Mining Comprehensible and Interesting Rules: A Genetic Algorithm Approach

Jyoti Vashishtha
Department of CSE
GJUST, Hisar

Dharminder Kumar
Department of CSE
GJUST, Hisar

Saroj Ratnoo
Department of CSE
GJUST, Hisar

Kapila Kundu
Department of CSE
GJUST, Hisar

ABSTRACT

A majority of contribution in the domain of rule mining overemphasize on maximizing the predictive accuracy of the discovered patterns. The user-oriented criteria such as comprehensibility and interestingness are have been given secondary importance. Recently, it has been widely acknowledged that even highly accurate discovered knowledge might be worthless if it scores low on the qualitative parameters of comprehensibility and interestingness. This paper presents a classification algorithm based on evolutionary approach that discovers comprehensible and interesting in CNF form in which along with conjunction in between various attributes there is disjunction among the values of an attribute. A flexible encoding scheme, genetic operators with appropriate syntactic constraints and a suitable fitness function to measure the goodness of rules are proposed for effective evolution of rule sets. The proposed genetic algorithm is validated on several datasets of UCI data set repository and experimental results are presented which clearly indicate lower error rates and more comprehensibility across a range of datasets. Some of the rules show the interesting and valuable nuggets of knowledge discovered from small disjuncts of high accuracy and low support which are very difficult to capture otherwise.

General Terms

Data Mining, Evolutionary algorithms

Keywords

Comprehensibility, interestingness, classification rules, genetic algorithm

1. INTRODUCTION

Classification in machine learning is a technique that is used to predict group membership of data instances in a dataset [1]. An instance or a record consists of a set of 'm' predicting attributes and a goal attribute. General relationships between predicting and goal attributes are discovered on training data and then these relationships are validated on test data.

Several kinds of techniques exist in literature for classification rule mining such as Decision Trees, Neural Networks, Bayesian nets, K-nearest neighbors, Fuzzy Logic, Genetic Algorithms, and Support Vector Machines etc. For last few decades, genetic algorithms (GA) have been extensively employed in rule mining and revealed promising results [2, 3, 4, 5, 6, 7, 8, 9]. The GA has the capability of avoiding the convergence to local optimal solutions and it also takes care of attribute interactions while evolving rules whereas most of the other rule induction methods tend to be quite sensitive to attribute-interaction problems. These methods select one attribute at a time and evaluate a partially constructed candidate solution, rather than a full candidate rule. In case of GA, all the interactions among attributes are taken into account and fitness function evaluates the individual as a whole [10].

In the classification task, the discovered knowledge usually represented in the form of *IF-THEN* production rules (PR), which have the advantage of being a high-level representation. The symbolic form of production rule for classification contains 'n' number of attribute value conditions which are separated by "AND" operator on the left hand side and a decision, based on conjunction of these attribute value conditions, on right hand side i.e.

$$If\ (\mathrm{C}_1 \wedge \mathrm{C}_2 \wedge \ldots\ldots \wedge \mathrm{C}_n) Then\ (\mathrm{D} = \mathrm{D}_l) \quad (1)$$

$condition\ C_i\ is\ represented\ as < A_j, Op, V_{jk} >, i \in [1, n]$
$A_j = j^{th}\ attribute\ in\ the\ dataset\ ,$
$Op = a\ relational\ operator,$
$V_{jk} = k^{th}\ value\ of\ j^{th}\ attribute$
$D_l = l^{th}\ class\ in\ the\ multi\ class\ dataset$

This type (1) of symbolic knowledge representation contributes towards the comprehensibility of the discovered knowledge however a large number of decision rules are produced even for smaller datasets. Although a rule set discovered with large number of rules may lead to higher predictive accuracy, comprehensibility remains yet low as users fail to gain complete insight into the application domain. Moreover, the rules with low support and high confidence are very often not captured by knowledge representation scheme given in (1). According to Suzuki [11] exceptions have low support and high confidence and discovery of such exceptions is challenging as well as interesting.

The purpose of this paper is to discover accurate, comprehensible, concise, simple and interesting knowledge in the form of classification rule sets from databases. It employs a crowding GA based scheme that mines Production Rules (PRs) in Conjunctive Normal Form (CNF) where a conjunctive relationship exists between two attributes and disjunction is there among the values of the same attribute. Moreover, an entropy based filter is also used to bias the initial population towards more relevant or informative attributes so that the GA starts with better fit rules covering many training instances. This initialization scheme is simple and produces better quality rules in the initial population leading to the discovery of significantly better quality rules in lesser number of generations.

In present study, the basic structure of the knowledge representation scheme is same, as shown in (1), with the

addition that each attribute is allowed to have more than one value with disjunction operator between them as shown below:

$$C_i = \left(< A_j\ Op\ V_{j1} > \vee < A_j\ Op\ V_{j2} > \vee \ldots\ldots \vee < A_j\ Op\ V_{jk} >\right)$$
$$where\ V_{jk} = k^{th}\ value\ of\ j^{th}\ attribute$$

The following rule structure in CNF form is used for mining classification rules.

