Using Rough Sets and Genetic Algorithm to Build Decision Tree

Mostafa Fahmy, Osama Badawy [a], Yasser Foaud [b], Wedad Sagar [b]

[a] *College of Computing and Information Technology, Arab Academy for Science &Technology & Maritime Transport, Alexcandira, Egypt*
[b] *Department of Mathematics (Computer, Science), Alexcandira University , Egypt*

Abstract: Rough set theory, which emerged about 20 years ago, is nowadays a rapidly developing branch of artificial intelligence and soft computing. In this paper, we present a new approach (RS_DT) for construction of decision tree based on rough set theory, which will be induced a simpilified tree. This tree is transformed into an initial population of genetic probabilistic rough induction (GA+GDT-RS), which is a combination between probabilistic rough induction and genetic algorithm. We will use probabilistic rough induction for modeling a classification system and applying genetic operators to a population of chromsomes. However, it is interesting to try to incorporate these approaches into the hybrid system. The challenge is to get as much as possible from this association

1. Introduction

Historically different approaches for knowledge extraction evolved [7], such as symbolic approaches and computational learning theory. Among them we can find many classical approaches, like decision trees, rough sets, case based reasoning, neural networks, support vector machines, different fuzzy methodologies, ensemble methods [4], but they all have some advantages and limitation. Evolutionary approaches (EA) are also a good alternative, because they are not inherently limited to local solutions [5]. Recently, taking into account the limitations of classical approaches many researches focused their research on hybrid approaches.

Current studies show that the selection of appropriate method for data analysis can be crucial for the success. Therefore, for a given problem, different methods should be tried to increase the quality of extracted knowledge.

According to the previous paragraph a logical step would also be to combine different methods into one more complex methodology in order to overcome the limitation of a single method.

The paper is organized as follows: section 2 introduces a basic model, section 3 shows the new approach (RS_DT) for construction of decision tree based on rough set theory, section 4 discusses our hybrid method in detail.

Applications and results in section 5, section 6 concludes our paper.

2. Basic Models

2.1 Rough Set Methodology

In the rough set methodology for rule discovery, data base is regarded as a decision table, which is denoted $T = (U, A, \{Va\}_{a \in A}, f, C, D)$, where U is a finite set of instances (or objects), called the universe, A is a finite set of attributes, each V_a is the set of values of attribute a, f is a mapping from $U \times A$ to $V(= \bigcup_{a \in A} Va)$, C and D are two subsets of A, called the sets of attributes and decision attributes, respectively, such that $C \cup D = A$, and $C \cap D = \phi$. Equivalence classes in U/C and U/D are called condition classes and decision classes, respectively [??-3], [??-4], [??-5], [??-6].

The process of rule discovery is that of simplifying a decision table and generating minimal decision algorithm. In general, an approach for decision table simplification consists of the following steps:

(1) Elimination of duplicate condition attributes. It is equivalent to elimination of some columns from the decision table.

(2) Elimination of duplicate rows.

(3) Elimination of superfluous values of attributes.

A representative approach for the computation of reducts of condition attributes is to represent knowledge in the form of a discernibility matrix [??-5], [??-6]. The basic idea can be briefly presented as follows:

Let $T = (U, A, \{Va\}_{a \in A}, f, C, D)$, be a decision table with U={$u_1, u_2 \ldots, u_n$}. By a discernibility matrix of T, denoted M(T), we will mean nxn matrix defined as:

$$mij = \left\{ \begin{array}{ll} \{c \in C : c(ui) \neq c(uj)\ \text{if}\ \exists d \in D\,[d(ui) \neq d(uj)] \\ \lambda & \text{if}\ \forall d \in D\,[d(ui) = d(uj)] \end{array} \right\}$$

for i,j=1,2,…,n such that u_i or u_j belongs to the C-Positive region of D.

Thus entry mij is the set of all the condition attributes that classify objects u_i and u_j into different decision classes in U/D. Since M(T) is symmetric and $mii = \phi$, M(T) are represented only by elements in the lower triangle, that is, the mij with $1 \leq j < i \leq n$. Furthermore, $mij = \lambda$ denotes that this case does not need to be considered. Hence it is interpreted as logic truth.

