基于FOIL算法的一阶规则集学习器设计方法

徐 彤，张 莉，谢 波
（空军工程大学 导弹学院，陕西 三原 713800）

摘 要：对FOIL算法进行了深入剖析，提出了一种基于该算法利用Visual Prolog实现一阶规则集学习器的设计方法，给出了实现学习器的关键代码和试验结果，验证了该方法在一阶规则集提取中的有效性。

关键词：FOIL；一阶规则；序列覆盖；Horn子句；Visual Prolog
中图分类号：TP312　　**文献标识码**：A　　**文章编号**：1009-3516(2005)06-0080-04

从样例数据中学习规则是数据挖掘的一项重要内容。学习规则集合的一种方法是首先学习决策树，然后将所学到的决策树转化为等价的规则集合[1]。另一种方法是使用遗传算法，将位串按照规则编码，然后通过遗传操作数来搜索整个目标概念的假设空间。但是这两种算法学习到的规则是命题规则，不能含有变量[2]。以序列覆盖算法为核心的一组算法解决了该问题，它们可以学习包含变量的一阶规则集。由于一阶规则具有强大的表示能力，所以这些算法在数据挖掘中显得十分有效。FOIL算法便是这些算法中较为典型的一个。

Visual Prolog是基于一阶Horn子句的编程语言和环境，是目前国际上研究和开发智能化应用的主流工具之一[3]。用Visual Prolog来编写一阶规则集的学习器，将能充分发挥其模式匹配、合一、回溯等内在优势，这是其它语言所不具备的。本文将在深入剖析FOIL算法的基础上，探讨用Visual Prolog编写一阶规则集学习器的方法。

1 FOIL算法剖析

在许多学习析取规则集的算法中，都体现了序列覆盖的思想。序列覆盖算法的策略为：学习一个规则，移去它覆盖的训练数据，再重复这一过程。Clark Niblett(1989)提出的CN2算法就采用了这样的思想，然而该算法仍然只适用于学习命题规则集。FOIL算法扩展了CN2算法，可以处理类似于Horn子句的一阶规则集学习问题，但它所学习到的规则与Horn子句有两处不同：文字不允许包含函数符号；文字可以为负文字。

基本的FOIL算法如下。

符号约定：目标谓词 p，谓词集 $p\{\}$，样例 E，正例集 $E+$，反例集 $E-$，已学到的规则集 $r\{\}$，新规则 r，
备选文字集 $l\{\}$，最优文字 l，新规则覆盖的反例集 $rE-$，空集 $\{\}$，赋值 =，不等于 ! =

算法：foil(p,$p\{\}$,E)
{ $E+$ = E 中使 p 为真的成员集。$E-$ = E 中使 p 为假的成员集。$l\{\}$ = $\{\}$。
while($E+$! = $\{\}$)
{ r = p:-{}。$rE-$ = $E-$ 。
while($rE-$! = $\{\}$)
{ $l\{\}$ = 基于 $p\{\}$，按原则 principle 生成的备选文字集

收稿日期：2005-05-27
基金项目：国防科技预研基金资助项目(51406050301DZ01)
作者简介：徐 彤(1978-)，男，山东菏泽人，博士生，主要从事智能决策、自动推理软件研究.

$l = \arg\max_{literal \in l\{\}} \text{foil\_gain}(\text{literal}, r)$

把 l 加入 r 的前件。$rE- = rE-$ 中满足 r 前件的子集

}

$r\{\} = r\{\} + r$。$E+ = E+$ 移去被 r 覆盖的正例。

}

}

其中,外层循环对应于序列覆盖算法,它每次使用内层循环学习一条新规则,然后将新规则覆盖的所有正例移去,再学习下一条规则。内层循环集中于解决如何学习一条新规则的问题,与 CN2 算法相比,FOIL 算法已经被扩展以适合处理一阶规则,这正是 FOIL 算法与 CN2 算法的不同之处。

FOIL 算法的内层循环执行的是一般到特殊的爬山搜索,它开始于一个前件为空的规则 r。由于 r 前件为空,所以 r 在任何情况下都成立,因此该规则是最一般的(能够接受所有样例)。在循环的过程中,算法通过一个评判机制不断地在一组候选文字中挑选最适合的文字加入规则 r 的前件中,于是 r 被不断地特殊化(有越来越多的反例被规则拒绝),直到规则 r 拒绝所有反例,内层循环停止,一条新规则学习完毕。

算法中,principle 原则用于生成备选文字集,这个原则的具体内容为:①文字可以为规则引入新的变量,但至少包含一个当前规则中的已有变量;②文字中的变量全部都是当前规则中的已有变量;③满足上述条件的文字的负文字。

在对最优文字的评判法则中,foil_gain(literal,r)为

$$\text{foil\_gain}(\text{literal}, r) \equiv t\left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0}\right) \tag{1}$$

