

基于分辨关系的决策树构建算法

刘 军

(南京工业大学 电子与信息工程学院, 江苏 南京 210009)

摘 要: 针对完全以核属性为首选构建决策树算法的不足, 本文以粗糙集的分辨关系为基础, 提出以决策属性的分辨量和分辨类别两个参数确定条件属性的重要性, 择优自顶向下构造决策树. 该算法的优点在于算法简洁有效和时空复杂度低. 实例分析的结果表明, 该方法能获得更为简洁有效的决策树.

关键词: 粗糙集; 决策树; 分辨关系

中图分类号: TP18

文献标识码: A

文章编号: 1000-7180(2012)01-0109-04

Algorithm of Constructing Decision Tree Based on Definite Relation

LIU Jun

(College of Electronic and Information Engineering, Nanjing University of Technology, Nanjing 210009, China)

Abstract: The algorithm of decision tree is achieved by the relative core of attributes, which is incomplete. The algorithm is proposed in this paper on the definite relation of rough set, which chooses important attribute by the definite count and the classifying value, and makes decision tree from top to bottom with the attribute. The benefit of the algorithm is simple in calculation and is effectively used in the classification. The experiments and comparison show that the algorithm provides more precise and simple decision tree.

Key words: rough set; decision tree; definite relation

1 引言

决策树是一种直观的知识表示方法, 同时也是高效的分类器. 决策树与规则在应用上的判定方法有所不同, 规则应用是一个遍历过程, 所以规则越简单越好; 而决策树采用自顶向下的顺序, 取决于节点的数量和节点的深度, 所以构建决策树的重要问题是各级节点属性的选择, 即用最少的属性使所生成的决策树中节点深度较小. 目前的决策树算法还存在下列不足:

(1) 基于信息增益为启发式来构造决策树, 该算法以信息量的大小确定属性重要性, 即倾向于取值种类较少的属性, 是一种局部优化策略^[1].

(2) 基于粗糙集的决策树生成算法, 以属性约简为出发点, 存在核属性优先及属性值选择范围小(有的属性被约简)的前题, 所以构造的决策树往往

增加节点的深度和上层叶结点少、下层叶结点多的问题, 另外属性约简本身仍存在诸多难题^[2].

针对上述问题, 给出了一种决策树的构造算法, 以粗糙集分辨关系^[2]为理论基准, 从 $\{A_n\}$ 中属性 A_i 对 D 分辨关系前提, 将分辨能力较强且可尽快分出叶结点的属性作为优选属性递进地完成决策树的构造. 这样能保证生成的决策树的上层叶结点多且可尽量减少树中结点深度. 最后, 用实例及其比较分析说明算法相对于其它算法的优越性.

2 基本理论和算法

2.1 基本理论

设 $S=(U, A_n, D, V, F)$ 是一决策表, 其中, A_n 为条件属性集, D 为决策属性; $U=\{x_1, x_2, \dots, x_m\}$ 是论域; V 是属性值集合; F 是 $U \times (A_n \cup D) \rightarrow V$ 的映射.

收稿日期: 2011-04-25; **修回日期:** 2011-06-13

基金项目: 国家自然科学基金项目(60673185); 教育部留学回国人员科研启动基金项目(2007111108)

定义 1 在知识表达系统 $S=(U, An, D, F)$ 中, 对于每个属性子集 $B \subset (An \cup D)$, 定义

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

为一个不可分辨关系。

不可分辨关系是构建决策表的主要理论依据。

定义 2 设属性 $A_i \in An, v \in A_i, (A_i, v)$ 表示属性 A_i 取值为 v 的条件. $NA_{i,v} = \text{count}(x_i \in U \mid (A_i, v))$ 为属性 A_i 中取值为 v 时属性 A_i 中为 v 值的属性值数量, 称为 $A_{i,v}$ 的同值量。

$NA_{i,v}$ 是属性 A_i 中取同一种值 v 所对应决策表的行数, 表示其能分辨 D 的基本能力, A_i 是相对独立的参数. A_i 有几种取值就有几个这样的同值量集与之对应. 例如: 表 1 中, 属性 A_1 中有两种取值 0 和 1, 其对应 $NA_{i,v}$ 分别是: $NA_{1,0} = 6; NA_{1,1} = 15$.

