Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Experienced Optimization with Reusable Directional Model for Hyper-Parameter Search*

Yi-Qi Hu, Yang Yu, Zhi-Hua Zhou
National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
{huyq, yuy, zhouzh}@lamda.nju.edu.cn

Abstract

Hyper-parameter selection is a crucial yet difficult issue in machine learning. For this problem, derivative-free optimization has being playing an irreplaceable role. However, derivative-free optimization commonly requires a lot of hyper-parameter samples, while each sample could have a high cost for hyper-parameter selection due to the costly evaluation of a learning model. To tackle this issue, in this paper, we propose an *experienced optimization* approach, i.e., learning how to optimize better from a set of historical optimization processes. From the historical optimization processes on previous datasets, a directional model is trained to predict the direction of the next good hyper-parameter. The directional model is then reused to guide the optimization on learning new datasets. We implement this mechanism within a state-of-the-art derivative-free optimization method SRACOS, and conduct experiments on learning the hyper-parameters of heterogeneous ensembles and neural network architectures. Experimental results verify that the proposed approach can significantly improve the learning accuracy within a limited hyper-parameter sample budget.

1 Introduction

Machine leaning has become a main driving force for the development of artificial intelligence. Open source packages such as Weka [Hall *et al.*, 2009] and Scikit-learn [Pedregosa *et al.*, 2011] with a variety of learning algorithms provide users an easy way to apply machine learning. But the algorithm performance highly depends on the hyper-parameter setting. For example, the performance of neural network model is sensitive to the architecture, learning rate, and so on [Orr and Müller, 2012]. The hand-crafted hyper-parameter tuning requires expert domain knowledge and tremendous manpower. Thus, automatic machine leaning (autoML), with the aim of choosing the best hyper-parameter without human interference, is appealing. AutoML is usually studied as algorithm selection problems [Adankon and Cheriet, 2009; Biem, 2003; Brazdil *et al.*, 2003], hyper-parameter tuning problems [Bengio, 2000; Bergstra and Bengio, 2012], and more recently, as the *combined algorithm selection and hyper-parameter optimization problem* (CASH) [Thornton *et al.*, 2013; Feurer *et al.*, 2015]. Despite the different formulations, derivative-free optimization methods are often employed to solve them. Unlike gradient-based optimization that relies on derivatives, these methods work through sampling from the search space. Derivative-free optimization has shown great potential in solving sophisticated problems, such as non-differentiable and non-convex functions.

However, derivative-free optimization commonly needs to sample a lot of hyper-parameters before finding a good one. This is due to the sampling nature of these methods, i.e., a sufficient exploration of the search space by many trial-and-errors is inevitable. The low sample efficiency issue is even severer in autoML tasks. Evaluation of a sampled hyper-parameter in autoML is highly expensive because of, e.g., the high time consumption of the k-fold cross validation on large datasets. Therefore, improving the sample effectiveness of derivative-free optimization, and thus reducing the required sample amount, is a key issue for autoML.

In this paper, we propose an *experienced optimization* approach to improve the sample effectiveness, which extracts information from previous tasks to guide the optimization on new tasks. Although the tasks can be various, we observed that the search direction of different optimization processes can be aligned. Therefore, we propose to learn a directional model from a set of experienced optimization tasks, and reuse the directional model to guide the search in new optimization tasks. The directional model can help reduce unnecessary explorations and thus save wasted samples. We then incorporate the idea into the recently proposed derivative-free optimization method SRACOS [Hu *et al.*, 2017], and implement the EXPSRACOS approach. Experiments on synthetic functions, hyper-parameter search for heterogeneous ensembles, and architecture design for deep neural networks disclose that the directional model can effectively capture the search direction information, and help EXPSRACOS improve the sample effectiveness significantly within limited sample budget.

*This work is supported by the National Key R&D Program of China (2018YFB1004300), NSFC (61751306), Jiangsu SF (BK20160066), and Collaborative Innovation Center of Novel Software Technology and Industrialization. Yang Yu is the corresponding author.

2 Background

In this paper, we focus on the hyper-parameter optimization problems. Let $\mathbf{C}$ denote a machine learning model and $\boldsymbol{\delta} \in \boldsymbol{\Delta}$ denote a hyper-parameter setting, where $\boldsymbol{\Delta}$ is its hyper-parameter space. Taking k-fold cross validation as the evaluation criterion of $\mathbf{C}^{\boldsymbol{\delta}}$, the evaluation can be formulated as follows:

$$f(\boldsymbol{\delta}) = \frac{1}{k}\sum_{i=1}^{k} \mathcal{L}(\mathbf{C}^{\boldsymbol{\delta}}, \mathcal{D}^i_{\text{train}}, \mathcal{D}^i_{\text{valid}}),$$

where $\mathcal{L}(\cdot)$ is a loss function, $\mathcal{D}^i_{\text{train}}$ and $\mathcal{D}^i_{\text{valid}}$ are the training and validation data in the i-th fold. The hyper-parameter optimization problem is to find the best hyper-parameter setting as $\boldsymbol{\delta}^* = \text{argmin}_{\boldsymbol{\delta}\in\boldsymbol{\Delta}} f(\boldsymbol{\delta})$.

