Adaptive Ripple Down Rules Method based on Minimum Description Length Principle

Tetsuya Yoshida
I.S.I.R., Osaka University
8-1 Mihogaoka, Ibaraki, 560-0047, Japan
yoshida@ar.sanken.osaka-u.ac.jp

Takuya Wada*
I.S.I.R., Osaka University
8-1 Mihogaoka, Ibaraki, 560-0047, Japan
wada@ar.sanken.osaka-u.ac.jp

Hiroshi Motoda
I.S.I.R., Osaka University
8-1 Mihogaoka, Ibaraki, 560-0047, Japan
motoda@ar.sanken.osaka-u.ac.jp

Takashi Washio
I.S.I.R., Osaka University
8-1 Mihogaoka, Ibaraki, 560-0047, Japan
washio@ar.sanken.osaka-u.ac.jp

Abstract

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 both the change in class distribution on a domain and the change in knowledge source as typical changes in the environment. When class distribution changes, 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 cope with the change in knowledge source, RDR knowledge based systems can be constructed adaptively by acquiring knowledge from both domain experts and data. By incorporating inductive learning methods, knowledge can be acquired (learned) even when only either data or experts are available by switching the knowledge source from domain experts to data and vice versa at any time of knowledge acquisition. Since experts need not be available all the time, it contributes to reducing the cost of personnel expenses. Experiments were conducted by simulating the change in knowledge source and the change in class distribution using the datasets in UCI repository. The results show that it is worth following this path.

Correspondence to:

Tetsuya Yoshida
I.S.I.R., Osaka University
8-1 Mihogaoka, Ibaraki, Osaka 560-0047, Japan
tel: +81-6-6879-8542, fax: +81-6-6879-8544
yoshida@ar.sanken.osaka-u.ac.jp

*Currently with Fujisawa Pharmaceutical Co.LTD

1 Introduction

An implicit assumption of the development of Knowledge-Based System (KBS) is that the expert knowledge is stable and that it is worth investing into knowledge acquisition from human experts. However, the rapid innovation in technology in recent years makes existing knowledge out-of-date very soon and requires frequent updates. We should now think of expertise not as static but as dynamic in many real world problems. In addition, the advancement of worldwide networks such as the Internet enables the communication of huge amount of data in various forms through the network. Receiving up-to-date data over the network can also induce changes of a concept description in the domain from which knowledge is acquired. Thus, it is not appropriate to assume that knowledge is stable and maintains its usefulness all the time. Old useless knowledge in the KBS might hinder efficient acquisition of new knowledge and thus can be harmful. It is necessary not only to incorporate new knowledge into a KBS but also to discard obsolete knowledge from the KBS so that the KBS can adapt to a dynamic environment.

Under these circumstances, a new methodology for constructing a KBS for continuous change is in great demand [13, 20]. "Ripple Down Rules (RDR)" method [3], including "Multiple Classification RDR (MCRDR) method"[11] [1] , is a promising approach to making such knowledge based systems a reality and to directly acquiring and encoding knowledge from human experts. RDR and MCRDR are performance systems [2] that don't require high level models of knowledge as a prerequisite to knowledge acquisition, and have been shown to be very effective in knowledge maintenance as well as in acquisition of reflexive knowledge. Knowledge is incrementally acquired from human experts and there is no clear distinction between knowledge acquisition and knowledge maintenance.

RDR exclusively relies on knowledge acquired from human experts, and one of the advantages is that it does not, as many machine learning techniques do, require any data statistics. However, even human experts can make mistakes. Since huge amount of data are getting stored in databases these days, use of these accumulated data becomes even more important and would enhance RDR's performance if its amount is huge. It is difficult for human experts and knowledge engineers to process data manually. Utilizing machine learning methods is an effective approach to discover new knowledge from accumulated data automatically. On the other hand, human experts are capable of capturing valuable insights and using them in problem solving. which is difficult for machines. Thus, it is important to devise a methodology for constructing a KBS which can exploit the information processing capability of both human experts and machines. In addition, although past researches on RDR assume that the problem domain is stable, this is not the case in reality, especially when an environment changes. Things change over time. Thus, it is also important that the methodology enables the adaptation of the constructed KBS for the change in concept description.

This paper proposes an adaptive RDR method based on the Minimum Description Length Principle (MDLP) [16, 10] aiming at incremental knowledge acquisition when an environment changes. In our approach construction of an effective RDR knowledge based system is based on the notion of searching for the one with smaller description length. To cope with the change in class distribution on a problem domain, 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. In addition, a KBS can be constructed adaptively by acquiring knowledge from both domain experts and data so that the knowledge source can be switched adaptively from domain experts to data and vice versa at any time during the construction of a KBS. It is possible that an expert is involved only at an initial phase of the construction of a KBS when there are not enough data available and a human expert is the sole knowledge source. Later, the knowledge source is switched to data so that the KBS is constructed inductively only from data afterward without rebuilding the KBS from scratch. Being able not to rely on a human expert at all times during the construction of a KBS will contribute to reducing the cost of personnel expenses.

Inducing the RDR knowledge base from data has been studied by Gains and Compton [8, 9]. They used Induct [6, 7] in which the basic algorithm is to search for the premise for a given conclusion that is least likely to predict that conclusion by chance. Their method, although it produces the RDR knowledge base, does not seem to have a common strategy by which a knowledge acquisition approach and a machine

[1] In this paper, we focus on RDR.

[2] Here, the word "performance system" means that the main objective of the system is to attain high performance and not to model how an expert solves a problem. In this paper, performance refers to the predictive accuracy of class label of unseen data.

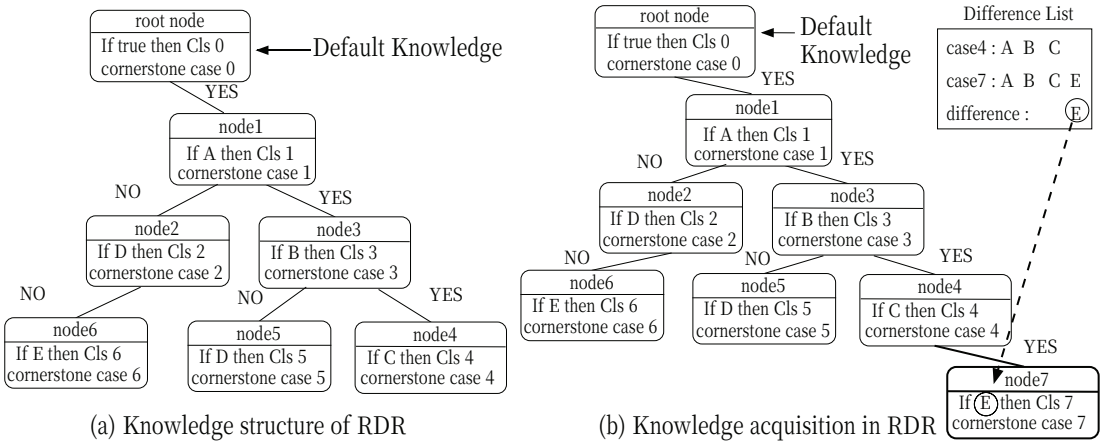

Figure 1: Knowledge structure of the Ripple Down Rules method

learning approach are integrated. In addition, Induct assumes that a KBS is constructed in batch, not incrementally. In our approach knowledge acquisition from experts and inductive learning from data are integrated uniformly based on MDLP. Furthermore, knowledge is incrementally acquired into a KBS, and is even deleted from the KBS, by reflecting the change in the environment. Utilization of MDLP in RDR was studied in our previous approach to characterize the default knowledge [18]. This paper further extends the use of MDLP in RDR to incrementally add and delete knowledge to construct an effective RDR knowledge base.

Knowledge dealt with in RDR is classification knowledge (relationship between attribute-value pairs and class label), and a piece of knowledge in a rule format is acquired and stored at a node in a binary tree of RDR incrementally within the framework of supervised learning. When the class distribution has changed in accordance with concept drift, the proposed method discards nodes which store obsolete knowledge from a binary tree while maintaining other nodes which still store valid knowledge. Thus, our methods enables the adaptation to concept drift by adding and deleting nodes within the framework of incremental knowledge acquisition in RDR.

This paper is organized as follows. Section 2 briefly explains the Ripple Down Rules (RDR) method. Section 3 explains the use of Minimum Description Length Principle (MDLP) in RDR and how the description length is calculated in our approach. Section 4 describes the details of the proposed adaptive RDR method, followed by Section 5 in which experimental results to verify the proposed method are reported. Finally, Section 6 gives a short conclusion and indicates future directions.

2 Ripple Down Rules

Ripple Down Rules (RDR) is a knowledge acquisition technique that challenges the *knowledge acquisition bottleneck* problem by allowing rapid development of knowledge bases by human experts without the need for analysis or intervention by a knowledge engineer. From long experience of knowledge-based systems development [3], it is clear that human experts are not good at providing information on how they reach conclusions, rather they can justify that their conclusions are correct [4, 2]. The basis of RDR is the maintenance and retrieval of cases. In RDR the expert provides his/her expertise to justify the system's judgments [12]. The cases and associated justifications (rules) are added incrementally when a case is misclassified in the retrieval process. When a case's class is incorrectly retrieved by an RDR system, the knowledge acquisition (maintenance) process requires the expert to identify how the case stored in the knowledge base differs from the current case. Thus, RDR has reduced knowledge acquisition to a simple task of assigning a conclusion to a case and choosing the differentiating features between the current misclassified case and the previously correctly classified case.

The structure of an RDR knowledge base is shown in Figure 1(a). Each node in the binary tree is a rule with a desired conclusion (If-Then rule). We call the condition part of an If-Then rule at a node as an antecedent and the conclusion part as a consequent. Each node has a "cornerstone case" associated with it, that is, the case that prompted the inclusion of the rule. An inference process for an incoming case starts from the root node of the binary tree. Then the process moves to the YES branch of the

Table 1: An example of cases

ID No.	Att. Swim	Att. Breath	Att. Legs	Class
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	Dog
6	can_not	lung	4legs	Monkey
7	can_not	gill	2legs	Penguin
8	can_not	gill	4legs	Dog
9	can_not	skin	2legs	Dog
10	can_not	skin	4legs	Monkey

present node if the case satisfies the antecedent of the node, and if it doesn't, the process moves to the NO branch. This process continues until there is no branch to move on. The conclusion for the incoming case is given by the consequent of the node in the inference path for the case whose antecedent is lastly satisfied. Note that this node which has induced the conclusion for the case is called "last satisfied node (LSN)".

