論 文

Technical Papers

記述長に基づく適応的Ripple Down Rules法

Adaptive Ripple Down Rules Method based on Description Length

吉田 哲也 Tetsuya Yoshida
大阪大学産業科学研究所
The Institute of Scientific and Industrial Research, Osaka University.
yoshida@ar.sanken.osaka-u.ac.jp

和田 卓也 Takuya Wada
（同　上）
washio@ar.sanken.osaka-u.ac.jp

元田 浩 Hiroshi Motoda
（同　上）
motoda@ar.sanken.osaka-u.ac.jp

鷲尾 隆 Takashi Washio
（同　上）
washio@ar.sanken.osaka-u.ac.jp

keywords: ripple down rules method, minimum description length principle, knowledge deletion, pruning

Summary

A knowledge acquisition method Ripple Down Rules (RDR) can directly acquire and encode knowledge from human experts. It is an incremental acquisition method and each new piece of knowledge is added as an exception to the existing knowledge base. Past researches on RDR method assume that the problem domain is stable. This is not the case in reality, especially when an environment changes. Things change over time. This paper proposes an adaptive Ripple Down Rules method based on the Minimum Description Length Principle aiming at knowledge acquisition in a dynamically changing environment. We consider the change in the correspondence between attribute-values and class labels as a typical change in the environment. When such a change occurs, some pieces of knowledge previously acquired become worthless, and the existence of such knowledge may hinder acquisition of new knowledge. In our approach knowledge deletion is carried out as well as knowledge acquisition so that useless knowledge is properly discarded to ensure efficient knowledge acquisition while maintaining the prediction accuracy for future data. Furthermore, pruning is incorporated into the incremental knowledge acquisition in RDR to improve the prediction accuracy of the constructed knowledge base. Experiments were conducted by simulating the change in the correspondence between attribute-values and class labels using the datasets in UCI repository. The results are encouraging.

1. はじめに

知識ベースシステムの開発の前提として人間の専門家が持っている知識は正しくかつ不変であり，時間とコストを費やしても専門家から知識を獲得することは非常に価値があると考えられてきた．しかし，インターネットに代表される通信ネットワークの進展によって大量のデータを即座に入手することが可能になるに伴い，常に最新の知識を保持するためには頻繁な知識の更新が必要となり得る [Morik 93, Wrobel 94]．そのような状況下で実用的な知識ベースシステムを構築する枠組みとして，Ripple Down Rules 法 [Compton 89]（以下 RDR 法と呼ぶ）と呼ばれる人間の専門家からの知識獲得を目的とした手法が有望と我々は考えている．RDR 法では知識獲得を既存の知識の逐次的な洗練と捉え，システムが事例を誤推論するたびに，今まで正しく判断を下してきた知識の価値を下げることなく新しい知識を既存の知識の例外知識として追加する．知識獲得は専門家とインタラクティブに行われ，知識獲得段階とシステム保守段階の間に明確な境界はない．

従来の RDR 法は逐次的に例外知識を追加することで知識ベースを更新するが，問題領域の環境は静的で事例の発生源となる母集団の性質は不変であることを仮定し，知識ベースのサイズが大きくなっても例外知識を追加し続け，遭遇した事例を全て正しく推論できる知識ベースを構築する．しかし，技術革新の激しい今日，知識の有効性は時間とともに変化し、逐次的に知識ベースを構築する過程で以前正しいと判断され追加された知識が新しい状況では有効性*1を失うこともある．有効性を失った知識をも知識ベースに保持し続けると，変化した環境下で重要な知識の獲得を阻害する恐れがある．また，専門家の知識も不変ではなく動的に変わりうるものであるため，一旦獲得した知識は将来にわたって有効性を持続するという仮定は成立しにくい．このため，環境が変化す

*1　本稿では未知データに対する予測精度から有効性を考える．

る問題領域において継続的に知識ベースシステムの性能を維持するためには，一旦獲得した知識であっても有効性を失った時点で知識ベースから削除することが重要となる．

我々は最小記述長原理 [Rissanen 78]（Minimum Description Length 原理:MDL 原理）を用いて専門家からの知識獲得とデータからの帰納学習手法を RDR 法に統合化する手法を提案した [和田 01b]．本稿では環境変化の一例として同じ属性-属性値の組合せを持つ事例に対するクラスラベルの変化を考え，知識獲得に加えて最小記述長原理に基づく知識削除機能と知識汎化機能を導入して RDR 法の知識ベースシステム (以下，RDR システムと呼ぶ) を構築する手法を提案する．提案する手法では知識の追加，削除，汎化を統一的に扱うために正規化した記述長を用いる．本稿では 15 種類のデータセットを用い，環境変化をシミュレートする人工データからの逐次的な知識ベース構築を行い，提案手法が環境変化への適応に対して有効であることを確認する．

2 章では Ripple Down Rules 法による知識獲得について説明し，3 章では提案する知識削除および知識汎化機能のアルゴリズムについて説明する．4 章では提案手法を実験的に評価する．最後に 5 章で結論を述べる．

2. Ripple Down Rules 法による知識獲得

2·1 Ripple Down Rules 法

RDR 法は知識エンジニアによる分析やインタビューなどを必要とせず，専門家が直接知識ベースシステムを開発することによって，知識獲得ボトルネックを解消しようとする知識獲得技術である．RDR システムでは，ある事例が誤って推論された場合，知識獲得（保守）段階において，なぜその事例に対する推論が知識ベースに蓄積されている過去に正しく推論された事例に対する推論と異なるのかを専門家に判断させる必要がある．

RDR システムは図 1.(a) に示すように二分木として表現される．各ノードはルール（If-Then ルール）とそのノード自身が追加される原因となった代表事例 (cornerstone case) を記憶し，二つの枝（YES 枝と NO 枝）を持つ．事例に対する RDR 法の推論は根ノードから始まり（RDR 法が推論する事例を推論事例と呼ぶ），その事例がノードの持つルールの条件部を満足すれば YES 枝のノードに，そうでなければ NO 枝のノードに推論を移行し，これ以上進むべきノードが存在しなくなるまでこの過程を繰り返す．あるノードの YES 枝の下のノードは，そのノードの条件部が満足された場合に対する例外知識を表現するため，そのノードに対する例外ノードと呼ぶ．ノードの遷移が停止したノードを end node と呼び，根ノードから end node までの経路を推論パスと呼ぶ．事例に対する RDR 法の結論は，推論パス上で一番最後に条件部が満足されたノード (last satisfied node, 以下では LSN と呼ぶ) の持つ If-Then ルールの帰結部となる．根ノードのルールの条件部はどのような事例に対しても成り立ち，その帰結部は事例に対する結論が RDR システムから導かれなかった際の暗黙の結論（デフォルト知識）を表すデフォルトクラスが設定される．デフォルトクラスはシステム構築前に設定される．

事例に対する推論結果が専門家の判断する結果と異なる場合，専門家から知識（新しい If-Then ルール）を獲得し，既存の二分木に追加する．図 1.(b) に RDR 法における知識獲得の様子を示す．RDR システムに追加するためのルールの条件部を決めるために二つの事例の差異 (difference list, 以下では D-List と呼ぶ) を専門家に提示し，追加ノードの If-Then ルールの条件部となる要素を選んで貰う．二つの事例とは RDR による推論によって誤判断された推論事例と，それに対する誤った結論を出したノードの持つ代表事例である．D-List は代表事例が否定し，推論事例が肯定する条件を要素とする集合であり，二つの事例を区別する条件集合の選択に使用する．誤判断された事例に対する正しい結論を正当化するために専門家が D-List から選んだ条件の連言を条件部，正しい結論を帰結部とする If-Then ルールを，追加するノードが保持するルールとし，誤推論された推論事例をこのノードの代表事例とする．新しいノードの追加位置は，LSN が end node であれば end node の YES 枝の下，そうでなければ end node の NO 枝の下とする．

RDR 法では知識獲得を既存の知識の逐次的な洗練と捉え，知識ベースを再編することなく，新しいノードを最後に条件部が満足されたルールを持つノードの例外ノードとして代表事例とともに追加するだけであり，知識は決して移動したり変更されることがない．このため，RDR 法は知識獲得段階で各知識が追加されたときの文脈と同じ文脈でそれぞれの知識が使われることを保証する．

