文章编号: 2096-1618(2017)06-0618-05

基于键树的粗糙集属性约简算法

张文阳, 蒋 瑜
(成都信息工程大学软件工程学院,四川 成都 610225)

摘要:属性约简是粗糙集理论研究的核心内容之一。差别矩阵因其简洁、直观而被广泛应用于属性约简中,但其包含了大量冗余元素,从而造成存储空间的极大浪费。基于键树的思想,提出一种对差别矩阵非空元素存储的新方法,该方法消除了差别矩阵中的重复元素,并使部分具有子父关系的元素共享键树中父集所在的路径,从而实现了对差别矩阵的压缩存储。最后,基于该键树提出了一属性约简算法。

关 键 词:粗糙集理论;差别矩阵;键树;属性约简
中图分类号:TP18 **文献标志码**:A
doi:10.16836/j.cnki.jcuit.2017.06.009

0 引言

粗糙集(rough set,RS)理论是由波兰科学院 Pawlak 院士于 1982 年提出的一种关于数据处理和推理的理论[1]。这一理论为处理模糊、不确定或不完备信息的分类问题提供了一种新工具,目前该理论已经被广泛用于人工智能、机器学习和模式识别等领域[2-6]。其主要思想是以保持知识库分类能力不变为前提,删除知识库中不必要或者不相关的属性,进而在很大程度上提高了系统潜在知识的清晰度,更能进一步反映决策表的本质信息[7]。然而,S. K. M Wong 等[8]已经证明了找出一个决策表的最小约简是 NP-hard 问题,而导致 NP-hard 问题的主要原因是属性的组合爆炸问题。

就目前而言按照约简思想的不同,人们已经提出很多不同的约简算法。其中,Skowron 提出的基于差别矩阵的约简算法,由于在表达决策系统核、约简以及其他概念和计算上有许多优点,因此深受广大学者的关注。基于差别矩阵属性约简算法的主要思想是:求出决策表的差别矩阵,利用差别矩阵的非空元素,根据启发式信息构建决策表的属性约简。然而,差别矩阵存在众多冗余元素[9-10],这些元素的存在不仅占据大量的存储空间,而且增加求简的复杂度[10]。

键树[11]又称数字查找树(digital search trees,DST),其特点是使相同集合映射到同一路径上,并基于共享前缀实现了对集合的压缩存储。基于 DST 的这个特性,提出基于键树的差别矩阵非空元素的存储方法:在该键树中除根结点和叶子结点(以$符号标识的结点)外每一个结点代表一个属性元素,并且每一条路径代表差别矩阵中一个或者多个差别信息,消除差别矩阵中相同的元素,同时实现了对差别矩阵的压缩存储。最后,基于该键树提出一属性约简算法。基于 UCI 数据库的实验测试,该算法是有效的。

收稿日期:2017-06-06
基金项目:四川省教育厅重点资助项目(17ZA0071);国家自然科学基金青年基金资助项目(61602064);四川省科技计划-重点研发资助项目(2017HH0088)

1 基本概念

文献[2]给出了粗糙集及差别矩阵的相关介绍。

定义 1 决策表 DT(decision table),可用一个四元组来表示, $DT=(U,C\cup D,V,f)$,其中,

U:$U=\{x_1,x_2,\cdots,x_n\}$ 为对象的非空有限集合,称为论域;

$C\cup D$:$C=\{\alpha\mid\alpha\in C\}$ 称为 U 的条件属性集,其中每一个 $\alpha_j\in C(1\leqslant j\leqslant m,m$ 为条件属性的总数),称为 C 的一个简单属性; $D=\{d\mid d\in D\}$ 称为 U 的决策属性,且 $C\cap D=\varnothing,C\neq\varnothing,D\neq\varnothing$;

V:$V=\cup V_\alpha(V_\alpha\in C\cup D)$ 是信息函数 f 的值域,而 V_α 表示值域;

f:$f=\{f_\alpha\mid f_\alpha:U\rightarrow V_\alpha,V_\alpha\in(C\cup D)\}$ 表示决策表的信息函数, f_α 为属性 α 的信息函数。

