_e(\mathbf{x})$ and $y_c(\mathbf{x})$ are classification errors of a ML algorithm for a classification task, we would naturally have $(b_1, b_2) = (-1, 0)$. Given (b_1, b_2), the regression coefficient ρ can be manually specified or precisely estimated according to the GP-TGP model. If we are confident that $y_e(\mathbf{x})$ and $y_c(\mathbf{x})$ are at the same scale a priori, we can simply specify $\rho = 1$. Otherwise, we can treat ρ as a free parameter for a more flexible GP-TGP model. Mean functions μ_e and μ_δ can be specified based on prior knowledge or in a data-driven fashion. In case that there are insufficient prior knowledge and data available to specify a nuanced trend, we can use constant mean functions for simplicity. Our experiments show that this simple strategy works well in most cases.

3.3 Parameter Estimation of the Improved Model

Assume that m ET runs and n CT runs have been implemented at configuration sets $\mathcal{X}_e = \{\mathbf{x}_1^e, \ldots, \mathbf{x}_m^e\}$ and $\mathcal{X}_c = \{\mathbf{x}_1^c, \ldots, \mathbf{x}_n^c\}$, respectively, resulting in a group of approximate performance measurements $\mathbf{y}_e = (y_e(\mathbf{x}_1^e), \ldots, y_e(\mathbf{x}_m^e))^T$ and a group of accurate performance measurements $\mathbf{y}_c = (y_c(\mathbf{x}_1^c), \ldots, y_c(\mathbf{x}_n^c))^T$. Note that in most machine learning algorithms, an accurate performance measurement is always reported after the corresponding approximate performance measurement due to the iterative nature of the training process, so the two configuration sets $\mathcal{X}_c$ and $\mathcal{X}_e$ are always nested in this study in terms of $\mathcal{X}_c \subseteq \mathcal{X}_e$.

For simplicity, denote the correlation matrices of $\mathbf{y}_e$ and $\mathbf{y}_c$, i.e., $\mathbf{R}_{\boldsymbol{\phi}_e}$ and $\mathbf{R}_{\boldsymbol{\phi}_\delta}$, as $\mathbf{R}_e$ and $\mathbf{R}_\delta$, respectively. Based on the definition of TGP, the n-dimensional random vector $\boldsymbol{\delta}_n = (\delta(\mathbf{x}_1^c), \ldots, \delta(\mathbf{x}_n^c))^T$ follows a multivariate truncated normal (TN) distribution, that is, $\boldsymbol{\delta}_n \sim \mathcal{TN}_n(\mu_\delta \mathbf{1}_n, \sigma_\delta^2 \mathbf{R}_\delta;\ b_1\mathbf{1}_n, b_2\mathbf{1}_n)$, with $\mathbf{R}_\delta$ being the $n \times n$ correlation matrix, $b_1\mathbf{1}_n$ and $b_2\mathbf{1}_n$ being the lower and upper bounds of $\boldsymbol{\delta}_n$, respectively. Given the model parameters $\boldsymbol{\psi} = (\rho, \mu_e, \mu_\delta, \sigma_e^2, \sigma_\delta^2, \boldsymbol{\phi}_e, \boldsymbol{\phi}_\delta)$ and the pre-specified constraints b_1 and b_2, the conditional distribution of $\mathbf{y}_c$ given $(\mathbf{y}_e, \boldsymbol{\psi}, b_1, b_2)$ is a multivariate truncated normal distribution: $\mathbf{y}_c |\ \mathbf{y}_e, \boldsymbol{\psi}, b_1, b_2 \sim \mathcal{TN}_n(\rho \mathbf{y}_{e_c} + \mu_\delta \mathbf{1}_n,\ \sigma_\delta^2 \mathbf{R}_\delta;\ \rho \mathbf{y}_{e_c} + b_1 \mathbf{1}_n,\ \rho \mathbf{y}_{e_c} + b_2 \mathbf{1}_n)$, where $\mathbf{y}_{e_c} = (y_e(\mathbf{x}_1^c), \ldots, y_e(\mathbf{x}_n^c))^T$. The log-likelihood of $\boldsymbol{\psi}$ given $(\mathbf{y}_c, \mathbf{y}_e)$ is

$$\begin{aligned} l(\boldsymbol{\psi}) &= \log\left(p(\mathbf{y}_c|\ \mathbf{y}_e, \boldsymbol{\psi}, b_1, b_2) p(\mathbf{y}_e|\ \boldsymbol{\psi}, b_1, b_2)\right) \\ &\propto -\frac{1}{2\sigma_\delta^2}(\mathbf{y}_c - \rho\mathbf{y}_{e_c} - \mu_\delta \mathbf{1}_n)^T \mathbf{R}_\delta^{-1}(\mathbf{y}_c - \rho\mathbf{y}_{e_c} - \mu_\delta\mathbf{1}_n) - \ln\left(Z_c(\rho; \mu_\delta, \sigma_\delta^2, \boldsymbol{\phi}_\delta; b_1, b_2)\right) \\ &\quad -\frac{m}{2}\ln\sigma_e^2 - \frac{1}{2}\ln|\mathbf{R}_e| - \frac{1}{2\sigma_e^2}(\mathbf{y}_e - \mu_e\mathbf{1}_m)^T\mathbf{R}_e^{-1}(\mathbf{y}_e - \mu_e\mathbf{1}_m), \end{aligned}$$

where

$$Z_c(\rho; \mu_\delta, \sigma_\delta^2, \boldsymbol{\phi}_\delta; b_1, b_2) = \int_{\mathbf{y}_c \in \mathbf{A}_n} \exp\left\{-\frac{1}{2}(\mathbf{y}_c - \rho\mathbf{y}_{e_c} - \mu_\delta\mathbf{1}_n)^T(\sigma_\delta^2\mathbf{R}_\delta)^{-1}(\mathbf{y}_c - \rho\mathbf{y}_{e_c} - \mu_\delta\mathbf{1}_n)\right\} d\mathbf{y}_c,$$

and $\mathbf{A}_n = [\rho y_e(\mathbf{x}_1^c) + b_1, \rho y_e(\mathbf{x}_1^c) + b_2] \times \cdots \times [\rho y_e(\mathbf{x}_n^c) + b_1, \rho y_e(\mathbf{x}_n^c) + b_2]$.

The lemma below shows the MLEs of the parameters in the GP-TGP model can be obtained by dividing the parameters into two parts.

