特徴的ルール生成における冗長ルール排除と探索高速化

牧 秀行[†] 大河内 一弥[†] 森田 豊久[†]

データマイニングの代表的手法である特徴的ルール生成手法として，ITRULE や CHRIS などがある．これらは，データの特徴を if-then ルールの形式で出力し，通常，if 部は複数の条件の積条件である．任意の if-then ルールに対して，その if 部に条件を追加してできるルールを考え，そのルールの評価値が，条件を追加する前のルールの評価値を上回らなかった場合，これを冗長なルールと考える．データマイニングにおいては，有用なルールをユーザに見やすく提示することが重要であり，そのためには，このような冗長ルールは排除することが望ましいが，ルールの探索順序によっては，探索中に冗長であることが分からず，結果として冗長ルールが残ったり，その影響で，本来であれば結果に採用されるべきルールが欠落したりするといった問題が起こりうる．本論文では，特徴的ルール生成手法 CHRIS において，深さ優先探索，広さ優先探索のそれぞれについて冗長ルールを完全に排除する方法を示す．また，探索処理の高速化手法について述べる．深さ優先探索で冗長ルールを排除する方法は並列処理に向かないという短所がある．広さ優先探索による方法は，単体の計算機では深さ優先探索と比べて高速化の効果が小さいが，冗長ルールを排除し，かつ，処理を並列化することが可能である．

Techniques of Eliminating Redundant Rules and Acceleration for Characteristic Rule Induction

Hideyuki Maki,[†] Kazuya Ohkouchi[†] and Toyohisa Morita[†]

Characteristic rule induction is one of the typical methods of data mining. Several methods have been proposed for it, such as ITRULE and CHRIS. They describe characteristics of data in if-then rules whose if-parts contain multiple conditions. Suppose the rule which is derived by adding one or more conditions to the if-part of a rule. When the evaluation value of the rule including the extra conditions is not higher than that of the original rule, the derived rule is recognized as being redundant. It is preferable to eliminate redundant rules for ease of understanding the result of rule induction. This paper presents the algorithms of characteristic rule induction which eliminate redundant rules in both of depth-first and breadth-first manner. The methods to accelerate the rule search process are shown as well. The depth-first rule search can achieve larger reduction of processing time than breadth-first one in acceleration on a single processor. However, it has the drawback of difficulty in parallelization. On the other hand, rule search in breadth-first manner is suitable for parallel processing.

1. はじめに

データベースが大規模化し，大量のデータが蓄えられるようになるにつれて，データから有用な情報を抽出するための技術としてデータマイニングの研究が行われるようになった[1]．なかでも，データに内在する情報をルールの形式で抽出する「ルール生成」が代表的である．ルール生成手法としては，相関ルールを生成する “Apriori” アルゴリズム[2] がよく知られている．Apriori は，複数のアイテムを含むトランザクションの集合を入力とし，1 つのトランザクションに同時に含まれる傾向にあるアイテムの組合せを抽出する．

一方，これとは別のタイプのルール生成手法として，Smyth らによる “ITRULE” アルゴリズムがある[3]．ITRULE アルゴリズムは，複数の属性からなるサンプルの集合を入力とし，属性間の関係を以下のような if-then ルールとして抽出する．

if $A = a_i$ and $B = b_j$ then $X = x_k$

ここで，A，B，X が属性，a_i，b_j，x_k が属性値，属性と属性値の組が「条件」である．if 部の条件を「条件節」，then 部の条件を「結論節」と呼ぶことにする．if 部は 1 個以上の条件節，then 部は 1 個の結論節からなり，どの属性も 1 つのルール中にはたかだか 1 回しか現れない．このようなルールの形式は，決定木[4] によって表されるクラス分類のルールと類似している

† 株式会社日立製作所システム開発研究所
Systems Development Laboratory, Hitachi, Ltd.

が，決定木ではサンプルをクラスに分類することが目的であるのに対し，ITRULE ではサンプル集合の特徴を記述することが目的である．したがって，すべてのサンプルがいずれかのルールに合致する必要はなく，また，1 つのサンプルが複数のルールに合致してもよい．このようなルールは「特徴的ルール」と呼ばれる．

1 つのデータ集合について記述できるルールは，形式上は多数ある．データが n 個の属性からなり，各属性が m 通りの属性値をとりうるとき，ITRULE では，可能なルールの数 R を以下で与えている．

$$R = nm\{(2m+1)^{n-1} - 1\} \quad (1)$$

nm は then 部に現れうる結論節の数，$n-1$ は if 部に現れうる属性の数を表す．$2m+1$ は，if 部の属性の任意の 1 つ A について，“$A = a_i$” という形式の条件が m 通り，“$A \neq a_i$” の形式が m 通りと，A が現れない場合の，$2m+1$ 通りの場合があることを表す．右辺最後の -1 は，if 部にどの属性も現れないという場合を除くことを表す．ITRULE では，“J-measure” と呼ぶ，ルールの有用性を評価する尺度を用い，これら R 個の候補ルールの中で評価値の大きいルールを，ユーザによって指定された数だけ出力する．ルール “if Y then X” の J-measure は以下で与えられる．

$$J = P(y)\left[P(x|y)\log\left(\frac{P(x|y)}{P(x)}\right) + (1-P(x|y))\log\left(\frac{1-P(x|y)}{1-P(x)}\right)\right] \quad (2)$$

$P(y)$，$P(x)$，$P(x|y)$ は，それぞれ，条件 Y が成り立つ確率，条件 X が成り立つ確率，条件 Y の下で条件 X が成り立つ確率である．

文献 3) では，評価値の大きいルールを探索する手順として，以下のような深さ優先探索によるアルゴリズムが示されている．まず，条件節を 1 個だけ含むルール（1 条件ルール）の 1 つについて評価（評価値の算出とルール生成結果として採用するかどうかの判定）を行う．次いで，そのルールに別の条件節を積条件として追加してできるルールの 1 つについて評価を行う．条件節を追加することをルールの「特殊化」と呼ぶ．ルールの特殊化と評価を，特殊化の停止条件が成立するまで繰り返し，停止条件が成立すると，探索してきた経路をバックトラックし，また別の条件節による特殊化を行う．このような，特殊化と評価，バックトラックの操作を，すべての 1 条件ルールについて行う．特殊化の停止条件は，評価中のルールの if 部に合致するサンプル数が 0 となった場合，または，以後の特殊化によってできるルールの評価値が，現時点で採用されているルールの評価値の最小値を上回らないことが分かった場合である．特殊化の停止により，R 個の候補ルールのすべては評価せずに済むことが期待される．これは探索木の枝刈りを意味する．

ITRULE と似た，別の特徴的ルール生成手法として，“CHRIS” がある[5)~7)]．ITRULE とのおもな違いは，ルールの評価尺度と候補ルールである．CHRIS では，以下に示す評価尺度 μ を用いる．

$$\mu = P(y)^{\beta} P(x|y)\log\frac{P(x|y)}{P(x)} \quad (3)$$

パラメータ β は $0 \leq \beta \leq 1$ の値をとる．また，CHRIS では，候補ルールに以下の制限を加えている．

- 結論節を 1 つに固定する．
- 条件節の数の上限を定める．
- 条件節には否定の形式 “$A \neq a_i$” を含まない．

これにより，CHRIS の候補ルールは ITRULE の候補ルールの部分集合となっている．

ルールの特殊化の目的は，条件節を追加することによって評価値がより大きいルールを発見することである．特殊化によってできるルールの評価値が，元のルールを上回らなければ，特殊化の効果がないことになる．特殊化によってできるルールのうちで元のルールよりも評価値が大きくないものを「冗長ルール」と呼ぼう．データマイニングにおいては，有用なルールをユーザに見やすく提示することが重要であり，そのためには，冗長ルールは排除した方がよい[8)]．

冗長ルールを排除するには，ルールを評価する際に，そのルールが冗長であるかを判定し，冗長であれば採用しないとする．しかし，探索順序によっては，採用判定の後でそのルールが冗長であることが判明する場合がある．すると，冗長ルールとして排除すべきルールがいったんは採用されたために，本来であれば採用されるべきルールが欠落するという問題が起こりうる．冗長ルールの判定が確定するまで，採用の可能性のあるルールをすべて仮採用として保持しておくことも考えられるが，ルール総数は非常に大きくなりうるため，このような方法は非現実的であり，やはり，冗長ルールは評価と採用判定の段階で排除される必要がある．

本論文では，CHRIS において冗長ルールを完全に排除するルール探索方法を示す．また，冗長ルールを排除するルール探索方法における探索処理の高速化手法について述べる．

2. 深さ優先探索によるルール探索

2.1 冗長ルールの排除

CHRIS において，候補ルールは**図 1** に示すような木構造の上に対応づけることができる．これを「ルー

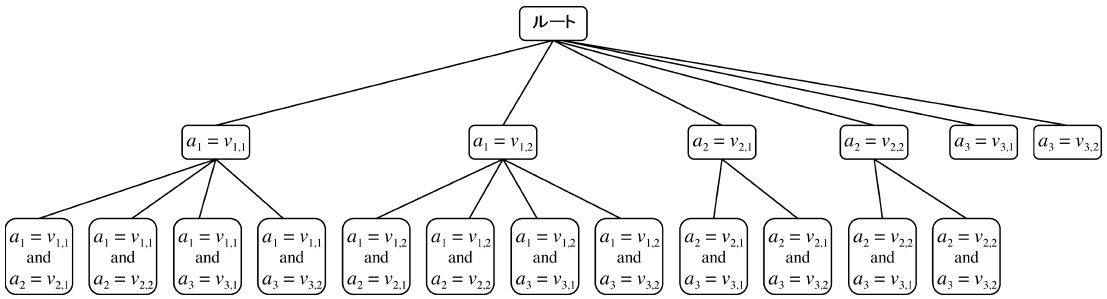

図 1 ルール木の例
Fig. 1 An example of rule tree.

ル木」と呼ぶことにする．1 つのノードが 1 つのルールに対応し，ノードに記述された a_i，$v_{i,j}$ は，それぞれ，ルールの条件節の属性と属性値を示す．属性は $a_1, a_2, \ldots$ と順序づけされるものとし，ルール木では，順序が前である属性を持つルールがより左になるように配置されている．また，複数の条件節を持つ任意のルール R_c を考え，R_c から条件節を 1 つ取り除いてできるルールを R_c の「親ルール」と呼び，これに対して，R_c を「子ルール」と呼ぶことにしよう．R_c の親ルールは複数存在するが，R_c の条件節のうちで最も順序が後の属性からなるものを取り除いてできる親ルールを「直系の」親ルールと呼ぶこととし，ルール木では，直系の親ルールの下位にルール R_c を配置する．

このルール木に沿って，深さ優先でルール探索を行うことを考えたとき，左側のルールから先に探索すると，つまり，条件節が 1 個のルールのうちで最も左側に位置する「$a_1 = v_{1,1}$」を最初に評価し，次いで，その子ルールのうちで最も左側に位置する「$a_1 = v_{1,1}$ and $a_2 = v_{2,1}$」を評価する，というようにすると，冗長ルールを排除できない場合がある．たとえば，ルール「$a_1 = v_{1,1}$ and $a_2 = v_{2,1}$」について考える（図 1 の下層左端のルール）．これは「$a_1 = v_{1,1}$」の子ルールであり，同時に，「$a_2 = v_{2,1}$」の子ルールでもある．このルールの探索順序を見ると，まず，直系の親ルール「$a_1 = v_{1,1}$」が，次に，この「$a_1 = v_{1,1}$ and $a_2 = v_{2,1}$」が評価され，もう 1 つの親ルール「$a_2 = v_{2,1}$」が評価されるのは，その後である．もし，「$a_1 = v_{1,1}$ and $a_2 = v_{2,1}$」の評価値が「$a_2 = v_{2,1}$」より大きくない場合，「$a_2 = v_{2,1}$」に対して冗長であり，このルールは排除されなければならない．しかし，「$a_1 = v_{1,1}$ and $a_2 = v_{2,1}$」の評価時点では「$a_2 = v_{2,1}$」は未評価であるため評価値の比較ができず，冗長であると分からないので，排除できない．このように，ルール木を左から深さ優先で探索すると，直系でない親ルールに対して冗長であるかどうかの判定が，子ルールの評価時に確定できない．

```
(01) 採用ルールリストを評価値 0 に初期化する.
(02) 第 1 階層の右端をカレントルール r_c とする.
(03) while (true) do begin
(04)    r_c の評価値 μ_c を算出.
(05)    r_c の採用判定.
(06)    if (r_c の子ルールが存在) then
(07)       右端の子ルールを r_c とする.
(08)    else
(09)       while (true) do begin
(10)          if (r_c の左に兄弟ルールが存在) then
(11)             左隣の兄弟ルールを r_c とする.
(12)             break
(13)          else if (r_c の階層 > 1) then
(14)             r_c の直系の親ルールを r_c とする.
(15)          else
(16)             goto (21)
(17)          end if
(18)       end
(19)    end if
(20)end
(21) 採用ルールリストをルール生成結果とする.
```

図 2 深さ優先によるルール探索アルゴリズム
Fig. 2 Rule search algorithm in depth-first manner.

これに対し，ルール木の右側のルールから先に評価すれば，複数ある親ルールのうちで直系の親ルールが最後に評価され，その後に子ルールが評価されるので，確実に冗長ルールの判定ができる．この方法によるルール探索アルゴリズムを**図 2** に示す．この中で，「採用ルールリスト」は，あらかじめ定められた出力ルール数だけルールを保持するリストであり，そこに保持されるルールはつねに評価値の大きい方から順に並べられる．また，「兄弟ルール」というのは，同じ直系の親ルールを持つルールのことである．

「(05)r_c の採用判定」について説明する．複数の条件節を持つルールから 1 つ以上の条件節を取り除いてできるルールを，そのルールの「先祖ルール」と呼ぶことにする．親ルールは先祖ルールに含まれる．r_c の

評価値を μ_c，その時点の採用ルールリストの末尾の評価値を μ_{th} とする．r_c が採用されるのは，以下の 2 つの条件の両方を満たしたときである．

- $\mu_c > \mu_{th}$.
- r_c の先祖ルールで，μ_c 以上の評価値を持つものが，その時点の採用ルールリスト中に存在しない．

これらの条件を満たした場合，採用ルールリストの末尾のルールを削除し，r_c を採用ルールリスト中の，評価値に応じた位置に挿入する．

2.2 探索の高速化

ルールの条件節の属性として用いられる属性（条件属性）の数を n，1 つのルールの持つ条件節数の上限（最大条件節数）を L としたとき，候補ルールをすべて評価する処理時間は $O(n^L)$ で増加する．したがって，多くの属性を持つデータを対象とする場合，処理時間は非常に大きくなる．文献 3) では，ITRULE において，特殊化によってできるルールの評価値の上界を用いることによりルールの特殊化を停止できることが示されているが，CHRIS でも同様にして，評価するルールの数を削減することができる．

任意のルール r_a を “if A then X” とし，その評価値を μ_a とする．A は複数の条件節の積条件であってよい．このとき，以下の式で与えられる μ_{UB} は，r_a の条件節に 1 つ以上の条件節を追加してできるルールの評価値の上界である．

$$\mu_{UB} = \max\left[\left(\frac{N_{xa}}{N}\right)^{\beta} \log\frac{N}{N_x}\ ,\ \mu_a\right] \qquad (4)$$

ここで，N は対象データに含まれるサンプル数であり，N_x，N_{xa} はそれぞれ，条件 X を満たすサンプルの数，条件 X と条件 A の両方を満たすサンプルの数である．評価値上界 μ_{UB} に関する以下の 2 つの条件（上界条件と呼ぶことにする）の少なくとも一方を満たすとき，r_a の条件節に 1 つ以上の条件節を追加してできるルールはルールの採用条件を満たしえない．

- $\mu_{UB} \leq \mu_a$
- $\mu_{UB} \leq \mu_{th}$

したがって，この場合，ルール r_a の子ルールは評価せずに済ますことができる（ルール木の枝刈り）．そこで，図 2 中のステップ（06）

if (r_c の子ルールが存在) **then**

に，条件を追加して

if (r_c の子ルールが存在 **and**
上界条件が不成立) **then**

とし，カレントルールの評価後，上記の上界条件の一方でも満たされた場合は，子ルールの評価をスキップすることによって評価するルールの数を削減する．この上界条件は，枝刈りができるための十分条件である．

3. 広さ優先探索によるルール探索

3.1 広さ優先探索の必要性

深さ優先探索において冗長ルールを排除する方法は，ルール木におけるルールの配置と探索順序に依存する．冗長ルールが確実に排除されるためには，必ず順序どおりに探索しなければならず，並列処理による高速化ができない．そこで，冗長ルールを排除でき，さらに並列処理に向いたルール探索方法を検討する．

冗長ルールを排除するために必要なのは，ルールを評価する際に，そのすべての先祖ルールの評価が終了していることである．これは，ルール木を上位から順に広さ優先で探索することによって実現できる．また，同じ階層のルールどうしには探索順序の制約はない．したがって，各階層の中では並列処理が可能である．

3.2 広さ優先によるルール探索アルゴリズム

広さ優先によるルール探索では，同じ階層に含まれるルールどうしについては探索順序の制約はないので，ルール木を左右どちらから探索してもよい．ここでは，前述の深さ優先探索に合わせて，右から左へ向かって探索する場合を例にとる．広さ優先によるルール探索アルゴリズムは，実は深さ優先探索とよく似ている．この様子を**図 3** に示す．直系の親ルールが異なるルールの間を移動するところ，図ではルール「$a_1 = v_{1,2}$ and $a_2 = v_{2,1}$」から「$a_1 = v_{1,1}$ and $a_3 = v_{3,2}$」へ移るところは，それまでの直系の親ルール「$a_1 = v_{1,2}$」へバックトラックし，その兄弟ルールである「$a_1 = v_{1,1}$」へ進み，その子ルール「$a_1 = v_{1,1}$ and $a_3 = v_{3,2}$」へ進むという深さ優先探索の動きと同じである．このことは，後述の探索高速化において重要になる．このルール探索アルゴリズムを**図 4** に示す．

3.3 探索の高速化

深さ優先探索の場合と同様に，子ルールの評価値の上界を利用した評価ルール数の削減による探索高速化を考える．深さ優先と広さ優先で大きく異なるのは，任意のルール r_a の評価において，深さ優先では，r_a における枝刈りの判定（上界条件の判定）の後，ただちに枝刈りを実行（子ルールの探索をスキップ）できるのに対し，広さ優先の場合は，次の階層の探索まで枝刈り判定結果を保持しておかなければならないという点である．そのための方法として，ある階層において枝刈り判定が真となったルールをすべて記憶しておくことも考えられるが，ルールの条件節数が大きい場合，必要な記憶量が大きくなるという問題がある．

そこで，ルール単位ではなく，条件節単位に枝刈り

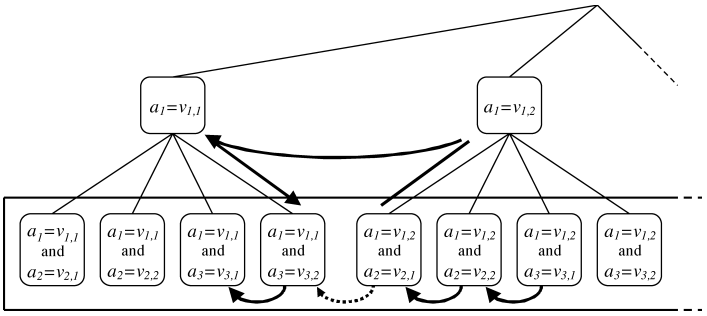

図 3 広さ優先におけるルール探索順序
Fig. 3 Rule search sequence in breadth-first manner.

```
最大条件節数:L
(01) for (i = 1 ; i ≤ L ; ++i) do begin
(02)   第 i 階層の右端をカレントルール r_c とする
(03)   while (true) do begin /* 第 i 階層探索 */
(04)     r_c の採用判定
(05)     do begin
(06)       if (r_c の階層 < i
(07)           and r_c の子ルールが存在) then
(08)         右端の子ルールを r_c とする
(09)       else
(10)         while (true) do begin
(11)           if (r_c の左に兄弟ルールが存在) then
(12)             左隣の兄弟ルールを r_c とする
(13)             break
(14)           else if (r_c の階層 > 1) then
(15)             r_c の直系の親ルールを r_c とする
(16)           else
(17)             goto NEXT /* 第 i 階層終了 */
(18)           end if
(19)         end
(20)       end if
(21)     end until (r_c の階層 == i)
(22)   end
(23) NEXT:
(24) end
```

図 4 広さ優先によるルール探索アルゴリズム
Fig. 4 Rule search algorithm in breadth-first manner.

判定結果を保持する方法をとる．条件節の総数はルールの条件節数によらず，各々の条件属性がとりうる属性値の数の総和である．

条件節単位で枝刈り判定結果を記憶しておく方法は，以下のようになる．任意の条件節「$a = v$」を考える．第 i 階層において，条件節「$a = v$」を持つルールは 1 個以上存在するが，これらのすべてで枝刈り判定が真となった場合，かつ，その場合のみ条件節「$a = v$」の第 i 階層における枝刈り判定を真とする．この場合，第 $i+1$ 階層において条件節「$a = v$」を持つルールは，その親ルールの少なくとも 1 つで枝刈り判定が真となっていることになる．親ルールにおいて枝刈り判定が真となることは，子ルールの評価をスキップできるための十分条件なので，少なくとも 1 つの親ルールで枝刈り判定が真となった子ルールは評価せずに済ませることができる．

第 $i+1$ 階層の探索において枝刈りを実行するには，各々のルールを評価する際に，まず，そのルールの条件節（$i+1$ 個ある）の各々について，第 i 階層における枝刈り判定結果を参照し，1 つでも判定結果が真であれば，そのルールは評価せずに次のルールへ進むようにすればよい．しかしながら，この方法では，すべてのルールについて枝刈り判定結果のチェック処理を行うことになり，効率が悪い．ここで，前述したように，広さ優先のルール探索順序が深さ優先とよく似ていることを利用すると，いくつかのルールの評価をまとめてスキップできる場合がある．このためには，図 4 において，(06)〜(07) の，子ルールをカレントルールにする条件に枝刈り判定を追加し，

if (r_c の階層 $< i$
and r_c の子ルールが存在
and r_c における枝刈り判定が偽) **then**

とする．つまり，ルール評価を行っている階層よりも上位の階層で，子ルールをスキップするのである．たとえば，図 3 において，条件節「$a_1 = v_{1,2}$」の枝刈り判定結果が真であった場合，その下位のルールを探索せず，ただちにルール「$a_1 = v_{1,1}$」へ移り，その下位のルールの評価へ進むことにより，ルール「$a_1 = v_{1,2}$」の子ルールの評価をまとめてスキップできる．

条件節単位で枝刈り判定結果を保持する方法では，条件節「$a = v$」を持つすべてのルールで枝刈り判定が真とならなければ，条件節「$a = v$」の枝刈り判定が真とならないので，深さ優先探索における，ルールごとに枝刈り判定をする方法に比べて枝刈り判定が真となる場合が少ないと予想される．一方で，深さ優先探索の枝刈りにはない利点もある．深さ優先探索では，直系の親ルールにおける枝刈り判定しか利用しないのに対し，広さ優先探索では，すべての親ルールにおけ

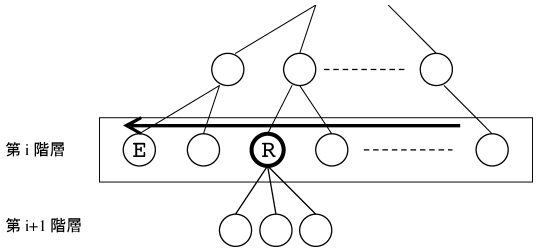

図 5 枝刈りの遅延判定
Fig. 5 Delayed determination of pruning.

る枝刈り判定を利用できる．たとえば，条件節が「A and B and C」であるルールを評価する際，深さ優先探索では，直系の親ルール「A and B」の枝刈り判定結果しか利用しない．これに対し，広さ優先探索における条件節単位の枝刈り判定では，ルール「A and B」,「A and C」,「B and C」の枝刈り判定結果を利用することができる．

さらに，以下のような手法が考えられる．**図 5** で，R は第 i 階層に属するルールの 1 つとする．R の子ルールの評価値の上界を UB_R としよう．また，R の評価時点での採用ルールリストの最下位のルール評価値を th_R とし，このとき，$UB_R > th_R$ で，枝刈り判定が偽であったとする．その後，第 i 階層のルール評価を進め，左端のルール E の評価が終わった時点での採用ルールリストの最下位のルール評価値を th_E とすると，必ず $th_R \leq th_E$ であるが，このとき，$UB_R \leq th_E$ であれば，R の子ルールは枝刈りできることになる．つまり，R の評価時に算出した UB_R を保持しておき，第 i 階層のルール評価の終了時に枝刈り判定を行えば，R の評価時に判定をするよりも枝刈り判定が真となる機会が増えることが期待できる．これを「枝刈りの遅延判定」と呼ぶことにする．

4. 評価実験

以上で述べた，深さ優先，および広さ優先によるルール探索について，それらの処理時間を評価する実験を行った．入力データには，インターネット上に公開されている，DNA マイクロアレイによる遺伝子発現データを用いた．これは，DNA マイクロアレイデータ解析の会議である Critical Assessment of Microarray Data Analysis 2000（CAMDA '00）[9] において使用されたものであるが，もとは Golub らの 1999 年の論文[10] で用いられたものである．このデータは，サンプル数は 72 と比較的少数だが，属性数は 7,129 で，属性数の多いデータである．実験では，いくつかの属性を抜粋して使用した．各々の属性は DNA マイクロアレイで計測された遺伝子発現量に対応する．発現量は実数値なので，データの最大値と最小値の間を等間隔に 3 分割して「大」「中」「小」の 3 カテゴリに離散化した．また，本実験では評価尺度（式 (3)）のパラメータ β の効果は評価しないこととし，値を 1 とした．計算には CPU クロック 1.6 GHz，メモリ 256 MB の計算機を用いた．生成されたルールの一例を**図 6** に示す．「M11507_3_at」などの番号が条件属性であり，「ALL/AML」,「ALL」がそれぞれ，結論節の属性と属性値である．上記の例は 3 個の条件節を持つルールで，“〈95.7%〉[44/46]” の数字は，条件節に合致するサンプル数が 46，そのうち結論節にも合致するサンプル数が 44 で，その割合が 95.7%であることを示す．

```
[1]評価尺度:0.233
 IF
  M11507_3_at = 小
  D14664_at = 小
  D50915_at = 小
 THEN
  ALL/AML = ALL  <95.7%>[44/46]

[2]評価尺度:0.229
 IF
  M11507_M_at = 小
  D14664_at = 小
  D50915_at = 小
 THEN
  ALL/AML = ALL  <97.6%>[41/42]

[3]評価尺度:0.220
 IF
  M11507_3_at = 小
  D14664_at = 小
  D26579_at = 小
 THEN
  ALL/AML = ALL  <93.6%>[44/47]
```

図 6 生成されたルールの例
Fig. 6 Example of induced rules.

まず，深さ優先探索と広さ優先探索の処理時間の比較を行った（**図 7**）．ここでは，枝刈りによる高速化は行っていない．深さ優先探索では，冗長ルールを排除するためにルール木の右から探索する方法をとっている．広さ優先探索では，ルール木を左右どちらから探索してもよいが，比較のために両方の場合を実験した．ルールの最大条件節数が 2，属性数が 5,000 の場合（Case1）と，最大条件節数が 3，属性数が 500 の場合（Case2）の，2 通りで評価したところ，各々のケースにおいて，深さ優先と広さ優先で処理時間の差はほとんどない．広さ優先で，ルール木を探索する向きによって処理時間が若干異なるのは，探索の途中で採用ルールリストに含まれるルールが異なるからと考えられる．むろん，探索終了時の採用ルールリストの内容はすべての探索方法において同一である．

次に，枝刈りによる高速化を取り入れて実験した（**図 8**）．深さ優先探索の方が高速化の効果が大きいといえる．広さ優先探索では，枝刈りの遅延判定を取り

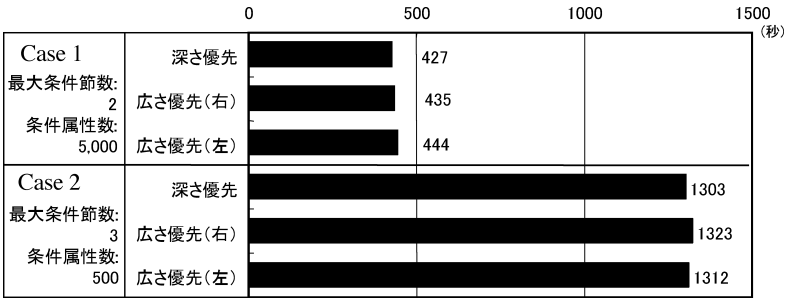

図 7 深さ優先，広さ優先探索の処理時間比較
Fig. 7 Comparison of processing time between depth-first and breadth-first search.

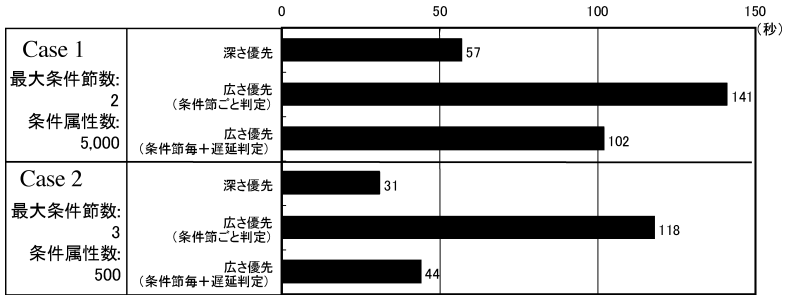

図 8 高速化を取り入れた処理時間の比較
Fig. 8 Comparison of processing time of accelerated rule search.

入れると高速化の効果が増大する．特に，最大条件節数が多い方が遅延判定の効果が大きい．

5. 他のルール探索手法との関係

冗長ルール排除と探索高速化について，CHRIS 以外のルール探索手法との関係を考察する．

(1) ITRULE

ITRULE では，可能なすべての結論節を対象としているが，その各々についての候補ルール集合が CHRIS の候補ルール集合に対応する．また，探索高速化に関しては，子ルールの評価値上界を用いた枝刈りは，ITRULE における特殊化停止条件と同じ考え方によっている．したがって，ITRULE の結論節の各々についてのルール探索処理の中では，本論文で述べた冗長ルール排除と探索処理の高速化手法が適用できる．

(2) 相関ルール

相関ルールは，support と confidence の 2 種の尺度の組で評価される点が ITRULE や CHRIS と異なる．したがって，本論文の手法を相関ルール生成に適用することを考えるには，相関ルールにおける冗長ルールの定義や評価値上界の算出方法の検討が必要である．

(3) 決定木

決定木におけるルール探索では，特徴ルール生成と同様に，条件節を追加し（特殊化），評価する処理を繰り返すが，特徴ルール生成では，枝刈りされた場合を除き，採用されなかったルールについても特殊化を行うのに対し，決定木では，採用されなかったルールの特殊化を行わない．特殊化および評価の考え方が異なるため，本論文の手法は決定木には適さない．

(4) プロダクションシステム

プロダクションシステムは，ルール集合とデータが与えられ，データに合致するルールを選び出し，そのルールに従って結論を導くという，推論手法の一種である．データを用いてルールを評価し，選び出すという点では，ルール生成手法と類似の処理と見ることもできる．プロダクションシステムにおいては，ルール探索が処理全体の大部分を占め，その高速化について多くの研究がなされてきた[11]．プロダクションシステムのルール集合では，条件節の属性値の部分に変数を用いることができる．その場合，1 個のルールの評価の途中で変数が値に束縛され，以後の評価処理に影響を与える．これは，これまで述べたルール生成手法との大きな違いであり，プロダクションシステムのルール評価処理を複雑で処理量の多いものとしている．本論文の探索高速化手法を，このようなルール評価処理に適用するのは困難である．

6. おわりに

本論文では，特徴的ルール生成において冗長ルールが発生し，そのために，本来であればルール生成結果として得られるべきルールが欠落する場合があることを述べ，特徴的ルール生成手法の 1 つである CHRIS について，冗長ルールを排除するルール探索方法を示した．また，並列処理向けに，広さ優先探索に基づいて，冗長ルールを排除するルール探索方法を示した．さらに，枝刈りによる高速化手法を広さ優先探索に適用し，実験によってその効果を評価した．

単体の計算機を用いた場合，高速化手法を取り入れることにより，広さ優先探索の処理時間は短縮されたが，深さ優先探索の方が高速化の効果がより大きく，処理時間は短かった．しかしながら，広さ優先探索では，冗長ルールを排除しながら処理を並列化することが可能なので，並列分散化によって処理時間を短縮することが期待できる．

参考文献

1) Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P.: Knowledge Discovery and Data Mining: Towards a Unifying Framework, *Proc. KDD 1996*, Menlo Park, CA, AAAI, pp.82–88, AAAI Press (1996).
2) Agrawal, R. and Srikant, R.: Fast Algorithm for Mining Association Rules, *Proc. 20th VLDB Conference 1994*, pp.487–499 (1994).
3) Smyth, P. and Goodman, R.M.: An Information Theoretic Approach to Rule Induction from Databases, *IEEE Trans. Knowledge and Data Engineering*, Vol.4, No.4, pp.301–316 (1992).
4) Quinlan, J.R.: Induction of Decision Trees, *Machine Learning*, Vol.1, pp.81–106 (1986).
5) Maeda, A., Maki, H. and Akimori, H.: Characteristic Rule Induction Algorithm for Data Mining, *Proc. 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp.399–400 (1998).
6) Maki, H., Maeda, A. and Akimori, H.: Data Mining Application to LSI Fault Analysis, *Proc. Int. Conf. Electrical Engineering*, Vol.1, pp.417–420, The Korean Institute of Electrical Engineers (1998).
7) Maki, H. and Teranishi, Y.: Development of Automated Data Mining System for Quality Control in Manufacturing, *Proc. 3rd Data Warehousing and Knowledge Discovery*, DEXA, pp.93–100 (2001).
8) 鈴木英之進：データベースからの特徴的ルール発見のための一般性と正確性の信頼性同時評価手法，人工知能学会誌，Vol.14, No.1, pp.139–147 (1999).
9) Lin, S.M. and Johnson, K.F. (Eds.): *Methods of Microarray Data Analysis*, Kluwer Academic Publishers (2002).
10) Golub, T.R., et al.: Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring, *Science*, Vol.286, pp.531–537 (1999).
11) 石田 亨：プロダクションシステムの発展，朝倉書店 (1996).

(平成 17 年 10 月 28 日受付)
(平成 18 年 5 月 9 日採録)

牧 秀行（正会員）

1967 年生．1990 年京都大学工学部情報工学科卒業．同年（株）日立製作所入社．システム開発研究所に所属．データマイニング，数理システム技術等の研究開発に従事．人工知能学会会員．

大河内一弥（正会員）

1974 年生．1999 年岡山大学大学院工学研究科情報工学専攻修士課程修了．同年（株）日立製作所入社．システム開発研究所に所属．データマイニング，サイバー攻撃対策の研究開発に従事．2004 年より独立行政法人情報通信研究機構専攻研究員を兼任．

森田 豊久

1964 年生．1989 年東京大学大学院工学系研究科機械工学専攻修士課程修了．同年（株）日立製作所入社．システム開発研究所に所属．データマイニング等の研究開発に従事．電子情報通信学会，日本機械学会各会員．