帰納推論における知識の戦略的探索

2J－1

今木 常之，　渕 一博
東京大学工学部 *

1　はじめに

自動プログラミングにおいて，所望のプログラムの振舞いを入出力の例で示し，それからプログラムを合成する方法がある．このようなアプローチは，帰納推論に基づいている．特に対象を論理プログラムに絞ったものを，帰納論理プログラミングと呼ぶ．この手法ではプログラム生成の際，既に得られている論理プログラムを背景知識として再利用することができる．その一方で，プログラムはそれらの組合せとして合成される為，背景知識として適切なものが用意されていないとプログラム生成が難しくなってしまう．そこで本研究では，プログラム生成の際に戦略的な探索制御をおこなうことを提案し，この手法の改善を計る．

2　帰納推論に基づくプログラム合成

2.1　帰納論理プログラミングの一般的枠組

リストを反転する論理プログラム $reverse/2$ を例にとると，これは次のように与えられる:

$reverse([],[]) \leftarrow$
$reverse([u|x],y) \leftarrow reverse(x,z), append(z,[u],y)$

これから次のグランドアトムが導出される:

$reverse([],[]), reverse([a],[a]),$
$reverse([a,b],[b,a]), reverse([a,b,c],[c,b,a]), \ldots$

帰納論理プログラミングは，逆にこのグランドアトムから論理プログラム $reverse/2$ を導く．ここで，$append/3$ は背景知識として与えられているものと仮定する．

2.2　GOLEM での帰納推論

Plotkin によって形式化された相対汎化に対し幾つかの制約を加え，これを効率的に求める方法は Muggleton によって提案され，代表的な帰納論理プログラミング・システム GOLEM として実装されている [1]．このシステムでは，背景知識はグランドアトムとして与え，その下でグランドアトムとして与えられる事例を説明する理論として論理プログラムを生成する．まず最初に説明すべき事例がランダムに決定され，それに対して生成された理論を他の事例も説明できるように修正していく，という方針をとり，背景知識は，その引数と事例の引数との対応関係において利用される．そのため，背景知識の豊富なグランドインスタンスを用意する必要がある．

3　戦略的探索の利用

3.1　差異法による理論探索

本研究では差異法による帰納推論を用い，背景知識の組合せとしてのみでなく戦略的に理論を探索することを提案する．グランドアトムである事例の中で，引数の値が近いもの同士を比較し，その差の原因を同定するという方針をとる．以下，幾つかの例を取り上げる:

reverse

正… $reverse([1,3,5],[5,3,1]), reverse([2,3,5],[5,3,2])$
負… $reverse([2,3,5],[5,3,1])$

[3,5] の部分を固定して考えると，第一引数のリストの先頭要素と，第二引数のリストの最後尾の要素が同じ場合，述語 $reverse$ は真である，ということが導かれる．このことが残りのリスト [3,5] にも言えるので，再帰的にこの事例を説明する理論を記述できることが分かる．また，ここで必要なのが，リストの最後尾の要素を取り出す論理プログラムであるということが分かる．

sort

$sort([3,2,5,1,7],[1,2,3,5,7]), sort([4,2,5,1,7],[1,2,4,5,7])$
$sort([0,2,5,1,7],[1,2,0,5,7]), sort([6,2,5,1,7],[1,2,6,5,7])$

1,2,5,7 の位置を固定して考えると，第一引数のリストの先頭要素を変えていき，第二引数の要素も対応して変化させると，上の二つのグランドアトムは正だが，下の二つは負である．結局変化要素が 2 以上 5 以下なら正であることが導かれる．ところでこ

---

* "Strategic Search for Theory in Inductive Inference"
Tsuneyuki IMAKI, Kazuhiro FUCHI
Faculty of Engineering, The University of Tokyo

の値は第二引数の，変化要素の一つ前の値と一つ後ろの値である．更に他の例について調べた場合もこの事が成り立つと分かり，先頭要素を除いた残りのリストに対しても同様の事が成り立っていることが分かると，これをプログラムとして整理して挿入ソートが導かれる．

mult

正… $mult(4,1,4), mult(4,2,8), mult(4,3,12),$
$mult(5,1,5), mult(5,2,10)\ldots$

第一引数を 4 に固定して，第二，第三引数のみを抽出した述語 $p/2$ を考える:

$p(1,4).p(2,8),p(3,12)$ これから

$p(B,C):-dec(B,D),p(D,E),add(4,E,C).$

という理論が得られたとして，更に第一引数を 5 に固定した場合の理論:

$p(B,C):-dec(B,D),p(D,E),add(5,E,C).$

が得られたとすると，これらを更に第一引数についても一般化して

$mult(A,B,C):-dec(B,D),mult(A,D,E),add(A,E,C).$

が得られる．

以上の例が示すように，事例間の微妙な差異に着目することで，プログラム生成の際に幾つかの指針を得ることができる．その指針をもとにして，背景知識が無い時でもある程度必要な物を自動生成することが可能である．また，*mult* の例が示すように，一部の引数を固定し，一部の引数の変化のみに着目することで，段階的に理論を形成することができる．

3.2 対話型処理系としての実装

GOLEM は一括型のシステムであり，本研究の手法も一括型として実装することができる．しかし，本研究の手法は GOLEM の様にランダムに事例を選ぶのではなく，どのような事例が必要かはシステム側が決めることになる．つまり，ランダムに用意された事例からは適切な推論を行なえない可能性がある．また，一括型として実装する場合には，与えられた事例を適当に整理する必要がある．

一方，対話型として実装される場合は，人間がシステムによって必要とされる事例について情報を与えなくてはならない．対話型のシステムとして有名な物に CIGOL[2] がある．本研究の手法が CIGOL と異なる点は，CIGOL ではシステムが推論の過程で返す理論について，その正否を人間が判断してシステムに教えるのに対し，本研究の場合はシステムが必要とする事例を人間が捕捉する，という形をとることである．人間側への負担は後者の方が軽いと思われる．また，システムが最小限必要とする事例を与えれば良いので，一括型のように余分な事例を用意する手間を回避できる．また，対話相手が人間である必要はなく，例えば実世界の環境を相手とする場合，その環境を観察する能力がシステムに備わっていれば，システムが自動的に必要な事例を獲得し，学習することができる．

4 様相論理への展開

上記の差異法において，固定した引数というのは様相として捉えることができる．可能世界ごとに理論を生成し，その後で他の世界での理論と比較してそれらを統合する．理論を様相論理として記述することの意義として，引数によって理論が全く異なってしまう様な場合，それを可能世界の推移として考えれば，理論をより美しく記述できることが期待される．また，例外を扱う際にも同様のことが言える．

5 まとめ

帰納論理プログラミングにおける問題として背景知識の用意に人間側の負担がかかることを挙げ，その原因がプログラムを背景知識の組合せとしてのみ探索していることにあると指摘した．またその改善方法として，差異法による帰納推論を用いることを提案し，実際の例に適用した．ただし，今回取り上げた例はトーイプログラムであり，今後はこの手法の実装を完成させ，応用問題にも適用して評価する予定である．

参考文献

[1] S. Muggleton. Efficient induction of logic programs. In *Proceedings of the First International Workshop on Algorithmic Learning Theory*, pp. 368–381, 1990.

[2] S. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In *Proceedings of 5th International Conference on Machine Learning*, pp. 339–352, 1988.