利用数据库技术实现的可扩展的分类算法*

刘红岩[1], 陆宏钧[2], 陈 剑[1]

[1](清华大学 经济管理学院,北京 100084);
[2](香港科技大学 计算机系,香港)
E-mail: {liuhy,chenj}@em.tsinghua.edu.cn; luhj@cs.ust.hk
http://www.cs.ust.hk/~luhj

摘要: 重点研究将数据挖掘中的分类技术与数据库技术紧密结合的高效的可扩展的分类算法.提出一种基于分组记数技术构造分类器的方法,利用数据库系统的结构化查询语言来实现主要计算任务.为了提高算法的执行效率,还提出了优化策略和冗余规则的剪裁策略,并将分类规则的发现过程与相关属性的选择方法有机地结合在一起.使用这些方法和策略,分类算法能够从大规模数据集中快速地发现一组简洁的规则.除了具有与现有分类算法相当的准确度和较高的执行效率以外,该分类算法还具有良好的基于训练集元组个数和属性个数两方面的可扩展性和易于实现的特点.

关 键 词: 数据挖掘;分类;关系数据库管理系统;结构化查询语言
中图法分类号: TP181 文献标识码: A

数据挖掘(data mining)又称数据库中的知识发现,是一个从大规模数据库的数据中抽取有效的、隐含的、以前未知的、有潜在使用价值的有用信息的过程.它是当今众多学科领域特别是数据库领域最前沿的研究课题之一.在数据挖掘算法中,分类(classification)是具有广泛应用领域的最重要的问题之一.它是发现属于同一类的数据对象的共同特性的过程,其目的是通过分析训练数据集的特点构造一个准确的分类器,该分类器可用于对未知类别的样本进行类别的判断.分类问题虽已在其他领域进行了很多的研究[1],但面对新出现的海量数据,如何快速而准确地发现隐藏于其中的主要分类规则,仍是一个值得研究的问题,为此,很多以前提出的算法被重新进行研究[2~4].特别是如何将分类算法与数据库技术结合起来,更是现今该领域研究的关键问题之一.Agrawal 等人曾尝试从系统的角度将数据挖掘中关联规则的挖掘算法与关系数据库系统紧密地结合起来[5].他们采用的方法是将完成关联规则挖掘的部分计算任务以用户定义的函数(user-defined function,简称 UDF)的方式交给关系数据库系统完成.Wang Min 等人也提出了用 UDF 实现决策树分类算法[3].由于 UDF 并不是关系数据库系统本身提供的函数,其功能仍然由用户编程实现,没有用到数据库的查询和处理功能,因此性能的提高受到限制.另外,随着数据挖掘技术的发展,一些新的分类算法也被提出来,如基于关联规则的算法[4].这种算法的分类准确度虽然有时较高,但是其多遍顺序扫描数据集的方法限制了算法的执行速度.

基于上述分析,本文提出一种名为 GAC-RDB(grouping and counting-relational database)的算法,采用分组记数的方法构造可扩展性良好的分类器.在该算法中,用于构造分类器的数据集、训练集(training set)由若干带有类别的元组构成.这些元组都由共同的 n 个属性描述,设共有 m 个类别,则每个元组可以描述为$(a_1,a_2,...,a_n,c_k)$,a_i 是属性 A_i 的一个取值,$A_i \in \{A_1,A_2,...,A_n\}$,$c_k \in \{c_1,c_2,...,c_m\}$是 m 个类别之一.假设所有属性的取值都是离散化的,对

* 收稿日期: 2000-07-31; 修改日期: 2000-11-23
基金项目: 国家重点基础研究发展规划 973 资助项目(G1998030414);清华大学 985 基础研究基金资助项目

作者简介: 刘红岩(1968－),女,山东烟台人,博士,讲师,主要研究领域为数据库,数据仓库与数据挖掘,信息系统与商务智能;陆宏钧(1945－),男,上海人,博士,教授,主要研究领域为数据库,数据仓库,数据挖掘;陈剑(1962－),男,福建连江人,博士,教授,主要研究领域为复杂系统建模与决策,电子商务,供应链管理.

