**学校编码：10384 分类号 密级**

**学 号：24320151153220 UDC**



**硕 士 学 位 论 文**

**纠错输出编码方法的研究和应用**

**Research and Application of Error Correct Output Code**

**钟天云**

|  |  |
| --- | --- |
| **指导教师姓名：** | **王备战 教授** |
| **专业名称：** | **软件工程** |
| **论文提交日期：** | **2018年 4 月** |
| **论文答辩日期：** | **2018年 5 月** |
| **学位授予日期：** | **2018年 月** |

**指 导 教 师：**

**答辩委员会主席：**

**2017年 4 月**

**厦门大学学位论文原创性声明**

本人呈交的学位论文是本人在导师指导下,独立完成的研究成果。本人在论文写作中参考其他个人或集体已经发表的研究成果，均在文中以适当方式明确标明，并符合法律规范和《厦门大学研究生学术活动规范（试行）》。

另外，该学位论文为（ ）课题（组）的研究成果，获得（ ）课题（组）经费或实验室的资助，在（ ）实验室完成。（请在以上括号内填写课题或课题组负责人或实验室名称，未有此项声明内容的，可以不作特别声明。）

本人声明该学位论文不存在剽窃、抄袭等学术不端行为，并愿意承担因学术不端行为所带来的一切后果和法律责任。

声明人 （签名）：

指导教师（签名）：

年 月 日

**厦门大学学位论文著作权使用声明**

本人同意厦门大学根据《中华人民共和国学位条例暂行实施办法》等规定保留和使用此学位论文，并向主管部门或其指定机构送交学位论文（包括纸质版和电子版），允许学位论文进入厦门大学图书馆及其数据库被查阅、借阅。本人同意厦门大学将学位论文加入全国博士、硕士学位论文共建单位数据库进行检索，将学位论文的标题和摘要汇编出版，采用影印、缩印或者其它方式合理复制学位论文。

本学位论文属于：

（ ）1.经厦门大学保密委员会审查核定的涉密学位论文，于　　 年　 月 　日解密，解密后适用上述授权。

（ ）2.不保密，适用上述授权。

（请在以上相应括号内打“√”或填上相应内容。涉密学位论文应是已经厦门大学保密委员会审定过的学位论文，未经厦门大学保密委员会审定的学位论文均为公开学位论文。此声明栏不填写的，默认为公开学位论文，均适用上述授权。）

声明人（签名）：

年 月 日

**摘 要**

纠错输出编码（ECOC）是一种多类分类的集成学习方法框架，比单个分类器具有更好的分类效果。ECOC算法的研究至今仅有二十年时间，但已广泛应用于人脸识别、交通信号识别、微阵列数据分析等领域。纠错输出编码框架包含编码和解码两部分，相关的研究者主要针对这两个方面提出不同的算法理论，而研究的重点是编码部分。本文从编码和解码方面分别对纠错输出编码进行探索。

在解码方面，本文介绍了解码方法的作用和种类，以及常用的几种解码方法。文中基于弱欧氏距离解码（AED）的多零倾向等不足，提出了新的解码方法：比例欧式距离解码（RED），并结合文中提出的编码算法完成了两个完整的新ECOC算法框架。

在编码方面，本文介绍编码的重要性和种类。在分析一些经典编码方法的基础上，本文提出两个新的编码方法。（1）是针对微阵列数据的自调节编码算法（SA-ECOC）。该算法针对微阵列数据高维小样本的特点设计了动态自适应的编码算法进行分类，即先生成初始编码矩阵，然后根据数据的分类难度基于迭代增加矩阵的列。该算法还提出一种选列的标准“距离间隔”，保证新加入的列对分类结果有提升。与DECOC等经典ECOC算法的对比实验表明，SA-ECOC分类准确率略优且更稳定。（2）应用于不同UCI数据的个性化编码方法（CECOC）。该方法对数据样本的特点分析，为不同样本设计个性化编码矩阵。基于多个UCI数据集的验证结果显示，CECOC方法比没有个性化编码的矩阵效果更好，且更稳定。

本文还介绍现有ECOC工具包，并介绍了本人用Python自行开发并开源的ECOC工具包，在附录里添加工具包使用方法的简单介绍。

**关键词**：ECOC；微阵列数据；集成学习；多分类

**Abstract**

Error correcting output coding (ECOC) is a framework of ensemble learning method for multi-class classification, which offers a better classification effect than a single classifier. After twenty years of its birth, ECOC has been applied to face recognition, traffic signal recognition, microarray data analysis and other fields. The error correcting output coding framework consists of two stages: coding and decoding. The researches in this field concerns of these two aspects, but the most of work focuses on the design of coding algorithm. In this dissertation, the coding and decoding of error correction output coding are explored respectively.

In the aspect of decoding, this dissertation introduces the functions and types of decoding methods, and also introduces several common decoding methods. The attenuated Euclidean distance decoding (AED) is analyzed, and two shortcomings are put forward. Then a new decoding method, the ratio Euclidean distance decoding (RED), is proposed.

In the aspect of coding, this dissertation introduces the importance and different types of coding algorithms. It also introduces several classical coding methods, based on which two new coding methods are proposed. (1) Self-adjusting error correct output coding algorithm (SA-ECOC) for microarray data. The algorithm designs coding matrix according to the unique characteristics of microarray data. This method first generates a coding matrix, and then iteratively add columns. The algorithm also proposes a column selection standard "distance interval" to ensure that the newly added columns improve the classification results. SA-ECOC was used in the experiments based on cancer microarray data. Some popular ECOC algorithms are deployed for comparisons. The results showed that the accuracy of SA-ECOC classification was a little higher and more stable. (2) Customization coding method (CECOC) applied to ordinary data. This method analyzes the characteristics of data samples, and designs characteristic coding matrix for different samples. The method is verified by UCI data sets, and the experimental results show that the CECOC method is better and more stable than the matrix without custom coding.

In addition, this dissertation introduces the existing ECOC toolkit. Specially, it introduces a Python ECOC toolkit that is developed and uploaded by the author. Add a brief introduction to the toolkit use method in the appendix.

**Key Words**: ECOC; microarray data; multi-class; ensemble learning.

# 目录

[第一章 绪论 1](#_Toc509758346)

[1.1纠错输出编码研究背景和意义 1](#_Toc509758347)

[1.1.1 纠错输出编码研究发展历史 2](#_Toc509758348)

[1.2 DNA微阵列 3](#_Toc509758349)

[1.2.1微阵列数据的特点 4](#_Toc509758350)

[1.2.2微阵列数据分析方法 5](#_Toc509758351)

[1.2.3 纠错输出编码应用于分类微阵列数据 7](#_Toc509758352)

[1.3 ECOC工具包介绍 7](#_Toc509758353)

[1.3.1 matlab工具包 7](#_Toc509758354)

[1.3.2 Python工具包 8](#_Toc509758355)

[1.4 本文结构介绍 8](#_Toc509758356)

[1.5本章小结 9](#_Toc509758357)

[第二章、相关理论介绍 10](#_Toc509758358)

[2.1 监督学习 10](#_Toc509758359)

[2.2 集成学习 10](#_Toc509758360)

[2.3 特征选择 12](#_Toc509758361)

[2.3.1 特征选择方法分类 12](#_Toc509758362)

[2.3.2 常用特征选择方法 13](#_Toc509758363)

[2.4 纠错输出编码 14](#_Toc509758364)

[2.4.1 ECOC原理 15](#_Toc509758365)

[2.4.2 二元编码和三元编码 17](#_Toc509758366)

[2.4.3 固定编码 17](#_Toc509758367)

[2.4.4随机编码 18](#_Toc509758368)

[2.4.5 数据依赖编码 19](#_Toc509758369)

[2.5解码方法介绍 21](#_Toc509758370)

[2.5.1 二元解码和三元解码 21](#_Toc509758371)

[2.5.2 硬解码和软解码 21](#_Toc509758372)

[2.5.3 常用解码方法 22](#_Toc509758373)

[2.6 改善弱欧式解码 22](#_Toc509758374)

[2.6.1 弱欧式解码的弊端 23](#_Toc509758375)

[2.6.2 优化计算量 23](#_Toc509758376)

[2.6.3 RED解码解决“多零倾向” 23](#_Toc509758377)

[2.7 本章小结 24](#_Toc509758378)

[第三章、自调节编码ECOC算法 26](#_Toc509758379)

[3.1 算法介绍 26](#_Toc509758380)

[3.1.1 算法思路 26](#_Toc509758381)

[3.1.2相关内容：数据复杂度 27](#_Toc509758382)

[3.1.3算法第一步：构造编码矩阵 28](#_Toc509758383)

[3.1.4 算法第二步：自动拓展矩阵 30](#_Toc509758384)

[3.2实验介绍 33](#_Toc509758385)

[3.2.1数据集介绍 33](#_Toc509758386)

[3.2.2特征覆盖度 34](#_Toc509758387)

[3.2.3矩阵长度 35](#_Toc509758388)

[3.2.4迭代过程 37](#_Toc509758389)

[3.2.5最终分类结果与对比 39](#_Toc509758390)

[3.2.6 实验总结 41](#_Toc509758391)

[3.3本章小结 42](#_Toc509758392)

[第四章、基于距离个性化编码ECOC算法 43](#_Toc509758393)

[4.1方法介绍 43](#_Toc509758394)

[4.1.1 方法思路 43](#_Toc509758395)

[4.1.2 详细介绍 44](#_Toc509758396)

[4.1.3 距离测度 44](#_Toc509758397)

[4.2 UCI数据集上的实验与分析 46](#_Toc509758398)

[4.2.1 实验数据集 46](#_Toc509758399)

[4.2.2 预处理方法 47](#_Toc509758400)

[4.2.3 选择特征个数 48](#_Toc509758401)

[4.2.4 实验结果 50](#_Toc509758402)

[4.2.5 不同距离比较 52](#_Toc509758403)

[4.2.6 实验总结 53](#_Toc509758404)

[4.3微阵列数据实验和分析 53](#_Toc509758405)

[4.3.1数据集介绍&预处理方法 54](#_Toc509758406)

[4.3.2选择特征个数 54](#_Toc509758407)

[4.3.3实验结果 57](#_Toc509758408)

[4.4本章小结 59](#_Toc509758409)

[第五章、ECOC开发工具包 61](#_Toc509758410)

[5.1 ecoctlk介绍 61](#_Toc509758411)

[5.1.1 ECOC框架类 61](#_Toc509758412)

[5.1.2 编码方法 62](#_Toc509758413)

[5.1.3 解码方法 62](#_Toc509758414)

[5.1.4 数据复杂度 63](#_Toc509758415)

[5.1.5 特征选择方法 63](#_Toc509758416)

[5.2 本章小结 64](#_Toc509758417)

[第六章、总结与展望 65](#_Toc509758418)

[6.1 总结 65](#_Toc509758419)

[6.2 展望 66](#_Toc509758420)

[附录 67](#_Toc509758421)

[参考文献 75](#_Toc509758422)

[致谢 78](#_Toc509758423)

# Contents

[Chapter 1 Introduction 1](#_Toc508875163)

[1.1 Background and significance of ECOC 1](#_Toc508875164)

[1.1.1 history of ECOC researches 2](#_Toc508875165)

[1.2 DNA microarray 3](#_Toc508875166)

[1.2.1 microarray data 5](#_Toc508875167)

[1.2.2 microarray data analysis methods 5](#_Toc508875168)

[1.2.3 classify microarray data with ECOC 7](#_Toc508875169)

[1.3 ECOC toolkit 7](#_Toc508875170)

[1.3.1 matlab toolkit 8](#_Toc508875171)

[1.3.2 Python toolkit 8](#_Toc508875172)

[1.4 Structure arrangements 8](#_Toc508875173)

[1.5 Summary 9](#_Toc508875174)

[Chapter 2 Introduction of Relevant Theories 10](#_Toc508875175)

[2.1 supervised learning 10](#_Toc508875176)

[2.2 ensemble learning 10](#_Toc508875177)

[2.3 feature selection 12](#_Toc508875178)

[2.3.1 types of feature selection method 12](#_Toc508875179)

[2.3.2 common feature selection method 13](#_Toc508875180)

[2.4 error corret output coding 14](#_Toc508875181)

[2.4.1 ECOC principle 15](#_Toc508875182)

[2.4.2 binary coding and tenary coding 17](#_Toc509758366)

[2.4.3 fixed coding 17](#_Toc509758367)

[2.4.4 random coding 18](#_Toc509758368)

[2.4.5 data-depended coding 19](#_Toc509758369)

[2.5 Introduction of decoding methods 21](#_Toc508875186)

[2.5.1 binary decoding and ternary decoding 21](#_Toc508875187)

[2.5.2 hard decoding and soft decoding 21](#_Toc508875188)

[2.5.3 common decoding methods 22](#_Toc508875189)

[2.6 Optimize attenuated euclidean decoding 22](#_Toc508875190)

[2.6.1 disadvantages of attenuated euclidean decoding 23](#_Toc508875191)

[2.6.2 calculation reduction 23](#_Toc508875192)

[2.6.3 RED solve zero tendency 23](#_Toc508875193)

[2.7 Summary 24](#_Toc508875194)

[Chapter 3 Self-adjusting ECOC 26](#_Toc508875195)

[3.1 Algorithm introduction 26](#_Toc508875196)

[3.1.1 thinking 26](#_Toc508875197)

[3.1.2 relevance: data complexity 27](#_Toc508875198)

[3.1.3 step one: construct coding matrix 28](#_Toc508875199)

[3.1.4 step two: iteratively extend coding matrix 30](#_Toc508875200)

[3.2 Experiments 33](#_Toc508875201)

[3.2.1 dataset introduction 33](#_Toc508875202)

[3.2.2 data overlapping 34](#_Toc508875203)

[3.2.3 length of matrix 36](#_Toc508875204)

[3.2.4 iteration process 38](#_Toc508875205)

[3.2.5 comparation of classification results 40](#_Toc508875206)

[3.2.6 summary and prospect 41](#_Toc508875207)

[3.3 Summary 41](#_Toc508875208)

[Chapter 4 Distance-based Customization ECOC 43](#_Toc508875209)

[4.1 Method introduction 43](#_Toc508875210)

[4.1.1 thinking 43](#_Toc508875211)

[4.1.2 detailed introduction 44](#_Toc508875212)

[4.1.3 distance measure 44](#_Toc508875213)

[4.2 Experiments and analysis 46](#_Toc508875214)

[4.2.1 experimental dataset 46](#_Toc508875215)

[4.2.2 preprocessing method 47](#_Toc508875216)

[4.2.3 number of selected features 47](#_Toc508875217)

[4.2.4 experimental result 50](#_Toc508875218)

[4.2.5 comparasion of different types of distance 51](#_Toc509078547)

[4.2.6 summary of experiments 53](#_Toc509078548)

[4.3 Summary 53](#_Toc509078549)

[Chapter 5 ECOC Toolkit 54](#_Toc509078550)

[5.1 Introduction of ecoctlk 54](#_Toc509078551)

[5.1.1 ECOC frame class 54](#_Toc509078552)

[5.1.2 coding methods 55](#_Toc509078553)

[5.1.3 decoding methods 55](#_Toc509078554)

[5.1.4 data complexity 56](#_Toc509078555)

[5.1.5 feature selection methods 56](#_Toc509078556)

[5.2 Summary 57](#_Toc509078557)

[Chapter 6 Conclusion and Prospect 58](#_Toc509078558)

[6.1 Conclusion 58](#_Toc509078559)

[6.2 Prospect 59](#_Toc509078560)

[Appendix 60](#_Toc509078561)

[Reference 68](#_Toc509078562)

[Acknowledgements 71](#_Toc509078563)

# 绪论

微阵列技术是一项生物基因检测技术，对其数据的研究分析将有效解决疾病预防的问题。然而微阵列数据往往包含多个类别，其“特征多样本少”的特点使其分类分析方法与其他种类的数据集不同。纠错输出编码（error correct output code, ECOC）是一种集成学习方法，用于解决机器学习中数据的多类分类问题。大约从2000年开始，ECOC被应用于微阵列数据的多多分类。本章1.1节就纠错输出编码的背景和研究历程进行介绍；1.2节介绍微阵列技术与数据分析方法；1.3节介绍当前ECOC工具包优点与不足。

## 1.1纠错输出编码研究背景和意义

纠错输出编码来自于通信领域，是纠正通信过程中消息传输错误的技术。这项技术最早于1995年被应用到统计机器学习领域，作为一种集成学习方法（框架）提高多类数据的分类问题。

机器学习是一门多领域交叉学科，涉及概率论、统计学、计算机科学等。研究计算机如何模拟或实现人类的学习行为，以获取新的知识或技能。机器学习下分类问题是应用面最广的一类问题，是从已知类别的数据中学习知识，训练模型，预测未知类别数据的问题。机器学习发展多年以来，诞生许多优秀的分类算法，如K近邻、决策树、支持向量机等，但是在实际应用中，对于类别标签多于2的多类分类问题，单个学习算法不容易取得准确的效果。因为其不能够有效地拟合整体样本；或者，样本的多样性致使分类模型的复杂度急剧增加，灵活度减少，难以应付样本的变动，引发过拟合现象。除此以外，对于一些成熟的分类算法，例如SVM，因为其设计原理而仅仅适用于解决二分类问题。

ECOC作为一种集成学习框架，将多分类问题分解成多个二分类问题，再将二分类结果解码为多类分类结果。经过二十余年的发展，ECOC已经被广泛应用于人脸识别、图像识别、微阵列数据分类等多个领域。关于ECOC更详细的介绍放在第二章。

### 1.1.1 纠错输出编码研究发展历史

ECOC理论方面研究发展如下：

|  |
| --- |
| (1995) Dietterich和Bakiri[1, 2]第一次提出用ECOC处理数据多分类。  (1995) Kong和Dietterich[3] 证明了基于纠错输出编码的算法框架可以减少二分类学习器产生的偏置和方差错误。  (2000) Masulli 和 Valentini[4]给出了影响ECOC编码算法有效性的关键因素，包括编码行的相关性、二分类器的结构特点和分类准确率，以及数据本身的复杂性。依然没有一些通用的规则指引学习最优化的ECOC编码矩阵。  (2002)Crammer 和 Singer [5]证明了问题相关领域的最优化的ECOC编码矩阵设计是一个NP问题。  (2002) Allwein, Schapire和Singer[6]. 给出了基于边界的分类器(margin-based classifier)作为基础的二分类器的情况下的一些理论分析和实验结果，并给出了建议的 code 长度为L=10logK, L=15logK  (2011)李建武[7]从最小封闭球集合描述的角度出发，对二元多类分类器进行了几何解释。 |

ECOC遗传算法编码方面的研究如下：

|  |
| --- |
| (2008) Pedrajas 和 Fyfe[8]使用基于遗传算法的CHC算法优化稀疏随机编码矩阵。在该方法中，不考虑训练数据集的分布特点，将二分类学习器的个数限制在 [30, 50] 的区间内。  (2007)Lorena 和 Carvalho[9]同样是结合遗传算法和稀疏随机法，进而优化编码矩阵的构建过程；与前一种方法对比区别在于编码矩阵行向量的长度限制在 [log2K,K] 区间内。  (2011)Miguer 和 Sergio[10]提出了新的遗传算子，达到了避免出现无效的个体、减少遗传算法的搜索空间的效果。  (2012)Bautista[11]等人提出最小 ECOC 编码矩阵算法，使用标准遗传算法在不降低区分能力的前提下将二分类学习器的个数降低到 [log2K]。 |

ECOC编码方法的研究如下：

|  |
| --- |
| (1999) Alpaydin和Mayoraz[12]首次提出基于数据的反向传播编码，开辟了数据依赖编码的领域。  (2006) Pujol和Radeva[13]给出了discriminant ECOC（DECOC）， 利用二叉树生成编码矩阵。二叉树分支节点表示一个二分类器，二叉树的叶子节点表示一个所属的类别标签。这样的方式也被后来的研究沿用。  (2007) Escalera, Oriol和Pujol[14]给出一种基于DECOC的增列方法。利用多颗DECOC二叉树构成的森林拓展编码矩阵。  (2008) Pujol和Escalera[15]给出一种基于DECOC的迭代验证增列的ECOC编码方法：ECOC-ONE。ECOC-ONE在DECOC编码的基础上，利用验证数据不断为原编码增长，直到长度到达2K为止。  (2008) Escalera和David[16]提出一种针对基类线性不可分的情况，对基类再向下划分的subclass ECOC的编码方式。  (2012)Sergio Escalera[11]提出一种极少量编码的编码方式。  (2013)为了增加基分类器的独立性和多样性，Bagheri[17]引入特征子集的概念，与“类别”和“分类器”一起构成一个三维编码空间。基分类器使用不同特征子集训练得来。 |

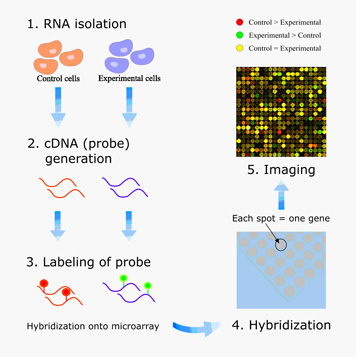
## 1.2 DNA微阵列

DNA微阵列（DNA microarray），又称DNA芯片（gene chip）或者生物芯片（biochip），是近几十年发展起来的一种能快速、高效检测DNA片段序列、基因型及其多态性或基因表达水平的新技术。科学家使用DNA微阵列来同时测量大量基因的表达水平，或对基因组的多个区域进行基因型分析。微阵列技术主要的优势在于能全面性、大量地检测几万个基因的表达。1998年美国Affymetirx公司将第一片基因芯片推向市场，标志着DNA微阵列的产业化，从此基因芯片或DNA微阵列的研究和应用得到了广泛的重视。使用基因芯片，可在短时间内找出可能受疾病影响的基因，作为早期诊断的生物标记(biomarker)。它可用于生物学和临床研究，并已成功地应用于癌症诊断。归一化后的数据可用于理论研究或应用研究（包括疾病诊断和药物开发）。



**图1-1： Affymetrix公司的基因芯片**

（资料来源：Wikipedia - DNA Microarray）



**图1-2： 基于cDNA 微阵列的基因表达检测实验基本流程图**

（资料来源：Wiki spaces - Southern Blotting and DNA Microarray）

### 1.2.1微阵列数据的特点

微阵列数据（microarray data），又称基因表达数据（gene expression data），是微阵列技术产生的数据。由于这类技术的规模化、微型化和高度自动化的特性，他们所生成的数据量非常庞大。不仅如此，其数据形态比一般实验数据更加复杂。目前有大量统计学和计算机学的研究致力于处理这些数据。

机器学习数据挖掘领域的研究者已经发表了越来越多的微阵列数据分析研究工作，包括聚类[18]和分类[19]。然而，微阵列数据的特点是特征极多样本极少，且多数数据存在不平衡问题（某些类别样本数量过多或过少）。从机器学习的角度来说，有以下特点的数据适合于进行分类任务：

|  |
| --- |
| 1. 大量样本数； 2. 有代表性且数目适量的特征； 3. 均匀分类的类别。 |

基因微阵列数据正好与上述特点相反。这些特性使得针对多类微阵列数据的分类是一个困难的任务。

微阵列数据可以被视为一个数值型二维矩阵，其中的行向量表示一个样本的所有基因表达强弱，而列向量表示一个基因在所有样本上的基因表达强弱。微阵列技术的探针点数量庞大，因此基因数量通常在5,000-15,000的量级。但是由于费用昂贵、实验复杂，样本数量却稀少且珍贵，样本个数通常小于100。这种样本与特征间过度不平衡的特点被称为“高维度、小样本”。一方面，DNA微阵列技术的出现为生物学研究、遗传疾病的诊断、以及药物研发和带来前所未有的机遇，而另一方面，其基因微阵列数据具有高维度、小样本的特点，传统的数据分析方法常常不能获得希望的结果。一般而言维度之间并不完全独立，潜在相互关系又非常复杂，引发两种趋势：大量有用的特征被覆盖或隐藏，或者计算复杂度也急剧增加。此外，基因微阵列数据还生而带有高变异和高噪声等难点，这进一步加大了分析基因微阵列数据的难度。

