计算机科学
COMPUTER SCIENCE

传统机器学习模型的超参数优化技术评估

李海霞, 宋丹蕾, 孔佳宁, 宋亚飞, 常海艳

引用本文

李海霞, 宋丹蕾, 孔佳宁, 宋亚飞, 常海艳. 传统机器学习模型的超参数优化技术评估[J]. 计算机科学, 2024, 51(8): 242-255.

LI Haixia, SONG Danlei, KONG Jianing, SONG Yafei, CHANG Haiyan. Evaluation of Hyperparameter Optimization Techniques for Traditional Machine Learning Models [J]. Computer Science, 2024, 51(8): 242-255.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230600164

传统机器学习模型的超参数优化技术评估

李海霞[1]　宋丹蕾[2]　孔佳宁[2]　宋亚飞[3]　常海艳[1]
1 北方自动控制技术研究所　太原 030006
2 西安交通大学数学与统计学院　西安 710049
3 空军工程大学防空反导学院　西安 710051
(keweitta@163.com)

摘　要　合理的超参数能够保证机器学习模型适应不同背景和不同任务。为了避免在模型超参数数量过多、搜索空间过大的情况下出现手动调节导致的效率低下问题,多种超参数优化技术已经被研发并运用到机器学习模型训练中。文中首先回顾了8种常见的超参数优化技术,即网格搜索、随机搜索、贝叶斯优化、Hyperband、BOHB、遗传算法、粒子群优化算法和协方差矩阵自适应进化策略,并从时间性能、最终结果、并行能力、可拓展性、稳健性和灵活性5个方面分析各类方法的优缺点。其次,将8种方法应用到LightGBM、XGBoost、随机森林和KNN这4种传统机器学习模型上,在4个基准数据集上完成了回归、二分类和多分类的实验,对各类方法进行了比较。最后总结了各类方法的优缺点,给出了不同方法的适用情景。

关键词:传统机器学习;超参数优化;贝叶斯优化;多保真技术;元启发式算法
中图分类号　TP181

Evaluation of Hyperparameter Optimization Techniques for Traditional Machine Learning Models

LI Haixia[1],SONG Danlei[2],KONG Jianing[2],SONG Yafei[3] and CHANG Haiyan[1]
1 North Automatic Control Technology Institute,Taiyuan 030006,China
2 School of Mathematics and Statistics,Xi'an Jiaotong University,Xi'an 710049,China
3 Air and Missile Defense College,Air Force Engineering University,Xi'an 710051,China

Abstract　Reasonable hyperparameters ensure that machine learning models can adapt to different backgrounds and tasks. In order to avoid the inefficiency caused by manual adjustment of a large number of model hyperparameters and a vast search space, various hyperparameter optimization techniques have been developed and applied in machine learning model training. At first, Paper reviews eight common hyperparameter optimization techniques: grid search, random search, Bayesian optimization, Hyperband, Bayesian optimization and Hyperband(BOHB), genetic algorithms, particle swarm optimization algorithm, and covariance matrix adaptation evolutionary strategy(CMA-ES). The advantages and disadvantages of these methods are analyzed from five aspects: time performance, final results, parallel capability, scalability, robustness and flexibility. Subsequently, these eight methods are applied to four traditional machine learning models: LightGBM, XGBoost, Random Forest, and *K*-Nearest Neighbors(KNN). Regression, binary classification and multi-classification experiments are performed on four standard datasets: Boston house price dataset, kin8nm power arm dataset, credit card default customer dataset and handwritten digit dataset. Different methods are compared by evaluating their performance using output evaluation metrics. Finally, pros and cons of each method and are summarized, and the application scenarios of different methods are given. The results highlight the importance of selecting appropriate hyperparameter optimization methods to enhance the efficiency and effectiveness of machine learning model training.

Keywords　Traditional machine learning, Hyperparameter optimization, Bayesian optimization, Multi-fidelity technology, Metaheuristic algorithms

---

到稿日期:2023-06-20　返修日期:2023-11-23
基金项目:国家自然科学基金(61806219,61876189);陕西省高校科协青年人才托举计划(20220106);陕西省创新能力支撑计划(2020KJXX-065)
This work was supported by the National Natural Science Foundation of China(61806219,61876189), Young Talent Fostering Program of the Science and Technology Associations of Universities in Shaanxi Province, China(20220106) and Innovation Capacity Support Program of Shaanxi Province, China(2020KJXX-065).
通信作者:宋亚飞(yafei_song@163.com)

1 引言

机器学习模型因其极强的实用性和优越的性能现已被广泛应用于各个领域，包括但不限于经济学、材料科学、物理学、农业、生物学等[1-5]。传统的机器学习模型发展至今已较为完备，虽然无法像深度神经网络一样有效地满足人们在图像、音视频和自然语言数据方面的任务需求，但对于基本的分类和回归任务仍然具有极大的应用价值。在使用传统机器学习模型完成分类和回归任务时，并非简单地将特征数据和标签数据输入模型中直接进行训练，还需要对模型自带的超参数进行调节，以改变模型的结构，从而使得模型在不同任务下实现最佳性能。

超参数是在训练模型之前手动设置的参数，不能直接从数据中学习得到，定义了模型的体系结构，用于控制模型的学习过程。虽然可以通过人工手动调节超参数，但这种方式仅适用一些超参数数量较少且超参数之间不存在相关关系的机器学习模型，例如，对于支持向量机模型，影响其泛化性能的超参数主要包括松弛因子和核类型[6-7]；对于 K 近邻回归和分类模型，其性能主要受到近邻数的影响[8]；逻辑回归的性能主要由惩罚项类型和惩罚因子决定[9]。然而，这类机器学习模型相对较少，并且手动调整超参数并不一定能够找到全局最优的超参数设置。当遇到超参数很多且不同超参数之间具有相关关系的模型时，例如梯度提升树[10]、AdaBoost[11]、XGBoost[12]、LightGBM[13]、Catboost[14]等集成模型，超参数的调节工作将变得格外繁琐，这些模型涉及许多能够显著影响模型性能的超参数，并且其中一些超参数之间存在依赖关系，在这种情况下，继续使用手动调节的工作量极大。因此，为了减少研究人员对大型数据集或具有大量超参数的复杂机器学习模型调参的任务量，使模型和研究更具可重复性[15]，研发适合机器学习模型超参数自动调优方法是必要的。

由于机器学习模型本身的超参数不仅有数值型还有分类型，而且一些超参数在模型训练过程中并不直接参与到损失函数的运算中，因此很难计算这类超参数的梯度信息，故使用类似梯度下降法等传统优化算法来优化机器学习模型的超参数并不合适。很多学者也提出了更适合优化机器学习模型超参数的算法，具体包括暴力搜索方法、贝叶斯优化算法、多保真算法、元启发式算法以及基于强化学习和元学习的超参数优化方法等[16-28]。暴力搜索方法中最常用的就是网格搜索[16]和随机搜索[17]；贝叶斯优化（Bayesian Optimization，BO）根据选取的概率代理模型不同分为基于高斯过程（Gaussian Process，GP）的贝叶斯优化[18]、基于 TPE（Tree-structured Parzen Estimators）的贝叶斯优化[19]和基于序列模型的优化方法（Sequential Model Algorithm Configuration，SMAC）[20]；常见的多保真优化算法包括连续减半算法（Successive Halving）[21]、Hyperband[22]和 BOHB[23]；元启发式算法包括遗传算法（Genetic Algorithm，GA）[24]、粒子群算法（Particle Swarm Optimization，PSO）[25]和协方差矩阵自适应进化策略（Covariance Matrix Adaptation Evolution Strategy，CMA-ES）[26]等。在解决超参数优化问题上，强化学习和元学习方法展现出了巨大的潜力，Wu 等以强化学习和元学习技术为基础，提出了一种能够高效解决超参数优化问题的方法[27]，然后又在此基础上进行了改进，依次提出了基于上下文的元强化学习[28]和基于任务感知的上下文元强化学习的超参数优化方法[29]。

本文回顾了当前用于优化机器学习模型的超参数优化技术，详细分析了除强化学习的各类方法的理论和基本算法流程，并比较了各类算法的优缺点；同时还将这些优化算法应用到 4 种传统机器学习模型的分类和回归任务中，通过比较使用不同超参数优化技术得到的最优超参数组合的机器学习模型在测试集上的评价指标和优化所用的时长，分析各类超参数优化技术的特点。本文第 2 章将超参数优化任务进行数学表达并给出评价超参数优化技术的指标；第 3 章详细介绍了常用超参数优化技术的理论和具体算法流程；第 4 章对前文介绍的超参数优化技术进行实验比较；第 5 章给出了本文研究的结论；最后总结全文。

2 机器学习模型超参数优化的数学形式表达

当前机器学习领域已经涌现出各种各样的超参数优化方法，但不同方法究其本质都可以抽象成为一个数学问题：设机器学习模型为 M，该模型既可以是分类模型也可以是回归模型；机器学习模型内部需要人为设置的超参数组为 α，超参数组中可以包含不止一种超参数，每种超参数的类别可以为连续型、离散型、二元型和分类型[30]；超参数组的搜索空间为 A；用于训练和验证机器学习模型的训练集和验证集分别为 X_{tra} 和 X_{val}；用于计算机器学习模型损失值的损失函数为 f；对于回归问题可以使用均方误差（MSE）或者均方根误差（RMSE），对于分类问题则可以使用对数损失等。这从侧面说明在机器学习的超参数优化过程中目标函数 f 并非黑箱函数，因此机器学习模型超参数优化过程可以表示为一个最优化问题。其目标是在超参数搜索空间中找到一组使得经过训练集训练的机器学习模型在验证集上达到最小损失值的最优超参数组合。

$$\alpha^{*}=\arg\min_{\alpha\in A} f\{M(\alpha|X_{\text{tra}})|X_{\text{val}}\} \tag{1}$$

机器学习超参数优化流程一般为[31]：首先选取目标函数和性能指标；其次选择需要调优的超参数，汇总它们的类型，并确定合适的超参数优化算法；接着使用机器学习模型默认超参数设置在训练集上进行训练，并将其作为基准模型；然后在一个人为规定或者根据先验知识确定的尽可能大的超参数搜索范围中运行超参数优化算法，根据超参数优化得到性能良好的超参数值，逐渐缩小搜索空间，或者在必要时探索新的搜索空间；最终选择性能最佳的超参数设置作为优化结果。然而，一般化流程中并没有考虑到优化所需时间的问题，当不限制计算时间和计算资源时，在足够大的超参数搜索空间进行多次寻找总能找到使得损失函数达到最小的全局最优或近似全局最优解；当计算资源有限时，需要高效快捷的超参数优化方法。因此，Falkner 等[23]认为一种优秀的超参数优化方法应该满足以下 5 个要求：

1）强大的时间性能。在处理大规模数据集或者具有大量特征的问题时，机器学习模型所需要的时间会随着数据集规模的扩大或特征数的增多而增加。在这种情况下，研究人员

通常会面临时间预算的限制，因为他们需要在研究和实验中保持高效，无法轻易访问大量的计算资源，所以超参数优化的预算通常有限，故一个优秀的超参数优化算法要求在尽可能短的时间内寻找到最优的超参数设置。

