Experimental Evaluation of Discretization Schemes for Rule Induction

Jesus Aguilar–Ruiz[1], Jaume Bacardit[2], Federico Divina[3]

[1] Dept. of Computer Science, University of Seville, Seville, Spain
aguilar@lsi.us.es
[2] Intelligent Systems Research Group, Universitat Ramon Llull, Barcelona, Spain
jbacardit@salleURL.edu
[3] Dept. of Computer Science, Vrije Universiteit, Amsterdam, The Netherlands
F.Divina@few.vu.nl

Abstract. This paper proposes an experimental evaluation of various discretization schemes in three different evolutionary systems for inductive concept learning. The various discretization methods are used in order to obtain a number of discretization intervals, which represent the basis for the methods adopted by the systems for dealing with numerical values. Basically, for each rule and attribute, one or many intervals are evolved, by means of ad–hoc operators. These operators, depending on the system, can add/subtract intervals found by a discretization method to/from the intervals described by the rule, or split/merge these intervals. In this way the discretization intervals are evolved along with the rules. The aim of this experimental evaluation is to determine for an evolutionary–based system the discretization method that allows the system to obtain the best results. Moreover we want to verify if there is a discretization scheme that can be considered as generally good for evolutionary–based systems. If such a discretization method exists, it could be adopted by all the systems for inductive concept learning using a similar strategy for dealing with numerical values. Otherwise, it would be interesting to extract relationships between the performance of a system and the discretizer used.

1 Introduction

The task of learning a target concept in a given representation language, from a set of positive and negative realizations of that concept (examples) and some background knowledge, is called inductive concept learning (ICL).

Real life learning tasks are often described by nominal as well as continuous, real-valued, attributes. However, most inductive learning systems treat all attributes as nominal, hence cannot exploit the linear order of real values. This limitation may have a negative effect not only on the execution speed but also on the learning capabilities of such systems.

In order to overcome these drawbacks, continuous-valued attributes are transformed into nominal ones by splitting the range of the attribute values in a finite number of intervals. The so found intervals are then used for treating continuous-valued attributes as nominal. Alternatively, the intervals can be determined during the learning process. This process, called discretization, is supervised when it uses the class labels of examples, and unsupervised otherwise. Discretization can be applied prior or during the learning process (global and local discretization, respectively), and can either discretize one attribute at a time (univariate discretization) or take into account attribute interdependencies (multivariate discretization) [1].

Researchers in the Machine Learning community have introduced many discretization algorithms. An overview of various types of discretization algorithms can be found, e.g., in [2]. Most of these algorithms perform an iterative greedy heuristic search in the space of candidate discretizations, using different types of scoring functions for evaluating a discretization.

In [3–6] various multivariate local discretization methods are introduced and embedded into systems for rules induction. The idea behind the methods is similar. A number of basic discretization intervals are used in order to evolve the best discretization for each rule. A discretization interval for an attribute in a rule is formed by the union of a number of basic discretization intervals.

In this paper we want to experimentally evaluate the effect of using different basic discretization intervals. In order to do this we use a multivariate discretization method inside three evolutionary rule induction systems: HIDER* [5], ECL [7] and GAssist [4]. All these systems take as input a set of discretization intervals, and adapt them during the learning process, by means of ad–hoc genetic operators.

The paper is structured in the following way. In Section 2 we give a brief description of the discretization methods used for finding the basic discretization interavals. Section 3 contains the experimental evaluation of the various discretization methods. First an overview of the rules induction systems is given, then the experiment settings are described and the results of the experiments are presented and discussed. Section 4 summarizes important conclusions and the future work. Finally, in Section 5 some related work is presented.

2 Discretization Methods

All the systems used in this paper treat numerical values locally. Starting from a set of basic discretization intervals the systems evolve the discretization intervals for each rule. At this end some operators are used, which can merge the basic discretization intervals. Thus, at the end of the evolution the discretization intervals present in the evolved rule are the union of n basic discretization intervals, where $n \geq 1$.

