基于SQL的属性核与约简高效计算方法

冯 林[1,2] 李天瑞[2]
(四川师范大学计算机科学学院 成都 610068)[1]
(西南交通大学信息科学与技术学院 成都 610031)[2]

摘 要 结合关系数据库中高性能的SQL语言操作,提出了关系数据库中属性核、属性约简与值核的计算方法。仿真实验结果表明,该方法与经典粗糙集相关方法相比,有更高的执行效率。

关键词 粗糙集,SQL语言,属性核,属性约简
中图法分类号 TP301 **文献标识码** A

High Efficiency Approaches for Computing Attributes Core and Reductions Based on SQL

FENG Lin[1,2] LI Tian-rui[2]
(College of Computer Science, Sichuan Normal University, Chengdu 610068, China)[1]
(School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China)[2]

Abstract Calculating approaches for attributes core, attributes reductions and values core were developed in relation databases by combing rough set theory and SQL. Experimental results illustrate these methods have the better performance in run-time efficiency comparing with the related approaches of the classical rough set theory in processing decision tables.

Keywords Rough set, SQL, Attributes core, Attributes reductions

粗糙集理论由波兰逻辑学家Pawlak教授于1982年提出。粗糙集理论由于能够定量分析处理不严密、不确定或不完全的信息和知识,因此作为一种具有极大潜力和有效的智能信息处理技术受到了AI工作者的广泛关注,已在特征选择[1]、决策规则生成[2]、数据融合[3]、故障诊断[4]等方面取得了较成功的应用。

然而,对原始数据集的处理,粗糙集理论的传统处理方法存在计算复杂度高的缺点。针对这一问题,众多研究者目前仍然集中在研究如何改进各种方法的性能[5-7]。而对于如何改进它的体系结构,使之快速处理原始数据集,则关注甚少。在现实生活中,粗糙集理论的主要应用是对数据库的数据挖掘。如果能把粗糙集理论与关系数据库技术结合起来,将会提高粗糙集方法在数据库中属性约简及知识获取方面的实用性,且具有许多优点:(1)数据库发展了很多年,比较成熟。通过将数据挖掘集成在现有的成熟的数据库中,数据挖掘可以平滑地集成到各种应用系统上,而不像当前那样每一个数据挖掘应用系统都得从头开发设计。(2)数据挖掘的研究工作更有继承性,通过数据库通用的查询语言来定义和查询数据挖掘的知识,好的查询优化(涉及数据挖掘性能的改进)可以形成标准的库函数而嵌入数据库系统中[8,9]。

本文以粗糙集理论为基础,结合关系数据库中高效的查询语言操作,对基于粗糙集理论的数据挖掘方法的体系结构进行了分析与研究,提出了在关系数据库中基于SQL语句的属性核、属性相对约简和规则中值核的计算方法,最后给出了相关的实验测试。

1 粗糙集理论的基本概念

为了便于叙述,首先简单介绍与本文相关的粗糙集理论的基本概念,本节中所有定义均出自文献[10]。

定义1(决策表) 一个决策表$S=(U,R,V,f)$,其中U是对象的集合,也称为论域;$R=C\cup D$是有限非空属性集合,$C\cap D=\varnothing$,$D\neq\varnothing$,子集C和D分别称为条件属性集和决策属性集;$V=\bigcup_{r\in R}V_r$是属性值的集合,V_r表示属性r的值域;$f:U\times R\to V$是一个信息函数,它指定U中每个对象x的属性值。

定义2(不分明关系) 给定$S=(U,R,V,f)$,$B\subseteq R$,不分明关系$IND(B)$定义如下:

$$IND(B)=\{(x,y)\mid(x,y)\in U\times U,\forall b\in B\ \forall x\in U\ \forall y\in U\ (f(x,b)=f(y,b))\} \quad (1)$$

