Evolution of Classification Rules for Comprehensible Knowledge Discovery

Emiliano Carreño, Guillermo Leguizamón, Neal Wagner *Member, IEEE*

***Abstract*— This article, which lies within the data mining framework, proposes a method to build classifiers based on the evolution of rules. The method, named REC (Rule Evolution for Classifiers), has three main features: it applies genetic programming to perform a search in the space of potential solutions; a procedure allows biasing the search towards regions of comprehensible hypothesis with high predictive quality and it includes a strategy for the selection of an optimum subset of rules (classifier) from the rules obtained as the result of the evolutionary process. A comparative study between this method and the rule induction algorithm C5.0 is carried out for two application problems (data sets). Experimental results show the advantages of using the method proposed.**

I. Introduction

The method proposed in this article uses genetic programming (GP) for the evolution of classification rules. The application of GP to the discovery of classification rules from a data set is not suitable when the size of trees (S-expressions) increases significantly. In such cases, the complexity of the model obtained makes it almost impossible to understand the underlying data generating process. Thus, if a model composed of many high complexity rules is obtained, it can be as hard to understand as a complex neural network. On the other hand, the measures of support and precision determine the predictive quality of a given hypothesis. Nevertheless, an appropriate model should provide an adequate balance between both parameters. For example, a rule with a 0.5 precision does not provide any information on whether an instance belongs or not to a given class, however, a rule with high precision and low support is not very useful either.

The approach proposed in this article aims to establish an appropriate balance between a rule's precision, support and complexity (directly related to comprehensibility) by incorporating an adaptive procedure which ranks individuals probabilistically based on calculated values for support, precision, and comprehensibility. This procedure allows biasing the search towards hypothesis regions with high comprehensibility and an appropriate balance between support and precision.

For each class of the target attribute, a rule set is obtained as result of an evolutionary process. Then, these sets are combined through a strategy to obtain the final model. The idea is to exploit the solutions generated during the evolutionary process by selecting an optimum rule set (as regards to predictive quality). In this sense, it may be the case that the optimum rule set may not be formed by the best solutions found during the evolutionary process, instead it may contain rules that complement each other adequately. The final result is a classifier expressed as a set of rules of the type **if-then**.

In the literature there are several studies where the process of knowledge discovery is focused on obtaining comprehensible and interesting rules with high predictive capacity. Some examples include [1], [2], [3], and [4]. In [2], classification rules are evolved using a Multi-objective Genetic Programming approach. In [4], an approach is presented to discover interesting prediction rules by applying a genetic algorithm in which the adaptive function (fitness function) is divided into two parts. One part measures the degree of interest of rules, while the other measures their predictive capacity. In [1], GP is proposed for the discovery of comprehensible rules, where a penalty for complexity is added in the adaptive function. In [3] this is also achieved by applying a genetic algorithm with a multi-objective approach. Other examples include [5], [6] and [7]. Here we propose a new approach to bias the search towards regions of comprehensible rules with high predictive quality, in several application problems. Moreover, a strategy to build classifiers by means of selecting a subset of the rules obtained as result of the evolutionary process, is introduced. This strategy is intended to obtain as a final model, an optimum subset of comprehensible rules with high predictive quality.

In this work a comparative study of the method proposed against C5.0 [8], a state-of-the-art rule induction algorithm for building classifiers, is carried out. This study analyzes mainly the predictive quality and the comprehensibility of models obtained with these two methods. Also, run time is reported. Data sets used come from the repository of the University of California at Irvine (UCI) [9].

The rest of this paper is organized as follows. Section II describes the use of GP for the discovery of classification rules. In section III the method proposed in this paper is presented and analyzed. Section IV shows experimental results obtained in the comparative study for two data sets. Finally, conclusions are given in Section V.

II. Rule Discovery Using GP

The main idea of GP is to evolve computer programs (S-expressions) which produce a solution for a particular problem, where candidate solutions are hierarchically structured

Emiliano Carreño and Guillermo Leguizamón are members of the Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC), Departamento de Informática, Universidad Nacional de San Luis, (D5700HHW) - San Luis - Argentina (phone: +54-2652-420823; fax: +54-2652-430224; email: {ecarreno, legui}@unsl.edu.ar).

Neal Wagner is member of the Department of Mathematics & Computer Science, Augusta State University, 2500 Walton Way Augusta, GA 30904 USA (phone: (706)667-4479; email: nwagner@aug.edu).

computer programs represented as trees. Once a function and terminal set are provided, the solution (model) is obtained by means of an evolutionary process. The function set (F) may contain arithmetic and logical operators, among other elements. The terminal set (T) contains the program's variables and the random ephemeral constant $\Re$, which represents random numbers within some range and decimal precision. It is required that $F \cup T$ be sufficient to express a program that can solve the problem under consideration. The fitness function measures the capability of individuals for solving the problem at hand. Several fitness measures may be adopted, some of which are: raw fitness, standardized fitness, normalized fitness, among others. These measures are explained in detail in [10].