于连续型取值的属性可以使用现有的离散化方法将其离散化.该算法的输出由若干条分类规则构成,每个规则由 n+3 个属性描述,分别为(A_1,A_2,...,A_n,*Class*,*Sup*,*Conf*).形式为(a_{1i},a_{2i},...,a_{ni},c_i,sup_i,$conf_i$)的每个规则的实际含义为

$$\text{if } (A_1=a_{1i}) \text{ and } (A_2=a_{2i}) \text{ and } \dots \text{ and } (A_n=a_{ni}) \text{ then } class=c_i \quad (sup_i,conf_i).$$

其中 sup_i和 $conf_i$是用于描述该规则的两个变量——支持度(support)和置信度(confidence).支持度用于衡量一个规则在统计意义上的普遍性,而置信度则是一个条件概率:$P(class=c_i|(A_1=a_{1i})\wedge(A_2=a_{2i})\wedge\dots\wedge(A_n=a_{ni}))$.

本文提出的构建分类器的算法,巧妙地利用关系数据库管理系统提供的聚集计算功能,将原来需编程实现的复杂的绝大部分操作用 SQL 语句实现,完成各种类别分布的统计工作,初步形成候选分类规则集,然后利用有效的策略剪裁掉冗余和无效的规则,形成包含较少个数规则且正确的分类规则集.

1 基于分组记数的分类方法

1.1 基本概念

定义 **1**. 一个具有 n 个属性 A_1,A_2,...,A_n的数据集 D 的决策表(decision table)定义为关系数据库中的一个表,其关系模式为 $R(A_1,A_2,...,A_n,Class,Sup,Conf)$.该表中的每一行 $R_i=(a_{1i},a_{2i},...,a_{ni},c_i,sup_i,conf_i)$代表一条分类规则,其中 $a_{ij}(1\le j\le n)$可以是属性 A_i 的域 $DOM(A_i)$中的一个值或者是值“*ANY*”(代表相应属性的任一个取值),$c_i\in\{c_1,c_2,...,c_m\}$,$0<sup_i\le 1$,$0<conf_i\le 1$.每行代表的规则的含义为:if $(A_1=a_1)$ and $(A_2=a_2)$ and … and $(A_n=a_n)$ then $class=c_i$,置信度为 $conf_i$,支持度为 sup_i,$a_j\ne ANY$,$1\le j\le n$.

定义 **2**. 给定一个训练集 $D(A_1,A_2,...,A_n,Class)$,其候选决策表定义为关系模式为 $S(A_1,A_2,...,A_n,Class,c_count)$的表,它包含了所有可能构成决策表的行,且满足如下要求:

若候选决策表中存在一行$(a_1,a_2,...,a_n,c_j,count_j)$,则与该行分组属性(不包括类别属性)取值相同的其他所有行也必须同时存在于该候选决策表中.

定义 **3**. 给定一个元组 $t=(a_1,a_2,...,a_n,c_k)$和一个规则 $R_i=(a_{1i},a_{2i},...,a_{ni},c_i,conf_i)$,若对于该元组的任一分量 $a_j(1\le j\le n)$,$a_{ji}=ANY$ 或者 $a_j=a_{ji}$,则称 t 与 R_i 匹配.若 t 与规则 R_i 匹配且 $c_k=c_i$,则我们称 t 被规则 R_i 覆盖,或 R_i 覆盖 t.

定义 **4**. 给定规则 $R_i=(a_{1i},a_{2i},...,a_{ni},c_i,conf_i)$和规则 $R_j=(a_{1j},a_{2j},...,a_{nj},c_j,conf_j)$,若被规则 R_j 覆盖的元组也被规则 R_j 覆盖,则我们称 R_j 是冗余的.

定理 1. 给定两个规则 $R_i=(a_{1i},a_{2i},...,a_{ni},c_i,conf_i)$和 $R_j=(a_{1j},a_{2j},...,a_{nj},c_j,conf_j)$,若满足:(1) $c_i=c_j$;(2) 若 $a_{li}\ne ANY$,则 $a_{lj}=a_{li}$,$1\le l\le n$ 这两个条件,则 R_j 是冗余的.