The basic discretization intervals are the results of the application of a discretization scheme. In this paper we used the following discretization method for finding the basic discretization intervals:

1. The method used by ID3 [8], as no pruned version of the Fayyad & Irani's algorithm (which is described below). The values of each continuous attribute

A are sorted in increasing order. The midpoints of two successive values of A occuring in examples with different classes are considered as potential cut points. The cut points are then found by recursively choosing the potential cut points that minimizes the entropy, until all the intervals determined in this way contains values relative to examples of the same class;
2. USD [9] divides the continuous attributes in a finite number of intervals with maximum goodness, so that the average-goodness of the final set of intervals will be the highest. The main process is divided in two different parts: first, it calculates the initial intervals by means of projections, which will be refined later, depending on the goodnesses obtained after carrying out two possible actions: to join or not adjacent intervals. The main features of the algorithm are: it is deterministic, does not need any user–parameter and its complexity is subquadratic;
3. Fayyad & Irani's algorithm [10]. This supervised recursive algorithm uses the class information entropy of candidate intervals to select the boundaries of the bins for discretization. Given a set S of instances, an attribute p, and a partition bound t, the class information entropy of the partition induced by t is given by: $E(p,t,S) = Entropy(S_1)\frac{|S_1|}{|S|} + Entropy(S_2)\frac{|S_2|}{|S|}$ where S_1 is the set of instances whose values of p are in the first half of the partition and S_2 the set of instances whose values of p are in the second half of the partition. Moreover $|S|$ denotes the number of elements of S and $Entropy$ is defined as: $Entropy(S) = -p_+ \cdot log_2(p_+) - p_- \cdot log_2(p_-)$ with p_+ and p_- the proportion of positive and negative examples in S respectively.
   For a given attribute p the boundary t which minimizes $E(p,t,S)$ is selected as a binary discretization boundary. The method is then applied recursively to both the partitions induced by the selected boundary t^* until a stopping criterion is satisfied. The MDL principle [11] is used to define the stopping criterion;
4. Random discretizer. In this paper we have considered using a random discretizer as a baseline for the tests. This discretizer selects, for each test, a random subset of all the midpoints between the values in the attribute domain;
5. Equal interval width method. In this method the continuous values are simply divided into n equal sized bins, where n is a parameter. In this paper we consider values of n equal to $5, 10, 15, 20$;
6. Equal frequency method. In this method the continuous values are divided into n bins, each bin containing the same number of values. Thus, the regions of the attribute domain with more density of values have more intervals. Again, n is a parameter, considering for this paper the values $5, 10, 15, 20$;

3 Experimental Evaluation

In this section we first give a brief description of the three rule induction systems used in the experiments. The results of the experiments are then presented and discussed. We have omitted the results from equal–width5, equal–width15, equal–freq5 and equal–freq15 because they are very similar to those described in Table 2 with equal–width10, equal–width20, equal–freq10 and equal–freq20, respectively.

3.1 Rule Induction Systems

ECL [7] is a hybrid evolutionary algorithm for ICL. The systems evolves rules by means of the repeated application of selection, mutation and optimization. The mutation operators applied do not act randomly, but consider a number of mutation possibilities, and apply the one yielding the best improvement in the fitness of the individual. The optimization phase consists in a repeated application of mutation operators until the fitness of the individual does not worsen, or until a maximum number of optimization steps has been reached. In the former case the last mutation applied is retracted.

Numerical values are handled by means of inequalities, which describes discretization intervals. Inequalities can be initialized to a given discretization interval, e.g., found with the application of the Fayyad & Irani's algorithm. ECL can modify inequalities using class information, however for allowing a fair comparison with the other systems, this feature is not used here. Instead, inequalities are modified during the learning process, by mutation operators that can add or subtract a basic discretization interval to the interval described by an inequality.

HIDER* [12] is a tool that produces a hierarchical set of rules. When a new example is going to be classified, the set of rules is sequentially evaluated according to the hierarchy, so if the example does not fulfil a rule, the next one in the hierarchy order is evaluated. This process is repeated until the example matches every condition of a rule and then it is classified with the class that such rule establishes. An important feature of HIDER* is its encoding method [5]: each attribute is encoded with only one gene, reducing considerably the length of the individuals, and therefore the search space size, making the algorithm faster while maintaining its prediction accuracy.

GAssist [4] is a Pittsburgh Genetic–Based Machine Learning system descendant of *GABIL* [13]. It evolves individuals that are ordered variable–length rule sets. The control of the bloat effect is performed by a combination of a rule deletion operator and hierarchical selection [14]. The knowledge representation for real–valued attributes is called Adaptive Discretization Intervals rule representation (*ADI*) [4]. This representation uses the semantics of the *GABIL* rules (Conjuntive Normal Form predicates), but using non–static intervals formed by joining several neighbour discretization intervals. These intervals can evolve through the learning process splitting or merging among them. The representation can also combine several discretizations at the same time, allowing the system to choose the correct discretizer for each attribute. This feature will not be used in this paper, to allow a fair comparison with the other two rule induction systems tested.