定义 2 决策表 $DT=(U,C\cup D,V,f)$,其中,令 $R\subseteq(C\cup D)$,则决策表的不可分辨关系为

$\mathrm{ind}(R)=\{(xi,xj)\mid f(xi,a)=f(xj,a),\forall a\in R\wedge x_1,x_2\in U\}$,即这是一个等价类记作 $[x]_R$。R 对 U 的划分记为 U/R。

定义 3 决策表 $DT=(U,C\cup D,V,f)$,中,若 $\forall X\in U$,R 为论域上的一个等价关系,则定义 X 的 R 正域为:$POS_R(X)=\{x\mid(\forall x\in U)\wedge([x]_R\subseteq X)\}=\cup\{Y$

$|(\forall Y\in\frac{U}{R})\wedge(Y\subseteq X)\}$。

决策表 1 中,$U=\{x_1,x_2,x_3,x_4,x_5,x_6\}$,$C=\{a,b,c,d\}$,$D=\{f\}$;

表 1　某一决策表

U	C				$\frac{D}{f}$
	a	b	c	d	
x_1	0	1	1	1	1
x_2	1	1	0	0	2
x_3	1	0	1	0	3
x_4	1	0	0	1	4
x_5	1	1	1	0	5
x_6	1	1	0	1	6

定义 4　设决策表中的论域个数$|U|=n$,则 DT 的差别矩阵是一个包含 $n\times n$ 个元素的矩阵。差别矩阵中的每一个元素称为差别信息,应满足如下定义:

$$m_{i,j}=\{a\in C\mid f(x_i,a)\neq f(x_j,a)\wedge r(x_i,x_j)=1\}$$

其中 $x_i,x_j\in U$

$$r(x_i,x_j)=\begin{cases}1, & x_i\in POS_C(D)\wedge x_j\notin POS_C(D)\\ 1, & x_i\notin POS_C(D)\wedge x_j\in POS_C(D)\\ 1, & x_i,x_j\in POS_C(D)\wedge(x_i,x_j)\notin \mathrm{ind}(D)\\ 0, & \text{otherwise}\end{cases}$$

因为差别矩阵是一个对称矩阵(关于对角线对称),因此只考虑矩阵的上三角或者下三角即可,在表 2 中写出表 1 对应的差别矩阵。

表 2　表 1 所对应的差别矩阵

U	x_1	x_2	x_3	x_4	x_5	x_6
x_1	$\varnothing$					
x_2	$\{a,c,d\}$	$\varnothing$				
x_3	$\{a,b,d\}$	$\{b,c\}$	$\varnothing$			
x_4	$\{a,b,c\}$	$\{a,d\}$	$\{c,d\}$	$\varnothing$		
x_5	$\{a,d\}$	$\{c\}$	$\{b\}$	$\{b,c,d\}$	$\varnothing$	
x_6	$\{a,c\}$	$\{d\}$	$\{b,c,d\}$	$\{b\}$	$\{b,c,d\}$	$\varnothing$

定义 5　DT 中的差别函数 f 定义如下:当前差别矩阵中的所有取值为非空集合的元素 $m_{i,j}$ 建立相对应的析取逻辑表达式,对所有析取逻辑表达式进行合取运算。差别函数的极小析取范式中的所有合取式就是 C 的所有 D 约简,简称约简。

例 1　由表 2 的差别矩阵得出表 1 所对应的差别函数为

$$\begin{aligned}f=&(a\vee c\vee d)\wedge(a\vee b\vee d)\wedge(a\vee b\vee c)\wedge(a\vee d)\\&\wedge(a\vee c)\wedge(b\vee c)\wedge(b\vee d)\wedge c\wedge d\wedge(c\vee d)\\&\wedge b\wedge(b\vee c\vee d)\wedge(b\vee c\vee d)\wedge b\wedge(c\vee d)\\=&b\wedge c\wedge d\end{aligned}$$

最终得出决策表 1 的约简为$\{b,c,d\}$。

定义 6　决策表 DT 中,设 $P\subseteq C$(C 为 DT 的所有条件属性集合),若 P 为 DT 所有约简的交集,并且 $P\neq\varnothing$,则 P 为 C 相对于 D 的核,记作 $CoreD(C)$。在差别矩阵中,只包含一个属性的差别信息的并集构成了核。所以由表 2 可知决策表 1 的核为$\{b,c,d\}$。

