Combinando multiples discretizadores para aprendizaje de reglas evolutivo con enfoque de Pittsburgh

Jaume Bacardit, Josep Maria Garrell y Pere Miralles

Resumen— **Una de las posibles maneras de resolver problemas de clasificación con atributos reales es el uso de algorismos de discretización, que nos permiten usar las representaciones del conocimiento clasicas para atributos simbólicos. El problema principal de los discretizadores es que es muy difícil encontrar un único algoritmo que permita resolver de forma satisfactoria un número grande de problemas. Para solucionar el problema de escoger el discretizador ideal para cada problema, en trabajos previos hemos desarrollado una representacion del conocimiento llamada "Adaptive Discretization Intervals rule representation" (ADI). Esta representacion permite usar múltiples discretizadores al mismo tiempo y dejar al proceso evolutivo la tarea de escojer el discretizador/es ideal para cada problema. Hasta el momento se habia trabajado mayoritariamente con discretizaciones uniformes de diversos tamaños. En este artículo se experimenta con un especro amplio de discretizadores, uniformes y no uniformes, combinándolos con diversos criterios. Los resultados de las pruebas realizadas con diversos dominios muestran que se consigue obtener un sistema clasificador robusto y preciso.**

Palabras clave— **aprendizaje evolutivo, repesentaciones para atributos reales, discretizaciones**

I. Introducción

La aplicación de los Algoritmos Genéticos [1], [2] para solucionar problemas de clasificación se suele identificar como "Genetic Based Machine Learning" (GBML), y tradicionalmente se han usado dos enfoques diferentes: El de Pittsburgh y el de Michigan, representativos por los sistemas *GABIL* [3] y XCS [4] respectivamente.

Para tratar con problemas con atributos reales una de las posibles alternativas es el uso de un algoritmo de discretización, ya que nos permite reutilizar las representaciones del conocimiento clasicas de este tipo de sistemas. Un buen algoritmo de discretizacion tiene que buscar el equilibrio entre la pérdida de información inevitable de la discretización y tener un número razonable de puntos de corte. Por otro lado, es muy difícil encontrar un único algoritmo de discretización que dé resultados satisfactorios para un rango grande de problemas.

Para solucionar este tipo de problemas, en trabajos anteriores hemos desarrollado una representación del conocimiento llamada "Adaptive Discretization Intervals Rule Representation" (ADI) [5], [6]. Esta representación, descendiente de la usada en el sistema *GABIL* [3], realiza dos tareas: permite combinar múltiples algoritmos de discretización y, para acotar el espacio de búsqueda, permite fusionar los intervalos propuestos por los discretizadores.

En su versión inicial [5] se combinaban discretizadores uniformes de diversos tamaños, y posteriormente en su segunda revisión (*ADI2*) [6] se realiza una prueba inicial de mezclar esquemas de discretización uniformes y no uniformes. Los resultados obtenidos indicaban que se consigue un sistema clasificador muy robusto y que obtiene resultados bastante satisfactorios en un especro amplio de problemas.

En este artículo se continua la línea de mezclar discretizadores de muy diverso tipo y origen. En concreto se has probada 8 esquemas de discretización que con diversas parametrizaciones han dado lugar a 24 variantes. Primero se realizan pruebas individuales cuyos resultados sirven para proponer 12 posibles combinaciones de esquemas de discretización que se prueban posteriormente.

Este artículo esta estructurado de la siguiente forma: La sección II presenta algunas muestras de trabajo relacionado. A continuacion se relata el marco de trabajo de nuestro sistema clasificador en la sección III. La sección IV describe la representación *ADI* y los esquemas de discretización implementados. Posteriormente, se detallan las pruebas realizadas en la sección V, los resultados de las pruebas en la sección VI y, finalmente, las conclusiones y lineas de futuro en la sección VII.

II. Trabajo relacionado

En el campo de los *GBML* existen muchos enfoques sobre como tratar atributos reales. De una forma muy simplista se pueden clasificar en dos grandes grupos: Los sistemas que usan dis-

Intelligent Systems Research Group, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Psg. Bonanova 8, 08022-Barcelona, {jbacardit,josepmg,is08388}@salleURL.edu

cretización y los que manejan directamente los valores reales.