证明:设元组$(a_{1k},a_{2k},...,a_{nk},c_k)$被规则 R_j 覆盖,根据定义,则对于每一个满足 $a_{lj}\ne ANY$ 的属性取值,都满足 $a_{lk}=a_{lj}$,并且 $c_{k=}c_j$.下面证明该元组必被规则 R_i 覆盖.

因为 $c_i=c_j$,,所以 $c_i=c_j=c_{k.}$因为若 $a_{li}\ne ANY$,$a_{lj}=a_{li}(1\le l\le n)$,且 $a_{lk}=a_{lj}$,则若 $a_{li}\ne ANY$,$a_{lk}=a_{lj}=a_{li}(1\le l\le n)$.因此元组$(a_{1k},a_{2k},...,a_{nk},c_k)$必被规则 R_i 所覆盖.即所有被 R_j 覆盖的元组也被规则 R_i 所覆盖,所以 R_j 是冗余的.

1.2 优 化

为了减少分组属性的组合数目,降低算法的复杂度,在进行分组记数之前,我们采用一种信息论中的技术,用于计算按某属性的取值对数据集划分之后所获得的信息量,在本算法中用于选择一个需要与其他任意一种属性组合进行组合的属性.

假设属性 A 有 k 个不同取值,根据该属性的取值可将训练集 D 分成 k 个子集:D_1,D_2,...,D_K.划分后所获得的信息量可用公式(1)计算如下:

$$I_A=E(D)-\sum_{i=1}^{k}\frac{|D_i|}{|D|}E(D_i)\ ,\ E(X)=-\sum_{i=1}^{m}\frac{count(c_i,X)}{|X|}\cdot\log\frac{count(c_i,X)}{|X|}. \tag{1}$$

其中,$E(X)$计算的是数据集 X 的信息熵,$count(c_i,X)$指的是数据集 X 中属于类别 c_i 的元组个数,$|X|$代表数据集 X 中包含的元组总个数.在训练集包含的 n 个属性中,选择划分 D 获得信息量最大的一个作为最好的组合属性.

1.3 GAC-RDB(grouping and counting-relational database)算法

算法 **1**. 基于分组记数方法的分类算法.

Algorithm GAC-RDB (*TrainD*: *table*, minsup, min*conf*, min*error*: *real*)

1. **begin**
2. *sortedAttrList*:=*SortAttr*(*TrainD*);
3. **repeat**
4. *curAttrs*:=*SelectAttrs*(*sortedAttrList*);
5. *candDTable*:=*CandidateDTable*(*TrainD*,*curAttrs*);
6. *decisionTable*:=*PrunDTable*(*candDTable*,minsup,min*conf*);
7. *currenterror*:=*EstimateError*(*TestD*,*decisionTable*);
8. **until** *currenterror*≤*lasterror* **or** timeout
9. **end**

在该算法中,首先调用 *SortAttr*(*D*)函数计算利用每个属性进行类别判定的信息含量,从而选择一个最好的分裂属性,并且按照信息含量的大小对属性进行排序.接着循环地进行属性的选择、决策表的生成、剪裁以及分类误差的计算,直到满足结束条件为止,这里所用的结束条件可以是准确度或时间达到事先指定的阈值,也可以是准确度降低时停止.其中 *SelectAttrs*()选择将参加 CUBE 运算的属性,通过 CUBE 运算可以获取相关属性所有组合的类别分步信息,从而构造候选决策表;*CandidateDTable*()用于生成候选决策表;*PrunDTable*()对候选决策表进行剪裁;*EstimateError*()则利用当前得到的规则集对测试集进行测试求出分类误差.

1.4 根据决策表进行分类

生成决策表之后,就可以对测试集的元组或类别未知的其他元组进行分类了,对测试集的元组进行分类的目的是检验算法的准确度.分类器的真正用途则是对未知类别的元组进行分类.

为一个元组 $u(a_{1u},a_{2u},...,a_{nu})$进行分类的过程就是在决策表中寻找匹配规则的过程,即找到所有属性取值与 u 中相应值匹配的所有规则.查找匹配规则的结果有 3 种,一种是找到一个匹配的规则,另一种是找到多个匹配的规则,还有一种是找不到匹配的规则.若与 u 匹配的规则只有一个,则该规则中的类别就是 u 的类别.若匹配的规则有多个,则可选择具有最大置信度的规则为其分类;若置信度取值最大的规则不止一个,则选其中支持度最大的一个;若支持度取值为最大的规则仍不止一个,则选择最早生成的一个.

