自动化机器学习中的超参调优方法

献给方开泰教授 80 华诞

张爱军*, 杨泽斌

香港大学统计与精算学系, 香港

E-mail: ajzhang@hku.hk, yangzb2010@hku.hk

收稿日期: 2019-04-01; 接受日期: 2019-06-20; 网络出版日期: 2020-05-07; * 通信作者

摘要 本文介绍近年逐渐兴起的自动化机器学习框架, 着重讨论其中颇具挑战的超参调优问题. 常用的调参方法有格子点法、随机搜索等批量抽样策略, 还包括 Bayes 优化、群体搜索算法和强化学习等序贯策略. 由方开泰和王元早在 1990 年提出的贯序数论优化算法, 利用序贯均匀设计对复杂响应曲面寻求全局最优值, 同样适用于超参调优. 本文以支持向量机和极限梯度推进机这两种常用的机器学习模型为例, 结合两组典型的二分类数据集, 对多种超参调优方法进行测试. 通过比较分析发现一种改进的贯序数论优化算法, 对解决自动化机器学习中的调参问题, 颇具潜力.

关键词 自动化机器学习 超参调优 Bayes 优化 序贯均匀设计

MSC (2010) 主题分类 62K20, 68T05, 90C26

1 引言

近年来, 受益于数据规模和计算资源的快速增长, 复杂机器学习算法特别是深度神经网络得到越来越多的关注, 为数据科学和人工智能领域的发展带来了巨大变革. 随着机器学习与深度学习的不断发展, 模型变得越来越复杂, 不仅参数数量成百上千, 超参的数量也大幅上升. 超参选择对于机器学习模型的表现有着决定性作用: 即使对同一个模型, 不同超参组合的微小差异也可能对结果带来较大的变动. 同时, 训练机器学习模型是一个“昂贵”的过程. 在实践中, 复杂模型 (如深度神经网络) 的调参往往需要几天甚至更长的时间, 机器学习工程师们需要花费大量时间和精力进行枯燥乏味的手工调参, 从而限制了机器学习的普及应用. 因此, 如何高效地配置和训练机器学习模型成为新的挑战.

自动化机器学习 (automated machine learning, AutoML), 旨在实现机器学习实践中涉及的一系列步骤的自动化, 包括数据预处理、特征工程、模型选择和超参优化等. 图 1 勾勒了目前常用的一种 AutoML 框架, 其中超参优化是最重要的一个环节. 自动化超参调优相较于手工调参有很多优势, 例如, 它能够提高模型性能, 减少机器学习的应用壁垒, 避免人工调参带来的主观误差, 且结果容易复现.

英文引用格式: Zhang A J, Yang Z B. Hyperparameter tuning methods in automated machine learning (in Chinese). Sci Sin Math, 2020, 50: 695–710, doi: 10.1360/N012019-00092

图 1 (网络版彩图) 自动化机器学习框架

然而, 自动化超参优化同样面临着诸多难题: (i) 需要考虑不同类型的超参, 常见的有连续型 (如支持向量机的核函数参数与惩罚因子)、整数型 (如决策树的最大深度) 和定性变量 (如神经网络激活函数) 等; (ii) 作为一个组合优化问题, 当超参个数增大时, “维度灾难” 不可避免; (iii) 超参与模型表现之间的映射关系往往是一个复杂的 “黑盒” 响应曲面, 常见的优化算法很容易陷入局部最优. 因此, 如何科学有效地对机器学习模型进行超参优化是一个具有挑战性的问题.

本文将围绕自动化机器学习中的超参调优这一主题, 对相关文献和方法进行归纳总结. 内容安排如下: 第 2 节回顾传统的超参调优方法, 包括批量抽样、Bayes 优化、群体搜索算法和强化学习; 第 3 节将着重介绍均匀试验设计和序贯均匀设计, 并对后者加以改进, 使之适用于机器学习超参优化; 第 4 节以两种典型的机器学习算法和两组典型的二分类数据集为例, 通过数值实验对各种调参方法进行比较分析, 探讨其优势与劣势; 最后, 第 5 节总结全文, 并提出未来的一些研究方向.

2 传统方法综述

本节将介绍几种传统的机器学习超参调优方法, 包括批量抽样、Bayes 优化、群体搜索算法及最近出现的强化学习. 其中, 后 3 种方法利用序贯优化策略, 所选取的试验点会随着试验信息的更新而动态调整, 因而优化效率更高.

2.1 批量抽样

下面介绍的几种批量抽样方法, 共同点在于所有试验点都可以提前给定. 其优势在于试验点的选择简单、便捷, 但是如何预先确定试验点的数量并非易事: 太多的试验点可能会造成浪费, 而过少的点则不能很好地找到最佳超参组合. 为了对下列 4 种方法有一个直观的了解, 我们给出示例图 (见图 2).

(1) 格子点法

格子点法预先将每一维超参划分成多个格点, 然后对所有维度的格点组合进行穷举搜索. 这种方法由于其简单、易操作的特性, 在实践中被广泛使用. 假设超参维度为 s , 每个超参可以选取的格点数相同且均为 g , 穷举搜索的总次数即为 g^s . 因为格子点数量会随着超参维度的增加呈几何级数增加, 所以对于高维超参优化问题, 格子点法往往无能为力. 其次, 格子点法对每个超参维度仅可以探索 g 个格点位置, 而真正最优的区域很可能被忽略掉.

图 2 (网络版彩图) 4 种方法的比较: 在 2 维空间生成 25 个试验点. (a) 格子点法; (b) 随机搜索; (c) 超拉丁方抽样; (d) Sobol 序列

(2) 随机搜索

Bergstra 和 Bengio^[1]提出了一种同样简单的随机搜索调参方法, 即在搜索空间随机地添加试验点. 与格子点法相比, 随机搜索的主要优势在于其搜索空间更为广阔. 对于每一个超参维度, 随机搜索将不再局限于有限个格子点, 而是可以尝试更多不同的值. 当不同维度超参的重要性相差较大时, 这种方法的优势更为明显. 其次, 随机搜索非常灵活, 在优化过程中可以随时停止. 最后, 随机搜索的试验次数不会受维度限制, 对于高维问题仍可使用.