The discernibility function f_T for T is defined as follows :

for any $ui \in U$

$$f_T(ui) = \underset{j}{\wedge} \{\vee mij : j \neq i, j \in \{1,2,....,n\}\}$$

where

i) $\vee mij$ is the disjunction of all variables such that $c \in mij \; if \; mij \neq \phi$

ii) $\vee mij = \perp \; (false), if \; mij = \phi$

iii) $\vee mij = T \; (true), if \; mij = \lambda$

Each logical product in the minimal disjunctive normal form (DNF) of f_T (ui) is called a reduct of instance ui.

Generating minimal decision algorithm is to eliminate the superfluous decision rules associated with the some decision class . It is obvious that some decision rules can be dropped without disturbing the decision-making process, since some other rules can take over the job of the eliminated rules.

2.2 GDT-RS Principles

GDT-RS is a soft hybrid induction system for discovering classification rules from databases with uncertain and incomplete data [Zhong etal., 1998). The system is based on hybridization of GDT and the rough set methology.

2.2.1 GDT and Rule Strength

Any GDT consists of three components: possible instances, possible generalizations of instances, and probabilistic relationships between possible instances and possible generations. Here the possible instances are all possible combinations of attribute values in a database; the possible generations for instances are all possible cases of generation for all possible instances; the probabilistic relationships between possible instances and possible generations, represented by entries Gij of a given GDT, are defined by means of a probabilistic distribution describing the strength of the relationship between any possible instance and any possible generation. The prior distribution is assumed to be uniform, if backgound knowledge is not available. Thus, it is defined by Eq. (1)

$$Gij = p(PIj|PGi) = \begin{cases} 1/N_{PGi} & \text{if } PIj \supseteq PGi \\ 0 & \text{otherwise} \end{cases} \quad ---(1)$$

where PIj is the jth possible instance , PGi is the ith possible generalization . and N_{PGi} is the number of the possible instances (PI) satisfying the ith possible generation, i.e.,

$$N_{PGi} = \prod_{j}^{m} n_j \qquad ----(2)$$

where j=1,..m and j is not equal the attribute that is contained by the ith possible generalization.

2.2.2 Rule Strength

The rules are exprssed in the following from:

$P \rightarrow Q$ with S

i.e., "if P then Q with the strength S" where P denotes a conjunction of conditions (i.e.,$P \subseteq C$), Q denotes a concept that the rule describes (i.e., $Q \subseteq D$), S is a "measure of strength" of the rule. Furthermore, S consists of three parts: s(P), accuracy and coverage, where s(P) is the strength of the generalization P (i.e., the condition of the rule), the accuracy of the rule is measured by a noise rate function: $r(P \rightarrow Q)$, coverage denotes how many instances are covered by the rule. If some instances covered by the rule also belong to another class, the coverage is a set:{ number of instances belonging to the class, number of instances belonging to another class}.

The strength of a given rule reflects the incompleteness and uncertainly in the process of rule inducing influenced both by unseen instances and noise. The strength of the generalization P=PG is given by Eq. (3) under that assumption that the prior distribution is uniform

$$s(P)=\sum_{j} p(PI_j/P)=card([P]) \times 1/N_p \quad \text{--- (3)}$$

Where card([P]) is the number of observed instances satisfying the generation P. The strength of the generalization P represent explicitly the prediction for unseen instances since possible instances are considered. On other hand, the noise rate is given by Eq. (4)

$$r(P \rightarrow Q)=card([P] \cap [Q])/card([P]) \quad (4)$$

where card([Q]) is the number of all instances from class Q within the instances satisfying the generalization P.

2.2.3 Searching Algorithm for an Optimal Set of Rules

In [Zhong, 2001] they outline the idea of searching algorithm for a set f rules. They use a sample decision table shown in Table 1 to illustrate the idea. Let Tnoise be a threshold value.

Table 1. A sample database

o	a	b	c	d
u_1	a_o	b_o	c_1	y
u_2	a_o	b_1	c_1	y
u_3	a_o	b_o	c_1	y
u_4	a_1	b_1	c_0	n
u_5	a_o	b_o	c_1	n
u_6	a_o	b_2	c_1	n
u_7	a_1	b_1	c_1	y

Step 1. Create GDT, if prior background knowledge is not available, the prior distribution of a generalization is calculated using Eqs.(1) and (2).

