# OBLIC: Classification System Using Evolutionary Algorithm

J.L. Alvarez[1], J. Mata[1], and J.C. Riquelme[2]

[1] Universidad de Huelva, Spain,
{alvarez,mata}@uhu.es
[2] Universidad de Sevilla, Spain,
riquelme@lsi.us.es

**Abstract.** We present a new classification system based on Evolutionary Algorithm (EA), OBLIC. This tool is an OBLIque Classification system whose function is to induce a set of classification rules no hierarchical from a database or training set. The core of the algorithm is a EA with real-coded and Pittsburgh approach. Each individual is composed by a no fixed classification rules set what split in regions the search space. The fitness of each classification is obtained by means of the exploration of these regions. The result of the tool is the best classification obtained in the evolutionary process.
This paper describe and analyze this new method by comparing with other classification systems on UCI Repository databases. We conclude this paper with some observations and future projects.

## 1 Introduction

Nowadays, the growing of data acquisition technologies produces a huge volume of information. Actually, people can't handle this volume of information by manual procedures. Thus, there has been an explosion of machine learning and Data Mining systems that attempts to knowledge discovery of this information. Between the most popular techniques are the classification systems. The classification systems have as objective to induce a rules set from a labeled database.

Decision trees are a particularly useful technique to hierarchical classification system [8][7]. These techniques generate a decision tree from which the classification is interpreted. On the other hand, EA's [2][5][9] have been applied in numerous machine learning and Data Mining problems with excellent results [1][3][4].

In this paper, we describe the theoretical base and development of a classification system based on EA whose focus is to induce a classification rules set from a labeled database with numerical attributes. Each potential solution (classification) consists of a rules set that splits into regions the search space. The fitness of each possible classification is evaluated in the evolution process attending to the correctly classified elements by each region.

The developed tool is framed inside the denominated oblique classification systems. This type of tools restricts its use domain to databases with exclusively numeric attributes. For it, the databases with exclusively symbolic or heterogeneous attributes should be adapted.

In the following sections the basic concepts of the tool are detailed. In section 2 we start with brief description of the EA framework used. We follow with a overview about OBLIC implementation in section 3. Then, in section 4 we compare the results obtained on UCI Repository database [6], with the traditional classification systems C4.5 [8] and OC1 [7]. Finally, in section 5, we conclude with some observation about the development tool and future projects.

## 2 Algorithm

The core of the algorithm is a EA with real-coded individual (potential solutions). Each individual consists of a vector set, where each vector represents a classification rule. A classification rule consists of a float numbers vector, that it describes the equation of hyperplane that splits into regions the search space. The evaluation function determines the fitness of these individuals according to the number of correctly classified elements of training set. The crossover and mutation genetics operators generate the next population in this evolutionary process.

In the following subsections we approach with more detail each one of the significant elements of the EA. The most significant parameters are shown in section 3.

### 2.1 Individual Data Structure

Each potential classification are composed a string set of float numbers vectors. In this words, a two-dimensional matrix where each *ith* string represent the *ith* classification rule. Thus, each *jth* column represent the *jth* float coefficient of the hyperplane equation for the *ith* rule.

$$a_{i1}X_1 + a_{i2}X_2 + \cdots + a_{id}X_d + a_{d+1} \tag{1}$$

Thus, for d dimensional search space, the *ith* string implies the classification rule shown in the equation 1.

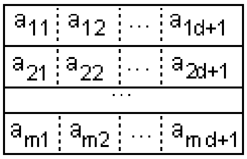

$a_{11}$ $a_{12}$ … $a_{1d+1}$
$a_{21}$ $a_{22}$ … $a_{2d+1}$
…
$a_{m1}$ $a_{m2}$ … $a_{md+1}$

**Fig. 1.** Individual data structure

$$\begin{array}{c} a_{11}X_1 + a_{12}X_2 + \cdots + a_{1d}X_d + a_{1d+1} = 0 \\ a_{21}X_1 + a_{22}X_2 + \cdots + a_{2d}X_d + a_{2d+1} = 0 \\ \vdots \\ a_{m1}X_1 + a_{m2}X_2 + \cdots + a_{md}X_d + a_{md+1} = 0 \end{array} \tag{2}$$

For the individual representation, we use the Pittsburgh approach, thus, each individual in the population consists of a variable-length rules set. Graphically, figure 1 show the individual structure for d dimensional search space ($X_1$, $X_2$, $\cdots$, $X_d$ attributes) and m-rules. Equation 2 show the rules set deduced from the individual of the figure 1.

## 2.2 Initial Population

The initial population of the individuals is choosing randomly. For each rule, d+1 float values are generated, where d is the number of attributes or dimension of the search space. These values represent the equation of the hyperplane of each classification rule according to 1. The number of rules of the individuals is selected between 1 and an maximum value and the coefficients of each rule are chosen about a range. This values are detailed in section 3.

