An Online Classification and Prediction Hybrid System for Knowledge Discovery in Databases

Richard Dazeley
School of Computing
University of Tasmania
Sandy Bay, Tasmania, Australia
rdazeley@utas.edu.au

Byeong Ho Kang
Smart Internet Technology Cooperative Research Centre, Bay 8, Suite 9/G12
ATP Eveleigh NSW 1430, Australia
bhkang@utas.edu.au

ABSTRACT

Expert Systems in data mining generally use knowledge extraction methods to form a classifier or predictor. These have the advantage of forming high quality results due to the inclusion of expert knowledge. The problem, however, is that they do not allow for autonomous knowledge discovery. Therefore, such systems will only find results that the expert is capable of giving examples about and will not find unknown patterns. Thus, Knowledge Discovery in Database (KDD) generally relies on other machine learning tools, forgoing the advantage of expert knowledge. This paper presents a method for incrementally building a knowledge base using expert knowledge in such a way that potentially the system is still able to autonomously discover unknown patterns. It does this through prudence checking of the knowledge base for each case presented and informs the expert when the knowledge base has an inconsistency needing clarification. Initial results show strong potential for the system in identifying mis-classifications allowing its potential application to KDD.

1. INTRODUCTION

The world is becoming more like the infinite library from *The Library of Babel* [1] every day, overflowing with data from which people are unable to extract meaningful information. Knowledge discovery in databases (KDD) is a field of research attempting to solve this dilemma and is seen as the process of extracting "…implicit, previously unknown and potentially useful knowledge from data"[2]. One area of research within KDD is concerned with the ability to find meaningful classifications and predictions of values for the tuples contained within a database.

One method that is highly effective at automatically generating rule bases, capable of classifying and predicting values from smaller data sets is decision trees. However, while in theory decision trees could scale effectively to larger databases, due to only having $n(\log n)$ complexity, they suffer from requiring the training set to be resident in memory [3]. More recent systems such as SLIQ and SPRINT have attempted to address this issue. However, they require pre-sorting of data sets as well as complex and expensive data structures that reduce their effectiveness with large training sets [3].

This paper describes an augmented hybrid system, called Rated MCRDR (Multiple Classification Ripple Down Rules [4]) or simply RM. This method provides a means for a domain expert to incrementally build a knowledge base that can be used for classification and prediction in a large database, without the need for pre-sorting or the retention of large training sets in resident memory. Additionally, the expert only needs to review the occasional special case, providing a semi-automated process with the advantage of built in expert knowledge. The system takes MCRDR conclusions and their justifications and feeds them into a purpose built RBF Neural Network and trained using the single-step-Δ-update-rule [5]. RM is capable of classifying database tuples into single or multiple classifications. Additionally, RM has been shown to be able to provide a prediction or evaluation for continuous value ranges [5].

MCRDR has previously been shown to be a highly effective incremental learning Knowledge Based System (KBS)[4, 6, 7]. It allows a domain expert to add rules online by providing justifications identifying the differences between cases within the context provided. The methodology has also been shown to provide significantly more compact rule bases than decision tree based systems such as C4.5 [4, 8]. Additionally, due to its lack of requiring a training-set it does not suffer the memory problems inherit in decision tree systems. Thus, with the exception of one major drawback, MCRDR could provide a highly effective system for knowledge discovery in large databases. The inherit problem with using MCRDR for KDD is that the human expert must be in a position to review each and every case to ensure that it is classified correctly. Therefore, this does not scale to large databases and clearly prevents its application as a data-mining tool.

RM, however, provides a solution, by also using a prudence check that is able to warn the expert when it recognises that the existing knowledge base does not adequately classify a case. It achieves this by giving a confidence-*like*-factor with its final classification. The prudence system then decides, based on this confidence level, whether the current knowledge base was inadequate and, if so, alerts the user to the potential misclassification. Therefore, the expert is no longer required to check every case and now is only required to check cases that have generated warnings.

