摘 要

基于贝叶斯技术的分类是当前数据挖掘领域的一个研究热点。本文从两个方面对贝叶斯分类模型进行了深入的研究：半朴素贝叶斯分类与增量贝叶斯分类。

半朴素贝叶斯分类模型对朴素贝叶斯分类模型的结构进行了扩展，其目的是为了突破朴素贝叶斯分类模型特征属性间独立性假设限制，提高分类性能。目前半朴素贝叶斯分类模型学习的关键是如何有效组合特征属性。针对已有的学习算法中存在的效率不高及部分组合意义不大的问题，本文提出了条件互信息度量半朴素贝叶斯分类学习算法(CMI-BSNBC)。运用实验数据进行了比较实验，实验取得了大量的数据。在分析结果的基础上得出了相应的结论，证明了模型的有效性。

增量贝叶斯分类模型的关键是测试实例的选择策略，本文研究的重点是如何充分利用训练集的先验知识并使其在学习过程中向前传递，提出了新的模型。新模型的基本思想是基于 0-1 分类损失用训练集对候选测试实例进行检验，这保证了与训练集相容性较好的测试实例被优先选择。

关键字：数据挖掘、贝叶斯理论、分类规则、信息熵、属性组合、主动学习策略。

Abstract

Classifying based on Bayes Technology has got more and more interests in the field of data mining. This thesis makes a study of two Bayesian classifying models which are Semi-Naïve Bayesian Classifier and Increasing Bayesian Classifier.

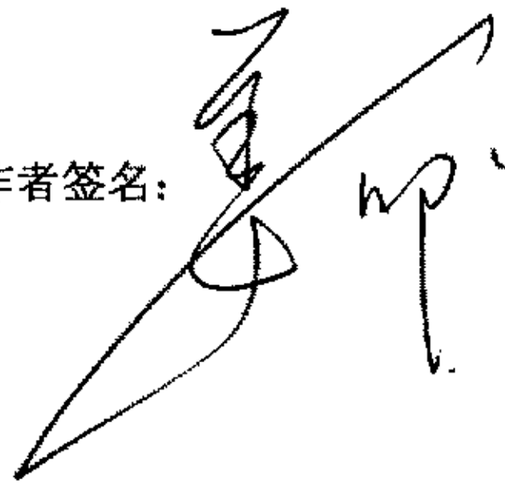
Semi-Naïve Bayesian Classifier extends the structure of Naïve Bayesian Classifier in order to get rid of the limit of the assumption of independence between feature attributes of Naïve Bayesian Classifier and improve the performance of classification. The key of model learning of Semi-Naïve Bayesian Classifier is how to combine feature attributes effectively. Since most algorithms are not effective and not very meaningful in combining, this thesis proposes an algorithm based on a kind of Semi-Naïve Bayesian Classifier which is measured by conditional mutual information(CMI-BSNBC). This thesis implements the CMI-BSNBC model and uses it to carry out series of comparing experiments on experimental data ,with plenty of resultant data been obtained .After synthetically analyzing the experimental result we make some conclusion which show the effectiveness of the model.

The key of Increasing Bayesian Classifier is the policy of how to choose test samples. This thesis studies how to make full use of prior knowledge and transmit it. The new model is presented which is based on the 0-1 loss of classification and uses training set to verify the test samples, which assures that the test sample more compatible with the training set be chosen firstly.

Key words: **D**ata Mining, Bayes Theory, Classification Rule, Information Entropy, Attribute Combining, Policy of Initiative Learning

独 创 性 声 明

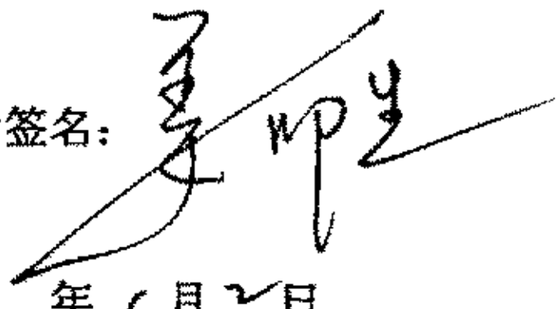
本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 合肥工业大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：  签字日期：2004 年 6 月 2 日

学位论文版权使用授权书

本学位论文作者完全了解 合肥工业大学 有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 合肥工业大学 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，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：  导师签名： 

签字日期：2004 年 6 月 2 日 签字日期：04 年 6 月 2 日

学位论文作者毕业后去向：

工作单位： 电话：

通讯地址： 邮编：

致 谢

论文是在我的导师王浩教授的悉心指导下完成的。感谢王老师三年来对我无微不至的关怀与孜孜不倦的教诲！王老师严谨的治学态度、渊博的专业知识、敏锐的学术洞察力将对我以后的工作、学习产生深远的影响。论文的字里行间无不浸透了王老师的心血。王老师在学术上带给我启迪，拓宽了我的思路，引导了我的学术思维。王老师不仅仅是我三年的指导老师，更是我终生的榜样。

在此我要真诚地感谢胡学钢教授！胡老师在学术上对我们要求严格，生活中是我们的良师益友。

我还要感谢我的师兄姚宏亮博士，另外还有我们 KDD 课题组的方宝富师兄、于磊、王骋、杨静等。我与他们在学习上互相帮助，生活上情同手足。

最后，感谢所有关心过我，帮助过我的老师和同学。

作者 姜卯生

2004 年 5 月 20 日

第一章 绪 论

1.1 数据挖掘技术概述

本研究课题的学术背景是数据挖掘（Data Mining，简称为 DM）。下面简单介绍数据挖掘的产生背景，数据挖掘基本概念、种类及其研究现状和发展趋势。

1.1.1 数据挖掘产生的背景

我们已经处于数字时代。半个多世纪以来，计算机技术的高速发展使得信息技术已经渗透到人类活动的各个领域。数据库，数据仓库以及 Intenet 技术的应用普及使得我们可以获得和需要处理的数据规模越来越巨大[1]。这些数据都是非常宝贵的资源。

然而，在拥有海量数据的同时，我们对数据知识的提取很大程度上依旧停留在过去查询、简单检索的水平上。信息的载体是数据，但是数据本身不等于信息。激增的数据后面蕴涵着大量的“宝藏”——事先未知而潜在有用的信息，这就导致了所谓的“数据爆炸但知识贫乏”现象。比如，公司的经理如何从数据中发现顾客的偏好，以便有针对性地开发新产品？医学研究人员采用什么方法才能从大量病历中找出患某种病的病人的共同病症，以便采取措施增加预防和治愈机会？这些问题从传统的数据库中无法找出答案。传统的数据库不适合于处理分析性问题。

总的来说，当信息在人类生活中逐渐扮演越来越重要角色且数据资源充足的时候，人们希望能从繁杂的数据中挖掘出有用的信息，发现其中存在的关系和规则。这正是数据挖掘产生并发展的现实基础。

从技术角度来看，数据挖掘也是很有必要的。仅凭人去理解一个大的数据集合是很困难的或者说是根本不可能的。数据的增加一般沿两个层面：领域的数目和案例数。人类的分析和抽象能力不适宜于高维和海量数据，处理高维数据的一个标准方法是把数据投影到一个维数较低的子空间，然后在这个简化的空间中进行分析和建模。随着维数的不断增加，降维所可能组合的个数呈爆炸性增长；另外，向低维子空间投影后，可能把本来相对容易识别的问题转化为一个难以识别的问题。而某些挖掘算法，能利用反转技术有目的地增加维数，使得模式变得更加简单。此外，数据集合的增长速度也远远超过了传统的手工分析技术所能处理的程度。如果我们想及时地利用由数据提供的信息，用传统的分析技术方法是不可能达到目的的。

1.1.2 数据挖掘的基本概念

一提起数据挖掘人们就会想起基于数据库的知识发现（KDD：Knowledge Discovery in Databases）。KDD 与数据挖掘是两个息息相关的概念。由于现在的工作大部分是基于数据库的，所以在实际研究与应用过程中提起更多的是 KDD。下面介绍两者各自的概念及相互关系，这有利于理解后续内容。

人们从不同的层面提出了不同的 KDD 定义，一种大家普遍接受的定义形式是[2]：KDD 是一种从数据中发现真实、新颖、有潜在应用价值而且最终可以被理解的模式的非平凡过程。它包括从数据库中对数据的选取和采样，清理和预处理，转换和必要的简化，从数据中挖掘产生模式，直到对得到的模式进行解释和评估等过程。这里所说的模式是对一个数据子集的狭义描述，不同于模型。提取的知识表示为概念、规则、规律、模式、约束和可视化等形式。

数据挖掘[3]是从大量的、不完全的、有噪声的、模糊的、随机的实际应用数据中，提取隐含在其中的、人们事先不知道的、但又是潜在有用的信息和知识的过程。

从概念可以看出，数据挖掘的范围比 KDD 广泛，KDD 是面向数据库的，而数据挖掘面向的数据形式可以有多种多样，它可以是数据库，还可以是图像，声音等媒体数据。从过程上看，数据挖掘又可以被看作是从数据库中提取有用信息这一过程的同义词，它是 KDD 的一个步骤[3][4]。

本文的主要内容是数据挖掘中的分类问题。本文中的分类是面向数据库的分类，所以严格地说它属于 KDD 范畴。有些分类是面向文本，页面等其它媒体形式的，它们属于一般意义上的数据挖掘的范畴。

1.1.3 数据挖掘的种类

根据被挖掘知识的种类，数据挖掘可分类以下几种类型[3][5]：

(1) 概化规则(Summarization)挖掘 它主要做的是从用户指定的数据库中挖掘出（从不同的角度或在不同的层次上的）平均/最小/最大值、总和、百分比等等。挖掘结果用交叉表，特征规则，统计的曲线图表等表示。

(2) 关联规则(Association)挖掘 它要做的是从数据库中挖掘出满足一定条件的依赖关系或相关关系。

(3) 分类(Classification)规则挖掘 它的任务是在已知训练数据的特征和分类结果的前提下，为每一个分类找到一个合理的描述或模型。然后再用这些分类的描述或模型对类别未知的新的数据进行分类。分类是数据挖掘中一个十分重要的课题，许多数据挖掘问题本质上都可以等价或转化为分类问题。例如语音识别(Speech Identification)、图像识别(Image Identification)等问题。这些问题实际上是为某一语音或图像数据找到合理的特征描述。如果把特征集合当成类别集合，那上述识别问题就

纯粹是一分类问题了。本文的研究工作即是围绕数据挖掘中的分类问题展开的。

(4) 聚类(Clustering)规则挖掘 它也是一种特殊的分类过程，有时称之为无监督分类，其宗旨在于按被处理对象的内在特征分类，有相同特征的数据被归为一类。它与分类规则挖掘的区别在于分类是基于训练数据的，而聚类则直接对数据进行处理。

(5) 预测(Prediction)分析 当分类的工作偏向于处理漏掉的数据、预测数据的分类或发展趋势时，这时的工作就属于预测分析的范畴。

(6) 趋势(Trend)分析 又称时间序列分析，它是从一时间段的发展过程中发现数据的时序特性，以利于决策分析。

(7) 偏差(Deviation)分析 又称比较分析，它将找出一系列判别式的规则，以区别用户设定的两个不同类。

1.1.4 数据挖掘的研究现状和发展趋势

目前，对数据挖掘的研究主要体现在以下几个方面：对知识发现方法的研究进一步发展，如近年来注重对 Bayes(贝叶斯)方法以及 Boosting 方法的研究和提高[6]；传统的统计学回归法在 DM 中的应用[3]；DM 与数据库的结合越来越紧密。在应用方面：KDD 商业软件工具不断产生和完善，注重建立解决问题的整体系统，而不是孤立的过程。用户主要集中在大型银行、保险公司、电信公司和销售业。国外很多计算机公司非常重视 DM 系统的开发应用，IBM 与微软都成立了相应的研究中心进行这方面的工作。许多著名的计算机公司开始尝试着 KDD 软件的开发，比较典型的有 SAS 公司的 Enterprise Miner，IBM 公司的 Intelligent miner,SGI 公司的 SetMiner ，SPSS 公司的 Clementine 等。Web 数据挖掘产品有 Net percerptions， Accrue Insight 和 Accrue Hit List， CommerceTrends 等。

与国外相比，国内对 DM 的研究稍晚，目前进行的大多数研究项目是由政府资助进行的，如国家自然科学基金、863 计划、“九五”、“十五”计划等。1993 年国家自然科学基金开始对数据挖掘研究进行支持。国内从事数据挖掘研究的人员主要集中在大学，也有部分在研究所或公司。所涉及的研究领域很多，一般集中于学习算法的研究、数据挖掘的实际应用以及有关数据挖掘理论方面的研究[5]。如北京系统工程研究所对模糊方法在数据挖掘中的应用研究、北京大学对数据立方体的研究、华中理工大学、复旦大学、浙江大学等对关联规则的研究等。但是到目前为止，国内还没有比较成熟的数据挖掘产品。

数据挖掘研究的趋势体现在以下几个方面：

(1) 研究专门用于知识发现的数据挖掘语言，也许会像 SQL 语言一样走向形式化和标准化；

(2) 寻求数据挖掘中的可视化方法，使得知识发现的过程能够被用户理解，也便于在知识发现过程中的人机交互；

(3) 研究在网络环境下的数据挖掘技术，特别是在 Internet 上建立 DM Server，与数据库服务器配合，实现数据挖掘；

(4) 加强对各种非结构化数据的挖掘，如文本数据、图形图象数据、多媒体数据；

(5) 与未来的网格技术相结合，研究基于网格的数据挖掘技术，其研究需要将随着网格技术的发展而日臻紧迫。

1.2 数据挖掘中的分类问题

分类在数据挖掘中是一个非常重要的课题，目前在商业上应用最多。分类的任务是找出一个类别的概念描述(通常称之为分类器)，它代表了这类数据的整体信息，即该类的内涵描述，一般用规则或决策树模式表示。该模式能够把数据库中的元组映射到给定类别集中的某一个。例如：可以建立一个疾病诊断分类器，用于根据病症特征集自动判断该病症所对应的疾病，以帮助医生诊断。

一个类的内涵描述分为：特征描述和辨别性描述[7]。

特征描述是对类中对象的共同特征的描述；辨别性描述是对两个或多个类之间的区别性描述。特征描述允许不同类中具有共同特征；而辨别性描述对不同类不能有相同的特征。分类就是寻找合适的辨别性描述的过程。

我们可以将分类模型都抽象成用分类函数的形式来表示。这样一来可以用如下规范化的形式来定义分类：

分类要学习的分类器就相当于一个函数 f(x)，它给需要分类的实例 x 赋予类标签 $c_j \in C(j=1,2,\cdots,m)$，实例 x 由一组属性值 $a_1,a_2,\cdots,a_m$ 描述，C 是类变量集合，取有限值。本文所涉及的分类模型就建立在上述定义之上。

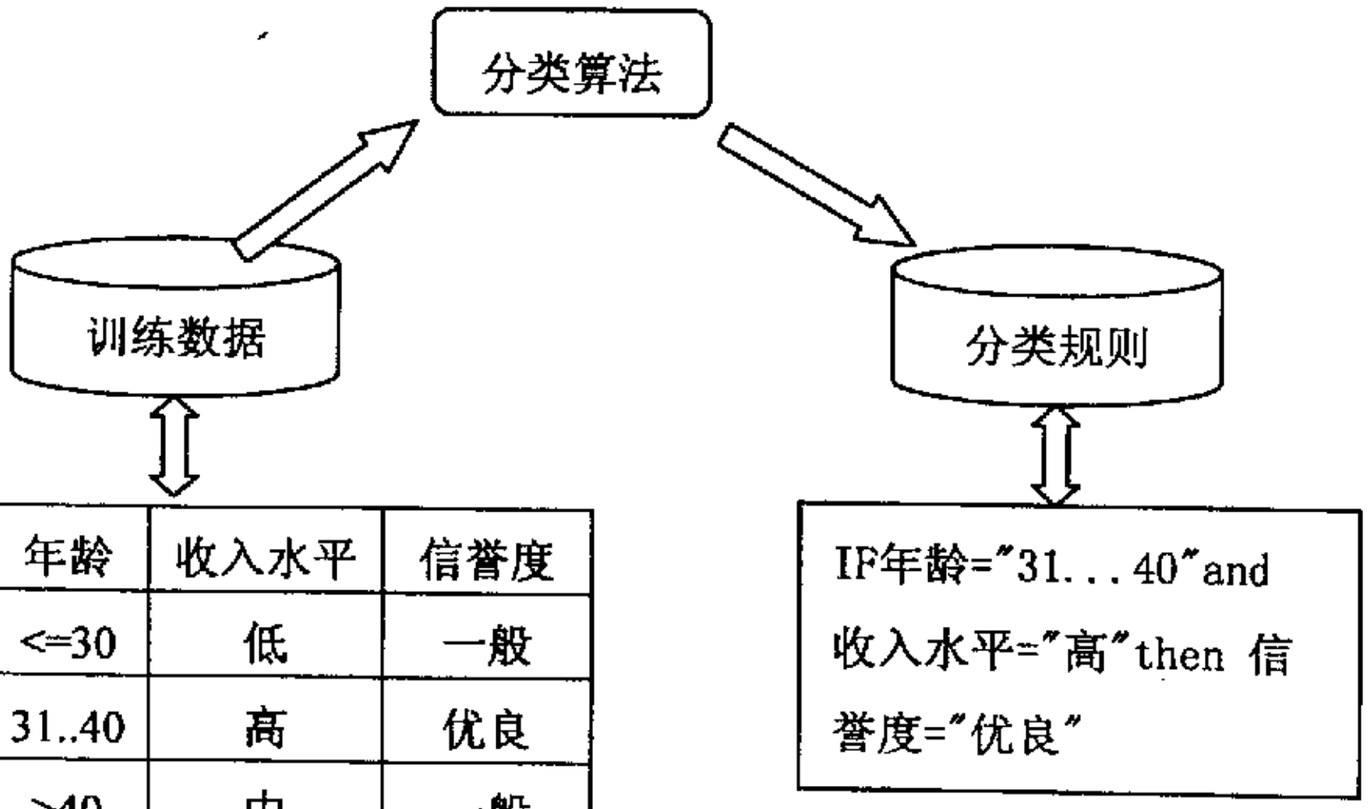
1.2.1 数据分类的过程

分类包括两个过程：分类模型的建立以及运用模型进行新实例分类：

1.2.1.1 分类模型的建立

通过分析由属性描述的数据库元组来构造模型。假定每个元素属于一个预定的类，由一个类标签属性(Class Label Attribute)表示。对于分类，数据元组也称为样本、实例或对象。为建立分类模型而被分析的数据元组构成训练数据集。训练数据集中的单个元组称为训练样本。由于预先知道每个训练样本的类别属性值，这个建立模型的学习过程属于有监督的学习，与无监督学习相对。无监督学习的分类就是所谓的聚类。聚类过程中每个训练样本的类标签事先是未知的，要学习的类属性可能取值及可能取值的个数事先也可能不知道，其目的是将“距离”相近或“个性”相似的元组放在一起，构成一类。

通常，通过第一步的学习建立的模型用分类规则、决策树或数学公式的形式表示。例如：给定一个顾客信用信息的数据库，通过分类算法学习得出分类规则，根据这些规则，可以判断顾客信誉的好坏(如图 1.1)。这样的规则就是一种分类模型。以后就可以利用这个模型为其它顾客的数据进行分类。



顾客 ID	年龄	收入水平	信誉度
No.1	<=30	低	一般
No.2	31..40	高	优良
No.3	>40	中	一般
No.4	>40	中	一般
No.5	31..40	高	优良
……	……	……	……

图 1-1 分类模型的学习，在训练数据上用分类算法学习，学习模型用分类规则的形式表示

1.2.1.2 模型的运用

首先要评估模型的预测准确率。常用的评估手段是保持[8](Holdout)方法。该方法使用类标签测试样本集，这些样本随机选取，并与前面使用过的训练集相独立，即测试样本集完全不同于训练样本集。模型在测试样本集上的准确率是指正确被模型分类的测试样本的百分比。对于每个测试样本，将分类模型学习得出的预测类与已知的类标签相比较，如果相同，则表示分类成功。评估过程中之所以使用与训练集相独立的测试集，是为了避免出现过分拟合的现象。

如果通过测试认为模型是可以接受的，那么就可以利用这个模型对类标签未知的数据实例或对象进行分类。例如：在通过分析现有顾客数据的基础上学习得到的分类规则可以用于预测新的顾客的信誉度(如图 1.2)。

