A New Look at Default Knowledge in Ripple Down Rules Method

Takuya Wada, Tadashi Horiuchi, Hiroshi Motoda and Takashi Washio

Institute of Scientific and Industrial Research,
Osaka University
Mihogaoka, Ibaraki, Osaka 567-0047, JAPAN

Abstract. "Ripple Down Rules (RDR)" Method is one of the promising approaches to directly acquire and encode knowledge from human experts. It requires data to be supplied incrementally to the knowledge-base being constructed and new piece of knowledge is added as an exception to the existing knowledge. Because of this patching principle, the knowledge acquired strongly depends on what is given as the default knowledge. Further, data are often noisy and we want the RDR noise resistant. This paper reports experimental results about the effect of the selection of default knowledge and the amount of noise in data on the performance of RDR using a simulated expert. The best default knowledge is characterized as the class knowledge that maximizes the minimum description length to encode rules and misclassified cases. The effect of noise is sensitive at an earlier stage of knowledge acquisition where its performance strongly depends on the selection of default knowledge, but RDR eventually converges to its stable performance although the size of the knowledge base is strongly affected by the chosen default knowledge.

1 Introduction

Implicit assumption of the development of knowledge-based systems is that the knowledge of expertise is stable and it is worth investing much on knowledge acquisition from human experts. However, the rapid innovation of technology in recent years makes the existing knowledge out-of-date so soon and requests frequent update. In other words, we should now think that the knowledge of expertise is not static but dynamic in real world problems. In addition, the advancement of worldwide networks such as Internet has changed the computer usage practice drastically. It is now possible that many users access to a single knowledge-based system through a network [9], and further, the needs arise that multiple experts supply the new knowledge continuously through the same network.

Under these circumstances, a new methodology to construct knowledge-based systems for continuous changes is in great demand. Knowledge is value-added data and includes such things as experience, insight and skill. "(Multiple Class) Ripple Down Rules ((MC)RDR)" [1, 8][1] is one of the promising approaches to

[1] In this paper, we limit our analysis to RDR.

realize such knowledge-based systems and to directly acquire and encode knowledge from human experts. (MC)RDR is a performance system that does not require high level models of knowledge as a prerequisite to knowledge acquisition, and has been shown very effective in knowledge maintenance as well as acquisition of reflexive knowledge. Recent advancement shows that MCRDR can be applied to a configuration task [15]. Handling multiple sources by MCRDR has not yet been resolved.

RDR exclusively relies the knowledge to acquire on human experts and one of the advantages is that it does not require any data statistics as machine learning techniques do. Use of this knowledge, if available, may enhance RDR's performance. Default knowledge is one such knowledge that requires statistics to characterize itself. In this paper we try to identify what characterizes good default knowledge when the data available have noise, and show that selection of good default knowledge contributes to building a noise-resistant compact knowledge base.

2 Ripple Down Rules Revisited

Ripple Down Rules (RDR) is a knowledge acquisition technique that challenges the *KA bottleneck* problem by allowing rapid development of knowledge bases by human experts without the need of analysis or intervention of a knowledge engineer (KE). From long experience of knowledge-based systems development [1], it was made clear that human experts are not good at providing information on how they reach the conclusions, rather they can justify that their conclusions are correct[2, 4]. Its basis is the maintenance and retrieval of cases. It tries to use the historical way in which the expert provides his/her expertise to justify the system's judgements[7]. The cases and associated justifications (rules) are added incrementally when a case is misclassified in the retrieval process. This is similar to "failure-driven memory" which was introduced by Schank[16]. Experts are very good at justifying their conclusions about a case in terms of differences from other cases [10]. When a case is incorrectly retrieved by an RDR system, the knowledge acquisition (maintenance) process requires the expert to identify how the case differs from the present case. This has a similar motivation to work on personal construct psychology where identifying differences is a key strategy[6]. The notion of on going refinement by differentiating cases is a useful model of human episodic memory. 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 previously correctly classified case.

Each time a rule is added incrementally when a case is misclassified, the case that prompted the rule is stored (called the "cornerstone case"). Depending on whether the conditions in the rule that incorrectly fired are true or false, the new rule and its cornerstone case are added at the end of either the YES or NO branch of the old rule. Knowledge is never removed or changed, simply modified by the addition of exception rules. The tree structure of the knowledge and the

