XIX Congreso Español sobre Tecnologías y Lógica Fuzzy (XIX ESTYLF)

ESTYLF 3:
SISTEMAS BASADOS EN REGLAS DIFUSAS

Relevancia, Precisión e Interpretabilidad en Sistemas Basados en Reglas Difusas

Marta Galende
CARTIF Centro Tecnológico
Boecillo, Valladolid, España
margal@cartif.es

M. Isabel Rey
INDOMAUT S.L.
Valladolid, España
i.rey@indomaut.com

M.J. Fuente
Dpto. Ingeniería de Sistemas y Automática
Universidad de Valladolid
Valladolid, España
mjfuente@eii.uva.es

G.I. Sainz-Palmero
Dpto. Ingeniería de Sistemas y Automática
Universidad de Valladolid
Valladolid, España
gresai@eii.uva.es

***Resumen*—Los Sistemas Basados en Reglas Difusas (SBRD) permiten modelar problemas reales, manejando no solo la no precisión en lo referente al conocimiento a manejar, sino también la precisión con la que modelan el problema y la capacidad para "interpretar" su comportamiento. Por otro lado aparece el concepto de relevancia de las reglas difusas del SBRD, parece que lo idóneo es que sus reglas sean relevantes, o lo más relevantes posible. Relevancia, precisión e interpretabilidad son las tres métricas que se consideran, y analizan, en este trabajo para conseguir SBRDs que presenten buenas prestaciones respecto a estos tres objetivos, centrándonos finalmente en cómo es la relevancia de las reglas de los SBRD en el equilibrio precisión-interpretabilidad.**

Basándose en Transformaciones Ortogonales (SVD, PQR, OLS) es posible estimar la *Relevancia* de una regla difusa, y analizar como influye dicho valor en la búsqueda del equilibrio precisión-interpretabilidad en un SBRD. Usando nueve conjuntos de datos del repositorio KEEL, dos algoritmos de modelado: uno aproximativo (FasArt) y otro lingüístico (NefProx), y siguiendo una estrategia de optimización multi-objetivo (SPEA2), se presentan a continuación los resultados obtenidos que muestran el concepto de relevancia como un factor importante, y contradictorio, a tener en cuenta a la hora de generar SBRD.

***Index Terms*—Relevancia, Transformación Ortogonal, Precisión-Interpretabilidad, Sistemas Basados en Reglas Difusas**

I. Introducción

Los Sistemas Difusos Basados en Reglas (SBRD) son una forma muy habitual de aplicar la lógica difusa tanto en áreas de investigación como para abordar problemas del mundo real [1]–[3]. Existen múltiples aproximaciones para generar estos SBRDs [4], [5], y una de las cuestiones a afrontar es cómo evaluar sus prestaciones: en este punto, las metodologías lingüísticas y precisas para la generación de SBRD [6]–[8] son las más usadas y debatidas.

Este aspecto ha sido ampliamente discutido: la Precisión es un aspecto básico para un modelo, difuso o no, pero además cuando la Lógica Difusa es utilizada aparecen otros puntos de vista, como la capacidad para expresar, representar y explicar conocimiento en términos lingüísticos, similares a los usados por las personas. Este punto de vista es intrínseco a los principios de la Lógica difusa.

¿Cómo estimar la prestación de un SBRD? este es un aspecto básico tanto en desarrollos teóricos, como en aplicaciones prácticas, donde un modelo basado en SBRD tiene que ser perfectamente claro y definido. Por tanto, aprovechando las prestaciones de la Lógica Difusa, la obtención de un SBRD con una adecuada Interpretabilidad es de gran interés, pero siempre que el modelo implementado mediante un SBRD sea suficientemente preciso, sin este nivel de precisión cualquier modelo no es útil. Así, buscamos alcanzar modelos usando SBRD que puedan aunar Precisión e Interpretabilidad. Esto es lo que se conoce habitualmente como equilibrio Precisión-Interpretabilidad, aspectos que habitualmente se consideran mutuamente contraproducentes y contradictorios [7], [9]. ¿Cómo conseguirlo?, esto es una cuestión abierta, y múltiples aproximaciones y propuestas se pueden encontrar en la bibliografía especializada, como los sistemas genéticos difusos y la selección de reglas [6], [10]–[14], considerando distintas métricas de complejidad o semántica [15]–[18], etc.

En este ámbito, la propuesta aquí planteada se centra en el equilibrio Precisión - Interpretabilidad considerando una metodología basada en la selección de reglas, eligiendo estas reglas de acuerdo a las métricas de Precisión, Interpretabilidad, y añadiendo Relevancia. En la mayoría de los casos, la Relevancia se ha empleado para reducir la complejidad de los SBRD [19], [20], que está relacionada con la Interpretabilidad, pero aquí la Relevancia es un factor a considerar tan importante como la Precisión o la Interpretabilidad.

El resto del artículo se organiza como sigue: la sección II presenta el concepto de relevancia, su definición y posibles métricas a utilizar. La sección III muestra el problema de la búsqueda de equilibrio precisión-interpretabilidad en los SBRD y la sección IV presenta la metodología propuesta. La sección V muestra los resultados obtenidos en la parte expe-

Trabajo realizado gracias al soporte del Ministerio de Economía y Competitividad español y al European Regional Development Fund (FEDER) a través del proyecto no. DPI2015-67341-C2-2-R.

rimental y finalmente la sección VI presenta las principales conclusiones obtenidas y los futuros trabajos a realizar.

II. Relevancia

La relevancia de las reglas difusas se ha definido habitualmente mediante transformaciones ortogonales [21] como *Single Value Descomposition* (SVD), *Pivoting-QR* (PQR) o *Orthogonal Least Square* (OLS) aplicada sobre la matriz de activación de las reglas difusas, de forma similar a otras áreas como selección y extracción de características en reconocimiento de patrones. En SVD y PQR solo se tiene en cuenta la activación de los antecedentes de las reglas, mientras que en OLS también se considera el consecuente de las mismas. Como resultado de estas descomposiciones se obtienen estimaciones de la relevancia en forma de valores singulares, R-Values o varianzas de cada regla difusa para el problema dado [20], [22], [23].

III. Equilibrio Precisión-Interpretabilidad

La prestación de un SBRD se puede analizar bajo un doble punto de vista:

- **Precisión**: o capacidad para emular el comportamiento del sistema modelado. Métrica: $Error$ [24].
- **Interpretabilidad**: capacidad para "explicar" su comportamiento o de la realidad que modelan. Métrica: existes diversas opciones [15]–[18].

En el proceso de modelado se necesita conseguir la suficiente precisión para que el modelo resultante sea útil, pero en el caso de SBRD además se puede optar a alcanzar un grado de interpretabilidad del mismo. Esto es el equilibrio precisión-interpretabilidad, que es el objetivo de este trabajo.

IV. Precisión, Interpretabilidad y Relevancia en SBRD

Siguiendo la metodología resumida en la Fig. 1 [10] se busca una mejora en la prestación del SBRD a través de un post-procesamiento basado en selección de sus reglas: la idea básica es buscar un equilibrio entre la precisión y la interpretabilidad considerando las reglas más relevantes del SBRD. En esta ocasión se aborda la estrategia de descartar las reglas menos relevantes durante el proceso.

Fase 1: generación de SBRDs de diferente naturaleza [25] en base a conjuntos de datos.

- Aproximativos: FasArt [26].
- Lingüísticos: NefProx [27].

Mediante las conocidas transformaciones ortogonales: SVD, PQR y OLS, se estima la relevancia de cada una de las reglas de los SBRD generados.

Fase 2: proceso de selección de reglas del SBRD que busca mejorar su Precisión-Interpretabilidad-Relevancia. Esto se realiza mediante el Algoritmo Evolutivo Multi-Objetivo (AEMO) SPEA2 [28].

Para esto los objetivos comprometidos son:

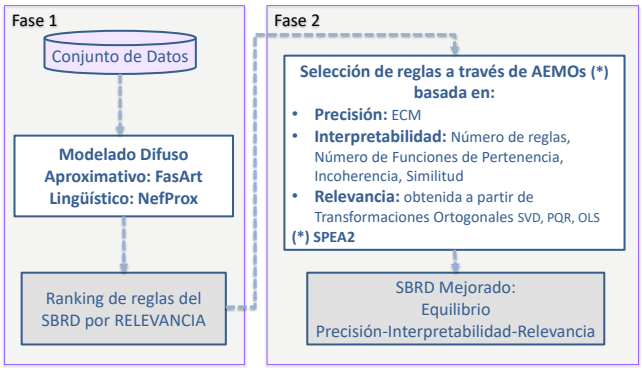

Figura 1. Mejora del equilibrio precisión-interpretabilidad de un SBRD mediante precisión, interpretabilidad y relevancia.

- Maximizar la precisión del sistema, minimizando el error cuadrático medio (Ec. 1).

$$ECM = \frac{1}{|N|}\sum_{i=1}^{|N|}(F(x_i) - y_i)^2 \tag{1}$$

- Maximizar la interpretabilidad del sistema, minimizando el valor de uno de los siguientes índices:
  - Número de reglas (Ec. 2).

$$NR = Numero\ de\ reglas \tag{2}$$

  - Número de funciones de pertenencia (Ec. 3).

$$NMF = Numero\ de\ funciones\ de\ pertenencia \tag{3}$$

  - Incoherencia de la base de reglas (Ec. 4).

$$\begin{aligned} &Inc = \frac{|(S_{kA}(R_i,R_j)>(1-\beta_I)\ \ AND\ \ S_{kC}(R_i,R_j)<\beta_I)|}{(NumerodeReglas-1)!} \\ &\forall 1 \leq i < j \leq NumerodeReglas \\ &\forall 1 \leq kA \leq NumeroDeAntecedentes \\ &\forall 1 \leq kC \leq NumeroDeConsecuentes \end{aligned} \tag{4}$$

  - Similitud de la base de datos (Ec. 5).

$$\begin{aligned} &Sim = F_{kA}(F_{l,m}(S(MF_{kA.l}, MF_{kA.m}))) \\ &F \Rightarrow MediaAritmetica \\ &\forall 1 \leq l < m \leq NumeroDeFuncionesDePertenencia \\ &\forall 1 \leq kA \leq NumeroDeAntecedentes \end{aligned} \tag{5}$$

- Maximizar la relevancia de la relevancia del sistema, en este trabajo descartando las reglas menos relevantes (minimizando Ec. 6).

$$Rel_{RB} = \sqrt[k]{\prod_{i=1}^{k}(1 - Relevancia_{Regla_i})} \tag{6}$$

Esta metodología y las métricas utilizadas en detalle se pueden consultar en [10].

V. Trabajo experimental

El impacto de relevancia en la búsqueda del equilibrio precisión-interpretabilidad se han comprobado mediante la experimentación que se detalla a continuación.

En [10] se muestran la estrategia de preservación de las reglas más relevantes, aquí se muestra, y analiza, la estrategia basada en descartar las reglas menos relevantes (Ec. 6).

V-A. Casos de estudio

Se han utilizado nueve conjuntos de datos reales del proyecto KEEL [29] y una metodología de validación cruzada 5-fold. La Tabla I muestra las principales características de dichos conjuntos de datos.

Cuadro I
Conjuntos de datos considerados

Conjunto de datos	Nombre	Variables	Registros
Plastic Strength	PLA	3	1650
Quake	QUA	4	2178
Electrical Maintenance	ELE	5	1056
Abalone	ABA	9	4177
Stock prices	STP	10	950
Weather Izmir	WIZ	10	1461
Weather Ankara	WAN	10	1609
Mortgage	MOR	16	1049
Treasury	TRE	16	1049

El resto de opciones utilizadas en este trabajo han sido:

- Métrica de relevancia: Rel_{RB} descartando las reglas menos relevantes.
- Transformaciones Ortogonales: SVD, PQR y OLS.
- Algoritmos de modelado: FasArt como algoritmos aproximativo y NefProx como lingüístico, parametrizados según [11].
- Métricas de interpretabilidad: NR, NMF, Inc y Sim siguiendo las recomendaciones de [9], [15].
- Algoritmo evolutivo multi-objetivo: $SPEA2$ utilizando codificación binaria, cruce HUX [30], mutación clásica [31] con probabilidad 0,2, mecanismo de prevención de incesto basado en los conceptos de CHC [30], operador de reinicialización y una población de padres que se va reduciendo progresivamente [32].

En cuanto a los planos de proyección de las soluciones finales [12], [33], [34] se presentan los resultados en el plano precisión-interpretabilidad, considerando el SBRD más preciso, el SBRD más interpretable, y SBRD en la mediana precisión-interpretabilidad.

V-B. Análisis de resultados

La Tabla II muestra los resultados medios obtenidos para todos los conjuntos de datos cuando la relevancia se estima mediante la transformación OLS, analizando el plano precisión-interpretabilidad. Específicamente se muestran las siguientes métricas: NR, NFP, Inc y Sim como medidas de interpretabilidad; ECM en entrenamiento E_{tra} y test E_{tst}; número de reglas NR; métrica Rel_{RB} de relevancia optimizada y relevancia media Rel_{SD} de las reglas que permanecen en el sistema tras el proceso de optimización.

La Tabla III muestra los mismos resultados para SVD y la Tabla IV para PQR.

Realizando un análisis conjunto de las soluciones alcanzadas al aplicar los tres métodos de transformación ortogonal se puede decir que:

Modelado Aproximativo.

1. **Solución más precisa**: los valores medios de la *Interpretabilidad*, el *error*, el *número de reglas* y la *Relevancia* mejoran en todos los casos, hasta alcanzar una mejoría del 26,04 % en el primer caso, del 17,03 % en el segundo caso, del 26,32 % en el caso del número de reglas, y del 9,73 % en la Relevancia.
2. **Solución en la mediana Precisión-Interpretabilidad**: la *Interpretabilidad* ha mejorado hasta un 32,88 %, el *error* se reduce hasta un 16,36 %, el *número de reglas* también mejora siempre hasta el 34,99 %, y la *Relevancia* mejora hasta el 17,09 %.
3. **Solución más interpretable**: todas las métricas estudiadas han mejorado: la *Interpretabilidad* ha mejorado hasta un 51,67 %, el *error* se ha reducido hasta el 14,56 %, el *número de reglas* ha mejorado hasta un 43,53 %, y la *Relevancia* hasta el 30,04 %.

