



华南理工大学
South China University of Technology

硕士学位论文

Stacking 算法的研究及改进

作者姓名	徐慧丽
学科专业	计算数学
指导教师	凌卫新 副教授
所在学院	数学学院
论文提交日期	2018 年 4 月

The Study and Improvement of Stacking

A Dissertation Submitted for the Degree of Master

Candidate: Xu Huili

Supervisor: Prof. Ling Weixin

South China University of Technology

Guangzhou, China

分类号:TP181

学校代号:10561

学 号: 201520121743

华南理工大学硕士学位论文

Stacking 算法的研究及改进

作者姓名: 徐慧丽

指导教师姓名、职称: 凌卫新副教授

申请学位级别: 理学硕士

学科专业名称: 计算数学

研究方向: 计算智能与信息处理

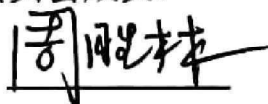
论文提交日期: 2018 年 4 月 20 日

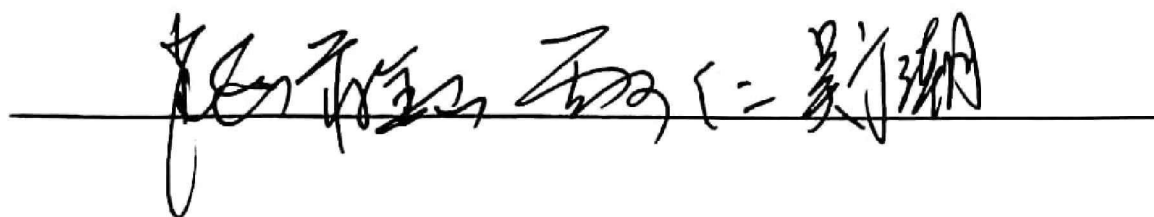
论文答辩日期: 2018 年 6 月 2 日

学位授予单位: 华南理工大学

学位授予日期: 年 月 日

答辩委员会成员:

主席: 

委员: 

华南理工大学

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：徐慧丽

日期：2018年6月2日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属华南理工大学。学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许学位论文被查阅(除在保密期内的保密论文外)；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。本人电子文档的内容和纸质论文的内容相一致。

本学位论文属于：

☐ 保密，(校保密委员会审定为涉密学位时间：____年____月____日) 于____年____月____日解密后适用本授权书。

☒ 不保密，同意在校园网上发布，供校内师生和与学校有共享协议的单位浏览；同意将本人学位论文提交中国学术期刊(光盘版)电子杂志社全文出版和编入 CNKI《中国知识资源总库》，传播学位论文的全部或部分内容。

(请在以上相应方框内打“√”)

作者签名：徐慧丽

指导教师签名：李发

作者联系电话：

联系地址(含邮编)：

日期：2018年6月2号

日期：2018年6月2号

电子邮箱：

摘要

分类问题是数据挖掘领域中常见问题之一。随着传统的数据挖掘分类技术（如逻辑回归、决策树等算法）的不断研究和发展，它们在分类问题上取得了越来越好的效果。但是，这种传统算法在计算上容易陷入过拟合。

集成学习可以有效缓解传统的单一分类算法遇到的过拟合问题。Stacking 算法是一种特殊的集成方法，它通过结合不同个体学习器的预测结果产生元层学习器，之后对数据进行预测。当训练数据很多时，Stacking 算法是一种较强的集成方法。Stacking 算法至少有两层学习器，因此 Stacking 算法具有较长的计算时间。

本文针对 Stacking 算法计算时间较长和样本数据较少的问题，采用类概率输出和多响应线性回归的概念，提出改进的 Stacking 算法。具体工作内容如下：

(1) 提出三层的 Stacking 算法结构。第一层训练原始数据集；第二层采用一种新的输入属性表示以增加第二层的训练数据，同时使得第二层个体分类器的输入属性大小不会随着分类器的增加而增大；第二层用个体分类器对前一层的的学习结果再次学习，以减弱第一层个体分类器输出类概率中的噪声。在预测时加入投票策略得到样本的类别。为了智能的选择改进的 Stacking 算法中的个体分类器，用遗传算法优化改进的 Stacking 算法，对算法的个体分类器组合进行优化。

(2) 在多个 UCI 数据集和 ORL 图像数据集上进行实验，并比较改进算法与其它算法在准确率，查准率，F1 值和运行时间上的结果。结果表明该算法在准确率、查准率和 F1 值上都显著优于其他集成方法。在运行时间上比传统的基于概率分布和多响应线性回归的 Stacking 算法在大部分数据上都有所降低。同时遗传算法优化得到的分类器的预测效果和人工调参得到的分类器的预测效果相当，这说明在优化改进的 Stacking 算法中可以用遗传算法代替人工调参。

关键词：集成学习；Stacking 算法；多响应线性回归；遗传算法

Abstract

The classification problem is one of the common problems in the field of data mining. With the continuous research and development of traditional data mining classification techniques (eg, Logistic Regression, Decision Trees, etc.), the performance of them is getting better and better in classification problems. However, this traditional algorithm is computationally easy to fall into over fitting.

Integrated learning can effectively alleviate the over fitting problems encountered by traditional single classification algorithms. The Stacking algorithm is a special integration method that generates a meta-layer learner by combining the prediction results of different individual learners. When there is a lot of training data, Stacking algorithm is a strong integration method. The Stacking algorithm has at least two layers of learners, therefore the Stacking algorithm has a high computational cost.

This paper proposes an improved Stacking algorithm based on the concept of class-probability output and multiple-response linear regression to decrease the computing time of Stacking algorithm and to solve the problem of less sample data. The specific work is as follows:

(1) Present a three-layers Stacking algorithm structure. Individual classifiers of the first layer are trained by the original data set; The second layer is represented by a new input attribute to increase the second level of training data, while making the input attribute of individual classifier in the second layer not increase with the increase of the classifier; The second layer uses some individual classifiers to relearn the learning results of the previous layer to reduce the noise in the output probability of the individual classifiers in the first layer. Add a voting strategy at the prediction stage to get the sample' category. In order to intelligently select the individual classifier in the improved Stacking algorithm, the improved Stacking algorithm is optimized by a genetic algorithm, and the combination of individual classifier is optimized.

(2) Compare the results of the improved algorithm with other algorithms on multiple UCI data sets and ORL image dataset in accuracy, precision, F1, and run time. The results show that the algorithm is superior to other integration methods in accuracy, precision and F1. Compare with the traditional Stacking algorithm which bases on probability distribution and multi-response linear regression, the running time is reduced in most of the data set. At the same time, the performance of the classifier optimized by the genetic algorithm is fairly equal to that of the classifier obtained by manual adjustment. This shows that the genetic algorithm can replace the manual adjustment in the optimization of the improved Stacking algorithm.

Keywords: ensemble learning; Stacking algorithm; multiple response linear regression; genetic algorithm

目 录

摘 要.....	I
Abstract.....	II
目 录.....	IV
第一章 绪论.....	1
1.1 课题背景及意义.....	1
1.2 国内外研究现状.....	2
1.3 本文主要的研究工作.....	4
1.4 论文的结构.....	5
第二章 集成学习及分类.....	7
2.1 集成学习概述.....	7
2.1.1 Boosting 算法.....	7
2.1.2 Bagging 算法.....	8
2.1.3 偏差与方差.....	9
2.2 结合策略.....	9
2.3 模型评估.....	12
2.3.1 评估方法.....	13
2.3.2 性能度量.....	14
2.4 本章小结.....	16
第三章 Stacking 算法.....	17
3.1 Stacking 算法介绍.....	17
3.2 Stacking 算法框架.....	17
3.3 元层学习器的属性表示和元层学习算法的研究.....	21
3.4 本章小结.....	22
第四章 Stacking 算法的改进.....	23
4.1 改进算法.....	23
4.1.1 2-层和 3-层个体学习器的输入属性表示.....	23

4.1.2 基分类器的选择	26
4.1.3 结合方法	26
4.1.4 改进的 Stacking 算法	27
4.2 实验设置	33
4.2.1 实验数据集及参数设置	33
4.2.2 实验结果比较及分析	34
4.3 改进的 Stacking 算法的人脸识别	40
4.3.1 数据描述及参数设置	40
4.3.2 实验结果与分析	41
4.4 遗传算法优化结构	42
4.4.1 遗传算法优化结构	42
4.4.2 实验结果及分析	43
4.5 本章小结	43
总结与展望	45
参考文献	46
附 录	51
攻读硕士学位期间取得的研究成果	52
致 谢	53

主要符号表

符号	描述
D	数据集
x	标量
\mathbf{x}	向量
$ \mathbf{x} $	向量 \mathbf{x} 的维数
c_j	类标签
$h_i^{(j)}$	分类器
$h_i^j(\mathbf{x})$	分类器 h_i 预测向量 \mathbf{x} 属于 c_j 类的类概率向量
$\{\cdots\}$	集合
$(:, :, \cdot)$	行向量
$\ \cdot\ _p$	L_p 范数
$A \setminus B$	所有属于数据集 A，而不属于数据集 B 的数据

第一章 绪论

1.1 课题背景及意义

在数据挖掘领域中，分类是一项重要的技术。比较常用的分类学习算法包括逻辑回归^[1]、决策树^[2]、神经网络^[3]、朴素贝叶斯^[4]、支持向量机等^[5]。通常情况下，我们可以训练多个学习算法，根据一定的评价标准，选择泛化性能最好的分类器，并调节最好的参数称之为模型选择。随着信息化时代的到来，如何从数据中构造强泛化性能模型吸引越来越多的研究者。

集成方法是机器学习的主要研究方向之一^[6,7]，和一般的通过训练集只构造一个学习器的传统学习方法不同，集成学习试图构造分类器的集合，并用一定的方式结合集合中个体分类器的输出。由于数据分布的复杂度和传统学习算法学习偏好，每种传统学习算法的适应范围比较有限，且泛化性能比较低。集成方法通过融合不同学习器的准确性和差异，来提高集成的泛化能力。目前在实际任务中集成仍是比较常用的机器学习方法之一。值得注意的是，构造一个集成的计算成本并不比构造一个单一的学习器高很多，这也是集成学习被广泛应用的原因。

目前集成的理论成果相对来说比较成熟，然而对集成学习的优化问题仍是众多研究者的方向。集成学习是多个模型的融合，虽然它的计算成本并不比构造一个单一的学习器高很多，但在某些集成方法中，随着分类器的增加，时间开销也会大大增加，而且当分类器个数不断增加时，集成的泛化能力并不一定提高。Stacking 算法是一种有效的集成方法，它用不同的分类器产生的预测作为下一层学习算法的输入^[8]。Stacking 算法常常由不同分类算法产生，能够融合不同学习算法的学习机制，因此可以产生比组成它的任何基分类算法更高的准确率。将 Stacking 算法和其他一般的集成方法在模拟数据和真实数据上进行实证分析，发现 Stacking 方法比其他的一般集成方法表现较好^[9]。但是 Stacking 算法及其某些扩展方法的计算成本和泛化能力就受分类器个数的影响，所以有必要对集成方法进行改进优化。集成方法和 Stacking 发展迅速，目前已被广泛应用到各个领域，包括计算机视觉，图像识别。Stacking 易扩展，可以根据不同的任务来建立算法模型，有很大的提升空间，因此对 Stacking 进行研究具有广泛的意义。

1.2 国内外研究现状

集成学习就是通过结合策略将一系列个体学习器的预测结果结合,并对新样例进行预测。集成学习的主要思想是团体决策,利用多模型的思想在人类社会中已经有很长的时间。1998年,Keam 和 Valiant 在 PCA(Probably Approximately Correct)模型中首次提出了弱学习器和强学习器的概念,指出弱学习器要至少要比随机猜测好。且 Schapire 在理论上证明集成能够将弱学习器提升为强学习器^[10], Schapire 指出在知道弱学习器正确率下限的前提下,可以通过适当的集成方法,将弱学习器能够提升为强学习器。由于集成通常比组成它的单个分类器的精度要高^[6,7],所以从 1990s,在监督学习中构造好的集成学习方法已经是热点趋势之一,各领域的研究者从集成的不同方面探索了集成方法。

早期集成的研究主要有两个方面:组合分类器和弱学习器的集成。组合分类器主要用在模式识别,研究者主要是研究强分类器,并设计更强的组合策略来得到更强的组合分类器。这对深刻理解结合策略积累了经验。弱学习器的集成主要用在机器学习中,研究者主要研究弱学习器并试图设计更强的算法将弱学习器提升为强学习器,例如,Shapire 提出了 Boosting 算法,但是 Boosting 算法要求事先知道在最坏的情况下学习器的分类错误率,这很难做到。这促使了著名集成方法 AdaBoost 的产生,它们的可行性很高,也较多的用在实际任务中^[10]。但是标准的 AdaBoost 只是用于二分类任务。为了适应多分类和回归问题。Freund 和 Shapire 对标准的 AdaBoost 算法进行多次修改,形成了目前已有的适用的 AdaBoost 变体算法: AdaBoostM1, AdaBoost.M2 和 AdaBoost.R。Bagging 和 AdaBoost 都属于弱学习器集成^[11]。

以上两个方面的研究,产生了目前已有的各种集成方法,包括 AdaBoost 和 Bagging。总的来说,集成方法的构造关键在于个体学习器和结合方法。同质集成通过同种类型的个体学习器来产生,是主要的集成方法之一^[12]。这里的个体学习器可以通过数据样本的扰动,输入属性的扰动,输出表示的扰动,算法参数的扰动,或者在学习算法中引入随机性来产生。数据样本扰动就是利用训练集的不同子集来产生个体学习器。例如 Bagging 使用自助采样,AdaBoost 使用序列采样。不稳定学习器对数据样本扰动产生的集成比较有效。决策树和神经网络都是不稳定学习器,训练样本的稍加变化就会导致学习器有显著变动。通过 4 个不同大小的网络和 119 维属性的 8 个子集训练产生 32 个神经网络,集成的结果和人类专家识别相当^[13]。输入属性的扰动就是用训练样本属性的不同属性子

集来训练个体学习器。著名的随机子空间算法就依赖于随机属性扰动^[14]。以 C4.5 决策树作为个体分类器，采用随机子空间生成不同的特征空间^[15]。输出属性扰动就是对输出进行操作，增强分类器之间的多样性：用纠错输出编码将多分类任务拆解为二分类任务，可以提高 C4.5 决策树算法和反馈神经网络在多个分类问题上的性能^[16,17]，Bremiman 利用翻转法随机改变一些训练样本的类标记^[18]，AdaBoost.OC 算法将 AdaBoost 和误差校正输出编码结合^[19]；通常情况下，研究者喜欢将几种扰动同时用在一个集成学习中。例如决策树容易引入随机性，所以随机森林以决策树作为个体学习器，且引入了样本扰动和属性扰动，产生的分类器可以用典型的多数投票法或者加权投票法进行结合^[20]。随机森林在很多现实任务中具有很强大的性能。

异质集成也是产生具有差异性个体分类器的集成方法，它是由不同类型的个体学习器组成。因为每个学习算法都有自己的参数，不同的参数会产生具有较大差异性的学习器，比如神经网络的隐层神经元数和初始连接权值等，决策树的深度和非叶子节点中的划分属性标准等。更复杂的分类器结合方法被典型的应用在这种异质集成中。Stacking 可以是同质集成，但常用的是异质集成^[8,21]。Stacking 通常用另一个学习器作为结合方法，对个体学习器的输出进行再次学习。Stacking 算法中的个体学习器也称为初级学习器，用来结合的学习器称为元学习器。后来发展了许多 Stacking 的变体方法。将初级学习器的输出类概率作为元学习器的输入，实证表明多响应线性回归 MLR(Multi-response Linear Regression)作为元学习器可以很好的学习类概率^[22]。Stacking 方法的变体 SCANN，用对应分析来发现基分类器之间的相关性，并通过对原始的元层特征空间（类值预测）进行转化来消除这些相关性，以产生新的特征空间^[23]。可以用新的元层学习方法来结合分类器：元决策树 MDTs(meta decision trees)用基分类器预测的概率分布的特性（例如熵和最大概率）作为元层的属性，而不是概率分布本身^[24]。在多分类任务中，对多响应线性回归中的每个二分类问题，使用分类器对应正例的类预测的概率值，可以减少每个分类器的输入维度^[25]。用多响应模型树代替多响应线性回归来提高分类性能^[26]。

因为 Stacking 算法可以构造多层的个体分类器集合，塑造性强，可以根据具体分类问题建立对应的 Stacking 算法，所以被广泛应用到各种实际任务中：用 Stacking 算法识别命名实体^[27]；用 Stacking 方法构建了一个情感分类模型^[28]；使用 XGBTree 模型及 Stacking 多模型相融合的思想构建用户画像的二级融合算法框架，并对 CIFAR-10 图像

数据进行分类得到较好的分类结果^[29]；将进化的非线性 Stacking 方法用于 GO Player 数据中^[30]；文献针对图像分类问题建立了堆叠降噪自动编码分类器^[31]；基于网络入侵数据对 Stacking 算法在个体分类器产生、选择和结合方法上进行了改进^[32]；针对用户生成内容提出了有效的识别潜在用户的 Stacking 算法^[33]。Stacking 算法的层数可以根据具体问题进行适当的增加或减少：通过构建三层组合分类器，提高 stacking 方法在多分类任务上的分类性能^[34]；研究了专门用于大数据的大型的自动迭代的多层分类器 Stacking 方法，大大提高了分类准确性^[35]。增加数据样本也是目前提高 Stacking 算法分类性能的方法之一：用 DECORATE 算法产生一定比例的人工数据，增加了个体分类器的训练数据^[36]。用原始数据训练神经网络集成，并用神经网络集成预测的标签代替原始标签生成新的数据集，并基于这些数据产生一个 C4.5 决策树^[37]。Stacking 算法结构的优化配置也是目前的研究方向：采用人工蜂群算法优化了 Stacking 算法^[38]。用蚁群算法优化 Stacking 算法^[39,40]。

