渐进式深度集成架构搜索算法研究

朱光辉[1),2)] 祁加豪[1),2)] 朱振南[1),2)] 袁春风[1),2)] 黄宜华[1),2)]

[1)](南京大学计算机软件新技术全国重点实验室 南京 210023)

[2)](南京大学计算机科学与技术系 南京 210023)

摘 要 深度神经网络已在各个领域取得巨大成功.然而,深度学习模型并不只包含深度神经网络.近年来,以深度森林为代表的深度集成学习模型,凭借着无需反向传播训练、计算开销更小、模型复杂度支持自适应确定以及表数据建模任务性能优异等特点,引起了学界和业界的广泛关注,并且取得了良好的应用效果.深度森林为探索DNN(Deep Neural Network)之外的深度学习模型打开了另一扇门.然而,现有的深度集成模型主要以深度森林为主,深度集成架构较为单一,基学习器的数量与集成方式较为固定,需要探索除深度森林之外的深度集成学习模型架构.另外,实际应用中,很难存在一种深度集成学习模型架构能够在不同数据集上均取得优异性能,尤其是对于数据特征差异较大的表格型数据集.因此,也需要一种高效的数据自适应的深度集成学习架构设计方法.为此,本文从搜索空间和搜索算法两个层面,研究提出了一种高效的基于代理模型的渐进式深度集成架构搜索方法PMPAS(Proxy Model-based Progressive Architecture Search).首先,通过归纳分析已有深度集成学习模型的特点,给出了深度集成架构的形式化定义.其次,研究提出了两种全新的深度集成架构搜索空间,即基于完全并行的搜索空间和基于有向无环图的搜索空间.然后,在上述两种搜索空间的基础上,研究提出了基于代理模型的渐进式搜索方法与算法,实现从简单到复杂逐步地在搜索空间中进行探索,并采用代理模型作为指导,降低模型评估开销.最后,本文从时间复杂度和空间复杂度两个方面对搜索算法进行分析.在分类、回归等公开的表格型数据集上的大量实验结果表明,通过PMPAS算法搜索得到的深度集成架构,其性能不仅优于已有的集成学习模型、深度学习模型以及以深度森林为代表的深度集成学习模型,而且优于已有的自动化模型选择算法.随着时间预算的不断增加,性能优势更为明显.PMPAS开源地址为:https://github.com/PasaLab/PMPAS.

关键词 深度学习;深度集成架构;架构搜索;代理模型

中图法分类号 TP391 **DOI号** 10.11897/SP.J.1016.2023.02041

Research on Progressive Deep Ensemble Architecture Search Algorithm

ZHU Guang-Hui[1),2)] QI Jia-Hao[1),2)] ZHU Zhen-Nan[1),2)] YUAN Chun-Feng[1),2)]
HUANG Yi-Hua[1),2)]

[1)](*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210023)

[2)](*Department of Computer Science and Technology, Nanjing University, Nanjing* 210023)

Abstract Deep neural networks have achieved great success in various fields. However, deep learning models do not only contain deep neural networks. In recent years, due to the features such as no need for back-propagation training, lower computational overhead, support for adaptive determination of model complexity, and excellent performance in tabular data modeling tasks, the deep ensemble learning models represented by deep forest have attracted a lot of attention in the academia and industry, and achieved good application results. Deep forests open another door for

收稿日期:2022-09-15;在线发布日期:2023-04-18. 本课题得到国家自然科学基金项目(62102177)、江苏省自然科学基金项目(BK20210181)、江苏省重点研发计划项目(BE2021729)、江苏省软件新技术与产业化协同创新中心资助.**朱光辉**,博士,助理研究员,中国计算机学会(CCF)会员,主要研究领域为大数据智能分析与自动机器学习.E-mail:zgh@nju.edu.cn. **祁加豪**,硕士,主要研究领域为集成学习与自动化机器学习. **朱振南**,硕士,主要研究领域为数据挖掘与图神经网络. **袁春风**,教授,中国计算机学会(CCF)会员,计算机体系结构与并行计算. **黄宜华**(通信作者),博士,教授,中国计算机学会(CCF)会员,主要研究领域为大数据、云计算与并行计算.E-mail:yhuang@nju.edu.cn.

exploring deep learning models beyond DNN (Deep Neural Network). However, the existing deep ensemble models are mainly based on deep forests. The deep ensemble architecture is relatively simple, and the number and integration methods of basic learners are relatively fixed. It is necessary to explore deep ensemble learning model architectures other than deep forests. In addition, in practical applications, it is difficult to have a deep ensemble learning model architecture that can achieve excellent performance on different datasets, especially for tabular datasets with large differences in data characteristics. Therefore, an efficient data-adaptive deep ensemble learning architecture design method is also needed. To this end, this paper proposes an efficient proxy model-based progressive deep ensemble architecture search method PMPAS (Proxy Model-based Progressive Architecture Search) from two levels of search space and search algorithm. First, the characteristics of existing deep ensemble learning models are analyzed by induction, and the deep ensemble architecture is formally defined. Second, the research proposes two new search spaces for deep ensemble architectures, namely a fully parallel search space and a directed acyclic graph-based search space. For each layer of the fully parallel search space, all base learners are completely independent. For the directed acyclic graph-based search space, all base learners of each cell form a directed acyclic graph. On the basis of the proposed two kinds of search spaces, the research further proposes a progressive search method and algorithm based on surrogate model, which realizes the exploration in the search space from simple to complex step by step, and uses the surrogate model as a guide. Thus, the model evaluation overhead can be reduced. Finally, the research analyzes the search algorithm in terms of both time complexity and space complexity. The experiments in this paper use publicly available tabular datasets from OpenML, including 23 classification datasets and 10 regression datasets. Extensive experimental results on tabular datasets show that the performance of the deep ensemble architecture searched by the PMPAS algorithm is not only better than the existing ensemble learning models, deep learning models, deep ensemble learning models represented by deep forests, but also outperforms existing automated model selection algorithms. Performance benefits become more apparent as time budgets continue to increase. The open source address of PMPAS is https://github.com/PasaLab/PMPAS.

Keywords deep learning; deep ensemble architecture; architecture search; surrogate model

1 引 言

大数据智能分析时代，深度学习已广泛应用于图像处理[1]、语音识别[2]以及文本分析[3]等领域，并且取得了巨大的成功. 深度学习模型通常是深度神经网络(Deep Neural Network，简称DNN). DNN具有强大的表征学习能力，其通过逐层训练和学习，能够自动提取高层特征. 尽管DNN在上述领域取得了很大的成功，但是DNN也存在以下若干方面的不足. 首先，DNN需要大规模训练数据，对数据规模要求较高；其次，大部分DNN层数多、参数量巨大，导致DNN的训练需要昂贵的算力(如GPU等)和时间开销；然后，DNN包含大量的超参数，而且超参数调优难；最后，DNN并非在所有任务上都能够取得最好的效果，尤其是对于表数据等离散型数据.

针对上述DNN的不足，学术界提出了很多探索和改进. 通过对各种DNN模型的研究和分析，Zhou等人归纳并总结出了DNN能够取得成功的三个要素：逐层的计算、模型内部的特征变换以及足够的模型复杂度[4]. 基于上述三个要素，进一步提出了深度森林[5-6]，其由多个级联层构成，每个级联层是多个森林的集成. 与DNN相比，深度森林存在无需反向传播训练、内存与计算开销更小以及模型复杂度可以自适应确定等优点. 尤其是，深度森林在表格型数据上逐渐得到了广泛运用，例如多标签学习[7]、弱监督学习[8]、欺诈检测[9]、疾病预测[10-11]、5G通信[12]、声音事件检测[13]、轮廓检测[14].

深度森林为探索DNN之外的深度学习模型打

开了另一扇门．如图1所示，这种借鉴了DNN中“深度”思想，并结合集成学习的新式深度学习模型，可称为深度集成学习模型．然而，现有的深度集成模型主要以深度森林为主[5,15-18]，深度集成架构较为单一，具体表示为每个级联层的基学习器均为随机森林，而且基学习器的数量与集成方式较为固定．因此，需要探索除深度森林之外的深度集成学习模型架构．另外，在实际应用中，很难存在一种深度集成学习模型架构能够在不同数据集上均取得优异性能，尤其是对于数据特征差异较大的表格型数据集．针对不同的输入数据集以及任务场景设计一个性能优异的深度集成架构需要大量的专家知识和经验，并需要反复尝试基学习器类型、基学习器数量以及基学习器集成方式等各种超参数的组合，费时费力．因此，也需要一种高效的数据自适应的深度集成学习架构设计方法．

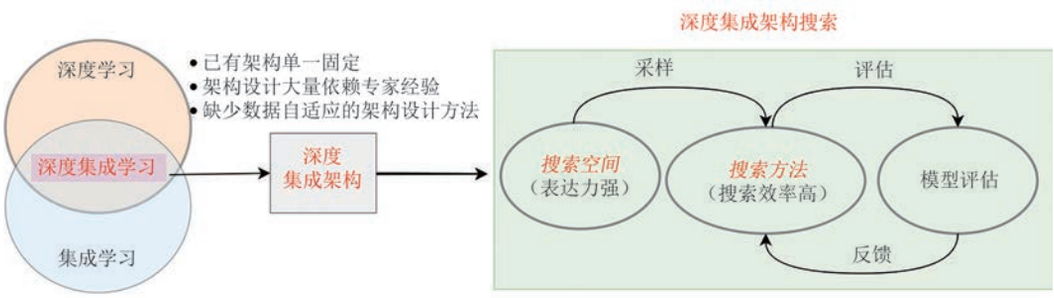

图1　深度集成架构搜索动机以及主要流程

近年来，神经架构搜索（Neural Architecture Search，简称NAS）[19-21]技术引起了工业界和学界的广泛关注．受到NAS的启发，本文将研究设计面向深度集成学习模型的架构搜索方法，针对特定的数据集和任务场景搜索性能优异的深度集成架构，从而降低深度集成学习模型的设计门槛．图1显示了深度集成架构搜索的主要流程．然而，深度集成架构的自动化搜索面临以下技术挑战：首先，在搜索空间设计方面，需要研究探索通用的且表达能力强的深度集成架构搜索空间，不仅能够包含以深度森林为代表的深度集成学习模型架构，而且能够涵盖更多未被探索的深度集成架构；其次，在搜索算法设计方面，基学习类型、数量以及集成方式的互相组合导致搜索空间爆炸性增长，因此需要设计高效的深度集成架构搜索方法与算法，在保证模型预测性能的前提下，提升搜索效率．

为此，本文从搜索空间和搜索算法两个层面，研究提出了一种高效的基于代理模型的渐进式深度集成架构搜索方法PMPAS（Proxy Model—based Progressive Architecture Search）．首先，通过归纳分析已有深度集成学习模型的特点，对深度集成架构进行形式化定义．在此基础上，研究提出了两种全新的深度集成架构搜索空间，即基于完全并行的搜索空间和基于有向无环图的搜索空间．

其次，在上述两种搜索空间的基础上，研究提出了基于代理模型的渐进式搜索方法与算法，实现从简单到复杂逐步地在搜索空间中进行探索，并采用代理模型作为指导，从而降低模型评估开销．本文研究提出的深度集成架构搜索方法高度适合并行化，并且能够支持分类、回归等常见的机器学习任务．综上所述，本文聚焦于大数据智能分析算法设计，针对深度集成学习，研究提出一种自动化深度集成架构的设计方法与算法．

本文主要研究点和贡献点如下：

（1）据我们所知，本文提出的PMPAS是第一个面向深度集成学习模型的架构搜索方法；

（2）研究给出了深度集成学习模型架构的形式化定义，并研究设计了基于完全并行和基于有向无环图的两种深度集成架构搜索空间；

（3）研究提出了基于代理模型的渐进式深度集成架构搜索方法与算法，提出从简单到复杂逐步构建深度集成架构的基本思路，并采用神经网络代理模型，指导整个搜索过程；

（4）在分类、回归等表格型数据集上的大量实验结果表明，通过PMPAS算法搜索得到的深度集成架构，其性能不仅优于已有的集成学习方法以及以深度森林为代表的深度集成学习模型，而且优于已有的自动化机器学习算法auto-sklearn．

2 相关工作

2.1 深度集成学习

集成学习主要包含Boosting、Stacking和Cascade三种方法. Boosting方法的代表算法为Adaboost[22],其基本思想是在集成过程中不断改变训练数据集的概率分布. 与Boosting较为相似的方法是Bagging方法,其主要策略在于自助采样[23]. 目前代表性的Bagging方法为随机森林[24]. Stacking[25]是一种多级训练算法,其核心思想在于利用训练数据集训练初级学习器,将这些初级学习器的输出看成是新的训练数据集,然后在新的训练数据集上训练元学习器.

为了缓解Stacking方法容易产生过拟合的问题,Gama等人采用了一种称为Cascade[26]的级联结构. Cascade针对分类问题而设计,其每一层由多个基学习器组成. 在训练过程中,除第一层外,每一层的输入数据均来自于上一层训练数据集的类别概率输出和原始训练数据集的拼接,上述训练过程直到达到预先设定的最大层数T,同时将第T层的输出作为Cascade方法的输出.

顾名思义,深度集成学习是在上述集成学习的基础上,引入深度神经网络中的"深度"思想,提升模型的复杂度以及表示能力. Zhou等学者提出的深度森林[5-6]就是一种代表性的深度集成学习模型. 深度森林以随机森林作为基本学习器进行集成. 级联森林结构是深度森林的核心部分. 如图2所示,每个级联层由两种随机森林构成,分别为随机森林和完全随机森林.

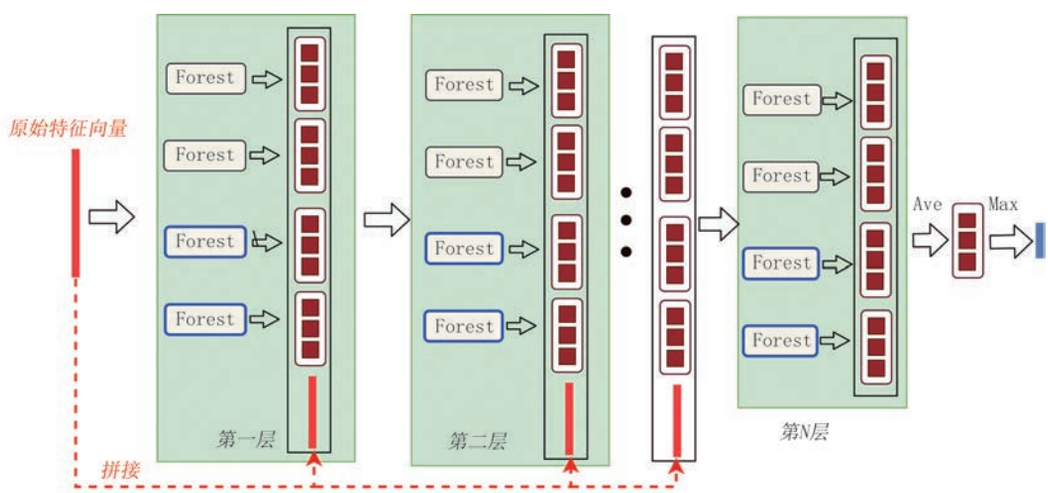

图2 深度森林中级联森林结构示意

在级联森林结构训练过程中,对于每一层都会得到该层对应的表现,如果连续T层的性能不发生明显变化,则提前终止训练,进而得到最终的深度森林模型. 因此,深度森林级联森林结构的层数可以自适应调整,而且在整个过程中,没有梯度计算,这相比DNN无疑有着较大的性能优势.

针对深度森林运行过程中巨大的内存开销导致的训练效率低下问题,研究人员对原有的级联森林结构层与层之间进行优化,提出了一种称为置信度筛选(Confidence Screening)的技术[16-17]. 置信度筛选机制会将分类置信度高于阈值的样本丢弃,仅仅将低于阈值的类别概率和原始数据集拼接传入下一层,这一步大大减少了内存开销. 除了置信度筛选外,其他筛选技术如哈希和窗口筛选也相继被提出[18]. 与深度森林类似,Feng等人提出了多层的梯度提升树[27].