最后一种情况是找不到匹配的规则,在这种情况下,最简单的处理办法是利用训练集找到一个缺省类别,该类别是训练集中未被任何规则匹配的元组中类别取值最多的一个.这种方法虽然简单,但有时不易解释.为此,本文采用如下所述的基于贝叶斯概率的方法.

设 $u(a_{1u},a_{2u},...,a_{nu})$是未知类别的元组,$P(c_k|u)$是 u 属于类别 c_k 的概率.根据贝叶斯定理,假设各属性的取值互相独立,则可以得到

$$P(c_k\mid u)\propto p(c_k)\prod_{i=1}^{n}p(a_{iu}\mid c_k)=\frac{\prod_{i=1}^{n}p(a_{iu}\wedge c_k)}{p(c_k)^{n-1}}. \tag{2}$$

根据公式(2)就可以计算出 u 属于每个类别的概率 $P(c_k|u)(1\le k\le m)$,然后选择概率最大的一个就可以作为 u 的类别.

2 用 SQL 语句实现算法

2.1 选择一个分组属性并排序属性

由于 CUBE 操作是一个相对来说比较费时的操作,因为若干属性的组合数目随属性个数的增加而增长迅速.因此为了减少参加组合的属性的个数,我们采用了信息论中信息熵的概念,计算分别按照每个属性的取值对训练集进行分裂后所获得的信息量,选择其中最大的属性为第 1 个分组属性,即不需参加每个 CUBE 操作的属

性.具体算法如下所示.

算法 **2**. 属性排序的算法.

Algorithm *SortAttr* (*TrainD*: *relation*);

1. **begin**
2. 通过执行下列 SQL 语句计算每个属性的类别分布
3. **SELECT** $A_1,A_2,\ldots,A_n,class,count(*)$
4. **FROM** *TrainD*
5. **GROUP BY GROUPING STES** $((A_1,class),(A_2,class),\ldots,(A_n,class))$
6. **ORDER BY** $A_1,A_2,\ldots,A_n$
7. **for each** attribute $A_i \in \{A_1,A_2,\ldots,A_n\}$ **do**
8. Compute entropy $E(A_i)$;
9. Sort attributes on $E(A_i)$ into *sortedAttrList*;
10. **return** *sortedAttrList*;
11. **end**.

其中计算信息熵所需的每个属性的类别分布由第 3~6 行的 SQL 语句计算,在从数据库传送到用户程序的过程中直接计算到每个属性的信息熵的值中,同时可以计算每个结果行的支持度和置信度,若满足最小支持度和最小置信度的要求,则插入到决策表中(第 7~8 行).最后按照信息熵值的大小将属性进行排序(第 9 行).

2.2 参与CUBE运算的属性的选择

假设 *ic* 是 CUBE 运算的初始大小,*mc* 是 CUBE 运算所涉及的属性的最大个数,函数 *SelectAttrs*()选择参与 CUBE 运算的属性集的方法如下:

(1) 初始情况下,直接取出已排好序的属性中前 *ic* 个属性作为当前 CUBE 运算的属性组,称为 curAttrs.

(2) 每次循环结束时,从决策表中选择出与类别相关的所有属性组成一个属性集,称为 relevantAttrs,其大小记为 *R*.

(3) 若 *R* 小于 *ic*,*RelevantAttrs* 以及当前前(*ic*−*R*)个排好序的属性共同组成当前的属性组 *curAttrs*.若 *R* 大于等于 *ic*,但小于 *mc*,则 *curAttrs* 则由 *relevantAttrs* 和排好序的属性列表中的下一个属性组成.

(4) 当 *R* 大于等于 *mc* 时,*curAttrs* 则由 *relevantAttr* 中的前 *mc* 个属性和排好序的属性列表中的下一个属性组成.