除了个体分类器产生方式的研究，个体分类器类型的选择也是众多研究者的热点。可以使用 AdaBoost 作为基分类器^[41]。可以使用随机森林和完全随机森林作为基分类器来处理图像和序列数据^[42]。文献[43]以线性分类器 SVM 作为基分类器，且把 SOM 作为元层学习方法。文献[44]将卷积神经网络作为 Stacking 算法的个体分类器，并用主成份分析方法对个体分类器的输出进行降维。文献[45]将线性和非线性结合方法做了比较。其他研究者基于 Stacking 也做了相应的研究^[22,46,47,48]。

1.3 本文主要的研究工作

对目前已有的 Stacking 算法进行研究分析，找出算法的优点和不足，并对不足之处进行改进，最后在大量的 UCI 实验数据和真实的图像数据上，通过实验验证了改进方法的可行性。本文主要对 Stacking 算法做了以下改进内容：

层级之间的属性表示。目前已有的 Stacking 算法，一般是将 T 个不同个体分类器对同一个样本的输出结果同时作为元层分类器一个输入向量的特征元素，当基分类器个数增加时，元层特征维数也会增加，这就大大增加了算法的运行时间。所以本文将 T 个不同个体分类器对同一个样本的输出类概率向量分别作为元层分类器 T 个输入向量，这样不仅增加了下一层分类器的训练数据的样本，而且下一层分类器的训练数据的维度不

会因为基分类器的增加而增加。

基分类器的选择和结合方式。弱学习器提升为强学习器的集成方法，都是以传统的分类器作为个体分类器。**Stacking** 算法更适合强的个体学习器。本文结合 **Bagging** 和 **Boosting** 学习机制的不同，以标准的随机森林，完全随机森林和梯度提升树作为候选的个体分类器。现有的集成方法的结合方式都是一种，投票法是一种常见的结合方法，简单快捷，常用于同质集成学习 **Bagging** 中，以随机森林常见。学习法常用于 **Stacking** 算法中，通过学习一个新的分类器来学习上一层个体分类器得到的结果，这可以纠正基分类器的误差。为了保持和训练过程的一致性，本文将投票法和结合法同时运用到样本预测中，提高了分类器的泛化误差。

增加 **Stacking** 的层数。为了提高泛化性能，增加了 **Stacking** 的层数。虽然本文改进的 **Stacking** 的增加了一层初级层，但是他的运行时间并不比使用相同个数初级分类器的 **Stacking** 算法的运行时间长，且增加的这一层大大提高了泛化性能。

1.4 论文的结构

第一章：绪论。首先对课题的研究背景和意义进行了介绍，详述了集成学习国内外研究现状和本课题 **Stacking** 算法的研究现状。

第二章：首先介绍了集成学习中的主要集成方法 **Bagging** 和 **Boosting**，接着介绍了集成中的结合方法，其中包括各种投票法和代数法，并从理论上证明了代数方法的贝叶斯原理计。最后介绍了模型评价方法和评价度量公式，在模型评价方法中详细介绍了 k 折交叉验证，在评价度量中详细也介绍了多分类评价度量的计算方法，为第四章中模型的评价方法提供理论基础。

第三章：主要介绍了 **Stacking** 算法的原理和各种变体算法。首先介绍了 **Stacking** 算法的框架，并详细描述了通过 k 折交叉验证产生元层学习器训练数据的具体过程和方法。然后对自己本文的主要研究方法做了简单描述，确定了在改进的方法中使用类概率向量作为个体学习器的输出，使用多响应线性回归作为元层分类器。

第四章：主要介绍改进的 **Stacking** 算法：本章建立了三层的 **Stacking** 算法，首先分别从每层的输入属性、基分类器和结合方式这三个方面详细阐述了改进的原因；接着通过文字和图相结合的方式详细介绍了改进的算法的整个训练和预测过程。然后在 15 个

UCI 数据集进行论证实验，实验结果表明了改进算法的有效性。改进算法在 ORL 图像数据集上的应用和卷积神经网络、最近集成方法的比较，表明 **Stacking** 算法的实用性。最后用遗传算法优化改进的算法，实验结果表明在实际问题中可以用遗传算法对改进的 **Stacking** 算法进行优化。

总结与展望：对本文所做的工作进行总结，并明确了后续的工作及研究方向。

第二章 集成学习及分类

集成学习又叫基于团体的学习(committee-based learning)，这个团体中的学习器称为个体学习器，它通过学习算法在训练数据上训练得到。大部分的集成方法通过一个学习算法产生同质集成，在同质集成中个体学习器也可以称为基学习器；也有一些集成方法，用多个学习算法来产生异质集成。性能度量就是模型泛化能力的评价标准，不同的性能度量往往会导致不同的评判结果。而准确率(accuracy)是分类任务中最常用，也是最基本的性能度量。实证和理论的研究证明集成的精度高于组成它的最好的基分类器。它能够从多样化的模型中平衡噪音，因此能够强化它的泛化能力。目前集成学习方法大致可分为两类，一个是序列化产生基分类器的方法，个体学习器之间存在强依赖关系，以 AdaBoost 为代表；一个是并行化集成方法，可同时生成基分类器，个体之间不存在强依赖关系，以 Bagging 为代表。AdaBoost 和 Bagging 均是从训练集选择性的重采样来产生衍生的训练集^[49]，并用于基学习算法的学习。Stacking 算法是一种有效的、通用的集成方法，它通过高层的算法来组合低层不同学习算法产生的预测结果，以达到更高的预测精度。一般来说，产生集成的步骤主要是基分类器的选择及其产生方法、基分类器的结合策略。如何获得比单一学习器更好的性能，是集成学习的关键。通过简单的理论分析，欲构建泛化能力强的集成，个体学习器要有一定的“准确性”和“精确性”。

2.1 集成学习概述

2.1.1 Boosting 算法

序列化方法的基本目的是探索基分类器之间的相关性，因为可以通过残差减少的方式来提高整体的性能。Boosting 是一族可将弱学习器提升为强学习器的算法，它是同质集成。Boosting 族算法的工作机制类似：它根据前一个分类器的表现，不断调整样本分布来序列化的产生基学习器，每次样本分布的调整都是为了让后续的分类器更多的关注先前基分类器分类错误的样本。简单的说，Boosting 序列化的训练一个学习器集合，并结合它们。在序列化训练的过程中，后者的分类器主要通过样本权重集中在前者分类器的分类错误样本上，所以 Boosting 的基学习器之间具有强相关性。

AdaBoost 算法是 Boosting 族算法最著名的代表。它的基分类器是由同一个算法在

不同的衍生训练集上训练产生，即它的训练集的每个样例是带有不同权重的：当形成第一个衍生训练集 D_1 时，每个样例的权重初始化为 $\frac{1}{m}$ ， m 为样本的个数。通常情况下，令第 j 个衍生训练集上的第 i 项样例的权重为 w_{ij} ，先前基学习器 h_1, \dots, h_{j-1} 在先前的衍生训练集 T_1, \dots, T_{j-1} 上的性能评价函数为 f_j 。AdaBoost 算法可以根据这些权重定义的训练数据的分布来进行自助抽样，用原始训练集的一部分来训练基学习算法，即“重采样法”；它也可以用带有这些权重的所有训练样例输入基学习算法中来学习基分类器，使得算法在不同的样例上关注程度不同，称为“重赋权法”。

2.1.2 Bagging 算法

并行式集成方法的基本目的是探索基分类器之间的独立性，因为可以通过结合独立的基分类器来显著性地降低误差。虽然在实际中，基分类器是从同样的训练数据中得到，不可能得到完全独立的基分类器，但是可以在学习的过程中引入随机性来降低基分类器的相关关系，常用的随机性包括样本的扰动、属性的扰动和基学习算法的不同。Bagging 是 Bootstrap AGGREGatING 的缩写，是并行式集成学习方法最著名的代表。正如名字所示意的，Bagging 的两个关键构成要素是自助和融合。

自助采样法就是在包含 m 个样本的数据集 D 中，经过 m 次的有放回随机采样操作，得到具有相同个数 m 的采样集，初始训练集 D 中有的样本在采样集中多次出现，有的没有出现。样本在 m 次采样过程中始终不被采到的概率是 $(1 - \frac{1}{m})^m$ ，取极限得到

$$\lim_{m \rightarrow \infty} \left(1 - \frac{1}{m}\right)^m = \frac{1}{e} \approx 0.368 \quad (2-1)$$

即通过自助采样，每个基学习器只用了初始训练集中约的 63.2% 的样本，剩下的约 36.8% 的样本可用来进行包外估计。

通过 T 次的这样的自助采样方法，可采样出 T 个含有 m 个样例的采样集，然后基于每个采样集训练出一个基学习器。Bagging 算法的输入是基学习算法 ζ 和训练集 D ，返回的是基分类器团体 $\{h_1, \dots, h_T\}$ ，对每个基分类器对样例的预测值进行结合，输出的就是每个样例的最终预测值。对分类任务，Bagging 常用简单投票法作为结合方法。

随机森林 RF(Random Forest)是现有集成方法的代表,基于 Bagging 的思想进行了改进,是 Bagging 的扩展变体^[20]。随机森林以决策树作为基分类器构建 Bagging,除了通过对初始训练集重采样进行样本扰动外,还进行了属性扰动,即在基决策树构造的过程中,通过从每个结点的属性集合中随机选择一部分样本属性来决定该结点的最优划分属性,进一步增强了模型的泛化能力。两个随机性的引入,使得随机森林不容易陷入过拟合,且具有很好的抗噪声能力。

随机森具有以下优点: 1) 随机森林对缺失数据和非平衡的数据不太敏感,结果比较稳健^[50,51]; 2) 随机森林可以高度并行化,在运算量没有显著提高的前提下提高了预测精度; 3) 因为决策树在非叶子节点中进行了特征选择,所以它不用做特征选择也能够处理高维度的数据;

2.1.3 偏差与方差

“偏差-方差分解”(bias-variance decomposition)是解释学习算法泛化性能的一种重要工具,它试图对学习算法的期望泛化错误率进行分解。偏差是期望输出与真实标记的差别,度量了学习算法的期望预测与真是结果的偏离程度,即刻画了学习算法本身的拟合能力;方差由样本数相同的不同训练集产生,度量了同样大小的训练集变动所导致的学习性能的变化,即刻画了数据扰动所造成的影响。给定学习任务,为了取得好的泛化性能,则需使偏差较小,即能够充分拟合数据,并使方差较小,即使得数据扰动产生的影响较小。偏差-方差分解来源于回归任务,这种分解并不适用分类任务,目前已经有不同的偏差和方差度量公式。文献[31]提供了分类任务中常用的五种偏差方差度量。Bagging 可以降低分类器的方差^[11],在实证研究中证明 AdaBoost 可以既可以降低偏差也可以降低方差^[52]。这对集成中个体分类器的选择有很大的指导作用。

2.2 结合策略

产生基分类器集合之后,不是试图找到最好的一个分类器,集成方法采取结合的方式达到强的泛化性能,所以结合对提高集成的泛化性能很重要。文献[12]把结合的好处归因于以下三个方面: 第一,在统计方面,减小因误选单个学习器导致泛化性能不佳的风险; 第二,在计算方面,降低了陷入糟糕局部及小点的风险; 第三,在表示方面,扩

大假设空间，有可能更接近学习任务的真实假设，学得更好的近似。已经通过实证研究证实，通过结合，可以降低学习算法的这些风险^[51,52,53]。

一般来说，集成学习方法主要包含两个步骤，也就是说：产生基分类器，结合基分类器。为了产生一个较好的集成，一般认为基分类器应该尽可能的精确且多样性，且找一个合适的分类器结合方法。集成的结合方法也是多样的，对于分类问题，常用的基分类器结合方法包括：面向类标签的投票法；面向类概率分布的统计算子法；学习法(combing by learning)。

假设给定大小为 T 的个体分类器集合 $\{h_1, h_2, \dots, h_T\}$ ，我们的任务是结合这 T 个基分类器，从大小为 l 的可能类标签集 $\{c_1, c_2, \dots, c_l\}$ 中预测类标签。对于样例 \mathbf{x} ，分类器 h_i 对样例 \mathbf{x} 的预测输出为 l 维向量

$$z_i(\mathbf{x}) = (h_i^1(\mathbf{x}), h_i^2(\mathbf{x}), \dots, h_i^l(\mathbf{x})) \quad (2-2)$$

其中 $h_i^j(\mathbf{x})$ 是 h_i 对类标签 c_j 的输出。根据个体分类器提供的信息， $h_i^j(\mathbf{x})$ 可以是不同的类型值，例如，可以是类标记： $h_i^j(\mathbf{x}) \in \{0, 1\}$ ， $h_i^j(\mathbf{x})$ 值为1，如果 h_i 预测作为类标签 c_j ，否则为0。可以是类概率： $h_i^j(\mathbf{x}) \in [0, 1]$ 。

1. 面向类标签的投票法：

投票法针对的 $h_i^j(\mathbf{x})$ 值是类标记。投票法是最常用，也是最基础的的结合方法，包括绝对多数投票法(majority voting)，相对多数投票(plurality voting)和加权投票(weighted voting)。

绝对多数投票法：某标记的票过半数，则预测为该标记；否则拒绝预测

$$H(\mathbf{x}) = \begin{cases} c_j, & \text{如果 } \sum_{i=1}^T h_i^j(\mathbf{x}) > \frac{1}{2} \sum_{k=1}^l \sum_{i=1}^T h_i^k(\mathbf{x}) ; \\ \text{拒绝}, & \text{否则} \end{cases} \quad (2-3)$$

相对多数投票法：预测得票数最多的标记，若同时有多个标记同时获得高票，则从中随机选择一个。

$$H(\mathbf{x}) = c_{\arg \max_j \sum_{i=1}^T h_i^j(\mathbf{x})} \quad (2-4)$$

加权投票法： w_i 是 h_i 的权重，通常 $w_i \geq 0$ ， $\sum_{i=1}^T w_i = 1$ 。

$$H(\mathbf{x}) = c_{\arg \max_j \sum_{i=1}^T w_i h_i^j(\mathbf{x})} \quad (2-5)$$

2. 面向类概率分布的代数方法:

类概率可以看作是后验概率 $P(c_j | \mathbf{x})$ 的估计, 由于类概率涵盖的信息量比较丰富, 所以有很多的组合方法用来发现类概率中涵盖的隐含信息。

基于概率产生的结合规则比较直接, 常见的组合方法有统计算子法最大、最小、均值、中值、累加、乘积方法。对于样例 \mathbf{x} , 分类器 h_i 对样例 \mathbf{x} 的预测输出的 l 维向量 $z_i(\mathbf{x})$ 中, 其中 $h_i^j(\mathbf{x})$ 表示分类器 h_i 预测样例 \mathbf{x} 属于类标签 c_j 的类概率。所有基分类器对样本 \mathbf{x} 的预测结果表示为一个 DP (decision profile) 矩阵:

$$DP(\mathbf{x}) = \begin{bmatrix} h_1^1(\mathbf{x}) & \cdots & h_1^2(\mathbf{x}) & \cdots & h_1^l(\mathbf{x}) \\ h_2^1(\mathbf{x}) & \cdots & h_2^2(\mathbf{x}) & \cdots & h_2^l(\mathbf{x}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ h_T^1(\mathbf{x}) & \cdots & h_T^2(\mathbf{x}) & \cdots & h_T^l(\mathbf{x}) \end{bmatrix} \quad (2-6)$$

在矩阵中, 第 i 行表示基分类器 h_i 对样本 \mathbf{x} 的属于所有可能类标签的后验概率, 每行表示为式(2-2)。第 j 列元素表示每个基分类器关于 $\mathbf{x} \in c_j$ 的概率。由 DP 矩阵可以看出, 矩阵涵盖了所有 T 个基分类器判定样本 \mathbf{x} 属于各个类别 $\{c_1, \dots, c_l\}$ 的后验概率。统计算子法实质上就是根据式 (2-6) 的 DP 矩阵得到一个决策向量 $[\mu_1(\mathbf{x}), \mu_2(\mathbf{x}), \dots, \mu_l(\mathbf{x})]$, 然后由决策向量得到最终分类结果 c_m : 当且仅当 $m = \max_j \mu_j$ 。 μ_j 是通过对矩阵按列进行最大、最小、均值、中值、累加、乘积运算。其思想表示为:

$$\mu_j(\mathbf{x}) = F(h_1^j(\mathbf{x}), h_2^j(\mathbf{x}), \dots, h_T^j(\mathbf{x})), \quad j = 1, \dots, c \quad (2-7)$$

上式中, F 代表了相应的操作: 最大、最小、均值、中值、累加、乘积方法。

下面给出乘积规则的贝叶斯理论。根据贝叶斯决策理论, 样本 \mathbf{x} 的预测类别应该为能够最大化后验概率 $P(c_m | h_1^m(\mathbf{x}), h_2^m(\mathbf{x}), \dots, h_T^m(\mathbf{x}))$ 的类别 c_m 。根据贝叶斯理论, 有

$$P(c_m | h_1^m(\mathbf{x}), h_2^m(\mathbf{x}), \dots, h_T^m(\mathbf{x})) = \frac{P(c_m)P(h_1^m(\mathbf{x}), h_2^m(\mathbf{x}), \dots, h_T^m(\mathbf{x}) | c_m)}{\sum_{j=1}^l P(c_j)P(h_1^m(\mathbf{x}), h_2^m(\mathbf{x}), \dots, h_T^m(\mathbf{x}) | c_j)} \quad (2-8)$$

