

CGO3: AN OBLIQUE CLASSIFICATION SYSTEM USING AN EVOLUTIONARY ALGORITHM AND C4.5

José L. Álvarez, Jacinto Mata and José C. Riquelme*

Dpt. Ing. Electrónica, Sist. Informáticos y Automática
E.P.S. de La Rábida. Universidad de Huelva

*Dpt. Lenguajes y Sistemas Informáticos

Facultad de Informática. Universidad de Sevilla

e-mail: {alvarez, mata}@diesia.uhu.es, riquelme@lsi.us.es

Recommended by: A. P. Engelbrecht

Received: May 2001

Accepted in the final form: October 2001

Abstract. This paper presents a new oblique hyperplane classification system based on an Evolutionary Algorithm and C4.5, namely CGO3. The main objective of this work is to solve the problems that C4.5 has on databases that are not adjusted parallel classification models. Thus, the purpose of CGO3 is to improve the results of C4.5, but offering a classification model with the same format. This improvement consists on reducing the number of rules and the error percentage. To reach our objective, this method uses the C4.5 algorithm to build the decision tree, but it adds new attributes on the training set, which are linear combinations of the numeric attributes of the original database. A typical evolutionary algorithm is used to generate the best linear combination of the original numeric attributes. The nominal attributes are not included in this linear combination, and they are used independently. A comparison of the results of CGO3 and C4.5 are presented, using problems from UCI Repository. Tenfold cross validation measure of the performance of CGO3 are also given.

Keywords: Decision tree, classification, data mining, evolutionary algorithm.

1 Introduction

Nowadays, the capabilities of producing and collecting data are increasing rapidly. The wealth of information embedded in databases has generated an urgent need of new algorithms to discover useful qualitative information, previously unknown and potentially useful from these quantitative databases [2, 10, 21]. So, the field of data mining, sometimes referred to as knowledge discovery in databases or advanced data analysis, is awaking great interest in topics such as decision making, information management, medical diagnosis, performance prediction, and many other applications.

The data mining technique used depends on the particular objective and database to analyze. Classification systems are useful techniques in data mining, which are supervised learning methods that induce a classification model from a database [2]. The database, or the training set, is composed by a set of attributes, or features, of objects and each tuple has a known class, or label, associated with it. The objective is to induce a set of classification rules (classification model) for each class using the attributes available in the database. Such rules are used to classify future objects according to the value of their attributes.

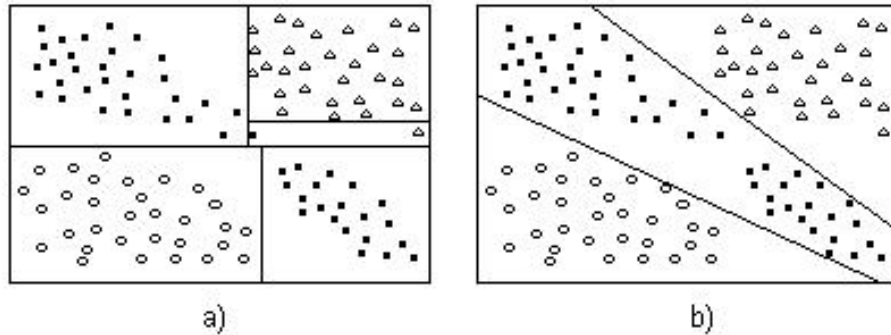


Figure 1: Classification systems a) parallel hyperplanes and b) obliques hyperplanes

Decision Trees are the most used classification method [10]. Decision tree method performs the classification in a tree structure, where each interior tree node contains a test, the edges from each node are labeled with a possible test value, and the leaves contain the classes. The classification of an object starts at the root, evaluates the test at the node, branches according to the result until a leaf is reached and this leaf determines the class of the object.

These classification systems can be basically divided into two categories [13]: axis parallel hyperplanes and axis oblique hyperplanes. In axis parallel hyperplane methods, a classification rule is composed by univariate tests, which compare the value of a single attribute to a constant. The tests have the form $x_i < k$, where x_i is of the attributes and k is a constant. They are equivalent to partitioning the search space with axis parallel hyperplanes. In axis oblique hyperplane methods, a classification rule is composed by a multivariate test, which compare a linear combination of attributes to a constant. The tests have the form $\sum_{i=1}^n a_i x_i < K$, where a_i is the real-valued coefficient for the attribute x_i . They are equivalent to partitioning the search space with oblique hyperplanes. Figure 1 shows an example of both methods for two attributes.

Axis parallel hyperplane methods have a disadvantage as opposed to axis oblique hyperplane methods. Classification models by axis parallel methods are not efficient for all databases, because there are databases that do not adapt to axis parallel models. This problem is shown in Figure 1. In these cases, axis parallel hyperplane methods induce classification models with a bigger number of rules and a bigger error percentage.

