一种基于重构相容决策表的属性约简算法

赵洪波[1] 江 峰[1] 曾惠芬[2]
(青岛科技大学信息科学技术学院 青岛 266061)[1] (九江职业技术学院 九江 332007)[2]

摘 要 基于正区域的属性约简是目前最常用的一类约简算法。现实中的决策表有可能存在不一致的对象。另外,在约简过程中随着属性个数的减少,也有可能产生新的不一致对象。对于基于正区域的约简算法来说,不一致的对象并没有提供任何有用的信息,删除不一致的对象不会改变正区域的计算结果以及最终的约简结果,而且可以显著提高算法的效率。然而现有的基于正区域的约简算法并没有考虑到这个问题,它们采用论域中的所有对象来计算正区域并得出约简结果。针对这一问题,定义了重构相容决策表和重构相容决策子表的概念。引入这两个概念的目的是在约简过程中删除初始决策表中的不一致对象,从而获得一个相容决策表。借助于这两个概念,提出了一种新的基于正区域的属性约简算法。在真实数据集上的实验表明,与传统的算法相比,该算法能够获得较小的约简结果和较高的分类精度,并且具有相对较低的时间复杂度。

关键词 粗糙集,正区域,属性约简,不相容决策表,重构相容决策表
中图法分类号 TP39 **文献标识码** A

New Attribute Reduction Algorithm Based on Reconstructed Consistent Decision Table

ZHAO Hong-bo[1] JIANG Feng[1] ZENG Hui-fen[2]
(College of Information Science and Technology, Qingdao University of Science and Technology, Qingdao 266061, China)[1]
(Jiujiang Vocational and Technical College, Jiujiang 332007, China)[2]

Abstract By now, the positive-based attribute reduction is one of the most popular algorithms for attribute reduction. Some inconsistent objects may be present in the real world decision tables. And with the decrease of the number of attributes during the process of reduction, some new inconsistent objects may also occur in the decision tables. For a positive-based attribute reduction algorithm, the inconsistent objects can not provide any useful information. Therefore, deleting those objects from the decision table will not change the results of positive regions, and the final result of reduction. Moreover, this operation may improve the efficiency of the algorithm obviously. However, most of the current positive-based attribute reduction algorithms have not concerned this problem. They use all objects in the domain to calculate the positive regions and obtain the results of reduction. To solve this problem, we defined the notions of reconstructing consistent decision table and reconstructing consistent decision sub-table. The aim for introducing the two notions is to delete the inconsistent objects in the original decision table and obtain a consistent decision table during the process of reduction. By virtue of the two notions, we proposed a novel positive-based attribute reduction algorithm. The experimental results on real datasets demonstrate that our algorithm can obtain smaller reducts and higher classification accuracies than the traditional algorithms. And the time complexity of our algorithm is relatively low.

Keywords Rough sets, Positive region, Attribute reduction, Inconsistent decision table, Reconstruction consistent decision table

1 前言

粗糙集理论是由 Pawlak 在 1982 年提出的[5],是一种刻画不完整性和不确定性的数学工具,能有效地分析和处理不精确、不一致、不完整等各种不完备信息,并从中发现隐含的知识[1,2]。

属性约简是粗糙集的核心内容。由于属性约简的结果直接影响到约简后规则提取的繁简和性能,人们往往期望找到最小约简。然而,已经证明寻找最小约简是一个 NP-hard 问题[1,3]。但是人们仍然期望通过改进约简算法,在代价较低的情况下找到尽可能小的约简。目前,属性约简算法大体可分为 3 类:(1)基于正区域的方法[4,9,10];(2)基于熵的方法[3,11];(3)基于分明矩阵的方法[7]。在基于正区域的算法中,正区域的计算起到至关重要的作用。然而,现有的基于正

到稿日期:2011-06-11 返修日期:2011-09-17 本文受国家自然科学基金项目(60802042),山东省自然科学基金项目(ZR2011FQ005, ZR2010FQ027),山东省高等学校科技计划项目(J11LG05)资助。

赵洪波 硕士生,主要研究方向为数据挖掘、粗糙集理论;江 峰 博士,副教授,CCF 会员,主要研究方向为人工智能、粗糙集理论等,E-mail:jiangkong@163.net(通信作者)。

区域的算法通常都是针对决策表中所有的对象来计算正区域并求出约简，并没有考虑到决策表中可能包含的不一致对象，而这些不一致对象在约简过程中并不会影响正区域的变化。因此，在约简过程中，有必要删除决策表中的不一致对象，以在一个更小的对象论域上计算正区域和约简。

针对基于正区域的约简算法存在的问题，本文提出了重构相容决策表和重构相容决策子表的概念。基于这两个概念，我们提出了一种新的属性约简算法。算法的主要思想是：删除初始决策表中的不一致对象，得到重构相容决策表，并在约简过程中不断删除新产生的不一致对象，从而得到多个重构相容决策子表。以重构相容决策表的核为基础，逐步添加后续重构相容决策子表中的重要性最大的属性，直至得到一个约简。本文的算法为基于正区域的属性约简提供了新的思路。通过删除初始决策表中的不一致信息和约简过程中新产生的不一致信息，可以有效减少计算量，从而在较短时间内得到一个理想的约简结果。

2 粗糙集的基本知识

在粗糙集中，信息表是一个四元组 $IS=(U,A,V,f)$。其中 U 和 A 分别表示对象集和属性集；V 是所有属性论域的并，即 $V=\bigcup_{a\in A}V_a$，其中 V_a 为属性 a 的值域；$f:U\times A\to V$ 是一个信息函数，使得对任意 $a\in A$ 以及 $x\in U$，$f(x,a)\in V_a$[2]。

进一步，属性集 A 又可以划分为两个不相交的子集——条件属性集 C 和决策属性集 D。这种特殊的信息表称为决策表，简记为 $DT=(U,C,D,V,f)$。

给定决策表 $DT=(U,C,D,V,f)$，对于任意 $B\subseteq C\cup D$，定义由 B 所决定的一个不可分辨关系 $IND(B)$ 为 $IND(B)=\{(x,y)\in U\times U:\forall a\in B(f(x,a)=f(y,a))\}$[1,2]。可以证明，$IND(B)$ 是 U 上的一个等价关系。$IND(B)$ 将 U 划分成多个等价类，所有这些等价类的集合就构成 U 的一个划分，记为 $U/IND(B)$。

定义 1(不相容决策表) 给定决策表 $DT=(U,C,D,V,f)$，如果存在 $x,y\in U$ 使得 $(x,y)\in IND(C)$，但 $(x,y)\notin IND(D)$，那么称 DT 为一个不相容决策表。

定义 2(正区域) 给定决策表 $DT=(U,C,D,V,f)$，对任意 $B\subseteq C$，定义 D 的 B-正区域 $Pos_B(D)$ 为[1,2]

$$Pos_B(D)=\bigcup\{Y\mid Y\subseteq X,X\in U/IND(D),Y\in U/IND(B)\}$$

定义 3(属性依赖度) 给定决策表 $DT=(U,C,D,V,f)$，对于任意 $B\subseteq C$，D 对 B 的依赖度定义为[1,2]

$$\gamma_B(D)=|Pos_B(D)|/|U|$$

定义 4(属性重要性) 给定决策表 $DT=(U,C,D,V,f)$，对于任意 $B\subseteq C$ 和 $a\in B$，a 在 B 中相对于 D 的重要性定义为[1]

$$SGF(a,B,D)=\gamma_B(D)-\gamma_{B-\{a\}}(D)$$

定义 5(基于正区域的属性约简) 给定决策表 $DT=(U,C,D,V,f)$，对任意 $B\subseteq C$，如果 $Pos_B(D)=Pos_C(D)$，则称 B 为属性集 C 相对于决策属性集 D 的一个约简集。

定义 6(核) 给定决策表 $DT=(U,C,D,V,f)$，假设 C 的所有约简为 $B_1,B_2,\cdots,B_n$，则 C 相对于 D 的核 $Core_D(C)$ 定义为

$$Core_D(C)=\bigcap_{i=1}^{n}B_i$$

3 约简算法

由定义 5 知，基于正区域的属性约简主要通过计算 D 的 B-正区域 $Pos_B(D)$ 是否与原正区域 $Pos_C(D)$ 相等，来判断 B 是否为 C 的约简。而决策表通常又分为相容决策表和不相容决策表，不相容决策表中存在着不一致的对象。如果在约简过程中删除不相容决策表中的不一致对象，将不会导致该决策表中正区域发生变化，因而也不会影响最终的约简结果，但却可以有效提高约简算法的效率。

例 1 给定决策表 $DT=(U,C,D,V,f)$，其中 $U=\{u_1,u_2,\cdots,u_{14}\}$，$C=\{a_1,a_2,a_3,a_4\}$，$D=\{d\}$，如表 1 所列。

表 1 决策表 DT

U	a_1	a_2	a_3	a_4	d
u_1	3	1	2	3	2
u_2	2	1	1	2	2
u_3	3	2	3	3	2
u_4	3	3	2	2	2
u_5	1	3	3	1	1
u_6	2	3	1	3	1
u_7	2	1	2	1	2
u_8	3	2	1	1	1
u_9	2	3	3	2	1
u_{10}	2	3	3	2	2

在表 1 中，对象 u_9 和 u_{10} 是不一致的。这是因为在 C 下它们属于同一等价类，但在 d 下它们却不属于同一等价类。显然，在 C 的任意子集下，u_9 和 u_{10} 仍然属于同一等价类。因此，在判断 C 的某个子集 B 是否为约简集时，判断 u_9 和 u_{10} 是否属于正区域 $Pos_B(D)$ 是没有必要的，因为它们在任何情况下都不可能属于 $Pos_B(D)$。而现有的基于正区域的约简算法在约简过程中，每当判断 C 的某一子集 B 是否为约简时，都要重新计算一次 u_9 和 u_{10} 是否属于 $Pos_B(D)$，在不一致信息上浪费了很多时间，从而降低了算法的效率。

现实中的决策表很有可能存在不一致的对象。另外，在约简过程中，由于属性个数逐步减少，也有可能产生新的不一致对象。例如，如果从表 1 中约简去属性 a_3 和 a_4 之后，原来一致的对象 u_3 和 u_8 也变得不一致了。针对不相容的决策表，有必要提出一种新的属性约简算法，在约简过程中，该算法能够不断删除那些新出现的不一致对象，在保证约简质量的前提下，尽可能地减少参与约简计算的对象个数，从而提高算法的执行效率。

为了表示从不相容决策表中删除不一致对象之后所得的新的相容决策表，我们提出重构相容决策表和重构相容决策子表的概念。

定义 7(重构相容决策表) 给定决策表 $DT=(U,C,D,V,f)$，如果 DT 中存在不一致的对象，则将 $RDT=(U\_C,C,D,V,f)$ 称为 DT 的重构相容决策表，其中 U_clear 为论域 U 在 C 下的相容对象子集，即 $U\_clear=\{x\in U:\forall y\in U((x,y)\in IND(C)\to(x,y)\in IND(D))\}$。

定义 8(重构相容决策子表) 给定决策表 $DT=(U,C,D,V,f)$，令 $DT\_sub=(U,B,D,V_B,f_B)$ 为 DT 的一个子决策

表，其中 $B \sqsubset C$。假设 $RDT\_sub=(U\_B,B,D,V_B,f_B)$ 为 DT_sub 的重构相容决策表，则称 RDT_sub 为 DT 的重构相容决策子表。

定义 9(重构核) 给定决策表 $DT=(U,C,D,V,f)$ 以及 DT 的重构相容决策表 $RDT=(U\_clear,C,D,V,f)$。令 $Core_D(C)$ 为在 DT 中计算得到的 C 相对于 D 的核，并且令 $RCore_D(C)$ 为在 RDT 中计算得到的 C 相对于 D 的核，则将 $RCore_D(C)$ 称为 $Core_D(C)$ 的重构核。

定义 10(重构子核) 给定决策表 $DT=(U,C,D,V,f)$ 以及 DT 的重构相容决策子表 $RDT\_sub=(U\_clear,B,D,V_B,f_B)$。令 $Core_D(C)$ 为在 DT 中计算得到的 C 相对于 D 的核，并且令 $RCore\_sub_D(B)$ 为在 RDT_sub 中计算得到的 B 相对于 D 的核，则将 $RCore\_sub_D(B)$ 称为 $Core_D(C)$ 的重构子核。

下面给出基于重构相容决策表的属性约简算法。

算法 1

输入：决策表 DT=(U,C,D,V,f)，其中 C={a_1,a_2,…,a_m}，D={d}

输出：约简集 Reduct

步骤 1 在决策表 DT 中计算 D 对于 C 中每个属性 a 的依赖度 $\gamma_{\{a\}}(D)$。

步骤 2 删除 DT 中的不一致对象，得到 DT 的重构相容决策表 RDT=(U_C,C,D,V,f)。

步骤 3 (1)在 RDT 中计算 C 相对于 D 的核 $RCore_D(C)$

(1.1)如果 $RCore_D(C)$ 为空，则将 C 中 $\gamma_{\{a\}}(D)$ 值最大的属性 a 添加到 Reduct 中；

(1.2)如果 $RCore_D(C)$ 不为空，则令 Reduct=$RCore_D(C)$；

(2)在 RDT 中计算 $Pos_{Reduct}(D)$，并判断 $Pos_{Reduct}(D)$ 是否等于 $Pos_C(D)$

(2.1)如果相等，转步骤 5；

(2.2)如果不相等，转步骤 4。

步骤 4 (1)令 C_Remain=C−Reduct，并得到 RDT 的重构相容决策子表 RDT_sub=(U_C_Remain,C_Remain,D,V_Remain,f_Remain)。

(2)在子表 RDT_sub 中计算 C_Remain 相对于 D 的核 $RCore\_sub_D(C\_Remain)$，即 $RCore_D(C)$ 的重构子核。

(2.1)如果 $RCore\_sub_D(C\_Remain)$ 为空，则将 C_Remain 中 $\gamma_{\{a\}}(D)$ 值最大的属性 a 添加到 Reduct；

(2.2)如果 $RCore\_sub_D(C\_Remain)$ 不为空，则对任意 $a \in RCore\_sub_D(C\_Remain)$，在 RDT_sub 中计算 a 的重要性 SGF(a,C_Remain,D)，并选择重要性最大的属性添加到 Reduct 中。

(3)在 RDT 中计算 $Pos_{Reduct}(D)$，并判断 $Pos_{Reduct}(D)$ 是否等于 $Pos_C(D)$。

(3.1)如果相等，转步骤 5；

(3.2)如果不相等，重新执行步骤 4。

步骤 5 (1)得到 RDT 的重构相容决策子表 RDT_Reduct=(U_Reduct,Reduct,D,V_Reduct,f_Reduct)；

(2)在子表 RDT_Reduct 上执行步骤 3 和步骤 4，得到新的约简集 Child_Reduct，并判断 Child_Reduct 是否等于 Reduct

(2.1)如果相等则转步骤 6；

(2.2)如果相不等，则令 Reduct=Child_Reduct，继续执行步骤 5。

步骤 6 输出约简集 Reduct。

算法 1 中采用了一种预先对 U 中对象进行基数排序，然后求划分 $U/IND(B)$ 的方法[16]，从而使得计算划分 $U/IND(B)$ 的时间复杂度降低为 $O(|B|\times|U|)$。在最坏的情况下，算法 1 的时间复杂度为 $O(m^3\times n)$，空间复杂度为 $O(m\times n)$，其中 m 和 n 分别为 C 与 U 的势。

传统的基于正区域的约简算法的时间复杂度大多为 $O(m\times n^2)$ 或者 $O(m^2\times n\times \log n)$[4,9,10]。由于在大多数情况下，$m$ 的值要远小于 n，因此与传统算法相比，我们的算法具有相对较低的时间复杂度。

4 实验

我们利用 C# 语言实现了算法 1，并且在 5 个 UCI 数据集上测试了算法 1 的性能[15]：Vote，Mushroom，Lymphography，SOYBEAN-Small，Zoo(注：完整的 Vote 数据集包括 435 个实例，本文使用的是 Vote 的一个子集，包含 300 个从 Vote 中随机抽取的实例)。

算法 1 在上述 5 个数据集上得到的属性约简结果如表 2 所列。

表 2 属性约简结果

数据集	正区域中实例个数	约简集中属性个数
Vote	300	8
Mushroom	3764	4
Lymphography	148	7
SOYBEAN-Small	47	2
Zoo	101	5