(3) 超拉丁方抽样

超拉丁方抽样 (Latin hypercube sampling, LHS)^[2]是一种近似随机的分层抽样方法. 它能够保证搜索空间的每一维度、每个小区间都有一个试验点. 其抽样过程可以通过如下步骤实现: (i) 对 s 维搜索空间中的每一维进行 n 等分, 每个小区间依顺序分别标记为 $\{1, 2, \dots, n\}$; (ii) 将序列 $\{1, 2, \dots, n\}$ 进行 s 次随机置换, 得到一个 $n \times s$ 的设计矩阵, 其中每一行代表一个超立方格; (iii) 在设计矩阵每个超立方格中随机选一个点, 得到最终的超拉丁方抽样. 超拉丁方抽样实现过程简单, 且对于高维超参优化问题同样适用. 一般而言, 超拉丁方抽样的试验点要优于随机采样, 但对整体空间的代表性不足, 往往还需要进一步利用其他空间填充准则进行优化.

(4) Sobol 序列

Sobol 序列是一种特殊的伪随机序列, 其目标是使搜索空间任何子区域内试验点的数量尽可能与子空间大小成正比. Sobol 序列的生成依赖于二进制编码, 并通过一定的规则迭代生成新的试验点, 最后再转换为十进制编码 (参见文献 [3]). 如果试验点数量足够大, Sobol 序列可以近似均匀地覆盖整个搜索空间. 文献 [1] 通过模拟实验比较了随机搜索、超拉丁方抽样和 Sobol 序列算法在全局优化问题中的表现, 结果显示 Sobol 序列表现最佳.

2.2 Bayes 优化

Bayes 优化是一种针对黑盒问题的序贯式全局优化策略, 在自动化超参调优这一领域, 占据着重要的地位. 这一类方法首先对已知试验点的潜在响应曲面建立代理模型, 然后通过最大化采集函数的方式添加新的试验点. 相较于格子点法和随机搜索等算法, Bayes 优化可以有效利用已知试验点的信息, 动态调整搜索位置, 减少试验的盲目性. Bayes 优化在确定新试验点的位置时, 涉及“探索”和“寻优”两个目标. “探索”的目的是更好地理解搜索空间, 而“寻优”则是为了找到全局最优. 为了便于读者理解 Bayes 优化算法在机器学习超参优化中的应用, 我们借鉴文献 [4] 给出这类方法的一般步骤, 见算法 1.

算法 1 Bayes 优化算法在机器学习超参优化中的应用

- 1: 生成 n_0 个初始试验点 $\{\mathbf{x}_1, \dots, \mathbf{x}_{n_0}\}$, 每个试验点代表一组超参.
 - 2: 对每一组超参, 训练机器学习模型, 计算其模型表现 $\{y_1, \dots, y_{n_0}\}$.
 - 3: 基于已知试验点和对应表现, 构建代理模型 $p(y | \mathbf{x})$, 并生成相应的采集函数 $\alpha(\mathbf{x})$.
 - 4: **for** $t = 1, 2, \dots$ **do**
 - 5: 添加新的试验点 $\mathbf{x}_{n_0+t} = \arg \max_{\mathbf{x}} \alpha(\mathbf{x})$, 并计算该超参对应的模型表现.
 - 6: 更新代理模型和采集函数 $\alpha(\mathbf{x})$.
 - 7: **end for**
 - 8: 输出最佳超参组合.
-

接下来介绍 3 种经典的 Bayes 优化算法, 包括: (i) Gauss 过程 - 期望提升算法 (Gaussian process-expected improvement, GP-EI), 通过 Gauss 过程 (Gaussian process, GP) 回归建立代理模型 $p(y | \mathbf{x})$ [5]; (ii) 基于贯序模型的算法框架 (sequential model-based algorithm configuration, SMAC), 使用随机森林作为代理模型 $p(y | \mathbf{x})$ [6]; (iii) 树状结构 Parzen 估计算法 (tree-structured Parzen estimators, TPE), 通过核密度方法估计 $p(\mathbf{x} | y)$ [7]. 这 3 种方法的共同点在于它们都使用期望提升 (expected improvement, EI) 作为采集函数. 假设优化目标为最大化模型精度, 则期望提升可以表达为

$$\alpha_{\text{EI}}(\mathbf{x}) = \int_{-\infty}^{\infty} \max(y - y^*, 0) p(y | \mathbf{x}) dy, \quad (2.1)$$

其中, y^* 为已知试验点中最优表现值. 除期望提升外, 其他常用的 Bayes 优化采集函数还包括概率提升 (以超过当前最优点的概率为决策标准) 和置信区间上界 (乐观准则, 以代理模型所预测置信区间上界为决策标准) 等, 这里不再赘述.

(1) GP-EI 方法

GP-EI 方法使用 Gauss 过程回归构建代理模型, 并以最大化期望提升为序贯选点准则 [5]. Gauss 过程回归是一种常用的非参数模型, 给定试验点与对应结果集合 $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, 其关系可以写为

$$y = f(\mathbf{x}) + \epsilon, \quad (2.2)$$

其中, 噪声项 $\epsilon \sim N(0, \sigma^2)$. 假设模型的先验分布为

$$\begin{aligned} \mathbf{y} | \mathbf{f}, \quad \sigma^2 &\sim \mathcal{N}(\mathbf{f}, \sigma^2 \mathbf{I}), \\ \mathbf{f} &\equiv [f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_n)]^T \sim \mathcal{N}(0, \mathbf{K}), \end{aligned} \quad (2.3)$$

其中, 符号 $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{n \times n}$ 表示协方差矩阵, k 为核函数. 给定任意新试验点 $\mathbf{x}$, 记 $\mathbf{x}$ 与 $\mathbf{x}_{1:n}$ 的协方差向量为 $\mathbf{k}(\mathbf{x})$, 则 Gauss 过程回归的预测分布 $p(y | \mathbf{x})$ 符合正态分布, 其均值与方差分别为

$$\begin{aligned} \mu(\mathbf{x}) &= \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}, \\ \sigma^2(\mathbf{x}) &= k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}(\mathbf{x}). \end{aligned} \quad (2.4)$$

那么, 相应的期望提升根据 (2.1) 可被写为

$$\alpha_{\text{EI}}(\mathbf{x}) = \sigma(\mathbf{x}) [z^* \Phi(z^*(\mathbf{x})) + \phi(z^*(\mathbf{x}))], \quad (2.5)$$

其中,

$$z^*(\mathbf{x}) = \frac{\mu(\mathbf{x}) - y^*}{\sigma(\mathbf{x})},$$

ϕ 与 Φ 分别表示标准正态分布的概率密度函数和累积分布函数. 图 3 演示了 GP-EI 方法在一维空间的决策过程. 可以看到, 当前已知 3 个试验点 (圆圈), 通过 Gauss 过程回归, 可以得到一维空间所有点的预测均值 (虚线) 和置信区间 (阴影部分). 在图 3 下半部分, 我们可以计算每个点的期望提升值, 并找到期望提升最大的 $\mathbf{x}$ 值作为下一个试验点. Gauss 过程回归所拟合的曲线相对比较平滑, 当真实响应曲面并不符合平滑假设时, GP-EI 方法存在进一步改进空间.

