Aprendizaje Evolutivo de Reglas: Mejoras en la Codificación y Evaluación de Individuos

Raúl Giráldez, José C. Riquelme y Jesús S. Aguilar–Ruiz

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Sevilla
Avda. Reina Mercedes s/n, Sevilla, 41012

Resumen Enmarcado dentro del área del aprendizaje supervisado, este trabajo describe brevemente diversos métodos algorítmicos dirigidos hacia la mejora de este tipo de técnicas para la generación de reglas de decisión [10]. El principal objetivo es reducir el coste computacional asociado a los aspectos críticos de los algoritmos de aprendizaje evolutivo, así como aumentar la calidad de los resultados mediante una búsqueda más eficiente y eficaz de las soluciones. Para ello, se desarrolla una nueva codificación de los individuos de la población (*Codificación Natural*), así como una estructura de datos (*EES*) que organiza la información del conjunto de datos para acelerar la evaluación de los individuos durante el proceso evolutivo. Estas propuestas son integradas en la herramienta de generación de reglas de decisión HIDER, cuyo rendimiento ha sido probado experimentalmente, ofreciendo resultados muy satisfactorios.

Palabras Clave: Aprendizaje Supervisado, Algoritmos Evolutivos, Codificación, Evaluación, Reglas de Decisión.

1. Introducción

En el área del Aprendizaje Automático, cuando el usuario conoce la clase de los ejemplos del conjunto de datos y desea obtener un modelo de conocimiento capaz de predecir la clase de nuevas instancias a partir de los valores de los atributos de éstas, nos encontramos en el campo del Aprendizaje Supervisado. Dicho modelo de conocimiento es usualmente representado mediante reglas de decisión del tipo "*Si* $\mathcal{C}$ *Entonces* $\mathcal{E}$", donde $\mathcal{C}$ es una conjunción de condiciones que establecen qué valores deben tomar los atributos para clasificar un ejemplo con la etiqueta de clase $\mathcal{E}$. Por otra parte, los Algoritmos Evolutivos (AE) conforman una de las más importantes familias de modelos computacionales con aplicación en el campo del aprendizaje automático, cuya validez y efectividad han sido ampliamente estudiadas en la bibliografía [4,5,12,18]. Entre las numerosas propuestas, la herramienta evolutiva de generación de reglas COGITO [1], se presenta como principal punto de partida de este trabajo.

El objetivo de nuestra investigación es mejorar las técnicas de aprendizaje evolutivo de reglas de decisión, abordando estas mejoras desde los dos puntos de vista fundamentales de un algoritmo: la *eficiencia*, reduciendo el coste computacional en tiempo y espacio de las tareas críticas del algoritmo y acelerando la convergencia en la búsqueda de soluciones; y la *eficacia*, aumentando la calidad de los resultados mediante una representación más adecuada del modelo y una búsqueda más efectiva de las soluciones.

2. Motivación

Inspirados en el concepto Darwiniano de evolución, los AE emplean un método de búsqueda aleatoria para encontrar soluciones a un problema particular [11]. Partiendo de un conjunto de soluciones iniciales, genera nuevas soluciones mediante la selección de las mejores y recombinación de éstas, simulando así la evolución de una población de individuos. En general, un AE consta de los siguientes componentes básicos [13]:

1. Representación de las soluciones potenciales al problema (*codificación*), transformando éstas en *individuos genéticos*.
2. Método para generar las soluciones iniciales (*población inicial*).
3. Función de *evaluación* que juega el papel del ambiente, calificando las soluciones en términos de aptitud o bondad.
4. Método de *selección*, que emula la selección natural según la aptitud de los individuos.
5. Operadores genéticos (*cruce* y *mutación*), que simulan la reproducción de los individuos, alterando la composición de los descendientes para desencadenar la evolución.

Aunque todos estos aspectos tienen una influencia notable en la eficiencia y la eficacia del algoritmo, las dos tareas principales en la aplicación de un AE son el diseño de la codificación y la evaluación de los individuos. Por ello, inicialmente enfocamos nuestros esfuerzos a la mejora de estos dos aspectos junto al de los operadores genéticos, estrechamente relacionados con la codificación. No obstante, durante el transcurso de la investigación, fueron necesarias diversas adaptaciones sobre el resto de factores.

La codificación influye directamente sobre el tamaño del espacio de búsqueda, el cual condiciona a su vez la convergencia del algoritmo, así como la probabilidad de encontrar buenas soluciones. En concreto, la codificación real aplicada habitualmente hace que el espacio de búsqueda sea teóricamente infinito para dominios continuos. Esto nos motivó a desarrollar la denominada *Codificación Natural*, con el objeto de minimizar, o al menos reducir, el número de soluciones potenciales, sin que se produjera pérdida en la precisión en las mismas. Conjuntamente, fueron diseñados los operadores genéticos específicos que contribuyeran a la aceleración de la búsqueda.

