Aprendizaje de reglas difusas mediante programación genética en problemas con alta dimensionalidad

Francisco José Berlanga,
María José del Jesus
Dept. de Informática
Univ. de Jaén
Campus Las Lagunillas
23071 Jaén
berlanga@ujaen.es, mjjesus@ujaen.es

Francisco Herrera
Dept. de Ciencias de la Computación
e Inteligencia Artificial
ETS Ingeniería Informática
Univ. de Granada
18071 Granada
herrera@decsai.ugr.es

Resumen

El aprendizaje inductivo de un sistema de clasificación basado en reglas difusas (SCBRD) que tenga un alto nivel de interpretabilidad, es una tarea difícil cuando el problema a resolver es de alta dimensionalidad por la presencia de un gran número de características o variables de entrada. Esta dificultad viene dada por el crecimiento exponencial que se produce en el espacio de búsqueda de reglas difusas cuando se incrementa de forma lineal el número de variables consideradas.

En este trabajo proponemos un método basado en programación genética, en el que las reglas difusas en forma normal disyuntiva (DNF) compiten entre sí para obtener un SCBRD con alta interpretabilidad y precisión. Los buenos resultados obtenidos sobre distintos problemas de clasificación abalan nuestra propuesta.

Palabras clave: Clasificación, problemas con alta dimensionalidad, programación genética, reglas difusas, alta interpretabilidad.

1. Introducción

Una de las áreas de aplicación más importantes de la teoría de conjuntos difusos son los Sistemas Basados en Reglas Difusas (SBRDs) aplicados con gran éxito a campos tales como el control, el modelado o la clasificación. Tradicionalmente, el diseño de un SBRD considera como principal objetivo la mejora en el rendimiento o precisión, aunque algunos estudios recientes tienen también en cuenta su interpretabilidad [3].

El proceso de aprendizaje de un SBRD se enfrenta a un espacio de búsqueda de reglas difusas que se incrementa exponencialmente al aumentarse de forma lineal el número de variables. Esto dificulta el proceso de aprendizaje y, en la mayoría de las ocasiones, lleva a la obtención de un SCBRD con una base de reglas (BR) de elevada cardinalidad, lo que disminuye el nivel de interpretabilidad del sistema.

Este problema puede abordarse de formas diferentes, a) compactando y reduciendo el conjunto de reglas previamente obtenido en una etapa de post-procesamiento ([18] [21]), y b) llevando a cabo un proceso de selección de características, antes o durante el proceso de aprendizaje inductivo del SBRD. En la literatura especializada podemos encontrar diversos métodos de selección de características propuestos para el diseño de SCBRDs ([2][4][14][24][25]).

En este trabajo, abordaremos el aprendizaje de SCBRDs interpretables mediante programación genética (PG). La definición de una gramática libre de contexto que permita aprender reglas difusas DNF, junto con el uso de un mecanismo de competición entre reglas que elimine reglas redundantes durante el proceso de aprendizaje, nos permitirán obtener SCBRDs compactos (con pocas reglas y condiciones por regla) que tienen una gran capacidad de predicción.

El artículo se ha organizado de la siguiente forma. En la sección 2 se hace una breve revisión del aprendizaje evolutivo de SCBRDs. En la sección 3, se explican con detalle cada uno de los distintos componentes de nuestra propuesta. La experimentación y el análisis de los resultados obtenidos se describen en la sección 4. Por último, en la sección 5, se muestran las conclusiones y líneas de trabajo futuro.

2. Aprendizaje evolutivo de SCBRDs

En los últimos años, la investigación en el campo de los sistemas difusos ha evolucionado dando lugar a un marco de trabajo más general que contempla la integración de la lógica difusa con otras técnicas tales como los algoritmos evolutivos. Un claro ejemplo son los llamados sistemas basados en reglas difusas evolutivos, en los que el proceso de diseño se afronta como un problema de optimización o búsqueda que el algoritmo evolutivo tiene que resolver.

Dentro de los algoritmos evolutivos, los algoritmos genéticos (AGs) se consideran en la actualidad una de las técnicas de búsqueda global más conocida y empleada. Por este motivo, son numerosos los autores que han usado los AGs para el aprendizaje de SCBRDs. En [10] podemos encontrar un extenso estado del arte.