Step 2. Consider the indiscernibility classes with respect to the condition attribute set C (such as u1,u3 and u5 in the sample database of Table 1) as one instance, called the compound instance (such as u1'=[u1]IND(a,b,c) in the following Table 2).

Then the probabilities of generalizations can be calculated correctly.

Table 2: Table 1 after makes compound instances

U \ A	a	b	c	d
$u_1'(u_1,u_3,u_5)$	a_o	b_o	c_1	y,y,n
u_2	a_o	b_1	c_1	y
u_4	a_1	b_1	c_o	n
u_6	a_o	b_2	c_1	n
u_7	a_1	b_1	c_1	y

Step 3. For any compound instance u' (such as the instance u_1' in the above table), let d(u') be the set of the decision classes to which the instances in u' belong. Furthermore, Let $Xv=\{x\in U:d(x)=v\}$ be the decision class corresponding to the decision value v. The rate r_v can be calculated by Eq(4). If there exists a $v\in d(u')$ such that $r_v(u')=\min\{ r_{v'}(u') \mid v'\in d(u')\} <$ Tnoise, then they let the compound instance u' point to the decision class corresponding to v. If there is no $v \in d(u')$ such that $r_v(u')<$ Tnoise, they treat the compound instance u' as a contradictory one, and set the decision class of u' to $\perp$ (uncertain).

For example, we have

U \ A	a	b	c	d
$u_1'(u_1, u_3,u_5)$	a_o	b_o	c_1	$\perp$

Let U' be the set of all the instances except the contradictory ones.

Step 4. Select one instance u from U'. Using the idea of discernibility matrix, create a discernibility vector (i.e. the row or the column with respect to u in the discernibility matrix) for u. For example, the discernibility vector for instance $u_2:a_0b_1c_1$ is as follows:

U \ U	$u_1'(\perp)$	u_2(y)	u_4(n)	u_6(n)	u_7(y)
u_2(y)	b	Ø	a,c	b	Ø

Step 5. Compute all the so-called local relative reducts for instance u by using the discernibility function. For example, from instance $u_2:a_0b_1c_1$, they obtain two reducts: {a,b} and {b,c}.

Step 6. Construct rules from the local reducts for instance u, and revise the strength of each rule using (3). For example, the following rules are acquired:

$\{a_0\, b_1\} \longrightarrow y$ with $S=1 \times ½ = 0.5$,

and

$\{b_1\, c_1\} \longrightarrow y$ with $S=2 \times ½ =1$

for instance $u_2 :a_0b_1c_1$

Step 7. Select the best rules from the rules (for u) obtained in step 6 according to its priority :

- Selecting the rules that contain as many instances as possible.
- selecting the rules that contain as little attributes as possible, if they cover the same number of instances
- Selecting the rules with larger strength, if they have same number

of condition attributes and cover the same number of instances.

For example, the rule '{b_1c_1}$\longrightarrow$y' is selected for the instance u_2:$a_0b_1c_1$ because it matches more instances than the rule '{a_0b_1}$\longrightarrow$y' .

Step 8. U'=U'-{u}. If U $\neq$ Ø , then go back to step 4. Otherwise, goto setp 9.

Step 9. Finish if the number of rules selected in step 7 for each instance is 1. Otherwise find a minimal set of rules, which contains all of the instances in the decision table.

The following Table 3, shows the result for the sample database shown in Table 1.

Table 3: Results for a sample database

U	Rules	Strength
u_2,u_7	$b_1 \wedge c_1 \rightarrow y$	1
u_4	$c_0 \rightarrow n$	0.167
u_6	$b_2 \rightarrow n$	0.25

2.3 Genetic Algorithm

The origin of Genetic Algorithms (GAs) is attributed to Holland's [2] works on cellular automata. There has been significant interest in GA over last two decades. The range of applications of GA includes such diverse areas as job shop scheduling, training neural nets, image feature extraction, and image feature identification [1].