Respecto a la evaluación de los individuos hay que tener en cuenta dos aspectos importantes. En primer lugar, la función de evaluación es el elemento más influyente en la calidad del modelo final, puesto que precisamente mide la bondad de las soluciones respecto al conjunto de datos de entrada durante el proceso evolutivo. En segundo lugar, el método evaluación afecta sustancialmente al tiempo de ejecución del algoritmo, ya que la evaluación de un solo individuo lleva consigo habitualmente un recorrido de los datos de entrenamiento. Este problema de eficiencia nos impulsó a plantear nuevos métodos de evaluación para mitigar el coste computacional asociado a esta fase del AE, los cuales confluyeron finalmente en el desarrollo la *Estructura de Evaluación Eficiente* (EES, *Efficient Evaluation Structure*).

Los resultados de esta investigación se integran en la herramienta evolutiva de aprendizaje de reglas de decisión HIDER (*Hierarchical Decision Rules*), cuyas principales características son detalladas en las siguientes secciones.

3. HIDER

HIDER usa un AE para buscar las mejores soluciones y producir un conjunto de reglas jerárquicas, de forma que cada regla influye sobre las siguientes en el orden jerárquico. Siguiendo este orden, un ejemplo será clasificado por la i-ésima regla si éste no cumple las condiciones establecidas por las $i-1$ reglas anteriores. Así, una regla es tomada en cuenta sólo cuando las anteriores no han sido satisfechas. Durante el proceso de obtención de las reglas, HIDER genera reglas secuencialmente hasta que todos los ejemplos del fichero de entrenamiento han sido cubiertos.

Las reglas obtenidas por HIDER están influenciadas por dos factores críticos: la codificación y la evaluación de los individuos de la población genética. La codificación es la representación interna del espacio de búsqueda, es decir, la representación que el AE utiliza para los individuos genéticos. Por otra parte, para decidir qué individuos son los mejores dentro de la población, éstos deben ser evaluados, asignándole a cada uno de ellos un valor que representa su bondad frente en la población. Dicho valor de bondad es calculado aplicando la función de evaluación, la cual devuelve un valor numérico a partir de, entre otros factores, los aciertos y los errores que un individuo tenga, es decir, el número de ejemplos que la regla clasifique correcta e incorrectamente. La elección de una representación apropiada de los individuos de la población puede reducir considerablemente el espacio de búsqueda. Del mismo modo, una función de evaluación apropiada mejora la precisión de las soluciones obtenidas. La evaluación de los individuos de la población es un proceso altamente costoso, debido a la necesidad de recorrer el conjunto de datos para contar los aciertos y errores de cada individuo. Sin embargo, una adecuada organización de la información puede reducir significativamente este coste. Todos estos aspectos son detallados en las subsecciones siguientes.

3.1. Codificación Natural

Antes de ejecutar un AE, es necesario llevar a cabo un análisis del problema para seleccionar la codificación más adecuada. La herramienta COGITO implementa la denominada codificación híbrida [1], la cual usa la codificación binaria y real para representar los atributos discretos y continuos respectivamente. Sin embargo, esta codificación presenta dos importantes inconvenientes: por un lado, la componente binaria hace que el tamaño de los individuos dependa del numero de valores discretos presentes en el conjunto de datos; y por otro lado, la codificación real define un alfabeto de símbolos teóricamente infinito, lo cual hace que el espacio de búsqueda también lo sea.

En vista de los problemas de la codificación híbrida y con el propósito de mejorar la codificación de COGITO, nos planteamos un nueva codificación más compacta y en la que no hubiera que realizar constantes conversiones para la aplicación de los operadores genéticos. Con este propósito desarrollamos la denominada Codificación Natural [2,10], donde cada gen representa los valores de un atributo, ya sea continuo o discreto. Sin embargo, al contrario que otras codificaciones, en la natural cada gen es un único número natural. Aunque los atributos continuos y discretos usen la misma codificación, el modo de codificar los valores de ambos tipos de atributos es diferente. La figura 1 muestra un ejemplo de la codificación natural de una regla de decisión con un atributo continuo y otro discreto.

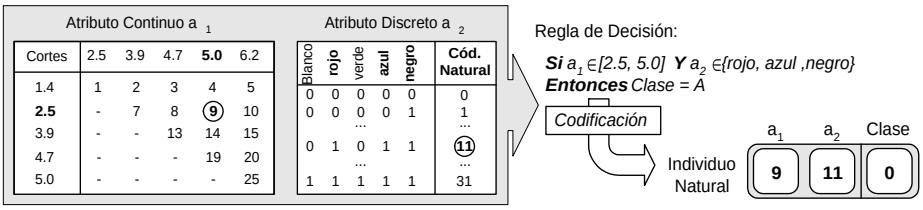

Figura 1. Codificación Natural.