$$If\ \left(< A_1\ Op\ V_{11} > \vee < A_1\ Op\ V_{12} > \vee \ldots\ldots \vee < A_1\ Op\ V_{1j} >\right)$$
$$\wedge \left(< A_2\ Op\ V_{21} > \vee < A_2\ Op\ V_{22} > \vee \ldots\ldots \vee < A_2\ Op\ V_{2j} > \ldots\right)$$
$$\wedge \ldots \wedge \left(< A_i\ Op\ V_{i1} > \vee < A_i\ Op\ V_{i2} > \vee \ldots \vee < A_i\ Op\ V_{ij} > \ldots\right)$$
$$Then\ CLASS \qquad (2)$$

For example, one of the rules discovered by proposed classifier using Mushroom data set is as follows:

$$\text{IF } \left((\text{odor} = \text{creosote}) \wedge (\text{odor} = \text{foul}) \wedge (\text{odor} = \text{musty}) \wedge (\text{odor} = \text{pungent})\right) \text{ THEN poisonous} \qquad (3)$$

For rule shown in (3), only 16 instances out of 5610 instances in Mushroom dataset match the rule condition 'Odor=Musty' and 'class= poisonous. This type of knowledge is not captured by the rule representation shown in (1) even though this rule condition has high confidence value. The knowledge mined by employing representation schemes in (2) is not only comprehensible, clear, and concise but also impound interesting facts and small disjuncts [20] which go undiscovered otherwise.

The rest of the paper is organized as follows: Section II provides a brief overview of several implementations of genetic algorithms for mining classification rules. Section III presents new GA design that includes encoding scheme, fitness function, genetic operators applied, along with covering algorithm. The data sets used for validation are presented in Section IV besides the analysis and summarization of the simulation results. Conclusions and future directions are given in Section V.

2. RELATED WORK

A Numerous attempts have been made to apply evolutionary algorithms (EAs) in data mining to tackle the problem of knowledge extraction and classification. Several GA designs, for discovering comprehensible classification rules, have been proposed in the literature. A brief review of some of the relatively more important of them is given below.

In early 90's, GA-based systems, GABIL [12], GIL [13], COGIN [14], REGAL [15], HDPDCS [16] etc. continually learned and refined the classification rules. GABIL was explored to produce rules achieving validation set accuracy comparable to that of decision trees induced using *ID3* and *C4.5*.

The evolutionary algorithms for rule induction GABIL, GIL, HDPDCS follow the Pittsburgh approach while COGIN and REGAL follow the Michigan approach. The Pittsburgh approach takes into account rule interaction directly during the computation of fitness function. However, each individual encodes a set of rules and individuals encoded with this approach are syntactically longer, thus making the optimal solution difficult to find when the search space is large. On the other hand, the Michigan approach encoded individuals are simpler and syntactically shorter, thus making the search of solutions easier and faster. In this approach, however, each rule is encoded and evaluated separately without taking into account the interactions among different rules [10, 17]. Both the encoding schemes have been with equal frequency in the domain of rule mining. Many others proposals such as XCS [18], SIA [19], UCS [20] and HIDER [21] cannot be categorized in any explicit framework. .A fixed encoding scheme is applied to the chromosomes and specific design is used for the mutation operator for GA designed by Fidelis et al. [3] to discover comprehensible classification rules. Another GA for discovering rules following the approach of associating a chromosome with a conjunction of conditions of a single rule is described in [22].

When most of the data mining work emphasized only on the predictive accuracy and comprehensibility, Noda et al. [2] put forward a GA approach designed to discover the interesting rules. Their GA was designed for dependence modeling, a data mining task which can be seen as a generalization of the classification task. Carvalho & Freitas [23, 24] proposed a hybrid approach for rule discovery that combines decision trees and genetic algorithms (GAs) where decision-tree algorithm is used to classify examples belonging to large disjuncts and a GA is employed to discover rules classifying examples belonging to small disjuncts. Decision-tree algorithms have a bias towards generality that is well suited for large disjuncts, but not for small disjuncts. On the other hand, GAs are robust, flexible algorithms which tend to cope well with attribute interactions [10, 17], and can be more easily tailored for coping with small disjuncts. A few researchers propose a pruning strategy using a GA to find the best combination of interesting rules with the maximized (or greater) accuracy [25]. Dehuri et al. [26] used an elitist multi-objective genetic algorithm for mining classification rules from large databases.

In recent studies, a lot of variations in the basic structure of PRs, are suggested by researchers for the discovery of accurate, comprehensible and interesting classification or pattern discovery. In this context genetic algorithms have been suggested for discovery of Censored Production Rules (CPRs), Hierarchical production rules (HPRs), Hierarchical censored production rules (HCPRs), Ripple Down Rules (RDRs) etc. [27-36]. These special kinds of knowledge structures augment exceptions to their corresponding commonsense rules of high generality and support.

According to the review of GA-based classification methods, it is revealed that a vast majority of algorithms discover large rule sets which may contain too much redundancy. Such rule sets are very difficult to be analyzed by a user and would be assigned a low grade in the comprehensibility criterion. The approach presented in this paper is successful in obtaining a rule set which is simpler containing very few decision rules, and therefore easier to understand, which is a clear advantage in many domains.

3. THE PROPOSED GA DESIGN

In the proposed GA, to keep the individuals simple, "Michigan-style approach" [1] is used to encode each solution in the population. The population in a simple GA would converge to a single best rule. Therefore a Crowding GA has been implemented with the intention of discovering a set of classification rules. Crowding is a generalization of pre-selection. It uses overlapping populations. In selection and reproduction a portion of the population is replaced by the newly generated individuals. The replacement is a distinct feature. Before replacement, the new offspring will execute a

comparison with individuals of the population using a distance function as a measure of similarity. The population member that is most similar to the offspring gets replaced. This strategy maintains the diversity in the population and avoids premature convergence of the traditional GA to the single best solution. The proposed GA discovers classification rules in CNF form as explained in equation (2) and (3). The present genetic approach works only with categorical (nominal) attributes. The continuous attributes are required to be discretized at preprocessing stage. The remaining details of GA are as given below.