其中 $P(h_1^m(\mathbf{x}), h_2^m(\mathbf{x}), \dots, h_T^m(\mathbf{x}) | c_m)$ 是分类器输出的联合概率分布, 假设分类器的输出是条件独立的, 也就是说,

$$P(h_1^m(\mathbf{x}), h_2^m(\mathbf{x}), \dots, h_T^m(\mathbf{x}) | c_m) = \prod_{t=1}^T P(h_t^m | c_m) \quad (2-9)$$

根据式 (2-8), 得到

$$P(c_m | h_1^m(\mathbf{x}), h_2^m(\mathbf{x}), \dots, h_T^m(\mathbf{x})) = \frac{P(c_m) \prod_{t=1}^T P(h_t^m(\mathbf{x}) | c_m)}{\sum_{j=1}^l P(c_j) \prod_{t=1}^T P(h_t^m(\mathbf{x}) | c_j)} \propto P(c_m)^{-(T-1)} \prod_{t=1}^T P(c_m | h_t^m(\mathbf{x})) \quad (2-10)$$

因为 $h_t^j(\mathbf{x})$ 是概率输出, 且 $P(c_j | h_t(\mathbf{x})) = h_t^j(\mathbf{x})$ 。如果所有的类别有相等的先验概率, 我们可以得到结合的乘积规则:

$$\mu_m(\mathbf{x}) = \prod_{t=1}^T h_t^m(\mathbf{x}) \quad (2-11)$$

类似地, 可以有最大、最小、中值、累加、平均结合规则^[54]。例如: 若(2-7)式的 F 操作为均值, 则表示为:

$$\mu_j(\mathbf{x}) = F(h_1^j(\mathbf{x}), h_2^j(\mathbf{x}), \dots, h_T^j(\mathbf{x})) = \sum_{t=1}^T h_t^j(\mathbf{x}) / T \quad (2-12)$$

Kittler^[55,56]等学者则使用贝叶斯决策论框架对这类方法(最大、最小、均值、中值、累加、乘积方法)之间的关系进行研究, 并在 M2VTS 数据库上进行了实验, 结果证明了基于累加规则的组合方法比其他方法有更好的分类效果。统计算子法同样可以给每个基分类器赋予权值, 用权值来表示每个基分类器的相对重要度, 通常每个基分类器的权值取决于它在训练样本集上的准确率, 再对加权后的基分类器进行组合得到最终分类结果。在没有考虑加权的情况下, 基分类器的组合会更加简单, 但如果基分类器的性能相差较大的话, 那么差的基分类器的表现很容易影响到组合的结果。

2.3 模型评估

在分类任务中, 我们期望以适当的方式从已存在的部分样本中学习出所有潜在样本的普遍规律, 而在现实任务中, 我们往往要学习多个分类器, 并以某种方式和期望达到的目的来选择合适的分类器。这就需要评估方法和性能度量。

2.3.1 评估方法

通常我们将已存在的能够得到的数据集划分为训练集和测试集。我们用训练集来训练调节模型，用测试集测试学习得到的模型对新样本的判别能力。常见的数据集划分方式包括留出法、交叉验证法。

留出法直接将数据集 D 划分为两个互斥的子集，一个作为训练集 S ，一个作为测试集 T 。留出法常用到的窘境是：当训练集过大，测试集过小时，评估结果的方差较大。当训练集过小，测试集过大时，评估结果的偏差较大。对这种问题，常见的做法是将大约 $2/3 \sim 4/5$ 的样本用于训练，剩余的样本用于测试。

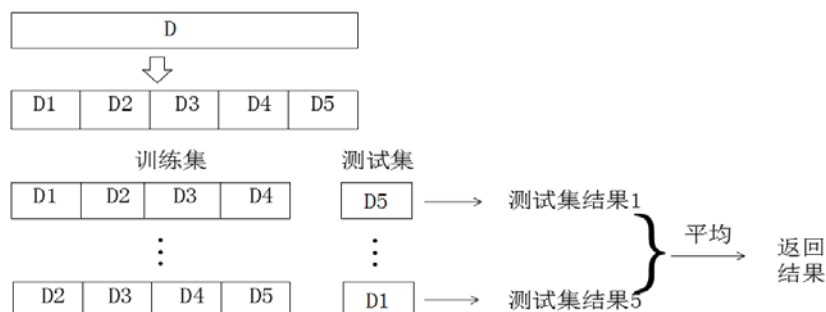


图 2-1 5 折交叉验证评估模型示意图

k 折交叉验证(k-fold cross validation)也是常用的评估方法。它的主要思想是通过数据的不同折上来重复同一个概念，即序列化的在一部分数据上训练一个模型，然后在留下的另一部分数据上观察训练得到的模型。图 2-1 给出了 5 折交叉验证评估模型的示意图，它直观的表示了图 2-2 的方法。

交叉验证先将数据 D 划分为 k 个大小相似的互斥子集，且每个子集尽可能保持数据分布的一致性，即从 D 中分层采样得到。然后每次用 $k-1$ 个子集的并集作为训练集，余下的那个子集作为测试集；这样就可以获得 k 组训练/测试集，从而可以对算法 ζ 进行 k 次训练，最终在 k 个测试集上返回 k 个预测集合 s_1, \dots, s_k ，然后对这 k 个结果分别用方法 g 进行评估得到 $g(s_1), \dots, g(s_k)$ ，其中 g 是计算模型度量的函数。

然后对这 k 个评估球平均值：
$$\sum_{i=1}^k \frac{g(s_i)}{k}$$
。交叉验证法的评估结果的稳定性和保真性

在很大程度上取决于 k 的取值。 k 常用的取值有 5, 10, 20 等。在数据的 k 折划分过程

中, 存在很多中划分方式, 不同的划分方式会导致不同的模型评估结果, 为了减少因样本划分方式引入的差别, k 折交叉验证通常要随机选择使用不同的划分重复 p 次最终的评估结果是这 p 次 k 折交叉验证结果的均值, 例如常见的 10 次 k 折交叉验证。对于包含 m 个样本的数据集 D , 若令 $k = m$, 则得到交叉验证的特例: 留一法, 在数据集比较大时, 留一法的计算开销比较大。

输入: 数据集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$;

学习算法 ζ ;

g 为评估函数: 评估分类器在数据集上的评估效果;

1. 将数据集划分为 K 个大小相似的互斥子集:

$$D = D_1 \cup D_2 \cup \dots \cup D_k, D_i \cap D_j \neq \emptyset, i \neq j$$

2. for $i = 1, \dots, k$ do

3. $h_i = \zeta(D \setminus D_i)$

4. $s_i = h_i(D_i)$

5. $g(s_i)$

6. end for

输出: $\sum_{i=1}^k \frac{g(s_i)}{k}$

图 2-2 K 折交叉验证评估模型方法

2.3.2 性能度量

在对算法进行模型评估以比较算法好坏时, 我们需要合适的评价标准。在二分类任务中, 样本的类别分为正例和反例, 图 2-3 的混淆矩阵可以表示分类器对样本分类的四种结果: 真正例 TP (true positive); 真反例 TN (true negative); 假正例 FP (false positive); 假反例 FN (false negative)。准确率(accuracy)最常用的、基准的分类性能度量指标, 是正确预测类别的样例数占总数的比例

$$\text{accuracy} = \frac{(TP + TN)}{(TP + FN + FP + TN)} \quad (2-13)$$

对于平衡数据，准确率能够很好的评价模型的好坏。但在实际任务中，数据比较复杂，类别对应的样本数分布常常不平衡，只有准确率并不能很好的衡量一个分类器的性能，查准率 $P(\text{precision})$ 和查全率 $R(\text{recall})$ 也是很好的度量方法。

表 2-1 混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP	FN
反例	FP	TN

查准率和查全率分别定义为

$$P = \frac{TP}{TP + FP} \quad (2-14)$$

$$R = \frac{TP}{TP + FN} \quad (2-15)$$

查准率和查全率是一对矛盾的度量，当查准率高时，查全率较低。反之也成立。 F_β 同时考虑了查准率和查全率，它是查准率和查全率曲线（简称 **P-R** 曲线）上的平衡点 **BEP**（Break-Event Point），即 **P-R** 曲线上“查准率=查全率”时的取值。这个取值越大，我们可认为该分类器越好。定义为

$$F_\beta = \frac{(1 + \beta^2) \times R \times P}{\beta^2 \times P + R} \quad (2-16)$$

F_β 中 β 的取值可以表达对查准率和查全率的不同偏好。当 β 为 1 时表示为

$$F_1 = \frac{2 \times R \times P}{P + R} \quad (2-17)$$

F_1 是查准率和查全率的调和平均值，很多推荐系统的评测指标就是用 F_1 值。

图 2-3 的混淆矩阵是从二分类任务中得到的，当分类任务是多分类时，我们可以通过一对多或者一对一的分类方法，得到多个二分类混淆矩阵。对于 l 个类别的多分类任务的度量，我们可以计算它们的宏(macro)度量和微(micro)度量。宏度量是先在各混淆矩

阵上分别计算出查准率和查全率，记为 $(P_1, R_1), \dots, (P_l, R_l)$ ，再计算宏查准率($macro-P$)，宏查全率($macro-R$)，宏 F_1 ($macro-F_1$)：

$$macro-P = \frac{1}{l} \sum_{i=1}^l P_i \quad (2-18)$$

$$macro-R = \frac{1}{l} \sum_{i=1}^l R_i \quad (2-19)$$

$$macro-F_1 = \frac{2 \times macro-R \times macro-P}{macro-R + macro-P} \quad (2-20)$$

微度量是先将各混淆矩阵的对应元素进行平均，得到 TP, FP, TN, FN 的平均值，再基于这些平均值计算微查准率($micro-P$)，微查全率($micro-R$)，微 F_1 ($micro-F_1$)：

$$micro-P = \frac{\overline{TP}}{(\overline{TP} + \overline{FP})} \quad (2-21)$$

$$micro-R = \frac{\overline{TP}}{(\overline{TP} + \overline{FN})} \quad (2-22)$$

$$micro-F_1 = \frac{2 \times micro-R \times micro-P}{micro-R + micro-P} \quad (2-23)$$

从公式可以看出，宏度量平等对待每个类，在多类别的不平衡数据下，受少数类影响更大，更能体现模型在少数类上的性能。

2.4 本章小结

本章先介绍了集成学习中的 Bagging 和 Boosting 学习机制，并从偏差和方差的角度说明了他们的区别。接着介绍了集成中常用的结合策略，包括投票法。并通过计算理论介绍了处理概率分布的代数方法。最后介绍了模型评估方法和二分类、多分类常用的度量公式。

第三章 Stacking 算法

3.1 Stacking 算法介绍

基分类器可以是同一个学习算法产生的同质分类器，也可以是不同的学习算法产生的异质分类器。前面介绍的 **Boosting** 和 **Bagging** 族集成学习的基分类器一般是同一个学习算法产生的。**Stacking** 本身是一种著名的集成学习方法，它的基分类器通常是不同学习算法训练得到的，有不少集成学习算法可视为其变体或特例。**Stacking** 也可看作一种特殊的、具体化的结合策略，它是学习法的典型代表。所以本小节单独介绍 **Stacking** 集成。**Stacking** 的通用过程是训练下一层的学习器来结合上一层学习器的输出。图 3-1 是一个简单的通用的 **Stacking** 分类算法拓扑结构，它是两层的结构，第一层的学习器为初学习器，也称为个体学习器。用于结合的第二层的分类器称为次级学习器或者是元学习器。

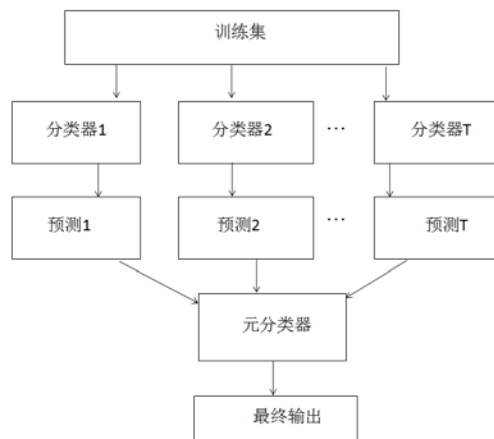


图 3-1 通用的 Stacking 算法结构

3.2 Stacking 算法框架

Stacking 集成方法的基本思想是用初始数据集训练第一层分类器，产生新的数据集训练第二层的分类器，第一层分类器的输出是第二层分类器的输入特征，同时原始标签仍被作为新数据集的标签。**Stacking** 就是进一步进行泛化，是通过元学习器来取代 **Bagging** 的投票法来综合降低偏差和方差的方法。第一层分类器通常是由不同的学习算

法产生, 因此, Stacking 算法通常是异质集成, 尽管也可以构造同质的 stacked generation。

输入: 数据集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$;

一层学习算法 ζ_1, \dots, ζ_T ;

二层学习算法 ζ 。

1: 步骤一: 通过训练数据 D 来训练一层学习算法来产生初级学习器

2: for $t = 1, \dots, T$ do

3: $h_t = \zeta_t(D)$

4: end for

5: 步骤二: 产生一个新的训练集 D'

6: $D' = \emptyset$

7: for $i = 1, \dots, m$ do

8: for $t = 1, \dots, T$ do

9: $z_{it} = h_t(\mathbf{x}_i)$

10: end for

11: $D' = D' \cup ((z_{i1}, \dots, z_{iT}), y_i)$

12: end for

13: $h' = \zeta(D')$

输出: $H(\mathbf{x}) = h'(h_1(\mathbf{x}), \dots, h_T(\mathbf{x}))$

图 3-2 通用的 Stacking 算法

通用的 Stacking 算法的思想如图 3-2。图 3-2 中算法的描述和图 3-1 中的结构是一致的, 他们是最简单的 Stacking 方法。但是图 3-1 和图 3-2 显示在训练阶段, 用训练一层分类器的数据, 也用来产生第二层学习器的训练数据, 这样容易有较高的过拟合风险。因此 Stacking 一般用交叉验证和留一法来产生下一层分类器的训练数据, 以减少这种过拟合的风险。在 Stacking 算法的训练阶段所使用的交叉验证和模型评估方法中的交叉验证的目的不一样。评估算法中, 是将所有数据划分成 k 组训练/测试集, 他的主要目的是

测试（具体过程见图 2-1 和图 2-2）。而 Stacking 算法巧妙地扩展了交叉验证，Stacking 算法需要将原始数据通过一定的方式划分为一组训练集 D 和测试集 D_{test} ，然后在训练集 D 上用交叉验证产生下一层级的训练集，最后在测试集 D_{test} 上评估算法的好坏。图 3-3 显示了用交叉验证训练一层基分类器并产生下一层学习算法的训练数据的方法。 m 表示训练集 D 的大小，它小于总数据大小 n 。

为了更容易理解 Stacking 算法，接下来解释 k 折交叉验证产生 Stacking 算法元层分类器训练数据的过程。图 3-3 中 Stacking 算法有两层：初级学习层（一层）和元学习层（二层）。我们给定初级学习层的个体学习器使用不同的学习算法产生。这里我们给定 T 个初级学习算法 ζ_1, \dots, ζ_T 和一个元层算法 ζ 。

训练阶段：先将训练数据 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ 随机划分为 k 个大小相似且互斥的集合 D_1, D_2, \dots, D_k ， $D_i \cap D_j = \emptyset$ 。令 $\bar{D}_j = D \setminus D_j$ ， D_j 分别表示第 j 折的训练集和测试集。接下来迭代 k 次，每次迭代时，对每个学习算法 ζ_t ，初级学习器 $h_t^{(j)}$ 是第 t 个学习算法 ζ_t 在 \bar{D}_j 上训练得到，训练好的学习器 $h_t^{(j)}$ 对 D_j 中每个样本 \mathbf{x}_i 进行预测，表示为 $h_t^{(j)}(\mathbf{x}_i)$ ，当 T 个模型都进行完上述操作后，我们就可以得到 T 个模型对 D_j 中所有样本的预测结果。当 k 次迭代都完成后，我们就获得了 T 个模型对所有训练样本的预测结果。则由 \mathbf{x}_i 产生的次级训练样例的示例部分为 $\mathbf{x}'_i = (h_1^{(j)}(\mathbf{x}_i), h_2^{(j)}(\mathbf{x}_i), \dots, h_T^{(j)}(\mathbf{x}_i))$ ，标记仍为 y_i 。于是从这 T 个初级学习器产生的次级训练集（新数据）是 $D' = \{(\mathbf{x}'_i, y_i)\}_{i=1}^m$ 。然后将 D' 用于训练次级学习算法 ζ ，得到次级学习器 $h' = \zeta(D')$ 。

测试阶段：由于在训练阶段，每个学习算法 ζ_t 在 K 折训练集上产生 k 个初级学习器 $h_t^{(j)}$ ， $j=1, \dots, k$ ，所以在测试阶段，这 k 个学习器在测试集 D_{test} 中的一个样本 \mathbf{x} 上产生 k 个预测结果 $(h_1^{(1)}(\mathbf{x}), h_1^{(2)}(\mathbf{x}), \dots, h_1^{(k)}(\mathbf{x}))$ ，将这 k 个结果取平均：

$$\bar{h}_t(\mathbf{x}) = \sum_{j=1}^k h_t^{(j)}(\mathbf{x}) / k \quad (3-1)$$

作为算法 ς_t 在测试样例 \mathbf{x} 上的预测值，将测试样例 \mathbf{x} 输入这 T 个学习算法后，得到 $(\bar{h}_1(\mathbf{x}), \bar{h}_2(\mathbf{x}), \dots, \bar{h}_T(\mathbf{x}))$ ，然后输入已经训练好的次级学习器 h' 上，输出样例 \mathbf{x} 的最终预测结果 $H(\mathbf{x}) = h'(\bar{h}_1(\mathbf{x}), \bar{h}_2(\mathbf{x}), \dots, \bar{h}_T(\mathbf{x}))$ 。