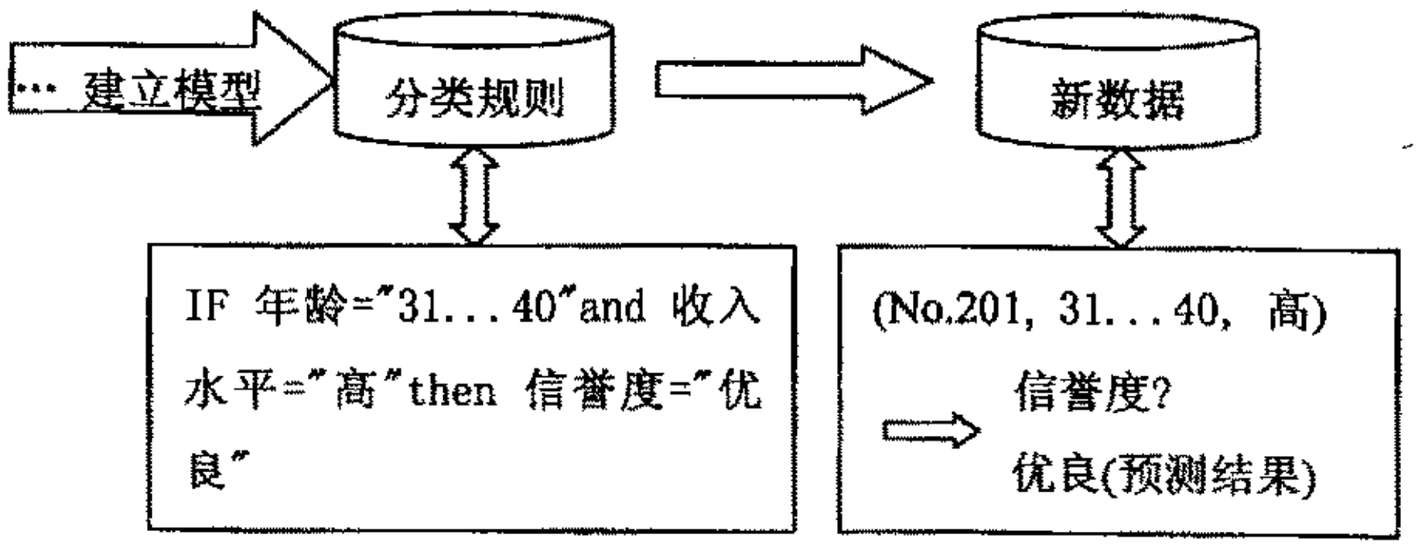


图 1-2 分类模型的运用

分类具有广泛的应用，包括信誉断定、医疗诊断、性能预测和选择购物等。

1.2.2 数据的预处理

为了提高分类的准确性、有效性和可伸缩性，需要对分类所用的数据进行必要的预处理。

1.2.2.1 数据转换

为了便于分类，需要对原始数据进行必要的转换。例如将连续型的数据离散化、将属性值数字化、对数据进行概念抽象等。

1.2.2.2 数据清理

数据清理的目的是清除或减少噪声数据以及处理空缺数据。可采用平滑技术消除或减少噪声数据；对于空缺值，可用该属性最常出现的值，或根据统计，用最可能的值代替。尽管大部分的分类算法都有处理噪声数据和空缺值的机制，但经过清理的数据将更有助于提高学习的执行效率。

1.2.2.3 相关性分析

数据中的许多属性可能与分类任务不相关。例如：记录银行贷款申请日期的数据可能与客户的信誉度不相关。此外，还可能有些属性是冗余的，如果包含这些冗余属性将减慢或误导学习步骤。因此，可以进行相关性分析，删除学习过程中不相关的或冗余的属性。在机器学习中，这一过程称之为特征选择。

在理想情况下，用在相关性分析上的时间，加上从压缩了的属性集上学习的时间，应当少于在原来属性集上学习所用的时间。这种分析就可以帮助提高分类精度及分类效率。

1.2.3 分类方法的比较和评估

分类方法可以根据下列标准进行比较和评估

✧ **预测准确度** 预测准确度是用得最广泛的一种比较尺度，特别是对于预测型分类任务。常见的方法是 N 交叉验证法(CV-N)。

✧ **计算复杂度** 计算复杂度依赖于算法的实现细节与硬件环境。在 DM 中，由于操作对象是大型数据库，并且在实际应用中数据规模越来越大。因此空间和时间的复杂度问题将是一个非常重要的环节。

✧ **健壮性** 这涉及对于数据集中噪声数据或空缺数据的处理，它反应在有噪声数据或空缺数据的情况下模型是否有正确分类的能力。

✧ **可伸缩性** 大部分的分类算法是内存驻留算法，通常假定数据量很小。算法的可伸缩性意味着对于海量数据而言是否具有有效的构造模型的能力。这一点在硬件性能提高且数据规模不断扩大的情况下显得很重要。

✧ **模型的简洁度与可理解性** 对于描述型的分类任务，模型描述越简洁且越容易理解就越受欢迎。例如，采用规则表示的分类器比较简明好用，而用神经网络构造产生的分类器则比较难以理解。

1.2.4 几种主要的分类方法

数据挖掘领域中分类的方法很多，本节介绍几种常见的分类思想。

1.2.4.1 线性判别函数分类方法

我们知道，如果实际问题的决策面是线性的（直线的或者超平面的），计算和构造过程就相当简单。因此即使遇到的决策面不是线性的，我们也宁可牺牲错误率最小这个最优原则，努力构造成线性函数。

线性分类法[2]的目标就是寻找一条直线：$g(x)=W^T*X+w_0$，这条直线能够能够尽可能地将两类样本分开。Fisher 线性判别函数是一个经典的判别方法。它的核心思想是进行坐标变换，寻找能将样本尽可能分开的方向。考虑把 n 维空间的样本投影到一条直线上，形成一维空间。为了避免投影后不同样本混杂在一起，不易区分，可以将直线转动，寻找一个方向使样本的投影尽量分开。也就是说，使得类间差异尽可能大，类内差异尽可能小。

1.2.4.2 决策树分类方法

决策树[5]是较早应用于数据挖掘分类问题的一种方法。在数据量较大时，决策树方法能较快地构造出分类器；其树型结构可以很方便地转化为 SQL 语言

形式，以便用来更有效地访问数据库；且 IF-THEN 规则可以很容易地从这种结构转化中得到，因此这种方法引起了研究者的广泛兴趣。

绝大多数决策树分类方法分两步构造分类器：树的生成与树的剪枝。在树的生成阶段，决策树是通过反复地分拆训练集而成。在每一次分拆时，都是利用某种分拆准则选择一个属性。由所选属性值不同将训练集分成多个子集。然后在每个子集上重复同样的分拆过程，直到每个分拆后的训练集的子集样本均属于同一类别为止。

对树的剪枝操作是为了避免出现模型的过分拟合现象。因为如果完全按训练集中的样本生成决策树，那么当样本数据存在噪声时，就会出现过分拟合的现象，即把噪声数据当作正确的样本而同样要求决策树拟合。这实际会导致决策树泛化能力的下降，甚至可能会使生成的决策树几乎不可用。因此必须对过分拟合的分支进行修剪。通常的修剪方法有两种：一是利用测试集，选择使得对测试集分类的误差最小的子树；另外的一种方法是借助于 MDL(最小描述长度)原理进行剪枝，它是从概率描述的层面来验证决策树的结构。上述两种方法的基本思想和目的是一致的，都是为了弱化噪声数据的消极影响，提高分类模型的表达能力。

这种分类方法的关键是在树的生成阶段找出合适的分拆准则。目前用得最多的是 Quinlan[5]于 1983 年提出来的 ID3 准则和 CART(分类与回归树)准则。

1.2.4.3 粗糙集分类方法

粗糙集理论[9]是 Z.Pawlak 于 1982 年提出来的。这一理论从新的角度对知识进行了定义，把知识看作是对论域的划分，认为知识是有粒度的。引入代数学中的等价关系来讨论知识。该理论近年来主要被用于知识约简、知识的相关性分析及分类挖掘。

粗糙集的基本理论是：在数据库中将行元素看成对象，列元素当成属性(分为条件属性与决策属性)。等价关系 R 定义为不同的对象在某个(或几个)属性上取值相同，这些满足等价关系的对象的集合称之为等价关系 R 的等价类。条件属性上的等价类 E 与决策属性上的等价类 Y 之间的关系分如下三种情形：(1)下近似：Y 包含 E；(2)上近似：Y 与 E 的交集非空；(3)无关：Y 与 E 的交集为空。对下近似建立确定性规则，对上近似建立不确定性规则(含可信度)，对无关情况不存在规则。

1.2.4.4 概念格方法

概念格[10]是基于二元关系构造的，它描述了对象和特征之间的联系，表明了概念之间的泛化和例化关系，其相应的哈斯图实现了对数据的可视化。作为

知识的一种表示形式，它有助于挖掘概念间的各种规则。概念是把所感知的事物的共同本质特点抽象出来，并加以概括。概念都具有内涵和外延，基于对概念的这种理解，R.Wille[11]在 1982 年首先提出根据二元关系来构造相应概念格（或 Galois 格）的思想，也称为形式概念分析。其基本内容是以概念格中的每个节点表示一个形式概念，其中概念的外延代表相应的一组对象，内涵则表示这组对象所具有的公共特征（属性）。概念格所对应的哈斯图形象地揭示了概念间的泛化和例化关系，反映出一种概念层次结构（Concept Hierarchy），实现了对数据的可视化。上述这些特性使得概念格成为数据挖掘领域一种颇受青睐的分类工具。

1.2.4.5 神经网络分类方法

在数据挖掘领域，神经网络方法由于其结构复杂，且学习过程中的非线性优化存在局部极小值等问题而研究得较少。最早正式将神经网络理论引入分类领域的是 H.Lu、R.Setiono 及 H.Liu[12]。神经网络模仿生物神经元对信息的传递特性构建分层网络模型。可以用图形的方式说明其分类思想。图 1.3 是一个三层神经网络模型。数据属性从其中的输入层进入网络，输出层反应了对其分类的预测信息。其中的每个点就是神经网络中的“神经元”。

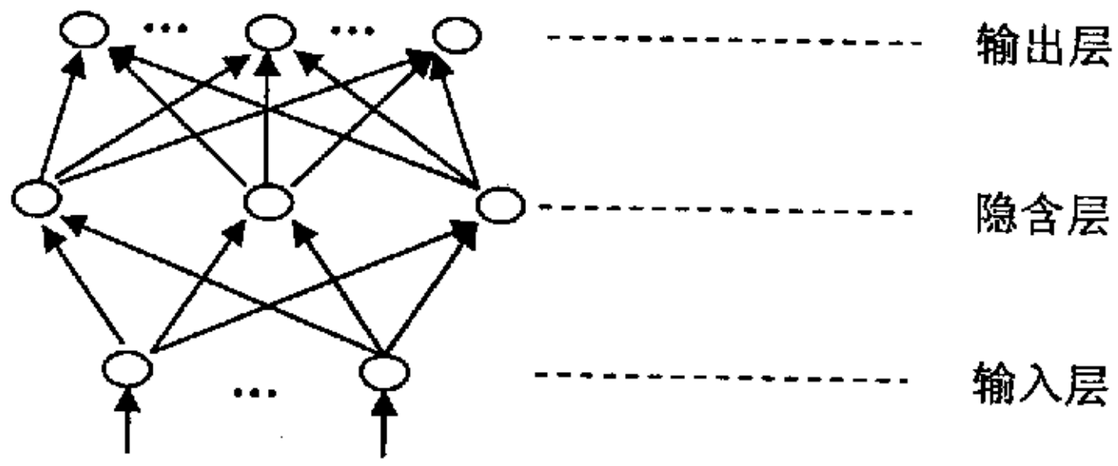


图 1-3 神经网络分类模型图示

1.2.4.6 距离函数法和最近邻判别法

模式分类中最简单直观的方法就是基于距离函数的分类法。它的核心思想是使用一类的重心来代表这个类，计算待分类样本到各类重心的距离，归入距离最近的类。在判别分析中常采用马氏距离，因为马氏距离既考虑了类的均值，又包含了类内方差的信息，对训练样本中蕴涵的信息利用得比较充分。采用马氏距离的基本假设是各类均服从正态分布。

如果允许类中全部样本点都可有资格作为类的代表的话，这就是最近邻法。最近邻法不是仅仅比较与各类均值的距离，而是计算和所有样本点之间的距离，选择距离最近的将新实例归入所属类。

为了克服最近邻法错判率较高的缺陷，k-近邻法不是仅选取一个最近邻进行分类，而是选取 k 个近邻，然后检查它们的类别，归入比重最大的那一类。

上述分类称为“聚类”，也叫做无监督分类。

1.2.4.7 支持向量机分类方法

支持向量机[13]是上世纪九十年代中期提出来的一种分类方法。它脱离传统方法中降维的定式，利用反转技术有目的增加问题空间的维数，使得分类问题变得相对容易。对某些简单的问题来说，统计的方法可以较精确地将那些需要考虑的因素区分出来，以便成功地进行学习。而在实际应用中，不得不使用比较复杂的算法和模型，比如神经网络等。支持向量机具有两者的优点。它能构造相当复杂的模型，其中包含大量的神经网络，RBF 网络和作为特例的多项式分类器。但是它的基本思想又是相当简单的，因为它对应于高维空间中的线形方法。

1.2.4.8 基于贝叶斯技术的分类方法

贝叶斯学派[14][15][16][17][18]形成于上世纪五六十年代，关于贝叶斯技术的研究久盛不衰。八十年代，贝叶斯网络成功地应用于专家系统。九十年代以来，贝叶斯学习一直是机器学习研究的重要方向。基于这种技术的分类方法是本文介绍的重点内容，它以完善的贝叶斯理论为基础，这种分类方法有较强的模型表示、学习和推理能力。本文后面的章节都围绕这一主题展开。

在本节介绍上述各种不同分类思想的目的是为了帮助理解数据分类的内涵。这些是本文所选课题的研究背景。

1.3 课题来源和本文的组织

课题来源：

本文受安徽省自然科学基金：基于贝叶斯网技术的智能 Agents 自组织和学习的研究（03042305）的资助。

本文的组织：

本文系统介绍基于贝叶斯技术的分类模型，在介绍一般理论的基础上作了进一步的探索，提出了自己的见解。本文的具体安排如下：

第一章 绪言：介绍数据挖掘的相关概念，并引入本文的研究主题：分类。介绍分类的基本思想、过程以及几种分类方法。

第二章 贝叶斯理论与贝叶斯分类器：比较系统地介绍了贝叶斯基本理论、贝叶斯分类模型的分类思想以及其它相关理论，如信息度量理论、先验分布的选取等。

第三章 半朴素贝叶斯分类模型：介绍半朴素贝叶斯分类模型的基本思想。在此深入研究了借助于条件互信息的半朴素贝叶斯分类算法。

第四章 增量贝叶斯分类器：介绍了增量贝叶斯分类器的基本理论，并就其中基于朴素贝叶斯分类思想的增量分类过程进行了深入探讨。

第五章 总结与展望：展望这一领域以后的发展趋势。介绍将来要完成的工作。

第二章 贝叶斯理论与贝叶斯分类器

2.1 贝叶斯分类器的一般原理

贝叶斯分类器建立在经典的贝叶斯概率理论[14]与贝叶斯网络技术的基础上，下面分别予以介绍：

2.1.1 贝叶斯定理

定义 2.1：一个随机试验所有可能的“基本结果”ω构成的集合称为该随机试验的基本空间，常用集合Ω={ω}表示，基本空间又称为**样本空间**，其元素ω称为**样本点**。

例如：对于任意掷一枚硬币的随机实验，其样本空间Ω={正，反}。

定义 2.2：给定了基本空间Ω，一个**随机事件**就是Ω的一个子集，也就是由某些基本结果组成的集合。随机事件表示随机试验的某种结果。随机事件可以简称为**事件**。

例如：投掷两颗骰子，“其和为 4 点”这一事件可用集合的形式表示为A={(1,3),(2,2),(3,1)}。

定义 2.3:给定基本空间Ω中的两个事件 A 与 B，很显然$A\in\Omega$，$B\in\Omega$，如果$A\cap B=\phi$，则称 A 与 B 互为**不相容事件**。

定义 2.4：若给定一个事件A，则“A不发生”这个事件称为A的**对立事件**。用子集A在Ω中的补集 $\overline{A}=\Omega-A$表示。

从定义中容易看出不相容事件与对立事件两个概念之间的区别，这两个概念在实际问题中容易混淆。

定义 2.5：在概率论中为了保证符合规定性质的的事件概率存在，并不总是逐一讨论基本空间Ω中的一切子集。实际上当Ω为不可数无穷集时，事件数也有无穷个。因此，为了研究事件间的各种关系，记**全体事件构成的集类**为$\Re$，要求$\Re$满足：

(1) $\Omega\in\Re$；

(2) 若$A\in\Re$，则$\overline{A}\in\Re$；

(3) 若$A_1,A_2,\ldots,A_n,\ldots\in\Re$，则$\bigcup_{i=1}^{\infty}A_i\in\Re$。

下面给出概率的定义：

定义 2.6：如果P是$\Re$上的一个实值函数，即对每一个$A\in\Re$，有一个实函

数P(A)与之对应，并且满足以下三点：

非负性　对 $\forall A \in \Re$，$P(A) \geq 0$；

规范性　$P(\Re) = 1$；

可列可加性　若 $A_1, A_2, ..., A_n, ...$ 是 $\Re$ 中两两不相容的事件，则

$$P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i) \tag{2.1}$$

则称P是$(\Omega, \Re)$上的一个概率（测度），$P(A)$称为事件A的概率，三元组$(\Omega, \Re, P)$称为概率空间。

条件概率反应了事件之间的概率关系：

定义 2.7：设$(\Omega, \Re, P)$为一概率空间，$A, B \in \Re$，且$P(A)>0$，则

$$P(B \mid A) = \frac{P(AB)}{P(A)}, \tag{2.2}$$

称为已知A发生时B的**条件概率**。

下述三个公式是贝叶斯技术的直接理论依据：

乘法公式：

$$P(AB) = P(A)P(B \mid A) \qquad (P(A)>0);$$

$$P(AB) = P(B)P(A \mid B) \qquad (P(B)>0);$$

更一般的情形是：设 $A_1, A_2, ..., A_n \in \Re$，$n \geq 2$，$P(A_1, A_2, ..., A_n) > 0$则

$$P(A_1, A_2, ..., A_n) = P(A_1)P(A_2 \mid A_1)P(A_3 \mid A_1, A_2)...P(A_n \mid A_1, A_2, ..., A_n) \tag{2.3}$$

全概率公式：

设 $A_1, A_2, ..., A_n \in \Re$，两两不相容，$P(A_i) > 0, i = 1,2,..,n$，且 $\bigcup_{i=1}^{n} A_i = \Omega$，则对任何事件 $B \in \Re$，有：

$$P(B) = \sum_{i=1}^{n} P(B \mid A_i)P(A_i) \tag{2.4}$$

贝叶斯公式：

若 $A_1, A_2, ..., A_n, ... \in \Re$，两两不相容，$P(A_i) > 0, i = 1,2,..,n$；则对于任何满足 $P(B)>0$ 的 B，$B \in \Re$，有：

$$P(A_j \mid B) = \frac{P(B \mid A_j)P(A_j)}{\sum_{i=1}^{n} P(B \mid A_i)P(A_i)} \tag{2.5}$$

贝叶斯分类技术正是基于上述理论的。在具体讨论分类技术之前，先给出下述带有一般性意义的结论：

2.1.2 最大后验假设与最大似然假设

在观察到数据之前，根据背景知识或经验确定某个假设空间H中的假设h

成立的概率为 P(h)，称之为假设 h 的**先验概率**。令 D 是一个训练数据集合，在没有关于哪个假设成立的知识而观察到的 D 的概率，称为 D 的**先验概率**，用 P(D)表示。在假设 h 成立的条件下，观察到 D 的概率记为 P(D|h)。在观察到训练数据据 D 的条件下，假设 h 成立的概率 P(h|D)称为 h 的**后验概率**。后验概率反映了训练数据对假设成立概率的影响，它是依赖于数据 D 的。已知 P(h)、P(D|h)和 P（D），贝叶斯定理提供了一个计算假设 h 的后验概率的方法，因而成为贝叶斯理论的基石[6]：

$$P(h|D)=\frac{P(D|h)P(h)}{P(D)}$$

通常，学习的任务是：对于给定的观察数据 D，在 H 中发现最可能的假设 $h\in H$。任何这样的具有最大可能的假设称为最大后验假设(MAP,maximum aposteriori)，记为 h_{MAP}：

$$h_{MAP}=\arg\max_{h\in H}P(h|D)=\arg\max_{h\in H}P(D|h)P(h)/P(D)$$

$$=\arg\max_{h\in H}P(D|h)P(h) \qquad (2.6)$$

如果 h 表示对数据分类的假设，上述(2.6)式就是一个原始的分类模型。贝叶斯分类就是根据上述 MAP 假设找出新实例最可能的分类。所有对贝叶斯分类器的所有研究工作都是以此假设为前提。