Existen muchos algoritmos de discretización, en la sección IV se detallaran totos los algoritmos con los que se ha experimentado para este artículo. Para citar un algoritmo aqui, podemos nombrar el de Fayyad e Irani [7], que és uno de los algoritmos de discretización más utilizados en la actualidad.

Como aplicación específica de las discretizaciones en el campo de los *GBML* encontramos la Codificación Natural de Aguilar y Riquelme [8], que es una propuesta cercana a nuestra representación ADI, pero usando un tipo de regles semànticamente diferentes y unos operadores de cruce y mutación especializados. Usan su propio algoritmo de discretización, llamado *USD* [9].

Hay diveras alternativas de representaciones del conocimiento que trabajan directamente con valores reales. Encontramos reglas con intervalos reales (el sistema XCSR [10] de Wilson), o el sistema GALE [11] de Llorà y Garrell, donde coevolucionan varias representaciones del conocimiento como árboles de decisión o conjuntos de instancias usadas como núcleo de un clasificador de tipo vecino más cercano.

III. Marco de Trabajo

En esta sección describimos las características principales de nuestro sistema classificador, que és un descendiente del sistema *GABIL* [3] de De Jong y Spears. Directamente del GABIL hemos utilizado el operador de cruze semanticamente correcto y la función de evaluación (procentaje de acierto elevado al quadrado).

Estrategia de clasificación: Nuestro conjunto de regles és ordenado, és decir, que la primera regla cuyo predicado lógico sea cierto para el ejemplo a clasificar serà la seleccionada, usando la estructura llamada *Lista de decisión* [12].

Mutation operators: Definimos p_{mut} como la probabilidad de mutar un individuo. Cuando un individuo ha sido seleccionado para mutarse, aleatoriamente se escoje uno de sus genes para ser mutado.

Control del tamaño de los individuos: Trabajar con individuos de tamaño variable implica tener que añadir un cierto control sobre este tamaño para impedir un crecimiento descontrolado de los individuos (el efecto *Bloat* [13]) y para realizar presión de generalización necesaria para obtener un buen rendimiento en la fase de test. Este control se realiza mediante la combinación de dos operadores:

- *Eliminación de regles:* Este operador borra las reglas de los individuos que no clasifican ningun ejemplo de entrenamiento. El operador se aplica con un par de restricciones: (a) el proceso se activa después de un cierto número de iteraciones, para evitar una pérdida masiva de diversidad y (b) el operador no se aplica si el número de reglas del individu es inferior a un cierto umbral.
- *Selección jerárquica [14]:* Usamos un algoritmo de selección basado en torneo con una función de comparación a dos niveles (precisión y tamaño) definida en la figura 1.

Fig. 1. Método de comparación de la selección jerárquica

Si |precisión$_a$ - precisión$_b$ | < umbral **Entonces**
  Si tamaño$_a$ < tamaño$_b$ **Entonces** a es mejor que b
  Si tamaño$_a$ > tamaño$_b$ **Entonces** b es mejor que a
  Si tamaño$_a$ = tamaño$_b$ **Entonces** usamos el caso general
Sino Caso general: usamos el individuo con mayor precisión

IV. La representación ADI y los algoritmos de discretización

A. La representación ADI

La estructura semàntica de regla es heredada de *GABIL*: Cada regla consiste en una condición y una clase asociada, entendiendo que predecimos la clase asociada para los ejemplos que hagan cierta la condición. La condición és un predicado lógico en Forma Normal Conjuntiva definido como:

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

Donde A_i es el atributo i del problema y V_i^j és el valor j que puede tomar el atributo i.

En *GABIL* este tipo de predicados se pueden codificar en una cadena binario de la siguiente forma: Supongamos que tenemos un problema con dos atributos, donde cada atributo puedo tomar tre valores diferentes {1,2,3}. Una regla de la forma "Si el primer atributo tiene el valor 1 o 2 y el segundo tiene el valor 3, luego predecimos la clase 1" se puede codificar con la cadena binaria 110|001|1. Para atributos reales, cada bit (excepto el de la clase) seria asociado a un intervalo de discretización.

