学校代码：10289
分类号：TP18
密　级：公开
学　号：211210701420

江苏科技大学
专业硕士学位论文
（全日制专业学位）

基于元启发式算法的快速特征选择方法研究

研究生姓名 **张帅帅**　　导师姓名 **徐泰华**

申请学位类别 **电子信息硕士学位**　　学位授予单位 **江苏科技大学**

学科专业 **计算机技术**　　论文提交日期 **2024 年 04 月 16 日**

研究方向 **粒计算、粗糙集**　　论文答辩日期 **2024 年 06 月 01 日**

答辩委员会主席 **韩飞**　　评阅人 **盲审**

盲审

2024 年 04 月 16 日

基于元启发式算法的快速特征选择方法研究
张帅帅
江苏科技大学

分类号：TP18

密　级：公开

学　号：211210701420

电子信息硕士学位论文

基于元启发式算法的快速特征选择方法研究

学生姓名　张帅帅

指导教师　徐泰华　副教授

江苏科技大学

二〇二四年四月

A Thesis Submitted in Fulfillment of the Requirements

for the Degree of Master of Engineering

Research on fast feature selection method based on meta-heuristic algorithm

Submitted by

Shuaishuai Zhang

Supervised by

Associate Professor Taihua Xu

Jiangsu University of Science and Technology

April 2024

学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果．除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果．对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明．本人完全意识到本声明的法律结果由本人承担．

学位论文作者签名：

年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅．本人授权大学可以将本学位论文的全部或部分内容编入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和汇编本学位论文．

本学位论文属于：

(1)保密□，在___年解密后适用本授权书．

(2)不保密□．

学位论文作者签名：　　指导教师签名：

年　月　日　　年　月

摘 要

在大数据和复杂数据集日益普遍的今天，传统的元启发式算法虽然能够寻找全局最优解，但往往面临计算量大、耗时长的问题。特别是在特征选择任务中，随着样本数量增加，算法的性能瓶颈愈发凸显。大多数研究采用近似最优解来妥协耗时问题，但实际上依旧消耗时间过多。针对上述问题，本文提出了一种新的特征选择框架。首先，该框架包含了一个随机采样加速器，它可以缩小算法的搜索样本空间，进而解决时间消耗过大的问题。其次，该框架采用 k-means SMOTE 重采样的方式来弥补随机采样带来的类别不平衡问题。接着，该框架还采用了投票集成策略，通过减少错误分类标签对整体分类结果的影响来提高分类稳定性。最后，该框架使用数据扰动策略来增强分类准确率。本文所提出框架的主要意义在于，它能够在保证元启发式算法分类效率的同时，显著提升其分类效果，从而更好地应对各类分类任务。

本文的研究内容和创新成果主要有以下四点：

一、提出结合元启发式算法和邻域粗糙集理论的新型特征选择方法

针对需要处理连续型数值特征的问题，本研究结合了元启发式算法和邻域粗糙集的优势，提出了一种新的特征选择方法。元启发式算法在理论上能够搜索到全局最优解，从而显著提升了特征选择的准确性。而邻域粗糙集则能够帮助元启发式算法处理连续型数值特征，同时去除不相关和冗余特征，从而增强了算法的可解释性。

二、使用随机采样加速器来加速元启发式算法特征选择过程

针对面对大规模样本数据时，元启发式算法常常遭遇时间消耗过大的问题。本研究引入了随机采样加速器，通过对样本数据进行随机分组来缩小搜索空间的规模。按照分组结果，逐步地对每组进行特征选择，每一步选出的特征被标记并用于加速后续组的特征选择过程，这一方式大大提高特征选择的效率。针对随机采样加速器在选择样本时的随机性可能导致选定样本的类别分布不平衡，从而影响所选特征组的分类性能的问题。本文采用 k-means SMOTE 重采样来弥补随机采样加速器的缺陷，实现样本类别的均衡分布，进而确保所选特征组的高质量，有助于提升分类模型的性能。

三、采用集成策略和数据扰动策略对元启发式算法的结果进行集成

针对本文使用的加速算法提升了分类速度，但是分类性能没有进行提升的问题。鉴于此，引入集成策略的概念，能够更好地对候选特征组进行评估，进而挑选出更为稳健的特征，以利于后续的分类。但是，集成选择器本质上依赖于更多的识别能力来提高分类性能。为了产生更多不同的特征选择结果集，本文采用了数据扰动策略，对原始数据结构进行的邻域重构，产生更丰富的特征选择结果集用以集成。

关键词 特征选择；元启发式算法；粗糙集；数据扰动；随机采样

Abstract

Nowadays, with the increasing popularity of big data and complex data sets, although traditional meta-heuristic algorithms can find the global optimal solution, they often face the problem of large amount of calculation and time-consuming. Especially in the feature selection task, as the number of samples increases, the performance bottleneck of the algorithm becomes more prominent. Most studies use approximate optimal solutions to compromise time-consuming problems, but in fact they still consume too much time. In view of the above problems, this thesis proposes a new feature selection framework. Firstly, the framework includes a random sampling accelerator, which can reduce the search sample space of the algorithm and solve the problem of excessive time consumption. Secondly, the framework uses k-means SMOTE resampling to compensate for the class imbalance problem caused by random sampling. Then, the framework also uses a voting ensemble strategy to improve the classification stability by reducing the impact of misclassification labels on the overall classification results. Finally, the framework uses a data perturbation strategy to enhance the classification accuracy. The main significance of the framework proposed in this paper is that it can significantly improve the classification effect while ensuring the classification efficiency of the meta-heuristic algorithm, so as to better cope with various classification tasks.

The research content and innovation achievements of this thesis mainly cover the following three points:

1. A new feature selection method combining meta-heuristic algorithm and neighborhood rough set theory is proposed.

Aiming at the problem of dealing with continuous numerical features, this study combines the advantages of meta-heuristic algorithm and neighborhood rough set, and proposes a new feature selection method. The meta-heuristic algorithm can theoretically search for the global optimal solution, which significantly improves the accuracy of feature selection. The neighborhood rough set can help the meta-heuristic algorithm to deal with continuous numerical features while removing irrelevant and redundant features, thereby enhancing the interpretability of the algorithm.

2. The random sampling accelerator is used to accelerate the feature selection process of meta-heuristic algorithm.

In the face of large-scale sample data, meta-heuristic algorithms often encounter the problem of excessive time consumption. In this study, a random sampling accelerator is introduced to reduce the size of the search space by randomly grouping the sample data. According to the grouping results, feature selection is performed on each group step by step. The features selected at each step are marked and used to accelerate the feature selection process of the subsequent groups, which greatly improves the efficiency of feature selection. The randomness of the random sampling accelerator in selecting samples may lead to the imbalance of the class distribution of the selected samples, thus affecting the classification performance of the selected feature group. In this paper, k-means SMOTE resampling is used to make up for the defects of random sampling accelerator, to achieve the balanced distribution of sample categories, and to ensure the high quality of the selected feature groups, which is helpful to improve the performance of the classification model.

3. The results of the meta-heuristic algorithm are integrated by the ensemble strategy and the data perturbation strategy.

The acceleration algorithm in this paper improves the classification speed, but the classification performance is not improved. In view of this, the concept of integration strategy is introduced, which can better evaluate the candidate feature groups, and then select more robust features for subsequent classification. However, the ensemble selector essentially relies on more recognition capabilities to improve classification performance. In order to generate more different feature selection result sets, this thesis uses the data perturbation strategy to reconstruct the neighborhood of the original data structure and generate richer feature selection result sets for ensemble.

Keywords Feature Selection; Meta-heuristic Algorithm; Rough Set; Data Perturbation; Random Sampling

目 录

摘 要 I
Abstract III
第 1 章 绪 论 1
1.1 研究背景与意义 1
1.2 国内外研究现状 2
1.3 存在问题与挑战 3
1.4 研究内容和组织结构 4
第 2 章 相关基础知识 7
2.1 二元关系 7
2.2 不确定性度量 8
2.3 适应度函数 8
2.4 特征选择 9
2.5 K-means 算法 10
2.6 SMOTE 算法 11
第 3 章 基于邻域粗糙集的元启发式算法 13
3.1 问题描述 13
3.2 本章工作 13
3.2.1 基于邻域粗糙集的帝王蝶优化算法 13
3.2.2 基于邻域粗糙集的遗传算法 15
3.2.3 基于邻域粗糙集的人工鱼群算法 15
3.2.4 基于邻域粗糙集的森林优化算法 19
3.3 实验分析 21
3.3.1 实验数据 21
3.3.2 实验设置 21
3.3.3 实验结果 22
3.4 本章小结 26
第 4 章 基于随机采样加速器和重采样的快速元启发式算法 27
4.1 问题描述 27
4.2 本章工作 29
4.2.1 随机采样加速器 29
4.2.2 K-means Smote 方法 30
4.2.3 基于随机采样加速器的快速元启发式算法 30
4.2.4 基于随机采样加速器和重采样的快速元启发式算法 32
4.3 实验分析 33
4.3.1 实验数据 33
4.3.2 实验设置 34
4.3.3 实验结果 34
4.4 本章小结 45
第 5 章 基于采样加速和扰动集成的快速元启发式算法 47
5.1 问题描述 47
5.2 相关知识与工作 47

5.2.1 集成选择器 47
5.2.2 数据扰动策略 48
5.3 基于采样加速和扰动集成的快速元启发式算法 49
5.4 实验分析 50
5.4.1 实验数据 50
5.4.2 实验设置 51
5.4.3 实验结果 52
5.5 本章小结 61
总结与展望 63
参考文献 65
攻读硕士学位期间取得的研究成果 71
致　谢 73

Contents

Chinese Abstract I
Abstract III
Chapter 1 Preface 1
1.1 Research Background and Significance 1
1.2 Current Research Situation at Home and Abroad 2
1.3 Problems and Challenges 3
1.4 Work and Structure 4
Chapter 2 Relevant Basic Knowledge 7
2.1 Binary Relationship 7
2.2 Uncertainty Measure 8
2.3 Fitness Function 8
2.4 Feature Selection 9
2.5 K-means Algorithm 10
2.6 SMOTE Algorithm 11
Chapter 3 Meta-heuristic Algorithm Based on Neighborhood Rough Set 13
3.1 Problem Description 13
3.2 Detailed Work 13
3.2.1 MBO Based on Neighborhood Rough Set 13
3.2.2 GA Based on Neighborhood Rough Set 15
3.2.3 AFSA Based on NeighborhoRough Set 15
3.2.4 FOA Based on Neighborhood Rough Set 19
3.3 Experiment Analysis 21
3.3.1 Experiment Data Sets 21
3.3.2 Experiment Settings 21
3.3.3 Experiment Results 22
3.4 Summary 26
Chapter 4 Fast Meta-heuristic Algorithm Based on KRS 27
4.1 Problem Description 27
4.2 Detailed Work 29
4.2.1 Random Sampling Accelerator 29
4.2.2 K-means Smote Method 30
4.2.3 Fast Meta-heuristic Algorithm Based on RS 30
4.2.4 Fast Meta-heuristic Algorithm Based on KRS 32
4.3 Experiment Analysis 33
4.3.1 Experiment Data Sets 33
4.3.2 Experiment Settings 34
4.3.3 Experiment Results 34
4.4 Summary 45
Chapter 5 Fast Meta-heuristic Algorithm Based on REP 47
5.1 Problem Description 47
5.2 Detailed Work 47
5.2.1 Ensemble Selector 47

5.2.2 Data Perturbation Strategy 48
5.3 Fast Meta-heuristic Algorithm Based on REP 49
5.4 Experiment Analysis 50
5.4.1 Experiment Data Sets 50
5.4.2 Experiment Settings 51
5.4.3 Experiment Results 52
5.5 Summary 61
Conclusions and Future Perspectives 63
References 65
Research Projects in the Period of Graduate 71
Acknowledgments 73

第 1 章 绪 论

1.1 研究背景与意义

自古以来，数据在人类社会的发展中始终扮演着举足轻重的角色[1]。从最初的结绳记事到文字的发明，数据的形式与载体不断演变，逐渐成为了我们认识世界、解读信息的基石。然而，尽管数据的重要性日益凸显，但人类对其处理的能力一直受限于技术发展的水平。直到电子计算机等现代信息技术出现，人类才在数据处理方面实现了质的飞跃。这些技术的飞速进步不仅极大地提升了数据收集和分析的速度与效率，更使得数据成为了继物质和能源之后的又一重要战略资源，深刻影响着人类社会的各个方面。

随着时间的推进，计算能力的增强和复杂算法的发展使得数据量急剧增加，促进科技的快速发展和商业模式的根本变革。牛津大学教授维克托·迈尔-舍恩伯格在他的著作《大数据时代》中提出，数据分析已从依赖随机样本、寻求精确解和强调因果关系的传统方式，转变为使用全部数据、接近解决方案和关注数据间关联的新模式。MGI 在其报告《分析的时代：在大数据的世界竞争》中也指出，大数据分析在多个领域显示出巨大的增长潜力。

此外，大数据在推动国家治理体系和治理能力现代化的进程中扮演着举足轻重的角色，成为社会治理和政府决策的关键支撑力量。习近平总书记[2]多次强调大数据的重要性，将其视为工业社会的“自由”资源，指出掌握数据的一方即掌握了发展的主动权，并特别强调了信息资源对国家软实力和国际竞争力的巨大影响。在数字经济领域，中国信息通信研究院的报告揭示，数字经济已成为多国 GDP 的主导力量，同时，各国际组织也在积极推动全球数字经济的蓬勃发展。

决策过程通常包括三个关键步骤：认知现状、预测未来以及选择策略[3]。这一过程与大数据分析所包含的三种类型相契合。计算机最基础的任务便是将相关的信息和知识有效地展现给人类专家。然而，随着数据结构的日益复杂和特征数量的急剧增加，从海量的数据中提炼出问题的本质变得愈发困难。特征选择作为一种高效的数据预处理技术，通过筛选出能够代表整体特征的特征组合，有效降低了数据的维度。这一技术已被广泛应用于粗糙集、粒计算、模式识别和机器学习等领域，极大地推动了这些领域数据处理模块的快速发展[4-8]。

然而，面对基因数据、人脸数据、遥感数据等高维数据的挑战，如何提升特征选择的效率成为了一个亟待解决的问题。这一问题的解决对于实际应用具有重大的意义，有助于我们更好地应对大数据时代的挑战[3]，推动相关领域的技术进步和应用发展。

1.2 国内外研究现状

特征选择作为当前数据预处理领域的重要研究方向，在数据挖掘和模式识别等应用中发挥着至关重要的作用[4-8]。随着数据量的不断增长和特征维度的不断扩展，特征选择技术变得愈发关键。通过选择相关性最强、最显著的特征子集，特征选择不仅能够显著节省存储空间[9-10]，降低计算成本，还能有效增强学习模型的泛化能力和准确性[11-12]。在特征选择的研究过程中，通常有两种主要策略：穷举搜索和启发式搜索。穷举搜索方法通过遍历所有可能的特征组合来寻找最优特征子集[13-16]，它在特征选择研究的早期阶段得到了蓬勃发展。例如，基于判别矩阵[17-18]的方法就是一种典型的穷举搜索策略，它通过计算特征与输出之间的相关性或差异性来评估特征的重要性。然而，随着数据量的不断增长，穷举搜索方法面临着计算量大、时间消耗长等挑战，尤其是在处理大规模数据集时，其效率变得难以接受[19-20]。为了有效应对这一挑战，研究者们开始转向启发式搜索策略，以期在可承受的时间范围内找到相对较优的特征子集。启发式搜索通过引入启发式信息或规则来指导搜索过程，从而加快搜索速度并提高搜索效率。在特征选择领域，元启发式算法因其全局搜索能力和自适应性强的特点，成为了解决这一难题的有力工具。

元启发式算法是一类基于自然现象的优化算法，它们通过模拟自然界中的某些现象或规律来进行问题求解。在特征选择中，元启发式算法可以根据特定的启发式信息或规则，在搜索空间中生成和选择候选特征子集，并通过迭代优化来逐渐逼近最优解。这些算法具有全局搜索能力强、易于实现和扩展性好等优点，因此在特征选择领域得到了广泛应用。近年来，研究者们提出了许多基于元启发式算法的特征选择方法。例如，Ritam 等人[21]提出了一种将离散均衡优化器与模拟退火算法相结合的特征选择方法，通过引入离散均衡优化器的机制来改进模拟退火算法的搜索性能。Elaziz 等人[22]则提出了一种基于原子轨道搜索的元启发式算法，通过模拟原子在轨道上的运动来指导特征选择过程。此外，还有蚁群优化算法[23]、人工鱼群算法[24]、帝王蝶优化算法[25]、遗传算法[26]和森林优化算法[27]等多种元启发式算法被用于优化特征选择过程，取得了显著的效果。

粗糙集理论（Rough Set Theory）[28-34]作为一种数学工具，自从其被提出以来，就以其处理不确定性、模糊性和不完整性数据的能力，在多个领域尤其是特征选择领域展现出了显著的优势。特征选择是机器学习、数据挖掘等领域中一个重要的步骤，它的目的是从原始特征集合中选取出与目标变量最相关、信息含量最高的特征子集，以减少计算成本、提高模型性能，并增强模型的可解释性。在特征选择的研究中，粗糙集理论提供了一种基于数据本身特性的特征选择方法，不需要任何先验知识或额外的参数设置。通过将粗糙集理论与元启发式算法结合，可以有效地处理特征空间中的高

维数据和复杂关系，自动寻找出最优或接近最优的特征子集。例如，Penmatsa 等人[23]将蚁群优化和粗糙集理论相结合来简化和优化恶意软件检测系统的特征。Luan 等人[24]基于改进的人工鱼群算法和粗糙集理论提出了一种新的特征选择算法。这种结合方式不仅提升了特征选择的效率和准确性，而且通过减少冗余特征和不相关特征，提高了模型的泛化能力和鲁棒性。

然而，在实际应用中，数据集中往往包含大量的数值特征，这些特征可能具有连续取值范围，而经典的粗糙集理论主要适用于处理离散特征。为了克服这一局限性，邻域粗糙集理论应运而生。邻域粗糙集理论通过在原始数据空间中定义邻域关系，将连续特征映射到离散的邻域空间，从而扩展了粗糙集理论的应用范围。这种扩展使得粗糙集理论能够直接处理包含连续特征和离散特征的混合数据集，进一步增强了其在实际应用中的灵活性和实用性。近年来，随着数据挖掘和机器学习技术的快速发展，基于邻域粗糙集的元启发式特征选择算法已成为该领域的研究热点之一。这些算法通过结合邻域粗糙集理论和元启发式算法的优点，能够在复杂的特征空间中快速找到最优或接近最优的特征子集，为数据分类、聚类、回归等任务提供有力的支持。例如，Zou 等人[41]提出的基于人工鱼群算法和邻域粗糙集理论的特征选择算法，通过自适应函数控制人工鱼的视野和步长，实现了对连续特征和离散特征的统一处理；Feng 等[42]提出的基于粗糙邻域集和改进粒子群优化的特征选择方法，通过改进粒子群优化算法中的更新策略，提高了算法的稳定性和收敛速度；Ahmed 等人[43]提出的基于混合鲸鱼优化算法的特征选择方法，则通过引入鲸鱼优化算法的全局搜索能力和局部搜索能力，实现了对阿拉伯手写字符数据集中特征的有效选择。综上所述，基于邻域粗糙集的元启发式特征选择算法是一种高效、灵活且实用的数据分类策略，它能够在复杂的特征空间中快速找到最优或接近最优的特征子集，为数据挖掘和机器学习等领域的研究和应用提供有力的支持。随着技术的不断发展和研究的深入，相信这些算法将在未来发挥更加重要的作用。

1.3 存在问题与挑战

(1) 在处理大样本数据时，算法的时间消耗过大：首先，在处理大规模样本数据时，元启发式算法在分类任务中的时间消耗问题变得尤为突出。这主要是因为算法在每一代迭代过程中都需要对大量的解决方案进行适应度评估。随着样本量的增加，评估过程涉及的数据点数量急剧上升，进而显著延长了算法的运行时间。由于元启发式算法需要反复迭代以逐步逼近最优解，因此即使每次迭代的时间增加幅度较小，也会导致整体时间消耗显著增长。因此，优化算法以提高其处理大样本数据时的效率成为了一个亟待解决的问题。

(2) 类别不平衡造成分类性能下降：其次，我们要讨论的是元启发式算法在分类任务中面临的分类性能问题。在尝试通过加速算法提升元启发式算法效率的过程中，我们注意到数据集中存在的类别不平衡现象，即不同类别的样本数量差异显著。这种不平衡可能导致算法在执行分类任务时产生倾向性，过度关注样本量较大的类别，而忽视了样本量较少的类别。这种倾向性不仅会降低模型对少数类别的预测性能，还可能影响整体分类的准确度和鲁棒性。因此，在使用元启发式算法进行分类时，我们必须采取有效的策略来应对类别不平衡问题，确保模型能够全面、均衡地学习并分类所有类别的样本，从而巩固和提升分类性能。

(3) 分类性能表现不佳：最后，尽管元启发式算法在多种场景下均表现出良好的性能，但在某些特定的分类任务中，其分类准确性和稳定性仍有待进一步提升。一方面，由于解空间的复杂性和可能存在多个局部最优解的情况，元启发式算法在搜索全局最优解的过程中可能会陷入局部最优，这尤其在分类界限模糊或数据特征复杂多变的场景中表现得尤为明显，从而影响了分类的准确性。另一方面，元启发式算法的分类稳定性也是一个需要重视的问题。这类算法往往通过反复迭代在训练数据上进行性能优化，如果缺乏有效的控制和调节机制，就可能导致过拟合现象的发生，即模型在训练集上表现优秀，但对新数据的泛化能力较弱。因此，构建一个集成框架[44-47]来提升元启发式算法在分类任务上的性能变得尤为迫切。这不仅要求设计一个能够支持算法平滑整合的通用接口或标准，还需要解决不同算法之间的兼容性问题。由此可见，开发一个适合元启发式算法的集成框架是一项充满挑战且具有重要意义的工作。

1.4 研究内容和组织结构

本文的整体目标是面向高维海量数据，通过新提出的元启发式算法框架，降低进行特征选择的时间消耗，并且有效提升算法的分类性能。

全文总共分为五章，论文的组织结构如下所示：

第一章为绪论，主要介绍了研究背景与意义、国内外研究现状、存在问题与挑战。

第二章阐述了本文的相关基础知识。包括二元关系、不确定性度量、适应度函数、特征选择、k-means 算法和 SMOTE 算法。

第三章是本文的基础研究内容，首先，本文阐明了为何选择将邻域粗糙集与元启发式算法相结合，其关键原因在于邻域粗糙集能够有效处理连续型数据，从而提升元启发式算法在复杂数据处理方面的能力。随后，通过精心设计的适应度函数，本文将

邻域粗糙集与元启发式算法进行有机整合，创新性地提出了四种算法：NMBO、NGA、NAFSA 和 NFOA。这些算法不仅继承了元启发式算法的优化特性，还融入了邻域粗糙集处理连续数据的能力，为数据特征选择提供了新的思路和方法。最后，为了深入剖析这四种基于邻域粗糙集的元启发式算法的性能特点、差异以及各自的优劣势，本文进行了一系列严谨的验证性实验。实验中，我们选择了多个广泛使用的数据集，并将这些新算法与前向贪心算法（FGS）进行了对比。通过对比分析，我们可以更全面地了解这些算法在特征选择方面的效果，为后续的研究提供了坚实的实验基础。值得一提的是，这四种基于邻域粗糙集的元启发式算法不仅是本文的创新成果，更构成了后续对比研究的核心基础。

第四章是本文的核心重点研究内容，其聚焦于随机采样加速器和 k-means Smote 方法的应用与整合。首先，我们介绍了随机采样加速器的概念，它是一种有效的数据处理技术，能够显著提高计算效率和数据分析速度。紧接着，我们介绍了 k-means Smote 方法，它是一种专门设计用于解决类别不平衡问题的技术，通过合成少数类样本来平衡数据集。在随后的研究中，我们将随机采样加速器与第二章所介绍的元启发式算法相结合，创新性地提出了 RS 算法。这一算法结合了随机采样加速器的速度和元启发式算法的优化能力，旨在提高特征选择的效率和准确性。为了弥补随机采样加速器可能带来的类别不平衡问题，我们进一步将 k-means Smote 方法引入到 RS 算法中，形成了 KRS 算法。这一算法不仅继承了 RS 算法的优点，还通过 k-means Smote 方法有效平衡了数据集中的类别分布，从而提高了分类器的性能。最后，在本文中，我们详细描述了 RS 算法和 KRS 算法的设计步骤和实现过程，并通过实验对比分析了这两种算法与上述四个元启发式特征选择算法的性能。实验结果表明，RS 算法和 KRS 算法在特征选择方面均表现出色，尤其是 KRS 算法在解决类别不平衡问题方面取得了显著成效。

