中国科学技术大学

博士学位论文

分类集成问题中的多样性研究与应用

作者姓名： 卞一珺

学科专业： 计算机软件与理论

导师姓名： 陈欢欢 教授

完成时间： 二〇二〇年七月

University of Science and Technology of China

A dissertation for doctor's degree

Research and Applications of Diversity in Ensemble Classification

Author: Bian Yijun

Speciality: Computer Software and Theory

Supervisor: Prof. Huanhuan Chen

Finished time: July 2020

中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文，是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名：_____　　签字日期：_____

中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一，学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅，可以将学位论文编入《中国学位论文全文数据库》等有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。本人提交的电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

□ 公开　□ 保密（____ 年）

作者签名：_____　　导师签名：_____

签字日期：_____　　签字日期：_____

摘　要

海量增长的数据为机器学习和人工智能方向的发展提供了坚实的基础，同时也带来了诸多挑战，如来源多样化、增长速度快、价值密度低、计算量大、分析复杂度大等。面对这些挑战，由传统机器学习方法所构建的单个学习器或许难以满足问题求解的精度需求，因此集成学习以其优越的性能而吸引了诸多研究者的注意。集成学习的成功依赖于两大要素，即其基学习器的准确度和多样性。然而学界在多样性的研究上尚未达成共识。尽管多样性对于构建一个性能优越的集成器而言非常重要，但如何度量多样性及其究竟如何作用于集成学习等问题仍然是开放性的。此外，随着基学习器准确度的提高，它们之间的多样性往往会有所下降，因此在构造一个性能良好的集成器的过程中，如何平衡这两者也是一项重要的研究课题。

为此，本文主要研究分类集成问题中的多样性及其应用，以期回答分类集成器中的多样性与泛化性能之间的关系这一问题，并利用多样性和准确度的合理平衡来完成集成剪枝任务，具体如下：

(1) 首先，受启发于回归集成器的误差分解，本文提出了分类集成器的误差分解并随之定义了一种多样性度量方法；随后利用该度量方法提出了分类集成器中的多样性与泛化性能之间的关系；最后利用该关系提出了基于多样性的集成剪枝方法，能够在不损害集成器性能的同时有效缩减其规模。

(2) 其次，为了对准确度和多样性进行合理平衡，本文从信息熵角度提出了一种基于信息熵的目标最大化的剪枝算法，包括集中式和分布式两个版本；随后从中抽取出一个可普适于现有剪枝算法的通用并行框架，能够在不损害剪枝后子集成器 (即由组成原始集成器的基分类器集合的一个子集所组成的集成器) 性能的同时大幅加速算法执行。

(3) 最后，本文将多样性应用到了神经架构搜索领域中的集成网络结构中，利用多样性对其进行剪枝，以期获得一个规模更小而性能持平甚至更优的子集成网络结构；三种剪枝策略均可被用作该过程中的指导，从而生成多样的且规模更小的子集成网络结构。

实验结果验证了本文所提出方法的合理性和有效性。

关键词：集成学习；误差分解；多样性；集成剪枝方法；机器学习

ABSTRACT

Massively growing data with extensive sources provides a solid foundation and brings many challenges as well for the development of machine learning and artificial intelligence, such as diverse sources, fast growth, low-value density, large amounts of computation, and huge complexity. Confronting these problems, one single learner built from traditional machine learning algorithms might not be able to meet the accuracy requirements that we need, which is why ensemble learning is attracting researchers' attention gradually due to its remarkable potential. The success of ensemble learning depends on two crucial elements, namely the accuracy and diversity of its individual learners that constitute a more powerful ensemble learner. However, the academic community has not reached a consensus on the study of diversity. Although it is widely accepted that diversity is substantial for building a powerful ensemble learner, how to measure diversity and what role diversity plays in ensemble learning remains an open question. Besides, the diversity between individual learners usually decreases with the increase of their accuracy, thus how to handle the trade-off between accuracy and diversity appropriately is a fundamental problem as well.

To this end, this dissertation focuses on the research and application of diversity in classification ensembles, to answer the question about the relationship between diversity and generalization performance in classification ensembles and utilize diversity to prune an ensemble by handling the trade-off between diversity and accuracy properly. To be specific, this dissertation mainly investigates three works.

(1) Firstly, inspired by the error decomposition of regression ensembles, this dissertation proposes the error decomposition of classification ensembles and defines a measure of diversity naturally; then the proposed diversity measure is used to analyze the relationship between diversity and generalization performance of classification ensembles; at last, two diversity-based ensemble pruning methods are proposed to reduce the size of an ensemble effectively without the damage of its generalization performance.

(2) Secondly, to balance the accuracy and diversity appropriately, this dissertation proposes an ensemble pruning method based on objection maximization from the perspective of information entropy, including a centralized version and a distributed version; subsequently, a general distributed framework for ensemble pruning is extracted to achieve less time consuming without much accuracy

degradation, which could be widely suitable for most of existing ensemble pruning methods.

(3) Thirdly, this dissertation investigates the application of diversity in the ensemble neural architectures in the field of neural architecture search, in which diversity is used to prune ensemble architectures and obtain sub-ensembles with smaller size; three pruning solutions are proposed to guide the searching process and generate diverse and smaller sub-ensemble architectures.

Experimental results verify the reasonableness and effectiveness of these proposed methods in the dissertation.

Key Words: Ensemble Learning; Error Decomposition; Diversity; Ensemble Pruning; Machine Learning

目　录

插 图 清 单

表 格 清 单

算 法 索 引

符 号 说 明

如无特殊说明，文中的标量、向量、张量和矩阵变量分别由普通小写字母 (例如 x)、粗体小写字母 (例如 $\mathbf{x}$)、倾斜加粗小写字母 (例如 $\boldsymbol{x}$) 和普通大写字母 (例如 X) 表示。无附加说明的向量均可视为列向量形式。花体字母 (如 $\mathcal{X}$) 标识集合或空间；无衬线字母 (如 X) 则标识着随机变量。哥特字母 (如 $\mathfrak{X}$) 通常表示一些特殊含义，具体可参照文中说明。

其他所使用的符号可简单罗列如下：

$[n]$	取值集合 $\{1, 2, ..., n\}$ 的简记形式
$[\cdot]^\mathsf{T}$	向量或矩阵的转置
$\mathbf{x}$	样本特征的向量表示, 即 $\mathbf{x} = [x_1, ..., x_d]^\mathsf{T}$, 其中 d 代表特征数
$\boldsymbol{x}$	样本特征; 当其为向量时, 与 $\mathbf{x}$ 相同
y	样本标签
$h(\cdot)$	分类器, 表示一个映射 $h : \boldsymbol{x} \mapsto y$, 无歧义时也可简写作 h
w	权值, 是一个标量
$\mathcal{D}$	数据集, 即 $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m)\}$, 其中 m 代表样本数
$\mathcal{H}$	用于构造 (原始) 集成器的基分类器集合, 即 $\mathcal{H} = \{h_1, h_2, ..., h_n\}$, 其中 n 代表基分类器的数目
$\mathcal{P}$	用于构造剪枝后的子集成器的基分类器集合, 即 $\mathcal{P} = \{h_{n_1}, h_{n_2}, ..., h_{n_k}\}$, 其中 k 代表剪枝后子集成器的规模, 与使用的剪枝算法有关
$\mathcal{F}$	函数的假设空间, 且 $h \in \mathcal{F}$
$\mathcal{X}$	样本的输入空间 / 特征空间, 且 $\boldsymbol{x} \in \mathcal{X}$
$\mathcal{Y}$	样本的输出空间 / 标签空间, 且 $y \in \mathcal{Y}$
X	样本特征的随机变量, 且 $\boldsymbol{x} \sim \mathsf{X}$
Y	样本标签的随机变量, 且 $y \sim \mathsf{Y} = \{c_1, c_2, ..., c_\ell\}$, 其中 ℓ 表示标签数. 在二分类问题中, $\ell = 2$; 在多分类问题中, $\ell > 2$, 且 $\ell \in \mathbb{Z}^+$
$\mathbf{P}$	随机变量的概率函数
$\mathbb{E}$	随机变量的期望函数
$\mathbb{Z}^+$	正整数域
$\mathbb{R}$	实数域
$\mathbb{I}(\cdot)$	指示函数: 当命题为真时, 其值为 1; 当命题为假时, 其值为 0
$\mathrm{sign}(\cdot)$	符号函数: 若自变量为正, 则其值为 1; 若自变量为负, 则其值为 -1; 否则值为 0. 一些文献中也写作 $\mathrm{sgn}(\cdot)$

$\mathfrak{A}$ 学习算法

$\mathfrak{T}$ 学习任务

$\mathfrak{W}$ 样本分布

第 1 章　绪　　论

1.1　研究背景与意义

近年来，随着互联网的普及和各种智能终端的广泛应用，全球数据总量呈现爆炸式增长态势，人类社会已进入大数据时代。在丰富的数据基础上，机器学习和人工智能方向的发展也迎来了新的挑战，如数据来源的多样化、数据增长的快速性、待挖掘数据的低价值密度和分析处理的计算复杂度等 [13-14]。面对这些问题，由传统机器学习方法所构建的单个学习器或许难以满足学习任务的精度需求，因此，集成学习作为一种能显著提高弱分类器性能的有效手段，逐渐进入了人们的视野。

集成学习，又名基于委员会的学习、多分类器系统或是专家组合 [2-3, 15]，以其卓越的性能吸引了诸多研究者的注意。集成学习训练一组分类器，并联合它们的意见用于给出最终决策，通常可以获得比单一的分类器更优越的性能 [1]。根据组成集成器的基分类器 (又名弱分类器或个体分类器) 的类别差异，集成器可以分为同质集成器或异质集成器两种 [2]，例如常见的自助法 [4] 和提升法 [16-17] 均是同质集成器。现如今，集成学习已被广泛应用于多个领域中的学习任务，如半监督学习 [18-19]、情感分析 [20]、目标检测 [21]、目标识别 [22-23]、目标跟踪 [24]、图像检索 [25]、图像分类 [20, 26-28]、基因函数预测和 RNA 拼接 [29] 等。

在集成学习领域中存在一个广泛的共识，即集成器的性能通常优于单个分类器的性能。Dietterich [1] 指出集成器的成功依赖于两大要素，即构成该集成器的基分类器的准确度和多样性。从直觉上来说，“一个分类器是准确的” 意味着它在未知样本上犯错的概率优于随机猜测；“两个分类器是多样的” 意味着它们在未知样本上所犯的错误是不同的。当集成器中基分类器的准确度有所提高时，它们之间的多样性往往会呈现出一种衰退趋势 [30]。因此，如何权衡取舍这两个目标以获得最优性能的集成器是集成学习中所研究的一个重要课题。

“多样性” 是集成学习中的一个重要概念，它来源于回归集成问题中的误差分解 [31]，其中回归集成器的误差项可以分解为准确度和多样性两项。然而在分类问题中，学者对多样性度量的定义尚未达成共识 [32-34]。现有的关于多样性的研究主要集中在三大问题上：1. 多样性应当如何定义和度量；2. 多样性与集成器泛化性能之间的关系；3. 如何利用多样性来进一步提高集成器的性能。

Brown [32] 从信息理论角度指出，集成器中的多样性有多重存在形式，即不同基分类器之间的多重水平的交互作用。Zhou et al. [33] 在此基础上提出，应最大化共有信息以最小化集成器的泛化误差。随后，Yu et al. [35] 认为基分类器中的多样性以成对形式存在，并可以用来缩减假设空间的复杂度，即多样性在集成

方法中起正则作用，并据此提出了多样性正则机。Guo [36] 定义了 “排他性” 来度量基分类器之间的多样性，将其作为一个正则项，并提出了排他性正则机。然而多样性正则机的所需条件并不总是成立，且收敛所消耗的时间较长，因而适用范围有限；而排他性正则机则是在支持向量机的基础上提出的，并不适合于其它种类的基分类器。最近，Jiang et al. [37] 把经典的模糊分解从回归问题扩展到分类问题上，并证明了模糊项的一些关键性质，但是没有量化多样性对集成器性能的影响。

集成方法虽然效果显著优越，但是也有着不可忽视的缺点，即它需要的时间和空间代价都随着基分类器数目的增加而有显著增长。为了解决这个问题，集成剪枝方法应运而生。集成剪枝问题也被称作是集成选择或集成稀疏 [38-44]，它的目标是从给定的一组已被训练好的基分类器中选出一个性能尽可能优的子集。集成剪枝能够缩减原始集成器的规模，并提高原始集成器的泛化性能 [45]。如何快速有效地达成这个目的，是集成学习中另一个值得研究的问题。

现有的集成剪枝方法可大体分为三类，即基于排序的方法、基于聚类的方法和基于优化的方法 [46-49]。但是这些方法通常是集中式的，即集成器中所有的基分类器都在同一台机器上存储和处理。随着大数据时代的到来，数据已有了海量的增长，而集成器本身也会随着数据量的增长而扩大，造成了集中式剪枝方法在执行过程中的性能瓶颈，因而给分布式的集成剪枝方法赋予了新的研究价值。

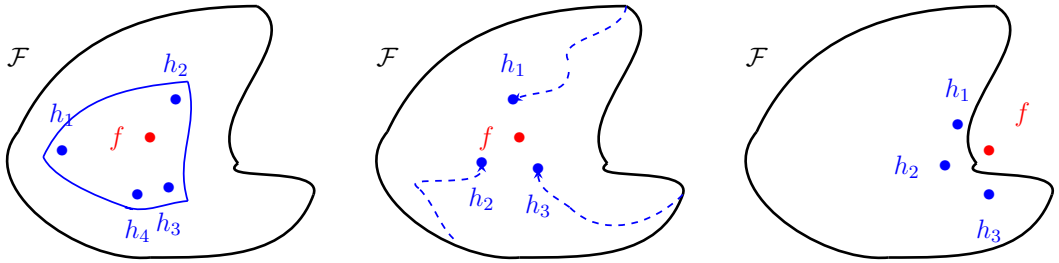
综上所述，对于多样性这一集成学习中的关键元素进行研究和探索，能够帮助研究者构建性能更优的集成器或集成剪枝方法，也是集成学习中的一项重要研究课题。

1.2　国内外研究现状

本节用于概述集成学习的优越性，并简要介绍现有多样性度量方法的分类，以及集成剪枝方法的分类。

1.2.1　集成学习的优越性

与仅构建单个分类器来解决学习问题不同，集成方法通过训练多个分类器并联合结果综合考虑来解决问题。用于构建集成器的若干分类器被称作基分类器，又名弱分类器、个体分类器或组件分类器。基分类器由基学习算法在训练数据上生成而得，可以是决策树、神经网络或是其他机器学习方法 [2]。绝大多数集成方法使用的是同一种基学习算法来构建基分类器，所构成的集成器被称为是同质集成器；也可以使用多个不同的基学习算法来构建基分类器，所构成的集成器被称作是异质集成器。除本节外，如非特意指明，后文中所提到的集成器均



(a) 统计角度　　(b) 计算角度　　(c) 可代表性角度

图 1.1　集成器多优于单个分类器的三个根本原因 [1]

指同质集成器。

一个集成器的泛化能力通常优于单个分类器的泛化能力。事实上，集成方法出现的主要原因就在于它们能够增强弱分类器的泛化性能，其中弱分类器指的是性能仅略微优于随机猜测的分类器；而强分类器指的是能够做出准确预测的分类器 [2]。集成器的性能之所以优于单个分类器的性能，其根本原因主要在于三点 [1]：

1. 从统计角度上来说，一个学习算法的学习过程可被看作是从假设空间 $\mathcal{F}$ 中搜索出最优假设。当可用的训练样本相比于假设空间来说太小时，会出现统计问题，即当数据不充足时，学习算法可以寻找到多个假设，而它们在训练集上均能表现出同等水平的准确度。通过构建一个集成器，算法可以“平均”它们的投票，并缩减选择分类器时失误的风险。如图 1.1(a) 所示。
2. 从计算角度上来说，许多学习算法使用了局部搜索的形式，因此可能最终陷于局部最优点，而非全局最优点。如使用梯度搜索算法的神经网络，其目标是在训练样本上最小化误差函数；又如决策树算法使用贪婪划分策略进行扩张。假设训练样本充足 (即统计问题可忽略)，对于一个学习算法来说，要从假设空间中找到最优假设可能仍然是一个难以计算的问题。事实上，神经网络和决策树的最优训练均是 NP 难问题 [50-51]。而通过从不同初始点多次执行学习算法可获得若干个学习器，用它们所组成的集成器或许会提供一个对于真实未知函数的更优近似。如图 1.1(b) 所示。
3. 从可代表性角度上来说，在机器学习的许多应用中，真实函数无法被学习算法的假设空间所表达。通过若干个基分类器所组成的带权投票，或许能够扩展这些函数的可表达空间，使其更接近真实函数。如图 1.1(c) 所示。

构建集成器的关键主要在于使用准确且多样的基分类器，如最常见的集成方法自助法 [4] 和提升法 [16-17]，分别通过在现有的训练数据上进行子采样或者

是重新赋权值来构建不同的基分类器；也可以通过使用不同的训练数据、不同的输入特征、不同的输出表达或者是不同的初始值来构建不同的基分类器。

1.2.2　集成学习中的多样性

从直观上来说，多样性代表集成器中不同基分类器之间的差异，又被称作依赖性、正交性、互补性或排他性 [3, 36]。在实践中，基分类器通常由训练集的不同子集训练而得，这使得它们高度相关，无法满足 “基分类器之间应相互独立” 的假设，也就很难使它们多样化 [2]。许多集成方法采用隐式或启发式的方法来构造多样的基分类器。常见的构造方法包括：使用不同的输入数据或输入特征，使用不同的输出目标，加入随机噪声 [1-2] 等。

大量研究提出了不同的多样性度量方式，可以在输入空间 [52] 或输出空间 [15, 33, 48, 53-54] 中度量集成器的多样性，其中后者又占绝大多数。现有的多样性度量方式可大体分为两类，即成对多样性度量和非成对多样性度量 [2-3]。成对多样性度量代表一对分类器之间行为的差异，其典型的度量方式包括 Q 统计量 [55]、κ 统计量 [56]、分歧度量 [57-58]、相关系数度量 [59] 和双重失误度量 [60] 等；若通过成对多样性度量计算一组分类器的多样性，需要先两两计算成对多样性，随后取其平均。而非成对多样性度量可以直接计算一组分类器之间的多样性，其典型的度量方式包括测试者彼此之间的一致性度量 [61]、Kohavi-Wolpert 差异度量 [62]、投票的熵度量 [63-64]、难度指数度量 [3, 65]、广义多样性 [66] 和一致失误多样性 [66] 等。此外还有两种无法直接归类的度量方式：一种是负相关学习中的相关惩罚函数 [67]，度量的是每个基分类器相对于整体集成器的差异性；另一种是模糊项 [68]，度量的是集成器中每个成员相对于整体输出的平均偏移量。但是现有研究无法证实在以上度量方式中哪一种更加优越 [15]。

尽管多样性在集成学习中的重要性已获得学者广泛认可，它对集成器泛化性能的确切作用尚未明确。许多学者试图从误差分解的角度对此进行分析，其中两种广为人知的方法是经典模糊分解和偏差-方差-协方差分解。然而这两种方法都仅适用于使用平方误差的回归问题。Brown et al. [69] 将分类集成器的损失与基分类器的平均损失之间的差异分成两项，一项度量在被分对的数据点上基分类器之间的差异，称作 “优质” 多样性；另一项度量在分错的数据点上基分类器之间的差异，称作 “劣质” 多样性。他们认为增大优质多样性的值可以减小集成器的泛化误差，而增大劣质多样性的值则会起到相反的效果。Jiang et al. [37] 把适用于回归任务的经典模糊分解扩展到分类问题上，只需其损失函数二次可微。可以使用的损失函数包括且不限于逻辑回归中的损失函数、提升法中的幂指数损失和分类任务中常用的 $0/1$ 损失。他们展示了两种分类问题的广义模糊分解，讨论了模糊项所具备的性质，并得出了与 [69] 一致且更普适的结论。但是这种方

法只是定性指出了多样性的一些特点，并没有量化多样性对集成器性能的影响。

1.2.3　集成学习中的剪枝方法

尽管集成方法具有显著的优越性和有效性，其所消耗的时间和空间代价也是巨大的。为了缓解这一缺点，集成剪枝方法应运而生。集成剪枝起源于 1997 年，Margineantu et al. [70] 首次展示了 “从原始集成器中挑选出一个子集仍能构成相同水平性能的子集成器” 的可能性是存在的。后来，Zhou et al. [45] 提出了误差的偏差-方差分解，用于解释所提出算法的优越性，并展示了集成剪枝可以获得规模更小而性能更优的子集成器。但是，从原始集成器中挑选出一个性能最优的子集是非常困难的。一个困难是对子集成器的泛化性能进行估计本身就存在很大的挑战，另一困难则是寻找最优子集是一个具有幂指数级计算复杂度的组合搜索问题 [48]。事实上，从一个集成器中搜索得到最佳子集是一个 NP 完全难问题，甚至连求其近似解都很难处理 [71]。因此近年来研究者也提出了很多剪枝方法用于解决这一问题，这些剪枝方法可以被大体分为三类，即基于排序的剪枝方法、基于聚类的剪枝方法和基于优化的剪枝方法。

基于排序的剪枝方法是其中最简单的一类方法，其思想是：首先根据不同的评估准则对集成器中的基分类器进行排序，然后选择性能最优的前若干个用于组成剪枝后的子集成器。不同的剪枝方法使用不同的评估准则，可以选择误差较小的基分类器 (如定向排序剪枝 [72])，可以选择多样性较大的基分类器 (如 KL 散度剪枝和 Kappa 剪枝 [70])，也可以综合考虑二者选择合适的基分类器 (如多样性正则化集成剪枝 [48])。基于聚类的剪枝方法则先使用聚类算法将不同的基分类器聚成若干组，其中相似的基分类器将会被聚入一组；随后在各组内分别挑选性能最优的基分类器用于构建剪枝后的子集成器，这样做是为了增加子集成器的多样性 [47]。值得注意的一点是，这类方法易于改造成并行式的方法进行处理，可用于加速整体的剪枝过程。而基于优化的剪枝方法将集成剪枝任务视作一个优化问题，其优化目标被定义为可反映原始集成器中所挑选出的子集的性能的一个度量。由于这是一个 NP 完全难问题，在集成器的子集空间中穷尽搜索并不可行 [48, 71]。因此也有许多技术被用于缓解这一困境，如基因算法 [73]、贪婪算法 [74]、爬山法 [75] 和双目标演化优化算法 [49] 等。

在众多剪枝方法中，基于排序的方法和基于优化的方法尤为出众。基于排序的方法胜在执行起来相对快速，稳健性较强，但是性能相对稍逊一筹；基于优化的方法胜在性能优异，因其目标即为直接选择最优或者接近最优的子集成器，但执行起来代价更高，耗时更久 [46, 49]。此外，Dai et al. [76] 提出了一个分层的剪枝方法，在层间可以并行地在该层基分类器的子集上执行剪枝。但是这种方法是一个不断迭代的过程，因此对于直接加速剪枝过程收效不大。

1.3　研究内容及方法

本文主要研究分类集成问题中的多样性及其在集成剪枝算法中的应用。为了有效地利用多样性，首先要明了的是多样性在集成器中是如何起作用的，即多样性与集成器泛化性能之间的关系。本文以多样性为核心，分别探讨了直接利用多样性的剪枝算法、隐式利用多样性的剪枝算法和其他领域中利用多样性的剪枝方法。具体如下：

本文第 3 章旨在探讨分类问题中集成器的多样性与泛化性能之间的关系。集成学习成功的关键在于其基分类器的准确度和多样性。在回归问题中，通过误差分解可以把集成器的泛化性能分成两项，一项是基分类器的准确度，另一项则被定义为多样性 [31]。常用的误差分解方法有经典模糊分解和误差方差协方差分解。但是它们都只适用于带有平方损失的回归任务。分类任务中的多样性问题尚悬而未决。为了解决这一问题，我们首先提出了适用于使用 0/1 损失函数的分类问题的误差分解，并据此定义了一种多样性度量方法；然后利用边界这一概念，将所提出的多样性度量与集成器的泛化误差联系起来，从而能够对这两者之间的关系加以理论分析；此外，根据所提出的多样性与集成器泛化性能之间的关系，我们也提出了两种直接利用多样性的剪枝方法。最后，在实际数据集和基准对比算法上的实验结果证明了第 3 章中所提出方法的有效性和优越性。

本文第 4 章旨在探讨基于信息熵来隐式利用多样性的集成剪枝方法，并对整个剪枝过程进行加速。鉴于目前尚无得到广泛共识的多样性度量方法，我们采用信息熵领域内的互信息、共有信息和信息变异等概念，提出了一个目标最大化的优化函数，用于隐式表达对集成学习中两大要素 (即准确度和多样性) 的平衡，并据此提出一种基于优化的剪枝方法。随后，我们引入了并行计算领域中可组合核心集的概念，可以把顺序执行的剪枝方法改造成可并行执行的剪枝方法，并据此提出一个可普适于现有剪枝方法的并行框架，能够大幅提高剪枝算法执行的效率。最后，在实际数据集和基准对比算法上的实验结果证明了第 4 章中所提出方法的有效性和优越性。

本文第 5 章旨在探讨利用多样性的集成剪枝方法在其他领域内的应用。这项工作的研究背景是集成学习能够与其他领域结合并发挥出自身优势。例如在深度神经网络中，自动化地搜索出一个性能良好的网络模型往往需要巨大的计算代价。为了缓解这一问题，一些研究者提出了利用集成学习的搜索方法，首先自动化地搜索出若干个规模较小的网络模型，再将它们集成起来，以构建一个性能良好的网络集成器。但是这些方法忽视了多样性在集成学习领域中所起到的重要作用，因此，我们利用了多样性这一重要因素，将其引入 AdaNet [77-78] 这一网络模型的自动化搜索框架中，并提出一个基于集成神经网络的剪枝方法，可

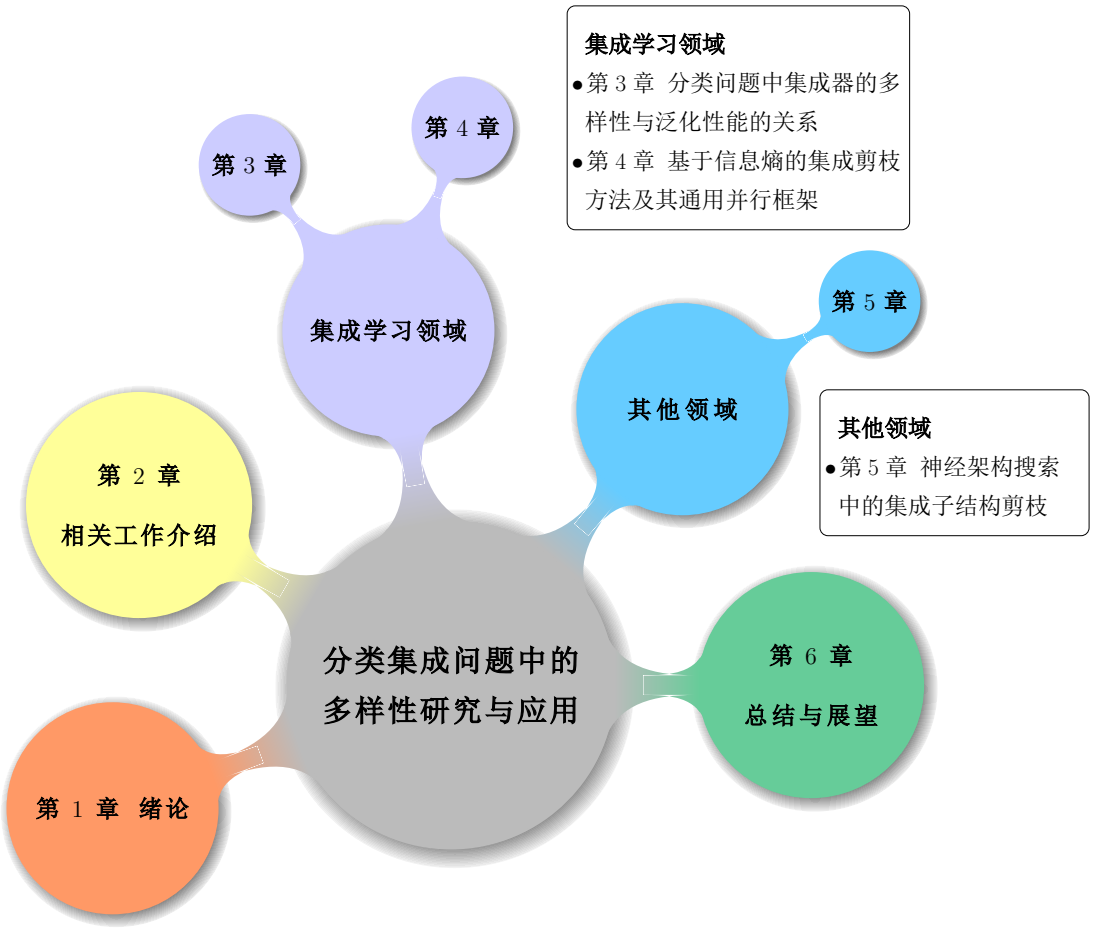


图 1.2　全文结构安排

以获得规模更小的和多样性更强的神经网络集成模型。最后，在实际数据集和基准对比算法上的实验结果证明了第 5 章中所提出方法的有效性。

总的来说，本文研究的是分类集成问题中的多样性与集成器泛化性能之间的关系，以及多样性在集成剪枝方法中的应用。概括地说，第 3 章和第 4 章分别对应的是集成学习领域中直接利用多样性和间接利用多样性的剪枝方法，而第 5 章则对应基于多样性的集成剪枝方法在其他领域中的应用。此外需要注意的一点是，尽管第 3–4 章都是在集成学习领域对多样性进行研究并提出相应的剪枝方法，但是第 4 章并未直接使用现有的或是第 3 章中所提出的多样性度量，而是利用信息熵中的概念来对多样性进行隐式表达，这使得第 4 章可以不局限于第 3 章的研究内容 (即二分类学习问题)，而是适用于更广泛的多分类学习问题。

1.4　本文结构安排

本文的结构安排如图 1.2 所示。第 1 章首先介绍本文的研究背景和研究意义，然后概述国内外研究现状，包括集成方法之所以优越的原因以及常见的多样性度量方法和集成剪枝算法的分类，最后介绍本文的研究内容和方法。第 2 章是相

关工作介绍，包括集成学习中的常见方法、多样性度量的计算方法、集成剪枝方法以及集成学习在深度学习领域中的应用。第 3 至 5 章为本文主要工作。具体来说，第 3 章和第 4 章分别讨论在集成学习领域中分类集成问题中的多样性与泛化性能的关系和基于信息熵的集成剪枝问题；第 5 章讨论在神经架构搜索中的集成子结构剪枝问题。最后，第 6 章总结本文工作并给出对未来工作的展望。

第 2 章　相关知识介绍

本章主要介绍本文所涉及到的一些相关知识，包括集成学习的常见方法、多样性度量方法、集成剪枝方法和集成学习在深度学习中的应用。具体来说，我们首先在第 2.1 节中介绍集成学习及其常见方法，如自助法 [4] 和提升法 [16-17]，以及集成器的联合方法。然后第 2.2 节介绍常见多样性度量的计算方法，包括成对多样性度量和非成对多样性度量。由于本文中的多样性主要被用于提高集成剪枝方法的性能，因此接下来的第 2.3 节是对集成剪枝方法进行概述。第 2.4 节紧随其后，介绍了集成学习在深度学习中的应用。最后第 2.5 节总结本章内容。

2.1　集成学习中的常见方法

一般来说，一个学习任务的目标是学习从输入数据到输出数据的映射，即 $f : \mathbf{x} \mapsto y$. 其中 $\mathbf{x}$ 和 y 分别代表样本特征和标签。特别地，在回归问题中，y 为连续值；而在分类问题中，y 为离散值。本文主要关注于分类集成问题的研究。

若给定数据集 $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$，其中，$\mathbf{x}_i \in \mathbb{R}^d$ 为第 i 个训练样本的特征向量，y_i 为该样本所对应的标签，而 m 和 d 分别代表训练样本和样本特征的数量。令 $h(\cdot)$ 表示一个被训练好的学习器，即在学习算法的假设空间中，$h(\cdot)$ 是对真实函数 $f : \mathbf{x}_i \mapsto y_i$ 的一个近似，则集成学习可以定义为训练一组基学习器，如 $\mathcal{H} = \{h_1(\cdot), h_2(\cdot), ..., h_n(\cdot)\}$，其中 n 表示训练好的基学习器的数量。这些基学习器的预测结果将被汇总到集成器中，以得出最终的预测结果。整个过程如图 2.1 [2] 所示。

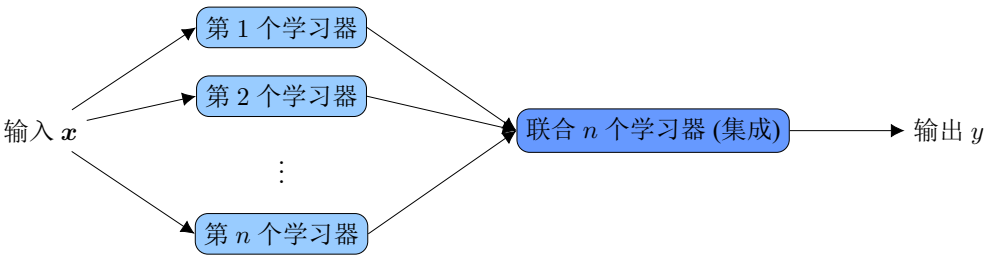


图 2.1　集成学习的过程 [2]

常见的集成方法包括自助法 (Bagging) [4]、提升法 (Boosting) [16-17] 和堆叠法 (Stacking) [10-12]。自助法通过在训练集上进行子采样的方式构造出多个训练子集，然后分别在这些训练子集上执行学习算法，可获得若干个基学习器，如算法 2.1 所示。而提升法则是以一种高度自适应的方式逐个学习基学习器，每次迭代所得的基学习器都依赖于当前已有的基学习器，其通用流程如算法 2.2 所示。

Data: 数据集 $\mathcal{D}=\{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),...,(\mathbf{x}_m,y_m)\}$;
基学习算法 $\mathfrak{A}$; 学习过程的迭代次数 n
Result: 集成器 $H(\cdot)$

```
for i = 1,...,n do
    h_i = 𝔄_i(𝒟, 𝔚_bs) ;                    // 其中的 𝔚_bs 代表自助分布
end
H(x) = arg max_{y∈𝒴} Σ_{i=1}^{n} 𝕀(h_i(x) = y) ;    // 获得 Bagging 集成器
return H
```

算法 2.1: 自助法 (Bagging) [4]

Data: 样本分布 $\mathfrak{W}$; 基学习算法 $\mathfrak{A}$; 学习过程的迭代次数 n
Result: 集成器 $H(\cdot)$

```
𝔚_1(x) = 𝔚 ;                                  // 初始化权重分布
for i = 1,...,n do
    h_i = 𝔄(𝔚_i) ;                            // 样本分布的初始化
    ε_i = P_{x∼𝔚_i}(h_i(x) ≠ f(x)) ;           // 评估基分类器 h_i 的错误率
    𝔚_{i+1} = 根据 ε_i 调整样本分布 𝔚_i 所得 ;    // 更新样本分布
end
H(x) = 联合所有基分类器 {h_1(x), h_2(x), ..., h_n(x)} 的输出结果 ;    //
                                              // 获得 Boosting 集成器
return H
```

算法 2.2: 提升法 (Boosting) [5-8] 的通用流程

此外，算法 2.3 展示了提升法算法族中的一个例子，即自适应增强 (AdaBoost) 算法 [9]，需要注意的一点是自适应增强法仅适用于二分类问题，且其标签空间为 $\mathcal{Y}=\{-1,+1\}$. 注意自助法和提升法所构成的集成器都是同质集成器。与之不同的是，堆叠法所构成的集成器是异质集成器。在堆叠法中，首先在训练集上使用不同的基学习算法学习出若干个基学习器 (又称一级学习器)；随后将这些一级学习器的输出作为输入特征，可以生成一个新的数据集，用于训练元学习器 (又称二级学习器)；最终可获得集成器所预测出的结果，如算法 2.4 所示。

若给定已训练好的基学习器集合 $\mathcal{H}=\{h_1(\cdot),h_2(\cdot),...,h_n(\cdot)\}$，将所有基学习器的预测结果进行联合可得集成器的最终预测结果。不同的联合方式可以获得不同的集成器。在回归问题中，学习器的输出是连续值，因此通常采用平均法来联合基学习器的预测结果，如：

- *简单平均法*

$$H(\mathbf{x})=\frac{1}{n}\sum_{i=1}^{n}h_i(\mathbf{x})\,; \tag{2.1}$$

Data: 数据集 $\mathcal{D}=\{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),...,(\mathbf{x}_m,y_m)\}$;
基学习算法 $\mathfrak{A}$; 学习过程的迭代次数 n
Result: 集成器 $H(\cdot)$

$\mathfrak{W}_1(\mathbf{x})=1/m$; // 初始化权重分布
for $i=1,...,n$ **do**
  $h_i=\mathfrak{A}(\mathcal{D},\mathfrak{W}_i)$; // 训练基分类器
  $\epsilon_i=\mathbb{P}_{\mathbf{x}\sim\mathfrak{W}_i}\big(h_i(\mathbf{x})\neq f(\mathbf{x})\big)$; // 评估该基分类器的错误率
  if $\epsilon_i>0.5$ **then**
    break ; // 终止循环
  end
  $\alpha_i=\frac{1}{2}\log\big(\frac{1-\epsilon_i}{\epsilon_i}\big)$; // 确定该基分类器的权值
  $\mathfrak{W}_{i+1}(\mathbf{x})=\frac{1}{Z_i}\mathfrak{W}_i(\mathbf{x})e^{-\alpha_i f(\mathbf{x})h_i(\mathbf{x})}$; // 更新权值分布，
  // 其中 Z_i 是一个正则化因子，能够使 $\mathfrak{W}_{i+1}$ 成为一个分布
end
$H(\mathbf{x})=\operatorname{sign}\big(\sum_{i=1}^{n}\alpha_i h_i(\mathbf{x})\big)$; // 获得集成器
return H

算法 2.3: 提升法 (Boosting) 示例: 自适应增强 (AdaBoost) 算法 [9]

- 带权平均法

$$H(\mathbf{x})=\sum_{i=1}^{n}w_i h_i(\mathbf{x})\,, \tag{2.2}$$

其中 w_i 是基分类器 h_i 的权值，且满足 $w_i\geqslant 0$ 和 $\sum_{i=1}^{n}w_i=1$.

在分类问题中，学习器的输出是离散值，因此通常采用投票法来联合基学习器的预测结果。令标签空间 $\mathcal{Y}=\{c_1,c_2,...,c_\ell\}$，其中 ℓ 代表标签数量。则常见的投票方法为：

- 最大化投票法

$$H(\mathbf{x})=\begin{cases}c_j\,, & \text{若 } \sum_{i=1}^{n}\mathbb{I}\big(h_i(\mathbf{x})=c_j\big)>\frac{1}{2}\sum_{k=1}^{\ell}\sum_{i=1}^{n}\mathbb{I}\big(h_i(\mathbf{x})=c_k\big)\,;\\ \text{拒绝}\,, & \text{否则}\,.\end{cases} \tag{2.3}$$

- 众数投票法

$$H(\mathbf{x})=\arg\max_{c_k}\sum_{i=1}^{n}\mathbb{I}\big(h_i(\mathbf{x})=c_k\big)\,. \tag{2.4}$$

- 带权投票法

$$H(\mathbf{x})=\arg\max_{c_k}\sum_{i=1}^{n}w_i\mathbb{I}\big(h_i(\mathbf{x})=c_k\big)\,, \tag{2.5}$$

其中 w_i 是分配给基分类器 h_i 的权值，且满足 $w_i\geqslant 0$ 和 $\sum_{i=1}^{n}w_i=1$，与带权平均法相似。

Data: 数据集 $\mathcal{D}=\{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),...,(\mathbf{x}_m,y_m)\}$;
一级学习算法 $\mathfrak{A}_1,\mathfrak{A}_2,...,\mathfrak{A}_n$; 二级学习算法 $\mathfrak{A}$
Result: 集成器 $H(\cdot)$

for $i=1,...,n$ **do**
  $h_i=\mathfrak{A}_i(\mathcal{D})$;　// 使用一级学习算法来训练一级学习器
end
$\mathcal{D}'=\emptyset$;　// 生成新的数据集
for $i=1,...,m$ **do**
  for $j=1,...,n$ **do**
    $z_{ij}=h_j(\mathbf{x}_i)$;
  end
  $\mathcal{D}'=\mathcal{D}'\cup\big((z_{i1},z_{i2},...,z_{in}),y_i\big)$;
end
$h'=\mathfrak{A}(\mathcal{D}')$;　// 使用二级学习算法在新的数据集上训练二级学习器
$H(\mathbf{x})=h'\big(h_1(\mathbf{x}),h_2(\mathbf{x}),...,h_n(\mathbf{x})\big)$;　// 获得 Stacking 集成器
return H

算法 2.4: 堆叠法 (Stacking) [10-12]

2.2　集成学习中的多样性度量

目前已有许多种多样性度量方法被提出，这些方法可大体分为两类，即成对的和非成对的多样性度量方法。若已知用于构建集成器的基分类器集合为 $\mathcal{H}=\{h_1(\cdot),h_2(\cdot),...,h_n(\cdot)\}$，其中 n 是基分类器的数量，即集成器规模。本节以给定数据集 $\mathcal{D}=\{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),...,(\mathbf{x}_m,y_m)\}$ 为例，简要介绍常见的多样性度量的计算方法，其中标签空间为 $\mathcal{Y}=\{c_1,c_2,...,c_\ell\}$，且 m 和 ℓ 分别为样本数量和标签数量。若 $\ell=2$，则是二分类问题；若 $\ell\in\mathbb{Z}^+$ 且 $\ell>2$，则是多分类问题。

2.2.1　成对的多样性度量

为了度量集成器的多样性，一种经典的计算方法是首先计算基分类器两两之间的多样性，然后再将这些多样性值做一平均。对于任意两个分类器 $h_i(\cdot)$ 和 $h_j(\cdot)$，可知其列联表如表 2.1 所示，其中样本数量 m 是一个非负整数，且满足

$$m=a+b+c+d\,. \tag{2.6}$$

表 2.1　两个分类器的列联表

	$h_j(\mathbf{x})=y$	$h_j(\mathbf{x})\neq y$
$h_i(\mathbf{x})=y$	a	c
$h_i(\mathbf{x})\neq y$	b	d

基于表 2.1，我们可以计算一些经典的成对多样性度量方法。举例来说，对于任意两个分类器 $h_i(\cdot)$ 和 $h_j(\cdot)$，它们之间的多样性可以通过以下几种方式来计算：

- Q 统计量 [55] 被定义为

$$Q_{ij} = \frac{ad - bc}{ad + bc}. \tag{2.7}$$

  注意 $Q_{ij} \in [-1, 1]$. 若 Q_{ij} 其值为零，则说明这两个分类器之间是相互独立的。此外，当两个分类器做出的预测相似时，Q_{ij} 值为正；当两个分类器做出不同的预测时，Q_{ij} 值为负。

- κ 统计量 [56, 79] 是统计领域中的一种经典方法。Margineantu et al. [70] 和 Dietterich [80] 将其首次用于度量两个分类器之间的多样性，即

$$\kappa_p = \frac{\Theta_1 - \Theta_2}{1 - \Theta_2}, \tag{2.8}$$

  其中 Θ_1 和 Θ_2 分别是这两个分类器意见一致或偶然一致的概率，即

$$\Theta_1 = \frac{a + d}{m}, \tag{2.9}$$

$$\Theta_2 = \frac{(a + b)(a + c) + (c + d)(b + d)}{m^2}. \tag{2.10}$$

  当两个分类器在数据集 $\mathcal{D}$ 上的意见完全一致时，$\kappa_p = 1$；当两个分类器意见一致的情况是随机出现时，$\kappa_p = 0$；当两个分类器意见一致的可能性甚至比偶然情况的概率还低时，$\kappa_p < 0$. 注意式 (2.8) 化简后可得

$$\kappa_p = \frac{2(ad - bc)}{(a + b)(b + d) + (a + c)(c + d)}. \tag{2.11}$$

- 分歧度量 [57-58] 定义的是两个分类器做出不同预测的样本比率，即

$$\begin{aligned} dis_{ij} =& \frac{b + c}{m} \\ =& \frac{1}{m} \sum_{k=1}^{n} \mathbb{I}\big(h_i(\mathbf{x}_k) \neq h_j(\mathbf{x}_k)\big). \end{aligned} \tag{2.12}$$

  注意 $dis_{ij} \in [0, 1]$，且其值越大，则两个分类器之间的多样性程度越大。

- 相关系数度量 [59] 被定义为

$$\rho_{ij} = \frac{ad - bc}{\sqrt{(a + b)(a + c)(c + d)(b + d)}}. \tag{2.13}$$

  这是度量两个二分类向量之间相关性的一种经典统计量。易知 ρ_{ij} 与 Q_{ij} 符号相同，且 $|\rho_{ij}| \leqslant |Q_{ij}|$.