Lemma 1. *The maximization of $l(\boldsymbol{\psi})$ is equivalent to solving the following two separable optimization problems:*

$$\max_{\mu_e,\sigma_e^2,\boldsymbol{\phi}_e} \Big\{ -\frac{m}{2}\ln\sigma_e^2 - \frac{1}{2}\ln|\mathbf{R}_e| - \frac{1}{2\sigma_e^2}(\mathbf{y}_e - \mu_e\mathbf{1}_m)^T\mathbf{R}_e^{-1}(\mathbf{y}_e - \mu_e\mathbf{1}_m)\Big\}, \tag{3.4}$$

$$\max_{\rho,\mu_\delta,\sigma_\delta^2,\boldsymbol{\phi}_\delta} \Big\{ -\frac{1}{2\sigma_\delta^2}(\mathbf{y}_c - \rho\mathbf{y}_{e_c} - \mu_\delta\mathbf{1}_n)^T\mathbf{R}_\delta^{-1}(\mathbf{y}_c - \rho\mathbf{y}_{e_c} - \mu_\delta\mathbf{1}_n) - \ln\big(Z_c(\rho;\mu_\delta,\sigma_\delta^2,\boldsymbol{\phi}_\delta;b_1,b_2)\big)\Big\} \tag{3.5}$$

It is easy to check that (3.4) takes its maximum at $\hat{\mu}_e = \big(\mathbf{1}_m^T\mathbf{R}_e^{-1}\mathbf{y}_e\big)/\big(\mathbf{1}_m^T\mathbf{R}_e^{-1}\mathbf{1}_m\big)$ and $\hat{\sigma}_e^2 = (\mathbf{y}_e - \hat{\mu}_e\mathbf{1}_m)^T\mathbf{R}_e^{-1}(\mathbf{y}_e - \hat{\mu}_e\mathbf{1}_m)/m$. Plugging $\hat{\mu}_e$ and $\hat{\sigma}_e^2$ into (3.4), $\boldsymbol{\phi}_e$ can be optimized by numerical methods, such as Nelder–Mead (Marler and Arora, 2004). The integration $Z_c(\rho;\mu_\delta,\sigma_\delta^2,\boldsymbol{\phi}_\delta;b_1,b_2)$ can be calculated via variable transformation to transform the intertwined multiple integral to an iterated integral which can be approximated by the quasi-randomized Monte Carol method (Genz, 1992). Finally, the maximizer of (3.5) can be obtained by numerical methods. We obtain the MLE of $\boldsymbol{\psi}$ and denote it as $\hat{\boldsymbol{\psi}}$.

3.4 Response Prediction Based on the Improved Model

Next, we consider the prediction of $y_e(\mathbf{x})$ and $y_c(\mathbf{x})$ for any $\mathbf{x} \in \mathcal{X}$ given $\hat{\boldsymbol{\psi}}$. Conditioning on $\mathbf{y}_e$, $y_e(\mathbf{x})$ learns no more information from $\mathbf{y}_c$, resulting in the equivalence of $(y_e(\mathbf{x})|\ \mathbf{y}_e, \hat{\boldsymbol{\psi}})$ and $(y_e(\mathbf{x})|\ \mathbf{y}_c, \mathbf{y}_e, \hat{\boldsymbol{\psi}}, b_1, b_2)$ (Kennedy and O'Hagan, 2000). This intuition gives insights for the prediction of $y_e(\mathbf{x})$ detailed in the following proposition.

Proposition 1. *The posterior distribution of $y_e(\mathbf{x})$ is the following normal distribution*

$$y_e(\mathbf{x})|\ \mathbf{y}_c, \mathbf{y}_e, \hat{\boldsymbol{\psi}}, b_1, b_2 \sim \mathcal{N}(\hat{y}_e(\mathbf{x}), \hat{\sigma}_e^2(\mathbf{x})). \tag{3.6}$$

The above proposition demonstrates that the posterior mean $\hat{y}_e(\mathbf{x})$ and variance $\hat{\sigma}_e^2(\mathbf{x})$ of $y_e(\mathbf{x})$ in (3.6) can be calculated according to (2.1). If $\mathbf{x} \in \mathcal{X}_e \setminus \mathcal{X}_c$, $y_e(\mathbf{x})$ is observed; if $\mathbf{x} \notin \mathcal{X}_e$, $y_e(\mathbf{x})$ can be imputed by $\hat{y}_e(\mathbf{x})$. Thus, we only need to consider the prediction of $y_c(\mathbf{x})$ when $y_e(\mathbf{x})$ is available due to the nested property $\mathcal{X}_c \subseteq \mathcal{X}_e$. The following theorem gives the main result that the distribution of $y_c(\mathbf{x})$ conditioning on $(\mathbf{y}_c, \mathbf{y}_e, \hat{\boldsymbol{\psi}}, b_1, b_2)$ is a truncated normal distribution with the proofs detailed in Appendix S1.

Theorem 1. *For any $\mathbf{x} \in \mathcal{X}_e \setminus \mathcal{X}_c$,*

$$y_c(\mathbf{x})|\ \mathbf{y}_c, \mathbf{y}_e, \hat{\boldsymbol{\psi}}, b_1, b_2 \sim \mathcal{TN}\big(\hat{\mu}_c(\mathbf{x}),\ \hat{\sigma}_c^2(\mathbf{x});\ \hat{\rho} y_e(\mathbf{x}) + b_1,\ \hat{\rho} y_e(\mathbf{x}) + b_2\big), \tag{3.7}$$

whose mean and variance are given as follows

$$\begin{aligned} \hat{y}_c(\mathbf{x}) &= \hat{\mu}_c(\mathbf{x}) + \frac{\phi(\alpha) - \phi(\beta)}{Z} \hat{\sigma}_c(\mathbf{x}), \\ \hat{s}_c^2(\mathbf{x}) &= \hat{\sigma}_c^2(\mathbf{x}) \left(1 + \frac{\alpha\phi(\alpha) - \beta\phi(\beta)}{Z} - \Big(\frac{\phi(\alpha) - \phi(\beta)}{Z}\Big)^2\right), \end{aligned} \tag{3.8}$$

where

$$\hat{\mu}_c(\mathbf{x}) = \hat{\rho} y_e(\mathbf{x}) + \hat{\mu}_\delta + \hat{\mathbf{r}}_\delta^T \hat{\mathbf{R}}_\delta^{-1} \big(\mathbf{y}_c - (\hat{\rho} \mathbf{y}_{e_c} + \hat{\mu}_\delta \mathbf{1}_n)\big), \quad \hat{\sigma}_c^2(\mathbf{x}) = \hat{\sigma}_\delta^2 \big(1 - \hat{\mathbf{r}}_\delta^T \hat{\mathbf{R}}_\delta^{-1} \hat{\mathbf{r}}_\delta\big),$$

$$\alpha = \frac{\hat{\rho} y_e(\mathbf{x}) + b_1 - \hat{\mu}_c(\mathbf{x})}{\hat{\sigma}_c(\mathbf{x})},\ \beta = \frac{\hat{\rho} y_e(\mathbf{x}) + b_2 - \hat{\mu}_c(\mathbf{x})}{\hat{\sigma}_c(\mathbf{x})},\ Z = \Phi(\beta) - \Phi(\alpha),$$