After the initial population has been created, the algorithm is executed generation after generation until certain termination criterion has been met. Then, the best solution found is selected. For example, a termination criterion may state that a run must terminate when a pre-specified maximum number G of generations have been run, whereas a result designation criterion may be to choose the best individual in the population of the generation at termination time as the result of the whole run.

In each generation, each individual's fitness is evaluated, selecting probabilistically the best ones in the population based on some selection method (proportionate, tournament, rank-based selection, etc.), in order to apply reproduction, crossover, and mutation. Each operator is applied based on a certain probability. Reproduction is achieved by simply copying an individual from the current population into the next generation. In the crossover operation a crossover point is randomly chosen for each genetic tree. Then, both trees are split at these points creating four sub-trees that are combined to create new individuals. When the mutation operator takes place, a random point (node) is selected in a tree. The tree having as its root this node, is substituted by a sub-tree generated randomly at that point. For a more detailed description of the genetic programming paradigm refer to [10].

A. A Basic GP System for the Evolution of Rules

Next, the basic GP system for the evolution of rules used in this work is described. Rules considered here are of the type *IF* $\langle$*antecedent*$\rangle$ *THEN* $\langle$*consequent*$\rangle$. The antecedent part of a rule is formed by logical combinations of conditions on the values of predictive attributes using the logical connectors *AND*, *OR*, and *NOT*, whereas the consequent part indicates to which class a determined instance is assigned. However, each individual, represented as a tree, codes only the antecedent part of the rule. It is not necessary to code the consequent part since the genetic program is executed as many times as there are different classes. In each run a two class classification problem is solved and all rules evolved predict the same class.

The function set includes the logical operators *AND*, *OR*, and *NOT*, together with the equality operator which relates each attribute to some nominal value. A binning method is used in order to turn a numeric attribute into a nominal (categorical) one. This involves dividing the range of possible values into sub-ranges called bins. For example, the equal-width binning process simply divides the range of possible values into N sub-ranges of the same size. Here, the MDL (minimum description length) supervised method is used. When necessary, attributes are binned. The equality operator is applied over (binned) attributes during the evolution of rules. The terminal set is conformed by predictive attributes and the ephemeral constant $\Re$. Figure 1 shows an example of the codification of the antecedent of a rule. The equality operator takes an attribute as its first argument and a nominal value as its second argument. A logical operator may take as any of its arguments another logical operator or the equality operator. This structure is preserved through crossover and mutation operators.

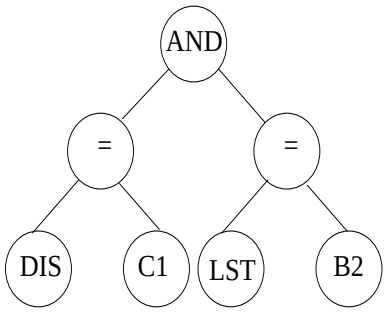


Fig. 1. Representation of an individual for rule discovery using GP, corresponding to the rule ***IF*** $(DIS = C1)$ ***AND*** $(LST = B2)$ ***THEN*** MEDV = High, where *C1* and *B2* represent the intervals $(2, 5]$ and $(6.5, 9]$, respectively.

We evaluate the quality of a rule by using an estimation value for precision and support. The precision value is calculated as the ratio between the number of instances to which the rule is applied and predicts correctly over the number of instances to which the rule is applied. Let *R1* be a rule that predicts class *c1*, *A* the set of instances which belong to class *c1*, and *B* the set of instances to which rule *R1* is applicable. Then, the precision of *R1* is given by equation 1.

$$Precision(R1) = \frac{|A \cap B|}{|B|} \tag{1}$$

That is to say, precision is the probability that the rule classifies correctly the instances to which it is applied. The support value is the ratio between the number of instances to which the rule is applied and predicts correctly over the total number of instances in the class corresponding to that rule. Support is calculated according to equation 2.

$$Support(R1) = \frac{|A \cap B|}{|A|} \tag{2}$$

That is to say, given an instance of the class *c1*, support is the probability rule *R1* has of being applicable to this instance. The fitness function can be established as an arithmetic equation including precision and support values. For example, we can use the measure F_β defined by equation 3, where β is a

parameter that controls the relative importance between both values, precision and support.

$$F_\beta = \frac{(1+\beta^2)\ support\ \times precision}{\beta^2\ \times precision\ +\ support} \quad (3)$$

So far, the basic GP system for rule discovery has been described. However, it must be taken into account that tree size could increase significantly. If we intend to obtain as a result, a set of comprehensible rules, some kind of mechanism to control the size of solutions is required. This can be done by adding a procedure to favor comprehensible rule discovery, as presented in the next section. The method proposed comprises a strategy for the selection of a subset of rules (classifier) from rules obtained as result of the evolutionary process.