Since the hyper-parameter optimization objectives are often non-differentiable, they are commonly solved by derivative-free optimization. Derivative-free optimization is designed for sophisticated problems such as non-convex, non-continuous, and the like. We consider the minimization problems, and let X denote the compact searching space and $f\colon X \to \mathbb{R}$ denote the objective function. The minimization problem can be presented as to find $\boldsymbol{x}^* \in X$ s.t. $\forall \boldsymbol{x} \in X\colon f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$. The processes of derivative-free optimization algorithms [Shahriari *et al.*, 2015; Munos, 2011; Yu *et al.*, 2016] share the same idea of optimization from samples. Without any gradient information, derivative-free optimization explores objective space according to the samples and their evaluations. The most important part of derivative-free optimization is how to generate new samples. For example, Bayesian optimization [Shahriari *et al.*, 2015] connects the searching space and the function values with a surrogate function modeled by the Gaussian process (GP), and chooses the sample with the best acquisition function value which is based on the GP model. However, autoML often faces integer and categorical searching spaces. SMAC [Hutter *et al.*, 2011] was proposed to adapt this situation by replacing GP with the random forest model during building the surrogate function. SRACOS [Hu *et al.*, 2017] is a recently proposed classification-based derivative-free optimization method. It shows outstanding efficiency and scalability in some applications

Employing derivative-free optimization approaches, autoML has achieved successful outcomes. AutoWeka and AutoSklearn, open source autoML projects using SMAC, have received outstanding performance in some autoML competitions. More recent studies focused on searching hyper-parameters for deep neural networks. In [Baker *et al.*, 2017], researchers employed deep reinforcement learning to model architecture designing process. The architectures of layers was considered as actions. The performance of designed architecture was considered as the reward. An reinforcement learning algorithm was then employed to train a policy to predict the actions.

Despite existing successes, evaluation of hyper-parameters in autoML tasks is commonly resource consuming, due to the large training data, repeated evaluations, large models, etc. To reduce the total sample cost, some previous studies have considered how to speed up the convergence by reusing the historical optimization experience for optimizing new tasks. Most of these studies are based on Bayesian optimization, e.g., [Swersky *et al.*, 2013; Lindauer and Hutter, 2018]. To transfer the experience, the previous Bayesian process models can be directly reused into new optimization processes. For example, in [Lindauer and Hutter, 2018], historical models are linearly combined as a warmstart for new tasks. In our work, we don't reuse the historical optimization models directly as they may not be well aligned across different optimization tasks, instead, we learn and reuse a directional model for the optimization process that can be better aligned and transferable.

3 Proposed Method

This section describes the proposed experienced optimization approach. We firstly present the overall framework, and then present the implementation details of this framework based on a state-of-the-art derivative-free optimization algorithm.

3.1 Overall Framework

This paper considers an optimization problem set $F = \{f\}$, where $f \sim \mathcal{F}$ and $\mathcal{F}$ is an underlying problem distribution. For example, optimizing hyper-parameters for a certain learning model on different datasets can be seen as a problem distribution. Given F_e as the experienced problem set, the experienced optimization aims at optimizing future problems $F_t = F - F_e$ more efficiently.

We have observed that, in different optimization processes, search direction can be aligned and can generalize across difference optimization processes. Therefore, our approach learns a directional model from the historical optimization processes, and guides new optimization processes.

The framework of the experienced optimization with directional model is presented in Algorithm 1. It mainly consists of three steps:

- Organizing the experience dataset $\mathcal{D}_{F_e}$ from the historical optimization processes (line 1 to 4). Derivative-free optimization methods often store some historical samples during optimization. The instances in $\mathcal{D}_{F_e}$ can be

Input:
F_e, F_t: Experienced and target problem sets;
$\mathcal{A}$: The optimization approach;
`Log&Assign`: Log and assign experience dataset;
`Train`: Train directional model.
Procedure:
1: $\mathcal{D}_{F_e} = \emptyset$
2: **for** $f \in F_e$ **do**
3: $\quad \mathcal{D}^f_{F_e} = \texttt{Log\&Assign}(\mathcal{A}, f)$
4: $\quad \mathcal{D}_{F_e} = \mathcal{D}_{F_e} \cup \mathcal{D}^f_{F_e}$
5: **end for**
6: $\Phi = \texttt{Training}(\mathcal{D}_{F_e})$
7: **for** $f \in F_t$ **do**
8: $\quad \boldsymbol{x}^*_f = \mathcal{A}(f, \Phi)$
9: **end for**

Algorithm 1: Framework of Experienced Optimization by Directional Model

extracted from the snippet of the stored samples. For each instance, we extract features and assign a label that is the direction to a later found better solution. In line 3, the `Log&Assign` sub-process is used to collect the labeled instances from the optimization processes.

