In this paper we focus on the induction of classification rules from examples. Conventional algorithms fail in discovering effective knowledge when the database contains irrelevant information. We present a new rule extraction method, RGT, which tackles this problem by employing only relevant and irredundant attributes. Simplicity of rules is also our major concern. In order to create simple rules, we estimate the purity of patterns and propose a rule enlargement approach, which consists of rule merging and rule expanding procedures. In this paper, we describe the methodology for the RGT algorithm, discuss its properties, and compare it with conventional methods.

Keywords: classification rules, prime test, relevant attributes, rule merging, inductive learning

1. Introduction

In this paper we focus on induction of classification rules from examples. There are many reasons for work of this kind. One is the growing need to analyze examples in large repositories of customer, operations, scientific, and other sorts of data to discover useful knowledge. Another reason comes from the field of rule based expert systems, where rule induction from examples has become an important tool for knowledge engineering [1]. The role of rules in medical decision making [2] is even greater than in other fields, since they provide a very important feature that is missing in other popular machine learning methods, such as perceptron based methods, i.e., the possibility of explaining the decisions in a way that is understandable by humans.

A rule extraction algorithm should meet several important requirements to be of practical use. Extracted rules need to be simple and comprehensive; otherwise, a human will not be able to comprehend them. It is also important to discover knowledge that is accurate. Simplicity and accuracy are often contradictory requirements, and depending on the task at hand the right balance between them should be achieved. Since the datasets often contain a lot of irrelevant information, rule induction algorithms should be capable of discovering effective knowledge when irrelevant and redundant attributes are present. The scalability of rule induction algorithms, i.e., the ability to process large data sets efficiently, is another problem that should be addressed.

A variety of rule induction algorithms have appeared in the literature. The most popular algorithms are decision trees, lists, and rules [3–6], and these are known to achieve satisfactory performance on many databases. However, conventional induction algorithms fail in discovering effective knowledge when the input dataset consists of irrelevant information [7–9]. In this paper we present a new rule induction algorithm RGT (Rule Growing based on Tests). The priority in design of RGT was given to its ability to deal with irrelevant and redundant attributes. Well-known classification rule induction algorithms select significant attributes and generate rules at the same time, often using some form of information gain metric for attribute selection. In RGT, the procedure for attribute selection and rule generation are separated. The aim of the first RGT step is to select only relevant and irredundant attributes. When attributes have been selected, they are then used in the next step to generate rules. RGT selects attributes based on prime tests [10]. Tests are known under different names such as determinations or functional dependences. In [10], it was proved that prime tests represent sets of relevant irredundant attributes, and it was shown that tests can significantly reduce the training dataset and improve the recognition rates of inductive classifiers.

A dataset can have many prime tests. Therefore some criteria are needed for their selection. A number of selection methods for tests were presented in [2, 10]. In this paper we adopt a different approach and focus on the selection of a prime test that can lead to the generation of simpler rules. We define the length of a test as the number of attributes in the test. As in [10], we prefer short prime tests. The shorter the prime test, the fewer attributes are needed to describe the concept. However, there could be many tests even with the minimal length. Therefore, we
need to introduce a measure of simplicity that we apply to short tests. Noting that the sequence (or pattern) of attribute values for an example as determined by a test can be different, we estimate the simplicity of a test in terms of the simplicity of the example patterns it generates using entropy. Entropy is commonly used in information theory to characterize the purity of an arbitrary collection of examples. We define the best test as the one for which examples have patterns with the lowest entropy. By selecting the best test we select attributes for which examples have the most uniform patterns. This property has two main consequences. The first is the simplicity of rules. In general, the more uniform the patterns, the fewer rules we need to cover them. We say that a rule “covers” an example if it correctly classifies the example. The second is the tolerance to noise. The more uniform the patterns, the fewer outliers appear in the selected part of database. From our point of view, the ability of an algorithm to select relevant and redundant attributes with uniform patterns is the most critical factor for inducing simple rules with high performance.

