一种基于模板的子句学习算法

于 鹏,刘大有,齐 红
(吉林大学计算机科学与技术学院,吉林长春 130012;吉林大学符号计算与知识工程教育部重点实验室,吉林长春 130012)

摘 要: 针对归纳逻辑程序设计中学习子句所遇到的较大搜索空间问题,提出子句模板的概念.用遗传算法先学习子句模板,再结合标记矩阵和按信息增益抽样的方法将其转化为子句.设计了相应的适应度函数及遗传算子.理论分析与实验对比表明,该算法可有效缩小搜索空间,提高学习效率,并且可以学习递归子句,是一种有效的子句学习算法.

关键词: 归纳逻辑程序设计;子句模板;遗传算法;递归子句;信息增益
中图分类号: TP18 **文献标识码:** A **文章编号:** 0372-2112 (2007) 11-2140-06

A Clause Learning Algorithm Based on Template

YU Peng, LIU Da-you, QI Hong
(*College of Computer Science and Technology, Jilin University, Changchun, Jilin* 130012, *China*;
Key Laboratory of Symbolic Computation and Knowledge Engineering of Education Ministry of China, Changchun, Jilin 130012, *China*)

Abstract: For the larger search space when learning clause in Inductive Logic Programming, we defined the clause template. Firstly, we learn the clause templates by Genetic Algorithm, and then convert it to the requisite clauses by combining tag matrix and information gain sampling. We designed the corresponding fitness function and genetic operators. Theoretical analysis and experiment comparison show that this algorithm can reduce the search space, improve the search efficiency and can learn recursion clause. It is an effective clause learning algorithm.

Key words: inductive logic programming; clause template; genetic algorithm; recursion clause; information gain

1 引言

近年来,数据挖掘领域探索关系数据的挖掘问题成为新的研究热点,归纳逻辑程序设计(Inductive Logic Programming, ILP)[1]和统计关系学习(Statistical Relational Learning, SRL)[2]是对该问题展开研究的两个领域,其应用较为广泛,包括基因科学、计算机视觉、语音理解和识别、故障诊断和处理等.ILP 结合逻辑程序和机器学习,适于对关系数据领域进行挖掘和描述.SRL 结合似然推理、逻辑表示和机器学习三方面可对复杂的关系领域进行建模,与 ILP 相比增加了对不确定性的表示及推理机制.SRL 中的许多关系结构采用逻辑子句的形式表示,其学习方法主要借鉴 ILP 技术.

ILP 主要研究如何从关系数据中学习逻辑子句,其学习算法[1]分为泛化、特化,以及结合泛化与特化的方法.由于子句中的项有多种取值形式,直接寻找子句使得搜索空间较大,降低了学习算法的效率,因此目前的研究工作主要集中在如何提高搜索算法的效率[3~5].通常,学习到最优的子句是 NP 难的,常规的确定性搜索方法,易陷入局部极值,不利于获得全局最优解.不确定性的搜索方法(如进化计算等)具有较强的搜索能力,可打破局部极值的束缚,适宜用来解决 NP 难题,将进化计算与 ILP 相结合的研究已展开[6~8],然而这些工作直接搜索子句,不可避免的会面对较大的搜索空间,因此研究缩小搜索空间的方法是有意义的.本文对此进行了研究,定义了抽象的子句模板表示子句结构,并利用遗传算法直接学习子句模板,从而缩小了问题的搜索空间,但子句模板不是待求的子句,为此本文结合标记矩阵和信息增益提出了一种将子句模板转化为子句的方法.理论分析和试验对比表明本文提出的基于模板的子句学习方法具有较高的效率,可学得较好的解.

收稿日期:2006-12-06;修回日期:2007-06-16
基金项目:国家自然科学基金重大项目(No.60496321);国家自然科学基金(No.60373098, No.60573073);国家 863 高技术研究发展计划(No.2006AA10Z245);吉林省科技发展计划重大项目(No.20020303);吉林省科技发展计划(No.20030523);欧盟 TH/Asia Link/010(No.111084)

2　归纳逻辑程序设计

归纳逻辑学习问题[1]可概括为给定实例集(包括正例集 E^+ 和负例集 E^-)和背景知识(实例或子句的集合),找到一个假设 H(子句有限集)使得 H 关于背景知识和实例集满足完备性与相容性.如果所有的正实例都被覆盖,则假设 H 关于背景知识和实例集是完备的;如果没有任何负实例被覆盖,则假设 H 关于背景知识和实例集是相容的.实例集对应的谓词通常称为目标谓词.在 ILP 中子句通常写为 $T \leftarrow C_1, C_2, \cdots, C_m$ 形式,其中 T 是文字称为子句头,每个 C_i 是一个文字,$C_1, C_2, \cdots, C_m$ 表示一个合取式称为子句体.