为了提升深度森林的训练效率,Zhu等人提出了一种分布式的深度森林训练方法[28],可将深度森林的训练性能提升10倍左右. Chen等人提出了一种自适应子森林划分的分布式训练方法[29].

2.2 自动化机器学习

自动化机器学习(Automated Machine Learning,简称AutoML)[30-31]是提高整个机器学习模型设计效率的关键技术,让机器替代人工完成模型设计,从而降低模型设计技术门槛,大幅提升建模效率. 神经网络架构搜索NAS技术[19-21]是AutoML中一项代表性的技术,用于实现深度神经网络架构的自动化搜索,近年来在计算机视觉领域取得了较大的成功,并引起

了学界和业界的广泛关注.

NAS的核心研究问题主要包含两个方面，即搜索空间和搜索策略的设计. 在图像处理领域，代表性的搜索空间为基于Cell的搜索空间[32]. 整个神经网络架构由多个Cell堆叠而成，只需要搜索Cell的架构即可. 因此，搜索空间的规模可大幅下降. 在搜索策略方面，Google提出的基于强化学习的NAS算法[33]需要800块GPU训练28天，算力及时间开销巨大. 为了提升NAS的计算性能，让NAS的算力和时间开销降低至可接受的范围内，研究人员开始探索其他各种NAS方法，如基于权重共享的NAS[34]、基于遗传算法的NAS[35-36]、基于贝叶斯代理模型的NAS[37-38]、基于渐进式搜索的NAS[39-40]、基于One-Shot的NAS[41-42]、可微分的NAS[43-46]等. 其中，可微分的NAS方法凭借其较高的搜索效率以及优异的网络架构预测性能，逐渐得到了广泛使用.

目前的NAS工作主要聚焦在深度神经网络架构的搜索. 受NAS技术的启发，本文将聚焦深度集成学习模型，在深度集成架构定义及研究基础上，实现深度集成架构的自动化搜索，从而进一步丰富自动化机器学习的研究内容.

3 深度集成架构搜索定义

本节首先介绍深度集成架构的形式化定义，然后给出深度集成架构搜索的定义.

3.1 深度集成架构定义

以分类问题为例，给定具有K个类别的数据集$D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_N,y_N)\}$，其中$N$是数据集的样本数量，$D$中的每个元素是由实例以及对应的标签构成的样本点，即实例$\boldsymbol{x}_i=(x_1,x_2,\cdots,x_m)^T\in\mathcal{X}\subseteq\mathbf{R}^m$，标记$y_i\in\mathcal{Y}=\{c_1,c_2,\cdots,c_K\}$，其中$\mathcal{X}$是由实例构成的特征空间，$\mathcal{Y}$是由标签构成的输出空间.

定义1(基学习器). 基学习器$\varepsilon\in\mathcal{E}:\boldsymbol{x}\to y$，是从特征空间到输出空间的映射，$\mathcal{E}=\{\varepsilon|y=\varepsilon(\boldsymbol{x}),\boldsymbol{x}\in\mathcal{X},y\in\mathcal{Y}\}$代表全体基学习器构成的集合.

通常来说，基学习器ε包含许多参数，所以常记为$\varepsilon=\varepsilon_\theta(D)$，其中$\theta$是参数向量. 将$\varepsilon$在$D$上进行训练，构建出来的模型被称为预测器，记为$\varepsilon=\varepsilon_\theta(\hat{\boldsymbol{x}},D)$，其中$\hat{\boldsymbol{x}}$表示待预测实例，预测器对于每个$\hat{\boldsymbol{x}}$都会输出对应的标签或者类别概率分布$\boldsymbol{p}=(p_1,p_2,\cdots,p_K)^T$，其中$p_k$表示$\hat{\boldsymbol{x}}$属于类别$c_k$的概率，即$p_k=\Pr(y=c_k|\hat{\boldsymbol{x}})$，将这种输出类别概率分布的预测器记为$\tilde{\varepsilon}=\tilde{\varepsilon}_\theta(\hat{\boldsymbol{x}},D)$. 可以由$\tilde{\varepsilon}$得到$\varepsilon$，即取$\tilde{\varepsilon}$中概率最大的类别作为最终分配给$\hat{\boldsymbol{x}}$的类别标签$\hat{y}$，即有：

$$\hat{y}=\underset{k\in\{1,2,\cdots,K\}}{\operatorname{argmax}}\ \Pr\left(y=c_k|\hat{\boldsymbol{x}}\right)$$

定义2(增广算子). 增广算子$c(\boldsymbol{x},\tilde{\varepsilon}(\boldsymbol{x},D))$，该增广算子会将$\boldsymbol{x}$和在$\tilde{\varepsilon}$上的类别概率输出$\boldsymbol{p}$进行拼接操作，得到新的实例(或称增广输出)$\boldsymbol{x}'$：$\boldsymbol{x}'=c(\boldsymbol{x},\tilde{\varepsilon}(\boldsymbol{x},D))=(x_1,x_2,\cdots,x_m,p_1,p_2,\cdots,p_K)^T\in\mathbf{R}^{m+K}$. 增广算子也称构建算子(constructive operator[26]).

将上述增广算子运用在D的每个实例上，得到了新的数据集$D'=\{(\boldsymbol{x}'_1,y_1),(\boldsymbol{x}'_2,y_2),\cdots,(\boldsymbol{x}'_N,y_N)\}$. 相比于原始数据集$D$，$D'$中的每个实例多了$K$个增广特征.

接下来介绍集成映射的定义，该定义给出了深度集成架构每一层内各基学习器之间的连接方式.

定义3(集成映射). 给定包含M个基学习器集合$\mathcal{E}=\{\varepsilon_1,\varepsilon_2,\cdots,\varepsilon_M\}$，$\mathcal{E}'=\mathcal{E}\cup\{0\}$，集成映射$\varphi:\mathcal{E}\to\mathcal{E}'$是从$\mathcal{E}$到$\mathcal{E}'$的映射. 将$\varphi$运用在$\mathcal{E}$的每个元素上，得到了新的集合$\varphi(\mathcal{E})=\{\varepsilon'_1,\varepsilon'_2,\cdots,\varepsilon'_M\}$，设$\varphi(\varepsilon_j)=\varepsilon'_j=\varepsilon_k$，$\varepsilon'_j=0$表示不存在与$\varepsilon_j$连接的基学习器，$\varepsilon'_j=\varepsilon_k$表示$\varepsilon_k$与$\varepsilon_j$存在前后连接关系，即$\varepsilon_k$的增广输出是$\varepsilon_j$的输入.

若将上述的增广算子运用在所有的基学习器上，并结合集成映射则构成了下面的级联层.

定义4(级联层). 给定基学习集合$\mathcal{E}=\{\varepsilon_1,\varepsilon_2,\cdots,\varepsilon_M\}$，定义在$\mathcal{E}$上的增广算子$c$和集成映射$\varphi$，级联层$L$定义为由$\mathcal{E},c,\varphi$构成的三元组，即$L=<\mathcal{E},c,\varphi>$.

假设对于$\mathcal{E}$中的任一基学习器，满足$\varphi(\varepsilon_j)=0(j=1,2,\cdots,M)$，那么级联层的输出相比于输入数据集$D$而言，实例数量一样，但是多了$M\times K$个增广特征，其中每个基学习器$\varepsilon_j$都会贡献$K$个新特征.

上述定义的级联层是构建深度集成架构的基础，深度集成架构即是依次生成总共T个级联层的学习模型. 可以看到，这种学习模型是一种序列组合，每一层的输出都作为下一层的输入，直到第T层得到最终的输出，下面给出深度集成架构的定义.

定义5(深度集成架构). 给定基学习器集合$\mathcal{E}=\{\varepsilon_1,\varepsilon_2,\cdots,\varepsilon_M\}$，$c$为增广算子，$\varphi$为集成映射，级联层

$L=<\mathcal{E},c,\varphi>$. 初始级联层 L_1 输出为 L_1'，在 L_1' 的基础上继续运用 c 和 φ 得到新的级联层 L_2 对应的输出 L_2'，将这种基本的序列操作记为 ∇，即有 $L_2'\nabla L_1'=c\left(L_1',\varphi(\mathcal{E})\right)=c\left(c\left(D,\varphi(\mathcal{E})\right),\varphi(\mathcal{E})\right)$. 对于具有 T 层的级联层，重复运用 ∇，最终结果为 $L_T'\nabla L_{T-1}'\nabla\cdots\nabla L_1'$，深度集成架构即定义为 $\mathrm{a}=<\mathcal{E},c,\varphi,T,\nabla>$.

在实际应用中，深度集成架构级联层层数 T 根据训练数据集自适应确定. 具体来说，对于每层级联层，通过交叉验证得到对应的预测性能，若性能不能提升就停止级联层的增长，从而得到最终的层数 T. 除了初始级联层以外，其他所有层相比原始的数据集 D 都会多出若干增广特征，这些增广特征对于增强深度集成架构的泛化能力具有重要作用[26]. 在深度集成架构的推理预测过程中，实例 $\hat{\boldsymbol{x}}$ 会依次通过每层级联层，并在第 T 层，得到最终的预测.

3.2 深度集成架构搜索定义

在上述深度集成架构定义的基础上，下面给出自动化深度集成学习的主要研究内容即深度集成架构搜索的定义.

定义6(深度集成架构搜索). 给定基学习器集合 $\mathcal{E}=\{\varepsilon_1,\varepsilon_2,\cdots,\varepsilon_M\}$，$\Phi$ 为全体集成映射构成的集合，c 为增广算子，由全体深度集成架构构成的集合表示为 $\mathcal{A}=\left\{<\mathcal{E},c,\varphi,T,\nabla>|\varphi\in\Phi,T\in N\backslash\{0\}\right\}$，$\mathcal{A}$ 也被称为深度集成架构的搜索空间. 深度集成架构搜索的目标是在给定数据集 D 上，在搜索空间 $\mathcal{A}$ 上找到最优的基学习器集合 $\mathcal{E}^*\subseteq\mathcal{E}$ 以及集成映射 $\varphi^*\in\Phi$，从而使得最终构成的深度集成架构 $\mathrm{a}=<\mathcal{E}^*,c,\varphi^*,T,\nabla>\in\mathcal{A}$ 的泛化误差最小.

理论的泛化误差难以直接获得，实践中通常采用 K 折交叉验证的方式近似代替泛化误差. 首先将数据集 D 划分成不相交的训练数据集 D_{train} 和验证数据集 D_{val}，在 D_{train} 训练深度集成架构，通过学习到的模型在 D_{val} 上获得预测效果，这样，深度集成架构搜索问题可以建模为如下形式：

$$\mathrm{a}^*\in\underset{\mathrm{a}\in\mathcal{A}}{\operatorname{argmin}}\frac{1}{K}\sum_{i=1}^{K}\mathcal{L}\left(\mathrm{a},D_{train}^{(i)},D_{val}^{(i)}\right)$$

其中 $\mathcal{L}(\mathrm{a},D_{train}^{(i)},D_{val}^{(i)})$ 代表 a 在 $D_{train}^{(i)}$ 上训练，在 $D_{val}^{(i)}$ 上进行验证时通过损失函数 $\mathcal{L}$ 得到的损失.

与神经网络架构搜索不同，深度集成架构搜索需要考虑每个级联层中基学习器集合(包括基学习类型和数量)以及基学习之间的连接方式. 下面将给出两种具体的深度集成架构搜索空间设计方法.

4 深度集成架构搜索空间设计

以深度森林为代表的深度集成学习虽然在表格型数据集上取得了优异的效果，但是其结构设计精巧，严重依赖专家经验，同时其包含的基学习器即随机森林也并不能在所有数据集上都取得最优的效果，因此仍然需要设计全新的搜索空间，使得可以针对具体任务场景和数据集自动从搜索空间中寻找一个最优的深度集成架构.

4.1 基于完全并行的搜索空间设计

在给出具体的搜索空间定义之前，需要确定构成深度集成架构中的基学习器. 为了尽可能地保证通用性与性能，本文采用的基学习器既包含了传统的机器学习算法，如逻辑回归、决策树、k-近邻算法和支持向量机等，也包含了效果优异的一些集成学习模型，如 Adaboost 和随机森林等. 下文将基学习器简称为 Cell.

受到深度森林模型以及由 Gama 提出的级联结构 Cascade 的启发，本文首先研究提出基于完全并行的深度集成架构搜索空间.

定义7(完全并行搜索空间). 设级联层 $L=<\mathcal{E},c,\varphi>$，若对于 $\mathcal{E}$ 中的所有基学习器 ε，它们之间没有任何的连接关系，即对 $\forall\varepsilon\in\mathcal{E}$，都有 $\varphi(\varepsilon)=\{0\}$，那么就称级联层 L 确定了一个完全并行(Completely Parallel，简称 CP)的深度集成架构 a_{CP}，由全体 CP 类型的深度集成架构构成的集合称为 CP 搜索空间，简记为 $\mathcal{A}_{\mathrm{CP}}$.

图3给出了 CP 搜索空间中深度集成架构级联层 Layer 的基本结构. 无论是 Gama 提出的 Cascade 结构，还是 Zhou 等人提出的深度森林模型，都不过是 $\mathcal{A}_{\mathrm{CP}}$ 的一个特例. 具体来说，深度森林中的每一个森林都是一个 Cell，可以表示为总共有8个 Cell、两种基学习器类型，分别是4个随机森林和4个完全随机森林. 这些 Cell 按照集成映射 $\varphi(\cdot)=0$ 的连接方式组成了级联层. 每层 Layer 有多个 Cell 构成，由 CP 搜索空间定义的集成映射 $\varphi(\cdot)$ 的定义可知，这些 Cell 之间是相互独立的，无依赖关系. 在每层 Layer 中，各个 Cell 在训练数据集上训练后得到的模型都是不一样的. 各个 Cell 给出类别概率输出和输入拼接传递到下一层. $\mathcal{A}_{\mathrm{CP}}$ 由 Cell 和 Layer 构成，首先确定 Layer 中的每个 Cell 的类型，然后由若干个 Cell 共同构成了一层 Layer. 按照深度森林的训练方式，最

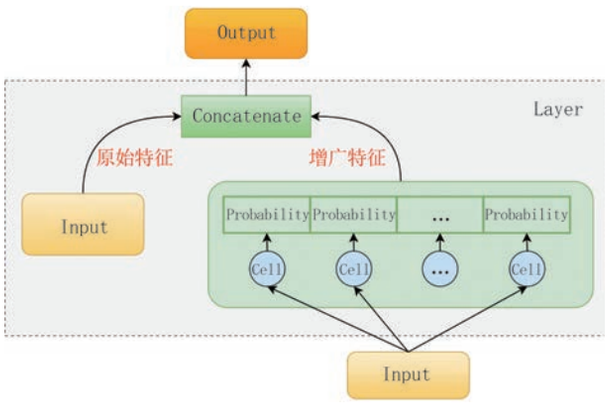

图3　完全并行深度集成架构中Layer的基本结构

终的层数T可以自适应确定. 图4给出了完全并行深度集成架构的训练过程.

首先,对于给定的初始训练数据集D_{train},第一层L_1会在每个Cell上进行完整的训练,通过交叉验证的方式产生每个Cell的类别概率输出$\boldsymbol{p}_1$,然后将所有的类别概率输出$\boldsymbol{p}_1$和初始训练数据集D_{train}拼接构成新的数据集D'_{train},同时计算出L_1对应的表现Score(L_1). 然后将D'_{train}传递到下一层L_2,在L_2上利用D'_{train}训练各个Cell,得到类别概率输出$\boldsymbol{p}_2$,$\boldsymbol{p}_2$再和D'_{train}拼接得到新的数据集D''_{train},同时得到该层对应的表现Score(L_2). 重复上述的过程,若早停轮数为R,当Score(L_{T+R})$\leqslant$Score(L_T)的时候,训练终止,同时得到深度集成架构的层数T. 利用$\mathrm{a_{CP}}$进行预测时,待预测实例$\hat{\boldsymbol{x}}$依次通过各个级联层,即由级联层给出的类别概率预测和数据集拼接传递到下一层,最终在第T层得到$\hat{\boldsymbol{x}}$的预测向量$\hat{\boldsymbol{p}}$,选取$\hat{\boldsymbol{p}}$中概率最大的类别作为最终的预测标签.

