计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20180759
56(1):138-154, 2019

归纳逻辑程序设计综述

戴望州　　周志华
(计算机软件新技术国家重点实验室(南京大学)　南京　210023)
(daiwz@lamda.nju.edu.cn)

A Survey on Inductive Logic Programming

Dai Wangzhou and Zhou Zhihua
(*National Key Laboratory for Novel Software Technology* (*Nanjing University*), *Nanjing* 210023)

Abstract　Inductive logic programming (ILP) is a subfield of symbolic rule learning that is formalized by first-order logic and rooted in first-order logical induction theories. The model learned by ILP is a set of highly interpretable first-order rules rather than black boxes; owing to the strong expressive power of first-order logic language, it is relatively easier to exploit domain knowledge during learning; the learned model by ILP can be used for modeling relationships between subjects, rather than predicting the labels of independent objects. However, due to its huge and complicated underlying hypothesis space, it is difficult for ILP to learn models efficiently. This paper tries to review most of the current researches in this area. Mainstream ILP approaches are introduced according to different categorizations of first-order logical induction theories. This paper also reviews the most recent progress in the ILP area, including ILP techniques based on second-order logical abduction, probabilistic inductive logic programming (PILP) and the ILP approaches that introduce differentiable components. This paper also introduces some representative applications of ILP approaches in practical problems, and then talks about its major challenges, and finally discusses about the prospects for future research directions.

Key words　machine learning; first-order logic; rule learning; inductive logic programming (ILP); probabilistic inductive logic programming (PILP)

摘　要　**归纳逻辑程序设计**(inductive logic programming, ILP)**是以一阶逻辑归纳理论为基础,并以一阶逻辑为表达语言的符号规则学习方法. ILP 学得的模型是易于理解的一阶逻辑符号规则,而非难以解释的黑箱模型;在学习中可以相对容易地显式利用以一阶逻辑描述的领域知识;学得模型能对领域中个体间的关系进行建模,而非仅仅对个体的标记进行预测. 然而,由于潜在假设空间巨大,进行高效学习有相当的困难. 综述了 ILP 领域的研究情况,从不同一阶逻辑归纳理论的角度对主流的 ILP 方法做出了梳理. 还介绍了近年来 ILP 基于二阶诱导推理理论的扩展、基于概率的扩展和引入可微构件的扩展. 最后,介绍了 ILP 在实际任务中的代表性应用,探讨了 ILP 方法目前所遇到的挑战,并对其未来发展进行了展望.**

收稿日期:2018-11-09;**修回日期**:2018-11-27
基金项目:**国家重点研发计划项目**(2018YFB1004300);**国家自然科学基金项目**(61751306)
This work was supported by the National Key Research and Development Program of China (2018YFB1004300) and the National Natural Science Foundation of China (61751306).
通信作者:**周志华**(zhouzh@nju.edu.cn)

关键词 机器学习;一阶逻辑;规则学习;归纳逻辑程序设计;概率归纳逻辑程序设计

中图法分类号 TP391

规则学习是机器学习中以符号规则为模型表达方式的一个分支,主要属于符号主义学习的内容.在文献[1]中有对规则学习的基本介绍,本文主要关注以一阶逻辑(first-order logic, FOL)为表达语言的归纳逻辑程序设计(inductive logic programming, ILP)[2].一阶逻辑是一种形式系统[3].与只能陈述简单命题的命题逻辑相比,FOL 引入了谓词、函数和量词等更多词汇,它可以通过谓词来量化地陈述命题,因此也被称为一阶谓词演算.

基于文献[1]的界定,ILP 指以 FOL 归纳理论为基础,以 FOL 规则为模型的机器学习方法.与其他种类机器学习方法相比,ILP 能够在学习过程中以 FOL 规则的形式引入复杂领域知识,并能自然地处理学习问题中的复杂推理.此外,它学得的模型具有较好的可理解性,这使得用户和领域专家能够方便地对学习成果进行解读和研究.

1 发展历程

"归纳逻辑程序设计"的命名来自于 Muggleton[2] 于 1991 年发表的奠基性论文以及同年召开的第一届国际归纳逻辑程序设计研讨会,它是如今国际归纳逻辑程序设计会议(International Conference on Inductive Logic Programming)的前身.

20 世纪七、八十年代出现并流行的专家系统[4]极大地推广了人工智能实际应用.专家系统的知识库一般由一组符号逻辑规则构成.早期专家系统中的规则必须由领域专家提供,这不仅成本高,在许多任务中还由于专家知识难以总结或专家不愿分享知识而难以完成,形成了所谓"知识工程瓶颈".为了突破这个瓶颈,出现了许多以符号逻辑表达语言的机器学习方法[1].

一般来说,此类方法存在 3 种局限[2]:1)它们学得的模型一般为命题逻辑规则.受限于模型的表达能力,难以描述对象之间的"关系"(relation),而在许多任务中关系信息非常重要.2)命题学习的输入一般为属性-值(attribute-value).许多领域知识难以被表示为这种形式,导致学习过程不易利用领域知识.3)命题规则只能由一组固定的属性特征所对应的原子构成,学习过程不能自主地对原始特征空间进行变换以弥补其不足.

为了解决这些问题,Muggleton[2] 指出应当使用表达能力更强的 FOL 作为机器学习的形式化语言.逻辑程序(logic programming)[5]作为计算机技术中应用最广泛的 FOL 系统之一,理所当然地成为了首选.由于机器学习是一种从数据中自动"归纳"(induction)的方法,因此它与逻辑程序设计结合而诞生的学科便被命名为"归纳逻辑程序设计".

事实上,早在 20 世纪 70 年代就已出现一些关于 FOL 归纳理论的先驱性工作[6-9],例如对 FOL 归纳学习可行性的证明,为 ILP 的出现奠定了基础.其中 Plotkin[6] 提出的"最小一般泛化"和"相对最小一般泛化"后来成为 ILP 的基础运算.20 世纪 80 年代中期,开始出现 Marvin[10] 等初步运用 FOL 归纳的机器学习算法.20 世纪 80 年代后期出现了许多关于 FOL 归纳理论的研究,例如 Muggleton 的逆归结[11-13] 以及 Rouveirol 和 Puget[14] 的工作等.Muggleton[12] 首次提出使用 FOL 归纳来实现一阶谓词发明[12],使得机器学习过程能自动对不够完善的背景知识进行弥补,这期间也出现了最早的 ILP 系统 Cigol[12].

在其诞生之初的 1990 年代,ILP 的理论研究取得了大量进展.Muggleton 提出了更完备的 FOL 归纳理论——逆语构蕴涵[15] 和逆语义蕴涵[16-17],Nienhuys-Cheng 和 De Wolf[18] 则对 ILP 使用的规则搜索算法进行了详细探讨.ILP 的可学习性[19]也受到大量研究者们的关注.其中 Džeroski 等人[20-21] 与 Cohen[22] 分别证明了某些规则形式受约束的 ILP 问题是 PAC 可学习的;Kietz[23] 则证明了一般情况下的 ILP 不是 PAC 可学习的.除了理论探索,研究者们还提出了大量 ILP 算法,包括 Muggleton 等人基于相对最小一般泛化[7] 提出的 Golem[24]、基于逆语义蕴涵提出的 Progol[16]、Lavrač等人[25] 基于命题化方法提出的 LINUS,以及 De Raedt 和 Bruynooghe[26] 提出的基于主动学习范式的 CLINT 等.ILP 技术也取得了实际应用[27-31].

由于逻辑程序无法直接描述不确定性,ILP 难以对带噪声的数据进行学习.为了解决此问题,Dantsin[32] 首次在逻辑程序中引入概率分布,提出了概率逻辑程序(probabilistic logic programming, PLP),Poole[33] 研究了 PLP 中的概率推断,Sato[34] 则在 Dantsin[32] 的基础上为 PLP 定义了严格的

FOL 语义. 2004 年 De Raedt[35] 等人提出了以 PLP 为模型的机器学习框架,它是 ILP 的一种概率扩展,被称为概率归纳逻辑程序设计(probabilistic inductive logic programming, PILP). 在这些工作的基础上,研究者们提出了众多 PLP 模型[36-43]与 PILP 算法[44-48].

2010 年至今,ILP 领域的研究又取得进一步发展. 例如在传统的 ILP 理论方面,研究者们提出采用二阶逻辑诱导来进行 FOL 归纳[49-50],此类 ILP 方法在谓词发明和学习递归规则的能力上取得了突破;PILP 领域依旧十分活跃[51-58],被成功应用在自然语言处理和生物信息学等许多兼具不确定性与复杂领域知识的实际任务中[59-64];此外,随着近年来深度学习的崛起,为了进一步提高 ILP 的学习效率,研究者们开始尝试在 ILP 中嵌入可微构件[65-66],并取得了一定成效.

2 基本内容

在不做特别说明的情况下,本文将以汉字、阿拉伯数字和小写英文字母开头的词代表谓词、函数与常量,以大写英文字母开头的词代表变量和 FOL 公式,并省略 FOL 公式中的全称量词"∀".

不含连接词的 FOL 公式被称为原子公式(atom),例如"奇数(X)";原子公式及其否定式被称为逻辑文字(literal),例如"¬ 奇数(X)";当公式中不含变量时被称为"具体的"(grounded),具体的原子公式被称为具体事实(ground facts),例如"奇数(3)".

ILP 模型是由逻辑公式组成的集合:

$$A \leftarrow B_1 \wedge B_2 \wedge \cdots \wedge B_n. \tag{1}$$

其中,A 和 B_i 分别为 FOL 原子和文字. 一般称这种公式为 FOL 规则,其中"←"右边的部分是有限个文字的合取,被称为规则体(body),表示此规则的前提;"←"左边的 A 是一个 FOL 原子,被称为规则头(head),表示规则的结果;"←"是数理逻辑中的蕴涵符,表示"推出"关系;$n \geqslant 0$ 为规则长度. 该规则表示:"若 $B_1, B_2, \cdots, B_n$ 均成立,则 A 也成立".

FOL 系统的语义被体现在 FOL 公式的赋值(assignment)中. 最基本地,可以对每一个具体事实进行赋值,令其为 true 或为 false. 然后 FOL 系统会按照一定规则进行演算,从而得到其他公式的真值. 若存在一种赋值方使得某公式为 true,则称此公式为可满足的(satisfiable). 公式(集)Γ 与 T 之间的可满足性被称为"语义蕴涵"(entailment),记为"$\Gamma \models T$". 它表示一切令 Γ 为真的赋值也使得 T 为真. 公式(集)之间还存在一种"语构蕴涵"(implication)关系,记为"$\Gamma \vdash T$",一般读作"Γ 可推出 T". 它表示 Γ 可以凭借 FOL 公理证明出 T.

对于一般的形式系统,这 2 种蕴涵关系并不等价. 若一个形式系统有$(\Gamma \models T) \rightarrow (\Gamma \vdash T)$,则称该系统为"完备的"(complete)[3]. 对机器学习而言,若令 Γ 和 T 分别表示假设(hypothesis)和训练样例(example),那么对完备的学习算法来说,其假设空间必然包含所有满足训练样例的假设.