另外，将算法 1 与文献中 5 个具有代表性的约简算法进行比较：(1)基于正区域的算法(POSAR)[4,9,10]；(2) 基于条件熵的算法(CEAR)[3,11]；(3) 基于分明矩阵的算法(DISMAR)[7]；(4) 基于遗传算法的算法(GAAR)[8,12,13]；(5) 基于粒子群优化和粗糙集的算法(PSORSAR)[6]。上述 5 个算法的实验结果可参考 Wang 的论文[6]。

表 3 给出了不同算法得到的约简的大小。

表 3 约简结果比较

数据集	约简中所包括的属性个数					
	算法 1	POSAR	CEAR	DISMAR	GAAR	PSORSAR
Vote	8	9	11	8	9	8
Mushroom	4	5	5	6	5	4
Lymphography	7	6	8	7	8	7
SOYBEAN-Small	2	2	2	2	6	2
Zoo	5	5	10	5	6	5

从表 3 可以看出，与其他算法相比，算法 1 在大部分数据集上都能获得较小的约简结果。除了 Lymphography 数据集之外，在其他数据集上，本文的算法都能获得最小约简。

另外，还比较了不同算法所获得的约简的分类精度。我们借鉴 Wang 的实验方法[6]，利用 RSES 软件从约简之后的训练集中提取决策规则[14]，并且利用这些规则来对测试集进行分类。具体而言，我们采用 10-折交叉验证的方式来评估每个算法所生成的约简的分类精度。

首先，将一个完整的数据集分成 10 个大小相等且互不相交的子集；

其次，反复选择其中 9 个子集组成训练集，剩下的 1 个子集组成测试集，总共获得 10 组训练和测试集，其中每个子集都被当作一次测试集；

第三，在这10组训练和测试集上分别评估约简算法的分类精度，并计算平均值。针对每一组训练和测试集，首先根据约简算法所生成的约简来删除训练集中的冗余属性；然后利用RSES软件中提供的LEM2算法来提取决策规则；最后，利用这些规则来对测试集进行分类。RSES中相关的参数设置如下：Shortening ratio：0.9；Conflicts resolve by：STANDARD VOTING[14]。

表4给出了不同算法所得到的约简的分类精度。

表4 分类精度比较

数据集	分类精度(%)					
	算法1	POSAR	CEAR	DISMAR	GAAR	PSORSAR
Vote	95.2	94.33	92.33	93.67	94.0	95.33
Mushroom	100	100	90.83	100	100	99.7
Lymphography	79.1	85.71	72.14	74.29	70.0	75.71
SOYBEAN-Small	100	100	100	100	97.5	100
Zoo	97.9	96.0	94.0	94.0	92.0	96.0

从表4可以看出，算法1的分类性能在大多数情况下都优于或等于其他算法。除了在Vote数据集上低于PSORSAR算法，在Lymphography数据上低于POSAR算法之外，我们的算法都具有最高的分类精度。因此，从总体上来看，我们的算法在这些数据集上具有更好的分类能力。

结束语 传统的基于正区域的约简算法往往忽略了对决策表中的不一致对象以及属性约简过程中所产生的不一致对象的处理。通过分析，我们发现删除这些不一致对象不会影响正区域的计算和属性约简的结果，但是不一致对象的存在却会降低算法的执行效率。因此，本文提出了一种基于重构相容决策表的属性约简算法。在本算法中，首先删除决策表和决策子表中的不一致对象，再通过计算正区域来得到约简，从而提高了算法的执行效率。在UCI数据集上的实验结果表明，我们的算法能得到较小的属性约简集，并且约简后所生成的规则具有较高的分类精度。

参考文献

[1] 王国胤. Rough集理论与知识获取[M]. 西安：西安交通大学出版社，2001

[2] 刘清. Rough集及Rough推理[M]. 北京：科学出版社，2005

[3] 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

[4] 刘少辉，盛秋戬，吴斌，等. Rough集高效算法的研究[J]. 计算机学报，2003，26(5)：525-529

[5] Pawlak Z. Rough sets[J]. Int. Journal of Computer and Information Sciences，1982，11(5)：341-356

