24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Derivative-Free Optimization with Adaptive Experience for Efficient Hyper-Parameter Tuning[1]

Yi-Qi Hu[1,2] and **Zelin Liu**[1,2] and **Hua Yang**[2] and **Yang Yu**[1] and **Yunfeng Liu**[2]

Abstract. Hyper-parameter tuning is a core part of automatic machine learning (AutoML), which aims to automatically configure machine learning systems in deployed applications. Previously, hyper-parameter tuning is usually formulated as a black-box optimization problem, for which derivative-free optimization (DFO) solver is often employed. Such solvers often suffered from low-efficiency. Thus experienced DFO was proposed, which utilizes historical optimization process data to guide the optimization on new problems. However, the effectiveness of experienced DFO is sensitive to the relevance between the experienced tasks and the target tasks. Relevant experience can accelerate the convergence, while irrelevant experience could injure the convergence. This paper proposes an adaptation mechanism for the experienced DFO. It learns a set of experience models to guide the DFO processes, and exams these models on a few labeled samples from the target task. By comparing model predictions with the ground-truth labels, it adaptively learns the relevant experience by weighting those models. The experiments on synthetic tasks verify that the proposed method can effectively adopt the relevant experience for a range of target tasks. Furthermore, we apply the proposed method to the tasks of configuring LightGBM hyper-parameters. The empirical results show that the proposed method effectively selects the relevant experience and significantly improves the performance of hyper-parameter tuning in only a few iterations.

1 Introduction

Machine learning has been proved as a practical technique and widely used in many applications, e.g., computer vision [6], natural language processing [10], recommendation systems [8], etc. However, successful applications of machine learning usually rely on careful configurations of learning processes. However, manual configurations deeply depend on expert knowledge and spend plenty of human power. To tackle this issue, automatic machine learning (AutoML) [35] is proposed. Without any human participation, it tries to automatically configure machine learning processes including data pre-processing [21], algorithm selection [1, 5, 7, 16], hyper-parameter tuning [2, 3, 17, 13, 29], etc. AutoML is usually formulated as a black-box optimization problem, such as the *combined algorithm selection and hyper-parameter optimization* (CASH) problem in [11]. This formulation is non-convex, non-continuous, and non-differentiable. Thus, the gradient-based optimization methods are hard to be applied. However, derivative-free optimization (DFO) is suitable for such situations [23]. Derivative-free optimization [19, 36, 15, 4, 14] follows the trial-and-error framework, which only relies on the evaluation values of samples to accomplish the DFO processes. By applying derivative-free optimization, some popular AutoML tools [11, 31] have been developed and achieved successes in some data mining competitions.

Without gradients, derivative-free optimization methods need many samples together with their evaluations to explore the search space. An evaluation of AutoML tasks involves a full validation process. Thus, many evaluations imply very high time-cost. As a result, previous AutoML solvers always suffer from the low-efficient issue, i.e., it spends a long time on finding a good configuration. Experienced derivative-free optimization [18] could be a way to alleviate this issue. It involves a two-part process. The first part is to extract experience data from some historical optimization processes and learn directional models based on the experience data. The second part is to utilize the directional models to guide the derivative-free optimization process on unseen tasks. With the guidance of the directional models, the experienced derivative-free optimization can successfully avoid unnecessary evaluations and accelerate convergence with only a few samples. While this method shows improvement in DFO processes from scratch, it could suffer from the uncontrollable relevance between the target task and the source tasks. Irrelevant directional models will provide wrong guidance as to slow down the convergence on target tasks. Hence, it is urgent to maintain experience relevance.

In this paper, we proposed the *experience adaptation* mechanism to selectively adopt relevant experience for experienced derivative-free optimization. It follows a simple principle: on target tasks, we will keep the directional models (experience) that correctly guide the DFO processes, while omitting the directional models that wrongly guide the DFO processes. The correctness of direction guidance is easy to be tested in the target tasks. Directional models guide the DFO process by predicting a search direction for the next sample. If the next sample successfully improves the optimization performance, we consider the predicted direction correctly guide the search. On the contrary, if the next sample unsuccessfully improves the optimization performance, we consider the predicted direction wrongly guide the search. Through the guidance-exam way, relevant directional models can be discovered. We implemented this experience adaptation mechanism based on the classification-based optimization RACOS [36, 15], and named it ADARACOS. The experiment results on synthetic tasks show that ADARACOS can effectively find the relevant directional models on target tasks. Then, we apply ADARACOS to tune LightGBM hyper-parameters on 40 datasets. It verifies that

[1] This work is supported by NSFC (61876077), Jiangsu SF (BK20170013), and Collaborative Innovation Center of Novel Software Technology and Industrialization. Yang Yu is the corresponding author. This work is done when Yi-Qi Hu and Zelin Liu were interns in Zhuiyi Technology Co.,Ltd.
Authors' institutions:
[1] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China.
[2] Zhuiyi Technology Co., Ltd., China.
Email addresses: {huyq,yuy}@lamda.nju.edu.cn

ADARACOS can significantly improve the performance of hyper-parameter tuning with a few evaluation budgets.

The rest four sections of this paper present background & related works, proposed method, experiments, and conclusions.

2 Background & Related Works

In this paper, we focus on the hyper-parameter tuning task that can be formulated as a black-box optimization problem. Let $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ denote the training and the testing datasets. For an algorithm, let $\boldsymbol{\delta} \in \Delta$ denote a hyper-parameter configuration, where Δ is the hyper-parameter space. We use a criterion $f(\cdot)$ to evaluate a hyper-parameter configuration such as accuracy, AUC score, F1 score, etc. The hyper-parameter tuning task can be formulated as follows:

$$\boldsymbol{\delta}^* = \underset{\boldsymbol{\delta}\in\Delta}{\operatorname{argmax}} \frac{1}{k}\sum_{j=1}^{k} f\left(\boldsymbol{\delta}, \mathcal{D}_j^{\text{train}}, \mathcal{D}_j^{\text{valid}}\right), \tag{1}$$

where $\mathcal{D}_j^{\text{valid}} = \mathcal{D}^{\text{train}} - \mathcal{D}_j^{\text{train}}$ is the validating dataset at j-th fold. Like the CASH problem [11], this formulation is essentially maximizing the k-fold validation performance on the hyper-parameter space of an algorithm. Due to the non-convex, non-continuous and non-differentiable properties, the hyper-parameter tuning is a hard optimization problem.

Derivative-free optimization is suitable for solving hyper-parameter tuning tasks like Eq. (1). Previous derivative-free optimization methods include evolutionary algorithms [12, 37, 26], Bayesian optimization [27, 29], classification-based optimization [36, 15], etc [25]. They all share the same optimization framework that consists of sampling and evaluating. Without gradients, derivative-free optimization needs plenty of samples and evaluations to explore the search space. Due to the high evaluation cost, derivative-free optimization suffers from the low-efficient issue when solving hyper-parameter tuning tasks. Thus, some high-efficient algorithms are applied as the task solvers such as SMAC [19], TPE [4], RACOS [36, 15], etc. These algorithms generate new samples and explore the search space only according to the evaluation values. Without extra information, derivative-free optimization is hard to break through the efficient bottleneck.

