GEPCLASS: A Classification Rule Discovery Tool Using Gene Expression Programming

Wagner R. Weinert and Heitor S. Lopes*

Bioinformatics Laboratory, CPGEI
Federal University of Technology – Paraná
Av. 7 de setembro, 3165 80230-901 Curitiba (PR), Brazil
weinert@cpgei.cefetpr.br, hslopes@pesquisador.cnpq.br

Abstract. This work describes the use of a recently proposed technique – gene expression programming – for knowledge discovery in the data mining task of data classification. We propose a new method for rule encoding and genetic operators that preserve rule integrity, and implemented a system, named GEPCLASS. Due to its encoding scheme, the system allows the automatic discovery of flexible rules, better fitted to data. The performance of GEPCLASS was compared with two genetic programming systems and with C4.5, over four data sets in a five-fold cross-validation procedure. The predictive accuracy for the methods compared were similar, but the computational effort needed by GEPCLASS was significantly smaller than the other. GEPCLASS was able to find simple and accurate rules as it can handle continuous and categorical attributes.

1 Introduction

Gene Expression Programming - GEP [3] is a novel evolutionary algorithm, recently proposed, that includes characteristics from Genetic Algorithms - GA and Genetic Programming - GP. The main difference from GEP to GA and GP is how individuals are encoded.

In GAs individuals are usually a fixed-size linear string of bits (chromosomes) whereas in GP, individuals are non-linear entities of different size and shapes, usually in the form of trees. In GEP, on the other hand, individuals are encoded as fixed-size strings of symbols (chromosome) that, in turn, are expressed as non-linear entities of different size and shapes known as "expression trees". Since GEP has emerged recently, few applications have been published to date [8]. In this work we apply GEP to the data mining task of classification, where it is aimed to find comprehensible rules capable of modelling a given set of data. The objective of data classification is to predict the value of a goal attribute, giving a set of predicting attributes. Usually, rules are represented in the form IF $<$ *antecedent* $>$ THEN $<$ *consequent* $>$, where "antecedent" is a logical

* This work was partially supported by a research grant from the Brazilian National Research Council – CNPQ (305720/04-0).

X. Li, O.R. Zaiane, and Z. Li (Eds.): ADMA 2006, LNAI 4093, pp. 871–880, 2006.

combination of the predicting attributes and their values, and the "consequent" part is the value of the goal attribute (class).

A GEP-based tool, named GEPCLASS (Gene-Expression Programming for CLASSification), was created for this purpose. We describe several modifications on the original GEP algorithm so as to efficiently cope with data classification. The application of GEPCLASS to a number of datasets is reported and we its performance is compared with other published papers in recent literature.

2 Gene Expression Programming

Ferreira [3] proposed a new evolutionary algorithm with linear genotype/non-linear phenotype, denominated Gene Expression Programming, using concepts from both GAs and PG. In GEP, chromosomes are simple, compact, linear and relatively small entities, that are manipulated by means of special genetic operators (replication, mutation, recombination, translocation, etc). Expression trees (ETs) are the phenotypical representation of the chromosome. Selection, the central engine of evolutionary computation paradigms, operates over ETs rather than chromosomes. During the reproduction cycle, chromosomes, not ETs, are generated, modified and transmitted to the next generations. In [3], PEG is presented using individuals with only one chromosome and, henceforth, individual and chromosome are used as synonymous. Although it would be possible to use multiple-chromosome individuals, we used that same approach.

Similarly to other evolutionary algorithms, GEP starts with an initial population, either created at random or using some previous knowledge about the problem. Next, chromosomes are expressed into ETs that, in turn, are evaluated according to the specific meaning of the problem, yielding a fitness measure. If a stop criteria is not met, the best individual(s) is(are) kept and the rest are submitted to a fitness-based selection procedure. Selected individuals undergo modifications by means of genetic operators leading to a new generation of individuals. The whole process is repeated until a stopping criterion is satisfied.

2.1 GEP Encoding

The genome in GEP is a single chromosome which, in turn, is composed by one or more genes (that is, multigenic), as in nature. Every gene is divided into two parts: head and tail. The size of the head (h) is determined by the user, and the size of the tail (t) is computed as: $t = h(n - 1) + 1$, considering n the largest arity found in the function set for the particular problem.

The phenotypical representation of a genome is the set of sub-trees (ETs), each one being expressed by a gene, linked together by means of a linking function. This function is user-defined and connects the roots of all sub-trees. For instance, this linking function can be sum or multiplication, for symbolic regression problems, or logical AND or OR for classification problems. Like biological genes, only part of it is really expressed as an ET.

In the same way as GP, GEP also uses a function set and a terminal set as building blocks of possible solutions. Considering that genes have two parts,

namely head and tail, some restrictions apply: in the tail part only terminal elements can occur, whereas in the head of a gene, both terminals and functions are permitted (except for the first position, where only functions are allowed).

2.2 Selection Method and Genetic Operators

The main selection method in GEP is the well-known roulette-wheel, a popular procedure in the early implementations of GA. This method favors the fittest individuals of the population, whose chances to be selected are proportional to their relative fitness. During run, individuals that will undergo the transposition and crossover operators (see below) are randomly selected.

GEP also implements a simple elitist mechanism, where the best individual of a generation is copied to the next, by means of a cloning operator. This procedure guarantees that the best individual found throughout generations is kept.

In the original work of Ferreira [3] , several genetic operators were defined.

The mutation operator works similarly as in GP and GA and aims to introduce new genetic material in the current population so as to increase genetic diversity. Due to the particular characteristics of the encoding in GEP,some integrity rules must be obeyed in order to avoid syntactically invalid individuals (see section 3.1).

GEP uses one-point or two-point crossovers, just like GA. The second type is somewhat more interesting since it can turn on and off noncoding regions within the chromosome. Also, a kind of uniform crossover was implemented, and is known as genic recombination. This operator randomly chooses genes of same position in two parent chromosomes to form two new offsprings.

There are three transposition operators: IS (insertion sequence), RIS (root IS) and genic. An IS element is a variable-size sequence of alelles extracted from a random starting point within the genome (even if the genome was composed by several chromosomes). Another position within the genome is chosen and it will be the place where the element will be inserted. This target site must be within the head part of a gene and cannot be the first alelle (gene root). The IS element is sequentially inserted in the target site, shifting all alelles from this point forth. The same number of alelles inserted are deleted from the gene head, from the end backwards. This operator simulates the transposition found in the evolution of biological genomes. RIS is similar to the IS transposition, except that the insertion sequence must have a function as first alelle and the target point must be also the first alelle of a gene (root). Finally, genic transposition swaps genes within a chromosome.

3 Methodology

In this section we describe all modifications in the original GEP algorithm to develop the GEPCLASS system, specifically designed for data classification. Some of these changes were also used successfully in another GEP-based system [8].

3.1 Rule Encoding

Most literature in data-mining use rules in the form *if A then C*, for classifying data. The antecedent A is a set of conditions to be met and the consequent C is the predicted class. Conditions are t-uples in the form $\{A_i\ Op\ V_{ij}\}$, where A_i is the i-th attribute, Op is a relational operator ($=$, $\neq$, $>$ or $<$), and V_{ij} is the j-th value belonging to the domain of attribute A_i. To combine several possible conditions in a rule, logical operators (*and*, *or*, *not*) are used. The consequent of a rule is simply a condition in the form $\{M_i = V_{ij}\}$, where M_i is one of the possible goal attributes, and V_{ij} is a possible value for this goal attribute.

Implementing data classification using evolutionary algorithms requires defining, a priori, whether an individual represents a single rule (Michigan approach) or a complete solution composed by a set of rules (Pittsburg approach) [4]. GEPCLASS can implement both approaches, either by an explicit decision of the user, or allowing the algorithm decide by itself which one is more adequate for a given classification task, during the evolutionary process. For instance, user can define that an individual will generate multiple rules, using more than one gene per chromosome and a logical *or* as linking function between genes. The use of GEP for data classification requires tight restrictions in the individuals' encoding, so as to avoid syntactically invalid individuals. In our approach, we define a closure property that states that any ET root must be a logical function. All logical functions have as offspring nodes other logical or relational functions. These, in turn, always have as offspring nodes attributes and correlated values.

To assure that the closure property of the encoding will be always met, we proposed in GEPCLASS some changes in the original encoding of GEP, at the phenotypical level, and we also defined filling rules for the chromosome. First, a crucial modification in the structure of the chromosome is regarding lengths of head and tail. The tail size (t) was defined before for symbolic regression problems [8], but for data classification, we propose a new way to compute it, as follows: $t = int([h.(n-1)+1]/2)$, where $int()$ returns the integer part of the argument. This new approach is justified considering the fact that every terminal element is attached to a set of possible values (domain), for instance $(a > 10)$, not to another attribute or constant, as in symbolic regression, for instance $(a + b)$. Therefore, a compact representation for both terminals and their values reduces half the size of the tail length in a gene. In words, at the implementation level (only), relational operators have arity 1.

Figure 1 presents a 2-genes chromosome with different lengths for heads and tails. Upward arrows show the points delimiting the coding sequence of each gene. The corresponding ET of the chromosome is presented. Notice that the ET maintains the original characteristics of GEP and, at the same time, guarantees rule integrity (syntactically valid rules). A filling rule was defined in GEPCLASS: the tail part of a gene always has only terminals (but head can have terminals and functions). An extension of this rule is the implicit precedence between functions (logical and relational) and terminals. In practice, this is accomplished by means of a constrained syntax, inspired in [2]. There is, the encoding must guarantee that a given operator will receive valid operands (terminals), and, therefore,

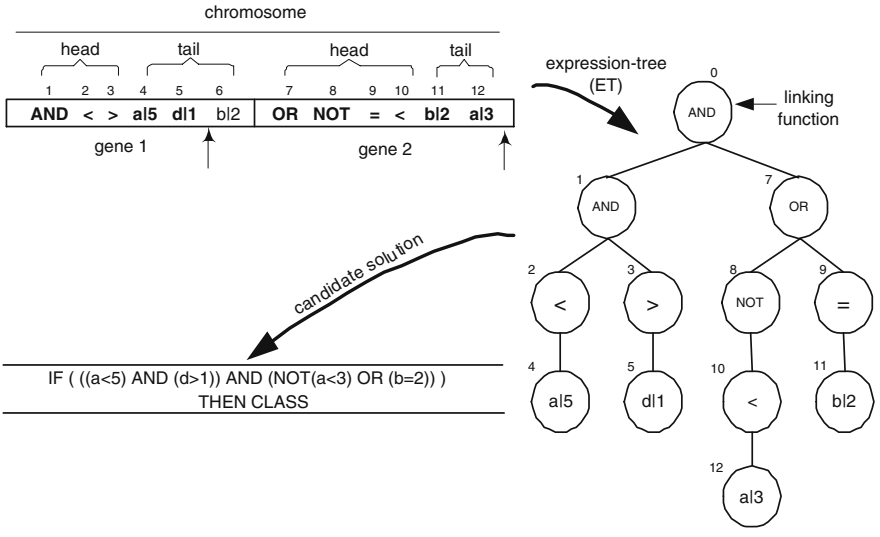

Fig. 1. Example of chromosome structure, ET and corresponding rule in GEPCLASS

will give a valid output to the parent node. Figure 1 clearly exemplifies this accomplishment: at root level (node 0) one can find a logical function (AND) linking two sub-trees; in the second level (nodes 1, 7 and 8) it is found logical operators (*and*, *or*, *not*); in next level (nodes 2, 3, 9 and 10) there are relational operators ($<$, $>$, $=$, $<$); finally, at leaf level (nodes 4, 5, 11 and 12) there are terminals and respective values.Notice that the arity of relational operators is not changed, only the representation in the tree.

A consequence of the filling rule is regards to the nature of the attributes. In a data set, one can have both continuous and categorical (nominal) attributes. If a given attribute is categorical, GEPCLASS uses only $=$ or $\neq$ as relational operators. Otherwise, if the attribute is continuous or ordered categorical (values mapped in a predetermined scale), all relational operators can be used. In GEPCLASS, any terminal can appear once, many times or none in a rule. This flexible characteristic allows one to find rules within a range, for instance, "$A_i > 10$) AND ($A_i < 15$)". On the other hand, this approach allows inconsistencies, such that: "$A_i = 10$) AND ($A_i = 15$)". Results obtained up to now (see section 5) have demonstrated that such characteristic is not a serious problem.

3.2 Chromosome Structure, Fitness Function and Genetic Operators

In the original GEP, determining an adequate length for the head of each gene is an open problem. It was stated that more complex problems may require lengthy gene heads, and, therefore, the ideal size is found by trial-and-error.

GEPCLASS uses variable-length chromosomes that can have one or more genes. Genes within a given chromosome are of the same size. Using chromosomes with different lengths in the population can introduce healthy genetic diversity during the search. This approach has also been proved beneficial for symbolic regression problems [8] and is a way to circumvent the open problem of the original GEP in defining adequate lengths for the head of a gene [3]. By default, 50% of the initial population is generated according to a user-defined parameter that defines the maximum and minimum head length of genes, and the rest is randomly generated in the same interval. This procedure was inspired in a similar technique proposed by Koza [6] for GP.

In GEPCLASS we propose two other selection methods over methods originally proposed for GEP . The first method always uses roulette-wheel for selecting individuals regardless of the genetic operator to be used further. This selection method introduces a strong selective pressure that can lead to fast convergence, usually to local maxima in the search space. The second implementation is the stochastic tournament, a successful method frequently used in GA. This strategy is guided by a user-defined parameter (k) that defines the number of individuals that will be randomly selected for a tournament. The individual with highest fitness value will be selected. This method is less aggressive than roulette-wheel and tends to make the algorithms less sensitive to local maxima.

Amongst the many fitness functions proposed before for data mining with evolutionary algorithms [4], we choose to implement in GEPCLASS that proposed by Lopes et al. [7]. This function is the product of two measures: sensibility (Se) and specificity (Sp). Sensitivity measures the fraction of positive instances that will be correctly classified by the system, and it is defined as $Se = tp/(tp + fn)$. Specificity measures the fraction of negative instances that will be classified as such, and it is defined as $Sp = tn/(tn + fp)$. These indicators take into account not only the number of correct classifications, but also the relationship between positive and negative classes. Therefore, the fitness function used has the advantage to maximize both Se and Sp at the same time. Sp and Se, are computed using the number of true-positive (tp), true-negative (tn), false-positive (fp) and false-negative (fn) scores of a rule.

Due to the way rules are encoded in GEPCLASS, most of the original operators needed functional modifications to comply with the closure property, maintaining the hierarchical structure of the ETs after the application of operators.

The mutation operator works in three different levels. When a logical function is selected for mutation, it will be substituted by another logical function. In the same way, when a relational function is selected, it will be replaced by another relational function. Finally, at the leaf level of the ET, when a terminal is selected for mutation, it will be changed by a new attribute and respective random value. There is an exception when the mutation operator is inoperative: when the node selected for mutation is a NOT function, since it is the only function with arity 1 and cannot be substituted by another logical function with higher arity.

The recombination operator works in the same way for chromosomes of different of the same lengths. This operator always swaps the genetic material between the head parts of two genes.

Both the IS and RIS transposition operators work over a single chromosome. IS transposition changes only the tail part of a gene (that is, terminals positions within the gene). Consequently, this operator can move genetic material from one gene to another or within a single gene. On the other hand, in RIS transposition, the donor site is in the tail of a gene, whereas the receptor site is always the first terminal of this same gene.

4 Computational Experiments and Results

Considering that GEP is an extension of GP, we understand that the most fair performance comparison would be GEPCLASS versus a GP-based system for data classification. Therefore, we compared GEPCLASS with a constrained-syntax genetic programming (CSGP) system proposed by [2], over four real-world data sets. In that work there is also a comparison with a "Booleanized" version of genetic programming (BGP) [1] that we reproduced here. The well-known C4.5 decision-tree induction algorithm [9] is often used as the baseline for performance comparisons in data classification literature, and we also included results for this algorithm using the same data. The data sets used for this work were: Ljubljana breast cancer (277/9/2), Wisconsin breast cancer (683/9/2), Dermatology (358/34/6) and Chest pain (138/161/12). Numbers within parenthesis correspond to the number of examples, attributes and classes, respectively. The first three data sets are available at the Machine Learning Repository (*http://www.ics.uci.edu/˜mlearn/*), and the last one was described in [1].

The CSGP used in [2] had the following parameters: maximum tree depth: 15; population size: 500 individuals; stopping criterion: 50 generations; recombination, reproduction and mutation operators probabilities: 95%, 5% and 0%. GEPCLASS used the following parameters: population size: 30 individuals; stopping criterion: 50 generations; number of genes per chromosome: 2; linking function: logical *and*; head size range: 6-10; selection method: stochastic tournament; genetic operators: all those defined by the original GEP [3] with the same probabilities. All results reported in this work were obtained by performing a 5-fold cross-validation procedure [5], and using exactly the same data partitions as in [2]. Table 1 shows the accuracy rate for the four data sets, using C4.5, CSGP, BGP, and GEPCLASS.

Table 1. Comparison of accuracy rates (in %)

Data set	C4.5	BGP	CSGP	GEPCLASS
Ljubljana	71.4 ± 0.60	63.9 ± 5.67	71.8 ± 4.68	68.5 ± 12.73
Wisconsin	94.8 ± 0.06	89.3 ± 4.37	93.5 ± 0.79	93.8 ± 2.89
Dermatology	89.1 ± 0.13	86.2 ± 6.24	96.6 ± 1.14	90.5 ± 11.19
Chest pain	73.2 ± 0.77	78.1 ± 4.85	80.3 ± 3.90	88.9 ± 9.61

Table 2. Best rules found by GEPCLASS for Ljubljana, Dermatology and Wisconsin data sets and respective classes

Data set	class	#nodes	accuracy	Rule
Ljubljana	1	16	83.63%	$IF((\sim (Node\_caps <> 1))\|AND\|$ $(((Age = 6)OR(Deg\_malig <> 2))OR$ $(Tumor\_Size = 3)))THEN(class = 1)$
	2	15	94.54%	$IF(((Irradiat <> 1)OR$ $(Inv\_nodes <> 0))\|AND\|((Node\_caps = 0)$ $OR(Inv\_nodes < 5)))THEN(class = 2)$
Dermatology	1	9	100%	$IF((\sim (Thinning\_suprapapil\_epid = 0))\|AND\|$ $(\sim (Follic\_horn\_plug = 1)))THEN(class = 1)$
	2	12	82.85%	$IF(((Polygonal\_papules < 0)OR$ $(Saw\_tooth\_appearance\_of\_retes < 1))$ $(\sim (Spongiosis < 2)))THEN(class = 2)$
	3	15	100%	$IF(((Polygonal\_papules <> 0)OR$ $(Knee\_and\_elbow\_involvement <> 2))$ $\|AND\|((Vacuolisation\_damage\_basal\_layer > 0)$ $OR(Oral\_mucosal\_involvemente <> 0)))$ $THEN(class = 3)$
	4	16	100%	$IF((((Koebner\_phenomenon <> 0)OR$ $(Disappearance\_of\_the\_granular\_layer = 1))$ $AND(Band\_like\_infiltrate < 2))\|AND\|$ $(\sim (Elongation\_of\_the\_rete\_ridges > 0)))$ $THEN(class = 4)$
	5	12	100%	$IF(((Oral\_mucosal\_involvemente <> 2)AND$ $(Vacuolisation\_damage\_basal\_layer <> 3))$ $\|AND\|(\sim (Fibrosis\_papillary\_dermis < 1)))$ $THEN(class = 5)$
	6	16	100%	$IF(((Disappearance\_granular\_layer <> 1)$ $AND(Perifollicular\_parakeratosis <> 0))$ $\|AND\|(\sim ((Follicular\_papules = 0)AND$ $(Perifollicular\_parakeratosis < 3))))$ $THEN(class = 6)$
Wisconsin	1	13	97,03%	$IF((\sim (\sim (Bare\_Nuclei < 4)))\|AND\|$ $((Bland\_Chromatin < 2)OR$ $(Uniformity\_of\_Cell\_Size < 4)))$ $THEN(class = 1)$
	2	19	98.51%	$IF((((Bare\_Nuclei > 5)OR$ $(Uniformity\_of\_Cell\_Shape > 3))OR$ $(Bland\_Chromatin = 7))\|AND\|$ $((Single\_Epithelial\_Cell\_Size > 1)OR$ $(Mitoses > 3)))THEN(class = 2)$

In order to show the simplicity of rules found by GEPCLASS, table 2 shows the best results found, over the five runs, for three out of the four data sets and respective classes. The chest pain data set has 12 classes and takes too much space to be reported. In this table it is shown the number of nodes of the best

solution, its accuracy rate and the composed rule itself. In each rule, the "|AND|" is the linking function and "∼" means the logical "NOT".

