ICOT Technical Memorandom: TM-1100

TM-1100

論理プログラムの並列帰納学習システム

坂本　忠昭（三菱）

September, 1991

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191 ~5
Telex ICOT J32964

Institute for New Generation Computer Technology

論理プログラムの並列帰納学習システム

Parallel inductive learning system for logic programming

坂本 忠昭

Tadaaki Sakamoto

三菱電機(株) 中央研究所

Mitsubishi Electric Corporation

与えられた例から論理プログラムを帰納的に学習するシステムについて述べる。従来の帰納学習システムでは、システムからの質問や背景知識の準備等によって教師に対して大きな負担を強いている、あるいは強いヒューリスティクスの使用等によって探索効率を得るために別解を犠牲にしているといった問題点が指摘されている。そこで、並列処理を用いて、教師にかかる負担を軽減し、かつ有効な解を全て探索する帰納学習システムを並列推論マシン上で構築した。本稿では、そのシステム概要を述べると共に、文法学習問題等の例題を通してその有効性を検討する。

1 はじめに

1988年にMuggletonとBuntineが逆導出を用いた論理プログラムの帰納学習システムCIGOL[5]を発表して以来、論理プログラムの帰納学習に関する研究が盛んに行なわれるようになり、現在までに数多くの学習方法や学習システムが提案されている[6][7][8]。

それらの方法やシステムに共通な目的は、背景知識K及び$K \not\vdash E^+$である正の例の集合E^+と負の例の集合E^-を与えると、$K \cup H \vdash E^+$かつ$K \cup H \not\vdash E^-$を満たす適切な仮説集合Hを効率的に生成あるいは発見するというものである。

しかし、このような仮説集合を効率良く発見するための手法はまだ確立されていない。一般に、探索効率を得るためには、教師に探索制御に関する質問を行なったり十分な背景知識を準備することによって探索空間を狭める方法や、背景知識や仮説の表現に制約を加えたり、ヒューリスティクスを用いた探索制御を行なうことによって探索空間を狭める方法が用いられている。ところが、これらの方法では、質問に答えたり背景知識を準備したりする教師の負荷が過過ぎるという点[3][8]や、探索効率を求めるために強い制約を用いることによって、より良い仮説が探索される可能性を放棄してしまう点[4]が指摘されている。

そこで、本稿では、並列処理を取り入れることによって有用な仮説集合を効率的に発見する能力を持つ並列帰納学習システムについて述べる。

2 システム概要

2.1 基本方針

まず、論理プログラムの帰納学習システムが持つべきと考えられる特長を以下に示す。

第1点は教師に過大な負担をかけないということである。CIGOL等では、生成された仮説の真偽の問い合わせを教師に行なっているが、このような概念レベルを扱うことは教師にとって大きな負担になる。特に、複雑な概念を学習対象とする場合には、中間的な仮説の真偽に対する質問への回答が非常に困難となることは容易に想像できる。従って、教師は、正・負の例の入力や例の所属性質問への回答といった具体例に関するものだけを行なえばよいことが必要である。

第2点は新しい述語の生成能力を持っていることである。多くのシステムでは、新しい述語の生成能力を持たず、それら全てを背景知識として予め持たせておくというアプローチをとっている。しかし、一般的に言って、ある概念を学習する場合に、そのサブ概念を全て背景知識として与えておくことは、それを与える教師にとって大きな負担となる。従って、新しい述語を自分で生成する能力を持っていることが望ましい。さらに、背景知識を用いる場合にも、生成された新しい述語は必要な背景知識の予測として効果的に働くことが期待される。

第3点は仮説候補を全て探索するということである。一般に、複雑な概念の学習は1回の例の入力では終了しないのが普通であり、学習サイクルを何回も繰り返しながら正しい概念に近付いて行く。このとき、仮説の探索経路は常に決定的とは限らず、むしろ非決定的要素が多いと考えられる。そのため、学習サイクルのある時点では複数の仮説候補が存在することがありうる。従来は、教師の手を借りたり、ヒューリスティクスを用いて1つに絞っているが、それらは本質的に未完成の概念であるため1つに絞り込むことは容易ではない。加えて、仮説を1つに絞ることは別解の探索可能性を放棄することになる。従って、ここでは複数の仮説候補は全て残していくという方針をと