Recently, meta-knowledge [30, 24, 18] is considered to improve the efficiency of hyper-parameter tuning. For example, in [24], the authors employ the best samples of historical tasks to initialize the Bayesian optimization processes on new tasks, i.e., warm-starting. Experienced DFO [18] is also a meta-knowledge based method. The meta-knowledge is a set of directional models that are learned from the DFO processes of historical tasks. With the guidance of predicted directions, it effectively avoids many unnecessary evaluations and accelerates DFO convergence within a few sample evaluations. However, the effectiveness of directional models is sensitive to the relevance between the source and target tasks. Figure 1 simply illustrates the relevant issue among source and target tasks. In the figure, the X-axis is the distance that measures the relevance between the source tasks and the target task. The small distance means high relevance. The Y-axis is the optimization performance in the Sphere function. It is a minimization task. The results show a positive correlation between the distance and the optimization performance. It means that experienced DFO with a relevant directional model is easier to obtain better performance.

The relevance among tasks is frequently considered on meta learning [32] such as one-shot learning [22, 33], few-shot learning [28], model reuse [38, 34], etc. Previously, these works manually define

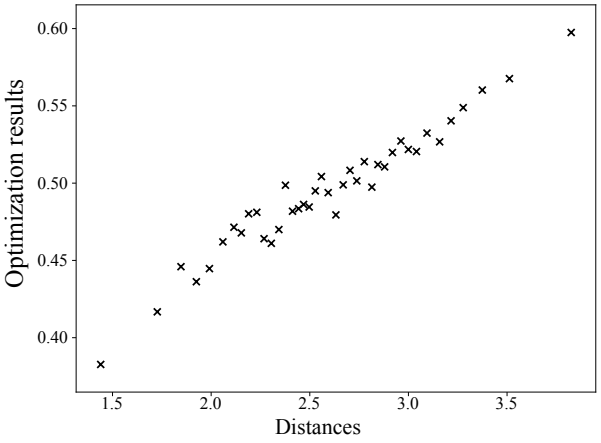


Figure 1. Illustration of relevance between source and target tasks. The task is minimizing Sphere function. We randomly shift the optimal points of Sphere function to construct the task distribution. The target task is Sphere function with $\{0.1\}^n$ as the optimal point, where $n = 10$ is the dimensionality. The X-axis is the Euclidean distance between the optimal points of source and target tasks that measures the relevance. The Y-axis is the performance of experienced DFO with only 50 evaluation budget.

the relevance among different tasks. For example, the target task is to classify tigers from images. We choose the model that learns from the cat classification task as the base model. Unfortunately, optimization usually indicates an abstract process. Thus, an intuitive or mathematical definition of relevance is unavailable for the optimization tasks. In online learning, prediction with expert advice [9] considers the situation of unavailable relevance. The problem of this setting is how to apply some pre-trained expert predictors to predict a target task. The core challenge is to select the most relevant expert predictor among the candidates. Prediction with expert advice employs expert predictors to test on the frequent-coming instances. With the labels of instances, the relevant expert predictors can be selected according to the test results.

Without pre-defined relevance between the source and target tasks, this paper proposed the experience adaptation mechanism that adaptively finds relevant directional models to improve the efficiency of experienced DFO.

3 Proposed Method

In this section, we propose the adaptive experienced derivative-free optimization method. We firstly introduce the problem setting, i.e., the source and target optimization tasks. We then briefly discuss the experienced DFO framework. Finally, we show the details of the experience adaptation mechanism, i.e., the core part of the proposed method.

3.1 Source and target tasks

This paper considers an optimization task set $F = \{f_1, f_2, \ldots, f_M\}$, where $\forall i \in \{1, 2, \ldots, M\}, f_i \sim \mathcal{F}$. $\mathcal{F}$ is an underlying task distribution. Since all tasks in F come from the same task distribution, the tasks are different but inner-relevant among each other. Hyper-parameter tuning tasks naturally follow this problem setting. A model can be applied to train on different datasets. Although it is the same model, the best hyper-parameters are totally different due to different datasets. Thus, the tasks are

similar but different. We split F into $F_s = \{f_1, f_2, \ldots, f_{M_s}\}$ and $F_t = \{f_1, f_2, \ldots, f_{M_t}\}$, where $F_s \cup F_t = F$, $F_s \cap F_t = \emptyset$ and $M_s + M_t = M$. F_s is a source task set. $f_{i_s} \in F_s$ is a source task. F_t is a target task set. $f_{i_t} \in F_t$ is a target task. Generally, the source tasks have been optimized by derivative-free optimization methods. In other words, the optimization spaces of source tasks have been explored by DFO methods. The target tasks are unseen now and will be optimized soon. For a learning model, the source task set is formed by hyper-parameter tuning tasks on datasets that we have already searched. The target task set is formed by the hyper-parameter tuning tasks on unseen datasets. Due to the inner relevance among source and target tasks, it is an intuitive idea to apply the experience from the source optimization tasks to accelerate the DFO processes on the target tasks.

3.2 Experienced Optimization

Experienced DFO [18] aims to extract experience from the DFO processes on the source tasks, and apply the extracted experience to accelerate optimization processes on target tasks. Experienced DFO applies three main steps as follows:

- Organizing an experience dataset from the DFO processes on source tasks.
- Training a directional model on the experience dataset.
- Applying the directional model to guide the DFO processes on target tasks.

Experienced derivative-free optimization utilizes the first two steps to extract experience from the DFO processes on the source tasks. The third step is used to accelerate DFO processes on the target tasks. For well introducing experienced DFO, we discuss the details of these three steps.