The three evolutionary approaches used in this paper are different in the way they look for solutions within the search space. GAssist encodes variable–length individuals, which will represent a whole set of decision rules. ECL and HIDER encode single rules, i.e. each individual is one decision rule. However, ECL finds the entire decision rule set at the final generation, whereas HIDER finds a single rule at the end of the evolutionary process. Therefore, HIDER needs to be run several times, until all the examples are covered by any decision rule, following a sequential covering methodology.

An example of these diffences on encoding is shown in Figure 1. We have selected a simple rule set composed by only two rules from Wisconsin dataset. The genetic representation of this rule set is illustrated for each system. The cutpoints have been obtained with ID3.

Attributes in GAssist rules codify the full attribute domain as one or many intervals. Each interval is formed by a subset of consecutive basic discretization intervals. The semantical definition of the rule is formed by the intervals with value 1. HIDER encodes every attribute with only one natural number, as it is described in [5]. Every possible interval defined by two cutpoints is associated to a natural number, so genetic operators are designed to handle this method efficiently. ECL uses a high level representation, where a rule is represented as a list of predicates, variables, constants and inequalities.

3.2 Experiments Settings

Table 1 shows the features of the datasets used in the experiments. These datasets were taken from the UCI Machine Learning repository [15]. We have chosen these datasets because they contain only numerical attributes, and no nominal attributes. For this reason they represent a good testing for the discretization schemes.

In the experiments a 10–fold cross–validation is used. Each dataset is divided in ten disjoint sets of approximately similar size; one of these sets is used as test set, and the union of the remaining nine forms the training set.

For the random discretization method, we have run the 10–fold cross–validation 15 times with different random seeds. Therefore, 7 datasets, with 8

Table 1. Features of the datasets used in the experiments. For each dataset the number of examples and the number of continuous attributes is given.

Code	Name	Examples (+,-)	Continuous
ION	ionosphere	351 (225,126)	34
LIV	liver	345 (145,200)	6
PIM	pima-indians	768 (500,268)	8
SON	sonar	208 (97,111)	59
WD	wdbc	569 (212,357)	30
WIS	wisconsin	699 (458,241)	10
WP	wpbc	198 (47,151)	33

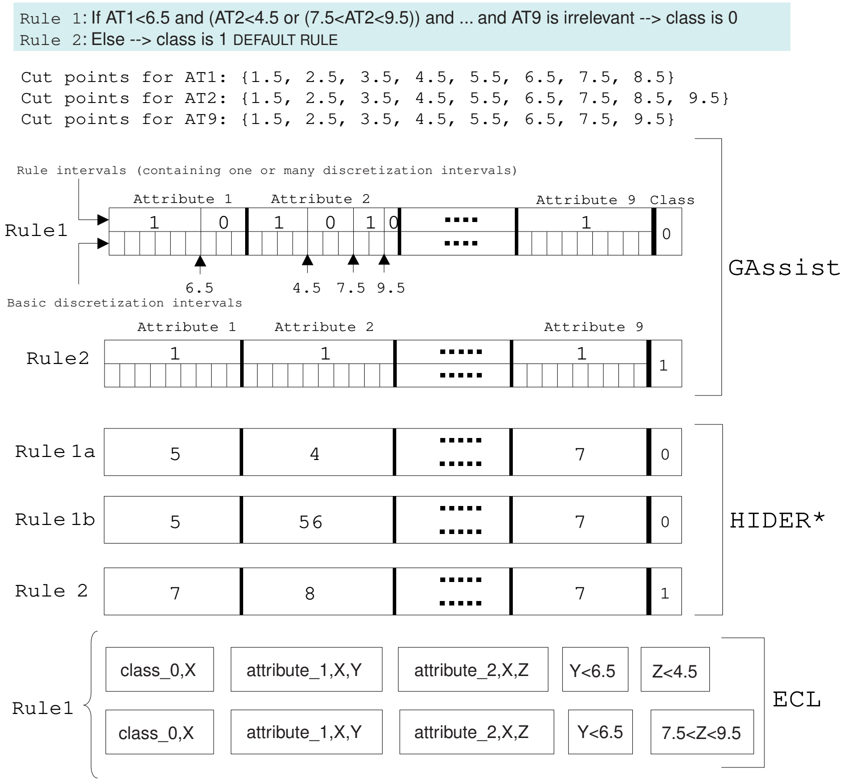

Fig. 1. Example from Wisconsin dataset. Each system encodes rule sets differently.

discretization methods (one of them 15 times) and using 10–fold cross–validation means 1540 runs for each system.