在没有任何背景知识的情况下，可以假定 H 中所有的假设有相同的先验。这时(2.6)式中的 $P(D|h)$成为给定 h 时数据 D 的似然。任何使 $P(D|h)$最大的假设称为最大似然(ML –maximum likelihood)假设 h_{ML}：

$$h_{ML}=\arg\max_{h\in H}P(D|h) \qquad (2.7)$$

在分类过程中，(2.7)式往往被用来在启发式搜索时进行模型检测。

2.1.3 贝叶斯网络与贝叶斯分类器

贝叶斯网络[19][20][21][22][23][24]是用来表示变量间连接概率的图形模式，它提供了一种自然的表示因果信息的方法，用来发现数据间的潜在联系。在这个网络中，用节点表示变量，有向边表示变量间的依赖关系。当我们在贝叶斯网络中把其中代表类别变量的节点作为根节点，其余所有变量都作为它的子节点时，贝叶斯网络就变成了分类器。

设某领域中类别变量为 $C=\{c_1,c_2,\cdots,c_l\}$，特征变量为 $X=\langle X_1,X_2,\cdots,X_m\rangle$，每个特征的值域为 $Val(X_i)$，$i\in[1\cdots m]$，特征的取值用小写字母 $x_i(i\in[1\cdots m])$表示。对每一个实例 $x=\langle x_1,x_2,\cdots,x_m\rangle$来说，分类的目的就是通过学习一定的训练样本集 D，来获得它的类别标签 c。根据最大后验假设原理，贝叶斯分类器采用下面的表达式 $\max_{i\in[1\cdots l]}\{p(c_i|x)\}$来决定它的类别，其中：

$$p(c_i \mid x)=\frac{p(c_i)\prod_{j=1}^{m} p(x_j \mid c_i;\pi(x_j))}{p(x)} \tag{2.8}$$

这里$\pi(x_j)$表示节点X_j除类别节点 C 之外的所有父节点。x_j表示实例 x 第 j 个特征的取值。所以学习贝叶斯分类模型的任务是从训练样本集 D 中学习概率分布函数：$p(c_i)$，$p(x_j \mid c_i;\pi(x_j))$，$i\in[1\cdots l]$，$j\in[1\cdots m]$。

学习并运用贝叶斯分类器时包括两个过程，一是对于每一特征节点找到除根节点之外的所有父节点，也就是学习贝叶斯网络结构；二是在已知结构的基础上获得上述参数的估计，即所谓的参数学习问题。从数据中学习网络的结构和分布参数正在成为贝叶斯学习理论的研究热点之一。与纯贝叶斯网络学习过程不同的是，贝叶斯分类器所采用的往往是满足一定限制条件的简化的结构形式。这是为了取得算法实践上的可行性。大部分研究工作都集中在如何使分类器在限制框架内取得最优或次优的分类的分类效果。

2.1.4 信息度量理论

美国数学家 Shannon[25]于 1948 年提出了熵的概念。熵是一种信息度量工具，它反映了不确定性问题的平均不确定程度。其在信息论、人工智能和数据挖掘领域中有着广泛的应用。

设随机变量在一个离散事件集合中进行取值，称为离散信源。离散事件集合$A=\{a_1,\cdots,a_n\}$称为信源符号表。如果事件之间相互独立，这样的离散信源称为离散无记忆信源，熵概率空间可以表示成如下形式：

$$X=\begin{Bmatrix} a_1 & a_2 & \cdots & a_n \\ p_1 & p_2 & \cdots & p_n \end{Bmatrix} \qquad p_i\geq 0 \qquad \sum_{i=1}^{n} p_i=1 \qquad i=1,2,\cdots,n$$

其中 (1) p_i为事件a_i发生的概率；

(2) A 又称有限事件集，$P=(p_1,\cdots,p_n)$又称为概率空间的概率矢量。

假设信源输出 N 个消息，其中有 n 个不同的消息（也就是上面概率空间的 n 个事件），第 i 个消息重复h_i次，这时，信源输出消息的平均信息量为：

$$I=\frac{h_1I_1+h_2I_2+\cdots h_nI_n}{N}$$

其中$I_i=\log_2(1/p_i)=-\log_2 p_i$是第 i 个消息的自信息，比值$h_i/N$为输出消息的重复频率，即$h_i/N=p_i$。这样一来，信源输出消息的平均信息量为：

$$I=p_1(-\log_2 p_1)+p_2(-\log_2 p_2)+\cdots+p_n(-\log_2 p_n)=-\sum_{i=1}^{n} p_i\log_2 p_i \tag{2.9}$$

定义 2.8：当信源(变量)X 的概率矢量是$P=(p_1,\cdots,p_n)$时，函数

$$H(X)=-\sum_{i=1}^{n} p_i\log_2 p_i=H(P) \tag{2.10}$$

这称为 X 的**熵函数**。

熵的性质：

对称性：$H(p_1,p_2,\cdots,p_n)=H(p_2,p_1,\cdots,p_n)=\ldots=H(p_n,p_{n-1},\cdots,p_1)$

即熵函数中所有变量 p_i 的位置可以互换，不会影响熵函数的值，其物理意义说明熵仅与随机变量的总体结构有关。

非负性：$H(p_1,p_2,\cdots,p_n)\geq 0$

其中，等号成立的充要条件是当且仅当对某个 i，$p_i=1$，其余 $p_k=0(k\neq i)$。这表明确定性事件的熵最小为零，它是无信息可言的。

极值性：$H(p_1,p_2,\cdots,p_n)\leq H(1/n,1/n,\cdots,1/n)=\log n$

上式表明所有概率 p_i 构成的熵，以等概率分布时达到最大，这又称为最大熵定理。

加法性：如果各随机变量彼此独立，则它们的联合熵 H(XY)=H(X)+H(Y)。

扩展性：$H(p_1,p_2,\cdots,p_n)=\lim_{\varepsilon\to 0}H(p_1,p_2,\cdots,p_n,p_{n+1})$，令 $p_{n+1}=\varepsilon$

这说明了熵的扩展性，设随机变量 X 原有 n 种取值，如果增加一种取值，只要第 n+1 种取值的概率趋近于零，而其它概率不变，则其集合熵不变。实际上，概率很小值的出现，虽然能给接受者以较大的信息，但在熵的的计算中它只占了很小的比重（$\lim_{\varepsilon\to 0}\varepsilon\log\varepsilon\to 0$），它对于集合熵值的贡献也就趋近于零了，这体现了熵的总体平均性。

定义 2.9：
$$H(X|Y)=-\sum_{x,y}P(x,y)\log_2 P(x|y) \tag{2.11}$$

称为变量 X 相对于变量 Y 的**条件熵**。

定义 2.10：
$$I_m(X;Y)=H(X)+H(Y)-H(X,Y)$$
$$=\sum_{X,Y}P(X,Y)\log_2\frac{P(X,Y)}{P(X)P(Y)} \tag{2.12}$$

称为变量 X 与变量 Y 之间的**互信息**，它反映了变量之间的相关程度，互信息越大，变量之间的关联程度越大。

定义 2.11：
$$I_p(X;Y|C)=\sum_{X,Y,C}P(X,Y,C)\log_2\frac{P(X,Y|C)}{P(X|C)P(Y|C)} \tag{2.13}$$

称为变量 X 与 Y 相对于变量 C 的**条件互信息**，其值越大，X 与 Y 之间相对于条件变量 C 的联系越紧密。

2.1.5 先验分布的选取

在贝叶斯分类器的学习过程中，在观察到数据之前，各个特征属性的先验分布是未知的。而先验分布在算法过程中却是不可或缺的。所以必须事先确定好先验分布的形式与参数。这些往往是根据经验知识来确定的。

总的来说，选取先验分布[3][5]有三个原则：**最大熵原则**、杰弗莱原则和共

轭分布原则。在这里简单说明共轭分布的原理：

Raiffa 和 Schaifeer 提出先验分布应选取共轭分布，即要求后验分布与先验分布属于同一分布类型。这么选取使得算法过程中分布参数的计算相当简单。共轭分布的定义形式是：设样本 X_1，X_2，…，X_n 对参数 θ 的条件分布为 $p(x_1,x_2,\cdots,x_n|\theta)$，如果先验分布密度函数 $\pi(\theta)$ 决定的后验密度 $\pi(\theta|x)$ 与 $\pi(\theta)$ 同属于一种分布类型，则称 $\pi(\theta)$ 为 $\pi(x|\theta)$ 的共轭分布。

用共轭分布作为先验可以将历史上做过的各次试验进行合理整合，也可以为今后的试验结果分析提供一个合理的前提。由于非共轭分布的计算实际上是相当困难的，相比之下，共轭分布计算后验只需要利用先验做乘法，其计算特别简单。可以说共轭分布族为贝叶斯学习的实际使用铺平了道路。

二项分布、多项分布、正态分布、Gamma 分布、Poisson 分布、多变量正态分布以及 Dirichlet 分布等都是共轭分布。

本文第四章的增量贝叶斯分类器中采用的就是 Dirichlet 共轭分布，下面介绍这一分布的定义及计算特点：

定义 2.8：设事件变量 Y 有 Y^1，Y^2，…，Y^r 共 r 个可能的状态，参数向量为 $\theta=(\theta_1,\theta_2,\cdots,\theta_r)$，其中 $\theta_k = p(Y=Y^k|\theta,I_0)$(k=1, 2, …, r)，若它的分布密度满足：

$$p(\theta|I_0)=Dir(\alpha_1,\alpha_2,\cdots\alpha_r)=\frac{\Gamma(\alpha)}{\prod_{k=1}^{r}\Gamma(\alpha_k)}\prod_{k=1}^{r}\theta_k^{\alpha_k-1} \qquad (\alpha=\alpha_1+\alpha_2+\cdots+\alpha_r)$$

则称 θ 关于先验信息 I_0 具有 Dirichlet 分布。这里 $(\alpha_1,\alpha_2,\cdots,\alpha_r)$ 为超参数，当分布为均匀分布时，$\alpha_i=1\ (i=1,2,\cdots,r)$。

如果在样本 S 中 $Y=Y^k$ 出现的次数为 n_k $(k=1,2,\cdots,r)$，根据共轭分布的定义，它的后验分布也服从 Dirichlet 分布：$p(\theta|S,I_0)=Dir(\alpha_1+n_1,\alpha_2+n_2,\cdots,\alpha_r+n_r)$。可以用 θ 的后验条件期望作为其后验估计值。具体的计算公式是：

$$\hat{\theta}_k=\int\theta_k Dir(\alpha_1+n_1,\alpha_2+n_2,\cdots,\alpha_r+n_r)d\theta_k=\frac{\alpha_k+n_k}{\alpha+n} \tag{2.14}$$

很显然，最后的计算过程相当简单。另外，从计算公式可以看出，随着学习的推进，数据量的增多，公式中超参数 α 的作用越来越小，实际数据的分布在公式中的表现力得到逐步增强，这是 Dirichlet 先验分布的增量特性。第四章的增量贝叶斯分类算法中参数的计算运用了这一特性。

2.2 几种贝叶斯分类模型

本节简单综述各种贝叶斯分类模型，在介绍每种分类模型特点的基础上引出第三章，第四章中本文所做的工作。

2.2.1 朴素贝叶斯分类模型（NBC）

2.2.1.1 朴素贝叶斯分类原理

朴素贝叶斯分类器[26][27](NBC-Naïve Bayesian Classifier)是贝叶斯分类器中一种最简单的、有效的而且在实际使用中很成功的分类器。其性能可以与神经网络、决策树分类器(例如 C4.5)相比，在某些场合优于其它分类器。

NBC 使用公式(2.8)来判定新实例的类别。运用公式之前，需要学习贝叶斯网络的结构，以确定每个特征节点的除根节点之外的所有父节点。朴素贝叶斯分类模型采用的是最简单的贝叶斯网络结构。在该模型中，假设所有的属性 $A_i(i\in[1\cdots m])$都条件独立于类变量 C，即每一个属性变量都以类变量作为唯一的父节点。由于结构简单，有时将其与严格的贝叶斯网络分类器相区别，仅称之为朴素贝叶斯分类器。图 2-1 直观地描述了朴素贝叶斯分类模型的结构特点：

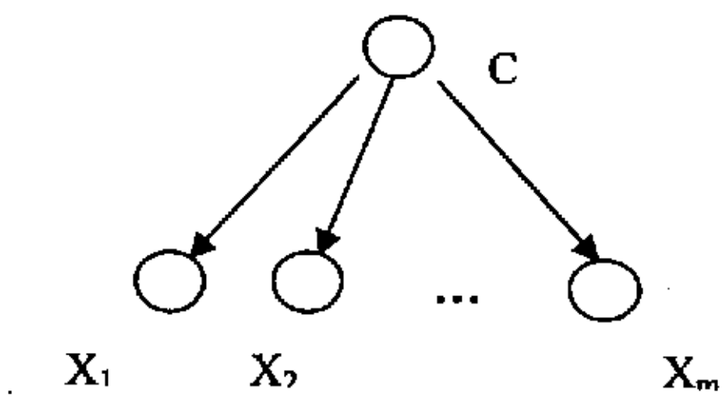


图 2-1　朴素贝叶斯分类器结构示意图

由独立性假设可知，公式(2.8)中每个属性除类别属性之外没有其它的父节点，及对任一 $x_j(j\in[1\cdots m])$而言，$\pi(x_j)=\phi$。所以公式(2.8)有如下简化形式：

$$p(c_i\mid x)=\frac{p(c_i)\prod_{j=1}^{m}p(x_j\mid c_i;\pi(x_j))}{p(x)}=\frac{p(c_i)\prod_{j=1}^{m}p(x_j\mid c_i)}{p(x)}$$

$$=\alpha\cdot p(c_i)\cdot\prod_{j=1}^{m}p(x_j\mid c_i)\qquad(2.15)$$

其中 α 是正规化常数。在分类过程中以后验概率作为分类指示，即输出具有最大后验概率的类标签 c_{NB}：

$$c_{NB}=\arg\max_{c_i\in C}p(c_i)\prod_{j=1}^{m}p(x_j\mid c_i)\qquad(2.16)$$

c_{NB} 表示贝叶斯分类器输出的目标值，常数 α 可以省略。通常公式(2.16)也作为朴素贝叶斯分类器的定义式。实际计算时，公式中的 $p(c_i)$可以通过计算训练样例中 c_i 出现的频率来估计，　$p(x_j\mid c_i)$也可以通过相应的频率来估计，具体的计算公式是：

$$p(c_i)=\frac{count(C=c_i)}{|D|},\qquad p(x_j\mid c_i)=\frac{count(X_j=x_j\wedge C=c_i)}{count(C=c_i)}$$

其中 $count(\varpi)$表示训练例中满足条件 ϖ 的实例出现的次数。

朴素贝叶斯分类模型优点是：

(1) 算法逻辑简单，易于实现；

(2) 分类过程中时间空间开销小；

(3) 算法稳定，对于不同的数据特点其分类性能差别不大，不会特别“偏好”某种特定的数据集，也就是说健壮性比较好。其健壮性使得它经常被作为其它分类模型的参照模型。

朴素贝叶斯分类模型中的独立性假设同时也是它的先天不足所在：独立性假设在许多实际问题中并不成立，如果在这些问题中忽视这一点，理应会引起分类的误差。为了克服这一不足，人们主要从两个方面出发做了大量的研究工作：提升 NBC 的分类性能或者改变网络结构假设以放宽独立性条件的限制。

2.2.1.2 朴素贝叶斯分类器的提升（Boosted NBC）

对朴素贝叶斯分类模型进行“提升”(Boosting)[6][28]是在不改变独立性假设的前提下提高分类性能的一种方法。这以思想由 Freund 和 Schapire 于 1995 年提出，其主要思想是从训练实例中学习一系列的分类器。每一个分类器根据前一个分类器错误分类的实例，对训练例的权重进行修正，再学习新的分类器。例如，学习得到分类器 H_k 后，将其错误分类的训练样例的权重提高，然后再从改变权重的训练样例中学习下一个分类器 H_{k+1}。此过程重复 T 次，最后得到的分类器输出各个 H_k 的输出的加权平均，权重对应于 H_k 在训练集的分类准确性。下面是算法实现[6]：

Input：　N 个训练实例：D＝$\langle(x^1,c^1),\cdots,(x^N,c^N)\rangle$ 以及待分类实例 x，

N 个训练实例上的分布 D：　w，　$w=\langle w_1,w_2,\cdots,w_N\rangle$ 为训练实例的权向量，

T：训练重复的趟数。

Output：　$h(x)=\arg\max_{c\in C}\sum_{t=1}^{T}\left(\log\frac{1}{\beta^{(t)}}\right)I\left(h^{(t)}(x)=c\right)$，其中 $I(\varpi)$ 是示意函数，当 ϖ ＝T 时 $I(\varpi)$ ＝1，否则 $I(\varpi)=0$。

Initialize：　$w_i=1/N$，　$i\in[1\cdots N]$

过程：

for t=1 to T

对于给定的权值 $w_i^{(t)}$ 得到一个假设 $H^{(t)}$:X→C;

估计假设 $H^{(t)}$ 的总体误差，　$e^{(t)}=\sum_{i=1}^{N}w_i^{(t)}I\left(c^i\neq h^{(t)}(x^i)\right)$;

计算 $\beta^{(t)}=e^{(t)}/\left(1-e^{(t)}\right)$;

计算下一轮样本的权值 $w_i^{(t+1)}=w_i^{(t)}\left(\beta^{(t)}\right)^{1-I\left(c^i=h^{(t)}(x^i)\right)}$;

正规化 $w_i^{(t+1)}$，使其总和为 1;

End for.