る。ただし、仮説候補に至らない枝はできるだけ刈ることが望ましいため、問題に依存せずに有効な制約やヒューリスティクスは積極的に用いるものとする。

2.2 並列処理の導入

次に、基本方針の実現方法について述べる。

まず、教師の負担を軽くするために、教師は例の入力以外は行なわないようにする。さらに、仮説候補は全て探索するという方針と合わせると、探索時に発生する非決定的要素を扱う枠組が必要になる。そこで、本システムでは、非決定的な分岐を持つ探索空間において複数の仮説候補を同時にかつ高速に探索する手法として並列処理を用いる。特に、一度分岐した枝は他の枝に無関係に探索を行なえるため、並列処理による高速化が期待できる。

また、帰納方法としては新しい述語を生成する能力を持つ帰納操作 intra-construction を含む逆導出 [5] の枠組を用いる。新しい述語の生成能力により、背景知識を完備する必要がなくなり、この点でも教師の負担の軽減がはかれる。

2.3 並列帰納学習アルゴリズム

並列探索を取り入れた帰納学習アルゴリズムを図1に示す。

システムは、教師に対して3種類の例の入力を要求する。具体例と正の例はどちらも学習対象概念の正の例であるが、具体例はその概念を生成する際に帰納操作の適用対象となるのに対し、正の例は生成された仮説概念の試験にのみ用いられ帰納操作の適用対象にはならない点が異なる。また、負の例も正の例と同様に試験にのみ用いられる。

各背景知識に対する処理はお互いに独立であるため並列に実行される。また、各背景知識 K_j^i に帰納操作を施して仮説知識集合 K_j' を求める処理においても、並列処理が行なわれている。これは、ある背景知識に対する帰納操作の適用が非決定的であり、適用可能な帰納操作が同時に複数存在するためである。これらの帰納操作はお互いに独立であるため、やはり並列に実行される。

なお、一般の帰納学習システムでは具体例が入力されると直ちに帰納操作が適用されるのに対し、本システムは背景知識によって導くことができない正の例が存在して初めて帰納操作が適用されるという方針をとっている。同時に、その正の例が帰納操作適用の終了条件として働いている。すなわち、正の例を満たす仮説知識が生成された時点で帰納操作の適用を終了するというものである。これは、本システムにおける帰納操作が全て一般化を行なうものであるため、操作の過剰な適用による過度の一般化 [1] を防ぐための手段である。

2.4 帰納操作

帰納操作としては、基本的には CIGOL で用いられている3つの操作を用いているが、オリジナルのアルゴリズムには非決定的要素が多いため新たに幾つかの制限を追加している。また、次節で述べる文法学習の例題用に intra-construction を特化した帰納操作を追加している。

truncation

2つの単一節間の最小一般化を行なう操作である。ただし、生成された一般化節のうち、それに含まれる変数が全て異なるようなものは削除する。これは、そのような節が一般的過ぎて意味を持たないと考えられるからである。この制約は他の操作についても同様である。

absorption

2つの節 C_1, C_2 から導出によって節 C が導かれるという関係を仮定し、与えられた C_1 と C から C_2 を求める操作である。ただし、C_1 は単一節に制限されている。さらに、本システムでは absorption を自己再帰節を生成する操作として限定している。これによって、C の頭部と C_1 の述語記号及びアリティが等しいことが操作適用の条件となっている。

intra-construction

単一節でない節 A と単一節集合 $B_1, B_2, \ldots, B_m$ から節集合 $C_1, C_2, \ldots, C_m$ が導出されるという関係を仮定し、与えられた $C_1, C_2, \ldots, C_m$ から A と $B_1, B_2, \ldots, B_m$ を求める操作である。求められた節には、新しく生成された述語が含まれている。CIGOL では、新しい述語が生成された場合には教師に対してその述語名の問い合わせを行なっているが、本システムでは行なっていない。なお、新しく生成された述語が背景知識に既に存在すると判断した場合には、システムが自動的に述語名の付け替えを行なう。

intra-construction-g

節集合 $C_1, C_2, \ldots, C_m$ が与えられたときに、intra-construction がこれら全てに対して適用するのに対し、その一部に対して適用する操作が intra-construction-g である。操作の内容は intra-construction と同様であるが、適用対象節の組み合わせ数の増加を抑えるため、及びこの操作が次節の文法学習問題用であるため、入力される節の形式が

$$s([t_1, \ldots, t_h|X], X)$$

に制限されている。ただし、$t_1, \ldots, t_h$ $(h \geq 1)$ は変数を含まない項、X は変数である。

なお、本システムは、並列論理型プログラミング言語 KL1[2] で記述され、並列推論マシン Multi-PSI（16PE 版）上で動作している。

[step 0] 背景知識の初期集合を仮説知識集合 $\mathcal{K}_0$ とする。（背景知識のない場合は $\mathcal{K}_0 = \{\emptyset\}$ とする。）

[step i+1]