If the conclusion is different from that judged by an expert, knowledge (new rule) is acquired from the expert and added to the existing binary tree. The knowledge acquisition process in RDR is illustrated in Figure 1(b). To add a new rule into the tree structure, it is necessary to identify the conditions of the rule as well as its location. When the expert wants to add a new rule, there must be a case that is misclassified by a rule in RDR. The system asks the expert to select conditions for the new rule from the "difference list" between two cases: the misclassified case and the cornerstone case. Then, the misclassified case is stored as a refinement case (a new cornerstone case) with the new rule whose antecedent distinguishes these two cases. Depending on whether the LSN is the same as the end node (the last node in the inference path), the new rule and its cornerstone case are added at the end of YES or NO branch of the end node. Knowledge that is already in the KB is never removed or modified. It is simply added by the addition of exception rules. The tree structure of the knowledge and the keeping of cornerstone cases ensure that the knowledge is always used in the same context as when it was added.

3 Minimum Description Length Principle in RDR

Occam's razor, or minimizing the description length, is the normal practice for selecting the most plausible one of many alternatives. Occam's razor prefers the simplest hypothesis that explains the data. Simplicity of hypothesis can be measured by description length (DL), originally proposed by Rissanen [16]. DL can express the complexity of specifying a hypothesis, and the value of DL is calculated as the sum of two encoding costs: one for the hypothesis and the other for cases misclassified by the hypothesis. The MDLP has been used as a criterion to select a good model in machine learning, *e.g.* in decision trees [15], neural networks [10] and Bayesian networks [17].

We illustrate the concept of the MDLP using a communication problem. Let us suppose that both a sender A and a receiver B have the same list of cases in Table 1 except that B does not know their class labels. A communication problem is to send the class label of each case from A to B through a communication path with as few bits as possible. A knowledge-base model such as a decision tree or a binary tree in RDR can be thought to be composed of a splitting method and the representative class label for each split subset of cases. The calculation of DL consists of the following 4 steps.

Step 1 : Split the cases in the list into several subsets according to the attribute values based on some splitting method (model).

Step 2 : Encode the model according to a certain coding method, and send the resultant bit string to B.

Step 3 : Encode the representative class label of each subset, and send the respective bit string to B.

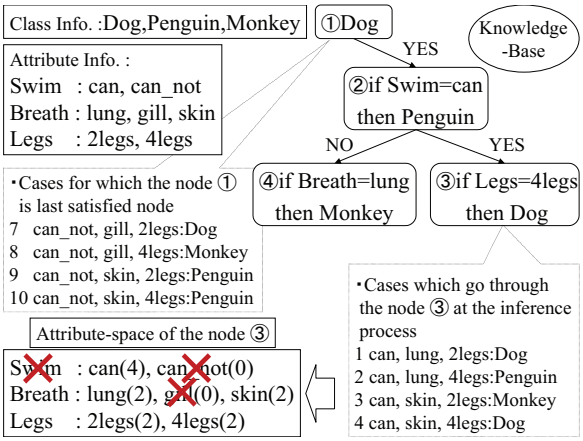

Figure 2: An example of knowledge base in RDR to calculate the DL

Step 4 : Encode the true class label of each misclassified case, which is different from the representative class label, and send the respective bit string to B.

If the splitting method used in **Step 1** is complex, most likely that each DL in **Step 2** and **Step 3** is large, and the DL in **Step 4** is small. In reverse, the DL in **Step 4** will be large if the splitting method is simple. Therefore, there is a trade-off relation between 1) the DL for the splitting method (**Step 2**) and the DL for the representative class labels (**Step 3**) and 2) the DL for the class label of the misclassified cases (**Step 4**). The total DL (the DL for a knowledge-base model + the DL for misclassified cases in the subsets) can be calculated by assuming a certain encoding method. The MDLP says that a knowledge-base model with the smallest total DL predicts the class labels of unseen cases well. In the context of RDR, it is to say that a binary tree with the smallest total DL has the lowest error rate for unseen cases.

One of the differences between the proposed adaptive RDR method and the standard one is that each node in a binary tree of RDR in our approach keeps not only the cornerstone case but also the cases whose LSN is assigned to that node for the calculation of DL. In the next subsections we explain how we encode the DL. In our encoding the DL for the tree is calculated first and then the DL for the misclassified cases is calculated. Note that this is not the only way to encode the DL. There are other ways to do so, but the experimental results show that our encoding method is reasonable. Calculation of DL is illustrated using the RDR knowledge base described in Figure 2 and the set of cases in Table 1. Let us assume that a problem domain has n attributes $\{A_i | i = 1, ..., n\}$ and m_i attribute-values for each nominal attribute A_i. Further assume that the number of different class labels is $class\_num$. Let Q be a set consisting of q cases that has passed a node α in the inference process, and let R be a subset of Q ($R \subseteq Q$), consisting of r cases for which the node α is the LSN.

3.1 The DL for a Binary Tree

A binary tree can be encoded by encoding every node in the tree from the root node downward. Two kinds of information need to be encoded for each node: one for the branches below the node and the other for the If-Then rule that is stored as knowledge in the node. For instance, the branch-information of node No.3 is {YES-branch:no NO-branch:no} (there is no branch to follow below this node) and the rule-information is {If Legs=4legs then Dog}. For a node α, the frequency distribution of each attribute-value in q cases which have passed through the node α is used to obtain the attribute-space of the node α. The attribute-space consists of a set of attributes each having at least 2 attribute-values with its frequency of at least 1 case. The attribute-space of node No.3 in Figure 2 is {Breath:lung,skin, Legs:2legs,4legs}, which is depicted in the lower left-hand side of Figure 2.

There are four branch-combinations for all nodes except for the root node, that is, the number of candidates is 4. Therefore, $\log_2 {}_4C_1$ bits are necessary to encode this information. The DL for the branch-information of node No.3 is also $\log_2 {}_4C_1$ bits.

The information needed to describe the If-Then rule consists of 4 components:

(1) the number of attributes used in the antecedent

(2) attributes used in the antecedent

(3) the attribute value for each used attribute

(4) the class label used in the consequent

In the case of node No.3, (**1**)the number of attributes: 1, (**2**)attributes: Legs, (**3**)the value of Legs: 4legs, (**4**)the class label: Dog. The sum of bits to encode (**1**), (**2**), (**3**) and (**4**) is the DL necessary to encode the If-Then rule information for a node. To encode the information of (**1**) $\log_2 {}_nC_1$ bits are necessary because there are n candidates, {1,2,...,n}. In the case of node No.3 it is $\log_2 {}_2C_1$ because the attribute-space has two attributes. Let the number of attributes used in the antecedent be t. The information of (**2**) can be encoded by $\log_2 {}_nC_t$ bits because the number of combinations of having t 1's and n-t 0's for specifying the used t attributes out of n attributes is given by ${}_nC_t$. For node No.3 it is $\log_2 {}_2C_1$ bits. For each nominal attribute used in (**2**) $\log_2 {}_{m_i}C_1 + \log_2 {}_2C_1$ bits are necessary to encode the information of (**3**). The second term is necessary to specify whether the condition is negated or not for nominal attribute. For node No.3 it is $\log_2 {}_2C_1 + \log_2 {}_2C_1$ bits because the attribute Legs has 2 values. Finally, to encode the information of (**4**) $\log_2 {}_{class\_num-1}C_1$ bits are necessary because the number of class labels that are possible to use as the conclusion in each node except for the root node is $class\_num - 1$. For node No.3 it is $\log_2 {}_2C_1$ bits because the candidates for conclusion are Dog and Monkey.

For a numerical attribute A_i, the condition can be either $\{? < A_i\}$, $\{A_i \leq ?\}$ or $\{? < A_i \leq ?\}$. Thus, $\log_2 {}_3C_1$ bits are necessary to identify which one to use for **(3)**. Suppose that c_i is the number of candidates for a cut-off value for an attribute A_i. When the condition is $\{? < A_i\}$ or $\{A_i \leq ?\}$, another $\log_2 {}_{c_i}C_1$ bits are necessary. On the other hand, when the condition is $\{? < A_i \leq ?\}$, another $\log_2 {}_2C_1$ bits are needed to indicate which of the lower and upper bound is encoded first. In the former case, the upper bound is encoded with $\log_2 {}_{c_i-k_i}C_1$ bits after the lower one is encoded with $\log_2 {}_{c_i}C_1$ bits. In the latter case, the lower bound is encoded with $\log_2 {}_{c_i-l_i}C_1$ bits after the upper one is encoded with $\log_2 {}_{c_i}C_1$ bits. Here, k_i (l_i) means that the lower (upper) cut-off value is the k_i-th (l_i-th) one from the left edge (right edge).

The sum of the DL for the branch-information, which was mentioned in the beginning, and the one for the rule-information is the DL for the node α. In the case of node No.3, it is $\log_2 {}_4C_1 + \log_2 {}_2C_1 + \log_2 {}_2C_1 + \log_2 {}_2C_1 + \log_2 {}_2C_1 = 5\log_2 {}_2C_1 + \log_2 {}_4C_1$ bits. If the sender A encodes the information of all nodes in the tree and send it to the receiver B, B can obtain the used splitting method and the representative class labels (namely, the binary tree itself) by decoding the communicated information.

3.2 The DL for Misclassified Cases

We next explain how to calculate the DL that is necessary to encode the true class information for the cases misclassified by the given binary tree. This DL can be calculated at each node in the tree. Suppose that out of r cases in R for which a node α is their LSNs, k cases has the same class label with the consequent of the rule in the node α. First, $\log_2 {}_rC_1$ bits are necessary to express that $r - k$ cases have different class labels from the consequent. If $k = r$, there is no misclassified case and no further encoding is required. If $k < r$, it is necessary to represent which class label the remaining $r - k$ cases have. Next, it is necessary to calculate the DL for specifying the true class labels of the misclassified cases at the node α. Suppose that the number of class labels which are different from the class label of the cornerstone case is s and the number of cases for each class is $p_i (i = 1, 2, ..., s)$. The class labels are sorted in descending order with p_i, i.e., $p_s \geq p_{s-1} \geq ... \geq p_2 \geq p_1$. The DL for misclassified cases is calculated using the algorithm shown in Figure 3. The function **ceil()** returns the least greatest integer for the argument.

The DL for the i-th different class label is calculated at the line "$DL := DL + \log_2(j) + ...$". The second term is to specify which class label the cases with the i-th class label have, the third one is to specify the number of cases p_i with this class label, and the last one is to specify which of the cases have the class label. Note that the value of p_i is determined by decoding the second term to calculate the last term. For the root node, j is initialized to $class_{num}$, not $class_{num}$ - 1, since the root node does not necessarily store the cases with the class label at the node.

