北京交通大学

硕士学位论文

多分类器组合中的基分类器选取方法

姓名：付彬

申请学位级别：硕士

专业：计算机科学与技术

指导教师：王志海

20090601

中文摘要

在数据挖掘领域中，分类是一种非常重要的技术．然而，现有的多种分类技术如贝叶斯，决策树等都是单分类器技术．目前单分类器性能的提升已经达到了一个瓶颈，人们遂提出了多分类器组合的概念．多分类器组合使用多个基分类器进行分类，并综合所有分类结果形成一个最终结果．实验表明，多分类器组合能显著提高分类器的分类性能．因此对其进行研究，具有重要的理论价值及现实意义．

本文首先对多分类器组合的各个主要研究方向做了综述性的阐述．包括多分类器组合的概念，构建多分类器组合的关键问题，例如器组合的多样性评价，基分类器的生成策略，基分类器的选取，以及基分类器的组合方法等相关问题的基本概念和典型算法．

在已有研究成果的总结上，本文针对多分类器组合的基分类器选取问题提出了两种尝试性的算法．首先，如何利用多样性指导组合的生成是研究的一个重要问题．本文提出了一种 Boosting 方法与多样性评价指标相结合的基分类器选取方法．进而，本文指出了传统的 Boosting 方法存在的不足，提出了一种新的利用当前测试实例来动态地对基分类器进行权重赋值的方法，试图克服传统 Boosting 方法中静态权重赋值所可能带来的缺陷．最后，介绍了数据挖掘平台 Weka 系统的概况及其架构，在此平台上实现上述提出的方法，并通过实验比较了上述方法，与传统的多分类器组合算法如 Boosting 算法，及经典分类器算法如决策树分类器的分类性能．实验结果与分析表明，本文提出的这些算法在大部分数据集上都具有更好的分类性能．

关键词：数据挖掘；分类器；多分类器组合；基分类器选取；动态权重赋值

分类号：TP301.6

ABSTRACT

In the field of data mining, classification has always been a most important technology. However, many classification techniques that have been introduced are of single classifier method. Because the improvement of single classifier's accuracy has reached the bottleneck, so people propose the concept of the combination of multiple classifiers, which means using multiple classifies to classify the instance and combine every classifier's result to give a final result. Experiments show that combination can improve the classifier's performance obviously. Therefore, couducting research on classifiers combination is of significant theoretical value and practical benefit.

Firstly, this paper gives a comprehensive description of every research direction relating to classifiers combination, including basic theory and representative methods. The issues included are what is combination and some critical questions, such as the diversity of combination, the strategy of creating base classifier, how to select base classifiers and how to combine the choosed base classifiers.

Based on the existed reseach, this paper proposes two particular algorithms about selection of base classifiers. Firstly, how to use diveristy to create combination becomes a research focus. A new selection method using diversity is proposed firstly based on the Boosting method. Furthermore, this paper analyzes the boosting method and point out its potential disadvantage, and proposes another new method that can dynamically allocate weights to base classifiers using current test instance, in order to overcome the defect of static weight allocation in boosting method. Finally, this paper describes weka, the platform used in experiment. Then it implements the algorithms mentioned above and gives a detail on the original implementment of meta-learning in weka. After that, this paper gives the experiment results of the algorighms and some classical methods and compares the difference between those results. The results show that the algorithms proposed in this paper have better performance in majority of the dataset.

KEYWORDS: Data Ming; Classifier; Combination of Classifiers; Selection of Classifier; Dynamic Weighting.

CLASSNO: TP301.6

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京交通大学或其他教育机构的学位或证书而使用过的材料．与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意．

学位论文作者签名： 签字日期： 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北京交通大学有关保留、使用学位论文的规定．特授权北京交通大学可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅．同意学校向国家有关部门或机构送交论文的复印件和磁盘．

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：付彬　　导师签名：王志涛

签字日期：2009年6月19日　　签字日期：2009年6月16日

致谢

本论文的工作是在我尊敬的导师王志海教授的悉心指导下完成的，从论文的选题，论文的撰写，一直到论文的最终定稿，王老师一直给予我细心的指导和无尽的关怀．王志海教授严谨的治学态度和科学的工作方法给了我极大的帮助和影响．在此衷心感谢三年来王志海老师对我的关心和指导．

衷心感谢黄厚宽和田盛丰教授，两位教授宽广豁达的长者风范、以及严谨的治学态度始终让我深深地敬仰．他们在此期间对我的关心和鼓励让我深受感动．

瞿有利老师对于我的科研工作和论文都提出了许多的宝贵意见，在此表示衷心的感谢．

在实验室工作及撰写论文期间，冯浩、王世强、邵鲁杰、邵进智等同学对我论文研究工作给予了热情帮助，在此向他们表达我的感激之情．

另外也感谢我的父母，他们的理解和支持使我能够在学校专心完成我的学业．

最后，衷心地感谢在百忙之中审阅论文的各位老师和专家，恳请各位老师多多批评指正，并提出宝贵的意见．

1 引言

随着当前计算机技术的广泛应用，我们已经能够捕获和存储大量的数据．然而，人们对数据的掌握了解速度却永远赶不上数据升级的速度．当人们正在被海量数据所淹没时，对数据的理解程度却在降低，对大量数据中潜在的重要的知识不能充分的发掘出来并有效利用．数据的利用率太低降低了数据的有用性，并使数据成为了数据“坟墓”．因此，如何将大量数据转换为有用的信息和知识成为当前业界一个巨大的挑战和研究热点．机器学习(Machine Learning)与数据挖掘(Data Mining)就是研究从数据中提取知识的理论和技术[1,2]，这些理论与技术已经在实际生产领域中开始得到应用，并有着巨大的潜在利用价值．

数据挖掘是指采用模式识别，统计等技术通过分析大量数据来发现有用的相互关系、模式和趋势并且以简单的数据模型归纳之的过程．针对要发现的模型的不同，数据挖掘技术主要可以分为挖掘频繁模式，分类和预测，聚类分析等主要方向[3]．

1.1 研究背景

近二十年来，研究人员已经建立并发展了许多机器学习与数据挖掘的理论体系，并开发了许多实用技术，其中分类是最重要的研究问题之一．当前针对分类问题已提出了多种分类模型，如贝叶斯方法(Bayesian Method)、决策树(Decision Tree)和k-最近邻等[2,3,4]．他们都是通过利用分类模型对已知类别的训练数据集进行学习来得到一个单一的分类器，来对类别未知的数据进行分类．另外，在具体的分类任务下，面对众多的分类器如何评价和比较分类器也是一个主要研究点．当前提出的评价方法有分类错误率，分类成本，最短描述长度等[3,5]，其中分类错误率是最基本的评价方法．分类错误率测量的是分类器在训练集或测试集上分错的实例的与所有实例的比，错误率低的分类器被认为更适合当前的分类问题．然而研究表明，并没有一种分类器可以在所有的分类问题上都有较低的错误率，在存在噪声数据等情况下分类器甚至会产生糟糕的分类性能．因此，研究如何降低分类器的错误率或按其他分类器评价标准来指导分类器的设计成为了一个受关注的研究问题．

为了降低分类器的错误率，研究人员提出了多分类器系统 (Multiple Classifier System)的理论，并已经提出了多种组合方法．例如装袋法(Bagging)、提升法(Boosting)和随机子空间法(Random Subspace)[6,7,8]．多分类器组合是组合多个分

类器对实例进行分类的系统，其中每个分类器被称为基分类器．在分类阶段，每个基分类器都参与对测试实例的分类，然后运用某种组合方法，综合所有基分类器的分类结果以形成一个最终分类结果．因此多分类器组合主要包括两个问题：一是如何产生多个基分类器；另一个是如何组合这些基分类器[9]．分类器组合是建立这样的假设上的，即在分类时每个分类器所做的决定是独立的，不同的．直观上可以看出，这样即使有分类器做出错误的分类，其他的分类器也会纠正这个错误．

通过实验观察，分类器组合方法能显著地降低分类器的错误率．关于多分类器组合为什么能够降低分类错误率，Dietterich 从统计的(Statistical)、计算的(Computational)和代表性的(Representational)三个角度说明多分类器模型能更接近最优的分类模型[10]．Kohavl 将分类错误率划分成方差和偏差[11]．研究认为分类错误率的减少可以通过减少方差和偏差来实现，且分类器组合能显著的减小方差[12]．经过多年的研究发展，进一步提升传统的单一分类器的分类性能变得十分困难．因此多分类器组合成为了当前研究和应用的一个重点．

1.2 本文所完成的工作

在数据挖掘领域中，分类是一种非常重要的技术，在金融、证券、科学、工程等领域有着广泛的应用．

多分类器组合技术是将多个不同的单分类器组合成一个分类器，组合的目的是利用多个分类器的差异来改善最终分类器的分类性能．Boosting 方法是一种应用广泛的多分类器组合方法．该方法中的一个典型算法是 AdaBoost 算法，它能够较显著的提高基分类器的分类性能而且容易实现．然而 Boosting 算法中是静态地对基分类器进行权重赋值，即在测试实例到来之前就已完成对基分类器的权重赋值，这样就可能给一个对当前测试实例会错误分类的基分类器赋以较高权重，降低了组合的分类正确率．

另外，在生成多分类器组合后，如何评价组合的质量来预测其在未知数据上的性能是一个重要研究问题．除了传统的正确率，错误率等方法外，组合的多样性是一种公认的多分类器组合评价标准．然而，关于多样性的定义并没有一个统一的标准，人们提出了多种多样性的度量方法并通过实验比较它们．另外，在确定了多样性的度量方法后，如何在多分类器组合的生成过程中利用多样性，使得最终的组合质量更好也是一个重要研究问题．当前存在着多种利用多样性从基分类器集合中选择，评价基分类器子集以产生组合的方法．但传统的穷举法或其他的启发式算法通常时间复杂度较高或者比较简单，因此研究其他的利用多样性对

基分类器进行选择也是一个重要的研究问题.

在上述背景之下，本文完成的工作是：

(1) 首先，介绍分类的基本概念以及相关技术.

(2) 其次，叙述了多分类器组合的基本概念及其各个关键问题. 包括：多分类器提高分类精度的原因，组合多样性的定义和应用，基分类器的生成策略，基分类器的选取策略等多分类器组合研究中的各关键问题.

(3) 第三，在分析了如何利用多样性指导基分类器选取的基础上，提出了一种新的多样性测量方法，并利用贪心策略结合 Boosting 方法提出了一种新的选取方法. 该方法首先使用了 Boosting 方法生成多个基分类器，然后利用多样性从中选取了一个多样性最高的子集来形成最后的组合.

(4) 第四，在详细分析了 Boosting 方法的基础上，指出了其静态权重赋值的潜在问题，提出了一种利用动态选取思想的动态权重赋值的方法. 跟静态权重赋值相比，该方法能够根据当前测试实例属性取值的不同，按照其被每一个基分类器可能正确分类的可能性大小而对其动态分配权重. 基分类器的权重随着当前测试实例的变化而变化，从而保证了对当前测试实例可能分类精度最高的分类器具有较大的权重.

(5) 进而，讨论了著名的数据挖掘开源平台 Weka 系统，着重分析了 Weka 中元学习的相关内容以及实现.

(6) 最后，并通过实验比较了上述两种新提出的多分类器组合方法，传统的多分类器组合 Boosting 及经典的决策树分类器. 最终得出结论，提出的新算法在大多数数据集上可以提高原有分类器的分类性能.

1.3 论文的组织安排

本文的主要框架和结构如下：

第 1 章给出了课题的出发点以及研究的问题及范围，叙述了数据挖掘产生的背景以及数据挖掘相应的分析方法，分析了多分类器组合的研究现状，介绍了本文所完成的工作.

第 2 章介绍了多分类器组合的理论和相关知识，包括多分类器组合的基本概念，多分类器组合能提高分类精度的分析，以及多分类器组合研究中的各个关键问题.

第 3 章在详细分析了研究所采用的基分类器及 Boosting 方法的基础上. 提出了在 Boosting 方法的基础上利用贪心策略结合多样性来对基分类器进行选择的一种基分类器选取方法的具体实现.

第 4 章在分析了 Boosting 算法的基础上，指出了其静态权重赋值可能带来的问题，提出了被错误分类子集的概念，并在其基础上提出了一种动态权重分配的方法.

第 5 章首先对 weka 平台下进行了介绍，介绍了 Weka 中原学习部分的相关情况. 接着叙述了实验方法，描述了实验数据集，并在 Weka 平台下进行实验比较了本文所提出的几种方法和传统的分类器组合方法以及经典的决策树分类器. 给出了本文的实验过程和结果，最后对结果进行比较和分析.

第 6 章总结全文，对本课题研究做了分析和总结，分析了算法的不足之处，并给出了本课题将来的研究内容和方向.

2 多分类器组合技术综述

数据是知识的基本载体．人们往往通过对数据的收集，分析来获得所需的知识并用于指导当前的生产活动．近十几年，随着科学技术飞速的发展，经济和社会都取得了极大的进步．与此同时，随着当前计算机技术的广泛应用，在各个领域产生了大量的数据．然而，人们对数据的掌握了解速度永远赶不上数据升级的速度．当人们正在被海量数据所淹没时，人们对数据的理解程度却在降低，对大量数据中潜在的重要的知识不能充分的发掘出来并有效利用．于是人们迫切需要一种能够从这些数据中自动抽取知识的学习方法，能够将大量数据转换为有用的信息和知识．在超大规模数据库的出现、计算机技术的迅猛发展和较难的统计方法的运用等因素激发下，数据挖掘这门新兴科学应运而生，成为当前研究和应用的一个热点．

数据挖掘技术就是在上述背景下产生的一种用于从数据中获取知识的新技术．数据挖掘是指采用模式识别，统计等技术通过分析大量数据来发现有用的相互关系、模式和趋势并且以简单的数据模型归纳之的过程．针对要发现的模型的不同，数据挖掘技术主要可以分为挖掘频繁模式，分类和预测，聚类分析等主要方向．其中，分类是数据挖掘技术的一个重要分支．

2.1 分类的概念

分类是数据挖掘一项十分重要的任务，当前研究人员已经提出了众多的关于分类的基本理论和技术，并在实际中获得了比较广泛的应用．

分类是指预测数据所属的类标号．这里，数据集中每一条数据都属于某一个离散无序的类别．其通过分析训练集中的数据，为每个类别建立分类模型，然后利用这个分类模型对其他未知类别的数据进行分类．分类的目的是学会一个分类函数或分类模型（也常常称作分类器），该模型能够把数据映射到给定类别集合中的某一个类别．因此，本质上分类器就是一个从数据到类标的映射[13]．因此分类可以被定义为：给定一个数据集 $D=\{x_1, x_2,\ldots,x_n\}$和一组类 $C=\{c_1, c_2,\ldots,c_k\}$，分类问题是去确定一个映射：$f$: $D\rightarrow C$，使每个实例 x_i 被分配到一个类中．一个类 c_j 包含映射到该类中的所有实例，即 $c_j=\{x_i|f(x_i)=c_j, 1\le i\le n\}$且 $x_i\in D$.

分类与预测是两个不同的技术．两种方法都是数据挖掘中对数据进行分析的方法．但分类不能等同于预测，两者具有相同点和不同点．相同点是两者都需要先构建模型，然后用构建的模型来估计未知值．二者的区别在于分类主要是用于

预测离散的类标签，而预测则是用来估计连续值.

在分类中，数据通常可以分为训练集和测试集．训练集用于构造分类器，测试集用来评估该分类器的分类精度．训练集和测试集一般采用不同的数据，以防止产生过于乐观的估计．分类的过程由以下两个步骤组成[13].

第一步是生成阶段，生成阶段是为了在训练数据集合上生成一个用于分类的分类器，通常这一阶段也被称作为学习阶段．训练数据集就是用来建立分类模型所使用的训练数据的集合．训练数据集中的单条训练数据称作训练样本．分类算法通过分析训练样本的取值分布及特征来构造分类器．其中每一条训练例都属于其中一个预定义的类，由一个称作类标签属性的属性确定．由于提供了每个训练样本的类标号，所以这一步也被称作有监督的学习，即模型的学习在被告知每个训练样本属于哪个类的“指导”下进行．它不同于另外一种无监督的学习，也称作聚类，此时每个样本的类标签是未知的，要学习的类的个数或者集合也可能事先不知道，也就是没有类标签作为学习的“指导”．通常学习模型用决策树，分类规则或其他数学公式形式来表示．学习模型使用的形式不同，学习出来的分类器性能也有差别.

要注意的事，分类模型和分类器并不是一个概念．分类器的产生依赖于两个条件：一个是分类模型，另一个是训练数据集，两者缺一不可．如果只存在模型，而没有数据来训练它，只能称之为分类器算法，当有了训练数据进行训练学习之后，才能够产生用于对新数据进行分类的分类器．同时，仅仅有训练数据集，但是没有模型，也不可能产生分类器，没有模型计算机无法从训练集中产生分类规则对新数据进行分类，也就是说分类器建立在分类模型之上．另外，虽然不同模型决定了产生的分类器的分类性能，但是对于同一种模型，分类器还和训练集有关，训练集中训练数目的多少以及属性个数的多少，同样决定了分类器的分类性能．对于某些模型，如果训练例数目过少分类器性能可能很差，而有些模型对于训练例属性个数较多时产生的分类器性能同样较差．因此，分类器是模型和训练数据集的产物，受模型和训练数据的影响.

第二步是使用模型进行分类．保持法是一种使用类标号样本已知测试集的简单方法，这些样本随机选取并独立于训练样本．分类正确率是评估模型的一个重要指标．模型在给定测试集上的准确率是被模型正确分类的测试样本的百分比．对于每个测试样本，将已知的类标签与通过学习模型预测的结果相比较．如果模型的准确率是根据训练数据集评估，那么评估的结果可能是乐观的，因为学习模型倾向于过度拟合数据．因此，一般使用交叉验证法来评估模型．如果认为模型的准确率可以接受，就可以用它来对类标签未知的数据元组或对象进行分类.