假设每一个分类器都是实际有用的，$e^{(t)}<0.5$，也就是说，在每一次分类的结果中，正确分类的样本个数始终大于错误分类的样本个数。可以看出，此时$\beta^{(t)}<1$，那么当对某个训练实例x^i分类结果不正确时，示函数$I\left(c^i-h^{(t)}\left(x^i\right)\right)=0$，导致$w_i^{(t+1)}$增加，因此满足了提升的思想。

上述提升朴素贝叶斯分类器的时间复杂度是O(TNf)，其中f是每个样本的属性的个数。在一般情况下，提升后的分类性能有了较大地提高。这种提升方法存在的不足是：

不能捕捉属性间的相关性，也就是说不能突破独立性假设条件的限制。

从提升的思想看，当训练集中存在噪音时，提升的方法会把这些噪音当作有用的信息通过权值而放大，这会降低提升的性能。当噪音很多时，这种提升会导致更糟的结果。

上述基于Boosting的朴素贝叶斯分类模型是一种比较流行的改善朴素贝叶斯分类性能的方法。另外还有其它的如“选择贝叶斯分类器”等改善分类性能的方法。

2.2.1.3 半朴素贝叶斯分类模型（SNBC）

为了突破朴素贝叶斯分类器的独立性假设条件的限制，除了上述“提升”等方法之外，还可以通过改变其结构假设的方式来达到目的，为此有人提出了半朴素贝叶斯分类(SNBC-Semi-Naïve Bayesian Classifier)[29][30][31]的构想。从名称可以看出，SNBC 依然属于朴素贝叶斯分类的范畴。这是因为除了结构上的差别之外，计算推导过程与NBC无异。SNBC的结构比NBC紧凑，在SNBC的模型构建过程中，依照一定的标准将关联程度较大的基本属性(即NBC中的特征属性)合并在一起构成“组合属性”(也称之为“大属性”)。逻辑上，SNBC中的组合属性与NBC中的基本属性没有根本性差别，SNBC的各个组合属性之间也是相对于类别属性相互独立的。图2-2是SNBC的模型示意图：

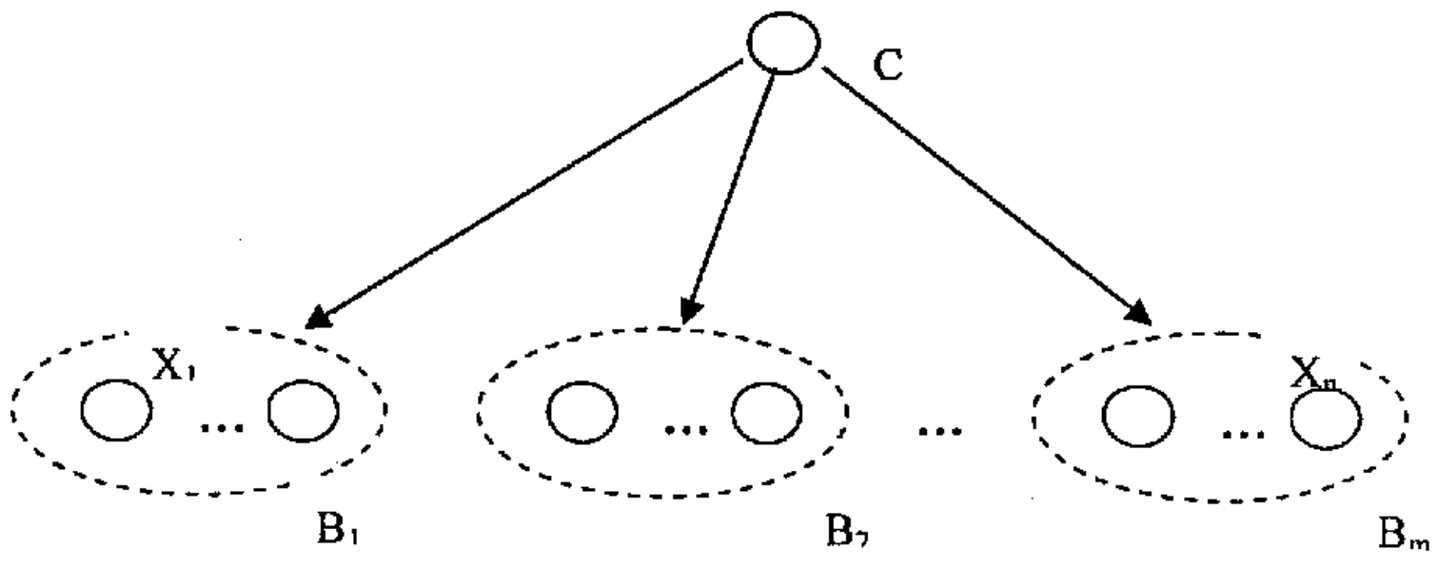


图2-2 半朴素贝叶斯分类器结构示意图

这类模型通过将依赖性强的基本属性结合在一起构建新的模型，这样可以部分屏蔽NBC中独立性假设对分类的负面作用。针对这些模型人们提出了一些

算法构想，本文也对此做了较深入的研究，提出了自己的算法。第三章将予以详细介绍。

2.2.2 贝叶斯网络分类模型（BNC）

一般的贝叶斯网络表示了变量集 $X=\langle X_1,X_2,\cdots,X_m\rangle$ 的联合概率分布：

$$P(X_1,X_2,\cdots,X_m)=\prod_{i=1}^{m}P\left(X_i|\prod X_i\right) \tag{2.17}$$

其中 $\prod_{x_i}$ 表示父节点。也就是说，属性变量可以有一个以上的父节点，因而属性变量之间的依赖关系可以得到表示。这里，学习贝叶斯网络的问题描述为：给定 X 中的一组实例构成的训练集合 $D=\{x^1,x^2,\cdots,x^N\}$，找到一个与 D 匹配最好的网络 B。这样，学习贝叶斯网络的问题转化为优化问题。这时类变量和属性变量不加区别。

实际处理这个问题的方法是在可能的网络构成的空间中进行启发式搜索。搜索成功的关键是确定一个合理的评分函数，评价网络对训练数据的匹配程度，以指导搜索。有两种主要的评分函数：贝叶斯评分函数和最小描述长度原理(MDL-minimal description length)评分函数。它们是渐进正确的，即随着样本数目的增加，得分最高的网络将任意逼近样本的概率分布。

贝叶斯网络分类器[32][33]的问题在于：

分类性能依赖于贝叶斯网络技术[34][35][36][37][40][41][42][43]的发展，网络结构学习的时间开销相当大；

Nir Friedmen 等人深入研究了贝叶斯网络分类器的性能。实验表明，对于某些数据集合，贝叶斯网络分类器的性能明显优于朴素贝叶斯分类器，而对另一些数据集合，则不如朴素贝叶斯分类器，主要表现在分类的准确性下降。以用 MDL[38][39]原则指导学习得到的贝叶斯网络分类器为例，在分类过程中有可能出现这样的情况：学习过程得到 MDL 评分很好的网络，作为分类器的性能却很差。究其原因，是当变量属性增多时，类变量对于属性变量的条件概率的偏差在 MDL 评分函数中反映迟钝，而这对分类却有决定意义。这使得贝叶斯网络分类器在属性很多时性能变差。

基于以上原因，人们提出了一些结构简单的贝叶斯网络分类模型，Nir Friedmen 等人[32]提出的扩展的朴素贝叶斯网络分类器是其中的一个典型。

2.2.2.1 树扩展朴素贝叶斯网络分类器（TAN:Tree-Augmented Naïve Bayesian Classifier）

在本文中，我们将 TAN 与一般的朴素贝叶斯分类器相区别，将其纳入贝叶斯网络分类器的范畴。其基本思想是在朴素贝叶斯分类器的基础上，在属性之

间增添连接弧，以消除朴素贝叶斯分类器关于条件独立的假设。这样的弧称为扩展弧。从节点 X_i 到 X_j 的扩展弧表示属性 X_j 对分类的影响也取决于 X_i 的取值。可能有这样的情况：待分类实例的属性值 x_i 和 x_j 对分类的影响都不大，$P(x_i|c)$ 和 $P(x_j|c)$ 的值低，但是 $P(x_j|x_i,c)$ 的值却高，这时朴素贝叶斯分类器会过度降低实例属于类 c 的概率，而扩展的贝叶斯网络分类器却可以避免这一点。图 2-3 是 TAN 的模型示意图：

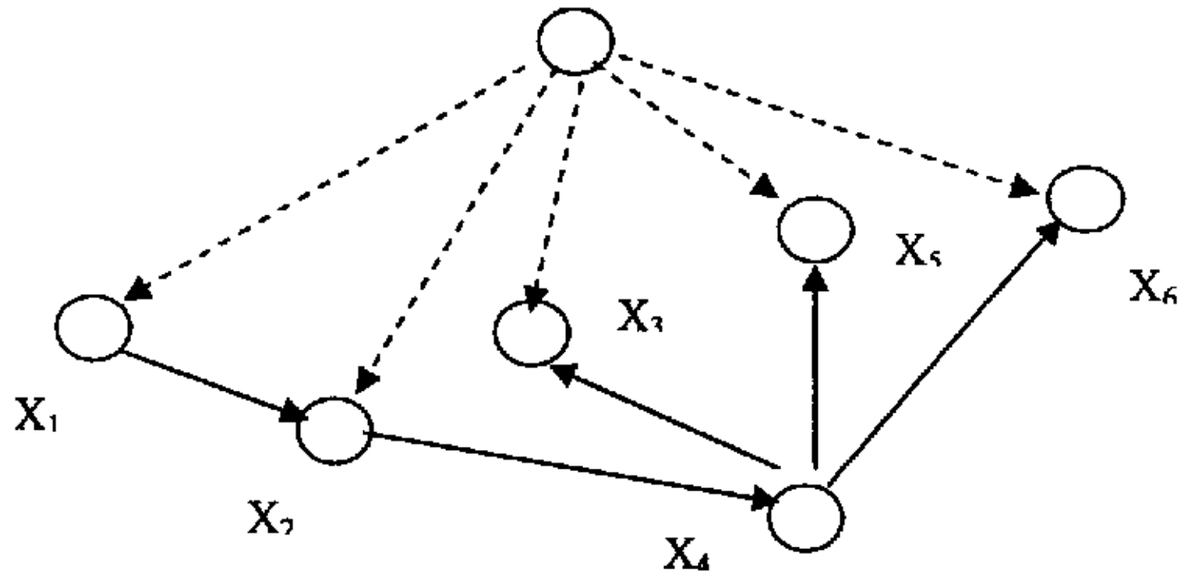


图 2-3 TAN 分类模型示意图

TAN 分类模型的网络构建过程如下[6]：

(1) 通过训练集并运用公式(2.13)计算属性对之间的条件互信息 $I_{\hat{P}_D}(X_i;X_j|C)$，$i \neq j$

(2) 建立一个以 X_1，…，X_m 为顶点的完全无向图，顶点 X_i 到 X_j 标以权重 $I_{\hat{P}_D}(X_i;X_j|C)$

(3) 建立一个最大权重跨度树

(4) 选择一个根结点(相当于图 2-3 中的结点 X_1)，将所有边的方向设置成由根结点向外辐射，把无向树转换为有向树

(5) 增加一个类变量结点 C，并加上从 C 到每一个属性结点 X_i 的弧(相对于图 2-3 中的有向虚线)。

在许多领域，特别是在数据库规模很大时 TAN 分类性能明显优于 NB，并且 TAN 与 NB 一样，对于存在噪声的数据具有良好的健壮性。其不足之处在于：

首先，除了计算复杂度较大之外，当数据规模不够大时，TAN 的性能不如 NB。因为为了学习分类所需要的参数，理论上数据规模越大越好。

其次，NB 可以处理遗漏数据，TAN 却不能。

从提高分类的时间效率与分类精度出发，在 TAN 的基础上有一些变化模型，下面简单介绍其中的简化模型与扩展模型的分类思想：

2.2.2.2 TAN 的简化模型

与 TAN 不同的是，TAN 的简化模型的基本思想是：从属性变量中选择一

个属性作为其它所有属性的父节点。即除了被选择的父节点属性外，所有其它的属性都有两个完全相同的父节点。研究表明，这样的简化模型是介于朴素贝叶斯分类器与 TAN 之间的分类模型。其长处是一般情况下分类精度比 NBC 高，分类的时间性能比 TAN 强。

2.2.2.3 TAN 的扩展模型

这里的扩展模型又称为“受限依赖贝叶斯分类器”，由 Mehran Sahami(Stanford University)提出。这种模型规定：每个特征属性最多可以有包含类别属性在内的 K 个父节点属性。这对算法提出了更高的要求，因此这种扩展模型的时间复杂度大于 TAN。实验结果表明，对于给定的数据库，如果找到合适的界限值 K，其分类性能要好于 TAN。

总的来说，上述两种 TAN 变化模型具有与一般的 TAN 相同的优缺点。另外，还有一种 TAN 的变化形式称之为多网分类器，即将数据集合按照不同的类分组，对不同的类添加不同的扩展弧和构成不同的树结构，对每一个类 c_i 都建立一个关于属性变量 X_1，…，X_m 的贝叶斯网络，称为 c_i 的局部网。局部网的集合连同 C 的先验 P(C)就称为贝叶斯多网分类器。这种分类器的性能与 TAN 分类器相当。

2.2.3 增量贝叶斯分类模型

增量分类[44]是一种动态分类过程。其特点是随着分类过程的推进，训练集的规模不断扩大，分类过后的实例逐一被纳入训练集合中。也就是说，新的训练实例的加入使得分类器的参数在不断更新过程中。这种分类模型的关键是在有众多候选实例的情况下，选择什么样的实例优先加入训练集能促进分类性能的提高，使得当所有的待分类实例都被分类结束后总体分类性能最高。

优先实例的选择有两种处理策略[45]：被动选择策略与主动选择策略。

在增量分类过程中，被动策略随机地选择新的实例加入训练集，然后在新的训练集的基础上更新分类器参数。尽管这种策略算法简单，但是它是一种消极策略，因为它存在明显的不足：

首先，顺序地选择新实例往往会使学习的分类器具有顺序相关性，对数据过分敏感；

其次，遇到噪音样本时，会使这种噪音一直传播下去，影响分类精度；

另外，缺乏综合未带类别标注样本信息的能力。由于未带类别标注的样本往往包含有助于分类的信息。在这种情况下，选择好的未带类别标注的样本，把它加入到当前的分类器中，是相当重要的。

基于以上分析，本文重点介绍优先实例选择中的主动策略。这种策略对新

样本的选择是主动的，它首先选择最有利于分类器性能的样本来训练分类器，属于更高层次的，具有潜意识的学习，图 2-4 描述了这种增量分类的大致过程：

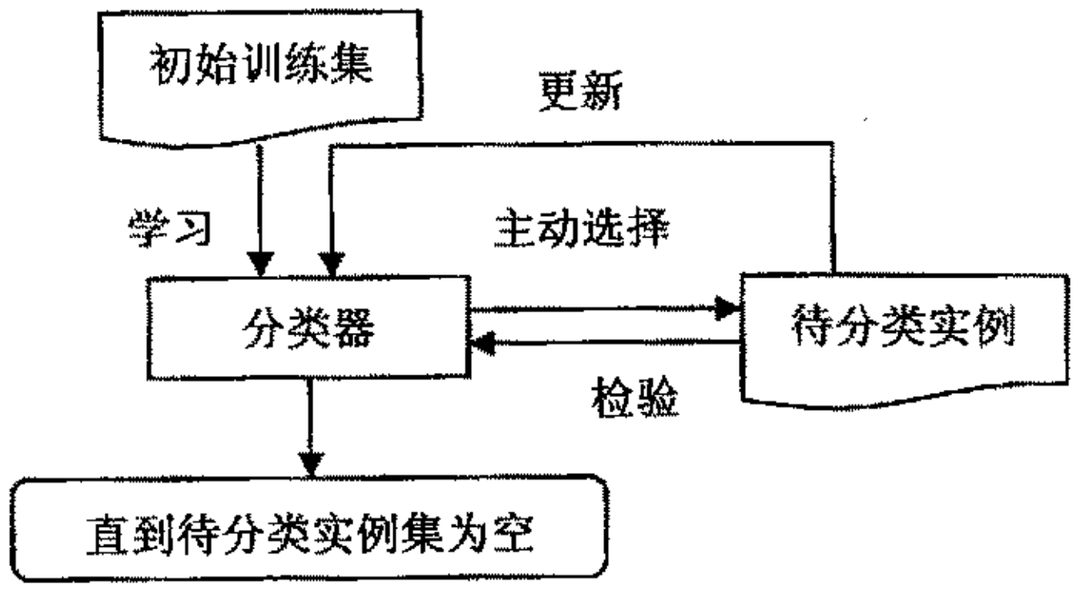


图 2-4 主动选择优先实例增量分类过程

机器学习中的主动学习并不是一个陌生的概念。Simon 和 Lea[46]在 1974 年就提出了主动学习的有关思想。他认为，机器学习不同于一般的问题求解系统，它包括对两个空间的同时搜索：假设空间和实例空间，对假设空间的搜索结果会影响实例空间的搜索方式。Winston[47]认为在学习中将要学习的下一个例子应该是几乎错过的例子“Near missing instance”。

贝叶斯技术能充分利用先验信息，这使得它成为增量学习中的最佳选择模型。同时，它更适宜于主动学习，运用它在增量分类和增量学习概率分布过程中可以大大提高效率。在此基础上，人们提出了增量贝叶斯分类模型。

本文第四章着重研究贝叶斯理论用于增量分类的问题。主要内容包括：增量分类一般理论、优先测试实例选择策略、对已有模型算法的分析以及本文在这一领域所做的工作。

2.3 小结

本章对基于贝叶斯技术的分类问题作了概述性介绍：首先给出了贝叶斯理论的基本知识。其中贝叶斯定理是贝叶斯分类模型的理论基础，所有贝叶斯分类器都围绕贝叶斯定理展开；信息度量理论在数据挖掘领域举足轻重，其中本章所介绍的信息熵、条件熵、互信息函数等都是很有实用价值的信息测度工具。

本章后半部分分类给出了常见的贝叶斯分类模型。内容涉及各种模型的分类思想，各自优缺点。介绍各种不同的分类模型是为了帮助从不同的角度理解贝叶斯分类，拓展领域思维。

第三章 半朴素贝叶斯分类模型

3.1 SNBC 模型选择的依据

第二章介绍了各种贝叶斯分类模型的分类思想，从中可以看出：最初的贝叶斯分类模型是 NBC，其独立性假设使得概率归纳变得容易而有效。

为了充分运用 NBC 的优势且在其基础上提高分类精度，人们提出了一些 NBC 变化模型以突破其严格的独立性假设限制。变化后模型的结构无一例外比 NBC 要复杂，这自然涉及到结构复杂程度与分类精确度之间关系的问题。

理论上一个较复杂的结构应该更能准确地近似描述训练数据集。因此看起来结构越复杂，它越会是一个准确的分类器。然而事实显然不是这样。研究表明，复杂的结构将会产生一个过分拟和的问题[36]，也就是说，分类器在学习训练数据时性能良好，然而在用于估计一个新实例时出错率却很高。这样一来我们就处于这样一种境地：如果倾向于简单的结构，其简单性所带来的约束限制会经常被突破；如果倾向于复杂的结构，就有可能产生过分拟和的问题。

在上述情形下，其中一个平衡策略是首先限制网络的结构复杂度，然后在其基础上寻求能近似描述数据集的最好的结构。半朴素贝叶斯分类模型(SNBC:Semi-Naïve Bayesian Classifiers)就是对这一策略的一种有效尝试。SNBC 模型提出的时间还不太长，模型原理简单，其关键是如何通过启发式搜索过程有效地将关联度大的基本属性“并”在一起构成组合属性。这里的“并”只在计算过程中体现出来，是概念层次上的一个抽象过程，并不是真正合并。

本章主要从理论与实践角度分析 SNBC 模型的实现。

3.2 SNBC 的定义

定义[30]3.1：给定一个相互独立的 N 个观察值集合 $D=\{x^1,\ldots,x^N\}$，其中 $x^i=(a_1^i,a_2^i,\cdots a_n^i,c^i)$，$A_1$，$A_2$，…，$A_n$表示 n 个原始属性，类别属性用 C 表示，BSNBC 是满足下述条件的最大似然贝叶斯网络：

(1) 网络由类别属性 C 以及 m 个“组合属性” B_1，B_2，…，B_m（$1\le m\le n$）构成，其中 $B_I=\{A_{I_1},A_{I_2},\cdots A_{I_{k_I}}\}$是属性集$\{A_1,A_2,\cdots,A_n\}$的子集。

(2) “组合属性”之间不存在覆盖，且所有组合属性结点的并集构成原属性集。即：

① 当$i\neq j$且$1\le i,j\le m$时，$B_i\cap B_j=\phi$；

② $B_1\cup B_2\cup\cdots\cup B_m=\{A_1,A_2,\cdots,A_n\}$。

(3) B_i 与 B_j 之间相对于类别属性 C 而言条件独立，即当 $i \neq j$ 且 $1 \leq i, j \leq m$ 时，$P(B_i, B_j \mid C) = P(B_i \mid C)P(B_j \mid C)$。

如果事先给定一个常数 K 且规定每个组合属性结点 $B_l(1 \leq l \leq m)$ 中基本属性的个数(基值 cardinality)不超过常数 K，那么这样的模型叫做有界半朴素贝叶斯分类模型 (BSNBC :Bounded Semi-Naïve Bayesian Classifier)，这样的限制条件可以控制模型的复杂度。

在运用上述 SNBC 进行分类时，组合属性 B_i 与 B_j 之间的条件独立关系与 NBC 中基本属性间的条件独立关系一致。此时 SNBC 的与 NBC 中的定义式(2.16)相对应的定义式是：

$$c_{SNBC} = \arg\max_{c_i \in C} p(c_i) \prod_{j=1}^{m} p(b_j \mid c_i) \tag{3.1}$$

这里的 b_j 不是 NBC 中的单个数值，而是一个数值向量。

3.3 SNBC 模型的三个发展阶段

3.3.1 传统 SNBC 模型：

早期 SNBC 满足定义 3.1，其特点是每个组合属性 $B_i(1 \leq i \leq m)$ 中包含两个基本属性。研究表明，在一定情况下这种模型对于 NBC 来说有一定的改进[35]，但大部分情况下性能与 NBC 相当。下一节的两个与网络最大似然条件相关的引理揭示了该模型的局限性。