$\hat{\mathbf{R}}_\delta$ is the correlation matrix of $\boldsymbol{\delta}_n$ whose (i, j) element is $R_{\hat{\boldsymbol{\phi}}_\delta}(\mathbf{x}_i^c - \mathbf{x}_j^c)$, $\hat{\mathbf{r}}_\delta$ is the correlation

vector between $\delta(\mathbf{x})$ *and* $\boldsymbol{\delta}_n$ *with estimated length-scale parameters* $\hat{\boldsymbol{\phi}}_\delta$*,* $\phi(\cdot)$ *and* $\Phi(\cdot)$ *are the density function and cumulative distribution function of the standard normal distribution.*

Note that the only difference between the DGP model in (3.2) and the GP-TGP model in (3.3) is that the bias function $\delta(\mathbf{x})$ is bounded in the GP-TGP model. By specifying $b_1 = -\infty$ and $b_2 = \infty$, (3.3) is equivalent to (3.2). For the MLE stage, the term $Z_c(\rho; \mu_\delta, \sigma_\delta^2, \boldsymbol{\phi}_\delta; b_1, b_2)$ disappears in the log-likelihood of the DGP model, and we denote the MLE of $\boldsymbol{\psi}$ in DGP as $\tilde{\boldsymbol{\psi}}$, to distinguish it from $\hat{\boldsymbol{\psi}}$ in the GP-TGP model. For the inference stage, the distribution of $y_c(\mathbf{x})$ conditioning on $(\mathbf{y}_c, \mathbf{y}_e, \hat{\boldsymbol{\psi}}, b_1, b_2)$ for the DGP model in (3.2) simplifies to

$$y_c(\mathbf{x})|\ \mathbf{y}_c, \mathbf{y}_e, \tilde{\boldsymbol{\psi}}, b_1, b_2 \sim \mathcal{N}\left(\hat{\mu}_c(\mathbf{x}),\ \hat{\sigma}_c^2(\mathbf{x})\right). \tag{3.9}$$

We present a toy example in Figure 3 for the illustration of the proposed GP-TGP model, where $y_e(x) = (6x-2)^2 \sin(12x-4)$, $y_c(x) = y_e(x) + 8x - 10$, $x \in [0,1], b_1 = -10, b_2 = -2$. In the plot, the solid lines in grey and black represent the true response curve of $y_e(x)$ and $y_c(x)$, respectively. We generate three CT runs with $\mathcal{X}_c = \{0, 0.73, 1\}$ in black dots and six ET runs with $\mathcal{X}_e = \{0, 0.16, 0.6, 0.73, 0.86, 1\}$ in grey upper triangles. The red dashed line denoted by $\hat{y}_{GPTGP}(x)$ represents the predictive curve of $y_c(x)$ using GP-TGP based on the three CT runs and six ET runs, and the green dashed line denoted by $\hat{y}_{GP}(x)$ represents the predictive curve of $y_c(x)$ using GP based on the three CT runs. By incorporating the ET runs of $y_e(x)$, $\hat{y}_c(x)$ lies closer to $y_c(x)$ indicating that the GP-TGP prediction is more accurate than the GP prediction in this example.

Statistica Sinica: Preprint
doi:10.5705/ss.202023.0310

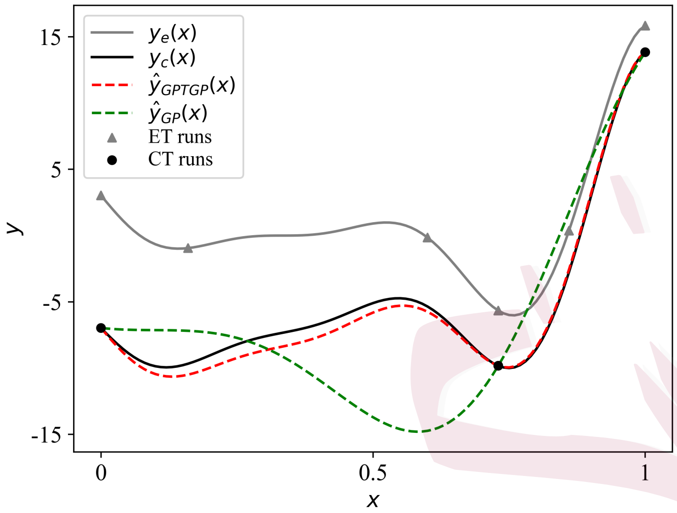


Figure 3: The comparison of GP approximation using three CT runs and GP-TGP approximation using three CT runs and six ET runs.

4. Efficient Exploration of the Hyperparameter Space

In this section, we first review existing exploration strategies for multi-fidelity computer experiments in Section 4.1. Leveraging the established surrogate model as showed in (3.6) and (3.7), we propose an efficient strategy to sequentially explore the hyperparameter space in Section 4.2. The new exploration strategy involves judiciously selecting the next configurations for both CT and ET runs to investigate, which is another major contribution of this study.

4.1 The Baseline Exploration Strategies

In the literature on multi-fidelity computer experiments, exploration strategies generally fall into two categories:: strategies for approximation, aimed at finding a good surrogate model to well approximate the response surface with as few experimental resources as possible, and strategies for optimization, which emphasizes on finding the global optimum of the response surface instead of outlining its global landscape. For cases where response surface approximation is of interest, typically just high-fidelity experiments are considered along the sequential exploration process, with a fixed number of low-fidelity experiments implemented in the initialization only to provide a rough overview of the design space with low cost (Xiong et al., 2013; Ezzat et al., 2018; Gahrooei et al., 2019). For response surface optimization, both low-fidelity and high-fidelity experiments are employed throughout the sequential exploration, guided by an acquisition function that considers both the location impact and fidelity cost, under the assumption that experimental costs at each fidelity level are fixed and known in advance (Huang et al., 2006; Picheny et al., 2013; He et al., 2017).

In the HPO problem, response surface optimization is of our primary interest, and thus both ET and CT runs should be considered along the exploration process. Huang et al. (2006) proposed the augmented EI criterion by adding cost-relevant terms to the classic EI function. Picheny et al. (2013) developed a quantile-based EQI criterion, which scores the candidate location by giving a prespecified fidelity level. But the EQI method in Picheny et al. (2013) pays more attention on dealing with response with noise, which is not suitable for the deterministic computer experiments considered in this article. He et al. (2017) proposed the EQIE criterion by adding a cost function to EQI to sequentially select the candidate location and the fidelity level at the same time. However, directly applying the above exploration strategies is inefficient in the HPO problem, because the computational costs for ET and CT runs are usually not fixed, but vary substantially across different specifications of the

Statistica Sinica: Preprint
doi:10.5705/ss.202023.0310

hyperparameters, and the nested structure and correlation between ET and CT runs are not properly considered in these studies.