Los intervalos usados en la representación ADI no son estáticos, sino que evolucionan fusionandose (*merge*) o partiendose (*split*) (teniendo un tamaño mínimo llamado *micro-intervalo*. Es necesario extender la codificacion binaria de *GABIL*, y se hace tal como esta representado en la figura 2. La figura 3 muestra las operaciones de *split* y *merge*.

Los detalles de la representación ADI se definen a continuación:

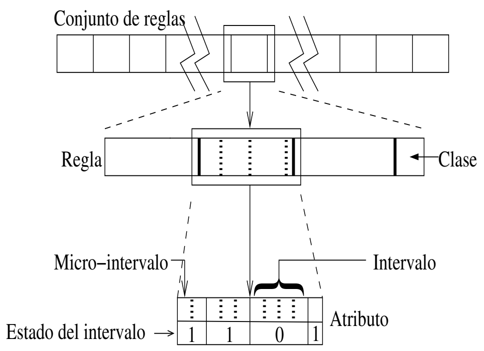

Fig. 2. La representación ADI

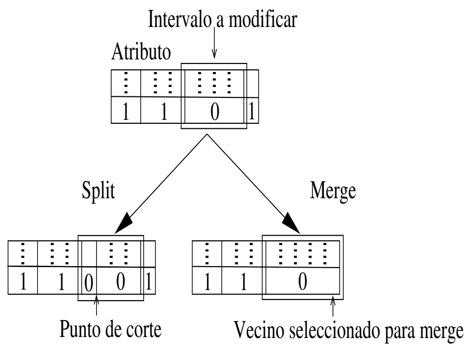

Fig. 3. Los operadores de split y merge

1. Un conjunto de intervalos de discretización estàticos (llamados *micro-intervalos*) es asignado en la fase de inicialización del *GA* a cada atributo de cada regla de cada regla.
2. Los intervalos de las reglas de forman juntando *micro-intervalos* contiguos.
3. En la población coexistiran diversos conjuntos de intervalos de discretización. La evolución escogerà el discretizador correcto para cada atributo.
4. Por razones de coste computacional, definiremos un límite superior sobre el número de intervalos permitido per atributo, que en la mayoria de casos sera inferior al número de *micro-intervalos* asignado a cada atributo.
5. Cuando aplicamos una operación de "split" a un intervalo, seleccionamos aleatoriamente un punto en sus *micro-intervalos* para aplicar la operación.
6. Cuando aplicamos una operación de "merge" a dos intervalos, el estado (0 o 1) del intervalo resultante será el del intervalo que contenga mas *micro-intervalos*. Si tienen igual número, el estado se escoge aleatoriamente.
7. La discretización asignada en la inicialización a cada atributo se escoge de un conjunto predefinido.
8. El número y tamaño (número de *micro-intervalos* adyacentes) se seleccionan al azar.
9. Los puntos de corte del operador de cruze solo podran estar en las fronteras entre atributos, para garantizar la integridad semántica de la reglas.

Para integrar los operadores de "split" y "merge" en el proceso evolutivo, hemos ampliado el ciclo de funcionamento del *GA* con dos fases especiales que se aplican a la población de hijos después del operador de mutación. La figura 4 contiene el pseudocódico del operador de "merge". El operador de "split" funciona de forma equivalente.

B. Algoritmos de discretización integrados en la representación ADI

V. 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.

A. 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* [11]) that has non-orthogonal class boundaries. We also use several problems provided by the University of California at Irvine (UCI) repository [15]. 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* [16]) and on mammograms (*mmg* [17]), whereas the last one is related to the prediction of student qualifications (*lrn* [18]). The characteristics of the problems are listed in table I. The partition of the examples into the train and test sets was done using *stratified ten-fold cross-validation* [19].

B. 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 configura-

```
PorCada Individuo i de la Población
        PorCada Regla j del Individuo i
                PorCada Atributo k de la Regla j del Individuo i
                        Si random [0.,1] < p_merge
                            Seleccionar un intervalo al azar del atributo
                            Aplicar la operación de "merge" al intervalo
                        FinSi
                FinPorCada
        FinPorCada
FinPorCada
```

Fig. 4. Pseudocodigo del operador de "merge"

TABLA I
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

tion **ADI2** the use of the Fayyad & Irani discretization method [7].

