An Approach to Reduce the Cost of Evaluation in Evolutionary Learning*

Raúl Giráldez, Norberto Díaz-Díaz, and Isabel Nepomuceno,
and Jesús S. Aguilar-Ruiz

Department of Computer Science, University of Seville,
Avenida Reina Mercedes s/n, 41012 Sevilla, Spain
{giraldez, ndiaz, isabel, aguilar}@lsi.us.es

Abstract. The supervised learning methods applying evolutionary algorithms to generate knowledge model are extremely costly in time and space. Fundamentally, this high computational cost is fundamentally due to the evaluation process that needs to go through the whole datasets to assess their goodness of the genetic individuals. Often, this process carries out some redundant operations which can be avoided. In this paper, we present an example reduction method to reduce the computational cost of the evolutionary learning algorithms by means of extraction, storage and processing only the useful information in the evaluation process.

1 Introduction

Machine Learning is used when we want to build a knowledge model from a training dataset and predict the outcome of a new unseen instance. When the class of the training data is known, we work in the Supervised Learning field. There are several methods and algorithms in the specific literature that extract the inherent knowledge to a set of labelled data. A large number of these methods (Hider [1, 11], Cn2 [6], Rise [9], Oc1 [15], Gabil [7], GAssist [3], Gil [13], Sia [17], Ecl [8], etc.) use probabilistic algorithms to search solutions that able to model the behaviour of data. When the learning process is carried out by applying techniques of evolutionary computation, particularly evolutionary algorithms, it is called evolutionary learning, which is the framework of our approach.

The evolutionary learning methods usually evaluate the rules directly from the database. That is to explore such database sequentially, taking each of the examples and testing the quality of the rule through the correct classification of those examples. We can see, therefore, that the learning process of these systems is very costly in terms of time and space. Some approaches is focused on improving the learning process in order to reduce its computational cost by applying methods of incremental learning and *windowing* [4] or techniques of

* This research was supported by the Spanish Research Agency CICYT and European FEDER Funds, under grants TIN2004–00159 and TIN2004–06689–C03–03.

J. Cabestany, A. Prieto, and D.F. Sandoval (Eds.): IWANN 2005, LNCS 3512, pp. 804–811, 2005.

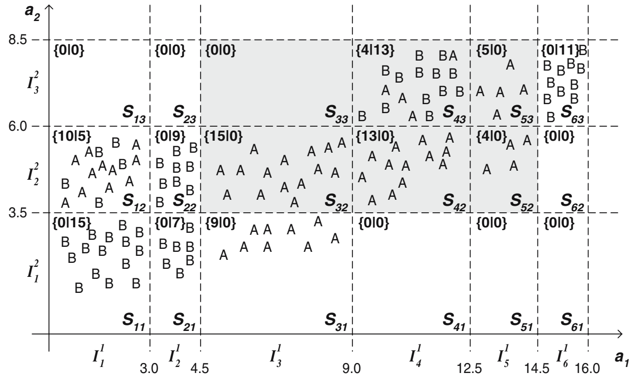


Fig. 1. Example

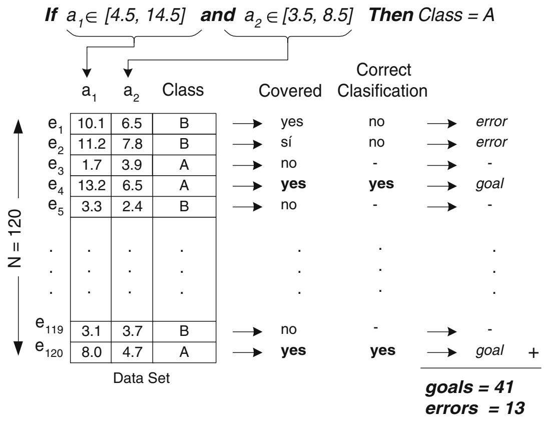
scalability [16]. Nevertheless, an appropriated organization of the information can also contribute to a reduction of the computing time. In this sense, the structure named EES [12] allows us to process only those examples, the values of which are covered by such rule will be processed, and not the totality of the database. However, although this solution reduces the computational cost, the use of the EES reflects a redundant process of the data for the rules sharing regions in the space.

The aim of this research is to avoid repetitive counting of examples during the evaluation process of the individuals of the population. In this paper, we propose a preprocessing method which extracts the useful information from the data to be used in the evaluation and stores it in a format that allows an efficient access of such data.