La programación genética (PG) [20] es un tipo de algoritmo evolutivo que utiliza árboles de longitud variable para representar a los diferentes individuos en la población, en vez de vectores de longitud fija (con codificación binaria, entera o real) como hacen de los AGs.

Podemos encontrar diferentes propuestas que usan el paradigma de la PG para evolucionar conjuntos de reglas difusas. Un trabajo inicial en el tema es la PG Difusa, propuesta por Geyer-Schulz [12], que combina un simple AG que opera sobre un lenguaje libre de contexto con un lenguaje de reglas difusas libre de contexto. Chien y otros, aprenden reglas difusas a partir de funciones discriminantes en [7]. Sánchez y otros proponen un proceso de aprendizaje de SCBRDs en [26] y [11] que combina los operadores de la PG con el enfriamiento simulado y un AG, respectivamente, para establecer las funciones de pertenencia. Mendes y otros, desarrollan en [22] un algoritmo co-evolutivo que incluye un algoritmo basado en PG para el aprendizaje de SCBRDs y un algoritmo evolutivo para obtener las funciones de pertenencia. Tsakonas y otros, proponen en [27] un algoritmo basado en PG para el aprendizaje de sistemas de clasificación basados en reglas difusas y no difusas. Por último, otros autores también han usado la PG para el aprendizaje de SBRDs para modelado ([1], [16]).

3. Propuesta de aprendizaje de SCBRDs mediante PG

En esta sección comentaremos con detalle cada uno de los componentes y/o pasos de nuestra propuesta.

Comenzaremos mostrando el esquema de representación de la solución, así como los diferentes aspectos relacionados con la definición de la gramática y la base de datos. A continuación, explicaremos cada uno de los distintos componentes de la PG como la función de adaptación, el mecanismo de mantenimiento de la diversidad, los operadores genéticos y, por supuesto, los detalles del proceso evolutivo. Finalizaremos describiendo la etapa de post-procesamiento en la que se simplifica la base de reglas y se calcula el grado de certeza de cada regla.

3.1. Representación de la solución

Un factor de especial interés en el diseño de todo proceso de aprendizaje evolutivo de reglas es el esquema de representación usado para codificar cada una de las soluciones [10]. Las diferentes propuestas evolutivas siguen dos enfoques distintos:

- el enfoque *"Cromosoma = Base de Reglas"* o enfoque *Pittsburgh*, y
- el enfoque *"Cromosoma = Regla"*.

En el primer enfoque, cada individuo representa por si mismo una solución completa al codificar un conjunto de reglas, mientras que en el segundo, cada individuo representa una parte de la solución (una regla), por lo que la solución final se obtiene combinando varios individuos. Podemos encontrar tres propuestas genéricas con este último tipo de enfoque:

- el enfoque *Michigan*, en el que cada individuo codifica una única regla. Este tipo de sistemas habitualmente se denominan sistemas clasificadores. Son sistemas de paso de mensajes, basados en reglas, que utilizan aprendizaje por refuerzo y un AG para aprender las reglas guiando su ejecución en un entorno dado [19].
- el enfoque *IRL (Iterative Rule Learning)*, en el que el proceso evolutivo devuelve sólo la

mejor regla aprendida y la solución final se forma mediante la unión de los individuos obtenidos en una serie de ejecuciones sucesivas. MOGUL [9] y SLAVE [15] son propuestas que siguen este enfoque.

- el enfoque "*cooperativo-competitivo"*, en el que la base de reglas se codifica en toda la población o bien en un subconjunto de esta. REGAL [13] y LOGENPRO [29] son ejemplos de este tipo de enfoque.

Nuestra propuesta sigue el enfoque cooperativo-competitivo e incluye un mecanismo que fomenta la creación de nichos para mantener la diversidad dentro de la población. Este mecanismo se describe en la sección 3.5.

