ARTÍCULO

Achieving an Appropriate Balance between Precision, Support, and Comprehensibility in the Evolution of Classification Rules

Emiliano Carreño, Guillermo Leguizamón

Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)
Departamento de Informática
Universidad Nacional de San Luis
Ejército de Los Andes 950 - Local 106
(D5700HHW) - San Luis - Argentina
Tel: +54-2652-420823 / Fax: +54-2652-430224
{ecarreno, legui}@unsl.edu.ar

Abstract

This article proposes a method for achieving an appropriate balance between the parameters of support, precision, and complexity during the evolution of classification rules by means of genetic programming. The method includes an adaptive procedure in order to achieve such balance. This work lies within the data mining context, more precisely, it focuses on the extraction of comprehensible knowledge where the approach introduced plays a predominant role. Experimental results demonstrate the advantages of using the proposed method.

Key words: Data Mining, Classification Rules, Genetic Programming, Ranking, Comprehensible Knowledge.

1 Introduction

The method proposed in this work 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.

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, 6]. In [6], 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 [2] this is also achieved by applying a genetic algorithm with a multi-objective approach.

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.

The classification problem approached in this article consists in predicting the housing price (numerical value binned in three intervals) from information about its zone location (crime rate, etc.). The data set used, called *Boston Housing*, comes from the repository of the University of California at Irvine (UCI) [3]. It has 13 continuous attributes (including the "class" attribute "MEDV"), 1 binary-valued attribute and 506 instances. Instances reflect housing conditions in the suburbs of Boston.

The rest of this article is organized as follows. Section 2 describes the use of genetic programming for the discovery of classification rules (the basic GP system for the evolution of rules). In section 3 the approach proposed in this paper is presented and analyzed. In the same section a basic GP system, tailored with the proposed approach in order to discover comprehensible knowledge (rules), is described. Section 4 shows experimental results obtained for the Boston Housing data set. Finally, conclusions are given in section 5.

2 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 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 [5].

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 chose 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 [5].

2.1 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. 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.

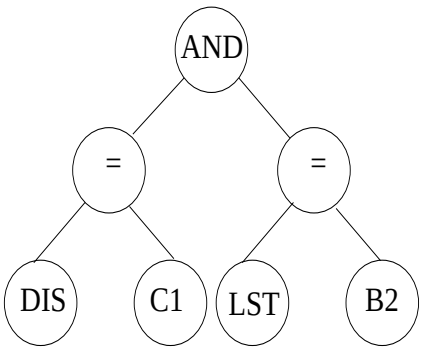


Figure 1: Representation of an individual for rule discovery using GP, corresponding to the rule *IF* $(DIS = C1)$ *AND* $(LST = B2)$ *THEN* MEDV = High, where *C1* and *B2* represent the intervals $(2, 5]$ and $(6.5, 9]$, respectively.

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.

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

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.

3 The Proposed Approach

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).

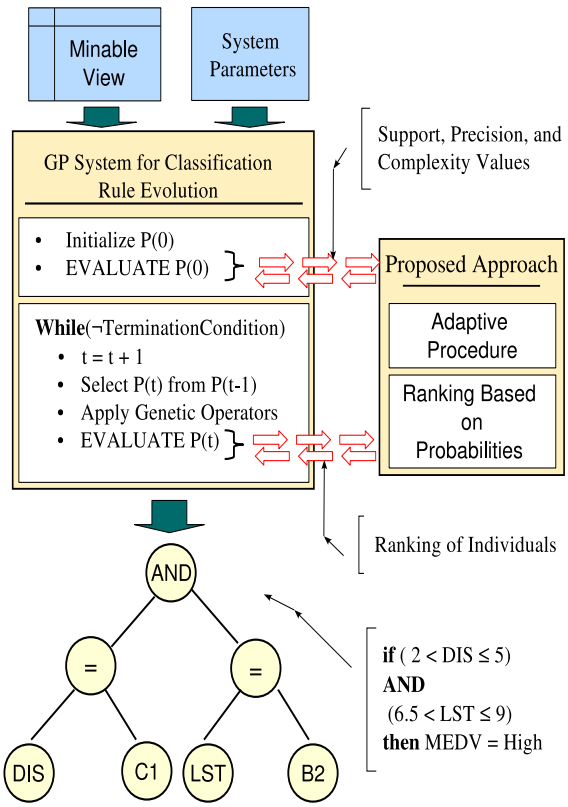


Figure 2: A GP system including the proposed approach for rule evolution

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 measure, by considering the above probabilities in the following way:

i. If $\boldsymbol{rnd} < P\_Sup$ then the comparison is based on the support measure.