- 双重失误度量 [60] 定义的是两个分类器都误分类的样本比率，即

$$\begin{aligned} df_{ij} =& \frac{d}{m} \\ =& \frac{1}{m} \sum_{k=1}^{m} \mathbb{I}\big(h_i(\mathbf{x}_k) \neq y_k \wedge h_j(\mathbf{x}_k) \neq y_k\big). \end{aligned} \tag{2.14}$$

通过上述方法可以计算基分类器集合中两两之间的多样性程度，则集成器的多样性可以被定义为所有基分类器对的多样性的平均值，即

$$\mathrm{div}(\mathcal{H}) = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathrm{div}(h_i, h_j)\,, \tag{2.15}$$

其中 $\mathrm{div}(\cdot)$ 可以代表上述任意一种成对的多样性度量方法。

2.2.2　非成对的多样性度量

与成对的多样性度量不同的是，非成对的多样性度量试图直接计算整个集成器的多样性，而不是先两两计算之后再取平均值。令 $\rho(\mathbf{x})$ 表示能够正确分类该样本的基分类器的数目，则常见的非成对多样性度量包括以下几种：

- *Kohavi-Wolpert 差异度量* [62] 起源于分类器错误率的偏差方差分解，最早出现在1996年，由 Kohavi et al. [62] 提出。在任意样本 $\mathbf{x}$ 上，被预测的样本标签 y 的可变性被定义为

$$var_{\mathbf{x}} = \frac{1}{2}\left(1 - \sum_{y \in \mathcal{Y}} \mathbf{P}(y|\mathbf{x})^2\right). \tag{2.16}$$

  后来，根据两个分类器的预测输出结果，Kuncheva et al. [3] 修改并提出了度量多样性的可变性，即

$$kw = \frac{1}{mn^2} \sum_{k=1}^{m} \rho(\mathbf{x}_k)\big(n - \rho(\mathbf{x}_k)\big)\,. \tag{2.17}$$

  容易看出，kw 值越大，则多样性程度越大。另外还有一点很有趣的性质是，kw 值仅与通过分歧度量计算所得的集成器多样性相差一个系数，即

$$kw = \frac{n-1}{2n} dis(\mathcal{H})\,, \tag{2.18}$$

  其中 $dis(\mathcal{H}) = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} dis_{ij}$. 因此，若简记 acc 为通过最大化投票所得到的集成器的准确度，那么 acc 与 $dis(\mathcal{H})$ 之间的关系将与 acc 与 kw 之间的关系完全一致。

- *测试者彼此之间的一致性度量* [61] 是一种分类器之间的可信度度量的计算方法，后来被 Kuncheva et al. [3] 用于度量一组分类器之间的意见一致性程度，即

$$\kappa = 1 - \frac{\frac{1}{n} \sum_{k=1}^{m} \rho(\mathbf{x}_k)\big(n - \rho(\mathbf{x}_k)\big)}{m(n-1)\bar{p}(1-\bar{p})}\,, \tag{2.19}$$

  其中，

$$\bar{p} = \frac{1}{mn} \sum_{i=1}^{n} \sum_{k=1}^{m} \mathbb{I}\big(h_i(\mathbf{x}_k) = y_k\big)\,, \tag{2.20}$$

代表着集成器中所有基分类器的准确度的平均值。与 κ 统计量 (κ_p) 相似的是，当基分类器在数据集上的意见完全一致时，$\kappa = 1$；若基分类器之间意见的一致性甚至差于偶然概率时，$\kappa \leqslant 0$．此外，κ 与 kw 之间也存在一定关系，即

$$\kappa = 1 - \frac{n}{(n-1)\bar{p}(1-\bar{p})} kw\,. \tag{2.21}$$

需要注意的一点是，非成对多样性度量 κ 无法通过计算成对多样性度量 κ_p 的平均值来得到 [3]。

- 投票的熵度量 [63-64] 有两种计算方法。其思想主要在于：对于单个样本 $\mathbf{x}_k$ 而言，当基分类器的投票出现平局时，它们之间的意见不一致性程度最大。Cunningham et al. [63] 直接计算了训练样本的香农熵期望，即

$$Ent_{cc} = \frac{1}{m} \sum_{k=1}^{m} \sum_{y \in \mathcal{Y}} -\mathbf{P}(y|\mathbf{x}_k) \log \mathbf{P}(y|\mathbf{x}_k)\,, \tag{2.22}$$

其中 $\mathbf{P}(y|\mathbf{x}_k) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\big(h_i(\mathbf{x}_k) = y\big)$，可以通过基分类器的预测结果来估计，不需要了解该分类器的预测结果的正确性。而 Shipp et al. [64] 假设已知分类器预测结果的正确性，并定义了它们之间的熵为

$$Ent_{sk} = \frac{1}{m} \sum_{k=1}^{m} \frac{\min\big(\rho(\mathbf{x}_k),\, n - \rho(\mathbf{x}_k)\big)}{n - \lceil n/2 \rceil}\,. \tag{2.23}$$

注意 $Ent_{sk} \in [0, 1]$，且其值越大，代表多样性程度越大；$Ent_{sk} = 0$ 则说明基分类器间的意见完全一致，多样性程度为零。

- 难度指数度量 [3, 65] 由 Hansen et al. [65] 提出，后来被 Kuncheva et al. [3] 加以规范。令随机变量 $\mathsf{X} \sim \{0, \frac{1}{n}, \frac{2}{n}, ..., 1\}$ 表示在随机样本 $\mathbf{x}$ 上能够正确分类的个体分类器的比例。则难度指数度量被定义为

$$\theta = variance(\mathsf{X})\,. \tag{2.24}$$

易知 θ 值越小，则多样性程度越大。

- 广义多样性 [66] 被定义为

$$gd = 1 - \frac{p(2)}{p(1)}\,, \tag{2.25}$$

其中，

$$p(1) = \sum_{i=1}^{n} \frac{i}{n}\, p_i\,, \tag{2.26}$$

$$p(2) = \sum_{i=1}^{n} \frac{i}{n} \left(\frac{i-1}{n-1}\right) p_i\,, \tag{2.27}$$

且 p_i 表示在随机样本 $\mathbf{x}$ 上有 i 个分类器分类失误的概率。可知 $gd \in [0, 1]$，且当 $gd = 0$ 时，多样性程度最小，此时 $p(2) = p(1)$；当 $gd = 1$ 时，多样性程度最大，此时 $p(2) = 0$．

- 一致失误多样性 [66] 受启发于广义多样性，被定义为

$$cfd = \begin{cases} 0\,, & 若\, p_0 = 1\,; \\ \dfrac{1}{1-p_0} \displaystyle\sum_{i=1}^{n} \frac{n-i}{n-1}\, p_i\,, & 若\, p_0 < 1\,. \end{cases} \tag{2.28}$$

当所有基分类器的预测结果相同时，$cfd = 0$；当每个基分类器误分类的样本都不同时，$cfd = 1$.

2.2.3　多样性度量的总结

本节介绍了 12 种常见的多样性度量方法，并总结了它们各自的特点，如表 2.2 所示。其中，判断一种多样性度量方法是否对称的方法是检查 $\mathrm{div}(h_i, h_j) = \mathrm{div}(h_j, h_i)$ 是否成立。若等式成立，则认为该多样性度量是对称的度量方法。

表 2.2　常见多样性度量方法的总结 [2-3]

多样性度量	符号	↑ / ↓	是否成对地计算	是否对称
Q 统计量	Q	↓	是	是
κ 统计量	κ_p	↓	是	是
分歧度量	dis	↑	是	是
相关系数度量	ρ	↓	是	是
双重失误度量	df	↓	是	否
Kohavi-Wolpert 差异度量	kw	↑	否	是
测试者彼此之间的一致性度量	κ	↓	否	是
投票的熵度量 (C&C's)	Ent_{cc}	↑	否	是
投票的熵度量 (S&K's)	Ent_{sk}	↑	否	是
难度指数度量	θ	↓	否	否
广义多样性	gd	↑	否	否
一致失误多样性	cfd	↑	否	否

2.3　集成学习中的剪枝方法

给定一组已被训练好的基分类器，集成剪枝的目标是选出其中的一个子集，使之组成一个规模更小的性能更优的集成器。集成剪枝可以缩减集成器所需要的存储空间和计算代价，可以提高集成器的效率 [2]。更重要的一点是，集成剪枝甚至可以获得比原始集成器泛化性能更好的集成器 [45]。关于集成剪枝的研究可以追溯到 1997 年，Margineantu et al. [70] 第一次展示出剪枝后的增强集成器可以获得与剪枝前的原始集成器相差无几的泛化性能。随后，Zhou et al. [45] 提出了基因选择算法，利用误差的偏差方差分解来解释其有效的原因，证实了剪枝

后的集成器确实可以获得更好的泛化性能。但是选择出性能最佳的子集成器是困难的：其中一个难点是对于子集成器的泛化性能的估计；另一个难点在于寻找最优子集是一个计算复杂度为指数级的组合搜索问题，它是一个 NP 完全难问题，即使想获得近似解也并不容易 [71]。对于这样一个难题，大量研究提出了不同的解决方法。目前已被提出的集成方法可大体分为三类，即基于排序的方法、基于聚类的方法和基于优化的方法。

基于排序的剪枝方法是最简单的一种，它将集成器中的基分类器按照某种规则进行排序，然后选择性能最优的一部分作为剪枝后的集成器。不同的方法使用不同的排序规则，这些规则包括：最小化误差的规则，如方向排序 [72]、降低误差剪枝 [70] 等；最大化多样性的规则，如 KL 散度剪枝、Kappa 剪枝 [70] 和互补剪枝 [81] 等；和同时考虑两者的规则，如多样性正则化集成剪枝 [48] 和个体贡献排序剪枝 [30] 等。

基于聚类的剪枝方法首先使用一个聚类算法把原始集成器划分成几个子集，每个子集内的基分类器的性能是相似的；然后在每个子集中挑选出有代表性的基分类器构成剪枝后的集成器，以促进集成器中基分类器之间的多样性。第一阶段的区别主要在于不同的聚类算法，例如 Giacinto et al. [82] 把基分类器预测结果不一致的概率作为距离度量，使用了分层凝聚聚类；Lazarevic et al. [83] 使用基于欧几里得距离的 k 均值聚类；Bakker et al. [84] 使用确定性退火方法进行聚类等。第二阶段的区别在于如何选出子集中的代表性元素。举例来说，Giacinto et al. [82] 在每一类中挑出距离其它类别最远的基分类器；Lazarevic et al. [83] 逐个移除每一类别中准确度较差的基分类器，直到集成器的准确度开始下降为止；而 Bakker et al. [84] 则选择类别内最靠近质心的一个作为被挑选出的基分类器。值得注意的是，基于聚类的集成剪枝方法可以使用并行方法来加速，然而这个性质常常被忽略了。

基于优化的剪枝方法把集成剪枝问题看作是一个子集选择的优化问题，目标是最大化集成器泛化性能的某种度量。这是一个 NP 完全难问题，即使是对于一个规模适中的集成器而言，穷举搜索也并不可行。许多技术被用来解决这一问题，包括基因算法 [73]、贪婪算法 [74]、爬山法 [85]、双目标演化优化 [86] 和帕累托剪枝 [49] 等。

2.4　集成学习在深度学习中的应用

深度学习是当前机器学习领域中炙手可热的研究方向之一。自 2015 年以来，利用深度神经网络解决学习问题的方法大放异彩 [87]，并迅速发展壮大，许多研究领域都受益于深度神经网络的应用，取得了当前最优的实践效果，如语音识

别、目标识别、目标检测、计算机视觉、自然语言处理等 [87]。深度神经网络由若干层非线性操作组成，能够在大规模数据集中发现复杂的结构关系并学习高层级的概念知识。目前也有一些研究致力于深度学习领域中针对集成学习方向进行探索 [88]，这些研究多关注于深度神经网络的集成方向，如深度神经决策森林 [89] 是将卷积神经网络和决策森林联合起来的一种学习方法，利用了随机梯度下降法对决策树进行改造；又如 Wen et al. [90] 利用了卷积神经网络的集成器来解决人脸识别问题；再如 Qiu et al. [91] 利用深度神经网络的集成器来解决时间序列预测问题，它将深度神经网络的输出作为支持向量机的输入，随后输出最终的预测结果；而 Deng et al. [92] 则将卷积神经网络、循环神经网络和全连接网络进行堆叠，用于应对语音识别的挑战。他们的实验结果也都证明了所提出算法的有效性。此外还有一种创新的将集成学习和深度神经网络结合在一起的方法，即深度森林 (gcForest) [93]，它将深度神经网络的神经元替换成了随机森林模型，其中每个随机森林的输出都作为下一层的输入，如此反复；这也是深度森林与上述其他方法的最大不同之处。不仅如此，当输入为高维数据时，深度森林也可以通过多粒度扫描来支持更丰富的特征表达。此外还值得一提的是 Srivastava et al. [94] 所提出的创新的 Dropout 方法，其中一些神经元会在深度神经网络的迭代过程中被丢弃，这也可以视为是一种不同的神经网络的集成方法，只不过它所集成的是不同的被丢弃的神经元。

2.5　本章小结

本章概述了全文的背景和相关知识。我们首先从集成学习中的常见方法出发，介绍了自助法、提升法和堆叠法，并概述了集成器的常见联合策略，如平均法和投票法。随后我们介绍了常见的多样性度量，包括成对的和非成对的度量方法。其中，非成对的度量方法直接计算了整个集成器所有基分类器之间的多样性；而成对的度量方法则需要先两两计算基分类器对之间的多样性，再对这些值求平均，才能获得整个集成器的多样性。接下来，我们总结了集成学习中常见的剪枝策略。最后，我们介绍了集成学习在深度学习里的应用，多关注于深度神经网络的集成方向。

第 3 章　分类问题中集成器的多样性与泛化性能的关系

多样性一直是集成学习领域中的重要概念，然而目前学界尚未对多样性的度量方式达成共识 [2]。对多样性的研究起源于回归集成问题中的误差分解 [31]，其中回归集成器的误差项可以被分解为准确度和多样性两项；但是这种方法只适用于使用平方损失的回归问题，无法直接应用到分类问题中。本章首先提出一种使用 0/1 损失函数的分类集成器的误差分解，并定义分解后的其中一项为多样性度量；随后以边界为桥梁，建立起了多样性与集成器泛化误差之间的关系。通过对多样性与集成泛化性能之间关系的理论分析，我们可以获知在某些情况下，增加多样性可以提高集成器的泛化性能；然后将之应用到集成剪枝任务中，并提出了两种基于多样性的剪枝方法，其中一种剪枝方法可使用现有的任意一种多样性度量。最后，我们在实际数据集上进行实验，通过对比所提出算法和基准对比算法的性能，验证了所提出算法的有效性和优越性。

3.1　背景介绍

集成学习以其非凡的潜力吸引了大量研究者的注意 [2]，并被广泛应用在实际场景中，如情感分类 [20]、图像检索 [25]、图像分类 [20, 26-28]、基因函数预测和 RNA 拼接 [29] 等。一个集成器是经一组已训练好的学习器联合后共同做出决策，而非仅依赖于单个学习器 [34, 95-97]。通常认为，集成器的泛化性能优于单个分类器。Dietterich [1] 指出，分类集成器之所以性能优越，是因为组成它的基分类器是准确且多样化的，即基分类器在新样本上的错误率优于随机猜测，且基分类器之间在新样本上所犯的错误有所不同 [30]。随着基分类器的准确度的提升，它们所组成的集成器的多样性往往会有所下降。因此，如何合理地平衡准确度和多样性就变成了集成学习中的一个重要问题。

然而就多样性的定义和度量问题而言，集成学习领域内尚未达成共识。现有的集成方法通常是通过隐式的或是启发式的方法来构建多样的基分类器，如利用输入数据、输入特征或输出特征等。与之相似的是，多样性的度量也可以在输入空间 (即样本特征) 或输出空间 (即分类结果) 上进行 [34]。目前已有的多样性度量多是在输出空间上进行计算的。研究者们已提出了大量的多样性度量方法，然而究竟何者更优依旧众说纷纭 [15]。这些方法可以被大体地分为两类，即成对的和非成对的多样性度量 [3]。许多学者进行了 “多样性在集成学习中所起到的作用” 方向上的有益探索 [32-33, 35, 37]。在回归问题的集成器上，多样性的定义是基于误差分解 [31] 而得的，其中回归问题的集成器的误差可以被分解为准确

度和多样性两项。经典的模糊分解 [31] 和误差-偏差-协方差分解 [98] 是两种常用的误差分解机制，但是它们都只适用于使用平方损失的回归学习任务。

本章工作受启发于回归问题的集成器的误差分解，并依此提出针对分类问题中使用 0/1 损失函数的集成器的误差分解和一种多样性度量方法。其中，分类集成器的误差被分解为两项，一项是基分类器的平均损失误差，可反映其准确度；另一项则被本章定义为多样性。与回归问题的集成器相似的是，多样性项度量了分类集成器的成员之间的意见差异程度。基于这种多样性度量方法，本章根据 Herbrich et al. [99, 100] 的研究成果提出了多样性与集成器泛化性能之间的一种关系。实验部分以自助法和自适应增强法这两种集成方法为例，验证了所提出的这种关系是否存在。随后本章提出了基于多样性的集成剪枝方法，它利用了前文所提到的多样性和集成器的泛化性能之间的关系，以期尽可能地提高集成器的泛化性能。不止于此，实验结果也可验证所提出的剪枝算法的有效性。

概括来说，本章的贡献主要包括以下四点：

- 基于分类集成问题中的误差分解提出了一种多样性度量方法，用于量化集成器与其基分类器之间的差异程度，并分析了其中各项的性质；
- 探索了所提出的多样性与集成器泛化性能之间的理论关系，并证明了在不同取值范围内，多样性能够对集成器的泛化性能造成不同的影响；
- 根据多样性与集成器泛化性能之间的关系，提出了一种基于多样性的集成剪枝方法，用于从原始集成器中挑选一个性能良好的子集；并提出了一种基于多样性的集成剪枝框架，该框架在进行集成剪枝任务时可使用现有的任意一种多样性度量；
- 通过在实际数据集上与基准算法的实验结果进行对比，验证了所提出方法的有效性和优越性。

本章其余部分安排如下：首先在第 3.2 节对集成学习中的多样性做一简要介绍；随后在第 3.3 节介绍所提出的方法，包括分类问题中集成器的误差分解、多样性度量及其与泛化性能的关系和基于多样性的剪枝方法；之后在第 3.4 节构建实验评估所提出的算法性能，并与基准算法进行比较；最后在第 3.5 节对本章进行总结和展望。

3.2　相关工作

从直观上来讲，多样性被认为是集成器中的基分类器之间的意见差异，也可被称作依赖性、正交性或者互补性 [2-3]。本节将介绍现有的集成方法构建多样化的基分类器的方法，并随后说明当前常见的多样性度量方法，最后总结目前研

究多样性在集成学习中所起到作用的相关工作。

3.2.1　多样性的生成

许多集成方法都通过使用隐式的或启发式的方法来生成多样的基分类器，如使用不同的输入数据或输出特征，或者是使用不同的输出表达等。举例来说，自助法 [4] 和提升法 (包括许多种变体) [16-17, 101] 通过操纵输入数据来提高多样性，即使用训练样本的不同子集；随机森林 [102] 操纵输入数据的特征来创造多样性，并给出有竞争力的结果；旋转森林 [103] 在每个子集上使用了主成分分析来改进结果。除此之外，神经网络集成器也通过使用不同的初始权值、不同的网络体系结构或是不同的学习算法来构建多样化的基分类器。

3.2.2　多样性的度量

多样性既可以在输入空间中度量，也可以在输出空间中度量 [34]。绝大多数研究工作都关注于输出空间中的多样性度量方法 [15, 33, 48, 53-54]，只有一小部分工作关注的是输入空间中的多样性度量方法 [52]。现有的多样性度量方法通常可以被分为成对多样性度量和非成对多样性度量两类 [3]。基于基分类器对上的一致性错误，成对多样性度量表达的是它们同时在新样本上的预测是否相同还是并不相同的行为模式。在这种情形下，整体的多样性就是所有可能的基分类器对的多样性的平均值 [34]。成对多样性度量包括 Q 统计量 [55]、κ 统计量 [56]、分歧度量 [57-58]、相关系数度量 [59] 和双重失误度量 [60] 等。与之相反的是，非成对多样性度量直接计算了一组个体分类器之间的差异，比如说利用它们之间的方差、熵或者是在随机样本上犯错的个体分类器的比例等 [34]。这类多样性度量方法包括测试者彼此之间的一致性度量 [79]、Kohavi-Wolpert 差异度量 [62]、投票的熵度量 [63-64]、难度指数度量 [3, 65]、广义多样性 [66] 和一致失误多样性 [66] 等。除此之外，还有两种多样性度量方法无法被归入上述的两种类别。一种是由 Liu et al. [67] 在负相关学习中所提出的相关惩罚函数，它度量了每个基分类器和整个集成器之间的多样性；另一种是模糊项，度量了每个基分类器与整个集成器输出的偏差 [68]。

3.2.3　多样性的作用

虽然 “多样性在集成学习中扮演着重要的角色” 已成为一种共识，但是 “多样性究竟在集成方法中是如何起作用的” 仍然是一个开放性的问题。在过去的十年间，Brown [32] 从信息理论角度提出，一个集成器中的多样性是以多种层次上的交互形式而存在的。他们的工作启发了 Zhou et al. [33]，使之从互信息的角度

提出了应当最大化共有信息，以便于最小化集成器的预测误差。随后，Yu et al. [35] 提出了一种成对形式的多样性度量，并用到了他们所提出的多样性正则机中，能够缩减假设空间复杂度，这也暗示了控制多样性在集成方法中起到了正则的效果。最近，Jiang et al. [37] 把回归问题中的经典模糊分解扩展到了分类问题中，并证明了模糊项的若干基本性质。

3.3　方法介绍

本节首先探索了分类集成问题中的误差分析，并据此提出了一种新的多样性度量方法；随后根据这种多样性度量方法，采用概率近似正确 学习理论，量化了多样性和集成器泛化性能之间的关系，并推导了相关理论分析；最后利用所提出的多样性与集成器泛化性能之间的关系，提出了两种基于多样性的集成剪枝方法，可以有效提高剪枝后子集成器的性能。

本章所用到的符号可简单罗列如下：向量由粗体小写字母表示，如 $\boldsymbol{x}$ 是一个由 d 元实数值对 $(x_1, x_2, ..., x_d)$ 表示的样本特征；标量由斜体小写字母表示，如 y 为样本标签；随机变量、(向量) 空间和向量空间的子集分别由无衬线大写字母、书法大写字母和罗马大写字母表示，如 $\mathsf{X}, \mathcal{X}$ 和 X. 基分类器由函数表示，如 $f(\cdot)$. 符号 $\mathbf{P}, \mathbb{E}, \mathbb{R}$ 和 $\mathbb{I}$ 分别表示随机变量的概率函数、期望函数、实数空间和指示函数，其中

$$\mathbb{I}(x) = \begin{cases} 1, & \text{若 } x \text{ 命题为真;} \\ 0, & \text{若 } x \text{ 命题为假.} \end{cases} \tag{3.1}$$

3.3.1　集成器的误差分解及其多样性度量

本节采用概率近似正确理论，研究了二分类问题中集成器的误差分解。令 $\mathcal{X}, \mathcal{Y}$ 分别表示样本的输入/特征空间和输出/标签空间，则一个学习任务 $\mathfrak{T}$ 可对应于输入-输出空间 $\mathcal{X} \times \mathcal{Y}$ 上的一个分布。令 $(\boldsymbol{x}, y)$ 表示来自于 $\mathfrak{T}$ 的一个样本，其中 $\boldsymbol{x} \in \mathbb{R}^d$, $y \in \mathcal{Y}$. 则分类问题的目标是利用若干个基分类器所组成的集成器来尽可能地近似真实函数 $f_{true} : \mathcal{X} \mapsto \mathcal{Y}$. 为方便起见，本节只探讨二分类问题，即 $\mathcal{Y} = \{-1, +1\}$, 但相应结论也可扩展到多分类情形。

给定训练样本集 $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_m, y_m)\}$, 且给定一组已训练好的基分类器 $\mathcal{H} = \{h_1(\cdot), h_2(\cdot), ..., h_n(\cdot)\}$, 其中 $h_i(\boldsymbol{x}) \in \mathcal{Y}$, 则使用带权投票法的集成器可表示为

$$h_{ens}(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} w_i h_i(\boldsymbol{x})\right), \tag{3.2}$$

其中 w_i 是基分类器 $h_i(\cdot)$ 的权重，满足 $\sum_{i=1}^{n} w_i = 1$, 且 $\forall\, i \in [n]$, $w_i \geqslant 0$. 当 $\sum_{i=1}^{n} w_i h_i(\boldsymbol{x}) = 0$ 时，意味着在集成器的汇总过程中出现了平局。

已知二分类问题中的标签空间 $\mathcal{Y}=\{-1,+1\}$, 则对于任意基分类器 $h(\cdot)$, 可知

$$h(\boldsymbol{x})\cdot y=\begin{cases}+1\,, & \text{当 } h_i \text{ 将样本正确分类时}\,;\\ -1\,, & \text{当 } h_i \text{ 将样本错误分类时}\,,\end{cases} \tag{3.3}$$

故该项可被定义为该基分类器在该样本上的边界，即

$$\operatorname{margin}(h,\boldsymbol{x})=h(\boldsymbol{x})y\,. \tag{3.4}$$

对多样性的研究来源于回归问题的误差分解 [31, 104-105]，但是它使用的是平方损失函数，并不适用于分类问题。因此我们使用 $0/1$ 损失函数来分解分类集成器的损失误差，其中分类器 $h(\cdot)$ 在任意样本 $\boldsymbol{x}$ 的损失函数被定义为

$$\operatorname{Err}(h,\boldsymbol{x})=\begin{cases}1\,, & \text{当 } h(\boldsymbol{x})y=-1 \text{ 时}\,;\\ 0.5\,, & \text{当 } h(\boldsymbol{x})y=0 \text{ 时}\,;\\ 0\,, & \text{当 } h(\boldsymbol{x})y=1 \text{ 时}\,,\end{cases} \tag{3.5}$$

这也是铰链损失函数的离散化表达，并满足

$$\operatorname{Err}(h,\boldsymbol{x})=-\frac{1}{2}\big(h(\boldsymbol{x})y-1\big)\,. \tag{3.6}$$

受启发于 [37, 104]，我们提出了分类问题中的集成器的误差分解。

定理 3.1 (分类集成器的误差分解)　给定一组已训练好的基分类器 $\mathcal{H}=\{h_1(\cdot),h_2(\cdot),...,h_n(\cdot)\}$, 所构成的使用带权投票法的集成器如式 (3.2) 所示，其中权重满足 $\sum_{i=1}^{n}w_i=1$, 且 $\forall\, i\in[n]$, $w_i\geqslant 0$. 则集成器的 $0/1$ 损失可以被分解为两项，即

$$\operatorname{Err}(h_{ens},\boldsymbol{x})=\sum_{i=1}^{n}w_i\operatorname{Err}(h_i,\boldsymbol{x})-\frac{1}{2}\left(\operatorname{margin}(h_{ens},\boldsymbol{x})-\sum_{i=1}^{n}w_i\cdot\operatorname{margin}(h_i,\boldsymbol{x})\right)\,. \tag{3.7}$$

通过计算各项在样本空间中的期望，可得集成器的泛化误差分解为

$$\bar{G}=\bar{A}-\bar{D}\,, \tag{3.8}$$

其中，

$$\bar{G}=\mathbb{E}_{\mathcal{D}}\big(\operatorname{Err}(h_{ens},\boldsymbol{x})\big)\,, \tag{3.9a}$$

$$\bar{A}=\sum_{i=1}^{n}w_i\,\mathbb{E}_{\mathcal{D}}\big(\operatorname{Err}(h_i,\boldsymbol{x})\big)\,, \tag{3.9b}$$

$$\bar{D}=\frac{1}{2}\,\mathbb{E}_{\mathcal{D}}\big(\operatorname{margin}(h_{ens},\boldsymbol{x})\big)-\frac{1}{2}\sum_{i=1}^{n}w_i\,\mathbb{E}_{\mathcal{D}}\big(\operatorname{margin}(h_i,\boldsymbol{x})\big)\,, \tag{3.9c}$$

且 $\mathbb{E}_{\mathcal{D}}(\cdot)$ 代表在样本空间上的期望。

证明　对于任意样本 $\boldsymbol{x}$, 一个集成器的损失误差被定义为

$$\mathrm{Err}(h_{ens}, \boldsymbol{x}) = \mathrm{Err}\left(\mathrm{sign}\left(\sum_{i=1}^{n} w_i\, h_i(\boldsymbol{x})\right) y\right). \tag{3.10}$$

受启发于回归问题的误差分解，可得集成器的损失误差与基分类器的平均损失误差之间的差异为

$$\begin{aligned}\mathrm{Err}(h_{ens}, \boldsymbol{x}) - \sum_{i=1}^{n} w_i\ \mathrm{Err}(h_i, \boldsymbol{x}) &= \sum_{i=1}^{n} w_i \Big(\mathrm{Err}(h_{ens}, \boldsymbol{x}) - \mathrm{Err}(h_i, \boldsymbol{x})\Big) \\ &= -\frac{y}{2}\sum_{i=1}^{n} w_i\big(h_{ens}(\boldsymbol{x}) - h_i(\boldsymbol{x})\big)\,, \end{aligned} \tag{3.11}$$

且该式可被重写为

$$\mathrm{Err}(h_{ens}, \boldsymbol{x}) = \sum_{i=1}^{n} w_i\ \mathrm{Err}(h_i, \boldsymbol{x}) - \frac{1}{2}\sum_{i=1}^{n} w_i\big(h_{ens}(\boldsymbol{x}) - h_i(\boldsymbol{x})\big)y\,. \tag{3.12}$$

其中，右式第一项是基分类器的带权平均损失误差；而右式第二项则被定义为多样性项，用以度量集成器 $h_{ens}(\cdot)$ 与其基分类器之间的差异，且该项可被重写为

$$\frac{1}{2}\sum_{i=1}^{n} w_i\big(h_{ens}(\boldsymbol{x}) - h_i(\boldsymbol{x})\big)y = \frac{1}{2}\sum_{i=1}^{n} w_i\big(\,\mathrm{margin}(h_{ens}, \boldsymbol{x}) - \mathrm{margin}(h_i, \boldsymbol{x})\big)\,. \tag{3.13}$$

注意此时的多样性项是针对于某一样本 $\boldsymbol{x}$ 的多样性，故我们将其定义为

$$\mathrm{div}(h_{ens}, \boldsymbol{x}) = \frac{1}{2}\,\mathrm{margin}(h_{ens}, \boldsymbol{x}) - \frac{1}{2}\sum_{i=1}^{n} w_i \cdot \mathrm{margin}(h_i, \boldsymbol{x})\,. \tag{3.14}$$

至此，我们获得了在单一样本 $\boldsymbol{x}$ 上的集成器的误差分解，形如式 (3.7) 所示。随后，通过计算各项在样本空间中的期望，可得集成器在整个样本空间中的误差分解，形如式 (3.8) 所示。　□

由此，我们可以根据定理 3.1 获得一种多样性度量方法，如式 (3.14) 和式 (3.9c) 所示，它可以作为独立变量来使用，用于分析和集成器泛化性能之间的关系。在探索多样性与集成器泛化性能之间的关系之前，我们先来分析一下定理 3.1 中各项的性质。

推论 3.2 ($\bar{G}, \bar{A}$ 和 $\bar{D}$ 都只依赖于 $\mathrm{div}(h_{ens}, \boldsymbol{x})$)　考虑二分类问题。给定一组已训练好的基分类器 $\mathcal{H} = \{h_1(\cdot), h_2(\cdot), ..., h_n(\cdot)\}$, 所构成的使用带权投票法的集成器如式 (3.2) 所示，其中权重满足 $\sum_{i=1}^{n} w_i = 1$, 且 $\forall\, i \in [n]$, $w_i \geqslant 0$. 那么该集成器的泛化误差 $\bar{G}$、准确度 $\bar{A}$ 和多样性 $\bar{D}$ 都依赖于且只依赖于 $\mathrm{div}(h_{ens}, \boldsymbol{x})$, 即

$$\bar{G} = \frac{1}{2}\Big(1 - \mathbb{E}_{\mathcal{D}}\big(\,\mathrm{sign}\,(\lambda - 2\,\mathrm{div}(h_{ens}, \boldsymbol{x}))\big)\Big)\,, \tag{3.15a}$$

$$\bar{A} = \frac{1}{2}\Big(1 - \mathbb{E}_{\mathcal{D}}\big(\lambda - 2\,\mathrm{div}(h_{ens}, \boldsymbol{x})\big)\Big)\,, \tag{3.15b}$$

$$\bar{D} = \mathbb{E}_{\mathcal{D}}\big(\,\mathrm{div}(h_{ens}, \boldsymbol{x})\big)\,, \tag{3.15c}$$

其中，

$$\lambda = \begin{cases} 1\,, & \text{若 } \operatorname{div}(h_{ens}, \boldsymbol{x}) \in (0, \frac{1}{2})\,; \\ 0\,, & \text{若 } \operatorname{div}(h_{ens}, \boldsymbol{x}) = 0\,; \\ -1\,, & \text{若 } \operatorname{div}(h_{ens}, \boldsymbol{x}) \in (-\frac{1}{2}, 0)\,. \end{cases} \tag{3.16}$$

证明　如前所述，单个分类器 $h(\cdot)$ 的损失误差为

$$\begin{aligned} \operatorname{Err}(h, \boldsymbol{x}) =& -\frac{1}{2}\big(h(\boldsymbol{x})y - 1\big) \\ =& -\frac{1}{2}\big(\operatorname{margin}(h, \boldsymbol{x}) - 1\big)\,. \end{aligned} \tag{3.17}$$

因此，由定理 3.1 可知

$$\bar{G} = \frac{1}{2}\Big(1 - \mathbb{E}_{\mathcal{D}}\big(\operatorname{margin}(h_{ens}, \boldsymbol{x})\big)\Big)\,, \tag{3.18a}$$

$$\bar{A} = \frac{1}{2}\left(1 - \mathbb{E}_{\mathcal{D}}\left(\sum_{i=1}^{n} w_i \cdot \operatorname{margin}(h_i, \boldsymbol{x})\right)\right)\,, \tag{3.18b}$$

$$\bar{D} = \frac{1}{2}\sum_{i=1}^{n} w_i \cdot \mathbb{E}_{\mathcal{D}}\big(\operatorname{margin}(h_{ens}, \boldsymbol{x}) - \operatorname{margin}(h_i, \boldsymbol{x})\big)\,. \tag{3.18c}$$

而由式 (3.14) 可知，

$$\begin{aligned} \operatorname{div}(h_{ens}, \boldsymbol{x}) =& \frac{1}{2}\operatorname{sign}\left(\sum_{i=1}^{n} w_i h_i(\boldsymbol{x}) \cdot y\right) - \frac{1}{2}\sum_{i=1}^{n} w_i \cdot \operatorname{margin}(h_i, \boldsymbol{x}) \\ =& \frac{1}{2}\operatorname{sign}\left(\sum_{i=1}^{n} w_i \cdot \operatorname{margin}(h_i, \boldsymbol{x})\right) - \frac{1}{2}\sum_{i=1}^{n} w_i \cdot \operatorname{margin}(h_i, \boldsymbol{x})\,, \end{aligned} \tag{3.19}$$

则定义集成器的平均边界为

$$\overline{\operatorname{margin}}(h_{ens}, \boldsymbol{x}) \overset{\text{def}}{=} \sum_{i=1}^{n} w_i \cdot \operatorname{margin}(h_i, \boldsymbol{x})\,, \tag{3.20}$$

并简记其为

$$\overline{\operatorname{margin}}(h_{ens}, \boldsymbol{x}) = \lambda - 2\operatorname{div}(h_{ens}, \boldsymbol{x})\,, \tag{3.21}$$

其中 λ 的定义如式 (3.16) 所示。将式 (3.21) 代入式 (3.18)，即可得推论 3.2。　□

3.3.2　集成器的多样性与泛化性能的关系

在集成学习领域，多样性与集成器泛化性能之间的关系仍然是一个开放性的问题。在现实世界的模式识别问题中，一些学者仍对 “多样性是否真能在分类集成器中起到作用” 这一观点持否定态度 [3]；也有一些学者认为基分类器之间的多样性对于集成器而言是一个重要的问题 [48]。这是两种完全相反的观点，因此，两种观点都能找到可以支持其观点的研究和实验成果是一件非常奇怪的事。比如说，一些研究通过改变集成器的多样性，展示了多样性和集成器泛化误差之间的较小的关联性 [3]。同时，也有一些研究表明了截然相反的观点 [35]。因此，探索多样性究竟能否对集成器的泛化性能起到积极促进作用就成了一个重要的

问题。Herbrich et al. [99, 100] 的研究成果表明了集成器泛化性能和边界之间存在一定的关系，而上节所提出的多样性度量与边界有关，因此，本节以边界为桥梁，建立起集成器多样性与其泛化性能之间的关系，如图 3.1 所示。

仍以 $\mathcal{X}$ 和 $\mathcal{Y}$ 分别表示输入/特征空间和输出/标签空间，且训练样本集 $\mathcal{D}$ 如前所述，其中任意样本 $(\boldsymbol{x}, y)$ 满足 $(\boldsymbol{x}, y) \sim (\mathsf{X}, \mathsf{Y}) \in (\mathcal{X}, \mathcal{Y})$. 样本随机变量均来自于某一未知概率分布 $\mathbf{P}_{\mathsf{Y}|\mathsf{X}}\mathbf{P}_{\mathsf{X}}$, 且相互之间独立同分布。本节只考虑形如式 (3.2) 的线性分类器作为分类集成器，即函数假设空间为

$$\mathcal{F} = \{\boldsymbol{x} \mapsto \operatorname{sign}(\langle \mathbf{w}, \phi(\boldsymbol{x}) \rangle) \,|\, \mathbf{w} \in \mathcal{W}\}, \tag{3.22}$$

其中 $\phi(\boldsymbol{x}) = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), ..., h_n(\boldsymbol{x})]^{\mathsf{T}}$, 且 $\mathbf{w} = [w_1, w_2, ..., w_n]^{\mathsf{T}}$. 注意 $\sum_{i=1}^{n} w_i = 1\,(\forall\, i \in [n],\, w_i \geqslant 0)$ 这一条件使得从函数 $h_{ens}(\boldsymbol{x}, \mathbf{w})$ 到其权重参数 $\mathbf{w}$ 构成了一个一一映射，其中 $h_{ens} \in \mathcal{F}$, $\mathbf{w} \in \mathcal{W}$. 故后文中可用权重 $\mathbf{w}$ 来指代相应的函数 $h_{ens}(\boldsymbol{x}, \mathbf{w})$. 假设真实函数存在于该假设空间内，即 $\mathbf{w}^* \in \mathcal{W}$, 那么构成了一个概率近似正确似然，即

$$\mathbf{P}_{\mathsf{Y}|\mathsf{X}=\boldsymbol{x}}(y) = \mathbb{I}\big(y = \operatorname{sign}\left(\langle \mathbf{w}^*, \phi(\boldsymbol{x}) \rangle\right)\big). \tag{3.23}$$

由于真实函数 $\mathbf{w}^*$ 存在于假设空间中，则存在一个版本空间 $V(\mathcal{D}) \subseteq \mathcal{W}$, 使得

$$V(\mathcal{D}) = \{\mathbf{w} \in \mathcal{W} \,|\, \forall\, (\boldsymbol{x}, y) \in \mathcal{D},\, \operatorname{sign}(\langle \mathbf{w}, \phi(\boldsymbol{x}) \rangle) = y\}. \tag{3.24}$$

则一致假设 $\mathbf{w} \in V(\mathcal{D})$ 的真实误差风险 $R(\mathbf{w})$ 为

$$R(\mathbf{w}) = \mathbb{E}_{\mathsf{XY}}\big[\mathbb{I}\big(\operatorname{sign}(\langle \mathbf{w}, \phi(\mathsf{X}) \rangle) \neq \mathsf{Y}\big)\big]. \tag{3.25}$$

由于版本空间 $V(\mathcal{D})$ 中的函数的错误率在训练集 $\mathcal{D}$ 上并无区别，所以 Herbrich et al. [99, 100] 在其研究中引入了集成器 $\mathbf{w}$ 的边界 $\gamma(\mathbf{w})$ 的概念，即

$$\gamma(\mathbf{w}) = \min_{(\boldsymbol{x}, y) \in \mathcal{D}} y \langle \mathbf{w}, \phi(\boldsymbol{x}) \rangle. \tag{3.26}$$

随后，Herbrich et al. [99, 100] 提出了集成器的泛化误差 $R(\mathbf{w})$ 可由其边界 $\gamma(\mathbf{w})$ 界定，如引理 3.3 所示。

引理 3.3 (集成器的泛化误差与其边界之间的关系 [100])　对于任意概率分布 $\mathbf{P}_{\mathsf{X}}$, 如 $\mathbf{P}_{\mathsf{X}}(\|\phi(\boldsymbol{x})\| \leqslant \delta) = 1$, 以及对于任意小数 $\xi \in (0, 1]$, 以至少 $(1 - \xi)$ 的概率从训练样本集 $\mathcal{D} \in (\mathcal{X}, \mathcal{Y})^m$ 中随机抽取，那么对于任意一致的分类集成器 $\mathbf{w} \in V(\mathcal{D})$, 且满足存在正边界 $\gamma(\mathbf{w}) > \sqrt{32\delta^2/m}$, 其泛化误差风险 $R(\mathbf{w})$ 可由下式所界定，即

$$R(\mathbf{w}) \leqslant \frac{2}{m}\left(\kappa(\mathbf{w}) \log_2\left(\frac{8em}{\kappa(\mathbf{w})}\right) \log_2(32m) + \log_2\left(\frac{2m}{\xi}\right)\right), \tag{3.27}$$

其中 $\kappa(\mathbf{w}) = \left\lceil \left(\frac{8\delta}{\gamma(\mathbf{w})}\right)^2 \right\rceil$.

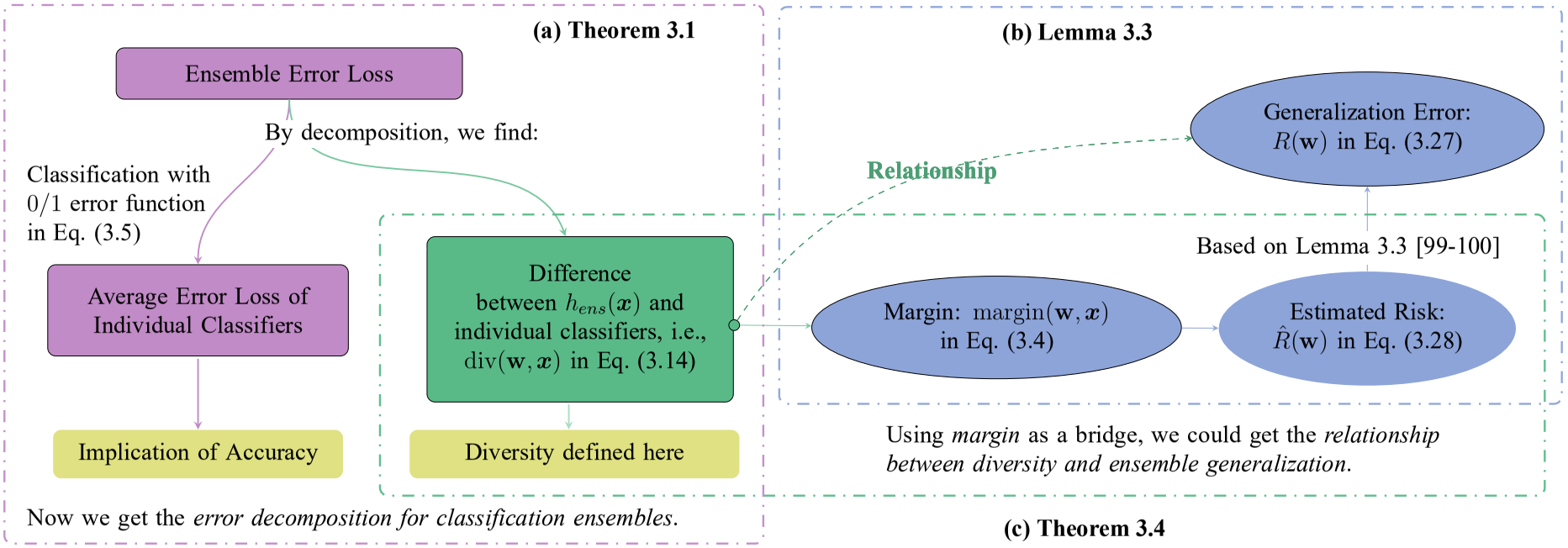


图 3.1　本章所提出的分析方法的说明图解

注：(a) 分类集成器的误差分解，即定理 3.1。(b) 引理 3.3 [99-100]。(c) 集成器的多样性与其泛化性能之间的关系，即定理 3.4。Herbrich et al. [99, 100] 提出了集成器的边界与其泛化性能之间的关系，即引理 3.3；而本章所提出的多样性度量又与边界之间存在关系，因此以边界为桥梁，我们可以获知集成器的多样性与其泛化性能之间的关系，即定理 3.4。注意从 A 到 B 的箭头说明 B 与 A 有关，其中实线箭头代表直接关系，而虚线箭头代表间接关系。

受启发于引理 3.3 [100]，本节提出了集成器的多样性与其泛化性能之间的关系，如定理 3.4 所示。值得一提的是，集成器的边界 $\gamma(\mathbf{w})$ 可以与上节所提出的多样性度量 $\operatorname{div}(h_{ens}, \boldsymbol{x})$ 建立关系，这也正是启发我们探索并建立集成器多样性与泛化性能之间关系的灵感来源，如图 3.1 所示。为简化分析，本节仅考虑式 (3.27) 中最重要的非常数项，即 $\kappa(\mathbf{w})\log_2\left(\frac{8em}{\kappa(\mathbf{w})}\right)$, 记为 $\hat{R}(\mathbf{w})$.

定理 3.4 (集成器的多样性与其泛化性能之间的关系)　对于任意概率分布 $\mathbf{P}_{\mathsf{X}}$, 如 $\mathbf{P}_{\mathsf{X}}(\|\phi(\boldsymbol{x})\| \leqslant \delta) = 1$, 以及对于任意小数 $\xi \in (0, 1]$, 以至少 $(1-\xi)$ 的概率从训练样本集 $\mathcal{D} \in (\mathcal{X}, \mathcal{Y})^m$ 中随机抽取，且抽取过程以 ε 的概率受噪声影响。那么对于任意一致的分类集成器 $\mathbf{w} \in V(\mathcal{D})$, 其泛化误差 $R(\mathbf{w})$ 的上界的变化趋势与其估计值 $\hat{R}(\mathbf{w})$ 相同，即

$$\hat{R}(\mathbf{w}) = \left(\frac{8\delta}{\gamma(\mathbf{w})}\right)^2 \log_2\left(8em\left(\frac{\gamma(\mathbf{w})}{8\delta}\right)^2\right), \tag{3.28}$$

$$\gamma(\mathbf{w}) = \min_{(\boldsymbol{x},y)\in\mathcal{D}} (1-2\varepsilon)(\lambda - 2\operatorname{div}(h_{ens}, \boldsymbol{x})), \tag{3.29}$$

其中 λ 的定义如式 (3.16) 所示。

证明　由于真实数据中存在噪声，对于任意样本 $\boldsymbol{x} \in \mathcal{D}$, 令 z 表示该样本观测到的标签，而 y 表示其真实未知样本。因此，集成器在训练数据集 $\mathcal{D}$ 上的错误率可记作 $\mathbf{P}(z \neq y \,|\, \boldsymbol{x} \in \mathcal{D}) \overset{\text{def}}{=} \varepsilon$, 且分类器 $h(\cdot)$ 的错误率可记作 $\mathbf{P}(h(\boldsymbol{x}) \neq z \,|\, \boldsymbol{x} \in \mathcal{D}) \overset{\text{def}}{=} \theta$. 由于训练集受噪声和其他因素的影响，我们相信 $\theta \geqslant \varepsilon$. 因此我们可以用能够观测到的标签 z 来估计真实标签 y, 即 $\hat{y} = (1-\varepsilon)z + \varepsilon(-z) = (1-2\varepsilon)z$. 因此对于给定数据集 $\mathcal{D} = \{(\boldsymbol{x}_i, z_i) \,|\, i \in [m]\}$, 训练若干个基分类器用于构建集成器，即 $\mathcal{H} = \{h_j(\cdot) \,|\, j \in [n]\}$. 由式 (3.4)，基分类器 h_j 在样本 $\boldsymbol{x}_i$ 上的边界为

$$\operatorname{margin}(h_j, \boldsymbol{x}_i) = h_j(\boldsymbol{x}_i) z_i . \tag{3.30}$$

根据式 (3.20)–(3.21) 以及式 (3.26), 可得集成器的边界为

$$\gamma(\mathbf{w}) = \min_{1\leqslant i\leqslant m} \big(1-2\varepsilon\big)\big(\lambda - 2\operatorname{div}(h_{ens}, \boldsymbol{x}_i)\big), \tag{3.31}$$

且该项主要受组成集成器的基分类器之间意见的差异程度 (即多样性) 影响。

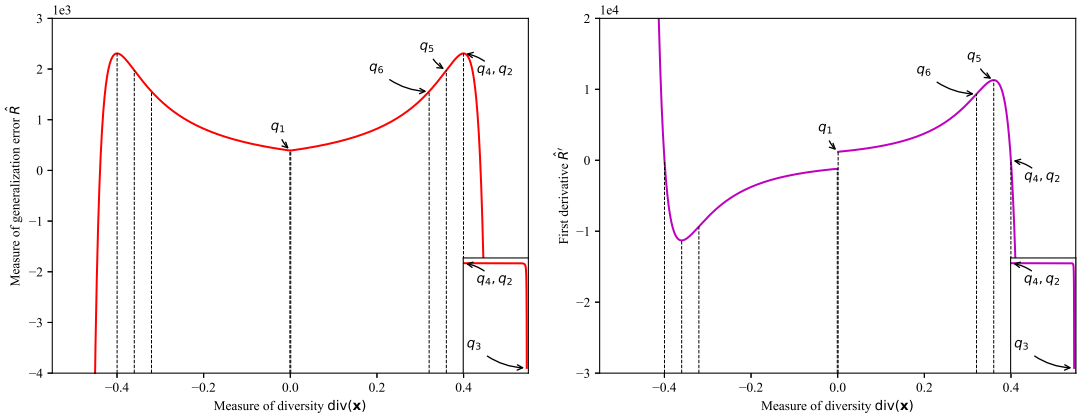
令 $\boldsymbol{x}^*$ 表示集成器达到最小边界值的样本点，即

$$\gamma(\mathbf{w}) = \big(1-2\varepsilon\big)\big(\lambda - 2\operatorname{div}(h_{ens}, \boldsymbol{x}^*)\big), \tag{3.32}$$

则由式 (3.27) 可得 $\kappa(\mathbf{w})$. 为简化分析，本节只考虑式 (3.27) 中最重要的一项，即 $\kappa(\mathbf{w})\log_2\left(\frac{8em}{\kappa(\mathbf{w})}\right)$, 记为 $\hat{R}(\mathbf{w})$, 即

$$\begin{aligned}\hat{R}(\mathbf{w}) &\overset{\text{def}}{=} \kappa(\mathbf{w})\log_2\left(\frac{8em}{\kappa(\mathbf{w})}\right) \\ &\approx \left(\frac{8\delta}{(1-2\varepsilon)(\lambda - 2\operatorname{div}(h_{ens}, \boldsymbol{x}^*))}\right)^2 \log_2\left(8em\left(\frac{(1-2\varepsilon)(\lambda - 2\operatorname{div}(h_{ens}, \boldsymbol{x}^*))}{8\delta}\right)^2\right),\end{aligned} \tag{3.33}$$

且 $\hat{R}(\mathbf{w})$ 与 $R(\mathbf{w})$ 具有相同的变化趋势。　□



(a) 集成器的多样性与其泛化误差风险的关系　　(b) 集成器的多样性与泛化误差一阶导的关系

图 3.2　以多样性为自变量，集成器泛化误差风险的估计值及其一阶导数的图示说明

注：实际上，图中的端点 (如 q_4, q_5 和 q_6 等) 均指横轴上的对应位置。在此图示中，所用到的参数是：$\delta = 1$, $\varepsilon = 0.01$, $m = 200$.