3.2. Definición de la gramática

Una vez que hemos aclarado la representación de la solución que se ha utilizado, el siguiente paso en nuestra propuesta consiste en definir una gramática que nos permita aprender una regla difusa por individuo. Además, dicha gramática debe permitir el aprendizaje de reglas en forma normal disyuntiva (DNF), es decir, reglas con la siguiente forma:

$$Si\ X_1\ es\ \hat{A}_1\ y\ ...\ y\ X_n\ es\ \hat{A}_n\ entonces\ Y\ es\ C$$

donde cada variable de entrada X_i toma como valor un conjunto de términos lingüísticos $\hat{A}_i = \{\hat{A}_{i1}\ o\ ...\ o\ \hat{A}_{iLi}\}$, unidos mediante un operador de disyunción.

El uso de este tipo de estructura proporciona una descripción más compacta, lo que mejora la interpretabilidad. Es más, esta estructura permite cambios en la granularidad mediante la combinación de los términos lingüísticos usando el operador de disyunción *"o"* y es un soporte natural para permitir la ausencia de algunas variables de entrada en cada regla (simplemente haciendo que $\hat{A}_i$ sea todo el conjunto de términos lingüísticos).

Por otra parte, se ha incluido en cada regla difusa DNF un grado de certeza ($GC \in [0,1]$), que representa la confianza en la clasificación en la clase indicada en el consecuente de la regla. Este valor se calcula de acuerdo al ratio de ejemplos positivos cubiertos por la regla, pero no se aprende en el proceso evolutivo sino que se calcula una vez finalizado el mismo, según el proceso explicado en la sección 3.8.

En la Tabla 1 podemos ver un ejemplo de una gramática para un problema de clasificación con dos variables de entrada (X_1, X_2), tres etiquetas lingüísticas por variable (Bajo, Medio, Alto) y tres clases (C_1, C_2, C_3), que permite aprender reglas difusas DNF, así como la ausencia de alguna de las variables de entrada.

comienzo ➔ [Si] antec [entonces] consec.
antec ➔ descriptor1 [y] descriptor2.
descriptor1 ➔ [nada].
descriptor1 ➔ [X1 es] etiqueta.
descriptor2 ➔ [nada].
descriptor2 ➔ [X2 es] etiqueta.
Etiqueta ➔ {member (?a, [B, M, A, B o M, B o A, M o A, B o M o A])}, [?a].
consec ➔ [Clase es] descriptorClase.
descriptorClase ➔ {member (?a, [C1, C2, C3])}, [?a].

Tabla 1. Ejemplo de gramática

3.3. Definición de la base de datos

El siguiente paso después de la definición de la gramática, consiste en la definición de la base de datos (BD), es decir, en fijar los parámetros de los conjuntos difusos asociados a las etiquetas presentes en dicha gramática. En nuestros experimentos, hemos usado conjuntos difusos triangulares, y hemos dividido los intervalos de definición de cada variable de forma uniforme.

3.4. Función de adaptación

Hemos usado una función de adaptación basada en el calculo de las dos siguientes medidas:

1. *Confianza*: Mide la precisión de un individuo, es decir, la confianza de que el consecuente es cierto cuando se verifica el antecedente. Se calcula mediante la siguiente expresión:

$$Confianza = \frac{pv}{(pv+pf)} \times \frac{nv}{(nf+nv)}$$

donde *pv* es el número de ejemplos positivos verdaderos, *pf* el de positivos falsos, *nv* el negativos verdaderos y *nf* el de negativos falsos.

2. *Soporte*: Mide la generalidad del conocimiento representado en el individuo y se calcula mediante la siguiente expresión:

$$Soporte = \frac{pv}{(pv + nf)} \times \frac{nv}{(pf + nv)}$$

Ambas medidas se combinan en la función de fitness mostrada en (1), en la que el valor de confianza sólo se tiene en cuenta si el soporte de la regla está por encima de un cierto umbral mínimo definido por el usuario (soporte_min), para evitar así el esfuerzo de evolucionar individuos precisos pero con bajo soporte.

$$F = \begin{cases} \text{Soporte}, & \text{si Soporte} < \text{soporte\_min} \\ \text{Soporte} \times \text{Confianza}, & \text{en otro caso} \end{cases} \quad (1)$$

3.5. Mecanismo de mantenimiento de la diversidad

Son varias las propuestas que se han diseñado para mantener la diversidad usando AGs (crowding, fitness sharing, etc). Estas propuestas se basan en los dos siguientes principios:

1. Los padres se encuentran entre los individuos más similares a sus hijos.
2. El cálculo de alguna medida de similitud entre individuos.