2)优越的最终结果。优秀的超参数优化方法需要在利用尽可能多的计算资源和预算时，能够寻找到使机器学习模型实现最佳或接近最佳性能的超参数配置。

3)并行资源的有效利用。随着并行计算的兴起，可用的并行资源大量涌现(例如计算集群或云计算)，优秀的超参数优化方法需要能够有效地利用这些资源。

4)可拓展性。优秀的超参数优化方法既能做到优化少量超参数，同时也应该能完成优化大量超参数任务。

5)稳健性和灵活性。优秀的超参数优化方法应当具备适应不同机器学习模型的超参数优化需求的能力，同时应能够有效地处理机器学习模型中可能存在的连续型和离散型超参数。

下文也会将以上 5 种要求作为衡量标准，对当前已有并广泛使用的超参数优化方法进行比较。

3 超参数优化技术

本章主要介绍常用的超参数优化技术原理以及算法流程。

3.1 网格搜索

网格搜索是传统机器学习超参数优化的一种方法[16]，其基本流程是在预先给定的超参数搜索空间中，通过遍历所有可能的超参数组合，根据评价指标来寻找最优的超参数设置。然而，网格搜索存在明显的缺点：首先，在给定超参数搜索范围时，需要人为枚举每个超参数的所有可能取值，对于连续型超参数，这可能会导致难以确定枚举规律；其次，当搜索范围过大时，由于无法进行并行计算，网格搜索所需时间会呈指数级增长，影响优化效率。假设有 n 个待优化超参数，每个超参数有 k 个可能取值，则网格搜索的时间复杂度为 $O(n^k)$[32]。此外，由于超参数搜索范围是人为设定的，即使提供了许多可能取值，也可能遗漏全局最优超参数组合，导致最终结果并不是最优的超参数设置。

3.2 随机搜索

为了弥补网格搜索的不足，随机搜索采用在给定超参数搜索范围内随机抽样的方式生成待选超参数，然后从中选择最优超参数组合。相比网格搜索，随机搜索中的计算次数是一个固定值 n，因此计算复杂度为 $O(n)$[33]，能够缓解网格搜索中超参数搜索范围扩大导致搜索时间呈指数级增长从而效率降低的问题。Bergstra 等[17]也证明，随机搜索在超参数优化方面比网格搜索更为有效。然而，随机搜索与网格搜索一样，仍存在无法并行运行和受人为设置影响而得不到最优超参数设置的缺陷。

3.3 贝叶斯优化

尽管学者们已经对提高网格搜索效率方面做出了各种优化[34-35]，但仍然无法在效率和精度上做到双赢。若希望更快速地进行搜索，并找到一组泛化能力尽可能强的超参数，目前的常见做法是使用带有先验过程的调参工具，即贝叶斯优化。贝叶斯优化的本质思想就是根据先验信息，通过采集函数采样来建模机器学习模型性能的评价指标分布[18](称为概率代理模型)，该模型描述了超参数组合对模型性能的影响。根据概率代理模型的不同，可以分为 3 种不同的优化方法，分别是基于高斯过程的贝叶斯优化[18]、基于 TPE 的贝叶斯优化[19]以及基于序列模型的优化方法[20]。尽管贝叶斯优化能够充分利用之前的超参数经验，更快、更高效地选择下一次的超参数组合，但传统的贝叶斯优化也存在一些缺点，包括高维空间下效率的降低、对先验或假设条件的依赖、并行性限制和噪声敏感性等。因此，近年来，不断有学者针对贝叶斯优化存在的问题对其进行改进，使其更加适应高维度超参数估计[36-39]、进行并行化扩展[40-43]、多任务扩展[44-46]、能够解决整数约束优化问题[47-48]以及与其他方法结合提高性能[49-51]等。

使用贝叶斯进行超参数优化时，在一定迭代次数内，采集函数以概率代理模型为先验信息，采集新的超参数组合用于更新模型。最后，根据更新后的概率代理模型找到使得机器学习模型性能最佳的超参数组合。因此，在贝叶斯优化中，采集函数和概率代理模型是至关重要的两个方面。

3.3.1 采集函数

采集函数利用先前优化迭代的先验信息，在超参数组合的候选集中选择新的超参数组合，用于构建最终机器学习模型评价指标的分布。贝叶斯优化的效果与采集函数密切相关，不合适的采集函数可能导致陷入局部最优解。采集函数的主要功能是找到影响概率代理模型的超参数组合，并选择影响最优的超参数组进行下一步观测。常用的采集函数是 EI(Expected Improvement)[18]。

首先定义 $I(x)=\max\{0, f_{t+1}(x)-f(x^+)\}$，其中 $f(x^+)$ 代表截至上一轮优化迭代所得到的概率代理模型在能够使机器学习模型性能达到最优的超参数组合 x^+ 的函数值，$f_{t+1}(x)$则是将超参数组候选集中的每个观测值作为新入选的超参数组得到的新概率代理模型的最优值。由于 $I(x)$是一个随机变量并且服从正态分布，可以计算出 $I(x)$的概率密度函数。

$$PDF=\frac{1}{\sqrt{2\pi}\sigma(x)}\exp\left\{-\frac{(\mu(x)-f(x^+)-I)^2}{2\sigma^2(x)}\right\} \tag{2}$$

其中，$\mu(x)$和 $\sigma^2(x)$为利用不同概率代理模型得到的均值和方差函数。根据概率密度函数可以计算出 $I(x)$的期望，记为 $EI(x)$：

$$EI(x)=\int_0^{\infty} I\,\frac{1}{\sqrt{2\pi}\sigma(x)}\exp\left\{-\frac{(\mu(x)-f(x^+)-I)^2}{2\sigma^2(x)}\right\}\mathrm{d}I \tag{3}$$

一般可以将 $EI(x)$写为：

$$EI_{y^*}(x)=\int_{-\infty}^{y^*}\max(y^*-y,0)P(y|x)\mathrm{d}y \tag{4}$$

3.3.2 概率代理模型

在贝叶斯优化过程中，由于每次取样的超参数组合数量有限，且需要利用有限的超参数组和来模拟整个模型评价指标的分布变化，因此需要概率代理模型根据有限的观测值对模型评价指标的分布进行估计。这些概率代理模型自带一些假设，并能够利用少数观测点估计出目标函数的分布，包括

每个点的取值和对应的置信度。这里介绍了包括高斯过程[18]和 TPE 方法[19]两种概率代理模型。

高斯过程的联合分布是连续域(例如时间或空间)上函数的分布,它可以由均值函数 $m(\cdot)$ 和协方差函数 $k(\cdot)$ 来表示。

$$f(x)=GP(m(x),k(x,x'))\tag{5}$$

常见的协方差函数为平方指数函数:

$$k(x_i,x_j)=\exp\left(-\frac{1}{2}\|x_i-x_j\|^2\right)\tag{6}$$

根据已有的前 t 次优化得到的入选的超参数组数据 $\{X_{1:t}:x_1,\cdots,x_t\}$ 可以得到一个协方差矩阵,也就是核矩阵:

$$K=\begin{bmatrix}k(x_1,x_1) & \cdots & k(x_1,x_t)\\ \vdots & \ddots & \vdots\\ k(x_t,x_1) & \cdots & k(x_t,x_t)\end{bmatrix}\tag{7}$$

那么就可以推导出前 t 次和第 $t+1$ 次优化得到的概率代理模型的分布,其本质也是一个多元高斯分布:

$$\begin{bmatrix}f_{1:t}\\ f_{t+1}\end{bmatrix}\sim N\left(0,\begin{bmatrix}K & k_{t+1}\\ k_{t+1}^T & k(x_{t+1},x_{t+1})\end{bmatrix}\right)\tag{8}$$

$$k_{t+1}=[k(x_{t+1},x_1),\cdots,k(x_{t+1},x_t)]\tag{9}$$

给定前 t 个观察样本,我们可以利用高斯过程计算出观察值的可能分布,即:

$$P(f_{t+1}|X_{1:t},x_{t+1})=N(\mu_t(x_{t+1}),\sigma_t^2(x_{t+1}))\tag{10}$$

其中,$\begin{cases}\mu_t(x_{t+1})=k_{t+1}^TK^{-1}f_{1:t}\\ \sigma_t^2(x_{t+1})=k(x_{t+1},x_t)-k_{t+1}^TK^{-1}k_{t+1}\end{cases}$。

高斯过程是根据 $P(x|y)$ 进行建模,而 TPE 是根据 $P(x|y)$ 和 $P(y)$ 进行建模,TPE 定义为如下概率密度:

$$P(x|y)=\begin{cases}l(x), & y<y^*\\ g(x), & y\geqslant y^*\end{cases}\tag{11}$$

其中,$l(x)$ 是满足 $f(x)<y^*$ 的概率密度函数,反之 $g(x)$ 亦然。设 $P(y<y^*)=\gamma$,则可对 $EI_{y^*}(x)$ 进行改写:

$$EI_{y^*}(x)=\int_{-\infty}^{y^*}(y^*-y)\frac{P(x|y)P(y)}{P(x)}\mathrm{d}y\tag{12}$$

因为 $P(y<y^*)=\gamma$,可得:

$$P(x)=\int P(x|y)P(y)\mathrm{d}y=\gamma l(x)+(1-\gamma)g(x)\tag{13}$$

进一步可以推导出 $EI_{y^*}(x)$ 变形。

$$EI_{y^*}(x)=\frac{\gamma y^*l(x)-l(x)\int_{-\infty}^{y^*}P(y)\mathrm{d}y}{\gamma l(x)+(1-\gamma)g(x)}\propto\left(\gamma+\frac{g(x)}{l(x)}(1-\gamma)\right)^{-1}\tag{14}$$

通过它正比于后面这一项也可以看出,要想得到比较大的增益,倾向于选择 $l(x)$ 较大、$g(x)$ 较小的 x。

3.3.3 贝叶斯优化算法的流程

算法 1 贝叶斯优化算法

输入:数据集 D,目标函数 f,超参数搜索空间 A,采集函数 S,概率代理模型 L,循环优化次数 T

输出:超参数优化结果 α^*

1. 初始化 D 并将 f,A,S 和 L 作为输入;
2. for t←1 to T do
3. 根据 D 和 L 计算出本次迭代目标函数值的可能的分布 $P(f_{t+1}|X_{1:t},x_{t+1})$;
4. 基于分布 $P(f_{t+1}|X_{1:t},x_{t+1})$ 利用采集函数求解出本次迭代的最优超参数组 x_{t+1};
5. 利用 x_{t+1} 计算目标函数值 f_{t+1},并将该组数据 (x_{t+1},f_{t+1}) 加入 D 中;
6. end for。