- Learning directional model Φ on D_{F_e} (line 6). With the labeled experience dataset, training Φ is a supervised learning problem. Note that Φ can be trained by any state-of-the-art learning algorithm.
- Utilizing Φ to predict the direction of the next sample during optimizing in new problems (line 7 to 9). The directional models can be embedded in optimization method by adding a pre-sampling step, which generates a set of candidate samples. Among the candidate samples, the one most close to the direction predicted by Φ is selected as the next sample.

3.2 EXPSRACOS: Experienced SRACOS

We implement the idea within SRACOS, and propose the EXPSRACOS method for experienced optimization.

SRACOS. To explain EXPSRACOS clearly, it is necessary to introduce SRACOS briefly. SRACOS also follows the trial-error process. Throughout the optimization process, SRacos maintains two set of solutions B^+ and B^-. B^+ contains the best k solutions, and B^- contains a collection of the rest solutions. They are initialized from randomly sampled of solutions. In every iteration of SRACOS, it learns a classifier from B^+ as the positive samples and B^- as the negative samples. It then samples a new solution from the positive area of the classifier, and updates B^+ or B^- according to the evaluation result of the new solution. More details of SRACOS can be found in [Hu *et al.*, 2017].

Collect experience dataset. We extract the experience dataset from SRACOS optimization processes on previous tasks. In the t-th iteration, the sampling area of SRACOS is learned on $(\boldsymbol{x}_t^+, B_t^-)$ during optimizing, where $\boldsymbol{x}_t^+ \in B_t^+$. B_t^- stores the remaining solutions. Therefore, we organize the depending dataset $(\boldsymbol{x}_t^+, B_t^-)$ as a context matrix:

$$\boldsymbol{\kappa}_t = \begin{bmatrix} \boldsymbol{x}_{t,1}^- - \boldsymbol{x}_t^+ \\ \boldsymbol{x}_{t,2}^- - \boldsymbol{x}_t^+ \\ \vdots \\ \boldsymbol{x}_{t,m}^- - \boldsymbol{x}_t^+ \end{bmatrix}, \text{where } \boldsymbol{x}_{t,i}^- \in B_t^-, i = 1, 2, \ldots, m.$$

Note that $\boldsymbol{\kappa}_t$ is a $m \times n$ matrix, where $m = |B_t^-|$ and n is dimension size of searching space. Each row of $\boldsymbol{\kappa}_t$ is a sample of B_t^- which is centralized around $\boldsymbol{x}_t^+$. With the centralization, the search behavior at different time and in different optimization tasks can be aligned, thus the model trained on $\boldsymbol{\kappa}$ could be reused to other problems more easily.

Suppose $\boldsymbol{x}_t'$ is the generated sample using the context matrix $\boldsymbol{\kappa}_t$. We combine the two to compose an experience instance, $[\boldsymbol{\kappa}_t; \boldsymbol{x}_t']$. We assign a label to this instance according to the quality of $\boldsymbol{x}_t'$. If it improves the objective value, the label is positive, otherwise, the label is negative, i.e.,

$$\ell_t\left([\boldsymbol{\kappa}_t; \boldsymbol{x}_t']\right) = \begin{cases} 1, & f(\boldsymbol{x}_t') < f(\tilde{\boldsymbol{x}}_t); \\ 0, & f(\boldsymbol{x}_t') \geq f(\tilde{\boldsymbol{x}}_t). \end{cases}$$

Input:
f: Objective function to be minimized;
P: The number of pre-sampling;
r: The number of samples in initialization;
N: The evaluation budget;
Φ: Directional model;
`Initialize`: Initialization steps;
`Sample`: Get a new sample by SRACOS.

Procedure:
1: $(B^+, B^-, (\tilde{\boldsymbol{x}}, \tilde{y})) = \texttt{Initialize}(\mathcal{U}_X)$
2: **for** $t = r + 1$ to N **do**
3: $\quad \mathcal{P} = \emptyset$
4: $\quad$ **for** $i = 1$ to P **do**
5: $\quad\quad (\boldsymbol{\kappa}, \boldsymbol{x}) = \texttt{Sample}(B^+, B^-, \lambda, \mathcal{C})$
6: $\quad\quad p = \Phi(\boldsymbol{\kappa}, \boldsymbol{x})$
7: $\quad\quad \mathcal{P} = \mathcal{P} \cup ((\boldsymbol{\kappa}, \boldsymbol{x}), p)$
8: $\quad$ **end for**
9: $\quad ((\hat{\boldsymbol{\kappa}}, \hat{\boldsymbol{x}}), \hat{p}) = \text{argmax}_{((\boldsymbol{\kappa}, \boldsymbol{x}), p) \in \mathcal{P}}\, p$
10: $\quad \hat{y} = f(\hat{\boldsymbol{x}})$
11: $\quad (B^+, B^-) = \texttt{Update}\left((\hat{\boldsymbol{x}}, \hat{y}), B^+, B^-\right)$
12: $\quad (\tilde{\boldsymbol{x}}, \tilde{y}) = \text{argmin}_{(\boldsymbol{x}, y) \in B^+ \cup \{(\tilde{\boldsymbol{x}}, \tilde{y})\}}\, y$
13: **end for**
14: **return** $(\tilde{\boldsymbol{x}}, \tilde{y})$

Algorithm 2: Experienced SRACOS (EXPSRACOS)

where $\tilde{\boldsymbol{x}}_t$ is the best-so-far solution. Putting all experience instances into a dataset, we obtain the experience dataset as $\mathcal{D}_{F_e} = \{([\boldsymbol{\kappa}_1; \boldsymbol{x}_1'], \ell_1), ([\boldsymbol{\kappa}_2; \boldsymbol{x}_2'], \ell_2), \ldots\}$.

