経路構造による再帰定義の帰納的学習

6G－2

古澤 光枝　犬塚 信博　世木 博久　伊藤 英則
名古屋工業大学

1 はじめに

帰納論理プログラミング (ILP) は論理プログラムの形で与えられる背景知識と事例から事例を満たす述語を学習する枠組みである．CRUSTACEAN，TIM などのこれまでに提案されたボトムアップ手法は少数事例から効率よく学習するが限られた範囲の述語しか学習できない．これらは1つの非再帰節と1つの再帰節をもつ論理プログラムのみを学習する．本研究では TIM で用いられた経路構造 (path structure) の分析手法を発展させ，経路構造のもつ共通部分に着目した新しいアルゴリズムを提案する．これにより1つの非再帰節と複数の再帰節をもつ論理プログラムを学習することができる．

2 定義

背景知識において真となる基底リテラルを **K** リテラル，引数の入力モードが1つで，残りの引数が1つ以上あり，すべて出力モードである述語を**経路述語**，引数のすべてが入力モードである述語を**判定述語**といい，各々述語を述語記号とする K リテラルを**経路リテラル**，**判定リテラル**という．また，K リテラルは経路リテラルと判定リテラル以外にはないとする．

リテラル $C_1,\ldots,C_h$ に対し，C_h の出力モードである第 m_h 引数が a であり，C_g $(1 \le g \le h)$ の出力モードの第 m_g 引数と C_{g+1} の入力モードの第 n_{g+1} 引数が等しいとき $[c_1-m_1,\cdots,c_g+n_g-m_g,\cdots,c_h+n_h-m_h]$ (c_g は C_g の述語記号) と表わし，a までの**経路**とよぶ．簡単のため経路の一部を α,β,γ とし，また必要なときには $d=[f-j,\ldots]$ を d^{fj} と表わす．判定リテラル $q(a_1,\ldots,a_k)$ に対して $a_i(1 \le i \le k)$ までの経路を d_i としたとき述語名とその各引数までの経路の組 $(q,(d_1,\ldots,d_k))$ を**経路構造**とよぶ．

ホーン節において，左辺のリテラルと同じ述語記号かつ同じ引数の数の右辺のリテラルを**再帰リテラル**とよぶ．また再帰リテラルを使用するホーン節を**再帰節**，使用しないホーン節を**非再帰節**とよぶ．

Inductive Leaning of Recursive Definitions by Path Strucrures
Mitsue Furuzawa, Nobuhiro Inuzuka, Hirohisa Seki and Hidenori Itoh.
Nagoya Institute of Technology.
Gokiso-cho, Showa-ku, Nagoya 466, Japan

3 アルゴリズム

アルゴリズムへの入力はホーン節で表わされる正事例の集合 $F_1,\ldots,F_n$，背景知識 K，出力は1つの非再帰節と複数の再帰節をもつ正事例を導出できる再帰定義 である．対象とする再帰節は1つの再帰リテラルを持ち，再帰リテラルの第 j 引数は再帰節の左辺のリテラルの第 j 引数から導出されるとする．

アルゴリズムを以下に示す．

1. 正事例 $F_1,\cdots,F_n$，背景知識 K を入力する．
2. $i=1$ から n まで 3 ～ 5 を繰り返す．
3. F_i のすべての経路の集合を A_i，すべての経路構造の集合を S_i とする．
4. $u_1^{f1},\cdots,u_m^{fm} \in A_i$ を選び，$U_i=(u_1,\cdots,u_m)$ とし，U_i を非再帰節の目標述語の引数までの経路の組と仮定する．
5. $(q,PS) \in S_i$ より PS,U_i に#演算が施せるならば $PS'=PS\#U_i$ とし，(q,PS') の集合を非再帰節候補 B_i とし，それ以外のときの (q,PS) の集合を S_i' とする．
6. $B := B_1 \cap \cdots \cap B_n$ とする．
7. $i=1$ から n まで 8 ～ 9 を繰り返す．
8. すべての $(q,PS) \in S_i'$ について PS,U_i に\$演算が施せるならば，$D=PS\$U_i$，$PS'=PS\#D$ とし，$(D,(q,PS'))$ の集合を G_i とする．
9. $G_i = \{(D_1,Q_1),(D_2,Q_2),\cdots\}$ を $G_i' = ((D_1',L_1),(D_2',L_2),\cdots,(D_k',\{Q_t, Q_s,\cdots\},\cdots)$ $(D_x' \le D_{x+1}',\ D_k'=D_t=D_s=\cdots)$ とする．
10. 再帰節候補の集合 $R_i = \{(D_2'\#D_1',L_1),(D_3'\#D_2',L_2),\cdots,(U_i\#D_z',L_z)\}$ とする．
11. $R := R_1 \cup \cdots \cup R_n$ とする．
12. B,R をホーン節に書き換える．

経路 $d_1=[f-j,\alpha,\beta]$，$d_2=[f-j,\alpha,\gamma]$ のとき $d_1\$d_2=[f-j,\alpha]$，$d_1\#d_2=[f-j,\beta]$ とする．経路の組 $E=(e_1,\cdots,e_z)$，$D=(d_1^{f1},\cdots,d_m^{fm})$ を考える．d_j^{fj} に対し，$c_j=d_j^{fj}\$e_{s_1}^{fj}\$e_{s_2}^{fj}\$\cdots$ となるとき $E\$D=(c_1,\cdots,c_m)$ とする．また e_k^{ft} に対し $g_k=e_k^{ft}\#d_i^{ft}$ ならば $E\#D=(g_1,\cdots,g_z)$ とする．経路 $d_1=[f-j,\alpha]$，$d_2=[f-j,\alpha,\beta]$ のとき $d_1 \le d_2$ と書く．経路の組 $E_1=(e_{11},\cdots,e_{1z})$，$E_2=(e_{21},\cdots,e_{2z})$ に対して $e_{1x}^{fj} \le e_{2x}^{fj}$ $(1 \le x \le z)$ を満たすならば $E_1 \le E_2$ と書く．

次にアルゴリズムにしたがって例の一部を示す.

