

基于值约简和决策树的最简规则提取算法

罗秋瑾, 陈世联

(昆明理工大学理学院, 云南昆明 650093)

(qiujiinluo@sina.com)

摘要:粗糙集理论中的值约简和数据挖掘领域中的决策树都是有效的分类方法,但二者都有其局限性。将这两种方法结合起来,生成一种新的基于值核的极小化方法对决策树进行修剪,提出了约简规则的判定准则,缩小了约简的范围,最后再对生成的规则进行极大化处理,以保证规则覆盖信息的一致性,实验验证了该算法的有效性。

关键词:粗糙集;数据挖掘;决策树;值约简;分类规则

中图分类号: TP311.13 **文献标识码:** A

Algorithm based on value reduction and decision tree to generate minimal rules

LUO Qiu-jin, CHEN Shi-lian

(School of Science, Kunming University of Science and Technology, Kunming Yunnan 650093, China)

Abstract: Value reduction in rough set theory and decision tree in data mining are effectively used in the classification, but each of them has shortcomings. Those two methods were combined to generate a new minimal method based on value core to pollard the decision tree. Then judgmental standards of rule reduction were proposed to decrease the quantity of reduced rules. In the end, the classing rules were dealt with by maximal method to ensure the consistency of the knowledge contained by the rules. The algorithm is efficient which was proved by the experiment.

Key words: rough set; data mining; decision tree; value reduction; classing rule

0 引言

分类是数据挖掘领域中重要的研究课题之一,分类规则是在已知训练信息的特征和分类结果的基础上,为每一种类型找到一个合理的描述或模型,然后再用这些分类的描述或模型对未知的新数据进行分类。目前已有多种分类模型,如粗糙集中的值约简、神经网络、统计模型、贝叶斯分类器和决策树等,其中,值约简和决策树是两个较为常用的方法。

值约简是采用逐条对表中记录进行考察的方法,删除所有不影响规则表达的冗余的条件属性值,把约简后得到的实例作为最终的规则。而决策树是采用递归的方法自顶向下建树,从生成的树中建立规则基。一个规则基包含一组规则,每一条规则对应决策树的一条路径,这条路径代表它经过结点所表示的条件的一个连接^[1]。我们用衡量规则的三个标准:可靠性、完备性、互斥性^[2]对上述方法产生的规则进行测试发现,值约简产生的规则具有很好的可靠性和完备性,但需要对决策表进行多次遍历,影响了计算时间且互斥性较差;相反,决策树产生的规则有很好的互斥性和完备性,但由于在生成树的过程中采用近乎贪心策略的方法,并且一旦确定一个结点就不再回溯,使得某些树枝对应的规则的支持度较小,所

以可靠性较差。

针对上述方法的优缺点,我们考虑将二者结合起来,在保留各自优点的基础上互相弥补各自的不足。本文首先给出一种改进的基于可辨识矩阵的求属性值核的算法,并且提出约简规则的判定准则,然后在此基础上将值核引入到决策树的修剪过程中,生成一种新的极小化算法,最后再对生成的规则进行极大化处理。

1 有关概念介绍

1.1 基于粗糙集的值约简

粗糙集理论^[3]是一种处理不确定问题的新型数学工具,它引入了信息系统作为知识的定义,把知识看作是论域的划分,引入了不可分辨关系衡量一条规则的近似质量,一个关系表可以看作一个信息系统,Skowron提出的可辨识矩阵为求取最佳规则提供了很好的思路。

1.1.1 可辨识矩阵

可辨识矩阵定义为:令 $S = (U, A)$ 是一个信息系统, U 为论域,且 $U = \{x_1, x_2, \dots, x_n\}$, C 是条件属性集, D 是决策属性集, $a(x)$ 是记录 x 在属性 a 上的值,可辨识矩阵可表示为:

$$(m_{ij}) = \begin{cases} \{a \in A: a(x_i) \neq a(x_j)\} & D(x_i) \neq D(x_j) \\ 0 & D(x_i) = D(x_j) \\ 1 & a(x_i) = a(x_j), D(x_i) \neq D(x_j) \end{cases} \quad i, j = 1, 2, \dots, n$$

1.1.2 值约简

决策系统中的每个实例,存在属性冗余的问题,求每个实例最小属性约简就是值约简问题,得到的约简结果成为规则。

对于决策表中的任意实例,可以得到一集合族 $F = \{[x]_{c_1}, [x]_{c_2}, \dots, [x]_{c_n}\}$,其中 $C = \{c_1, c_2, \dots, c_n\}$ 。令子集 $Y = [x]_D$,则有关系 $\cap F \subseteq Y$ 成立。

对于任意条件属性子集 $C' \subseteq C$ 且 $C' = \{c_1', c_2', \dots, c_m'\}$, 称由 C' 决定的 $\{[x]_{c_1'}, [x]_{c_2'}, \dots, [x]_{c_m'}\}$ 为 F 的 Y 约简, 如果满足: 1) $\cap C' \subseteq Y$; 2) 不存在 $C'' \subset C'$, 满足 $\cap C'' \subset Y$.