There are three widely used systems for decision tree induction: C4.5 [15], OC1 [13] and M5 [14, 20]. The C4.5 is an excellent classification system, but it is an axis parallel hyperplane method. OC1 and M5' are axis oblique hyperplane methods, but OC1 permits only numerical attributes, and M5' permits only numerical classes.

In this paper, we present CGO3 a new oblique hyperplane classification system based on an Evolutionary Algorithm [4, 6] and the C4.5. The main objective of this work is to solve the problems that C4.5 has on databases that are not adjusted parallel classification models. Thus, the purpose of CGO3 is to improve the results of C4.5, but offering a classification model with the same format. This improvement consists on reducing the number of rules and the error percentage. To reach our objective, CGO3 uses the C4.5 algorithm to build the decision tree, but it adds new attributes on the training set, which are linear combinations of the numeric attributes on the original database. A typical evolutionary algorithm is used to generate the best linear combination of the original numeric attributes. The nominal attributes are not included in this linear combination. Thus, the test at the node of the decision tree is a linear combination of original numerical attributes or a simple original attribute (numerical or nominal).

The rest of the paper is organized as follows. We offer the basic aspects about the C4.5 algorithm and we present the details of a simple evolutionary algorithm in section 2. Section 3 presents the CGO3 algorithm. Section 4 offers an illustration of its implementation on Iris dataset and a synthetic dataset. Section 5 presents a comparison of the results of CGO3 versus C4.5 on UCI Repository

C4.5 Algorithm

1. $T \leftarrow$ Read Training Set
 2. $D(\text{Size}, \text{Error}) \leftarrow$ Building Decision Tree
 3. $D'(\text{Size}', \text{Error}') \leftarrow$ Pruning Decision Tree
 4. Show Classification from Decision Trees
 5. Evaluate Test
- end
-

Figure 2: C4.5 Algorithm

datasets [12]. Finally, we conclude with some observations and future projects about this method in section 6.

2 Preliminaries

In this section, we present the basic aspects of CGO3. We also discuss the basic details of the C4.5 decision tree algorithm and present a simple evolutionary algorithm.

2.1 C4.5 Algorithm

A brief scheme of the C4.5 algorithm is shown in Figure 2. The C4.5 generates a decision tree from a training set. The decision tree has a size, number of nodes, and error (misclassified instances) associated. If it is possible, the decision tree is pruned by heuristic methods, with the aim of producing a simpler tree without compromising the accuracy of the original tree. Then, the classification is shown from the decision tree as a decision list. Finally, the classification is evaluated by a test set.

The decision tree building process can be expressed recursively by a divide and conquer technique. Thus, in each iteration, new nodes (test) are added, which have one branch for each possible value according to a certain criterion. This splits up the training set into subsets, one for every possible value of the test at the node. The process can be repeated for each branch, using the instances of this branch. The process finalizes when all instances or a significant number of them are of the same class. The accuracy of the classification is defined by the misclassified instances.

A splitting criterion is used to determinate to appropriate test. This criterion forms a fundamental decision in the decision tree building process. C4.5 uses the gain ratio criterion based on the gain criterion used in ID3. This criterion is based on: “the information conveyed by a message depends on its probability and can be measured in bits as minus the logarithm to base 2 of that probability”.

$$\frac{freq(C_i, S)}{|S|} \tag{1}$$

$$-\log_2\left(\frac{freq(C_i, S)}{|S|}\right) \text{ bits} \tag{2}$$

Imagine selecting one case at random from a set S of cases and announcing that it belongs to some class C_i . This message has probability (1) and so the information it conveys is (2).

$$info(S) = -\sum_{i=1}^k \frac{freq(C_i, S)}{|S|} \times -\log_2\left(\frac{freq(C_i, S)}{|S|}\right) \text{ bits} \tag{3}$$

To find the expected information from such a message pertaining to class membership, it sum over the classes in proportion to their frequencies in S , giving (3). When applied to the set training cases,

$info(T)$ measures the average amount of information needed to identify the class of a case in T . This quantity is also known as the entropy of the set S .

$$info_X(T) = - \sum_{j=1}^n \frac{|T_j|}{|T|} \times info(T_j) \quad (4)$$

$$gain(X) = info(T_j) - info_X(T) \quad (5)$$

Now consider a similar measurement after T has been partitioned in accordance with the n outcomes of a test X . The expected information requirement can be found as the weighted sum over the subsets, as shown in (4). The equation (5) measures the information that is gained by partitioning T in accordance with the test X . Thus, the gain criterion selects a test to maximize this information gain.

This splitting criterion is suitable for databases matching parallel classification models. In another case, this method induces a very high decision tree.