总之，我们可以把分类归结为模型建立和使用模型进行分类两个步骤．其实，

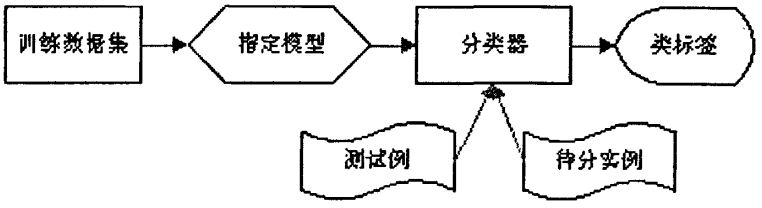

图2.1 分类过程模型
Figure 2.1 the model of classification process

模型建立的过程就是使用训练数据进行学习的过程，使用模型分类过程就是对类标签未知的数据进行分类的过程．该过程如图 2.1 所示．

近二十年来，许多研究人员已经建立并发展了许多分类的理论体系，并开发了许多实用技术．当前已提出了多种分类模型，如贝叶斯方法、决策树方法、*K*-最近邻方法[2,3,4]等．它们都是通过利用分类模型对已知类别的训练数据集进行学习来得到一个单一的分类器，进而对类别未知的数据进行分类．然而实验表明，并不存在一个分类器对所有的分类问题都是最优的．这就迫使我们在面对众多不同的分类器时，思考如何对其进行评价，并针对当前问题选择一个最优分类器．由于寻找最优分类器是一个难点，研究人员提出多分类器组合概念．组合利用了其中每个分类器的信息，来模拟当前问题的最优分类器．另外，单分类器的分类性能的提升也已经到了一个瓶颈，很难再有明显的提升．而实验表明多分类器组合与单分类器相比，能明显的提升分类性能．因此，对多分类器组合的研究和应用已经成为当前的一个研究热点．本章以下部分将结合现有的理论、概念和近些年来的典型方法等对多分类器组合的基本概念，基本理论和基本问题进行阐述．

2.2 多分类器组合的基本概念

经过多年的研究与实践，单分类器的分类精度的提升已经达到了一个瓶颈，很难再有大的提升．正是在这种背景下研究人员才提出了多分类器组合的方法．本节对多分类器组合的基本概念和关键问题进行了综述，介绍了多分类器组合的基本概念、理论、研究中的关键问题和当前的典型方法．

2.2.1 什么是多分类器组合

多分类器组合也被称为分类器集成，是为了提高分类性能尤其是分类正确率而提出的一种技术．顾名思义，多分类器组合是一个包含了多个不同的分类器的组合，其中每个分类器被称为基分类器．在给定训练集合后，单分类器方法仅仅训练出一个分类器来执行分类任务．而多分类器则通过某种方法训练出多个不同的基分类器．在分类阶段时，每个基分类器都参与对测试实例的分类并给出一个分类结果，然后按照某种方法（比较典型的如投票法）组合这些分类结果并给出一个对测试实例的最终分类结果．由上述可看出，多分类器组合实际上仍然是一个从实例到类标的映射．设训练实例集合为$\{(x_1, y_1), \ldots, (x_n, y_n)\}$，$x_n$表示第$n$个训练实例，$y_n$表示其所属的类标．多分类器组合为$D=\{D_1, D_2, \ldots, D_n\}$，其中$D_n$为第$n$个基分类器，其中每一个基分类器都是一个从实例到类标的映射：$D_n(x) \rightarrow y$．则多分类器组合则代表这样的一个映射即：$D_{1,2,\ldots,n}(x) \rightarrow y$，即使用了某种方法综合了多个基分类器的映射结果形成一个最终的映射．

由上述的概念可以看出，多分类器组合主要包含两个关键的方面：一个是基分类器的生成阶段，即如何生成多个不同的基分类器；另外一个方面是组合阶段，即如何使用基分类器来对测试实例进行分类，并组合它们的分类结果来形成一个最终的分类结果．

2.2.2 为什么多分类器组合能提高分类性能

实验表明，多分类器组合与单分类器相比，能显著地提升分类算法的分类精度．多年来，关于为什么多分类器组合能更显著地提高分类精度，人们给出了多种解释．通常情况下，多分类器组合比其中任何一个基分类器分类精度都高的一个必须且充分的条件是：基分类器是准确且多样的[10]，基分类器是准确的是指该分类器在面对新的测试实例时，分类错误的概率应该比随机猜想的要小，也就是小于0.5；基分类器是多样的指在对测试实例分类，尤其是当分类错误时，各基分类器的分类结果应该尽可能是不同和独立的．直观上可以看出，这样即使有分类器做出错误分类，其他分类器也会纠正这个错误．为了解释为什么准确性和多样性是必要的，我们假设有一个包含三个基分类器的组合：$\{c_1, c_2, c_3\}$和一个新的测试实例x．如果三个分类器是相同的，那么当$c_1(x)$是错的话，$c_2(x)$和$c_3(x)$就都是错的．这样在采用投票法的前提下，组合的最终分类结果肯定就是错的．然而当这三个分类器是多样的时，例如当$c_1(x)$分类错误时，而$c_2(x)$和$c_3(x)$却是正确的，那么利用投票法组合的最终结果就是正确的．更精确地来讲，如果每一个基分类器的分类错误概率都为$p<0.5$，如果使用投票法组合分类器，那么只有在超过一半的基分类器分类错误时，分类器组合的分类结果才会错误．即分类器组合的分类错

误概率为 $C_n^m p^m (1-p)^{n-m}$ (其中 $m>n/2$)，显然这个概率值的大小要小于 p，所以其分类精度要大于任何一个单一的基分类器．当然当基分类器的分类错误概率大于 0.5 时，多分类器组合的分类错误率就会增加．因此，构建多分类器组合的一个关键就是：其包含的基分类器的错误率要小于 0.5，并且尽可能是多样的．

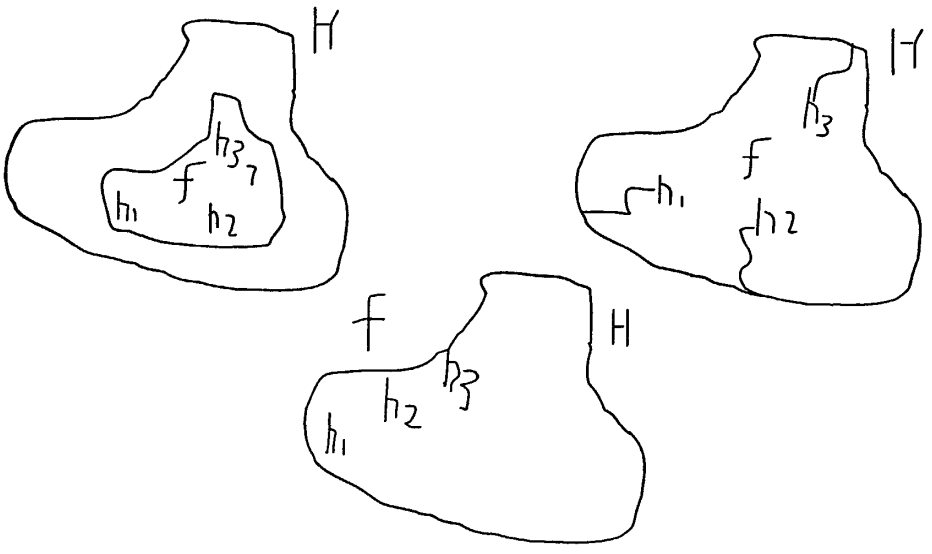

图2.2 为什么多分类器组合能比单分类器性能更好的三种原因

Figure 2.2 Three reasons why an ensemble can work better than a single classifier

那么到底能不能在实践中构建一个性能良好的多分类器组合呢？Dietterich 总结了以下三种原因，来说明在实际中为什么能构建一个好的多分类器组合．

(1) 第一个原因是从统计角度上出发的．一个学习算法实际上可以看作在一个假设空间 H 中搜索一个最好的，跟实际假设 f 最接近的一个假设．然而由于通常情况下的训练数据量相对于整个假设空间是不充分的，所以通常找到的在当前数据集合上比较准确的假设很可能并不接近真正的假设．通过组合多个这样的假设，学习算法就可以“平均化”它们的结果，减少错误分类的概率，最后得到的最终假设可以更加接近真实的假设 f，如图 2.2 左上假设空间所示．

(2) 第二个原因是从计算的角度上出发的．许多实际中的学习算法都是通过在局部假设空间中进行搜索来工作，这样它们就可能陷入局部最优的缺陷中．例如，神经网络算法使用了梯度下降来减少其在训练集上的分类错误率，决策树算法使用了贪心分裂规则来生成决策树．这样即使训练集合非常充分，分类算法也只是在有限的局部假设空间内搜索，很难得到最好的假设．而通过组合多个分类器，

每个分类器在一个不同的局部空间内，可以从空间的不同点开始搜索最优假设．这样，就有可能比任何一个单分类器搜索到一个跟真实的假设 f 接近的假设．如图2.2右上假设空间所示．

(3) 第三个原因是从表示的角度出发的．在大多数的学习算法的实际应用时，搜索空间 H 中的任何一个假设可能都不能用来表示真实的假设 f．而通过组合多个从假设空 H 间得到的假设，有可能会扩展假设空间，得到一个空间之外的假设来更能接近实际的假设 f．如图2.2下方的假设空间所示．

2.2.3 构建多分类器组合要解决哪些问题

在上述两节中，已经给出了多分类器组合的概念和两个关键步骤．而且，构造性能良好的多分类器组合的一个必要且充分的条件是：基分类器应该是准确且多样的．因此，根据建立多分类器组合的步骤过程，目前在多分类器组合方面的研究问题大致可以从以下几个方面来分析．

(1) 多分类器组合的评价问题

当生成了组合后，应当如何评价该组合的质量，从而推测其是否具有较好的分类性能．当前存在着多种评价标准，那应该如何使用这些标准来评价组合？这些评价标准之间的关系是什么？更进一步，如何利用这些评价标准来指导多分类器组合的构建过程，使构建出来的组合具有良好的性能．另外，组合的多样性是当前公认的影响多分类器组合分类性能的一个重要因素，但目前人们在多样性的定义，如何度量多样性等问题上还没有达成共识[14]．因此如何度量多样性，如何利用多样性来评价，指导多分类器组合的产生是一个重要的研究方向．

(2) 基分类器的生成

多分类器组合是要组合多个基分类器，因此首先要考虑的问题是如何生成多个不同的基分类器，并且各基分类器应该尽可能是准确且多样的．当前已经存在了多种生成方法，不同的生成方法产生的组合有着不同的分类性能．这些分类性能上的差异到底是由什么原因引起的，是否存在一个标准来指导生产方法的设计，使产生的组合具有较好的性能？

(3) 基分类器的选取问题

当生成多个基分类器后，传统的方法只是简单地使用了所有的基分类器来参与分类，并未对基分类器进行筛选．但通常情况下有两个问题：一是当生成过多的基分类器时，使用所有的基分类器来分类是一个非常耗时的过程；另一个更重要的问题是，使用经过某种方法挑选出来的基分类器子集形成的组合，跟使用全部的基分类器相比，可能具有更好的分类性能．于是，人们提出了“过量产生—

选取”的方法来形成多分类器组合[15]. 过量产生基分类器阶段，可以使用已有的多种方法实现，而如何从大量的基分类器中选择一个子集来形成分类性能良好的组合，就成了研究中的一个关键问题. 选取问题就是在基分类器空间中进行搜索. 因此该问题主要包含两个关键点：一个如何搜索的目标函数的确定，即以什么样的标准评判搜索到的基分类器；另外一个是搜索方法，即如何在基分类器空间内进行搜索，使最终得到的基分类器组合在当前的目标函数下是最优的. 另外，并不是对所有的测试实例，当前的基分类器组合对其都有良好的分类性能. 因此还必须考虑到当前测试实例，根据其各属性的具体值动态地选择合适的基分类器形成组合，已达到更好的分类性能.

(4) 基分类器的组合方法

在确定了组合中包含的基分类器后，面对测试实例时如何组合基分类器的分类结果来形成对一个对实例类标的最终预测是研究的另一个重要问题. 当前存在着多种组合方法，如投票法等[16]. 那么这些组合方法间存在着什么关系？存不存在一种组合方法在任何情况下都比其他组合方法好？不同的组合方法和不同的生产方法之间有什么影响？是否存在一个通用的理论基础来指导新的组合方法的产生？此外，投票法等组合方法的前提假设是各个分类器之间是独立的，而在实际的分类算法中可能不能满足这个条件，那如何设计新的组合方法使其能不受该假设的限制等等，都是需要解决的问题.

以上部分介绍了多分类器组合的基本概念和研究的主要问题，在本章接下来的部分将分别按照上述的几个研究方面，对当前已存在的理论基础和代表性的算法进行逐一阐述.

2.3 多分类器组合的评价标准—多样性的度量

多样性被认为是制约多分类器组合性能的一个关键因素. 组合的多样性是指在对测试实例进行分类时，不同的分类器的分类结果是独立且不相同的. 这样即使有分类器做出错误的分类，其他的分类器也会纠正这个错误，这样就有可能提高分类器的分类性能. 然而，当前关于多样性并没有一个严格，统一的定义[14]. 另外，在构建多分类器组合中基分类器的正确率和组合的多样性这两个因素是互相制约的. 如果所有基分类器都有较高的正确率，分类器的分类趋向于统一，多样性就低；如果无限制的提高多样性就会降低基分类器的正确率，从而影响组合的正确率. 因此当前关于多样性研究的主要问题包括：

(1) 如何定义和度量多样性.

(2) 如何确定组合多样性和组合的正确率之间的关系.

(3) 如何平衡组合的多样性和基分类器的正确率之间的关系.

(4) 如何利用多样性指导多分类器组合的生成.

2.3.1 多样性的定义及度量

由于并不存在一个统一明确的多样性定义及度量方法，许多研究已经给出了众多的度量方法，这些定义大都是基于基分类器正确或错误的分类结果的. 设 $Z=\{z_1, z_2,\ldots,z_N\}$ 为已标明类别的数据集, N 为实例个数; $D=\{D_1, D_2, \ldots, D_L\}$ 为组合中的基分类器集合，L 为基分类器个数; $Y_i=\{y_{1,}, \ldots, y_{n,i}\}$ 为第 i 个分类器的分类结果，如果基分类 D_j 正确分类实例 z_i 则 y_{ij} 为 1，否则为 0; l_i 表示对实例 z_i 正确分类的基分类器个数. 另外，利用分类结果可以定义两个基分类器的相似度，如表 2.1 所示.

表 2.1 两个分类器之间的分类结果

Fig. 2.1 the result of classification between two classifiers

	被 D_j 正确分类	被 D_j 错误分类
被 D_i 正确分类	N_{11}	N_{10}
被 D_i 错误分类	N_{01}	N_{00}

Kuncheva 和 Tang 等人已经分别对当前的多样性度量方法进行了总结[14,17]，主要包括以下几种度量方法.

(1) Q 统计量

这种统计方法是由 Yule 提出的，如公式 2.1 所示.

$$Q_{i,j}=\frac{N^{11}N^{00}-N^{01}N^{10}}{N^{11}N^{00}+N^{01}N^{10}} \tag{2.1}$$

Q 统计量用来测量两个分类器之间的相似度，由公式可以看出，Q 的大小与两个分类器分类结果相同个数的大小成正比，并且在-1 和 1 之间变化. 对于独立的基分类器来说，度量的期望值应该是 0. 对整个分类器组合的 Q 统计量是所有两两基分类器的 Q 统计量的均值，如公式 2.2 所示. 当 Q 统计量大时，说明基分类器间的相似程度比较高，即组合的多样性比较低.

$$Q_{av}=\frac{2}{L(L-1)}\sum_{i=1}^{L-1}\sum_{j=i+1}^{L}Q_{i,j} \tag{2.2}$$

(2) 协方差 P

对于两个基分类器来说，协方差 P 的度量如公式 2.3 所示.

$$P_{i,j}=\frac{N^{11}N^{00}-N^{01}N^{10}}{\sqrt{(N^{11}+N^{10})(N^{01}+N^{00})(N^{11}+N^{01})(N^{10}+N^{00})}} \tag{2.3}$$

协方差 P 与上述的 Q 统计量比较类似，也是用来统计两个基分类器之间的相似度的. 对整个分类器组合的协方差 P_{av} 是所有两两基分类器的协方差 P 的均值，如公式 2.4 所示. 当协方差 P 大时，说明基分类器间的相似程度比较高，即组合的多样性比较低.

$$P_{av}=\frac{2}{L(L-1)}\sum_{i=1}^{L-1}\sum_{j=i+1}^{L}P_{i,j} \tag{2.4}$$

(3) 差异量的度量 *Dis*

这种度量方法是由 Skalak 提出的一种用来利用两个基分类器之间的差异来度量多样性的方法，如公式 2.5 所示.

$$Dis_{i,j}=\frac{N^{01}+N^{10}}{N^{11}+N^{10}+N^{01}+N^{00}} \tag{2.5}$$

Ho 就在决策森林方法(decision forest)中使用了这种度量方法来度量组合的多样性[8]. 这种度量方法简单地认为多样性的分类器的分类结果在相同的训练集上应该尽可能不同. 其度量了相同训练集上，两个分类器分类结果不一致的实例个数与整个训练集中实例个数的比值. 对整个分类器组合的差异量的度量 Dis_{av} 是所有两两基分类器的差异量的度量 *Dis* 的均值，如公式 2.6 所示. 当 *Dis* 大时，说明基分类器间的差异量比较高，即组合的多样性比较高.

$$Dis_{av}=\frac{2}{L(L-1)}\sum_{i=1}^{L-1}\sum_{j=i+1}^{L}Dis_{i,j} \tag{2.6}$$

(4) 双重错误的度量 *DF*

这种度量方法是由 Giacinto 和 Roli 提出来的，是用来从基分类器集合中选出最不相关的基分类器的. Giacinto 和 Roli 认为分类器越不相关，它们同时分错的次数就会越少. 因此该度量方法被计算了被两个基分类器同时误分的实例个数，如公式 2.7 所示.

$$DF_{i,j}=\frac{N^{00}}{N^{11}+N^{10}+N^{01}+N^{00}} \tag{2.7}$$

对整个分类器组合的差异量的度量 DF_{av} 是所有两两基分类器的差异量的度量 DF 的均值．如公式 2.8 所示．当 DF_{av} 小时，说明基分类器间的差异量比较高，即组合的多样性比较高．

$$DF_{av}=\frac{2}{L(L-1)}\sum_{i=1}^{L-1}\sum_{j=i+1}^{L}DF_{i,j} \tag{2.8}$$

(5) Kohavi-Wolpert 方差

Kohavi-Wolpert 方差的度量方法是由 Kohavi 和 Wolpert 在他们对分类器分类错误的分解公式中提出的，起源于分类器分类错误的偏差—方差分解．对一个实例 x，在某个具体的分类模型下，其类标 y 的预测的变化的表达式如公式 2.9 所示．

$$variance_x=\frac{1}{2}\left(1-\sum_{i=1}^{c}P(y=\omega_i\mid X)^2\right) \tag{2.9}$$