输入：数据集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ；

一层学习算法 $\varsigma_1, \dots, \varsigma_T$ ；

二层学习算法 ς ；

- 1: 步骤一：用交叉验证的方法训练一层学习器，得到元层学习器的训练集
 - 2: 将数据集 D 划分为 K 个大小相似的互斥子集
 - 3: for $j = 1, \dots, k$ do
 - 4: for $t = 1, \dots, T$ do
 - 5: $h_t^{(j)} = \varsigma_t(D \setminus D_j)$
 - 6: end for
 - 7: for $\mathbf{x}_i \in D_j$ do
 - 8: 计算 $h_t^{(j)}(\mathbf{x}_i)$
 - 9: end for
 - 10: end for
 - 11: 将 T 个基分类器对所有训练样本的输出结果表示为 $(h_1^{(j)}(\mathbf{x}_i), h_2^{(j)}(\mathbf{x}_i), \dots, h_T^{(j)}(\mathbf{x}_i))$
 - 12: 新的训练数据集表示为 $D' = \{(\mathbf{x}'_i, y_i)\}_{i=1}^m = \{(h_1^{(j)}(\mathbf{x}_i), \dots, h_T^{(j)}(\mathbf{x}_i), y_i)\}_{i=1}^m$
 - 13: $\bar{h}_t(\mathbf{x}) = \sum_{j=1}^k h_t^{(j)}(\mathbf{x}) / k$
 - 14: 然后将 D' 用于训练次级学习算法 ς ，得到次级学习器 $h' = \varsigma(D')$
-

输出： $H(\mathbf{x}) = h'(\bar{h}_1(\mathbf{x}), \bar{h}_2(\mathbf{x}), \dots, \bar{h}_T(\mathbf{x}))$

图 3-3 用 K 折交叉验证训练个体分类器

3.3 元层学习器的属性表示和元层学习算法的研究

Wolpert 表明新数据的属性表示和次级学习算法对 Stacking 集成的泛化性能影响很大^[8]。我们已经知道式 (2-6) 矩阵中的元素值可以是类标签也可以是类概率。类概率不仅考虑了预测值,而且还充分考虑了基分类器的置信度和差异性。即使对同一个样本做出相同类别预测的个体分类器,它们预测的后验概率也不一定相同。对于可以输出类概率的初级分类器,分类器 h 对样例 \mathbf{x} 的预测是 \mathbf{x} 属于所有可能类标签的概率 $p^h(c_j | \mathbf{x})$, $j=1, \dots, l$, 其中 $p^h(c_j | \mathbf{x})$ 定义了分类器 h 估计样例 \mathbf{x} 属于类别 c_j 的概率值。尽管分类器预测 \mathbf{x} 的所属类别为拥有最高概率值 $p^h(c_j | \mathbf{x})$ 的类别 c_j , 但是得到的所有类的类概率包含更多有用的信息。因此可以对一层分类器在样例 \mathbf{x} 上预测的类概率 $p^h(c_j | \mathbf{x})$, $j=1, \dots, l$, $t=1, \dots, T$, 用一定的方式进行处理, 然后和 \mathbf{x} 的真正类标签 y 组成一个新的训练样例。直接将一层基分类器对所有类预测的后验概率 $p^h(c_j | \mathbf{x})$, $j=1, \dots, l$, $t=1, \dots, T$, 作为元层学习器的输入属性^[22]。即将式 (2-6) 的每个元素值同时做为次级输入属性用于次级学习器。所产生的特征维数等于一层基分类器个数与类别数的乘积 $T \times l$ 。文献[26]在概率分布属性集的基础上增加了两个额外的属性集:

乘以最大概率的概率分布

$$P^h(c_j | \mathbf{x}) = p^h(c_j | \mathbf{x}) \times M_{h_t(\mathbf{x})} = p^h(c_j | \mathbf{x}) \times \max_{j=1}^l (p^h(c_j | \mathbf{x})) \quad (3-2)$$

概率分布的熵

$$E_{h_t}(\mathbf{x}) = \sum_{j=1}^l p^h(c_j | \mathbf{x}) \times \log_2 p^h(c_j | \mathbf{x}) \quad (3-3)$$

由此产生的特征维数为 $2T \times l + T$, 即两倍的个体分类器个数和类别数的乘积, 加上分类器个数。这种元层属性的扩展提高了算法的泛化性能, 但随着一层基分类器数量的增加, 元层学习器的属性也会成倍的增加, 导致训练时间增加。为了降低元层学习器的输入维数, Tsoumakas G 将基分类器的类概率输出按类求平均^[57]。然后输入到元层分类器中, 但是这种方法却弱化了每个分类器的作用。周志华在处理图像和序列数据时, 在概率分布的基础上, 又在每一层分类器的输入属性中均加入了原始特征向量^[42]。

为了增加 Stacking 的泛化性能,许多研究者对元层分类器的选择也做了研究。有研究表明,对于初级分类器输出的类概率分布,多响应线性回归 MLR (Multi-response linear regression) 是一个较好的学习方法^[22]。MLR 是基于线性回归的分类器,对于有 l 个类别值 $\{c_1, c_2, \dots, c_l\}$ 的分类问题,形成了 l 个分类问题:对每个类别 c_j ,构造一个线性分类器 LR_j 来预测一个二值变量,当 c_j 是正确的类标签时,这个二值变量是 1, 否则为 0。给一个新的待分类样例 \mathbf{x} , 对于所有 j 的取值,计算 LR_j , 并预测 \mathbf{x} 的类别为拥有最大值 $LR_k(\mathbf{x})$ 的 k 。文献[19]提出对于 MLR 中的 l 个线性分类器,应该用不同的特征集。也就是说,在构造类别 c_j 的分类模型 LR_j 时,只需要不同分类器预测的关于类 c_j 的概率,即 $p^h(c_j | \mathbf{x}), t=1, \dots, T$, 因此每个线性回归问题都有 T 个特征,而不是 $m \times T$ 个特征,减少了每个 LR_j 的属性维度。报告显示其性能超过用全部的概率属性。

文献[24]介绍了一个新的元层学习方法:在元层学习器 MDTs(meta decision trees)中将基分类器预测的概率分布性质(例如熵和概率最大值)作为元层分类器 MDTs 的输入属性。这些概率分布性质不仅能反映了基层分类器的置信度,且产生相对较小的 MDTs,使得 MDTs 更容易解释。基于 MDTs 学习法的 Stacking 明显比以决策树作为基分类器的投票法和 Stacking 方法表现好,也优于以决策树作为基分类器的 Boosting 和 Bagging^[25]。但是和用 MLR 作为元层分类器的 Stacking 方法比较,MDTs 稍逊一些^[26]。

总的来说,MDTs,采用 MLR 的 Stacking,还有 SelectBest,他们的性能似乎相当,但在他们当中,用 MLR 作为元层学习法、且元层属性值为类概率值的 Stacking 算法(本文称此算法为 SMLR 算法)表现更好一些。所以本文基于 SMLR 提出改进的 Stacking 算法,使改进的算法能够有较小的计算复杂度和较好的分类性能。

3.4 本章小结

本章主要介绍了 Stacking 算法的相关理论。首先介绍了通用 Stacking 算法的框架和不足,接着引出并详细介绍了 K 折交叉验证在 Stacking 算法中的使用方法。第三节介绍了影响 Stacking 算法性能的较为关键的因素:属性表示和学习器的选择。基于这两个因素,通过分析现有的 Stacking 算法的研究成果,总结出了现有方法的优点:以概率分布作为输出;缺点:分类性能,特别是在非平衡数据集上的分类性能有待进一步提高,算法运行时间长。由此引出下文对传统的 Stacking 算法进行改进的必要性。

第四章 Stacking 算法的改进

用概率分布和多响应线性回归组成的 SMLR 算法是较为常用的一个 Stacking 算法，基于 SMLR 的优势，仍然使用概率分布输出而不是类标签值，且最后一层学习算法仍使用 MLR，本文在三个方面进行了改进：一是改变层级之间的输入属性表示方法，降低特征维数；二是将不同的集成分类器作为个体分类器；三是结合方法，在整个模型中，本文在常用的两层 Stacking 算法的基础上增加了一层初级层。结果表明改进的 Stacking 算法减少了运行时间，且分类性能在 15 个 UCI 数据集和 ORL 图像数据集上表现较优^[58]。

4.1 改进算法

常见的 Stacking 算法一般采用两层学习结构：第一层称为初级学习层，初级学习层由多个学习器组成；第二层称为元学习层，它使用一个学习算法对前一层学习器的输出进行学习。本文提出的 Stacking 算法的扩展，使用三层结构学习数据集的潜在规律。第一层和第二层是初级学习层，分别称为 1-层学习层和 2-层学习层，学习器分别称为 1-层个体学习器和 2-层个体学习器。第三层是元学习层，对前面两层的学习结果进行学习，在本文中也可称它为 3-层学习层，三层的学习器称为元学习器。

4.1.1 2-层和 3-层个体学习器的输入属性表示

给定训练集 $D = \{s_i | i = 1, \dots, m\} = \{\mathbf{x}_i, y_i | i = 1, \dots, m\}$ ， $y_i \in \{c_1, \dots, c_l\}$ 。 m 为样本个数， l 为类别数。SMLR 算法在第一层，通过训练集 D 和 T 个学习算法，使用 K 折交叉验证产生二层学习器的训练集（具体过程见图 3-3）

$$D' = \{(\mathbf{x}'_i, y_i) | i = 1, \dots, m\} = \{\mathbf{z}_{i1}, \mathbf{z}_{i2}, \dots, \mathbf{z}_{iT}, y_i | i = 1, \dots, m\}$$

元层的输入向量 \mathbf{x}'_i 表示为 $(\mathbf{z}_{i1}, \mathbf{z}_{i2}, \dots, \mathbf{z}_{iT})$ ，其中 $\mathbf{z}_{it} = (p^h(c_1 | \mathbf{x}_i), \dots, p^h(c_l | \mathbf{x}_i))$ ， $p^h(c_j | \mathbf{x}_i)$ 表示第 t 个分类器判断样本 \mathbf{x}_i 属于第 j 类的概率值。由元层的输入向量表示，我们知道，随着一层基分类器数量的增加，元层学习器的属性维度也会成倍的增加，不仅增加计算时间，而且随着维度的增加，元层输入数据集中，每个数据的距离也会增加，

这是因为：给定一个分类任务， $p_t(\mathbf{x}_1)$ ， $p_t(\mathbf{x}_2)$ 分别表示第 t 个分类器对原始样本 \mathbf{x}_1 ， \mathbf{x}_2 的预测的类概率向量。当 Stacking 算法中基分类器的个数为 T 个时，元层分类器的两个输入向量表示为 $P(\mathbf{x}_1) = (p_1(\mathbf{x}_1), p_2(\mathbf{x}_1), \dots, p_T(\mathbf{x}_1))$ ， $P(\mathbf{x}_2) = (p_1(\mathbf{x}_2), p_2(\mathbf{x}_2), \dots, p_T(\mathbf{x}_2))$ ，它们的欧氏距离表示为

$$L(\mathbf{x}_2, \mathbf{x}_1) = \|P(\mathbf{x}_2) - P(\mathbf{x}_1)\|_2 \quad (4-1)$$

当基分类器的个数增加到 T_1 时，此时($T_1 > T$)，元层分类器的输入属性向量表示为 $P'(\mathbf{x}_1) = (p_1(\mathbf{x}_1), \dots, p_T(\mathbf{x}_1), \dots, p_{T_1}(\mathbf{x}_1))$ ， $P'(\mathbf{x}_2) = (p_1(\mathbf{x}_2), \dots, p_T(\mathbf{x}_2), \dots, p_{T_1}(\mathbf{x}_2))$ ，它们的欧氏距离表示为

$$L'(\mathbf{x}_2, \mathbf{x}_1) = \|P'(\mathbf{x}_2) - P'(\mathbf{x}_1)\|_2 \quad (4-2)$$

很明显 $L'(\mathbf{x}_2, \mathbf{x}_1) > L(\mathbf{x}_2, \mathbf{x}_1)$ 。所以当基分类器个数增加时，元层训练集的数据分布会越来越稀疏，维度越来越高。当维数越来越多时，分析和处理多维数据的复杂度和成本指数级增长。然而我们期望得到的是有效的低维大样本数据。所以这就促使我们寻找其他的属性表示方法。

本章的改进算法中，1-层算法和 2-层分类器的输出值的处理方式不同。即 2-层分类算法和元层分类算法的输入属性表示方法不同。

首先介绍 2-层分类算法的输入属性表示。我们已经知道第 t 个分类器在第 i 个样本上的预测值 z_{it} 表示类概率向量，即 $z_{it} = (p^h(c_1 | \mathbf{x}_i), \dots, p^h(c_l | \mathbf{x}_i))$ ，其中 $p^h(c_j | \mathbf{x}_i)$ ，表示第 t 个分类器判断样本 \mathbf{x}_i 属于第 j 类的概率。然后将每个 z_{it} 和产生它的原始样例 \mathbf{x}_i 组合作为 2-层分类算法的输入数据，即将

$$\{(\mathbf{x}_i, z_{it}, y_i) | i = 1, \dots, m; t = 1, \dots, T\} \quad (4-3)$$

作为 2-层分类算法的训练集。这不同于 SMLR 中元层训练数据的表示方法：将每个分类器对同一个样例预测的输出同时作为一个新样例的属性：即 2-层的训练数据为 $\{(z_{i1}, \dots, z_{iT}, y_i) | i = 1, \dots, m\}$ 。式(4-1)这种新的属性表示方法使得 2-层训练数据大小为 $m \times T$ ，二层属性的大小为保持 $l + |\mathbf{x}_i|$ 不变，其中 $|\mathbf{x}_i|$ 表示向量 \mathbf{x}_i 的维数大小。

之所以采用式 (4-3) 这种 2-层训练数据表示方式，是因为：

Stacking 更适合大样本数据集，为了增加数据样本，本文通过 1-层个体分类器对原始训练集进行学习，基于个体分类器的输出结果来构造所有原始样本的“近邻”，增加样本量和样本分布密度。我们知道 T 个不同分类器对同一个样本 \mathbf{x} 的预测的类标签的概率会有差别，即 T 个类概率向量 $(p^h(c_1|\mathbf{x}), \dots, p^h(c_l|\mathbf{x}))$ ， $t=1, \dots, T$ 是不同的。在用单个学习器 h 判定 \mathbf{x} 的类别时，如果分类器的输出是类概率向量，则分类器判定类概率向量中具有最大概率值的类 c_j 是 \mathbf{x} 的类别。

假设 T 个个体分类器均能对 \mathbf{x} 正确预测其类标签，那么预测出的每个类概率向量中具有最大概率值的类和 \mathbf{x} 真实类别是一样的，则 T 个类概率向量的相同维对应的元素值是接近的。所以对于同一个样例 \mathbf{x}_i 产生的二层训练数据 $((\mathbf{x}_i, \mathbf{z}_{it}), y_i)$ 的属性 $(\mathbf{x}_i, \mathbf{z}_{it})$ ， $t=1, \dots, T$ 之间也是相似的（基于欧氏距离）。如果我们将原始训练集 D 的每个样例 \mathbf{x}_i 均扩展成向量 $\tilde{\mathbf{x}}_i = (\mathbf{x}_i, c_{i0}, \dots, c_{ij}, \dots, c_{il})$ ，其中 $c_{ij}=1$ ，如果 \mathbf{x}_i 的真实标记 y_i 是 c_j 类，反之 $c_{ij}=0$ 。则 2-层的 T 个输入向量 $(\mathbf{x}_i, p^h(c_1|\mathbf{x}), \dots, p^h(c_l|\mathbf{x}))$ ， $t=1, \dots, T$ 和向量 $(\mathbf{x}_i, c_{i0}, \dots, c_{ij}, \dots, c_{il})$ 之间的相似性均比较大（基于欧氏距离）。因为每个类概率向量的 L1 范式为 1，即 $\sum_{j=1}^l p^h(c_j|\mathbf{x})=1$ ，其中 $0 \leq p^h(c_j|\mathbf{x}) \leq 1$ ，所以 2-层学习器的 T 个输入向量 $(\mathbf{x}_i, p^h(c_1|\mathbf{x}), \dots, p^h(c_l|\mathbf{x}))$ ， $t=1, \dots, T$ ，两两之间的欧氏距离小于 \sqrt{l} ，且它们和向量 $\tilde{\mathbf{x}}_i = (\mathbf{x}_i, c_{i0}, c_{i1}, c_{i2}, \dots, c_{il})$ 的欧氏距离均小于 \sqrt{l} 。所以可以将这 T 个向量看成和 $\tilde{\mathbf{x}}_i$ 相关的 T 个“近邻”样本（基于欧氏距离）。

由此可以知道这种方法带来的几个好处：首先随着基分类器的增加，产生的新样本之间的欧氏距离也不会超过 \sqrt{l} 。其次这种表示方式仍然可以保留每个分类器预测的概率值本身，而不是处理类概率后的值。再者 2-层学习器输入属性维度为 $m+|\mathbf{x}_i|$ 不会变（其中 $|\mathbf{x}_i|$ 表示向量的维度），这不会较大程度的增加计算成本。这样我们就可以基于所有原始训练样本的 T 个“近邻”来训练 2-层学习算法，测试过程中，我们也可以通过预测测试样例的 T 个近邻来判断测试样例的最终类别。