3.3.2 有界半朴素贝叶斯分类模型(BSNBC)：

如定义 3.1 所述，这种模型放宽了传统 SNBC 对组合属性所含基本属性个数的要求，这里规定每个组合属性节点中基本属性的个数不超过某个给定常数 K。显然，当 BSNBC 中的 K 值增大至特征属性个数 n 时，网络就成了完全图。第 2.2.2 节介绍了这种完全贝叶斯网络分类模型的优缺点。另一方面，如果 K 值为 1，那就蜕变成了朴素贝叶斯网络。

3.3.3 K-规范 BSNBC 模型：

为了便于设计算法，人们提出了 K-规范 BSNBC 模型，要求每个组合属性中包含同样多的 K 个属性。这种模型的分类性能不一定比非规范 BSNBC 模型优越[35]。这是因为：为了使得每个组合属性的基都等于 K，它容易将某些本来关联不是很大的基本属性“强扭”在一起。往往这种组合增加了运算量可对分类性能未必有好处。

正因如此，本文的工作集中于一般的 BSNBC 模型，提出了一种有效的

BSNBC 模型构造算法。

3.4 关于对数似然的两个引理

本节的两个引理[35]给出了 BSNBC 模型的理论依据：

引理 1：一个 SNBC 的描述为 l_{SNBC} 形式的对数似然函数可以表示为如下形式：

$$l_{SNBC} = -\sum_{j=1}^{m} H(B_j) \tag{3.2}$$

其中 $H(B_i)$ 是组合属性 B_i 的熵。一个有 k 个变量的集合 $\{X_1, X_2, \cdots, X_k\}$ 中变量间的熵值被定义为：

$$H(X_1, X_2, \cdots, X_k) = -\sum_{x_1, \cdots, x_k} P(x_1, \cdots, x_k) \log P(x_1, \cdots, x_k)$$

其中小写字母 $x_i (1 \le i \le k)$ 代表对变量 X_i 的具体取值。

证明：对于定义 3.1 所描述的 SNBC 模型，假设 A_1，A_2，…，A_n 已经被分在 m 个不相互覆盖的组合属性 B_1，B_2，…，B_m 中，现用 $a_i (1 \le i \le n)$ 表示基本属性 A_i 的取值，用数值向量 $b_j (1 \le j \le m)$ 表示组合属性 B_j 的取值，那么其对数似然可以表示成：

$$\begin{aligned}
l_{SNBC} &= \sum_{(A_1, A_2, \cdots, A_n)} P(a_1, a_2, \cdots, a_n) \log P(a_1, a_2, \cdots, a_n) \\
&= \sum_{(B_1, B_2, \cdots, B_m)} P(b_1, b_2, \cdots, b_m) \log P(b_1, b_2, \cdots, b_m) \\
&= \sum_{(B_1, B_2, \cdots, B_m)} P(b_1, b_2, \cdots, b_m) \sum_{j=1}^{m} \log P(b_j) \quad \cdots\cdots \text{组合节点之间条件独立} \\
&= \sum_{j=1}^{m} \sum_{(B_1, B_2, \cdots, B_m)} P(b_1, b_2, \cdots, b_m) \log P(b_j) \\
&= \sum_{j=1}^{m} \sum_{B_j} P(b_j) \log P(b_j) \\
&= -\sum_{j=1}^{m} H(B_j)
\end{aligned}$$

引理 2：令 μ 和 μ' 是在数据集 D 上的两个 SNBC。如果 μ 能由 μ' 中的组合节点组合而成且 μ' 本身的组合节点不用拆分，那么 μ 比 μ' 更能近似描述数据集 D。

证明：不失一般性，假设数据集 D 上的其中一个 SNBC 的组合属性构成是 $\mu = (B_1, B_2, \cdots, B_m)$，另外还有 $\mu' = (B_1, B_2, \cdots, B_{m-1}, B_{m1}, B_{m2})$。且存在下述关系：

$$B_{m1} \cap B_{m2} = \phi, \quad B_{m1} \cup B_{m2} = B_m$$

由引理 1，我们得到：

$$l^{\mu}_{SNBC} = -\sum_{j=1}^{m} H(B_j) = -\sum_{j=1}^{m-1} H(B_j) - H(B_m)$$
$$\geq -\sum_{j=1}^{m-1} H(B_j) - H(B_{m1}) - H(B_{m2})$$
(信息熵中：$H(XY) \leq H(X) + H(Y)$)
$$= l^{\mu'}_{SNBC}$$

即 μ 的对数似然大于等于 μ' 的对应似然值，由定义可知，μ 比 μ' 更能近似描述数据集的分布特性。

引理 2 表明，SNBC 中组合属性的基值越大，分类器的性能越高。这也说明了只能组合两个基本属性为组合属性的传统 SNBC 的局限性。BSNBC 与 K-规范 BSNBC 模型中组合节点最多可以容纳 K 个基本属性，突破了传统 SNBC 的限制。

3.5 BSNBC 分类算法

在此着重分析 BSNBC 与 K-规范 BSNBC 模型的算法实现，并在此基础上引出本文的算法。从理论上分析本文算法的依据。

从 BSNBC 的定义可以看出，在合并形成组合属性的过程中。模型的搜索空间是 $\sum_{\{k_1,k_2,\cdots,k_m\}\in G} C_n^{k_1} C_{n-k_1}^{k_2} \cdots C_{n-\sum_{i=1}^{m-1} k_i}^{k_m}$，$G = \left\{ \{k_1,k_2,\cdots,k_m\} : \sum_{i=1}^{m} k_i = n, 0 \leq k_i \leq K \right\}$，$k_i$ 对应于 BSNBC 中组合属性 B_i 所含基本属性的个数。学习的目的是从上述模型空间中选择 l_{BSNBC} 最大的网络结构。如果不采用启发式搜索过程，算法的时间复杂度相当大。

由引理 2 可知，在给定边界值 K 时，我们不应该把特征属性集分成太多的小子集。或者说更可能的情况是，可以把这些子集结合在一起构成一个新的子集，只要新子集的基不大于 K 值即可。这样一来，新的 BSNBC 将比合并前的 BSNBC 表达力更强。从这一观点出发，我们可以减小 BSNBC 的搜索空间至 K-规范 BSNBC 空间，因为基本上可以断定在 K-BSBNC 空间范围内不可能存在一个比 K-规范 BSBNC 更具表达力的 BSNBC(尽管这一断定不很实际[35]！)。

针对 K-规范 BSNBC，人们提出了一些切实可行的算法。现介绍其中的整数编码(IP-Integer Programming)算法的实现思想：

3.6 IP 编码算法

算法的实质[30]：从基本特征属性集中找到 $m = [n/K]$ 个 K-基子集，这些子集满足 BSNBC 定义条件，且使得公式(3.2)对数似然值最大。这里[x]表示取对于 x 的最靠近 x 的整数值。

很显然这是一个组合问题。然而如果使用贪婪搜索算法去寻求最优解，那将是行

不通的。我们很容易算出贪婪搜索的代价将是 $n!/(K!)^{\lfloor n/K \rfloor}$，对于一个简单的例子 n=18, K=3 来说，其可能模型的数目将会达到 1.372×10^{11}。

转化为 IP 形式后问题可描述为：

$$Min \sum_{V_1,V_2,\cdots,V_K} x_{V_1,V_2,\cdots V_K} H(V_1,V_2,\cdots,V_K) \tag{3.3}$$

$$\text{对于}(\forall V_K): \quad \sum_{V_1,V_2,\cdots,V_{K-1}} x_{V_1,V_2,\cdots,V_K} = 1 \qquad (x_{V_1,V_2,\cdots,V_K} = \{0,1\}) \tag{3.4}$$

这里 $V_1,V_2,\cdots,V_K$ 代表任意 K 个特征属性。公式（3.3）反映了这样的事实：对于任何一个属性而言，它仅仅能从属于一个子集。也就是说，当它属于某一个子集时，它一定不会属于另一个子集。$H(V_1,V_2,\cdots,V_K)$ 表示属性集 $\{V_1,V_2,\cdots,V_K\}$ 的熵。

IP 算法可以通过不同的方式来实现[35]，如切盘算法(Cutting Plane)、模拟退火算法(Simulating Annealing)以及近似线性编码(LP-Linear programming)等。通过放松 $x_{V_1,V_2,\cdots,V_K} = \{0,1\}$ 的限制，使它们满足 $0 \le x_{V_1,V_2,\cdots,V_K} \le 1$，IP 问题就转化成了 LP 问题。假设 LP 中所有 $x_{V_1,V_2,\cdots,V_K}$ 构成的集合用 X 表示。为了快速减小搜索空间，在 LP 执行过程中有如下操作[31]：

① 如果所有的基本特征属性都组合成功，构建规范 BSNBC 的过程结束。

② 置 X 中最大的 $x_{V_1,V_2,\cdots,V_K}$ 的值为 1，其下标是 $\{V_{M_1},V_{M_2},\cdots,V_{M_K}\}$，从 X 中删除 $x_{\{V_{M_1},V_{M_2},\cdots,V_{M_K}\}}$。

③ 将 X 中与 $\{V_{M_1},V_{M_2},\cdots,V_{M_K}\}$ 有覆盖所有的的 $x_{V_1,V_2,\cdots,V_K}$ 的值置为 0，从而将它们从 X 中删除。

④ 转①。

至于可能会碰到的 n 不能被 K 整除的情形，理论上此时当然无法找到一个 K-规范 BSNBC。解决这一问题的方案是：假设 $(n \mod K) = l \neq 0$, 先从所有的特征属性集的基大小为 l 的子集中选择一个熵值最小的集合作为一个组合属性。然后在剩下的 $n-K$ 个特征属性组成的集合上执行上述构造过程。

IP 以及 LP 是目前规范 BSNBC 中用得比较多的算法，实验也证明了其有效性[35]。但是这种算法存在这样的问题：为了尽量使得每个组合属性中基本属性的个数达到给定的界限值，容易将本来关联不大的、几乎可以断定为相互间条件独立的基本属性组合在一起，这样对分类往往产生负面作用。另外，在构造 BSNBC 的过程中，每次都要用似然公式进行检验，这里存在一定的盲目性，占用了大量的运行时间。针对上述问题在不牺牲分类精度的情况下本文提出了一种新的条件互信息度量 BSNBC(CMI-BSNBC:Conditional Mutual Information BSNBC)分类学习算法。

3.7 CMI-BSNBC 分类学习算法

本文引入公式(2.13)定义的条件互信息函数。条件互信息值反映了在给定条件属性的条件下特征属性间的相互关联程度大小，条件互信息值越大，关联度越大。这里需要说明的是，定义式：

$$I_p(X;Y|C)=\sum_{X,Y,C}P(X,Y,C)\log_2\frac{P(X,Y|C)}{P(X|C)P(Y|C)} \tag{3.5}$$

中 X 与 Y 既可以是 BSNBC 中的基本属性，还可以是算法过程中产生的组合属性，也就是说，如果 $X=\{X_1,X_2,\cdots,X_{k_x}\}$，$Y=\{Y_1,Y_2,\cdots,Y_{k_y}\}$，其中 k_x 与 k_y 分别是组合结点 X 与 Y 的基，上述计算公式依然成立。

为了避免将关联度不大的属性放到一起构成组合属性，在算法中引入条件互信息阈值 θ。在计算过程中保证每个基值小于 K 的组合属性中属性间的条件互信息值都不小于 θ。这样一来，所学习的结构就不会拘泥于规范 BSNBC。

3.7.1 算法描述

算法过程中要用到的四个列表：

ComAttrList：用于存放最终形成的组合属性，ComAttrList 中各项代表一个组合属性，列表初始化为空；

MidList：过渡组合属性列表，该表最初由原始基本属性构成；

MCPTPtr：互信息列表，采用链表结构，其中存放 MidList 中当前过渡组合属性之间的互信息值，初始化为空；

CPTPtr：条件概率列表，链表结构，用于存放 MidList 当前各组合属性相对于类别属性的条件概率信息。

输入：$D=\{x^1,\ldots,x^N\}, x^i=(A_1^i,A_2^i,\cdots A_n^i,C^i)$,

A_1，A_2，…，A_n：原始基本属性，

C：类别属性，集合类型，

K:组合属性所含基本属性个数边界值，

θ:条件互信息阈值.

输出：组合属性所包含各基本属性间的条件互信息不小于 θ 且组合属性最多含有 K 个基本属性的 BSNBC，将其用于实例分类。

图 3-1 是算法基本流程，附录二给出了主要实现代码。

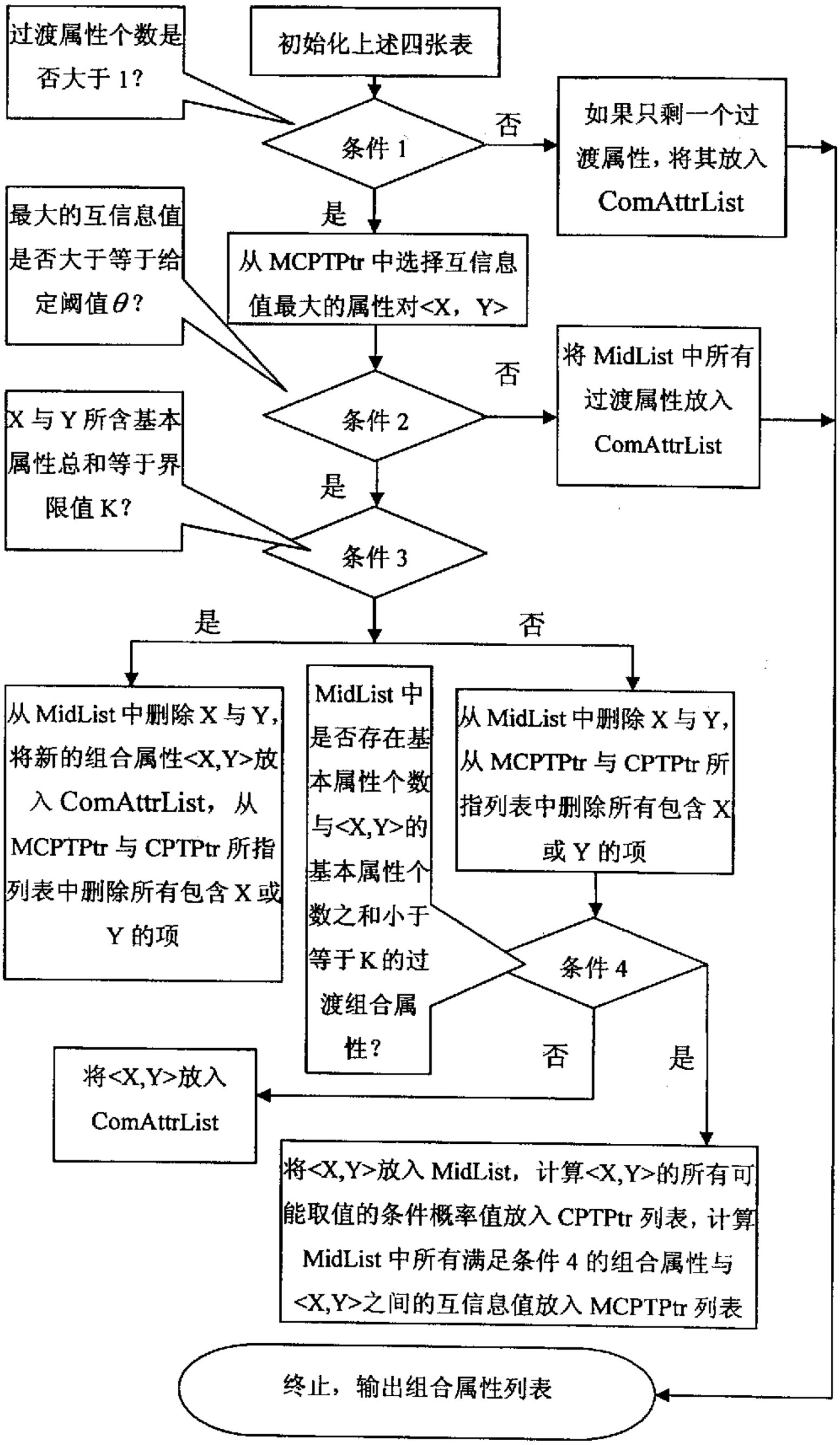


图 3-1 CMI-BSNBC 属性这算法基本流程

3.7.2 算法复杂度分析比较

该算法逐步构成组合属性节点，最终保证每个组合属性所包含的基本属性个数不会超过 K，对于基本属性个数小于 K 的组合属性，其属性间的条件互信息值都不小于 θ 。由算法过程可知，构造组合属性的过程中每个基本属性仅被选择一次，选择一个基本属性的时间复杂度是 O(n)，其中 n 是基本属性个数，所以构造组合属性的时间复杂度是 $O(n^2)$ 。另外，此算法免去了每次用似然函数检验的过程。上述两点是本算法跟其它相关的算法相比较而言最突出的优势所在。

在算法过程中需要计算任意两个属性间的条件互信息 $I_p(A_i;A_j|C)$ 。其时间复杂度是 $O(Nn^2cw^2)$ ，其中 N 是观察值集合 D 中元素的个数，c 是类别属性可能取值的个数，w 是每个基本属性可能取值个数的最大值。在后续的循环过程中还要计算新的条件互信息值，不过这只是一个局部过程。

因此条件互信息度量 BSNBC 的总的时间复杂度是 $O(Nn^2cw^2+n^2)$，由于对于给定数据集而言，其中 c，w 是常量，所以上述复杂度可写成 $O(Nn^2)$。

研究表明[35]，在 K<<n 的前提下，借助于 LP 实现的 K-规范 BSNBC 模型的总的时间复杂度是 $O(n^{K+2})$。

由上述分析可知，本文的 BSNBC 学习算法有明显的效率优势。

3.8 实验结果与分析

本文实现了 CMI-BSNBC 分类模型，运用实验数据对其分类性能进行了综合分析。

3.8.1 实验数据

本实验所用的数据来自 UCI(University of California, Irvine)机器学习数据库，可从 URL: http://www.ics.uci.edu/~mlearn/MLRepository.html 获得。本文从中选择两个数据集：tic-tac-toe 与 Vote，它们的基本情况是：

表 3-1 实验数据基本情况表

数据集	特征属性数目	类别数目	实例数	离散连续	测试方式	有无缺失数据
tic-tac-toe	9	2	958	离散	CV-5[1]	无
Vote	15	2	435	离散	CV-5	无

[1] CV-n 是一种测试策手段，其工作步骤是：首先将所有实例分成随机的 n 等份，然后进行 n 次测试，每次以其中的一份作为测试集，另外的 n-1 份作为训练集，取分类精度的平均值作为对分类器的分类精度评估。

tic-tac-toe 数据集的特征属性及类别属性的名称与可能取值：

特征属性：

1. top-left-square: {x, o, b}
2. top-middle-square: {x, o, b}
3. top-right-square: {x, o, b}
4. middle-left-square: {x, o, b}
5. middle-middle-square: {x, o, b}
6. middle-right-square: {x, o, b}
7. bottom-left-square: {x, o, b}
8. bottom-middle-square: {x, o, b}
9. bottom-right-square: {x, o, b}

类别属性：

class: {positive, negative}

Vote 数据集的特征属性及类别属性的名称与各自可能取值：

特征属性：

1. handicapped-infants: {y, n}
2. water-project-cost-sharing: {y, n}
3. adoption-of-the-budget-resolution: {y, n}
4. physician-fee-freeze: {y, n}
5. el-salvador-aid: {y, n}
6. religious-groups-in-schools: {y, n}
7. anti-satellite-test-ban: {y, n}
8. aid-to-nicaraguan-contras: {y, n}
9. mx-missile: {y, n}
10. immigration: {y, n}
11. synfuels-corporation-cutback: {y, n}
12. education-spending: {y, n}
13. superfund-right-to-sue: {y, n}
14. crime: {y, n}
15. duty-free-exports: {y, n}
16. export-administration-act-south-africa: {y, n}

类别属性：

class: {democrat, republican}

在实验过程中，在不影响区分的情况下对上述各种名称进行了简化。去除名称中的间隔符号并缩短了部分过长名称的长度。

3.8.2 CMI-BSNBC 分类模型介绍

本实验在 windows 环境下以 Microsoft 公司的 Visual C++.net 2003 作为开发工具，运用面向对象的设计思想。所实现系统的开放性使得它具有很强的扩充能力。另外，该系统具有友好的交互界面，操作简单。