表 1　谓词 P 的基谓词

(1) $p(mary, spring, rose, mary, tom)$	+
(2) $p(tom, summer, john, john, mary)$	+
(3) $p(rose, spring, rose, tom, mary)$	−
(4) $p(tom, summer, john, john, rose)$	+
(5) $p(mary, spring, rose, rose, john)$	−
(6) $p(tom, summer, tom, tom, rose)$	+
(7) $p(john, summer, mary, mary, tom)$	−
(8) $p(rose, spring, tom, rose, tom)$	−

例如表 1 中 8 个基谓词组成的实例集,其中基谓词(1)、(2)、(4)、(6)组成正例集(标记为"+"),其它为负例集(标记为"−").其中谓词 p 是目标谓词.

3　基于模板的子句学习算法 CLBT(Clause Learning Based on Template)

3.1　概念介绍

文字中的项可能有多种取值形式,例如若 $p(x_1, x_2, \cdots, x_n)$中的每个项的可能取值为变量集合$(y_1, \cdots, y_m)$,则该文字有 m^n 种取值,这使得搜索空间很大.由于谓词中的项都与类型相关,例如表 1 中谓词 p 的第 1,3,4,5 项是人名,而第 2 项是季节.若仅仅描述谓词中项的取值类型而不考虑具体的取值,则寻找这种"谓词"的搜索空间将会缩小,我们提出谓词模板、子句模板的概念以表示这种结构.

定义 1　设在一领域中有集合 Hc 和 C,其中 C 是一常量集合,Hc 是由该领域中所有值域为 C 的变量与函数构成的集合,如果变量 v 值域为集合 $Hc \cup C$,则称变量 v 为关于常量集合 C 的超变量.Hc 中的变量称为 v 类型变量,函数称为 v 类型函数,C 称为 v 类型常量集合.

定义 2　若 $p(x_1, x_2, \cdots, x_n)$是 n 元谓词符号,$v_1, v_2, \cdots, v_n$ 是超变量,则 $p(v_1, v_2, \cdots, v_n)$或者$\neg\ p(v_1, v_2, \cdots, v_n)$称为谓词模板.

定义 3　多个谓词模板由析取符号连接组成的表达式称为子句模板.

3.2　CLBT 算法描述

由 3.1 节的定义可见,搜索子句模板的空间比搜索子句的空间要小得多.基于这种思想,本文采用遗传算法从实例与背景知识中学习子句模板,之后再将学得的子句模板转化为子句.整个算法描述如下:

步骤 1　如果数据库中的正例全部被覆盖,则算法结束;

步骤 2　用遗传算法学习子句模板;

步骤 3　用转换算法对学得的子句模板进行转化,若找不到仅覆盖正例而不覆盖任一负例的子句则转步骤 2;

步骤 4　删除数据库中由符合条件的子句所覆盖的全部正例,转步骤 1;

下面将详细介绍学习子句模板及转化子句模板的算法.

3.3　学习子句模板

用遗传算法学习子句模板需对子句模板进行基因编码、设计相应的适应度函数及遗传操作、产生初始群体等.

基因编码

将子句模板写成蕴含的形式,直接将子句模板体部作为基因编码,例如子句模板 $p_1(x, y), \neg p_2(y), p_3(x, z) \rightarrow p(x, y, z)$对应的基因编码为 $p_1(x, y) | \neg p_2(y) | p_3(x, z)$.