2. Multiple Classification Ripple Down Rules (MCRDR)

MCRDR uses an n-ary tree where each node contains a rule. Inference is performed by passing each case to the root node, which in turn feeds it on to any children with rules that evaluate it to *true*. Thus, the case continues to ripple down, level by level, until either a leaf node is reached or all of the child rules evaluate to false. Due to the fact that any, or all, of a node's children have the potential to fire, the possibility exists for a number of conclusions or classifications to be reached by the system for each case presented [7]. The system then lists the collection of classifications and the paths they followed.

Knowledge Acquisition is achieved in the system by inserting new rules into the MCRDR tree when a misclassification has occurred. The new rule must allow for the incorrectly classified case, identified by the expert, to be distinguished from the existing stored cases that could reach the new rule [9]. This is accomplished by the user identifying key differences between the current case and one of the earlier cornerstone cases. Where, a cornerstone case is any case that was used to create a rule and was also classified in the parent's node, or one of its child branches, of the new node being created. This is continued for all stored cornerstone cases, until there is a composite rule created that uniquely identifies the current case from all of the previous cases that could reach the new rule. The idea here is that the user will select differences that are representative of the new class they are trying to create [9].

3. Rated MCRDR (RM)

Individual classifications in MCRDR, however, are all uniquely derived with no consideration for what other classification paths may have also been followed. Thus, there is no cohesion between any of the classifications found, however, the fact that this case was classified in these classes; means there must be either a conscious or subconscious relationship between these cases in the experts mind. The intention of RM is to try to capture these relationships between various classifications that may exist, either consciously or otherwise. If we can identify a set of relative values for the various relationships, this information could be used to improve the functionality available to the user. For instance, in a KDD tool, one value could be trained to fill in a missing value for a particular missing field; or, identify if this case is one of particular interest to the expert. Most importantly though, and the focus of this paper, this value can be used to identify a confidence-*like*-factor. This could then be used to identify when the system should issue a warning that the case is unfamiliar and the conclusions should be checked by the expert. This system, referred to as RM-KAW (Knowledge Acquisition Warnings) allows the expert to incrementally build the KB without having to review each and every case presented.

3.1. Implementation

Firstly, looking at what RM must accomplish mathematically, it can be seen that the output from the MCRDR methodology is essentially a set of classifications, denoted *C*, where $C \in \wp(C^*)$, and *C** is the set of all possible classifications. The output from the RM engine is a set of values, $\bar{v}$, to provide one or more varying results in applications where dissimilar tasks may need to be rated differently. For instance, v_0, may identify the desirability, importance or confidence in its own classification for the case presented. Therefore, a mapping must be found from the set $C \rightarrow \bar{v}, \forall C \in \wp(C^*)$. Additionally, RM should be able to learn this mapping for both linear and non-linear sets of classifications quickly and be able to generalise effectively.

Thus, RM needs to identify patterns of classifications and then associate a value for each pattern. While there are a number of techniques used for pattern recognition, in this implementation of RM an Artificial Neural Network (ANN) was selected, primarily because of its adaptability, ease of application to the problem domain, and because pattern recognition is one of the dominating areas for the application of ANN's [10].

The neural network was integrated into MCRDR by linking each possible rule or class to an input neuron. Then, for each rule or classification found by the MCRDR system, an associated neuron will fire. In this implementation of RM a purpose built resource-allocating radial basis function (RBF) network was used. The output nodes use the standard sigmoid thresholding function, equation 1, using a *modified* generalised delta rule. A subset of the input nodes is selected by the hidden layer by using the Gaussian function, equation 2, where the distance measure *r* is taken to be Euclidean, equation 3.