### 1.2.2微阵列数据分析方法

目前，基因微阵列数据分析的方法主要包括：

一、数据处理：评估数据的重复性和可靠性；数据规格标准化；量化基因表达强度；基因筛选或者剔除奇异样本等，可以使研究人员较有效地利用基因微阵列数据。数据处理是基因微阵列数据分析前的关键步骤，对于消除人为失误、降低系统误差，及保证基因表达水平的重复性、可比性有重要的意义。许多研究致力于探索和改进其处理方法，形成了该研究领域内的一项重要研究分支。常见的预处理包括：数据归一化（normalization）、奇异值（outlier）剔除、缺省值（missing value）修补等。

二、图像分析：一是指基因微阵列数据可视化或者分析结果可视化。二是，对微阵列生物实验取得的图像数据，采用热点识别（spot recognition）网格化等图像算法[20]，通过图像达到标识有效基因或者移除冗余特征等目的。

三、特征选择或提取：在分析基因微阵列数据之前，一般需要减少维度。特征选择或提取也被统称为纬度规约。例如主成份分析（principal components analysis, PCA）[21]、local linear embedding[22]、 locally preserving projections[23]、sammon's mapping[24]以及laplacian eigenmaps[25]等。

四、基于假设驱动的统计分析：对于基因微阵列数据，采用ANOVA, t-test，Mann-Whitney test，贝叶斯等方法，区别基因表达强度在样本之间所具有的统计学差异[26]。这类方法通常假设数据存在某种形式的分布，计算数据分布与该形式差异的统计学意义，进而减少后续分析中“存伪”和“弃真”两类错误。此外这种统计显著性的强弱，可以用作特征选择。

五、无监督学习：将基因或者样本进行聚类学习，常用的聚类算法包括k-means，Self-Organizing Maps（SOM）[27]，以及层次聚类算法等，在构造类簇（cluster）时，计算两两基因对之间的距离。这种距离的定义是广义的，距离的概念可以是欧式距离，线性复杂度等。但是基因微阵列数据结构本身具有多样性，无法通过类簇的概念简单地描述[28]。因此，为了探索挖掘基因相互之间非线性关系的聚类算法，一些研究人员考虑时间开销，提出一种折中方案，相关提升算法包括CLIFF[29], biclustering[30], FLAME[31]和GeneClust[32]等。

六、监督式学习：根据基因微阵列数据的特点，依据已知标签样本构建分类模型。可以用来对未知标签的样本进行分类。常用的分类算法包括k-近邻算法、决策树、随机森林、神经网络、以及线性回归等。此外，研究人员探索了基于不同种类的进化算法，例如蚁群算法、遗传算法、以及粒子群算法等。在监督学习中，基因微阵列数据通过基于不同测度的特征选择方法筛选，例如Gini指数、信息增益等。

总之，基因微阵列技术不仅仅涉及生物信息学的内容，而且包含数据的处理和分析方法。

### 1.2.3 纠错输出编码应用于分类微阵列数据

基于微阵列数据的纠错输出编码算法的研究应用方兴未艾，尚存许多难题需要深入研究。微阵列数据多分类的难点可以简单概括为“高维度、小样本”。目前，研究人员探索了许多解决微阵列数据多分类问题的方法，例如

|  |
| --- |
| (2011)Elizabeth T等[33]使用递归 ECOC 算法对微阵列数据集进行多分类研究。他们采用归约的基因集合来解决高纬度的问题，并进一步分析了一个二分类莫平行能够处理的最多的基因数量。  (2012)Yu H. L[34].等人利用分类置信度配合SVM算法对微阵列数据直接进行多分类学习，主要研究的是微阵列数据的生物信息原理。  (2013)M Tong, KH Liu[35]设计了基于基因对的SVM集成分类系统，不但能够进行癌症分类，而且可以同步识别致癌基因。 |

使用ECOC框架对微阵列数据分类的研究相对较少。

## 1.3 ECOC工具包介绍

从ECOC出现到现在历经二十几年的时间，学界的一些优秀的机器学习工具提供ECOC算法框架，主要包括ecoclib和fitecoc。fitecoc是matlab软件提供的，而ecoclib则由知名ECOC研究组提供。这两款工具都是基于matlab语言。相比matlab，使用Python实现的工具包数量少，且完整性不够。为了弥补这一缺憾，本研究作者开发了一款基于Python的ECOC工具包供项目组使用。该工具包具体功能将在第六章介绍。

### 1.3.1 matlab工具包

* ecoclib[36]

包括OVO ,OVA, Forest, DECOC, ECOC-ONE, Sparse Random等编码；HD, ED, AED等解码。DECOC与ECOC-ONE作者整理发布的ECOC工具包。此外工具包还内置了几个二分类方法。

* fitecoc[37]

Matlab官方提供的ECOC工具包，主要包括OVO, OVA, Dense Random, Sparse Random等编码；HD, Binary loss等解码。此工具应用的到的解码理论来自于ecoclib的作者

### 1.3.2 Python工具包

* Scikit-learn[38]

Scikit-learn是Python开发者最喜欢的机器学习工具包之一，但是在ECOC方面Scikit-learn仅仅提供了OVO与OVA两种编码方法和一种需要开发者自己拓展的ECOC框架。Scikit-learn解码也仅仅提供欧氏距离解码。

* Shogun[39]

Shogun是可以应用于多种语言的工具，C++, java, Python等。他除了非数据依赖的编码方法之外，还包含了从ecoclib移植过来的DECOC编码方法。Shogun缺点在于只适用于linux的开发环境，而且在ECOC方面的功能做的不够好。

总的来说，Python的工具包对ECOC工具的开发很不重视，相比Matlab的工具包差得很远。主要Python工具包仅仅包含少有的几个简单编码和解码方法，而且在针对三元矩阵的训练和预测方面也存在大量设计上的缺陷。

## 1.4 本文结构介绍

本文主要分为七章。

第一章介绍微阵列数据、纠错输出编码背景和相关研究以及目前市面上ECOC工具包。

第二章介绍本文相关的理论基础，除了监督学习和特征选择等机器学习知识，还包括ECOC多分类原理和常见的编码、解码方法等。

第三章介绍一种自动延长的ECOC编码算法，该算法结合微阵列数据特点进行编码，对微阵列数据分类有较好的效果。

第四章介绍一种个性化ECOC编码算法，该算法根据距离测度为不同数据点设计不同的ECOC编码。

第五章介绍本文作者基于Python语言开发的ECOC工具包。

第六章总结全文并展望未来的研究方向。

## 1.5本章小结

本章1.1节粗略介绍了一种处理多类别分类的机器学习方法纠错输出编码，并且依据时间顺序罗列了一些重要的研究成果，包括ECOC理论层面研究、应用遗传算法编码研究和编码算法的研究。然后介绍ECOC应用于微阵列数据的研究成果。

1.2节简单介绍微阵列技术的背景，介绍微阵列技术对基因技术和医学发展的意义。并且描述微阵列数据的“高维度、小样本”的特点和数据处理方法。从2005年开始ECOC被应用于分类微阵列数据，本节也罗列了与微阵列数据相关的ECOC研究。

1.3节分别介绍基于matlab和Python语言的ECOC工具包的情况。

1.4节介绍全文内容结构。

# 第二章、相关理论介绍

本章介绍本文相关理论。包括监督学习、集成学习的相关理论；也包括本文第四、五章算法所用到的特征选择方法；还包括纠错输出编码相关理论。本章还详细介绍编码方法和解码方法的实现细节以及一些经典的研究成果。

## 2.1 监督学习

“现在看来，创造最多价值的还是监督学习。”

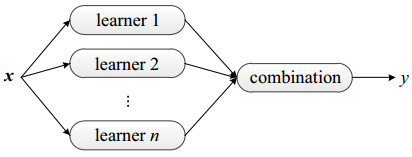
吴恩达在O’reilly举办的AI Conference上的讲话

监督学习（supervised learning）是机器学习中的一种，它利用带有标签的训练数据训练模型。这里的标签可以是表示类别的离散值，也可以是连续值。监督学习任务又可以简单的分为分类任务和回归任务。

分类（classification）任务，是机器学习任务的一种，其数据分为训练数据（training data）和测试数据（testing data）。数据由多个样本组成，在训练数据里，每一个样本（sample）都有n个特征（feature）和一个类别标签（label）。对应的测试数据里，每一个样本有同样多的n个特征，没有类别标签。分类任务的目标就是利用训练数据训练分类器（classifier），用分类器预测测试数据样本所属的类别标签。

## 2.2 集成学习

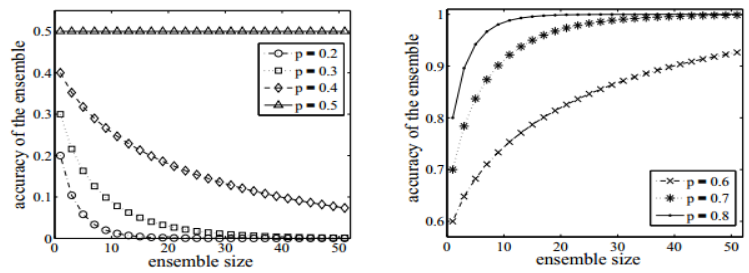
对于一些较难分类的数据，单个分类器的效果很有限，这个时候需要多个准确率高的分类器合作。集成学习（ensemble learning）采用多个解决同一个任务的基分类器（base learner），预测数据集得出一个综合的预测结果。这就像医生们碰到疑难杂症之后要专家们一起开会讨论，讨论出一个方案。集成学习的目标是提高分类器的分类能力。图2-1展示的是其结构。



**图2-1：集成学习框架图**

并不是集成的分类器越多，集成学习的效果就越好。图2-2展示的是集成分类准确率，当单分类器分类准确率小于0.5时，集成准确率随着基分类器数量增多而下降；当单分类器分类准确率大于0.5时，集成准确率随着基分类器数量增多而上升。而且，单分类器准确率越极端，集成准确率上升或下降的速率越快。因此可以得知良好的集成学习包含两个关键因素：

|  |
| --- |
| 1. 单分类器之间应该具有差异性。 2. 各单分类器的预测准确率要足够高。基本要求是p>0.5。 |



**图2-2：集成框架准确率vs分类器数量**

关于如何构建差异性分类器，一是可以采用采样技术，构建不同的数据集；二是利用特征选择方法构造不同的特征空间；三是可以调整分类模型的训练参数。而关于如何对分类器整合，一是可以采用投票法，少数赞成多数；二是利用加权投票法；三是利用概率投票法，因为有些分类器输出的不是类别标签，而是输出预测概率。

## 2.3 特征选择

“一个正确的数学模型应当在形式上是简单的。”

吴军《数学之美》

特征选择（feature selection）是机器学习中一种基本的数据处理方法,是从所有特征中挑选部分特征用于训练模型。构造机器学习的模型的目的是从原始的特征数据集中学习出问题的结构与问题的本质。而减少特征，挑选重要特征能够突出问题的本质，简化数学模型。特征选择方法的目标大致如下：

|  |
| --- |
| 1. 提高预测的准确性 2. 构造更快，消耗更低的预测模型 3. 能够对模型有更好的理解和解释 |

### 2.3.1 特征选择方法分类

特征选择的方法有很多种，大体上可以分为三类：Filter方法、Wrapper方法、Embedded方法。

1. Filter方法

主要思想是对每一维的特征“打分”，即给每一维的特征赋予权重，这样的权重就代表着该维特征的重要性，然后依据权重排序。此类方法的优点是可自定义选择特征的个数。主要的方法例如：卡方检验 (chi-squared test)、信息增益 (information gain)和相关系数 (correlation coefficient scores)。

1. Wrapper方法

主要思想是将特征子集的选择看作是一个搜索寻优问题，生成不同的特征组合，并对组合进行评价，再与其他的组合进行比较。这样就将子集的选择看作是一个优化问题，这里有很多的优化算法可以解决，尤其是一些启发式的优化算法，如GA，PSO，DE，ABC等。主要方法例如：递归特征消除算法 (RFE)。

1. Embedded方法

主要思想是在模型既定的情况下学习出对提高模型准确性最好的属性，或者在确定模型的过程中，挑选出那些对模型的训练有重要意义的属性。主要方法例如：正则化。

### 2.3.2 常用特征选择方法

本小节介绍第四、五章算法采用的特征选择方法，这7种方法的前5种是给特征打分的Filter类型，后两种分别是Embedded类型和Wrapper类型。本小节涉及公式规则如下：*C*表示类别的个数，*k*表示第*k*个类别，*j*表示第*j*个特征，*n*表示样本个数，*i*表示第*i*个样本。*x*表示数据（或者称样本），*y*表示数据标签。

**ANOVA F-value**[40]: ANOVA F-test几乎是在固定的假阳性错误率（false positive errors, FPE）下，最小化假阴性错误率（false negative errors, FNE）中最优检测方式。该检测的F值被用来筛选特征。F值高的特征价值高。这种特征选择是一种Filter方法，定义如公式(2-1)、公式(2-2)、公式(2-3)所示。

(2-1)

(2-2)

(2-3)

**互信息**（Mutual Information, MI）[41]：两个随机变量之间的互信息是一个非负值，它测量变量之间的依赖关系。当且仅当两个随机变量是独立的时，等于零，而高值则意味着更高的依赖性。计算特征与标签的依赖性可以评价特征。这种方法是Filter方法，定义如下：

(2-4)

**BSS/WS**[42]:是一个简单的且众所周知的基于Fisher线性判别分析[15]的特征测度。这个测度测量的是特征的类间平方和与类内平方和之比。BSS/WSS值大说明，类间距离较大，类别较容易区分，特征价值较高。这种特征选择是一种Filter方法，对于特征*j*，这个度量定义如下：

(2-5)

上式中，表示所有样本在特征j上数值的平均值，表示属于*k*类的所有样本在特征*j*上数值的平均值，是指示函数，表示当样本*i*的标签为*k*时，函数值是1，否则是0。

**方差**(variance): 计算的是特征的方差。方差高的特征被认定为好特征。这种方法属于Filter方法。

(2-6)

**卡方统计**（chi-squared）[43]：计算每个非负特征和类之间的卡方统计量。这种方法也属于Filter方法。卡方检验基本思想是通过理论值与实际值差值来判定理论的正确与否。如果偏差够小，可以认为误差属于自然误差，误差来自测量不准确或者偶然。如果偏差大到一定程度，便不太可能是测量不精确或者偶然产生导致。理论值为*E*，实际值为*x*，偏差程度的计算公式为：

(2-7)

**L1正则化**：将系数*w*的L1范数作为损失函数的惩罚项，因为正则项是非零的，这迫使弱特征的系数为0。因此，L1正则化倾向于使模型非常稀疏（系数*w*为0），使得L1正则化是一种很好的特征选择方法。这种特征选择方法属于Embedded方法。

**Extra Tree Classifier**[44]：（ETC）：ETC一种非常随机的树分类器，分类器在训练过程中会对特征进行评分，借此筛选特征。Extra Tree Classifier不同于经典决策树的构建方式。当寻找将一个节点的样本分成两个组的最佳分割时，对随机选择的特征中的每一个最大特征绘制随机分割，并选择其中的最佳分割。该方法通过特征的重要性来选择特征，被称为Gini指数。

(2-8)

上式中P(*k*)是第*k*个类的样本在该结点所占比例。

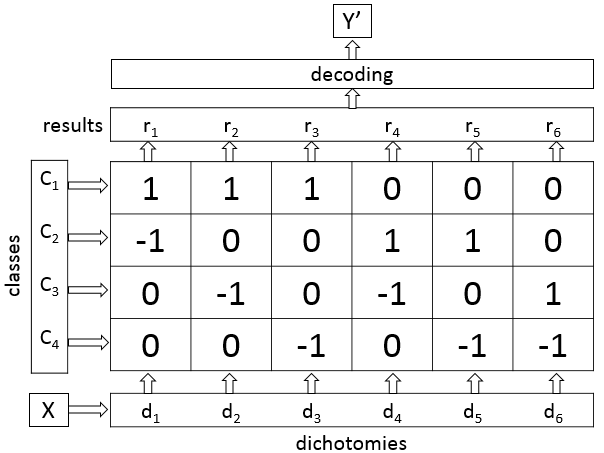
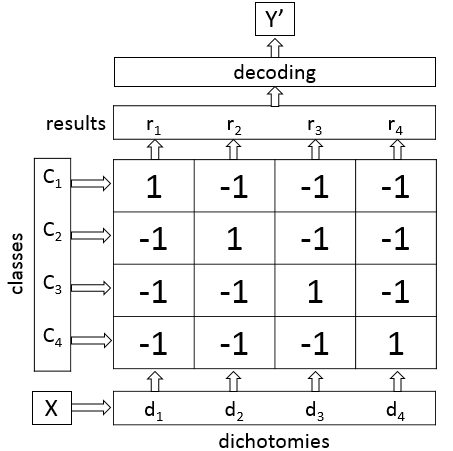
## 2.4 纠错输出编码

如本章2.1节描述，分类任务是利用带类别标签的训练数据训练分类模型，然后用训练好的模型预测不带标签的未知数据所属的类别。而分类任务可以进一步细分为二分类任务（binary classification）和多分类任务（multiclass classification）。二分类任务中，备选类别只有两个，待预测样本不是正类就是负类，常用分类模型种类有逻辑回归（logistic regression）、支持向量机（Support Vector Machine, SVM）、决策树（decision tree）、*k*近邻（*k* nearest neighbors）等。而实际生产生活的数据往往包含不止两类，类别数大于2的分类任务被称为多分类任务。除了上面所说的决策树和*k*近邻以外，多数分类模型无法直接应用于多分类任务。

纠错输出编码（error correct output code，ECOC）[1, 2]是解决多分类问题的一种集成算法框架，可以将多个擅长二分类任务的分类模型集成起来。ECOC算法框架分为两个部分：编码（encoding）和解码（decoding）。

### 2.4.1 ECOC原理

在编码中，ECOC生成一个编码矩阵（coding matrix）来指导二分类模型或者叫二分类器（dichotomy）的训练和预测。编码矩阵由“+1”，“-1”或者“0”三种编码组成。编码矩阵每一行代表一个类别（class），每一列代表一个二分类器。对于二分类器来说，它们只训练对应列上非0编码对应类别数据，并把所有“+1”对应类别的训练数据统统视为正类，把所有-1对应类别的训练数据统统视为负类。比如图(2-3)(a)中编码矩阵的第一列是“[1,-1,-1,-1]”，这表示第一列对应的二分类器*d*1把{*C*1}类别的数据视为正类，把{*C*2,*C*3,*C*4}类别的数据统视为负类。然后利用分成两类的训练数据训练模型*d*1，然后用同样的方法训练其他列对应的分类器。可以看到ECOC正是利用编码矩阵的方式，把多类数据分成两类，然后再应用于二分类器。图(2-3)是两种经典编码矩阵一对多编码（one vs all, OVA）和一对一编码（one vs one, OVO），两种编码在本章2.4.2 详细介绍。

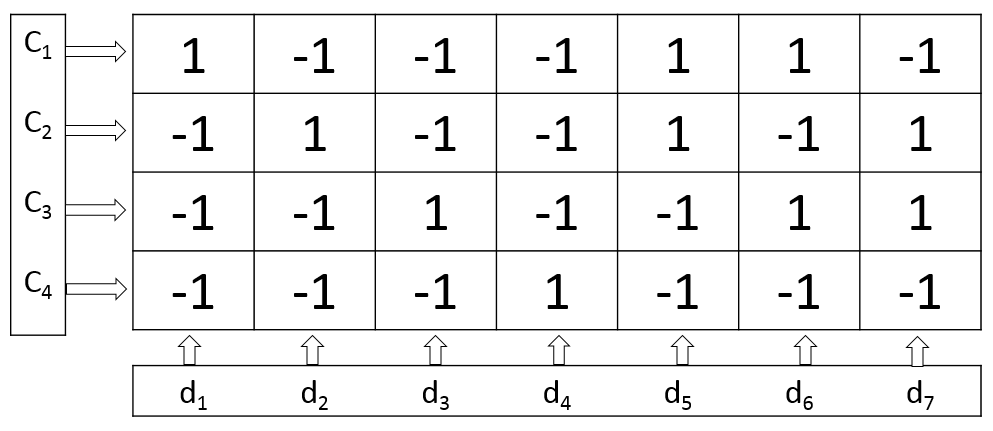


**(a) OVA编码 (b) OVO编码**

**图2-3：经典编码矩阵**

解码过程在编码之后。利用编码矩阵，ECOC训练出了与列数相同数量的二分类器，因为待预测样本类别未知，所以样本被每一个二分类器依此预测之后得到一个结果向量（result vector）。结果向量依次与编码矩阵的每一个行向量进行比较，ECOC认为相似度较高的行向量对应的类别就是样本的预测类别。实际应用中，计算相似度的方法往往用计算汉明距离（hamming distance, HD）或者欧式距离（euclidean distance, ED）来代替，向量间距离小的，相似度高。比如在图2-3(a)中，一个待预测样本的结果向量是“[1,-1,-1,-1]”，它与*C*1,*C*2,*C*3,*C*4对应行向量的汉明距离分别是“0,2,2,2”。结果向量与*C*1对应行向量的距离最小，因此判定该带预测样本属于*C*1类别。汉明距离的计算方法在本章2.5.3详细介绍。

ECOC之所以称为“纠错”编码，是因为ECOC能够容忍一定数量的二分类器分类错误，编码的容错率与二分类器数量有关，正常情况下二分类器数量越多，容错率越强。例如图2-4的编码矩阵比图2-3(a)的OVA多了三个二分类器，不同类别所对应的编码行之间的汉明距离增大，*C*1与*C*3之间的汉明距离增大到4。现假设存在某个属于*C*3的数据点，所有的二分类器都预测正确的理想情况下，被预测得到结果向量为“[-1,-1,+1,-1,-1,+1,+1]”。但如果第3个分类器*d*3预测成错误结果“-1”，结果向量变成“[-1,-1,-1,-1,-1,+1,+1]”。解码该结果向量得到与4个类别的之间的汉明距离为“[3,3,1,3]”，解码距离最小的类别仍然是正确类别*C*3。这就是“纠错”的意思。根据上述示例可以看出ECOC编码矩阵能够容忍小部分二分类器发生错误，只要发生错误的二分类器数目小于类别间相互距离的一半，ECOC仍然能正确分类。



**图2-4：OVA增长矩阵**

于是设计ECOC矩阵的原则也就显而易见了：尽量使得任意两个类别的行编码之间的距离的最小值达到最大。不过，给定一个矩阵长度L，我们也并不能通过暴力枚举所有码长度为L的矩阵选出最好的（最小距离最大的那个）码表，因为这是个NP问题，稍微大一点的L就无法计算。而且，通常也没有必要得到最好的那个编码矩阵，其实也并不是编码矩阵本身的性质越好，最终的分类问题就处理得越好，其中还牵涉到很多其他因素，比如，不同类别之间的区分度不一样，不同的编码列所得到的二分类问题的难度也会因此而相差很大。一个本身性质很好，但是对应的二分类问题都很难（于是就很容易分错）的编码矩阵，和另一个性质稍差一些，但是对应的二分类问题都相对简单的编码矩阵，谁的效果更好也是不好说的。

### 2.4.2 二元编码和三元编码

根据编码取值不同，ECOC编码可以二元编码（Binary Code）和三元编码（Ternary Code）。二元编码取值{+1,-1 }，三元编码包含取值{+1,-1 ,0}。 在训练二分类器的时候，“0”对应的类别的样本不参与训练过程。