由定理 3.4 可知，集成器的多样性可用来量化其泛化误差风险。式 (3.33) 反映了集成器的多样性 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*) \in (-\frac{1}{2}, \frac{1}{2})$ 与泛化误差风险 $R(\mathbf{w})$ 之间的量化关系。根据式 (3.33)，对泛化误差的估计值 $\hat{R}(\mathbf{w})$ 关于多样性 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*)$ 求偏导，可得其一阶偏导数以及二阶偏导数，即

$$\hat{R}' = 4\left(\frac{8\delta}{1-2\varepsilon}\right)^2 \frac{1}{(\lambda - 2\,\mathrm{div}(h_{ens}, \boldsymbol{x}^*))^3} \log_2\left(8m\left(\frac{(1-2\varepsilon)(\lambda - 2\,\mathrm{div}(h_{ens}, \boldsymbol{x}^*))}{8\delta}\right)^2\right), \tag{3.34}$$

$$\hat{R}'' = \frac{8}{\ln 2}\left(\frac{8\delta}{1-2\varepsilon}\right)^2 \frac{1}{(\lambda - 2\,\mathrm{div}(h_{ens}, \boldsymbol{x}^*))^4} \cdot \left(3\ln\left(8em\left(\frac{(1-2\varepsilon)(\lambda - 2\,\mathrm{div}(h_{ens}, \boldsymbol{x}^*))}{8\delta}\right)^2\right) - 2\right). \tag{3.35}$$

又已知 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*) \in (-\frac{1}{2}, \frac{1}{2})$, 故泛化误差上界的估计值 $\hat{R}(\mathbf{w})$ 的图象关于纵轴对称，如图 3.2 所示。在后文中，我们主要以区间 $(0, \frac{1}{2})$ 为例，来分析集成器的多样性 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*)$ 与其泛化误差之间的关系。

至此，我们已获知集成器的多样性 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*)$ 与其泛化误差风险估计值 $\hat{R}(\mathbf{w})$ 之间的关系，如式 (3.33) 所示，则根据一阶导数和二阶导数[①]可以分析其单调性和增减性。以横轴正半轴为例，令 q_1 和 q_3 分别为自变量 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*)$ 取值的左端点和右端点，则函数拐点 q_2 可由令 $\hat{R}'(\mathbf{w}) = 0$ 而得，也可记作 q_4. 则函数图象 $\hat{R}(\mathbf{w})$ 的单调递增区间为 (q_1, q_2), 且其单调递减区间为 (q_2, q_3). 令 $\hat{R}''(\mathbf{w}) = 0$, 可得一阶导数的驻点，记作 q_5, 用于分析函数 $\hat{R}(\mathbf{w})$ 的增减性；令 $\hat{R}'''(\mathbf{w}) = 0$, 可得二阶导数的驻点，记作 q_6, 用于分析函数 $\hat{R}(\mathbf{w})$ 的凹凸性。这些端点的值分

[①]函数 $f(x)$ 的一阶导数可记作 $f'(x)$, 是该函数在自变量取值为 x 时的切线斜率。通过分析函数的一阶导数，可获知该函数的增减性以及增减区间。函数 $f(x)$ 的二阶导数可记作 $f''(x)$, 对其分析可获知该函数的导数的增减性以及增减区间 [106]。

表 3.1　在多样性区间上的集成器泛化误差风险的变化趋势表

$\mathrm{div}(h_{ens}, \boldsymbol{x}^*)$	$\hat{R}(\mathbf{w})$	$\hat{R}'(\mathbf{w})$	$\Delta\hat{R}$	$\Delta\hat{R}'$
$(-q_3, -q_2)$	↗ 凸	↘ 凹	减小	增大
$(-q_2, -q_5)$	↘ 凸	↘ 凹	减小	增大
$(-q_5, -q_6)$	↘ 凹	↗ 凹	增大	增大
$(-q_6, -q_1)$	↘ 凹	↗ 凸	增大	减小
(q_1, q_6)	↗ 凹	↗ 凹	增大	增大
(q_6, q_5)	↗ 凹	↗ 凸	增大	减小
(q_5, q_2)	↗ 凸	↘ 凸	减小	减小
(q_2, q_3)	↘ 凸	↘ 凸	减小	减小

注：第一列是多样性 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*)$ 的取值范围，第二至三列分别是集成器泛化误差风险的估计值 $\hat{R}(\mathbf{w})$ 及其一阶导数 $\hat{R}'(\mathbf{w})$. 注意集成器 $h_{ens}(\cdot)$ 的泛化误差 $R(\mathbf{w})$ 与其泛化误差的估计值 $\hat{R}(\mathbf{w})$ 关于多样性 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*)$ 的变化趋势相同。第四至五列分别是泛化误差估计值与其一阶导数在取位于该区间内的自变量值时的增量。

别为

$$q_1 = \varepsilon\,, \quad q_3 = \frac{1}{2}\left(1 - \frac{\varepsilon}{1-2\varepsilon}\right), \tag{3.36a}$$

$$q_2 = q_4 = \frac{1}{2}\left(1 - \frac{\delta}{1-2\varepsilon}\sqrt{\frac{8}{m}}\right), \tag{3.36b}$$

$$q_5 = \frac{1}{2}\left(1 - \frac{\delta}{1-2\varepsilon}\sqrt{\frac{8}{m}e^{\frac{2}{3}}}\right), \tag{3.36c}$$

$$q_6 = \frac{1}{2}\left(1 - \frac{\delta}{1-2\varepsilon}\sqrt{\frac{8}{m}e^{\frac{7}{6}}}\right), \tag{3.36d}$$

其中包含一个隐含条件，即

$$\frac{\varepsilon}{1-2\varepsilon} \leqslant \frac{\delta}{1-2\varepsilon}\sqrt{\frac{8}{m}} \leqslant 1 - 2\varepsilon\,. \tag{3.37}$$

注意 $q_1 < q_2 < q_3$ 且 $q_6 < q_5 < q_4 = q_2$. 则多样性 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*)$ 的定义域可以被这些端点划分为八个区间，即如表 3.1 中所示。至此，我们就可以具体分析在不同区间内集成器的多样性对其泛化性能的影响了。

1) 当 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*) = 0$ 时，本章只考虑两种情形，即 “所有基分类器都将该样本 $\boldsymbol{x}^*$ 分类正确” 或 “所有基分类器都将该样本 $\boldsymbol{x}^*$ 分类错误”。不考虑平局。
2) 在实际情况中，绝大部分 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*)$ 为负的情形都意味着 “该分类集成器将对应于被选择出的 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*)$ 所对应的样本分类错误”。注意在有限次数的实验中，当训练样本集的基数 m 满足式 (3.37) 时，就很少会出现 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*) > 0$ 的情形。当 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*) < 0$ 时，增加多样性能够对提高集成器的泛化性能有促进作用，因此在这种情形下应当提高多样性。

3) 当 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*)$ 落到横轴正半轴区域时，分类集成器能够将相应的样本分类正确。而在接近于零的区域，能够正确分类该样本的基分类器将占据较大优势，且 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*)$ 愈接近于零，这些基分类器所占据的优势愈大。而在接近于 $\frac{1}{2}$ 的区域，能够正确分类该样本的基分类器将占据较小的优势，且 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*)$ 愈接近于 $\frac{1}{2}$，这些基分类器所占据的优势越小。在这种情形下，应当减少多样性以提高集成器的泛化性能，其基本思想是为了维持集成器能够正确分类该样本的能力。总之，在这种情形下增加多样性会削减能够正确分类该样本的基分类器的优势，而这也正是我们所极力避免的。

4) 当 $\mathrm{div}(h_{ens}, \boldsymbol{x}^*)$ 落到横轴负半轴区域时，分类集成器无法对相应的样本进行正确分类。可类比上一情形做类似分析。

根据上述分析可以发现：集成器的多样性与其泛化性能之间的关系会随着多样性落在不同的取值区间内而变化。概括地说，由表 3.1 可知，有两个特殊的区间值得引起我们的重视，即 $(-q_5, -q_6)$ 和 $(-q_6, -q_1)$. 在这两个区间内，增加多样性对提高集成器的泛化性能有积极影响，此时为了提高集成器的性能，有必要增加集成器的多样性。但是在其他区间内，增加多样性并非总能得到性能更好的集成器，所以完全可以保持当前的集成器状态并继续当下的学习过程。

3.3.3　基于多样性的集成剪枝方法及框架

受节 3.3.2 中集成器的多样性与泛化性能之间关系的启发，本节提出了一个“基于多样性的集成剪枝方法 *(Ensemble Pruning Based on Diversity, EPBD)*”，如算法 3.1 所示。*EPBD* 旨在同时利用多样性和准确度，以缩减集成器的规模并维持集成器的性能不衰退。

在算法 3.1 中，其输入有四：其一为有效数据样本的训练集 $\mathcal{D}$; 其二为已训练好的集成器 $\mathcal{H}$, 即一组基分类器的集合，这些基分类器使用带权投票法的方法来构建原始集成器，如式 (3.2) 所示；其三为剪枝比例 ρ, 表示在原始集成器中有多少比例的基分类器将被保留在剪枝后的子集成器中；其四为平衡因子 λ, 用于平衡准确度和多样性这两大要素。注意这里为了简化计算，每个基分类器的权重系数都被设为 $1/n$. 而算法 3.1 的输出是剪枝后的子集成器 $\mathcal{P}$, 其初始值为空集，且满足 $|\mathcal{P}| \leqslant \lceil \rho \cdot |\mathcal{H}| \rceil$. 该算法的终止条件是当子集成器的规模达到预期大小 (即 $|\mathcal{P}| > \lceil \rho \cdot |\mathcal{H}| \rceil$) 时，或 $\mathcal{H}$ 中剩下的基分类器都无法正确分类样本点 $(\boldsymbol{x}, y)$ 时。

EPBD 算法的目标是获得一个性能尽可能优的子集成器。如算法 3.1 所示，它首先挑出那些多样性和准确度都尽可能高的基分类器，如第 3 行的目的在于找到当前边界值最小也即多样性最大的样本点；第 4 行的目的在于尽可能地增加准确度，所使用的排序准则主要是基分类器的准确度和平衡因子 λ 与多样性

Data: 训练数据集 $\mathcal{D}=\{(\boldsymbol{x}_1,y_1),...,(\boldsymbol{x}_m,y_m)\}$;
　　原始集成器 $\mathcal{H}=\{h_1(\cdot),...,h_n(\cdot)\}$; 剪枝比例 ρ; 平衡因子 λ
Result: 剪枝后的子集成器 $\mathcal{P}$, 满足 $\mathcal{P}\subset\mathcal{H}$
1 令 $\mathcal{P}=\emptyset$; // 初始化
2 **repeat**
3　　搜索满足式 (3.31) 的样本点，记为 $(\boldsymbol{x},y)$;
4　　将 $\mathcal{H}$ 中能够正确分类该样本的基分类器按性能递减的顺序排序;
5　　选择上一步中最好的基分类器 $h(\cdot)$, 将其从 $\mathcal{H}$ 中移入 $\mathcal{P}$;
6 **until** 终止条件满足;

算法 3.1: 基于多样性的集成剪枝方法 (*EPBD*)

Data: 训练数据集 $\mathcal{D}=\{(\boldsymbol{x}_1,y_1),...,(\boldsymbol{x}_m,y_m)\}$;
　　原始集成器 $\mathcal{H}=\{h_1(\cdot),...,h_n(\cdot)\}$; 阈值 ρ; 平衡因子 λ
Result: 剪枝后的子集成器 $\mathcal{P}$, 满足 $\mathcal{P}\subset\mathcal{H}$
1 令 $\mathcal{P}=\emptyset$; // 初始化
2 **repeat**
3　　使用某种多样性度量方法来计算训练集上各样本点的多样性，选择其中多样性最大的一点记为 $(\boldsymbol{x},y)$;
4　　将 $\mathcal{H}$ 中能够正确分类该样本的基分类器按性能递增顺序排列;
5　　选择上一步中最好的基分类器 $h(\cdot)$, 将其从 $\mathcal{H}$ 中移入 $\mathcal{P}$;
6 **until** 终止条件满足;

算法 3.2: 基于准确度和多样性之间平衡的集成剪枝框架 (*FTAD*)

$\mathrm{div}(h_{ens},\boldsymbol{x})$ 之积的差值。随后将被选择出的基分类器从原始集成器集合 $\mathcal{H}$ 移入子集成器集合 $\mathcal{P}$ 内，如第 5 行所示。如此循环，直至终止条件满足。

由 *EPBD* 算法可知，我们希望寻找一些多样性和准确度都较高的基分类器来加入剪枝后的子集成器。而 *EPBD* 使用的是本文中所提出的多样性度量，若将其替换成其他不同的多样性度量方法，我们可以提出一种 "基于准确度和多样性之间平衡的集成剪枝框架 *(Ensemble Pruning Framework Utilizing the Trade-off Between Accuracy and Diversity, FTAD)*"，如算法 3.2 所示。注意在算法 3.2 中第 3 行中，所使用的多样性度量方法可以是现存的任意一种多样性度量方法。

3.3.4　*EPBD* 算法的复杂度分析

本节将给出 *EPBD* 算法的复杂度分析。由算法 3.1 可知：

- 首先，当 $i\in[k]$ 时，第 3 行的复杂度为 $\mathcal{O}(jm+m\log m)$, 其中 $j=n-i+1$, 且 n 和 m 分别为基分类器和训练样本的数量。注意 k 被简记为子集成器的规模，即 $\mathcal{P}$ 的基数。

- 其次，当 $i \in [k]$ 时，第 4 行的复杂度为 $\mathcal{O}(j \log j)$, 其中 $j = n - i + 1$.
- 再次，第 5 行的复杂度为 $\mathcal{O}(1)$.

因此，*EPBD* 算法的计算复杂度为 $\mathcal{O}\big(\sum_{j=n-k+1}^{n}(jm + m\log m + j\log j)\big)$，即 $\mathcal{O}\big(km(n - \frac{k-1}{2}) + km\log m + \sum_{i=n-k+1}^{n}(i\log i)\big)$.

3.4　实验评估

在本节中，我们将构建实验来验证所提出的集成器的多样性与泛化性能之间的关系，并评估所提出的基于多样性的集成剪枝方法 (*EPBD* 和 *FTAD*) 的有效性。所使用的数据集包括一个包含 $12,500$ 张图片的图像数据集 (猫狗分类[①]) 和 28 个来自 UCI 仓库[②]的数据集 [107]。所有实验均使用了标准的 5 折交叉验证，即在每一迭代中，整个数据集将被分成两部分，其中 80% 的数据样本将作为训练集使用，而另外 20% 的数据样本将作为测试集来使用。用于构建集成器的方法包括自助法 [4] 和自适应增强法 [16-17] 两种；用于构建集成器的基分类器包括多种模型，如决策树、朴素贝叶斯分类器、k-近邻分类器、线性模型分类器、线性支持向量机、支持向量机和多层感知机等。为了验证本章所提出的集成器的多样性与泛化性能之间关系的正确性，本节验证了集成器泛化风险与其估计值之间的散点图和相关系数。此外，为了评估本章所提出的剪枝算法的有效性，本节所使用的对比算法包括大量现有的剪枝算法，即：

- 基于排序的剪枝方法，如提前终止 (Early Stopping, ES)、KL 散度剪枝 (KL-divergence Pruning, KL)、Kappa 剪枝 (Kappa Pruning, KP) [70]、定向排序剪枝 (Orientation Ordering Pruning, OO) [72]、多样性正则化集成剪枝 (Diversity Regularized Ensemble Pruning, DREP) [48]、基于准确度的集成剪枝 (Accuracy-based Ensemble Pruning, TSP.AP) [108] 和基于多样性的集成剪枝 (Diversity-based Ensemble Pruning, TSP.DP) [108]；
- 基于优化的剪枝方法，如单目标集成剪枝 (Single-objective Ensemble Pruning, SEP)、帕累托集成剪枝 (Pareto Ensemble Pruning, PEP) [49]、最大相关性最小冗余度的剪枝方法 (the Maximal Relevant Minimal Redundant method, MRMR)、最大相关性最大互补性的剪枝方法 (the Maximum Relevancy Maximum Complementary method, MRMC) [109] 和判别集成剪枝 (the Discriminative Ensemble Pruning, DiscEP) [110]。

不止于此，本节还改造了两种基于可组合核心集的多样性最大化方法，使之能够

[①] http://www.kaggle.com/c/dogs-vs-cats

[②] http://archive.ics.uci.edu/ml/datasets.html

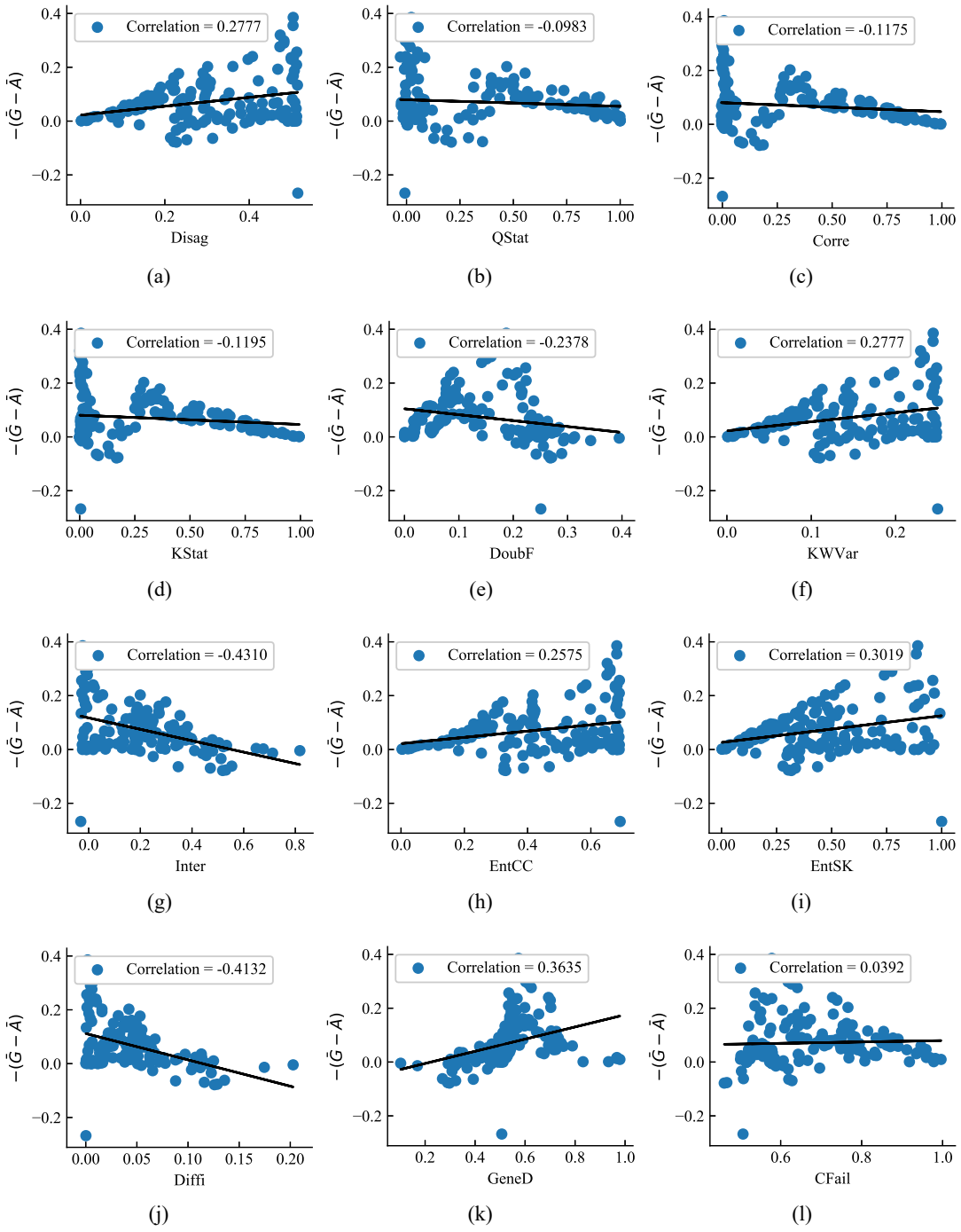


图 3.3　自助法所构造的集成器的损失差与不同多样性度量之间的关系

注：其中损失差由式 (3.8) 中的 $-(\bar{G}-\bar{A})$ 计算而得，而基分类器使用的是多层感知机。(a) 分歧度量 [57-58]；(b) Q 统计量 [55]；(c) 相关系数度量 [59]；(d) κ 统计量 [56]；(e) 双重失误度量 [60]；(f) Kohavi-Wolpert 差异度量 [62]；(g) 测试者彼此之间的一致性度量 [61, 79]；(h–i) 投票的熵度量 [63-64]；(j) 难度指数度量 [3, 65]；(k) 广义多样性 [66]；(l) 一致失误多样性 [66]。

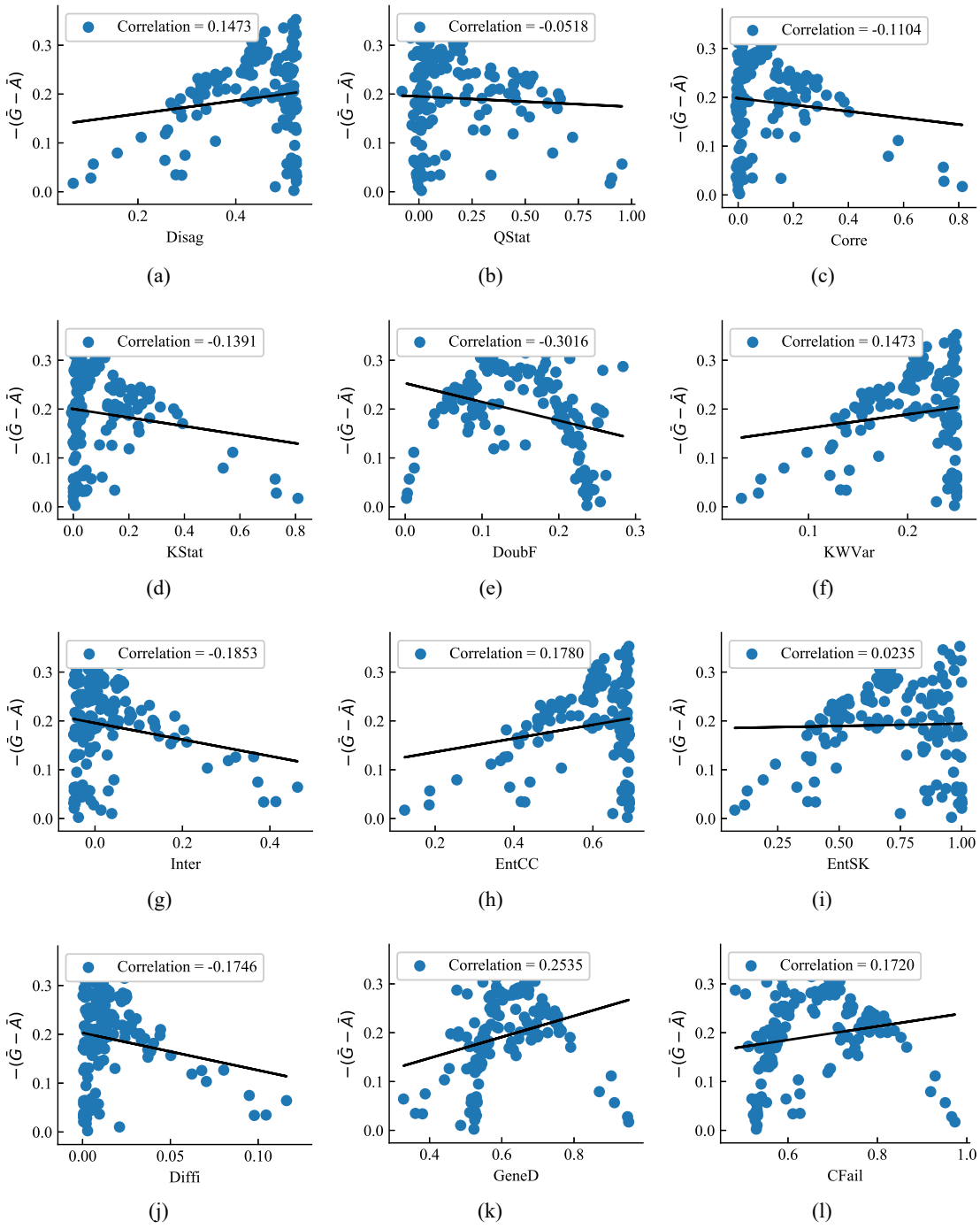
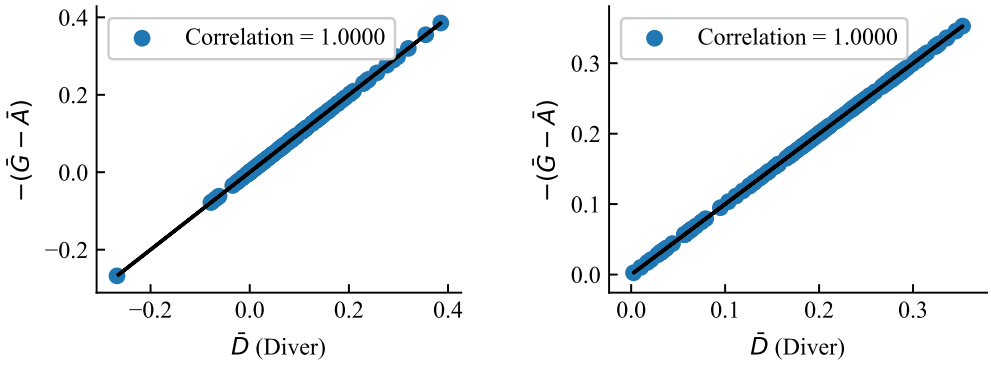


图 3.4　自适应增强法所构造的集成器的损失差与不同多样性度量之间的关系

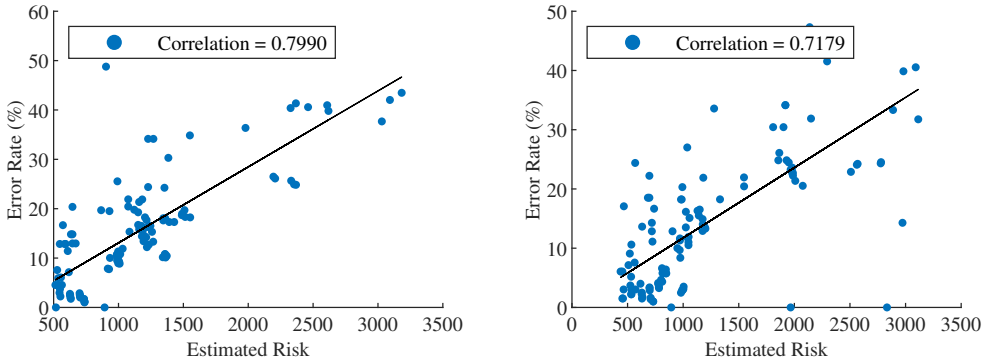
注：其中损失差由式 (3.8) 中的 $-(\bar{G}-\bar{A})$ 计算而得，而基分类器使用的是多层感知机。(a) 分歧度量 [57-58]；(b) Q 统计量 [55]；(c) 相关系数度量 [59]；(d) κ 统计量 [56]；(e) 双重失误度量 [60]；(f) Kohavi-Wolpert 差异度量 [62]；(g) 测试者彼此之间的一致性度量 [61, 79]；(h–i) 投票的熵度量 [63-64]；(j) 难度指数度量 [3, 65]；(k) 广义多样性 [66]；(l) 一致失误多样性 [66]。



(a)　(b)

图 3.5　集成器的损失差与本章所提出的多样性度量之间的关系

注：其中损失差和多样性分别指的是式 (3.8) 中的 $-(\bar{G}-\bar{A})$ 和式 (3.9c)，使用多层感知机作为基分类器。(a–b) 分别使用自助法和自适应增强法来构造集成器。



(a)　(b)

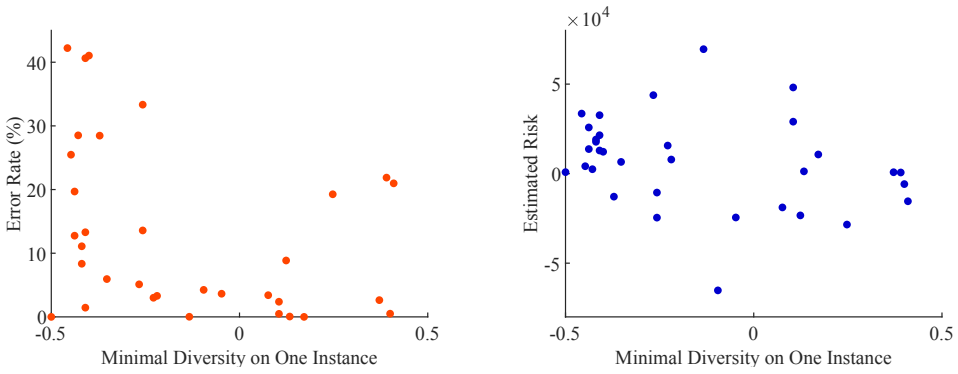
图 3.6　自助法所构造的集成器的泛化误差与其风险估计值之间的关系

注：其中风险估计值由式 (3.28) 计算而得。(a–b) 在二分类问题中分别使用朴素贝叶斯分类器和线性模型分类器作为基分类器。

执行集成剪枝任务，即冈萨雷斯算法 (the Gonzalez's Algorithm, GMA) 和局部搜索算法 (Local Search Algorithm, LCS) [111-112]。一个集成器将在训练集上训练得到，并在训练集上进行剪枝，随后在测试集上验证剪枝后子集成器的性能。值得一提的是，一些剪枝方法不能确定剪枝后子集成器的规模大小，如 OO、DREP、SEP、PEP 和 LCS；但是其他方法可以提前通过一个剪枝率参数来固定剪枝后子集成器的大小，其中剪枝率代表的是原始集成器中将被丢弃的基分类器所占的比例。无法确定子集成器规模的剪枝方法有可能会导致更大或者更小的子集成器，这将影响它们的空间代价。

3.4.1　验证分类集成器的误差分解

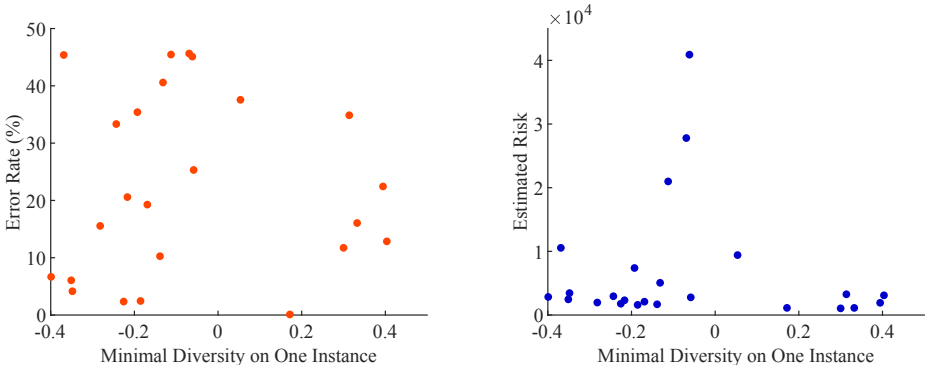
本小节所构造的实验用于验证所提出的定理 3.1，即分类集成器的误差分解。为了验证式 (3.9) 中所提出的多样性度量的有效性，本节比较了所提出的多



(a)　(b)

图 3.7　自助法所构造的集成器的多样性与其泛化性能之间的关系

注：参见定理 3.4，在二分类问题中使用决策树作为基分类器。(a) 集成器的多样性与错误率的关系；(b) 集成器的多样性与风险估计值的关系。



(a)　(b)

图 3.8　自适应增强法所构造的集成器的多样性与泛化性能之间的关系

注：参见定理 3.4，在二分类问题中使用线性模型分类器作为基分类器。(a) 集成器的多样性与错误率的关系；(b) 集成器的多样性与风险估计值的关系。

样性度量与文献中现有的多样性度量方法，包括分歧度量 (Disagreement measure, Disag) [57-58]、Q 统计量 (Q-statistic, QStat) [55]、相关系数度量 (Correlation coefficient, Corre) [59]、κ 统计量 (Kappa statistic, KStat) [56]、双重失误度量 (Double-fault measure, DoubF) [60]、Kohavi-Wolpert 差异度量 (Kohavi-Wolpert variance, KWVar) [62]、测试者彼此之间的一致性度量 (Interrater agreement, Inter) [61, 79]、投票的熵度量 (C&C's) (Entropy (C&C's), EntCC) [63]、投票的熵度量 (S&K's) (Entropy (S&K's), EntSK) [64]、难度指数度量 (Measure of difficulty, Diffi) [3, 65]、广义多样性 (Generalized diversity, GeneD) [66] 和一致失误多样性 (Coincident failure diversity, CFail) [66]。注意在这 12 个基准多样性度量方法中，前七个属于成对多样性度量，而后五个属于非成对多样性度量；而本章所提出的多样性度量无法直接归入成对或非成对的多样性度量，因其计算的是每个基分类器与整体集成器之间的差异。正如图 3.5(a) 和图 3.3 中所示，损失差 $-(\bar{G}-\bar{A})$ 与式 (3.9c) 中所

表 3.2　在自助法所构造的集成器上对比 *EPBD* 与基准剪枝算法

注：第二列 “Ensem” 代表未经剪枝的原始集成器的准确度。对于每一个数据集 (即每一行) 来说，更优的准确度和更低的标准差将用粗体标出。

Dataset	Ensem	ES	KL	KP	OO	DREP	SEP	PEP	MRMR	MRMC	DiscEP	TSP.AP	TSP.DP	GMA	LCS	*EPBD*
waveform	89.09±1.27	84.34±8.76	83.48±8.39	83.48±8.39	89.47±0.40	89.35±0.66	89.03±0.83	89.19±0.81	88.43±0.94‡	89.43±0.82	89.37±0.38	**89.71±0.75**	88.19±0.95‡	88.51±1.29	88.95±1.37	89.53±0.48
heart	55.56±0.00	55.56±0.00	55.56±0.00	55.56±0.00	55.56±0.00	55.56±0.00	55.56±0.00	**55.93±0.74**	**55.93±0.74**	**55.93±0.74**	**55.93±0.74**	55.56±0.00	55.56±0.00	55.56±0.00	55.56±0.00	**55.93±0.74**
page	91.26±0.22‡	91.28±0.19	91.26±0.24	91.33±0.29	91.37±0.29	91.35±0.20	91.24±0.11	91.28±0.27	91.33±0.20	91.19±0.20‡	91.24±0.16	91.26±0.19	**91.39±0.21**	91.12±0.16‡	91.12±0.16‡	91.39±0.26
sensor (2d)	94.12±0.80‡	94.08±0.95‡	94.15±0.98	93.97±0.93‡	94.21±1.08	94.37±1.01	94.02±0.92‡	**94.41±0.83**	94.24±0.55	94.21±1.01	93.73±0.95‡	94.10±1.09	94.08±0.80	94.02±1.04‡	94.10±1.02	94.37±0.85
sensor (24d)	89.37±0.61	89.13±0.27	88.95±0.73	88.91±0.75	**89.39±1.38**	88.82±1.05‡	88.82±1.11‡	89.31±1.25	89.15±0.97	89.09±0.92	88.87±0.86	88.93±0.93‡	88.95±1.02	89.07±0.93	88.74±1.16	89.33±0.95
EEGEyeState	55.13±0.01	**55.13±0.00**	**55.13±0.00**	**55.13±0.00**	55.13±0.01	**55.13±0.00**	**55.13±0.00**	**55.13±0.00**	55.13±0.01	55.13±0.01	55.13±0.01	**55.13±0.00**	55.13±0.01	**55.13±0.00**	**55.13±0.00**	55.13±0.01
gmm (10d)	99.70±0.60	99.70±0.60	99.70±0.60	99.70±0.60	**99.80±0.40**	99.70±0.60	99.70±0.60	**99.80±0.40**	**99.80±0.40**	**99.80±0.40**	**99.80±0.40**	99.70±0.60	99.60±0.59	99.70±0.60	99.60±0.59	**99.80±0.40**
ames	61.17±4.01	58.54±7.61	55.77±6.67‡	56.50±4.72	59.71±6.93	57.96±5.57‡	58.54±4.53	59.85±5.28	59.12±4.21	57.66±3.72	**61.61±4.49**	60.44±4.78	59.71±6.04	56.50±2.64	59.56±3.53	61.02±5.80
wisconsin	96.89±0.55	96.44±0.55	96.44±0.86	96.89±0.73	96.59±1.00	96.59±0.76	96.00±0.59	96.44±0.98	96.74±0.76	96.89±1.09	97.04±0.47	96.74±1.00	96.89±0.55	96.59±0.36	**96.89±0.30**	96.89±0.73
ecoli	95.76±2.23	95.76±2.42	95.76±2.42	95.76±2.42	95.76±2.42	95.76±2.42	95.76±2.42	**96.36±2.06**	**96.36±2.06**	95.76±2.23	95.76±2.23	95.76±2.23	95.76±2.42	96.06±2.06	96.06±2.27	**96.36±2.06**
liver	62.90±2.98‡	61.16±2.49	62.03±5.75	65.22±2.43	63.77±3.04	65.51±3.60	64.64±3.38	66.38±2.96	65.80±2.98	63.19±2.98	62.90±2.84	61.74±2.98	64.93±3.82	61.45±6.12	61.74±4.36‡	**66.96±3.36**
yeast	69.59±4.61	67.77±2.86‡	67.70±2.36‡	69.39±1.66	69.73±3.27	65.41±2.60‡	69.80±4.23	69.39±3.44	69.86±3.06	69.12±2.89	68.24±3.82‡	68.58±3.15	68.72±3.44	67.36±3.66‡	68.45±3.95	**70.54±2.92**
sonar	80.00±6.62	76.59±5.69	78.05±6.90	79.02±6.28	80.49±6.54	80.00±5.85	79.51±3.65	79.51±5.89	**81.95±8.10**	80.49±6.36	76.10±4.97	79.02±3.31	76.10±6.79	79.51±7.49	80.49±6.17	80.49±7.07
wilt	93.90±1.34	93.71±1.49	94.37±0.85	93.34±1.51	94.33±0.87	92.12±1.81‡	93.90±1.44	93.59±1.59	94.33±0.98	94.35±0.93	94.00±1.25	94.46±0.95	93.86±1.00	93.92±1.44	93.75±1.17	**94.48±0.87**
spam	78.67±3.44	78.54±2.74	77.34±2.18	73.32±5.74	74.56±6.96	79.09±2.84	77.87±1.39	79.61±2.67	79.04±2.80	72.82±8.46	78.91±3.13	79.76±2.42	77.87±3.55	73.93±5.00	77.39±1.58	**79.80±2.41**
landsat	96.25±0.63	96.31±0.63	95.57±0.98	95.30±1.02	96.27±0.58	95.68±0.84	96.31±0.54	95.96±0.91	96.22±0.56	95.65±1.16	96.22±0.61	**96.33±0.50**	95.69±0.90	96.13±0.75	96.07±0.58	96.27±0.65
t-test (W/T/L)	3/13/0	2/14/0	2/14/0	1/15/0	0/16/0	4/12/0	2/14/0	0/16/0	1/15/0	1/15/0	2/14/0	1/15/0	1/15/0	4/12/0	2/14/0	—
Average Rank	7.47	10.84	11.47	11.03	6.00	9.41	9.78	6.56	5.13	7.88	8.28	8.22	9.28	11.44	10.69	2.53

[1] 表中所汇报的实验结果均是每种剪枝方法使用标准 5 折交叉验证的实验在各个数据集的测试集上的平均准确度 (%) 与其相应的标准差。

[2] 通过在 5% 显著水平下的双尾成对 t 检验对比各算法之间是否有明显差异，其中 ‡ 和 † 分别表示 *EPBD* 显著优于或者是显著差于对比算法的性能；若无符号标记，则表示 *EPBD* 与相应对比算法之间没有显著差异。

[3] 表中最后两行分别展示了检验结果和各算法的平均序值。各算法的平均序值由 Friedman 检验 [113] 计算而得，而 “W/T/L” 则表示 *EPBD* 显著优于、并无显著不同或者显著差于相应的对比算法的次数。

表 3.3　在自适应增强法所构造的集成器上对比 *EPBD* 与基准剪枝算法

注：第二列 “Ensem” 代表未经剪枝的原始集成器的准确度。对于每一个数据集 (即每一行) 来说，更优的准确度和更低的标准差将用粗体标出。

Dataset	Ensem	ES	KL	KP	OO	DREP	SEP	PEP	MRMR	MRMC	DiscEP	TSP.AP	TSP.DP	GMA	LCS	*EPBD*
spam	79.33±2.04	81.44±1.58	79.39±3.15	78.32±2.76	79.17±2.23	79.91±1.44	79.06±2.90	**81.85±1.36**	81.44±1.58	78.78±2.24	78.78±2.24	79.04±2.11	76.63±2.51	73.54±6.13	79.04±2.11	79.98±2.12
credit	77.96±0.06	77.93±0.06	77.93±0.13	77.91±0.12	77.99±0.08	77.99±0.09	77.93±0.09	77.95±0.08	77.98±0.14	77.98±0.09	77.96±0.14	77.91±0.13	77.88±0.11‡	77.86±0.10	77.95±0.06	**78.03±0.10**
page	90.40±0.12‡	90.35±0.26‡	90.40±0.27‡	90.29±0.33‡	90.93±0.32	**90.99±0.24**	90.49±0.32	90.88±0.35	90.60±0.25	90.93±0.32	90.91±0.31	**90.99±0.24**	90.22±0.42‡	90.80±0.32	90.77±0.38	**90.99±0.24**
shuttle	96.75±0.08‡	97.33±0.06‡	97.33±0.06‡	92.40±2.09‡	97.81±0.10	97.70±0.05‡	94.75±2.43	96.80±2.10	**97.85±0.05**	97.65±0.10‡	**97.85±0.05**	**97.85±0.05**	92.05±2.09‡	97.59±0.12‡	**97.85±0.05**	**97.85±0.05**
wilt	94.64±0.04	94.66±0.08	94.66±0.08	94.66±0.08	94.66±0.08	94.66±0.08	94.62±0.00	94.62±0.00	**94.71±0.10**	94.66±0.08	**94.71±0.10**	**94.71±0.10**	**94.71±0.10**	94.66±0.08	**94.71±0.10**	**94.71±0.10**
segmentation	**67.40±0.56**	65.93±0.67‡	65.19±1.24‡	63.94±0.95‡	67.14±1.14	68.14±1.07†	66.02±1.79	66.32±3.21	65.89±0.87‡	65.71±0.95‡	66.67±0.43	66.28±0.73	65.50±1.66	66.15±0.67	66.54±0.45	66.80±0.56
iono	81.43±1.56	77.71±5.16	78.00±7.15	80.57±5.16	**84.29±3.61**	83.71±3.68	80.57±6.43	81.43±6.32	82.57±3.43	82.00±5.90	82.57±2.77	82.57±2.77	76.00±6.29	72.86±4.52‡	80.57±5.16	84.00±2.91
wilt	97.17±0.24	96.38±0.52	97.35±0.38	95.37±1.25‡	97.29±0.65	97.35±0.38	97.13±0.56	97.31±0.44	97.27±0.29	97.25±0.51	**97.37±0.40**	97.35±0.38	96.34±1.28	97.35±0.38	97.13±0.48	97.35±0.38
ecoli	**69.70±2.71**	64.24±4.35	62.73±4.94	63.33±3.76	66.36±6.24	64.55±3.26	63.94±3.76	63.03±4.75	64.55±3.12	61.82±4.64	60.91±8.10	66.06±6.82	65.15±1.66	64.55±4.94	62.73±4.66	67.27±4.85
yeast	63.45±1.90	**66.15±2.35**	63.24±1.94	63.31±1.04	63.78±2.80	60.61±2.65	61.96±1.46	63.18±2.73	62.84±3.64	62.64±3.29	62.50±2.87	63.18±2.79	59.80±3.74	62.09±2.85	62.50±2.45	63.78±3.60
ringnorm	68.32±1.27‡	68.67±1.21	68.21±1.69	67.88±0.68‡	69.20±0.90	**69.86±1.31**	67.05±1.40‡	68.55±1.22	69.17±0.82	68.53±1.03	68.88±1.10	69.32±1.04	68.15±1.22‡	68.80±0.86	68.86±0.77	69.17±0.92
waveform	83.46±0.60	83.00±0.64	82.52±0.62	81.72±1.11‡	82.78±1.21	83.26±0.78	82.82±1.03‡	82.86±0.74	83.44±0.19	83.14±1.14	82.40±0.94‡	82.90±0.30‡	82.80±1.05	82.32±0.67‡	82.54±0.47‡	**83.86±0.74**
gmm (10d)	99.70±0.25	**99.80±0.25**	95.58±5.24	97.59±4.58	99.70±0.25	99.60±0.38	99.70±0.60	**99.80±0.25**	97.39±4.48	99.60±0.38	97.49±4.53	99.50±0.32	99.50±0.32	99.70±0.25	99.60±0.38	**99.80±0.25**
t-test (W/T/L)	3/10/0	3/10/0	3/10/0	6/7/0	0/13/0	1/11/1	2/11/0	0/13/0	1/12/0	2/11/0	1/12/0	1/12/0	4/9/0	3/10/0	1/12/0	—
Average Rank	7.62	8.85	10.96	13.15	5.35	5.42	11.19	8.08	6.58	9.23	8.04	6.35	12.77	10.58	9.15	2.69

[1] 表中所汇报的实验结果均是每种剪枝方法使用标准 5 折交叉验证的实验在各个数据集的测试集上的平均准确度 (%) 与其相应的标准差。

[2] 通过在 5% 显著水平下的双尾成对 t 检验对比各算法之间是否有明显差异，其中 ‡ 和 † 分别表示 *EPBD* 显著优于或者是显著差于对比算法的性能；若无符号标记，则表示 *EPBD* 与相应对比算法之间没有显著差异。

[3] 表中最后两行分别展示了检验结果和各算法的平均序值。各算法的平均序值由 Friedman 检验 [113] 计算而得，而 “W/T/L” 则表示 *EPBD* 显著优于、并无显著不同或者显著差于相应的对比算法的次数。