2　键树的设计与实现

定义 7　键树是一颗有序树[12-13],即在同一层中兄弟结点之间依所含字符自左至右有序。它是一颗度≥2 的树,标记根结点为‘null’,所有叶子结点的属性名标记为$;基于键树提出了构建决策表差别信息的存储算法,具体步骤如算法 1 所示:

算法 1　基于键树的决策表差别信息存储构建算法。

输入:决策表 DT;

输出:决策表差别信息存储键树;

Step1:创建决策表的根结点(root);

Step2:对决策表中所有对象对$<x_i,x_j>$,根据定义 4 计算其差别信息 CB;

Step3:调用 $buildTree(CB,root)$ 函数,把 Step2 中得到的差别信息 CB 插入到以 $root$ 为根结点的键树中;

Step4:算法结束。

$buildTree(CB,root)$ 函数的具体操作:

(1)若 CB 为 $\varnothing$,则转(6);

(2)选择 CB 中最左边一条属性 d,并记 $C=C-\{d\}$;

(3)若 $root$ 的所有子结点中,不存在一个结点 N 的值为 d,则转(5);

(4)若 $root$ 的所有子结点中,存在结点 N 值等于 a,则结点 $root$ 指向结点 N,回到(1);

(5)创建一个新的子结点 N,N 的值为 d,$root$ 指向

结点 N,返回(1);

(6)插入叶子结点\$,函数结束。

例 2 基于算法 1 和表 1,给出决策表表 1 所对应的差别信息压缩存储键树的构建过程如下:

首先,创建树的 *root* 根结点,根据定义 4 得到差别信息,并按 DT 中的条件属性从右到左进行排序。

然后,构建差别信息压缩存储键树。把第一条差别信息$\{d\}$,插入进树,并以\$结点作为叶子结点结束。第二条差别信息$\{c\}$,在 *root* 结点下没有一子结点包含属性 c,则在 *root* 结点下新建一子结点 c,并以\$结点作为叶子结点。第三条差别信息$\{c,d\}$插入时,发现根结点下存在 c 结点,则结点 c 不再插入,此时以 c 结点作为根结点继续插入,c 结点下没有一子结点 d,所以在 c 结点下添加 d 结点,以\$结点作为叶子结点结束。重复以上过程直至所有差别信息插入树中。

最后,得出决策表 1 所对应的差别信息存储键树,如图 1 所示。

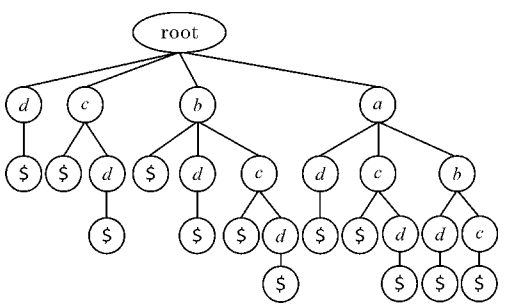

图 1 基于定义 4 和算法 1 构建的差别信息存储键树

2.1 决策表差别信息存储键树的性质

由算法 1 可知,决策表差别信息存储键树有如下性质:

性质 1 根结点不包含属性名,叶子结点包含的属性名都为\$。除根结点和叶子结点外每一个结点只包含一个属性名即 DT 的一个条件属性。

性质 2 每一个从根结点到叶子结点的路径上对应了差别矩阵中的一条或多条差别信息。

性质 3 同一路径下,每个结点包含的属性名都不相同。

性质 4 树的深度最大为$|C|+2$(C 为 DT 的条件属性个数)。

2.2 算法 1 的复杂度

在给定具有$|U|$个对象和$|C|$个条件属性的决策表中,则差别信息个数最多为$|U|^2$,除去重复差别信息以及空集,假设差别信息的实际个数 N,则算法的最大空间复杂度$((|C|+1)N)$,因此,算法的空间复杂度为$O(|C|N)$。由于 N 远远小于$|U|^2$,因此算法 1 所构建的树实现了对差别矩阵的压缩存储。

在构造键树的过程中,从根结点开始先进行差别元素的比较,每次迭代最多比较$|C|$次,迭代次数最多为$|U|^2$,则算法的时间复杂度为$O(|C||U|^2)$。

3 基于差别信息存储键树的属性约简算法

基于算法 1 所构建的差别信息存储键树提出基于该树的属性约简算法,如算法 2 所示。

算法 2 采用自左向右的策略获取最终的属性的约简。

输入:差别信息存储键树。

输出:决策表的属性约简。

Step1:从根结点开始,自上而下逐层查找属性值为\$的叶子结点;