Sin embargo, estos principios presentan problemas cuando tratamos de usarlos con la PG, ya que los padres y los hijos pueden ser totalmente diferentes debido a la naturaleza variable de los individuos. Es más, es mucho más complejo calcular el nivel de similitud entre dos individuos (árboles) en PG. Para resolver estos problemas, se necesita usar una propuesta que no tenga en cuenta la estructura de los individuos. Es por esto, que en nuestra propuesta hemos usado la denominada Competición de Tokens [29].

La idea es la siguiente: En la naturaleza, si un individuo encuentra un buen lugar para vivir (un nicho), intentará explotarlo y prevenir la llegada a este de otros nuevos individuos y tener así que compartir sus recursos, a menos que esos nuevos individuos sean más fuertes que él. Si no es el caso, esos nuevos individuos estarán forzados a explorar y encontrar sus propios nichos. De esta forma se incrementa la diversidad de la población.

Basándonos en esta idea, supondremos que cada ejemplo de entrenamiento puede proporcionar un recurso al que denominaremos *token*. Los individuos de la población competirán para capturar tantos tokens como les sea posible. Cuando un individuo (una regla) consigue emparejar con un ejemplo, establece una bandera o flag para indicar que ha capturado el token asociado a dicho ejemplo. De esta forma, otros individuos más débiles no podrán conseguir el token al no estar ya disponible (aunque también emparejen con el ejemplo al que dicho token está asociado).

El orden en que los individuos compiten para capturar los tokens viene dado por su fortaleza o valor de fitness. Aquellos individuos que presenten valores altos de fitness, podrán capturar tantos tokens como les sea posible. Sin embargo, otros individuos más débiles que entren al mismo nicho verán penalizado su valor de fitness al no poder competir con los otros más fuertes. Para esto, se modifica el valor de fitness de cada individuo de la siguiente forma:

$$\text{Fitness\_modificado} = \text{F} \times \frac{\text{contador}}{\text{ideal}} \quad (2)$$

donde *F* es el valor de fitness obtenido por el individuo en la función de adaptación, *contador* es el número de tokens que el individuo ha conseguido capturar e *ideal* es el número máximo de tokens que el individuo puede capturar, el cual es igual al número de ejemplos con los que el individuo empareja.

Al finalizar la competición de tokens pueden existir individuos que tengan un valor de fitness igual a cero. Esto indicará que dichos individuos no habrán podido capturar ningún token. Por tanto, estos individuos son redundantes y pueden ser reemplazados por otros nuevos, lo que puede aportar un mayor grado de diversidad a la población y además proporcionar cambios adicionales para la generación de buenos individuos. Una opción para generar mejores nuevos individuos consiste en hacer uso de *semillas*. Un ejemplo de posibles semillas, son aquellos ejemplos de entrenamiento cuyos tokens hayan quedado sin capturar al termino de la competición de tokens. Para crear un nuevo individuo, se elige una semilla de forma aleatoria y entonces se genera una regla que la cubra.

3.6. Operadores genéticos

Los descendientes se generan mediante alguno de los siguientes tres operadores genéticos:

1. *Cruce*: Produce un hijo de dos padres. Se selecciona una parte en el primer padre y se reemplaza por otra del segundo padre. Ambas partes se eligen de forma aleatoria pero bajo la restricción de que el hijo generado debe ser válido según las reglas de la gramática.
2. *Mutación*: Una parte de la regla es seleccionada y reemplazada por otra parte generada aleatoriamente. Esta nueva parte se genera usando el mismo mecanismo de derivación que en la creación de la población inicial. Para que el hijo generado sea válido según las reglas de la gramática, se impone la restricción de que la parte seleccionada sólo puede mutar a otra parte con una estructura compatible.
3. *Dropping condition*: Debido a la naturaleza probabilística de la PG, se pueden generar descriptores de variables redundantes, es decir, que no aportan verdadero conocimiento a la regla pues este viene dado por el resto de descriptores. Por este motivo es necesario *generalizar* las reglas. Para generalizar una regla es necesario "podar" alguno de descriptores en la parte del antecedente. Para esto, el operador de dropping condition selecciona aleatoriamente un descriptor del antecedente y lo cambia a "nada" (es decir, aplica la regla de la gramática descriptor$_i$ ➔ [nada]), por lo que ya no se considera en la regla y así esta se generaliza. Este operador fuerza la selección de características en las reglas, permitiéndonos así obtener reglas con un número pequeño de variables en el antecedente.