2.2 Evolutionary Algorithm

An evolutionary algorithm is a heuristic search process based on the principle of the evolution formulated by Darwin: “survival of the fitness” [4, 6].

```

Evolutionary Algorithm
1.   Pi ← Initialize population
2.   Repeat
3.       Evaluate Pi
4.       Select the best of Pi to Pi+1
5.       Select % of Pi to Pi+1
6.       Crossing Pi individuals to Pi+1
7.       Mutations on Pi+1
8.       Pi is Pi+1
9.   Until number of generations exceeded
end

```

Figure 3: Evolutionary Algorithm

An initial population, P_i , is generated randomly. Each individual of the population represents a potential solution to a problem. Each solution is evaluated to give some measure of its fitness. The next generation, P_{i+1} , is composed of the best individual, a percentage of the rest of the individuals, and new individuals generated by crossover of the individuals of the previous generation. The individuals of the new generation can be altered by mutations. This process can be repeated a number of generations. Figure 3 shows a simple genetic algorithm [11].

In our algorithm, an evolutionary algorithm is used for optimization of the linear combination of numerical attributes. We discuss its basic features in the next section.

3 The CGO3 Algorithm

In this section we discuss details of CGO3, our classification system. CGO3 is an oblique classification system based on C4.5 and an evolutionary algorithm. The core of the algorithm is an evolutionary algorithm where the individuals are real coefficients (weights) which represent linear combinations of the original numerical attributes [3, 5, 7, 8, 22]. The C4.5 algorithm is used to read the training set

```

CGO3 Algorithm
T ← Read Training Set
Pi ← Initialize population
Repeat
  For each I ∈ Pi
    T' ← Add Linear combination of I to T
    D(Size, Error) ← Building Decision Tree
    D'(Size', Error') ← Pruning Decision Tree
    Calculate fitness of I
  Select the best of Pi to Pi+1
  Select % of Pi to Pi+1
  Crossing Pi individuals to Pi+1
  Mutations on Pi+1
  Pi is Pi+1
Until number of generations exceeded
Show Classification from Decision Tree
Evaluate Test
end

```

Figure 4: CGO3 Algorithm

and to treat missing attribute values. Moreover, it builds the decision tree, determines the quality of a potential classification, builds the decision tree, shows the final classification, and evaluates the test set.

Initially, the CGO3 algorithm reads the numerical and nominal attributes of the training set and initializes a population of potential linear combinations, which are added to the existing training set. Then, it iterates for a number of generations. Finally, it shows a more accurate classification, which may contain original attributes and linear combinations of numerical attributes. In each iteration, the linear combinations of an individual are added to the original training set to evaluate the accuracy (fitness) of the classification that can be induced by the corresponding rule.

Figure 4 shows the CGO3 algorithm. We discuss details of the evolutionary algorithm in the next subsection. We present the values of the parameters used in CGO3 in Table 1.

a_{11}	a_{12}	a_{13}	...	a_{1d}
a_{21}	a_{22}	a_{23}	...	a_{2d}
\vdots	\vdots	\vdots	\ddots	\vdots
a_{m1}	a_{m2}	a_{m3}	...	a_{md}

Figure 5: Individuals data structure

Q_1	Q_2	Q_3	...	Q_m
-------	-------	-------	-----	-------

Figure 6: Abbreviate Individuals data structure

$$\begin{aligned}
Q_1 &= a_{11}x_1 + a_{12}x_2 + \dots + a_{1d}x_d \\
Q_2 &= a_{21}x_1 + a_{22}x_2 + \dots + a_{2d}x_d \\
&\vdots \\
Q_m &= a_{m1}x_1 + a_{m2}x_2 + \dots + a_{md}x_d
\end{aligned} \tag{6}$$

3.1 Individual Data Structure

The representation of individuals forms an important part of the evolutionary algorithm, since further components depend on it. In our algorithm, we use an array where each vector represents the real coefficients of the linear combinations.

The length of the vector depends on the original numerical attributes, and the number of vectors or linear combinations is a predetermined value. Thus, Figure 5 shows the individual data structure on d attributes $\langle x_1, x_2, \dots, x_d \rangle$ and m linear combinations. The m new attributes $\langle q_1, q_2, \dots, q_m \rangle$ added to the original training set is shown by equation (6).

The number of linear combinations influence the CGO3 algorithm. The higher combinations, the higher probability of find better decision hyperplanes, but this value increases the computational cost. The best value for this parameter is between 25 and 50. If this parameter takes values greater than 50, then the computational time is very high and the classification model does not have a substantially improvement.

In the next section, the individual data structure is abbreviated as shown in Figure 6. So, only the m linear combinations are represented.

3.2 Initial and Following Population

The initial population is generated randomly. Thus, $d * m$ coefficients are chosen randomly between a predetermined range ($[-10, 10]$). A linear combination is represented by each d coefficients.

The next populations are generated via the evolution process. Thus, a new population is generated with the best individual, a percentage of individuals of the previous population and new individuals generated by crossover operator. Some members of the new population undergo transformations by mutation operator. The best individual obtained during the evolutionary process will produce the final classification.

There are two parameters that influence the evolutionary process: the number of generations and the number of individuals. The election of these parameters should consider the required optimization and the computation cost, in order to avoid local maximum. The higher individuals, the higher the increase of the scope of the search process (breadth search) and the higher generations, the higher the accuracy of the search (depth search). In our experiments, these values have been determined by trial and error methods. These values are shown in Table 1.

3.3 The Fitness Function

In our algorithm, there are three concepts that determine the accuracy of a classification model: positive cases (classified instances), negative cases (misclassified instances) and the size of decision tree (or number of rules). Therefore, the fitness function must be able to analyze these concepts.

$$f(i) = Class(i) - MClass(i) - Size(i) * FRules \tag{7}$$

CGO3 uses the C4.5 algorithm to determine the fitness of individuals. Thus, the original training set is increased with new attributes that are deducted by linear combinations of the original numerical attributes according to Figure 5. A decision tree is generated by the C4.5 algorithm on the new training set. The classified instances, the misclassified instances and the size of the decision tree determine the fitness of this individual.

The fitness function f for an individual i is given by the equation (7), where $Class(i)$ are the number of classified instances on the decision tree induced by individual i ; $MClass(i)$ are the number of misclassified instances on the decision tree induced by individual i ; $Size(i)$ is the size of the decision tree induced by individual i ; and $FRules$ is the factor of influence of the rules and determines the importance of the number of rules on the classification model. $FRules$ takes its values between 0 and 1, so if it is close to 0 the induced classification model has a bigger number of rules than if it is close to 1.

The optimum value for $FRules$ depends of the datasets. The researcher have to establish this value in order to obtain classification models with different error rate, according to the number of rules.

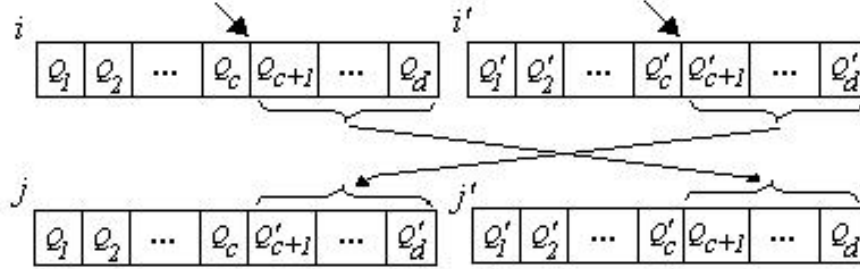


Figure 7: Crossover genetic operator

3.4 Genetic Operators

There are three typical genetic operators: selection, crossover and mutation. The selection operator selects the fitter individuals for the new population. The crossover operator combines the structure of individuals for the new population. The mutation operator transforms the individuals of a generation.

For the selection process, a roulette wheel [9] with slots sized according to fitness is used in CGO3. Thus, a random number b is generated from the range $[0..1]$, and then the ith individual, such that $\sum_{i=1}^j f(j) > b$, is selected. Otherwise, the first individual is selected. This process is repeated a number of times similar to a predetermined percentage of the number of individuals.

The crossover operator [11] selects two individuals, i and i' , by selection operator. For each pair of coupled individuals, two new individuals, j and j' , are added in the new population. Thus, a random number c is generated from the range $[1..d - 1]$, where d is the number of original numerical attributes. The number c indicates the position of the crossing point. The new individuals are generated as Figure 7 shows.

The next operator, mutation, is performed on the coefficients of individuals [9]. Three parameters are used for the mutation operator: P_i , P_q and P_c . The P_i parameter is the probability of mutation of an individual, the P_q parameter is the probability of mutation of a linear combination, and the P_c parameter is the probability of mutation of a coefficient.

$$coef = coef \pm Cant * coef * PMut \quad (8)$$

For each individual, a random number p_i is generated from range $[0..1]$; if $p_i > P_i$ then a random number p_q is generated from range $[0..1]$, for each linear combination; if $p_q > P_q$ then a random number p_c is generated from range $[0..1]$, for each coefficient; if $p_c > P_c$ then this coefficient is mutated by equation (8), where $coef$ is the actual value of the coefficient, $Cant$ is a random value from range $[0..1]$, $PMut$ is a predetermined percentage from range $[0..1]$. $PMut$ takes its values between 0 and 1, so if it is close to 0, the mutation is less significant that if it is close to 1.

4 CGO3 Implementation: an Illustration

To illustrate the development tools we present the results obtained on a synthetic dataset and the typical Irises dataset [12]. In Table 1, we show the parameters and their default values used in CGO3.

4.1 Synthetic Dataset

In this section, we present the best results obtained by CGO3 on a synthetic dataset during the training process. This synthetic dataset has been created to illustrate the characteristics of the tool.

$$l(x_1, x_2, x_3) = x_1 + x_2 + x_3 \quad (9)$$

$$c(p_1, p_2, p_3, p_4, p_5) = \begin{cases} A & \text{if } p_2 = p_{2a} \wedge l(p_1, p_3, p_4) < 150 \\ B & \text{if } (p_2 = p_{2b} \vee p_2 = p_{2c}) \wedge l(p_1, p_3, p_4) < 150 \\ C & \text{if } l(p_1, p_3, p_4) \geq 150 \end{cases} \quad (10)$$

The synthetic dataset has been elaborated with 5 attributes (p_1, p_2, p_3, p_4 and p_5) and 3 classes (A, B and C), where p_1, p_3 and p_4 are numerical attributes and p_2 and p_5 are nominal attributes. The attribute p_2 can take the values p_{2a}, p_{2b} and p_{2c} and the attribute p_5 can take p_{5a}, p_{5b} and p_{5c} . The distribution of the classes has been carried out according to function c , which is presented in equation (10). Function c uses a linear combination of attributes, l , which is shown in equation (9).

Figure 8 shows the induced classification model by CGO3 and C4.5 on the synthetic dataset. As it is appreciated in the figure, CGO3 finds a linear combination of attributes (Q6). By means of this linear combination, the number of rules and the error rate are reduced.

C4.5 [release 8] decision tree generator					CGO3 basado C4.5R8 - decision tree generator				
-----					-----				
Decision Tree:					Linear Combinations				
p2 = p2a: A (12.0)					Q6 -> (-2.2)p1+(-8.6)p3+(-3.0)p4.				
p2 = p2b:					Decision Tree:				
p5 = p5a: C (6.0)					Q6 <= -282.81 : B (24.0)				
p5 = p5b: B (13.0/1.0)					Q6 > -282.81 :				
p5 = p5c: B (0.0)					p2 = p2a: A (12.0)				
p2 = p2c:					p2 = p2b: C (7.0)				
p4 > 6 : B (9.0/1.0)					p2 = p2c: C (7.0)				
p4 <= 6 :									
p3 <= 19 : C (6.0)									
p3 > 19 : B (4.0)									
Before Pruning		After Pruning			Before Pruning		After Pruning		
-----		-----			-----		-----		
Size	Errors	Size	Errors	Estimate	Size	Errors	Size	Errors	Estimate
11	2(4.0%)	11	2(4.0%)	(19.7%) <<	6	0(0.0%)	6	0(0.0%)	(10.3%) <<
a)					b)				

Figure 8: Results of C4.5 and CGO3 on synthetic dataset

4.2 Iris Dataset

Another vision of the improvements of CGO3 over C4.5 can be obtained in a comparative on the classic Iris dataset. In this comparative, we offer the results of CGO3 for different values of the parameter $FRules$ in contrast with the classification model induced by C4.5

Table 1: CGO3 Parameters

Par.	Description	Value
-f	Training set	-
-u	Perform Tets	No
-q	Show Linear Combination	No
-nq	Number of Linear Combin.	40
-ng	Number of generations	50
-ni	Number of individuals	75
-ps	Percentage Selected	15%
-pi	Probability mutation ind	60%
-pq	Probability mutation L.C.	70%
-pc	Probability mutation coeff.	80%
-pm	Percentage mutation	70%
-rf	FRules factor	2%
-rc	coeff. predeterminate range	[-10,10]

Table 2: Datasets UCI Repository

Databases	#Cases	#Att	#Class	#Att num	#Att nom	#miss
BCW	699	10	2	10	0	yes
BUPA	345	6	2	6	0	no
CRX	690	15	2	6	9	yes
ECOLI	336	8	8	7	1	no
GLASS*	214	9	7	9	0	no
HAYES**	132	5	3	0	5	no
Hepatitis	155	19	2	6	13	yes
LABOR	57	16	2	8	8	no
N-Thyroid	215	5	3	5	0	no
PIMA	768	8	2	8	0	no

*Suppressing *id* attribute

** The Nominal attributes have been transformed to numerical attributes

```

C4.5 [release 8] decision tree generator
-----
Decision Tree:
pw <= 0.6 : Iris-setosa (50.0)
pw > 0.6 :
| pw > 1.7 : Iris-virginica (46.0/1.0)
| pw <= 1.7 :
| | pl <= 4.9 : Iris-versicolor (48.0/1.0)
| | pl > 4.9 :
| | | pw <= 1.5 : Iris-virginica (3.0)
| | | pw > 1.5 : Iris-versicolor (3.0/1.0)

Before Pruning      After Pruning
-----
Size  Errors Size  Errors Estimate
  9   3(2.0%)   9   3(2.0%)  (6.5%) <<

```

Figure 9: Results of C4.5 on Iris dataset

Table 3: Study of the Randomness

Databases	Size			Error		
	Max.	Av.	Min.	Max.	Av.	Min.
BCW	21	18.8	17	3	1.2	1
BUPA	67	63.2	61	4	2.5	2
CRX	119	116.4	113	12	10.8	8
ECOLI	31	29.6	29	7	5.8	2
GLASS	43	41.2	37	5	3.1	2
HAYES	27	25.2	21	3	1.7	0
Hepatitis	29	26.6	23	3	1.2	0
LABOR	16	13.2	12	1	0.8	0
N-Thyroid	7	6.6	5	1	0.9	0
PIMA	117	98.4	93	22	13.6	10

Figure 9 and 10 show the results of both tools on the Iris dataset. The results show the induced classification model in the training process. C4.5 induces a decision tree with 9 nodes and 2.0% error. CGO3 induces a decision tree with 9 nodes and 0.0% error when $FRules$ is 0.7 and a decision tree with 5 nodes and 0.7% error when $FRules$ is 0.2.