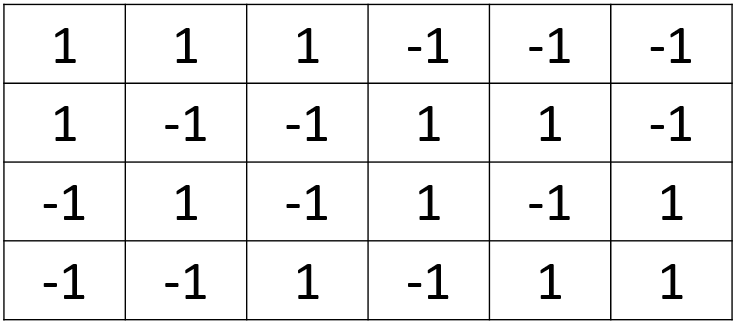
经典的ECOC编码一对多（one vs all, OVA）和一对一（one vs one, OVO）编码分别属于二元和三元编码。图2-3(a)和(b)分别是类别数量*Nc*为4的OVA和OVO编码。OVA每列定义某一类为正类，此类以外所有类为负类，因此OVA编码矩阵尺寸是。OVO每列从所有类中选择两类进行比较，定义一类为正类另一类为负类，因此OVO编码矩阵尺寸为。

### 2.4.3 固定编码

从2.4.2节可以看出，OVA与OVO矩阵的形式仅仅与训练数据的类别数量有关系，而与数据其他方面的特点没有关系。OVA与OVO编码矩阵的大小也相对于*NC*固定这类ECOC编码被称为固定编码（fixed coding）。

在很多数据集上，由于数据收集能力的限制或者数据本身特点的原因，数据集样本集中属于部分类别，且多数类样本数远多于少数类样本数。这种现象被称为“数据不均衡” （data unbalance）。在一些存在数据不均衡现象的数据集上面，简单的编码矩阵（如OVA与OVO）显出较差的效果。这两种简单的编码算法每一列都以某一个类为正类，如果一个少数类正好被作为唯一的正类去训练分类模型，那么这个分类模型会存在严重的分类倾向：倾向于把样本预测为多数类。这种结果属于训练数据过少引发的“训练不足”问题，训练不足会严重影响预测的准确性。

为了解决这类问题，研究者对OVA做了改进：采用二对多（2 vs all, 2VA）如图2-5，或者三对多（3 vs all, 3VA,）的编码矩阵等。这类矩阵也是固定矩阵，且拥有更长的长度，适合在数据类别数量适中的时候应用。更重要的是，这类矩阵可以一定程度上解决数据不均衡的问题：将某少数类和某多数类放在一起从而避免训练不足的问题。



**图2-5: 2-Vs-All编码矩阵**

### 2.4.4随机编码

另一些经典的编码方式是随机编码（random coding），包括稠密随机编码（dense random）和稀疏随机编码（sparse random），他们分属于二元编码和三元编码。这类方法随机产生编码列，如果产生了不合法的编码就删除重新生成，直到编码列数目达到预设数目为止。不合法编码如下：

|  |
| --- |
| 1. 编码中存在相同的两列； 2. 编码中存在互逆的两列（互逆即两列相加为零向量）； 3. 编码中存在相同的两行； 4. 编码中存在全0的行。 |

一个数据集的类别数目固定*Nc*，可能的编码组合数量有限。如果二分类器所用的算法都相同且没有额外随机性的话，那么重复的编码列是没有意义的。根据上述编码不合法规则，*Nc*类数据稠密随机编码数量上限是，而稀疏随机编码数目上限是。相关研究[6]建议二元随机编码长度，三元随机编码长度是。

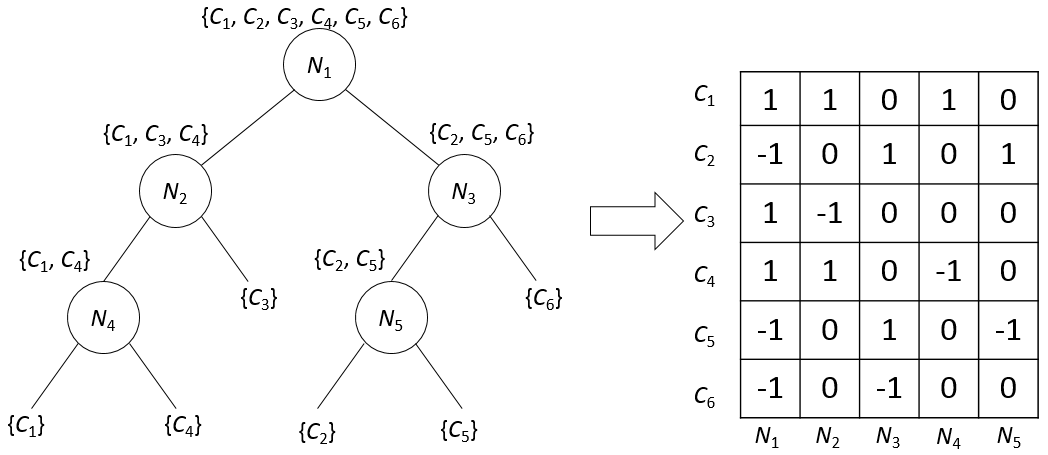
随机编码的时候，设定三种编码的概率之和是1，不过 稀疏随机编码比稠密随机编码更容易得到不合法的编码列，尤其是在*Nc*比较小的时候，非常容易出现没有-1或者没有+1的列。一个编码技巧就是在生成每一个编码列的时候，首先随机选定两个位置，赋值为+1和-1（或-1和+1），剩下的位置使用正常的方法采样0、+1、-1进行赋值，这样就能保证每一列总是至少同时有+1和-1出现了。

### 2.4.5 数据依赖编码

2.4.2节和2.4.3节介绍的经典编码不依赖于数据的内容生成，所以又被称为非数据依赖编码（Data-independent Coding）。他们可以应用于简单的多分类问题，但较难应用于较难被分类的数据，比如基因微阵列数据[45]、人脸图像数据[46]等。

ECOC编码研究的重点是根据数据特点设计编码，也就是数据依赖编码（Data-dependent Coding）。这些编码算法研究训练数据的分布特点或者相互关联设计编码列。也就是说，数据依赖编码编码算法对不同的数据设计的编码矩阵是不同的。

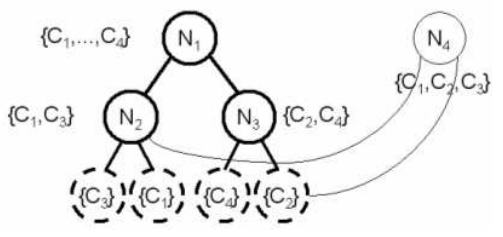
**判别式纠错输出编码**（Discriminant ECOC, DECOC）[13]是一种数据依赖编码，它试图最大化二分类器对应问题中两类数据的可区分性。DECOC把所有类别放到根节点，然后利用互信息（MI, Mutual Information）把类别集合划分为两个类别子集合。子集合再划分成两个子集合，直到每个子集合里只存在一个类别为止。由此构成的二叉树中，每个结点都代表一个类别划分，反映到编码矩阵上，每个结点代表一个编码列。DECOC编码列数比类别数少1，矩阵大小是。图(2-5)展示的是DECOC编码。



**图2-6：DECOC编码图例**

**森林纠错输出编码**（Forest-ECOC）构建多个DECOC的二叉树。这种编码方式拓展了二分类器的多样性，因为其包含随机产生的多余的二分类器。

**ECOC-ONE**（Optimal Node Embedded）[15]是一种基于DECOC矩阵，迭代添加新编码列的方法。由于DECOC矩阵产生的编码列太少、纠错能力不强，所以研究者设计一种迭代方式，每次验证并依据验证结果拓展编码矩阵，推荐编码长度是2*NC*。图2-6是ECOC-ONE拓展矩阵的示意图。



**图2-7：ECOC-ONE编码图例**

**子空间纠错编码**（subspace ECOC）[17]提出一种三维的ECOC矩阵，如图2-7所示三维包括{类别，二分类器，特征子空间}。从图中可以看出，不同的二分类器对应不同的特征子空间。不同列特征子空间（或者叫特征子集）是随机产生的。这种方法给ECOC研究者一种启发，不同二分类器对应不同的特征子空间也让所有的二分类器更加独立。这种给编码配置特征子集的方法，可以极大的丰富编码数量，本文第四、五章介绍的算法均基于这种方法。



**图2-8: Subspace ECOC图例**

## 2.5解码方法介绍

解码也是ECOC框架里面至关重要的一环。解码器把结果向量与编码矩阵中每一个代表类别的编码行进行比较。结果向量与哪一个编码行相似就指定该编码行对应的类别为预测结果。自ECOC诞生以来，相关研究的重点在于编码方法的研究，相比之下解码方法研究比较少。

本节包含公式的规则：分别表示预测结果和行编码，角标*i, j*分别表示矩阵的第*i*行，第*j*列，*n*表示矩阵的列数。

### 2.5.1 二元解码和三元解码

解码可以分为二元解码（Binary Decode）[47]和三元解码（Ternary Decode）[47]分别处理二元编码和三元编码的解码问题。经典的二元解码有汉明距离解码和欧式距离解码；经典的三元解码有弱欧式解码。

### 2.5.2 硬解码和软解码

从结果向量数值范围来看，解码可以分为硬解码（Hard Decode）和软解码（Soft Decode）。硬解码应用于结果向量取值为离散值{+1,-1}的情况，结果向量如果是[-1,+1]范围内的连续值可以采用软解码。汉明距离解码只能应用于硬解码，而其他解码方式可以应用于软解码。从科学研究的角度而言，软解码更具备研究价值。

### 2.5.3 常用解码方法

**汉明距离解码**（hamming distance, HD）[47]是一种简单经典的解码方式。在比较结果向量*x*与编码行*mi*对应位置上数值不相等的个数即是汉明距离。如公式2-9。

(2-9)

**欧式距离解码**（euclidean distance, ED）[47]是结果向量*x*与编码行*mi*的欧式距离，如公式2-10。

(2-10)

上述两种解码方式不适用于三元编码，因为解码过程中，ED和HD都没有忽略三元编码中的“0”。

**弱欧式解码**（attenuated euclidean distance, AED）[47]是一种三元解码，用来改进ED。AED在ED的基础上乘以了和的绝对值。因为中为0的类别不参与训练，所以相应类别也不应该参与解码，乘以是为了在为0的时候不参与欧氏距离的计算。乘以的绝对值是为了增强*x*绝对值对距离的影响，从而弱化对整个距离的影响。如公式2-11。

(2-11)

## 2.6 改善弱欧式解码

在ECOC研究领域，不同解码方法对分类效果的影响差异相对较小，所以解码并不是ECOC主要研究方向，这导致研究者往往忽略解码方法的一些不足。本章将弱欧式解码的弊病呈现出来，讨论并提出解决办法。

### 2.6.1 弱欧式解码的弊端

欧式解码不适合应用于三元编码的解码，因为ED没有忽略编码行中“0”对应的分类器的分类结果。为了解决这个问题，Escalera[47]将弱欧式解码应用于三元编码。弱欧式解码是一种三元解码，用来改进ED，如公式(2-11)所示。这种解码方式存在两个弊端：运算量较大、“多零倾向”问题。“多零倾向”是本文定义的名词，指的是AED倾向于给含“0”多的码字较小的距离。

### 2.6.2 优化计算量

优化的弱欧式解码（EuD）可以减少弱欧式解码计算量。解码公式如(3-1)，分别表示预测结果和行编码，角标*i, j*分别表示矩阵的第*i*行，第*j*列，*n*表示矩阵的列数。EuD的计算量比AED少用了的运算量。在实际运算的时候，NS可能会很大，所以在样本数非常多的情况下，使用EuD解码可以减少运算。

(3-1)

弱欧式解码需要计算距离差的平方和，在计算大量数据时，欧式解码会花费较多时间。原欧式解码计算量如公式(3-2)，优化的弱欧式解码将欧式距离的平方计算变成了矩阵的乘法运算，其计算量如公式(3-3)

(3-2)

(3-3)

其中*NS、NC、Nd*分别表示样本（sample）个数、类别（class）个数、分类器（dichotomy）个数，*P、M、A、S*分别表示一次加减法（plus）运算、一次乘法（multiply）运算、一次绝对值（absolute）运算、一次开方（square root）运算。

### 2.6.3 RED解码解决“多零倾向”

弱欧式解码是最常被使用的三元解码方式，但它倾向于多零的码字（编码行）。比如结果向量是[0.2,0.2,0.6,-0.2,-0.1,-0.6]，而矩阵是[[1,1,1,-1,-1,-1],[1,0,0,0,0,0]]。结果向量与第1条码字的距离1.746，是6个非零数和的开方；与第2条码字的距离0.8，仅是1个非零数和的开方。解码结果判定该结果向量属于第2条码字对应的类。得到这样的解码结果并不是因为结果向量与第2条码字更相似，而是因为第2条码字包含的“0”更多。码字“0”过多引发的解码不准确。

针对AED的“多零倾向”，我们提出了一种比例欧氏距离解码（RED, Ratio Euclidean Decode）。利用削减上述“多零倾向”。

(3-4)

上文例子被RED解码之后的距离分别是[0.713,0.8]。解码结果属于第一类。

其实在应用解码方法解决实际问题的时候，AED的弊端较少情况下才会体现出来，因此AED方法被应用最广的三元解码方法。

## 2.7 本章小结

本章介绍本文涉及的相关理论。

2.1节介绍了机器学习中监督学习概念，说明监督学习中分类任务的概念，分类任务就是预测未知样本所属的类别。

2.2节介绍集成学习的概念，阐述了被集成的单分类器既需要够大的差异性，也需要足够高的准确率。

2.3节介绍特征选择的目的和效果。并根据主要思想把特征选择方法分成三类：Filter方法、Wrapper方法、Embedded方法。接着详细介绍了本文第四、五章使用的7个特征选择方法：F-value、互信息、BSS/WS、方差、卡方统计、L1正则化、Extra Tree Classifier。

2.4节详细介绍了ECOC的原理和工作过程，ECOC分为编码和解码。编码有固定编码、随机编码、数据依赖编码。前二者编码矩阵与数据没有关系，只与数据类别个数有关系，这样的编码有OVA、OVO、稠密随机编码和稀疏随机编码。而数据依赖编码算法考虑数据特点，编码矩阵与数据有直接关系，这样的编码方法包括DECOC、Forest-ECOC和ECOC-ONE等。

2.5节介绍的是解码方法。从编码取值出发，解码分为三元解码和二元解码。三元解码是在解码过程忽略“0”编码，而二元解码没有忽略。从结果矩阵数值角度出发，解码分为硬解码和软解码。硬解码对应的结果是二值离散的，软解码对应的结果是连续值。本节依此介绍3个常用解码方法和他们的适用范围，汉明解码是二元硬解码。欧式距离是二元解码，可以应用于软解码。弱化欧氏距离可以应用于三元解码和软解码。

2.6节对弱欧式编码的两个弊端，提出解决方法。首先介绍弱欧式编码的两个弊病：解码运算量较大且存在“多零倾向”。然后提出一种减少解码运算量的方法，把解码中的平方和计算转变为矩阵运算。为了解决弱欧式解码的“多零倾向”，本节提出了比例欧式解码并对解码方法进行分析。

# 第三章、自调节编码ECOC算法

输出纠错编码（error correct output code, ECOC）被用于许多多类分类领域，包括人脸识别、交通标志识别等。2000年之后，研究者开始将ECOC用于分类微阵列(microarray)数据，希望借此区分疾病亚种。微阵列数据“多特征少样本”且特征高度重叠，分类较困难。而许多相关ECOC研究并没有充分考虑到微阵列数据特点。本章节提出针对微阵列数据的自调节纠错输出编码算法（Self-adjust error correct output code, SA-ECOC）,算法针对数据特点设计编码矩阵，并实现自动拓展编码矩阵，以期达到提高微阵列数据分类准确的目的。

本章3.1节介绍算法，包括算法思路、算法相关数据测度，然后分两部分详细介绍算法细节。3.2节介绍算法实验情况，使用癌症微阵列数据，分析了数据特征子集覆盖情况、矩阵长度、迭代过程准确率变化情况。

## 3.1 算法介绍

SA-ECOC算法简单分为两步骤：一是设计编码矩阵、二是自动拓展矩阵。

第一步是构造编码矩阵：利用多个特征选择方法，构建多个特征子集。并在不同子集上利用判别式纠错输出编码（DECOC）建树的方法（如第二章2.4.4所示的方法），配合与特征相关的数据复杂度，构建不同的编码矩阵。所有编码矩阵的列编码汇集成一个列池，利用留一验证法给列池评分，选择评分高的编码拼成一个编码矩阵。第二步是自动拓展矩阵：第一步构造的编码矩阵并不一定最优，算法利用一个循环验证的过程，保证编码矩阵整体的效果下，给编码矩阵逐步增加新的列编码。为了每次迭代增列后新编码矩阵效果良好，SA-ECOC提出一种新的筛选标准“距离间隔”，从新增列中删除部分列。当迭代过程达到一定的条件，终止迭代。

### 3.1.1 算法思路

出于对微阵列数据特点的考虑，提出SA-ECOC解决微阵列数据的分类问题。针对“多特征”，最普遍的应对思路是进行特征选择或特征提取来减少无用特征的干扰。本算法亦采用这种方式，利用多种不同特征选择方法构建多个不同的特征子集。针对“少样本”，常规方法是随机过采样，而考虑到微阵列数据样本太少，随机过采样导致分类结果波动大，且实验不具重复验证性。本算法两步中采用的验证方法都是留一验证法，此方法可以充分利用样本数据。微阵列数据特征重叠度很高，对此本算法在借鉴DECOC建树编码方法，而利用数据复杂度为二叉树分支，因为数据复杂度在[42]被验证对微阵列数据处理效果较好。

根据集成学习理论，ECOC要取得好的效果需要被集成的分类器之间具备一定的差异，同时各个分类器分类效果不差。算法第一步采用多个不同特征子集编码是利用特征子集的差异带来分类器的差异；而通过验证评分选择列编码使被选中的分类器具备一定的分类效果。此外，ECOC理论认为编码矩阵需要一定数据的列，这样分类器的纠错能力才得以保证。所以，算法第二步采用验证自增编码的方式，适当增加编码长度。

### 3.1.2相关内容：数据复杂度

SA-ECOC第一步中有借鉴DECOC建立二叉树的过程，在原DECOC方法中，分支标准使用的是互信息（mutual information）,本算法里使用的是数据复杂度。

数据复杂度（data complexity[48]是研究数据分布对模式识别影响的一个领域，它针对二类数据，评估的是这两个类数据分类的难易度。不同类别的数据复杂度有不同的视角。在特征工程中，特征的重要性是一个重要的课题，所以提出了一些与特征相关的数据复杂度：F1、F2、F3、N2、N3。前三个复杂度从数据重叠度出发，描述了类之间的特征数值重叠情况，高重叠度的两个类之间有较高的数据复杂度。后两个复杂度从样本点空间分布的角度出发，描述样本点在空间中的可分性，可分性大的两个类之间有较低的数据复杂度。以下是每种复杂度的详细介绍。

F1计算每个特征中两个类之间的重叠度，分别是两个类在第i个特征上的均值和方差，取值范围是[0,+∞)。 F1值小说明重叠度高。F1定义如 公式(3-2)：

(3-1)

(3-2)

F2衡量的是重叠区域占比。假设是第i个特征在第j个类中的最大值和最小值，F2值的计算方法如公式(3-3)。F2值小表示小的重叠量。

(3-3)

F3的定义是样本在非重叠区域中的比例，如公式(3-4)所示。两个类的重叠可能导致每个特征维度中的误分类。重叠区域的样本越多，分离两个类的难度就越大。F3的范围是[0,1]，较小的值表示较高的重叠。

(3-4)

N1是类边界上样本点所占的比例。在样本空间构建一棵最小生成树（Minimum Spanning Tree），与不同类样本直接相连的样本点就是边界上的样本点。这样的样本点占所有样本点数量的比值就是N1数据复杂度的值。

N2是类内距离与类间距离的比值，如公式(3-5)所示。首先计算所有点的最近邻距离，所有类内最近距离的总值除以所有类间最近距离的总值就是N2的值。

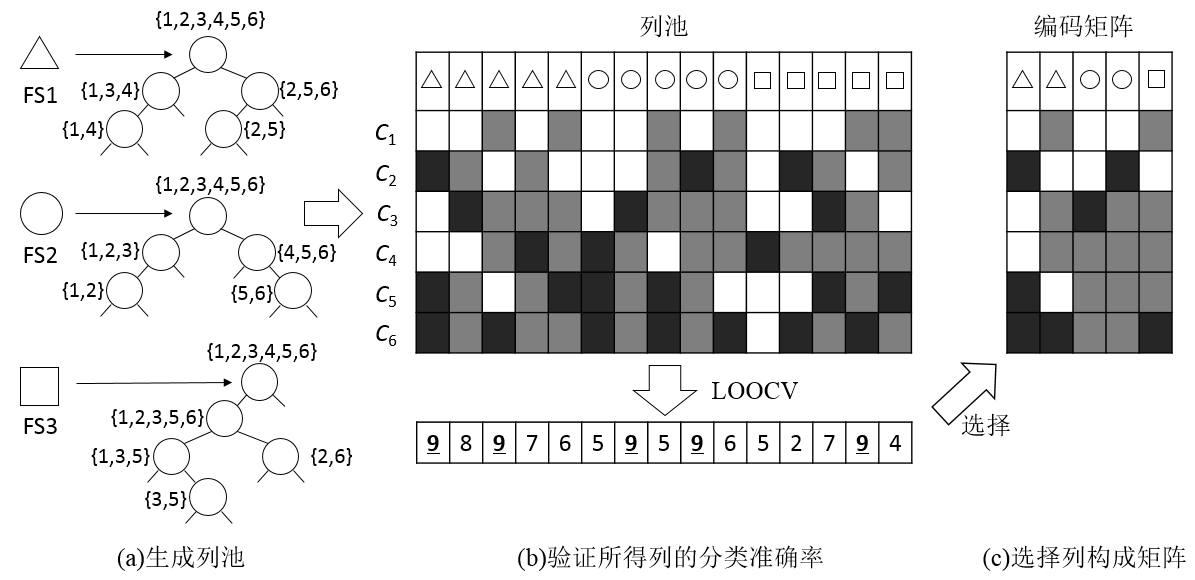
(3-5)

N3是最近邻分类器分类的错误率。基于一个最近邻分类器，用留一法验证计算错误率。N3度量在区间[0, 1]的范围。这个度量的低值表明类边界有很大的间隙。

### 3.1.3算法第一步：构造编码矩阵

该部分分为3个步骤。在步骤1中，利用*Nf*个不同的特征选择方法来生成*Nf*个特征子集。在每个特征子集，模仿DECOC建二叉树的过程（如图3-1(a)）产生*NC*-1个列（*NC*是类的数目），二叉树分支标准是数据复杂度。收集所有列（编码）产生一个列池，池中包含个列。每一列都与一个特征子集对应，后面分类器的训练和预测是基于该列所对应的特征子集。步骤2使用留一验证法（leave one out cross-validation, LOOCV）单独验证每一列的准确率，作为列的评分。步骤3选择评分较高列构造编码矩阵***M***。

图3-1是含有6个类别的数据构造编码矩阵的3个步骤，黑表示-1，白表示+1，灰表示0。图3-1(a) 三角形、圆形和方形表示三个特征子集，三棵树分别表示对应子集生成的二叉树。集合所有列编码形成的列池中，每个列都对应着特征子集。



(a)建树 (b)列池评分 (c)选列

**图3-1构造编码矩阵**

算法3-1展示的是模仿DECOC建二叉树的过程，此算法与DECOC中建树算法除了划分标准从互信息换为数据复杂度之外，多处细节不同。算法1第1步寻找“最优划分”的方法是顺序前向浮动搜索算法（sequential forward floating search, SFFS），如算法3-2所示。

|  |
| --- |
| 算法3-1建树算法 |
| 初始： 类别标签集合  第1步：对集合的最优划分  是数据复杂度测度，是训练数据集，是标签二分方法  第2步：  如果，则令，然后重复第1步  如果，则直接划分  注：算法第1步中找“最优划分”的方法是SFFS |

