Evolution of Decision Trees

Xavier Llorà[†] , **Josep M. Garrell**[†]

[†]Grup de Recerca en Sistemes Intel·ligents,
Enginyeria i Arquitectura La Salle, Universitat Ramon Llull,
Psg. Bonanova 8, 08022-Barcelona, Catalonia
{xevil,josepmg}@salleURL.edu

Abstract

This paper addresses the issue of the induction of orthogonal, oblique and multivariate decision trees. Algorithms proposed by other researchers use heuristic, usually based on the information gain concept, to induce decision trees greedily. These algorithms are often tailored for a given tree type (e.g orthogonal), not being able to induce other types of decision trees. Our work presents an alternative way. We propose to induce a decision trees (without regarding the type) with an unified algorithm based on artificial evolution. Experiments were performed with GALE, our fine-grained parallel Evolutionary Algorithm, and another well-known induction technique on several datasets. Results suggest that Evolutionary Algorithms are competitive and robust for inducing all kinds of decision trees, achieving sometimes better performance than traditional approaches.

Keywords: Genetic Algorithms, Decision Trees, Fine-Grained Parallelism.

1 Introduction

Decision tree induction is a well-known discipline in Machine Learning. Since the work presented by Quinlan in 1986 [21], there has been a great number of researchers interested in this area. Early efforts focus on the induction of orthogonal decision trees using heuristics (i.e. information gain) to build the decision tree greedily. These algorithms can be termed orthogonal due to the parallel-axis boundaries that generate in the instance space. Some examples are ID3 [21] and C4.5 [22]. In order to overcome this constrain, some researchers have proposed oblique decision trees that can define non parallel-axis boundaries (e.g. OC algorithm proposed by Breiman in [4]), or multivariate decision trees that propose non-linear boundaries (e.g. AHA trees proposed by Llorà et al. in [15]). However, each algorithm is usually tailored for the given decision tree type to induce. Moreover, there is another key point when non-orthogonal decision trees are induced, the complexity required by induction makes it infeasible. For instance, deciding the best hyperplane split for an oblique decision tree is *NP-Hard* [12].

We present an Evolutionary Algorithm (EA) [9, 11, 13, 14] to address some of these weaknesses in a new way. Instead of designing a specialized algorithm for inducing a particular kind of decision tree, we propose a general purpose EA for tree induction. EAs have proved to be robust algorithms for search, optimization and machine learning. Therefore, we use EAs robustness to induce decision trees, independently of their type. This evolutionary-driven algorithm, named GALE, induces orthogonal, oblique or multivariate decision trees, scaling up linearly with the number or examples and dimensions in the training set. This algorithm is tested on eleven different domains, showing that it can induce competitive decision trees when compared to well-know machine learning approaches.

Section 2 presents a brief overview of decision tree types. Section 3 summarizes some machine learning work done in EA community for tree induction. In this section we also present GALE, an evolutionary-driven algorithm for inducing decision trees based on fine-grained parallel EAs. Section 4 describes experiments on eleven datasets, comparing the performance of GALE with a well-known

induction technique like C4.5 [22]. Finally, section 5 summarizes some conclusions and further work.

2 Decision Trees

This section briefly describes the three kinds of decision trees used along the paper. It is important to mention here that, throughout, the problems that this work focuses are supervised classification problems. Thus, for a given problem there is a dataset available that contains l instances. Each instance is a numeric vector of dimension d and output class χ.

The simplest decision tree is the one termed *orthogonal* [21, 22]. The internal nodes of this kind of trees define a simple test over a single attribute of the classification problem. This test is usually presented as:

$$a_i \leq \alpha \tag{1}$$

where a_i is a problem attribute and α is a numeric constant. The leaves of the tree are labeled with the class linked to the set of instances represented by the path between the root of the tree and the leaf itself. The boundaries that the equation 1 defines are parallel to the axis of the instance space. This kind of trees have been effectively build using heuristic algorithms based on the *information gain* concept.