适应度函数

适应度函数作为衡量子句模板好坏的标准,由 ILP 技术可知子句头部覆盖的正例越多,负例越少则越符合标准;子句越长表达的意义可能越复杂,不易理解,因此较长的子句模板应给予惩罚;不同谓词间出现的相同类型的变量越多,该语句可能越有意义;在一个子句中若同一谓词的正负文字都出现,则该子句将为恒真,我们不希望学得这样的子句,然而这种结构的子句模板可能隐藏有递归子句[9],而且递归子句极有可能是最终的目标,因此对该结构的子句模板暂时给予一定的惩罚,以后的进化过程会进一步选择这种结构.基于上述考虑,我们定义适应度函数如下:

定义 4　学习子句模板的遗传算法的适应度函数定义为

$$k\left(\frac{M_a}{M}+\frac{p_h}{p_g}\right)+\frac{1}{N}\left(\sum_{i=1}^{n}(h_i-1)-m\right) \qquad (1)$$

其中,N 表示子句模板中出现的谓词模板个数即子句长度;M_a 表示该子句模板的子句体中出现了多少类超变量;M 表示该子句模板中共出现了多少类超变量;p_h 表示子句模板头部覆盖的正实例个数;p_g 表示子句模板头部覆盖的所有实例(包括正、负实例)的个数;$k>0$ 是调节系数可根据子句模板的长度来取值;m 表示子

句模板中有多少类谓词模板的正负真值同时出现；h_i 表示子句中出现的第 i 类超变量在多少个谓词模板中同时出现.

初始群体 初始群体可以由专家依据先验知识给出，也可采用随机生成的方法.本文采用随机生成的方法，通过限制子句的长度不超过某一正常数 q，随机生成一个小于等于 q 的常数 a 作为子句的长度；从待学习的数据集中随机选取 a 个谓词模板，每个谓词模板的真值随机给定，这样生成一个基因编码.按上述方法生成多个子句模板，组成初始群体.

选择操作 选择操作采用“余数随机选择”的方法[10].该选择方法要求适应度与平均适应度的比值为正数，很显然定义 4 给出的适应度函数满足该条件.

交叉操作 交叉操作采用多点交叉的方法，对待交叉的两个子句模板，选择长度较大的一个，随机选取其中的几个位置与另一子句模板的相同位置进行交叉.例如下列两个子句模板：

(1) $Friend(x,x) \vee Teach(x,x) \vee Course(y)$

(2) $Boy(z) \vee GoodAt(u)$

子句(1)的长度大，若选取其中的 1,3 位置进行交叉，操作后获得如下两条新子句模板：

(1) $Boy(z) \vee Teach(x,x)$

(2) $Friend(x,x) \vee GoodAt(u) \vee Course(y)$

变异操作 针对编码方式，设计了四种变异算子：

(1)向子句基因串中随机加入一个谓词模板；

(2)从子句基因串中随机删除一个谓词模板；

(3)从子句基因串中随机选一谓词模板用其它的谓词模板加以取代；

(4)对子句基因串中的某一谓词模板的真值取反；

对发生变异的个体，每次随机选择上述的一种变异算子作用于基因串.

保留最优个体 标准的遗传算法是不收敛的，主要是因为迭代过程中无法保证上一代最优个体在下一代中得以保留.本文的遗传算法复制当前种群，对获得的复制种群进行选择、交叉、变异操作，之后新一代种群由当前种群与经过遗传操作后的复制种群按个体适应度大小共同产生.这样在种群中保留了最大适应度的个体.

定理 1 子句模板学习算法在进化中较优个体呈指数增长，该算法是收敛的.

证明 证明过程可将上述子句模板学习算法映射为二进制编码的遗传算法.对于子句模板的编码部分，假定整个领域中有 n 类谓词模板外加一个空模板，因此可用 $\log_2^{n+1}$ 个二进制位来表示这 $n+1$ 类谓词模板，谓词模板的真值使用一个二进制 $\log_2^{n+1}+1$ 位表示.最终，每个谓词模板可用位表示.本文设计的交叉与变异算子与二进制编码的遗传算法中多点交叉和多个基因变异本质上是相同的.于是，本文的算法与基于二进制编码的遗传算法是等价的，又因为本文在进化种群中保留了最优个体，由文献[11]定理 1 得证.

3.4 子句模板到子句的转化

由 3.3 节的遗传算法学习得到的仅仅是子句的一种模式，它只表明其中可能包含待求的子句，为此需要对学得的子句模板进行转换.本文假设谓词中的项是函数无关[12]的，因此子句模板到子句转化的本质就是将超变量用变量加以取代.我们利用基谓词中常量的对应关系先对子句头部进行转化，之后再利用子句体与子句头部中项的对应关系来转化子句体.