3.1 Encoding

A rule contains two parts antecedent and consequent. Antecedent (IF part) of the rule consists of a conjunction conditions on the values of at most *n-1* predicting attributes, where *n* is the number of attributes being mined and consequent (THEN part) of the rule consists of a single condition specifying the value predicted for a given goal attribute. A binary genotypic approach is considered for encoding the conditions in antecedent part of classification rule and a single bit is used for consequent part of classification rule. In a genome, the 'm' number of genes or bits are used to encode an attribute with 'm' values. For example, an attribute-value condition, such as *Education status* and its three values - "uneducated", "educated" and "highly educated" would be encoded in the genome by three bits. If these bits take on, say, the values "0 1 1" then they would be representing the following rule antecedent: IF (*Education status* = "educated" OR *Education status* = "highly educated"). This encoding scheme allows the representation of conditions with internal disjunctions, i.e. the logical OR operator within an attribute condition. This encoding scheme is extended to represent rule antecedents with several attribute conditions (linked by a logical AND) and the length of genome is the number of bits to represent all attribute-value conditions plus one bit to represent antecedent. If all the *k* bits of a given attribute condition are "on" or "off", this means that the corresponding attribute is effectively being ignored by the rule antecedent.

For example, in data set 'Mushroom', 22 attributes have 125 values in all. Therefore, 125 genes/bits are used to encode antecedent part. The class attribute has two classes and one bit is used for its representation. Genome representation is given as:

Gene representation (125 Genes)																					Class
$A_1..A_4$	A_5									$A_6..A_{11}$							A_{12}				$A_{13}..A_{22}$	A_{23}
0-21	22-30									31-59							60-63				64-124	125
0..0	1	0	0	0	0	0	1	0	0	0..0							0	0	0	1	0..0	0

The corresponding rule (phenotype) for the above genotype is given below:

IF $(((\text{odor} = \text{almond}) \wedge (\text{odor} = \text{none})) \wedge (\text{ssa\_ring} = \text{smooth}))$ THEN edible

3.2 Initialization

The complete random initialization provides very poor initial rules and is not effective to discover optimal rule set. Therefore, the proposed algorithm uses the initial population biased towards attributes that are more relevant and informative instead of randomly generated initial population. A better fit initial population is likely to converge to more reliable rule set with high predictive accuracy in lesser number of generations, consecutively, enhances the performance of genetic algorithm for rule mining techniques. Also, in data mining applications with large datasets, this approach is anticipated to reduce the number of fitness evaluations significantly resulting into a considerable gain on efficiency front [37].

For example, in mushroom dataset, if attribute 'odor' is the more relevant attribute according to entropy calculation and probability of initialization, it would appear more frequently than any other attributes in the rules of initial population.

3.3 Fitness Function

Fitness functions are used to evaluate the quality of rule. The choice of the fitness function is very crucial as it leads the search by evolutionary algorithms towards the optimal solution. Several fitness functions have been suggested for discovery of accurate and interesting rules. Various factors like coverage, confidence factor, support, simplicity, entropy and predictive accuracy etc. are taken into account while evaluating the fitness function [4]. Before we can define the fitness function, it is necessary to recall a few basic concepts on classification rule evaluation. The evaluation is intended for evaluating single rules R_i (IF P THEN D_i; P is antecedent part and D_i is the consequent part) for classifying an example[38].

Precision: The *precision* of the rule R_i can be defined as:

$$Precision(R_i) = \frac{P \wedge D_i}{|P|}, where\ 0 \leq Precision(R_i) \leq 1$$

In set-theoretic terms, *precision* is the degree to which *P* is included in D_i. In classification problems, *precision* measure is defined as the ratio of the number of objects in *P* that are correctly classified as decision class of D_i and the number of objects in *P*. In data mining literature the alternative names for *precision* measure are *confidence*, *accuracy*, *strength* and *consistency*. They are essentially referred to the same measure.

Coverage: The *coverage* of the rule R_i can be defined as:

$$Coverage(R_i) = \frac{P \wedge D_i}{|D|}, where\ 0 \leq Coverage(R_i) \leq 1$$

Coverage reflects the applicability of the rule and measures the proportion of actual positives which are correctly identified. In classification problems, *coverage* measure is defined as the ratio of the number of objects in *P* that are correctly classified as decision class of D_i and the number of objects satisfying D_i. Other alternative names for *coverage* measure are *recall*, *completeness* and *sensitivity.*

Both *precision* and *coverage* are important indicators of a rule's reliability. A higher *precision* suggests a strong association between P and D_i. A higher *coverage* suggests that the relationships of more objects can be derived from the rule. The *precision* and *coverage* are not independent of each other; one may observe a tradeoff between *precision* and *coverage*. A rule with higher *coverage* may have a lower *precision*, while a rule with higher *precision* may have a lower *coverage*.