III. The Proposed Method

The method proposed in this article has three main features:

i. It applies genetic programming to perform a search in the space of potential solutions.
ii. A procedure allows biasing the search towards regions of comprehensible hypothesis with high predictive quality. In each generation, the procedure ranks solutions in the current population in order to evaluate them (determine their fitness). Each individual is assigned to the most probable position in the ranking according to the binomial probability distribution, considering its values of support, precision, and complexity.
iii. It includes a strategy to produce an optimum subset of rules (classifier) from the rules obtained as result of the evolutionary process. The aim is to exploit those solutions generated during the evolutionary process by selecting an optimum set of rules.

Figure 2 illustrates the proposed method. The idea is to evolve classification rules, biasing the search towards comprehensible hypothesis regions with high predictive quality (features i and ii). Then, a selection strategy builds the rule set corresponding to the final model (classifier) using the best solutions generated during the evolutionary process (feature iii). For this work, we chose to run the GP system as many times as there are classes of the target attribute, obtaining for each one a set of rules formed by the best solutions found during evolution. Thus, for k classes, k rule sets are generated, each of them formed by rules which predict a specific class. A selection strategy builds the final model from these sets.

The GP system for the evolution of rules (feature i) was explained in section II. In the following subsections, the procedure for biasing the search and the selection strategy are explained (features ii and iii).

A. Biasing the Search

The proposal of this work includes the application of a stochastic and adaptive component which ranks solutions probabilistically considering the support, precision, and comprehensibility of individuals in the population (see Figure 2).

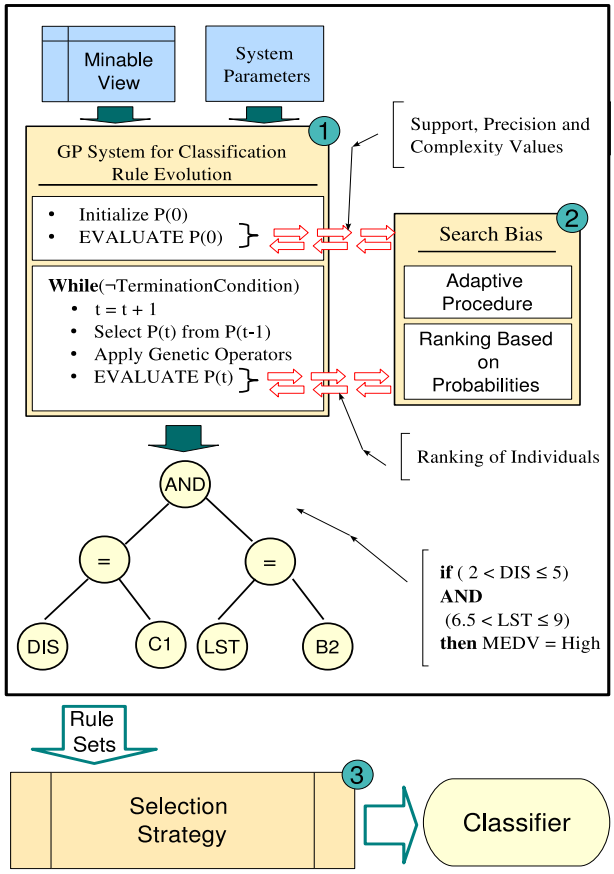


Fig. 2. The method proposed. 1) GP system for classification rule evolution. 2) Search bias procedure. 3) Selection strategy.

1) Description: The aim is to bias the search towards hypothesis regions with high comprehensibility and an appropriate balance between support and precision measures. Solutions in the population can be ranked by using a sorting algorithm (e.g., Hoare's quicksort) applying certain comparison criteria based on three probability values (see below) which are adaptively adjusted as a function of support, precision, and complexity values of individuals in the population of the current generation:

- *P_Sup*: is the probability of using the support factor to compare two solutions.
- *P_Conf*: is the probability of applying the precision (confidence) factor to perform the comparison.
- *P_Leng*: is the probability of applying the comprehensibility (complexity, length) factor when comparing two individuals.

where $P\_Sup + P\_Conf + P\_Leng = 1$.

After adjusting these probabilities by applying the adaptive procedure, solutions are ranked. Let ***rnd*** be a random number in the interval [0, 1]. The comparison between two solutions is carried out either according to support, or precision, or comprehensibility measures by considering the above probabilities in the following way:

i. If ***rnd*** $<$ *P_Sup* then the comparison is based on the support measure.
ii. If *P_Sup* $\leq$ ***rnd*** $<$ *P_Sup* + *P_Conf*, the comparison is based on the precision measure.
iii. Otherwise, the comparison is carried out according to the complexity measure.