Modelado Lingüístico.

1. **Solución más precisa**: la *Interpretabilidad* ha mejorado hasta el 47,26 %, el *error* es reducido hasta el 36,36 %, el *número de reglas* mejora hasta un 47,32 %, y la *Relevancia* siempre mejora hasta el 24,91 %.
2. **Solución en la mediana Precisión-Interpretabilidad**: la *Interpretabilidad* se ve que mejora en casi todos los casos hasta conseguir un porcentaje de mejora del 51,28 %. Lo mismo sucede con el resto de métricas estudiadas que mejoran en todos los casos hasta alcanzar un porcentaje de mejora del 35,85 % en el caso del *error*, del 51,28 % en el caso del *número de reglas*, y del 27,18 % cuando lo analizado es la *Relevancia*.
3. **Solución más interpretable**: todas las métricas mejoran: la *Interpretabilidad* ha mejorado hasta un 62,74 %, el *error* hasta un 34,29 %, el *número de reglas* hasta un 56,48 %, y la *Relevancia* hasta alcanzar una importante mejoría del 47,98 %.

En cuanto al análisis de resultados en el resto de planos la Tabla V muestra un resumen de los valores obtenidos cuando se utiliza la transformación ortogonal OLS. Aquí las relaciones son más complejas y están en consonancia con la discusión en Sección V-C.

V-C. Reglas relevantes y Precisión-Interpretabilidad

Un aspecto interesante a plantearse es ver y analizar cómo son las reglas del SBRD en el equilibrio precisión-interpretabilidad. En la metodología llevada a cabo, las reglas poco relevantes son las candidatas a no ser seleccionadas para el SBRD final. En la Tabla VI puede observarse la distribución de las reglas de los SBRD iniciales de acuerdo a su relevancia (normalizada en $[0, 1]$ y clasificado en cuatro cuartos): en general las relevancia de las reglas presente en los modelos

Cuadro II
Valores Medios (%) de SBRDs Mejorados: Rel_{RB}, OLS, Plano Precisión-Interpretabilidad

	Mejor *Pre*						Mediana *Pre-Inter*						Mejor *Inter*					
Inter	*Inter*	E_{tra}	E_{tst}	NR	Rel_{RB}	Rel_{SD}	*Inter*	E_{tra}	E_{tst}	NR	Rel_{RB}	Rel_{SD}	*Inter*	E_{tra}	E_{tst}	NR	Rel_{RB}	Rel_{SD}
FasArt																		
NR	25,88	19,64	16,61	25,88	0,13	8,01	30,60	18,67	15,79	30,60	0,27	14,00	38,12	7,68	5,96	38,12	0,57	25,65
NFP	25,14	19,63	16,71	26,26	0,13	8,40	31,57	17,47	14,83	32,90	0,34	17,09	40,82	−9,55	−7,28	41,79	0,68	30,04
Inc	−0,89	19,62	16,66	25,98	0,17	9,73	13,37	19,28	16,36	27,18	0,21	11,87	51,16	17,50	14,56	29,36	0,28	15,47
Sim	2,44	19,67	16,95	25,94	0,14	8,12	8,76	9,23	7,08	34,81	0,36	16,96	14,35	−39,77	−36,14	42,29	0,53	23,19
NefProx																		
NR	46,98	37,44	35,96	46,98	0,26	17,76	50,56	37,39	35,85	50,56	0,36	24,78	55,14	32,02	30,31	55,14	0,57	39,36
NFP	5,07	37,41	35,84	47,10	0,31	21,90	7,33	36,40	34,86	48,80	0,38	26,63	10,30	27,30	25,96	51,53	0,56	38,59
Inc	9,97	38,05	36,29	47,32	0,36	24,91	41,22	36,06	34,25	48,30	0,44	27,18	62,67	21,52	20,40	53,18	0,73	47,98
Sim	−4,71	37,36	35,85	46,68	0,30	20,68	−4,60	37,25	35,77	46,72	0,30	20,78	−3,87	35,53	34,25	47,30	0,37	25,67

Cuadro III
Valores Medios (%) de SBRDs Mejorados: Rel_{RB}, SVD, Plano Precisión-Interpretabilidad

	Mejor *Pre*						Mediana *Pre-Inter*						Mejor *Inter*					
Inter	*Inter*	E_{tra}	E_{tst}	NR	Rel_{RB}	Rel_{SD}	*Inter*	E_{tra}	E_{tst}	NR	Rel_{RB}	Rel_{SD}	*Inter*	E_{tra}	E_{tst}	NR	Rel_{RB}	Rel_{SD}
FasArt																		
NR	25,83	19,63	16,68	25,83	0,04	2,05	30,72	17,91	15,10	30,72	0,10	4,61	38,47	2,11	0,64	38,47	0,22	9,71
NFP	24,94	19,67	16,74	26,04	0,04	2,18	31,18	17,79	14,83	32,62	0,13	6,11	40,81	−28,57	−23,94	41,80	0,26	11,14
Inc	0,64	19,64	16,49	25,75	0,05	2,65	14,72	19,25	16,12	26,97	0,05	3,17	51,67	17,19	13,98	29,11	0,08	4,59
Sim	2,22	19,58	16,69	25,83	0,05	2,51	8,97	9,55	6,15	34,99	0,13	5,43	14,69	−42,39	−38,72	42,80	0,22	8,31
NefProx																		
NR	46,59	37,51	35,87	46,59	0,06	6,63	50,00	37,33	35,76	50,00	0,13	11,63	55,31	33,39	32,08	55,31	0,28	21,54
NFP	5,19	37,56	35,90	46,53	0,07	7,27	7,51	35,52	33,82	48,56	0,12	10,61	10,85	25,80	24,06	51,19	0,20	16,96
Inc	11,49	38,18	36,36	46,01	0,09	9,36	40,95	36,34	34,49	47,84	0,13	13,61	62,74	21,75	20,25	52,16	0,24	21,55
Sim	−4,86	37,54	35,91	46,28	0,06	6,76	−4,49	37,08	35,68	46,40	0,07	7,18	−3,86	35,59	34,29	46,90	0,09	8,66

Cuadro IV
Valores Medios (%) de SBRDs Mejorados:Rel_{RB}, PQR, Plano Precisión-Interpretabilidad

	Mejor *Pre*						Mediana *Pre-Inter*						Mejor *Inter*					
Inter	*Inter*	E_{tra}	E_{tst}	NR	Rel_{RB}	Rel_{SD}	*Inter*	E_{tra}	E_{tst}	NR	Rel_{RB}	Rel_{SD}	*Inter*	E_{tra}	E_{tst}	NR	Rel_{RB}	Rel_{SD}
FasArt																		
NR	26,04	19,63	16,80	26,04	0,04	2,62	31,77	17,40	14,91	31,77	0,09	5,08	39,96	−4,56	−5,53	39,96	0,20	9,89
NFP	25,26	19,65	16,75	26,32	0,04	2,47	32,88	16,06	13,25	34,01	0,11	5,77	42,70	−35,05	−29,01	43,53	0,24	11,19
Inc	−0,96	19,62	16,82	26,08	0,04	2,82	15,83	19,19	16,25	27,09	0,05	3,30	49,22	17,27	14,31	29,18	0,07	4,73
Sim	2,42	19,62	17,03	25,92	0,04	2,67	8,91	9,83	7,63	34,48	0,11	5,02	14,54	−40,03	−37,78	42,69	0,21	8,63
NefProx																		
NR	47,26	37,20	35,61	47,26	0,05	2,48	51,28	37,10	35,56	51,28	0,09	5,13	56,48	22,64	21,31	56,48	0,18	12,42
NFP	5,20	37,34	35,73	47,24	0,06	2,71	7,38	36,03	34,46	48,97	0,08	4,08	10,30	27,15	25,52	51,72	0,12	6,92
Inc	14,02	37,94	36,18	45,25	0,06	3,31	41,87	35,86	34,06	46,82	0,08	5,71	62,62	19,08	17,95	50,26	0,13	9,80
Sim	−4,77	37,36	35,75	46,73	0,05	2,53	−4,68	37,15	35,52	46,80	0,05	2,60	−4,15	35,06	33,71	47,32	0,06	3,16

Cuadro V
Valores Medios (%) de SBRDs Mejorados: Rel_{RB}, OLS, Planos Pre-Rel_{RB} (PR), Rel_{RB}-$Inter$ (RI)

Modelo	**Inter**	PR	Mejor *Pre*						Mediana *Pre*-Rel_{RB}						Mejor Rel_{RB}					
		RI	Mejor Rel_{RB}						Mediana Rel_{RB}-*Inter*						Mejor *Inter*					
		Plano	*Inter*	E_{tra}	E_{tst}	NR	Rel_{RB}	Rel_{SD}	*Inter*	E_{tra}	E_{tst}	NR	Rel_{RB}	Rel_{SD}	*Inter*	E_{tra}	E_{tst}	NR	Rel_{RB}	Rel_{SD}
FasArt	NR	PR	25,92	19,63	16,70	25,92	0,07	4,23	30,87	18,00	15,03	30,87	0,18	8,95	37,90	3,44	1,90	37,90	0,39	17,01
		RI	37,90	3,43	1,89	37,90	0,39	17,01	38,05	2,87	1,26	38,05	0,39	16,93	38,85	−1,29	−2,56	38,85	0,36	15,98
	NFP	PR	25,11	19,65	16,73	26,21	0,07	4,35	31,12	17,29	14,45	32,76	0,23	11,00	39,41	−2,35	−3,28	41,17	0,48	21,05
		RI	39,42	−2,37	−3,30	41,17	0,48	21,05	40,01	−5,71	−6,08	41,45	0,47	20,54	41,44	−27,04	−22,76	42,43	0,42	18,28
	Inc	PR	−0,64	19,63	16,66	26,03	0,09	5,18	0,95	18,33	15,52	31,28	0,22	11,11	32,29	7,88	5,78	37,82	0,43	19,57
		RI	32,29	7,88	5,78	37,82	0,43	19,57	39,94	8,01	5,78	37,39	0,42	18,91	50,68	7,78	5,62	37,32	0,41	18,02
	Sim	PR	2,36	19,62	16,89	25,89	0,08	4,43	3,50	13,81	11,91	35,56	0,34	16,83	7,60	−21,72	−21,45	45,50	0,66	30,42
		RI	2,31	19,62	16,89	25,94	0,08	4,50	11,35	−28,18	−27,11	44,02	0,54	24,85	14,53	−40,79	−37,61	42,63	0,32	13,40
NefProx	NR	PR	46,91	37,38	35,81	46,91	0,13	9,17	50,51	37,21	35,67	50,51	0,22	16,51	55,21	28,38	26,81	55,21	0,39	28,03
		RI	55,21	28,37	26,79	55,21	0,39	28,03	55,30	28,37	26,79	55,30	0,39	27,89	55,65	27,42	25,96	55,65	0,36	25,66
	NFP	PR	5,15	37,44	35,83	46,98	0,15	10,65	5,76	37,14	35,51	51,96	0,30	21,85	8,94	21,86	20,05	57,37	0,56	39,98
		RI	8,96	21,83	20,01	57,37	0,56	39,98	9,35	22,94	21,34	57,35	0,55	39,33	10,49	21,99	20,41	57,08	0,51	36,48
	Inc	PR	11,68	38,05	36,28	46,24	0,17	12,61	10,05	35,73	33,95	52,87	0,54	40,09	37,54	6,28	5,05	58,02	0,91	66,73
		RI	37,57	6,24	5,00	58,02	0,91	66,73	47,62	4,21	3,07	57,20	0,85	61,10	62,68	0,61	−0,51	56,99	0,75	51,88
	Sim	PR	−4,78	37,42	35,83	46,56	0,14	9,99	−5,40	37,22	35,67	50,35	0,25	18,68	−6,11	26,97	25,21	54,82	0,45	31,98
		RI	−4,79	37,42	35,84	46,59	0,14	10,05	−5,45	27,57	25,92	54,73	0,43	31,14	−3,96	29,03	27,62	54,46	0,41	29,62

difusos es baja, $60\,\% - 90\,\%$, si se considera las reglas con relevancia Baja y MediaBaja la cifra está entre el $88\,\%$ y casi el $100\,\%$. En el SBRD lingüístico las cifras son algo mayores que en el aproximativo, lo cual podría ser algo sorprendente.

Estos resultados implican que en los SBRD resultantes de esta metodología, que presentarán una mejora en el equilibrio precisión-interpretabilidad, las reglas con baja relevancia van a tener, y tienen, un destacado impacto tanto cualitativamente, como cuantitativamente. Por otro lado, aproximaciones muy conocidas para la simplificación de modelos difusos como [23], [35], [36] basadas en las reglas con relevancia "alta", o punto de ruptura en los valores de relevancia, parecen ser de difícil aplicación real si además se analizan las magnitudes de las relevancias.

En la Tabla VII se puede observar la distribución de reglas por su relevancia en el SBRD optimizado para los modelos aproximativo y lingüístico, considerando las 4 métricas de interpretabilidad, en tres puntos de optimización: mejor precisión, mejor interpretabilidad y mediana precisión-interpretabilidad. También aparece el $\%$ de las reglas del modelo inicial, de acuerdo a su relevancia, que NO han sido seleccionadas en el SBRD final. Todo esto considerando la transformación OLS para la estimación de las relevancias.

En general, estos SBRD han aumentado su ratio de reglas consideradas de baja relevancia, ahora es mayor: $\geq 92\,\%$, siendo algo mayor para el SBRD lingüístico. Esto es coherente con la presencia de reglas poco relevantes en el SBRD de partida, incluso teniendo en cuenta que entre el $40 - 50\,\%$ de ellas han sido descartadas. Más relevante en este descarte en lo referente a reglas con Alta, MediaAlta, MediaBaja relevancia: una parte importante de estas reglas, incluidas las de alta relevancia, no son, sorprendentemente, mantenidas en el SBRD. Este aspecto resulta contraproducente: reglas con la relevancia alta no son seleccionadas en $38 - 10\,\%$, alrededor del $60\,\%$ MediaAlta, etc... pero también, como se observa en las Tablas II, III y IV la relevancia de las reglas de los SBRD finalmente obtenidos ha aumentado en todos los modelos y métricas de interpretabilidad.