SFFS算法并非本文原创，它是一种搜索算法，目的是从输入的集合中找到一个子集合，使评价标准 最优。本章中最优指的是数据复杂度最小。值得说一句的是，SFFS属于贪心搜索算法，其输出结果是局部最优并非全局最优，当类别数量多的时候使用SFFS可以节省运算时间。

|  |
| --- |
| 算法3-2 顺序前向浮动搜索算法（SFFS） |
| 输入：  输出：  初始：  终止：当搜索标准不再变化时终止  第1步：（增添）  .  .  第2步：（删减）  .  如果    跳到第2步  否则  跳到第1步 |

### 3.1.4 算法第二步：自动拓展矩阵

延长编码矩阵是一个迭代过程，每次迭代往***M***后面接续一些新列。图3-2展示一次迭代，图中矩阵黑色表示“-1”，白色表示“+1”，灰色表示“0”。在每次迭代中，先留一法验证找到错误率最高的类*CK*，然后找到所有与*CK*产生混淆的类，让它们与*CK*建立一对多（one vs all, OVA）和一对一（one vs one, OVO）的列。生成列的方法如公式(3-6)所示，例如图2(a)和(b)，根据混淆矩阵图2(a)，*C*2是该迭代中错误率最高的类，它与*C*1、*C*5和*C*6发生了混淆，图2(b) 第1列为OVA列（{2}vs{1,5,6}），2,3,4列为OVO列（{2}vs{1}，{2}vs{5}，{2}vs{6}）。

(3-6)

上式和分别表示正类序号集合和负类序号集合，是产生的列，*Nc*是类别数目。

因为编码矩阵各列对应着特征子集，所以新生成OVO和OVA列也需要对应的特征子集（如图3-2(c)），对应特征子集的方法如下：

|  |
| --- |
| 1. 对应特征子集后的列在原矩阵中不存在 2. 选择使该列数据复杂度最小的特征子集 |

对应好特征子集的OVO和OVA列加入原编码矩阵（如图3-2(d)）；此时增加级几列之后的矩阵并不一定比原矩阵好，所以依此时编码矩阵做一次留一验证，依据验证结果删除OVO和OVA列中部分列，如图3-2(e) 和算法3-3所示。简单来说，删除算法是从结果矩阵*Y*一条条删减新列（OVO和OVA列），直到结果矩阵*Y*对应的“距离间隔”最大的算法。

“距离间隔”（distance Interval, DI）是本算法定义的衡量标准，用来剔除列，如图3-2(e)。根据原理，ECOC分类的最终结果取决于二分类器输出结果和码字之间的距离。定义一个样本的二分类器输出结果到正确类的距离是正确距离（DC）；到错误类的距离是错误距离（DE）。*di*被定义为DC 和DE中最小值的差值，如公式(3-7)所示。因为解码方式是将距离最近的类作为该样本的预测类，当样本被错误分类时，*di*是一个正数，样本被正确分类时，*di*是一个负数。DI被定义为通过计算所有验证集样本的*di*之和，如公式(3-9)所示。

(3-7)

(3-8)

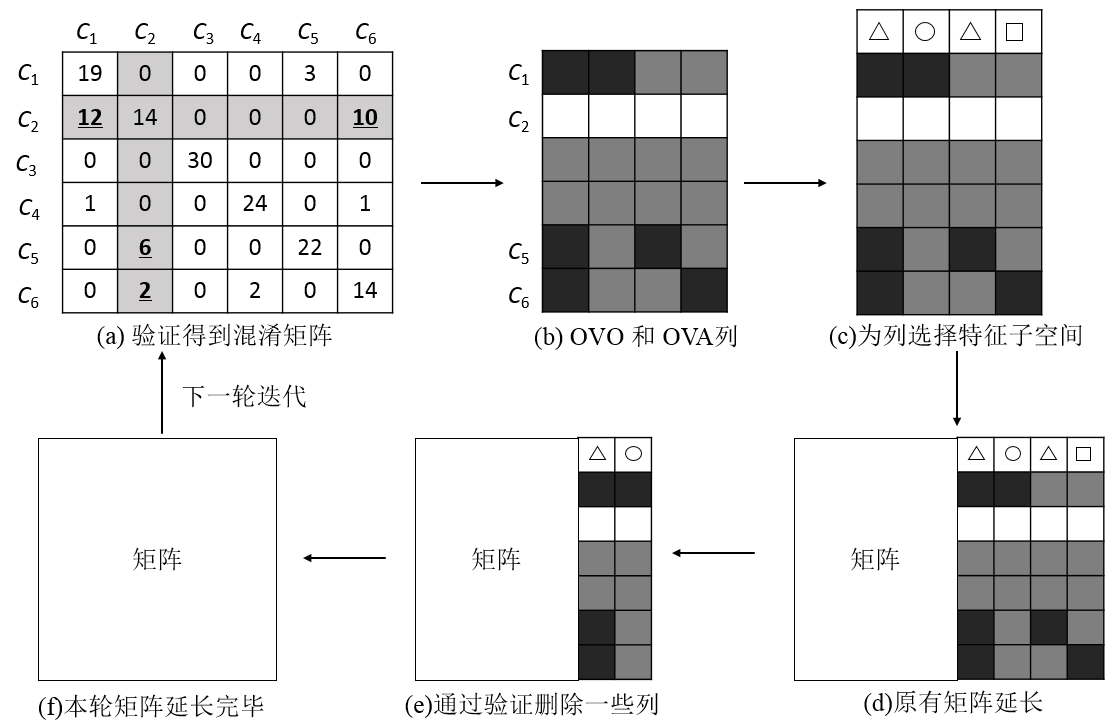
(3-9)

上式中***D***表示距离矩阵，*n*表示误分类样本的个数，*d*是样本输出结果与所有码字的距离。编码矩阵***M***（尺寸：*Nc*×*Nd*）；结果矩阵***Y***（尺寸：*n*×*Nd*）是一个包含每个样本对应每个二分类器交叉验证的预测结果；距离矩阵**D**（尺寸：*n*×*Nc*）包含每个样本到每个类的距离。

算法3-3展示的是利用距离间隔从新列（OVO和OVA列）中删除列的方法。改方法主要为了验证新列个体对矩阵整体的影响，删除有负面影响的列。

|  |
| --- |
| 算法3-3删除列算法 |
| 初始： 结果矩阵和对应距离间隔  第1步：找一列，对应最小的距离间隔  第2步：如果，则; 重复第1步  如果，则停止算法 |

完成算法3之后，整个算法的第二步完成了一次迭代。图3-2描述了自动拓展矩阵过程，图中用颜色表示编码，黑表示-1，白表示+1，灰表示0。图只显示一次迭代，只有当迭代次数达到列的最大大小或矩阵的验证准确率为100%时，算法才停止。SA-ECOC通过两个自调整步骤选择富含信息的列，由此可以得到更好的分类结果。



**图3-2自动拓展矩阵**

总结算法的两个步骤。第一步在不同特征子集上生成不同的列编码，每个列编码都有对应的特征子集。（这种做法减少了特征数量，简化了计算；又使特征不被过量丢弃；同时还丰富了列编码的多样性。）而后所有列编码组成列池，在列池中选择验证效果好的列编码组成编码矩阵。（这种做法选择了分类效果优秀的列，使单个二分类器的分类效果不会很差。）第二部在编码矩阵的基础上，迭代延长矩阵，在适当的条件下停止延长（这种做法延长的矩阵的长度，而又不过度延长矩阵）。

## 3.2实验介绍

为了验证算法的性能，本研究实验中选取了四个数据集，表3-1列出了相关数据集的细节。本研究选择了五个特征选择BSS\WSS[42]，ANOVA F-value，mutual information，L1/Lasso，extra tree[44]，每种特征选择方法选择约100个特征。二分类器选择了高斯朴素贝叶斯（Gaussian naive bayes, GNB）和决策树（decision tree, DT），这两个分类模型提供自scikit-learn[38]。文中展示了SA-ECOC组合上述两种二分类器的迭代延长过程。然后与一些经典的ECOC方法组合上述两种二分类器的分类结果进行了比较。经典ECOC方法包括discriminant ECOC、Forest ECOC，OVA，OVO，以上ECOC方法由[36]提供。在所有二分类器中，都采用默认参数。解码方式采用弱欧式解码(attenuated euclidean decoding, AED)。

本节分为五个小节，分别介绍数据集、特征子集之间重叠程度、设计得到编码矩阵的长度、迭代增列时准确率变化情况、和最终分类准确率。

### 3.2.1数据集介绍

本研究选择8个癌症基因微阵列数据集，并列出了每个数据集的样本数、类别数、特征数。同时给8个数据集编号，下文中展示实验结果时用数据集编号代替数据集名称。Breast为乳腺癌数据集；DLBCL（diffuse large B-cell lymphoma）为B细胞淋巴癌数据集；Cancers数据集来源于Cancer Research，包括卵巢癌、前列腺癌、胰腺癌等各类癌症数据；Leukemia1和Leukemia2为血癌数据集；Lung1和Lung2为肺癌数据集；SRBCT（small round blue cell tumors）为小圆蓝细胞肿瘤数据集。以上数据每条数据都代表一名癌症病患的基因微阵列数据，每个数据集都包含多于两类的癌症亚种。

**表3-1 数据集 单位：个**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | 数据集名 | 样本数（训练集/测试集） | 类别数 | 特征数 | 引用 |
| 1 | Breast | 84(54/30) | 5 | 9216 | [49] |
| 2 | DLBCL | 88(58/30) | 6 | 4026 | [50] |
| 3 | Cancers | 160 (86/74) | 9 | 12533 | [51] |
| 4 | Leukemia1 | 72(38/34) | 3 | 7129 | [52] |
| 5 | Leukemia2 | 72(57/15) | 3 | 12582 | [52] |
| 6 | Lung1 | 96(64/32) | 3 | 7129 | [53] |
| 7 | Lung2 | 203(136/67) | 5 | 12600 | [53] |
| 8 | SRBCT | 83(63/20) | 5 | 2308 | [54] |

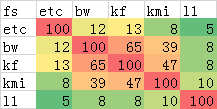
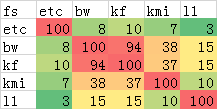
### 3.2.2特征覆盖度

针对基因表达数据特征数多的特点，本实验中SA-ECOC选取了5个特征子集合，特征子集代号与特征选择方法对应关系如表3-2所示。两两特征子集之间存在共有的特征，或者叫覆盖特征。不同数据集覆盖情况记录如图3-3，这种表展示的是覆盖特征百分比（为了简化实验，每个特征子集均包含100个特征），是对角线为100的斜对称矩阵。

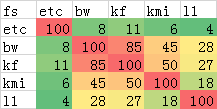
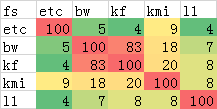
**表3-2特征子集对应特征选择方法**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | etc | bw | kf | kmi | l1 |
| 选择方法 | 公式(2-8) | 公式(2-5) | 公式(2-3) | 公式(2-4) | 2.3.2 L1正则化 |

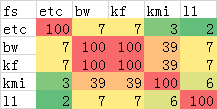
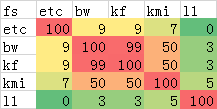
如图3-3所示，不同数据集特征覆盖情况相似：比如bw与kf覆盖度很高，kmi与bw或kf的覆盖度较高，l1和etc覆盖度较低等。这是因为bw与kf在选择特征的方式很相近，本质都是特征数值的平方和的比值。而etc是基于概率的基尼指数选择特征，l1是利用L1范数得到的特征，两者本质存在差异等原因。之所以选择相似和很不相似的特征子集，是希望保留特征子集覆盖梯度，展示算法在这种情况下表现效果。



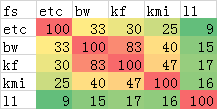
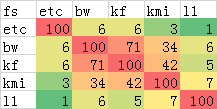
**(a) Breast (b) DLBCL**



**(c) Cancers (d) Leukemia1**



**(e) Leukemia2 (f) Lung1**



**(g) Lung2 (g) SRBCT**

**图3-3 各数据集上特征覆盖百分比**

图3-3(f)和(g)中，bw与kf特征覆盖度达到了99%和100%，主要是因为这两个数据集被bw和kf认定为“好”的特征恰巧一样。如果将选择特征的数量减少，高覆盖度的情况会减轻。

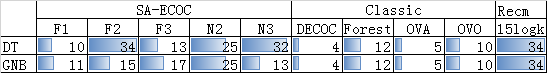
### 3.2.3矩阵长度

ECOC具备“纠错”能力，在于ECOC集成许多二分类器，在这些二分类器分类效果普遍较好的情况下，一个二分类器分类错误会被其他二分类器“纠错”。对于一个集成系统来说，被集成的优秀的二分类器数量越多，最终分类效果越好。然而，据Crammer 和 Singer[5]证明ECOC编码矩阵是个NP问题，编码矩阵长度不能过长，否则在编码过程或者分类过程中，计算任务会过重。对于分类microarray数据而言，由于数据量稀少，计算任务并不重，因此可以适当增加二分类器的个数，也意味着增加编码矩阵的长度。

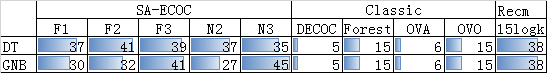
表3-2至表3-9展示了八个数据集上SA-ECOC编码算法得到矩阵的长度。表中第2~6列F1、F2、F3、N2、N3表示使用SA-ECOC分别配合上述5种数据复杂度得到的编码矩阵；第7~10列表示使用经典编码算法得到矩阵的列数；最后一列表示Allwein[6]建议的三元编码长度15log*k*（*k*是类别个数）。表第3、4行表示分别使用GNB和DT两种基分类器得到的矩阵。因为SA-ECOC是一种基于验证的编码算法，编码矩阵与参与验证的基分类器有关系，所以同一个算法不同的分类器得到矩阵不一样。而后4列编码算法不基于验证，故长度相同。

从表中可以看出，前5列SA-ECOC普遍比后4列经典编码算法长，更接近建议长度。除此以外SA-ECOC会依据基分类器的不同而设计与分类器相适应的编码矩阵，不同基分类器对应的算法有不同的矩阵形式。总结以上，SA-ECOC根据验证结果设计矩阵，编得矩阵长度较合理。

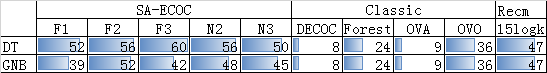
**表3-2: Breast数据集上矩阵长度对比** **单位：个**



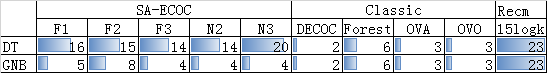
**表3-3: DLBCL数据集上矩阵长度对比 单位：个**



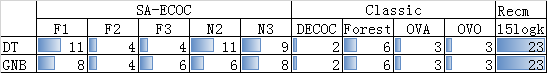
**表3-4: Cancers数据集上矩阵长度对比 单位：个**



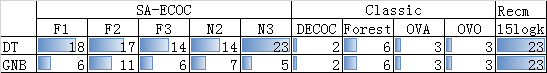
**表3-5: Leukemia1数据集上矩阵长度对比 单位：个**



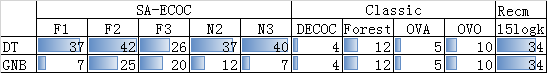
**表3-6: Leukemia2数据集上矩阵长度对比 单位：个**



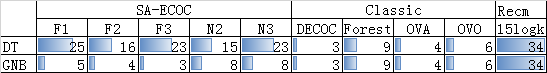
**表3-7: Lung1数据集上矩阵长度对比 单位：个**



**表3-8: Lung2数据集上矩阵长度对比 单位：个**



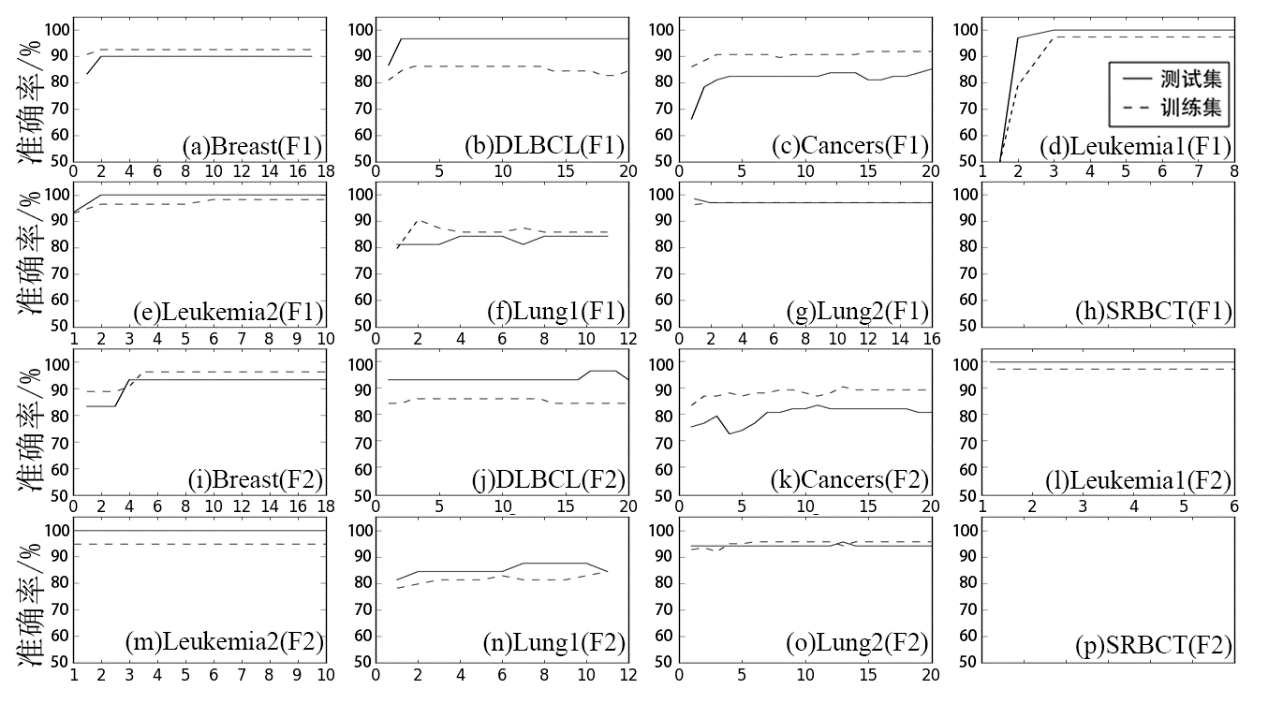
**表3-9: SRBCT数据集上矩阵长度对比 单位：个**



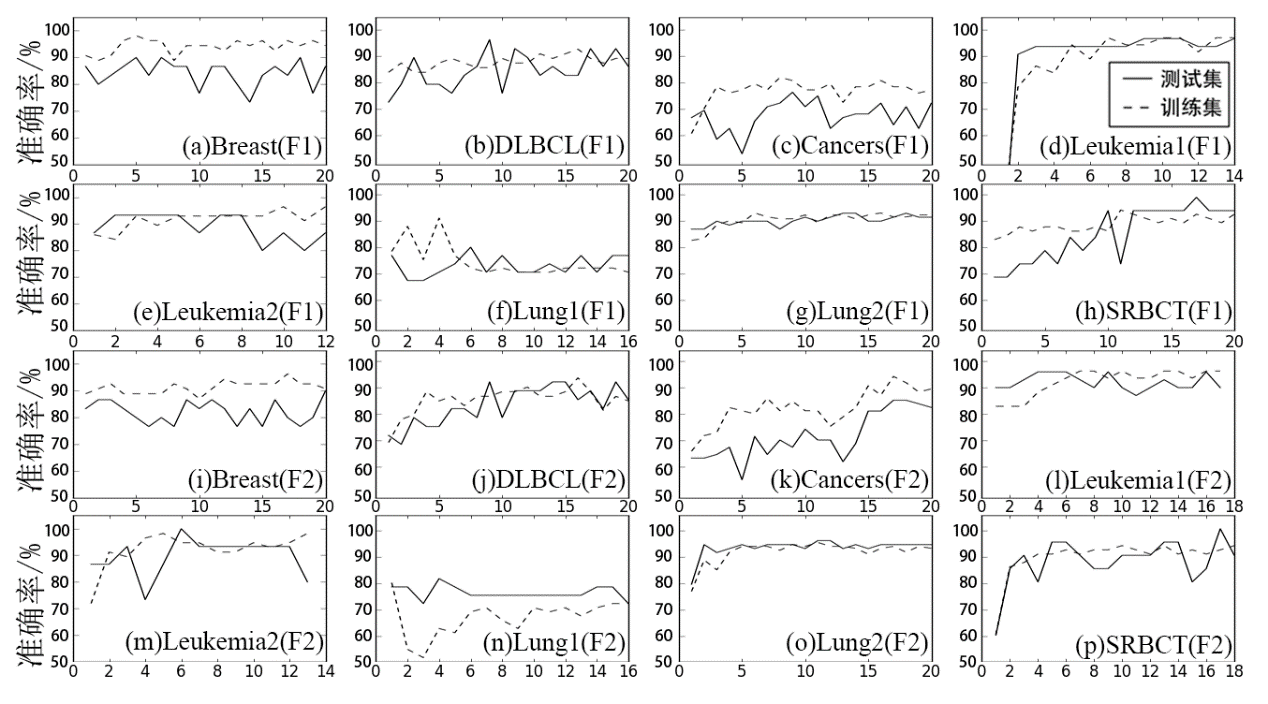
### 3.2.4迭代过程

本文中算法是一个迭代增加列编码的过程，实验迭代次数上限设置为20。为了展示迭代过程对分类训练集和测试集的影响，本文中记录了每轮迭代过后矩阵的变化，并计算每轮迭代之后，分类器分别在训练集和测试集的分类准确率。图3-4和3-5展示了数据复杂度选用F1和F2，基分类器选择GNB和DT的时候，迭代过程的准确率变化。*x*轴代表迭代次数，*y*轴表示准确率。实线/虚线表示测试/训练数据的准确率。图3-4(h)和3-4(p)是空白因为实验在迭代延长前已经达到100%的准确率从而停止迭代。

本算法三次用到了留一法（图3-1(b)、图3-2(a)和图3-2(e)）验证训练集，希望利用验证信息改善（延长）编码矩阵，目的是提高编码矩阵在测试集的分类准确率。从图3-4和图3-5可以看出大部分情况下，随着迭代次数增加，实线（测试集）与虚线（训练集）保持相似的趋势，说明算法对训练集和测试集的改变具有一致性。而且，多数情况下，虚线和实线大体呈现上升趋势，说明该算法能够通过逐步提高训练准确率来提高测试准确率，本算法具有意义。



**图3-4：数据复杂度选择F1和F2时，GNB基分类器准确率随迭代次数变化**



**图3-5：数据复杂度选择F1和F2时，DT为基分类器准确率随迭代次数变化**

把图3-4与图3-5出现的曲线分类，大致可以分成四种

“一”型曲线。准确率随着迭代次数的增加没有变化（如图3-4(l)和(m)等），或者准确率仅最初少许提升而后没有变化（如图3-4(a)和(b)等）。这类曲线多出现在用GNB做基分类器的情况下，其原因是在迭代初期，分类准确率便已达到较高的水平，但每次迭代仍有误分类样本，这些样本往往顽固而难以分类，每次新增的列被图2(e)过程删除干净，矩阵实际没有被延长。

“缓坡”型曲线。准确率平稳上升（如图3-5(g)和图3-5(k)等）。这类实验结果较为理想，符合算法预期的效果。

“陡坡”型曲线。准确率在迭代初期大幅提高，中后期稳定（如图3-4(d)和图3-5(d)等）。这类曲线主要因为迭代开始时，矩阵***M***的分类准确率很低，经过一两次迭代优化，准确率有了较好提升。之后分类准确率平稳原因与“一”型曲线出现原因相似。这类实验结果同样符合预期。

“波动”型曲线。准确率波动较大（如图3-5(a)和(f)），且往往伴有准确率降低的趋势。这类曲线多出现在用决策树做基分类器的情况下，决策树是一个敏感的分类器，在某些情况下，训练数据的微小变化会导致剧烈变化。这些实验测试准确率在迭代末期反而下降，而且训练准确率明显高于测试准确率。这是因为多次迭代没有及时停止，编码矩阵过于拟合训练数据。对于这种迭代次数过多而造成的过拟合，本研究需要更进一步研究迭代停止条件加以改进。