接下来我们介绍 3-层（元层）学习器的输入属性。在 2-层输入属性的介绍中，本文进行了假设，假设 T 个分类器均能对 \mathbf{x} 正确预测其类标签，但在实际中，集成的个体分类器不可能都对所有样本分类正确，所以式 (4-3) 表示的数据集中带有噪声，所以增加一层初级学习层，通过 2-层个体学习器进行再次学习，并通过对 2-层个体学习器的输出结果进行平均来减弱噪声。式 (2-8) 到式 (2-11) 表明了代数方法的理论有效性。且通过平均 3-层的输入数据大小和 2-层的输入数据大小一样，为 $m \times T$ 。

在 3-层中我们仍然加入原始特征，之所以将原始特征加入到 2-层和 3-层的输入属性中是为了保留原始属性和其类概率之间的隐含关系。在实际中可以根据分类结果选择加入或者不加入原始特征。

4.1.2 基分类器的选择

AdaBoost 算法和随机森林中常用的基分类器是传统的弱分类器，它们是将弱学习器 (weak learners) 提升为强学习器。而 Stacking 算法是一种特殊的融合方法，它需要用下一层学习算法来学习（融合）当前层的输出，所以融合模型要优质且不同。本文改进的算法在 1-层和 2-层使用的基分类器是三种集成算法：随机森林 RF (Random Forest)、完全随机森林 CRF (Completely-random Tree Forest)、梯度提升树 GBDT (Gradient Boosting Decision Tree)^[20,59,60]。

建立一个多样性的基分类器集合是集成学习的关键。已经注意到：在不影响个体误差率的情况下，基分类器预测的不一致性会增加集成学习的准确率^[46]。我们期望用多个方法产生的基分类器可以增加基分类器之间的多样性，RF, CRF, GBDT 本身也是集成算法，一般情况下它们的性能相对于传统的单一学习算法要好。且从偏差-方差的角度看，RF、CRF 和 GBDT 侧重点不同，RF、CRF 主要是降低误差的方差项，GBDT 既降低方差又降低偏差。假设他们是通过不同的机制来产生的有效性，那可以猜想这些集成算法的结合会产生更好的结果。所以在改进的算法中选择 RF, CRF, GBDT 作为候选的个体分类器。

4.1.3 结合方法

一些研究已经证明，MLR 是适用于学习类概率的算法。改进的 Stacking 算法的最

后一层的学习算法为 MLR。2-层和 3-层的输入属性表示方法决定了在预测过程中，经过 1-层和 2-层的基学习器，测试样例 \mathbf{x} 会产生 T 个“近邻”的样本，然后用 MLR 对 \mathbf{x} 的这 T 个“近邻”预测出 T 个类别，最后需要投票来决定 \mathbf{x} 最终类别。

4.1.4 改进的 Stacking 算法

经过对已有的 Stacking 算法 SMLR 的以上三个方面的扩展，本文建立了三层的改进的 Stacking 算法模型。在第一层和第二层中均以 RF、CRF 和 GBDT 为个体分类器，第三层的分类器为多响应线性回归 MLR。为了方便，本文称改进的 Stacking 算法为 StackingM。为了描述的方便，假设 1-层和 2-层的学习算法集合分别为 $\{\varsigma_1, \varsigma_2, \dots, \varsigma_T\}$ 、 $\{\tau_1, \tau_2, \dots, \tau_{T_1}\}$ ，个数分别为 T 、 T_1 ，3-层的学习算法为 ς 。给定训练数据 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ，其中 $y_i \in \{c_1, \dots, c_l\}$ ，进行的是 l 分类任务，测试样例为 \mathbf{x} ，训练数据特征大小为 m 。为了避免过拟合，1-层和 2-层的训练数据均通过 k 折交叉验证产生（图 3-3）。改进的 Stacking 算法的训练和测试过程如图 4-1、图 4-2、图 4-3，它们分别表示 1-层、2-层和 3-层的训练和测试过程。下面我们分别介绍这三个图。

1-层训练过程和测试过程：用原始训练集 D 通过 k 折交叉验证和 1-层的 T 个学习算法 $\{\varsigma_1, \varsigma_2, \dots, \varsigma_T\}$ ，来预测训练集 D 中的每个样本 \mathbf{x}_i 的预测结果 $\mathbf{z}_{it} = h_t^{(j)}(\mathbf{x}_i)$ ， $i = 1, \dots, m; t = 1, \dots, T$ ，其中 $h_t^{(j)}$ 表示第 t 个算法在第 j 折数据子集上训练得到的分类器， \mathbf{z}_{it} 表示的是类概率向量 $(p^h(c_1 | \mathbf{x}), \dots, p^h(c_l | \mathbf{x}))$ ，其中 $p^h(c_j | \mathbf{x}_i)$ 表示第 t 个分类器 h_t 预测样本 \mathbf{x}_i 属于 c_j 类的概率。将第一层得到的训练集的预测结果 $\mathbf{z}_{it} = h_t^{(j)}(\mathbf{x}_i), t = 1, \dots, T$ 和原始特征 \mathbf{x}_i 作为新训练集的属性 $\mathbf{x}'_{it} = (\mathbf{z}_{it}, \mathbf{x}_i)$ 。则新数据集表示为 $D' = \{(\mathbf{x}'_{it}, y_i) | i = 1, \dots, m; t = 1, \dots, T\} = \{(\mathbf{z}_{it}, \mathbf{x}_i, y_i) | i = 1, \dots, m; t = 1, \dots, T\}$ ，其中每个基学习算法预测的值 \mathbf{z}_{it} 均与样例 \mathbf{x}_i 及其对应的标签值 y_i 组合，新训练集的大小为 $m \times T$ ，属性维度为 $l + m$ ， m 表示向量 \mathbf{x}_i 的维数。

然后将测试集输入第一层训练得到的分类器 $h_t^{(j)}$ 中，得到所有分类器对测试样本 \mathbf{x}

的测试结果 $\{\bar{h}_1(\mathbf{x}), \bar{h}_2(\mathbf{x}), \dots, \bar{h}_T(\mathbf{x})\}$, $\bar{h}_t(\mathbf{x})$ 表示第 t 个算法学习得到的 k 个分类器预测 \mathbf{x} 的类概率向量的平均。然后将测试结果 $\{\bar{h}_1(\mathbf{x}), \bar{h}_2(\mathbf{x}), \dots, \bar{h}_T(\mathbf{x})\}$ 中的元素分别与 \mathbf{x} 组合：
 $\{(\bar{h}_1(\mathbf{x}), \mathbf{x}), (\bar{h}_2(\mathbf{x}), \mathbf{x}), \dots, (\bar{h}_T(\mathbf{x}), \mathbf{x})\}$

输入：数据集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ ；1-层学习算法 ζ_1, \dots, ζ_T 。

```

1: 将训练集划分为  $k$  个大小相似的互斥子集
2: for  $j = 1, \dots, k$  do
3:   for  $t = 1, \dots, T$  do
4:      $h_t^{(j)} = \zeta_t(D \setminus D_j)$ 
5:   end for
6:   for  $\mathbf{x}_i \in D_j$  do
7:      $z_{it} = h_t^{(j)}(\mathbf{x}_i)$ 
8:   end for
9: end for
10: 2-层的训练集表示为  $D' = \{(\mathbf{x}'_{it}, y_i) \mid i = 1, \dots, m; t = 1, \dots, T\}$ 

```

输出： $D' = \{(\mathbf{x}'_{it}, y_i) \mid i = 1, \dots, m; t = 1, \dots, T\}$ ；分类器 $h_t^{(j)}$, $t = 1, \dots, T$, $j = 1, \dots, k$ 。

图 4-1 1-层训练过程

输入：测试样例 \mathbf{x} ；训练好的 1-层个体分类器 $h_t^{(j)}$, $t = 1, \dots, T$, $j = 1, \dots, k$ 。

```

1: for  $t = 1, \dots, T$  do
2:    $\bar{h}_t(\mathbf{x}) = \frac{1}{k} \sum_{j=1}^k h_t^{(j)}(\mathbf{x})$ 
3: end for
4:  $\bar{h}_t(\mathbf{x}) = \frac{1}{k} \sum_{j=1}^k h_t^{(j)}(\mathbf{x})$ 
5: 将每个  $\bar{h}_t(\mathbf{x})$  与原始  $\mathbf{x}$  组成 2-层测试集： $\{(\bar{h}_1(\mathbf{x}), \mathbf{x}), (\bar{h}_2(\mathbf{x}), \mathbf{x}), \dots, (\bar{h}_T(\mathbf{x}), \mathbf{x})\}$ 

```

输出： $\{(\bar{h}_1(\mathbf{x}), \mathbf{x}), (\bar{h}_2(\mathbf{x}), \mathbf{x}), \dots, (\bar{h}_T(\mathbf{x}), \mathbf{x})\}$

图 4-2 1-层测试过程

2-层训练和测试过程：在新训练集 D' 上，同样用 K 折交叉验证训练 2-层的 T_1 个基学习算法 $\{\tau_1, \tau_2, \dots, \tau_{T_1}\}$ ，得到 T_1 个学习算法训练得到的分类器对训练集 D' 中所有样本 \mathbf{x}'_{it} 的预测结果 $\mathbf{z}'_{it'} = g_{t'}^{(j)}(\mathbf{x}'_{it})$ ， $i=1, \dots, m$ ， $t=1, \dots, T$ ， $t'=1, \dots, T_1$ ；其中 $\mathbf{z}'_{it'}$ 表示的是第 t' 个算法预测样本 \mathbf{x}'_{it} 的类概率向量 $(p^{h_{t'}}(c_1 | \mathbf{x}'_{it}), p^{h_{t'}}(c_2 | \mathbf{x}'_{it}), \dots, p^{h_{t'}}(c_l | \mathbf{x}'_{it}))$ 。将它们表示成 DT 矩阵：

$$DT(\mathbf{x}'_{it}) = \begin{bmatrix} g_1^{(1)}(\mathbf{x}'_{it}) & g_1^{(2)}(\mathbf{x}'_{it}) & \cdots & g_1^{(l)}(\mathbf{x}'_{it}) \\ g_2^{(1)}(\mathbf{x}'_{it}) & g_2^{(2)}(\mathbf{x}'_{it}) & \cdots & g_2^{(l)}(\mathbf{x}'_{it}) \\ \vdots & \vdots & \cdots & \vdots \\ g_{T_1}^{(1)}(\mathbf{x}'_{it}) & g_{T_1}^{(2)}(\mathbf{x}'_{it}) & \cdots & g_{T_1}^{(l)}(\mathbf{x}'_{it}) \end{bmatrix}, \quad i=1, \dots, m, \quad t=1, \dots, T \quad (4-4)$$

对每个样本 \mathbf{x}'_{it} 的 DT 矩阵按列求平均，得到向量

$$\bar{\mathbf{z}}'_{it} = \sum_{t'=1}^{T_1} \mathbf{z}'_{it'}, \quad i=1, \dots, m, \quad t=1, \dots, T \quad (4-5)$$

然后将原始特征向量 \mathbf{x}_i 和向量 $\bar{\mathbf{z}}'_{it}$ 组合 $\mathbf{x}''_i = (\bar{\mathbf{z}}'_{it}, \mathbf{x}_i)$ ，将 \mathbf{x}''_i 作为为 3-层学习算法 MLR 的输入属性，所以 MLR 的训练数据为 $D'' = \{(\bar{\mathbf{z}}'_{it}, \mathbf{x}_i, y_i) | i=1, \dots, m; t=1, \dots, T\}$ ， D'' 的大小仍为 $m \times T$ ，属性维度为 $l+m$ ， m 表示向量 \mathbf{x}_i 的维数。然后输入 2-层训练得到的分类器，得到测试集每个样例 $(\bar{h}_t(\mathbf{x}), \mathbf{x})$ 的测试结果 $\bar{g}'_{t'}(\bar{h}_t(\mathbf{x}), \mathbf{x}), t=1, \dots, T, t'=1, \dots, T_1$ ，其中 $\bar{g}'_{t'}(\bar{h}_t(\mathbf{x}), \mathbf{x})$ 表示第 t' 个算法产生的 k 个分类器预测向量 $(\bar{h}_t(\mathbf{x}), \mathbf{x})$ 的类概率向量平均后的向量 $\bar{g}'_{t'}(\bar{h}_t(\mathbf{x}), \mathbf{x}) = \sum_{j=1}^k \frac{g_{t'}^{(j)}(\bar{h}_t(\mathbf{x}), \mathbf{x})}{k}$ ，将 $\{(\bar{g}'_{t'}(\bar{h}_t(\mathbf{x}), \mathbf{x}) | t=1, \dots, T; t'=1, \dots, T_1)\}$ 按 2-层分类器进行累加求平均，得到 T 个向量：

$$\sum_{t'=1}^{T_1} \frac{\bar{g}'_{t'}(\bar{h}_t(\mathbf{x}), \mathbf{x})}{T_1}, \quad t=1, \dots, T \quad (4-6)$$

将集合 $\left\{ \sum_{t'=1}^{T_1} \frac{\bar{g}'_{t'}(\bar{h}_t(\mathbf{x}), \mathbf{x})}{T_1} \middle| t=1, \dots, T \right\}$ 中的每个向量和原始特征 \mathbf{x} 组合得到

$$\left\{ \left(\mathbf{x}, \sum_{t'=1}^{T_1} \bar{h}'_{t'}(\bar{h}_t(\mathbf{x})) \right) \middle| t=1, \dots, T \right\} \quad (4-7)$$

输入: $D' = \{\mathbf{x}'_i, y_i \mid i = 1, \dots, m; t = 1, \dots, T\}$; 2-层学习算法 $\tau_1, \tau_2, \dots, \tau_{T_1}$.

- 1: 同样将数据集 D' 划分为 k 个大小相似的互斥子集:
 - 2: for $j = 1, \dots, k$ do
 - 3: for $t' = 1, \dots, T_1$ do
 - 4: $g_{t'}^{(j)} = \tau_{t'}(D' \setminus D'_j)$
 - 5: end for
 - 6: for $\mathbf{x}'_i \in D'_j$ do
 - 7: $\mathbf{z}'_{it'} = g_{t'}^{(j)}(\mathbf{x}'_i)$
 - 8: end for
 - 9: end for
 - 10: 将 $g_{t'}^{(j)}(\mathbf{x}'_i)$, $i = 1, \dots, m$, $t = 1, \dots, T$ 表示 $m \times T$ 个成(4-4)式的矩阵 $DT(\mathbf{x}'_i)$
 - 11: 按式(4-5)按 DT 矩阵的列求平均得到 $\bar{\mathbf{z}}'_i$, $i = 1, \dots, m$, $t = 1, \dots, T$
-

输出: $D'' = \{(\mathbf{x}''_i, y_i) \mid i = 1, \dots, m; t = 1, \dots, T\}$; 分类器 $g_{t'}^{(j)}$, $j = 1, \dots, k$, $t' = 1, \dots, T_1$.

图 4-3 2-层训练过程

输入: 1-层得到的测试集 $\{(\bar{h}_1(\mathbf{x}), \mathbf{x}), (\bar{h}_2(\mathbf{x}), \mathbf{x}), \dots, (\bar{h}_T(\mathbf{x}), \mathbf{x})\}$;

训练好的 2-层分类器 $g_{t'}^{(j)}$, $j = 1, \dots, k$, $t' = 1, \dots, T_1$.