ii. If $P\_Sup \leq \boldsymbol{rnd} < P\_Sup + P\_Conf$, 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.

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

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 ($P\_Sup + P\_Conf + P\_Leng = 1$).

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

$$Increase = |Sup\_Mean - Conf\_Mean| \quad (6)$$

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 ($0 \leq Sup\_Mean \leq 1 \quad and \quad 0 \leq Conf\_Mean \leq 1$). 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, so as to then distribute the remaining probability between *P_Sup* and *P_Conf*, as described above.

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.

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

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

Table 1: Results for approaches GP_B and GP_{AR} (MaxLeng = 75 and $\beta = 0.8$)

Appro.	Class	Support	St.Dev.	Precision	St.Dev.	Length	St.Dev.	Time	St.Dev.
GP_B	Low	0.6411	0.033	0.7453	0.019	2567.33	112.61	237.88	22.13
	Med.	0.9112	0.024	0.7842	0.0254	9011.82	500.45	395.04	53.17
	High	0.8309	0.047	0.4755	0.0435	2924.19	201.15	310.29	20.34
GP_{AR}	Low	0.6440	0.012	0.7545	0.027	55.247	6.771	110.13	11.22
	Med.	0.7605	0.045	0.8701	0.031	42.931	8.486	75.72	9.46
	High	0.8361	0.028	0.5539	0.024	166.468	26.535	133.77	10.15

the probability of an element X being successful in i comparisons can be approximated by the binomial probability distribution. The appendix gives more details on this test. 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 a 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 $n = N - 1$.

4 Experimental Results

This section presents preliminary results obtained with the proposed approach aiming at:

1. Making a comparative analysis against the basic GP system for the evolution of classification rules explained in section 2. As stated in section 3, the approach proposed in this article incorporates to a basic GP system a procedure which ranks population individuals for assesing the respective quality.

2. Studying the influence of the **MaxLeng** parameter regarding predictive quality and comprehensibility of the models obtained. As mentioned in section 3, this parameter is the threshold from which the comprehensibility of solutions starts influencing their fitness.

In all experiments, the selection method based on linear ranking proposed by Baker [4] is applied. The individual with the highest value of F_β (see equation 3) in any generation is designated as the result of the whole run (result designation criterion). Runs are carried out with a population size of 300 individuals through 500 generations. Crossover probability is set to 0.95 and the mutation operator is applied to 1 out of 5 individuals which are the result of applying the reproduction and crossover operators. Statistical data is obtained by performing 30 independent runs.

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

4.1 Application Problem

The classification problem chosen for our experimental study 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 comes from the UCI repository (Housing Database) and 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 (distance to the five working centers in Boston), etc.

Attribute selection, together with the binning process and selection of the training and test sets, are performed with the data mining tool WEKA (Waikato Environment for Knowledge Analysis) [7]. 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. Table 2 presents information regarding each objective attribute bin. 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.

Table 2: Information regarding each objective attribute bin

Class	Lim.Inf	Lim.Sup.	#Inst.Training
Low	-INF	16.666	39
Medium	16.666	33.333	118
High	33.333	INF	17

4.2 Comparative Analysis

In this subsection, a comparative study between a basic system for the evolution of classification rules, named as system GP_B, and the system incorporating the adaptive and ranking procedure proposed in section 3, named as system GP_{AR} , is presented. System GP_B uses F_β as the fitness function, whereas GP_{AR} incorporates a sorting algorithm based on probabilities to rank solutions.