其中 C 表示类标的个数．在只判断对错的情况下 C=2，并且 $P(y=1|x)+P(y=0|x)=1$．因此，我们可以将上式推算成如公式 2.10 所示．

$$\begin{aligned}variance_x&=\frac{1}{2}\left(1-P(y=1\mid x)^2-P(y=0\mid x)^2\right)\\&=P(y=1\mid x)P(y=0\mid x)\end{aligned} \tag{2.10}$$

然后在整个训练集上对所有实例计算该量并平均化，就可以得到多样性的 Kohavi-Wolpert 方差度量方法，如公式 2.11 所示．当 KW 大时，说明基分类器间的差异量比较高，即组合的多样性比较高．

$$kw=\frac{1}{NL^2}\sum_{i=1}^{N}l_i(L-l_i) \tag{2.11}$$

(6) 基分类器之间的同意度 K

这是一种用来度量各个基分类器之间的可依赖度的方法．如公式 2.12 所示．

$$k=1-\frac{\sum_{j=1}^{N}(L-l(z_j))l(z_j)}{NL(L-1)(1-\bar{P})} \tag{2.12}$$

其中

$$\bar{P}=\frac{1}{NL}\sum_{j=1}^{N}\sum_{i=1}^{L}y_{j,i}$$

为基分类器的平均正确率．

(7) 一般化的多样性

这种度量方法是由 Partidge 和 Krzanowski 提出来的．从组合中随机选出两个基本分类器对一个实例分类，Partidge 和 Krzanowski 认为只有在其中一个基分类器

错误分类而另外一个正确分类的情况下，多样性才能达到最大值．而当两个基分类器都错误分类时，多样性最小．因此，对于一个从训练集中随机抽取出的测试实例 z_i，设 T_j 表示 $l_i=j$ 的概率，则一般化的多样性的度量方法如公式 2.13 所示．当 GD 大时，说明基分类器间的差异量比较高，即组合的多样性比较高．

$$GD = 1 - \frac{\sum_{j=1}^{L} \frac{j(j-1)}{L(L-1)} T_j}{\sum_{j=1}^{L} \frac{j}{L} T_j} \tag{2.13}$$

(8) 困难度的测量 *diff*

这种度量方法是由 Hansen 和 Salamon 提出来的．我们首先定义一个离散的随机变量 X，对一个从训练集中随机抽取的实例 z_i 来说，$V_i=(L-l_i)/L$．这样困难度的测量就定义成 X 在全体训练集上所有取值的方差，如公式 2.14 所示．

$$diff = var(V_i) \tag{2.14}$$

以上几种度量方法是目前为止比较典型的多样性度量方法，在不同的实际分类算法中都得到了具体的应用，并取得了良好的性能．

2.3.2 组合的多样性对正确率的影响

多样性好的组合分类器在分类时每个分类器所做的决定是独立且不同的．这样即使有分类器做出错误分类，其他分类器也会纠正这个错误．因此，人们认为多样性好的组合通常能产生较高的分类正确率．为了验证一个观点，研究人员做了大量的实验．

Kuncheva 总结了多种多样性的度量方法[19]．对每一种度量方法，都使用了多个不同的数据集生成多个不同的分类器组合，通过测量统计这些组合的多样性和准确率，来试图发现这两者之间的关系．实验表明在相同的训练集合上，Bagging 法产生的多样性要低于 Boosting 法产生的多样性，这也许可以解释为什么 Boosting 法生产的组合的准确率要高于 Bagging 法的组合．实验还表明多样性和正确率之间存在着一定的关系，但不是太明显，甚至在有的情况下，多样性高的组合分类精度反而较低．这中情况表明当前的多样性度量方法可能不太完善，有待于设计新的多样性测量方法．

另外 Kuncheva 还做了大量的实验来观察多样性和组合方法之间的影响[9]，及不同的多样性度量与组合方法的组合对分类正确率的影响．在试验中，Kuncheva 使用了 10 种多样性度量方法，10 种组合方法，并使用了 2 个数据集，在生成的多

个分类器组合中试图来发现多样性在不同的组合方法下与正确率的关系，并找出每个多样性方法所对应的最优的组合方法．其实验表明，双重错误的度量 *DF* 和困难度的测量 *diff* 在投票法和朴素贝叶斯法这两种组合方法下跟准确率有明显的关系．

此外人们还不断的提出了新的多样性度量方法，并做了大量的时间观测多样性与正确率之间的关系[20, 21, 22]．Tang 在分析了 6 种多样性度量方法后，介绍了它们之间的关系，并与边际(Margin)这个的概念联系在一起解释了为什么大部分度量方法跟准确率的关系并不明显，并依此指导新的多样性度量方法的设计[17]．

2.3.3 利用组合的多样性指导组合的生成

多样性的另一个问题是如何利用多样性的概念指导多分类器组合的产生．例如，Tsymbal 利用多样性概念指导属性选择[23]，使得选取的不同的属性子集形成的分类器组合具有较高的多样性．Golestani 提出了在 Boosting 算法中利用多样性来提高正确率的算法[24]，Banfield 利用多样性的标准来对基分类器集进行裁剪[25]，还有其他的方法等．当前，研究人员还提出众多以多样性或为了平衡多样性和基分类器的正确率而采用多样性和准确率的组合为目标函数的遗传算法来进行属性选择，分类器选择或者子分类器组合选择[26, 27]．

2.4 基分类器的生成策略

多分类器组合是在多个基分类器的基础上构建的．因此构建多分类器组合的首要问题是如何生成多个不同的基分类器．其次为了构建性能良好的组合，在基分类器的生产过程中，还应该考虑到多样性等评价指标，用它们来指导生产过程，使产生的基分类器的组合具有良好的分类性能．

那如何产生多个不同的基分类器呢？由于分类器是由分类模型在训练集上训练得到的．那通过使用不同的分类模型，或者不同的训练集都可以产生不同的分类器．当前存在的基分类器生成方法也基本都属于这两类：一类是使用不同的分类模型；另一类是使用相同的分类模型，而通过对训练集中的实例，属性，类标等的变化来形成不同的训练输入，以最终生成不同的分类器．

通过改变分类模型的方法比较简单，通过使用常见的不同的分类模型，如贝叶斯，决策树，神经网络，*K* 最近邻法等模型可以在相同的训练集合上生成不同的分类器，然后组合它们．David H. Wolpert 在 1992 年提出的 Stacking 方法就是一种

具有代表性的此类方法之一[28]. Stacking 算法中使用了两层分类模型，其中 0 层组合了多种不同的分类模型，它们的输出作为 1 层分类模型的输入，最终由 1 层分类模型给出最终的分类结果.

而使用相同的分类模型，通过改变训练输入来生成不同的分类器是目前广泛使用的一种方法，进过了多年的研究已经形成了众多的性能良好且广泛应用的具体算法. 本节接下来将按照改变训练输入的不同途径，并结合每种途径下的具有代表性的算法，对该类方法进行总结.

2.4.1 操作训练集合的训练实例

当给出原始的训练集合后，可以通过删除，添加，抽取等方法来操作训练集合中的部分实例来形成新的训练集合. 重复该过程多次后，形成多个有差异的训练集合. 在每个新的训练集合上都使用某个分类模型进行训练，可以得出多个不同的基分类器. 这种方法比较简单且通常情况下表现较好. 但是应用这样方法存在一个前提条件，即所采用的分类模型必须是不稳定的. 分类模型是不稳定的是指训练集合的较小的改动就能使分类模型产生的分类器产生显著改变，以及分类准确率的显著改变. 这一点是比较好理解的，因为该方法改变的实例跟原数据集合相比，毕竟是占比较小的比例的，如果分类模型是稳定的，那产生的基分类器之间的差异比较小，组合的多样性就低，因此其分类性能很可能就不好. 当前典型的该类方法有：

(1) 样本重采样技术

Bagging 的思想最早是由 Leo Breiman 在 94 年提出来的，是一种较为简单的重新选择和使用实例来达到改善分类器性能目的的方法，其采用了样本重采样技术[6]. 样本重采样是指对训练数据集进行随机抽取，来形成一个与原数据集不同的训练数据集，并在此训练集上建立分类器. Bagging 的具体操作方法是，从原始数据集中使用放回抽取随机选取若干个数据实例，使选取的数据集的大小和原始数据集的大小相等，数据集中的数据实例允许重复选取. 这样的话，选取后形成的数据集中，有一些原始数据实例在该数据集中可能会出现多次，而另外那些数据实例可能会一次也不出现. 每个新的训练集包含原始训练集合大约 63.2%的实例，其中一些实例重复出现. 通过多次迭代后，形成多个不同的训练集合，并在其上训练多个不同的基分类器. Bagging 方法通过重新选取数据集增加了形成的分类器的差异度，从而提高了泛化能力.

其实 Bagging 法形成数据集的过程实际上就是形成自助聚集(bootstrap

aggregating)，Bagging 的英文单词也来源于此．Bagging 法和其他非组合分类算法的主要区别在于训练数据集形成的方法不同．它使用了非独立的数据集来代替从某一领域中获得的独立数据集，而非独立的数据集的形成也仅仅是对原始数据集的重新取样．重新取样数据集出来的结果是不同的，然而肯定不是独立的，因为它们是由一个数据集产生的．Bagging 法的结果是产生的组合分类器的性能通常要比在原始数据集上产生的单个分类器要好．

(2) 去除部分实例

第二种简单的改变实例的方法是通过从原始训练集合中去除一个与其他部分不相交的实例子集来形成新的训练集合．例如，可以将原始的训练集合随机的分成 10 个不相交的子集，这样就可以通过每次去掉一个不同的子集，仅使用剩下的 9 个子集来形成一个新的训练集，并最终可以生成 10 个不同的训练集．这种方法类似于评估单分类器分类正确率时采用的 10 重交叉验证方法．因此用这种方法构建的多分类器组合有时候也被称为交叉验证组合．

(3) Boosting 方法

Boosting 方法是应用比较广泛的一种方法，最早由 Schapire 提出，Freund 对其加以改进[7]．和 Bagging 方法类似，Boosting 方法也是通过对原训练集合采用抽取技术形成新的训练集．然而不同的是，Boosting 法的每一个产生的分类器是受到上一次已建立的分类器的性能表现影响的．因此 Boosting 方法的构建各个分类器过程是循环迭代的．它鼓励新分类器成为处理先前分类器所不能正确分类的实例的专家，并根据分类模型的性能来对每一个分类器产生权重．

在 Bagging 算法中，各个实例都是以等概率被抽取的．而在 Boosting 算法中赋给每一个实例一个权重，根据权重的大小决定其被抽取的概率，且在每次迭代过程中权重都会发生变化．算法首先赋予训练集中每一个实例相同的权值，一般情况为 1 除掉所有实例总数．然后应用弱学习算法，在这些加权的训练集合上学习出一个分类器，再根据这个分类器的分类结果对每一个实例重新赋权．赋权的原则是增加上次被分类器错误分类的实例的权值，相应减小被分类器正确分类的实例的权值．这样产生了一组权值较低的“容易对付”的实例和另一组权值较高的“难以对付”的实例．使得在下一轮循环迭代以及所有以后的循环中，分类模型都是建立在经过重新加权的数据上，并且专注于对那些“难以对付”的实例进行正确分类．然后再依据这个新分类模型的分类结果增加或者减小相应实例的权值．结果是部分较难对付的实例可能变得更难，而另一部分较容易对付的实例可能变得更加容易；也可能是部分较难对付的实例变得比较容易，而交易对付的实例则变得较难对付．事实上，这两种情况在实际过程中都会发生．在每一轮的循环迭代之后，权值反映出目前所生成的分类器对当前训练实例的正确分类能力．通过对

每个实例“难度”的衡量，Boosting提供了一种巧妙的方法来生成一系列互补的专家.

最后通过加权投票组合这些基分类器．每个基分类器的权重由其在产生它的训练集上的分类正确率决定．正确率高越高，权重越大.

Bagging 和 Boosting 的区别主要是：Bagging 的训练集是随机生成的，但是Boosting的训练集是和上一次的训练结果相关的；Bagging的各个训练集之间的关系是相互独立的，可以并行生成，但是 Boosting 的各个训练集之间的关系并不是独立的，之间有关系，不可以并行生成；Bagging的各个分类器之间没有权重之分，而 Boosting 之间是有权重的．对于非常耗时的操作，如果 Bagging 和 Boosting 都能应用的话，Bagging比Boosting要节省大量的时间，因为Bagging的各个分类器可以并行生成.

另外，方差是表明在各个不同的数据集上分类器的预测的差异大小；偏差表明预测的值与实际值的差异程度[8]．在提高分类精度，减小误差上这两个方法的区别是：Bagging主要是通过减小方差的方式来提高精度；Boosting不仅仅是减小方差，而且还减小偏差，通过这两种方法来减小误差提高分类精度．因此，实际试验结果也显示Boosting比Bagging的分类精度要高；但是Bagging对方差的减小程度要比Boosting要大.大量的试验表明，Bagging具有比Boosting更稳定的特性，但是Boosting能够在降低错误率的程度上比Bagging要更有效.

2.4.2 操作属性集合

改变训练输入的另一种常用的方法就是操作当前训练集的属性集合，通过选取不同的属性子集，来形成不同的训练输入．和传统的属性选择只选取一个最优的属性子集不同，这里要选取多个合适的属性子集，以用来训练多个基分类器来形成组合．例如，Chekauer 就曾经训练了一个包含了 32 个神经网络分类器的组合．其中每个神经网络都是使用了包含了 119 个可用的属性 8 个不同的子集和 4 种不同网络的大小生成的，其中属性子集的划分是通过手动完成的．另外，Tumer和Ghosh也在一个包含了25个属性的训练集上使用了类似的方法，然而他们发现即使少量属性的去除，也会明显地降低基分类器性能．显然，这种方法只有在包含了大量属性的情况下，才会有较好的效果.

如何选取属性子集是该方法的关键问题．当前的方法主要包括随机选取属性来形成属性子集，例如 Ho 提出的随机子空间(Random Subspace)方法[8]，每次迭代中，都通过随机选取若干属性组合训练用的属性子集．还有就是利用更复杂的算法如遗传算法来生成多个属性子集，研究人员还提出了以多样性和正确率为目

标函数的多目标进化算法来选择属性子集[29,30].利用遗传算法能按照某个目标函数选择更加合适的属性子集．在实际应用中发现，当属性个数较大的时候，遗传算法比随机选取产生的分类器组合的错误率要低；而当属性个数小的时候，随机选择方法比遗传算法方法产生的分类器组合的错误率要低．这是因为属性个数小的时候，遗传算法选出的属性子集趋向于一致，产生的基分类器趋向于一致，组合的多样性减小．这也证明了组合的多样性是分类器组合的一个重要因素．

2.4.3 操作类标值

构建一个性能较好的分类器组合的第三种方法是操作学习算法可以学习到的类标值．与重采样技术相比较，该方法中所有训练集包含的实例都是一样的，但相同的实例在不同的训练集中所属的类标却不一样，这样也能使各个训练集互相不同．Dietterich 和 Bkairi 提出一种被称为错误纠正输出编码的技术[31]，就是典型的此种方法．

误差纠正输出编码是用于改进多类学习问题的分类算法性能的一种技术．有些学习算法只能用于二类问题．为了将这类算法用于多类数据集，要将数据集分解成几个独立的二类问题，每个运行一遍该算法并组合结果给出最终的预测．误差纠正输出编码就是做这种转换的一种方法．事实上，这个方法效果如此之好，即使学习算法能够直接处理多类的数据集，应用这个方法也经常能够获益．

在误差纠正输出编码中，假设类标的个数为 K，且是个很大的数字．每次在构建新的基分类器时，将这 K 个类标随机地划分成 A_t 和 B_t 两个子集．这样，输入的训练数据可以重新被赋给类标，所有原来的类标值属于子集 A_t 的实例其类标都被赋成 0，而所有原来的类标值属于子集 B_t 的实例其类标都被赋成 1．经过重新赋类标的训练集再被用来训练生成基分类器 D_t．重复这个过程多次，每次都随机地产生一个对类标的不同划分，就会得到多个不同的基分类器．

在面对一个新的测试实例 x 时，使用所有的上述过程得到的基分类器对其进行分类．如果分类结果为 0，那在该基分类器下，所有属于子集 A_t 的类标都得到一个投票；如果分类结果为 1，则所有属于子集 B_t 的类标都得到一个投票．在所有基分类器都投完票后，得票最多的类标就是最终的对测试实例的预测结果．

Dietterich 和 Bkairi 指出这种技术能够在很多困难的分类问题中改进 C4.5 决策树和神经网络模型的分类性能．但是误差纠正输出编码有一个缺点，就是它不能用来应用于局部学习算法，譬如通过观察附近训练实例来预测一个实例类别的基于实例的学习．优点是比较简单，可以使用任意用来解决两类问题的分类器．

2.4.4 人工添加数据

在分类器组合中，分类器的多样性是衡量分类器组合好坏的一个重要的理论特征．分类器的多样性指各分类器在与其所给训练数据集一致的前提下，对测试实例分类尽量不同．产生多样的分类器有多种方法，其中一种是产生不同的训练集，通过训练集的差异来产生不同的分类器．当前广泛流行的 Bagging 算法和 Boosting 算法都基于此方法．

然而，Bagging 和 Boosting 的限制在于：它们所采用的不同训练集都是从同一个原始数据集中按某种概率分布抽取得来的，并没有引进新的训练数据．这样，当原始数据集比较小时，所产生的不同数据集间必定会有大量的重复数据，从而间接限制了分类器的多样性．针对上述问题，Melville 和 Mooney 在 2003 年提出了 Decorate 算法[32]，该算法通过在每次迭代产生基分类器时，在训练数据集添加人工构建的新的训练数据来增大数据集的差异，以产生更加多样的分类器．为了产生与训练数据集中数据差异较大的数据，人工构建数据分为两步：首先根据各非类属性的值在训练数据集中的分布，随机构建一条数据；然后使用当前已有的分类器组合对其分类，将产生的类值概率取倒数并标准化，找出概率最大的类值并赋给该数据．这样所产生的数据会跟原训练集中相似数据具有不同的类标，从而加大数据集的差异．Decorate 算法的过程为：迭代产生基分类器．每次迭代时：首先构建若干条新的训练数据，并根据上述的构建方法为它们赋类值并加入原始训练集；根据当前训练集构建基分类器并验证加入该分类器后的分类器组合的分类错误率，如果比上次迭代完的分类器组合错误率小，则将该新构建的基分类器加入到组合中，反之，就丢弃它并开始新的迭代．当迭代到产生一定数量的基分类器时停止迭代．对测试数据进行分类时，采用一般的投票方法．