After selecting the best test we reduce the database by eliminating all attributes not in the test. Rule generation is then performed on the reduced database. For each class RGT generates a set of rules that together cover examples of the class. Most rule learning algorithms, including CN2 [5] and AQ [6], use a general-to-specific covering approach. This approach starts from a maximally general rule and moves toward specialized rules, i.e., rules become more precise when they fail to cover examples. The approach is more suitable for outputting general rules, rather than high accuracy rules. Since one of our purposes is the high accuracy of rules, we take a different approach. We choose a specific-to-general covering approach. First, one rule for each example is constructed, i.e., specific rules are formed. From these specific rules we then generate general rules. This is done by the proposed rule enlargement approach, which consists of rule merging and rule expansion procedures. In general, rules represent hyperboxes in the attribute space. Adjacent hyperboxes that have identical consequents can be merged. For this purpose we developed a procedure that is similar to the Quine-McCluskey minimization method of Boolean functions [11]. This procedure finds so called maximal hyperboxes (MH) (a hyperbox that is not a part of any other hyperbox) and then selects a minimal set of maximal hyperboxes covering all examples. Rule merging has also been studied in [12], where a greedy search method for merging hyperboxes has been proposed. This method has bias toward promising joins, i.e., it prefers to combine many rules than a few rules. The method proposed in [12], is not systematic and does not guarantee to find all maximal hyperboxes. Our proposed rule merging method finds all maximal hyperboxes and then searches for an efficient minimal covering of examples.

The problem of finding maximal hyperboxes, without using rule merging procedures has been investigated by several researchers. In [13, 14] exhaustive search algorithms based on a combinational test of the training examples were proposed. In [15, 16] heuristics were introduced to reduce computational time of exhaustive algorithms, such as pre- and post-processing, and a randomized algorithm. In our case, the search for MH is carried out by the adapted Q-M method. There are several reasons for its use. It is deterministic, it does not degrade the accuracy of rules, it finds all MHs, and as we show it achieves this in a polynomial time. The proposed Q-M method is also a logical extension to the test approach. A test generates rules, which represent hyperboxes. In order to simplify rules we need to merge them. That is why the Q-M method is introduced. First it merges all adjacent hyperboxes and then identifies the minimal cover of the examples.

The merged rules have some useful properties, such as, consistency, and efficiency in covering all examples. However, they can be still complex, especially for domains containing scarce data. In order to simplify the merged rules, we expand them. The idea of rule expansion is not new. It was used in [17] for simplification of fuzzy classification rules. Here, rule expansion was conducted in combination with rule pruning. Our rule expansion approach is similar to [17], however, to keep high accuracy of rules we separate rule expansion from pruning. Our rule expansion is performed without sacrificing the accuracy. We expand rules by allowing them to cover more space until rules from different classes collide. Expanded rules are checked for inclusion, and rules that are covered by other rules are eliminated. After performing rule expansion the obtained rules are pruned. By separating rule expansion and pruning we allow the user to influence the learning process. If the accuracy of rules is most important, learning can be performed without pruning. If simplicity of rules is the main concern the pruning procedure is executed.

RGT outputs propositional rules. These rules are easily understandable, since the length and the number of rules is intended to be as small as possible, and all rules in the knowledge base refer to the same attributes and can be graphically displayed as a table with one column for each attribute and one row for each combination of attribute values.

To test the performance of the proposed method, we apply it to artificial and real data. The accuracy and simplicity of rules are measures used for comparison. Empirical results show the effectiveness of the proposed method in detecting classification rules, leading to improved performance over conventional algorithms.

2. RGT

We consider the problem of automatically inferring a set of classification rules from a database $D$. In the standard formulation, the database is known as the training set and each example in the database is known as an instance. Each instance is described by the values of a collection of its attributes (the same attributes are used for each instance), and each instance belongs to one of a set
of mutually exclusive classes. The aim is to develop classification rules, which for every instance predicts the class from the values of its attributes. We identify each instance with a vector of attributes $\bar{x} = (x_1, \ldots, x_n)$. For a set of attributes $\tau$, let us denote $D_\tau$ to be $\bar{x}$ with all the attributes not in $\tau$ eliminated. Similarly, let $D_\tau[x_i]$ be the data set $D$ with each $\bar{x} \in D$ replaced by $\bar{x}[x_i]$.