According to the comparison criterion, the probability of a certain hypothesis (*X*) winning a comparison (against another hypothesis *Y*) in the sorting procedure is given by equation 4.

$$\begin{aligned} P(X \succ Y) \;=\; & P(\; X.Sup > Y.Sup \;) \cdot \; P\_Sup \\ & + P(\; X.Conf > Y.Conf \;) \cdot \; P\_Conf \\ & + P(\; X.Leng < Y.Leng \;) \cdot \; P\_Leng \end{aligned} \tag{4}$$

In the adaptive procedure, the values of *P_Sup*, *P_Conf*, and *P_Leng* are modified according to a parameter named **MaxLeng** and to population statistics. **MaxLeng** is a parameter defining the threshold from which the complexity of solutions starts influencing the fitness function. Population statistics correspond to the mean values of support (*Sup_Mean*), precision (*Conf_Mean*), and complexity (*Leng_Mean*) of the solutions in the current population or in a subset of it. This procedure consists in the following steps:

i. Set *P_Leng* according to equation 5. If *P_Leng* > 0.95, set *P_Leng* to 0.95.
ii. If $Sup\_Mean > Conf\_Mean$ add to *P_Conf* the amount given by equation 6, else add this amount to *P_Sup*. If the increased probability is greater than $1 - P\_Leng$, set it to $1 - P\_Leng$.
iii. Set the probability which wasn't modified in previous steps (*P_Conf* or *P_Sup*) to the value of the remaining probability (*P_Sup* + *P_Conf* + *P_Leng* $= 1$).

$$P\_Leng = \begin{cases} 0 & \text{if } Leng\_Mean \leq MaxLeng, \\ 1 - \frac{MaxLeng^2}{Leng\_Mean^2} & \text{otherwise.} \end{cases} \tag{5}$$

$$Increase = |Sup\_Mean - Conf\_Mean| \tag{6}$$

In step 1, if the reference parameter *Leng_Mean* exceeds the **MaxLeng** threshold, the value of *P_Leng* is established by using a quadratic function. Otherwise, the value of *P_Leng* is set to 0, stating that this probability will have no influence on comparing two solutions when performing the sorting process. An upper bound is set to the growth of *P_Leng* to avoid making all comparisons based on comprehensibility, hence allowing those solutions with high predictive accuracy and moderate complexity to obtain an adequate position in the ranking. Preliminary studies show that better results are obtained by bounding this probability. In this manner, the search is biased towards regions with the proper complexity (comprehensibility).

Next, the values of *P_Sup* and *P_Conf* are calculated by comparing the reference parameters of the population, *Sup_Mean* and *Conf_Mean*, in a way such that the probability associated with the smaller reference parameter value is increased by an amount proportional to the absolute value of the difference between the two reference parameters ($0 \leq Sup\_Mean \leq 1 \;\; and \;\; 0 \leq Conf\_Mean \leq 1$). This step insures that an appropriate balance between support and precision measures is achieved. To summarize, the value of *P_Leng* must be set first, then the remaining probability is distributed between *P_Sup* and *P_Conf*, as described above.

2) Analysis: Next we will perform an analysis in order to achieve a better understanding of the role that the probabilities play on ranking solutions. For this, we consider a hypothetical algorithm for ranking solutions. Given *P_Sup*, *P_Conf*, *P_Leng*, and the values of support, precision, and complexity of individuals in the population, we intend to determine the probability a hypothesis *X* has of being assigned to the i^{th} position in the ranking of solutions. Let *X* and *Y* be two hypothesis in the population selected at random. According to the comparison criterion, the probability of a hypothesis winning a comparison in the sorting procedure is given by equation 4.

In a population *Pop* of size *N*, the probability that hypothesis *X* wins a comparison according to some measure $m \in \{support, precision, complexity\}$ is given by the ratio between the number of instances in the population for which *X* is better relative to the *m* measure over $N - 1$, as it is shown in equation 7.

$$P(X.m > Y.m) = \frac{|\{Y' \in Pop : X.m > Y'.m\}|}{N - 1} \tag{7}$$

To simplify this analysis, we consider a ranking procedure (establishing a partial order) that, even if not efficient, is useful in order to carry out the analysis. This procedure partially sorts elements in an array *A* of size *N*, performing the following steps until a ranking position has been assigned to every element:

i. Choose an element *X* from *A* to which a ranking position has not yet been assigned and compare it (according to the criterion mentioned) against all other elements in *A*. Let *i* be the number of comparisons won (i.e., with a positive result).
ii. Assign *X* to the i^{th} position in the ranking, assuming that higher positions correspond to better solutions.

The first step of the algorithm above presents certain features which may lead to believe that it is a binomial experiment. However, the outcomes of trials (comparisons) are dependent events, since the probability *P* of success in equation 4 may vary from one trial to another. In preliminary experiments, on running step 1 of the algorithm several times (for an element *X*) and counting the amount of comparisons won, it was observed that the data distribution corresponding to the number of comparisons won has approximately the shape of a binomial distribution. Then experiments were performed (for several populations and different values of *P_Sup*, *P_Conf*, and *P_Leng* generated randomly) in order to check whether such data distribution could be approximated

by a binomial probability distribution. According to results of experiments carried out applying the Chi-square goodness-of-fit test (with a significance level α of 0.05), we concluded that the probability of an element *X* being successful in *i* comparisons can be approximated by the binomial probability distribution[1]. The formula for the binomial probability distribution is shown in equation 8.

$$P(i) = \frac{n!}{i!(n-i)!} \cdot p^i q^{(n-i)} \tag{8}$$

where p is given by equation 4 and $q = 1 - p$.