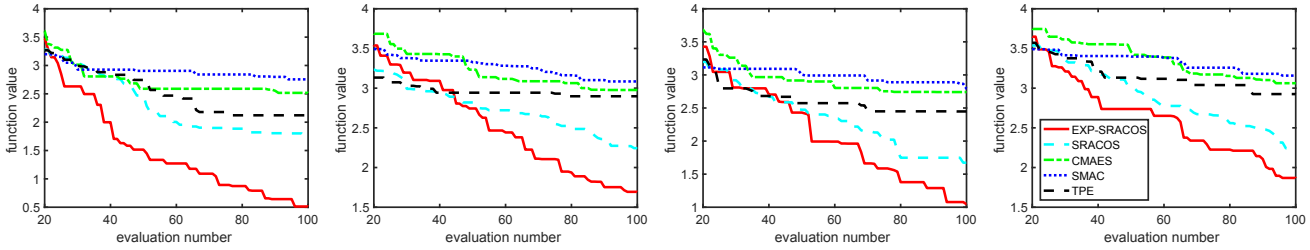
Training directional model. Because the $\mathcal{D}_{F_e}$ is a labeled dataset, any classifier can be employed. From the previous subsection, we notice that an instance in $\mathcal{D}_{F_e}$ consists of two parts a matrix $\boldsymbol{\kappa}$ and a vector $\boldsymbol{x}'$. Thus, we should re-organize the instance from $[\boldsymbol{\kappa}; \boldsymbol{x}']$ by reshaping $\boldsymbol{\kappa}$ as a vector and combining it with $\boldsymbol{x}'$. In our work, we just apply a simple multilayer perceptron (MLP) as the directional model Φ. By the last layer of Φ, we map the directional model output to [0,1] to reflect the goodness of the sample.

EXPSRACOS. EXPSRACOS follows the algorithm framework of SRACOS. Before evaluating a sample, a pre-sampling step is added. It generates a set of samples which will be filtered by the directional model. Algorithm 2 presents the pseudo code of EXPSRACOS. Line 1 is the initialization step. Line 4 to 8 is pre-sampling process. The directional model Φ will predict goodness of each pre-sampling solution (line 6). Only the solution with highest prediction will be evaluated (line 9 and 10). And then the solution and its evaluation value will be used to update (B^+, B^-). An implementation of EXPSRACOS can be found at `https://github.com/eyounx/ExpSRacos`.

Why EXPSRACOS works. We discuss why experienced optimization works under the following two assumptions:

- We assume that optimization tasks $f \in F_e$ and F_t share the same search space X;
- For any two instances $([\boldsymbol{\kappa}_\mathbf{a}; \boldsymbol{x}_\mathbf{a}'], \ell_\mathbf{a})$ and $([\boldsymbol{\kappa}_\mathbf{b}; \boldsymbol{x}_\mathbf{b}'], \ell_\mathbf{b})$ in $\mathcal{D}_{F_e}$, we assume $\ell_\mathbf{a} = \ell_\mathbf{b}$ if $[\boldsymbol{\kappa}_\mathbf{a}; \boldsymbol{x}_\mathbf{a}'] = [\boldsymbol{\kappa}_\mathbf{b}; \boldsymbol{x}_\mathbf{b}']$.

The centralization process of $\boldsymbol{\kappa}$ makes sure that the second as-



(a) $x_i^* = 0.04, n = 10$ (b) $x_i^* = 0.04, \quad n = 20$ (c) $x_i^* = 0.2, \quad n = 10$ (d) $x_i^* = 0.2, \quad n = 20$

Figure 1: The convergence rate on Ackley functions with $x_i^* = 0.04$ and 0.2, $n = 10$ and 20, where $i = 1, 2, \ldots, n$.

sumption can be met in most cases. The directional model Φ of historical optimization processes is to predict whether new sample $\boldsymbol{x}$ is good. Based on these two assumptions, Φ can be reused to predict the goodness of new samples on new tasks. In EXPSRACOS, only the sample with the highest prediction of Φ will be evaluated. Compared with SRACOS that wastes many samples for exploration, EXPSRACOS avoids evaluating many inferior samples.

4 Experiments

In this section, we empirically investigate the effectiveness of the proposed experienced optimization framework (using its implementation EXPSRACOS) on some tasks. We firstly study EXPSRACOS on a synthetic function. Then, EXPSRACOS is employed to tackle two autoML tasks, tuning hyper-parameters on ensemble classifier and optimizing architecture of deep neural network. In these tasks, the directional models are all structures of MLP. But the detail settings are slightly different according to tasks.