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 [10] (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 II and III. 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 III.

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.

VI. 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 IV and a summary of them taking the form of a methods ranking is in table V. 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 [19] with a significance level of 1 % in order to determine if

TABLA II

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

TABLA III

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}	p_{merge} orig	E_{split} orig	E_{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

there were significant outperformances between the methods tested. The results of the t-tests are shown in table VI. None of the *ADI* was outperformed.

The results were also analyzed using the two-sided t-test [19] 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 [20].

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.

VII. CONCLUSIONS AND FURTHER WORK

This paper focused on a revision and generalization of our previous work on represen-

TABLA IV

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

TABLA V

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

TABLA VI

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	

tations for real-valued attributes: the Adaptive Discretization Intervals (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.

Referencias

[1] John H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

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

[3] Kenneth A. DeJong and William M. Spears, "Learning concept classification rules using genetic algorithms," *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 651–656, 1991.

[4] Steward W. Wilson, "Classifier fitness based on accuracy.," *Evolutionary Computation*, vol. 3, no. 2, pp. 149–175, 1995.

[5] Jaume Bacardit and Josep M. Garrell, "Evolution of multi-adaptive discretization intervals for A rule-based genetic learning system," in *Proceedings of the Eighth Iberoamerican Conference on Artificial Intelligence*. 2002, pp. 350–360, LNAI vol. 2527, Springer.

[6] Jaume Bacardit and Josep M. Garrell, "Evolving multiple discretizations with adaptive intervals for a pittsburgh rule-based learning classifier system," in *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO2003*. 2003, (in press), LNCS, Springer.

[7] Usama M. Fayyad and Keki B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *IJCAI*, 1993, pp. 1022–1029.

[8] Jesus S. Aguilar-Ruiz, José C. Riquelme, and Carmelo Del Valle, "Improving the evolutionary coding for machine learning tasks," in *Proceedings of the European Conference on Artificial Intelligence, ECAI'02*, Lyon, France, 2002, pp. pp. 173–177, IOS Press.

[9] R. Giráldez, J. S. Aguilar-Ruiz, and J. C. Riquelme, "Discretización supervisada no paramétrica orientada a la obtención de reglas de decisión," in *Proceedings of the CAEPIA2001*, 2001, pp. 53–62.

[10] Stewart W. Wilson, "Get real! XCS with continuous-valued inputs," in *Festschrift in Honor of John H. Holland*, L. Booker, Stephanie Forrest, M. Mitchell, and Rick L. Riolo, Eds. 1999, pp. 111–121, Center for the Study of Complex Systems.

[11] Xavier Llorà and Josep M. Garrell, "Knowledge-independent data mining with fine-grained parallel evolutionary algorithms," in *Proceedings of the Third Genetic and Evolutionary Computation Conference*. 2001, pp. 461–468, Morgan Kaufmann.

[12] Ronald L. Rivest, "Learning decision lists," *Machine Learning*, vol. 2, no. 3, pp. 229–246, 1987.

[13] Terence Soule and James A. Foster, "Effects of code growth and parsimony pressure on populations in genetic programming," *Evolutionary Computation*, vol. 6, no. 4, pp. 293–309, Winter 1998.

[14] Jaume Bacardit and Josep M. Garrell, "Métodos de generalización para sistemas clasificadores de Pittsburgh," in *Proceedings of the "Primer Congreso Español de Algoritmos Evolutivos y Bioinspirados (AEB'02)"*, 2002, pp. 486–493.

[15] C. Blake, E. Keogh, and C. Merz, "Uci repository of machine learning databases," 1998, (www.ics.uci.edu/mlearn/MLRepository.html).

[16] E. Martínez Marroquín, C. Vos, and et al., "Morphological analysis of mammary biopsy images," in *Proceedings of the IEEE International Conference on Image Processing*, 1996, pp. 943–947.

[17] 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.

[18] 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*, vol. 22, pp. 132–141, 2000.

[19] Ian H. Witten and Eibe Frank, *Data Mining: practical machine learning tools and techniques with java implementations*, Morgan Kaufmann, 2000.

[20] José C. Riquelme and Jesús S. Aguilar, "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, pp. 161–167.