Para los atributos continuos, se aplica previamente un método de discretización para disminuir la cardinalidad del conjunto de valores, convirtiendo su dominio infinito en un conjunto finito de intervalos disjuntos. En realidad, no nos interesan los intervalos propiamente dichos, sino los límites de éstos, es decir, los puntos que separan un intervalo de otro. A estos puntos de separación se les suele llamar cortes. Aunque se puede aplicar cualquier método de discretización supervisada, nosotros usamos USD (Unparametrized Supervised Discretization)[7], el cual maximiza la bondad global de los intervalos que obtiene. Este método de discretización devuelve un conjunto de cortes los cuales son los posibles límites de los intervalos. La codificación natural asigna un número natural a cada posible combinación de límites. La figura 1 muestra un ejemplo de codificación natural para un atributo continuo (a_1) cuyos cortes calculados son {1.4, 2.5, 3.9, 4.7, 5.0, 6.2}. Si k es el número de cortes, las filas de la tabla están etiquetadas con los $(k-1)$ primeros cortes (desde el primero hasta el $(k-1)$-ésimo) y las columnas con los $(k-1)$ últimos (desde el segundo al k-ésimo). Cada elemento de la tabla es un número natural que identifica al intervalo $[i, j]$. Los elementos se numeran empezando desde el 1, de izquierda a derecha y desde arriba a bajo. Aquellos números que pertenezcan a intervalos no válidos, es decir todos los $[i, j]$ donde $i \geq j$, no son considerados y se representan por "$-$".

Aunque los atributos discretos son codificados también con números naturales, el significado de estos números es diferente al de la codificación para atributos continuos. En este caso, el número natural identifica un conjunto de valores discretos y pertenece al intervalo $[0, 2^{|A|} - 1]$, donde $|A|$ es el cardinal del conjunto de valores posibles del atributo. Cada valor representa un bit en un código binario de longitud $|A|$, donde la ausencia de un valor es representada por 0 y la inclusión por 1. El código que identifica un determinado conjunto de valores es la conversión de este código binario a número natural. Así, la figura 1 muestra también un ejemplo de codificación natural para un atributo discreto (a_2) que puede tomar cinco valores distintos: *blanco*, *rojo*, *verde*, *azul* y *negro*. La codificación de la clase es una simple enumeración de las etiquetas.

Es fácil notar que la codificación natural reduce el tamaño de los individuos respecto a la híbrida, ya que esta última hubiera precisado de ocho genes para codificar la regla del ejemplo (dos para a_1, cinco para a_2 y uno para la clase) mientras que la codificación natural sólo necesita tres (uno para cada atributo y uno para la clase).

Respecto a los operadores genéticos, tanto la mutación como el cruce son expresiones algebraicas simples, de coste computacional despreciable, que transforman los genes naturales sin necesidad de decodificar tales valores. En [10], se da la descripción

detallada de estos operadores. Por ejemplo, la mutación natural de k-ésimo bit de un gen n que representa un atributo discreto se calcularía aplicando la siguiente expresión.

$$mut_k(n) = \left(n + 2^{k-1}\right) \%2^k + 2^k \left\lfloor \frac{n}{2^k} \right\rfloor \quad (1)$$

donde % es el operador módulo que obtiene el resto de la división entera y $\lfloor\_\rfloor$ es el operador la parte entera por defecto.

3.2. Función de Evaluación

La función de evaluación se encarga de medir la calidad de los individuos respecto de la clasificación que éstos realizan. En otras palabras, asigna un valor numérico, denominado bondad, a un individuo dependiendo de los aciertos y los errores que éste cometa al clasificar los ejemplos del fichero de entrenamiento. Un individuo comete un error cuando cubre a un ejemplo pero no lo clasifica con la misma clase que éste tiene en el fichero de entrenamiento. En caso contrario, si la clase coincide, entonces decimos que el individuo tiene un acierto. Debido al ruido que la mayoría de bases de datos incorpora, encontrar una función de evaluación apropiada no es una tarea trivial.

El AE maximiza la función de evaluación f de forma que un individuo r es mejor cuanto mayor sea $f(r)$. La función de evaluación que HIDER aplica viene dada por la ecuación 2.

$$f(r) = N - EC(r) + A(r) + cobertura(r) \quad (2)$$

donde N es el número de ejemplos en proceso, es decir, el número de ejemplos que quedan por cubrir en el fichero de entrenamiento; $EC(r)$ es el error de clase, es decir, el número de ejemplos que pertenecen a la región definida por la regla pero que no comparten la misma clase que dicha regla; $A(r)$ es el número de ejemplos correctamente clasificados; y la $cobertura(r)$ es el volumen normalizado de la región cubierta por la regla.

3.3. Estructura de Evaluación Eficiente

Como se ha mencionado anteriormente, la función de evaluación cuantifica la bondad de cada individuo de la población a partir del número de aciertos y errores que éstos cometen al clasificar los ejemplos del conjunto de datos. Para contabilizar esos aciertos y errores, el método usado tradicionalmente por los algoritmos de aprendizaje evolutivo, incluido COGITO, consiste en realizar un recorrido lineal de los datos de entrenamiento para cada individuo de la población. Este proceso de evaluación, aunque sencillo de aplicar, resulta altamente costoso. Algunos autores [15,16] han concentrado su esfuerzo en mejorar el proceso de aprendizaje. Otros han abordado el problema desde la perspectiva de la escalabilidad [17]. Sin embargo, la organización apropiada de la información podría contribuir también a la reducción del coste computacional. Este aspecto, no menos importante que los anteriores, ha sido quizá más descuidado.