2 Motivation

One of the critical factors related to applying the evolutionary algorithms in supervised learning is the evaluation of the individuals of the population. Usually, each individual represents one or several rules as potential solutions to the problem. In this work, we assume that each individual codifies only one rule, however, our approach can be easily adapted to individual with variable length. The evaluation function measures the goodness of each individual of the population. This goodness is based on the number of goals (number of examples correctly classified) and errors (number of misclassified examples) that the encoded rule has during the classification of the examples from the dataset.

Gabil and Gil are named *concept learner*, since they handle with discrete attributes exclusively. Other advanced tools, like Hider, GAssist or Ecl can treat continuous and discrete attributes thanks to a discretization algorithm that diminishes the cardinality of the set of values of the continuous attribute. Thus, the rules can only establish conditions using a finite set of intervals. We


If $a_1 \in$ [4.5, 14.5] and $a_2 \in$ [3.5, 8.5] Then Class = A
N = 120
Data Set
+
goals = 41
errors = 13

	a_1	a_2	Class	Covered	Correct Clasification	
e_1	10.1	6.5	B	yes	no	error
e_2	11.2	7.8	B	sí	no	error
e_3	1.7	3.9	A	no	-	-
e_4	13.2	6.5	A	**yes**	**yes**	goal
e_5	3.3	2.4	B	no	-	-
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
e_{119}	3.1	3.7	B	no	-	-
e_{120}	8.0	4.7	A	**yes**	**yes**	goal

Fig. 2. Linear Evaluation

name these *atomic intervals*, because once the intervals are obtained, they can not be split. The discretization process turns the initial search space, that it is theoretically infinite, into a finite space of solutions. Figure 1 shows an example of a dataset with 120 examples, two classes (A and B) and two continuous attributes. The discretization method has obtained 6 intervals (I_i^1) for At_1 and 3 intervals (I_j^2) for At_2. Each pair of intervals $(I_i^1,\ I_j^2)$ defines a subspace or region $(\mathcal{S}_{ij})$. The values in brackets $\{\varepsilon_A|\varepsilon_B\}$ represent the number of examples of each class in the correspondent region. These subspaces can be linked in a rule, but they can not be split, since their decision bounds are given by atomic intervals. Consequently, these regions are called *atomic subspaces.*

Evaluation by means of a linear search processes each and every one of the examples in the database independently of the conditions established by the rule. The computational cost of a individual evaluation[1] is $\Theta(Nm)$, where N is the number of examples and m is the number of attributes in the database. For example, the rule

$$\textbf{\textit{If}}\ a_1 \in [4.5,\ 14.5]\ \textbf{\textit{and}}\ a_2 \in [3.5, 8.5] \Rightarrow \textit{Class=A}$$

is represented by the shaded area in Figure 1 and it means that if an example belongs to the subspace $\{I_3^1 \cup I_4^1 \cup I_5^1\} \cap \{I_2^2 \cup I_3^2\}$, such example must be classified as A. In order to count the goals and errors for this rule, each example of the dataset is analysed. If the example is covered, that is it fulfills the conditions of the rule, then the class is compared, as Figure 2 illustrates. In this case, the rule has 41 goals and 13 errors.This process is repeated for each individual, which means an unnecessary computational cost due to two aspects principally:

[1] Evaluation cost for only one individual.

1. Redundant count of examples for those space areas shared by some rules.
2. The whole space exploration to evaluate rules which only cover a part of such space.

In general, since the cost of the individual evaluation is very high, normally it is tried to reduce some of the two parameters which take part in this: N and m. The techniques that reduce the number of attributes m are commonly named *feature selection methods* [14], and their goal is to remove those attributes which are irrelevant and/or harmful for learning. In other way, the example pruning methods are included into the *instance editing techniques* [18], and they are focused on reduce the size of dataset (N). This work is framed in these last techniques. We propose an instance reduction method that benefits from the methodology followed by the typical evolutionary algorithms for rule discovery.