4.3 The Randomness of the Results

Evolutionary algorithms have two features that must be analyzed: the high parameterization and the randomness of the method.

The first problem needs a search of the values to fill the parameters in order to get the best algorithm performance. This task is tedious and often a frustrating experience. In our algorithm, the trial and error method has been used to find these values. Thus, for each parameter, several values have been tested on different datasets. The final values for each parameters are shown in Table 1.

In order to avoid the effects the second problem causes in CGO3, a comparison is offered on the UCI Repository datasets, whose features is shown in Table 2.

Table 3 shows the results of this comparison. These results have been obtained by 10 tries for each dataset, showing the average of the maximum, medium and minimum of both number of rules and percentage of error. These results have been induced on the same training and test sets.

<pre>CGO3 basado C4.5R8 - decision tree generator ----- Linear Combinations Q6 -> (-1.0)sl+(1.2)sw+(-1.0)pl+(-4.8)pw. Q11 -> (-4.4)sl+(-1.3)sw+(8.8)pl+(7.7)pw. Decision Tree: pw <= 0.6 : Iris-setosa(50.0) pw > 0.6 : Q11 <= 23.7338 : Iris-versicolor (48.0) Q11 > 23.7338 : Q6 <= -15.6566 : Iris-virginica (47.0) Q6 > -15.6566 : Q6 <= -15.5219 : Iris-versicolor (2.0) Q6 > -15.5219 : Iris-virginica (3.0) Before Pruning After Pruning ----- Size Errors Size Errors Estimate 9 0(0.0%) 9 0(0.0%) (4.1%) << a)</pre>	<pre>CGO3 basado C4.5R8 - decision tree generator ----- Linear Combinations Q14 -> (-1.1)sl+(-3.2)sw+(6.4)pl+(8.5)pw. Decision Tree: pw <= 0.6 : Iris-setosa(50.0) pw > 0.6 : Q14 <= 29.3895 : Iris-versicolor (49.0) Q14 > 29.3895 : Iris-virginica (51.0/1.0) Before Pruning After Pruning ----- Size Errors Size Errors Estimate 5 1(0.7%) 5 1(0.7%) (3.5%) << b)</pre>
---	---

Figure 10: Results of CGO3 on Iris dataset. a) FRules=0.7. b) FRules=0.2

The analysis of this table offers the typical variations of the randomness of the evolutionary process for each dataset. However, the results show that excessive differences do not exist, since the ranges are very reduced for both the size of the decision tree (± 4.7 nodes) and the error (± 3.6 errors). Also, these variations are not qualitative, but quantitative. Thus, we conclude that CGO3 is a robust method, in spite of this randomness of the results.

5 Empirical Evaluation

The previous results show a illustration of CGO3 in contrast to C4.5, but they do not offer a performance measurement of the new tool. So, a k -fold cross validation approach is used as a performance measurement of CGO3. The cross validation is performed k different times, each time using a different partitioning of the data into training and test sets, and the results are then averaged. Thus, the n available instances are partitioned into k subsets, each of size n/k . The cross validation procedure is then run k times, each time using a different one of these subsets as the test and combining the other subsets for training set.

A simple comparison is offered on 10 datasets. Table 2 shows the characteristics of the datasets used. The best classification for each dataset is presented in Table 4. These results refer to the training process and show the size of the decision tree, the number of errors and error rate, both before and after pruning. We can see that CGO3 reduces the number of rules and error percentage in most of the datasets.

To evaluate the performance of CGO3, a tenfold cross validation has been realized with each dataset. Table 5 offers the results of this tenfold cross validation for C4.5 and CGO3, showing the size of the decision tree, the number of errors and the error percentage, both before and after pruning.