(2) SMAC 方法

由 Hutter 等^[6]提出的 SMAC 方法是另一种经典的 Bayes 优化方法. 与 GP-EI 方法类似, SMAC 方法将期望提升作为采集函数, 并使用随机森林作为代理模型来拟合潜在的响应曲面. 随机森林模型最早由 Breiman^[8]提出, 本质上是一组并行化的决策树集成模型, 具有较强的非线性拟合能力. 其中, 每棵子决策树只对从完整数据中随机抽取的一小部分样本和特征进行训练, 所有子决策树的平均即为

图 3 (网络版彩图) Bayes 优化示例

最终预测结果. 然而, 经典的随机森林算法并没有提供预测点的方差估计. 针对这一问题, 文献 [6] 提出了使用各子决策树预测值的方差作为替代. 由于随机森林能够很好地处理不同类型的数据, 且对高维数据建模有较好的拓展性, SMAC 方法在实际应用中有时会比 GP-EI 方法表现优异. 然而, 由于使用随机森林作为代理模型, 它所拟合的响应曲面不连续, 并不适合对平滑响应曲面进行优化 [4].

(3) TPE 方法

与上述两种方法不同的是, TPE [7] 并不直接对 $p(y | \mathbf{x})$ 建模, 而是转而估计 $p(\mathbf{x} | y)$ 和 $p(y)$. 首先定义响应曲面低于 y^* 和高于 y^* 的条件概率密度函数 $g(\mathbf{x})$ 和 $\ell(\mathbf{x})$:

$$p(\mathbf{x} | y) = \begin{cases} \ell(\mathbf{x}), & \text{若 } y < y^*, \\ g(\mathbf{x}), & \text{若 } y \geq y^*, \end{cases} \quad (2.6)$$

并通过非参数核密度方法进行估计. 为了使 $g(\mathbf{x})$ 不为 0, 取 y^* 为稍小于最优点的值, 一般设定为 y 的分位数 ($\gamma = 0.85$). 在此情形下, 采用如下形式的采集函数:

$$\alpha_{\text{EI}}(\mathbf{x}) \propto \left(\frac{\ell(\mathbf{x})}{g(\mathbf{x})} \gamma + 1 - \gamma \right)^{-1}. \quad (2.7)$$

该方法避免了直接计算期望提升, 在寻找新的试验点时效率更高. 由于其特殊的层次化结构, 该方法更适用于有依赖关系的超参优化问题. 例如, 优化神经网络的结构时, 往往需要先确定层数, 再确定节点个数.

这 3 种 Bayes 优化方法在 AutoML 实践中取得了很大的成功, 并且有专业的 Python 程序包方便用户使用, 如 **Spearmint**、**SMAC** 和 **Hyperopt** [9]. 此外, 它们也被集成在很多专业软件中, 例如, **HPOLib** 为上述 3 种 Bayes 优化算法提供了统一的调用接口; **Auto-sklearn** [10] 和 **AutoWeka** [11] 均使用 SMAC 作为优化器, 分别为机器学习库 **Scikit-Learn** 和 **Weka** 提供超参优化功能.

然而, 上述经典的 Bayes 优化方法仍存在不少缺陷, 我们从以下 3 个角度加以分析.

(1) 高维超参空间

Bayes 优化没有考虑试验点的代表性和均匀性, 使得大量空间未被探索, 尤其对高维超参优化这一问题影响更大. 例如, 文献 [12] 指出经典的 Bayes 优化模型在高维搜索空间的表现仅接近于随机搜索. 针对这一问题, 一种可行的方案是将高维问题转化成低维问题进行优化. 例如, 文献 [12] 提出了一种随机嵌入算法, 通过随机矩阵建立高维与低维之间的关联; 文献 [13] 利用了多个相互正交的加性模型对响应函数进行建模.

(2) 代理模型局限性

Bayes 优化的核心步骤是建立代理模型并通过优化采集函数来得到下一个试验点. 这一步骤的局限性在于, 首先, 机器学习超参所对应的真实响应曲面一般很粗糙, 而常用代理模型 (如 Gauss 过程回归) 会将粗糙的曲面进行平滑, 使其难以找到全局最优点; 其次, Gauss 过程本身对于高维回归建模存在着瓶颈; 最后, 采集函数优化也是一个高维优化问题, 我们往往需要在优化性能与资源消耗之间做出权衡 [13]. 在实践中, 我们用大规模随机采样来获取的“最优”采集函数值, 往往不是采集函数真正的最优值. 为了提升采集函数的优化效率, 一些学者提出使用诸如多初始值重启 [6]、基于梯度的数值优化 [14] 和协方差矩阵自适应进化策略 (covariance matrix adaptation evolution strategy, CMA-ES, 一种不需要梯度的群体搜索算法) [7] 等对其进行改进. 另一种方式是直接避免采集函数的优化, 例如, 文献 [15] 提出了一种基于树状分区搜索和 Gauss 过程回归的多尺度乐观优化算法 (Bayesian multi-scale optimistic optimization, BaMSOO), 从而省去采集函数优化这一过程.

(3) 并行优化

Bayes 优化的另一个问题在于计算难以并行化. 标准的 Bayes 优化策略每次添加一个试验点, 只有当前试验点所对应的训练和验证过程结束后, 才可以继续添加新的试验. 针对这一问题, 一些学者尝试对原有算法进行改进, 在并行化方面做出尝试. 文献 [16] 提出对待运行试验点的实验结果进行预测插值, 如常数插值和 Gauss 过程回归插值等. 基于对实验结果的预测, 我们可以一次性添加多个后续试验点. 例如, 文献 [5] 提出使用期望采集函数不断地生成新的试验点, 当新试验点的数量满足批量运行条件时, 就可以将其并行化执行. 类似的策略还包括 Gauss 过程回归 - 置信区间上界批量优化方法 (Gaussian process batch upper confidence bound, GP-BUCB)^[17], 这里不再赘述. 另一种并行化策略是一次性批量给出后续试验点. 例如, 文献 [18] 提出可以基于不同的置信区间上界作为采集函数, 从而实现一次性得到多个试验点. 提高 Bayes 优化效率的另一个改进方向是多任务优化, 通过共享相似数据集的历史优化信息, 进而提升效果, 参见文献 [19–21].