A genetic algorithm is a search process that follows the principles of evolution through natural selection [1]. The domain knowledge is represented using a candidate solution called an organism or chromsome. Typically, an organism is a single genome represented as a vector of length n: $c = (c_i | 1 \le i \le n)$, where c_i is called a gene.

An abstract view of a generational GA is given in figure 1. A group of organisms is called a population. Successive populations are called generations. A generational GA starts from initial generation G(0) , and for each generation G(t) generates a new generation G(t+1) using genetic operators such as mutation and crossover. The mutation operator creates new genomes by changing values of one or more genes at random as shown in figure 2. The crossover operator joins segments of two or more genomes to generate a new genome. Figure 3 depicts an example of a crossover operation.

```
Genetic Algorithm :
generate initial population   G(0);
evaluate G(0);
for(t=1;solution is not found;t++)
{ generate G(t) using G(t-1);
evaluate G(t); }
```

Figure 1. Abstract view of a enerational genetic algorithm

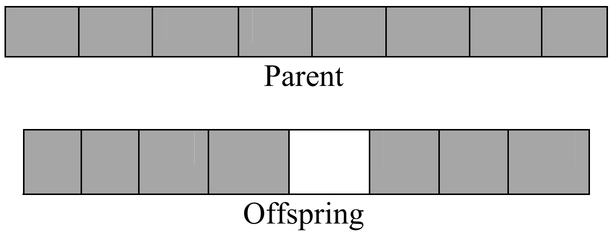

Figure 2. Mutation operation

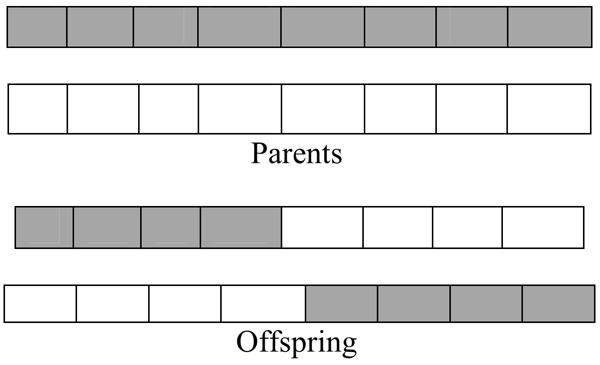

Figure 3. Crossover operation

3. Proposed Method of Decision Tree (RS_DT)

In many literatures [?->1], [??->3], rough set has been used for selecting attributes, consequently a reduct of attributes will be found which is regarded as the best reduction of attributes. The goal is to reduce the volume of data. In this paper, we introduced a new approach of decision tree construction based on rough sets.

The basic algorithm for construction of decision tree is as follows. The input of the algorithm for construction of decision tree based on discernibility vector is a collection of data. The output is a decision tree corresponding to the data.

The algorithm RS_DT is presented in receursive form.

Algorithm

Input

U= a data collection.

A = condition attributes.

Output: decision tree.

`RS_DT (U) , parameter U indicates collection of data .`

Step 1 : Compute the discernibility vector for all instances in U.

Step 2: Compute the redundancy of all condition attributes in all discernibility vectors.

Step 3: Choose the attribute with the maximal number as the node of this layer of current branch .

Step 4: Construct decision tree on the current branch according to the possible values of the selected attribute that appear in U .

Step 5: For each branch of the selected attribute of the decision tree, if it has not reached leaf node then call RS_DT(U) with the collection of the data of this branch as its parameter .

Step 6 : Return .

Example:

We will illustrate the main idea of our method by way of an example. Suppose, we have an information table that shown in table 4.

Table 4. A sample Information Table

Suppose, we have an information table

U \ A	a	b	c	d
u_1	a_o	b_1	c_1	y
u_2	a_o	b_o	c_1	y
u_3	a_1	b_1	c_o	n
u_4	a_o	b_2	c_1	n
u_5	a_1	b_1	c_1	y

that shown in table1. Let U be a set of all instances, select one instance u from U. Using the idea of discernibility matrix, create a discernibility vector (i.e., the row or column with respect to u in the discernibility matrix) for u. For the example u_1=a_0 b_1 c_1 is as in table 5, where λ means that we donot care the element since it is with the same class. And so on u_2(y), u_3(n), u_4(n), u_5(y) compute discernibility vector.