The basic form of many rule induction algorithms, including decision trees and CN2, can handle only discrete data. These algorithms were extended to handle continuous data by partitioning the continuous attribute values into a discrete set of intervals, i.e., by discretizing them. Since RGT relies on tests and entropy, which are essentially of a discrete nature, the basic form of RGT can handle only discrete data. RGT can be extended to handle continuous attributes. For this purpose we need to introduce new concepts for tests and entropy on continuous attributes. Although we describe below how this can be done, our main focus in this paper will be on RGT with discrete data.

The basic definition of a test [10] was introduced for discrete attributes. A test is a collection of features that is sufficient to distinguish instances from different classes. An irreducible set of attributes that forms a test is called a prime test. Defining a test requires an inequality measure. For discrete attributes (including nominal and integer-valued) an ordinary inequality $\neq$ is used. There are two ways to handle continuous attributes. One way is to discretize them. Another way is to use an inequality $R$, where $aRb$ iff $|a - b| > r$. The threshold $r > 0$ defines the ordinary inequality, and $r > 0$ is used to introduce stronger relations. Using an ordinary inequality $\neq$ on continuous attributes is not desirable since all values are likely to be different. For integer-valued attributes $r > 0$ can also be used. Below we assume that for each attribute $x_i$ the threshold $r_i \geq 0$ is defined.

Prime tests are very important sets since they define relevant, irredundant attributes [10]. Prime tests allow us to filter a database for irrelevant data and reduce the database to a smaller size. Smaller databases, in turn, will generate shorter rules. However, the simplicity of rules depends not only on the length of the rules but also on their number. In order to generate a small number of rules we select a short prime test $\tau$ which has a minimal encoding length, i.e., which maximizes the purity of $D_\tau$. We use entropy to evaluate the encoding length. For a test $\tau$, we calculate the entropy as

$$\text{Entropy}(\tau) = - \sum_{d \in D_\tau} P(d) \log_2 P(d),$$

where $P(d)$ is the proportion of the pattern $d$ in $D_\tau$. We define the best test as the one with lowest entropy. By selecting a prime test with minimal encoding length, we are selecting attributes where examples have the most uniform patterns. The uniformity of $D_\tau$ affects the number of rules. Different instances from $D$ can have the same pattern on $\tau$, therefore a rule covering one pattern can cover several instances from $D$. $\tau$ with fewer patterns will generate fewer rules. To make this statement more clear, let us consider an example database $D$ in Fig. 1. $\tau = \{x_1, x_2\}$ is a prime test for this database. $D_\tau$ contains 3 patterns $\{11, 10, 01\}$. A set of rules that covers only these 3 patterns will cover all 8 instances from the database $D$.

The basic definition of entropy was introduced for discrete attributes. Although this definition can be extended to continuous attributes, it requires knowledge of a probability density function. Since it is quite difficult to give a reasonable estimation of the probability density function for continuous attributes based on only training examples, we introduce a more practical approach based on the connected component approach. We consider patterns as nodes of a graph, and connect two nodes $\bar{x}$, $\bar{y}$ iff $|\bar{x} - \bar{y}| \leq r$. From the definition of a test, it follows that all nodes of a connected component come from the same class. The number of nodes can be used to determine the probability of occurrence of a connected component. In the continuous case, instead of calculating the purity of a collection of patterns we calculate the purity of a collection of connected components. However, we note that for the continuous case the approach is more complex and requires further consideration. In this paper, we only focus on discrete attributes, and suppose that continuous attributes are discretized beforehand.

The proposed RGT first extracts a set of minimal prime tests and then selects the prime test with the minimal encoding length. For test extraction, an asymptotically optimal test detection algorithm, PTD3.1, can be used. This algorithm is developed by one of the authors, and is available at:


After selecting the best test and reducing the original database to $D_\tau$, RGT starts to generate rules. RGT uses a specific-to-general covering approach. The purpose of the rule generation procedure is to create accurate rules with maximum covering volume.

