The role of interval initialization in a GBML system with rule representation and adaptive discrete intervals

Jaume Bacardit and Josep Maria Garrell

Enginyeria i Arquitectura La Salle,
Universitat Ramon Llull,
Psg. Bonanova 8, 08022-Barcelona,
Catalonia, Spain, Europe. {jbacardit,josepmg}@salleURL.edu

Abstract. This paper examines some initialization methods for a genetic based machine learning (GBML) rule representation [2] which works with adaptive discretization intervals. The methods studied apply different degrees of uniformness to the initial intervals of the population. The tests done show that except the test problems with more attributes, the differences between the tested methods accuracies are not significant. This proves that we only have to be aware of it in a limited kind of problems.

Keywords: Evolutionary Computation, Machine Learning, Adaptive Discretization.

1 Introduction

The application of the Genetic Algorithms (GA) to classification problems is a field with a lot of work done, and traditionally it has been addressed from two different points of view: the Pittsburgh approach, and the Michigan approach, early exemplified by LS-1 [17] and CS-1 [9], respectively.

The classical solution proposed by those systems is a set of rules which use a prefixed finite number of intervals to handle real-valued attributes. If the discretization intervals are not good, the accuracy of the solution probably will not be good enough.

A simple approach to apply discrete rules is the use of a high number of prefixed uniform discretization intervals. However, it has the problem that the search space grows easily, which slows the evolution process without a clean accuracy improvement of the solution.

Lately, several alternatives to the discrete rules were developed in the genetic-based classifier systems field [19, 5, 13]. Those alternatives present better performance on problems with real-valued attributes but also have more computational cost.

A middle point between these two point of view is the use of a smart discretization algorithm [7, 12, 11] , which usually is more accurate and faster than the uniform discretization, but they are not robust enough in some domains as was shown in our previous work [2].

As a more robust alternative, in this paper we present a work on a rule representation with adaptive discrete intervals which split and merge through the evolution process in the training stage. This approach avoids the higher computational cost of the approaches which work directly with real values and finds a good discretization only expanding the search space with small intervals where necessary.

This representation was first presented at [2] and in this paper we focus on a specific issues: the method used to create the initial intervals. The initialization method has to generate enough diversity in the initial pool to make sure that the evolution will converge to a good solution, and we want to test which degree of diversity is the correct.

The paper is structured as follows: After the related work in section 2 we describe the framework of the work presented in this paper in section 3 and the initialization methods in section 4. Next, the description of the test suite in section 5, its results in section 6 and finally, there are the conclusions and further work in section 7.

2 Related work

There exist several alternatives to discrete rules for handling problems with real-valued attributes inside the evolutionary computation field. Some of these alternatives work on rules composed by interval predicates with real numbers [19], fuzzy rules [5], instances and decision trees [13].

The field of discretization methods is also rich, There are methods that work with the entropy of the information [7], the χ^2 statistic [12] or a multi-dimensional non-uniform discretization [11].

About the intervals initialization, there is not much work published or it is not very clear, because usually it is taken for granted some kind of random initialization, but it is not explained well. Moreover, most of the systems following the Michigan approach rely on a covering operator to fill the population, avoiding the initialization stage.

Anyway, some examples can be found in the literature. Aguilar et al. [1] evolve axis-parallel hyper-rectangles where the the interval for each hyper-rectangle dimension is centered on a random value from the training set attributes and have a random width.

Other loosely related work can be found in the genetic fuzzy systems also. Cordón et al. [5] evolve fuzzy set membership functions initializing them from previously known ones, but each individual with little variations.

3 Framework

In this section we describe the main features of our classifier system, called GAssist (*Genetic Algorithms based claSSIfier sySTem*) [8], a Pittsburgh style classifier system inspired by GABIL [6]. Also, we will describe the working of the adaptive intervals rule representation. The structure of the rules, the crossover and the fitness computation have been borrowed from GABIL. The uncommon parts of the system are explained in the following subsections.