3 Example Reduction Method

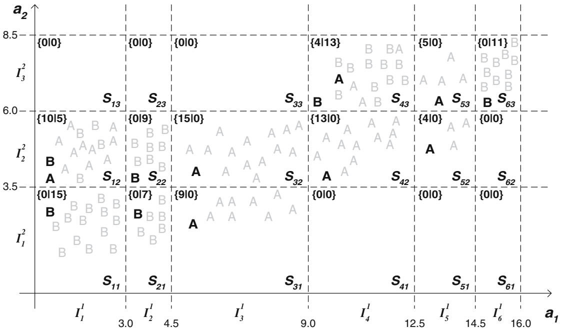
As we mentioned before, the atomic subspaces can not be divided. Therefore, although examples which belong to a same subspace can be different syntactically, from the point of view of the individual evaluation they are semantically similar. This fact makes possible to count how many examples of each class coexist in each atomic subspace and to store these values for its later utilization in evaluation process. This idea is put into practice in a novel editing method that removes all those examples that result redundant for evaluation.

To explain our proposal in a clear way, we are going to use the example showed in Figures 1 and 2, to later generalize the solution to data collections with any kind of attributes and greater number of classes.

3.1 Algorithm

The aim of the editing method is to reduce the number of examples of the dataset $\mathcal{D}$ to obtain a subset $\mathcal{D}^* \subseteq \mathcal{D}$ which contains the same knowledge that $\mathcal{D}$, but with a smaller number of examples N^*. Initially, we begin from a dataset $\mathcal{D}$ with a number of examples N, where continuous attributes have already been discretized. From this discretization results a set of atomic intervals per attribute that define the atomic subspaces. Each example $e = (a_1, a_2, \ldots, a_m | c)$ is made up by a collection of attributes and a class (*v.g,* $e_1 = ($*10.1, 6.5* $| B)$. For each atomic subspace $\mathcal{S}_{ij}$, the instances of each class $\{\varepsilon_A | \varepsilon_B\}$ are counted and they are chosen as many representative examples (e^c_{ij}) as different class coexist in the subspace. These representative examples[2] are added to the reduced dataset $\mathcal{D}^*$. Each representative example have the same form that an original example, but we add a weight ω^c_{ij} equal to the ε_c which counts the instances of the class c in the subspace $\mathcal{S}_{ij}$. The regions with some ε_c equal to 0 do not have representative in $\mathcal{D}^*$ for the class c. For example, in Figure 1, $e^A_{43} = ($*10.5, 7.2* $|A, 4)$ and

[2] Although these representatives not have to coincide with some original example, for simplicity, we choose the first which is in the data set.



N* = 13

	aa_1	a_2	Class	w^c_{ij}
e^B_{11}	0.5	2.8	B	15
e^A_{12}	0.5	3.9	A	10
e^B_{12}	0.5	4.3	B	5
e^B_{21}	3.3	2.5	B	7
e^B_{22}	3.2	3.7	B	9
e^A_{31}	5.3	2.2	A	9
e^A_{32}	5.7	4.0	A	15
e^A_{42}	9.6	3.9	A	13
e^A_{43}	10.5	7.2	A	4
e^B_{43}	10.1	6.5	B	13
e^A_{52}	13.0	4.7	A	4
e^A_{53}	13.2	6.5	A	5
e^B_{63}	14.8	6.4	B	11

Reduced Dataset D*

Fig. 3. Reduction for example of Figures 1 and 2

$e^A_{43} = (10.1,\ 6.5\ |B, 13)$ represent the subspace $\mathcal{S}_{43}$, whereas $e^A_{53} = ($*13.2, 6.5* $|A, 5$*)* is the only representative for the subspace $\mathcal{S}_{53}$. For the empty regions, like $\mathcal{S}_{13}$, $\mathcal{D}^*$ does not contain any example.

Figure 3 shows the result obtained by this editing method for the example explained in Figure 1. Representative examples of each atomic subspace are displayed in bold type, whereas removed examples appear with a lighter color. The reduced dataset ($\mathcal{D}^*$) are shown to the right and it will be used to the evaluate process. As we can see, the original dataset, with $N = 120$ examples, has been replaced by $\mathcal{D}^*$, with $N^* = 13$ weighted examples.

Once editing process has finished, the evaluation of the individuals can be carried out in a linear way, as it was shown by Figure 2, but now, the number of examples smaller. Note that it is necessary to take into account the examples' weight when the goals and errors are counted, since each $e^c_{ij} \in \mathcal{D}^*$ represents to ω^c_{ij} examples in $\mathcal{D}$. This method solves only the first of the problems mentioned in Section 2, since it continues being needed a linear search through $\mathcal{D}^*$. However, our approach is very simple to apply and it achieves satisfactory experimental results, as Section 4 shows.