第五章在第四章的基础上继续深入探讨如何提升元启发式算法的分类性能。第三章的实验结果虽然表明 k-means Smote 技术缓解了类别不平衡的问题，但在对比分析中发现，与其对应的元启发式算法相比，KRS 算法的性能提升并不显著。为了解决这一矛盾并进一步提高性能，本文提出了一种基于数据扰动的集成策略。首先介绍集成选择器和数据扰动的概念，接着将其运用到第三章所提的算法中，新的算法框架命名为 REP。最后给出了算法框架设计的各个步骤和描述，并与上述 4 个元启发式特征选择算法以及集成框架 ES[47]进行实验对比分析，得出了预期的实验结果与结论。实验结果表明，REP 算法框架在分类性能上取得了显著的提升，验证了基于数据扰动的集成策略的有效性。图 1.1 直观地展示了本文各章的研究内容。

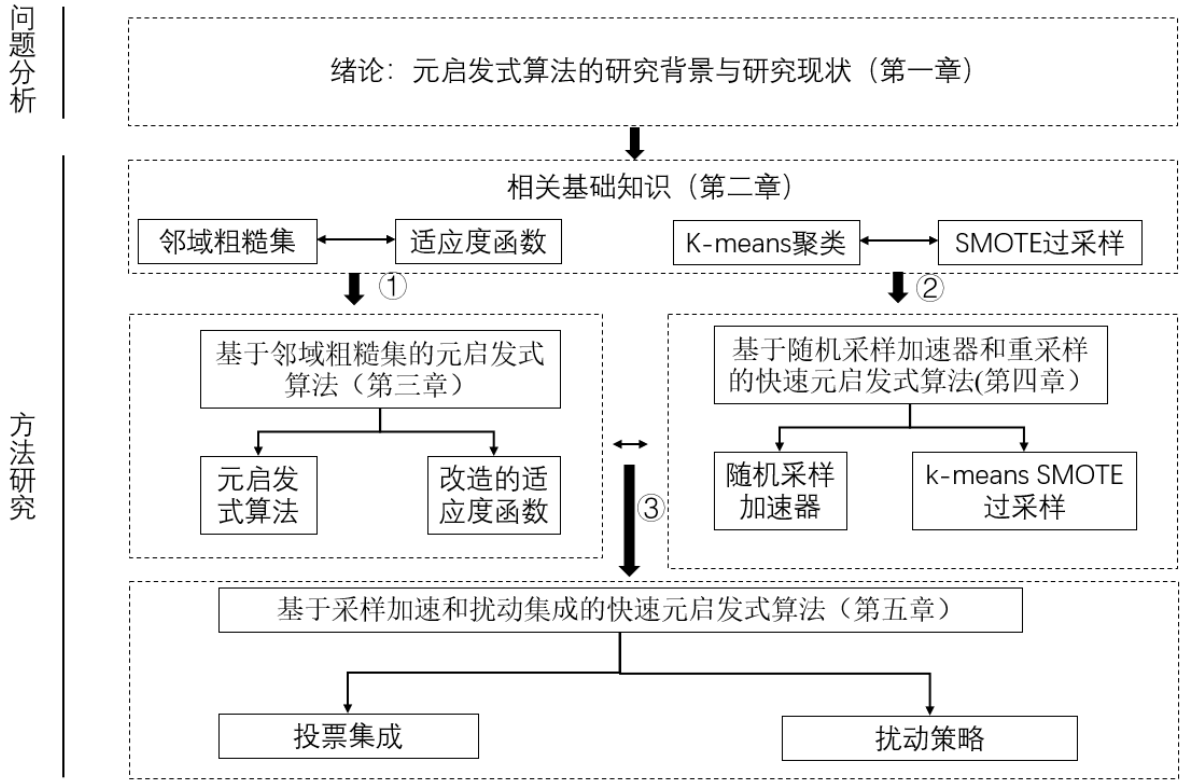

图 1.1 本文各章的研究内容

Fig.1.1 The content of each chapter of this paper is studied

相关说明：

①表示第三章采用邻域粗糙集来解决元启发式算法处理连续型数值特征的问题。此时，产生的新问题是算法的执行效率需要提升。

②表示第四章采用随机采样加速器提升了第三章算法的执行效率，并通过重采样技术弥补其带来的类别不平衡的缺陷。此时，产生的新问题是算法的分类效果需要进一步提升。

③表示第五章在前两章的基础上主要通过扰动策略和投票集成策略提升算法的分类效果。最终，本文提出的框架能够在保证元启发式算法分类效率的同时，显著提升其分类效果，从而更好地应对各类分类任务。

第 2 章 相关基础知识

2.1 二元关系

在粗糙集相关理论[48]中，一个决策系统可以被形式化地表示为二元组 $\mathrm{DS}=<U, AT \cup d>$，其中 U 表示所有样本的非空有限集合，即论域，AT 表示条件特征集，d 表示决策特征。相应地，$\forall x \in U$ 和 $\forall a \in AT$，$a(x)$表示样本 x 在条件 a 上的值，$d(x)$表示样本 x 在决策特征 d 上的值。

定义 2.1[48] 给定一个决策系统 $\mathrm{DS}=<U, AT \cup d>$，$\forall A \subseteq AT$，根据条件特征集 A 诱导出的等价关系可以被描述为：

$$IND_A(U,a)=\{(x,y)\in U\times U:\forall a\in A, a(x)=a(y)\}. \tag{2.1}$$

从粒计算的角度而言，根据条件特征 a 所划分等价类的过程就是信息粒化[55]。特别地，可以根据决策特征 d，论域 U 可以被划分成不同的等价类，即 $U/\mathrm{IND}(U,d)=\{X_1, X_2, ..., X_q\}$。其中，$\forall X_p \in U/\mathrm{IND}(U,d)$，$X_p$ 表示具有相同标签的样本构成的第 p 个决策类。此外，为应对复杂数据的处理挑战，研究者们还探索了其他粒化策略。比如，有的研究者提出了基于聚类技术的信息粒度[56]，有的则聚焦于模糊理论，开发了基于模糊技术的信息粒度[57]，还有的研究着眼于邻域关系，创建了基于邻域技术的信息粒度[58]。这些策略都为复杂数据的处理提供了新的视角和解决方案。

随着信息技术的飞速发展，数据类型逐渐变得丰富多样，特别是连续型与混合型数据的涌现，它们已逐渐成为信息处理领域的重要组成部分。然而，传统的粗糙集理论，其核心在于处理离散型数据，其等价关系的定义在连续或混合数据上并不适用。这一局限促使研究者们如 Hu 等[50]开始探索理论的扩展与革新。他们创新性地引入了邻域关系的概念，对粗糙集理论进行了有益的改进与补充。这一创新不仅拓展了粗糙集理论的应用范畴，使其更加灵活多变，也使其能够更好地适应现代信息技术的快速发展，实现对更广泛数据类型的有效处理。

定义 2.2[49] 给定一个决策系统 $\mathrm{DS}=<U, AT \cup d>$，$\forall A \subseteq AT$，$\delta$ 为半径，根据条件特征集 A 诱导出的邻域关系可以被描述为：

$$N_A^{\delta}=\{(x,y)\in U\times U:\Delta_A(x,y)\le\delta\}, \tag{2.2}$$

其中，$\Delta_A(x,y)$ 是计算样本 x 与样本 y 的距离函数。并且 $\forall x,y,z\in U$，该距离函数满足以下四点：

(1) $\Delta_A(x,y)\geq 0$;

(2) $\Delta_A(x,y)=0$ 当且仅当 $x=y$;

(3) $\Delta_A(x,y)=\Delta_A(y,x)$;

(4) $\Delta_A(x,z)\leq \Delta_A(x,y)+\Delta_A(y,z)$.

根据定义 2.2 所求的邻域关系，样本 x 关于 A 的 δ 邻域可以被描述为：

$$N_A^{\delta}(x)=\{y\in U:\Delta_A(x,y)\leq\delta\}. \tag{2.3}$$

定义 2.3[50] 给定一个决策系统 $\mathrm{DS}=<U,AT\cup d>$，$\forall A\subseteq AT$，δ 为半径，$\forall X\in U/IND(D)$，X_i 的 δ 邻域上、下近似集分别定义为：

$$\overline{N_A^{\delta}}(X_i)=\{x\in U\mid \delta_A(x)\bigcap X_i\neq\varnothing\}, \tag{2.4}$$

$$\underline{N_A^{\delta}}(X_i)=\{x\in U\mid \delta_A(x)\subseteq X_i\}. \tag{2.5}$$

2.2 不确定性度量

量子力学的基石之一在于不确定性，这种不确定性在诸如坐标与动量、时间与能量等多个层面均有所体现。由于在不同时间点上，动量和能量的测量结果可能会产生差异，因此，科学家们一直在寻求更为精确的数学工具来刻画这种不确定性。近年来，粗糙集理论作为计算智能领域的一个新兴分支，其在处理信息的不完整性和不确定性方面展现出了巨大的潜力。为了更加系统地描述这种不确定性，众多学者和专家基于邻域关系和下近似集提出了多种度量标准及性质。这些度量方法不仅有助于我们更深入地理解不确定性的本质，还为我们提供了一套有效的数学框架，用于精确地量化和处理不确定性。本文主要采用的度量是近似质量，如定义 1.4 所示：

定义 2.4[59] 给定一个决策系统 $\mathrm{DS}=<U,AT\cup d>$，$\forall A\subseteq AT$，δ 为半径，决策特征 d 相对于 A 的近似质量(AQ)可以被描述为：

$$AQ^{\delta}(A,d)=\frac{|\underline{N_A^{\delta}}(d)|}{|U|}, \tag{2.6}$$

其中，$|X|$表示 X 的基数。

近似质量用于反映条件特征集 A 对决策特征 d 的逼近能力。近似质量的取值范围为[0,1]，近似质量的值越大，表明条件特征 A 的逼近能力越强。

2.3 适应度函数

根据邻域关系，可以定义一个适应度函数，如定义 2.5 所示。适应度函数是最常

用的启发式信息类型之一，可以指导特征选择过程中元启发式算法的迭代优化方向。

定义 2.5[25] 给定一个邻域决策系统 DS 和邻域半径δ，$\forall A \subset AT$，决策特征 d, 则基于邻域依赖度的新的适应度函数表示为：

$$fitness=\lambda AQ^{\delta}(A,d)+(1-\lambda)\frac{|A|}{|AT|}. \tag{2.7}$$

2.4 特征选择

特征选择是从大规模数据中挑选对分类结果具有最大价值的特征的过程。这是机器学习中极为重要的数据预处理任务之一。在没有进行特征选择的情况下，处理大规模数据会消耗大量时间和 CPU 性能。因此，特征选择的主要任务之一是依据特定准则从数据中筛选出符合条件的特征。例如，可以识别出一个或多个最重要的特征，并在评估后添加到特征选择候选者中[60-63]。本文主要研究分类问题中基于元启发式的特征选择算法。特征选择的一般形式可以抽象为定义 2.6 [64]。

定义 2.6[64] 给定一个邻域决策系统$\mathrm{DS}=<U, AT\cup d>$，$\forall A \subset AT$，在论域$U$ 上与度量有关的一个约束记为$\rho(U, AT)$. $\forall A \subset AT$，当且仅当下列条件成立时，A 可称为合格的特征子集：

(1) A 满足约束$\rho(U, AT)$;

(2) $\forall A' \subset A$不满足约束$\rho(U, AT)$.

鉴于特征选择的定义，如何有效地求解合格的特征子集成为了当下的研究焦点。特征选择算法通常可分为穷举法和启发式搜索两大类。然而，穷举法因涉及 NP-hard 问题而显得不切实际，它要求列出所有可能的特征组合以寻找全局最优解，这无疑会导致巨大的时间开销。相对而言，启发式算法，特别是前向贪心搜索策略，因其较低的时间复杂度而备受关注。这种策略的核心机制在于逐步评估候选特征，并选择合适的特征加入结果集合，直至满足特定的约束条件，具体过程详见算法 2.1。

算法 2.1 前向贪心搜索策略 (FGS)

输入： 决策系统$\mathrm{DS}=<U, AT\cup d>$,
约束条件C_{ρ}^{U}.

输出： 特征选择结果集A.

步骤 1: $A \leftarrow \varnothing$;

步骤 2: 计算$\rho(U, AT)=\odot_{x\in U}\rho(x, AT)$;

步骤 3: **Repeat**

(3.1) $\forall a \in AT-A$，计算$\rho(U, A\cup\{a\})$;

(3.2) 选择一个合适的特征 $b \in AT - A$；

(3.3) $A = A \cup \{b\}$；

(3.4) 计算 $\rho(U, A)$；

Until 约束条件 C_{ρ}^{U} 被满足；

步骤 4: **Repeat**

(4.1) $\forall c \in A$，计算 $\rho(U, A-\{c\})$；

(4.2) **If** 约束条件满足

$A = A-\{c\}$；

End

Until A 没有发生改变或者$|A|$=1；

步骤 5: 输出 A.

经过对算法 2.1 的深入分析，我们发现使用前向贪心搜索策略进行特征选择主要包含两大关键环节。首先是逐步将符合要求的特征纳入特征结果集合中，直至满足预设的约束条件。紧接着，则是对特征选择集合进行精炼，剔除其中的冗余或不相关特征。这两个环节分别对应于特征选择定义中的两个核心约束条件，从而确保所选特征集既全面又精简。

通常来说，在最差的情况下，每个特征都是必要的，需要被加入特征选择结果的集合中。因此，评估特征的次数为 $(|AT|+1)\times|AT|/2$。因此，该算法的时间复杂度为 $O(|U|^2 \times[(|AT|+1)\times|AT|]/2)$，即 $O(|U|^2 \times |AT|^2)$。

然而，前向贪心算法主要着眼于寻找局部最优解，当遇到噪声数据时，这种算法在分类稳定性方面表现不佳。相比之下，元启发式算法因其能够探索并可能获得全局最优解，因此在特征选择任务中得到了广泛的应用。本文的第二章将主要围绕在噪声数据背景下，对比元启发式算法与前向贪心算法的优劣。

2.5 K-means 算法

K-means[74]是一种非常流行的聚类算法，用于将数据点划分为 k 个不同的簇，以便将具有相似特征的数据点归入同一个簇中。这种算法的核心思想是通过迭代找到 k 个簇的中心（即质心），使得每个数据点到其最近的质心的距离之和最小化，从而达到将数据分组的目的。K-means 算法的基本步骤如下：

(1) 初始化：随机选择 k 个数据点作为初始质心。

(2) 分配：将每个数据点分配给最近的质心，从而形成 k 个簇。

(3) 更新：计算每个簇中所有点的均值，并将该均值设置为新的质心。

(4) 重复：重复步骤 2 和 3，直到质心不再发生变化，或者达到某个预定的迭代次

数，此时算法结束。

K-means 算法因其直观易行和高效计算的特性，在处理大型数据集时表现出色，因此本文选择它作为主要的聚类方法。然而，任何算法都有其局限性，k-means 算法也不例外。其中，最为显著的问题在于对初始质心选择的敏感性，即不同的初始化可能导致截然不同的簇划分结果，这使得算法的稳定性受到挑战。此外，k-means 算法还基于簇呈现凸形且大小相近的假设，这种假设在复杂的数据分布中可能并不成立，从而影响了聚类的准确性和适用性。

另一个值得注意的挑战在于，k-means 算法需要预先设定簇的数量 k。然而，在实际应用中，确定合适的 k 值往往是一个难题，因为它依赖于对数据分布和结构的深入理解。幸运的是，在本文所关注的带有决策类的数据集中，簇的数量 k 可以自然地与决策类的种类数相对应。这一特性使得我们能够更为准确地设定 k 值，从而提高 K-means 算法的聚类效果和实用性。

综上所述，尽管 K-means 算法存在一些局限性，但其在处理大型数据集时的高效性和直观性仍使其成为本文的首选聚类方法。通过结合数据的特点和算法的特性，我们能够更好地应用 K-means 算法，实现有效的数据聚类和分析。图 2.1 直观地展示了 k-means 算法的迭代过程，进一步加深了对该算法的理解。

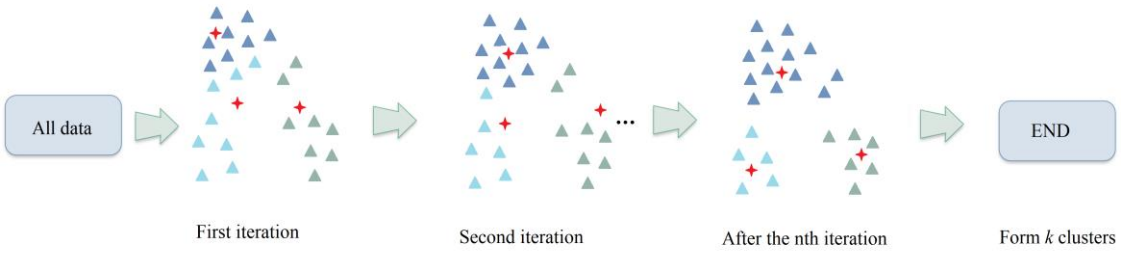

图 2.1 k-means 算法的迭代过程

Fig.2.1 The iterative process of k-means algorithm

2.6 SMOTE 算法

SMOTE（Synthetic Minority Over-sampling Technique）算法[72]是一种过采样技术，用于处理不平衡数据集中的少数类问题。不同于简单的过采样方法（如随机复制少数类样本），SMOTE 算法通过线性插值技术，在少数类样本之间插入新合成的样本，从而增加少数类的样本数量。这一方法旨在解决分类问题中普遍存在的类别不平衡问题，进而改善分类器的性能。具体而言，如图 2.2 所示，SMOTE 算法首先随机选择一个少数类样本，然后找到其邻近的少数类样本。随后，通过线性插值的方式，在这两

个样本之间生成新的合成样本。这些新生成的样本被添加到训练集中，从而提高了少数类在数据集中的比例。通过这种方式，SMOTE 算法有助于分类器更好地学习少数类的特征，进而提高分类的准确性和稳定性。因此，在处理类别不平衡问题时，SMOTE 算法是一种有效的解决方案。具体来说，SMOTE 算法生成合成样本的过程包括三个主要步骤：首先，随机选择一个少数类样本 $\vec{a}$。然后，在 $\vec{a}$ 的 k 个最近的少数类邻居中，随机选择一个邻居样本 $\vec{b}$。最终，通过对 $\vec{a}$ 和 $\vec{b}$ 进行随机线性插值来产生新的样本 $\vec{x}$，计算公式为：$\vec{x} = \vec{a} + w \times (\vec{b} - \vec{a})$，其中 w 是一个在[0, 1]区间内的随机权重。这个过程有效地在样本 $\vec{a}$ 和 $\vec{b}$ 之间创建了一个新的合成样本。

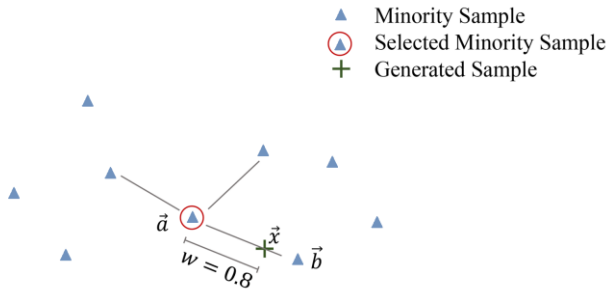

图 2.2 SMOTE 对随机选择的少数类样本和它的一个 $k = 3$ 的最近邻进行线性插值

Fig.2.2 SMOTE linearly interpolates a randomly selected minority sample and one of its $k = 3$ nearest neighbors

第 3 章 基于邻域粗糙集的元启发式算法

3.1 问题描述

随着大数据时代的蓬勃发展，数据的规模和维度呈现爆炸式增长，特征选择作为数据预处理领域的研究重点，在数据挖掘和模式识别中发挥着至关重要的作用。特征选择不仅能够节省存储空间，还能显著提升学习模型的泛化能力。在策略层面，特征选择主要分为穷举搜索和启发式搜索两大类，二者均以寻找最具相关性和显著性的特征子集为目标。尽管穷举搜索在早期研究中取得了显著进展，但由于其固有的 NP-hard 问题，随着数据量的激增，其时间消耗变得难以承受。鉴于此，众多学者转向启发式搜索策略，以在合理的时间范围内寻找更具相关性和显著性的特征子集。特别地，为了更精确地找到全局最优特征子集，元启发式算法在特征选择领域的应用日益广泛。然而，实际应用中的数据集往往包含大量数值特征，而传统的粗糙集理论在处理这些特征时存在局限性。为了克服这一挑战，本文将邻域粗糙集理论引入元启发式算法中，以同时支持连续和离散特征的处理。邻域粗糙集理论与元启发式算法的结合，能够从数值决策系统中有效地提取出最优且最小的特征子集。在本节中，我们将重点介绍四个基于邻域粗糙集的元启发式算法，这些算法在特征选择领域展现出了巨大的潜力和优势。

3.2 本章工作

3.2.1 基于邻域粗糙集的帝王蝶优化算法

基于帝王蝶[25]的迁移行为，提出了帝王蝶优化算法 (Monarch Butterfly Optimization, MBO). 在该算法中，每只帝王蝶的位置代表了一个可行的解（特征子集）。该算法采用适应度函数来衡量可行解的有效性，即可行解代入适应度函数得到的适应度值越大，其对分类问题就越有效。帝王蝶的两种主要行为是迁移行为和适应行为，每只帝王蝶可以采取两种行为的一种。MBO 的主要步骤是：首先，将领地分为两个部分，即陆地 1 和陆地 2，位于陆地 1 上的帝王蝶被称为亚种群 1，而位于陆地 2 上帝王蝶被称为亚种群 2。然后，让亚种群 1 或亚种群 2 的部分个体执行迁移行为，而陆地 2 上的帝王蝶执行适应行为。接着，通过对比适应度值，找出适应度值最大的个体，并保留到下一次迭代中。最后，判断该帝王蝶个体是否满足阈值，若满足则停止迭代并输出个体代表的特征子集；否则，重复迭代至迭代次数达到算法设置的上限。

接下来，我们详细描述了帝王蝶的两种主要行为，即迁移行为和适应行为：

(1) 迁移行为：迁徙行为是指帝王蝶从陆地 1 到陆地 2 或从陆地 2 到陆地 1。p是陆地 1 中包含的帝王蝶在总种群中的占比。使用的等式 (3.1) 得到的新的个体是：

$$x_{i,k}^{t+1} = x_{m,k}^{t}, \tag{3.1}$$

x_i表示帝王蝶i的位置。$x_{i,k}^{t+1}$是x_i在迭代次数为$t+1$时的第k位的值。同样地，x_m表示帝王蝶m的位置。$x_{m,k}^{t}$是x_i在迭代次数为t时的第k位的值。帝王蝶m是从亚群 1 或亚群 2 中随机选择的个体。设$r = rand \times peri$，$peri$表示迁移周期，$rand$是区间 [0, 1]中均匀分布的随机数。对于$peri$的值，根据参考文献[25]设置为 1.2。如果是$r \le p$，则在亚种群 1 中随机选择个体m，否则，则在亚种群 2 中随机选择个体m。可见，MBO 可以通过调节p来控制后代的来源。p越大，陆地 1 中新的帝王蝶更有可能是亚种群 1 的后代；否则，土地 1 中新的帝王蝶更有可能是亚种群 2 的后代。

(2) 适应行为：适应行为只能对亚种群 2 的个体有效。下面简要说明适应行为的详过程。如果是$rand \le p$，则亚种群 2 中的个体j将根据以下等式 (3.2) 进行更新

$$x_{j,k}^{t+1} = x_{best,k}^{t}, \tag{3.2}$$

其中，x_{best}表示种群中的全局最优个体。如果$rand > p$，个体 j 将使用等式 (3.3) 对于位置更新：

$$x_{j,k}^{t+1} = x_{m,k}^{t}, \tag{3.3}$$

其中，x_m是在亚种群 2 中随机选择的一个个体。设BAR为帝王蝶调整率，根据参考文献[25]设为5/12。如果是$rand > BAR$，对应等式 (3.2) 和等式 (3.3) 的位置进一步更新为：

$$x_{j,k}^{t+1} = x_{j,k}^{t+1} + \alpha \times (dx_k - 0.5), \tag{3.3}$$

其中α表示加权因子，它可以由$\alpha = S_{\max} / t^2$计算，其中$S_{\max}$是帝王蝶个体在一次迭代中可以移动的最大步长。dx是帝王蝶 j 的行走步长，可以通过执行$Levy(x_j^t)$函数来计算出来。dx_k表示dx的第k个元素的行走步长。根据参考文献[25]，$Levy(x_j^t)$的功能被称为$Levy$飞行。如果一只帝王蝶使用了$Levy$飞行，它将移动一次较大的步长。α的值越大，帝王蝶种群就有机会获得更大的适应度值，这样有利于跳出局部最优适应度值。

这两种行为都利用适应度函数产生了可行的解。本文用邻域粗糙集定义了所使用的适应度函数。因此，提出了一种基于邻域粗糙集的帝王蝶优化算法，见算法 3.1 所

示。

算法 3.1： 基于邻域粗糙集的帝王蝶优化算法 (NMBO)

输入: 邻域决策系统 DS = <*U*, *AT*∪{*d*}>;

输出: *A*.