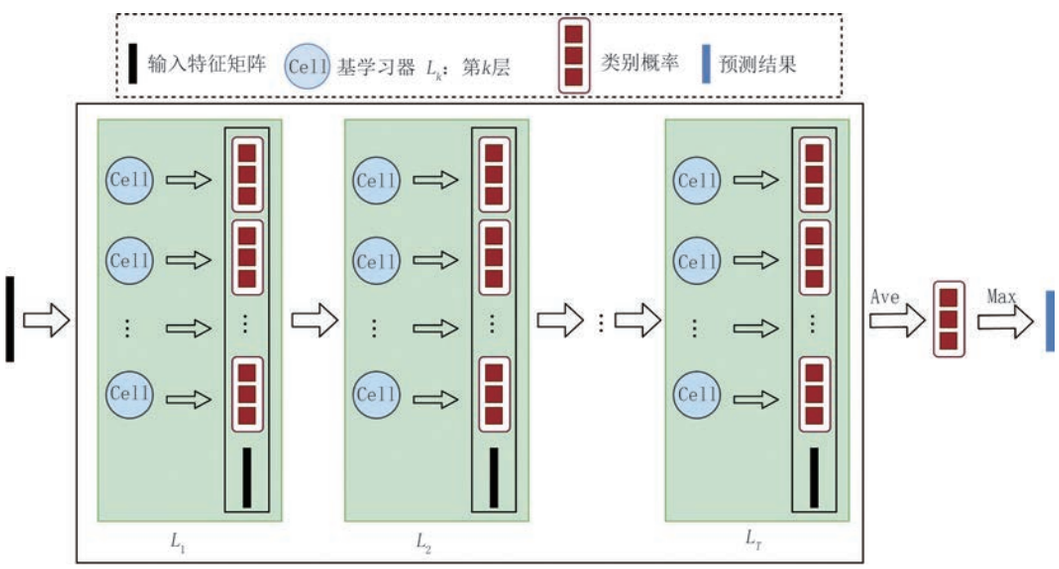

图4　完全并行深度集成架构的训练过程

4.2　基于有向无环图的搜索空间设计

观察DNN的网络层和神经元结构可以发现,一个神经元可以和多个神经元通过参数化的信息连接,最后通过反向传播计算出这些参数. 每个神经元通过计算都保留了对于原始训练数据集的某些内在模式,而多个神经元之间相互传递这些信息,每一层实现一次数据蒸馏(Data Distillation). 通过逐层地处理,这些类似“筛子”的神经网络层就会学到越来越好的表示,相应的“筛子”也会越来越精细. 借鉴上述DNN的设计思想,在深度森林和Cascade结构的基础上,本文研究设计了另一种基于有向无环图(Directed Acyclic Graph,简称DAG)的搜索空间,简记为$\mathcal{A}_{\mathrm{DAG}}$. 简单来说,$\mathcal{A}_{\mathrm{DAG}}$和$\mathcal{A}_{\mathrm{CP}}$最大区别在于$\mathcal{A}_{\mathrm{DAG}}$的集成映射$\varphi(\cdot)$和$\mathcal{A}_{\mathrm{CP}}$有所区别,下面给出$\mathcal{A}_{\mathrm{DAG}}$搜索空间的定义:

定义8(有向无环图搜索空间). 设级联层$L=<\mathcal{E},c,\varphi>$,若对于$\mathcal{E}$中的每个基学习器$\epsilon$都至多有一个基学习器与之对应,即对$\forall\epsilon\in\mathcal{E},\varphi(\epsilon)\in\mathcal{E}'=\mathcal{E}\cup\{0\}$,那么就称由$L$确定了一个有向无环图类型的深度集成架构$\mathrm{a_{DAG}}$,由全体DAG类型的深度集成架构构成的集合称为DAG搜索空间,简记为$\mathcal{A}_{\mathrm{DAG}}$.

相较于$\mathrm{a_{CP}}$,$\mathrm{a_{DAG}}$的基本构建单元分为Cell和Unit两个级别. 首先由Cell组成Unit,然后由这些Unit再按照特定的连接方式构成Layer. 图5给出了有向无环图搜索空间中Unit的基本结构.

如图5所示,类似于DNN,每个Cell可以和其他Cell相互连接,这种连接不是简单的连接,而是需要

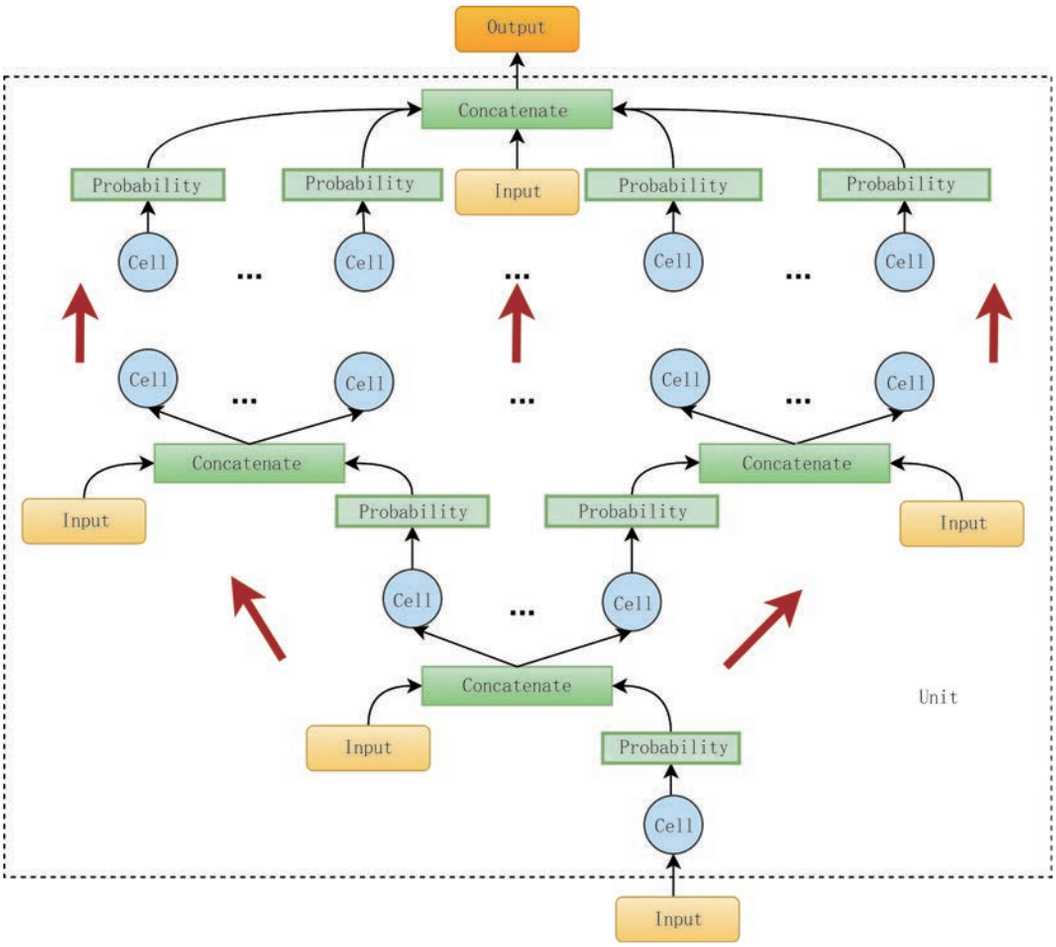

图5 有向无环图搜索空间中Unit的基本结构

上一个Cell的输出和原始特征数据集进行拼接得到新的数据集，以此作为下一个Cell的输入. a_{DAG} 整个的训练和预测过程仍然和 a_{CP} 一样，模型的复杂度即层数 T 是根据数据集以及早停轮数 R 自动确定. a_{DAG} 的每层Layer由一个或者多个Unit组合而成，这些Unit之间没有关联，因而可以进行并行训练和预测. 图6展示了有向无环图深度集成架构训练过程. $\mathcal{A}_{DAG}$ 表示的深度集成架构搜索空间范围相较于 $\mathcal{A}_{CP}$ 又进一步扩大，即有 $\mathcal{A}_{CP}\subseteq\mathcal{A}_{DAG}$，这种搜索空间的扩大并不是盲目的，而是根据DNN中各个神经元之间相互连接的方式的启发，当前层每个Cell学习到的表示有可能和其他Cell学习到的表示相互补充.

5 基于代理模型的渐进式深度集成架构搜索

本节首先介绍渐进式搜索策略的实现方法，在对未经优化的搜索空间大小进行定量分析的基础上，提出基于部分架构保留的优化搜索策略. 其次，介绍基于神经网络的代理模型设计，通过代理模型对部分架构的性能进行预测，从而避免评估所有可能的架构，降低评估计算开销. 然后，介绍基于代理模型的渐进式搜索方法PMPAS的整体框架.

5.1 渐进式搜索策略

深度集成架构搜索问题面临庞大的搜索空间，需要首先确定每个Cell的类型，然后再确定Cell之间的连接方式. 在深度集成学习中，每个Cell都是泛化能力较强的机器学习算法. 因此，可以充分利用该特性，从Cell数量较少的深度集成架构开始搜索，逐步地构建Cell数量较多的深度集成架构. 简单的深度集成架构，其在验证集上的性能评估相对较为容易，而且在从简单到复杂逐步构建的过程中，产生的很多新架构与之前评估过的架构只有稍微的不同，此时引入代理模型，对这些新的架构进行预测，将会大大加速整个架构搜索过程.

从简单到复杂逐步构建架构的过程中，不同搜索策略产生的深度集成架构数量也会有很大的差别. 下面将具体分析广度优先搜索与集束搜索策略所对应的搜索空间大小.

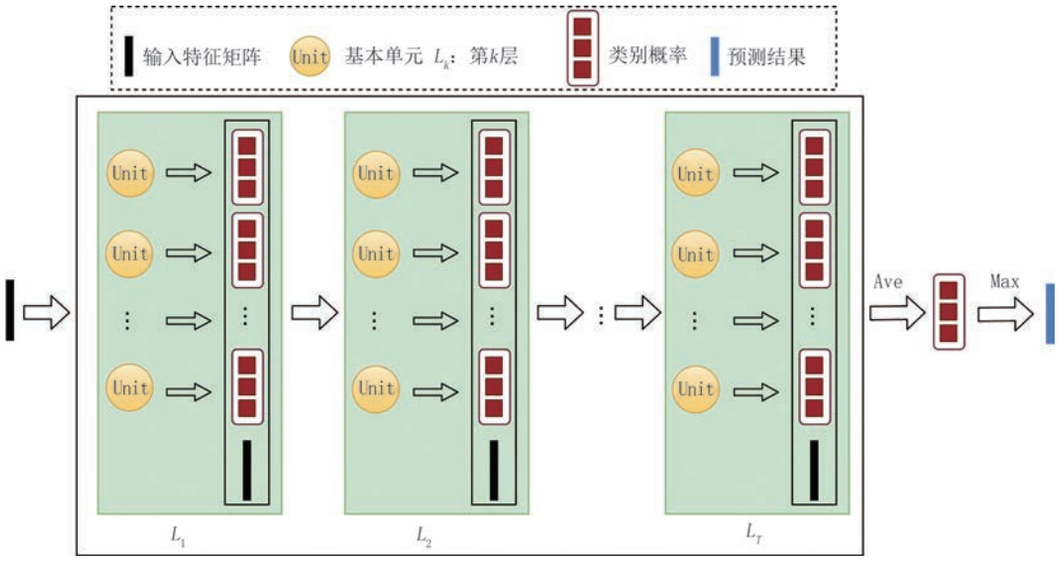

图6　有向无环图的深度集成架构训练过程

5.1.1　广度优先搜索

若采用广度优先搜索策略(Breadth-First Search)，以$\mathcal{A}_{\mathrm{DAG}}$搜索空间和分类任务为例，其初始的Cell数量为1. 若每个Cell共包含了M种机器学习算法，那么Cell数量为1的深度集成架构总数为M. 当Cell数量增加到2的时候，需要考虑Cell之间的连接方式，将产生两种可能的连接方式. 图7给出了在广度优先搜索策略下Cell数量增加情况示意图.

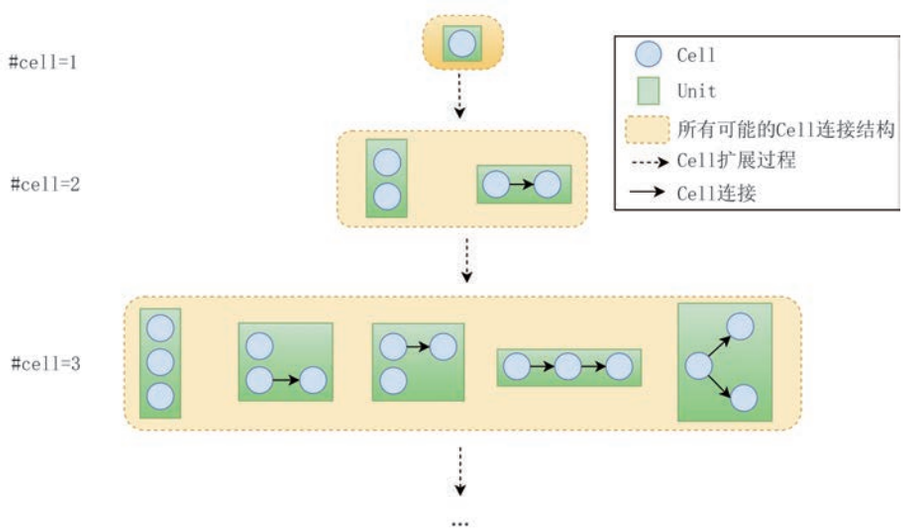

图7　广度优先策略搜索方式Cell数量增加情况示意

对于新扩展的1个Cell，由于仍然有M种可能的机器学习算法，因而在Cell为2时，包含$M^2\times 2$种可能的深度集成架构. 经过推导可以发现，若限制每个深度集成架构的最大Cell数量为N，那么$\mathcal{A}_{\mathrm{DAG}}$可能的架构数约$O(M^N\cdot N!)$. 即使对于较$\mathcal{A}_{\mathrm{DAG}}$更为简单的$\mathcal{A}_{\mathrm{CP}}$，可能的架构数也会达到$O(M^N)$. 从上述分析可以看到，整个深度集成架构的搜索空间十分庞大. 在如此庞大的搜索空间下，完整地评估所有的深度集成架构显然是不现实的，这也给从搜索空间中寻找最优的深度集成架构带来了巨大挑战. 因此本节将引入优化的广度优先搜索方式，即集束搜索方式，从搜索空间中寻找最优的深度集成架构.