5 Discussion and Conclusions

In this work, we proposed a gene expression programming system for data classification and we compared its performance in four data sets.

The straight comparison for the accuracy rates between GEPCLASS and all other classifiers shows no statistically significant differences, except for the Chest pain data set comparing with C4.5 (t-test with confidence level 5%).

The standard deviations of GEPCLASS results were higher than those for CSGP. This fact could mislead to a false conclusion that GEPCLASS is unstable. In fact this is a direct consequence of the number of overall number of evaluations for each method. The computational effort can be measured by the product of the number of individuals in the population by the average number of generations to achieve the best result. Thus, GEPCLASS needed, at most 30×50=1,500 evaluations, whereas CSGP needed 500×50=25,000 evaluations. For all datasets, the computational effort needed by GEPCLASS was significantly smaller than that needed by CSGP. Therefore, it can be concluded that, for the datasets tested, GEPCLASS can achieve similar results to those found by CSGP, but with less computational effort. Accuracy is also similar to BGP and C4.5.

Comprehensibility (short rules), not only accuracy, is an important issue in data mining. Since GEPCLASS uses a single chromosome with user-defined number of genes, the concept of parsimony (simplicity) is intrinsically implemented, at the same time giving to the algorithm degrees of freedom to find flexible combinations of attributes (see in table 2 the small number of nodes for the rules found). These are the main advantages inherent to the use of the encoding scheme of GEP. On the other hand, CSGP needs an explicit parsimony term in the fitness function to favor smaller rules.

Some considerations are worth to be done about the underlying structure of CSGP and GEPCLASS. The model used in CSGP demands a logical OR in the root node of individuals. Therefore, an individual will represent, at least two rules (following the Pittsburgh approach). Another constraint of CSGP is that offspring nodes having logical AND as parent node, must have also a logical AND or a relational operator. These constraints were devised to simplify the representation of rule antecedents, as normal disjunctive form. In GEPCLASS, there are no such constraints over the structure of rules, allowing a more flexible combination of attributes. GEPCLASS can handle both Pittsburgh and Michigan approaches at the same time, increasing the possibility to find good solutions for complex classification problems. As a consequence, rules found by GEPCLASS can be, sometimes, substantially different (regarding the attributes of the antecedent) from those found by CSGP. Notwithstanding, table 2 shows that the best rules found have excellent accuracy.

The main contribution of this work is to propose a new methodology for the classification task in data mining inspired in the original GEP algorithm. The

developed system can handle both continuous and categorical attributes and is computationally efficient. Future work will include more experiments with other datasets and the implementation of user-defined fitness functions. Also, the usefulness of the original genetic operators for data classification tasks will be evaluated in depth. Authors intend to put GEPCLASS in public domain soon aiming at fostering further research and applications using this tool.

References

1. Bojarczuk, C.C., Lopes, H.S., Freitas, A.A.: Genetic programming for knowledge discovery in chest pain diagnosis. IEEE Eng. Med. Biol. **19** (2000) 38-44
2. Bojarczuk, C.C., Lopes, H.S., Freitas, A.A.: A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets. Artif. Intell. Med. **30** (2004) 27-48
3. Ferreira, C.: Gene expression programming: a new adaptive algorithm for solving problems. Complex Syst. **13** (2001) 87-129
4. Freitas, A.A.: Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer-Verlag, Berlin, (2002)
5. Hand, D.: Construction and Assessment of Classification Rules. John-Wiley & Sons, New-York (1997)
6. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge (1992)
7. Lopes, H.S., Coutinho, M.S., Lima, W.C.: An evolutionary approach to simulate cognitive feedback learning in medical domain. In: Sanchez, E. et al. (eds.): Genetic Algorithms and Fuzzy Logic Systems. World Scientific, Singapore (1997) 193-207.
8. Lopes, H.S., Weinert, W.R.: EGIPSYS: an enhanced gene expression programming approach for symbolic regression problems. Int. J. Appl. Math. Comput. Sci. **14**:3 (2004) 375-384.
9. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kauffmann, San Mateo, (1993).