定义 3 设属性条件子集 $B \subset (A_i \cup D) \mid (A_i, v), d \in D. CA_{i,v} = \text{count}(x_i \in U \mid (A_i, v), d \in D)$ 是属性 A_i 中取值为 v 时等价类集中不同等价类别的种类个数, 即对应 (A_i, v) 的 $\{x_i\}$ 中 $d \in D$ 种类个数. 称为 A_i 基于取值 v 时 D 的类别值。

$CA_{i,v}$ 是属性 A_i 中值为 v 的属性值与对应决策属性值 D 的间分辨关系, 表示其能分辨 D 的难度, 是 A_i 相对于 D 的关联参数. A_i 有几种取值就有几个这样的类别值与之对应. 例如: 表 1 中, 属性 A_1 中的两种取值 0 和 1, 其对应 $CA_{i,v}$ 分别是: $CA_{1,0} = 1 (D=3); CA_{1,1} = 3 (D=1, 2, 3)$.

2.2 算法

通过研究 An 与 D 之间的分辨关系来构建决策树的首要问题是如何确定划分决策树的结点属性, 即如何在 An 选择最优属性的问题, 每个属性 $A_i \in An$ 都是由属性值组成, 所以可以先确定属性 A_i 中每种属性值的分辨能力, 其累加和即为该属性的分辨能力。

属性 $A_i \in An$ 中每种同值量 $NA_{i,v}$ 对决策属性值 $d_i \in D$ 的可分辨的数量(行数)是由其 $NA_{i,v}$ 和 $CA_{i,v}$ 两个参数(定义 2、3)决定的. 根据定义 1, 属性 A_i 的 $NA_{i,v}$ 值越大且 $CA_{i,v}$ 值越小, 则 A_i 的分辨能力越强. 所以依据 $NA_{i,v}$ 和 $CA_{i,v}$ 值可推出判别属性分辨能力强弱的一般式: 若 $CA_{i,v} = 1$, 则经一次结点划分, 同值量 $NA_{i,v}$ 可一次分辨出 $d_i \in D$ 中的 $NA_{i,v}$ 行, 且得 1 个叶结点; 若 $CA_{i,v} = 2$, 一次结点划分后, 要在 2 个分区内与其它属性值组合完成对 $d_i \in D$ 的分辨, 即需要外配 2 组其它属性的属性值对 $d_i \in D$ 进行分辨, 所以至少要经二次结点划分, 同值量

$NA_{i,v}$ 才可二次分辨出 $d_i \in D$ 的 $NA_{i,v}$ 行, 得 2 个叶结点; 若 $CA_{i,v} = 3$, 一次结点划分后, 要在 3 个分区内与其它属性值组合完成对 d_i 的分辨, 需要外配 3 组属性值对 $d_i \in D$ 进行分辨, 这要比外配 2 组属性值增加 1 级的难度, 所以可能需经三次结点划分, 同值量 $NA_{i,v}$ 才可三次分辨出 $d_i \in D$ 的 $NA_{i,v}$ 行, 得 3 个叶结点; …… 据此, 得属性 A_i 中为 v 的值集对 $d_i \in D$ 的分辨度 $IA_{i,v}$ 一般算式:

$$IA_{i,v} = NA_{i,v} / CA_{i,v}$$

由此可得属性 $A_i \in An$ 对 $d_i \in D$ 的分辨度 IA_i 算式:

$$IA_i = \sum IA_{i,v} \mid \forall v \in A_i = \sum (NA_{i,v} / CA_{i,v}) \mid \forall v \in A_i$$