In preliminary experiments, best results for GP_B were obtained by setting $\beta = 0.8$. On the other hand, for GP_{AR} the best ones were achieved by assigning the same value to β with **MaxLeng**= 75. Therefore, these values for the β and **MaxLeng** parameters were used in order to carry out a comparative study in the final experiments reported here. The results are shown in table 1, where it can be observed that system GP_{AR} achieves an important reduction in the complexity of the rules obtained for each class, thus improving their comprehensibility. This improvement is obtained without compromising the predictive quality of the model. Moreover, a slight improvement in support and precision values can be noticed regarding system GP_B (for certain classes).

Also, the reduction in CPU time taken for the evolution of rules is noticeable. It takes about one half of the time for **Low** and **High** classes, while for **Medium** class, it takes approximately 5 times less. Solution evaluation is the most time consuming task in the evolutionary process, so this improvement in CPU time is a direct consequence of the reduction of hypothesis complexity. Therefore, the overhead introduced by the proposed approach has been overcome by the improvement in run time.

An example of a result obtained by using $MaxLeng = 5$ and $\beta = 0.8$ for the **Medium** class is:

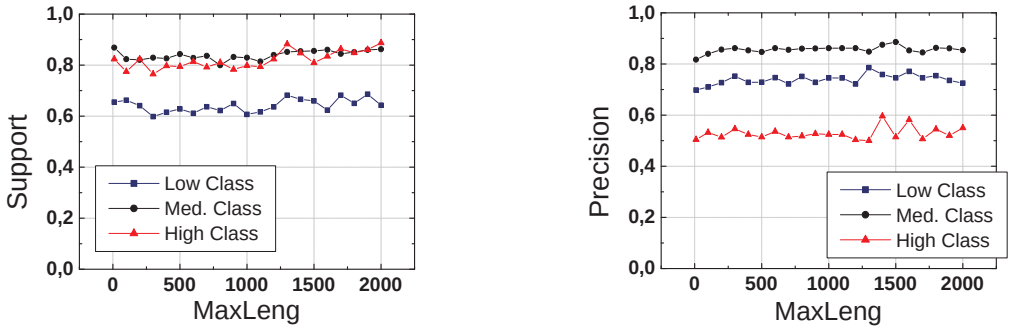
IF $NOT\ (DIS = H1)$
THEN $Class =$ **Medium**

Where *H1* represents the interval $(-INF, 2.02]$. The above solution has a support value of 0.89 and a precision value of 0.75, whereas its complexity is 3. Regarding the results shown in table 1, complexity has been reduced noticeably as the value of support increased, however, there is also an important decrease in the precision measure. On evolving classification rules we must recall that best solutions as regards to predictive quality may be found in other regions of the search space than those where hypothesis have the desired comprehensibility. This is why it is necessary to reach a consensus between comprehensibility on the one hand, and predictive quality on the other. Thus, the value of the **MaxLeng** parameter must be tuned in order to achieve such balance in the obtained model.

4.3 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.

Figure 3 shows results for support and precision measures. On the left side of Figure 3 we can observe that neither support nor precision values vary significantly for different values of the **MaxLeng** parameter. In both cases, there is a significant difference between classes, showing that some classes are more difficult to predict than others.



(a) Support Vs. **MaxLeng**.

(b) Precision Vs. **MaxLeng**

Figure 3: Influence of MaxLeng on support and precision values.

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.

5 Conclusions and Future Work

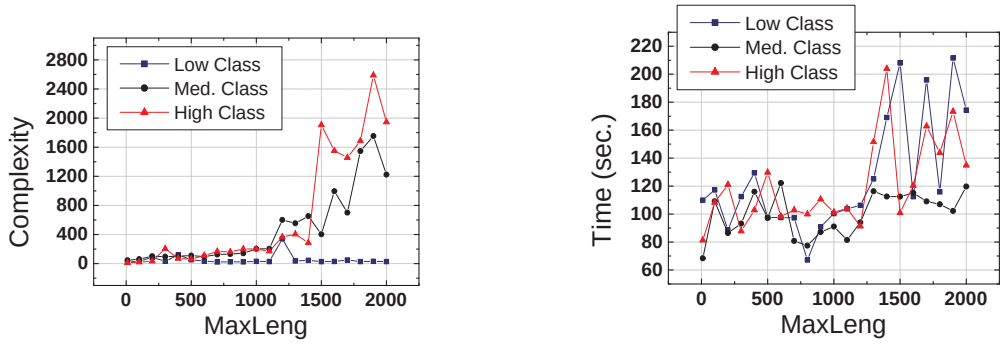
The proposed method intends to balance support, precision and comprehensibility measures in the evolution of classification rules. The aim is to direct the search towards regions with the desired characteristics, i.e., comprehensible hypothesis with a high predictive accuracy.

