UN SISTEMA DE CLASIFICACIÓN BASADO EN REGLAS DIFUSAS JERÁRQUICO CON PROGRAMACIÓN GENÉTICA PARA PROBLEMAS DE CLASIFICACIÓN ALTAMENTE NO BALANCEADOS

Victoria López[1], **Alberto Fernández**[2], **María José del Jesus**[2], **Francisco Herrera**[1]

[1]*Departamento de Ciencias de la Computación e Inteligencia Artificial, CITIC-UGR, Granada, España,* {vlopez,herrera}@decsai.ugr.es

[2]*Departamento de Informática, Universidad de Jaén, Jaén, España,* {alberto.fernandez,mjjesus}@ujaen.es

Resumen

En muchas aplicaciones del mundo real aparece el problema de clasificación con datos no balanceados. Este problema se da cuando la distribución de datos entre las clases es bastante diferente. Los algoritmos de clasificación no suelen funcionar adecuadamente al abordar esta situación ya que tienden a clasificar incorrectamente las instancias de la clase minoritaria que a menudo es el foco de interés de la aplicación.

Los sistemas de clasificación basados en reglas difusas han conseguido un buen rendimiento en el marco de la clasificación con datos no balanceados. En esta contribución, nos centramos en el uso de un sistema jerárquico con diferentes niveles de granularidad combinado con un procedimiento genético que une la selección de reglas y el ajuste lateral con el modelo lingüísitico de 2-tuplas. Los resultados empíricos en conjuntos con alto desbalanceo reflejan un mejor comportamiento de la propuesta frente a otros sistemas de clasificación basados en reglas difusas y C4.5.

Palabras Clave: Sistemas de clasificación basados en reglas difusas, sistemas difusos evolutivos, particiones difusas jerárquicas, selección, ajuste, clasificación no balanceada

1 Introducción

El problema de clasificación con datos no balanceados, también conocido como de desequilibrio entre clases, aparece cuando el número de ejemplos de una clase es superado notablemente por el número de ejemplos de la otra clase [13, 18]. Este problema se da en aplicaciones reales donde la clase minoritaria es el foco de interés del problema. Los algoritmos de clasificación tradicionales no suelen funcionar correctamente con el desbalanceo ya que se suelen ignorar los ejemplos de la clase minoritaria en favor de los de la clase mayoritaria.

Los Sistemas de Clasificación Basados en Reglas Difusas (SCBRDs) [14] son herramientas útiles dentro del ámbito del aprendizaje automático ya que presentan un buen equilibrio entre interpretabilidad y precisión. El buen comportamiento de los SCBRDs en conjuntos de datos no balanceados ha sido analizado en [10, 11]. La definición automática de los componentes de un sistema difuso utilizando algoritmos evolutivos ha dado lugar a la aparición de los sistemas difusos evolutivos. Dentro de ellos, la programación genética evoluciona un conjunto de soluciones en forma de árbol con longitud variable, permitiendo obtener una base de reglas adecuada para un SCBRD [5]. Sin embargo, una desventaja de los SCBRDs es la inflexibilidad del concepto de variable lingüística porque impone fuertes restricciones en la estructura de la regla difusa que pueden suponer una pérdida de precisión en problemas de alta complejidad.

Para mejorar el rendimiento de los SCBRDs se han propuesto diversos mecanismos que incrementan la cooperación entre las reglas de la Base de Conocimiento (BC). En este trabajo se presenta un SCBRD que promueve la sinergia entre varias de estas estrategias: el uso de una Base de Conocimiento Jerárquica (BCJ) [8], la selección evolutiva de reglas y el ajuste lateral. En concreto, se propone el uso de un SCBRD Jerárquico dividido en tres etapas para abordar el problema de clasificación no balanceada. En un primer paso, la técnica "Synthetic Minority Over-sampling Technique" (SMOTE) [6] equilibra la distribución del conjunto de entrenamiento. A continuación, se genera una BCJ con el algoritmo "Genetic Programming-based learning of COmpact and ACcurate fuzzy rule-based classification systems for High-dimensional problems" (GP-COACH) [5] modificado para permitir la expansión de una BC clásica en una BC jerárquica. Finalmente, se añade un postprocesamiento evolutivo que combina selección de reglas con un ajuste lateral de las etiquetas basado en el modelo lingüístico de 2-tuplas para incrementar el rendimiento final del modelo [1]. Este modelo se ha denominado "GP-COACH Hierarchical" (GP-COACH-H).

Para mostrar la bondad de la propuesta hemos realizado una comparación frente a otros SCBRDs y C4.5 sobre 44 conjuntos de datos de alto no balanceo, escogidos del repositorio KEEL [2]. Este estudio ha sido contrastado a través de test estadísticos no paramétricos [12, 17].

La organización del trabajo es como sigue. En la Sección 2 se introduce el problema de clasificación con datos no balanceados. La Sección 3 describe el método propuesto detallando sus etapas. A continuación, en la Sección 4, mostramos el estudio experimental llevado a cabo.