这个属性选择的过程实际上是一个特征抽取(feature selection)的过程.但是与一般的特征抽取算法不同的是,它不耗费额外的时间.通常特征抽取算法都是非常费时的,需要对整个数据集进行多次扫描.本文则巧妙地利用规则发现过程中得到的数据进行属性的选择,节省了大量时间.由于在分类器产生的规则集中有很多规则只包含很少的属性,但却覆盖了很多的训练集样本,因此本文就采用了这种贪婪式逐步缩小搜索空间式的选择方法.

2.3 生成候选决策表

算法 SortAttr 中第 5 行的 CandidateDTable 函数调用是用来生成候选决策表的,其主要操作是计算所有分组的类别分布数值,这可以由下列的 GROUP BY CUBE 语句完成:

SELECT $A_1,A_2,\ldots,A_l,class,count(*)$

FROM *TrainD*

GROUP BY A_k,**CUBE**$(A_1,A_2,\ldots,A_{k-1},A_{k+1},\ldots,A_l),class$

ORDER BY $A_1,A_2,\ldots,A_l$

其中属性组$(A_1,A_2,\ldots,A_l)$是选择出的需要参与当前 CUBE 运算的属性,A_k 是第 1 个分裂属性.实际上候选决策表并不是一个物理上存在的表,其表中的数据在进一步处理之前只需留在数据库的缓冲区即可.

2.4 剪裁候选决策表生成决策表

有了候选决策表之后,过滤其中的每一行,保留满足条件的就可生成最终的决策表.具体过程如下:

(1) 从候选决策表中按组取出行,统计该组包含的行的总个数.计算组中每一行的支持度和置信度.

(2) 去掉支持度和置信度小于给定阈值的行.对满足最小支持度和最小置信度的行,对照决策表中已有规则根据定理 1 检查该行是否冗余.若不冗余,将其插入决策表;否则,将其丢弃.

(3) 继续处理候选决策表中其他组,直到所有行均被处理过为止.

3 算法测试与实验

3.1 分类准确度

为了验证算法的准确度,本文选择 UCI(University of California, Irvine)的机器学习数据库中的若干数据集进行实验,并将实验结果与文献[6]中列出的各种分类器的准确性数据进行比较.比较结果见表 1.

Table 1 Accuracy of classifiers

表 1 分类器的准确度

Data set	Properties				Accuracy					
	#attrs	#classes	#train	#test	C4.5	NB	TAN	CBA	LB	GAC
Australian	14	2	690	CV-10	0.843	0.857	0.852	0.855	0.857	**0.883**
Chess	36	2	2 130	1 065	**0.995**	0.872	0.921	0.981	0.902	0.944
Diabetes	8	2	768	CV-10	0.717	0.751	0.765	0.729	0.767	**0.767**
Flare	10	2	1 066	CV-10	0.812	0.795	0.826	0.831	0.815	**0.843**
German	20	2	1 000	CV-10	0.717	0.741	0.727	0.732	0.748	**0.768**
Heart	13	2	270	CV-10	0.767	0.822	0.833	0.819	0.822	**0.838**
Letter	16	26	15 000	500	0.777	0.749	**0.857**	0.518	0.764	0.800
Lymph	18	4	148	CV-10	0.784	0.819	0.838	0.773	**0.846**	0.839
Pima	8	2	768	CV-10	0.711	0.759	0.758	0.730	0.758	**0.780**
Satimage	36	6	4 435	2 000	0.852	0.818	**0.872**	0.849	0.839	0.847
Segment	19	7	1 540	770	**0.958**	0.918	0.935	0.935	0.942	0.943
Splice	60	3	2 126	1 064	0.933	0.946	0.946	0.700	0.946	**0.956**
Shuttle-small	9	7	38 661	934	0.995	0.987	0.996	0.995	0.994	**0.998**
Vehicle	18	4	846	CV-10	0.698	0.611	**0.709**	0.688	0.688	0.681
Voting Records	16	2	435	CV-10	**0.957**	0.903	0.933	0.935	0.947	0.956
Waveform-21	21	3	300	4 700	0.704	0.785	0.791	0.753	**0.794**	0.761
Yeast	8	10	1 484	CV-10	0.557	0.581	0.572	0.551	**0.582**	0.574
Average					0.810	0.807	0.831	0.787	0.824	**0.834**

表 1 中除了本文提出的算法 GAC-RDB(表中以 GAC 代替)之外另有 5 个著名的算法,它们分别是:(1) C4.5,由 Quinlan 提出的决策树分类器[1];(2) NB,一种简单的贝叶斯分类器[7];(3) TAN,贝叶斯网络分类器[7];(4) CBA,基于关联规则的分类器[4];(5) LB,使用大的项集基于概率统计计算的分类器[6].