5.1.2　集束搜索

在采用基本的广度优先方式增加Cell数量的过程中，随着Cell数量的增加，将产生大量新的深度集成架构. 如图8所示，为了减少待评估的架构数量，降低评估开销，对广度优先搜索过程进行剪枝，通过

构建代理模型，在Cell数量的增加过程中仅仅保留预测性能最好的K个深度集成架构，然后再基于K个最好的深度集成架构扩展新的架构．扩展方式为：将Cell的数量增加1，然后将新增加的Cell与原有架构进行连接，形成新的架构对应的DAG图．如图7所示，不同的连接方式（串行或并行）以及基学习器类型对应不同的架构．上述过程不仅整体所需要评估的架构数量大大减少，而且能够实现从简单到复杂逐步构建深度集成架构的目标．这种方式也可称之为集束搜索．

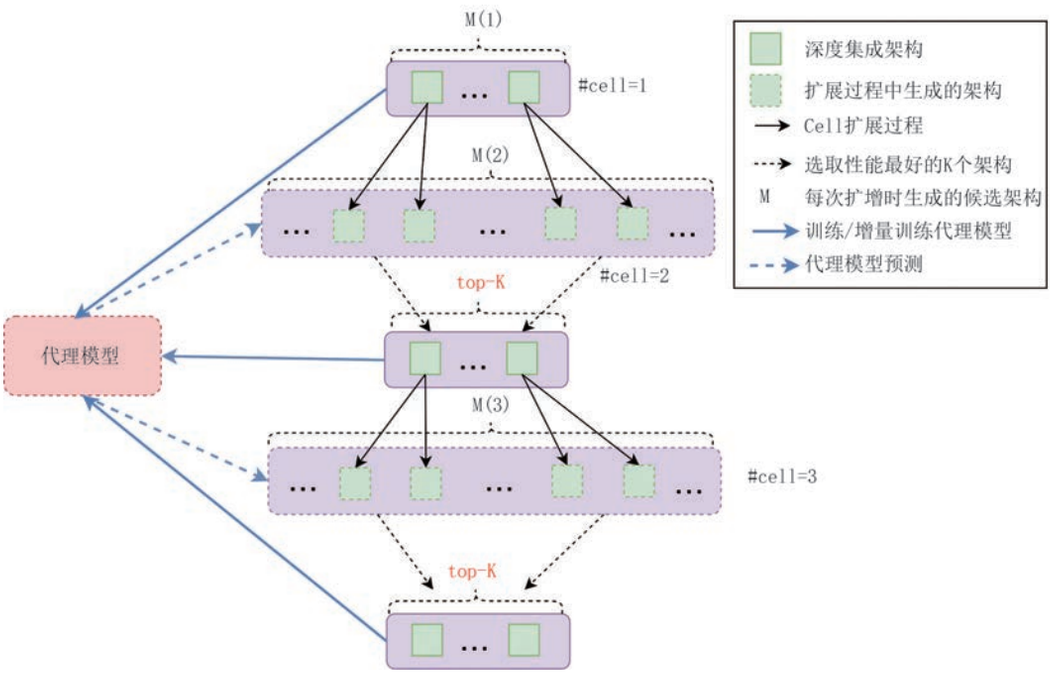

图8 基于代理模型的架构生成

若每个Cell共包含M种机器学习算法，每个深度集成架构的最大Cell数量为N．对于$\mathcal{A}_{\mathrm{DAG}}$搜索空间而言，若在增长过程中每次保留$K$个架构，虽然总的搜索空间大小不变，但是若每次只对Cell增加过程中的K个深度集成架构进行完整的评估，那么最终需要完整评估的架构数量仅仅为$O(KN)$．由于每次保留Top-K个性能最好的架构，部分Cell之间的连接方式将不会出现，因此采用集束搜索可以有效地对搜索空间进行剪枝．

5.3 代理模型设计

在Cell的数量进行增长的过程中，产生了很多新的架构．如果评估所有产生的新架构，则将造成非常大的计算开销．为了降低评估开销，可采用基于SMBO[47]的方法设计代理模型，利用代理模型预测架构提升性能．代理模型实质上是一种能够以较小的计算代价预测架构性能的模型．一般情况下，代理模型适用于模型评估开销较大的场景，其以架构的统一表示作为输入，输出为架构的预测性能，从而无需代价较高的真实评估．

具体来说，本节将采用基于长短期记忆循环神经网络（Long Short-Term Memory Recurrent Neural Network，简称LSTM）[48]的代理模型，这种循环神经网络对于处理具有时序依赖关系输入的问题效果较为优异，通常会结合Embedding层，使得网络的学习能力进一步的提升．

5.3.1 基于循环神经网络的代理模型

根据深度集成架构搜索的特点，代理模型的设计至少需要满足以下三个方面的要求．

（1）能够处理变长的输入．由于Cell数量的增长过程是一个动态的过程，因此，需要能够预测具有不同Cell数量的深度集成架构的性能．代理模型不仅要给出Cell数量为n的时候架构的预测性能，还要给出Cell数量为$n+1$的时候架构的预测性能．

（2）代理模型的预测输出需要和架构的真实性能相关．代理模型没有必要输出每个架构的真实表现，只需输出架构的相对表现性能，从而反映出架构之间的差异即可．

（3）代理模型的预测开销不能太高．代理模型的引入，对于架构搜索过程是额外的计算开销，因此在保证预测性能的同时，选用预测时间开销较低的

代理模型，从而能够较快速地输出预测性能.

基于以上三个方面的要求，本文采用LSTM作为代理模型，图9给出了该代理模型的设计.

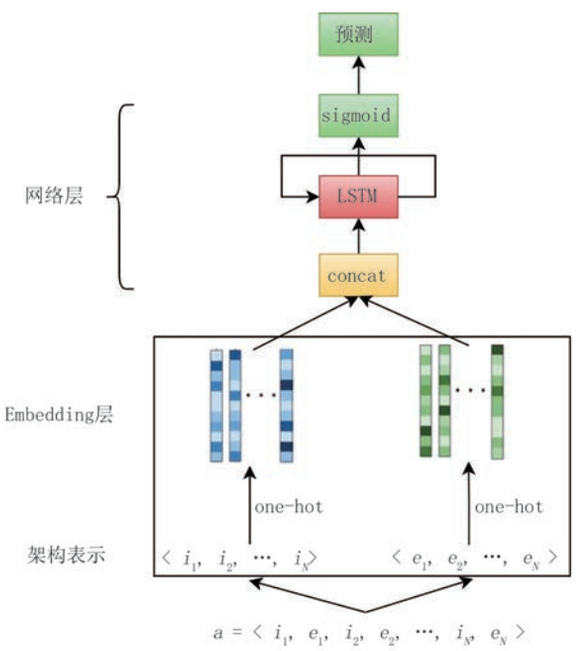

图9　代理模型设计

在图9给出的代理模型中，假设架构搜索空间的最大Cell数量为N，根据前面的搜索空间定义可知，无论是$\mathcal{A}_{\mathrm{CP}}$还是$\mathcal{A}_{\mathrm{DAG}}$，搜索空间中每个架构都可以表示为$\mathrm{a}=<i_1,e_1,i_2,e_2,\cdots,i_N,e_N>$，其中$i_k$代表第$k$个Cell对应的输入Cell索引，$i_k\in[0,k-1]$，$i_k=0$表示输入为原始数据集. e_k代表第k个Cell对应的基学习器类型，可表示为M种机器学习算法组成的列表中的索引. 进一步将架构表示划分为两个部分，$\varepsilon=<i_1,e_1,i_2,e_2,\cdots,i_N,e_N>=I\cup E=<i_1,i_2,\cdots,i_N>\cup<e_1,e_2,\cdots,e_n>$，对于I和E中的每个元素，在LSTM的每一步中，使用大小分别为card(I)和card(E)的one-hot向量进行编码[49]，然后传入Embedding层，再将Embedding层的输出进行拼接(Concatenate)，传入LSTM单元. LSTM隐藏状态通过一个使用sigmoid激活函数的全连接层. 损失函数采用了L_1损失(L_1 loss)，L_1损失定义如下：

$$L_1\mathrm{loss}=\sum_{i=1}^{N'}\left|y_i-f(x_i)\right|$$

其中N'为数据集的大小，y_i为第i个样本(即深度集成架构)对应的真实标签，$f(\cdot)$为代理模型，$f(x_i)$表示x_i的预测输出.

经过上述步骤，代理模型最终会输出一个[0,1]之间的值，代表当前架构在验证集上的预测性能.

5.3.2　代理模型指导过程

在集束搜索策略中，每次新增加的需要完整训练的架构数量相对而言较少. 因此在代理模型训练阶段，将充分利用所有已完整训练过的架构.

假设当前的Cell数量为i，集束搜索的参数为K，有M种类型的Cell. 当Cell数量为1时，完整训练的架构构成集合为$S_1=\{\mathrm{a}_1^{(1)},\mathrm{a}_2^{(1)},\cdots,\mathrm{a}_M^{(1)}\}$，在Cell数量为$i(i\geqslant 2)$时进行完整训练的架构集合为$S_i=\{\mathrm{a}_1^{(i)},\mathrm{a}_2^{(i)},\cdots,\mathrm{a}_K^{(i)}\}$，代理模型会在已有的架构集合$S=S_1\cup S_2\cup\cdots\cup S_i$上利用$\mathrm{L}_1$ loss计算出损失. 在计算出损失后，再按照时间反向传播计算方法计算梯度，从而实现通过优化L_1 loss来更新神经网络内部参数的目的，使得神经网络对于以后产生的新架构，能够给出更好的预测. 由上述的梯度计算和优化损失过程即完成了神经网络指导的渐进式搜索.

5.3.3　搜索算法理论分析

首先，从深度集成模型架构评估数量的角度，对搜索算法进行分析. 以$\mathcal{A}_{\mathrm{DAG}}$搜索空间为例，在初始化阶段(即Cell的数量为1时)，深度集成架构总数为基学习器的数量M. 在渐进式搜索阶段，每次扩增Cell时只保留最优的K个架构进行实际评估. 因此，需要评估的架构总数为$M+K\times(N-1)$. 实际评估的模型数量主要由最大的Cell数量N和集束搜索参数K来控制. N和K越大，需要真实评估的架构数量就越多，计算开销也就越大，但代理模型的预测也会更加准确，对搜索空间的探索也更加全面. 但是，如果参数N和K设置过大，将会造成由于代理模型的训练开销而导致其在整个自动化架构搜索过程中耗时占比增加. 而N和K越小则需要真实评估的模型越少，由于代理模型的训练数据集较小，代理模型的预测效果会有所降低. 因此，权衡参数N和K是能否进行有效架构搜索的重要方面.

其次，从计算复杂度和空间复杂度的角度对搜索算法进行分析.

计算复杂度分析：搜索过程的主要计算开销包含模型评估总开销、代理模型预测总开销以及代理模型训练总开销. 令$T(\mathrm{a}_j^{(i)})$为第i次扩增时(即Cell的数量为i)，评估第j个模型的计算开销. 根据架构评估的总数量，可得到模型评估的计算总开销为$\sum_{j=1}^{M}T\left(\mathrm{a}_j^{(1)}\right)+\sum_{i=2}^{N}\sum_{j=1}^{K}T(\mathrm{a}_j^{(i)})=O(K*N*T^*)$，其中$T^*=\mathrm{argmax}_{1\leqslant j\leqslant K,\,1\leqslant i\leqslant N}T(\mathrm{a}_j^{(i)})$. 令$p$为代理模型的预测开销. 由于在第$i$次扩增过程中，根据上一次扩增保留的最优的$K$个架构生成一批新的候选架

构. 因此,每次扩增过程中,待预测的架构总数为$O(i*K*M)$. 搜索阶段代理模型预测总开销为$\sum_{i=2}^{N}(O(i*K*M)*p)=O\left(N^2*K*M*p\right)$. 令$q$为代理模型的训练开销,在渐进式深度集成架构搜索过程中,代理模型共训练了N次,因此代理模型训练总开销$O(N*q)$. 综上述,总的计算复杂度为$O\left(K*N*T^*\right)+O\left(N^2*K*M*p\right)+O(N*q)$.

空间复杂度分析:搜索过程中的主要空间开销为深度集成模型空间开销以及代理模型空间开销. 在搜索过程中由于每个需要评估的深度集成模型其Cell的种类、数量以及Cell之间的连接方式都不相同,因此深度集成模型的空间开销也不相同. 令$C(\mathrm{a}_j^{(i)})$为第i次扩增时(即Cell的数量为i),评估第j个模型的空间开销. 根据架构评估的总数量,可得到模型评估的空间总开销为$\sum_{j=1}^{M}C\left(\mathrm{a}_j^{(1)}\right)+\sum_{i=2}^{N}\sum_{j=1}^{K}C(\mathrm{a}_j^{(i)})=O(K*N*C^*)$, 其中$C^*=\mathrm{argmax}_{1\leqslant j\leqslant K,\ 1\leqslant i\leqslant N}C(\mathrm{a}_j^{(i)})$. 另外,在搜索过程中,代理模型的空间开销是固定不变的,主要包括代理模型的Embedding表空间开销与网络层空间开销. 令d为Embedding维度,W为网络层参数量,则Embedding表的空间开销为$O(d*N)$,代理模型的空间开销是$O(d*N+W)$. 综上述,总的空间复杂度为$O\left(K*N*C^*\right)+O(d*N+W)$.

5.4 整体算法框架

算法1 PMPAS整体流程

1. **输入**:训练数据集D_{train}
2. 验证数据集D_{val}
3. 基学习器类型数量M
4. 最大的Cell数量N
5. 神经网络的训练轮次E
6. 集束搜索过程中保留的架构数量K
7. 评价指标f
8. **输出**:架构搜索过程中在验证集上表现最好的深度集成架构

1. $n\leftarrow 1$
2. /* 在训练数据集上训练所有Cell数量为1的深度集成架构*/
3. $T_1\leftarrow train(S_1,D_{\mathrm{train}})$
4. /* 获得T_1中的深度集成架构在验证集上的性能 */
5. $V_1\leftarrow validate\left(T_1,D_{\mathrm{val}},f\right)$
6. /* 初始化代理模型 */
7. $P\leftarrow train\_proxy(S_1,V_1,E)$
8. /*保存完整训练的深度集成架构及其对应性能 */
9. $history\leftarrow\{<t_i,v_i>|t_i\in T_1,v_i\in V_1,i\in[1,M]\}$
10. **FOR** $n\leftarrow 2$ to N do
11. $\tilde{S}_n\leftarrow grow\_cell(S_{n-1})$
12. /* 利用代理模型对进行架构性能预测 */
13. $\widehat{V_n}\leftarrow proxy\_predict(P,\tilde{S}_n)$
14. /*选取其中最好的K个架构 */
15. $S_n\leftarrow select\_topK(\tilde{S}_n,\ \widehat{V_n},K)$
16. /* 训练代理模型预测的最好的K个架构 */
17. $T_n\leftarrow train(S_n,D_{\mathrm{train}})$
18. /* 获得深度集成架构在验证集上的性能 */
19. $V_n\leftarrow validate(T_n,D_{\mathrm{val}},f)$
20. /*更新代理模型*/
21. $P\leftarrow update\_proxy(P,S_n,V_n,E)$
22. /* 更新history */
23. $history\leftarrow history\cup\{<t_i,v_i>|t_i\in T_n,v_i\in V_n,\ i\in[1,K]$
24. **END FOR**
25. /* 获得history中评价指标最高的深度集成架构*/
26. $S^*\leftarrow get\_best(history)$
27. return S^*

从算法1中可看出,在Cell数量为1时,代理模型还不能够给出任何的参考信息,需要根据基学习器类型数量M,完整地训练出M个深度集成架构,得到其在验证集上的表现,利用<深度集成架构,验证集性能>数据集对代理模型进行初始化. 之后整个算法主体是一个迭代的过程. 在每一次的迭代过程中,以当前已经完成构建的架构集合为基础,将集合里每个架构的Cell数量再扩展一个,这一步对于$\mathcal{A}_{\mathrm{CP}}$和$\mathcal{A}_{\mathrm{DAG}}$而言是不同的. 对于$\mathcal{A}_{\mathrm{CP}}$,只需从$M$种类型的Cell中随机采样一个即可,且新采样得到的Cell和前面的Cell没有任何的连接关系. 而对于$\mathcal{A}_{\mathrm{DAG}}$,新扩展的Cell不仅要从$M$种类型的Cell中随机采样特定类型的Cell,还需要确定新的Cell和已有Cell的连接方式.

