知能と複雑系　132－4
(2003. 3. 13)

特殊化による仮説の融合を用いた分散型知識発見環境の構築

菊池　敏幸†　　山本　章博†

† 北海道大学 工学研究科・知識メディアラボラトリー　〒060-8628 北海道札幌市北区北13条西8丁目
E-mail: †{kikuchi,yamamoto}@meme.hokudai.ac.jp

あらまし　本研究の目標は, 機械学習や知識発見の分野においてこれまで開発された様々な知識発見システムを利用して, 分散された位置に蓄積された大量の情報から有意義な知識を引き出せるようにするためのソフトウェア環境を構築することである. 本稿では, まず分割されたデータから個別に規則を求め, そして得られた規則を特殊化によって一つに融合する手法を提案し, その手法を用いた分散型知識発見環境 Distributed EVLD(Distributed Environment for Various Logics of Discovery) の構築について述べる.

キーワード　帰納論理プログラミング, 分散処理, 特殊化

Distributed Knowledge Discovery based on Merging Hypotheses

Toshiyuki KIKUCHI† and Akihiro YAMAMOTO†

† MemeMedia Laboratory, Hokkaido University　N 13 W 8, Sapporo 060-8628 JAPAN
E-mail: †{kikuchi,yamamoto}@meme.hokudai.ac.jp

Abstract　In this paper we propose a method for knowledge discovery from distributed data on distributed computers. Each of the computers is assumed to have a knowledge discovery system which searches hypotheses from general ones to specific ones. We also assume that each hypothesis can be represented as a logic program. Then the discovery method is to correct hypotheses generated by the computers and then merge them. For the merge of hypotheses we use the most general specialization of clauses. We also validate each clause in the merged hypotheses. We implement our method on the EVLD system, which we developed previously, and call it Distributed EVLD (Distributed EVLD). In the paper we report how our method works well with a well-known data set.

Key words　inductive logic programming, distributed processing, specialization

1. はじめに

本研究の目標は, 機械学習や知識発見の分野においてこれまで開発された様々な知識発見システムを利用して, 分散された位置に蓄積された大量の情報から有意義な知識を引き出せるようにするためのソフトウェア環境を構築することである. そのプロトタイプとして本稿では, Distributed EVLD(Distributed Environment for Various Logics of Discovery) を構築し, 知識を引き出したい情報がすべて同じ位置に存在していなくても知識を引き出すことが可能であることを確かめる.

従来, 知識発見のような大規模または複雑な解析は計算サーバーを用いていた. 一方, 近年パソコンやワークステーションの発展と普及に伴い, 高性能のプロセッサが安価に入手可能となり, これを複数台高速のネットワークで結合して, 計算を行わせる分散処理技術に注目が集まっている. 従って知識発見プロセスも分散化する必要が生じている.

さらに分散が必要なのは, データ収集である. 例えば全国に支店の存在するような店の POS データは支店ごとにそれぞれで収集が行われる.

それらのデータからデータマイニングを行う場合, 発見プロセス, 特にトップダウン型の発見プロセスでは与えられたすべてのデータを用いたある基準に従って適切な仮説を生成していくため, 発見プロセス中はすべてのデータを読みとれる状態にしておかなくてはいけない. そのため, 一般的には一度本店でデータを集計してから仮説を生成する. この場合データを転送する際ネットワークの帯域がボトルネックとなり, またネットワーク上をデータそのものが流れてしまうためセキュリティ上にも問題が生じる.

そこで本稿では, 発見プロセスでの計算を並列に処理さるのではなく, データを分割して並列に計算させ, そして得られた仮説を融合することで最終的に一つの仮説を得る手法を提案する. この手法を用いることで既存の機械学習・知識発見システムをそのまま分散処理に適用させることが可能となり, データの分割および発見プロセス計算の並列化に関しての問題も解決

される．このような手法を実現するためには，融合する仮説の表現方法が重要となるが，本研究では論理プログラムで表現されている仮説を対象とする．論理プログラムは仮説を表現することが可能な上，演繹などの演算を行うことができることが知られている．本稿では論理プログラムに対するこのような演算を用いることで仮説の融合を可能であることを示す．

本稿は次のように構成されている．次章では，基礎となる知識発見支援環境 EVLD の概要および EVLD を構築する際の基本概念となる発見の論理について述べる．3 章では，異なる事実の集合から得られた仮説を融合する手法について記述する．4 章では，仮説の融合を用いた分散型知識発見環境について説明する．5 章では，構築した環境を用いて実験した結果を示す．6 章では，まとめと今後の課題について述べる．

2. 知識発見支援環境 EVLD

2.1 支援環境の概要

本研究では分散知識発見を行う環境として知識発見支援環境 EVLD(Environment for Various Logics of Discovery) [6] を利用する．EVLD は，従来個別に開発されていた知識発見システムを，下で述べる**発見の論理** [10] の枠組みで捉える環境である．

入力として関係データベースから知識発見の対象となるデータおよび仮説を生成させる知識発見システムを選択する．EVLD は入力された情報を基に SQL 文を生成し，関係データベースからデータを引き出す．引き出されたデータ及び入力された補足情報を用いて選択された各知識発見システムへデータを変換して受け渡す．その際データは各システムに適切な入力フォーマットに変換される．そして知識発見システムを実行し，得られた仮説を論理プログラムに変換して出力する．現状で選択可能な知識発見システムは C4.5 [11], Progol [8], FOIL [12] である．なお，本稿で取り扱うデータは離散値だけである．また，EVLD は仮説を論理プログラムで表現しているため，EVLD を利用することで得られた仮説を演繹などに利用するアプリケーションを構築することが可能である [5].

2.2 論理的基礎

EVLD は，学習・発見システムにデータベース理論のような 3 層スキーマ構造を導入している．3 層スキーマ構造の中間層である概念スキーマ (論理構造) として発見の論理の枠組みを採用し，その下位層 (内部スキーマ, 物理構造) として個々の知識発見システムとその入力データを保持した関係データベースを位置付けている．発見の論理の上位層には，論理プログラムを利用した様々な知識処理システムが位置する．以下では，オリジナルの発見の論理を若干変更して，論理プログラムの概念を用いて記述する．

仮説生成の対象となる**事実**を表す基礎アトムの集合を $F = \{f_1, f_2, \ldots, f_n\}$ とする．発見システムが仮説を生成する際に補助的に用いる基礎アトムは，連言の集合 $V = \{v_1, v_2, \ldots, v_n\}$ という形で用いる．集合 V 中の各連言 v_i は F 中のアトム f_i と強い連関があると考えられる**証拠** (evidence) を表す．発見システムが保持している論理プログラム B はすべての $f_1, f_2, \ldots, f_n$ と $v_1, v_2, \ldots, v_n$ に共通の**背景知識**となる．F, V, B の間には $B \cup \{v_i\} \not\vdash f_i$ $(i = 1, 2, \ldots, n)$ が成立しているものとする．

[定義 1] 組 (F, V, B) にして論理プログラム H が $B \cup H \cup \{v_i\} \vdash f_i$ $(i = 1, 2, \ldots, n)$ を満たすとき，H は (F, V, B) に対する**正しい仮説**であるという．また，H を**仮説発見問題** $HFP(F, V, B)$ に対する**解**ともいう．$HFP(F, V, B)$ の解全体の集合は有限とは限らない．$HFP(F, V, B)$ の自明な解 $H_{F,V} = \{f_1 \leftarrow v_1, \ldots, f_n \leftarrow v_n\}$ を**初期仮説**とよぶ．

[定義 2] 論理プログラム H_1, H_2, B に対して，$B \cup H_1 \vdash H_2$ が成立するとき，H_1 は H_2 より B **に関して相対的に一般的である**という．

発見の論理の枠組みにおける**知識発見システム**とは HFP (F, V, B) に対する解で初期仮説 $H_{F,V}$ より B に関して相対的に一般的な仮説 H を (何らかの基準で選択しながら) 生成するシステムのことである．

知識発見システムを発見の論理の下位層に位置付けるために，EVLD は以下の機能を利用者に提供する．

（1） データベースからタプル集合を得るための SQL 文と発見の論理における述語記号との対応を与える機能．

（2） 発見の論理における事実を表す述語記号と証拠を表す述語記号を定義する機能．

また，個々の知識発見システムに対しては，これらの機能によって定義された述語記号および事実と証拠の指定によって，データベースから得られるタプル集合をそのシステムに応じた入力フォーマットに変換する変換系を用意している．さらに，そのシステムが論理プログラム以外の形で仮説を生成する場合には，その仮説を論理プログラムに変換するための変換系も用意している．

3. 異なる事実の集合から得られた仮説の融合

3.1 本研究の対象となる知識発見システム

現在までに，様々な知識発見システムが研究・開発されているが，これらを仮説を探索する方向 (探索順序) という観点から見ると，大きくトップダウン探索とボトムアップ探索の二つに分類できる [4].

トップダウン探索とは，最も一般的な仮説から探索を始め，順番により特殊な仮説を生成しながら探索を進めるという探索方式である．本論文では，トップダウン探索の知識発見システムで得られた仮説の融合について議論する．一般的にトップダウン探索の方が効率がよいため，データベースからの知識発見によく利用されている．実際 EVLD で利用可能な C4.5, Progol, FOIL はすべてトップダウン探索である．

また，本研究では仮説発見問題のうち**分類問題** (classification problem) について考える．分類問題では，物事を分類するために分類カテゴリが与えられている必要がある．以降，分類カテゴリに**一貫性制約** (integrity constraint) を導入する．すなわち，“二つの分類カテゴリに同時に分類されない” という制約条件が成り立っている上で話を進める．

3.2 共通特殊化

異なる事実および証拠の組の集合から生成された仮説の融合について考える．

ID	$class$	$size$	$color$	$animal$
a	$dangerous$	$small$	$black$	$bear$
b	$dangerous$	$medium$	$black$	$bear$
c	$dangerous$	$large$	$black$	dog
d	$safe$	$small$	$black$	cat
e	$safe$	$medium$	$black$	$horse$
f	$dangerous$	$large$	$black$	$horse$
g	$dangerous$	$large$	$brown$	$horse$

図 1　動物に関する関係 R_{animal}

ID	$class$	$size$	$color$	$animal$
a	$dangerous$	$small$	$black$	$bear$
b	$dangerous$	$medium$	$black$	$bear$
c	$dangerous$	$large$	$black$	dog

R_1

ID	$class$	$size$	$color$	$animal$
d	$safe$	$small$	$black$	cat
e	$safe$	$medium$	$black$	$horse$
f	$dangerous$	$large$	$black$	$horse$
g	$dangerous$	$large$	$brown$	$horse$

R_2

図 2　二つに分割した動物に関する関係

［定義 3］ 事実 f および事実に対する証拠 v が与えられたとき，$f \leftarrow v$ で表される節を**例**という．例は $F = \{f\}$, $V = \{v\}$ としたときの仮説発見問題の自明な解の一つである．

あいまいになるのを防ぐため，今後仮説を生成するために初めに与えられた事実と証拠の組の集合を**例空間**と呼ぶことにする．また事実の集合 F，証拠の集合 V のとき，この例空間を $E = (F, V)$ と書く．

［定義 4］ ある例空間 E に含まれる例 $f \leftarrow v$ が仮説 H によって**被覆** (cover) されるとは，その例が仮説 H によって説明されることを意味する．

仮説 H とそれに関する背景知識 B によって説明できる例の集合は例空間 E の部分集合であり，それを $cvr(E, H, B)$ と書く．

ここでは最も一般的な仮説から出発して，求める仮説に至るトップダウン探索手続きを考える．この操作に必要となるのは，負事例を含まないように仮説を除々に弱める，あるいは狭める方法である．仮説の最大共通特殊化を用いて例空間の制限を取り除く手法を提案する．その前に特殊化について定義する．

［定義 5］ 仮説を弱める操作を仮説の**精密化** (refinement) または**特殊化** (specialization) と言う．仮説の特殊化を行うためには，仮説間に一般性に関する半順序がなくてはならない．仮説が確定節集合であるとき，その中の一つの節を特殊化するための半順序関係として，**包摂関係**が知られている．

［定義 6］ 節 C が節 D を**包摂**するとは，ある代入 θ が存在し，$C\theta$ 中のリテラルがすべて $D\theta$ 中に出現することをいい，$C \succeq D$ と書く．節 D から $C \succeq D$ となる節 C を求めることを D を**包摂に基づいて汎化する**という．逆に節 C から $C \succeq D$ となる節 D を求めることを C を**包摂に基づいて特殊化する**という．

［定義 7］ 節 C, D に対して，$C \succeq H$ かつ $D \succeq H$ となる節 H を C と D の**包摂に基づく共通特殊化**という．

特に，C と D の包摂に基づく特殊化の中では，半順序関係 $\succeq$ のもとで最大となる節 H が必ず存在する [13]．そのような H を C と D の**包摂に基づく最大共通特殊化**といい，$mgs(C, D)$ と書く．

まず仮説が単一節集合，すなわち $H = \{C\}$ のときを考える．なお，以降では節 C の頭部を $hd(C)$，節 C の体部を $bd(C)$ と記述する．

［命題 1］ 背景知識 $B = \emptyset$ とする．例空間 E_1 において仮説 $\{C_1\}$ で被覆する集合を $S_1 = cvr(E_1, \{C_1\}, \emptyset)$，例空間 E_2 において仮説 $\{C_2\}$ で被覆する集合 $S_2 = cvr(E_2, \{C_2\}, \emptyset)$ とする．$hd(C_1) = hd(C_2)$, $S = S_1 \cup S_2$, $E = E_1 \cup E_2$ であり，$E_1 \cap E_2 = \emptyset$ であるならば $E \supseteq S \supseteq cvr(E, \{mgs(C_1, C_2)\}, \emptyset)$ である．

[命題 1 の証明] 確定節 C_1, C_2 をリテラル L_0 とリテラルの列 Φ_1, Φ_2, Φ_3 を用いて次のように定義する．ただし，Φ_1, Φ_2, Φ_3 の間で述語を共有していないとする．

$$C_1 = \{L_0 \leftarrow \Phi_1, \Phi_2\},$$
$$C_2 = \{L_0 \leftarrow \Phi_1, \Phi_3\}.$$

このとき最大共通特殊化節は

$$mgs(C_1, C_2) = \{L_0 \leftarrow \Phi_1, \Phi_2, \Phi_3\}$$

である．E の中で $\{L_0 \leftarrow \Phi_1, \Phi_2\}$ によって被覆される例からなる集合を $S^{E_1}_{\Phi_1,\Phi_2}$, $\{L_0 \leftarrow \Phi_1, \Phi_2, \Phi_3\}$ によって被覆される例からなる集合を $S^{E_1}_{\Phi_1,\Phi_2,\Phi_3}$ とし，E_2 に対しても同様に定義したとき，

$$S^{E_1}_{\Phi_1,\Phi_2} \supseteq S^{E_1}_{\Phi_1,\Phi_2,\Phi_3}$$
$$S^{E_2}_{\Phi_1,\Phi_3} \supseteq S^{E_2}_{\Phi_1,\Phi_2,\Phi_3}$$

が成り立つ．$E_1 \cap E_2 = \emptyset$ であるので

$$S^{E_1}_{\Phi_1,\Phi_2,\Phi_3} \cap S^{E_2}_{\Phi_1,\Phi_2,\Phi_3} = \emptyset$$

である．$H = \{mgs(C_1, C_2)\}$ によって被覆される例からなる集合を $S^{E}_{\Phi_1,\Phi_2,\Phi_3}$ とした場合，

$$S^{E}_{\Phi_1,\Phi_2,\Phi_3} = S^{E_1}_{\Phi_1,\Phi_2,\Phi_3} \cup S^{E_2}_{\Phi_1,\Phi_2,\Phi_3}$$

である．$S = S^{E_1}_{\Phi_1,\Phi_2} \cup S^{E_2}_{\Phi_1,\Phi_3}$ であるから $S \supseteq S^{E}_{\Phi_1,\Phi_2,\Phi_3}$ が成り立つ．

［命題 2］ 異なる例空間 E_1, E_2 から得られた単一節からなる仮説 C_1, C_2 を最大共通特殊化した単一節 $mgs(C_1, C_2)$ は例空間 E における $HFP(F, V, \emptyset)$ の解の一つである．

[命題 2 の証明] 命題 1 より仮説 $H = mgs(C_1, C_2)$ としたとき $B \cup H \cup V \models F$ を満たす．

［例 1］ 関係 R_{animal} からなる組 $(F_{animal}, V_{animal}, \emptyset)$ に対して，図 2 の関係 R_1, R_2 に分割した場合を考える．関係 R_1 からなる組 $HFP(F_1, V_1, \emptyset)$, 関係 R_2 からなる組 $HFP(F_2, V_2, \emptyset)$ の解をそれぞれ H_1, H_2 とする．得られた仮説が

$$H_1 = \left\{ dangerous(X) \leftarrow large(X), black(X) \right\},$$

```
Input:  節の集合 H1 = {C1, C2, ..., Cn} , H2 = {D1, D2, ..., Dm}
Output: H1, H2 を融合した節の集合 H
Procedure ClauseCrossedSpecialization:
1     for 1 ≦ i ≦ n do
2         for 1 ≦ j ≦ m do
3             hC =: hd(Ci);
4             hD =: hd(Dj);
5             if p(hC) = p(hD) then
6                 S := mgs(Ci, Dj);
7                 AddClause(H, C);
8             end if
9         end for
10    end for
11    return H.
    end .
    Subrutine AddClause(H, C):
12    for for each H に含まれる C1 do
13        if C1 が C を包摂する then
14            return ;
15        end if
16    end for
17    H に C を追加する;
18    return ;
    end .
```

図 3　節交差特殊化法

$$H_2 = \left\{ \begin{array}{l} dangerous(X) \leftarrow horse(X), black(X) \end{array} \right\}$$

であった場合, これらを融合して仮説

$$H = \left\{ \begin{array}{l} dangerous(X) \leftarrow \\ \qquad large(X), black(X), horse(X). \end{array} \right\}$$

が得られる.

3.3　複数の節からなる仮説の融合

仮説が複数の節からなる場合も同様に融合可能である. 本節では融合を行うための**節交差特殊化法**を提案する.

[定義 8]　二つの異なる例空間 E_1, E_2 から得られた複数の節からなる仮説を H_1, H_2 とする. 頭部の述語記号が等しい節 C, D を H_1, H_2 からそれぞれ選択し, $mgs(C, D)$ で得た節の集合を包摂に基づいて汎化して得られた仮説 H を出力する手法を**節交差特殊化法**という.

節交差特殊化法のアルゴリズムは図 3 に示す.

[例 2]　関係 R_{animal} からなる組 $(F_{animal}, V_{animal}, \emptyset)$ に対して, 図 2 の関係 R_1, R_2 に分割した場合を考える. 関係 R_1 からなる組 $HFP(F_1, V_1, \emptyset)$, 関係 R_2 からなる組 $HFP(F_2, V_2, \emptyset)$ の解をそれぞれ H_1, H_2 とする. 得られた仮説が

$$H_1 = \left\{ \begin{array}{l} dangerous(X) \leftarrow bear(X). \\ dangerous(X) \leftarrow black(X). \\ safe(X) \leftarrow small(X). \end{array} \right\}$$

$$H_2 = \left\{ \begin{array}{l} dangerous(X) \leftarrow large(X). \\ dangerous(X) \leftarrow black(X). \\ safe(X) \leftarrow cat(X). \end{array} \right\}$$

であった場合, これらを融合して仮説

$$H = \left\{ \begin{array}{l} dangerous(X) \leftarrow bear(X), large(X) \\ dangerous(X) \leftarrow black(X) \\ safe(X) \leftarrow small(X), cat(X) \end{array} \right\}$$

が得られる.

節交差特殊化法で融合された仮説は, 最大共通特殊化を利用しているため, 融合する前の仮説に比べて具体的な仮説になる傾向がある.

4.　Distributed EVLD

4.1　Distributed EVLD の構成

Distributed EVLD の構成を図 4 に示す. Distributed EVLD は一つの EVLD サーバー及び複数の EVLD クライアントからなる. EVLD サーバーおよび EVLD クライアントは一台のコンピュータと対応している. それぞれの EVLD クライアントには EVLD で扱うことのできる知識発見システムおよびデータベースが含まれている. 個々のデータベースには EVLD クライアントが処理を担当するデータのみ含まれており, すべてのデータベースの内容は同一ではない.

EVLD サーバーは EVLD クライアントに指示を出すコントローラの役割と得られた仮説を節交差特殊化法を用いて仮説を結合する役割を持つ. まず, EVLD サーバーは利用者によって指定された情報をすべての EVLD クライアントに送る. 情報には事実, 証拠, 背景知識の集合を生成するのに必要なデータベース内のテーブル名及びフィールド名, 仮説を生成させる知識発見システムが含まれる. EVLD クライアントに情報を送ることで仮説が生成されるが, 得られた仮説は EVLD クライアントの数だけ存在し, それぞれ例空間は異なっている. そこで EVLD サーバーは節交差特殊化法を用いて仮説を融合させ, 利用者に融合された仮説を出力する.

EVLD クライアントは 2.1 節 [に説明した EVLD とほぼ同等の処理を行う. EVLD サーバーから送られてきた情報を元に SQL を生成して, 事実, 証拠, 背景知識の集合を生成する. そして指定された知識発見システムの入力フォーマットに変換し, 知識発見システムを実行する. そして得られた仮説を EVLD の仮説フォーマットである節に変換し, EVLD サーバーに送る.

4.2　妥当性検証

節交差特殊化法を用いた場合, 特殊化した節としては正しいがその節によって被覆される例がデータベースに存在しないことがある. 仮説に含まれる節数が多い場合, このような節は取り除かれている方が望ましい. そこで Distributed EVLD では**妥当性検証** (validation) を用いて被覆される例がデータベース内で少なく, しきい値を満たさない場合, その節を融合した仮説から除去する.

EVLD クライアントには与えられた節を被覆する例を数える

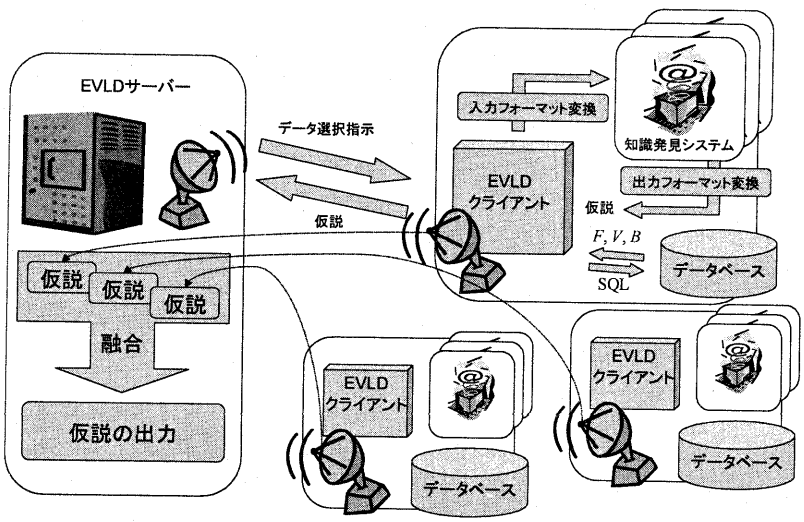

図 4　Distributed EVLD の構成

機能が備えてある．その機能は SQL でデータベースに問い合わせることで実現している．例えば R_{animal} から得られた仮説 $dangerous(X) \leftarrow large(X)$ で説明できる例とデータベースに含まれる例が一致する場合を数える SQL は属性値を含む属性を使用して

SELECT count(∗) FROM R WHERE

$class$ = '$dangerous$' AND $size$ = '$large$';

と記述する．逆に $dangerous(X) \leftarrow large(X)$ で説明できる例がデータベースに含まれる例と一致しない場合を数える SQL は

SELECT count(∗) FROM R WHERE

$class$! = '$dangerous$' AND $size$ = '$large$';

と記述する．EVLD クライアントごとに例空間は排他であるので，個々の EVLD クライアントに問い合わせて得られた結果を単純に合計することで，データベース全体の被覆する例数を得ることが可能である．

4.3　実現上の技法

Distributed EVLD は Java 言語を用いて構築されている．データベースへのアクセスは JDBC を利用している．JDBC とは Java 言語上において，統一した方法で異なる複数のデータベースサーバーとやり取りすることを可能にする API である．EVLD サーバーと EVLD クライアントとのデータのやりとりはソケット通信を使用して行う．個々の知識発見システムは Java で記述されている必要はなく，外部プログラムとして実行されてもよい．実際，C4.5 は WEKA プロジェクト [14] において Java 言語で記述された J4.8 を使用しているが，Progol および FOIL は外部プログラムとして実行している．

5.　評価実験

本稿で提案した手法の有効性を示すために Mushroom Database [3] を用いて実験を行った．Mushroom Database は形状や色など 22 の説明属性と有毒か無毒の判定が含まれている．今回の実験では C4.5, Progol, FOIL の知識発見システムを使用して，データを分割なし，二分割，三分割で生成した仮説を融合して得られた仮説の結果について比較した (図 5)．

“生成された節数” は生成された仮説 H の中に含まれている節の数を表す．分割しないで生成された仮説を表す節集合を H_0, H_0 に含まれる節の一つを C_0 とした場合，H_0 と比較して，“一致する節数” は H 中の節と等しい節の数，“包摂する節数” は H 中の節で C_0 が包摂している節の数である．逆に “包摂される節数” は C_0 を包摂する節の数である．“含まれない節数” は H_0 に含まれていて H に含まれない節の数を表し，“新規の節数” は H_0 に含まれず H に含まれる節の数である．図 5 には Mushroom Database を H によって妥当性検証した結果も載せた．“例が真である数” は妥当性検証の結果，H が説明した例の数を表す．一方，“例が偽である数” は H が説明した例が Mushroom Database に含まれる例と一致せず，一貫性制約条件を満たさなかった数を表す．

Mushroom Database で仮説を生成した場合，どの分割数および知識発見システムでも誤差が発生しなかった．そのため “例が偽である数” はすべて 0 であった．節交差特殊化法を使用した場合，生成された仮説が真である例のみを説明していれば，必ず融合された仮説も真である例のみを説明する．結果は知識発見システムの特性を反映していることが分かる．例えば C4.5 では生成された節が異なっていてもすべてデータのタプル数と等しい例が得られる．共通した傾向として分割数が増えると包摂する節数が増加する．これらは冗長である可能性があるので，なんらかの手法で汎化をすることでより分割をしない仮説に近づけることが可能である．

知識発見システム	分割数	生成された節数	一致する節数	包摂する節数	包摂される節数	含まれない節数	新規の節数	真である例数	偽である例数
FOIL	1	15	-	-	-	-	-	9884	0
FOIL	2	12	11	1	0	3	0	10040	0
FOIL	3	11	9	2	0	1	0	9428	0
Progol	1	33	-	-	-	-	-	16858	0
Progol	2	60	6	55	1	14	14	18514	0
Progol	3	101	2	264	1	9	34	20163	0
C4.5	1	19	-	-	-	-	-	8124	0
C4.5	2	19	19	0	0	0	0	8124	0
C4.5	3	25	8	48	0	0	0	8124	0

図 5　Mushroom Database から生成された仮説の節数

6. おわりに

本稿では節交差特殊化法を用いることで, 異なる例空間から得られた仮説を融合可能であることを示し, 仮説を融合することで分散した位置に存在する情報から仮説を生成可能な環境 Distributed EVLD を構築した. また Distributed EVLD で実際に仮説を生成させる実験を行った.

今回基となっている EVLD は様々な知識発見システムの仮説を論理プログラムで出力する環境である. 仮説を論理プログラムで表すことで, 論理プログラムの特性を利用して様々な応用例が考えられる. Distributed EVLD は応用例の一つで, 論理プログラムは論理演算が可能であるという特性を利用している.

Distributed EVLD ではデータを直接やり取りするのではなく, データの傾向を表す仮説をネットワーク上でやり取りすることでネットワークの帯域をあまり占有せず, またデータが直接ネットワークを流れることもないためデータの漏洩の心配もない. 例えば 1 章で述べた全国に支店の存在する店の POS データからデータマイニングを行う場合, 支店からネットワークで仮説のみが本店に送られる. そして本店で仮説を融合することで全体の仮説を得られる. このとき POS データに含まれる個人情報は一切ネットワーク上を流れない. データの傾向を表す仮説から個人情報を得ることは不可能である. このような視点からみると, 仮説とはデータの不可逆圧縮を行っているともいえる.

今回仮説を融合させる際, 仮説に誤差が含まれている場合について議論しなかった. しかし, 特に Apriori アルゴリズム [1] などでは, 仮説に誤差が含まれているため, 得られた結果をそのまま融合すると誤差がさらに大きくなることが予想される. そのため, 誤差が大きくなるのを防ぐ何らか手法が必要となるであろう. これは今後の課題である.

文　献

[1] R. Agrawal, T. Imielinski, and A. Swami : Mining Association Rule between Sets of Items in Large Database. *Proceedings of the ACM SIGMOD Conference on Management of Data*, pp. 207–216 (1993).

[2] H. Arimura and A. Yamamoto : Inductive Logic Programming : From Logic of Discovery to Machine Learning. *IEICE Transactions on Information and Systems*, Vol. E83-D, No.1, pp. 10–18 (2000).

[3] Blake, C.L. and Merz, C.J. : UCI Repository of machine learning databases [http://www.ics.uci.edu/ mlearn/ MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science (1998).

[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 (2001).

[5] 菊池敏幸, 山本章博 : 発見の論理に基づく知識発見環境とアプリケーションの接続. 情報処理学会第 64 回全国大会, CD-ROM (2002).

[6] 菊池敏幸, 山本章博 : 様々な発見システムを統一的に利用可能な環境の構築. 2002 年度人工知能学会 (第 17 回) 論文集, pp. 576–584 (2002).

[7] T. Kikuchi, A. Yamamoto. : Unifying Various Knowledge Discovery Systems in Logic of Discovery. *Proceedings of the 12th European-Japanese Conference on Information Modelling And Knowledge Bases*, pp. 131-140 (2002).

[8] S. Muggleton :Inverse entailment and PROGOL. *New Gen. Comput.*, Vol. 13, pp. 245–286 (1995).

[9] H. Ohwada, H. Nishiyama, and F. Mizoguchi : Concurrent Execution of Optimal Hypothesis Search for Inverse Entailment. *Inductive Logic Programming, Lecture Note in Artificial Intelligence*, 1866, pp. 165–173 (2000).

[10] G. D. Plotkin : A Further Note on Inductive Generalization. *Machine Intelligence 6*, pp. 101–124. Edinburgh University Press (1971).

[11] J. R. Quinlan : *C4.5: Programs for Machine Learning*. Morgan Kaufmann (1993).

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

[13] S.-H. Nienhuys-Cheng, R. de Wolf: *Foundations of Inductive Logic Programming*. Springer-Verlag (1997).

[14] I. H. Witten and E. Frank : *DataMining* . Morgan Kaufmann (2000).