Dado el problema del alto coste de evaluación, y teniendo en cuenta que las estructuras existentes [6] no son suficientemente adecuadas, desarrollamos la *Estructura*

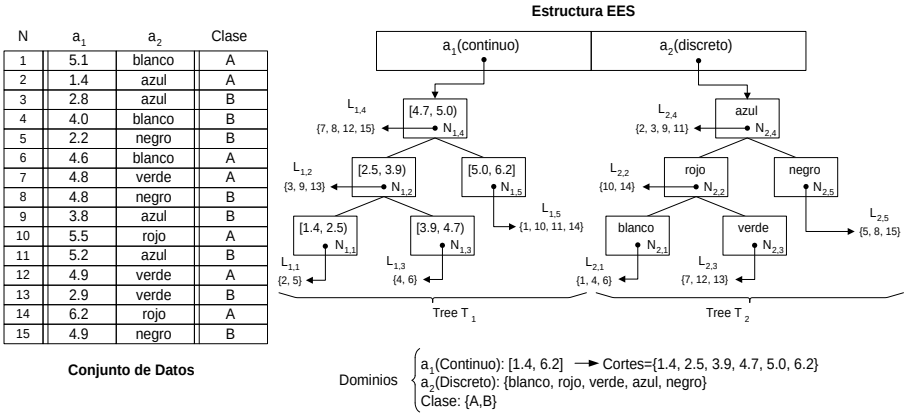

Figura 2. Estructura de Evaluación Eficiente.

de Evaluación Eficiente [8,9], en adelante EES (*Efficient Evaluation Structure*), la cual acelera del proceso de evaluación de reglas de decisión durante la aplicación del algoritmo evolutivo. Esta estructura indexa el conjunto de datos de forma que se aprovecha la semántica de las reglas de decisión para discriminar aquellos ejemplos que no son cubiertos por las mismas. De este modo, sólo son procesados los ejemplos estrictamente necesarios.

En general, para cada atributo a_i denotaremos por Ω_i al conjunto finito de valores que a_i puede tomar. En el caso de que a_i sea un atributo discreto, Ω_i contendrá valores que representaremos como V_{ij} (con $1 \leq j \leq |\Omega_i|$). Por el contrario, si se trata de un atributo continuo, Ω_i contendrá intervalos que llamaremos I_{ij} (con $1 \leq j \leq |\Omega_i|$), cuyos límites inferior y superior denotaremos por li_{ij} y ls_{ij} respectivamente. De este modo, EES almacenará la información de los atributos continuos de forma similar a como lo hace para los discretos: se guardarán los límites inferior y superior de cada intervalo, en lugar de los valores individuales que son almacenados para los atributos discretos. La figura 2 muestra un ejemplo de la estructura de datos para un conjunto con dos atributos.

Básicamente, EES es un vector de árboles de búsqueda binarios y balanceados, de forma que el i–ésimo elemento del vector contendrá información sobre el i–ésimo atributo (a_i) del conjunto de datos. En concreto, se almacenan los diferentes valores o intervalos que a_i puede tomar en el árbol, que denotaremos por T_i. Por simplicidad, la Figura 2 presenta únicamente dos árboles para los dos atributos del conjunto de datos. Si el atributo a_i es continuo, se almacenarán los límites del intervalo I_{ij} correspondiente (li_{ij} y ls_{ij}). En caso de que sea discreto, se almacenará el valor V_{ij}. Además de V_{ij} o I_{ij}, cada nodo N_{ij} del árbol T_i contiene una lista (L_{ij}) de índices que indican las posiciones de los ejemplos dentro del conjunto de datos. Si a_i es discreto, los índices contenidos en la lista L_{ij} corresponderán a los ejemplos cuyo i–ésimo atributo toma el j–ésimo valor (V_{ij}) dentro del conjunto Ω_i de posibles valores de dicho atributo. Si por el contrario a_i es continuo, los índices contenidos en L_{ij} corresponderán a aquellos

ejemplos cuyo valor para el i–ésimo atributo está incluido en el j–ésimo intervalo (I_{ij}) dentro del conjunto Ω_i de posibles intervalos de dicho atributo. Es importante señalar que los árboles se encuentran ordenados, de forma que cualquier búsqueda sobre ellos tiene coste logarítmico.