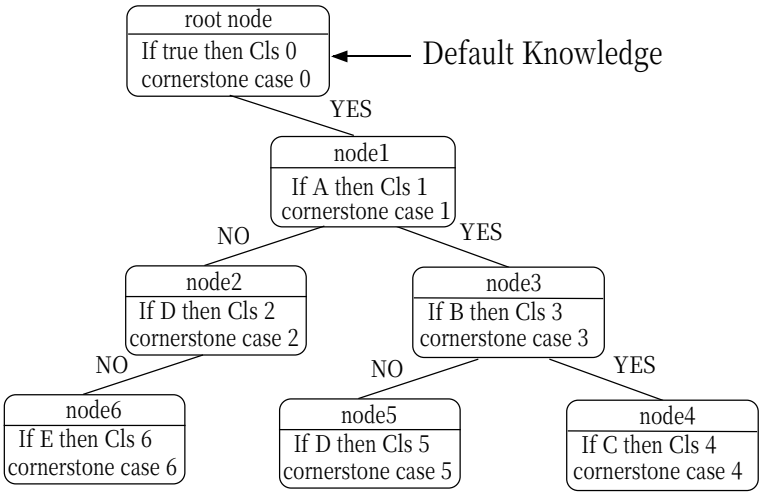

Fig. 1. Knowledge structure of Ripple Down Rules

keeping of cornerstone cases ensure that the knowledge is used always in the same context when it was added.

The tree structure of RDR Knowledge Base is shown in Figure 1. Each node in the binary tree is a rule with desired conclusions. Each node has a cornerstone case associated with it, the case that prompted the inclusion of the rule. The classification (conclusion) comes from the last rule that was satisfied by the current data. The knowledge acquisition process in RDR is illustrated in Figure 2. To add a new rule into the tree structure, it is necessary to identify the location and the condition of the rule. When the expert wants to add a new rule, there must be a case that is misclassified by a rule which was applicable to the current case. The system asks the expert to select conditions from the difference list between these two cases: misclassfied case and the cornerstone case. Then the misclassified case is stored as the refinement case (new cornerstone case) with the new rule whose condition part distinguishes these cases.

3 Default Knowledge in Ripple Down Rules

The rule at the root node in RDR Knowledge Base, which is represented by a binary tree, is special. Its condition part is empty (it means that it is always satisfied) and its consequent part is a special conclusion called "default class". Here, the default class means a tacit conclusion for the current case when no other conclusion of the case is derived from the knowledge base.

Suppose that a conclusion A is set as the default class in RDR and a training case which has the conclusion A is supplied to the system. Even if no rule in the knowledge base fires for the case, the system correctly classifies the case using the default class A. Although the classification is correct, no explicit knowledge that

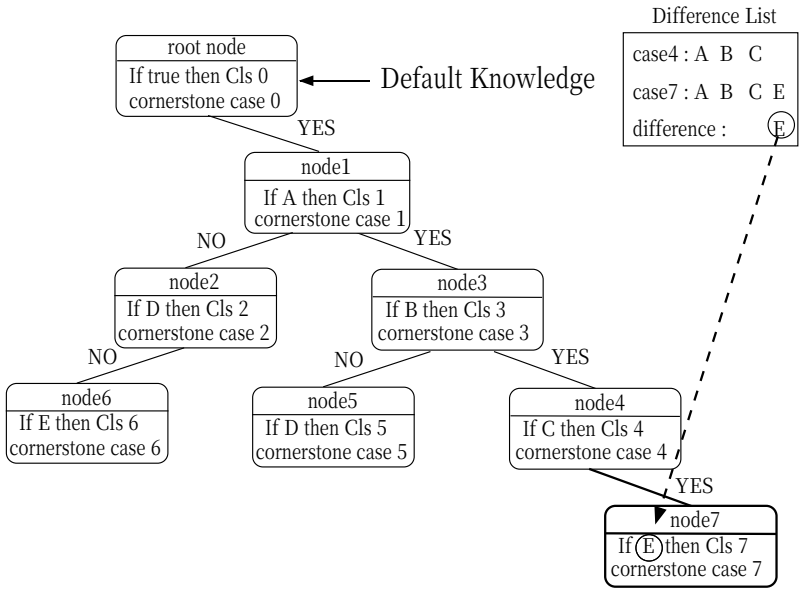

Fig. 2. Knowledge acquisition in Ripple Down Rules

characterizes the conclusion A is acquired. It is simply saying that since there is no conflicting knowledge, the conclusion must be A. Therefore, a different setting of the default class yields a different knowledge base with different performance (accuracy and size) even for the same problem domain.

The root node has only YES branch (Figures 1 and 2), while all the other nodes have both YES and NO branches. All rules below the root node are the exception rules of the root node. More generally, all rules below a certain node is the exception rules of that node.

In the previous research it was common that the majority class in the domain was assumed as the default class[8]. Majority class in the domain means the class of the most frequent cases in the data set. This looks natural because we only need to acquire rules (knowledge) for the other (minority) classes as the exceptions of the default class and some refinement rules for the default class. The size of the knowledge base is expected to be compact. The majority class does not necessarily mean that it is easy to describe and characterize that class. The same is true for non-majority classes.