3.7. Proceso evolutivo

El proceso evolutivo comienza generando de forma aleatoria una población inicial de reglas (de acuerdo a las reglas de producción de la gramática).

La selección de individuos para la reproducción se lleva a cabo mediante el esquema de selección por ranking. En cada iteración se genera un número de descendientes igual al tamaño inicial de la población.

Los individuos de la población actual y los descendientes se unen formando una única nueva población (de tamaño igual al doble del inicial) y entonces se ordenan según su valor de fitness. Una vez que la población está ordenada, se aplica la competición de tokens.

Al termino de esta, los individuos que hayan modificado su fitness a cero (individuos redundantes) se reemplazan por otros nuevos o simplemente se eliminan, en el caso de que no queden ejemplos de entrenamiento sin cubrir. Finalmente, los individuos de la población se vuelven a ordenar según su valor de fitness y se reduce el tamaño de la población a su original.

Es posible, que al finalizar el proceso evolutivo, el tamaño de la población final sea menor que el de la inicial. Esto se debe a la eliminación de individuos redundantes de la población. Esto hace que nuestro método sea capaz de obtener conjuntos de reglas difusas compactos y reducidos, pero que a su vez contienen las reglas necesarias para cubrir a todo el conjunto de entrenamiento. Podemos ver así, como nuestro método está claramente orientado a la obtención de SCBRDs con una alta interpretabilidad sin que ello implique una perdida en el rendimiento del sistema.

3.8. Simplificación de la base de reglas

Una vez que el proceso evolutivo ha terminado, se lleva a cabo una etapa de post-procesamiento para eliminar reglas redundantes. Puede ocurrir que durante el proceso de aprendizaje de la base de reglas, nuestro método aprenda dos reglas tales que una está incluida en la otra. Consideremos por ejemplo las dos reglas mostradas en (3).

R1: Si X_1 es Bajo entonces Clase es C_1
R2: Si X_1 es Bajo o Medio entonces Clase es C_1 (3)

Como podemos ver, la segunda de ellas incluye a la primera, por lo que no tiene sentido mantenerlas a ambas en la población. En este caso, la solución lógica sería eliminar la primera, ya que todos los ejemplos que esta cubre también son cubiertos por la segunda.

Ambas reglas sólo pueden existir en la población final, si R1 entra siempre antes que R2 al proceso de la competición de tokens. Es imposible que ambas lo puedan hacer en el orden

inverso, ya que eso implicaría que R1 debería modificar su fitness a cero (y por tanto se reemplazaría o eliminaría de la población) ya que R2 habría capturado previamente todos sus tokens (ya que R2 cubre al menos los mismos ejemplos que R1).

Este proceso de simplificación tiene como objetivo la mejora de la interpretabilidad del SCBRD aprendido.

3.9. Cálculo del grado de certeza

Finalmente, nuestro método termina con el cálculo del grado de certeza de cada uno de los individuos (reglas) de la población final.

El grado de certeza se calcula como el cociente S_j / S, donde S_j es la suma de los grados de emparejamiento de la regla con los ejemplos de entrenamiento que son cubiertos por el antecedente de la misma y que pertenecen a la misma clase que la que representa el consecuente, y S es la suma de los grados de emparejamiento de la regla con todos los ejemplos de entrenamiento que son cubiertos por el antecedente de la regla, independientemente de la clase a la que estos pertenezcan.