### 3.2.5最终分类结果与对比

表3-10和表3-11分别列出SA-ECOC算法分别配合5种数据复杂度（F1、F2、F3、N2和N3）得到的分类准确率与经典ECOC算法（DECOC，Forest，OVA，OVO）的对比结果。SA-ECOC集成5个特征子集得出一个结果，而经典ECOC算法分别对5个特征子集分类，生成5个不同的结果并取平均数±标准差展示于表3-10和表3-11中，方便与SA-ECOC结果比较。表的最后一行是每种方法结果的均值。（注：因为部分数据集在不同特征子集分类效果波动较大，会出现均值±标准差超过100的情况）。

**表3-10: 使用高斯朴素贝叶斯得到的分类准确率 单位：%**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | SA-ECOC | | | | | DECOC | Forest | OVA | OVO |
| F1 | F2 | F3 | N2 | N3 |
| 1 | 90.00 | 93.33 | **96.67** | **96.67** | **96.67** | 74.0±7.23 | 80.67±9.83 | 85.33±7.67 | 86.67±9.13 |
| 2 | 96.67 | 93.33 | **100.0** | 96.67 | 96.67 | 71.33±17.73 | 75.33±22.19 | 85.33±6.91 | 90.67±2.79 |
| 3 | **83.78** | 81.08 | **83.78** | **83.78** | 81.08 | 63.78±4.82 | 69.19±7.67 | 77.84±7.19 | 55.95±22.4 |
| 4 | **100.0** | **100.0** | 97.06 | 97.06 | 97.06 | 98.82±1.61 | 97.65±2.46 | 96.47±2.46 | 98.82±2.63 |
| 5 | **100.0** | **100.0** | **100.0** | **100.0** | **100.0** | 84.0±7.6 | 84.0±7.6 | 89.33±3.65 | 86.67±16.33 |
| 6 | **84.38** | **84.38** | **84.38** | **84.38** | 81.25 | 71.25±13.15 | 70.62±11.4 | 76.25±4.74 | 71.88±9.38 |
| 7 | **97.01** | 94.03 | 95.52 | **97.01** | 95.52 | 71.64±20.36 | 74.93±19.15 | 89.85±4.88 | 86.87±1.25 |
| 8 | 85.00 | 90.00 | 80.00 | 90.00 | **100.0** | 86.0±10.84 | 85.0±10.61 | 82.0±8.37 | 83.0±10.95 |
| A | 92.11 | 92.02 | 92.18 | 93.20 | **93.53** | 77.6±10.42 | 79.67±11.36 | 85.3±5.73 | 82.57±9.36 |

**表3-11: 使用DT得到的分类准确率 单位：%**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | SA-ECOC | | | | | DECOC | Forest | OVA | OVO |
| F1 | F2 | F3 | N2 | N3 |
| 1 | **86.67** | 80.00 | 80.00 | 83.33 | 83.33 | 64.67±9.31 | 68.67±10.43 | 68.67±6.5 | 64.67±5.58 |
| 2 | 86.67 | 86.67 | **93.33** | 86.67 | 83.33 | 70.67±15.17 | 80.0±7.82 | 71.33±3.8 | 69.33±12.11 |
| 3 | 82.43 | 78.38 | **83.78** | 77.03 | 75.68 | 43.78±14.35 | 48.65±10.98 | 55.68±4.72 | 54.86±9.77 |
| 4 | 91.18 | 91.18 | **97.06** | 91.18 | 88.24 | 85.29±6.58 | 88.82±3.22 | 90.0±1.61 | 87.06±6.77 |
| 5 | **93.33** | 66.67 | 80.00 | 80.00 | **93.33** | 84.0±5.96 | 81.33±7.3 | 85.33±5.58 | 86.67±6.67 |
| 6 | 75.00 | **78.13** | **78.13** | **78.13** | **78.13** | 68.75±2.21 | 66.25±4.64 | 72.5±4.64 | 69.38±6.01 |
| 7 | 92.54 | 94.03 | 94.03 | 92.54 | **95.52** | 85.07±5.48 | 88.66±4.17 | 87.76±2.87 | 85.97±3.44 |
| 8 | **95.00** | **95.00** | 80.00 | **95.00** | 90.00 | 75.0±10.0 | 73.0±17.54 | 85.0±5.0 | 74.0±7.42 |
| A | **87.85** | 83.76 | 85.79 | 85.49 | 85.95 | 72.15±8.63 | 74.42±8.26 | 77.03±4.34 | 73.99±7.22 |

与经典ECOC算法得到的结果相比较，SA-ECOC在所有数据集中都较大幅度高过经典方法结果的均值，表现出稳定和最高的性能。基于这5种数据复杂度的算法在不同数据集各具优势，但综合所有数据集，发现基于N3和F1对应算法在8个数据集上的平均得分最高。而且基于5种数据复杂度的SA-ECOC算法效果多数高于经典ECOC的平均分数。总的来说，可以认为SA-ECOC性能稍微好一些。表3-10与表3-11相比，SA-ECOC在使用GNB作为基分类器时，结果比用DT做基分类器的效果更好。

表3-10的实验结果表明，实验结果与数据本身有重要关系。比如表3-10第4和5个数据集（Leukemia1和Leukemia2），SA-ECOC的分类准确率非常高。这主要因为数据本身仅含3个类，较容易分类。而且因为类别数量稀少，经典ECOC的编码矩阵长度比经过延长的SA-ECOC的编码矩阵短很多，这意味着经典ECOC的基分类器数量很少，分类能力比较差。

表3-11第5个数据集（Leukemia2）准确率仅66.67%，与表3-10第5个数据集（Leukemia2）的高准确率出现巨大差异。分析其原因是用DT作为基分类器的情况下，模型出现过拟合现象，即过于拟合训练数据而导致测试准确率降低。从图3-5 (e)和(m)也可以看出，测试准确率在迭代结束前明显下降。除了DT本身容易过拟合、SA-ECOC没有采取有效防过拟合措施原因以外，微阵列数据集训练数据过少也是过拟合的重要原因。过拟合在微阵列数据上普遍且难以改善，是研究者必须直面的问题。

从对以上问题的分析来看，多分类微阵列数据具有难度，SA-ECOC具有一定的效果也存在亟待解决的问题。但是总体而言，不管基分类是高斯朴素贝叶斯还是DT， SA-ECOC在所有数据集上平均分类准确率都优于所有其他算法。可以确切地得出结论，SA-ECOC可以为不同的癌症微阵列数据集实现相对稳定和健壮的结果。

### 3.2.6 实验总结

为了针对微阵列数据“特征多样本少”的特点进行多类分类，本研究提出了一个新的纠错输出编码算法SA-ECOC。首先借助5个与特征相关的数据复杂性度量方法，配合二叉树编码方法来生成一个列池，并提出了一种列选择方法来构造一个编码矩阵。然后通过矩阵延长过程对矩阵进行迭代验证与延长。这样，就实现了一个强大的纠错算法的。为了验证SA-ECOC算法的有效性，将SA-ECOC算法应用于分类多类癌症微阵列数据，并与一些经典的ECOC算法进行比较，结果显示该算法可以产生好而稳定的分类准确率，迭代延长矩阵的过程和效果也在实验中得到了展示验证。从结果来看，SA-ECOC是在应用了这5个数据复杂度基础上才有较好的效果。在其他数据的复杂性测度的进一步研究将有利于文中的ECOC框架和帮助产生更好的编码矩阵，这是今后的研究方向。

## 3.3本章小结

本章提出一种针对微阵列数据的自调节纠错输出编码算法SA-ECOC。

3.1节粗略介绍算法步骤。接着介绍算法思路：针对微阵列“多特征少样本”、特征重叠度高的特点，提出相应划分特征子集、留一验证、采用数据复杂度等解决方法。此外，本节还介绍了5种数据复杂度，F1、F2、F3、N2、N3，用来作为二叉树编码方法的分支标准。3.1.3和3.1.4小节分别就算法的两个步骤进行了详细描述。首先利用建立二叉树的方法，配合5个与特征相关的数据复杂性度量方法来生成一个列池，选择准确率高的列构造一个编码矩阵。然后通过矩阵延长过程对矩阵进行迭代优化。这样，就实现了一个强大的纠错编码算法。

3.2节主要介绍实验内容。介绍8个做实验的癌症微阵列数据集情况和来源；介绍5个特征选择方法，用热力图展示8个数据集上特征子集两两覆盖情况，并分析了覆盖度对实验的影响；介绍了SA-ECOC编码矩阵的长度，与经典ECOC方法比较长度更长，接近理论推荐长度；为了展示SA-ECOC迭代过程的有效性，这部分展示了迭代过程中，随着矩阵长度增加，编码矩阵分别在训练集和测试集上准确率的变化情况，所有数据集的分类准确率整体上是增加趋势；最后展示了SA-ECOC配合5钟数据复杂度编码矩阵的分类准确率，与经典ECOC方法比较更好更稳定。

# 第四章、基于距离个性化编码ECOC算法

现有的ECOC编码方法，所有测试数据采用同一种编码矩阵，没有根据样本个体编码的方法。本章提出个性化纠错输出编码方法（customization error correct output code, CECOC）针对不同的样本，生成样本特有的个性化编码矩阵。

## 4.1方法介绍

个性化纠错输出编码方法（CECOC）采用固定编码矩阵，即每个样本的编码矩阵都一样。这似乎与“各样本特有的个性化矩阵”矛盾，实际上每个样本的编码矩阵行和列都相同，不同的是每个列对应的特征子集不同。CECOC先生成一个固定编码矩阵，然后将训练数据分成多个特征子集，根据样本的不同，为固定矩阵的每一列对应不同的特征子集，这就产生了各样本特有的个性化编码矩阵。 假如特征子集有*n*个，固定矩阵有*m*列，则可以产生*n*×*m*个不同的矩阵。本方法被分别应用于UCI数据集和微阵列数据集上，分类效果都有一定提升。

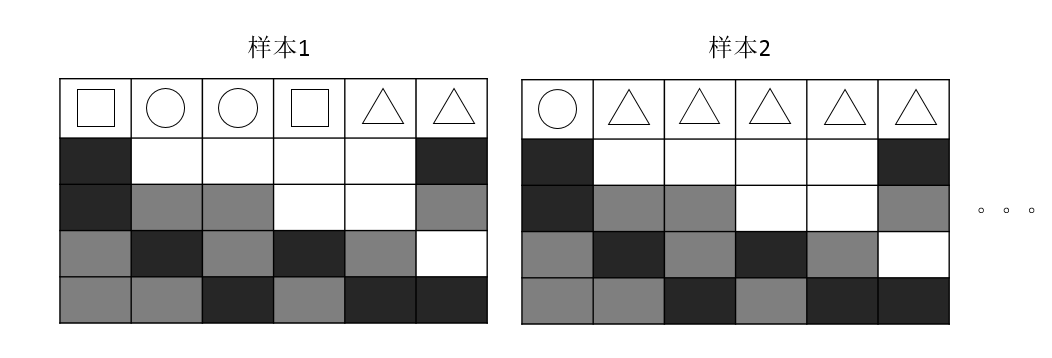
### 4.1.1 方法思路

Kuncheva[55]提出一种集成方法Random Linear Oracle，将数据样本集合随机线性划分成两个子集合（划分线叫oracle）。对不同集合分别训练，根据待预测样本到两个集合的距离，选择较近集合训练出来的分类器预测。在UCI数据集上的实验显示，这项工作分类效果有了较好的提升。该研究的两个启示：一是针对相同数据不同方面可以训练出冗余分类器，然后进行筛选；二是通过距离测度，选择适合的分类器。Random Linear Oracle粗略划分数据集，训练分类器，分类效果提升的原因可能是测试样本“二选一”分类器的机会，从某种意义上来说是“个性化”分类的一种。只是划分数据集随机性太强，划分样本的oracle缺乏依据。

本方法借鉴Random Linear Oracle的思路，在同一个固定编码矩阵中，对每个样本矩阵每一列对应不同特征子集。是 “多选一”特征子集的思想。

### 4.1.2 详细介绍

首先选择一种固定矩阵M，如一对一或者稠密随机矩阵等。然后对训练数据划分特征子集（特征数量一样且经过归一化）。对于每一个待预测样本和M中每一个列选择一个特征子集与该列对应。选择方法是计算该样本与不同特征子集的距离测度，选择距离测度小的特征子集。因为不同特征子空间展示数据不同方面的特点。所以在不同的特征子空间，同一种分类模型的分类效果也有高有低。在数据的特征空间被划分成若干个特征子空间的时候，本方法希望在对样本分类的时候，能利用到最能反映该样本特点的特征子空间，或者说是针对该样本最贴近的特征子空间。图4-1展示的个性化编码矩阵的例子，不同样本编码相同，但列对应的特征子集不同。图中三角形、方形、圆形代表不同特征子集。



**图4-1：个性化编码矩阵**

### 4.1.3 距离测度

本方法利用距离测度找到列对应的特征子集，所有方法关键点在于如何定义距离测度。实际上很多高效的分类器都采用欧式距离来作为分类标准，比如K近邻和SVM都采用欧式距离作为分类依据。欧氏距离公式如(4-1)：

(4-1)

但是本方法中，最终决定特征子集的距离测度并不仅仅是欧氏距离，而是整体距离与正负差距的差值，如公式(4-3)所示。

整体距离：一个样本点与一个样本集的距离被称为整体距离。整体距离可以分成如下三种：中心距离、最近距离和最远距离。

|  |
| --- |
| 1. 中心距离：样本与特征子空间中心点的距离。  2. 最近距离：样本与特征子空间点距离的最小值。  3. 最远距离：样本与特征子空间点距离的最大值的倒数。 |

正负差距：一个样本点与正类样本集的整体距离被称为正类距离，一个样本点与负类样本集的整体距离被称为负类距离，正类距离和负类距离的差值的绝对值被称为正负差距。如公式(4-2)所示。

(4-2)

(4-3)

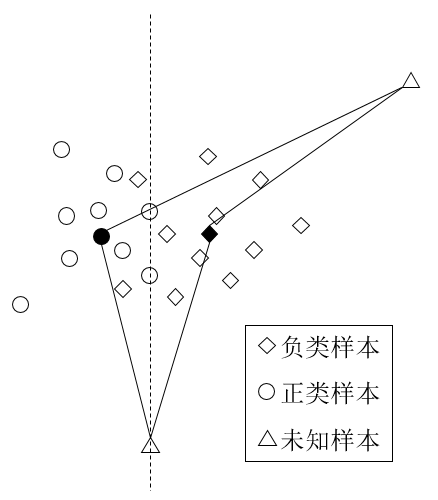
上式中*x*是一个样本，***Y***是样本集合，表示整体距离，表示正负差距，和分别表示正类样本集和负类样本集。表示正负差距系数，表示***Y***对应特征子集的权重。具体选择方法如下：

|  |
| --- |
| 对每个未知样本的每个矩阵列选择特征子集  k表示特征子集，表示k上的测试样本和训练样本集，表示矩阵第j列对应编码不为0的类别。 |

关于在距离测度中加入正负差距的原因，如果一个样本的正负差距较大，说明该样本偏向于正类或者负类，说明其更容易被正确分类。如图4-2，对于线性分类器来说，图中右上角的未知样本更容易被正确分类，因为它正负差距较大，离分类线较远；而下方的未知样本正负差距较小，接近分类线，较难被正确分类。正负差距在一定程度上反映了未知样本被正确分类的难易程度。

关于特征子集的权重*w*，为了提高分类的准确率，引入分类模型在特征子集上对验证数据集的分类错误率作为特征子集的权重。一般而言，选择验证正确率作为权重，因为数值越大表明重要性越大。但是在此处，距离作为测量标准是越小重要性越大，所以选择错误作权重。

在将运用权重的过程中，为了避免权重或距离任一方对乘积影响过大，所以同时对两者进行标准化，将两者的取值范围限定到同一个取值范围(0,1)。权重标准化采用“权重/权重之和”的方式。距离标准化采用sigmoid方法，因为距离中存在负数。



**图4-2 正类距离和负类距离**

## 4.2 UCI数据集上的实验与分析

UCI数据库[56]是加州大学欧文分校（University of California Irvine, UCI）建立的数据库，该数据库免费提供许多科研领域的数据，许多先进研究成果都利用该数据库中数据得到的。截止本文编写时间，UCI数据库已有425个数据集，数据集包括适用的研究种类包括分类、回归、聚类；标签类别包括类别标签、数值标签、和混合标签；数据类型有连续性数值、离散型数值、时间序列、文本等。数据涉及领域更是包括日常生活、物理科学、计算机科学、社会学、商业、游戏等等。

### 4.2.1 实验数据集

实验使用的数据集是从UCI下载的数据。其中mfeat数据集包括从荷兰实用地图集合中提取的手写数字（“0”-“9”）的特征；isolate数据集为独立字母语音识别数据；satimage数据集是卫星图像中像素3x3地区多光谱值，分类目标是预测每个地区中心像素。wearable数据是可穿戴设备计算举重练习二头肌旋度变化；pendigits数据集是从44为作家手写文本的图片数据。各数据详情如下。

**表4-1：数据集 单位：个**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据集名 | 样本数（训练集/验证集/测试集） | 类别数 | 特征数 | 引用 |
| mfeat | 2000(1200/400/400) | 10 | 649 | [57] |
| isolet | 7797(4677/1558/1562) | 26 | 617 | [58] |
| satimage | 6435(3857/1284/1294) | 6 | 36 | [59] |
| wearable | 4024(2413/804/807) | 5 | 42 | [60] |
| pendigits | 10992(6592/2195/2205) | 10 | 16 | [61] |

### 4.2.2 预处理方法

对数据进行预处理主要是为了方便距离测度的计算，不经过预处理所计算的距离测度很难相互比较。预处理方法包括划分数据集、正则化数据、特征选择和奇异点剔除。

数据划分采用3:1:1的方式，也就是3/5的样本是训练样本集，1/5的样本是验证样本集，1/5的样本是测试样本集。部分数据存在不均衡的情况，我们在划分样本集的时候，同一类数据在三个样本集中的分布也服从3:1:1的比例。

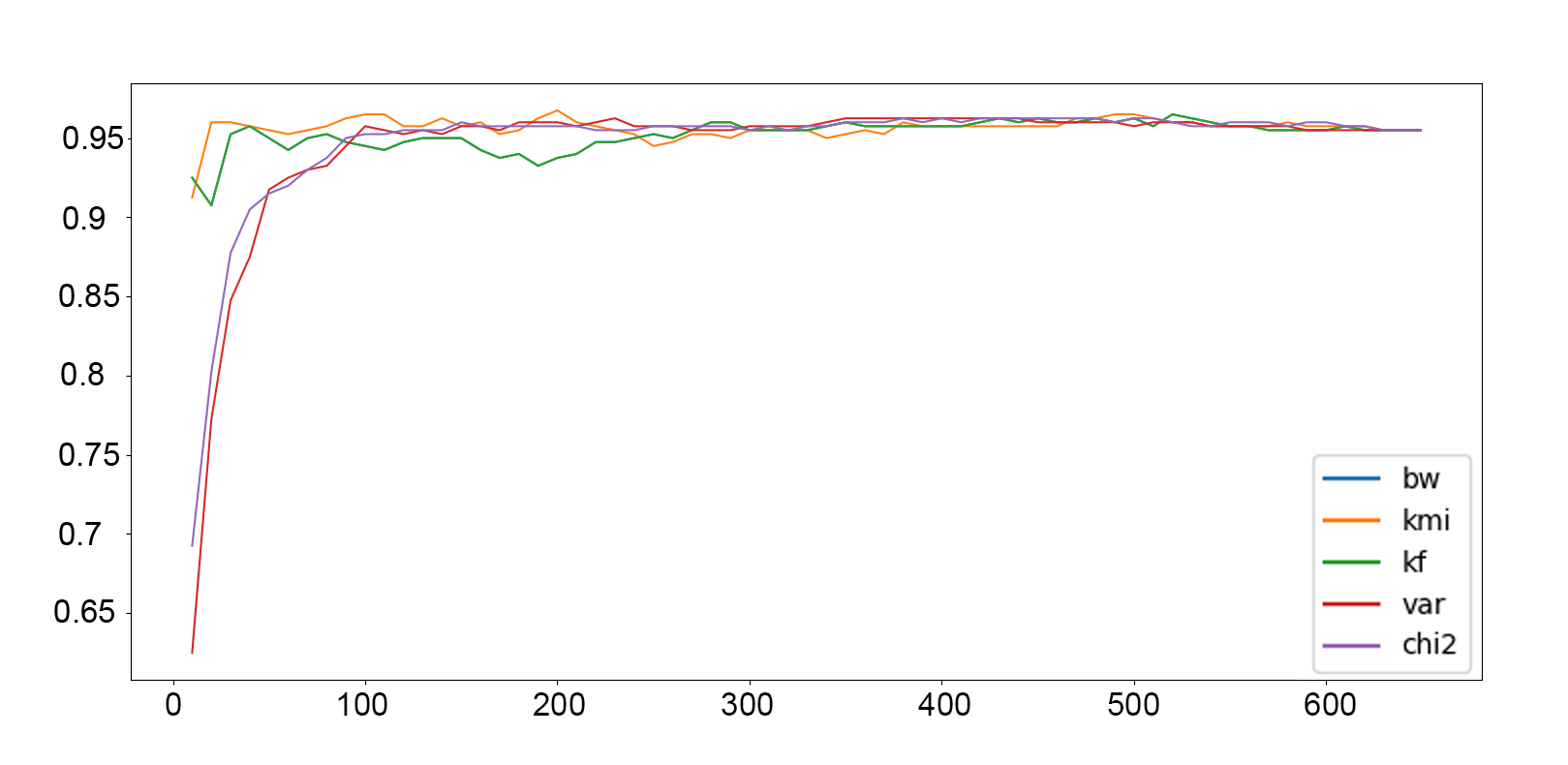
在数据预处理的阶段，采用MinMax的正则化（normalization）方式。因为这些数据将用来计算样本间距离，MinMax相对于其他一些正则化方法（如Z-scores和sigmoid）的优点是，它能在保有数值之间距离尺度的情况下，将数据映射到(0,1)范围。这种方法正则化之后的数据适合用于计算欧氏距离。

采用五种特征选择方法（BSS/WSS、互信息、ANOVA F-value、方差、卡方检验）构造五个特征子空间。BSS/WSS和方差的方法来自于ecoclib工具包，互信息、ANOVA F-value和卡方检验由scikit-learn提供。特征选择方法只在训练数据集上执行，验证数据和测试数据沿用在训练数据集上选取的特征。