By decoding the bit string with the $\log_2 {}_rC_1$ and the DL returned by the algorithm in Figure 3, it is possible to identify the true class labels for the misclassified $r - k$ cases if the encoded bits are decoded. The remaining k cases have the same class with that of the cornerstone case. Encoding the entire binary

DescriptionLength
initialize
$cases := r$; : the # cases in R
$corrects := k$; : the # of correct cases
$j := class\_num - 1$; : the # of possible class labels
$i := s$; : the # of different class labels
$p_0 := \infty$; : dummy
$p_i^{cand} := 0$; : the possible # of cases for the i-th class label
$p_i^{upper} := \infty$; : the upper bound of p_i
$p_i^{lower} := 0$; : the lower bound of p_i
$DL := 0$; : DL for the misclassified cases
repeat while $(cases \neq corrects)$
if $cases - corrects < p_i^{upper}$
then $p_i^{upper} := cases - corrects$
else $p_i^{upper} := p_{i-1}$;
if $cases - corrects > j$
then $p_i^{lower} :=$ **ceil**$((cases - corrects) \div j)$
else $p_i^{lower} := 0$;
$p_i^{cand} := p_i^{upper} - p_i^{lower}$;
$DL := DL + log_2(j) + \log_2(p_i^{cand}) + \log_2({}_{cases}C_{p_i})$;
$cases := cases - p_i$;
$p_i^{upper} := p_i$;
decrement j and i;
return DL;

Figure 3: Algorithm for calculatig the DL to specify the true class labels of misclassified cases

tree in top-down produces the bit string for the true class labels of the misclassified cases in each node. Note that it is necessary to have the information of the binary tree in Subsection 3.1 beforehand to decode the bit string for the misclassified cases.

3.3 The MDLP for the RDR method

By sending the encoded signal with the total DL (the sum of the DLs mentioned in subsection 3.1 and subsection 3.2), the receiver B is able to find the class label of all cases in the list. Based on the MDLP, the binary tree with the smallest total DL should be most accurate for prediction. However, it is empirically known that most encoding methods tend to overestimate the DL for the knowledge base, compared with the one for the class label of the misclassified cases [14]. Therefore, it is common to use a weighted sum of the two to estimate the total DL:

$$\begin{aligned} \text{Total DL} &= (\textit{DL for the tree in Subsection 3.1}) \times w \\ &\quad + (\textit{DL for misclassified cases in Subsection 3.2}) \end{aligned} \tag{1}$$

In equation (1), w is a coefficient, which is less than 1, and in this paper we empirically found that 0.3 is a good value for w based on our experience [19].

Hereafter, we refer to the total DL for a knowledge base simply as the DL for a knowledge base.

4 Adaptive Ripple Down Rules Method

4.1 Adaptation for the change in Knowledge Source

In addition to knowledge acquisition from human experts in standard RDR, we propose two kinds of knowledge acquisition methods in RDR based on the MDLP. One is for knowledge acquisition only from data and the other is for knowledge acquisition from both data and human experts.

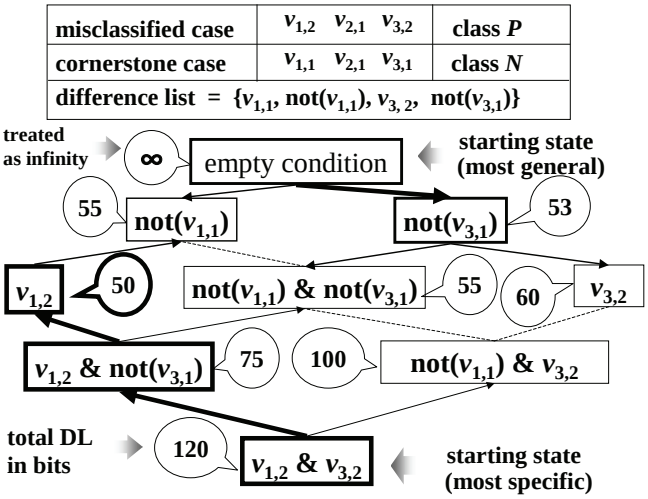

Figure 4: Greedy search by data alone (the values of DLs are imaginary)

4.1.1 Knowledge Acquisition from Data

In the standard RDR, when a case is misclassified by an RDR binary tree, a set of conditions selected from the difference list by a human expert becomes the antecedent of a new node which is added below the end node for the misclassified case, as described in Section 2. However, there are multiple candidates for the condition which distinguishes between the misclassified case and the cornerstone case. Based on the MDLP, we want to select a set of conditions for the antecedent from the difference list in such a way to minimize the DL for the knowledge base.

Let us assume that a problem domain has n attributes $\{A_i | i = 1, ..., n\}$ and m_i attribute-values $\{v_{i,j} | j = 1, ..., m_i\}$ for each attribute A_i. Let the set of attribute-values for A_i as V_i. Let the case misclassified by the so far grown RDR tree as *case a* : $\{v_{1,a_1}, v_{2,a_2}, ..., v_{n,a_n}(v_{i,a_i} \in V_i)\}$, and the cornerstone case stored in the node that has derived the false conclusion (the LSN) as *case b* : $\{v_{1,b_1}, v_{2,b_2}, ..., v_{n,b_n}(v_{i,b_i} \in V_i)\}$. Figure 4 shows an example in which the the misclassified *case a* is $\{v_{1,2}, v_{2,1}, v_{3,2}\}$ and the last satisfied *case b* is $\{v_{1,1}, v_{2,1}, v_{3,1}\}$ [3] . Note that the value of DL in the balloon in Figure 4 is imaginary one for illustration.

The search for the condition with which the knowledge base has a smaller DL is started both from the most specialized condition to the misclassified case and from the most general condition, and is carried out as follows.

Step 1 : Express the difference list between *case a* and *case b* as:

$$\textit{difference list} = \{v_{i,a_i}, not(v_{i,b_i}) | v_{i,a_i} \neq v_{i,b_i}, 1 \leq i \leq n\} \tag{2}$$

(*difference list* = $\{v_{1,2}, not(v_{1,1}), v_{3,2}, not(v_{3,1})\}$ in Figure 4)

Represent the state in search based on the condition for each attribute as:

$$State \equiv \begin{pmatrix} \alpha_1, \alpha_2, ..., \alpha_n \\ \beta_1, \beta_2, ..., \beta_n \end{pmatrix} \tag{3}$$

$$(\alpha_i, \beta_i) \quad = \quad \begin{cases} (0,0)(\text{neither } v_{i,a_i} \text{nor } not(v_{i,b_i}) \text{ is used for the condition of the } i-\text{th attribute}) \\ (0,1)(\text{condition of the } i-\text{th attribute is } not(v_{i,b_i})) \\ (1,0)(\text{condition of the } i-\text{th attribute is } v_{i,a_i}) \\ (1,1)(\text{both } v_{i,a_i} \text{and } not(v_{i,b_i}) \text{ are used for the condition of the } i-\text{th attribute}) \end{cases} \tag{4}$$

In the state representation with (3) and (4), α_i represents the existence of the condition $A_i = v_{i,a_i}$, and β_i represents the existence of the condition $A_i = not(v_{i,b_i})$.

[3] Here, we assume that the value for A_2 is the same for *case a* and *case b* and that the values for A_1 and A_3 are different for these cases.

Step 2 : Start the search for the antecedent with smaller DL from the state of the most specific antecedent. Represent the state of the antecedent which is specialized to the misclassified *case a* as a list of condition:

$$State \equiv \begin{pmatrix} \alpha_1^S, \alpha_2^S, ..., \alpha_n^S \\ \beta_1^S, \beta_2^S, ..., \beta_n^S \end{pmatrix}$$

$$(\alpha_i^S, \beta_i^S) \quad = \quad \begin{cases} (0,0)(v_{i,a_i} = v_{i,b_i}) \\ (1,0)(v_{i,a_i} \neq v_{i,b_i}) \end{cases} \tag{5}$$

Since only the conditions which are satisfied by *case a* are represented, the set of conditions represented by the state with equation (5) is specialized to *case a*. In the example of Figure 4, the misclassified case is represented by a tuple of $(v_{1,2}, v_{2,1}, v_{3,2})$ and the cornerstone case is represented by a tuple of $(v_{1,1}, v_{2,1}, v_{3,1})$. Thus, the most specialized condition is represented by $\binom{1,0,1}{0,0,0}$ in the example since $v_{1,2} \neq v_{1,1}, v_{2,1} = v_{2,1}, v_{3,2} \neq v_{3,1}$.

Step 3 : Calculate the DL for *State* as the antecedent of the newly added node.

Step 4 : Calculate the DLs for neighboring states. Neighboring states are enumerated by generalizing and specializing one condition represented by *State* each time in turn. A neighboring state $State^{'}$ $(= \binom{\alpha_1^{'}, \alpha_2^{'}, ..., \alpha_n^{'}}{\beta_1^{'}, \beta_2^{'}, ..., \beta_n^{'}})$ is defined by generalizing or specializing one condition in *State* as follows:

Select an attribute A_i which satisfies $(v_{i,a_i} \neq v_{i,b_i})$ in the *difference list* of equation (2) and set the value of α_i and β_i as either

$$Generalize : (\alpha_i^{'}, \beta_i^{'}) \quad = \quad \begin{cases} (0,1)\ ((\alpha_i^S, \beta_i^S) = (1,0)) \\ (0,0)\ ((\alpha_i^S, \beta_i^S) = (0,1)) \end{cases} \tag{6}$$

or

$$Specialize : (\alpha_i^{'}, \beta_i^{'}) \quad = \quad \begin{cases} (0,1)\ ((\alpha_i^S, \beta_i^S) = (0,0)) \\ (1,0)\ ((\alpha_i^S, \beta_i^S) = (0,1)) \end{cases} \tag{7}$$

For the remaining attributes which are not set by the equation (6) or (7), set (i.e., copy) the value of α_i and β_i as

$$(\alpha_i^{'}, \beta_i^{'}) \quad = \quad (\alpha_i^S, \beta_i^S) \tag{8}$$

Each of the conditions which can be deduced from (6) or (7) and (8) is set as $State^{'}$. A state represented by $\binom{0,0,...,0}{0,0,...,0}$ means that no condition is used for the antecedent for a new node. However, if no condition is specified, it is impossible to discriminate between the misclassified case and the cornerstone case and it is meaningless to add a node with such a rule. Thus, to exclude such a state in search, the DL with such an antecedent is set to ∞. The representation of If-Then rule is restricted to such that the conjunction of both positive and negative conditions for the same attribute is not allowed in our framework. Thus, a state in which $(\alpha_i^{'}, \beta_i^{'}) = (1,1)$ for some attribute is not searched.

Step 5 : When the DL in **Step 3** is larger than the smallest one in **Step 4**, update *State* to $State^{'}$ with the smallest DL and go back to **Step 3**. Otherwise, go to **Step 6**.