完成上述扩展过程后,将会产生较多新架构,由于已经根据之前的架构及其真实表现对代理模型进行了训练,因此可以利用代理模型给出这些新架构的预测性能. 然后,按照降序排序从中得到表现最好的K个架构,将这K个架构进行完整的训练,得到对应的真实表现,这样就完成了一步迭代. 在迭代过程中,还会记录这些真实训练的架构和对应的表

现，将其放入history列表中.

在Cell的数量达到最大值N后，整个迭代过程终止，这时可根据history列表中保存的最优架构及其对应的表现，得到其中最优的深度集成架构.

6　实验评估与分析

6.1　实验设置

6.1.1　实验环境与数据集

本文的实验环境如下：操作系统是Red Hat 4.8.2，CPU型号是2 x Intel(R) Xeon(R) CPU E5-2620 v2@ 2.10GHz，6-Core 12-Thread. 内存大小为64 GB. 主要软件包及版本如下：Python3.6.5，scikit-learn 0.24.1，auto-sklearn 0.12.3，XGBoost 1.1.1，分布式计算框架Ray 1.2.0，Tensor Flow1.12，Open ML 0.10.2.

本文实验采用了来自OpenML中的公开数据集[50]. 对分类数据集，除gcforest[5]以及gcforest_cs[16]在论文中测试使用过的数据集adult、letter和yeast外，还加入了来自OpenML-CC18[51]里的公开数据集，总计23个数据集. 这些数据集样本数量在500到100 000之间，既有二分类问题，也有多分类问题. 对于回归数据集，采用来自OpenMLRegression 30[51]中的数据集. 数据集ID为数据集在OpenML中的唯一标识. 数据集的统计信息如附录A中表A.1及表A.2所示. 无论是分类还是回归问题，上述选用的数据集都具有通用性和广泛性，能够比较客观地反映各个方法之间的真实效果.

为了验证提出的深度集成架构搜索算法PMPAS的有效性，本文和如下方法展开性能对比：

(1)集成学习方法：包括常用的XGBoost、GBDT以及Random Forest，设置采用默认超参数；

(2)神经网络方法：多层感知机MLP，通过超参数调优选择在大部分数据集上均取得优异效果的超参数，即64-32的隐藏层结构；此外，还采用了文献[52]提出的处理表格型数据的神经网络方法ResNet和FT-Transformer.

(3)深度森林方法：原始深度森林(gcf)的实验环境设置与论文[5]中的配置保持一致，级联森林层包含4个随机森林和4个完全随机森林，每个森林都由500棵决策树构成，通过3折交叉验证产生每个森林的类别概率输出. 对于带置信度筛选的深度森林(gcf_cs)[16]，其基本结构和原始深度森林一致，同时参照论文中的置信度的阈值a的确定方式，根据第一层级联层的表现ϵ_1，若$\epsilon_1>90\%$，则设置$a=1/10$，否则设置$a=1/3$；

(4)自动化模型选择方法：为验证PMPAS的通用性，还加入了与目前开源的AutoML算法库auto-sklearn(简称AS)[53]的性能对比. AS每次搜索的是完整的机器学习流水线，包含了元学习、数据预处理、特征预处理、模型选择、模型集成等多个阶段. 为了使对比实验尽量公平，AS关闭了元学习的功能.

另外，PMPAS搜索空间支持的分类和回归基学习器类型如附录A中表A.3所示，共包含16种常见的分类基学习器以及13种常见的回归基学习器.

6.1.2　实验参数设置

对于PMPAS，需设置的主要参数如下.

(1)集束搜索参数K，实验中设置为8；

(2)代理模型参数，主要包括以下参数：

①神经网络训练的最大轮次(epoch)，实验中设置为8；

②神经网络训练的样本批大小(batch size)，实验中设置为256；

③神经网络的学习率，初始学习率为0.001，采用指数衰减方式，每500轮训练之后乘以0.98；

④神经网络正则化参数(regularization parameter)，实验中设置为0.0001；

⑤神经网络LSTM层的单元数量，实验中设置为64；

⑥神经网络嵌入维度，实验中设置为16.

PMPAS既可以根据时间预算(Time Budget)T，也可以根据搜索空间中深度集成架构的最大Cell数量N停止整个迭代过程. 为避免单个深度集成架构的评估时间过长影响整体搜索效率，实验中设置每个Cell的最大评估时间为120秒. 在实验效果评估指标上，对分类问题采用了分类准确率(Acc)作为评价指标，对回归问题采用决定系数(R2)作为评价指标. 最终实验结果按上述指标通过5折交叉验证产生.

6.2　分类任务对比实验

表1、表2、表3分别展示了PMPAS在时间预算分别为0.5 h(半小时)、1 h(一小时)和2 h(两小时)的情况下在分类任务上的对比实验结果，其中AS一列表示auto-sklearn运行同样时间预算的结果，PMPAS一栏的CP和DAG分别代表PMPAS在$\mathcal{A}_{\mathrm{CP}}$和$\mathcal{A}_{\mathrm{DAG}}$上的结果. 从表1、表2、表3可看出，

PMPAS-CP的整体效果好于PMPAS-DAG. 在时间预算为0.5 h的最终预测效果上，PMPAS-CP在OpenML-ID为22的数据集上相比XGBoost提高了10.5个百分点，在OpenML-ID为54的数据集上相比gcf提高了5.1个百分点，在OpenML-ID为18的数据集上相比MLP提升了57.5个百分点，在OpenML-ID为22的数据集上相比AS提升了6.1个百分点.

表1 PMPAS运行0.5 h的分类任务对比实验结果

ID	集成学习方法			深度森林方法		神经网络方法			AS	PMPAS	
	XGBoost	GBDT	RF	gcf	gcf_cs	MLP	ResNet	FT-Transformer		CP	DAG
6	0.9621	0.9176	0.9666	0.9740	0.9708	0.9232	0.8738	0.6542	0.9510	**0.9709**	0.9664
12	0.9670	0.9560	0.9680	0.9720	0.9720	0.7480	0.9845	0.4870	**0.9760**	0.9735	0.9740
14	0.8445	0.8235	0.8325	0.8445	0.8440	0.8330	0.8205	0.4975	0.8345	0.8435	**0.8455**
16	0.9515	0.9335	0.9645	0.9700	0.9690	0.9645	0.9625	0.9445	0.9765	0.9770	**0.9780**
18	0.6940	0.7075	0.7095	0.7845	0.7820	0.2130	0.4775	0.2345	0.7480	**0.7880**	0.7805
22	0.7860	0.7740	0.7780	0.8800	0.8815	0.7935	0.8164	0.7340	0.8300	0.8910	**0.8980**
23	0.5206	0.5519	0.5098	0.5024	0.5159	0.5424	0.4528	**0.5579**	0.5553	0.5526	0.5533
28	0.9769	0.9758	0.9820	0.9874	0.9872	0.9838	0.9809	0.9608	0.9868	**0.9890**	0.9875
32	0.9915	0.9885	0.9923	0.9932	0.9929	0.9917	0.9903	0.9417	0.9929	**0.9947**	0.9946
50	0.9843	0.9781	0.9906	0.9812	0.9916	0.9812	0.9801	0.8423	0.9885	0.9833	**0.9948**
54	0.7707	0.7683	0.7518	0.7790	0.7743	0.6678	0.6571	0.4148	**0.8392**	0.8298	0.8250
181	0.5930	0.6011	0.6280	**0.6345**	0.6092	0.5903	0.6030	0.5687	0.5991	0.6253	0.6011
182	0.9156	0.9026	0.9180	0.9260	0.9240	0.9093	0.8956	0.8875	0.9073	**0.9277**	0.9232
300	0.9481	0.9298	0.9475	0.9586	0.9581	0.9597	0.9533	0.9301	0.9604	0.9626	**0.9633**
458	0.9869	0.9869	0.9893	0.9929	0.9940	0.9869	0.9929	0.9786	**0.9964**	0.9940	0.9952
1462	0.9964	0.9956	0.9949	0.9985	0.9985	**1.0000**	**1.0000**	**1.0000**	0.9993	0.9978	0.9978
1501	0.9253	0.9278	0.9441	0.9466	0.9466	0.9190	0.8977	0.6447	0.9215	**0.9548**	0.9517
1590	**0.8733**	0.8665	0.8540	0.8640	0.8611	0.8172	0.7864	0.7868	0.8125	0.8650	0.8649
4534	0.9699	0.9515	0.9715	0.9717	**0.9718**	0.9693	0.9687	0.9253	0.9651	0.9708	0.9705
23381	0.5360	0.5980	0.5740	0.5700	0.5580	0.5880	0.5820	0.6000	0.6040	0.6000	**0.6320**
40499	0.9824	0.9789	0.9780	0.9882	0.9885	0.9935	**0.9984**	0.8414	0.9991	0.9911	0.9925
40670	**0.9655**	0.9601	0.9564	0.9614	0.9605	0.9479	0.9517	0.8981	0.9567	0.9623	0.9592
40983	0.9841	0.9837	0.9831	0.9849	0.9845	0.9707	0.9861	0.9557	**0.9882**	0.9835	0.9849

表4给出了对表1、表2、表3中信息汇总后的数据，其中的每一项代表PMPAS-CP或PMPAS-DAG随着时间预算增加，相对其他方法效果更优的数据集个数. PMPAS相比其他方法，在分类实验采用的23个数据集的大多数数据集上都取得了更优的效果.

从表4中可看出，随着时间预算不断增加，PMPAS相比集成学习方法、深度森林方法和MLP效果更优的数据集个数在不断增加，而PMPAS相对AS占优的数据集个数在不断减少. AS同样采用了SMBO方法，时间预算越长，其内部的贝叶斯优化模型效果越好，然而计算开销相对PMPAS采用的神经网络会更小，从而探索了更多的机器学习流水线，随着时间预算的不断增加，效果越来越好. 现实场景时间预算往往是非常宝贵的，在0.5 h的时候，PMPAS-CP相比AS在23个数据集的14个数据集上效果更优，PMPAS-DAG相比AS在23个数据集的16个数据集上效果更优，PMPAS在较短运行时间(1 h以内)的情况下较优于AS. 另外，AS包含数据预处理、特征预处理与模型选择等多个阶段. 相比之下，PMPAS仅包含深度集成架构搜索，实验结果也显示了深度集成架构本身的有效性.

图10展示了PMPAS在时间预算为0.5 h、1 h、2 h时，和其他方法在所有数据集上的平均排名对比结果. 平均排名参照AUTO-WEKA[54]和auto-sklearn[53]论文中的计算方法. 具体来说，若得到M个机器学习方法$\{\varepsilon_1,\varepsilon_2,\cdots,\varepsilon_M\}$在$N$个数据集$\{D_1,D_2,\cdots,D_N\}$上的交叉验证结果$\mathrm{CV}=\left(\mathrm{cv}_{i,j}\right)_{N\times M}$，根据CV按降序计算出$\varepsilon_i$在$D_j$上的排名矩阵$\mathrm{Rank}=\left(\mathrm{rank}_{i,j}\right)_{N\times M}$，那么$\varepsilon_i$的ave_rank即为：

表2　PMPAS运行1 h的分类任务对比实验结果

ID	集成学习方法			深度森林方法		神经网络方法			AS	PMPAS	
	XGBoost	GBDT	RF	gcf	gcf_cs	MLP	ResNet	FT-Transformer		CP	DAG
6	0.9621	0.9176	0.9666	0.9740	0.9708	0.9232	0.8738	0.6542	0.9607	**0.9764**	0.9718
12	0.9670	0.9560	0.9680	0.9720	0.9720	0.7480	**0.9845**	0.4870	0.9800	0.9730	0.9775
14	0.8445	0.8235	0.8325	0.8445	0.8440	0.8330	0.8205	0.4975	**0.8800**	0.8575	0.8380
16	0.9515	0.9335	0.9645	0.9700	0.9690	0.9645	0.9625	0.9445	**0.9775**	0.9765	0.9730
18	0.6940	0.7075	0.7095	0.7845	0.7820	0.2130	0.4775	0.2345	0.7425	**0.8000**	0.7785
22	0.7860	0.7740	0.7780	0.8800	0.8815	0.7935	0.8164	0.734	0.8500	0.8930	**0.8950**
23	0.5206	0.5519	0.5098	0.5024	0.5159	0.5424	0.4528	0.5579	**0.5864**	0.5458	0.5526
28	0.9769	0.9758	0.9820	0.9874	0.9872	0.9838	0.9809	0.9608	0.9868	**0.9906**	0.9875
32	0.9915	0.9885	0.9923	0.9932	0.9929	0.9917	0.9903	0.9417	**0.9950**	0.9943	0.9945
50	0.9843	0.9781	0.9906	0.9812	0.9916	0.9812	0.9801	0.8423	**1.0000**	0.9916	0.9802
54	0.7707	0.7683	0.7518	0.7790	0.7743	0.6678	0.6571	0.4148	**0.8470**	0.8238	0.8380
181	0.5930	0.6011	0.6280	**0.6345**	0.6092	0.5903	0.6030	0.5687	0.6262	0.6078	0.6038
182	0.9156	0.9026	0.9180	0.9260	0.9240	0.9093	0.8956	0.8875	0.9230	**0.9305**	0.9266
300	0.9481	0.9298	0.9475	0.9586	0.9581	0.9597	0.9533	0.9301	0.9500	0.9608	**0.9610**
458	0.9869	0.9869	0.9893	0.9929	**0.9940**	0.9869	0.9929	0.9786	**0.9940**	**0.9940**	**0.9940**
1462	0.9964	0.9956	0.9949	0.9985	0.9985	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9978	0.9971
1501	0.9253	0.9278	0.9441	0.9466	0.9466	0.9190	0.8977	0.6447	**0.9624**	0.9529	0.9460
1590	**0.8733**	0.8665	0.8540	0.8640	0.8611	0.8172	0.7864	0.7868	0.8710	0.8716	0.8672
4534	0.9699	0.9515	0.9715	0.9717	0.9718	0.9693	0.9687	0.9253	0.9696	**0.9727**	0.9711
23381	0.5360	0.5980	0.5740	0.5700	0.5580	0.5880	0.5820	0.6000	**0.6300**	0.5720	0.6120
40499	0.9824	0.9789	0.9780	0.9882	0.9885	0.9935	**0.9984**	0.8414	0.9981	0.9975	0.9953
40670	**0.9655**	0.9601	0.9564	0.9614	0.9605	0.9479	0.9517	0.8981	0.9529	0.9595	**0.9655**
40983	0.9841	0.9837	0.9831	0.9849	0.9845	0.9707	0.9861	0.9557	**0.9885**	0.9849	0.9857

