変数を埋め込んだ自由文でホーン節を構成する論理プログラミング環境へのホーン節機械学習機能の付与

中村 圭介

変数を埋め込んだ自由文により,Prolog 等で用いられるホーン節のヘッドやボディを構成できる論理プログラミング環境を開発している [2]. このような特殊な構文的自由度をもつ自由文で構成されるホーン節は, これまでの帰納論理プログラミング [4] による機械学習の有効性が明らかではなく, あくまで人間がコーディングすることを想定してきた。しかし, 小さなアンケート等, 主な選択肢の自由文の間とさらに自由感想欄に使用される自由文の文型や語彙に共通性がある場合には, 変数を埋め込んだ自由文からなるホーン節（仮説や, 確率のある法則）を, 少ないアンケート等のデータから帰納的, 確率的に導出することが可能であると考え, そのような機械学習機能を付与した. そのアルゴリズムと簡単な実験結果等について報告する.

We are developing a logic programming environment [2] wherein head and body of horn clauses (used for Prolog etc.) are composed of variable-embedded-free-strings. Previously we assumed that only human write the codes, because the availability of existing inductive logic programming methods [4] is not clear for the above horn clauses, which are composed of variable-embedded-free-strings and accordingly have special freedom of composition. However, we thought that we can automatically induce such horn clauses (hypotheses and/or probabilistic rules) from probabilistic data of small questionnaires etc. which may have common compositions and/or words in the free-strings of main choices or even in the free-comment-texts of the questionnaire, and so we added a machine learning (from small data) function to the above environment, We report, in this paper, the algorithm and results of some simple experiments.

1 はじめに

地域の製造・福祉・教育等の現場において, その地域の方言やチームだけの合言葉などといった, 標準語ではない”自由文”を人工知能等で処理することによりコミュニケーション、想像、論理的/創造的な思考をより手軽にかつ楽しくできると考え, 変数を埋め込んだ”自由文”によって, ホーン節 (Prolog [1] 等で用いられる) のヘッドやボディを構成できる論理プログラミング環境を開発している [2] [3].

開発を始めた 2011 年からの数年間は, このような特殊な構文的自由度をもつ自由文で構成されるホーン節は, 帰納論理プログラミング [4] 等による機械学習 (ないし自動プログラミング) の有効性が明らかではなく, あくまで人間がコーディングすることを想定し, 年少者から高齢者までの全ての人間がコーディングしやすいことを目標としてきた。

しかし, 小さなアンケート等, 主な選択肢の自由文の間とさらに自由感想欄に使用される自由文の文型や語彙に共通性がある場合には, 変数を埋め込んだ自由文からなるホーン節（仮説や, 確率のある法則）を, 少ないアンケート等のデータから帰納的, 確率的に導出することが可能であると考え, そのような機械学習機能を開発した.

2 では、標準的な Prolog と”自由文による”論理プログラミング [3] との差異, この差異がもたらす, 既存の帰納論理プログラミング [4] 手法適用の問題点にふれ, 3 ではこれを回避するアルゴリズムのアイディア, 4 ではこの実装による簡単な実験結果を示す.

Machine Learning of Horn Clauses composed of Variable-Embedded-Free-Strings for Logic Programming Environment.

Keisuke Nakamura, ナレルシステム株式会社, Knowrel System, Inc..

2 ”自由文”と既存の帰納論理プログラミング

2.1 Prolog の”原子文”と比較した”自由文”の特徴

Prolog において同様に扱われる”原子文”に比して”自由文”は以下の特徴がある.

- **”好き（太郎, 花子)”**のような括弧やコンマで区切って文字列の意味的役割を明示した式表現ではなく,**”太郎は花子が好きに違いない”**のように括弧やコンマで区切られず役割は助詞等で推測しなければならず, 細かい様相も文字で表現される (一文字でも違うなら意味の類似性や関連性が一旦「ないもの」と割り切って、完全同一パターンを利用して遠くまでの推論を簡潔にし, 文脈の泣き別れは等価表現を標準方向に変換するための別の規則等で救うという方針).
- **”好き（X,Y)”**のように大文字アルファベットを変数とするのではなく,**”$X は$Y が好き”**や**”${ 営業担当 } は${ お客さん } が好き”**のように変数が**”$”**という目印の後の大文字アルファベット又は **{}** で区切られた名前によって表現される.

2.2 帰納論理プログラミング適用の問題点

”自由文”の上記特徴は以下の問題点となる.

- Prolog の原子式が各述語の格構造の同定を前提とするのに対し、”自由文”には同定できる格構造や型制約（帰納論理プログラミングにおける”背景知識”）が, 計算機に取扱いやすい形式言語や論理式では存在せず, **文型や語彙の制約に沿った仮説生成が困難**.
- 特に Prolog における最汎単一化代入 (most general unifier, MGU) が,”自由文”では一意に決まらない (例:**”太郎と次郎と三郎は男兄弟”と”$X と$Y は男兄弟”**の MGU は 2 つ).