4. Estudio experimental

Para analizar el comportamiento del método propuesto, se ha llevado a cabo un estudio experimental usando las bases de datos Wisconsin, Lung Cancer y Thyroid (Tabla 2) obtenidas del *UCI repository of machine learning Databases* de la Univ. de California (http://www.ics.uci.edu/~mlearn/MLRepository.html).

Nombre	Nº de variables	Nº de ejemplos
Wisconsin	9	699
Lung Cancer	56	32
Thyroid	21	7200

Tabla 2. Bases de datos del UCI

Para cada base de datos, se ha usado el procedimiento 10-fold cross-validation para estimar el error del SCBRS.

Nuestro método (a partir de ahora SCBRD_PG) se ha comparado con otras técnicas de obtención de reglas difusas y con un método de aprendizaje de árboles de decisión:

1. *Wang y Mendel*: En [28] se propone un método de aprendizaje de reglas difusas para control que Chi y otros extienden a problemas de clasificación en [5], [6]. Este método genera una regla difusa por cada ejemplo en el conjunto de entrenamiento, y no lleva a cabo ningún proceso de selección de características.
2. *Ravi y otros*: En [24] se propone un método para derivar reglas difusas, que extrae un conjunto de características reducido a partir del conjunto original mediante el uso del análisis de componentes principales. Tras esto, se lleva a cabo un proceso de aprendizaje de reglas difusas siguiendo el método propuesto en [17]. Finalmente, usa un algoritmo de aceptación de umbral [25] para obtener un conjunto de reglas más compacto, pero que presente un alto poder de clasificación.
3. *SLAVE*: En [15] se propone un método de aprendizaje de reglas difusas DNF mediante un AG con codificación binaria, que utiliza el enfoque *IRL* de representación de la solución. En [14], los autores extienden dicho método al incluir un proceso de selección de características (añadiendo un vector binario que indica si una característica es relevante o no). Denominaremos 2SLAVE a dicha extensión.
4. *Tsakonas y otros*: En [27] se propone un método de aprendizaje de reglas difusas mediante PG que codifica toda la base de reglas en un único individuo (enfoque *Pittsburgh*).
5. *C4.5*: Propuesto por Quinlan [23], es una extensión del algoritmo de clasificación ID3. Está basado en la teoría de la información e incluye también un método de selección de características. Este algoritmo usa un método Divide-y-Vencerás y un criterio denominado ganancia de información, para construir árboles de decisión que pueden ser transformados a posteriori en un conjunto de reglas no difusas.

Nuestro algoritmo termina después de 1000 iteraciones, el tamaño inicial de la población es 20, la probabilidad de cruce es 0.5, la probabilidad de mutación es 0.4, la probabilidad de dropping condition es 0.1, y el umbral mínimo de soporte usado en la función de fitness es 0.01. En todos los experimentos se han usado 5 etiquetas lingüísticas por cada variable.

Los resultados obtenidos para los distintos problemas se muestran en la Tabla 3, donde *#R*

indica el número medio de reglas, *#Var* el número medio de variables por regla, *#Cond* el número medio de condiciones por regla y *%Pru* el porcentaje de acierto sobre el conjunto de prueba. Los subíndices que aparecen junto al porcentaje de prueba están relacionados con el método de razonamiento difuso (FRM) usado, de manera que un 1 se corresponde con el FRM clásico (max-min) y un 2 con el de la suma normalizada [8] (salvo para el algoritmo C4.5 en el que no existe FRM y en el que se ha decidido colocar todos los resultados en la columna correspondiente al 1).

Wisconsin

Método	#R	#Var	#Cond	$\%Pru_1$	$\%Pru_2$
WM	296,5	9	9	66,33	66,19
Ravi	44,77	5	5	86,21	86,21
2SLAVE	3,87	5,47	15,37	88,93	89,77
Tsakonas	18,13	1,27	1,38	62,2	60,37
C4.5	25	4,46	5,08	94,43	-
SCBRD_PG	7,97	1	2,12	91,97	92,8

Lung Cancer

Método	#R	#Var	#Cond	$\%Pru_1$	$\%Pru_2$
WM	28,8	56	56	0	0
Ravi	26,97	7	7	4,44	4,44
2SLAVE	3,73	26,43	59,22	77,97	74,87
Tsakonas	16,3	1,82	1,87	54,44	48,89
C4.5	6,2	2,74	2,82	78,34	-
SCBRD_PG	3,23	1	2,6	82,47	80,5