Generality: The *generality* of the rule R_i can be defined as:

$$Generality(R_i) = \frac{P \wedge D_i}{|U|}, where\ 0 \leq Generality(R_i) \leq 1$$

It indicates the relative size of the subset *P*. A subset is more general if it covers more instances of the universe U.

Simplicity: The *simplicity* of the rule R_i can be defined by *number of attributes* on the left hand side of the rule.

$$Simplicity_{r_i} = \#\ of\ attributes\ in\ the\ rule$$

A rule is considered more complex if it contains more attributes.

In this subsection, the fitness for a rule r_i for decision D = D_k (k^{th} class in the database) is computed as per the following formula:

$$Fitness_{r_i} = \frac{(Precision_{r_i} * Coverage_{r_i})}{Simplicity_{r_i}}$$

A single objective fitness function facilitate to achieve the target of discovering classification rules with multiple criteria i.e. accuracy, comprehensibility and interestingness.

3.4 Genetic Operators

Genetic operators are being used to manipulate or recombine the genetic material of candidate rules and introduce new genetic material.

3.4.1 Initialization

The basic part of the selection process is to stochastically select from one generation to create the basis of the next generation. The requirement is that the fittest individuals have a greater chance of survival than weaker ones. A number of selection methods - roulette wheel, tournament, rank-based, top percent etc. are described and used in literature so far. But the most commonly and widely used section method is roulette selection method. It is used as the fitness proportionate selection operator for selecting potentially useful solutions for recombination. In this method of selection, firstly, the total fitness value of all the individuals in the population is calculated and then a random number *'n'* is generated between 0 to total fitness, and finally, the population member whose fitness, added to the preceding population members, is greater or equal to *'n'*, is selected. In present design, we are applying Roulette wheel method for selection.

3.4.2 Crossover

Crossover is the process by which genetic material from one parent is combined with genetic material from the second parent to produce potentially promising new offspring. The intention of this operation is to avoid the solution process to converge to local optimum. Main types of crossover operations are one-point crossover, two-point crossover, uniform crossover etc. In present work, one-point crossover is used. However, in our case, selected crossover point is carefully chosen only to be the starting point of any attribute in the genome. For example, consider the one point of crossover is between genes 30 and 31 where attribute A5 ends and A6 starts.

Before crossover

Gene representation (125 Genes)					class
$A_1..A_4$	A_5	$A_6..A_{11}$	A_{12}	$A_{13}..A_{22}$	A_{23}
0-21	22-30	31-59	60-63	64-124	125
0..0	0 0 0 0 0 0 1 0	0..0	0 0 1 0	0..0	0

IF ((odor = pungent) ∧ (ssa_ring = silky)) THEN edible

Gene representation (125 Genes)					class
$A_1..A_4$	A_5	$A_6..A_{11}$	A_{12}	$A_{13}..A_{22}$	A_{23}
0-21	22-30	31-59	60-63	64-124	125
0..0	1 0 0 0 0 0 1 0 0	0..0	0 0 0 1	0..0	1

IF (((odor = almond) ∨(odor = none)) ∧ (ssa_ring = smooth))THEN poisonous

After crossover

Gene representation (125 Genes)					class
$A_1..A_4$	A_5	$A_6..A_{11}$	A_{12}	$A_{13}..A_{22}$	A_{23}
0-21	22-30	31-59	60-63	64-124	125
0..0	0 0 0 0 0 0 0 1 0	0..0	0 0 0 1	0..0	1

IF ((odor = pungent) ∧ (ssa_ring = smooth)) THEN poisonous

Gene representation (125 Genes)					class
$A_1..A_4$	A_5	$A_6..A_{11}$	A_{12}	$A_{13}..A_{22}$	A_{23}
0-21	22-30	31-59	60-63	64-124	125
0..0	1 0 0 0 0 0 1 0 0	0..0	0 0 1 0	0..0	0

IF ((odor = almond) ∨ (odor = none) ∧ (ssa_ring = silky)) THEN edible

The crossover is carried out according to the crossover probability. Crossover probability is a parameter of GA for the application of crossover with moderation. It indicates a ratio of how many couples would be picked for mating.

3.4.3 Mutation

The purpose of mutation in GAs is preserving and introducing diversity. Mutation may specialize or generalize a candidate rule by inserting or removing conditional clauses in the antecedent part of the rule. It is a random deformation of chromosomes with a certain probability in order to preserve genetic diversity and to avoid local optimum. As compare to crossover probability, mutation probability is kept low. Higher mutation probability can turn the genetic algorithm in a random search algorithm. For instance, if a binary coded chromosome has 100 genes and mutation probability is 1% then it means 1gene out of 100 would be picked and flipped. In our GA implementation, mutation is applied only on the bits of selected attributes (with a probability that is directly proportional to their relative importance) i.e. an attribute with higher relative relevance is mutated with more probability as compared to an attribute with lower importance. If mutation is applied on all attributes then it keeps on exploring the search space more wildly instead of exploring the search space in the optimal rule set direction. This technique leads to faster discovery of better rule set.

For example, in mushroom data set the attribute 'odor' is highly relevant to contribute to make rules. Mutating the value of this attribute leads discovery of good rules.