2 Conjuntos de datos no balanceados en clasificación

Comenzamos esta sección introduciendo el problema de clasificación con datos no balanceados. Seguidamente, describimos qué técnicas se utilizan para abordar este problema destacando el algoritmo SMOTE [6]. Terminamos explicando las métricas de evaluación para estos problemas.

2.1 El problema de clasificación no balanceada

El aprendizaje a partir de conjuntos de datos no balanceados es un problema que ha cobrado importancia en los últimos tiempos en minería de datos [13, 18]. Este problema es muy representativo ya que aparece en muchas aplicaciones del mundo real como en usos médicos, detección de anomalías, telecomunicaciones o finanzas. Nos referimos a conjuntos de datos no balanceados cuando la distribución entre clases no es uniforme. En esta situación el número de ejemplos que representan a una clase (normalmente, la que acapara el mayor interés) es mucho menor que el del resto.

Los algoritmos clásicos para clasificación tienden a clasificar correctamente los ejemplos de la clase negativa (clase predominante) ya que dichos algoritmos suelen buscar reglas generales que abarquen el mayor número de ejemplos, de acuerdo a las métricas utilizadas tradicionalmente. De esta manera, las instancias que pertenecen a la clase positiva (clase minoritaria) se clasifican incorrectamente con más frecuencia que las que pertenecen a la clase negativa.

En este trabajo utilizamos el "imbalance ratio" (IR) [15] para distinguir distintos grados de no balanceo en los datos. El IR se define como la razón entre el número de ejemplos de la clase negativa y el número de ejemplos de la clase positiva. Consideraremos que un conjunto de datos presenta un alto grado de no balanceo cuando su IR es mayor que 9 (menos de un 10 % de instancias de la clase positiva).

2.2 Aproximaciones para la clasificación de conjuntos de datos no balanceados: el algoritmo SMOTE

Se han utilizado un gran número de enfoques para abordar el problema de clasificación con datos no balanceados. Estos enfoques se pueden catalogar en dos grupos: enfoques a nivel algorítmico que crean o modifican procedimientos teniendo en cuenta el no balanceo de los datos o enfoques a nivel de datos que preprocesan el conjunto de entrenamiento para disminuir el efecto del desbalanceo. En este trabajo se utilizan métodos de sobremuestreo que son un referente en el área: el algoritmo SMOTE [6] y una variante, "SMOTE + Edited Nearest Neighbor" (SMOTE+ENN) [4].

El algoritmo SMOTE aumenta el número de ejemplos de la clase positiva introduciendo instancias sintéticas en los segmentos que unen cualquiera de las instancias positivas con alguno de sus k vecinos más cercanos, consiguiendo que la región de instancias positivas sea más general. En ocasiones, los ejemplos generados por SMOTE ocupan zonas del espacio de la clase negativa afectando negativamente al resultado final de la clasificación. Para evitarlo, se utiliza el enfoque híbrido SMOTE+ENN en el que se aplica el algoritmo ENN tras el uso de SMOTE.

2.3 Métricas de evaluación

Muchas de las propuestas de aprendizaje automático de clasificadores utilizan alguna medida de precisión como el porcentaje de ejemplos bien clasificados. Sin embargo, el uso de este tipo de medidas puede conducir a conclusiones erróneas en conjuntos de datos no balanceados ya que no tienen en cuenta la proporción de ejemplos de cada clase.

La medida utilizada en este trabajo es la media geométrica (MG) [3], que se define como $MG = \sqrt{VP_{tasa} \cdot VN_{tasa}}$, donde VP_{tasa} es el porcentaje de ejemplos de la clase positiva bien clasificados y VN_{tasa} es el porcentaje de ejemplos de la clase negativa bien clasificados. Esta medida trata de maximizar el acierto de cada una de las dos clases equiparando estos dos objetivos en una única medida.

3 GP-COACH-H: Un clasificador basado en reglas difusas jerárquico con programación genética

En esta sección se describe nuestra propuesta de un clasificador basado en reglas difusas jerárquico para problemas de clasificación con alto no balanceo, GP-COACH-H. Este método está compuesto por tres etapas:

1. Preprocesamiento del conjunto de entrenamiento con el algoritmo SMOTE.
2. Generación de la Base de Reglas Jerárquica (BRJ) con el algoritmo GP-COACH modificado.
3. Post-procesamiento evolutivo: selección de reglas y ajuste lateral 2-tuplas.

El funcionamiento de este algoritmo se encuentra resumido en la Figura 1. A continuación se describen con mayor detalle las operaciones de este algoritmo.