We choose the state-of-the-art derivative-free optimization methods to compare with EXPSRACOS including SRACOS (code from `https://github.com/eyounx/ZOOpt`), CMAES [Hansen *et al.*, 2003] (code from `https://pypi.python.org/pypi/cma`), SMAC (code from `https://github.com/automl/SMAC3`) and TPE [Bergstra *et al.*, 2011] (code from `http://jaberg.github.io/hyperopt/`).

4.1 On Synthetic Functions

We firstly compare EXPSRACOS with SRACOS, CMAES, SMAC and TPE on a highly non-convex function Ackley: $f(\boldsymbol{x}) = -20e^{(-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - x_i^*)^2})} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi(x_i - x_i^*)} + e + 20$. $\boldsymbol{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ denotes the optimal solution, where n is dimension size. The search domain is $X = [-1, 1]^n$. Because Ackley function has many local optima, it provides an extreme situation to test the effectiveness of the experienced optimization.

Problem settings. The problem distribution $\mathcal{F}$ is constructed by randomly shifting $\boldsymbol{x}^*$ of Ackley function. The relationship among different functions depends on the size of the space X^{shift}. If it defines a tiny space, problems are more similar with each other. We investigate how X^{shift} influences the performance of EXPSRACOS by setting $X^{shift} = [-0.1, 0.1]^n$ and $[-0.5, 0.5]^n$, where we set $n = 10$ and 20.

Settings X^{shift}	n	EXPSRACOS	SRACOS	CMAES	SMAC	TPE
$[-0.1, 0.1]^n$	10	**1.67**±0.42	2.37±0.36	2.85±0.36	3.01±0.21	2.69±0.26
$[-0.1, 0.1]^n$	20	**2.76**±0.26	2.87±0.20	3.41±0.25	3.31±0.13	3.09±0.17
$[-0.5, 0.5]^n$	10	**2.26**±0.41	2.42±0.36	2.93±0.39	2.95±0.25	2.71±0.29
$[-0.5, 0.5]^n$	20	**2.89**±0.27'	2.98±0.22	3.56±0.28	3.35±0.18	3.18±0.18

Table 1: Average target-problem performance for each problem setting of Ackley function. X^{shift} is the optimal solution shifting region and n denotes the search space dimension. A number in bold means the best function value in its setting.

Considering combination, there are 4 different problem distributions on synthetic function experiments.

Training Φ for EXPSRACOS. For each X^{shift} setting, we sample a batch of problems to form F_e and F_t, where $F_e \cap F_t = \emptyset$. Because $X^{shift} = [-0.5, 0.5]^n$ is much larger than $X^{shift} = [-0.1, 0.1]^n$, we set $|F_e| = 200$ for $[-0.1, 0.1]^n$ and $|F_e| = 400$ for $[-0.5, 0.5]^n$. We collect experience dataset $\mathcal{D}_{F_e}$ by applying SRACOS to optimize $f \in F_e$ with 400 evaluation budget and repeat running for 5 times. We set $|F_t| = 100$ target problems for all 4 settings. Because of the inferior sample effectiveness of SRACOS, the dataset $\mathcal{D}_{F_e}$ is highly imbalanced in classes. The number of positive instances is much less than negative instances. The upsampling strategy is used to adjust the balance, on which the directional model is trained.

On convergence rate. We test the convergence rate of each method on Ackley with a specific optimal solution $\boldsymbol{x}^*$ and the budget of 100 samples. We set $\boldsymbol{x}^* = (0.2, 0.2, \ldots)$ for $X^{shift} = [-0.5, 0.5]^n$ and $\boldsymbol{x}^* = (0.04, 0.04, \ldots)$ for $X^{shift} = [-0.1, 0.1]^n$. Ackley functions with these two $\boldsymbol{x}^*$ are not in F_e, so it can verify the transfer performance of Φ for each problem distribution. Each method runs for five times independently on each function. In Figure 1, two settings of the optimal solutions are presented. It can be observed that the reuse of the directional model is generally less affective for the larger change of the optimal solution. Meanwhile, the convergence speed of EXPSRACOS is the fastest on all settings. Especially compared with SRACOS, in Figure 1 (a), (b), the convergence of EXPSRACOS is significantly faster.