Before mutation,

Gene representation (125 Genes)					class
$A_1..A_4$	A_5	$A_6..A_{11}$	A_{12}	$A_{13}..A_{22}$	A_{23}
0-21	22-30	31-59	60-63	64-124	125
0..0	0 0 0 0 0 0 0 1 0	0..0	0 0 1 0	0..0	0

IF ((odor = pungent) ∧ (ss_above_ring = silky)) THEN edible

After mutation,

Gene representation (125 Genes)																						class
$A_1..A_4$	A_5									$A_6..A_{11}$	A_{12}				$A_{13}..A_{22}$							A_{23}
0-21	22-30									31-59	60-63				64-124							125
0..0	0	0	0	0	0	0	1	1	0	0..0	0	0	1	0	0..0							0

IF ((odor = pungent) ∨ (odor = none)) ∧ (ssa_ring = silky)) THEN edible

4. EXPERIMENTAL SETUP AND RESULTS

The proposed GA approach is implemented using GALIB247 on a Pentium core 2 duo processor with Ubuntu release 9.10 as operating system. The performance of the suggested approach is validated on five sets of data obtained from UCI Machine Learning Repository, which is a collection of widely used benchmark and real-world data sets for data mining and KDD community. The datasets are described in Table 1.

Table 1. Description of datasets used for experimentation

Name of Dataset	No. of Examples	No.of Attributes	No. of Classes	No. of alleles with class attribute
Mushroom(MS)	5610	22	2	126
Car (CR)	1728	6	4	22
Nursery (NS)	12960	9	4	28
Diabetes (DB)	768	8	2	31
Breast Cancer (BC)	699	9	2	91
German Credit (GC)	1000	21	2	102

Each data set is unique in terms of number of attributes and number of examples in Table 1. The data sets, Mushroom and Nursery have relatively large candidate rules search space; Mushroom and German Credit have more number of attributes and alleles as compared to the other data sets. The Diabeties and German Credit, data sets have mixed attributes. The continuous attributes are discretized using entropy based discretization technique before applying the genetic algorithms. The datasets are divided into training (60%) and tests sets (40%). The reason for taking training sample larger is to capture extraordinary interesting conditions which may be present in very few examples anywhere in the dataset. During the rule discovery, the data specific parameters and other parameter values set for GA, are listed in Table 2.

Table 2. List of Parameters used for simulation

Data set	No. of alleles with class	Popsize	No. of Generations	Crossover Probability	Mutation Probability
MS	126	50	200	0.66	0.01
CR	22	40	50	0.8	0.01
NS	28	25	50	0.66	0.01
DB	31	50	100	0.66	0.01
BC	91	50	200	0.8	0.01
GC	102	50	150	0.66	0.01

The experimental design is composed of three phases viz. preprocessing phase, genetic algorithm phase and post processing phase. The experimental design is depicted in figure 1 and the underlying procedure for rule discovery is given in Annexure I.

The Table 3 and 4 show the rules produced by simple GA and the proposed GA, respectively on Mushroom data set. All results shown are the best scores taken as average of 10 runs of GAs. The number of rules discovered by simple GA is six whereas the proposed GA produces only three rules. At the same time

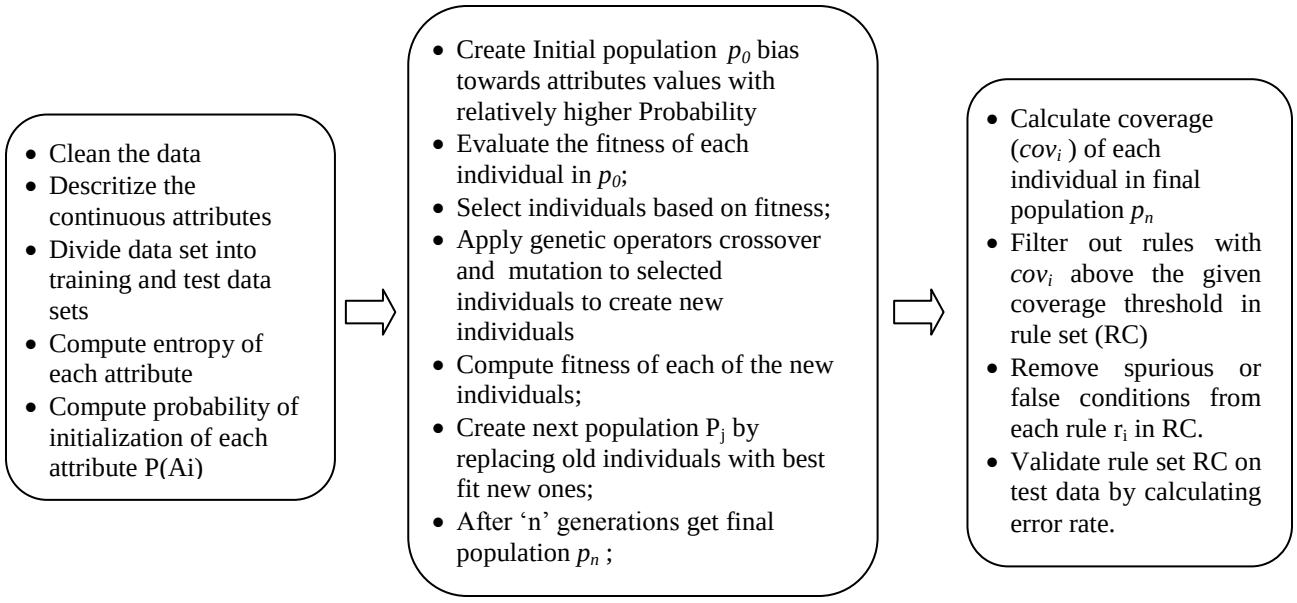

Figure1: Experimental Design

disjunction among the values of an attribute, in the rules discovered by proposed GA, increases the fitness values and makes the rule concise and more comprehensible. The rule number 3 in Table 4 is an interesting as it represents an exception stating that an odorless mushroom becomes poisonous if its spore print color is green. This rule is interesting as well as useful as it helps us in identifying an odorless mushroom which is poisonous.