从表 1 中可以看到,与其他的 5 种分类器相比,在 17 个数据集中,有 8 个数据集利用本文的算法 GAC-RDB 得到的准确度最高,次高的有 5 个,且不存在准确度低于其他所有算法的情况.因此,我们可以得出结论,算法 GAC-RDB 的准确度超过或至少等同于其他分类器的准确度.

3.2 速度及随元组个数增长的可扩展性

由于上述测试数据集中不存在适合于完成此类实验的数据集,因此,本文采用合成的具有各种不同复杂程度分类函数的数据集.为了与其他文献中的数据进行比较,也为了测试各种不同复杂程度的分类函数的分类效果,本文将对函数 2、5 和 10 进行测试,其中函数 10 是 10 个函数中最复杂的分类函数.

为了将算法 GAC-RDB 与当前流行的可扩展的决策树算法 SLIQ(supervised learning in quest)进行比较,本文自行实现了 SLIQ 算法,经测试表明与文献[2]中的执行速度相当.图 1 是这两个算法的执行时间的比较.从图 1 中可以看出,随着训练集元组

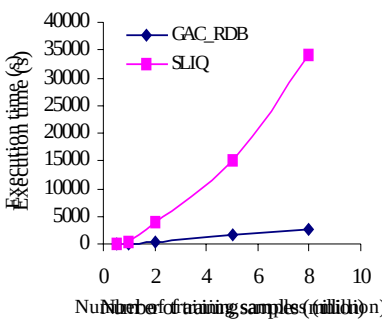

Fig.1 Execution time and No. of samples

图 1 执行时间和样本个数

个数从 50 万个增长到 800 万个,算法 GAC-RDB 所需的执行时间是呈线性增长的,其执行速度明显快于 SLIQ.

为了与其他算法进行比较,我们获取了算法 CBA 的可执行程序[4],对函数 10(仅选用相关的 5 个属性)分别用算法 CBA 和 GAC-RDB 进行实验,结果如图 2 所示.图中实线所示的 CBA 的执行时间仅仅是该算法对训练集每次扫描时间的总和,而不包括在扫描之前必须对数据集进行的数据格式的转换,该格式是利用传统的关联规则的挖掘算法所需要的.若加上该转换时间,则实际所需时间大约为图中虚线 CBA-2 所示的时间.从图 2 可以看到,本文提出的算法 GAC-RDB 要比 CBA 快大约 10 倍.值得注意的是,当训练集的元组个数大于等于 500 万时 CBA 已经无法运行.

3.3 算法随属性个数增长的可扩展性

为了测试算法随着属性个数的增长算法的可扩展性,对于函数 10 的数据集从 9 个属性一直增加到 400 个属性,训练集中元组个数固定在 10 万个,测试集为 1 万个,执行 GAC-RDB 所需时间如图 3 所示.当属性个数急剧增长时,算法 GAC-RDB 的执行时间随着增长,但增长比较缓慢,增长趋势是接近线性的.对于算法 SLIQ 来说,当属性个数少于 200 个时,其训练集所对应的属性列表和类别列表全部可以存放在内存中,因此执行速度较快,而当属性个数大于等于 200 个时,每个属性列表存于硬盘中,算法执行时间增长显著.

