論理和を含む概念の学習アルゴリズムとその応用

3E－1

藤波　努，　辻　洋
(株) 日立製作所　システム開発研究所

1．はじめに

例から一般的な知識を発見する方式として，「例」と「正負の判断」，例と概念の記述に用いる語彙の「汎化階層」を入力すると，正例を含み負例を含まない概念を学習するヴァージョン空間法[1]が研究されている．

本論では，ヴァージョン空間法で論理和を含む概念を学習する場合の問題点を示し，それらを解決するアルゴリズムと適用例を述べる．

2．問題の記述

2．1　ヴァージョン空間法

ヴァージョン空間法は，例から帰納される目標概念の仮説Hを，Hの最も特殊な要素の集合Sと，最も一般的な要素の集合Gに挟まれた空間として管理する．論理和を含まない概念の学習は，逐次例を入力し，SとGが単一要素となって一致する（収束）まで，候補消去アルゴリズムによりSとGを更新することにより行われる．

この候補消去アルゴリズムは，正例に適用される「S集合更新アルゴリズム」と，負例に適用される「G集合更新アルゴリズム」から成る．

（1）S集合更新アルゴリズム
①正例をカバーしない仮説をGから取り除く
②正例を含むようにSを一般化する

（2）G集合更新アルゴリズム
①負例をカバーする仮説をSから取り除く
②負例を含まないようにGを特殊化する

2．2　論理和表現の学習

ヴァージョン空間法で論理和を含む概念を学習する方法として，次のアルゴリズムが知られている[2]．

Step１　１つの正例によりSを初期化し，Gは空の記述に初期化して，ヴァージョン空間を生成する．
Step２　全ての負例により，GにG集合更新アルゴリズムを適用する．
Step３　Gから要素G_iを取り出し，論理和表現を構成する記述の１つとする．G_iにカバーされる正例は考慮済みとし，学習の対象から外す．
Step４　すべての正例がカバーされるまで，Step１からStep３を繰り返す．

2．3　問題点

従来のアルゴリズムは，あらかじめ十分な正例と負例が与えられていることを前提としている．Step１では，十分な正例が存在するので，論理和を構成する全ての記述について，逐次対応するヴァージョン空間を初期化することができる．また，Step２では，十分な負例が存在するので，Gは過剰一般化のないように特殊化できる．

しかしながら，学習機能を実用システムに適用する場合，記録されているのは正例であり，十分な負例を集めるのは難しいことが多い．従って，この前提のない学習方式も必要である．

3．提案方式

3．1　例生成方式

本方式では，あらかじめ十分な例が与えられていない場合でも，例を生成して正負をユーザに質問することにより，正例と負例を得ることができる．

Disjunctive Concept Learning Algorithm and its Application,　Tsutomu FUJINAMI, Hiroshi TSUJI,
Systems Development Laboratory, Hitachi, Ltd.

論理和を構成する各ヴァージョン空間のSに，S集合更新アルゴリズムを適用することにより仮説Hを管理し，Hにカバーされる例の１つを利用する．

しかし，単にHにカバーされるという条件だけで例を生成すると，ヴァージョン空間が論理和を構成する複数の記述空間をカバーして収束する過剰一般化が起こり，記述空間を分離する負例が生成されない場合がある．（図１）

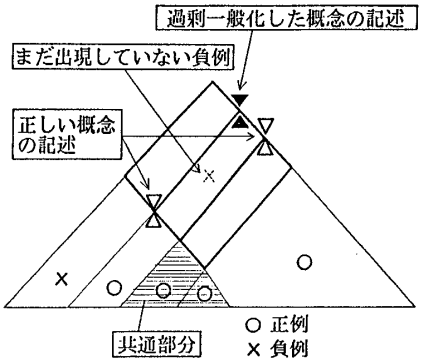

図１　過剰一般化のヴァージョングラフ

この問題を回避するために，本方式では，サブ空間という考えを導入する．ヴァージョン空間が収束した記述に，特殊化を一回施した記述の集合をG'とする．G'から要素G'_iを取り出し，サブ空間SUB_iのG集合とする．次に，G'_iにカバーされる正例でSを初期化し，S集合更新アルゴリズムによりSを更新する．サブ空間SUB_iは，SとGに挟まれた空間である．

サブ空間にカバーされる例を生成することにより，論理和を構成する記述空間を分離する負例の候補を生成し，過剰一般化されたヴァージョン空間の生成を避けることができる．

3．2　アルゴリズム

Step１　次の優先順序で例を生成する．
①収束していないサブ空間にカバーされ，かつ他の収束しているサブ空間にカバーされない例を生成する．
②収束していないサブ空間にカバーされる例を生成する．
③収束していないヴァージョン空間にカバーされる例を生成する．