$$f(net) = 1 \big/ \left(1 + e^{-k\ net}\right) \qquad (1)$$

$$\phi(r) = e^{\left(-r^2\right)} \qquad (2)$$

$$\|x - y\| = \sum_i (x_i - y_i)^2 \qquad (3)$$

There are three possible methods for the association of neurons to the MCRDR structure: the Class Association method (CA), the Rule Association method (RA) and the Rule Path Association method (RPA). The different methods arise from the possibility of many paths through the tree that result in the same class as the conclusion. The *class association* method, where each unique *class* has an associated neuron, can reduce the number of neurons in the network and potentially produce faster, but possibly less general, learning. The *rule association* method, where each *rule* has an associated neuron and only the terminating rule's neuron fires, allows for different results to be found for the same class depending on which path was used to generate that class as the conclusion. Therefore, it is more capable of finding variations in meaning and importance within a class than may have been expected by the user that created the rules. The *rule path association* method, where all the *rules* in the *path* followed, including the terminating rule, cause their associated neuron to fire, would be expected to behave similarly but may find some more subtle results learnt through the paths rules, as well as being able to learn meaning hidden within the paths themselves.

Thus, the full RM algorithm, given in pseudo code in Figure 1 and shown diagrammatically in Figure 2, consists of two primary components. Firstly, a case is pre-processed to identify all of the usable data elements, such as stemmed words or a patient's pulse. The data components are presented to the standard MCRDR engine, which classifies them according to the rules previously provided by the user. Secondly, for each rule or class identified an associated input neuron in the neural network will fire. The network finally produces a set of outputs, $\overline{v}$, for the case presented. The system, therefore, essentially provides two separate outputs; the case's classifications and the associated set of values for those classifications.

```
1.  Pre-process Case
        Initialise Case c
        c ← Identify all useful data elements.
2.  Classification
        Initialize list l to store classifications
        Loop
            If child's rule evaluates Case c to true
                l ← goto step 2 (generate all classifications in child's branch).
        Until no more children
        If no children evaluated to true then
            l ← Add this nodes classification.
        Return l.
3.  Rate Case
        $\overline{i}$ ← Generate input vector from l.
        NN ← $\overline{i}$
        v ← NN output value.
4.  Return RM evaluation
        Return list l of classifications for case c and
        Value v of case c.
```

Figure 1: Inference Algorithm for RM.

For example, in Figure 2, the document {a b b a c f i} has been pre-processed to a set of unique tokens {a, b, c, f, i}. It is then presented to the MCRDR component of the RM system, which ripples the case down the rule tree finding three classifications: *Z*, *Y*, and *U*; from the terminating rules: 1, 5, and 8. In this example, which is using the RA method, the terminating rules then cause the three associated neurons to fire and feed forward through the neural network producing a single value of 0.126. Thus, this document has been allocated a set of classifications that can be used to store the document appropriately, plus a value, which in the case of RM-KAW, can be used as a measure of confidence in the systems conclusion.

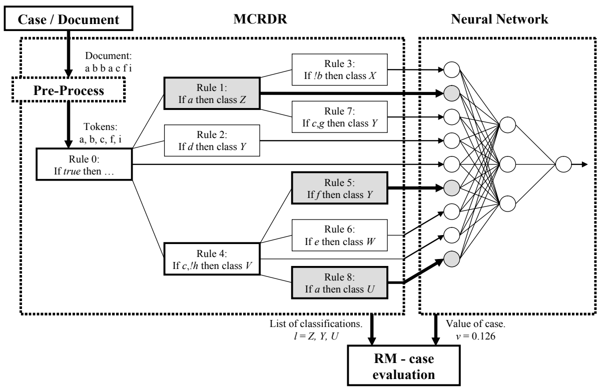
Case / Document
MCRDR
Neural Network
Document: a b b a c f i
Pre-Process
Tokens: a, b, c, f, i
Rule 0: If true then …
Rule 1: If a then class Z
Rule 2: If d then class Y
Rule 3: If !b then class X
Rule 7: If c,g then class Y
Rule 4: If c,!h then class V
Rule 5: If f then class Y
Rule 6: If e then class W
Rule 8: If a then class U
List of classifications. l = Z, Y, U
Value of case. v = 0.126
RM - case evaluation

Figure 2: RM illustrated diagrammatically.