3.1 Matching strategy

The matching process evaluates the rules using a "if .... then .... else if ... then..." structure usually called *Decision List* [16]

3.2 Mutation operators

The variable-length individuals that we have in this system makes more difficult the sizing of the the classic gene-based mutation probability. To simplify this sizing, we will define p_{mut} as the probability that an individual has of being mutated. After an individual is selected for mutation with this probability, we choose a random gene inside its chromosome which is mutated.

3.3 Control of the individuals length

Like all the systems following the Pittsburgh approach, and, in general, most of the representations with variable-length individuals [18], the control of the size of the individuals is a very important issue. This control is achieved with two operators:

Rule deletion We delete the rules of the individuals which do not match any training example after the evaluation process. This rule elimination is done after the fitness computation. It has the following constraints:

- The elimination process is activated after some number of iterations, to prevent a great loss of diversity in the early stage of the evolution process.
- The number of rules of an individual never is reduced below a certain minimum value. This minimum can be decided using information of the problem domain or with the following options:
  - The minimum number of rules that let the crossover operator generate longer individuals again.
  - The number of classes in the problem.
- We follow the second policy in the tests done in this work.

Selection based on individual size The selection operator, as usual, is guided by the fitness (the accuracy), but it also gives certain degree of importance to the size of the individuals. We use tournament selection because its local behavior lets apply this policy. The criterion of the tournament will be given by an operator called "size-based comparison" [3], defined as follows:

- The operator will not be active until a minimum number of iterations have been reached, using only the fitness to do the selection.
- Given two individuals a and b, each one with fitness f_a and f_b and size (in rules) r_a and r_b.
- We will define a "distance" parameter called d_{comp} which will act in the following way:
  - If $|f_a - f_b| < d_{comp}$ then:
    - If $r_a < r_b$ then a is better than b
    - If $r_a > r_b$ then b is better than a
    - If $r_a = r_b$ then we will use the general case
  - Else, we use the general case: we select the individual with higher fitness.

3.4 Discrete adaptive intervals rule representation

Our aim is to find good discretization intervals without a great expansion of the search space. To achieve this objective, we present a rule representation with discrete adaptive intervals where the discretization intervals are not fixed, and evolve through the iterations, merging and splitting between them.

In order to make this representation achieve a reasonable computational cost and to control well the search space, it has to follow the next constraints:

- We define a number of "low level" uniform and static intervals for each attribute which we call *micro-intervals*.
- The adaptive intervals are built joining *micro-intervals*.
- Attributes with different number of *micro-intervals* can coexist in the population, but attributes can not change its number of *micro-intervals* during the evolution.
- For computational cost reasons, we will have a upper limit in the number of intervals allowed for an attribute, which in most cases will be less that the number of *micro-intervals* assigned to each attribute.
- When we split an interval, we select a random point in its *micro-intervals* to apply the operator.
- When we merge two intervals, the value of the resulting interval is taken from the one which has more *micro-intervals*. If both have the same number of *micro-intervals*, the value is chosen randomly.
- The initial number of micro-intervals is chosen form a predefined set.
- The number and size of the initial intervals is selected randomly

The adaptive intervals and the split and merge operations are represented in figure 1.

In order to make the interval splitting and merging part of the evolutionary process, we have to include it in the GA's genetic operators. We have chosen to add to the GA cycle two special stages applied to the offspring population after the mutation stage. The new cycle is represented in figure 2. For each stage (split and merge) we have a probability (p_{split} or p_{merge}) of applying a split or merge operation to an individual. If an individual is selected for split or merge, a random point inside its chromosome is chosen to apply the operation.

This representation requires some changes to the other parts of the GA:

Fig. 1. Diagrams of the adaptive intervals representation and the split and merge operations.