试验结果表明，在以决策树为基分类器时，该算法产生的分类器组合的预测精度比 Bagging 和 Boosting 算法产生的都要高．尤其当原始训练数据集较小时，能比 Boosting 算法更早地获得较高精度．

2.4.5 多策略学习

以上的各类方法不仅可以单独使用，而且为了使产生的组合具有更好的多样性，充分利用不同方法的优点，可以结合起来同时使用．综合多个方法有可能同时利用每个算法的优点，产生更好的分类器组合．Webb 提出了一种将 Boosting 法和一种 Bagging 法的变形 wagging 方法结合到一起的一种多策略学习方法[33]，在 Wagging 方法每次迭代过程中产生的新训练集上运用 Boosting 法产生一个子分类

器组合，最后合并多个子分类器组合形成一个最终的分类器组合．实验证明这种方法同时有 Wagging 算法的较大地减小方差和 Boosting 法能同时减小偏差和方差的优点．另外人们还提出了其他的多策略方法，如 Kotsianti 提出了将装袋、提升和 dagging 方法综合起来的一种算法[34]．Guruswamii 提出了一种将提升法和错误纠正输出编码结合在一起的方法[35]等等．

2.5 基分类器的选取

当生成多个基分类器后，传统的方法只是简单地使用了所有的基分类器来参与分类，并未对基分类器进行筛选．但通常情况下有两个问题：一是当生成过多的基分类器时，使用所有的基分类器来分类是一个非常耗时的过程；另一个更重要的问题是，使用经过某种方法挑选出来的基分类器子集形成的组合，跟使用全部的基分类器相比，可能具有更好的分类性能．另外，并不是对所有的测试实例，当前的基分类器组合对其都有良好的分类性能．因此还必须考虑到当前测试实例，根据其各属性的具体值动态地选择合适的基分类器形成组合，已达到更好的分类性能．因此，对基分类器进行选取就成了一个重要的研究问题．当前主要有两类选取方法，分别为“过量产生—选取”的静态选取方法和根据测试实例的分类器动态选取方法．

2.5.1 多分类器组合的过量产生－选取设计方法

近年来，为了提高多分类器组合的分类性能，研究人员提出了过量生成－选择(Overproduct and select)的设计方法来设计多分类器组合[15]．这是一种并没有考虑到当前测试实例的静态选取方法，其本质上是按照某种评价函数对基分类器空间的搜索．其基本步骤是：首先产生过量的不同的基分类器，形成一个分类器池；然后按照某种选取方法从上述产生的分类器池选取若干个分类器进行组合以达到最优的分类正确率．过量产生基分类器阶段，可以使用已有的多种方法实现，而如何从大量的基分类器中选择一个子集来形成性能良好的子集，则是了研究中的一个关键问题和难点．选取问题就是在基分类器空间中进行搜索，因此该问题主要包含两个关键点：一个如何搜索的目标函数的确定，即以什么样的标准评判搜索到的基分类器；另外一个是搜索方法，即如何在基分类器空间内进行搜索，使最终得到的基分类器组合在当前的目标函数下是最优的．

在过量生成阶段，可利用装袋，提升等基本方法．在选择阶段，最简单的可以利用穷举法来检查所有可能的子集，选取分类错误率最低的组合[15]．然而当基

分类器数目较多时，这种方法的计算复杂度太高．因此，多种方法已经被提出来限制选择阶段的计算复杂度．Partridge 和 Yates 提出了利用启发式规则来进行选择的方法[36]，其中一种是他们首先假设了组合一个固定大小的值 n，然后从集合中选出前 n 个分类错误率最低的基分类器形成组合．他们提出的另一种方法是选出对每个类标分类错误率最低的基分类器来形成组合，这样如果是一个三类问题，组合就会包括三个基分类器．很显然这些方法极大地减小了计算复杂度，因为它们没有对不同的子集进行评价比较，但是这些启发式规则并不总是能保证．

Roli 提出了三种类似属性选择的搜索方法[37]，这些方法以分类正确率为评价函数，包括向前搜索，即组合初始时只包括一个基分类器，每次搜索一个能使评价函数得到最大值的基本分类器添加进去，直到再添加基分类器会使评价函数变小时就停止搜索；向后搜索，即组合初始时包括所有基分类器，每次搜索减去一个能使评价函数得到最大值的基本分类器，直到再减去基分类器会使评价函数变小时就停止搜索；Tabu 搜索，前面两种方法在目标函数值减少时就会停止，但可能再后续的过程中还会增加，因此此时停止搜索是不合适的．Tabu 搜索就是基于这一考虑提出的．

另外，Kim 还提出了以遗传算法来进行分类器子集选择的方法[38]．利用了两层遗传算法搜索，一是基分类器层上的，一是组合层次上的来形成组合．该方法的优点在于生成的组合的基分类器个数少，并且是一种元搜索框架，独立于基分类器．

2.5.2 分类器的动态选取技术

多分类器组合另一个基本的研究方向就是基分类器动态选择[39]．在传统的基于组合的多分类器系统是基于这样一个假设：即每个分类器错误分类的情况都是独立的．基于这个假设有两个思路设计分类器组合：怎么设计多个分类错误独立的分类器，或者能不能找到一种组合方法能不需要这个假设．因此，人们提出了动态选取方法，该方法并不需要上述的假设，而是基于组合中至少有一个分类器能正确对测试实例分类的假设．同选择多个分类器进行组合不同，该方法根据测试实例选择最好的一个基分类器来进行分类．Giacinto 提出了动态选取的理论框架[40]，并证明了动态选取技术能产生最优的贝叶斯分类器．

对于动态选择算法，首先需要一种对输入样本划分的方法，即划分特征空间，然后对每个特征空间都找出一个该空间的最佳分类器．在分类阶段，首先判断测试实例属于哪个特征空间，然后使用该特征空间上的的最佳分类器对该实例进行分类．

Woods 提出了本地准确率(local accuracy)的概念[41]，首先根据 K 最近邻法确定测试实例的本地区域(local region)，测试每个基分类器在本地区域上的准确率，然后选择错误率最低的基分类器来对测试实例进行分类．这种方法同静态选取不同，考虑到了测试实例的属性值动态地选择不同的基本分类器．然而该方法易受 K 值和最近邻采用的计算距离的公式的影响，不同的 K 值可能导致性能上较为明显的差异．

因此在 Woods 研究的基础上，为了避免 K 值对选择的影响，Giorgio 利用 BKS 产生的分类结果向量来比较训练实例和测试实例之间的相似度，并设立一个阈值，来动态的决定 K 值[42]，这样 K 近邻的选择即利用了空间上的近似又利用了分类情况的近似．

Eschrich 则提出了利用聚类方法将训练集划分成不相交的子集，在每个子集上训练分类器[43]，分类阶段选择测试实例所最可能所属的子集上的分类器进行分类．

Zhu 本文提出了一种新的动态选取算法[44]，该算法利用了属性值的统计信息，用属性对评价数据集进行划分，在这些划分上测试每个基分类器的分类正确率．在分类阶段，根据测试实例的属性值，确定对应的划分，并计算每个基分类器在这些划分上的平均分类正确率，正确率最高的被选出来对测试实例进行分类．Zhu 认为基于 local accuracy 的算法有三个缺点：效果受 K 值和计算距离的函数的影响；速度上的劣势，对每个测试实例都要遍历整个训练集（评价集）来寻找最近邻；对噪音敏感，如果 local region 比较小的话．Zhu 认为在分类过程中每个属性的作用是不一样的,并且每个分类器的优势可以体现在不同的属性上，该算法利用属性值进行划分，一方面没有受到 *KNN* 算法的 K 值和计算距离的函数不同的影响，且不用再根据测试实例计算 local region 加快了计算速度；二也体现了不同分类器对不同属性有不同的专家知识这一假设．

总之，动态选择问题有两个主要问题：一是如何决定测试实例所属的特征空间，另一是如何决定该特征空间上的最优分类器．此外，在动态选取时可以选择一个基分类器组合而不仅仅是一个最优的分类器，Ko 就提出了一种利用验证集形成本地区域，并动态选取分类器组合的一种方法[45]．Shin 提出了一种将静态组合和动态选取相结合的一种方法[46]，用聚类方法对分类器进行划分，然后选择一个在测试实例的 local region 上正确率最高的一个划分和跟此划分距离最远的一个划分，使用投票法将这两个划分中的分类器结合起来．

2.6 基分类器的组合方法

在确定了多分类器组合的基分类器后，如何组合这些基分类器对测试实例进行分类是多分类器组合中的一个基本问题.

在确定采用哪种组合方法之前，首先要先确定基分类器的分类结果的表示形式. 不同的组合方法适用于不同的表示形式[47]. 当前，分类结果的表示形式主要有：

(1) 基分类器的分类结果仅仅给出一个类标.

(2) 基分类器的分类结果是一个所有类标按可能性大小的一个排序

(3) 基分类器的分类结果是一个向量，向量的每一个分量都给出了可能是每一个类标的程度大小，一般用概率来表示.

当前已经针对这三种输出结果形成，已经存在了多种组合方法设训练实例集合为$\{(x_1,y_1),\ldots,(x_n,y_n)\}$，$x_n$表示第$n$个训练实例，$y_n$表示其所属的类标. 多分类器组合为 $D=\{D_1, D_2,\ldots, D_L\}$，其中$D_L$为第$L$个基分类器. 则对一个测试实例$x$多个基分类器对其分类结果如表 2.2 所示.

如表所示，Y_n代表第n个类标，而D_L代表第L个基分类器. 如果分类结果是第一种情况，即仅给出类标，则如果基分类器D_i将x分成类标y_j，则p_{ij}=1，否则p_{ij}=0. 如果分类情况是第三种情况，则p_{ij}表示的是基分类器D_i将x分成类标y_j的概率，且$\sum_{j=1}^{n} p_{ij}=1$.

表 2.2 对一个测试实例 x,多个基分类器对其分类结果

Fig. 2.2 the result of classifications for a testing data x

	Y_1	Y_2	……	Y_n
D_1	p_{11}	p_{12}	……	p_{1n}
D_2	p_{21}	P_{21}	……	p_{2n}
…				
D_L	p_{L1}	p_{L2}		p_{Ln}

下面总结出几种常见的组合方法.

(1) 投票法

投票法是常用的比较简单的一种组合方法，它适用于分类结果为仅给出类标的情况. 其统计每个类值被预测的次数，次数最多的类值当认为是测试实例的类值[6]. 其中Y_x为最终的预测类标，如公式 2.15 所示.

$$Y_x = \underset{j=1}{\overset{n}{Max}} \sum_{i=1}^{L} p_{ij} \qquad 2.15$$

(2) 乘积法

适用于输出分类结果形式为每个类标概率情况,. 对每一个类标，首先将所有基分类器给出的预测类标可能是它的概率相乘，然后取乘积最大的类标作为最终的分类结果，如公式 2.16 所示.

$$Y_x = \underset{j=1}{\overset{n}{Max}} \prod_{i=1}^{L} p_{ij} \tag{2.16}$$

(3) 最大值法

适用于输出分类结果形式为每个类标概率情况,. 对每一个类标，首先找出所有基分类器给出的预测类标可能是它的概率的最大值，然后这些值中最大的类标作为最终的分类结果，如公式 2.17 所示.

$$Y_x = \underset{j=1}{\overset{n}{Max}} (\underset{i=1}{\overset{L}{Max}}\, p_{ij}) \tag{2.17}$$

(4) 最小值法

适用于输出分类结果形式为每个类标概率情况,. 对每一个类标，首先找出所有基分类器给出的预测类标可能是它的概率的最大值，然后这些值中最大的类标作为最终的分类结果，如公式 2.18 所示.

$$Y_x = \underset{j=1}{\overset{n}{Max}} (\underset{i=1}{\overset{L}{Min}}\, p_{ij}) \tag{2.18}$$

(5) 均值法

适用于输出分类结果形式为每个类标概率情况,. 对每一个类标，首先将所有基分类器给出的预测类标可能是它的概率相加并求平均值，然后这些值中最大的类标作为最终的分类结果，如公式 2.19 所示.

$$Y_x = \underset{j=1}{\overset{n}{Max}} (\underset{i=1}{\overset{L}{avg}}\, p_{ij}) \tag{2.19}$$

另外，Xu 还给出了投票法的最一般的公式[47]. Kittler，kuncheva 等人提出了一个组合方法的理论框架，指出当前存在的最大值，最小值，均值，加法，乘法，投票法等组合方法都是该理论框架下的具体实例 [48]. 进一步，这些理论框架为设计新的组合方法提出了理论基础和方向. 此外，Wolpert 提出了栈的组合方法[28], 该方法提出了元分类器的概念，以基分类器的输出作为训练集并在之上再训练生成一个元分类器对数据进行分类.

以上本章内容就是目前关于多分类器组合研究的主要内容，同时也给我们提出了许多值得深入研究的问题. 也是本论文实现的工作的重要理论基础，本文在以下两章中将在上述的几个研究问题中进行进一步研究，提出新的实现方法.

3 基于多样性的基分类器选取方法

本文第 2 章中阐述了多分类器组合的概念和实现中的几个关键问题．指出了实现一个多分类器组合的步骤和为产生性能优异的多分类器组合而应该注意的问题．其中，基分类器的选取是多分类器组合中一个重要的研究问题．当生成多个基分类器后，传统的方法只是简单地使用了所有的基分类器来参与分类，并未对基分类器进行筛选．但通常情况下有两个问题：一是当生成过多的基分类器时，使用所有的基分类器来分类是一个非常耗时的过程；另一个更重要的问题是，使用经过某种方法挑选出来的基分类器子集形成的组合，跟使用全部的基分类器相比，可能具有更好的分类性能．因此，十分有必要对基分类器进行选取以提高分类性能．

本章在总结前人研究的基础上对上述的基分类器选取中的关键问题进行了进一步的研究．利用了当前存在的多样性度量方法来指导基分类器的选取，并结合了贪心算法提出了一种基于 Boosting 算法基础上的基分类器选取策略．

3.1 决策树分类器

决策树表示方法是分类算法中应用最广泛的方法之一，它从一组无次序，无规则的事例中推理中决策树表示形式的分类规则．决策树分类方法采用了自顶向下的递归方式，在决策树的内部结点进行属性值的比较并根据不同的属性值判断从该结点往下的分支，在决策树的叶结点得到结论．所以从决策树的根到叶结点的一条路径就对应着一条合取规则，整棵决策树就对应着一组析取表达式规则．

决策树是以样本为基础的归纳学习分类方法，其表现形式是类似于流程图的树结构，其中每一个内部节点表示在一个属性上的测试，每个分支代表一个测试输出，而每个树叶节点代表类或者是类分布．树的最顶层节点是根节点．一个典型的决策树如图 3.1 所示，它预测顾客是否可能购买计算机．为了对未知的样本进行分类，样本的属性值在决策树上测试，路经由根到存放该测试样本的叶节点．决策树容易将决策结果转换成分类规则．基于决策树的分类方法在训练过程中不需要用户了解很多背景知识，只要训练样本能够用属性－值的方式表达，就可以使用决策树来训练并分类．

若上述的决策树的一个测试样本是<*age*=20，*student*=yes，*credi_rating*=fair>，那么从决策树的根节点开始测试属性，并按属性值对应的分支向下走，直到叶节点，可以判定该样本属性标记为“*yes*”的类．

决策树的基本算法是贪心算法，采用自顶向下的递归方式构造决策树．Hunt等人于1966年提出的概念学习系统CLS是最早的决策树算法，以后的许多决策树算法都是对CLS的改进或者由CLS衍生而来．Quinlan于1979年提出了著名的ID3算法[4]．以ID3为蓝本的C4.5算法是一个能处理连续属性的算法[49]．其他决策树方法还有 ID4 和 ID5 等．强调数据挖掘中有伸缩性的决策树算法有 SLIQ、SPRINT、RainForest等算法[50,51,52]．

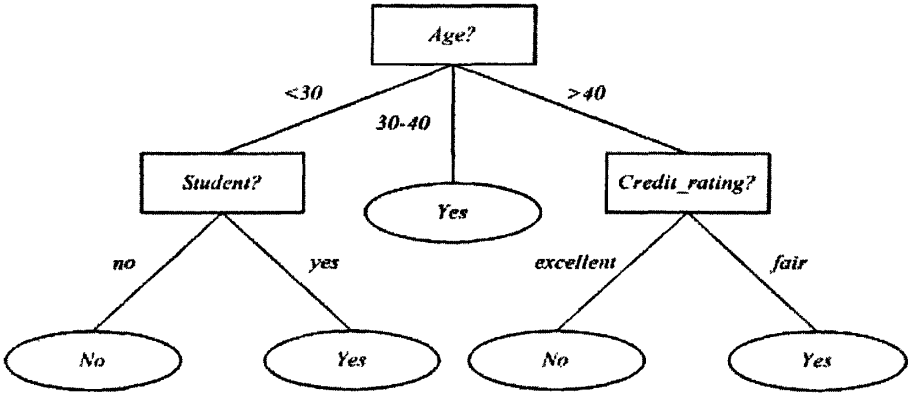

图3.1 是否买电脑的决策图模型
Figure 3.1 decision tree model for whether buying computer

为了产生多样性较高，分类效果优良的多分类器组合，本论文的研究的算法仍然是基于基分类器是不稳定的分类器的假设上的．而决策树分类器就是一种不稳定的分类器，因为在训练时，少量训练数据的变化，就能引起决策树的内部测试结点的变化，能导致分类结果较为明显的变化．而且，决策时分类器是一种分类性能良好，应用广泛的算法．因此，在本论文以下的研究中，均采用了决策树分类器作为基分类器，具体采用了C4.5算法来生成基分类器．

3.2 基分类器空间搜索的问题

分类器的生成—选取思想基本步骤是：首先产生过量的不同的基分类器，形成一个分类器池；然后按照某种选取方法从上述产生的分类器池搜索选取若干个分类器进行组合以达到最优的分类正确率．

一般搜索问题都包含两个关键点：一是搜索满足什么样标准的个体，即评价函数的确定；二是如何进行搜索．

因此，在基分类器的选取过程中，首先要确立的是要选取满足何种标准的基

分类器子集，即子集评价标准的确立．通常，最简单的评价标准就是分类正确率，即在测试集合上测试所选择的基分类器子集的正确率，然后选择正确率最高的基分类器子集．然而，在测试集上分类性能好的子集在对未知类标的实例分类性能未必好，因此研究人员提出了许多其他的评价标准．其中，组合的多样性就是被公认的一种影响多分类器组合性能的重要因素．组合的多样性是指在对测试实例进行分类时，不同的分类器的分类结果是独立且不相同的．这样即使有分类器做出错误的分类，其他的分类器也会纠正这个错误，这样就有可能提高分类器的分类性能．当前人们提出了多种多样性的度量，并将这些度量做为选择基分类器的评价标准．在基分类器空间进行搜索时，选择多样性好的基分类器子集做为组合的候选集合，以希望产生的组合在对未知数据分类时能有较好的分类性能．在本文第二部分已经总结了多样性的概念及现有的多种度量方法，在本节的研究中将使用一种多样性的度量来做为选择基分类器的评价标准．