Organizing experience dataset. We note that derivative-free optimization generates new samples based on some temporary-stored samples. For example, evolutionary algorithms follow a population-based framework. New samples are generated by crossover or mutation according to the samples in the population. Thus, the samples that are used to generate a new sample can be stored as the experience. Previous derivative-free optimization methods generally follow this framework. Let $\boldsymbol{\kappa}$ denote the temporary-stored samples. Let $\boldsymbol{x}'$ denote the new sample. Thus, an instance of experience dataset can be presented by $[\boldsymbol{\kappa}; \boldsymbol{x}']$. Given an evaluation function f_{i_s}, it is easy to give $[\boldsymbol{\kappa}; \boldsymbol{x}']$ a label (in the minimization task) by comparing $\boldsymbol{x}'$ with the best-so-far sample $\tilde{\boldsymbol{x}}$ as follows:

$$\ell\left([\boldsymbol{\kappa}; \boldsymbol{x}']\right) = \begin{cases} 1, & f_{i_s}(\boldsymbol{x}') < f_{i_s}(\tilde{\boldsymbol{x}}); \\ 0, & f_{i_s}(\boldsymbol{x}') \geq f_{i_s}(\tilde{\boldsymbol{x}}). \end{cases} \tag{2}$$

More details of experience dataset organization please refer to [18]. Eq. (2) includes the information of search direction by comparing the new sample with the best-so-far sample. Meanwhile, the quality of search direction can be easily defined as whether the new sample has improved optimization performance. If the new sample is better than the best-so-far performance, it indicates a good search direction and gets a positive label. Otherwise, it is a bad search direction and gets a negative label. Based on this organization, the experience extraction is transformed as a supervised binary classification problem. For a new sample, we can construct a labeled experience data instance. For a DFO process on a source task f_{i_s}, we can organize an experience dataset as $\mathcal{D}^{\text{exp}}_{f_{i_s}} = \{([\boldsymbol{\kappa}_1; \boldsymbol{x}'_1], \ell_1), ([\boldsymbol{\kappa}_2; \boldsymbol{x}'_2], \ell_2), \ldots\}$. From different source tasks, we can obtain some different experience datasets.

Algorithm 1 Experienced DFO framework

Input:
f: The objective function of a target task;
P: The pre-sample size;
N: Total evaluation budget;
Φ: A directional model;
X: The search space;
`Initialize`: Initializing for optimization;
`Sample`: Getting a new sample;
`Update`: Updating optimization with new sample.

Procedure:
1: $B, \tilde{\boldsymbol{x}} = \texttt{Initialize}(\mathcal{U}_X)$
2: **for** $t = 1$ to N **do**
3: $\quad \mathcal{P} = \emptyset$
4: $\quad$ **for** $p = 1$ to P **do**
5: $\quad\quad [\boldsymbol{\kappa}_p; \boldsymbol{x}_p] = \texttt{Sample}(B)$
6: $\quad\quad \mathcal{P} = \mathcal{P} \cup \{[\boldsymbol{\kappa}_p; \boldsymbol{x}_p]\}$
7: $\quad$ **end for**
8: $\quad [\boldsymbol{\kappa}'; \boldsymbol{x}'] = \text{argmax}_{[\boldsymbol{\kappa}; \boldsymbol{x}] \in \mathcal{P}} \Phi([\boldsymbol{\kappa}; \boldsymbol{x}])$
9: $\quad B = \texttt{Update}(B, \boldsymbol{x}', f(\boldsymbol{x}'))$
10: $\quad$ **if** $f(\boldsymbol{x}') < f(\tilde{\boldsymbol{x}})$ **then**
11: $\quad\quad \tilde{\boldsymbol{x}} = \boldsymbol{x}'$
12: $\quad$ **end if**
13: **end for**
14: **return** $\tilde{\boldsymbol{x}}$

Training a directional model. With the labeled experience datasets, it is a supervised learning problem to train models on them. Because instances in the experience datasets indicate the search directions during optimization. We name the training models the directional models. A directional model essentially is a binary classification predictor. In [18], a simple multi-layer perception (MLP) is employed as the training model. Let Φ denote a trained directional model. The output of Φ is a real number that is in $[0, 1]$ and predicts the goodness of the new sample. For example, we get a new sample $\boldsymbol{x}'$ and its previous samples $\boldsymbol{\kappa}$ that generates $\boldsymbol{x}'$. $\Phi([\boldsymbol{\kappa}; \boldsymbol{x}'])$ is a score that means whether the new sample $\boldsymbol{x}'$ indicates a good search direction.

Experienced DFO framework. We obtain a directional model Φ by training a classifier on the experience dataset. Experienced DFO utilizes the directional model by adding a pre-sampling phase to derivative-free optimization methods. Algorithm 1 shows the details of the experienced DFO framework. `Initialize`, `Sample` and `Update` are basic components of derivative-free optimization. In the pre-sampling phase (lines 4 to 7), we obtain some temporary samples. These samples will not be evaluated by the evaluation function immediately. We firstly employ the directional model Φ to predict scores for each of them. Only the sample with the largest predicted value will be evaluated. Owing to the directional model, experienced DFO avoids evaluating samples that have bad search directions, thus improving the optimization efficiency. Combining with classification-based optimization RACOS [15], experienced DFO is implemented as EXPRACOS [18] algorithm.

Discussion. The key that experienced DFO effectively accelerates convergence is that the directional model can correctly predict the goodness of new samples on target tasks. The directional models from similar source tasks can provide more accurate predictions for target tasks. It is very important to select the relevant directional models for target tasks. Unfortunately, the relevance among optimization tasks is usually unavailable for us. Thus, the selection of

the relevant directional model becomes a bottleneck towards high-efficient experienced derivative-free optimization.

3.3 Experience Adaptation Mechanism

In machine learning applications, a learning model is usually applied to several different datasets. Thus, hyper-parameter tuning should be run on these datasets to find the best configuration. All these tasks are source tasks. The experience can be collected during the DFO processes on all source tasks. Based on the experienced DFO framework, we can train a directional model Φ_{i_s} on an experience dataset of a source task f_{i_s}. Thus, we can collect many directional model candidates. Ideally, the experience (directional model) which corresponding source task is most relevant to the target task f_{i_t} should be applied to guide the DFO processes. However, the relevance between source and target tasks is unknown. Thus, EXPRACOS [18] applies the union of all experience datasets to train a directional model. In other words, EXPRACOS utilizes all experience from all source tasks to guide the DFO processes of the target task. Based on the conclusion that is showed in Figure 1, irrelevant directional models have negative impacts on optimization. Hence, the efficiency of EXPRACOS will be possibly injured, because the manually selected experience is unsuitable for the target task. This paper proposed the experienced DFO with experience adaption mechanism that aims to adaptively discover the relevant directional models and apply them to accelerate the DFO processes on target tasks.

For a source task f_{i_s}, we can train a directional model Φ_{i_s} on the experience dataset of f_{i_s}. Thus, we can get a basic directional model set $\mathbf{\Phi} = \{\Phi_1, \Phi_2, \ldots, \Phi_{M_s}\}$ based on the source task set $F_s = \{f_1, f_2, \ldots, f_{M_s}\}$. In experience adaptation, we utilize an ensemble directional model that is weighted sum of all the basic directional models in $\mathbf{\Phi}$ as follows:

$$\bar{\Phi}\left([\boldsymbol{\kappa}; \boldsymbol{x}]\right) = \sum_{i_s=1}^{M_s} w_{i_s} \Phi_{i_s}\left([\boldsymbol{\kappa}; \boldsymbol{x}]\right), \tag{3}$$

where $\boldsymbol{w} = \{w_1, w_2, \ldots, w_{M_s}\}$ are weights of basic directional models. The weights intuitively reflect the relevance between the directional models and the target task. We note that the ground-truth label of an experience instance, i.e., $[\boldsymbol{\kappa}; \boldsymbol{x}]$ can be easily obtained according to the evaluation value of $\boldsymbol{x}$ by Eq. (2). With labeled instances, experience adaptation mechanism follows a simple idea that is adjusting weights according to the testing result of the directional model on the target task.