Table 5: First discernibility vectors of instances

U \ U	u_1(y)	u_2(y)	u_3(n)	u_4(n)	u_5(y)
u_1 (y)	λ	λ	a,c	b	λ
u_2 (y)	λ	λ	a,b,c	b	λ
u_3 (n)	a,c	a,b,c	λ	λ	c
u_4 (n)	b	b	λ	λ	a,b
u_5 (y)	λ	λ	c	a,b	λ

Now, compute the redundancy of each attributes condition in all discernibility vectors. The attribute which has maximal number is the root. The redundancy of ‘b’ is 8 and redundancy of ‘a’ the same as redundancy of ‘c’ is 6. Therefore attribute 'b' is selected as the root of the decision tree, and the data set will be partitioned into three data subsets. In each data subset, all tuples have the same value of the attribuate 'b', if all tuples in the same branch have the same class label then no action is needed to construct the decision tree to a deeper layer. Otherwise, another attribute will have to be chosen as node to construct the decision tree to further depth until the tree reaches leaf node that all tuples in the leaves have the same class label, see figure 1.

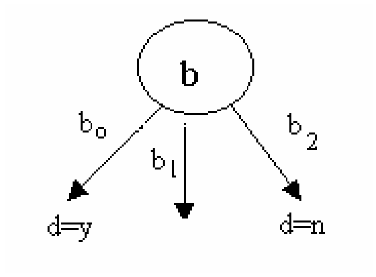

Figure4. The root of decision tree

Since condition attribute 'b' has been used in upper layer of the tree, only the other two condition attributes (attribute 'a' and 'c') are tested on branch $b=b_1$, we obtain the discernibility vector of those instances u_1, u_3, u_5 as see in table 6, and compute the reducency of attributes condition ‘a’ and ‘c’.

Table 6: Second discernibility vectors of instances

U \ U	u_1(y)	u_3(n)	u_5(y)
u_1 (y)	λ	a,c	λ
u_3 (n)	a,c	λ	c
u_5 (y)	λ	c	λ

Since condition attribute 'c' has maximal number (equal 4) , 'c' is selected as the winner of branch $b=b_1$. the class labels of the leaf node of the two branches $c=c_0$ and $c=c_1$ are labelled in accordance with the class labels of the tuples of the corresponding branches, i.e labelled to be d=y and d=n.

Since $b=b_0$ and $b=b_2$ need no further classification, the final decision tree can be obtained as shown in figure 5.

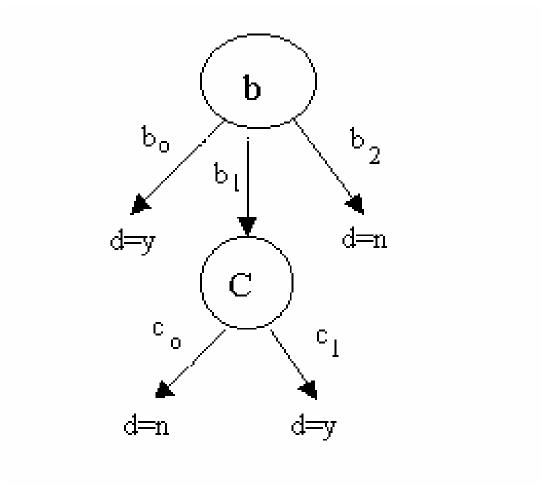

Figure 5. Final of decision tree

From a root node to leaf node, rules can be obtained

r_1. $(b=b_0) \rightarrow (d=y)$

r_2. $(b=b_2) \rightarrow (d=n)$

r_3. $(b=b_1, c=c_1) \rightarrow (d=y)$

r_4. $(b=b_1, c=c_0) \rightarrow (d=n)$

4. Hybrid Method (Evolution Rough Sets)

In this Section we describe the main characteristics of our method for classification. This is a hybrid method that combines decision tree (RS_DT) and genetic probabilistic rough induction (GA+GDT_RS).

The basic idea is to use a decision tree (RS_DT) algorithm to produce rules before run the algorithm of genetic probabilistic rough induction (GA+GDT_RS).