3.2. LEARNING IN RM

Learning in RM is achieved in two ways. Firstly, the rating component receives feedback from the environment concerning the accuracy of its predicted rating. Thus, a system using RM must provide some means of either directly gathering or indirectly estimating the correct rating. For example, in RM-KAW feedback with a high value may be given if the conclusion produced by the system is not changed by the expert. Secondly, the MCRDR component still acquires knowledge in the usual way; by the user identifying incorrect classifications and creating new rules and occasionally new classifications.

3.3. NEURAL NETWORK STRUCTURE AND LEARNING

The neural network selected in developing RM for KAW was the standard radial basis function where the output nodes use the standard sigmoid thresholding function, equation 1, and the hidden layer uses the Gaussian function, equation 2, with a Euclidean distance measure, equation 3. Further experimentation still needs to be carried out to determine if other functions would provide better results. For instance, due to the discrete inputs currently used, it is expected that a broader fitness function at the hidden layer may provide better generalisation.

Some variations to the standard RBF network were required, due to the creation of additional rules and classifications, described earlier. Primarily, the capability to increase its number of input nodes to ensure one input node for each possible rule or class. Likewise, due to the growing nature of the input space, it can be expected that the number of significant patterns would also increase. Thus, a system was also developed for automatically allocating addition resources, in the form of hidden nodes.

A new input node is added to the system only when the user has corrected a conclusion. This, therefore, also means the user has identified a new significant pattern (whatever the new input vector is after the correction), which should automatically be captured in the hidden layer. Thus, a new hidden node is added with appropriate input weights assigned that will produce a Euclidean distance of zero only for that input sequences. However, the weight assigned to determine the contribution of this pattern to the overall confidence of future cases also must be found.

The simplest approach is to assign a random start up weight in the same fashion used when first initialising the network. However, we already have an accurate measure of the confidence for the new pattern, because the expert just created the new conclusion. Thus, we can be reasonably sure it is correct and assign the maximum confidence level for the new pattern.

Furthermore, in this implementation, additional hidden node resources are also added even when input nodes aren't added. This is done when a significant error is found, generally when a warning was made and the user decided that the conclusion was correct. If, when such a situation occurs, the system will check to see whether there are any nodes providing a reasonably close representation of the input pattern and if not a new hidden node is added with a Euclidean distance of zero. Once again a valid estimate can be made from the user's behaviour, providing a recommended value for the new patterns contribution to the system's overall confidence.

SINGLE-STEP-Δ-UPDATE-RUle

In order to calculate the weight needed for a new connection from a just created hidden node and each output node, w_{no}, the system uses the *single-step Δ-update-rule* [5]. Where the result that was required is passed back through the inverse of the sigmoid function, giving the weighted sum required by the network. The actual weighted sum that was originally calculated during the feed forward process is then subtracted from the required weighted sum. This is then divided by the input at the newly created hidden node, h_n, which is always one in this implementation due to it being assigned a Euclidean distance of zero for the given input vector.

$$w_{no} = \frac{\left(\log\left[\frac{f(net)_o + \delta_o}{1 - \left(f(net)_o + \delta_o \right)} \right] \Big/ k \right) - \left[\sum_{h=0}^{m} x_h w_{ho} \right]}{x_h} \quad \textbf{(4)}$$

4. TESTING RM

The problem with testing a system, such as RM, is the use of expert knowledge that cannot be easily gathered without the system first being applied in a real world application. This is a similar problem that has been encountered with the testing of any of the RDR methodologies [4, 8]. Thus, these systems built their KB incrementally through the use of a simulated expert. The simulated expert essentially provided a rule trace for each case run through another KBS with a higher level of expertise in the domain than the RDR system being trained [4, 7]. Testing of RM also relied on the use of simulated experts.