2.3 群体搜索算法

群体搜索算法是一种针对全局优化的启发式搜索方法. 它把每个试验点当作种群中的一个个体, 每个个体有着不同的适应度. 种群内个体间可以通过相互交换信息, 不断地更新以找到最大化适应度的位置. 常用的有遗传算法、粒子群优化算法及其他模拟动物族群行为的算法, 如萤火虫算法和蚁群算法等.

(1) 遗传算法

遗传算法 (genetic algorithm, GA) 是通过模拟“优胜劣汰, 适者生存”这一生物进化理论所创建的一种启发式搜索算法. 在遗传算法中, 每个个体代表问题的一个解, 可以通过一定的规则被编码为一组数字, 通常称为“染色体”. 例如, 在机器学习超参优化中, 可以使用二进制码对超参进行编码, 并将模型交叉验证的表现作为适应函数. 一个完整的遗传算法迭代寻优过程如下:

(i) 初始化: 随机生成一定数量的个体.

(ii) 个体评价: 计算种群中每个个体的适应度.

(iii) 选择: 选择一组父代种群, 原则是适应度越高的个体被选中的概率越大. 常用的选择方法有轮盘赌法和竞争法.

(iv) 交叉: 对父代种群进行两两配对, 每组配对依给定交叉概率进行交配. 如果一对父代个体进行了交配, 则用其子代代替父代; 否则, 父代保持不变.

(v) 突变: 每个个体会根据概率进行突变. 通过突变, 可以有效防止种群陷入局部最优.

重复步骤 (ii)–(v), 可以不断对种群进行更新. 当种群迭代次数达到预先设定的阈值时, 就可以终止算法. 遗传算法作为一种成熟的全局优化算法, 已经被广泛应用于机器学习超参优化这一领域. 例如, 文献 [22] 使用遗传算法对支持向量机的超参进行优化, 文献 [23] 利用改进的遗传算法搜索神经网络的最佳结构.

(2) 粒子群算法

粒子群算法 (particle swarm optimization, PSO) 是另一种经典的群体搜索算法. 与遗传算法不同, 它假设在搜索空间中有一群没有质量和体积的微小粒子. 每个粒子所在位置都有一个适应度, 且都在以一定的速度移动. 为了找到最佳适应度的位置, 这些粒子会根据自身及其他粒子的历史信息动态调整其速度和位置, 具体步骤如下:

(i) 初始化: 随机初始化每个个体的位置 $\mathbf{o}_i$ 和速度 $\mathbf{v}_i$ ($i = 1, 2, \dots, m$), m 为个体数量;

- (ii) 个体评价: 计算种群中个体的适应度, 并统计每个个体的历史最佳 I_i 和种群的历史最佳 g ;
- (iii) 速度与位置更新:

$$\mathbf{v}_i = \omega \mathbf{v}_i + c_1 r_1 (\mathbf{I}_i - \mathbf{o}_i) + c_2 r_2 (\mathbf{g} - \mathbf{o}_i), \quad \mathbf{o}_i = \mathbf{o}_i + \mathbf{v}_i, \quad (2.8)$$

其中, ω 为惯性权重, r_1 和 r_2 是两个在 $[0, 1]$ 区间服从均匀分布的随机数, c_1 和 c_2 为加速常数.

不断重复步骤 (ii) 和 (iii), 直至达到给定的最大迭代次数. 粒子群算法同样被大量应用于自动化机器学习这一领域. 例如, 文献 [24] 将遗传算法与粒子群算法相结合, 研究循环神经网络的结构优化问题; 文献 [25] 将粒子群算法用于选择最小二乘支持向量机的超参; 文献 [26] 提出利用粒子群算法来优化卷积神经网络的结构, 取得了较好的表现.

2.4 强化学习

强化学习是一种序贯决策方法, 其决策过程不仅考虑当前收益, 还考虑未来收益期望的最大化. 为了实现这一目标, 个体会不断地观察当前的环境状态, 从而执行动作. 与传统的监督学习相比, 强化学习没有固定的学习目标, 所有的数据都是在决策过程中不断生成的, 因而其探索空间更为广阔. 强化学习的应用场景很多, 如游戏智能体和机器人自动控制等.

文献 [27] 首次提出将强化学习应用于深度神经网络的超参优化. 该研究提出使用循环神经网络作为智能体来生成超参组合, 因此能够更好地对网络结构之间的依赖关系进行建模. 然后, 不同超参的表现会作为奖励或惩罚, 以训练智能体. 这一方法已经被成功应用在卷积神经网络图像识别问题中, 取得了很高的准确率. 该方法同样可以用于优化循环神经网络的单元结构, 在实验中, 该方法甚至找到了一种与经典的长短期记忆单元相近但效果更好的结构单元. 类似地, 文献 [28] 基于强化学习提出了一种 meta Q 学习神经网络 (meta Q-learning neural network, metaQNN) 模型, 用于卷积神经网络的网络结构优化. 然而, 强化学习也有其局限, 主要体现在训练强化学习模型需要大量的计算资源.

3 均匀试验设计

20 世纪 70 年代末, 方开泰和王元为解决导弹设计问题, 提出利用数论方法构造均匀试验设计. 多年来, 均匀试验设计方面的研究取得了丰硕成果 [29–33].