Step２　Step１で生成された例が
・ヴァージョン空間のGにカバーされない正例ならば，Sを初期化し，これまでに得られた全ての負例により，GにG集合更新アルゴリズムを適用する．
・サブ空間を持つヴァージョン空間のSにカバーされる負例ならば，ヴァージョン空間を破棄してサブ空間をヴァージョン空間とする．
・サブ空間を持たないヴァージョン空間のSにカバーされる負例ならば，負例を含まないようSを特殊化し，カバーされなくなった正例を未考慮例とする．
・ヴァージョン空間かサブ空間にカバーされる例ならば，候補消去アルゴリズムを適用する．

Step３　収束したヴァージョン空間のサブ空間を生成

する．

Step4　・全ての正例がカバーされ
　　・全てのヴァージョン空間が収束し
　　・全てのサブ空間が収束する
　　までStep1からStep4を繰り返す．

4．適用例

4．1　目標概念

提案方式をコンピュータ教育の時間割りを作成するスケジューリング・システム[3]に適用し，科目の前後関係に関する制約を学習させた．

時間割りの対象となる科目を内容にしたがって分類し，例と概念の記述に用いる語彙の汎化階層とする（図2）．

目標概念は「『言語講座の前に情報処理システム入門を履修する』または『COBOL言語の前にはどの科目を履修してもよい』」である．

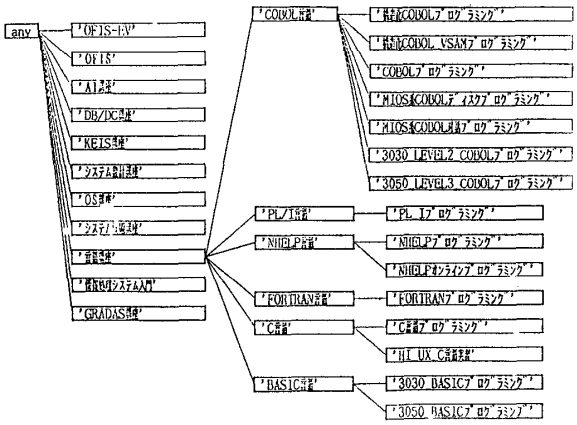

図2　科目の分類

以下，図3，4を用いて学習過程を説明する．図中，例と概念は語彙のリストで表現し，語彙は'word(タイプ，語彙，属性)'として表現している．例えば『COBOL言語の前にはどの科目を履修してもよい』は，'[word(given,COBOL言語,後)，word(given,any,前)]'と表現する．また，あらかじめ与える例は，最初の正例のみであり，他の例は全てプログラムが生成したものである．なお，紙面の都合で一部を削除している．

（1）ヴァージョン空間の収束（図3）

5番目の例で，ヴァージョン空間(メイン1番)は，「言語講座の前に履修しなければならない科目はない」という意味の'[word(given,言語講座,後)，word(given,any,前)]'に収束する．そこで，サブ空間（サブ2番～10番）を作成する．

（2）過剰一般化の発見（図4）

10番目の例で，収束した記述は過剰一般化していることが発見される．元のヴァージョン空間は破棄し，各サブ空間を新たなヴァージョン空間とする．このとき，収束しているヴァージョン空間には，サブ空間を生成する．

これ以降，24番目の例で'[word(given,COBOL言語,後)，word(given,any,前)]'（メイン7番）が確定し，同様にして，'[word(given,言語講座,後)，word(given,情報処理システム入門,前)]'が確定する．

5．まとめ

あらかじめ十分な負例が与えられない場合を想定して，論理和表現を含んだ概念の学習アルゴリズムを提案した．提案したアルゴリズムは，学習に必要な例を生成し，ユーザに正負を質問することにより，正例と負例を得ることができる．また，過剰一般化を回避することができる．

参考文献

1）Mitchell他：実験による学習－問題解決ヒューリスティックスの獲得と改善，学習と問題解決，pp29～55，共立出版(1987)

2）Feigenbaum編：ヴァージョン空間，人工知能ハンドブック，pp517～522，共立出版(1984)

