第17卷 第7期
2007年7月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol. 17 No. 7
Jul. 2007

动态模糊决策树学习算法研究

蔡 晨,李凡长

(苏州大学 计算机科学与技术学院,江苏 苏州 215006)

摘 要:针对自动控制领域中普遍存在的动态模糊信息,提出了基于DFS(动态模糊集)建模的动态模糊决策树算法,并给出了对包含非动态模糊属性、缺少属性值的输入样例的匹配算法,很好地解决了模糊控制系统所不能解决的动态性问题。

关键词:动态模糊集;动态模糊决策树;动态模糊规则

中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2007)07-0073-04

Research of Algorithm of Dynamic Fuzzy Decision Tree

CAI Chen, LI Fan-zhang

(College of Computer Science and Technology, Soochow University, Suzhou 215006, China)

Abstract: According to large numbers of dynamic fuzzy systems exist in the fields of automatic control, design the arithmetic of dynamic fuzzy decision tree in this paper, moreover, give the method how to deal with those input examples which include non-dynamic fuzzy attributes or lack some attributes. The theory paper can solve the dynamic problems that the fuzzy control system can not solve.

Key words: dynamic fuzzy set; dynamic fuzzy decision tree; dynamic fuzzy rule

0 引 言

决策树归纳学习方法以Quinlan在1986年提出的ID3方法为代表。由于该算法比较简单,容易事先而且适于处理规模较大的学习问题,现已成为归纳学习的一个重要分支[1]。

决策树是结点的集合,一棵决策树的内部结点是属性或属性的集合,叶结点是所要学习划分的类。使用训练集建立决策树后,便可根据属性的取值对未知实例集进行分类,由树根开始对该实例的属性逐渐测试,顺着分支向下走,直至到达某个叶结点,此叶结点代表的类别即为该实例所属的类[2]。

目前,决策树在处理精确数据或者模糊数据上已有了很多成果和应用。可是,使用决策树思想对动态模糊信息进行处理的研究工作还比较初步。

使用决策树学习方法处理动态模糊问题分为以下步骤:对问题形式化(用数学工具描述动态模糊信息)→根据形式化后的数据建立决策树→使用已建立决策树进行预测。DFS理论是一种描述、处理动态模糊信息的理论工具。文中叙述了使用DFS对动态模糊信息进行描述、处理,在此基础上构建的一种动态模糊决策树算法(DFDTA),并且用实际数据对决策树的预测效果进行验证。

收稿日期:2006-10-02

作者简介:蔡 晨(1982-),男,江苏苏州人,硕士研究生,主要研究方向为人工智能、机器学习、动态模糊逻辑;李凡长,教授,主要研究方向为人工智能、动态模糊逻辑、机器学习、多Agent系统理论与方法。

1 动态模糊集

动态模糊集(DFS, dynamic fuzzy set)是处理动态模糊信息的理论工具[3]。文中使用DFS对动态模糊性问题进行建模。对学习问题形式化,动态模糊信息转变为计算机能够处理的符号和数字。

定义1 设在论域 U 上定义的一个映射:

$(\overleftarrow{A},\overrightarrow{A}):(\overleftarrow{U},\overrightarrow{U})\to[0,1]\times[\leftarrow,\rightarrow]$, $(\overleftarrow{u},\overrightarrow{u})\mapsto(\overleftarrow{A}(\overleftarrow{u}),\overrightarrow{A}(\overrightarrow{u}))$ 记为 $(\overleftarrow{A},\overrightarrow{A})=\overleftarrow{A}$ 或 $\overrightarrow{A}$,则称 $(\overleftarrow{A},\overrightarrow{A})$ 为 $(\overleftarrow{U},\overrightarrow{U})$ 上的动态模糊集,简称DFS,称 $(\overleftarrow{A}(\overleftarrow{u}),\overrightarrow{A}(\overrightarrow{u}))$ 为隶属函数(Membership function)对 $(\overleftarrow{A},\overrightarrow{A})$ 的隶属度(Membership degree)。