而最强分辨属性(集):

$$\{MIA_j\}_0 = \text{Max}(IA_i) \ (i=1, 2, \dots, n) \mid j \in n$$

决策树的首选属性(0 层)必定在 $\{MIA_j\}$ 中. 再对 $\{MIA_j\}_0$ 中的每个属性进行一次划分后求得各自的 $\{MIA_j\}_1$, 对 $\{MIA_j\}_1$ 进行比较得第一层的最强分辨属性(集), 这样逐层进行择优, 直至最底层(N 层)属性分辨完成. 由 0 层至 N 层一直存留属性划分就是最终的决策树, 而其中的每次属性划分就是该树的结点. 这是判优(属性)建树一次完成的过程: 在当前表中, 用 $\{MIA_j\}$ 进行一次划分生成一个枝结点且得若干子表, 再对各子表求其各自的 $\{MIA_j\}$, 直到子表中 $\{MIA_j\}$ 中只有一种 $d \in D$ 值, 则生成一个叶结点. 算法如下:

```
CreateDtree(An, D, IA) { // 递归程序入口
    if (conut(D) = conut(IA)) { // 判子集 D 是否仅一类值
        CreateNode(IA, D); return // 建叶结点且回退
    }
    for (;  $\forall A_i \in An$ ; ) { // 对 An 中的每个属性循环
        for (;  $\forall v \in A_i$ ; ) { // 对 A_i 中的每种值循环
            if ( $IA_i > \{MIA_j\}$ ) {  $\{MIA_j\} = IA_i$ ; } // 求最优属性  $\{MIA_j\}$ 
        }
        CreateNode( $\{MIA_j\}$ ); // 用  $\{MIA_j\}$  构建 1 个划分结点
        for (;  $\forall v \in \{MIA_j\}$ ; ) { // 对  $\{MIA_j\}$  中的每种值循环
             $U = \{x_i \mid (v \in \{MIA_j\})\}$ ; // 求  $\{MIA_j\}$  中等于 v 值的行号
             $\{An, D\} \mid U = \{x_i\}$ ; // 按 U 对  $\{An, D\}$  划分
             $\{An\} - = \{MIA_j\}$ ; // 在  $\{An\}$  去除属性  $\{MIA_j\}$ 
            CreateDtree(An, D,  $\{MIA_j\}$ ); } // 递归调用
```

3 应用实例分析与比较

3.1 应用实例分析

文献[3]中表 2 是一个关于汽车数据的信息系统,其中,条件属性集为 A_n ;决策属性为 D . 用符号替代后见表 1.

表 1 决策表

U	A1	A2	A3	A4	A5	A6	A7	A8	A9	D
1	1	0	0	2	1	0	1	1	2	1
2	1	0	0	1	1	0	2	0	2	1
3	1	0	1	1	1	1	2	1	1	2
4	1	0	0	1	1	0	2	0	1	2
5	1	0	0	1	1	1	2	0	1	2
6	1	0	0	1	1	0	1	0	1	2
7	1	0	0	2	0	1	0	0	1	2
8	1	0	0	2	0	1	1	1	1	2
9	1	0	0	2	1	0	1	0	0	2
10	1	0	1	1	1	0	2	0	0	2
11	1	0	0	1	1	0	2	1	0	2
12	1	0	0	1	1	0	2	1	0	2
13	1	0	0	2	0	1	1	0	0	2
14	1	0	1	1	1	1	2	0	0	3
15	0	0	0	2	0	1	0	0	0	3
16	0	0	0	1	0	1	0	0	0	3
17	0	0	0	1	0	0	1	0	0	3
18	0	0	1	1	0	1	2	0	0	3
19	0	0	0	2	0	0	0	0	0	3
20	0	0	0	1	0	1	1	0	0	3
21	1	0	0	1	0	1	1	0	0	3