- 1: 用 $g_{t'}^{(j)}$ 对 1-层得到的测试集进行预测, 得到它们的类概率向量 $g_{t'}^{(j)}(\bar{h}_t(\mathbf{x}), \mathbf{x})$
 - 2: 平均得到: $\bar{g}'_t(\bar{h}_t(\mathbf{x}), \mathbf{x})$, $t = 1, \dots, T$, $t' = 1, \dots, T_1$
 - 3: 将 $\{\bar{g}'_t(\bar{h}_t(\mathbf{x}), \mathbf{x}) \mid t = 1, \dots, T; t' = 1, \dots, T_1\}$ 按 2-层分类器进行累加求平均: 式(4-6),
 - 4: 原始特征 \mathbf{x} 组合得到集合: 式(4-7)
-

输出: $\left\{ \left(\mathbf{x}, \sum_{t'=1}^{T_1} \bar{h}'_{t'}(\bar{h}_t(\mathbf{x})) \right) \mid t = 1, \dots, T \right\}$

图 4-4 2-层测试过程

3-层训练和测试过程：用训练集 D'' 训练 ζ ，得到分类器 h'' ：

将 $\left\{ \left(\mathbf{x}, \sum_{t'=1}^{T_1} \bar{h}_{t'}(\bar{h}_t(\mathbf{x})) \right) \middle| t=1, \dots, T \right\}$ 输入 h'' 中，预测这 T 个向量的标签值 $\bar{y}_t(\mathbf{x}) = h'' \left(\left(\mathbf{x}, \sum_{t'=1}^{T_1} \bar{h}_{t'}(\bar{h}_t(\mathbf{x})) \right) \right)$, $t=1, \dots, T$ 。因为这 T 个向量是测试样例 \mathbf{x} 经过 1-层和 2-层基分类器预测、运算产生的，所以要对这 T 个预测的类标签 $\bar{y}_t(\mathbf{x})$, $t=1, \dots, T$ 进行投票，得出原始测试样例 \mathbf{x} 的最终预测标签值 $H(\mathbf{x})$ 。

输入：3-层训练集 $D'' = \{\mathbf{x}_i'', y_i \mid i=1, \dots, m; t=1, \dots, T\}$ ；3-层学习算法 ζ 。

1：用 D'' 训练得到 3-层分类器： $h'' = \zeta(D'')$

输出：3-层的元分类器 h''

图 4-5 3-层训练过程

输入：3-层的元分类器 h'' ；

测试集 $\left\{ \left(\mathbf{x}, \sum_{t'=1}^{T_1} \bar{h}_{t'}(\bar{h}_t(\mathbf{x})) \right) \middle| t=1, \dots, T \right\}$ 。

1：用 h'' 预测测试集中 T 个测试样例的类标签 $\bar{y}_t(\mathbf{x}) = h'' \left(\left(\mathbf{x}, \sum_{t'=1}^{T_1} \bar{h}_{t'}(\bar{h}_t(\mathbf{x})) \right) \right)$, $t=1, \dots, T$

2：对 T 个预测的类标签 $\bar{y}_t(\mathbf{x})$ 用投票法决定测试样例 \mathbf{x} 的最终标签 $H(\mathbf{x})$

输出： \mathbf{x} 的最终类标签 $H(\mathbf{x})$

图 4-6 3-层测试过程

为了更加形象具体的表示改进的 Stacking 算法的过程，本文用具体的原始输入向量维数和类别数具体的表示改进的 Stacking 算法的训练和预测过程。假定原始特征向量是 4 维的，对数据进行的是二分类任务，1-层和 2-层基分类器个数分别为 T 和 T_1 。图 4-7 表示了模型的训练预测过程，黑色阴影箭头表是训练过程，空白箭头表示对测试样例的测试过程。图 4-7 中的符号和图 4-1 到图 4-6 这三个图中的符号是一致的。

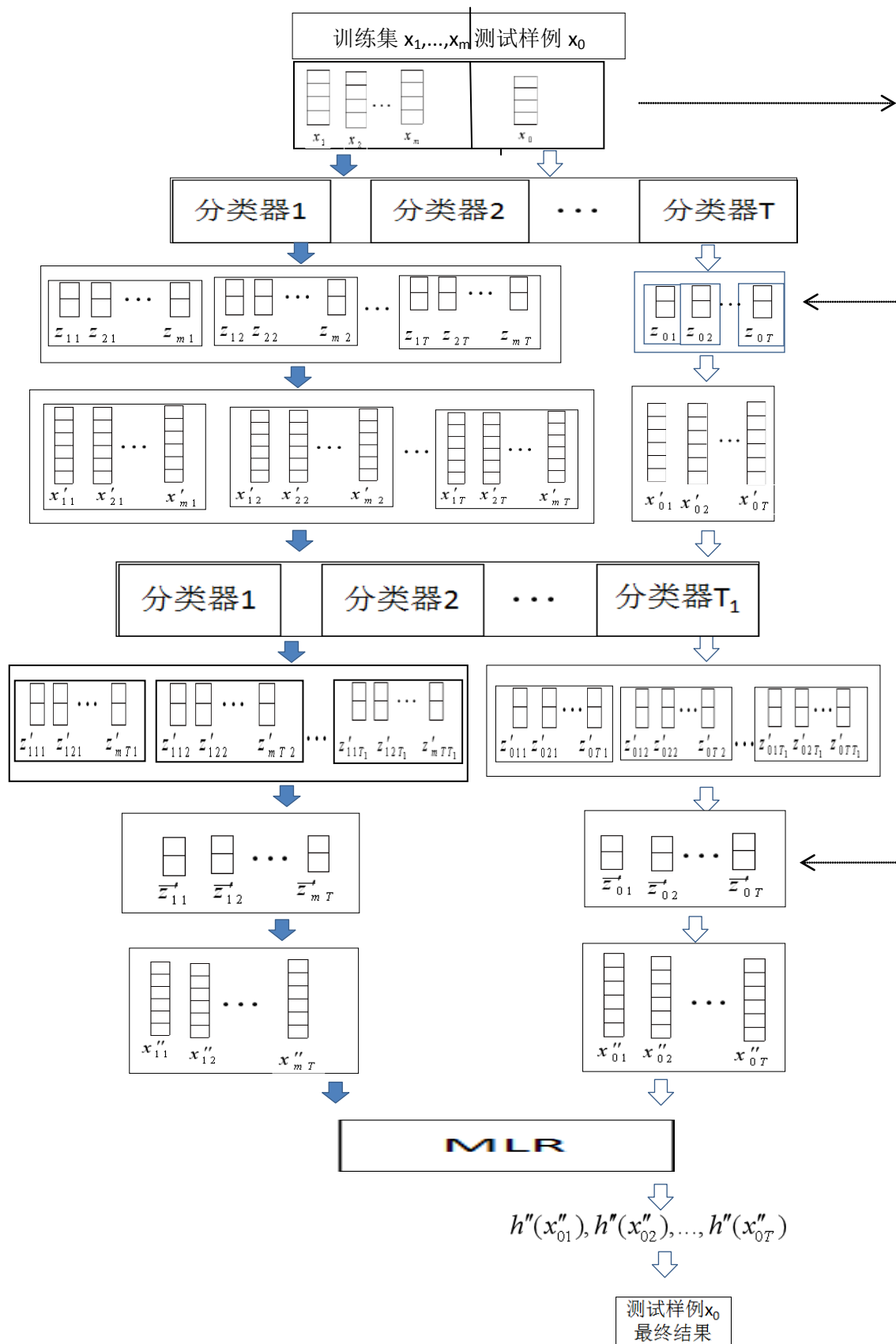


图 4-7 改进的 Stacking 算法框架图

4.2 实验设置

为了确定本文提出算法的有效性，本章在 UCI 数据集上做出如下实验：

(1)对分类结果的影响。在改进的 Stacking 算法中，在 2-层和 3-层中均加入了原始样本的特征向量，如果在 2-层和 3-层中去掉原始特征，对分类结果会有影响吗？(2)层数对分类结果的影响。去掉改进的 Stacking 算法的第二层，或者是加入和第二层模块一样的模块对分类结果有影响吗？(3)StackingM 算法和组成它的基分类器的相对性能进行比较。StackingM 算法和其它方法的比较。

4.2.1 实验数据集及参数设置

表 4-1 实验数据集

数据集	样本数	类别数	特征数	分布情况	UR
iris	150	3	4	(50,50,50)	1
wine	178	3	13	(59,71,48)	1.47
glass	214	6	9	(70,76,13,29,9,17)	8.44
ecoli	336	8	7	(143,77,2,2,35,20,5,52)	35.75
segment	2310	7	18	(330, 330, 330, 330, 330, 330, 330)	1
arrhythmia	452	13	279	(245,44,15,15,13,25,3,2,9,50,4,5,22)	61.25
ionosphere	351	2	34	(225,126)	1.79
page-blocks	5473	5	10	(4913,329,28,88,115)	175.46
transfusion	748	2	4	(571,177)	3.23
pendigits	7493	10	16	(780,779,780,719,780,720,719,778,719,719)	1.08
seeds-dataset	210	3	7	(70,70,70)	1
auto-mpg	398	3	7	(249,70,79)	3.56
beast	699	2	9	(458,241)	1.90
contraceptive	1473	3	9	(629,333,511)	1.89
semeion	1593	2	249	(1435,158)	9.08

为了比较不同结合算法的性能，我们在 15 个 UCI 数据集上做实验，这些数据包括 4 个平衡数据集和 11 个非平衡数据集，其中有 4 个数据集是二分类任务。表 1 是实验所使用的数据及其相关的具体信息，包括样本总数，类别数，特征维度，类别分布情况和不平衡率 UR(Unbalanced Ratio)。其中不平衡率 UR 是数据集中，多数类的样本集合 M 大

小和少数类样本集合 S 大小的比例: $UR = \frac{|M|}{|S|}$ 。实验中基分类器采用的算法包括, RF(10 trees), CRF(10 trees), GBDT (10 trees)。所有的实验结果均是 10 次 5 折交叉验证得到的平均结果, 每次交叉验证用不同的随机种子。

4.2.2 实验结果比较及分析

为了比较改进算法的有效性, 用于比较的结合算法包括以下 7 种。VOTE 投票法^[11]; SelectBest: 组成集成的个体分类器 RF, CRF, GBDT 中最好的一个个体分类器算法, 它是通过交叉验证选择的最好的单一分类器; SMLR: 元层分类器为多响应线性回归, 元层的属性为概率分布(SMLR); StackingM: 改进的 Stacking 算法; S2L(Stacking2level): 去掉改进的 Stacking 算法的第二层; S4L(Stacking4level): 加入和改进的 Stacking 算法的第二层模块一样的模块, 它对前面一层的输出结果的处理方式也是平均; StackingM-F: 将原始输入特征(feature)向量从改进的 Stacking 算的 2-层和 3-层输入向量中去掉。

为了对每个算法在 15 个数据上的整体性能有一个清晰的了解, 我们在给定的度量上采用赢输 W/d/l (win/draw/loss) 记录来比较模型, 它们分别代表算法 a 在给定的度量上在多少个数据集上优于、相当、较差于算法 b 。信号测试中常用这种方法, 从中可以估计信号得到的概率, 或者从中可以了解到这种信号是偶然得到的。如果信号测试结果极为低, 这种输出有可能是偶然发生的, 结果有可能并不合理, 因此这种赢和损失的记录可以表明算法在测试集的优劣。

表 4-2 表头中的 3 个个体分类器表明: StackingM 中 1-层和 2-层的个体分类器个数均为 3, 且每一层都是 1 个 RF, 1 个 CRF, 1 个 GBDT; Stacking2level、VOTE 和 SMLR 算法的个体分类器数为 6, 即 2 个 RF, 2 个 CRF, 2 个 GBDT。表 4-3~表 4-7、图 4-5 和附录中 5 个个体分类器表明: StackingM 中 1-层和 2-层的个体分类器个数均为 5, 且每一层都是 2 个 RF, 2 个 CRF, 1 个 GBDT; Stacking2level、VOTE 和 SMLR 算法的个体分类器数为 10, 即 4 个 RF, 4 个 CRF, 2 个 GBDT。

表 4-2 和表 4-3 表示的就是每对模型关于准确率的 w/l 记录: 行算法相对于列算法在准确率相对提高率上的赢 / 输数据集个数记录。定义算法 a 相对于算法 b 的准确率提

高率为: $\left(\frac{ACC_a}{ACC_b} - 1\right) \times 100\%$, 其中 ACC_a 和 ACC_b 分别表示算法 a 和算法 b 在数据集上的准确率。本文中, 当相对提高率大于 1% 时, 认定算法 a 为 win; 小于 -1 时认定算法 a 为 loss, 大于等于 -1 且小于等于 1 时认为算法 a 和算法 b 性能相当。

表 4-2 3 个个体分类器的不同集成算法关于准确率的 w/l 记录

	VOTE	SelectBest	SMLR	Stacking2level	StackingM
VOTE	-	1/5	2/7	5/4	1/8
SelectBest	5/1	-	4/5	7/3	1/4
SMLR	7/2	5/4	-	7/1	2/5
Stacking2level	4/5	2/7	1/7	-	1/8
StackingM	8/1	4/0	5/2	8/1	-

表 4-3 5 个个体分类器的不同集成算法关于准确率的 w/l 记录

	VOTE	SelectBest	SMLR	Stacking2level	StackingM
VOTE	-	3/6	1/6	5/3	0/7
SelectBest	6/3	-	3/7	5/4	0/8
SMLR	6/1	7/3	-	4/0	3/5
Stacking2level	3/6	4/5	0/4	-	1/6
StackingM	7/0	8/0	5/3	6/1	-

表 4-2 和表 4-3 中虽然集成的个体分类器个数不同, 但是它们共同表现出: SMLR 赢得的数据个数明显比 VOTE、SelectBest、Stacking2level 多, 这表明 SMLR 比本文所说的 VOTE, SelectBest 还有 Stacking2level 好; StackingM 算法和其它 4 个方法在 15 个数据集上的 w/l 记录表明, StackingM 算法在 15 个 UCI 数据集上整体优于其它方法, 因此说明, Stacking 的性能通常优于其它四种方法。表 4-2 和表 4-3 中, StackingM 和 SMLR 的 w/l 记录 6/2 和 5/3 表明 StackingM 算法明显优于 SMLR。表 4-2 和表 4-3 中 StackingM 和 Stacking2level 的记录 5/2, 6/1, 表明 StackingM 算法中第 2 层是必要的, 它可以提高分类准确率。在判断 StackingM 性能上, 和组成他的最好的分类器 SelectBest 比较是基本的。表 4-2 和表 4-3 中 StackingM 和 SelectBest 的 w/l 记录 4/0, 8/0, 表明集成算法 StackingM 优于组成他的最好的基分类器, 说明集成是有效的。

表 4-4、表 4-5 和表 4-6 是分别是 7 个方法在 15 个数据上的查准率(precision)和 F1

值和准确率(accuracy)。由于非平衡数据较多,对于多分类数据来说,这里用 macro-F1 和 macro-P 来评估分类器的性能。表 4-4、表 4-5 和表 4-6 的加粗数据表示 StackingM 算法和 SMLR 算法在每个数据集中的最高度量值(当一个算法 a 的度量值高于另一个算法 b 的度量值超过 1%时,我们才对相应的度量值进行加粗)。

对比表 4-4 中 StackingM 算法和 MLR 算法的查准率,从加粗数据可以明显看出,StackingM 在 4 个数据上明显优于 SMLR,同时在 4 个数据 *ecoli*, *arrhythmia*, *page-blocks*, *transfusion* 上 F1 性能没有 SMLR 好,可以看出 *ecoli*、*Arrhythmia* 和 *page-blocks* 的非平衡率分别为 37.75, 61.25, 175.4, 他们是这 15 个数据中不平衡率最高的三个数据。在 7 个数据上性能相当,但是在性能相当的 7 个数据上,除了在大样本平衡数据集 *pendigits* 数据集上 StackingM 的 precision 比 SMLR 小 0.41%,在其他数据上均高于 SMLR。

表 4-4 5 个个体分类器的不同集成算法的查准率(%)比较

数据集	VOTE	SelectBest	SMLR	S4L	S2L	StackingM-F	StackingM
auto-mpg	77.93	80.44	77.65	66.96	75.26	74.38	80.76
ecoli	55.15	62.29	64.07	52.99	61.46	57.42	63.03
glass	68.79	73.76	69.53	65.09	64.55	63.07	76.85
arrhythmia	27.61	43.79	43.86	41.79	27.04	32.89	40.31
ionosphere	93.23	93.02	93.55	85.68	93.39	91.80	93.36
page-blocks	89.13	88.49	87.65	64.34	88.10	88.90	86.55
contraceptive	48.77	49.63	51.70	48.08	53.71	44.00	51.74
breast	98.26	96.97	96.14	93.21	97.79	95.89	98.66
pendigits	98.70	98.29	99.50	97.44	99.07	98.86	99.09
transfusion	58.53	41.07	57.26	58.48	68.32	38.10	52.04
segment	97.86	97.10	97.70	96.20	97.86	96.88	97.77
iris	94.27	94.26	95.31	93.96	92.30	95.39	96.96
wine	97.74	97.74	98.84	96.67	98.84	98.83	98.84
seeds-data	93.97	92.86	93.33	92.00	92.06	92.28	93.65
semeion	96.89	96.22	76.08	95.20	86.58	95.46	97.39

对比表 4-5 中 StackingM 算法和 SMLR 算法的准确率,从加粗数据可以明显看出,StackingM 在 5 个数据上明显优于 SMLR。在 2 个数据上准确率没有 SMLR 好,在 8 个数据上性能相当。其中在 *auto-mpg* 数据上,StackingM 的 macro-F1 比 SMLR 提高了 2.71%;在 *glass* 数据上提高了 2.56%。这都是小样本的非平衡数据。

表 4-5 5 个个体分类器的不同集成算法的准确率(%)比较

数据集	VOTE	SelectBest	SMLR	S4L	S2L	StackingM-F	StackingM
auto-mpg	83.17	84.19	83.15	72.65	81.91	81.22	85.86
ecoli	79.24	83.55	85.54	69.15	85.04	81.29	84.42
glass	74.42	76.28	75.49	70.52	71.96	75.30	78.05
arrhythmia	63.05	66.20	68.70	53.08	55.96	64.51	69.94
ionosphere	92.86	92.70	93.04	86.03	90.28	92.30	93.20
page-blocks	97.20	97.42	97.28	75.35	97.17	97.20	97.44
contraceptive	50.92	51.93	53.70	48.82	53.97	47.92	53.16
breast	96.99	95.99	95.12	93.57	96.00	96.51	96.84
pendigits	98.67	98.91	99.49	97.29	99.07	98.85	99.08
transfusion	77.27	73.80	77.06	55.02	76.88	76.56	76.74
segment	97.04	97.06	96.84	95.97	97.14	97.71	97.75
iris	94.00	94.00	95.33	92.00	94.67	95.00	94.21
wine	97.81	97.11	98.86	96.66	98.86	98.70	98.86
seeds-data	93.97	92.16	93.33	90.95	92.70	91.43	93.65
semeion	94.85	96.61	95.35	96.87	95.23	96.08	97.17

表 4-6 5 个个体分类器的不同集成算法的 F1 值比较(%)

数据集	VOTE	SelectBest	SMLR	S4L	S2L	StackingM-F	StackingM
auto-mpg	75.20	76.02	76.27	76.81	74.23	76.25	79.74
ecoli	50.04	60.22	62.44	60.12	61.14	60.92	61.39
glass	64.44	71.54	66.67	62.70	61.23	72.11	73.58
arrhythmia	25.52	36.96	37.11	40.45	27.10	28.44	36.33
ionosphere	92.04	91.78	92.19	85.23	94.07	91.60	92.47
page-blocks	84.19	85.64	83.55	64.55	82.51	81.99	86.05
Contraceptive	47.55	49.32	50.41	47.47	53.35	35.81	50.95
breast	97.69	96.94	96.30	93.30	96.91	96.18	97.56
pendigits	98.69	98.10	99.50	97.43	99.07	98.86	99.09
transfusion	27.82	30.50	31.20	53.25	53.78	43.24	41.26
segment	97.84	97.04	97.01	95.94	97.84	96.84	97.75
iris	93.98	93.28	95.31	91.75	89.76	93.94	96.65
wine	97.80	97.80	98.90	96.72	98.90	98.89	98.90
seeds-data	93.97	92.86	93.33	90.78	92.06	91.39	93.65
semeion	64.92	79.45	76.92	87.20	86.85	86.99	83.42

