第12卷 第24期 2012年8月
1671—1815(2012)24-0000-05

科 学 技 术 与 工 程
Science Technology and Engineering

Vol. 12 No. 24 Aug. 2012
ⓒ 2012 Sci. Tech. Engrg.

基于属性递减策略的属性约简递归算法

陈炎龙

(郑州牧业工程高等专科学校信息工程系, 郑州 450011)

摘 要 属性约简是粗糙集理论的核心研究内容之一。通过对某些现存属性约简算法分析,发现它们并不能有效地或正确地获取约简结果。为此,提出了一种基于属性递减策略的属性约简递归算法。该算法首先求出每个条件属性的依赖度。然后依次从条件属性集中减去依赖度较小的属性,并判断剩余属性集依赖度是否为1。如果是,则算法递归执行。最后把所获属性集并入约简集并求得核。该算法不仅能够快速计算出所有约简和核,而且运算简单、计算量较少,从而提高了算法效率。实例验证表明,该算法能更有效地对决策表进行约简,具有很强的实用性。

关键词 属性约简 粗糙集 依赖度 决策表

中图法分类号 TP301.4; **文献标志码** A

波兰华沙理工大学的科学家帕拉卡(Z. Pawlak)于1982年开创性地提出了粗糙集理论(rough sets)[1],该理论是一种处理含糊性(vagueness)、不精确(imprecision)和不确定性(uncertainty)问题的新型软计算(softcomputing)数学工具。粗糙集理论的主要优点在于不需要任何先验知识,就能有效地分析不精确、不一致、不完整等各种不完备的信息,并从中发现隐含的知识,揭示潜在的规律[2,3]。目前已在医学、金融、气象、图像处理、语音识别等领域取得了广泛应用[4]。

粗糙集理论的核心问题之一就是约简。如何在保持信息系统或知识库分类能力不变的条件下,删除其中不必要的、不重要或不相关的属性,是信息系统属性约简算法要解决的问题。目前主要的属性约简算法有:基于差别矩阵的属性约简算法,基于信息表示的属性约简算法,基于正区域的属性约简算法和基于粒计算的属性约简算法[5]。文献[6]提出了计算条件属性所有子集的信任函数和似真函数判断其值进行约简的方法,其缺点是计算量为2^n,很难应用于实际;文献[7]对以上缺点进行了改进,对属性依赖度从大到小进行排序,并以此顺序依次添加计算属性子集的信任函数,但该算法只适合相容决策表。文献[8]提出了比较依赖度相等的算法,但凭经验取排序的$n/2$进行添加属性,缺乏严谨性,应用于某些实例并不能获得正确的结果。文献[9]对相容和不相容决策表给出了通用算法,但需要计算U^*(不相容规则对象),增加了算法计算量。综上所述,文献[7,8]的属性添加法,只能求出一个包含约简的属性集,而不是约简;文献[9—11]的属性减去法,只是求核,没有求出所有的属性约简。为此,本文提出一种改进算法,该算法能递归地计算所有属性约简和核,而且运算简单、计算量较少。

2012年5月14日收到 河南省基础与前沿技术研究计划项目(102300410266)资助

作者简介:陈炎龙(1976—),男,汉族,山西晋城人,硕士,讲师,研究方向:信息系统。E-mail:chenyanlong0330@163.com。

1 相关概念

定义1 不可分辨关系[4] 论域U上的一簇等价关系S,若$P\subseteq S$,且$P\neq\Phi$,则$\cap P$(P中所有等价关系的交集)仍然是是U上的一个等价关系,称为P上的不可分辨关系$IND(P)$,常记为P。而且,

$$\forall x \in U,[x]_{IND(P)} = [x]_P = \bigcap_{\forall R\in P}[x]_R \quad (1)$$

定义2 决策系统[4] 决策系统是一个四元组$DT=(U,C\cup D,V,f)$。其中$U=\{x_1,x_2,\cdots,x_n\}$是非空论域,C称为条件属性集,D称为决策属性集,属性值域f: $U\rightarrow V$, V是一信息函数,它指定U中的