In the example in Figure 4, for the *State* represented by $\binom{1,0,1}{0,0,0}$, the DLs in bit of the neighboring states $\binom{1,0,1}{0,0,0}$, $\binom{1,0,0}{0,0,1}$, $\binom{0,0,1}{1,0,0}$ are assumed to be 120, 75 and 100, respectively. Thus, the *State* is updated to $\binom{1,0,0}{0,0,1}$. After that, since the DLs in bit for $\binom{1,0,0}{0,0,0}$, $\binom{0,0,0}{1,0,1}$, $\binom{1,0,1}{0,0,0}$ are 50, 55 and 120, the *State* is updated to $\binom{1,0,0}{0,0,0}$ again. Since the DLs for $\binom{0,0,0}{1,0,0}$, $\binom{1,0,0}{0,0,1}$ are larger than 50, the search process goes to **Step 6** with the *State* being updated to $\binom{1,0,0}{0,0,0}$.

Step 6 : Set the total description length of the binary tree with the current *State* as DL_s.

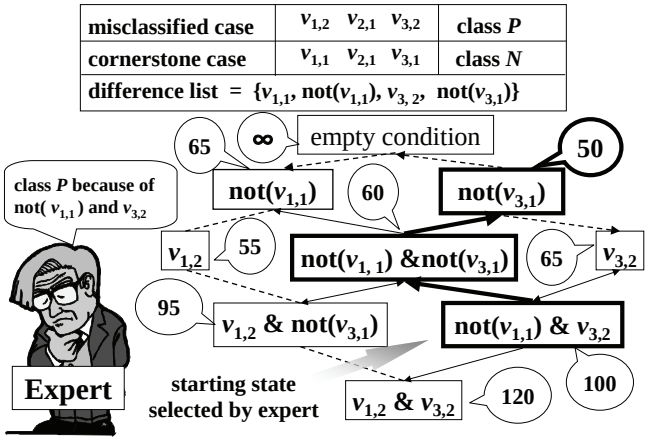

Figure 5: Search for smaller DL from both expert and data (the values of DLs are fictious)

Step 7 : Start the search for the antecedent with smaller DL from the state for the most general antecedent. The most general antecedent is the one with no condition since it can be satisfied by any case. Thus, represent *State* as $(\alpha_i^S, \beta_i^S) = (0, 0)$ for all i so that no condition is used for the antecedent. In the example in Figure 4, *State* is represented by $\binom{0,0,0}{0,0,0}$. However, if no condition is specified, it is impossible to discriminate the misclassified case and the cornerstone case and it is meaningless to add a node with such a rule. Thus, to exclude the most general state from the search space, define the DL as ∞. Repeat **Step 3** to **Step 5** and set the total description length of the binary tree with the current condition *State* as DL_g.

Step 8 : Set the DL for the newly added node to the smaller of DL_s and DL_g with the corresponding condition. If the DL is smaller than that of the original binary tree, accept the conditions which is represented with *State* as the antecedent of the newly added node. Otherwise, do not add a new node.

In the example, DL_s with the condition $\{v_{1,2}\}$ searched from $\{v_{1,2}\&v_{3,2}\}$ is 50 and DL_g with $\{not(v_{3,1})\}$ searched from the empty condition is 53. Thus, the condition $\binom{1,0,0}{0,0,0}$ corresponding to $\{v_{1,2}\}$ is selected in **Step 8**. If the DL for the original binary tree is larger than 50 bit, a new node with the If-Then rule {If $A_1 = v_{1,2}$ Then class P} and a cornerstone case $\{v_{1,2}, v_{2,1}, v_{3,2}\}$ is inserted into the binary tree.

The search process is terminated when the DL is captured in a local minimum. This method enables to construct an RDR knowledge base by using data alone[4] without the help of human experts. Admittedly exhaustive search will enable the construction of more accurate knowledge bases at the cost of much more computation. However, greedy search is employed to narrow the search space and to open the possibility for human experts to intervene with their insights.

4.1.2 Knowledge Acquisition from Both Data and Human Experts

Since the antecedent is sought by greedy search, from which state the search starts can affect the searched antecedent. When a human expert is available as the knowledge source, it is natural to exploit the expert's knowledge to search for a better antecedent. In our approach the set of conditions which are selected by the expert is used as the starting state in the search for an antecedent with smaller DL in Subsection 4.1.1, as illustrated in Figure 5. To utilize the set of conditions selected by an expert in search, **Step 2** is modified as follows and **Step 7** is omitted by simply setting DL_g to ∞.

Step 2 : Define the *State* as a set of conditions selected by an expert as follows:

$$State \equiv \binom{\alpha_1^S, \alpha_2^S, ..., \alpha_n^S}{\beta_1^S, \beta_2^S, ..., \beta_n^S}$$

[4] Implicit assumption is that the data are labeled.

$$(\alpha_i^S, \beta_i^S) = \begin{cases} (0,0)\ (v_{i,a_i} = v_{i,b_i}, \text{or, } v_{i,a_i} \neq v_{i,b_i} \text{and the expert selected neither } v_{i,a_i} \text{nor } not(v_{i,b_i})) \\ (1,0)\ (v_{i,a_i} \neq v_{i,b_i} \text{and the expert selected } v_{i,a_i}) \\ (0,1)\ (v_{i,a_i} \neq v_{i,b_i} \text{and the expert selected } not(v_{i,b_i})) \end{cases} \quad (9)$$

This enables to initiate the search from the condition selected by the expert. As for the example in Figure 5, since the expert selected $\{v_{1,2}, not(v_{3,1})\}$ from the difference list $\{v_{1,2}, not(v_{1,1}), v_{3,2}, not(v_{3,1})\}$, the condition *State* is set to $\binom{1,0,0}{0,0,1}$. The merit of using both data and human experts is that it can lead to finding a better condition from the viewpoint of MDLP, compared with the condition selected by the expert.

4.2 Adaptation for the change in Class Distribution

When the class distribution on a problem domain changes over time, part of knowledge acquired previously may become worthless and old useless knowledge accumulated in the KBS might hinder efficient acquisition of new knowledge. For instance, the difference list may have no condition even if a case is misclassified by the RDR due to the change in class distribution. The standard RDR method can't add a new node for that situation. Even if the difference list is not empty, it does not make sense to add a node when no condition in the list is judged as important by the expert. The change in class distribution can be regarded as a form of concept drift since the relationship between attribute-value pairs and class label changes over time. A simple way to cope with this issue is to discard the constructed knowledge base completely when a change in class distribution is detected and to re-construct a new knowledge base under the new environment. However, when the change is slow, it is difficult to detect. Furthermore, since some previously acquired knowledge might still be valid for the new environment, it would be desirable to reuse such knowledge as much as possible.

It may be reasonable to treat the LSN for the misclassified case as useless and to delete the node. However, if the misclassification is brought about due to some noise, the knowledge stored at that node can still be valid and deletion of that node might incur inconsistency of the knowledge base. Since the policy of the RDR method is to acquire a new piece of exceptional knowledge based on the misclassification of a case by the current knowledge base, a node should be deleted from the knowledge base with caution: otherwise, exceptional knowledge cannot be acquired if misclassification of a case always triggers the deletion of the LSN for the case.

Our criterion to cope with the change in class distribution is that a new node should not be added even if a case is misclassified by RDR as long as adding the node to patch the misclassification does not decrease the DL. Furthermore, the LSN for the misclassified case should even be deleted if doing so decreases the DL for the tree. Note that because the DL monotonically increases in proportion to the number of cases, comparing the DL for knowledge bases with different number of cases makes no sense. To enable the comparison of DL with different number of cases, the DL for the knowledge base is normalized as DL/DL', which is called NDL hereafter. Here, DL' denotes the description length for encoding the true class label of the whole cases stored in the binary tree *without* using the tree information, i.e. using only the root node information. Since the number of cases in a KB does not reflect the class distribution, DL' is used for normalization to reflect the class distribution of the cases.

4.2.1 Deletion of a Node from a Binary Tree

The proposed algorithm for node deletion is shown in Figure 6. Suppose that a case is misclassified by a binary tree of RDR and that the node ζ is the LSN for the case. First, the node ζ is tentatively deleted and the remaining binary tree is reorganized using the procedure *delete_node*, which is explained below. Among the cases which are contained in the deleted node, the cases with the same class label as that assigned for the deleted node ζ are also deleted. Other cases with different class labels in node ζ, which is represented by $cases_{diff}$ in Figure 6, are classified again by the reorganized tree and redistributed to their new LSNs. If the NDL for the knowledge base after deletion is smaller than that for the one before deletion, the node ζ is actually deleted. Otherwise, recover the original knowledge base by retracting the deletion process. Thus, our node deletion method reorganizes a binary tree of RDR by searching for the smaller (normalized) DL within the framework of MDLP as in Figure 4.

It is necessary to preserve the consistency of knowledge (If-Then rules) stored in the binary tree of RDR before and after deleting a node from the tree. The procedure *delete_node* in Figure 7 preserves

Algorithm $deleted\_RDR(RDR, node)$
  RDR : RDR KBS
  $node$: a node in RDR KBS
$w_d := 0.3$;
$NDL_{org} := ndl(RDR, w_d)$;
$cases := cases(node)$;
$classlabel := classlabel(\text{node})$;
$cases_{diff} := cases\_diff\_label(cases, classlabel)$;
$RDR_{del} := RDR$;
$delete\_node(RDR_{del}, node)$;
$distribute(RDR_{del}, cases_{diff})$;
$NDL_{del} := ndl(RDR_{del}, w_d)$;
if $NDL_{org} > NDL_{del}$
  then return RDR_{del};
  else return RDR;

$ndl(RDR, w)$: NDL of RDR (use weight w in equation (1))
$cases(node)$: cases stored at $node$
$classlabel(node)$: class label of the If-Then rule stored at $node$
$cases\_diff\_label(cases, label)$: cases with class label different from $label$
$distribute(RDR, cases)$: classify $cases$ by RDR and distribute them to their LSNs

Figure 6: Node deletion algorithm

the consistency of RDR KBS by adding the condition of deleted node to the nodes in the reorganized tree. Adding the condition achieves that the LSNs of the preserved cases are the same before and after node deletion and thus the same conclusions are drawn for the cases. Node deletion from a binary tree of RDR is illustrated in Figure 8. Suppose a case is misclassified by the binary tree in Figure 8 and its LSN is node No.2. First, node No.2 is deleted from the tree. Next, node No.4, which is the child node of node No.2 (below YES branch), is connected to node No.1 with YES branch. At the same time, the condition "cond.2", which is the antecedent of node No.2, is added to node No.4 and No.7 ("+cond.2" in Figure 8 denotes the addition of the condition "cond.2" to the original condition, *i.e.*, taking the conjuction of the condition "cond.2" with the original condition, in node No.4 and No.7). Finally, the subtree below node No.3 is connected to node No.7 with NO branch. It is easy to confirm that no inconsistency arises for the remaining cornerstone cases in the reorganized tree. Before the reorganization of the tree, inference for the cases stored at node No.3, 5 and 6 goes through a subtree whose root node is node No.3, since the condition "cond.2" of node No.2 is not satisfied. After the reorganization, inference for the same cases goes thought the subtree since the new conditions of node No.4 and 7 are not satisfied by adding the condition "cond.2" to node No.4 and 7. Thus, the same conclusion is drawn for these cases.