4.1. SIMULATED EXPERTS

Initial tests were carried out using a heuristic expert that randomly generates a table of values, representing the level that each possible attribute, $a \in A$, in the environment contributes to each possible class, $c \in C$. Two rules were used in generating the expert: each attribute contributes to one class positively (1 to 3) and one class negatively (-1 to -2) and the remaining classes are given a value of zero; secondly, each class has one positive and one negative attribute from every |*C*| number of possible attributes. When a case is presented to the expert it is tested to see which class it belongs by adding all the associated values for each attribute in each class. The expert will then classify the case according to which classes provided a positive total. The reason for setting up the expert in this way was to ensure that every case presented to the expert would be classified in at least one or more classes. An example of an expert used is shown in Table 1. When creating a new rule, the expert selects the attribute from the difference list that distinguishes the new case from the cornerstone case most. This expert was used for the multiple conclusion dataset.

	a	b	c	d	e	f	g	h	i	j	k	l
C1	0	0	-1	3	0	0	0	0	0	0	-1	3
C2	0	0	0	-2	2	0	0	-2	0	0	1	0
C3	0	-2	1	0	0	0	0	0	0	1	0	-1
C4	-1	3	0	0	0	0	1	0	-1	0	0	0
C5	0	0	0	0	-2	2	-2	0	2	0	0	0
C6	2	0	0	0	0	-2	0	1	0	-2	0	0

Table 1: Example of a randomly generated table used by the first simulated expert.

The second expert used was based on those used in other RDR testing environments [4, 8] using C4.5 to decide what rules should be created. Basically, attributes where selected from the difference list according to how high they were in the C4.5 tree. In the results below 2 attributes were selected each time a new rule was created. This expert was used for all the remaining single classification datasets tested.

4.2. DATASETS

The Chess and Tic-Tac-Toe datasets are from the University of California Irvine Data Repository [11].

MULTI-CLASSIFIABLE – This dataset builds cases by randomly selecting attribute from the environment. Each case has selected between 3 and 9 attributes giving a possible 3938 different cases plus 1062 repeated cases.

CHESS – Using the Chess end game of King+Rook (black) vs King+Pawn (white) on a7. There are 36 attributes and 3196 cases producing a binary classification.

TIC-TAC-TOE – This dataset uses the complete collection of possible terminating board configurations for Tic-Tac-Toe. There are 9 attributes and 958 cases producing a binary classification.

4.3. EXPERIMENTAL METHOD

RM was tested 10 times with each dataset randomly reordered. For each dataset tested there were small variations made to various learning parameters and the threshold used to identify when warnings were given. When classifying each case the level of confidence the system has in its conclusion was also gauged. The test carried out at this stage did not use this confidence directly. Instead, it simply gathered statistics on how accurate its predictions actually were. Thus, the simulated user actually still checked every case, ignoring any warnings, and created new rules whenever it found a case incorrectly classified. Rewards were given to the RM system depending on what it predicted and the action taken by the expert.

The aim at this stage was to test RM's ability to identify misclassifications. If successful this opens the methodology up to be used for data mining with expert knowledge. It also shows the systems ability to identify unusual cases. In the test discussed in this paper, one of the following four details where recorded for each case as it was processed:

- If the conclusion was correct and RM gave a warning, then a *False Positive* was recorded.
- If the conclusion was correct and RM did not give a warning, then a *True Negative* was recorded.
- If the conclusion was incorrect and RM gave a warning, then a *True Positive* was recorded.
- If the conclusion was incorrect and RM did not give a warning then a *False Negative* was recorded.

5. RESULTS AND DISCUSSION

Figure 3, shows the accuracy of RM in identifying errors in the multi-classifiable dataset. The solid line shows how many times the user corrected rules overtime. While the dotted line shows when a rule that the user corrected was warned about by RM; *true positives*. It can clearly be observed that RM was able to identify nearly all, 98% (± 1.34% at 95% confidence), incorrect classifications.