It is worth investigating how differently an RDR system behaves in terms of accuracy and size when a different setting of the default class is made. There are the same number of settings of the default class as are the number of classes in the domain. We discuss the reason why and how different performance of an RDR system is obtained with reference to the knowledge characterization of an expert. In this study a simulated expert is used as a replacement of human expert[5].

Another important aspect which the study of RDR so far has not addressed much is noise handling of data. It is always assumed that an human expert is available and judges the quality of data. In real world problems, noise is abound and we cannot get rid of it. We have to admit that human experts make mistakes. Further, there are always cases where some of the feature values are missing.

It is known that one of the merits of RDR is that it can acquire the majority of knowledge at an early stage of knowledge acquisition compared by a standard empirical machine learning approach[11]. Intervention of human expert is the source of this high performance. How will these good properties be affected or not affected with the introduction of noise and a selection of default class is the main focus of the current study.

4 Experiments

4.1 Data Sets

We have selected three data sets from University of California Irvine Data Repository[12]. The attribute values in the data sets are all nominal and no continuous numeric attributes are used. This is to simplify the analysis, and how the results obtained by these limited data sets can be generalized must await for further study although we expect a similar conclusion for other data sets.

1) Tic-Tac-Toe: Tic-Tac-Toe Endgame Database. This database encodes the complete set of possible board configuration at the end of the game. There are 9 attributes for 958 cases and 2 classifications which are *positive* (win) and *negative* (loss).

2) Car: Car Evaluation Database. This database was derived artificially from a simple decision model for the evaluation of car acceptability. There are 6 attributes for 1728 cases and 4 classifications which are *unacc* (unacceptable), *acc* (acceptable), *good* and *vgood* (very good).

3) Nursery: Nursery Database. This database was derived from a hierarchical decision model originally developed to rank applications for nursery schools. There are 8 attributes for 12960 cases and 5 classifications which are *not_recom* (not recommend), *recommend*, *very_recom* (very recommend), *priority* and *spec_prior* (special priority).

4.2 Experimental method

We have conducted parametric studies using a simulated expert [5] for the above three data sets. For each default class, we have evaluated the classification accuracy and knowledge base size with different noise levels and different training data size. The results are compared for these parameters and with other machine learning method.

Simulated expert As a simulated expert, we use C4.5[13]. We run C4.5 with a default setting over an entire data for each noise level and for each data set. This is the maximum attainable performance that can be induced from the given data and we use this as the knowledge (expertise) of an expert. Here, we use the induction rule set which is translated from the decision tree derived by C4.5.

While a human expert checks the output of the RDR system and, when the conclusion is wrong, generates a new rule by selecting conditions from the difference list, a simulated expert similarly selects conditions from the difference list based on machine learning techniques. All conditions from the intersection of C4.5 rule conditions for the case and the difference list for the case are selected in this study. It should not be expected that the simulated expert will perform better than a real human expert. For noisy data, the simulated expert may misclassify some of the cases. Though a human expert may perform better than a simulated expert in this case, we adopt the classification by the simulated expert as the conclusion made by the expert. It is also noted that the simulated expert has small error rate even when the data is noise-free. This is because pruning is automatically made in C4.5.

Data preparation Since RDR accepts data sequentially, the order of incoming data affects the performance. The whole data for each domain was reordered by random sampling and ten different data sets are generated to cancel out this ordering effect. For each data set of a fixed order, the first 75% of the data is taken as training data and the remaining 25% of the data as test data. To see the effect of how rapidly RDR can acquire the correct knowledge, the predicted error rate of test data is periodically evaluated at the points where 1%, 2%, 4%, 5%, 7%, 10%, 15%, 20%, 30% 45%, 50%, 60% and 75% (all) of the data are used.

Accuracy The predicted error rate is used as a measure of accuracy. It is the average of the error rates of the remaining 25% of the ten data sets, each with a different ordering.

Noise handling The noise was randomly added only to the attribute values and not to the class information with a fraction of 2%, 4%, 6%, 8%, 10%, 12%, 14%, 16%, 18% and 20%. Adding noise is equivalent to randomly changing the value of the features to other alternatives. The evaluation was performed at two points where 5% and 50% of all cases are given as a training set. This is to see the effect of noise both at an earlier stage of knowledge acquisition where the system has not seen enough data and a later stage where the system's performance is approaching to an equilibrium.

Machine learning method RDR knowledge base is compared with the knowledge base built using a standard machine learning method C4.5 [13]. Since the learning by C4.5 is not incremental, we run C4.5 at the same data point where we check the accuracy of RDR knowledge base. This means that at an early stage, C4.5 must learn with small data set. The test data set is the same as the one used for RDR testing (the last 25% data).

Size of knowledge base The size of the knowledge base is defined respectively to be the sum of the decision node in RDR binary tree and the number of induction rules in C4.5. This is because a decision node in RDR corresponds to an if-then rule.