2.1 学习任务

ILP 主要关注"概念学习"[1],即学习一个关于目标概念(target concept)的描述(该描述也可视为一个单类别分类器). 学习过程的输入是一组关于目标概念的样例和背景知识(background knowledge),输出为一个满足所有样例的假设,即学得模型. 本文将背景知识记为 B,假设模型记为 H,训练样例记为 $E = E^+ \cup E^-$,其中 E^+ 和 E^- 分别代表正负样例. 那么,概念学习任务可以用 FOL 语言形式化为

$$B \cup H \models E. \tag{2}$$

对于传统的概念学习问题来说,B 和 E 的形式一般为属性-值数据集,E 中的每个示例(instance)均被表示为一个特征向量,H 可以是任何形式. 而对 ILP 来说,这里的 B,E 和 H 均为逻辑程序[5]. 通常情况下,E 是一组关于目标概念的具体事实;B 则是一系列关于原始概念(primitives)的具体事实;H 是一个 FOL 规则集,每条规则均由原始概念组成.

表 1[1]展示了一个 ILP 任务所使用的数据集. 其中的原始概念为瓜与瓜之间在某些属性上的比较关系;每行是一个关于原始概念的具体事实;目标概念是"更好". 此类数据集中的条目不再是关于某一个对象的特征或标记,而是多个对象之间的"关系". 故而 ILP 的任务属于关系学习(relational learning)[1],此类数据集也被称为关系型数据集(relationaldata).

Table 1 An Example of Relational Dataset

表 1 关系型数据集

Object1	Relation	Object2
瓜 1	根蒂更蜷	瓜 4
瓜 1	色泽更青	瓜 4
瓜 2	色泽更黑	瓜 3
瓜 2	敲声更闷	瓜 3
瓜 2	更好	瓜 3

对象间的关系可能十分复杂,例如表1中的目标概念"更好"便是一个递归关系:

更好(X,Y)←更好$(X,Z)\wedge$更好(Z,Y).

"归纳逻辑程序设计中"的"归纳"是逻辑系统中演绎(deduction)推理的逆过程."演绎"指从一般规律出发推演出具体结论的特化(specialization)过程.譬如式(2)自左向右所描述的便是以$B\cup H$作为前提,最终推导出结论E的演绎推理.而"归纳"则与演绎相反,它需要从具体的样例出发,总结出一般规律以概括这些样例,故而它是一个泛化(generalization)过程.换言之,ILP可视为FOL中演绎推理的逆运算.

表2总结了FOL中常见的演绎和归纳运算.大部分ILP方法[11-12,14-16,67]均源自这些归纳运算.此外,还有一些ILP系统试图先将关系型数据转化为属性-值数据,再使用命题学习技术来获取FOL规则[25,68-70].本节将简要介绍这些基本的ILP方法.

Table 2 Deduction and Induction Operations in FOL

表2 FOL中的演绎与归纳运算

Deduction	Induction
Most General Unification	Least General Generalization
Resolution Principle	Inverse Resolution Principle
Implication	Inverse Implication
Entailment	Inverse Entailment

2.2 最小一般泛化

Robinson[71]提出的"合一"(unification)是最简单的FOL演绎运算,其目标是将逻辑变量替换(substitute)为常量以使2个逻辑文字相等,这可以直观地理解为对背景知识B进行"查询"[1].例如当B中存在事实"$A_1=a(1,2)$"时,若欲推出"$A_2=a(X,2)$"的赋值,只需将A_2与A_1合一即可得到一个替换"$\theta=\{X=2\}$"使得$A_2\theta=A_1$,即当$X=2$时A_2为true.合一过程中同时存在多种可行的替换时,往往希望通过替换最少的变量来使合一成立,这被称为"最一般合一"(most general unification, MGU).

合一的逆运算是由Popplestone[72]所提出的"泛化"运算,它反过来将原子公式中的常量替换为变量以使逻辑文字变得更一般.MGU的逆运算是Plotkin[6]提出的最小一般泛化(least general generalization, LGG),它能够通过替换最少的常量使得FOL公式的泛化性最强.例如,若B中同时存在2条具体规则:

1) 更好(瓜1,瓜2)←色泽更黑(瓜1,瓜2).

2) 更好(瓜3,瓜2)←色泽更黑(瓜3,瓜2).

LGG可将它们中的常量代换为变量,以得到一条能够概括这2条规则的一般规则:

更好$(X$,瓜2)←色泽更黑$(X$,瓜2).

在ILP问题中,$B\cup E$一般为关于原始概念和目标概念的具体事实,不存在可供LGG直接泛化的具体FOL规则.因此Plotkin[7]提出先用$B\cup E$来生成关于目标概念的具体规则,再使用LGG对它们进行泛化.例如根据表1中的所有事实即可构造出具体规则:

更好(瓜2,瓜3)←色泽更黑(瓜2,瓜3)∧
根蒂更蜷(瓜3,瓜4)∧
…

这种根据全体$B\cup E$生成的具体FOL规则后来被称为饱和规则(saturation)[14];由于针对饱和规则进行的LGG与$B\cup E$相关,所以这种LGG被称为相对最小一般泛化(relative LGG, RLGG)[7].

Plotkin[7]证明对于任意FOL文字集合S,只要它不包含矛盾文字——即不存在A使得$A\in S$且$(\neg A)\in S$,则可通过LGG得到至少一条可满足S的非空FOL规则R.当B与E分别为由n和m个具体事实构成的集合时,RLGG可以求出所有满足式(2)的H,但最坏情况下学得的规则将包含$O((n+1)^m)$个文字[8].

但是,若B中包含形式不受限制的FOL公式,则(R)LGG运算无法保证完备性[8].此外,若目标概念必须由多条FOL规则才能表示,则(R)LGG将无法学得正确的模型[8].因此,一般情形下基于(R)LGG的ILP不满足完备性.

基于RLGG归纳的代表性ILP算法是Muggleton和Feng[24]提出的Golem,它有较高的学习效率,是早期应用最广泛的ILP系统之一.

2.3 逆归结

在合一的基础上,Robinson[71]提出了归结原理(resolution principle)以处理更复杂的多步演绎.它试图借助MGU运算来找到背景知识与演绎目标中相反的项,然后对它们进行消解.将归结原理的过程逆转过来便构成一种FOL归纳运算.Muggleton[11-12]首次将该方法应用于机器学习,并将它命名为逆归结(inverse resolution).文献[1]中对FOL中的归结原理与逆归结有详细介绍.

图1展示了4种逆归结运算.其中,p,q和r代表规则头中的原子公式;A,B和C代表规则体中的

FOL 文字集合;每种运算的上下 2 行分别为逆归结的输入与输出;二者间的箭头代表逆转后的归结原理[1].

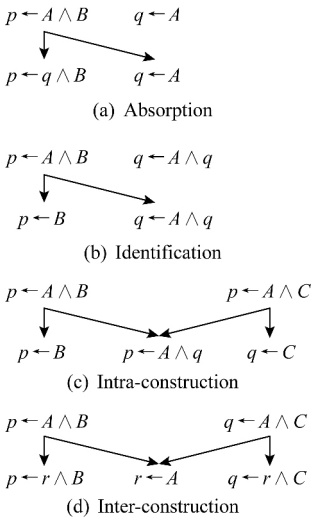

Fig. 1 Four kinds of inverse resolution operators

图 1 4 种逆归结运算

逆归结首次实现了 FOL 归纳中的谓词发明[12],它在学习过程中能够自动构建背景知识 B 中未出现的概念,例如图 1 内构(intra-construction)与互构(inter-construction)运算输出的新原子 q 和 r. 新发明的谓词可作为新的"原始概念"被用于构成 H. 直观来看,谓词发明是当 B 中的原始概念不足以简洁地描述目标概念时,ILP 将某些逻辑文字的组合归纳为一个事先未定义的新概念,并将之用于构建 H,以减少 H 的规则条数或者规则长度. 换言之,谓词发明能够对 B 所定义的领域结构进行补充,它可以帮助用户对领域建立新的认识[2].

对命题规则学习而言,只需吸收与内构 2 种运算即可保证逆归结的完备性[11],但对 ILP 该结论并不成立[13].

基于逆归结的 Duce[11] 和 Cigol[12] 是最早实现的 ILP 系统,它们分别被用于命题规则学习和 ILP 任务中,后者第一次实现了 FOL 中的谓词发明. 逆归结的缺点在于内、互构运算会不断地对规则集进行扩充,而得到的新规则又会作为下一阶段逆归结的输入,从而导致生成更多规则. 此外,这些新谓词、新规则的可满足性在学习过程中很难验证. 因此在缺乏有效的剪枝策略时,逆归结运算会造成组合爆炸. 所以 Cigol 的效率远不如基于 RLGG 的 Golem[24].

2.4 逆语构蕴涵

2.2 节和 2.3 节讲述的 2 种 FOL 归纳运算均无法满足完备性,根本原因在于它们所采用的基础泛化运算 LGG 与 FOL 中的语构蕴涵关系"→"并不等价[15].

"P 是 Q 的 LGG"仅代表公式 P 可以通过变量替换与 Q 合一,不妨将这种关系记为"$P\theta\subseteq Q$",其中 θ 代表合一时使用的变量替换. 而语构蕴涵关系"$P\rightarrow Q$"的本义是"P 可推出 Q". 事实上,若 $P\theta\subseteq Q$ 则必有 $P\rightarrow Q$;反之则并不成立. 例如,考虑 2 个 FOL 公式:

$$P\equiv p(f(X))\leftarrow p(X),$$

$$Q\equiv p(f(f(X)))\leftarrow p(X),$$

使用 $f(x)$ 替换掉 P 中的 X 得到:

$$R\equiv p(f(f(x)))\leftarrow p(f(X)).$$

再对 P,R 使用三段论(modus ponens)[3] 即可证 $P\rightarrow Q$,即 P 蕴涵 Q;但是,P 却无法通过任何变量替换来使之等价于 Q. 也就是说,Q 无法通过 LGG 得到 P. 因此 LGG 不能保证 FOL 归纳的完备性.

可以看到,上例中从 P 到 Q 的推导过程只利用了规则 P 自身,因此 Q 被称为由 P"自归结"(self-resolution)得到. 为了补全 LGG 运算,Muggleton[15] 给出了完备语构蕴涵的 3 种情况. 在 FOL 中,若 $P\rightarrow Q$,则要么 Q 恒真;要么 $P\theta\subseteq Q$;要么 $\exists R(\theta R\subseteq Q)$,其中 R 可通过对 P 不断进行自归结得到.

依据该结论,Muggleton[15] 给出了对 FOL 归纳完备的逆语构蕴涵(inverse implication)运算,它在使用 LGG 对具体规则进行泛化的同时,还需对所有 LGG 输出的中间结果 R 进行逆归结,以找到所有可通过自归结得到 R 的规则 R'. 根据上面的结论,只需令 $H=R\cup R'$ 即可保证 FOL 归纳的完备性.