子句模板头部的转化 在子句模板头部的谓词模板中，对于仅出现一次的超变量，可以直接转为变量，而对于多次出现的超变量，情况比较复杂，因为各超变量对应的变量可能是相同或不同的，称这种关系为变量的对应形式.为获得这种对应形式，我们提出由基项构建针对某类超变量的标记矩阵方法.

定义 5 给定谓词模板 pt 及 pt 的某一基谓词集合 P_c，P_c 中元素的个数为 n_c，pt 中超变量 x 出现的个数为 m，则关于 pt 中超变量 x 的标记矩阵是一 $n_c \times m$ 矩阵，矩阵中第 $i(i \geqslant 1)$行第 $j(j \geqslant 1)$列的值 a_{ij}为 P_c 中第 i 个基谓词的第 j 个 x 超变量位置的常量与前 $j-1$ 个 x 位置的常量比较的结果，如与位置 $h(h<j)$的常量相同则 $a_{ij}=a_{ih}$，否则 $a_{ij}=j-1$，当 $j=1$ 时 $a_{ij}=0$.

子句模板头部的转化算法描述为：

步骤 1 如果子句模板头部的谓词模板不包括多次出现的超变量，转步骤 4；

步骤 2 选择一类在谓词模板中多次出现的超变量，并针对头部所覆盖的正例集构建该超变量的标记矩阵；

步骤 3 将标记矩阵中的各行分为不同的类，并计算出现的概率，按概率超过规定阈值的各类标记行将超变量转化为变量，转步骤 1；

步骤 4 如果谓词模板中有仅出现一次的超变量，则直接将其转化为变量；

例 1 表 1 中谓词模板 $p(t_1,t_2,t_1,t_1,t_1)$及由基谓词(1)、(2)、(4)、(6)组成的集合，针对超变量 t_1 的标记矩阵为：

$$\begin{vmatrix} 0 & 1 & 0 & 3 \\ 0 & 1 & 1 & 3 \\ 0 & 1 & 1 & 3 \\ 0 & 0 & 0 & 3 \end{vmatrix}$$

针对某一超变量 x 的标记矩阵，可表示矩阵中所

有对应的基谓词中 x 类型变量的对应形式,定理 2 可保证这一点.

定理 2　谓词模板 pt 的某一有限基谓词集合 C 所对应的关于超变量 x 的标记矩阵,可表示集合 C 中所反映的 x 类型变量的所有对应形式.

证明:用反证法.假设有一种变量的对应形式没有在标记矩阵中描述出来,由于对应形式至少由集合 C 中的一个基谓词所反映,而该基谓词对应标记矩阵中的一行,因此该行中必有两个变量的对应形式标记错误,由于两个变量的对应形式只能是相同和不同,下面分别讨论:若真实的对应形式为两个变量相同,而在标记矩阵中却被标记为不同的两个整数,这必然是基谓词中的两个常量不同所致,可见集合 C 反映不出这种对应形式,与事实矛盾;若真实的对应形式为两个变量不同,而在标记矩阵中却被标记为相同的,这必然是每一包含这一对应形式的基谓词中的两个常量都相同所致,可见集合 C 中反映不出这种对应形式,因此是不可能的.综上,定理 2 得证.

我们将矩阵的相同行分为一类,每一类对应变量的一种对应形式,计算每类标记行在整个矩阵中出现的概率,若该类标记行的概率大于规定的阈值(可预先设定),则该类标记行可转化为变量,否则抛弃.例 1 中有三类标记行:0103、0113、0003,假定只有 0113 超过阈值,则该类标记行对应的变量为:x, y, y, z.在实际应用中可将谓词模板中所有的超变量构建在一个标记矩阵中,不必为每类多次出现的超变量分别构建标记矩阵.