4.3 Results

Tables 1 $\sim$ 3 summarize the data characteristics and the main results. The first three rows are 1) the number of cases for each class (*i.e.* what the majority class is), 2) the number of induction rules and 3) the error rate of the simulated expert C4.5 (as calculated by the command *C4.5rules*).

Figures 3 and 4 show the results of the machine learning method C4.5 and the four kinds of RDR each with different default class when trained by various data size in Car data set. The graphs respectively show the improving performance (error rate) and increasing size of the knowledge base as the more data are available. From Figure 3, it is seen that the learning speed of RDRs are faster than that of C4.5. This is because RDR acquires the knowledge from the simulated expert who has enough knowledge of all cases in the problem domain.

Among the four kinds of RDR, the learning speed of RDR(*acc*) whose default class is *acc* is fastest, and the size of the knowledge base of RDR(*acc*) is smallest as can be seen in the corresponding learning curves. It should be noted that RDR which has the majority class as the default class does not necessarily show the best performance. The majority class in this data set is *unacc*. This is also summarized in Table 2. Since it is not necessary to acquire the knowledge about the default class in RDR, RDR(*acc*) does not acquire the knowledge about the class *acc* but acquires the knowledge about the other classes, more precisely the knowledge to differentiate these classes from the class *acc*. The fifth and sixth rows show the ranking of the learning speed and the size of knowledge base for each default class. Note that the error rate of the simulated expert is lowest and the number of rules that are required to describe the class is largest for the class *acc*, which ranks the first. It means that RDR(*acc*) quickly acquires the knowledge about the other classes which happens to have higher error rates and do not require large description length. It does not acquire the explicit knowledge about the default class *acc* which has low error rate. The curves for the other defaults (*unacc, good, vgood*) are nearly the same, *vgood* being the worst. In Table 2 these are ranked either 2 or 3. The default class *vgood* has the highest error rate of the simulated expert and has the second smallest rule size to describe the class.

From this observation, we can conjecture that the more induction rules are required to describe the class and the lower the error rate of the simulated expert for the default class is, the faster the learning speed of RDR is and the smaller the size of RDR knowledge base is. By having a class that requires more induction rules to describe itself as the default we can avoid adding these many rules later in the knowledge acquisition process. Likewise by having a class for which the predicted error is smaller, the refinement rule for the default class can be more error free and the probability of recursive exception taking place would be less.

Table 1. Tic-Tac-Toe Endgame Database

Class	*positive*	*negative*
Number of cases	626	330
Number of induction rules	8	10
Error rate of simulated expert (%)	0.0	0.0
Minimum description length (bits)	54.7	147.4
Learning speed (ranking*)	2	1
Size of knowledge base (ranking*)	2	1
Noise tolerance (ranking*)	1	2

* Smaller, the better. Both the speed and the size are evaluated at 20% of the total data seen, and the noise tolerance at 10% noise level.

Table 2. Car Evaluation Database

Class	*unacc*	*acc*	*good*	*vgood*
Number of cases	1210	384	69	65
Number of induction rules	11	53	15	15
Error rate of simulated expert (%)	2.7	1.3	7.7	15.4
Minimum description length (bits)	261.4	638.2	271.3	232.1
Learning speed (ranking*)	3	1	2	3
Size of knowledge base (ranking*)	2	1	2	2
Noise tolerance (ranking*)	2	1	3	3

* Smaller, the better. Both the speed and the size are evaluated at 20% of the total data seen, and the noise tolerance at 10% noise level. When the difference is small, the rank is rated the same.

Table 3. Nursery Evaluation Database

Class	*not_recom*	*rec'mend*[1]	*v_recom*[2]	*priority*	*s_prior*[3]
Number of cases	4320	2	328	4266	4044
Number of induction rules	1	0	62	187	132
Error rate of simulated expert (%)	0.0	-	3.0	1.2	0.6
Minimum description length (bits)	15.9	-	907.5	2021.1	1309.5
Learning speed (ranking*)	4	4	3	2	1
Size of knowledge base (ranking*)	4	4	3	2	1
Noise tolerance (ranking*)	3	3	3	2	1

[1] recommen, [2] very_recom, [3] spec_prior
* Smaller, the better. Both the speed and the size are evaluated at 20% of the total data seen, and the noise tolerance at 10% noise level. When the difference is small, the rank is rated the same.