4.3 Pruning in RDR

To construct a knowledge base system inductively by utilizing accumulated data, it is important to construct a model (knowledge base) which avoids overfitting to training data so that it sustains high prediction accuracy for future data. C4.5, a machine learning method for constructing a model in batch, utilizes pruning to generalize the classifier which is constructed for the training data. Following this approach, pruning is also incorporated into the proposed RDR method for the incremental construction of a KBS. Pruning is expected to be effective even for the static environment in which class distribution does not change, since it plays the role of avoiding overfitting to the incoming data (*i.e.*, training data) as in C4.5.

The proposed algorithm for node deletion is shown in Figure 9. The major difference from node deletion in Subsection 4.2 is that the cases stored at the pruned node are not removed from the knowledge base but just redistributed in the reorganized tree. Thus, what is removed from the KBS is only the piece of knowledge which is represented by the If-Then rule at the pruned node, not the stored cases

procedure *delete_node*(*RDR*,*node*)

if *has_child*(*node*, YES)
then
  parent := *parent*(*node*);
  if *edgelabel*(*parent*, *node*) = YES
  then
    connect *child*(*node*, YES) to *parent* with YES label;
    node′ := *child*(*parent*, YES);
  else
    connect *child*(*node*, NO) to *parent* with NO label;
    node′ := *child*(*parent*, NO);
  repeat while (*has_node*(*node*′, NO))
    add_cond(*node*,*node*′);
    node′ := *child*(*node*′, NO);
  add_cond(*node*,*node*′);
  if *has_child*(*node*, NO)
  then connect *child*(*node*, NO) to *node*′ with NO label;
else if *has_childe*(*node*, NO)
  then
    parent := *parent*(*node*);
    if *edgelabel*(*parent*, *node*) = YES
    then connect *child*(*node*,NO) to *parent* with YES label;
    else *child*(*node*,NO) to *parent* with NO label;
return *RDR*;

has_child(*node*, *label*): *node* has a child node which is connected with *label*
edgelabel($node_A$, $node_B$): label of edge from $node_A$ to $node_B$
add_cond(*node*,*node*′): add the conditions of If-Then rule at *node* to those at *node*′

Figure 7: Deletion of a node from a binary tree of RDR

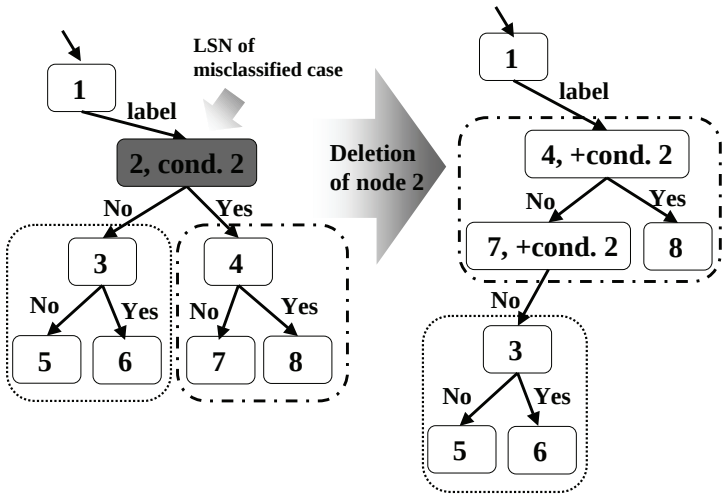

Figure 8: An example of node deletion

```
Algorithm pruned_RDR(RDR, case)
  w_p := 0.7;
  nodes := pr_nodes(RDR,case);
  NDL := ndl(RDR,w_p);
  repeat while nodes ≠ ∅
    NDL_best := ∞;
    for each node' ∈ nodes
      cases := cases(node');
      RDR' := delete_node(RDR,node');
      distribute(RDR',cases);
      NDL' := ndl(RDR',w_p);
      if NDL_best > NDL'
        then
          NDL_best := NDL'
          RDR_best := RDR'
          node_best := node'
    if NDL_best ≤ NDL
      then
        NDL := NDL_best;
        RDR := RDR_best;
        nodes := nodes - node_best;
      else return RDR;
  return RDR;
pr_nodes(RDR,case): nodes to test pruning when case is classified by RDR
```

Figure 9: Pruning algorithm

themselves. When a case is inferred (classified) by a binary tree in RDR, pruning is invoked either when a new node is added, or when its LSN is deleted, or when its LSN is the root node. Thus, pruning is never invoked whenever the RDR system correctly classifies a case except at the root node, since the knowledge stored in the RDR system is consistent with the case. When the root node is the LSN, the conclusion is returned by default knowledge which is not supported by any cornerstone case. Thus, pruning is invoked when the root node is the LSN.

Admittedly treating all nodes except the root node as the candidates for pruning is likely to construct a knowledge base with higher prediction accuracy. However, this approach does not scale up for large trees in terms of time complexity. Thus, only several nodes are selected as the candidates for pruning by *pr_nodes* in Figure 9. Currently a node is selected as a candidate for pruning when it lies on the inference path for the current case and it does not have a child which is connected with YES label. Nodes on the inference path for the case affects the classification of the case by RDR and thus is considered for pruning. In addition, when a binary tree of RDR is reorganized by removing a node with the procedure *delete_node*, the reorganized tree tends to have many nodes along NO branch if the node has a child node along YES branch, as shown in Figure 8. Thus, the nodes which have a child along YES branch are excluded from pruning to avoid an unbalanced tree.

Note that w_p, which is different from w in Section 3, is used for pruning. Currently w_p is set to 0.7 (w is set to 0.3) from our preliminary experiments. Larger weight puts the bias on emphasizing the DL for a binary tree and thus facilitates pruning. With small weight (*e.g.*, $w_p = 0.3$) pruning was not often invoked, which hindered the construction of KB with high prediction accuracy in our preliminary experiments. However, turning off pruning and utilizing larger weight (*i.e.*, $w > 0.3$) for knowledge acquisition or node deletion does not necessarily lead to the construction of a knowledge base with high prediction accuracy.

Table 2: Summery of datasets

Dataset Name	#of Case	#of Class	#of Attribute	Dataset Name	#of Case	#of Class	#of Attribute
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				

*Nominal attribute, **Numerical attribute, ***This database has two kinds of attributes: nominal attribute / numerical attribute

5 Evaluations and Discussions

The proposed adaptive RDR method is evaluated using 15 datasets from University of California Irvine Data Repository [1] (see Table 2). Two types of experiments were conducted. The influence of the change in class distribution was examined in the first experiment. The influence of the change in knowledge source (i.e., from an expert to data and vice versa) during incremental knowledge acquisition was examined in the second experiment. Synthesized cases were created and used in the experiments and the results were evaluated with respect to the prediction accuracy and the NDL for the constructed RDR KBS.

[**Generation of Data with different Class Distribution**] Let a set of cases (data) according to a certain class distribution be X_{org}. We synthesized a set of cases X_{chg} with different class distribution from the original cases X_{org} for each dataset as follows. First, all the cases in X_{org} are sorted in lexical order for class label. By preserving this order, the cases with the same class label are then sorted with respect to values in lexical order for nominal attributes and in ascending order for numerical attributes. Finally, to synthesize X_{chg} in which about x% of cases have different class labels from those in X_{org}, the class labels of (#of cases $\div$ #of class labels $\div$ (100 $\div$ x)) cases in X_{chg} are changed by shifting them to the neighboring ones in lexical order. The above process is illustrated in Figure 10.

[**Training Data and Test Data**] Each set of cases was divided into the 75% training data (e.g., X_{org}^{train}, X_{chg}^{train}) and the 25% test data (e.g., X_{org}^{test}, X_{chg}^{test}). A predefined number of cases were sampled randomly with replacement from the training data and a knowledge base in RDR was constructed incrementally. When the total number of sampled cases reached the specified number, the training data was switched to the next one to simulate the change in class distribution.

[**Simulated Expert**] Simulated Expert [5] (SE) is usually used instead of a human expert for the reproduction of experiments and consistent performance estimation in the RDR research community. Therefore, this paper follows this tradition. A SE consists of a set of If-Then rules, which are derived by running standard C4.5 [14] for the training data and then running C4.5rules. Note that when X_{org}^{train} (X_{chg}^{train}) is the population, we created the SE for the population using X_{org} (X_{chg}) as the training data. This means that the SE is a really good expert, and the SE can immediately change his/her internal model or expertise for the domain according to the changes of class distribution. Said differently, the label of incoming data is always assumed to be correct, reflecting the environment.

For the If-Then rule of the SE which correctly predicts the class label of a misclassified case, a set of conditions selected from the difference list by the SE is defined as the intersection between the difference list and the antecedent of the If-Then rule. For a numeric attribute, if there is a condition in the difference list that satisfies the inequality condition of the If-Then rule by the SE, this inequality is interpreted as a condition in the intersection. If there is no stored case in the

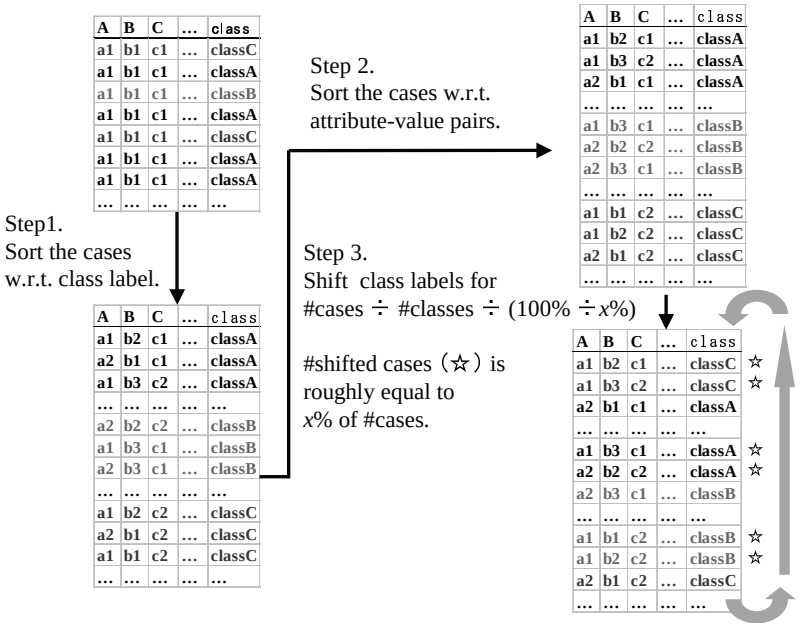

Figure 10: Synthesized cases with different class distribution

binary tree with the attribute-value for the inequality condition in the intersection, the attribute-value in the stored cases which is the nearest to the original value is treated as the cut-point in inequality to start the search. Note that no negative expression (not) is used in the If-Then rule set induced by C4.5. In order to enable the selection of negative conditions from the difference list, we binarized attribute-values and force the rule set to have negative ones.