3）坂尾　秀樹：学習機能付割当計画ＥＳの開発(予)，第40回情報処理学会全国大会(1989)

```
** 次の事例について正負を教えてください
質問 :
[word(given,構造化COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,構造化COBOL_VSAMﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,前)]
** 正例->1　負例-> 0　不明-> 2を入力してください
|: 1.
================================================
5 : 正 :
[word(given,構造化COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,構造化COBOL_VSAMﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,前)]
☆☆☆　メイン　－－ 1 番－－　○
★★★ 条件 = [2,3,4,5,6,8,10]
G[word(given,言語講座,後),word(given,any,前)]
S[word(given,言語講座,後),word(given,any,前)]
    ☆☆☆　サブ　－－ 2 番－－　×
    G[word(given,構造化ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,any,前)]
    S[word(given,構造化COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,any,前)]
    ☆☆☆　サブ　－－ 3 番－－　×
    G[word(given,PL/I言語,後),word(given,any,前)]
    S[word(given,PL_Iﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,情報処理ｼｽﾃﾑ入門,前)]
    ☆☆☆　サブ　－－ 4 番－－　×
    G[word(given,NHELP言語,後),word(given,any,前)]
    S まだインスタンシエイトされていない
    ☆☆☆　サブ　－－ 5 番－－　×
    G[word(given,FORTRAN言語,後),word(given,any,前)]
    S まだインスタンシエイトされていない
    ☆☆☆　サブ　－－ 6 番－－　×
    G[word(given,C言語,後),word(given,any,前)]
    S まだインスタンシエイトされていない
    ☆☆☆　サブ　－－ 7 番－－　○
    G[word(given,COBOL言語,後),word(given,any,前)]
    S[word(given,COBOL言語,後),word(given,any,前)]
    ☆☆☆　サブ　－－ 8 番－－　×
    G[word(given,BASIC言語,後),word(given,any,前)]
    S まだインスタンシエイトされていない
    ☆☆☆　サブ　－－ 9 番－－　○
    G[word(given,言語講座,後),word(given,情報処理ｼｽﾃﾑ入門,前)]
    S[word(given,言語講座,後),word(given,情報処理ｼｽﾃﾑ入門,前)]
    ☆☆☆　サブ　－－ 10 番－－　×
    G[word(given,言語講座,後),word(given,言語講座,前)]
    S[word(given,構造化COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),
     word(given,構造化COBOL_VSAMﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,前)]
```

図3　ヴァージョン空間の収束

```
** 次の事例について正負を教えてください
質問 : [word(given,PL_Iﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,構造化COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,前)]
** 正例->1　負例-> 0　不明-> 2を入力してください
|: 0.
================================================
>>入力事例
10 : 負 :
[word(given,PL_Iﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,構造化COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,前)]
☆☆☆　メイン　－－ 1 番－－　×
☆☆☆　メイン　－－ 2 番－－　×
G[word(given,構造化ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,any,前)]
S[word(given,構造化COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,any,前)]
☆☆☆　メイン　－－ 3 番－－　×
G[word(given,PL/I言語,後),word(given,情報処理ｼｽﾃﾑ入門,前)]
S[word(given,PL_Iﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,情報処理ｼｽﾃﾑ入門,前)]
☆☆☆　メイン　－－ 4 番－－　×
G[word(given,NHELP言語,後),word(given,any,前)]
S[word(given,NHELPﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,情報処理ｼｽﾃﾑ入門,前)]
☆☆☆　メイン　－－ 5 番－－　×
G[word(given,FORTRAN言語,後),word(given,any,前)]
S[word(given,FORTRANﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,情報処理ｼｽﾃﾑ入門,前)]
☆☆☆　メイン　－－ 6 番－－　×
G[word(given,C言語,後),word(given,any,前)]
S[word(given,C言語ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,情報処理ｼｽﾃﾑ入門,前)]
☆☆☆　メイン　－－ 7 番－－　○
★★★ 条件 = [19,20,21,22,23,24,25]
G[word(given,COBOL言語,後),word(given,any,前)]
S[word(given,COBOL言語,後),word(given,any,前)]
    ☆☆☆　サブ　－－ 18 番－－　○
    G[word(given,構造化COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,any,前)]
    S[word(given,構造化COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,any,前)]
    ☆☆☆　サブ　－－ 19 番－－　×
    G[word(given,構造化COBOL_VSAMﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,any,前)]
    S まだインスタンシエイトされていない
    ☆☆☆　サブ　－－ 20 番－－　×
    G[word(given,MIOS系COBOLﾃﾞｨｽｸﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,any,前)]
    S まだインスタンシエイトされていない
    ☆☆☆　サブ　－－ 21 番－－　×
    G[word(given,MIOS系COBOL対話ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,any,前)]
    S まだインスタンシエイトされていない
    ☆☆☆　サブ　－－ 22 番－－　×
    G[word(given,COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,any,前)]
    S[word(given,COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,情報処理ｼｽﾃﾑ入門,前)]
    ☆☆☆　サブ　－－ 23 番－－　×
    G[word(given,2050_LEVEL2_COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,any,前)]
    S まだインスタンシエイトされていない
    ☆☆☆　サブ　－－ 24 番－－　×
    G[word(given,2020_LEVEL2_COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ,後),word(given,any,前)]
    S まだインスタンシエイトされていない
    ☆☆☆　サブ　－－ 16 番－－　○
```

図4　過剰一般化の発見