对于任何一个实数 $a\in[0,1]$,都可以把 a 动态模糊化为:$a\overset{DF}{=}(\overleftarrow{a},\overrightarrow{a})$, $a\overset{DF}{=}\overleftarrow{a}\ or\ \overrightarrow{a}$, $\max(\overleftarrow{a},\overrightarrow{a})\overset{\Delta}{=}\overrightarrow{a}$, $\min(\overleftarrow{a},\overrightarrow{a})\overset{\Delta}{=}\overleftarrow{a}$。通过动态模糊化,就可以把 a 状态的发展变化趋势直观地表示出来了。

在论域(Domain of discourse) U 上可以有多个DF集,记 U 上的DF集的全体为 $DF(U)$,即:

$DF(U)=\{(\overleftarrow{A},\overrightarrow{A})\mid(\overleftarrow{A},\overrightarrow{A}),(\overleftarrow{u},\overrightarrow{u})\mapsto[0,1]\times[\leftarrow,\rightarrow]\}=\{(A\times(\leftarrow,\rightarrow))\mid(A\times(\leftarrow,\rightarrow)),(u\times$

$(\leftarrow,\rightarrow))\mapsto[0,1]\times[\leftarrow,\rightarrow]\}$

若论域 U 为时变域,则表示为 U_T(其中 T 表示时间),相应的 DF(U_T) 记为

$DF(Ur)=\{(\overleftarrow{A},\overrightarrow{A})\mid(\overleftarrow{A},\overrightarrow{A}),(\overleftarrow{u},\overrightarrow{u})\mapsto[0,1]\times[\leftarrow,\rightarrow]\}=\{(A_t\times(\leftarrow,\rightarrow))\mid(A_t\times(\leftarrow,\rightarrow)),(u\times(\leftarrow,\rightarrow))\mapsto[0,1]\times[\leftarrow,\rightarrow]\}$(其中 $t\in T$)

一组决策树的训练样例(或验证样例) 是多个动态模糊数的析取,即:$A\wedge B\wedge\cdots\wedge C$。其中 A,B,C 是样例的某一个条件(如描述温度、天气等),是由 DFS 描述的。$A=\sum(\overleftarrow{A_i},\overrightarrow{A_i})/(\overleftarrow{U_i},\overrightarrow{U_i})$,$U_i$ 表示 A 论域内的一个成员,A_i 表示 U_i 的隶属度。

模糊决策树是一种处理模糊信息的决策树,对于一个预测样例得到的结果是不同的。即:任何一个叶子结点都可能是输出结果[4]。文中得到的决策树在建立完毕后,对任何一个输入样例都有唯一的判定结果,可以和传统决策树进行性能对比。

定义 2 定义以 DFS 表示的析取规则式称为动态模糊规则。

动态模糊规则的每一个析取子句是对动态模糊属性隶属度的一个限制条件。推导出的目标属性是精确数据。

2 动态模糊决策树

文献[5]提出了处理模糊信息的模糊决策树算法,一个结点的任何两个子结点都可能有非空交集。由一棵建立完毕的决策树提取的规则是模糊规则,一个输入样例和所有规则相匹配,得到匹配的符合程度。因此,输出结果是不确定的,符合程度高的结果输出的可能更大。这使符合程度非常低的结果也可能出现,不适合用于自动控制。

文中建立的决策树用划分的思想扩展子树,由一个结点扩展出的所有子结点都没有非空交集,并且构成了对该结点的一个划分。任何一个输入样例在一个结点处判断后只可能进入一条子结点分支。因此,划分思想建立的决策树对任何一个输入样例都只有唯一的判定结果[6]。

2.1 连续隶属度离散化

定义 3 设训练实例集为 X,属于第 i 类的训练实例个数是 N_i,X 中总的实例个数为 $|X|$,若记一个实例属于第 i 类的概率为 p_i,则

$$p_i\equiv\frac{N_i}{|X|}\tag{1}$$

如果决策属性具有 k 个不同的值,那么训练实例集 X 相对于 k 个状态分类的熵为:

$$\text{Entropy}(X)\equiv\sum_{i=1}^{k}-p_i\log_2 p_i\tag{2}$$

一个属性 A 相对于实例集合 X 的信息增益被定义为:

$$\text{Gain}(X,A)\equiv\text{Entropy}(X)-\sum_{v\in\text{Values}(A)}\frac{|X_v|}{|X|}\text{Entropy}(X_v)\tag{3}$$