Each system uses its usual configuration settings, defined in previous work [4–6]. Common values of the population size, generations, etc. are not suitable in this case because we are testing systems with diverse structure (One *GA* run generating a complete rule set vs. sequential *GA* runs learning one rule at a time). As a consequence, the search space size for each system can vary, and this leads to each system needing a specific set of parameters.

3.3 Results

We here report the results obtained by the three systems on each dataset for all the discretization schemes. We also report the average performance achieved by each discretization schemes, as a way of summarizing the results.

Table 2 reports the results obtained by ECL, *GAssist* and HIDER*, respectively. For each dataset, the average accuracy and the average number of rules are reported for each discretization method used for obtaining the basic discretization intervals, including standard deviations.

Table 2. Average accuracies (Acc.) and average number of rules contained in the solution (# rules) obained by each sistems on the datasets with the different discretization methods. Best and worst results are highlighted. EW stands for equal width, EF stands for equal frequency.

ID	System		ID3	USD	Fayyad	Rand	EW10	EW20	EF10	EF20
ION	ECL	Acc.	**88.1±4.4**	74.4±4.9	87.4±3.1	**69.8±4.0**	71.6±4.5	73.3±5.0	72.6±4.3	71.2±5.4
		# rules	12.5±1.5	15.5±0.7	9.8±1.3	14.7±1.7	18.2±1.8	15.5±5.7	12.6±1.7	12.5±1.8
	GAssist	Acc.	90.4±5.0	90.7±4.9	**93.3**±4.3	90.2±5.1	92.3±4.1	92.5±3.7	90.0±4.8	**89.5**±4.5
		# rules	3.4±1.2	3.5±1.3	2.3±0.7	4.2±1.5	2.9±1.0	2.7±0.9	4.0±1.4	4.0±1.4
	HIDER*	Acc.	74.3±7.7	74.9±7.2	**89.5±5.6**	70.1±3.6	**58.4±11.5**	60.0±12.9	86.7±5.2	83.8±3.1
		# rules	33.2±1.9	32.3±2.5	2.0±0.0	5.6±2.8	1.0±0.0	1.1±0.3	10.1±3.4	22.0±2.6
LIV	ECL	Acc.	61.5±5.0	**66.6±4.5**	63.2±4.5	57.9±4.9	58.3±4.5	59.4±4.8	**57.1±4.4**	59.2±3.4
		# rules	13.5±1.7	11.3±1.3	1.3±0.7	12.0±1.5	7.9±1.8	8.6±1.7	9.3±1.8	12.7±1.3
	GAssist	Acc.	**65.5**±7.8	65.0±7.6	**59.5**±6.2	64.5±8.6	63.7±7.7	63.7±8.3	64.3±7.9	65.3±7.9
		# rules	9.1±2.0	8.7±1.9	2.9±1.4	8.2±1.9	7.7±1.8	8.0±1.7	8.8±2.0	9.3±2.2
	HIDER*	Acc.	58.3±6.5	**65.2±3.9**	61.2±5.1	**51.9±4.5**	59.4±6.9	58.6±7.5	60.6±7.1	62.4±5.8
		# rules	3.9±0.5	4.3±0.6	1.8±0.8	4.1±0.7	3.0±0.0	3.0±0.0	5.0±0.6	4.4±0.8
PIM	ECL	Acc.	**71.4±1.9**	69.7±4.2	68.3±2.3	**60.9±4.5**	63.2±2.6	62.4±1.9	63.9±1.5	61.7±3.9