4.2 The Proposed BOPT-HPO Algorithm

To fill in this gap, we propose a straightforward but effective exploration strategy for the HPO problem with both CT and ET runs. In most HPO problems in ML, the trend of ET loss $y_e(\mathbf{x})$ is consistent with the trend of CT loss $y_c(\mathbf{x})$ (see the example in Figure 2). In such case, if a hyperparameter configuration $\mathbf{x}'$ performs well in the ET stage with a small ET loss $y_e(\mathbf{x}')$, then it also performs well in the CT stage with a small CT loss $y_c(\mathbf{x}')$. Such a phenomenon has been confirmed by many studies (Kandasamy et al., 2016; Li et al., 2018; Falkner et al., 2018), suggesting that the optimal locations of $y_e(\mathbf{x})$ and $y_c(\mathbf{x})$ often locate in the same local region. Therefore, it is often effective in practice to guide the search for the optimal hyperparameters taking advantage of level-2 fidelity data via the proposed two-stage procedure. Moreover, the HPO problem in ML enjoys a unique feature: for any $\mathbf{x}$, we always get the ET loss $y_e(\mathbf{x})$ first, before we can get the CT loss $y_c(\mathbf{x})$. This is very different from the classic setting of multi-fidelity computer experiments, where experiments at different fidelity levels are implemented separately. Such a nested structure of $y_e(\mathbf{x})$ and $y_c(\mathbf{x})$ makes it practical to only consider $\mathbf{x} \in \mathcal{X}_e \setminus \mathcal{X}_c$ to implement a CT run, because even if we choose $\mathbf{x} \notin \mathcal{X}_e$ to implement a CT run, we will still get $y_e(\mathbf{x})$ at the first place.

Instead of making decision based on a comprehensive acquisition function that considers both location and fidelity of the next configuration to explore simultaneously as in Huang et al.

(2006) and He et al. (2017), we simply consider the optimal locations of a group of ET and CT runs in the configuration space $\mathcal{X}$ together, with one CT run after s ET runs, where s is a pre-given integer depending on the computational budget. We should choose a moderate s for the implementation of ET runs, otherwise, it is either too slow to find the potential candidates with too few ET runs, or it is so expensive to waste resources on some invalid tries with too many ET runs. Note that if the computation time of implementing a CT run is c times that of implementing an ET run, then it is recommended to set the value of s to be less than c. Otherwise, directly implementing a CT run is more worthwhile to obtain accurate information than implementing s ET runs. To be specific, we define the acquisition function based on the UCB criterion of an ET run $\varphi_e(\mathbf{x}) = -\hat{y}_e(\mathbf{x}) + \sqrt{\beta_m}\hat{s}_e(\mathbf{x})$, or the UCB criterion of a CT run $\varphi_c(\mathbf{x}) = -\hat{y}_c(\mathbf{x}) + \sqrt{\beta_n}\hat{s}_c(\mathbf{x})$, where β_m and β_n can be specified as suggested by Srinivas et al. (2010). Hyperparameter configuration of the next ET or CT run can be determined by maximizing $\varphi_e(\mathbf{x})$ or $\varphi_c(\mathbf{x})$, respectively. In case that a new ET run is needed, we search the whole configuration space $\mathcal{X}$ for the optimal candidate:

$$\mathbf{x}_{m+1}^{e} = \arg\max_{\mathbf{x}\in\mathcal{X}} \varphi_e(\mathbf{x}). \tag{4.1}$$

When a new CT run is needed, however, we only search $\mathcal{X}_e \setminus \mathcal{X}_c$ for the optimal candidate:

$$\mathbf{x}_{n+1}^{c} = \underset{\mathbf{x}\in\mathcal{X}_e\setminus\mathcal{X}_c}{\mathrm{argmax}}\, \varphi_c(\mathbf{x}). \tag{4.2}$$

Since all CT runs are implemented after ET runs in this strategy, $\mathcal{X}_c$ and $\mathcal{X}_e$ are naturally nested. Note that we need to record the detailed status of the ET runs that have not been evaluated in CT runs yet in the memory, to guarantee that we can easily start from an intermediate status and move forward from an ET run to a CT run.

Integrating the fitting stage in Section 3 and the exploration stage in Section 4 into the general BO framework, we come up with an intact BO algorithm as shown in Algorithm 1, which is referred to as *Bayesian Optimization with Pareto-Principled Training for Hyperparameter Optimization* (BOPT-HPO). The BOPT-HPO contains two BO procedures, one for $y_e(\cdot)$ and another for $y_c(\cdot)$. The inner *for* loop is a BO procedure to select s ET runs which have potentials in improving the prediction model and current minimum, where the fitting stage of $y_e(\cdot)$ is implemented via (3.6), and the exploration stage is implemented via (4.1). The outer *while* loop is another BO procedure to select one CT run parsimoniously which has the most potential from the remaining set $\mathcal{X}_e \setminus \mathcal{X}_c$, via the fitting stage using GP-TGP in (3.7) or DGP in (3.9), and the exploration stage using (4.2).

In practice, we also need an initialization stage to start up the BO procedure. We recommend using the *nested Latin hypercube design* (NLHD) (Qian, 2009) to generate the initial configurations $(\mathcal{X}_e, \mathcal{X}_c)$ satisfying $\mathcal{X}_c \subseteq \mathcal{X}_e$, which can guarantee the space-filling properties for both fidelity levels of initial configurations. There is no golden standard in setting the initial sample size for multi-fidelity experiments, which depends on the cost of running different fidelity levels of experiments or other prior knowledge.

Under the BOPT-HPO framework, by specifying the model in the fitting stage as GP-TGP in (3.3), an algorithm referred to as TGP-BOPT is obtained as our primary method for HPO. On the other hand, we can also replace the GP-TGP model by the traditional DGP model to

Statistica Sinica: Preprint
doi:10.5705/ss.202023.0310

Algorithm 1 Pseudocode for BOPT-HPO

```
Initialization: Generate ET and CT configurations $(\mathcal{X}_e, \mathcal{X}_c)$ satisfying $\mathcal{X}_c \subseteq \mathcal{X}_e$, $n = |\mathcal{X}_c|, m = |\mathcal{X}_e| = sn$, and evaluate the corresponding validation errors $(\mathbf{y}_e, \mathbf{y}_c)$.
while the stopping criterion is not met do
  for $j = 1, \ldots, s$ do
    Fit the prediction model of $y_e(.)$ based on ET data $(\mathcal{X}_e, \mathbf{y}_e)$.
    Calculate $\mathbf{x}^e_{m+1} = \mathrm{argmax}_{\mathbf{x}\in\mathcal{X}}\, \varphi_e(\mathbf{x})$.
    Get the ET run $y_e(\mathbf{x}^e_{m+1})$, and set $\mathcal{X}_e = \mathcal{X}_e \cup \{\mathbf{x}^e_{m+1}\}$ and $\mathbf{y}_e = \mathbf{y}_e \cup \{y_e(\mathbf{x}^e_{m+1})\}$.
  end for
  Fit the prediction model of $y_c(.)$ based on ET and CT data $(\mathcal{X}_e, \mathbf{y}_e, \mathcal{X}_c, \mathbf{y}_c)$.
  Choose the configuration with the largest $\varphi_c$ value from $\mathcal{X}_e \setminus \mathcal{X}_c$ as $\mathbf{x}^c_{n+1}$.
  Get the CT run $y_c(\mathbf{x}^c_{n+1})$, and set $\mathcal{X}_c = \mathcal{X}_c \cup \{\mathbf{x}^c_{n+1}\}$ and $\mathbf{y}_c = \mathbf{y}_c \cup \{y_c(\mathbf{x}^c_{n+1})\}$.
end while
return $\mathbf{x}^*_c = \mathrm{argmin}_{\mathbf{x}\in\mathcal{X}_c}\ \mathbf{y}_c$.
```

get an alternative baseline method referred to as DGP-BOPT under the BOPT-HPO framework, for comparison purpose. Because DGP-BOPT does not consider the constraint between a CT run and the corresponding ET run as in TGP-BOPT, we would expect performance degradation from TGP-BOPT to DGP-BOPT. By analyzing the degree of the performance degradation, we can gain more insights on the advantages of TGP-BOPT over other methods.