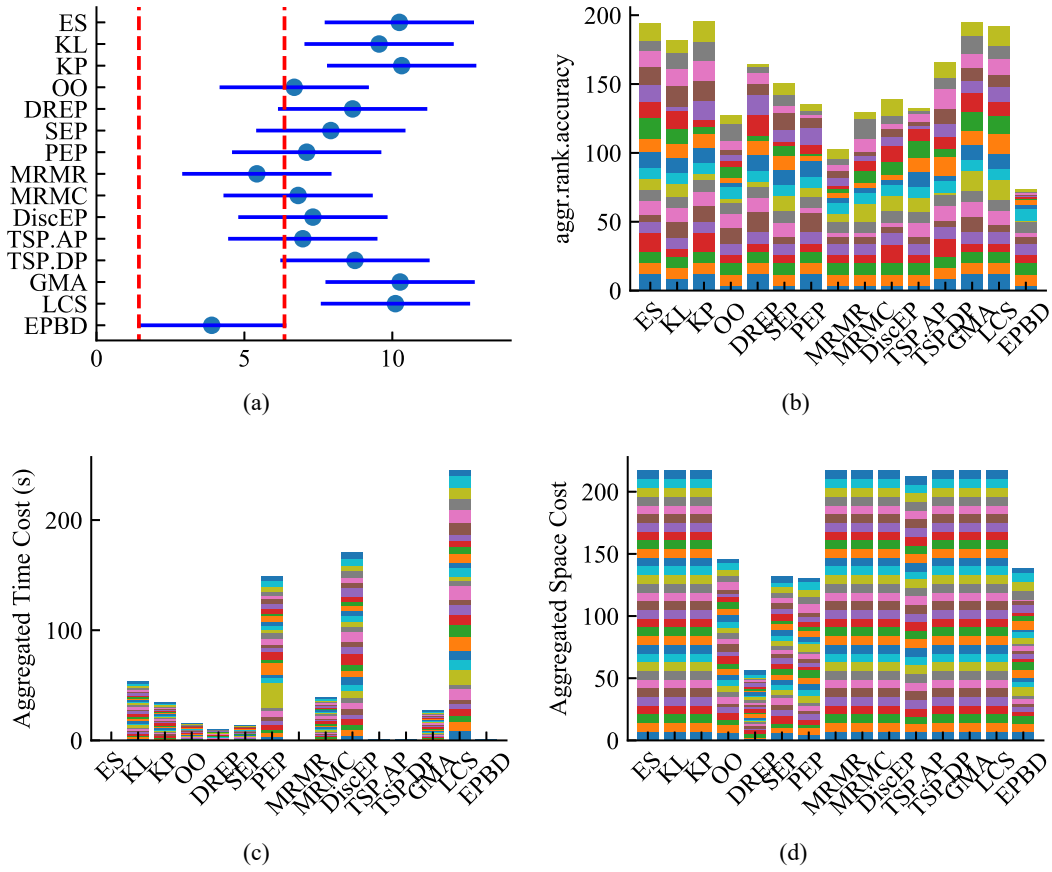


图 3.9　在自助法所构造的同质集成器上对比 *EPBD* 和基准算法

注：即对比 *EPBD* 与其他基准剪枝算法的测试准确度。(a) Friedman 检验图 (两种方法间若没有重叠则代表着有显著差异) [113]。(b) 每种剪枝方法根据测试准确度排序的总计序值图 (序值越小，则该方法越优) [49]。(c) 总计时间代价。(d) 总计空间代价。

提出的多样性度量 $\bar{D}$ 的相关系数[①]高于损失差值与其他多样性度量值之间的相关系数。从图 3.5(b) 和图 3.4 中也可得出相似的结论。因此，我们可以认为所提出的分类集成器的误差分解是合理的。

3.4.2　验证集成器中多样性与其泛化性能之间的关系

本小节所构造的实验用于验证所提出的定理 3.4，即分类问题中集成器的多样性与其泛化性能的关系。首先，我们需要验证集成器的泛化误差风险与风险误差估计值之间的关系，这部分的实验结果可见图 3.6，其中风险估计值是根据式 (3.28) 计算所得；且风险估计值与集成器的真实错误率之间的相关系数也经计算后标注在图 3.6(a)–3.6(b) 中。此外，图 3.7–3.8 中的实验结果也展示了集成器中

[①]本章所使用的相关系数 [114] 指的是皮尔逊积矩相关系数，度量的是两个随机变量 x 和 Y 之间的线性相关程度，取值范围为 $[-1, 1]$. 若其值为 1，则表示两者之间完全正相关，0 表示不相关，而 -1 则表示完全负相关 [115]。

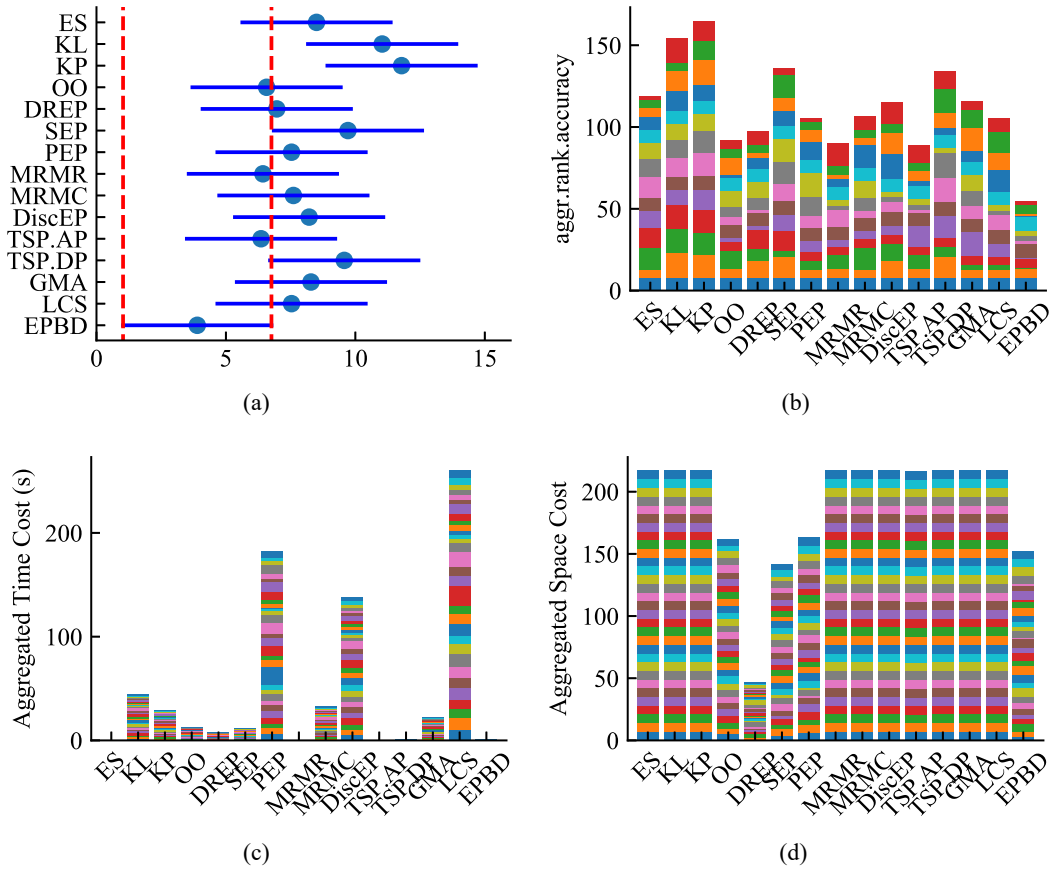


图 3.10　在自适应增强法所构造的同质集成器上对比 *EPBD* 和基准算法

注：即对比 *EPBD* 与其他基准剪枝算法的测试准确度。(a) Friedman 检验图 (两种方法间若没有重叠则代表着有显著差异) [113]。(b) 每种剪枝方法根据测试准确度排序的总计序值图 (序值越小，则该方法越优) [49]。(c) 总计时间代价。(d) 总计空间代价。

的多样性与泛化性能的关系 (如定理 3.4 所示) 是可信的。从图 3.7(a) 和图 3.8(a) 中可以看出，随着多样性的提高，集成器的泛化性能大致呈现出一种 “上升-下降-上升-下降” 的趋势，这也与第 3.3.2 节中的分析一致。因此，可以利用所提出的集成器的多样性与其泛化性能之间的关系来指导我们对多样性的理解，这也是探索多样性在集成学习中所起作用的一个良好开端。

3.4.3　对比 *EPBD* 与当前集成剪枝的基准算法

本小节对比了 *EPBD* 与现有集成剪枝算法之间的性能，这些基准算法包括 ES、KL、KP、OO、DREP、SEP、PEP、MRMR、MRMC、DiscEP、TSP.AP、TSP.DP、GMA 和 LCS。实验结果汇总在表 3.2–3.3 中，包括每种剪枝算法使用标准 5 折交叉验证法在每个数据集上的平均测试准确率及其相应的标准差。以表 3.2 为例，其中每一行 (即每一个数据集) 对比的都是使用同样基分类器和原始集成器的不同剪枝算法的性能，并用粗体字表示更高的准确度和更低的标准差。此外，我

表 3.4　在自助法所构造的集成器上对比 *EPBD* 与 *FTAD*

注：其中 *FTAD* 使用了不同的多样性度量方法，而基分类器使用的是朴素贝叶斯分类器。第二列 “Ensem” 代表未经剪枝的原始集成器的准确度。对于每一个数据集 (即每一行) 来说，更优的准确度和更低的标准差将用粗体标出。

Dataset	Ensem	Disag	QStat	Corre	KStat	DoubF	KWVar	Inter	EntCC	EntSK	Diffi	GeneD	CFail	*EPBD*
ames	57.37±3.53‡	**59.71±4.39**	59.12±4.36	59.12±4.36	59.12±4.36	58.69±4.07	59.12±4.36	59.12±4.36	58.39±4.08	58.39±4.08	59.12±4.36	58.69±4.07	58.69±4.07	58.69±4.07
card	**79.56±4.48**	79.42±5.57	79.12±5.03	79.12±5.03	79.12±5.03	79.42±5.57	79.42±5.57	79.42±5.57	79.56±5.44	79.56±5.44	79.12±5.03	79.42±5.57	79.42±4.85	79.42±5.57
heart	84.07±3.23	**84.44±1.89**	84.07±2.22	84.07±2.22	84.07±2.22	83.70±2.72	84.07±3.01	84.07±3.01	83.70±2.72	83.70±2.72	84.07±2.22	83.70±2.72	83.70±2.72	83.70±2.72
ringnorm	**98.69±0.24**	98.66±0.25	98.66±0.25	98.66±0.25	98.66±0.25	98.66±0.25	98.66±0.25	98.66±0.25	98.66±0.25	98.66±0.25	98.66±0.25	98.66±0.25	98.66±0.25	98.66±0.25
wisconsin	**96.00±1.11**	96.00±1.37	96.00±1.37	96.00±1.37	96.00±1.37	96.00±1.37	96.00±1.37	96.00±1.37	96.00±1.37	96.00±1.37	96.00±1.37	96.00±1.37	96.00±1.37	96.00±1.37
landsat	90.30±0.73	**90.48±0.73**	**90.48±0.73**	**90.48±0.73**	**90.48±0.73**	**90.48±0.73**	**90.48±0.73**	**90.48±0.73**	90.47±0.71	**90.48±0.73**	**90.48±0.73**	**90.48±0.73**	90.47±0.71	**90.48±0.73**
shuttle	89.48±0.16‡	**89.71±0.14**	**89.71±0.14**	**89.71±0.14**	**89.71±0.14**	**89.71±0.14**	**89.71±0.14**	**89.71±0.14**	**89.71±0.14**	**89.71±0.14**	**89.71±0.14**	**89.71±0.14**	**89.71±0.14**	**89.71±0.14**
ecoli	92.12±2.78	**92.42±3.46**	**92.42±3.46**	**92.42±3.46**	**92.42±3.46**	94.24±2.61	**92.42±3.46**	**92.42±3.46**	**92.42±3.46**	**92.42±3.46**	**92.42±3.46**	**92.42±3.46**	**92.42±3.46**	94.24±2.61
yeast	58.24±2.64‡	60.20±1.67	57.30±3.33	57.30±3.33	57.30±3.33	**61.82±1.46**	**61.82±1.46**	**61.82±1.46**	60.27±1.64	60.20±1.67	57.30±3.33	61.69±1.70	61.49±1.61	**61.82±1.46**
mammo_graphic	**80.73±3.04**	79.79±2.76	79.69±2.57	79.69±2.57	79.69±2.57	79.58±2.72	79.48±2.61	79.48±2.61	79.79±2.76	79.79±2.76	79.69±2.57	79.58±2.72	79.58±2.72	79.58±2.72
madelon	58.85±2.37	58.96±2.28	59.19±1.31	59.19±1.31	59.19±1.31	59.04±2.07	59.04±2.07	59.04±2.07	**59.58±2.39**	58.96±2.28	59.19±1.31	59.04±2.07	58.92±2.22	59.04±2.07
sensor (2d)	81.41±0.86	81.61±0.91	81.81±0.57	81.81±0.57	81.81±0.57	**82.02±0.70**	81.81±0.57	81.81±0.57	81.59±0.93	81.61±0.91	81.81±0.57	**82.02±0.70**	81.92±0.76	**82.02±0.70**
sensor (4d)	74.57±0.88‡	75.44±0.99	**75.55±0.89**	**75.55±0.89**	**75.55±0.89**	**75.55±0.89**	**75.55±0.89**	**75.55±0.89**	75.33±0.85	75.44±0.99	**75.55±0.89**	**75.55±0.89**	75.51±0.94	**75.55±0.89**
waveform	85.29±0.95	85.35±0.94	85.33±0.92	85.33±0.92	85.33±0.92	85.33±0.92	85.33±0.92	85.33±0.92	**85.35±0.91**	85.35±0.94	85.33±0.92	85.33±0.92	85.33±0.92	85.33±0.92
t-test (W/T/L)	4/10/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	—
Average Rank	9.89	6.71	7.21	7.21	7.21	6.79	7.21	7.21	7.68	7.86	7.21	7.43	8.57	6.79

[1] 表中所汇报的实验结果均是每种剪枝方法使用标准 5 折交叉验证的实验在各个数据集的测试集上的平均准确度 (%) 与其相应的标准差。

[2] 通过在 5% 显著水平下的双尾成对 t 检验对比各算法之间是否有明显差异，其中 ‡ 和 † 分别表示 *EPBD* 显著优于或者是显著差于对比算法的性能；若无符号标记，则表示 *EPBD* 与相应对比算法之间没有显著差异。

[3] 表中最后两行分别展示了检验结果和各算法的平均序值。各算法的平均序值由 Friedman 检验 [113] 计算而得，而 “W/T/L” 则表示 *EPBD* 显著优于、并无显著不同或者显著差于相应的对比算法的次数。

表 3.5　在自助法所构造的集成器上对比 *EPBD* 与 *FTAD*

注：其中 *FTAD* 使用了不同的多样性度量方法，而基分类器使用的是 k-近邻分类器。第二列 “Ensem” 代表未经剪枝的原始集成器的准确度。对于每一个数据集 (即每一行) 来说，更优的准确度和更低的标准差将用粗体标出。

Dataset	Ensem	Disag	QStat	Corre	KStat	DoubF	KWVar	Inter	EntCC	EntSK	Diffi	GeneD	CFail	*EPBD*
gmm (2d)	96.90±0.63	96.92±0.66	**96.97±0.66**	**96.97±0.66**	**96.97±0.66**	**96.97±0.66**	**96.97±0.66**	**96.97±0.66**	96.92±0.66	96.92±0.66	**96.97±0.66**	**96.97±0.66**	**96.97±0.66**	**96.97±0.66**
card	67.74±2.33	68.03±2.67	67.74±3.28	67.74±3.28	67.74±3.28	**68.91±2.83**	**68.91±2.83**	**68.91±2.83**	67.88±2.85	68.03±2.67	67.74±3.28	68.32±2.15	**68.91±2.83**	**68.91±2.83**
liver	64.06±4.62	62.61±5.45	63.48±5.53	63.48±5.53	63.48±5.53	**64.35±4.36**	63.19±6.77	63.19±6.77	62.61±5.45	62.61±5.45	63.48±5.53	62.61±6.94	63.48±5.53	**64.35±4.36**
credit	74.18±0.35‡	74.69±0.44	74.63±0.45	74.63±0.45	74.63±0.45	**74.72±0.35**	**74.72±0.35**	**74.72±0.35**	74.70±0.44	74.69±0.44	74.63±0.45	**74.72±0.35**	74.62±0.31	**74.72±0.35**
page	95.96±0.35	96.11±0.34	**96.11±0.29**	**96.11±0.29**	**96.11±0.29**	**96.11±0.29**	**96.11±0.29**	**96.11±0.29**	96.11±0.34	96.11±0.34	**96.11±0.29**	**96.11±0.29**	**96.11±0.29**	**96.11±0.29**
shuttle	**99.84±0.03**	99.83±0.02	99.83±0.03	99.83±0.03	99.83±0.03	99.83±0.03	99.83±0.03	99.83±0.03	99.83±0.02	99.83±0.02	99.83±0.03	99.83±0.03	99.83±0.03	99.83±0.03
wilt	98.01±0.50	97.97±0.54	97.81±0.34	97.81±0.34	97.81±0.34	**98.04±0.38**	97.75±0.34	97.75±0.34	97.97±0.54	97.97±0.54	97.81±0.34	97.91±0.48	97.95±0.47	**98.04±0.38**
ecoli	**96.36±1.55**	95.76±1.13	95.76±1.13	95.76±1.13	95.76±1.13	95.76±1.13	95.76±1.13	95.76±1.13	95.76±1.13	95.76±1.13	95.76±1.13	95.76±1.13	95.76±1.13	95.76±1.13
mammo_graphic	77.71±2.70	78.02±2.76	77.92±2.90	77.92±2.90	77.92±2.90	77.92±2.90	77.92±2.90	77.92±2.90	**78.12±2.81**	78.02±2.76	77.92±2.90	77.92±2.90	77.71±3.06	77.92±2.90
madelon	70.38±2.21	**70.46±1.96**	69.69±2.05	69.69±2.05	69.69±2.05	70.31±1.89	69.92±1.58	69.92±1.58	70.15±1.96	**70.46±1.96**	69.69±2.05	70.31±1.89	70.35±1.94	70.31±1.89
sensor (4d)	**96.99±0.50**	96.83±0.45	96.98±0.43	96.98±0.43	96.98±0.43	96.98±0.43	96.98±0.43	96.98±0.43	96.87±0.43	96.83±0.45	96.98±0.43	96.98±0.43	96.98±0.43	96.98±0.43
sensor (24d)	**88.82±0.78**	88.78±0.74	88.43±0.66	88.43±0.66	88.43±0.66	88.56±0.57	88.56±0.57	88.56±0.57	88.74±0.75	88.78±0.74	88.43±0.66	88.56±0.57	88.56±0.57	88.56±0.57
waveform	85.03±0.83	85.09±0.60	85.07±0.78	85.07±0.78	85.07±0.78	**85.19±0.53**	85.15±0.57	85.15±0.57	85.05±0.63	85.09±0.60	85.07±0.78	85.07±0.32	85.17±0.26	**85.19±0.53**
EEGEyeState	**96.72±0.20**†	96.37±0.21	96.24±0.24	96.24±0.24	96.24±0.24	96.38±0.15	96.38±0.15	96.38±0.15	96.37±0.21	96.37±0.21	96.24±0.24	96.38±0.15	96.38±0.15	96.38±0.15
t-test (W/T/L)	1/12/1	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	0/14/0	—
Average Rank	6.64	7.68	9.18	9.18	9.18	5.00	6.89	6.89	8.50	7.68	9.18	7.18	6.82	5.00

[1] 表中所汇报的实验结果均是每种剪枝方法使用标准 5 折交叉验证的实验在各个数据集的测试集上的平均准确度 (%) 与其相应的标准差。

[2] 通过在 5% 显著水平下的双尾成对 t 检验对比各算法之间是否有明显差异，其中 ‡ 和 † 分别表示 *EPBD* 显著优于或者是显著差于对比算法的性能；若无符号标记，则表示 *EPBD* 与相应对比算法之间没有显著差异。

[3] 表中最后两行分别展示了检验结果和各算法的平均序值。各算法的平均序值由 Friedman 检验 [113] 计算而得，而 “W/T/L” 则表示 *EPBD* 显著优于、并无显著不同或者显著差于相应的对比算法的次数。

表 3.6　在自助法所构造的集成器上对比 *EPBD* 与 *FTAD*

注：其中 *FTAD* 使用了不同的多样性度量方法，而基分类器使用的是支持向量机。第二列 “Ensem” 代表未经剪枝的原始集成器的准确度。对于每一个数据集 (即每一行) 来说，更优的准确度和更低的标准差将用粗体标出。

Dataset	Ensem	Disag	QStat	Corre	KStat	DoubF	KWVar	Inter	EntCC	EntSK	Diffi	GeneD	CFail	*EPBD*
gmm (10d)	**99.80±0.25**	99.80±0.40	99.80±0.40	99.80±0.40	99.80±0.40	99.80±0.40	99.80±0.40	99.80±0.40	99.80±0.40	99.80±0.40	99.80±0.40	99.80±0.40	99.80±0.40	99.80±0.40
sonar	62.93±3.90	64.39±4.25	64.39±7.96	64.39±7.96	64.39±7.96	62.44±4.25	65.85±7.40	65.85±7.40	61.46±4.47	61.95±5.69	64.39±7.96	65.85±7.40	65.85±7.40	**66.83±7.00**
waveform	88.21±0.50‡	89.97±0.52	89.81±0.48‡	89.81±0.48‡	89.81±0.48‡	89.51±0.77‡	89.51±0.77‡	89.51±0.77‡	89.97±0.52	89.97±0.52	89.81±0.48‡	89.51±0.77‡	89.51±0.77‡	**90.71±0.75**
landsat	**65.24±0.00**	**65.24±0.00**	**65.24±0.00**	**65.24±0.00**	**65.24±0.00**	**65.24±0.00**	**65.24±0.00**	**65.24±0.00**	**65.24±0.00**	**65.24±0.00**	**65.24±0.00**	**65.24±0.00**	**65.24±0.00**	**65.24±0.00**
page	90.59±0.21	90.99±0.51	90.95±0.48	90.95±0.48	90.95±0.48	90.95±0.48	90.95±0.48	90.95±0.48	90.86±0.49	90.86±0.49	90.95±0.48	90.95±0.48	90.97±0.44	**90.99±0.41**
shuttle	97.12±0.15	97.28±0.18	**97.28±0.16**	**97.28±0.16**	**97.28±0.16**	**97.28±0.16**	**97.28±0.16**	**97.28±0.16**	97.28±0.18	97.28±0.18	**97.28±0.16**	**97.28±0.16**	**97.28±0.16**	95.25±2.08
yeast	65.00±1.14	55.41±12.42	55.41±12.42	55.41±12.42	55.41±12.42	54.93±12.09	54.93±12.09	54.93±12.09	59.80±9.89	59.93±9.93	55.41±12.42	54.93±12.09	45.20±10.17‡	**65.07±1.18**
sensor (4d)	**85.63±0.90**	84.47±0.78	84.00±1.45	84.00±1.45	84.00±1.45	77.58±8.33	84.01±0.85	84.01±0.85	84.47±0.78	84.47±0.78	84.00±1.45	81.21±7.51	81.28±7.54	84.01±0.85
EEGEyeState	**55.13±0.00**	**55.13±0.00**	**55.13±0.00**	**55.13±0.00**	**55.13±0.00**	**55.13±0.00**	**55.13±0.00**	**55.13±0.00**	**55.13±0.00**	**55.13±0.00**	**55.13±0.00**	**55.13±0.00**	**55.13±0.00**	**55.13±0.00**
t-test (W/T/L)	1/8/0	0/9/0	1/8/0	1/8/0	1/8/0	1/8/0	1/8/0	1/8/0	0/9/0	0/9/0	1/8/0	1/8/0	2/7/0	—
Average Rank	9.33	5.39	7.56	7.56	7.56	9.56	7.72	7.72	6.72	6.50	7.56	8.50	8.17	5.17

[1] 表中所汇报的实验结果均是每种剪枝方法使用标准 5 折交叉验证的实验在各个数据集的测试集上的平均准确度 (%) 与其相应的标准差。

[2] 通过在 5% 显著水平下的双尾成对 t 检验对比各算法之间是否有明显差异，其中 ‡ 和 † 分别表示 *EPBD* 显著优于或者是显著差于对比算法的性能；若无符号标记，则表示 *EPBD* 与相应对比算法之间没有显著差异。

[3] 表中最后两行分别展示了检验结果和各算法的平均序值。各算法的平均序值由 Friedman 检验 [113] 计算而得，而 “W/T/L” 则表示 *EPBD* 显著优于、并无显著不同或者显著差于相应的对比算法的次数。

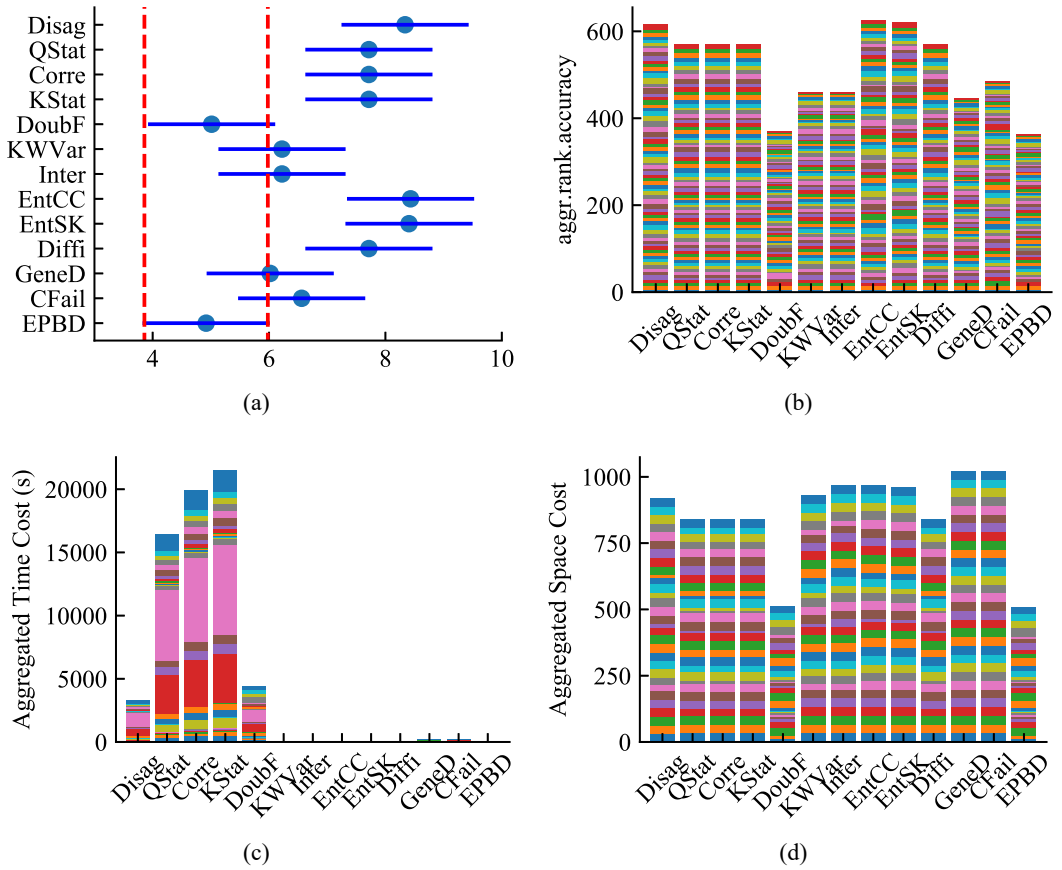


图 3.11　在自助法所构造的同质集成器上对比 *EPBD* 和 *FTAD*

注：即对比 *EPBD* 与使用其他多样性度量的 *FTAD* 的测试准确度。(a) Friedman 检验图 (两种方法间若没有重叠则代表着有显著差异) [113]。(b) 每种剪枝方法根据测试准确度排序的总计序值图 (序值越小，则该方法越优) [49]。(c) 总计时间代价。(d) 总计空间代价。

们还使用了双尾成对 t 检验来检查 *EPBD* 与其他对比算法之间是否有显著差异，其中显著水平为 5%，以此来说明在 *EPBD* 与其他算法的对比过程中，*EPBD* 是取胜还是平局还是失败。若两种剪枝算法之间并无显著差异，则认为它们平局；否则，获得更高准确度的剪枝算法将获胜。算法性能的对比汇报在表 3.2 中的倒数两行，分别代表其平均序值和双尾成对 t 检验的结果，即 *EPBD* 在与其他剪枝算法对比的过程中 *EPBD* 取胜、平局或者失败的次数。需要注意的一点是：有时 OO 或者 PEP 可能获得相对于 *EPBD* 而言更优的结果，但是这两种方法都无法预先确定剪枝后子集成器的规模，这就导致了在多数情形中，它们都维持了比我们预期中更大规模的子集成器。而且，尽管它们偶尔能够获得比 *EPBD* 更优的子集成器，但是与 *EPBD* 所获得的结果并无显著不同。与此同时，也可以推出即使是只保持更小规模的子集成器，*EPBD* 也能够获得有竞争力的结果，这也说明了我们用以指导剪枝过程的准则是有效的，且对所提出的集成器的多样性与泛化性能之间关系的利用也是合理的。图 3.9(a) 也展示出：比之 OO、PEP、

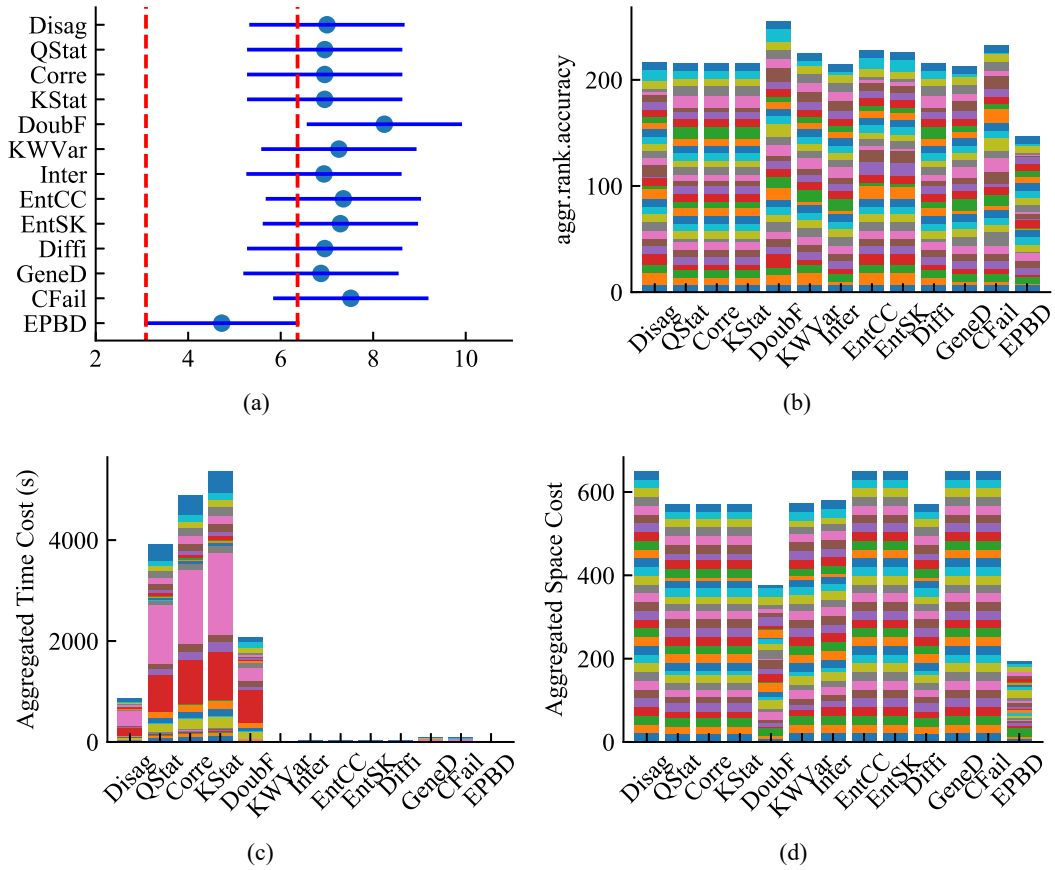


图 3.12　在自适应增强法所构造的同质集成器上对比 *EPBD* 和 *FTAD*

注：即对比 *EPBD* 与使用其他多样性度量的 *FTAD* 的测试准确度。(a) Friedman 检验图 (两种方法间若没有重叠则代表着有显著差异) [113]。(b) 每种剪枝方法根据测试准确度排序的总计序值图 (序值越小，则该方法越优) [49]。(c) 总计时间代价。(d) 总计空间代价。

MRMR 和 TSP.AP 而言，*EPBD* 可以获得有竞争力的结果，且显著优于其他对比算法。图 3.9(b) 展示了每种剪枝算法的总计排序[①]，也可以得出与图 3.9(a) 中相似的结论。图 3.9(c)–3.9(d) 分别展示了每种剪枝算法的总计时间代价和总计空间代价，可以推断出：PEP、DiscEP 和 LCS 通常需要消耗巨大的时间代价来完成集成剪枝任务。参考图 3.10 也可得到类似的结论。

3.4.4　对比 *EPBD* 及使用其他多样性度量的 *FTAD*

本小节评估了使用其他多样性度量方法的 *FTAD* 的性能，包括成对的多样性度量 (Disag、QStat、Corre、KStat 和 DoubF) 和非成对的多样性度量 (KWVar、Inter、EntCC、EntSK、Diffi、GeneD 和 CFail)，并与 *EPBD* (即在 *FTAD* 中使用本章所提出的多样性度量的一种特例) 进行对比。实验结果汇总在表 3.4–3.6 中，

[①]说明：以图 3.9(b) 为例，该子图中的不同颜色代表不同实验中的结果，即在同一实验中所得的不同剪枝方法的实验结果将使用同一种颜色来表达。

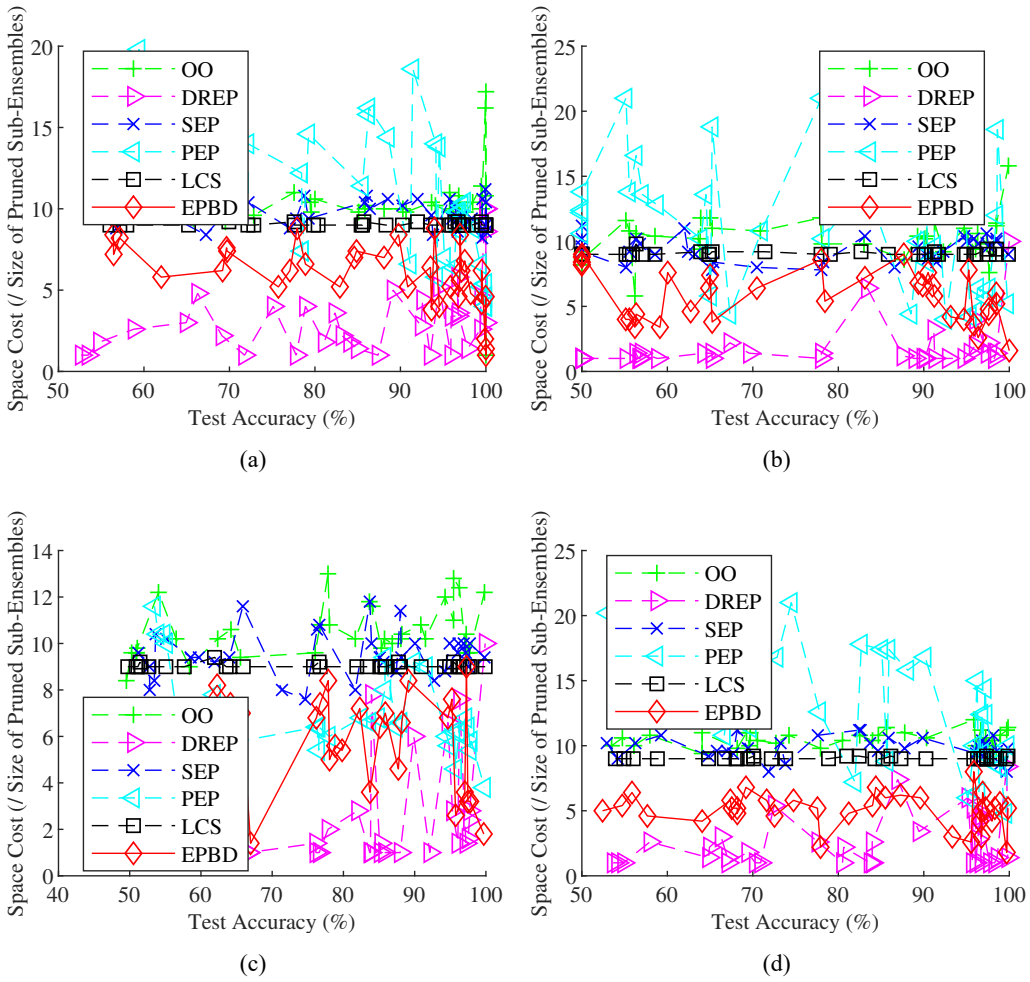


图 3.13　剪枝后子集成器规模不定的剪枝算法与 *EPBD* 的空间代价对比

注：虽然 *EPBD* 无法确定剪枝后子集成器的具体规模，但是它可以确保其值不会超出由剪枝率所确定的子集成器的预期规模上限。(a–d) 使用自助法构建集成器，所用到的基分类器分别为决策树、支持向量机、线性支持向量机和 k 近邻分类器。

包括每种方法使用标准 5 折交叉验证法在每个数据集上的平均测试准确率及其相应的标准差。以表 3.4 为例，其中每一行 (即每一个数据集) 对比的都是当使用 Bagging 构造同质集成器时的不同算法的性能，并用粗体字标注了更高的准确度和更低的标准差。此外，我们还使用了双尾成对 t 检验来检查在 *FTAD* 中所使用的两种不同的多样性度量之间是否有显著差异，其中显著水平为 5%。若两种多样性度量之间并无显著差异，则认为它们平局；否则，获得更高准确度的多样性度量将获胜。算法性能的对比汇报在表 3.4 中的倒数两行，分别代表其平均序值和双尾成对 t 检验的结果，即使用所提出的多样性度量的 *EPBD* 在与其他多样性度量对比的过程中 *EPBD* 取胜、平局或者失败的次数。此外，图 3.11–3.12 中展示了使用不同多样性度量的时间代价和空间代价，可以看出：在 *FTAD* 中使用成对的多样性度量在剪枝过程中往往需要消耗更多的时间，如图 3.11(c) 及图 3.12(c)

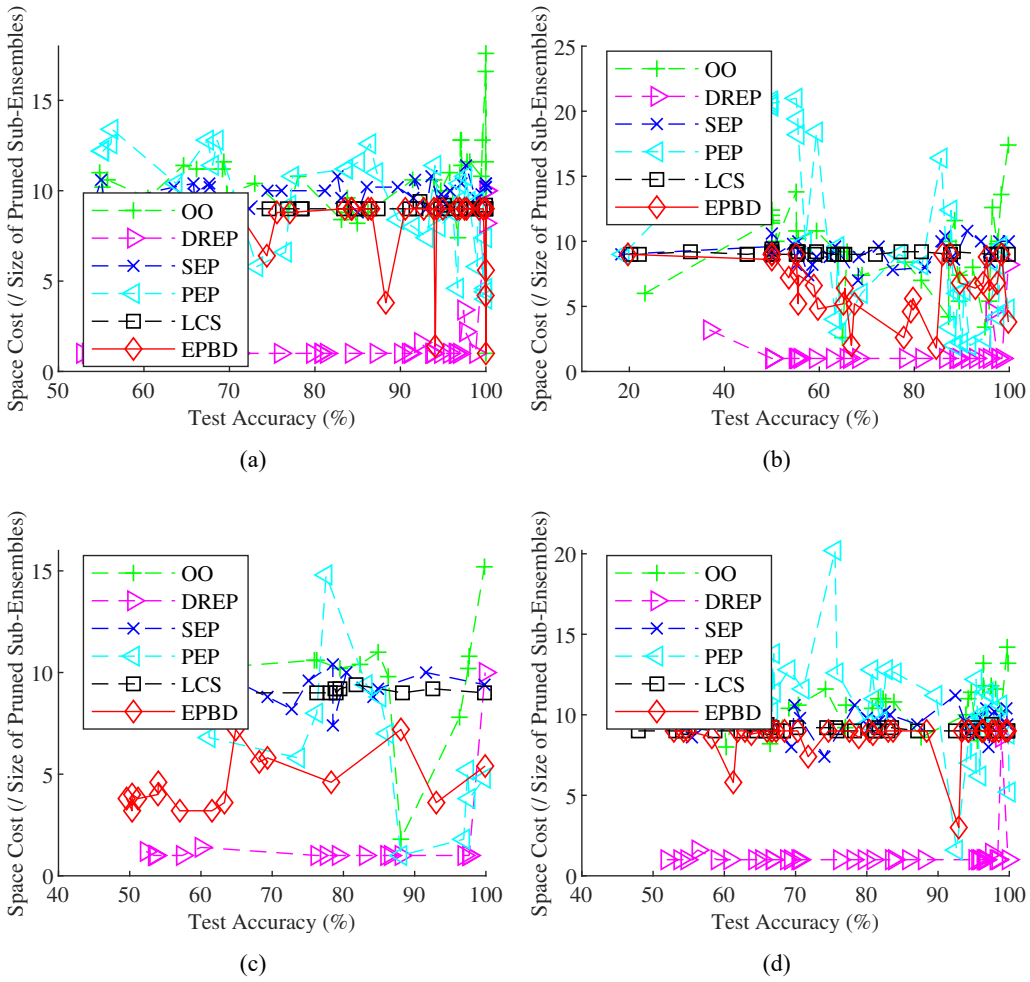


图 3.14　剪枝后子集成器规模不定的剪枝算法与 *EPBD* 的空间代价对比

注：虽然 *EPBD* 无法确定剪枝后子集成器的具体规模，但是它可以确保其值不会超出由剪枝率所确定的子集成器的预期规模上限。(a–d) 使用自适应增强法构建集成器，所用到的基分类器分别为决策树、支持向量机、线性支持向量机和 k 近邻分类器。

所示。

3.4.5　对比 *EPBD* 与基准算法的空间代价

本小节对比了 *EPBD* 与其他无法确定剪枝后子集成器规模的剪枝算法 (如 OO、DREP、SEP、PEP 和 LCS)，主要关注于它们的测试准确度和空间代价比较。实验结果在图 3.13–3.14 中汇报。如图 3.13–3.14 所示，在大多数情况下，DREP 在剪枝后都只维持了一个基分类器，其实也根本构不成一个集成器；而相较于所预期的子集成器的规模上限，OO、SEP 和 PEP 通常会生成更大规模的子集成器；由 LCS 所生成的子集成器的规模相对来说更为稳定，只是在子集成器的预期规模上限附近上下浮动。此外，虽然 *EPBD* 也无法完全确定剪枝后子集成器的规模，但是至少它可以确保子集成器的规模不会超过由剪枝率所确定的剪枝后子

集成器的预期规模上限。因此，我们可以做出结论，即相对比于未剪枝的原始集成器而言，*EPBD* 能够如预期般获得性能良好且规模更小的子集成器。

3.5　本章小结

本章通过探索和利用分类集成器中的多样性，为研究多样性在集成器中所起到的作用这一课题提出了一种新的观点。更具体地说，受启发于回归问题的误差分解，本章首先提出了分类问题中集成器的误差分解，其中集成器的误差损失可以被划分为两项，一项是基分类器的平均损失，而另一项则被我们定义为多样性度量。经验结果也确认了我们所提出的多样性度量是一种合理而有效的度量方法。其次，本章从理论角度探索了集成器中的多样性和其泛化性能之间的关系，这些理论分析为解决集成学习中多样性这一开放性的课题 (例如在何种情形下，多样性将对提高集成器的性能起到积极作用?) 迈出了微小的一步，也为将来更进一步的分析提供了一些理论基础。根据我们的分析，随着多样性落在不同的区域内，集成器的多样性与泛化性能之间的关系也会发生一定程度上的变化。在某些区域内，提高多样性将获得性能更优的集成器；而在另外一些区域内，提高多样性并不总是对于集成器有利。再次，为了验证和使用我们所提出的集成器中的多样性与泛化性能之间的关系，我们提出了 *EPBD* 和 *FTAD*，能够在不过多损失集成器性能的基础上有效缩减集成器的规模。虽然集成器的泛化性能的边界可能会进一步放宽，但是我们的工作可以为将来揭示多样性对于集成器泛化性能的影响相关的更进一步的理论分析提供一个方向。在未来的工作中，我们会考虑将所提出的方法扩展到多分类问题中去。

第 4 章　基于信息熵的集成剪枝方法及其通用并行框架

本章针对集成学习中的剪枝问题进行研究，探索了多样性在剪枝问题中的应用。集成剪枝的任务是从原始的集成器中选择出一个基分类器的子集，使其可构成一个性能良好的子集成器，其目的在于缓解集成学习所需要的巨大时间代价和空间代价的缺陷。在集成学习领域，准确度和多样性是两个重要因素，但是它们往往有所冲突。为了平衡这两者，本章将集成剪枝问题形式化为一个基于信息熵的目标最大化问题，并据此提出一种基于信息熵的目标最大化的剪枝算法，包括集中式和分布式两个版本，其中分布式版的剪枝算法是为了加速集中式版的剪枝算法而提出的。最后，本章从中抽取出一个可普适于现有剪枝算法的通用并行框架，能够在不损害剪枝后子集成器性能的同时加速剪枝算法执行的过程。实验结果也验证了本章所提出的算法和框架的有效性。

4.1　背景介绍

集成学习因其出众的潜力而引起了诸多机器学习领域研究者的注意 [2]，且被应用到了许多实际问题中，如目标检测、目标识别和目标跟踪等 [21, 23-24, 116]。集成学习，又可被称作是基于委员会的学习、多分类器系统或是专家组合等 [2-3, 15]，一个集成器是一组基学习器的集成，且其预测结果由这些基学习器的结果共同决定，而非仅依赖于单个基分类器的预测。根据基学习器类型的不同可以获得同质集成器或是异质集成器，当前大多数集成方法关注的主要是同质集成器，如自助法 [4] 和提升法 [16-17] 等。

集成学习的成功主要来源于两大要素，即基学习器的准确度和这些基学习器之间的多样性 [1]。对于分类问题而言，若一个分类器在新样本上犯错的概率优于随机猜测，则它是一个准确的分类器；若两个分类器在新样本上犯错的可能有所不同，则它们是两个多样的分类器 [1]。然而不幸的是，不像准确度，目前集成学习领域尚无对于多样性的公认定义或者度量方法。且随着基分类器准确度的提高，基分类器之间的多样性往往会有所下降。因此，如何对这两者进行合理的平衡就变成了集成学习中的一个重要问题。

虽然集成方法非常有效，但是也有一个显著的缺点，那就是随着基分类器数量的增多，它们往往需要更大的空间代价和时间代价。为了缓解这一问题，集成剪枝方法应运而生。集成剪枝，又被称为是集成选择或是集成稀疏 [38-44, 117]，其目标在于选出原始集成器中的一个基分类器的子集，从而构成一个规模更小的子集成器。尽管剪枝后子集成器的规模缩小了，但是在某些情况下，这些子集

成器甚至能获得比原始集成器更优的性能 [45]。在过去的二十年间，有许多集成剪枝方法被提出，但是这些方法大多是属于集中式的方法，即它们只能在单机上存储并执行。而在大数据的时代背景下，随着数据规模和集成器规模的急剧扩大，集成式方法的时间性能逐渐变成了算法执行时间的一个瓶颈，这也正是为何我们需要研究分布式方法的原因。

为了更快速地解决集成剪枝问题，同时有效地平衡准确度和多样性，本章首先将集成剪枝问题看作是一个基于信息熵的目标最大化问题，其中利用信息熵来隐式地反映了准确度和多样性之间的平衡，即本章用于最大化的优化目标是从信息熵角度来对准确度和多样性之间进行平衡。其次，受启发于近年来逐渐涌现的 “可组合核心集” 这一概念，本章将这种剪枝方法成功改造成了可以并行化执行的版本，其目的是为了加速剪枝算法的执行过程。不止如此，改造后的算法所采用的类似于分治法的策略也很适合于并行化的执行设置。再次，本章从这种分布式剪枝方法中抽取出了一个可普适于现有剪枝方法的通用并行框架。这一框架可以被广泛应用于现有的剪枝算法上，并大大加速它们的算法执行过程，同时也不会对原有剪枝算法的性能带来过多的损害。

简而言之，本章的贡献包括以下四个部分：

- 从信息熵角度把集成剪枝问题形式化为了一个目标最大化问题，合理地平衡了集成学习中的两大重要因素，即准确度和多样性。
- 提出了一种基于目标最大化的剪枝方法，同时利用了准确度和多样性，且该剪枝方法包括集成式和并行式两个版本。
- 提出了一个关于集成剪枝问题的通用并行框架，且该框架可广泛应用于现有的剪枝方法，在缩减其时间代价的同时也不会对算法性能造成过多损害。
- 在实际数据集上的实验结果也验证了本章所提出的剪枝算法和剪枝框架的有效性。

本章其余部分安排如下：首先在第 4.2 节中介绍集成学习中的多样性和集成剪枝方法，并概述了可组合核心集和本章的贡献；然后在第 4.3 节中介绍所提出的方法，包括基于目标最大化的剪枝方法和集成剪枝的通用并行框架；随后在第 4.4 节中构建实验来评估所提出的方法，并与现有的剪枝方法进行比较；最后在第 4.5 节中给出本章小结以及未来的工作计划。

4.2　相关工作

本节将简要介绍四点：1. 集成学习中的要素 (即多样性) 及其相关研究；2. 集成剪枝问题的难点和现有的解决方案；3. 可组合核心集的概念和相关发展；4.