步骤 1: 初始化，令 *A*=∅, 并生成初始种群 *P*;

步骤 2: 根据定义 1.5 计算适应度值并排序，找到最优的帝王蝶位置和其对应的适应度值;

步骤 3: **While** 不满足最大迭代或停止条件 **Do**

将种群分为 N_1 和 N_2;

For *i*=1:N_1 **Do**

通过迁徙行为生成新的子种群 N_1;

End For

For *j*=1:N_2 **Do**

通过适应行为生成新的子种群 N_2;

End For

将新生成的子种群合并为一个整体种群;

按其适应度值从小到大排列，组成新的规模为 *N* 的种群;

将 *A* 置为当前适应度最大的个体;

t=*t*+1;

End While

步骤 4 输出 *A*.

影响 NMBO 时间消耗的主要因素是计算种群中每个帝王蝶的适应度值所花费的时间。根据参考文献[53]的分析，计算单个帝王蝶个体适应度值的时间复杂度为 $O(|U^2|)$。在最坏的情况下，所有个体都必须进行 *T* 次迭代评估，NMBO 的时间复杂度可以表示为 $O(N\times|U^2|\times T)$。

3.2.2 基于邻域粗糙集的遗传算法

遗传算法 (Genetic Algorithm, GA)[26]是对自然界生物进化过程的模拟，通过遗传机制寻找问题的最优解决方案。它已被用于解决各种问题，包括特征选择。在本文中，每个基因个体代表一个特征选择结果。此外，“1”或“0”的单个基因位的值表示一个特征是否被选中。因此，遗传算法利用选择、交叉和突变三种行为，产生新的基因个体，寻找具有合适适应度值的个体，即满足要求的特征结果集。对这三种行为的具体描述是：

(1) 选择：基因选择是 GA 的核心组成部分。它使用一种特定的机制从一些个体中选择遗传信息，然后将其传递给下一代。有三种流行的选择机制：基于轮盘赌的选择[70]，基于排名的选择[26] 和基于比赛规则的选择[71]。通过轮盘赌选择，遗传算法可以快速获得优秀的基因个体。因此，本文利用轮盘赌选择，为适应度值较高的个体分配较高的选择概率。

(2) 交叉：在 GA 中，基因交叉是一种合并来自双亲本个体的遗传信息以产生新的后代个体的基本操作。这个过程对于促进对特征空间的探索是至关重要的。均匀交叉有利于保持特征信息的交换。因此，本文采用均匀交叉[70]作为基因交叉机制。具体来说，均匀交叉交换两个父代个体在随机选取的索引 i 上的基因位，交叉概率以 pc 表示。

(3) 变异：在 GA 的背景下，基因突变是指对单个基因的修改。基于个体编码方法的两种常用的变异类型是实数变异[71]和二进制变异[70]。本文采用二进制突变，其中单个基因位可以取值为“1”或“0”。本研究的突变操作包括两个基本步骤：第一，根据突变概率pm确定一个基因是否应该发生突变；第二，通过将目标基因的值在“1”和“0”之间翻转来修改目标基因位。

利用这三种行为，遗传算法可以通过适应度函数生成可行的解，并迭代地提高总体的质量，直到满足终止准则。本文所利用的适应度函数是由邻域粗糙集来定义的。因此，提出了一种基于邻域粗糙集的遗传算法，见算法 3.2 所示。

算法 3.2：基于邻域粗糙集和遗传算法的特征选择算法。(NGA)

输入： 邻域决策系统 DS = <*U*, *AT*∪{*d*}>;

输出： *A*.

步骤 1: 初始化，令 *A*=∅, 初始种群 *P*;

步骤 2: 根据定义 1.5 计算适应度值并排序，找到最优的基因个体位置和其对应的适应度值;

步骤 3: **While** 不满足最大迭代或停止条件 **Do**

通过基因选择产生新个体;

通过基因交叉交换个体间的基因;

通过基因基因变异使个体的基因段发生改变;

按其适应度值从小到大排列，组成新的规模为 *N* 的种群;

将 *A* 置为当前适应度最大的个体;

t=*t*+1;

End While

步骤 4: 输出 *A*.

如 NGA 所述，主要的时间消耗来自于基因选择行为中新个体的产生。具体来说，在每一代中，算法必须计算种群中所有基因个体的适应度值，这就需要评估所有个体的适应度函数。这个计算中单个基因个体的时间复杂度为$O(|U^2|)$。假设该算法运行于种群规模 N 为的 T 次迭代，则 NGA 的总时间复杂度将为$O(N\times|U^2|\times T)$。需要注意的是，这是最坏时间复杂度，实际运行时间可能会根据数据集的大小和代数等因素而变化。

3.2.3 基于邻域粗糙集的人工鱼群算法

鱼群在食物周围有觅食、聚集、跟随和随机游泳的行为。人工鱼群算法 (Artificial Fish Swarm Algorithm, AFSA)根据鱼类的生存特征[24]，制定了四种行为来模拟鱼类的生活习性。每条鱼的位置可以看作是它的食物浓度，它具有一个由适应度函数评估的适应度值。在每次迭代中，每条鱼都通过觅食、群聚、尾随和随机游动的行为来满足其食物（适应度值）需求。并且，人工鱼群算法在每次迭代中都记录了适应度值最大的个体。单条鱼的状态记为向量 X，当前鱼的食物浓度记为 $Y(X)=fitness(X)$，其中，Y 为目标函数的值。任意两条鱼之间的距离记为 $d_{i.j}=\left\|X_i-X_j\right\|$，鱼类种群总数为 N。根据参考文献[24]，其他参数包括鱼的视野 $visual=(|AT|-1)/2$，步长 $step=1$，拥塞因子 $\varphi=0.618$，和尝试次数 $try\_number=10$。对这四种行为的具体描述如下：

(1) 觅食行为：这是鱼类对食物的基本生物学行为。一般来说，人工鱼通过视觉感知周围食物的浓度来决定向哪里移动。假设人工鱼此时位于状态 X_i，然后从其 $visual$ 范围中随机选择一个状态 X_j。设 $X_{i,next}$ 表示鱼在下一次迭代中的位置，$rand()$ 表示一个可以在(0, 1)之间产生一个随机数的函数。如果 $Y(X_i)<Y(X_j)$，鱼会根据等式(3.3) 移动；否则，再次随机选择 X_j，判断是否满足条件 $Y(X_i)<Y(X_j)$。如果在 try_number 次尝试后条件仍然不满足，X_j 将使用等式(3.6)进行随机游泳。

$$X_{i,next}=X_i+step\times rand()\times\frac{X_j-X_i}{\left\|X_j-X_i\right\|}. \tag{3.3}$$

(2) 聚集行为：这是一种自然的行为，鱼群在捕食时会自然地聚集在一起。这种行为不仅可以引导鱼类个体去捕猎，而且还可以确保鱼类群体的生存和避免危险。首先，以鱼的现状 X_i 为起点，探索邻近区域的数量 nf 和中心位置 X_c（即 $d_{i.j}<visual$）。如果 $nf/N<\varphi$ 和 $Y(X_c)>Y(X_i)$，那么可以认为中心位置不拥堵，食物浓度也更大。因此，鱼将通过使用等式(3.4) 在邻居中心的方向上前进一步。如果不满足 $nf/N<\varphi$ 或 $Y(X_c)>Y(X_i)$，该鱼将执行觅食行为。

$$X_{i,next}=X_i+step\times rand()\times\frac{X_c-X_i}{\left\|X_c-X_i\right\|}. \tag{3.4}$$

(3) 跟随行为：如果一条人工鱼找到了食物，它周围的其他鱼会很快跟随它到达食物。以 X_i 为鱼 i 的当前状态，以 $X_{\max}$ 为一种与鱼 i 周围适应度值最大的鱼的状态。然后，我们检查 $X_{\max}$ 的邻居（即 $d_{i.j}<visual$），并计算它们的数量为 nf。如果 $nf/N<\varphi$ 和 $Y(X_{\max})>Y(X_i)$，那么 $X_{\max}$ 的所在的周围就没有那么拥挤并且食物浓度更大，所以 X_i 将通过使用等式(3.5) 向邻近的 $X_{\max}$ 移动一步。如果条件 $nnf/N<\varphi$ 或 $Y(X_{\max})>Y(X_i)$ 不满足，则 X_i 将执行觅食行为。

$$X_{i,next} = X_i + step \times rand() \times \frac{X_{\max} - X_i}{\left\| X_{\max} - X_i \right\|}. \tag{3.5}$$

(4) 随机游泳行为: 等式(3.6) 用于人工鱼群的随机游泳行为。人工鱼可以在视觉范围内随机选择一个新的位置。

$$X_{i,next} = X_i + step \times rand(). \tag{3.6}$$

通过使用这四种行为，AFSA 可以通过适应度函数生成最优鱼，并迭代地提高种群的质量，直到满足终止准则。本文使用的适应度函数是由邻域粗糙集定义的。为此，提出一种基于邻域粗糙集的人工鱼群算法，见算法 3.3 所示。

算法 3.3： 基于邻域粗糙集的人工鱼群算法。(NAFSA)

输入： 邻域决策系统 DS = <*U*, *AT*∪{*d*}>;

输出： *A*.

步骤 1: 初始化，令 *A*=∅, 并生成初始种群 *P*;

步骤 2: 根据定义 1.5 计算适应度值并排序，记录最优人工鱼的位置和其对应的适应度值;

步骤 3: **While** 不满足最大迭代或停止条件 **Do**

For *i* = 1: *N* **Do**

计算人工鱼个体 *i* 视野内鱼类数量 *nf*;

If $Y_c/n_f > \varphi \times Y_i$

根据等式 3.4 执行鱼群聚集行为;

Else

根据等式 3.3 执行鱼群觅食行为;

End If

If $Y_{max}/n_f > \varphi \times Y_i$

根据等式 3.5 执行鱼群跟随行为;

Else

根据等式 3.4 执行鱼群觅食行为;

End If

End For

将 *A* 置为当前适应度最大的个体;

t=*t*+1;

End While

步骤 4: 输出 *A*.

NAFSA 的时间消耗主要源自计算个体适应度值的聚集行为。具体而言，每条人工鱼在执行聚集行为时，需要计算其他鱼的适应度值。此时，每个人工鱼所需的时间复杂度为 $O(N \times |U|^2)$。假设该算法运行的*T*的种群规模为*N*，那么 NAFSA 的总时间复杂度将为 $O(N^2 \times |U|^2 \times T)$。值得注意的是，这是最坏时间复杂度。实际运行时间可能较低。它取决于各种因素，如收敛速度和数据集的大小。

3.2.4 基于邻域粗糙集的森林优化算法

森林优化算法[27](Forest Optimization Algorithm, FOA)是一种针对特征选择问题提出的元启发式算法，已被证明在实践中取得了良好的效果。FOA 中的每棵树都反映了一个可能用于特征选择问题的特征子集。FOA 由五个阶段组成，其具体过程如下：

(1) 初始化：在初始化阶段，随机生成一定数量且相同长度的树，形成初始森林。这里的长度等于参与特征选择的原始数据的特征维度的长度。每棵树都由“0”或“1”组成。而树中的“0”或“1”表示是否选择了对应位置所代表的特征。此外，FOA 使用 *age* 表示树的年龄。因此，每棵新树的 *age* 的值被设置为 0。

(2) 局部播种：在局部播种阶段，该算法根据以下步骤生成新的树。将 *LSC* 设置为选定位置的数量。首先，从 0 岁的老树中随机选择 *LSC* 个不同的位置。接下来，我们通过复制来保存这些老树。然后，修改老树中所选位置的值，以便通过将值从“0”更改为“1”或从“1”更改为“0”来生成新树。最后，将新树和老树的备份一起添加到森林中。此外，新树的年龄设置为 0，而老树的年龄增加了 1。通过这些步骤，该算法可以在森林中快速生成新的树木，从而扩大了森林的多样性，提高了算法的搜索能力。

(3) 种群限制：这个阶段主要用于限制森林的种群数量，形成候选森林。候选森林是由从当前森林中移除的树木形成的。它将作为全局播种阶段的一片森林。由于局部播种阶段，已经创造了大量的新树木。为了限制森林的种群数量，在以下两种情况下，将树木从森林中移除，并放置在候选森林中：设置 *left _ time* 作为树的年龄上限。比 *left _ time* 大的 *age* 被从森林中移除，放置在候选森林中。设置 *area _ time* 为森林种群的上限。如果当前林中的树的总数仍然超过 *area _ time* 。森林中的树按各自的适应度值降序排序，排名在 *area _ time* 以上的树也被从森林中移除。

(4) 全局播种：FOA 的主要特征搜索方法是全局播种阶段。通过改变位置，它将比当地的播种阶段产生更多不同的新树。将 *transfer _ rate* 设置为在全球播种阶段使用的候选种群的百分比，将 *GSC* 设置为要选择的位置数 ($GSC \geq 2 \times LSC$). 在全球播种阶段，候选树中的树木按照 *transfer _ rate* 的比例形成了一个森林。在森林中，每棵树随机选择 *GSC* 位置的值进行改变。（从“1”到“0”或从“0”到“1”）。生成的新树将被添加到初始森林中，并参与更新阶段。

(5) 更新：FOA 算法在更新阶段选择适应度值最大的树，并将其 *age* 设置为 0。然后，这些树被释放回原来的森林中。这样，森林中的最佳树木总是会参与到森林的优化中来。

上述行为将产生一个具有最大适应度值的最优树。每次迭代时，FOA 都可以确定最优树是否满足约束条件。如果满足约束条件，则输出由最优树所表示的特征选择结

果。否则，在下一次迭代中由上述行为生成新的最优树。本文通过邻域粗糙集中的近似质量这一度量来定义适应度函数。因此，提出了一种基于邻域粗糙集的森林优化算法，见算法 3.4 所示。

算法 3.4：基于邻域粗糙集的森林优化算法。(NFOA)

输入： 邻域决策系统 DS = <*U*, *AT*∪{*d*}>;

输出： *A*.

步骤 1: 初始化，令 *A*=∅, 适应度函数 *fitness*(*x*), 决定在局部播种的过程中选择特征变量的个数 *LSC*, 决定在全局播种的过程中选择变量的个数 *GSC*, 区域种群数量限制 $area\_time$，寿命限制 $left\_time$，群体的规模 *N*, 空间维度 *D* 和最大迭代次数 *T*, 并生成初始种群 *P*, 使初始种群的年龄为 0;

步骤 2: 根据定义 1.5 计算每个个体的适应度值;

步骤 3: **While** 不满足最大迭代或停止条件 **Do**

For *i* = 1 : *LSC* **Do**

随机选择选中树的一个位置;

将此位置的值从 0 更改为 1，反之亦然;

End For

将所有树木的年龄增加 1;

通过 $area\_time$ 和 $left\_time$ 的值限制种群规模;

在候选树上进行全局播种;

选择候选人口的转移率百分比;

For 每个被选中树 **Do**

随机选择所选树的 *GSC* 个不同位置;

将所选位置的值从 0 更改为 1，反之亦然;

End For

计算所有个体的适应度值，并更新当前最优树;

对树按照适应度值进行排序并找到当前的最优个体;

记录最优树并将其年龄置为 0;

t=*t*+1;

End While

步骤 4: 输出 *A*.

根据算法过程可以，NFOA 的时间消耗主要从全局播种阶段开始。而执行全局播种阶段的时间复杂度为 $O(N \times GCS^t \times |U^2|)$，其中*N*为初始森林中树的数量。在最坏的情况下，计算必须在*T*迭代中执行，那么它是 $O(N \times GCS^t \times |U^2| \times T)$。事实上，这一阶段的表现取决于在种群限制阶段中产生的候选森林。每次迭代生成的候选森林越大，运行此阶段所需的时间就越多。

3.3 实验分析

通过实验分析四种元启发式算法和前向贪心搜索策略的优劣，具体的实验结果在本章节进行讨论和分析。

3.3.1 实验数据

本节从 UCI 标准数据集中选取 13 个数据集进行实验，以证明本文算法的有效性。数据的具体信息见下表 3.1。实验中，所有的数据集的条件特征值都是经过归一化处理的。所有实验均在一台装有 Windows 10, CPU 为 Intel® Core(TM) i5-7300HQ (2.50 GHz) 和 8.00 GB 内存的个人计算机上进行。实验平台为 MATLAB R2018a。

表 3.1 数据集描述

Table 3.1 Description of data sets

ID	数据集名称	样本数	特征数	决策类数	来源
1	Urban Land Cover	675	147	9	UCI
2	LSVT Voice Rehabilitation	126	256	2	UCI
3	Synthetic Control Chart Time Series	600	60	6	UCI
4	Statlog (Heart)	270	13	2	UCI
5	Ionosphere	351	34	2	UCI
6	Musk (Version 1)	476	166	2	UCI
7	Yale_64x64	165	4096	15	UCI
8	ORL_32x32	400	1024	40	UCI
9	TOX_171	171	5748	4	Gene
10	Brain_Tumor	90	5919	5	Gene
11	waveform	5000	21	3	UCI
12	Wall-Following Robot Navigation	5456	24	4	UCI
13	Crowdsourced Mapping	10845	29	6	UCI

3.3.2 实验设置

本次实验将对比五种算法分别进行特征选择的时间消耗以及利用得到的特征选择的结果求得的分类准确率和稳定性。

在本次实验中，选择了 20 个不同的半径，即 δ=0.02, 0.04,..., 0.4．并且，本次实验选择近似质量构造约束条件来进行特征选择。

实验采用 10 折交叉验证的方法测试算法的性能，交叉验证具体的过程如下所示：将数据集平均划分成 10 份，即 $U_1 \cup U_2 \cup ... U_{10}$．在第一轮计算过程中，$U_1 \cup U_2 \cup ... U_9$ 作

为训练集来计算特征选择结果集，U_{10} 视为测试集。利用求得的特征选择的结果对 U_{10} 上的数据进行分类，从而评估不同算法的分类能力。以此类推，在第十轮计算过程中，$U_2 \cup U_3 \cup ... U_{10}$ 作为训练集来计算特征选择结果集，U_1 视为测试集。利用求得的特征选择的结果对 U_1 上的数据进行分类，从而评估不同算法的分类能力。实验使用 KNN (k=3) 和 CART (默认参数)两种分类器对测试数据集进行分类。此外，将元启发式算法生成的初始种群中个体的初始长度设置为 $10\% \times |AT|$, 需要特别指出是，我们使用了带有标签噪声的数据集进行实验，以检验我们所提算法的有效性。为了模拟数据集中的标签噪声，采用以下方法生成标签噪声：给定一个原始数据集，如果噪声比为 ω%，则通过随机选择 ω% ($0 < \omega < 100$) 样本，并随机替换这些样本的标签来实现注入噪声。为了使噪声比例处在合理范围内，即过小达不到产生噪声干扰的效果，过大不易区分算法性能。因此，本文选择在比例为 30%的标签噪声环境中进行实验。

3.3.3 实验结果

在本组实验中，将对比五个不同的算法(算法 2.1: FGS，算法 3.1:NMBO，算法 3.2: NGA、算法 3.3: NAFSA 和算法 3.4: NFOA)在近似质量作为度量的情况下分别进行特征选择的时间消耗。

表 3.2 特征选择算法的时间消耗（秒）

Table 3.2 Time consumptions of Algorithms for feature selection (second)

ID	FGS	NMBO	NGA	NAFSA	NFOA
1	5.2652	**0.2791**	1.6414	0.6731	3.6413
2	0.3824	**0.0186**	0.1701	0.1712	0.3217
3	3.2262	**0.1398**	1.0032	0.2366	1.6109
4	**0.0224**	0.0581	0.4245	1.0211	0.6506
5	0.1054	**0.0398**	0.3978	1.1996	0.6321
6	0.8370	**0.0725**	0.2156	0.6542	0.1554
7	16.8569	**0.7624**	3.7880	2.2264	5.4187
8	35.1392	**0.3100**	2.4650	2.1473	7.6402
9	14.4865	**1.5100**	5.2954	3.5454	7.4415
10	6.3722	**1.2386**	3.9695	2.5415	3.5366
11	**18.3426**	25.7281	157.3470	1079.4309	3471.1192
12	**30.1060**	45.5498	954.8862	2340.0245	1324.2882
13	135.2546	**124.5448**	758.2014	792.8896	1624.6950
平均	20.4920	**15.4040**	145.3696	325.1355	496.2424

通过观察表 3.2，不难发现：

(1) 与算法 2.1 比较，在条件特征数较多情况下，在邻域粗糙集下采取近似质量为度量，利用算法 3.1-3.4 进行特征选择后的时间消耗低于利用算法 2.1 进行特征选择的时间消耗。这主要是因为前向贪心搜索策略每次搜索添加单个特征进行评估，而元启发式算法通过随机搜索多个特征的方式，可以减少特征选择的迭代次数。以数据集“TOX_171”(ID: 9) 为例，利用算法 1.1 进行特征选择的时间消耗为 14.4865 秒，而算法 3.1-3.4 进行特征选择的时间消耗分别为 1.5100、5.2954、3.5465、7.4415 秒。算法 3.1-3.4 显然加快了特征选择的过程。但是，在一些数据集上，特别是特征较少而样本较多时，利用算法 1.1 进行特征选择的时间消耗少于算法 3.1- 3.4。以数据集“Ionosphere”(ID: 5) 为例，利用算法 1.1 进行特征选择的时间消耗为 0.0224 秒，而算法 3.1-3.4 进行特征选择的时间消耗分别为 0.0581、0.4245、1.0211、0.6506 秒。显然，元启发式算法需要在样本较多时进行一定的加速以减少时间消耗。

(2) 在四种元启发式算法中进行比较，在绝大多数情况下，在邻域粗糙集下利用算法 3.1 特征选择的时间消耗低于利用算法 3.2-3.4 特征选择的时间消耗。以数据集“Urban Land Cover”(ID: 1) 为例，利用算法 3.2、算法 3.3 以及算法 3.4 进行特征选择的时间消耗分别为 1.6414、0.6731 和 3.6413 秒，而使用算法 3.1 仅为 0.2791。显然，算法 3.1 相比于算法 3.2-3.4 更加提升了进行特征选择的时间效率。其主要原因是算法 3.1 利用帝王蝶个体之间的信息共享来指导搜索方向，所有个体都朝向当前已知的最优解移动，这种集体学习的过程可以加速算法找到全局最优解。即最优特征选择结果集。

接下来，将对比利用不同方法进行特征选择的分类性能。在该组实验中，采用 KNN 和 CART 分类器进行分类。具体的实验结果如表 3.3-3.6 所示。

表 3.3 在邻域粗糙集下 KNN 分类器的分类稳定性

Table 3.3 The classification stability of KNN classifier under neighborhood rough set

ID	FGS	NMBO	NGA	NAFSA	NFOA
1	0.5692	0.6567	0.6532	0.6448	**0.6843**
2	0.6236	**0.6516**	0.6417	0.6420	0.6453
3	0.5824	0.6888	0.7071	0.6910	**0.7170**
4	0.6407	0.6661	0.6732	0.6736	**0.7085**
5	0.8006	**0.8411**	0.8253	0.8387	0.8358
6	0.5381	0.6812	0.6567	**0.6989**	0.6877
7	0.7109	0.8885	**0.9030**	0.8828	0.8880
8	0.7515	0.8043	0.8132	**0.8211**	0.8202
9	0.5606	0.7429	0.7293	**0.7603**	0.7460
10	0.6339	0.8783	0.8762	0.8793	**0.8808**

（续表 3.3）

ID	FGS	NMBO	NGA	NAFSA	NFOA
11	0.6245	0.5562	0.7025	0.7215	**0.7347**
12	0.6465	0.5328	0.8250	0.8252	**0.8454**
13	0.7654	**0.8964**	0.8546	0.8654	0.8867
平均	0.6498	0.7296	0.7585	0.7650	**0.7754**

表 3.4 在邻域粗糙集下 CART 分类器的分类稳定性

Table 3.4 The classification stability of CART classifier under neighborhood rough set