In this partial sorting algorithm, the position of an element *X* in the ranking is determined by the number of comparisons won. Then, the probability an hypothesis *X* has of being assigned to the i^{th} position in the ranking of solutions can be approximated by the binomial probability distribution (eq. 8). Finally, given an element *X*, the ranking position to which it will more probably be assigned is given by the expected value of the binomial distribution $\mu = p \cdot n$ where $n = N-1$.

3) Ranking Based on the Binomial Probability Distribution: From the previous analysis it is possible to derive a procedure for ranking solutions. This procedure assigns to each element *X* a position calculated according to the expected value $\mu = p \cdot (N-1)$, where p is given by equation 4 and *N* is the population size.

Let *Y* be an individual in the population selected at random. The position of each individual *X* in the ranking is obtained by performing the following steps:

i. Calculate $P(X.Sup > Y.Sup)$, $P(X.Conf > Y.Conf)$ and $P(X.Leng < Y.Leng)$ according to equation 7.
ii. Calculate $P(X \succ Y)$ according to equation 4.
iii. Assign *X* to the most probable position according to: $Pos(X) = P(X \succ Y) \cdot (N-1)$.

Results reported in this article are obtained using the ranking based on the binomial distribution, as a means to bias the search towards regions of comprehensible rules with high predictive quality.

B. Selection Strategy and Classifier Construction

As mentioned earlier, the selection strategy builds a classifier from rules obtained from an evolutionary process. In this work, a GP system is run as many times as there are classes, obtaining a rule set for each class. Each set will be composed of the best rules generated during the evolutionary process. Then the selection strategy builds the final model (classifier) from these sets. It begins by forming a set choosing the best rule in each set corresponding to a particular class. Then, the remaining rules are progressively evaluated in order to determine whether they will be included in the set. On evaluating the inclusion of a rule, the strategy considers if the predictive quality of the rule set is improved. The maximum size of the rule set comprising the final model must be specified, so as to bound its complexity. This is why we must consider whether or not the inclusion of a new rule will cause the size of the set to exceed the maximum. If the maximum will not be exceeded, it is evaluated whether the rule improves the predictive quality of the rule set. If it decreases the error percentage, it is added to the set; otherwise it is not. If the maximum will be exceeded, the idea is to evaluate all possible sets that may be formed on replacing a rule in the set with the new rule, and then select the set with lower percentage of error. Let $A_1 \ldots A_n$ be rule sets for classes $C_1 \ldots C_n$ respectively, and C an initially empty set. The algorithm builds the final model from these sets as follows:

i. Add to C the best hypothesis in each set $A_1 \ldots A_n$.
ii. While not every rule in $\bigcup_{i=1}^{n} A_i$ has been considered do:
    a. Select a rule in $\bigcup_{i=1}^{n} A_i$ not yet considered for evaluation.
    b. Evaluate its inclusion in C.
iii. Return C as the result.

The final model will be formed by a subset of rules in $\bigcup_{i=1}^{n} A_i$ properly combined and not necessarily by those with the highest predictive quality (in $\bigcup_{i=1}^{n} A_i$). The above strategy has been used with good results for experiments reported in section IV.

IV. Experimental Results

This section presents a comparative study of the method proposed, named **REC** (Rule Evolution for Classifiers), and the algorithm **C5.0**. Also, a study of the influence of the **MaxLeng** parameter over predictive quality and complexity of rules obtained is carried out.

The application problems (data sets) considered in the experiments are Breast Cancer (BC), German Credit (GC), and Boston Housing (BH). Instances in BC data set consist of visually assessed nuclear features of fine needle aspirates (FNAs) taken from patients' breasts. The classification problem is to determine whether a sample is benign or malignant. Each instance is composed of a sample code number, 9 predictive attributes (taking values in the interval 1 to 10) and the target attribute. There are 699 instances. This breast cancer database comes from the University of Wisconsin Hospitals, Madison (collected by Dr. William H. Wolberg) [11]. The German Credit data set classifies people described by a group of attributes as good or bad credit risks. There are 20 predictive attributes (7 numerical and 13 categorical) and 1000 instances. Some predictive attributes are credit amount, purpose, etc. Boston Housing classification problem consists in predicting the housing value (numerical value binned into three intervals) from information reflecting housing conditions (amount of rooms, crime rate, etc.). The data set has 13 continuous attributes (including the "class" attribute "MEDV"), 1 binary-valued attribute and 506 instances. Instances reflect housing conditions in the suburbs of the City of Boston. Some attributes are CRIM (crime rate), DIS (weighted distances to five Boston employment centres), etc. For a detailed description of these data sets refer to [9].

Comprehensibility is of utter importance within the data mining context. Therefore, the main point of this work is

[1]Goodness-of-fit test details are too lengthy to be given here, but can be provided on request.

to obtain rules that are comprehensible to the user. In this article we use structural complexity (length) to approximate rule complexity. This is done by counting the number of logic connectives (in $\{$*OR*, *AND*, *NOT* $\}$), attributes (variables), and terminals (nominal or numerical values) in the antecedent part of the rule.