其中 Values(A) 是属性 A 所有可能的值的集合;X_v 表示 X 中属性 A 的值为 v 的实例子集,即 $X_v=\{x\in X\mid A(x)=v\}$。

一个论域成员的隶属度在[0,1] 范围内,并且是连续的。引入一个阈值组 $V=\{v_1,v_2,\cdots,v_n\}$ 对其离散化,对于任何一隶属度在(V_i,$V_{(i+1)}$) 建立一个子结点。

对某论域成员 A 进行离散化的过程如下:

算法 1(隶属度离散化算法):

(1) 对属性 A 的取值进行排序,设得到的序列为:$a_1,a_2,\cdots,a_n$。

(2) 每个 $\text{Mid}_i=(a_i+a_{i+1})/(i=1,2,\cdots,n-1)$ 为一个可能的区间边界,Mid_i 称为候选分割点。Mid_i 将实例集合 U 划分为两个集合:

$$U_i^{\triangleleft}=\{x\in U\mid V_A(x)\leqslant\text{Mid}_i\}$$

$$U_i^{\triangleright}=\{x\in U\mid V_A(x)>\text{Mid}_i\}$$

选择 Mid_i,使其对实例集 U 划分后的熵最小,熵的计算公式为

$$\text{Entropy}(U,\text{Mid}_i)=\frac{|U_i^{\triangleleft}|}{|U|}\text{Entropy}(U_i^{\triangleleft})+\frac{|U_i^{\triangleright}|}{|U|}\text{Entropy}(U_i^{\triangleright})$$

其中 $\text{Entropy}(U_i^{\triangleleft})$ 与 $\text{Entropy}(U_i^{\triangleright})$ 的计算公式为式(2),此时实例集 U 被划分为两个子集 $U_i^{\triangleleft}$,$U_i^{\triangleright}$。

(3) 判断 $\text{Entropy}(U,\text{Mid}_i)\leqslant\omega$,若是,则停止划分,否则,递归地对 $U_i^{\triangleleft}$,$U_i^{\triangleright}$ 进行(1)、(2) 的划分操作。其中,ω 为制定的停止划分的阈值。

算法 1 中最终提取的 Mid 集合即为阈值组 V,{V 集合,0,1} 是构成结点一个划分的所有边界。

2.2 扩展结点论域选取方法

由划分思想建立的决策树(包括 ID3、C4.5)是以某个评价标准作为选取测试结点条件的标准(如:信息增益、信息增益率)[7]。系统选取训练样例中的一个条件来扩展子结点。静态精确数据使用一个实数来描述一个条件,系统可以根据该条件的数据大小来判断分支。动态模糊数据是一组动态化的模糊数据,无法使用处理精确数据的方法。

对训练样例每个条件的每个论域成员平等对待,以一种评价标准选取其中一个论域成员,根据这个论域成员的隶属度值来扩展该结点的分支。算法 1 计算

出一个论域成员的若干个阈值 $V_1 \sim V_n$,该 n 个阈值将一个结点扩展出 $n+1$ 个子结点,计算出该划分的信息增益率(如公式(4))。计算出所有论域成员的信息增益率,选取信息增益率最大的论域成员作为扩展子结点的依据。因为各论域成员的阈值 V 数量各不相同,建立的决策树是多叉树。在同一层上的所有结点可能存在数量不同的子结点。

$$\text{Gainratio}(S,A) = \frac{\text{Gain}(S,A)}{\text{SplitInformation}(S,A)} \tag{4}$$

$$\text{SplitInformation}(S,A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|} \tag{5}$$

2.3　处理非动态模糊属性、缺省属性

训练样例的某些条件可能是精确数据(如:性别),以处理动态模糊数的算法处理精确数据会造成效率低下。系统将精确化的属性作为一个论域成员,精确属性的论域成员集合相对于动态模糊属性的一个论域成员的阈值集合。精确属性的信息增益率是根据论域成员集合计算得出的。当一个精确属性被选取作为扩展子结点时,所有论域成员各生成一个子结点。

采集到的数据常常出现缺省属性,在计算机每个论域成员的信息增益率时,对于该论域成员缺省的训练样例不归入统计计算行列。

2.4　动态模糊决策树算法(DFDTA)