According to the results presented in section 4, it can be concluded that the proposed approach is capable of focusing the search on regions where hypothesis have a structural complexity such that allows for an appropriate understanding. Improvements achieved with respect to GP that does not apply this approach are significant.

Regarding support and precision values, the global quality of the results obtained is equal to or better than that obtained without applying the proposed approach; i.e., the quality of the solutions is not affected by the decreased structural complexity (which, in addition, increases understanding). A good balance between support and precision measures was also achieved. Execution time for the evolution of rules decreases considerably when applying GP_{AR}. Although run time taken for the evolution of rules is reasonable, it could be reduced by parallel approaches.

Even if it is possible to evaluate separately the quality of rules by using support and precision measures, it would be advantageous to evaluate the predictive quality of the set of rules as a whole (the set conformed by rules from all classes). Also, it would be convenient to make a comparative analysis against a different algorithm for rule discovery (e.g., **C5.0**). Additionally, it would be possible to evolve more than one rule for each class. However, if a large number of rules from each class are evolved, the complexity of the model will increase significantly.

In this sense, we are working on a method to build classifiers based on the evolution of rules. The method has three main features: it applies genetic programming to perform a search in the space of potential solutions, it uses the procedure proposed in section 3 in order to bias the search



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

(b) Time Vs. **MaxLeng**

Figure 4: Influence of MaxLeng on solution complexity and CPU time.

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 being carried out for several application problems (data sets).

Finally, all the results presented suggest directions for future work from theoretical and experimental perspectives, which can lead to improvements of models both in their understanding and predictive quality.

Appendix: Goodness-of-Fit Test

This appendix gives more details of experiments carried out applying the Chi-square goodness-of-fit test (with a significance level α of 0.05) in order to check whether the probability of an element X being successful in i comparisons can be approximated by the binomial probability distribution. The Chi-square test is used to test if a sample of data came from a population with a specific distribution.

Consider the following ranking procedure, proposed in subsection 3:

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.

As stated, 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. Experiments were carried out according to the following steps:

i. Generate randomly a population and values for *P_Sup*, *P_Conf*, and *P_Leng*.

ii. Randomly choose an element X from the population and repeat 250 times:

   a. Compare it (according to the comparison criterion) against all other elements .

   b. Count the number of comparison won.

iii. Apply the Chi-square goodness-of-fit test (with a significance level α of 0.05) to the data set generated in ii (in order to check whether such data distribution can be approximated by a binomial probability distribution).

This experiment was performed 1000 times, each time selecting randomly a point in an $Pop\_Size * 3 + 3$ dimensional space. The term $Pop\_Size * 3$ corresponds to individuals in the population, represented by their support, precision, and complexity measures and the second term corresponds to *P_Sup*, *P_Conf*, and *P_Leng*. For each point, the hypothesis that the data distribution

corresponding to the number of comparisons won can be approximated by the binomial distribution could not be rejected.

References

[1] Celia C. Bojarczuk, Heitor S. Lopes, and Alex A. Freitas. Genetic programming for knowledge discovery in chest-pain diagnosis. *IEEE Engineering in Medicine and Biology Magazine*, 19(4):38–44, July-August 2000.

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

[3] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[4] J. E. Baker J. Adaptive selection methods for genetic algorithms. In *Proc. ICGA 1*, pages 101–111, 1985.

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

[6] Edgar Noda, Alex A. Freitas, and Heitor S. Lopes. Discovering interesting prediction rules with a genetic algorithm. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1322–1329, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press.

[7] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, second edition, 2005.