背景知識K, 正事例F_1, F_2を入力する. +,−は引数の入力, 出力モードを表わす.

$K = \{dec(+[X|S], -X, -S), nil(+[]), eq(+X, +X)\}$
$F_1 = remove(a, [a], [])$
$F_2 = remove(c, [b, c], [b])$

F_1, F_2と Kから経路の集合 A_1, A_2と経路構造の集合 S_1, S_2を求める. S_1, S_2のみ次に示す. (3)

$S_1 = \{(nil, ([remove-2, dec+1-3])),$
$(nil, ([remove-3])),$
$(eq, ([remove-1], [remove-2, dec+1-2]))\}$
$S_2 = \{(nil, ([remove-2, dec+1-3, dec+1-3])),$
$(nil, ([remove-3, dec+1-3])),$
$(eq, ([remove-2, dec+1-2],$
$[remove-3, dec+1-2])),$
$(eq, ([remove-1],$
$[remove-2, dec+1-3, dec+1-2]))\}$

A_1, A_2から非再帰節までの経路の組 U_1, U_2を選ぶ. (4)

5. の操作により B_1, B_2と S_1', S_2'を求める. (5)

B_1, B_2, Bを合わせて非再帰構造 Bとする. (6)

$B_1 = \{(nil, ([remove-2])), (nil, ([remove-3]))\}$
$B_2 = \{(nil, ([remove-2])), (nil, ([remove-3]))\}$
$B = \{(nil, ([remove-2])), (nil, ([remove-3]))\} \cdots i)$

8. の操作により G_1, G_2を求める. (8) G_2のみ示す.

$G_2 = \{(([remove-1], [remove-2], [remove-3]),$
$(eq, ([remove-2, dec+1-2],$
$[remove-3, dec+1-2]))),$
$(([remove-1], [remove-2, dec+1-3],$
$[remove-3, dec+1-3]),$
$(eq, ([remove-1],$
$[remove-2, dec+1-2])))\}$

G_1, G_2より再帰節構造の集合 R_1, R_2を求める. (9,10)

R_1, R_2を合わせてすべての再帰節構造の集合 R を求める. 11) 次に Rを示す.

$R = \{(([remove-1], [remove-2, dec+1-3],$
$[remove-3, dec+1-3]),$
$\{(eq, ([remove-2, dec+1-2],$
$[remove-3, dec+1-2]))\}), \cdots ii)$
$(([remove-1], [remove-2, dec+1-3],$
$[remove-3]),$
$\{(eq, ([remove-1],$
$[remove-2, dec+1-2]))\})\} \cdots iii)$

B,R をホーン節に書き換える. (12)

$i)\ remove(A, B, C) \ :- \ nil(B), nil(C).$
$ii)\ remove(A, B, C) \ :- \ dec(B, D, E), dec(C, F, G),$
$eq(D, F), remove(A, E, G).$
$iii)\ remove(A, B, C) \ :- \ dec(B, D, E), eq(A, D),$
$remove(A, E, C).$

4 評価

CRUSTACEAN および本システムに2つの正事例をランダムに与え, 表1の結果を得た. 正答率とは1回の学習で正答が含まれている割合いであり, 出力数とは1回の学習で出力される再帰定義の数である. TIM には比較できるデータがないため割愛した. TIM は CRUSTACEAN よりも正答率が高いが remove,merge などの複数の再帰節をもつ再帰定義を導出できない. 本システムは導出時間においてはやや劣るが正答率において CRUSTACEAN を上回る. また2つの再帰節を含む remove,merge の定義を学習することができた.

	正答率 [%]		出力数		時間 [s]	
	C	M	C	M	C	M
delete	64.5	97.4	9.2	3.1	0.8	0.9
member	70.9	100.0	1.8	2.5	0.2	0.7
append	60.0	95.3	7.8	8.5	0.7	5.4
reverse	79.0	97.2	5.0	4.5	0.7	1.4
remove	-	87.2	-	5.2	-	3.7
merge	-	91.6	-	8.0	-	5.7

表 1: CRUSTACEAN(C) と本システム (M) の比較

5 考察

ILP システム CRUSTACEAN は正事例の引数を項で表わし, 項の繰り返しの分析により再帰節を求める. また TIM は経路の前方部の繰り返しにより再帰節を求める. このため, これらは1つの再帰節を持つ論理プログラムしか学習できない. 本システムでは経路構造間の経路の共通部分を調べることにより, 複数の再帰節を持つ論理プログラムも学習可能になった. 目標述語の引数の数を m, K リテラルの引数の最大の数を h, K リテラルの数を最大の数を g, 背景知識を探索する深さを d をすると, 経路の最大の数$= (h*g)^d * m$ となる. 学習のための計算量はこの経路数に対し多項式時間となるが CRUSTACEAN や TIM とほぼ同じである. 出力定義数も多くなっているので, 負事例を利用して出力された定義をしぼり込むことは今後の課題である.

参考文献

[1] D.W.Aha,S.Lapointe,C.X.Ling, and S.Matwin.: Learning recursive relations with randomly selected small traning sets, Proc 11th Int'l Conf on Machine Learning(1994)

[2] Peter Idestam-Almquist : Efficient Induction of Recursive Definitions by Structural Analysis of Saturations, Proc.ILP95(1995)