5. Experiments

In this section, we evaluate the performances of the proposed BOPT-HPO algorithms (TGP-BOPT and DGP-BOPT) via a series of optimization or HPO tasks. Six competing methods, including five methods in the HPO literatures (Random Search (RS), TPE, SMAC, GP-BO and BOHB), and one multi-fidelity optimization method in computer experiments (EQIE), are used for comparison. In all of these ML tasks, we have found that the computation time of implementing a CT run is 3-4 times that of implementing an ET run (see Figure 1 in Appendix), so we fix $s = 2$ for the BOPT-HPO. The settings of b_1 and b_2, and the definitions of ET runs and CT runs of the three ML tasks can be found in Appendix S2.

5.1 Optimizing Synthetic Black-Box Functions

Our first experiment is about finding the maximum or minimum of a synthetic black-box function $y_c(\mathbf{x})$, which can be approximated by a cheaper function $y_e(\mathbf{x})$. The 2-dimensional Currin exponential example, 4-dimensional Park example, and 10-dimensional Rosenbrock example are considered (Xiong et al., 2013; Mainini et al., 2022). The definitions of the three examples can be found in Appendix S2.1.

Unlike evaluating on machine learning models that is time-consuming, evaluating on synthetic functions for $y_e(.)$ and $y_c(.)$ using the actual computational time as cost is not suitable. Similar to the cost settings for the synthetic functions in Huang et al. (2006) and Stroh et al. (2022), we assume that the cost for evaluating one CT run is 1 cost unit, and the cost for evaluating one ET run is 1/5 cost unit. Denote the true optimal value of $y_c(\mathbf{x})$ as y_c^*, where $y_c^* = \max_{\mathbf{x}\in\mathcal{X}} y_c(\mathbf{x})$ for maximization problem (Currin exponential and Park example) or $y_c^* = \min_{\mathbf{x}\in\mathcal{X}} y_c(\mathbf{x})$ for minimization problem (Rosenbrock example). For any black-box function optimization method with $\mathcal{X}_c$ as the locations it has explored in the exploration space

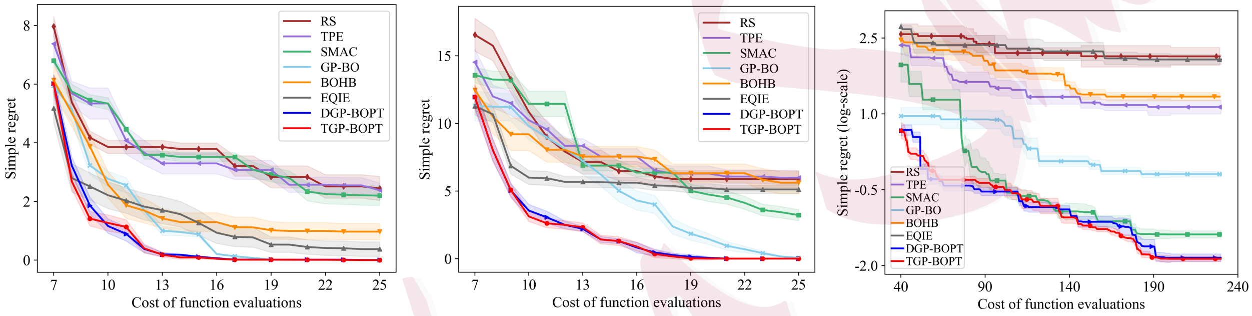
$\mathcal{X}$, let $\hat{y}_c^*$ denote the best value explored in $\mathcal{X}_c$ for one method. The Simple Regret (SR) defined below serves as a natural performance measurement: SR$= |y_c^* - \hat{y}_c^*|$, where a smaller SR demonstrates a better performance.

Competing methods RS, TPE, SMAC, GP-BO, BOHB and EQIE can also be directly applied to solve these problems and used for comparison. We generated initial CT runs and ET runs for BOPT-HPO algorithms using NLHD. For the 2-dimensional and 4-dimensional case, we generated 5 CT runs and 10 ET runs at an initial budget of $5 + 10 \times \frac{1}{5} = 7$ cost units. For the 10-dimensional case, we generated 30 CT runs and 60 ET runs at an initial budget of $30 + 60 \times \frac{1}{5} = 42$ cost units. For a fair comparison of those approaches, we reported the corresponding SRs of other approaches starting from the initial budget as in BOPT-HPO algorithms. Note that although the synthetic functions themselves are deterministic, stochastic behaviors in initialization and optimization algorithms (e.g., using Monte Carlo strategies in model parameter estimation, and sampling the configurations for the HPO approaches, such as RS) may introduce randomness into the HPO procedure. To average out the impact of randomness in performance evaluation, we replicated the experiments of each method for 10 times, and summarized the average performance for method comparison in Figure 4.

Figure 4 demonstrates how the average SR value changes with the cost under different methods in the three synthetic examples respectively, with the standard errors of the SR values across 10 replicates highlighted by the shadow region as well. We find that TGP-BOPT and DGP-BOPT perform similarly and outperform all the other methods significantly in all three examples, achieving the smallest regret at almost every cost level. The performance gap among

Statistica Sinica: Preprint
doi:10.5705/ss.202023.0310

GP-BO, EQIE and BOPT-HPO enlarges from the Currin exponential example in Figure 4(a) to the Rosenbrock example in Figure 4(c) with the dimensionality of the problem increases from 2 to 10. These results suggest that by jointly modeling the ET and CT evaluations, BOPT-HPO can approximate $y_c(\mathbf{x})$ more precisely, and find the optimal configuration more efficiently.



(a) Currin exponential ($d = 2$)

(b) Park ($d = 4$)

(c) Rosenbrock ($d = 10$)

Figure 4: The simple regret against the cost of function evaluations. All curves are produced by averaging over 10 replications with the standard errors highlighted by the shadow region. Log-transformation is applied to the simple regret of Rosenbrock for better visualization.