RGT takes each class and seeks a way to cover all instances in it. First, RGT constructs one rule for each pattern $i.e.,$ constructs maximally specific rules on $D_\tau$. For each $(a_1, \ldots, a_m)$ from $D_\tau$ the following rule is generated.

$$a_1 - r_1 \leq x_1 \leq a_1 + r_1, \ldots, a_m - r_m \leq x_m \leq a_m + r_m(1),$$

where $\tau = (x_1, \ldots, x_m)$ is the selected test. To be more precise, we should note that rules defined by (1) already

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 1. A sample database.
have a degree of generalization, when they are considered on the original database \( D \). They do not contain all attributes, and involve inequalities, hence they are different from the maximally specific rules defined on the set \( D \).

After generating (1), RGT seeks to merge rules in order to obtain broader rules and decrease their number. In general, a rule (1) represents a hyperbox in the attribute space. We call two hyperboxes adjacent iff their boundaries coincide on all attributes except one and boundaries that don’t coincide are overlapped. Adjacent hyperboxes can be merged (Fig. 2). For this purpose we employ an easily adapted version of the Quine-McCluskey (Q-M) minimization method of Boolean functions \([11, 18]\). The adapted method consists of the following steps, below we give only a brief description of the adapted algorithm.

First, we define terms that provide the basis of the adapted Q-M method. A minterm is a hyperbox defined by (1). A maximal hyperbox (MH) is a hyperbox that is not a part of any other hyperbox. An essential maximal hyperbox (EMH) is a maximal hyperbox that covers at least one minterm that is not covered by any other maximal hyperbox. The terms maximal hyperbox and essential maximal hyperbox have corresponding names in the original Q-M method, which are prime implicant and essential prime implicant, respectively. The goal of the adapted Q-M method is to find a minimal set of MHs that covers all minterms of the class.

The adapted Q-M method consists of the following steps.

**Part I**

- **Step 1.** List in a column all the minterms.
- **Step 2.** Eliminate a hyperbox in the column if it is covered by another hyperbox. Perform an exhaustive search for adjacent hyperboxes. If two hyperboxes are adjacent then combine them into a new column of hyperboxes checking off each hyperbox that is merged. Repeat step 2 on the new column of hyperboxes until no further hyperboxes can be merged. Any hyperbox not checked off represents MH.

**Part II**

- **Step 3.** Construct a maximal hyperbox chart that lists minterms along the horizontal and MHs along the vertical, with a \( \bullet \) entry placed wherever a certain MH covers a given minterm.
- **Step 4.** Identify all EMH for cover on the chart. Remove rows corresponding to the identified EMH. Remove columns corresponding to minterms covered by the removed rows. Remove any MHs that are redundant; that is, they do not cover any remaining minterms.
- **Step 5.** Reduce the chart by eliminating undesirable MH, i.e., MH that covers the same uncovered minterms as other MH and has a smaller volume. Eliminate columns that cover other columns. The reduced chart contains only the minterms that remain to be covered and remaining MH candidates for inclusion in the cover.
- **Step 6.** If a cycle chart \([18]\) results after completing step 5, go to step 7. Otherwise, return to step 4 to find MH which became essential on the reduced chart.
- **Step 7.** Apply the cyclic chart procedure. Repeat step 7 until a void chart occurs or until a noncyclic chart is produced. In the latter case return to step 4.

The cyclic chart procedure in step 7 starts when no MH is found to be essential in the chart. This procedure relies on heuristics to identify additional MH to add to EMH for the cover. Therefore, when a cyclic chart is encountered, Q-M does not guarantee to identify a minimum set of MH to complete the cover after the EMHs have been found. However, we should note that the problem of finding a minimum set of MH is a minimum set cover problem which is NP-hard.