子句模板体部的转化　子句模板头部经过转化后,其中的超变量将全部转化为变量,如果将头部的变量与子句模板体中的相应超变量进行对应则可将子句模板体的谓词模板转化为具体的文字.由于子句模板体中同一个超变量可能出现多次,因此无法确定哪一超变量与头部的具体变量对应.一种简单的解决方法是列举出基子句来进行对应,这可能需要所覆盖正例的全部基子句,通常是难以做到的.但是,若能抽取一些有代表性的基子句进行对应,则是一种可行的方法.谓词中的项可视为属性,如果可以确定哪个属性能够较好的用来区分被子句覆盖的正负实例,则针对这一属性来抽取基子句,从而比较超变量的对应关系是比较恰当的.利用信息增益[13]可以达到这一目的,针对某一谓词属性的信息增益的定义如下:

定义 5　包括属性 A 的谓词可以根据属性 A 的值将基谓词划分为多个子集 $E_1, \cdots, E_e$,其中 A 可以有 e 个不同的值.每个子集 E_i 包含着 p_i 个正例和 n_i 个反例,属性 A 的信息增益是原始信息需求和新的信息需求之间的差异,其表达式为:

$$Gain(A) = I\left(\frac{p_c}{p_c + n_c}, \frac{n_c}{p_c + n_c}\right) - \text{Remainder}(A) \quad (2)$$

其中

$$\text{Re } mainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p_c + n_c} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right),$$

$I(p_1, p_2)$是期望信息量其计算公式为 $I(p_1, p_2) = -p_1\log_2(p_1) - p_2\log_2(p_2)$,式中 p_c 表示谓词的正例个数,n_c 表示谓词的负例个数.

例 2　对于表 1 所列出的谓词 $p(x_1, x_2, x_3, x_4, x_5)$ 的 8 条基谓词,我们分别计算变量 x_1, x_2, x_3 的信息增益为 $Gain(x_1) = 0.75$, $Gain(x_2) = 0.189$, $Gain(x_3) = 0.4$, $Gain(x_4) = 0.5$, $Gain(x_5) = 0.4$.以我们的常识来看,属性 x_1 用来划分比较合理,这与信息增益计算的结果相同.

子句模板体部的转化算法描述为:

步骤 1　按基谓词计算子句模板头部谓词中各属性的信息增益;

步骤 2　选取头部谓词中具有最大信息增益的属性(若有多个相同最大信息增益的属性可任选一个),获取其在被覆盖的正实例中出现的常量集 D.

步骤 3　对常量集 D 随机选取若干常量,在正例中选择一些包含这些常量的基谓词(可根据正例的个数选择)作为基子句头部,按照子句模板的结构从背景知识中构建基子句.将这些基子句进行关系最小一般泛化(rlgg)[1]获得一子句,则算法结束.

若子句模板体转化完成后获得的子句仍覆盖负例,则可重新选择常量和基谓词,并产生不同的基子句再进行 rlgg.为避免该过程无限进行下去,可设一最大选择次数,若达到该最大次数仍覆盖负例,则认为子句不符合要求,选择下一子句头部(转化子句模板头部可能产生多个子句头)再转化子句模板体部.

子句模板转化算法的复杂性分析　首先,分析构建标记矩阵的复杂度,假设谓词模板有 m 个基实例,其中某一类超变量在头部同时出现 $n(n > 1)$次,可见构建标记矩阵中的一行最坏的情况下需进行 $n(n-1)/2$ 次比较运算,因此构建这类超变量的复杂度为 $O(mn^2)$.由定义 5,转化子句模板体时计算信息增益的复杂度为 $O(n^2\log^n)$.对于抽样子句操作,算法的复杂度主要由测试覆盖的算法复杂度来决定,由于不同的覆盖测试算法的复杂度可能不同[1],因此设其为一函数形式 $O(f(n))$.由上述分析,整个子句模板转化算法的复杂度为 $O(mn^2 + n^2\log n + f(n))$.

去除冗余结构　最终获得的子句可能存在一些冗余结构,例如,同一文字在子句中多次出现、子句体中出现与其它文字没有公共项的文字等.为此,采用下面方法来消除子句的冗余结构,获得简洁的子句:(1)如

果子句体中同一文字出现多次,则只保留其中的一个;(2)如果子句体中出现与子句头相同的文字,则删除子句体中的该文字;(3)如果子句中的某一文字与其它文字没有公共的项,则删除该文字.

4 实验对比

我们作了两类实验,第一类实验将本文的 CLBT 学习算法与文献[8]中的 GILP 算法(GILP 算法是将文字进行二进制编码来学习子句的遗传算法)关于同一数据集进行学习比较,以检验 CLBT 学习的性能与有效性;第二类试验用 CLBT 学习递归子句,以检验学习递归子句的有效性.