[**Accuracy of Knowledge Base**] The error rate of misclassified case in the test data was examined using the knowledge base at pre-specified time points (at the pre-specified number of sampled case) for each datasets. We used X_{org}^{test} (X_{chg}^{test}) as the test data when the population is X_{org}^{train} (X_{chg}^{train}), respectively.

In the following experiments, when only data is used as the knowledge source and knowledge acquisition is conducted as described in Subsection 4.1.1, it is expressed as "Data". On the other hand, when both data and SE are used as the knowledge source as described in Subsection 4.1.2, it is expressed as "SE&Data". Node deletion in Subsection 4.2 and pruning in Subsection 4.3 were used by default. When node deletion is not used, it is expressed as "(w/o ND)". Likewise, when pruning is not used, it is expressed as "(w/o PR)".

5.1 Experiment on the change in Class Distribution

On each dataset the class distribution of the problem domain was changed abruptly twice during the simulation for the RDR system to acquire knowledge from the data. A set of cases X_{chg} with about 10% different class distribution from the original data X_{org} were generated. To simulate the change in class distribution the environment (population) from which the cases were sampled was change from X_{org} to X_{chg} to X_{org}. The total number of cases sampled from each population was set to three times large as that of the training data. Since a different ordering of sampled cases results in a different knowledge base of RDR [19], we repeated the simulation 10 times for each dataset by changing the parameter of random sampling from the population at each simulation. Finally, the error rate was calculated as the average of 10 simulations for each dataset. Pruning in Subsections 4.3 was used throughout the following experiments.

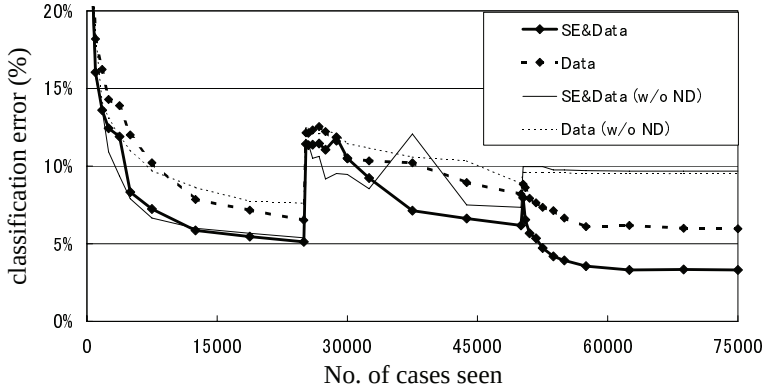

Figure 11: One of 10 simulations for "PenDigits" (error rate)

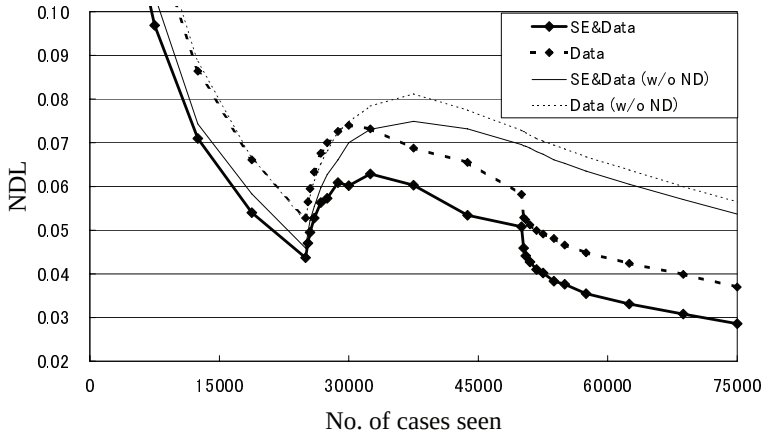

Figure 12: One of 10 simulation for "PenDigits" (NDL)

Figures 11 and 12 show the change of error rate and the NDL in one of 10 simulations for the dataset "PenDigits". For each simulation the RDR system received 75000 cases. In these figures the lines for "Data" correspond to the knowledge acquisition method in Subsection 4.1.1 and the ones for "SE&Data" to the method in Subsection 4.1.2. Two lines with marks corresponds to the methods with node deletion in Subsections 4.2.1. The other two lines with no marks (w/o ND) correspond to the methods without node deletion. By comparing two lines for "SE&Data" and "Data" in Figure 11, the error rate for the former is smaller than that for the latter up to the 25000th sampled case. Likewise, the error rate for "SE&Data (w/o ND)" is smaller than that for "Data (w/o ND)" up to the 25000th sampled case. The result shows that incorporating the knowledge of expert is useful, since an expert can select a set of conditions from which the antecedent with more smaller DL can be searched compared with the situation in which only data is used. The class distribution was changed abruptly at the 25001st case and 50001st case in the simulation. Such changes are reflected as the sharp increase of the error rate and the NDL in Figures 11 and 12. The figures also show that deleting nodes which contain inconsistent knowledge from the knowledge base contributes to reducing the error rate. Figure 12 shows that the NDL decreases monotonically by knowledge acquisition while the class distribution does not change and that it increases when the class distribution changes. The NDL of "SE&Data" ("Data") with node deletion is smaller than that of "SE&Data (w/o ND)" ("Data (w/o ND)"). Although the difference of the NDL with and without node deletion is not large before the change in class distribution, the difference increases after the change as more nodes are deleted. This also suggests that our deletion mechanism works well to keep the size of knowledge base concise. Note that the method with the lowest NDL at the end of simulation, namely "SE&Data", has also the lowest error rate, which confirms the validity of the MDLP in RDR.

Table 3 summarizes the results of experiment with both node deletion and pruning for 15 datasets at the end of each simulation when the RDR system have received all the cases. The columns for "size" show the number of nodes of the binary tree of RDR and that of the decision tree by C4.5. The ones for "%

Table 3: Summary of results for the change in class distribution

Dataset	SE&Data				Data				C4.5 (whole)	
	size	% of cases	RDR	C4.5 (retained)	size	% of cases	RDR	C4.5 (retained)	size	C4.5 (whole)
Car	77.8	95.3	*4.2**	6.3	77.9	95.2	*4.3**	6.4	401.5	6.8
Nursery	224.3	94.1	3.0*	2.7	240.9	93.9	2.9*	2.7	1285.2	3.2
Mushrooms	21.2	93.0	3.7*	2.3	22.2	93.9	5.0	2.4	929.2	4.5
Krvkp	31.5	93.5	*2.3**	3.1	39.3	92.7	*2.0**	2.2	199.8	3.9
VotingRecord	14.7	95.6	6.3	3.7	14.3	95.3	6.1	3.5	56.6	3.2
BreastCancer	21.2	95.1	*4.6**	5.3	24.3	93.5	4.9*	4.7	142.0	5.6
Splice	80.0	90.4	7.9*	6.2	50.0	74.9	*8.8**	10.4	1396.2	9.1
Image	75.8	93.9	*2.6**	3.3	101.1	94.7	5.4*	3.8	207.8	8.0
PageBlocks	69.2	93.0	*4.4**	5.1	91.8	90.9	*4.9**	5.2	325.4	5.8
PenDigits	381.9	93.9	*3.5**	3.8	529.0	94.2	5.0*	4.5	985.0	8.4
Yeast	230	56.6	42.9	40.0	297.6	62.2	41.5	37.3	969.2	24.2
PimaIndians	29.1	62.4	26.0	23.0	24.0	41.7	33.3	24.5	269.0	15.2
GermanCredit	60.9	72.2	22.5	19.4	71.1	73.5	20.6	17.1	513.0	14.4
Cmc	116.6	44.6	47.9	43.5	126.6	46.3	47.4	42.5	1299.0	29.1
AnnThyroid	30.6	95.9	*1.2**	2.2	37.9	95.2	*1.8**	2.7	235.6	5.0

size: the number of nodes of the binary tree of RDR and that of the decision tree by C4.5
% of cases: the ratio (%) of cases kept in the knowledge base w.r.t. the whole sampled cases
RDR, C4.5(retained), C4.5(whole): error rate (%)
*: the error rate of RDR is lower than that of C4.5 (whole)
Italic value: the error rate of RDR is lower than that of C4.5 (retained)

of cases" show the ratio of cases which were kept inside the knowledge base to the whole sampled cases. The ones for "RDR" represent the error rate of the knowledge base for the test data. The decision trees were constructed by C4.5 using the cases held in the knowledge bases. The columns for "C4.5 (retained)" show the error rate of the decision tree. In addition, the column for "C4.5 (whole)" shows the error rate of the decision trees using all sampled cases.

From Table 3 it can be said that node deletion is effective to adapt for the change in class distribution since the column for "RDR" shows lower error rate than that for "C4.5(whole)" in 10 datasets out of 15 for "SE&Data" and in 9 datasets for "Data" (with * in Table 3). This probably is the result that the worthless knowledge can be properly deleted from the knowledge base by deleting the nodes which contain such knowledge. However, with node deletion the number of cases used to construct a knowledge base becomes different from that for "C4.5 (whole)". Comparison of the column for "RDR" and that for "C4.5 (retained)" shows the difference in error rate for the same number of cases. Although C4.5 (retained) showed lower error rate in many datasets, C4.5 constructs a classifier in batch, contrary to the incremental approach in RDR. If C4.5 were modified to construct a decision tree incrementally, the error rate of the constructed decision tree would increase. The result for the dataset "Mushrooms" suggests that knowledge from the SE is effective since the RDR for "SE&data" shows the lower error rate than the other method, namely, the RDR for "Data". Other data sets where such a tendency is observed are "Nursery", "Krvkp" and "Image". Moreover, for the dataset "AnnThyroid", more accurate knowledge bases were constructed for "RDR" compared with "C4.5 (retained)" both for "SE&Data" and "Data". Other datasets with italic value for "RDR" in Table 3 also indicates the same result.