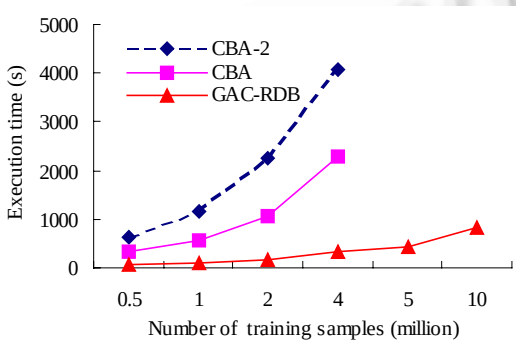

Fig.2 Comparing with CBA
图 2 与算法 CBA 比较

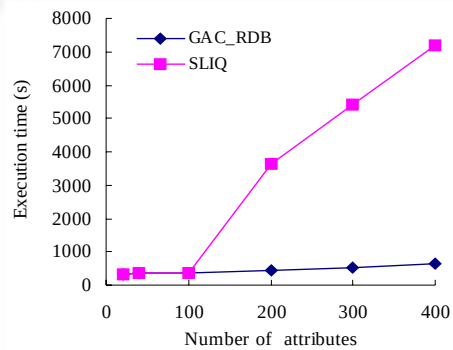

Fig.3 Execution time and No. of attributes
图 3 执行时间和属性个数

4 总 结

本文提出了一种新颖的分类算法,可以构建高效的具有良好可扩展性的分类器.利用文中提出的算法,可以将构建分类器所需的统计数据通过关系数据库管理系统计算得到,避免了应用程序对训练集的多次顺序扫描和大量的数据传送,提高了分类系统与数据库系统的集成度,同时也提高了算法执行的速度.利用算法中的剪裁策略,可以有效地将大量的冗余规则去掉,从而生成简洁的规则集.利用本文提出的有效的属性选择方法逐步构造决策表,可以使算法具有随元组个数增长和随属性个数增长两方面的良好的可扩展性.

References:

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

[2] Mehta, M., Agrawal, R., Rissanen, J. SLIQ: a fast scalable classifier for data mining. In: Apers, P., Bouzeghoub, M., Gardarin, G., eds. Proceedings of the 5th International Conference on Extending Database Technology. Berlin: Springer-Verlag, 1996. 18~32.

[3] Wang, M., Iyer, B., Vitter, J.S. Scalable mining for classification rules in relational databases. In: Eaglestone, B., Desai, B.C., Shao, Jian-hua, eds. Proceedings of the 1998 International Database Engineering and Applications Symposium. Wales: IEEE Computer Society, 1998. 58~67.

[4] Liu, B., Hsu, W., Ma, Y. Integrating classification and association rule mining. In: Agrawal, R., ed. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining. New York: AAAI Press, 1998. 80~86.

[5] Agrawal, R., Shim, K. Developing tightly-coupled data mining applications on a relational database system. In: Simoudis, E., ed. Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining. Cambridge, MA: AAAI Press, 1996. 112~118.

[6] Meretakis, D., Wüthrich, B. Extending Naïve Bayes classifiers using long itemsets. In: Chaudhuri, S., ed. Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining. San Diego, CA: AAAI Press, 1999. 295~301.

[7] Friedman, N., Geiger, D., Goldszmidt, M. Bayesian network classifier. Machine Learning, 1997,29(1):131~163.

A Scalable Classification Algorithm Exploring Database Technology*

LIU Hong-yan[1], LU Hong-jun[2], CHEN Jian[1]

[1](*School of Economics and Management, Tsinghua University, Beijing* 100084, *China*);

[2](*Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, China*)

E-mail: {liuhy,chenj}@em.tsinghua.edu.cn; luhj@cs.ust.hk

http://www.cs.ust.hk/~luhj

Abstract: This paper focuses on the study of efficient and scalable classification algorithm that tightly integrates classification technology with relational database system technology. In this paper, an approach based on grouping and counting is proposed to build classifier, which uses SQL (structured query language) provided by relational database to implement the major computation tasks. In order to improve the performance, several optimization strategies and a redundant rules' pruning strategy together with a feature selection method integrating with the process of finding classification rules are also proposed. With all these methods and strategies, the classification algorithm can find a compact set of classification rules quickly from a large volume of data. In addition to the same classification accuracy with current popular classification algorithms and high training speed, the unique features of the classification algorithm also include its linear scalability with respect to the number of training samples and the number of attributes, and the simplicity in implementation.

Key words: data mining; classification; RDBMS (relational database management system); SQL (structured query language)

* Received July 31, 2000; accepted November 23, 2000

Supported by the National Grand Fundamental Research 973 Program of China under Grant No.G1998030414; the 985 Basic Research Foundation of Tshinghua University of China