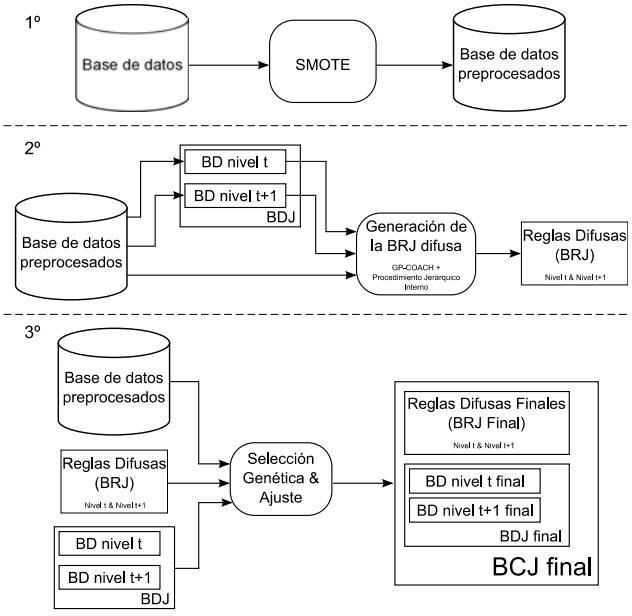

Figura 1: Funcionamiento de GP-COACH-H

3.1 Sistemas de clasificación basados en reglas difusas

Un SCBRD tiene dos componentes principales: el Sistema de Inferencia y la BC. En un SCBRD, la BC está compuesta de la Base de Reglas (BR), constituida por el conjunto de reglas difusas y la Base de Datos (BD), que contiene las funciones de pertenencia de las particiones difusas asociadas a las variables de entrada. Si no existe información experta sobre el problema, es necesario utilizar algún proceso de aprendizaje automático de la BC a partir de ejemplos.

En este trabajo se usan reglas difusas tipo "Disjunctive Normal Form" (DNF) (Ecuación 1). El peso de una regla se calcula con el factor de certeza (Ecuación 2). GP-COACH-H utiliza la suma normalizada como método de razonamiento difuso para clasificar nuevos ejemplos (Ecuación 3).

$$\begin{array}{c}\text{Regla } R_j: \text{ Si } x_1 \text{ es } \hat{A}_1^j \text{ y } \ldots \text{ y } x_n \text{ es } \hat{A}_n^j \\ \text{entonces Clase} = C_j \text{ con } RW_j\end{array} \quad (1)$$

$$RW_j = CF_j = \frac{\sum_{x_p \in Clase C_j} \mu_{\hat{A_j}}(x_p)}{\sum_{p=1}^{N} \mu_{\hat{A_j}}(x_p)} \quad (2)$$

$$\begin{array}{c} Sum_{Clase\ h}(x_p) = \frac{\sum_{R_j \in BR, C_j = h} \mu_{\hat{A_j}}(x_p) \cdot CF_j}{\text{máx}_{c=1,\cdots,M} \sum_{R_j \in BR, C_j = c} \mu_{\hat{A_j}}(x_p) \cdot CF_j} \\ Clase(x_p) = \arg\text{máx}(Sum_{Class\ h}(x_p)) \end{array} \quad (3)$$

En estas ecuaciones, R_j es la etiqueta de la j-ésima regla, $x = (x_1, \ldots, x_n)$ es un vector de muestras n-dimensional (variables de entrada), $\hat{A}_i^j$ es el conjunto de etiquetas lingüísticas $\{L_i^1 \text{ o } \ldots \text{ o } L_i^{l_i}\}$, C_j es la etiqueta de una clase, RW_j es el peso de la regla y $\mu_{A_j}(x_p)$ es el grado de emparejamiento del ejemplo x_p con el antecedente de la regla difusa R_j. Representamos las funciones de pertenencia por medio de funciones triangulares.

3.2 El algoritmo GP-COACH

El algoritmo GP-COACH [5] es un algoritmo de programación genética para el aprendizaje de bases de reglas difusas. Utilizamos este método como base de nuestra propuesta jerárquica modificando su comportamiento para incluir varios niveles de granularidad dentro de su comportamiento.

Este algoritmo trabaja con una población de reglas tipo DNF (Ecuación 1) que permite la ausencia de algunas variables de entrada. Cada individuo de la población codifica una regla y la BR está formada por todos la población al completo. Durante la evolución de la población se utilizan dos funciones de evaluación: una función local que mide la idoneidad de cada regla y una función global que evalúa la población al completo. Además, se incorpora un mecanismo para mantener la diversidad en la población y eliminar reglas irrelevantes, el procedimiento de 'token competition". GP-COACH incorpora un procedimiento de inferencia en dos fases utilizando primero reglas de tipo 1 y si no es posible su aplicación se aplican reglas de tipo 2. Para una descripción más detallada del algoritmo GP-COACH se recomienda consultar [5].