In order to solve the limitations that parallel-axis boundaries provide, some authors suggest to used more elaborated tests for the internal nodes of the decision trees. *Oblique* decision trees [4, 23] base the test of the internal nodes of the tree in the following equation:

$$\sum_{i=1}^{d} \omega_i a_i + \omega_{d+1} > 0 \tag{2}$$

This test is based on an orientable hyperplane that can be adjusted computing the vector of coefficients defined by $\overline{\omega} = \langle \omega_1, \omega_2 \ldots \omega_{d+1} \rangle$. Thus, non parallel-axis boundaries can be easily defined choosing the right values for the $\overline{\omega}$ vector. Unfortunately, it has been proved that finding the coefficients for obtaining the best split is *NP-Hard* [12]. Therefore the algorithms used for building this kind of trees use several heuristics [23] to obtain suboptimal trees.

The last kind of decision trees used in this paper belong to *multivariate* decision trees. This kind of trees look for defining non-linear boundaries on the instance space. In order to achieve this goal, each internal node contains a prototype δ [7, 24] (i.e. an artificially defined instance) and an activation threshold θ. A node is *active* if the following equation is satisfied

$$\sqrt{\sum_{i=1}^{d} (\delta_i - e_i)^2} \leq \theta \tag{3}$$

where e_i is the value of a_i in the instance e to classify. This activation check defines hyperspheric boundaries across the instance space. Next, if the node is *active*, the instances e is given to the *nearest child* using the *nearest neighbor* algorithm [1] based on the Euclidean distance. Thus, children defines non parallel-axis hyperplane splits inside the parents hyperspheres. Detailed descriptions of these decision trees can be found in [15].

3 GALE

Genetic and Artificial Life Environment (GALE) [15, 17] explores an alternative path for inducing decision trees. It can induces any of the decision trees presented in section 2 using the same algorithm. The induction algorithm explored by GALE is based on EA, being an evolutionary-driven induction approach. In this section, we briefly review some tree induction efforts of EA community. Next, we describe in detail the EA proposed for inducing decision trees. Finally, the section concludes discussing some scalability issues of GALE.

3.1 Evolutionary Algorithms and Machine Learning

EAs can induce several knowledge representations like rule set or decision trees among others [5, 6, 13, 14, 25, 26]. Some relevant efforts have been done by the Genetic Algorithms (GA) and Genetic Programming (GP) communities. For instance, Cantú-Paz & Kamath [5] used a GA to overcome the complexity constrains of oblique decision trees computing the orientation coefficients ($\overline{\omega}$) along the tree induction process proposed by OC algorithm [4]. They also proposes alternative approaches based on Evolution Strategies (ES) [9]. Other approaches focus on the induction of decision trees using GP. A representative algorithm is proposed by Bot [3], inducing decision trees evolving them directly using

a specially tailored GP model. However, these algorithms tend to be time-consuming and some authors have proposed to exploit the parallel behavior of evolution to reduce the time required for the EA to achieve a solution [2, 8, 10, 15, 16, 17].

3.2 Induction Algorithm

GALE uses a fine-grained parallel EA to obtain a decision tree induced from a set of training examples $\mathcal{T}$. EAs usually evolve a population of individuals (set of feasible solutions) that fit the environment where they are placed (solve efficiently a given problem). The survival of the fittest and their genetic material recombination model the artificial evolution of GALE. Detailed descriptions of EAs can be found in [11, 13].

Each individual in the population of GALE is a decision tree (to be chosen among orthogonal, oblique or multivariate). The population is placed on a 2D grid. Every cell of the grid contains up to one individual that has a *fitness* measure of its degree of adaptation (accuracy solving the problem). The *fitness* measure chosen is $\left(\frac{c}{t}\right)^2$; c is the number of instances in $\mathcal{T}$ that are classified correctly by the individual (decision tree), and t is the number of instances in $\mathcal{T}$. This *fitness* measure provides a non-linear bias toward correctly classifying instances in $\mathcal{T}$ while providing differential reward for imperfect decision trees [6]. Each individual codifies a decision tree into its genotype (genetic representation of the decision tree that undergoes evolution) arranging the tree as a dynamic structure.