Step2:删除属性值为\$的叶子结点和它的所有兄弟结点,只留下属性值为\$的叶子结点的父结点;

Step3:创建一个空集 R;

Step4:若树只包含 root 结点,则转 Step7;

Step5:若树中除根结点外存在只有一个结点的路径,则记这些结点组成的集合为 R',同时删除含有 R' 中元素的其它路径,记 $R=R'$;

Step6:假设树中只包含根结点,则转 Step7;否则取树中根结点最左子结点 d。则令 $R=R\cup\{d\}$,同时删除树上含有结点 d 的路径,返回 Step4;

Step7:算法结束。

例 3 基于图 1 和算法 2 属性约简过程如下:

根据 Step2 删除属性值为\$的叶子结点和它所有的兄弟结点,留下\$结点的父结点,则图 1 根据 Step2 的所得到的化简树如图 2 所示。

此时树中存在除根结点外只有一个结点的路径为:<*d*>、<*c*>、<*b*>,所以由 Step5 得:集合 $R'=\{b,c,d\}$,同时删除含有 d、c、b 结点的路径,则 $R=R'$。

由于此时差别信息存储键树中除 *root* 结点外没有其他结点,因此 $R=\{b,c,d\}$,算法结束,得到最后的约简$\{b,c,d\}$。

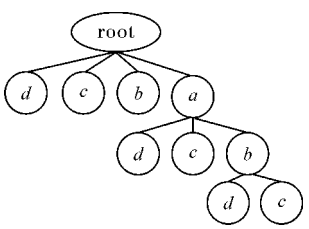

图 2 删除图 1 中\$结点及其兄弟结点后的树

依据 2.2 的分析,差别信息存储键树中节点数最多为$|C||U|^2$个,实际结点个数 N 远远小于$|C||U|^2$。从算法 2 的求解过程中可得,该算法最多迭代的次数为 C。如果每次迭代过程中删除结点数目为 N_i,则在 C 次迭代过程中删除结点的总数目 $N_1+N_2+\cdots+N_C=|C||U|^2$,所以,算法 2 的时间复杂度为 $O(|C||U|^2)$。

4　实例分析

为验证本文所提出的差别信息存储键树的有效性,选用 UCI 的 6 个数据集作为实验对象,在 Pentium dual-core 3.2GH 3G 内存,Microsoft Windows7 操作系统上进行实验对比,分别给出基于差别矩阵算法和差别矩阵存储键树算法中空间复杂度对比,其结果如表 3 所示。

表 3　差别矩阵和差别信息存储键树的实验对比结果

序号	数据集	论域数目	条件属性数	差别信息个数		属性个数总和	
				差别矩阵	存储键树	差别矩阵	存储键树
1	表 1	6	4	15	12	31	27
2	Lenses	24	5	155	15	535	32
3	imports-85	205	25	20879	2744	598162	19453
4	Hayeresroth	401	5	5661	16	21708	33
5	Anneal	798	38	127288	11341	1192558	33301
6	Car	1728	6	682721	63	2988153	127
7	Abalone	4177	8	7811786	131	59415995	326

由表 3 可得,差别信息存储键树中的差别信息个数与属性个数总和都要远远小于差别矩阵。特别是在论域数目较多,条件属性数较少的情况下这种差距更加显著。例如在数据集 Car 中,论域数目为 1728,条件属性个数为 6,在差别矩阵中含有的差别信息个数为 682721 个,属性个数总和为 2988153,而差别信息存储键树中只有 63 个差别信息个数,属性个数总和仅为 127。因此证明基于键树的压缩存储是可行的。