3 アルゴリズム

3.1 問題点を回避するアイディア

1. 関連するかもしれない複数の事象が同じタイミングで発生したかをチェックする形式のアンケートを用意する
2. チェックの対象となるそれぞれの事象を表現する自由文中の名詞や動詞は同じ意味であれば表現をそろえる
3. 各回のアンケート結果を, チェックした選択肢を表現する文の集合からなる単一の文脈（一つのデータ）とみなす
4. 単一の文脈中の複数の文に跨る共通文字列を最長一致で同名の変数におきかえる (元の共通文字列そのものを変数名とすることもでき, この場合, 下の”変数名の差異を吸収してカウントする”処理はやや複雑になるが, 元の文字列を残すことにより利用時に過汎化防止等の手がかりとすることができる).
5. 複数のアンケート回答で得られた文脈の集合について, 単一の文脈を構成するあらゆる文集合間に,(少ないデータによる) 仮説としての一方向の依存関係（例えばＡかつ B ならば必ず C, しかし,C でも必ずＡかつ B とは限らない）があれば”A かつ B なら C”を発生頻度とともに報告する. 一方, 逆の”C ならＡ”及び”C なら B”については確率オプションがＯＮであれば頻度および確率とともに報告する.
6. 仮説に該当するデータの発生頻度または該当する確率を計算する際に, 変数名の差異を吸収してカウントするモードも用意する. これにより, 異なる場所でそれぞれ起こった複数の自然現象の関連性等をより多くの (同型の) 証拠をもって抽出可能とする.

3.2 アルゴリズムの概要

1. アンケートの回答データセット（各データは一回の回答でチェックされた（多くの場合複数の）自由文の組）を”閉世界”(これ以上データが存在しない)”とみなし, 一つ以上のデータを説明する無矛盾なホーン節 (仮説）をすべて列挙する (回答データが少ない場合は一瞬で終了する).
2. 共通文字列を変数に置き換える
   (a) 共通文字列名を保存するモード:列挙したホーン節ごとに, ヘッドから順についでボ

ディの 1 つ目, 2 つ目 … 最後のボディへと, 2 文字以上で 2 自由文以上にまたがる共通文字列を, 最長のものから順に, 同名変数で (かつ元の共通文字列をその${ 変数名 } として残しながら) 置き換えていく. 一旦変数の名前になってしまった共通文字列はより短い名前の変数名等では置き換えない.

(b) 共通文字列名を捨象するモード:列挙したホーン節ごとに, ヘッドから順についでボディの 1 つ目, 2 つ目 … 最後のボディへと, 2 文字以上で 2 自由文以上にまたがる共通文字列を, 最長のものから順に, 同名変数で (出現した順に$X1,$X2,... と) 置き換えていく.

3. 共通文字列を同名変数 (モード 1 と 2 で異なる) に置き換えた結果同一の意味となったホーン節については一つのホーン節に統合する.
4. 各ホーン節について, ボディ部が該当した頻度とヘッド部が該当した頻度からホーン節が正しい確率を求める.

4 実験

corei7(3.6-4.5GHz), 8GB メモリ,Windows10 環境において VisualC++2013 で実装した結果を示す.

4.1 実験 1:共通文字列を変数名として残すモード

以下は, 図 1 のアンケートにより, 5 回 (日) 分のサンプルデータを作成したものである.

```
金沢は雨がふった
小松は雨がふった
金沢はすずしくなった
---
小松は雨がふった
金沢はすずしくなった
小松はすずしくなった
---
金沢はすずしくなった
小松はすずしくなった
---
金沢はすずしくなった
小松はすずしくなった
---
金沢は雨がふった
金沢はすずしくなった
小松はすずしくなった
---
```

上記アルゴリズムの概要 1 . に従って, 上のいずれかのデータを 0.5 超の確率で説明しうるホーン節を列挙すると以下のようになる.

```
4/4 金沢はすずしくなった
        :- 小松はすずしくなった ;

2/2 金沢はすずしくなった
        :- 金沢は雨がふった ;

2/2 金沢はすずしくなった
        :- 小松は雨がふった ;

1/1 金沢はすずしくなった
        :- 金沢は雨がふった ;
           小松は雨がふった ;

1/1 金沢はすずしくなった
        :- 金沢は雨がふった ;
           小松はすずしくなった ;

1/1 金沢はすずしくなった
        :- 小松は雨がふった ;
           小松はすずしくなった ;

4/5 小松はすずしくなった
        :- 金沢はすずしくなった ;
```