Aquí, de acuerdo a los resultados, se ha comprobado que la relevancia de las reglas aporta en el proceso de mejora de la precisión e interpretación preservando en el SBRD las reglas más relevantes compatibles con esto. Considerando todo lo anterior, hay una cuestión básica a afrontar: la necesidad comprender el por qué de la no selección de reglas estimadas con relevancia alta, o incluso más alta que la mayoría de aquellas que si se mantienen en el modelo. Esto implica una caracterización de estas reglas no seleccionadas a pesar de su relevancia importante para dar respuesta a preguntas como: ¿Cómo son? ¿Por qué no son seleccionadas?, así poder generar algún criterio que permita estimar o no una regla para un SBRD.

VI. Conclusiones

En este trabajo se presentan los resultados obtenidos a partir de una metodología mostrada en [10] pero introduciendo el criterio de descartar las reglas difusas menos relevantes durante el proceso. Esto se ha traducido en la generación SBRD que, en general, mejoran sus prestaciones de precisión, interpretabilidad y relevancia en los modelos considerados: Mejor Precisión, Mejor Interpretabilidad y Mediana Precisión-Interpretabilidad. Analizando las reglas que son incluidas en esos modelos, se observa el hecho sorprendente de que reglas con una relevancia estimada alta, o al menos más alta que la mayoría de las seleccionadas, no son mantenidas en el SBRD. Conocer cómo son estas reglas, su caracterización y la razón de su no permanencia en el SBRD es una de las tareas actualmente en desarrollo. Por otro lado, estos modelos están formados de forma abrumadora por reglas consideradas poco relevantes, analizar que características tienen, y por qué son elegidas durante el proceso en contraposición a las relevantes

Cuadro VI
Distribución en % de reglas por relevancia en el SBRD inicial

	SBRD	R_Baja	R_MediaBaja	R_MediaAlta	R_Alta
SVD	FasArt	70.24	23.35	5.36	1.06
	NefProx	88.60	6.53	3.40	1.47
PQR	FasArt	60.54	27.97	9.25	2.24
	NefProx	84.57	9.08	2.00	4.35
OLS	FasArt	94.55	4.45	0.91	0.09
	NefProx	97.54	1.98	0.36	0.12

Cuadro VII
Distribución en % de las reglas por relevancia del SBRD optimizado considerando la transformación ortogonal OLS: distribución de reglas por su relevancia, y reglas NO seleccionadas del modelo inicial

Inter	**Modelo**	**SBRD**	**Reglas**	R_Baja	R_MediaBaja	R_MediaAlta	R_Alta
NR	Mejor *Inter*	FasArt	SBRD	92.50	6.66	0.77	0.07
			NoSelec.	39.36	11.12	33.25	38.89
		NefProx	SBRD	96.35	3.05	0.39	0.21
			NoSelec.	55.50	27.02	61.67	11.67
	Mejor *Pre*	FasArt	SBRD	93.86	5.40	0.68	0.06
			NoSelec.	26.42	9.75	30.75	38.89
		NefProx	SBRD	96.86	2.65	0.30	0.18
			NoSelec.	47.23	25.12	61.67	11.67
	Mediana *Pre-Inter*	FasArt	SBRD	93.48	5.75	0.71	0.07
			NoSelec.	31.36	10.92	30.75	38.89
		NefProx	SBRD	96.69	2.77	0.34	0.19
			NoSelec.	50.85	26.31	61.67	11.67
NFP	Mejor *Inter*	FasArt	SBRD	92.06	7.09	0.77	0.08
			NoSelec.	43.21	12.91	31.03	41.67
		NefProx	SBRD	96.36	3.04	0.40	0.19
			NoSelec.	51.89	24.05	61.67	10.00
	Mejor *Pre*	FasArt	SBRD	93.87	5.41	0.66	0.07
			NoSelec.	26.81	9.94	29.36	41.67
		NefProx	SBRD	96.82	2.67	0.31	0.19
			NoSelec.	47.38	23.81	61.67	10.00
	Mediana *Pre-Inter*	FasArt	SBRD	93.29	5.92	0.72	0.07
			NoSelec.	33.78	11.49	29.36	41.67
		NefProx	SBRD	96.67	2.79	0.35	0.19
			NoSelec.	49.12	23.81	61.67	10.00
INC	Mejor *Inter*	FasArt	SBRD	93.28	5.70	0.95	0.07
			NoSelec.	30.21	10.17	31.39	33.33
		NefProx	SBRD	96.23	3.05	0.46	0.26
			NoSelec.	53.48	31.43	65.49	11.67
	Mejor *Pre*	FasArt	SBRD	93.68	5.41	0.85	0.07
			NoSelec.	26.61	9.67	31.81	38.89
		NefProx	SBRD	96.80	2.64	0.35	0.20
			NoSelec.	47.53	32.86	61.67	11.67
	Mediana *Pre-Inter*	FasArt	SBRD	93.54	5.50	0.89	0.07
			NoSelec.	27.87	9.87	31.39	33.33
		NefProx	SBRD	96.83	2.58	0.35	0.24
			NoSelec.	48.51	33.33	62.71	10.00
SIM	Mejor *Inter*	FasArt	SBRD	92.55	6.41	0.93	0.10
			NoSelec.	43.37	18.89	35.97	33.33
		NefProx	SBRD	96.69	2.78	0.34	0.19
			NoSelec.	47.59	24.17	61.87	10.00
	Mejor *Pre*	FasArt	SBRD	93.80	5.38	0.75	0.06
			NoSelec.	26.49	10.02	32.11	47.22
		NefProx	SBRD	96.85	2.66	0.30	0.19
			NoSelec.	46.94	24.40	62.71	10.00
	Mediana *Pre-Inter*	FasArt	SBRD	93.21	5.86	0.85	0.08
			NoSelec.	35.69	14.12	33.19	38.89
		NefProx	SBRD	96.85	2.66	0.30	0.19
			NoSelec.	46.99	24.40	62.71	10.00

pero no que son eliminadas en el SBRD final, es la línea de trabajo a muy corto plazo.

Referencias

[1] J. Kacprzyk, W. Pedrycz, Springer Handbook of Computational Intelligence, Springer, 2015.

[2] A. Konar, Computational Intelligence: Principles, techniques and applications, Springer-Verlag, Berlin, 2005.

[3] F. O. Karray, C. d. De Silva, Soft Computing and Intelligent Systems Design. Theory, Tools and Applications, Addison Wesley, 2004.

[4] J. Alcalá-Fdez, J. Alonso, A survey of fuzzy systems software: taxonomy, current research trends and prospects, IEEE Transactions on Fuzzy Systems 24 (1) (2016) 40 – 56.

[5] L. Magdalena, Fuzzy Rule-Based Systems, in: Springer Handbook of Computational Intelligence, Springer, 2015, pp. 203 – 218.

[6] A. Fernández, V. López, M. J. del Jesus, F. Herrera, Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges, Knowledge-Based Systems 80 (2015) 109–121.

[7] J. Casillas, O. Cordón, F. Herrera, L. Magdalena, Accuracy improvements to find the balance interpretability-accuracy in fuzzy modeling: An overview, in: J. Casillas, O. Cordón, F. Herrera, L. Magdalena (Eds.), Accuracy Improvements in Linguistic Fuzzy Modelling, Vol. 129 of Studies in Fuzziness and SoftComputing, Springer-Verlag, Berlin Heildelberg, 2003, pp. 3–24.

[8] J. Casillas, O. Cordón, F. Herrera, L. Magdalena, Interpretability improvements to find the balance interpretability-accuracy in fuzzy modeling: An overview, in: J. Casillas, O. Cordón, F. Herrera, L. Magdalena (Eds.), Interpretability Issues in Fuzzy Modelling, Vol. 128 of Studies in Fuzziness and SoftComputing, Springer-Verlag, Berlin Heildelberg, 2003, pp. 3–22.

[9] J. M. Alonso, C. Castiello, C. Mencar, Interpretability of fuzzy systems: Current research trends and prospects, in: Springer Handbook of Computational Intelligence, Springer, 2015, pp. 219–237.

[10] M. I. Rey, M. Galende, M. J. Fuente, G. I. S. Palmero, Multi-objective based fuzzy rule based systems (frbss) for trade-off improvement in accuracy and interpretability: A rule relevance point of view, Knowledge-Based Systems 127 (2017) 67–84. doi:10.1016/j.knosys.2016.12.028.
URL https://doi.org/10.1016/j.knosys.2016.12.028

[11] M. Galende, M. J. Gacto, G. Sainz, R. Alcalá, Comparison and design of interpretable linguistic vs. scatter FRBSs: Gm3m generalization and new rule meaning index for global assessment and local pseudo-linguistic representation, Information Sciences 282 (2014) 190–213.

[12] M. Galende, G. I. Sainz, M. J. Fuente, Complexity reduction and interpretability improvement for fuzzy rule systems based on simple interpretability measures and indices by bi-objective evolutionary rule selection, Soft Computing 16 (3) (2012) 451 – 470.

[13] M. Fazzolari, R. Alcalá, Y. Nojima, H. Ishibuchi, F. Herrera, A review of the application of multiobjective evolutionary fuzzy systems: Current status and further directions, IEEE Transactions on Fuzzy Systems 21 (2013) 45 – 65.

[14] O. Cordón, A historical review of evolutionary learning methods for mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems, International Journal of Approximate Reasoning 52 (2011) 894 – 913.

[15] M. J. Gacto, R. Alcalá, F. Herrera, Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures, Information Sciences 181 (2011) 4340 – 4360.

[16] J. M. Alonso, L. Magdalena, G. González-Rodríguez, Looking for a good fuzzy system interpretability index: An experimental approach, International Journal of Approximate Reasoning 51 (1) (2009) 115 – 134.

[17] C. Mencar, A. Fanelli, Interpretability constraints for fuzzy information granulation, Information Sciences 178 (24) (2008) 4585 – 4618.

[18] S.-M. Zhou, J. Q. Gan, Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modelling, Fuzzy Sets and Systems 159 (2008) 3091 – 3131.

[19] S.-M. Zhou, J. M. Garibaldi, R. I. John, F. Chiclana, On constructing parsimonious type-2 fuzzy logic systems via influential rule selection, IEEE Transactions on Fuzzy Systems 17 (3) (2009) 654 – 667.

[20] M. Setnes, Simplification and reduction of fuzzy rules, in: J. Casillas, O. Cordón, F. Herrera, L. Magdalena (Eds.), Interpretability Issues in Fuzzy Modelling, Vol. 128 of Studies in Fuzziness and SoftComputing, Springer-Verlag, Berlin Heildelberg, 2003, pp. 278–302.

[21] G. H. Golub, C. F. Van Loan, Matrix computations, Vol. 3, JHU Press, 2012.

[22] S. Destercke, S. Guillaume, B. Charnomordic, Building an interpretable fuzzy rule base from data using orthogonal least squares - Application to a depollution problem, Fuzzy Sets and Systems 158 (18) (2007) 2078 – 2094.

[23] J. Yen, L. Wang, Simplifying fuzzy rule-based models using orthogonal transformation methods, IEEE Transactions on Systems, Man and Cybernetics. Part B: Cybernetics 29 (1) (1999) 13–24.

[24] I. H. Witten, E. Frank, M. A. Hall, Data Mining: Practical machine learning tools and techniques, 3rd Edition, Morgan Kaufmann, 2011.

[25] F. Herrera, Genetic fuzzy systems: Taxonomy, current research trends and prospects, Evolutionary Intelligence 1 (2008) 27 – 46.

[26] J. M. Cano Izquierdo, Y. A. Dimitriadis, E. Gómez Sánchez, J. López Coronado, Learnning from noisy information in FasArt and Fasback neuro-fuzzy systems, Neural Networks 14 (4-5) (2001) 407–425.

[27] D. Nauck, R. Kruse, Neuro-fuzzy systems for function approximation, Fuzzy Sets and Systems 101 (2) (1999) 261–271.

[28] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization, in: Proc. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Barcelona, Spain, 2001, pp. 95–100.

[29] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Computing 13 (3) (2009) 307 – 318.

[30] L. J. Eshelman, The CHC adaptive search algorithm : How to have safe search when engaging in nontraditional genetic recombination, Foundations of Genetic Algorithms 1 (1991) 265–283.

[31] H. Ishibuchi, T. Murata, I. B. Türksen, Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems, Fuzzy Sets and Systems 89 (2) (1997) 135 – 150.

[32] R. Alcalá, M. J. Gacto, F. Herrera, J. Alcalá-Fdez, A multi-objective genetic algorithm for tuning and rule selection to obtain accurate and compact linguistic fuzzy rule-based systems, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 15 (5) (2007) 539 – 557.

[33] C. H. Nguyena, V. T. Hoangc, V. L. Nguyenc, A discussion on interpretability of linguistic rule based systems and its application to solve regression problems, Knowledge-Based Systems 88 (2015) 107 – 133.

[34] M. Antonelli, P. Ducange, B. Lazzerini, F. Marcelloni, Learning knowledge bases of multi-objective evolutionary fuzzy systems by simultaneously optimizing accuracy, complexity and partition integrity, Soft Computing 15 (12) (2011) 2335 – 2354.

[35] M. Setnes, R. Babuška, Rule base reduction: Some comments on the use of orthogonal transforms, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 31 (2) (2001) 199 – 206.

[36] R. Babuska, Fuzzy Modeling for Control, Kluwer Academic Press, 1998.