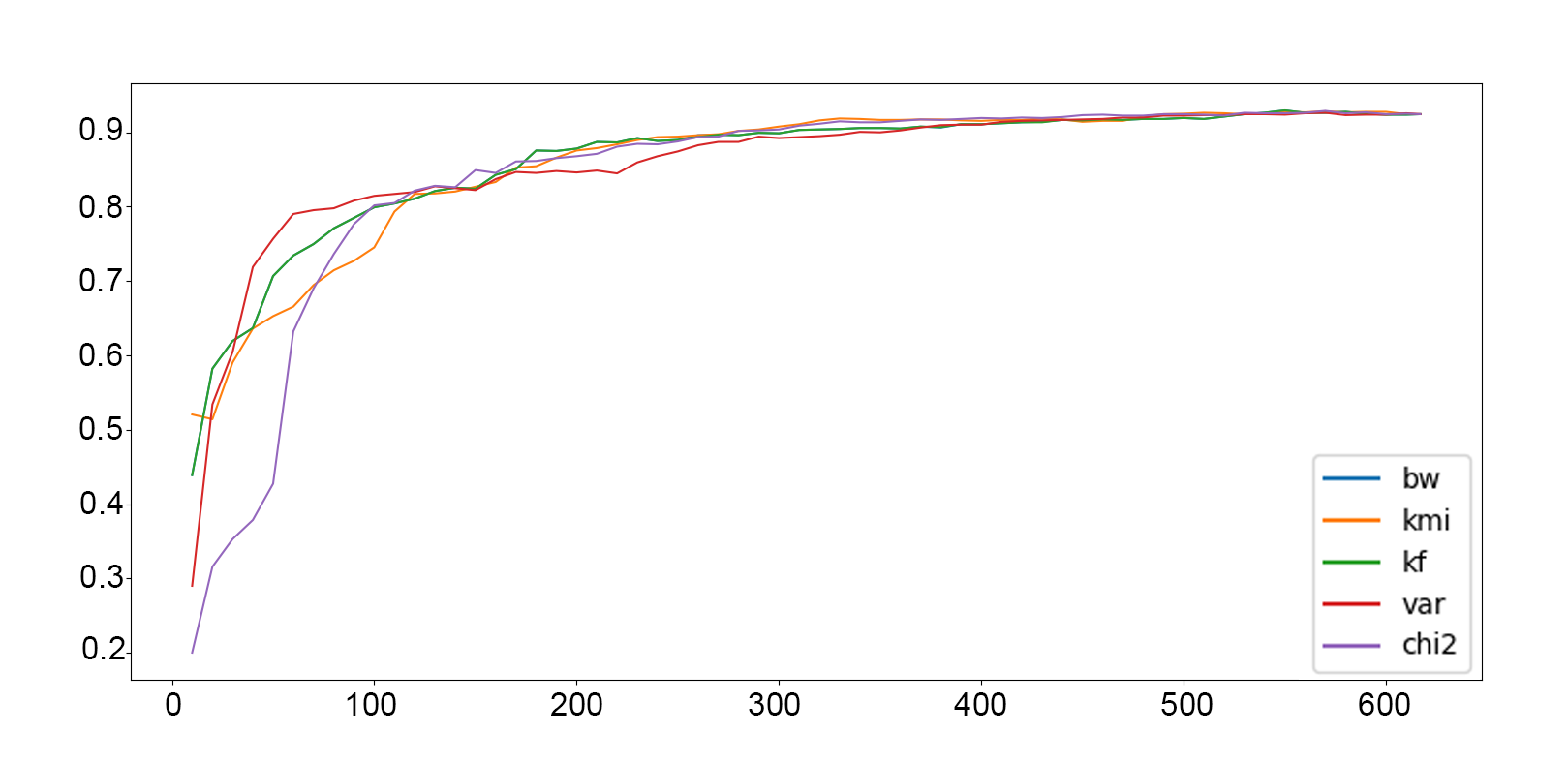
预处理的另一个方面是奇异点（outlier）剔除，奇异点是，使用的是scikit-learn的EllipticEnvelope方法。基本思想是绘制一个椭圆，将椭圆之外的数据点作为奇异点。本实验用此方法剔除了5%的样本点，这个过程同样只在训练数据集上执行。

### 4.2.3 选择特征个数

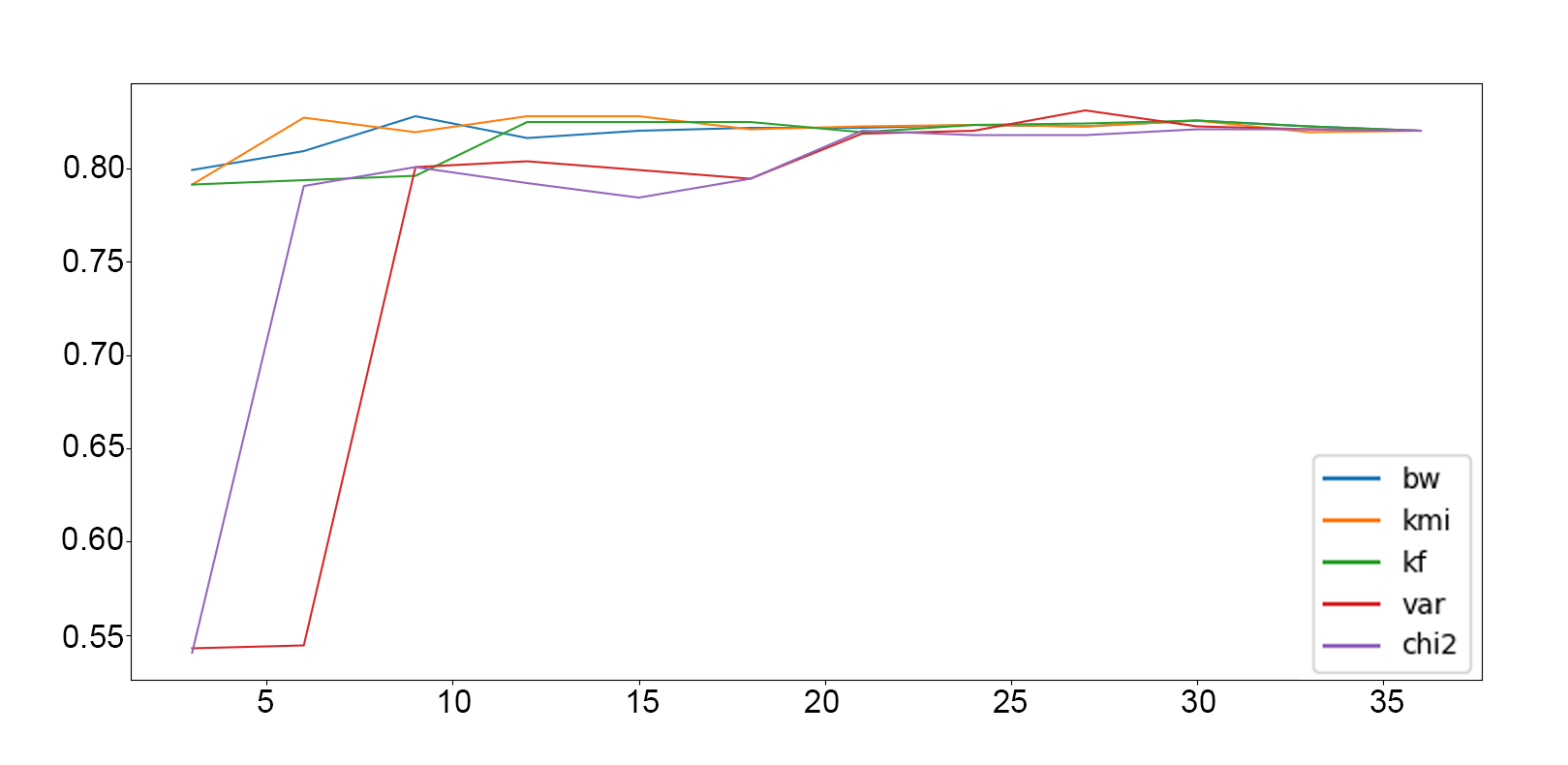
实验中我们选择固定数量的特征（即每一个特征子集包含的特征数目相同），这样做主要的目的是计算的距离不会因为特征数目不同而产生偏差。选择特征个数需要满足另一个条件：分类器在每个特征子集的验证准确率超过65%。因为根据集成学习的理念，被集成的分类器要有一定的准确率，集成结果才好。图4-3至图4-7展示的是5个数据集5个特征子集在SVM分类器下分类准确率。表4-2是图中特征子集代号对应的特征选择方法。依据图4-3至图4-7所示，不同数据集选择特征个数如表4-3所示。



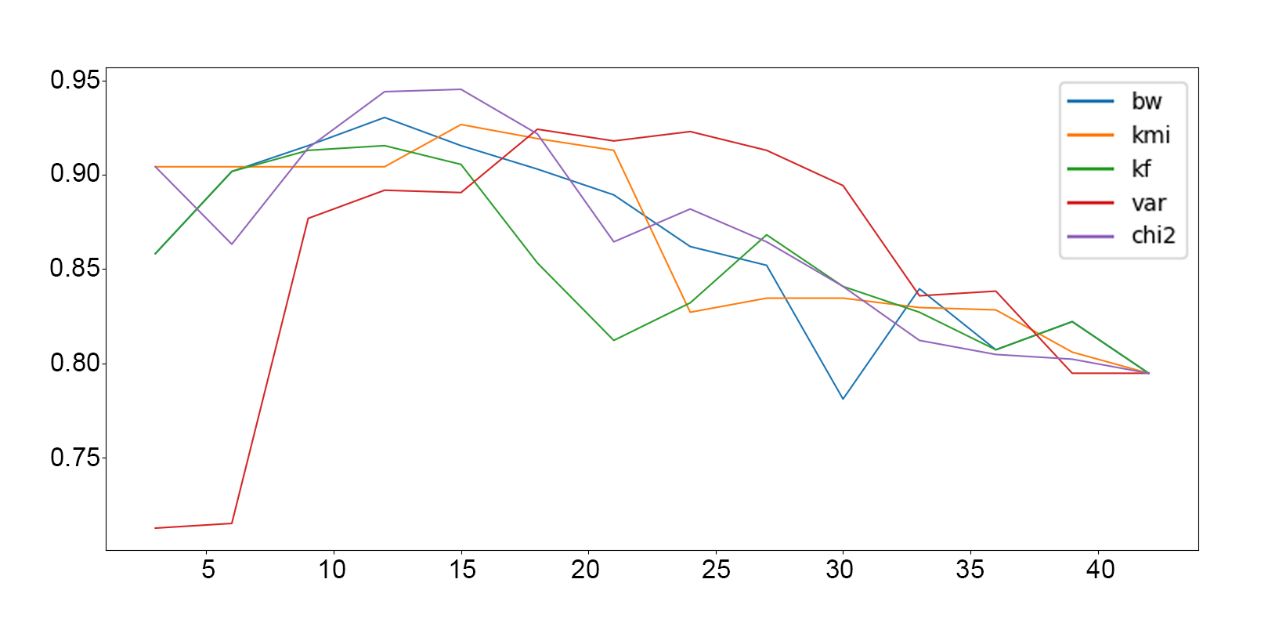
**图4-3:mfeat数据集特征数量和分类准确率关系**



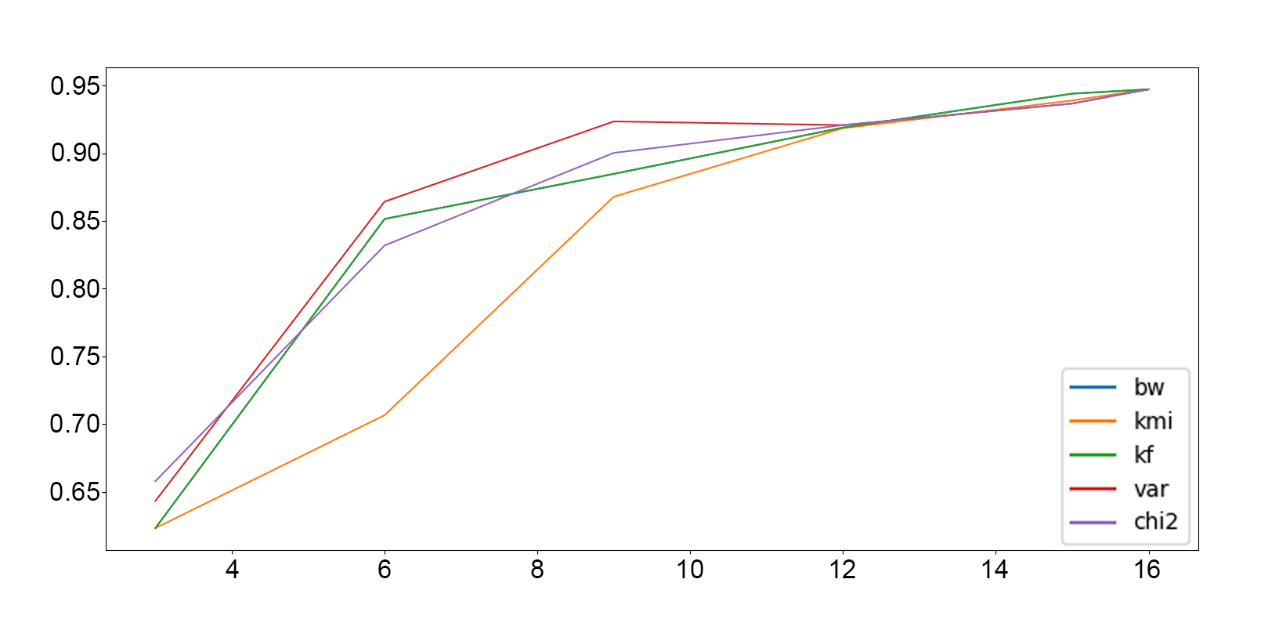
**图4-4:isolet数据集特征数量和分类准确率关系**



**图4-5:satimage数据集特征数量和分类准确率关系**



**图4-6:wearable数据集特征数量和分类准确率关系**



**图4-7:pendigits数据集特征数量和分类准确率关系**

**表4-2：图中特征子集代号对应特征选择方法**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 特征子集 | bw | kmi | kf | var | chi2 |
| 对应方法 | 公式(2-5) | 公式(2-4) | 公式(2-3) | 公式(2-6) | 公式(2-7) |

**表4-3：选择特征个数对应情况**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 数据集 | mfeat | isolet | satimage | wearable | pendigits |
| 特征个数 | 100 | 200 | 10 | 10 | 10 |

### 4.2.4 实验结果

实验采用稀疏随机编码方式，选择SVM作为基分类器。挑选了三种整体距离测度（EF最远欧氏距离、EN最近欧氏距离、EC中心欧式距离）。表4-4至表4-6中，“EC”、“EF”和“EN”展示的是CECOC分别使用中心欧氏距离、最远欧氏距离和最近欧氏距离时，个性化矩阵分类实验结果。“非CECOC”展示的是非个性化编码在5个特征子集上实验准确率的“均值±标准差”，“最优”和“最差”分别表示非个性化编码5个结果中最好准确率和最差准确率。值得注意的是，表4-4至表4-6的实验差别在于数值不同，这回影响CECOC的分类结果，但不会影响非CECOC的分类结果，所以三个表的后三列完全相同。

正负差距系数选择0.5时，表4-4显示，5组对比实验中，4组实验CECOC准确率高于非CECOC均值，不同整体距离的表现均高于最差值。其中3组更是高于最优值。

**表4-4：CECOC（=0.5）分类准确率对比**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据 | CECOC | | | 非CECOC | | |
| EC | EF | EN | 平均 | 最优 | 最差 |
| mfeat | 0.885 | 0.885 | 0.885 | 0.8920±0.0116 | **0.915** | 0.885 |
| isolet | 0.6492 | 0.6492 | **0.6517** | 0.6324±0.0139 | 0.6492 | 0.614 |
| satimage | 0.7009 | 0.6955 | **0.7017** | 0.6790±0.0187 | 0.6994 | 0.6522 |
| wearable | 0.9504 | 0.9504 | 0.9504 | 0.9214±0.0250 | **0.9529** | 0.8947 |
| pendigits | 0.7624 | **0.7805** | 0.7705 | 0.7595±0.0156 | **0.7805** | 0.7429 |

正负差距系数选择1.0时，表4-5显示，5组对比实验中，4组实验CECOC准确率高于非CECOC均值，不同整体距离的表现均高于最差值。其中2组高于最优值。与表4-4的分类结果相比，“satimage”和“pendigits”上表现略有下降，“mfeat”和“isolate”略有上升。

正负差距系数选择1.0时，表4-6显示，5组对比实验中，所有组不同整体距离的表现均高于均值和最差值，其中4组实验CECOC准确率高于最优值。

**表4-5：CECOC（=1.0）分类准确率对比**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据 | CECOC | | | 非CECOC | | |
| EC | EF | EN | 平均 | 最优 | 最差 |
| mfeat | 0.8875 | 0.885 | 0.8975 | 0.8920±0.0116 | **0.915** | 0.885 |
| isolet | 0.6511 | 0.6492 | **0.6581** | 0.6324±0.0139 | 0.6492 | 0.614 |
| satimage | 0.6971 | **0.6994** | 0.6917 | 0.6790±0.0187 | **0.6994** | 0.6522 |
| wearable | 0.9504 | 0.9504 | 0.9504 | 0.9214±0.0250 | **0.9529** | 0.8947 |
| pendigits | 0.7596 | 0.7796 | 0.7746 | 0.7595±0.0156 | **0.7805** | 0.7429 |

**表4-6：CECOC（=2.0）分类准确率对比**

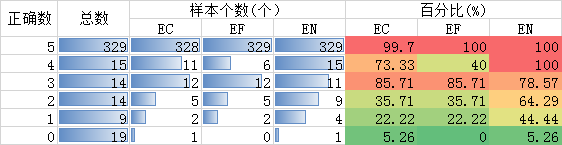
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据 | CECOC | | | 非CECOC | | |
| EC | EF | EN | 平均 | 最优 | 最差 |
| mfeat | 0.8975 | 0.885 | **0.9225** | 0.8920±0.0116 | 0.915 | 0.885 |
| isolet | 0.6466 | 0.6492 | **0.6767** | 0.6324±0.0139 | 0.6492 | 0.614 |
| satimage | 0.6886 | 0.6901 | **0.7032** | 0.6790±0.0187 | 0.6994 | 0.6522 |
| wearable | 0.9504 | 0.9504 | 0.9393 | 0.9214±0.0250 | **0.9529** | 0.8947 |
| pendigits | 0.7578 | 0.7737 | **0.781** | 0.7595±0.0156 | 0.7805 | 0.7429 |

从表4-4至表4-6中可以看出，CECOC对分类效率的提升明显。其集成的稳定性高、分类提升效果好。尤其当取值2.0的时候效果较为出众。

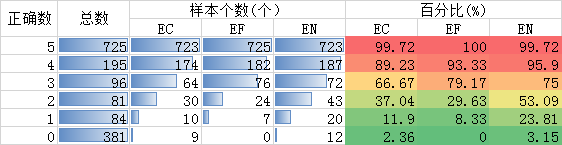
### 4.2.5 不同距离比较

表4-7至表4-11展示的是=2.0时5个数据集在三种整体距离上集成的效果，表格第1列“正确数”指在几个特征子集上分类结果正确，“总数”表示样本数。比如表4-7数据第1行表示在5个特征子集都分类正确的样本有329个，其中用EC作为整体距离分类正确的有328个，占比99.7%；EF和EN作为整体距离分类正确的都是329个，占比100%。对于正确数较少（2,1或者0）的样本，若某种整体距离的集成方法能将样本正确分类，说明集成效果较好。从表4-7至表4-11可以看出的一个普遍规律是以EN为整体距离的集成效果较好。表4-6的准确率也印证了这一点。

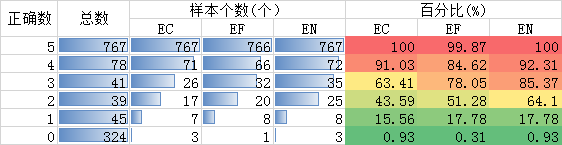
**表4-7：mfeat数据集三种整体距离上集成效果展示**



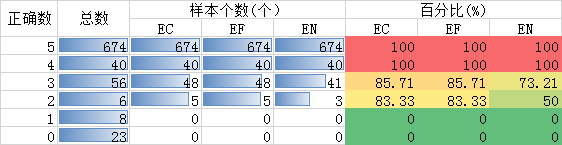
**表4-8：isolet数据集三种整体距离上集成效果展示**



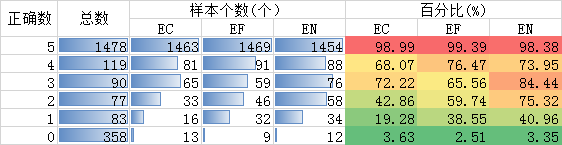
**表4-9：satimage数据集三种整体距离上集成效果展示**



**表4-10：wearable数据集三种整体距离上集成效果展示**



**表4-11：pendigits数据集三种整体距离上集成效果展示**



### 4.2.6 实验总结

实验采用5中特征选择方法，利用UCI数据集，选择了3种距离（EF最远欧氏距离、EN最近欧氏距离、EC中心欧式距离），在取值0.5,1.0,2.0时分别做了对比实验。对比没有进行个性化编码的结果有相对稳定的结果，在某些数据集上，分类准确率比非CECO的最高准确率更好。

CECOC方法也存在弊端，为每个样本每个列计算选择特征子集，有很高的运算量。

## 4.3微阵列数据实验和分析

微阵列数据与UCI数据不同，其特点已在第一章“1.2.1微阵列数据特点”中进行过描述，此处不再赘述。针对微阵列数据的特点，数据预处理方法与UCI数据的预处理方法略有不同，下文将就不同点进行阐述。

### 4.3.1数据集介绍&预处理方法

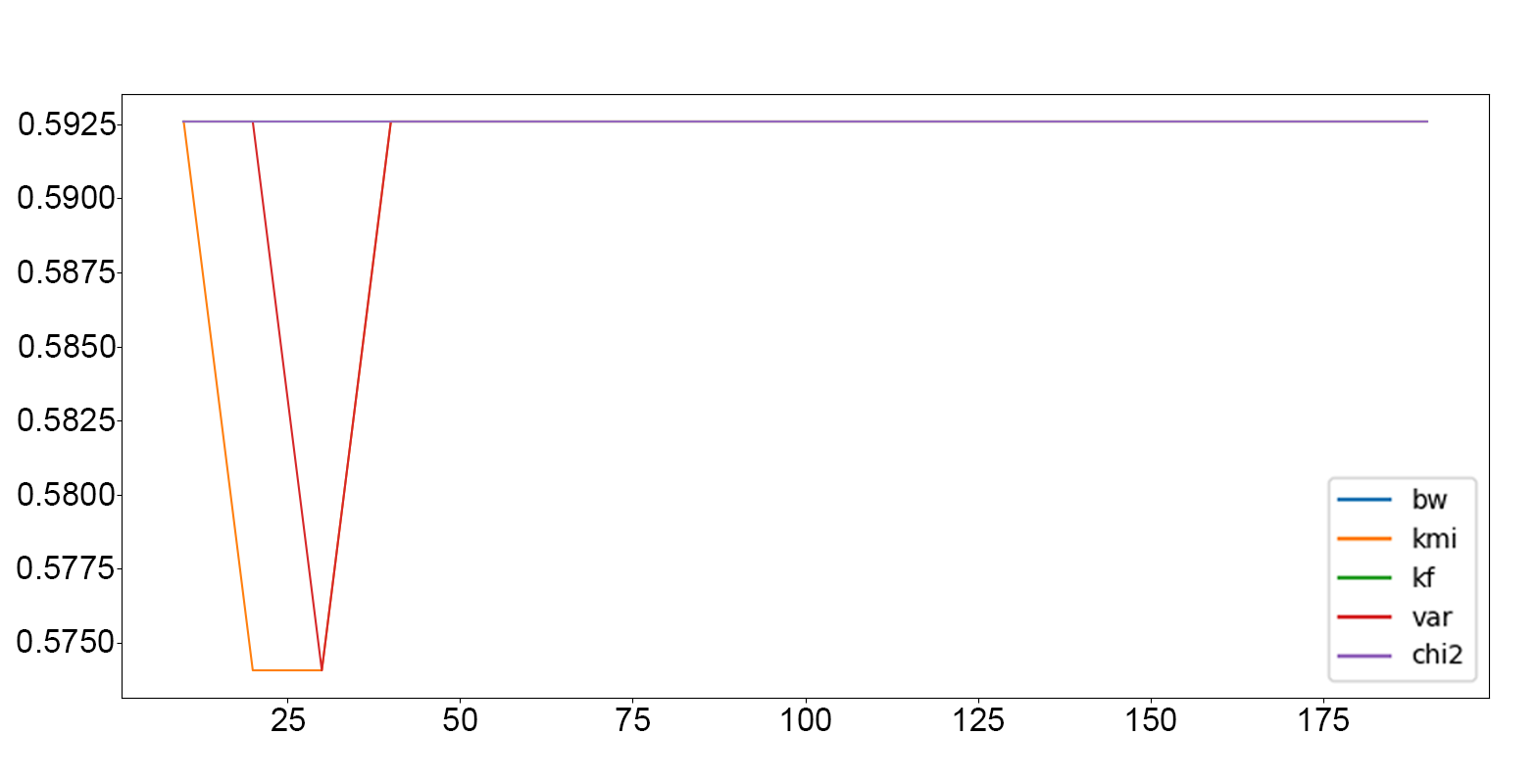
本章使用的微阵列数据集与本文第三章“3.2.1数据集介绍”一样，此处不再赘述。

如本章“4.2.2预处理方法”所述，对UCI数据的预处理包括：划分验证集、MinMax归一化、使用5种方法选择特征、剔除异常点。而对微阵列数据处理时，不划分验证集也不剔除异常点，原因是这两种预处理是建立在数据数据量较大的基础上的，微阵列数据样本数据特别少，不适宜做以上预处理。CECOC中有利用验证准确率作为特征子集的权重w的步骤，对微阵列数据做实验时，采用留一验证法计算权重。