操作步骤如下：

(1) 主界面：

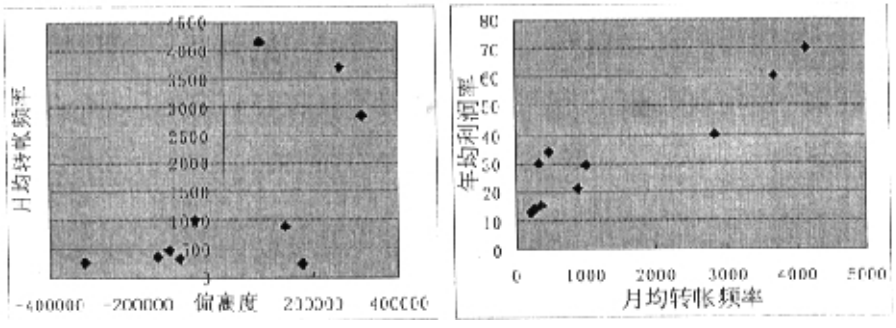


图 3-2 CMI-BSNBC 分类模型主界面

主界面中三个主菜单项[选择数据]、[数据处理]、[测试]分别对应系统的三个功能模块：

(2) 选择数据：

选择数据模块完成两个功能：选择数据源、选择数据表。

点击[选择数据]菜单出现图 3-3 所示选择数据源窗口：

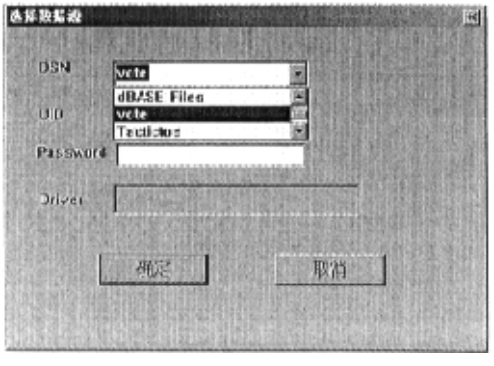


图 3-3 选择数据源窗口

下拉列表中显示了当前系统所有 ODBC 数据源。选择其中一个数据源按[确定]按钮进入选择数据表窗口(图 3-4)：

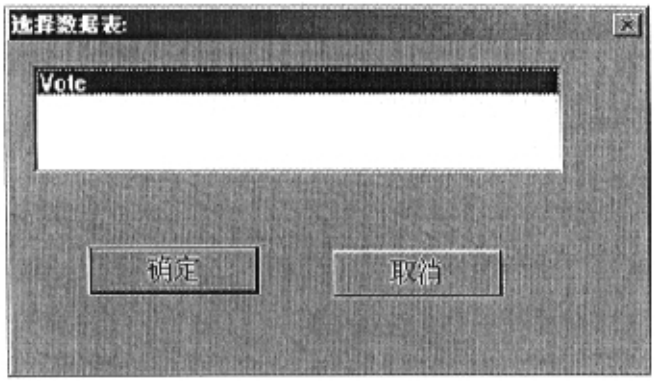


图 3-4 选择数据表窗口

由于某个数据源可能包含有多个数据表，所以设计上述选择窗口以从中仅选择一个数据表。

(3) 数据处理：

该模块完成两个功能：组合属性、数据分类。

点击[数据处理]->[组合属性]，出现图 3-5 所示参数编辑窗口：

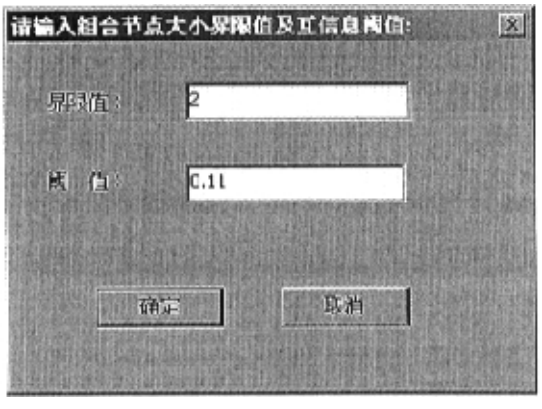


图 3-5 组合属性参数输入界面

组合属性时需要输入两个参数：组合属性节点所含基本属性个数的界限值以及条件互信息阈值。可以改变窗口中的默认值。点击[确定]按钮进入属性组合过程，组合结束会出现提示信息。

属性组合成功后，点击[数据处理]->[CMI-BSNBC 分类]可以对用户输入的单个实例进行分类。

(4) 测试：

点击菜单[测试]-[测试 BSNBC]，出现下述窗口供用户输入交叉认证 CV-K 中的参数 K：

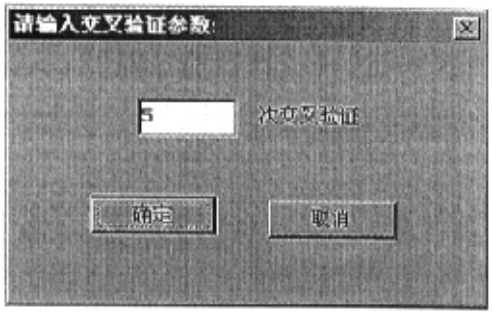


图 3-6 交叉验证参数输入窗口

测试的结果保存在以当前数据源名称命名的文本文件中。假设当前的数据源是 vote，测试结果保存在 vote.txt 中。图 3-7 显示了其中一次测试结果。

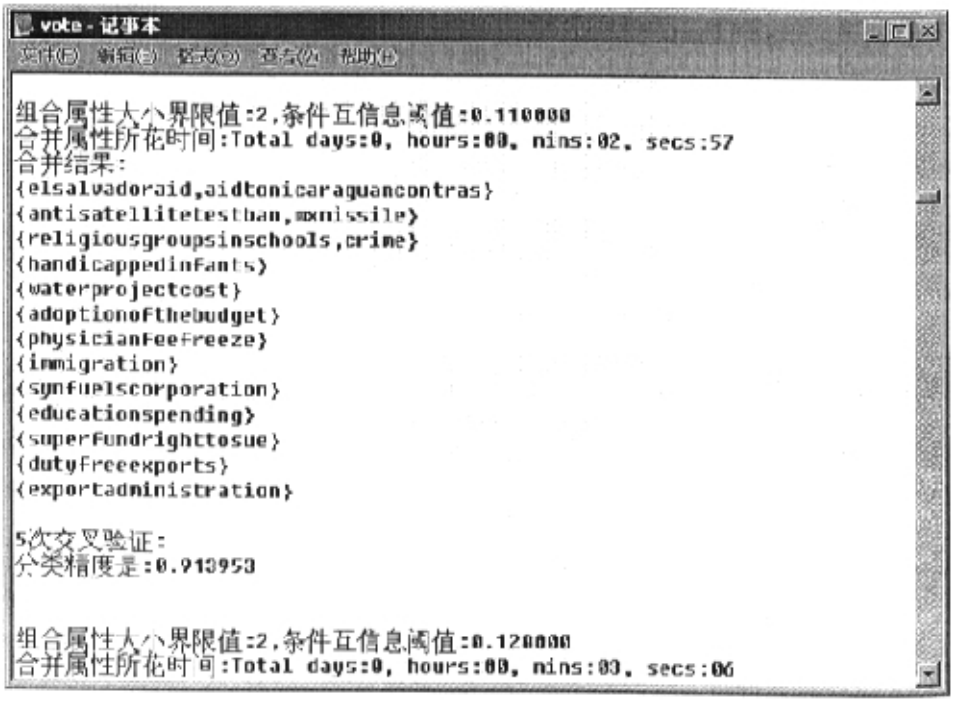


图 3-7 数据源 vote 分类测试结果

3.8.3 实验结果分析

针对每个数据源，分别取不同的参数<θ,K>组合值来进行实验。考察属性组合过程中所花时间及所产生 BSNBC 的分类精度。这两者反映了不同的组合

参数对组合效率及分类精度的影响。实验结果如下：

表 3-2 数据源 tictactoe 属性组合时间 (单位：秒)

阈值 \ K	2	3	4	5
0.02	1.17	2.10	3.83	8.17
0.03	0.85	2.37	4.52	8.18
0.04	1.25	2.33	4.48	8.33
0.05	1.20	2.27	4.42	8.58
0.06	1.88	2.40	4.40	8.33
0.07	1.20	1.18	0.95	1.03

从上表可以看出，随着 K 值的增大，属性组合的时间消耗越来越大。最后一行所示属性组合时间都很小，这是由于阈值较大，原始属性两两间的条件互信息无一大于阈值，此时原始属性间没有组合操作，所以较早退出函数体。此时的 BSNBC 就蜕化成了 NBC。

表 3-3 数据源 tictactoe 分类精度 (单位：%)

阈值 \ K	2	3	4	5
0.02	***63.0526***	***65.8947***	***60.4211***	***60.0000***
0.03	62.8421	***65.8947***	***60.4211***	***60.0000***
0.04	62.8421	***65.8947***	***60.4211***	***60.0000***
0.05	62.8421	65.5789	***60.4211***	***60.0000***
0.06	62.8421	65.5789	***60.4211***	***60.0000***
0.07	62.5263	62.5263	62.5263	62.5263

（注：下划线数据表明是 NBC 的分类精度，加粗斜体数据代表规范 BSNBC 的分类精度，加粗下划线数据代表 CMI-BSNBC 的最优性能）

综合组合时间与分类性能两因素可以看出，对数据源 tictactoe 而言，在合适选择组合参数(本例中 K=3，θ=0.03；K=3，θ=0.04)的条件下，CMI-BSNBC 的分类性能明显优于 NBC，与规范 BSNBC 分类最优性能相当。

上图中当 K=4 或 K=5 时，分类性能较弱。其原因是：

由 3.4 节引理 2 可以分析出，对于给定的数据源，如果组合属性 A 包含组合属性 B 的所有基本属性，那么 A 基本属性间的条件互信息值会大于 B 中对应的互信息值。对本例而言，当 K 的值较大时，往往会出现下面的情况：

最初被选择的条件互信息值最大的两个基本属性间拥有最强的相关性。这

是算法过程中产生的第一个过渡性组合属性。由于此过渡组合属性的基值小于界限值 K，所以它还会与其它基本属性进行组合，由引理 2 可知它与任何其它基本属性间的条件互信息值肯定大于其本身的互信息值。所以属性组合过程会一直进行下去，直到基值等于界限值 K 或在基本属性个数小于 K 的情况下将所有基本属性合并为一个组合属性为止。

这种组合过程可能会产生这样的结果：形成为数不多的基值靠近于界限值 K 的“大”的组合属性。这样的组合属性中只有最初被组合的属性间具有较强的相关性。随着新属性的加入，组合属性整体相关性在增加，但是属性相互之间的相关性却不强。同时可能将本该属于其它组合属性的基本属性合并了进来。这对分类性能没有什么促进作用。

由上述分析可以看出，本算法只有在界限值 K 较小时才会有良好的表现。实际上，界限值过大时还存在以下不足：

首先：本章一开始就分析指出，当 K 值较大时，可能会出现过分拟和的情况。

其次，由表 3-2 可以看出，随着 K 值的增大，属性组合的时间会随之加长。

鉴于以上分析，本文在下面针对数据源 vote 的实验过程中对界限值 K 只考虑两种取值：2 和 3。

表 3-4 数据源 vote 属性组合时间（单位：秒）

阈值 \ K	2	3
0.03	3.00	6.37
0.04	3.00	5.82
0.05	2.97	6.15
0.06	2.80	6.20
0.07	3.12	6.08
0.08	3.10	6.18
0.09	3.12	6.15
0.1	3.12	5.50
0.11	2.95	5.25
0.12	3.10	5.06
0.13	3.03	4.93
0.14	3.25	4.58
0.15	3.10	4.78
0.16	3.12	5.06

0.17	3.33	4.70
0.18	3.77	4.22
0.19	3.38	4.15
0.2	3.27	4.55
0.21	3.23	4.30
0.22~0.31	2.80~3.20	4.12~4.80
0.32	2.95	3.52

（注：当 K 值不变时，阈值在 0.22~0.31 范围内时形成的组合属性完全一样，当阈值取 0.32 时不进行属性组合，此时蜕化为 NBC）

表 3-5 数据源 vote 分类精度（单位：%）

阈值 \ K	2	3
0.03	*90.6977*	*91.8605*
0.04	*90.6977*	*91.8605*
0.05	90.4651	*91.8605*
0.06	90.4651	*91.8605*
0.07	90.4651	*91.8605*
0.08	90.6977	91.6279
0.09	90.6977	91.6279
0.1	90.4651	91.6279
0.11	91.3953	92.3256
0.12	91.3953	92.3256
0.13	91.3953	92.3256
0.14	91.3953	92.3256
0.15	91.3953	92.3256
0.16	91.3953	92.3256
0.17	90.2326	91.3953
0.18	90.2326	91.3953
0.19	90.2326	91.3953
0.2	90.2326	91.3953
0.21	90.0000	91.3953
0.22~0.31	90.0000	91.3953
0.32	90.0000	90.0000

由上表可以看出，当 K=3，θ=0.11~0.16 时，CMI-BSNBC 对 vote 数据源的分类效果优于 NBC 与规范 BSNBC。

从上述实验可以看出，对于具体的数据源，不同的参数组合对分类效果会产生不同的影响。在实际操作过程中，为数据指定合适的组合参数是构建 BSNBC 的一个重要环节。

3.9 小结

本章重点介绍了 SNBC 的分类原理，着重分析了 SNBC 模型的实现。引入了本文所提出的 CMI-BSNBC，用实验给予实现并将实验结果与其它分类方法进行比较，比较结果证明了算法的有效性。

第四章 增量贝叶斯分类器

在第二章的基础上现在来深入讨论增量贝叶斯分类模型。本章主要内容有：分类过程中常见的新测试实例的抽样策略、贝叶斯增量分类中参数的学习与推理及本文所做的工作。

4.1 常见的新实例抽样策略

如第二章所述，增量分类是一种动态分类过程。在分类过程中，新的测试实例被逐渐纳入训练集，每次新实例的加入都导致分类参数的更新。也就是说，新实例的加入对后续分类过程会产生影响，并且这种影响会随着分类过程的推进逐步累积。很显然，在这种分类模型中，如何从众多候选实例中选择优先实例加入训练集是关键。我们的目的是选择能够促进分类性能提高的实例进入训练集，即，应该优先选择对后续分类产生积极影响的实例。

从选择策略上来分，优先实例的选择有两种处理策略，即被动选择策略与主动选择策略。基于第二章的分析，在此仅讨论主动选择策略。

4.1.1 基于确定性与不确定性的主动策略

确定性抽样每次从候选实例集中选取能由当前分类器比较准确地预测其类别的实例进入训练集；不确定性抽样每次选择不能由当前分类器准确分类的奇异的实例。

确定性抽样与不确定性抽样各有所长、各自适合的情形有所不同。确定性抽样适合于初始训练集较大的情形，它偏向于选择“最喜欢”的样本，这样，有时容易使某些含有较为丰富信息的样本，由于较晚地加入到训练集，而影响分类的性能[44]；不确定性抽样适合于初始训练集较小的情形，此时选择奇异的实例的依据是：由于初始训练集中实例数目不大，从训练集学习得到的分类器往往具有片面性，选择奇异的实例能避免这种片面性对分类的消极影响。不确定性抽样的局限在于：正因为它特别关注奇异的实例，当前分类器对新实例的分类误差必然偏大，随着学习过程的进行，这种误差还会继续传播下去，从而影响分类的性能。

一种综合考虑确定性抽样与不确定性抽样特点的策略是：同时兼顾确定性实例与不确定性实例。下面介绍的“基于最大熵最小熵的主动学习[45]”就是一种平衡策略。

第二章介绍了熵的相关理论，我们知道它是一种衡量事物不确定性的手段。基于最大熵最小熵的主动学习策略正是利用了它的这一特性。

最大最小熵的处理方法是：首先从测试样本中选择出类条件熵最大和最小的候选样本(MaxExample,MinExample)，然后将这两个样本同时加入到训练集中。类条件熵最大的样本的加入，使得分类器能够对具有特殊信息样本的及早重视；而类条件熵最小的样本是分类器较为确定的样本，对它的分类也更加准确，从而部分地抑制了由于不确定性样本的加入而产生的误差传播问题。

这种策略的不足在于：只考虑了待选测试样本本身的特点，不能充分利用原有训练实例集的先验信息。而当初始训练集较大时，先验信息在增量分类过程中的作用尤其不能被忽视。

4.1.2 基于分类损失的主动策略

分类损失是另一种衡量分类精度的标准。假设当前的训练集为 D，实例集 T 在训练集 D 的基础上的分类损失有如下两种计算方法：

对数损失：

$$Loss_{\log} = \frac{1}{|T|}\sum_{x\in T}\sum_{i}\hat{P}_D(c_i \mid x)\log \hat{P}_D(c_i \mid x) \tag{4.1}$$

0-1 损失：

$$Loss_{0-1} = \frac{1}{|T|}\sum_{x\in T}\left(1-\max_{i}\left\{\hat{P}_D(c_i \mid x)\right\}\right) \tag{4.2}$$

其中 c_i 是类别标签。

这两种损失函数在绝大部分情况下，它们的顺序关系是一致的，为简便起见，在本文中使用 0-1 损失。

宫秀军等[44]提出了一种基于分类损失的增量贝叶斯分类模型，其基本思想是：从测试集 T 中选择有助于提高当前分类器精度的实例，将其加入训练集并更新参数。这里衡量分类器精度的标准是其对测试集中实例的分类效果。在 0-1 损失假设前提下，通过最小化当前的分类损失来选择有利于分类器性能的测试实例。

上述分类思想考虑到了测试实例的特性，但是也存在一定的不足：

首先，训练集是整个增量分类过程的基础，在增量学习过程中训练集的先验知识没得到充分利用。尽管判断测试实例的类别属性是建立在训练集基础上，但在确定优先选择什么样的测试实例加入训练集时没用到先验知识。

其次，没有考虑到新加入的测试实例对当前训练集的影响。测试实例的加入有可能强化了训练集 D 的知识储备，也有可能起弱化作用，这种作用在算法中应该有所考虑。

针对上述问题，本文提出了一种综合考虑上述因素的贝叶斯增量分类模型

TSCL-SLA(Training Set Classification Loss Based Sequence Learning Algrithm)，TSCL-SLA 考虑到了先验知识的传递以及新实例对当前训练集的影响作用。在具体介绍模型之前，首先给出贝叶斯增量分类过程中参数的学习与推理。参数学习过程体现了模型的增量特性以及先验知识传递特性。

4.2 贝叶斯增量分类过程中参数的学习与推理

由第二章的分析可知，贝叶斯分类过程中给测试实例分类的依据是取具有最大后验概率的类为对应实例的类，即我们所判定的实例 $x=(x_1,x_2,\cdots,x_m)$ 的类别标签 c_{tag} 应满足：

$$c_{tag}=\arg\max_{c_i\in C} P(C=c_i\mid x) \tag{4.3}$$

运用朴素贝叶斯的条件独立性假设可以得到下式：

$$P(C=c_i\mid x)=\frac{P(C=c_i)\times P(x\mid C=c_i)}{P(x)}=\frac{P(C=c_i)\times\prod_{j=1}^{m}P(X_j=x_j\mid C=c_i)}{\sum P(C=c_i)\times\prod_{j=1}^{m}P(X_j=x_j\mid C=c_i)}$$

$$=\frac{\theta_i\times\prod_{j=1}^{m}\theta_{j|i}}{\sum\theta_i\times\prod_{j=1}^{m}\theta_{j|i}} \tag{4.4}$$

其中：$\theta_i=P(C=c_i)$，$\theta_{j|i}=P(X_j=x_j\mid C=c_i)$。

对于具体实例 x 而言，P(x)是常数，所以在仅需判断 x 的类别标签 c_{tag} 时没有必要求出具体的 $P(C=c_i|x)$，只须用下式判断即可：

$$c_{tag}=\arg\max_{c_i\in C}P(C=c_i\mid x)=\arg\max_{c_i\in C}M\cdot P(C=c_i)\times P(x\mid C=c_i)$$

$$=\arg\max_{c_i\in C}\theta_i\times\prod_{j=1}^{m}\theta_{j|i} \tag{4.5}$$

公式中的 M 式正规常数。

下面给出训练实例及测试实例的参数 θ_i 与 $\theta_{j|i}$ 的具体计算方法：

在无信息 Dilichlet 共轭先验假设条件下，训练集 D 中 θ_i 与 $\theta_{j|i}$ 的计算公式是：

$$\theta_i=P(C=c_i)=\frac{1+count(c_i)}{|C|+|D|}\qquad\theta_{j\zeta|i}=P(X_j=X_{j\zeta}\mid C=c_i)=\frac{1+count(X_{j\zeta}\wedge c_i)}{|X_j|+count(c_i)}$$