ID	FGS	NMBO	NGA	NAFSA	NFOA
1	0.6181	0.6947	0.6853	0.6973	**0.7289**
2	0.6124	0.6588	0.6776	0.6599	**0.6798**
3	0.6522	0.7355	0.7500	0.7400	**0.7611**
4	0.5722	0.5544	0.5570	0.5536	**0.5750**
5	**0.7579**	0.7457	0.7525	0.7503	0.7453
6	0.5644	0.6660	0.6856	0.6685	**0.6959**
7	**0.6727**	0.6273	0.6310	0.6257	0.6455
8	**0.6793**	0.6040	0.6188	0.5947	0.6370
9	0.5531	0.5526	0.5464	0.5625	**0.5905**
10	**0.6589**	0.6050	0.6075	0.6141	0.6447
11	0.6224	0.5763	0.6546	0.6425	**0.6752**
12	0.8355	0.7655	0.9054	**0.9320**	0.9245
13	0.7654	0.8235	**0.8355**	0.8240	0.8125
平均	0.6588	0.6623	0.6852	0.6819	**0.7012**

表 3.5 在邻域粗糙集下 KNN 分类器的分类准确性

Table 3.5 The classification accuracy of KNN classifier under neighborhood rough set

ID	FGS	NMBO	NGA	NAFSA	NFOA
1	0.4867	0.6019	0.6231	0.5855	**0.6577**
2	0.6636	0.6972	0.6957	0.7063	**0.7351**
3	0.5194	0.6804	0.6857	0.6691	**0.6957**
4	0.6324	0.6676	0.6600	0.6931	**0.6812**
5	0.8293	0.8320	0.8716	**0.8812**	0.8434
6	0.5733	0.6849	**0.7089**	0.6702	0.6743
7	0.2233	0.7194	0.7236	0.7316	**0.7488**
8	0.1721	0.8026	0.8043	**0.8270**	0.8078

(续表 3.5)

ID	FGS	NMBO	NGA	NAFSA	NFOA
9	0.3734	0.7609	0.7530	0.7781	**0.8000**
10	0.5933	0.9178	0.9073	0.9105	**0.9423**
11	0.4244	0.5540	0.7254	0.7422	**0.7555**
12	0.5024	0.5200	0.7954	0.8200	**0.8555**
13	0.7086	0.8186	0.8154	0.8254	**0.8351**
平均	0.5156	0.7121	0.7515	0.7569	**0.7717**

表 3.6 在邻域粗糙集下 CART 分类器的分类准确性

Table 3.6 The classification accuracy of CART classifier under neighborhood rough set

ID	FGS	NMBO	NGA	NAFSA	NFOA
1	0.6166	0.7047	0.7069	0.7218	**0.7465**
2	0.6644	0.7180	**0.7442**	0.7163	0.7351
3	0.6390	0.7357	0.7320	**0.7431**	0.7358
4	0.5944	0.6072	**0.6403**	0.6250	0.6274
5	**0.8356**	0.8157	0.8070	0.7914	0.7994
6	0.6213	0.7212	0.7154	0.7034	**0.7401**
7	0.3003	0.5439	0.5418	0.5463	**0.5557**
8	0.2793	0.4515	0.4627	0.4675	**0.4870**
9	0.4151	0.5246	0.5152	0.5219	**0.5581**
10	0.6406	0.6711	0.6691	0.6797	**0.7087**
11	0.5240	0.6200	0.7054	0.6954	**0.7039**
12	0.6500	0.8201	0.9234	0.9221	**0.9472**
13	0.7558	0.7944	0.8244	0.8065	**0.8354**
平均	0.5797	0.6714	0.6914	0.6877	**0.7062**

由表 3.3-3.6 可知，使用 KNN 和 CART 分类器时，算法 3.1-3.4 的性能优于算法 2.1。这表明元启发式算法在特征较多时分类性能优于前向贪心搜索策略。然后，结合表 3.2 的时间消耗对比，具体来说：

(1) NMBO、NGA、NAFSA 和 NFOA 与 FGS 相比分类性能更好。考虑在 13 个数据集上的整体表现即均值情况。NMBO、NGA、NAFSA 和 NFOA 与 FGS 相比，在 KNN 分类稳定性上分别提高了 12.28%、16.73%、17.73%和 19.33%。CART 的分类稳定性分别提高了 0.53%、4.01%、3.50%和 6.44%。KNN 的分类准确率也分别提高了 38.11%、45.75%、46.80%和 49.67%。CART 分类精度分别提高

了 15.82%、19.27%、18.63 %和 21.82%。

(2) 在四种元启发式算法中进行比较，在绝大多数情况下，四种算法在两个分类器上的表现相当，其均值差异在 5%以内。其中，NFOA 算法的分类性能较好与其他三种元启发式算法（NMBO、NGA、NAFSA），但是其时间消耗也在四个元启发算法中最多。综合考虑，MBO 算法的性能最好，时间消耗最少并且具有相当的分类性能。

(3) 四种元启发式算法在样本较多时，面临着时间消耗增大和分类性能降低的问题。以数据集“Ionosphere”(ID: 5) 为例，四个元启发式算法的时间消耗多于算法 3.1，且分类性能也低与算法 3.1，如 CART 分类器的分类准确性。算法 2.1 为 0.8356, 算法 3.1-3.4 分别为 0.8157, 0.8070, 0.7914, 0.7994。

3.4 本章小结

在本章中，首先，分别介绍 MBO、GA、AFSA 和 FOA 这四种元启发式算法的原理。接着，通过邻域粗糙集定义的适应度函数，本文将元启发式算法与邻域粗糙集相结合提出了四种新算法：NMBO、NGA、NAFSA 和 NFOA。最后，为了深入剖析这四种基于邻域粗糙集的元启发式算法的性能特点、差异以及各自的优劣势，本文进行了一系列严谨的验证性实验。实验中，我们选择了多个广泛使用的数据集，并将这些新算法与前向贪心算法（FGS）进行了对比。通过对比分析，我们可以更全面地了解这些算法在特征选择方面的效果，为后续的研究提供了坚实的实验基础。本章中的 4 个新算法也是后续对照实验所采用的算法，在后文中将直接进行实验对比结果，不再具体阐述每个算法的具体内容。

第 4 章 基于随机采样加速器和重采样的快速元启发式算法

4.1 问题描述

当前，随着数据集规模的急剧扩张，数据维度呈显著上升趋势，这使得元启发式算法在运行时面临严重的时间消耗问题。为了解决这一挑战，众多专家学者已投入大量精力设计了一系列加速算法。据我们了解，这些加速方法大致可以归为两类不同的加速器。接下来，我们将详细介绍这两类加速器。

(1) 以样本角度[53-54,68]切入的加速器：这类加速器的关键步骤是通过减少遍历样本的迭代次数或者样本之间的比较次数来提升进行特征选择的时间效率。例如，通过聚类的方式进行样本的选择。通过这种方法，只需要按照实验需要选择符合聚类标准的样本即可，而不是整个样本集。这就是为什么该方法可以从样本角度来加快进行特征选择的过程。

(2) 以特征角度[69]切入的加速器：此类加速器的核心在于优化特征评估与特征搜索的效率。该方法以特征间的相似度为核心考量，构建不同的特征簇，从而在特征选择过程中，只需基于这些特征簇来筛选候选特征。通过这种方式，我们显著减少了评估候选特征的次数，进而从特征层面提升了特征选择的时间效率。这种方法不仅逻辑清晰，而且符合论文的严谨性要求，为高效特征选择提供了新的途径。

因此，可以发现这些设计出的加速器可以用来加快特征选择的过程。然而，在面对元启发式算法时，基于特征角度的加速器存在以下两点不足：

- 基于特征角度，加速效果不明显：元启发算法采用指定的求解过程进行，基于特征角度的加速器与具体的元启发式算法难以匹配，导致加速器并不能有效地提高算法的性能，甚至会对算法的迭代过程产生负面影响。
- 改变特征空间结构，增加局部收敛风险：特征变换可能会改变优化问题的结构，使得一些局部最优解在新的特征空间中变得更为显著，全局最优解会变得更难寻找。这可能会导致元启发式算法在新的特征空间中表现不佳，或者需要调整算法参数以适应变化后的问题结构。这样会破坏元启发式算法的求解过程，可能会改变问题的特征表示，使得算法更快地收敛到局部最优解，降低算法的分类性能。

综上所述，基于特征角度的加速器存在很多不足之处，探究更适合元启发式算法的加速器具有一定的价值和意义。采用随机采样加速器有以下两点好处：

- 极大地加快特征选择：随机采样加速器能够引入样本的随机性，使特征空间中

的优化过程更为均匀且全面。由于采样的随机特性，算法能够更广泛地覆盖样本空间，进而增加发现全局最优解或更优解的可能性。

- 不增加局部收敛风险：随机采样加速器可以在不增加局部收敛风险的情况下加速算法的收敛速度，因为它通过引入随机性增加了算法的探索性，使得算法更有可能找到更优的解决方案，而不会受到局部最优解的束缚。

随机采样在处理数据时，会遇到类别不平衡的问题。针对这一问题，研究者们提出了多种策略，包括调整分类算法，考虑不同类别在分类过程中的错误成本，或者修改训练数据。其中，重新采样是一种常用的方法，通过欠采样或过采样来调整训练数据的分布。过采样策略可以是简单地重复现有样本，或者生成新的人工样本来丰富少数类的数据。然而，简单的随机过采样可能会引发过拟合问题，为了解决这一问题，研究者们提出了 SMOTE 算法。与简单地复制样本不同，SMOTE 算法通过在少数类样本及其邻近样本之间进行线性插值来生成新的合成样本。尽管 SMOTE 算法在处理类别间不平衡方面表现出色，但它也面临一些挑战。首先，SMOTE 在处理类内不平衡和噪声问题时存在一定的局限性。其次，当对包含噪声的少数类样本及其邻居进行插值时，SMOTE 可能会加剧数据中的噪声问题。最后，SMOTE 算法无法避免在类边界附近生成样本，这可能导致过采样对决策边界的明确划分产生不利影响。因此，在使用 SMOTE 算法时，需要谨慎考虑其适用场景和潜在问题。

尽管 SMOTE 存在这些限制，但它被广泛采用，并催生了多种旨在解决这些缺点的扩展技术。例如，Borderline-SMOTE[72]和 SVM-SMOTE[73] 都关注于类的边界样本，但它们仍未能同时解决噪声产生和类内不平衡问题。此外，这些改进技术因复杂性高而难以实施和使用。

K-means SMOTE[74] 结合了简单且广为人知的 k-means 聚类算法和 SMOTE 过采样技术来重新平衡数据集。该方法专注于只在被认为是安全的区域内（类内样本间）进行过采样，从而减少噪声的生成。它通过增加少数类别中的样本数量，有效提升了数据集的平衡性。因此，这种方法不仅解决了类别间的不平衡问题，还成功应对了类别内部稀疏性所带来的挑战。由于结合了 k-means 和 SMOTE 算法的优点，该方法不仅易于实施，而且相较于其他技术，其复杂度相对较低。值得一提的是，该方法的独特之处在于它依据聚类的分布来生成新的样本，这一点与其他方法有着显著的区别。通过这种方式，该方法能够更准确地反映数据的真实分布，从而提高分类算法的性能。

本文将 k-means Smote[74]方法运用到算法中，用以弥补随机采样加速器造成的类别不平衡的问题。

4.2 本章工作

为了解决 4.1 节中所述的元启发式算法在样本较多时耗时过多的问题，本节将提出随机采样加速器，从样本角度分析如何实现该加速器。接着介绍 k-means Smote 方法。最后，将 k-means Smote 方法添加进随机采样加速器中。

4.2.1 随机采样加速器

首先，将整个样本空间随机分为若干组。接着，遍历样本组且使用算法求解特征集。然后，评估此时所选的特征集是否满足约束条件，如近似质量达到阈值等。若满足直接添加当前样本组后的样本组，此时无须再使用算法进行特征选择；否则，要在添加样本组后仍须再使用算法进行特征选择，且此过程是在上一次特征选择结果的引导下进行。之后进行下一轮评估，如此循环，直到遍历完所有样本组。一般情况下，选出合适的特征集不需要遍历整个样本集。显然，随机采样加速器在样本层面进行了加速。

为了便于理解我们所提算法的主要原理，以下给出一个简单的示例。

例 1 对于表 4.1 所示的决策系统 $DS = <U, AT \cup \{d\}>$, 其中 $U=\{x_1, x_2, x_3, x_4\}$, $AT=\{a_1, a_2, a_3, a_4\}$。

表 4.1 决策系统示例

Table 4.1 A example of decision system

	a1	a2	a3	a4	d
x1	1.0	1.2	1.2	1.4	1
x2	1.2	1.0	1.0	1.2	1
x3	1.0	1.8	1.8	1.6	2
x4	1.2	2.0	2.0	1.8	2

具体计算过程如下所示：

(1) 随机对样本进行分组，设置分组数为 2, 假设样本划分为$\{x_1, x_2\}$和$\{x_3, x_4\}$。

(2) 使用第一个分组使用算法进行特征选择结果集的求解，得到第一次特征选择的结果集$\{a_1\}$。

(3) 对特征选择结果进行度量，发现不满足要求，继续添加样本组$\{x_3, x_4\}$, 且再次使用算法进行特征选择的过程(在上一次的特征选择集$\{a_1\}$基础上), 得到$\{a_1, a_2, a_4\}$。

(4) 对步骤 3 得到结果进行度量，满足特征选择的条件，循环结束。

4.2.2 **K-means Smote 方法**

K-means SMOTE[74]是一种面向处理类别不平衡数据集的过采样技术。它旨在通过在数据输入空间的安全和关键区域内生成属于少数类别的样本来增强分类效果。此方法有效地避免了噪声产生，同时解决了类别间和类别内部的不平衡问题。K-means SMOTE 的实现流程分为三个要步骤：首先，采用 *k* 均值聚类算法对整体数据集进行聚类处理。接着，识别并选择那些少数类样本较多的簇，以便在这些簇中分配更多的合成样本。对于少数类样本稀疏分布的簇，将会有更多合成样本被加入。最后，每个经过筛选的簇采用 SMOTE 算法进行过采样处理，以此来增强数据集的类别平衡。因此，k-means SMOTE 通过其有针对性的方法有效地应对了类别不平衡问题，优于随机采样加速器可能带来的噪声和过拟合挑战。它通过以下关键策略实现这一目标：

(1) 有目标的样本创造：与随机采样加速器通过随机分割样本空间不同，k-means SMOTE 专注于精准识别和增加少数类样本在输入空间中的分布，保障新样本有效缓解类别不平衡。

(2) 控制质量的过采样：通过在少数类样本间进行插值创建新实例，k-means SMOTE 在生成新样本时注重样本相似性，这比简单复制少数类样本更能保持数据集的原始分布特征，降低过拟合风险。

(3) 平衡稀疏分布与强化少数类：k-means SMOTE 直接强化少数类样本的稀疏区域，解决类内不平衡问题，帮助分类模型更好地识别少数类别。

(4) 聚类与过采样的综合方法：结合聚类和过采样优势的 k-means SMOTE 策略不仅提速处理过程，也增强了解决类别不平衡的能力。这与主要加速特征选择的随机采样加速器不同，k-means SMOTE 提供了一种针对类别不平衡问题的直接解决方案。

总体而言，k-means SMOTE 的结构化及有目的性方法，有效克服了随机采样加速器忽视的类别不平衡挑战，展示了一种兼顾数据质量和处理速度的解决方案。

4.2.3 基于随机采样加速器的快速元启发式算法

首先，我们将整体样本空间进行随机分组，形成若干个样本组。随后，我们开始遍历这些样本组，并应用算法来求解特征集。完成特征集的求解后，我们对其进行评估，检查是否满足预设的约束条件，例如近似质量是否达到特定的阈值。如果当前样本组的特征集满足约束条件，我们则直接将其添加至已选择的样本组集合中，无需再次使用算法进行特征选择。然而，若不满足条件，我们则需要在添加该样本组后，再次使用算法进行特征选择。这一过程是基于前一次特征选择的结果进行的，能够更有效地指导后续的选择过程。接着，我们继续进行下一轮的评估与选择。最后循环此过

程，直至遍历完所有的样本组。值得注意的是，通常情况下，我们并不需要遍历整个样本集就能选出合适的特征集。因此，随机采样加速器在样本层面实现了显著的加速效果。根据章节 4.2.1 的讨论，随机采样加速器将在这一章节与元启发式算法相结合，基于随机采样加速器的快速元启发式算法 (Fast Meta-heuristic Algorithm based on Random Sampling Accelerator, RS) 的流程如下算法 4.1 所示。

算法 4.1： 基于随机采样加速器的快速元启发式算法。(RS)

输入： 决策系统 DS = <*U*, *AT*∪{*d*}>, 约束条件 C_{ρ}^{U};

输出： *A*.

步骤 1: 初始化，令 *A*=∅, 数据集 *U* 随机分为 *m* 组 $U_1, \cdots, U_m$;

步骤 2: 利用 U_1 使用上述四种元启发式算法中的一种求解特征选择集 A_1;

步骤 3: **For** *i* = 2 : *m* **Do**

令 $U' = \sum_{k=1}^{i} U_k$, A_i=$A_{i\text{-}1}$; //在上次特征选择集的指导下继续特征选择

ρ=ρ(U', A_i); //计算此时的特征选择结果集的度量值

While $C_{\rho}^{U'} \le \rho$ 不满足 **Do**

利用元启发式算法从剩余特征中继续挑选出特征组 *b*;

通过计算 $\rho(U', A \cup \{b\})$评估特征组 *b*;

$A_i = A_i \cup \{b\}$;

End While

End For

步骤 4: 输出 *A*.

其中，$\rho(U', A)$ 表示特征组 *A* 在样本集合 U' 下评估出的度量值，这里使用近似质量作为度量。按照算法 4.1 的过程，最坏的情况包括两个方面：(1) 每次迭代至少会添加一个特征到特征选择结果集中；(2) *AT* 中的所有特征都是最终特征选择结果集所必需的。因此，由上述元启发式算法的时间复杂度可知，算法 4.1 的最坏复杂度为：$T(n) = O\left(\sum_{i=1}^{m} n_i\right)$, 其中 $n_i = (\frac{i}{m} \cdot |U| \cdot T_i)^2$ $(i = 1, 2, \ldots, m)$ 是算法 4.1 在第 *i* 次循环中的时间消耗。*T* 表示整个样本集下使用的元启发式算法所需的迭代数，$T_i (i = 1, 2, \ldots, m)$ 为每次分组使用的元启发式算法的迭代数，$T = T_1 + T_2 + \ldots + T_m$. 综上，$T(n) \le O\left(|U|^2 \cdot N \cdot T\right)$.

对于前两个算法(算法 2.1，算法 2.2)显然产生了加速，对于后两个算法(算法 2.3，算法 2.4)，它们只是在此基础上进行了部分累乘，依然可得出加速效果。因此，该算法能够产生加速效果。

4.2.4 基于随机采样加速器和重采样的快速元启发式算法

基于随机采样加速器和重采样的快速元启发式算法(Fast Meta-heuristic Algorithm based on Random Sampling Accelerator and Resampling, KRS)是一种结合了随机采样和 k-means SMOTE 重采样技术的元启发式算法。在这种算法中，随机采样用于快速生成候选解，而 k-means SMOTE 重采样则用于增强搜索的类别平衡。

具体而言，该算法首先通过随机采样从原始数据中获取候选样本，然后，通过对这些候选样本进行评估和筛选，采用 k-means 聚类算法对这些数据集进行聚类处理。接着，识别并选择那些少数类样本数量较多的簇，以便在这些簇中分配更多的合成样本。对于少数类样本稀疏分布的簇，将会有更多合成样本被加入。最后，对每个经过筛选的簇采用 SMOTE 算法进行过采样处理，从而进行样本扩展和调整，以探索更广泛的解空间。这种基于随机采样和重采样的元启发式算法能够在保持解的多样性的同时，加快搜索过程并提高算法的全局搜索能力。下面的图 4.1 给出了 k-means Smote 的方法进行重采样弥补随机采样加速器的实现过程。

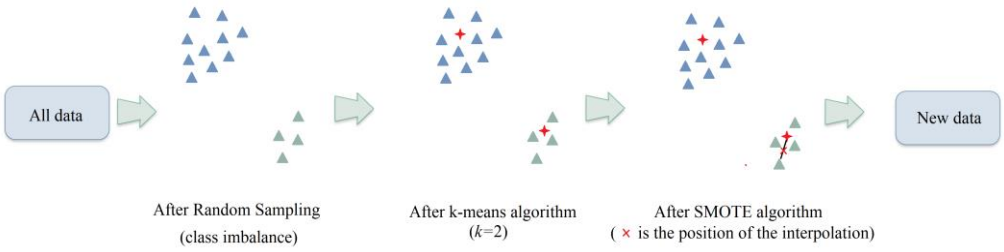

图 4.1 k-means Smote 的方法进行重采样的实现过程

Fig.4.1 The implementation process of k-means Smote method for resampling

下面的算法 4.2 给出了基于随机采样加速器和重采样的快速元启发式算法的实现过程。

算法 4.2： 基于随机采样加速器和重采样的快速元启发式算法。(KRS)

输入： 决策系统 DS = <$U, AT \cup \{d\}$>, 约束条件 C_ρ^U;

输出： A.

步骤 1: 初始化，令 $A=\varnothing$, 数据集 U 随机分为 m 组 $U_1, \cdots, U_m$;

步骤 2: 对 U_1 采用 k-means Smote 的方法进行重采样获得新的 U_1;

步骤 3: 利用 U_1 使用上述四种元启发式算法中的一种求解特征选择集 A_1;

步骤 4: **For** i =2 : m **Do**

令 $U' = \sum_{k=1}^{i} U_k$, A_i=A_{i-1}; //在上次特征选择集的指导下继续特征选择

对 U' 采用 k-means Smote 的方法进行重采样获得新的 U';

ρ=ρ($U^{'}$, A_i); //计算此时的特征选择结果集的度量值

While $C_{\rho}^{U^{'}} \le \rho$ 不满足 **Do**

利用元启发式算法从剩余特征中继续挑选出特征组 *b*;

通过计算 $\rho(U^{'}, A \cup \{b\})$评估特征组 *b*;

$A_i = A_i \cup \{b\}$;

End While

End For

步骤 5: 输出 *A*.