2·2 最小記述長原理に基づく RDR 法

RDR 法は人間の専門家から知識を獲得する手法であるため専門家の判断力に強く依存する．しかし専門家といえども時には間違いも犯すため，専門家からの知識獲得だけに頼るのではなく，データからの帰納学習手法を利用して知識ベースシステムを構築することが望ましい．我々は MDL 原理を用いてデータからの帰納学習手法を RDR 法に統合化する手法を提案し，その有効性を確認した [和田 01b]．MDL 原理は与えられた観測データをもとに，複数の選択肢の中から最良と思われる仮説を選ぶ際の一般的な選択規範であり，データを説明するのに最もシンプルな仮説を選択すべきとの規範である．仮説の複雑さは Rissanen [Rissanen 78] が提案した “記述長（Description Length: DL）” として求められる．

二分木で表現される RDR 法の知識モデルは，データの分割の仕方と，それに基づいて分割されたデータの部分集合に対する代表クラスから構成されていると見なす

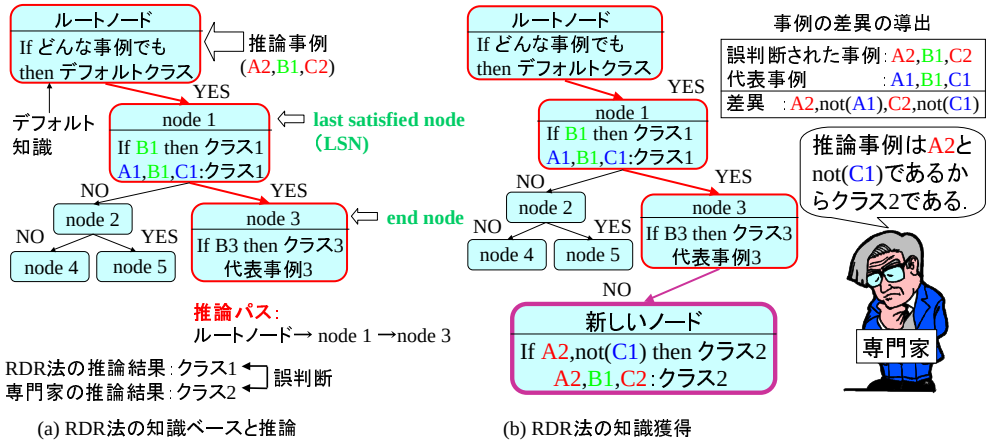

図 1 RDR の知識構造と知識獲得アルゴリズム

表 1 データの例

No.	属性 Swim	属性 Breath	属性 Legs	クラス
1	can	lung	2legs	Dog
2	can	lung	4legs	Penguin
3	can	skin	2legs	Monkey
4	can	skin	4legs	Dog
5	can_not	lung	2legs	Cat
6	can_not	lung	4legs	Cat
7	can_not	gill	2legs	Dog
8	can_not	gill	4legs	Monkey
9	can_not	skin	2legs	Penguin
10	can_not	skin	4legs	Penguin

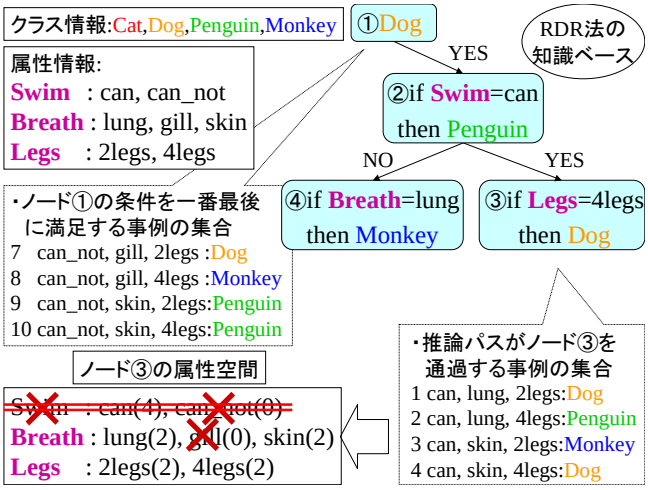

図 2 DL の計算を例証するための二分木の例

ことができる．従って，ある適切な符号化方法の下で，RDR システムの DL は 1) 知識モデル（RDR 法の二分木）に対する DL と 2) その知識モデルで誤推論される事例（これを誤推論事例と呼ぶ）の真のクラス情報に対する DL の和として計算される．我々が提案した符号化方法の詳細は文献 [和田 01b, Wada 01a] を参照されたい．1) の二分木の符号化では各ノードに対して枝情報と If-Then ルール情報を符号化する必要があり，推論過程で各ノードを通過する事例の集合から表現すべき情報（これを属性空間と呼ぶ）を考慮して符号化を行う．このため，従来の RDR 法と異なり，MDL 原理に基づく RDR 法では図 2 に示すように各ノードは代表事例以外にもそのノードを LSN とする事例集合も保持する．2) の誤推論事例のクラス情報の符号化では，各ノードを LSN とする事例のうち誤推論事例となるもののクラスラベルをそれぞれ符号化する．

MDL 原理に従えば全体の DL が最も小さくなるモデルを選択すべきであるため，RDR システム全体の DL が最小となる二分木を選択すればよい．ただし，符号化方法のほとんどは誤推論事例のクラス情報に比べて知識モデルの DL を多く見積もりすぎる傾向があることが知られている [Quinlan 93]．このため，RDR システム全体の DL を

$$\text{全体の DL} = (\text{知識モデルの } DL) \times w + (\text{誤推論事例のクラス情報の } DL) \quad (1)$$

と計算し，著者らの実験に基づく経験的な値として $w = 0.3$ を採用している．

MDL 原理に基づく RDR 法では下記のように知識獲得を行う．

専門家からの知識獲得 従来の RDR 法と同様，D-List から専門家に 1 個以上の要素を選択して貰い，選択された条件集合を新しく追加するノードでの If-Then ルールの条件部とする．

データからの知識獲得 D-List の部分集合のうち，新しいノードの条件とした場合に RDR システム全体の DL が最小になる部分集合を欲張り探索する．探索の詳細は [和田 01b] を参照されたい．この方法により，専門家に頼ることなくデータのみから帰納的に知識ベースを構築することが可能となる．

データと専門家からの知識獲得 専門家の知識も積極的に利用するために，専門家が D-List から選択した条件集合を探索の開始点として記述長が更に小さくなる条件を欲張り探索する．MDL 原理の観点から、専門家が選んだ条件よりもさらに良い条件を見つける可能性がある．

Algorithm $deleted\_RDR(RDR, node)$
RDR : RDR システム
$node$: RDR システム中のノード
1: $w_d := 0.3$;
2: $NDL_{org} := ndl(RDR, w_d)$;
3: $cases := cases(node)$;
4: $classlabel := classlabel(\text{node})$;
5: $cases_{diff} := cases\_diff\_label(cases, classlabel)$;
6: $RDR_{del} := delete\_node(RDR, node)$;
7: $distribute(RDR_{del}, cases_{diff})$;
8: $NDL_{del} := ndl(RDR_{del}, w_d)$;
9: **if** $NDL_{org} > NDL_{del}$
10: **then** return RDR_{del};
11: **else return** RDR;

$ndl(RDR, w)$: RDR システムの正規化した記述長（式 (1) で重み w で計算）
$cases(node)$: $node$ の保持事例集合
$classlabel(node)$: $node$ が保持する If-Then ルールの帰結部のクラスラベル
$cases\_diff\_label(cases, label)$: クラスラベルが $label$ 以外の事例集合
$distribute(RDR, cases)$: 事例集合 $cases$ を RDR システムで推論し，LSN となるノードに分配

図 3 ノード削除アルゴリズム

3. 適応的 Ripple Down Rules 法

知識削除機能と知識汎化機能を実現する手法として，二分木で表現される RDR システムのノード削除と枝刈りを提案する．ノード削除や枝刈りは推論事例の誤判断を契機に実施する．後述するようにノード削除の方がより積極的に環境変化に適応する手法であるため，併用する場合には，まずノード削除を行い，次に枝刈りを行う．以下では，RDR システムが入力として受け取る個々の事例を単に事例と呼び，RDR の推論結果が間違っていた事例を特に誤推論事例と呼ぶ．また，各ノードが保持している個々の事例（そのノードを LSN とする事例）を保持事例と呼び，保持事例の集合を保持事例集合と呼ぶ．