Generación eficiente de reglas candidatas de calidad en problemas de alta dimensionalidad

Javier Cózar
Dept. de Sistemas Informáticos
Universidad of Castilla-La Mancha
Albacete, España
javier.cozar@uclm.es

Luis de la Ossa
Dept. de Sistemas Informáticos
Universidad of Castilla-La Mancha
Albacete, España
luis.delaossa@uclm.es

José Antonio Gámez
Dept. de Sistemas Informáticos
Universidad of Castilla-La Mancha
Albacete, España
jose.gamez@uclm.es

***Resumen*—Existen multitud de métodos para la generación de modelos basados en reglas difusas a partir de conjuntos de datos. Debido a que el número de reglas que puede generarse es exponencial con respecto al número de variables, los algoritmos se suelen apoyar en técnicas de búsqueda, generalmente metaheurísticas, para determinar qué reglas componen el sistema final. Aún así, sigue siendo necesario reducir el conjunto de reglas candidatas de manera sustancial. En esta línea, existen métodos que generan un primer conjunto de reglas, y posteriormente lo filtran mediante un algoritmo iterativo, basándose en criterios de calidad y genericidad. Sin embargo, este proceso es muy costoso. En este trabajo proponemos un método de generación de reglas candidatas de una sola etapa, basado en el principio Apriori, y que incorpora el uso de una métrica de calidad. Hemos llevado a cabo una experimentación con 7 problemas de regresión, desde 16 hasta 80 variables. Los resultados muestran un buen comportamiento en términos de precisión, y una mejora muy notable en cuanto a los tiempos de ejecución.**

***Index Terms*—Sistemas Basados en Reglas Difusas, Regresión, Takagi Sugeno Kang, Alta dimensionalidad**

I. Introducción

Los sistemas basados en reglas difusas (SBRDs) [1]–[3] son modelos compuestos por un conjunto de reglas del tipo "Si *antecedente* entonces *consecuente*". Su expresividad radica en que los espacios de entrada que activan distintas reglas pueden solaparse, por lo que una entrada generalmente activa más de una regla, y la predicción del modelo se obtiene mediante la composición de las salidas correspondientes.

La ventaja de los SRBDs se centra en dos aspectos. Por un lado, son modelos altamente interpretables, por lo que proporcionan al experto información relativamente clara sobre las relaciones que existen en los datos, además de permitir la incorporación de conocimiento externo antes o durante el proceso de aprendizaje. Por otro lado, estos modelos trabajan de forma inherente con la incertidumbre asociada a los datos, por lo que son muy robustos.

Muchos SBRDs representan la información mediante un conjunto de variables difusas, y un conjunto de reglas difusas construídas a partir de estas variables. Esta división da lugar, principalmente, a dos tipos de algoritmos para el aprendizaje de SRBDs: los que aprenden el conjunto de variables difusas (las particiones) y la base de reglas simultáneamente; y los que aprenden la base de reglas una vez establecido el conjunto de variables difusas.

Este trabajo propone un método que parte de un conjunto de variables difusas previamente establecido, y lleva a cabo el aprendizaje de la base de reglas difusas, en problemas de (relativamente) alta dimensionalidad. En concreto, para reglas TSK-0, cuyo consecuente es un número real. El aprendizaje de SBRDs es un proceso costoso debido, principalmente, a la explosión combinatoria de reglas: en un problema con n variables, y en el que cada variable difusa se define mediante una partición con k conjuntos difusos, pueden generarse hasta $k^{(n+1)} - 1$ reglas diferentes. Este hecho, hace impracticable trabajar con el conjunto completo de posibles reglas. Por ello se suele dividir el proceso de aprendizaje en dos etapas [4], [5]: 1) generación de un conjunto de reglas candidatas; y 2) uso de un algoritmo de búsqueda para determinar el subconjunto de reglas que componen el modelo final.

La generación de reglas candidatas tiene un impacto crítico tanto en el rendimiento (error) del modelo, como en el coste del proceso de aprendizaje. Una estrategia para llevarla a cabo consiste en generar, en una primera fase, aquellas reglas candidatas que tienen mayor relevancia en relación a la cobertura de las instancias del conjunto de datos. Para ello, se utilizan algoritmos basados en el principio Apriori [6]. Posteriormente, en una segunda fase, y mediante un proceso iterativo, se seleccionan aquellas reglas de mayor calidad, minimizando el solapamiento entre las reglas selecionadas en la medida de lo posible. Este enfoque permite abordar el problema en entornos de alta dimensionalidad. Sin embargo, la primera fase, basada en el algoritmo Apriori, genera todavía un número demasiado elevado de reglas, por lo que la posterior selección es excesivamente costosa. En este trabajo proponemos unificar ambas fases, incorporando una métrica de calidad monótona al proceso basado en el principio Apriori, que permita obtener el conjunto de reglas candidatas de manera más eficiente.

La estructura de este trabajo se divide en 4 secciones además de esta introducción. En la Sección II se describen los SBRDs y, en particular, los basados en reglas tipo TSK de orden cero. En la Sección III se describirá cómo se afronta el problema de la generación de reglas candidatas en trabajos previos, y la propuesta que se expone en este trabajo para realizarlo de una forma más eficiente. Posteriormente, en la Sección IV, se detallará la configuración y los resultados de los experimentos. Por último, en la Sección V, se expondrán las conclusiones.

II. Sistemas basados en reglas difusas TSK-0

Los SBRDs de tipo *Takagi-Sugeno-Kang* (TSK) [7] son modelos compuestos por: un conjunto de variables difusas [1], que se caracteriza por los conjuntos y etiquetas lingüísticas en que se particiona cada una de ellas [8]; y una base de reglas definida sobre estas variables. El consecuente de cada regla TSK es una función polinomial de las variables de entrada y, en el caso de los sistemas de orden cero (TSK-0) el consecuente es un polinomio de orden cero, es decir, un número real.

El antedecente de una regla difusa está formado por uno o varios predicados de la forma X es F, donde X es una variable del problema y F es un conjunto difuso incluído en la partición asociada a la variable X. De esta forma, una regla de tipo TSK-0 se representa:

$$R_s : \text{Si } X_1 \text{ es } F_1^s \text{ y } \ldots \text{ y } X_n \text{ es } F_n^s \text{ entonces } Y = b_s$$

La regla describe una relación entre las variables de entrada y la salida. Dada una instancia $e_l = (x_1^l, x_2^2, \ldots, x_n^l)$, el proceso de inferencia asocia un grado de verdad h_s^l a la salida de la regla b_s, donde $h_s^l = T(\mu_{X_1}^{x_1^l}, \mu_{X_2}^{e_2^l}, \ldots, \mu_{X_n}^{e_n^l})$, siendo T una T-norma[1]. Debido a que una misma instancia puede disparar varias reglas, la inferencia en los SBRDs de tipo TSK obtiene la salida, $\hat{y}^l$, como la media de las salidas individuales generadas por cada regla $R_s \in \mathcal{RB}$, ponderada por su grado de verdad (ver Ecuación 1).

$$\hat{y}^l = \frac{\sum_{R_s \in \mathcal{RB}} h_s^l b_s}{\sum_{R_s \in \mathcal{RB}} h_s^l} \tag{1}$$

III. Generación de reglas candidatas TSK-0 en problemas de alta dimensionalidad

Los algoritmos que aprenden SBRDs partiendo de una definición de las variables difusas, suelen dividir la tarea en la generación de un conjunto de reglas candidatas, y la posterior selección del subconjunto de reglas y ajuste del consecuente de las mismas [4], [5], [9], [10]. Este trabajo se centra en la fase de extracción de reglas candidatas, por lo que puede ser aplicado a cualquier algoritmo que utilice esta estrategia.

Como nos centraremos únicamente en la construcción del conjunto de reglas candidatas, utilizaremos un algoritmo que genera la partición de las variables difusas en un solo paso. En trabajos relacionados se utiliza comunmente una distribución de los conjuntos difusos triangulares de igual anchura (ver Figura III). Sin embargo, este enfoque puede ser inadecuado cuando los datos no están distribuidos homogéneamente a lo largo del dominio de la variable, ya que la importancia de cada conjunto difuso sería diferente al cubrir una proporcion desigual del número de instancias del problema. La alternativa que utilizamos en este trabajo consiste en distribuir estos conjuntos difusos triangulares de forma equifrecuencual, de tal manera que cada conjunto cubra el mismo porcentaje de instancias del problema (ver Figura 2).

[1]En este trabajo usamos min como T-Norma.

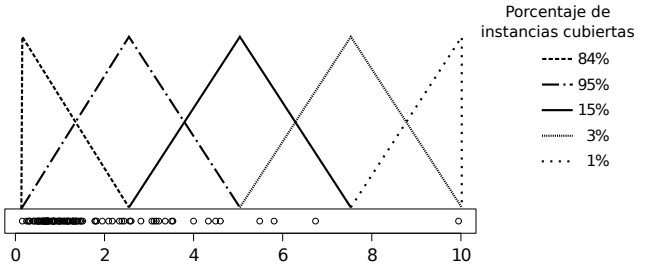

Figura 1. Particiones difusas con distribución equiespacial

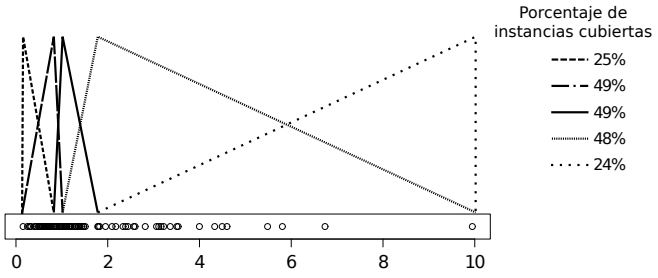

Figura 2. Particiones difusas con distribución equifrecuencial

De cara a seleccionar el subconjunto de reglas candidatas y fijar su consecuente, utilizaremos el algoritmo básico del estado del arte en sistemas TSK-0 [11]. Éste consiste en seleccionar el conjunto completo de reglas candidatas y, utilizando mínimos cuadrados, estimar sus consecuentes de forma que se minimice el error del sistema completo de reglas con respecto al conjunto de datos de entrenamiento $\mathcal{E}$. La expresión matricial utilizada por el método de mínimos cuadrados se muestra en la Ecuación 2, donde $c_s^l = \frac{h_s^l}{\sum_{\mathcal{RB}} h_s^l}$. El vector columna de consecuentes, $[b_1, \ldots, b_{|\mathcal{RB}|}]^T$, se obtiene como $B = C^{-1} \cdot Y$, donde C^{-1} se calcula utilizando la pseudoinversa de Moore-Penrose construida a mediante el método SVD [12].

$$Y = C \cdot B = \begin{bmatrix} y^1 \\ \vdots \\ y^l \\ \vdots \\ y^{|\mathcal{E}|} \end{bmatrix} = \begin{bmatrix} c_1^1 & \cdots & c_{|\mathcal{RB}|}^1 \\ \vdots & \vdots & \vdots \\ c_1^l & \cdots & c_{|\mathcal{RB}|}^l \\ \vdots & \vdots & \vdots \\ c_1^{|\mathcal{E}|} & \cdots & c_{|\mathcal{RB}|}^{|\mathcal{E}|} \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ \vdots \\ b_{|\mathcal{RB}|} \end{bmatrix} \tag{2}$$

Nuestra propuesta para la generación de reglas candidatas, parte del trabajo propuesto en [5], en el que se genera el conjunto de reglas candidatas TSK-0 en dos fases. En primer lugar, extrayendo aquellas que tienen una alta interacción con el conjunto de datos de entrenamiento; y posteriormente filtrando aquellas reglas que tienen un mínimo grado de calidad individual, medida como la varianza de dicha regla con respecto al conjunto de instancias que la activan. A continuación se detallan ambas etapas, y el método alternativo propuesto.

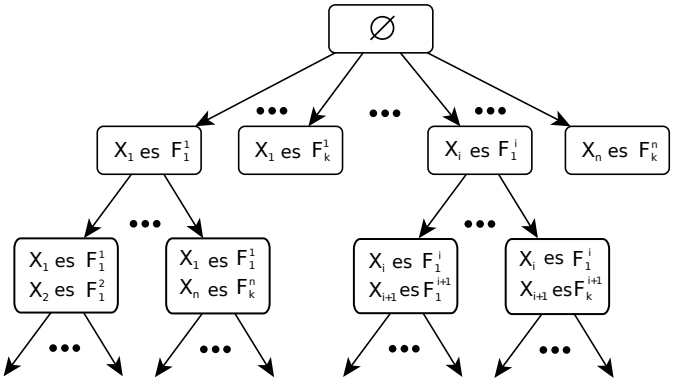

Figura 3. Estructura general del árbol de reglas candidatas

III-A. *Extracción de reglas candidatas*

El proceso para la extracción de reglas difusas propuesto en [5] está basado en el conocido método *Apriori* para la extracción de reglas asociativas [6].

Un conjunto $F_s = (F_i^s, \ldots, F_j^s)$, representa una serie de predicados X_i es $F_i^s, \ldots, F_j$ es F_j^s, tal que $X_i \neq X_j \forall i, j$. Visto de otra manera, F_s es la representación del antecedente de una regla candidata R_s. El soporte de un conjunto F_s cuantifica el número de ejemplos que emparejan con la regla, y en qué medida lo hacen. Se expresa como:

$$soporte(F_s) = \frac{\sum_{e_l} T(\mu_{F_i}^{x_i^l}, \ldots, \mu_{F_j}^{x_j^l})}{|\mathcal{E}|} = \frac{\sum_{e_l} h_s^l}{|\mathcal{E}|}$$

Se denomina conjunto frecuente a aquel cuyo soporte es mayor que un umbral $min\_soporte$. El significado de este parámetro está asociado con el número de instancias que activan el antecedente de la regla asociada. De esta manera, si $min\_soporte = 0{,}05$, significa que el conjunto será frecuente si un 5 % de las instancias están cubiertas por la regla asociada con grado de cobertura 1, o un mayor porcentaje si este grado de cobertura no es máximo para todas ellas.

El método de generación de reglas candidatas devuelve aquellas que superen el umbral de mínimo soporte. Para conseguirlo de manera eficiente, se construye un árbol de búsqueda [4] en el que cada nodo representa un conjunto o antecedente F^s, de modo que un nodo correspondiente al conjunto $F^{s'}$ es hijo de otro, correspondiente al conjunto F^s, si $F^{s'}$ se obtiene añadiendo un predicado a F^s (Figura 3).

