1K-06 複数の最弱仮説を利用する仮説探索

中谷 唯† 近山 隆††
†東京大学大学院工学系研究科 ††東京大学大学院新領域創成科学研究科

1 概要

帰納論理プログラミング (ILP) は1階述語論理上の帰納推論システムである。ILP では、入力として与えられた正例、負例、背景知識の条件を満たす仮説を、一般・特殊の半順序関係で結ばれた仮説候補のなす概念束 (仮説空間) 内を探索することにより求める。代表的 ILP システム Progol では、1つの正例から最弱仮説 (msc) を求め、これをボトムとする部分束内を効率的に探索している。本発表では、Progol において従来正例だけを用いてきた msc のほかに負例から求めた msc を併せて用いることにより、効率的な仮説探索が可能で、場合によっては仮説空間が縮小可能であることを述べる。また実際に効率的な仮説探索の手順を提案し、適用を試みる。

2 ILP システム Progol の仮説探索

帰納論理プログラミング (Inductive Logic Programming、ILP) は一階述語論理の部分系である論理プログラムを知識表現とする、帰納推論の一種である。帰納推論は演繹推論と対をなしており、「具体的事例が与えられるとき、その事例を説明する一般法則を導き出すプロセス」である。

ILP システムでは、一般・特殊の半順序関係で候補仮説が結ばれた仮説空間の中を探索する。ILP システムの一つ Progol では、探索に先立ち正例の一つを選んでそれを伴意する仮説の中で最も特殊な仮説 (最弱仮説 : msc) を求め、探索すべき仮説空間を縮小している [1]。また Progol では A^* アルゴリズムに基づく探索によって仮説を求める。

A Search Algorithm with Multiple Most Specific Clauses
Tadashi Nakaya and Takashi Chikayama
The University of Tokyo
7-3-1 Hongo, Bunkyo, Tokyo 113-8656 Japan

3 負例の利用による仮説探索の効率化の可能性とその手順

Progol では msc 生成の際に正例を用いているが、一方の負例は仮説の評価に用いるだけで探索空間の縮小には利用されてこなかった。本研究では正例と似たような形をした負例を利用して、より効率的に仮説を探索することができないか考える。

ここでは X、Y、Z を変数として X が Y の「おじさん」であるという関係を導くことを考える。また、次の仮説が求まるような一連のデータ (正例、負例、背景知識) を Progol に与えて実行したものとする。

$$X\ is\ uncle\_of\ Y \leftarrow X\ is\ brother\_of\ Z,\ Z\ is\ parent\_of\ Y.$$

背景知識 Bとある例 e により求められる msc を $msc(B,e)$ と書くと、与えられた正例の一つ

$e_1^+ = uncle\_of(\text{カツオ}, \text{タラオ}).$

について $msc(B,e_1^+)$ は

$$uncle\_of(X,Y) \text{ :- } same\_parent(X,Z), brother\_of(X,Z), parent\_of(Z,Y), mother\_of(Z,Y).$$

となった。従来 msc を求める際には例として正例のみを用いていたが、与えられた負例の一つ

$e_1^- = uncle\_of(\text{ワカメ}, \text{タラオ}).$

について $msc(B,e_1^-)$ を求めると、

$$uncle\_of(X,Y) \text{ :- } same\_parent(X,Z), sister\_of(X,Z), parent\_of(Z,Y), mother\_of(Z,Y).$$

となった。この $msc(B,e_1^-)$ を $msc(B,e_1^+)$ と比較するとその違いは述語1つのみであり ($bother_of$が $sister_of$になっている)、これは e_1^+ と e_1^- の違いを反映しているものと考えられる。すると、$msc(B,e_1^+)$ にのみ存在する$\overline{brother_of(X,Z)}$は、求める仮説の本体部の一部となる可能性が高いのではないか、と推定することができる。そこで仮説空間の探索の際にこの$\overline{brother_of(X,Z)}$に関連する仮説候補から探

索を始めることで、早いうちに高い評価値が得られ、枝を苅ることができるので、結果として計算時間を短縮することができるものと思われる。特にこの場合は

$uncle\_of(X,Y)$:- $brother\_of(X,Z)$.