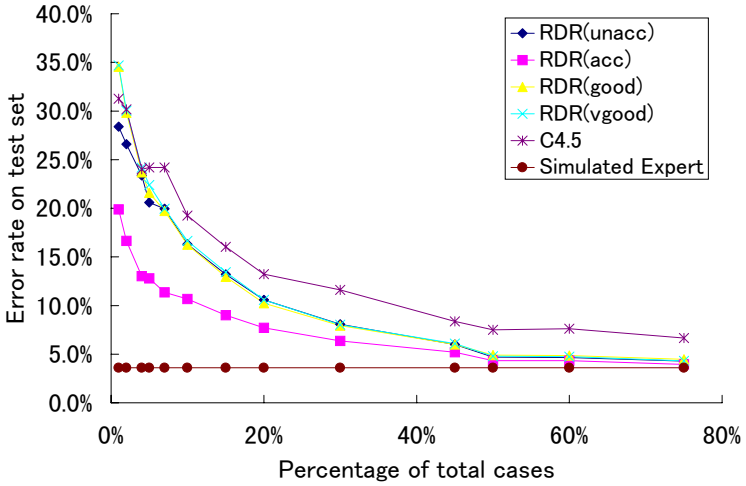

Fig. 3. Error rate for various sized training sets in Car data set. For example, RDR(*unacc*) means RDR whose default class is *unacc*.

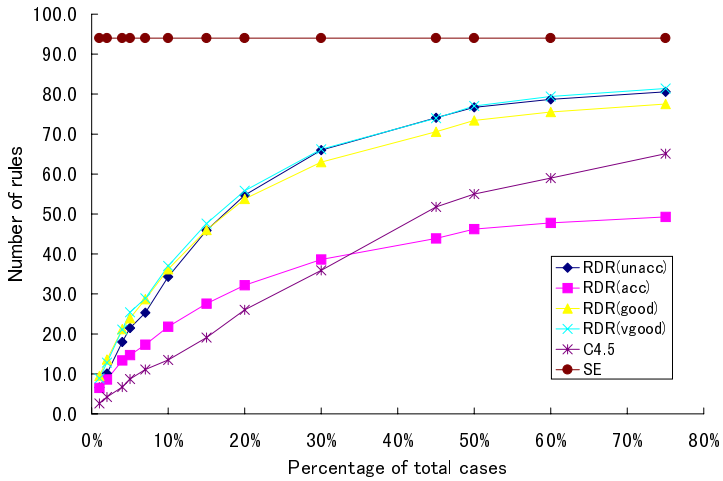

Fig. 4. Size of the knowledge base for various sized training sets in Car data set

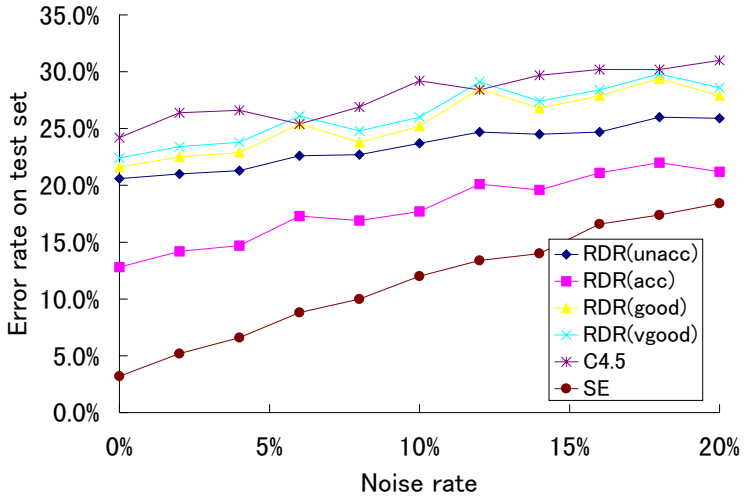

Fig. 5. Error rate for various rate of noisy data in Car data set when 5% of all cases are given as training set.

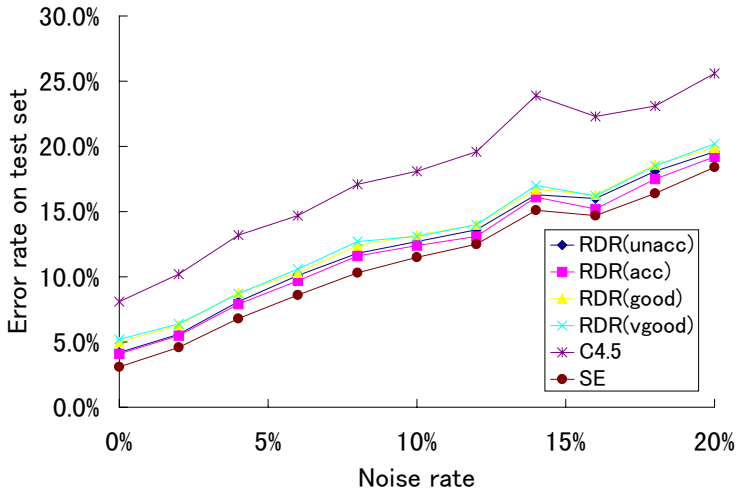

Fig. 6. Error rate for various rate of noisy data in Car data set when 50% of all cases are given as training set.