Next we give a rough estimate of the running time of the MH search (step 2), which is the most time consuming part in RGT. If an attribute \( x_i \) takes \( d \) values then there are \( d(d+1)/2 \) possible \( a_i \leq x_i \leq b_i \) intervals, i.e., this number is bounded by \( O(d^2) \). The number of possible hyperboxes that can be generated by the algorithm is bounded by \( O(d^{2m}) \), where \( m \) is the length of the selected test, and \( d \) is a maximal number of values taken by attributes. The exhaustive search for adjacent pairs of hyperboxes requires \( 2md^2m \) time, i.e., \( O(md^{4m}) \) time. The same time takes to test whether a hyperbox is covered by another hyperbox. So the running time of the algorithm is bounded by \( O(md^{4m}) \). Denote \( p \) as the number of minterms. By assuming \( d^m \sim p \), we estimate \( d \) by \( p^{1/m} \), and substituting this into the above we get the running time of RGT \( O(p^{m/p^4}) \), where \( p \) is less than the number of examples, and \( m \) is less than the number of attributes in the database.
Until this point we concentrated on one class at a time, disregarding what happens to other classes. For each class, RGT constructs a minimal set of rules that covers all patterns in the class. The constructed rules are limited to the set \( D_1 \) in the sense that their covering range is the same as rules (1), which are defined for each pattern of \( D_1 \). In other words any instance \( \tilde{x} \not\in D \), which does not satisfy (1), will not be covered by the merged rules. To increase the generalization ability of the rules, RGT expands them. Each rule

\[
\alpha_1 \leq x_1 \leq \alpha_2, \ldots, \alpha_m \leq x_m \leq \alpha_m,
\]

is expanded by increasing the covering range of the \( x_i (i = 1, \ldots, m) \) attribute by the value \( tr_i \) (for \( tr_i = 0 \), we assume \( tr_i = 1 \)), i.e., by considering

\[
\alpha_i - tr_i \leq x_i \leq \alpha_i + tr_i,
\]

iff the latter does not overlap with the rules from other classes. This procedure is performed for each rule one by one, and then is repeated until rules from different classes run into each other. The idea is to include as many neighbor points of each rule as possible based on the principle of nearest neighbor. At the end, for each class, expanded rules are checked for inclusion. If a rule is covered by another rule, it is eliminated. At this point the rule enlargement process is terminated. We expect that the resulting rules cover almost all the domain. However, there may appear instances from the test set that are not covered by any rule. We consider such instances as a part of the nearest biggest hyperbox.

Rules constructed by RGT are consistent with the set of training examples. Although this is sometimes a reasonable strategy, it can lead to difficulties when there is noise in the data. Although, RGT has capability to tolerate noise by selecting a test with the most uniform patterns, it still can produce rules that overfit the training data. There are several approaches to avoid overfitting [19]. Since, in this paper, the overfitting is not our main concern, we use one of the simplest approaches for rule pruning. We discard a rule generated by RGT, if it covers only a few patterns. We denote by RGT-n the set of rules output by RGT, where rules covering less than \( n \) patterns are discarded.

3. Experimental Results

We use three induction algorithms as a basis for comparisons. These are ID3 [20], C4.5 [3], and CN2 [5]. ID3 and C4.5 generate classification rules by first constructing a decision tree and then converting the learned tree into an equivalent set of rules. The decision tree is converted into an equivalent set of rules by creating one rule for each path from the root node to a leaf node. CN2 is a general-to-specific covering algorithm that induces rules directly. Decision trees and CN2 are well known in the machine learning community and represent completely different approaches to rule learning, hence we hope that our results are of a general nature.

In our experiments, we use the monk1, promoter gene sequences (promoters), breast cancer, and pima-indians (pima) databases from the Machine Learning Database Repository at the University of California. With the exception of pima all these databases are discrete. In this paper, since we are considering only discrete databases, the pima database was also discretized in preprocessing using the discretize filter from [21] based on the MDL method.

We randomly selected 70% of each database as a training set and the remaining 30% as a test set, and 10 such trials were carried out. For each trial, we calculate the rule accuracy for the training and test sets and the rule simplicity on the training set. We estimate the simplicity of rules by

\[
\frac{\sum_i n_i}{n \cdot |D|} \cdot 100
\]

where, \( n_i \) is the number of attributes used in the \( i \)th rule, and \( n \) and \( |D| \) are the number of attributes and the number of instances in the training set respectively.