表 2 UW-CSE 中的 3 条子句

$TaughtBy(c,p,q) \wedge CourseLevel(c,level\_500) \rightarrow Professor(p)$
$TempAdvisedBy(p,s) \rightarrow Position(s,faculty)$
$TA(c,p,q) \vee AdvisedBy(p,s) \rightarrow Student(p)$

实验 1 实验数据是由参考文献[12]中的 UW-CSE 数据库抽取的两组实验数据集:数据集一是同时满足表 2 中三条子句的 UW-CSE 中的 441 个基谓词,三条子句中 c、q、p、s 是变量,$level\_500$、$faculty$ 是常量;数据集二是去掉 UW-CSE 中 10 类带有 *same* 字样的等价谓词后的全部基谓词.以上两数据集的负例由封闭世界[13]假设来产生.

评价学习结果的优劣使用覆盖度来度量,其定义为 $\frac{p^*}{p^*+n^*}$,其中 p^* 表示子句覆盖的正实例数,n^* 表示子句覆盖的负实例数.

对数据集一,两种学习算法的交叉概率取 0.6,变异概率取 0.1,初始种群设为 6,规定子句的最大长度为 4,CLBT 转化算法抽样 4 条基子句.每种算法连续学习 6 次取平均覆盖度,试验结果如图 1 所示.

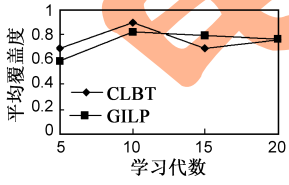

图 1 CLBT 与 GILP 对数据集一的学习结果

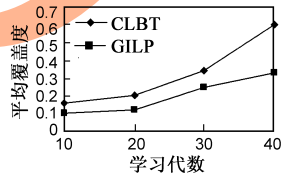

图 2 CLBT 与 GILP 对数据集二的学习结果

对数据集二,两种学习算法的交叉概率取 0.7,变异概率取 0.1,初始种群设为 10,规定子句的最大长度为 4,CLBT 转化算法抽样 20 条基子句.每种算法连续学习 6 次取平均覆盖度,试验结果如图 2 所示.

图 1 与图 2 所示的曲线表明,两种学习算法在参数相同以及学习相同代数的条件下,对于文字较少的数据集,获得的学习结果接近,而对于文字较多的数据集,CLBT 算法学得的结果更好,效率更高.

实验 2 对于递归子句的学习我们使用文献[9]的八边形实例集:设$(a1,b1,\cdots,h1)$和$(a2,b2,\cdots,h2)$为两个八边形,背景谓词 *succ* 和 *next* 表示其点的顺序关系,目标谓词 *map* 为两点之间的映射关系;背景知识:$succ(a1,b1)$, $succ(b1,c1)$, $succ(c1,d1)$, $succ(d1,e1)$, $succ(e1,f1)$, $succ(f1,g1)$, $succ(g1,h1)$, $succ(h1,a1)$, $next(a2,b2)$, $next(b2,c2)$, $next(c2,d2)$, $next(d2,e2)$, $next(e2,f2)$, $next(f2,g2)$, $next(g2,h2)$, $next(h2,a2)$;目标基谓词:$map(a1,a2)$, $map(b1,b2)$, $map(c1,c2)$, $map(d1,d2)$, $map(e1,e2)$, $map(f1,f2)$, $map(g1,g2)$, $map(h1,h2)$,反例按封闭世界假设给定.CLBT 算法在交叉概率 0.6,变异概率 0.1,初始群体 10,子句体最大长度为 4 时,连续学习 50 代,运行多次 CLBT 算法得到的多数子句模板为:

$$next(sv,sv), map(sv,sv), succ(sv,sv) \rightarrow map(sv,sv)$$

对子句头构建标记矩阵获得头部文字 $map(v1,v2)$,属性 $v1$,$v2$ 的信息增益相同,针对 $v1$ 属性抽样 4 条基子句泛化后获得子句

$$next(v4,v2), map(v3,v4), succ(v3,v1) \rightarrow map(v1,v2)$$

其中 $v1$, $v2$, $v3$, $v4$ 是变量,该子句与文献[9]的 FOILPlus 算法获得的结果相同.可见,本文算法可以学习递归子句.