5.2 Support Vector Machines on MNIST

Next, we assess the performance of BOPT-HPO on a real HPO problem of a *support vector machine* (SVM) for an image classification task on the MINST dataset consisting of 70,000 black-and-white images of handwritten digits in 10 classes (60,000 images for training and 10,000 images for testing) (LeCun et al., 2010). As a supervised machine learning algorithm for classification, SVM creates a hyperplane with the maximum distance between data points of different classes in training data, so that new points in test data can be classified with more confidence. The behavior of SVM is determined by two hyperparameters: the regularization hyperparameter C, which adds a penalty for each mis-classified data point, and the kernel hyperparameter γ, which controls the influence of similarity measurement of data points when the non-linear radial basis function kernel is applied. The HPO problem in SVM aims to optimize $\mathbf{x} = (C, \gamma)$ over the pre-specified configuration space $\mathcal{X} = [2^{-10}, 2^{10}] \times [2^{-10}, 2^{10}]$.

Considering that directly searching the vast configuration space $\mathcal{X}$ for the next configuration is inefficient, we take the log-transformation on $\mathcal{X}$, and optimize $\mathbf{x}_{log} = \log \mathbf{x}$ over $\log \mathcal{X} = [-10, 10] \times [-10, 10]$ when applying HPO methods and back transform $\mathbf{x}_{log}$ to $\mathbf{x}$ instead when applying the SVM. Given a hyperparameter configuration, SVM is trained on the training set, and the classification error on the validation set (referred to as *validation error*) is reported as the measurement over the running time (Snoek et al., 2012; Falkner et al., 2018). For the MNIST data, we randomly sampled 40,000 images from the original training images as the training set, and use the rest 20,000 images as the validation set, following the similar splitting rule in Domhan et al. (2015). The other six methods RS, TPE, GP-BO, SMAC, BOHB and EQIE, are applied to this experiment, and all the eight methods are allocated with the same amount of computational resources for fair comparison.

Figure 5 shows the average validation error of the methods mentioned above with five replicated experiments. The markers with different colors on the x-axis indicate the initialization time of different methods. The figure demonstrates that the model-based methods GP-BO, SMAC, BOHB and BOPT-HPO reach almost the same comparative final performances and perform much better than RS and TPE. Considering the time to reach the best performance,

Statistica Sinica: Preprint
doi:10.5705/ss.202023.0310

there is not much difference between BOHB and TGP-BOPT, however, TGP-BOPT is faster in reducing the validation error during the middle searching stage. Benefiting from the constraints of the GP-TGP model, TGP-BOPT reaches the best performance a bit earlier than DGP-BOPT. As for the two GP-based approaches, TGP-BOPT is almost two times faster than GP-BO for reaching the best performance.

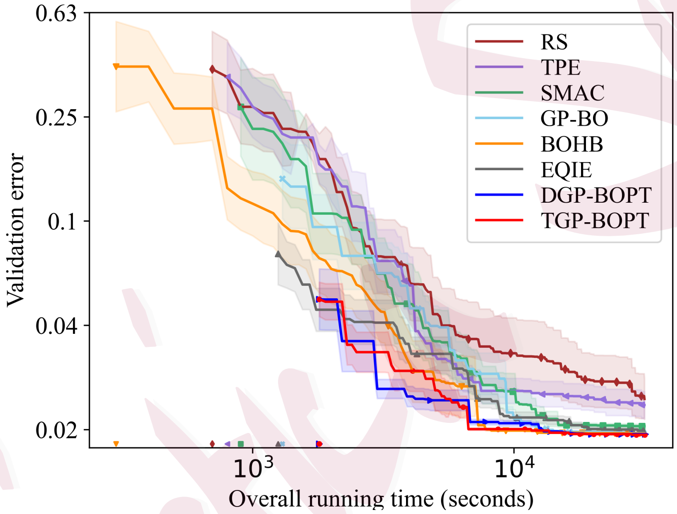


Figure 5: The average results of different HPO methods for optimizing two hyperparameters of SVM on MNIST with standard errors highlighted by the shadow region.

5.3 Feed-Forward Neural Networks on MNIST

In this experiment, we explore the effectiveness of BOPT-HPO on a more complex machine learning algorithm, a *feed-forward neural network* (FFNN), which consists of three main components: an input layer, one or more hidden layers, and an output layer. We want to optimize the hyperparameters of a FFNN, for the image classification task on MNIST. The size of the training set and validation set is the same as in the previous experiment. We optimize six hyperparameters which determines the architecture (number of layers, neurons per layer) and the training process (batch size, initial learning rate, exponential decay factor for learning rate, dropout rate) of a feed-forward neural network, following the settings in Section 5.2.2 of Falkner et al. (2018). The ranges of those hyperparameters are shown in Table 1 in Appendix.

Different from the intuition in Section 5.2, the range of the learning rate is too small which makes HPO methods hard to find the hyperparameter meeting the required accuracy, so we take log-transformation on the learning rate when applying the HPO methods and then back transform it when applying the FFNN. With the number of layers, number of neurons and batch size being integer values, we treat the log-transformed spaces of the two hyperparameters (number of neurons and batch size) as continuous when applying HPO methods, and then round those values to the closest integers to implement the FFNN. Since the hyperparameter 'number of layers' in the neural network takes only three integer values from 1 to 3, we optimize it directly in the original discrete space by exhaustive enumeration.

The results of the eight HPO methods with five replicated experiments are summarized in Figure 6. The model-based methods TPE, GP-BO, SMAC, BOHB, EQIE and BOPT-HPO perform better than the model-free method RS. EQIE does not perform well in this experiment, due to the fact that its performance is sensitive to the choice of the cost function which is difficult to specify without prior knowledge. Benefiting from the joint modeling and smart design strategies for the exploration and exploitation on ET and CT runs, BOPT-HPO consistently outperforms other HPO methods and achieves a performance improvement of 5% to 31%. The performances of TGP-BOPT and DGP-BOPT are comparable in this experiment.

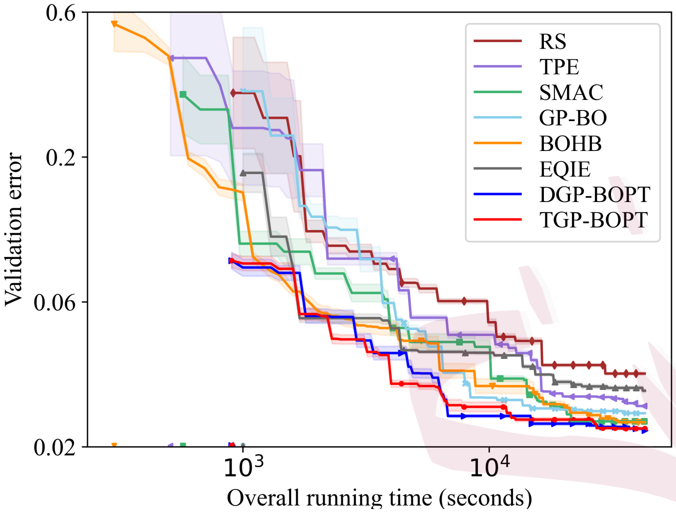


Figure 6: A comparison of different HPO methods for the hyperparameter optimization of the FFNN on MNIST with standard errors highlighted by the shadow region.