其次，确定了评价标准后，另外一个问题就是如果高效地在基分类器空间进行搜索．最简单的可以利用穷举法来检查所有可能的子集．然而这种方法的计算复杂度太高．当存在 n 个基分类器时，子集的个数为 2^n 个，子集的个数成指数级增长，当 n 值较大时，遍历所有的子集将会是一个非常耗时的过程．因此，需要提出一些方法来限制搜索的计算复杂度，产生高效的搜索过程．在本文的第二部分已经总结了当前已有的搜索方法，如 Partridge 和 Yates 的利用启发式规则选择，Roli 提出的三种搜索算法和当前较为常见的利用遗传算法进行搜索的算法．

然而，这些算法中要么是搜索过程比较简单，例如利用启发式规则的搜素算法仅搜索一个固定值 n 大小的基分类器子集，这只能搜索到很少的一部分子集，因此大多数情况下不能找到最优的子集；另外，Roli 提出的三种搜索算法中仅使用了当前分类错误率做为评价标准，并不一定能找出对未知实例分类性能好的子集．因此，为了解决这些问题，本节将重点将多样性和一些简单高效的搜索算法结合起来，完成基分类器的搜索和选取．

3.3 选取问题的分治思想

任何一个计算机可以求解的问题所需的计算时间都与其规模有关．问题的规模越小，所需时间越短．当想解决一个较大的问题有时候是十分困难的．常规的分治思想是：将一个较大规模的问题，分割成一些规模较小的相同问题，以便各个击破，分而治之．反复应用分治法，最终使子问题缩小到很容易求解．

设基分类器空间为 $D=\{d_1, d_2,...,d_i,...,d_n\}$，选取就是从这 n 个基分类器中挑选一个性能最优的基分类器子集．该问题就相当于要求解出出一个向量 $\{x_1, x_2,...,$

$x_n\}$，其中若选中了基分类器 d_i，则相应的 x_i 为 1，否则为 0，当 n 值较小时，遍历所有的子集并不是件很困难的事情；而当 n 值较大时，遍历所有的子集就变得十分困难．这时就需要利用分治思想来将一个较大的问题划分成多个较小的相同的子问题来解决．而基分类器选取问题就是一个典型的可以应用分治法划分的问题．当原问题为从 n 个基分类器中挑选一子集时，可以将其划分成从 n-1 个基分类器中挑选一个子集，然后再考虑是否选取剩下的那个基分类器的子问题．依次类推，可以将原问题的规模从 n 划到 2，此时的子问题就非常好解决了．

然而，如何划分并组合子问题的解以得到原问题的解，在不同的具体策略下有着不同的实现方法，在接下来将贪心算法这种常见的策略实现子问题划分并求解的方法．

3.4 利用贪心策略选取基分类器

贪心策略通过一系列选择来得到一个最优的解．它所做的每一个选择都是当前状态下的某种意义的最优选择，即贪心选择．希望每次所做的贪心选择导致的最终结果是问题的最优解．贪心测量有两个重要的性质：贪心选择性质和最优子结构性质．

所谓贪心选择性质是指所求问题的整体最优解可以通过一系列局部最优的选择，即贪心选择来达到．这是贪心策略的第一个要素，也是贪心策略与动态规划测量主要区别．在动态规划算法中，每一步所做的选择都与相关子问题的解相关，因此只有在解出相关子问题后才能做出选择．而在贪心策略中，仅在当前状态下做出最好选择，即局部最优选择．然后去解做出这个选择后的相应子问题．

当一个问题的最优解包含着它的子问题的最优解时，称该问题具有最优子结构问题．问题具有这个性质是该问题可以利用动态规划或者贪心策略求解的一个关键特征．

对于从 n 个基分类器集合中选取一个子集问题，利用贪心策略的一个求解过程为：初始时，子集只包括一个基分类器，然后逐一向其中添加基分类器，使得每次添加进去的基分类器都能使该子集的某种评价标准(多样性) 在当前情况下得到最大值，直到再添加基分类器会使评价函数变小时就停止搜索．

根据上述的求解过程，本节给出了一个具体的算法 GreedSelection 的实现．在基分类器集合的生成阶段，使用了经典的 Boosting 算法来首先生成了大量的基分类器；在选择阶段，使用了本文第二部分总结的多样性度量方式作为评价标准，利用上述的贪心策略过程从这些基分类器中逐步挑选基分类器加入到候选的子集中，使每一次加入的基分类器都能使当前的多样性度量的值为最大值，在本文的

实现中，将采用多种不同的多样性度量方法，观察在不同的度量方法下多分类器组合的分类正确率的不同，以及多样性对正确率的影响；分类阶段，仍将使用传统的加权投票方法来组合基分类器的分类结果.

表 3.1 GreedSelection 算法描述

Table3.1 The description of GreedSelection algorithm

算法：GreedSelection
输入：一个训练集 S，
输出：训练集 S 上经过选择后得出的多分类器组合,基分类器个数为 n
训练阶段：
1. 初始化，赋予每个训练实例相同的权值．循环计数 t=0.
2. 循环过程：
(1) 算法应用于加权的数据集上并保存分类模型.
(2) 分类模型在加了权的数据集上的误差 e 并保存该误差 e，如果 e 等于 0 或者大于等于 0．5：终止建模型.
(3)据集中的每个实例,如果模型将实例正确分类，则将实例的权值乘上 e/(1-e).
(4) 将所有的实例权值进行正规化.
选取阶段
1. 初始化选出的基分类器子集 D 为空，随机选取一个基分类器 d_i 进入子集.
2　计算将剩下每个基分类器加入子集后子集的多样性大小.
3．循环过程：
(1) 对剩下的还没有被选择的基分类器，选出加入子集后能使子集的多样性最大的基分类器
(2) 如果其能提高子集的多样性，就将其加入，否则就停止循环.
(3) 更新剩下的基分类器加入子集后子集的多样性大小.
分类阶段：
1. 赋予所有的类权值为 0.
2. 对于选取阶段选出的子集中的每一个基分类器的每一个：
给其所预测的类加权-log(e/(1-e)).
3. 返回权值最高的类.

GreedSelection 算法流程如表 3.1 所示．该算法的特点主要有如下几点.

首先该算法是基于这样一种假设的，即如果选取出的集合的多样性最大，那么它所包含的任意子集在相应的空间中，都是该空间中多样性最大的子集．例如，

假设共生成了 20 个基分类器，选出的一个多分类器子集 $D^1=\{d_1, d_3, d_7, d_{20}\}$ 为这 20 个分类器中多样性最高的子集，则 $D^2=\{d_1, d_3, d_7\}$ 应该是前 7 个分类器中多样性最高的子集．这在直观上时容易理解的，因为整体的多样性跟组合中的每个基分类器的分类结果都有关，如果部分基分类器的多样性不是最优的，那肯定会影响到组合整体的多样性的．正是在这种假设下，基分类器的选取问题才符合贪心算法的最优子结构性质，使其能利用贪心策略来解决．

其次，该算法每次在计算子集的多样性时，并不会在每添加一个基分类器后都重新计算整个子集的多样性，而是在保存当前多样性大小的基础上，仅计算新加进来的基分类器与当前子集中的基分类器之前对多样性的影响并加上原来的多样性大小，形成最终的结果．例如，在采用 Q 统计量来计算多样性时，仅计算新加进来的基分类器与当前子集中的每一个基分类器的 Q 统计量再加上当前的组合的 Q 统计量并平均化即可．这样就节约了计算时间，提高了分类速度．

该算法主要存在的问题是，其所依赖的假设并不总是一定成立的，其在搜索到使子集的多样性减小后就停止搜素，而实际中再继续添加基分类器反而可能会继续增大子集的多样性，这是在以后的研究中要解决的一个问题．正是这种基于贪心策略的算法不能搜索评估到所有的基分类器子集，为了解决这个问题并继续保持高效的搜索过程，本文对该问题继续研究，提出了利用动态规划策略来实现选取过程的方法．

4 Boosting 算法中基分类器权重的动态赋值

Boosting 算法是应用比较广泛的一种分类器组合算法[7]. 然而在基分类器的权重分配问题上，该方法仅仅利用了基分类器在其训练集上的错误率的某种变形来静态地为基分类器分配权重. 这种方法，并没有考虑到待测试实例的可能取值情况. 当基分类器在其训练集上具有较低的错误率时，并不能说明该基分类器对待测实例分类正确的概率就大. 这时，对其赋一个较高的权重反而可能会降低分类精度. 为了解决 Boosting 算法中基分类器权重的静态分配可能带来的问题，本文采取动态选取思想[39]提出了一种动态地对基分类器分配权重的方法. 即根据测试实例的取值情况按照某种方法度量它被每个基分类器正确分类的可能性大小，然后按照这种度量动态地对基分类器分配权重. 实验证明，经过该方法动态对基分类器进行权重分配后，组合的分类精度在大多数情况下要高于静态分配权重的经典 Boosting 算法.

4.1 传统的 Boosting 算法

1996 年，Freund 和 Schapire 提出了 AdaBoost(Adaptive Boost) 算法[7]. 该算法是一种能够适用于任何分类的学习算法. 算法首先赋予训练集中每一个实例相同的权值，一般情况为 1 除掉所有实例总数. 然后应用弱学习算法，用这些加权的训练集合学习出一个分类器，再根据这个分类器的分类结果对每一个实例重新赋权. 赋权的原则是增加上次被分类器错误分类的实例的权值，相应减小被分类器正确分类的实例的权值. 这样产生了一组权值较低的“容易对付”的实例和另一组权值较高的“难以对付”的实例. 使得在下一轮循环迭代以及所有以后的循环中，分类模型都是建立在经过重新加权的数据上，并且专注于对那些“难以对付”的实例进行正确分类. 然后再依据这个新分类模型的分类结果增加或者减小相应实例的权值. 在每一轮的循环迭代之后，权值反映出目前所生成的分类器对当前训练实例的正确分类能力. 通过对每个实例“难度”的衡量，AdaBoostM1 提供了一种巧妙的方法来生成一系列互补的专家.

分类阶段，每个基分类器都赋给一个权重，权重的分配仅仅利用了相应基分类器在其训练集上的错误率的某种变形来静态地为基分类器分配权重. 这种情况下，基分类器的权重在对不同的实例分类时，权重是不变的. 而直观上不同的基分类器应该有其侧重的数据，不同的测试实例对应的最优分类器也应该不同. 因此，应该根据测试实例的取值情况动态地对基分类器分配权重.

4.2 基分类器权重的动态度量

在该改进算法中，利用到了被每个基分类器所错误分类的训练数据的子集所包含的信息．用 C_i 表示被第 i 次迭代产生的基分类器所错误分类的数据子集．我们可以在 Boosting 过程中保存被每一个基分类器所错误分类的训练数据子集，C_1, C_2, ..., C_m (有 m 个基分类器)．当待测数据来临时，首先判断该待测实例属于每个被错误分类的数据子集的程度，如果属于某个数据子集的程度大，那么就相应地增加关注于该错误分类的数据子集的基分类器的权重，即下次迭代生成的基分类器的权重．

在该算法中，我们使用了类似 K-最近邻法的方法[3]来测量测试实例属于某一被错分的数据子集的程度．设 $C_i=\{n_{i1}, n_{i2}, ..., n_{im}\}$ 为第 i 个产生的基分类器所错误分类的数据子集，其中 n_{ij} 为该子集中第 j 个数据．设数据 $z=\{x_1, x_2, ..., x_n\}$，x_i 代表数据 z 的第 i 个属性．测量待测实例属于某一被错分的数据子集的程度可分为以下步骤．

一是首先测量待测实例 z 与每个被错误分类的训练数据子集 C_i 中每个实例 n_{ij} 之间的距离，如公式 4.1 所示．

$$d_{ij}=\sum_{k=0}^{k=n} f(x_k, x_{ijk}) \tag{4.1}$$

其中 x_k 为实例 z 的第 k 个属性，x_{ijk} 为被错误分类的训练数据子集 C_i 中实例 n_{ij} 第 k 个属性．若属性为名称性则如果两者形同，则 $f(x_k, x_{ijk})$ 为 0，否则为 1；若为数值型，则 $f(x_k, x_{ijk})$ 为两个属性值的差的绝对值比上该属性的取值范围．

二是测量待测实例 z 与每个被错误分类的训练数据子集 C_i 的距离 d_i，如公式 4.2 所示．

$$d_i=(\sum_{j=0}^{j=n} d_{ij})/n \tag{4.2}$$

距离 d_i 表明了测试实例属于该训练数据子集的程度，也间接表明了它能被关注此被错误分类训练子集的基分类器正确分类的可能性大小．d_i 值大的表明该测试实例不属于次错误分类子集的程度较低，即被相应基分类器分对的可能性较低，因此应该对其赋一个较低的权重．因此，在对基分类器权重的动态调整过程中，将使用 $1/d_i$ 为基础对基分类器权重进行动态调整．本文中，对其进行标准化使其总和为 1，然后使用标准化后的 $1/d_i$ 作为第 $i+1$ 基分类器的权重．因此第 i 个基分类器的权重计算如公式 4.3 所示．

$$v_i = (\frac{1}{d_{i-1}} \Big/ \sum_{i=1}^{m} \frac{1}{d_i}) \qquad 4.3$$

由于在 Boosting 算法中，第一个基分类器之前并没有产生相应的错误分类子集．因此，在这次动态调整权重的算法中，仅仅简单地舍弃了第一个基分类器．

4.3 动态调整基分类器权重的流程

在本文上述中已经阐述了动态调整基分类器权重的思想及计算公式．本节给出了依据上述思想的一个具体算法 DynamicAdjust 的实现．在该新的算法中，在生成分类器组合阶段采用和经典的 Boosting 算法类似的流程，但要保存被每一个基分类器所错误分类的训练数据子集：C_1, C_2,...,C_m (有 m 个基分类器)，以用来在测试阶段计算相应基分类器的权重．在分类阶段，要首先根据测试实例属性的取值情况计算每个基分类器的权重，然后按照加权投票方式对多个基分类器进行组合．该算法的流程如表 3.4 所示．

表 4.1 DynamicAdjust 算法描述

Table4.1 The description of DynamicAdjust algorithm

算法：DynamicAdjust
输入：一个训练集合 S,
输出：训练集 S 上经过选择后得出的多分类器组合,基分类器个数为 n
训练阶段：
1. 初始化，赋予每个训练实例相同的权值．循环计数 t=0.
2. 循环过程：
(1) 算法应用于加权的数据集上并保存分类模型.
(2) 分类模型在加了权的数据集上的误差 e 并保存该误差 e，如果 e 等于 0 或者大于等于 0.5：终止建模型.
(3)据集中的每个实例,如果模型将实例正确分类，则将实例的权值乘上 e/(1-e).
(4) 将所有的实例权值进行正规化.
分类阶段：
1. 赋予所有的类权值为 0.
2. 对于选取阶段选出的子集中的每一个基分类器的每一个：
依据公式 3.4 给其所预测的类加权.
3. 返回权值最高的类.

在表 4.1 所示的算法模型下，由于不同测试实例的属性取值不同，在每次进行

分类时基分类器的权重都会根据测试实例属性值的变化而变化．这就克服了经典 Boosting 算法中静态基分类器权重赋值过程中，权重一直保持不变的缺点．

以上部分是本文在已有的研究工作基础上对多分类器组合的所做的进一步研究．在上述工作中，本文提出了一种新的算法，在本文接下来部分将通过在具体数据集合上实验来进一步分析算法的性能和优缺点，并指出进一步的研究问题．

5 实验结果分析

本章中，本文实现了在 Weka 平台在上面章节中提出的 GreddSelction、以及 DynamicAdjust 两种算法并进行了实验，并通过分析实验结果比较了这些算法与已有算法的性能差异，进一步分析产生这些性能差异的原因，并进一步指出需要改进的问题，或算法的适用情况.

5.1 实验平台 Weka 简介

Weka 全称 Waikto Environment for Knowledge Analysis，即怀卡托智能分析环境的缩写，是一款免费的非商业化的机器学习和数据挖掘软件[53]. Weka 开发的本意用于服务新西兰国家，解决新西兰工业遇到的知识发现问题，使机器学习理论联系实际，为机器学习领域提供一个框架. 其设计者为数据挖掘界的大师 Ian H. Witten 和 Eibe Frank. 目前已经向所有机器学习和数据挖掘的爱好者开放，现在经过这些研究者和爱好者的补充，Weka 平台的内容得到了很大的扩充，尤其是经典的优秀的算法在 Weka 上基本上都能找的到，为研究者提供了良好的实验条件.

Weka 平台采用了跨平台的 Java 语言开发，由于 Java 程序的平台无关特性使得 Weka 可以运行于任何装有 Java 虚拟机的操作系统上. 在已经测试过的平台包括 Linux，Windows 和 Macintosh 操作系统，甚至还包括个人数字化助手. 2005 年 8 月，在第 11 届 ACM SIGKDD 国际会议上，怀托卡大学的 Weka 小组荣获了数据挖掘和知识发现领域的最高服务奖，Weka 系统得到广泛的认可，被誉为数据挖掘和机器学习历史上的里程碑，是现今最完备的数据挖掘工具之一. Weka 的每月下载次数已超过万次. 由于 Weka 具有开源、兼容和结构规范等特性，自从发布以来，已经吸引了众多的用户，并且也不断得到扩展. Weka 目前的稳定版本是 Weka3-4-13 版，最新的版本号已达到了 Weka3-6-0.

Weka 工作平台汇集了当今最前沿的机器学习算法及数据预处理工具，目的是为了让用户能够快速灵活地将现有的处理方法应用于新的数据集. 它为数据挖掘实验的整个过程，包括准备要输入的数据，统计地评估学习方案，以及可视化输入数据及学习结果，提供了广泛的支持. Weka 不但包含多样化的学习算法，还提供大量适应范围很广的预处理工具. 用户可通过一个公用的操作界面运用所有已包含的工具组件，从而比较不同的学习算法，找出能够解决当前问题的最有效的方法. Weka 平台为数据挖掘实验的整个过程，包括数据输入、数据分析、学习过程、评价方案以及可视化输出等，提供了非常广泛的支持.