GALE spreads the population over a 2D grid in order to exploit massive parallelism and locality relations. Every cell in the grid computes in parallel the same algorithm, that summarizes as follows:

```
FOR-EACH cell C in Grid
DO
  initialize the cell C
  evaluate the accuracy of individual in C
  REPEAT
    merge among neighborhood(C)
    split individual in C
    evaluate the accuracy of individual
             in C
    survival among neighborhood(C)
  UNTIL <end-criterion>
DONE
```

The *initialization* phase decides if the cell contains an individual, or if it remains empty. Empirical experiments show that a 70-80% of occupied cells is a good initialization rate. The individuals of the population are built at random, using the *growing* technique presented in [14]. Next, if the cell contains an individual, C computes the *fitness* of the individual using the fitness function explained before. At this point, the cell is ready to enter the evolutionary process.

Merge, *split*, and *survival* are the phases of the evolutionary cycle that improves the accuracy of individuals. This cycle stops when an individual classifies correctly all the training instances in $\mathcal{T}$, or when a certain amount of evolutionary iterations are done. *Merge* and *split* modify the decision tree (individual), whereas *survival* implements the survival of the fittest, removing poorly adapted individuals from the grid and pruning useless parts of the evolved decision trees.

Merge recombines the individual in C with and individual chosen at random among the neighborhood of the cell, with a given p_m probability. Thus, *merge* is a two-step process: (1) chooses a mate among the neighborhood, and (2) recombines the genetic material obtaining just one offspring. The neighborhood of cell C is the set of its adjacent cells. Therefore, *neighborhood*(C) are the eight cells that surrounds C. The neighborhood topology is beyond the scope of this paper. Next, merge recombines the genetic material using one point crossover proposed by GP [14]. This crossover operator chooses for each tree a node. The two selected nodes of the tree (one for each individual) are swapped, exchanging the genetic material contained on the subtree that they are holding. Finally, the offspring obtained after crossover replaces the individual in C.

Split reproduces and mutates the individual in C, if the probability defined as $p_s \cdot fitness(ind_c)$ is satisfied. The *split* rate is proportional to the performance of the individual. Thus, adapted individuals expand their genetic material rapidly. The splitted individual is placed in the cell with higher number of neighbors (occupied cells), or if all cells in the neighborhood are occupied, it is placed in the cell that contains the worst individual. This technique biases the evolution towards the emergence of useful subpopulations or *demes*, balancing the survival pressure introduced by the *survival* phase. The mutation is done by changing some values of the genotype randomly. For instance, if GALE is evolving orthogonal decision trees, a_i or α might be have a new value generated at random. The

same approach is used for oblique and multivariate decision trees, where $\overline{\omega}$, δ and θ are randomly perturbed.

Survival decides if an individual is kept in the cell for the next cycle. It also prunes the useless parts of the decision that are never used classifying the instances in $\mathcal{T}$. *Survival* uses the number of neighbors in order to decide if the individual is removed form the cell.

0-1 Neighbors: these configurations show a quasi-isolated subpopulation without genetic diversity, thus, there are few chances to improve genetic the material exchanging it with the neighborhood. The survival of the individual is proportional to its performance, $p_{sr}(ind_c) = fitness(ind_c)$, leading to extreme survival pressures. If the individual is removed, the cell becomes empty.

2-6 Neighbors: the subpopulations change rapidly, exchanging a great deal of genetic material. The individual is removed, leaving the cell empty, if $fitness(ind_c) < \overline{\mu}_{nei} + k_{sr} \times \sigma_{nei}$; $\overline{\mu}_{nei}$ is the average fitness value of the occupied neighbor cells, and σ_{nei} their standard deviation. The parameter k_{sr} controls the survival pressure on the current cell, as well as the convergence of GALE.

7-8 Neighbors: this is a crowded situation, without space it the grid available, extreme survival pressure is applied to the individual in C. The individual of the cell is always replaced by its best neighbor, thus $p_{sr}(ind_c) = 0$. Replacement introduces a distributed approach to elitism, a selection technique usually used in many EAs [11].