首次最优属性 $\{MIA_j\}$ 的选择:

$$IA_1 = \sum IA_{1_v} | (\forall v \in A_1)$$
$$= \sum (NA_{1_v} / CA_{1_v}) | (\forall v \in A_1)$$
$$= \sum IA_{1_v} (v=0,1) = (NA_{1_0} / CA_{1_0})$$
$$+ (NA_{1_1} / CA_{1_1})$$
$$= (15/3) + (6/1) = 11$$

$$IA_2 = (21/3) = 7$$
$$IA_3 = (17/3) + (4/2) = 7.7$$
$$IA_4 = (14/3) + (7/3) = 7$$
$$IA_5 = (10/2) + (11/3) = 8.7$$

$$IA_6 = (11/2) + (10/3) = 8.8$$
$$IA_7 = (8/3) + (4/2) + (9/3) = 7.7$$
$$IA_8 = (16/3) + (5/2) = 8.8$$
$$IA_9 = (13/2) + (6/1) + (2/1) = 14.5$$
$$\{MIA_j\} = \max(IA_i) = IA_9 \ (i=1,2,\dots,9)$$

经 $\{MIA_j\}$ 划分后,得第一层的 2 个叶结点:
 $A_9=2$ then $D=1$; $A_9=1$ then $D=2$.

在 $A_9=0$ 的划分集中,即 $\{A_1 \sim A_8\}$ 、 $U=\{9 \sim 21\}$ 子表中,最优属性 $\{MIA_j\}$ 的选择:

$$IA_1 = 9.5; IA_2 \sim IA_6 = 6.5; IA_7 = 8; IA_8 = 7.5.$$
$$\{MIA_j\} = \max(IA_i) = IA_1 \{=9.5\} \ (i=1,2,\dots,8)$$

经 $\{MIA_j\}$ 划分后,得第二层的 1 个叶结点:
 $A_1=0$ then $D=3$.

在 $A_1=1$ 的划分集中,即 $\{A_2 \sim A_8\}$ 、 $U=\{9 \sim 14,21\}$ 子表中,最优属性 $\{MIA_j\}$ 的选择:

$$IA_2, IA_3, IA_5, IA_7 = 3.5; IA_4, IA_8 = 4.5;$$
$$IA_6 = 5.5$$
$$\{MIA_j\} = \max(IA_i) = IA_6 = 9.5 \ (i=2,\dots,8)$$

经 $\{MIA_j\}$ 划分后,得第三层的 1 个叶结点:
 $A_6=0$ then $D=2$.

在 $A_6=1$ 的划分集中,即 $\{A_2 \sim A_5, A_7, A_8\}$ 、 $U=\{9 \sim 14,21\}$ 子表中,最优属性 $\{MIA_j\}$ 的选择:

$$IA_2, IA_8 = 1.5; IA_3, IA_5, IA_7 = 2; IA_4 = 3$$
$$\{MIA_j\} = \max(IA_i) = IA_4 = 3 \ (i=2,\dots,5,7,8)$$

经 $\{MIA_j\}$ 划分后,得最底层 2 个叶结点: $A_4=2$ 则 $D=2$; $A_4=1$ 则 $D=3$.

表示为决策树如下:

[A9]
A9. 2→D. 1
A9. 1→D. 2
A9. 0
[A1]
A1. 0→D. 3
A1. 1
[A6]
A6. 0→D. 2
A6. 1
[A4]
A4. 2→D. 2; A4. 1→D. 3

按属性的重要性算法^[2],属性的分辨强度顺序为: A_9 、 A_6 、 A_4 、 A_8 、 A_1 、 A_7 、 A_3 、 A_5 、 A_2 ,其中 A_9 、 A_6 、 A_4 为核属性. 若按核属性顺序进行结点划分所

得决策树如下:

```

[A9]
A9.2→D.1
A9.1→D.2
A9.0
[A6]———[A6]
A6.1      A6.0
[A4]      [A4]———[A4]
A4.1→D.3
A4.2      A4.2      A4.1
[A1]      [A1]      [A1]
A1.1→D.2  A1.1→D.2  A1.1→D.2
A1.0→D.3  A1.0→D.3  A1.0→D.3