此时, 称 C' 为实例 x 的属性值约简, 它最简单地表示出了实例 x 的决策属性集 D 对条件属性的依赖。 F 的所有 Y 约简族记为: $VRED_Y(F)$, 从而可以得到 F 的 Y 核记为:

$$VCORE_Y(F) = \cap VRED_Y(F)$$

1.2 决策树的修剪^[4]

目前最流行的决策树算法是 ID3 算法的改进算法 C4.5, 但是, C4.5 评价决策树最主要的依据是信息增益比率, 对树的深度、结点的个数等不进行考虑。而树的平均深度直接对应着规则的平均长度, 树的结点个数代表了规则数目, 所以直接从由 C4.5 算法构造的决策树提取的规则集不是最简的, 需要对这些规则进行约简, 其原则是在保持信息系统内在依赖信息不受损失的前提下去除冗余信息使规则最简化, 把这个过程称为对决策树进行修剪。

1.3 规则的支持度^[2]

设 R 中的两个划分为 E 和 Y , 把 E 视为分类条件, Y 视为分类结论, 当等价类 $E_i \cap Y_j \neq \emptyset$, 可得分类规则 $r_{ij}: Des(E_i) \rightarrow Des(Y_j)$ 。 $Des(E_i)$ 和 $Des(Y_j)$ 分别是等价类 E_i 和 Y_j 中的记录的特征描述。规则的支持度 $sp(r) = |S(r)| / |U|$, 其中 $S(r) = M(r) \cap N(r)$, $M(r)$ 和 $N(r)$ 分别表示所有能匹配规则 r 的前提和结论的实例集合。

2 基于可辨识矩阵的属性值核的改进算法

求取处理对象的所有值约简是一个 NP-hard 问题, 因此很难通过枚举法求出问题的最小值约简。目前, 在运用粗糙集理论进行值约简方面已经有一些方法, 其中以采用值核作为附加信息的启发式算法的使用最为普遍, 所以就将值约简问题转移到求值核的问题上来。

2.1 对原算法的一些分析

目前, 求值核的方法主要有以下几种: 用值核的定义来求^[5], 或去掉某个属性前后考察正域是否发生变化来确定值核^[6]。这些算法都存在一定的不足, 如: 前者是在得到每条记录的最简规则后才能得到值核, 这显然不符合启发式算法的要求; 后者将属性约简和值约简统一起来, 需要逐个对条件属性进行测试, 速度慢, 效率低。

2.2 改进算法的基本思想

本算法是在文献[6]的基础上作出改进, 引入可辨识矩阵, 矩阵中属性组合数为 1 的项即为决策表的核属性(仅限于相容决策表^[7]), 所以这两条记录对应的核属性值就为值核。算法描述如下:

算法 1 基于可辨识矩阵的属性值核的提取算法

- (1) 用可辨识矩阵计算核属性集: $C_0 = \{m_{ij} : |m_{ij}| = 1\}$;
- (2) 统计出核属性所在的行和列, 行和列对应于决策表中的记录号;
- (3) 将这些记录的核属性值记为值核。

但是, 因为由算法 1 不能得到所有实例的值核, 并且生成的规则覆盖较多的反例, 所以仅由值核表不能得到规则集。

3 决策树规则简化

前面提到要对决策树生成的规则进行约简, 以提高规则

的支持度, 下面用一种新的极小化方法和极大化方法对决策树进行修剪, 但在修剪之前, 引入两条约简规则的判定准则, 将需要约简的规则筛选出来, 缩小了修剪的范围。

3.1 约简规则的判定准则

首先引入下面两个概念:

定义 1 子树

根结点和根结点下的每条分支就构成了一棵子树。如一棵树的根结点下有 N 条分支, 那么它就有 N 棵子树。

定义 2 树的深度

从根结点到叶子结点经过的最多的结点个数(注: 不包括叶子结点)。

决策树的叶子结点数对应着生成规则的数量, 数的深度对应生成规则的条件个数, 如果这棵树生成的叶子结点很多, 但树不深, 则说明这棵树的子结点少而每个结点的分支很多, 即每个条件属性的取值较多, 那么生成规则的条件就存在冗余。需要注意的是, 位于叶子结点上一层的结点表示的条件是必要的, 否则, 此结点在生成树的过程中就已经被删除了, 由此给出准则 1 和准则 2。