3·1 ノード削除

環境変化した母集団から抽出した事例を RDR システムにより推論した場合，その事例と同じ属性－属性値の組合せを持つ事例を過去に正しく推論した知識を保持していても，事例に対するクラスラベルが変化しているために RDR システムの推論結果は誤推論となってしまう．この場合はシステムが保持する代表事例と推論事例との差がないため D-List は空集合になり，従来の RDR 法ではノードの追加による知識獲得が行えない．また，D-List が空集合でない場合でも，専門家が重要と考える条件が D-List に含まれなければ，D-List から選択された条件を用いた If-Then ルールを知識とするノードを追加することは意味がない．

環境変化への単純な対応策としては，変化を検知する毎に構築した RDR システムを破棄し，新たな環境の下で RDR システムを再構築する事が考えられる．しかし，変化を検知する毎にシステムを再構築することは逐次的な例外知識の追加という RDR 法のアプローチにそぐわない．また知識の再利用という観点から，無効となった知識のみを削除し，新たな環境下でも有効なものを保持し続けるという枠組みの方が望ましい．そこで，RDR システムでは各ノードが推論知識を表現する If-Then ルールを保持しているため，無効となった知識を保持しているノード，つまり RDR システムが誤推論した事例に対する LSN を削除することが考えられる．しかし RDR 法は既存の知識に矛盾する事例を契機として新たな例外知識を獲得する手法であるため，RDR システムが誤推論するたびにノードを削除すると知識獲得自体ができなくなってしまう．

本稿ではシステムの推論結果が誤っているたびに例外知識を追加するという元来の RDR 法を MDL 原理に基づいて以下のように拡張する：

(1) [和田 01b] の方法と同様に，RDR システムが事例を誤推論した場合でも，誤推論を修正するためのノードの追加が全体の記述長を減少させなければノードを追加しない．
(2) 更に，記述長が減少するならば，誤推論事例に対する LSN となるノードを RDR システムから削除し，二分木を再編する．

この (2) にあたる、RDR システムからのノード削除アルゴリズムを図 3 に示す．

2·2 節で述べた MDL 原理に基づく RDR 法では，各ノードは If-Then ルールとともにそのノードを LSN とする事例の集合を保持する．誤推論事例に対する LSN を RDR システムから削除する際，図 4 に示す関数 $delete\_node$ により二分木からノードを削除して木構造を再編するとともに，そのノードの保持事例集合の中で If-Then ルールの帰結部と同じクラスラベルを持つ事例もあわせて削除する．これは，帰結部と同じクラスラベルを持つ保持事例だけがその If-Then ルールの有効性を支持していると考えるからである．削除されたノードが保持する残りの事例，すなわち If-Then ルールの帰結部と異なるクラスラベルを持つ事例は，再編した RDR システムで再推論して新たな LSN に分配する．

なお，RDR システムの記述長は事例数に対し単調に増加するため，ノードの削除前後の記述長を単純に比較しても意味がない．この点を補正するため，RDR システムが保持する事例に対して RDR システムの記述長を DL/DL′ と正規化して比較する（以下，正規化した記述長を NDL と呼ぶ）．ここで，DL′ は二分木の情報を用

関数 *delete_node*(*RDR*,*node*)
1:$RDR' := RDR$ \\ copy RDR into RDR'
2:**if** *has_child*(*node*, YES)
3:**then**
4: *parent* := *parent*(*node*);
5: **if** *edgelabel*(*parent*, *node*) = YES
6: **then**
7: *child*(*node*, YES) を *parent* に YES 枝で接続;
8: $node'$:= *child*(*parent*, YES);
9: **else**
10: *child*(*node*, NO) を *parent* に NO 枝で接続;
11: $node'$:= *child*(*parent*, NO);
12: **repeat while** (*has_node*($node'$, NO))
13: *add_cond*(*node*,$node'$);
14: $node'$:= *child*($node'$, NO);
15: *add_cond*(*node*,$node'$);
16: **if** *has_child*(*node*, NO)
17: **then** *child*(*node*, NO) を $node'$ に NO 枝で接続;
\\ node が YES 枝の子ノードを持たない場合
18:**else if** *has_childe*(*node*, NO)
19: **then**
20: *parent* := *parent*(*node*);
21: **if** *edgelabel*(*parent*, *node*) = YES
22: **then** *child*(*node*,NO) を *parent* に YES 枝で接続;
23: **else** *child*(*node*,NO) を *parent* に NO 枝で接続;
24:**return** RDR';

has_child(*node*, *label*): *node* は *label* で接続する子ノードを持つ
edgelabel($node_A$, $node_B$): $node_A$ から $node_B$ に接続する枝のラベル
add_cond($node_A$,$node_B$): $node_A$ の If-Then ルールの条件部を $node_B$ の条件部に連言として追加

図 4 関数 *delete_node*: RDR システムから指定したノードを削除

いずに各保持事例の正しいクラス情報を符号化するのに必要な記述長として根ノードの情報のみを用いた場合の記述長を表す.

RDR システムからノードを削除する例を図 5 に示す. 図 5 で誤推論事例に対する LSN はノード 2 であり，その If-Then ルールの条件部を “cond.2” と表記する．まず，図 4 に示す関数 *delete_node* の 7 行目でノード 2 に YES 枝で接続する子ノード 4 をノード 2 の親ノードであるノード 1 に YES 枝で接続してノード 2 を削除する．その際，ノード 4 と，ノード 4 の NO 枝に繋がるノードを根ノードとする部分木に含まれるノードの If-Then ルールは，ノード 2 の If-Then ルールの条件部 “cond.2” が満たされる場合に対する例外知識である．これらのノードが保持する知識の整合性を保つために，関数 *delete_node* の 13 行目で削除したノード 2 での条件部 “cond.2” を

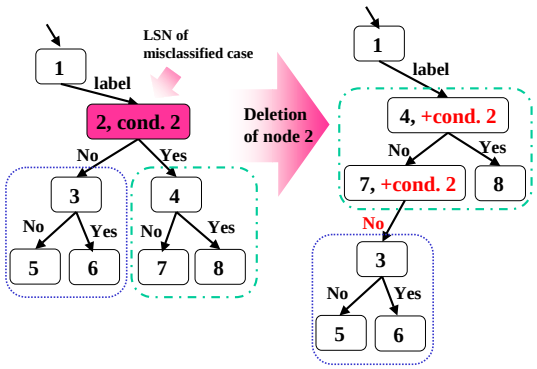

図 5 ノード削除の例

ノード 4 の条件部に追加する（図 5 で “+cond.2” は条件部を追加したことを表す）．同様に，ノード 4 から NO 枝のみを辿って到達できる全てのノード（図 5 ではノード 7 のみ) の If-Then ルールに条件部 “cond.2” を追加する．次に，ノード 2 に NO 枝で接続していた，ノード 3 を根ノードとする部分木を，関数 *delete_node* の 17 行目でノード 4 を根ノードとする部分木で NO 枝を辿った葉ノード（この場合はノード 7）に NO 枝で接続する．ノード削除アルゴリズムはノード削除前後の NDL を比較し，再編後の RDR システムの NDL が再編前の NDL より減少していれば，再編後の RDR システムを返す.

上記の操作において，削除したノード 2 での条件 “cond.2” を追加することで再編後の二分木に含まれるノードが保持する知識の整合性が維持されていることがわかる．たとえば，ノード 2 を削除する前では，ノード 3, 5, 6 の各保持事例に対する RDR による推論はノード 2 の条件部 “cond.2” が満たされないためノード 3 を根とする部分木で行われる．再編後の二分木においても，ノード 4, 7 の条件部に条件 “cond.2” を追加したためにノード 4, 7 の条件部が満たされず，ノード 3, 5, 6 の各保持事例に対する RDR による推論はノード 3 を根とする同じ部分木で行われる．ノード 2 の削除前後で同じ推論結果が得られるため，RDR システムが保持する知識の整合性を維持したノード削除を実現できる.

3·2 デフォルトクラス

デフォルトクラスが異なれば同じ事例集合に対しても性能の異なる RDR システムが構築される [和田 00]．2 章で述べたように従来の RDR 法ではデフォルトクラスは固定されており，RDR システムが事例を誤推論するたびに新しいノードを追加していた．他方，提案手法では記述長が減少しなければノードは追加されず，誤推論事例はその LSN に保持される．また，各ノードの保持事例集合はシステムが逐次的に受け取る事例に依存して変化する．システムへの入力となる事例の性質が変化した場合には，その変化に追随して適切なデフォルトクラスを設定できることが望ましい．このため，システムが受け

取る事例に基づいて逐次的にデフォルトクラスを設定する手法として，システム全体の記述長が最小化されるクラスラベルをデフォルトクラスとするよう拡張した．根ノードは条件部がなく，枝を必ず 1 つ持つため，根ノードの保持事例集合において最頻なクラスをデフォルトクラスとすると記述長が最小になる．逐次的に受け取る事例を反映しデフォルトクラスを用いることにより，環境が変化した場合に RDR システムから正しい推論結果を得られやすくなる．

しかし，根ノードの保持事例集合における最頻クラスをデフォルトクラスと設定することでシステムの正答率は向上するものの，デフォルトクラスに対する推論は事例の頻度に基づいおり，根ノードが LSN となる事例を推論するための明示的な推論知識（If-Then ルール）がシステム内部に表現されているわけではない．このため，ある事例に対して根ノードが LSN となった場合には，推論結果の正否に関わらず例外ノードを追加した場合と追加しない場合の記述長を比較し，記述長の小さい方を選択することとした．例外ノードが追加された場合には，その事例を推論するための If-Then ルールが明示的な知識としてシステムに追加されることになる．本来 RDR システムの推論結果が正しい場合は D-List が定義できないが，この場合には根ノードの推論結果が間違っていたと仮定して D-List を定義し，2･2 節で述べた手法で例外ノードを追加した場合の条件部を探索する．ただし根ノードは代表事例を持たないため，D-List に含まれる要素は推論事例の属性値のみとなる．

事例に対する LSN が根ノードの場合に対するシステムの挙動を下記にまとめる．

（1）システムは根ノードの保持事例集合における最頻クラスをデフォルトクラスとして推論する．

（2）推論結果の正否にかかわらず，その事例を推論するための If-Then ルールを表現する例外ノードの追加を試みる．MDL 原理に基づき，ノードを追加した場合のほうが，追加しない場合よりもシステム全体の記述長が小さいならばそのノードを追加する．

（3）ノードが追加された場合には，ノード追加後の RDR システムの整合性を保つために根ノードの保持事例集合をノード追加後の RDR システムで再推論し，新しい LSN に分配する．この処理により，新たに追加したノードが LSN となる事例がそのノードの保持事例として移動される．

3･3 枝刈りによる知識汎化

RDR システムを分類器と捉えると，予測精度を向上させるにはシステム構築に用いた訓練事例に対して特化し過ぎない分類器を構築することが重要である．バッチ処理で分類器を構築する機械学習法の一つである C4.5 では訓練事例に特化した分類器 (決定木) を枝刈りにより汎化して予測精度の高い分類器を構築することに倣い，

Algorithm $pruned\_RDR(RDR, case)$

```
1:  w_p := 0.7;
2:  nodes := pr_nodes(RDR,case);
3:  NDL := ndl(RDR,w_p);
4:  repeat while nodes ≠ ∅
5:    NDL_best := ∞;
6:    for each node' ∈ nodes
7:      cases := cases(node');
8:      RDR' := delete_node(RDR,node');
9:      distribute(RDR',cases);
10:     NDL' := ndl(RDR',w_p);
11:     if NDL_best > NDL'
12:       then
13:         NDL_best := NDL'
14:         RDR_best := RDR'
15:         node_best := node'
16:   if NDL_best < NDL
17:     then
18:       NDL := NDL_best;
19:       RDR := RDR_best;
20:       nodes := nodes - node_best;
21:     else return RDR;
22: return RDR';
```

$pr\_nodes(RDR,case)$: RDR により $case$ を推論した際の枝刈り対象ノードの集合

図 6 枝刈りアルゴリズム

逐次的な知識ベース構築手法である RDR 法に枝刈りによる知識汎化機能を導入した．一旦システムに追加したノードを後に削除するという意味で 3･1 節のノード削除の一種とみなすことができる．ただし，枝刈りの目的は予測精度の向上であるため，属性-属性値の組み合わせとクラスラベルの対応が変化しない静的な環境においても適用可能である．

提案する RDR 法での枝刈りアルゴリズムを図 6 に示す．3･1 節のノード削除と同様に正規化した記述長の最小化という規範で枝刈りを行う．ノード削除との違いは事例が削除されないことであり，枝刈りされたノードが保持していた各事例は削除されることはなく，全て再編後の RDR システムのノードに再分配される．このため，If-Then ルールとして表現される知識のみが RDR システムから削除されることになる．RDR システムが事例を正しく分類した場合にはシステムが保持する知識は事例と無矛盾であるため枝刈りを行わないが，3･2 節のようにデフォルトクラスを設定しているため根ノードでの推論結果は事例とともに変化する可能性がある．このため，現状では RDR システムにより事例を分類した際に，1) 新しいノードを追加した場合，2) 事例に対する LSN が削除された場合，3) 事例に対する LSN が根ノードの

場合，に枝刈りを行うこととしている．

二分木に含まれる全ノードを枝刈り候補とすることも考えられるが，ノード数に比例して枝刈り処理に要する計算時間が増加するため関数 pr_nodes により枝刈り候補ノードを絞っている．現状では RDR システムにより事例を分類した際の推論パス上のノードのうち，YES 枝に子ノードを持たないノードを候補としている．理由は，関数 $delete\_node$ により二分木を再編した際，削除したノードが YES 枝を持つと図 5 に示すように再編後の二分木が NO 枝に偏りがちになるため，枝刈りによって二分木が偏ることを防ぐためである．

なお，知識獲得やノード削除では式 (1) における重み w を 0.3 としており，枝刈りでは予備実験の結果に基づいて w=0.7 とした．大きな重み $w(0.7>0.3)$ を用いて知識モデル（二分木）の DL に対する影響を大きくして積極的に枝刈りを行うことに対応する．

4. 評 価 実 験

環境変化に対する知識削除機能および枝刈り機能の有効性を検証するため，California 大学 Irvine 校の機械学習データセットライブラリ [Blake 98] のうち 15 データセット（表 2 参照）を用いて評価実験を行った．環境変化をシミュレートする人工データを生成し，提案手法を用いて構築した RDR システムの予測精度と NDL を評価した．

4·1 実 験 条 件

[人工データ] ある環境に対応する事例集合 X_{org} に対し，異なる環境に対応する事例集合 X_{chg} を以下の操作で生成した．まず X_{org} に含まれる全事例をクラスラベルに関して辞書順にソートし，次にソート後の順序を保ったまま同じクラスラベルを持つ事例を属性-属性値のペアに関して離散属性は辞書順，数値属性は昇順にソートした．最後に，x%分の事例でクラスラベルを変化させるために（全事例数 ÷ クラス数 ÷$(100\div x)$）の事例分だけクラスラベルをスライドさせ，事例集合 X_{chg} を生成した．X_{chg} は X_{org} から x%の事例のクラスラベルが変化した環境に対応する．

[訓練データと評価データ] 事例集合 X を 75%の訓練データ（X^{train}）と 25%の評価データ（X^{test}）に分割した．ある環境を表す事例集合 X の下での知識獲得では，X^{train} を母集団として指定した事例数分だけ復元抽出により X^{train} から事例を無作為抽出して逐次的に RDR システムを構築した．

[環境変化] 抽出事例から逐次的にシステムを構築する過程で環境が二度変化する状況を想定し，各データセットごとに元の事例集合 X_{org} から環境が変化した二つの事例集合（X_{chg1}，X_{chg2}）を生成し，X_{chg2} → X_{chg1} → X_{org} と変化した場合と X_{org} → X_{chg1} → X_{chg2} と変化した場合を評価した．環境変化の割合は X_{chg1} では 10%，X_{chg2} では 20%とした．環境 X からの抽出事例数を $|X^{train}|$ の 3 倍とし，たとえば X_{chg2} からの抽出事例数が $3|X^{train}_{chg2}|$ に達した時点で事例集合を X_{chg1} に変更した．実験で使用した環境変化を図 7 に示す．図 7 ではクラスラベルを濃淡で表示し，環境変化のたびにクラス分布における部分集合に対応するクラスラベルが変化することを示す．

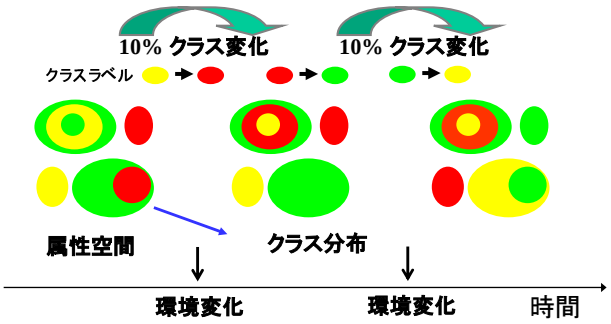

図 7 実験で使用した環境変化

[代替専門家] 実験の再現性と性能評価の必要性から，従来研究では計算機プログラムで構築した代替専門家（Simulated Expert: SE）[Compton 95] から知識獲得を行い性能評価しているため，本稿でも代替専門家を用いて評価した．環境を反映する事例集合が X である際には X^{train} に対して機械学習法 C4.5rules[Quinlan 93] を用いて導出した If-Then ルール集合を SE の知識源とした．SE が D-List から選択する条件は，RDR システムが誤推論した事例を正しく分類する SE の If-Then ルール（これを発火した If-Then ルールと呼ぶ）の条件部と D-List の共通部分とした．共通部分が無い場合は SE の知識 (If-Then ルール) を活用できないためデータからの知識獲得を行った．

環境が X_α から X_β に変化した時点で，SE の知識源である If-Then ルール集合を X^{train}_α に対するものから X^{train}_β に対するものに切り替えた．これは，SE は非常に性能の高い専門家であり，環境変化に合わせて自身の知識を直ちに切替え可能であることを意味する．なお C4.5rules で得られる If-Then ルールでは否定条件（not）が表現されないため，SE が否定条件も D-List から選択できるように属性値のバイナリー化を行って SE の If-Then ルールに否定条件が表現されるようにした．

[分類精度] 逐次的に構築した RDR システムのエラー率を評価データに対して計測した．母集団が X^{train}_α である間は X^{test}_α を評価データとして，X^{train}_β に変化した後は X^{test}_β を評価データとした．なお，RDR

表 2 データセット概要

データセット名	事例数	クラス数	属性数	データセット名	事例数	クラス数	属性数
Car	1728	4	Nom.* 6	PageBlocks	5473	5	Num. 10
Nursery	12960	5	Nom. 8	PenDigits	10992	10	Num. 16
Mushrooms	8124	2	Nom. 22	Yeast	1484	10	Num. 8
Krvkp	3196	2	Nom. 36	PimaIndians	768	2	Num. 6
VotingRecord	435	2	Nom. 16	GermanCredit	1000	2	Mix.*** 13/7
BreastCancer	699	2	Nom. 9	Cmc	1473	3	Mix. 7/2
Splice	3190	3	Nom. 60	AnnThyroid	7200	3	Mix. 15/6
Image	2310	7	Num.** 19				

* 離散値属性，** 連続値属性，*** 離散値属性と連続値属性が混在：離散値属性の数/連続値属性の数

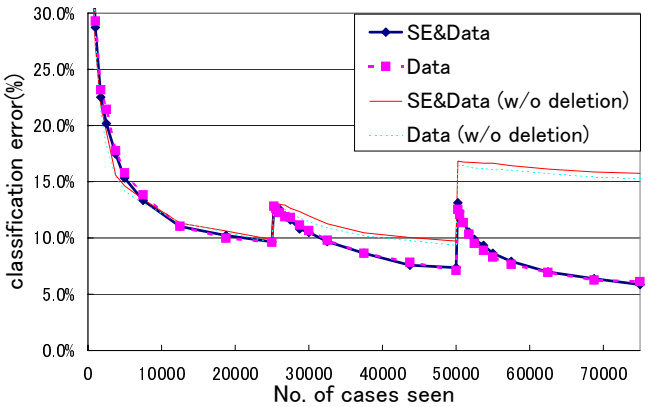

図 8 データセット “PenDigits” に対するエラー率の変化 (10 回試行平均, $X_{chg2} \rightarrow X_{chg1} \rightarrow X_{org}$)

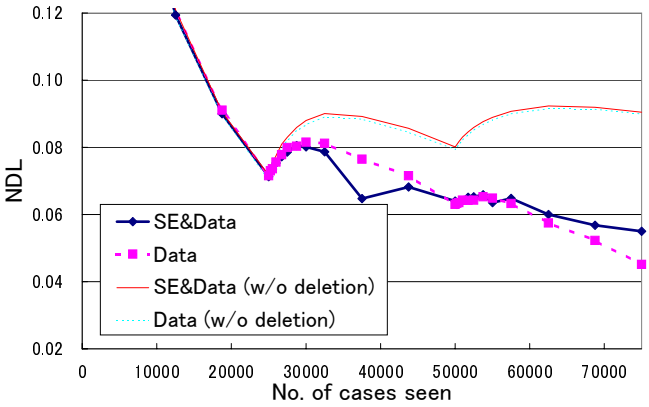

図 9 データセット “PenDigits” に対する NDL の変化 (10 回試行平均, $X_{chg2} \rightarrow X_{chg1} \rightarrow X_{org}$)

法は逐次的な手法であるため，構築した RDR システムの性能は抽出した事例の順序に依存する．このため，各データセットに対して無作為抽出の順序を変えた 10 回試行の平均をエラー率とした．また，削除機能，枝刈り機能を用いる場合と用いない場合の予測精度を対応のある t 検定（片側検定）を用いて有意水準 99%で検定した．

上記の実験条件は環境変化の一例にすぎないが，本稿では実験の再現性と一貫した精度評価のために上記の実験条件を用いることとした．図 7 に示すように，本稿での実験条件はクラス比を固定したまま各属性を次元とする属性空間のある部分空間に対応するクラスラベルが変化するような環境変化に対応する．これは，たとえば組織を部署と部署内の掛の単位で考えた場合に，組織再編により掛の単位で別の部署に人員を移動し，移動した掛を補充するために別の部署に属していた掛を移動することに対応する．この場合，掛に属する個々の構成員の性質は変化しないが掛を組織の一部として含む部署の性質が変化することになる．

4·2 実 験 結 果

データと専門家からの知識獲得，およびデータのみからの知識獲得に対してそれぞれ実験した．$X_{chg2} \rightarrow X_{chg1} \rightarrow X_{org}$ の変化における実験結果の一例を図 8，図 9 に示す．図 8，図 9 はデータセット “Pen-Digits” におけるエラー率，NDL に対する 10 回試行平均の変遷を示す．このデータセットでは 25001 事例目と 50001 事例目で環境が変化し，それに伴いエラー率および NDL は一旦上昇するものの，提案するノード削除機能により RDR システムから無効となった知識（ノード）を削除することによりエラー率および NDL が減少していることがわかる．特に，環境が静的な間 (1~25000 事例まで) は知識獲得に伴い NDL の割合は単調に減少し，環境が変化した時点で NDL は一旦増加し，変化後に RDR に与えられた事例に基づいて知識獲得を行うにつれて再び減少するという変化が 25001 事例目の前後に現れており，ノード削除アルゴリズムが適切に機能したことを示唆している．

各環境 X から無作為抽出した事例数を基準として，変化の直前までに知識ベースに蓄積された事例数とその中で変化後の環境に対しては有効ではない事例数の比を考えると，最初の変化で 10.0%，2 番目の変化で 10.5%[*2] となり，削除機能により矛盾した事例が全て削除されるとは限らないため後者のほうが環境変化の影響はさらに大きくなる．このため，25001 事例目の変化に比して 50001

*2 各環境 X から無作為抽出した事例数を基準 (100%) とすると，理想的には削除機能により X_{chg2} で蓄積した事例のうち X_{chg1} にとって矛盾した 10%の事例が削除され，2 番目の変化の直前では 190%分蓄えられている．その中で，X_{chg2} で蓄積した 10%と X_{chg1} で蓄積した 10%は X_{org} に対して矛盾する事例となるため，比は 20%/190% = 10.5%となる．

表 3 ノード削除を用いた結果（枝刈りあり）

Data Set	データと専門家				データのみ				C4.5 (全事例)
	事例数	NDL	RDR	C4.5 (保持事例)	事例数	NDL	RDR	C4.5 (保持事例)	
Car	92.8%	16.1%	*7.7**%	9.3%	92.1%	16.2%	*8.0**%	9.1%	11.6%
Nursery	91.4%	6.2%	6.2*%	5.3%	91.5%	5.3%	5.9*%	5.2%	8.7%
Mushrooms	92.7%	9.6%	6.6*%	5.2%	91.2%	8.6%	4.5*%	4.4%	12.2%
Krvkp	88.1%	8.1%	*5.0**%	5.2%	87.3%	9.4%	5.4*%	5.3%	10.9%
VotingRecord	85.1%	16.5%	8.0%	5.2%	86.1%	19.3%	6.7*%	6.0%	7.3%
BreastCancer	89.9%	12.2%	*4.8**%	4.9%	89.5%	12.0%	5.5*%	4.5%	9.1%
Splice	87.6%	10.1%	11.1*%	10.3%	89.1%	11.6%	12.4*%	11.4%	17.4%
Image	90.7%	6.6%	*4.9**%	5.1%	89.7%	7.0%	*4.9**%	5.1%	14.1%
PageBlocks	84.5%	19.3%	*5.6**%	6.8%	84.9%	21.1%	6.0*%	6.0%	9.4%
PenDigits	90.1%	5.5%	5.9*%	5.8%	90.6%	4.5%	6.1*%	5.9%	14.2%
Yeast	60.4%	28.4%	39.1%	34.4%	61.8%	27.2%	38.9%	34.1%	17.7%
PimaIndians	44.9%	39.5%	28.8%	24.4%	45.5%	45.8%	29.8%	24.6%	16.7%
GermanCredit	63.7%	42.3%	24.2%	23.7%	73.8%	43.0%	21.0%	17.9%	13.0%
Cmc	46.3%	49.8%	49.0%	43.9%	33.5%	52.9%	50.2%	48.6%	23.3%
AnnThyroid	87.8%	17.2%	*2.4**%	3.7%	86.3%	17.8%	*3.5**%	4.4%	10.3%

*: RDR のエラー率 < C4.5（全事例）のエラー率, イタリック体: RDR のエラー率 < C4.5（保持事例）のエラー率

表 4 ノード削除を用いた結果（枝刈りなし）

Data Set	データと専門家				データのみ				C4.5 (全事例)
	事例数	NDL	RDR	C4.5 (保持事例)	事例数	NDL	RDR	C4.5 (保持事例)	
Car	95.8%	23.8%	13.0%	10.0%	95.8%	29.9%	13.9%	10.3%	11.6%
Nursery	95.8%	13.2%	10.4%	7.1%	95.2%	14.2%	9.7%	6.5%	8.7%
Mushrooms	93.0%	10.8%	*4.3**%	5.2%	94.6%	12.9%	*4.9**%	6.9%	12.2%
Krvkp	91.1%	11.7%	*4.8**%	5.8%	90.7%	13.2%	6.0*%	5.4%	10.9%
VotingRecord	89.6%	24.8%	9.9%	6.2%	88.6%	25.4%	8.2%	5.5%	7.3%
BreastCancer	91.2%	15.8%	5.4*%	5.4%	92.0%	16.0%	5.4*%	5.4%	9.1%
Splice	85.4%	16.3%	12.5*%	12.8%	83.3%	25.9%	16.7*%	13.0%	17.4%
Image	95.3%	36.1%	12.1*%	10.3%	95.7%	38.0%	12.7*%	11.1%	14.1%
PageBlocks	95.4%	32.3%	*8.0**%	8.2%	95.2%	34.8%	*8.2**%	8.6%	9.4%
PenDigits	95.3%	22.3%	12.4*%	11.4%	95.4%	22.1%	12.5*%	12.0%	14.2%
Yeast	96.6%	35.6%	18.3%	17.5%	96.8%	36.2%	18.9%	17.8%	17.7%
PimaIndians	81.4%	46.2%	16.4*%	15.9%	80.8%	45.9%	16.3*%	16.1%	16.7%
GermanCredit	83.7%	45.4%	15.7%	16.0%	83.8%	46.6%	15.8%	15.0%	13.0%
Cmc	87.9%	43.3%	26.0%	24.6%	89.3%	44.9%	25.5%	23.6%	23.3%
AnnThyroid	92.9%	30.1%	8.0*%	6.7%	93.0%	32.8%	8.2*%	6.5%	10.3%

*: RDR のエラー率 < C4.5（全事例）のエラー率, イタリック体: RDR のエラー率 < C4.5（保持事例）のエラー率

事例目の変化が多少影響が大きかったと考えられる．

図 8, 図 9 に対応する，$X_{org} \rightarrow X_{chg1} \rightarrow X_{chg2}$ の変化における実験結果の一例を図 10, 図 11 (データセット “Pen-Digits” におけるエラー率，NDL に対する 10 回試行平均の変遷) に示す．この実験においても，図 8, 図 9 と同様なノード削除機能の効果が見られた．$X_{chg2} \rightarrow X_{chg1} \rightarrow X_{org}$ での実験と $X_{org} \rightarrow X_{chg1} \rightarrow X_{chg2}$ での実験ではほぼ同様な結果が得られたため，以下では前者の実験結果に限定して議論する．

15 種類のデータセットに対し，各試行終了時に構築された RDR システムの評価（X_{org}^{test} に対するエラー率 (%) と NDL の 10 回試行における平均値）を表 3，表 4，表 5 に示す．表 3，表 4 は 3･1 節で提案したノード削除機能を用いた場合（表 3 は枝刈り機能を用いた場合，表 4 は枝刈り機能を用いない場合），表 5 はノード削除機能を用いない場合を示す．各表で “NDL”，“RDR” の列は構築した RDR システムの正規化した記述長およびエラー率を示し，“C4.5（全事例）” の列は全ての訓練事例に対して C4.5 を用いて構築した決定木のエラー率を示す．表 3，表 4 の “事例数” の列は RDR システムが各試行終了時に保持していた事例数の全訓練事例数に対する割合を示し，“C4.5（保持事例）” の列は保持された事例集合に対して C4.5 を用いて構築した決定木のエラー率を示す．また，表 6 に提案した削除機能，枝刈り機能を用いた場合を各バリエーションと対比した場合の p 値を示す．表 6 中の $\leq 0.01^{+}$ は有意水準 99%以上で削除機能，枝刈り機能を用いた場合にエラー率が低い結果であったことを示す（逆に，$\leq 0.01^{-}$ はエラー率が高い結果であった

表 5 ノード削除を用いない結果

Data Set	データと専門家				データのみ				C4.5
	枝刈りあり		枝刈りなし		枝刈りあり		枝刈りなし		
	NDL	RDR	NDL	RDR	NDL	RDR	NDL	RDR	(全事例)
Car	23.5%	13.6*%	27.9%	15.3%	23.9%	13.8*%	34.0%	16.3%	11.6%
Nursery	11.2%	11.4*%	16.8%	13.6%	11.1%	11.4*%	18.0%	13.9%	8.7%
Mushrooms	19.6%	11.5*%	20.2%	11.8%	19.5%	12.2%	20.5%	11.5%	12.2%
Krvkp	23.1%	12.8*%	24.7%	16.0%	23.3%	13.9*%	27.0%	15.7%	10.9%
VotingRecord	34.2%	11.8%	38.0%	11.7%	34.0%	12.1%	38.6%	12.1%	7.3%
BreastCancer	27.8%	12.3%	30.2%	11.8%	27.7%	11.9*%	31.7%	12.1%	9.1%
Splice	19.1%	17.7*%	32.3%	20.0%	19.4%	17.4*%	37.5%	21.0%	17.4%
Image	11.4%	16.0*%	40.3%	18.0%	11.6%	16.7*%	44.0%	17.9%	14.1%
PageBlocks	30.4%	11.4%	40.7%	11.2%	30.3%	11.3%	41.7%	11.2%	9.4%
PenDigits	9.1%	15.7*%	26.7%	17.5%	9.0%	15.3*%	26.4%	17.6%	14.2%
Yeast	26.4%	26.3%	38.0%	20.7%	26.3%	25.0%	38.6%	20.8%	17.7%
PimaIndians	45.8%	21.6%	53.2%	16.4%	45.4%	18.3%	53.1%	17.5%	16.7%
GermanCredit	45.4%	18.8%	53.5%	14.0%	45.7%	18.2%	54.6%	14.1%	13.0%
Cmc	49.7%	35.5%	49.3%	27.5%	49.3%	34.1%	50.7%	27.1%	23.3%
AnnThyroid	39.1%	10.9*%	42.4%	11.5%	38.8%	10.1*%	44.5%	11.1%	10.3%

*: 枝刈り機能を用いたエラー率が低い場合

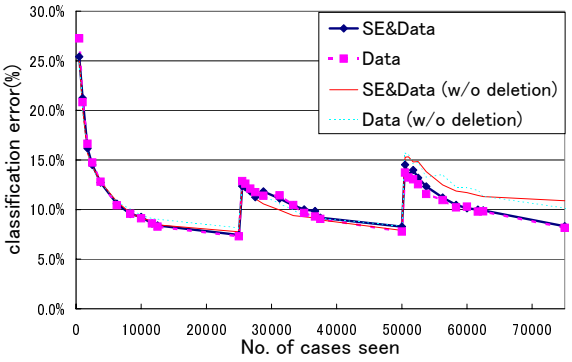

図 10 データセット “PenDigits” に対するエラー率の変化 (10 回試行平均，$X_{org} \rightarrow X_{chg1} \rightarrow X_{chg2}$)

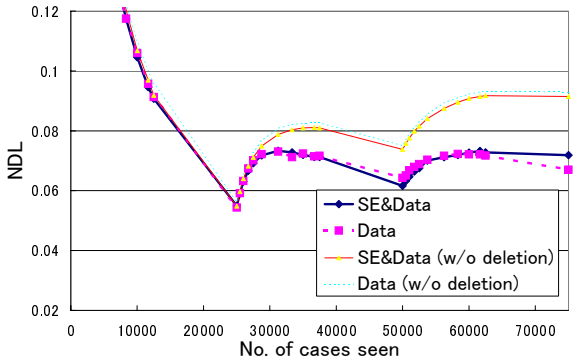

図 11 データセット “PenDigits” に対する NDL の変化 (10 回試行平均，$X_{org} \rightarrow X_{chg1} \rightarrow X_{chg2}$)

ことを示す）．たとえば，表 6 の第 1 列目は，専門家とデータからの知識獲得において削除あり，枝刈りありの場合と削除なし，枝刈りなしの場合を比較した場合の p 値を示し，データセット Car では 99%有意水準で前者のエラー率が有意に低かったことを示す．

削除機能および枝刈り機能を用いて構築した RDR システム（表 3）と両機能を用いずに構築した RDR システム（表 5 右側）を比較すると，データと専門家からの知識獲得およびデータのみからの知識獲得に対して両機能を用いた場合が 11 データセットに対して有意にエラー率が低かった．このため，提案した削除機能および枝刈り機能は環境変化に対して有効に機能したと考えられる．更に，“C4.5（全事例）” と比較しても，データのみから逐次的に知識獲得して構築した RDR システムのほうが 10 データセットに対してエラー率が有意に低く，RDR システムのほうがエラー率が有意に高くなることはなかった．上記より，環境変化に対して提案手法は有効に機能するといえる．

無効となった知識を保持するノードのみを削除することの効果に対して，RDR システムが保持する事例集合を C4.5 を用いて構築した決定木のエラー率の観点から評価した．“C4.5（保持事例）” と “C4.5（全事例）” と比較すると，前者に対するエラー率が 11 データセットに対して有意に低かった．このため，環境変化に伴い無効となった知識を保持し続けると知識ベースシステムの性能が劣化するという課題に対し，提案手法では無効な知識を保持するノードが削除され，有効な知識を保持するノードが知識ベースに保存されていることがわかる．

残念ながら 4 データセット (“Yeast”, “PimaIndians”, “GermanCredit”, “Cmc”) に対してはノード削除機能および枝刈り機能を用いた場合にエラー率は悪化した．表 5 でこの 4 データセットはノード削除を行わない場合でも枝刈りを行うと予測精度が悪化しているため，枝刈りが有効に機能しにくい性質を持つと考えられる．表 3 の “事例数” の列を見ると，他のデータセットと比較して保持事例の割合が非常に低くなっており，削除機能と枝刈

表 6 対応のある t 検定の結果 (片側検定での p 値)

	データと専門家（削除あり，枝刈りあり）				データのみ（削除あり，枝刈りあり）			
	枝刈りなし		C4.5		枝刈りなし		C4.5	
	削除なし	削除あり	(全事例)	(保持事例)	削除なし	削除あり	(全事例)	(保持事例)
Car	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	0.0844^{+}
Nursery	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	0.0408^{-}	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	0.1554^{-}
Mushrooms	$\leq 0.01^{+}$	0.0106^{-}	$\leq 0.01^{+}$	$\leq 0.01^{-}$	$\leq 0.01^{+}$	0.3625^{+}	$\leq 0.01^{+}$	0.4474^{-}
Krvkp	$\leq 0.01^{+}$	0.4207^{-}	$\leq 0.01^{+}$	0.3638^{+}	$\leq 0.01^{+}$	0.2346^{+}	$\leq 0.01^{+}$	0.3880^{-}
VotingRecord	0.0338^{+}	0.1285^{+}	0.3346^{-}	$\leq 0.01^{-}$	$\leq 0.01^{+}$	0.2160^{+}	0.3250^{+}	0.2790^{-}
BreastCancer	$\leq 0.01^{+}$	0.1630^{+}	$\leq 0.01^{+}$	0.4615^{+}	$\leq 0.01^{+}$	0.4450^{-}	$\leq 0.01^{+}$	0.0955^{-}
Splice	$\leq 0.01^{+}$	0.1408^{+}	$\leq 0.01^{+}$	0.0519^{-}	$\leq 0.01^{+}$	0.0109^{+}	$\leq 0.01^{+}$	0.1203^{-}
Image	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	0.2717^{+}	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	0.3297^{+}
PageBlocks	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	0.0215^{+}	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	0.4439^{+}
PenDigits	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	0.4845^{-}	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	0.2105^{-}
Yeast	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$
Pima	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$
GermanCredit	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	0.3662^{-}	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	0.0384^{-}
Cmc	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	$\leq 0.01^{-}$	0.1854^{-}
AnnThyroid	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	0.0261^{+}	$\leq 0.01^{+}$	$\leq 0.01^{+}$	$\leq 0.01^{+}$	0.1361^{+}

$^{+}$: 削除機能，枝刈り機能を用いた場合のエラー率が低い
$^{-}$: 削除機能，枝刈り機能を用いた場合にエラー率が高い
$\leq 0.01^{+}$: 有意水準 99%以上で削除機能，枝刈り機能を用いた場合のエラー率が低い
$\leq 0.01^{-}$: 有意水準 99%以上で削除機能，枝刈り機能を用いた場合のエラー率が高い

り機能を組み合わせて用いた場合に枝刈りによる二分木の再編が過剰なノード削除を誘引したと考えられる．表 3 と表 4 を比較すると，この 4 データセットに対しては枝刈りを用いずにノード削除を行った場合に保持事例の割合が増加し，エラー率も減少している．また，表 4 で “RDR” と “C4.5（全事例）” を比較すると，枝刈りを用いずにノード削除のみを行った場合でもエラー率は有意に低くなった．上記の結果より，データセットによっては削除機能と枝刈り機能を組み合わせると性能（エラー率）が悪化する場合もあるが，その場合は枝刈りを行わないことで対処できると考えられる．

同じ保持事例に対してであっても，逐次的に知識獲得を行う RDR 法のほうがバッチ処理的な C4.5 よりも低いエラー率を示したデータセットがあることは注目に値する．たとえば，表 3 ではデータと専門家からの知識獲得に対しては RDR システムのほうが 5 データセットに対してエラー率が低く，データのみからの知識獲得に対しては 2 データセットに対してエラー率が低かった．また，エラー率と NDL がほぼ同期しており，NDL が低い場合にはエラー率も低くなっている．上記の結果より，NDL に基づいて RDR システムを構築することは環境変化への対応を実現するうえで有効であるといえる．