3.3 Construcción del Sistema Basado en Reglas Difusas Jerárquico

Los Sistemas Basados en Reglas Difusas Jerárquicos tratan de mejorar el rendimiento de los sistemas basados en reglas difusas en zonas del espacio de búsqueda especialmente difíciles. Para ello, en lugar de utilizar la definición clásica de la BC, utilizamos una expansión conocida como BCJ [8], que se compone de una serie de niveles. Cada uno de estos niveles contiene una BD y una BR asociadas. La BD de cada nivel contiene tantas particiones lingüísticas como indique la granularidad del nivel y la BR contiene las reglas que toman valores en las particiones lingüísticas anteriores. Este conjunto de niveles está organizado en una jerarquía de manera que la granularidad de los diferentes niveles aumenta conforme lo hace su orden.

GP-COACH-H utiliza una BCJ con dos niveles de granularidad. Las BD de ambos niveles se generan a partir del conjunto de entrenamiento con etiquetas triangulares simétricas y uniformemente distribuidas de forma similar a la utilizada en [8, 10]. El uso de diferentes niveles de jerarquía obliga a introducir modificaciones en GP-COACH.

GP-COACH-H mantiene una población mixta que contiene reglas de los diferentes niveles de granularidad, adapta los operadores genéticos a esta situación y modifica la función de evaluación de la población en consecuencia. El procedimiento para jerarquizar las reglas actuales comienza tras la creación de reglas de tipo 2. Para ello, se cataloga cada una de las reglas como *buena* o *mala*:

if $(Ac(X, R_j) \leq (1-\alpha) \cdot Ac_i(X_i, RB))$ **then**
R_j = *regla buena*

else
R_j = *regla mala*
end if

$$Ac(X,R_j) = \frac{|X^+(R_j)|}{|X(R_j)|} \quad (4)$$

$$Ac_i(X_i,RB) = \frac{|x_p \in X_i / MRD(x_p,RB) = Clase(x_p)|}{|X_i|} \quad (5)$$

donde $|\,|$ es el número de ejemplos, $X^+(R_j)$ y $X(R_j)$ son los conjuntos de ejemplos en los que la regla colabora en su clasificación (correcta clasificación para $X^+(R_j)$), X_i es conjunto de ejemplos de entrenamiento pertenecientes a la i-ésima clase, $MRD(x_p,RB)$ es la clase predicha para un ejemplo y $Clase(x_p)$ es la clase del ejemplo x_p.

Las reglas que se han considerado *buenas* se mantienen en la población mientras que las reglas *malas* se eliminan de la población actual. Se añaden a la población nuevas reglas de alta granularidad generadas con el algoritmo Chi [7] sobre los ejemplos que eran clasificados por las reglas *malas*.

Gracias al uso de estas operaciones, conseguimos que GP-COACH-H obtenga un buen conjunto de reglas de diferentes granularidades. Para una información más extensa de la estructura de la BCJ y la generación de reglas de alta granularidad a partir de un conjunto de reglas ya generado se sugiere la lectura de [8, 10].

3.4 Selección Evolutiva de Reglas Jerárquicas y Ajuste Lateral

En este último paso, se presenta el uso de algoritmos genéticos para seleccionar un subconjunto de reglas difusas compacto y cooperativo y ajustar las funciones de pertenencia de las etiquetas en la BD realizando desplazamientos laterales en las mismas según el modelo 2-tuplas [1, 10, 11]. La representación de las etiquetas según el modelo 2-tuplas permite realizar la búsqueda del desplazamiento lateral teniendo en cuenta un único parámetro de transición simbólica: un número del intervalo $[-0.5, 0.5)$ que expresa el dominio en el que se puede mover una etiqueta lingüística.

Para llevar a cabo este proceso de selección de reglas y ajuste lateral utilizamos el modelo genético del método CHC [9]. Los individuos de la población codifican los dos objetivos a alcanzar: una parte del cromosoma indica si una regla ha sido seleccionada para la solución final o no y la otra parte del cromosoma contiene valores reales que indican el desplazamiento para cada una de las etiquetas de las variables en las dos BD de diferentes niveles de jerarquía.

Para evaluar un cromosoma se calcula el acierto sobre el conjunto de entrenamiento usando la BC con las modificaciones representadas en su codificación. El cruce entre dos individuos se realiza de manera distinta para cada parte del cromosoma: se usa el operador HUX para la parte con codificación binaria y el operador PCBLX para la parte con codificación real del cromosoma. Se recomienda consultar trabajos previos que se usan un esquema similar [1, 10, 11].

Tabla 1: Conjuntos de datos altamente no balanceados

Nombre	#Ej.	#Var.	Clase (+,-)	%Clase(+; -)	IR
ecoli034vs5	200	7	(p,imL,imU; om)	(10.00, 90.00)	9.00
yeast2vs4	514	8	(cyt; me2)	(9.92, 90.08)	9.08
ecoli067vs35	222	7	(cp,omL,pp; imL,om)	(9.91, 90.09)	9.09
ecoli0234vs5	202	7	(cp,imS,imL,imU; om)	(9.90, 90.10)	9.10
glass015vs2	172	9	(build-win-non_float-proc,tableware, build-win-float-proc; ve-win-float-proc)	(9.88, 90.12)	9.12
yeast0359vs78	506	8	(mit,me1,me3,erl; vac,pox)	(9.88, 90.12)	9.12
yeast02579vs368	1004	8	(mit,cyt,me3,vac,erl; me1,exc,pox)	(9.86, 90.14)	9.14
yeast0256vs3789	1004	8	(mit,cyt,me3,exc; me1,vac,pox,erl)	(9.86, 90.14)	9.14
ecoli046vs5	203	6	(cp,imU,omL; om)	(9.85, 90.15)	9.15
ecoli01vs235	244	7	(cp,im; imS,imL,om)	(9.83, 90.17)	9.17
ecoli0267vs35	224	7	(cp,imS,omL,pp; imL,om)	(9.82, 90.18)	9.18
glass04vs5	92	9	(build-win-float-proc,containers; tableware)	(9.78, 90.22)	9.22
ecoli0346vs5	205	7	(cp,imL,imU,omL; om)	(9.76, 90.24)	9.25
ecoli0347vs56	257	7	(cp,imL,imU,pp; om,omL)	(9.73, 90.27)	9.28
yeast05679vs4	528	8	(me2; mit,me3,exc,vac,erl)	(9.66, 90.34)	9.35
ecoli067vs5	220	6	(cp,omL,pp; om)	(9.09, 90.91)	10.00
vowel0	988	13	(hid; remainder)	(9.01, 90.99)	10.10
glass016vs2	192	9	(ve-win-float-proc; build-win-float-proc, build-win-non_float-proc,headlamps)	(8.89, 91.11)	10.29
glass2	214	9	(ve-win-float-proc; remainder)	(8.78, 91.22)	10.39
ecoli0147vs2356	336	7	(cp,im,imU,pp; imS,imL,om,omL)	(8.63, 91.37)	10.59
led7digit02456789vs1	443	7	(0,2,4,5,6,7,8,9; 1)	(8.35, 91.65)	10.97
glass06vs5	108	9	(build-win-float-proc,headlamps; tableware)	(8.33, 91.67)	11.00
ecoli01vs5	240	6	(cp,im; om)	(8.33, 91.67)	11.00
glass0146vs2	205	9	(build-win-float-proc,containers,headlamps, build-win-non_float-proc;ve-win-float-proc)	(8.29, 91.71)	11.06
ecoli0147vs56	332	6	(cp,im,imU,pp; om,omL)	(7.53, 92.47)	12.28
cleveland0vs4	177	13	(0; 4)	(7.34, 92.66)	12.62
ecoli0146vs5	280	6	(cp,im,imU,omL; om)	(7.14, 92.86)	13.00
ecoli4	336	7	(om; remainder)	(6.74, 93.26)	13.84
yeast1vs7	459	8	(nuc; vac)	(6.72, 93.28)	13.87
shuttle0vs4	1829	9	(Rad Flow; Bypass)	(6.72, 93.28)	13.87
glass4	214	9	(containers; remainder)	(6.07, 93.93)	15.47
page-blocks13vs2	472	10	(graphic; horiz.line,picture)	(5.93, 94.07)	15.85
abalone9vs18	731	8	(18; 9)	(5.65, 94.25)	16.68
glass016vs5	184	9	(tableware; build-win-float-proc, build-win-non_float-proc,headlamps)	(4.89, 95.11)	19.44
shuttle2vs4	129	9	(Fpv Open; Bypass)	(4.65, 95.35)	20.5
yeast1458vs7	693	8	(vac; nuc,me2,me3,pox)	(4.33, 95.67)	22.10
glass5	214	9	(tableware; remainder)	(4.20, 95.80)	22.81
yeast2vs8	482	8	(pox; cyt)	(4.15, 95.85)	23.10
yeast4	1484	8	(me2; remainder)	(3.43, 96.57)	28.41
yeast1289vs7	947	8	(vac; nuc,cyt,pox,erl)	(3.17, 96.83)	30.56
yeast5	1484	8	(me1; remainder)	(2.96, 97.04)	32.78
ecoli0137vs26	281	7	(pp,imL; cp,im,imU,imS)	(2.49, 97.51)	39.15
yeast6	1484	8	(exc; remainder)	(2.49, 97.51)	39.15
abalone19	4174	8	(19; remainder)	(0.77, 99.23)	128.87

4 Experimentos y análisis de resultados

En esta sección, nuestro objetivo consiste en analizar el comportamiento del algoritmo GP-COACH-H. Para ello, escogemos 44 conjuntos de datos con un grado alto de no balanceo ($IR > 9$) del repositorio KEEL[1] [2] mostrados en la Tabla 1, donde se indica el número de ejemplos de cada conjunto (#Ex.), número de variables (#Var.), nombre de cada clase (positiva y negativa), distribución de ejemplos y tasa de no balanceo (IR). Los conjuntos de datos están ordenados en orden ascendente de IR.