Because of the property of derivative-free optimization, we sequentially generate the new samples. The labeled instance $\left([\boldsymbol{\kappa}; \boldsymbol{x}], \ell\right)$ can be got one by one. We note that the prediction value of a basic directional model is a real number in $[0, 1]$. For each instance $\left([\boldsymbol{\kappa}; \boldsymbol{x}], \ell\right)$, we employ the weighted ensemble directional model (Eq. (3)) to predict on it to get the prediction value. Noting that each basic directional model has also been applied to give a prediction value for this instance. We define a squared loss to measure the error that a basic directional model makes on this instance: $\left(\Phi_{i_s}\left([\boldsymbol{\kappa}; \boldsymbol{x}]\right) - \ell\right)^2$. The weights of all basic directional models can be adapted according to the loss as follow:

$$w_{i_s} = \exp\left(-\alpha\left(\Phi_{i_s}\left([\boldsymbol{\kappa}; \boldsymbol{x}]\right) - \ell\right)^2\right) w_{i_s}, \tag{4}$$

where α is a hyper-parameter to scale the square loss. Based on the experienced DFO framework (Algorithm 1), we utilize this weight adaptation mechanism on the weighted ensemble directional model.

Algorithm 2 DFO with Adaptive Experience

Input: (extra input than Algorithm 1)
$\mathbf{\Phi} = \{\Phi_1, \Phi_2, \ldots, \Phi_{M_s}\}$: Basic directional model set;
`Normalize`: A normalization procedure.

Procedure:

1: $B, \tilde{\boldsymbol{x}} = \texttt{Initialize}\left(\mathcal{U}_X\right)$
2: $\boldsymbol{w} = \{w_1, w_2, \ldots, w_{M_s}\} = \left\{\frac{1}{M_s}\right\}^{M_s}$
3: **for** $t = 1$ to N **do**
4: $\quad \mathcal{P} = \emptyset$
5: $\quad$ **for** $p = 1$ to P **do**
6: $\quad\quad [\boldsymbol{\kappa}_p; \boldsymbol{x}_p] = \texttt{Sample}\left(B\right)$
7: $\quad\quad \mathcal{P} = \mathcal{P} \cup \{[\boldsymbol{\kappa}_p; \boldsymbol{x}_p]\}$
8: $\quad$ **end for**
9: $\quad [\boldsymbol{\kappa}'; \boldsymbol{x}'] = \operatorname{argmax}_{[\boldsymbol{\kappa}; \boldsymbol{x}] \in \mathcal{P}} \sum_{i_s=1}^{M_s} w_{i_s} \Phi_{i_s}\left([\boldsymbol{\kappa}; \boldsymbol{x}]\right)$
10: $\quad \ell' = \begin{cases} 1, & f\left(\boldsymbol{x}'\right) < f\left(\tilde{\boldsymbol{x}}\right); \\ 0, & f\left(\boldsymbol{x}'\right) \geq f\left(\tilde{\boldsymbol{x}}\right). \end{cases}$
11: $\quad$ **for** $i_s = 1$ to M_s **do**
12: $\quad\quad w_{i_s} = \exp\left(-\alpha\left(\Phi_{i_s}\left([\boldsymbol{\kappa}'; \boldsymbol{x}']\right) - \ell'\right)^2\right) w_{i_s}$
13: $\quad$ **end for**
14: $\quad \boldsymbol{w} = \texttt{Normalize}\left(\boldsymbol{w}\right)$
15: $\quad B = \texttt{Update}\left(B, \boldsymbol{x}', f\left(\boldsymbol{x}'\right)\right)$
16: $\quad$ **if** $f\left(\boldsymbol{x}'\right) < f\left(\tilde{\boldsymbol{x}}\right)$ **then**
17: $\quad\quad \tilde{\boldsymbol{x}} = \boldsymbol{x}'$
18: $\quad$ **end if**
19: **end for**
20: **return** $\tilde{\boldsymbol{x}}$

The derivative-free optimization with experience adaptation mechanism method is presented in Algorithm 2.

Algorithm 2 still follows the pre-sampling mechanism to utilize the directional model. The algorithm starts with optimization initialization. We set the same weights $\frac{1}{M_s}$ for all basic directional models (line 2). Line 5 to 8 is the pre-sampling phase. We utilize the weighted ensemble directional model to predict the goodness score for each temporary sample (line 9). The sample with the highest predicted value will be evaluated by the evaluation function (lines 10 and 15). We adapt weights for all basic directional models during lines 10 to 14. Firstly, we get the ground-truth label for the experience instance $[\boldsymbol{\kappa}'; \boldsymbol{x}']$ (line 10). Then, we adjust the weight for each directional model with the prediction loss according to Eq. (4) (lines 11 to 13). Finally, we apply a normalization procedure to guarantee that $\sum_{i_s=1}^{M_s} w_{i_s} = 1$. With the selected sample and its evaluation value, we update the optimization procedure (line 15) and the best-so-far sample (lines 16 to 18). When the evaluation budget is exhausted, the best-so-far sample will be returned as the optimization solution (line 20).

Discussion. We implement the experience adaptation mechanism based on experienced derivative-free optimization. With the weighted ensemble, all basic directional models from different source tasks are integrated as a directional model. When optimizing on target tasks, we firstly employ the ensemble directional model to select samples that are worth being evaluated. We then test all basic directional models on the labeled experience instance. According to the testing results, the relevant directional models will be selected by adapting the weights. We discuss the experience adaptation mechanism by focusing on the Eq. (4). If basic directional models give a correct prediction, they will obtain a small squared loss. The corresponding weights can get a small discount. But when basic directional models give a wrong prediction, their weights will get a huge

Table 1. Average performance on synthetic target task, i.e., Sphere and Rosenbrock functions with the optimal points $\boldsymbol{x}^* = \{0.10\}^{10}$, $\{0.25\}^{10}$ and $\{0.40\}^{10}$. We set two different directional model sets for ADARACOS and EXPRACOS. In Sphere-Exp, all directional models are from Sphere source tasks. In Mixed-Exp, half of the directional models are from Sphere source tasks and half of the directional models are from Rosenbrock source tasks. The bold number in each row is the best result among the compared methods.