El principal objetivo que buscamos con la utilización de la estructura de datos es no tener que procesar aquellos ejemplos cuyos valores no son cubiertos por la regla que se está evaluando. Cada nodo N_{ij} del árbol contiene en la lista L_{ij} los índices de aquellos ejemplos que satisfacen la condición $a_i \in I_{ij}$ o $a_i \in V_{ij}$, según el caso. Si tomamos un nodo N_{ij} de cada árbol T_i, la intersección de las listas L_{ij} será el conjunto de los índices de ejemplos que satisfacen que cada uno de sus atributos a_i toman valores que son cubiertos por cada nodo N_{ij} correspondiente. La ventaja que ofrece la estructura radica en que las intersecciones se realizan de forma incremental, es decir, primero se realiza la intersección de la lista para el atributo a_1 y la lista para el atributo a_2. Si dicha intersección no es vacía, se busca la lista para el atributo a_3 y se realiza una nueva intersección entre esta lista y el resultado de la intersección anterior. Este proceso se repite hasta completar todos los atributos o bien hasta que una de las intersecciones resulte vacía. Si el proceso se completa, la lista resultante contendrá los índices de los ejemplos que cumplen las condiciones correspondientes a los nodos N_{ij} elegidos. Si estos nodos N_{ij} son buscados según las condiciones establecidas por una regla de decisión, estaremos evaluando dicha regla, siendo la lista final el conjunto de índices de ejemplos cubiertos.

Es fácil colegir que, usando la estructura EES, sólo son tratados los ejemplos que van siendo cubiertos por las condiciones de regla en evaluación, puesto que en cada iteración, el numero de índices contenidos en las listas se va decrementando o, a lo sumo, se queda igual. Si en alguna iteración intermedia, la intersección resulta vacía, el proceso se detiene, puesto que esa regla ya no podrá cubrir a ningún ejemplo. Si esa misma regla hubiera sido evaluada mediante el recorrido lineal del conjunto de datos, éste hubiera sido tratado completamente aunque la regla no cubriera ningún ejemplo.

Para probar la rendimiento de la EES frente al recorrido lineal, se llevaron a cabo pruebas específicas usando algunas bases de datos disponibles en el UCI *ML Repository* [3]. Como resultado general se obtuvo que EES obtiene una reducción del tiempo de evaluación de más del 52 % en promedio.

4. Experimentos

Se han realizado dos estudios comparativos para probar la calidad de HIDER usando algunas bases de datos de UCI, en concreto: *Breast Cancer (BC), Bupa Liver Dis (BU), Cleveland (CL), German Credit (GC), Glass (GL), Heart Disease (HD), Hepatitis (HE), Horse Colic (HC), Iris (IR), Lenses (LE), Mushrooms (MU), Pima Indian (PI), Vehicle (VE), Vote (VO), Wine (WI) y Zoo (ZO)*. El primer estudio compara la herramienta con C4.5 Release 8 y C4.5Rules [14], en términos de precisión (tasa de error) y complejidad (número de reglas). El segundo estudio analiza la codificación natural frente a la codificación híbrida, cotejando los resultados de HIDER con los obtenidos por la herramienta COGITO. Es importante señalar que tanto la estimación del error como el número de reglas generadas por cada método han sido obtenidos mediante validación cruzada es-

Tabla 1. Parámetros de COGITO (Codificación Híbrida) y HIDER (Codificación Natural).

Parámetro	COGITO	HIDER
Tamaño de la Población	100	70
Número de Generaciones	300	100
Porcentaje de Réplicas	20 %	20 %
Porcentaje de Cruces	80 %	80 %
Probabilidad de Mutación Individual	0.5	0.5
Probabilidad de Mutación por Gen	$\frac{1}{\|atributos\|}$	$\frac{1}{\|atributos\|}$
Probabilidad de Mutación de los Valores Discretos	$\frac{1}{\|valores\|}$	$\frac{1}{\|valores\|}$

tratificada de 10 conjuntos (*stratified 10-fold cross validation*). Todos los experimentos fueron realizados usando los mismos conjuntos de entrenamiento y test para todos los algoritmos de modo que los resultados obtenidos fueran comparables.

Un aspecto notable se refiere a la parametrización del algoritmo evolutivo aplicado por HIDER. Puesto que tanto C4.5 como COGITO no fueron configurados exclusivamente para cada base de datos, y dado que pretendíamos que los experimentos fueran lo más justos posibles, mantuvimos todos los parámetros de HIDER constantes en todas las pruebas. Los valores concretos usados para parametrizar tanto HIDER como COGITO fueron los expresados en la tabla 1.

4.1. HIDER vs. C4.5/C4.5Rules

Con el objetivo de dar una visión global del rendimiento de nuestra herramienta, comparamos HIDER con uno de los sistemas de referencia en aprendizaje supervisado, C4.5, el cual es habitualmente utilizado en la bibliografía para este tipo de comparativas. Dado que HIDER genera reglas de decisión, hemos incluido en este este estudio comparativo la versión denominada C4.5Rules, la cual construye un conjunto de reglas de decisión a partir del árbol de decisión obtenido por C4.5 aplicando criterios de refinado y poda a posteriori.