事实上,在 FOL 中能够进行自归结的公式均为递归公式,即类似于"$p(f(X))\leftarrow p(X)$"这样规则体与规则头中出现相同谓词的 FOL 规则. 在逆语构蕴涵出现之前,基于(R)LGG 与逆归结的 ILP 无法处理此类递归概念.

遗憾的是,找寻递归语句的逆自归结公式是一个不可判定(undecidable)[3] 问题,所以完备的逆语构蕴涵只停留在理论阶段,并未形成真正的算法. 目前为止,基于逆语构蕴涵思想的 ILP 方法仅有 Idestam-Almquist[67] 在对其进行放松后提出的逆 T-语构蕴涵.

2.5 逆语义蕴涵

2.4 节介绍的(R)LGG、逆归结和逆语构蕴涵的目标均是对可证关系(即"→")进行逆转,它们很难保证 FOL 归纳的完备性. 为了解决"→"带来的问题,Muggleton[16-17] 提出应当从 FOL 语义——即 $B\cup H$ 对 E 可满足性的角度来进行归纳.

基于该想法,Muggleton[16]提出了语义蕴涵"⊢"的逆运算——逆语义蕴涵(inverse entailment).通俗地理解,它基于"逆否命题与原命题等价"这一原则,将式(2)等价地转化为

$$B \cup \neg E \models \neg H. \tag{3}$$

这种做法巧妙地将原归纳问题化归为一个演绎问题:即通过 $B \cup \neg E$ 来推导出可能的 $\neg H$,最后只需将推导出的结果取反,即可得到原问题所求解的 H.

具体来说,在应用逆语义蕴含时,先要将所有样例 E 取反得到 $\neg E$,再对 $B \cup \neg E$ 进行演绎得到 $\perp$.由于演绎推理得到的结果只能是一组具体事实,因此不能将 $\perp$ 直接取反作为 H.不过,若 H 存在,则由式(3)可知 $(\perp \models \neg H) \wedge (H \models \neg \perp)$.根据"∧"的右半部分,只需先将 $\perp$ 中所有具体事实取反,再进行泛化即可得到 H.也就是说,逆语义蕴涵的步骤是"先特化、后泛化".

理论上,对形如式(1)的 FOL 规则集合,逆语义蕴涵能保证归纳的完备性[17],但这在实际操作中却并不简单.一个主要原因是在进行式(3)中的演绎推理时难以保证得到的 $\perp$ 形式和数量足够丰富.例如对 2.4 节提到的递归公式,它们通过自归结演绎得到的事实可能有无穷多个.因此一旦 B 中存在递归规则,就无法保证获得足够大的 $\perp$ 并泛化出一切满足式(3)的 H.所以,在实现过程中往往要引入一些先验来对 $\perp$ 的形式和演绎过程进行约束,在保证逆语义蕴涵的效率的同时使之尽可能完备.

最早基于逆语义蕴涵的 ILP 方法是 Muggleton[16]于 1995 年提出的 Progol,它通过设置超参数来约束 $\perp$ 的大小;并使用模式声明(mode declaration)来保证 $\perp$ 中事实的形式足够丰富.而在对 $\neg \perp$ 进行泛化时,Progol 还提出使用精化算子(refinement operator)[18]来对搜索过程进行剪枝来提高学习效率,避免在进行泛化时出现冗余规则.

2.6 命题化

基于命题化方法[68]的 ILP 试图先将关系型数据转化为属性-值数据,然后再使用命题规则学习技术来建立模型.命题化数据中的特征为 FOL 文字或 FOL 文字构成的合取式,因此命题化方法学得的"命题规则"事实上仍旧是 FOL 规则.

此类方法中最著名的是Lavrač等人[25]提出的 LINUS.它首先根据 B 针对目标概念构造 FOL 关系型特征,然后计算每个训练样例在关系型特征上的取值,并以之作为特征向量来构建属性-值数据.例如,对表 1 中的数据,直接将基本概念作为关系型特征即可建立如表 3 所示的数据集.其他命题化 ILP 方法大多基于此框架进行改进,区别一般在于关系型特征的构造方式[68-70].

Table 3 An Example of Propositionalized Dataset

表 3 命题化数据集

(X,Y)	色泽更黑	根蒂更蜷	…	更好
(瓜 2,瓜 3)	true	true	…	true
(瓜 1,瓜 4)	false	true	…	false
⋮	⋮	⋮		⋮

与其他 ILP 方法相比,基于命题规则学习的命题化方法既能更自然地处理领域中的噪声,又拥有更高的学习效率.但它在生成数据集时需要耗费一定时间.此外,它的假设空间十分依赖于关系型特征的构造,若领域中的关系过于复杂,则需要生成大量的关系型特征才能保证学得足够好的 H.此外,基于命题化方法的 ILP 不能学习递归规则,因此无法保证 FOL 归纳的完备性.所以,命题化方法一般被应用于以个体为中心的领域[68],即目标概念只描述个体属性而不涉及个体间关系的领域.

2.7 ILP 的 PAC 可学习性

基于命题规则学习的 PAC 可学习性结论[19,73],Page 和 Frisch[74]证明了单条不含函数符且非递归的一类特殊 FOL 公式通过 LGG 是 PAC 可学习的.此类 FOL 公式形如式(1),但所有出现在 $\{B_1, B_2, \cdots, B_n\}$ 中的 FOL 变量必须也在 A 中出现.

上述结论只适用于单条 FOL 规则学习,为了研究规则集可学习性,Džeroski 等人[20]定义了一类"确定子句"来对 FOL 规则的形式进行约束.这里的"确定"代表对于式(1)规则体中的每个文字 B_i:若 B_i 中存在一个变量 X 从未在其之前的文字 A,$B_1, \cdots, B_{i-1}$ 中出现,那么对 X 来说,一旦 $A, B_1, \cdots, B_{i-1}$ 中除 X 以外的变量取值确定,则 X 的取值也能唯一确定.他们证明了对于符合某些简单概率分布、不含函数符且无递归的 FOL 确定子句集,可在多项式时间内将其扩张为 PAC 可学习的命题规则集.

若不对目标概念所服从的概率分布进行假设,Džeroski 等人[21]证明不含函数符、无递归且每条规则长度有限的一类 FOL 规则集是 PAC 可学习的.但此时学习算法的时间复杂度关于目标概念中最大规则的长度呈指数级增长.Cohen[22]进一步去掉了该结论中对"非递归"的要求,但增加了一些额外假设,例如每个 FOL 变量在规则中的出现次数有限.

虽然陆续有理论证明某些形式受约束的 FOL 规则集 PAC 可学习，但 Kietz[23] 证明，一般形式的 FOL 公式集不是 PAC 可学习的. 这是因为 ILP 在搜索 H 的过程中必须不停地检测它是否与 $B \cup E$ 冲突，而对于形式不加约束的 B 和 E，这是 NP-hard 问题.

这些结果均说明 FOL 规则学习在本质上比命题规则学习更加困难. Cohen 与 Page[75] 总结了 ILP 关于 PAC 可学习性研究的大部分结论. 他们指出，FOL 模型远比命题规则复杂，所以不能仅以命题规则学习的样例数与规则复杂度来衡量 FOL 模型的可学习性，而应当试图从 FOL 模型本身的角度对可学习性进行探索.

3 研究进展

第 2 节所介绍的基本 ILP 方法有 3 个主要局限：1）除了逆归结以外，其他实用性更强的 ILP 方法均无法直接实现谓词发明，并且对递归规则的学习效率较低；2）受 FOL 语言表达能力的限制，大部分 ILP 方法不能在数据带噪声的情况下进行学习；3）由于 ILP 的假设空间结构十分复杂，进行高效的学习有相当的困难.

近年来，ILP 研究者们在这 3 个问题上均取得了进展. 其中以元解释学习（meta-interpretive learning, MIL）[50] 为代表的二阶诱导方法较好地解决了谓词发明和递归规则学习的问题；概率归纳逻辑程序（PILP）[35] 则对 ILP 进行了概率扩展，使之能够应对学习问题中的噪声；还有一些工作尝试在 ILP 中嵌入神经网络等可微构件[65-66]，利用梯度下降和误差反向传播算法来加速 ILP 学习. 本节将对这些方向中的代表性工作进行简要介绍.

3.1 二阶诱导推理

诱导推理（abductive reasoning）[76] 是在演绎和归纳之外的另一种逻辑推理形式. 它能根据背景知识为已观测事实找到一种合理的原因作为解释，因此也被称为“溯因”推理. 例如，若背景知识中存在如下 3 条 FOL 规则：

1）出汗(X) ←觉得热(X)，

2）觉得热(X)←天气热，

3）觉得热(X)←运动后(X).

那么，在观测到“出汗（张三）”为 true 时，根据第 1 条规则便能诱导出唯一的原因：“觉得热（张三）”. 接下来，根据后 2 条规则我们可以进一步推测出：要么现在天气热、要么张三刚刚运动完、要么此二者均成立. 可以看到，诱导推理是一种已知一般规则（因果关系），从具体事实（关于某个对象的观测事实）中推演出其他具体事实（发生在该对象身上的特殊原因）的过程.

遗憾的是，ILP 的目标并不是获得关于某个概念的具体事实，而是学习由一般规则组成的 FOL 规则集 H. 因此，上述 FOL 诱导推理不能被直接用于 ILP. 为此，Inoue 等人[49] 提出了元诱导（meta-level abduction）方法，试图使用二阶逻辑诱导来进行 FOL 归纳.

早在 2003 年，Lloyd[77] 就曾指出高阶逻辑（higher-order logic）语言[3] 或许能更有效地形式化并解决 FOL 归纳问题. 二阶逻辑（second-order logic, SOL）即是一种高阶逻辑，它可以对 FOL 中的谓词和函数进行量化，因此对于 SOL 公式来说，FOL 公式只是它的具体个例. 比如对于 SOL 规则：

$$P(X,Y) \leftarrow Q(X,Z) \wedge R(Z,Y).$$

其中，P，Q 和 R 均为 SOL 变量，可被替换为任意二元 FOL 谓词. 若将它们全部替换为“更好”，便可得到关于“更好”关系的递归 FOL 规则. 元诱导学习试图将 SOL 规则作为背景知识 B、将训练样例 E 作为观测到的具体事实来进行诱导推理，最终得到的便是具体的 SOL 规则集合，即 FOL 规则集 H.

不过，元诱导学习并未实现真正意义上的 SOL 诱导. 它需要将描述原始概念的 FOL 谓词转化为逻辑常量，并将背景知识中的 SOL 规则转化为 FOL 规则，以将 SOL 诱导表示为 FOL 中的诱导问题进行求解. 最终还需将诱导得出的具体事实进行还原，才能获得待求解的 FOL 规则集 H.

Muggleton 等人[50] 提出的元解释学习（meta-interpretive learning, MIL）在逻辑程序中实现了一个可以解释 SOL 规则的元解释器（meta-interpreter），真正实现了 SOL 诱导推理. MIL 允许用户在 B 中定义一组被称为“元规则”（metarule）的 SOL 公式 B_{MR}，并通过元解释器来对这些元规则进行 SOL 中的诱导推理.