在上述结论下,建立动态模糊决策树(简称 DFDT)。

算法 2(动态模糊决策树算法):

(1)如所有输入的训练样例有相同目标属性,则产生仅一个结点的决策树(目标结点为目标属性取值)。

(2) 如存在不同目标属性的训练样例,对所有条件属性的论域成员离散化处理,取得阈值集合 V(按照算法 1)。

(3) 根据 V 集合计算出所有论域成员的信息增益率,取得最高信息增益率成员 X。

(4) 按照 X 的阈值集合 V 对 X 划分,$(V_i, V_{(i+1)})$ 为 X 结点划分所得子集合之一(一条分支),其中 $V_0 = 0, V_{n+1} = 1, V = \{V_1, V_2, \cdots, V_n\}$。

(5)对 X 的所有划分所得子集合进行(1)~(4)处理。

算法 2 处理的是 DFS 理论表示后的训练样例,以划分结点集合的方式建立的决策树。在计算阈值、最高信息增益率时,系统忽略该项属性缺省的训练样例。

2.5　规则提取、规则匹配

从动态模糊决策树中提取出规则,更形式化地描述决策树。系统自顶向下中序遍例整个决策树,每个结点分支都代表一个判断式(如:$A_i < 0.3$)。提取的规则是所有判断式的析取,各判断式没有先后次序。输入样例自决策树顶而下逐一匹配判断各结点,直至叶子结点而得到目标属性的值。

对于含有缺省属性的输入样例,在自上而下匹配中可能遇到缺省属性结点,系统跳过该匹配结点,对该结点的所有子树进行自上而下的匹配判断,最后按照所有匹配结果提取占据比例最多的目标属性值。算法如下:

算法 3(规则匹配算法):

(1)if(结点 X 是叶子) 记录该叶子的目标属性取值;

(2)else if(结点 X 所在属性不是输入样例的缺省属性) 匹配结点 X 的条件和输入样例,选择正确的子结点;

(3)else if(结点 X 所在属性是输入样例的缺省属性) 对 X 的所有子树进行(1)、(2),取得所有子树的目标属性取值;

(4)对于所有目标属性取值计数,选取数量最多的目标属性取值作为输入样例的判断结果。

算法 3 在判断含有缺省属性的输入样例时,完全根据非缺省属性来进行判断,在多个可能出现的结果中选择可能性最高的取值。

3　解决过度拟合问题

决策树学习方法普遍具有过度拟合问题,当训练样例数目很大时,决策树能更好符合训练样例,但是对真实情况的预测效果反而下降。造成这种情况的根本原因是采集到的训练样例存在错误,没有反映真实情况[8]。

根据奥姆剃刀原理,简单的树更好。以 2.5 节中的方法从一棵 DFDT 中提取动态模糊规则集合,对其中过长的规则进行修剪。修剪后的规则集合重新建立一棵新的 DFDT,系统可另选论域成员作为根结点。

统计缺省数目最少的论域成员作为根结点,建立后的决策树可以提高对包含缺省属性输入样例的判断速度。

修剪规则和剪枝有相似的步骤,都必须使修改过的决策树有更好的预测准确率,否则就不做修改。

4　实例验证

将动态模糊决策树用于控制,系统根据室内的温度、湿度等条件决策是否打开空调制冷。人对室内条件的感知是动态模糊的,相同的温度会因为它变化趋势、其他条件而产生完全不同的感觉。采集相关条件属性的动态模糊数据。使用动态模糊决策树和普通的 C4.5 算法进行预测。C4.5 算法中任一条件属性均使

用一个实数来描述,DFDT 算法的条件属性使用一个动态模糊数进行描述。实验如下:

实验数据从数据库中任意抽取一组训练样例,样例数目分别为 30、100、200 组。建立树后分别用训练样例、数据库其他数据进行验证。

(1)使用训练样例验证:

DFDT: 86.21%(30 组) 91.32%(100 组) 92.67%(200 组)

C4.5: 82.10%(30 组) 87.67%(100 组) 88.10%(200 组)

(2)随机从数据库抽取验证样例(修剪之前):

DFDT: 80.14%(30 组) 86.12%(100 组) 87.67%(200 组)