Table 3. Rules generated by using simple GA on Mushroom dataset

Rule #	Rule Description	Generality	Precision	Coverage	Simplicity	Fitness
R_1	*If* (odor = none) *Then* edible	0.477	0.969	0.77	1.0	0.462
R_2	*If* (odor = pungent) *Then* poisonous	0.046	1.0	0.59	1.0	0.046
R_3	*If* (odor = almond) *Then* edible	0.071	1.0	0.12	1.0	0.071
R_4	*If* (odor = foul) *Then* poisonous	0.282	1.0	0.74	1.0	0.282
R_5	*If* (odor = creosote) *Then* poisonous	0.034	1.0	0.09	1.0	0.034
R_6	*If* (odor = anise) *Then* edible	0.071	1.0	0.12	1.0	0.071

Table 4. Rules generated by using proposed GA on Mushroom dataset

Rule #	Rule Description	Generality	Precision	Coverage	Simplicity	Fitness
R_1	*If* (odor = almond ∨ anise ∨ none) *Then* edible	0.618	0.973	1.0	1.0	0.973
R_2	*If* (odor = creosote ∨ foul ∨ musty ∨ pungent) *Then* poisonous	0.365	0.999	0.956	1.0	0.956
R_3	*If* ((odor = none)) ∧ ((spore_print_color = chocolate ∨ green) *Then* poisonous	0.017	0.944	0.79	2.0	0.373

For classification problems, it is natural to measure a classifier's performance in terms of the *error rate* and *comprehensibility (the number of rules in the discovered rule set)*. The classifier predicts the class of each instance: if it is correct, that is counted as a *success;* if not, it is an *error*. The error rate is just the proportion of errors made over a whole set of instances, and it measures the overall performance of the classifier. The results of proposed GA based classifier are put side by side with JRIP, Naïve Bayes and J48 methods for knowledge discovery, shown in Table 5. The performance of proposed GA(P GA) is shown in figures 2 and 3 in comparison of other rule discovery methods viz. JRIP, J48 and Naïve Bayes. In figure 2, the datasets are depicted on horizontal axis and numbers of rules discovered are shown vertically. The proposed GA has outperformed with the discovery of comparatively very less number of rules. In figure 3, the error rate is depicted vertically and a significant difference is observed in the error rates of other methods in comparison to the proposed GA.

Table 5. Comparison of number of rules(NR) and error rate(ER)

Data Set	JRIP		Naïve Bayes		J48		Proposed GA	
	NR	ER	NR	ER	NR	ER	NR	ER
MS	6.0	0.0	-	1.9	25.0	0.0	3.0	0.3
CR	27.0	16.8	-	11.5	131.0	7.6	6.0	2.5
NS	131.0	4.2	-	9.5	359.0	3.6	4.0	2.7
DB	3.0	21.8	-	24.1	20.0	26.4	4.0	0.4
BC	12.0	4.0	-	3.3	55.0	5.5	3.5	3.0
GC	3.0	31.0	-	23.5	97.0	29.2	5.5	13.0

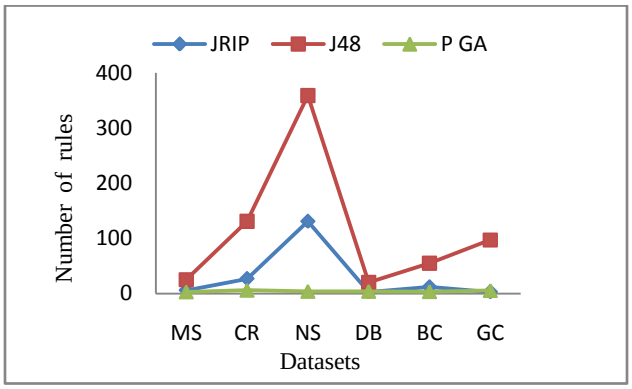

Figure 2: Performance of proposed GA in comparison of other techniques

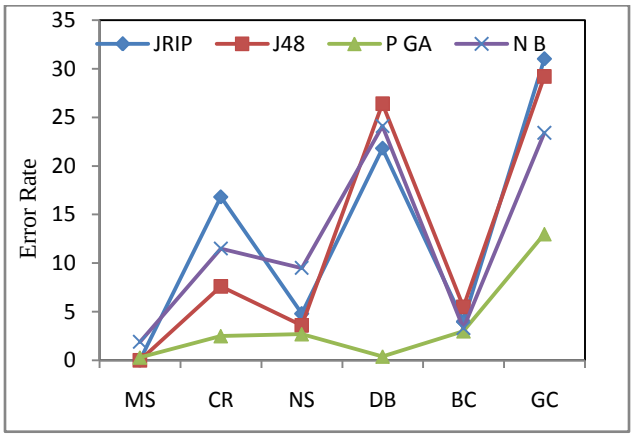

Figure 3: Performance of proposed GA in comparison of other techniques

5. CONCLUSION

In the present work, an evolutionary approach is proposed for the discovery of production rules in CNF form that allows conjunction in between various attributes and disjunction among the values of an attribute. The crowding genetic algorithm scheme has been devised with flexible chromosome encoding, appropriate crossover and mutation operators, and appropriate fitness function. The underlying rule encoding in CNF form successfully captures the attention-grabbing and valuable nuggets of knowledge that have relatively low support and high confidence. The suggested approach is tested on six datasets from UCI machine learning repository and the rule sets discovered are not only comprehensible, concise and accurate with lower error rate percentage but also useful and interesting. One of the most important future research directions would be the discovery of inter class and intra class exceptions from datasets using evolutionary algorithm.