Function & x^*	ADARACOS		EXPRACOS		RACOS	SMAC	Bayes
	Sphere-Exp	Mixed-Exp	Sphere-Exp	Mixed-Exp			
Sphere $\{0.10\}^{10}$	$\mathbf{0.0694} \pm 0.02$	0.0747 ± 0.02	0.1132 ± 0.05	0.1165 ± 0.07	0.7941 ± 0.29	0.0700 ± 0.01	0.4894 ± 0.05
Sphere $\{0.25\}^{10}$	$\mathbf{0.0775} \pm 0.03$	0.1165 ± 0.07	0.1091 ± 0.05	0.1250 ± 0.08	0.8046 ± 0.39	0.2749 ± 0.11	0.4500 ± 0.11
Sphere $\{0.40\}^{10}$	$\mathbf{0.0909} \pm 0.04$	0.1528 ± 0.22	0.1978 ± 0.05	0.2938 ± 0.19	0.8306 ± 0.36	0.6778 ± 0.25	0.3444 ± 0.09
RosenB. $\{0.10\}^{10}$	12.394 ± 2.27	$\mathbf{11.010} \pm 0.69$	13.351 ± 3.39	14.109 ± 4.62	26.903 ± 5.18	17.176 ± 1.15	45.523 ± 17.1
RosenB. $\{0.25\}^{10}$	25.549 ± 9.54	$\mathbf{15.814} \pm 6.62$	26.008 ± 8.93	17.771 ± 3.16	33.065 ± 29.1	43.701 ± 8.37	45.733 ± 13.9
RosenB. $\{0.40\}^{10}$	57.388 ± 20.4	$\mathbf{45.408} \pm 34.8$	93.370 ± 40.7	54.763 ± 22.6	61.955 ± 24.2	99.798 ± 43.6	48.504 ± 12.9

discount. After the normalization step, the weights of basic directional models that make correct predictions will increase relatively. On the contrary, the weights of basic directional models that make wrong predictions will decrease relatively. In this way, relevant directional models that make fewer mistakes on the target task can be adaptively selected with large weights. Irrelevant directional models that make more mistakes can be adaptively omitted with small weights.

4 Experiments

We implement derivative-free optimization with experience adaptation mechanism based on EXPRACOS [18] and name it ADARACOS. We firstly test ADARACOS on synthetic tasks. Because the relevance among synthetic tasks is available, we can easily investigate the effectiveness of ADARACOS. Then, ADARACOS is applied to solve hyper-parameter tuning tasks.

We select experienced DFO method, i.e., EXPRACOS and some the state-of-the-art derivative-free optimization methods including RACOS (code from https://github.com/eyounx/ZOOpt/), SMAC [19] (code from https://github.com/automl/SMAC3) and Bayes [27] as compared methods. SMAC and Bayes are Bayesian optimization methods that are widely used in AutoML tasks. For ADARACOS and EXPRACOS, we apply a simple MLP as the directional model, while network structures depend on the tasks.

4.1 On Synthetic Tasks

We select Sphere and Rosenbrock as basic synthetic functions. Sphere is a convex function as follows:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} (x_i - x_i^*)^2 , \tag{5}$$

Rosenbrock is a non-convex function as follows:

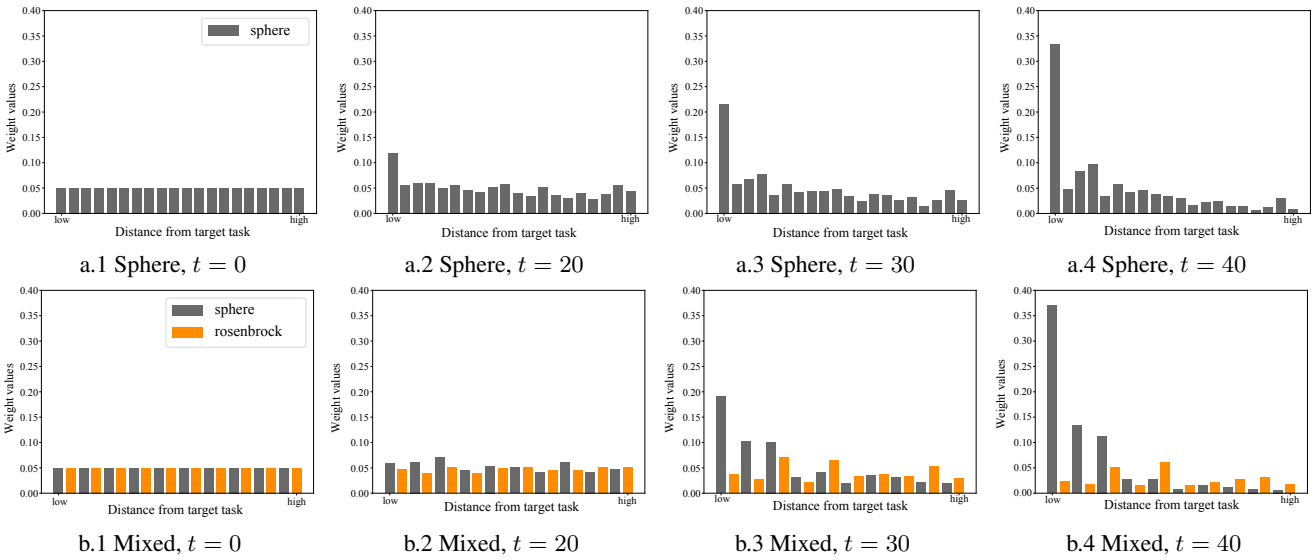
$$f(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_{i+1}^* - (x_i - x_i^*)^2)^2 + (1 - x_i + x_i^*)^2\right] , \tag{6}$$

where $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ is a sample, n is the dimensionality, $\boldsymbol{x}^* = \{x_1^*, x_2^*, \ldots, x_n^*\}$ is the optimal point. The synthetic task is to minimize function value in a constraint region. The relevance among different synthetic tasks can be captured easily (we will discuss it in the next paragraph). With known relevance, we can deeply investigate the effectiveness of ADARACOS on these experiments.

Task settings. We construct the task distribution $\mathcal{F}$ by randomly shifting the optimal point for Sphere and Rosenbrock functions. In other words, functions with different $\boldsymbol{x}^*$ are different tasks. In experiments, we organize two different source task sets: Sphere source set and Mixed source set. The shifting region is $[-0.5, 0.5]^n$. In Sphere source set, we uniformly sample 2000 different $\boldsymbol{x}^*$ for only Sphere function. In the Mixed source set, we uniformly sample 1000 different $\boldsymbol{x}^*$ for Sphere and Rosenbrock functions separately. We set $\boldsymbol{x}^* = \{0.1\}^n$, $\{0.25\}^n$ and $\{0.40\}^n$ for Sphere and Rosenbrock separately to construct 6 target tasks. We set $n = 10$ for all source and target tasks. The relevance between two synthetic tasks can be measured by the Euclidean distance between two optimal points. A small distance indicates strong relevance. A large distance indicates weak relevance. This conclusion has been verified in Figure 1. Intuitively, two different functions share weak relevance. On target tasks, the search space is $[-1, 1]^n$. The evaluation budget is only 50.