Weka平台目前包含的功能主要有：属性选择、数据预处理、分类、聚类、和关联规则．分类器是Weka平台的重要组成部分，也是其中内容最多的部分，其中包括了决策树、决策表、贝叶斯分类、贝叶斯网络、规则学习分类、懒惰式学习分类、元学习分类以及神经网络学习分类等．Weka平台底层统一了接口，只需要继承或者实现其相应的接口就能实现自己的算法，而对于数据集的表示、数据的预处理、相应结果的可视化输出、一些标准的评价方式等都不需要做过多的关注，甚至根本不用编写代码．这就为研究者节省了许多宝贵的时间，而将重点放在自己的算法实现上；同时，Weka平台有相应的一些优秀的经典的算法，在实现自己算法后可以和同类的算法相比较性能的好坏．

5.2 Weka系统中的元学习

5.2.1 相关概念

元学习是指对学习算法的学习．在数据挖掘分类问题中，ID3, Naive Bayes, Tan等都属于基本学习算法，输入为待分类的实例，输出分类结果．而元学习以多个学习算法的分类结果为输入，通过进一步学习输出一个最终的分类结果．目前，对于分类算法的元学习的使用方法主要有两种：采用相同的算法，但对数据集采取不同的处理策略，以产生多个不同的训练集来生成多个分类器；采用不同的算法，来生成多个结构不同的分类器．

5.2.2 Weka下meta学习算法的结构分析

在Weka.classifiers包中包含了常见的meta学习算法的实现，这些meta学习算法主要是针对基本的分类学习算法的学习．通过面向对象的设计，共生成31个类文件．根据元学习算法的不同，这些文件中可分为：代表不同元学习算法的基类文件，这些基类文件定义了同一类的算法的框架，应该实现的函数；代表具体元学习算法实现的文件，这些类继承上述的基类文件，实现了具体的元学习算法．生成分类器时可以迭代生成基分类器，也可以利用随机种子随机生成基分类器．这些文件大致可分为如下．

对分类模型学习的元学习算法的基类．这些基类规定了元学习算法的最基本成员，但不说明基分类器的产生方式．被提升的基分类器可以是单分类器，也可以是属于相同或不同分类模型的分类器组合．基分类器可以属于相同或不同的分类模型．包含文件：SingleClassifierEnhancer.java文件，MultipleClassifier.java文件．

规定了基分类器产生方式的元学习算法的类．这些类继承了上述的基类，同时它们也说明了基分类器的产生方式．根据基分类器生成方法的不同可以分为如下两类．

迭代生成基分类器：使用相同的分类模型，每一次迭代生成一个新的分类器．包含文件：IterativeClassifier.java 文件，IteratedSingleClassifierEnhancer.java 文件．

随机生成基分类器：即在生成基分类器的时候，使用随机数种子随机生成基分类器．生成的分类器组合中的基分离器既可以属于相同的分类模型，也可以属于不同的分类模型．　这一类的文件包括类文件如下：

RandomizableClassifier.java 文件，RandomizableSingleClassifierEnhancer.java 文件，RandomizableMultipleClassifiersCombiner.java 文件．

此外，随机生成基分类器还可以与迭代生成基分类器相结合，同时使用．这包括 RandomizableIteratedSingleClassifierEnhancer.java 文件．

这些文件之间的关系图如图 5.1 所示．

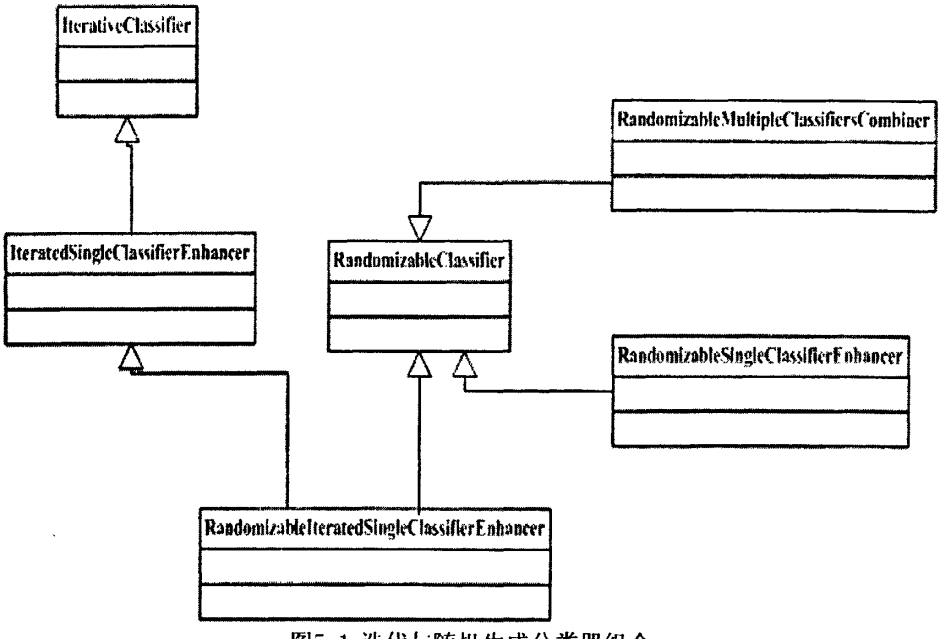

图5. 1 迭代与随机生成分类器组合

Fig 5.1 iteration and random creationof claasifiier combination

对上述文件的具体划分类别如下．

(1) 对单分类器提升的元学习．在 weka.classifers.meta 包下包括了常见的属于基本的分类器组合提升类的具体的元学习算法．其中一类是对单一分类器提升的

元学习算法．主要包括以下类文件．

FilteredClassifier.java 文件：该算法表示在已经经过任意的 filter 离散化处理过的训练数据集上建立的任意的分类器．

AttributeSelectedClassifier.java 文件：该算法表示在已经经过属性选择处理过的训练数据集上建立的任意的分类器．

RegressionByDiscretization.java 文件：该算法是一个用于回归方案的一个算法，其中使用了在类属性上离散化的数据集的任何分布的分类器．预测值是每一个基于预测概率的离散内部间隔的均值．

OrdinalClassClassifier.java 文件：在机器学习分类算法中通常假设类值是无序的，然而在实际应用中类值是拥有一个自然的顺序的．该算法表示一个元学习算法，它把一个类值有序的分类问题转化为一系列的二类分类问题．

ClassificationViaRegression.java 文件：该算法使用回归方法进行分类．

它们之间的关系如图 5.2 所示．

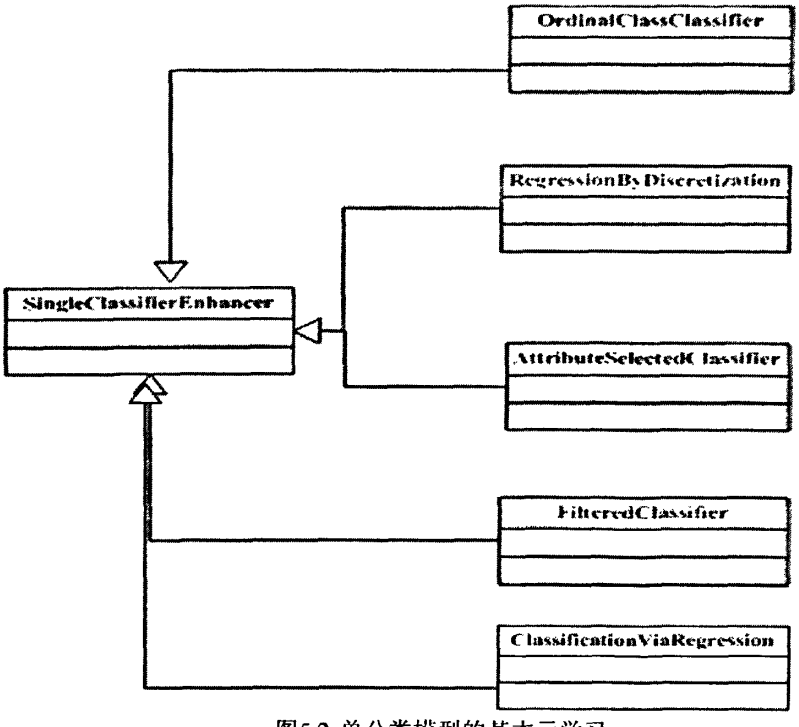

图5.2 单分类模型的基本元学习

Fig 5.2 the basic Meta-Learning about single classification model

(2) 基本的组合分类器提升的元学习算法．所谓分类器组合，也就是将多个不同的分类器，通过一定的方法，利用相关的技术最终形成一个组合分类器．组合分类器有时也叫做“委员会”，被集成的分类器称为子分类器．在对数据进行预测

时，首先使用每个子分类器对数据进行预测并产生结果，然后根据某种规则来综合各子分类器的分类结果，并最终形成最终的分类结果．在 weka.classifiers.meta 包中包含的该类型的具体元学习方法包括以下文件．

Vote.java 文件：分类器投票的方法同样可以构建分类器用于分类．具体的策略是，在刚开始，就有相应的一系列的基分类器，这些基分类器必须是相同类型的分类器．构建分类器的时候，直接用训练集依次训练每一个基分类器，这样就得到了分类器全体(committees)．在分类的时候，利用各个基分类器对该条测试例进行分类，并将各个基分类器的分类结果累加，最后再除上分类器的个数，将结果作为整个分类器全体的分类结果．这里面，在分类的时候就利用了投票(Vote)来进行对某一条测试例进行分类，其中投票不包含各个分类器的加权值，每一个基分类器对于分类的作用都一样．

它们之间的关系如图 5.3 所示．

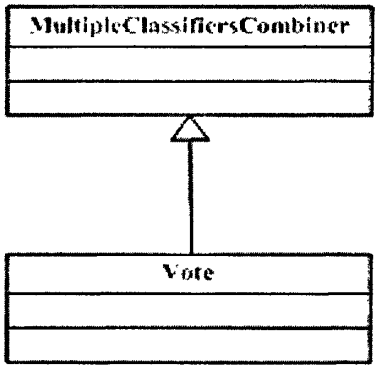

图5.3 基本的组合分类器的元学习
Fig 5.3 basic Meta-Learing of classifier combination

(3) 迭代生成基分类器的元学习算法．在 weka.classifiers.meta 下包含了常见基本的迭代生成基分类器的元学习算法，迭代生成的基分类器应属于同一分类模型，每次迭代生成的基分类器都依赖于上次迭代生成的分类器的分类性能．所包含的文件如下．

AdditiveRegression.java 文件：该元学习算法提升了基于回归的分类器的性能．每次迭代产生的分类器都要考虑上次迭代产生的分类器分类的剩余误差．对实例预测时把所有分类器的预测都相加在一起，并通过变化学习率参数来平滑预测结果．它们的关系如图 5.4 所示．

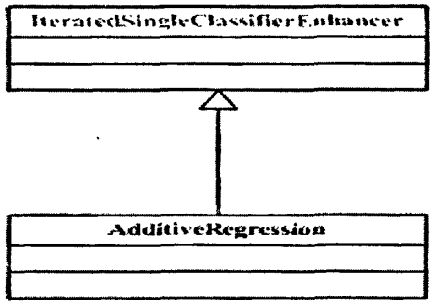

图5.4 迭代与随机生成分类器组合
Fig 5.4 basic Meta-Learing of interated creation of classifiers

(4) 随机生成单分类模型的学习算法. 在 weka.classifiers.meta 包下根据随机种子随机化生成单基分类器的元学习算法. 它们的关系如图 5.5 所示.

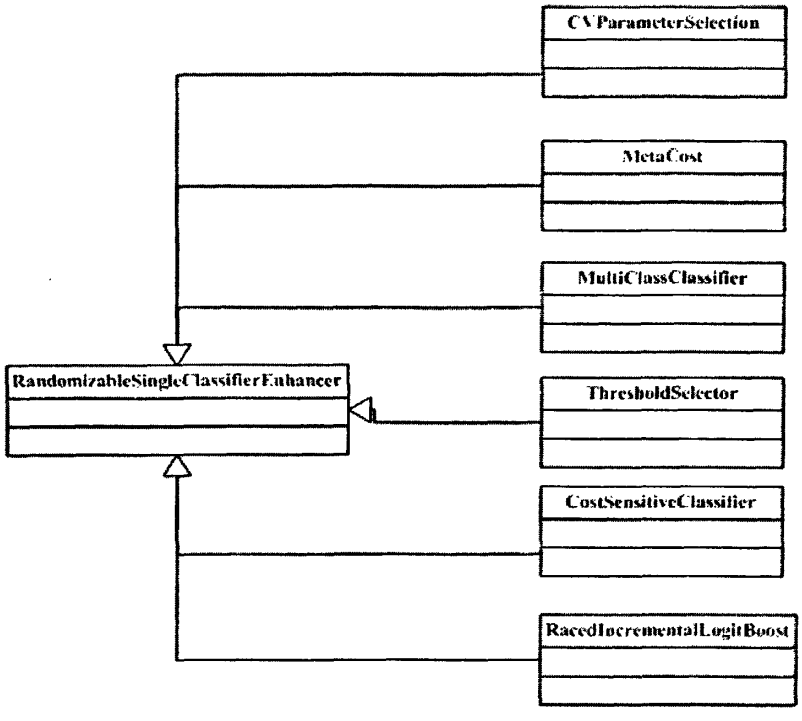

图5.5 迭代与随机生成分类器组合
Fig 5.5 the iteration and random of classifier combination

CVParameterSelection.java 文件：该算法能能通过交叉验证的方式对任何分类器进行参数选择.

MetaCost.java 文件：该算法使用具体的方法使它的基分类器成为成本敏感的

分类器.

MultiClassClassifier.java 文件：该算法能够使用解决两类问题的分类器来解决多类问题.

ThresholdSelector.java 文件：该算法为基分类器输出的概率选择一个阈值，用来优化分类性能检验的标准. 通常这个检验指的是 F 检验.

CostSensitiveClassifier.java 文件：该算法使它的基分类器成为成本敏感的. 通常有两种方法可以用来引进成本敏感：根据每一个类值的总成本调整训练实例的权重；或者根据最小分类错误成本来预测类标.

RacedIncrementalLogitBoost.java 文件：该算法产生通过竞争对数增加的委员会方法来增量地学习大规模的数据集的分类器.

(5) 随机生成多分类模型的元学习算法. 在 weka.classifiers.meta 包下根据随机种子随机化生成多分类器的元学习算法. 它们的关系如图 5.6 所示.

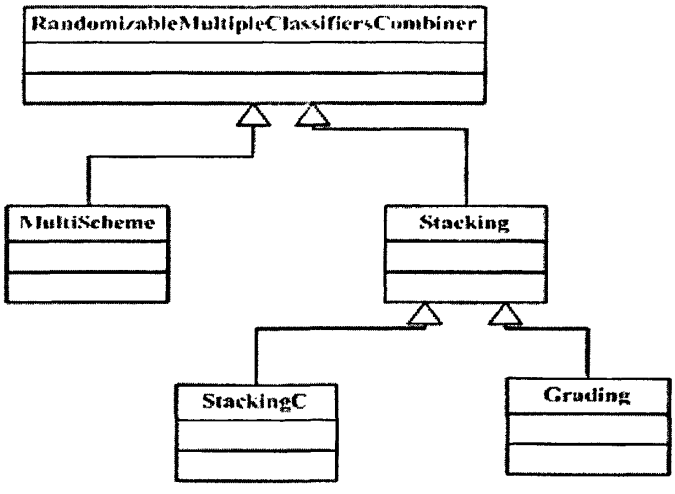

图5.6 随机生成多分类器模型
Fig 5.6 randomized creation of mutlple classifiers

MultiScheme.java 文件：该算法从多种分类器中选择一个最好的作为当前分类器，来对实例进行分类. 选择最佳分类器的评价标准是根据交叉验证的结果或者是各个基分类器的分类结果来取决.

Stacking.java 文件：该算法应用来组合不同学习算法所建的分类器. 通常的做法是使用交叉验证法估计每一个分类器的期望误差率，然后从中选择一个最好的分类器用于在未来的数据上做预测.

StackingC.java 文件：该算法继承与 Stacking 类，是 Stacking 算法的一种更有效的实现方法.

Grading.java 文件：该算法也继承与 Stacking 类，grading 可以看成是通过交叉验证选择的推广. Grading 可以解释为将错误特征技术转换成通过一个元分类器全体而形成的强分类器学习算法的技术.

(6) 随机迭代生成单分类模型的元学习算法

在 weka.classifiers.meta 包下根据随机种子随机化生成多分类器的元学习算法. 它们的关系如图 5.7 所示.

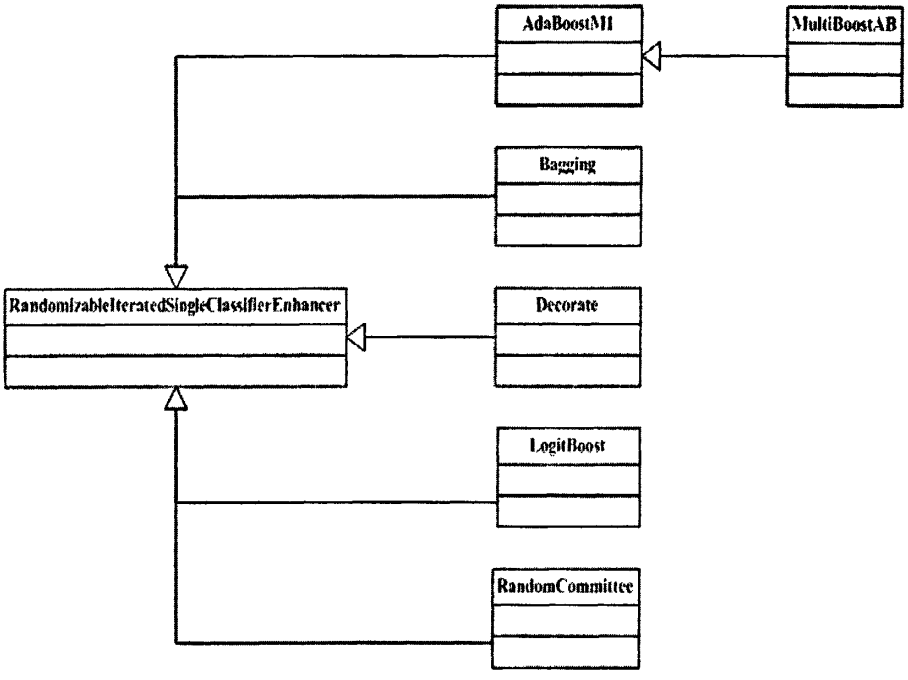

图5.7 随机迭代生成单分类模型
Fig 5.7 randomized creation of single classification model

AdaBoostM1.java 文件：该算法迭代地生成同一分类模型的多个分类器. 每次迭代过程中根据训练数据的权重，随机生成一个训练数据集. 根据当前是否被当前分类器分对来调整训练实例的权重，使被分错的实例尽可能得出现在下次迭代产生的训练集中，使分类器重点关注这些被上次分类器分错的实例.