3.3 Speedup Analysis

Throughout, GALE is a fine-grained parallel model that induces decision trees independently of their type. As a parallel processing algorithm, the theoretical degree of scalability, or speedup, is a useful measure. In order to compute the theoretical speedup equations, we need to obtain the time complexity equations of both a general-purpose evolutionary algorithm for machine learning (GABL [6]) and GALE. In the model equations n is the population size, k the number of iterations of the evolutionary algorithm, l the number of instances in the dataset, d the number of dimensions of the dataset, r the size of the neighborhood, and p is the number of processors used. The time complexity equations for GABL are:

$$\begin{aligned} t_a &= k \cdot t_{loop} \\ &= k\left(t_{eval} + t_{sel} + t_{cross} + t_{mut}\right) \\ &= k\left(t_{cls} nld + t_{copy} n \log n + t_{xalg} p_x n + t_{malg} p_{mut} n\right) \\ &= kn\left(\alpha_1 ld + \alpha_2 \log n + \alpha_3\right) \end{aligned} \tag{4}$$

where t_{cls} is the time of classifying an instance, t_{copy} the time of copying an individual, t_{xalg} the time of the crossover algorithm, and t_{malg} the time of the mutation algorithm. Once we have the time complexity equations of a serial evolutionary algorithm, $t_a \in O\left(kn\left(ld + \log n\right)\right)$, we need the model of GALE. To obtain these equations we assume that each cell of GALE contains one individual, being mapped on a different processor, $n = p$. Each processor is connected to its neighbors in the grid with a latency of $O(1)$. Beneath this assumptions, the equations for GALE can be calculated as follows,

$$\begin{aligned} t_b^r &= \frac{1}{p} nk \cdot t_{cell} \\ &= \frac{1}{n} nk \cdot t_{cell} \\ &= k\left(t_{eval} + t_{merge} + t_{split} + t_{survival}\right) \\ &= k\left(t_{cls} ld + t_{ralg} p_m + t_{salg} p_s + t_{copy}\left(2r+1\right)^2\right) \\ &= k\left(\alpha_1 ld + \alpha_2\left(2r+1\right)^2 + \alpha_3\right) \end{aligned} \tag{5}$$

where t_{cls} is the time of classifying an instance, t_{copy} the time of copying an individual, t_{ralg} the time of the merge algorithm, and t_{salg} the time of splitting. Thus, if $r = 1$ (minimizes communication efforts) then $t_b^1 \in O\left(kld\right)$. Finally, the speedup equation becomes:

$$s^1 = \frac{t_a}{t_b^1} = \frac{kn\left(ld + \log n\right)}{kld} = n\left(1 + \frac{\log n}{ld}\right) \approx n \tag{6}$$

The speedup equation shows that it grows linearly to the number of processors used, avoiding the serial bottleneck selection based on roulette. Equation 5 also shows that the expended time using GALE does not depend on the population size, it is just a linear function of the number of instances, their number of dimensions, and iterations.

4 Experimental Results

The test suit designed for evaluating the classification performance of GALE consists of several datasets and machine learning algorithms. This fact let us study the performance of GALE using statistical tools, like stratified ten-fold cross-validation runs and paired t-tests on these runs [27]. Time performance of GALE as a parallel processing algorithm is beyond of the scope of this paper, being part of the further work.

4.1 Datasets

In order to evaluate the performances of GALE on different domains, we performed experiments on eleven datasets. Datasets can be grouped up to three different kinds of datasets: artificial, public and private.

We used one *artificial datasets* to tune GALE, because we knew its solution in advance. `Tao` is a dataset obtained from sampling the TAO figure using a grid. The grid used is 2D ranging from [-6,6] centimeters with 4 instances per centimeter. The class of each instance is the color (black or white) of the sample in the TAO figure. This grid has nothing in common to the one used by GALE, it is just used for sampling the TAO figure.

Public datasets are obtained from UCI repository [20]. We chose eight datasets from this repository: `breast-w`, `diabetes` (pima-indians), `glass`, `heart-statslog`, `ionosphere`, `iris`, `sonar`, and `vehicle`. These datasets contain numeric attributes, as well as binary and n-ary classification tasks. Finally, we also used two *private datasets* from our own repository. They deal with diagnosis of breast cancer, `biopsies` [19], and `mammograms` [18]. `Biopsies` is the result of digitally processing biopsy images, whereas `mammograms` uses mammographic images.