本章的主要贡献，详细阐述了本章所提出方法与现有剪枝方法的主要区别。

4.2.1　集成学习中的多样性

多样性，又可被称为是依赖性、正交性或是互补性，是集成学习领域中的重要概念 [3]。从直观意义上理解，它被描述为集成器中基分类器之间的意见差异程度 [2]。从实践上来说，为了获得多样的基分类器，一种常见的方法是在训练数据的不同子集上分别训练若干个基分类器，但是这样获得的基分类器之间具有高度的相关性，违背了 “基分类器之间相互独立” 的假设，也就很难获得真正多样的基分类器。无数集成方法试图通过隐式或启发式的方法来鼓励更多的多样性 [35]。例如提升法和自助法分别试图通过在训练样本上重新分配权重和重新采样的方法来促进多样性 [4, 16-17, 118-119]；神经网络的集成器也可以通过不同的初始权重、不同的网络结构和不同的学习算法来创建多样的子网络。

然而截至目前，研究者尚未就多样性度量的正式定义这一议题达成共识。他们提出了很多种度量方法用于表达多样性，这些方法通常可以划分为成对的和非成对的多样性两类 [3]，但是没有研究证明哪种方法显著优越于其他的度量方法 [15]。根据两个基分类器在样本上同时发生的错误情形，成对的多样性度量表达了二者之间在预测样本时意见是否一致或者是有异议的情况，这类方法包括 Q 统计量 [55, 120]、κ 统计量 [56]、分歧度量 [57-58]、相关系数度量 [59]、双重失误度量 [60] 等。而集成器的多样性则是所有可能的基分类器对的多样性的平均值。与之相反，非成对的多样性度量直接度量了集成器中所有基分类器之间的意见差异性，这类方法包括测试者彼此之间的一致性度量 [79]、Kohavi-Wolpert 差异度量 [62]、投票的熵度量 [63-64]、难度指数度量 [3, 65]、广义多样性 [66]、一致失误多样性 [66] 等。除此之外，还有两种度量方法无法被归类到上面的两种分类，其一是相关惩罚函数 [67]，度量了在负相关学习 [121-122] 中的集成器与其基分类器之间的多样性；其二是模糊项 [68]，度量了基分类器相对于整体集成器的偏置平均值。

不仅仅是多样性的度量尚未得以定论，关于多样性究竟在集成器中起到了什么样的量化作用这一问题，也很少有研究能得出一个广为学者所接受的结论。在过去十年间，Brown [32] 从信息理论的角度提出，集成器中的多样性的确存在，且并不仅限于一种形式存在，而是以基分类器之间的多水平的交互作用形式存在着。他们的研究工作启发了 Zhou et al. [33]，使其进一步深入研究并从互信息角度提出了：应当最大化共有信息，其目的在于最小化集成器的预测误差。随后，Yu et al. [35] 提出了一种基分类器之间的成对形式的多样性，并将其用于所提出的多样性正则机，能够缩减假设空间复杂度，揭示了控制多样性在集成方法中相当于是一种正则方法。

4.2.2　集成学习中的剪枝方法

集成剪枝希望缩减原始集成器的规模，可以在提高效率的同时不会过多损害集成器的性能 [47]。Margineantu et al. [70] 展示了 “从原始集成器中选择出一个子集，能够获得与原始集成器相同水平的预测性能” 的可能性，这也是集成剪枝问题研究的开端。Zhou et al. [45] 提出了集成器的误差-方差分解，将其作为所提出方法 “基于基因算法的集成剪枝” 成功的主要原因，并提出了集成剪枝可以在获取更小规模的集成器的同时，也获得性能更优的子集成器。但是，从原始集成器中挑选出能够获得最优泛化性能的子集是很困难的。困难之一在于估计子集成器的泛化误差本身就是一个难题；困难之二在于在一个集合中寻找最优子集是一个计算复杂度为幂指数级的组合搜索问题 [48]。从集成器中寻找到一个最佳组合是一个 NP 完全难问题，甚至连求其近似解都很难处理 [71]。

在过去的二十年间，无数集成剪枝方法被提出，用于克服集成学习所需巨大时空代价的缺点。这些剪枝方法可以被大体分为三类，即基于排序的剪枝方法、基于聚类的剪枝方法和基于优化的剪枝方法。从概念来说最简单的是基于排序的剪枝方法，它们会根据某种评估准则对所有基分类器进行排序，然后选出前若干个作为剪枝后的子集成器。根据评估准则的不同可以获得不同的剪枝方法。这些评估准则包括最小化误差，如定向排序剪枝 [72]；最大化多样性，如 KL 散度剪枝和 Kappa 剪枝 [70]；或者是结合两者，如多样性正则化集成剪枝 [48]。基于聚类的剪枝方法首先使用聚类算法将原始集成器集合划分成若干组基分类器，预测结果相似的基分类器将被划分入一组；随后对每一组基分类器集合进行剪枝，最后合并作为最终的剪枝后的子集成器，这样做可以提高基分类器之间的多样性 [47]。需要注意的一点是这类方法的第二阶段从本质上来说可以自然地以并行化的方式执行，但就目前来说这一性质常常被忽视了。而基于优化的剪枝方法则是将集成剪枝作为一个优化问题来处理，往往会定义一个可以隐含剪枝后子集成器的泛化性能的优化函数，并根据这个优化目标来寻找原始集成器中的子集。但即使是对于一个中等规模的原始集成器而言，穷尽搜索集成器子集空间里的所有解也不甚可行，因为这是一个 NP 完全难问题 [48, 71]。因此，研究者利用了大量的技术手段来缓解这一困境，包括基因算法 [73]、贪婪算法 [74]、爬山法 [75] 和双目标演化优化算法 [49] 等。

4.2.3　可组合核心集

在过去的几年内，一种名为 “可组合核心集” 的有效技术逐渐进入了人们的视野。这种技术出现在分布式计算领域，其目标在于在大规模数据集中解决优化问题。在许多机器学习应用中，它的有效性已经从经验实践的角度得以确认，如

多样最近邻搜索 [112]、多样性最大化 [111] 和特征选择 [123] 等。

“可组合核心集” 的概念首次出现于 Indyk et al. [112] 的研究工作中，而 “核心集” 相关的研究则可追溯于 2004 年 [124]。在优化问题中，一个核心集被定义为是一个带有可保证的近似因子的子集，满足条件为在该子集上求解可获得从原始数据上求解的近似解。可组合核心集则是一组核心集，且这些核心集的并集能够给出对于原始数据子集的并集的一个核心集 [112]。不止于此，一个带有 α 近似因子的可组合核心集可以获得原优化问题最优解的一个近似解，且近似程度由该近似因子 α 确保，其中 α 值为 $1/12$ [111]，但可以被提高到 $8/25$ [123]。

4.2.4　本章贡献概述

本章所提出的方法与现有的剪枝方法存在一个重大的区别，那就是我们的方法能够以一种并行化的方式来执行，这可以大幅加速剪枝过程，缩短所需耗时。在本章所提出的剪枝方法中，我们没有采用现有的多样性度量方法，而是从信息熵的角度来隐式表达，并利用共有信息 $\mathrm{I}(\cdot;\cdot)$ [125]、标准共有信息 $\mathrm{MI}(\cdot,\cdot)$ 和标准信息变异 $\mathrm{VI}(\cdot,\cdot)$ [123] 来构造意欲优化的函数目标，同时考虑了准确度和多样性的平衡。此外，本章所提出的 “基于多样性的集成剪枝框架 (Diversity-Based Framework for Ensemble Pruning, *EPDF*)” 可以被广泛应用到现有的剪枝算法中，能够在不影响原算法剪枝效果的同时，大幅缩短算法执行的耗时，这也正是所提出方法的一大优势所在。

4.3　方法介绍

本节首先详细阐述了基于信息熵的目标最大化的集成剪枝算法，然后利用 “可组合核心集” 的概念将这一算法改造成可以并行执行的版本，最后抽取出了一个可普适于现有剪枝方法的并行化框架。

4.3.1　集中式的基于目标最大化的集成剪枝算法

给定一个大型数据集 $\mathcal{D}$, 包括 m 个带标签的样本，且这些样本的标签可表示为一个 m 维向量，记作 $\mathbf{y}$. 原始集成器由一组已训练好的基分类器通过带权投票法的方式构成，即 $\mathcal{H} = \{h_1(\cdot), h_2(\cdot), ..., h_n(\cdot)\}$, 其中每个函数都代表了从样本特征空间到标签空间的一个映射。与数据集的标签向量相似的是，其中每个基分类器 h_i 在该数据集上的预测结果也可由一个 m 维向量表示，即 $\mathbf{h}_i$. 集成剪枝任务的目标是从原始集成器中寻找一个紧致子集，且该子集所构成的子集成器也拥有良好的性能。为了达成这一目的，这些被保留在子集成器中的基分类

器既需要是准确的，也需要相互之间是多样的。为了达成这一目的，我们的目标是从原始集成器中选择出一些多样化的基分类器，且这些被选中的基分类器均与标签向量有所关联。受启发于 [123]，我们定义了一个基分类器之间的距离度量，能够同时考虑基分类器的准确度及彼此间的多样性。此时，集成剪枝问题就可以被形式化成是一个目标最大化的问题。

给定两个离散变量 X 和 Y, Cover et al. [125] 定义了两者之间的*共有信息* I(·;·), 即

$$\begin{aligned}\mathrm{I}(\mathsf{X};\mathsf{Y}) =& \mathrm{H}(\mathsf{X}) - \mathrm{H}(\mathsf{X}|\mathsf{Y}) \\ =& \sum_{x\in\mathsf{X},y\in\mathsf{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}\,,\end{aligned} \tag{4.1}$$

随后，Zadeh et al. [123] 定义了两者之间的*标准共有信息* MI(·, ·), 即

$$\begin{aligned}\mathrm{MI}(\mathsf{X},\mathsf{Y}) =& \frac{\mathrm{I}(\mathsf{X};\mathsf{Y})}{\sqrt{\mathrm{H}(\mathsf{X})\mathrm{H}(\mathsf{Y})}} \\ =& \frac{\sum_{x\in\mathsf{X},y\in\mathsf{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}}{\sqrt{\sum_{x\in\mathsf{X}} p(x)\log p(x)\sum_{y\in\mathsf{Y}} p(y)\log p(y)}}\,,\end{aligned} \tag{4.2}$$

和*标准信息变异* VI(·, ·), 即

$$\begin{aligned}\mathrm{VI}(\mathsf{X},\mathsf{Y}) =& 1 - \frac{\mathrm{I}(\mathsf{X};\mathsf{Y})}{\sqrt{\mathrm{H}(\mathsf{X},\mathsf{Y})}} \\ =& 1 - \frac{\sum_{x\in\mathsf{X},y\in\mathsf{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}}{-\sum_{x\in\mathsf{X},y\in\mathsf{Y}} p(x,y)\log p(x,y)}\,,\end{aligned} \tag{4.3}$$

其中 $p(\cdot,\cdot), \mathrm{H}(\cdot)$ 和 $\mathrm{H}(\cdot,\cdot)$ 分别代表联合概率、熵函数和联合熵函数。

考虑样本标签向量 $\mathbf{y}$ 和两个基分类器 (即 $h_i(\cdot)$ 和 $h_j(\cdot)$) 在数据集 $\mathcal{D}$ 上的预测结果向量 (即 $\mathbf{h}_i$ 和 $\mathbf{h}_j$)，那么标准共有信息 $\mathrm{MI}(\mathbf{h}_i,\mathbf{y})$ 可以表达该基分类器 $h_i(\cdot)$ 与标签向量 $\mathbf{y}$ 之间的相关性，即反映了该基分类器在训练数据集上的准确度；而标准信息变异 $\mathrm{VI}(\mathbf{h}_i,\mathbf{h}_j)$ 可以表达两个基分类器 $h_i(\cdot)$ 和 $h_j(\cdot)$ 之间的冗余度，即反映了它们之间的多样性。由于我们研究的是分类问题，标签向量已经是离散变量，且其中的元素值只与数据集中的标签空间有关，所以我们在计算标准共有信息和标准信息变异所需要用到的概率值时，不需要像 Zadeh et al. [123] 一样先将连续变量离散化。

为了同时考虑多样性和准确度，我们将*两个基分类器之间的多样性与准确度的平衡 (a trade-off between diversity and accuracy of two individual classifiers,* TDAC*)* 定义为

$$\mathrm{TDAC}(h_i,h_j) = \begin{cases}\lambda\mathrm{VI}(\mathbf{h}_i,\mathbf{h}_j) + (1-\lambda)\frac{\mathrm{MI}(\mathbf{h}_i,\mathbf{y})+\mathrm{MI}(\mathbf{h}_j,\mathbf{y})}{2}\,, & \text{若 } h_i \neq h_j\,; \\ 0\,, & \text{否则}\,,\end{cases} \tag{4.4}$$

其中引入了一个正则因子 λ 用于平衡多样性和准确度，也暗示了它们的重要性程度。在式 (4.4) 中，第一项通过避免两个基分类器间的冗余度过小而隐式地促

进多样性的提高；而第二项则是通过最大化基分类器与标签向量之间的相关程度而隐式地提高准确度。注意 $\mathrm{VI}(\cdot,\cdot)$ 是一种度量函数，且 $\mathrm{MI}(\cdot,\mathbf{y})$ 是非负的，则 $\mathrm{TDAC}(\cdot,\cdot)$ 也是一种度量函数，这就意味着

$$\mathrm{TDAC}(h_i,h_j)+\mathrm{TDAC}(h_j,h_k)\geqslant\mathrm{TDAC}(h_i,h_k)\,, \tag{4.5}$$

其中 $h_i(\cdot)$, $h_j(\cdot)$ 和 $h_k(\cdot)$ 是相异的基分类器。

获得了两个基分类器之间的多样性与准确度的平衡之后，紧随其后的，我们可以将整个集成器的多样性与准确度的平衡 *(a trade-off between diversity and accuracy of an ensemble,* TDAS*)* 定义为

$$\mathrm{TDAS}(\mathcal{H})=\frac{1}{2}\sum_{h_i\in\mathcal{H}}\sum_{h_j\in\mathcal{H}}\mathrm{TDAC}(h_i,h_j)\,. \tag{4.6}$$

需要注意的是，式 (4.6) 中的 $\mathrm{TDAS}(\mathcal{H})$ 可被重写为

$$\mathrm{TDAS}(\mathcal{H})=\frac{1}{2}\lambda\sum_{h_i\in\mathcal{H}}\sum_{h_j\in\mathcal{H}}\mathrm{VI}(\mathbf{h}_i,\mathbf{h}_j)+\frac{n-1}{2}(1-\lambda)\sum_{h_i\in\mathcal{H}}\mathrm{MI}(\mathbf{h}_i,\mathbf{y})\,, \tag{4.7}$$

其中，若两个基分类器越相似，则它们的标准信息变异 $\mathrm{VI}(\cdot,\cdot)$ 的值就会越接近于零。注意式 (4.7) 的第一项意在避免选择出过多相似的基分类器，即被选中的基分类器是多样的；而其第二项则意在保证这些被选中的基分类器与样本的标签向量是相关的，即这些基分类器是准确的；至此，集成剪枝任务就被形式化为了一个基于信息熵的目标最大化问题，即

$$\max_{\substack{\mathcal{P}\subset\mathcal{H},\\|\mathcal{P}|=k}}\mathrm{TDAS}(\mathcal{P})=\max_{\substack{\mathcal{P}\subset\mathcal{H},\\|\mathcal{P}|=k}}\frac{1}{2}\sum_{h_i\in\mathcal{P}}\sum_{h_j\in\mathcal{P}}\mathrm{TDAC}(h_i,h_j)\,, \tag{4.8}$$

其目标在于寻找一个子集 $\mathcal{P}$, 且满足 $\mathcal{P}\subset\mathcal{H}$ 和 $|\mathcal{P}|=k$, 其中后者用于限制剪枝后子集成器的规模。

在集成剪枝任务中同时考虑准确度和多样性是一个双目标优化问题，但是通过引入式 (4.4) 中的 $\mathrm{TDAC}(\cdot,\cdot)$, 我们可将其转换成一个单目标优化问题。这只是一种利用目标赋权的解决方法 [126]，也可以通过引入 “支配” 的概念来获得一个帕累托最优解 [49, 127]，它将是我们未来的研究方向。除此之外，我们的方法可以预估剪枝后子集成器的规模，而后者无法确保此事，且常常会获得相较预期而言规模更大的子集成器，这将影响到算法执行所需要的空间代价。

至此，我们已将集成剪枝任务形式化为了一个如式 (4.8) 所示的基于信息熵的目标最大化问题，这足以形成一个集中式的集成剪枝算法。我们将所提出的集中式的剪枝方法命名为 “集中式的基于目标最大化的集成剪枝算法 *(Centralized Objection Maximization for Ensemble Pruning, COMEP)*”，如算法 4.1 所示。在该算法中，每一步都是贪婪地选择当前更优的基分类器。由 [128] 可知，这种算法对基于目标函数最大化的优化问题来说，可以获得 $1/2$ 的近似因子。

4.3.2　分布式的基于目标最大化的集成剪枝算法

“分布式的基于目标最大化的集成剪枝算法 *(Distributed Objection Maximization for Ensemble Pruning, DOMEP)*” 是 “集中式的基于目标最大化的集成剪枝算法 (Centralized Objection Maximization for Ensemble Pruning, *COMEP*)” 算法的分布式版本，如算法 4.2 所示。*DOMEP* 采用了两阶段的分治策略和可组合核心集作为指导，这恰好也适合于分布式的执行设置。首先，它将一组基分类器划分为若干个子集，随后在每个子集上分别求解集成剪枝问题，最后从这些解的并集中获得一个最终的解。

考虑一组已训练好的基分类器集合，即 $\mathcal{H} = \{h_1(\cdot), h_2(\cdot), ..., h_n(\cdot)\}$, 作为原始的集成器。在算法的第一阶段，一台主机器将原始集成器中所有的基分类器随机划分成 m 组，即 $\{\mathcal{H}_i\}_{i=1}^m$，并将它们分配到不同的机器上。注意 $\cup_{i=1}^m \mathcal{H}_i = \mathcal{H}$, 其中 m 是机器数，且这台主机器可以是这 m 台机器中的任意一个。对于每一台机器来说，如机器 i 将在分配给它的子集 $\mathcal{H}_i$ 独立地执行剪枝算法 *COMEP*，并得到剪枝后的子集 $\mathcal{P}_i$; 且这 m 台机器相互之间是并行执行的。在算法的第二阶段，主机器将所有剪枝后的子集汇总到一起，并在它们的并集 (即 $\cup_{i=1}^m \mathcal{P}_i$) 上执行剪枝算法 *COMEP*，以获得剪枝后的子集 $\mathcal{P}'$; 然后根据式 (4.8) 将 $\mathcal{P}'$ 与 $\mathcal{P}_i\,(1 \leqslant i \leqslant m)$ 进行比较，最终输出这些剪枝后子集中的最好的一个。在实践中，经过这两个阶段后，已足以输出令人满意的剪枝后的子集成器了 (如算法 4.2 中的第 1–5 行所示)；而额外的比较过程则是为了获得一个更好的近似解，如从理论上来说该算法的近似因子是 $1/4$, 但是在某些特殊条件下也可以达到 $8/25$ [123]。

最后需要说明的一点是，关于 “如何将 $\mathcal{H}$ 中的 n 个基分类器随机均匀地划分成 m 组” 这一问题，也许有些读者会有疑问。这里的第一步是将这些基分类器以一种随机的方式打乱顺序；然后第二步是将它们安排到 m 组中去。当 n 可以被 m 整除时，如何将这 n 个基分类器划分成 m 组想必没有任何疑问，也就是每组将包括 n/m 个基分类器。但是当 n 无法被 m 整除时，这些基分类器的划分方式将变成：在其中的 $(n \mod m)$ 组中，每组包括 $\lceil n/m \rceil$ 个基分类器；而在

Data: 原始集成器集合 $\mathcal{H}$, 代表剪枝后子集成器规模的阈值 k
Result: 剪枝后的子集成器集合 $\mathcal{P}$ (满足 $\mathcal{P} \subset \mathcal{H}$ 和 $|\mathcal{P}| \leqslant k$)
1 $\mathcal{P} \leftarrow$ 从 $\mathcal{H}$ 中选择任意一个基分类器 h_i ; // 初始化
2 **for** $2 \leqslant i \leqslant k$ **do**
3 　$h^* \leftarrow \arg\max_{h_i \in \mathcal{H} \setminus \mathcal{P}} \sum_{h_j \in \mathcal{P}} \mathrm{TDAC}(h_i, h_j)$;
4 　将 h^* 从 $\mathcal{H}$ 中移入 $\mathcal{P}$;
5 **end**

算法 4.1: 集中式的基于目标最大化的集成剪枝算法 (*COMEP*)

Data: 原始集成器集合 $\mathcal{H}$, 代表剪枝后子集成器规模的阈值 k, 可并行执行的机器数 m
Result: 剪枝后的子集成器集合 $\mathcal{P}$ (满足 $\mathcal{P} \subset \mathcal{H}$ 和 $|\mathcal{P}| \leqslant k$)
1 将 $\mathcal{H}$ 随机均匀划分成 m 组，即 $\{\mathcal{H}_1, \mathcal{H}_2, ..., \mathcal{H}_m\}$;
2 **for** $1 \leqslant i \leqslant m$ **do**
3 　$\mathcal{P}_i \leftarrow$ *COMEP* $(\mathcal{H}_i, k)$;
4 **end**
5 $\mathcal{P}' \leftarrow$ *COMEP* $(\cup_{i=1}^{m} \mathcal{P}_i, k)$;
6 $\mathcal{P} \leftarrow \arg\max_{\mathcal{T} \in \{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_m, \mathcal{P}'\}} \mathrm{TDAS}(\mathcal{T})$;

算法 4.2: 分布式的基于目标最大化的集成剪枝算法 (*DOMEP*)

Data: 原始集成器集合 $\mathcal{H}$, 可并行执行的机器数 m, 任意一种集成剪枝方法 *ALG*
Result: 剪枝后的子集成器集合 $\mathcal{P}$ (满足 $\mathcal{P} \subset \mathcal{H}$)
1 将 $\mathcal{H}$ 随机均匀划分成 m 组，即 $\{\mathcal{H}_i\}_{i=1}^{m}$;
2 **for** $1 \leqslant i \leqslant m$ **do**
3 　$\mathcal{P}_i \leftarrow$ 任意剪枝算法 *ALG* 在集成器 $\mathcal{H}_i$ 上的输出;
4 **end**
5 $\mathcal{P}' \leftarrow$ 任意剪枝算法 *ALG* 在集成器 $\cup_{i=1}^{m} \mathcal{P}_i$ 上的输出;
6 $\mathcal{P} \leftarrow$ 根据某些准则对 $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_m$ 和 $\mathcal{P}'$ 加以比较，输出其中最好的一个，其中比较准则可以自定，如根据准确度比较;

算法 4.3: 可通用的分布式集成剪枝框架 (*EPFD*)

另外的 $(n \operatorname{mumble} m)$ 组中，每组包括 $\lfloor n/m \rfloor$ 个基分类器。其中根据 [129] 可知，$n \operatorname{mumble} m = m\lceil n/m \rceil - n$.

4.3.3　可通用的分布式集成剪枝框架

从上节的 “分布式的基于目标最大化的集成剪枝算法 (Distributed Objection Maximization for Ensemble Pruning, *DOMEP*)” 中，我们可以抽取出一个可普适于现有剪枝算法的并行执行框架，并命名为 “*可通用的分布式集成剪枝框架 (Ensemble Pruning Framework in a Distributed Setting, EPFD)*”，如算法 4.3 所示。它同样采用了两阶段的分治策略和可组合核心集 [130] 的概念，能够以一种并行化的方式快速解决集成剪枝问题。

集成剪枝问题通常被描述成是从一个原始集成器中选取一个最优子集的过程。令 *ALG* 表示任意一个集成剪枝算法，且 $\mathcal{H}$ 表示一个未经剪枝的原始集成器。与 *DOMEP* 相似，*EPFD* 也包括两个算法阶段，只不过在 *DOMEP* 中使用的剪枝算法是 *COMEP*，故 *DOMEP* 相当于 *EPFD* 的一个特例 (如算法 4.3 中的第 3, 5 行所示)。另一个不同的要点是 *EPFD* 中的比较准则 (如算法 4.3 中的第 6 行所示)

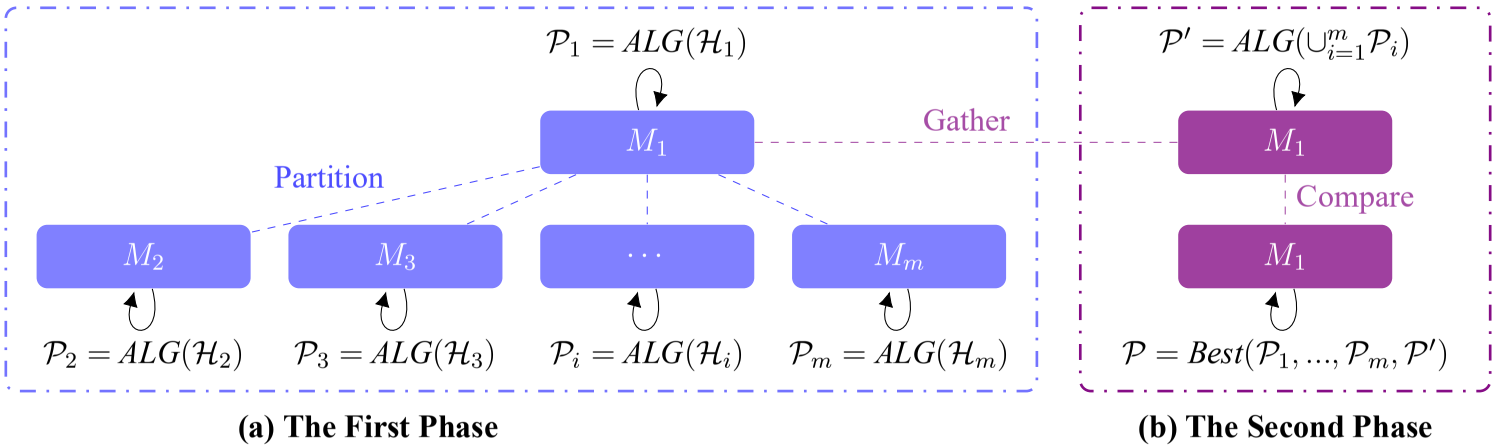


图 4.1　可通用的分布式集成剪枝框架 (*EPFD*) 示意图

注：即算法 4.3 (Ensemble Pruning Framework in a Distributed Setting, *EPFD*)。(a) 在算法的第一阶段，一台主机器 (如 M_1) 将原始集成器中的所有基分类器随机均匀地划分成 m 组，即 $\{\mathcal{H}_i\}_{i=1}^{m}$，并将它们分配到不同的机器上去，然后在每台机器上分别独立地并行地执行剪枝算法。如对于 $i\,(1 \leqslant i \leqslant m)$ 来说，机器 M_i 在分配给它的基分类器子集 $\mathcal{H}_i$ 上独立地执行剪枝算法 *ALG*，可获得剪枝后的子集成器 $\mathcal{P}_i$。这个过程对于 m 台机器来说是可并行化执行的过程。(b) 在算法的第二阶段，主机器 (如 M_1) 将上一阶段所获得的所有剪枝后子集汇总到一起，然后在它们的并集 $\cup_{i=1}^{m}\mathcal{P}_i$ 上执行剪枝算法 *ALG*，可获得剪枝后的子集成器 $\mathcal{P}'$；随后比较这些剪枝后的子集成器，即 $\mathcal{P}_1, ..., \mathcal{P}_m$ 和 $\mathcal{P}'$，从中选取最好的一个子集作为最终的输出结果。注意：(1) *ALG* 可以是现有剪枝方法中的任意一个，包括 *COMEP*。(2) 最终的输出结果 $\mathcal{P}$ 是根据某种准则对比 $\mathcal{P}_1, ..., \mathcal{P}_m$ 和 $\mathcal{P}'$ 而得到的，这种准则可以是准确度，也可以是 *DOMEP* 中的 TDAS$(\cdot)$.

并不局限于式 (4.8)，例如它也可以使用准确度或者其他与数据相关的度量来比较这些不同子集之间的优劣。*EPFD* 看似简单，但同时也是一种非常有效的策略，能够在不损害原有剪枝算法性能的同时，大幅提高剪枝过程执行的效率。更多细节可参考第 4.4.3 节。

4.4　实验评估

本节将构建实验来评估所提出的算法，其中用到了 17 个二分类数据集和 12 个多分类数据集，包括一个包含了 12,500 张图片的图像数据集 (猫狗分类[①]) 和 28 个来自 UCI 仓库的数据集 [107]。在所用实验中都使用了标准 5 折交叉验证，其中每一次迭代，整个数据集都被划分为三部分，即 60% 作为训练集、20% 作为验证集、以及剩余的 20% 作为测试集。除此之外，本节使用自助法来构建同质集成器，其中基分类器包括决策树、朴素贝叶斯分类器、k-近邻分类器、线性模型分类器、线性支持向量机和支持向量机等。一个集成器将首先在训练集上训练得到，随后在验证集上被剪枝，最后在测试集上验证效果如何。本节所考虑用于对比的基准算法包括大量的基于排序的剪枝方法和基于优化的剪枝方法。其中所用到的基于排序的剪枝方法包括 KL 散度剪枝 (KL-divergence Pruning, KL)、Kappa 剪枝 (Kappa Pruning, KP) [70]、定向排序剪枝 (Orientation Ordering Pruning, OO) [72]、降低误差剪枝 (Reduce Error Pruning, RE) [46]、多样性正则化集成剪枝 (Diversity Regularized Ensemble Pruning, DREP) [48] 以及顺序集成剪枝 (Ordering-based Ensemble Pruning, OEP) [49]；所用到的基于优化的剪枝方法包括单目标集成剪枝 (Single-objective Ensemble Pruning, SEP) 和帕累托集成剪枝 (Pareto Ensemble Pruning, PEP) [49]。注意其中一些方法无法预先确定剪枝后的子集成器规模，如 OO、DREP、SEP、OEP 和 PEP；而另一些方法则可以通过给定一个剪枝率因子来预先确定剪枝后子集成器的规模，其中剪枝率代表的是被剪掉的基分类器所占原始集成器中的比例上限。那些无法确定剪枝后子集成器规模的剪枝方法或许会导致规模过大或过小的子集成器，这会影响到算法的空间代价。限于篇幅，本章将只汇报这些算法的时间代价和测试准确度的对比结果。

4.4.1　评估基于目标最大化的集成剪枝算法

本小节对比了现有剪枝算法 (包括 KL、KP、OO、RE、DREP、SEP、OEP 和 PEP) 和本章所提出的基于目标最大化的集成剪枝算法 (包括集中式版本 *COMEP* 和分布式版本 *DOMEP*)。注意这些用于对比的基准算法均是集中式的方法。表 4.1 中的实验结果包括每种方法在标准 5 折交叉验证下的测试准确度的平均值和其

[①] http://www.kaggle.com/c/dogs-vs-cats

表 4.1　在自助法所构造的集成器上对比基准算法和 *COMEP* 及 *DOMEP*

注：其中使用决策树作为基分类器。第二列 “Ensem” 代表未经剪枝的原始集成器的准确度。对于每一个数据集 (即每一行) 来说，更高的准确度和更低的标准差将用粗体标出。

Dataset	KL	KP	OO	RE	DREP	SEP	OEP	PEP	*COMEP*	*DOMEP*
Iono	89.43±3.44	90.29±3.41	90.57±4.69	88.57±3.03	89.71±3.70	90.86±2.96	89.71±4.33	**91.71±3.41**	91.14±3.41	91.71±3.56
Liver	61.45±8.24	62.32±5.12	64.06±7.35	61.16±4.96‡	56.23±8.41	60.87±4.35‡	64.64±4.54	62.32±4.23	62.90±4.18	**64.64±4.30**
Spam	93.28±1.13	93.21±0.93	93.30±1.43	93.54±1.16	92.08±1.48	93.43±0.82	**93.56±1.24**	93.38±0.80	93.41±0.90	93.45±1.00
Wisconsin	94.37±3.58	94.67±3.25‡	95.56±4.06	95.41±3.25	93.93±3.68	94.96±3.86‡	95.26±3.50	95.41±3.79	95.56±3.70	**95.70±3.45**
Credit	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
Landsat	97.34±0.45	97.22±0.66	97.29±0.37	97.31±0.28	96.08±1.06‡	97.26±0.26	97.34±0.35	97.29±0.56	97.15±0.43	**97.42±0.34**
Wilt	98.24±0.32	98.24±0.26	98.37±0.42	98.28±0.32	98.01±0.51	98.37±0.51	98.16±0.18	98.39±0.53	**98.41±0.33**	98.35±0.38
Shuttle	99.96±0.03	99.96±0.03	**99.97±0.02**	**99.97±0.02**	**99.97±0.02**	99.96±0.03	99.96±0.03	**99.97±0.02**	99.96±0.03	99.97±0.03
Ecoli	94.55±3.14	94.85±2.03	93.64±2.71	95.15±2.49	91.52±3.49‡	94.85±2.75	93.03±2.30	94.24±2.71	95.15±2.49	**95.15±1.98**
SensorReadings	99.25±0.40	99.19±0.50	99.41±0.47	**99.52±0.24**	99.28±0.25	99.45±0.23	99.50±0.19	99.36±0.43	99.43±0.33	99.38±0.41
t-Test (W/T/L)	0/10/0	1/9/0	0/10/0	1/9/0	2/8/0	2/8/0	0/10/0	0/10/0	0/10/0	—
Average Rank	7.15	7.20	4.80	4.60	8.40	5.55	5.20	4.75	4.40	2.95

[1] 表中所汇报的实验结果均是每种剪枝方法使用标准 5 折交叉验证的实验在各个数据集的测试集上的平均准确度 (%) 与其相应的标准差。粗体字表示更高的准确度和更低的标准差。

[2] 通过在 5% 显著水平下的双尾成对 t 检验对比各算法之间是否有明显差异，其中 ‡ 和 † 分别表示 *DOMEP* 显著优于或者是显著差于对比算法的性能；若无符号标记，则表示 *DOMEP* 与相应对比算法之间没有显著差异。

[3] 表中最后两行分别展示了检验结果和各算法的平均序值。各算法的平均序值由 Friedman 检验 [113] 计算而得，而 “W/T/L” 则表示 *DOMEP* 显著优于、并无显著不同或者显著差于相应的对比算法的次数。

表 4.2　在自助法所构造的集成器上对比基准算法和 *COMEP* 及 *DOMEP*

注：其中使用支持向量机作为基分类器。第二列 “Ensem” 代表未经剪枝的原始集成器的准确度。对于每一个数据集 (即每一行) 来说，更高的准确度和更低的标准差将用粗体标出。

Dataset	KL	KP	OO	RE	DREP	SEP	OEP	PEP	*COMEP*	*DOMEP*
Liver	**58.84±1.65**	**58.84±1.65**	58.26±1.21	**58.84±1.65**	**58.84±1.65**	58.55±1.65	**58.84±1.65**	58.55±1.65	**58.84±1.65**	**58.84±1.65**
Ringnorm	98.43±0.37	**98.53±0.30**†	98.51±0.36†	98.47±0.29	98.43±0.29	98.47±0.36	98.39±0.25	98.44±0.31	98.43±0.31	98.43±0.31
Waveform	91.45±1.60	91.23±1.50	91.35±1.58	91.17±1.67	90.83±1.52‡	91.27±1.53	91.13±1.61	91.37±1.63	**91.45±1.49**	91.23±1.63
Credit	78.02±0.07	78.01±0.11	78.03±0.08	78.02±0.08	77.96±0.12	**78.04±0.12**	78.02±0.11	78.02±0.13	78.00±0.09	78.01±0.11
Landsat	65.24±0.00	65.24±0.00	65.24±0.00	65.24±0.00	65.24±0.00	65.24±0.00	65.24±0.00	65.24±0.00	65.24±0.00	65.24±0.00
Page	91.26±0.41	91.26±0.45	91.28±0.44	91.24±0.46	91.28±0.52	91.22±0.32	91.24±0.44	**91.30±0.53**	91.26±0.45	91.22±0.44
Wilt	94.68±0.09	94.68±0.09	94.68±0.09	94.68±0.09	94.68±0.09	94.68±0.09	94.68±0.09	94.68±0.09	94.68±0.09	94.68±0.09
SensorReadings	89.17±0.78	89.24±0.88	89.28±0.52	89.35±0.51	88.91±0.60	89.11±0.91‡	89.26±0.50	**89.51±0.94**	89.42±0.83	89.37±1.04
EEGEyeState	55.13±0.00	55.13±0.00	55.13±0.00	55.13±0.00	55.13±0.00	55.13±0.00	55.13±0.01	55.13±0.00	55.13±0.00	55.13±0.00
WaveformNoise	86.35±0.74	86.11±0.83	86.37±0.79	86.21±0.94	85.99±0.83	86.49±1.02	86.11±0.98	86.29±0.77	**86.55±0.68**	86.43±0.81
t-Test (W/T/L)	0/10/0	0/9/1	0/9/1	0/10/0	1/9/0	1/9/0	0/10/0	0/10/0	0/10/0	—
Average Rank	5.20	5.60	4.60	5.50	7.05	5.50	6.60	4.55	4.65	5.75

[1] 表中所汇报的实验结果均是每种剪枝方法使用标准 5 折交叉验证的实验在各个数据集的测试集上的平均准确度 (%) 与其相应的标准差。粗体字表示更高的准确度和更低的标准差。

[2] 通过在 5% 显著水平下的双尾成对 t 检验对比各算法之间是否有明显差异，其中 ‡ 和 † 分别表示 *DOMEP* 显著优于或者是显著差于对比算法的性能；若无符号标记，则表示 *DOMEP* 与相应对比算法之间没有显著差异。

[3] 表中最后两行分别展示了检验结果和各算法的平均序值。各算法的平均序值由 Friedman 检验 [113] 计算而得，而 “W/T/L” 则表示 *DOMEP* 显著优于、并无显著不同或者显著差于相应的对比算法的次数。

表 4.3　在自助法所构造的集成器上对比基准算法和 *COMEP* 及 *DOMEP*

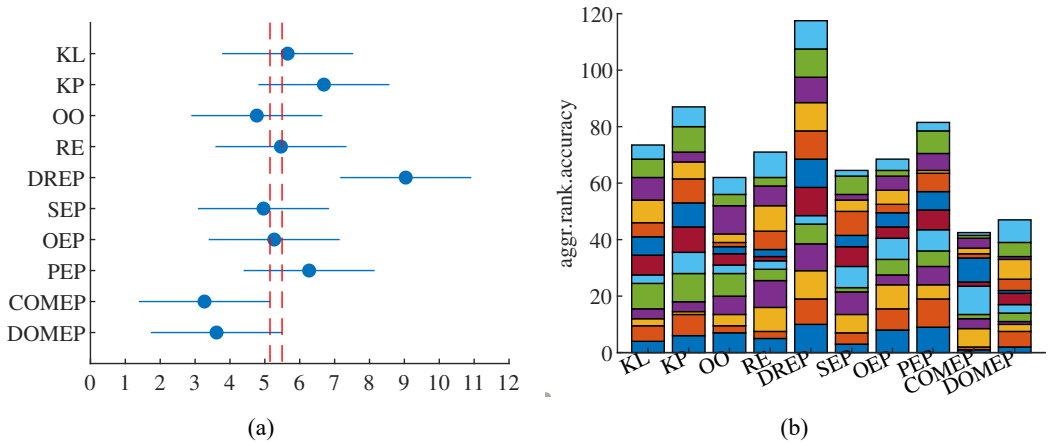
注：其中使用 k-近邻分类器作为基分类器。第二列 “Ensem” 代表未经剪枝的原始集成器的准确度。对于每一个数据集 (即每一行) 来说，更高的准确度和更低的标准差将用粗体标出。

Dataset	KL	KP	OO	RE	DREP	SEP	OEP	PEP	*COMEP*	*DOMEP*
Ames	58.83±5.48	58.54±7.20	**60.29±7.45**	57.81±7.14	57.52±7.29	59.71±6.18	59.56±6.96	58.69±5.90	60.15±6.90	59.12±5.95
Card	67.59±2.10	66.86±1.68	66.72±1.68	67.01±2.27	64.38±2.27	67.88±2.87	65.99±1.22	65.55±2.39‡	**68.32±3.33**	68.03±3.44
Sonar	80.98±7.98	80.00±8.52	81.95±7.03	81.95±5.62	79.02±5.62	81.46±5.87	80.00±5.29	78.05±5.72	**82.44±7.19**	80.98±8.69
Page	95.87±0.35	95.87±0.49	95.81±0.45	95.76±0.35	95.76±0.39	95.80±0.57	95.87±0.55	95.81±0.46	95.87±0.64	**95.89±0.64**
Wilt	97.93±0.28	97.89±0.32	**98.01±0.24**	97.99±0.20	97.77±0.36	97.93±0.35	97.89±0.40	98.01±0.32	97.97±0.19	98.01±0.26
Landsat	90.28±0.88	90.22±1.18	90.30±0.95	90.40±0.83	90.31±0.94	**90.47±0.78**	90.37±0.85	90.37±0.76	90.47±1.21	90.44±0.87
Shuttle	99.82±0.03	99.81±0.03	99.82±0.04	**99.82±0.02**	99.82±0.04	99.81±0.04	99.81±0.03	99.81±0.03	99.80±0.03‡	99.82±0.04
Ecoli	95.15±2.91	94.55±3.14	**95.76±2.91**	95.45±3.03	93.33±3.14	94.85±2.95	94.24±3.11	94.85±3.65	95.45±3.03	**95.76±2.91**
WaveformNoise	84.34±1.05	84.20±1.46	84.48±2.09	84.36±2.24	83.54±0.74	84.36±1.33	84.48±2.24	84.16±1.58	84.48±1.81	**84.62±1.77**
EEGEyeState	95.99±0.55	96.06±0.52	96.16±0.50	95.98±0.59	95.03±0.37‡	96.11±0.50	96.08±0.66	**96.28±0.55**	96.19±0.62	96.01±0.46
SensorReadings	96.29±0.47	96.11±0.39	96.33±0.62	96.37±0.53	96.00±0.51	96.29±0.64	96.39±0.42	96.28±0.60	**96.42±0.64**	96.31±0.31
t-Test (W/T/L)	0/11/0	0/11/0	0/11/0	0/11/0	1/10/0	0/11/0	0/11/0	1/10/0	1/10/0	—
Average Rank	5.82	7.45	3.77	5.27	8.95	5.09	5.77	6.55	3.05	3.27

[1] 表中所汇报的实验结果均是每种剪枝方法使用标准 5 折交叉验证的实验在各个数据集的测试集上的平均准确度 (%) 与其相应的标准差。粗体字表示更高的准确度和更低的标准差。

[2] 通过在 5% 显著水平下的双尾成对 t 检验对比各算法之间是否有明显差异，其中 ‡ 和 † 分别表示 *DOMEP* 显著优于或者是显著差于对比算法的性能；若无符号标记，则表示 *DOMEP* 与相应对比算法之间没有显著差异。

[3] 表中最后两行分别展示了检验结果和各算法的平均序值。各算法的平均序值由 Friedman 检验 [113] 计算而得，而 “W/T/L” 则表示 *DOMEP* 显著优于、并无显著不同或者显著差于相应的对比算法的次数。



(a)　　(b)

图 4.2　基准剪枝算法和 *COMEP*、*DOMEP* 在测试准确度上的对比

注：(a) Friedman 检验图 (两种方法间若没有重叠则代表着有显著差异) [113]。(b) 每种剪枝方法根据测试准确度排序的总计序值图 (序值越小，则该方法越优) [49]。

相应的标准差。在表 4.1 中的每一行所对比的都是相应方法的分类准确度，且这些方法使用的都是同样类型的基分类器所构成的集成器。对于每个数据集 (表中每一行)，含有更高的准确度和更低的标准差的实验结果将以粗体形式标出。此外，通过利用显著水平为 5% 的双尾成对 t 检验，我们还验证了这些方法两两之间在准确度上是否具有显著的不同。若两种方法之间并无显著不同，则认为它们达成平局；否则，含有更高准确度的方法将获胜。在表 4.1 中的最后两行，我们根据 Friedman 检验 [113] 计算了每种方法的平均序值，且这些方法都是用于和 *DOMEP* 对比的，并汇报了 *DOMEP* 获胜、平局或认输的数据集的个数。可以推断出：尽管 *DOMEP* 只利用了局部特征，但是在许多数据集上 *DOMEP* 仍然获得了不弱于集中式方法的效果。这也确认了 *DOMEP* (和 *COMEP*) 算法中同时利用准确度和多样性的合理性。尽管在某些情形下，*DOMEP* 所获得的测试准确度可能会有些微的劣势，但是整体而言仍能保持在可以被接受的水平线上。类似的结果也出现在表 4.2–4.3 中，只是使用了不同类型的基分类器。不仅如此，图 4.2 也从统计角度 [49, 113] 展示了基准算法和 *COMEP* 及 *DOMEP* 在测试准确度上的对比。通过 Friedman 检验 [113]，图 4.2(a) 展示了 *COMEP* 和 *DOMEP* 显著优越于其他集中式的对比算法；而图 4.2(b) 展示了每种算法的总计排序，可以得出与图 4.2(a) 相似的结论。

4.4.2　对比 *DOMEP* 和 *COMEP* 的时间代价

本小节给出了所提出算法的复杂度分析，并展示了相应的实验结果。根据算法 4.1，*COMEP* 的计算复杂度可分析如下：

- 首先，一个基本项 $\mathrm{TDAC}(h_i, h_j)$ 的时间复杂度为 $\mathcal{O}(d^2 n_c^2)$，其中 d 和 n_c 分

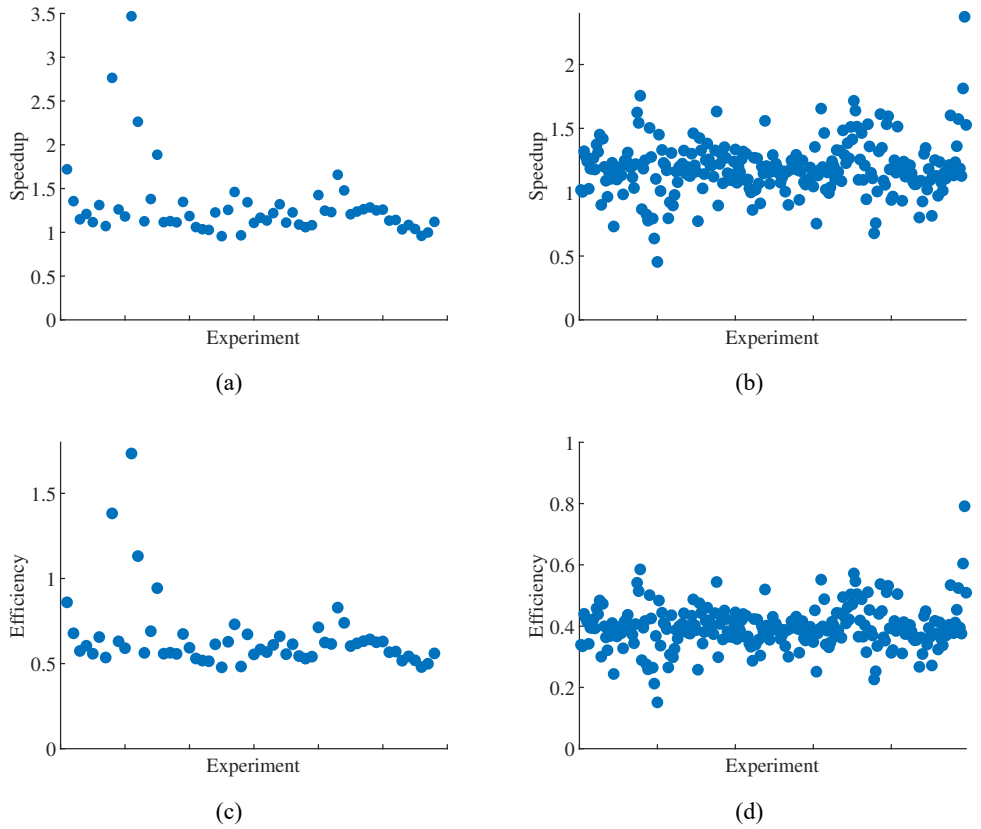


图 4.3　算法 *COMEP* 与 *DOMEP* 的加速比和效率对比

注：(a) 使用两台机器的加速比。(b) 使用三台机器的加速比。(c) 使用两台机器的效率。(d) 使用三台机器的效率。

别代表样本数量和标签数量。

- 其次，获得被选出的基分类器的时间复杂度是 $\mathrm{TDAC}(h_i, h_j)$ 项的 $\mathcal{O}(-\frac{1}{3}k^3+\frac{1}{2}nk^2)$ 倍，因为 $\sum_{i=2}^{k}(i-1)(n-i+2) = -\frac{1}{3}k^3+\frac{n+2}{2}k^2-\frac{3n+4}{6}k$，其中 k 是剪枝后子集成器的规模大小。

因此，算法 *COMEP* 的整体计算复杂度为 $\mathcal{O}\big((-\frac{1}{3}k^3+\frac{1}{2}nk^2)d^2n_c^2\big)$. 此外，在算法 *DOMEP* 中，由于原始集成器 $\mathcal{H}$ 被划分成了 m 组，则 *DOMEP* 的整体时间复杂度为 $\mathcal{O}\big((-\frac{1}{3}k^3+\frac{n}{2m}k^2)d^2n_c^2\big)$.