3.1 均匀设计

均匀设计的目标是找到能够均匀覆盖搜索空间 $C^s = [0, 1]^s$ 的一组试验点集合 D_n , 一般通过优化如下目标实现:

$$\phi(D_n^*) = \min_{D_n \subset C^s} \phi(D_n), \quad (3.1)$$

这里的 ϕ 代表一种均匀性度量函数, 如星偏差

$$D_*(D_n) = \sup_{P \in J^*} \left| \frac{\mathcal{A}(P, \mathbf{x})}{n} - \lambda(P) \right|, \quad (3.2)$$

其中, J^* 为搜索空间内所有形如 $\prod_{i=1}^s [0, u_i)$ ($u_i \in [0, 1)$) 的超矩形集合, $\mathcal{A}(P, \mathbf{x})$ 为落在超矩形 P 内试验点的数量, $\lambda(P)$ 为子区间 P 的体积大小. 然而, 星偏差存在一定的局限性, 例如它没有考虑低维投影空间的均匀性; 所有超矩形都从原点开始, 不具有旋转不变性. 针对这些问题, 文献 [34] 提出了

广义 L_2 - 偏差, 包括中心化 L_2 - 偏差 (centered L_2 -discrepancy, CD_2) 和可卷 L_2 - 偏差 (wrap-around L_2 -discrepancy, WD_2) 等. 其中, 基于 CD_2 准则生成的试验设计往往有着较好的均匀性, 因此本文使用 CD_2 准则构造均匀设计, 其公式如下:

$$CD_2(D_n)^2 = \left(\frac{13}{12}\right)^s - \frac{2}{n} \sum_{k=1}^n \prod_{j=1}^s \left(1 + \frac{1}{2} \left|x_{kj} - \frac{1}{2}\right| - \frac{1}{2} \left|x_{kj} - \frac{1}{2}\right|^2\right) + \frac{1}{n^2} \sum_{k=1}^n \sum_{j=1}^n \prod_{i=1}^s \left[1 + \frac{1}{2} \left|x_{ki} - \frac{1}{2}\right| + \frac{1}{2} \left|x_{ji} - \frac{1}{2}\right| - \frac{1}{2} |x_{ki} - x_{ji}|\right]. \quad (3.3)$$

相较于 Sobol 序列和超拉丁方设计, 通过均匀设计所得到的试验点往往能更好地覆盖整个搜索空间. 然而, 如何找到一组试验点以优化目标 (3.1) 本身是一个难题. 常用的策略是将搜索空间限制为 U- 型设计, 然后进行优化. 假设一个试验设计包含 n 个试验点和 s 个因子, 每个因子包含 q 个水平, U- 型设计要求每个因子的 n 个数值平衡分布于下列水平:

$$\left\{ \frac{2i-1}{2q} \mid i = 1, 2, \dots, q \right\}, \quad (3.4)$$

即每个水平出现 n/q 次 (U- 型设计需要满足试验点数量 n 为水平数 q 的正整数倍). 针对 (3.1), 通常采用列元素交换策略进行优化: 给定一个随机生成的 U- 型设计, 将均匀设计表同一列 (因子) 不同行 (试验点) 的元素进行位置交换. 不断重复这一过程, 通常可以得到均匀性更好的设计, 常见的随机优化方法有模拟退火算法、门限接受法和进化算法等 [32, 35].

3.2 序贯均匀设计

序贯试验设计使用序贯策略对传统的批量试验设计做出改进, 虽然其整体空间填充性能可能会下降, 但这一方法却更为灵活: 试验点数量可以在优化过程中动态决定, 避免出现过度选点或过少选点的问题. 近期研究包括: 文献 [36] 提出了一种混合局部线性近似 (local linear approximation, LOLA)-Voronoi 方法, “探索” 与 “寻优” 这两个目标可以分别通过局部非线性拟合和 Voronoi 近似实现; 文献 [37] 对 LOLA-Voronoi 方法进行了拓展, 提出了一种基于模糊规则的序贯设计方法, 对于解决高维优化问题有一定优势.

为了加快寻找全局最优的速度, 方开泰和王元早在 1990 年提出了一种序贯均匀设计方法 (sequential number-theoretic optimization, SNT0) [29]. 与标准的均匀设计不同, SNT0 初始化时仅在搜索空间中添加少量满足均匀设计的试验点. 假设每一维的初始搜索边界为 $a_i^{(0)} = 0$, $b_i^{(0)} = 1$, $i = 1, 2, \dots, s$. 当第一批均匀设计试验点得到验证后, 将搜索空间缩放至已知最优试验点 $\mathbf{x}^{(0)}$ 附近, 并添加一批新的均匀设计试验点. 该过程可以不断循环, 第 t 步缩放空间的边界可以通过以下公式得到:

$$a_i^{(t)} = \max \left\{ \mathbf{x}_i^{(t-1)} - \frac{\gamma(b_i^{(t-1)} - a_i^{(t-1)})}{2}, a_i^{(0)} \right\}, \quad b_i^{(t)} = \min \left\{ \mathbf{x}_i^{(t-1)} + \frac{\gamma(b_i^{(t-1)} - a_i^{(t-1)})}{2}, b_i^{(0)} \right\}, \quad (3.5)$$

其中, 参数 γ 控制缩放空间的比率, 一般设为 0.5. 当缩放后子空间各维度的宽度小于阈值 δ 后, 则终止搜索, 详细描述见算法 2.

在 SNT0 的基础上, 文献 [38] 提出了一种简化的两阶段嵌套均匀设计, 用于支持向量机模型中的 Gauss 核宽参数和惩罚因子的优化. 该方法首先在初始搜索空间使用包含 13 个试验点的均匀设计. 然后, 以前一阶段最优点为中心, 将每一维度的范围减半, 并在缩放后的子空间添加一组包含 9 个试

算法 2 SNT0 算法

```

1: 在搜索区域  $C^s$  生成包含  $n_0$  个点的均匀设计  $D = D_0$ , 并验证其对应超参表现.
2: for  $t = 1, 2, \dots$  do
3:   从已知试验点中找到表现最好的一个  $\mathbf{x}^{(t-1)}$ .
4:   以  $\mathbf{x}^{(t-1)}$  为中心, 根据 (3.5) 计算缩放后空间的边界值  $\{a_i^{(t)}, b_i^{(t)}\}$ .
5:   if  $(b_i^{(t)} - a_i^{(t)}) < \delta$  then
6:     终止算法.
7:   end if
8:   在缩放后的子空间添加一组包含  $n_t$  个试验点的均匀设计  $D_t$ .
9:   验证新的试验点, 并更新  $D = D \cup D_t$ .
10: end for
11: 输出最优试验点  $x^* \in D$ .
```

验点的均匀设计. 由于该试验的特殊性, 第二阶段的中心点恰好与新添加的 9 个试验点其中之一重合, 因此, 总的试验点数为 21 个. 针对于大规模数据集, 该文献还提出可以在两个阶段分别使用 9 和 5 个试验点来简化计算 (共计 13 个, 其中一个试验点重复). 实验结果表明, 该方法可以通过较少的试验次数达到与格子点法相似甚至更好的准确度. 然而, 该方法所使用的两阶段嵌套均匀设计过于特殊, 难以用于优化其他更加复杂的机器学习模型.