より特殊な仮説のみを探索すればよい。

以上を踏まえて、効率的な仮説探索の手順としておおよそ以下のようなものが考えられる。

1. 与えられたすべての正例、負例についてmscを求める。
2. 正例のmscと負例のmsc 1つずつを選び、最も類似度の高い組を求める。
3. 2. で求めた組について、正例のmscに含まれ負例のmscに含まれないリテラルを求める。
4. 3. で求めたリテラルが優先されるように、A^*アルゴリズムに基づく仮説探索を行なう。

4 例題による提案手法の試行

4.1 Muggletonの事例作成プログラム

今回取り上げたのはMichalskiの列車の分類問題である。この問題ではMuggletonのプログラム[2]を利用することによって簡単に例題を作成することができる。

車両の形状等に関していくつかの述語が用意されており、それを用いて例えば

east(T) :- has_car(T,C), short(C), closed(C).

という仮説を与えるとこれを満たす正例・負例(とその背景知識)をランダムに作ってくれる。

今回は事例となる列車を正負各8つ生成した。そのうち第6番目の正例の列車は、以下のように表現されるものとなった。

$east(east6)$.
$train(east6).has\_car(east6, car\_6\_1)$.
$car(car\_6\_1).closed(car\_6\_1).short(car\_6\_1)$.
$load(car\_6\_1, triangle, 1).wheels(car\_6\_1, 2)$.
$has\_car(east6, car\_6\_2).car(car\_6\_2)$.
$infront(east6, car\_6\_2, car\_6\_1)$.
$short(car\_6\_2).open(car\_6\_2)$.
$double(car\_6\_2).load(car\_6\_2, circle, 1)$.
$wheels(car\_6\_2, 2)$.

4.2 提案した探索法の試行

提案手法に基づき、生成された正・負の事例すべてについて最弱仮説を求めたところ、第6番目の正例、第2番目の負例の最弱仮説はそれぞれ

$east(A)$:-
$infront(A,B,C), has\_car(A,B)$,
$has\_car(A,C), closed(C), short(B), short(C)$,
$open(B), double(B), load(B, circle, 1)$,
$load(C, triangle, 1), wheels(B,2), wheels(C,2)$.

$east(A)$:-
$infront(A,B,C), has\_car(A,B)$,
$has\_car(A,C), short(B), short(C), open(B)$,
$open(C), double(B), load(B, triangle, 1)$,
$load(C, triangle, 1), wheels(B,2), wheels(C,2)$.

と、類似したものが得られた。*closed*は第6番目の正例の最弱仮説にのみ含まれるのでこの最弱仮説を用い、*closed*を優先して探索することにする。「優先」については、現段階では正例msc内のリテラルを並びかえることで代用した。こうしてProgolを動作させた結果、探索ノード数は

*closed*を優先した方:567
*closed*を優先しなかった方:585

となって正例mscのみに含まれる述語である*closed*を優先すると、探索ノード数が少なくなった。

5 まとめと今後の方針

本稿では、ILPの概要とその主なシステムの一つであるProgolについて述べたあと、負例を利用して仮説空間を縮小することにより、計算量の縮小を実現できる可能性があることを示し、そのための手順の外枠を述べ、例で試行して得た結果を記した。

今後は効率的な仮説探索のさらに具体的な手順の構築を目指す。

参考文献

[1] Muggleton, S. : "Inverse Entailment and Progol," New Generation Computing, Special issue on Inductive Logic Programming 13 (3 - 4), 1995

[2] ftp://ftp.cs.york.ac.uk/pub/ML_GROUP/GenerateTrains/