Los resultados obtenidos por HIDER son comparados con los generados por C4.5 y C4.5Rules en la tabla 2. Para dar validez a los resultados obtenidos, se llevó a cabo un test estadístico con una confianza del 95 % con el fin de determinar qué resultados son significativos y cuáles no. En concreto, se aplicó el Test de Student de Diferencia de Medias con $\alpha = 0{,}05$. La lectura de la tabla es la siguiente: la primera columna indica la base de datos a la que se le ha aplicado cada método; el siguiente bloque de dos columnas da los resultados de HIDER, es decir, la tasa media de error junto a la desviación estándar (ER$\pm\sigma$) y el número medio de reglas junto a la desviación estándar (NR$\pm\sigma$). De igual modo, el siguiente bloque de dos columnas presenta los resultados obtenidos por C4.5, con el mismo significado que las dos columnas anteriores, es decir, la tasa media de error y el número medio de reglas así como ambas desviaciones estándar; el siguiente bloque de dos columnas muestra los resultados del test estadístico. La interpretación de los símbolos de estas dos columnas es la siguiente: un "$-$" indica que HIDER obtiene peor resultado que C4.5; un "$+$" denota que HIDER mejora a C4.5; y por último, un "$\bullet$" significa que el resultado es significativo, ya sea positivo o

Tabla 2. HIDER vs. C4.5/C4.5Rules.

	HIDER		C4.5		HIDER vs C4.5		C4.5R		HIDER vs C4.5R	
BD	ER$\pm\ \sigma$	NR$\pm\ \sigma$	ER$\pm\ \sigma$	NR$\pm\ \sigma$	ER	NR	ER$\pm\ \sigma$	NR$\pm\ \sigma$	ER	NR
BC	4.1$\pm$ 2.0	2.0$\pm$ 0.0	6.3$\pm$ 3.0	22.9$\pm$ 3.0	+	+●	4.9$\pm$ 2.4	9.6$\pm$ 1.1	+	+●
BU	37.3$\pm$11.4	4.2$\pm$ 0.8	33.7$\pm$ 9.3	29.7$\pm$ 5.1	−	+●	32.0$\pm$ 6.2	11.9$\pm$ 2.2	−	+●
CL	25.3$\pm$10.5	5.9$\pm$ 0.9	23.5$\pm$ 7.0	38.3$\pm$ 5.8	−	+●	25.9$\pm$14.7	12.2$\pm$ 4.6	+	+●
GE	27.4$\pm$ 3.9	8.0$\pm$ 1.4	32.9$\pm$ 4.3	204.2$\pm$ 8.5	+●	+●	28.8$\pm$ 3.1	6.2$\pm$ 2.2	+	−
GL	35.2$\pm$ 7.8	11.7$\pm$ 1.6	30.8$\pm$11.4	25.8$\pm$ 2.0	−	+●	18.5$\pm$ 5.9	15.0$\pm$ 2.8	−●	+●
HD	21.9$\pm$ 8.8	4.3$\pm$ 0.5	25.5$\pm$ 5.1	33.5$\pm$ 4.5	+	+●	20.7$\pm$ 7.0	11.5$\pm$ 2.0	−	+●
HE	16.7$\pm$11.0	3.7$\pm$ 0.7	19.4$\pm$ 7.0	15.5$\pm$ 1.7	+	+●	16.9$\pm$ 6.1	6.3$\pm$ 3.6	+	+●
HC	20.0$\pm$ 7.7	11.1$\pm$ 1.9	19.0$\pm$ 7.7	44.4$\pm$ 3.8	−	+●	17.5$\pm$ 8.2	5.0$\pm$ 1.9	−	−●
IR	3.3$\pm$ 4.7	3.2$\pm$ 0.4	5.3$\pm$ 6.9	5.7$\pm$ 0.5	+	+●	4.7$\pm$ 7.1	5.0$\pm$ 0.0	+	+●
LE	25.0$\pm$26.4	4.5$\pm$ 0.9	30.0$\pm$21.9	5.2$\pm$ 0.9	+	+	16.7$\pm$22.2	4.1$\pm$ 0.3	−	−
MU	1.2$\pm$ 0.6	3.5$\pm$ 0.5	0.0$\pm$ 0.0	17.6$\pm$ 1.0	−●	+●	0.7$\pm$ 2.3	17.9$\pm$ 1.9	−	+●
PI	25.7$\pm$ 3.4	5.1$\pm$ 0.7	26.1$\pm$ 5.4	24.4$\pm$ 8.1	+	+●	29.7$\pm$ 3.8	8.3$\pm$ 3.1	+	+●
VE	33.8$\pm$ 7.4	19.7$\pm$ 3.3	27.5$\pm$ 3.6	74.8$\pm$10.0	−●	+●	57.6$\pm$14.7	4.1$\pm$ 4.1	+●	−●
VO	4.4$\pm$ 2.9	2.2$\pm$ 0.4	6.2$\pm$ 3.1	15.9$\pm$ 3.0	+	+●	5.3$\pm$ 3.7	7.5$\pm$ 0.7	+	+●
WI	8.8$\pm$ 4.2	5.6$\pm$ 0.8	6.7$\pm$ 7.8	6.5$\pm$ 0.9	−	+●	6.7$\pm$ 7.8	5.6$\pm$ 0.7	−	−
ZO	4.0$\pm$ 5.2	7.9$\pm$ 0.9	7.0$\pm$10.6	10.9$\pm$ 1.8	+	+●	29.8$\pm$20.7	6.3$\pm$ 2.0	+●	−●
ME.	18.4$\pm$ 7.4	6.4$\pm$ 1.0	18.7$\pm$ 7.1	36.9$\pm$ 3.8			19.7$\pm$ 8.5	8.5$\pm$ 2.1		

negativo. Los resultados obtenidos por C4.5Rules (C4.5R en la tabla) son mostrados en las cuatro últimas columnas, teniendo estas una interpretación similar a las anteriores. Finalmente, la última fila da el promedio de cada columna.