Notice that the generalization of the methods for k class and m attribute is trivial. Simply we would have a collection of class accountant $\{\varepsilon_{c_1}|\varepsilon_{c_2}|\ldots|\varepsilon_{c_k}\}$; and a indexes collection to denote a subspace $(\mathcal{S}_{i_1\ldots i_m})$ or example $(e^c_{i_1\ldots i_m})$.

3.2 Discrete Attributes

By applying our proposal for data set with discrete attributes is similar to the previous one for continue attributes, although we should emphasize some important peculiarities.

Continuous attributes, though they are discretized, often define a space much more complex than the discrete ones, principally due to two reasons: first, the number of intervals is normally larger in real applications, which multiplies the number of subspaces; and second, the regions usually include more than one example, that is, there are examples with the same semantical meaning. Thus, our proposal has a priori more justification when the dataset contains continuous attributes.

When the dataset contains only discrete attributes and there are not repeated examples, each atomic subspace contains at the most one example only, that is, the editing process would not produce reduction in the number of examples. However, although the multiplicity of examples can look like not much habitual, it is relatively common. For example, the application of some feature selection method can cause that some examples are identical if those attributes that distinguish them was eliminated. Another clear example is given when there is noise in the dataset. In this case we can remove the repeated examples in a same atomic subspace by setting w_{ij}^{c} to 1. However, this solution is not advisable because we could be eliminating useful information for the learning.

Therefore, the our approach is favourable whenever the dataset contains similar examples from the point of view of the learning process, otherwise it does not reduce the size of data. Anyway, our method does not cause a significant increment in the computational cost with regard to the evolutionary algorithm, and we advise its use when there is no previous information about the multiplicity of dates.

4 Empirical Results

In order to show the reduction of the computational cost of the evaluation process, we have designed the following experiments with some datasets from UCI Repository [5]. The the evolutionary tool used was Hider [1,11], that generates a set of hierarchical decision rules from a labeled dataset. This tool uses its own discretization method, named Usd [10], before running the evolutionary algorithm that obtains the rules. Thus, the editing method must be applied after the discretization and before the learning process. This algorithm required some changes, though minimal, to adapt the evaluation of the individuals to the new dataset with weight. To check that the editing does not damage the accuracy of the rules, a 10-fold cross-validation was achieved with each dataset. In this sense, it is important to point out that the accuracy and complexity of the knowledge models resulted similar by using the editing method and without it.

The datasets used were: *Breast Cancer*, *Bupa*, *Cleveland*, *Glass*, *Hayes Roth*, *Heart*, *Hepatitis*, *Horse Colic*, *Iris*, *Led7*, *Pima Diabetes*, *Tic Tac Toe*, *Vote* and *Zoo*. Hider was run for each database by using the original dataset *(D)* and the reduced dataset *(D*)* in order to compare the computational cost in time and space. Thus, the number of examples was reduced for 8 cases and kept for the other 6 datasets. Logically, for those last ones, the cost shown a light increase by using $D*$, although this never exceeded the 5% with respect to the inverted time by using D. Among to the 8 databases where the editing had a favourable effect, the reduction of the dataset was higher than 20% for 6 cases. Table 1 shows this results. The five first columns show the features of each database: name, number of examples, number of attributes, type of attributes (continuous or discrete) and number of class, respectively. The next column is the number of examples after the editing process. Finally, the three last columns give the relative cost concerned to the runtime, the evaluation time and the space used. This values is obtained by dividing, in each case, the cost with $D*$ by the cost with D. The last row shows each previous relative cost on average.

Table 1. Results

Features					Editing	Relative Cost		
Dataset	N	m	Type	#Classes	N^*	Runtime	Evaluation Time	Space
Breast Cancer W.	699	9	C	2	263	0.44	0.44	0.38
Hayes Roth	132	4	C	3	28	0.33	0.23	0.21
Iris	150	4	C	3	70	0.59	0.57	0.47
Led7	3200	7	D	10	336	0.31	0.13	0.11
Vote	435	16	D	2	342	0.82	0.74	0.79
Zoo	101	16	D	7	59	0.49	0.38	0.58
Average						0.49	0.41	0.42

By observing the average results, the size of dataset is reduced to 42%. This caused a decrease in the evaluation time to 41%. The rule-learning methods that EAs use invest approximately 85% of their time in evaluating the individuals (the mean of the executions of a 10-fold cross-validation with 20 UCI Repository databases [2]). Therefore, this proves the importance of the evaluation with regard to the efficiency of the algorithm. We can deduce that the editing method does not add significant computational cost to the algorithm, so that the runtime is very similar to the evaluation time. In short, for the dataset of table 1, we concludes that the our approach speeds up the learning process, by using less than a half of computational resources, on average.

