ELSEVIER

Applied Soft Computing 1 (2002) 257–269

Applied Soft Computing

www.elsevier.com/locate/asoc

# Discovering interesting classification rules with genetic programming

I. De Falco [a,*], A. Della Cioppa [b], E. Tarantino [a]

[a] *ISPAIM, National Research Council of Italy, Via Patacca 85, I-80056 Ercolano (NA), Italy*
[b] *Department of Computer Science and Electrical Engineering, University of Salerno, Via Ponte Don Melillo 1, Salerno, I-84084 Fisciano (SA), Italy*

Received 7 December 2000; received in revised form 23 October 2001; accepted 30 October 2001

**Abstract**

Data mining deals with the problem of discovering novel and interesting knowledge from large amount of data. This problem is often performed heuristically when the extraction of patterns is difficult using standard query mechanisms or classical statistical methods. In this paper a genetic programming framework, capable of performing an automatic discovery of classification rules easily comprehensible by humans, is presented. A comparison with the results achieved by other techniques on a classical benchmark set is carried out. Furthermore, some of the obtained rules are shown and the most discriminating variables are evidenced. © 2002 Elsevier Science B.V. All rights reserved.

*Keywords:* Data mining; Classification; Genetic programming

## 1. Introduction

In the last years information collection has become easier, but the effort required to retrieve relevant pieces of it has become significantly greater, especially in large-scale databases. As a consequence, there has been a growing interest in powerful tools capable of facilitating the discovering of interesting and useful information within such a huge amount of data.

The process of knowledge discovery, known as data mining [6], brings together the latest research in statistics, databases, machine learning and artificial intelligence. The core of this process is the application of machine learning-based algorithms to databases. There are two basic ways of performing data mining: the supervised and the unsupervised learning. The former exploits known cases that show or imply well-defined patterns to find new patterns by means of which generalizations are formed. Experts lead search towards some features which are supposed to be of preminent interest, and the relationships between those features and the remaining ones are sought. In the unsupervised learning, instead, data patterns are found starting from some logical characterization of the regularities in a set of data. In this case, no preassumptions are made about the forms of relations among attributes.

Data classification represents perhaps the most commonly applied supervised data mining technique. It consists in generating from a set of class-labeled training examples a set of grouping rules which can be used to classify future patterns.

There are many methods used to face the classification task. They include decision-tree methods which

* Corresponding author. Tel.: +39-081-5608330; fax: +39-081-6139219.
*E-mail address:* i.defalco@ispaim.na.cnr.it (I. De Falco).

1568-4946/02/$ – see front matter 
PII: S1568-4946(01)00024-2

operate performing a successive partitioning of cases until all subsets belong to a single class [25,26], and statistical and rough sets approaches [6,33]. Though these classification tools are algorithmically reliable, they require significant expertise to work effectively and do not provide intelligible rules. Also neural networks [13,27], widely applied within the classification task, suffer from the latter drawback.

The classification problem becomes very hard when the number of possible different combinations of parameters is high. Hence, heuristics which effectively explore large solution spaces without performing exhaustive searches and, in some cases, provide intelligible classification rules are very appealing. Most data mining-related genetic algorithms (GAs) [11] proposed in the literature address the task of rule extraction in propositional and first-order logic [1,2,10,21]. Further interesting GA-based methods for choosing appropriate sets of fuzzy if-then rules for classification problems can be found in [14,22]. Hybrid classification learning systems involve a combination of artificial neural networks with evolutionary techniques [32] and with linear discriminating models [7,9], and an integration of rule induction and lazy learning [17]. Finally, genetic programming (GP) [16] frameworks for discovering comprehensible classification rules have been investigated [3,8,15,20].

In this work the aim is to exploit the effectiveness of a GP framework to find as-compact-as-possible explicit classification rules for real-world problems making up the PROBEN1 benchmark set [24]. The hope is that GP detects for any problem rules containing the most discriminating attributes only. These attributes combined with the user knowledge allow to make a well-informed decision, rather than blindly trusting in the incomprehensible output of a 'black box' system.

The paper is organized as follows: in Section 2 our GP-based classification system is outlined together with its implementation details. In Section 3 the test problems are briefly described and some related work is reported. In Section 4 the performance of our system compared with that achieved by other tools is discussed, some of the achieved rules are reported and several discriminating variables are identified. The last section illustrates our final remarks and future work.

## 2. The genetic programming system

The aim is the implementation of a genetic system able to automatically extract an intelligible classification rule for each class in a database, given the values of some attributes, called predicting attributes. Each rule is constituted by a logical combination of these attributes. This combination determines a class description which is used to construct the classification rule.

Once defined a fitness function, the classification problem becomes a search problem of the best description in the search space of all the possible solutions, that is to say an optimization of the fitness function for which optimization techniques can be used. It is comprehensible that the optimal search of a classification rule includes two tightly coupled subproblems: the search of the most discriminating attributes and the search of the variation interval within the domains of these attributes.

Given a number of attributes describing each object and their related domains, it is easily understandable that for complex classification problems the number of possible descriptions is enormous. An exhaustive search by enumerating all the possible descriptions is computationally impracticable. Hence, we appeal to GP which is a powerful search method inspired by natural selection. It does not guarantee to find the global optimum, nonetheless it usually allows to retrieve a suboptimal solution in a reasonable computation time.

The evolving population is constituted by individuals or 'programs' representing the classification rules in the form of trees whose sizes are intrinsically variable in length. Each rule is constituted by a number of conditional clauses, in which conditions on or between certain attributes are set, and by a predictive clause representing the class. A class together with its description forms a classification rule of type 'if ⟨description⟩ then ⟨class⟩'. The conditional part of the rule is formed by all the active conditional clauses.

A population composed by a set of these candidate rules is maintained and gradually improved by constructing new fitter ones until rules of sufficient quality are found or other stopping criteria are satisfied.

The system allows to attain rules covering the different classes by running the evolutionary algorithm as many times as the number of the classes to predict. In practice the system will analyze one class at a time.

To construct the classification model, data is partitioned into two sets: the training and the test sets. The training set contains the known objects used during the evolution process to find an explicit classification rule able to separate an instance of a class from instances of all other classes, while the test set is used to evaluate the generalization ability of the rule found. The major steps of this genetic system can be formalized as follows:

(1) generate at random an initial population of rules representing potential solutions to the classification problem for the class at hand;
(2) evaluate each rule on the training set by means of a fitness function;
(3) select the rules to undergo the mechanism of reproduction;
(4) apply the genetic operators *crossover*, *copy* and *mutation* to produce new rules;
(5) reinsert these offspring to create the new population;
(6) repeat steps (3) to (6) until an acceptable classification rule is found or the specified maximum number of generations has been reached;
(7) repeat steps (2) to (7) until one rule is determined for each class in the database;
(8) assign each example in the training and in the test sets to one and only one class.