Thyroid

Método	#R	#Var	#Cond	$\%Pru_1$	$\%Pru_2$
WM	1078	21	21	91,42	91,44
Ravi	35,87	6	6	77,19	77,19
2SLAVE	3,46	9,24	22,93	92,5	92.5
Tsakonas	17,47	1,53	1,63	80,38	57,18
C4.5	24,5	3,69	3,88	99,61	-
SCBRD_PG	5,36	1	2,21	92,13	92,13

Tabla 3. Tablas de resultados

Analizando las tablas de resultados podemos extraer las siguientes consideraciones:

- Nuestro método es el que aprende el conjunto de reglas con el número más bajo de variables y condiciones (etiquetas) por regla, para todos los problemas considerados. Además también aprende bases de reglas con pocas reglas. Por lo tanto, los SCBRDs resultantes presentan un nivel alto de interpretabilidad.
- Si analizamos el comportamiento de nuestra propuesta, podemos ver que nuestro método obtiene buenos porcentajes de prueba en todos los problemas, siendo en algunos el que obtiene el mejor resultado.

5. Conclusiones

En este trabajo, hemos propuesto un método basado en programación genética para obtener SCBRDs con alta interpretabilidad. El hecho de que los individuos de la PG sean representados mediante árboles de longitud variable, puede permitir la ausencia natural de alguna de las variables de entrada, obteniendo reglas con número más bajo de condiciones en el antecedente. Por otro lado, el uso de un mecanismo de formación de nichos que incremente la diversidad de la población, hace que las reglas compitan entre sí, obteniéndose un número más bajo de reglas pero con una gran capacidad de generalización.

La efectividad del método propuesto se ha demostrado sobre varios problemas de clasificación y los resultados son prometedores. Por lo tanto, consideramos que esta propuesta puede ser una alternativa interesante para el aprendizaje de SCBRDs interpretables para problemas que presenten una alta dimensionalidad.

Como trabajo futuro, incorporaremos un algoritmo multiobjetivo apropiado dentro del proceso de aprendizaje.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Ciencia y Tecnología y los fondos FEDER bajo los proyectos TIC-2002-04036-C05-01 y TIC-2002-04036-C05-04, y las redes TIN2004-20061-E y TIN2004-21343-E.

Referencias

[1] A. Bastian. "Identifying fuzzy models utilizing genetic programming". Fuzzy Sets and Systems 113 (3) 1, pp. 333-350, 2000.

[2] J. Casillas, O. Cordón, M.J. Del Jesús, and F. Herrera, "Genetic feature selection in a fuzzy rule-based classification system learning process for high-dimensional problems". Information Sciences 136 (1-4), pp. 135-157, 2001.

[3] J. Casillas, O. Cordón, F.Herrera, L.Magdalena (Eds.). Interpretability Issues in Fuzzy Modeling. Springer-Verlag, 2003. Series Studies in Fuzziness and Soft Computing, Vol. 128.

[4] D. Chakraborty, and N.R. Pal, "A neuro-fuzzy scheme for simultaneous feature selection and fuzzy rule-based classification". IEEE Transactions on Neural Networks 15 (1), pp. 110-123, 2004.

[5] Z. Chi, J. Wu, and H. Yan, "Handwritten numeral recognition using self-organizing maps and fuzzy rules". Pattern Recognition 28 (1), pp. 59-66, 1995.

[6] Z. Chi, H. Yan, and T. Pham, Fuzzy Algorithms with Applications to Image Processing and Pattern Recognition. World Scientific, 1996.

[7] B-C. Chien, J.Y. Lin, and T-P. Hong. "Learning discriminat functions with fuzzy attributes for classification using genetic programming". Expert Systems with Applications 23 (1), pp. 31-37, 2002.

[8] O. Cordón, M.J. del Jesús, and F. Herrera. "A Proposal on Reasoning Methods in Fuzzy Rule-Based Classification Systems". International Journal of Approximate Reasoning 20, pp. 21-45, 1999.