[6] Wang X Y，Yang J，Teng X L，et al. Feature Selection Based on Rough Sets and Particle Swarm Optimization[J]. Pattern Recognition Letters，2007，28(4)：459-471

[7] Hu K Y，Lu Y C，Shi C Y. Feature ranking in rough sets[J]. AI Communication，2003，16(1)：41-50

[8] Wroblewski J. Finding minimal reducts using genetic algorithms [C]// Proc. of the 2nd Annual Joint Conf. on Information Sciences. 1995，186-189

[9] Jensen R，Shen Q. Finding rough set reducts with ant colony optimization[C]// Proc. of the 2003 UK Workshop on Computational Intelligence. 2003：15-22

[10] Hu X H. Knowledge discovery in databases：an attribute-oriented rough set approach[D]. Regina University，1995

[11] Wang G Y，Zhao J. Theoretical study on attribute reduction of rough set theory：comparison of algebra and information views [C]// Proc. of the 3rd IEEE Int. Conf. on Cognitive Informatics. 2004：148-155

[12] Bazan J. A comparison of dynamic and non-dynamic rough set methods for extracting laws from decision table[M]. Rough Sets in Knowledge Discovery，1998：321-365

[13] Bazan J，Nguyen H S，Nguyen S H，et al. Rough set algorithms in classification problem[M]. Rough Set Methods and Applications，2000：49-88

[14] Skowron A，et al. RSES 2.2 User's Guide[M]. Warsaw University，2005

[15] Bay S D. The UCI KDD repository[EB/OL]. http://kdd.ics.uci.edu，1999

[16] 徐章艳，刘作鹏，杨炳儒，等. 一个复杂度为max(O(|C‖U|)，O(|C|^2|U/C|))的快速属性约简算法[J]. 计算机学报，2006，29(3)：391-399

(上接第146页)

[9] Hardavellas N，Ferdman M. Reactive NUCA：near-optimal block placement and replication in distributed caches[C]// Proceedings of the 36th Annual International Symposium on Computer Architecture(ISCA '09). Austin，TX，USA，2009：184-195

[10] Xie T. SEA：A Striping-based Energy-aware Strategy for Data Placement in RAID-Structured Storage Systems [J]. IEEE Transactions on Computers，2008，57：748-761

[11] Cope J M，Trebon N，Tufo H M，et al. Robust data placement in urgent computing environments[C]// IEEE International Symposium on Parallel & Distributed Processing (IPDPS 2009). 2009：1-13

[12] Xiao Nong，Chen Tao，Liu Fang. RSEDP：An Effective Hybrid Data Placement Algorithm for Large-scale Storage Systems[J]. 2009，55(1)：103-122

[13] Ghemawat S，Gobioff H，Leung S-T. The Google file system[J]. SIGOPS Oper. Syst. Rev，2003，37：29-43

[14] Hadoop[EB/OL]. http://hadoop.apache.org/2011-03-25

[15] Amazon Elastic Computing Cloud[EB/OL]. http://aws.amazon.com/ec2/，2011-03-30

[16] Henzinger M，Raghavan P，Rajagopalan S. Computing on Data Streams[R]. TR-1998-011. Digital Equipment Corp，Aug 1998

[17] Barroso L A，Dean J，Hölzle U. Web search for a planet：The Google cluster architecture[J]. IEEE Micro，2003，23(2)：22-28

[18] Lenk A，Klems M，Nimis J，et al. What's inside the cloud? An architectural map of the Cloud landscape[C]// Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing. May 2009：23-31

[19] 陈超，蒋建春，丁治明. 基于时序片段评价的数据分配算法[J]. 计算机研究与发展，2010，47(z1)：385-393

[20] 郑湃，崔立真，王海洋，等. 云计算环境下面向数据密集型应用的数据布局策略与方法[J]. 计算机学报，2010(8)：1472-1480

[21] 文明波，丁治明. 适用于云计算的面向查询数据库数据分布策略[J]. 计算机科学，2010，37(9)：168-172

[22] 王道祥. 基于分布式数据库的综合信息系统关键技术研究[D]. 长沙：国防科学技术大学，2006