贝叶斯优化的具体流程就是在规定的循环次数内,反复利用上一次循环得到的超参数组集合求解概率代理模型;然后将概率代理模型作为先验信息,使用采集函数求解出最新采样的超参数组,并计算在该超参数组下机器学习模型在验证集上的性能表现;此后将该超参数组和相应的模型性能并入上一轮得到的超参数组集合中。循环结束后,可以得到一个机器学习模型在不同超参数组条件下在验证集上的性能曲线。由于该曲线近似于模型在验证集上的真实性能,因此可以取该曲线上的最小值作为贝叶斯优化超参数结果。对于一个大小为 n 的数据集,基于高斯过程的贝叶斯优化(BO-GP)时间复杂度为 $O(n^3)$[52],基于 TPE 的贝叶斯优化(BO-TPE)的时间复杂度为 $O(n\log n)$[53]。

3.4 多保真优化

多保真超参数优化技术结合了高保真优化和低保真优化两种方法。高保真优化使用更多数据进行模型训练和优化,得到的模型泛化性能强,但需要更多计算资源和时间。低保真优化在预算有限的情况下,使用较少样本和时间进行模型训练,得到的模型精度较低,但仍有一定合理性[54];多保真优化将这两种方法相结合,通过不断舍弃表现较差的超参数组合,最终得到表现优异的超参数组合作为优化结果。在超参数优化领域,被广泛使用的多保真优化算法包括:连续减半算法[21]、Hyperband[22]和 BOHB[23]。同时,学者们不断对多保真优化进行改进,使其能够适应更多情况,如 Li 等提出了异步连续减半算法[55],与连续减半算法相比更适用于时间有限的大规模场景,随后 Schmucker 等将异步连续减半算法扩展到多目标[56];Awad 等将 Hyperband 与差分进化算法结合使其更适应离散高维问题[57],并将其扩展到多目标问题[58]。这里只介绍最常用的 3 种算法。

3.4.1 连续减半算法

连续减半算法是一种统一为每个超参数组分配资源,并通过不断剔除性能较差的组合来进行优化的方法。因此,该算法就是将更多的资源分配到更有前景的配置上。然而,由于资源有限,增加待选超参数组的数量会降低每组超参数组的资源预算,影响后续评估。因此,连续减半算法面临着在有限资源预算 B 和待选超参数组数量 n 之间的取舍:是牺牲平均分配到每一组超参数组中的资源 B/n 以评估更多超参数组,还是减少待选超参数组数量 n 来为每个超参数组提供更多的资源 B/n?

3.4.2 Hyperband

Hyperband 算法是在连续减半算法基础上加入了对于计算资源的控制,旨在解决有限资源预算 B 和待选超参数组数量 n 之间的平衡问题。其核心思想是通过网格搜索确定在给定预算 B 下最优的待选超参数组数量 n,因此连续减半算法可以视为 Hyperband 算法的一个子程序。Hyperband 算法的流程如算法 2 所示。

算法 2 Hyperband 算法

输入：单个超参数组合所能分配的最大预算 R；连续减半算法中每次迭代后的淘汰参数比例 η

输出：超参数优化结果 α^*

1. 初始化算法优化超参数的最大循环次数 $s_{max}=\lfloor\log_\eta(R)\rfloor$ 和总体计算资源 $B=R(s_{max}+1)$；
2. for $s\leftarrow s_{max}$ to 0 do
3. 计算超参数组数量 $n=\lceil B\eta^s/R(s+1)\rceil$ 和单个超参数组合实际分配的预算资源 $r=R\eta^{-s}$；
4. 利用随机均匀采样得到 n 个待选超参数组；
5. for $i\leftarrow 1$ to s do
6. 计算当前循环下每个待选超参数组的数量 $n_i=\lfloor n\eta^{-i}\rfloor$；
7. 计算每个超参数组合实际所能分配的预算 $r_i=r\eta^i$；
8. 对这 n 个待选超参数组进行连续减半算法，直至找到当前条件下唯一最优超参数值组；
9. end for
10. end for
11. $s_{max}+1$ 次循环之后得到 $s_{max}+1$ 个不同条件下的超参数组，比较评价指标找到最优超参数组合。

从算法流程中可以看出，整个 Hyperband 算法包括内部循环和外部循环两个循环，其中内部循环就是连续减半算法的标准流程。Hyperband 算法的计算复杂度为 $O(n\log n)$[22]。

3.4.3 BOHB

BOHB 的名字来自于 Bayes Optimization 和 Hyperband 的拼接，算法改进了 Hyperband 的采样过程，将 Hyperband 的随机采样过程用贝叶斯优化代替，从而能够利用之前使用过的超参数组作为先验信息，加快收敛速度[23]。BOHB 的采样算法的流程如算法 3 所示。

算法 3 BOHB 的采样算法

输入：数据集 D，随机抽样概率 ρ，数据集样本数 N_s，构建模型需要的最小样本个数 N_{min}，带宽参数 b_s

输出：超参数采样结果

1. 如果随机数小于 ρ，则继续均匀随机采样；
2. $b=\text{argmin}\{D_b:|D_b|>N_{min}+2\}$；
3. 如果 b 不为空，则继续均匀随机抽样；
4. 计算 $l(x)=P(y<\alpha|x,D)$ 和 $g(x)=P(y\geqslant\alpha|x,D)$；
5. 计算 $N_{b,l}=\max(N_{min},qN_b)$ 和 $N_{b,g}=\max(N_{min},N_b-N_{b,l})$；
6. 根据步骤 4、步骤 5 的计算结果拟合多维的核密度估计；
7. 返回 $l(x)/g(x)$ 最高的结果作为超参数组的采样结果。

算法 3 第 5 步中的 N_b 代表在预算为 b 的情况下，待评估超参数组的个数，第 5 步可以保证两个密度函数都有足够的超参数组来建模。从第 4 步中可以看出，BOHB 算法中利用先验信息构建概率密度函数的过程与 TPE 类似，不同点在于 TPE 使用一维核估计的层次结构，而 BOHB 使用多维的核密度估计，以更好地处理输入空间中的交互作用。后续在采样过程中，BOHB 选取的样本是能够使得 $l(x)/g(x)$ 尽可能大的超参数组，超参数组的选择与前面介绍的贝叶斯优化中使用 TPE 作为概率代理模型、$EI_{y^*}(x)$ 作为采集函数的方法一致，因此 BOHB 算法的复杂度也为 $O(n\log n)$[23]。

3.5 元启发式算法

元启发式算法是一类基于自然界或社会行为的优化算法，主要通过模拟生物群体、群体智能或其他群体的行为来进行问题求解，在解决复杂、高维和非线性等问题上表现较为优秀。使用较为广泛的元启发式算法包括遗传算法[24]、粒子群优化算法[25]、协方差矩阵自适应进化策略[26]和差分进化算法[59]等。近年来，随着研究的深入，学者们不断地提出新的元启发式算法[60-61]，或将已有的元启发式算法进行改进[62]，或与其他方法相结合[57,63]，以提升算法性能。下面介绍常用的 3 种元启发式算法。

3.5.1 遗传算法

遗传算法是一种基于自然进化理论的进化算法[24]，根据达尔文的“优胜劣汰，适者生存”的思想来找到最优解。遗传算法的研究对象是种群，即很多个体的集合，对应于求解的问题，个体代表一个解，种群代表这些解的集合，将这些解进行编码、选择、交叉、变异之后，根据适应度函数逐代进化，从子代中可以找到求解问题的全局最优解。其中，末代种群中的最优个体经过解码后可以作为问题的近似最优解。

将遗传算法应用于超参数优化问题，种群对应于超参数的搜索空间；适应度函数对应每组参数的评价指标；每个超参数对应一条染色体，即一个个体。每条染色体上有多个基因，基因是十进制的超参数值经过编码得到的二进制数，交叉和变异操作是对基因开展的。其算法流程如算法 4 所示。

算法 4 遗传算法

输入：种群大小 N，适应度函数 $F(x)$，交叉概率 P_c，变异概率 P_m，最大迭代次数 T

输出：适应度最高的个体（最优超参数组）

1. 编码：根据实际问题选择相应的编码方式；
2. 初始化种群：根据种群规模的需要，随机产生确定长度的 n 个染色体，并将其作为初始种群；
3. for $t\leftarrow 1$ to T do
4. for $i\leftarrow 1$ to n do
5. 计算适应度值：计算每个个体的适应度值 $F(x_i)$；
6. 进化计算：根据 P_c 和 P_m，对染色体进行选择、交叉和变异操作，生成下一代个体；
7. end for
8. end for
9. 达到最大迭代次数，停止并将适应度最高的个体进行解码后输出。

上述步骤中，在给定搜索空间中进行超参数组合的随机初始化，随机初始化的参数值通常不包括最优参数值，因此必须对染色体进行选择、交叉和变异操作，以找到最优超参数配置[64]。选择操作一般是基于个体的适应度，对个体进行评估和排序，然后根据个体的适应度值选取优良的个体用于繁殖。交叉是将两个父代染色体中的一部分基因片段进行交换，生成新的个体。变异指在个体基因型中随机改变一个或多个基因的值，生成新的后代个体。交叉和变异操作可以增加种群的多样性，避免算法过早陷入局部最优解。

3.5.2 粒子群优化算法

粒子群优化算法源于对鸟群捕食行为的研究[25]。粒子群优化算法的基本思想是群体中的每一个粒子都会受益于所有成员在这个过程中所发现和累积的经验，通过群体中粒子之间的协作和信息共享来寻找最优解。以鸟群觅食为例，与粒子群优化算法进行对比，结果如表 1 所列。粒子群优化算法中，随机产生一定数量的粒子作为问题搜索空间的有效解，

通过该问题对应的适应度函数确定粒子的适应值,然后进行迭代搜索,不断更新粒子的速度和位置,最终得到优化结果。粒子群优化算法的流程如算法 5 所示。

表 1 鸟群觅食和粒子群优化算法的基本定义

Table 1 Basic definitions of bird foraging and particle swarm optimization algorithms

鸟群觅食	粒子群优化算法
鸟群	搜索空间的一组有效解(表现为种群规模 N)
觅食空间	问题的搜索空间(表现为维数 D)
飞行速度	解的速度向量 $\boldsymbol{v}_i=[v_i^1,\cdots,v_i^D]$
所在位置	解的位置向量 $\boldsymbol{x}_i=[x_i^1,\cdots,x_i^D]$
个体认知与群体协作	每个粒子 i 根据自身历史最优位置 $pBest$ 和群体的全局最优位置 $gBest$ 更新速度和位置
找到食物	算法结束,输出最优解