A classifier is represented as a set of **if-then** rules. The next subsection explains the behavior of classifiers on classifying instances and the measures used to evaluate them. Then, in subsection IV-B, the main results are presented.

A. Evaluation and Use of Classifiers

1) Evaluation Measures of Rule Sets: Classifiers are evaluated by two performance evaluation measures. When all classification errors have the same cost, the measure of evaluation used is the error rate, calculated as the ratio between the number of misclassified instances over the total number of instances used in evaluation (*%_Error* $=$ *#Errors* / *#Instances*). Here, the misclassification cost is of 1. When the cost associated with a classification error depends on the class the model predicts and the true class of the misclassified instance, the measure used is the average misclassification cost. For this, first the total cost is calculated by adding up the misclassification costs associated to each instance (using the confusion matrix corresponding to the application problem). Then, the average cost is calculated as the ratio between the total cost over the number of instances used in evaluation (*Average_Cost*= *Total_Cost* / *#Instances*). In both cases, if the predicted class is correct, its cost is taken to be 0.

2) Handling Instances with Unknown Attribute Values: To an instance with unknown value of the *X* attribute, it is assigned the average value (if numeric) or the most frequent value (if nominal) of *X* in the training set. In this way, instances in training and test sets are completed.

3) Instance Classification: On rule set evaluation, an instance is classified according to the following cases:

- There is at least one rule applicable to the instance. Among applicable rules, it is chosen the one with higher precision value and the instance is assigned to the class this rule predicts.
- None of the rules apply to the instance. In this case the instance is assigned to the class corresponding to the rule with the **least** value of support.

This choice is reasonable from the viewpoint of probabilities. According to equation 1, precision is the probability that the rule classifies correctly the instances to which it is applied. Then, if at least one rule is applicable to the instance, the rule chosen according to this measure is the one with the highest probability of correctly classifying this instance (from all applicable rules).

On the other hand, according to equation 2, given an instance of the class *c1*, support is the probability rule *R1* has of being applicable to this instance. Then $1 - support$ is the probability rule *R1* has of **not** being applicable to this instance. Thus, $1 - support$ is the probability of *R1* misclassifying this instance (assuming that, once a rule that correctly describes a class *c1* is obtained, instances to which this rule is not applicable do not belong to this class). Then, if none of the rules apply, the instance is assigned to the class corresponding to the rule with the highest probability of misclassifying this instance.

B. Main Results

In this work, a comparative study of the method proposed, named **REC** (Rule Evolution for Classifiers), and the algorithm **C5.0**, is carried out. The data sets used in these experiments are Breast Cancer (BC) and German Credit (GC). In all experiments, the selection method based on linear ranking proposed by Baker [12] is applied. Individuals with the highest values of F_β (see equation 3) in any generation are designated as the result of a GP system run. These individuals form the rule set corresponding to a class.

REC system parameter values for Breast Cancer (BC) and German Credit (GC) problems were set as follows: population size, 152; number of generations, 90; crossover probability, 0.95; mutation probability, 0.2; MaxLeng, 5; maximum number of rules obtained for each class, 40; and maximum complexity allowed for rules in the final model, 33. The β parameter was set to 0.7 for BC and to 0.2 for GC. The maximum number of rules forming the classifier was of 9 for BC and 4 for GC. These values were adjusted through several runs.

Parameters for **C5.0**, were set according to its authors´ recommendations in [8], as follows: For both problems, Global pruning, Pruning CF, and Minimum cases were left to default values, Winnow attributes option was not used, and Boost option was set to 1 and 10. Subsets of values option was used only for BC problem, in order to obtain better results. For a detailed description of these parameters, refer to [8]. In the appendix, we give an example of a classifier obtained with **REC** *system* for the German Credit data set.

Table I shows the best results obtained with both methods. According to statistical tests carried out with a sample size $N = 40$ and a significance level $\alpha = 0.05$, the results presented in Table I are statistically significant. Column "COM" reports the complexity of the classifier, and column "TIME" reports run time (seconds). Complexity is calculated by multiplying the amount of rules in the classifier by the average length of rules. In GC problem the cost associated with a classification error depends on the class the model predicts and the true class of the misclassified instance, on the other hand in BC all classification errors have the same cost. Thus, column "E-AC" corresponds to the error rate for BC and to the average cost for GC problem. For **C5.0** which uses a boosting technique, results are shown separated by a vertical slash ("|") and correspond to 1 and 10 trials, respectively. The authors state that, trials over numerous data sets (large and small) show that on average 10-classifier boosting reduces the error rate for test cases by about 25 $\%$ [8].

It can be seen from the "E-AC" column that, for both problems, the classifiers generated with the proposed method (**REC**) have a better predictive quality than those obtained

TABLE I
RESULTS OBTAINED WITH **REC** AND **C5.0** FOR BREAST CANCER (BC) AND GERMAN CREDIT (GC) APPLICATION PROBLEMS

DataSet/ Method	REC			C5.0		
	E-AC	COM	TIME	E-AC	COM	TIME
BC	2.95	121.9	14.02	4.9 | 4.2	14.7 | 411.4	0.0 | 0.1
GC	0.48	17.2	17.7	0.66 | 0.57	26.2 | 530.8	0.1 | 0.4

with **C5.0**. Even though the predictive quality of **C5.0** classifiers improves with 10 trials, it still remains under that of **REC** system classifiers.