在实验中，我们在算法 *DOMEP* 中使用了不同数量的机器数，这是为了对比它与算法 *COMEP* 的加速比。Zadeh et al. [123] 指出，在理想情况下，算法的分布式版本与集中式版本甚至可以达到根据所使用的机器数的线性加速比，因为在这些机器之间的信息交换是没有时间开销的。为了验证算法 *DOMEP* 是否能够获得比 *COMEP* 更加令人满意的加速性能，我们从并行计算领域引入了两个性能指标，即加速比和效率[①]，其中加速比被定义为算法 *COMEP* 的执行时间与算法 *DOMEP* 的执行时间的商。受限于我们所使用机器的算力，在两到三台机器上执

[①] 效率是用来度量并行计算被用来解决一个特定问题时的效果的。若一个并行算法的渐近运行时间与计

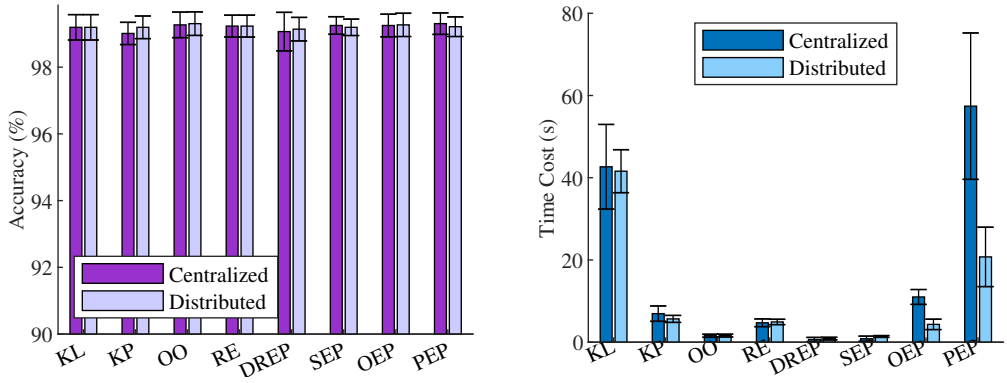


图 4.4　现有剪枝方法与其分布式版本的实验结果对比

注：在 SensorReadings 数据集上使用决策树作为基分类器和自助法来解决二分类问题。(a) 测试准确度。(b) 时间代价。

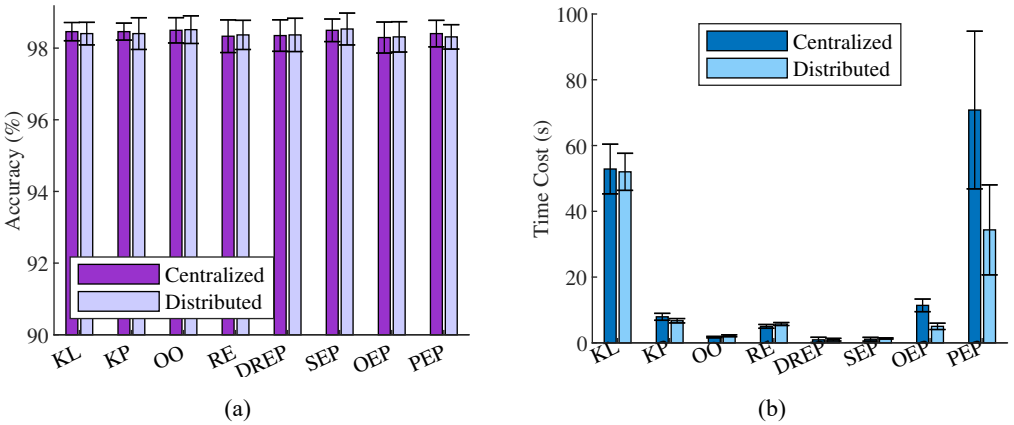


图 4.5　现有剪枝方法与其分布式版本的实验结果对比

注：在 SensorReadings 数据集上使用 k-近邻分类器作为基分类器和自助法来解决二分类问题。(a) 测试准确度。(b) 时间代价。

行剪枝算法作为典型实例，用以展示算法 *DOMEP* 的性能和效果。这些实验结果已展示在图 4.3 中，可以看出：算法 *DOMEP* 比 *COMEP* 执行得更快，在极少数情形下甚至能达到超线性加速比[①]。

4.4.3　评估可通用的分布式集成剪枝框架

在本小节中，为了验证 *EPFD* 的有效性，我们对比了大量现有剪枝方法和它们由 *EPFD* 所生成的相应分布式版本的算法性能，即子集成器的准确度和时

算中所使用的处理单元数目的乘积可以与最佳序列算法的运行时间持平，则认为该并行算法是代价有效的 [131]。

[①]在某些情况下，使用 m 个处理单元时的执行时间甚至能达到大于 m 的加速比，这种情况被称作是超线性加速比 [131]。

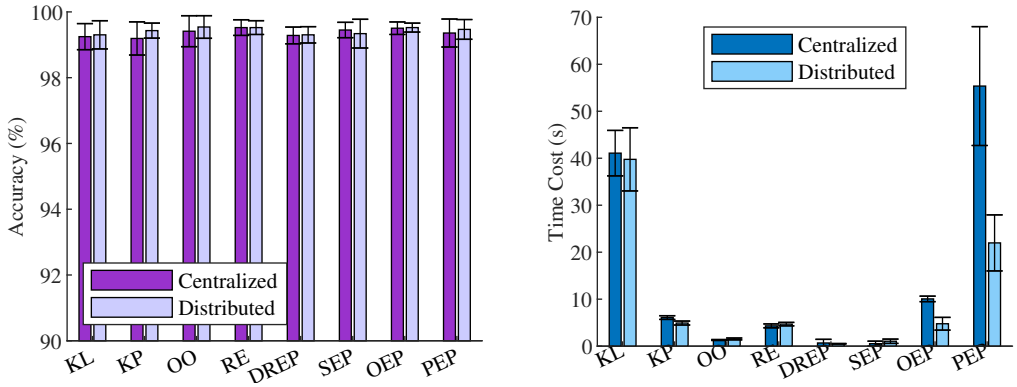


图 4.6　现有剪枝方法与其分布式版本的实验结果对比

注：在 SensorReadings 数据集上使用决策树作为基分类器和自助法来解决多分类问题。(a) 测试准确度。(b) 时间代价。

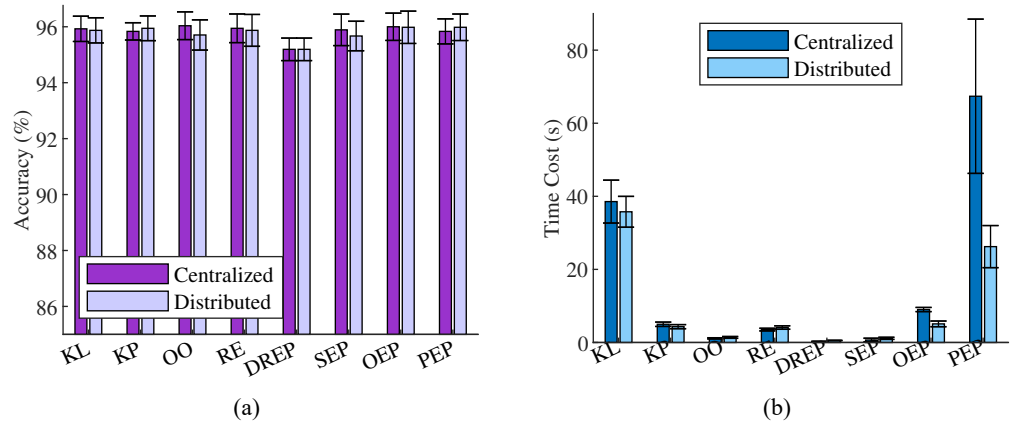
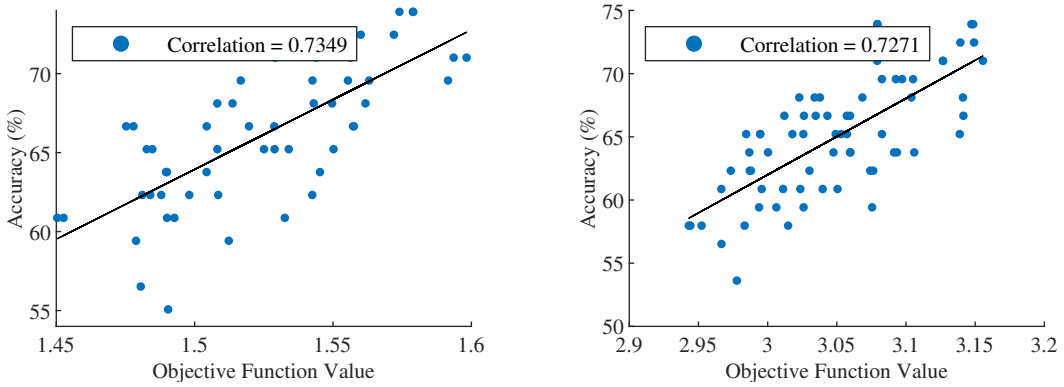


图 4.7　现有剪枝方法与其分布式版本的实验结果对比

注：在 SensorReadings 数据集上使用 k-近邻分类器作为基分类器和自助法来解决多分类问题。(a) 测试准确度。(b) 时间代价。

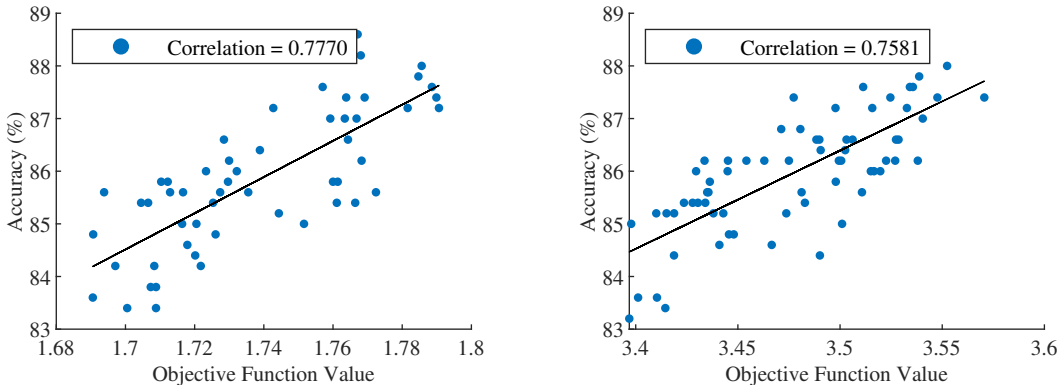
间代价。为了评估每种剪枝方法所生成的子集成器的质量，我们控制了实验变量 (包括所使用的集成方法和基分类器的种类)，即在同一实验中，只改变所使用的剪枝方法，而维持其他实验条件不变。图 4.4 展示了在二分类问题中使用决策树作为基分类器和自助法来构建集成器的对比结果。可以推知，对于每种剪枝方法 (横轴上的每一组) 来说，其分布式版本的准确度可达到与原集中式版本相等甚至更优的水平。考虑到分布式版本所消耗的时间代价更小这一点，我们可以认为每种剪枝方法的分布式版本是优于其原集中式版本的性能的。除此之外，从图中我们还可以看出 *EPFD* 在剪枝算法 PEP 上的提高尤为明显，这是一种利用演化帕累托优化和局部搜索策略的复杂方法，由 *EPFD* 所生成的 PEP 分布式版本的算法能够在更短的时间内获得同等准确度水平的子集成器。不仅如此，图 4.5 展示了二分类问题中使用 k-近邻分类器作为基分类器的实验结果，图 4.6–4.7 则



(a)　(b)

图 4.8　二分类问题中集成器的准确度和目标函数值之间的关系

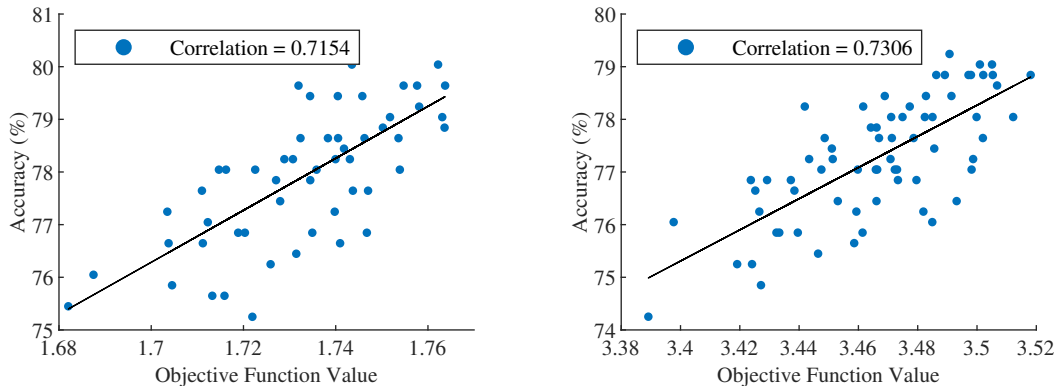
注：在 Ames 数据集上使用决策树作为基分类器和自助法来构造集成器。(a) 基分类器之间的 3-组合。(b) 基分类器之间的 4-组合。



(a)　(b)

图 4.9　二分类问题中集成器的准确度和目标函数值之间的关系

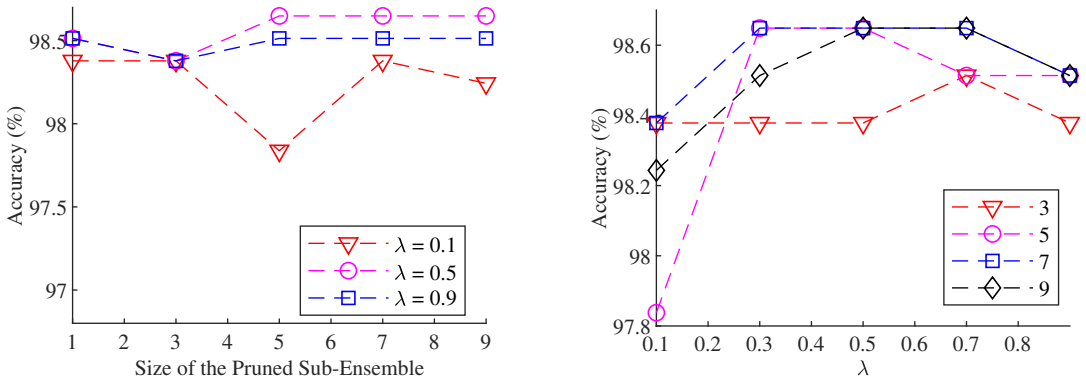
注：在 Waveform 数据集上使用决策树作为基分类器和自助法来构造集成器。(a) 基分类器之间的 3-组合。(b) 基分类器之间的 4-组合。



(a)　(b)

图 4.10　多分类问题中集成器的准确度和目标函数值之间的关系

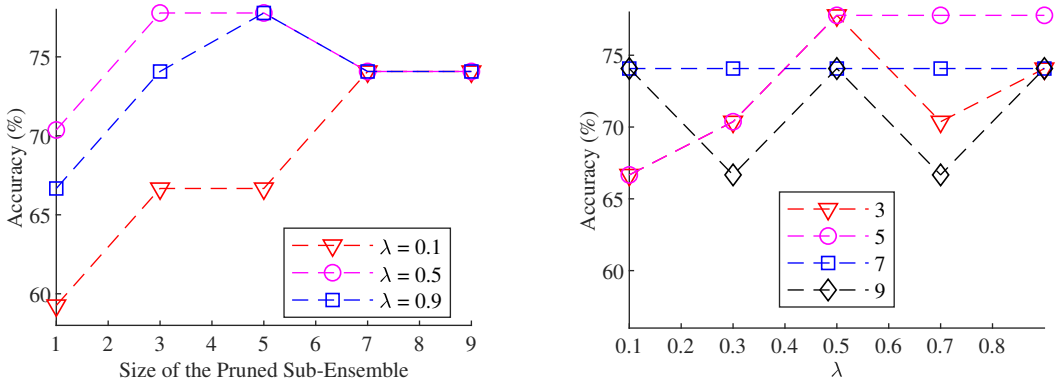
注：在 Waveform 数据集上使用决策树作为基分类器和自助法来构造集成器。(a) 基分类器之间的 3-组合。(b) 基分类器之间的 4-组合。



(a)　(b)

图 4.11　二分类问题中不同 λ 值对准确度的影响

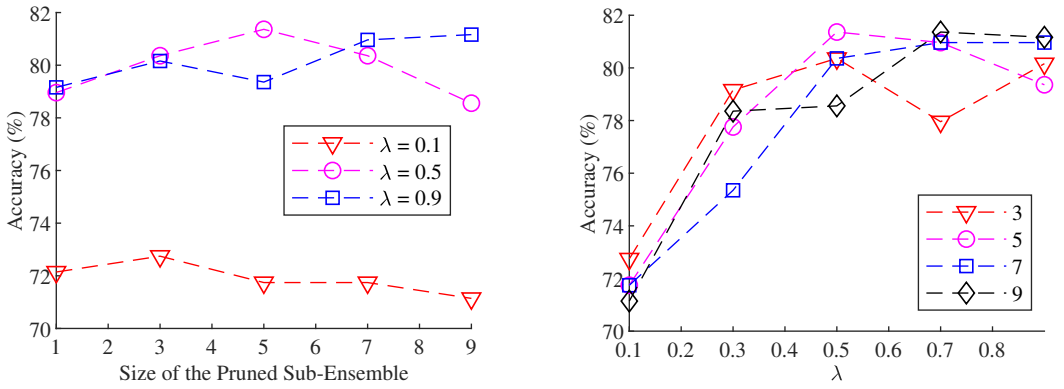
注：在 Ringnorm 数据集上使用支持向量机作为基分类器和自助法来构建集成器。(a) 分别使用每个准则的准确度。(b) 当选择不同规模的子集成器 (即由 3, 5, 7, 或 9 个基分类器组成) 时，不同 λ 值的些微不同。



(a)　(b)

图 4.12　二分类问题中不同 λ 值对准确度的影响

注：在 Heart 数据集上使用 k-近邻分类器作为基分类器和自助法来构建集成器。(a–b) 同图 4.11。



(a)　(b)

图 4.13　多分类问题中不同 λ 值对准确度的影响

注：在 Waveform 数据集上使用决策树作为基分类器和自助法来构建集成器。(a–b) 同图 4.11。

展示了多分类问题中分别使用决策树和 k-近邻分类器作为基分类器的实验结果，同样可以得出类似结论。

4.4.4　评估优化目标

对于一个新的目标函数来说，它与分类准确度之间的关系是一个非常重要的问题。为了验证这一问题，我们选择两个小规模的集成器 (即基分类器的数量较少)，并评估这些基分类器之间所有可能的组合情形。在这一实验中，我们比较了原始集成器中基分类器的所有可能的 3-组合或 4-组合的测试分类准确度和其相应的目标函数值，其中 λ 参数取值为 0.5, 这意味着式 (4.4) 中的两个准则在优化目标中同等重要。图 4.8 中蓝色的点表示在一个规模为 8 的集成器中的基分类器的 3-组合或 4-组合在 Ames 数据集上学习二分类问题的分类准确度，而黑色的实线则表示回归线。图 4.9–4.10 分别展示了在 Waveform 数据集上学习二分类问题和多分类问题的实验结果，可得到与之相似的结论。我们可以从图 4.8–4.10 中观察到目标函数值与分类准确度之间具有高度相关性，这也意味着最大化目标函数可以指引我们达成目标，即获得具有高准确度的子集成器。

4.4.5　参数 λ 值的影响

另一个关键的问题是在所定义的优化目标函数中，这两个准则之间的关系究竟如何。为了探究分类结果所受到的正则因子 λ 的影响这一问题，本节中的实验使用了不同的 λ 值 (从 0.1 到 0.9 中变化，其步长为 0.2)。图 4.11 展示了不同 λ 值在 Ringnorm 数据集上的实验结果图。图 4.11(a) 说明了将式 (4.7) 中的这两个准则同时纳入考虑的算法性能优于只重点关注于其中的某一项，如 MI 项 ($\lambda = 0.1$) 或 VI 项 ($\lambda = 0.9$)，尽管对于不同的数据集来说，寻找 λ 的最优取值可能就是另一个挑战了。图 4.11(b) 展示了不管剪枝后的子集成器规模如何，正则因子 λ 的一个全局最优值的确存在，也暗示着该最优值或许与其所使用的数据集本身所具有的性质有关。图 4.12–4.13 分别展示了在二分类和多分类问题中的实验结果，从中观测也可得出类似的结论。尽管为方便计，将 λ 值设置为 0.5 也可获得令人满意的实验结果 (如表 4.1–4.3 中所示)；但若针对不同的数据集调整 λ 的取值，*DOMEP* 或可达到更优的算法性能。

4.5　本章小结

本章从信息熵角度将多样性和准确度同时纳入考虑范围，从而将集成剪枝问题形式化为一个目标最大化的优化问题。随后根据这一优化目标，提出了一种

基于目标最大化的集成剪枝算法 (包括两个版本，即集中式的 *COMEP* 和分布式的 *DOMEP*)，通过对比所提出算法与现有剪枝方法的性能，可以发现所提出的算法能够快速有效地处理大规模的数据集，这也说明了该算法对于多样性和准确度的平衡是合理的。此外，本章还提出了一个可通用的分布式集成剪枝框架 (*EPFD*)，它能被广泛用于现有的各种剪枝方法，并在耗时更短的同时获得等准确度水平的子集成器，说明了 *EPFD* 的显著有效性对于现实世界的海量数据而言具有积极意义。在未来的工作中，探索这一工作的理论基础并尝试其他的优化目标以期获得更优的算法性能是个值得研究的课题。

第 5 章　神经架构搜索中的集成子结构剪枝

本章研究集成学习和多样性在神经架构搜索中的应用。近年来，神经架构搜索 (Neural Architecture Search, NAS) 因其显著的可扩展性和改善深度神经网络负担的能力而愈受重视。然而为获得更优的网络性能，搜索过程往往伴有极大的计算代价，这对于研究者和实践者而言或许是难以承受的。于是一些研究对集成方法在神经架构搜索领域中的应用进行探索，以缓和所需的计算代价。但是这些研究往往忽略了集成学习中的一个重要因素 (即多样性)，这可能造成更多相似的子网络结构被加入到最终的网络结构中去，从而带来一定程度上的冗余。为了弥合这一点，本章提出了一个针对于 NAS 集成网络结构的剪枝方法，命名为“神经架构搜索中的集成子结构剪枝 (Sub-Architecture Ensemble Pruning in Neural Architecture Search, *SAEP*)”，旨在利用多样性这一要素来获得相较原集成网络结构而言性能持平而规模更小的子集成网络结构。在这一搜索过程中，共有三种可供选择的剪枝策略被提出，用于决定何种子网络结构将被剪枝。最后，本章在实际数据集上的实验结果验证了所提出算法的有效性，它的确能够在维持原集成网络结构性能的基础上有效缩减集成网络结构的规模。不止于此，当搜索过程中的多样性不够充分时，*SAEP* 或可发现一些独特的更深层次的子网络结构。

5.1　背景介绍

神经网络结构的设计通常需要精巧的结构工程、广泛的专家经验和昂贵的计算代价。故旨在缓和这些挑战的神经架构搜索 (Neural Architecture Search, NAS) 正逐渐吸引了越来越多研究者的注意 [132-134]。然而为获得一个令人满意的神经网络结构，神经架构搜索方法通常需要消耗极大的计算代价，而这在基础设置的布置中是极其昂贵的，对于研究者来说也或许是难以承担的 [135]。

最近一些研究工作 [78, 136-137] 尝试将集成方法应用到神经架构搜索中，通过联合多个计算代价更小的弱网络结构来构造一个强大的集成网络结构，用以缓和巨大的计算代价。AdaNet 作为其中的一个杰出范例，提出了神经网络结构与其参数的学习理论分析，并首次展示了对神经网络结构性学习的泛化边界 [77-78, 138]。但是这些工作在搜索新的子网络结构时都忽视了集成学习中的一个重要因素 (即多样性)，而这一要素恰恰对于创建更好的集成模型是非常有利的 [121-122, 139]。此外，许多集成剪枝方法也利用了多样性这一特性来生成相比原始集成器而言规模更小的子集成器 [140-141]。已有研究证明，相比于使用所有的基学习器所构成的原始集成器来说，少数多样的基学习器甚至可以构建出一

个性能更强大的子集成器 [117, 142]。这激励了我们去探索神经架构搜索中的集成子结构剪枝问题，其目标在于生成多样的子神经网络，从而构建一个规模更小而性能仍然强大的集成网络结构。

但是从不同的子网络结构中描述它们的多样性特性并决定哪些子网络结构将被剪枝是一项具有挑战性的工作。首先，在集成学习领域中存在着多样性的众多定义或度量方法 [142]。与模型准确度这一明确概念不同，目前多样性尚无一个广为研究者所接受的正式定义 [143]。其次，当组成集成器的基学习器的准确度得以提高时，它们之间的多样性往往会有所下降 [30]。而联合一些多样而准确度略有不足的基学习器有时比仅联合准确度高的基学习器效果更优，因为多样性比纯粹的准确度更加重要。再次，从一个集成架构中选择出子结构的最佳联合并非易事，因为它是一个具有幂指数级计算复杂度的 NP 完全难问题 [48, 71]。因此，如何恰当地处理准确度与多样性之间的平衡并选择出集成架构中的最优子集是神经架构搜索里集成子结构剪枝问题中需要考虑的一个重要问题。

为了处理神经架构搜索中的集成子结构剪枝问题，我们希望获得的是一个多样的且规模更小的子集成网络结构，同时该子集成网络仍能维持剪枝前集成网络的性能。研究思想是在搜索过程中同时对集成网络进行剪枝，只保留一些更有价值的子网络结构用于构建最终的集成网络结构。我们的神经架构搜索的集成子结构剪枝方法被命名为“神经架构搜索中的集成子结构剪枝 *(Sub-Architecture Ensemble Pruning in Neural Architecture Search*, *SAEP)*”，它受启发于 AdaNet [78]，可以根据三种不同的策略来决定何种子网络将会被剪枝。此外，当搜索过程中子网络结构之间的多样性不够充足时，*SAEP* 可以生成一些独特的更深层次的网络结构，这或许是剪枝所带来的意外收获。

概括起来，本章的主要贡献可分为三点阐述如下：

- 本章提出了一个神经架构搜索的集成子结构剪枝方法，用于寻求规模更小的子集成网络结构，它受益于集成学习中的一个重要因素 (即多样性)，并能获得与未剪枝的原始集成网络结构相差无几的准确度性能。
- 此外，当搜索过程中所获得的子网络结构之间的多样性不够充分时，本章所提出的方法还能够生成一些比之未剪枝的原始集成网络结构而言独特的更深层次的子网络结构。
- 实验结果验证了本章所提出算法的有效性，即能够在维持原始集成网络结构性能的基础上提高多样性并缩减其规模。

本章的其余部分安排如下：首先在第 5.2 节简要介绍一些相关工作；接着在第 5.3 节详细介绍所提出的方法，包括问题的定义、神经架构搜索中的集成子结构剪枝方法以及三种不同的剪枝策略；然后在第 5.4 节中对所提出的算法在实际

数据集上进行评估，并与基准算法进行比较；最后在第 5.5 节中对本章工作进行总结，并描述了未来工作的一些研究方向。

5.2　相关工作

本节将对神经架构搜索进行简单介绍。“神经架构搜索” 这一概念始于 Zoph et al. [135] 在 2017 年所提出的工作，该研究将神经架构搜索作为一个用于寻找最优网络结构的基于梯度的方法进行介绍。其中，由一个循环神经网络所表示的 “控制器” 用于生成可变长度的字符串，能够表达神经网络的结构和连通性；而由字符串所表示的 “生成子网络” 将在实际数据集上做训练，其得到的准确度可用作反馈信号，用于生成一个性能更优的神经网络结构 [132, 135, 144]。现有的神经架构搜索方法可通过三个维度来区分，即搜索空间、搜索策略和性能估计策略 [133, 145-147]。经典的神经架构搜索方法生成的是链式结构的神经网络 [133, 148]，这忽略了一些更现代的网络结构，如来自于残差神经网络的跳跃连接 [149]。因此一些研究也试图通过引入这些现代的网络结构来构建一些复杂的多分支的神经网络结构，以期获得性能更优的神经网络结构 [150-157]。

最近一些与集成学习相结合的神经架构搜索方法逐渐吸引了研究者的注意。Cortes et al. [78] 提出了一个数据依赖的学习保证，用于指导额外的子神经网络的选择，并展示了一个可适应性地学习神经网络的算法，即 AdaNet。他们声称 AdaNet 可以精准处理神经架构搜索方法中的一些关于数据、时间和资源等方面高昂代价的问题，因为 AdaNet 所处理的优化问题是凸性的，可确保一个全局最优解的存在。与之不同的是，Huang et al. [136] 将子神经网络结构特例化为残差神经块，并声明所提出的 BoostResNet 能够对多分支的特征表达有所促进。而 Macko et al. [137] 则提出了另一种名为 AdaNAS 的尝试，能够自动化地利用集成方法来构建神经网络，它是 AdaNet 的一个扩展，相异之处在于该方法使用了层叠的神经架构搜索块 (stacked NASNet) [132, 135]。但是在这些方法中，所有生成的子神经网络结构都将用于构建最终的神经网络结构，却忽视了集成方法成功的一大关键原因，即集成模型可受益于多样化的基学习器。

不仅如此，Chang et al. [158] 提出了带有集成 Gumbel-Softmax 的可微分的神经架构搜索 (Differentiable ARchiTecture Search with Ensemble Gumbel-Softmax, DARTS-EGS)，并提出了集成 Gumbel-Softmax 方法，用于维持搜索过程中的效率。Ardywibowo et al. [159] 构建了一个集成模型，能够在他们的神经架构分布搜索中表现出分布之外的探测，该方法的搜索目标是网络结构的分布，而非像一个标准的神经架构搜索方法一样，搜索目标是一个性能最优的神经网络。这两种方法并非在搜索过程中对子神经网络结构进行集成，因此未在本章中多做讨论。

表 5.1　本章中所使用到的符号及其表示含义

符号	定义
$[n]$	简记 $\{1, ..., n\}$
$\boldsymbol{x} \in \mathcal{X}$	神经网络的输入
$f(\cdot) \in \mathcal{F}$	一个 l 层神经网络所表示的函数
n_s	在第 s^{th} 层的神经元的数量
$h_{k,j}(\cdot)$	在第 k^{th} 层 ($k \in [l]$) 的一个神经元所表示的函数
$\mathbf{u}_s \in \mathbb{R}^{n_s}$	第 k^{th} 层的神经元的权重向量
$\mathbf{h}_k(\cdot)$	第 k^{th} 层的所有神经元所表示的函数向量
$\mathbf{w}_k \in \mathbb{R}^{n_k}$	在最终的网络结构中，第 k^{th} 层的函数所对应的权重向量
$\|\mathbf{w}_k\|_p$	$\mathbf{w}_k$ 的 l_p-范数，其中 $p \geqslant 1$
$T \geqslant 1$	神经架构搜索过程中的迭代次数
Γ	基于 Rademacher 复杂度的一个特定复杂度约束

5.3　方法介绍

本节介绍“*神经架构搜索中的集成子结构剪枝 (Sub-Architecture Ensemble Pruning in Neural Architecture Search, SAEP)*”。该算法的主要思想是在搜索新的候选子网络的过程中，并非一味地把选择出的所有子网络都加入集成网络结构内，而是经过判断和剪枝后才确定究竟哪些子网络结构将会被保留在最终的集成网络结构中。在本节的其余部分，我们首先描述了神经架构搜索中的集成子结构剪枝的问题定义，然后介绍了所提出的神经架构搜索中的集成子结构剪枝方法，最后依次介绍了在集成子结构剪枝方法中所用到的三种剪枝策略。

5.3.1　问题定义

符号: 本章分别使用粗体斜体小写字母 (如 $\boldsymbol{x}$)、粗体小写字母 (如 $\mathbf{x}$) 和斜体小写字母 (如 x) 来表示张量、向量和标量；使用 $\mathbf{x}^{\mathsf{T}}$ 来表示向量的转置；使用粗体手写大写字母 (如 $\mathcal{X}$) 来表示数据/假设空间；使用 $\mathbf{P}$, $\mathbb{E}$, $\mathbb{R}$ 和 $\mathbb{I}$ 来分别表示随机变量的概率函数、随机变量的期望、实数空间和指示函数。

我们将所用到的符号及其含义总结在表 5.1 中。由于 AdaNet 是神经架构搜索领域中一个流行的集成搜索方法，故本节所使用的搜索空间及其符号与 AdaNet 相同，以便于形式化本章欲解决的问题和所提出的剪枝方法。值得一提的是，本节所提出的剪枝策略也可被泛化地用于其他集成神经架构搜索方法中去，这可以作为未来一个有益探索的方向。

令 f 表示一个由 AdaNet 方法搜得的 l 层神经网络 [78, 138]，其中每一层都

与之前的所有层相连。对于任意输入 $\boldsymbol{x} \in \mathcal{X}$, 其输出与所有中间层单元相连，即

$$f(\boldsymbol{x}) = \sum_{k=1}^{l} \mathbf{w}_k \cdot \mathbf{h}_k(\boldsymbol{x}) \,, \tag{5.1}$$

其中 $\sum_{k=1}^{l} \|\mathbf{w}_k\| = 1$ 且 $\mathbf{h}_k = [h_{k,1}, ..., h_{k,n_k}]^{\mathsf{T}}$. 令 $h_{k,j}$ 代表第 k^{th} 层的一个神经元所表示的函数，即

$$h_{k,j}(\boldsymbol{x}) = \sum_{s=0}^{k-1} \mathbf{u}_s \cdot \phi_s\big(\mathbf{h}_s(\boldsymbol{x})\big) \,,\ k \in [l] \,, \tag{5.2}$$

而 $h_0(\boldsymbol{x}) = \boldsymbol{x}$ 则表示第 0^{th} 层，即输入层。注意 $\phi_s\big(\mathbf{h}_s\big) = \big(\phi_s(h_{s,1}), ..., \phi_s(h_{s,n_s})\big)$, 假设其中的 ϕ_s 属于 1-Lipschitz 激活函数，如 ReLU[①]或 sigmoid[②]函数 [78]。若对于任意 $s < k-1$ 满足 $\mathbf{u}_s = 0$, 且对于 $k < l$ 满足 $\mathbf{w}_k = 0$, 则该集成网络结构 f 与标准的多层前馈神经网络一致 [78]。

为了探索其使用的搜索空间 $\mathcal{F}$, 令 $\mathcal{H}_k$ 表示第 k^{th} 层神经元所表示的函数族。令 $\widetilde{\mathcal{H}}_k \overset{\text{def}}{=} \mathcal{H}_k \cup (-\mathcal{H}_k)$ 表示 $\mathcal{H}_k$ 与其反射的并集，且 $\mathcal{H} \overset{\text{def}}{=} \cup_{k=1}^{l} \widetilde{\mathcal{H}}_k$ 表示函数族 $\widetilde{\mathcal{H}}_k$ 的并集。则搜索空间 $\mathcal{F}$ 与 $\mathcal{H}$ 的凸包一致，这也意味着集成方法的泛化边界可以用于分析在 $\mathcal{F}$ 空间上的学习 [78]。因此，Cortes et al. [78, 138] 尝试提出了一个基于 Rademacher 复杂度分析 [164] 的学习保证，用于指导所提出算法的设计。

受启发于集成方法，AdaNet 试图使用更少的计算代价训练多个弱网络结构来构成更强大的集成网络结构 [78]，而集成学习中的要素 “多样性” 可为获得一个由多样的子网络结构组成的规模更小而性能持平的子集成网络结构带来机遇。基于上述术语，我们正式定义神经架构搜索中集成结构剪枝问题如下。

定义 5.1 (神经架构搜索中的集成结构剪枝)　给定一个由神经架构搜索集成方法 (如 AdaNet) 所搜索得到的集成网络结构 $f(\boldsymbol{x}) = \sum_{k=1}^{l} \mathbf{w}_k \cdot \mathbf{h}_k(\boldsymbol{x}) \in \mathcal{F}$, 和一个基数为 m 的训练集 $S = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_m, y_m)\}$, 其中所有的训练样本都是从 $\mathcal{X} \times \{c_1, ..., c_{n_c}\}$ 之上的分布 $\mathcal{D}$ 中独立抽取所得，而 n_c 表示样本的标签数。则目标为对该集成神经网络结构 f 进行剪枝，以期获得一个规模更小的子集成网络结构，同时仍能维持与原有集成网络结构性能持平的准确度水平。

5.3.2　神经架构搜索中的集成子结构剪枝

本小节将详细阐述所提出的神经架构搜索中集成结构的剪枝方法，其目标在于获得一个规模更小然而仍然有效的集成网络结构。在对一些价值更小的子网络结构进行剪枝之前，我们首先需要生成一些候选的子网络结构。我们在这里利用 AdaNet 来生成候选的子网络结构，因为它是一种在神经架构搜索的集成方法研究中非常流行且性能优越的方法。换言之，在搜索过程中，我们利用了

[①] 整流线性单元 (Rectified Linear Unit, ReLU) [160-162] 被定义为 $g(z) = \max\{0, z\}$.

[②] Sigmoid 函数 [163] 被定义为 $\sigma(z) = \frac{1}{1+e^{-z}}$.

AdaNet 的优化目标函数来生成候选的子网络结构。这一用于生成新的候选子网络的优化目标函数被定义为

$$\mathcal{L}_g(\mathbf{w}) = \hat{R}_{S,\rho}(f) + \Gamma\,, \tag{5.3}$$

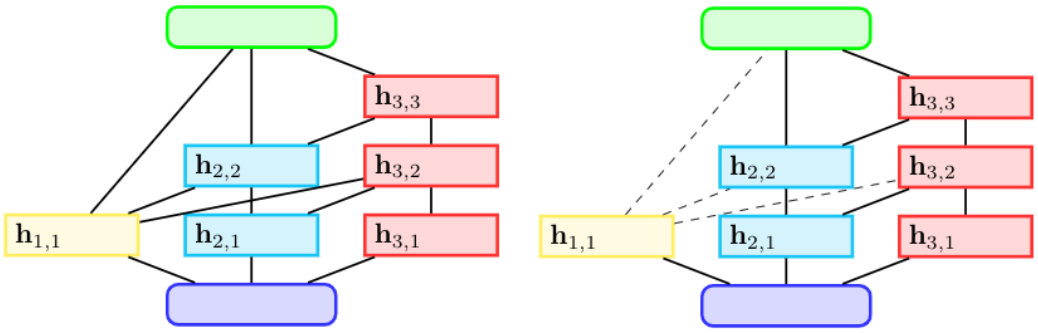
其中 $\hat{R}_{S,\rho}(f)$ 表示函数 f 在训练集 S 上的经验边界误差，而 Γ 则表示一个特定的复杂度约束。由于 AdaNet 中的学习保证是针对于二分类问题的，为了将该优化目标扩展到多分类问题中，我们引入了一个辅助函数 $g(\boldsymbol{x}, y, f)$，即

$$g(\boldsymbol{x}, y, f) = 2\mathbb{I}\big(f(\boldsymbol{x}) = y\big) - 1\,. \tag{5.4}$$

此时经验边界误差 $\hat{R}_{S,\rho}(f)$ 将变成

$$\hat{R}_{S,\rho}(f) = \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}\big(g(\boldsymbol{x}_i, y_i, f) \leqslant \rho\big)\,. \tag{5.5}$$

由式 (5.3) 可知，AdaNet 只通过最小化经验误差和架构复杂度来生成新的候选子网络，却忽视了这些不同的子网络结构之间的差异或多样性。为了获得规模更小但仍然有效的子集成网络结构，我们将不同子网络结构之间的多样性也纳入考量，随之获得一个相应的优化目标函数，用于指导我们在搜索过程中选择出一些更有价值的子网络结构。具体而言，我们提出了三种不同的策略用于增强不同的子网络结构之间的多样性。除了第一种剪枝策略以外，后两种都提供了一个特定的或包括多样性的优化目标用于指导剪枝过程。此外，这些生成子网络结构之间的多样性也可加以量化，来验证这些剪枝策略是否起到了预期中的效果。



(a)　　(b)

图 5.1　*SAEP* 与 AdaNet 在神经架构的增量构建过程中的差异说明

注：蓝色的和绿色的层分别表示输入层和输出层。黄色的、青色的和红色的神经元是分别在第一次迭代、第二次迭代和第三次迭代过程中被加入神经网络架构中去的。

(a) AdaNet [78]: 在两个神经元块之间的直线表示这些块之间是全连接的。

(b) *SAEP*: 只有一些价值较高的神经元块将会被保留在最终的神经网络架构中 (被剪掉的神经元块使用黑色虚线表示)，这也是该算法与 AdaNet 的关键性不同所在。本章提出了三种准则可用于在 *SAEP* 中决定哪些子网络结构将被剪枝，即 *PRS*、*PAP* 和 *PIE*。

Data: 数据集 $S=\{(\boldsymbol{x}_i,y_i)\}_{i=1}^m$, 迭代次数 T
Result: 最终的集成网络结构 f
1 令 $f^{(0)}=\mathbf{0}$, 且 $l^{(0)}=1$; // 初始化
2 **for** $1\leqslant t\leqslant T$ **do**
3 $\quad\mathbf{w}',\mathbf{h}'=\arg\min_{\mathbf{w},\mathbf{h}}\mathcal{L}_g(f^{(t-1)}+\mathbf{w}\cdot\mathbf{h})$ s.t. $\mathbf{h}\in\mathcal{H}_{l^{(t-1)}}$;
4 $\quad\mathbf{w}'',\mathbf{h}''=\arg\min_{\mathbf{w},\mathbf{h}}\mathcal{L}_g(f^{(t-1)}+\mathbf{w}\cdot\mathbf{h})$ s.t. $\mathbf{h}\in\mathcal{H}_{l^{(t-1)}+1}$;
5 $\quad$**if** $\mathcal{L}_g(f^{(t-1)}+\mathbf{w}'\cdot\mathbf{h}')\leqslant\mathcal{L}_g(f^{(t-1)}+\mathbf{w}''\cdot\mathbf{h}'')$ **then**
6 $\quad\quad f^{(t)}=f^{(t-1)}+\mathbf{w}'\cdot\mathbf{h}'$;
7 $\quad$**else**
8 $\quad\quad f^{(t)}=f^{(t-1)}+\mathbf{w}''\cdot\mathbf{h}''$;
9 $\quad$**end**
10 $\quad$根据某种特定准则选择出 $\mathbf{w}_p$, 如基于随机选择的剪枝、基于准确度性能的剪枝和式 (5.6) 中的 $\mathcal{L}_d(\mathbf{w})$、或基于信息熵的剪枝和式 (5.16) 中的 $\mathcal{L}_e(\mathbf{w})$;
11 $\quad$将所选出的 $\mathbf{w}_p$ 设为零;
12 **end**

算法 5.1: 神经架构搜索中的集成子结构剪枝 (*SAEP*)

我们将所提出的神经架构搜索中的集成网络结构的剪枝方法命名为 "神经架构搜索中的集成子结构剪枝 *(Sub-Architecture Ensemble Pruning in Neural Architecture Search, SAEP)*"，如算法 5.1 所示。又如图 5.1 所示，其与 AdaNet 的关键不同点在于：*SAEP* 将会在搜索过程中对一些价值相对更小的子网络结构进行剪枝 (如算法 5.1 中第 10–11 行所示)，而不是像 AdaNet 一样将所有的子网络结构都予以保留。在算法 5.1 的第 t^{th} 次迭代中，令 $f^{(t-1)}=\sum_{k=1}^{l}\mathbf{w}_k\cdot\mathbf{h}_k$ 表示在第 t^{th} 次迭代开始前已构建的神经网络架构，其层深度为 $l^{(t-1)}$. 第 t^{th} 阶段的目标之一是生成新的候选子网络 (如第 3–4 行所示)，并选择其中更好的一个加入模型 $f^{(t-1)}$ (如第 5–9 行所示)，这样做的原因是我们希望这个搜索过程稳步发展。第 t^{th} 阶段的目标之二是对模型中价值较小的子网络结构进行剪枝，只将保留的子网络结构用于构建最终的集成神经网络结构 (如第 10–11 行所示)。

我们提出了三种策略用以评估哪些子网络结构的价值较高，并在接下来的三个小节中具体介绍。

5.3.3　基于随机选择的剪枝 (*PRS*)

第一种剪枝策略被命名为 "基于随机选择的剪枝 *(Pruning by Random Selection, PRS)*"，其思想是在搜索过程中随机地选择若干个子网络结构进行剪枝。在 *PRS* 中，我们首先需要随机地决定在当前迭代过程中是否需要剪枝；若决定剪枝，再从中随机地选择一个子网络结构进行剪枝。*PRS* 与后两种剪枝策略的不同

点在于 *PRS* 没有一个明确而具体的优化目标。而这也是 *PRS* 的不足之处，因为一些较有价值的子网络结构也可能会被随机地剪掉。因此，我们需要寻找一些更明确的优化目标来指导这一剪枝过程。

5.3.4　基于准确度性能的剪枝 (*PAP*)

为了更好地度量和比较不同的子网络结构，我们基于它们的准确度性能提出了第二种剪枝策略。这种策略被命名为 "基于准确度性能的剪枝 *(Pruning by Accuracy Performance, PAP)*"。为了从原有集成网络结构中选择出更有价值的子网络结构，迭代过程中的第二优化目标可被定义为

$$\mathcal{L}_d(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}\left[g(\boldsymbol{x}_i, y_i, f) - g(\boldsymbol{x}_i, y_i, f - \mathbf{w}\cdot\mathbf{h})\right], \tag{5.6}$$

其中 $\mathbf{h}$ 是相应于权重 $\mathbf{w}$ 的子网络结构。该优化目标是通过最小化式 (5.6) 来寻找合适的 $\mathbf{w}$ 和 $\mathbf{h}$，若其损失值小于零，则对其进行剪枝。这么做的原因如下所述。

将所有的子网络结构都集成到一起的泛化误差可定义为

$$R(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[\mathbb{I}\big(g(\boldsymbol{x}, y, f) \leqslant 0\big)\right]. \tag{5.7}$$

若将其中的第 j^{th} 个子网络结构从集成模型中排除，则该剪枝后的子集成网络结构的泛化误差即为

$$R(\bar{f}_j) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\mathbb{I}(g(\boldsymbol{x}, y, f - \mathbf{w}_j\cdot\mathbf{h}_j) \leqslant 0)]. \tag{5.8}$$

那么若希望剪枝后的子集成网络结构的性能更优于剪枝前的原始集成网络结构，则须保证不等式 $R(f) - R(\bar{f}_j) \geqslant 0$ 成立，即

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[g(\boldsymbol{x}, y, f) - g(\boldsymbol{x}, y, f - \mathbf{w}_j\cdot\mathbf{h}_j)\right] \leqslant 0. \tag{5.9}$$

因此，当第 j^{th} 个子网络结构满足式 (5.9) 时，将其从原始集成网络结构中剪枝可以获得性能持平甚至更优的子集成网络结构。式 (5.9) 的深意是：若原始的集成网络结构出现了错误，对满足条件的第 j^{th} 个子网络结构进行剪枝可以提高所获得的子集成网络结构正确工作的概率；换而言之，我们希望对那些所犯错误严重到能影响最终集成网络结构的子网络结构进行剪枝，从而构成了这一剪枝策略的优化目标，即式 (5.6)。此时，该剪枝过程可以在不破坏原有的学习保证的前提下获取性能更优的子集成网络结构。

但是优化目标式 (5.6) 只考虑了不同子网络结构的准确度性能，却忽略了集成方法中的多样性特点。因此我们需要一个能平衡多样性和准确度的优化目标。

5.3.5　基于信息熵的剪枝 (*PIE*)

为了同时考虑子网络结构的准确度和多样性特性，我们从信息熵的角度提出了第三种剪枝策略，可命名为 "基于信息熵的剪枝 *(Pruning by Information*

Entropy, PIE)"。令 $\mathbf{y} = [y_1, ..., y_m]^\mathsf{T}$ 表示数据集 S 的标签向量，而 $\boldsymbol{w}_j = \mathbf{w}_j \cdot [\mathbf{h}_j(\boldsymbol{x}_1), ..., \mathbf{h}_j(\boldsymbol{x}_m)]^\mathsf{T}$ 则表示子网络结构 $\mathbf{w}_j$ 在该数据集上的分类结果，其中 $\mathbf{w}_j$ 可以是集成网络结构中的任意一个子网络结构。注意 $\mathrm{H}(\cdot)$ 和 $\mathrm{H}(\cdot,\cdot)$ 分别代表随机变量的熵函数和联合熵函数，即

$$H(\boldsymbol{w}_i) = -\sum_{w \in \boldsymbol{w}_i} p(w) \log p(w)\,, \tag{5.10}$$

$$H(\boldsymbol{w}_i, \mathbf{y}) = -\sum_{w \in \boldsymbol{w}_i} \sum_{y \in \mathbf{y}} p(w, y) \log p(w, y)\,. \tag{5.11}$$

为了描述该子网络结构和标签向量之间的相关性，我们使用标准共有信息 [123] 来反映该子网络结构的准确度，即

$$\begin{aligned} \mathrm{MI}(\boldsymbol{w}_i, \mathbf{y}) =& \frac{\mathrm{I}(\boldsymbol{w}_i; \mathbf{y})}{\sqrt{\mathrm{H}(\boldsymbol{w}_i)\mathrm{H}(\mathbf{y})}} \\ =& \frac{\sum_{w \in \boldsymbol{w}_i, y \in \mathbf{y}} p(w, y) \log \frac{p(w,y)}{p(w)p(y)}}{\sqrt{\sum_{w \in \boldsymbol{w}_i} p(w) \log p(w) \sum_{y \in \mathbf{y}} p(y) \log p(y)}}\,, \end{aligned} \tag{5.12}$$

其中

$$\begin{aligned} \mathrm{I}(\boldsymbol{w}_i; \mathbf{y}) =& \mathrm{H}(\boldsymbol{w}_i) - \mathrm{H}(\boldsymbol{w}_i | \mathbf{y}) \\ =& \sum_{w \in \boldsymbol{w}_i, y \in \mathbf{y}} p(w, y) \log \frac{p(w, y)}{p(w)p(y)}\,, \end{aligned} \tag{5.13}$$

代表共有信息 [125]。为了描述两个不同的子网络结构 ($\mathbf{w}_i$ 和 $\mathbf{w}_j$) 之间的冗余度，我们使用标准信息变异 [123] 来反映这两个子网络结构之间的多样性，即

$$\begin{aligned} \mathrm{VI}(\boldsymbol{w}_i, \boldsymbol{w}_j) =& 1 - \frac{\mathrm{I}(\boldsymbol{w}_i; \boldsymbol{w}_j)}{\mathrm{H}(\boldsymbol{w}_i, \boldsymbol{w}_j)} \\ =& 1 - \frac{\sum_{w \in \boldsymbol{w}_i, y \in \mathbf{y}} p(w, y) \log \frac{p(w,y)}{p(w)p(y)}}{-\sum_{w \in \boldsymbol{w}_i, y \in \mathbf{y}} p(w, y) \log p(w, y)}\,. \end{aligned} \tag{5.14}$$

为了妥善处理两个子网络结构之间准确度和多样性的平衡，优化目标可定义为

$$\mathcal{L}_p(\mathbf{w}_i, \mathbf{w}_j) = (1 - \alpha)\mathrm{VI}(\boldsymbol{w}_i, \boldsymbol{w}_j) + \alpha \frac{\mathrm{MI}(\boldsymbol{w}_i, \mathbf{y}) + \mathrm{MI}(\boldsymbol{w_j}, \mathbf{y})}{2}\,. \tag{5.15}$$

当这两个子网络结构相同时，其优化目标值为零。值得注意的是 α 作为一个正则因子被引入，用于平衡准确度和多样性这两个准则，也反映了二者之间的相对重要性程度。我们的目标是通过最小化式 (5.16) 中的 $\mathcal{L}_e(\mathbf{w})$ 来选择符合条件的 $\mathbf{w}$ 和其相应的 $\mathbf{h}$，并对其进行剪枝，即

$$\mathcal{L}_e(\mathbf{w}_i) = \sum_{\mathbf{w}_j \cdot \mathbf{h}_j \in f \setminus \{\mathbf{w}_i \cdot \mathbf{h}_i\}} \mathcal{L}_p(\mathbf{w}_i, \mathbf{w}_j)\,. \tag{5.16}$$

这一损失函数同时考虑了集成学习中的两个重要因子，即多样性和准确度，可用于指导搜索过程中的剪枝。

5.4　实验评估

本节评估所提出的方法 *SAEP* 的有效性。在本节中，我们试图回答四个主要问题：(1) *SAEP* 能否获得较集成网络结构而言规模更小且性能持平的子集成网络结构? (2) *SAEP* 能否生成较集成网络结构而言更多样的子网络结构? (3) 参数 α 在利用 *PIE* 策略进行剪枝时会否对所生成的子网络结构造成影响? 若会，则其影响是什么? (4) *PIE* 能否生成较原始集成网络结构而言更独特的子网络结构?