Bagging.java 文件：该算法主要的思想就是先有一个预测分类器，然后选择数据集来对这个分类器进行训练，利用多个数据集训练形成多个有差异的预测分类器，最后将这多个预测分类器的预测结果通过投票（voting）来进行组合形成最后

的预测结果．该算法与 AdaBoostM1 的区别在于不考虑实例的权重，当前分类器的生成无需考虑上一个分类器的分类性能，因此可以并行地生成多个分类器．

MultiBoostAB.java 文件：Mutiboost 方法是一种将 boosting 和 bagging 结合在一起的方法．该方法将 bagging 结合到 boosting 上去，使得 Mutiboost 具有 bagging 的并行处理能力，还兼有 boosting 的高精度优点．

Decorate.java 文件：该算法通过使用附加人工构建的训练样本直接创建有差异性的弱假设．该技术方法就是能够使用任何强的学习器作为基分类器来构建有差异性的组合分类器(committee)的简单的、普通的元学习方法．

LogitBoost.java 文件：该算法使用了叠加逻辑回归．

RandomCommittee.java 文件：该算法创建了一个由随机产生的基分类器组成的委员会．这些基分类器必须实现 RandomizableClassifier 类的接口．

5.3 实验方法和实验数据描述

5.3.1 实验方法描述

本文所进行的实验在 Weka3-4-13 平台下完成，分类器选用 C4.5 决策树分类器，分别对 GreedSelection 以及 DynamicAdjust 算法进行了实验．实验方式是对每一个数据集采用 10 重交叉验证运行 5 次，然后取其分类错误率的平均结果作为最终的分类性能，平均分类错误率越低表示分类效果越好．

数据集运行 5 次的方法是将随机数种子改变，分别在 seed 取 1、3、5、7、11 下运行．随机数种子不同，交叉验证时选用的训练集和测试集也将不同．

数据集选取了 UCI[54]上的数据集，这些数据集都含有名称型属性，其中有些数据集中只含有名称型属性，有些数据集则是既有数值型属性又有名称型属性的混合型数据集．

5.3.2 数据集描述

本次实验一共选取了 25 个数据集，所有的数据集都来自于 UCI 站点．各数据集描述如表 5.1．

总共使用的 25 个数据集的描述如表 5.1 所示，描述的方面包括数据集中实例的条数、数据集中实例所拥有的属性个数、数值型属性的个数，非数值型属性的

个数、类标的个数等等.

本次所采用的数据集给出的这些数据还是原始的数据，经过预处理之后才能用于训练和测试.

表 5.1 数据集描述

Table 5.1 the description of datasets

数据集名称	实例数	属性数	数值型属性数	名称型属性数	类标数
audio	226	69	0	69	24
balance-scale	625	4	4	0	3
balloons	76	4	0	4	2
bcwo	699	9	9	0	2
breast-cancer	286	9	0	9	2
crx	690	15	6	9	2
echocardiogram	131	6	5	1	2
glass7A	214	10	10	0	7
hepatitis	155	19	6	13	2
horse-colic	368	21	8	13	2
House-vote-84	435	16	0	16	2
hungarian	294	13	6	7	2
hypothyroid	3163	25	7	18	2
iris	150	4	0	4	3
Kr-vs-kp	3196	36	0	36	2
labor	57	16	8	8	2
led	1000	7	0	7	10
lyn	148	18	0	18	4
pendigits	10992	16	16	0	10
segment	2310	19	19	0	7
sign	12546	8	8	0	3
solarflare-x	1389	10	0	10	3
splice-c4.5	3177	60	0	60	3
ttt	958	9	0	9	2
wine	178	13	13	0	3

5.4 GreedSelection 算法实现与实验结果分析

我们采用上面表格 5.1 所示的 25 个数据集训练和测试分类器，使用到的分类器分别是 GreedSelection 分类器算法，传统的 Boosting 算法以及决策树 C4.5 算法，记录每一个分类器运行后的分类正确率.

表 5.2 GreedSelection 方法实验结果

Table 5.2 The experimental results of GreedSelection method

数据集名称	Seed=1	Seed=3	Seed=5	Seed=7	Seed=11	平均
audio	84.0708	84.0708	84.0708	82.3009	85.3982	83.9823
balance-scale	77.1200	78.0800	76.4800	76.6400	77.4400	77.1520
balloons	77.6316	81.5789	80.2632	78.9474	80.2632	79.7369
bcwo	95.7082	96.1373	95.5651	96.1373	96.8526	96.0801
breast-cancer	59.4406	65.3846	67.8322	64.6853	67.1329	64.8951
crx	85.2174	84.2029	84.6377	83.6232	83.6232	84.2609
echocardiogram	59.5420	64.8855	65.6489	64.1221	66.4122	64.1221
glass7A	73.3645	73.3645	75.7009	75.7009	72.4299	74.1121
hepatitis	83.2258	82.5806	85.1613	83.2258	85.1613	83.8710
horse-colic	79.8913	80.1630	79.3478	80.4348	82.3370	80.4348
House-vote-84	95.4023	95.4023	94.0230	96.0920	94.7126	95.1264
hungarian	82.3129	79.9320	80.2721	76.8707	78.2313	79.5238
hypothyroid	98.8618	99.0515	98.9883	99.1148	98.8618	98.9756
iris	93.3333	94.0000	94.0000	94.6667	92.0000	93.6000
Kr-vs-kp	99.7497	99.6871	99.5932	99.4994	99.6245	99.6308
labor	84.2105	85.9649	87.7193	80.7018	85.9649	84.9123
led	72.6000	73.2000	73.6000	72.1000	73.4000	72.9800
lyn	81.7568	80.4054	83.7838	82.4324	81.0811	81.8919
pendigits	98.9993	99.0721	98.8355	99.0357	98.9902	98.9866
segment	97.9221	98.2251	97.9654	98.1818	98.3117	98.1212
sign	86.9201	86.8165	86.8165	87.1911	87.4382	87.0365
solarflare-x	98.7761	98.7761	98.6321	98.8481	98.4161	98.6897
splice-c4.5	94.4287	94.0195	94.3972	94.3658	94.1454	94.2713
ttt	95.7203	96.5553	96.2422	96.0334	96.2422	96.1587
wine	95.5056	98.8764	97.1910	97.1910	96.0674	96.9663

首先，我们在数据集上测试 GreedSelection 算法，该分类器在所有数据集上每

一次的运行结果在如表 5.2 中所示.

在测试完 GreedSelection 算法之后，我们测试原有的经典 Boosting 方法分类器的分类精度，采用的实验方法和上面的一样，也是采用 10 重交叉验证，运行 5 遍，并给出最后的平均分类精度. 经典 Boosting 方法分类的实验结果由下面的表 5.3 给出.

表 5.3 经典 Boosting 方法实验结果

Table 5.3 The experimental results of original boosting method

数据集名称	Seed=1	Seed=3	Seed=5	Seed=7	Seed=11	平均
audio	84.5133	84.0708	83.6283	81.4159	84.9558	83.7168
balance-scale	75.5200	77.2800	75.8400	75.8400	75.6800	76.0320
balloons	69.7368	59.2105	63.1579	67.1053	67.1053	65.2632
bcwo	95.5651	96.2804	95.2790	95.9943	96.4235	95.9085
breast-cancer	58.7413	62.9371	66.7832	65.3846	65.7343	63.9161
crx	84.4928	83.9130	84.2029	83.0435	82.8986	83.7102
echocardiogram	62.5954	63.3588	64.8855	62.5954	64.8855	63.6641
glass7A	75.2336	73.3645	74.7664	74.2991	73.8318	74.2991
hepatitis	82.5806	81.2903	85.1613	82.5806	85.1613	83.3548
horse-colic	74.7283	73.9130	77.1739	77.9891	76.0870	75.9783
House-vote-84	95.4023	94.7126	94.4828	95.4023	93.7931	94.7586
hungarian	77.2109	78.5714	76.8707	75.8503	75.1701	76.7347
hypothyroid	98.7038	98.8618	98.5457	98.7038	98.6721	98.6974
iris	93.3333	93.3333	94.0000	94.0000	92.0000	93.3333
Kr-vs-kp	99.6558	99.6558	99.6558	99.4681	99.5932	99.6057
labor	80.7018	87.7193	87.7193	80.7018	82.4561	83.8597
led	69.1000	68.7000	67.8000	68.7000	67.3000	68.3200
lyn	80.4054	79.0541	83.1081	81.0811	79.7297	80.6757
pendigits	98.9720	99.0175	98.7718	99.0084	98.9174	98.9374
segment	97.8355	98.3117	98.0087	98.1818	98.2684	98.1212
sign	85.5332	85.4695	85.4854	85.9955	86.3144	85.7596
solarflare-x	97.2642	97.3362	97.4082	97.4802	97.3362	97.3650
splice-c4.5	93.7677	93.8307	93.8307	93.5788	93.5788	93.7173
ttt	96.2422	96.6597	95.9290	96.3466	96.4509	96.3257
wine	95.5056	98.3146	97.1910	96.6292	96.6292	96.8539

接下来在测试完经典的 Boosting 算法之后，我们测试原有的 J48 方法分类器的分类精度，在此省略该表，仅给出最后的评价正确率．为了方便比较三种算法的优劣，本文将这三种算法的在不同种子下的平均正确率，在不同数据集合上的平均正确率，以及每个正确率之间的差异以表的形式提供．具体数据如表 5.4 所示．

表 5.4 三种方法实验结果比较

Table 5.4 the comarision of three method's classification result

数据集	GS	BT	J48	GS-BT	GS-J48	BT-J48
audio	**83.9823**	83.7168	77.5221	0.2655	6.4602	6.1947
balance-scale	77.1520	76.0320	**77.6640**	1.1200	-0.5120	-1.6320
balloons	**79.7369**	65.2632	69.4737	14.4737	10.2632	-4.2105
bcwo	**96.0801**	95.9085	94.7640	0.1716	1.3161	1.1445
breast-cancer	64.8951	63.9161	**74.2657**	0.9790	-9.3706	-10.3496
crx	84.2609	83.7102	**85.9131**	0.5507	-1.6522	-2.2029
echocardiogram	**64.1221**	63.6641	62.9008	0.4580	1.2214	0.7633
glass7A	74.1121	**74.2991**	66.9159	-0.1869	7.1962	7.3832
hepatitis	**83.8710**	83.3548	80.1290	0.5161	3.7419	3.2258
horse-colic	80.4348	75.9783	**84.5109**	4.4565	-4.0761	-8.5326
House-vote-84	95.1264	94.7586	**95.3103**	0.3678	-0.1839	-0.5517
hungarian	79.5238	76.7347	**80.5442**	2.7891	-1.0204	-3.8095
hypothyroid	98.9756	98.6974	**99.2475**	0.2782	-0.2719	-0.5501
iris	93.6000	93.3333	**94.8000**	0.2667	-1.2000	-1.4667
Kr-vs-kp	**99.6308**	99.6057	99.4493	0.0250	0.1815	0.1564
labor	**84.9123**	83.8597	78.2456	1.0526	6.6667	5.6140
led	72.9800	68.3200	**73.0000**	4.6600	-0.0200	-4.6800
lyn	**81.8919**	80.6757	77.7027	1.2162	4.1892	2.9730
pendigits	**98.9866**	98.9374	96.5520	0.0491	2.4345	2.3854
segment	**98.1212**	**98.1212**	96.7359	0.0000	1.3853	1.3853
sign	**87.0365**	85.7596	85.1889	1.2769	1.8476	0.5707
solarflare-x	98.6897	97.3650	**99.1361**	1.3247	-0.4464	-1.7711
splice-c4.5	**94.2713**	93.7173	93.9817	0.5540	0.2896	-0.2644
ttt	96.1587	**96.3257**	84.8434	-0.1670	11.3152	11.4822
wine	**96.9663**	96.8539	93.2584	0.1124	3.7079	3.5955
平均值	86.6207	85.1563	84.8822	1.4644	1.7385	0.2741

接下来就该表内容对实验结果进行分析，表中的黑体表示是该数据集合上最优的分类器．首先比较传统的 Boosting 算法和 J48 算法之间的优劣．由表 5.4 可以得出，在某些数据集合如 balloons，ttt 上，传统 Boosting 方法相比 J48 算法分类性能明显地提升分类性能，都在 10%左右，显示出组合分类器的优势．但是，进一步看，在总共 25 个数据集合上，传统的 Boosting 算法共在 13 个数据集合上的平均正确率比 J48 算法高，而在 12 个数据集合上比它差．就在所有的数据集合的平均分类精度上来说，传统的 Boosting 算法的平均分类精度为 85.1563%，J48 的平均分类精度为 84.8822%，两者相差 0.2741%．从这一点上看，经过传统 Boosting 方法优化过的 J48 算法的性能虽然分类精度有所增大，但并没有较大的提升，这也可能跟所选用的基分类器或数据集合有关．另外，从每个数据集上的最优分类器的个数比较，传统 Boosting 方法为 3 个，而 J48 算法为 10 个，从这一点上看经过 Boosting 算法优化过的 J48 算法反倒不如原来的 J48 算法．

分析造成这种 Boosting 算法性能并不能显著提升 J48 分类器的分类性能的原因，其主要的可能为：一是受基分类器的影响，因为 J48 分类器本身就是一种分类性能非常好的分类器，对其做一些已经不能再明显地提升其分类性能，反而有可能降低其性能．第二可能是受所选的训练集合有关，因为本实验仅仅选择了 25 个数据集合，而现实环境中存在着海量的数据，在这 25 个数据集合上的实验结果只能是有可能，而并不一定代表现实环境海量数据下的分类器性能．第三点，也是本文特别关注的一点，就是关于多分类器组合的多样性对分类正确率的影响，即只有当组合的多样性高时，其分类正确率才往往较好．在 Boosting 算法产生多分类器组合时，并没有利用多样性标准来衡量组合的质量，指导组合的生成使组合朝着多样性高的趋势发展．因此，在这种情况下产生的多分类器组合中基分类器之间相似度就有可能比较大，当一个基分类器对一个实例做出错误分类时，并没有其他的基分类器能对这个错误分类进行纠正．甚至很有可能在一个分类器做出正确分类后，而其他多个相似的基分类器做出错误的分类，最后仍导致错误分类．这样就降低的组合的分类精度．

接下来，对利用多样化进行基分类器选取的 GreedSelection 算法和传统的 boosting 算法的分类结果进行比较．由表 5.4 看出，在全部的 25 个数据集中，在全部集合上的平均正确率上，GreedSelction 为 86.6207%，boosting 为 85.1563%．GreedSelection 在 22 个数据集合上的正确率比 boosting 高，在 1 个数据集上跟 boosting 正确率相等，仅在 2 个数据集 glass7A, ttt 上正确率稍微低于 boosting 算法．GreedSelection 算法占据了决对性的优势． 其中在 balloons 数据集上性能提升明显，接近 15%．这表明经过多样性选择后，组合的性能在基分类器

和数据集都不变的情况下，分类性能得到了普遍的提升. 再将 GreedSelection 和 J48 比较，在全部的 25 个数据集上，GreedSelection 在其中的 15 个数据集上比 J48 的分类正确率要高，这高于 boosting 算法在和 J18 算法比较的 13 个. 另外，在所有数据集上的平均正确率增加了 1.7385%，而 boosting 算法仅增加了 0.2741%. 在每个数据集合的最优分类器上，GreedSelection 与 J48 为 13 比 10，也远高于 Boosting 与 J48 的 3 比 10. 这些实验结果对上述分析的该试验中 boosting 算法没有能显著提高 J48 分类器的可能原因进行了验证. 在基分类器和数据集合都没有改变的情况下，通过多样性选择能够显著地提高分类正确率，表明了组合的多样性是决定多分类器组合分类性能是否优异的一个重要因素. 多样性较好的多分类器组合往往具有较好的分类性能.

但是 GreedSelection 算法的一个潜在问题是，其在多样性降低时就停止加入基分类器. 然而再继续加入基分类器后有可能多样性会继续提高，因此该算法并不总能选取多样性最高的基分类器子集. 如何改善搜索过程，使其能找出多样性最高的子集是要继续研究的问题.

5.5 DynamicAdjust 算法实现与实验结果分析

该算法的实验分析手段与上节对 GreedSelection 算法的分析手段相同. 采用上面表格 5.1 所示的 25 个数据集训练和测试分类器，使用到的分类器分别是 DynamicAdjust 分类器算法，传统的 Boosting 算法以及决策树 C4.5 算法，记录每一个分类器运行后的分类正确率.

我们将分类器运行 5 遍，每一次改变随机数种子，以影响分类器在交叉验证时候对于数据的划分，随机数种子分别取 1、3、5、7、9，评价分类器采用 10 重交叉验证的方式，然后再将这 5 个结果的值累加再除掉 5，得出分类器的平均分类正确率，作为该分类器在数据集上的分类精度.

首先，我们在数据集上测试 DynamicAdjust 算法，该分类器在所有数据集上每一次的运行结果在如下表 5.5 中所示.

为了方便比较这三种算法的优劣，本文将这三种算法的在不同种子下的平均正确率，在不同数据集合上的平均正确率，以及每个正确率之间的差异以表的形式提供. Boosting 算法与 J48 算法的分类结果如表 5.3 和 5.4 所示. 具体数据如表 5.6 所示.