The method discovers rules into two training phases. In the first phase it runs RS_DT decision tree induction algorithm [1]. The induced, simplified tree by Fisher's exact test [?] is transfored into a set of rules. This rule set can be thought of as expressed in an initial population.

The second phase consists of using a GA+GDT_RS genetic probabilistic rough induction to discover rules. This two-phase process is summarized in Figure 6.

Algorithm of the Method

We now describe an idea of searching algorithm for a set of rules. We use a sample database shown in Table 4 to illustrate the idea. The input is data in the form of a decision table structure, the output is the set of rules, and the process is as follows.

Step 1. Read the decision table, determine the thresholds Tnoise and λ for classification accuracy, if the decision table contains inconstent data do steps 1-3 in algorithm of GDT-RS, otherwise goto step2.

Step 2. By a decision table generate a decision tree by running RS_DT algorithm, simplified the tree and convert the tree as an initial population.

Step 3. Iteratively perform the following substeps until the maximum of generation, G, is reached.

a. Evaluate each individual in the population via the following:

   i. Calculate the classification accuracy and rule converage for each chromosome (rule) as we explained in section 2.2.2, see in table 6, and remove the rule if its accuracy or converage is less than the thresholds λ.

ii. Use a percentage of converageEq.( ) as a fitness measure to assign a value for this set of rules, see table 7.

Create a new population by applying the followin operations:

i. Reproduce an existing individual (selected based on its fitness) and copy it into the new population.

ii. Create two new individuals from two existing individuals by genetically recombining randomly chosen parts of two existing chromosomes using crossover operation.

iii. Create a new individual from an existing one by mutating a randomly chosen part of the selected chromosome usin the mutation operation.

iv. Insert these chromosomes into the new population.

Step 5. The best-so-far individual (the individual that has the highest fitness value over all the generations) is designated as the result of the run.

4.1 Genetic Operators

To evolve a population of individuals we need to apply genetic operators that produce new individuals (which will from the next generation) from the best individuals of the current generation.

We briefly describe below the genetic operators used by our hybrid system.

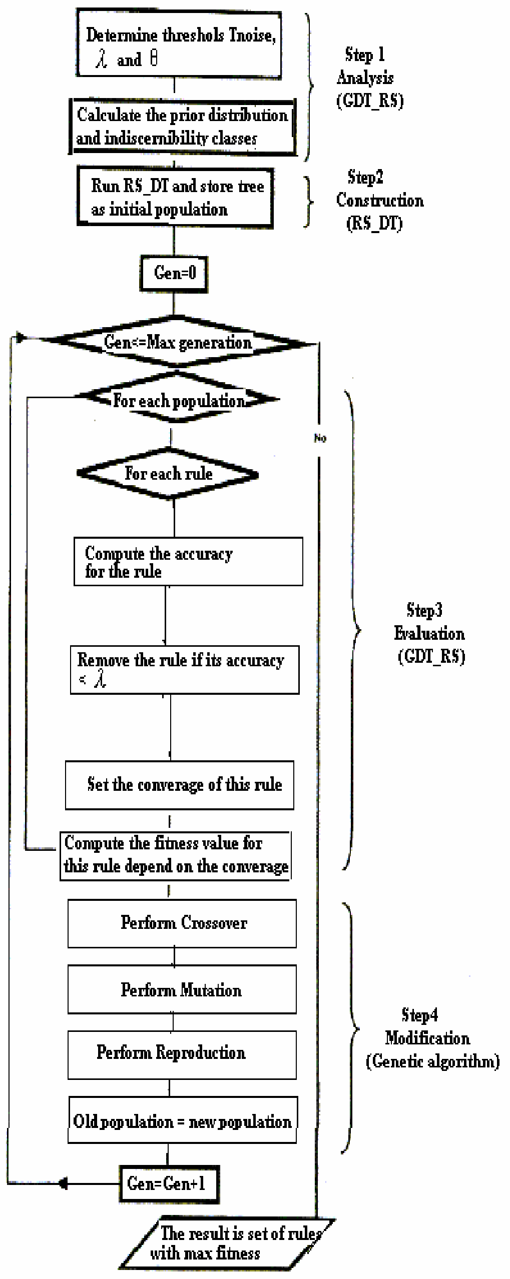
Determine threshols Tnoise, λ and θ
Calculate the prior distribution and indiscernibility classes
Step 1 Analysis (GDT_RS)
Run RS_DT and store tree as initial population
Step2 Construction (RS_DT)
Gen=0
Gen<=Max generation
For each population
No
For each rule
Compute the accuracy for the rule
Remove the rule if its accuracy < λ
Set the converage of this rule
Compute the fitness value for this rule depend on the converage
Step3 Evaluation (GDT_RS)
Perform Crossover
Perform Mutation
Perform Reproduction
Old population = new population
Step4 Modification (Genetic algorithm)
Gen=Gen+1
The result is set of rules with max fitness