Regarding model comprehensibility, both approaches obtain classifiers with a complexity such that it allows for an adequate understanding. However, on running **C5.0** with 10 trials, the complexity of the obtained models increases notably (making them very difficult to understand), without excelling **REC** system's classifiers predictive quality.

On the other hand, **C5.0** takes much less time to build the classifier than the proposed approach. This was to be expected, given that **REC** applies an evolutionary algorithm to carry out the search in the space of potential solutions. Nevertheless, **REC** makes up for this disadvantage by having a better predictive quality.

It is worth to emphasize that with **REC** system it was only necessary to modify two parameter values in order to obtain good solutions for both problems (BC & GC).

C. Analysis of the MaxLeng Parameter

In this subsection we study the influence of the **MaxLeng** parameter over predictive quality and complexity of rules obtained, and its influence on CPU time required to obtain them.

The data set used in these experiments is the BostonHousing. Attribute selection, together with the binning process and selection of the training and test set, are performed with the data mining tool WEKA (Waikato Environment for Knowledge Analysis) [13]. Attributes are binned into intervals of equal length allowing the binning method to find the optimum amount of bins, except for the objective attribute, which is binned arbitrarily into three intervals of equal length, representing high, medium, and low housing prices. The Boston Housing data set is divided as follows: 66% of the data are chosen randomly for training, the remaining data conform the testing set.

We measure predictive quality by using the F_β measure defined by equation 3. In this case, β was set to 1, giving the same importance to support and precision measures. On Figure 3 we can observe that the predictive quality does not vary significantly for different values of the **MaxLeng** parameter. On the other hand, there is a significant difference between classes, showing that some classes are more difficult to predict than others.

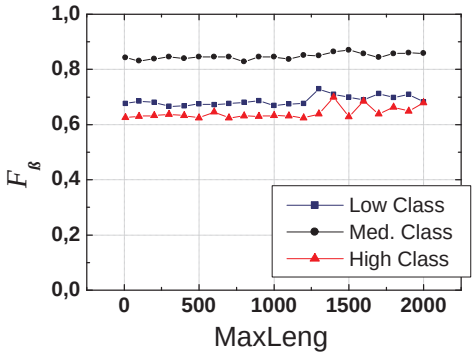


Fig. 3. Influence of **MaxLeng** on predictive quality.

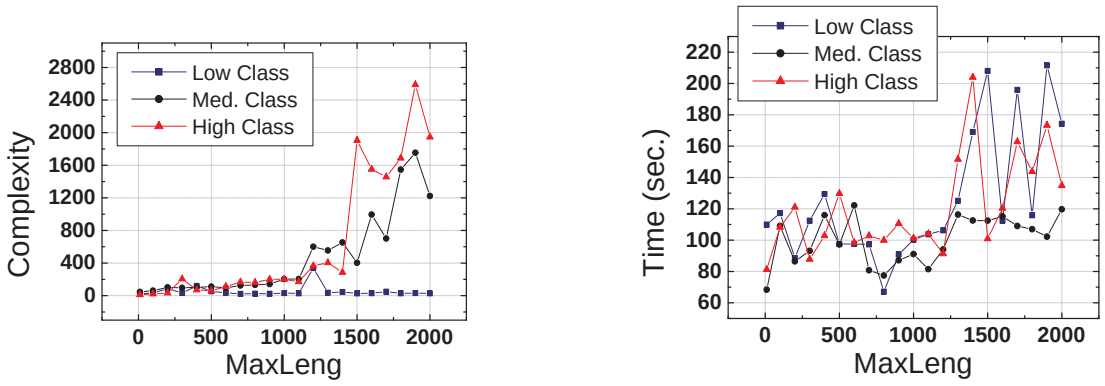
Figure 4 shows results for complexity and CPU time. With respect to **Medium** and **High** classes, it can be seen (left side of Figure 4) a clear increment on the complexity of the model obtained on incrementing the values of **MaxLeng**. However, for **Low** class, the variation in the complexity of solutions obtained is not meaningful. This is due to the low complexity solutions found in earlier generations exceeding in predictive quality the more complex solutions found in later generations of the evolutionary process. However, in all cases, average population complexity increments. Thus, this parameter biases the search towards regions where there are hypothesis with a certain complexity. On the right (Figure 4) it can be observed that CPU time tends to increase on incrementing **MaxLeng** values. This last result is a consequence of the increment in the average structural complexity of the whole population.

V. CONCLUSIONS AND FUTURE WORK

In this article, a method to build classifiers based on rule evolution within the data mining framework has been proposed. The method has three components: an evolutionary algorithm to perform a search in the space of hypothesis, a procedure that biases the search toward space regions of comprehensible hypothesis with high predictive quality, and a strategy for building an optimum rule set from solutions obtained during the evolutionary process.