Directional model training. We employ RACOS to optimize on source tasks with 500 evaluation budget and independently run for 10 times to collect experience datasets. Then, we sort all experience datasets according to the distance between the optimal points of the source and target task. Every 100 experience datasets in the order form a new dataset. We totally form 20 experience datasets. For the mixed source task set, we form dataset groups according to function types, i.e., 10 experience datasets for Sphere and Rosenbrock respectively. We can totally train 20 MLP directional models on experience datasets for each setting. For EXPRACOS, the directional model is an average weighted ensemble of all 20 MLP models. For ADARACOS, the initialized directional model is the same as EXPRACOS. But the weights will adapt during optimization.

Empirical analysis. We independently repeat experiments on target tasks for 10 times. And the average performances of all compared methods are reported in Table 1. ADARACOS receives the best performances on all 6 target tasks. It indicates that experienced DFO with experience adaptation releases strong power on unseen tasks. We compare experienced DFO methods (ADARACOS, and EXPRACOS) with classical optimization (RACOS, SMAC, and Bayes). Experienced DFO methods generally outperform classical optimization on most of the tasks. SMAC receives good performance on Sphere with $\boldsymbol{x}^* = \{0.10\}^{10}$. Because the initialization point of SMAC is a central point of the search space, i.e., $\{0.0\}^{10}$. This point is very near to the optimal point. Focusing on ADARACOS and EXPRACOS, the results indicate that experience adaptation is effective to avoid the negative impact of irrelevant experience (ADARACOS receives better performances on all 6 target tasks). To verify how ADARACOS select experience, we show the weights changing of directional models in Figure 2. The X-axis from left to right means that the relevance is from strong to weak. The figures indicate that the relevant directional models can be effective selected (the weights in left are becoming larger and larger) and the irrelevant directional models can



a.1 Sphere, $t = 0$ a.2 Sphere, $t = 20$ a.3 Sphere, $t = 30$ a.4 Sphere, $t = 40$

b.1 Mixed, $t = 0$ b.2 Mixed, $t = 20$ b.3 Mixed, $t = 30$ b.4 Mixed, $t = 40$

Figure 2. Illustration of AdaRacos weights changing on the target task, i.e., 10 dimensional Sphere function with the optimal point $\boldsymbol{x}^* = \{0.1\}^{10}$. Weights of Figure a. and b. are from AdaRacos with Sphere experience and Mixed experience at optimization step $t = 0$, 20, 30 and 40. In one Figure, the X-axis from left to right means that the distances of optimal points between the source and target task are from low to high.

be effectively omitted (the weights in right (Figure 2 a.) and in orange (Figure 2 b.) are becoming smaller and smaller).

4.2 On Hyper-Parameter Tuning Tasks

We employ AdaRacos to solve AutoML tasks: the hyper-parameter tuning for an ensemble classifier LightGBM [20] on 40 datasets. We select 11 hyper-parameters that will be tuned for LightGBM including boosting type, learning rate, n-estimators, number of leaves, etc. The evaluation criterion is the F1 score. Thus, the hyper-parameter tuning tasks are maximizing the F1 validation score for LightGBM on 40 datasets.

Task setting. We select 30 datasets (top 30 datasets in Table 2) as source datasets that organize source tasks. And the rest of the 10 datasets (bottom 10 datasets in Table 2) are target datasets that organize target tasks. To collect the experience datasets, we employ Racos to tune hyper-parameters for LightGBM on each source dataset with 300 evaluation budget for 10 independent repeats. All the optimization processes are logged as the experience dataset. Then, we train directional models on these experience datasets. Hence, we obtain all 30 basic directional models for experienced optimization. The directional model of ExpRacos is still the average weighted ensemble of all 30 basic directional models. For AdaRacos, the weights of directional models will be adapted. We test all compared methods on all source and target tasks. we set only 30 evaluation budget for all compared methods on all tasks. Each experiment is independently repeated for 5 times.

Empirical analysis. Table 2 shows the optimization and generalization the F1 scores of all compared methods on all 40 datasets. After hyper-parameter tuning, we can get the best hyper-parameter configuration for each dataset. Then we train the LightGBM with the best configuration on the training dataset. The generalization F1 scores can be obtained by testing the learned LightGBM model on the testing dataset. Compared methods outperform the baseline on most of the datasets. It indicates that hyper-parameter tuning is necessary for machine learning applications. But on datasets G.H.20 and Cylinder, hyper-parameter tuning over-fits on training dataset (baseline is the best). It is caused by the data distribution shifting issue between training and testing datasets. In all 5 compared methods, AdaRacos and ExpRacos are experienced optimization methods. Racos, SMAC and Bayes are basic derivative-free optimization methods that don't use experience. **On source datasets**, we compare ExpRacos with basic optimization methods. ExpRacos even worse than Racos and SMAC (ExpRacos has larger avg. rank). This phenomenon shows that the hyper-parameter tuning tasks share weak relevance even on the same learning model. Thus some irrelevant directional models exist and have negative impacts on ExpRacos. AdaRacos outperform other compared methods on 29/30 datasets and obtain 1.13 avg. rank. It indicates that selecting relevant directional models is necessary for improving hyper-parameter tuning performance. And the proposed experience adaptation mechanism can effectively eliminate the negative impact of irrelevant basic directional models and correctly select the relevant directional models. **On target tasks**, the optimization results show that AdaRacos can effectively transfer optimization experience into unseen tasks (AdaRacos beats other compared methods on all 10 datasets). ExpRacos beats Racos on 9/10 datasets. It shows that experienced optimization is helpful to improve optimization performance on unseen tasks. But comparing them with AdaRacos, AdaRacos significantly improve F1 score on all 10 datasets with only 30 evaluation budget. It indicates that experience adaptation can significantly improve the efficiency of hyper-parameter tuning.

5 Conclusions

Hyper-parameter tuning plays an important role in AutoML. A classical solver is employing derivative-free optimization to discover the hyper-parameter configuration with the best performance. Due to the high evaluation cost, previous hyper-parameter tuning methods suffer from the low-efficient issue, i.e., it spends a long time to find a good enough hyper-parameter configuration. To tackle this issue, experienced DFO approach was proposed to utilize the experience

Table 2. Results of optimization and generalization F1 score for LightGBM hyper-parameter tuning on 40 datasets. All compared methods only have 30 evaluation budget. ADARA. and EXPRA. mean ADARACOS and EXPRACOS. B.L. means baseline that is the F1 score of LightGBM with default hyper-parameters. The first 30 datasets are source datasets. Hyper-parameter tuning tasks on them are the source tasks. The last 10 datasets are target dataset. Hyper-parameter tuning tasks on them are the target tasks. In the table, the numbers with • and ○ are the 1st and 2nd rank performances in compared methods. We analyze the number of 1st/2nd/3rd ranks and average rank (Avg. Rank) among compared methods for source tasks and target tasks separately.