3.3 改进的序贯均匀设计

直接将 SNT0 算法应用于机器学习超参优化可能会存在一些问题. 第一, 如果前一阶段的最优值靠近搜索区域的边界, 则新的搜索空间将被过度压缩. 第二, 为了方便, 一般在 SNT0 的不同阶段重复使用同一个均匀设计, 而实践中使用的均匀设计试验点往往有着独特的分布特征, 嵌套使用同一个设计可能会导致一定的系统性偏差. 针对这两个问题, 我们提出一种改进的 SNT0 方法, 简称 SNT0-New, 包含如下 3 点改进 (详见算法 3).

算法 3 改进的 SNT0 算法

```

1: 在搜索区域  $C^s$  生成包含  $n_0$  个点的均匀设计  $D = D_0$ , 并验证其对应超参表现.
2: for  $t = 1, 2, \dots$  do
3:   从已知试验点中找到表现最好的一个  $\mathbf{x}^{(t-1)}$ .
4:   以  $\mathbf{x}^{(t-1)}$  为中心对每一维进行缩放. 如果部分缩放空间超出边界, 则将其向中心平移直到全部位于边界内.
5:   if  $(b_i^{(t)} - a_i^{(t)}) < \delta$  then
6:     终止算法.
7:   end if
8:   在缩放后的子空间确定一组包含  $n_t$  个试验点的均匀设计  $D_t$ .
9:   对  $D_t$  中每个试验点的所有因子各自添加一个随机扰动, 并对不同的因子进行随机置换.
10:  验证新的试验点, 并更新  $D = D \cup D_t$ .
11: end for
12: 输出最优试验点  $x^* \in D$ .
```

(1) 搜索空间缩放时, 以 $\mathbf{x}^{(t-1)}$ 为中心, 将空间按比例缩放:

$$a_i^{(t)} = \mathbf{x}_i^{(t-1)} - \frac{\gamma(b_i^{(t-1)} - a_i^{(t-1)})}{2}, \quad b_i^{(t)} = \mathbf{x}_i^{(t-1)} + \frac{\gamma(b_i^{(t-1)} - a_i^{(t-1)})}{2}. \quad (3.6)$$

然后, 判断新的边界是否超出原始 $[0, 1]^s$ 区间. 如果超出, 则将缩放空间后的子空间向内平移, 直到子空间完全处于边界之内.

图 4 (网络版彩图) SNTO 与 SNTO-New 示意图. (a) 均匀设计 $U_{20}(20^2)$; (b) SNTO; (c) SNTO-New

(2) 在每次生成新的子空间均匀设计时, 对每个试验点的所有因子各自添加一个随机扰动:

$$\mu \sim \text{Unif}\left[-\frac{\gamma^t}{2q}, \frac{\gamma^t}{2q}\right]. \quad (3.7)$$

(3) 对不同的因子进行随机置换.

图 4 给出了均匀设计、SNTO 和 SNTO-New 的示意图. 可以看到, 均匀设计的试验点较为均匀地覆盖了搜索空间. 图中所示的两种序贯均匀设计所使用的初始试验点相同, 均为 $U_{20}(20^2)$. 其中, 第一阶段的试验点用圆点表示, 而最优试验点用符号 $\oplus$ 表示; 第二阶段的试验点用三角形点表示, 并用虚线框表示缩小后的搜索空间. 当某一步的最优试验点靠近边界时, SNTO 会将超出边界部分进行截断, 而 SNTO-New 则将搜索区域进行平移, 从而防止搜索空间被过度压缩. 其次, 由于我们对搜索子空间内的试验点进行了随机扰动和置换操作, 可以在一定程度上避免系统性偏差.

4 数值实验

本节数值实验模拟自动化机器学习中的超参优化环节, 目的是比较多种调优方法在不同数据集和不同机器学习算法上的优劣表现. 本文介绍的如下几种方法将纳入比较实验: 格子点法 (Grid)、随机搜索 (Rand)、超拉丁方设计 (LHS)、Sobol 序列 (Sobol)、均匀设计 (uniform design, UD)、GP-EI、SMAC、TPE 和 SNTO-New. 群体搜索和强化学习等方法由于所需要的计算量大, 本文暂不考虑.

4.1 实验设定

测试各方法在最大尝试次数分别为 20、40、60、80 和 100 时所能达到的最佳表现 (格子点法最大尝试次数分别设为 25、49、64、81 和 100). 对于批量方法, 所有试验点均依最大尝试次数一次性生成; 而对于序贯方法, 试验点则是分批或逐个生成, 直到达到最大尝试次数. 其中, SNTO-New 每个批次均重复使用同一个均匀设计 $U_{20}(20^s)$. 我们以 5-折交叉验证准确率作为优化目标, 并选取以下两种经典的机器学习算法作为基础模型.

(1) 支持向量机 (support vector machine, SVM). 使用 Gauss 核函数, 并优化 SVM 最为常见的两个超参: 惩罚因子和核函数参数 (分别记为 LogCost 和 LogGamma).

(2) 极限梯度推进机 (extreme gradient boosting, XGBoost). 挑选出 8 个较为重要的超参进行优化, 包括 1 个定性超参 (基础模型的选择)、2 个整数型超参 (基础模型的数量和最大树深度) 和 5 个连续型超参 (特征比率、学习率、最小误差降低阈值、 ℓ_1 及 ℓ_2 正则化因子).

考虑如下两个典型的二分类数据集对各种方法进行测试. (i) Statlog Heart 数据集: 记录 270 个患者的 13 个与身体状况相关的自变量, 以预测患者是否患有心脏病. 该数据集在著名的 LibSVM 软件包中曾被用作示例数据 [39]. (ii) Breast Cancer 数据集: 记录 569 个乳腺癌患者的 30 个细胞核图像特征, 用于预测癌症的类别, 包括良性和恶性两种.