每一个对象在每个属性下的取值。

定义 3 下近似和正域[4] 知识库 $K=(U,S)$，其中 U 为论域，S 为论域 U 上的等价关系簇，则对 $\forall X\subseteq U$，U 上的一个等价关系 $R\in IND(K)$，定义子集 X 关于知识 R 的下近似和正域分别为：

$$\underline{R}(X)=\{x\mid(\forall x\in U)\wedge([x]_R\subseteq X)\} \tag{2}$$

$$pos_R=\overline{R}(X) \tag{3}$$

定义 4 依赖度[4] 知识库 $K=(U,S)$，其中 U 为论域，S 为论域 U 上的等价关系簇，则对 $\forall P,Q\in IND(K)$，定义

$$\gamma_P(Q)=k=\frac{\mid pos_P(Q)\mid}{\mid U\mid}=\frac{\mid\bigcup_{X\in U/Q}\overline{P}(X)\mid}{\mid U\mid} \tag{4}$$

为知识 Q 依赖于知识 P 依赖度。

定义 5 独立[4] 知识库 $K=(U,S)$，知识库的两个等价关系族 $P,Q\subseteq S$，对 $\forall R\in P$，若

$$pos_{IND(P)}(IND(Q))=pos_{IND(P-\{R\})}(IND(Q)) \tag{5}$$

成立，则称知识 R 为 P 中 Q 不必要的，否则称为必要的。如果每一个 $R\in P$，R 都是 P 中 Q 必要的，则 P 为 Q 独立的。

定义 6 相对约简[4] 知识库 $K=(U,S)$，知识库的两个等价关系族 $P,Q\subseteq S$，对 $\forall G\subseteq P$，若满足以下 2 个条件：

1) G 是 Q 独立的

2) $\mathrm{pos}_G(Q)=\mathrm{pos}_P(Q)$

则 G 是 P 的一个 Q 约简，记为 $G\in RED_Q(P)$。

定义 7 相对核[4] 知识库 $K=(U,S)$，知识库的两个等价关系族 $P,Q\subseteq S$，对 $\forall R\in P$，若满足

$$\mathrm{pos}_{IND(P-\{R\})}(IND(Q))\neq\mathrm{pos}_{IND(P)}(IND(Q)) \tag{6}$$

则 R 是 P 中 Q 必要的，P 中所有 Q 必要的知识组成的集合称为 P 的 Q 核。记为 $CORE_Q(P)$，$CORE_Q(P)=\cap RED_Q(P)$。

定理 1 对相容决策系统 $DT=(U,C\cup D)$，$U/R_D=\{D_1,D_2,(,Dr\}$，$B\subseteq C$，有以下等价关系[6]：

1) B 是 S 的相对约简。

2) $\sum_{j=1}^{r}Bel_B(D_j)=1$，且对任意的 $Q\subset B$，有 $\sum_{j=1}^{r}Bel_Q(D_j)<1$。

2 以往算法分析

首先简要介绍参考文献[7]中的描述：

定义：属性集合相对于决策属性的依赖度，等于属性集合中各属性相对于决策属性的依赖度之和[7]。此定义参看文献[12]或下面实例即知是错误的，例如属性集 ac 相对决策属性的依赖度是 1，但 a 相对决策属性的依赖度是 0.25，c 相对决策属性的依赖度是 0.125，很明显属性集 ac 的依赖度并不是 a 和 c 这两个属性的依赖度之和。

基于依赖度的属性约简算法的步骤[7]：

1) Core $=C$; // Core 表示核，C 表示条件属性集。

$$\gamma(C,D)=\frac{card(POS_C(D))}{card(U)};$$

For ($i=1$; $i<n$; $i++$) // n 表示属性值的个数。

$$\{M[i]=(\gamma\{C_i\},D)=\frac{card(POS_{\{ci\}}(D))}{card(U)};$$

/* Ci 表示第 i 个条件属性，$M[i]$ 中实际存放的是第 i 个条件属性，card (X) 表示 X 的基数。*/

}

2) 完成对条件属性的排序。

3) B =M[1]; //给 B 赋初值。

For (i = 1; i < n; i + +) //n 表示属性值的个数。

{ If ($\sum_{j=1}^{r}Bel_B(D_j)=1$)

{Core =Core∩B } ; //求出属性的核。

B =B∪{M [i + 1] } ; //修改 B

}

以下面例子验证该算法的不足：

某相容决策系统，条件属性集 C = {a,b,c,d}，决策属性集 D = {e}，具体如表 1 所示。

表 1　决策表

U	a	b	c	d	e
1	1	0	2	0	d_2
2	1	1	2	0	d_2
3	0	1	2	0	d_2
4	0	2	1	0	d_1
5	0	2	0	2	d_1
6	2	2	1	2	d_2
7	2	2	2	2	d_0
8	2	2	2	2	d_0

各条件属性和条件属性集对 U 的划分如下:

U/{a} = {{1,2},{3,4,5},{6,7,8}}

U/{b} = {{1},{2,3},{4,5,6,7,8}}

U/{c} = {{1,2,3,7,8},{4,6},{5}}

U/{d} = {{1,2,3,4},{5,6,7,8}}

U/{e} = {{1,2,3,6},{4,5},{7,8}}

U/{a,b} = {{1},{2},{3},{4,5},{6,7,8}}

U/{a,c} = {{1,2},{3},{4},{5},{6},{7,8}}

U/{a,d} = {{1,2},{3,4},{5},{6,7,8}}

U/{b,c} = {{1},{2,3},{4,6},{5},{7,8}}

U/{b,d} = {{1},{2,3},{4},{5,6,7,8}}

U/{c,d} = {{1,2,3},{4},{5},{6},{7,8}}

U/{a,b,c} = {{1},{2},{3},{4},{5},{6},{7,8}}

U/{a,b,d} = {{1},{2},{3},{4},{5},{6},{6,7,8}}

U/{a,c,d} = {{1,2},{3},{4},{5},{6},{7,8}}

U/{b,c,d} = {{1},{2,3},{4},{5},{6},{7,8}}

根据公式(4)可以得到各属性集依赖度如表 2:

表 2　属性集依赖度

属性集	依赖度	属性集	依赖度
a	0.25	bc	0.75
b	0.375	bd	0.5
c	0.125	cd	1
d	0	abc	1
ab	0.625	abd	0.625
ac	1	acd	1
ad	0.373	bcd	1

根据文献[7]中的算法,求出各条件属性依赖度,按其从大到小排序后为:b,a,c,d。

1) Core = C = {a,b,c,d};

2) B = M[1] = {b};　//给 B 赋初值。

3) For 语句

4) If 语句不满足,B = B∪{a} = {a,b},i + +,下一循环;

5) If 语句仍不满足,B = B∪{c} = {a,b,c},i + +,下一循环;

6) If 语句满足,Core = Core∩B = {a,b,c},

B = B∪{d} = {a,b,c,d},i + +,下一循环;

7)不满足 i < n;结束。

由以上执行结果可以看出,求得结果 Core = {a,b,c},但是采用经典算法求得的 Core = { c },这说明文献[7]中算法存在一定的错误。经过深入分析可知,文献[7]中算法的错误在于:把依赖度大的属性集中在一起,首先判断是该属性集否满足约简条件,如果满足就认为是一个约简,然后再求交集。其实并不是依赖度大的属性结合起来就一定是一个约简,上例中求得的只是包含约简的属性集,不是一个约简。因为定理 1 中不只是判断 $\sum_{j=1}^{r} Bel_B(D_j) = 1$,还要满足对任意的 $B' \subset B$,有 $\sum_{j=1}^{r} Bel_B(D_j) < 1$(独立性验证),才算是一个约简。同理,不难验证文献[8]中算法的错误性。

3　算法的改进

3.1　所提相关定理

为了对上述算法进行改进,本文提出 2 个定理,其中第 1 个定理适用于相容决策系统,第 2 个定理适用于所有决策系统。

定理 2　对于相容决策系统 DT = (U,C∪D,,f),$U/R_D = \{D_1, D_2, (, Dr\}$,B ⊆ C,有以下等价关系:

1)B 是 S 的相对约简。

2) $\gamma_B(D) = 1$,且对任意的 $Q \subset B$,有 $\gamma_Q(D) < 1$。

证明:由定义 4 可知:

$$\gamma_B(D) = k = \frac{|pos_B(D)|}{|U|} = \frac{|\bigcup_{X \in U/D} \overline{B}(X)|}{|U|}。$$

由于信任函数[6]

$$\sum_{j=1}^{r} Bel_B(D_j) = \sum_{j=1}^{r} \frac{|\overline{B}(D_j)|}{|U|} = \bigcup_{X \in U/D} \frac{|B(X)|}{|U|} = \frac{|\bigcup_{X \in U/D} \overline{B}(X)|}{|U|}。$$

因此,$\sum_{j=1}^{r} Bel_B(D_j) = \gamma_B(D)$。

又由定理 1 可知：$\sum_{j=1}^{r} Bel_B(D_j) = 1$，且对任意的 $Q \subset B$，有 $\sum_{j=1}^{r} Bel_Q(D_j) < 1$，因此 $\gamma_Q(D) < 1$。定理得证。

定理 3　对相容或不相容决策系统 $DT = (U, C \cup D, V, f)$，$U/R_D = \{D_1, D_2, (, Dr\}$，$B \subseteq C$，有以下等价关系：