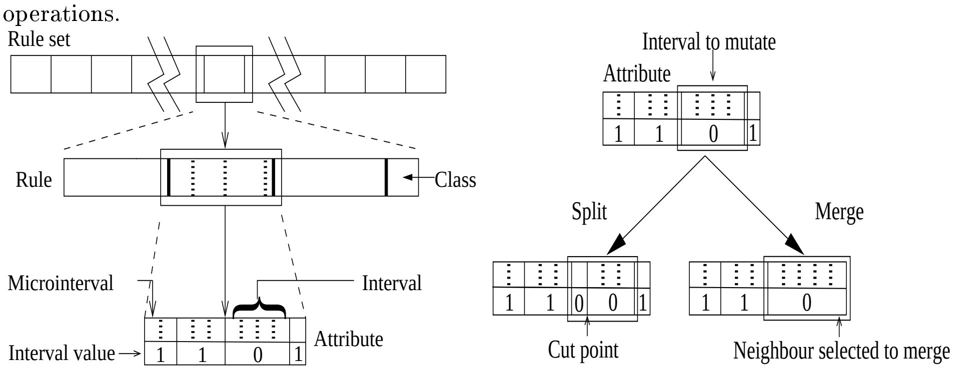
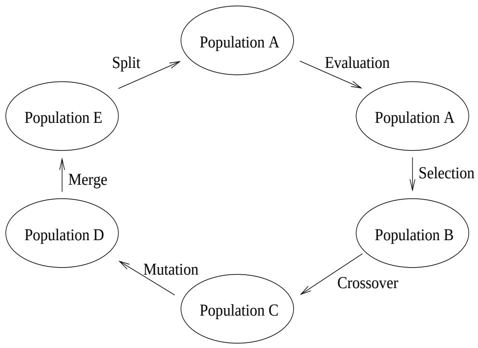


Fig. 2. Cycle of the GA with the added *split* and *merge* operators.



- The crossover operator can only select points of crossover in the attribute boundaries.
- The "size-based comparison" operator uses the length (number of genes) of the individual instead of the number of rules, because now the size of a rule can change when the number of intervals that contains change. This change also makes the GA prefer the individuals with fewer intervals in addition to fewer rules, further simplifying and generalizing them [3].

The correct sizing of p_{split} and p_{merge} will be critical for achieving a good evolution process. A value too small will prevent a good exploration, and a value too big can bring over-learning to the rule set. The sizing of these two probabilities depends on the number of attributes of the problem domain, because the probabilities are defined as affecting individuals. So, domains with very different number of attributes will have also a very different sizing of p_{split} and p_{merge}.

4 Initialization of the adaptive intervals

In the previous section we stated that the number and size of the initial intervals is chosen randomly. In this section we present some alternative initialization policies, with different degrees of randomness that will range from an uniform initialization to the previous used one.

These policies explained in this section are applied after a number of *micro-intervals* has been selected for an attribute, and decide two aspects: (a) having uniform or non-uniform interval size and (b) the number of initial intervals for each attribute term.

Initial number of intervals for each attribute In this point we can choose two policies:

- Uniform number of intervals: All the attributes of the population have a predefined number of intervals.
- Non-Uniform number of intervals: Initialize each attribute of the population with a random number of intervals between two and the maximum number of intervals allowed for an attribute.

Size (in micro-intervals) of the intervals We can have three different policies:

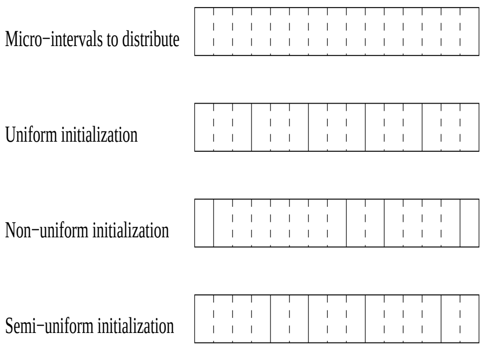
- Uniform intervals size: All intervals have the same number of micro-intervals.
- Non-Uniform intervals size: Fully random-length interval.
- Semi-Uniform intervals size: A policy with a behavior which resides in the middle of the two previous ones. We will choose a size ranging from one to the double of the size of the uniform interval size, that is:

$$[1 : 2 \cdot \frac{\text{Total number of micro-intervals}}{\text{Total number of intervals}}]$$

These three initialization methods are represented in the example in figure 3.

If we combine the two kind of policies, we can have 6 initialization methods with more or less uniformity.

Fig. 3. Initialization policies dealing with the intervals size.



5 Test suite

In this section we describe the tests that have been done to evaluate the accuracy and simplicity of the methods presented in this paper and also the sizing of their control parameters.

5.1 Test problems

The test problems selected for this paper include different classes of problems, in order to give us a broad overview of the performance of the methods being tested.

First of all, we will have a synthetic problem (*Tao* [13]) which has non-orthogonal frontiers between classes. Also, we use several problems of the University of California at Irvine (UCI) repository [4]. The problems selected are: Pima-indians-diabetes (*Pima*), *Iris*, *Glass* and breast-cancer-winsconsin (*Breast*). Finally we will use three problems of our own private repository. The first two deal with diagnosis of breast cancer, biopsies (*Bps* [15]) and mammograms (*Mamm* [14]) and the last one with the prediction of student qualifications (*Lrn* [8]). The characteristics of the problems are listed in table 1.

Table 1. Characteristics of the test problems.

Name	examples	real attr.	discrete attr.	Classes
Tao	1888	2	-	2
Pima	768	8	-	2
Iris	150	4	-	3
Glass	214	9	-	6
Breast	699	-	9	2
Bps	1027	24	-	2
Mamm	216	21	-	2
Lrn	648	4	2	5

The partition of the examples into the train and test sets has been done using *stratified ten-fold cross-validation* method [10].

5.2 Configurations of the GA to test

We have tested the combinations of the two kind of initialization policies explained in the previous section, giving the 6 test configurations shown in table 2.

Table 2. Configurations of the GA to test

Configuration	Number of Intervals	Size of Intervals
conf1	Uniform	Uniform
conf2	Uniform	Semi-Uniform
conf3	Uniform	Non-Uniform
conf4	Non-Uniform	Uniform
conf5	Non-Uniform	Semi-Uniform
conf6	Non-Uniform	Non-Uniform

The parameters of the GA for the tests are shown in table 3 which are distributed in two tables. The first one contains the common parameters for all the tests while the second table contains the parameter which take a different value for each problem. Most of these parameters have been determined empirically with some short tests.

The reader can see that the sizing of both p_{split} and p_{merge} is the same for all the test problems except the tao problem, where giving a similar value to p_{merge} than p_{split} produced a solution with too small number of rules and intervals, and less accurate than the results shown in the next section. This is an issue that needs further study.

In these tests, as we have explained in the framework section, we will have attributes with different number of intervals coexisting in the population, instead of the tests done in a previous paper [2] where all the attributes of the population have the same number of *micro-intervals* and can achieve better results in some of the problem domain. The reason is simple, the *multi-adaptive* tests will be more sensitive to the initialization method as there is more diversity in the population. This can give more clear results about which initialization method is best.

6 Results

In this section we will show the results of the tests done. The objective of the tests was to compare the method presented in their accuracy, but also to detect if the evolution process works well and generates an accurate but also minimal solution.

Table 3. Common and problem-specific parameters of the GA.

Parameter	Value
Crossover probability	0.6
Iter. of rule deletion activation	30
Iter. of size comparison activation	30
Sets of *micro-intervals* used	5,6,7,8,10,15,20,25
Tournament size	3

Code	Parameter
#iter	Number of GA iterations
p_{mut}	Probability of mutating an individual
$popsize$	Size of the population
d_{comp}	Distance param. in the "size-based comparison"
p_{split}	Probability of splitting an interval
p_{merge}	Probability of merging an interval

Probl.	Parameter					
	#iter	p_{mut}	$popsize$	d_comp	p_{split}	p_{merge}
Tao	1500	0.8	200	0.0005	0.3	0.6
Pima	1500	0.6	200	0.01	0.25	0.25
Iris	500	0.8	500	0.02	0.25	0.25
Glass	1500	0.6	300	0.015	0.6	0.6
Breast	1250	0.6	200	0.01	0.7	0.7
Bps	1000	0.6	200	0.015	0.6	0.6
Mamm	1000	0.6	300	0.01	0.8	0.8
Lrn	2000	0.7	200	0.01	0.3	0.3

For these reasons, the results that follow show, for each tested method and test problem: the average and standard deviation values of the percentage of correct classification adjusted by the test fold size and the size of the individuals in number of rules and also intervals per attribute.

Finally, for each configuration, test and fold, 15 runs with different initial seeds are done. After the tests have been explained, the results of them are showed in the table 4. These results are summarized with the ranking of the table 5 build based on the accuracy on each of the test problems.

The first thing we can see in the results is that in only two problems (*Bps* and *Mamm*, the two problems with more attributes) the differences between the test configurations are quite significant in accuracy but specially in the number of rules and intervals per attribute, showing that these problems are quite sensitive to the initialization method. Another significant aspect of the results for these two problems is that the best method for both (conf6) gives also the most reduced solution.

Also, we can observe a curious situation in the domains *Pima*, *Iris*, *Glass* and *Breast*, where the number of intervals per rule is less than two. This means that there are several attributes with only one interval which indicates that they are irrelevant in the rules where they appear.

Finally, if we try to extract a global meaning of the results looking at the ranking table, we see that the methods with a non-uniform number of intervals (configurations 4, 5 and 6) have a better ranking than the other three methods

Table 4. Mean and standard deviation of the accuracy (percentage of correctly classifier examples) and number of rules for each method tested. Bold entries show the method with best results for each test problem and type of result.

Problem	Config	Accuracy	# of Rules	Interv. per Attr.
Tao	conf1	**94.5±0.9**	15.7±3.2	3.9±0.5
	conf2	94.2±0.9	15.4±3.2	3.8±0.5
	conf3	93.8±1.2	**14.5±3.3**	**3.5±0.5**
	conf4	94.4±1.1	15.5±3.0	3.8±0.5
	conf5	94.0±0.7	15.6±3.6	3.8±0.5
	conf6	94.0±1.1	14.6±3.4	3.6±0.6
Pima	conf1	74.4±2.7	5.2±1.9	1.5±0.3
	conf2	73.9±3.2	4.5±1.4	**1.3±0.2**
	conf3	74.3±3.3	**4.0±1.5**	**1.3±0.2**
	conf4	74.4±3.0	6.3±3.0	1.8±0.5
	conf5	74.3±3.1	5.4±2.5	1.5±0.4
	conf6	**74.6±3.1**	4.4±1.4	**1.3±0.2**
Iris	conf1	**96.3±1.9**	**3.4±0.6**	1.3±0.2
	conf2	96.0±2.6	3.6±1.4	1.3±0.2
	conf3	96.0±2.2	3.6±0.8	1.3±0.2
	conf4	95.9±2.4	3.7±1.1	1.3±0.2
	conf5	95.9±2.2	3.7±1.0	1.3±0.2
	conf6	95.6±2.2	3.6±0.8	**1.3±0.1**
Glass	conf1	65.5±4.4	7.9±2.2	1.7±0.2
	conf2	**65.8±3.2**	7.5±2.0	**1.6±0.2**
	conf3	64.9±3.5	7.3±1.9	**1.6±0.2**
	conf4	65.7±3.0	7.3±2.2	**1.6±0.2**
	conf5	64.9±3.0	7.2±1.9	**1.6±0.2**
	conf6	65.3±3.4	**7.1±2.0**	**1.6±0.2**
Breast	conf1	95.2±2.2	2.5±0.8	1.3±0.2
	conf2	95.1±2.4	2.5±0.8	**1.3±0.1**
	conf3	95.2±2.5	2.6±0.8	1.3±0.2
	conf4	95.2±2.3	2.7±0.8	**1.3±0.1**
	conf5	**95.5±2.4**	**2.4±0.6**	**1.3±0.1**
	conf6	95.2±2.1	2.5±0.6	**1.3±0.1**
Bps	conf1	75.2±3.0	20.3±10.0	4.5±0.6
	conf2	75.1±2.8	20.1±9.9	4.5±0.6
	conf3	77.8±3.1	13.1±6.7	4.2±0.4
	conf4	79.4±3.0	8.9±5.4	2.7±0.4
	conf5	79.5±2.9	6.2±3.8	2.3±0.4
	conf6	**80.1±2.9**	**5.2±3.2**	**2.1±0.4**
Mamm	conf1	64.0±4.7	11.6±3.2	3.5±0.3
	conf2	63.5±4.4	9.8±3.1	3.2±0.3
	conf3	64.9±4.9	7.7±3.0	2.7±0.4
	conf4	64.2±4.5	11.0±3.5	3.1±0.4
	conf5	63.8±4.8	10.1±3.6	3.0±0.5
	conf6	**66.2±4.9**	**6.8±2.8**	**2.6±0.5**
Lrn	conf1	65.8±4.3	8.8±3.1	3.0±0.2
	conf2	66.3±4.7	8.5±2.9	3.0±0.2
	conf3	66.7±3.8	8.2±3.1	2.9±0.2
	conf4	**66.8±4.4**	7.4±2.4	2.9±0.1
	conf5	66.5±3.9	6.9±2.0	**2.8±0.1**
	conf6	66.4±4.1	**6.8±2.0**	**2.8±0.1**

Table 5. Performance ranking of the tested methods. Lower number means better ranking.

Problem	Conf1	Conf2	Conf3	Conf4	Conf5	Conf6
Tao	1	3	6	2	4	5
Pima	2	6	5	3	4	1
Iris	1	3	2	5	4	6
Glass	3	1	6	2	5	4
Breast	3	5	6	4	1	2
Bps	6	5	4	3	2	1
Mamm	4	6	2	3	5	1
Lrn	6	5	2	1	3	4
Average	3.25	4.25	4.125	2.875	3.5	3
Final rank	3	6	5	1	4	2

(1, 2 and 3). If we group the results with the other kind of policy, the size of the intervals, The method with uniform-size intervals is the best.

7 Conclusions and further work

In this paper we have presented several initialization methods for our adaptive discretization intervals rule representation. The objective of the tests done was to detect the degree of sensitivity of the final result to the initialization method.

The results show that, except the two problems with more attributes, there is no significant difference between the accuracy results obtained by the tested methods, and limits the type of problems where we have to be aware to choose a good initialization method. If we have to choose a method, the ranking table shows that the configuration with non-uniform number of intervals but uniform-length intervals is the more regular.

As a further work, we consider that it is important to study the effect of the sizing of the control parameters of the representation, p_{split} and p_{merge}, but also d_{comp}, the parameter of the size-based comparison operator. These parameters, if correctly sized, allow a correct exploration of the search space, but also can bring a premature convergence if their value is not good.

Also as further work, it can be interesting to analyze the results of the tests presented in this paper from the computational cost point of view. If the differences of the accuracy are not significant, maybe the differences in computational time resulting from a good initialization method and also from the adjust of the control parameters can be bigger.

Acknowledgments

The authors acknowledge the support provided under grant numbers 2001FI 00514, CICYT/Tel08-0408-02 and FIS00/0033-02. The results of this work were partially obtained using equipment cofunded by the *Direcció General de Recerca de la Generalitat de Catalunya (D.O.G.C. 30/12/1997)*. Finally we would like to thank Enginyeria i Arquitectura La Salle for their support to our Intelligent Systems Research Group.

References

1. Jesús Aguilar, José Riquelme, and Miguel Toro. Three geometric approaches for representing decision rules in a supervised learning system. In *Late Breaking Papers of the Genetic and Evolutionary Computation Conference (GECCO'99)*, pages 8–15, 19919999.
2. Jaume Bacardit and Josep M. Garrell. Evolution of adaptive discretization intervals for a rule-based genetic learning system. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002) (to appear)*, 2002.
3. Jaume Bacardit and Josep M. Garrell. Métodos de generalización para sistemas clasificadores de Pittsburgh. In *Proceedings of the "Primer Congreso Iberoamericano de Algoritmos Evolutivos y Bioinspirados (AEB'02)"*, pages 486–493, 2002.
4. C. Blake, E. Keogh, and C. Merz. Uci repository of machine learning databases, 1998. Blake, C., Keogh, E., & Merz, C.J. (1998). UCI repository of machine learning databases (www.ics.uci.edu/mlearn/MLRepository.html).
5. O. Cordón, M. del Jesus, and F. Herrera. Genetic learning of fuzzy rule-based classification systems co-operating with fuzzy reasoning methods, 1998. Cordn, O., del Jesus, M.J and Herrera, F. (1998), Genetic learning of fuzzy rule-based classification systems co-operating with fuzzy reasoning methods, International Journal of Intelligent Systems, Vol. 13 (10/11), pp.1025-1053.
6. Kenneth A. DeJong and William M. Spears. Learning concept classification rules using genetic algorithms. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 651–656, 1991.
7. Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, pages 1022–1029, 1993.
8. Elisabet Golobardes, Xavier Llorà, Josep Maria Garrell, David Vernet, and Jaume Bacardit. Genetic classifier system as a heuristic weighting method for a case-based classifier system. *Butlletí de l'Associació Catalana d'Intel.ligència Artificial*, 22:132–141, 2000.
9. John H. Holland. Escaping Brittleness: The possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In Mitchell, Michalski, and Carbonell, editors, *Machine learning, an artificial intelligence approach. Volume II*, pages 593–623. Morgan Kaufmann, 1986.
10. Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145, 1995.
11. Alexander V. Kozlov and Daphne Koller. Nonuniform dynamic discretization in hybrid networks. In *Proceedings of the 13th Annual Conference on Uncertainty in AI (UAI)*, pages 314–325, 1997.
12. H. Liu and R. Setiono. Chi2: Feature selection and discretization of numeric attributes. In *Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence*, pages 388–391. IEEE Computer Society, 1995.
13. Xavier Llorà and Josep M. Garrell. Knowledge-independent data mining with fine-grained parallel evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 461–468. Morgan Kaufmann, 2001.
14. J. Martí, X. Cufí, J. Regincós, and et al. Shape-based feature selection for microcalcification evaluation. In *Imaging Conference on Image Processing, 3338:1215-1224*, 1998.
15. E. Martínez Marroquín, C. Vos, and et al. Morphological analysis of mammary biopsy images. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'96)*, pages 943–947, 1996.

16. Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.
17. Stephen F. Smith. Flexible learning of problem solving heuristics through adaptive search. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI-83)*, pages 421–425, Los Altos, CA, 1983. Morgan Kaufmann.
18. Terence Soule and James A. Foster. Effects of code growth and parsimony pressure on populations in genetic programming. *Evolutionary Computation*, 6(4):293–309, Winter 1998.
19. Stewart W. Wilson. Get real! XCS with continuous-valued inputs. In L. Booker, Stephanie Forrest, M. Mitchell, and Rick L. Riolo, editors, *Festschrift in Honor of John H. Holland*, pages 111–121. Center for the Study of Complex Systems, 1999.