算法 5 粒子群优化算法

输入:适应度函数 F(x),粒子数量 N,最大迭代次数 T

输出:最优粒子

1. 初始化粒子:初始化所有粒子速度和位置,并将个体的历史最优 pBest 作为当前位置,将群体中最优个体的位置作为当前的全局最优 gBest;
2. for t←1 to T do
3. for i←1 to N do
4. 计算适应度值:计算各个粒子的适应度函数值 $F(x_i)$;
5. 更新 pBest 和 gBest:if $F(x_i)>$pBest,do pBest$=F(x_i)$,if $F(x_i)>$gBest,do gBest$=F(x_i)$;
6. for d←1 to D do
7. 更新粒子速度 $\mathbf{v}_i^d$ 和位置 $\mathbf{x}_i^d$:

$$\mathbf{v}_i^d=\omega\mathbf{v}_i^d+c_1\times rand_1^d\times(\mathbf{pBest}_i^d-\mathbf{x}_i^d)+c_2\times rand_2^d\times(\mathbf{gBest}_i^d-x_i^d) \tag{15}$$

$$\mathbf{x}_i^d=\mathbf{x}_i^d+\mathbf{v}_i^d \tag{16}$$

其中,ω 是惯性权值,表示粒子保持的运动惯性,一般初始化为 0.9,然后随着进化过程线性递减到 0.4;c_1 和 c_2 是加速系数,也称学习因子,一般取固定值 2;$rand_1^d$ 和 $rand_2^d$ 是两个[0,1]区间上的随机数;

8. end for
9. end for
10. end for
11. 输出最优粒子。

相比遗传算法,粒子群优化算法的原理更简单,涉及的参数更少,实现更容易。此外,遗传算法的计算复杂度为 $O(n^2)$[65],粒子群优化算法的计算复杂度为 $O(n\log n)$[66],因此粒子群优化算法在多数情形下具有更快的收敛速度。但粒子群优化算法的主要限制是它需要适当的种群初始化,否则容易陷入局部最优。许多种群初始化技术已经被用来提升算法的性能,如基于反向学习的种群初始化算法[67],但进行额外的种群初始化会需要更多的计算资源和计算时间。

3.5.3 协方差矩阵自适应进化策略

协方差矩阵自适应进化策略是 Hansen 等提出的一种算法[26],通过模拟自然界生物进化过程来达到寻优目的。CMA-ES 算法是在进化策略(Evolution Strategy,ES)算法的基础上进行改进得到的。ES 算法是通过反复迭代调整一个正态分布 $N(\mu,\sigma^2)$ 来进行搜索,以实现个体突变。CMA-ES 算法引入了协方差矩阵适应性策略,通过更加智能化的变异操作和适应度函数处理方式,更加有效地进行优化,主要思路是通过反复迭代调整一个正态分布进行搜索。

CMA-ES 算法通过对正态分布 $N(\boldsymbol{\mu},\sigma^2\boldsymbol{C})$ 采样产生种群分布,其中 $\boldsymbol{\mu}$ 是种群分布的均值,σ 是标准差,算法中也称为步长,$\boldsymbol{C}$ 是协方差矩阵,反映了种群分布的形状。CMA-ES 算法的流程如算法 6 所示[68]。

算法 6 CMA-ES 算法

输入:适应度函数 F(x),问题搜索空间的维度 N,子代个体数(种群规模)m,父代个体数(种群中选定的搜索点的数量)n,阻尼因子 d_σ,学习率 c_σ,c_c,c_1,c_n,重组权值 w_i,最大迭代次数 G

输出:最优个体

1. 初始化:均值 $\boldsymbol{\mu}^{(0)}\in\mathbf{R}^N$,步长 $\sigma^{(0)}\in R_+$;演化路径 $\mathbf{p}_\sigma^{(0)}=\mathbf{p}_c^{(0)}=0$;协方差矩阵 $\mathbf{C}^{(0)}=\mathbf{I}\in\mathbf{R}^{N\times N}$;进化代数 $g^{(0)}=0$。
2. for g←1 to G do
3. for k←1 to m do
4. 种群采样:采样计算式为:

$$\mathbf{x}_k^{(g+1)}=\boldsymbol{\mu}^{(g)}+\sigma^{(g)}N_k(0,\mathbf{C}^{(g)}) \tag{17}$$

其中,$\mathbf{x}_k^{(g+1)}$ 是第 g+1 代中的第 k 个个体;$\boldsymbol{\mu}^{(g)}$ 是第 g 代种群分布的均值;$\sigma^{(g)}$ 是第 g 代的步长;$\mathbf{C}^{(g)}$ 是第 g 代的协方差矩阵。对 $\mathbf{C}$ 进行特征根分解,$\mathbf{C}=\mathbf{B}\mathbf{D}^2\mathbf{B}^T$,采样计算式变为:

$$\mathbf{x}_k^{(g+1)}=\boldsymbol{\mu}^{(g)}+\sigma^{(g)}\mathbf{B}^{(g)}\mathbf{D}^{(g)}N_k(0,\mathbf{I}) \tag{18}$$

其中,$\mathbf{B}$ 的各列是 $\mathbf{C}$ 的特征向量的标准正交基;$\mathbf{D}$ 的对角线元素是 $\mathbf{C}$ 的相应特征值的平方根。

5. 评价与选择,对子代个体计算适应度函数值 $F(x_k)$ 并排序,选择适应度排名靠前的 n 个个体组成当前最优子群。
6. 均值更新:对现有最优子群进行加权重组和均值更新。具体更新公式如下:

$$\boldsymbol{\mu}^{(g+1)}=\boldsymbol{\mu}^{(g)}+\sigma^{(g)}\langle\mathbf{y}\rangle_w^{(g)}=\sum_{i=1}^{n}w_i\,\mathbf{x}_{i;m}^{(g)} \tag{19}$$

其中,

$$\begin{cases}\langle\mathbf{y}\rangle_w^{(g)}=\sum_{i=1}^{n}w_i\,\mathbf{y}_{i;m}^{(g)}\\ \mathbf{y}_{i;m}^{(g)}=(\mathbf{x}_{i;m}^{(g)}-\boldsymbol{\mu}^{(g)})/\sigma^{(g)}\end{cases};\ \sum_{i=1}^{n}w_i=1,w_1>w_2>\cdots>w_n$$ 表示适应度越高的个体,其权重越大;$\mathbf{x}_{i;m}^{(g)}$ 是从 $\mathbf{x}_1^{(g)},\cdots,\mathbf{x}_m^{(g)}$ 中选出的适应度排名为第 i 的个体。

7. 协方差矩阵自适应调整:

$$\mathbf{p}_c^{(g+1)}=(1-c_c)\mathbf{p}_c^{(g)}+h_\sigma^{(g+1)}\sqrt{c_c(2-c_c)}\sqrt{n_{eff}}\frac{\boldsymbol{\mu}^{(g+1)}-\boldsymbol{\mu}^{(g)}}{\sigma^{(g)}} \tag{20}$$

$$\mathbf{C}^{(g+1)}=(1-c_1-c_n)\mathbf{C}^{(g)}+c_1(\mathbf{p}_c^{(g+1)}\mathbf{p}_c^{(g+1)T}+\delta(h_\sigma^{(g+1)})\mathbf{C}^{(g)})+c_n\sum_{i=1}^{n}w_i\,\mathbf{y}_{i;m}^{(g+1)}\mathbf{y}_{i;m}^{(g+1)T} \tag{21}$$

其中,c_c 是 $\mathbf{p}_c$ 的更新学习率;n_{eff} 是方差有效选择质量;c_1 和 c_n 分别为 $\mathbf{C}$ 的秩 1 和秩 n 的更新学习速率;h_σ 是 Heaviside 函数,即:

$$h_\sigma=\begin{cases}1, & \frac{\|\mathbf{p}_\sigma\|}{\sqrt{1-(1-c_\sigma)^{(g+1)}}}<\left(1.4+\frac{2}{N+1}\right)E(\|N(0,\mathbf{I})\|)\\ 0, & \text{其他}\end{cases} \tag{22}$$

其中,$E(\|N(0,\mathbf{I})\|)$ 是归一化演化路径在随机选择下的期望值,h_σ 用于防止 $\mathbf{p}_c$ 增长过大;$\delta(h_\sigma)=(1-h_\sigma)c_c(2-c_c)$。

8. 步长控制:

$$\sigma^{(g+1)}=\sigma^{(g)}\exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma^{(g+1)}\|}{E(\|N(0,\mathbf{I})\|)}-1\right)\right) \tag{23}$$

其中,c_σ 是 $\mathbf{p}_\sigma$ 的更新学习率;d_σ 是步长更新的阻尼因子,$d_\sigma\approx1$。

9. end for
10. end for
11. 输出最优个体

CMA-ES算法不需要计算目标函数的梯度，因此也适用于非光滑和非线性问题；且采用了一种自适应的策略，能够更好地利用历史信息，并加速算法的收敛。但CMA-ES算法也存在不足：算法参数较多，自适应过程比较复杂[69]，且需要频繁地进行协方差矩阵的分解，计算代价较高。

3.6 超参数算法的总结比较

本节根据Falkner等[23]提出的优秀的超参数优化方法需满足的5个要求（强大时间性能、优越的最终结果、有效利用并行资源、具有可拓展性以及具有稳健性和灵活性）对上述超参数优化方法进行比较评价。首先对于网格搜索和随机搜索来说，两者的原理简单并且能够适用于多种类型超参数同时调优，满足灵活性，但是网格搜索会因为待优化超参数种类过多、单个超参数可能的取值过多而需要消耗庞大的计算资源进而降低效率，随机搜索在这方面做出了极大的改进；此外，网格搜索和随机搜索都十分依赖人为事先枚举出的每种超参数可能的取值，存在遗漏全局最优超参数值组合的可能性，因此对于网格搜索来说，只有规定的超参数范围足够大并且粒度足够细时，优化最终结果可能是较为优越的结果。此外，网格搜索仅在某种情况下支持并行搜索，即当每一种参数组合互相独立且互不影响时，可以开启多线程进行网格搜索，此种情况在简单的传统机器学习模型中较为常见，但是当同时优化具有相关关系的超参数时（例如LightGBM中的最大深度和叶子节点数）可能无法并行实现。

对于贝叶斯优化，它与网格搜索和随机搜索一样能够对多种类型的超参数同时进行优化，具有灵活性，并且相比网格搜索和随机搜索，贝叶斯优化能够在人为给定的超参数搜索范围内实现全局最优超参数组合的寻找，而且优化速度更快。但贝叶斯优化的一个明显缺陷是无法进行并行计算，这是它需要通过不断的采样来建模超参数值组合与验证集损失函数值之间的非线性关系所导致的。对于属于多保真优化技术的Hyperband和BOHB来说，它们相比贝叶斯优化最为明显的优势就是能够实现并行计算，还将有限的计算资源考虑到超参数优化过程中，因此能够提升优化算法的整体效率。

对于元启发式算法，遗传算法不依赖于初始解，不易受噪声和局部最优解的影响，其稳健性较强，且交叉和变异操作使其不容易陷入局部最优解，增强了全局搜索能力。此外，遗传算法适用于多种优化问题，如连续、离散、组合等多种类型的优化问题。但遗传算法也存在明显的缺陷，其收敛速度慢且不能并行计算，由于遗传算法需要对整个种群进行操作，因此在种群规模较大时，可能需要较长时间才能收敛到最优解。粒子群算法虽然能够实现并行计算，但是由于其比较依赖初始值，因此极易陷入局部最优，从而导致算法稳健性较差。CMA-ES算法具有良好的全局搜索能力和对高维度问题的适应性，收敛速度也较快，但计算代价较高，需要较高的计算能力，同时算法的参数较多，超参数的设置可能会对算法的性能产生较大的影响。以上各种超参数优化方法对5种要求的比较汇总结果如表2所列。