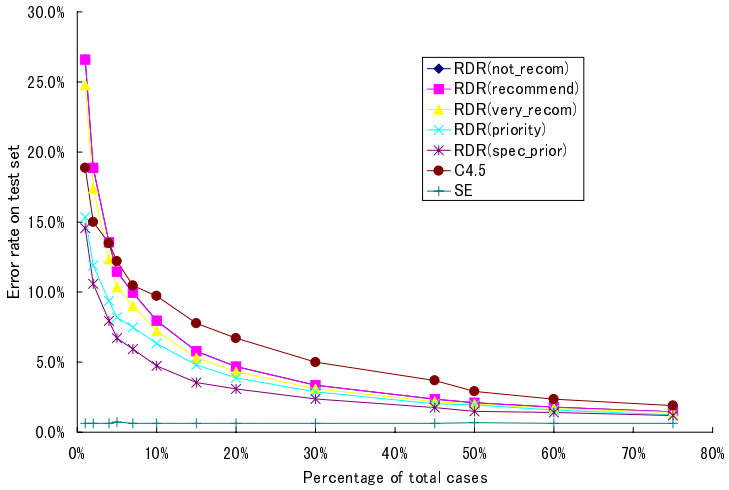

Fig. 7. Error rate for various sized training sets in Nursery data set

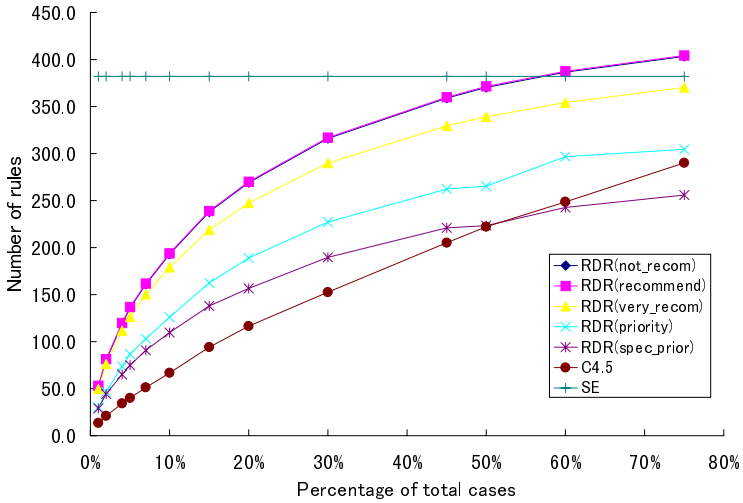

Fig. 8. Size of the knowledge base for various sized training sets in Nursery data set

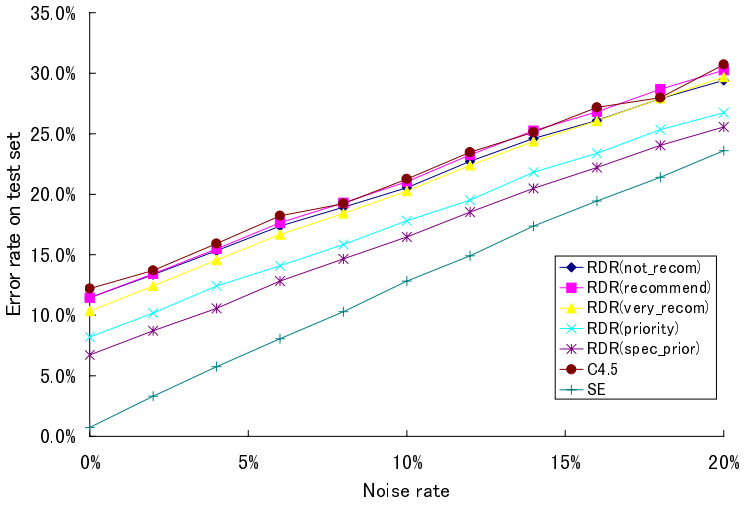

Fig. 9. Error rate for various rate of noisy data in Nursery data set when 5% of all cases are given as training set.

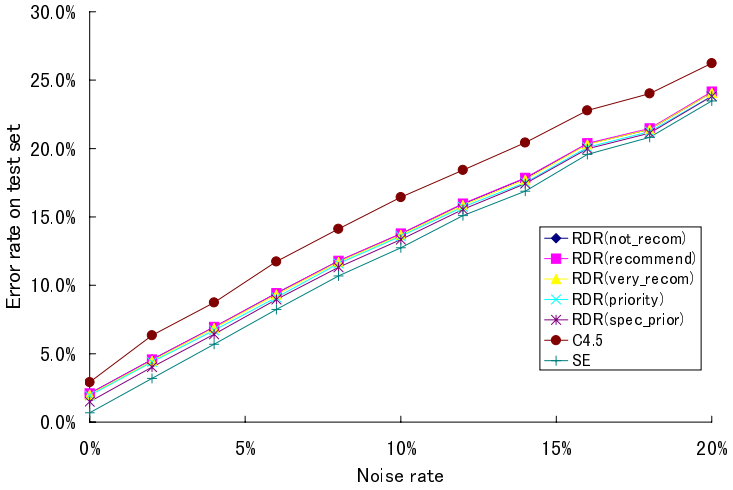

Fig. 10. Error rate for various rate of noisy data in Nursery data set when 50% of all cases are given as training set.