表3　PMPAS运行2 h的分类任务对比实验结果

ID	集成学习方法			深度森林方法		神经网络方法			AS	PMPAS	
	XGBoost	GBDT	RF	gcf	gcf_cs	MLP	ResNet	FT-Transformer		CP	DAG
6	0.9621	0.9176	0.9666	0.9740	0.9708	0.9232	0.8738	0.6542	0.9622	0.9732	**0.9759**
12	0.9670	0.9560	0.9680	0.9720	0.9720	0.7480	**0.9845**	0.4870	0.9800	0.9725	0.9730
14	0.8445	0.8235	0.8325	0.8445	0.8440	0.8330	0.8205	0.4975	0.8580	**0.8615**	0.8550
16	0.9515	0.9335	0.9645	0.9700	0.9690	0.9645	0.9625	0.9445	**0.9800**	0.9790	0.9775
18	0.6940	0.7075	0.7095	0.7845	0.7820	0.2130	0.4775	0.2345	0.7495	**0.7950**	0.7685
22	0.7860	0.7740	0.7780	0.8800	0.8815	0.7935	0.8164	0.734	0.8365	**0.9015**	0.8955
23	0.5206	0.5519	0.5098	0.5024	0.5159	0.5424	0.4528	0.5579	**0.5661**	0.5614	0.5600
28	0.9769	0.9758	0.9820	0.9874	0.9872	0.9838	0.9809	0.9608	0.9879	**0.9884**	0.9877
32	0.9915	0.9885	0.9923	0.9932	0.9929	0.9917	0.9903	0.9417	0.9943	0.9942	**0.9946**
50	0.9843	0.9781	0.9906	0.9812	0.9916	0.9812	0.9801	0.8423	**1.0000**	0.9916	0.9937
54	0.7707	0.7683	0.7518	0.7790	0.7743	0.6678	0.6571	0.4148	0.8179	**0.8310**	0.8085
181	0.5930	0.6011	0.6280	**0.6345**	0.6092	0.5903	0.6030	0.5687	0.6267	0.6206	0.6159
182	0.9156	0.9026	0.9180	0.9260	0.9240	0.9093	0.8956	0.8875	0.9201	**0.9291**	0.9285
300	0.9481	0.9298	0.9475	0.9586	0.9581	0.9597	0.9533	0.9301	0.9609	**0.9624**	0.9619
458	0.9869	0.9869	0.9893	0.9929	0.9940	0.9869	0.9929	0.9786	**0.9964**	0.9952	0.9952
1462	0.9964	0.9956	0.9949	0.9985	0.9985	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9978	0.9978
1501	0.9253	0.9278	0.9441	0.9466	0.9466	0.9190	0.8977	0.6447	0.9523	0.9517	**0.9529**
1590	**0.8733**	0.8665	0.8540	0.8640	0.8611	0.8172	0.7864	0.7868	0.8534	0.8701	0.8697
4534	0.9699	0.9515	0.9715	0.9717	0.9718	0.9693	0.9687	0.9253	**0.9720**	0.9711	0.9715
23381	0.5360	0.5980	0.5740	0.5700	0.5580	0.5880	0.5820	0.6000	**0.6160**	0.6040	0.5940
40499	0.9824	0.9789	0.9780	0.9882	0.9885	0.9935	**0.9984**	0.8414	0.9982	0.9971	0.9967
40670	0.9655	0.9601	0.9564	0.9614	0.9605	0.9479	0.9517	0.8981	0.9605	0.9620	**0.9658**
40983	0.9841	0.9837	0.9831	0.9849	0.9845	0.9707	0.9861	0.9557	**0.9890**	0.9857	0.9853

表4 PMPAS分类任务对比实验结果汇总

方法	集成学习方法			深度森林方法		神经网络方法			AS
	XGBoost	GBDT	RF	gcf	gcf_cs	MLP	ResNet	FT-Transformer	
PMPAS-CP	18,21,21	21,20,23	20,21,21	17,20,19	18,20,21	21,21,22	19,18,19	20,21,22	14,10,10
PMPAS-DAG	21,20,22	21,23,22	20,20,21	16,15,19	16,16,20	21,21,22	18,19,19	21,21,21	16,9,9

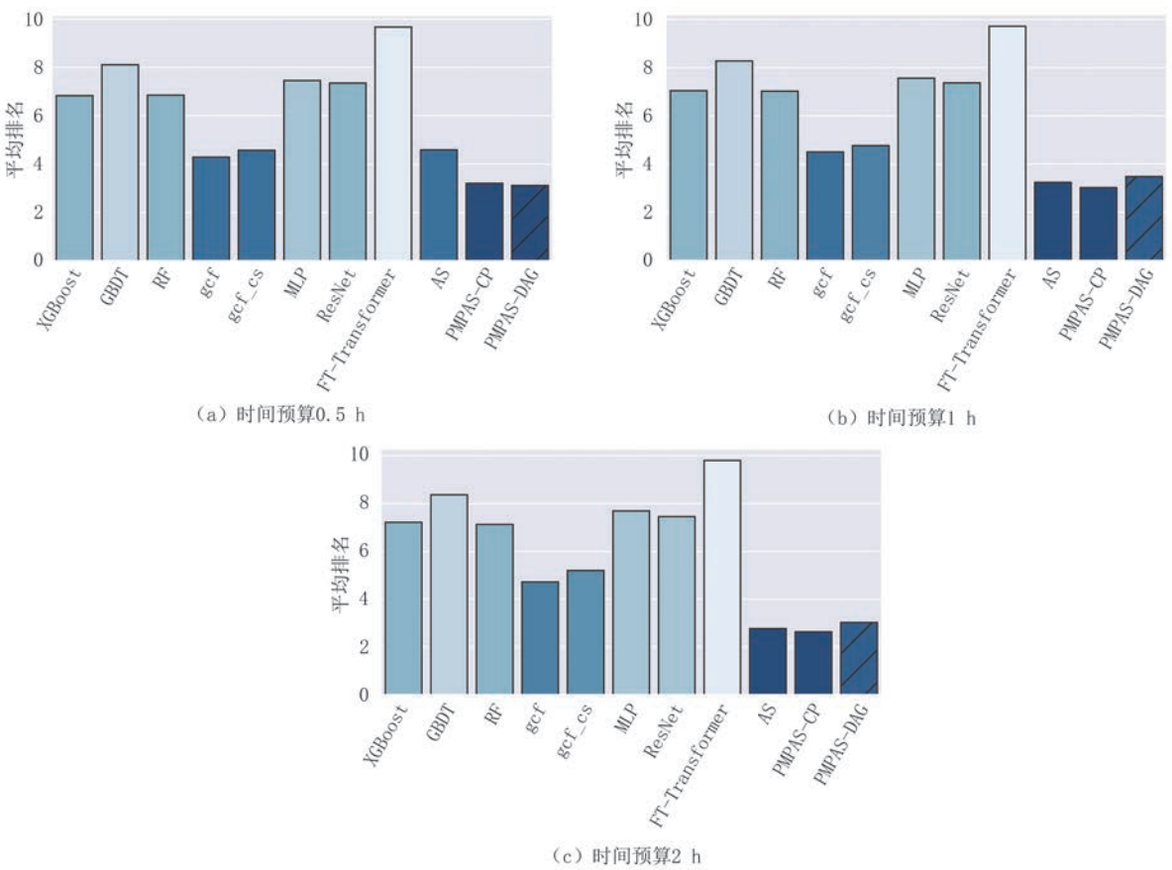

图10 PMPAS分类任务不同时间预算下平均排名对比结果

$$\text{ave\_rank}(\varepsilon_i)=\frac{\sum_{j=1}^{N}\text{rank}_{i,j}}{N}$$

ave_rank(ε_i)较为全面地反映了ε_i的整体效果. ave_rank(ε_i)$\in[1,M]$,平均排名越小说明ε_i的整体效果越好.

从图10中可以看到PMPAS相比于其他的方法效果明显. 在预算时间为0.5 h时,PMPAS-DAG的效果最好,并且几乎和PMPAS-CP效果相同,都明显优于其他方法;在预算时间为1h时,PMPAS-CP的效果最好,PMPAS效果优势依然明显;在预算时间为2 h时,PMPAS-CP的效果最好. 由于运行时间为0.5 h时PMPAS-CP和PMPAS-DAG两者评估的深度集成架构都不是非常多,PMPAS-CP和PMPAS-DAG的真正差异尚不明显,而在时间预算增加后,PMPAS在$\mathcal{A}_{CP}$中的效果更加明显,这也说明了分类任务更适合CP搜索空间.

综上所述,在分类任务上,PMPAS在大部分数据集上能够取得更为优异的性能,而且随着时间预算的不断增加,性能优势更为明显. 另外,基于完全并行的搜索空间更适合分类任务.

6.3 回归任务对比实验

由于深度森林方法gcf和gcf_cs均不支持回归任务,因此没有加入与gcf和gcf_cs的对比实验结果. FT-Transformer在所有数据集上R2分数均小于0,因此不进行对比. PMPAS完全可以运行在回归任务上(将R2预测作为增广特征),这说明相对于已有的深度集成学习方法,PMPAS的通用性比较好.

表5、表6、表7分别展示了PMPAS在运行0.5 h、1 h和2 h的情况下在回归任务上的对比实验结果. 其中'-'符号表示对应方法在数据集上得到的

R2分数小于0，说明此种结构不适合于回归任务，因而相应的分数未在表格中显示.

表8对表5、表6、表7的结果进行了汇总，其中的每一项代表PMPAS-CP或PMPAS-DAG在回归任务上对比其他方法效果更优的数据集个数.

图11展示了不同时间预算下各个方法在回归数据集上的平均排名. 可以看出，PMPAS在回归任务上的效果明显，说明了深度集成架构的优势. 另外可以发现在回归任务中$\mathcal{A}_{\mathrm{DAG}}$是较为优异的搜索空间. 主要原因是，在回归任务中，深度集成架构不再像分类任务一样给出类别概率作为增广特征，因而回归任务中深度集成架构的Layer信息不如分类

表5　PMPAS运行半小时回归任务对比实验结果

ID	集成学习方法			神经网络方法		AS	PMPAS	
	XGBoost	GBDT	RF	MLP	ResNet		CP	DAG
201	0.9851	0.9335	0.9876	**0.9949**	0.003	0.9003	0.9926	0.9927
216	**0.8818**	0.7909	0.8372	-	-	0.6026	**0.8818**	0.8749
287	0.4507	0.3865	0.5172	0.2434	-	0.4150	0.5220	**0.5403**
505	0.9913	0.9953	0.9905	0.9699	-	0.9947	**0.9968**	0.9967
507	0.6894	0.6466	0.6470	-	-	0.6357	**0.7233**	0.7158
574	0.6344	0.5790	0.6439	-	-	0.3296	**0.6576**	0.6344
41021	0.9258	0.9376	0.9265	0.8207	-	**0.9451**	0.9349	0.9332
42688	0.7032	0.6835	0.6421	0.9997	0.011	0.5116	**1.0000**	**1.0000**
42726	0.5061	0.5533	0.5494	**0.5663**	-	0.5452	0.5220	0.5442
42730	0.6195	**0.6637**	0.6564	-	0.0003	0.6585	0.6615	0.6610

表6　PMPAS运行一小时回归任务对比实验结果

ID	集成学习方法			神经网络方法		AS	PMPAS	
	XGBoost	GBDT	RF	MLP	ResNet		CP	DAG
201	0.9851	0.9335	0.9876	**0.9949**	0.003	0.9759	0.9930	0.9925
216	**0.8818**	0.7909	0.8372	-	-	0.8246	0.8770	0.8803
287	0.4507	0.3865	0.5172	0.2434	-	0.4707	0.5201	**0.5406**
505	0.9913	0.9953	0.9905	0.9699	-	0.9953	0.9968	**0.9972**
507	0.6894	0.6466	0.6470	-	-	0.7068	0.6730	**0.7098**
574	0.6344	0.5790	0.6439	-	-	0.5299	**0.6649**	0.6538
41021	0.9258	0.9376	0.9265	0.8207	-	**0.9462**	0.9315	0.9334
42688	0.7032	0.6835	0.6421	0.9997	0.011	0.9364	**1.0000**	**1.0000**
42726	0.5061	0.5533	0.5494	0.5663	-	**0.5716**	0.5496	0.5519
42730	0.6195	0.6637	0.6564	-	0.0003	0.6572	**0.6799**	0.6602

表7　PMPAS运行两小时回归任务对比实验结果

ID	集成学习方法			神经网络方法		AS	PMPAS	
	XGBoost	GBDT	RF	MLP	ResNet		CP	DAG
201	0.9851	0.9335	0.9876	**0.9949**	0.003	0.9811	0.9856	0.9927
216	**0.8818**	0.7909	0.8372	-	-	0.8301	0.8530	0.8798
287	0.4507	0.3865	0.5172	0.2434	-	0.4807	0.5031	**0.5293**
505	0.9913	0.9953	0.9905	0.9699	-	0.9959	0.9963	**0.9966**
507	0.6894	0.6466	0.6470	-	-	0.6732	0.6907	**0.6993**
574	0.6344	0.5790	0.6439	-	-	0.5794	0.6070	**0.6479**
41021	0.9258	0.9376	0.9265	0.8207	-	**0.9462**	0.9393	0.9338
42688	0.7032	0.6835	0.6421	0.9997	0.011	0.9364	0.9682	**1.0000**
42726	0.5061	0.5533	0.5494	**0.5663**	-	0.5636	0.5592	0.5545
42730	0.6195	**0.6637**	0.6564	-	0.0003	0.6520	0.6623	0.6577

表8 PMPAS回归任务统计信息

方法	集成学习方法			神经网络方法		AS
	XGBoost	GBDT	RF	MLP	ResNet	
PMPAS-CP	10,8,8	7,8,9	9,10,7	8,8,7	10,10,10	8,7,8
PMPAS-DAG	9,9,9	7,7,8	8,10,10	8,8,8	10,10,10	8,8,8

任务中的信息丰富.$\mathcal{A}_{\mathrm{DAG}}$中的深度集成架构结构更为复杂,各个Cell之间的关联性增强,每个Cell给出的R2预测作为增广特征相比原始的数据有了更多的信息,更有可能在回归数据集上表现更好.

综上所述,在回归任务上,PMPAS性能优于已有的集成学习、深度学习以及自动化机器学习方法.另外,基于DAG的搜索空间更适合回归任务.

6.4 代理模型效果分析

本节对PMPAS使用的神经网络代理模型效果进行分析.从OpenML随机选择4个分类数据集.图12展示了在分类任务下,时间预算为2 h,代理模型给出的架构在验证集上的准确率变化情况.图中横坐标表示搜索过程中需要评估的架构(以架构索引所示),纵坐标代表了其在验证集上的准确率,阴影部分表示平滑窗口为20的95%置信度区间.

从图12可看出,无论是PMPAS-CP还是PMPAS-DAG,开始的运行结果都不太稳定,这是由于刚开始时Cell数量为1,需要进行评估的Cell之间有着较为显著的差别,此时代理模型还没有起到搜索过程的指导作用.而随着Cell数量的不断增长,代理模型开始指导整个搜索过程的进行.从图中的验证集准确率变化曲线可以看到,一旦代理模型起作用,其整个收敛过程相对较快.

综上所述,本文引入的代理模型能够有效地预测架构的性能,从而指导和加速整个架构搜索过程.