超参数的选择还会受到具体模型、数据集以及最终任务的影响。从模型方面出发，不同的模型其内含超参数数量不同，例如相比其他传统机器学习模型算法来说，Boosting模型的超参数较多，超参数值可能还存在复杂的相关关系；从数据方面出发，对于不同的数据分布设置不同的超参数也会影响模型最后的表现，以二分类任务为例，如果数据集中正例（标签为1）占比较小而负例（标签为0）的数据不平衡的情况发生，那么对于超参数的调整就应该偏向提升模型的泛化性能，即通过调整超参数使得模型精确率、召回率、F1得分等指标获得较大提升，此时下游的任务也会反向影响超参数优化。因此，面对以上情况，本文在第4部分设置实际数据实验，旨在对以上8种超参数优化算法在相同模型、相同数据集中统一比较优化效果。

表2 超参数优化算法的比较结果

Table 2 Comparison results of hyperparameter optimization algorithms

优化算法	时间性能	优越结果	并行能力	可拓展性	稳健性&灵活性
GS	×	×	×	√	√
RS	√	×	×	√	√
BO	√	√	×	√	√
Hyperband	√	√	√	√	×
BOHB	√	√	√	√	√
GA	×	√	×	√	√
PSO	×	√	√	√	×
CMA-ES	√	√	×	√	√

4 实际数据实验

本章设置3类实验以比较上述提到的网格搜索、随机搜索、贝叶斯优化、多保真优化以及元启发式算法在优化LightGBM、XGBoost、随机森林以及K近邻4种传统机器学习模型超参数优化任务中的性能。为保证实验结果的合理性，不同实验的机器学习模型设置不同的超参数搜索范围，相同实验中每个机器学习模型的训练集、验证集和测试集保持一致，最终通过比较默认设置下和经过超参数优化之后的机器学习模型在测试集中的测试指标，以及使用不同超参数优化方法下模型在测试集中的优化用时来综合评价各种超参数优化方法。

4.1 实验设置

本节共设置了3类实验任务对超参数优化技术进行验证，选取4个标准数据集进行实验，分别为使用波士顿房价数据集[70]、kin8nm动力臂数据集[71]的回归任务实验，使用信用卡违约数据集[72]的二分类任务以及使用MNIST手写数字数据集[70]的多分类任务。

4.1.1 回归任务设置

对于波士顿房价数据集，选取303条数据作为训练集用于训练预测模型，选取101条数据作为验证集计算验证误差，其余101条数据作为测试集检验超参数优化结果。待优化的机器学习模型有LightGBM、XGBoost、随机森林以及KNN回归模型，待优化超参数的搜索范围如表3所列。

表 3 波士顿房价数据集任务下机器学习模型超参数的搜索范围

Table 3 Hyperparameter search scope of machine learning model under Boston housing price dataset task

机器学习模型	超参数名称	取值类型	搜索范围
LightGBM	learning_rate	连续型	0.001～0.9
	min_data_in_leaf	离散型	10～30
	min_child_weight	连续型	0.001～0.1
	max_bin	离散型	100～300
	num_leaves	离散型	20～50
	reg_alpha	连续型	0～10
	reg_lambda	连续型	0～10
	max_depth	离散型	1～5
	n_estimators	离散型	100～500
XGBoost	learning_rate	连续型	0.01～0.9
	colsample_bytree	连续型	0.5～1.0
	min_child_weight	连续型	0.001～2
	gamma	连续型	0～20
	reg_alpha	连续型	0～10
	reg_lambda	连续型	0～10
	max_depth	离散型	1～10
	n_estimators	离散型	100～500
RF	max_depth	离散型	1～10
	n_estimators	离散型	100～500
	min_samples_spilt	离散型	2～10
	min_samples_leaves	离散型	1～10
	max_features	离散型	1～5
KNN	n_neighbors	离散型	2～20

对于 kin8nm 数据集,选取 6530 条数据作为训练集,用于训练预测模型,将 838 条数据作为验证集,用于计算验证误差,将其余数据作为测试集以检验超参数优化结果。待优化的机器学习模型有 LightGBM、XGBoost、随机森林以及 KNN 回归模型,待优化超参数的搜索范围如表 4 所列。

表 4 kin8nm 数据集任务下机器学习模型的超参数搜索范围

Table 4 Hyperparameter search scope of machine learning model under kin8nm dataset task

机器学习模型	超参数名称	取值类型	搜索范围
LightGBM	learning_rate	连续型	0.01～0.9
	min_date_in_leaf	离散型	10～30
	min_child_weight	连续型	0.001～0.1
	max_bin	离散型	100～300
	num_leaves	离散型	20～50
	reg_alpha	连续型	0～10
	reg_lambda	连续型	0～10
	max_depth	离散型	1～5
	n_estimators	离散型	100～500
XGBoost	learning_rate	连续型	0.001～0.9
	colsample_bytree	连续型	0.5～1.0
	min_child_weight	连续型	0.001～2
	gamma	连续型	0～20
	reg_alpha	连续型	0～10
	reg_lambda	连续型	0～10
	max_depth	离散型	1～10
	n_estimators	离散型	100～500
RF	max_depth	离散型	1～10
	n_estimators	离散型	100～500
KNN	n_neighbors	离散型	2～10

为了比较不同超参数优化方法的性能,两个回归任务的实验结果都输出了在默认设置下和配置各种超参数优化方法所得的最优超参数组合的机器学习模型在测试集上的 MSE、RMSE、MAE、拟合优度 R2 以及优化超参数所需时间 T 等评价指标。

4.1.2 二分类任务设置

对于信用卡违约客户数据集,选取 18000 条数据作为训练集用于训练预测模型,将 6000 条数据作为验证集用于计算验证误差,将 6000 条数据作为测试集以检验超参数的优化结果。待优化的机器学习模型有 LightGBM、XGBoost、随机森林以及 KNN 分类器,待优化超参数的搜索范围如表 5 所列。

表 5 信用卡违约客户数据集任务下机器学习模型的超参数搜索范围

Table 5 Hyperparameter search scope of machine learning model under credit card defaulting customer dataset task

机器学习模型	超参数名称	取值类型	搜索范围
LightGBM	learning_rate	连续型	0.01～0.2
	min_data_in_leaf	离散型	1～10
	min_child_weight	连续型	1～10
	max_bin	离散型	5～15
	reg_alpha	连续型	0.01～10
	reg_lambda	连续型	0.1～10
	max_depth	离散型	1～2
	n_estimators	离散型	300～500
XGBoost	learning_rate	连续型	0.001～0.2
	colsample_bytree	连续型	0.5～1.0
	min_child_weight	连续型	1～10
	gamma	连续型	0～20
	reg_alpha	连续型	0.01～10
	reg_lambda	连续型	0.1～10
	max_depth	离散型	1～2
	n_estimators	离散型	300～500
RF	max_depth	离散型	2～10
	min_samples_spilt	连续型	2～10
	min_samples_leaf	连续型	1～10
KNN	n_neighbors	离散型	2～20

为了全方位地评价二分类机器学习模型在测试集上的性能,实验结果输出了默认设置和配置各种超参数优化方法所得的最优超参数组合的机器学习模型在测试集上的准确率(Accuracy)、精确率(Precision)、召回率(Recall)、F1 得分(F1 score)以及优化超参数所需时间 T 等评价指标。在二分类问题中精确率代表模型预测正例中真实为正例的比例;召回率代表在真实为正例中,模型预测正确的比例;F1 得分是召回率与精确率的调和平均数,这个系数认为精确率和召回率同等重要,只有两者均很大时,F1 得分才会高,这时二分类模型的预测效果才会好。

4.1.3 多分类任务设置

对于手写数字数据集,选取 1257 张图片作为训练集用于训练预测模型,将 270 张图片作为验证集计用于算验证误差,将 270 张图片作为测试集以检验超参数优化结果。待优化的机器学习模型有 LightGBM、XGBoost、随机森林以及 KNN 分类模型,待优化超参数搜索范围如表 6 所列。

由于此任务为多分类任务,精确率、召回率、F1 得分 3 种评价指标不再适用,因此只输出了配置各种超参数优化方法所得的最优超参数组合的机器学习模型在测试集上的准确率以及优化超参数所需时间 T 等评价指标。

表 6 手写数字数据集任务下各个机器学习模型超参数的搜索范围

Table 6 Hyperparameter search scope of machine learning model under handwritten digits dataset task

机器学习模型	超参数名称	取值类型	搜索范围
LightGBM	learning_rate	连续型	0.001～0.5
	min_data_in_leaf	离散型	5～50
	min_child_weight	连续型	0.0001～1
	max_bin	离散型	100～500
	reg_alpha	连续型	0～1.0
	reg_lambda	连续型	0～1.0
	max_depth	离散型	1～10
	n_estimators	离散型	100～800
	num_leaves	离散型	20～50
XGBoost	learning_rate	连续型	0.001～0.5
	colsample_bytree	连续型	0～1.0
	min_child_weight	连续型	1～5
	gamma	连续型	0～0.5
	reg_alpha	连续型	0～1.0
	reg_lambda	连续型	0～1.0
	max_depth	离散型	1～10
	n_estimators	离散型	100～500
RF	max_depth	离散型	5～50
	min_samples_spilt	离散型	2～10
	min_samples_leaf	离散型	1～5
	n_estimators	离散型	50～300
	max_features	离散型	1～20
KNN	n_neighbors	离散型	2～10

4.2 性能比较

表 7－表 22 列出了 3 类实验的实验结果。其中，表 7－表 10、表 11－表 14、表 15－表 18 以及表 19－表 22 分别列出了 LightGBM、XGBoost、随机森林和 KNN 模型在波士顿房价数据集、kin8nm 数据集、信用卡违约客户数据集以及手写数字数据集上的评价指标。在性能方面，与默认设置相比，运用不同的超参数优化方法均使得 4 种模型在测试集上呈现出不同程度的提升，例如在波士顿房价数据集测试集上超参数优化方法最多可降低机器学习模型约 38%的均方误差，在信用卡违约客户数据集上 KNN 二分类模型的准确率最多提升了 20%。然而，在波士顿房价数据集上使用超参数优化技术优化 KNN 回归器的超参数时，出现了优化结果不如默认参数设置的情况，原因可能是 KNN 模型只有一种较为重要的超参数，即选取的近邻数，当近邻数过大时会提高模型泛化性能，但会增大偏差，而近邻数选取较小时则会降低泛化性能，因此出现了使用超参数优化方法来优化一种超参数的模型性能不如默认参数时的情况。

表 7 LightGBM 超参数优化结果在波士顿房价数据集测试集上的表现

Table 7 Performance of LightGBM hyperparameter optimization results in Boston house price dataset