In order to quantify the above observation, we introduce a new index to consider both the number of rules and the error (misclassification) rate of the simulated expert for each default class, namely "*minimum description length*"as shown in the fourth row in Tables 1 $\sim$ 3.

This index is calculated as the sum of the coding costs for induction rules and misclassified cases. The number of bits required to encode the induction rules is

$$\sum_{i=1}^{N}\{\sum_{j=1}^{M_i}(-\log_2 p_{ij}) - \log_2(M_i!)\}$$

where N is the number of rules, M_i is the number of conditions in the rule R_i, p_{ij} is the probability for which the condition C_j is satisfied in the rule R_i for all training cases. The number of bits required to encode the misclassified cases is

$$\log_2\left(\binom{r}{fp}\right) + \log_2\left(\binom{n-r}{fn}\right)$$

where the rules cover r of the n training cases with fp false positives and fn false negatives. Here, we used a weighting factor of 0.5 to the former because it is known that the rule coding length as defined in the above equation is overestimated. This value is the default setting of C4.5.

The total encoding length for rules and exceptions is *minimized* for each class to find the best subset of rules[2]. The result of Table 2 suggests that RDR performs best in terms of the speed of knowledge acquisition and the size of knowledge base when the default is taken to be the class for which the minimum description length is *maximum* (*heuristic principle of maximum minimum description length*). It is also shown that RDR behaves worst in terms of its accuracy and size when the default is taken to be the class for which the minimum description length is minimum. This relation holds for noisy data as discussed below.

Figures 5 and 6 show the results of the machine learning method C4.5 and the four kinds of RDR for different default classes with increasing rate of noisy data. It is natural that the error rate of all RDRs and C4.5 become higher as more noisy data are included. The error rate of RDRs is lower than that of C4.5 even with noise, the reason being the same as before.

It is true that, even for noisy data, among the four kinds of RDR, the error rate of RDR(acc) whose default class is acc is smallest because RDR(acc) quickly acquires the knowledge about the other classes which have high error rate. The ranking order is about the same as with the speed and the size. Similar relation holds with noisy data to the goodness of default class. The more induction rules are required to describe the class and the lower the error rate of the simulated expert for the default class is, the lower the error rate of RDR is.

Figure 7 $\sim$ 10 are for Nursery data set. In this case, the heuristic measure gives the second ranking, but not the third or fourth. There seems to be a trade

[2] This is what C4.5 does in inducing rules from the decision table.

off between the number of rules and misclassification rate. In case of Tic-Tac-Toe for which the resulting figures are omitted because it has only two classes the measure gives the best default for the speed and the size.

The proposed measure is based on the qualitative analysis of the RDR tree, and is certainly heuristic, but it appears that this can be a useful measure to characterize the goodness of default class.

5 Discussion

Error rate of simulated expert for each class

We have evaluated RDR using the simulated expert that is a computer program based on a machine learning technique instead of using a human expert. Therefore, the error rate of the simulated expert for each class can be derived in advance, and it is shown that RDR whose default class has 1) the lowest error rate of the simulated expert and 2) the largest number of rules to describe demonstrates the best performance in terms of accuracy and size among the different settings of the default class.

As RDR is a methodology to acquire the useful knowledge from vague and disordered knowledge of a human expert by forcing him/her to place in a specific context, it may not be the case that the expert knows clearly the error rate for each class in advance. However, there must be some difference in certainty or confidence for the knowledge which the expert has for each class, and the result in the previous section shows that it is better to use the most informative (confident) class for the expert as the default class in RDR. This dose not necessarily mean to use the majority class.

Handling of missing values in RDR

We have not so far discussed the missing value problems. We have tested two ways of handling the missing values in RDR. One is to create a new attribute value *'unknown'* and the other is to interpret the case with missing values to be false (not match) to the condition of the rule. However, the results were not convincing.

We are reevaluating missing value problem in RDR in the framework of the techniques used in machine learning methods. That is, we are following C4.5 in adopting a probabilistic approach. If a rule node is tested by an instance for which some relevant attribute value is unknown, the outcome of the test is not determined. The system has to explore both outcomes of YES and NO branches, assigning a weight representing the probability that the case belongs to each subset. Since there can be multiple paths from the root of a tree to the leaves, a classification is a class distribution rather than a single class and the class with the highest probability is regarded as the final conclusion. In the knowledge acquisition process of RDR, it is necessary to identify the location and the condition of the rule in order to add a new rule into the tree structure. As there are multiple paths in the inference process of RDR, there are multiple

candidates to add the new rule. We think it is natural to choose the node with the highest probability value to add it. The condition of the rule can be selected from the difference list by human or simulated expert as usual. The problem is the sensitivity of the weight values which have to rely on the statistics. As the RDR system sees more data, more probable weights can be estimated, but updating the weight may affect the previously tested voting results. We have yet to see the results about how this well or bad this works.