	Dataset	Optimization F1 score on training dataset					Generalization F1 score on testing dataset					B.L.
		ADARA.	EXPRA.	RACOS	SMAC	Bayes	ADARA.	EXPRA.	RACOS	SMAC	Bayes	
Source Datasets	Australian	.9026•	.8817	.8871○	.8871○	.8724	.8966•	.8954○	.8924	.8528	.8922	.8389
	Breast	.9999•	.9999•	.9999•	.9999•	.9999•	.9697•	.9470	.9549○	.9548	.9512	.9402
	Electricity	.7431•	.7398○	.7377	.7345	.7322	.7365	.7387•	.7384○	.5686	.5527	.5492
	Buggy.C.	.8943•	.8693	.8825	.8864○	.8825	.8957•	.8811	.8889	.8842	.8927○	.8552
	CMC	.5860•	.5738	.5754○	.5715	.5741	.5452•	.5152	.5123	.5410○	.4927	.4614
	Contrac.	.5785•	.5762○	.5750	.5715	.5725	.5459•	.5171	.5029	.5444○	.5172	.4614
	Credit.A.	.8938•	.8921	.8927○	.8864	.8895	.8462•	.8461○	.8371	.8273	.8435	.8250
	G.E.2-1000	.5772•	.5433	.5518	.5590○	.5378	.5187•	.4984	.4907	.5187•	.5108	.5368
	G.E.2-200	.7534•	.7040	.7041	.7534•	.7131	.6880•	.6856○	.6676	.6672	.6832	.6187
	G.E.3-20	.5784•	.5623	.5657○	.5485	.5601	.5300•	.5123	.5125	.4612	.5286○	.4936
	G.H.20	.7221•	.7040	.7169	.7187○	.7076	.6250•	.6216	.6220○	.5874	.5819	.6747
	H.V.wo.N.	.5934•	.5875	.5786	.5642	.5888○	.6225•	.5873	.5860	.6145	.6161○	.5977
	H.V.w.N.	.5931•	.5889	.5871	.5910○	.5823	.5785•	.5659○	.5544	.5648	.5393	.5241
	Mfeat.K.	.9713•	.9713•	.9680	.9692	.9693	.9701•	.9661	.9596	.9700○	.9632	.9197
	Mfeat.M.	.7235•	.7212○	.7161	.7140	.7212○	.7204•	.7102	.7146	.7083	.7186○	.6967
	Mfeat.P.	.9722•	.9674	.9685	.9721○	.9661	.9688•	.9684○	.9669	.9655	.9633	.9501
	Mfeat.Z.	.7867•	.7758	.7780○	.7771	.7766	.7867•	.7713	.7729	.7616	.7737○	.7411
	Monk2	.8732•	.6981	.6548	.5774	.7413○	.8197•	.6392	.6559	.6755	.7961○	.6089
	Parity5.	.4948•	.4815	.4847○	.4820	.4837	.4582•	.4480	.4481○	.4451	.4364	.2291
	Pima	.7236•	.7102	.6948	.7173	.7177○	.7555•	.7139	.7179	.6177	.7414○	.6590
	Tic.T.T	.9741•	.9163	.9401	.9741•	.9241	.9838•	.9600	.9756	.9776○	.9770	.7898
	Tokyo.1	.9248•	.9203	.9168	.9243○	.9210	.9430•	.9398○	.9293	.9311	.9382	.9081
	Vehicle	.7943•	.7853	.7911○	.7763	.7814	.7633○	.7562	.7284	.8004•	.7345	.7610
	Wine.Q.R.	.4218•	.3875	.3617	.4021○	.3620	.3621•	.3119○	.3081	.3099	.3097	.2589
	Yeast	.4754•	.4435	.4447○	.4388	.4388	.5033○	.4834	.4685	.5213•	.4726	.4716
	Airlines	.6488•	.6483○	.6467	.6438	.6464	.6530•	.6527○	.6517	.6417	.6467	.5943
	Titanic	.8238•	.8221○	.8187	.8099	.8048	.8364•	.8192	.8149	.8237○	.8091	.8217
	Twonorm	.9749	.9750	.9751	.9782•	.9757	.9792•	.9783	.9776	.9784○	.9768	.9541
	Glass	.7499•	.7088	.7125	.7071	.7497○	.7422•	.6604	.6622	.7178○	.4353	.4345
	Horse.C.	.8724•	.8586	.8602	.8702○	.8616	.8628•	.8339	.8551○	.8510	.8528	.7989
	$1st/2nd/3rd$	29/0/0	2/5/8	1/8/11	4/9/2	1/6/7	27/2/1	1/8/7	0/5/9	3/7/5	0/7/6	-
	$Avg.Rank$	1.13	3.43	3.13	3.27	3.47	1.13	3.27	3.63	3.33	3.60	-
Target Datasets	Messidor	.7548•	.7525○	.7462	.7353	.7505	.6507○	.6474	.6420	.6799•	.6452	.6581
	Adult	.8137•	.8121	.8104	.8128	.8129○	.8070•	.8023	.8030	.8061○	.8049	.7558
	Balance.S.	.5448•	.5399	.5380	.5409○	.5398	.5559•	.5479	.5368	.5515○	.5421	.5294
	CNAE	.9060•	.8924	.8955○	.8946	.8920	.8962•	.8922	.8928	.8897	.8942○	.8227
	Credit.G	.7190•	.7174○	.7052	.7173	.7168	.7058•	.6912	.7009○	.5921	.6454	.6894
	CRX	.8864•	.8854○	.8843	.8690	.8810	.9153•	.9066○	.9034	.8975	.8978	.8974
	Cylinder	.8094•	.8065○	.7487	.7791	.7953	.7935•	.7408	.6654	.7900○	.6897	.7990
	Flare	.7174•	.6816	.6704	.7141○	.6954	.6065•	.5769	.5607	.5007	.5846○	.4518
	Solar.F.	.6724•	.6233	.6131	.6448○	.6195	.6073•	.5996	.6014○	.5709	.5968	.5758
	German	.7498•	.7457	.7331	.7463○	.7432	.6491•	.6300	.6246	.6081	.6378○	.5482
	$1st/2nd/3rd$	10/0/0	0/4/3	0/1/1	0/4/3	0/1/3	9/1/0	0/1/7	0/2/2	1/3/0	0/3/1	-
	$Avg.Rank$	1.00	2.90	4.40	3.10	3.60	1.10	3.20	3.70	3.70	3.30	-

of historical DFO processes to accelerate DFO processes on target tasks. However, the irrelevant experience will make a negative impact on experienced DFO. In this paper, we proposed the *experience adaptation* mechanism. It tests the experience on the target tasks. The relevant experience that makes fewer mistakes will be adaptively selected. And the irrelevant experience that makes more mistakes will be omitted. We implement experience adaptation mechanism based on EXPRACOS and name it ADARACOS. The experiments on synthetic tasks verify that ADARACOS can effectively discover the relevance among tasks. The empirical results of AutoML applications on 40 datasets show that ADARACOS significantly improve the efficiency of hyper-parameter tuning.