## 2.3 Evaluation Function

The fitness of a potential classification is obtained by the exploration of search space. So, the rules of a classification split in regions the search space, this regions are analyzed and the accuracy of the classification is returned as fitness.

Thus, we establish an ordination of classification rules of the individual. This ordination allows a binary coding of the regions. Each region is coded by a bit series, where the *ith* element corresponds with the *ith* classification rule. The value of the *ith* bit is 0 if the region is to the left of the *ith* rules and it is 1 if the region is to the right.

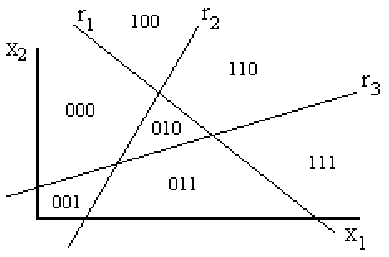


**Fig. 2.** Binary coded regions

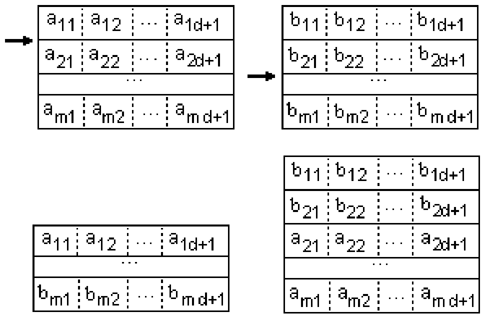
Figure 2 shows an example of the binary coding of a individual about a two-dimensional space (two attributes: $X_1$, $X_2$) with three rules.

$$f(n) = Pos(n) - Neg(n) - Rules(n) * FIR \tag{3}$$

The items of the database are examined about each region to obtain the fitness of each individual. Thus, a region will be labeled with majority class. The items with the same class that a region is positive cases, while the cases with a different class are negative cases for this region.

Let $n$ be individuals, $Pos(n)$ be total positive cases, $Neg(n)$ be total negatives cases, $Rules(n)$ be the number of rules of the classification and $FIR$ be the Factor Influence of Rules, between 0 a 1, the fitness of the individuals is evaluated by maximized 3.

The $FIR$ factor allows to determine the influence of the number of rules in the classification. So, if this value is near to 0 then the number of rules has very low influence, and if this value is near to 1 then the number of rules has very high influence.



$a_{11}$ $a_{12}$ ... $a_{1d+1}$
$a_{21}$ $a_{22}$ ... $a_{2d+1}$
$a_{m1}$ $a_{m2}$ ... $a_{md+1}$
$b_{11}$ $b_{12}$ ... $b_{1d+1}$
$b_{21}$ $b_{22}$ ... $b_{2d+1}$
$b_{m1}$ $b_{m2}$ ... $b_{md+1}$

**Fig. 3.** Crossover operator

## 2.4 Genetic Operators

New individuals are gererated by the crossover operator. The crossover operator selects two individuals and produces a new individual of them. Thus, a crossing point is selected in each father individual and the new individual is created by the previous genes to the crossing point of one and the genes next to the point of crossing of the other one. Graphically, the figure 3 show this crossover operator.

The mutation operators alter the individuals of a population. There are three mutation operators. This operators are:

- Alters
- Add
- Rest

This mutation operators affect the individuals randomly. Thus, the crossing possibility is raffled and if this is bigger than the mutation rate then one of the mutation operators will be applied. The mutation operator is selected randomly.

The first mutation operator alters one o more genes (coefficients) of the individual, randomly. The genes are altered a randomly proportional quantity with equal percent mutation. The second, Add mutation operator, add new genes for individual. Thus, one new rule is added to classification. And the Rest operators eliminate genes for one existent rule. Thus, one existent rules is suppressed of the classification.

**Table 1.** EA parameters

| Parameter | Value |
|---|---|
| individuals | 100 |
| generations | 200 |
| Mutation rate | 80% |
| Mutation percent | 70% |
| rules individuals | 12 |
| Range coefficients | [-10,10] |
| FIR | 10% |

## 3 Implementations

The previous algorithm has been developed as OBLIC. The values of the principal parameters are shown in table 1.

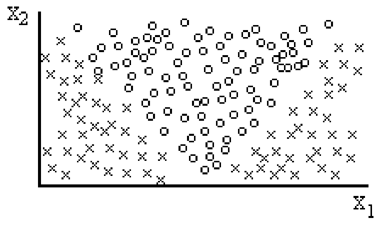


**Fig. 4.** Synthetic database

### 3.1 Practical Implementation

In this subsection, we show the results obtained on a synthetic dataset. The dataset is composed by two attributes ($X_1, X_2$) and two class (x, o). The classes have been distributed with 4 by random values of the $X_1$ attribute. An approximate representation is shown in the figure 4.