MIL 的目标可以这样描述：给定背景知识 B、训练样例 E 和 SOL 元规则 B_{MR}，MIL 需要找到一种 SOL 变量替换 θ 将元规则的 SOL 变量替换为关于原始概念的 FOL 谓词，以得到 FOL 规则（集）$B_{MR}\theta$，使得 $B_{MR}\theta \cup B \models E$. 换言之，MIL 通过令 $H = B_{MR}\theta$ 将式（1）中对 H 的搜索转化为对 θ 的搜索.

这种搜索是通过 SOL 中的诱导推理完成的，而诱导推理中的已观测事实是“E 可证”. 因此 MIL

需要在对 E 的证明过程中诱导出 θ 的取值. 以表 2 的任务为例,待证明的样例为"E=更好(瓜 2,瓜 3)",设此时 B_{MR} 只包含一条元规则 $R\equiv P(X,Y)\leftarrow Q(X,Y)$. 证明过程开始时,MIL 会将 R 的规则头与 E 进行合一,以表示 E 是可证明的结论,为此需采用变量替换:

$$\theta_1=\{P=\text{更好},X=\text{瓜 }2,Y=\text{瓜 }3\},$$

将其应用于 B_{MR} 中的元规则 R,得到:

$$R\theta_1=(\text{更好}(\text{瓜 }2,\text{瓜 }3)\leftarrow Q(\text{瓜 }2,\text{瓜 }3)).$$

为了令证明成立,必须保证该规则体中的"Q(瓜 2,瓜 3)"赋值为 true. 通过对表 2 中的事实进行查询,可看出只需使用变量替换:

$$\theta_2=\{Q=\text{色泽更黑}\}$$

即可令 $B\cup R\theta_1\theta_2\models E$ 成立. 保留 θ_1 中的 SOL 变量替换可得 $\theta_1'=\{P=\text{更好}\}$,最终令 $\theta=\theta_1'\theta_2$ 即可得到一个候选假设模型

$$R\theta=(\text{更好}(X,Y)\leftarrow\text{色泽更黑}(X,Y)).$$

显然,MIL 可看作一种以 SOL 元规则为模板进行的 FOL 规则搜索.

由于 SOL 变量能被替换为任意 FOL 谓词,MIL 的规则搜索比基于 FOL 归纳运算的 ILP 更加灵活:当元规则头和规则体中的 SOL 变量被替换为相同谓词时,便形成了递归规则;当元规则头被替换为 B 中不存在的谓词时,便实现了谓词发明.

例如文献[50]以楼梯的 3D 点云作为训练样例,学会了关于楼梯概念的模型假设:

$$staircase(X)\leftarrow s_1(X).$$
$$staircase([X,Y,Z|Planes])\leftarrow s_1([X,Y,Z])\wedge staircase([Z|Planes]).$$
$$s_1([X,Y,Z])\leftarrow vertical(X,Z)\wedge horizontal(Z,Y).$$

其中,(X,Z),(Z,Y) 和 $Planes$ 均为点云中的平面;$\{X,Y,Z\}$ 为平面的边界;谓词 $vertical$ 和 $horizontal$ 分别表示平面的方向为垂直和水平,谓词 $staircase(X)$ 表示平面序列 X 是一段楼梯. 该模型中第 2 条规则即为递归规则,而 s_1 则是被发明的谓词. 经过解读可知 s_1 代表"一级台阶",而第 2 条规则表示"楼梯加上一级台阶依然是楼梯".

显然,MIL 的假设空间结构依赖于元规则的选取. 若仅使用"$P(X,Y)\leftarrow Q(X,Y)$"作为元规则,那么 MIL 将永远无法学会形式为 $P(X,Y)\leftarrow Q(X,Z)\wedge R(Z,Y)$ 的 FOL 规则. 此外,由于 MIL 的诱导过程中可以使用 B_{MR} 中的任意一条元规则作为模板,元规则的总数会影响 MIL 的搜索效率. Cropper 与 Muggleton[78-79] 研究了元规则选择对 MIL 假设空间完备性的影响,证明 B_{MR} 只需包含 2 条元规则,其假设空间即可包含所有通用图灵机可计算函数. 但对于一般的目标概念 H^*,MIL 通过这 2 条元规则学得的模型 H' 只能保证在语义上与 H^* 等价. 而在形式上,H' 所包含的规则条数可能比 H^* 多很多,这往往要求 MIL 在对样例的证明过程中增加搜索深度. 因此仅仅减少元规则条数不一定能够提高 MIL 的学习效率.

3.2 概率扩展

3.2.1 概率逻辑程序

经典 FOL 只能用于形式化"非真即假"的推理过程. 为了在 FOL 中引入概率分布,需要对原始的 FOL 语义进行扩展——即允许逻辑事实依概率为真. 那么式(2)的学习问题可以重写为

$$\max_H P(E|B\cup H). \tag{4}$$

式(4)不再要求 H 与训练样例 E 绝对一致,仅要求 H 尽可能在对 E 赋值的判断上少犯错误.

为了适应模型语义的变化,还需对式(1)中 FOL 规则的语法进行扩展. 一般可以采用如下形式的概率逻辑规则(probabilistic logic rules):

$$p::A\leftarrow B_1\wedge B_2\wedge\cdots\wedge B_n. \tag{5}$$

其中,$p\in[0,1]$ 为概率值. 当 $n=0$ 时,式(5)退化为一个概率原子公式(probabilistic atom)$p::A$,当 A 不含变量时它被称为概率事实(probabilistic facts),它表示 A 是一个随机变量,且 A 服从参数为 p 的二项分布.

由概率 FOL 规则构成的逻辑程序被称为概率逻辑程序[32],为了使用概率逻辑程序进行推理,必须严格定义其上的概率分布形式.

FOL 系统的语义由领域内所有具体事实的赋值决定,因此最直接的方法便是将概率分布定义在这些具体事实上,这种定义方式被称为"分布语义"(distribution semantics)[34]. 它将领域中所有具体事实作为随机变量,并在其上定义联合概率分布. 考虑图 2 中的 PLP 模型:

$$0.8::\text{根蒂更蜷}(\text{瓜 }1,\text{瓜 }2).$$
$$0.3::\text{敲声更闷}(\text{瓜 }1,\text{瓜 }2).$$
$$0.6::\text{色泽更黑}(\text{瓜 }1,\text{瓜 }2).$$
$$\text{false}\leftarrow\text{根蒂更蜷}(X,Y)\wedge\text{根蒂更蜷}(Y,X).$$
$$\text{false}\leftarrow\text{敲声更闷}(X,Y)\wedge\text{敲声更闷}(Y,X).$$
$$\text{false}\leftarrow\text{色泽更黑}(X,Y)\wedge\text{色泽更黑}(Y,X).$$
$$\text{更好}(X,Y)\leftarrow\text{根蒂更蜷}(X,Y)\wedge\text{敲声更闷}(X,Y).$$
$$\text{更好}(X,Y)\leftarrow\text{敲声更闷}(X,Y)\wedge\text{色泽更黑}(X,Y).$$

Fig. 2 An example of probabilistic logic program

图 2 概率逻辑程序示例

其中行1～3定义了3个带权的具体事实,它们是该领域中分布的来源;其他几行为一般的FOL规则.

若将PLP中所有FOL规则构成集合记为R,所有概率事实的集合记为F.假设F中的概率事实相互独立,$W \in F$是所有赋值为true的概率事实,那么W出现的概率为

$$P_F(W) = \prod_{f_i \in W} p_i \prod_{f_i \in F \setminus W} (1 - p_i). \tag{6}$$

例如,若W仅包含图2中的前2个概率事实,那么它出现的概率为$P_F(W)=0.8\times0.3\times(1-0.6)=0.096$.$W$加上原有的FOL规则集$R$即可得到一个普通的一阶逻辑程序:

根蒂更蜷(瓜1,瓜2).

敲声更闷(瓜1,瓜2).

false←根蒂更蜷(X,Y)∧根蒂更蜷(Y,X).

false←敲声更闷(X,Y)∧敲声更闷(Y,X).

false←色泽更黑(X,Y)∧色泽更黑(Y,X).

更好(X,Y)←根蒂更蜷(X,Y)∧敲声更闷(X,Y).

更好(X,Y)←敲声更闷(X,Y)∧色泽更黑(X,Y).

根据W不同取值而产生的所有一阶逻辑程序被称为"可能世界"(possible worlds)[80].在每个可能世界的内部都可进行正常的FOL推理.如此便可以计算PLP中任意原子公式A为真的概率:

$$P(A) = \sum_{W \subseteq F, W \cup R \models A} P_F(A). \tag{7}$$

式(7)表示原子公式A为真的概率等于所有蕴涵A的可能世界的概率之和.以图2为例,除去互斥事实后该领域的可能世界共有8个,其概率分布如表4所示.通过计算可知,"更好(瓜1,瓜2)"为真的概率是0.144+0.096+0.036+0.024=0.3.

Table 4 Possible Worlds of the PLP in Fig. 2

表4 PLP(图2)的可能世界与概率分布

Possible Worlds	更好(瓜1,瓜2)	Probability
蜷(1,2)∧闷(1,2)∧黑(1,2)	true	0.8×0.3×0.6=0.144
蜷(1,2)∧闷(1,2)∧¬黑(1,2)	true	0.8×0.3×(1−0.6)=0.096
蜷(1,2)∧¬闷(1,2)∧黑(1,2)	false	0.8×(1−0.3)×0.6=0.336
蜷(1,2)∧¬闷(1,2)∧¬黑(1,2)	false	0.8×(1−0.3)×(1−0.6)=0.224
¬蜷(1,2)∧闷(1,2)∧黑(1,2)	true	(1−0.8)×0.3×0.6=0.036
¬蜷(1,2)∧闷(1,2)∧¬黑(1,2)	true	(1−0.8)×0.3×(1−0.6)=0.024
¬蜷(1,2)∧¬闷(1,2)∧黑(1,2)	false	(1−0.8)×(1−0.3)×0.6=0.084
¬蜷(1,2)∧¬闷(1,2)∧¬黑(1,2)	false	(1−0.8)×(1−0.3)×(1−0.6)=0.056

早期的PLP语法不允许FOL规则像式(5)那样拥有自己的权重,实际上,使用概率事实(布尔随机变量)加FOL规则的PLP的表达能力已经足够强大,能够直观地定义基于布尔随机变量的贝叶斯网、马尔可夫链和隐马尔可夫模型等常见概率模型.

Dantsin[32]在提出PLP时便确定了"概率事实加FOL公式"的基础语法规则,但由于其语法过于灵活,在PLP上进行精确概率推断比较困难.为此,Poole[33]对PLP语法进行了简化,并证明贝叶斯网是这种PLP的一个特例.Muggleton[37]提出的随机逻辑程序(stochastic logic programs)在PLP中引入了递归规则,能够通过采样来进行概率推断.由于不断递归产生的具体事实概率值呈指数级减小,因此SLP的概率推断可以保证有界.随后Poole[38]和Vennekens[40]在PLP中引入了带权规则,进一步扩展了PLP的语法,使其有能力表达多值概率分布.