El árbol se construye desde la raíz, y considerando el principio apriori, según el cual, si un nodo es no frecuente, ninguno de sus nodos hijos será frecuente, por lo que no es necesario expandirlo.

Si la métrica utilizada, en este caso el soporte, es monótonamente creciente o decreciente, el resultado del algoritmo es equivalente al devuelto a partir de una exploración completa del árbol. En el caso del soporte, éste es monotonamente decreciente. Por convenio, se parte de un soporte inicial en el nodo raíz (regla sin antecedentes) igual a uno. Conforme se añaden predicados al antecedente vacío, el soporte solo puede disminuir.

III-B. *Filtrado de reglas candidatas*

En problemas de alta dimensionalidad, el conjunto de reglas candidatas generado por el proceso anterior puede ser, todavía, excesivamente elevado. Esto se debe, en parte, a que muchas reglas son redundantes (cubren conjuntos muy parecidos de ejemplos). Además, las reglas son seleccionadas por cobertura, por lo que no se tiene en cuenta la capacidad de predicción de las mismas. Debido a esto, en una fase posterior, se han de elegir las mejores reglas dentro del conjunto total de reglas candidatas.

El proceso de filtrado, llamado *prescreening*, realiza una selección iterativa de las reglas candidatas basada en su calidad predictiva, y trata de evitar la redundancia entre ellas. La métrica utilizada para calcular la calidad individual de las reglas, es el error cuadrático medio de predicción de cada regla con respecto a las instancias la activan. Como los valores de esta métrica permanecen invariables a lo largo del proceso, son precalculados antes de comenzar el proceso iterativo. Para evaluar el grado de redundancia se mide, para cada instancia e_l, en la iteración i, el número de reglas ya seleccionadas que ala activan, denotado como c_l^i. Para cada regla, se calcula entonces la media de los grados de activación con cada instancia del conjunto de datos, ponderada por el peso asociado a cada una de ellas, siendo éste $w_l^i = \frac{1}{c_l^i + 1}$. De esta manera, si una instancia está cubierta por varias reglas ya seleccionadas, la suma de los grados de activación ponderados disminuirá, favoreciendo la selección de otras reglas cuyo grado de redundancia es menor.

Inicialmente el conjunto de reglas candidatas preseleccionadas es $\mathcal{RB}_c = \emptyset$. En cada iteración, i, se seleccionará la mejor regla de acuerdo a la métrica expresada en la Ecuación 3, donde $ECM(\mathcal{E}^s)$ es el error cuadrático medio de la variable objetivo para las instancias que activan la regla R_s. Dado que la parte que mide el grado de redundancia y el ECM tienen un dominio diferente, éstos se normalizan al rango $[0-1]$ en cada iteración. Si todas las instancias están cubiertas un mínimo de k_t reglas, el proceso iterativo termina.

$$wWRAccR(R_s, i) = \sum_{e_l} \frac{w_l^i(e_l) \cdot h_s^l}{w_l^i(e_l)} \cdot ECM(\mathcal{E}^s) \quad (3)$$

III-C. *Propuesta de unificación de las fases de generación de reglas y filtrado*

El coste del algoritmo de filtrado (*prescreening*) descrito anteriormente es muy elevado. Esto se debe, por una parte, a que el número de reglas generadas con *apriori* suele ser muy grande; por otra parte, en cada iteración se debe calcular, para cada instancia, el grado de cobertura con cada regla ponderado por el peso de dicha instancia. En este trabajo proponemos el diseño de una métrica que permita cuantificar

la calidad de una regla, y que sea monónonamente creciente o decreciente, de tal manera que se pueda integrar en la fase inicial de generación de reglas candidatas (basada en *apriori*), evitando el posterior uso de un algoritmo de filtrado.

Se puede comprobar fácilmente que el ECM no es una métrica montótona. Por ejemplo, supóngase el conjunto de valores $Y = \{1, 2, 3, 2\}$, y una predicción $\hat{Y} = \{2, 2, 2, 2\}$. En ese caso, $ECM(Y, \hat{Y}) = \frac{2}{4}$. Si se reduce el conjunto de instancias, de tal manera que $Y' = \{1, 2, 3\}$, y la predicción es $\hat{Y}' = \{2, 2, 2\}$, $ECM(Y', \hat{Y}') = \frac{2}{3}$. Es decir, aumenta. Por otro lado, si el conjunto $Y' = \{2, 2\}$, y $\hat{Y}' = \{2, 2\}$, entonces $ECM(Y', \hat{Y}') = \frac{0}{2}$, es decir, el ECM se reduce.

Debido a esto, la métrica que se propone utilizar es el error cuadrático, que sí es monótonamente decreciente, al igual que el soporte. Sea N el número de instancias del conjunto de datos ($N = |\mathcal{E}|$). Y sean $\sum_{i=1}^{N}(y_i - \overline{y})^2$ el error cuadrático para N instancias (ECM_N), y $\sum_{i=1}^{N+1}(y_i - \overline{y})^2$ el análogo para $N+1$ instancias (ECM_{N+1}). Entonces,

$$ECM_N \leq ECM_{N+1}$$

$$\sum_{i=1}^{N}(y_i - \overline{y})^2 \leq \sum_{i=1}^{N}(y_i - (\overline{y} + \delta))^2 + (y_{N+1} + \delta)^2,$$

donde se expresa la media para las $N + 1$ instancias como la media para N instancias más una variación δ. Para demostrar que la inecuación anterior siempre se satisface, se parte de que, como $(y_{N+1} + \delta)^2$ es un número positivo, la expresión se puede simplificar a:

$$\sum_{i=1}^{N}(y_i - \overline{y})^2 \leq \sum_{i=1}^{N}(y_i - (\overline{y} + \delta))^2$$

$$\sum_{i=1}^{N}(y_i^2 + \overline{y}^2 - 2x\overline{y}) \leq \sum_{i=1}^{N}(y_i^2 + (\overline{y} + \delta)^2 - 2y(\overline{y} + \delta)$$

$$\sum_{i=1}^{N}(y_i^2 + \overline{y}^2 - 2x\overline{y}) \leq \sum_{i=1}^{N}(y_i^2 + \overline{y}^2 + \delta^2 + 2\overline{y}\delta - 2y\overline{y} - 2y\delta).$$

Restando la parte izquierda de la inecuación a ambos lados:

$$0 \leq \sum_{i=1}^{N}(\delta^2 + 2\overline{y}\delta - 2y\delta)$$

$$0 \leq N\delta^2 + 2N\overline{y}\delta - 2\delta\sum_{i=1}^{N} y_i.$$

La media de $y_1, \ldots, y_N$, $\overline{y}$ se puede expresar como:

$$0 \leq N\delta^2 + 2N\delta\frac{\sum_{i=1}^{N} y_i}{N} - 2\delta\sum_{i=1}^{N} y_i$$

$$0 \leq N\delta^2 + 2\delta\sum_{i=1}^{N} y_i - 2\delta\sum_{i=1}^{N} y_i$$

$$0 \leq N\delta^2,$$

siendo N siempre un número positivo, al igual que δ^2. Por lo tanto, la inecuación se cumple siempre, por lo que se demuestra que el error cuadrático es monótono.

Utilizando el EC como métrica de calidad, se puede proceder de manera similar a como se hace con el soporte para la construcción del árbol de reglas candidatas. Es decir, puede fijarse un umbral máximo de EC, y solo las reglas que cumplen este criterio serán incluidas en $\mathcal{RB}_c$ como reglas candidatas. Sin embargo, como las métricas de cobertura (soporte) y de calidad de predicción (EC) son métricas contrapuestas (un valor alto de soporte es positivo, mientras que un valor alto de EC es negativo), no se pueden combinar para computar la decisión de podar o no una rama. Por ello, la solución que se adopta consiste en continuar con el mismo criterio de generación, basado únicamente en el soporte, y utilizar el EC como criterio para incluir o no la regla derivada del nodo en $\mathcal{RB}_c$.

En estudios preliminares se observó que el parámetro de maxímo EC es muy dependiente del conjunto de datos, y debía ser ajustado específicamente. Para solucionar este problema, se aplica una técnica de poda selectiva sobre los nodos descendientes, en lugar de aplicarla a nivel de expansión de nodos. Cuando un nodo cumple el criterio de mínimo soporte, éste será considerado como regla candidata y será expandido, generando el conjunto de nodos descendientes. En este momento, se filtrarán estos nodos en función de la calidad *con respecto al padre*. En conctreto, se descartarán los nodos cuyo EC dividido por el soporte de la regla R_s no disminuya en un porcentaje con respecto al EC del padre dividido por el soporte del nodo padre. El parámetro que determina este porcentaje se denomina *min_quality_improvement*. La intuición que sostiene este criterio (dividir el EC por el soporte de cada nodo) es la siguiente: un nodo puede ver disminuido el error cuadrático con respecto al padre simplemente por el hecho de que tiene menos instancias, pero aún así, los datos correspondientes (los que emparejan la regla) presentan la misma dispersión. Por ello, dividiendo la división por el soporte, constituye una buena métrica para indicar el porcentaje de mejora con respecto al padre.

IV. Experimentación

Para validar la calidad de las reglas candidatas generadas, se comparará la estrategia expuesta en [5] y la propuesta descrita anteriormente. Para ello, se partirá de un SBRDs con unas variables difusas prefijadas, calculadas a partir de una distribución equifrecuencial, utilizando 5 conjuntos difusos. En el último paso, con las reglas candidatas generadas, se estimará el consecuente de todas ellas mediante mínimos cuadrados. La comparación entre los dos enfoques se hará en relación a la capacidad de predicción, medido como la raíz cuadrada del error cuadrático medio (RECM), y en cuanto

a la complejidad del sistema generado, tanto en número de reglas candidatas generadas como en tiempo consumido (en segundos). Respecto al hardware utilizado, la máquina donde se ejecutaron los experimentos consta de un procesador Intel Xeon E5450 a 3.00GHz y 32GB de memoria RAM.

Para la comparativa se han utilizado 7 datasets del repositorio KEEL[2], con número de variables que va desde 16 hasta 85. En la Tabla I se puede observar, para cada problema, su número de variables e instancias, además del dominio de la variable de salida. Con respecto a la parametrización de los algoritmos de generación de reglas candidatas, se ha fijado el parámetro *min_soporte*= 0,05 en ambos casos, $k_t = 20$ para la fase *prescreening* del algoritmo descrito en [5] (como recomiendan los autores), y *min_quality_improvement*= 0,5 en nuestra propuesta. Finalmente, en problemas de alta dimensionalidad hemos notado que el algoritmo tendía a incluir un elevado número de antecedentes por regla. Este hecho disminuye enormemente la interpretabilidad del sistema. Para evitar este problema, hemos usado un umbral variable para el mínimo soporte, siendo éste $min\_support_i = min\_support \cdot \sqrt{depth\_level}$, siendo $depth\_level$ la profundidad del nodo en el árbol de búsqueda.

Tabla I
Caracteristicas de los problemas utilizados en la experimentación

problema	#variables	#instancias	rango Y
house	16	22784	[0, 500001]
elevators	18	16599	[0.012, 0.078]
compactiv	21	8192	[0.0, 99.0]
pole	26	14998	[0.0, 100.0]
puma32h	32	8192	[-0.086661, 0.089805]
ailerons	40	13750	[-0.0036, 0.0]
tic	85	9822	[0, 1]

En la Tabla II se muestran los errores de predicción (RECM tanto para entrenamiento como para test) y el tiempo de aprendizaje en segundos. En la Tabla III se muestra la complejidad de los modelos generados por ambos algoritmos, mostando el número medio de reglas candidatas (|reglas|) y el número medio de antecedentes por regla ($|\#ant|$).

Como puede observarse, en cuanto a capacidad de predicción, nuestra propuesta es mejor para los 7 conjuntos de datos, tanto en error de entrenamiento como de test. Atendiendo a la complejidad de los sistemas de reglas generados, observamos que el método propuesto produce siempre conjuntos con un mayor número de reglas, aunque en 4 de los 7 problemas éstas tienen un menor número medio de antecedentes. Analizando conjuntamente esta información podemos concluir que, aunque nuestra propuesta siempre mejora en términos de error de entrenamiento, puede deberse en parte al aumento en el número de reglas candidatas, ya que la técnica de mínimos cuadrados tiende a sobreajustar el conjunto de entrenamiento. No obstante, también observamos una mejora en cuanto al error de test, lo que nos lleva a concluir que, además, la calidad de las reglas candidatas es similar e incluso mejor.

[2]www.keel.es/

Tabla II
Raíz del error cuadrático medio de entrenamiento y de test, y tiempo de ejecución para el algoritmo de generación basado en dos fases. En el caso de los problemas elevators, puma32h y ailerons se muestra el error $\cdot 10^3$.

	Generación en dos fases		
problema	RECM (entr.)	RECM (test)	Tiempo (s)
house	37292.24	37454.70	220.73
elevators	5.41	5.47	447.97
compactiv	10.92	11.04	183.58
pole	32.16	32.22	411.43
puma32h	24.46	24.59	419.07
ailerons	0.29	0.29	5204.02
tic	0.23	0.24	16784.83
	Generación en una fase		
problema	RECM (entr.)	RECM (test)	Tiempo (s)
house	**35506.00**	**36083.98**	**96.67**
elevators	**3.71**	**3.82**	**113.29**
compactiv	**3.07**	**3.36**	**103.88**
pole	**17.07**	**17.18**	**31.66**
puma32h	**20.00**	**20.42**	**63.39**
ailerons	**0.18**	**0.19**	**719.26**
tic	**0.22**	**0.23**	**313.71**

Tabla III
Número de reglas, reglas candidatas y número medio de antecedentes por regla

	Generación en dos fases		Generación en una fase	
problem	|reglas|	$\|\#ant\|$	|reglas|	$\|\#ant\|$
house	**76.60**	**1.39**	188.90	1.66
elevators	**68.10**	1.82	292.07	**1.81**
compactiv	**66.97**	**1.35**	446.37	1.88
pole	**42.97**	1.48	78.80	**1.17**
puma32h	**53.27**	**1.03**	207.00	1.23
ailerons	**46.60**	1.91	906.83	**1.89**
tic	**38.70**	2.26	384.47	**1.42**

Respecto al esfuerzo computacional, podemos ver una clara diferencia en cuanto a los tiempos de ejecución. En la Tabla II se aprecia una diferencia que puede alcanzar varios órdenes de magnitud. Este resultado es muy importante, y demuestra la capacidad de encontrar un conjunto de reglas candidatas de calidad de una forma muchísimo más eficiente.