Como se puede observar, HIDER mejoró a C4.5 respecto al número de reglas en todas las bases de datos, de las cuales, 15 de estas mejoras resultaron significativas y sólo una no lo fue. Pudiera parecer que este resultado implicaría una pérdida notable de precisión en las reglas, sin embargo, observamos que para 9 de las bases de datos, HIDER presentó mejoras en la tasa de error, aunque sólo una de éstas tuvo significatividad estadística. De las 7 bases de datos en las que nuestra propuesta presentó pérdidas en la tasa de error, en 2 casos dicha pérdida fue significativa.

Respecto a la comparación entre HIDER y C4.5Rules, la tabla presenta también resultados claramente favorables a HIDER. En lo que concierne al número de reglas, HIDER mejoró en 10 casos, teniendo todos éstos casos significación estadística respecto a las reglas generadas por C4.5Rules. De las 6 bases de datos donde C4.5Rules obtuvo menos reglas, sólo 3 casos resultaron significativos. Con respecto a los errores, C4.5Rules y HIDER presentan resultados muy similares, teniendo 9 casos a favor de HIDER, 2 de los cuales fueron significativos, y 7 favorables a C4.5Rules, siendo sólo uno estadísticamente significativo. Nótese que esta comparativa no es totalmente justa con nuestra propuesta, ya que C4.5Rules realiza una poda de las reglas a posteriori, es decir, una vez que ha generado las reglas, elimina aquellas que no aportan beneficio al conjunto total de reglas, quedándose así sólo con las mejores y manteniendo la tasa de error.

Tabla 3. HIDER vs. COGITO.

	COGITO (híbrida)		HIDER (natural)		Mejora	
Database	ER	NR	ER	NR	ϵ_{er}	ϵ_{nr}
BD	4.3	2.6	4.1	2.0	1.05	1.30
BU	35.7	11.3	37.3	4.2	0.96	2.69
CL	20.5	7.9	25.3	5.9	0.81	1.34
GC	29.1	13.3	27.4	8.0	1.06	1.66
GL	29.4	19.0	35.2	11.7	0.84	1.62
HD	22.3	9.2	21.9	4.3	1.02	2.14
HE	19.4	4.5	16.7	3.7	1.16	1.22
HC	17.6	6.0	20.0	11.1	0.88	0.54
IR	3.3	4.8	3.3	3.2	1.00	1.50
LE	25.0	6.5	25.0	4.5	1.00	1.44
MU	0.8	3.1	1.2	3.5	0.67	0.89
PI	25.9	16.6	25.7	5.1	1.01	3.25
VE	30.6	36.2	33.8	19.7	0.91	1.84
VO	6.4	4.0	4.4	2.2	1.45	1.82
WI	3.9	3.3	8.8	5.6	0.44	0.59
ZO	8.0	7.2	4.0	7.9	2.00	0.91
ME.	17.6	9.7	18.4	6.4	1.02	1.55

4.2. HIDER vs. Cogito

El objetivo principal de estas pruebas es mostrar la mejora en eficiencia que el uso de la codificación natural de HIDER aporta frente a la codificación híbrida que COGITO aplica. Los parámetros para ambas herramientas fueron ajustados con los mismos valores (véase la tabla 1) a excepción del número de generaciones y el tamaño de la población. El uso de la codificación natural reduce el espacio de búsqueda y acelera la convergencia del algoritmo. Por ello, HIDER sólo precisó 100 generaciones y 70 individuos para obtener mejores resultados que COGITO, el cual necesitó 300 generaciones y 100 individuos.

Al igual que en la sección anterior, el rendimiento de ambos métodos fue determinado midiendo la tasa media de error (ER) y el número medio de reglas (NR) para cada base de datos. Estas pruebas arrojaron los resultados que aparecen en la tabla 3, la cual, además de la tasa de error y el número de reglas que cada herramienta produjo, muestra la mejora obtenida por HIDER (ϵ_{er} y ϵ_{nr}). Tal mejora es calculada como el cociente de los valores obtenidos por COGITO y HIDER para cada medida. Finalmente, los valores de la última fila representan la media aritmética de cada columna.