1）B 是 S 的相对约简。

2）$\gamma_B(D) = \gamma_C(D)$，且对任意的 $Q \subset B$，有 $\gamma_Q(D) < \gamma_C(D)$。

证明：由定义 4 可知：

$$\gamma_B(D) = k = \frac{|pos_B(D)|}{|U|} = \frac{|\bigcup_{X \in U/D} \overline{B}(X)|}{|U|}。$$

由定义 6 可知：

1）B 是 C 独立的。

2）$POS_B(D) = POS_C(D)$，则 B 是 C 的一个 D 约简。

若 $\gamma_B(D) = \frac{|pos_B(D)|}{|U|} = \gamma_C(D) = \frac{|pos_C(D)|}{|U|}$ 则

$pos_B(D) = pos_C(D)$. 又根据定理 1 可知：对任意的 $Q \subset B$，有 $\gamma_Q(D) < \gamma_C(D)$，保证了 B 是 C 独立的，因此得证。

3.2　算法改进思想

由文献[7]可知：依赖度越大的属性，将以更大的概率接近于 1，因此对依赖度从大到小排序，效率更高。但使用依次选择依赖度大的属性进入约简属性集的方法并不可取，因为这样做增加了算法的复杂性，而且仍然需要验证所有的属性集组合，且不能保证得到属性集的独立性。然而，通过仔细分析可知，如果采用属性减去法，就可以减少很多不必要的判断，从而大大减少算法时间复杂度。为此，本文算法改进思想如下：首先求出每个条件属性的依赖度，然后依次从条件属性集中减去依赖度较小的属性，并判断剩余属性集依赖度是否为 1，如果是，则算法递归执行，直到其子集依赖度不为 1。最后把所获属性集并入约简集 Reduct，并求得核 Core = ∩Reduct。

3.3　算法描述

```
Input：信息系统 S = <U, C(D,V,f>
Output：信息系统 S 的属性约简集 Reduct 和核 Rore.
Step1：Reduct = Φ; //Reduct 表示所有属性约简集合。
Core = Φ; //Core 表示核。
                γ(C,D) = card(POS_C(D)) / card(U);
For ( i = 1; i <=n; i++ ) //n 表示属性值的个数。
        {M[i] = (γ{C_i} , D) = card(POS_{c_i}(D)) / card(U);
/* C_i 表示第 i 个条件属性，M[i]中实际存放的是第 i 个条件属性，card (X)表示 X 的基数。 */
                }
Step2：Reduction(C,D,n)
  { flag = 0;
    For ( i=n; i>0; i--)
    { X = C - C[i]; //减去 C 集合中第 i 个属性
    If( γ_X(D) = =1) //对不相容决策表，1 改为 γ_C(D)
      { flag = 1;
        Reduction(X,D,n-1);
    //判断其子集是否满足，即验证独立性
      }
    }
    If( flag = =0)
    Reduct = Reduct ∪ {C}; //把约简 C 并入约简集
  }
Step3：Core = ∩Reduct。//核等于所有约简的交集。
```

3.4　算法分析

通过仔细分析，本文提出的改进算法采用递归的形式，能够快速求出所有的约简，时间复杂度为为 $O(|C|^2|U|)$，虽然时间复杂度和文献[13—15]的一样。但是，由于本文算法在约简过程中采用属性减去策略，减少了很多不必要的判断，这样就省略了很多不必要的计算，而且每次搜索的空间还逐步减少，这就大大节省了算法运行时间，提高了算法效率。

4　实例验证

以表 1 为实验数据，在具有操作系统为 Microsoft Windows XP Professional (SP2)、CPU 规格为 Intel(R) Celeron(R) CPU 2.40GHz、内存 512M、硬盘 80G 的计算机上分别实现了本文算法和文献[14]中的算法。其中本文算法执行过程如图 1