5 Conclusions and Future Works

In this paper we present new editing method that reduces the evaluation cost of individuals in evolutionary algorithms for supervised learning. This method identifies those regions of attribute space that are indivisible during the learning process, and, furthermore, the method extracts the useful information from each of them. The method takes advantage of those examples which share the same region are identical from the point of view of the learning.

After the empirical experiments, we conclude that our proposal produces a reduction of the computational cost associated to the evaluation of individuals during the learning, in time and space. This reduction is proportional to the reduction in the number of the examples that are the result from the editing process. This no affects to the quality of the knowledge model obtained by the learning algorithm. If the method does reduce the number of examples, our method does not cause a significant increment in the computational cost with regard to the evolutionary algorithm.

References

1. J. S. Aguilar–Ruiz, J. C. Riquelme and M. Toro. Evolutionary Learning of Hierarchical Decision Rules. *IEEE Transactions on Systems, Man and Cybernetics – Part B*, **33**(2)(2003), 324–331.

2. J. S. Aguilar. *Discovering Hierarchical Decision Rules with Evolutionary Algorithms in Supervised Learning.* PhD thesis, University de Seville, 2001.
3. J. Bacardit and J. M. Garrell. Evolving multiple discretizations with adaptive intervals for a Pittsburgh Rule-Based Learning Classifier System. *Genetic and Evolutionary Computation Conference - GECCO 2003.* Lecture Notes in Computer Science 2724, pp. 1818–1831, Springer-Verlag, 2003.
4. J. Bacardit y J. M. Garrell. Incremental Learning for Pittsburgh Approach Classifier Systems. *2nd. Spanish Conference on Metaheuristics and Evolutionary Algorithms (MAEB'03)*, pp. 303–311. Gijón, Spain, 2003.
5. C. L. Blake and C. J. Merz. UCI Repository of machine learning databases [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.
6. P. Clark and R. Boswell. Rule induction with cn2: Some recents improvements. In *Machine Learning: Proceedings of the Fifth European Conference (EWSL-91)*, pages 151–163, 1991.
7. K. A. DeJong, W. M. Spears and D. F. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 1(13):161–188, 1993.
8. F. Divina and E. Marchiori, Evolutionary Concept Learning, *Genetic and Evolutionary Computation Conference - GECCO 2002*, W.B. Langdon et al. eds, Morgan Kaufmann, NY, USA, 2002, pp. 343–350.
9. P. Domingos. Rule induction and instance-based learning: A unified approach. In *Proceedings of International Joint Conference on Artificial Intelligence*, 1995.
10. R. Giráldez, J. S. Aguilar-Ruiz, J. C. Riquelme y D. Mateos Discretization Oriented to Decision Rule Generation, *In Proceedings of International Conference on Knowledge-Based Intelligent Information & Engineering Systems*, pp. 275–279, IOS Press, Crema, Italy, 2002.
11. R. Giráldez, J. S. Aguilar-Ruiz and J. C. Riquelme. Natural Coding: A More Efficient Representation for Evolutionary Learning. *Genetic and Evolutionary Computation Conference - GECCO 2003*, pp. 279–290. Chicago, USA, 2003.
12. R. Giráldez and J. S. Aguilar–Ruiz and J. C. Riquelme. Knowledge-based Fast Evaluation for Evolutionary Learning, *IEEE Transactions on Systems, Man & Cybernetics – Part C*, (in press), 2005.
13. C. Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 1(13):169–228, 1993.
14. H. Liu and H. Motoda *Feature Selection for Knowledge Discovery and Data Mining.* Kluwer Academic, 1998.
15. S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 1994.
16. K. Shim. *SIGKDD Explorations.* December 2000. Volume 2, Issue 2.
17. G. Venturini. SIA: a supervised inductive algorithm with genetic search for learning attributes based concepts. In *Proceedings of European Conference on Machine Learning*, pages 281–296, 1993.
18. D. R. Wilson and T. R. Martinez, Reduction Techniques for Instance–Based Learning Algorithms. *Machine Learning*, 38(3):257–286,2000.