```

将此算法与本文算法比较,都涉及 4 个属性,但在本文算法的决策树中,共用 4 层,每层都是单枝且都产生叶结点;而属性的重要性算法决策表则多在最底层叶结点,显见其复杂度远高于本文算法的决策表。

3.2 比较

3.2.1 与粗集决策树构造算法比较

文献[1]的粗集决策树构造算法对该文中表 1 计算得其首选结点划分属性应是 a_4 。按本文算法的决策树算法:

$$Ia_1 = (8/2) + (2/2) = 5$$

$$Ia_2 = (3/2) + (6/2) + (3/2) = 6$$

$$Ia_3 = (1/1) + (8/2) + (3/2) = 6.5$$

$$Ia_4 = (4/2) + (6/2) + (2/1) = 7$$

$\{MIA_j\} = \max(Ia_i) = Ia_4$ 。所以首选结点划分属性应是 a_4 ,与文献[1]结果相同但算法简洁。

3.2.2 与改进的正区域算法比较

文献[4]的改进的正区域算法对该文中表 1 计算得其首选结点划分属性应是 a_2 。将文献[4]表 1 的 13~15 相同的行合并为 1 行,按本文算法的决策树算法:

$$Ia_1 = (3/1) + (4/2) + (2/1) + (4/2) = 9$$

$$Ia_2 = (5/1) + (3/2) + (3/2) + (2/2) = 9$$

$$Ia_3 = (2/1) + (5/2) + (5/2) + (1/1) = 8$$

$$Ia_4 = (6/3) + (3/1) + (4/2) = 7$$

$\{MIA_j\}_0 = \max(Ia_i) = \{Ia_1, Ia_2\}$ 。所以首选结点划分属性应是 a_1 和 a_2 , $\{MIA_1\}_1 = \{a_3, a_4\} | IA_3 = 2.5, IA_4 = 3; \{MIA_2\}_1 = \{a_3, a_4\} | IA_3 = 2, IA_4 = 6$ 。由于在第一层中 $\{MIA_2\}_1$ 优于 $\{MIA_1\}_1$, 所以以 a_2 再对 a_3, a_4 进行划分即直接得到由 $a_2(0$ 层)和 $a_3, a_4(1$ 层)的决策树。其结果与文献[4]相同

但算法简洁。

3.2.3 与加权平均粗糙度算法比较

按加权平均粗糙度算法,由文献[5]中表 3 中首选结点划分属性是 a ,所得涉及 3 个属性具有 4 个枝结点的决策树。见文献[5]的图 1;按本文的决策树算法:

$$Ia = (2/1) + (4/2) + (4/2) = 6$$

$$Ib = (2/2) + (5/2) + (3/2) = 5$$

$$Ic = (3/2) + (5/2) + (2/1) = 6$$

$$Id = (3/1) + (5/2) + (2/2) = 6.5$$

$$\{MIA_j\} = \max(Ia_i) = Id$$

所以首选结点划分属性是 d ,经二次划分得出了只涉及 2 个属性且具有 3 个枝结点的决策树如下:

[d]

d.1→e.3

d.2

[a]

a.1→e.1; a.2→e.2; a.3→e.2

d.3

[a]

a.1→e.1; a.2→e.2

该决策树优于文献[5]的决策树。

3.2.4 与最简规则比较

表 1 的最简规则(由 11 个属性值构成 6 条规则)与 3.1 的决策树展开规则(由 15 个属性值构成 6 条规则)相比较,见表 2。若将树结点重复属性值取平均数则决策表展开规则基本接近最简规则水平,可见 3.1 节得到的决策树是较佳的决策树。

