評価順序の固定されたルールを導出する帰納推論手法 WINE

1H−6

毛利 隆夫　田中 英彦
{mohri,tanaka}@mtl.t.u-tokyo.ac.jp
東京大学　工学部

1　はじめに

与えられた正／負のデータから，それらを判別するようなルールを導出する帰納推論は，エキスパートシステムにおける知識獲得の自動化などの応用を目指して，活発な研究が行なわれている分野である．

本研究で提案する帰納推論手法 WINE は，従来の手法では考慮されていなかったルールの適用順序を，ルールを生成する段階で固定してしまう．これによって後の方で評価されるルールに，一部の負事例も説明するようなラフなルールが利用できるようになる．その結果，従来の方法よりも効率の良い帰納推論を行ない，しかも簡潔なルールを得られる場合があることを示す．

2　ルールの必要条件と評価順序

通常の帰納推論システムでは，正の事例を説明し，負の事例は許容しないようなルールを作成することが目的となる．作成されたルールはすべてこの必要条件を満たしているので，必ずしも作成された順番に評価される必要はなく，どのような順序で評価されても構わない．

ところが，評価順序があらかじめ固定されている場合には，後の方で評価されるルールは，一部の負の事例も説明してしまうような，ラフなルールでも構わなくなる．つまり，ある質問に対して正しい答を返すルールが先に評価される場合には，その質問に対して誤った答えを返すルールが，その後に並んでいても構わない．なぜなら，先に並べられたルールを評価する時点で解答が終了してしまい，後のルールまで評価が行なわれないからである．

このことについて，図1のような，「鳥は飛ぶ，ペンギンは飛ばない」という，非単調推論でよく用いられる例で考えてみよう．従来の，ルールの評価順序を固定化しない場合，すなわち，すべてのルールが，負の例を含まないことが要請される場合，帰納推論を用いると，次のようなルールが得られるだろう．

$$fly(X,Y) \ :- \ pigeon(X), yes(Y).$$
$$fly(X,Y) \ :- \ swallow(X), yes(Y).$$
$$fly(X,Y) \ :- \ eagle(X), yes(Y).$$
$$fly(X,Y) \ :- \ penguin(X), no(Y).$$

しかし，ルールを Prolog のように，上から順に評価するこ

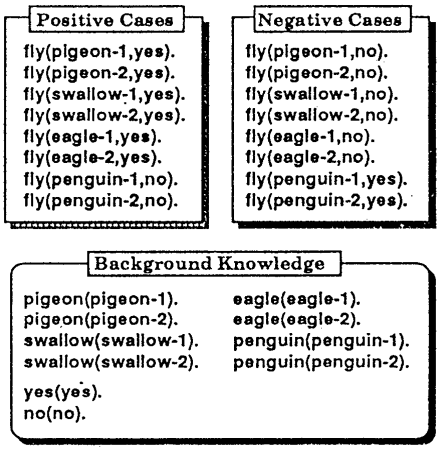

図 1:「Penguin の例題」

とにすれば，次のような簡潔な表現でも解答になっている．

$$fly(X,Y) \ :- \ penguin(X), no(Y). \quad \cdots \ (1)$$
$$fly(X,Y) \ :- \ yes(Y). \quad \cdots \ (2)$$

ここで，(2) のルールは，負の事例 (fly(penguin-1,yes), fly(penguin-2,yes)) を満たしてしまっている．しかし，ルールの適用順序を (1),(2) の順番に限定すれば，このルール群からは，fly(penguin-1,yes) という答が返ってくることはない．なぜならば，例えば fly(penguin-1,X) (X は変数) という問い合わせがなされた場合，必ず (1) のルールによって fly(penguin-1,no) という解が得られ，(2) のルールが評価されることがないからである．

3　WINE のアルゴリズム

WINE(foil WIth NEgative rules) はこのような，一部の負の事例を許容したラフなルールを生成する帰納推論手法である，WINE は，基本的には FOIL[Qui90, Qui91] のアルゴリズムを用いている．FOIL は有名な帰納推論プログラムの一つで，情報量に基づくヒューリスティクスを用いて，仮説空間をトップダウンに探索してルールを求めていく．FOIL では負の事例を許容しなくなるまで節に述語を追加していくが，WINE では，負の事例が事例集合の一定割合を下回れば，その段階で仮説の選択を打ち切り，そこまでのルールを答として一旦出力する．その後で，その負事例に対応する正事例を

WINE: An Inductive Reasoning Method Which Produces Fixed Order Rules
Takao MOHRI and Hidehiko TANAKA
The University of Tokyo