由上述算法 4.1 的时间复杂度可知，算法 4.2 的最坏时间复杂度也为：$T(n) = O\left(\sum_{i=1}^{m} n_i\right)$，其中 $n_i = (\frac{i}{m} \cdot |U| \cdot T_i)^2$ $(i = 1, 2, \ldots, m)$ 是算法 4.1 在第 i 次循环中的时间消耗。T 表示整个样本集下使用的元启发式算法所需的迭代数，$T_i (i = 1, 2, \ldots, m)$ 为每次分组使用的元启发式算法的迭代数，$T = T_1 + T_2 + \ldots + T_m$. 综上，$T(n) \le O\left(|U|^2 \cdot N \cdot T\right)$. 同样地，对于四个元启发式算法 (算法 2.1-2.4) 也产生了加速效果。

4.3 实验分析

4.3.1 实验数据

本节从 UCI 标准数据集中选取 20 个数据集进行实验，以证明本文算法的有效性。数据的具体信息见下表 4.2。所有实验均在一台装有 Windows 10, CPU 为 Intel® Core(TM) i5-7300HQ (2.50 GHz)和 8.00 GB 内存的个人计算机上进行。实验平台为 MATLAB R2018a。

表 4.2 数据集描述

Table 4.2 Description of data sets

ID	数据集名称	样本数	特征数	决策类数	来源
1	Cardiotocography	2126	21	10	UCI
2	SPECTF Heart	267	44	2	UCI
3	Statlog (Vehicle Silhouettes)	846	18	4	UCI
4	Wdbc	569	30	2	UCI
5	Dermatology	366	34	6	UCI
6	Forest Type Mapping	523	27	4	UCI
7	Wine	178	13	3	UCI

（续表 4.2）

ID	数据集名称	样本数	特征数	决策类数	来源
8	Urban Land Cover	675	147	9	UCI
9	German	1000	24	2	UCI
10	QSAR Biodegradation	1055	41	2	UCI
11	Synthetic Control Chart Time Series	600	60	6	UCI
12	Climate Model Simulation Crashes	540	20	2	UCI
13	Statlog (Heart)	270	13	2	UCI
14	Ionosphere	351	34	2	UCI
15	waveform	5000	21	3	UCI
16	Wall-Following Robot Navigation	5456	24	4	UCI
17	Pen-Based Recognition of Handwritten Digits	10992	17	10	UCI
18	MAGIC Gamma Telescope	19020	11	2	UCI
19	twonorm	7400	20	2	UCI
20	Crowdsourced Mapping	10845	29	6	UCI

4.3.2 实验设置

在本节实验中，我们采用邻域粗糙集理论来定义特征选择的形式。本研究旨在通过筛选掉无关或重复的特征，保证最终特征集所计算的近似质量至少可以达到原始特征集近似质量的 95%。此外，为了精确控制邻域的规模，本实验选定了从 0.02 到 0.40 的 20 个不同的半径值。同时，采用 10 折交叉验证方法对特征选择进行验证，并在测试集上评估了采用不同特征集的分类器性能表现。实验使用 KNN (k=3)和 CART (默认参数)两种分类器对测试数据集进行分类。此外，将元启发式算法生成的初始种群中个体的初始长度设置为 $10\,\% \times |AT|$。

我们选取了两种其他的加速算法来对比我们的算法 4.1 和算法 4.2 在四种元启发式算法上的表现，他们分别为：参考文献[68]中的 BBSAR 算法、参考文献[69]中的算法 AGAR。

4.3.3 实验结果

值得注意的是，本节采用四种元启发式算法与四种加速算法分别结合，并在 20

个 UCI 数据中将被互相对比。

表 4.3 在加速 NMBO 上的时间消耗

Table 4.3 Time consumption of accelerating MBO

ID	NMBO	BBSAR-NMBO	AGAR-NMBO	RS-NMBO	KRS-NMBO
1	47.9497	10.6587	21.1767	**0.9719**	2.0477
2	0.1042	0.0825	0.0851	**0.0347**	0.0377
3	4.3831	2.9924	4.1131	**0.1790**	0.1815
4	0.5040	0.3792	0.4842	**0.0818**	0.1050
5	0.1201	0.0585	0.0536	**0.0359**	0.0391
6	2.0721	0.7942	1.5123	**0.1681**	0.2697
7	0.0405	0.0361	0.0377	**0.0227**	0.0230
8	0.2791	0.1155	0.2300	**0.0875**	0.0924
9	0.8446	0.7726	0.7853	**0.2212**	0.2265
10	6.8049	3.8926	2.5672	**0.3404**	0.5464
11	0.1398	0.0889	0.1058	**0.0548**	0.0576
12	0.0793	0.0685	0.0556	**0.0335**	0.0363
13	0.0581	0.0404	0.0561	**0.0280**	0.0292
14	0.0398	0.0374	0.0348	**0.0321**	0.0322
15	36.8335	15.1948	25.8980	**3.1411**	4.3622
16	49.8467	18.2329	35.8514	**5.5934**	5.7853
17	253.0885	154.0934	102.5919	**35.5560**	38.7391
18	1325.1548	335.4182	407.3417	**83.8554**	88.6435
19	294.4798	54.0676	64.0214	**7.7634**	8.1811
20	109.6315	78.4897	48.4633	**16.2997**	17.6485
平均	106.6227	33.7757	35.7733	**7.7250**	8.3542

表 4.4 在加速 NGA 上的时间消耗

Table 4.4 Time consumption of accelerating NGA

ID	NGA	BBSAR-NGA	AGAR-NGA	RS-NGA	KRS-NGA
1	136.1728	28.2545	38.6572	**12.9745**	14.4653
2	1.2168	0.7937	0.7016	**0.1472**	0.1601
3	40.6925	16.9453	10.2096	**2.8734**	3.8912
4	20.2095	6.3906	5.4026	**0.9260**	1.0533
5	0.4513	0.3366	0.3806	**0.1757**	0.1806

（续表 4.4）

ID	NGA	BBSAR-NGA	AGAR-NGA	RS-NGA	KRS-NGA
6	57.0385	18.3569	11.9796	**2.6186**	3.5155
7	0.2770	0.2203	0.1949	**0.1459**	0.1785
8	1.6414	0.9838	1.0034	**0.3122**	0.4376
9	5.3340	4.0824	4.7329	**0.7601**	0.7978
10	211.8428	96.2791	108.4427	**18.7692**	22.6075
11	1.0032	0.8656	0.8114	**0.2414**	0.2446
12	0.7359	0.6824	0.6943	**0.1975**	0.1989
13	1.3016	0.9680	1.1528	**0.2502**	0.3619
14	0.3978	0.3286	0.3356	**0.1839**	0.2319
15	222.3042	65.6888	76.7288	**14.2383**	20.2654
16	1234.5499	314.9995	580.0055	**53.5822**	61.5606
17	2335.7810	1056.2990	743.9978	**103.7522**	131.2465
18	3244.6915	866.4063	1136.4944	**187.9952**	238.3027
19	746.3525	194.7481	165.9483	**28.1250**	32.3213
20	967.2155	196.6205	397.1485	**56.0428**	70.3337
平均	461.4605	143.5125	164.2511	**24.2156**	30.1177

表 4.5 在加速 NAFSA 上的时间消耗

Table 4.5 Time consumption of accelerating NAFSA

ID	NAFSA	BBSAR-NAFSA	AGAR-NAFSA	RS-NAFSA	KRS-NAFSA
1	212.2761	84.7308	54.9099	**6.1040**	6.7913
2	4.9257	3.1112	2.5892	**0.7775**	0.9819
3	14.3639	4.9652	7.9768	**0.8242**	1.0570
4	12.2268	7.2160	6.2317	**1.2486**	1.5207
5	0.2672	0.1562	0.1764	**0.0500**	0.0673
6	15.1733	10.7858	8.6051	**1.7442**	2.3897
7	0.1680	0.1415	0.1353	**0.0418**	0.0477
8	0.6731	0.3469	0.2879	**0.1687**	0.2218
9	14.0919	8.8309	12.9939	**0.7607**	0.9007
10	580.2302	326.0675	439.5683	**10.5029**	13.9668
11	0.2366	0.1627	0.1965	**0.0811**	0.0924
12	0.1400	0.1069	0.0891	**0.0398**	0.0540

（续表 4.5）

ID	NAFSA	BBSAR-NAFSA	AGAR-NAFSA	RS-NAFSA	KRS-NAFSA
13	0.6021	0.5253	0.5328	**0.0827**	0.1229
14	1.1996	0.9082	1.0613	**0.1739**	0.2323
15	1079.4309	388.4024	503.6303	**18.4590**	20.4273
16	2897.5100	1043.4301	975.8554	**66.6854**	71.0961
17	1081.7891	453.4282	372.5733	**90.5613**	103.3304
18	3398.1025	851.7828	880.6257	**147.5171**	179.1743
19	772.4650	175.3251	205.3063	**27.1360**	29.1898
20	942.1020	295.3668	386.4400	**50.6506**	64.5137
平均	551.3987	182.7895	192.9893	**21.1805**	24.8089

表 4.6 在加速 NFOA 上的时间消耗

Table 4.6 Time consumption of accelerating NFOA

ID	NFOA	BBSAR-NFOA	AGAR-NFOA	RS-NFOA	KRS-NFOA
1	304.8315	126.9338	104.2195	**7.7438**	10.5331
2	2.1948	1.0452	1.1342	**0.1528**	0.1933
3	51.6666	16.3146	12.9176	**1.8570**	2.3621
4	13.7486	6.4321	10.6545	**0.5161**	0.5295
5	0.5253	0.3456	0.4239	**0.0680**	0.0756
6	28.4386	10.8428	8.4578	**2.0465**	2.2966
7	0.2177	0.1650	0.1609	**0.0422**	0.0477
8	3.6413	1.9416	2.0635	**0.3844**	0.4302
9	13.9799	9.3674	11.5629	**0.4420**	0.5992
10	12.2091	7.5295	10.1236	**1.7203**	2.0910
11	1.6109	1.3919	1.3800	**0.1651**	0.2173
12	0.4668	0.3985	0.3082	**0.0595**	0.0740
13	0.8506	0.7998	0.6490	**0.1022**	0.1416
14	0.6321	0.5279	0.5926	**0.0760**	0.1107
15	3471.1192	1510.2986	1066.1995	**27.5292**	36.4844
16	1188.5205	389.4490	239.6307	**17.6723**	25.8245
17	3121.2945	917.3263	738.5057	**160.6980**	170.2435
18	4100.7325	991.4011	1148.4394	**248.7847**	327.8485
19	1101.5132	231.2743	270.8550	**66.8548**	77.0234
20	2056.6406	824.5031	550.4430	**131.5072**	192.5923

平均	773.7417	252.4144	208.9361	**33.4211**	42.4859

通过观察表 4.3-4.6，不难得出以下结论：

(1) 与四种元启发式算法、BBSAR 和 AGAR 进行特征选择的时间进行比较，我们提出的算法 RS 明显需要更少的时间来进行特征选择。对比总体的时间消耗均值时，使用算法 2.1-2.4 进行特征选择的时间分别需要 106.6227、461.4605、551.3987、773.7417 秒；使用 BBSAR 加速四种元启发式算法的时间分别为 33.7757、118.5567、136.6739、252.4144 秒；使用 AGAR 加速四种元启发式算法的时间分别为 35.7733、164.2511、192.9893、208.93618 秒。然而，我们提出的加速四种元启发式进行特征选择的算法 RS 仅仅花费 7.7250、24.2156、21.1805、33.4211 秒。鉴于此，我们的方法的确提高了进行特征选择的时间效率。

(2) 通过 k-means SMOTE 重采样弥补 RS 算法增加了一些时间消耗。对比总体的时间消耗均值时，使用 KRS 算法进行特征选择的时间分别需要 8.3542、30.1177、24.8089、42.4859 秒，对比 RS 算法仅仅分别增加了 8.14%、24.37%、17.13%、27.12%.

为了清晰地展示在 KNN 和 CART 分类器上不同算法得出的特征选择结果集的分类能力，将被用来展示不同数据集上的结果，如表 4.7-4.14 所示。

表 4.7 在 NMBO 和 NGA 上的 KNN 分类稳定性

Table 4.7 KNN classification stability on NMBO and NGA

ID	NMBO	BBSAR-NMBO	AGAR-NMBO	RS-NMBO	KRS-NMBO	NGA	BBSAR-NGA	AGAR-NGA	RS-NGA	KRS-NGA
1	**0.8443**	0.8327	0.8387	0.8100	0.8393	**0.8498**	0.8387	0.8319	0.8331	0.8493
2	**0.7698**	0.7672	0.7482	0.7462	0.7640	0.7303	0.7259	0.7224	0.7189	**0.7367**
3	**0.8273**	0.8144	0.8074	0.7960	0.8182	**0.8073**	0.7948	0.7958	0.7841	0.8029
4	**0.9703**	0.9613	0.9389	0.9420	0.9548	**0.9656**	0.9497	0.9381	0.9421	0.9607
5	**0.9063**	0.8897	0.8857	0.8814	0.8981	**0.8876**	0.8721	0.8586	0.8534	0.8763
6	0.9235	0.9222	0.9161	0.9245	**0.9251**	**0.9327**	0.9145	0.9202	0.9276	0.9475
7	**0.9229**	0.9201	0.9143	0.8628	0.9016	**0.9330**	0.9170	0.8921	0.9071	0.9478
8	0.8576	0.8495	0.8538	0.8377	**0.8662**	**0.8593**	0.8491	0.8252	0.8354	0.8467
9	0.7382	0.7310	0.7275	0.7395	**0.7586**	0.7294	0.7200	0.7145	0.7241	**0.7358**
10	0.9038	0.9029	0.8812	0.8828	**0.9051**	**0.9219**	0.9153	0.9122	0.8709	0.9064
11	0.9060	0.8966	0.8658	0.9029	**0.9217**	**0.9104**	0.8991	0.8910	0.8801	0.9032
12	**0.9407**	0.9283	0.9037	0.9285	0.9302	**0.9561**	0.9475	0.9396	0.9406	0.9480
13	**0.7563**	0.7471	0.7371	0.6956	**0.7587**	0.8510	0.8506	0.8417	0.8096	**0.8534**
14	0.8861	0.8808	0.8674	0.8701	**0.9053**	**0.9376**	0.9287	0.9034	0.9006	0.9297
15	**0.5870**	0.5789	0.5773	0.5440	0.5645	**0.7602**	0.7500	0.7241	0.7447	0.7519

(续表 4.7)

ID	NMBO	BBSAR-NMBO	AGAR-NMBO	RS-NMBO	KRS-NMBO	NGA	BBSAR-NGA	AGAR-NGA	RS-NGA	KRS-NGA
16	**0.5542**	0.5520	0.5476	0.5338	0.5411	**0.8839**	0.8716	0.8517	0.8690	0.8829
17	0.9227	0.9147	0.9114	0.9014	**0.9282**	**0.9378**	0.9256	0.9261	0.9054	0.9251
18	0.8721	0.8699	0.8592	0.8644	**0.8770**	**0.8869**	0.8788	0.8468	0.8804	0.8818
19	**0.9067**	0.9053	0.9019	0.8854	0.8973	**0.8935**	0.8838	0.8834	0.8762	0.8872
20	**0.9394**	0.9235	0.9153	0.9245	0.9278	**0.9447**	0.9323	0.9151	0.9264	0.9423
平均	**0.8468**	0.8394	0.8299	0.8237	0.8441	**0.8790**	0.8683	0.8567	0.8565	0.8758

表 4.8 在 NAFSA 和 NFOA 上的 KNN 分类稳定性

Table 4.8 KNN classification stability on NAFSA and NFOA

ID	NAFSA	BBSAR-NAFSA	AGAR-NAFSA	RS-NAFSA	KRS-NAFSA	NFOA	BBSAR-NFOA	AGAR-NFOA	RS-NFOA	KRS-NFOA
1	**0.8295**	0.8157	0.8057	0.7932	0.8193	**0.8591**	0.8535	0.8461	0.8403	0.8525
2	**0.7616**	0.7519	0.7517	0.7313	0.7541	**0.7866**	0.7804	0.7538	0.7644	0.7779
3	**0.8040**	0.7883	0.7771	0.7784	0.7953	**0.8082**	0.7980	0.7981	0.7773	0.8008
4	**0.9622**	0.9516	0.9519	0.9269	0.9505	**0.9644**	0.9586	0.9439	0.9336	0.9591
5	**0.8045**	0.7932	0.7873	0.7750	0.7942	**0.8630**	0.8507	0.8387	0.8384	0.8587
6	**0.9230**	0.9053	0.8817	0.9012	0.9104	**0.9108**	0.9029	0.8802	0.8837	0.9031
7	**0.8860**	0.8746	0.8605	0.8315	0.8776	**0.9109**	0.8981	0.8885	0.8489	0.9008
8	**0.8513**	0.8411	0.8219	0.8281	0.8460	**0.8267**	0.8223	0.7931	0.7980	0.8172
9	**0.7187**	0.7058	0.7085	0.6976	0.7109	**0.7330**	0.7283	0.7218	0.7056	0.7281
10	**0.9033**	0.8892	0.8651	0.8852	0.8955	**0.9023**	0.8920	0.8765	0.8687	0.8921
11	**0.8951**	0.8776	0.8712	0.8640	0.8856	**0.8909**	0.8791	0.8798	0.8726	0.8860
12	**0.9223**	0.9131	0.9078	0.8920	0.9100	**0.9397**	0.9347	0.9203	0.9168	0.9350
13	**0.8345**	0.8234	0.8248	0.7881	0.8263	**0.8374**	0.8278	0.8214	0.7738	0.8315
14	**0.9074**	0.8948	0.8663	0.8867	0.8941	**0.8750**	0.8630	0.8534	0.8440	0.8669
15	**0.7414**	0.7338	0.7091	0.7134	0.7333	**0.7811**	0.7730	0.7660	0.7561	0.7721
16	**0.8983**	0.8884	0.8881	0.8679	0.8875	**0.8610**	0.8551	0.8430	0.8393	0.8536
17	**0.9351**	0.9232	0.9120	0.8936	0.9258	**0.9372**	0.9296	0.8996	0.9082	0.9261
18	**0.8918**	0.8756	0.8590	0.8597	0.8832	**0.9309**	0.9235	0.9027	0.8992	0.9233
19	**0.8736**	0.8638	0.8586	0.8528	0.8683	**0.8956**	0.8886	0.8593	0.8736	0.8906
20	**0.9331**	0.9216	0.9076	0.9053	0.9243	**0.9407**	0.9339	0.9108	0.9181	0.9312
平均	**0.8638**	0.8516	0.8408	0.8336	0.8546	**0.8727**	0.8647	0.8499	0.8430	0.8653

表 4.9 在 NMBO 和 NGA 上的 CART 分类稳定性

Table 4.9 CART classification stability on NMBO and NGA

ID	NMBO	BBSAR-NMBO	AGAR-NMBO	RS-NMBO	KRS-NMBO	NGA	BBSAR-NGA	AGAR-NGA	RS-NGA	KRS-NGA
1	**0.8112**	0.7994	0.7807	0.7814	0.7951	**0.8060**	0.7926	0.7797	0.7751	0.7891
2	**0.7190**	0.7078	0.7004	0.6932	0.7046	**0.7159**	0.7100	0.6895	0.6867	0.7033
3	**0.7125**	0.7043	0.6867	0.6964	0.6998	**0.6899**	0.6810	0.6718	0.6646	0.6781
4	**0.9260**	0.9126	0.8995	0.9019	0.9140	**0.8997**	0.8918	0.8822	0.8702	0.8869
5	**0.8944**	0.8809	0.8757	0.8604	0.8829	**0.8696**	0.8606	0.8535	0.8396	0.8553
6	**0.8617**	0.8511	0.8323	0.8328	0.8461	**0.8655**	0.8578	0.8424	0.8438	0.8480
7	**0.8301**	0.8192	0.7976	0.7764	0.8200	**0.8272**	0.8127	0.7974	0.7587	0.8178
8	**0.7756**	0.7656	0.7511	0.7573	0.7662	**0.7965**	0.7869	0.7785	0.7666	0.7857
9	**0.6980**	0.6888	0.6820	0.6797	0.6901	**0.6860**	0.6784	0.6720	0.6620	0.6780
10	**0.8051**	0.7944	0.7832	0.7765	0.7925	**0.8068**	0.7978	0.7810	0.7838	0.7943
11	**0.8060**	0.7994	0.7897	0.7790	0.7926	**0.8172**	0.8071	0.7829	0.7981	0.8079
12	**0.8474**	0.8366	0.8210	0.8207	0.8332	**0.8858**	0.8730	0.8647	0.8638	0.8757
13	**0.7080**	0.6958	0.6835	0.6506	0.6971	**0.7541**	0.7436	0.7224	0.7015	0.7424
14	**0.8803**	0.8659	0.8638	0.8579	0.8712	**0.8706**	0.8582	0.8345	0.8457	0.8597
15	**0.6102**	0.6033	0.5895	0.5906	0.5995	**0.6883**	0.6766	0.6682	0.6666	0.6805
16	**0.7903**	0.7779	0.7717	0.7700	0.7788	**0.9798**	0.9629	0.9503	0.9455	0.9633
17	**0.9174**	0.9030	0.8944	0.8964	0.8978	**0.9001**	0.8879	0.8745	0.8646	0.8897
18	**0.8123**	0.8007	0.7789	0.7819	0.7992	**0.8310**	0.8179	0.8015	0.8111	0.8186
19	**0.7785**	0.7691	0.7483	0.7525	0.7641	**0.7584**	0.7510	0.7330	0.7355	0.7473
20	**0.8446**	0.8379	0.8199	0.8237	0.8305	**0.8455**	0.8382	0.8259	0.8271	0.8349
平均	**0.8014**	0.7907	0.7775	0.7740	0.7888	**0.8147**	0.8043	0.7855	0.8028	

表 4.10 在 NAFSA 和 NFOA 上的 CART 分类稳定性

Table 4.10 CART classification stability on NAFSA and NFOA

ID	NAFSA	BBSAR-NAFSA	AGAR-NAFSA	RS-NAFSA	KRS-NAFSA	NFOA	BBSAR-NFOA	AGAR-NFOA	RS-NFOA	KRS-NFOA
1	**0.7959**	0.7834	0.7627	0.7727	0.7834	**0.7909**	0.7770	0.7617	0.7651	0.7814
2	**0.7565**	0.7452	0.7275	0.7343	0.7461	**0.7365**	0.7304	0.7176	0.7141	0.7211
3	**0.7310**	0.7203	0.7009	0.7101	0.7233	**0.6862**	0.6768	0.6601	0.6627	0.6746
4	**0.8979**	0.8844	0.8752	0.8661	0.8830	**0.8850**	0.8770	0.8673	0.8655	0.8755

（续表 4.10）

ID	NAFSA	BBSAR-NAFSA	AGAR-NAFSA	RS-NAFSA	KRS-NAFSA	NFOA	BBSAR-NFOA	AGAR-NFOA	RS-NFOA	KRS-NFOA
5	**0.8127**	0.8047	0.7910	0.7873	0.7961	**0.8365**	0.8260	0.8120	0.8117	0.8187
6	**0.8839**	0.8763	0.8522	0.8539	0.8718	**0.8694**	0.8554	0.8520	0.8388	0.8587
7	**0.8783**	0.8689	0.8427	0.8062	0.8596	**0.8421**	0.8324	0.8076	0.7832	0.8316
8	**0.7653**	0.7538	0.7448	0.7446	0.7560	**0.7463**	0.7332	0.7195	0.7248	0.7313
9	**0.6919**	0.6842	0.6750	0.6736	0.6778	**0.6868**	0.6790	0.6588	0.6659	0.6779
10	**0.8132**	0.7990	0.7969	0.7962	0.8026	**0.8024**	0.7939	0.7753	0.7734	0.7856
11	**0.8060**	0.7990	0.7889	0.7784	0.7921	**0.8069**	0.7983	0.7907	0.7875	0.7898
12	**0.8371**	0.8242	0.8119	0.8138	0.8229	**0.8749**	0.8641	0.8556	0.8488	0.8621
13	**0.7819**	0.7687	0.7625	0.7203	0.7730	**0.7560**	0.7493	0.7304	0.6978	0.7418
14	**0.8733**	0.8612	0.8432	0.8503	0.8641	**0.8105**	0.8000	0.7769	0.7820	0.8022
15	**0.6691**	0.6614	0.6418	0.6537	0.6572	**0.6974**	0.6880	0.6711	0.6809	0.6868
16	**0.9748**	0.9625	0.9459	0.9514	0.9637	**0.9396**	0.9279	0.9029	0.9121	0.9237
17	**0.9017**	0.8875	0.8843	0.8814	0.8833	**0.9131**	0.9010	0.8931	0.8900	0.9013
18	**0.8298**	0.8156	0.8100	0.8040	0.8158	**0.7966**	0.7893	0.7794	0.7736	0.7824
19	**0.7492**	0.7383	0.7257	0.7252	0.7342	**0.7505**	0.7406	0.7196	0.7317	0.7387
20	**0.8389**	0.8309	0.8170	0.8109	0.8224	**0.8246**	0.8172	0.7966	0.8017	0.8091
平均	**0.8144**	0.8035	0.7900	0.7867	0.8014	**0.8026**	0.7928	0.7774	0.7756	0.7897

表 4.11 在 NMBO 和 NGA 上的 KNN 分类准确率

Table 4.11 KNN classification accuracy on NMBO and NGA

ID	NMBO	BBSAR-NMBO	AGAR-NMBO	RS-NMBO	KRS-NMBO	NGA	BBSAR-NGA	AGAR-NGA	RS-NGA	KRS-NGA
1	**0.7401**	0.7318	0.7224	0.7148	0.7391	**0.7426**	0.7339	0.7275	0.7217	0.7365
2	**0.7719**	0.7672	0.7459	0.7583	0.7711	**0.7554**	0.7464	0.7293	0.7357	0.7478
3	**0.6562**	0.6520	0.6403	0.6457	0.6494	**0.6343**	0.6308	0.6157	0.6137	0.6282
4	**0.9696**	0.9610	0.9501	0.9511	0.9655	**0.9550**	0.9432	0.9273	0.9346	0.9418
5	**0.9311**	0.9204	0.9045	0.8997	0.9229	**0.9300**	0.9173	0.9093	0.9133	0.9168
6	**0.8704**	0.8579	0.8510	0.8475	0.8602	**0.8664**	0.8577	0.8436	0.8389	0.8524
7	**0.9433**	0.9298	0.9286	0.8908	0.9401	**0.9333**	0.9213	0.9144	0.8887	0.9241
8	**0.8012**	0.7971	0.7819	0.7740	0.7939	**0.7427**	0.7319	0.7235	0.7271	0.7297
9	**0.6658**	0.6608	0.6537	0.6447	0.6635	**0.6924**	0.6838	0.6751	0.6782	0.6822