Aunque el propósito de estos experimentos era mostrar la mejora en eficiencia de HIDER, podemos ver que éste también supera a COGITO en eficacia. Pese a que la tasa de error sólo se logró disminuir en la mitad de las bases de datos, el número de reglas sí fue reducido en 12 de los 16 casos. Así, aunque la mejora media respecto al error fue muy reducida (2 %), la mejora en el número de reglas alcanza el 55 % en promedio, aumentando así la calidad del modelo de conocimiento. Es cierto que la mejora en la precisión y la complejidad no es tan sensible como en las comparativas con C4.5 y

C4.5Rules, pero no debemos olvidar que HIDER necesitó sólo la tercera parte de las generaciones y tres cuartas partes del tamaño de la población requeridos por COGITO.

Respecto a la reducción en el tiempo de ejecución, HIDER invirtió 1 hora y 20 minutos en completar todas la pruebas, aproximadamente la cuarta parte del tiempo empleado por COGITO que fue de unas 5 horas y media. La razón de este descenso radica en la reducción del espacio de búsqueda que la codificación natural provoca, principalmente en los dominios continuos donde la previa aplicación del algoritmo de discretización USD decrementa el número de posibles soluciones. Dado que esta discretización maximiza la bondad de los intervalos que genera, su aplicación no ocasiona pérdidas de precisión en las reglas, como comprobamos en la tabla 3. La reducción del espacio de búsqueda junto a las características del propio algoritmo hace que éste converja más rápidamente, lo cual posibilita la obtención del modelo en pocas generaciones y con poblaciones relativamente pequeñas. Sin embargo, la rápida convergencia del algoritmo no es el único factor responsable del decremento del tiempo de ejecución, ya que no hay que olvidar que la estructura EES también aporta beneficios en este sentido.

5. Conclusiones

Durante el transcurso de esta investigación, se han desarrollado diferentes propuestas, abordando principalmente los dos aspectos de codificación y evaluación comentados, con el objetivo de mejorar la eficiencia y la eficacia en el aprendizaje evolutivo de reglas de decisión. HIDER es el fruto de la integración de esas propuestas en una única herramienta, siendo la codificación natural y la estructura EES sus principales aportaciones.

La codificación natural representa el dominio de los atributos (discretos y continuos) mediante conjuntos finitos de números naturales. Así, se reduce el tamaño del espacio de búsqueda respecto a la codificación híbrida, lo que hace posible que la búsqueda de soluciones se lleve a cabo de forma más eficaz y eficiente. Por una parte, el uso de la codificación natural, unida a los operadores genéticos diseñados para la misma, produce mejoras en la calidad del modelo, aumentando su precisión y disminuyendo el número de reglas generadas.

La estructura de datos EES organiza la información del conjunto de datos de manera que la evaluación de la población sea eficiente. El proceso usado tradicionalmente para evaluar un individuo, recorre todos los ejemplos del conjunto de entrenamiento, independientemente de que éstos sean o no clasificados por dicho individuo. El uso de esta estructura permite discriminar aquellos ejemplos que no son cubiertos por una determinada regla, de forma que durante la evaluación de un individuo, sólo sean considerados los ejemplos estrictamente necesarios.

Agradecimientos

Este trabajo ha sido subvencionado por la Comisión Interministerial de Ciencia y Tecnología con el proyecto TIC2001-1143-C03-02.

Referencias

1. J. S. Aguilar. *Generación de Reglas Jerárquicas de Decisión con Algoritmos Evolutivos en Aprendizaje Supervisado*. PhD thesis, Universidad de Sevilla, 2001.
2. J. S. Aguilar-Ruiz, R. Giráldez, and J. C. Riquelme. Natural coding: A more efficient representation for evolutionary learning. In *Lecture Notes in Artificial Intelligence 1415. Springer-Verlag. Genetic and Evolutionary Computation Conference (GECCO'03)*, pages 979–990, Chicago, USA, July 2003.
3. C. Blake and E. K. Merz. UCI repository of machine learning databases, 1998.
4. K. A. DeJong, W. M. Spears, and D. F. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 1(13):161–188, 1993.
5. F. Divina and E. Marchiori. Evolutionary concept learning. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02)*, pages 343–350, New York, 2002.
6. V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.
7. R. Giráldez, J. Aguilar-Ruiz, and J. Riquelme. Discretization oriented to decision rule generation. In *International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES'02), IOS Press*, pages 275–279, Crema, Italy, 2002.
8. R. Giráldez, J. Aguilar-Ruiz, J. Riquelme, and D. Mateos. An efficient data structure for decision rules discovery. In *18th ACM Symposium on Applied Computing, Data Mining Track (SAC'03)*, pages 475–479, Melbourne, Florida, USA, March 2003.
9. R. Giráldez, 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), 2004.
10. R. Giráldez. *Mejoras en Eficiencia y Eficacia de Algoritmos Evolutivos para Aprendizaje Supervisado*. PhD thesis, Universidad de Sevilla, 2004.
11. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.
12. C. Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 1(13):169–228, 1993.
13. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1994.
14. J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, California, 1993.
15. J. R. Quinlan. See5.0 (http://www.rulequest.com), 1998-2001.
16. S. Ruggieri. Efficient C4.5. Technical Report TR-00-01, 2, 2000.
17. K. Shim. *SIGKDD Explorations*, volume 2(2). ACM Press, December 2000.
18. 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.