5.4 Convolutional Neural Network on CIFAR-10

For the last experiment, we consider a more difficult classification task on a more complex database, CIFAR-10 (Krizhevsky et al., 2014), which consists of 60,000 colour images in 10 classes, with 6,000 images per class. We design a more expensive machine learning algorithm, a *convolutional neural network* (CNN), to better solve the classification task. The convolutional layer is the core layer of a CNN architecture, aiming to extract features from local region to decrease the spatial redundancy via some filters. In this experiment, we try to evaluate the performance of the proposed BOPT-HPO using a CNN. We consider eight hyperparameters (number of convolutional layers, number of filters per convolutional layer, number of fully-connected neurons in the last hidden layer, batch size, initial learning rate, dropout rate, optimizer, momentum) of the CNN model, with the ranges and log-transform strategy shown in Table 2 in Appendix. We randomly select 70% of the images from the original training images as the training set, and use the remaining images as the validation set to evaluate the performance.

The average results in Figure 7 with five replicates shows that all model-based methods substantially outperform RS, but TPE, GP-BO and EQIE are not so competitive as SMAC, BOHB and BOPT-HPO in terms of the best validation error. TGP-BOPT achieves the best validation error of 0.261, which performs slightly better than SMAC and BOHB with the best validation error of 0.263. While BOPT-HPO outperforms SMAC and BOHB during the middle optimization stages, which leads to an optimization efficiency improvement of 14% to 24%.

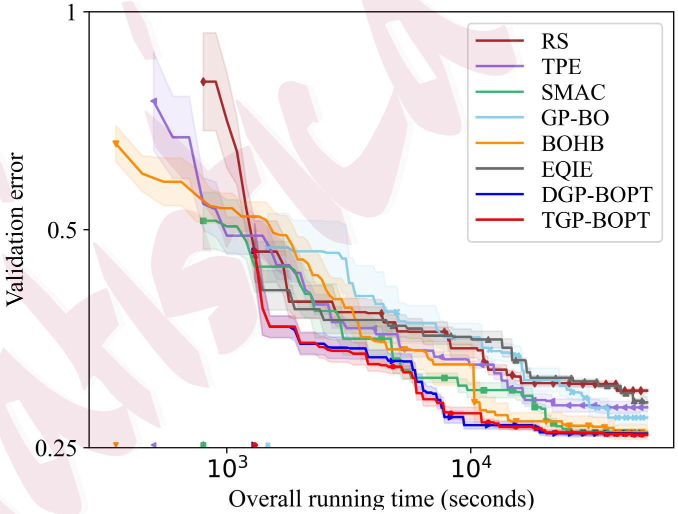


Figure 7: The average results of different HPO strategies for optimizing the hyperparameters of the CNN on CIFAR-10 with standard errors highlighted by the shadow region.

We have shown the superiority of the proposed BOPT-HPO (DGP-BOPT or TGP-BOPT) algorithm over existing HPO methods in terms of overall running time in Figures 5-7. In Appendix S2.4, we also provide a detailed analysis of the computational time required for ET

Statistica Sinica: Preprint
doi:10.5705/ss.202023.0310

runs and CT runs in BOPT-HPO. Those findings further confirm the effectiveness of the two-stage exploration strategy via using ET runs to guide the search of the optimal hyperparameter configuration.

The results in Figures 4-7 demonstrate that both TGP-BOPT and DGP-BOPT achieve similar optimal performances. This phenomenon is intuitively natural because with more and more data collected along the BO process, the DGP model can implicitly learn constraints from the data, even without explicitly setting constraints in the model. However, if we look closely into the result curves in these figures, we will find that the result curves of TGP-BOPT in general converge a bit faster than those of DGP-BOPT. To provide quantitative evidences to such a claim, we introduce the concept of the Area Under the Error-Cost Curve (EC-AUC) to precisely evaluate the overall performance of an approach, with detailed results shown in Table 4 of Appendix S2.5. A smaller EC-AUC indicates that the corresponding approach converges quickly in general, and thus performs better. In most experiments except the SVM case, TGP-BOPT has a smaller EC-AUC than DGP-BOPT. In the SVM case, although the EC-AUC of TGP-BOPT is slightly larger than DGP-BOPT, TGP-BOPT reaches the optimal performance faster. Our investigations indicate the advantage of modeling the constraint explicitly in TGP-BOPT.

6. Conclusion

In this paper, we propose BOPT-HPO, a multi-fidelity Bayesian optimization method, to solve the hyperparameter optimization problem, which has attracted great attentions in the machine learning community. Modelling complete training runs and eighty-percent training runs jointly via GP-TGP and selecting the configurations for future ET and CT runs with a sequential design strategy under the framework of Bayesian optimization, BOPT-HPO explores the hyperparameter space efficiently with constraints between ET and CT runs properly considered. Experiments on both synthetic and real examples show that BOPT-HPO outperforms state-of-the-art HPO methods for various machine learning algorithms.

Although BOPT-HPO is only applied to solve HPO problem for several shallow neural networks with only a few hyperparameters in this work, while it is in principle capable of dealing with deeper neural networks with more hyperparameters. When the deep neural networks of interest involves too many hyperparameters, HPO becomes a very challenging task due to the high dimensionality of the hyperparameter space. In such cases, using dimension reduction techniques suggested by Binois and Wycoff (2022) can somehow alleviate this issue.

Moreover, the proposed framework can be further extended in several directions. For instance, it is plausible to replace the truncated Gaussian process for the bias function $\delta(\cdot)$ by a log-Gaussian process to avoid the complication caused by the boundary effect of the truncated Gaussian process. In addition, ET runs and CT runs are selected sequentially in this work based on two independent acquisition functions, one for each fidelity level. A possibly more efficient strategy, however, is to design an augmented acquisition function that considers location, and computational costs of different fidelity levels together to guide the exploration of hyperparameter space with multiple fidelity levels once at all. In fact, this could be achieved by building a pair of new surrogate models for the running time of ET and CT runs, i.e., $t_e(\mathbf{x})$

and $t_c(\mathbf{x})$, respectively. But, this would need much more efforts that clearly go beyond the scope of this work.

Supplementary Material

Details about the proof of Theorem 1 and some experimental settings and results can be found in the supplementary materials.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (Grant No. 2023YFF0614702), and the National Natural Science Foundation of China (Grant Nos. 12401353 and 12371269). The authors knowledge the summer support of Huzhou University.

References

Batra, R., L. Song, and R. Ramprasad (2021). Emerging materials intelligence ecosystems propelled by machine learning. *Nature Review Materials 6*(8), 655–678.

Bergstra, J., R. Bardenet, Y. Bengio, and B. Kégl (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, Volume 24.

Binois, M. and N. Wycoff (2022). A survey on high-dimensional gaussian process modeling with application to bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization 2*(2), 1–26.

Statistica Sinica: Preprint
doi:10.5705/ss.202023.0310

Candelieri, A. and F. Archetti (2021). Sparsifying to optimize over multiple information sources: an augmented gaussian process based algorithm. *Structural and Multidisciplinary Optimization 64*, 239–255.