优化算法	*MSE*	*RMSE*	*MAE*	R^2	*T*/s
Default	17.7529	4.2143	2.7820	0.7736	0.1
GS	14.1291	3.7589	2.5460	0.8198	269.1
RS	14.0735	3.7515	2.3971	0.8205	1.2
BO-TPE	14.4642	3.8032	2.4921	0.8155	6.2
Hyperband	14.0576	3.7493	2.4134	0.8207	6.1
BOHB	**13.4067**	**3.6615**	**2.2680**	**0.8290**	8.6
GA	14.0557	3.7491	2.2838	0.8208	3.2
PSO	13.9698	3.7376	2.4228	0.8218	1.4
CMA-ES	13.7113	3.7029	2.4879	0.8251	**0.8**

表 8 XGBoost 超参数优化结果在波士顿房价数据集测试集上的表现

Table 8 Performance of XGBoost hyperparameter optimization results in Boston house price dataset

优化算法	*MSE*	*RMSE*	*MAE*	R^2	*T*/s
Default	17.7529	4.2143	2.7820	0.7736	0.3
GS	**13.2665**	**3.6423**	**2.3091**	**0.8308**	2864.9
RS	13.6016	3.6880	2.5922	0.8265	8.0
BO-TPE	17.5187	4.1855	2.6985	0.7766	35.8
Hyperband	16.7591	4.0968	2.5053	0.7863	190.3
BOHB	14.8977	3.8598	2.9063	0.8100	164.3
GA	13.8393	3.7201	2.4300	0.8235	14.6
PSO	13.7051	3.7020	2.4688	0.8252	5.3
CMA-ES	13.5427	3.6800	2.6863	0.8273	**3.6**

表 9 随机森林超参数优化结果在波士顿房价数据集测试集上的表现

Table 9 Performance of random forest hyperparameter optimization results in Boston house price dataset

优化算法	*MSE*	*RMSE*	*MAE*	R^2	*T*/s
Default	18.6467	4.3182	2.6980	0.7622	0.7
GS	12.1477	3.4853	2.3721	0.8451	338.8
RS	11.9020	3.4499	2.3135	0.8482	15.1
BO-TPE	11.6995	3.4204	2.3160	0.8508	68.3
Hyperband	12.6241	3.5530	2.4338	0.8390	17.8
BOHB	14.0306	3.7457	2.5209	0.8211	33.4
GA	11.8856	3.4476	2.3698	0.8484	36.3
PSO	11.5297	3.3955	2.3446	0.8530	18.5
CMA-ES	**11.4283**	**3.3806**	**2.3437**	**0.8543**	**10.3**

表 10 KNN 超参数优化结果在波士顿房价数据集测试集上的表现

Table 10 Performance of KNN regression hyperparameter optimization results in Boston house price dataset

优化算法	*MSE*	*RMSE*	*MAE*	R^2	*T*/s
Default	**38.9760**	**6.2431**	**4.6849**	**0.5029**	0.20
GS	43.2550	6.5769	4.8402	0.4483	0.30
RS	40.2721	6.3460	4.6351	0.4864	0.10
BO-TPE	40.2721	6.3460	4.6351	0.4864	1.20
Hyperband	43.2550	6.5769	4.8402	0.4483	2.90
BOHB	43.2550	6.5769	4.8402	0.4483	4.30
GA	40.2721	6.3460	4.6351	0.4864	**0.08**
PSO	39.5595	6.2896	4.7196	0.4955	0.15
CMA-ES	40.2721	6.3460	4.6351	0.4864	0.12

表 11 LightGBM 超参数优化结果在 kin8nm 数据集测试集上的表现

Table 11 Performance of LightGBM hyperparameter optimization results in kin8nm dataset

优化算法	*MSE*	*RMSE*	*MAE*	R^2	*T*/s
Default	0.0194	0.1395	0.1091	0.7080	0.1
GS	0.0132	0.1149	0.0878	0.8017	933.2
RS	0.0157	0.1252	0.0969	0.7647	5.8
BO-TPE	0.0126	0.1122	0.0971	0.8108	17.9
Hyperband	**0.0132**	**0.1148**	**0.0886**	**0.8018**	60.3
BOHB	0.0183	0.1354	0.1054	0.7245	5.5
GA	0.0154	0.1241	0.0945	0.7687	8.6
PSO	0.0167	0.1293	0.0986	0.7491	7.2
CMA-ES	0.0151	0.1229	0.0921	0.7762	**2.3**

表 12 XGBoost 超参数优化结果在 kin8nm 数据集测试集上的表现

Table 12 Performance of XGBoost hyperparameter optimization results in kin8nm dataset

优化算法	*MSE*	*RMSE*	*MAE*	R^2	*T*/s
Default	0.0163	0.1277	0.0980	0.7550	0.4
GS	0.0167	0.1292	0.0999	0.7923	17127.8
RS	0.0143	0.1196	0.0987	0.7932	67.6
BO-TPE	0.0134	0.1161	0.0904	0.7976	155.5
Hyperband	0.0150	0.1229	0.0954	0.7732	36.6
BOHB	0.0392	0.1980	0.1572	0.4119	117.3
GA	**0.0126**	**0.1123**	**0.0874**	**0.8106**	116.3
PSO	0.0138	0.1175	0.0932	0.8021	112.5
CMA-ES	0.0135	0.1162	0.0903	0.8031	**33.7**

表 13 随机森林超参数优化结果在 kin8nm 数据集测试集上的表现

Table 13 Performance of random forest hyperparameter optimization results in kin8nm dataset

优化算法	*MSE*	*RMSE*	*MAE*	R^2	*T*/s
Default	0.0196	0.1399	0.1100	0.7060	3.1
GS	0.0192	0.1385	0.1082	0.7122	1917.1
RS	0.0189	0.1376	0.1078	0.7157	408.5
BO-TPE	0.0192	0.1385	0.1082	0.7122	508.4
Hyperband	0.0193	0.1390	0.1089	0.7100	253.6
BOHB	0.0192	0.1385	0.1082	0.7122	454.4
GA	0.0191	0.1383	0.1080	0.7130	266.5
PSO	0.0191	0.1380	0.1078	0.7139	420.0
CMA-ES	**0.0188**	**0.1373**	**0.1073**	**0.7171**	**160.7**

表 14 KNN 超参数优化结果在 kin8nm 数据集测试集上的表现

Table 14 Performance of KNN regression hyperparameter optimization results in kin8nm dataset

优化算法	*MSE*	*RMSE*	*MAE*	R^2	*T*/s
Default	0.0142	0.1192	0.0934	0.7864	0.3
GS	0.0130	0.1141	0.0901	0.8047	3.2
RS	0.0130	0.1141	0.0901	0.8047	7.6
BO-TPE	0.0132	0.1148	0.0910	0.8021	8.6
Hyperband	0.0132	0.1148	0.0910	0.8021	70.9
BOHB	0.0132	0.1148	0.0910	0.8021	6.2
GA	0.0130	0.1141	0.0901	0.8047	1.8
PSO	0.0130	0.1141	0.0901	0.8047	6.2
CMA-ES	**0.0130**	**0.1141**	**0.0901**	**0.8047**	**1.5**

表 15 LightGBM 超参数优化结果在信用卡违约数据集测试集上的表现

Table 15 Performance of LightGBM hyperparameter optimization results in credit card default customer dataset

优化算法	*Accuracy*	*Precision*	*Recall*	*F*1 *score*	*T*/s
Default	0.8197	0.6430	0.3827	0.4798	0.41
GS	0.8228	0.6613	0.3788	0.4817	303.7
RS	0.8227	0.6671	0.3673	0.4738	**10.4**
BO-TPE	0.8225	0.6648	0.3696	0.4751	16.2
Hyperband	0.8125	0.6507	**0.3857**	**0.4844**	30.2
BOHB	0.8221	0.6639	0.3681	0.4736	29.4
GA	0.8242	0.6736	0.3704	0.4780	42.4
PSO	0.8222	0.6681	0.3612	0.4689	30.1
CMA-ES	**0.8238**	**0.6747**	0.3658	0.4744	14.6

表 16 XGBoost 超参数优化结果在信用卡违约数据集测试集上的表现

Table 16 Performance of XGBoost hyperparameter optimization results in credit card default customer dataset

优化算法	*Accuracy*	*Precision*	*Recall*	*F*1 *score*	*T*/s
Default	0.8147	0.6197	0.3811	0.4720	2.27
GS	0.8222	**0.6725**	0.3543	0.4641	11176.80
RS	0.8205	0.6561	0.3658	0.4697	58.02
BO-TPE	**0.8233**	0.6685	0.3712	**0.4773**	**54.10**
Hyperband	0.8197	0.6453	**0.3781**	0.4768	66.30
BOHB	0.8227	0.6671	0.3673	0.4738	138.10
GA	0.8222	0.6639	0.3681	0.4736	179.21
PSO	0.8212	0.6648	0.3574	0.4648	99.32
CMA-ES	0.8228	0.6690	0.3658	0.4730	63.18

表 17 随机森林超参数优化结果在信用卡违约数据集测试集上的表现

Table 17 Performance of Random Forest hyperparameter optimization results in credit card default customer dataset

优化算法	*Accuracy*	*Precision*	*Recall*	*F*1 *score*	*T*/s
Default	0.8185	0.6366	**0.3842**	0.4792	8.09
GS	0.8161	0.6823	0.2883	0.4053	215.40
RS	0.8223	0.6662	0.3658	0.4723	48.40
BO-TPE	**0.8227**	0.6630	0.3742	**0.4784**	**17.80**
Hyperband	0.8225	0.6775	0.3497	0.4613	80.70
BOHB	0.8153	**0.6835**	0.2799	0.3972	49.60
GA	0.8218	0.6634	0.3658	0.4716	96.50
PSO	0.8220	0.6720	0.3535	0.4633	131.00
CMA-ES	0.8222	0.6630	0.3696	0.4746	55.20

表 18 KNN 超参数优化结果在信用卡违约数据集测试集上的表现

Table 18 Performance of KNN classification hyperparameter optimization results in credit card default customer dataset

优化算法	*Accuracy*	*Precision*	*Recall*	*F*1 *score*	*T*/s
Default	0.7527	0.3562	0.1710	0.2310	3.1
GS	0.7715	0.4087	0.1150	0.1795	67.8
RS	0.7798	0.4564	0.0683	0.1187	61.6
BO-TPE	0.7758	0.4163	0.0782	0.1317	105.4
Hyperband	0.7758	0.4163	0.0782	0.1317	102.1
BOHB	0.7758	0.4163	0.0782	0.1317	262.8
GA	0.7740	0.4037	**0.0836**	**0.1385**	**39.3**
PSO	**0.7833**	**0.5120**	0.0652	0.1156	96.1
CMA-ES	0.7830	0.5062	0.0629	0.1119	67.4

表 19 LightGBM 超参数优化结果在手写数字数据集测试集上的表现

Table 19 Performance of LightGBM hyperparameter optimization results in handwritten digit dataset

优化算法	*Accuracy*	*T*/s
Default	0.9630	1.4
GS	0.9630	1767.2
RS	0.9630	46.3
BO-TPE	0.9704	65.3
Hyperband	0.9667	85.1
BOHB	0.9704	55.8
GA	0.9667	124.4
PSO	0.9704	54.1
CMA-ES	**0.9704**	**24.3**