1. step i で得られた仮説知識集合 $\mathcal{K}_i = \{\boldsymbol{K}_1, \ldots, \boldsymbol{K}_s\}$ を背景知識集合とする。
2. 教師から、具体例の集合 $\boldsymbol{I} = \{I_1, \ldots, I_l\}$ 、正の例の集合 $\boldsymbol{E}^+ = \{E_1^+, \ldots, E_m^+\}$ 、負の例の集合 $\boldsymbol{E}^- = \{E_1^-, \ldots, E_n^-\}$ を入力する。
3. 各背景知識 $\boldsymbol{K}_j (1 \leq j \leq s)$ に対して以下の処理を並列に行ない、仮説知識集合 $\mathcal{K}'_j$ を求める。
   (a) $\boldsymbol{K}_j \vdash I_k$ なる I_k を除いた具体例の集合を I' とし、$\boldsymbol{K}'_j = \boldsymbol{K}_j \cup I'$ とする。
   (b) $\boldsymbol{K}'_j \vdash E_k^-$ なる E_k^- が存在すれば、$\mathcal{K}'_j = \emptyset$。
   そうでなければ、
   $\boldsymbol{E}^+ = \emptyset$ または全ての k $(1 \leq k \leq m)$ において $\boldsymbol{K}'_j \vdash E_k^+$ であれば、$\mathcal{K}'_j = \{\boldsymbol{K}'_j\}$ とする。
   そうでなければ、$\boldsymbol{K}'_j$ に帰納操作を施して $\boldsymbol{E}^+$ を満たし $\boldsymbol{E}^-$ を満たさないような仮説知識を全て求め、その集合を $\mathcal{K}'_j = \{\boldsymbol{K}''_{j1}, \ldots, \boldsymbol{K}''_{jt}\}$ とする。
4. 各仮説知識集合をまとめ、$\mathcal{K}_{i+1} = \bigcup_j \mathcal{K}'_j$ とする。

図 1: 並列帰納学習アルゴリズム

3 例題

帰納学習の例題として、アーチ問題 [5] 及び文法学習問題をとりあげた。アーチ問題は、

```
具体例：
arch([ ],beam,[ ]).
arch([block],beam,[block]).
arch([brick],beam,[brick]).
正の例：
arch([block,brick],beam,[block,brick]).
```

といったアーチの構造に関する例を入力し、その概念を学習させる問題である。なお、[5] では上記の例を全て具体例として扱っているが、本システムでは正の例の入力が操作適用のトリガとなるため、4 番目の例を正の例として用いている。一方、文法学習問題は、

```
具体例：
s([det,noun,verb|X],X).
s([det,noun,verb,det,noun|X],X).
s([det,noun,verb,det,adj,noun|X],X).
s([det,adj,noun,verb|X],X).
s([det,adj,noun,verb,det,noun|X],X).
正の例：
s([det,noun,verb,det,adj,adj,noun|X],X).
s([det,adj,adj,noun,verb,det,adj,adj,noun|X],X).
```

といった文の例を与え、

```
s(X,Z) :- np(X,Y),vp(Y,Z).
np([det|X],Y) :- n(X,Y).
n([noun|X],X).
n([adj|X],Y) :- n(X,Y).
vp([verb|X],X).
vp([verb|X],Y) :- np(X,Y).
```

という文法を求めるものである。

これらの実行結果を表 1に示す。 なお、文法学習 I は上記の例を一度に与えた場合であり、文法学習 II は上記の例を 2 つに分けて 2 段階の学習を行なった場合である。すなわち、第 2 段階の学習は、第 1 段階で求まった仮説知識集合を背景知識とし、第 1 段階で用いなかった具体例と正の例を用いて新たな仮説知識集合を生成するものである。

表より、これらの仮説知識が数秒から十数秒の短時間で生成されていることがわかる。得られた仮説知識はいずれも与えられた全ての具体例及び正の例を満たしている。ただし、アーチ問題では 6 つの仮説知識が得られたが、このうち 4 つは一般的過ぎて正しい概念ではなかった。しかし、これらは最初に負の例を入力することによって除くことができる。一方、文法学習問題 I 及び II-2 で得られた 8 つの仮説知識は、先の文法を表すもの及びそれと少しずつ形の違うものであったが、意味的には全て同じものであった。

表 1: 例題の実行結果

	探索木のノード数	仮説知識数	探索時間 (sec)
アーチ	10	6	1.3
文法学習 I	385	8	14.1
文法学習 II-1	24	8	2.1
文法学習 II-2	85	8	5.1