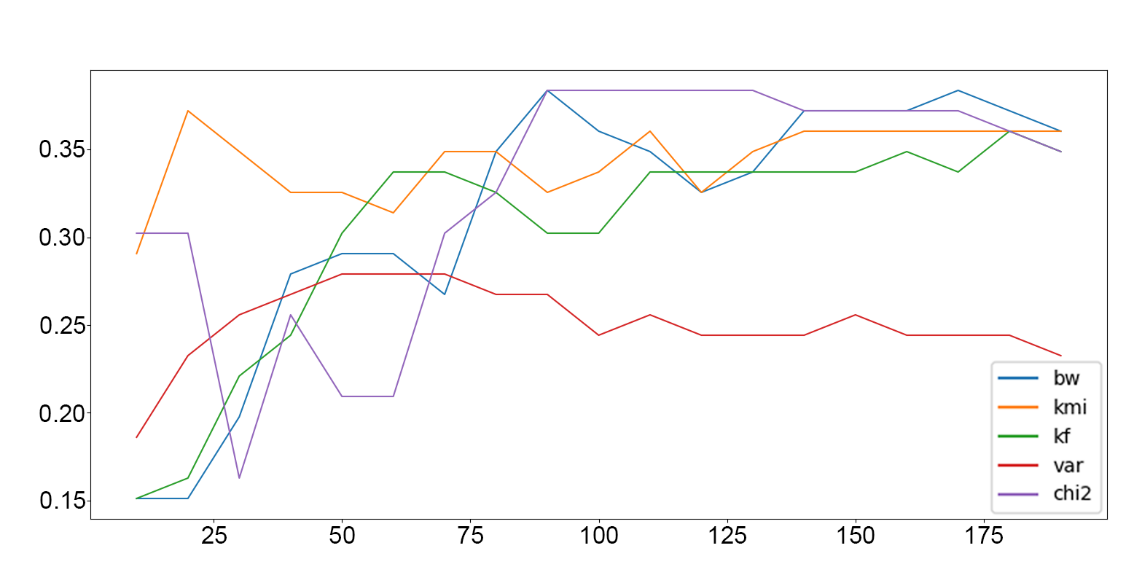
### 4.3.2选择特征个数

实验中我们选择固定数量的特征（即每一个特征子集包含的特征数目相同），这样做主要的目的是计算的距离不会因为特征数目不同而产生偏差。选择特征个数需要满足另一个条件：分类器在每个特征子集的验证准确率普遍较高。因为根据集成学习的理念，被集成的分类器要有一定的准确率，集成结果才好。

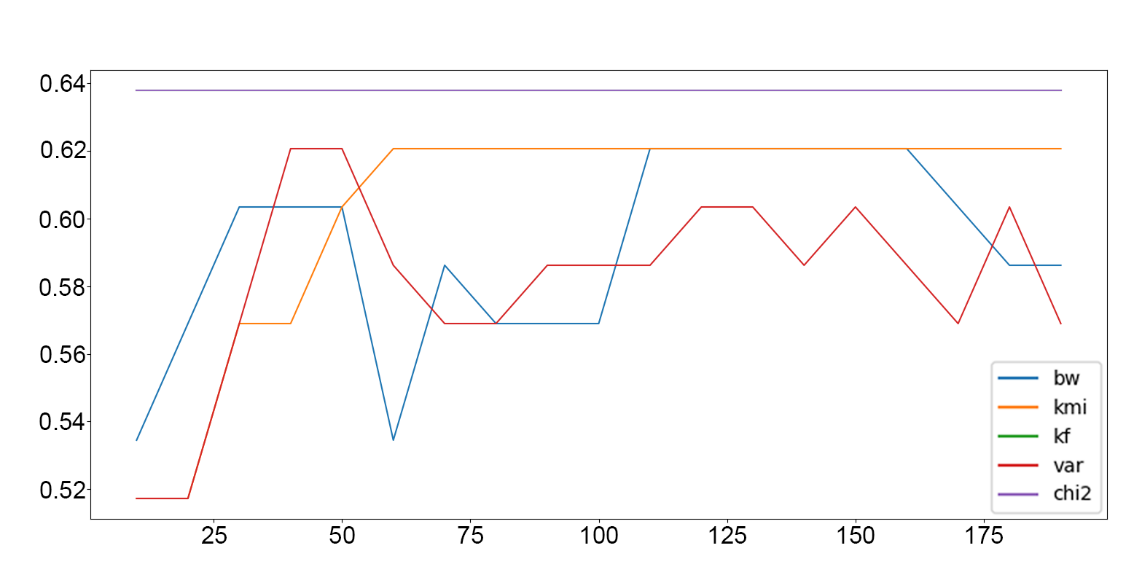
微阵列数据的特征数远远多于200，但图4-8至图4-15只展示200以内特征个数与分类准确率的关系，特征多于200时，其分类准确率均不比200以内特征对应准确率高，因此仅展示200特征以内的关系图。



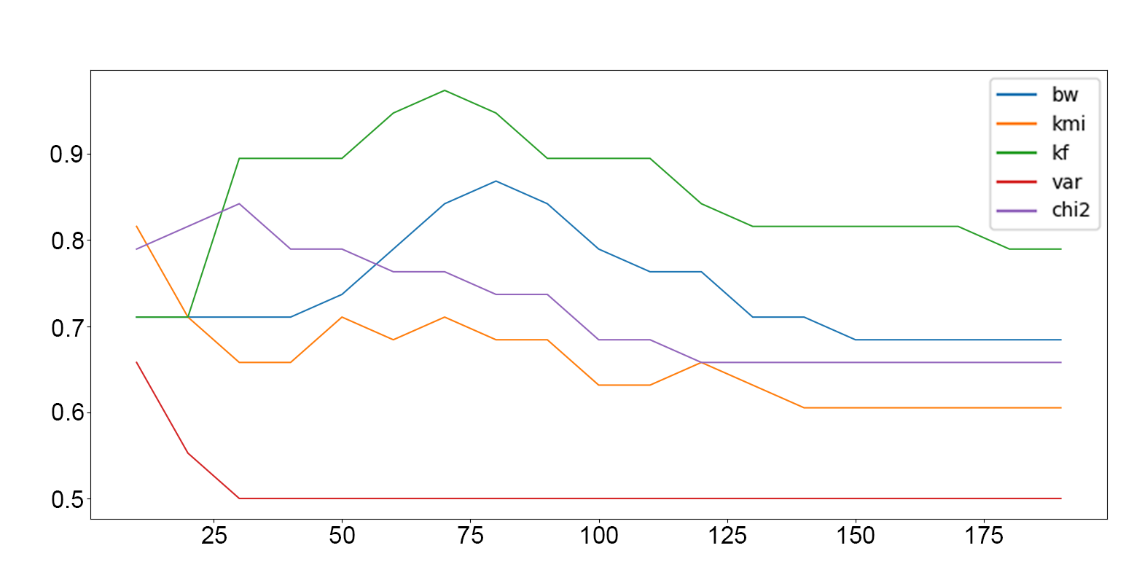
**图 4-8:** **Breast数据集特征数量和分类准确率关系**



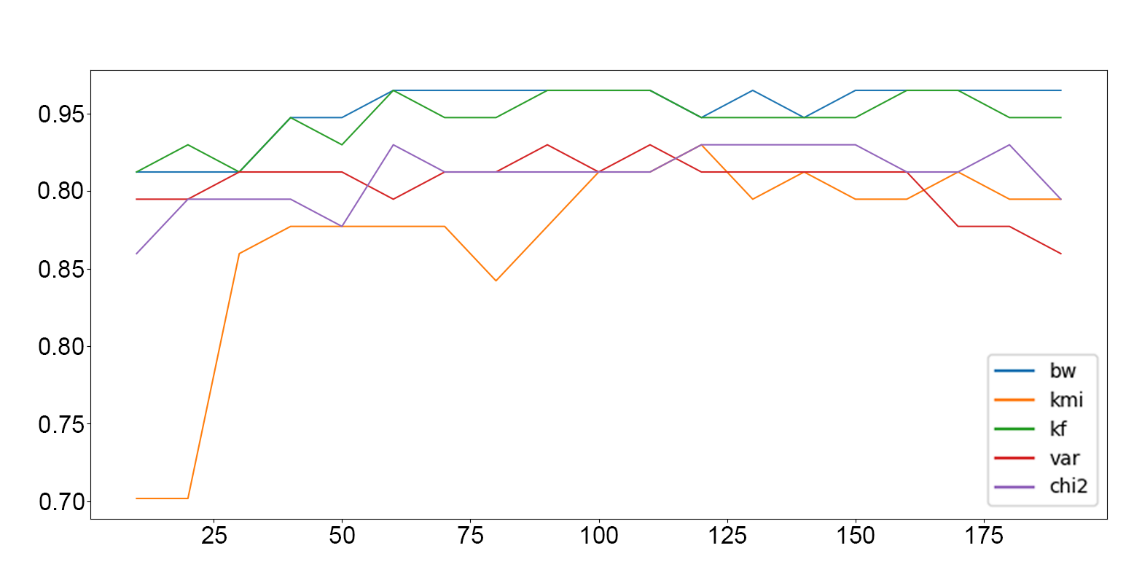
**图 4-9:** **Cancers数据集特征数量和分类准确率关系**



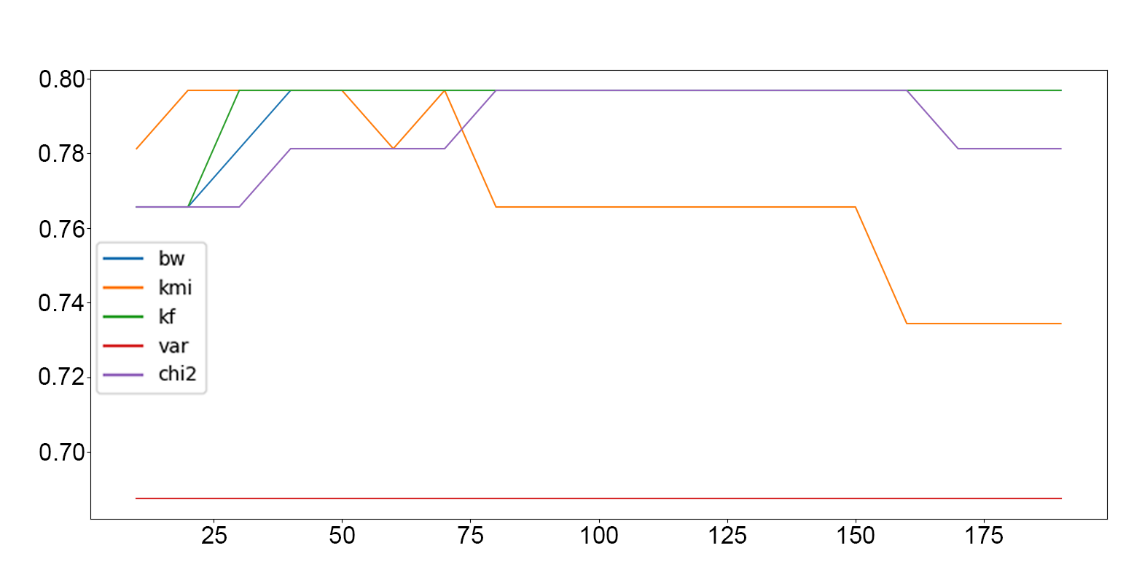
**图 4-10:** **DLBCL数据集特征数量和分类准确率关系**



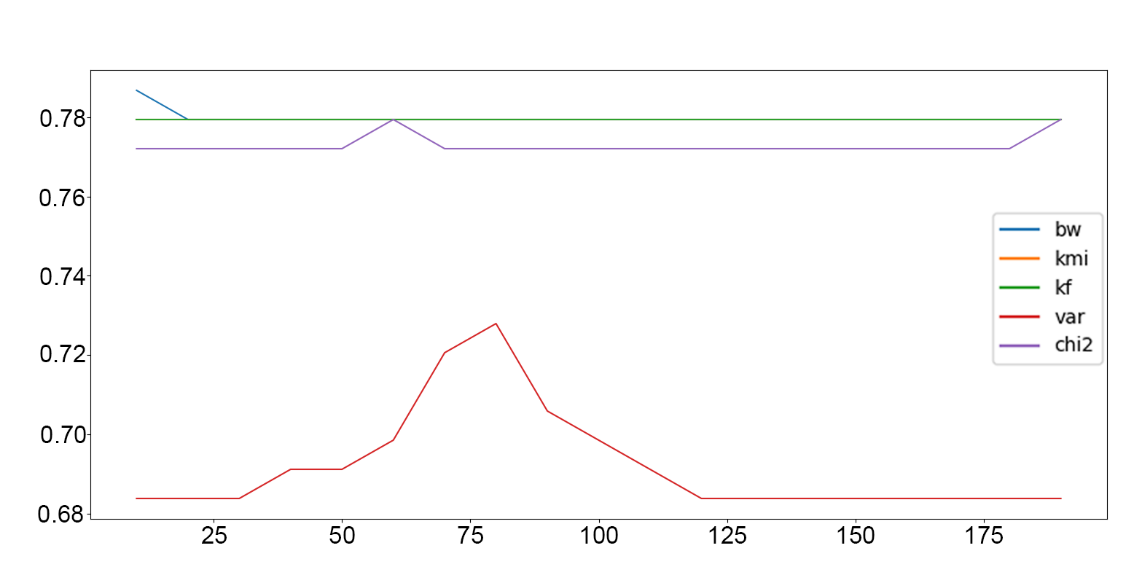
**图 4-11:** **Leukemia1数据集特征数量和分类准确率关系**



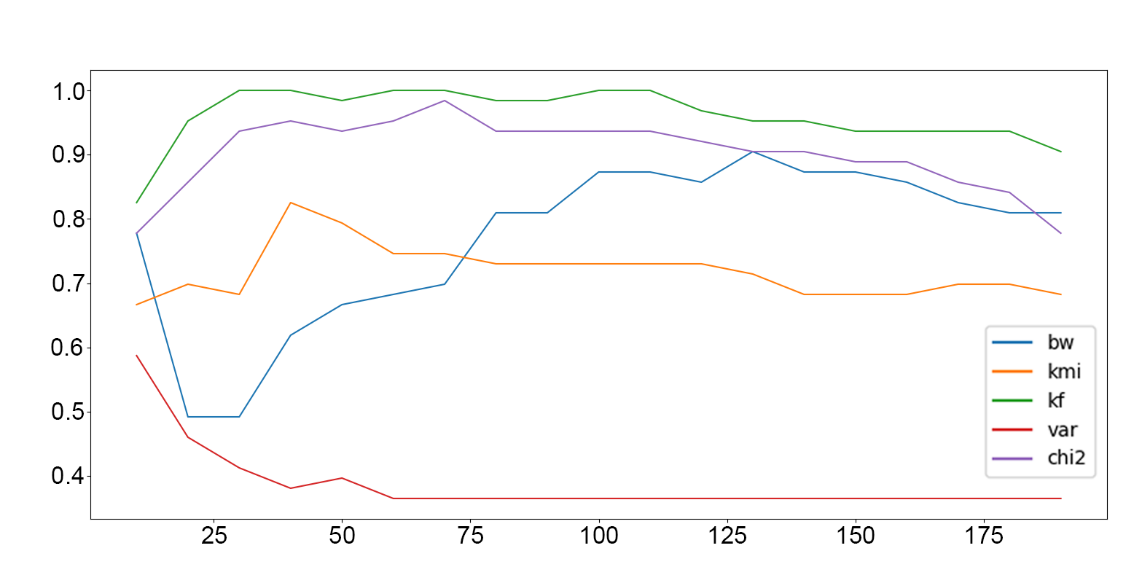
**图 4-12:** **Leukemia2数据集特征数量和分类准确率关系**



**图 4-13:** **Lung1数据集特征数量和分类准确率关系**



**图 4-14:** **Lung2数据集特征数量和分类准确率关系**



**图 4-15: SRBCT数据集特征数量和分类准确率关系**

根据图图4-8至图4-15为各个数据集选择的特征的个数如表4-12所示，保证在对应特征个数的数据集中，各个特征子集上的分类准确率不差。

**表4-12：选择特征个数对应情况**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 数据 | Breast | Cancers | DLBCL | Leukemia1 | Leukemia2 | Lung1 | Lung2 | SRBCT |
| 特征 | 40 | 90 | 120 | 70 | 120 | 70 | 80 | 40 |

### 4.3.3实验结果

实验采用稀疏随机编码方式，选择SVM作为基分类器。挑选了三种整体距离测度（EF最远欧氏距离、EN最近欧氏距离、EC中心欧式距离）。表4-4至表4-6中，“EC”、“EF”和“EN”展示的是CECOC分别使用中心欧氏距离、最远欧氏距离和最近欧氏距离时，个性化矩阵分类实验结果。“非CECOC”展示的是非个性化编码在5个特征子集上实验准确率的“均值±标准差”，“最优”和“最差”分别表示非个性化编码5个结果中最好准确率和最差准确率。值得注意的是，表4-4至表4-6的实验差别在于数值不同，这回影响CECOC的分类结果，但不会影响非CECOC的分类结果，所以三个表的后三列完全相同。

正负差距系数选择0.5时，表4-13显示，8组对比实验中，仅仅Lung1在EC距离下准确率不高于非CECOC均值，不同整体距离的表现均高于最差值。其中Cancers和DLBCL在EN距离下更是高于最优值。

**表4-13：CECOC（=0.5）分类准确率对比**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据 | CECOC | | | 非CECOC | | |
| EC | EF | EN | 平均 | 最优 | 最差 |
| Breast | **0.6333** | **0.6333** | **0.6333** | **0.6333+0.0000** | **0.6333** | **0.6333** |
| Cancers | 0.2568 | 0.2297 | **0.3108** | 0.2676+0.0232 | 0.2973 | 0.2297 |
| DLBCL | 0.7333 | 0.8000 | **0.8333** | 0.7333+0.0211 | 0.7667 | 0.7000 |
| Leukemia1 | **0.7647** | **0.7647** | **0.7647** | 0.7059+0.0767 | **0.7647** | 0.5588 |
| Leukemia2 | **0.8000** | **0.8000** | **0.8000** | **0.8000+0.0000** | **0.8000** | **0.8000** |
| Lung1 | 0.7812 | **0.8125** | **0.8125** | 0.7937+0.0375 | **0.8125** | 0.7188 |
| Lung2 | 0.7761 | **0.7910** | **0.7910** | 0.7761+0.0299 | **0.7910** | 0.7164 |
| SRBCT | 0.8500 | 0.8500 | 0.8500 | 0.7200+0.2421 | **0.9500** | 0.3000 |

当是1.0或2.0时（如图4-14和图4-15），实验准确率情况与是0.5的时候相似。

**表4-14：CECOC（=1.0）分类准确率对比**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据 | CECOC | | | 非CECOC | | |
| EC | EF | EN | 平均 | 最优 | 最差 |
| Breast | **0.6333** | **0.6333** | **0.6333** | **0.6333+0.0000** | **0.6333** | **0.6333** |
| Cancers | 0.2568 | 0.2162 | **0.3243** | 0.2676+0.0232 | 0.2973 | 0.2297 |
| DLBCL | 0.8000 | 0.7667 | **0.8333** | 0.7333+0.0211 | 0.7667 | 0.7000 |
| Leukemia1 | **0.7647** | **0.7647** | **0.7647** | 0.7059+0.0767 | **0.7647** | 0.5588 |
| Leukemia2 | **0.8000** | **0.8000** | **0.8000** | **0.8000+0.0000** | **0.8000** | **0.8000** |
| Lung1 | **0.8125** | **0.8125** | **0.8125** | 0.7937+0.0375 | **0.8125** | 0.7188 |
| Lung2 | **0.7910** | **0.7910** | **0.7910** | 0.7761+0.0299 | **0.7910** | 0.7164 |
| SRBCT | 0.8500 | 0.8500 | 0.8500 | 0.7200+0.2421 | **0.9500** | 0.3000 |

**表4-15：CECOC（=2.0）分类准确率对比**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据 | CECOC | | | 非CECOC | | |
| EC | EF | EN | 平均 | 最优 | 最差 |
| Breast | **0.6333** | **0.6333** | **0.6333** | **0.6333+0.0000** | **0.6333** | **0.6333** |
| Cancers | 0.2973 | 0.2162 | **0.3784** | 0.2676+0.0232 | 0.2973 | 0.2297 |
| DLBCL | 0.8000 | 0.7667 | **0.8333** | 0.7333+0.0211 | 0.7667 | 0.7000 |
| Leukemia1 | **0.7647** | **0.7647** | **0.7647** | 0.7059+0.0767 | **0.7647** | 0.5588 |
| Leukemia2 | **0.8000** | **0.8000** | **0.8000** | **0.8000+0.0000** | **0.8000** | **0.8000** |
| Lung1 | **0.8125** | **0.8125** | **0.8125** | 0.7937+0.0375 | **0.8125** | 0.7188 |
| Lung2 | **0.7910** | **0.7910** | **0.7910** | 0.7761+0.0299 | **0.7910** | 0.7164 |
| SRBCT | 0.8500 | 0.8000 | 0.8500 | 0.7200+0.2421 | **0.9500** | 0.3000 |

从表4-13至表4-15中可以看出，CECOC方法始终不低于平均值，表现较稳定。对于分类准确率整体较低的数据集也有一定的提升效果（如Cancers）。

但是也可以看出CECOC对分类效率的提升效果不如该方法对UCI数据的提升。究其原因主要是：(1) 特征选择比较困难。从图4-8至图4-15就可以看出特征子集之间表现差异大，有些特征子集效果好、有些效果差；而Breast和Leukemia2两个数据集的特征子集几乎没有差异。(2) 本实验采用的线性支持向量机，对数据样本较少的数据集比较无力。

分类效果不理想根本原因是微阵列数据太复杂，考虑解决方法可以是(1) 进行两层特征选择。第一层选择一个分类效果良好的特征子集，第二层则随机采样这个特征子集，这样各特征子集的差异不会太大也不会太小。

## 4.4本章小结

本章提出一种个性化纠错输出编码方法（CECOC），为每个样本设计个性化编码矩阵的方法。方法为一个固定编码矩阵的列选择对应特征子集，选择方法是判定该样本与特征子集的距离测度，选择距离测度小的特征子集。这种方法被引用于UCI数据集，取得较好的效果。

4.1节描述了方法思路和实现细节，有提出了方法中距离测度的定义。

4.2节介绍实验，介绍了UCI数据的具体情况；介绍了数据预处理包含特征选择、正则化数据和奇异值提出，这些预处理方法有利于距离测度的比较；介绍了实验中用折线图确定选择特征的个数；最后进行试验比较了CECOC和没用CECOC方法准确率的区别。

4.3节将4.2节同样的方法应用于3.2.1小节介绍的微阵列数据，展示实验结果并分析微阵列数据实验效果不如UCI数据实验效果的原因。

# 第五章、ECOC开发工具包

一直以来，ECOC的研究工作主要是借助matlab实现的，matlab官方和一个顶尖ECOC研究组提供了相关matlab开发工具包。然而一些优秀的Python工具包（比如scikit-learn, shogun虽然也提供多分类框架，但是其功能重点不是ECOC。这些工具包只提供了三种以内的基本编码方法和一种固定的解码方法；不仅编码解码方法稀少，其ECOC框架实现也有部分错误（比如在三元编码中训练基分类器的时候，训练数据并没有剔除0对应类别数据）。