Figure 1: Overview of our hybrid method for classification.

We used a population size of 100 individuals a parameter value often used in the literature [2] and adequate for our experiment.

The reproduction operator was roulette wheel selection with elitism. Out of the 100 individuals, 98 are reproduced by roulette wheel selection and the best 2 individuals of the current population are passed on, unaltered, to the next generation, in an elitism-reproduction scheme. Elitism avoids the danger of possible loss of the best two individuals of the current generation. Note that this loss might occur if roulette wheel selection was used alone (without elitism), due to the stochastic nature of roulette wheel selection.

The individuals selected for reproduction undergo the application of two other stochastic operators, namely mutation and crossover. In our experiments reported in this chapter mutation is applied with 0.05 probability. Nevertheless, for third Monk's problem data and medical data, we increase the probability rate for the mutation opertor to 0.2. And crossover with 0.8 probability.

4.2 Fitness Function and Evaluation of the Set of Rules

Decision tree (RS_DT) method induces a set of rules. First we simpilifed these rules. Second, we delete from this set of rules any duplicate ones. Third, we use these rules as an initial population of a hybrid system genetic probabilistic rough induction GA+GDT_RS, after converage or after the number of iteration. Any weak rule which its fitness value is less than some threshold, will be removed from the set of rules. Therefore, we evaluated the whole set of rules as ???????

We only use training cases to generate rules and testing cases to measure the efficient of the method.

We define the fitness function of the individual by combining probabilistic rough induction in genetic algorithm and use percentage of how many instances in probabilistic rough induction covers this individual.

Depending on the fitness value, the genetic operators select the individual to be processed (the better the fitness, the more likely the individual is to be selected).

The hybrid method RS_DT/GA+GDT_RS proposed here is slower than a decision tree RS_DT method in the processing time. But on the other hand hybrid method RS_DT/GA+GDT_RS gives more accuracy and less number of rules than a decision tree RS_DT as we shown in next Section. So the incorporation of knowledge into evolutionary algorithms is interesting of optimization and machine learning problems.

5 Applications and Results

We have evaluated the performance of our hybrid method RS_DT/GA+GDT_RS across dataset taken from [6].

The experiment used RS_DT [1] as the decision tree component of our hybrid method. The performance of the hybrid method RS_DT/GA+GDT_RS was compared with the performance of RS_DT method and standard rough set method for the number of rules and accuracy.

We divided this dataset to 80% training and 20% testing, the results of computations of rules have been only done to training data, and the accuracy computed on testing data.

Figure 2 shows a comparison of the number of rules obtained from RS_DT method, standard rough set method and our hybrid method. Figure 3 presents the acuracy of dataset on RS_DT method, standard rough set method and our hybrid method,