導くためのルールを作成して，先に評価するような順番にそれを並べるのである．つまり，WINE ではまず一般的な (負事例を含む) ルールが生成され，残された例外的な事例を説明するようなルールがその後で生成される．Penguin の例題では，まず一般的なルール (2) の後に例外を扱うルール (1) が生成され，(1),(2) の順に評価される．

```
main(){
    事例集合を作成する.
    while (事例集合に正事例が残されている){
        ルール群の作成 (事例集合).
        そのルール群で説明される正事例を,
            事例集合から取り除く.
    }
}
ルール群の作成 (事例集合){
    while (not(正事例の割合 ≥ α)){
        ルールに新しいリテラルを加える.
        事例集合をつくりなおす.
    }
    if ((ルールは負事例を許容しない) &&
        (以前許容した負事例に対応する正事例は
            すべて説明した)){
        return <ルール >;
    }else {
        新しい事例集合の作成.
        ルール群=ルール群の作成 (新しい事例集合).
        return <ルール群, ルール >;
    }
}
```

図 2: WINE のアルゴリズム

4 WINE の健全性と完全性

WINE と FOIL の健全性についてまとめると表 1 のようになる．

表 1: FOIL と WINE の健全性

	Query of T/F		Query of Value	
	T as T	F as F	T as T	F as F
FOIL	OK	OK	OK	OK
WINE	OK	*no*	OK	OK

ここで，'T as T' は正事例を質問した時に真だと答を返すことを，'F as F' は負事例を質問した時に偽だと答えることを示している．また，Query of T/F は，真か偽かを答えればいいような質問，Query of Value は，未束縛な変数に値を返す必要がある質問である．

表 1 の *no* は，正しい答を返すはずのルールが，負事例の質問を素通りさせてしまい，ラフなルールが評価されて，負事例に対して真だという答が返ってしまう可能性があることを示している．

また，WINE で Query of Value を行なった場合には，学習に使ったすべての正事例が返されることは保証されるので，その意味で完全性は成り立つといえる．その答の中には負事例として与えたものも混じってしまう可能性はあるが，最初に得られる解が負事例でないことは保証されている．

以上のような性質から，WINE は Query of Value 型の質問で，小数の解しか必要としていない場合に有効であることがわかる．

5 実験結果

帰納推論の例題として良く用いられているリスト処理問題を例にとって，FOIL と WINE の性能比較を行なった．ただし，例題 insert と penguin は，今回の実験のために作成し，quick sort, member, combination は [Qui90] で使われているものをそのまま用いた．

表 2 からは，まず，WINE が起動されない場合，つまり負の事例を許容するようなルールが作成されない場合があることがわかる．これは，事例の構成が，WINE の得意とする多数の一般事例と少数の例外事例の組合せになっていないからである．また，penguin の例題では，従来の半分程の長さの簡潔なルールが，2 倍程度高速に生成できていることがわかる．

表 2: リスト処理問題の学習結果

例題	推論時間* (秒)	ルール数	リテラル数	WINEed
quick sort	93 / 95	2 / 2	10/ 10	なし
member	0.8 / 0.85	2 / 2	6 / 6	なし
insert	48 / 49	3 / 3	15 / 13	3→2
combination	316 / 317	2 / 2	10 / 10	なし
penguin	0.56 / 0.27	4 / 2	12 / 5	2→1

*表中の値は'FOIL での値/WINE での値' を表す

6 結論および今後の課題

本研究提案した帰納推論手法 WINE では，ルールの適用順序が固定化されているため，一部の負の事例を許容するできるようになり，そのため効率の良いルール表現ができる場合がある．実験によって，良い場合には 2 倍程度の推論速度の向上，ルール数，リテラル数の減少が達成できることが示された．しかし，健全性，完全性や質問の方法に関して，従来の方法にいくつかの制約が加わるため，今後はこれらの制約に見合うような WINE の用途を探す必要があるだろう．

謝辞

NTT ソフトウェア研究所の小野諭氏からは，研究全般に渡り多大な御指導を頂きました．また東京工業大学工学部情報工学科の沼尾正行氏からは，数多くのコメントを頂きました．この場をお借りしてお礼を申し上げます．

参考文献

[Qui90] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, Vol. 5, pp. 239–266, 1990.

[Qui91] J.R. Quinlan. Determinate literals in inductive logic programming. In *Proc. of 12th International Joint Conference on Artificial Intelligence*, pp. 746–750, 1991.

[毛利 92] 毛利隆夫田中英彦. ルールの適用順序を考慮した効率的な帰納推論手法. 情報処理学会研究会報告 92-AI-82, pp. 31–40, 1992.