On average performance. We report the average performance of the compared methods on all problems in F_t with 50 evaluation budget. Each method runs five times on each problem independently. The average performance is presented in Table 1. EXPSRACOS shows the best performance among all

	Dataset	Optimization performance on training samples					Generalization performance on testing samples					Default $\mathbf{C_V}$
		ExpSRacos	SRacos	SMAC	TPE	Best C	ExpSRacos	SRacos	SMAC	TPE	Best C	
Source datasets	Annealing	.0401•	.0460○	.0877	.0522	.0590	.0000•	.0100○	.0200	.0100○	.0100○	.0400
	Arcene	.1386○	.1656	.2395	.1478	.1257•	.1866○	.2266	.2600	.3000	.1600•	.3700
	Balance S.	.0822•	.0848○	.1142	.1042	.1063	.0834•	.2354	.1666	.1904	.0968○	.2300
	Banknote	.0000•	.0000•	.0000•	.0018	.0000•	.0000•	.0000•	.0000•	.0000•	.0000•	.0000
	Breast C. W.	.0357•	.0405○	.0429	.0538	.0466	.0638○	.0685	.0780	.0567•	.0936	.0638
	Car	.0260○	.0231•	.0744	.0758	.1439	.3410•	.3786	.3526	.3757	.3421○	.3872
	Chess	.0089•	.0096○	.0462	.0497	.0755	.0671•	.1130○	.1312	.1453	.1375	.1109
	CMC	.4315•	.4355○	.4451	.4462	.4520	.4155•	.4346○	.4459	.4391	.4391	.4290
	CNAE9	.0439•	.0444	.0477	.0441○	.0454	.0416•	.0493	.0416•	.0509	.0527	.0555
	Credit	.0889•	.0938○	.1243	.1244	.0981	.2571•	.2757	.2661	.2589○	.3741	.2517
	Cylinder	.1370•	.1506○	.3827	.3189	.4100	.4036○	.3957•	.4220	.4220	.4128	.4128
	Drug C.	.2177○	.2173•	.2270	.2250	.2270	.2269•	.2321○	.2321○	.2321○	.2321○	.2321
	Ecoli	.1459•	.1466○	.3029	.1887	.1691	.1301	.1000•	.1714	.1142○	.1342	.3000
	Flag	.3121•	.3233○	.3353	.3252	.3576	.3317•	.4065	.4146	.4146	.3658○	.3902
	German C.	.2312•	.2320○	.2425	.2500	.2425	.2299•	.2583	.2450	.2500	.2400○	.2700
	Glass	.2071•	.2104○	.2595	.2440	.2983	.4444○	.4740	.4888	.4666	.3377•	.4888
	Horse C.	.1168•	.1220	.1299	.1198○	.1431	.1470	.1274•	.1323○	.1470	.1558	.1617
	Image S. C.	.0666•	.0698○	.0904	.0857	.1000	.0564•	.0634	.0610○	.0614	.0834	.0688
	Iris	.0166•	.0222	.0250	.0250	.0166•	.0333	.0222○	.0000•	.0333	.0333	.0333
	Madelon	.2385•	.2610	.2840	.2655	.2565○	.2400•	.2577	.2983	.2483	.2410○	.3266
	Messidor	.2500•	.2576	.2933	.2781	.2510○	.2597○	.2496•	.2683	.2597○	.2597○	.2900
	Seismic	.0624○	.0617•	.0763	.0677	.0657	.0676○	.2121	.0793	.2030	.0657•	.0831
	WDBC	.0351•	.0359○	.0396	.0440	.0440	.0573•	.0695	.0608○	.0869	.0695	.0782
	WPBC	.1715•	.1909	.1833○	.1837	.1971	.1376•	.2032	.1951	.1951	.1463○	.2439
	1st/2nd/3rd	20/4/0	4/13/5	1/1/6	0/2/9	3/2/4	15/6/3	5/5/5	3/4/7	2/5/7	4/9/4	-
	Avg. rank	1.1667	2.2083	3.8750	3.6250	3.6667	1.5000	2.9583	3.2500	3.1667	2.7917	-
Target datasets	**Mushroom**	.0000•	.0000•	.0001	.0009	.0407	.1511○	.1619	.1642	.1623	.1023•	.1642
	Occupancy	.0030•	.0032○	.0147	.0085	.0391	.0399○	.0468	.0816	.0392•	.0768	.0509
	Spambase	.0543•	.0557○	.0750	.0782	.0801	.0588•	.0684○	.0716	.0694	.0738	.0966
	Statlog S.	.0205•	.0216○	.0233	.0238	.0254	.0238•	.0339	.0281○	.0432	.0311	.0389
	Wilt	.0165•	.0172○	.0204	.0204	.0202	.0314○	.0146•	.1120	.0420	.1060	.1000
	Wine Q. R.	.3241○	.3177•	.4217	.4209	.4258	.4409•	.4813	.4720	.4751	.4565○	.4472
	Yeast	.4025•	.4103○	.4556	.4561	.4640	.4186•	.4313○	.4417	.4481	.4320	.4983
	Gisette	.0190•	.0203○	.0216	.0210	.0270	.0180•	.0259	.0214	.0210○	.0214	.0270
	Jsbach	.2334•	.2535	.2537	.2578	.2474○	.3135•	.3154○	.3317	.3408	.3723	.4047
	Nursery	.0064•	.0310	.0340	.0713	.0295○	.3141	.3097○	.3350	.3151	.3028•	.3149
	1st/2nd/3rd	9/1/0	2/6/2	0/0/4	0/0/3	0/2/1	6/3/1	1/4/2	0/1/3	1/1/2	2/1/3	-
	Avg. rank	1.1000	2.0000	3.6000	3.9000	4.2000	1.5000	2.9000	3.9000	3.5000	3.1000	-

Table 2: Optimization and generalization performance on hyper-parameter optimization for ensemble classifier. The entries are marked • and ○ mean the best and second error rate in each corresponding group. The datasets in bold are the target datasets. The number of 1st/2nd/3rd ranks and average rank (avg. rank) for each method are reported on experienced and target datasets respectively.

problem settings. The conclusion of this experiment is similar with that of the convergence experiment. ExpSRacos is much better than the compared methods when shifting space X^{shift} is small, but gets smaller advantage for larger shifts.