REFERENCES

[1] Mathias M Adankon and Mohamed Cheriet, 'Model selection for the LS-SVM. application to handwriting recognition', *Pattern Recognition*, **42**(12), 3264–3270, (2009).
[2] Yoshua Bengio, 'Gradient-based optimization of hyperparameters', *Neural computation*, **12**(8), 1889–1900, (2000).
[3] James Bergstra and Yoshua Bengio, 'Random search for hyper-parameter optimization', *Journal of Machine Learning Research*, **13**, 281–305, (2012).
[4] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl, 'Algorithms for hyper-parameter optimization', in *Advances in Neural Information Processing Systems*, pp. 2546–2554, (2011).
[5] Alain Biem, 'A model selection criterion for classification: Application to hmm topology optimization', in *Proceedings of the 7th International Conference on Document Analysis and Recognition*, pp. 104–108, (2003).
[6] Gary R Bradski, 'Computer vision face tracking for use in a perceptual user interface', (1998).
[7] Pavel B Brazdil, Carlos Soares, and Joaquim Pinto Da Costa, 'Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results', *Machine Learning*, **50**(3), 251–277, (2003).
[8] Andrei Z Broder, 'Computational advertising and recommender systems', in *Proceedings of the ACM Conference on Recommender Systems*, pp. 1–2. ACM, (2008).
[9] Nicolo Cesa-Bianchi and Gabor Lugosi, *Prediction, learning, and games*, Cambridge university press, 2006.
[10] Ronan Collobert and Jason Weston, 'A unified architecture for natural language processing: Deep neural networks with multitask learning', in *Proceedings of the 25th International Conference on Machine Learning*, pp. 160–167, (2008).
[11] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost , Manuel Blum, and Frank Hutter, 'Efficient and robust automated machine learning', in *Advances in Neural Information Processing Systems*, pp. 2962–2970, (2015).
[12] David B Fogel, 'An introduction to simulated evolutionary optimization', *IEEE Transactions on Neural Networks*, **5**(1), 3–14, (1994).
[13] XC Guo, JH Yang, CG Wu, CY Wang, and YC Liang, 'A novel LS-SVMs hyper-parameter selection based on particle swarm optimization', *Neurocomputing*, **71**(16), 3211–3215, (2008).
[14] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos, 'Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)', *Evolutionary Computation*, **11**(1), 1–18, (2003).
[15] Yi-Qi Hu, Hong Qian, and Yang Yu, 'Sequential classification-based optimization for direct policy search.', in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pp. 2029–2035, (2017).
[16] Yi-Qi Hu, Yang Yu, and Jun-Da Liao, 'Cascaded algorithm-selection and hyper-parameter optimization with extreme-region upper confidence bound bandit', in *Proceeding of the 28th International Joint Conference on Artificial Intelligence*, (2019).
[17] Yi-Qi Hu, Yang Yu, Wei-Wei Tu, Qiang Yang, Yuqiang Chen, and Wenyuan Dai, 'Multi-fidelity automatic hyper-parameter tuning via transfer series expansion', in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, (2019).
[18] Yi-Qi Hu, Yang Yu, and Zhi-Hua Zhou, 'Experienced optimization with reusable directional model for hyper-parameter search.', in *Proceeding of the 27th International Joint Conference on Artificial Intelligence*, pp. 2276–2282, (2018).
[19] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown, 'Sequential model-based optimization for general algorithm configuration.', *LION*, **5**, 507–523, (2011).
[20] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu, 'Lightgbm: A highly efficient gradient boosting decision tree', in *Advances in Neural Information Processing Systems*, pp. 3146–3154, (2017).
[21] SB Kotsiantis, Dimitris Kanellopoulos, and PE Pintelas, 'Data preprocessing for supervised leaning', *International Journal of Computer Science*, **1**(2), 111–117, (2006).
[22] Fei-Fei Li, Fergus Rob, and Perona Pietro, 'One-shot learning of object categories', *IEEE Transactions on Pattern analysis and Machine Intelligence*, **28**(4), 594–611, (2006).
[23] Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar, 'Hyperband: A novel bandit-based approach to hyperparameter optimization', *Journal of Machine Learning Research*, **18**, 185:1–185:52, (2017).
[24] Marius Lindauer and Frank Hutter, 'Warmstarting of model-based algorithm configuration', in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pp. 1355–1362, (2018).
[25] Rémi Munos, 'Optimistic optimization of a deterministic function without the knowledge of its smoothness', in *Advances in Neural Information Processing Systems*, pp. 783–791, (2011).
[26] Chao Qian, Yang Yu, and Zhi-Hua Zhou, 'Pareto ensemble pruning.', in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pp. 2935–2941, (2015).
[27] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando De Freitas, 'Taking the human out of the loop: A review of Bayesian optimization', *Proceedings of the IEEE*, **104**(1), 148–175, (2015).
[28] Jake Snell, Kevin Swersky, and Richard Zemel, 'Prototypical networks for few-shot learning', in *Advances in Neural Information Processing Systems*, pp. 4077–4087, (2017).
[29] Jasper Snoek, Hugo Larochelle, and Ryan P Adams, 'Practical bayesian optimization of machine learning algorithms', in *Advances in Neural Information Processing Systems*, pp. 2951–2959, (2012).
[30] Kevin Swersky, Jasper Snoek, and Ryan P Adams, 'Multi-task bayesian optimization', in *Advances in Neural Information Processing Systems*, pp. 2004–2012, (2013).
[31] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown, 'Auto-weka: Combined selection and hyperparameter optimization of classification algorithms', in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 847–855, (2013).
[32] Ricardo Vilalta and Youssef Drissi, 'A perspective view and survey of meta-learning', *Artificial intelligence review*, **18**(2), 77–95, (2002).
[33] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al., 'Matching networks for one shot learning', in *Advances in Neural Information Processing Systems*, pp. 3630–3638, (2016).
[34] Xi-Zhu Wu, Song Liu, and Zhi-Hua Zhou, 'Heterogeneous model reuse via optimizing multiparty multiclass margin', in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 6840–6849, (2019).
[35] Quanming Yao, Mengshuo Wang, Hugo Jair Escalante, Isabelle Guyon, Yi-Qi Hu, Yu-Feng Li, Wei-Wei Tu, Qiang Yang, and Yang Yu, 'Taking human out of learning applications: A survey on automated machine learning', *arXiv preprint arXiv:1810.13306*, (2018).
[36] Yang Yu, Hong Qian, and Yi-Qi Hu, 'Derivative-free optimization via classification.', in *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pp. 2286–2292, (2016).
[37] Yang Yu, Xin Yao, and Zhi-Hua Zhou, 'On the approximation ability of evolutionary optimization with application to minimum set cover', *Artificial Intelligence*, **180**, 20–33, (2012).
[38] Peng Zhao, Le-Wen Cai, and Zhi-Hua Zhou, 'Handling concept drift via model reuse.', *Machine Learning*, (2019).