El rendimiento del algoritmo GP-COACH-H se compara con otros SCBRDs, en concreto, comparamos la propuesta frente a la versión básica del algoritmo GP-COACH [5] con diferentes niveles de granularidad (5 y 9 etiquetas), el SCBRD jerárquico HFRBCS(Chi) [10] y el árbol de decisión C4.5 [16]. Los parámetros de configuración de los algoritmos considerados (abreviados como GP-C-H, GP-C-5 y 9, Chi-H, y C4.5 en las tablas) se muestran en la Tabla 2 y corresponden a los parámetros recomendados por los autores de los algoritmos. Para GP-COACH-H, escogemos los parámetros asociados a su algoritmo base GP-COACH y al

[1] http://www.keel.es/imbalanced.php

Tabla 2: Parámetros de configuración de los algoritmos

Algoritmo	Parámetros
Parámetros de los SCBRD	
GP-C y GP-C-H	T-norma mínimo, T-conorma máximo, Peso de reglas = Factor de certeza, Método de razonamiento difuso = Suma normalizada, Número de etiquetas difusas = 5 ó 9 (en GP-C), Número de etiquetas difusas = 5 para reglas de baja granularidad y 9 para reglas de alta granularidad (en GP-C-H)
Chi-H	T-norma producto, Peso de reglas = Factor de certeza penalizado, Método de razonamiento difuso = Regla ganadora, Número de etiquetas difusas = 3 para reglas de baja granularidad y 5 para reglas de alta granularidad
Parámetros de las versiones GP-COACH	
GP-C y GP-C-H	Evaluaciones = 20000, Tamaño de la población inicial = 200, α (raw fitness) = 0.7, Probabilidad de cruce = 0.5, Probabilidad de mutación = 0.2, Probabilidad de pérdida de una condición = 0.15, Probabilidad de inserción = 0.15, Tamaño del torneo = 2, $w_1 = 0.8$, $w_2 = w_3 = 0.05$, $w_4 = 0.1$
Parámetros asociados al proceso de expansión jerárquica	
Chi-H y GP-C-H	α (expansión regla) = 0.2, Evaluaciones CHC = 10000, Tamaño de la población CHC = 61, bits por gen CHC (GP-C-H) = 30
Parámetros de C4.5	
C4.5	Podado = true, Confianza = 0.25, Número mínimo de conjuntos por hoja = 2

proceso de expansión jérarquica de HFRBCS(Chi).

Para contrarrestar el efecto del no balanceo en los datos se utiliza el algoritmo SMOTE [6] para preprocesar los conjuntos de datos a utilizar por los SCBRDs y la variante SMOTE+ENN [4] para los conjuntos de datos con C4.5. Ambos algoritmos equilibran el número de ejemplos en el conjunto de entrenamiento al 50 % utilizando únicamente el vecino más cercano según la distancia euclídea.

Los distintos experimentos se llevan a cabo bajo un esquema de validación cruzada con 5 particiones, esto es, 5 particiones con el 20 % de los datos siendo la combinación de 4 de ellas (80 %) el conjunto de entrenamiento y la restante el conjunto de test. Para cada conjunto de datos se muestran los resultados medios de las cinco particiones.

Además, se utilizan técnicas estadísticas para encontrar diferencias significativas entre los métodos estudiados [12]. Consideramos el uso de test estadísticos no paramétricos debido a que las condiciones iniciales de aplicabilidad de test paramétricos podrían no ser satisfechas por lo que el análisis estadístico perdería credibilidad [17]. Concretamente, usamos el test de Iman-Davenport para detectar si existen diferencias significativas entre un grupo de algoritmos y el test post-hoc de Holm que muestra las diferencias que hay entre los algoritmos y un método de control.

La Tabla 3 muestra la medida MG por conjunto de datos en test para cada uno de los algoritmos utilizados en la comparación. Observando los resultados medios que aparecen en dicha tabla, podemos ver que GP-COACH-H es el algoritmo que obtiene el mayor valor de MG. Para contrastar estos resultados por medio de test estadísticos comenzamos con la aplicación del test de Iman-Davenport que determina la existencia de diferencias significativas entre algoritmos. El p-valor obtenido $(0,0779)$ es suficientemente bajo como para rechazar la hipótesis de igualdad con un nivel

Tabla 3: Tabla de resultados para los SCBRDs y C4.5 en los conjuntos de datos con alto desbalanceo. Preprocesamiento con SMOTE para los SCBRDs y SMOTE+ENN para C4.5