Table 1 shows the accuracy and simplicity estimations of RGT, RGT-1, CN2, C4.5 and ID3 on the training set of monk1, promoters, breast cancer, and pima databases. The reported accuracy and rule simplicity are the mean of the ten trials. For accuracies we also show the standard deviation from the mean. As can be seen from Table 1, CN2 generates the smallest set of rules in many cases, but results in worse accuracies. The simplification of rules is achieved by significantly sacrificing recognition accuracy. In contrast, RGTs show a good balance between accuracy and simplicity. The accuracies are high, while generated rules are reduced reasonably. It is interesting to compare RGT to ID3, and RGT-1 to C4.5, since these rule induction algorithms have some common features. RGT and ID3 are exact learners, while RGT-1 and C4.5 are their noise tolerant versions. As shown in Table 1, RGT and ID3 achieve a 100% recognition rate. However, RGT shows a better ability to generate simple rules than ID3. RGT-1 also shows better performance than C4.5 in many cases, despite the simple overfitting approach that was adopted.

Table 2 shows the recognition results on the testing set of monk1, promoters, breast cancer, and pima databases. As we can see, RGTs perform significantly better than conventional algorithms, demonstrating that the proposed method is an effective classification rule generation algorithm. We also note, that RGT-1 failed to improve the recognition rate of RGT on all databases. The proposed rule overfitting approach resulted in removing non-noisy rules. One possible reason for such performance is the RGT tolerance to noise. There is a difference in noise presence in the databases \( D \) and \( \frac{|D|}{\tau} \). \( \frac{|D|}{\tau} \) is a part of \( D \), with the most uniform patterns. It contains less spurious patterns than other parts of \( D \). Furthermore a rule covering one pattern from \( \frac{|D|}{\tau} \), can cover several instances from \( D \), and therefore its removal may not be appropriate.
4. Conclusions

Conventional rule induction algorithms perform poorly when the database consists of irrelevant information. In this paper we present an algorithm RGT, which tackles this problem by employing a prime test that is a set of relevant irredundant attributes. The selection of attributes by a test results in the generation of rules with high accuracies. The simplicity of the rules is also a major concern of RGT. RGT uses an encoding length measure and rule enlargement techniques to generate rules with a high coverage range.

We applied our algorithm to real and artificial databases. Experimental results revealed the effectiveness of the proposed method in detecting classification rules, leading to improved performance over conventional algorithms. We showed that RGT meets many essential requirements to be of practical use, including the ability to handle irrelevant attributes, a high recognition ability and the ability to generate sets of simple rules.

RGT has several limitations. The major limitation of RGT is its scalability. Decision tree and CN2 induction algorithms scale well to large datasets due to their divide-and-conquer and general-to-specific approach. RGT uses a test detection algorithm for selecting relevant, irredundant attributes. The disadvantage in selecting attributes by tests is its time requirement. The proposed rule merging approach has a poor scaling ability. When the number of examples increases in exponential order it becomes unpractical as with the original Q-M method. To deal with this problem, when the dimension of the database becomes very large, heuristic test and MH detection algorithms can be used. One useful feature of the proposed method is that all of its procedures are independent allowing for their easy replacement or deletion.

RGT has another limitation in that it can become ineffective for domains containing scarce but overlapped data. Because rule merging and rule expansion procedures on which RGT is based, may not work well on such domains, RGT is likely to generate complicated rules. Although RGT can still retain a high recognition performance on such domains, the simplification of rules by the proposed method is problematic.

There are many directions for future work. A better approach to overfitting should be developed. A rule post-pruning approach evaluated on a spatial validation set could be a good candidate for consideration. Another direction is to investigate the effectiveness of tests for conventional rule induction algorithms. These algorithms rely on their own feature selection strategy. They can be modified to incorporate tests, and ways of doing this should be investigated.

Acknowledgements

The authors are grateful to the Chukyo University Research Fund for financial assistance with this research. The authors would also like to thank graduate school students A. Kozuki and S. Hisamoto for help with conducting experiments.