Unfortunately, compared with "C4.5 (whole)", the error rate for "RDR" was high for 4 datasets ("Yeast", "PimaIndians", "GermanCredit", "Cmc") for which the number of cases is relatively small. Since the construction of a KBS is based on the MDLP, if only small amount of cases are available, it is difficult to construct a KB with high prediction accuracy. Our current conjecture is that the deletion

Table 4: Significance test of parired t test (p-value of one-side test)

Dataset	SE&Data vs. Data effect of expert		RDR vs. C4.5(whole) as classifier		C4.5(retained) vs. C4.5(whole) effect of node deletion	
	RDR	C4.5(retained)	SE&Data	Data	SE&Data	Data
Car	0.4509	0.2475	0.0000^{+}	0.0001^{+}	0.0045^{+}	0.0488^{+}
Nursery	0.4122	0.4125	0.3567	0.2470	0.0227^{+}	0.0451^{+}
Mushrooms	0.0272^{+}	0.4660	0.1032	0.2411	0.0003^{+}	0.0001^{+}
Krvkp	0.3121	0.0514	0.0070^{+}	0.0018^{+}	0.0073^{+}	0.0033^{+}
VotingRecord	0.4393	0.3677	0.0045^{-}	0.0317^{-}	0.1904	0.2602
BreastCancer	0.2955	0.0375^{-}	0.0338^{+}	0.0929	0.2063	0.0081^{+}
Splice	0.2998	0.0762	0.0629	0.3870	0.0000^{+}	0.3193
Image	0.0026^{+}	0.1228	0.0000^{+}	0.0011^{+}	0.0000^{+}	0.0000^{+}
PageBlocks	0.0435^{+}	0.3104	0.0000^{+}	0.0080^{+}	0.0002^{+}	0.0077^{+}
PenDigits	0.0002^{+}	0.0416^{+}	0.0000^{+}	0.0000^{+}	0.0000^{+}	0.0000^{+}
Yeast	0.2799	0.1264	0.0000^{-}	0.0000^{-}	0.0000^{-}	0.0000^{-}
PimaIndians	0.0289^{+}	0.2636	0.0001^{-}	0.0000^{-}	0.0003^{-}	0.0003^{-}
GermanCredit	0.2581	0.1738	0.0014^{-}	0.0019^{-}	0.0107^{-}	0.0169^{-}
Cmc	0.4382	0.3963	0.0000^{-}	0.0000^{-}	0.0005^{-}	0.0000^{-}
AnnThyroid	0.1205	0.1155	0.0000^{+}	0.0000^{+}	0.0000^{+}	0.0000^{+}

C4.5(retained): C4.5 for the retained cases in RDR
C4.5(whole): C4.5 for the whole sampled cases
$^{+}$: the error rate of A in "A vs. B" is lower than that of B with 95% confidence level, e.g., the first column shows that the error rate of RDR from SE&Data is lower than that of RDR from Data for "Mushrooms" with 95% confidence level
$^{-}$: the error rate of B in "A vs. B" is lower than that of A with 95% confidence level, e.g., the second column shows that the error rate of C4.5 for the retained cases in the RDR from Data is lower than that of C4.5 for the retained cases in the RDR from SE&Data for "BreastCancer" with 95% confidence level

algorithm tends to delete too large fraction of cases, especially when the size of the original datasets is relatively small. For instance, only 44.6% of the original cases were held in the knowledge base for "Cmc" after deletion. In addition, C4.5 constructs a classifier in batch, contrary to the incremental construction of RDR KB. Thus, the result indicates that the proposed method can construct a reasonablly good KB with slightly larger error rate.

Table 4 summarizes the comparison of error rates by using paired t-test. P values of one-side test are shown in Table 4 and the ones with less than or equal to 0.05 are marked with either $^{+}$ or $^{-}$ to indicate the significance with 95% level. A value with $^{+}$ indicates that the error rate of A in "A vs. B" is statistically lower than that of B with 95% confidence level. For instance, the first column shows that the error rate of RDR from SE&Data is lower than that of RDR from Data for "Mushrooms" with 95% confidence level. Likewise, a value with $^{-}$ indicates that the error rate of B in "A vs. B" is lower than that of A with 95% confidence level. From the first column in Table 4, it can be concluded that utilizing the expertise of an expert for constructing a RDR KBS when class distribution changes is effective and does not lower the performance of the constructed KBS. Excluding the above 4 datasets for which RDR did not work well, the third and forth column indicate that the error rate of RDR is mostly lower than that of C4.5. Thus, the proposed method is effective for constructing a classifier with high prediction accuracy when class distribution changes. Finally, the columns for the effect of node deletion indicate that the node deletion algorithm works well to filter out the obsolete cases by deleting the nodes which hold these cases.

5.2 Experiment on the change in the Knowledge Source

The second experiment was conducted to investigate the effect of changing the knowledge source during the consecutive course of knowledge acquisition. Suppose a human expert is available only for a certain duration to construct a knowledge base and the knowledge source can be switched to data when the expert is not available. If the constructed KBS has the equivalent capability with the one for which the

Table 5: Results when the knowledge source is changed

Dataset	SE&Data		SE→Data		Data		C4.5	
	RDR	size	RDR	size	RDR	size	C4.5	size
Car	17.2	10.4	16.6	10.3	15.7	11.3	17.0	64.8
Nursery	10.2	22.5	9.6	25.1	10.9	24.1	6.6	211.2
Mushrooms	0.1	7.1	0.1^{-**}	7.1	0.0	7.7	0.1	31.3
Krvkp	5.1^{+*}	7.1	6.2	7.2	10.6	6.2	2.8	36.4
VotingRecord	6.3	2.5	6.3^{+**}	2.5	7.4	2.9	5.6	6.0
BreastCancer	8.2	3.4	7.4	3.6	7.8	3.7	8.2	21.0
Splice	11.5	6.9	10.6	7.1	10.0	7.6	11.2	193.8
Image	20.3	14.7	20.6	14.5	18.6	16.6	7.1	43.2
PageBlocks	8.0	7.7	8.5	6.9	9.1	6.9	4.7	37.0
PenDigits	14.4	70.5	14.8	70.1	15.0	72.9	7.1	181.6
Yeast	61.5	6.8	65.0	7.0	62.9	7.7	49.6	103.8
Pima	30.8	1.7	32.9	1.5	32.9	1.5	27.7	24.6
GermanCredit	28.1	2.4	28.1	2.4	28.1	2.4	27.8	53.7
Cmc	58.6^{-*}	1.9	53.9	2.7	52.9	3.0	49.4	115
AnnThyroid	1.3^{+*}	6.6	2.0	6.8	2.5	6.8	0.8	15.2

size: the number of nodes of the binary tree of RDR and that of the decision tree by C4.5
RDR, C4.5: error rate (%)
$^{+*}$ ($^{-*}$): the error rate of SE&Data is lower (higher) than that of SE→Data with 95% confidence level
$^{+**}$ ($^{-**}$): the error rate of SE→Data is lower (higher) than that of Data with 95% confidence level

expert is available all the time, it will contribute to reducing the cost of personnel expenses.

Three methods "SE&Data", "SE→Data" and "Data" were compared. The method "SE&Data" represents that both the SE and data were used as the knowledge source. On the other hand, the method "Data" represents that only data was used as the knowledge source. The method "SE→Data" represents that both the SE and data were used for the initial phase and then the knowledge source was switched to data thereafter. In this experiment the knowledge source was switched when the number of sampled cases reached one third of the total sample cases. Inductive learning method would be eventually result in a correct KBS when there is a sufficiently large number of cases available. The knowledge of an expert is helpful when there is not much data accumulated. Thus, the total number of sampled cases was set to 25% of original cases for each dataset in the experiment. Results are summarized in Table 5.

Our conjecture is that "SE→Data" is equivalent to "SE&Data" and is superior to "Data" so that an expert is not necessarily required to be available all the time. With paired t-test (one-side test) with 95% confidence level, the error rate of "SE→Data" is equivalent to that of "SE&Data" for 12 datasets, inferior to for 2 datasets (with $^{+*}$ in Table 5) and superior to for 1 dataset (with $^{-*}$). On the other hand, the error rate of "SE→Data" is equivalent to that of "Data" for 13 datasets, inferior to for 1 datasets (with $^{-**}$) and superior to for 1 dataset (with $^{+**}$). Thus, there was no distinct difference in the prediction accuracy between three methods and the results did not support our hypothesis for the situation in which only the knowledge source changes.

5.3 Experiment on the change in both Class Distribution and Knowledge Source

The effect of the change in knowledge source might manifest when it occurs in conjunction with the change in class distribution. Thus, another experiment was conducted by mixing these changes using the dataset "Nursery" as X_{org}. We chose this dataset since it contains many cases and the prediction accuracy of the SE for the dataset is sufficiently high. A set of cases X_{chg} were generated so that the

Table 6: Methods for the change in the knowledge source

method	knowledge source for sampled cases				
	1∼1000	1001∼2500	2501∼3000	3001∼4500	4501∼9000
SE→Data	SE&Data (w/o ND,PR)	Data	Data	Data	Data
Data	Data	Data	Data	Data	Data
SE/Data	SE&Data (w/o ND,PR)	Data (w/o ND,PR)	Data	SE&Data (w/o ND)	Data
SE'	SE&Data (w/o ND,PR)	SE&Data (w/o ND,PR)	SE&Data (w/o ND,PR)	SE&Data (w/o PR)	SE&Data (w/o PR)
SE	SE&Data (w/o ND,PR)	SE&Data (w/o ND,PR)	SE&Data (w/o ND,PR)	SE&Data (w/o ND,PR)	SE&Data (w/o ND,PR)
C4.5	C4.5	C4.5	C4.5	C4.5	C4.5

SE: Simulated Expert, ND: Node Deletion, PR: PRuning

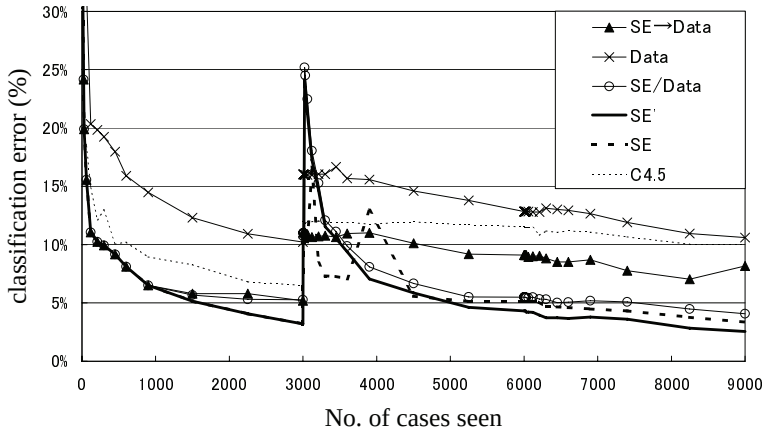

Figure 13: Result for the change in knowledge source (error rate)

class label for about 10% in X_{org} was changed in X_{chg}. First, by treating X_{org} as the original population, cases which were sampled randomly from X_{org}^{train} were used to construct a knowledge base. When the total number of sampled cases reached 3000, the population was changed to X_{chg}. After that, the system received 6000 cases sampled from X_{chg}^{train}. Thus, the change in class distribution occurred at the 3001st case and 9000 cases were used in total.