Por último, en la Figura 4 podemos ver una comparativa de los tiempos aplicando una transformación logarítmica ($f(x) = \log_{10}(1+x)$), para poder visualizar las diferencias de tiempo para todos los problemas a la vez. Como se puede observar, las diferencias de tiempos se hacen más notables conforme aumenta el número de variables del problema.

V. Conclusión

En este trabajo se ha propuesto un algoritmo para la generación de reglas candidatas tipo TSK-0 en problemas de regresión. Para ello, se ha tomado como punto de partida el trabajo realizado en [5], el cual propone un algoritmo dividido en dos fases. La primera de ellas, apoyada en el principio Apriori para generar un conjunto de reglas general de forma eficientemente. Para ello, el proceso es guiado por la métrica de soporte de cada regla candidata. En una segunda fase, se aplica un filtrado enfocado a evitar la redundancia de reglas

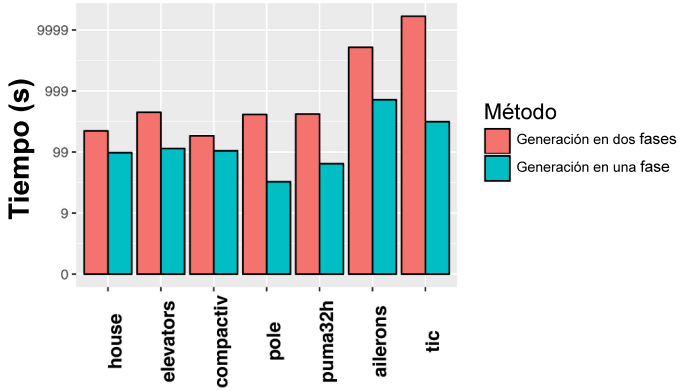

Figura 4. Comparativa de tiempos para ambos métodos en escala logarítmica

candidatas. Además, este proceso es guiado por una métrica de calidad individual de las reglas, centrada en evaluar la calidad predictiva de cada una de ellas.

Existen principalmente dos problemas en la propuesta anterior. Por un lado, se genera un elevado número de reglas candidatas en la primera fase. Por otro lado, la segunda fase requere un gran esfuerzo computacional al aplicar una selección de reglas candidatas (generadas por la primera fase) de forma iterativa. Estos problemas hacen que su uso sea ineficiente y, en ocasiones, impracticable en problemas de alta dimensionalidad. Nuestra propuesta se centra en la unificación de las dos fases, diseñando una métrica de calidad de predicción individual monótona, capaz de ser combinada con el soporte y que sirva como guía en el proceso de construcción del árbol de búsqueda del que se extraen las reglas candidatas.

Los resultados demuestran una calidad similar o incluso superior en cuando a la capacidad de predicción del conjunto de reglas candidatas, y una grandísima mejora en cuanto a la eficiencia. En trabajos futuros pretendemos ampliar la experimentación, utilizando un mayor conjunto de problemas, a la vez que validando el conjunto de reglas candidatas con varios algoritmos de búsqueda para la selección de las reglas que formarán el sistema de reglas. En este sentido, [13] y [14] se centran en utilizar algoritmos de búsqueda local, los cuales permiten realizar esta búsqueda eficientemente explotando la localidad espacial de las etiquetas difusas, lo que los hace adecuados para problemas de alta dimensionalidad.

Agradecimientos

Este artículo ha sido parcialmente financiado por fondos FEDER y la Agencia Estatal de Investigación (AEI/MINECO) mediante el proyecto TIN2016-77902-C3-1-P, y por la Junta de Comunidades de Castilla-La Mancha mediante el proyecto SBPLY/17/18050/000493. Javier Cózar también está financiado por el MECD mediante la beca FPU12/05102.

Referencias

[1] L. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, no. 1, pp. 28–44, 1973.

[2] E. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, pp. 1–13, 1975.

[3] E. Mamdani, "Applications of fuzzy algorithm for control a simple dynamic plant," pp. 1585–1588, 1974.

[4] J. Alcala-Fdez, R. Alcala, and F. Herrera, "A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning," *Fuzzy Systems, IEEE Transactions*, vol. 19, no. 5, pp. 857–872, 2011.

[5] J. Cózar, L. delaOssa, and J. A. Gámez, "TSK-0 fuzzy rule-based systems for high-dimensional problems using the apriori principle for rule generation," *Rough Sets and Current Trends in Computing*, vol. 8536, pp. 270–279, 2014.

[6] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *SIGMOD Rec.*, vol. 22, no. 2, pp. 207–216, Jun. 1993.

[7] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications for modeling and control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, no. 1, pp. 116–132, 1985.

[8] L. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning," *Information Science*, vol. 8, pp. 199–249, 1975.

[9] L. Wang and J. Mendel, "Generating fuzzy rules by learning from examples," *Systems, Man and Cybernetics, IEEE Transactions*, vol. 22, no. 6, pp. 1414–1427, 1992.

[10] O. Cordón and F. Herrera, "A proposal for improving the accuracy of linguistic modeling," *Fuzzy Systems, IEEE Transactions*, vol. 8, no. 3, pp. 335–344, 2000.

[11] K. Nozaki, H. Ishibuchi, and H. Tanaka, "A simple but powerful heuristic method for generating fuzzy rules from numerical data," *Fuzzy Sets and Systems*, vol. 86, pp. 251–270, 1997.

[12] V. C. Klema and A. J. Laub, "The singular value decomposition: Its computation and some applications," *Automatic Control, IEEE Transactions*, vol. 25, no. 2, pp. 164–176, 1980.

[13] J. Cózar, L. delaOssa, and J. A. Gámez, "Learning TSK-0 linguistic fuzzy rules by means of local search algorithms," *Applied Soft Computing*, vol. 21, no. 0, pp. 57–71, 2014.

[14] J. Cózar, L. delaOssa, and J. A. Gámez, "Learning compact zero-order tsk fuzzy rule-based systems for high-dimensional problems using an apriori + local search approach," *Information Sciences*, vol. 433 - 434, pp. 1 – 16, 2017.

Un modelo difuso lingüístico adaptativo para regresión con selección de reglas en Big Data

Antonio Ángel Márquez Ana María Roldán
Departamento de Tecnologías de la Información
Universidad de Huelva
Huelva – España
amarquez, amroldan@dti.uhu.es

Francisco Alfredo Márquez Antonio Peregrín
C.E. Avanzados en Física, Matemáticas y Computación
Universidad de Huelva
Huelva – España
alfredo.marquez, peregrin@dti.uhu.es

***Abstract*—En el diseño de sistemas basados en reglas difusas para regresión, son bien conocidos los modelos que permiten obtener la base de reglas a partir de conjuntos de datos por cubrimiento, así como los que posteriormente permiten mejorar la precisión del sistema empleando, por ejemplo, un ajuste del operador de defuzzificación adaptativo que, además, permita obtener un conjunto de reglas más compacto y con mejor nivel de cooperación. En este trabajo se lleva a cabo este proceso en entornos de *Big Data*, es decir, adaptando los algoritmos a esta tecnología para que se puedan utilizar partiendo de conjuntos de ejemplos que no serían manejables sin disponer de una infraestructura escalable.**

Keywords—regresión difusa; bases de reglas; defuzzificación adaptativa; Big Data; MapReduce.

I. Introducción

En el área de los *Sistemas Basados en Reglas Difusas* en general, y para modelado y control en particular, existe una considerable cantidad de trabajos que emplean algoritmos evolutivos para aprender y/o ajustar elementos del sistema, que van por ejemplo desde los de la Base de Conocimiento (BC) como el aprendizaje de la Base de Reglas (BR), o ajuste de la Base de Datos (BD), hasta incluso los operadores, como en [1], [2]. A todo este ámbito se le llamó el de los *Sistemas Difusos Evolutivos* [3], [4], [5], y en él se han desarrollado pues, gran cantidad de recursos muy útiles desde hace más de dos décadas. Estos algoritmos utilizan conjuntos de ejemplos desde los que aprender o evaluar los distintos citados elementos.

La posibilidad de emplear conjuntos de ejemplos más grandes con algunas de estas técnicas ha sido tratada a veces por distintos autores: cuando los conjuntos de ejemplos son muy grandes, las técnicas utilizadas pueden necesitar modificaciones o re-implementaciones para que sigan siendo aptas. Tradicionalmente, las mejoras en este sentido [5] se han orientado a la reducción de datos, al uso de aproximaciones o estimaciones, a la introducción de heurísticas en los algoritmos para mejorar su eficiencia, y en algunos pocos casos, al uso de computación distribuida. Sin embargo, es con la llegada de los recursos para afrontar el tratamiento de grandes conjuntos de datos en entornos de Big Data, cuando se dispone de herramientas para hacer verdaderamente escalables algunas de estas técnicas tradicionales.

Este trabajo trata precisamente del uso de recursos computacionales distribuidos del ámbito del *Big Data*, para emplear una técnica tradicional, como es el ajuste de la defuzzificación adaptativa en regresión y control difusos, cuando el conjunto de ejemplos es grande, o simplemente cuando el aprovechamiento de estos recursos nos permite realizar el mismo trabajo en tiempos menores, lo cual por ejemplo con el uso de algoritmos evolutivos, puede ser una interesante mejora.

II. Conocimientos Previos

En esta Sección se van a repasar brevemente dos elementos que empleamos en nuestra propuesta: la defuzzificación adaptativa, y la tecnología *MapReduce*.

A. Defuzzificación Adaptativa Evolutiva

La defuzzificación adaptativa es un recurso tradicional y bien conocido que permite adaptar el comportamiento del mecanismo de obtención de una salida específica desde la aportación de cada regla disparada en la fase de inferencia en regresión difusa, consiguiendo mejorar la precisión del modelo resultante, pues cada regla participa en dicho proceso de una forma específica y diferente al resto [1],[2].

Es posible encontrar múltiples propuestas en la literatura para conseguir regular o adaptar el mecanismo de defuzzificación mediante el uso de parámetros. Sin embargo, lo más frecuente suele ser utilizar la expresión (1), dados sus buenos resultados, su sencillez computacional y facilidad de implementación [2],

$$y_0 = \frac{\sum_i^N h_i \cdot \alpha_i \,.\, CG_i}{\sum_i^N h_i \,.\, \alpha_i}, \qquad (1)$$

donde h_i es el *matching degree* o grado de emparejamiento, α_i es el parámetro que se aplica con la regla i-esima, R_i, con $i=1$ *hasta* N, y CG_i es el Centro de Gravedad de la figura obtenida de la inferencia con la regla R_i. Como se puede ver, se trata de la expresión de un método de defuzzificación que actúa convirtiendo la aportación de cada regla en un número, y posteriormente calculando una media ponderada, es decir, del llamado *Modo-B*. El parámetro α_i pues, tomando valores en el intervalo [0,1], permite a su vez ponderar o reducir el efecto del

grado de emparejamiento, h_i, como si de un peso, w_i, se tratase [6].

Para conseguir el ajuste de los N citados α_i parámetros, es decir, reducir la diferencia entre la salida del sistema difuso y el conjunto de ejemplos, se podrían emplear diferentes métodos. En este sentido, el uso de algoritmos evolutivos suele ser una buena opción. La codificación empleada en este caso es de tipo real, pues como se ha indicado, el valor de los parámetros α_i se encuentra entre 0 y 1.

Obsérvese que este mecanismo de ajuste paramétrico no sólo permite que la aportación de cada regla sea regulada sino que a través de él, se consigue mejorar el nivel de cooperación entre las reglas de la BC [1]. Incluso llevado al extremo, la defuzzificación adaptativa puede emplearse como mecanismo de selección y reducción de reglas: si una regla tiene un valor 0 para su parámetro, dicha regla no participa o deja de ser considerada. El hecho de disponer de un mecanismo que permita eliminar reglas es particularmente interesante cuando dichas reglas provienen de un proceso de aprendizaje que no las ha valorado en su conjunto, como suelen ser los métodos basados en cubrimiento de ejemplos, donde cada regla se obtiene con criterios de calidad individual.

Cabe mencionar también en este sentido, que en [7] se propone un mecanismo para facilitar la eliminación de reglas con aportación baja, basado en eliminar aquellas reglas cuyo parámetro desciende por debajo de cierto umbral (es decir, no sólo cuando el valor es exactamente 0). En el citado trabajo, también se elimina el parámetro de las reglas cuyo valor está cercano a 1, suponiendo por tanto una reducción de la carga computacional y reduciendo la complejidad de la BC, pues se reduce el número de reglas, y el número de reglas con parámetro asociado.

B. *Tecnologías para Big Data*

En los últimos años se han desarrollado un conjunto de tecnologías para el almacenamiento, gestión y tratamiento de grandes conjuntos de datos de forma distribuida y escalable. A estas tecnologías se les suele llamar genéricamente, tecnologías para *Big Data* [8] y los tres pilares en los que se apoyan estas tecnologías actualmente son:

- Sistemas de archivos distribuidos, donde los grandes archivos de datos se almacenan divididos entre varios servidores, tales como el popular *Apache Hadoop Distributed File System* (HDFS) [10].
- Paradigmas de programación tales como *MapReduce* [11] o *Pregel* [12], que permiten implementar procesos de computación en *clusters* de computadoras.
- Entornos para *clusters* de computación como *Apache Hadoop* [10] o *Apache Spark* [13] que nos permiten organizar y gestionar grupos de computadoras tanto como sistemas de almacenamiento (a través de sistemas de archivos distribuidos) como estructuras de procesamiento de datos (implementando modelos de programación distribuida) eficientemente.