5 结论

本文提出了一种通过子句模板来获得子句的算法,该算法首先使用遗传算法来学得子句模板,然后基于标记矩阵和信息增益方法抽取基子句进行泛化,进而将子句模板转化为子句.理论分析与实验对比表明,该算法可有效地学习子句,尤其适宜对文字较多的数据集进行学习.未来的研究工作可结合其它搜索算法更加快速高效地发现子句模板,以及提出新的子句模板转化方法,进一步提高子句学习算法的效率.

参考文献:

[1] N Lavraˇc, S Dˇzeroski. Inductive Logic Programming: Techniques and Applications [EB/OL]. http://www-ai.ijs.si/SasoDzeroski/ILPBook/, 1994.

[2] L De Raedt, K Kersting. Probabilistic logic learning[A]. ACM-SIGKDD Explorations: Special Issue on Multi-Relational Data Mining[C]. New York: ACM Press, 2003.5(1).31 – 48.

[3] Ozaki Tomonobu, Furukawa Koichi, Rouveirot Céline, Sebag Michèle. Application of pruning techniques for propositional learning to progol[A]. Inductive Logic Programming: 11th International Conference: ILP 2001 [C]. Strasbourg: Springer, 2001.206 – 219.

[4] J Struyf, H Blockeel. Query optimization in inductive logic pro-

gramming by reordering literals[A]. Proceedings of the 13th International Conference on Inductive Logic Programming[C]. Szeged: Springer, 2003. 329 – 346.

[5] Maloberti Jérome, Sebag Michèle. Fast theta-subsumption with constraint satisfaction algorithms[J]. Machine Learning, 2004, 55(2): 137 – 174.

[6] Po Shun Ngan, Man Leung Wong, Kwong Sak Leung, et al. Using grammar based genetic programming for data mining of medical knowledge[A]. Proc of the 3rd Annual Genetic Programming Conf[C]. San Francisco, CA: Morgan Kaufmann, 1998. 254 – 259.

[7] A Tamaddoni-Nezhad, S H Muggleton. Searching the subsumption lattice by a genetic algorithm[A]. Proceedings of the 10th International Conference on Inductive Logic Programming[C]. London: Springer-Verlag, 2000. 243 – 252.

[8] 杨新武,刘椿年. 遗传归纳逻辑程序设计的个体编码生长现象[J]. 计算机研究与发展. 2003, 40(8): 1238 – 1243.
Yang Xinwu, Liu Chunnian. Growth phenomenon of individuals' code length in genetic inductive logic programming[J]. Journal of Computer Research and Development, 2003, 40(8): 1238 – 1243. (in Chinese)

[9] 张润琦,陈小平,刘贵全. 一个不受常量限制的归纳逻辑程序设计方法[J]. 软件学报. 1999, 10(8): 868 – 876.
Zhang Runqi, Chen Xiaoping, Liu Guiquan. An ILP algorithm without restriction of constant ordering[J]. Journal of Software, 1999, 10(8): 868 – 876. (in Chinese)

[10] Goldberg D E. Genetic Algorithms in Search, Optimization, and Machine Learning[M]. Boston: Addison-Wesley, 1989.

[11] 陈国良,王煦法,庄镇泉,王东生. 遗传算法及其应用[M]. 北京:人民邮电出版社, 1996.

[12] S Kok, P Domingos. Learning the structure of markov logic networks[A]. Proceedings of the Twenty-Second International Conference on Machine Learning[C]. Bonn: ACM Press, 2005. 119. 441 – 448.

[13] S Russell, P Norvig. Artificial Intelligence: A Modern Approach[M]. New Jersey: Pearson Education. 1995.

作者简介:

于　鹏　男,1979 年生于吉林省柳河县,博士研究生,现就读于吉林大学计算机科学与技术学院. 主要研究方向:进化算法、机器学习等. E-mail: yu_peng79@126.com

刘大有　男,1942 年生于河北省乐亭县,吉林大学计算机科学与技术学院教授,博士生导师. 主要研究方向:知识工程与专家系统、分布式 AI 与多 Agent 系统、不确定性推理、空间推理与 GIS 应用等. E-mail: dyliu@jlu.edu.cn

齐　红　女,1970 年生于吉林省吉林市,副教授. 主要研究方向:数据挖掘等.
E-mail: qihong@jlu.edu.cn