Conjunto de datos	GP-C-5	GP-C-9	Chi-H	GP-C-H	C4.5
ecoli034vs5	0,9018	**0,9250**	0,8421	0,8660	0,8761
yeast2vs4	0,8987	0,9036	0,8932	**0,9304**	0,9029
ecoli067vs35	0,8185	**0,8509**	0,8267	0,7286	0,7206
ecoli0234vs5	0,8286	0,8472	0,8425	0,8473	**0,8861**
glass015vs2	0,3732	0,2115	0,5590	0,6301	**0,7788**
yeast0359vs78	0,5111	0,4467	**0,7330**	0,7189	0,6894
yeast02579vs368	0,9087	0,9044	0,8946	0,9107	**0,9125**
yeast0256vs3789	0,7954	0,7955	0,7927	**0,7982**	0,7707
ecoli046vs5	**0,9171**	0,8793	0,8800	0,8677	0,8776
ecoli01vs235	0,8682	**0,9138**	0,8709	0,8471	0,8277
ecoli0267vs35	0,8407	0,8365	0,8247	**0,9028**	0,8061
glass04vs5	0,9064	0,9632	0,7092	0,9429	**0,9748**
ecoli0346vs5	0,8772	0,8934	0,8729	0,8847	**0,8946**
ecoli0347vs56	0,8525	0,8571	**0,9007**	0,8767	0,8413
yeast05679vs4	**0,8136**	0,8080	0,7318	0,6988	0,7678
ecoli067vs5	0,8356	**0,8923**	0,8559	0,8671	0,8376
vowel0	0,9598	0,9400	**0,9882**	0,9465	0,9417
glass016vs2	0,5419	0,4211	0,5837	**0,6467**	0,6063
glass2	0,6223	0,3949	0,5484	0,5886	**0,7377**
ecoli0147vs2356	0,7976	0,7972	**0,8477**	0,8263	0,8119
led7digit02456789vs1	**0,9011**	0,8874	0,8276	0,9000	0,8370
glass06vs5	**0,9949**	0,9060	0,8907	0,9120	0,9628
ecoli01vs5	0,8739	**0,9190**	0,8689	0,8946	0,8081
glass0146vs2	0,6651	0,5675	0,5117	**0,7300**	0,6157
ecoli0147vs56	0,8472	0,8577	**0,8886**	0,8372	0,8250
cleveland0vs4	0,7232	0,7578	0,3961	**0,8646**	0,7307
ecoli0146vs5	0,8832	0,8862	0,8674	**0,9194**	0,8880
ecoli4	**0,9373**	0,9008	0,9302	0,9357	0,8947
yeast1vs7	0,6802	0,6093	0,7074	0,6900	**0,7222**
shuttle0vs4	0,9991	0,9954	0,9912	**1,0000**	0,9997
glass4	0,7231	**0,8811**	0,7039	0,7303	0,7639
page-blocks13vs4	0,9035	0,8706	0,9864	0,9482	**0,9909**
abalone9-18	0,6922	0,6699	0,6756	**0,7500**	0,6884
glass016vs5	**0,9644**	0,9090	0,7796	0,8550	0,7738
shuttle2vs4	0,9568	0,9960	0,9749	0,9918	**1,0000**
yeast1458vs7	0,3546	0,5353	0,6249	**0,6304**	0,3345
glass5	0,5801	0,6758	0,6873	**0,7877**	0,5851
yeast2vs8	0,7274	0,7601	0,7247	0,7381	**0,8033**
yeast4	0,7923	0,7807	**0,8264**	0,8175	0,6897
yeast1289vs7	0,5262	0,5860	0,6937	**0,6939**	0,5522
yeast5	0,9020	0,9229	0,9420	**0,9428**	0,9390
ecoli0137vs26	**0,7215**	0,7203	0,7148	0,7067	0,7062
yeast6	**0,8856**	0,8574	0,8492	0,8170	0,8029
abalone19	0,6425	0,5828	**0,7019**	0,5532	0,1550
Media	0,7897	0,7845	0,7901	**0,8175**	0,7848

alto de confianza. Por ello, podemos concluir que existen diferencias significativas especificadas en la Tabla 4 con los ranking medios de los algoritmos y su p-valor ajustado calculado según el test post-hoc de Holm. En esta tabla se puede ver que GP-COACH-H obtiene el menor ranking de los algoritmos, lo que lo convierte en el método de control utilizado en el cálculo del procedimiento post-hoc. Como todos los p-valores son bajos, se puede rechazar la hipótesis nula en todos los casos, reforzando así la suposición de que GP-COACH-H es el método que mejor se comporta para conjuntos de datos altamente no balanceados.