		# rules	75.5±5.2	20.9±2.1	2.2±0.6	72.5±7.5	25.0±1.6	24.1±1.0	21.5±3.4	37.7±7.3
	GAssist	Acc.	73.9±3.9	73.7±4.0	**72.5**±4.6	73.6±4.4	73.6±4.3	73.8±4.7	73.6±4.0	**74.3**±3.8
		# rules	5.4±0.8	5.3±0.7	5.1±0.5	5.2±0.6	5.3±0.8	5.2±0.6	5.5±1.1	5.3±0.8
	HIDER*	Acc.	**74.2±1.8**	**74.2±4.1**	72.8±2.8	**71.7±2.6**	73.3±3.6	73.7±3.0	74.1±3.1	72.4±4.0
		# rules	5.8±0.6	4.9±0.5	2.7±1.2	4.2±0.8	4.1±0.8	3.8±0.7	4.2±0.6	4.4±1.1
SON	ECL	Acc.	64.4±5.1	72.0±6.9	**76.4±2.1**	**63.1±4.8**	69.3±5.9	66.4±5.1	72.4±4.0	68.3±4.2
		# rules	30.0±3.6	17.9±5.8	3.2±0.6	29.1±8.8	9.9±3.6	14.8±4.1	13.6±4.7	21.4±7.4
	GAssist	Acc.	73.3±9.8	73.4±9.1	74.5±9.4	73.1±10.2	74.3±10.1	**74.8**±8.8	72.8±9.6	**72.4**±9.8
		# rules	8.8±2.1	8.9±2.1	6.4±0.7	8.7±2.1	8.3±1.7	8.6±2.0	9.3±2.0	9.7±2.1
	HIDER*	Acc.	68.8±4.7	68.7±6.6	**72.6±6.5**	66.4±4.4	**62.4±11.4**	64.8±8.6	66.9±12.6	64.7±10.2
		# rules	30.9±3.0	24.4±3.1	4.4±1.3	16.5±3.9	11.6±1.9	14.6±0.9	60.3±5.3	63.2±3.4
WD	ECL	Acc.	91.4±4.3	93.3±4.1	**94.2±3.1**	**88.2±4.8**	91.2±3.6	89.3±5.3	89.9±5.7	90.0±4.3
		# rules	15.5±5.7	22.7±2.8	5.6±2.0	24.5±13.6	6.2±2.1	7.4±2.8	7.0±2.4	9.3±4.1
	GAssist	Acc.	**94.2**±3.1	93.9±3.1	94.0±3.1	93.7±3.5	**93.6**±3.2	94.0±3.2	**94.2**±3.1	94.0±3.1
		# rules	4.0±1.1	3.9±1.1	3.7±0.8	4.2±1.3	3.5±0.8	3.6±0.8	4.0±1.0	4.1±1.2
	HIDER*	Acc.	**85.6±5.9**	92.8±6.1	**93.5±1.5**	88.8±2.5	91.3±3.5	89.6±4.9	90.7±3.2	88.3±5.0
		# rules	22.3±2.1	16.8±1.0	3.9±0.7	14.3±1.5	4.9±0.5	5.5±0.5	7.9±1.3	11.2±1.2
WIS	ECL	Acc.	94.7±2.2	**95.6±2.4**	93.4±2.2	94.7±2.4	94.6±2.6	95.0±2.6	93.8±2.7	**93.3±3.0**
		# rules	13.7±2.1	3.3±0.8	6.1±1.3	5.6±1.6	11.4±2.3	14.5±2.1	11.3±2.8	11.2±2.7
	GAssist	Acc.	95.4±2.4	**96.0**±2.2	**95.1**±2.5	**95.1**±2.5	95.8±2.3	95.9±2.2	95.7±2.2	95.8±2.0
		# rules	2.5±0.7	2.2±0.5	3.3±0.5	2.6±0.6	2.3±0.6	2.4±0.6	2.7±0.7	2.5±0.6
	HIDER*	Acc.	96.4±2.3	**96.6±1.8**	95.8±2.4	**93.4±2.0**	96.3±2.1	96.4±2.0	96.4±1.6	95.4±2.1
		# rules	3.7±0.6	2.0±0.0	4.0±0.0	2.5±0.8	3.5±0.8	3.7±0.6	3.7±0.6	3.9±1.1
WP	ECL	Acc.	**76.9±3.5**	74.5±3.6	76.4±3.7	74.2±5.1	72.9±4.0	72.8±4.2	**70.2±5.4**	76.5±5.8
		# rules	21.6±2.1	20.6±1.7	2.2±0.6	21.3±0.9	15.5±2.3	20.2±2.3	19.6±1.1	20.3±2.1
	GAssist	Acc.	**72.7**±7.4	72.8±8.6	74.0±3.5	74.2±7.4	**75.7**±6.7	75.3±7.3	75.2±7.2	74.5±7.6
		# rules	5.5±1.6	5.4±1.4	2.0±0.2	5.2±1.3	3.8±1.4	3.6±1.3	4.5±2.3	5.0±2.0
	HIDER*	Acc.	69.2 ±6.5	**74.9±2.8**	72.9±3.1	**62.2±7.1**	74.3±3.6	73.8±9.8	68.0±8.3	67.7±7.8
		# rules	18.0±1.4	14.8±2.1	14.6±1.8	13.5±1.5	3.6±1.1	5.2±1.0	11.2±1.2	16.2±2.1