The results of synthetic function experiments indicate that the experienced optimization mechanism can have significantly improved performance when problems are similar.

4.2 On AutoML Tasks

In this section, we apply ExpSRacos to solve autoML tasks: the hyper-parameter optimization on heterogeneous weighted voting ensemble classifier and the architecture optimization on deep neural networks. These two tasks with the k-fold cross validation error as the criterion have been formulated in Section 2. For ExpSRacos, the datasets are split as source and target datasets. We organize the experience dataset on source datasets, and test the reusing performance of Φ on the target datasets. ExpSRacos is compared with SRacos, SMAC, TPE. In the experiment results, the optimization and generalization performance mean the validation error on training dataset when optimization and test error on testing dataset. We run each method for 3 times independently and the best solution is selected for testing generalization performance. For each optimal solution, we test the generalization performance for 5 times and report its average performance.

On Ensemble Classifier Hyper-Parameter Optimization

This task is optimizing hyper-parameters for heterogeneous weighted voting ensemble classifier $\mathbf{C_V}$. We choose ten base classifiers as follows: decision tree, multilayer perceptron, logistic regression, SVM, Gaussian processes, passive aggressive classifier, Gaussian naive Bayes, linear classifiers with SGD training, random forest, k-nearest neighbors classifier. All the 24 hyper-parameters (17 continuous, 4 integer and 3 categorical) need to be optimized including voting weights and hyper-parameters in the base classifiers. Our codes are based on scikit-learn [Pedregosa *et al.*, 2011]. We report the generalization performance of $\mathbf{C_V}$ with default hyper-parameters (noted as Default $\mathbf{C_V}$) on each dataset as base-

Dataset	optimization error				generalization error				HC-Net
	ExpSRacos	SRacos	SMAC	TPE	ExpSRacos	SRacos	SMAC	TPE	
Source dataset (MNIST)	**.0069**	.0079	.0103	.0093	**.0078**	.0081	.0095	.0091	.0083
Target dataset (**SVHN**)	**.0557**	.0617	.0782	.0772	**.0567**	.0664	.0759	.0796	.0634

Table 3: The optimization and generalization performance of neural network architecture optimization. The value in bold means the best error rate on its dataset. The dataset in bold is the target dataset.

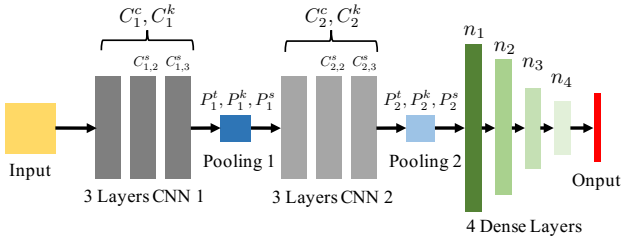


Figure 2: The neural network architecture for the datasets.

line. A tricky method Best C is choosing a base classifier with best 5-fold cross validation error. Best C is employed as the baseline on optimization performance.

We collected 34 UCI classification datasets. Among these, 24 datasets are the source datasets (top 24 data sets in Tabel 2). The rest of them are the target datasets (dataset name in bold in Tabel 2). We organize the experience dataset by running SRacos on training datasets. For organizing the experience dataset, SRacos has 200 evaluation budget and repeat for 10 times. The label imbalanced problem exists on this task too. We still use upsampling to balance positive and negative instances. ExpSRacos is then employed to optimize the hyper-parameters. We set 50 sample budget for all the compared methods.

The results are presented in Table 2. The generalization performance by optimization methods is better than the baseline (Default $\mathbf{C_V}$) in most cases. It indicates that hyper-parameter optimization can improve generalization performance effectively. ExpSRacos receives the best optimization and generalization error (the avg. rank is the lowest compared with other methods). ExpSRacos outperforms SRacos in most cases. Especially, ExpSRacos beats SRacos on 9 target datasets. It indicates the effectiveness of reusing directional model in ExpSRacos. On the generalization performance, similar results have been obtained. ExpSRacos outperforms other methods on both source and target datasets, and its performance on the target datasets is significantly better. All these results indicate that the experienced optimization can improve optimization efficiency significantly.