6 Conclusion

This paper demonstrated experimental results about the effect of the selection of default knowledge and the amount of noise in data on the performance of RDR which is a knowledge acquisition technique without the need of analysis or intervention of a knowledge engineer. Because of the patching principle in RDR, the knowledge acquired strongly depends on what is given as the default knowledge.

It is clearly shown that RDR shows good performance in terms of its accuracy and size when the default is taken to be the class for which the minimum description length of induction rules and error cases is maximum (*heuristic principle of maximum minimum description length*). This is the situation where the default class in RDR is taken to be the class for which the number of rules to describe is largest and the error rate of the simulated expert is lowest. This property also holds even when the data are noisy.

The experiments also show that the good properties of RDR that it can acquire the majority of knowledge at an early stage of knowledge acquisition faster than a standard empirical machine learning approach also holds for noisy data. The effect of noise is sensitive at an earlier stage of knowledge acquisition where its performance strongly depends on the default knowledge, but RDR eventually converges to its stable performance although the size of knowledge base is strongly affected by the chosen default knowledge.

The above characteristics of RDR will be favorable to build a knowledge-bases system which assumes a network environment, where multiple users and experts interacts with each other through the computer network. Acquiring or updating knowledge from multiple experts in a network environment is a new challenge.

Acknowledgement

Authors benefited much through the discussions with Dr. B. H. Kang in Hoseo University (Korea).

References

1. P. Compton, K. Horn, J. R. Quinlan and L. Lazarus. Maintaining an Expert System. In *Application of Expert Systems*, ed. J. R. Quinlan, pp. 366–385, Addison Wesley, 1989.

2. P. Compton and R. Jansen. A Philosophical Basis for Knowledge Acquisition. *Knowledge Acquisition*, Vol. 2, pp. 241–257, 1990.
3. P. Compton, G. Edwards, B. H. Kang, et al. Ripple Down Rules: Possibilities and Limitations. *Proc. of the 5th Knowledge Acquisition for Knowledge-Based Systems Workshop*, 1991.
4. P. Compton, G. Edwards, et al. Ripple Down Rules: Turning Knowledge Acquisition into Knowledge Maintenance. *Artificial Intelligence in Medicine*, Vol. 4, pp. 47–59, 1992.
5. P. Compton, P. Preston, B. H. Kang. The Use of Simulated Experts in Evaluating Knowledge Acquisition. *Proc. of the 9th Knowledge Acquisition for Knowledge-Based Systems Workshop*, 1995.
6. B. Gaines and M. Shaw. Cognitive and Logical Foundations of Knowledge Acquisition. *Proc. of the 5th Knowledge Acquisition for Knowledge-Based Systems Workshop*, 1990.
7. B. H. Kang and P. Compton. Knowledge Acquisition in Context: Multiple Classification Problem. *Proc. of PRICAI'92*, Vol. 2, pp. 847–853, 1992.
8. B. H. Kang. Validating Knowledge Acquisition: Multiple Classification Ripple Down Rules. *Ph.D Thesis, University of New South Wales*, 1996.
9. B. H. Kang, K. Yosida, H. Motoda, P. Compton Help Desk System with Intelligent Interface. *Applied Artificial Intelligence, Vol. 11, pp. 611–631*, 1997.
10. J. L. Kolodner. A Process Model of Case-Based Reasoning in Problem Solving. *Proc. of IJCAI'85*, Vol. 1, pp. 284–290, 1985.
11. Y. Mansuri, J. G. Kim, P. Compton, C. Sammut. An Evaluation of Ripple Down Rules. *Proc. of AKAW'91*, 1991.
12. P. M. Murphy and D. W. Aha. UCI Repository of Machine Learning Databases. ftp://ftp.ics.uci.edu/pub/machine-learning-databases/, 1994.
13. J. R. Quinlan. *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.
14. J. R. Quinlan. Induction of decision trees. *Machine Learning*, Vol. 1, pp. 81–106, 1986.
15. Z. Ramadan, P. Compton, P. Preston, T. Le-Gia, V. Chellen, M. Mulholland, D. B. Hibbert, P. R. Haddad, and B. Kang. From Multiple Classification RDR to Configuration RDR. *Proc. of the 10th Knowledge Acquition for Knowledge-Based System Workshop*, 1997.
16. R. C. Schank. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*, Cambridge University Press, 1982