El paradigma de programación *MapReduce* lo introdujo Google en 2004 [11]. Una de sus mejor conocidas y empleadas implementaciones en código abierto es *Apache Hadoop*, la cual, implementa características como el almacenamiento y procesamiento escalables, los cuales se pueden usar de forma relativamente sencilla, tienen elevada tolerancia a fallos, alta disponibilidad, redundancia de datos automática, etc. A nivel de procesamiento, *Hadoop* está pensado para emplearse procesando datos en una sola pasada, esto es, no es eficiente implementando múltiples pasadas de procesamiento sobre los datos, así como tampoco está pensando para procesamiento de datos interactivamente. *Apache Spark* [13] sin embargo, sí resuelve estas situaciones.

Apache Spark es igualmente un entorno para la programación distribuida de código abierto, ideado como un paso adelante en cuanto a flexibilidad y eficiencia. No sólo permite utilizar el modelo de computación *MapReduce*, si bien considerablemente más rápido [13] que *Hadoop*, sino también *Pregel* [12] y el suyo propio. Pero una de las ventajas más interesantes de *Spark* es que permite implementar de forma eficiente el procesamiento de datos iterativo o multi-pasada, debido al uso particular de la memoria RAM que implementa, a través de una abstracción de memoria distribuida conocida como RDD (*Resilent Distributed Dataset*) [13], que le permite reducir el acceso a disco intermedio mejorándose así dramáticamente su rendimiento. Los RDDs son conjuntos de datos que se ubican físicamente troceados entre distintos servidores del *cluster*, y que pueden ser procesados en paralelo. Los programadores emplean estos RDDs a través de transformaciones, que los procesan produciendo otros RDDs, o bien mediante acciones, que les aplican un procesamiento que ofrece como salida un valor o cálculo. La tolerancia a fallos se obtiene precisamente gracias a los citados RDDs, ya que estos pueden ser reconstruidos automáticamente si por alguna razón, se perdiesen durante su procesamiento.

Los programas implementados para ser ejecutados en *Spark* se componen de una única aplicación *driver* que se ejecuta en el nodo maestro, y un conjunto de tareas que corren en paralelo en los *ejecutores* sobre los nodos trabajadores del *cluster*, devolviendo sus resultados al *driver*.

Cabe destacar que *Spark* también permite realizar trabajos de computación distribuida de forma interactiva, es decir, instrucción a instrucción desde intérprete de comandos, y que puede utilizar HDFS como sistema de archivos distribuido.

El mecanismo que se propone en este trabajo utiliza *Spark* y el modelo de computación *MapReduce*.

III. Modelo con Post-Procesamiento para el Ajuste de la Defuzzificación y Selección de Reglas

En esta Sección se propone un modelo distribuido escalable implementado en el entorno *Spark*, que consta de dos etapas: en una primera etapa se obtiene la BR basándonos en la versión *MapReduce* de un método archiconocido para la obtención de reglas por cubrimiento de ejemplos, y una segunda en la que también de modo distribuido y escalable, se ajustarán mediante un algoritmo evolutivo los parámetros de la defuzzificación, empleando umbrales para seleccionar el subconjunto de reglas que mejor cooperan.

A. *Obtención Escalable de la BR desde Conjuntos de Datos*

El llamado "Método de Wang y Mendel" dirigido por datos [14] (WM en adelante) es un algoritmo extraordinariamente conocido para obtener BRs partiendo de un conjunto de ejemplos. En un trabajo previo [15], propusimos una versión adaptada del mismo basada en el uso de *MapReduce*, que llamamos *WM-Escalable*. Los resultados que produce son idénticos a los que produciría la metodología secuencial original de WM, y resumimos a continuación:

Los conjuntos de datos de ejemplos son troceados y repartidos entre los servidores del *cluster* para ser procesados de forma separada. En este caso, para obtener la BR usando el método de WM, sólo se necesita un esquema de una sola pasada, es decir, una sola fase *Map* y a continuación una fase *Reduce*. Los elementos son:

- Programa *Driver*: El programa *driver* ejecutado en el nodo maestro, en primer lugar, crea la BD utilizando un número predefinido de funciones de pertenencia distribuidas uniformemente en el universo de discurso de cada variable. En segundo lugar, el conjunto de entrenamiento se divide en *n* particiones disjuntas con el mismo número de instancias, y cada una de estas particiones se distribuye entre los nodos trabajadores del *cluster* junto con la citada BD.
- Función *Map*: Las funciones *Map* consisten en la aplicación de la metodología original de WM por parte de los procesos *ejecutores* sobre nodos trabajadores, cada uno en su subconjunto de ejemplos, que quedan así cubiertos por alguna de las reglas generadas. El método de WM consigue esto en primer lugar cubriendo cada ejemplo con una regla difusa que utiliza las etiquetas que tienen el mayor grado de pertenencia para cada variable, y además le asigna un valor que es el grado de emparejamiento de esa regla con ese ejemplo, el cual será empleado en una segunda fase para reducir el conjunto de reglas, es decir, para no tener tantas reglas como ejemplos sino sólo un subconjunto combinado de las reglas más representativas. La salida de estos procesos *Map* se produce en términos de pares *clave-valor* (siendo recibida por el *driver*), donde la **clave** son *las etiquetas de los antecedentes de las reglas halladas*, y el **valor** asociado son *el consecuente de las reglas encontradas, junto con el grado de emparejamiento de las reglas*.
- Función *Reduce*: La tarea de las funciones *Reduce* en este esquema es la de recibir las distintas reglas producidas por las funciones *Map*, (RB_i), y agruparlas para construir la BR definitiva. Para ello, debe eliminar las reglas repetidas, es decir, las que tienen los mismos antecedentes y el mismo consecuente, y resolver las reglas contradictorias, es decir, las que con los mismos antecedentes tienen distinto consecuente, lo cual se lleva a cabo eligiendo la que viene acompañada por el mayor grado de emparejamiento. En términos del diseño *clave-valor* de *MapReduce*, la función *Reduce* recibe una lista de valores intermedios compuesta por pares agrupados por claves (**clave**: *etiquetas de la parte antecedente de una regla*, y **lista de valores**: *consecuentes para esa parte antecedente de una regla y sus grados de emparejamiento*). La BR final está pues compuesta por el conjunto de reglas obtenidas por las funciones *Reduce*.

B. *Aprendizaje Escalable Evolutivo de los parámetros para la Defuzzificación Adaptativa con Selección de Reglas*

En esta Subsección se describe la parte original de este trabajo, donde se propone un método evolutivo para aprender o ajustar los valores de la defuzzificación adaptativa utilizando conjuntos de ejemplos muy grandes de forma eficiente, y especialmente apropiada al uso de BRs aprendidas mediante métodos de cubrimiento de ejemplos, ya que incorpora un mecanismo de selección de reglas, tal como se introdujo previamente en la Sección II, al que llamaremos *Defuzzificación Adaptativa Evolutiva con Selección de Reglas Escalable* (abreviado como *E-DAESR*). La originalidad en este trabajo está, no en el diseño evolutivo de defuzzificación adaptativa [2], sino en la escalabilidad de la propuesta. No repetiremos pues los detalles ya descritos previamente sino que nos centraremos en el diseño *MapReduce* del mecanismo, el cual, es equivalente exactamente al modelo secuencial original [2].

1) *Algoritmo Evolutivo*

Previamente, se va a describir el esquema evolutivo empleado, ya que de su detalle depende la posterior descripción del mecanismo para la escalabilidad de la propuesta. Se basa en el modelo CHC [16]. Sus componentes son:

Esquema de codificación y población incial

El esquema de codificación empleado es el de [1], que consiste en un cromosoma de tipo real con N genes, α_i, que representan cada uno al parámetro asociado a cada regla R_i de la RB, y cada uno de ellos toma valores en el intervalo [0,1].

$$C = (\alpha_1, \ldots, \alpha_N) \mid \alpha_i \in \{0, 1\}$$

La población inicial se inicializa con todos los genes de cada cromosoma elegidos aleatoriamente en el intervalo anteriormente citado, excepto los de uno, cuyos genes se fijan a 1, lo que equivale al método estándar conocido como *CG ponderado por el grado de emparejamiento*.

Evaluación

La evaluación de los cromosomas consiste en tasar el modelo difuso con los parámetros de ese cromosoma. Dado que lo que se pretende resolver con esta propuesta es precisamente el uso de conjuntos de ejemplos muy grandes, este proceso es el que se realiza de forma distribuida por los *ejecutores* de *Spark*, como se describirá posteriormente. Para maximizar la precisión, minimizamos el Error Cuadrático Medio (MSE) cuya expresión es (2), donde $S[i]$ denota a los diferentes modelos difusos probados. Para ello, se utiliza un conjunto de evaluación compuesto por P pares de datos numéricos $Z_k = (x_k, y_k)$, $k=1,..,P$, siendo x_k los valores de las variables de entrada, e y_k los valores correspondientes de las variables de salida.

$$MSE(S[i]) = \frac{\frac{1}{2}\sum_{k=1}^{N} (y_k - S[i](x_k))^2}{N} \qquad (2)$$

Destacamos que, para llevar a cabo la selección de reglas mediante umbrales como en [7] se tiene en cuenta sólo y

simplemente cuando se evalúa el sistema difuso, es decir, los valores de los parámetros en los cromosomas se mantienen, pero por debajo de cierto umbral (se ha utilizado el valor 0.1), no se usa la regla, al igual que por encima de cierto umbral (se ha usado 0.9), se considera que el peso es 1.

Operador de cruce y criterio de rearranque

Empleamos el operador de cruce BLX-α [17], con $\alpha = 0.5$, y umbral inicial *L/4*. Cuando el umbral es menor que 0, se rearranca, construyéndose la población aleatoriamente excepto para el mejor de los padres de la población anterior.

2) Esquema MapReduce para la Defuzz. Adaptativa

A continuación se describe el diseño MapReduce realizado para construir el modelo *E-DAESR,* ilustrado en la Figura 1.

- Programa *Driver*: El programa *driver*, ejecutado en el nodo maestro, es el que se ocupa de llevar acabo la mayor parte del algoritmo evolutivo que aprende los parámetros de la defuzzificación asociados a cada una de las reglas R_i de la BR, cediendo el proceso más pesado, la evaluación con el conjunto de ejemplos, a la función *Map*, que se ejecuta de forma distribuida en el *cluster* en cada iteración del algoritmo evolutivo. El programa *driver* al inicio, divide y distribuye el conjunto de ejemplos en tantas unidades diferentes como elementos de procesamiento tenga el *cluster*. Asimismo, también distribuye una copia de la BC y la población completa de cromosomas a evaluar.
- Función *Map*: La evaluación de la población de cromosomas del algoritmo evolutivo es llevada a cabo por parte de los procesos *ejecutores* dentro de los nodos *trabajadores* del *cluster*, con su fracción del conjunto de ejemplos, obteniéndose por tanto el MSE para ese subconjunto. La función *Map* produce una lista de pares *clave-valor* intermedios que son devueltos al proceso *driver* tras su ejecución, de modo que la **clave** son *los cromosomas (parámetros asociados con cada regla)* y el **valor** es *el valor de evaluación (MSE) obtenido por la BR con los parámetros para la defuzzificación que lleva cada cromosoma.*
- Función *Reduce*: También son realizadas por los *ejecutores*: una o varias funciones *Reduce* reciben los resultados de los *Map*s y los combinan para tener el resultado completo de la evaluación para cada cromosoma. En términos de pares *clave-valor*, las funciones *Reduce* reciben una lista de valores intermedios agregados por la clave, de modo que la **clave** *es el cromosoma (los valores de los parámetros para cada regla)*, y como **lista de valores**, una *lista de la evaluación de cada cromosoma en cada partición de datos*, y ofrecen una lista de cromosomas con su valor de evaluación con el conjunto de ejemplos completo. La evaluación obtenida por todos los cromosomas es enviada al programa driver para que continúe el proceso evolutivo.

IV. Estudio Experimental

En esta Sección se describe el estudio experimental llevado a cabo para comprobar el funcionamiento del modelo propuesto, tanto en cuanto a precisión como a escalabilidad.

En primer lugar, se hará referencia a los conjuntos de datos empleados, posteriormente se describirá la configuración seleccionada para el estudio experimental y los test estadísticos, y finalmente se mostrarán los resultados y su análisis, así como el estudio de escalabilidad.

Se van a utilizar 10 problemas con diferente nivel de dificultad, por ejemplo, con distinto número de variables e instancias, etc. Provienen del repositorio KEEL [18], y se muestran en la Tabla I junto con sus características.

A. Configuración de los Experimentos

En cuanto la configuración de los experimentos, se ha utilizado validación cruzada de orden 5, es decir, se han hecho 5

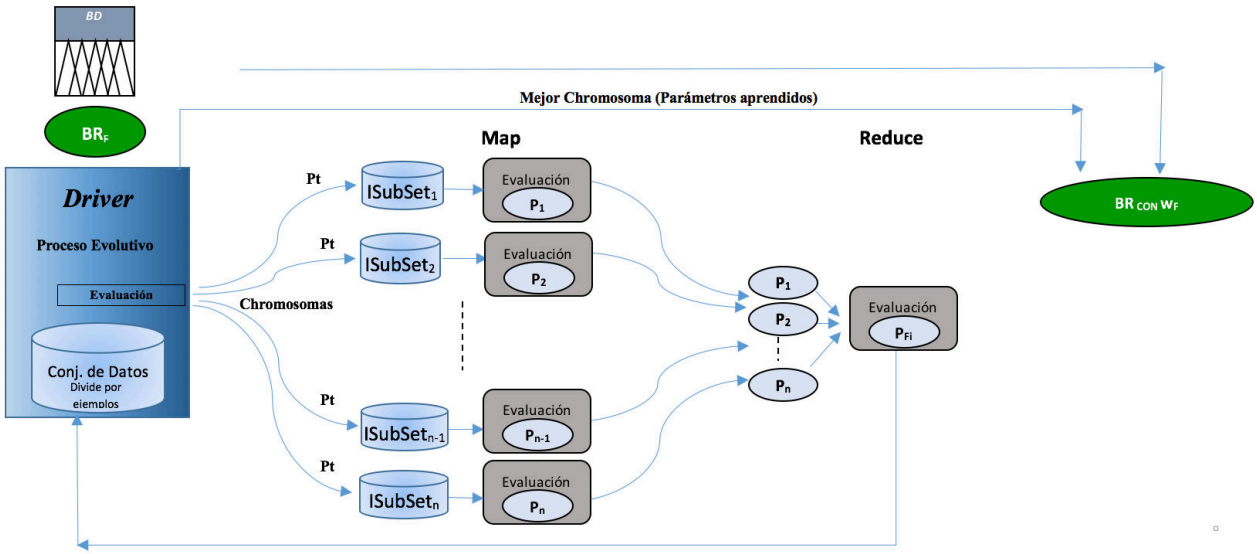

Fig. 1. Esquema de E-DAESR

TABLE I. CONJUNTOS DE DATOS UTILIZADOS, DISPONIBLES EN EL REPOSITORIO KEEL (HTTP://SCI²S.UGR.ES/KEEL/DATASETS.PHP)

Problem	Abbreviation	Instances	Variables
Delta-elv	DELV	9517	6
California	CAL	20640	8
Mv	MV	40768	10
House	HOU	22768	16
Elevators	ELV	16599	18
Compactiv	CA	8192	21
Pole	POL	14998	26
Puma32	PUM	8192	32
Airelons	AIL	13750	40
Tic	TIC	9822	85

particiones aleatorias de los conjuntos de datos, cada una con un 20%, y la combinación de 4 de ellos, es decir, el 80%, se han usado como conjunto de entrenamiento, mientras que los restantes son el conjunto de prueba.