表 20 XGBoost 超参数优化结果在手写数字数据集测试集上的表现

Table 20 Performance of XGBoost hyperparameter optimization results in handwritten digit dataset

Algorithm	*Accuracy*	*T*/s
Default	0.9519	0.53
GS	0.9630	1591.60
RS	0.9593	88.10
BO-TPE	0.9630	53.60
Hyperband	0.9593	29.40
BOHB	0.9593	**24.10**
GA	**0.9667**	166.80
PSO	0.9630	57.20
CMA-ES	0.9630	52.20

表 21 随机森林超参数优化结果在手写数字数据集测试集上的表现

Table 21 Performance of random forest hyperparameter optimization results in handwritten digit dataset

Algorithm	*Accuracy*	*T*/s
Default	0.9667	0.6
GS	0.9740	243.1
RS	0.9704	17.6
BO-TPE	0.9740	108.5
Hyperband	0.9778	28.7
BOHB	0.9741	31.1
GA	**0.9778**	35.0
PSO	0.9704	**6.3**
CMA-ES	0.9704	15.6

表 22 KNN 超参数优化结果在手写数字数据集测试集上的表现

Table 22 Performance of KNN hyperparameter optimization results in handwritten digit dataset

Algorithm	*Accuracy*	*T*/s
Default	0.9778	0.8
GS	0.9741	2.1
RS	0.9815	0.9
BO-TPE	0.9778	8.7
Hyperband	0.9815	2.1
BOHB	0.9815	3.3
GA	**0.9815**	**0.5**
PSO	0.9815	1.0
CMA-ES	0.9815	0.6

以上情况也说明了面对超参数较多的机器学习模型时，使用超参数优化技术进行调参是必须的，而并非所有机器学习模型都需要利用理论和算法结构复杂的超参数优化技术进行调优。

由上述结果可以发现，使用网格搜索和随机搜索能够极大地提升机器学习模型的泛化性能，并且相比其他超参数优化方法，在尽可能给出较大范围的搜索空间时，GS 和 RS 有可能给出相比其他超参数优化技术更优的超参数组合，并且在相同的搜索空间中，RS 相比 GS 计算时间更短。然而，在很多情况下，GS 相比其他超参数优化方法优化时间会更长，尤其是在优化 LightGBM 和 XGBoost 这种超参数较多的集成学习模型的情况下，时间的增加更为明显。因此，这两种方法适用于超参数搜索空间较小的情况，或者在初始探索阶段用于快速定位一些较好的超参数组合。由于它们在给定范围内进行穷举搜索，会花费大量的计算资源和时间，因此在有充足的资源且超参数搜索空间不是很大时，网格搜索和随机搜索可以提供较好的结果。

BO-TPE 与两种多保真方法 Hyperband 和 BOHB 相比模型默认设置具有较大提升；通过对三者进行内部比较发现，对于超参数种类较多的 LightGBM 和 XGBoost 模型，BO-TPE 的表现更优，具体表现为：在回归实验中，LightGBM 和 XGBoost 模型使用 BO-TPE 得到的超参数组合的 MSE，RMSE，MAE 都要比使用 Hyperband 和 BOHB 更低；在二分类和多分类实验中，使用 BO-TPE 得到的超参数组合的 LightGBM 和 XGBoost 模型的分类准确率更高。对应地，当机器学习模型需要优化的超参数数量较少时，如随机森林和 KNN，多保真方法将会比 BO-TPE 的优化效果更好。此外，通过比较 Hyperband 和 BOHB 优化用时可以发现，BOHB 优化用时在大部分情况下都少于 Hyperband，这是因为 BOHB 在采样过程中加入了先验知识，从而加速了收敛。总的来说，BO-TPE 适用于具有较大的超参数搜索空间和复杂的学习任务，其采用贝叶斯模型建模超参数和性能之间的关系，从而更加高效地探索优化空间。Hyperband 和 BOHB 更加适合超参数较少且优化时间有限的情况，因为它们的收敛速度较快。

元启发式算法相比模型默认设置也具有较大提升。比较 3 种元启发式算法可以发现，遗传算法普遍耗时更多，但有时能表现出更优的结果，粒子群优化算法因为受初始值影响较大更不稳定，而 CMA-ES 算法无论在回归还是分类任务上都能表现出良好的稳定性，模型使用 CMA-ES 优化得到的超参数组合，经过训练后在测试集上的表现都较为优越，而且相比遗传算法和粒子群优化算法，CMA-ES 算法优化超参数的时间更短。从实验结果来看，与 BO-TPE 和两种多保真算法相比，元启发式算法在超参数优化中更容易得到更优的结果（在 16 组结果中有 11 组最优结果均来自于元启发式算法），这可能是由于其采用的搜索策略更加智能（如遗传算法中的选择交叉变异操作和 CMA-ES 中的协方差矩阵自适应调整），使其在实际应用中更有可能找到更接近全局最优解的超参数组合，从而在机器学习模型的性能提升方面具有潜在优势。

综合来看，超参数优化方法的性能依赖于学习任务和数据分布特性。当超参数搜索范围给得足够大并且详细时，网格搜索和随机搜索的超参数优化结果会很好，但是会以巨大的计算资源和超长的计算时间作为代价；对于 BO-TPE，当机器学习模型超参数个数较多并且搜索范围较大时，其超参数优化效果比多保真优化算法 Hyperband 和 BOHB 更优；而待优化超参数较少时，Hyperband 和 BOHB 更加适合并且 BOHB 优化所需时间会更短。元启发式算法适用于多种学习任务，在超参数优化问题上表现较好，CMA-ES 算法相比遗传算法和粒子群优化算法效果更好，更加稳定且耗时也更短。结合第 3 章对超参数 5 个评价指标的分析可以发现，对于超参数较多的 Boosting 模型 BOHB、Hyperband、CMA-ES 等方法都表现出了相比默认设置和网格搜索、随机搜索方法在上述实际数据模拟中更为优越的性能；在优化时间方面，CMA-ES 在多个任务中都呈现出了相比其他算法更短的优化时间，这与定性分析中该方法的时间性能吻合。

5 结论

在机器学习广泛运用的今天,合理调整超参数能够保证机器学习模型在应对不同情景、不同任务时总能保持良好的性能。然而,当机器学习模型较为复杂且超参数种类较多时,手动调节方法会变得效率低下,从而需要寻求更合适机器学习模型的自动超参数优化方法的帮助。本文回顾了当前较为常用的8种超参数优化方法,包括网格搜索、随机搜索、贝叶斯优化、Hyperband、BOHB、遗传算法、粒子群优化算法和CMA-ES算法,先后介绍了各类算法的基本理论和算法流程,然后按照时间性能、最终结果、并行计算、可拓展性、稳健性和灵活性6项标准分别分析各类方法的优劣,并将各类方法运用到实际数据集中进行尝试,比较计算结果。总的来说,网格搜索和随机搜索适用于超参数搜索空间较小的情况,或者在初始探索阶段用于快速定位一些较好的超参数组合。BO-TPE适用于具有较大的超参数搜索空间和复杂的学习任务。Hyperband和BOHB的优化时间相比贝叶斯优化有所缩短,适合超参数较少且优化时间有限的情况。元启发式算法的效果普遍较优,尤其是CMA-ES算法,耗时较少且稳定,适用于多种学习任务,在综合性能上的表现较为优秀。

结束语 本文回顾了8种常见的超参数优化方法,分析了它们在时间性能、最终结果、并行能力、可拓展性、稳健性和灵活性等方面的优缺点,并进行了回归和分类的实验验证。尽管这些超参数优化方法在不同情景下都表现出一定的优势,但它们各自仍存在一些局限性。未来的研究工作可以从以下3个方面展开:首先,扩展现有超参数优化方法进行实验测试,包括基于强化学习和元学习方法等,并与现有的方法进行对比研究;其次,增加对生成型任务的实验,确保实验设计的全面性和准确性;此外,考虑增加对深度学习相关模型的超参数优化方法的比较和验证,更加全面地比较各类方法的差异,为当前学术界和产业界更加关心的深度学习模型的亿级超参数的优化问题提供有价值的参考。

参 考 文 献

[1] ATHEY S. The Impact of Machine Learning on Economics [M]// The Economics of Artificial Intelligence: An agenda. University of Chicago Press, 2018: 507-547.

[2] WEI J, CHU X, SUN X Y, et al. Machine Learning in Materials Science[J]. InfoMat, 2019, 1(3): 338-358.

[3] CARLEO G, CIRAC I, CRANMER K, et al. Machine Learning and Physical Sciences[J]. Reviews of Modern Physics, 2019, 91(4): 045002.

[4] LIAKOS K G, BUSATA P, MOSHOU D, et al. Machine Learning in Agriculture: A Review[J]. Sensors, 2018, 18(8): 2674.

[5] TARCA A L, CAREY V J, CHEN X, et al. Machine Learning and Its Applications to Biology[J]. PLoS Computational Biology, 2007, 3(6): e116.

[6] CORTES C, VAPNIK V. Support-Vector Networks [J]. Machine Learning, 1995, 20: 273-297.

[7] BOSER B E, GUYON I M, VAPNIK V N. A Training Algorithm for Optimal Margin Classifiers[C]// Proceedings of the Fifth Annual Workshop on Computational Learning Theory. 1992: 144-152.

[8] COVER T, HART P. Nearest Neighbor Pattern Classification [J]. IEEE Transactions on Information Theory, 1967, 13(1): 21-27.

[9] HOSMER J R D W, LEMESHOW S, STURDIVANT R X. Applied Logistic Regression[M]. John Wiley & Sons, 2013.

[10] FRIEDMAN J H. Greedy Function Approximation: A Gradient Boosting Machine[J]. Annals of Statistics, 2001, 29: 1189-1232.

[11] FREUND Y, SCHAPIRE R E. A Decision-Theoretic Generalization of on-line Learning and an Application to Boosting[J]. Journal of Computer and System Sciences, 1997, 55(1): 119-139.

[12] CHEN T, GUESTRIN C. Xgboost: A Scalable Tree Boosting System[C]// Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining. 2016: 785-794.

[13] KE G, MENG Q, FINLEY T, et al. Lightgbm: A Highly Efficient Gradient Boosting Decision Tree[J]. Advances in Neural Information Processing Systems, 2017, 30: 3147-3155.

[14] PROKHORENKOVA L, GUSEV G, VOROBEV A, et al. CatBoost: Unbiased Boosting with Categorical Features[J]. Advances in Neural Information Processing Systems, 2018, 31: 6638-6648.

[15] HUTTER F, KOTTHOFF L, VANSCHOREN J. Automated Machine Learning: Methods, Systems, Challenges[M]. Springer Nature, 2019.

[16] LIASHCHYNSKYI P, LIASHCHYNSKYI P. Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS [J]. arXiv: 1912.06059, 2019.

[17] BERGSTRA J, BENGIO Y. Random Search for Hyper-Parameter Optimization[J]. Journal of Machine Learning Research, 2012, 13(1): 281-305.