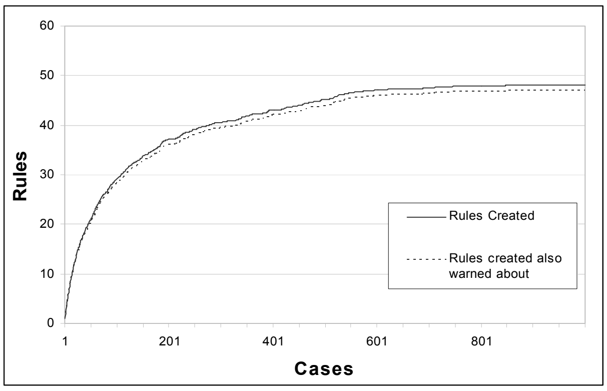

Figure 3: Comparison of the total rules created and the total rules first warned about then created over the first 1k multi-classifiable cases.

However, if the system simply provides a warning for every case then this result is meaningless. Therefore, RM would also need to minimise the amount of warnings generated when a case is correct, *false positive*, without reducing the amount of correct warnings. Figure 4, shows the average results from the multi-classifiable dataset for the four parameters being recorded. The *false positive* result illustrates how RM is highly successful at reducing the amount of incorrect warnings, continuing down to only 8.7% after 5000 cases, being generated as the knowledge base grows and RM learns. Furthermore, it has accomplished this without loosing its ability to identify the incorrect cases. It can also be seen that the few *false negatives* that did occur only occurred during the very early stages of the expert systems development. Thus, the expert would need to check all cases initially but could confidently only check cases that actually generate warnings later on significantly reducing their work. Similar trends were also evident with both the Chess and Tic-Tac-Toe datasets.

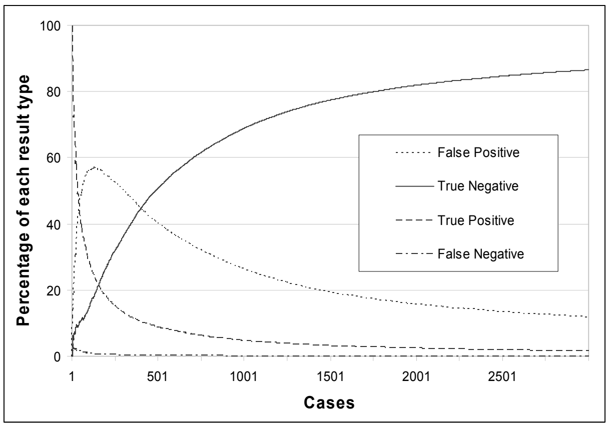

Figure 4: Comparison of the average percentage of cases that produced each of the four types of results over ten randomly generated trials.

The Chess and Tic-Tac-Toe tests where carried out to compare RM directly with Compton's prudence checking system [12]. Compton's system compared each attributes value with previously seen values for that same attribute and warned if they were different. While, effective it was concluded that the results were not sufficient for an expert to avoid reviewing cases. Table 2 shows a comparison of Compton's system with RM.

	False Neg %	True Pos %	True Neg %	False Pos %	Correctly Warned %
D-Multi-C	0.02	0.9	90	9	98
C-Garvan	0.2	2.4	83	15	92
D-TTT	0.2	2.1	80	18	91
C-TTT	1.5	3.8	81	14	72
D-Chess	0.07	0.5	85	14	87
C-Chess	0.3	1.3	91	7	81

Table 2: Comparison of results between Compton's prudence checking system and RM. In the first column the D's and C's stand for Dazeley and Compton respectively.

In this collection of results it can be seen that Compton also tested his system using the Garvan dataset from the Garvan Institute of Medical Research in Sydney. However, RM has not been tested using this dataset due to the authors being unable to get access to it at this time. The results gathered so far do show that RM was able to correctly warn with a higher degree of accuracy than Compton's system for both the Chess and Tic-Tac-Tie datasets. However, RMs performance on these was significantly worst than its performance on the multi-classification dataset. This is believed to be primarily because the datasets are binary. Therefore, a case often tends to follow only one or two paths through the tree. Due to the algorithm trying to find patterns between the paths followed, too few paths prevent it from being able to sufficiently differentiate between cases.