En cuanto a los sistemas difusos, las BCs utilizadas emplean 3 etiquetas lingüísticas triangulares en todos los casos. El operador utilizado como conjunción y como inferencia para ambos casos es la t-norma del mínimo. El método de defuzzificación empleado para la obtención de la BR, por tanto, no adaptativo, es el *CG ponderado por el grado de emparejamiento*, mientras que el adaptativo de la propuesta es el de la expresión (1) (los cuales son equivalentes cuando el parámetro de cada regla tiene el valor 1).

En cuanto al algoritmo evolutivo, cuyo modelo se ha descrito en la Sección III, se ha ejecutado 30 veces: 6 semillas para el generador de números aleatorios, y 5 particiones para cada conjunto de datos o problema. La longitud de población utilizada es 61 cromosomas; la probabilidad de cruce se ha fijado a 1, y el número máximo de evaluaciones configurado ha sido 100000.

Para realizar la experimentación hemos utilizado un *cluster virtual* formado por 17 servidores con 4 núcleos y 8 GB de RAM cada uno. Uno de ellos se ocupa de ejecutar exclusivamente el programa *driver*, es decir, actúa de nodo maestro, y los 16 restantes como trabajadores para los procesos *ejecutores*.

Para realizar los estudios de escalabilidad se ha optado por hacer las dos siguientes configuraciones del *cluster*:

- Una configuración simple, con solo 2 núcleos, para poder medir el tiempo necesario por un servidor básico.
- Una configuración en modo *cluster* con el total de los 16 servidores, tanto con 16 como con 32 núcleos.

Se han realizado test estadísticos [19] [20], para comparar los resultados, empleando un *signed-rank test* de Wilcoxon.

B. Resultados Obtenidos y Análisis

La Tabla II muestra los resultados obtenidos por el modelo propuesto con defuzzificación adaptativa escalable, *E-DAESR,* en comparación con el modelo que no la utiliza, ambos con las mismas BRs obtenidas por el método de *WM-Escalable* [15]. En particular, para cada modelo, se muestra el número de reglas obtenidas, y los MSE en entrenamiento y prueba. Observando esta información, analizamos que:

- El número de reglas que se obtienen en general empleando el método propuesto *E-DAESR* es considerablemente inferior al que se obtiene empleando el método de cubrimiento (*WM-Escalable*), por tanto, se consiguen modelos más compactos. La excepción son precisamente los problemas que por su complejidad, presentan mayor número de reglas: quizás el espacio de búsqueda es "demasiado grande" para la configuración del algoritmo evolutivo básica empleada, por lo que habría que probar otros ajustes que hiciesen la búsqueda más efectiva.
- La precisión en general es de nuevo considerablemente mejor en el modelo propuesto. Sin embargo, se observa que en algunos problemas dicha reducción es escasa, coincidiendo con los problemas que en el párrafo anterior destacábamos con una inferior reducción en el número de reglas: se hace conveniente probar configuraciones diferentes.

Las Tablas III y IV muestran los resultados de los test de Wilcoxon empleados para confirmar que la reducción del número de reglas y la mejora en la precisión son significativas.

TABLA II. WM-ESCALABLE VS. E-DAESR (LOS VALORES DE MSE EN ESTA TABLA DEBEN MULTIPLICARSE POR 10^{-6}. 10^{8}. 10^{9}.10^{-6}. 10^{-8} Y 10^{-4} PARA DELV. CAL. HOU. ELV. AIL Y TIC RESPECTIVAMENTE).

Datasets	WM-Escalable			E-DAESR		
	#R	MSE_{tra}	MSE_{tst}	*#R*	MSE_{tra}	MSE_{tst}
DELV	129.4	2.372718	2.375347	**47.1**	1.602017	**1.628900**
CAL	123.4	5.319310	5.335769	**41.2**	3.215826	**3.225589**
MV	3677.8	13.795066	13.998455	**3126.4**	6.281729	**6.669283**
HOU	756.4	16.445739	16.724978	**281.8**	9.706077	**10.455755**
ELV	508.8	16.446985	16.494120	**108.3**	9.829008	**10.052612**
CA	424.8	40.383642	40.956170	**139.0**	5.345838	**6.636041**
POL	1087.0	399.773700	401.554456	**559.1**	174.488994	**181.557838**
PUM	6553.6	0.000250	0.000589	**6500.1**	0.000242	**0.000587**
AIL	1072.0	3.450536	3.496296	**470.3**	1.893806	**2.004145**
TIC	5802.4	151.609692	500.967975	**5707.9**	126.557568	**499.817874**

TABLA III. WILCOXON TEST PARA COMPARAR LA PRECISIÓN DEL MODELO WM-ESCALABLE CON EL MODELO E-DAESR. (R+ SE CORRESPONDE CON LA SUMA DE LOS RANKINGS PARA EL E-DAESR Y R- AL WM-ESCALABLE)

Comparación	R+	R-	p-value
E-DAESR vs. WM-Escalable	0	55	0.0019532

TABLA IV. WILCOXON TEST PARA COMPARAR EL #R DEL MODELO WM-ESCALABLE CON EL DEL MODELO E-DAESR. (R+ SE CORRESPONDE CON LA SUMA DE LOS RANKINGS PARA EL E-DAESR Y R- AL WM-ESCALABLE)

Comparación	R+	R-	p-value
E-DAESR vs. WM-Escalable	0	55	0.0019532

C. Escalabilidad

Uno de los elementos más interesantes de emplear un enfoque escalable es precisamente, que escale correctamente, es decir, que aumentando el número de elementos de proceso en el *cluster,* le permita reducir los tiempos de ejecución.

Como se comentó en la descripción del estudio experimental, se ha empleado una configuración en el *cluster* con 2 núcleos como equipo básico, y posteriormente con 16 y 32. Los resultados se muestran en la Figura 2: se observa como un mayor número de elementos de cómputo, se refleja en el

speed-up (ganancia del tiempo empleado por 32 y 16 núcleos frente al empleado con sólo 2) si bien no es lineal debido a la carga computacional derivada del *framework*.

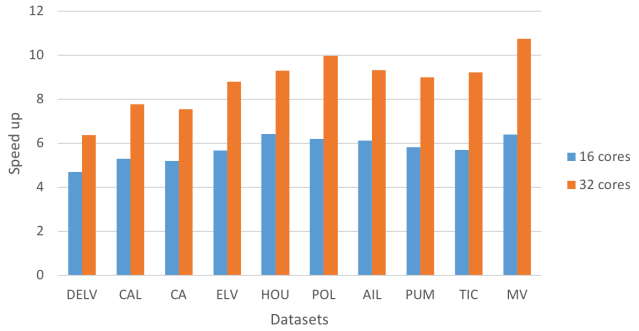

Fig. 2. Escalabilidad obtenida para cada conjunto de datos

V. Conclusiones

El objetivo planteado en este trabajo era estudiar la adaptación de la metodología del uso de defuzzificación adaptativa con selección de reglas mediante umbrales, a entornos escalables, es decir, la adaptación para su uso en *frameworks* de *Big Data*, y al modelo de computación *MapReduce* implementado en *Apache Spark*.

El resultado de este estudio preliminar es positivo, es decir, se ha conseguido un modelo equivalente (sus resultados son idénticos al modelo secuencial del que proviene) que funciona correctamente, si bien también se observa la necesidad de estudiar algunos casos, con un perfil muy claro (mayor número de reglas) en los que seguramente por falta de una configuración específica del algoritmo evolutivo, la calidad de los resultados no es tan buena, y por tanto, este particular será materia de estudio futuro.

Como se ha visto, en este trabajo se parte del uso del clásico método de WM para aprender las RBs. Este método es muy sencillo e incluso podría decirse que "*muy malo*" en cuanto a precisión y compacidad de la BR aprendida comparativamente con otros métodos, algunos de ellos evolutivos, aparecidos en la literatura durante varias décadas. Sin embargo, queremos hacer notar que lo utilizamos intencionadamente porque el modelo evolutivo empleado posteriormente precisamente permite corregir el principal defecto de las reglas aprendidas siguiendo el método de WM, que es el de la falta de cooperación entre ellas, lo cual es fundamental en regresión difusa. El modelo evolutivo empleado no sólo *gradúa* la contribución de cada regla a través del parámetro o peso, sino que busca la cooperación seleccionado las reglas más apropiadas de las encontradas por el método de WM, produciéndose así una BR final de considerable mayor calidad, más precisa y compacta, sin necesidad de usar un modelo más complejo y de mejor calidad en la primera fase. De cualquier modo, y queda pendiente para un futuro, realizar un estudio sobre cómo afecta la calidad del modelo inicial previo al post-procesamiento evolutivo distribuido propuesto empleando otras alternativas al método de WM, aunque *a priori*, por lo expuesto anteriormente, creemos que será escasa.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Educación y Ciencia dentro del Proyecto TIN2017-89517-P.

Referencias

[1] J. Alcala-Fdez, F. Herrera, F.A. Márquez, and A. Peregrín, "Increasing fuzzy rules cooperation based on evolutionary adaptive inference systems", Int J Intell Syst. 2007;22(9), pp. 1035–1064.

[2] F.A. Márquez, A. Peregrín, and F. Herrera, "Cooperative evolutionary learning of linguistic fuzzy rules and parametric aggregation connectors for Mamdani fuzzy system", IEEE Trans on Fuzzy Sys. 2007;15(6), pp.1168–1178.

[3] F. Herrera, "Genetic fuzzy systems: taxonomy, current research trends and prospects",. Evol Intell. 2008;1(1), pp.27–46.

[4] M. Fazzolari, R. Alcalá, Y. Nojima, H. Ishibuchi, and F. Herrera, "A review of the application of multi-objective evolutionary systems: Current status and further directions", IEEE Trans Fuzzy Syst. 2013; 21(1), pp.45–65.

[5] A. Fernández, V. López, M.J. del Jesus, and F. Herrera, "Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges", Knowl Based Syst. 2015;80, pp.109–121.

[6] J.S. Cho, and D.J. Park, "Novel fuzzy logic control based on weighting of partially inconsistent rules using neural network", J Intel Fuzzy Syst. 2000;8, pp.99–100.

[7] A. Márquez, F.A. Márquez, and A. Peregrín, "A Mechanism to Improve the Interpretability of Linguistic Fuzzy Systems with Adaptive Defuzzification based on the use of a Multi-objective Evolutionary Algorithm", Int. J. of Comp. Intell. Syst., 2012; 5 (2), pp. 297–321.

[8] D. Laney, "3D data management: controlling data volume, velocity and variety", META Group Research Note 6. 2001; 70.

[9] A. Fernández, S. del Río, V. López, A. Bawakid, M.J. del Jesus, J.M. Benítez, and F. Herrera, "Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks", Wiley Interdiscip. Rev. Data Mining Knowl. Discovery. 2014;4(5), pp.380–409.

[10] T. White, "Hadoop: The Definitive Guide", O'Reilly Media, Inc.; 2012.

[11] J. Dean, and S. Ghemawat, "MapReduce: a flexible data processing tool", Commun ACM. 2010; 53(1), pp.72–77.

[12] G. Malewicz, M.H. Austern, A.J. Bik, J.C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: a system for large-scale graph processing", In Proceedings of the ACM SIGMOD International Conference on Management of Data. 2010, pp. 135–146.

[13] M. Zaharia, R.S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, et al., "Apache spark: a unified engine for big data processing", Commun ACM. 2016; 59(11), pp.56–65.

[14] L. Wang and J. Mendel, "Generating fuzzy rules by learning from examples", IEEE Trans Syst, Man, Cybern. 1992;22(6), pp.1414–1427.

[15] A.A. Márquez , F.A. Márquez, and A. Peregrín, "A scalable evolutionary linguistic fuzzy system with adaptive defuzzification in big data", In Proceedings of the IEEE International Conference on Fuzzy System (FUZZ-IEEE). 2017; pp.1–6.

[16] L.J. Eshelman, "The CHC adaptive search algorithm: how to safe search when engaging in nontraditional genetic recombination", in G.J.E. Rawlings (Ed.), Foundations of genetic algorithms. 1991, pp.265–283.

[17] F. Herrera, M. Lozano, and A. Sánchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study", Int J Intell Syst. 2003;18, pp.309–338.

[18] J. Alcala-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J. Garrell, et al., "Keel: a software tool to assess evolutionary algorithms for data mining problems", Soft Comput. 2009;13(3), pp.307–318.

[19] J. Demšar, "Statistical comparisons of classifiers over multiple data sets", J Mach Learn Res. 2006;7, pp.1–30.

[20] S. García and F. Herrera, "An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons", J Mach Learn Res. 2008;9, pp.2579–2596.