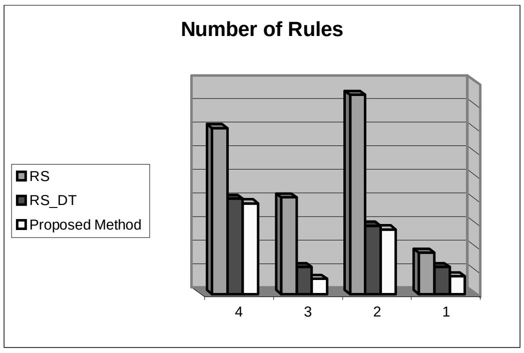

Figure 2 Comparsion of number of rules on datasets

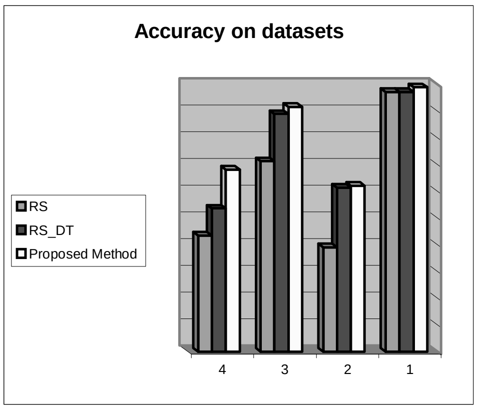

Figure 3 The accuracy on datasets

Since we are started with decision tree RS_DT, and the results with RS_DT are the initial population of hybrid system GA+GDT_RS. So the results obtained from hybrid system GA+GDT_RS are the same as of decision tree RS_DT or better.

The set of rules that is produced by our new method contains the fewest rules with better accuracy rate than that is produced by the decision tree RS_DT.

For example, if we take Monk's data problem, in the third Monk's problem data there exist a rule produced from decision tree RS_DT: IF attribute#1=2 AND attribute#5=4 THEN Class 0 with certainty factor 95%. This rule is reduced in our method to the following: IF attribute#5=4 THEN Class 0, with certainty factor 95.16%. Some rules in decision tree (RS_DT) are presented in the list of rules produced form hybrid system (RS_DT/GA+GDT_RS) such as: IF attribute#2=3 AND attribute#5=2 THEN Class 0, because the fitness function is high, and other rules from RS_DT is omitted, because rules with high fitness will appeared.

The processing time of our hybrid decision tree RS_DT/GA+GDT_RS is the half of the processing time of genetic probabilistic rough induction GA+GDT_RS or less, i.e., for example if the genetic probabilistic rough induction method runs the first Monk's problem data in processing time two an hours, in hybrid decision tree RS_DT/GA+GDT_RS runs the same dataset in one an hour or less.

6 Conclusions

In this paper, we have described a hybrid system RS_DT/GA+GDT_RS and compared it's performance with the performance of decision tree RS_DT and standard rough set method on some dataset.

We showed that a hybrid system can be used to improve the performance of decision tree. A hybrid system as we saw with three goals: increase the accuracy of dataset, reduce the number of rules if we compared with decision tree (RS_DT) and standard rough set method. Also reduce the processing time if we compared with standard genetic algorithm method.

References

[1] Badawy, O., Hassan, Y., and Sagar, W., Rough Set Approach to Decision Tree Construction, Fourth International Conference on Informatics and Systems(INFOS-2006), 2006.

[2] Carvalho, R., and Freitas, A., A Genetic Algorithm with Sequential Niching for Discovery Small-Disjunt Rules, Proceedings Genetic and Evolutionary Computation Conf. (GECCO-2002), Morgan Kaufmann, pp.1035 – 1042, 2002.

[3] Corcoran, A., and Wairwright, R., GA library written in C (LibGA), URL: www.aic.nrl.navy.mil/galist/src

[4] Dietterich., G., Ensemble Methods in Machine Learning. In: Firt International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science, Springer-Verlag, pp. 1-15, 2000.

[5] Goldberg, D., Genetic Algorithm in search, Optimization and Machine Learning, Addison-Wesley, 1989.

[6] Merz, C., and Murphy, P., UCI Repository of Machine Learning Database. http://www.ics.uci.edu/~mlearn/mlrepository.html. Dept. Information and

Computer Science, University of Californi, Irvine, CA, 1998.

[7] Thrun, S., and Pratt, L., Learning to Learn, Kluwer Academic, 1998.

Table 7: Results for a sample database

U	Rules	s(P)	Accuracy	Converage
u_1'	$b_0 \rightarrow y$	0.25	0.67	1
u_2,u_7	$b_1 \wedge c_1 \rightarrow y$	1	1	2
u_4	$c_o \rightarrow n$	0.17	1	1
u_6	$b_2 \rightarrow n$	0.25	1	1