其中:p_0 为规则 r 覆盖的正例个数;n_0 为规则 r 覆盖的反例个数;p_1 为规则 r 的前件加入文字 literal 后;新规则覆盖的正例个数;n_1 为新规则覆盖的反例个数;t 为新规则和原规则 r 都能覆盖的正例个数。

从信息论的角度来看,由于规则 r 的前件加入了文字 literal,产生的新规则比原规则 r 更加特殊,覆盖的正例更少,因此对正例进行编码所需的二进制位数减少。式(1)中减号前的对数项表示对新规则覆盖的正例进行编码所需的最少二进制位数的相反数;减号后的对数项表示对原规则 r 覆盖的正例进行编码所需的最少二进制位数的相反数。因此,foil_gain(literal,r)的含义为:要表达新规则和原规则 r 共同覆盖的 t 个正例,所需的总二进制位数的减少量。可以看出,FOIL 算法认为在备选文字中使这个减少量最大的文字为最优文字。这样的选择意味着算法属于一种无回溯的"贪心"搜索。

基于上述分析可知,FOIL 算法本质上采用了一般到特殊和特殊到一般的双向搜索策略进行一阶规则集的学习。对于整个一阶规则集来说,搜索按特殊到一般序进行:从最特殊的规则集开始(空集,拒绝所有规则),随着新规则的不断加入而逐渐泛化;对于每一条规则的学习来说,搜索按一般到特殊序进行:从最一般的规则开始(规则前件为空集,接受所有样例),随着规则前件的不断增加而逐渐特殊化。在训练数据无噪声的情况下,FOIL 可持续增加新文字到规则中,直到规则不覆盖任何反例为止。

2　FOIL 学习器的设计

通过上面对 FOIL 算法的分析可以看出,在一阶规则集的学习过程中,包括大量对集合的操作,如正例集、反例集、规则集、备选文字集、新规则覆盖的反例集等,而且还涉及到了一阶谓词的表示等问题。Visual Prolog 本身是基于一阶 Horn 子句的语言,而且其自身的回溯、模式匹配、合一机制为集合操作提供了很大方便,因此,选择 Visual Prolog 实现 FOIL 学习器是合适的。

2.1　论域

由于要实现通用的学习器,学习器本身无法预测学习任务中变量、常量、谓词、规则、事实的有关信息,所以对这些基本元素的表示不能像普通的 Visual Prolog 程序那样直接进行,需要重新定义以适应可变的变量名称、变量个数、谓词名称等。下列代码:var = var(integer) / * 变量 * /; con = symbol / * 常量 * /; varList = var * / * 变量列表 * /; conList = con * / * 常量列表 * /; literal = symbol / * 文字 * /; condition = condition (literal, varList, boolean); conditionList = condition * ; boolean = true; false 展示了本文对学习器基本论域的定义方法,其中定义了变量、常量、变量表、常量表、文字、规则前件、规则前件表和布尔值(在表示一个文字是否

为谓词名的否定时使用)。

2.2 事实

在对一阶规则集学习的过程中,学习任务的初始描述、中间状态、学习到的规则等都需要借助事实库保存起来,因此必须定义一组基本事实。本文所设计的学习机实现中使用的主要事实谓词有:nondeterm fact(literal,conList);nondeterm rule(literal,varList,conditionList);nondeterm factName(literal,integer);nondeterm consName(symbol);nondeterm newLiteral(literal,varList,boolean);nondeterm varName(var);nondeterm varValue(var,symbol);single varNameExist(boolean)其中,fact(literal,conList)表示了以 literal 为谓词名,参数列表为 conList 的谓词;rule(literal,varList,conditionList)表示了以 conditionList 为前件,以 literal(varList)为后件的规则;factName(literal,integer)表示待学习的样例中,包含了一个名为 literal 带有 integer 个参数的谓词;consName(symbol)表示待学习的样例中有一个 symbol 常量;其他事实用于保存学习规则的过程中产生的新文字、规则中已经使用的变量、规则覆盖的正例、规则覆盖的反例以及验证新规则覆盖正例和反例的过程中,变量与常量的匹配情况等。

2.3 谓词

只有编写恰当的谓词子句并进行正确调用才能实现学习功能。在本文所设计的学习器中,定义并实现的主要谓词功能包括:产生与规则中所含变量不同的新变量;产生新的文字集;从文字集中挑选最优文字;将规则前件与最优文字合并为新的规则前件;把规则在样例中进行变量与常量的匹配,并把匹配情况、成功匹配的个数、不能成功匹配的个数插入事实库,同时生成正例集、反例集;从事实库中删除成功匹配的样例。由于涉及到许多对集合的操作,所以在这些谓词的实现中大量的使用了回溯机制。下面是用于学习一条一阶规则的谓词子句。

```
声明:procedurestudy(literal,varList,conditionList);
实现:study(Literal,ParamList,ConditionList):-
                assert(rule(Literal,ParamList,ConditionList)), insertVarList(ParamList),
                generateNegSets(),negCaseNum(X) ,X<>0,generateNewLiteral(),selectLiteral(L),
                NewList = appendConditionList(ConditionList, L),
                study(Literal,ParamList, NewList).
    study(_,_,_).
```