6.5 渐进式增长策略评估

集束搜索过程中每次保留验证集上效果最好的K个架构.为了评估本文提出的渐进式增长策略,本节将搜索空间类型限制为$\mathcal{A}_{\mathrm{DAG}}$架构的最大Cell数量为8,并通过实验对比分析不同的保留前K个架构的策略.按照如何保留K个架构,依次设计了最优PMPAS-DAG、利用PMPAS-DAG、探索PMPAS-DAG和随机PMPAS-DAG共4种方法.最优PMPAS-DAG指保留最优的K个架构策略;利用PMPAS-DAG指保留80%的效果最优架构和20%的随机选取架构;探索PMPAS-DAG指保留20%效果最优架构和80%随机选取架构;随机

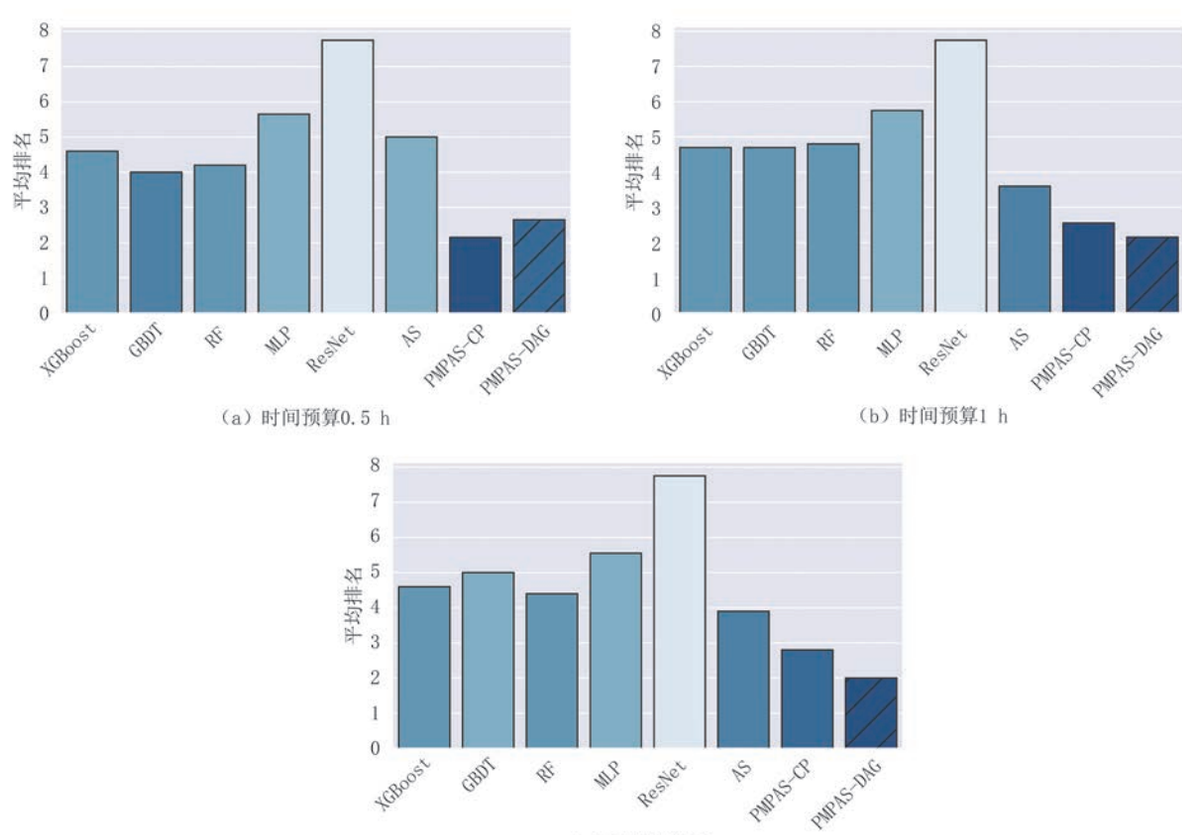

图11 PMPAS不同时间预算下回归任务平均排名对比结果

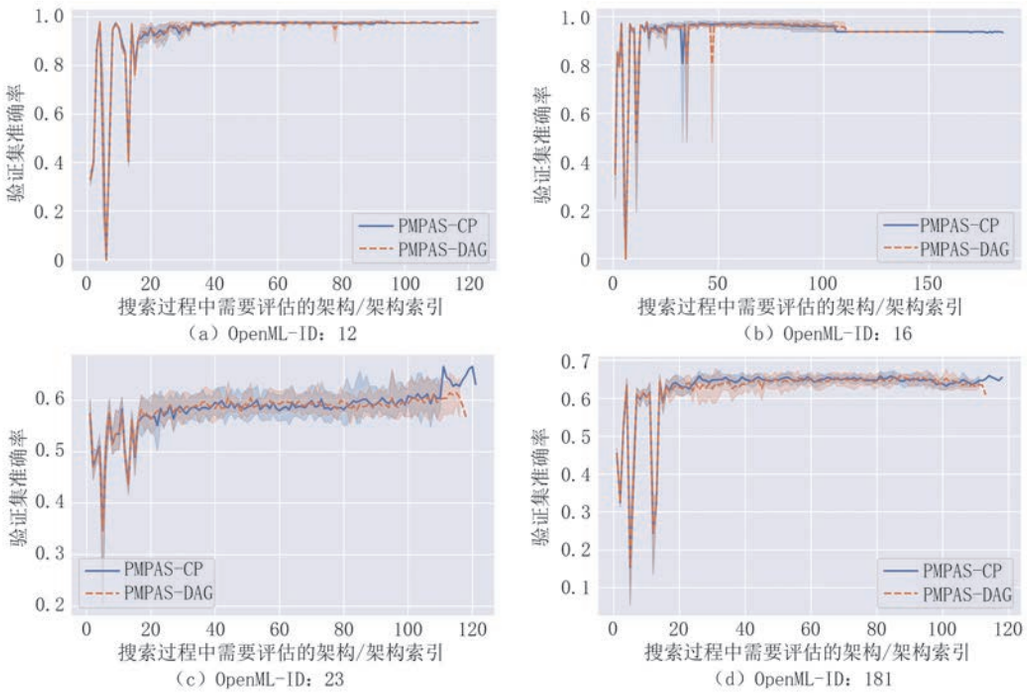

图12　代理模型效果分析

PMPAS-DAG指完全随机选取架构.

实验结果如图13所示,其中横坐标表示当前架构的Cell数量,纵坐标表示当前Cell数量下K个架构在所有数据集上的平均排名.从图13中可以看出,在Cell数量为1时,各个方法评估的架构集合相同,从而平均排名一致.而在最终的平均排名上来看,最优PMPAS-DAG有着较为明显的优势,随机PMPAS-DAG的效果最差.在代理模型给出的<架构,预测效果>数据集中,保留预测效果优秀的架构越多(利用PMPAS-DAG),则相应的完整训练和评估的架构效果也会更好.因此,随着Cell数量的增加,最优PMPAS-DAG和利用PMPAS-DAG效果不断接近.

综上所述,通过PMPAS-DAG渐进式增长策略评估实验结果,可以发现,在构建最终架构时,需要以优异的架构作为基础,优异的架构更可能是构建最优架构的基础.

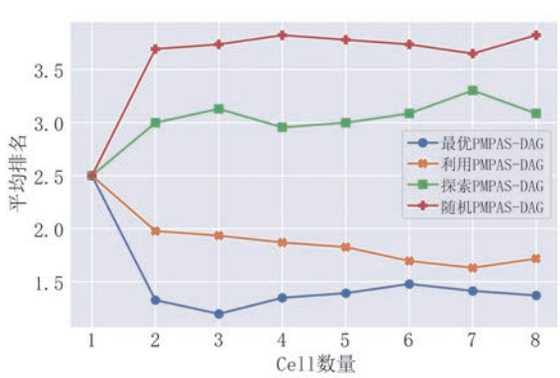

图13　PMPAS-DAG渐进式增长策略评估

6.6　搜索策略评估

为了验证本文提出的基于代理模型的渐进式深度集成架构搜索策略的有效性,本节和其他典型常用的搜索策略进行了性能对比.实验设定搜索空间为$\mathcal{A}_{\mathrm{DAG}}$,时间预算为10 h.

(1)基于进化算法的架构搜索:采用一种基于最优个体保留进化算法[55]的启发式搜索,首先设计深度集成架构的编码空间,然后基于进化算法在编码空间中寻找最优深度集成架构,该方法简称EPEAAS(Elitist Preserved Evolutionary Algorithm based Architecture Search);

(2)随机搜索:在预先定义的最大Cell数量的搜索空间中随机采样深度集成架构,简称RS.

6.6.1　长时间运行效果对比

图14展示了长时间运行时各个搜索方法平均排名的变化情况,其中横坐标代表运行时间,纵坐标代表各个搜索方法的平均排名,平均排名越小,代表该搜索方法的整体效果越好.

从图14可以看出,整个过程中RS-DAG的效果几乎总是最差,而EPEAAS-DAG和PMPAS-DAG两者在运行了较长时间后,平均排名较为接近,但在最终效果上EPEAAS-DAG会持平或略优于

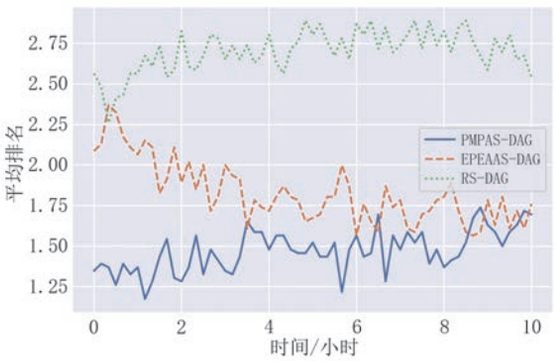

图14 自动化搜索方法长时间运行效果对比

PMPAS-DAG. 观察开始的一段时间区间可以发现，PMPAS-DAG在三者中的效果最好，它总是能够迅速给出更优的架构，这是因为EPEAAS进化算法迭代过程中较为耗时，而PMPAS通过引入代理模型，能够实现在较短的预算时间内给出更加优异的深度集成架构的目标. PMPAS-DAG在时间资源受限的场景中，能够有更加优异的效果，但若时间预算较为充足，则EPEAAS效果也比较优异.

另外，随机选择4个分类数据集，图15展示了三种自动化搜索方法运行过程中的验证集准确率变化对比结果.

从图15可以看出，在所有的数据集上PMPAS-DAG在开始的一小时到两小时内，都能够针对当前的数据集给出效果明显优于EPEAAS-DAG和RS-DAG的架构. 而在大致运行两小时后，EPEAAS-DAG的效果开始变得显著.

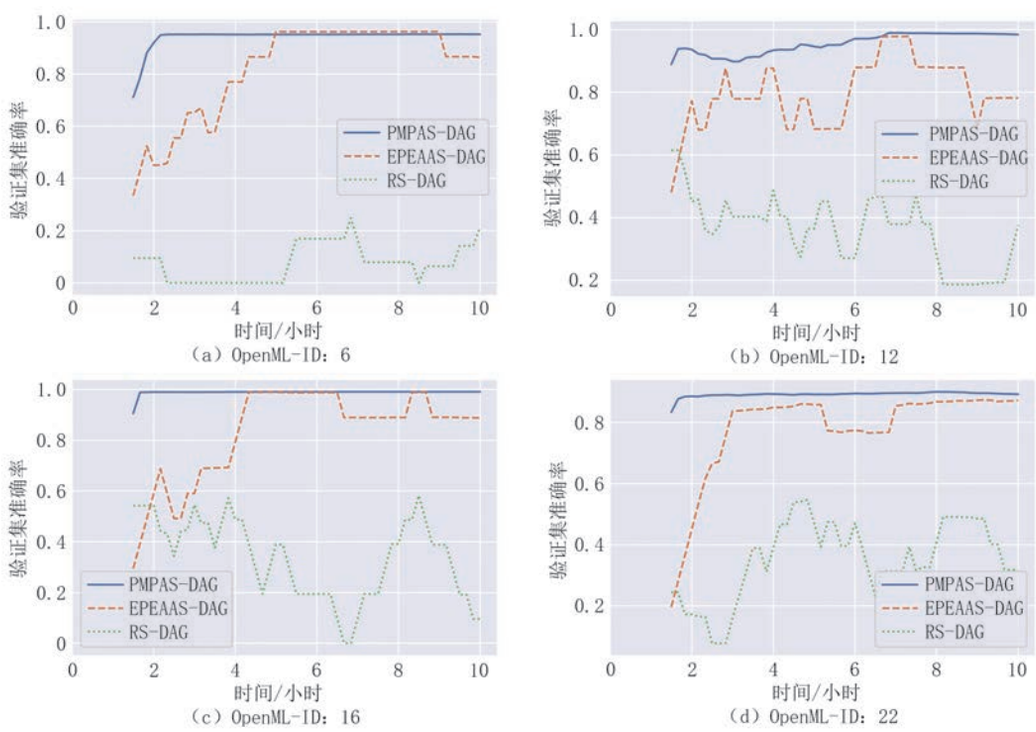

图15 自动化搜索方法运行过程效果对比

综上所述，随着时间预算的增加，PMPAS算法的准确率能够取得明显的提升. 当运行时间达到特定阈值后，准确率则趋于稳定.

6.6.2 评估次数对比

在自动化搜索过程中，对于每个架构进行完整的训练，并在验证集上获得当前架构的效果(如Acc或R2)称为一次评估. 图16展示了三种自动化搜索方法在搜索过程中平均评估次数的对比，其中横坐标代表运行时间，纵坐标代表每种方法在所有数据集上的平均评估次数.

从图16可以看出，在运行时间较短时(小于两小时)各个方法平均评估次数基本接近，但随着时间不断增加，PMPAS-DAG的平均评估次数不断减少，这是由于随着Cell最大数量的不断增加，架构变得更为复杂，因而用来训练代理模型的时间成本不断地增加，导致PMPAS-DAG实际进行评估的架构数量变少. 而EPEASAS-DAG在进化迭代过程中涉及到的额外计算开销相对而言较小，每次进化迭代产生的新种群相对上一代种群结构变化相对较小，从而导致了较多的平均评估次数. RS-DAG由

于完全随机地从搜索空间中进行采样，导致每次有很大概率得到的是较为复杂的结构，耗费在每个复杂架构上的评估时间大大增加，从而平均评估次数较少. 从三种搜索方法的平均评估次数对比中可以看出，EPEAAS对$\mathcal{A}_{\mathrm{DAG}}$探索的最彻底，PMPAS则最充分地利用了已有架构的信息，搜索过程的指导效果最为明显，而RS-DAG则完全随机采样，对于架构搜索没有任何指导作用.

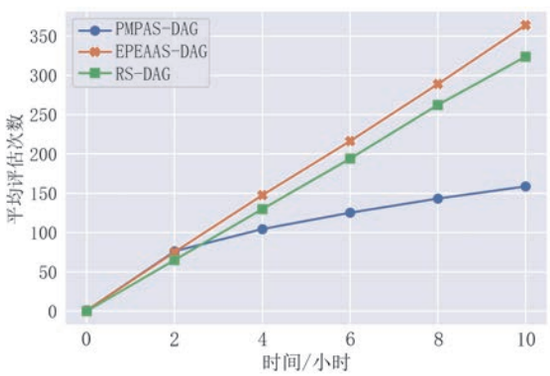

图16　自动化搜索方法平均评估次数对比

综上所述，随着时间预算的增加，PMPAS算法的代理模型训练成本也会逐渐增加，导致平均架构评估次数低于EPEAAS以及RS方法.

6.7　架构搜索空间分析

图17展示了不同搜索方法在分类任务下搜索到的最优架构Layer中基本机器学习算法的出现频率，其中横坐标各机器学习算法相应的缩写，纵坐标代表了相应算法出现的频数.

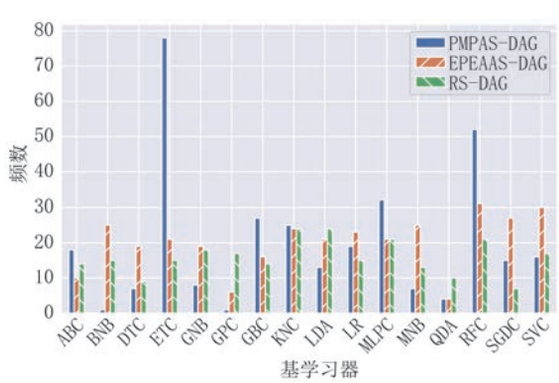

图17　分类任务中最优深度集成架构基本机器学习算法出现频率对比

从图17可以发现，随机森林(RFC)和完全随机森林(ETC)的出现次数最多，这两种集成算法都以决策树为基本进行构建，在实践中对表格型数据集具有较好的效果，作为深度集成学习优秀代表的深度森林较多地使用了RFC和ETC. 同时，从图中可以发现，总体效果较好的PMPAS-DAG搜索出来的最优架构中RFC和ETC的出现次数最多，而EPEAAS-DAG次之，效果最差的RS-DAG最少，说明对于深度集成架构，森林类的算法仍然是较好的Cell选择.

综上所述，如图17所示，除了随机森林等出现频率较高的基学习器之外，其他类型的机器学习算法也有相应的出现，说明搜索空间依然需要保证算法的多样性，算法的多样性仍然是深度集成架构搜索方法能够在不同的数据集下取得优异性能的重要方面.