A measure or ratio of the improvement [16] about the number of rules and error percentage for our tool is shown in Table 6. These values have been obtained from the results the CGO3, divided by the corresponding results of the C4.5, in Table 5. The column $R_{cgo3}/R_{c4.5}$ presents the results for the size of the decision tree of CGO3 divided by the size of the decision tree of C4.5. The column $E_{cgo3}/E_{c4.5}$

Table 4: Best results on Datasets UCI Repository

Databases	C4.5		CGO3	
	Before Pruning Size Error(%)	After Pruning Size Error(%)	Before Pruning Size Error(%)	After Pruning Size Error(%)
BCW	45 8(1.1)	31 11(1.6)	19 4(0.6)	15 6(0.9)
BUPA	53 52(15.1)	45 16(7.5)	69 13(3.8)	59 17(4.9)
CRX	140 34(4.9)	41 64(9.3)	108 29(4.2)	39 53(7.7)
ECOLI	51 21(6.3)	43 22(6.5)	43 12(3.6)	39 14(4.2)
GLASS	51 14(6.5)	45 16(7.5)	55 3(1.4)	55 3(1.4)
HAYES	27 13(9.8)	21 14(10.6)	33 5(3.8)	31 6(4.5)
Hepatitis	31 6(3.9)	23 11(7.1)	29 3(1.9)	23 5(3.2)
LABOR	22 2(3.5)	5 7(12.3)	14 0(0.0)	8 2(3.5)
N-Thyroid	17 3(1.4)	17 3(1.4)	7 0(0.0)	7 0(0.0)
PIMA	43 120(15.6)	43 120(15.6)	121 38(4.9)	115 41(5.3)

Table 5: Tenfold cross-validation results

Databases	C4.5				CGO3			
	Before Pruning		After Pruning		Before Pruning		After Pruning	
	Size	Error(%)	Size	Error(%)	Size	Error(%)	Size	Error(%)
BCW	39.2	3.1(4.5)	19.8	3.2(4.6)	19.8	3.4(4.9)	18.6	3.7(5.4)
BUPA	47.8	4.6(13.5)	41.7	5.4(15.9)	65.6	3.0(8.8)	61.6	3.6(10.5)
CRX	143.5	8.2(11.9)	31.1	8.4(12.2)	119.0	11.1(16.0)	39.1	8.7(12.6)
ECOLI	48.6	5.2(15.7)	39.4	5.2(15.7)	35.0	4.4(13.3)	33.8	4.6(13.9)
GLASS	50.6	3.6(17.2)	47.8	4.0(19.0)	41.8	3.4(16.1)	39.8	3.9(18.5)
HAYES	27.4	2.2(16.9)	24.6	2.4(18.5)	26.2	1.8(13.8)	24.2	2.1(16.1)
Hepatitis	30.0	1.7(11.3)	22.2	2.0(13.3)	26.8	1.2(8.0)	22.6	1.6(10.6)
LABOR	14.8	1.0(16.6)	7.8	1.0(16.6)	12.2	0.8(13.3)	6.8	1.0(16.6)
N-Thyroid	16.6	1.1(5.2)	15.8	1.2(5.7)	6.6	0.5(2.4)	6.6	0.6(2.8)
PIMA	38.2	14.6(19.2)	34.4	14.7(19.3)	88.4	13.6 (17.9)	82.6	14.4 (18.9)
Av.1	45.6	4.5(13.2)	28.4	4.7(14.0)	45.9	4.3(11.4)	35.3	4.4(12.5)
Av.2(*)	46.4	3.3(12.5)	27.7	3.5(13.4)	41.1	3.2(10.6)	30.0	3.2(11.7)

*Not considering PIMA dataset

Table 6: A general comparison of C4.5 and CGO3

Databases	$R_{cgo3}/R_{c4.5}$	$E_{cgo3}/E_{c4.5}$
BCW	0.50	1.08
BUPA	1.37	0.65
CRX	0.82	1.34
ECOLI	0.72	0.84
GLASS	0.82	0.93
HAYES	0.95	0.81
Hepatitis	0.89	0.70
LABOR	0.82	0.80
N-Thyroid	0.39	0.46
PIMA	2.31	0.93
Av.1	0.95	0.85
Av.2(*)	0.80	0.84

*Not considering PIMA dataset

presents the results for the error percentage of CGO3 divided by the error percentage of C4.5.

As the overall averages at the foot of Table 6 indicate, the classification models produced by CGO3 are smaller (5%) and more accurate (15%) than those generated by C4.5. CGO3 is less accurate than C4.5 on only two (BCW and CRX) of the ten datasets, but the decision tree of CGO3 is substantially smaller (50% and 18%, respectively) than the C4.5 decision tree in these cases.

A special case in this comparison is PIMA dataset. In this dataset, the size of the decision tree of CGO3 is bigger (47%) than the size of the decision tree of C4.5. However the error percentage of CGO3 is not proportionally smaller (7%) than the error percentage of C4.5. This problem is due to the distribution of the classes on PIMA dataset, since cases of different classes are very close to each other. Classification tools based on the nearest neighbor induce approximately a 25% margin of error. Thus, in order to reduce the error percentage a great number of rules are necessary. As the aim of CGO3 has been to reduce error percentage, the size of the decision tree for PIMA dataset has a great number of nodes.

If the PIMA dataset is not considered, then the classification models produced by CGO3 are smaller (20%) and more accurate (16%) than those generated by C4.5

6 Conclusions and Future Work

In this paper we have presented a new classification system based on C4.5 and an evolutionary algorithm. The main objective of this method is to improve the classification models induced by C4.5 on databases that are not adjusted to parallel classification models. In order to attain this improvement, our method induces oblique classification models.