De Raedt等人提出的ProbLog[41]是现在应用最广泛的PLP系统.它集成了许多成熟的PLP语法,并进一步优化了PLP概率推断的效率.它通过将PLP编译为二元决策图(binary decision diagrams, BDD)[81]来将PLP的概率推断转化为一种可满足性(SAT)问题,然后使用带权模型计数(weighted model counting, WMC)来实现高效的PLP推理.最新的版本ProbLog2[82]将基于BDD的WMC改进为基于d-DNNF(deterministic decomposable negation normal form)[83]的WMC,进一步提高了推理效率.

近年来,针对不同的应用领域,研究者们依旧不断地提出新的PLP系统.例如与贝叶斯网相结合的CLP(BN)[39]、与ASP(answer set programming)相结合的P-Log[42]、与因果推断相结合的CP-logic[43]、为信息检索设计的PROPPR[55]等.它们大多仍使用分布语义对PLP进行建模,仅在做概率推断时有一定区别.有兴趣的读者可参阅相关文献[46,84].

3.2.2 概率归纳逻辑程序

概率归纳逻辑程序(PILP)[46]是以PLP为模型的机器学习方法,其任务为找到满足式(4)的假设模型H.为便于讨论,令其中的$H=(\Gamma;\Theta)$,这里的Γ表示假设模型中的FOL规则结构,Θ代表所有概率规则和事实的参数.在无歧义的情况下,可以省略式中的背景知识B,将式(4)重写为

$$\max_{\Gamma,\Theta} P(E \mid \Gamma,\Theta). \tag{8}$$

虽然PILP使用的数据形式上还是逻辑事实构成的集合,但由于引入了概率分布,其本质是关于随机变量的采样.因此PILP中的样例一般被写为$D=\{E_1=e_1,E_2=e_2,\cdots,E_M=e_M\}$,其中的$E_m$表示第$m$次采样的随机变量(具体事实)集合,$e_m$表示本次采样的结果,即$E_m$中所有具体事实的赋值.为了能够用$D$来对模型进行估计,一般假设这些采样结果服从独立同分布.于是,式(8)中的后验概率可以写成:

$$P(D \mid \Gamma,\Theta) = \prod_{m=1}^{M} P(E_m = e_m \mid \Gamma,\Theta). \tag{9}$$

PILP学习的目标即为最大化似然函数$L(\Gamma,\Theta|D)=P(D|\Gamma,\Theta)$.通常采用对数似然函数,那么PILP最终被形式化为

$$\max_{\Gamma,\Theta} \sum_{m=1}^{M} \ln P(E_m = e_m \mid \Gamma,\Theta). \tag{10}$$

若Γ已知,则PILP退化为参数估计问题:

$$\max_{\Theta} \sum_{m=1}^{M} \ln P_{\Gamma}(E_m = e_m \mid \Theta), \tag{11}$$

其中,P_Γ是结构为Γ的概率逻辑模型的分布函数,对基于分布语义的PLP,P_Γ就是式(7)中的P.

当E_m为模型中所有随机变量——即领域中全部具体事实时,PILP的参数学习问题可以直接使用极大似然估计[1]解决.例如Cussens[45]提出的FAM(failure-adjusted maximization)和ProbLog[41]使用的LFI-ProbLog[51]等.

然而在大部分应用中,训练数据仅包含模型中一部分具体事实的赋值.这种情况相当于参数估计时存在隐变量.记模型第i组样例中的隐变量为X_i,则式(11)的目标可以重写为

$$\max_{\Theta} \sum_{m=1}^{M} \ln \sum_{x_m \in \chi_m} P_{\Gamma}(E_m = e_m, X_m = x_m \mid \Theta), \tag{12}$$

其中χ_m为X_m所有可能赋值的集合,式(12)表示在隐变量X_m的期望下对Θ进行最大似然估计.此类问题可以使用EM算法进行求解[1].Sato[36]提出的PRISM是最早提出使用EM进行参数估计的PLP系统之一,而ProbLog所采用的LFI-ProbLog[51]同样可以通过EM进行参数估计.此外,有些PILP算法还可以利用FOL模型的特性对EM进行加速[53].

当Γ未知时,PILP需要同时学习规则结构和模型参数,大部分算法选择交替地对它们进行学习.

Muggleton[44]提出的SLP学习是最早实现结构学习的PILP方法之一.它首先收集训练数据D中所有出现过的具体事实$E=\bigcup_m \{E_m\}$,然后利用Progol[16]学习一个满足式(2)的FOL规则集H作为Γ,再使用极大似然估计来学习Θ并删去无用的规则.为了保证尽量不漏掉最优的假设模型,SLP第1步的ILP会输出大量FOL规则,但往往它们中的很大一部分并非必需.此外,一旦ILP输出的模型中包含递归规则,SLP基于采样实现的参数估计效率会变得很低.

De Raedt等人[46]提出的PLP模型压缩方法试图通过贪心算法来逐条删除PLP中的规则,并确保每次删除规则后得到的PLP关于训练数据集的似然值最大.该方法虽能对PLP模型结构进行修改,但难以从数据中学习新规则.

Bellodi和Riguzzi[54]提出的SLIPCASE通过ILP的规则精化[18]来生成候选Γ,并采用EMBLEM[53]算法来学习模型参数.Bellodi和Riguzzi[58]提出的SLIPCOVER算法则通过演化计算来改进SLIPCASE中基于规则精化的结构搜索,可有效地缩小搜索空间.

戴望州和周志华[56]提出的SUL(statistical unfolded learning)试图将ILP中的命题化[68]方法扩展至PILP.它首先将训练数据表示为一个超图,然后通过关系型路径查找[85]来构建关系型特征并对数据集进行命题化.接下来,SUL采用统计模型在命题化后的数据集上进行学习,最终将模型转化为概率逻辑规则集.与其他PILP方法相比,SUL能够以较高的效率同时学习规则结构与参数.此外,SUL还实现了PILP中的谓词发明.

3.3 可微构件嵌入

在决策树等模型中嵌入神经网络这样的可微构件早有研究[86].在ILP中嵌入可微构件的策略目前有2种:1)使用线性运算和神经网络来替代逻辑运算,并对FOL系统中的真值进行连续放松;2)将规则结构搜索中的组合优化转换为参数空间中的连续优化.与ILP和PILP相比,可微构件的嵌入进一步提升了学习效率,并允许模型拥有更大的数据吞吐量.

Rocktäschel 与 Riedel[65] 提出的 neural theorem prover(NTP)方法采用了第 1 种策略. 它保留了逻辑程序中的后向链(backward chaining)推理,并使用神经网络来代替推理中用到合一等运算. 后向链是逻辑程序中一种对原子公式进行证明的方式,MIL[50] 的元解释器便借鉴了其思想. 对于一条 FOL 规则 $a(X,Y)\leftarrow b(X,Y)\wedge c(X,Y)$,当用户查询目标概念具体事实 $a(1,2)$ 的赋值时,它会首先被用来与规则头 $a(1,2)$ 进行合一,得到变量替换 $\theta=\{X=1,Y=2\}$;再将规则体中的原子经过 θ 替换可得 $b(1,2)\wedge c(1,2)$. 后向链递归地证明这 2 个原子,若它们均为 true,那么对该规则的证明成功,得到 $a(1,2)$ 的赋值为 true.

NTP 将证明过程中使用的合一运算转变为神经网络 $\text{unify}_\sigma(A,B,S)=S'$,其中 A 与 B 为待合一的 2 个原子,每个原子 $p(e_1,e_2,\cdots,e_m)$ 被表示为一个 $k(m+1)$ 维向量 $(\sigma(p),\sigma(e_1),\cdots,\sigma(e_m))$,其中 $\sigma(X)$ 表示符号 X 的嵌入形式,k 是其长度;S 与 S' 分别为合一之前与之后的证明状态,它包含一个集合 S_ϕ 和一个标量 $S_\psi\in[0,1]$. 前者用于存储目前为止所有使用过的变量替换 θ,后者代表目前状态下合一成功的可能性. σ 为 NTP 网络的所有参数,它包含所有逻辑符号(谓词及常量)的嵌入以及所有 $\text{unify}_\sigma(A,B,S)$ 网络的参数. A 与 B 的合一可能涉及到多种情况,NTP 分别对它们进行了定义. 例如若 $A=e_1$ 和 $B=e_2$ 均为逻辑常量时,则 unify_σ 会直接输出其嵌入的相似度;若 A 是变量时,则 B 被直接赋值为 $\sigma(A)$.

此外,为了应对后向链证明过程中遇到逻辑连接词,NTP 网络还分别定义了 AND 分支与 OR 分支,来将证明中所有用到的 unify_σ 网络按照后向链的分支结构进行连接. 一旦给定了背景知识 B、待证明的原子公式 h 与后向链的最大递归深度 d,NTP 即可编译出一个结构固定的神经网络 $\text{ntp}_\sigma^B(h,d)$. NTP 通过优化该网络在训练样例 $E=\{\langle e,y\rangle\mid y\in\{0,1\}\}$ 上的负对数似然来对参数 σ 进行学习.

NTP 的 FOL 规则结构学习可通过使用 SOL 元规则实现. 例如,可以将一条谓词符号未知的 SOL 规则 $\gamma\equiv P(X,Y)\leftarrow Q(X,Y)$ 加入背景知识用来编译 NTP 网络 $\text{ntp}_\sigma^{B\cup\gamma}(h,d)$. 通过 NTP 的参数学习能得到未知谓词 P 和 Q 的嵌入向量 $\sigma(P)$ 与 $\sigma(Q)$. 然后,只需在已知谓词中找到与 $\sigma(P)$ 与 $\sigma(Q)$ 最相似的嵌入即可将 γ 还原成 FOL 规则. 若未知谓词的嵌入与所有已知谓词均差别很大,则可将其作为新发明的谓词保留下来.

NTP 与早年 KBANN[87] 的思想相似,使用逻辑规则描述的领域知识来初始化神经网络并进行学习. NTP 虽然用神经网络替代了一部分 FOL 演算,但在结构上保留了逻辑程序的推理过程,因此它比一般的神经网络拥有更好的可理解性. 其局限在于初始化 ntp_σ^B 网络结构时必须将后向链中所有的证明分支全部展开以构建网络结构,当背景知识中的 FOL 规则较多、较复杂时,NTP 的网络结构会变得十分庞大,而这其中许多分支可能是无效的. 尤其是在进行 FOL 规则结构学习时,SOL 规则会在后向链展开过程中不断进行递归,直至到达最大深度 d. 因此 NTP 的效率并不高. 此外,它的结构一旦固定就无法再对领域中添加新的规则或常量.

Evans 和 Grefenstette[66] 提出的∂ILP 则主要采用了第 2 种策略. 它首先将所有可能的候选假设模型提前构建出来;然后为每个候选模型设置一个权重;最终根据训练样例对权重进行优化,筛选出权重最高的模型作为输出.