其中，某属性 X_j 可能有多个取值，公式中的 $X_{j\zeta}$ 表示 X_j 的第 ζ 个取值，$|X_j|$ 表示属性 X_j 的取值个数。

上述计算公式只是在原始训练集基础上学习分类器参数时使用一次，以后即进入增量学习过程。

在增量过程中，测试集 T 中的实例会被逐步纳入训练集，此时参数 θ_i 与 θ_{ji} 的计算过程与上述过程有所不同。假设当前选择 T 中的实例 $x_p'=(x_{p1}',x_{p2}',\cdots,x_{pm}')$ 加入训练集，并且它由当前分类器学习得到的类别标签是 c_p'。再设新加入的实例是增量过程加入的第一个实例，此时参数 θ_i' 的计算分下述两种情形：

$$\text{当 } c_p' \neq c_i \text{ 时，} \quad \theta_i' = P(C=c_i) = \frac{1+count(c_i)}{|C|+|D|+1} = \frac{\delta}{1+\delta}\theta_i \qquad (\text{令 } \delta = |C|+|D|)$$

$$\text{当 } c_p' = c_i \text{ 时，} \quad \theta_i' = P(C=c_i) = \frac{1+count(c_i)+1}{|C|+|D|+1} = \frac{\delta}{1+\delta}\theta_i + \frac{1}{1+\delta}$$

容易看出，以后每加入一个新的测试实例时参数 θ_i' 与其先验参数 θ_i 间存在上述同样的计算关系，由此得到：

$$\theta_i' = P(C=c_i) = \begin{cases} \dfrac{\delta}{1+\delta}\theta_i & c_p' \neq c_i \\ \dfrac{\delta}{1+\delta}\theta_i + \dfrac{1}{1+\delta} & c_p' = c_i \end{cases} \tag{4.6}$$

从上式可以看出其充分利用先验知识及参数传递的特性。

对于参数 $\theta_{j\zeta|i}'$，有下述计算公式：

$$\theta_{j\zeta|i}' = P(X_j = X_{j\zeta} \mid C=c_i) = \begin{cases} \dfrac{\xi}{1+\xi}\theta_{j\zeta|i} & c_p' = c_i \ \ and \ \ X_{j\zeta} \neq x_{pj}' \\ \dfrac{\xi}{1+\xi}\theta_{j\zeta|i} + \dfrac{1}{1+\xi} & c_p' = c_i \ \ and \ \ X_{j\zeta} = x_{pj}' \\ \theta_{j\zeta|i} & c_p' \neq c_i \end{cases} \tag{4.7}$$

其中 $\xi = |X_j| + count(c_i)$。

4.3 基于训练集分类损失的序列学习算法 TSCL-SLA

TSCL-SLA 的基本思想是：在从测试集 T 中选择合适的实例进入 D 时，应考虑到 T 中数据完备性的差别。有的实例属性相当充分地表现了其类别取向，它与训练集的内在联系紧密，很显然这样的实例应该尽早被选进训练集，以提供更充足的知识储备；另外一些实例表达力不强，一定程度上还扰乱正常的学习过程，降低分类精度，所以，这种实例被选择的时间应靠后。也就是说，在增量学习过程中，应该充分利用当前较完备测试实例的有价值信息。具体作法是通过运用训练实例进行反复检验以选择合适的测试实例进行学习。

TSCL-SLA 模型运用训练实例检验选择测试实例时要用到分类损失的思想，模型的特色在于衡量的是对训练实例的分类损失而非对测试实例的分类损失。这保证了模型能充分利用训练实例的先验知识，这尤其适合于原始训练集较大的情形。

4.3.1 算法基本思想

设训练集 D 与测试集 T 的并集用 S 表示，考虑为 S 中的所有实例 s_l 引入一个类条件概率参数 ρ_l，并且为 D 中的实例 s_l 引入一个估计损失权重系数 λ_l。对于 $x_l=(x_{l1},x_{l2},\cdots,x_{lm})\in T$，运用公式(4.5)找出 x_l 的类别标签 c_i 并运用公式(4.4)计算出 $\rho_l=P(C=c_i|x_l)$，而对于 $s_l=<x_l,c_r>=(x_{l1},x_{l2},\cdots,x_{lm},c_r)\in D$，计算 $\rho_l=P(C=c_r|x_l)$。其中 c_r 是实例 x_l 的已确定的类别标签。

实际上，引入的概率参数 ρ_l 可被理解成实例属性在当前情况下对它所属类别的支持度。在增量学习过程中，T 中实例会被逐渐加入到训练集 D 中从而对训练集产生影响。例如在一次学习过程中，假设某个实例 $x_p'\in T$ 通过学习获得了类别标签 c_p' 得到 $D'=D+\{<x_p',c_p'>\}$。由于 x_p' 的类别标签是学习估计出来的，有一定的不确定性，所以在新的数据集 D'中，D 的原有实例对其所属类别的支持度会受到新成员 x_p' 的影响。很显然，如果新成员 x_p' 比较完备（即与训练实例拟合性较好），它应该与 D 中原有的实例有很好的相容性，D 中实例的类别支持度前后不会有大的变化。相应地，如果 x_p' 完备性不好，在一定程度上是干扰因素，那么在其被加入 D 后，D 中原有实例的类别支持度会明显受其影响，先后支持度的差别就会比较大。我们当然要优先选择前者进行学习，并充实到训练集中来。

为此，算法还为 D 中实例 s_l 引入一个估计损失权重系数 λ_l，λ_l 反映了 D 中实例对新加入进来的实例的敏感程度。假设在学习新的测试实例前 s_l 的类条件概率保存在 ρ_l 中，再假设某个实例 $x_p'\in T$ 通过学习获得了类别标签 c_p'，得到 $D'=D+\{<x_p',c_p'>\}$，在 D'中运用公式(4.4)计算出 s_l 新的类条件概率用 ρ_l' 表示。定义 $\lambda_l=\rho_l\exp(\rho_l'-\rho_l)=\rho_l\exp(\Delta\rho_l)$。$s_l$ 的估计损失计算公式是：$Loss_l=\lambda_l|\Delta\rho_l|=|\Delta\rho_l|\rho_l\exp(\Delta\rho_l)$，其中 $|\Delta\rho_l|$ 是绝对损失。算法从 T 中寻找使得训练集 D 中所有实例的估计损失和

$$LossSum=\sum_{s_l\in D}Loss_l$$

最小的实例进入 D，逐步减少测试集 T 实例数，直到 T 为空。

4.3.2 算法描述

输入：训练集 $D=\{s_1,s_2,\cdots s_\eta\}$，$s_i=<x_i,c_i^0>$，其中 c_i^0 是 x_i 的已知类别；测试集

$T=\{x_1',x_2',\cdots x_\mu'\}$，依前述约定可知 $\eta+\mu=n=|S|$

输出：分类器 C

实现过程：

1 在训练集 D 的基础上学习分类器 C，求出 D 中所有实例 s_l 的类条件概率 ρ_l；

2 如果 $T=\phi$，算法结束，返回分类器 C，否则继续；

3 令 LeastLoss＝某个足够大的数；对 T 中每个实例 x_p'，重复以下过程：

3.1) 利用公式(4.5),学习获得 x_p'的类别标签 c_p'；

3.2) 在 $D'=D+\{<x_p',c_p'>\}$的基础上，重新计算 D 中所有实例 s_l 的类条件概率 ρ_l'　及其估计损失 $Loss_l$，从而得到估计损失和 $LossSum=\sum_{s_l\in D} Loss_l$；

3.3) 如果 LeastLoss>LossSum，则令 LeastLoss=LossSum，$x_{selected}=x_p', c_{selected}=c_p'$，并暂存 D 中所有训练实例 s_l 在 3.2）中计算的 ρ_l'；

4 更新分类器参数：$D=D+\{<x_{selected},c_{selected}>\}$，$T=T-\{x_{selected}\}$，把 3.3）中暂存的所有原训练实例 s_l 的 ρ_l'赋给相应的 ρ_l，在新的训练集中计算并保存新加入的 $s_{selected}=<x_{selected},c_{selected}>$的类条件概率 $\rho_{selected}$，转向 2)。

4.4 算法原理分析

在 TSCL-SLA 学习算法中，需要为每个训练实例 s_l 保存以前的类条件概率 ρ_l，以与后来的概率值结合计算总估计损失，这是该算法额外的空间需求。在时间性能上，为了提高精度，在选择每个测试实例时，都要重新计算训练实例的类条件概率和估计损失，其计算训练实例估计损失的基本操作共进行 $\sum_{i=1}^{|T|} i(|S|-i)$ 次，时间开销远远小于 $O(|S|^3)$，因此该算法是完全可行的。

我们很容易分析出该算法的学习功能：

首先，算法充分地结合了先验知识，实际上正是在结合先验知识的基础上来计算估计损失的。

其次，训练实例估计损失权重系数 $\lambda_l=\rho_l\exp(\Delta\rho_l)$ 恰如其分地反应了新实例对训练实例的影响作用，对于 D 中实例 s_l 而言，ρ_l 越大，说明该实例比较充分地反映了实例集的整体特点，其对外来实例所产生影响的敏感程度应该相对较高，所以我们把 ρ_l 作为 λ_l 的一个因子。另外，ρ_l' 与 ρ_l 之间的关系必是下述三种情况之一：

(1) $\rho_l'>\rho_l$，此时 $\Delta\rho_l>0$，说明新实例的加入强化了实例 s_l 的表达力，$|\Delta\rho_l|$ 越大，s_l 对新实例的敏感程度应该越小。

(2) $\rho_l'=\rho_l$，即 $\Delta\rho_l=0$，此时新实例对 s_l 的表达力没什么额外影响。

(3) $\rho_l'<\rho_l$，此时 $\Delta\rho_l<0$，说明新实例弱化了实例 s_l 的表达力，$|\Delta\rho_l|$越大，s_l 对新实例的敏感程度就越大。

为了反映上述事实，在 λ_l 定义式引入了因子 $\exp(\Delta\rho_l)$。由上述分析可知，

引入估计损失权重系数以后，在综合各种因素的基础上，该算法保证了相对可靠的测试实例被优先选择，从而优化分类器性能，提高分类精度。

最后，贝叶斯学习方法建立在属性间条件独立假设前提下。属性独立性假设往往不能满足，上述算法中反复用训练实例检验的过程弱化了属性间独立性假设的负面影响。

4.5 小结

贝叶斯方法的统计特性和处理先验知识的能力使得贝叶斯分类器在增量分类领域有明显的性能优势。本文针对增量分类器学习过程提出了一种改进的综合考虑类别估计损失的学习算法，在增量学习过程中，每次选择合适的测试实例时，用当前训练集的实例进行损失检验，努力让与当前训练集拟合度较好的测试实例被优先学习。这种算法能有效地结合先验知识并在一定程度上解决先验信息传递的问题。

第五章 总结与展望

本文比较系统地介绍了数据挖掘领域中基于贝叶斯理论的分类技术。分类问题是数据挖掘中一个非常重要的分支，图像识别、声音识别等许多科学问题最终都归结为分类。

分类以数据挖掘为学术背景，所以本文绪论重点介绍了数据挖掘的一般概念并在此基础上引出文章的主题。

贝叶斯分类技术以贝叶斯定理、最大后验假设、贝叶斯网络理论以及信息学理论为基础。文章在给出上述基础理论后介绍了当前贝叶斯分类领域的一些研究成果。本文将所有贝叶斯分类模型归结为三大类：朴素贝叶斯分类模型、贝叶斯网络分类模型及增量贝叶斯分类模型。

朴素贝叶斯分类模型由于其简单和易于实现受到人们的普遍青睐。目前对它的研究主要集中在探讨它的独立性条件和如何改善性能方面。Pedro Domingos[27]指出决定朴素贝叶斯分类性能的是条件后验概率的顺序而不是具体的后验概率值，因此在有些领域，属性之间虽然有很强的依赖关系，但仍能表现出很好的性能。

朴素贝叶斯分类模型具有算法稳定的特点[48][49]，对数据没有偏好性的要求，所以它在许多时候作为其它分类模型的参照模型出现。其独立性假设在实际运用中对分类有先天的消极作用，为此出现了以提升的朴素贝叶斯分类模型、半朴素贝叶斯分类模型为典型的变化模型。

贝叶斯网络分类技术随着贝叶斯网络技术的发展而发展，很多情况下，贝叶斯网络分类器的性能优于朴素贝叶斯分类器。其不足在于网络结构及参数学习的时间开销相当大，并且，贝叶斯网络分类器的分类性能与具体使用的网络结构评分函数关系密切。

TAN 及其变化模型是比较简单的贝叶斯网络分类模型。这些模型在一定程度上是朴素贝叶斯分类模型与贝叶斯网络分类模型之间的折中形式。

增量贝叶斯分类是一种动态分类过程，这里的动态贝叶斯分类模型与动态贝叶斯网络模型是两个不同的概念。

5.1 本文的创新工作

本文的创新性工作表现在如下两个方面：

其一，针对目前算法存在的问题：对每一个可选模型都要运用似然函数进行检验，占据了大量的计算时间，提出了新的半朴素贝叶斯分类模型属性组合

学习算法 CMI-BSNBC，本文算法不用此检验过程也能达到与其类似的目的；LP 等算法容易将本来关联度不大的基本属性组合到一起，本算法通过引入条件互信息阈值克服了这一弊端。

其二，在增量贝叶斯分类领域，基于充分利用训练集先验知识的考虑提出了新的主动学习策略 TSCL-SLA。本文的策略在先验知识的传递方面进行了有益的尝试。

5.2 工作展望

朴素贝叶斯分类模型的分类思想日臻成熟。现在研究的重点是探讨其最优性条件，目前的研究成果是[34]：在属性间完全独立及属性间存在完全函数依赖关系两种极端情况下朴素贝叶斯分类模型的性能最优。然而这一结论缺乏可操作性，关键在于真实概率难以估计，进一步探讨朴素贝叶斯分类器最优性可操作性的条件是一件必要而有意义的工作。

提升贝叶斯分类器以及选择贝叶斯分类器从不同的角度研究了如何在保证简单的模型结构基础上提高分类精度的问题。该方向进一步的工作包括：进一步探讨有效的模型结构学习算法和如何自动地选择模型结构以适应特定领域的需要。

以 TAN 为代表的贝叶斯网络分类模型通过增加扩展弧直接消除特征属性间独立性限制以改善分类性能。问题的关键是：选用什么样的模型结构分类效果最优？是否越复杂的模型结构，分类精度越高？一般来说答案是否定的。因为复杂的模型结构的学习带来了较高的方差，学习的复杂性也较高，并且结构复杂的分类器存在过分拟和的问题。

对增量贝叶斯分类模型的研究，集中在如何确定候选样本的主动选择策略。主动策略主要基于两种概念：确定性及分类损失。本文提出的策略的基本思想是：在选择测试实例时，用当前训练实例的分类损失进行检验以充分利用训练实例的先验知识。本文的算法倾向于训练集较大的情形。在此领域进一步的研究方向包括：主动学习中的误差传播机制、研究更加有效的抽样策略来选择样本及其如何利用尽可能少的训练样本来提高分类精度。

另外，联合分类器是分类领域的趋势。现在数据挖掘中分类思想种类繁多，各自在不同的领域有不俗的表现，并且还没有发现一个适合于所有数据特点的分类模型，所以联合不同的分类模型是必然选择。

这里的联合分为两类：同类分类模型内部的联合及异类分类模型间的外部联合。实际上，已经有很多人在内部联合上做了有益的工作，提出了一些联合分类模型。如 Kaizhu Huang, Irwin King, Michael R. Lyu[45] 在半朴素贝叶斯分类领域提出的有界半朴素贝叶斯分类器的有限联合分类器就是一个典型。此模型将在同一数据集上形成的半朴素分类器进行加权联合。实验表明了联合的可

行性。至于外部联合，由于不同的分类思想涉及的结构迥异，所以简单的加权联合是行不通的。可以考虑将分类过程分为不同的时序段或不同的块，在各个阶段或块应用不同的模型进行分类。这又将会是一件十分有意义的工作！

参 考 文 献

[1] Heikki Mannila.*Methods and problems in data mining*.Proceedings of International Conference on Database Theory,Delphi,Greece,1997

[2] Fayyad,Piatetsky-Shapiro,Smyth.*From Data Mining to Knowledge Discovery*.KDD96

[3] Jiawei Han and Micheline Kamber. *DATA MINING Concepts and Techniques*, Higher Education Press ,Morgan Kaufmann Publishers,2001

[4] 林士敏，田凤占，陆玉昌 《贝叶斯网络的建造及其在数据挖掘中的应用》 清华大学学报(自然科学版) 2001,41(1): 49-52

[5] 尹朝庆，尹皓 《人工智能与专家系统》 中国水利水电出版社，2002

[6] 林士敏，田凤占，陆玉昌《用于数据挖掘的贝叶斯分类器研究》 计算机科学 2000 27(10) 73-76

[7] Friedman,Geiger,Goldszmidt.*Bayesian Network Classifiers*,1997

[8] Zijian Zheng,Geoffrey I.Web,Kai Ming Ting.Lazy Bayesian Rules:*A Lazy Semi-Naïve Bayesian Learning Technique Competitive to Boosting Decision Trees*.Proceedings of ICML'99,Morgan Kaufmann

[9] Z.Pawlak,Jerzy Grzymala-Busse,Roman Slowinski,Wojciech Ziarko.*Rough Set Approach To Knowledge-Based Decision Support (1995)*.Communications of the ACM,38,11,88-95,1995

[10] 胡学钢 《从数据库中提取知识的模型研究》 合肥工业大学博士学位论文 合肥 2000, 5

[11] R.Wille,Restructuring lattice theory:An approach based on hierarchies on concepts,in Ordered Sets(I.Rival,ed.),445-470,Dordrecht-Boston:Reidel,1982

[12] H. Lu, R. Setiono and H. Liu, ``Towards Effective Classification Rule Extraction", DOOD'95 workshop on Integration of Knowledge Discovery in Databases with Deductive and Object-Oriented Databases, Dec. 1995, Singapore

[13] http://www.bioinfo.de/isb/gcb99/talks/zien/

[14] Jack Breese,Daphne Koller.*Tutorial on Bayesian Networks*.AAAP'97 tutorial

[15] Kevin Murphy. *A Brief Introduction to Graphical Models and Bayesian Networks* .2001

[16] 慕春棣，戴剑彬，叶俊 《用于数据挖掘的贝叶斯网络》 软件学报 2000，11(5):660-666

[17] 王玮，陈恩红，王煦法 《基于贝叶斯方法的知识发现》 小型微型计算机系统 2000,21(7): 703-705

[18] 林士敏，王双成，陆玉昌 《Bayesian 方法的学习机制和问题求解》 清华大学

学报(自然科学版) 2000,40(9):61-64

[19] Maurice Pagnucco. *Reasoning Under Uncertainty and Bayesian Networks*.COMP9414,2002

[20] David Maxwell,David Herkerman,Christopher Meek.*Technical Report:A Bayesian Approach to Learning Bayesian Networks with Local Structure*.March 1997

[21] 胡玉胜，涂序彦，崔晓瑜，程乾生 《基于贝叶斯网络的不确定性知识的推理方法》 计算机集成制造系统-CIMS 2001,7(12): 65-68

[22] 王辉 《用于决策支持的贝叶斯网络》 东北师大学报自然科学版 2001,33(4): 26-30

[23] 邢永康，沈一栋 《基于互信息和测度学习信度网结构》 重庆大学学报(自然科学版) 2001,24(1):78-83

[24] 汪荣贵，张佑生，王浩，姚宏亮 《改进学习贝叶斯网络结构的 MDL 准则》 计算机工程与应用 2002,(8):07-08

[25] C.E.SHANNON.*A Mathematical Theroy of Communication*.The System Technical Journal,Vol.27,pp.379-423,623-656,July,October,1948

[26] Mia K.Stern,Joseph E.Beck,and Beverly Park Woolf.*Naive Bayes Classifiers for User Modeling*

[27] Pedro Domingos,Michael Pazzzani.*On the Optimality of the Simple Bayesian Classifier under Zero-One Loss*.Machine Learning.29,103-130 (1997)