Compton's Best results were with the Garvan dataset where the system was able to successfully predict a high level of warnings. While it is not the same dataset it can be seen that RM was able to get a significantly higher level of accuracy, with the multi classifiable set, than Compton's best result and with much fewer false positives. While these results are only preliminary, they do show that RM has lots of potential when used for generating warnings. With 98% accuracy and only generating warnings in 10% of cases shows that it could possibly be used for knowledge acquisition and reducing the load on the user having to review every case. It would be particularly useful after the initial core of the knowledge base has been formed and we are primarily interested in only identifying those rare cases that require knowledge from outside the core domain already known to the system.

6. CONCLUSION AND FUTURE WORK

The system described in this paper was developed to provide a means for identifying cases requiring knowledge that sits outside the current knowledge held by the knowledge based system. The system was designed to learn which patterns of paths followed were likely to be correct or incorrect. When an unusual classification pattern is located then the system provides a warning bringing the case to the expert's attention, allowing them to verify the correctness of the conclusions.

The system has undergone preliminary testing which showed that the system was able to learn quickly, and in the appropriate environment, effectively identify misclassifications. Therefore, such a system as RM potentially could be applied to large databases providing an interesting new alternative to existing methods of classification and prediction. The system needs to be tested against the Garvan dataset as well as other data mining specific datasets.

7. ACKNOWLEDGEMENTS

The authors would like to thank Mr Robert Ollington and Sung Sik Park for their helpful discussions.

REFERENCES

[1] J. L. Borges, Fictions. 1956, Paris: Gallimard.

[2] W. J. Frawley, G. Piatetsky-Shapiro and C. J. Matheus, Knowledge Discovery in Databases: An Overview, AI Magazine, (1992).13:p.57-70.

[3] J. Han and M. Kamber, Data Mining: Concepts and Techniques. 2001, San Francisco: Morgan Kaufmann Publishers. 550.

[4] B. H. Kang, P. Compton and P. Preston. Multiple Classification Ripple Down Rules: Evaluation and Possibilities. in The 9th Knowledge Acquisition for Knowledge Based Systems Workshop. 1995. Department of Computer Science, University of Calgary, Banff, Canada: SRDG Publications.

[5] R. Dazeley and B. H. Kang. Rated MCRDR: Finding non-Linear Relationships Between Classifications in MCRDR. in 3rd International Conference on Hybrid Intelligent Systems. 2003. Melbourne, Australia: IOS Press.

[6] B. H. Kang, Validating Knowledge Acquisition: Multiple Classification Ripple Down Rules. 1996, University of New South Wales: Sydney.

[7] B. H. Kang, P. Preston and P. Compton. Simulated Expert Evaluation of Multiple Classification Ripple Down Rules. in Eleventh Workshop on Knowledge Acquisition, Modeling and Management. 1998. Voyager Inn, Banff, Alberta, Canada.

[8] Y. Mansuri, J. G. Kim, P. Compton and C. Sammut. A comparison of a manual knowledge acquisition method and an inductive learning method. in Australian workshop on knowledge acquisition for knowledge based systems. 1991. Pokolbin, Australia.

[9] P. Preston, P. Compton, G. Edwards and B. H. Kang. An Implementation of Multiple Classification Ripple Down Rules. in Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop. 1996. Department of Computer Science, University of Calgary, Calgary, Canada UNSW, Banff, Canada: SRDG Publications.

[10] R. Beale and T. Jackson, Neural Computing: An Introduction. 1992, Bristol, Great Britain: IOP Publishing Ltd.

[11] C. L. Blake and C. J. Merz, UCI Repository of machine learning databases. 1998, University of California, Irvine, Dept. of Information and Computer Sciences.

[12] P. Compton, P. Preston, G. Edwards and B. H. Kang. Knowledge based systems that have some idea of their limits. in Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop. 1996.