The importance of step (8) is in that in all of the previous steps we deal with one class at a time. As a consequence of this, given a database with *Cl* classes, it may happen that some samples cannot be assigned to one and only one class: they are called indeterminate cases. This means they are either captured by more than one rule ("*yes*" indeterminate case) or by no rule ("*no*" indeterminate case).

Since in this paper we wish to make use of our GP system to face several test sets taken from the PROBEN1 repository, we must comply with the guidelines related to this benchmark. One of these guidelines is that any sample must be classified as belonging to one and only one class, so we need to perform a post-processing phase to eliminate the indeterminate cases of both types. Since any rule represents in PROBEN1 a subset in $[0.0, 1.0]^N$ with $N$ denoting the number of attributes, if the example does not satisfy any of the found rules, it is assigned to the class from the frontier of which it has the lowest distance. If the example satisfies more than one of the found rules, it is assigned to the class from the frontier of which it has the highest distance.

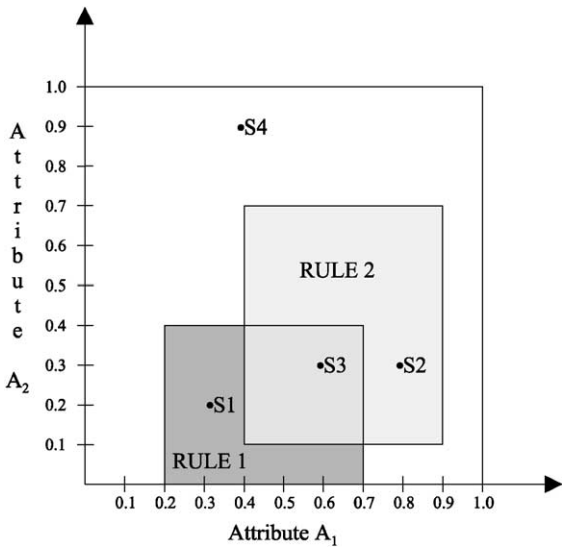


Fig. 1. Examples of assignment of indeterminate cases to only one class.

This can be intuitively explained by means of some visual examples, at least for simple rules. Let us suppose to be dealing with a two-class two-variable database and that any variable can vary within [0.0, 1.0]. In Fig. 1 we can see a graphical example of how cases are assigned to classes. In it each rule is graphically represented by the corresponding area.

The two rules are:

*Rule* 1: IF $((0.2 \leq A_1 \leq 0.7)$ AND $(A_2 \leq 0.4))$
THEN *class*1
*Rule* 2: IF $((0.4 \leq A_1 \leq 0.9)$ AND $(0.1 \leq A_2 \leq 0.7))$
THEN *class*2

According to these two rules, the sample $S1 = (0.3, 0.2)$ can be easily seen as belonging to *class*1. Likewise the sample $S2 = (0.8, 0.3)$ can be classified in *class*2. As regards the sample $S3 = (0.6, 0.3)$, both rules cover the point, i.e. the sample is a "*yes*" indeterminate case. Since, as we can visually evaluate, $S3$ is closer to the frontier of *Rule* 1 than it is to the frontier of *Rule* 2, it is more internal to *Rule* 2, hence it can be assigned to *class*2 rather than to *class*1.

Similarly, let us consider one more sample $S4 = (0.4, 0.9)$. It is covered by no rule, so it is a "*no*"

indeterminate case. Nonetheless, it is closer to the frontier of *Rule* 2 than it is to the frontier of *Rule* 1, so it can be seen as "missed" by *Rule* 2 by a smaller quantity than it is "missed" by *Rule* 1. Therefore, $S4$ is assigned to *class*2.

Things are similar when an OR clause is present in a rule. As an example, let us consider the two following rules:

*Rule* 1: IF $((A_1 \leq 0.3)$ OR $(A_2 \leq 0.3))$
THEN *class*1
*Rule* 2: IF $((0.6 \leq A_1 \leq 0.9)$ OR $(0.8 \leq A_2))$
THEN *class*2

If we take a look at Fig. 2 we can see that sample $S1 = (0.4, 0.1)$ belongs to *class*1, and that sample $S2 = (0.8, 0.9)$ belongs to *class*2. As regards $S3 = (0.5, 0.7)$, it is a "*no*" indeterminate case. It is clear that $S3$ is closer to the frontier of *Rule* 2, so it can be classified as belonging to *class*2. The sample $S4 = (0.1, 0.9)$ in the same figure, instead, is covered by both rules ("*yes*" indeterminate case), but it is closer to the frontier of *Rule* 2, so it is better covered by *Rule* 1, resulting in its assignment to *class*1.

For more complicate cases, where the rules are composed by several ANDs, ORs and NOTs, organized in a hierarchical way, a procedure has been devised

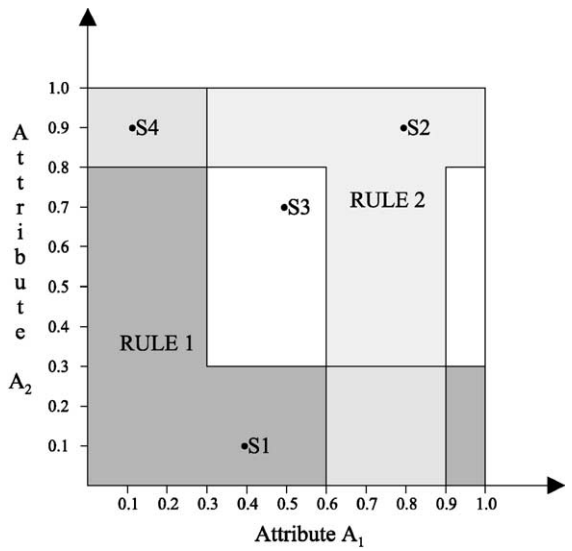


Fig. 2. More examples of assignment of indeterminate cases to only one class.

which computes the distance of a sample from the frontier of the rule.

### 2.1. Encoding

The individuals are rules encoded as tree structures for which limits on the tree depth can be specified. The tree nodes are either functions or terminals. The function set consists of the logical connectives {AND,OR,NOT}, of the relational operators $\{<, \leq, =, \geq, >\}$ and of two operators IN and OUT. The two latter are conceived to efficiently treat an internal and an external interval within the domain of the predicting attributes $A_i$, respectively. They have arity 3, to denote an attribute $A_i$ and the two constants indicating the extremes of the related interval. The terminals are simply $A_i$ or values in the domain of $A_i$.

The individuals in the initial population are randomly created by selecting from the function or terminal sets. Even though the selection is random, during the tree construction some restrictions have been imposed. In particular, the root node can be only a relational operator with a higher probability for AND and OR and with a lower probability for NOT. The internal nodes can be any function. However, if the function is a relation operator, its first son is always an attribute $A_i$ while the other is with equal probability either an attribute or a constant.

Moreover, a so-called half-and-half method has been chosen to create these random initial structures. This is a compromise between the full and the grow methods which are chosen 50% of times each. The full method generates only full trees, that is the tree path length from any terminal node to the root of the tree is the same. The grow, starting from a node chosen as root, recursively calls itself to produce child trees for any node which needs them. Naturally when the initial structures reach the maximum allowed depth all further nodes are restricted to be terminals. This method has been made because it allows to create unbalanced trees with a variety of their shapes and sizes.

### 2.2. Genetic operators

The new elements in the population are generated by means of three operators: *crossover*, *copy* and *mutation*.

#### 2.2.1. Crossover

Two parent individuals are selected and a subtree is picked on each one. Then crossover swaps the nodes and their relative subtrees from one parent to the other. This operator must ensure the respect of the depth limits. If a condition is violated the too-large offspring is simply replaced by one of the parents. There are other parameters that specify the frequency with which internal or external points are selected as crossover points.

#### 2.2.2. Copy

The copy operator simply chooses an individual in the current population and copies it without changes into the new population.

#### 2.2.3. Mutation

The mutation operator can be applied to either a function node or a terminal node. A node in the tree is randomly selected. If the chosen node is a terminal it is simply replaced by another terminal. If it is a function and *point mutation*is to be performed, it is replaced by a new function with the same arity. If, instead, *tree mutation* is to be carried out, a new function node (not necessarily with the same arity) is chosen, and the original node together with its relative subtree is substituted by a new randomly generated subtree. A depth ramp is used to set bounds on size when generating the replacement subtree. Naturally it is to check that this replacement does not violate the depth limit. If this happens mutation just reproduces the original tree into the new generation. Further parameters specify the probability with which internal or external points are selected as mutation points.

### 2.3. Fitness function

The population is constituted by the possible class descriptions, that is to say sets of conditions on attributes of the objects to classify. In these cases it is possible to use statistical fitness functions [12,23]. Let us introduce some definitions to formalize the fitness function $f_c$ which assigns a numerical value indicating its correctness to any description $d$ in the description space $D$.

To each description $d$ in $D$ it corresponds a subset of the training set $S$ denoted with $\sigma_D(S)$, i.e. the set of points where the conditions of a rule are satisfied. The correctness depends on the size of $\sigma_D(S)$, covered by the description, the size of the class $C$ representing the points where the prediction of the rule is true and the size of their overlapping region $\sigma_D(S) \cap C$. Moreover, if we denote with $\sigma_{D'}(S)$ the set of the points in the database in which the conditions are not satisfied and $C'$ the set of the points in which the prediction of the rule is false, the simplest fitness function can be devised as follows:

$$f_c = |\sigma_D(S) \cap C| + |\sigma_{D'}(S) \cap C'| - (|\sigma_{D'}(S) \cap C| + |\sigma_D(S) \cap C'|) \quad (1)$$

This function can be explained as the difference between the actual number of examples for which the rule classifies properly the belonging or not belonging to the class and the number of examples for which there is an incorrect classification whether the conditions or the prediction are not satisfied.

Besides these statistical factors, the fitness function $f_c$ also contains a further term which takes into account in some way the simplicity of the description. This term is conceived taking in mind Occam's razor [5] "the simpler a description, the more likely it is that it describes some really existing relationships in the database". This concept is incorporated in the function $f_s$ and it is related to the number of nodes ($N_{\text{nodes}}$) and the tree depth (depth) of the encoded rules. Namely, this topological term is:

$$f_s = N_{\text{nodes}} + \text{depth}$$

The total fitness function $F$ considered is:

$$F = (S\_tr - f_c) + \alpha f_s \quad (2)$$

where $S\_tr$ is the number of samples in the training set and $\alpha$ varies in [0.0, 1.0]. The choice of its value affects the form of the rules: the closer this value to 1the lower the depth tree and the number of nodes.

## 3. The database and related work

### 3.1. The databases

All the problems faced are intended as tests to evaluate the effectiveness of the approach proposed. Experiments have been performed on the PROBEN1 benchmark set of real-world problems [24] originated from the UCI Machine Learning Repository [19].

The first database, known as *cancer*, addresses a very important problem in the medical domain, the breast cancer diagnosis. The purpose is to find intelligible rules to classify a tumor as either benign or malignant. It is constituted by 699 examples of which 458 are benign and 241 are malignant examples. Each instance contains 10 integer-valued attributes of which the first is the diagnosis class, while the other nine attributes are related to cell descriptions gathered by microscopic examination [31]. All these attributes have the same range $(1, 10)$ in the original database, but they have been normalized in $(0.1, 1)$ in PROBEN1 [24].

The second database, denoted as *diabetes*, is related to diagnosis of diabetes of the Pima Indians population. It contains 768 instances of which 500 are tested as negative for diabetes and 268 as positive to diabetes. Each example is composed of eight real-valued attributes some representing personal data and others the results of medical examination in addition to the class.

The third database, known as *heartc*, concerns with heart disease. It is composed of 303 instances with 13 attributes related to personal data, subjective patient pain descriptions and results of various medical examinations. The database has 45% patients without disease and the remaining ones with disease.

The fourth dataset, indicated as *horse*, predicts the fate of a horse that has a colic. It contains 364 examples, three classes and 20 attributes denoting results of veterinary examinations to forecast whether the horse will survive, will die or will be euthanized. In 24% of the examples it died, in 64% it survived.

The fifth database, designated as *glass*, classifies glass types. It is constituted by 214 instances, 6 classes and 9 attributes related to chemical analysis of glass sprinters plus the refractive index. The classes represent different glass types and their sizes are 70, 76, 17, 13, 9 and 29 instances.

Finally, the sixth dataset, known as *card*, predicts the approval or non-approval of a credit card to a customer. There are 15 attributes unexplained for confidence reasons, 2 classes and 690 examples. The 44% of the examples represent an approval.

In Table 1 an overview of the specific problems with the number of attributes (*Attr*), the continuous (*Cont*) and discrete (*Dis*) inputs, the number of classes (*Cl*), the total number of examples (*Sam*) and their number in the training (*S_tr*) and test (*S_ts*) sets is outlined.

Table 1
The complexity of the problems

| Problem | *Attr* | *Con* | *Dis* | *Cl* | *Sam* | *S_tr* | *S_ts* |
|---|---|---|---|---|---|---|---|
| *Cancer* | 9 | 0 | 9 | 2 | 699 | 525 | 174 |
| *Diabetes* | 8 | 8 | 0 | 2 | 768 | 576 | 192 |
| *Heartc* | 13 | 6 | 29 | 2 | 303 | 227 | 76 |
| *Horse* | 20 | 14 | 44 | 3 | 364 | 273 | 91 |
| *Glass* | 9 | 9 | 0 | 6 | 214 | 161 | 53 |
| *Card* | 15 | 6 | 45 | 2 | 690 | 518 | 172 |

### 3.2. Related work

In [24] a neural network approach is followed on all of the problems, and it shows very good results, but it has the disadvantage of being lacking of explicit rules.

Sherrah et al. [28,29] propose a system which can perform both feature selection and feature construction but they do not focus on the discovery of comprehensible rules. They apply it to some of the problems making up the PROBEN1 benchmark set.

In [30] a tool composed by several artificial neural networks, the outputs of which are combined either in trimmed or in spread means, is evaluated on several benchmarking testbeds. A preliminary phase of experiments allowed to achieve the most suitable network structure for any problem. Reported results are always better than the ones by Prechelt [24] and Brameier and Banzhaf [4] though in many cases improvements are quite slight with respect to simple multi-layer perceptrons or radial basis functions, at the cost of a more complicate network structure. Also in this case no explicit rules are provided. The results are not shown here due to incompleteness of their problem set.

In [4], instead, a GP method with individuals which are programs represented as variable-length strings composed of simple C instructions is presented. Good results are claimed, comparable to those reported in [24], however no easily interpretable rules are reported.

## 4. The experimental results

### 4.1. Parameter setup and performance measures

To ensure the validity of the results and their reproducibility, the standard PROBEN1 benchmarking rules have been applied. The same operating way

as regards the benchmarking testbeds and rules is followed as in [4,24,28–30].

Each database has been subdivided as suggested in [24] including the first 75% of the examples in the training set and the remaining 25% in the test set. Please note that in [24] one more possible subdivision of database instances is proposed, when three partitions (train, validation and test) are generated. Actually, results in that paper, and in [4] as well, make reference to this latter choice. Based on [24], the two subdivision choices can be compared, provided that results from test sets are taken into account in both cases. For each experiment 30 runs have been performed with the same initial configuration but with a different random seed. The selection mechanism has been the *tournament* selection with a tournament size $\mu = 50$. For the generation method of the initial population a ramp in the range 2–5 of initial depth values has been chosen so as to produce trees with different depths. To avoid wasting time with very large trees, the maximum tree depth has been set equal to 7. Moreover, a depth limit of 2 when generating the replacement subtree of the mutation has been fixed. The fitness function chosen has been the (2). The value of $\alpha$ has been chosen equal to 1.0. This imposes a strong constraint on tree size, giving preference to simple rules. The simplicity is at cost of penalizing larger, more complicate rules even if they are capable of a better 'raw' classification rate. This has been done to provide experts with as short as possible rules which could be immediately interpreted by them.

The evolutionary classification system requires that further control parameters be specified. The common parameter setting used for all the problems is listed in Table 2. This setting does not result from a preliminary tuning phase on any of the problems to be faced.

Regarding performance of the whole system made up of as many rules as the number of classes, we recall that the goal of our genetic tool is to predict whether or not an example belongs to the class under examination. We indicate with *CC* and *UC* the total number of examples correctly and incorrectly classified, respectively. As the classification performance is concerned, the *classification error* (*CE*) denoting the percentage of incorrectly classified examples is evaluated [24]. This value is computed by means of the following formula:

$$CE = \frac{UC}{CC + UC} \times 100$$

Table 2
GP parameter settings

| Parameter | Setting |
|---|---|
| Population size | 2000 |
| Maximum number of generations | 30 |
| Crossover probability | 0.8 |
| Reproduction probability | 0.1 |
| Mutation probability | 0.1 |

### 4.2. Findings

The results of our GP system together with those achieved in [4,24] concerning the same PROBEN1 benchmark databases are shown in Table 3 in terms of the classification error *CE*. In particular, the table reports for each problem the average result ($A_v$) over 30 runs with the related standard deviation ($St_{dev}$).

Before all it is to note that, differently from [30], all of the three different PROBEN1 permutations of each dataset are reported. These partitions differ only in the ordering of the examples. The findings in the table demonstrate the effectiveness of the approach proposed. The comparison shows that our system has in most cases a higher *CE*, nonetheless it is on average more robust in terms of lower standard deviation.

The slightly worse results with respect to [4] are, in our opinion, counterbalanced by the availability of intelligible rules that once interpreted provide human experts with a further investigation tool. Moreover, the findings in [4] have been achieved by using a set of functions which includes arithmetic operations as well. By adding these functions to our set we might have been able to improve our results. In fact 30 runs on the *cancer*1 partition have yielded an average value of $CE = 2.00\%$ on test set, lower than that by Banzhaf. However, this operating way determines complex relationships among attributes. For instance the best rules for the *cancer*1 have resulted:

IF $((A_2 < 0.4)$ AND $((A_1 + A_6) < 0.5))$
THEN *class*1

IF $((A_1 \geq (A_6 \times A_9))$ OR $(A_2 > 0.4))$
THEN *class*2

As it can be seen, their interpretation can result difficult even for the experts. Therefore, we have decided

Table 3
The results of our GP system on the training and on the test sets compared with the NN and the LGP findings taken from [24] and [4], respectively

| Problem | Present work | | | | NN | | LGP | |
|---|---|---|---|---|---|---|---|---|
| | Training *CE* (%) | | Test *CE* (%) | | Test *CE* (%) | | Test *CE* (%) | |
| | $A_v$ | $St_{dev}$ | $A_v$ | $St_{dev}$ | $A_v$ | $St_{dev}$ | $A_v$ | $St_{dev}$ |
| *Cancer*1 | 3.65 | 0.37 | 2.46 | 0.66 | 1.38 | 0.49 | 2.18 | 0.59 |
| *Cancer*2 | 3.38 | 0.32 | 6.08 | 1.12 | 4.77 | 0.94 | 5.72 | 0.66 |
| *Cancer*3 | 3.26 | 0.26 | 4.47 | 0.65 | 3.70 | 0.52 | 4.93 | 0.65 |
| *Diabetes*1 | 23.41 | 1.43 | 24.84 | 1.30 | 24.10 | 1.91 | 23.96 | 1.41 |
| *Diabetes*2 | 22.71 | 0.75 | 30.36 | 1.15 | 26.42 | 2.26 | 27.85 | 1.49 |
| *Diabetes*3 | 24.30 | 1.01 | 26.09 | 0.29 | 22.59 | 2.23 | 23.09 | 1.27 |
| *Heartc*1 | 13.59 | 2.03 | 21.99 | 2.61 | 20.82 | 1.47 | 21.12 | 2.02 |
| *Heartc*2 | 18.59 | 1.37 | 7.19 | 2.96 | 5.13 | 1.63 | 7.31 | 3.31 |
| *Heartc*3 | 14.90 | 0.77 | 15.19 | 1.90 | 15.40 | 3.20 | 13.98 | 2.03 |
| *Horse*1 | 32.08 | 2.31 | 33.62 | 3.74 | 29.19 | 2.62 | 30.55 | 2.24 |
| *Horse*2 | 28.71 | 2.54 | 40.87 | 2.12 | 35.86 | 2.46 | 36.12 | 1.95 |
| *Horse*3 | 29.88 | 1.67 | 37.90 | 1.73 | 34.16 | 2.32 | 35.44 | 1.77 |
| *Glass*1 | 38.25 | 2.73 | 40.92 | 2.81 | 32.70 | 5.34 | – | – |
| *Glass*2 | 36.63 | 3.25 | 43.39 | 4.26 | 55.57 | 3.70 | – | – |
| *Glass*3 | 35.39 | 3.28 | 42.63 | 5.42 | 58.40 | 7.82 | – | – |
| *Card*1 | 13.93 | 0.49 | 14.93 | 0.39 | 14.05 | 1.03 | – | – |
| *Card*2 | 12.35 | 0.0 | 20.81 | 0.36 | 18.91 | 0.86 | – | – |
| *Card*3 | 13.31 | 1.20 | 15.57 | 1.41 | 18.84 | 1.19 | – | – |

not to make use of the arithmetic functions. Our results in Table 3 have been obtained without appealing to these operators.

In the following some rules are shown with reference to the databases they are extracted from. As concerns the *cancer* dataset we report the two rules obtained in the best run for the first and the second class for all of the three partitions, respectively:

IF $(((A_2 \leq 0.5)$ AND $(A_6 \leq 0.6))$ AND $((A_1 < 0.5)$
OR $(A_6 = A_9)))$
THEN *class*1

IF $((A_1 \geq 0.6)$ OR $(A_2 > 0.3)$ OR $(A_6 > 0.6))$
THEN *class*2

IF $(((A_1 \leq 0.6)$ AND $(A_2 \leq 0.6))$ AND $(A_6 \leq 0.5))$
THEN *class*1

IF $((A_2 > 0.3)$ OR $(A_6 > 0.5))$
THEN *class*2

IF $(((A_2 \leq 0.3)$ AND $(A_6 \leq 0.2))$ OR $(A_2 < 0.2))$
THEN *class*1

IF $((A_1 \geq 0.7)$ OR $(A_2 \geq 0.7)$ OR $(A_6 > 0.5))$
THEN *class*2

It is to note that the same two attributes $A_2$ (*cell size*) and $A_6$ (*bare nuclei*) are present in all of the six rules. For $A_6$ a lower value is associated to *class*1 and a higher one to *class*2. Furthermore, the attribute $A_1$ (*clump thickness*) is present in four out of the six rules. Therefore, they seem to be highly discriminating for a successful diagnosis. Also in [29] three parameters were indicated as measurements, but it was not reported which those three parameters were.

For the *diabetes*2 the rules for the first and the second class are:

IF $((0.36596 \leq A_8 \leq 0.59689)$ OR $(A_2 > 0.77046)$
OR $(A_6 \geq 0.70693))$
THEN *class*1

IF $((0.31845 \leq A_2 \leq 0.61811)$ OR $((A_2 < 0.77126)$
AND $(A_6 < 0.44703)))$
THEN *class*2

Both rules make use of the IN operator showing its usefulness. By considering also the best rules found on *diabetes*1and *diabetes*3 (not reported) we have ascertained that the parameter $A_2$ (*plasma glucose concentration*) is present in all of the six cases (lower $\Rightarrow$ *class*2, higher $\Rightarrow$ *class*1), while the parameter $A_6$ (*body mass index*) is present in five rules (lower $\Rightarrow$ *class*2, higher $\Rightarrow$ *class*1). Thus, they look important for a proper classification. The same parameters have been found useful in [28]. This is a further hint of the effectiveness of our system.

For the *glass*1 dataset very interesting rules have emerged. The best rule for the first class is:

IF $((A_3 \geq 0.61695)$ AND $(A_4 \leq 0.34491)$
AND $(A_9 < A_7))$
THEN *class*1

As it can be seen the rule exploits first-order logic by taking into account also a relation between attributes. As a further example, for the second class we have found a rule containing the operator OUT:

IF $(\mathrm{OUT}(A_7, 0.26065, 0.65797)$
AND $(A_7 \geq 0.16718))$
THEN *class*2

which shows the clear dependence of this class on *calcium*. The above rule is equivalent to:

IF $((0.16718 \leq A_7 \leq 0.26065)$ OR $(A_7 \geq 0.65797))$
THEN *class*2

which clearly demonstrates the system ability to determine more than one significant interval for an attribute.

Our results on the test set outperform those achieved in [24] as *glass*2 and *glass*3 are concerned. In fact, the reported values on the test set are $A_\mathrm{v} = 55.57$ with $St_\mathrm{dev} = 3.70$ on *glass*2 and $A_\mathrm{v} = 58.40$ with $St_\mathrm{dev} = 7,82$ on *glass*3.

For the database *card*2 we have on the test set better results than those obtained in [29] both in terms of raw *CE* and EPrep *CE* equal to 33.526 and 17.341, respectively. The rules found for the first and the second class are:

IF $((I_9 = 1)$ OR $(I_{42} = 1))$
THEN *class*1

IF $((I_{42} \leq I_{29})$ AND $(I_9 = 0))$
THEN *class*2

where $I_i$, differently from the previous cases, represent the PROBEN1 database inputs derived from the original problem attributes. By taking into account the best rules for the three permutations of the *card* problem we note that the parameters $I_9$ and $I_{42}$ (corresponding to binary variables) are always present and thus important for the discrimination.

Finally, on the *heartc* problem we have found that parameter $A_{30}$ (*number of major vessels colored by fluoroscopy*, lower $\Rightarrow$ *class*1, higher $\Rightarrow$ *class*2) always appears, and parameters $A_{32}$ (*normal thal*, lower $\Rightarrow$ *class*2, higher $\Rightarrow$ *class*1) and $A_6$ (*asymptomatic chest pain*, lower $\Rightarrow$ *class*1, higher $\Rightarrow$ *class*2) are present in five out of the six best rules.

As regards the improvement over time of the rules our system finds, as an example we show in Fig. 3 the evolution for one run concerning the malignant cases (*class*2) of *cancer*1 database. Namely, both the fitness of the best rule (Fig. 3(a)) and the number of cases correctly classified by it (Fig. 3(b)) are reported as a function of the number of generations. Furthermore, the corresponding rules are shown in Table 4. It can be noted that most of the evolution takes place during the first half of the allowed maximum number of generations. If we take into account that the number of cases in the training set is 525 (see Table 1), it is clear that the first generation already contains a quite satisfactory rule capturing correctly most of cases (488, i.e. 92.95%). This premature stagnation might be due to two causes. The former is the high population size chosen, yielding very good individuals in early generations. By using a lower one, evolutions with a higher number of improving generations have been obtained. Final rules, nonetheless, are the same obtained by using a wider population. The latter cause is the constraint of rule compactness we have imposed. In fact, by eliminating it, evolution more frequently finds new improving solutions, though these become larger and larger in size, so being not easily intelligible by humans. The evolution of the best rule shows that the system takes into account many of the parameters, in fact only $A_5$ and $A_7$ never appear in Table 4. It is to be noted that attributes $A_1$ and $A_6$ are shown important since generation 0, and they take part in the final rule; $A_1$ momentarily disappears at generations

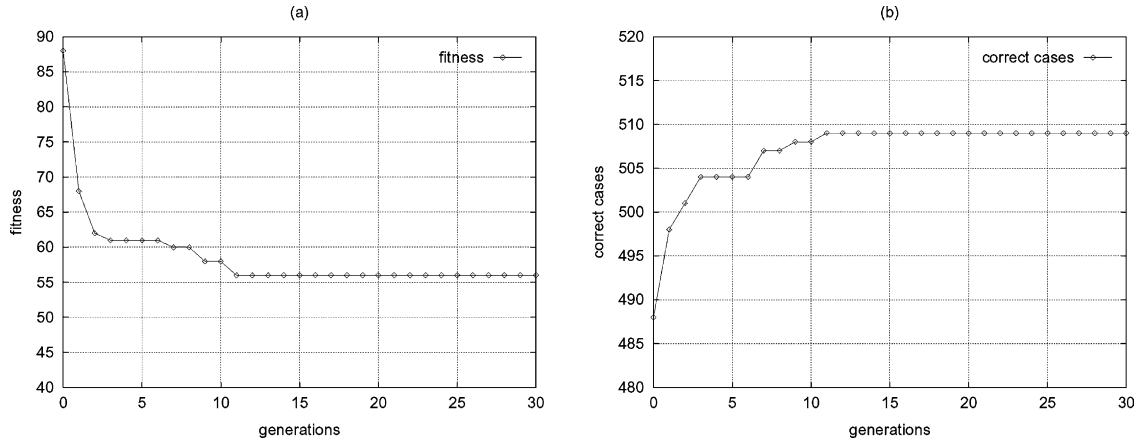


Fig. 3. Evolution of the best rule for malignant cases (*class*2) in *cancer*1 database.

Table 4
The evolution of the best rule related to Fig. 3

| | |
|---|---|
| Generation 0, best fitness = 88, correct cases = 488 | IF (((A1 > 0.7) OR (A6 > 0.7)) AND (A1 > 0.2)) THEN *class*2 |
| Generation 1, best fitness = 68, correct cases = 498 | IF (((A2 > 0.3) OR (A6 > 0.4)) AND (A3 > 0.2)) THEN *class*2 |
| Generation 2, best fitness = 62, correct cases = 501 | IF (((A2 > 0.3) OR (A6 > 0.4)) AND (A3 > 0.1)) THEN *class*2 |
| Generation 3, best fitness = 61, correct cases = 504 | IF (((A1 > 0.7) OR (A2 > 0.3) OR (A6 > 0.7)) AND (A3 > 0.1)) THEN *class*2 |
| Generation 7, best fitness = 60, correct cases = 507 | IF (((A1 > 0.7) OR (A2 > 0.3) OR (A6 > 0.7) OR (A8 > 0.7)) AND (A3 > 0.1)) THEN *class*2 |
| Generation 9, best fitness = 58, correct cases = 5087 | IF (((A1 > 0.7) OR (A4 > 0.3) OR (A6 > 0.7) OR (A8 > 0.7)) AND (A3 > 0.1)) THEN *class*2 |
| Generation 11, best fitness = 56, correct cases = 509 | IF (((A1 > 0.7) OR (A4 > 0.3) OR (A6 > 0.5) OR (A8 > 0.7)) AND (A3 > 0.1)) THEN *class*2 |

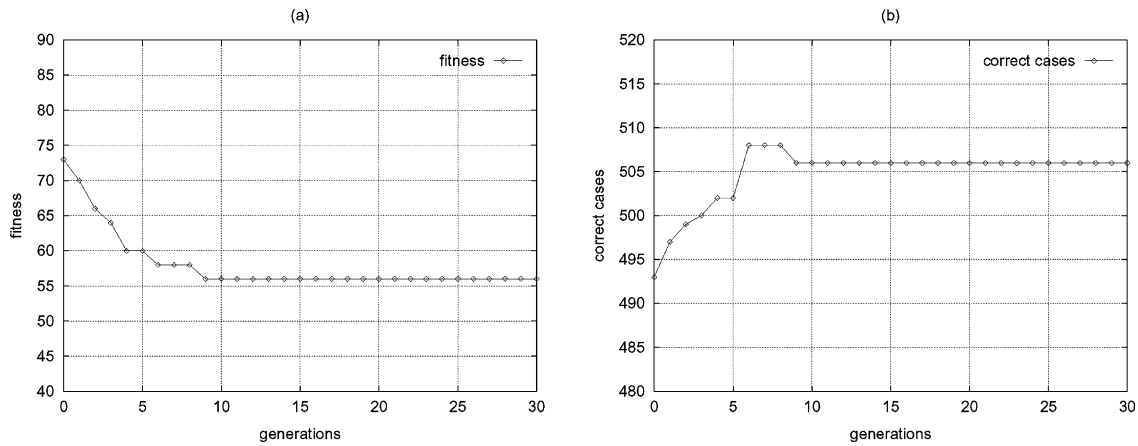


Fig. 4. Another evolution of the best rule for malignant cases (*class*2) in *cancer*1 database.