对比表 4-6 中 StackingM 算法和 SMLR 算法的 F1 值,从加粗数据可以明显看出,StackingM 在 6 个数据上优于 SMLR。在 2 个数据上 F1 性能没有 SMLR 好,在 7 个数据上性能相当。其中在 auto-mpg 数据上,StackingM 的 macro-F1 比 SMLR 提高了 2.47%;在 glass 数据上提高了 6.91%,在 transfusion 上提高 10.06。

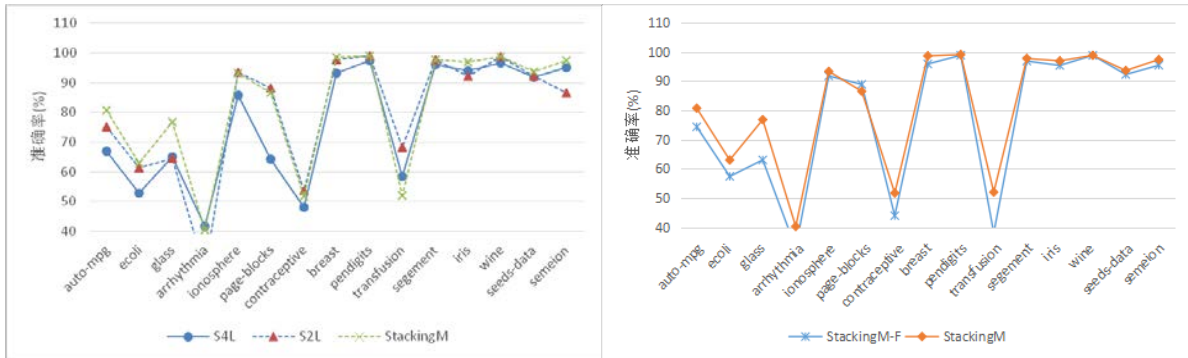


图 4-8 集成算法在 15 个数据上的准确率比较(%)

在表 4-4、表 4-5 和表 4-6 中通过 StackingM 和 S4L 的比较发现, S4L 在查准率,准确率和 F1 值上表现较差于 StackingM, 又由表 4-3 中 StackingM 和 S2L 的 w/l 记录 6/1, 实证表明了本文改进算法的三层结构是合理的。且表 4-4、表 4-5 和表 4-6 的数据表明改进 Stacking 算法 StackingM 的有效性, 特别在某些小样本非平衡数据集上。从图 4-8 可以明显看出 StackingM 在准确率整体上较高于 4 层的 S4L 和 2 层的 S2L, 图 4-8 中表示 StackingM 在 15 个 UCI 数据集上的表现整体上比不加原始特征的 3 层 StackingM-F 好。

表 4-7 5 个基分类器集成的 SMLR 和 StackingM 在每个类别上的预测准确率(%)

Accuracy	方法	第一类	第二类	第三类	第四类	第五类	第六类
glass	SMLR	73.92	75.69	75.47	92.96	75.33	23.80
	StackingM	76.02	79.69	77.67	91.92	89.44	46.33
auto-mpg	SMLR	91.13	69.05	72.79	-	-	-
	StackingM	93.19	74.34	74.74	-	-	-
breast	SMLR	96.14	93.36	-	-	-	-
	StackingM	98.65	93.73	-	-	-	-

为了表明 StackingM 在小样本非平衡数据集上的较大优越性。表 4-7 给出了算法 StackingM 和 SMLR 在 3 个非平衡数据 glass, auto-mpg, breast 中每个类的准确率。符号“-”表示没有此类别。特别在 glass 和 auto-mpg 的少数类上, StackingM 的准确率有明显提高, 这表明 StackingM 算法的 1-层个体分类器在某种程度上可以有效增加样本数,

使得 2-层和 3-层的训练数据增加，以提高分类性能。

以上的数据分析表明改进的算法 **StackingM** 在三种性能度量上均表现出较好的优势，但在实际分类任务中，我们不仅要考虑结果的好坏，还要考虑时间效率。 m 为样本个数， T 为个体分类器的个数。假定个体学习器的计算复杂度为 $O(m)$ 。则 **Stacking** 算法的复杂度大致为 $T \times O(m) + O(s)$ ， $O(s)$ 为 **Stacking** 算法的最后一层元学习层的计算复杂度。本文将维度的增长扩展到样本的增长上，所以个体分类器的计算复杂度会减少，进而整体的计算时间会减少。图 4-9 是 **StackingM** 算法和 **SMLR** 算法在 15 个数据上的运行时间比较：虽然 **StackingM** 的结构是一个三层的结构，比 **SMLR** 增加了一层，但图 4-9 表示 **StackingM** 算法相对于 **SMLR** 算法，计算效率有所提高。在 279 维 452 个样本的 13 类数据集 **arrhythmia** 和 10 维 5473 个样本的 5 类数据集 **page-blocks** 上，**StackingM** 的时间比 **SMLR** 的时间少了。**arrhythmia** 的维度为 13，对于 **SMLR** 算法，每增加一个分类器，元层学习器的维度就增加 13 维，这大大增加了训练时间；对于 **StackingM** 算法，1-层每增加一个分类器，元层的维度仍保持原始训练集的特征数加上类别数，但是元层的训练集仅会增加 452 个样本，所以并没有高维数据更加需要时间。从图 4-9 中也注意到，在 249 维 1593 个样本的 2 分类数据集 **semeion** 上，**StackingM** 消耗的时间会更长，这是因为对于 **SMLR** 算法，每增加一个分类器，元层学习器的维度仅增加 2 维；而对于 **StackingM** 算法，1-层每增加一个分类器，元层的训练集会增加 1593 个样本，样本的增加速度远远大于维数的增加速度。所以 **StackingM** 在大样本低类别的数据集上的时间会较长，这里的大样本是样本数相对类别数，同样低类别是类别数相对于样本数。

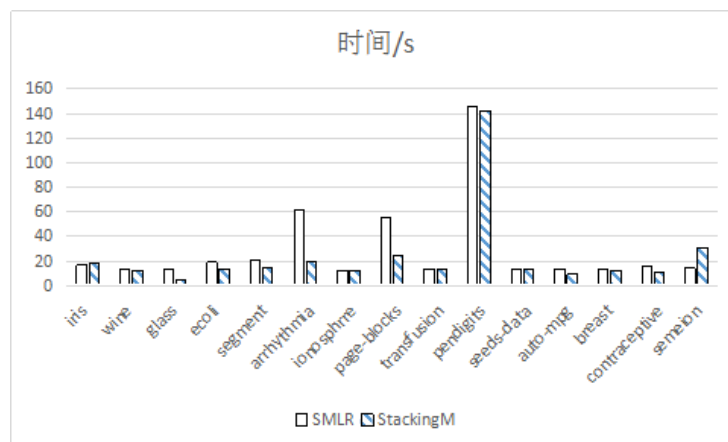


图 4-9 StackingM 算法和 SMLR 算法在 15 个数据上的运行时间比较

在 15 个 UCI 数据集上的实验结果表明, 在改进的 Stacking 算法 StackingM 中如果去掉原始特征, 分类性能整体会稍微降低。当去掉 StackingM 的第二层时, 分类性能在大部分数据集上会降低。当在 StackingM 上增加一层时, 分类性能整体会降低。将 StackingM 和其他集成方法比较, 结果显示, StackingM 的分类性能较好于 SMLR 和 VOTE, 且 StackingM 运行时间在大部分的数据集上比 SMLR 算法低。且 StackingM 在小样本不平衡数据上效果大大提高。

4.3 改进的 Stacking 算法的人脸识别

4.3.1 数据描述及参数设置

为了进一步验证 StackingM 算法的实际有效性, 特别是 1-层的个体分类器能够有效的产生训练样本, 本节基于真实的 ORL 图像数据集进行人脸识别。ORL 数据集包含了 40 个人的 400 幅灰度人脸图像, 每个人都有不同的 10 张人脸图像, 每幅图像大小为 112×92 像素, 每个像素是 256 级灰度。因此每幅图像可表示成 10304 维向量。由此可知, 该数据集有 40 类, 每类有 10 个样本。用本文所提算法 StackingM, 文献[34]提出的 Stacking 算法 StackingTier, 还有 CNN(卷积神经网络), 1000 棵树的标准随机森林进行对比实验。其中 StackingM 中 1-层个体分类器个数为 5, 分别是 3 个 RF, 1 个 CRF, 1 个 GBDT; 2-层个体分类器个数为 3, 是 3 个 RF; 其中个体分类器 RF, CRF, GBDT 的决策树个数均为 100 个; 文献[34]中的 Stacking 方法采用了一层初级层和三层融合层, 本文将它第一层的个体分类器个数定为 5, 分别是 3 个 RF, 1 个 CRF, 1 个 GBDT; 第二层用 1 对 1 的多响应线性分类器进行学习, 将第二层 1 对 1 线性分类器输出的关于每个类的概率值输入第三层的个体分类器中, 第三层个体分类器个数为 $l-1$, 均是 RF, 它们输出的是每个类的类概率值, 第四层是超级分类器逻辑回归, 对上一层输出的概率值进行学习并与测最终的类标签。对文献[42]中 CNN 的参数是经过实验选择的较优参数, 所以本文用于比较的 CNN 参数和文献[42]中的参数一样: 两个卷积层的过滤器尺寸为 3×3 , 深度为 32, 每个卷积层后有一个 2×2 的最大池化层, 最后有 128 个隐藏单元和 40 个隐藏单元的两个全连接层, ReLU 为激活函数, 使用交叉熵损失, dropout 率为 0.25, 而且引入了自适应学习率。我们随机选择每个人的 5、8、9 张图片为训练集, 在表 4-8 和图 4-6 中分别表示为 5 images、8 images、9 images 在余下的数据集上进行测试, 测试

结果是 10 次运行的平均结果。

4.3.2 实验结果与分析

灰度图像数据集 ORL 是平衡的人脸识别数据集，我们在准确率上评估算法在 ORL 数据集上的分类性能。

表 4-8 在 ORL 数据集上的准确率(%)比较

分类算法	5 images	8 images	9 images
StackingM	94.50	97.50	98.50
RF	91.25	94.50	96.75
CNN	88.21	93.96	96.28
StackingTier	90.57	94.00	96.83

当随机选择每个人的 5 张图片为训练数据时，训练数据和测试数据均为 200 个图像，图 4-10 的 5 images 折线表明，此时 StackingM 在准确率上明显优于其他 3 个算法，通过对表 4-8 的数据计算，得出 StackingM 是比 CNN 提高 6.29%。当训练集增多时，虽然 StackingM 算法仍然占有很大的优势，但是和其他算法的差距明显减少：随机选择每个人的 8 张图片为训练数据时，比 CNN 提高 3.54%；随机选择每个人的 9 张图片为训练数据时，比 CNN 提高 2.22%。这说明 StackingM 的 1-层能够有效的增加数据，表明了提出算法的实际应用性。

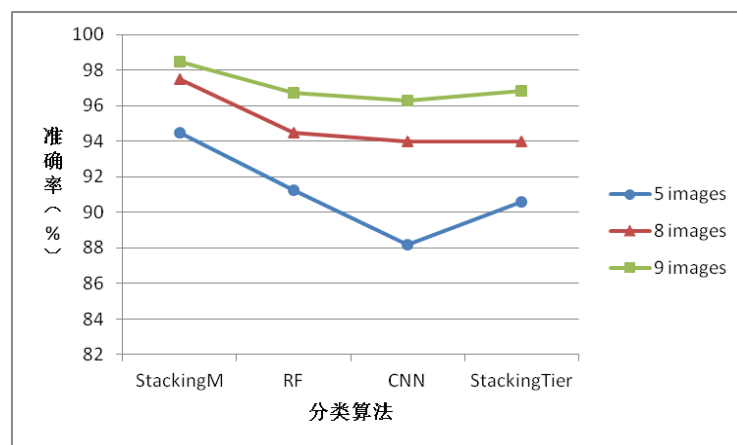


图 4-10 在 ORL 数据集上的准确率(%)比较

4.4 遗传算法优化结构

4.4.1 遗传算法优化结构

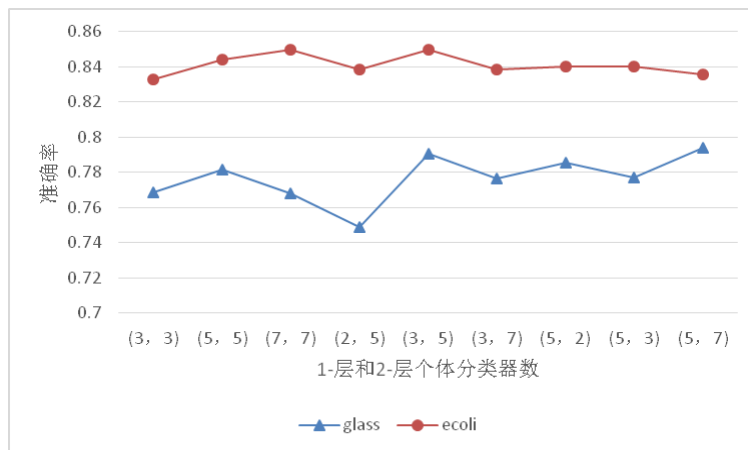


图 4-11 1-层和 2-层不同个体分类器数组合的 StackingM 算法在 glass 数据上的准确率

表 4-1 到表 4-7 的实验结果也表明，StackingM 在 15 个 UCI 数据集上就表现出了其优势。但是图 4-11 表示并不是基分类器越多越好。图 4-11 中的横坐标表示的是 1-层和 2-层基分类器的个数对，纵坐标表示的是在 glass 和 ecoli 上的分类准确率。折线代表准确率随着不同的基分类器组合方式的改变而改变。

接下来的实验主要研究解决：基分类器数量对改进算法 StackingM 性能的影响。10 次 5 折交叉验证用于计算分类器的准确率。图 4-11 的横坐标表示的是 1-层和 2-层基分类器的数量组合，纵坐标表示的是 StackingM 算法在数据 glass 上的准确率。折线图 4-11 表示 1-层，2-层的分类器数组合对 StackingM 算法分类结果的影响较大。且并不是分类器总数越高，分类效果就越好。在 StackingM 中面临一个和 SMLR 不同的分类器配置问题：对于同样的 T 个基分类器，它的顺序即是 SMLR 中元层属性的数序，这些对 SMLR 的分类是没有影响的。但是对于 StackingM，这 T 个基分类器会被分为两层，那 1-层和 2-层分别用这 T 个分类器的哪些分类器对分类结果肯定是有影响的，在表 4-1 到表 4-7 的实验中，我们只是将所有集成算法的初级层的基分类器个数均设定为 3 个和 5 个。这只是为了方便表示提出的算法在没有调节超参数的情况下一样能表现更好。为了进一步增加算法在每个不同数据上的性能，我们用遗传算法来调节 StackingM 的超参数：分类器的配置。

遗传算法是一种简单的智能优化算法，本文用遗传算法来组合 StackingM 的 1-层和 2-层分类器。且考虑到训练和预测时间，我们将每基分类器的最大数为 9，即 3 个 RF，3 个 CRF，3 个 GBDT。

9 个基分类器表示为 $BL=[RF1,RF2,RF3,CRF1,CRF2,CRF3,GBDT1,GBDT2,GBDT3]$ 我们采用二进制编码来操作这 9 个基分类器，一个个体表示为大小为 $2 \times |BL|$ 向量 $\vec{v}=(\vec{v}',\vec{v}'')$ ，如果基分类器 BL_i 属于 1-层集成的话， $\vec{v}'_i=1$ ，否则的话 $\vec{v}'_i=0$ ，二层类似。初始种群为 5，最大迭代次数为 200。种群由变异概率 P_m 来确定变异点，由交叉概率 P_c 来确定交叉点，个体 $personal=\vec{v}$ 和个体 $personal^1=\vec{v}^1$ 在 i 点交叉的输出个体为 $(\vec{v}_1,\vec{v}_2,...,\vec{v}_i,\vec{v}_{i+1}^1,...,\vec{v}_{2 \times |BL|}^1)$ ，即单点交叉。适应函数是在验证集上的分类准确率。

4.4.2 实验结果及分析

表 4-9 是人工设计的参数（1-层和 2-层分类器的个数均为 5 个：2 个 RF，2 个 CRF，1 个 GBDT）和遗传算法调节的参数得到分类器的准确率值。表中的数据表明遗传算法可以有效的对 StackingM 的参数进行选择，这说明在实际中，可以通过遗传算法代替人工对 StackingM 进行优化配置。

表 4-9 遗传算法和人工调参的 StackingM 的准确率(%)

数据集	人工结果	遗传算法结果
Wine	97.81	98.01
Glass	78.21	79.18
Ecoli	84.41	84.23
Ionosphere	93.75	93.93
Page-blocks	97.44	97.21
breast	96.25	96.80
Contraceptive	53.25	52.70

4.5 本章小结

本章首先提出了改进的 Stacking 算法 StackingM 的思想，第一节先分别介绍了改进的三个点：2-层和 3-层个体学习器的输入属性表示；基分类器的选择；结合方法，并一