5.4.1　图像分类数据集

我们使用三个公开的图像分类数据集来进行实验。ImageNet [165] 数据集并未包括在内，因为我们使用的是一台 GPU (NVIDIA GTX 1080) 服务器，其显存仅有 12,206 MiB (约 11.1 GB)；而 *SAEP* 在搜索过程中需要将整个数据集导入显存，ImageNet 的数据量[①]远远超出了显存的空间限制，实验难以为继。

CIFAR-10 [166]: 含 $60,000$ 张尺寸为 $32 \times 32 \times 3$ 的彩色图片，共计 10 个类别，分别代表飞机、汽车、鸟、猫、鹿、狗、青蛙、马、船和卡车。其中每个类别包括 $6,000$ 张图片。整个数据集中包括 $50,000$ 个训练样本和 $10,000$ 个测试样本。

MNIST [167]: 含 $70,000$ 张尺寸为 28×28 的灰度图片，共计 10 个类别，分别代表不同的手写数字。其中包括 $60,000$ 个训练样本和 $10,000$ 个测试样本。所有的数字图片都经过尺寸正则化和居中预处理，形成一个固定尺寸的图像。

Fashion-MNIST [168]: 含 $70,000$ 张尺寸为 28×28 的灰度图片，共计 10 个类别，分别代表 T 恤/上衣、长裤、套头衫、连衣裙、大衣、凉鞋、衬衫、运动鞋、背包和靴子。整个数据集中包括 $60,000$ 个训练样本和 $10,000$ 个测试样本。

5.4.2　基准算法

为了验证算法 *SAEP* 的有效性，我们将三种所提出的剪枝策略 (即 *PRS*、*PAP* 与 *PIE*) 分别与 AdaNet [78] 进行对比。AdaNet 的标准形式通常被设置为使用均匀的平均权重，此外还存在一种使用混合权重的变体，记作 AdaNet.W [169]。相似地，*PRS.W*、*PAP.W* 和 *PIE.W* (即 *SAEP.W*) 分别表示使用混合权重的 *PRS*、*PAP.W* 和 *PIE.W*，亦即使用这三种剪枝策略时的变体。用于对比的基准算法包括 AdaNet 与其相应形式的变体。

5.4.3　实验设置

为了公平地验证所提出方法的竞争力，我们在实验中采用了控制变量法，即在同一实验中，所有的算法将使用同类型的子网络类型来构建集成网络结构。可

[①]以 ILSVRC 2012 为例，未解压的图片数据大小为：训练集 138 GB, 验证集 6.3 GB, 和测试集 13 GB.

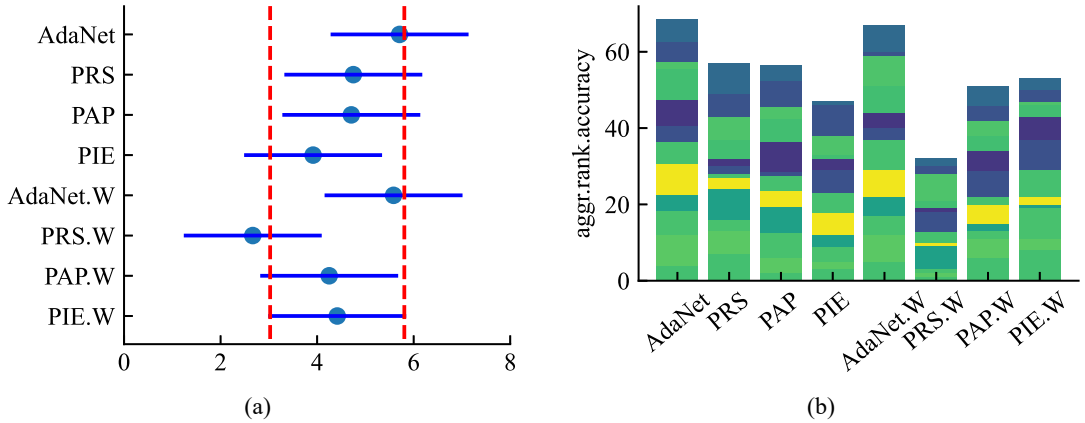


图 5.2　在二分类问题中对比基准算法和 *SAEP* 及其变体的测试准确度

注：实验中所使用的子网络类型是多层感知机。(a) Friedman 检验图 (两种方法间若没有重叠则代表着有显著差异) [113]，在 10% 显著水平下拒绝假设 “它们的测试准确度差异不显著”。(b) 每种方法根据测试准确度排序的总计序值图 (序值越小，则该方法越优) [49]。

供选则的子网络结构类型包括多层感知机 (Multi-Layer Perceptrons, MLPs) 和卷积神经网络 (Convolutional Neural Networks, CNNs)。注意所使用的卷积神经网络均由若干个包含 16 个通道的卷积层组成，不包括池化层。实验中所使用的超参数设置如下：学习率为 0.003，并在训练过程中使用动量优化器和余弦延迟来更新学习率；训练迭代次数设置为 5,000，而批量大小设置为 64.

我们使用上述数据集来进行图像分类任务。在多分类情形下将使用数据集中的所有类别；而在二分类情形下则只考虑若干标签对。举例来说，在 CIFAR-10 数据集中，考虑三个标签对，即 “鹿 v.s. 卡车”、“鹿 v.s. 马” 和 “汽车 v.s. 卡车” 标签对；在 Fashion-MNIST 数据集中，考虑五个标签对，即 “上衣 v.s. 套头衫”、“上衣 v.s. 大衣”、“上衣 v.s. 衬衫”、“长裤 v.s. 连衣裙” 和 “凉鞋 v.s. 靴子”；在 MNIST 数据集中，考虑两个标签对，即数字 “6 v.s. 9” 和 “5 v.s. 8”。

5.4.4　*SAEP* 可获得性能更优的集成网络结构

本小节验证的是剪枝后的子集成网络结构是否与原始集成网络结构性能持平甚至更优。实验结果汇报在表 5.2 和 5.5 中，包括每种方法在每个数据集上经标准 5 折交叉验证后的平均测试准确率 (%) 与其相应的标准差。如表 5.2 中的每一行 (标签对) 使用的都是同样类型的子网络结构，而粗体表示具有更高准确度和更低标准差的结果。当 AdaNet 与其中某种方法比较时，具有较高准确度值和较低标准差的方法将获胜；否则将通过双尾成对 t 检验来判断优胜者，即在 5% 显著性水平下判断这两种方法是否具有显著不同。具体来说，若二者之间无统计显著差异，则视为平局；否则具有更高准确度的方法将获胜。表 5.2 的最后两行

表 5.2　二分类问题中的 (子) 集成网络结构性能对比之测试准确度

注：表 5.2–5.4 应对照来看，每种方法包括三列，即测试准确度 *(%)*、搜索生成的 (子) 集成网络结构的规模大小和搜索过程的时间代价 *(分钟)*。每个数据集 (行) 中的最优实验结果将使用粗体字标出。注意实验中所使用的子网络类型是多层感知机。

Label Pair	**Test Accuracy (%)**							
	AdaNet	*PRS*	*PAP*	*PIE*	AdaNet.W	*PRS.W*	*PAP.W*	*PIE.W*
digits 6-9	99.85±0.07	99.83±0.05	99.85±0.07	99.85±0.09	99.84±0.05	**99.88±0.04**‡	99.84±0.14†	99.83±0.07
digits 5-8	99.14±0.13	99.18±0.21	99.19±0.18	99.21±0.10‡	99.16±0.18	**99.26±0.15**	99.19±0.18	99.20±0.15
top-pullover	97.03±0.47	97.06±0.15‡	97.03±0.39‡	97.04±0.34‡	97.04±0.30‡	**97.16±0.18**‡	97.08±0.21‡	96.94±0.17
top-coat	98.62±0.14	98.57±0.29†	98.61±0.07	98.64±0.32	98.61±0.22†	98.61±0.22†	98.66±0.23	**98.67±0.14**
top-shirt	85.87±0.77	86.36±0.82	86.33±0.77‡	86.18±0.80	86.14±0.54‡	**86.48±0.62**‡	86.27±0.69‡	86.36±0.54‡
trouser-dress	98.35±0.16	**98.42±0.16**	98.39±0.17	98.38±0.28	98.30±0.14	98.39±0.14‡	98.41±0.27	98.32±0.24†
sandal-ankle boot	98.74±0.19	98.78±0.27	**98.79±0.24**	98.71±0.11	98.78±0.14‡	98.71±0.10	98.69±0.19	98.69±0.31†
deer-truck	87.91±0.38	88.01±0.42	87.87±0.85†	87.99±0.67	87.95±0.35‡	**88.05±0.47**	87.93±0.40	87.91±0.49
deer-horse	75.54±1.45	76.22±1.17‡	76.22±1.07‡	**76.62±0.94**‡	76.10±1.28‡	76.31±1.15‡	76.22±0.41‡	76.25±0.46‡
automobile-truck	72.93±0.24	72.82±0.57†	72.91±0.29†	72.84±0.85†	72.58±0.94†	72.78±0.50†	72.90±0.81†	**72.95±1.10**
cat-dog	61.15±0.69	61.11±0.23	61.02±1.24†	60.67±1.05†	**61.63±0.81**	61.60±0.68‡	61.22±0.47‡	61.53±1.34
dog-horse	78.21±0.30	77.95±1.02†	78.29±0.61	**78.44±0.23**‡	78.20±0.81†	78.41±0.71	78.23±0.98	78.38±0.95
t-test (W/T/L)	—	3/7/2	3/6/3	2/6/4	3/4/5	2/4/6	2/6/4	2/8/2
Average Rank	5.71	4.75	4.71	3.92	5.58	2.67	4.25	4.42

[1] 表中所汇报的实验结果均是每种方法使用标准 5 折交叉验证法在各个数据集中测试集上的平均准确度 (%) 与其相应的标准差。

[2] 通过在 5% 显著水平下的双尾成对 t 检验对比各算法之间是否有明显差异，其中 ‡ 和 † 分别表示 AdaNet 显著弱于或优于 *SAEP* 及其变体的性能。

[3] 表中最后两行分别展示了检验结果和各算法的平均序值。其中 “W/T/L” 表示 AdaNet 显著优于、并无显著不同或者显著劣于相应的对比算法 (即 *SAEP* 及其变体) 的次数；各算法的平均序值由 Friedman 检验 [113] 计算而得。

表 5.3　二分类问题中的 (子) 集成网络结构性能对比之模型规模

注：表 5.2–5.4 应对照来看，每种方法包括三列，即测试准确度 *(%)*、搜索生成的 (子) 集成网络结构的规模大小和搜索过程的时间代价 *(分钟)*。每个数据集 (行) 中的最优实验结果将使用粗体字标出。注意实验中所使用的子网络类型是多层感知机。

Label Pair	Number of Sub-Architectures							
	AdaNet	*PRS*	*PAP*	*PIE*	AdaNet.W	*PRS.W*	*PAP.W*	*PIE.W*
digits 6-9	**4.20±1.47**	5.80±0.40	6.00±0.89	5.80±0.98	6.20±0.98	5.20±1.17	5.60±0.49	5.00±1.79†
digits 5-8	6.80±0.40	6.00±0.63‡	6.60±0.49	5.80±0.75	6.00±0.63‡	6.20±1.17	**5.60±0.49**‡	6.00±1.10
top-pullover	5.00±0.63	5.40±0.49	5.80±0.98†	5.20±0.98†	5.00±0.89	3.80±1.17	4.20±0.75	**3.20±0.40**‡
top-coat	5.40±0.80	4.80±0.40‡	4.60±0.80‡	5.40±0.80	5.20±0.75‡	5.40±0.49‡	4.40±0.80	**3.00±0.00**‡
top-shirt	5.60±0.49	5.40±0.80	5.60±0.80	5.20±1.47	5.60±1.02	5.80±0.75†	**4.20±0.98**	4.60±1.62
trouser-dress	4.20±1.47	5.20±0.75	5.20±1.17	4.40±1.36	5.00±1.10	4.00±1.79	**4.00±0.63**‡	4.60±1.62†
sandal-ankle boot	5.20±0.75	5.80±1.17†	5.40±1.02†	5.40±1.36†	6.20±0.75†	5.40±0.49	4.80±0.75‡	**3.40±0.80**‡
deer-truck	4.80±1.17	4.80±1.17	5.00±0.89	4.60±1.02‡	5.20±0.75	5.20±1.33†	4.60±0.80‡	**4.20±0.98**‡
deer-horse	4.00±0.63	4.40±0.80†	5.20±1.17†	**3.40±0.80**	5.00±0.00†	5.00±0.00†	5.00±0.63†	5.20±0.75†
automobile-truck	4.40±1.02	4.20±1.47	4.40±0.80‡	**3.20±1.33**	5.00±1.41†	5.00±1.26†	5.20±0.40	4.80±0.75
cat-dog	4.00±1.10	4.00±1.26	4.00±1.26	4.40±1.50†	**3.40±0.49**‡	5.40±0.80	4.60±1.02	3.60±0.49‡
dog-horse	**4.00±1.10**	5.00±0.89	5.40±1.02	5.00±0.63	4.60±0.80	5.00±0.63†	4.20±0.75	5.40±0.80
t-test (W/T/L)	—	2/8/2	3/7/2	3/8/1	3/6/3	5/6/1	1/7/4	3/4/5
Average Rank	3.96	4.79	6.00	4.00	5.38	5.38	3.25	3.25

[1] 表中所汇报的实验结果均是每种方法使用标准 5 折交叉验证法在各个数据集中测试集上的平均准确度 (%) 与其相应的标准差。

[2] 通过在 5% 显著水平下的双尾成对 t 检验对比各算法之间是否有明显差异，其中 ‡ 和 † 分别表示 AdaNet 显著弱于或优于 *SAEP* 及其变体的性能。

[3] 表中最后两行分别展示了检验结果和各算法的平均序值。其中 “W/T/L” 表示 AdaNet 显著优于、并无显著不同或者显著劣于相应的对比算法 (即 *SAEP* 及其变体) 的次数；各算法的平均序值由 Friedman 检验 [113] 计算而得。

表 5.4　二分类问题中的 (子) 集成网络结构性能对比之时间代价

注：表 5.2–5.4 应对照来看，每种方法包括三列，即测试准确度 *(%)*、搜索生成的 (子) 集成网络结构的规模大小和搜索过程的时间代价 *(分钟)*。每个数据集 (行) 中的最优实验结果将使用粗体字标出。注意实验中所使用的子网络类型是多层感知机。

Label Pair	**Time Cost (min)**							
	AdaNet	*PRS*	*PAP*	*PIE*	AdaNet.W	*PRS.W*	*PAP.W*	*PIE.W*
digits 6-9	**10.92±1.82**	12.35±0.99	12.91±1.31	13.04±0.76	12.74±1.29	12.24±0.92	13.50±0.28	12.61±1.84†
digits 5-8	13.00±0.46	12.25±0.64‡	13.96±0.34†	**12.16±0.49**‡	13.07±0.46†	13.07±0.83†	13.16±0.62†	13.55±1.18†
top-pullover	11.83±0.46	11.93±0.52†	12.78±1.29†	11.73±1.77	11.84±0.98†	11.22±1.01	12.08±0.58†	**11.02±0.28**‡
top-coat	12.00±0.68	11.21±0.34‡	**11.05±0.70**	11.78±1.27	12.42±0.52	11.90±0.67‡	12.04±0.62	11.09±0.17‡
top-shirt	11.80±1.06	12.06±1.18†	12.63±0.85	**10.89±2.53**	12.55±0.66	12.63±0.74	12.04±0.83	12.26±1.34†
trouser-dress	10.88±1.40	11.98±0.61	12.36±1.02†	11.24±1.74†	12.19±0.66	**10.75±1.72**	11.92±0.71	12.25±1.23
sandal-ankle boot	11.75±0.94	12.09±0.99†	12.55±1.04†	**10.20±2.35**	13.03±0.40	12.34±0.47	12.20±0.77	11.34±0.91‡
deer-truck	15.50±1.81	13.32±2.08	11.72±1.01‡	**11.39±0.81**‡	16.74±1.00	16.11±1.24	16.90±0.90	15.51±1.18
deer-horse	14.66±1.09	12.70±0.98‡	11.94±1.08‡	**9.82±0.58**‡	16.22±0.68	15.99±1.08	16.94±1.04	17.12±1.03†
automobile-truck	15.31±1.33	12.64±2.23	11.35±1.15‡	**10.04±1.31**‡	15.87±1.70†	16.42±1.36†	16.52±0.50	16.97±0.89
cat-dog	24.28±17.37	17.17±1.48‡	23.06±11.87‡	75.08±113.67†	**14.45±1.29**‡	78.41±104.25†	113.55±193.13†	35.34±38.05†
dog-horse	16.79±2.68	23.08±12.98†	77.02±116.93†	**16.07±1.02**‡	71.94±108.17†	119.54±189.82†	17.00±1.81	46.69±39.39†
t-test (W/T/L)	—	4/4/4	5/3/4	2/5/5	4/7/1	4/7/1	3/9/0	6/3/3
Average Rank	3.25	3.50	5.17	2.50	5.58	5.00	6.08	4.92

[1] 表中所汇报的实验结果均是每种方法使用标准 5 折交叉验证法在各个数据集中测试集上的平均准确度 (%) 与其相应的标准差。

[2] 通过在 5% 显著水平下的双尾成对 *t* 检验对比各算法之间是否有明显差异，其中 ‡ 和 † 分别表示 AdaNet 显著弱于或优于 *SAEP* 及其变体的性能。

[3] 表中最后两行分别展示了检验结果和各算法的平均序值。其中 “W/T/L” 表示 AdaNet 显著优于、并无显著不同或者显著劣于相应的对比算法 (即 *SAEP* 及其变体) 的次数；各算法的平均序值由 Friedman 检验 [113] 计算而得。

表 5.5　多分类问题中 (子) 集成网络结构的性能对比

注：每种方法包括三列，即 *测试准确度 (%)*、搜索生成的 (子) 集成网络结构的 *规模大小* 和搜索过程的 *时间代价 (分钟)*。每个数据集 (行) 中的最优实验结果将使用粗体字标出。注意实验中所使用的子网络类型是多层感知机和卷积神经网络。

Dataset	Test Accuracy (%)							
	AdaNet	*PRS*	*PAP*	*PIE*	AdaNet.W	*PRS.W*	*PAP.W*	*PIE.W*
MNIST	94.88±0.22	94.82±0.36†	94.79±0.23†	94.75±0.25†	94.94±0.29	94.66±0.13	94.56±0.28†	**94.94±0.19**‡
Fashion-MNIST	83.74±0.52	83.76±0.88	**83.95±0.50**‡	83.89±0.64	83.81±0.32‡	83.98±0.40‡	**84.24±0.18**‡	83.93±0.21‡
MNIST*	90.54±0.24	90.46±0.25†	90.44±0.15	90.35±0.24†	**90.55±0.18**‡	90.38±0.27†	90.27±0.16	90.23±0.35†
Fashion-MNIST*	81.39±0.43	**81.48±0.30**‡	81.40±0.23‡	81.32±0.45†	81.39±0.26‡	81.41±0.18‡	81.20±0.09	81.05±0.58†
t-test (W/T/L)	—	2/1/1	1/1/2	3/1/0	0/1/3	1/1/2	1/2/1	2/0/2
Average Rank	4.50	3.75	3.75	5.75	3.25	4.00	5.75	5.25

Dataset	Number of Sub-Architectures							
	AdaNet	*PRS*	*PAP*	*PIE*	AdaNet.W	*PRS.W*	*PAP.W*	*PIE.W*
MNIST	6.80±0.40	6.60±0.49	7.00±0.00	6.60±0.49	6.60±0.49	6.20±0.98	**5.20±0.75**‡	6.80±0.40
Fashion-MNIST	5.40±1.02	5.60±0.80	5.40±1.02	6.00±0.63	6.00±0.89	5.60±0.80	5.00±0.63‡	**4.00±1.10**
MNIST*	5.80±0.75	4.80±0.75‡	5.60±1.50	4.80±0.40‡	5.40±1.36	5.00±1.41	3.80±1.17	**3.00±0.00**‡
Fashion-MNIST*	5.40±1.36	3.80±1.47‡	6.40±0.49	5.00±0.63‡	5.60±0.49	4.00±0.63‡	4.00±0.89‡	**3.20±0.40**‡
t-test (W/T/L)	—	0/2/2	0/4/0	0/2/2	0/4/0	0/3/1	0/1/3	0/2/2
Average Rank	6.00	3.75	6.63	5.00	6.13	4.00	2.13	2.38

Dataset	Time Cost (min)							
	AdaNet	*PRS*	*PAP*	*PIE*	AdaNet.W	*PRS.W*	*PAP.W*	*PIE.W*
MNIST	78.65±89.02	23.42±2.32‡	**21.51±0.16**‡	31.73±12.46‡	81.13±87.66	43.36±12.72‡	92.39±117.81†	33.11±1.94‡
Fashion-MNIST	21.75±1.87	**20.07±0.65**‡	25.50±9.07†	39.30±6.03†	42.92±9.71†	91.41±117.32†	31.77±2.16†	75.92±89.19†
MNIST*	30.73±1.10	29.03±0.74‡	28.04±3.87	**20.47±1.27**‡	30.46±1.72	25.88±2.62‡	29.08±1.23	28.25±0.08‡
Fashion-MNIST*	29.61±1.53	27.74±1.75‡	30.63±1.53†	**20.90±0.83**‡	31.12±0.57†	25.01±3.19	28.22±1.92	28.28±0.19‡
t-test (W/T/L)	—	0/0/4	2/1/1	1/0/3	2/2/0	1/1/2	2/2/0	1/0/3
Average Rank	5.50	2.75	3.50	2.50	7.00	4.25	5.50	5.00

[1] 表中所汇报的实验结果均是每种方法使用标准 5 折交叉验证法在各个数据集中测试集上的平均准确度 (%) 与其相应的标准差。

[2] 通过在 5% 显著水平下的双尾成对 t 检验对比各算法之间是否有明显差异，其中 ‡ 和 † 分别表示 AdaNet 显著弱于或优于 *SAEP* 及其变体的性能。

[3] 表中最后两行分别展示了检验结果和各算法的平均序值。其中 “W/T/L” 表示 AdaNet 显著优于、并无显著不同或者显著劣于相应的对比算法 (即 *SAEP* 及其变体) 的次数；各算法的平均序值由 Friedman 检验 [113] 计算而得。

显示了每种方法的性能对比，分别表示其平均序值和 AdaNet 与之相较胜/平/负的次数。可以看出，*SAEP* 在大多数情况下都能够取得比 AdaNet 更优的结果，但在少数情况下需要消耗更久的时间。因此我们可以认为 *SAEP* 能够生成准确度性能更优的集成网络结构。图 5.2(a) 表示 *SAEP* (即 *PRS*、*PAP* 和 *PIE*) 的准确度性能至少可与 AdaNet 持平，且其变体的性能表现甚至优于 AdaNet 和 AdaNet.W。由图 5.2(b) 和表 5.5 中也可得出类似的结论。

5.4.5　*SAEP* 可获得规模更小的集成网络结构

本小节验证剪枝后的子集成网络结构是否比原始集成网络结构规模更小且性能持平甚至更优。实验结果已汇报在表 5.3–5.5 和图 5.3–5.4 中。如表 5.3 所示，

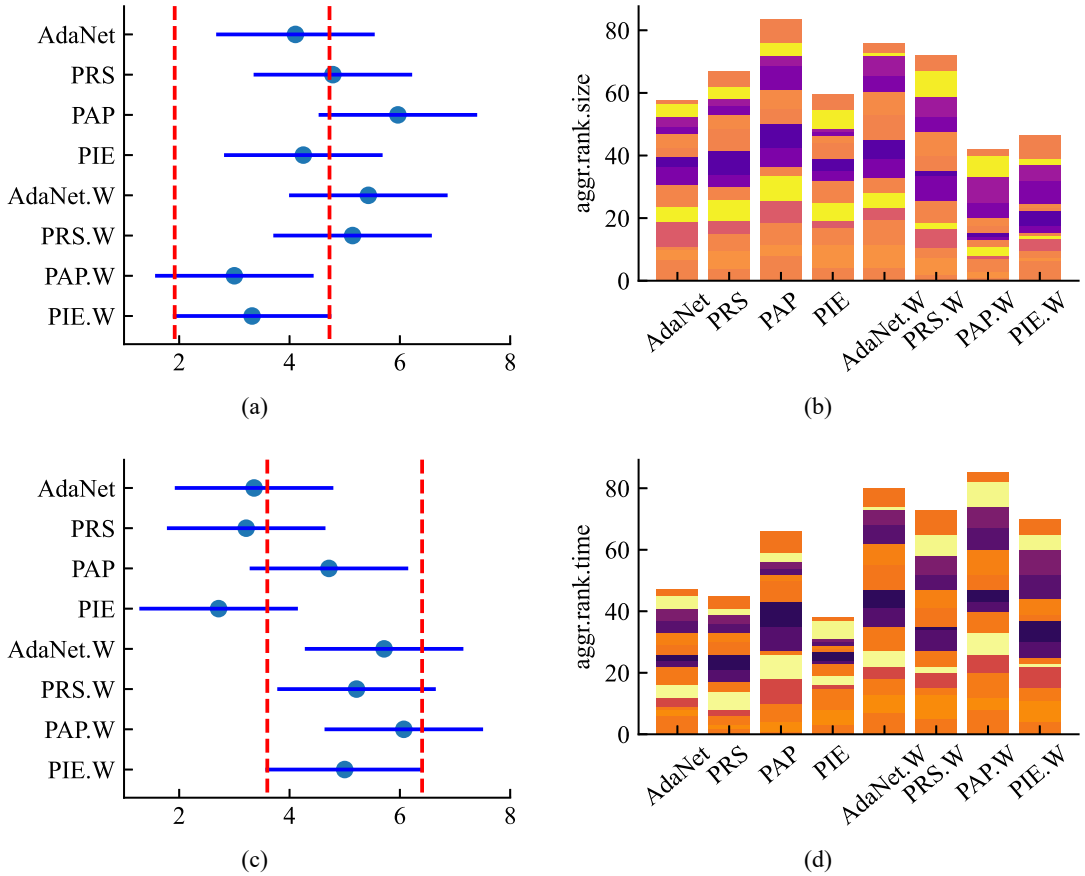


图 5.3　在图像分类任务中对比基准算法和 *SAEP* 及其变体

注：实验中所使用的子网络类型是多层感知机。(a–b) 搜索生成的 (子) 集成网络结构的模型规模对比。(c–d) 搜索过程中的时间代价对比。注意 (a) 和 (c) 中的 Friedman 检验图均在 5% 显著水平下拒绝假设，即认为这些方法差异显著。

SAEP 大多可以获得比 AdaNet 规模更小的集成网络结构，尽管这些方法之间的差异程度并不如准确度般显著，如图 5.3(a)–5.3(b) 所示。此外，表 5.4 和图 5.3(c)–5.3(d) 也揭示了这些方法的变体往往需要消耗更多时间。然而考虑到 *SAEP* 已获得了能与 AdaNet 性能持平的集成网络结构，且 *SAEP* 的确能够缩减该网络结构的规模大小，我们可以认为所提出的神经架构搜索中的集成子结构剪枝方法确乎有其价值。相似的结论也可从图 5.4 中观察得到：(1) *SAEP* 能够获得与 AdaNet 准确度性能持平的集成网络结构，如图 5.4(a) 和 5.4(d) 所示；(2) *SAEP* 能够生成与 AdaNet 性能持平且规模更小的集成网络结构，如图 5.4(e) 所示。

5.4.6　*PIE* 可生成更多样的子集成网络结构

本小节验证的是我们期冀增加集成网络结构多样性的目的是否得以满足。在 *PIE* 中，我们使用了标准信息变异 VI(·, ·) 来表达两个不同子网络结构之间的冗余度，用于反映二者之间的多样性。然而 *PRS* 和 *PAP* 中并不存在类似项可表达

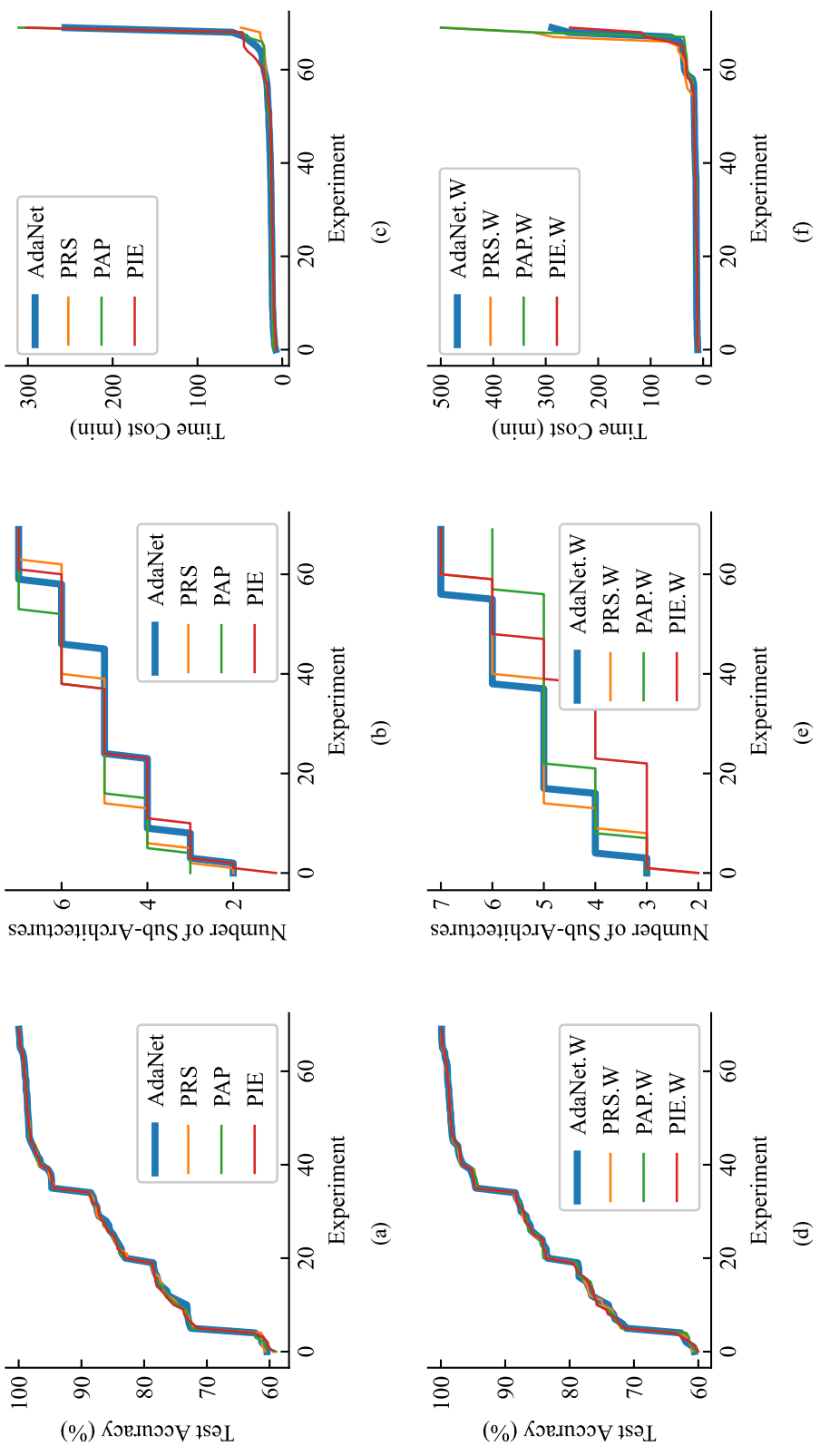


图 5.4　在图像分类任务中对比基准算法和 *SAEP* 及其变体

注： 实验中所使用的子网络类型是多层感知机。与表 5.2–5.5 中不同行的含义相似，图中横轴仅表示在不同实验中的经验结果。(a–c) AdaNet 与 *SAEP* 的性能对比。(d–f) 它们的相应变体的性能对比。

表 5.6　二分类问题中不同 α 值的经验结果对比

注：实验在 MNIST 数据集中的手写数字 5 与 8 标签对上进行，使用的子网络类型是多层感知机。每种方法包括四列，即测试准确度 *(%)*、网络结构的多样性 *(*分歧度量*)*、最终生成的 (子) 集成网络结构的模型规模和搜索过程的时间代价 *(*分钟*)*。其中 “orig.” 分别指代 AdaNet, *PRS*, *PAP* 或 *PIE*; 而 “vari.” 则对应于 AdaNet.W, *PRS.W*, *PAP.W* 或 *PIE.W*.

	Test Accuracy (%)		**Diversity (Disagreement)**		**Size**		**Time Cost (min)**	
	orig.	vari.	orig.	vari.	orig.	vari.	orig.	vari.
AdaNet	99.86±0.06	99.86±0.05	0.0003±0.0001	0.0005±0.0003	5.60±5.60	5.80±5.80	11.53±0.43	12.37±0.75
PRS	99.83±0.07	99.88±0.05	0.0037±0.0043	0.0019±0.0031	5.40±5.40	4.80±4.80	11.93±0.71	11.16±1.52
PAP	99.87±0.05	99.88±0.04	0.0011±0.0018	0.0003±0.0001	5.80±5.80	5.40±5.40	12.76±1.06	12.89±0.62
PIE ($\alpha = 0.5$)	99.80±0.06	99.85±0.04	0.0037±0.0069	0.0030±0.0053	4.80±4.80	5.80±5.80	12.29±1.14	13.15±0.54
PIE ($\alpha = 0.0$)	99.15±0.23	99.19±0.26	0.0022±0.0004	0.0399±0.0201	7.00±0.00	5.80±0.40	14.53±0.10	13.98±0.63
PIE ($\alpha = 0.05$)	99.16±0.24	99.16±0.13	0.0024±0.0004	0.0280±0.0135	6.60±0.80	6.00±1.10	9.56±2.14	13.60±1.16
PIE ($\alpha = 0.1$)	99.13±0.04	99.21±0.15	0.0019±0.0003	0.0456±0.0250	6.40±0.49	5.40±1.20	13.59±0.80	13.22±0.91
PIE ($\alpha = 0.15$)	99.24±0.14	99.25±0.17	0.0020±0.0004	0.0313±0.0299	6.00±0.89	5.80±0.98	12.95±0.64	14.00±0.75
PIE ($\alpha = 0.2$)	99.14±0.17	99.21±0.06	0.0022±0.0006	0.0574±0.0162	6.20±0.75	6.40±0.49	14.20±0.72	14.58±0.18
PIE ($\alpha = 0.25$)	99.22±0.12	99.15±0.19	0.0020±0.0004	0.0378±0.0294	6.00±0.63	6.40±0.80	13.08±0.61	14.30±0.71
PIE ($\alpha = 0.3$)	99.29±0.12	99.18±0.22	0.0023±0.0003	0.0415±0.0251	7.00±0.00	6.40±0.49	13.16±1.54	14.36±0.41
PIE ($\alpha = 0.35$)	99.19±0.11	99.24±0.09	0.0021±0.0006	0.0210±0.0158	5.60±1.02	5.40±1.20	12.85±0.73	13.20±0.93
PIE ($\alpha = 0.4$)	99.22±0.13	99.20±0.22	0.0022±0.0002	0.0364±0.0288	6.80±0.40	6.20±0.75	11.02±0.12	14.49±0.53
PIE ($\alpha = 0.45$)	99.12±0.12	99.18±0.18	0.0023±0.0009	0.0486±0.0232	6.60±0.80	6.60±0.80	13.62±0.40	14.59±0.44
PIE ($\alpha = 0.55$)	99.17±0.19	99.18±0.20	0.0022±0.0004	0.0454±0.0107	6.00±0.63	6.40±0.49	12.99±0.69	14.45±0.59
PIE ($\alpha = 0.6$)	99.23±0.08	99.19±0.15	0.0022±0.0002	0.0499±0.0275	6.00±1.10	5.60±0.80	9.96±1.46	13.65±1.03
PIE ($\alpha = 0.65$)	99.20±0.14	99.29±0.12	0.0018±0.0003	0.0173±0.0147	6.40±0.80	5.40±1.02	13.14±0.87	13.21±1.08
PIE ($\alpha = 0.7$)	99.23±0.21	99.25±0.12	0.0020±0.0003	0.0234±0.0228	6.60±0.80	6.00±1.26	8.17±0.64	13.78±0.83
PIE ($\alpha = 0.75$)	99.16±0.21	99.22±0.15	0.0022±0.0005	0.0428±0.0216	6.60±0.49	6.00±0.63	13.48±0.51	14.21±0.64
PIE ($\alpha = 0.8$)	99.17±0.15	99.19±0.03	0.0028±0.0017	0.0269±0.0213	6.40±0.49	5.40±1.20	7.80±0.47	13.35±1.05
PIE ($\alpha = 0.85$)	99.20±0.13	99.18±0.17	0.0020±0.0003	0.0411±0.0224	5.60±0.49	6.60±0.49	12.81±0.35	14.45±0.47
PIE ($\alpha = 0.9$)	99.20±0.23	99.28±0.16	0.0017±0.0003	0.0596±0.0172	6.00±0.63	6.00±0.63	7.67±0.92	13.99±0.45
PIE ($\alpha = 0.95$)	99.22±0.19	99.22±0.10	0.0020±0.0003	0.0283±0.0210	5.80±0.40	5.80±1.17	12.51±0.38	13.82±0.80
PIE ($\alpha = 1.0$)	99.19±0.19	99.28±0.15	0.0021±0.0006	0.0483±0.0159	6.20±0.75	6.40±0.49	8.07±0.35	14.34±0.27

其多样性程度，因此在这个实验中，我们使用另外一种名为分歧度量 [57-58] 的多样性度量方法，用以计算集成网络结构和剪枝后子集成网络结构的多样性。务请注意学界中有许多方法可用于计算多样性，而分歧度量只是其中一种 [2]。我们在此选择分歧度量的原因是该法易于计算和理解。在两个不同的子网络结构之间通过分歧度量计算而得的多样性可定义为

$$\mathrm{dis}(\mathbf{w}_i, \mathbf{w}_j) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}(\mathbf{h}_i(\boldsymbol{x}_i) \neq \mathbf{h}_j(\boldsymbol{x}_i)) \,, \tag{5.17}$$

而整个集成网络结构 f 的通过分歧度量计算而得的多样性则是

$$\mathbf{dis}(f) = \frac{2}{l(l-1)} \sum_{\mathbf{w}_i \cdot \mathbf{h}_i \in f} \sum_{\substack{\mathbf{w}_j \cdot \mathbf{h}_j \in f, \\ \mathbf{h}_j \neq \mathbf{h}_i}} \mathrm{dis}(\mathbf{w}_i, \mathbf{w}_j) \,, \tag{5.18}$$

且经剪枝后的子集成网络结构 $f \setminus \{\mathbf{w} \cdot \mathbf{h}\}$ 的多样性可类似计算得到。

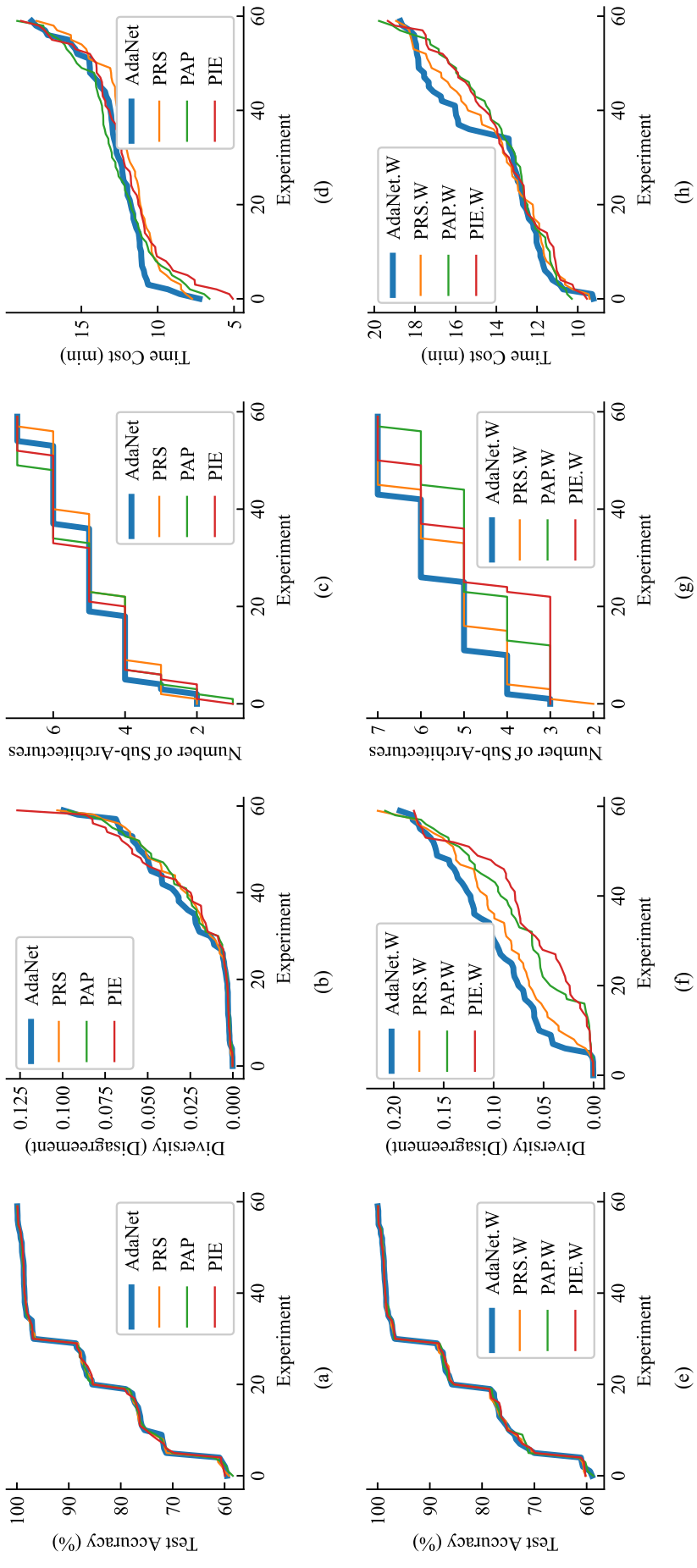


图 5.5　在二分类问题中对比基准算法和 *SAEP* 及其变体

注： 实验中使用的子网络类型是多层感知机。与表 5.2–5.5 中不同行的含义相似，图中横轴仅用作表示不同实验中的经验结果，别无他意。(a–c) AdaNet 和 *SAEP* 的性能对比。(d–f) 其相应变体的性能对比。