其含义是:在第一个子句中,首先将目前学习到的规则和规则中使用的变量列表插入事实库(其他谓词子句将会使用这些信息),然后生成当前规则的反例集,再从事实库中得到反例集中包含反例的个数。如果反例个数为 0,则回溯到第二个子句,学习成功;如果不为 0,则产生新的文字集,从中挑选最优文字,将最优文字与当前规则前件合并,最后利用尾部递归开始新一轮最优文字的搜索。

3 实例

对于下面一组样例:

father(a,b) father(a,c) father(b,d) female(d) grandDaughter(d,a)

如果学习的目的是要找到能推导出 grandDaughter(X,Y)的规则,那么这个学习任务在本文所设计的规则集学习器中,将被表示为

```
fact(grandDaughter,[d,a]).fact(father,[a,b]).fact(father,[a,c]).fact(father,[b,d]).
fact(female,[d]). factName(father,2).factName(female,1).
consName(a).consName(b).consName(c).consName(d).
rule(grandDaughter,[var(1),var(2)],).
```

实验表明,本文设计的学习器能够利用上面对学习任务的表示方法,成功的学习到规则:

```
rule(grandDaughter,[var(1),var(2)],[condition(female,[var(1)],true),condition(father,[var(3),var(1)],true),condition(father,[var(2),var(3)],true)]).
```

即 female(x1) ∧father(x3,x1) ∧father(x2,x3)→grandDaughter(x1,x2),这个结论显然是正确的。

4　结束语

以上深入分析了FOIL算法,讨论了利用Visual Prolog实现基于FOIL的一阶规则集学习器的设计方法。对于其他以FOIL算法为基础的改进算法,可以本文所设计的学习器为核心,进行适当改动来得到对应的学习器。

参考文献:

[1] QUINLAN J R. Learning logic Definitions From Relations[J]. Machine Learning, 1990,5:239-266.
[2] MITHELL T M. 机器学习[M]. 北京:机械工业出版社, 2003.
[3] 雷英杰. Visual Prolog语言教程[M]. 西安:陕西科学技术出版社, 2002.
[4] 雷英杰. Visual Prolog 编程、环境及接口[M]. 北京:国防工业出版社,2004.
[5] 雷英杰, 王　涛, 赵　晔. Visual Prolog的回溯机制分析[J]. 空军工程大学学报(自然科学版), 2004, 5(5): 80-84.
[6] 雷英杰, 王宝树, 赵　晔, 等. Visual Prolog的搜索控制机制分析[J]. 计算机科学, 2005, 32(3): 39-43.

(编辑:田新华)

Design of Learner for Sets of First - Order Rules Based on FOIL

XU Tong, ZHANG Li, XIE Bo
(The Missile Institute, Air Force Engineering University, Sanyuan, Shaanxi 713800, China)

Abstract: After a thorough analysis of the essence of FOIL algorithm, the paper discusses the way of programming a learner by using Visual Prolog and based on this algorithm, and at the same time demonstrates the key code and experiment results of this method in detail.

Key words: FOIL; first - order rules; sequential covering; horn clause; Visual Prolog

(上接第62页)

[2] 雷英杰,张　雷,邢清华,等. Visual Prolog语言教程[M]. 西安:陕西科学技术出版社,2002.
[3] 雷英杰,邢清华,孙金萍,等. Visual Prolog 编程、环境及接口[M]. 北京:国防工业出版社,2004.
[4] 雷英杰,王　涛,赵　晔. Visual Prolog的回溯机制分析[J]. 空军工程大学学报(自然科学版), 2004, 5(5): 80-84.
[5] 雷英杰,邢清华,王　涛,等. 人工智能程序设计(面向对象语言)[M]. 北京:清华大学出版社,2005.
[6] 雷英杰,王宝树,赵　晔,等. Visual Prolog的搜索控制机制分析[J]. 计算机科学, 2005, 32(4): 39-43.
[7] 雷英杰,华继学,徐　彤,等. Visual Prolog截断机制对回溯的作用机理[J]. 计算机工程, 2005, 31(8): 39-43.

(编辑:田新华)

On the Mechanisms of Interface Score Qualifications in Visual Prolog

TIAN Ye, LEI Ying-jie, JI Bo, SUN Chen
(The Missile Institute, Air Force Engineering University, Sanyuan, Shaanxi 713800, China)

Abstract: To the particular scores of interfaces in Visual Prolog, the specific functions and effects of mechanisms of support qualifications and open qualifications are analyzed and investigated in detail on the basis of study of principal issues of mechanisms of these qualifications. First, the fundamental notions of interfaces and objects are expatiated. Then, the functional mechanisms of support qualifications and open qualifications are exposed with a detailed analysis. Finally, the essential characteristics and application mechanisms of the interface score qualifications are exposed and illustrated through an instance.

Key words: AI; expert systems; programming language; programming in logic; Visual Prolog