（续表 4.11）

ID	NMBO	BBSAR-NMBO	AGAR-NMBO	RS-NMBO	KRS-NMBO	NGA	BBSAR-NGA	AGAR-NGA	RS-NGA	KRS-NGA
10	**0.8411**	0.8359	0.8164	0.8112	0.8373	**0.8530**	0.8472	0.8387	0.8374	0.8400
11	**0.9458**	0.9330	0.9163	0.9251	0.9338	**0.9453**	0.9356	0.9311	0.9194	0.9349
12	**0.9065**	0.9006	0.8924	0.8827	0.8974	**0.8819**	0.8702	0.8632	0.8512	0.8709
13	**0.7248**	0.7211	0.7110	0.6738	0.7143	**0.8137**	0.8021	0.7974	0.7670	0.8027
14	**0.8356**	0.8295	0.8206	0.8183	0.8302	**0.8709**	0.8624	0.8569	0.8473	0.8602
15	**0.6526**	0.6438	0.6406	0.6391	0.6459	**0.7727**	0.7656	0.7596	0.7470	0.7648
16	**0.5559**	0.5505	0.5433	0.5414	0.5484	**0.8620**	0.8546	0.8389	0.8434	0.8477
17	**0.9547**	0.9451	0.9287	0.9314	0.9506	**0.9461**	0.9329	0.9298	0.9180	0.9368
18	**0.7779**	0.7701	0.7609	0.7576	0.7768	**0.7921**	0.7879	0.7749	0.7712	0.7782
19	**0.9386**	0.9252	0.9171	0.9195	0.9297	**0.9235**	0.9136	0.9055	0.9061	0.9125
20	**0.9156**	0.9101	0.8991	0.8867	0.9070	**0.9083**	0.8978	0.8884	0.8854	0.8938
平均	**0.8199**	0.8121	0.8012	0.7957	0.8139	**0.8411**	0.8318	0.8225	0.8173	0.8301

表 4.12 在 NAFSA 和 NFOA 上的 KNN 分类准确率

Table 4.12 KNN classification accuracy on NAFSA and NFOA

ID	NAFSA	BBSAR-NAFSA	AGAR-NAFSA	RS-NAFSA	KRS-NAFSA	NFOA	BBSAR-NFOA	AGAR-NFOA	RS-NFOA	KRS-NFOA
1	**0.7346**	0.7254	0.7103	0.7082	0.7225	**0.7758**	0.7658	0.7593	0.7492	0.7684
2	**0.7565**	0.7478	0.7400	0.7407	0.7464	**0.7287**	0.7228	0.7106	0.7042	0.7197
3	**0.6570**	0.6484	0.6404	0.6355	0.6479	**0.6504**	0.6429	0.6370	0.6341	0.6421
4	**0.9374**	0.9270	0.9185	0.9157	0.9294	**0.9664**	0.9606	0.9504	0.9444	0.9565
5	**0.8444**	0.8373	0.8172	0.8150	0.8349	**0.8973**	0.8908	0.8735	0.8745	0.8896
6	**0.8432**	0.8349	0.8268	0.8193	0.8289	**0.8626**	0.8551	0.8471	0.8380	0.8534
7	**0.9120**	0.9045	0.8831	0.8513	0.9025	**0.9203**	0.9071	0.9021	0.8770	0.9102
8	**0.7674**	0.7622	0.7451	0.7434	0.7583	**0.7646**	0.7549	0.7397	0.7388	0.7579
9	**0.6855**	0.6813	0.6634	0.6609	0.6764	**0.7036**	0.6940	0.6836	0.6799	0.6951
10	**0.8745**	0.8654	0.8595	0.8568	0.8594	**0.8311**	0.8243	0.8039	0.8081	0.8209
11	**0.9233**	0.9175	0.9039	0.9069	0.9114	**0.9234**	0.9100	0.9015	0.8963	0.9107
12	**0.9373**	0.9292	0.9053	0.9198	0.9253	**0.8943**	0.8835	0.8702	0.8775	0.8810
13	**0.8087**	0.7979	0.7801	0.7685	0.7952	**0.7874**	0.7805	0.7748	0.7520	0.7782
14	**0.8800**	0.8675	0.8664	0.8484	0.8666	**0.8377**	0.8270	0.8173	0.8113	0.8245

（续表 4.12）

ID	NAFSA	BBSAR-NAFSA	AGAR-NAFSA	RS-NAFSA	KRS-NAFSA	NFOA	BBSAR-NFOA	AGAR-NFOA	RS-NFOA	KRS-NFOA
15	**0.7631**	0.7561	0.7462	0.7402	0.7510	**0.7643**	0.7588	0.7365	0.7507	0.7575
16	**0.8561**	0.8456	0.8420	0.8410	0.8491	**0.8634**	0.8549	0.8443	0.8328	0.8548
17	**0.9377**	0.9313	0.9094	0.9057	0.9241	**0.9225**	0.9135	0.8893	0.8928	0.9093
18	**0.7936**	0.7840	0.7755	0.7685	0.7833	**0.7624**	0.7519	0.7380	0.7458	0.7502
19	**0.8859**	0.8769	0.8619	0.8544	0.8770	**0.8856**	0.8761	0.8598	0.8589	0.8752
20	**0.8854**	0.8781	0.8614	0.8630	0.8733	**0.9051**	0.8930	0.8746	0.8792	0.8969
平均	**0.8342**	0.8259	0.8128	0.8082	0.8231	**0.8323**	0.8234	0.8107	0.8073	0.8226

表 4.13 在 NMBO 和 NGA 上的 CART 分类准确率

Table 4.13 CART classification accuracy on NMBO and NGA

ID	NMBO	BBSAR-NMBO	AGAR-NMBO	RS-NMBO	KRS-NMBO	NGA	BBSAR-NGA	AGAR-NGA	RS-NGA	KRS-NGA
1	**0.7959**	0.7882	0.7812	0.7727	0.7863	**0.7830**	0.7777	0.7569	0.7599	0.7711
2	**0.7581**	0.7522	0.7430	0.7413	0.7501	**0.7586**	0.7542	0.7442	0.7448	0.7469
3	**0.7005**	0.6942	0.6758	0.6810	0.6929	**0.6444**	0.6364	0.6256	0.6250	0.6383
4	**0.9290**	0.9232	0.9128	0.9035	0.9129	**0.9212**	0.9087	0.9038	0.8979	0.9064
5	**0.9191**	0.9138	0.8912	0.8976	0.9029	**0.9070**	0.8948	0.8802	0.8870	0.8962
6	**0.8427**	0.8380	0.8280	0.8260	0.8292	**0.8277**	0.8222	0.8128	0.8078	0.8168
7	**0.8651**	0.8530	0.8433	0.8051	0.8525	**0.8965**	0.8863	0.8741	0.8446	0.8856
8	**0.7531**	0.7470	0.7344	0.7305	0.7443	**0.7611**	0.7504	0.7365	0.7356	0.7535
9	**0.6488**	0.6402	0.6281	0.6287	0.6374	**0.6987**	0.6916	0.6795	0.6832	0.6874
10	**0.8069**	0.8006	0.7844	0.7925	0.7973	**0.8001**	0.7887	0.7759	0.7732	0.7932
11	**0.8671**	0.8590	0.8432	0.8458	0.8524	**0.8769**	0.8713	0.8637	0.8571	0.8699
12	**0.8771**	0.8678	0.8463	0.8451	0.8688	**0.8634**	0.8565	0.8407	0.8374	0.8533
13	**0.7062**	0.7009	0.6858	0.6544	0.6989	**0.7806**	0.7706	0.7645	0.7287	0.7670
14	**0.8930**	0.8818	0.8766	0.8765	0.8847	**0.8943**	0.8869	0.8777	0.8710	0.8826
15	**0.6809**	0.6721	0.6564	0.6655	0.6696	**0.7334**	0.7258	0.7069	0.7110	0.7211
16	**0.8493**	0.8444	0.8226	0.8269	0.8353	**0.9836**	0.9741	0.9672	0.9550	0.9712
17	**0.9271**	0.9186	0.9066	0.9101	0.9159	**0.9193**	0.9126	0.9029	0.8871	0.9106
18	**0.7765**	0.7715	0.7494	0.7577	0.7692	**0.7639**	0.7542	0.7409	0.7478	0.7563
19	**0.8484**	0.8414	0.8297	0.8332	0.8398	**0.8345**	0.8259	0.8046	0.8181	0.8214
20	**0.8566**	0.8440	0.8325	0.8352	0.8464	**0.8427**	0.8372	0.8224	0.8214	0.8287

平均	**0.8151**	0.8076	0.7936	0.7915	0.8043	**0.8245**	0.8163	0.8041	0.7997	0.8139

表 4.14 在 NAFSA 和 NFOA 上的 CART 分类准确率

Table 4.14 CART classification accuracy on NAFSA and NFOA

ID	NAFSA	BBSAR-NAFSA	AGAR-NAFSA	RS-NAFSA	KRS-NAFSA	NFOA	BBSAR-NFOA	AGAR-NFOA	RS-NFOA	KRS-NFOA
1	**0.7661**	0.7572	0.7481	0.7508	0.7589	**0.7984**	0.7899	0.7710	0.7837	0.7869
2	**0.7653**	0.7603	0.7433	0.7435	0.7560	**0.7165**	0.7083	0.7014	0.6911	0.7107
3	**0.6994**	0.6931	0.6835	0.6829	0.6905	**0.6609**	0.6546	0.6510	0.6423	0.6553
4	**0.9229**	0.9124	0.8905	0.9062	0.9142	**0.9172**	0.9106	0.8857	0.8938	0.9034
5	**0.8470**	0.8413	0.8209	0.8211	0.8358	**0.8801**	0.8710	0.8585	0.8589	0.8651
6	**0.8310**	0.8215	0.8122	0.8077	0.8218	**0.8631**	0.8543	0.8377	0.8466	0.8563
7	**0.8892**	0.8780	0.8737	0.8404	0.8743	**0.8733**	0.8653	0.8449	0.8292	0.8635
8	**0.7440**	0.7365	0.7213	0.7236	0.7320	**0.7422**	0.7376	0.7235	0.7170	0.7357
9	**0.6747**	0.6709	0.6608	0.6536	0.6632	**0.6766**	0.6685	0.6528	0.6602	0.6673
10	**0.8224**	0.8117	0.7965	0.8030	0.8135	**0.7998**	0.7929	0.7860	0.7711	0.7884
11	**0.8758**	0.8711	0.8531	0.8562	0.8626	**0.8589**	0.8476	0.8319	0.8299	0.8520
12	**0.8968**	0.8902	0.8788	0.8724	0.8856	**0.8817**	0.8697	0.8619	0.8559	0.8745
13	**0.7786**	0.7736	0.7516	0.7228	0.7694	**0.7540**	0.7479	0.7355	0.7100	0.7473
14	**0.8938**	0.8834	0.8652	0.8758	0.8782	**0.8435**	0.8392	0.8215	0.8253	0.8357
15	**0.7258**	0.7176	0.7075	0.6995	0.7190	**0.7306**	0.7232	0.7076	0.7105	0.7244
16	**0.9818**	0.9683	0.9519	0.9542	0.9709	**0.9672**	0.9608	0.9521	0.9500	0.9581
17	**0.9062**	0.8940	0.8846	0.8875	0.8963	**0.8971**	0.8881	0.8731	0.8715	0.8884
18	**0.7550**	0.7452	0.7354	0.7277	0.7427	**0.7772**	0.7678	0.7650	0.7521	0.7640
19	**0.7630**	0.7543	0.7442	0.7412	0.7566	**0.7841**	0.7784	0.7653	0.7596	0.7761
20	**0.8517**	0.8468	0.8345	0.8345	0.8409	**0.8509**	0.8429	0.8373	0.8348	0.8382
平均	**0.8195**	0.8114	0.7979	0.7952	0.8091	**0.8137**	0.8059	0.7932	0.7897	0.8046

通过观察表 4.7-4.14 得：

(1) 与其他算法相比，不论采用任何分类器我们提出的算法不会带来更低的分类准确率和稳定性。例如，在“Urban Land Cover”(ID: 8)数据集上，使用我们的算法求解出的特征选择结果集计算的分类准确率的值与其他几个算法求解的特征选择结果集计算出的分类准确率的中间值相差不大。

(2) 与 RS 算法相比，采用重采样在均值表现相当，但是在一些数据集上我们提出的 KRS 算法会显著提高 RS 算法的分类准确率和稳定性。例如，在“Wine” (ID: 7) 数据集上，在加速 NMB 算法时，使用 KRS 的算法求解出的特征选择

结果集的 KNN 分类稳定性和准确率的值比使用 RS 算法时分别提高了 4.50% 和 5.53%; 使用 KRS 的算法求解出的特征选择结果集的 CART 分类稳定性和准确率的值比使用 RS 算法时分别提高了 5.81%和 5.89%. 由此可知，此时的重采样弥补了 RS 算法造成的类别不平衡问题。

4.4 本章小结

我们提出的随机采样加速器算法实际上从样本角度考虑提升特征选择的时间效率。通过在 20 个 UCI 数据的实验结果可以得出以下结论：(1) 我们的算法可以明显地降低进行特征选择的时间消耗；(2) 计算出的特征选择结果不会导致分类准确率的大幅度降低，与其他算法基本保持平衡。(3) k-means SMOTE 可以有效解决随机采样加速器面临的类别不平衡问题。

下一步的研究可以从以下两方面进行：

(1) 我们提出的算法仅仅被应用到固定的粒度，多粒度可以被进一步引入到我们提出的算法中来。

(2) 其他度量如条件熵、邻域决策错误率可以被用来验证我们算法的有效性。

第 5 章 基于采样加速和扰动集成的快速元启发式算法

5.1 问题描述

在第二章和第三章曾描述过特征选择的重要性以及对进行特征选择过程加速的必要性，在这一章节我们将不再重复地阐述应用背景。在前两节我们自己设计出的加速算法存在一个不足之处：对比原先的元启发式算法，其分类性能并未改善甚至在不同数据集上出现了不同程度的下降。鉴于此，本文采用数据扰动策略和集成选择器提升特征选择的分类性能。

近年来，为了进一步提升特征选择的性能，已有众多学者在分类问题中引入了集成策略，大致可以分为两类：

(1) 在特征选择过程中引入集成机制，帮助搜索过程更好地对候选特征进行评估，进而挑选出更为稳健特征以利于后续的学习；

(2) 重复利用某一搜索或者利用不同的搜索进程，找出多个不同的特征选择结果的子集，为后续的学习提供基础性集成工具。一般来说，后者相较于前者来说，框架的搭建更为灵活，且能够较大幅度地提升分类器的性能。集成选择器就属于此类集成方法之一。

但是，集成选择器本质上是一种投票策略，依靠所选特征子集的区分度来提高分类性能。而同一个数据集在进行特征选择时得到的结果是基本相同的，这削弱了集成选择器的效果。为了产生更多不同且有效的特征选择结果子集，本文采用了数据扰动策略。数据扰动是对原始数据结构的重构，能够让同一个数据集产生不同的邻域分布，这样可以在特征选择后可以选择后产生更多不同且有效的特征选择结果。

综上所述，本章节主要解决基于随机采样加速器和重采样的快速元启发算法的分类性能问题。

5.2 相关知识与工作

5.2.1 集成选择器

集成选择器技术[47] (Ensemble Selector)，作为集成学习领域的一种应用，旨在通过综合多种模型或算法来增强预测、分类及特征选择的性能。在机器学习和数据科学的实践中，集成方法因其提升模型鲁棒性、降低过拟合风险和增加预测准确性的能力而

备受推崇。特别地，集成选择器聚焦于特征选择过程的优化，通过在众多特征选择模型或算法中进行挑选或结合，旨在寻找最优的特征子集，从而提高最终模型的表现力。集成选择器的核心要素可以概括为以下几点：

(1) 多样性：集成选择器的效能核心在于其内在的多样性，即通过汇集来自不同来源的模型或算法所提供的知识以增强性能。在特征选择环境下，这意味着要融合多种特征选择方法（如过滤法、包装法和嵌入法）的优势，每一种方法均以其独特的机制评估特征的重要性。

(2) 投票机制：集成选择器通常通过一种投票或汇总机制来整合多种特征选择方法的输出，例如，采用多数投票法，即只有得到多数模型支持的特征才会被选入最终特征集。

(3) 权重分配：较为复杂的集成选择器会为不同的特征选择模型或算法分配权重，这些权重基于各方法在先前性能上的表现或对当前任务的适应性。通过这种权重分配，可以进一步细化特征选择的成果。

(4) 优化策略：集成选择器采纳诸如遗传算法、粒子群优化（PSO）等复杂的优化策略，在特征子集空间内搜索最优解，旨在保持模型简洁的同时最大化预测性能。

(5) 自适应性：先进的集成选择器可以根据模型在验证集上的表现自我调整其组成部分（如模型或算法的选择、权重分配等），以针对特定数据集提供最优化的特征选择策略。

综上所述，集成选择器通过整合多种特征选择技术，利用各种方法的互补优点以及通过集成策略实现的整体评估，向机器学习模型提供了一种高效、强大的特征选择工具。本研究采用的集成选择器主要基于多数投票机制，通过此机制来综合多次运用算法 4.2 得到的结果，即只有那些得到大多数算法结果支持的特征，才会被纳入最终的特征集中。

5.2.2 数据扰动策略

虽然集成选择器通过整合多种特征选择技术能够增强算法的性能，但其效能可能面临局限。这主要是因为在同一数据集上进行特征选择时，得到的特征选择结果往往具有相似性。为了克服这一限制，作者提出了数据扰动策略。在原始特征空间上，采用数据扰动策略[65-67]可以构造多个具有显著差异的扰动数据。它能够在数据集中产生更多的邻域关系，从而揭示样本各种特征的分布情况。例如，Li 等人[51]提出了一种数据扰动策略来分类具有广泛特征信息的小数据集。Wang 等人[52]提出了一种针对数值型数据的耦合特征分析的数据扰动策略。因此，在元启发式算法中适当引入集成和扰动策略，以提高分类性能，尤其是分类稳定性。虽然数据扰动策略可以挖掘更多关于

特征空间的耦合信息，但改变后的特征空间对分类的有效性并不总是得到保证。集成策略[44-47]已经被许多学者使用来解决这个问题。首先，本文利用数据扰动生成了 5 个特征空间。其次，从 5 个特征空间中得到 5 个特征选择结果。接着，利用五个特征选择结果可以产生更有差异的预测。最后，基于预测的投票可以用来诱导最终的分类结果。数据扰动的过程如图 5.1 所示。

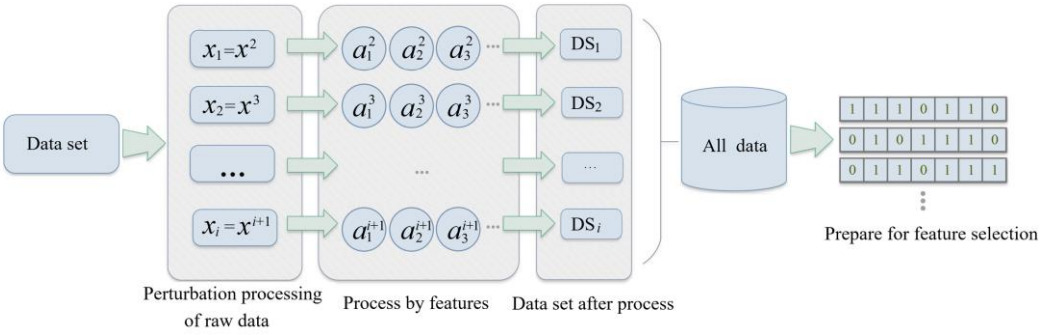

图 5.1 数据扰动过程。对于每个 $x \in U$, x_i (i= 1, 2, ...) 表示不同扰动处理后的数据

Fig.5.1 Data perturbation process. For each $x \in U$, x_i (i= 1, 2, ...) represents the data after different perturbation processing.

5.3 基于采样加速和扰动集成的快速元启发式算法

为了实现该方法，我们首先通过扰动策略将原始邻域决策系统转化为五个邻域决策系统。然后，使用算法 4.2 在 5 个扰动数据集上生成特征选择结果。接着，将多个分类器应用于这些特征选择结果，以生成五组预测标签。最后，使用多数投票法生成预测标签集合，并适当地获得分类结果。

在本节中，我们提出了一个新的框架“REP”，通过集成策略和数据扰动来提高算法 4.2 的性能。在这里，“R”、“E”和“P”分别表示随机采样加速器、集成策略和扰动策略。然后，构建了基于随机采样、加速器和数据扰动集成的快速元启发式特征选择算法。按照结合的元启发式算法种类，分别称为 REP-NMBO、REP-NGA、REP-NAFSA 和 REP-NFOA。

根据以上的关键步骤，具体的求解算法将在算法 5.1 中展示。

算法 5.1： 基于采样加速和扰动集成的快速元启发式算法。(REP)

输入： 决策系统 DS = <U, $AT \cup \{d\}$>, 约束条件 C_ρ^U;

输出： A.

步骤 1: 初始化，令 A=∅, SA=∅; // SA 表示一组特征选择集合

步骤 2: Set k=5; // k 是扰动数据处理的次数

步骤 3: **For** i =1 : k **Do**

数据集 U 通过扰动策略生成 U_i;

使用算法 5 对 U_i 进行特征选择，得到一个特征选择 A_i;

$SA=SA\cup A_i$;

End For

$A=vote(SA)$ //对 SA 结果集使用投票策略

步骤 4: 输出 A.

在算法 5.1 的过程中，其时间消耗主要由算法 4.2 在 k 个扰动数据集中产生。因此，它的主要时间消耗取决于算法 4.2。经过对算法 4.2 的分析，可以得出算法 5.1 在最坏时间复杂度为：$T(n)=k\cdot O\left(\sum_{i=1}^{m}n_i\right)$，其中 $n_i=(\frac{i}{m}\cdot|U|\cdot T_i)^2$ $(i=1,2,\ldots,m)$ 是算法 4.2 在第 i 次循环中的时间消耗。T 表示整个样本集下使用的元启发式算法所需的迭代数，$T_i(i=1,2,\ldots,m)$ 为每次分组使用的元启发式算法的迭代数，$T=T_1+T_2+\ldots+T_m$. 综上，当 的值较大时，$T(n)\le O\left(|U|^2\cdot N\cdot T\right)$ 成立。因此，该算法能够产生加速效果。并且由于基于投票策略的集成，在分类性能上优于原算法。

下面的图 5.2 给出了算法 5.1 的实现过程。

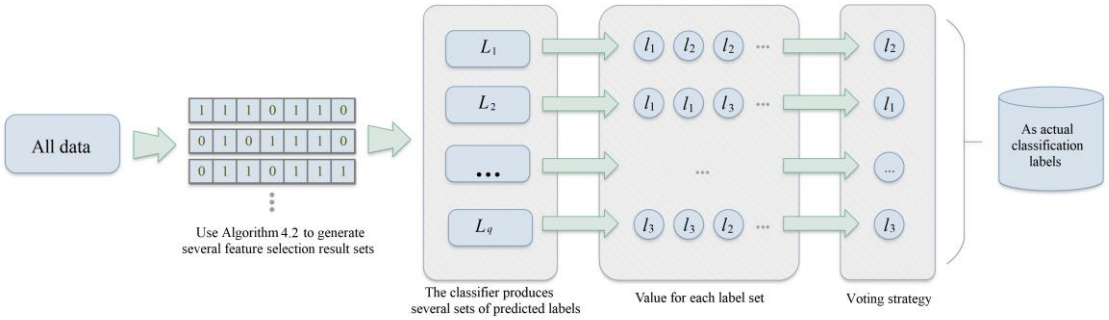

图 5.2 结合随机抽样加速器和使用数据扰动的集成的元启发式特征选择算法的过程

Fig.5.2 The process of a Meta-heuristic Feature Selection Algorithm Combining Random Sampling Accelerator and Ensemble Using Data Perturbation.