Domhan, T., J. T. Springenberg, and F. Hutter (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-fourth International Joint Conference on Artificial Intelligence*, pp. 3460–3468.

Ezzat, A. A., A. Pourhabib, and Y. Ding (2018). Sequential design for functional calibration of computer models. *Technometrics 60*(3), 286–296.

Falkner, S., A. Klein, and F. Hutter (2018). BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1437–1446. PMLR.

Forrester, A. I. J., A. Sóbester, and A. J. Keane (2007). Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 463*, 3251–3269.

Gahrooei, M. R., K. Paynabar, M. Pacella, and B. M. Colosimo (2019). An adaptive fused sampling approach of high-accuracy data in the presence of low-accuracy data. *IISE Transactions 55*(11), 1251–1264.

Genz, A. (1992). Numerical computation of multivariate normal probabilities. *Journal of Computational and Graphical Statistics 1*(2), 141–149.

Goh, J., D. Bingham, J. P. Holloway, M. J. Grosskopf, C. C. Kuranz, and E. Rutter (2013). Prediction and computer model calibration using outputs from multifidelity simulators. *Technometrics 55*(4), 501–512.

Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies 10*, 1–309.

Gramacy, R. B. (2020). *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences.* CRC press.

He, X., R. Tuo, and C. F. J. Wu (2017). Optimization of multi-fidelity computer experiments via the EQIE criterion. *Technometrics 59*(1), 58–68.

Hinton, G. E. (2012). A practical guide to training restricted boltzmann machines. In *Neural Networks: Tricks of the Trade*, pp. 599–619. Springer.

Huang, D., T. T. Allen, W. I. Notz, and R. A. Miller (2006). Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization 32*, 369–382.

Hutter, F., H. H. Hoos, and K. Leyton-Brown (2011). Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pp. 507–523. Springer.

Jones, D. R. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization 21*(4), 345–383.

Jones, D. R., M. Schonlau, and W. J. William (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization 13*(4), 455–492.

Kandasamy, K. (2018). *Tuning hyperparameters without grad students: Scaling up bandit optimisation.* Ph. D. thesis, Carnegie Mellon University.

Kandasamy, K., G. Dasarathy, J. B. Oliva, J. Schneider, and B. Poczos (2016). Gaussian process bandit optimisation with multi-fidelity evaluations. In *Advances in Neural Information Processing Systems*, Volume 29.

Kennedy, M. C. and A. O'Hagan (2000). Predicting the output from a complex computer code when fast approximations are available. *Biometrika 87*(1), 1–13.

Krizhevsky, A., V. Nair, and G. Hinton (2014). Cifar-10 and cifar-100 datasets. *Retrieved from* *https://www.cs.toronto.edu/ kriz/cifar.html*.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, Volume 25.

Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering 86*, 97–106.

Kuya, Y., K. Takeda, X. Zhang, and A. I. J. Forrester (2011). Multifidelity surrogate modeling of experimental and computational aerodynamic data sets. *AIAA Journal 49*(2), 289–298.

Le Gratiet, L. and C. Cannamela (2015). Cokriging-based sequential design strategies using fast ccross-validation techniques for multi-fidelity computer codes. *Technometrics 57*, 418–427.

LeCun, Y., L. Bottou, G. Orr, and K. Müller (2012). Efficient backprop. In *Neural Networks: Tricks of the Trade*, pp. 9–48. Springer.

LeCun, Y., C. Cortes, and C. Burges (2010). Mnist handwritten digit database. *Retrieved from http://yann.lecun.com/exdb/mnist*.

Li, L., K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research 18*(185), 1–52.

Mainini, L., A. Serani, M. P. Rumpfkeil, E. Minisci, D. Quagliarella, H. Pehlivan, S. Yildiz, S. Ficini, R. Pellegrini, F. Di Fiore, et al. (2022). Analytical benchmark problems for multifidelity optimization methods. *arXiv preprint arXiv:2204.07867*.

Marler, R. T. and J. S. Arora (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization 26*(6), 369–395.

Picheny, V., D. Ginsbourger, Y. Richet, and G. Caplin (2013). Quantile-based optimization of noisy computer experiments with tunable precision. *Technometrics 55*(1), 2–13.

Poloczek, M., J. Wang, and P. Frazier (2017). Multi-information source optimization. In *Advances in Neural Information Processing Systems*, Volume 30.

Prechelt, L. (2012). Early stopping-But when? In *Neural Networks: Tricks of the Trade*, pp. 53–67. Springer.

Qian, P. Z. G. (2009). Nested latin hypercube designs. *Biometrika 96*(4), 957–970.

Qian, P. Z. G. and C. F. J. Wu (2008). Bayesian hierarchical modeling for integrating low-accuracy and high-accuracy experiments. *Technometrics 50*(2), 192–204.

Qian, Z., C. C. Seepersad, V. R. Joseph, J. K. Allen, and C. F. J. Wu (2006). Building surrogate models based on detailed and approximate simulations. *Journal of Mechanical Design 128*, 668–677.

Rasmussen, C. E. (2004). Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, pp. 63–71. Springer.

Sanders, R. (1987). The pareto principle: Its use and abuse. *Journal of Services Marketing 1*(2), 37–40.

Shahriari, B., K. Swersky, Z. Y. Wang, R. R. Adams, and N. de Freitas (2016). Taking the human out of the loop: A review of bayesian optimization. In *Proceedings of the IEEE*, Volume 104, pp. 148–175.

Snoek, J., H. Larochelle, and R. P. Adams (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, Volume 25.

Srinivas, N., A. Krause, S. Kakade, and M. Seeger (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 1015–1022.

Stroh, R., J. Bect, S. Demeyer, N. Fischer, D. Marquis, and E. Vazquez (2022). Sequential design of multi-fidelity computer experiments: Maximizing the rate of stepwise uncertainty reduction. *Technometrics 64*(2), 199–209.

Tuo, R., C. F. J. Wu, and D. Yu (2014). Surrogate modeling of computer experiments with different mesh densities. *Technometrics 56*(3), 372–380.

Xiong, S., P. Z. G. Qian, and C. F. J. Wu (2013). Sequential design and analysis of high-accuracy and low-accuracy computer codes. *Technometrics 55*(1), 37–46.

Yang, Y., C. Ji, and K. Deng (2021). Rapid design of metamaterials via multitarget bayesian optimization. *The Annals of Applied Statistics 15*(2), 768–796.

Zhu, Z., D. W. Ng, H. S. Park, and M. C. McAlpine (2021). 3D-printed multifunctional materials enabled by artificial-intelligence-assisted fabrication technologies. *Nature Review Materials 6*, 27–47.

School of Statistics and Data Science, LPMC and KLMDASR, Nankai University, Tianjin 300071, China

E-mail: yangyang_nk@nankai.edu.cn

Department of Statistics and Data Science, Tsinghua University, Beijing 100084, China

E-mail: kdeng@tsinghua.edu.cn

Department of Statistics, Purdue University, West Lafayette, Indiana 47907, U.S.A.

E-mail: yuzhu@purdue.edu