In this work we apply GP to carry out a search for solutions, given that it is suitable to evolve rules of variable length. The bias procedure intends to balance support, precision, and complexity measures in the evolution of classification rules. The aim is to direct the search towards regions with the desired characteristics, i.e., comprehensible hypothesis with a high predictive accuracy. The final model is built by combining properly the rules evolved.

According to the results presented in section IV, it can be concluded that the proposed approach is capable of focusing



(a) Complexity Vs. **MaxLeng**.

(b) Time Vs. **MaxLeng**

Fig. 4. Influence of **MaxLeng** on solution complexity and CPU time.

the search on regions where hypothesis have a structural complexity such that allows for an appropriate understanding. Improvements achieved with respect to **C5.0** are significant. Results of the comparative study show the advantages of using the method proposed (**REC**), given that classifiers obtained have better predictive quality than those obtained using **C5.0**. For all problems, when **C5.0** is run with 1 trial the classifiers obtained with both methods are of similar structural complexity. On the other hand, on running **C5.0** with 10 trials (as recommended by its authors) the predictive quality of the models improves, but their complexity is noticeably increased and still their predictive quality remains below the predictive quality of models generated with **REC** system.

Although run time taken for the evolution of rules is reasonable, it could be reduced by parallel approaches. For example, there could be several processors, each evolving rules of a different class.

An additional advantage of **REC** system is that it is possible to set the complexity and structure of the classifier to be built. This can be done by parameters determining maximum rule complexity and maximum number of rules in the model. Finally, results demonstrate the high performance of the method proposed to build comprehensible classifiers with high predictive quality.

APPENDIX

CLASSIFIERS GENERATED WITH **REC** SYSTEM

This appendix shows a classifier generated for the German Credit problem approached in section IV. In this problem $Attribute\_1$ represents the status of existing checking account; $Attribute\_14$, other installment plans; $Attribute\_4$, the purpose; and $Attribute\_2$, duration in months. The rule set is:

Rule 1. **IF** ($Attribute\_1 = A14$) **AND**
($Attribute\_14 = A143$) **THEN** $Good$

Rule 2. **IF** $Attribute\_4 = A40$ **THEN** Bad

Rule 3. **IF** $43.2 < Attribute\_2 \leq 48.8$ **THEN** Bad

Rule 4. **IF** $Attribute\_1 = A11$ **THEN** Bad

REFERENCES

[1] C. C. Bojarczuk, H. S. Lopes, and A. A. Freitas, "Genetic programming for knowledge discovery in chest-pain diagnosis," *IEEE Engineering in Medicine and Biology Magazine*, vol. 19, no. 4, pp. 38–44, Jul.-Aug. 2000.

[2] A. Reynolds and B. de la Iglesia, *Rule Induction for Classification Using Multi-Objective Genetic Programming*, ser. In Evolutionary Multi-Criterion Optimization 4th International Conference (EMO 2007), 2007, vol. LNCS 4403, pp. 516–530.

[3] K. Deb and D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[4] E. Noda, A. A. Freitas, and H. S. Lopes, "Discovering interesting prediction rules with a genetic algorithm," in *Proceedings of the Congress on Evolutionary Computation*, vol. 2. IEEE Press, 6-9 1999, pp. 1322–1329.

[5] K. C. Tan, Q. Yu, C. M. Heng, and T. H. Lee, "Evolutionary computing for knowledge discovery in medical diagnosis," *Artificial Intelligence in Medicine*, vol. 27, no. 2, pp. 129–154, 2003.

[6] J.-H. Hong and S. B. Cho, "Lymphoma cancer classification using genetic programming with SNR features," in *Genetic Programming 7th European Conference, EuroGP 2004, Proceedings*, ser. LNCS, vol. 3003. Coimbra, Portugal: Springer-Verlag, 5-7 Apr. 2004, pp. 78–88.

[7] G. L. Pappa and A. A. Freitas, "Towards a genetic programming algorithm for automatically evolving rule induction algorithms," in *ECML/PKDD 2004 Proceedings of the Workshop W8 on Advances in Inductive Learning*, Pisa, Italy, 20-24 Sep. 2004, pp. 93–108.

[8] R. Quinlan, "Rulequest research data mining tools." 2006. [Online]. Available: http://www.rulequest.com/

[9] C. B. D.J. Newman, S. Hettich and C. Merz, "UCI repository of machine learning databases." 1998. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[10] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[11] O. L. Mangarasian, R. Setiono, and W. H. Wolberg, "Pattern-recognition via linear-programming: theory and application to medical diagnosis," in *Large-Scale Numerical Optimization, 1990*, T. F. Coleman and Y. Li, Eds. Philadelphia: SIAM, 1990, pp. 22–31.

[12] J. E. B. J, "Adaptive selection methods for genetic algorithms," in *Proc. ICGA 1*, 1985, pp. 101–111.

[13] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.