6. REFERENCES

[1] Han, J., Kamber, M. and Pei, J. 2011. Data Mining: Concepts and Techniques, Third Edition, Morgan Kaufmann.

[2] Noda E., Freitas A.A. and Lopes H.S. 1999. Discovering interesting prediction rules with a genetic algorithm. In Proceedings of the 1999 Congress on Evolutionary Computation, IEEE Washington, vol. 2, pp.1322-1329

[3] Fidelis, M.V., Lopes, H.S., Freitas, A.A. and Grossa, P. 2000. Discovering Comprehensible Classification Rules with a Genetic Algorithm. In Proceedings of the 2000 congress on Evolutionary Computation, La Jolla, CA, USA, IEEE, vol. 1, pp. 805--810.

[4] Freitas, A.A. 2003. A survey of evolutionary algorithms for data mining and knowledge discovery. In: A. Ghosh and S. Tsutsui (Eds.) Advances in Evolutionary Computation, Springer-Verlag New York, pp.819-845.

[5] Romão, W., Freitas, A.A. and Gimenes, I.M.D.S. 2004. Discovering interesting knowledge from a science and technology database with a genetic algorithm. Applied Soft Computing, vol. 4, pp. 121-137.

[6] Freitas, A.A. and Carvalho, D.R. 2002. A genetic algorithm with sequential niching for discovering small-disjunct rules. Applied Soft Computation, vol. 2, 75-88.

[7] Bharadwaj, K.K. and Basheer, M. Al-Maqaleh. 2005. Evolutionary approach for automated discovery of censored production rules. Enformatika. vol. 10, pp. 147-152.

[8] Jiao, L., Liu, J. and Zhong, W. 2006. An organizational co-evolutionary algorithm for classification. Transactions on Evolutionary Computation, IEEE, vol. 10, pp. 67-80.

[9] Saroj, Bharadwaj, K.K. 2007. A parallel genetic algorithm approach for automated discovery of censored production rules. In Proceedings of International Conference on Artificial Intelligence and Application (IASTED), ACTA Press, Innsbruck, Austria. pp. 435-441.

[10] Freitas, A. A. 2002. Data mining and knowledge discovery with evolutionary algorithms. Natural Computing Series, Springer-Verlag, New York.

[11] Suzuki, E., Zytkow, J.M. 2005. Unified algorithm for undirected discovery of exception rules. International Journal of Intelligent Systems, vol. 20(7), pp. 673 - 691.

[12] DeJong, K.A., Spears, W.M.; Gordon, D.F. 1993. Using genetic algorithms for concept learning. Machine Learning, vol. 13, pp. 161-188.

[13] Janikow, C.Z. 1993. A knowledge-intensive genetic algorithm for supervised learning. Machine Learning, vol. 13, pp. 189-228.

[14] Greene, D.P. and Smith, S.F. 1993. Competition-based induction of decision models from examples. Machine Learning vol. 13, 229-257.

[15] Giordana, A. and Neri, F. 1995. Search-intensive concept induction. evolutionary computation vol. 3(4), pp. 375-416.

[16] Pei, M., Goodman, E.D., Punch III, W.F. 1997. Pattern discovery from data using genetic algorithms. In proceedings of first Pacific-Asia Conference. Knowledge Discovery and Data Mining, 1997.

[17] Freitas, A.A. 2008. A review of evolutionary algorithms for data mining. Soft Computing for Knowledge Discovery and Data Mining, O. Maimon and L. Rokach, Eds., Boston, MA: Springer US, pp. 79-111.

[18] Wilson, S. W. 1995. Classifier fitness based on accuracy. Evolutionary Computation, vol. 3(2), pp. 149–175.

[19] Venturini, G. 1993. SIA: a supervised inductive algorithm with genetic search for learning attributes based concepts. In Machine Learning ECML–93, ser. LNAI, P. Brazdil, Ed., Springer, vol. 667, pp. 280–296.

[20] Bernad´o-Mansilla, E. and Garrell, J. M. 2003. Accuracy–based learning classifier systems: models, analysis and applications to classification tasks. Evolutionary Computation, vol. 11(3), pp. 209–238.

[21] Aguilar-Ruiz, J. S., Gir´aldez, R. and Riquelme, J. C. 2007. Natural encoding for evolutionary supervised learning. IEEE Transactions on Evolutionary Computation, vol. 11(4), pp. 466–479.

[22] Freitas, A.A. 1999. A genetic algorithm for generalized rule induction. In: Roy, et al. (Eds.), Advances in Soft Computing—Engineering Design and Manufacturing, Springer-Verlag, pp. 340–353.