3.4 Analysis of Results

The aim of this analysis is to show if there is one or a group of discretization methods that present better performance than the others for evolutionary–based learning systems. It is important to note that the output of each discretization method is handled as a set of boundary points to define conditions over attributes in the decision rules provided by the evolutionary approaches. Those outputs will define the search space. The quality of solutions generated by the evolutionary systems will depend on how many and how good are those boundary points, and how well the evolutionary systems are able to join discretization intervals by means of genetic operators.

From Table 2, it can be noticed that ECL is sensitive to the discretization method used for obtaining the basic discretization intervals, thus the difference in the quality of the solutions depends strongly on the discretization method used.

ECL obtained the worst results when the random discretizer is used for determining the basic discretization intervals. This is due to the mutation operators used for modifying discretization intervals inside rules. In fact, these operators can add or subtract only one basic discretization interval at a time to the intervals the operators are applied to. In this way, if the number of basic discretization intervals is high, like in the case of the random discretizator, it is likely that the individuals are evolved very slowly. This is because only a little change in the discretization intervals can be applied by each mutation. Moreover, if this mutation was applied during the optimization phase, and had negative effects on the fitness of the individual, then the mutation is retracted. This cause the production of individuals that are too specific. However, when there are few discretization intervals, it is more likely that ECL can establish a good discretization interval for a rule.

In general ECL encountered problems with unsupervised discretizers that produces many intervals, e.g., the random discretizer, while it produces good results when the basic discretization intervals were produced with supervised discretizers. And the worst results are always obtained when the basic discretization intervals are determined by an unsupervised discretization method. On average ECL obtained the best results when the Fayyad & Irani's algorithm was used for producing the basic discretization intervals.

As far as the number of rules is concerned, in general ECL obtained the simpliest results when the Fayyad & Irani's algorithm was used. This is due to the fact the this algorithm produces a limited number of basic discretization intervals, and ECL can not generate the same diversity of rules that it can generate when other discretization methods that produces more basic discretization intervals, e.g., the ID3 method, are used.

The results for GAssist show a different behavior. Looking at the results we can see that the sensitivity to the discretizer is much smaller than in ECL or HIDER* . Moreover, there is not a clear correlation between the number of cut points of each discretizer and the performance of the system, showing that GAssist can explore successfully the search space.

Table 3. Average results for each discretization method. From left to right, average accuracy, average average number of rules in the solutions, number of times the discretizator obtained the best and worst performance, rank of the discretizator and average number of cut points per attribute.

Method	Avg. Accuracy	Avg. # rules	# best	# worst	Rank	Cut points
ID3	78.1±11.9	16.1±16.3	6	2	3	83.2±48.1
USD	79.0±11.1	11.9±8.6	7	0	2	72.1±42.4
Fayyad	80.0±12.1	4.3±3.0	6	2	1	1.3±1.5
Random	75.1±13.2	13.3±15.1	0	9	7	106.1±97.5
EW10	76.5±13.2	7.6±5.9	1	3	6	9
EW20	76.5±13.0	8.4±6.3	1	0	6	19
EF10	77.6±12.4	11.2±12.0	1	2	4	9
EF20	77.2±12.1	13.8±13.8	1	3	5	19

Another important observation is that the discretizer performing worst in GAssist is the Fayyad & Irani's one, which was the best method for ECL. The reason of the poor performance of this discretizer in GAssist is due to the *hierarchical selection* operator used to control the *bloat effect* [16]. This operator introduces a bias towards compact individuals, in both rules and intervals (as this system uses rules in Conjunctive Normal Form, the number of intervals per rule can vary). While this feature is usually good, the difference between a well generalized solution and a too simple one is very small, specially if we use a discretizer that generates few cut points (like the Fayyad & Irani's). Therefore, it is easier to get the system stuck in a local optimum.

The group of supervised discretization methods have better performance with HIDER* than the others. HIDER* expands the limits of intervals within rules slowly, similar to ECL, so it needs the boundary points to be accurate, i.e. they might be possible interval limits for conditions in decision rules. As ID3 and USD generates more intervals, boundary points can be more precise but the evolutionary system needs more time to find them. HIDER* has a specific mutation operator to remove attributes from rules, so when there are a lot of attributes it is more difficult to remove any if the number of intervals is high (ionosphere and sonar datasets are good examples). In contrast, when the number of discretization intervals is small, as provided by Fayyad & Irani's method, results can be better.