Analysis of the results shown in sections 4 and 5 illustrates that the objective has been reached. The classification models induced by the new developed algorithm are smaller and more accurate than those induced by the original algorithm, as for the number of rules, the error rates or both. Also, the new algorithm allows the expert to intervene in the learning process, for example, establishing the importance of the number of rules ($FRules$). The expert can need a classification model with a low number of rules ($FRules \approx 1$), or a model with a low error and a bigger number of rules ($FRules \approx 0$).

The disadvantages of CGO3 compared to the C4.5 are the high computational time and the interpretation of the extracted rules. The high computational time is an inherent feature of evolutionary algorithms, the CGO3 computational complexity is $O(g \times i \times n \times d \times \log n)$, where g is the number of generation, i is the number of individuals, n is the number of database instances and d is the number of attributes.

Oblique classification models are more complex than parallel classification models according to the way it interprets the extracted rules. The CGO3 rule complexity is due to the linear combinations. However, this complexity does not implicate that the CGO3 classification models are more complex than the C4.5 classification models, since a rule as $x_1 > x_2$ is directly induced by CGO3, but the C4.5 can not induce it.

At present, we are working in two lines of improvement for our tool: expert's interaction and to reduce the linear combination. Thus, we are working on the fitness function in order to allow the expert to decide the maximum error percentage or the maximum number of rules of the final classification model. As a first approximation, the fitness function penalizes individuals that do not match the petitions requested by the expert. Moreover, we are developing a new algorithm where the linear combination does not have all numerical attributes.

References

- [1] J. L. Alvarez, J. Mata and J. C. Riquelme, "CGO 1.0: Clasificador Genetico Oblicuo basado en el C4.5," in *Proceedings of VII Conferencia de la Asociacin Espaola para la Inteligencia Artificial, CAEPIA'99*, pp. 26–33, 1999.
- [2] M. S. Chen, J. Han and P. S. Yu, "Data mining: An overview from database perspective," *IEEE Transactions on Knowledge and Data Engineering*, 8(6), 1996.
- [3] K. A. De Jong, "Using genetic algorithms for concepts learning," *Machine Learning*, pp. 161–188, 1993.
- [4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Pub. Company, inc., 1989.
- [5] L. J. Eshelman and J.D. Schaffer, "Real-coded genetic algorithms and interval schemata," *Foundations of Genetic Algorithms 2*, Morgan Kaufmann Publishers, pp. 187–202, 1993.
- [6] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University Michigan Press, Ann Arbor, 1975.
- [7] C. Z. Janikov, "A knowleged-intensive genetic algorithm for supervised learning," *Machine Learning*, pp. 189–228, 1993.
- [8] J. R. Koza, *Concept Formation and Decision Tree Induction using the Genetic Programming Paradigm*, Springer-Verlag, 1991.
- [9] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, 1996.
- [10] T. M. Mitchell, *Machine Learning*, McGraw Hill, 1997.
- [11] Z. Michalewicz, *Genetics Algorithms + Data Structures = Evolution Programs*, Third Edition, Springer-Verlag, 1999.
- [12] P. M. Murphy and D. W. Aha, *UCI Repository of Machine Learning Databases*, Department of information and Computer Science, University of California, available at: <ftp://ftp.ics.uci.edu/pub/machine-learning-databases/>, 1994.
- [13] S. K. Murthy, S. Kasif and S. Salzberg, "A system for induction of obliques decision trees," *Journal or Artificial Intelligence Research*, vol. 2, pp. 1–32, 1994.
- [14] J. R. Quinlan, "Learning with continuous classes," in *Proceedings AI'92*, pp. 343–348, 1992.

- [15] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.
- [16] J. R. Quinlan, “Improved use of continuous attributes in C4.5,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 77–90, 1996.
- [17] J. Riquelme, J. Aguilar and M. Toro, “A tool to obtain a hierarchical qualitative rules from quantitative data,” in *Proceedings of 11th International Conference on Industrial and Engineering Applications of AI and ES*, pp. 105–114, 1998.
- [18] T. Van de Merckt, “A system that learns flexible concepts based on decision trees for numerical attributes,” in *Proceedings of Ninth International Workshop on Machine Learning*, pp. 322–331, 1992.
- [19] T. Van de Merckt, “Decision trees in numerical attribute spaces,” in *Proceedings of 13th International Conference on Artificial Intelligence*, pp. 1016–1021, 1993.
- [20] Y. Wang and I. H. Witten, “Induction of model trees for predicting continuous classes,” in *Proceedings of the Poster Paper of the European Conference on Machine Learning*, pp. 128–137, 1997.
- [21] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, 2000.
- [22] A. H. Wright, “Genetic algorithm for real parameter optimization,” En Rawlins, J. G., *Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers, 1991.