所示。

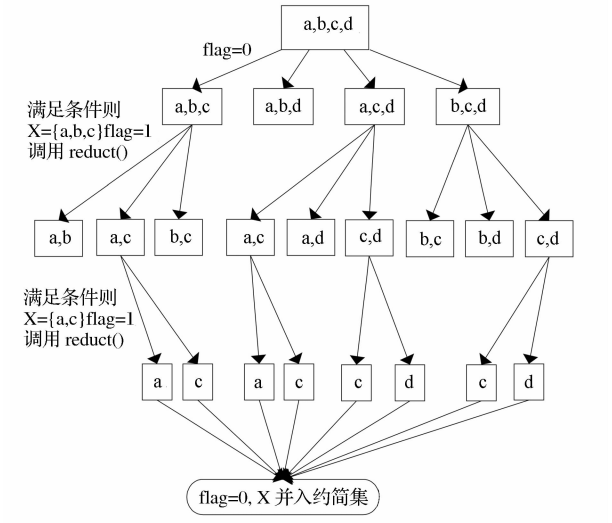

图 1　算法过程

通过上图递归调用,本文算法约简结果 Reduct = {{a,c},{c,d}},Core = ∩Reduct = {c},这与文献[14]结果相同。而在耗时方面,对这两个算法整个执行过程进行了计时操作,结果表明:本文算法耗时 237 ms,而文献[14]算法耗时 293 ms。这个实例验证表明:本文算法能够获得与经典算法相同的约简结果,而且耗时较少,这说明本文算法是具有一定实用价值的。

5　结束语

本文在研究现存属性约简算法的基础上,发现属性依赖度大的不一定就在属性约简中,单一属性依赖度为 0 的属性不能轻易舍弃,考察属性集依赖度比单一属性依赖度更加可信,同时在属性集满足依赖度条件的情况下,还要考察其子集是否满足条件,即独立性验证。为此,提出了一种新的属性约简递归算法,该算法不仅能够快速计算出所有约简和核,而且运算简单、计算量较少,从而提高了算法效率。通过实例验证,该算法具有很强的实用性。

参　考　文　献

1　Pawlak Z. Rough set. International Journal of Computer and Information Sciences,1982; 11(15):341—356

2　Pawlak Z. Rough Set. Communications of ACM,1995; 38(11): 89—95

3　Pawlak Z. Vagueness and uncertainty-a rough set perspective. Computational Intelligence. 1995;11(2):227—232

4　苗夺谦,李道国. 粗糙集理论、算法与应用. 北京:清华大学出版社,2008

5　丁　浩,丁世飞,胡立花. 基于粗糙集的属性约简研究进展. 计算机工程与科学,2010;32(6):92—94

6　Wu Weizhi, Zhang Mei, Li Huaizu, *et al.* Knowledge reduction in random information systems via Dempster Shafer theory of evidence. Information Science, 2005; 174:143—164

7　张学渊,高社生,胡　攀. 属性约简的新算法. 计算机仿真,2009;26(10):195—197

8　蔡　莉,胡学钢. 一种基于依赖度的决策表属性约简算法. 安庆师范学院学报(自然科学版),2008;14(1):37—39

9　钟　瑜. 基于依赖度的属性约简算法的注记. 中国集体经济,2010;(7):85—87

10　周　勇,杨兴江,徐　扬. 属性约简的依赖度算法研究. 计算机工程与应用,2004;40(4):78—79

11　张文东,李明壮,石小艳. 基于粗糙集理论的属性约简算法. 计算机工程与设计,2008;29(22):5795—5797

12　孟庆全,梅灿华. 一种新的属性集依赖度. 计算机应用,2007;27(7):1748—1750

13　刘少辉,盛秋戬,吴　斌,等. Rough 集高效算法的研究. 计算机学报, 2003; 26(3):524—529

14　徐章艳,刘作鹏,杨炳儒,等. 一个复杂度为 max(O(|C||U|),O(|C|²|U|))的快速属性约简算法. 计算机学报,2006;29(3):391—399

15　胡　彧,白琳林. 快速的属性约简算法. 计算机工程与应用,2009;45 (28):133—136

Attribute Reduction Recursion Algorithm Based on Attribute Diminishing Strategy

CHEN Yan-long

(Department of Information Engineering, Zhengzhou College of Animal Husbandry Engineering, Zhengzhou 450011, P. R. China)

[**Abstract**] Attribute reduction is one of core research subjects in rough set theory. By means of analyzing some existing attribute reduction algorithms, it found that they cannot effectively or correctly get reduction results. An attribute reduction recursion algorithm based on attribute diminishing strategy was presented. The proposed attribute reduction algorithm firstly calculate dependency degree of every condition attribute; and then in turn subtract condition attributes with smaller dependency degree; subsequently, calculate dependency degree of remaining attributes set and decide whether the dependency degree is 1, if it is, the algorithm is recursively implemented. Finally, the all won attributes set were merged into the attribute reduction set and the core attributes were obtained. The proposed attribute reduction algorithm can not only fast calculate out all attribute reduction and core attributes, but also operates simply and has less computation, so that efficiency of the algorithm is improved. The experiment shows that the proposed attribute reduction algorithm can more effectively reduct decision table and have stronger practicability.

[**Key words**] attribute reduction rough set dependency degree decision table