为了限制候选模型的数量,∂ILP 规定对每个目标概念 p 只能学习一个由 2 条 FOL 规则组成的假设模型,其中每条 FOL 规则都依照一条 SOL 元规则[50] 为模板而生成. 它将式(2)中的 ILP 学习目标改写为优化问题:

$$\max_{\Theta_p} P(B\cup cl(\Pi)\cup\Theta_p\models E),\tag{13}$$

其中,$P(X)$ 表示 X 成立的概率,B 为背景知识,$\Pi=\{\tau_p^1,\tau_p^2\}$ 是由 2 条 SOL 元规则 τ_p^i 组成的模板规则集,$cl(\Pi)$ 为所有依照模板 Π 生成的候选模型,Θ_p 是一个大小为 $|cl(\tau_p^1)|\times|cl(\tau_p^2)|$ 的矩阵,$\Theta_p^{j,k}$ 越接近于 1 表示目标概念 p 越可能由 $\{\tau_p^{1,j},\tau_p^{2,k}\}$ 两条规则组成,这里的 $\tau_p^{i,j}$ 表示依照第 i 条元规则生成的第 j 条 FOL 候选规则.

为了求解该目标,∂ILP 也采用策略 1 将 FOL 中的赋值放松为 $[0,1]$ 区间内的连续值,并定义关于 Θ_p 的负对数似然函数:

$$L(\Theta_p)=-\sum_{e,y\in E}[y\ln P(e\mid\Theta_p,\Pi,B,T)+(1-y)\ln(1-P(e\mid\Theta_p,\Pi,B,T))],$$

其中 e 和 y 分别为训练样例及其标记;$P(e\mid\Theta_p,\Pi,B,T)$ 表示当 (Θ_p,Π,B) 确定时,经过 T 步演绎后 e 赋值为 true 的可能性.

为了优化该目标函数,∂ILP 需要将 $p(e\mid\Theta_p,\Pi,B,T)$ 中的 FOL 演算转化为连续可微的函数运算. 它将候选假设模型中的每条 FOL 公式 R 转化为一个可微函数 $F_R:[0,1]^n\mapsto[0,1]^n$,这里的 n 为领域

中所有具体事实的个数,因此 F_R 的定义域和值域均为领域中全体具体事实的赋值.

∂ILP 关于目标概念 p 的候选假设模型只包含 2 条 FOL 规则 $\{\tau_p^{1,j},\tau_p^{2,k}\}$,故可定义该模型对领域内全体事实赋值 $\boldsymbol{a}$ 作用一次后的结果为 $G^{j,k}(\boldsymbol{a})=\boldsymbol{x}$,其中 $\boldsymbol{x}$ 每一维的取值:

$$\boldsymbol{x}[\boldsymbol{a}]=\max(F_j(a[i]),F_k(a[i])).$$

开始进行 FOL 推理时,∂ILP 将背景知识 B 中所有事实的赋值初始状态作为 $\boldsymbol{a}_0$,即对所有出现在 B 中的具体事实 c 有 $a_0[c]=1$,否则 $a_0[c]=0$. 然后,∂ILP 将所有可能的候选模型根据权重矩阵 $\Theta_p^{j,k}$ 一起作用于时刻 t 的领域状态 $\boldsymbol{a}_t$,以得到该时刻的输出 $\boldsymbol{b}_t$:

$$\boldsymbol{b}_t=\sum_{j,k}G^{j,k}(\boldsymbol{a}_t)\frac{\exp(\Theta_p^{j,k})}{\sum_{j',k'}\exp(\Theta_p^{j',k'})},$$

其中,等号右侧的分式代表模型 $\{\tau_p^{1,j},\tau_p^{2,k}\}$ 在全体候选模型中所占比重. 理论上,下一时刻的赋值 $\boldsymbol{a}_{t+1}$ 应当为 $\boldsymbol{a}_t\cup\boldsymbol{b}_t$,但为了保持整个过程可微,∂ILP 将其放松为 $\boldsymbol{a}_{t+1}=\boldsymbol{a}_t+\boldsymbol{b}_t-\boldsymbol{a}_t\boldsymbol{b}_t$.

∂ILP 虽然是一种可微分模型,但它几乎完全保留了 FOL 规则的形式. 与 PLP 的分布语义类似,它的 FOL 语义直接建立在领域中全体具体逻辑事实的赋值上. 在许多不带噪声的 ILP 任务中,∂ILP 的学习效果甚至可以与 MIL[50] 相媲美;而当数据中存在噪声时,它依然能够学到有效的 FOL 规则. 其局限在于假设模型的形式不够灵活. 当目标概念难以仅用 2 条规则描述时,用户必须将其分解为 2 个甚至更多待学习谓词,并为它们一一设定元规则作为模板. 此外,∂ILP 在进行参数学习之前必须枚举所有的候选模型,并将其全部用于 FOL 推理. 事实上,绝大部分候选模型可能是无效、甚至矛盾的,因此∂ILP可能将大量计算资源浪费在无效模型的推断中.

一方面,嵌入可微构件的 ILP 能够有效地利用 GPU 等并行计算设备,在一些仅需简单逻辑推理的关系型学习任务中实现高效学习;但另一方面,它们加入了大量约束来对 ILP 模型的形式进行简化,无法保证待学习的目标概念仍然存在于假设空间中.

4 应　　用

ILP 在早期一般被应用于生物、化学和制药等领域. Golem 曾被应用到蛋白质结构预测[27] 任务中,并取得了比神经网络更好的效果;Progol 被应用于生物医药研究上,并辅助科学家们发现了重要的生物诱变剂[29];基于诱导推理的 ILP 则被用于学习生物的新陈代谢规律[88].

除了经典 ILP 系统之外,PILP 也在许多任务中得到应用. 例如,自然语言是一种结构性强但形式很灵活的数据. PILP 既可以有效地引入语法等背景知识,又能从概率上对其不确定性进行建模,因此已被应用于自然语言处理[60].

ILP 也影响了其他机器学习领域的研究. 例如在统计关系学习(statistical relational learning, SRL)[89] 中,ILP 常被用于对 SRL 模型结构进行初始化[90]. 而在概率图模型中,ProbLog 可以作为一种效率非常高的概率推断引擎来对图模型上的推理和学习进行加速[41].

得益于 FOL 语言强大的表达能力,ILP 能用于辅助科学发现. 例如生态学中的一个重要问题是从生物数量的变化判断物种之间的捕食关系. 因为收集到的数据量十分庞大,若由科学家来对它们一一进行检验则需要消耗很长时间. 并且物种间的捕食关系可能十分复杂,一个物种的数量往往受十几种其他物种的影响. 生态学家们拥有大量复杂的领域知识,能通过 FOL 语言进行表达. Bohan 等人[91] 将 MIL 应用于该问题,不仅实现了数据驱动的食物网自动构建,还通过 MIL 自动发明的谓词,辅助科学家们发现了许多新的现象.

由于 ILP 学得的模型具有良好的解释性,用户可以更好地加以利用. 例如 Progol 曾被应用于一个叫“机器人科学家”(robot scientist)的项目中. 每当 ILP 从实验数据中学得一个模型,生物学家们便对它发现的 FOL 规则进行深入分析,然后根据这些分析结果来设计新的实验. 这种模式将人类的科学发现与机器学习组成一个闭环:人类知识被用于设计科学实验;实验结果最终成为机器学习的数据来源;从数据中学得的模型最终被用于更新人类的知识并设计下一步实验. 该计划运用这种模式成功发现了带有重要功能的基因片段[30-31].

上述应用中所使用的均为关系型数据. 而 Muggleton 等人[92-93] 提出的 Logical Vision 则试图将 ILP 直接应用于像素表示的图像数据以进行视觉概念学习. 该方法用其他学习模型来探测图像中分隔主物件与背景的“边界点”,同时结合 FOL 表达的几何概念作为背景知识,通过 ILP 学习更复杂的视觉概念. Logical Vision 采用了认知科学中的“假设-检验”模型,试图在符号表示的抽象概念学习与

原始数据感知之间架起桥梁. 借助领域知识,Logical Vision 甚至可以学会预测未出现在图像中的物体[92]. 例如,基于简单的光学常识,该方法仅从一幅图片中即可学会判断图像外光源的位置. 更有趣的是,得益于 FOL 强大的表达能力,Logical Vision 在光源学习任务中得到的 FOL 规则还能够从图像中解读出关于反射面凹凸形状的歧义,并通过对不同光源位置进行假设来解释这种歧义. 这在目前其他形式的机器学习中很难实现.

5 讨论与展望

与其他类型的机器学习技术相比,ILP 有其自身的优势.

1) 使用 FOL 作为表达语言的 ILP 拥有强大的表达能力和推理能力,便于对复杂结构的领域知识进行处理和利用.

2) FOL 规则有良好的可理解性. 每条 FOL 规则均遵循"若 X 成立,则 Y 成立"这样的简单模式,且其中的谓词、函数符号大多取自于背景知识中的原始概念,所以学得的模型有良好的语义.

3) FOL 规则有着良好的可重用性. 由于背景知识和假设模型均为 FOL 规则集,ILP 学得的模型可以作为背景知识被直接应用在其他 ILP 任务中. 例如,戴望州等人[93]提出的 Logical Vision 便将学习"三角形"概念所得到的 FOL 规则集作为背景知识重用于"直角三角形"的概念学习任务中. 依照这种模式,ILP 可以实现从简单到复杂任务的分阶段学习[94],并不断将学得的模型进行重用,这为实现"学件"[95]等构想提供了便利.

但另一方面,ILP 技术本身还存在很多局限,很多问题有待解决.

1) 由于 FOL 规则的形式和结构是离散的,其假设空间十分复杂. 为了提高学习效率,PILP 与可微构件的嵌入在一定程度上对 FOL 模型的语义和语法做出了放松. 但是,对 FOL 语义的放松导致难以区别错误规则和领域噪声造成的规则冲突,甚至无法保证放松后的假设空间仍包含目标概念. 此外,机器学习任务不仅面临着日益增长的数据量,还面临着越来越庞大的领域知识. 例如,WordNet[96]作为自然语言处理中最常用的领域知识库之一,它拥有的词汇量与二元具体事实的数量已分别超过 150 000 条与 200 000 个. 现有的 ILP 方法很难对如此庞大的背景知识加以利用.

2) 目前的 ILP 不能进行非单调学习. 只能不断地学习与背景知识不发生冲突的新规则. 这使得它们十分依赖于背景知识的正确性,然而现实机器学习任务中往往很难确保背景知识的正确性.

3) 几乎所有的 ILP 方法都只能从逻辑符号表示的数据中进行学习[97]. 但现实机器学习任务中往往涉及到很多非符号类型数据,例如像素构成的图像、波形组成的音频等. 此外,ILP 任务中的原始概念及具体事实通常需由领域专家从原始数据中标注而来,这往往需要消耗大量成本.