Python越来越流，基于该语言的机器学习工具越来越多。为了供项目组使用，以及这个时代程序语言的发展，同时推广Python在ECOC领域的应用。本章介绍作者自行开发的基于Python语言ECOC工具包“ecoctlk”。该工具包现已作为开源工具，发布到Github。

<https://github.com/MLDMXM2017/ECOC_Methods/tree/master/ecoctlk>

## 5.1 ecoctlk介绍

为了让Python语言应用于ECOC研究，也让ECOC研究者更方便的使用Python机器学习工具包，用Python语言自行开发了ECOC工具包ecoctlk。工具包提供ECOC框架类，实现多种热门编码算法，从matlab工具包移植多种解码算法。除此以外还提供一些数据复杂度测算方法、与数据复杂度相关的特征选择方法。

### 5.1.1 ECOC框架类

常用Python工具包如sklearn和shogun提供的框架类有很多不足甚至是错误。例如Scikit-learn提供的ECOC框架类是：

sklearn.multiclass.OutputCodeClassifier

该框架类不提供自定义编码方法和自定义解码方法，编码只有一种固定编码，解码只有欧式距离解码，而且仅仅适用于二元编码。若研究者想使用自己设计的编码或者解码进行试验，则需要修改大量源代码。

ecoctlk提供一种ECOC框架类:

Classifier.ECOCClassifier.**SimpleECOCClassifier**

该类可以自定义ECOC矩阵，可以选择工具包提供的多种解码方法。附录1展示框架类使用方法。

### 5.1.2 编码方法

常用Python工具包如sklearn和shogun只提供了OVA和OVO编码，ecoctlk除了提供经典的OVA、OVO编码之外，还包含从ecoclib移植过来的DECOC和ECOC-ONE，也包含了其他ECOC方法,函数名对应编码方法如下：附录2展示编码调用方法。

|  |
| --- |
| ova: Get a One-vs-All matrix.  ovo: Get a One-vs-One matrix.  dense\_rand: Get a Dense Random matrix.  sparse\_rand: Get a Sparse Random matrix.  decoc: Get a Discriminant ECOC matrix.  agg\_ecoc: Get an Agglomerative matrix.  cl\_ecoc: Get a Centroid loss ECOC, which use regressors as base estimators.  ecoc\_one: Get an Optimal node embedded ECOC. |

### 5.1.3 解码方法

常用Python工具包如sklearn和shogun只提供欧式解码，ecoctlk提供了汉明解码、欧式解码、拉普拉斯解码、基于损失的解码等。解码方法对应代码如下： 附录3展示解码调用方法。

|  |
| --- |
| 'HD' - Hamming Decoder. 'IHD' - Inverse Hamming Decoder. 'LD' - Laplacian Decoder. 'ED' - Euclidean Decoder. 'AED' - Attenuated EuclideanD ecoder. 'RED' - Ratio Euclidean Decoder. 'EuD' - Euler Decoder. 'LLB' - Linear Loss Based Decoder. 'ELB' - Exponential Loss Based Decoder. 'LLW' - Linear Loss Weighted Decoder. 'ELW' - Exponential Loss Weighted Decoder. |

### 5.1.4 数据复杂度

ecoctlk提供6种数据复杂度的计算函数。具体如下：附录4展示了数据复杂度函数调用方式。

|  |
| --- |
| F1: Maximum Fisher’s discriminant ratio  F2: Volume of overlap region  F3: Maximal (individual) feature efficiency  N1: Fraction of points on class boundary  N2: Ratio of average intra/inter class nearest neighbor distance  N3: Error rate of 1 nearest neighbor classifier |

### 5.1.5 特征选择方法

常用Python工具包如sklearn和shogun提供了很多特征选择方法，ecoctlk工具包也提供7种独有的特征选择方法。具体如下：附录5展示了特征选择方法。

|  |
| --- |
| BSSWSS  F1  F2  F3  N1  N2  N3 |

## 5.2 本章小结

本章介绍使用Python语言编写的ECOC工具包ecoctlk。首先阐述了Python工具包普遍对ECOC功能的忽视，解释了开发ecoctlk的动机。介绍ecoctlk包含的ECOC框架类、提供了许多编码方法和解码方法。提供了独有的7种特征选择方法和6中数据复杂度测度。

与本章内容有关的代码示例放在文末的附录里。

# 第六章、总结与展望

## 6.1 总结

本文提出纠错输出编码（ECOC）的相关研究，既有针对微阵列数据特点的编码算法，也有对普通的UCI数据样本个性化指定编码矩阵的方法；既提出和解决常用解码方法的缺陷，也作者开发发布的ECOC工具包。下面分条详细介绍本文工作。

第一、二章介绍了ECOC相关理论、研究发展史；同时也介绍了微阵列数据特点、分析方法；然后简单提到市面上Python语言的ECOC工具包存在缺陷。除了以上内容，第二章还简单介绍监督学习、集成学习、特征选择的相关理论。并且详细介绍了本文使用到的7种特征选择方法。

第三章主要介绍一种常用解码方法弱欧氏距离解码（AED）存在两点不足。分析不足存在的原因，分别提出解决不足的方法。并将方法应用在自行开发的工具包中。

第四章主要介绍一种应用于微阵列数据进行编码的方法：自调节纠错输出编码算法（SA-ECOC）。这个算法针对微阵列数据“多特征、少样本、高重叠”的三个特点，设计符合特点的编码，并且利用癌症微阵列数据进行实验验证。结果证明算法能提高微阵列数据的分类准确率。

第五章主要介绍了一种个性化纠错输出编码方法（CECOC），这种方法采用固定编码，但对每个样本依据方法提出的“距离测度”来选择列编码对应的特征子集，通过这种方式生成样本的个性化矩阵。使用UCI数据对该方法进行试验，发现个性化编码矩阵比非个性化编码分类准确率更加高。

第六章介绍了作者自行开发的ECOC工具包ecoctlk，该工具包使用Python语言进行开发，不仅填补了市面上Python工具包在ECOC方面功能的缺陷，也从matlab的ECOC工具包里移植了许多编码和解码的方法，还实现并提供了一些独有的方法。

## 6.2 展望

下面分析本文三个主要工作的不足，提出改进方向。

SA-ECOC是一种迭代增加列编码的算法，每次迭代新增加的列编码是误分类的“OVO和OVA”列。这种增列方式较为草率，增加的列也相对较少，应该在新增列上深入研究。

除此以外，新增列在加入矩阵之前都有一个删除部分列的过程，设计理念类似于决策树的剪枝。但是这个删除过程与验证数据靠得太紧，容易造成过拟合。应该利用一种类似数据复杂度的测度来指导删除过程。

CECOC是一种为列编码选择特征子集的方法，用到的是文中提到的“距离测度”。该项工作的关键点在于距离测度的定义方式，文中提到的是距离测度是基于欧氏距离的测度。除此以外还有诸如曼哈顿距离、余弦复杂度之类的距离，也可以进行尝试。

作者开发的工具包ecoctlk，里面包含了许多编码和解码方法，下一步的完善工作是添加更多的编码和解码方法，同时添加自己设定的解码方法。主要可添加的方法有森林纠错输出编码（Forest-ECOC），概率解码（probabilistic-based decoding）。

# 附录

附录1：使用ECOC框架类进行多分类的示例。

|  |
| --- |
| import numpy as np from Classifiers.ECOCClassifier import SimpleECOCClassifier from Classifiers.BaseClassifier import get\_base\_clf  X = np.array(  [[.8,.6,.1],  [.3,.1,.2],  [.5,.9,.3],  [0,.5,.8],  [.5,.1,.6],  [.9,.2,.7]]) # 训练数据 X\_ = np.array(  [[.5,.3,.9],  [.7,.1,.4],  [.3,.8,0],  [.1,0,.9]]) # 测试数据  y = np.array(['A', 'A', 'B', 'B', 'C', 'C']) # 标签 M = np.array(  [[1,1,0],  [-1,0,1],  [0,-1,-1]]) # 3类OVO编码矩阵  estimator = get\_base\_clf('SVM') # 基分类器  # ECOC框架类，使用自定义编码矩阵M，解码器指定“弱欧式距离解码” sec = SimpleECOCClassifier(estimator, M, decoder='AHD') sec.fit(X, y) # 训练分类模型 pred = sec.predict(X\_) # 预测测试数据类别 print(pred)  >>> ['C' 'C' 'B' 'B'] # 输出结果 |

附录2：调用编码函数示例。

|  |
| --- |
| import numpy as np from CodeMatrix.CodeMatrix import ovo, ova  X = np.array(  [[.8,.6,.1],  [.3,.1,.2],  [.5,.9,.3],  [0,.5,.8],  [.5,.1,.6],  [.9,.2,.7]]) # 训练数据 y = np.array(['A', 'A', 'B', 'B', 'C', 'C']) # 标签 ovo\_mat, labels = ovo(X, y) # 得到OVO编码矩阵 print(ovo\_mat)  >>> [[ 1. 1. 0.]  >>> [-1. 0. 1.]  >>> [ 0. -1. -1.]] # 输出结果  ova\_mat, labels = ova(X, y) # 得到OVA编码矩阵 print(ova\_mat)  >>> [[ 1. -1. -1.]  >>> [-1. 1. -1.]  >>> [-1. -1. 1.]] # 输出结果 |

附录3：使用不同解码器解码的示例。

|  |
| --- |
| import numpy as np from Decoding.Decoder import AttenuatedEuclideanDecoder, get\_decoder  # 获得解码器的方法有两种，一种是直接调用解码器类，如：  # decoder = AttenuatedEuclideanDecoder()  # 另一种是利用代码调用解码器类，如： # decoder = get\_decoder('AED')  classes = np.array(['A', 'B', 'C']) M = np.array(  [[1,1,0],  [-1,0,1],  [0,-1,-1]]) # 3类OVO编码矩阵  # 例1：弱欧氏距离解码（软解码） Y = np.array(  [[0.6,0.2,-0.8],  [-0.4,-0.8,-0.6],  [0,0.8,-0.4],  [-1,0,0.6]]) # 分类结果 decoder = AttenuatedEuclideanDecoder() # 弱欧氏距离解码器类 D = decoder.decode(Y, M) # 距离矩阵 print('Result decoded by AED is: ', classes[D.argmin(axis=1)]) >>> Result decoded by AED is: ['A' 'C' 'A' 'B'] # 输出结果  # 例2：汉明距离解码（硬解码） Y = np.array(  [[1,0,-1],  [-1,-1,-1],  [0,1,-1],  [-1,0,1]]) # 分类结果 decoder = get\_decoder('HD')  D = decoder.decode(Y, M) # 距离矩阵  print('Result decoded by HD is: ', classes[D.argmin(axis=1)]  >>> Result decoded by HD is: ['A' 'C' 'A' 'B'] # 输出结果 |

附录4：计算数据复杂度的示例。

|  |
| --- |
| import numpy as np from DataComplexity.datacomplexity import get\_data\_complexity  X = np.array(  [[.8,.6,.1],  [.3,.1,.2],  [.5,.9,.3],  [0,.5,.8]]) # 训练数据 y = np.array([1,1,-1,-1]) # 二类数据标签  dc = get\_data\_complexity('F1') print('Score of F1 data complexity: %.4f' % dc.score(X, y))  >>> Score of F1 data complexity: 0.4063 # 输出结果  dc = get\_data\_complexity('F2') print('Score of F2 data complexity: %.4f' % dc.score(X, y))  >>> Score of F2 data complexity: 0.1250 # 输出结果  dc = get\_data\_complexity('F3') print('Score of F3 data complexity: %.4f' % dc.score(X, y))  >>> Score of F3 data complexity: 0.3333 # 输出结果 |

附录5：使用BSSWSS特征选择方法选择特征的示例。

|  |
| --- |
| import numpy as np from FeatureSelection.FeatureSelector import BSSWSS  X = np.array(  [[.8,.6,.1],  [.3,.1,.2],  [.5,.9,.3],  [0,.5,.8]]) # 训练数据 X\_ = np.array([[.5,.3,.9],[.7,.1,.4]]) # 测试数据 y = np.array([1,1,-1,-1]) # 标签  fs = BSSWSS(k=2) # BSSWSS方法，保留评分最高的两个特征 fs.fit(X, y) X, X\_ = fs.transform(X), fs.transform(X\_) print('Train data after selection:\n', X)  >>> Train data after selection:  >>> [[ 0.6 0.1]  >>> [ 0.1 0.2]  >>> [ 0.9 0.3]  >>> [ 0.5 0.8]]  print('\nTest data after selection:\n', X\_)  >>> Test data after selection:  >>> [[ 0.3 0.9]  >>> [ 0.1 0.4]]  print('\nRanking indices:\n', fs.get\_ranking\_indices())  >>> Ranking indices:  >>> [2 1 0]  print('\nfeature selected:\n', fs.get\_support())  >>> feature selected:  >>> [False True True]  print('\nfeature indices selected:\n', fs.get\_support(indices=True))  >>> feature indices selected:  >>> [1 2]  print('\nfeature scores:\n', fs.scores\_)  >>> feature scores:  >>> [ 0.18 0.29878049 0.61538462] |

# 参考文献

[1]Dietterich T G and Bakiri G. Solving multiclass learning problems via error-correcting output codes[J]. Journal of Artificial Intelligence Research, 1995, 2(2): 263-286.

[2]Dietterich T G and Bakiri G. Error-correcting output codes: a general method for improving multiclass inductive learning programs[C]// National Conference on Artificial Intelligence. 1991: 572-577.

[3]Kong E B and Dietterich T G. Error-correcting output codes corrects bias and variance[J]. Machine Learning Proceedings, 1995: 313-321.

[4]Masulli F and Valentini G. Effectiveness of error correcting output codes in multiclass learning problems[J]. International Workshop on Multiple Classifier Systems, 2000: 107-116.

[5]Crammer K and Singer Y. On the learnability and design of output codes for multiclass problems[J]. Machine Learning, 2002, 47(2-3): 201-233.

[6]Allwein E L, Schapire R E, and Singer Y. Reducing multiclass to binary: A unifying approach for margin classifiers[J]. Journal of Machine Learning Research, 2001, 1(2): 113-141.

[7]李建武, 魏海周, and 宋玉龙. ECOC多分类器实现的最小封闭球模型[J]. 计算机研究与发展, 2011, 48(suppl): 22-30.

[8]Garcia-Pedrajas N and Fyfe C. Evolving output codes for multiclass problems[J]. IEEE Transactions on Evolutionary Computation, 2008, 12(1): 93-106.

[9]Lorena A C and Carvalho A C. Evolutionary design of multiclass support vector machines[J]. Journal of Intelligent and Fuzzy Systems, 2007, 18(5): 445-454.

[10]Bautista M Á, et al. Introducing the Separability Matrix for Error Correcting Output Codes Coding[M]. Springer Berlin Heidelberg, 2011. 227-236.

[11]Bautista M Á, et al. Minimal design of error-correcting output codes[J]. Pattern Recognition Letters, 2012, 33(6): 693-702.

[12]Alpaydin E and Mayoraz E. Learning error-correcting output codes from data.[C]// Proceedings of International Conference on Artificial Neural Networks. 1999: 743-748.

[13]Pujol O, Radeva P, and Vitria J. Discriminant ECOC: a heuristic method for application dependent design of error correcting output codes[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2006, 28(6): 1007-1012.

[14]Escalera S, Pujol O, and Radeva P. Forest Extension of Error Correcting Output Codes and Boosted Landmarks[J]. 2006, 4: 104-107.

[15]Escalera S, Pujol O, and Radeva P. ECOC-ONE: A Novel Coding and Decoding Strategy[J]. Icpr, 2006, 3(3): 578-581.

[16]Escalera S, et al. Subclass problem-dependent design for error-correcting output codes[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2008, 30(6): 1041-54.

[17]Bagheri M A, Montazer G A, and Kabir E. A subspace approach to error correcting output codes[J]. Pattern Recognition Letters, 2013, 34(2): 176-184.

[18]Monti S, et al. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data[J]. Machine Learning, 2003, 52(1-2): 91-118.

[19]Peng Y. A novel ensemble machine learning for robust microarray data classification[J]. Computers in Biology and Medicine, 2006, 36(6): 553-573.

[20]Watanabe T, Kato A, and Terashima H. The PFA-AMeX method achieves a good balance between the morphology of tissues and the quality of RNA content in DNA microarray analysis with laser-capture microdissection samples[J]. Journal of Toxicologic Pathology, 2015, 28(1): 43-49.

[21]Price A, et al. Principal components analysis corrects for stratification in genome-wide association studies[J]. Nature Genetics, 2006, 38(8): 904.

[22]Donoho D and Grimes C. Hessian Eigenmaps: Locally Linear Embedding Techniques for High-Dimensional Data[J]. Proceedings of the National Academy of Sciences of the United States of America, 2003, 100(10): 5591.

[23]He X and Niyogi P. Locality preserving projections[J]. Advances in Neural Information Processing Systems, 2003, 16(1): 186-197.

[24]Ridder D and Duin R. Sammon's mapping using neural networks: A comparison[J]. Pattern Recognition Letters, 1997, 18(11-13): 1307-1316.

[25]Belkin M and Niyogi P. Laplacian Eigenmaps for dimensionality reduction and data representation[J]. Neural Computation, 2003, 15(6): 1373-1396.

[26]Cavalieri Y. Fundamentals of cDNA microarray data analysis[J]. Trends in Genetics, 2003, 19(11): 649.

[27]Kohonen T. Self-organizing maps[M]. Springer Science & Business Media, 2001.

[28]Chen Y, Bittner M, and Dougherty E. Issues associated with microarray data analysis and integration[J]. Nature Genetics, 1999(Suppl 22): 213-215.

[29]Xing E and Karp R. CLIFF: clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts[J]. Bioinformatics, 2001, 17(suppl): 306-315.

[30]Sheng Q, Moreau Y, and Moor B. Biclustering microarray data by Gibbs sampling[J]. Bioinformatics, 2003, 19(suppl): 196.

[31]Fu L and Medico E. FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data[J]. Bmc Bioinformatics, 2007, 8(1): 3.

[32]Di Gesu V, et al. GenClust: A Genetic Algorithm for Clustering Gene Expression Data[J]. BMC Bioinformatics, 2005, 6(1): 289.

[33]Tapia E, et al. Multiclass classification of microarray data samples with a reduced number of genes[J]. Bmc Bioinformatics, 2011, 12(1): 59.

[34]Yu H, Gao S, and Qin B. Multiclass microarray data classification based on confidence evaluation[J]. Genetics and molecular research: GMR, 2011, 11(2): 1357-1369.

[35]Tong M, et al. An ensemble of SVM classifiers based on gene pairs[J]. Computers in Biology & Medicine, 2013, 43(6): 729.

[36]Escalera S, Pujol O, and Radeva P. Error-Correcting Output Codes Library[EB/01]. <http://jmlr.csail.mit.edu/papers/v11/escalera10a.html>. 2010.

[37]MathWorks. fitcecoc[EB/01]. <http://cn.mathworks.com/help/stats/fitcecoc.html>. 2017.

[38]Scikit-learn. sklearn.multiclass[EB/01]. <http://scikit-learn.org/stable/modules/classes.html#module-sklearn.multiclass>. 2017.

[39]Shogun. Shogun[EB/01]. <http://www.shogun-toolbox.org/api/latest/classes.html>. 2017.

[40]Lomax R G and Hahs-Vaughn D L. Statistical Concepts: A Second Course[M]. Vaughn, 2011.

[41]邓彩凤. 中文文本分类中互信息特征选择方法研究[M]. 2011.

[42]Lorena A C, et al. Analysis of complexity indices for classification problems: Cancer gene expression data[J]. Neurocomputing, 2012, 75(1): 33-42.

[43]王金桃, 周利锋, and 高尔生. 第六讲 卡方检验[J]. 实验动物与比较医学, 2000(4): 251-254.

[44]Scikit-learn. Extra Tree Classifier[EB/01]. <http://scikit-learn.org/stable/modules/generated/sklearn.tree.ExtraTreeClassifier.html>. 2017.

[45]Liu K-H, Zeng Z-H, and Ng V T Y. A Hierarchical ensemble of ECOC for cancer classification based on multi-class microarray data[J]. Information Sciences, 2016, 349-350: 102-118.

[46]Windeatt T. Weighted Decoding ECOC for Facial Action Unit Classification[M]. Springer Berlin Heidelberg, 1970. 59-77.

[47]Escalera S, Pujol O, and Radeva P. On the decoding process in ternary error-correcting output codes[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2010, 32(1): 120.

[48]Cano J. Analysis of data complexity measures for classification[J]. Expert Systems with Applications, 2013, 40(12): 4820-4831.

[49]Koboldt D, et al. Comprehensive molecular portraits of human breast tumours[J]. Nature, 2012, 490(7418): 61.

[50]Shipp M A, Ross K N, and Tamayo P. Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning[J]. Nature Medicine, 2002, 8(2002): 68-74.

[51]Su A, et al. Molecular classification of human carcinomas by use of gene expression signatures[J]. Cancer Research, 2001, 61(20): 7388.

[52]Ben-Dor A, et al. Tissue classification with gene expression profiles[J]. Journal of Computational Biology A Journal of Computational Molecular Cell Biology, 2000, 7(3-4): 559.

[53]Hong Z and Yang J. Optimal discriminant plane for a small number of samples and design method of classifier on the plane[J]. Pattern Recognition, 1991, 24(4): 317-324.

[54]Khan J, et al. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks[J]. Nature Medicine, 2001, 7(6): 673.

[55]Kuncheva L I and Rodrguez J J. Classifier ensembles with a random linear oracle[J]. IEEE Transactions on Knowledge & Data Engineering, 2007, 19(4): 500-508.

[56]Irvine U o C. UCI Machine Learning Repository[EB/01]. <http://archive.ics.uci.edu/ml/datasets.html>. 2017.

[57]Repository U M L. Multiple Features Data Set[EB/01]. <http://archive.ics.uci.edu/ml/datasets/Multiple+Features>. 2000.

[58]Repository U M L. ISOLET Data Set[EB/01]. <http://archive.ics.uci.edu/ml/datasets/ISOLET>. 1994.

[59]Repository U M L. Statlog (Landsat Satellite) Data Set[EB/01]. <http://archive.ics.uci.edu/ml/datasets/Statlog+%28Landsat+Satellite%29>. 1993.

[60]Repository U M L. Wearable computing weight lifting exercises biceps curl variations[EB/01]. <http://archive.ics.uci.edu/ml/machine-learning-databases/00273/>. 2013.

[61]Repository U M L. Pen-Based Recognition of Handwritten Digits Data Set[EB/01]. <http://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits>. 1998.

# 致谢

本文是在我的导师王备战老师的帮助下完成，王老师提供给我实验场地和实验设备，还有读硕士研究生的机会，更重要的是王老师教导我做科研“大胆假设小心求证”，也吩咐我要专心和耐心。王老师平时既威严又亲切，学生对他很信服，他也愿意分享他内心的思考。在此对王备战老师表示感谢。

另一位需要重点感谢的是指导科研的刘昆宏老师，刘老师将我引入了纠错输出编码的研究，教给我非常丰富的知识和技巧。我一直记得刘老师教育我：“实验效果差不要紧，慢慢改进，底线是不要作假。”刘老师对教学生非常负责，关于研究细节的讨论，我与刘老师经常一聊就是两个小时。在此感谢刘老师，希望他身体一直健康，生涯顺利。

然后感谢孙蒙新学妹、王寒蕊学妹、冯凯杰学弟、李克森学弟。蒙新和我讨论ECOC发展史和撰写技巧；寒蕊教我遗传算法编码思想，讨论特征选择方法；凯杰和我讨论随机编码最优长度；克森与我讨论AED解码的弊端与解决。几位学弟学妹都对我的论文提供了帮助，非常感谢。

感谢史亮老师、葛学凤老师和麻琳键老师对本文提出的建议。也感谢身边所有帮助过我的人，一同成长的人。