对每种方法重复 10 次实验, 并将其最终优化的准确度进行平均. 表 1 记录了最大尝试次数为 100 时, 各方法所达到的最佳模型精度; 图 5 和 6 展示了各超参优化方法在不同最大尝试次数下所

表 1 不同超参优化方法在各机器学习模型和数据集上的表现 (最大尝试次数 100)

模型	数据集	Grid	Rand	LHS	Sobol	UD	SMAC	GP-EI	TPE	SNT0-New
SVM	Heart	0.8560	0.8589	0.8582	0.8563	0.8586	0.8600	0.8630	0.8600	0.8619
	Breast	0.9793	0.9789	0.9782	0.9789	0.9789	0.9789	0.9798	0.9802	0.9808
XGBoost	Heart	-	0.8571	0.8571	0.8556	0.8556	0.8600	0.8545	0.8571	0.8600
	Breast	-	0.9796	0.9800	0.9800	0.9791	0.9807	0.9817	0.9816	0.9817

图 5 (网络版彩图) 不同最大尝试次数下, 各方法在 SVM 超参优化问题上的表现. (a) Heart 数据集; (b) Breast 数据集

图 6 (网络版彩图) 不同最大尝试次数下, 各方法在 XGBoost 超参优化问题上的表现. (a) Heart 数据集; (b) Breast 数据集

图 7 (网络版彩图) 不同超参优化方法的试验点分布情形 (以 SVM-Heart 为例). (a) 格子点法; (b) 随机搜索; (c) 超拉丁方抽样; (d) Sobol 序列; (e) 均匀设计; (f) SMAC; (g) GP-EI; (h) TPE; (i) SNT0-New

达到的最佳模型精度. 另外, 我们以 SVM-Heart 超参优化问题为例, 将各种方法其中一次实验的采点进行数据可视化, 见图 7.

4.2 结果分析

从实验结果可以观察到, 序贯试验设计整体表现要优于单次批量抽样方法. SNT0-New 在 3 次实验中取得最佳结果, GP-EI 在两次实验中取得最佳效果, SMAC 在一次实验中取得最佳. 新方法 SNT0-New 的另一优势在于其生成试验点的速度快: 大部分情形下, 初始的均匀设计试验点不仅可以通过查表得到, 且在之后的每一阶段都可以被重复嵌套使用. 而 GP-EI 等 Bayes 优化方法生成新试验点的速度则慢很多, 有时甚至要大于训练模型所需的时间. 除了优化效率较差外, 单次批量抽样的另一常见问题在于使用更多搜索次数不一定能提升效果. 这是由于不同最大搜索次数所对应的试验点之间不一定重叠, 从而丢失了试验次数与优化结果之间单调递增的函数关系. 同时还可以看到, 使用最为广泛的

格子点法有着更多的局限: 它对于高维超参优化问题往往无能为力, 因此, 我们只能在 SVM 这种二维超参优化问题中使用格子点法; 另外, 受限于格子点的数量, 真正的最优点很可能会被略过。

在图 7 中, 我们通过运行一组高密度格点, 得到不同超参组合的真实准确度, 并画出等高线。每种算法的试验点用圆点表示, 而各算法所找到的最优位置通过方形点来标注。可以看到, 该问题的响应曲面非常复杂, 有很多的局部最优值。与上文分析类似, 在几种批量抽样或设计方法中, 均匀设计所产生的试验点分布最有代表性, 覆盖了整个空间; 其次是 Sobol 序列和超拉丁方抽样, 虽然其均匀性比随机搜索要好, 但仍然有不少子空间没有被考虑到。这些批量搜索方法的主要问题是大量的试验点被浪费在没有意义的区域。相应地, 序贯试验设计能够减少这些不必要的浪费, 将有限的试验点放在更加重要的区域, 从而取得更好的优化效果。

5 总结与展望

机器学习超参优化是一个经典且具有挑战性的问题, 响应曲面未知且复杂, 全局最优往往很难确定。本文从试验设计的角度对常用的机器学习超参优化方法进行了总结, 将其分为批量试验设计和序贯试验设计; 然后, 在现有方法的基础上提出一种改进的序贯均匀设计方法。改进的序贯均匀设计方法的优势包括: 考虑到试验点的均匀性, 减少落入局部最优的机率; 避免使用代理模型, 减少选点这一费时的过程; 可以轻易并行化。在数值实验中, 通过使用不同的数据集和机器学习算法, 我们发现序贯均匀设计在解决自动化机器学习调参问题时颇具潜力。

目前, 基于序贯均匀设计的超参优化仍有一定的局限性: 在增加新的试验设计时, 没有考虑已知试验点的影响, 尤其是它们与追加试验点的冲突, 很可能导致一些低效试验点的出现。未来我们将考虑新添加试验设计点与已知试验点的均匀性关系, 并提出一套快速生成序贯均匀设计的算法, 以提升机器学习超参优化的效率。同时, 深度神经网络是近年来机器学习领域最为流行的一种算法, 然而由于其待优化的超参众多, 在实践中需要花费大量时间对其进行配置, 因此, 神经网络结构优化也将是我们未来研究的一个重要课题。

参考文献

- 1 Bergstra J, Bengio Y. Random search for hyper-parameter optimization. *J Mach Learn Res*, 2012, 13: 281–305
- 2 Mckay M D, Beckman R J, Conover W J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 2000, 42: 55–61
- 3 Bratley P, Fox B L. Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Trans Math Software*, 1988, 14: 88–100
- 4 Shahriari B, Swersky K, Wang Z, et al. Taking the human out of the loop: A review of Bayesian optimization. *Proc IEEE*, 2016, 104: 148–175
- 5 Snoek J, Larochelle H, Adams R P. Practical Bayesian optimization of machine learning algorithms. In: *Advances in Neural Information Processing Systems*. Red Hook: Curran Associates, 2012, 2951–2959
- 6 Hutter F, Hoos H H, Leyton-Brown K. Sequential model-based optimization for general algorithm configuration. In: *International Conference on Learning and Intelligent Optimization*. Berlin-Heidelberg: Springer, 2011, 507–523
- 7 Bergstra J S, Bardenet R, Bengio Y, et al. Algorithms for hyper-parameter optimization. In: *Advances in Neural Information Processing Systems*. Red Hook: Curran Associates, 2011, 2546–2554
- 8 Breiman L. Random forests. *Mach Learn*, 2001, 45: 5–32
- 9 Bergstra J, Komer B, Eliasmith C, et al. Hyperopt: A python library for model selection and hyperparameter optimization. *Comput Sci Disc*, 2015, 8: 014008