Finally, it is at least interesting to remark the performance of the random discretizer in GAssist, as it does perform quite well. Probably the reason is that the average number of cut points used across all domains was 106.1. This number is large enough to minimize the loss of information intrinsical in any discretization algorithm.

For summarizing the results, we propose two tables. In Table 3, we present the average results obtained by the discretization methods on all datasets. We also report the number of time a discretizator obtained the best and the worst performance on a dataset for a system. For instance, ID3 obtained six times the best results and three times the worst results. It can be seen that the supervised methods performed better than the unsupervised methods. In particular the random discretizer obtained the worst performance.

The second table, Table 4, proposes a ranking of the discretization methods. For each row of results from Table 2, the results are ranked. At the end of the table the average ranks are computed and the final rank is assigned to the discretization methods. From the Table 4 it emerges that USD results as the best discretizer. This is because USD has a stable behavior. USD never obtained the worst results, while in seven cases (out of 21, three methods by seven datasets) it obtained the best results. The worst rank is assigned to the random discretizer. In fact it never obtained the best results, and in nine cases it obtained the worst ones.

Table 4. Ranking of the discretization methods. For each dataset the single results obtained by the sistems with each discretization methods are ranked. At the end the average of the ranks is computed and the final rank is assigned. E, G and H mean ECL, GAssist and HIDER*, respectively.

Discretizer	ION			LIV			PIM			SON			WD			WIS			WP			Average	Rank
	E	G	H	E	G	H	E	G	H	E	G	H	E	G	H	E	G	H	E	G	H		
ID3	1	5	5	3	1	7	1	2	1	7	5	2	3	1	8	3	6	2	1	8	5	3.67	3
USD	3	4	4	1	3	1	2	4	1	3	4	3	2	6	2	1	1	1	4	7	1	2.76	1
Fayyad	2	1	1	2	8	3	3	8	6	1	2	1	1	3	1	7	7	6	3	6	4	3.62	2
Random	8	6	6	7	4	8	8	5	8	8	6	5	8	7	6	3	8	8	5	5	8	6.52	8
EW10	6	3	8	6	6	5	5	5	5	4	3	8	4	8	3	5	3	5	6	1	2	4.81	6
EW20	4	2	7	4	6	6	6	3	4	6	1	6	7	3	5	2	2	2	7	2	3	4.19	4
EF10	5	7	2	8	5	4	4	5	3	2	7	4	6	2	4	6	5	2	8	3	6	4.67	5
EF20	7	8	3	5	2	2	7	1	7	5	8	7	5	3	7	8	3	7	2	4	7	5.14	7

4 Conclusions and Future Work

In general, all of the systems provided good performance for all of the discretization methods. However, GAssist differs from ECL and HIDER* with respect to the representational methodology. GAssist generates one solution with a set of decision rules, while HIDER* uses a sequential covering technique to find one rule at a time, removing examples covered by previous rules, and ECL evolves a population of rules from which a subset of rules representing the final solution is extracted.

It seems that the performance of Pittsburgh–based methodology is more stable, especially when the number of attributes and discretization intervals is high, which leads to a better exploration of the search space. This means that GAssist is less sensitive to the effect of discretization choices. On the other hand, supervised discretization methods, e.g., ID3, USD and Fayyad & Irani's algorithm, seem more appropriate for evolutionary algorithms where an individual encodes a single rule, which simplifies the search space. This reason justifies the fact that neither ECL or HIDER* find better results with non–supervised discretization methods.

There is no doubt that intrinsic properties of datasets might have influence on the results, so our future research directions will include to analyze the relationship between the discretizer outputs and the dataset features, and also, how this can give us a clue to choose the evolutionary approach (Pittsburgh–based approach or sequential covering methodology).

5 Related Work

Discretization is not the only way to handle real–valued attributes in Evolutionary Computation–based Machine Learning systems. Some examples are induction of decision trees (either axis–parallel or oblique), by either generating a full tree by means of genetic programming operators [17] or using a heuristic method to generate the tree and later a Genetic Algorithm or an Evolutionary Strategy to optimize the test performed at each node [18]. Other examples are inducing rules with real–valued intervals [19] or generating an instance set used as the core of a *k-NN* classifier [17]. Other approaches to concept learning, e.g., Neural Network, do not need any discretization for handling numerical values.