5.4 实验分析

5.4.1 实验数据

本节从 UCI 标准数据集中选取 20 个数据集进行实验，以证明本文算法的有效性。数据的具体信息见下表 5.1。实验中，所有的数据集的条件特征值都是经过归一化处理的。所有实验均在一台装有 Windows 10, CPU 为 Intel® Core(TM) i5-7300HQ (2.50 GHz)

和 8.00 GB 内存的个人计算机上进行。实验平台为 MATLAB R2018a。

表 5.1 数据集描述

Table 5.1 Description of data sets

ID	数据集名称	样本数	特征数	决策类数	来源
1	Cardiotocography	2126	21	10	UCI
2	SPECTF Heart	267	44	2	UCI
3	Statlog (Vehicle Silhouettes)	846	18	4	UCI
4	Wdbc	569	30	2	UCI
5	Dermatology	366	34	6	UCI
6	Forest Type Mapping	523	27	4	UCI
7	Wine	178	13	3	UCI
8	Urban Land Cover	675	147	9	UCI
9	German	1000	24	2	UCI
10	QSAR Biodegradation	1055	41	2	UCI
11	Synthetic Control Chart Time Series	600	60	6	UCI
12	Climate Model Simulation Crashes	540	20	2	UCI
13	Statlog (Heart)	270	13	2	UCI
14	Ionosphere	351	34	2	UCI
15	waveform	5000	21	3	UCI
16	Wall-Following Robot Navigation	5456	24	4	UCI
17	Pen-Based Recognition of Handwritten Digits	10992	17	10	UCI
18	MAGIC Gamma Telescope	19020	11	2	UCI
19	twonorm	7400	20	2	UCI
20	Crowdsourced Mapping	10845	29	6	UCI

5.4.2 实验设置

在本节的实验中，邻域粗糙集被使用来定义特征选择的形式。我们选取了 20 个半径，分别为 0.02, 0.04, …，0.40，被用来控制邻域的大小。同时，我们还用了 10 折交叉验证来进行特征选择，在测试集上对比不同特征选择结果所对应分类器的性能。实验使用 KNN (k=3)和 CART (默认参数)两种分类器对测试数据集进行分类。此外，将元启发式算法生成的初始种群中个体的初始长度设置为 $10\% \times |AT|$。

我们选取四种元启发式算法（算法 3.1-3.4）分别与参考文献[47]中的 ES 结合来对比我们的算法 5.1.

5.4.3 实验结果

在本章节中，我们将对比不同算法进行特征选择的时间消耗。计算 NMBO、NGA、NAFSA、NFOA、ES 和 REP 的消耗时间，表 5.2 和表 5.3 列出了相应的消耗时间。在给定的表中，第一列表示的是数据集的编号 (ID)，而其他列被分为两部分。针对每一部分：第 1 列表示基于粗糙集 (NMBO、NGA、NAFSA 和 NFOA 中的一种)的元启发式算法的时间消耗数据；第二列表示 ES 集成策略的时间消耗数据；第三列表示 REP 集成框架的时间消耗数据。

表 5.2 结合 NMBO 或 NGA 时进行特征选择的时间消耗

Table 5.2 The time consumption of feature selection when combined with NMBO or NGA

ID	NMBO	ES- NMBO	REP- NMBO	NGA	ES- NGA	REP- NGA
1	47.9497	253.9560	**8.7344**	136.1728	716.8000	**91.3944**
2	**0.1042**	0.5490	0.1267	1.2168	6.2261	**1.0988**
3	4.3831	22.1658	**1.4273**	40.6925	207.2552	**14.0752**
4	**0.5040**	2.5765	1.4269	**20.2095**	102.9289	34.1026
5	**0.1201**	0.6114	0.3683	**0.4513**	2.3071	7.1114
6	2.0721	10.9357	**0.6727**	57.0385	291.7061	**14.3552**
7	**0.0405**	0.2209	0.2170	**0.2770**	1.3978	2.8549
8	**0.2791**	1.4534	0.4778	1.6414	8.2940	**1.6034**
9	0.8446	4.2713	**0.8346**	5.3340	27.5003	**1.5813**
10	6.8049	34.1967	**3.9806**	211.8428	1089.1899	**55.6336**
11	**0.1398**	0.7521	0.9809	**1.0032**	5.1942	3.1477
12	**0.0793**	0.4083	0.1898	**0.7359**	3.6898	1.0461
13	**0.0581**	0.3070	0.2415	**1.3016**	6.6825	2.0194
14	**0.0398**	0.2088	0.2580	**0.3978**	2.0586	1.6946
15	36.8335	190.4366	**18.0042**	222.3042	1112.9437	**112.6397**
16	49.8467	255.9628	**26.9559**	1234.5499	6338.3027	**280.9319**
17	253.0885	1326.1837	**203.4644**	2335.7810	11959.1990	**545.3263**
18	1325.1548	6825.0366	**560.4408**	3244.6915	16288.3512	**1026.5966**

（续表 5.2）

ID	NMBO	ES- NMBO	REP- NMBO	NGA	ES- NGA	REP- NGA
19	294.4798	1480.2029	**47.2183**	746.3525	3798.6881	**190.8134**
20	109.6315	550.3501	**82.9915**	967.2155	4988.3308	**375.5970**
平均	106.6227	548.0393	**47.9506**	461.4605	2347.8523	**138.1812**

表 5.3 结合 NAFSA 或 NFOA 时进行特征选择的时间消耗

Table 5.3 The time consumption of feature selection when combined with NMBO or NGA

ID	NAFSA	ES- NAFSA	REP- NAFSA	NFOA	ES- NFOA	REP- NFOA
1	212.2761	1069.9139	**50.9534**	304.8315	1608.6263	**115.6708**
2	4.9257	25.5569	**1.0161**	2.1948	11.2884	**0.6712**
3	14.3639	74.9855	**5.0032**	51.6666	264.2128	**10.0489**
4	**12.2268**	64.3960	17.9894	**13.7486**	70.5950	16.7336
5	**0.2672**	1.3571	4.3021	**0.5253**	2.7144	2.2261
6	15.1733	76.8530	**6.7454**	28.4386	145.6626	**8.5818**
7	**0.1680**	0.8459	0.9532	**0.2177**	1.1212	1.1110
8	**0.6731**	3.4060	0.8166	3.6413	18.9342	**2.0178**
9	14.0919	71.0319	**9.346**	13.9799	71.9142	**5.3603**
10	580.2302	2923.1996	**60.4212**	**12.2091**	62.5409	31.4046
11	**0.2366**	1.2534	3.3139	**1.6109**	8.3149	7.1120
12	**0.1400**	0.7367	0.583	0.4668	2.3827	**0.4050**
13	**0.6021**	3.0944	1.0022	0.8506	4.4071	**0.8101**
14	1.1996	6.0332	**1.1564**	0.6321	3.2428	**0.5239**
15	1079.4309	5465.6982	**138.5166**	3471.1192	17709.6503	**312.1738**
16	2897.5100	14794.3963	**376.7201**	1188.5205	6077.3808	**273.4878**
17	1081.7891	5575.5410	**460.5149**	3121.2945	15824.3389	**888.2521**
18	3398.1025	17373.1390	**810.9944**	4100.7325	25457.4364	**1678.7948**
19	772.4650	3949.1503	**178.2361**	1101.5132	5706.5316	**339.5264**
20	942.1020	4743.0127	**318.845**	2056.6406	10587.2031	**695.7599**
平均	551.3987	2811.1801	**122.3715**	773.7417	4181.9249	**219.5336**

通过观察表 5.2 和表 5.3，不难得出以下结论：

(1) 对于算法 5.1，使用随机采样加速器来减少特征选择的时间消耗。算法 5.1 比 ES 耗时少得多。以数据“Wdbc”(ID: 4)为例，使用 ES 解决特征选择问题时，耗时分别为 2.5765 秒 (ES-NMBO)、102.9289 秒 (ES-NGA)、64.3960 秒 (ES-NAFSA)

和 70.5950 秒 (ES-NFOA)，而算法 5.1 所需时间分别为 1.4269 秒(REP-NMBO)、34.1026 秒 (REP-NGA)、17.9894 秒 (REP-NAFSA)和 16.7336 秒 (REP-NFOA)。在集成情况下，算法 5.1 和 ES 使用的数据集的数量是相同的。因此可以看出，随机采样加速器的引入确实可以加快特征选择的过程。

(2) 对于算法 5.1，当样本数大于 2000 时，也可以发现算法 5.1 比 NMBO、NGA、NAFSA 和 NFOA。以数据“Cardiotocography” (ID:1)为例，样本数为 2126，NMBO、NGA、NAFSA 和 NFOA 所消耗的时间分别为 47.9497 秒、136.1728 秒、212.2761 秒和 304.8315 秒。算法 5.1 消耗的时间仅为 8.7344 秒 (REP-NMBO)、91.3944 秒 (REP-NGA)、50.9534 秒 (REP-NAFSA)和 115.6708 秒 (REP-NFOA)。

两个分类器，即 KNN (k = 3)和 CART (使用默认参数)，被用来证明分类性能。使用 KNN 分类器的主要原因是其简单，不需要参数估计，特别适用于多分类问题。使用 CART 分类器的原因在于其分类规则易于理解且准确率较高。表 5.4-5.7 给出了不同算法在测试集中得到的 KNN 和 CART 分类稳定性。表 5.8-5.11 分别给出了测试集中不同算法得到的 KNN 和 CART 分类准确率。

表 5.4 结合 NMBO 或 NGA 时的 KNN 分类稳定性

Table 5.4 KNN classification stability when combined with NMBO or NGA

ID	NMBO	ES- NMBO	REP- NMBO	NGA	ES- NGA	REP- NGA
1	0.8443	0.8755	**0.9186**	0.8498	0.8597	**0.8879**
2	0.7698	0.8331	**0.8638**	0.7303	0.7673	**0.8157**
3	0.8273	0.8409	**0.8882**	0.8073	0.8233	**0.8632**
4	0.9703	0.9853	**0.9901**	0.9656	0.9679	**0.9862**
5	0.9063	0.9455	**0.9865**	0.8876	0.9108	**0.9754**
6	0.9235	0.9366	**0.9635**	0.9327	0.9391	**0.9656**
7	0.9229	0.9344	**0.9726**	0.9330	0.9448	**0.9656**
8	0.8576	0.8600	**0.9072**	0.8593	0.8788	**0.9204**
9	0.7382	0.7983	**0.8395**	0.7294	0.7659	**0.8292**
10	0.9038	0.9192	**0.9426**	0.9219	0.9258	**0.9500**
11	0.9060	0.9140	**0.9630**	0.9104	0.9121	**0.9837**
12	0.9407	0.9523	**0.9948**	0.9561	0.9706	**0.9910**
13	0.7563	0.7609	**0.8481**	0.8510	0.8762	**0.9184**
14	0.8861	0.8977	**0.9265**	0.9376	0.9531	**0.9822**
15	0.5870	0.6301	**0.7070**	0.7602	0.8395	**0.8797**
16	0.5542	0.6202	**0.6810**	0.8839	0.8840	**0.9350**
17	0.9227	0.9562	**0.9946**	0.9378	0.9531	**0.9948**

（续表 5.4）

ID	NMBO	ES- NMBO	REP- NMBO	NGA	ES- NGA	REP- NGA
18	0.8721	0.9009	**0.9316**	0.8869	0.8987	**0.9552**
19	0.9067	0.9353	**0.9885**	0.8935	0.9111	**0.9766**
20	0.9394	0.9581	**0.9884**	0.9447	0.9623	**0.9916**
平均	0.8468	0.8727	**0.9148**	0.8789	0.8972	**0.9384**

表 5.5 结合 NAFSA 或 NFOA 时的 KNN 分类稳定性

Table 5.5 KNN classification stability when combined with NAFSA or NFOA

ID	NAFSA	ES- NAFSA	REP- NAFSA	NFOA	ES- NFOA	REP- NFOA
1	0.8295	0.8430	**0.8867**	0.8591	0.8705	**0.9142**
2	0.7616	0.7950	**0.8407**	0.7866	0.8187	**0.8556**
3	0.8040	0.8180	**0.8539**	0.8082	0.8206	**0.8566**
4	0.9622	0.9724	**0.9904**	0.9644	0.9757	**0.9913**
5	0.8045	0.8609	**0.9284**	0.8630	0.8979	**0.9435**
6	0.9230	0.9441	**0.9646**	0.9108	0.9305	**0.9708**
7	0.8860	0.9142	**0.9709**	0.9109	0.9319	**0.9783**
8	0.8513	0.8703	**0.9163**	0.8267	0.8457	**0.9047**
9	0.7187	0.7706	**0.8259**	0.7330	0.7901	**0.8399**
10	0.9033	0.9138	**0.9375**	0.9023	0.9134	**0.9338**
11	0.8951	0.9246	**0.9688**	0.8909	0.9289	**0.9767**
12	0.9223	0.9421	**0.9974**	0.9397	0.9529	**0.9977**
13	0.8345	0.8642	**0.9231**	0.8374	0.8443	**0.9187**
14	0.9074	0.9374	**0.9679**	0.8750	0.8934	**0.9529**
15	0.7414	0.8034	**0.8689**	0.7811	0.8002	**0.8810**
16	0.8983	0.9131	**0.9407**	0.8610	0.8723	**0.9243**
17	0.9351	0.9449	**0.9881**	0.9372	0.9479	**0.9886**
18	0.8918	0.9119	**0.9642**	0.9309	0.9578	**0.9908**
19	0.8736	0.9335	**0.9904**	0.8956	0.9273	**0.9811**
20	0.9331	0.9544	**0.9925**	0.9407	0.9685	**0.9937**
平均	0.8638	0.8916	**0.9359**	0.8727	0.8944	**0.9397**

表 5.6 结合 NMBO 或 NGA 时的 CART 分类稳定性

Table 5.6 CART classification stability when combined with NMBO or NGA

ID	NMBO	ES- NMBO	REP- NMBO	NGA	ES- NGA	REP- NGA
1	0.8112	0.8363	**0.8729**	0.8060	0.8260	**0.8761**

（续表 5.6）

ID	NMBO	ES- NMBO	REP- NMBO	NGA	ES- NGA	REP- NGA
2	0.7190	0.7639	**0.8138**	0.7159	0.7750	**0.8193**
3	0.7125	0.7385	**0.7534**	0.6899	0.7037	**0.7516**
4	0.9260	0.9346	**0.9476**	0.8997	0.9023	**0.9220**
5	0.8944	0.9231	**0.9769**	0.8696	0.9145	**0.9517**
6	0.8617	0.8619	**0.8843**	0.8655	0.8721	**0.8961**
7	0.8301	0.8683	**0.9110**	0.8272	0.8542	**0.9126**
8	0.7756	0.8030	**0.8507**	0.7965	0.8031	**0.8503**
9	0.6980	0.7169	**0.7676**	0.6860	0.7286	**0.7691**
10	0.8051	0.8364	**0.8687**	0.8068	0.8252	**0.8649**
11	0.8060	0.8567	**0.9025**	0.8172	0.8739	**0.9149**
12	0.8474	0.9041	**0.9452**	0.8858	0.9268	**0.9902**
13	0.7080	0.7545	**0.7844**	0.7541	0.7779	**0.8199**
14	0.8803	0.9054	**0.9395**	0.8706	0.9120	**0.9494**
15	0.6102	0.6832	**0.7346**	0.6883	0.7326	**0.7881**
16	0.7903	0.8079	**0.9249**	0.9798	0.9868	**1.0092**
17	0.9174	0.9184	**0.9521**	0.9001	0.9226	**0.9528**
18	0.8123	0.8299	**0.8636**	0.8310	0.8500	**0.8923**
19	0.7785	0.8159	**0.8561**	0.7584	0.7879	**0.8178**
20	0.8446	0.8847	**0.9292**	0.8455	0.8688	**0.8987**
平均	0.8014	0.8322	**0.8739**	0.8147	0.8422	**0.8824**

表 5.7 结合 NAFSA 或 NFOA 时的 CART 分类稳定性

Table 5.7 CART classification stability when combined with NAFSA or NFOA

ID	NAFSA	ES- NAFSA	REP- NAFSA	NFOA	ES- NFOA	REP- NFOA
1	0.7959	0.8275	**0.8647**	0.7909	0.8221	**0.8717**
2	0.7565	0.8265	**0.8665**	0.7365	0.7717	**0.8727**
3	0.7310	0.7745	**0.8089**	0.6862	0.7020	**0.8146**
4	0.8979	0.9208	**0.9268**	0.8850	0.9144	**0.9391**
5	0.8127	0.8519	**0.9041**	0.8365	0.8580	**0.9157**
6	0.8839	0.9167	**0.9261**	0.8694	0.8917	**0.9344**
7	0.8783	0.9084	**0.9487**	0.8421	0.8763	**0.9596**
8	0.7653	0.8212	**0.8608**	0.7463	0.7927	**0.8673**
9	0.6919	0.7315	**0.7728**	0.6868	0.7509	**0.7791**

（续表 5.7）

ID	NAFSA	ES- NAFSA	REP- NAFSA	NFOA	ES- NFOA	REP- NFOA
10	0.8132	0.8642	**0.8823**	0.8024	0.8430	**0.8905**
11	0.8060	0.8832	**0.9178**	0.8069	0.8690	**0.9285**
12	0.8371	0.9048	**0.9571**	0.8749	0.8931	**0.9691**
13	0.7819	0.8291	**0.8555**	0.7560	0.7825	**0.8624**
14	0.8733	0.8914	**0.9333**	0.8105	0.8695	**0.9399**
15	0.6691	0.7001	**0.7527**	0.6974	0.7164	**0.7637**
16	0.9748	0.9800	**0.9845**	0.9396	0.9646	**0.9937**
17	0.9017	0.9164	**0.9596**	0.9131	0.9378	**0.9653**
18	0.8298	0.8556	**0.8985**	0.7966	0.8482	**0.9058**
19	0.7492	0.8023	**0.8458**	0.7505	0.8206	**0.8565**
20	0.8389	0.8770	**0.8965**	0.8246	0.8834	**0.9042**
平均	0.8144	0.8542	**0.8881**	0.8026	0.8404	**0.8967**

表 5.8 结合 NMBO 或 NGA 时的 KNN 分类准确率

Table 5.8 KNN classification accuracy when combined with NMBO or NGA

ID	NMBO	ES- NMBO	REP- NMBO	NGA	ES- NGA	REP- NGA
1	0.7401	0.7434	**0.7580**	0.7426	0.7433	**0.7573**
2	0.7719	0.7790	**0.8114**	0.7554	0.7615	**0.7953**
3	0.6562	0.6634	**0.6816**	0.6343	0.6447	**0.6580**
4	0.9696	0.9726	**0.9829**	0.9550	0.9571	**0.9748**
5	0.9311	0.9400	**0.9750**	0.9300	0.9395	**0.9720**
6	0.8704	0.8772	**0.8942**	0.8664	0.8686	**0.8941**
7	0.9433	0.9456	**0.9694**	0.9333	0.9404	**0.9584**
8	0.8012	0.8042	**0.8265**	0.7427	0.7428	**0.7664**
9	0.6658	0.6667	**0.6941**	0.6924	0.7024	**0.7172**
10	0.8411	0.8471	**0.8596**	0.8530	0.8547	**0.8658**
11	0.9458	0.9576	**0.9724**	0.9453	0.9609	**0.9730**
12	0.9065	0.9166	**0.9374**	0.8819	0.8924	**0.9088**
13	0.7248	0.7313	**0.7554**	0.8137	0.8142	**0.8281**
14	0.8356	0.8403	**0.8552**	0.8709	0.8726	**0.8817**
15	0.6526	0.6566	**0.7490**	0.7727	0.7858	**0.8284**
16	0.5559	0.5579	**0.6714**	0.8620	0.8726	**0.8740**
17	0.9547	0.9579	**0.9954**	0.9461	0.9473	**0.9910**

（续表 5.8）

ID	NMBO	ES- NMBO	REP- NMBO	NGA	ES- NGA	REP- NGA
18	0.7779	0.7809	**0.8231**	0.7921	0.8052	**0.8578**
19	0.9386	0.9506	**0.9748**	0.9235	0.9281	**0.9693**
20	0.9156	0.9204	**0.9655**	0.9083	0.9162	**0.9443**
平均	0.8199	0.8255	**0.8576**	0.8411	0.8475	**0.8708**

表 5.9 结合 NAFSA 或 NFOA 时的 KNN 分类准确率

Table 5.9 KNN classification accuracy when combined with NMBO or NGA

ID	NAFSA	ES- NAFSA	REP- NAFSA	NFOA	ES- NFOA	REP- NFOA
1	0.7346	0.7441	**0.7578**	0.7758	0.7838	**0.7943**
2	0.7565	0.7644	**0.7922**	0.7287	0.7361	**0.7547**
3	0.6570	0.6644	**0.6773**	0.6504	0.6604	**0.6739**
4	0.9374	0.9464	**0.9605**	0.9664	0.9790	**0.9801**
5	0.8444	0.8490	**0.9457**	0.8973	0.9125	**0.9467**
6	0.8432	0.8483	**0.8726**	0.8626	0.8732	**0.8879**
7	0.9120	0.9268	**0.9516**	0.9203	0.9354	**0.9605**
8	0.7674	0.7740	**0.7942**	0.7646	0.7728	**0.7967**
9	0.6855	0.6923	**0.7201**	0.7036	0.7137	**0.7301**
10	0.8745	0.8890	**0.9002**	0.8311	0.8410	**0.8501**
11	0.9233	0.9311	**0.9669**	0.9234	0.9354	**0.9758**
12	0.9373	0.9522	**0.9703**	0.8943	0.9035	**0.9131**
13	0.8087	0.8216	**0.8359**	0.7874	0.8098	**0.8270**
14	0.8800	0.8890	**0.9201**	0.8377	0.8493	**0.8728**
15	0.7631	0.7716	**0.8281**	0.7643	0.7733	**0.8185**
16	0.8561	0.8693	**0.8728**	0.8634	0.8770	**0.8848**
17	0.9377	0.9518	**0.9855**	0.9225	0.9375	**0.9914**
18	0.7936	0.8065	**0.8437**	0.7624	0.8016	**0.8461**
19	0.8859	0.8921	**0.9401**	0.8856	0.8996	**0.9489**
20	0.8854	0.8903	**0.9356**	0.9051	0.9154	**0.9657**
平均	0.8342	0.8437	**0.8736**	0.8323	0.8455	**0.8710**

表 5.10 结合 NMBO 或 NGA 时的 CART 分类准确率

Table 5.10 CART classification accuracy when combined with NMBO or NGA

ID	NMBO	ES- NMBO	REP- NMBO	NGA	ES- NGA	REP- NGA
1	0.7959	0.8049	**0.8400**	0.7830	0.7999	**0.8251**

（续表 5.10）

ID	NMBO	ES- NMBO	REP- NMBO	NGA	ES- NGA	REP- NGA
2	0.7581	0.7659	**0.8001**	0.7586	0.7754	**0.8140**
3	0.7005	0.7092	**0.7337**	0.6444	0.6599	**0.6803**
4	0.9290	0.9470	**0.9511**	0.9212	0.9316	**0.9429**
5	0.9191	0.9295	**0.9871**	0.9070	0.9173	**0.9625**
6	0.8427	0.8595	**0.8683**	0.8277	0.8382	**0.8661**
7	0.8651	0.8741	**0.8975**	0.8965	0.9063	**0.9662**
8	0.7531	0.7712	**0.8114**	0.7611	0.7690	**0.8057**
9	0.6488	0.6626	**0.6786**	0.6987	0.7074	**0.7351**
10	0.8069	0.8229	**0.8474**	0.8001	0.8158	**0.8402**
11	0.8671	0.8878	**0.9346**	0.8769	0.8969	**0.9500**
12	0.8771	0.8923	**0.9391**	0.8634	0.8773	**0.9020**
13	0.7062	0.7215	**0.7460**	0.7806	0.7937	**0.8189**
14	0.8930	0.9066	**0.9294**	0.8943	0.9162	**0.9444**
15	0.6809	0.6954	**0.7741**	0.7334	0.7494	**0.7995**
16	0.8493	0.8674	**0.9372**	0.9836	0.9877	**0.9959**
17	0.9271	0.9486	**0.9611**	0.9193	0.9404	**0.9678**
18	0.7765	0.7906	**0.8344**	0.7639	0.7736	**0.8300**
19	0.8484	0.8625	**0.9069**	0.8345	0.8447	**0.9009**
20	0.8566	0.8670	**0.9046**	0.8427	0.8552	**0.8970**
平均	0.8151	0.8293	**0.8641**	0.8245	0.8378	**0.8722**

表 5.11 结合 NAFSA 或 NFOA 时的 CART 分类准确率

Table 5.11 CART classification accuracy when combined with NAFSA or NFOA

ID	NAFSA	ES- NAFSA	REP- NAFSA	NFOA	ES- NFOA	REP- NFOA
1	0.7661	0.7834	**0.8056**	0.7984	0.8094	**0.8168**
2	0.7653	0.7769	**0.8137**	0.7165	0.7239	**0.8183**
3	0.6994	0.7098	**0.7395**	0.6609	0.6718	**0.7493**
4	0.9229	0.9381	**0.9390**	0.9172	0.9387	**0.9519**
5	0.8470	0.8626	**0.9100**	0.8801	0.8933	**0.9216**
6	0.8310	0.8451	**0.8508**	0.8631	0.8751	**0.8622**
7	0.8892	0.9031	**0.9283**	0.8733	0.8903	**0.9357**
8	0.7440	0.7520	**0.8019**	0.7422	0.7577	**0.8138**
9	0.6747	0.6905	**0.7089**	0.6766	0.6903	**0.7147**