4 評価・検討

CIGOLではアーチ問題の学習の際に、新たに生成された仮説の真偽に関する問い合わせを6回、新しい述語名の入力要求を2回行なっている。これに対し、本システムでは教師は最初に例を与えるだけで良く、しかも、2つの妥当な仮説知識を生成している。アーチ問題よりも探索空間の広い文法学習問題に対しても、教師は例の入力を行なうだけで文法を生成することができた。また、新しい述語の生成能力を持つため、背景知識を与えても与えなくても良いという点からも、教師の負担は大幅に軽減されたと言える。ただし、幾つかの例題の実行の結果、例の与え方に工夫が必要であり、例を上手に与えなければたちまち探索空間が広がるということもわかった。従って、この例の与え方つまり教師の教え方に関する検討が必要であると思われる。

一方、探索効率については、本稿で取り上げた例題程度の規模の探索空間を持つ学習問題に対しては、妥当な時間内で全解を求めることができると思われる。また、同じ例を与える場合でも、一度に全てを与えるよりも、幾つかに分けて何段階かかけて学習させる方が全体として探索空間が小さくなり効率も良いことがわかった。ただし、この例の分け方は重要な問題の1つであり、先に述べた教師の教え方の1つであると考えられる。さらに、学習アルゴリズムに含まれる並列性が高いことから、並列推論マシンのPE数の増加による探索の高速化も期待できる。

ところで、最初に述べたように、本システムの方針は、教師からの探索制御情報を減らし、かつ有用な仮説を全て探索するというように非決定的要素を増加させ探索空間を広げる方向にある。そして、広がった探索空間を、並列処理の持つ計算能力を用いて全解探索するというものである。しかし、並列処理が高速とはいえ、探索空間の拡大は指数関数的であるため、このまま問題の規模を拡大しても対処できるかといえばそれは疑問である。従って、そのような場合にも探索効率を保つためには、探索空間をある程度の規模に抑えておくための何らかの方法が必要と思われる。もちろん、その方法は教師に負担をかけず、有用な仮説の探索を妨げないなければならない。そのような方法としては例えば、先に述べた教師の教え方に関するガイドラインのようなものを見つけ出す方法や、与えられた例をシステムが自分でクラスタリングし何段階かの学習サイクルに分けることによって1回の学習当りの探索空間を狭める方法等が考えられる。また、負の例は枝刈りに用いられるため、教師から上手に負の例を引き出す方法も探索空間を狭めるために有効であると考えられる。

5 おわりに

並列処理を取り入れることによって有用な仮説集合を効率的に発見する能力を持つ並列帰納学習システムについて述べた。並列処理による全解探索機能を用いることによって、教師は例の入力を行なうだけで良く、その負担が軽減された。また、文法学習問題等の例題を通して、効率良く仮説集合を求めることができた。今後の課題としては、問題の規模が大きくなった場合にも探索効率を保つために、探索空間の拡大を抑制する方法の検討が必要である。

謝辞

本研究は第5世代コンピュータ・プロジェクトの一環として行なわれた。日頃御指導頂いているICOTの古川康一研究所次長、長谷川隆三第5研究室室長に感謝致します。また、貴重な助言を頂いた三菱電機中央研究所の竹内彰一氏（現在(株)ソニーコンピュータサイエンス研究所）、高橋和子氏に感謝致します。

参考文献

[1] Bain,M.: Experiments in Non-monotonic First-order Induction, *Proc. of ILP-91*, 1991, pp.195 206.

[2] Chikayama,T.,Sato,H.and Miyazaki,T.: Overview of the Parallel Inference Machine Operating System (PIMOS), *Proc. of Int. Conf. on FGCS'88*, 1988, pp.230–251.

[3] 石坂裕毅, 有川節夫：モデル推論, 情報処理, Vol.32, No.3, 1991, pp.236–245.

[4] Kietz,J.-U. and Wrobel,S.: Controlling the Complexity of Learning in Logic through Syntactic and Task-Oriented Models, *Proc. of ILP-91*, 1991, pp.107–126.

[5] Muggleton,S. and Buntine,W.: Machine Invention of First-order Predicates by Inverting Resolution, *Proc. of Machine Learning 88*, 1988, pp.1–14.

[6] Muggleton,S. and Feng,C.: Efficient Induction of Logic Programs, *Proc. of the First Conf. on Algorithmic Learning Theory*, Ohmsha Ltd., 1990, pp.368–381.

[7] Quinlan,J.R.: Learning Logical Definitions from Relations, *Machine Learning*, 5, 1990, pp.239–266.

[8] Rouveirol,C.: ITOU: Induction of First Order Theories, *Proc. of ILP-91*, 1991, pp.127–157.