4.2 Classifier Schemes

As well as GALE algorithm described above, we also run two additional classifier schemes on the previous datasets. We want to compare the results obtained by GALE with the ones obtained using well-known non-evolutionary classifier schemes. These algorithms are coded into the *Waikato Environment for Knowledge Analysis* (WEKA) [27] (on-line code available from `http://www.cs.waikato.ac.nz/ml/weka`).

The first non-evolutionary scheme is **0-R** that predicts the majority class in the training data. This scheme can be useful for determining a baseline performance as a benchmark for other learning schemes [27]. The second classifier tested is **C4.5 revision 8** (C4.5r8). Using the training instances, it induces an orthogonal decision tree that it is in charge of predicting the class for test instances. C4.5r8 is the result of series of improvements to ID3 [21] that include methods for dealing with numeric attributes, missing values and noisy data [22].

4.3 Results

GALE was run using the same parameters for all datasets. The grid was 64×64 with a 80% of occupied cells after initialization. Merge and split probabilities were set to $p_m = 0.4$ and $p_s = 0.01$, and $k_{sr} = -0.25$. The maximum number of iterations was 150. These values have shown a good balance between exploration and exploitation along the evolutionary process [17]. The non-evolutionary classifier schemes used their default configurations.

Table 1 shows the percentage of correct classifications, averaged over stratified ten-fold cross-validation runs, with their corresponding standard deviations. The same folds were used for each scheme. This table also marks GALE results with a • if they show a significant improvement over the corresponding results for C4.5r8, and with a ◦ if they show a significant degradation. Throughout, we speak of results being "significantly different" if the difference is statistically significant at the 1% level according to a paired two-sided t-test, each pair of data points consisting of the estimates obtained in a stratified ten-fold cross-validation run for the two learning schemes being compared. Finally, table 2 summarizes the performance of the different methods compared with each other. Numbers indicate how often the method in row significantly outperforms the method in column.

The results listed in tables 1 and 2 show that 0-R is clearly defeated by the rest of the algorithms studied. Nevertheless, its results help us to identify some difficult datasets for the tree induction algorithms presented. Little improvement is achieved by tree induction algorithms in `diabetes`, `fis`, `heart-statslog`, and `vehicle` datasets, pointing out some further work in tuning specifically the algorithms for these datasets. However, GALE performs better in these datasets than non-evolutionary algorithms, evolving more

Table 1: Experimental results: percentage of correct classifications and standard deviation from stratified ten-fold cross-validation runs. GALE results are also marked with a • if they show a significant improvement (1% significant level on paired two-sided t-test) over the corresponding results for C4.5r8, and with a ○ if they show a significant degradation.

Dataset	0-R	C4.5r8	GALE-ort	GALE-obl	GALE-mul
biopsies	51.61±0.62	80.04±4.80	81.89±5.70	83.74±3.94	83.64±1.61
breast-w	65.52±1.16	95.42±1.69	94.42±1.88	91.70±3.24 ○	95.70±2.23
diabetes	65.10±1.00	73.05±5.32	75.78±4.01	69.40±3.24	74.22±4.34
fis	56.02±2.95	64.81±6.48	71.30±5.93	61.11±8.31	65.27±5.74
glass	35.51±4.49	65.89±10.47	65.42±11.89	49.07±9.20	61.21±10.01
heart-statslog	55.55±0.00	76.30±5.85	82.22±7.11	71.11±7.35	82.96±5.84
ionosphere	64.10±1.19	89.74±5.23	94.02±3.27	90.31±3.57	91.46±4.99
iris	33.33±0.00	95.33±3.26	96.00±3.46	98.67±2.98	94.00±5.83
sonar	53.37±3.78	71.15±8.54	74.52±7.42	68.27±10.03	79.32±6.10•
tao	49.79±0.17	95.07±2.11	89.78±2.29○	91.74±2.65○	93.20±1.87
vehicle	25.06±0.54	73.64±5.42	68.32±6.01○	58.87±5.37○	63.47±4.68○
Average	50.45	80.04	81.24	75.82	80.40

Table 2: Results of paired one-sided t-tests: number indicates how often the method in row significantly outperforms the method in column. Table lists the t-test results using p=0.05 and p=0.01.