[9] O. Cordón, M.J. del Jesús, F. Herrera and M. Lozano. "MOGUL: A Methodology to Obtain Genetic fuzzy rule-based systems Under the iterative rule Learning approach". International Journal of Intelligent Systems 14 (11), pp. 1123-1153, 1999.

[10] O. Cordón, F. Herrera, F. Hoffmann, L. Magdalena. Genetic Fuzzy Systems. Evolutionary tuning and learning of fuzzy knowledge bases. World Scientific, 2001.

[11] S. García, F. González, and L. Sánchez. "Evolving fuzzy based classifiers with GA-P: A grammatical approach". Genetic Programming: Second European Workshop, EuroGP'99. Lecture Notes in Computer Science 1598, pp 203-210, 1999.

[12] A. Geyer-Schulz. Fuzzy rule-based expert systems and genetic machine learning. Heidelberg: Physica-Verlag, 1995.

[13] A. Giordana, F. Neri. "Search-intensive concept induction". Evolutionary Computation 3 (4), pp. 375-416, 1995.

[14] A. González, and R. Pérez, "Selection of relevant features in a fuzzy genetic learning algorithm". IEEE Transactions on Systems, Man and Cybernetics – Part B 31 (3), pp. 417-425, 2001.

[15] A. González, and R. Pérez, "SLAVE: A genetic learning system based on an iterative approach". IEEE Transactions on Fuzzy Systems 27, pp. 176-191, 1999.

[16] F. Hoffmann, and O. Nelles. "Genetic programming for model selection of TSK-fuzzy systems". Information Sciences 136 (1-4), pp. 7-28, 2001.

[17] H. Ishibuchi, K. Nozaki, and H. Tanaka, "Distributed representation of fuzzy rules and its application to pattern classification". Fuzzy Sets and Systems 52, pp. 21-32, 1992.

[18] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms". IEEE Trans. Fuzzy Systems 3 (3), pp. 260-270, 1995.

[19] T. Kovacs, Strength or Accuracy: Credit Assignment in Learning Classifier Systems. Springer-Verlag, 2004.

[20] J.R. Koza, Genetic programming – on the programming of computers by means of natural selection. Cambridge MA, USA: The MIT Press. 1992.

[21] A. Krone, P. Krause, and T. Slawinski, "A new rule reduction method for finding interpretable and small rule bases in high dimensional search spaces". Proceedings of the Ninth IEEE International Conference on Fuzzy Systems vol. 2, pp. 694-699, May 2000.

[22] R.R.F. Mendes, F. de B. Voznika, A.A. Freitas, et al.. "Discovering Fuzzy Classification Rules with Genetic Programming and Co-evolution". Principles of Data Mining and Knowledge Discovery: 5th European Conference (PKDD'01). Lecture Notes in Computer Science 2168, pp. 314, Springer-Verlag, 2001.

[23] J.R. Quinlan, C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.

[24] V. Ravi, P.J. Reddy, and H.J. Zimmermann, "Pattern classification with principal component analysis and fuzzy rule bases". European Journal of Operational Research 126 (3), pp. 526-533, 2000.

[25] V. Ravi, and H.J. Zimmermann, "Fuzzy rule based classification with FeatureSelector and modified threshold accepting". European Journal of Operational Research 123 (1), pp. 16-28, 2000.

[26] L. Sánchez, I. Couso, and J.A. Corrales. "Combining GP operators with SA search to evolve fuzzy rule based classifiers" Information Sciences 136 (1-4), pp. 175-191, 2001.

[27] A. Tsakonas, G. Dounias, J. Jantzen, H. Axer, B. Bjerregaard and D.G. von Keyserlingk. "Evolving rule-based systems in two medical domains using genetic programming". Artificial Intelligence in Medicine 32 (3), pp. 195-216, 2004.

[28] L.X. Wang, and J.M. Mendel, "Generating fuzzy rules by learning from examples". IEEE Transactions on Systems, Man, and Cybernetics 22 (6), pp. 1414-1427, 1992.

[29] M.L. Wong, and K.S. Leung, Data Mining using grammar based genetic programming and applications. Kluwer Academics Publishers, 2000.