C4.5: 78.30%(30 组) 82.37%(100 组) 83.40%(200 组)

(3)随机从数据库抽取验证样例(修剪之后):

DFDT: 85.07%(30 组) 89.92%(100 组) 91.07%(200 组)

C4.5: 77.10%(30 组) 81.07%(100 组) 81.60%(200 组)

从实验数据中可知,在处理动态模糊性问题时 DFDT 比 C4.5 等算法更加有效。对已建立的 DFDT 修剪后,它的预测准确性比未修剪前更高。

5 结 论

DFS 理论在描述动态模糊信息时,能够有效地表示事物真实情况。而使用静态精确数据描述动态模糊信息时会造成信息缺失。DFDT 高效、准确地处理 DFS 理论表示的训练样例,能够有效地处理动态模糊问题。和模糊决策树不同,DFDT 对任何输入只有唯一的判断结果,匹配算法的时间复杂度低于模糊决策树。

参考文献:

[1] Mitchell T M. 机器学习[M]. 曾华军, 张银奎译. 北京: 机械工业出版社, 2003.

[2] 杨宏伟,赵明华,孙 娟,等. 基于层次分解的决策树[J]. 计算机工程与应用,2003 (23):108-110.

[3] 李凡长,刘贵全, 佘玉梅. 动态模糊逻辑引论[M]. 昆明: 云南科技出版社,2005.

[4] Olaru C, Wehenkel L. A Complete Fuzzy Decision Tree Technique[J]. Fuzzy Sets and Systems, 2003, 138(2): 221-254.

[5] Myles A J, Brown S D. Induction of Decision Trees using Fuzzy Partitions[J]. Journal of Chemometricis, 2003 (17): 531-536.

[6] Aoki K, Watanabe T, Kudo M. Design of Decision Tree Using Class-Dependent Features Subsets[J]. Systems and Computers in Japan, 2005, 36(4):37-47.

[7] 王熙照,孙 娟,杨宏伟,等. 模糊决策树算法与清晰决策树算法的比较研究[J]. 计算机工程与应用,2003(21):72-75.

[8] 李道国,苗夺谦,俞 冰. 决策树剪枝算法的研究与改进[J]. 计算机工程,2005,31 (8):19-21.

(上接第 72 页)

尔函数,不仅可以利用中动态变量交换技术方便地实现变量 x_i 和 x_{i+1} 的交换,而且在得到最优变量序的同时,也构建出了与之对应的 BDD,此外对存储空间的需求也减少了。

从上面的分析中可以看到,把 A* 算法引入到基于 Friedman 的最优变量的排序问题中,在处理器的处理时间上和存储器的空间需求上都有很大的改善,提高了算法的执行效率、验证和测试的生成效率。

4 小 结

将 A* 搜索算法引入到求解最优变量排序的问题中,由于 A* 搜索算法仅对 open 表中具有最小估价函数值的状态进行扩展,使得 2^n 个状态空间的大部分状态空间在搜索的过程中被删除了,同时在该算法引入了暂缓插入的条件和提前结束的条件,使该算法的状态空间得到了更进一步的缩减。因此该算法在搜寻变量最优序的过程中,有更好的执行效率,为最优排序算法向实际的工业应用提供了一个新思路和新方法。

参考文献:

[1] Bryant R E. Symbolic manipulation of Boolean functions using a graphical representation [C] // Proceedings of the 22nd ACM/IEEE conference on Design Automation. [s. l.]: [s. n.], 1985:688-694.

[2] Friedman S J, Supowit K J. Finding the Optimal Variable Ordering for Binary Decision Diagrams[J]. IEEE Trans Computers, 1990, 39(5):710-713.

[3] Hart P, Nilsson N, Raphael B. A formal basis for the heuristic determination of minimum cost paths[J]. IEEE Trans Syst Sci Cybern, 1968, 2:100-107.

[4] Bryant R E. Graph-based algorithms for Boolean function manipulation[J]. IEEE Transactions on Computers, 1986, 35 (8):677-689.

[5] Bollig B, Wegener I. Improving the variable ordering of OBDDs is NP-complete[J]. IEEE Trans on Computers, 1996, 45:993-1002.