	0-R	C4.5r8	GALE-ort	GALE-obl	GALE-mul	0-R	C4.5r8	GALE-ort	GALE-obl	GALE-mul
	t-test at p=0.05					t-test at p=0.01				
0-R	-	0	0	0	0	-	0	0	0	0
C4.5r8	11	-	2	6	2	11	-	2	4	1
GALE-ort	11	3	-	7	5	11	1	-	6	0
GALE-obl	11	2	2	-	1	10	1	0	-	0
GALE-mul	11	3	1	7	-	11	1	1	4	-

accurate orthogonal trees than the ones induced by C4.5r8. This fact happens again when regarding to the average performance, where GALE performs better than C4.5r8.

The oblique decision trees evolved by GALE perform slightly worse than the orthogonal ones. There is one exception to this rule, and it is the results obtained in the iris dataset. Oblique decision trees clearly outperform other decision tree types in this dataset. The runs of GALE evolving oblique decision trees suggest that some specific tuning has to be introduced (e.g double the number of iterations done) due to the size of the search space explored ($m(d+1)$ coefficients are adjusted, being m the number of nodes in the tree, and d the dimensionality of the dataset) when evolving the trees. Similar considerations apply to the GALE when it evolves multivariate decision trees. For instance, in sonar dataset obtains impressive results due to the non-linear boundaries that defines over the instance space. Regarding the t-test experiments, listed in table 2, the orthogonal trees evolved by GALE clearly beats, at different significant levels, the oblique and multivariate ones. Therefore, it seems that the specific tuning for the evolution in oblique and multivariate decision trees is a must to take into count as further work.

5 Conclusions and Further work

This paper presented an evolutionary approach to tree induction. In this paper, we have presented a general purpose evolutionary classifier scheme (GALE) that evolves different types of decision trees. Usually tree induction algorithms are tailored for a specific tree representation. GALE is not bounded to this constrain because it is knowledge independent, thus it can easily evolve orthogonal, oblique or multivariate decision trees. Another important point in tree induction is how the algorithm scales up respect to the size of the training set, as well as the used dimensions. Evolutionary Algorithms tend to be time-consuming. GALE can reduce the amount of real time required exploiting fine-grained parallel processing. The classifier scheme is based on spatial neighborhood relations on a 2D grid, where local genetic operators and massive parallel fitness computation are defined. When enough processors are provided, the time complexity of GALE is independent of the population size, being only bounded by the number of iterations, the size of the dataset to be mined and the dimensions used, $O(kld)$.

The results obtained show that GALE can effectively evolve accurate decision trees, whether they are orthogonal, oblique or multivariate. Although these results are first step into the evolution of decision trees, some conclusions can be summarized. The orthogonal decision trees evolved by GALE outperforms the ones induced by C4.5r8. On the other hand, GALE also evolve accurate oblique and multivariate decision trees. Nevertheless, the results arises the need for some specific tuning that exploits the capabilities of these non-orthogonal decision trees.

Finally, we want to summarize some further work already presented through the paper. GALE is an inherent parallel model, moreover it is a massive fine-grained parallel model. Therefore, we are interested in analyzing the time performance obtained using parallel implementations of GALE. We are also working on obtaining a specific tuning of GALE to evolve more accurate oblique and orthogonal decision trees. Thus, we can extend the comparison presented including other well-known induction schemes (i.e. OC [4]) and datasets.

Acknowledgments

We would like to thank *CIRIT* and *Fondo de Investigación Sanitaria (Instituto Carlos III)* for their support under grant numbers 1999FI-00719 and FIS-00/0033-2, as well as *Enginyeria i Arquitectura La Salle* for their support to our research group. We would also like to thank Ian H. Witten and Eibe Frank for providing their codes on-line. Finally, we want to thank Erick Cantú-Paz for its useful comments and discussions during the preparation of this paper.

References

[1] D. W. Aha, D. Kibler, and Mark K. Albert. Instance-based Learning Algorithms. *Machine Learning*, 6:37–66, 1991.

[2] Dieferson L.A. Araujo, Heitor S. Lopes, and Alex A. Freitas. Rule Discovery with a Parallel Genetic Algorithms. In *Workshop on Data Mining with Evolutionary Computation held in GECCO2000*, pages 89–92, 2000.