7　总结与展望

近年来，以深度森林为代表的深度集成学习模型已引起了广泛关注. 然而，深度森林并不是深度集成学习的全部，它并非能够在所有数据集上取得最好的效果. 为了探索除深度森林之外的深度集成学习模型架构，本文研究提出了一种高效的基于代理模型的渐进式深度集成架构搜索方法，简称PMPAS. 据我们所知，PMPAS是第一个面向深度集成学习模型的架构搜索方法. 首先，通过归纳分析已有深度集成学习模型的特点，在深度集成架构形式化定义的基础上，研究提出了两种全新的深度集成架构搜索空间，即基于完全并行的搜索空间和基于有向无环图的搜索空间. 其次，研究提出了基于代理模型的渐进式搜索方法与算法，实现从简单到复杂逐步地在搜索空间中进行探索，并采用代理模型作为指导，从而降低模型评估开销. 在分类、回归等表格型数据集上的大量实验结果表明，通过PMPAS算法搜索得到的深度集成架构，其性能不仅优于已有的集成学习方法以及以深度森林为代表的深度集成学习模型，而且优于已有的自动化机器学习算法.

未来工作中，将进一步扩展和优化深度集成架构搜索方法，使其能够支持图像、语音以及序列数据的架构搜索. 此外，将研究分布式的深度集成架构搜索方法，使其能够支持更大规模的数据集. 另外，也将探索支持Boosting集成方式以及Boosting与Cascade混合集成方式的深度集成架构搜索空间，并且在架构搜索过程中融入超参数优化，进一步提升深度集成学习模型的性能.

参 考 文 献

[1] Krizhevsky A, Sutskever I, and Hinton G E. ImageNet classification with deep convolutional neural networks. Communications of the ACM, 2017, 60(6):84-90

[2] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks//Proceedings of the 27th International Conference on Neural Information Processing Systems. Montreal, Canada, 2014: 3104-3112

[3] Hinton G, Deng L, Yu D, et al. Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Processing Magazine, 2012, 29(6):82-97

[4] Zhou Z H. Deep Learning: Why deep and is it only doable for neural networks. https://www.ijcai19.org/invited-talks.html

[5] Zhou Z H, Feng J. Deep forest: Towards an alternative to deep neural networks//Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. Melbourne, Australia, 2017: 3553-3559

[6] Zhou Z H, Feng J. Deep forest. National Science Review, 2019, 6(1), 74-86

[7] Yang L, Wu X Z, Jiang Y, et al. Multi-label learning with deep forest//Proceedings of the 24th European Conference on Artificial Intelligence. Santiago de Compostela, Spain, 2020: 1634-1641

[8] Wang Q W, Yang L, Li Y F. Learning from weak-label data: adeep forest expedition//Proceedings of the AAAI Conference on Artificial Intelligence.New York, USA, 2020:6251-6258

[9] Zhang Y L, Zhou J, Zheng W, et al. Distributed deep forest and its application to automatic detection of cash-out fraud. ACM Transactions on Intelligent Systems and Technology, 2019, 10 (5): 1-19

[10] Wang W, Dai Q Y, Li F, et al. MLCDForest: multi-labels classification with deep forest in disease prediction for long non-coding RNAs. Briefings in Bioinformatics, 2021, 22 (3) : bbaa104

[11] Su R, Liu X, Wei L, et al. Deep-Resp-Forest: A deep forest model to predict anti-cancer drug response. Methods, 2019, 166: 91-102

[12] Wang T, Wang S, Zhou Z H. Machine learning for 5G and beyond: From model-based to data-driven mobile wireless networks. China Communications, 2019, 16(1): 165-175

[13] Yu C Y, Liu H, Qi Z M. Sound event detection using deep random forest. Detection and Classification of Acoustic Scenes and Events. Munich: IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events, 2017

[14] Zhang C, Yan J, Li C, et al. Contour detection via stacking random forest learning. Neurocomputing, 2018, 275: 2702-2715

[15] Utkin L V, Ryabinin M A. A Siamese deep forest. Knowledge-Based Systems, 2018, 139: 13-22

[16] Pang M, TingK M, Zhao P, et al. Improving Deep Forest by Screening. IEEE Transactions on Knowledge and Data Engineering, 2020, 39(4):4298-4312

[17] Ma P, Wu Y, Li Y, et al. DBC-Forest: Deep forest with binning confidence screening. Neurocomputing, 2022, 475: 112-122

[18] Ma P, Wu Y, Li Y, et al. HW-Forest: Deep Forest with Hashing Screening and Window Screening. ACM Transactions on Knowledge Discovery from Data, 2022, 16(6):1-24

[19] MengZiyao, Gu Xue, Liang Yanchun, et al. Deep neural architecture search: a survey. Journal of Computer Research and Development, 2021, 58(1): 22-33
(孟子尧, 谷雪, 梁艳春等. 深度神经架构搜索综述. 计算机研究与发展, 2021, 58(1): 22-33)

[20] Li Hangyu, Wang Nannan, Zhu Mingrui, et al. Recent Advances in Neural Architecture Search: A Survey. Journal of Software, 2022, 33(1): 129-149.
(李航宇,王楠楠,朱明瑞等. 神经结构搜索的研究进展综述. 软件学报, 2022, 33(1): 129-149)

[21] Elsken T, Metzen J H, Hutter F. Neural architecture search: A survey. Journal of Machine Learning Research., 2019, 20(55): 1-21.

[22] Freund Y, Schapire R E. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 1997, 55(1):119-139.

[23] Efron B, Tibshirani R J. An introduction to the bootstrap. New York,USA: CRC Press, 1994

[24] Breiman L. Random Forests. Machine Learning, 2001, 45:5-32

[25] Breiman L. Stacked regressions. Machine Learning, 1996, 24 (1):49-64

[26] Gama J, Brazdil P. Cascade generalization. Machine learning, 2000, 41(3): 315-343

[27] Feng J, Yu Y, Zhou Z H. Multi-Layered Gradient Boosting Decision Trees//Proceedings of the 32nd International Conference on Neural Information Processing Systems. Montréal, Canada, 2018: 3555-3565

[28] Zhu G, Hu Q, Gu R, et al. ForestLayer: Efficient training of deep forests on distributed task-parallel platforms. Journal of Parallel and Distributed Computing, 2019, 132 (OCT.) : 113-126

[29] Chen Z, Wang T, Cai H, et al. BLB-gcForest: A high-performance distributed deep forest with adaptive sub-forest splitting. IEEE Transactions on Parallel and Distributed Systems. 2022, 33(11): 3141-3152

[30] Yao Q, Wang M, Chen Y, et al. Taking human out of learning applications: A survey on automated machine learning. arXiv preprint arXiv:1810.13306, 2018

[31] He X, Zhao K, Chu X. AutoML: A Survey of the State-of-the-Art. Knowledge-Based Systems, 2021, 212: 106622

[32] Zoph B, Vasudevan V, Shlens J, et al. Learning transferable architectures for scalable image recognition//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Salt Lake City，USA，2018:8697-871

[33] Zoph B，Le Q V. Neural architecture search with reinforcement learning//Proceedings of the 5th International Conference on Learning Representations. Toulon，France，2017：1-16

[34] Pham H，Guan M Y，Zoph B，et al. Efficient neural architecture search via parameter sharing//Proceedings of the 35th International Conference on Machine Learning. Stockholm，Sweden，2018:4095-4104

[35] Real E，Aggarwal A，Huang Y，et al. Regularized evolution for image classifier architecture search//Proceedings of the AAAI Conference on Artificial Intelligence. Hawaii，USA，2019：4780-4789

[36] Real E，Moore S，Selle A，et al. Large-scale evolution of image classifiers//Proceedings of the 34th International Conference on Machine Learning. Sydney，Australia，2017：2902-2911

[37] Liu C，Zoph B，Neumann M，et al. Progressive neural architecture search//Proceedings of the European Conference on Computer Vision.Munich，Germany，2018:19-34

[38] Jin H，Song Q，Hu X. Auto-Keras：An efficient neural architecture search system//Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Alaska，USA，2019:1946-1956

[39] Liu C，Zoph B，Neumann M，et al. Progressive neural architecture search//Proceedings of the European Conference on Computer Vision. Munich，Germany，2018：19-34

[40] Perez-Rua J M，Baccouche M，Pateux S. Efficient progressive neural architecture search. arXiv preprint arXiv：1808.00391，2018

[41] Brock A，Lim T，Ritchie J M，et al. SMASH：One-shot model architecture search through hypernetworks//Proceedings of the6th International Conference on Learning Representations. Vancouver，Canada，2018：1-22

[42] Bender G，Kindermans P，Zoph B，et al. Understanding and simplifying one-shot architecture search//Proceedings of the 34th International Conference on Machine Learning. Stockholm，Sweden，2018：549-558

[43] Liu H，Simonyan K，Yang Y. Darts：Differentiable architecture search//Proceedings of the 7th International Conference on Learning Representations. New Orleans，USA，2019：1-13

[44] Liang H，Zhang S，Sun J，et al. DARTS+：Improved differentiable architecture search with early stopping. arXiv preprint arXiv:1909.06035，2019

[45] Chu X，Zhou T，Zhang B，et al. Fair DARTS：Eliminating unfair advantages in differentiable architecture search//Proceedings of the 16th European Conference on Computer Vision.Glasgow，UK，2020：465-480

[46] Xu Y，Xie L，Zhang X，et al. PC-DARTS：Partial channel connections for memory-efficient differentiable architecture search//Proceedings of the 7th International Conference on Learning Representations，New Orleans，USA，2019：1-13

[47] Hutter F，Hoos H H，Leyton-Brown K. Sequential model-based optimization for general algorithm configuration//Proceedings of the 5th International Conference on Learning and Intelligent Optimization. Berlin，Germany：Springer，2011：507-523

[48] Lipton Z C，Berkowitz J，Elkan C. A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv：1506.00019，2015

[49] Potdar K，Pardawala T S，Pai C D. A comparative study of categorical variable encoding techniques for neural network classifiers. International Journal of Computer Applications，2017，175(4)：7-9

[50] Vanschoren J，Van Rijn J N，Bischl B，et al. OpenML：networked science in machine learning. ACM SIGKDD Explorations Newsletter，2014，15(2)：49-60

[51] Bischl B，Casalicchio G，Feurer M，et al. OpenML benchmarking suites//Proceedings of the International Conference on Neural Information Processing Systems. 2021

[52] Gorishniy Y，Rubachev I，Khrulkov V，et al. Revisiting deep learning models for tabular data//Proceedings of the International Conference on Neural Information Processing Systems. 2021：18932-18943

[53] Feurer M，Klein A，Eggensperger K，et al. Efficient and robust automated machine learning//Proceedings of the 28th International Conference on Neural Information Processing Systems.，Montreal，Canada，2015:2944-2952

[54] Thornton C，Hutter F，Hoos H H，et al. Auto-WEKA：Combined selection and hyperparameter optimization of classification algorithms//Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York，USA，2013：847-855

[55] Dejong K. Analysis of the behavior of a class of genetic adaptive systems[Ph.D. Thesis]. University of Michigan，USA，1975

附录A.

本文实验采用的数据集包括分类和回归数据集. 对于分类数据集，除了adult、letter和yeast等常见的分类数据集外，还加入了来自OpenML-CC18里的公开数据集，总计23个分类数据集，既有二分类问题，也有多分类问题. 对于回归数据集，采用来自OpenML Regression中的数据集，总计10个回归数据集. 数据集ID为数据集在OpenML中的唯一标识. 数据集的统计信息如表A.1及表A.2所示.

本文实验所采用的深度集成架构搜索空间中，共包含16种常见的分类基学习器以及13种常见的回归基学习器. 搜索空间支持的基学习器及其对应的缩写如表A.3所示.

表A.1 分类数据集统计表

id	数据集名称	特征数	类别数	样本数
6	letter	17	26	20000
12	mfeat-factors	217	10	2000
14	mfeat-fourier	77	10	2000
16	mfeat-karhunen	65	10	2000
18	mfeat-morphologica	7	10	2000
22	mfeat-zernike	48	10	2000
23	cmc	10	3	1473
28	optdigits	65	10	5620
32	pendigits	17	10	10992
50	tic-tac-toe	10	2	958
54	vehicle	19	4	846
181	yeast	10	9	1484
182	satimage	37	6	6430
300	isolet	618	26	7797
458	analcatdata_authorship	71	4	841
1462	banknote-authentication	5	2	1372
1501	semeion	257	10	1593
1590	adult	15	2	48842
4534	phishingwebsites	31	2	11055
23381	dresses-sales	13	2	500
40499	texture	41	11	5500
40670	dna	181	3	3186
40983	wilt	6	2	4839

表A.2 回归数据集统计表

id	数据集名称	特征数	样本数
201	pol	49	15000
216	elevators	19	16599
287	wine_quality	12	6497
505	tecator	125	240
507	space_ga	7	3107
574	house_16H	17	22784
41021	moneyball	15	1232
42688	brazilian_houses	13	10692
42726	abalone	9	4177
42730	us_crime	127	1994

表A.3 搜索空间支持的基学习器及其对应的缩写

分类算法名称	缩写	回归算法名称	缩写
AdaBoostClassifier	ABC	AdaBoostRegressor	ABR
BernoulliNB	BNB	BaggingRegressor	BR
DecisionTreeClassifier	DTC	DecisionTree Regressor	DTR
ExtraTreesClassifier	ETC	ExtraTreesRegressor	ETR
GaussianNB	GNB	GradientBoosting Regressor	GBR
GaussianProcess Classifier	GPC	KNeighborsRegressor	KNR
GradientBoosting Classifier	GBC	LinearSVR	LSVR
KNeighborsClassifier	KNC	MLPRegressor	MLPR
LinearDiscriminant Analysis	LDA	NuSVR	NSVR
LogisticRegression	LR	RandomForest Regressor	RFR
MLPClassifier	MLPC	Ridge	RR
MultinomialNB	MNB	XGBRegressor	XGBR
QuadraticDiscriminantAnalysis	QDA	XGBRFRegressor	XGBFR
RandomForestClassifier	RFC		
SGDClassifier	SGDC		
SVC	SVC		

ZHU Guang-Hui, Ph. D., assistant researcher, his research interests include big data intelligent analysis and automated machine learning.

QI Jia-Hao, M. S., his research interests include ensemble learning and automated machine learning.

ZHU Zhen-Nan, M. S. candidate, his research interests include data mining and graph neural network.

YUAN Chun-Feng, M. S., professor, her research interests include computer architecture and parallel computing.

HUANG Yi-Hua, Ph. D., professor, his research interests include big data, cloud computing, and parallel computing.

Background

Deep learning models do not only contain deep neural networks. In recent years, due to the features such as no need for back-propagation training, lower computational overhead, support for adaptive determination of model complexity, and excellent performance in tabular data modeling tasks, the deep ensemble learning models represented by deep forest have attracted a lot of attention in the academia and industry. However, the existing deep ensemble models are mainly based on deep forests. It is necessary to explore deep ensemble learning model architectures other than deep forests. Additionally, for a given dataset, designing a well-performing deep ensemble model requires extensive expert experience. Therefore, an efficient data-adaptive deep ensemble learning architecture design method is also needed. Inspired by neural architecture search (NAS), this paper proposes an efficient proxy model-based progressive deep ensemble architecture search method PMPAS (Proxy Model-based Progressive Architecture Search) from the perspectives of search space and search algorithm. First, by summarizing and analyzing the characteristics of existing deep ensemble learning models, a deep ensemble architecture and a formal definition of deep ensemblearchitecture search are given. Then, two new search spaces for deep ensemble architectures, namely a fully parallel search space and a directed acyclic graph-based search space, are proposed. On the basis of the above two search spaces, the study proposes a progressive search method and algorithm based on the surrogate model to realize the exploration in the search space from simple to complex step by step, and uses the surrogate model as a guide to reduce the model evaluation overhead. Extensive experimental results on tabular datasets show that PMPAS can not only outperform the existing ensemble learning models, deep learning models, deep ensemble learning models represented by deep forests, but also outperform existing automated model selection algorithms. To the best of our knowledge, PMPAS is the first search method for deep ensemble architectures.

This work was supported by the National Natural Science Foundation of China (No. 62102177), the Natural Science Foundation of Jiangsu Province (No. BK20210181), the Key R&D Program of Jiangsu Province (No. BE2021729), and the Collaborative Innovation Center of Novel Software Technology andIndustrialization, Jiangsu, China.