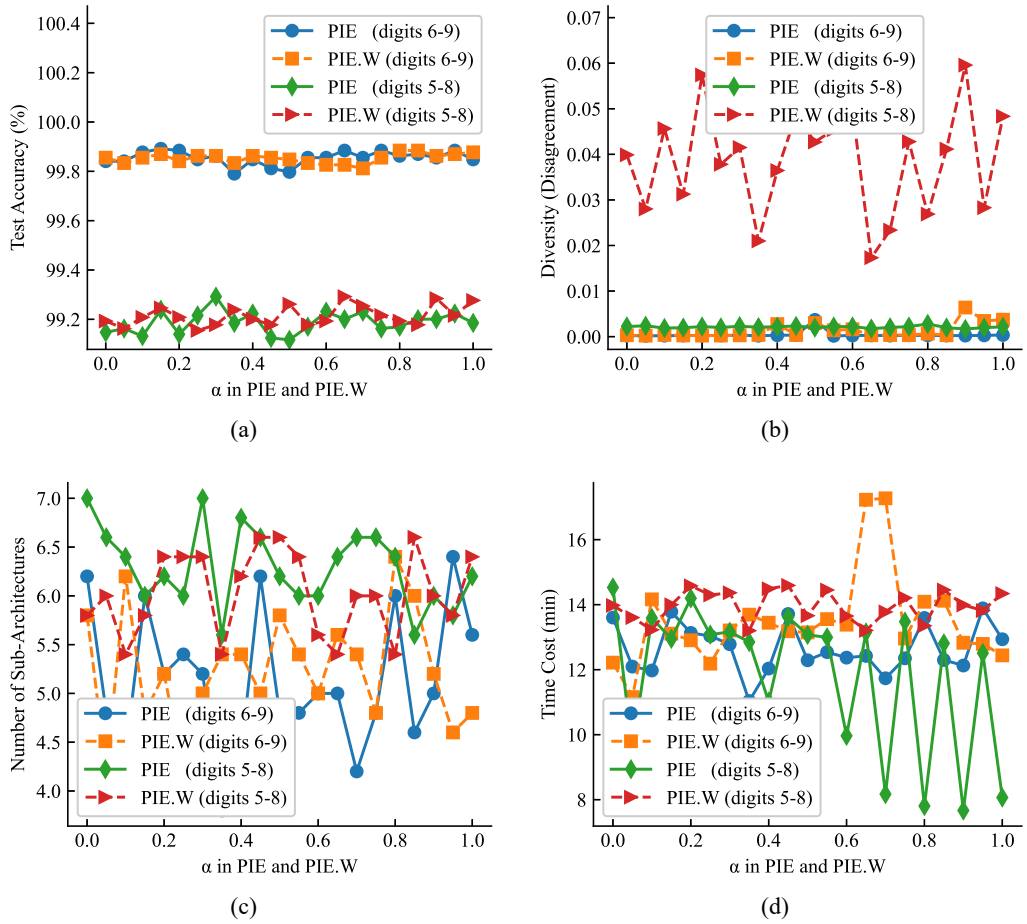


图 5.6　在二分类问题中不同 α 值对 *PIE* 和 *PIE.W* 的影响

注：(a–d) 分别展示了不同 α 值在子集成网络结构的准确度、多样性、模型规模和时间代价上的影响。

表 5.6 和图 5.5–5.6 汇报了这些方法的性能及其分歧度量值，以反映最终集成网络结构的多样性。表 5.6 同时汇报了使用 *PIE* 的子集成网络结构的多样性及相关信息。注意分歧度量值愈大，则集成网络结构或剪枝后子集成网络结构的多样性愈高。观察表 5.6 可知，*PAP* 同时获得了较高的准确度和多样性；而与 AdaNet.W 相比，*PRS.W* 和 *PAP.W* 也表现出相似的结果。这说明子集成网络结构的准确度也可受益于多样性的提高。与此同时，表 5.6 也展示了在某些情形下，规模较大的子集成网络结构反而对应的多样性值较小。此外，由图 5.6 可知，式 (5.15) 中参数 α 值的变化也会对子集成网络结构的性能略有影响。

5.4.7　参数 α 值的影响

本小节旨在探索 *PIE* 中超参数 α 值的影响，该超参数的值也隐含着式 (5.15) 中两个准则之间的关系。为了研究这一问题，实验中使用了多个 α 值，其变化范围由 0.0 到 1.0，步长为 0.05．图 5.6 分别展示了 MNIST 数据集中两个标签对

表 5.7　剪枝后子集成网络结构规模的实验结果

注：实验在 MNIST 数据集中的数字 5-8 这一标签对上进行，所使用的子网络类型是多层感知机。每种方法包括五列，即测试准确度 *(%)*、多样性 *(*分歧度量*)*、时间代价 *(*分钟*)*、模型规模大小 (即子网络结构的数量) 以及生成的子网络结构的索引。

	Test Accuracy (%)	Diversity (Disagreement)	Time Cost (min)	Size	Indexes
AdaNet	99.86	0.0001	10.71	6	[0,1,3,4,5,6]
PRS	99.92	0.0101	10.54	4	[0,1,2,5]
PAP	99.93	0.0002	13.95	7	[0,1,2,3,4,5,6]
PIE (α=0.25)	99.24	0.0016	13.31	6	[0,1,2,3,4,5]
PIE (α=0.5)	99.89	0.0002	12.89	5	[0,1,2,3,4]
PIE (α=0.75)	99.12	0.0024	13.53	6	[0,1,2,3,5,6]
AdaNet.W	99.82	0.0002	11.69	5	[0,1,2,3,6]
PRS.W	99.93	0.0006	13.82	7	[0,1,2,3,4,5,6]
PAP.W	99.86	0.0003	12.47	6	[0,1,3,4,5,6]
PIE.W (α=0.25)	99.20	0.0029	12.91	5	[0,1,2,3,6]
PIE.W (α=0.5)	99.78	0.0004	13.69	6	[0,1,2,3,4,5]
PIE.W (α=0.75)	99.01	0.0445	13.99	6	[0,1,2,3,4,6]

(即数字 5-8 和 6-9) 的实验结果。图 5.6(a) 说明了不同的 α 值对子集成网络结构的准确度性能影响有限，改变的 α 值不会过多地损害其性能；而图 5.6(b) 则说明 *PIE.W* 中子集成网络结构的多样性会受到一定影响，尽管其绝对值并不会发生过多改变，而 *PIE* 中的多样性则几乎不受不同 α 值的影响。图 5.6(c)–5.6(d) 展示出 *PIE* 中子集成网络结构的模型规模和时间代价将在不同的 α 值下受到更多影响，其大致趋势是随 α 值的增加而降低。

5.4.8　AdaNet 与 *SAEP* 的时间代价对比

本小节比较了 AdaNet 与 *SAEP* 及其相应变体的时间代价。实验结果已汇报在表 5.4–5.6 和图 5.3–5.6 中，包括每种方法在测试集上的准确性以及相应的耗时。尽管图 5.3(c)–5.3(d) 展示出 *SAEP* 在维持准确度的同时并不见长于削减耗时，表 5.2–5.5 也说明 *SAEP* 有时可以在更短时间内生成令人满意的子集成网络结构。尽管 *SAEP* 增加了剪枝这一过程，但因其与搜索过程同时进行，故时间代价通常仍取决于整个搜索过程中生成的子网络结构的数量。因此，如果搜索中生成了更多的子网络结构，那么 *SAEP* 需要消耗更久的时间也是合乎情理的。图 5.6(d) 展示了 *PIE* 中不同 α 值对生成更多样的子集成网络结构的时间代价影响。

5.4.9　*SAEP* 可以生成独特的更深层次的子网络结构

在少数情形下，我们观察到 *PIE* 获得了比 AdaNet 规模更大的集成网络结构，这令我们对 “*SAEP* 是否能生成比之 AdaNet 而言独特的子网络结构” 这一问题生

出探究之心。因此，本小节对剪枝后子集成网络结构的具体情形进行了更进一步的探索。正如我们在表 5.7 中所看到的那样，当子集成网络结构的规模与未剪枝前的集成网络结构规模持平甚至有所超出时，这些子集成网络结构的多样性往往低于未剪枝的集成网络结构的多样性。*PIE* (或 *PIE.W*) 生成独特的更深层次的子网络结构的原因或许在于在搜索过程中，式 (5.16) 中优化目标的多样性并未得以充分满足。那么在这种情况下，为了提高子集成网络结构的多样性，该优化目标就会指引对更深层次的子网络结构进行搜索。

5.5　本章小结

最近一些将神经架构搜索与集成学习方法进行结合的研究探索已经在缩减搜索复杂度和提高效率方面获得了显著的结果 [78]。但是这些方法通常都忽略了集成学习中的多样性这一重要因素。为了解决这个问题，本章意在研究神经架构搜索中的集成子结构剪枝问题，并提出了一种剪枝方法，名为“神经架构搜索中的集成子结构剪枝 (*SAEP*)”，可用于在搜索过程中缩减冗余的子网络结构。三种不同的剪枝策略被提出，包括基于随机选择的剪枝 (*PRS*)、基于准确度性能的剪枝 (*PAP*) 和基于信息熵的剪枝 (*PIE*)，用作搜索过程中的指导，决定哪些子网络结构将被剪枝。实验结果证明了 *SAEP* 能够在指导搜索过程生成更多样的子网络结构并构建规模更小的子集成网络结构的同时维持未剪枝前的集成网络结构的准确度性能。除此之外，当搜索过程中子网络结构之间的多样性不够充足时，*PIE* 能够指引搜索出独特的更深层次的子网络结构。在未来的工作中，将现有方法进行推广以获得更多样的集成策略，和探究其背后的理论基础将是两个值得探索的方向，有助于提高神经架构搜索中集成网络结构的性能。

第 6 章　总结与展望

本章首先总结全文提出的分类集成问题中的多样性的三个研究工作，然后从它们仍存在的问题以及基于多样性的剪枝方法所可能的扩展方向两方面来对未来工作进行展望。

6.1　全文总结

随着互联网的普及和各种智能终端的广泛应用，全球数据总量连年上涨，如何从这些海量信息中提取准确而有价值的信息是人工智能所面临的主要挑战之一。而在面对海量的低价值密度的大数据时，由传统机器学习方法所构建的单个学习器或许难以满足我们所需的精度需求。因此集成学习作为一种能显著提高弱分类器性能的方法，以其优越的性能吸引了诸多研究者的注意。由于多样性是集成学习成功的两大要素之一，故本文围绕分类集成问题中的多样性进行探索性研究。

具体而言，本文首先研究了分类问题中集成器的多样性与其泛化性能之间的关系，并在此基础上提出了直接利用多样性的剪枝算法；随后利用信息熵度量来平衡准确度和多样性，提出了隐式利用多样性的剪枝算法及其通用并行框架。除此之外，本文还探索了多样性在其他领域与集成学习相结合的研究中的应用，提出了神经架构搜索中的集成子结构剪枝方法。本文的贡献与创新点主要体现在以下几个方面：

(1) 首先研究的是分类问题中集成器的多样性与泛化性能的关系。多样性一直是集成学习领域中的一个重要概念，然而学界对于多样性在分类集成器中如何起作用这一问题的结论尚悬而未决。第 3 章受启发于回归集成器的误差分解，提出了分类集成器的误差分解及多样性度量；并随后利用该多样性度量方法提出了分类集成器中多样性与泛化性能的关系；最后利用该关系提出了基于多样性的集成剪枝方法。实验结果证明了该章所提出方法的合理性和有效性。

(2) 其次研究的是基于信息熵的集成剪枝及其通用并行框架。该章是在集成学习领域利用多样性来解决集成剪枝问题的一个探索。第 4 章从信息熵角度对准确度和多样性进行平衡，提出了一种基于信息熵的目标最大化的剪枝算法，且包括集中式和分布式两个版本；随后又提出了一个可普适于现有剪枝算法的通用并行框架，能够在不损害算法性能的同时大幅加速算法执行的过程。实验结果验证了该章所提出方法的合理性和有效性。

(3) 最后研究的是神经架构搜索中的集成子结构剪枝。第 5 章是在其他领域对集成学习和多样性的一个应用。特别地，该章提出了一个针对于神经架构搜索中的集成网络结构的剪枝方法，其意在利用多样性这一要素来获得相较原集成网络结构而言规模更小而性能持平甚至更优的子集成网络结构。在这一搜索过程中，共有三种可供选择的剪枝策略被提出，用于决定何种子网络结构将被剪枝。实验结果验证了该章所提出方法的合理性和有效性。

总的来说，本文关注分类集成问题中的多样性研究与应用。第 3 章和第 4 章是在集成学习领域中分析分类集成器中多样性与泛化性能的关系，并分别探索了多样性在集成剪枝问题中的直接应用和间接应用，通过对准确度和多样性的合理平衡来获取性能良好的子集成器；而第 5 章则是在神经架构搜索领域中探索多样性在集成网络结构中的应用，以期获得规模更小而性能持平甚至更优的子集成网络结构。

6.2　未来展望

本文通过探索集成学习中的多样性所起到的作用来改善集成剪枝方法，虽然在分类问题中集成器的多样性与泛化性能的关系以及在集成剪枝方法里的应用中取得了一定的成果，但是这些工作仍存在着很大的改进空间。同时多样性作为集成学习中的要素之一，在集成剪枝问题中仍有许多值得研究的问题。

对于多样性在分类集成问题中的作用，本文只探讨了在二分类场景下集成器的多样性与泛化性能的关系，但是实际问题并不局限于二分类任务。因此在多分类场景下集成器的多样性与泛化性能是否仍存在着同样的关系是一个值得研究的问题。而本文中的理论分析对二分类标签的依赖性较大，因此在多分类场景下分析多样性与泛化性能的关系也是一项很有挑战性的工作。

对于多样性在集成剪枝问题中的应用，本文只从实践角度展示了在集成学习和神经架构搜索领域内所提出的剪枝方法的有效性，但是对其理论基础的探索尚有不足。如果我们能够获得能够支持所提出算法有效性的理论依据，那么或许可据此对其进行改进，也有助于进一步提高剪枝后子集成器的性能。

此外，在剪枝过程中利用多样性来公平地挑选基分类器和评估不同剪枝方法的性能等问题中，多样性也存在着一定的用武之地。如何利用多样性快速有效地完成剪枝任务仍然是一个值得探索的研究方向。

参 考 文 献

[1] DIETTERICH T. Ensemble methods in machine learning[C]//MCS: volume 1857. 2000: 1-15.

[2] ZHOU Z H. Ensemble methods: Foundations and algorithms[M]. CRC press, 2012.

[3] KUNCHEVA L, WHITAKER C. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy[J]. Mach Learn, 2003, 51(2): 181-207.

[4] BREIMAN L. Bagging predictors[J]. Mach Learn, 1996, 24(2): 123-140.

[5] KEARNS M, VALIANT L. Cryptographic limitations on learning boolean formulae and finite automata[C]//STOC. 1989: 433-444.

[6] KEARNS M, VALIANT L. Cryptographic limitations on learning boolean formulae and finite automata[J]. J ACM, 1994.

[7] SCHAPIRE R E. The strength of weak learnability (extended abstract)[C]//FOCS. 1989.

[8] SCHAPIRE R E. The strength of weak learnability[J]. Mach Learn, 1990, 5(2): 197-227.

[9] FRIEDMAN J. Additive logistic regression : a statistical view of boosting[J]. Annals of Statistic, 2000, 38.

[10] WOLPERT D H. Stacked generalization[J]. Neural Netw, 1992, 5(2): 241-259.

[11] BREIMAN L. Stacked regressions[J]. Mach Learn, 1996, 24(1): 49-64.

[12] SMYTH P, WOLPERT D. Stacked density estimation[C]//NIPS. Cambridge, MA: MIT Press, 1998: 668-674.

[13] 韦鹏程, 冉维, 段昂. 大数据巨量分析与机器学习的整合与开发[M]. Beijing Book Co. Inc, 2017.

[14] 刘鹏看未来. 大数据时代：数据是如何爆炸式增长的？[Z]. 2015.

[15] TANG E, SUGANTHAN P, YAO X. An analysis of diversity measures[J]. Mach Learn, 2006, 65(1): 247-271. DOI: 10.1007/s10994-006-9449-2.

[16] FREUND Y. Boosting a weak learning algorithm by majority[J]. Inf Comput, 1995, 121(2): 256-285.

[17] FREUND Y, SCHAPIRE R E, et al. Experiments with a new boosting algorithm[C]//ICML: volume 96. Bari, Italy, 1996: 148-156.

[18] CHEN H, JIANG B, YAO X. Semisupervised negative correlation learning[J]. IEEE Trans Neural Netw Learn Syst, 2018, 29(11): 5366-5379.

[19] SOARES R G F, CHEN H, YAO X. Semisupervised classification with cluster regularization [J]. IEEE Trans Neural Netw Learn Syst, 2012, 23(11): 1779-1792.

[20] YOUNG S, ABDOU T, BENER A. Deep super learner: A deep ensemble for classification

problems[J]. arXiv preprint arXiv:1803.02323, 2018.

[21] GIRSHICK R, DONAHUE J, DARRELL T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]//CVPR. 2014: 580-587.

[22] SABOUR S, FROSST N, HINTON G E. Dynamic routing between capsules[C]//NIPS. 2017: 3859-3869.

[23] WANG J, LIU Z, WU Y, et al. Mining actionlet ensemble for action recognition with depth cameras[C]//CVPR. 2012: 1290-1297.

[24] ZHOU X, XIE L, ZHANG P, et al. An ensemble of deep neural networks for object tracking [C]//ICIP. 2014: 843-847. DOI: 10.1109/ICIP.2014.7025169.

[25] KIM W, GOYAL B, CHAWLA K, et al. Attention-based ensemble for deep metric learning [J]. arXiv preprint arXiv:1804.00382, 2018.

[26] WANG X, ZHANG S, LEI Z, et al. Ensemble soft-margin softmax loss for image classification[J]. arXiv preprint arXiv:1805.03922, 2018.

[27] PHAYE S S R, SIKKA A, DHALL A, et al. Dense and diverse capsule networks: Making the capsules learn better[J]. arXiv preprint arXiv:1805.04001, 2018.

[28] SHUI C, MOZAFARI A S, MAREK J, et al. Diversity regularization in deep ensembles[J]. arXiv preprint arXiv:1802.07881, 2018.

[29] STANESCU A, PANDEY G. Developing parsimonious ensembles using ensemble diversity within a reinforcement learning framework[J]. arXiv preprint arXiv:1805.02103, 2018.

[30] LU Z, WU X, ZHU X, et al. Ensemble pruning via individual contribution ordering[C]// SIGKDD. ACM, 2010: 871-880. DOI: 10.1145/1835804.1835914.

[31] KROGH A, VEDELSBY J. Neural network ensembles, cross validation, and active learning [C]//NIPS. 1995: 231-238.

[32] BROWN G. An information theoretic perspective on multiple classifier systems[C]//MCS. 2009: 344-353.

[33] ZHOU Z H, LI N. Multi-information ensemble diversity[C]//MCS. 2010: 134-144.

[34] GU S, CHENG R, JIN Y. Multi-objective ensemble generation[J]. Wiley Interdiscip Rev Data Min Knowl Discov, 2015, 5(5): 234-245.

[35] YU Y, LI Y F, ZHOU Z H. Diversity regularized machine[C]//IJCAI. 2011: 1603.

[36] GUO X. Exclusivity regularized machine[J]. arXiv preprint arXiv:1603.08318, 2016.

[37] JIANG Z, LIU H, FU B, et al. Generalized ambiguity decompositions for classification with applications in active learning and unsupervised ensemble pruning.[C]//AAAI. 2017: 2073-2079.

[38] LIU Z, DAI Q, LIU N. Ensemble selection by grasp[J]. Appl Intell, 2014, 41(1): 128-144.

[39] KOKKINOS Y, MARGARITIS K G. Confidence ratio affinity propagation in ensemble se-

lection of neural network classifiers for distributed privacy-preserving data mining[J]. Neurocomputing, 2015, 150: 513-528.

[40] ZHANG C X, ZHANG J S, YIN Q Y. A ranking-based strategy to prune variable selection ensembles[J]. Knowl Based Syst, 2017, 125: 13-25.

[41] LYSIAK R, KURZYNSKI M, WOLOSZYNSKI T. Optimal selection of ensemble classifiers using measures of competence and diversity of base classifiers[J]. Neurocomputing, 2014, 126: 29-35.

[42] PARTALAS I, TSOUMAKAS G, VLAHAVAS I P. Focused ensemble selection: A diversity-based method for greedy ensemble selection.[C]//ECAI. 2008: 117-121.

[43] CARUANA R, NICULESCU-MIZIL A, CREW G, et al. Ensemble selection from libraries of models[C]//ICML. ACM, 2004: 18.

[44] BANFIELD R E, HALL L O, BOWYER K W, et al. Ensemble diversity measures and their application to thinning[J]. Inf Fusion, 2005, 6(1): 49-62.

[45] ZHOU Z H, WU J, TANG W. Ensembling neural networks: many could be better than all [J]. Artif Intell, 2002, 137(1-2): 239-263.

[46] MARTÍNEZ-MUÑOZ G, HERNÁNDEZ-LOBATO D, SUÁREZ A. An analysis of ensemble pruning techniques based on ordered aggregation[J]. IEEE Trans Pattern Anal Mach Intell, 2009, 31(2): 245-259. DOI: 10.1109/TPAMI.2008.78.

[47] TSOUMAKAS G, PARTALAS I, VLAHAVAS I. An ensemble pruning primer[M]//SUEMA. Springer, 2009: 1-13.

[48] LI N, YU Y, ZHOU Z H. Diversity regularized ensemble pruning[C]//ECML-PKDD. 2012: 330-345.

[49] QIAN C, YU Y, ZHOU Z H. Pareto ensemble pruning[C]//AAAI. 2015: 2935-2941.

[50] HYAFIL L, RIVEST R L. Constructing optimal binary decision trees is np-complete[J]. Inf Process Lett, 1976, 5(1): 15-17.

[51] BLUM A L, RIVEST R L. Original contribution: Training a 3-node neural network is np-complete[J]. Neural Netw, 1992, 5(1): 117-127.

[52] GU S, JIN Y. Generating diverse and accurate classifier ensembles using multi-objective optimization[C]//MCDM. IEEE, 2014: 9-15.

[53] CHANDRA A, YAO X. Divace: Diverse and accurate ensemble learning algorithm[C]// IDEAL. Springer, 2004: 619-625.

[54] ROLI F, GIACINTO G, VERNAZZA G. Methods for designing multiple classifier systems [C]//MCS. Springer, 2001: 78-87.

[55] YULE G U. On the association of attributes in statistics: with illustrations from the material of the childhood society, &c[J]. Philos Trans Royal Soc A, 1900, 194(252-261): 257-319.

[56] COHEN J. A coefficient of agreement for nominal scales[J]. Educ Psychol Meas, 1960, 20 (1): 37-46.

[57] SKALAK D B, et al. The sources of increased accuracy for two proposed boosting algorithms [C]//AAAI: volume 1129. 1996: 1133.

[58] HO T K. The random subspace method for constructing decision forests[J]. IEEE Trans Pattern Anal Mach Intell, 1998, 20(8): 832-844.

[59] SNEATH P, SOKAL R. Numerical taxonomy: The principles and practice of numerical classification.[M]. WH Freeman, 1973.

[60] GIACINTO G, ROLI F. Design of effective neural network ensembles for image classification purposes[J]. Image Vis Comput, 2001, 19(9): 699-707.

[61] FLEISS J L, LEVIN B, PAIK M C. Statistical methods for rates and proportions[M]. John Wiley & Sons, 2013.

[62] KOHAVI R, WOLPERT D H, et al. Bias plus variance decomposition for zero-one loss functions[C]//ICML: volume 96. 1996: 275-83.

[63] CUNNINGHAM P, CARNEY J. Diversity versus quality in classification ensembles based on feature selection[C]//ECML. Springer, 2000: 109-116.

[64] SHIPP C A, KUNCHEVA L I. Relationships between combination methods and measures of diversity in combining classifiers[J]. Inf Fusion, 2002, 3(2): 135-148.

[65] HANSEN L K, SALAMON P. Neural network ensembles[J]. IEEE Trans Pattern Anal Mach Intell, 1990, 12(10): 993-1001.

[66] PARTRIDGE D, KRZANOWSKI W. Software diversity: practical statistics for its measurement and exploitation[J]. Inf Softw Technol, 1997, 39(10): 707-717.

[67] LIU Y, YAO X. Ensemble learning via negative correlation[J]. Neural Netw, 1999, 12(10): 1399-1404.

[68] ZENOBI G, CUNNINGHAM P. Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error[C]//ECML. Springer, 2001: 576-587.

[69] BROWN G, KUNCHEVA L I. “good” and “bad” diversity in majority vote ensembles[C]// MCS. Springer, 2010: 124-133.

[70] MARGINEANTU D, DIETTERICH T. Pruning adaptive boosting[C]//ICML: volume 97. 1997: 211-218.

[71] MARTÍNEZ-MUÑOZ G, SUÁREZ A. Using boosting to prune bagging ensembles[J]. Pattern Recognit Lett, 2007, 28(1): 156-165. DOI: https://doi.org/10.1016/j.patrec.2006.06.018.

[72] MARTÍNEZ-MUÑOZ, SUÁREZ. Pruning in ordered bagging ensembles[C]//ICML. 2006: 609-616. DOI: 10.1145/1143844.1143921.

[73] ZHOU Z H, TANG W. Selective ensemble of decision trees[C]//RSFDGrC. 2003: 476-483.

[74] PARTALAS I, TSOUMAKAS G, VLAHAVAS I. A study on greedy algorithms for ensemble pruning[J]. Aristotle University of Thessaloniki, Thessaloniki, Greece, 2012.

[75] GUO H, SUN F, CHENG J, et al. A novel margin-based measure for directed hill climbing ensemble pruning[J]. Math Probl Eng, 2016.

[76] DAI Q, YAO C. A hierarchical and parallel branch-and-bound ensemble selection algorithm [J]. Appl Intell, 2017, 46(1): 45-61.

[77] CORTES C, GONZALVO X, KUZNETSOV V, et al. Adanet: Adaptive structural learning of artificial neural networks[J]. arXiv preprint arXiv:1607.01097, 2016.

[78] CORTES C, GONZALVO X, KUZNETSOV V, et al. Adanet: Adaptive structural learning of artificial neural networks[C]//ICML. 2017: 874-883.

[79] FLEISS J L, LEVIN B, PAIK M C. Statistical methods for rates and proportions[M]. John Wiley & Sons, 2013.

[80] DIETTERICH T G. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization[J]. Mach Learn, 2000, 40(2): 139-157.

[81] MARTINEZ-MUNOZ G, SUÁREZ A. Aggregation ordering in bagging[C]//AIA. Citeseer, 2004: 258-263.

[82] GIACINTO G, ROLI F, FUMERA G. Design of effective multiple classifier systems by clustering of classifiers[C]//ICPR: volume 2. IEEE, 2000: 160-163.

[83] LAZAREVIC A, OBRADOVIC Z. Effective pruning of neural network classifier ensembles [C]//IJCNN: volume 2. IEEE, 2001: 796-801.

[84] BAKKER B, HESKES T. Clustering ensembles of neural network models[J]. Neural Netw, 2003, 16(2): 261-269.

[85] GUO Y, LIU Y, OERLEMANS A, et al. Deep learning for visual understanding: A review [J]. Neurocomputing, 2016, 187: 27-48.

[86] QIAN C, YU Y, ZHOU Z H. Analyzing evolutionary optimization in noisy environments [J/OL]. Evol Comput, 2015. https://arxiv.org/pdf/1311.4987.pdf.

[87] LECUN Y, BENGIO Y, HINTON G. Deep learning[J]. Nature, 2015, 521(7553): 436.

[88] SAGI O, ROKACH L. Ensemble learning: A survey[J]. Wiley Interdiscip Rev Data Min Knowl Discov, 2018, 8(4): e1249.

[89] KONTSCHIEDER P, FITERAU M, CRIMINISI A, et al. Deep neural decision forests[C]// ICCV. 2015.

[90] WEN G, ZHI H, LI H, et al. Ensemble of deep neural networks with probability-based fusion for facial expression recognition[J]. Cognit Comput, 2017, 9(4): 1-14.

[91] QIU X, ZHANG L, REN Y, et al. Ensemble deep learning for regression and time series forecasting[C]//CIEL. IEEE, 2014: 1-6.

[92] DENG L, PLATT J. Ensemble deep learning for speech recognition[C]//INTERSPEECH. 2014.

[93] ZHOU Z H, FENG J. Deep forest: Towards an alternative to deep neural networks[C]//IJCAI. 2017.

[94] SRIVASTAVA N, HINTON G, KRIZHEVSKY A, et al. Dropout: A simple way to prevent neural networks from overfitting[J]. J Mach Learn Res, 2014: 1929-1958.

[95] WU X, KUMAR V, QUINLAN J R, et al. Top 10 algorithms in data mining[J]. Knowl Inf Syst, 2008, 14(1): 1-37.

[96] ROKACH L. Ensemble-based classifiers[J]. Artif Intell Rev, 2010, 33(1-2): 1-39.

[97] HERRERA F, CHARTE F, RIVERA A J, et al. Ensemble-based classifiers[M]//Multilabel Classification. Springer, 2016: 101-113.

[98] GEMAN S, BIENENSTOCK E, DOURSAT R. Neural networks and the bias/variance dilemma[J]. Neural Comput, 1992, 4(1): 1-58.

[99] HERBRICH R, GRAEPEL T. A pac-bayesian margin bound for linear classifiers: Why svms work[C]//NIPS. 2001: 224-230.

[100] HERBRICH R, GRAEPEL T. A pac-bayesian margin bound for linear classifiers[J]. IEEE Trans Inf Theory, 2002, 48(12): 3140-3150.

[101] FREUND Y, SCHAPIRE R. A decision-theoretic generalization of on-line learning and an application to boosting[J]. J Comput Syst Sci, 1997, 55(1): 119-139.

[102] BREIMAN L. Random forests[J]. Mach Learn, 2001, 45(1): 5-32.

[103] RODRIGUEZ J J, KUNCHEVA L I, ALONSO C J. Rotation forest: A new classifier ensemble method[J]. IEEE Trans Pattern Anal Mach Intell, 2006, 28(10): 1619-1630.

[104] BROWN G, WYATT J L, TIŇO P. Managing diversity in regression ensembles[J]. J Mach Learn Res, 2005, 6(Sep): 1621-1650.

[105] MENDES-MOREIRA J, SOARES C, JORGE A M, et al. Ensemble approaches for regression: A survey[J]. ACM Comput Surv, 2012, 45(1): 10.

[106] DEPARTMENT D M. The first and second derivatives[EB/OL]. https://math.dartmouth.edu/opencalc2/cole/lecture8.pdf.

[107] LICHMAN M. UCI machine learning repository[EB/OL]. University of California, Irvine, School of Information and Computer Sciences, 2013. http://archive.ics.uci.edu/ml.

[108] ZHANG H, SONG Y, JIANG B, et al. Two-stage bagging pruning for reducing the ensemble size and improving the classification performance[J]. Math Probl Eng, 2019, 2019.

[109] XIA X, LIN T, CHEN Z. Maximum relevancy maximum complementary based ordered

aggregation for ensemble pruning[J]. Appl Intell, 2018, 48(9): 2568-2579.

[110] CAO J, LI W, MA C, et al. Optimizing multi-sensor deployment via ensemble pruning for wearable activity recognition[J]. Inf Fusion, 2018, 41: 68-79.

[111] AGHAMOLAEI S, FARHADI M, ZARRABI-ZADEH H. Diversity maximization via composable coresets[C]//CCCG. 2015: 43.

[112] INDYK P, MAHABADI S, MAHDIAN M, et al. Composable core-sets for diversity and coverage maximization[C]//PODS. 2014: 100-108. DOI: 10.1145/2594538.2594560.

[113] DEMŠAR J. Statistical comparisons of classifiers over multiple data sets[J]. J Mach Learn Res, 2006, 7(Jan): 1-30.

[114] WIKIPEDIA. Correlation coefficient[EB/OL]. https://en.wikipedia.org/wiki/Correlation_coefficient.

[115] WIKIPEDIA. Pearson correlation coefficient[EB/OL]. https://en.wikipedia.org/wiki/Pearson_correlation_coefficient.

[116] YKHLEF H, BOUCHAFFRA D. An efficient ensemble pruning approach based on simple coalitional games[J]. Inf Fusion, 2017, 34: 28-42.

[117] CHEN H, TIŇO P, YAO X. Predictive ensemble pruning by expectation propagation[J]. IEEE Trans Neural Netw, 2009, 21(7): 999-1013.

[118] MELVILLE P, MOONEY R. Constructing diverse classifier ensembles using artificial training examples[C]//IJCAI. 2003: 505-510.

[119] SOARES R, CHEN H, YAO X. A cluster-based semisupervised ensemble for multiclass classification[J]. IEEE Trans Emerg Top Comput Intell, 2017, 1(6): 408-420.

[120] YULE G U. On the association of attributes in statistics: with illustrations from the material of the childhood society, &c[J]. Philos Trans Royal Soc A, 1900, 194: 257-319.

[121] CHEN H, YAO X. Regularized negative correlation learning for neural network ensembles [J]. IEEE Trans Neural Netw, 2009, 20(12): 1962-1979.

[122] CHEN H, YAO X. Multiobjective neural network ensembles based on regularized negative correlation learning[J]. IEEE Trans Knowl Data Eng, 2010, 22(12): 1738-1751.

[123] ZADEH S, GHADIRI M, MIRROKNI V, et al. Scalable feature selection via distributed diversity maximization[C]//AAAI. 2017: 2876-2883.

[124] AGARWAL P, HAR-PELED S, VARADARAJAN K. Approximating extent measures of points[J]. J ACM, 2004, 51(4): 606-635. DOI: 10.1145/1008731.1008736.

[125] COVER T, THOMAS J. Elements of information theory[M]. John Wiley & Sons, 2012.

[126] SRINIVAS N, DEB K. Muiltiobjective optimization using nondominated sorting in genetic algorithms[J]. Evol Comput, 1994, 2(3): 221-248.

[127] DEB K, PRATAP A, AGARWAL S, et al. A fast and elitist multiobjective genetic algorithm:

Nsga-ii[J]. IEEE Trans Evol Comput, 2002, 6(2): 182-197.

[128] BIRNBAUM B, GOLDMAN K. An improved analysis for a greedy remote-clique algorithm using factor-revealing LPs[J]. Algorithmica, 2009, 55(1): 42-59. DOI: 10.1007/s00453-007-9142-2.

[129] GRAHAM R, KNUTH D, PATASHNIK O. Concrete mathematics: A foundation for computer science[M]. Addison-Wesley Longman Publishing Co., Inc., 2012. DOI: http://www.citeulike.org/group/8352/article/10360334.

[130] MIRROKNI V, ZADIMOGHADDAM M. Randomized composable core-sets for distributed submodular maximization[C]//STOC. 2015: 153-162. DOI: 10.1145/2746539.2746624.

[131] WIKIPEDIA. Speedup[EB/OL]. https://en.wikipedia.org/wiki/Speedup.

[132] ZOPH B, VASUDEVAN V, SHLENS J, et al. Learning transferable architectures for scalable image recognition[C]//CVPR. 2018: 8697-8710.

[133] ELSKEN T, METZEN J H, HUTTER F. Neural architecture search[M]//Automated Machine Learning. Springer, 2019: 63-77.

[134] WISTUBA M, RAWAT A, PEDAPATI T. A survey on neural architecture search[J]. arXiv preprint arXiv:1905.01392, 2019.

[135] ZOPH B, LE Q. Neural architecture search with reinforcement learning[C]//ICLR. 2017.

[136] HUANG F, ASH J, LANGFORD J, et al. Learning deep resnet blocks sequentially using boosting theory[C]//ICML. 2018.

[137] MACKO V, WEILL C, MAZZAWI H, et al. Improving neural architecture search image classifiers via ensemble learning[J]. arXiv preprint arXiv:1903.06236, 2019.

[138] CORTES C, MOHRI M, SYED U. Deep boosting[C]//ICML. 2014: 1179-1187.

[139] CHEN H, YAO X. Evolutionary random neural ensembles based on negative correlation learning[C]//IEEE CEC. IEEE, 2007: 1468-1474.

[140] BIAN Y, WANG Y, YAO Y, et al. Ensemble pruning based on objection maximization with a general distributed framework[J]. IEEE Trans Neural Netw Learn Syst, 2019.

[141] CHEN H, TINO P, YAO X. A probabilistic ensemble pruning algorithm[C]//ICDM Workshops. IEEE, 2006: 878-882.

[142] CHEN H. Diversity and regularization in neural network ensembles[D]. University of Birmingham, 2008.

[143] BIAN Y, CHEN H. When does diversity help generalization in classification ensembles?[J]. arXiv preprint arXiv:1910.13631, 2019.

[144] BAKER B, GUPTA O, NAIK N, et al. Designing neural network architectures using reinforcement learning[C]//ICLR. 2017.

[145] KANDASAMY K, NEISWANGER W, SCHNEIDER J, et al. Neural architecture search

with bayesian optimisation and optimal transport[C]//NeurIPS. 2018: 2020-2029.

[146] CAI H, CHEN T, ZHANG W, et al. Efficient architecture search by network transformation [C]//AAAI. 2018.

[147] LIU H, SIMONYAN K, YANG Y. Darts: Differentiable architecture search[J]. arXiv preprint arXiv:1806.09055, 2018.

[148] ZELA A, KLEIN A, FALKNER S, et al. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search[C]//ICML Workshop on AutoML. 2018.

[149] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[C]//CVPR. 2016: 770-778.

[150] CAI H, YANG J, ZHANG W, et al. Path-level network transformation for efficient architecture search[C]//ICML. 2018.

[151] REAL E, AGGARWAL A, HUANG Y, et al. Regularized evolution for image classifier architecture search[J]. arXiv preprint arXiv:1802.01548, 2018.

[152] ELSKEN T, METZEN J H, HUTTER F. Efficient multi-objective neural architecture search via lamarckian evolution[J]. arXiv preprint arXiv:1804.09081, 2018.

[153] BROCK A, LIM T, RITCHIE J, et al. Smash: One-shot model architecture search through hypernetworks[C]//NIPS Workshop on Meta-Learning,. 2017.

[154] ELSKEN T, METZEN J, HUTTER F. Simple and efficient architecture search for convolutional neural networks[C]//NIPS Workshop on Meta-Learning. 2017.

[155] ZHONG Z, YAN J, WU W, et al. Practical block-wise neural network architecture generation [C]//CVPR. 2018: 2423-2432.

[156] PHAM H, GUAN M, ZOPH B, et al. Efficient neural architecture search via parameter sharing[C]//ICML. 2018.

[157] ZHONG Z, YANG Z, DENG B, et al. Blockqnn: Efficient block-wise neural network architecture generation[J]. arXiv preprint arXiv:1808.05584, 2018.

[158] CHANG J, ZHANG X, GUO Y, et al. Differentiable architecture search with ensemble gumbel-softmax[J]. arXiv preprint arXiv:1905.01786, 2019.

[159] ARDYWIBOWO R, BOLUKI S, GONG X, et al. NADS: Neural architecture distribution search for uncertainty awareness[EB/OL]. 2020. https://openreview.net/forum?id=rJeXDANKwr.

[160] JARRETT K, KAVUKCUOGLU K, RANZATO M, et al. What is the best multi-stage architecture for object recognition?[C]//ICCV. IEEE, 2009: 2146-2153.

[161] NAIR V, HINTON G. Rectified linear units improve restricted boltzmann machines[C]// ICML. 2010: 807-814.

[162] GLOROT X, BORDES A, BENGIO Y. Deep sparse rectifier neural networks[C]//AISTATS.

2011: 315-323.

[163] GOODFELLOW I, BENGIO Y, COURVILLE A, et al. Deep learning: volume 1[M]. MIT press Cambridge, 2016.

[164] KOLTCHINSKII V, PANCHENKO D, et al. Empirical margin distributions and bounding the generalization error of combined classifiers[J]. Ann Stat, 2002, 30(1): 1-50.

[165] DENG J, DONG W, SOCHER R, et al. Imagenet: A large-scale hierarchical image database [C]//CVPR. IEEE, 2009: 248-255.

[166] KRIZHEVSKY A, HINTON G. Learning multiple layers of features from tiny images[R]. Citeseer, 2009.

[167] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient-based learning applied to document recognition[J]. Proc IEEE Inst Electr Electron Eng, 1998, 86(11): 2278-2324.

[168] XIAO H, RASUL K, VOLLGRAF R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms[J]. arXiv preprint arXiv:1708.07747, 2017.

[169] WEILL C, GONZALVO J, KUZNETSOV V, et al. Adanet: Fast and flexible automl with learning guarantees[J/OL]. GitHub repository, 2018. https://github.com/tensorflow/adanet.

致　谢

曾无数次梦想这一天，而今终于走到终点。曾以为我会为之狂喜，到头来发现只余平静。回首来路，这并非坦途，但我终于学会坦然相对，不悔不疑。

感谢我的导师陈欢欢教授，感谢陈老师带领我走上科研道路。在我刚来到科大为研究方向而迷茫时，陈老师帮我指明了研究方向，并在过程中不断给予反馈。感谢陈老师对我的严格要求和宝贵建议，感谢陈老师给我提供宽松的学习环境和充分的学术自由。科研是一段艰难甚至孤独的旅程，日复一日年复一年的修改和完善，可能换来的仍然是拒稿的打击。以前李玉洁师姐说过一句话，“读研呢，有的人收获了文章，有的人收获了能力，还有的人……收获了一个良好的心态。”如果要说读博带给我的最大收获，可能就是这个良好的心态了。天塌下来当被盖，事情一件一件来，慢慢做，也许结果未必好，但是总比不做好。

感谢南方科技大学的袁博副教授，感谢袁老师帮我修改了第一篇论文。现在回头看那篇初稿简直是惨不忍睹，感谢袁老师耐心细致的修改意见，给了我极大的帮助。感谢伯明翰大学的 Peter Tino 教授，感谢他对我的研究工作所提出的宝贵建议。感谢睿企科技的于伟博士和德州农工大学的 Xia Hu 教授，感谢他们在我访学和实习期间给予我的指点和帮助。我还记得那段时间其实是我心态最崩的时候，觉得自己什么都不行，什么都做不好。感谢两位老师的鼓励和支持，让我有信心有勇气坚持努力下去。感谢徐晓飞老师和邢鸿飞老师，感谢她们从学业上对我的指导和帮助。

感谢诸位同窗和同门，张琨、王怡君、吕广奕、姚亚强、李阳、刘晟材、李昌、王琦琦、潘镇和黄振亚，感谢他们在我修改论文期间所提出的建议。感谢师兄师姐们，李小康、高瑞敏、申爱丽、严正龙、钟祎、王英子和陈再毅，感谢他们在我遇到问题时所提出的建议。感谢访学期间的合作者，宋清全、杜梦楠、Tsung-Lin Yang、Diego Martinez Garcia、金海峰、查道琛、谭桥宇和周楷雄，感谢他们在我做研究期间的讨论和建议，感谢王鑫老师和汪波老师对我的指点和帮助。感谢实习期间的小伙伴们，从他们身上我学到了很多工作的方法。

感谢我的朋友们。感谢赵佳宁、梁亚萍和容小惠在我坚持不下去时的鼓励和安慰；感谢牛广豪、黎遇军、张奕聪、张婧姝和王弯弯在我读书期间的帮助；感谢周振坤、王童和袁珂在我申请出国期间的帮助；感谢陈付郴、孙悦和江伟辰在我访学期间的帮助；感谢卢丽华、宋玮、万淼、袁野、马兰堃和刘文轩在我求职期间的帮助。感谢实验室的师兄和同学们，李宗峰、董亚东、陈兵飞、刘佳伟、刘沿、陈彦敏等，感谢他们在学习中对我的帮助。感谢师弟师妹们，宫志晨、杨丹丹、洪骏远、齐琪、何慧敏、吕胜飞、程进、朱嘉润和陈良伟等，感谢他们在

生活中对我的帮助。感谢陌生人海里那些 reaching out 时向我展露善意的人，在那些无着无落的日夜，他们的有益建议曾令我备受鼓舞。

感谢我的家人。感谢我的父母支持我求学至今，感谢他们的理解和支持。尽管我的选择并不都是他们的期望，但是他们仍然尊重我的想法，给我足够的自由，让我天高任鸟飞。

最后感谢读到这里的你，送你一首河图的《致陌生的你》。愿你前行路上有志同道合者相伴，愿你如晦风雨中仍心境清明。世事一场大梦，人生几度秋凉。自大学起竟又是十载光阴。这一路屡遭困顿，我也曾自问：读博这个赌博究竟值不值得，如果早知这么难，我还会选择这条路吗？Ted Chiang 在《你一生的故事》中[①]写道："一瞥之下，过去与未来轰轰然同时并至，我的意识成为长达半个世纪的灰烬，时间未至已成灰。一瞥间五十年诸般纷纭并发眼底，我的余生尽在其中。还有，你的一生。" Louise 从一开始就知道结局，她依然选择踏上那条路。Ted Chiang 在《商人和炼金术士之门》中[②] 写道："无论是过去还是未来，我们都无法改变，只能更深刻地理解它们。" 或许这才是答案，无论知晓与否，过去的已然过去，不必追悔。我们能选择的只有把握现在，做好当下该做的事，问心无愧，然后等待未知的未来。

卞一琚

2020 年 7 月 7 日

于科大西区科技实验楼

[①] *But occasionally I have glimpses when Heptapod B truly reigns, and I experience past and future all at once; my consciousness becomes a half century-long ember burning outside time. I perceive—during those glimpses—that entire epoch as a simultaneity. It's a period encompassing the rest of my life, and the entirety of yours. (Story of Your Life)* 李克勤 译 / *[美]* 特德 • 姜. 你一生的故事. 译林出版社, *2019.*

[②] *Past and future are the same, and we cannot change either, only know them more fully. (The Merchant and the Alchemist's Gate)* 李克勤 译 / *[美]* 特德 • 姜. 呼吸. 译林出版社, *2019.*

在读期间发表的学术论文与取得的研究成果

已发表论文

1. **Yijun Bian** and Huanhuan Chen, "When Does Diversity Help Generalization in Classification Ensembles?" *IEEE Transactions on Cybernetics (TCYB)*, early access, Feb. 26, 2021, doi: 10.1109/TCYB.2021.3053165.

2. **Yijun Bian**, Yijun Wang, Yaqiang Yao, and Huanhuan Chen, "Ensemble Pruning Based on Objection Maximization With a General Distributed Framework," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, vol. 31, no. 9, pp. 3766–3774, Sept 2020.

3. **Yijun Bian**, Qingquan Song, Mengnan Du, Jun Yao, Huanhuan Chen, and Xia Hu, "Subarchitecture Ensemble Pruning in Neural Architecture Search," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, early access, Jun. 18, 2021, doi: 10.1109/TNNLS.2021.3085299.

主要参与的科研项目

1. 国家重点研发计划课题: 知识导航中的交互机理 (2016YFB1000905)

攻读学位期间获得的学术奖励

1. 2020 年 GitHub Arctic Code Vault Contributor (Badge)[①]
2. 2019 年环球数码奖学金
3. 2018 年优秀学生国际交流资助计划 (A 类)

[①] Contributed code to: tensorflow/adanet, thunlp/OpenNE, eustomaqua/PyEnsemble