References:


Table 1. Accuracy and simplicity estimations on the training set of monk1, promoters, breast cancer, and pima databases.

<table>
<thead>
<tr>
<th></th>
<th>Monk1</th>
<th>Promoters</th>
<th>Breast cancer</th>
<th>Pima</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>simplicity</td>
<td>accuracy</td>
<td>simplicity</td>
<td>accuracy</td>
</tr>
<tr>
<td>RGT</td>
<td>1.1%</td>
<td>100±0.0%</td>
<td>0.8%</td>
<td>100±0.0%</td>
</tr>
<tr>
<td>RGT-1</td>
<td>1.1%</td>
<td>100±0.0%</td>
<td>0.6%</td>
<td>94.1±3.2%</td>
</tr>
<tr>
<td>CN2</td>
<td>0.9%</td>
<td>79.0±3.3%</td>
<td>0.5%</td>
<td>91.6±3.2%</td>
</tr>
<tr>
<td>C4.5</td>
<td>2.1%</td>
<td>75.7±2.1%</td>
<td>0.6%</td>
<td>97.1±2.5%</td>
</tr>
<tr>
<td>ID3</td>
<td>27.5%</td>
<td>100±0.0%</td>
<td>1.9%</td>
<td>100±0.0%</td>
</tr>
</tbody>
</table>

Table 2. Recognition results on the testing set of monk1, promoters, breast cancer, and pima databases.

<table>
<thead>
<tr>
<th></th>
<th>Monk1</th>
<th>Promoters</th>
<th>Breast cancer</th>
<th>Pima</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>simplicity</td>
<td>accuracy</td>
<td>simplicity</td>
<td>accuracy</td>
</tr>
<tr>
<td>RGT</td>
<td>100±0.0%</td>
<td>85.0±4.7%</td>
<td>95.7±2.0%</td>
<td>83.5±2.9%</td>
</tr>
<tr>
<td>RGT-1</td>
<td>100±0.0%</td>
<td>83.8±7.6%</td>
<td>95.5±1.6%</td>
<td>78.4±5.5%</td>
</tr>
<tr>
<td>CN2</td>
<td>75.0±3.4%</td>
<td>41.6±6.2%</td>
<td>91.2±6.8%</td>
<td>57.1±5.5%</td>
</tr>
<tr>
<td>C4.5</td>
<td>75.7±2.5%</td>
<td>75.0±4.0%</td>
<td>92.9±3.9%</td>
<td>66.6±2.2%</td>
</tr>
<tr>
<td>ID3</td>
<td>78.4±2.8%</td>
<td>66.6±4.3%</td>
<td>90.7±5.2%</td>
<td>62.3±3.1%</td>
</tr>
</tbody>
</table>


Name: George Lashkia
Affiliation: Professor, School of Information Science and Technology, Chukyo University
Address: 101 Tokodate, Kaizu-cho, Toyota 470-0393, Japan
Brief Biographical History: 1996- Associate Professor, Okayama University of Science 2003- Professor, Chukyo University
Main Works:
Membership in Academic Societies:
- The Institute of Electrical and Electronics Engineers (IEEE)
- Association for Computing Machinery (ACM)

Name: Laurence Anthony
Affiliation: Associate Professor, School of Science and Engineering, Waseda University
Address: 3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan
Brief Biographical History: 1993- Lecturer, Okayama University of Science 2004- Associate Professor, Waseda University
Main Works:
Membership in Academic Societies:
- The Institute of Electrical and Electronics Engineers (IEEE)

Name: Hiroyasu Koshimizu
Affiliation: Professor, School of Information Science and Technology, Chukyo University
Address: 101 Tokodate, Kaizu-cho, Toyota 470-0393, Japan
Brief Biographical History: 1975- Graduated from Graduate School of Nagoya University (DR) 1986- Professor of Chukyo University 2004- Dean of School of Information Science and Technology, Chukyo University
Main Works:
Membership in Academic Societies:
- The Institute of Electrical Engineers of Japan (IEEJ), senior member
- The Institute of Electronics, Information, and Communication Engineers of Japan (IEICE)