Tabla 4: Ranking medio y p-valores ajustados según el procedimiento post-hoc de Holm para los SCBRDs y C4.5

Algoritmo	Ranking medio	p-valor ajustado
GP-COACH-H	2,4091	
GP-COACH-9	3,0227	0,0862
GP-COACH-5	3,0909	0,0862
C4.5	3,2045	0,0549
HFRBCS(Chi)	3,2727	0,0416

5 Conclusiones

En este trabajo hemos presentado un SCBRD con diferentes niveles de granularidad que integra la selección de reglas y el ajuste lateral 2-tuplas para mejorar el rendimiento en conjuntos de datos altamente no balanceados, GP-COACH-H. La propuesta combina el muestreo de datos con modificaciones algorítmicas del enfoque básico y adapta su comportamiento a los diferentes niveles de granularidad considerados, añadiendo un paso de postprocesamiento que ayuda al sistema de clasificación basado en reglas difusas jerárquico a adecuarse al contexto del problema y por tanto, mejorar el comportamiento global.

La propuesta se ha comparado con la versión básica de GP-COACH, el algoritmo HFRBCS(Chi) y con C4.5 para demostrar su buen comportamiento. Los resultados obtenidos muestran el buen rendimiento de la propuesta en el ámbito de la clasificación sobre los conjuntos altamente no balanceados superando al resto de algoritmos de la comparación. Estos buenos resultados se han obtenido gracias a una integración adecuada de las diferentes estrategias que se han usado en la propuesta, ya que han sido adaptadas teniendo en cuenta las características del problema a resolver.

Agradecimientos

Este trabajo ha sido posible gracias a la subvención del Ministerio de Ciencia e Innovación bajo los proyectos TIN2011-28488 y TIN2008-06681-C06-02; y la Junta de Andalucía mediante los proyectos P10-TIC-6858 y TIC-3928. V. López es beneficiaria de una beca FPU del Ministerio de Educación.

Referencias

[1] R. Alcalá, J. Alcalá-Fdez, F. Herrera: A proposal for the genetic lateral tuning of linguistic fuzzy systems and its interaction with rule selection, *IEEE Transactions on Fuzzy Systems* 15 (4), pp. 616–635, 2007.

[2] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera: KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, *Journal of Multi-Valued Logic and Soft Computing* 17 (2–3), pp. 255–287, 2011.

[3] R. Barandela, J. S. Sánchez, V. García, E. Rangel: Strategies for learning in class imbalance problems, *Pattern Recognition* 36 (3), pp. 849–851, 2003.

[4] G. E. A. P. A. Batista, R. C. Prati, M. C. Monard: A study of the behaviour of several methods for balancing machine learning training data, *SIGKDD Explorations* 6 (1), pp. 20–29, 2004.

[5] F. J. Berlanga, A. J. Rivera, M. J. del Jesus, F. Herrera: GP-COACH: Genetic programming-based learning of compact and accurate fuzzy rule-based classification systems for high-dimensional problems, *Information Sciences* 180, pp. 1183–1200, 2010.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer: SMOTE: Synthetic minority over-sampling technique, *Journal of Artificial Intelligent Research* 16, pp. 321–357, 2002.

[7] Z. Chi, H. Yan, T. Pham: *Fuzzy algorithms with applications to image processing and pattern recognition*. World Scientific, 1996.

[8] O. Cordon, F. Herrera, I. Zwir: Linguistic modeling by hierarchical systems of linguistic rules, *IEEE Transactions on Fuzzy Systems* 10 (1), pp. 2–20, 2002.

[9] L. J. Eshelman: The CHC adaptive search algorithm: How to have safe serach when engaging in nontraditional genetic recombination. En: G. J. E. Rawlins (Ed.) *Foudations of Genetic Algorithms*, Morgan Kaufmann, pp. 265–283, 1990.

[10] A. Fernández, M. J. del Jesus, F. Herrera: Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets, *International Journal of Approximate Reasoning* 50, pp. 561–577, 2009.

[11] A. Fernández, M. J. del Jesus, F. Herrera: On the 2-tuples based genetic tuning performance for fuzzy rule based classification systems in imbalanced data-sets, *Information Sciences* 180 (8), pp. 1268–1291, 2010.

[12] S. García, A. Fernández, J. Luengo, F. Herrera: A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, *Soft Computing* 13 (10), pp. 959–977, 2009.

[13] H. He, E. A. Garcia: Learning from imbalanced data, *IEEE Transactions on Knowledge and Data Engineering* 21 (9), pp. 1263–1284, 2009.

[14] H. Ishibuchi, T. Nakashima, M. Nii: *Classification and modeling with linguistic information granules: Advanced approaches to linguistic Data Mining*. Springer-Verlag, 2004.

[15] A. Orriols-Puig, E. Bernadó-Mansilla: Evolutionary rule-based systems for imbalanced datasets, *Soft Computing* 13 (3), pp. 213–225, 2009.

[16] J. Quinlan: *C4.5: Programs for Machine Learning*. Morgan Kauffman, 1993.

[17] D. Sheskin: *Handbook of parametric and nonparametric statistical procedures*. Chapman & Hall/CRC, 2006.

[18] Y. Sun, A. K. C. Wong, M. S. Kamel: Classification of imbalanced data: A review, *International Journal of Pattern Recognition and Artificial Intelligence* 23 (4), pp. 687–719, 2009.