对上述问题目前已经有一些探索. 例如,在工程上可以尝试采用分布式计算来提高 ILP 的搜索效率[98]. 而在理论上,限制 ILP 效率的瓶颈主要在于 FOL 对模型完备性和一致性的要求. 对这 2 个要求该如何放松,放松至何种程度,放松的过程中如何保证学习的效率,均是值得解决的问题.

对非单调学习问题,一种可行的方式是为 ILP 引入非单调推理模型,例如信念修正中的 AGM 模型[99]等. 对于 PILP 来说,该问题则更加复杂,由于它允许数据集中出现噪声,学习系统还必须能够区分由数据噪声和错误规则导致的冲突.

对如何从原始数据中自动抽取符号表示,在 ILP 中嵌入可微构件是一种探索,例如 Evans 和 Grefenstette[66]提出的∂ILP 即可接入预训练的卷积神经网络,从手写数字图像中学习大小与关系. 戴望州等人[100]提出的诱导学习(abductive learning)则将 FOL 诱导推理与深度学习相结合,实现了逻辑模型和深度学习模型的同时训练. 其他机制尚有待探索.

此外,ILP 良好的可理解性或可对其他形式的机器学习方法产生一定启发. 目前在该方向中的探索可大致分为 2 类:1)将 ILP 作为独立的部件与其他机器学习方法相结合;2)将 ILP 所使用的 FOL 演算嵌入机器学习过程. 例如,戴望州和周志华[101]采用第 1 种策略,试图通过诱导推理来约束统计机器学习. Socher 等人[102]提出的 NTN(neural tensor network)以及 Cohen[103]提出的 TensorLog 则采用第 2 种策略,将 FOL 演算表示为张量运算嵌入深度神经网络. 第 1 种策略有助于保持 FOL 的表达能力,但是将 FOL 演算与机器学习分离为 2 个相对独立的过程;第 2 种策略便于引入"端到端学习",但是一定程度上损失了 FOL 的表达能力.

总的来看,作为一个已经发展了 30 年的机器学习分支方向,ILP 已经取得了很多的进展和成果. 尤其在人们对机器学习的可理解性愈来愈关注的今天,ILP 的研究可能值得更多研究者关注和参与.

参 考 文 献

[1] Zhou Zhihua. Machine Learning [M]. Beijing: Tsinghua University Press, 2016 (in Chinese)
(周志华. 机器学习[M]. 北京: 清华大学出版社, 2016)

[2] Muggleton S H. Inductive logic programming [J]. New Generation Computing, 1991, 8(4): 295-318

[3] Enderton H. A Mathematical Introduction to Logic [M]. 2nd ed. Amsterdam: Academic Press, 2001

[4] Jackson P. Introduction to Expert Systems [M]. 3rd ed. Reading, MA: Addison-Wesley, 1999

[5] Baral C, Gelfond M. Logic programming and knowledge representation [J]. Journal of Logic Programming, 1994, 19(20): 73-148

[6] Plotkin G D. A note on inductive generalization [C] //Proc of the 5th Annual Machine Intelligence Workshop. Edinburgh, UK: Edinburgh University Press, 1969: 153-163

[7] Plotkin G D. A further note on inductive generalizatio [C] //Proc of the 6th Annual Machine Intelligence Workshop. Edinburgh, UK: Edinburgh University Press, 1970: 101-124

[8] Plotkin G D. Automatic Methods of Inductive Inference [M]. Edinburgh, Scotland: Edinburgh University, 1971

[9] Vere S A. Induction of concepts in the predicate calculus [C] //Proc of the 4th IJCAI. San Francisco, CA: Morgan Kaufmann, 1975: 282-287

[10] Sammut C, Banerji R B, Plotkin G D. Learning concepts by asking questions [J]. Machine Learning: An Artificial Intelligence Approach, 1969, 2: 167-192

[11] Muggleton S H. Duce, an oracle based approach to constructive induction [C] //Proc of the 10th IJCAI. San Francisco, CA: Morgan Kaufmann, 1987: 287-292

[12] Muggleton S H, Buntine W. Machine invention of first-order predicates by inverting resolution [C] //Proc of the 5th ICML. San Francisco, CA: Morgan Kaufmann, 1988: 339-352

[13] Muggleton S H. Inverting the resolution principle [J]. Machine Intelligence, 1991, 12: 93-104

[14] Rouveirol C, Puget J F. A simple and general solution for inverting resolution [C] //Proc of EWSL-89. London: Pitman, 1989: 201-210

[15] Muggleton S H. Inverting implication [C] //Proc of the 1st European Workshop on ILP. Berlin: Springer, 1992: 19-39

[16] Muggleton S H. Inverse entailment and Progol [J]. New Generation Computing, 1995, 13(3/4): 245-286

[17] Muggleton S H. Completing inverse entailment [C] //Proc of the 8th Int Workshop on ILP. Berlin: Springer, 1998: 245-249

[18] Nienhuys-Cheng S H, De Wolf R. Foundations of Inductive Logic Programming [M]. Berlin: Springer, 1997

[19] Valiant L G. A theory of the learnable [J]. Communications of the ACM, 1984, 27(11): 1134-1142

[20] Džeroski S, Muggleton S H, Russell S. PAC-learnability of determinate logic programs [C] //Proc of the 5th ACM Workshop on CLT. New York: ACM, 1992: 128-135

[21] Džeroski S, Muggleton S H, Russell S. Learnability of constrained logic programs [C] //Proc of the 6th ECML. Berlin: Springer, 1993: 342-347

[22] Cohen W. PAC-learning a restricted class of logic programs [C] //Proc of the 3rd Int Workshop on ILP. Berlin: Springer, 1993: 41-72

[23] Kietz J U. Some lower bounds on the computational complexity of inductive logic programming [C] //Proc of the 6th ECML. Berlin: Springer, 1993: 115-123

[24] Muggleton S H, Feng, C. Efficient induction of logic programs [C] //Proc of the 2nd Int Workshop on ILP. Amsterdam: Academic Press, 1992: 281-298

[25] Lavrač N, Dčeroski S, Grobelnik M. Learning non-recursive definitions of relations with LINUS [C] //Proc of the 5th European Working Session on Learning. Berlin: Springer, 1991: 265-281

[26] De Raedt L, Bruynooghe M. CLINT: A multistrategy interactive concept-learner and theory revision system [C] //Proc of the 1st Int Workshop on Multistrategy Learning. San Francisco, CA: Morgan Kaufmann, 1991: 175-191

[27] Muggleton, S H, King R D, Sternberg M J E. Protein secondary structure prediction using logic-based machine learning [J]. Protein Engineering, 1992, 5(7): 647-657

[28] Feng C. Inducing temporal fault diagnostic rules from a qualitative model [C] //Proc of the 1st Int Workshop on ILP. Amsterdam: Academic Press, 1991: 403-406

[29] King R D, Muggleton S H, Srinivasan A, et al. Structure-activity relationships derived by machine learning: The use of atoms and their bond connectives to predict mutagenicity by inductive logic programming [J]. Proceedings of the National Academy of Sciences, 1996, 93(1): 438-442

[30] King R D, Whelan K E, Jones F M, et al. Functional genomic hypothesis generation and experimentation by a robot scientist [J]. Nature, 2004, 427: 47-252

[31] King R D, Rowland J, Oliver S G, et al. The automation of science [J]. Science, 2009, 324(5923): 85-89

[32] Dantsin E. Probabilistic logic programs and their semantics [C] //Proc of the 1st Russian Conf on Logic Programming. Berlin: Springer, 1992: 152-164

[33] Poole D. Logic programming, abduction and probability [J]. New Generation Computing, 1993, 11(3): 377-400

[34] Sato T. A statistical learning method for logic programs with distribution semantics [C] //Proc of the 12th Int Conf on Logic Programming. Cambridge, MA: MIT Press, 1995: 715-729

[35] De Raedt L, Kersting K. Probabilistic inductive logic programming [C] //Proc of the 15th Int Conf on ALT. Berlin: Springer, 2004: 19-36

[36] Sato T, Kameya T. PRISM: A language for symbolic-statistical modeling [C] //Proc of the 15th IJCAI. San Francisco, CA: Morgan Kaufmann, 1997: 1330-1335

[37] Muggleton S H. Stochastic logic programs [J]. Advances in Inductive Logic Programming, 1996, 32:254-264

[38] Poole D. The independent choice logic for modelling multiple agents under uncertainty [J]. Artificial Intelligence, 1997, 94(1): 7-56

[39] Costa V S, Page D, Qazi M, et al. CLP (BN): Constraint logic programming for probabilistic knowledge [C] //Proc of the 19th Conf on UAI. San Francisco, CA: Morgan Kaufmann, 2002: 517-524

[40] Vennekens J, Verbaeten S, Bruynooghe M. Logic programs with annotated disjunctions [C] //Proc of the 27th ICLP. Berlin: Springer, 2004: 431-445

[41] De Raedt L, Kimmig A, Toivonen H. ProbLog: A probabilistic prolog and its application in link discovery [C] // Proc of the 20th IJCAI. San Francisco, CA: Morgan Kaufmann, 2007: 2462-2467

[42] Baral C, Gelfond M, Rushton N. Probabilistic reasoning with answer sets [J]. Theory and Practice of Logic Programming, 2009, 9(1): 57-144

[43] Vennekens J, Denecker M, Bruynooghe M. CP-logic: A language of causal probabilistic events and its relation to logic programming [J]. Theory and Practice of Logic Programming, 2009, 9(3): 245-308

[44] Muggleton S H. Learning stochastic logic programs [J]. Electronic Transactions on Artificial Intelligence, 2000, 4 (B): 141-153

[45] Cussens J. Parameter estimation in stochastic logic programs [J]. Machine Learning, 2001, 44(3): 245-271

[46] De Raedt L, Kersting K, Kimming A, et al. Compressing probabilistic prolog programs [J]. Machine Learning, 2008, 70(2): 151-168

[47] Gutmann B, Kimmig A, Kersting K, et al. Parameter learning in probabilistic databases: A least squares approach [C] //Proc of 2008 ECML PKDD. Berlin: Springer, 2008: 473-488

[48] Meert W, Struyf J, Blockeel H. Learning ground CP-logic theories by leveraging Bayesian network learning techniques [J]. Fundamenta Informaticae, 2008, 89(1): 131-160

[49] Inoue K, Furukawa K, Kobayashi I, et al. Discovering rules by meta-level abduction [C] //Proc of the 9th Int Conf on ILP. Berlin: Springer, 2010: 49-64

[50] Muggleton S H, Lin D, Pahlavi N, et al. Meta-interpretive learning: Application to grammatical inference [J]. Machine Learning, 2014, 94(1): 25-49

[51] Gutmann B, Thon I, De Raedt L. Learning the parameters of probabilistic logic programs from interpretations [C] // Proc of 2011 ECML PKDD. Berlin: Springer, 2011: 581-596

[52] De Raedt L, Thon I. Probabilistic rule learning [C] //Proc of the 21st Int Conf on ILP. Berlin: Springer, 2011 :47-58