次に, アルゴリズムの概要 2.(a) に従い, 共通文字列を同名の元の文字列を名前とする変数に置き換えると以下のホーン節集合 (頻度と確率付) が得られる.

```
4/4 金沢${はすずしくなった}
        :- 小松${はすずしくなった} ;

2/2 ${金沢は}すずしくな${った}
        :- ${金沢は}雨がふ${った} ;

2/2 金沢はすずしくな${った}
        :- 小松は雨がふ${った} ;

1/1 ${金沢は}すずしくな${った}
        :- ${金沢は}雨がふ${った} ;
           小松は雨がふ${った} ;

1/1 金沢${はすずしくなった}
        :- 小松は雨がふった ;
             小松${はすずしくなった} ;

1/1 ${金沢は}${すずしくなった}
        :- ${金沢は}雨がふった ;
           小松は${すずしくなった} ;

4/5 小松${はすずしくなった}
        :- 金沢${はすずしくなった} ;
```

4.2 実験2:共通文字列を変数名に残さないモード

上記アルゴリズムの概要 2.(b) に従って共通文字列を同名の変数 ($X1,$X2...) などに置き換えると以下のホーン節集合（頻度と確率付）が得られる.

```
1/2 金沢${X1}
        :- 小松${X1} ;

2/5 ${X1}雨がふ${X2}
        :- ${X1}すずしくな${X2} ;

1/2 ${X1}${X2}
        :- 小松は${X2} ;
           ${X1}すずしくなった ;

1/2 小松${X1}
        :- 金沢${X1} ;

2/5 小松は雨がふ${X1}
        :- 金沢はすずしくな${X1} ;

1/2 小松${X1}
        :- 金沢${X1} ;
           金沢はすずしくなった ;

2/2 ${X1}すずしくな${X2}
        :- ${X1}雨がふ${X2} ;

2/2 金沢はすずしくな${X1}
        :- 小松は雨がふ${X1} ;

1/1 ${X1}すずしくな${X2}
        :- ${X1}雨がふ${X2} ;
           小松は雨がふ${X2} ;

1/4 ${X1}雨がふ${X2}
        :- ${X1}すずしくな${X2} ;

1/4 ${X1}雨がふ${X2}
        :- 金沢はすずしくな${X2} ;
           ${X1}すずしくな${X2} ;

4/4 金沢${X1}
        :- 小松${X1} ;

1/1 金沢${X1}
        :- 小松は雨がふった ;
           小松${X1} ;

1/2 ${X1}すずしくな${X2}
        :- ${X1}雨がふ${X2} ;

4/5 小松${X1}
        :- 金沢${X1} ;

1/2 ${X1}${X2}
        :- ${X1}雨がふった ;
           金沢は${X2} ;

1/4 金沢は雨がふ${X1}
        :- 小松はすずしくな${X1} ;

1/4 ${X1}雨がふ${X2}
        :- ${X1}すずしくな${X2} ;
           小松はすずしくな${X2} ;

1/1 ${X1}${X2}
        :- ${X1}雨がふった ;
           小松は${X2} ;

1/2 小松はすずしくな${X1}
        :- 金沢は雨がふ${X1} ;

1/2 小松${X1}
        :- 金沢は雨がふった ;
           金沢${X1} ;
```

次に, 上記アルゴリズムの概要 3 および 4 に従ってホーン節と頻度と確率を統合した上で確率 0.5 超のホーン節を抽出すると以下が得られる.

```
2/2 金沢はすずしくな${X1}
        :- 小松は雨がふ${X1} ;

1/1 金沢${X1}
        :- 小松は雨がふった ;
           小松${X1} ;

1/1 ${X1}すずしくな${X2}
        :- ${X1}雨がふ${X2} ;
           小松は雨がふ${X2} ;

1/1 ${X1}${X2}
        :- ${X1}雨がふった ;
           小松は${X2} ;

5/6 金沢${X1}
        :- 小松${X1} ;

3/4 ${X1}すずしくな${X2}
        :- ${X1}雨がふ${X2} ;

5/7 小松${X1}
        :- 金沢${X1} ;
```

閉世界仮説の通用しない対象に適用する場合には, この状態から, さらに頻度が低いもの (1/1 等）はたとえ確率が高くても取り除くべきと考える.

5 まとめと今後の課題

文型や引数の型が Prolog の原子文のように決まっていない”自由文”の組の集合をデータセットとしてホーン節（但し, 変数を含みうる”自由文”からヘッドやボディのリテラルが構成される) の機械学習が可能と考えられる例について示した.

それは, 一つのデータ（一回分のアンケート回答）を構成する”自由文”の組が, 互いに関連性をもつアンケートの選択肢として共通文字列をもつように記述されている場合である.