准则 1 对决策树的子树进行修剪的判定条件: $leaves(i) \geq depth(i) \geq 2$, 条件成立就修剪, 否则不修剪, 其中 $leaves(i)$ 表示第 i 棵子树的叶子结点数, $depth(i)$ 表示第 i 棵子树的深度。

准则 2 位于叶子结点上一层的结点表示的属性值为值核。

3.2 新的极小化方法

文献[3]提出的极小化方法是: 如果该规则所有条件中存在某一个条件, 在删除该条件后使得该规则不覆盖反例或不覆盖比以前多的反例, 则可删去该条件, 否则经过上述条件删除操作后的规则已经是最小规则了。该算法存在局限性: 当规则条件很多的情况下这种方法搜索的速度很慢。本文提出的极小化算法的基本思想是: 对规则覆盖的记录执行算法 1, 把其中每条记录的值核和准则 2 得出的值核一起作为规则的必要条件, 如果仅由这些必要条件生成的规则不覆盖反例或不覆盖比以前多的反例, 则可删除其余条件, 否则从剩余条件中测试是否存在可删除的条件。算法描述如下:

算法 2 基于值核的极小化算法

- (1) 用准则 1 找出要修剪的子树;
- (2) 将该子树产生规则覆盖的记录统计出来, $F = \{X_i : X_i \theta rule(i), i = 1, 2, \dots, n\}$, 其中 $X_i \theta rule(i)$ 表示规则 $rule(i)$ 覆盖的实例集 X_i ;
- (3) 把子树的每条树枝写成规则的形式, 按从底向上的顺序排列: $rule(i) : \bigwedge (A_j = val_j) \rightarrow D_k, i = 1, 2, \dots, n, j = 1, 2, \dots, m, k = 1, 2, \dots, t$;
- (4) 得出规则的必要条件: $necessity(i) = VCORE(X_i) \wedge (A_1 = val_1)$, 其中 $VCORE(X_i) = \bigwedge \{VCORE(x_j) : x_j \in X_i, j = 1, 2, \dots, |X_i|, i = 1, 2, \dots, n\}$;
- (5) For $i = 1$ to n
 - For $j = 1$ to m
 - $ruletemp(i) : necessity(i) \rightarrow D_k$
 - if $ruletemp(i)$ 不覆盖反例或不覆盖比以前多的反例 then $rule(i) = ruletemp(i)$
 - else 测试是否存在可删除的条件 $A_j, A_j \in \{rule(i) \text{ 的条件} - necessity(i)\}$

End

End

3.3 极大化算法^[3]

极大化规则学习方法相对于极小化规则学习方法可以避免仅用最小特征集来区分概念而导致忽视其他同等重要的特征;可以避免在数据稀疏情况下最简原则容易造成过分泛化,因此,规则极大化可以保证信息系统的信息一致性。通常我们希望求出的规则在排斥所有反例的前提下能覆盖尽可能多的正例,而通过对任意一个给定的规则进行极小化后再极大化,可以达到上述目的。

求解规则极大化的思路是,对每个不在规则中出现的条件属性,如果该属性在被规则所覆盖的所有实例上的相应属性值都相同,则把该属性和值组成的条件加入到规则中。令 C 为条件属性, $A \subset C$ 是条件属性集的子集, 设 R 是规则集, $x(A_i)$ 为对象 x 在属性 A 上的取值。

算法3 极大化学习算法

(1) 初始化规则为:

$$rule: \bigwedge (A_i = val_i) \rightarrow D_j$$

其中, $i = 1, 2, \dots, m, A_i \in A, j = 1, 2, \dots, n$

(2)

for $p = m$ to $k, B_p \in C - A$

for all $x_i, x_j \in R$ and $x \theta rule$

if $x_i(B_p) = x_j(B_p)$ then

$rule: (B_p = V_p) \wedge (A_i = val_i) \rightarrow D_j$

$i = 1, 2, \dots, m, A_i \in A, j = 1, 2, \dots, n$

$$leaves(1) = depth(1) = 4$$

$$rule1: (a = 1) \wedge (d = 1) \rightarrow (e = 2), sp(1) = 3/12;$$

满足此规则的记录是: 8, 10, 11

查找值核表得到 $rule1$ 的值核: $(a = 1)$

故 $rule1$ 可约简为: $(a = 1) \rightarrow (e = 2), sp'(1) = 5/12;$

同理可约简其他规则:

$$rule'(3): (c = 1) \wedge (a = 0) \wedge (d = 1) \rightarrow (e = 1),$$

$$sp'(3) = 2/12;$$

$$rule'(4): (c = 0) \wedge (b = 0) \rightarrow (e = 2), sp'(4) = 3/12;$$

其余规则已是极小化, 作极大化处理后:

$$rule6: (a = 0) \wedge (c = 1) \wedge (d = 2) \rightarrow (e = 0)$$

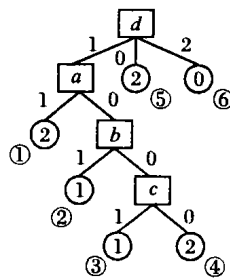


图1 C4.5 算法生成的决策树

本文给出的分类规则集生成算法有效地简化了单个规则, 使规则的支持度得到提高, 但这种方法仅适用于相容决策表, 对于不相容决策表的情况, 由于篇幅有限, 将另文发表。

参考文献:

- [1] QUINLAN JR. Induction of decision trees[J]. Machine Learning, 1986, 1: 81 - 106.
- [2] 陈欢. 基于粗糙集理论的值约简及规则提取[J]. 福州大学学报, 2004, 32(4): 472 - 475.
- [3] PAWLAK Z. Rough Set[J]. International Journal of Computer and Information Sciences, 1982, 11: 341 - 356.
- [4] 孙长嵩, 董西国, 张健沛. 一种基于粗糙集和决策树的最简分类规则集生成算法[J]. 哈尔滨工业大学学报, 2002, 23(5): 87 - 91.
- [5] 印勇, 曹长修, 张邦礼. 基于粗糙集理论的分类规则发现[J]. 重庆大学学报, 2000, 23(1): 88 - 93.
- [6] 朱红. 基于 Rough Set 的属性及属性值简约的一种算法[J]. 湘潭大学自然科学学报, 2002, 24(3): 36 - 39.
- [7] 王国胤. 决策表核属性的计算方法[J]. 计算机学报, 2003, 26(5): 611 - 615.

《计算机应用》杂志诚聘审稿专家

根据期刊的发展需要, 本刊向全国招聘计算机专业兼职审稿专家。条件如下:

1、在知名大学或研究单位从事计算机的应用、开发或教学工作;

2、职称为副教授以上或博士毕业(含博士后);

3、有较好专业英语水平;

4、熟悉并了解本学科领域在国内外的水平和发展趋势;

5、热爱期刊工作。

有意者请将相关简历资料发送到以下 E-mail 地址:

sg@computerapplications.com.cn

编辑部将致函选聘。

4 最小分类规则集生成算法

最后, 给出在给定信息系统中生成最简分类规则的算法。

算法4 最小分类规则集生成算法

(1) 对源数据进行数据选择, 进行离散化处理, 得到一个二元信息系统;

(2) 用传统方法生成决策树, 得到规则集 R ;

(3) 运行算法 1, 得到决策表的值核;

(4) 执行算法 2 和算法 3, 进行极小化和极大化处理, 得到新的规则 R' ;

(5) 合并 R' 中的同类项, 输出最简分类规则集。

5 实例说明

(1) 可辨识矩阵中含核属性的项为: $m_{1,8} = m_{6,11} = \{a\}$, $m_{6,12} = \{b\}$, $m_{1,12} = \{c\}$, $m_{1,2} = m_{1,4} = m_{2,4} = m_{3,9} = \{d\}$, 生成的值核表如表 2 所示。

表1 决策表

U	a	b	c	d	e
1	0	0	1	1	1
2	0	0	1	0	2
3	0	1	1	2	0
4	0	0	1	2	0
5	1	1	0	0	2
6	0	1	0	1	1
7	1	0	0	0	2
8	1	0	1	1	2
9	0	1	1	1	1
10	1	0	0	1	2
11	1	1	0	1	2
12	0	0	0	1	2

表2 值核表

U	a	b	c	d	e
1	0		1	1	1
2				0	2
3				2	0
4				2	0
6	0	1			1
8	1				2
9			1		1
11	1				2
12		0	0		2

(2) 用 C4.5 算法生成的决策树如图 1 所示, 并在图中标记出规则的顺序。

(2) 生成的决策树共有三棵子树, 显然只有第一棵需要修剪: