Evolving multiple discretizations with adaptive intervals for a Pittsburgh Rule-Based Learning Classifier System

Jaume Bacardit and Josep Maria Garrell

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

Abstract. One of the ways to solve classification problems with real-value attributes using a Learning Classifier System is the use of a discretization algorithm, which enables traditional discrete knowledge representations to solve these problems. A good discretization should balance losing the minimum of information and having a reasonable number of cut points. Choosing a single discretization that achieves this balance across several domains is not easy. This paper proposes a knowledge representation that uses several discretization (both uniform and non-uniform ones) at the same time, choosing the correct method for each problem and attribute through the iterations. Also, the intervals proposed by each discretization can split and merge among them along the evolutionary process, reducing the search space where possible and expanding it where necessary. The knowledge representation is tested across several domains which represent a broad range of possibilities.

1 Introduction

The application of Genetic Algorithms (GA) [1, 2] to classification problems is usually known as Genetic Based Machine Learning (GBML) or Learning Classifier Systems (LCS), and traditionally it has been addressed from two different points of view: the Pittsburgh approach (also known as Pittsburgh LCS), and the Michigan approach (or Michigan LCS). Some representative systems of each approach are *GABIL* [3] and XCS [4].

The classical knowledge representation used in these systems is a set of rules where the antecedent is defined by a prefixed finite number of intervals to handle real-valued attributes. The performance of these systems is tied to the right election of the intervals through the use of a discretization algorithm.

There exist several good heuristic discretization methods which have good performance on several domains. However, they lack robustness on some other domains because they loose too much information from the original data. The alternative of a high number of simple uniform-width intervals usually expands the size of the search space without a clear performance gain.

In a previous paper [5] we have proposed a representation called *adaptive discrete intervals (ADI) rule representation* where several uniform-width discretizations are used at the same time. Thus, allowing the GA choose the correct discretization for each rule and attribute. Also, the discretization intervals were split and merged through the evolutionary process. This representation has been used in our Pittsburgh approach *LSC*.

In this paper we generalize the ADI representation approach (proposing *ADI2*) by also using heuristic non-uniform discretization methods. In our case the well-known Fayyad & Irani discretization algorithm [6]. In our previous work se reported that using only the Fayyad & Irani discretization can lead to a non-robust system in some domains, but using it together with some uniform-width discretizations improves the performance in some domains and, most important, presents a robust behavior across most domains. Also, the probability that controls the split and merge operators is redefined in order to simplify the tuning needed to use the representation.

This rule representation is compared across different domains against the results of the original representation and also the *XCSR* representation [7] by Wilson which uses rules with real-valued intervals in the well known XCS Michigan LCS [4]. We want to state clearly that we have integrated the *XCSR* representation into our Pittsburgh system, instead of using it inside *XCS*.

The paper is structured as follows. Section 2 presents some related work. Then, we describe the framework of our classifier system section 3. The revision of the ADI representation is explained in section 4. Next, section 5 describes the test suite used in the comparison. The results obtained are summarized in section 6. Finally, section 7 discusses the conclusions and some further work.

2 Related work

There are several approaches to handle real-valued attributes in the LCS field. These approaches can be classified in a simple manner into two groups: Systems that discretize or systems that work directly with real values.

Reducing the real-valued attributes to discrete values let the systems in the first group use traditional symbolic knowledge representations. There are several types of algorithms which can perform this reduction. Some heuristic examples are the Fayyad & Irani method [6] which works with information entropy or the χ^2 statistic measures [8].

Some specific GBML applications of discretization algorithms were presented by Riquelme and Aguilar [9] and are similar to the representation proposed here. Their system evolves conjunctive rules where each conjunct is associated to an attribute and is defined as a range of adjacent discretization intervals. They use their own discretization algorithm, called *USD* [10].

Lately, several alternatives to the discrete rules have been presented. There are rules composed by real-valued intervals (XCSR [7] by Wilson or COGITO [11] by Aguilar and Riquelme). Also, Llorà and Garrell [12] proposed a knowledge independent method for learning other knowledge representations like instance sets or decision trees. Most of these alternatives have better accuracy than the discrete rules, but usually they also have higher computational cost [11].

3 Framework

In this section we describe the main features of our classifier system which is a Pittsburgh LCS based on GABIL [3]. Directly from GABIL we have borrowed the semantically correct crossover operator and the fitness computation (squared accuracy).

Matching strategy: The matching process follows a "if ... then ... else if ... then..." structure, usually called *Decision List* [13].

Mutation operators: The system manipulates variable-length individuals, making more difficult the tuning of the classic gene-based mutation probability. In order to simplify this tuning, we define p_{mut} as the probability of mutating an individual. When an individual is selected for mutation (based on p_{mut}), a random gene is chosen inside its chromosome for mutation.

Control of the individuals length: Dealing with variable-length individuals arises some serious considerations. One of the most important ones is the control of the size of the evolving individuals [14]. This control is achieved using two different operators:

- *Rule deletion:* This operator deletes the rules of the individuals that do not match any training example. This rule deletion is done after the fitness computation and has two constraints: (a) the process is only activated after a predefined number of iterations, to prevent a massive diversity loss and (b) the number of rules of an individual never goes below a lower threshold.
- *Selection bias using the individual size:* Selection is guided as usual by the fitness (the accuracy of the individual). However, it also gives certain degree of relevance to the size of the individuals, having a policy similar to multi-objective systems. We use tournament selection because its local behavior lets us implement this policy. The criterion of the tournament is given by our own operator called "hierarchical selection" [15]. This operator considers two individuals "similar" if their fitness difference is below a certain threshold (d_{comp}). Then, it selects the individual with fewer number of rules. Our previous tests showed that sizing d_{comp} to 0.01 was quite good for real problems and 0.001 was quite good for synthetic problems. Although a fine tuning of this parameter probably would improve the performance for each test domain, for the sake of simplicity we will use these values.

4 The adaptive discretization intervals (ADI) rule representation

4.1 The original ADI representation

The general structure of each rule is taken from *GABIL*. That is, each rule consists of a condition part and a classification part: $condition \rightarrow classification$. Each condition is a Conjunctive Normal Form (CNF) predicate defined as:

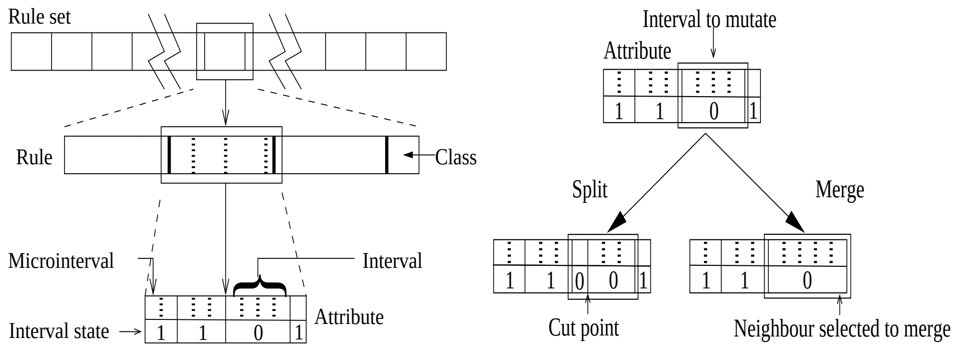
$$((A_1 = V_1^1 \vee \ldots \vee A_1 = V_m^1) \bigwedge \ldots \bigwedge (A_n = V_2^n \vee \ldots A_n = V_m^b))$$

Where A_i is the ith attribute of the problem and V_i^j is the jth value that can take the ith attribute.

In the *GABIL* representation this kind of predicate can be encoded into a binary string in the following way: if we have a problem with two attributes, where each attribute can take three different values {1,2,3}, a rule of the form "If the first attribute has value 1 or 2 and the second one has value 3 then we assign class 1" will be represented by the string 110|001|1. For real-valued attributes, each bit (except the class one) would be associated to a discretization interval.

The intervals of the rules in the ADI representation are not static, but they evolve splitting and merging among them (having a minimum size called *micro-intervals*). Thus, the binary coding of the *GABIL* representation is extended as represented in figure 1, also showing the split and merge operations.

Fig. 1. Adaptive intervals representation and the split and merge operators.



The ADI representation is defined in depth as follows:

1. Each individual, initial rule and attribute term is assigned a number of "low level" uniform-width and static discretization intervals (called *micro-intervals*).
2. The intervals of the rule are built joining together adjacent *micro-intervals.*
3. Attributes with different number of *micro-intervals* can coexist in the population. The evolution will choose the correct number of *micro-intervals* for each attribute.
4. For computational cost reasons, we will have an upper limit in the number of intervals allowed for an attribute, which in most cases will be less than the number of *micro-intervals* assigned to each attribute.
5. When we split an interval, we select a random point in its *micro-intervals* to break it.
6. When we merge two intervals, the state (1 or 0) 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.
7. The number of *micro-intervals* assigned to each attribute term is chosen from a predefined set.
8. The number and size of the initial intervals is selected randomly.

9. The cut points of the crossover operator can only take place in attribute terms boundaries, not between intervals. This restriction takes place in order to maintain the semantical correctness of the rules.
10. The *hierarchical selection* operator uses the length of the individuals (defined as the sum of all the intervals of the individual) instead of the number of rules as the secondary criteria. This change promotes simple individuals with more reduced interval fragmentation.

In order to make the interval splitting and merging part of the evolutionary process, we have to include it in the GAs genetic operators. We have chosen to add to the GA cycle two special stages applied to the offspring population after the mutation stage. For each stage (split and merge) we have a probability (p_{split} or p_{merge}) of applying the 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.

4.2 Revisions to the representation: ADI2

The first modification to the ADI representation affects the definition of the split and merge probabilities. These probabilities were defined at an individual-wise level, and needed an specific adjust for each domain that we want to solve. The need of this fine-adjusting is motivated by the fact that this probability definition does not take into account the number of attributes of the problem nor its optimum number of rules. Thus, a problem having the double of attributes than another problem would also need a probability two times higher. Also, it was empirically determined that it was useful to split or merge more than once for each individual, thus using an expected values instead of a probability to control the operators.

Our proposal is a probability defined for each attribute term of each rule. Thus, becoming independent of these two factors. The code for the merge operator probability is represented in figure 2. Code for the split operator is similar. This redefinition allows us to use the same probability for all the test domains with similar results to the original probability definition. The probability value has been empirically defined as 0.05.

Fig. 2. Code of the application of the merge operator.

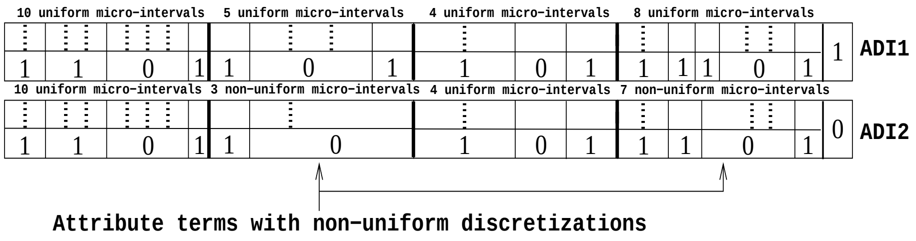
```
ForEach Individual $i$ of $Population$
    ForEach Rule $j$ of $Population$ individual $i$
        ForEach Attribute $k$ of Rule $j$ of $Population$ individual $i$
            If random [0..1] number $< p_{merge}$
                Select one random interval of attribute term $k$
                        of rule $j$ of individual $i$
                Apply a merge operation to this interval
            EndIf
        EndForEach
    EndForEach
EndForEach
```

Our second modification is related to the kind of discretization that we use. The experimentation reported of the original representation included a comparison with a discrete representation where the discretization intervals where given by the Fayyad & Irani method [6]. This discrete representation performed better than the adaptive intervals rule representation in some domains, but it was significantly outperformed in some other domains, showing a lack of robustness.

Our aim is to modify the representation in order to include non-uniform discretization into it, in a way that improves the performance of the system in the problems where it is possible, but maintains a robust behavior in the kind of problems where the systems using only heuristic representations fail. A graphical representation of the two types of rules is in figure 3. The changes introduced to the definition of the representation are minimum. A revised definition follows:

1. **A set of static discretization intervals (called *micro-intervals*) is assigned to each attribute term of each rule of each individual.**
2. The intervals of the rule are built joining together *micro-intervals*.
3. **Attributes with different number and sizes of *micro-intervals* can coexist in the population. The evolution will choose the correct discretization for each attribute.**
4. For computational cost reasons, we will have an upper limit in the number of intervals allowed for an attribute, which in most cases will be less than the number of *micro-intervals* assigned to each attribute.
5. When we split an interval, we select a random point in its *micro-intervals* to break it.
6. When we merge two intervals, the state (1 or 0) 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.
7. **The discretization assigned in the initialization stage to each attribute term is chosen from a predefined set.**
8. The number and size of the initial intervals is selected randomly.
9. The cut points of the crossover operator can only take place in attribute terms boundaries, not between intervals. This restriction takes place in order to maintain the semantical correctness of the rules.
10. The *hierarchical selection* operator uses the length of the individuals (defined as the sum of all the intervals of the individual) instead of the number of rules as the secondary criteria. This change promotes simple individuals with more reduced interval fragmentation.

Fig. 3. Example of the differences between ADI1 and ADI2 rules.


10 uniform micro-intervals
5 uniform micro-intervals
4 uniform micro-intervals
8 uniform micro-intervals
1 1 0 1 1 0 1 1 0 1 1 1 1 0 1
1
ADI1
10 uniform micro-intervals
3 non-uniform micro-intervals
4 uniform micro-intervals
7 non-uniform micro-intervals
1 1 0 1 1 0 1 0 1 1 1 0 1
0
ADI2
Attribute terms with non-uniform discretizations

5 Test suite

This section summarizes the tests done in order to evaluate the accuracy and efficiency of the method presented in this paper. We also compare it with some alternative methods. The tests were conducted using several machine learning problems which we also describe.

5.1 Test problems

The selected test problems are the same that were used in the original version of the representation being studied in this paper. The first problem is a synthetic problem (*tao* [12]) that has non-orthogonal class boundaries. We also use several problems provided by the University of California at Irvine (UCI) repository [16]. The problems selected are: Pima Indians Diabetes (*pim*), Iris (*irs*), Glass (*gls*) and Winsconsin Breast Cancer (*breast*). Finally we will use three real problems from private repositories. The first two deal with the diagnosis of breast cancer based on biopsies (*bps* [17]) and on mammograms (*mmg* [18]), whereas the last one is related to the prediction of student qualifications (*lrn* [19]). The characteristics of the problems are listed in table 1. The partition of the examples into the train and test sets was done using *stratified ten-fold cross-validation* [20].

Table 1. Characteristics of the test problems.

Name	ID.	Instances	Real attr	Discrete attr.	Type	Classes
Tao	tao	1888	2	-	synthetic	2
Pima-Indians-Diabetes	pim	768	8	-	real	2
Iris	irs	150	4	-	real	3
Glass	gls	214	9	-	real	6
Winsconsin Breast Cancer	bre	699	-	9	real	2
Biopsies	bps	1027	24	-	real	2
Mammograms	mmg	216	21	-	real	2
Learning	lrn	648	4	2	real	5

5.2 Configurations of the GA to test

The main goal of the tests is to evaluate the performance of the changes done to the ADI representation: the redefinition of the split and merge probabilities and also the inclusion of non-uniform discretizations. Thus, we performed two kind of tests: The first test (called **ADI2**) uses the new probabilities and only the uniform discretizations. The second one (called **ADI2+Fayyad**) adds to the configuration **ADI2** the use of the Fayyad & Irani discretization method [6].

This revision of the ADI representation is compared to three more configurations:

- Non-adaptive discrete rules (using the *GABIL* representation) using the Fayyad & Irani discretization method to determine the intervals used. This test is labeled **Fayyad**.
- The original ADI representation with the old split and merge probability definition and only using uniform discretizations (called *ADI1*).
- Rules with real-valued intervals using the *XCSR* representation [7] (called **XCSR**).

We have decided not to add any other non-uniform discretization to *ADI2+ Fayyad* because the comparison of these tests with the *Fayyad* test would not be fair if more non-uniform discretization methods were used.

The GA parameters are shown in tables 2 and 3. The first one shows the general parameters and the second one the domain-specific ones. The reader can appreciate that the sizing of both p_{split} and p_{merge} is the same for all the problems except the *tao* problem. Giving the same value to p_{merge} and p_{split} produce solutions with too few rules and intervals, as well as less accurate than the results obtained with the configuration shown in table 3.

The reason of this fact comes from the definition of this synthetic problem with only two attributes and rules with intervals as small as one 48th of an attribute domain in the optimum rule set. The probability of generating these very specific rules is very low because of the *hierarchical selection* operator used to apply generalization pressure. The operator is still used because it is beneficial for the rest of problems and we can fix this bad interaction by increasing p_{split}. The characteristics of this problem explained here are also the reason of the bad performance of the test using only the Fayyad & Irani intervals because the discretization generates too few intervals and, as a consequence, loses too much information from the original data.

Table 2. Common parameters of the GA.

Parameter	Value
General parameters	
Crossover probability	0.6
Selection algorithm	Tournament
Tournament size	3
Population size	300
Probability of mutating an individual	0.6
Rule Deletion operator	
Iteration of activation	30
Minimum number of rules before disabling the operator	number of classes + 3
Hierarchical Selection	
Iteration of activation	10
ADI rule representation	
Number of intervals of the uniform discretizations	4,5,6,7,8,10,15,20,25
XCSR rule representation	
Maximum radius size in initialization	0.7 of each attribute domain
Maximum center offset in mutation	0.7 of each attribute domain
Maximum radius offset in mutation	0.7 of each attribute domain

Table 3. Problem-specific parameters of the GA.

Code	Parameter
#iter	Number of GA iterations
d_{comp}	Distance parameter in the "size-based comparison" operator
E_{split} orig	Expected value interval split in *ADI1*
E_{merge} orig	Expected value of interval merging in *ADI1*
p_{split}	Probability of splitting an interval in *ADI2*
p_{merge}	Probability of merging an interval in *ADI2*

Problem	Parameter					
	#iter	d_{comp}	E_{merge} orig	E_{split} orig	p_{merge}	p_{split}
tao	900	0.001	1.3	2.6	0.05	0.25
pim	250	0.01	0.8	0.8	0.05	0.05
irs	275	0.01	0.5	0.5	0.05	0.05
gls	900	0.01	1.5	1.5	0.05	0.05
bre	300	0.01	3.2	3.2	0.05	0.05
bps	275	0.01	1.7	1.7	0.05	0.05
mmg	225	0.01	1.0	1.0	0.05	0.05
lrn	700	0.01	1.2	1.2	0.05	0.05

6 Results

In this section we present the results obtained. The aim of the tests was to compare the methods studied in this paper in three aspects: accuracy and size of the solutions as well as the computational cost. For each method and test problem we show the average and standard deviation values of: (1) the cross-validation accuracy, (2) the size of the best individual in number of rules and number of intervals and (3) the execution time in seconds. Obviously, the *XCSR* results lack the intervals per attribute column. The tests were executed in an AMD Athlon 1700+ using Linux operating system and C++ language. Runs for each method, problem and fold has been repeated 15 times using different random seeds and the results averaged.

The full detail of the results is shown in table 4 and a summary of them taking the form of a methods ranking is in table 5. The ranking for each problem and method is based on the accuracy. The global rankings are computed averaging the problem rankings. The results were also analyzed using two-sided Student t-tests [20] with a significance level of 1% in order to determine if there were significant outperformances between the methods tested. The results of the t-tests are shown in table 6. None of the *ADI* was outperformed.

The results were also analyzed using the two-sided t-test [20] with a significance level of 1% in order to determine if there were significant outperformances between the methods tested.

The first interesting fact from the results is the comparison between *ADI1* and *ADI2*, that is, the comparison between the split and merge probabilities definitions. *ADI2* the method that uses the same probabilities for all the domains is always better (with only one exception) than *ADI1*, the method that needed domain-specific probabilities. Thus, the gain is double, performance and simplicity of use.

The performance of *ADI2+Fayyad* is also quite good, being the second method in the overall ranking but very close to *ADI2*, the first one. The objective of increasing the *ADI* accuracy in the problems where the *Fayyad* discrete rules alone performed well and, at the same time, maintain the accuracy in the domains where *Fayyad* presented very poor performance has been achieved.

Also, the results of the *XCSR* representation show that the representations based on a discretization process, if well used, present good performance compared to representations that evolve real-valued intervals. This observation matches the ones reported by Riquelme and Aguilar [11].

The computational cost continues being the main drawback of the representation. The comparison of the ADI representation run time with the *Fayyad* discrete rules is clearly favorable to *Fayyad* test. The comparison with *XCSR*, however, is not so clear. In some domains one representation is faster and in other domains it is the reverse situation.

7 Conclusions and further work

This paper focused on a revision and generalization of our previous work on representations for real-valued attributes: the Adaptive Discretization Intervals

Table 4. Mean and deviation of the accuracy, number of rules and intervals per attribute for each method tested. Bold entries show the method with best results for each test problem

Problem	Configuration	Accuracy	Number of Rules	Intervals per attribute	Run time
tao	Fayyad	87.8±1.1	**3.1±0.3**	**3.4±0.1**	**37.3±2.1**
	ADI1	94.3±1.0	19.5±4.9	6.0±0.6	145.6±20.9
	ADI2	**94.7±0.9**	17.5±4.4	8.7±0.6	162.1±20.9
	ADI2+Fayyad	94.0±0.9	15.6±4.4	7.6±1.0	150.5±23.4
	XCSR	91.1±1.4	12.9±3.1	——	110.6±14.5
pim	Fayyad	73.6±3.1	6.6±2.6	2.3±0.2	**13.2±1.5**
	ADI1	74.4±3.1	5.8±2.2	**1.9±0.4**	29.9±4.5
	ADI2	**74.5±3.8**	3.7±1.2	2.1±0.3	30.2±3.2
	ADI2+Fayyad	**74.5±3.3**	3.6±0.9	2.0±0.3	32.1±3.0
	XCSR	74.2±3.3	**3.4±1.0**	——	18.7±2.9
irs	Fayyad	94.2±3.0	**3.2±0.6**	2.8±0.1	4.1±0.1
	ADI1	96.2±2.2	3.6±0.9	**1.3±0.2**	6.2±0.6
	ADI2	96.0±2.6	3.9±0.7	1.5±0.3	6.0±2.0
	ADI2+Fayyad	**96.3±2.4**	3.7±0.7	1.5±0.2	6.4±2.1
	XCSR	95.2±2.3	3.5±0.6	——	**3.5±0.1**
gls	Fayyad	65.7±6.1	8.1±1.4	2.4±0.1	**16.8±1.3**
	ADI1	65.2±4.1	**6.7±2.0**	**1.8±0.2**	46.1±6.0
	ADI2	65.6±3.7	7.6±1.5	2.6±0.2	41.8±3.8
	ADI2+Fayyad	**66.2±3.6**	7.7±1.6	2.5±0.2	42.7±3.5
	XCSR	65.2±5.2	8.4±1.2	——	37.5±4.9
bre	Fayyad	95.2±1.8	4.1±0.8	3.6±0.1	**8.3±0.9**
	ADI1	95.3±2.3	2.6±0.9	**1.7±0.2**	25.0±1.4
	ADI2	**95.9±2.3**	**2.4±0.6**	1.8±0.3	19.6±1.4
	ADI2+Fayyad	95.7±2.4	**2.4±0.6**	1.8±0.3	22.8±1.6
	XCSR	95.8±2.7	3.2±0.9	——	21.4±2.9
bps	Fayyad	80.0±3.1	7.1±3.8	2.4±0.1	**31.1±5.0**
	ADI1	80.1±3.3	5.1±2.0	**2.0±0.3**	99.6±17.0
	ADI2	**80.6±2.9**	**3.6±1.0**	2.1±0.2	113.8±9.9
	ADI2+Fayyad	80.3±2.9	**3.6±1.0**	**2.0±0.2**	119.2±7.7
	XCSR	79.7±3.1	4.3±1.1	——	151.6±13.4
mmg	Fayyad	65.3±11.1	**2.3±0.5**	2.0±0.1	**3.8±0.3**
	ADI1	65.0±6.1	4.4±1.9	**1.9±0.2**	12.3±2.5
	ADI2	**65.6±5.7**	4.9±1.0	2.3±0.1	13.7±1.2
	ADI2+Fayyad	65.2±4.3	4.7±1.1	2.2±0.1	14.4±1.2
	XCSR	63.1±4.1	5.6±1.6	——	14.9±2.4
lrn	Fayyad	67.5±5.1	14.3±5.0	4.4±0.1	**26.5±3.4**
	ADI1	66.7±4.1	11.6±4.1	**3.4±0.2**	53.9±7.2
	ADI2	67.4±4.3	7.2±1.7	3.5±0.2	46.6±5.1
	ADI2+Fayyad	**67.6±4.6**	**6.9±1.4**	**3.4±0.2**	45.9±5.0
	XCSR	67.1±4.4	9.1±2.5	——	52.6±7.5

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

Problem	Fayyad	ADI1	ADI2	ADI2+Fayyad	XCSR
tao	5	2	1	3	4
pim	5	3	1	1	4
irs	5	2	3	1	4
gls	2	4	3	1	4
bre	5	4	1	3	2
bps	4	3	1	2	5
mmg	2	4	1	3	5
lrn	2	4	3	1	4
Average	3.75	3.25	1.75	1.875	4
Final rank	4	3	1	2	5

Table 6. Summary of the statistical two-sided t-test performed at the 1% significance level. Each cell indicates how many times the method in the row outperforms the method in the column

Method	*Fayyad*	*ADI1*	*ADI2*	*ADI2+Fayyad*	*XCSR*	*Total*
Fayyad	-	0	0	0	0	0
ADI1	1	-	0	0	2	3
ADI2	1	0	-	0	2	3
ADI2+Fayyad	1	0	0	-	2	3
XCSR	1	0	0	0	-	1
Total	4	0	0	0	6	

(ADI) rule representation. This representation evolves rules that can use multiple discretizations, letting the evolution choose the correct discretization for each rule and attribute. Also, the intervals defined in each discretization can split or merge among them through the evolution process, reducing the search space where it is possible and expanding it where it is needed.

This revision has consisted of two parts: redefining the split and merge operators probabilities and using non-uniform discretizations in the representation in addition of the existing uniform ones. The redefinition of the operators has simplified enormously the tuning needed to use the representation because we have converted an operator which needed domain-specific tuning into an operator which uses the same probability for all the domains. The addition of non-uniform discretizations to the representation has improved its performance in some domains while maintaining the robust behavior that presented the original version, according to the statistical t-tests done.

The tests done show that the new version is always better than the original one and also better than the other representations included in the comparison when used in a Pittsburgh LCS. However, the computational cost continues being a major drawback, compared to the discrete rules.

As a further work, other non-uniform discretization methods beside the Fayyad & Irani one should be tested, to increase the range of problems where this representation performs well. This search for more discretization methods, however, should always have in mind maintaining the robustness behavior that the representation has showed so far. On the other hand, the ADI representation should be compared with other kind of representations dealing with real-valued attributes, for example non-orthogonal ones.

Acknowledgments

The authors acknowledge the support provided under grant numbers 2001FI 00514, TIC2002-04160-C02-02, TIC 2002-04036-C05-03 and 2002SGR 00155. Also, we would like to thank Enginyeria i Arquitectura La Salle for their support to our research group. The Wisconsin breast cancer databases was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg.

References

1. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
2. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Inc. (1989)

3. DeJong, K.A., Spears, W.M.: Learning concept classification rules using genetic algorithms. Proceedings of the International Joint Conference on Artificial Intelligence (1991) 651–656
4. Wilson, S.W.: Classifier fitness based on accuracy. Evolutionary Computation **3** (1995) 149–175
5. Bacardit, J., Garrell, J.M.: Evolution of multi-adaptive discretization intervals for A rule-based genetic learning system. In: Proceedings of the VIII Iberoamerican Conference on Artificial Intelligence (IBERAMIA'2002), LNAI vol. 2527, Springer (2002) 350–360
6. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: IJCAI. (1993) 1022–1029
7. Wilson, S.W.: Get real! XCS with continuous-valued inputs. In Booker, L., Forrest, S., Mitchell, M., Riolo, R.L., eds.: Festschrift in Honor of John H. Holland, Center for the Study of Complex Systems (1999) 111–121
8. Liu, H., Setiono, R.: Chi2: Feature selection and discretization of numeric attributes. In: Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence, IEEE Computer Society (1995) 388–391
9. Aguilar-Ruiz, J.S., Riquelme, J.C., Valle, C.D.: Improving the evolutionary coding for machine learning tasks. In: Proceedings of the European Conference on Artificial Intelligence, ECAI'02, Lyon, France, IOS Press (2002) pp. 173–177
10. Giráldez, R., Aguilar-Ruiz, J.S., Riquelme, J.C.: Discretización supervisada no paramétrica orientada a la obtención de reglas de decisión. In: Proceedings of the CAEPIA2001. (2001) 53–62
11. Riquelme, J.C., Aguilar, J.S.: Codificación indexada de atributos continuos para algoritmos evolutivos en aprendizaje supervisado. In: Proceedings of the "Primer Congreso Español de Algoritmos Evolutivos y Bioinspirados (AEB'02)". (2002) 161–167
12. Llorà, X., Garrell, J.M.: Knowledge-independent data mining with fine-grained parallel evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), Morgan Kaufmann (2001) 461–468
13. Rivest, R.L.: Learning decision lists. Machine Learning **2** (1987) 229–246
14. Soule, T., Foster, J.A.: Effects of code growth and parsimony pressure on populations in genetic programming. Evolutionary Computation **6** (1998) 293–309
15. Bacardit, J., Garrell, J.M.: Métodos de generalización para sistemas clasificadores de Pittsburgh. In: Proceedings of the "Primer Congreso Español de Algoritmos Evolutivos y Bioinspirados (AEB'02)". (2002) 486–493
16. Blake, C., Keogh, E., Merz, C.: 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).
17. Martínez Marroquín, E., Vos, C., et al.: Morphological analysis of mammary biopsy images. In: Proceedings of the IEEE International Conference on Image Processing (ICIP'96). (1996) 943–947
18. Martí, J., Cufí, X., Regincós, J., et al.: Shape-based feature selection for microcalcification evaluation. In: Imaging Conference on Image Processing, 3338:1215-1224. (1998)
19. Golobardes, E., Llorà, X., Garrell, J.M., Vernet, D., Bacardit, J.: Genetic classifier system as a heuristic weighting method for a case-based classifier system. Butlletí de l'Associació Catalana d'Intel.ligència Artificial **22** (2000) 132–141
20. Witten, I.H., Frank, E.: Data Mining: practical machine learning tools and techniques with java implementations. Morgan Kaufmann (2000)