アンケートがこのような性格をもつものになるようアンケート作成者を指導したり人工知能処理系とヒトとの対話をこのような性格をもつものになるよう自動構成したりすることによりこのようなアプローチが実用性をもつと考える.

今後は、以下の課題に取り組んでいく予定である.

1. オントロジーに依存しないホーン節等の知識データ書式の標準化,
2. そのような知識データ共有サイトの構築
3. 英語や中国語での機械学習実験
4. 学習前から存在するルールとの関係性の整理
5. 否定形のリテラル (条件を構成するもの) の学習
6. 不適切なストップワード（「った」等）を変数としないモード等の付加
7. 人工知能による対話式アンケートと関連づけた本手法の実証実験（図 2).

参 考 文 献

[1] ISO/IEC13211-1:1995(E): Information technology - Programming languages - Prolog -, 1995.

[2] N., K.: Method for Processing Knowledge or Information, Device, and Computer Program, *WIPO WO/2016/071942*, (2016).

[3] 中村圭介: 自由文による思考プログラミング, 情報処理学会第 *60* 回プログラミングシンポジウム, 2018.

[4] 古川康一, 尾崎知伸, 植野研: 帰納論理プログラミング, 共立出版株式会社, 2001, chapter 第 8 章帰納論理プログラミング.

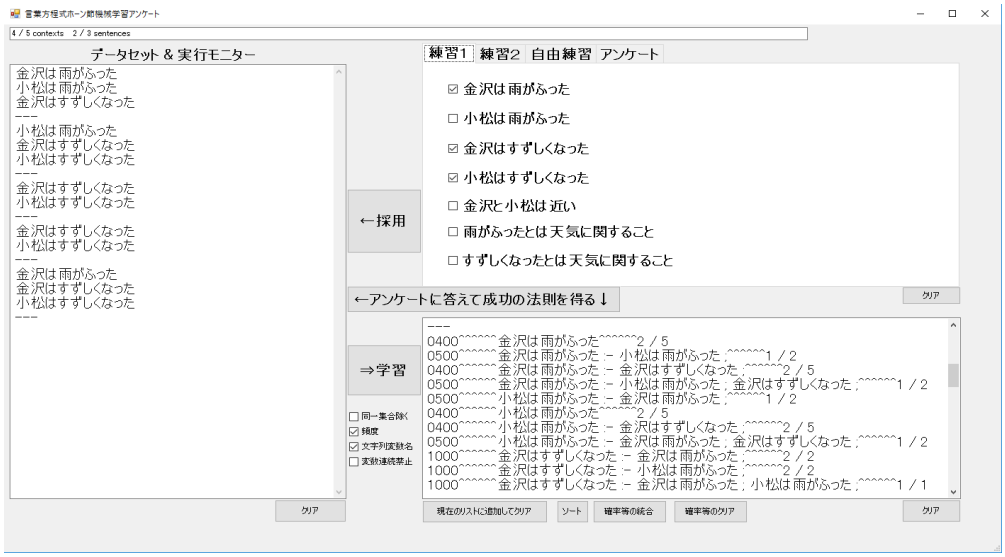

図 1　実装した UI

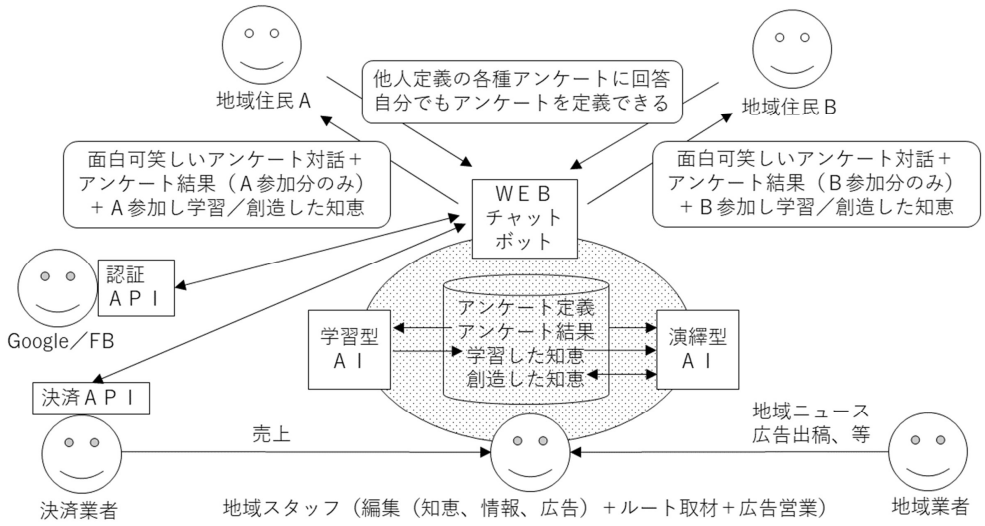

図 2　実証実験の概要