表 5 で枝刈り機能を用いた場合と用いない場合の RDR のエラー率を比較すると，データと専門家からの知識獲得とデータのみからの知識獲得でともに * をつけた 8 データセットで枝刈り機能を用いた場合にエラー率が低かった．また，表 6 で削除機能および枝刈り機能を用いた場合と削除機能のみを用いた場合を比較すると，データと専門家からの知識獲得とデータのみからの知識獲得でともに 6 データセット（表 6 で $\leq^{+}$）に対して枝刈り機能も併用したほうが有意にエラー率が低かった．汎化による予測精度の向上が枝刈りの目的であるが，環境変化への対応に関しても有効に機能するといえる．

5. お わ り に

本稿では環境変化の一例として同じ属性-属性値の組合せを持つ事例に対するクラスラベルの変化を考え，以前正しいと判断され蓄積された知識が環境変化に伴って有効性を失うことに対応するために，知識削除機能と知識汎化機能を導入して RDR 法の知識ベースシステムを構築する手法を提案した．[和田 01b] で提案した最小記述長原理を用いて専門家からの知識獲得とデータからの帰納学習手法を RDR 法に統合化する手法を拡張し，知識の追加，削除，汎化に伴う RDR 法の知識ベースシステムの再編を正規化した記述長の最小化という規範で実現できることを示した．15 種類のデータセットに対する人工データを用いた実験から，提案する両機能を用いた場合に知識ベースシステムの性能が向上することを確認した．また，対象データの性質によっては知識汎化機能を用いると逆に性能が悪化することがあるという課題も明らかになった．

今後の課題として以下の 3 点に取り組む．

1) RDR 法の二分木を再編すると，ノード削除前後で知識の整合性は維持されるものの，削除後の二分木が NO 枝に偏りがちになるという課題がある（関数 *delete_node*

を用いた二分木の再編）．もともと RDR 法の二分木は知識獲得の過程でも例外知識に対応するノードの追加により NO 枝に偏る傾向があるため，ノードの削除により更に木構造が偏ってしまうことを防ぐようなノード削除方法を検討する．

2) 1) の課題に対処するため，枝刈り対象候補ノードをヒューリスティックな方法で選択していることに対し，枝刈りが不必要なノードを記述長の観点から理論的に検討し，必要十分なノードだけを候補とする方法を検討する．

3) 全体の記述長を計算する際に用いる重みを，経験的に知識獲得およびノード削除では 0.3，枝刈りでは 0.7 と設定している．しかし，真に適切な符号化方法を採用すればこの重みは必要ないはずである．提案手法の効果が見られたデータセットの特性と見られなかったデータセットの特性の違いをもとに符号化の方法と重みの関係をより深く検討し，重みに鈍感な（あるいは重みを不要とする）符号化方法を導入することで，データセットの特性の違いにも頑強な提案手法を実現する．

謝　　辞

本研究の一部は文部科学省科研費特定領域研究「情報洪水時代におけるアクティブマイニングの実現」(No.13131101, No.13131206) の補助による．最後に，有益なご指摘を賜りました査読者の方々に深く謝意を表します．

◇ 参 考 文 献 ◇

[Blake 98] Blake, C. and Merz, C.: UCI Repository of machine learning databases (1998), http :// www.ics.uci.edu / ∼mlearn / MLRepository.html

[Compton 89] Compton, P., Horn, K., Quinlan, J., and Lazarus, L.: Maintaining an Expert System, in Quinlan, J. ed., *Application of Expert Systems*, pp. 366–385, Addison Wesley (1989)

[Compton 95] Compton, P., Preston, P., and Kang, B.: The Use of Simulated Experts in Evaluating Knowledge Acquisition, in *Proc. of the 9th Knowledge Acquisition for Knowledge Based Systems Workshop* (1995)

[Morik 93] Morik, K., Wrobel, S., Kietz, J., and Emde, W. eds.: *Knowledge Acquisition and Machine Learning: Theory, Methods, and Applications*, Academic Press (1993)

[Quinlan 93] Quinlan, J. ed.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann (1993)

[Rissanen 78] Rissanen, J.: Modeling by Shortest Data Description, *Automatica*, pp. 465–471 (1978)

[和田 00] 和田 卓也, 堀内 匡, 元田 浩, 鷲尾 隆：Ripple Down Rules 法における知識獲得の特性評価に基づくデフォルト知識の決定規範, 人工知能学会誌, Vol. 15, No. 1, pp. 177–186 (2000)

[Wada 01a] Wada, T., Motoda, H., and Washio, T.: Knowledge Acquisition from Both Human Expert and Data, in *Proc. of the Fifth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2001)*, pp. 550–561, HongKong China (2001), Springer-Verlag

[和田 01b] 和田 卓也, 元田 浩, 鷲尾 隆：最小記述長原理を用いた帰納学習の Ripple Down Rules 法への統合化, 人工知能学会誌, Vol. 16, No. 2, pp. 268–278 (2001)

[Wrobel 94] Wrobel, S. ed.: *Concept Formation and Knowledge Revision*, Kluwer Academic Publishers (1994)

〔担当委員：津本周作〕

2003 年 10 月 23 日　受理

著　者　紹　介

吉田　哲也（正会員）

1991 年東京大学工学部航空工学科卒業．1992 年から 1993 年にかけてエジンバラ大学大学院留学．1994 年から 1995 年にかけてカリフォルニア大学バークレー校交換留学．1997 年東京大学大学院博士課程修了．工学博士．同年，大阪大学大学院基礎工学研究科助手．現在，北海道大学大学院情報科学研究科助教授．主に機械学習，知識獲得，データマイニングなどの研究に興味を持つ．人工知能学会，情報処理学会，ヒューマンインタフェース学会，各会員．

和田　卓也（正会員）

1998 年大阪大学工学部通信工学科卒業．2002 年同大学院博士後期課程修了．現在，藤沢薬品工業株式会社勤務．人間からの知識獲得とその利用法に関する研究に興味を持つ．人工知能学会会員．

元田　　浩（正会員）

1965 年東京大学工学部原子力工学科卒業．1967 年同大学院原子力工学専攻修士課程終了．同年，日立製作所に入社．同社中央研究所，原子力研究所，エネルギー研究所，基礎研究所を経て平成 7 年退社．現在，大阪大学産業科学研究所教授 (知能システム科学研究部門、高次推論研究分野)．原子力システムの設計，運用，制御に関する研究，診断型エキスパート・システムの研究を経て，現在は人工知能の基礎研究，とくに機械学習，知識獲得，知識発見，データマイニングなどの研究に従事．工学博士．日本ソフトウェア科学会理事，人工知能学会理事，同編集委員会委員，日本認知科学会編集委員会委員，Knowledge Acquisition (Academic Press) 編集委員，IEEE Expert 編集委員を歴任．Artificial Intelligence in Engineering (Elsevier Applied Science) 編集委員，International Journal of Human-Computer Studies (Academic Press) 編集委員，Knowledge and Information Systems: An International Journal (Springer-Verlag) 編集委員．1975 年日本原子力学会奨励賞，1977，1984 年日本原子力学会論文賞，1989，1992 年人工知能学会論文賞受賞．1997 年人工知能学会研究奨励賞受賞，1997, 1998 年人工知能学会全国大会優秀論文賞受賞．人工知能学会，情報処理学会，日本ソフトウェア科学会，日本認知科学会，AAAI，IEEE Computer Society，各会員．

鷲尾　　隆（正会員）

1960 年生．1983 年東北大学工学部原子核工学科卒業．1988 年東北大学大学院原子核工学専攻博士課程修了．工学博士．1988 年から 1990 年にかけてマセチューセッツ工科大学原子炉研究所客員研究員．1990 年（株）三菱総合研究所入社．1996 年退社．現在，大阪大学産業科学研究所助教授（知能システム科学研究部門）原子力システムの異常診断手法に関する研究，定性推論に関する研究を経て，現在は人工知能の基礎研究，特に科学的知識発見，データマイニングなどの研究に従事．1988 年 2 月計測自動制御学会学術奨励賞受賞，1995 年 8 月人工知能学会全国大会優秀論文賞受賞，他 2 件．1996 年 3 月日本原子力学会論文賞受賞，1996 年 12 月人工知能学会研究奨励賞受賞，他 1 件．著書に “*Expert Systems Applications within the Nuclear Industry*”, American Nuclear Society,「知能工学概論」：第 2 章エージェント（共著，廣田　薫　編，昭晃堂）など．AAAI，人工知能学会，計測自動制御学会，情報処理学会，日本ファジイ学会，各会員．