- 10 Feurer M, Klein A, Eggenberger K, et al. Efficient and robust automated machine learning. In: *Advances in Neural Information Processing Systems*. Cambridge: MIT Press, 2015, 2962–2970
- 11 Kotthoff L, Thornton C, Hoos H H, et al. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *J Mach Learn Res*, 2017, 18: 826–830
- 12 Wang Z, Zoghi M, Hutter F, et al. Bayesian optimization in high dimensions via random embeddings. In: *Twenty-Third International Joint Conference on Artificial Intelligence*. Menlo Park: AAAI Press/International Joint Conferences on Artificial Intelligence, 2013, 1778–1784
- 13 Kandasamy K, Schneider J, Póczos B. High dimensional Bayesian optimisation and bandits via additive models. In: *International Conference on Machine Learning*. Reykjavik: JMLR, 2015, 295–304
- 14 Lizotte D J, Greiner R, Schuurmans D. An experimental methodology for response surface optimization methods. *J Global Optim*, 2012, 53: 699–736
- 15 Wang Z, Shakibi B, Jin L, et al. Bayesian multi-scale optimistic optimization. In: *Artificial Intelligence and Statistics*. Reykjavik: JMLR, 2014, 1005–1014
- 16 Ginsbourger D, Le Riche R, Carraro L. Kriging is well-suited to parallelize optimization. In: *Computational Intelligence in Expensive Optimization Problems*. Berlin-Heidelberg: Springer, 2010, 131–162
- 17 Desautels T, Krause A, Burdick J W. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *J Mach Learn Res*, 2014, 15: 3873–923
- 18 Hutter F, Hoos H H, Leyton-Brown K. Parallel algorithm configuration. In: *International Conference on Learning and Intelligent Optimization*. Berlin-Heidelberg: Springer, 2012, 55–70
- 19 Swersky K, Snoek J, Adams R P. Multi-task Bayesian optimization. In: *Advances in Neural Information Processing Systems*. Red Hook: Curran Associates, 2013, 2004–2012
- 20 Bardenet R, Brendel M, Kégl B, et al. Collaborative hyperparameter tuning. In: *International Conference on Machine Learning*. Reykjavik: JMLR, 2013, 199–207
- 21 Klein A, Falkner S, Bartels S, et al. Fast Bayesian hyperparameter optimization on large datasets. *Electron J Stat*, 2017, 11: 4945–4968
- 22 Wu C H, Tzeng G H, Goo Y J, et al. A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy. *Expert Syst Appl*, 2007, 32: 397–408
- 23 Tsai J T, Chou J H, Liu T K. Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm. *IEEE Trans Neural Netw*, 2006, 17: 69–80
- 24 Juang C F. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Trans Syst Man Cybern B*, 2004, 34: 997–1006
- 25 Guo X C, Yang J H, Wu C G, et al. A novel LS-SVMs hyper-parameter selection based on particle swarm optimization. *Neurocomputing*, 2008, 71: 3211–3215
- 26 Lorenzo P R, Nalepa J, Kawulok M, et al. Particle swarm optimization for hyper-parameter selection in deep neural networks. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. New York: ACM, 2017, 481–488
- 27 Zoph B, Le Q V. Neural architecture search with reinforcement learning. ArXiv:1611.01578, 2016
- 28 Baker B, Gupta O, Naik N, et al. Designing neural network architectures using reinforcement learning. ArXiv:1611.02167, 2016
- 29 Fang K T, Wang Y. A sequential algorithm for optimization and its applications to regression analysis. In: *Lecture Notes in Contemporary Mathematics*. Beijing: Science Press, 1990, 17–28
- 30 Fang K T, Wang Y. *Number-Theoretic Methods in Statistics*. London: Chapman and Hall, 1994
- 31 Fang K T, Lin D K J, Winker P, et al. Uniform design: Theory and application. *Technometrics*, 2000, 42: 237–248
- 32 Fang K T, Li R, Sudjianto A. *Design and Modeling for Computer Experiments*. London: Chapman and Hall, 2006
- 33 Fang K T, Liu M Q, Qin H, et al. *Theory and Application of Uniform Experimental Designs*. Singapore: Springer, 2018
- 34 Hickernell F J. A generalized discrepancy and quadrature error bound. *Math Comp*, 1998, 67: 299–322
- 35 Jin R, Chen W, Sudjianto A. An efficient algorithm for constructing optimal design of computer experiments. *J Statist Plann Inference*, 2005, 134: 268–287
- 36 Crombecq K, Gorissen D, Deschrijver D, et al. A novel hybrid sequential design strategy for global surrogate modeling of computer experiments. *SIAM J Sci Comput*, 2011, 33: 1948–1974
- 37 van der Herten J, Couckuyt I, Deschrijver D, et al. A fuzzy hybrid sequential design strategy for global surrogate modeling of high-dimensional computer experiments. *SIAM J Sci Comput*, 2015, 37: 1020–1039

- 38 Huang C M, Lee Y J, Lin D K J, et al. Model selection for support vector machines via uniform design. *Comput Statist Data Anal*, 2007, 52: 335–346
- 39 Chang C C, Lin C J. LIBSVM: A library for support vector machines. *ACM Trans Intell Syst Technol*, 2011, 2: 1–27

Hyperparameter tuning methods in automated machine learning

Aijun Zhang & Zebin Yang

Abstract In this paper, we make a comprehensive review of the challenging task of hyperparameter optimization in automated machine learning. The commonly used hyperparameter optimization methods include single-shot sampling strategies, e.g., grid search, random search and sequential strategies where new trials are gradually augmented based on existing information, including Bayesian optimization, evolutionary algorithms, reinforcement learning-based methods. We find the sequential number-theoretic optimization (SNT0) algorithm proposed by Kai-Tai Fang and Yuan Wang in 1990 can also be applied in hyperparameter optimization, where sequential uniform designs are utilized to search the global optima in complex response surfaces. For illustration, various hyperparameter optimization methods are tested with two widely-used machine learning models including the support vector machine (SVM) and extreme gradient boosting (XGBoost), on two classical binary classification datasets. By analyzing the experimental results, we find a modified SNT0 algorithm is quite promising in the hyperparameter optimization task.

Keywords AutoML, hyperparameter optimization, Bayesian optimization, sequential uniform designs

MSC(2010) 62K20, 68T05, 90C26

doi: 10.1360/N012019-00092