On Neural Network Architecture Optimization

We use two image datasets: MNIST [LeCun *et al.*, 1998] as the source dataset and SVHN [Netzer *et al.*, 2011] as the target dataset. The neural network architecture is illustrated in Figure 2. In each pooling layer, we set pooling type (P^t, max or average pooling), kernel size (P^k) and stride size (P^s) as hyper-parameters. In dense layers, the number of each hidden layer is set as hyper-parameters (n_1, n_2, n_3, n_4). In CNN 1 or 2, we use two hyper-parameters to dynamically choose the 2nd ($C^s_{,2}$) and 3rd ($C^s_{,3}$) layer, the other hyper-parameters (channel size C^c, kernel size C^k) are the same among layers. We use Adam as the optimizer, and its learning rate is also an important hyper-parameter. So, there are 19 hyper-parameters in this optimization problem. The generalization performance of a hand-crafted designing net (HC-Net) in [Springenberg *et al.*, 2015] is selected as the baseline with the learning rate 0.0002. We separately set optimization epoch size is 20 and 30 for all training in MNIST and SVHN.

We organize the experience dataset by running SRacos for optimizing on MNIST. The sample budget is set to 200, and optimization process is repeated for 5 times independently. Then, we test Φ on SVHN. The sample budget of all methods is 50. The results of this experiments is presented in Table 3. ExpSRacos receives the best performance in optimization. Comparing with SRacos on SVHN, the optimization and generalization error obtained by ExpSRacos are nearly 0.1% and 1% better than SRacos. On generalization performance, note that the baseline HC-Net is better than all the non-experienced optimization methods, but ExpSRacos achieves a significant improvement from the baseline.

5 Conclusions

AutoML tasks such as hyper-parameter optimization and neural network architecture optimization attract attentions incrementally. AutoML tasks are often solved by derivative-free optimization, which, however, commonly suffers from the high evaluation cost. This paper proposes an experienced optimization approach. It utilizes the experience of historical optimization processes to guide the optimization on new tasks. Specifically, the directional model is used to predict the direction of the next samples generated in every algorithm iteration. A pre-sample step is added in the optimization to generate candidate samples and filter them with the directional model. In this way, this framework can eliminate many unnecessary explorations and thus improve the sample effectiveness. Incorporating with a state-of-the-art derivative-free optimization method SRacos, we implement this framework as ExpSRacos. We empirically compare ExpSRacos with other state-of-the-art methods on Ackley function and two autoML tasks. The experiment results show that the experienced optimization with directional model can effectively reduce the sample budget and improve the optimization. Recently Zhou [Zhou, 2016] proposed the new concept of learnware, with properties of reusability, evolvability and comprehensibility. The effort of this paper can be viewed as an effort towards reusability from a new perspective.

References

[Adankon and Cheriet, 2009] Mathias M. Adankon and Mohamed Cheriet. Model selection for the LS-SVM. Application to handwriting recognition. *Pattern Recognition*, 42(12):3264–3270, 2009.

[Baker *et al.*, 2017] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *Proceedings of 5th International Conference on Learning Representations (ICLR'17)*, 2017.

[Bengio, 2000] Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000.

[Bergstra and Bengio, 2012] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[Bergstra *et al.*, 2011] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems (NIPS'11)*, pages 2546–2554, 2011.

[Biem, 2003] 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*, pages 104–108, 2003.

[Brazdil *et al.*, 2003] 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.

[Feurer *et al.*, 2015] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems (NIPS'15)*, pages 2962–2970, 2015.

[Hall *et al.*, 2009] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[Hansen *et al.*, 2003] 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.

[Hu *et al.*, 2017] 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 (AAAI'17)*, pages 2029–2035, 2017.

[Hutter *et al.*, 2011] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. *LION*, 5:507–523, 2011.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Lindauer and Hutter, 2018] Marius Lindauer and Frank Hutter. Warmstarting of model-based algorithm configuration. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*, pages 1355–1362, 2018.

[Munos, 2011] Rémi Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *Advances in Neural Information Processing Systems (NIPS'11)*, pages 783–791, 2011.

[Netzer *et al.*, 2011] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on Deep Learning and Unsupervised Feature Learning*, volume 2011, page 5, 2011.

[Orr and Müller, 2012] Genevieve B Orr and Klaus-Robert Müller. *Neural Networks: Tricks of the Trade - Second Edition*. Springer, 2012.

[Pedregosa *et al.*, 2011] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Shahriari *et al.*, 2015] 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.

[Springenberg *et al.*, 2015] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *Proceedings of 3rd International Conference on Learning Representations (ICLR'15)*, 2015.

[Swersky *et al.*, 2013] Kevin Swersky, Jasper Snoek, and Ryan P. Adams. Multi-task bayesian optimization. In *Advances in Neural Information Processing Systems (NIPS'13)*, pages 2004–2012, 2013.

[Thornton *et al.*, 2013] 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 (SIGKDD'13)*, pages 847–855, 2013.

[Yu *et al.*, 2016] Yang Yu, Hong Qian, and Yi-Qi Hu. Derivative-free optimization via classification. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*, pages 2286–2292, 2016.

[Zhou, 2016] Zhi-Hua Zhou. Learnware: On the future of machine learning. *Frontiers of Computer Science*, 10(4):589–590, 2016.