Table 5
The evolution of the best rule related to Fig. 4

| | |
|---|---|
| Generation 0, best fitness = 73, correct cases = 493 | IF $((A_2 > 0.2)$ AND $(A_3 > 0.2))$ THEN *class*2 |
| Generation 1, best fitness = 70, correct cases = 497 | IF $(((A_1 > 0.6)$ OR $(A_3 > 0.3))$ AND $(A_2 > 0.1))$ THEN *class*2 |
| Generation 2, best fitness = 66, correct cases = 499 | IF $(((A_5 > 0.1)$ OR $(A_6 > 0.1))$ AND $(A_3 > 0.1))$ THEN *class*2 |
| Generation 3, best fitness = 64, correct cases = 500 | IF $(((A_1 > 0.6)$ OR $(A_3 > 0.3))$ AND $(A_7 > 0.2))$ THEN *class*2 |
| Generation 4, best fitness = 60, correct cases = 502 | IF $(((A_5 > 0.3)$ OR $(A_6 > 0.2))$ AND $(A_3 > 0.2))$ THEN *class*2 |
| Generation 6, best fitness = 58, correct cases = 508 | IF $((((A_2 > 0.3)$ OR $((A_3 > 0.2)$ AND $(A_6 > 0.2)))$ OR $(A_1 > 0.6))$ AND $(A_7 > 0.2))$ THEN *class*2 |
| Generation 8, best fitness = 57, correct cases = 508 | IF $((((A_2 > 0.3)$ OR $(A_1 > 0.6))$ OR $((A_3 > 0.2)$ AND $(A_6 > 0.2)))$ AND $(A_7 > 0.2))$ THEN *class*2 |
| Generation 9, best fitness = 56, correct cases = 506 | IF $(((A_1 > 0.5)$ OR $(A_3 > 0.2))$ AND $((A_2 > 0.3)$ OR $(A_6 > 0.2)))$ THEN *class*2 |

1 and 2, whereas $A_6$ is always present in the rules. Furthermore, already at generation 1 the system finds out the importance of attribute $A_3$, which is highly discriminant in the final rule found in this run.

Another interesting situation related to rule evolution is shown in Fig. 4. In it we make reference to another run on *cancer*1 database. Also in this case the search for the best *class*2 (malignant) rule is involved and, as in the previous example, both the fitness of the best individual (Fig. 4(a)) and the number of cases correctly classified by it (Fig. 4(b)) are sketched as a function of the number of generations. We can see two interesting transitions in Table 5, where the best rules found during the evolution are depicted. The first takes

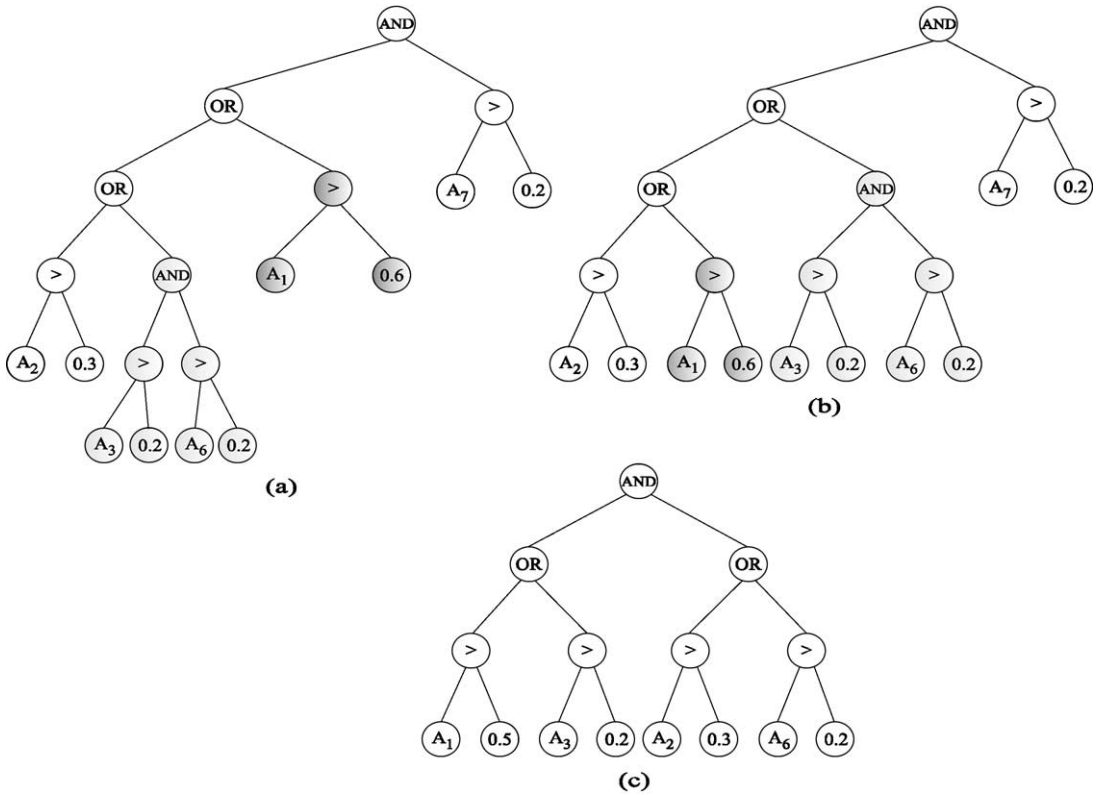


Fig. 5. Example of the topological term effect on the best rule evolution.

place at generation 8: a new best rule is found which results in the same rule as the previous best-so-far found at generation 6, so there is no difference between them from the point of view of classification quality. Nonetheless the new related tree is more compact, because it has the same number of nodes distributed over four levels (see Fig. 5(a)), rather than over five as it is the case for the previous best rule (Fig. 5(b)). This new individual has been achieved by means of a crossover operation performed on two copies of the same best-so-far tree, resulting in the two subtrees drawn in grey in Fig. 5(a) being swapped. The second situation of interest can be noted at generation 9, when the fitness of the best rule improves to the detriment of the number of correctly classified cases. This is due to the fact that a slightly worse-classifying, yet more compact rule emerges. What is happened is that, due to the value of 1.0 used for $\alpha$, the topological feature has outperformed the raw classification capability: in fact, the former best rule has 19 nodes and 4 levels, whereas the latter has only 15 nodes arranged over 3 levels (see Fig. 5(c)). This results in a topological term better by 5, while the classification term is worse by only 4. The fitness value for the rule at generation 8 is then $525 - 508 + 17 + (1.0 \times 23) = 57$, while at generation 9 the rule has a fitness equal to $525 - 506 + 19 + (1.0 \times 18) = 56$.

## 5. Conclusions and future works

In this paper we have presented a GP tool for explicit rule extraction. The system has been tested on publicly available databases. We have compared our system with neural network-based approaches and with other GP-based techniques. Experimental results have demonstrated the effectiveness of the approach proposed in providing the user with compact and comprehensible classification rules, and its robustness in terms of low standard deviation.