[23] Carvalho, D.E. and Freitas, A. A. 2000. A genetic algorithm-based solution for the problem of small disjuncts. In: D.A. Zighed, J. Komorowski, and J. Zytkow (Eds.), Principles of Data Mining and Knowledge Discovery (Proceedings of Fourth European Conference PKDD-2000, Lyon, France), Lecture Notes in Artificial Intelligence, Berlin, Springer-Verlag. vol. 1910, pp 345-352.

[24] Carvalho, D.R. and Freitas, A.A. 2002. A genetic-algorithm for discovering small-disjunct rules in data mining. Applied Soft Computing, vol. 2, pp. 75-88.

[25] Gopalan, J., Korkmaz, E., Alhajj, R. and Barker, K. 2005. Effective data mining by integrating genetic algorithm into the data preprocessing phase. Fourth International

Conference on Machine Learning and Applications (ICMLA'05), Los Angeles, CA, USA: , pp. 331-336.

[26] Dehuri, S., Patnaik, S., Ghosh, A. and Mall, R. 2008. Application of elitist multi-objective genetic algorithm for classification rule generation. Applied Soft Computing, vol. 8, pp. 477-487.

[27] Bharadwaj, K. K., and Al-Maqaleh B. M. 2005. Evolutionary approach for automated discovery of censored production rules. In Proceedings of 8th International Conference on Cybernetics, Informatics and Systemics (CIS-2005) Enformatika. vol. 10, pp. 147-152.

[28] Saroj, S. and Bharadwaj, K.K. 2007. A parallel genetic algorithm approach for automated discovery of censored production rules. In proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications, Innsbruck, Austria: ACTA Press, pp. 435–441.

[29] Bharadwaj, K.K. and Saroj. 2009. Parallel genetic algorithm approach to automated discovery of hierarchical production rules. Applications of Soft Computing, J. Mehnen, M. Köppen, A. Saad, and A. Tiwari, (Eds.), Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 327-336.

[30] Bharadwaj, K.K. and Saroj. 2010. A parallel genetic programming based intelligent miner for discovery of censored production rules with fuzzy hierarchy. Expert Systems with Applications, vol. 37, pp. 4601-4610.

[31] Bharadwaj, K.K. and AlMaqaleh, B. M. 2007. Evolutionary approach for automated discovery of censored production rules with fuzzy hierarchy. International Multi-conference of Engineers and Computer Scientists, Hong Kong, pp. 716-721.

[32] Bharadwaj, K.K. and Jain, N. 1992. Hierarchical censored production rules (HCPRs) system. Data & Knowledge Engineering, vol. 8, pp. 19-34.

[33] Jain, N.K. and Bharadwaj, K.K. 1998. Some learning techniques in hierarchical censored production rules (HCPRs) system. International Journal of Intelligent Systems, vol. 13, pp. 319-344.

[34] Gains, B.R. and Compton, P. 1995. Induction of ripple down rules applied to modeling large database. Journal of Intelligent Information System, vol. 5, pp. 211- 228.

[35] Scheffer, T. 1996. Algebraic foundation and improved methods of induction of ripple down rules. Second Pacific Knowledge Acquisition Workshop, Sydney, pp. 279-292.

[36] Fadl, M., Ba-Alwi, and Bharadwaj, K.K. 2005. Automated discovery of hierarchical ripple down rules (HRDRs). In proceedings of the 23rd IASTED International Multi-Conference Artificial Intelligence and Applications, Innsbruck, Austria.

[37] Saroj, Kapila, Kumar, D. and Kanika. 2011. A genetic algorithm with entropy based probabilistic initialization and memory for automated rule mining. Advances in Computer Science and Information Technology, N. Meghanathan, B.K. Kaushik, and D. Nagamalai, (Eds.), Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 604-613.

[38] Yiyu, Yao and Bing, Zhou. 2008. Micro and macro evaluation of classification rules. In Seventh IEEE International Conference on Cognitive Informatics, ICCI 2008, Stanford University, California, USA, pp. 441-448.

Annexure - I

Procedure

```
Input  T  :  Table of examples ( training set T_R, test set T_S)
        GA parameters : No.of generations(ngen), population
        size(popsize), crossover and mutation rate(ncross, nmut),
        Coverage Threshold(ncov) etc.
Output RC: Set of rules (sorted on coverage)
         P_error : Error rate on test set T_S
Begin
      RC = ϕ
      Per1 = | T_R|   // No. of examples in training set
      Per2 = | T_S|   // No. of examples in test set
      Initialize population POP_1 biased towards more relevant
      attributes
      i = 1
      Do
            Evaluate(POP_1)
            POP_i+1 = Best of (POP_i)
            POP_i+1 = POP_i+1 + Replicate(POP_i)
            POP_i+1 = POP_i+1 + Recombine(POP_i)
            POP_i+1 = POP_i+1 + Mutate(POP_i)
            i = i + 1
     until(i <= ngen)
     POP_n=POP_i+1 // POP_n is the final population sorted on fitness
     Remove spurious or false attribute value conditions
     For(i=1;i<=popsize;i++)
        For(j=1;j<=per1;j++)
          Flag[j]=0            // Flag used to mark the tuple as covered
          If (POP_n[i] = T_R[j])
                    Flag[j]=1
                    Count=Count+1
          Endif
        Endfor
        Cov[i]=count/|per1|
      Endfor
      Sort array Cov
      For(i=1;i<=popsize;i++)
            r_i = POP_n[i]
            If (Cov[i]>=ncov)
    RC=RC+r_i
      Endfor
```