[28] Mehran Sahami.*Learning Limited Dependence Bayedian Classifiers*.Computer Science Department Stanford University,1997 [5] Lior Rokach,Oded Mainon.*Theory and Applications of Attribute Decomposition*. Department of Industrial Engineering

[29] Geoffrey I.Webb,Janice R.Boughton,Zhihai Wang.*Not so naïve Bayesian classification*.(2001)

[30] Kaizhu Huang,Irwin King,Michael R.Lyu.*Learning Maximum Likelihood Semi-Naïve Bayesian Network Classifier*.2002

[31] Kaizhu Huang,Irwin King,Michael R.Lyu.*Finite Mixture Model of Bounded Semi-Naïve Bayesian Networks Classifier*.ICANN/ICONIP 2003,LNCS 2714,115-122

[32] Nir Friedman,Moises GoldszMidt.*Building Classifiers using Bayesian Networks*

[33] Sadeep Yaramakala,Jyotishman Pathak.*Bayesian Network Classifiers:A Comparative study of Tree Augmented Networks and MDL based aprroach*.1998

[34] 王君圣，李敏强 《基于数据库信息构建贝叶斯网络的 GA 方法》 系统工程与电子技术 2000,22(9): 54-57

[35] R.Cattoni etc.*Bayesian Belief Networks:Introduction and Learning*.IRST-Technical Report 9803-05

[36] Macro Ramoni,Paola Sebastiani.*Learning Bayesian Networks from Incomplete*

Databases.Knowledge Media Institute,1997

[37] P.Larranaga,M.Poza,Y.Yurranmendi,R.H.Murga,C.M.H.Kuijpers.*Structure Learning of Bayesian Networks by Genetic Algorithm:A Performance Analysis of Control Parameters*

[38] 姚宏亮，王浩，胡学钢，汪荣贵 《基于遗传算法和 MDL 原则的贝叶斯网络结构优化算法》 南京大学学报 (自然科学版) 2001 (11)

[39] Wai Lam,Fahiem Bacchus.*Learning Bayesian Belief Networks:An approach based on the MDL Principle*.Computational Intelligence,Vol 10:4,1994

[40] 林士敏，田凤占，陆玉昌 《贝叶斯学习、贝叶斯网络与数据挖掘》 计算机科学 2000 27(10) 69-72

[41] 冀俊忠，刘椿年，沙志强 《贝叶斯网模型的学习、推理和应用》 计算机工程与应用 2003.5 24-28

[42] 张宏伟，田凤占，陆玉昌 《对一种贝叶斯网络学习算法的改进及试验分析》 计算机科学 2002 29(5) 97-100

[43] Qiang Lei,Xiao Tian-Yuan,Qiao Gui-Xiu.*An Improved Bayesian Networks Learning Algorithm*.Journal of Computer Research and Development.2002 39(10) 1221-1226

[44] 宫秀军，刘少辉，史忠植 《一种增量贝叶斯分类模型》 计算机学报 2002,25(6): 645-650

[45] 宫秀军，孙建平，史忠植 《主动贝叶斯网络分类器》 计算机研究与发展 2002 39(5):574-579

[46] Herber A Simon,Glenn Lea.*Problem solving and rule reduction,a unified view*.In: Knowledge and Cognition.Erbuam,1974

[47] P H Winston.*Learning structure description from examples*.The Psychology of Computer Vision.NewYork:Mc-Graw-Hill,1975

[48] I.Rish,J.L.Hellerstein,T.S.Jayram.*An analysis of naïve Bayes classifier on low-entropy distributions*

[49] Michael J.Pazzani.*Searching for Dependence in Bayesian Classifiers*.Department of Information and Computer Science University of California,Irvine

[50] *URL: http://www.ics.uci.edu/~mlearn/MLRepository.html*

附录 1：攻读硕士学位期间发表的论文

1. 姜卯生，王浩，姚宏亮 《朴素贝叶斯分类器增量学习序列算法研究》《计算机工程与应用》，2004.7

2. 王浩，姜卯生，姚宏亮 《条件互信息度量 BSNBC 分类学习算法》(已投稿)

附录 2：CMI-BSNBC 分类模型部分代码

1 类说明

// (MFC)除了主程序类,视图类,文件类,主窗口类等内部产生的类之外,这里列出DSN,Table, CborderThres, CCvk以及CCMIBSNBC五个添加类的说明：

1) 选择数据源窗口类:DSN

```
class DSN : public CDialog
{
public:
    void GetDSNsAndDrivers();                          // 从系统获取ODBC数据源
    CComboBox  m_dsn;                                  // 用于对数据源列表控件进行控制
    CString   m_dsnvalue;                              // 用于保存用户选择的数据源名称
    CString sDsn[SQL_MAX_DSN_LENGTH];                  // 数据源名称数组
    CString sDescription[SQL_MAX_DSN_LENGTH];          // 数据源基本信息数组
    int DsnNum;                                        // 用于保存数据源个数
    CString m_uidvalue;                                // 用于保存用户输入的用户名
    CString m_pwdvalue;                                // 用于保存用户的密码
        afx_msg void OnCbnSelchangeCobdsn();           // 从数据源列表选择某一项时产
生的事件函数
    CString m_driver;                                  // 用于保存数据库驱动名称
protected:
    virtual BOOL OnInitDialog();                       // 对话框初始化函数
    virtual void OnOK();                               // 对话框"确定"按钮的事件函数
    virtual void OnCancel();                           // 对话框"取消"按钮的事件函数
};
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

2) 选择数据表窗口类:Table

```
class Table : public CDialog
{
public:
    CString sTable[20];                                // 数据表名称数组，一个数据源可能对应多个
数据表表
    CString sDsn;                                      // 数据源名称
    int TableNum;                                      // 数据表个数
```

```
    CListBoxm_table;                          // 用于控制数据表列表控件
    CString  m_tablevalue;                    // 用于保存用户选择的数据表名称
protected:
    void GetTable(CString dsn);               // 由数据源名称从系统获取对应的数据表
    virtual BOOL OnInitDialog();              // 窗口初始化
    virtual void OnOK();                      //“确定”按钮事件函数
    virtual void OnCancel();                  //“取消”按钮事件函数
};
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

3) 组合参数输入窗口类:CborderThres

```
class CBorderThres : public CDialog
{
public:
    int m_border;                             // 用于保存用户输入的组合属性所含基本属性个数的界限值
    double m_threshold;                       // 用于保存用户输入的条件互信息阈值
};
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

4) 交叉验证参数输入窗口类: CCvk

```
class CCvk : public CDialog
{
public:
    int m_cv_k;                               // 交叉验证参数
};
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

5) CMIBSNBC类

```
//下列AttrValList, ComAttr, ConditionalProNode, ConditionalPT, ClassPT, MutualConditionalProNode, MutualConditionalPT结构在CCMIBSNBC中被使用.
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
struct AttrValList                            // 属性可能取值列表,其作用有两个:用于保存基本信息中每个基本属性所有可能取值;另外,还用于保存一个组合属性的一种具体取值,此时表示为一个取值向量
{
    int Num;
    CString ComNode[ATTR_VALUE_NUM];
};
```

```
struct ComAttr                          // 组合属性节点
{
    int Num;                            // 组合属性实际包含基本属性的个数
    CString ComNode[COM_ATTR_NUM];// 基本属性名称列表
};
struct ConditionalProNode
{
    double Pro;                         // 概率值
    AttrValList ComValue;               //组合属性具体取值
    CString ClassValue;                 // 类别具体取值
    ConditionalProNode *Next;           // 指针指向下一个节点
};
struct ConditionalPT
{
    ComAttr ComNode;                    // 组合属性名称
    ConditionalProNode *Head;           // 指向ConditionalProNode 型结构的指针
    ConditionalPT *Next;                // 指向下一个ConditionalPT 型节点的指针
};
struct ClassPT                          //类概率节点
{
    CString ClassValue;                 // 类别属性的具体取值
    double Pro;                         // 对应的概率值
    int Num;                            // 具体取值在数据表中出现的次数
};
struct MutualConditionalProNode
{
    double Pro;                         // 互联合概率值
    AttrValList ComValueX;              // 组合属性具体取值
    AttrValList ComValueY;
    CString ClassValue;
    MutualConditionalProNode *Next;
};
struct MutualConditionalPT
{
    double MutualInfo;                  // 互信息值
    ComAttr ComNodeX;
```

```
    ComAttr ComNodeY;
    MutualConditionalProNode *Head;
    MutualConditionalPT *Next;
};
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//核心类CCMIBSNBC:
class CCMIBSNBC
{
    CCMIBSNBC(CString sDsn, CString sPid, CString sPwd, CString sTable, int Border, double
Threshold);
private:
    CString DsnName;                                    // 数据源名称
    CString Uid;                                        // 用户ID
    CString Pwd;                                        // 用户密码
    CString TableName;                                  // 数据表名称
    double loge2;                                       // 2的常用对数值
    int AttrNumber;                                     // 实际基本属性个数
    CString BaseAttrNameList[CMI_ATTR_NUM]; // 基本属性列表
    AttrValList AttrValues[CMI_ATTR_NUM];       // 基本属性可能取值列表
    CString ClassName;                                  // 类别属性名称
    int ClassValNum;                                    // 类列属性可能取值个数
    CString ClassValList[ATTR_VALUE_NUM];       // 类别属性可能取值列表
    ClassPT ClassPro[ATTR_VALUE_NUM];           // 类别概率表
    int Low,High;                                       // 下标在Low与High之间的数据是测试数据
public:
    int Border_K;                                       // 组合属性大小界限值
    double Thre;                                        // 条件互信息阈值
    int TotalRecordNum;                                 // 记录总数
    int MidComAttrNum;                                  // 过渡组合属性个数
    ComAttr MidList[CMI_ATTR_NUM];              // 过渡组合属性列表
    int ComAttrNum;                                     // 最终形成的目的组合属性个数
    ComAttr ComAttrList[CMI_ATTR_NUM];          // 目的组合属性列表
    AttrValList SQLValueList[CMI_ATTR_NUM]; // 组合属性的具体取值列表
    CString TrueClassValue;                             // 类别取值,在测试过程中使用
    ConditionalPT *CPTPtr;                              // 条件概率表指针
    MutualConditionalPT *MCPTPtr,*MaxPtr;       // 互信息表指针
```

```
//成员函数
private:
    // 返回基本属性NodeName可能取值的个数
    int ValueNumberOf(CString NodeName);
// 合并两个组合属性
    ComAttr CombineTwoNodes(ComAttr ComNodeX, ComAttr ComNodeY);
    // 返回指定基本属性在属性列表中的位置
    int IndexOf(CString BasicAttr);
    // 比较两个组合属性是否相同
    bool IsEqual(ComAttr ComNodeX, ComAttr ComNodeY);
    // 比较组合属性具体取值是否相等
    bool IsEqual(AttrValList ComValueX, AttrValList ComValueY);
    // 查询ComNode节点是否在CPTPtr所指向的条件概率列表中
    bool IsInConditionalPT(ComAttr ComNode,ConditionalPT *CPTPtr);
    // 查询ComNodeX,ComNodeY是否存在于MCPTPtr所指向的互信息列表中
    bool IsInMutualPT(ComAttr ComNodeX, ComAttr ComNodeY,MutualConditionalPT *MCPTPtr);
    // 由给定的类别值c计算对应取值的概率P(c)
    double GetProOfClassValue(CString ClassValue);
    // 计算当前数据集除下标Low至High之外的记录集中组合属性ComNode取ComValue相对于
类别属性取ClassValue 时的条件概率,当Low与High都等于0时,计算的范围是当前所有的记录构成
的记录集
    double ComputeCp(ComAttr ComNode, AttrValList ComValue, CString ClassValue, int Low, int
High);
    // 返回联合条件概率值 P(x,y|c)
    double ComputeCp(ComAttr ComNodeX, AttrValList ComValueX, ComAttr ComNodeY,
AttrValList ComValueY, CString ClassValue, int Low, int High);
    // 计算当前数据表中满足查询条件strSQl的记录的个数
    int SumConditionCount(CString strSQL);
    // 初始化过渡组合属性列表为基本属性列表
    void InitialMidList(void);
public:
    // 获取用户所选数据表基本信息,主要包括:记录总数,各个基本属性的基本信息等
    void GetBasicInformation(void);
    // 计算类别具体取值的绝对概率,对Low与High的说明同前
    void CalClassPro(int Low, int High);
    // 计算两个组合属性相对于类别的所有可能联合条件概率,放入全局指针变量MCPTPtr指向
```

```
的链表,并计算互信息值
    void CalAllProBetweenTwoComAttrs(ComAttr ComNodeX, ComAttr ComNodeY, int Low, int
High);
    // 计算组合属性ComNode相对于类别的所有可能条件概率值,放入全局指针变量CPTPtr指向
的链表
    void CalAllConditionPro(ComAttr ComNode, int Low, int High);
    // 获取给定组合属性取值与类别取值时的条件概率值
    double GetConditionPro(ComAttr ComNode, AttrValList ComValue, CString ClassValue);
    // 构建条件概率表及联合条件概率信息表
    void BuildCPT(int Low,int High);
    // 从CPTPtr链表中删除含有组合属性节点ComNode的项
    void DeleteNodeFromCPTPtr(ComAttr ComNode, ConditionalPT **CPTPtrAdd);
    // 从MCPTPtr链表中删除含有组合属性节点ComNodeX,ComNodeY的项
    void DeleteNodeFromMCPTPtr(ComAttr ComNodeX,ComAttr ComNodeY,MutualConditionalPT
**MCPTPtrAdd);
    // 清空CPTPtr指针
    void EmptyCPTPtr(ConditionalPT **CPTPtrAdd);
    // 清空MCPTPtr指针
    void EmptyMCPTPtr(MutualConditionalPT **MCPTPtrAdd);
    // 从MCPTPtr列表中找出互信息值最大的节点,用MaxPtr全局指针变量指向该节点
    void FindTheMaxInfo(void);
    // 组合属性
    void Combining(void);
    // 从过渡组合属性列表中删除指定的组合属性
    void DeleteNodeFromList(ComAttr Node);
    // 得到 ID=RecordIndex 的测试记录或待分类记录与属性组合成功后形成的组合属性相对应
的组合属性取值,放在全局数组SQLValueList中
    void GetAttrValue(int RecordIndex);
    // 给 ID=RecordIndex 的记录进行分类,返回类别标签
    CString * Classfy(int RecordIndex);
    // 分类模型测试,返回精确度
    double TestBody(int Low, int High);
};
```

2 CMIBSNBC类中成员函数Combining(void)的实现过程,略去其它实现过程:

```
void CCMIBSNBC::Combining(void)
{
    ConditionalPT *CPTTemp=NULL,*CPTForward=NULL;     // 条件概率节点指针
    MutualConditionalPT *Temp=NULL,*Forward=NULL;      // 互信息节点指针
    ComAttr TempNodeX;                                 // 临时组合属性X
    ComAttr TempNodeY;                                 // 临时组合属性Y
    ComAttr TempNode;                                  //临时保存由X节点与Y节点合并而成的节点
    bool Flag=false;
    ComAttrNum=0;                                      //目的组合属性列表初始化为空
    while(MidComAttrNum>1)
    {
        //所有基本属性对的互信息值事先已经放在全局指针变量MCPTPtr所指的列表中
        //下述函数从MCPPtr链表中找到互信息值最大的节点,由全局指针变量MaxPtr指向
        FindTheMaxInfo();
        //得到两个备份组合属性节点
        TempNodeX.Num=MaxPtr->ComNodeX.Num;
        for(int i=0;i<TempNodeX.Num;i++)
            TempNodeX.ComNode[i]=MaxPtr->ComNodeX.ComNode[i];

        TempNodeY.Num=MaxPtr->ComNodeY.Num;
        for(int i=0;i<TempNodeY.Num;i++)
            TempNodeY.ComNode[i]=MaxPtr->ComNodeY.ComNode[i];
        // 最大的互信息值小于给定的阈值时完成下述操作
        if(MaxPtr->MutualInfo<Thre)
        {
            for(int i=0;i<MidComAttrNum;i++)
            {
                ComAttrList[ComAttrNum].Num=MidList[i].Num;
                for(int j=0;j<MidList[i].Num;j++)
                    ComAttrList[ComAttrNum].ComNode[j]=MidList[i].ComNode[j];
                ComAttrNum++;
            }
            return;
        }
        //最大互信息值大于等于给定阈值且组合属性的基值等于界限值的情形
```

```
if(TempNodeX.Num+TempNodeY.Num= =Border_K)
{

    TempNode=CombineTwoNodes(TempNodeX,TempNodeY);
    ComAttrList[ComAttrNum].Num=TempNode.Num;
    for(int i=0;i<TempNode.Num;i++)
        ComAttrList[ComAttrNum].ComNode[i]=TempNode.ComNode[i];
    ComAttrNum++;
    DeleteNodeFromList(TempNodeX);          //从过渡组合属性列表中删除
    DeleteNodeFromList(TempNodeY);
    Temp=MCPTPtr;
    while(Temp!=NULL)
    if(IsEqual(TempNodeX,Temp->ComNodeX)||IsEqual(TempNodeX,Temp->ComNodeY)
    ||IsEqual(TempNodeY,Temp->ComNodeX)||IsEqual(TempNodeY,Temp->ComNodeY))
    {
        Forward=Temp->Next;
        DeleteNodeFromMCPTPtr(Temp->ComNodeX,Temp->ComNodeY,&MCPTPtr);
        Temp=Forward;
    }
    else Temp=Temp->Next;
    CPTTemp=CPTPtr;
    while(CPTTemp!=NULL)
    if(IsEqual(CPTTemp->ComNode,TempNodeX)||
            IsEqual(CPTTemp->ComNode,TempNodeY))
    {
        CPTForward=CPTTemp->Next;
        DeleteNodeFromCPTPtr(CPTTemp->ComNode,&CPTPtr);
        CPTTemp=CPTForward;
    }
    else CPTTemp=CPTTemp->Next;
}
else
//最大互信息值大于等于给定阈值且两组合属性的基值之和小于等于界限值的情形
{
    DeleteNodeFromList(TempNodeX);
    DeleteNodeFromList(TempNodeY);
```

```
TempNode=CombineTwoNodes(TempNodeX,TempNodeY); //产生新的临时组合节点
Temp=MCPTPtr;
while(Temp!=NULL)
if(IsEqual(TempNodeX,Temp->ComNodeX)||IsEqual(TempNodeX,Temp->ComNodeY)
||IsEqual(TempNodeY,Temp->ComNodeX)||IsEqual(TempNodeY,Temp->ComNodeY))
{
    Forward=Temp->Next;
    DeleteNodeFromMCPTPtr(Temp->ComNodeX,Temp->ComNodeY,&MCPTPtr);
        Temp=Forward;
}
else Temp=Temp->Next;
CPTTemp=CPTPtr;
while(CPTTemp!=NULL)
if(IsEqual(CPTTemp->ComNode,TempNodeX)||
    IsEqual(CPTTemp->ComNode,TempNodeY))
{
    CPTForward=CPTTemp->Next;
    DeleteNodeFromCPTPtr(CPTTemp->ComNode,&CPTPtr);
    CPTTemp=CPTForward;
}
else CPTTemp=CPTTemp->Next;
Flag=false;
for(int i=0;i<MidComAttrNum;i++)
    if(MidList[i].Num+TempNode.Num<=Border_K)
    {
        Flag=true;
        break;
    }
if(!Flag)
{
    ComAttrList[ComAttrNum].Num=TempNode.Num;
    for(int j=0;j<TempNode.Num;j++)
    ComAttrList[ComAttrNum].ComNode[j]=TempNode.ComNode[j];
    ComAttrNum++;
}
else
```

```
{
    MidList[MidComAttrNum].Num=TempNode.Num;
    for(int i=0;i<TempNode.Num;i++)
        MidList[MidComAttrNum].ComNode[i]=TempNode.ComNode[i];
    MidComAttrNum++;
    CalAllConditionPro(TempNode,0,0);

    for(int j=0;j<MidComAttrNum;j++)
        if((MidList[j].Num+TempNode.Num<=Border_K)&&
            !IsEqual(MidList[j],TempNode))
            CalAllProBetweenTwoComAttrs(MidList[j],TempNode,0,0);
    }
  }
}
if(MidComAttrNum= =1)
{
    ComAttrList[ComAttrNum].Num=MidList[0].Num;
    for(int j=0;j<MidList[0].Num;j++)
        ComAttrList[ComAttrNum].ComNode[j]=MidList[0].ComNode[j];
    ComAttrNum++;
  }
}
```