定义3(条件分类和决策分类) 给定$S=(U,C\cup D,V,f)$,设$U/IND(C)$和$U/IND(D)$分别为论域U在属性集C和D上形成的划分,条件分类定义为$E_i(E_i\in U/IND(C))$,决策分类定义为$X_j(X_j\in U/IND(D))$,其中,$i=1,2,\cdots,|U/IND(C)|$,$j=1,2,\cdots,|U/IND(D)|$,$|X|$为集合X的势。

定义4(确定与不确定决策表) 给定$S=(U,C\cup D,V,f)$,对任意$E_i\in U/IND(C)$,如果属于它的所有记录都有相

到稿日期:2009-03-02 返修日期:2009-05-16 本文受国家自然科学基金(No. 60873108),四川师范大学科研基金重点项目(No. 06lk012)资助。
冯 林(1972-),男,博士,副教授,研究方向为粗糙集理论、数据挖掘等,E-mail:scfengyc@126.com;李天瑞(1969-),男,博士后,教授,博士生导师,研究方向为粗糙集理论、粒计算、数学模型等。

同的决策值，则称 E_i 为一致的；否则称 E_i 为不一致。如果 S 所有的条件分类都是一致的，则称 S 为确定决策表，否则称 S 为不确定决策表。

定义 5（相对正域） 设 U 为一个论域，P，Q 为定义在 U 上的两个等价关系簇，Q 的 P 正域 $POS_P(Q)$ 定义如下：

$$POS_P(Q)=\bigcup_{X\in U/Q} P_-(X) \tag{2}$$

定义 6（属性核） 设 P，Q 是定义在 U 上的两个等价关系簇，若 $POS_P(Q)=POS_{P-\{r\}}(Q)$，则称属性 r 为 P 中相对于 Q 不必要的，否则称 r 为 P 中相对于 Q 必要的。P 中所有相对于 Q 必要的属性组成的集合称为 P 的 Q 核，记为 $CORE_Q(P)$。

定义 7（相对约简） 设 P，Q 是定义在 U 上的两个等价关系簇，对任意的 $r(r\in P)$ 都是 P 中相对于 Q 必要的，称 P 为相对于 Q 独立的。若 P 的 Q 独立子集 $S\subset P$，有 $POS_S(Q)=POS_P(Q)$，则称 S 为 P 的 Q 约简。

2 关系数据库中属性核、属性约简及核值的计算方法

2.1 用 SQL 语句求绝对必要属性

定理 1 给定 $S=(U,C\cup D,V,f)$，属性 r 是 C 中必要属性的充要条件是 $\exists E_i(E_i\in U/IND(C))$。设 $E_j(E_j\in U/IND(C\setminus\{r\}))$ 是与 E_i 相对应的等价类，满足 $|E_i|\neq|E_j|$，其中，$i=1,2,\cdots,|U/IND(C)|$，$j=1,2,\cdots,|U/IND(C\setminus\{r\})|$。

证明：如果一个属性 r 为绝对必要属性，那么决策表中由 C 和 $C\setminus\{r\}$ 所确定的等价类的划分必然不相等。根据定义 6 得证。

根据定理 1，可以给出利用 SQL 操作判定属性绝对必要的方法。首先，根据条件属性集和决策属性集，使用 SQL 语句对原始信息表进行排序，并得到两个临时表 $temp_1$ 和 $temp_2$，具体的 SQL 操作如下：

```
Create Table temp1
Select C\{r},{r}
From S
Order by C\{r},{r}
Create Table temp2
Select C\{r}
From temp1
Order by C\{r}
```

绝对必要属性和绝对不必要属性都是在决策表 S 的条件属性集上进行处理的，因而在得到临时表 $temp_1$ 和 $temp_2$ 后，只需要对 $temp_1$ 和 $temp_2$ 的记录进行处理就可以了。在 $temp_1$ 中，采用对相邻两行逐行比较的方法得到原决策表在 C 上的某一等价类中对象的个数 $Count_1$；在 $temp_2$ 中，已经根据条件属性 $C\setminus\{r\}$ 对所有记录进行了排序，现在要做的就是计算 $temp_2$ 中记录在 $C\setminus\{r\}$ 上的对应等价类中对象的个数 $Count_2$。如果 $Count_1\neq Count_2$，说明 r 是绝对必要属性。对 C 中的各个属性进行上述操作，即可得到所有的绝对必要和绝对不必要属性。