[3] Martijn C.J. Bot. Improving Induction of Linear Classification Trees with Genetic Programming. In *Genetic and Evolutionary Computation Conference*, pages 403–410. Morgan Kaufmann, 2000.

[4] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth International Group, 1984.

[5] Erick Cantú-Paz and Chandrika Kamath. Using Evolutionary Algorithms to Induce Oblique Decision Trees. In *Genetic and Evolutionary Computation Conference*, pages 1053–1060. Morgan Kaufmann, 2000.

[6] Kenneth A De Jong and William M. Spears. Learning Concept Classification Rules Using Genetic Algorithms. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 651–656. Morgan Kaufmann, 1991.

[7] Pedro Domingos. Rule Induction and Instance-based Learning: A Unified Approach. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1226–1232. Morgan Kaufmann, 1995.

[8] Ian W. Flockhart. GA-MINER: Parallel Data Mining with Hierarchical Genetic Algorithms (Final Report). Technical Report EPCC-AIKMS-GA-MINER-REPORT 1.0, University of Edinburgh, 1995.

[9] David B. Fogel. *Evolutionary Computation: Toward a New Philosopy of Machine Intelligence.* IEEE Press, 1992.

[10] Alex A. Freitas. A genetic algorithm for generalized rule induction. *Advances in Soft Computing - Engineering Design and Manufacturing*, pages 340–353, 1999.

[11] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Company Inc., 1989.

[12] D. Heath, Simon Kasif, and Steven Salzberg. Learning oblique decision trees. In *Proceedings of the 13th Intenational Joint Conference on Artificial Intelligece*, pages 1002–1007. Morgan Kaufmann, 1993.

[13] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press/Bradford Books edition, 1975.

[14] John R. Koza. *Genetic Programing: On the Programing of Computers by Means of Natural Selection (Complex Adaptive Systems).* MIT Press, 1992.

[15] Xavier Llorà and Josep M. Garrell. Automatic Classification and Artificial Life Models. In *Proceedings of Learning00 Workshop*, pages 1–6, 2000.

[16] Xavier Llorà and Josep M. Garrell. Inducing partially-defined instances with Evolutionary Algorithms. In *Proceedings of the 18th International Conference on Machine Learning (*to appear*)*, 2001.

[17] Xavier Llorà and Josep M. Garrell. Knowledge-Independent Data Mining with Fine-Grained Parallel Evoluationary Algorithms. In *Genetic and Evolutionary Computation Conference*, pages 461–468. Morgan Kaufmann, 2001.

[18] J. Martí, X. Cufí, and J. Regincós. Shape-based feature selection for microcalcification evaluation. In *Proceedings of the SPIE Medical Imaging Conference on Image Processing*, pages 1215–1224, 1998.

[19] E. Martínez and E. Santamaría. Morphological Analysis of Mammary Biopsy Images. In *8th Mediterranean Electrotechnical Conference on Industrial Applications in Power Systems, Computer Science and Telecommunications.*, pages 1067–1070, 1996.

[20] C. J. Merz and P. M. Murphy. UCI Repository for Machine Learning Data-Bases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. *Irvine, CA: University of California, Department of Information and Computer Science.*, 1996.

[21] R. Quinlan. Induction of decission trees. *Machine Learning, Vol. 1, No. 1*, pages 81–106, 1986.

[22] R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, 1993.

[23] T. Van de Merckt. Decision trees in numerical attribute spaces. In *Proceedings of the 13th International Joint Conference on Artificial Intelligece*, pages 1016–1021. Morgan Kaufmann, 1993.

[24] Dietrich Wettschereck. A hybrid Nearest-Neighbor and Nearest-Hyperrectangle Algorithm. In *Proceedings of the 7th European Conference on Machine Learning, LNAI*, volume 784, pages 323–335, 1994.

[25] Stewart W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[26] Stewart W. Wilson. Mining Oblique Data with XCS. *IlliGAL Report No. 2000028*, 2000.

[27] I. H. Witten and E. Frank. *Data Mining: practical machine learning tools and techniques with Java implementations.* Morgan Kaufmann, 2000.