（续表 5.11）

ID	NMBO	ES- NMBO	REP- NMBO	NGA	ES- NGA	REP- NGA
10	0.8224	0.8340	**0.8581**	0.7998	0.8148	**0.8660**
11	0.8758	0.8853	**0.9468**	0.8589	0.8781	**0.9557**
12	0.8968	0.9069	**0.9538**	0.8817	0.8997	**0.9599**
13	0.7786	0.7902	**0.8045**	0.7540	0.7672	**0.8153**
14	0.8938	0.9104	**0.9297**	0.8435	0.8540	**0.9349**
15	0.7258	0.7395	**0.7817**	0.7306	0.7433	**0.7893**
16	0.9818	0.9860	**0.9906**	0.9672	0.9779	**1.0018**
17	0.9062	0.9251	**0.9371**	0.8971	0.9174	**0.9433**
18	0.7550	0.7720	**0.8241**	0.7772	0.7938	**0.8308**
19	0.7630	0.7790	**0.8170**	0.7841	0.7959	**0.8241**
20	0.8517	0.8703	**0.8971**	0.8509	0.8687	**0.9092**
平均	0.8195	0.8330	**0.8619**	0.8137	0.8281	**0.8707**

由表 5.4-5.11 可知，使用 KNN 和 CART 分类器时，算法 5.1 的性能优于 NMBO、NGA、NAFSA、NFOA 和 ES。特别地，算法 5.1 的使用显著提高了分类的稳定性。这表明算法 5.1 能够生成更令人满意的特征子集。然后，结合表 5.2 和表 5.3 的时间消耗对比，可知：

(1) 算法 5.1 与 NMBO、NGA、NAFSA 和 NFOA 相比，也能提高分类性能。考虑数据“waveform” (ID：15)。与 NMBO、NGA、NAFSA 和 NFOA 相比，算法 5.1 在 KNN 分类稳定性上分别提高了 20.44%、15.72%、17.20%和 12.79%。CART 的分类稳定性分别提高了 20.39%、14.50%、12.49%和 9.51%。KNN 的分类准确率分别提高了 14.77%、7.21%、8.52%和 7.09%。CART 分类精度分别提高了 13.69%、9.01%、7.70%和 8.03%。

(2) 算法 5.1 与 ES 算法相比，也能提高分类性能。考虑数据“MAGIC Gamma Telescope”(ID：18). 与 ES 相比，算法 5.1 在 KNN 分类稳定性上分别提高了 3.41%、6.29%、5.74%和 3.45%，CART 的分类稳定性分别提高了 4.06%、4.98%、5.01%和 6.79%。KNN 的分类准确率分别提高了 5.40%、6.53%、4.61%和 5.55%的精度，CART 分类准确率分别提高了 5.54%、7.29%、6.75%、4.66%。

(3) 算法 5.1 具有可接受的时间消耗。考虑数据集(ID : 1,3,6,9,10,15,16,17,18,19,20)，算法 5.1 比 NMBO、NGA、NAFSA、NFOA 和 ES 的耗时更少，其分类性能也高于这些算法。

总体而言，算法 5.1 的性能优于 NMBO、NGA、NAFSA、NFOA 和 ES。

5.5 本章小结

本章专注于基于采样加速和扰动集成的元启发式算法，旨在解决特征选择的重要性和求解过程的加速问题。针对原始元启发式算法在分类性能上的不足，提出了集成选择器以增强性能。该选择器基于两种策略：特征选择过程的集成和不同搜索过程的多样化结果集。然而，由于特征选择结果集的相似性，集成选择器的效能受限。为此，引入了数据扰动策略，通过重构数据结构产生不同邻域分布，以获得更多样化的特征选择结果集。

本文提出的“REP”框架融合了随机采样加速器、集成策略和数据扰动策略，通过协同作用提升算法处理高维数据分类问题的性能。该框架首先通过数据扰动创建多个邻域决策系统，并在扰动数据集上应用特征选择算法，利用多个分类器生成预测标签，最终通过多数投票法得出分类结果。整个章节展示了一种创新的方法，通过集成学习和数据扰动技术来提高元启发式特征选择算法的性能，具有重要的理论价值和应用前景。下一步的研究可以从以下几个方面展开：

(1) 算法 5.1 是一个算法框架，不仅可以应用于本文提到的 4 种元启发式算法，还可以改进其他算法的实验结果。

(2) 数据扰动策略对集成分类的影响值得深入探讨。

(3) 一些其他的度量，如：条件熵、邻域鉴别指数、邻域决策错误率也可以被用来验证我们所提的选择器的有效性。

总结与展望

本研究的主要贡献可以详细归纳为以下四个方面：

(1) 利用元启发式算法与邻域粗糙集理论配置新的特征选择策略：本研究将四种不同的元启发式算法（如帝王蝶优化算法、遗传算法、人工鱼群算法、随机森林优化算法）与邻域粗糙集理论结合，提出了一种新的特征选择策略。这个策略充分利用了邻域粗糙集理论在处理连续和离散特征方面的优势，同时发挥元启发式算法在全局搜索中的高效性，有效提高了特征选择的准确性和效率。该策略能够更有效地移除不相关和冗余的特征，保留对决策过程最有贡献的特征，从而提升了数据分类和预测模型的性能。

(2) 利用随机采样加速器加速特征选择过程：针对元启发式算法在处理大规模数据集时时间消耗过大的问题，本研究引入了随机采样加速器。该加速器通过随机分组样本的方式，显著减少了搜索样本空间的规模。特别地，加速器采用“一组一组”进行特征选择的策略，即只在一个样本组上进行特征选择，然后将选择结果标记并加入下一个组中，这样不仅加快了特征选择过程，还确保了特征选择结果的有效性和准确性。利用 k-means SMOTE 弥补随机采样加速器带来的类别不平衡问题：它通过结合聚类和过采样技术，智能地在数据空间的关键区域生成少数类样本，不仅提高了数据的类别平衡，也增强了模型对少数类的识别能力。

(3) 利用数据扰动策略和集成策略提高分类性能：为了解决元启发式算法在特征选择过程中可能产生的特征选择结果不稳定的问题，本研究采用了数据扰动策略和集成方法。数据扰动策略通过重构原始数据结构，打破数据之间的耦合，从而产生更多不同的特征选择结果。同时，通过集成这些不同的特征选择结果，使用投票机制来产生最终的分类结果，显著提高了分类的稳定性和准确性。这种集成和扰动策略的应用，有效地提升了元启发式算法在特征选择方面的性能，特别是在分类稳定性方面的表现。

综上所述，本研究的主要贡献在于提出了一种结合邻域粗糙集理论和元启发式算法的新型特征选择策略，通过引入随机采样加速器和数据扰动策略，不仅加快了特征选择过程，还提高了分类模型的性能。这些贡献为处理大规模数据集的特征选择提供了新的思路和方法，具有重要的理论价值和实践意义。但是仍然存在着一些局限性，尚有可以改进的空间。根据上文的讨论，未来的研究工作可以从以下几个方面进行着手：

(1) 扩展应用领域：将元启发式特征选择方法应用于更广泛的领域，例如生物信息学、金融市场分析、社交网络数据分析等，以解决特定领域的特殊问题。

(2) 提高解释性和可视化：开发工具和方法，提高特征选择过程和结果的解释性和可视化，使研究人员和从业者能够更好地理解和解释模型的决策过程。

(3) 探究多粒度特征下元启发式算法特征选择中的加速问题：本文所探讨的加速算法目前仅限于单粒度特征选择的场景，尚未深入探究多粒度特征选择所特有的特点和结构。在人类的思维过程中，“粒度”反映了我们将复杂事件进行分层、分块处理的精细程度，而“多粒度”则体现了从多个不同层面和角度对问题进行求解的多样性。考虑到单粒度特征选择在高维数据上已可能消耗大量时间，多粒度特征选择在高维数据上的时间成本更是难以估量。因此，我们迫切需要探索更为高效的加速策略，以增强多粒度特征选择算法在高维数据上的适用性，从而使其在实际应用中具有更广泛的实用价值。

参考文献

[1] Editors. Launch of an online data journal[J]. Nature, 2013, 502: 142.

[2] 刘宏达, 王荣. 论新时代中国大数据战略的内涵、特点与价值——学习习近平总书记关于大数据的重要论述[J]. 社会主义研究, 2019(5): 9-14.

[3] 梁吉业, 钱宇华, 李德玉, 等. 大数据挖掘的粒计算理论与方法[J]. 中国科学: 信息科学, 2015, 45(11): 1355-1369.

[4] W. P. Ding, J. Nayak, B. Naik, et al. Fuzzy and real-coded chemical reaction optimization for intrusion detection in industrial big data environment[J]. IEEE Transactions on Industrial Informatics, 2021, 17 (6): 4298-4307.

[5] L. J. Dong, R. H. Wang, D. G. Chen. Incremental feature selection with fuzzy rough sets for dynamic data sets[J]. Fuzzy Sets and Systems, 2023, 467: 108503.

[6] X. Zhang, C. L. Mei, J. H. Li, et al. Instance and feature selection using fuzzy rough sets: a bi-selection approach for data reduction[J]. IEEE Transactions on Fuzzy Systems, 2023, 31(6): 1981-1994.

[7] J. K. Chen, Y. J. Lin, J. S. Mi, et al. A spectral feature selection approach with kernelized fuzzy rough sets[J]. IEEE Transactions on Fuzzy Systems, 2022, 30(8): 2886-2901.

[8] K. Y. Liu, T. R. Li, X. B. Yang, et al. Neighborhood rough set based ensemble feature selection with cross-class sample granulation[J]. Applied Soft Computing, 2022, 131: 109747.

[9] A. Ismail, M. Sandell. A low-complexity endurance modulation for flash memory[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2022, 69(2): 424-428.

[10] Y. J. Tang, X. M. Zhang. Low-complexity resource-shareable parallel generalized integrated interleaved encoder[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(2): 694-706.

[11] Z. J. Li, K. Kamnitsas, B. Glocke. Analyzing overfitting under class imbalance in neural networks for image segmentation[J]. IEEE Transactions on Medical Imaging, 2021, 40(3): 1065-1077.

[12] Y. B. Park, J. C. Ho. Tackling overfitting in boosting for noisy healthcare data[J]. IEEE Transactions on Knowledge and Data Engineering, 2021, 33(7): 2995-3006.

[13] M. Baisantry, A. K. Sao, D. P. Shukla. Discriminative spectral spatial feature extraction-based band selection for hyper spectral image classification[J]. IEEE Transactions on Geoscience and Remote Sensing, 2022, 60: 1-14.

[14] W. P. Ding, I. Triguero, C. T. Lin. Coevolutionary fuzzy at tribute order reduction with complete attribute-value space tree[J]. IEEE Transactions on Emerging Topics in Computing, 2021, 5(1): 29-41.

[15] N. Momeni, A. A. Valdes, J. Rodrigues, et al. CAFS: Cost-aware features selection method for multimodal stress monitoring on wearable devices[J]. IEEE Transactions on Biomedical Engineering, 2022, 69(3): 1072-1084.

[16] W. W. Yan, J. Ba, T. H. Xu, et al. Beam-influenced attribute selector for producing stable reduct[J]. Mathematics, 2022, 10(4): 553.

[17] W. Wei, X. Y. Wu, J. Y. Liang, et al. Discernibility matrix based incremental attribute reduction for dynamic data[J]. Knowledge-Based Systems, 2018, 140: 142-157.

[18] W. Wei, J. B. Cui, J. Y. Liang, et al. Fuzzy rough approximations for set-valued data[J]. Information Sciences, 2016, 360: 181-201.

[19] O. Etesami, W. Haemers. On NP-hard graph properties characterized by the spectrum[J]. Discrete Applied Mathematics, 2020, 285: 526-529.

[20] A. Zhang, Y. Chen, L. Chen, et al. On the NP-hardness of scheduling with time restrictions[J]. Discrete Optimization, 2018, 28: 54-62.

[21] R. Guha, K. K. Ghosh, S. K. Bera, et al. Discrete equilibrium optimizer combined with simulated annealing for feature selection[J]. Journal of Computational Science, 2023, 67: 1877-7503.

[22] M. A. Elaziz, S. Ouadfel, A. A. A. El-Latif, et al. Feature selection based on modified bio-inspired atomic orbital search using arithmetic optimization and opposite-based learning[J]. Cognitive Computation, 2022, 14(6): 2274-2295

[23] R. K. V. Penmatsa, A. Kalidindi, S. K. R. Mallidi. Feature reduction and optimization of malware detection system using ant colony optimization and rough sets[J]. International Journal of Information Security and Privacy, 2020, 14(3): 95-114.

[24] X. Y. Luan, Z. P. Li, T. Z. Liu. A novel attribute reduction algorithm based on rough set and improved artificial fish swarm algorithm[J]. Neurocomputing, 2016, 174: 522-529.

[25] G. G. Wang, S. Deb, Z. H. Cui. Monarch butterfly optimization[J]. Neural Computing and Applications, 2019, 31(7): 1995-2014.

[26] S. S. Shreem, H. Turabieh, S. A. Azwari, et al. Enhanced binary genetic algorithm as a feature selection to predict student performance[J]. Soft Computing, 2022, 26(4): 1811-1823.

[27] M. Ghaemi, M. R. Feizi-Derakhshi. Feature selection using forest optimization algorithm[J]. Pattern Recognition, 2016, 60: 121-129.

[28] A. Campagner, D. Ciucci, E. H¨ullermeier. Rough set-based feature selection for weakly labeled data[J]. International Journal of Approximate Reasoning, 2021, 136: 150-167.

[29] Z. Pawlak. Rough sets and intelligent data analysis[J]. Information Sciences, 2002, 147(1-4): 1-12.

[30] M. A. Tawhid, A. M. Ibrahim. Feature selection based on rough set approach, wrapper approach, and binary whale optimization algorithm[J]. International Journal of Machine Learning and Cybernetics, 2020, 11(3): 573-602.

[31] T. H. Xu, G. Y. Wang, J. Yang. Finding strongly connected components of simple digraphs based on granulation strategy[J]. International Journal of Approximate Reasoning, 2020, 118: 64-78.

[32] H. Fujita, A. Gaeta, V. Loia, et al. Hypotheses analysis and assessment in counterterrorism activities: a method based on OWA and fuzzy probabilistic rough sets[J]. IEEE Transactions on Fuzzy Systems, 2020, 28(5): 831-845.

[33] C. Zhang, D. Y. Li, J. Y. Liang. Multi-granularity three-way decisions with adjustable hesitant fuzzy linguistic multi-granulation decision-theoretic rough sets over two universes[J]. Information Sciences, 2020, 507: 665-683.

[34] J. Qian, X. Han, Y. Yu, et al. Multi-granularity decision-theoretic rough sets based on the fuzzy T-equivalence relation with new strategies[J]. Journal of Intelligent & Fuzzy Systems, 2023, 44(4): 5617-5631.

[35] X. B. Yang, S. C. Liang, H. L. Yu. Pseudo-label neighborhood rough set: Measures and attribute reductions[J]. International Journal of Approximate Reasoning, 2019, 105: 112-129.

[36] Q. H. Hu, D. R. Yu, J. F. Liu, et al. Neighborhood rough set based heterogeneous feature subset selection[J]. Information Sciences, 2008, 178(18): 3577-3594.

[37] K. Zhang, J. M. Zhan, W. Z. Wu. On multi-criteria decision-making method based on a fuzzy rough set model with fuzzy α-neighborhoods[J]. IEEE Transactions on Fuzzy Systems, 2021, 29(9): 2491-2505.

[38] Y. Y. Yao. Relational interpretations of neighborhood operators and rough set approximation operators[J]. Information Sciences, 1998, 111: 239–259.

[39] S. An, X. Y. Guo, C. Z. Wang, et al. A soft neighborhood rough set model and its applications[J]. Information Sciences, 2023, 624: 185-199.

[40] L. Yang, K. Y. Qin, B. B. Sang, et al. Dynamic fuzzy neighborhood rough set approach for interval-valued information systems with fuzzy decision[J]. Applied Soft Computing, 2021, 111: 107679.

[41] L. Zou, H. X. Li, W. Jiang, et al. An improved fish swarm algorithm for neighborhood rough set reduction and its application[J]. IEEE Access, 2019, 7: 90277-90288.

[42] J. D. Feng, Z. T. Gong. A novel feature selection method with neighborhood rough set and improved particle swarm optimization[J]. IEEE Access, 2022, 10: 33301-33312.

[43] A. T. Sahlol, M. A. Elaziz, M. A. A. Al-Qaness, et al. Handwritten arabic optical character recognition approach based on hybrid whale optimization algorithm with neighborhood rough Set[J]. IEEE Access, 2020, 8: 23011-23021.

[44] Y. D. Zhang, Z. D. Mao, J. T. Li, et al. Salient region detection for complex background images using integrated features[J]. Information Sciences, 2014, 281: 586-600.

[45] P. R. Kanna, P. Santhi. Unified deep learning approach for efficient intrusion detection system using integrated spatial-temporal features[J]. Knowledge-Based Systems, 2021, 226: 107132.

[46] Z. C. Gong, Y. X. Liu, T. H. Xu, et al. Unsupervised attribute reduction: improving effectiveness and efficiency[J]. International Journal of Machine Learning and Cybernetics, 2022, 13(11): 3645-3662.

[47] X. B. Yang, Y. Y. Yao. Ensemble selector for attribute reduction[J]. Applied Soft Computing, 2018, 70: 1-11.

[48] 杨习贝. 不完备信息系统中粗糙集理论研究[D]. 南京: 南京理工大学, 2010.

[49] Y. H. Qian, J. Y. Liang, W. Pedrycz, et al. Efficient accelerator for attribute reduction from incomplete data in rough set framework[J]. Pattern Recognition, 2011, 44: 1658-1670.

[50] Q. H. Hu, D. R. Yu, Z. X. Xie. Neighborhood classifiers[J]. Expert Systems with Applications, 2008, 34(2): 866-876.

[51] D. C. Li, C. W. Liu. Extending attribute information for small data set classification[J]. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(3): 452-464.

[52] C. Wang, Z. She, L. B. Cao. Coupled attribute analysis on numerical data[C]. In: International Joint Conference on Artificial Intelligence, Beijing, China, 2013, pp. 1736-1742.

[53] Z. Chen, K. Y. Liu, X. B. Yang, et al. Random sampling accelerator for attribute reduction[J]. International Journal of Approximate Reasoning, 2022, 140: 75-91.

[54] Q. Chen, T. H. Xu, J. J. Chen. Attribute reduction based on lift and random sampling[J]. Symmetry, 2022, 14(9): 1828.

[55] H. M. Chen, T. R. Li, X. Fan, et al. Feature selection for imbalanced data based on neighborhood rough sets[J]. Information Sciences, 2019, 483: 1-20.

[56] Y. Chen, P. X. Wang, X. B. Yang, et al. Granular ball guided selector for attribute reduction[J]. Knowledge-Based Systems, 2021, 229: 107326.

[57] X. Y. Jia, Y. Rao, L. Shang, et al. Similarity-based attribute reduction in rough set theory: A clustering perspective[J]. International Journal of Machine Learning and Cybernetics, 2020, 11(5): 1047-1060.

[58] Q. H. Hu, L. Zhang, D. G. Chen, et al. Gaussian kernel based fuzzy rough sets: Model uncertainty measures and applications[J]. International Journal of Approximate Reasoning, 2010, 51(4): 453-471.

[59] Q. H. Hu, W. Pedrycz, D. R. Yu, et al. Selecting discrete and continuous features based on neighborhood decision error minimization[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B, 2009, 40(1): 137-150.

[60] W. T. Li, H. X. Zhou, W. H. Xu, et al. Interval dominance-based feature selection for interval-valued ordered data[J]. IEEE Transactions on Neural Networks and Learning Systems, 2023, 34(10): 6898-6912.

[61] W. T. Li, S. C. Zhai, W. H. Xu, et al. Feature selection approach based on improved Fuzzy C-Means with principle of refined justifiable granularity[J]. IEEE Transactions on Fuzzy Systems, 2023, 31(7): 2112-2126.

[62] X. S. Rao, X. B. Yang, X. Yang, et al. Quickly calculating reduct: an attribute relationship based approach[J]. Knowledge-Based Systems, 2020, 200: 106014.

[63] K. Y. Liu, X. B. Yang, H. Fujita, et al. An efficient selector for multi-granularity attribute reduction[J]. Information Sciences, 2019, 505: 457-472.

[64] Y. Y. Yao, Y. Zhang, J. Wang. On reduct construction algorithms[J]. Transactions on Computational Science II, 2008, 5150: 100-117.

[65] M. Chapman-Rounds, U. Bhatt, E. Pazos, et al. FIMAP: Feature importance by minimal adversarial perturbation[C]. In: Association for the Advancement of Artificial Intelligence, Vancouver, Canada, 2021, vol. 35, no. 13, pp. 11433-11441.

[66] N. Inkawhich, W. Wen, H. Li, et al. Feature space perturbations yield more transferable adversarial examples[C]. In: IEEE Conference on Computer Vision and Pattern Recognition, CA, USA, 2019, pp. 7066-7074.

[67] V. Aksakalli, M. Malekipirbazari. Feature selection via binary simultaneous perturbation stochastic approximation[J]. Pattern Recognition Letters, 2016, 75: 41-47.

[68] Y. Liu, W. L. Huang, Y. L. Jiang, et al. Quick attribute reduct algorithm for Neighborhood Rough Set Model[J]. Information Sciences, 2014, 271: 65-81.

[69] Y. Chen, K. Y. Liu, J. J. Song, et al. Attribute group for attribute reduction[J]. Information Sciences, 2020, 535: 64-80.

[70] M. Zhou, Y. Long, W. Zhang, Q. Pu, et al. Adaptive genetic algorithm-aided neural network with channel state information tensor decomposition for indoor localization[J]. IEEE Transactions on Evolutionary Computation, 2021, 25 (5): 913-927.

[71] S. Nadarajah. An explicit selection intensity of tournament selection-based genetic algorithms[J]. IEEE Transactions on Evolutionary Computation, 2008, 12(3): 389-391.

[72] S. Mostafaei, A. Ahmadi, J. Shahrabi. USWAVG-BS: Under-Sampled Weighted AVeraGed BorderlineSMOTE to handle data intrinsic difficulties[J]. Expert Systems with Applications, 2023, 227: 120379.

[73] J. Q. Guo, H. Y. Wu, X. L. Chen, et al. Adaptive SV-Borderline SMOTE-SVM algorithm for imbalanced data classification[J]. Applied Soft Computing, 2024, 150: 110986.

[74] Z. Z. Xu, D. R. Shen, T. Z. N, et al. A cluster-based oversampling algorithm combining SMOTE and k-means for imbalanced medical data[J]. Information Sciences, 2021, 572: 574-589.

攻读硕士学位期间取得的研究成果

期刊论文：

[1] Shuaishuai Zhang, Keyu Liu, Taihua Xu, Xibei Yang, Ao Zhang. A meta-heuristic feature selection algorithm combining random sampling accelerator and ensemble using data perturbation[J]. **Applied Intelligence**, 2023, 53: 29781-29798. (SCI 检索, WOS:001097012500004)

致 谢

鲁迅先生曾说：“我们自动的读书，即嗜好的读书，请教别人是大抵无用，只好先行泛览，然后决择而入于自己所爱的较专的一门或几门；但专读书也有弊病，所以必须和现实社会接触，使所读的书活起来。”三年的硕士生涯也即将结束是该与生活拥抱了。

尊敬的导师徐泰华副教授和杨习贝院长，以及计算机学院的各位教授，包括高尚教授、于化龙教授、束鑫教授、王平心副教授和宋晶晶副教授，您们的细致指导和无私奉献极大地丰富了我的学术旅程，指引我在科研和学习中不断前进。

对于计算机学院的同学们和师兄师姐们，包括巴婧师姐、陈振师哥、郭启航同学、印振宇同学、孙嘉琪同学、王浩宇同学、张啸宇同学、华明烽同门和何会兴同门，我也表示衷心的感谢。在整个研究生学习期间，你们的支持和鼓励对我非常重要，帮助我在学术探索中稳步前行。

特别要感谢张家辉室友、吴宏杰同学、王树祥同学和孟匡胤同学，你们在学习和生活中给予的关心与照顾，使我在遇到挑战时更加坚定和自信。

此外，我深感感激我的父母及家人对我的爱与支持，你们是我持续奋斗的最大动力。在我面对学业和生活挑战的每一个困难时刻，你们始终是我的坚强后盾。

最后，我对自己始终不放弃追求梦想的决心表示赞赏。正如李大钊先生所言，我们应用青春的力量去建设一切。希望我们都能乘风破浪，勇敢地追求未来的无限可能。

我要感谢所有帮助和陪伴我的人，因为你们的支持，我今天能取得如此成就。在未来的道路上，我将继续努力，保持初心，勇往直前。