2.2 用 SQL 语句求相对属性核

属性约简是粗糙集理论知识获取的核心问题，而计算属性核往往是属性约简的基础。本节给出结合 SQL 操作求属性核的方法。

如果一个属性 a 是核属性，在不确定决策表 S 中，删除属性 a 有两种情况：一是导致至少有两个对象 x_1 和 x_2 产生冲突，而这两个对象在原来的系统中都属于正域；二是导致有一个正域中的对象 x_1 和一个原本在决策表中非正域上的对象 x_2，它们在条件属性 $C\setminus\{a\}$ 上的取值一样。这两种情况都导致了决策表中的正域发生变化（减小）。通过此分析易知，要判断某一属性 a 是否是核属性，必须得到决策表在包含或不包含属性 a 时系统正域的势。由以上分析，下面给出求决策表 S 中正域势的 SQL 语句：

```
Create Table temp3
Select C\{a},D
From S
Order by C\{a},D
Select Count(temp3.C\{a}) As PosCount
From temp3,S
Where Not Exist (
    Select * from temp3,S
    Where(temp3.C\{a}.Value=S.C\{a}.Value
    And temp3.D.Value≠S.D.Value))
```

当然，也可以通过比较删除属性 a 前后 S 中冲突对象个数的变化来判定。

由以上讨论可以得到定理 2 和算法 1。

定理 2 给定不确定决策表 $S=(U,C\cup D,V,f)$ 中正域对象的个数 PosCount，属性 $a\in C$ 是必要属性（核属性或不可去掉属性）的充要条件是 $\text{PosCount}(POS_{C\setminus\{a\}}(D))\neq\text{PosCount}(POS_C(D))$。

算法 1 求相对属性核

输入：不确定决策表 $S=(U,C\cup D,V,f)$；

输出：S 的相对核属性集 Core。

Step1 Core=∅；

Step2 For $a_i\in C(i=1,2,\cdots,|C|)$，进行如下操作：

If $\text{PosCount}(POS_{C\setminus\{a_i\}}(D))\neq\text{PosCount}(POS_C(D))$ Then Core=Core∪$\{a_i\}$；

$i=i+1$；

Step3 Stop。

2.3 用 SQL 语句求相对约简

在关系数据库中，计算最小约简是一个 NP 问题。而在实际应用中，一般只需要得到决策表的相对约简就可以了。计算相对约简，一般采用启发式方法。在得到属性核的基础上，根据某种启发式方法来添加新的属性，直到所得 C 的子集 Redu 相对于决策 D 的正域与 C 相对于决策 D 的正域相等。下面，给出利用 SQL 语句求相对属性约简的启发式方法。

定义 8（属性相对于属性集的重要性） 设 Redu$\subseteq C$，对任意的 $a\in C$，属性 a 相对于属性集 Redu 的重要性定义为：

$$\text{Imp}(\{a\})=1-\frac{\text{PosCount}(\text{Pos}_{\text{Redu}}(D))}{\text{PosCount}(\text{Pos}_{\text{Redu}\cup\{a\}}(D))} \tag{3}$$

定理 3 设 Redu$\subseteq$C，则 Redu 是条件属性集 C 的一个约简（相对约简）的充要条件是 $\text{PosCount}(\text{Pos}_{\text{Redu}}(D))=\text{PosCount}(\text{Pos}_C(D))$。

证明：根据定义 7 和定理 2，容易得证。

于是，我们给出如下计算不确定决策表 S 中相对约简的方法。

算法 2 计算相对约简的算法

输入：不确定信息表 $S=(U,C\cup D,V,f)$，S 的核属性集 Core；

输出：S 的一个相对约简 Redu。

Step1 Redu=Core；

Step2 利用 SQL 语句求出 $\text{PosCount}(\text{Pos}_C(D))$ 和 PosCount

$(Pos_{Redu}(D))$;

Step3 While($PosCount(Pos_{Redu}(D))<PosCount(Pos_C(D))$)Do

Importance=0;k=0;

For $\forall a_i \in C \backslash Redu(i=1,2,\cdots,|C\backslash Redu|)$,进行如下操作:

利用 SQL 语句和属性相对于属性集重要性的定义计算 $Imp(a_i)$;

If $Imp(a_i)>Importance$ Then

Importance= $Imp(a_i)$;$k=i$;

End if

Loop

Redu=Redu$\cup\{a_i\}$;

Loop

Step4 Return Redu and Stop。

2.4 用 SQL 求值核

通过值约简可以得到表示更简单、表现能力更强的规则[10]。在值约简过程中,值核的计算是一个重要环节。下面是用 SQL 操作求值核的方法。

```
Create Table temp4
Select {a},D
From S
Select Count(Temp4.{a}) As tempCount
From temp4,S
Where ((Select Count(Temp4.{a}) As ConCount from temp4,S
      Where(Temp4.{a}.Value=S.{a}.Value And temp4.D.
           Value=S.D.Value))
And (Select Count(Temp4.{a}) As TotalCount from temp4,S
      Where(Temp4.{a}.Value=S.{a}.Value))
And (ConCount/TotalCount ≥Confidence)
)
```

其中,Confidence=1,表示固定精度模型。如果 Confidence<1,则表示变精度模型。很显然,如果 tempCount=0,则表示某条规则中属性 a 的取值是值核。

根据上述讨论,下面给出求某条规则值核的算法。

算法 3 求某条规则值核

输入:不确定信息表 $S=(U,C\cup D,V,f)$,Redu 中的一条记录 Rcd;

输出:Rcd 的值核 ValueCore。

Step1 ValueCore=$\emptyset$;

Step2 For $\forall a_i \in Redu(i=1,2,\cdots,|Redu|)$ Do

利用 SQL 语句计算 a_i 的 tempCount;

If (tempCount= =0) Then

ValueCore=ValueCore$\cup\{a_i\}$

End If

Loop

Step3 Stop。

3 实验测试

为了验证上述方法的有效性,考虑表 1 所列气象决策表的例子。

首先,根据算法 1 可求得表 1 属性核 Core=$\{a_1,a_4\}$;根据属性核,由算法 2 可得到表 1 的一个相对约简 Redu=$\{a_1,a_3,a_4\}$,而这个相对约简也是最小约简;然后,根据 Redu 和算法 3,设定阈值 Confidence=1,依次对表 1 中的每一条约简后的记录进行处理,可以得到如下的值核:$\{a_1=0,1,2;a_3=0,1;a_4=0,1\}$。

表 1 关于气象例子的决策表

	条件属性集 C				决策属性 D
	Outlook(a_1)	Temperature(a_2)	Humidity(a_3)	Windy(a_4)	
1	Sunny	Hot	High	False	N
2	Sunny	Hot	High	True	N
3	Overcast	Hot	High	False	P
4	Rain	Mild	High	False	P
5	Rain	Cool	Normal	False	P
6	Rain	Cool	Normal	True	N
7	Overcast	Cool	Normal	True	P
8	Sunny	Mild	High	False	N
9	Sunny	Cool	Normal	False	P
10	Rain	Mild	Normal	False	P
11	Sunny	Mild	Normal	True	P
12	Overcast	Mild	High	True	P
13	Overcast	Hot	Normal	False	P
14	Rain	Mild	High	True	N

为了验证本文方法的高效性,我们对表 1 中的数据进行测试。实验的硬件环境是 CPU 为 Celeron 1.3GHz,内存为 128MB,操作系统为 Windows XP 的笔记本电脑。

具体测试过程如下:(1)用本文方法计算表 1 中的属性核、属性约简、值核,分别记录其运行时间;(2)用经典方法计算表 1 的属性核、属性约简、值核,分别记录运行时间。测试结果如表 2 所列。

表 2 测试结果

	本文方法所用时间(s)	经典方法所用时间(s)*
属性核	0.008	0.027
属性约简	0.017	0.033
值核	0.022	0.038

* 注:经典方法计算属性核用可辨识矩阵方法[10],属性约简用一般属性约简方法[10],核约简采用一般值约简方法[10]。

从表 2 测试结果来看,对比经典求属性核、属性约简和核值方法,本文方法具有较高的运行效率。

结束语 粗糙集理论是一种新的处理含糊和不确定性知识的数学工具。本文结合粗糙集理论和高效的 SQL 操作提出了使用 SQL 实现关系数据库中知识获取的方法,给出了用 SQL 语句求数据库中相对核属性、属性约简和值核的算法,并提供了相关的实例说明,证明了算法的有效性。虽然本文的工作只在一个小数据集上进行了实验,但这些方法可以扩展到海量数据集应用中。因此,本文方法对基于粗糙集理论的海量数据挖掘提供了一种有益的尝试。

参 考 文 献

[1] 王国胤,于洪,杨大春.基于条件信息熵的决策表约简[J].计算机学报,2002,25(7):759-766

[2] 张雪英,刘凤玉,Krause J.粗糙集分类算法的近似决策规则和规则匹配方法[J].计算机科学,2005,32(6):129-132

[3] 徐捷,徐从富,耿卫东,等.基于粗糙集理论的动态目标识别及跟踪[J].电子学报,2002,30(4):605-607

[4] 肖迪,胡寿松.实域粗糙集理论及属性约简[J].自动化学报,2007,33(3):253-258

[5] 赵军,王国胤,等.一种高效的属性核计算方法[J].小型微型计算机系统,2003,24(11):1950-1953

[6] 胡峰,王国胤.属性序下的快速约简算法[J].计算机学报,2007,30(8):1429-1435

[7] 高学东,丁军.基于简化差别矩阵的属性约简算法[J].系统工程理论与实践,2006,26(6):101-107

[8] 孙惠琴,熊璋.粗集理论集成 ORDBMS 的原型系统[J].计算机学报,2005,28(11):1875-1881

(下转第 267 页)

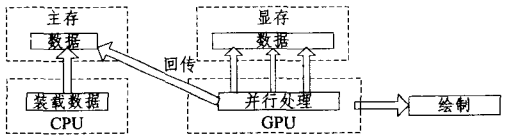


图 5　CPU 和 GPU 的计算分配

2.3　实现过程

本文实验方案的主要步骤如下：

(1)读入图像数据，按照图像的大小在 GPU 上分配设备内存 A，把数据传到设备内存 A；

(2)设定 GPU 的执行参数，即 grid 和 block 的维度；

(3)将图像数据与 GPU 的纹理内存绑定，然后建立与 GPU 内存的映射；

(4)在 OpenGL 中开辟 GPU 程序运行所需的缓冲区，完成 GPU 与 OpenGL 的连接；

(5)运行 kernel 函数对图像进行边缘检测；

(6)OpenGL 显示处理结果。

方案流程图如图 6 所示。

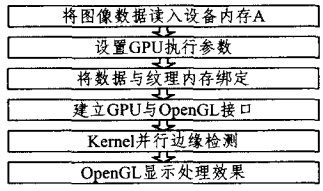


图 6　方案流程图

定义一个 kernel 函数要用到“__global__”限定符，而每次调用所并行执行的线程数量用扩展的“〈〈〈 〉〉〉”语法来指定。在程序中 kernel 被声明为：

EdgeDetect〈〈〈imageH，64〉〉〉(imageIn，imageW，imageW，imageH)　　(3)

其中，“〈〈〈〉〉〉”中参数表示分配了 imageH 个线程块，每个线程块中包含 64 个活动线程。imageIn 为要处理的图像数据起始地址指针，imageW 为图像宽度，imageH 为图像高度。

线程块的大小声明为图像高度 imageH，则每一个线程块负责一行像素的边缘检测运算；每一个线程块中包括线程数为 64 个(GPU 特性决定，线程块大小为 32 的整数倍时 GPU 运算效率相对较高)，则每一个线程负责 imageW/64 个像素点的卷积运算，这样就有 64×imageH 个独立的线程并行执行式(3)。

3　实验结果与分析

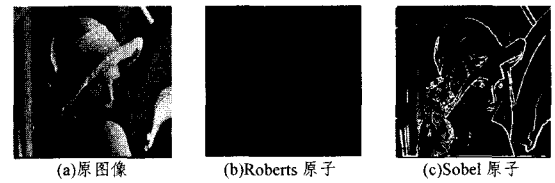
在 CPU 和 GPU 上分别运行 Roberts，Sobel 边缘检测算法，其中 CPU 实现根据算法步骤直接编程得到，并未对其进行任何优化，程序运行环境如表 1 所列。

表 1　实验平台参数

CPU(Intel Pentium Dual E2160)		GPU(GeForce 8800GT)	
主频	1.8GHz	主频	1.5GHz
L2 Cache	1024kB	处理器组数	14
内存	1.0 GB	设备内存	512MB

编译环境：Visual Studio 2005

取 LENA 图像作为实验图像，如图 7(a)所示。Roberts 算子与 Sobel 算子处理效果如图 7(b)、图 7(c)所示。



(a)原图像　(b)Roberts 原子　(c)Sobel 原子

图 7　处理效果图

使用 CUDA 自带的测定时间函数可以得到程序运行的时间，由于得到的时间只能为毫秒，因此要采用多次(1000 次或 100 次)反复处理后求平均时间的方式来得到。不同大小图像数据的 CPU 运算时间、GPU 运算时间及加速比如表 2、表 3 所列。

表 2　Roberts 算子耗时测试结果

图像尺寸	128×128	256×256	512×512
CPU(ms)	0.172	0.711	2.897
GPU(ms)	0.055	0.077	0.169
加速比	3.127	9.233	17.172

表 3　Sobel 算子耗时测试结果

图像尺寸	128×128	256×256	512×512
CPU(ms)	0.210	0.815	3.615
GPU(ms)	0.058	0.091	0.204
加速比	3.621	8.956	17.721

从表中可以看出，相对于 CPU，GPU 的处理速度有显著提升，平均加速比为 9.884 和 10.099；同时，随着数据量和运算量的增加，加速比也在增加，GPU 更能显出其在运算速度上的优势。

结束语　本文实现了基于 GPU 的边缘检测算法，并对实验结果进行了分析。结果表明，与 CPU 相比，GPU 依据其硬件结构先天的并行计算特点，在运行以边缘检测为代表的可分割为独立单元的算法中能够得到极高的效率提升。在图像数据量增大时，这种提升更加明显。在本实验平台上，GPU 运算获得了最高 17 倍和平均 10 倍的提升，显示出 GPU 在通用高密集数据运算方面具有较高的运算能力，可为目标快速检测与跟踪提供硬件支持。

参 考 文 献

[1]　章毓晋. 图像工程(上册)，图像处理(第二版)[M]. 北京：清华大学出版社，2006

[2]　Pharr M. GPU gem2[M]. Boston：Addison-Wesley，2005

[3]　Nvidia Official Site[DB/OL]. http://www.nvidia.com/

[4]　NVIDIA Corporation. NVIDIA CUDA Compute Unified Device Architecture Programming Guide(Version 1.1)[M]. 2007

[5]　段瑞玲，李庆祥，李玉和. 图像边缘检测方法研究综述[J]. 光学技术，2005，31(3)：415-419

[6]　田玲. 图像边缘检测中并行算法的应用与研究[D]. 成都：电子科技大学，2006：45-57

(上接第 238 页)

[9]　Han J C，Hu X H，Lin T Y. A New Computation Model for Rough Set Theory Based on Database Systems[C]//Proceedings of 5th International Conference on Data Warehousing and Knowledge Discovery. Prague，Czech Republic，2003：381-390

[10]　王国胤. Rough 集理论与知识获取[M]. 西安：西安交通大学出版社，2003