表 2 决策树展开规则与最简规则比较表

最简规则	决策树展开规则
A9.2→D.1	A9.2→D.1
A9.1→D.2	A9.1→D.2
A1.1∧A4.2∧A8.0→D.2	A9.0∧A1.1∧A6.1∧A4.2→D.2
A1.1∧A6.0∧A9.0→D.2	A9.0∧A1.1∧A6.0→D.2
A1.0→D.3	A9.0∧A1.0→D.3
A4.1∧A5.0→D.3	A9.0∧A1.1∧A6.1∧A4.1→D.3

4 结束语

为决策表的构建提供了一种有效算法,该算法具有以下特点:

(下转第 116 页)

统资源消耗比传统定点查表或者定点 Cordic 算法大。

7 结束语

本文介绍了两种定点数三角函数在 FPGA 上的实现方法,并且比较了这两种方法在消耗系统资源,运算速度和精度方面的优劣。查表算法计算精度高,占用系统资源少;定点 Cordic 算法运算速度快,精度易操控。

采用 Cordic 算法直接设计 32 位浮点数三角函数是本文的新意所在。该设计在不经定点浮点转换的基础上,通过对浮点数指数位减 1 操作和浮点加运算实现了 Cordic 迭代算法,使 32 位浮点数的三角函数运算更加便捷。

参考文献:

- [1] Volder. The cordic trigonometric computing technique. IRETrans [J]. Electronic Computers, 1959, EC-8(3): 334-334.
- [2] 李全,陈石平,付佃华. 正交三角函数的 Cordic 实现 [J]. 微计算机信息, 2008, 24(3).
- [3] 马士超,王贞松. 基于 DSP 的三角函数快速计算 [J]. 2005, 31(22).
- [4] 张天瑜. 基于旋转模式的改进型 Cordic 算法研究 微电子学与计算机 2010(3): 93-97.
- [5] 王亚春,蔡德林,张梦龙,等. 流水线 Cordic 算法的 FPGA 实现 [J]. 通信技术, 2010(11): 169-171.
- [6] 张建斌,梁芳,刘乃安. 一种改进型算法的 Cordic 实现 [J]. 微电子学与计算机, 2010(11): 181-184.

作者简介:



吴庆达 男,(1984—),硕士研究生. 研究方向为数字集成电路设计。

何书专 男,工程师. 研究方向为数字集成电路设计。
潘红兵 男,副教授. 研究方向为数字集成电路设计。
沙金 男,副教授. 研究方向为数字集成电路设计。
李伟 男,工程师. 研究方向为数字集成电路设计。

(上接第 112 页)

(1) 粗糙集的不分辨关系是以 $\{A_n\}$ 行属性值对 D 分辨关系为基础,而算法是以 $\{A_n\}$ 中属性 A_i 对 D 分辨关系为基础。

(2) 算法用属性局部参数同值量和全局参数类别值共同确定属性分辨率,只涉及单一属性与决策属性的关系。由 S 直接得决策树,不需求等价类和属性约简等全局性的复杂运算。

(3) 从比较分析可知,一般在 $\{MIA_j\}$ 集中就只有一个必选属性;而当 $\{MIA_j\}$ 集中有多个必选属性时,采用降层的方法继续分辨,这就极大地提高了算法的精确性。

(4) 算法简洁易理解;时间和空间复杂度极小。

参考文献:

- [1] 丁春荣. 基于粗糙集的决策树构造算法 [J]. 计算机工程, 2010, 36(11): 75-77.
- [2] 李永华. 一种基于 rough 集的属性约简的改进算法 [J]. 计算机应用, 2008, 28(8): 200-202.
- [3] 王杨. 基于小生境遗传算法的粗糙集属性约简方法 [J]. 计算机工程, 2008, 34(5): 66-70.
- [4] 高静. 一种改进的基于正区域的决策树算法 [J]. 计算机科学, 2008, 35(5): 138-142.
- [5] 陈婷. 基于粒度的粗集-决策树雷达信号识别模型 [J]. 微电子学与计算机, 2008, 25(12): 13-16.

作者简介:

刘军 女,(1977—),讲师. 研究方向为数据挖掘、粗糙集、粒计算。