基于算法 2,选用 UCI 的 5 个数据集作为实验对象,分别给出基于差别矩阵约简算法和算法 2 的约简结果以及执行时间,其对比结果如表 4 所示。

表 4　差别矩阵和算法 2 的约简结果与执行时间对比

序号	数据集	论域数目	条件属性数	核	最终约简结果		执行时间/s	
					差别矩阵	算法 2	差别矩阵	算法 2
1	表 1	6	4	$\{bcd\}$	$\{bcd\}$	$\{bcd\}$	0	0
2	Balance scale	223	4	$\{abcd\}$	$\{abcd\}$	$\{abcd\}$	2.716	1.915
3	Hayeresroth	401	5	$\{a\}$	$\{a\}$	$\{a\}$	5.026	4.022
4	Car	1728	6	$\{abcdef\}$	$\{abcdef\}$	$\{abcdef\}$	23.254	20.210
5	Abalone	4177	8	$\varnothing$	$\{abcdefg\}$	$\{abcdefg\}$	98.910	88.785
6	Nursery	12960	8	$\{abcdefgh\}$	$\{abcdefgh\}$	$\{abcdefgh\}$	301.965	265.021

由表 4 可得,依据算法 2 得出的约简结果与基于差别矩阵的结果相同,但是在执行时间上,算法 2 明显优于差别矩阵,并且随着论域数目的增多,两者之间的差距更加显著。因而算法 2 有效地提高了属性约简的效率。

5　结束语

依据键树的思想,建立差别信息存储键树,利用差别信息的公共前缀,对差别元素进行了有效的压缩存储,删除了重复元素。最后根据差别信息存储键树设计出一属性约简算法,最终求出属性约简。

虽然利用键树在一定程度上消除了差别矩阵中出现的冗余元素,但是仍有部分无用的信息存在于差别信息存储键树上,在接下来的研究工作中,在考虑消除更多的冗余信息的同时,加入核在差别信息存储键树属性约简中的作用,从而实现进一步的属性约简效果。

致谢:感谢成都信息工程大学中青年学术带头人科研基金(J201609)对本文的资助

参考文献:

[1] Pawlak Z. International Journal of Computer and Information Science[J]. Roughsets,1982,11(5): 341-356.

[2] Liang J Y,Mi J R,Wei W,et al. An accelerator for attribute reduceion based on perspective of objects and attributes [J]. Knowledge-Based Systems, 2013,44:90-100.

[3] Miao D Q,Zhao Y,Yao Y Y,et al. Relative reduces in consistent and inconsistent decision tables of the Pawlak rough setmodel[J]. Information Sciences,2009,179(24):4140-4150.

[4] Shen Q,Jensen R. Selecting informative features with fuzzy-rough sets and its application for complex systems monitoring[J]. Pattern Recognition, 2014,37(7):1351-1363.

[5] 齐亚丽. 基于模糊粗糙集属性约简方法的研究[D]. 成都:电子科技大学,2016.

[6] 付志耀,高岭,孙骞,等. 基于粗糙集的漏洞属性约简及严重性评估[J]. 计算机研究与发展,2016(5):1009-1017.

[7] 苗夺谦,李道国. 粗糙集理论、算法与应用[M]. 北京:清华大学出版社,2008:66-67.

[8] Wong S K M,ZiarkoW. On optional decision rules indecision tables[J]. Bulletin of Polish Academy of Sciences,1985,33(11/12):693-696.

[9] 黄国顺. 保正域的决策粗糙集属性约简[J]. 计算机工程与应用,2016(2):165-169.

[10] 付志耀,高岭,孙骞,等. 基于粗糙集的漏洞属性约简及严重性评估[J]. 计算机研究与发展,2016(5):1009-1017.

[11] 严蔚敏,吴伟民. 数据结构(C 语言版)[M]. 北京:清华大学出版社,2011:247-251.

[12] 聂作先. 基于约简树的粗糙集最小约简算法[J]. 福建电脑,2007(9):10-11.

[13] 蒋瑜. 基于差别信息树的 rough set 属性约简算法[J]. 控制与决策,2015(8):1533-1534.

Attribute Reduction with Rough Set based on Digital Search Trees

ZHANG Wen-yang, JIANG Yu

(College of Software Engineering,Chengdu University of Information Technology,Chengdu 610225,China)

Abstract: Attribute reduction plays a key role in rough set. Discernibility matrix is efficient in finding out reducts. However, there are many redundancy non-empty elements in discernibility matrix. In order to eliminate the related redundancy and pointless elements, in this paper, a new method to store elements in discernibility matrix was proposed based on digital search trees. And an algorithm is presented to address Pawlak reduction based on this new structure. The experimentresults show that the proposed algorithm is efficient in finding out an attribute reduction.

Keywords: rough set theory; discernibility matrix; digital search trees; attribute reduction