$$X_2 = (X_1 - 5)^2 \tag{4}$$

The results obtained for the previous database are:

```
Training set: synthetic.dat
200 cases. 2 attributes. 2 class.

Classification
==============
Regions
-------
Region 0 -> Class x. Pos: 56. Neg: 0
Region 2 -> Class o. Pos: 89. Neg: 0
Region 3 -> Class x. Pos: 55. Neg: 0

Rules
-----
(-6.00)X1 + (-3.00)X2 + (30.00) = 0
(-6.00)X1 + (3.00)X2 + (30.00) = 0

#Rules: 2     Errors: 0 ( 0 %)
```

The results show the regions and the rules for the classification. For each region, it shows its class and the positives and negatives cases. After, the classification rules are shown in the order to binary coding of the regions. Finally, the numbers of the rules, the numbers of the negatives cases and the percentage of errors are shown. The figure 5 represent, graphically, the result on the synthetic database.

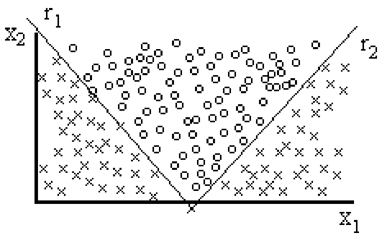


**Fig. 5.** Classification of synthetic database

## 4 Results

The performance of this method is compared with traditional classification system C4.5 and OC1 on real UCI Repository database. In table 2 we show the databases used. The results are summarized in the table 3.

In the table 3 we summarize the results obtained by the C4.5, OC1 and Oblic classifiers systems on five UCI Repository databases (bupa, iris, new-troid, pima, glass). For each one, we are shown the number of rules (#*Ru.*) and the error percentage (%*Er.*) obtained during the phase of tests.

The results have been generated starting from four tests for each database. In each test we have used 80% of the items for training set and 20% for test set. For each database, the results show the average value of tests.

**Table 2.** Databases

| Databases | #Attr. | #Cases | #Cases-test | #Classes |
|---|---|---|---|---|
| BUPA | 7 | 345 | 69 | 2 |
| IRIS | 4 | 150 | 27 | 3 |
| TROID | 5 | 215 | 43 | 3 |
| PIMA | 8 | 768 | 153 | 2 |
| GLASS | 10 | 214 | 42 | 6 |

**Table 3.** Comparative results

| | C45 | | OC1 | | OBLIC | |
|---|---|---|---|---|---|---|
| | #Ru. | %Er. | #Ru. | %Er. | #Ru. | %Er. |
| BUPA | 26 | 12.31 | 3 | 15.21 | 12 | 12.60 |
| IRIS | 4 | 2.77 | 3 | 1.85 | 3 | 0.92 |
| TROID | 8 | 1.74 | 5 | 1.16 | 6 | 0.58 |
| PIMA | 21 | 16.01 | 21 | 9.96 | 12 | 19.44 |
| GLASS | 5 | 1.19 | 5 | 1.19 | 5 | 1.19 |

## 5 Conclusion

In this paper, we present the theoretical aspects and we offer the implementation of a classification system based on an EA. This tool offers an alternative to the decision trees classification systems.

Obviously, this tool has the disadvantages characteristic of the AE, as the randomness and the high time of computation. But it presents other very significant advantages.

About the advantages we highlight: the classification rules is not hierarchical and the division criterion is not established a priori.

The non-hierarchical classification allows that the rules can be evaluated independently and no order. And the division criterion will be induced from the training set, itself.

Nowadays, we are working on new projects where a classification system is been developed. This new tools permits establish the number of rules and/or the maximum error rate of the final classification.

## References

1. K.A. De Jong: Using Genetic Algorithms for Concepts Learning". Machine Learning, **13**, pp. 161-188, (1993).
2. D.E. Golberg: Genetic Algorithms in Search, Optimization and Machine Learning". Addison-Wesley Pub. Company, inc., 1989.
3. C.Z Janikov: A Knowleged-Intensive Genetic Algorithm for Supervised Learning". Machine Learning, **13**, pp. 189-228, (1993).
4. J.R. Koza: Concept Formation and Decision Tree Induction using the Genetic Programming Paradimg. Springer-Verlag, (1991).
5. Z. Michalewicz: Genetics Algorithms + Data Structures = Evolution Programs. Third Edition. Springer-Verlag, (1999).
6. P.M. Murphy, D.W. Aha: UCI Repository of Machine Learning Databases. http://www.ics.uci.edu". Department of information and Computer Science, University of Californnia, (1994).
7. S.K. Murthy, S. Kasif, S. Salzberg: A System for Induction of Obliques Decision Tress. Journal or Artificial Intelligence Research, vol. **2**, pp. 1-32, (1994).
8. J.R. Quinlan: C4.5: Programs for Machine Learning. Morgan Kaufmann Pub., (1993).
9. A.H. Wright: Genetic Algorithm for Real Parameter Optimization. Morgan Kaufmann Pub., (1991).