Also, several discretization algorithms are reported in the literature. Some examples not tested in this paper are the Mántaras discretizer [20] which is similar to that of Fayyad & Irani's, but using a different formulation of the entropy minimization. Another example is ChiMerge [21]. This discretizer creates an initial pool of cut points containing the real values in the domain to discretize, and iteratively merges neighbour intervals that make true a certain criterion based on the χ^2 statistical test.

Acknowldegments

This work was partially supported by the Spanish Research Agency Comisión Interministerial de Ciencia y Tecnología (CICYT) under Grants TIC2001-1143-C03-02, TIC2002-04160-C02-02 and TIC2002-04036-C05-03. The second author acknowledges the support provided by the "Departament d'Universitats, Recerca i Societat de la Informació de la Generalitat de Catalunya" under grants 2001FI 00514 and 2002SGR 00155.

References

1. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: International Conference on Machine Learning. (1995) 194–202
2. Liu, H., Hussain, F., Tan, C., Dash, M.: Discretization: An enabling technique. Journal of Data Mining and Knowledge Discovery **6** (2002) 393–423
3. Bacardit, J., Garrel, J.M.: Evolution of adaptive discretization intervals for a rule-based genetic learning system. In: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann Publishers (2002) 677
4. Bacardit, J., Garrel, J.M.: Evolving multiple discretizations with adaptive intervals for a pittsburgh rule-based genetic learning classifier system. In: GECCO 2003: Proceedings of the Genetic and Evolutionary Computation Conference, Springer (2003) 1818–1831

5. Giráldez, R., Aguilar-Ruiz, J., Riquelme, J.: Natural coding: A more efficient representation for evolutionary learning. In: GECCO 2003: Proceedings of the Genetic and Evolutionary Computation Conference, Chicago, Springer-Verlag Berlin Heidelberg (2003) 979–990
6. Divina, F., Keijzer, M., Marchiori, E.: A method for handling numerical attributes in GA-based inductive concept learners. In: GECCO 2003: Proceedings of the Genetic and Evolutionary Computation Conference, Chigaco, Springer (2003) 898–908
7. Divina, F., Marchiori, E.: Evolutionary concept learning. In: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, Morgan Kaufmann Publishers (2002) 343–350
8. Quinlan, J.R.: Induction of decision trees. In Shavlik, J.W., Dietterich, T.G., eds.: Readings in Machine Learning. Morgan Kaufmann (1986) Originally published in *Machine Learning* 1:81–106, 1986.
9. Giraldez, R., Aguilar-Ruiz, J., Riquelme, J., Ferrer-Troyano, F., Rodriguez, D.: Discretization oriented to decision rules generation. Frontiers in Artificial Intelligence and Applications **82** (2002) 275–279
10. Fayyad, U., Irani, K.: Multi-interval discretization of continuos attributes as preprocessing for classification learning. In: Proceedings of the 13th International Join Conference on Artificial Intelligence, Morgan Kaufmann Publishers (1993) 1022–1027
11. Rissanen, J.: Stochastic Complexity in Statistical Inquiry. World Scientific, River Edge, NJ. (1989)
12. Aguilar-Ruiz, J., Riquelme, J., Toro, M.: Evolutionary learning of hierarchical decision rules. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **33(2)** (2003) 324–331
13. 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
14. Bacardit, J., Garrell, J.M.: Bloat control and generalization pressure using the minimum description length principle for a pittsburgh approach learning classifier system. In: Proceedings of the 6th International Workshop on Learning Classifier Systems, (in press), LNAI, Springer (2003)
15. Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
16. Langdon, W.B.: Fitness causes bloat in variable size representations. Technical Report CSRP-97-14, University of Birmingham, School of Computer Science (1997) Position paper at the Workshop on Evolutionary Computation with Variable Size Representation at ICGA-97.
17. 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
18. Cantu-Paz, E., Kamath, C.: Inducing oblique decision trees with evolutionary algorithms. IEEE Transactions on Evolutionary Computation **7** (2003) 54–68
19. Stone, C., Bull, L.: For real! xcs with continuous-valued inputs. Evolutionary Computation Journal **11** (2003) 298–336
20. De Mántaras, R.L.: A distance-based attribute selection measure for decision tree induction. Machine Learning **6** (1991) 81–92
21. Kerber, R.: Chimerge: Discretization of numeric attributes. In: Proc. of AAAI-92, San Jose, CA (1992) 123–128