Five methods, each of which was a combination of "SE&Data" and "Data" with a predefined interval of sampled cases, were examined with respect to the prediction accuracy and the NDL for the constructed knowledge base. The characteristics of the methods are summarized in Table 6. C4.5 was also examined for comparison. For instance, the method "SE→Data" represents the situation in which both the SE and data were used until 1000th case and node deletion nor pruning was used. After 1001st case, only data was used as the knowledge source and both node deletion and pruning were used. As another example, the method "SE' " represents the situation in which both the SE and data were used as the knowledge source all the time, as in the method "SE", but node deletion was used after the 3001st case in "SE' ". Note that since C4.5 is not an incremental method, each time a new case was sampled, the already constructed decision tree was discarded and a new tree was constructed for all the sampled cases. For instance, the decision tree for the 6000th case was constructed by treating all the sampled 6000 cases as the training data.

The results are illustrated in Figures 13 and 14, which show the change of error rate with respect to the test data and the change of the NDL, respectively. By comparing the line for "SE→Data" and that for "Data" (only data is used as the knowledge source), the speed of knowledge acquisition is faster in "SE→Data". Here, we say that knowledge acquisition is faster in "SE→Data" since the error

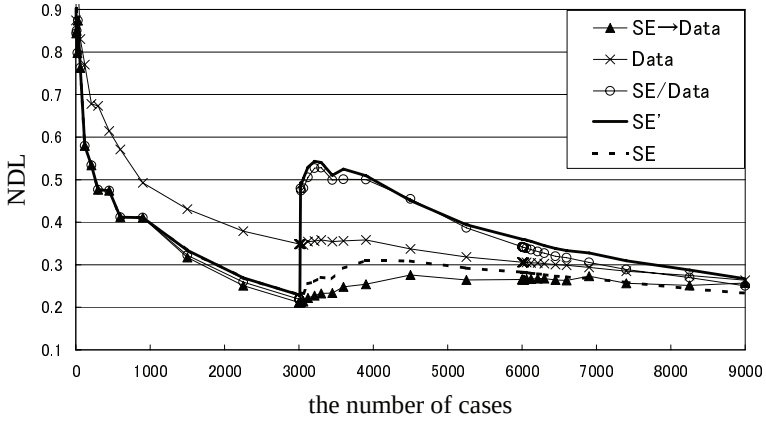

Figure 14: Result for the change in knowledge source (NDL)

rate decreases faster. Even after the change in class distribution at the 3001st case, the error rate was kept low in "SE→Data" by utilizing the expertise of the SE. The method "SE/Data"is the same with "SE→Data" except that the SE was also utilized from the 3001st to 4500th case. Compared with "SE→Data" and "Data", the method "SE/Data" showed even faster knowledge acquisition after the change in class distribution. The method "SE' ", which utilizes node deletion after the 3001st case, showed the immediate adaptation to the change and the error rate was the lowest. Note that the error rate in "SE' " is equivalent to that in "SE/Data". This indicates that even if an expert is not available all the time, it would be possible to construct an equivalent knowledge base if knowledge is acquired from the expert for an appropriate duration.

Compared with C4.5, the methods except "Data" showed lower error rate. Since learning (knowledge acquisition) is conducted incrementally and inductively from data in "Data", it seems reasonable that the error rate in "Data" is higher than that in C4.5 which works in batch. However, the difference of the error rate between these methods gets smaller after the change in class distribution. This can be attributed to the deletion of useless pieces of knowledge (nodes) by the proposed node deletion method, which enables to follow the change in class distribution adaptively.

In Figure 14 the NDL suddenly increases in "SE/Data" and "SE' " at the 3001st case when class distribution changed. However, it rapidly decreases compared with "SE→Data" and "Data". This rapid decrease of NDL is attributed to the deletion of useless pieces of knowledge by the expert. Although the error rate in "SE" is equivalent to that in "SE' " from the 6001st to 9000th case in Figure 13, the change of the NDL differs in these methods. The NDL is small in "Data" (which does not rely on the expert) and in "SE→Data" (which uses the expert only at the initial phase of knowledge acquisition). However, the error rate of these methods are larger than that of the ones in which the expert is more heavily used, namely, "SE/Data", "SE' " and "SE". In our approach DL is calculated based on only the already encountered data (cases) and the remaining not-yet encountered training data are not used even if they are used for constructing a knowledge base anyway. Thus, there is some limitation for constructing an effective knowledge base incrementally solely based on the MDLP with respect to the prediction accuracy for unseen future data.

6 Conclusion

This paper proposes an adaptive Ripple Down Rules (RDR) method based on the Minimum Description Length Principle (MDLP) aiming at effective knowledge acquisition when an environment changes. When class distribution changes, some pieces of knowledge previously acquired become worthless, and the existence of such knowledge may hinder acquisition of new knowledge. To cope with the change in class distribution, useless knowledge is properly discarded by deleting nodes which contain such knowledge from a binary tree of RDR. In addition, the method allows that a knowledge based system can be constructed adaptively by switching the knowledge source from domain experts to data and vice versa at any time depending on their availability. Being able not to rely on a human expert at all times during the construction of a KBS will contribute to reducing the cost of personnel expenses. The experiments

with synthesized data show that the proposed adaptive RDR method is effective for the change in class distribution and knowledge source and that it is worth following this path.

Our immediate future direction is as follows. First, since our method relies on MDLP, even when RDR misclassified a case, a new node is not necessarily added as long as the total description length of the RDR knowledge base with the added node does not decrease. This may derogate from the patching principle of RDR in which a patch (new node) is added whenever a case is misclassified. In addition, when the number of cases accumulated is not sufficiently large, the description lengh is not reliable to accurately estimate the performance of a knowledge base system. Even if an expert selects conditions which are effective for future unseen data, the conditions might not be utilized especially at the initial phase of knowledge base construction. Thus, it is necessary to elaborate the integration of knowledge acquisition from experts and inductive learning from data. Second, the coefficient w to calculate the total description length is set from our preliminary experiments. In terms of MDLP, this coefficient should be 1.0 if a truly appropriate encoding method is used. There are many coding methods other than our method in Section 3 and the calculation of description length differs depending on the coding. Thus, we plan to devise a more appropriate encoding method of the binary tree in RDR so that the description length can be more robust to the value of the coefficient. One possible appraoch is to reflect the probability distribution of attribute-value from data in the calculation of description length for a binary tree. Third, currently only one class label is inferred for a case by RDR. Extending our framework to incorporate Multiple Classification RDR (MCRDR) method"[11] will enable the inference of multiple classes by RDR based on MDLP. Finally, Knowledge acquisition in RDR is conducted within the framework of supervised learning and it is assumed that correct class labels are notified when the class distribution changes to trigger node addition, deletion and pruning in our approach. By calculating the DLs for each possible class label and employing the one with smallest DL is one direction to enable the reorganization of a binary tree without requiring correct class labels.

Acknowledgments

This work was partially supported by the grant-in-aid for scientific research 1) on priority area "Active Mining" (No. 13131101, No. 13131206) and 2) No. 13558034 funded by the Japanese Ministry of Education, Culture, Sport, Science and Technology.

References

[1] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/ ∼mlearn/MLRepository.html.

[2] P. Compton, G. Edwards, G. Srinivasan, et al. Ripple down rules: Turning knowledge acquisition into knowledge maintenance. *Artificial Intelligence in Medicine*, pages 47–59, 1992.

[3] P. Compton, K. Horn, J.R. Quinlan, and L. Lazarus. Maintaining an expert system. In J.R. Quinlan, editor, *Application of Expert Systems*, pages 366–385. Addison Wesley, 1989.

[4] P. Compton and R. Jansen. A philosophical basis for knowledge acquisition. *Knowledge Acquisition*, pages 241–257, 1990.

[5] P. Compton, P. Preston, and B.H. Kang. The use of simulated experts in evaluating knowledge acquisition. In *Proc. of the 9th Knowledge Acquisition for Knowledge Based Systems Workshop*, Banff, Canada, University of Calgary, 1995. SRDG Publications.

[6] B.R. Gaines. An ounce of knowledge is worth a ton of data: Quantiative studies of the trade-off between expertise and data based on statistically well-founded empirical induction. In *Proc. of the 6th International Workshop on Machine Learning*, pages 156–159, San Mateo, California, June 1989. Morgan Kaufmann.

[7] B.R. Gaines. The trade-off between knowledge and data in knowledge acquisition. In G. Piatetsky-Shapiro and W. Frawley, editors, *Knowledge Discovery in Databases*, pages 491–505, Cambridge, Mass, 1991. MIT Press.

[8] B.R. Gaines and P. Compton. Induction of ripple-down rules. In *Proc. of the 5th Australian Joint Conference on Artificial Intelligence*, pages 349–354, Singapore, 1992. World Scientific.

[9] B.R. Gaines and P. Compton. Induction of ripple-down rules applied to modeling large databases. *Journal of Intelligent Information Systems*, 5(2):211–228, 1995.

[10] D.K. Gary and J.H. Trevor. Optimal network construction by minimum description length. *Neural Computation*, pages 210–212, 1993.

[11] B.H. Kang. *Validating Knowledge Acquisition: Multiple Classification Ripple Down Rules*. PhD thesis, Dept. of Electrical Engineering, University of New South Wales, 1996.

[12] B.H. Kang and P. Compton. Knowledge acquisition in context: Multiple classification problem. In *Proc. of the Second Pacific Rim International Conference on Artificial Intelligence*, volume 2, pages 847–853, Seoul, 1992.

[13] K. Morik, S. Wrobel, J. Kietz, and W. Emde, editors. *Knowledge Acquisition and Machine Learning: Theory, Methods, and Applications*. Academic Press, London, 1993.

[14] J.R. Quinlan, editor. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[15] J.R. Quinlan and R.L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, pages 227–248, 1989.

[16] J. Rissanen. Modeling by shortest data description. *Automatica*, pages 465–471, 1978.

[17] J. Suzuki. A construction of bayesian networks from databases on an mdl principle. In *Proc. of the 9th Conference on Uncertainty in Artificial Intelligence*, The Catholic University of America, Washington, D.C., 1993. Morgan Kaufmann.

[18] T. Wada, T. Horiuchi, H. Motoda, and T. Washio. A description length-based decision criterion for default knowledge in the ripple down rules method. *Knowledge and Information Systems*, 3(2):146–167, 2001.

[19] T. Wada, H. Motoda, and T. Washio. Knowledge acquisition from both human expert and data. In *Proc. of the Fifth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 550–561, 2001.

[20] S. Wrobel, editor. *Concept Formation and Knowledge Revision*. Kluwer Academic Publishers, 1994.