Future work will include the application of the system proposed to other real-world datasets in order to further validate the promising results reported in the present paper, as for example in computational archaeology. Moreover, another interesting task to face will be the unsupervised data mining in which the goal is to discover rules that predict a value of a goal attribute which, unlike classification, is not chosen a priori. To this aim evolutionary techniques exploiting niching methods [18] able to maintain different rules (niches) at the same time are to be analyzed.

## Acknowledgements

The authors wish to thank Mr. Simone Guarino for drawing some of the figures present in this paper.

## References

[1] C. Anglano, A. Giordana, G. Lo Bello, L. Saitta, A network genetic algorithm for concept learning, in: Proceedings of the ICGA'97, Morgan Kaufmann, San Francisco, 1997, pp. 434–441.
[2] S. Augier, G. Venturini, Y. Kodratoff, Learning first order logic rules with a genetic algorithm, in: Proceedings of the First International Conference on Knowledge Discovery & Data Mining, AAAI Press, Menlo Park, CA, 1995, pp. 21–26.
[3] C.C. Bojarczuk, H.S. Lopes, A.A. Freitas, Discovering comprehensible classification rules using genetic programming: a case study in a medical domain, in: Proceedings of the GECCO'99, Morgan Kaufmann, San Francisco, 1999, pp. 953–958.
[4] M. Brameier, W. Banzhaf, A comparison of linear genetic programming and neural networks, IEEE Trans. on Evolutionary Computation 5 (1) (2001) 17–26.
[5] W. Derkse, On simplicity and Elegance, Eburon, Delft, 1993.
[6] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, From data mining to knowledge discovery: an overview, in: U.M. Fayyad, et al. (Eds.), Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1996, pp. 1–34.
[7] D.B. Fogel, E.C. Wasson, E.M. Boughton, V.W. Porto, P.J. Angeline, Linear and neural models for classifying breast masses, IEEE Trans. on Medical Imaging 17 (3) (1998) 485–488.
[8] A.A. Freitas, A genetic programming framework for two data mining tasks: classification and generalized rule induction, in: Proceedings of the Second Annual Conference on Genetic Programming, Morgan Kaufmann, San Francisco, 1997, pp. 96–101.
[9] G. Fung, O.L. Mangasarian, Semi-supervised support vector machines for unlabeled data classification, Technical Report, Department of Computer Science, University of Wisconsin, October 1999.
[10] A. Giordana, L. Saitta, F. Zini, Learning disjunctive concepts by means of genetic algorithms, in: Proceedings of the 11th International Conference on Machine Learning, 1994, pp. 96–104.
[11] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[12] M. Holsheimer, A. Siebes, Data mining: the search for knowledge in databases, Technical Report, CS-R9406, CWI, Amsterdam, The Netherlands, 1994.

[13] M.S. Hung, M. Hu, M. Shanker, Estimating breast cancer risks using neural network, International Journal of Operational Research 52 (2001) 1–10.
[14] H. Ishibuchi, K. Nozaki, N. Yamamoto, H. Tanaka, Selecting fuzzy if-then rules for classification problems using genetic algorithms, IEEE Trans. Fuzzy Systems 3 (3) (1995) 260–270.
[15] H.E. Johnson, R.J. Gilbert, M.K. Winson, R. Goodacre, A.R. Smith, J.J. Rowland, M.A. Hall, D.B. Kell, Explanatory analysis of the metabolome using genetic programming of simple, interpretable rules, Genetic Programming and Evolvable Machines 1 (3) (2000) 243–258.
[16] J.R. Koza, Genetic Programming: On Programming Computers by means of Natural Selection and Genetics, MIT Press, Cambridge, MA, 1992.
[17] C.H. Lee, D.G. Shin, A multistrategy approach to classification learning in databases, Data Knowledge Eng. 31 (1999) 67–93.
[18] S.W. Mahfoud, Niching methods for genetic algorithms, Doctoral Dissertation, Illinois Genetic Algorithms Lab. Rep. 95001, University of Illinois, USA, 1995.
[19] P. Murphy, D.W. Aha, UCI repository of machine learning databases [WWW page], Department of Information and Computer Science, University of California, Irvine, CA, 1992, Available from: http://www.ics.uci.edu/mlearn/MLRepository.html.
[20] P.S. Ngan, M.L. Wong, K.S. Leung, Using grammar based genetic programming for data mining of medical knowledge, in: Proceedings of the 3rd Annual Conference on Genetic Programming, Morgan Kaufmann, San Francisco, 1998, pp. 304–312.
[21] E. Noda, A.A. Freitas, H.S. Lopes, Discovering interesting prediction rules with a genetic algorithm, in: Proceedings of the Congress on Evolutionary Computation, IEEE Press, Piscataway, NJ, 1999.
[22] C.A. Peña, M. Sipper, Designing breast cancer diagnosis systems via a hybrid fuzzy-genetic methodology, in: Proceedings of the IEEE Int. Fuzzy Systems Conference, Vol. 1, IEEE Press, Piscataway, NJ, 1999, pp. 135–139.
[23] G. Piatesky-Shapiro, Discovery, analysis and presentation of strong rules, in: G. Piatesky-Shapiro, W. Frawley (Eds.), Knowledge Discovery in Databases, AAAI Press, Menlo Park, CA, 1991.
[24] L. Prechelt, PROBEN1—a set of neural network benchmark problems and benchmarking rules, Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, Germany, 1994.
[25] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.
[26] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Francisco, 1993.
[27] R. Setiono, L.C.K. Hui, Use of a quasi-Newton method in a feedforward neural networks construction algorithm, IEEE Trans. Neural Networks 6 (1) (1995) 273–277.
[28] J. Sherrah, R.E. Bogner, A. Bouzerdoum, Automatic selection of features for classification using genetic programming, in: Proceedings of the IEEE Australian and New Zealand Conference, IEEE Press, Piscataway, NJ, 1996.
[29] J. Sherrah, R.E. Bogner, A. Bouzerdoum, The evolutionary pre-processor: automatic feature extraction for supervised classification using genetic programming, in: Proceedings of the 2nd Annual Genetic Programming Conference, Morgan Kaufmann, San Francisco, 1997.
[30] K. Tumer, J. Ghosh, Classifier combining through trimmed means and order statistics, in: Proceedings of the IEEE International Conference on Neural Networks, IEEE Press, Piscataway, NJ, 1998, pp. 757–762.
[31] W.H. Wolberg, O.L. Mangasarian, Multisurface method of pattern separation for medical diagnosis applied to breast cancer cytology, Proc. Natl. Acad. Sci. 87 (1990) 9193–9196.
[32] X. Yao, Y. Liu, A new evolutionary system for evolving artificial neural networks, IEEE Trans. Neural Networks 8 (3) (1997) 694–713.
[33] W. Ziarko, Rough Sets, Fuzzy Sets and Knowledge Discovery, Springer, Berlin, 1994.