一详细介绍了改进的原因。最后从整体上描述了算法的训练和预测过程。为了验证改进算法的合理性和有效性，第二节通过在 15 个 UCI 数据集上进行实验验证，将改进算法和两层 Stacking 算法和三层 Stacking 算法作比较，结果表明改进的三层 Stacking 算法的合理性。将改进的 Stacking 算法和 SMLR 作比较，实验结果表明改进的方法提高了分类性能，且算法的运行时间也比现有的 SMLR 算法运行时间在大部分数据集上少。第三节将 StackingM 算法和最近的集成方法和卷积神经网络 CNN 在 ORL 数据集上做了对比实验，实验表明 StackingM 的实际可用性。第四节用遗传算法对改进算法进行优化配置，实验表明也可以用遗传算法对改进的算法进行超参数调节。

总结与展望

基于目前现有的 Stacking 算法的分析,了解到用概率分布代替类标签能够很好的表示个体分类器,而多响应线性回归能够很好的学习类概率。基于这两点,我们提出了三层的改进的 Stacking 算法。对结合方法和层级之间的属性表示做出改进:通过一层个体分类器输出的类概率向量来增加二层训练数据的样本数,且使得二层个体分类器的训练数据的维度不会随着分类器的增加而增大;通过二层个体分类器的学习减少一层个体学习器的输出类概率向量中带来的噪声;第三层用多响应线性回归对前面的预测进行再次学习;在预测时加入投票策略层来预测样本的最终类别;在这个三层结构中,在第二层和第三层的输入属性中加入了原始特征。因为三层 Stacking 算法中每层的基分类器对改进的 Stacking 算法有影响,所以为了进一步优化改进的 Stacking 算法的结构,用遗传算法对个体分类器组合进行优化。实验结果表明,本文给出的方法在 ORL 灰度图像和给定的 15 个 UCI 数据集上有较好的分类结果,特别在非平衡数据集上,这表明了改进的算法的有效性。

本文的主要工作总结如下:

(1) 阐述了本课题研究背景和发展现状。介绍了一些常见的集成方法和常使用的 Stacking 方法。

(2) 针对现有的 Stacking 算法中存在的元层输入属性维度逐渐变大和运行时间长的问題,提出了三层改进的 Stacking 算法,改进的算法主要在每层的输入属性表示和最终的结合方法上做了改进。在 15 个 UCI 数据集上,将改进的算法和其他 Stacking 算法、两层的 Stacking 算法、四层的 Stacking 算法和不加入原始特征的三层 Stacking 算法做了对比性能实验,验证了改进的 Stacking 算法的合理性,且在分类性能度和运行时间上均表现较优。最后在真实的灰度图像数据集上验证了改进算法的实际可用性。集成学习常和优化算法结合在一起,本文单独用一小节对遗传算法优化改进的 Stacking 算法做了简单的实验,表明遗传算法优化 Stacking 中分类器的配置的可行性。

本文的不足和下一步的工作:

(1) 本文的算法在 15 个不同类型的 UCI 数据集和高维小样本图像数据集上做了实验研究,以后可以用来解决大规模数据的分类问题。

(2) 2-层的输入数据集中,可以通过相应的准则来选择部分 1-层输出向量。

参考文献

- [1] Mcnevin D, Santos C, Gomez-Tato, et al. An assessment of Bayesian and multinomial logistic regression classification systems to analyse admixed individuals[J]. Forensic Science International Genetics Supplement, 2013, 4(1):e63-e64
- [2] Dobra A. Decision Tree Classification[M]. Springer US, 2009
- [3] Kavzoglu T. Increasing the accuracy of neural network classification using refined training data[J]. Environmental Modelling & Software, 2009, 24(7):850-858
- [4] Overview I, Notes T, Bayes N, et al. Naive Bayes Classifier[J]. 2014
- [5] Adankon M.M., Cheriet M. Support Vector Machine[J]. Computer Science, 2002, 1(4): 1-28
- [6] Dietterich T.G. Machine-Learning Research; Four Current Directions[M]. 1997
- [7] Bohanec M, Cestnik B. A schema for using multiple knowledge[C]. The Workshop on Computational Learning Theory and Natural Learning Systems. MIT Press, 1994:157-170
- [8] Wolpert D.H. Stacked Generalization[M]. Springer US, 2011
- [9] 鲁莹, 郑少智. Stacking 学习与一般集成方法的比较研究[J]. 2017
- [10] Schapire R.E. The Strength of Weak Learnability[M]. Kluwer Academic Publishers, 1990
- [11] Breiman L. Bagging predictors[J]. Machine Learning, 1996, 24(2):123-140
- [12] Dietterich T.G. Ensemble Methods in Machine Learning[J]. Proc International Workshop on Multiple Classifier Systems, 2000, 1857(1):1-15
- [13] Cherkauer K.J. Human Expert-Level Performance on a Scientific Image Analysis Task by a System Using Combined Artificial Neural Networks[J]. Working Notes of the Aaaai Workshop on Integrating Multiple Learned Models, 1996:15—21
- [14] Ho T.K. The random subspace method for constructing decision forests[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 1998, 20(8):832-844
- [15] 胡小生. 改进随机子空间与决策树相结合的不平衡数据分类方法[J]. 佛山科学技术学院学报(自然科学版), 2013, 31(5):22-26

- [16]Dietterich T.G, Bakiri G. Solving multiclass learning problems via error-correcting output codes[J]. Journal of Artificial Intelligence Research, 1995, 2(1):263—286
- [17]Schapire R.E. Using output codes to boost multiclass learning problems[C]. Fourteenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc. 1997:313-321
- [18]Breiman L. Randomizing Outputs to Increase Prediction Accuracy[J]. Machine Learning, 2000, 40(3):229-242.2000
- [19]Kwok S.W, Carter C. Multiple decision trees[J]. Machine Intelligence & Pattern Recognition, 1990, 9:327-335
- [20]Breiman L. Random Forests[J]. Machine Learning, 2001, 45(1):5-32
- [21]Breiman L. Stacked regressions[J]. Machine Learning, 1996, 24(1):49-64
- [22]Kai M.T, Witten I.H. Issues in stacked generalization[M]. AI Access Foundation, 1999
- [23]Merz C.J. Using Correspondence Analysis to Combine Classifiers[M]. Kluwer Academic Publishers, 1999
- [24]Todorovski L, Dzeroski S. Combining Multiple Models with Meta Decision Trees[C]. European Conference on Principles of Data Mining and Knowledge Discovery. Springer -Verlag, 2000:54-64
- [25]Seewald A.K. How to Make Stacking Better and Faster While Also Taking Care of an Unknown Weakness[C]. chine Learning, Proceedings of the Nineteenth International Conference. DBLP, 2002:554-561
- [26]GzVeroski S,ZVenko B. Is Combining Classifiers with Stacking Better than Selecting the Best One?[J]. Machine Learning, 2004, 54(3):255-273
- [27]任惠. 基于 Stacking 框架的命名实体识别[D]. 大连理工大学, 2008
- [28]李寿山, 黄居仁. 基于 Stacking 组合分类方法的中文情感分类研究[J]. 中文信息学报, 2010, 24(5):56-61
- [29]李恒超, 林鸿飞, 杨亮, 等. 一种用于构建用户画像的二级融合算法框架[J]. 计算机科学, 2018(1):157-161
- [30]Moudrik J, Neruda R. Evolving Non-Linear Stacking Ensembles for Prediction of Go

- Player Attributes[C] Computational Intelligence, 2015 IEEE Symposium. IEEE, 2015:1673-1680
- [31] Alvear-Sandoval R.F, Figueiras-Vidal A.R. On building ensembles of stacked denoising auto-encoding classifiers and their furtherimprovement – A correction[J]. Information Fusion, 2017, 39:41-52
- [32] Demir N, Dalkılıç G. Modified stacking ensemble approach to detect network intrusion[J]. Turkish Journal of Electrical Engineering & Computer Sciences, 2018, 26(1)
- [33] 蒋翠清, 宋凯伦, 丁勇, 等. 基于用户生成内容的潜在客户识别方法[J]. 数据分析与知识发现, 2018, 2(3)
- [34] Eitan M, Lior R, Yuval E. An improved stacking schema for classification tasks[J]. Information Sciences, 2009, 179(24):4097-4122
- [35] Abawajy J.H, Kelarev A, Chowdhury M. Large Iterative Multitier Ensemble Classifiers for Security of Big Data[J]. Emerging Topics in Computing IEEE Transactions on, 2014, 2(3):352-363
- [36] 韦艳艳, 李陶深, 刘美玲. 融合 DECORATE 的异构分类器集成算法[J]. 计算机应用研究, 2012, 29(11):4134-4136
- [37] Zhou Z.H, Jiang Y. NeC4.5: neural ensemble based C4.5[J]. IEEE Transactions on Knowledge & Data Engineering, 2004, 16(6):770-773
- [38] Shunmugapriya P, Kanmani S. Optimization of stacking ensemble configurations through Artificial Bee Colony algorithm[J]. Swarm & Evolutionary Computation, 2013, 12(12):24-32
- [39] Chen Y, Man L.W. An Ant Colony Optimization approach for Stacking ensemble[C] Nature and Biologically Inspired Computing. IEEE, 2011:7-8
- [40] Chen Y.J, Wong M.L, Li H. Applying Ant Colony Optimization to configuring stacking ensembles for data mining[J]. Expert Systems with Applications, 2014, 41(6):2688-2702
- [41] Webb G.I. MultiBoosting: A Technique for Combining Boosting and Wagging[J]. Machine Learning, 2000, 40(2):159-196
- [42] Zhou Z.H, Feng J. Deep Forest: Towards An Alternative to Deep Neural Networks[J].

2017

- [43]何艳, 周丽丽, 杨喆. 基于 SVM-SOM 集成学习的故障诊断方法研究[J]. 自动化技术与应用, 2015, 34(10):28-33
- [44]张笑铭, 王志君, 梁利平. 一种适用于卷积神经网络的 Stacking 算法[J]. 计算机工程, 2018, 44(4):243-247
- [45]Lacy S.E, Lones M.A, Smith S.L. A comparison of evolved linear and non-linear ensemble vote aggregators[C]. Evolutionary Computation. IEEE, 2015:758-763
- [46]Kai M.T, Witten I.H. Issues in stacked generalization[M]. AI Access Foundation, 1999
- [47]Zenko B, Dzeroski S. Stacking with an Extended Set of Meta-level Attributes and MLR[C].European Conference on Machine Learning. Springer-Verlag, 2008:493-504
- [48]Smyth P, Wolpert D. Stacked density estimation[C]. Conference on Advances in Neural Information Processing Systems. MIT Press, 1998:668-674
- [49]Breiman L. Using Iterated Bagging to Debias Regressions[J]. Machine Learning, 2001, 45(3):261-277
- [50]Xu L, Krzyzak A, Suen C.Y. Methods of combining multiple classifiers and their applications to handwriting recognition[J]. IEEE Transactions on Cybernetics, 1992, 22(3):418-435
- [51]Bauer E, Kohavi R. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants[J]. Machine Learning, 1999, 36(1-2):105-139
- [52]Maclin R, Opitz D. Popular Ensemble Methods: An Empirical Study[J]. Journal of Artificial Intelligence Research, 2011, 11:169-198
- [53]Pratt W.K. Digital image processing[M]. John Wiley, 1978
- [54]Ross A, Jain A.K, Qian J.Z. Information Fusion in Biometrics[C]. International Conference on Audio- and Video-Based Biometric Person Authentication. Springer, Berlin, Heidelberg, 2001:354-359
- [55]Kittler J, Hatef M, Duin R.P.W, et al. On combining classifiers[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 1998, 20(3):226-239
- [56]Kittler J, Alkoot F.M. Sum versus vote fusion in multiple classifier systems[J]. Pattern

- Analysis & Machine Intelligence IEEE Transactions on, 2003, 25(1):110-115
- [57] Tsoumakas G, Vlahavas I. Distributed Data Mining of Large Classifier Ensembles[J].
Proceedings of Companion, 2002:249--256
- [58] F. Samaria and A. C. Harter. Parameterisation of a stochastic model for human face
identification. In 2nd IEEE Workshop on Applications of Computer Vision, pages
138–142, 1994
- [59] Geurts P, Ernst D, Wehenkel L. Extremely randomized trees[J]. Machine Learning, 2006,
63(1):3-42
- [60] V. L. C. Becker, R. Rigamonti and P. Fua. Supervised feature learning for curvilinear
structure segmentation. In MICCAI, 2013

附 录

3 个个体分类器的不同集成算法的准确率(%)比较

数据集	VOTE	SelectBest	SMLR	S4L	S2L	StackingM-F	StackingM
auto-mpg	78.4	82.42	82.41	72.36	81.68	81.15	83.91
ecoli	79.8	83.11	84.69	70.73	83.57	81.02	83.97
glass	75.31	76.03	72.56	71.32	71.21	70.87	76.67
arrhythmia	63.46	67.23	68.70	52.49	60.11	62.30	66.82
ionosphere	92.29	92.58	92.29	85.44	90.01	92.15	92.31
page-blocks	97.21	96.62	97.35	78.49	97.19	96.10	97.39
contraceptive	51.16	51.66	53.16	52.20	53.50	45.76	53.41
breast	96.71	95.85	94.56	93.85	96.71	96.10	96.42
pendigits	98.59	99.05	99.25	97.25	97.51	98.16	99.10
transfusion	76.27	75.13	77.20	56.54	76.88	76.20	75.14
segement	96.06	97.62	95.75	95.32	96.66	96.58	97.62
iris	94.00	94.51	95.00	95.33	93.33	94.33	94.33
wine	97.22	97.32	98.81	97.20	98.46	97.73	98.36
seeds-data	92.48	92.86	92.86	90.38	91.33	90.52	93.29
semeion	94.16	96.55	94.67	94.98	94.55	95.07	96.52

攻读硕士学位期间取得的研究成果

一、已发表(包括已接受待发表)的论文, 以及已投稿、或已成文打算投稿、或拟成文投稿的论文情况(只填写与学位论文内容相关的部分):

序号	作者(全体作者, 按顺序排列)	题 目	发表或投稿刊物名称、级别	发表的卷期、年月、页码	相当于学位论文的哪一部分(章、节)	被索引收录情况
1	徐慧丽	基于随机森林的多阶段集成学习方法	高师理科学刊	2018,38(02): 25-28+53	第四章	

注: 在“发表的卷期、年月、页码”栏:

1 如果论文已发表, 请填写发表的卷期、年月、页码;

2 如果论文已被接受, 填写将要发表的卷期、年月;

3 以上都不是, 请据实填写“已投稿”, “拟投稿”。

不够请另加页。

二、与学位内容相关的其它成果(包括专利、著作、获奖项目等)

致 谢

本文的研究工作是在我的导师凌卫新副教授的悉心指导下完成，衷心感谢凌老师在研究生三年里给予我学业生活上的关心，帮助和教导。从课题的选择到论文完成，凌老师始终给予我细心的指导和不懈的支持。在研究生三年的学习中，凌老师不但为我提供了良好的学习科研环境，更培养了我独立从事科研的能力。凌老师广博的知识，严谨的治学态度和踏实的工作作风为我树立了榜样，对我潜移默化的熏陶定将使我受益终身。

一转眼，三年的研究生生活也即将结束了。在这三年里我学到了很多，也付出了很多，不仅学到了很多知识，也学会很多为人处事之道。总之，这三年是我进步最大、收获最多的三年，从老师、同学和家人身上得到了很多帮助和支持，在这里我要对他们表示真诚的感谢。

首先感谢学院各位老师向我传授各种专业知识，感谢凌老师在学业上的指导和在生活中的关怀。从论文的选题到论文的完成，每一步都是在老师的指导下完成，都倾注了老师大量的心血。在此谨向凌老师表示衷心的感谢。

其次，感谢我的同学和舍友在三年时光里给予我的关心和帮助，感谢已毕业的师姐在学业上给予的指导和帮助，因为你们，这三年收获良多；我更要感谢我的父母及家人这些年来对我的支持和鼓励，在我远离家乡求学过程中，做我坚强的后盾，给予我前进的动力。

最后，我要衷心感谢在百忙之中抽出时间评阅本论文，参加论文答辩的各位专家和教授！

IV - 2 答辩委员会对论文的评定意见

集成学习可以缓解单一分类器的过拟合问题而被广泛应用, Stacking 算法是一种较强的集成方法, 但在小样本上的分类性能有待进一步提高, 而且计算时间有待进一步优化。论文选题较新, 具有一定的理论意义和应用价值。

论文首先针对 Stacking 算法的不足进行改进: 提出三层 Stacking 算法结构, 在第一层和第二层采用类概率输出, 且每一层对类概率输出采用不同的处理方式来表达下一层的输入, 在预测过程中加入投票策略增加分类器的泛化性能。接着, 基于遗传算法来优化改进的 Stacking 算法的结构。然后, 在 15 个 UCI 数据集上进行实验, 结果表明, 改进算法具有较好的分类效果, 且减少了运行时间。最后, 将比较了改进的算法, 与卷积神经网络, 在灰度人脸图像上的应用, 结果表明改进算法的实用性。

本文的选题具有一定的理论意义和应用价值。论文论证合理, 结论正确, 反映了该生具有一定的独立从事科研工作的能力。

答辩过程中, 语言简洁流畅, 能够准确理解并回答出答辩老师提出的问题。

答辩委员会经过无记名投票表决, 一致通过, 认为徐慧丽的论文已达到硕士学位论文的水平, 建议授予硕士学位。

论文答辩日期: 2018 年 6 月 2 日

答辩委员会委员共 5 人, 到会委员 5 人

表决票数: 优秀 (0) 票; 良好 (5) 票; 及格 (0) 票; 不及格 (0) 票

表决结果 (打“√”): 优秀 (); 良好 (√); 及格 (); 不及格 ()

决议: 同意授予硕士学位 (√) 不同意授予硕士学位 ()

答辩 委员 会成 员签 名	<u>周晓林</u> (主席)		<u>陈</u>
	<u>张</u>	<u>王</u>	<u>吴</u>