[18] SNOEK J, LAROCHELLE H, ADAMS R P. Practical Bayesian Optimization of Machine Learning Algorithms[J]. Advances in Neural Information Processing Systems, 2012, 4: 2951-2959.

[19] BERGSTRA J, BARDENET R, BENGIO Y, et al. Algorithms for Hyper-parameter Optimization[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2011: 2546-2554.

[20] HUTTER F, HOOS H H, LEYTON-BROWN K. Sequential Model-based Optimization for General Algorithm Configuration [C]// Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5. Berlin, Heidelberg: Springer, 2011: 507-523.

[21] JAMIESON K, TALWALKAR A. Non-stochastic Best Arm Identification and Hyperparameter Optimization[C]// Artificial Intelligence and Statistics. PMLR, 2016: 240-248.

[22] LI L, JAMIESON K, DESALVO G, et al. Hyperband: A Novel Bandit-based Approach to Hyperparameter Optimization[J]. The Journal of Machine Learning Research, 2017, 18(1): 6765-6816.

[23] FALKNER S, KLEIN A, HUTTER F. BOHB: Robust and Efficient Hyperparameter Optimization at Scale[C]// International

Conference on Machine Learning. PMLR, 2018: 1437-1446.
[24] MITCHELL M. An Introduction to Genetic Algorithms[M]. MIT press, 1998.
[25] KENNEDY J, EBERHART R. Particle Swarm Optimization [C]// Proceedings of ICNN'95 International Conference on Neural Networks. IEEE, 1995, 4: 1942-1948.
[26] HANSEN N, OSTERMEIER A. Completely Derandomized Self-adaptation in Evolution Strategies[J]. Evolutionary Computation, 2001, 9(2): 159-195.
[27] WU J, CHEN S P, LIU X Y. Efficient Hyperparameter Optimization through Model-based Reinforcement Learning[J]. Neurocomputing, 2020, 409: 381-393.
[28] LIU X, WU J, CHEN S. A Context-based Meta-reinforcement Learning Approach to Efficient Hyperparameter Optimization [J]. Neurocomputing, 2022, 478: 89-103.
[29] WU J, LIU X, CHEN S. Hyperparameter Optimization through Context-based Meta-reinforcement Learning with Task-aware Representation [J]. Knowledge-Based Systems, 2023, 260: 110160.
[30] YANG L, SHAMI A. On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice[J]. Neurocomputing, 2020, 415: 295-316.
[31] LUO G. A Review of Automatic Selection Methods for Machine Learning Algorithms and Hyper-parameter Values[J]. Network Modeling Analysis in Health Informatics and Bioinformatics, 2016, 5: 1-16.
[32] LORENZO P R, NALEPA J, KAWULOK M, et al. Particle Swarm Optimization for Hyper-parameter Selection in Deep Neural Networks[C]// Proceedings of the Genetic and Evolutionary Computation Conference. 2017: 481-488.
[33] WITT C. Worst-case and Average-case Approximations by Simple Randomized Search Heuristics[C]// Annual Symposium on Theoretical Aspects of Computer Science. Berlin, Heidelberg: Springer, 2005: 44-56.
[34] LIU C, YIN S Q, ZHANG M, et al. An Improved Grid Search Algorithm for Parameters Optimization on SVM[J]. Applied Mechanics and Materials, 2014, 644: 2216-2219.
[35] SUN Y, DING S, ZHANG Z, et al. An Improved Grid Search Algorithm to Optimize SVR for Prediction[J]. Soft Computing, 2021, 25: 5633-5644.
[36] NAYEBI A, MUNTEANU A, POLOCZEK M. A Framework for Bayesian Optimization in Embedded Subspaces[C]// International Conference on Machine Learning. PMLR, 2019: 4752-4761.
[37] KIRSCHNER J, MUTNY M, HILLER N, et al. Adaptive and Safe Bayesian Optimization in High Dimensions Via One-dimensional Subspaces[C]// International Conference on Machine Learning. PMLR, 2019: 3429-3438.
[38] LIU Q, WANG D. Stein Variational Gradient Descent : A General Purpose Bayesian Inference Algorithm[J]. Advances in Neural Information Processing Systems, 2016, 29: 2378-2386.
[39] GONG C, PENG J, LIU Q. Quantile Stein Variational Gradient Descent for Batch Bayesian Optimization[C]// International Conference on Machine Learning. PMLR, 2019: 2347-2356.
[40] WANG Z, LI C, JEGELKA S, et al. Batched High-Dimensional Bayesian Optimization Via Structural Kernel Learning[C]// International Conference on Machine Learning. PMLR, 2017: 3656-3664.
[41] ROLLAND P, SCARLETT J, BOGUNOVIC I, et al. High-Dimensional Bayesian Optimization Via Additive Models with Overlapping Groups[C]// International Conference on Artificial Intelligence and Statistics. PMLR, 2018: 298-307.
[42] CONTAL E, BUFFONI D, ROBICQUET A, et al. Parallel Gaussian Process Optimization with Upper Confidence Bound and Pure Exploration[C]// Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Berlin, Heidelberg: Springer, 2013: 225-240.
[43] WANG Z, GEHRING C, KOHLI P, et al. Batched Large-scale Bayesian Optimization in High-dimensional Spaces[C]// International Conference on Artificial Intelligence and Statistics. PMLR, 2018: 745-754.
[44] SWERSKY K, SNOEK J, ADAMS R P. Multi-task Bayesian Optimization[J]. Advances in Neural Information Processing Systems, 2013, 26: 2004-2012.
[45] PEARCE M, BRANKE J. Continuous Multi-Task Bayesian Optimisation with Correlation[J]. European Journal of Operational Research, 2018, 270(3): 1074-1085.
[46] CHOWDHURY S R, GOPALAN A. No-regret Algorithms for Multi-task Bayesian Optimization[C]// International Conference on Artificial Intelligence and Statistics. PMLR, 2021: 1873-1881.
[47] GONZALEZ J, DAI Z, HENNIG P, et al. Batch Bayesian Optimization Via Local Penalization[C]// Artificial Intelligence and Statistics. PMLR, 2016: 648-657.
[48] GARRIDO-MERCHAN E C, HERNANDEZ-LOBATO D. Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes[J]. Neurocomputing, 2020, 380: 20-35.
[49] FEURER M, SPRINGENBERG J, HUTTER F. Initializing Bayesian Hyperparameter Optimization Via Meta-learning [C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2015.
[50] ROTHFUSS J, KOENIG C, RUPENYAN A, et al. Meta-learning Priors for Safe Bayesian Optimization[C]// Conference on Robot Learning. PMLR, 2023: 237-265.
[51] LAN G, TOMCZAK J M, ROIJERS D M, et al. Time Efficiency in Optimization with a Bayesian-evolutionary Algorithm[J]. Swarm and Evolutionary Computation, 2022, 69: 100970.
[52] HENSMAN J, FUSI N, LAWRENCE N D. Gaussian processes for Big data[C]// Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence. Arlington, Virginia, USA: AUAI Press, 2013: 282-290.
[53] ELSHAWI R, MAHER M, SAKR S. Automated Machine Learning: State-of-the-art and Open Challenges[J]. arXiv: 1906.02287, 2019.
[54] FERNANDEZ-GODINO M G, PARK C, KIM N H, et al. Review of Multi-fidelity Models[J]. arXiv: 1609.07196, 2016.
[55] LI L, JAMIESON K, ROSTAMIZADEH A, et al. A System for

Massively Parallel Hyperparameter Tuning[J]. Proceedings of Machine Learning and Systems,2020,2:230-246.

[56] SCHMUCKER R,DONINI M,ZAFAR M B,et al. Multi-objective Asynchronous Successive Halving[J]. arXiv:2106.12639,2021.

[57] AWAD N,MALLIK N,HUTTER F. Dehb:Evolutionary Hyperband for Scalable,Robust and Efficient Hyperparameter Optimization[J]. arXiv:2105.09821,2021.

[58] AWAD N,SHARMA A,HUTTER F. MO-DEHB:Evolutionary-based Hyperband for Multi-Objective Optimization[J]. arXiv:2305.04502,2023.

[59] STORN R,PRICE K. Differential Evolution:A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces[J]. Journal of Global Optimization,1997,11:341-359.

[60] JIA H,SUN K,ZHANG W,et al. An Enhanced Chimp Optimization Algorithm for Continuous Optimization Domains[J]. Complexand Intelligent Systems,2022,8(1):65-82.

[61] ZHAO W,WANG L,MIRJALILI S. Artificial Hummingbird Algorithm:A New Bio-inspired Optimizer with Its Engineering Applications[J]. Computer Methods in Applied Mechanics and Engineering,2022,388:114194.

[62] DENG W,SHANG S,CAI X,et al. An Improved Differential Evolution Algorithm and Its Application in Optimization Problem[J]. Soft Computing,2021,25:5277-5298.

[63] NOMURA M,WATANABE S,AKIMOTO Y,et al. Warm Starting CMA-ES for Hyperparameter Optimization[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2021:9188-9196.

[64] LESSMANN S,STAHLBOCK R,CRONE S F. Optimizing Hyperparameters of Support Vector Machines by Genetic Algorithms[C]// ICAI. 2005.

[65] LOBO F G,GOLDBERG D E,PELIKAN M. Time Complexity of Genetic Algorithms on Exponentially Scaled Problems[C]// Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation. 2000:151-158.

[66] YAN X H,HE F Z,CHEN Y L. A Novel Hardware/Software Partitioning Method Based on Position Disturbed Particle Swarm Optimization with Invasive Weed Optimization[J]. Journal of Computer Science and Technology,2017,32:340-355.

[67] AHANDANI M A,ALAVI-RAD H. Opposition-based Learning in the Shuffled Differential Evolution Algorithm[J]. Soft Computing,2012,16(8):1303-1337.

[68] QIAO S. Study and Application of CMA-ES Algorithm Based on Cloud Mode[D]. Taiyuan:Taiyuan University of Technology,2015.

[69] BEYER H G,SENDHOFF B. Covariance Matrix Adaptation Revisited—the CMSA Evolution Strategy[C]// International Conference on Parallel Problem Solving from Nature. Berlin,Heidelberg:Springer,2008:123-132.

[70] PEDREGOSA F,VARIQUAUX G,GRAMFORT A ,et al. Scikit-learn:Machine Learning in Python[J]. The Journal of Machine Learning Research,2011,12:2825-2830.

[71] SCHWAIGHOFER A,TRESP V. Transductive and Inductive Methods for Approximate Gaussian Process Regression[J]. Advances in Neural Information Processing Systems,2002,15(3):953-960.

[72] YEH I C,LIEN C. The Comparisons of Data Mining Techniques forPredictive Accuracy of Probability of Default of Credit Card Clients[J]. Expert Systems with Applications,2009,36(2):2473-2480.

LI Haixia, born in 1984, bachelor, researcher. Her main research interests include information system design and information fusion technology.

SONG Yafei, born in 1988, Ph.D, associate professor, postgraduate supervisor. His main research interests include intelligent reasoning and decision-making.

(责任编辑:喻藜)