[53] Bellodi E, Riguzzi F. Experimentation of an expectation maximization algorithm for probabilistic logic programs [J]. Intelligenza Artificiale, 2012, 6(1): 3-18

[54] Bellodi E, Riguzzi F. Learning the structure of probabilistic logic programs [C] //Proc of the 21st Int Conf on ILP. Berlin: Springer, 2011: 61-75

[55] Wang W Y, Mazaitis K, Cohen W. Programming with personalized pagerank: A locally groundable first-order probabilistic logic [C] //Proc of the 22nd Int Conf on Information & Knowledge Management. New York: ACM, 2013: 2129-2138

[56] Dai Wangzhou, Zhou Zhihua. Statistical unfolded logic learning [C/OL] //Proc of the 7th Asian Conf on Machine Learning. JMLR, 2015: 349-361. [2018-11-01]. http://proceedings. mlr. press/

[57] Fierens D, Van den Broeck G, Rekens J, et al. Inference and learning in probabilistic logic programs using weighted Boolean formulas [J]. Theory and Practice of Logic Programming, 2015, 15(3): 358-401

[58] Bellodi E, Riguzzi F. Structure learning of probabilistic logic programs by searching the clause space [J]. Theory and Practice of Logic Programming, 2015, 15(2): 169-212

[59] De Maeyer D, Renkens J, Cloots L, et al. PheNetic: Network-based interpretation of unstructured gene lists in E. coli. [J]. Molicular BioSystems, 2013, 9(7): 1594-1603

[60] Wang W Y, Kong L, Mazaitis K, et al. Dependency parsing for weibo: An efficient probabilistic logic programming approach [C] //Proc of 2014 EMNLP. Stroudsburg PA: ACL, 2014: 25-29

[61] Skarlatidis A, Artikis A, Filippou J, et al. A probabilistic logic programming event calculus [J]. Theory and Practice of Logic Programming, 2015, 15(2): 213-245

[62] Catherine R, Cohen W. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach [C] //Proc of the 10th ACM Conf on Recommender Systems. New York: ACM, 2016: 325-332

[63] Antanas L, Dries A, Moreno P, et al. Relational affordance learning for task-dependent robot grasping [C] //Proc of the 27th Int Conf on ILP. Berlin: Springer, 2017: 1-15

[64] Côrte-Real J, Dutra I, Rocha R. On applying probabilistic logic programming to breast cancer data [C] //Proc of the 27th Int Conf on ILP. Berlin: Springer, 2017: 31-45

[65] Rocktäschel T, Riedel S. End-to-end differentiable proving [C] //Advances in Neural Information Processing Systems 30. Red Hook, NY: Curran Associates, Inc., 2017: 3791-3803

[66] Evans R, Grefenstette E. Learning explanatory rules from noisy data [J]. Journal of Artificial Intelligence Research, 2018, 61: 1-64

[67] Idestam-Almquist P. Learning missing clauses by inverse resolution [C] //Proc of the 5th Int Conf on Generation Computer Systems. Ohmsha: IOS Press, 1992:610-617

[68] Kramer S, Lavrač N, Flach P. Propositionalisation approaches to relational data mining [G] //Relational Data Mining. Berlin: Springer, 2001: 262-291

[69] Kramer S, Pfahringer B, Helma C. Stochastic propositionalisation of non-determinate background knowledge [C] //Proc of the 8th Int Conf on ILP. Berlin: Springer, 1998: 80-94

[70] Flach P, Lachiche N. 1BC: A first-order Bayesian classifier [C] //Proc of the 9th Workshop on ILP. Berlin, Heidelberg: Slovenia, 1999: 92-103

[71] Robinson J A. A machine-oriented logic based on the resolution principle [J]. Journal of the ACM, 1965, 12(1): 23-41

[72] Popplestone R J. An experiment in automatic induction [C] //Proc of the 5th Annual Machine Intelligence Workshop. Edinburgh, UK: Edinburgh University Press, 1969: 203-215

[73] Kearns M, Li Ming, Pitt L, et al. On the learnability of Boolean formulae [C] //Proc of the 9th Annual ACM Symp on Theory of Computing. New York: ACM, 1987: 285-295

[74] Page C D, Frisch A. Generalization and learnability: A study of constrained atoms [C] //Proc of the 2nd Int Workshop on ILP. Amsterdam: Academic Press, 1992: 29-61

[75] Cohen W, Page C D. Polynomial learnability and inductive logic programming: Methods and results [J]. New Generation Computing, 1995, 13(3/4): 369-409

[76] Kakas A C, Flach P A. Abduction and induction in artificial intelligence [J]. Journal of Applied Logic, 2009, 7(3): 251-251

[77] Lloyd J W. Logic for Learning [M]. Berlin: Springer, 2003

[78] Cropper A, Muggleton S H. Logical minimization of meta-rules within meta-interpretive learning [C] //Proc of the 24th Int Conf on ILP. Berlin: Springer, 2015: 65-78

[79] Cropper A, Muggleton S H. Learning efficient logic programs [J/OL]. Machine Learning, 2018 [2018-11-09]. https://doi.org/10.1007/s10994-018-5712-6

[80] Nilsson N J. Probabilistic logic [J]. Artificial Intelligence, 1986, 28(1): 71-87

[81] Becker B, Drechsler R. Binary Decision Diagrams: Theory and Implementation [M]. Berlin: Springer, 1998

[82] Dries A, Kimmig A, Meert W, et al. ProbLog2: Probabilistic logic programming [C] //Proc of 2015 ECML PKDD. Berlin: Springer, 2015: 312-315

[83] Darwiche A. A compiler for deterministic, decomposable negation normal form [C] //Proc of AAAI-02. Menlo Park, CA: AAAI, 2002: 627-634

[84] De Raedt L, Kimmig A. Probabilistic (logic) programming concepts [J]. Machine Learning, 2015, 100(1): 5-47

[85] Richards B L, Mooney R J. Learning relations by pathfinding [C] //Proc of AAAI-92. Menlo Park, CA: AAAI, 1992: 50-55

[86] Zhou Zhihua, Chen Zhaoqian. Hybrid decision tree [J]. Knowledge-Based Systems, 2002, 15(8): 515-528

[87] Towell G, Shavlik J W. Knowledge based artificial neural networks [J]. Artificial Intelligence, 1994, 70(1): 119-165

[88] Tamaddoni-Nezhad A, Chaleil R, Kakas A, et al. Application of abductive ILP to learning metabolic network inhibition from temporal data [J]. Machine Learning, 2006, 64(1/2/3): 209-230

[89] Getoor L, Taskar B, (editors). Introduction to Statistical Relational Learning [M]. Cambridge, MA: MIT Press, 2007

[90] Richardson M, Domingos P. Markov logic networks [J]. Machine Learning, 2006, 62(1/2): 107-136

[91] Bohan D A, Caron-Lormier G, Muggleton S H, et al. Automated discovery of food webs from ecological data using logic-based machine learning [J]. PloS ONE, 2011, 6(12): e29028

[92] Muggleton S H, Dai Wangzhou, Sammut C, et al. Meta-interpretive learning from noisy images [J]. Machine Learning, 2018, 107(7): 1097-1118

[93] Dai Wangzhou, Muggleton S H, Zhou Zhihua. Logical vision: Meta-interpretive learning for simple geometrical concepts [C/OL] //Late Breaking Papers of the 25th Int Conf on ILP. CEUR-W S. org, 2015: 1-16. [2018-11-09]. http://ceur-ws.org/Vol-1636/

[94] Lin D, Dechter E, Ellis K, et al. Bias reformulation for one-shot function induction [C] //Proc of the 23rd ECAI. Ohmsha: IOS Press, 2014: 525-530

[95] Zhou Zhihua. Learnware: On the future of machine learning [J]. Frontiers of Computer Science, 2016, 10(4): 589-590

[96] Princeton University. About WordNet [OL]. 2010 [2018-11-09]. https://wordnet.princeton.edu/

[97] Russell S J. Unifying logic and probability [J]. Communications of the ACM, 2015, 58(7): 88-97

[98] Nishiyama H, Ohwada H. Parallel inductive logic programming system for superlinear speedup [C] //Proc of the 27th Int Conf on ILP. Berlin: Springer, 2017: 112-123

[99] Alchourròn C E, Gärdenfors P, Makinson D. On the logic of theory change: Partial meet contraction and revision functions [J]. The Journal of Symbolic Logic, 1985, 50(2): 510-530

[100] Dai Wangzhou, Xu Qiuling, Yu Yang, et al. Tunneling neural perception and logic reasoning through abductive learning [J]. arXiv preprint, 2018, abs/1802.01173

[101] Dai Wangzhou, Zhou Zhihua. Combining logical abduction and statistical induction: Discovering written primitives with human knowledge [C] //Proc of AAAI-17. Menlo Park, CA: AAAI, 2017: 4392-4398

[102] Socher R, Chen Danqi, Manning C D, et al. Reasoning with neural tensor networks for knowledge base completion [C] //Advances in Neural Information Processing Systems 26. Red Hook, NY: Curran Associates, Inc, 2013: 926-934

[103] Cohen W. Tensorlog: A differentiable deductive database [J]. arXiv preprint, 2016, abs/1605.06523, 2016

Dai Wangzhou, born in 1990. PhD candidate. Received his BSc degree in applied mathematics from Northwestern Polytechnical University in 2010. His main research interests include machine learning and data mining, especially in logic-based machine learning.

Zhou Zhihua, born in 1973. Professor and PhD supervisor. Foreign member of Academy of Europe, Fellow of ACM, AAAI, AAAS, IEEE, IAPR, CCF and CAAI. His main research interests include artificial intelligence, machine learning and data mining.

《计算机研究与发展》征订启事

《计算机研究与发展》(Journal of Computer Research and Development)是中国科学院计算技术研究所和中国计算机学会联合主办、科学出版社出版的学术性刊物,中国计算机学会会刊. 主要刊登计算机科学技术领域高水平的学术论文、最新科研成果和重大应用成果. 读者对象为从事计算机研究与开发的研究人员、工程技术人员、各大专院校计算机相关专业的师生以及高新企业研发人员等.

《计算机研究与发展》于 1958 年创刊,是我国第一个计算机刊物,现已成为我国计算机领域权威性的学术期刊之一. 并历次被评为我国计算机类核心期刊,多次被评为"中国百种杰出学术期刊". 此外,还被《中国学术期刊文摘》、《中国科学引文索引》、"中国科学引文数据库"、"中国科技论文统计源数据库"、美国工程索引(Ei)检索系统、日本《科学技术文献速报》、俄罗斯《文摘杂志》、英国《科学文摘》(SA)等国内外重要检索机构收录.

国内邮发代号:2-654;**国外发行代号**:M603

国内统一连续出版物号:CN11-1777/TP

国际标准连续出版物号:ISSN1000-1239

联系方式:

100190 **北京中关村科学院南路 6 号《计算机研究与发展》编辑部**

电话:+86(10)62620696(**兼传真**);+86(10)62600350

Email:crad@ict.ac.cn

http://crad.ict.ac.cn