表 5.5 DynamicAdjust 方法实验结果

Table 5.5 experimental results of DynamicAdjust method

数据集名称	Seed=1	Seed=3	Seed=5	Seed=7	Seed=11	平均
audio	84.9558	84.0708	83.6283	82.7434	84.5133	83.9823
balance-scale	76.1600	77.9200	75.5200	75.8400	75.2000	76.1280
balloons	72.3684	61.8421	68.4211	69.7368	67.1053	67.8947
bcwo	95.5651	96.1373	95.2790	95.9943	96.4235	95.8798
breast-cancer	59.4406	63.2867	66.7832	65.3846	66.7832	64.3357
crx	84.4928	83.9130	84.2029	83.0435	82.8986	83.7102
echocardiogram	60.3053	63.3588	64.1221	62.5954	67.1756	63.5114
glass7A	74.2991	72.8972	72.8972	76.6355	73.3645	74.0187
hepatitis	80.6452	81.9355	85.8065	82.5806	85.1613	83.2258
horse-colic	73.9130	74.7283	77.4457	78.5326	77.1739	76.3587
House-vote-84	94.2529	93.5632	92.6437	92.8736	93.1034	93.2874
hungarian	78.2313	77.5510	77.2109	77.5510	75.5102	77.2109
hypothyroid	98.7986	98.9251	98.9251	98.7986	98.7670	98.8429
iris	93.3333	94.0000	94.0000	94.0000	92.0000	93.4667
Kr-vs-kp	99.6558	96.9024	99.4994	98.8736	98.1227	98.6108
labor	82.4561	87.7193	91.2281	84.2105	84.2105	85.9649
led	66.8000	68.3000	67.9000	67.4000	68.4000	67.7600
lyn	82.4324	79.0541	83.7838	81.7568	80.4054	81.4865
pendigits	98.9447	99.0539	98.7991	98.9538	98.9083	98.9320
segment	97.9654	98.3983	98.0087	98.1385	98.2684	98.1559
sign	86.4180	85.9557	86.0434	86.1948	86.4977	86.2219
solarflare-x	97.9122	97.9842	98.0562	97.9842	97.9122	97.9698
splice-c4.5	94.0825	93.7362	93.7362	93.6103	93.7677	93.7866
ttt	96.2422	96.6597	95.9290	96.3466	96.4509	96.3257
wine	96.0674	98.8764	97.1910	97.1910	95.5056	96.9663

与前面的算法分析类似，表中的黑体表示是该数据集合上最优的分类器．首先比较传统的 Boosting 算法和 J48 算法之间的优劣．该分析比较已经在前面算法

的比较中分析过，下面仅比较一下 DynamicAdjust 算法与 boosting 算法的优劣.

表 5.6 三种方法实验结果

Table 5.6 The comarision of three method's classification result

数据集名称	DA	BT	J48	DA-BT	DA-J48	BT-J48
audio	**83.9823**	83.7168	77.5221	0.2655	6.4602	6.1947
balance-scale	76.1280	76.0320	**77.6640**	0.0960	-1.5360	-1.6320
balloons	67.8947	65.2632	**69.4737**	2.6316	-1.5790	-4.2105
bcwo	95.8798	**95.9085**	94.7640	-0.0286	1.1158	1.1445
breast-cancer	64.3357	63.9161	**74.2657**	0.4196	-9.9300	-10.3496
crx	83.7102	83.7102	**85.9131**	0.0000	-2.2029	-2.2029
echocardiogram	63.5114	**63.6641**	62.9008	-0.1527	0.6106	0.7633
glass7A	74.0187	**74.2991**	66.9159	-0.2804	7.1028	7.3832
hepatitis	83.2258	**83.3548**	80.1290	-0.1290	3.0968	3.2258
horse-colic	**76.3587**	75.9783	84.5109	0.3804	-8.1522	-8.5326
House-vote-84	93.2874	94.7586	**95.3103**	-1.4713	-2.0229	-0.5517
hungarian	77.2109	76.7347	**80.5442**	0.4762	-3.3333	-3.8095
hypothyroid	98.8429	98.6974	**99.2475**	0.1454	-0.4046	-0.5501
iris	93.4667	93.3333	**94.8000**	0.1333	-1.3333	-1.4667
Kr-vs-kp	98.6108	**99.6057**	99.4493	-0.9950	-0.8385	0.1564
labor	**85.9649**	83.8597	78.2456	2.1052	7.7193	5.6141
led	67.7600	68.3200	**73.0000**	-0.5600	-5.2400	-4.6800
lyn	**81.4865**	80.6757	77.7027	0.8108	3.7838	2.9730
pendigits	98.9320	**98.9374**	96.5520	-0.0055	2.3800	2.3854
segment	**98.1559**	98.1212	96.7359	0.0346	1.4200	1.3853
sign	**86.2219**	85.7596	85.1889	0.4623	1.0330	0.5707
solarflare-x	97.9698	97.3650	**99.1361**	0.6048	-1.1663	-1.7711
splice-c4.5	93.7866	93.7173	**93.9817**	0.0692	-0.1951	-0.2644
ttt	**96.3257**	**96.3257**	84.8434	0.0000	11.4823	11.4823
wine	**96.9663**	96.8539	93.2584	0.1124	3.7079	3.5955
平均值	85.3613	85.1563	84.8822	0.2050	0.4791	0.2741

出表 5.6 可以得出，在总共 25 个数据集合上，DynamicAdjust 算法共在 16 个数据集合上的平均正确率比 boosting 算法算法高，在一个数据集上相同，而在 8

个数据集合上比它差．这表明在大部分的情况下，经过动态赋权重的优化过的算法能提高原 boosting 算法的分类正确率．就在所有的数据集合的平均分类精度上来说，DynamicAdjust 算法的平均分类精度为 85.3613%，传统的 Boosting 算法的平均分类精度为 85.1563%，两者相差 0.2050%．从每个数据集上的最优分类器的个数比较，DynamicAdjust 算法为 8 个，传统 Boosting 方法为 7 个，而 J48 算法为 10 个．从这一点上看，经过动态赋权重方法优化过的 boosting 算法的性能虽然分类精度有所增大，但性能提升并不明显．

从所用的数据集中属性个数，类值个数和实例个数等特征以及算法模型等方面来对这种实验结果进行分析．首先，从实验结果上来看，在数据集和基分类器都不变的情况下，动态赋值权重算法比传统的 boosting 算法相比，确实在大多数情况下都具有更高的分类正确率．也就验证了本文对原 boosting 算法可能存在的问题的假设．即静态赋值方法并没有考虑到当前测试实例的取值情况，给一个对当前测试实例可能错误分类的基分类器赋与较大权重反而会降低分类正确率．应该首先判断当前测试实例被每个基分类器正确分类的可能性大小，然后按这种大小的度量对基分类器赋予权重．其次，该算法的根据被测试实例属于每个基分类器分错的实例集合的可能性大小为基础来分配权重，由于每个基分类器所选取的实例随概率抽取的，所以每个基分类器分错的实例很可能相差并不大，这就会对导致对所有基分类器赋的权重相差不大，反而减弱了 boosting 算法的加权赋值效果．另外该算法采用了基于距离的公式来计算测试实例属于被错分子集的可能性大小，易受实例个数，实例属性取值等的影响．因此，在以后的研究中，如何设计合适的权重赋值公式，使其能不受上述的影响并能明显体现各个基分类器之间的差异，是要继续研究的一个问题．

6 结论

本章将总结本文的工作，并指出了本文算法中尚存在的不足之处。并提出进一步改进工作的一些建议.

6.1 研究工作总结

多分类器组合通过组合多个基分类器来进行分类的. 实验表明，多分类器组合跟单分类器相比，能明显提分类器的升分类性能. 因此，多分类器组合一直是机器学习及数据挖掘领域中一个重要的研究问题.

本文主要研究了多分类器组合的多样性，如何利用多样性进行基分类器选取和动态权重赋值问题. 多样性是组合一个重要的因素，本文在分析了多样性的度量的基础上，提出了一种利用多样性进行基分类器选取的方法. 另外，针对 Boosting 算法静态权重赋值可能存在的问题，本文还提出了一种新的动态权重赋值的方法. 然后通过实验比较分析了这两种算法的性能. 本文所做的工作主要如下：

(1) 阐述了多分类器组合的概念，及关于多分类器组合的主要关键问题. 包括：基分类器的生成策略，多分类器组合的多样性评价，基分类器的选取问题，以及基分类器的组合方法. 本文详细介绍了这些问题的基本理论和当前的典型算法.

(2) 针对基分类器的选取问题，提出了一种在 Boosting 方法基础上，利用多样性度量方法进行基分类器选取的具体算法.

(3) 针对 Boosting 算法中基分类器权重的静态赋值可能带来的问题，利用动态选取思想，提出了一种利用当前测试实例的取值来动态地为基分类器赋予权重的方法.

(4) 通过了一系列的实验，比较了上述提出的两种算法与原 Boosting 算法和 J48 决策树分类器算法等的性能比较，验证了这两种算法相比原来的算法具有较好的性能. 分析了这两种算法较好或较差性能表现的原因，通过实验表明了多样性是影响多分类器组合分类性能的重要因素，以及动态权值分配能够提高组合的分类性能.

6.2 进一步研究的考虑

多样性是衡量多分类器组合的一个重要标准，然而目前为止却并没有一个统

一的多样性定义．研究人员已经提出了多种多样性度量方法，但实验效果并不理想．因此，有待于对多样性的度量方法有待于进一步研究．在接下来的研究中，将进一步对多样性度量方法进行研究并应用其指导多分类器组合的生成．另外，将继续对高效的基分类器选取算法进行研究，使得能快速有效地选取出分类性能良好的多分类器组合．

参考文献

[1] Dietterich, T. G. Machine-Learning Research: Four Current Directions. AIMAGAZINE, WINTER, 1997. pp. 97-136.

[2] Mitchell, T. M. Machine Learning. New York: The McGraw-Hill Companies, Inc., 2001. pp. 25-35.

[3] Han, J. Kamber, M.. 范明, 孟小峰译. 数据挖掘概念和技术, 第二版. 北京, 机械工业出版社, 2006.

[4] Quinlan, J. R. Induction of decision trees. Machine Learning, 1986. 1(1). pp. 81-106.

[5] Duda, R. O., Hart, P. E. and Stork, D.G. 李虹东, 姚天翔译. 模式分类, 第二版. 北京, 机械工业出版社, 2007. pp. 373-375.

[6] Breiman, L. Bagging predictors. Machine Learning. 1996. 24(2). pp. 123-140.

[7] Freund, Y., and Schapire, R. E. Experiments with a new boosting algorithm. Machine learning, Proceedings of the thirteenth international Conference (ICML 1996). Bari, Italy: Morgan Kaufmann, 1996. pp. 148-156.

[8] Ho, T. K. The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998, 20(8). pp. 832-844.

[9] Ship, C. A., and Kuncheva, L. I.. Relationships between combination methods and measures of diversity in combing classifiers. Information Fusion, 2002, 3(2). pp. 135-148.

[10] Dietterich, T.. Ensemble methods in machine learning. Multiple Classifier System(MCS 2000, LNCS 1857). Cagliari, Italy: Springer, 2000. pp. 1-15.

[11] Kohavi, R., and Wolpert, D. H.. Bias plus variance decomposition for zero-one loss functions. Machine Learning: Proceedings of the Thirteenth International Conference. 1996. pp. 275-283.

[12] Dietterich, T. G. . An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. Machine Learning, 2000, 40(2). pp. 139-157.

[13] 毛国君, 段立娟, 王实, 石云. 数据挖掘原理与算法, 第二版. 北京, 清华大学出版社, 2007. pp. 115-117.

[14] Kuncheva, L. I., and Whitaker, C. J.. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning, 2003, 51(2). pp. 181-207.

[15] Sharkey A.J.C. The "test and select" approach to ensemble combination. Multiple Classifier Systems. LNCS 1857. Springer-Verlag, 2000. pp. 30-44.

[16] Kuncheva, L. I. A theoretical study on six classifier fusion strategies. IEEE Transactions on Pattern and machine learning, 2001, 24(2). pp. 281-286.

[17] Tang, E. K., Suganthan, P. N., and Yao, X. An analysis of diversity measures. Machine learning, 2006, 65(1). pp. 241-271.

[18] Yule, G.. On the association of attributes in statistics. Phil. Trans., A, 194, 257–319.

[19] Kuncheva, L. I., Skurichina, M., and Duin, R. P. W. An experimental study on diversity for bagging and boosting with linear classifiers. Information Fusion, 2002, 3(4). pp. 245-258.

[20] Skurichina, M., Kuncheva, L. I., and Duin, R. P. W.. Bagging and boosting for the nearest mean classifier effects of sample size on diversity and accuracy. Multiple Classifier systems

(MCS 2002 LNCS 2364). Cagliari, Italy: Springer, 2002. pp. 307-311.

[21] Chung, Y., Hsu, D. F., and Tang, C. Y.. On the diversity-performance relationship for majority voting in classifier emsembles. Multiple Classifier Systems (MCS 2003, LNCS 4472). Prague, Czech Republic: Springer, 2007. pp. 407-420.

[22] Windeatt, T.. Diversity measures for multiple classifier system analysis and design. Information Fusin, 2005, 6(1). pp. 21-36.

[23] Tsymbal, A., Pechenizkiy, M, and Cunningham, P.. Diversity in search strategies for ensemble feature selection. Information Fusion, 2005,6(1). pp. 83-98.

[24] Golestani, A., Ahmadian, K., and Amiri, A. et al.. A novel adaptive-boost-based strategy for combining classifiers using diversity concept. 6th Annual IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007). Melbourne, Australia: IEEE Computer Society, 2007. pp. 128-134.

[25] Banfield R. E., Hall, L. O., and Bowyer, K. W. et al. Ensemble diversity measures and theire application to thinning. Information Fusion, 2005, 6(1). pp. 49-62.

[26] Dos Santos, E.M., Sabourin, R., and Maupin, P.. Ambiguity-guided dynamic selection of ensemble of classifiers. 10th International Conference on Information Fusion, 2007. pp. 1-8.

[27] Gabrys, B., and Ruta, D.. Genetic algorithms in classifier fusion. Applied Soft Computing, 2006, 6(4). pp. 337-347.

[28] Wolpert, D.H.. Stacked generalization. Neural Networks, 1992,5(1). pp. 241-259.

[29] Guerra-Salcedo, C., Whitley, D.. Genetic approach to feature selection for ensemble creation. Proceeding of the Genetic and Evolutionary Computation Conference (GECCO 1999). Orlando, Florida: Morgan Kaufmann, 1999. pp. 236-243.

[30] Kuncheva, L. I.. Designing classifier fusion systems by genetic algorithms. IEEE Transactions on Evolutionary Computation, 2000, 4(4). pp. 327-336.

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

[32] Melville, P., and Mooeny, R. J.. Constructing diverse classifier ensembles using artificial training examples. Proceedings of the seventeenth international joint Conference on artificial Intelligence. San Francisco, CA: Morgan Kaufmann, 2003. pp. 505-512.

[33] Webb, G. I.. MultiBoosting: a technique for combining boosting and wagging. Machine Learning, 2000, 40(2). pp. 159-196.

[34] Kotsianti, S. B., and Kanellopoulos, D.. Combining bagging, boosting and dagging for classification problems. Knowledge-Based Intelligent Information (KES 2007, LNAI 4693). Vietri sul Mare, Italy: Springer, 2007. pp. 493-500.

[35] Guruswami, V., and Sahai, A.. Multiclass learning, boosting and error-correcting codes. Proceedings of the Twelfth Annual Conference on Computational Learning Theory (COLT 1999). Santa Cruz, CA: ACM, 1999. pp. 145-155.

[36] Partridge,D., Yates, W. B. Engineering multiversion neural-net systems. Neural Computation 1996, 8. pp. 869-893.

[37] Roli, F., Giacinto, G., and Vernazza, G.. Methods for designing multiple classifier systems. Multiple Classifier systems (MCS 2001 LNCS 2096). Cambridge, UK: Springer, 2001. pp. 78-87.

[38] Kim, Y., Street, W. N., and Menczer, F.. Optimal ensemble construction via

meta-evolutionary ensembles. Experts System with Applications, 2006, 30(4), . pp. 705-714.

[39] Giacinto, G., and Roli, F.. Methods for dynamic classifier. Conference on Image Analysis and Processing (ICIAP 1999). Venice, Italy: IEEE Computer Society, 1999. pp. 659-664.

[40] Giacinto, G., and Roli, F.. A theoretical framework for dynamic classifier selection. International Conference on Pattern Recognition (ICPR 2000). Barcelona, Spain: IEEE Computer Society, 2000. pp. 2008-2011.

[41] Woods, K.. Combination of multiple classifiers using local accuracy estimates. IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 19, Number 4, 1997. pp. 405-410.

[42] Giacinto, G., and Roli, F.. Dynamic classifier selection based on multiple classifier behaviour. Pattern Recognition, 2001, 34(9). pp. 1879-1881.

[43] Eschirich, S., and Hall, L. O.. Soft partitions lead to better learned ensembles. Fuzzy Information Processing Society, 2002. pp. 406-411.

[44] Zhu, X., Wu X., and Yang, Y.. Dynamic classifier selection for effective mining from noisy data streams. Proceeding of the Fourth IEEE International Conference on Data Mining (ICDM 2004). Brighton, UK: IEEE Computer Society, 2004. pp. 305-312.

[45] Ko, A. H., Sabourin, R., and Jr, A.D. S. B.. A new dynamic ensemble selection method for numeral recognition. Multiple Classifier Systems (MCS 2007, LNCS 4472). Prague, Czech Republic: Springer, 2007. pp. 431-439.

[46] Shin, H. W., and Sohn, S. Y.. Combining both ensemble and dynamic classifier selection schemes for prediction of mobile internet subscribers. Expert Systems with Applications, 2003, 25(1). pp. 63-68.

[47] Xu, L., and Krzyzak, A.. Methods of combing multiple classifiers and their applications to handwriting recognition. IEEE Transactions on Systems, Man, and Cybernetics, 1992, 22(3). pp. 418-435.

[48] Kittler, J.. On combing classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998, 20(3). pp. 226-239

[49] J. R. Quinlan. C4.5: programes for Machine Learning. San Mateo, CA. Morgan Kaufmann. 1993.

[50] M. Mehta, R. Agrawal, J. Rissanen. SLIQ: A Fast Scalable Classifier for Data Mining. In Proceeding 1996 International conference Extending Database Technology (EDBT'96). Avignon. 1996, 3.

[51] J. Shafer, R. Agrawal, M. Mehta. SPRINT: A Scalable Parallel Classifier for Data Minging. In Proceeding 1996 International conference Very Large Data Bases (VLDB'96). Bombay. 1996, 9. pp. 544-555.

[52] J. Gehrke, R. Ramakrishnan, V. Ganti. Rainforest: A framework for Fast Decision Tree Construction of Large Datasets. In Proceeding 1998 International conference Very Large Data Bases (VLDB'98). New York. 1998, 8. pp. 416-427.

[53] Witten, I. H. and Frank, E. Data Mining—practical machine learning tools and techniques, second edition. 北京, 机械工业出版社, 2006.

[54] Asuncion, A. and Newman, D. J. UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

作者简历

付彬，男，1983 年生，河北邯郸人，主要研究方向为数据挖掘.

教育经历：2007.9-2009.7 北京交通大学，计算机科学与技术，工学硕士；

2003.9-2007.7 东北大学，软件工程，工学学士.