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.

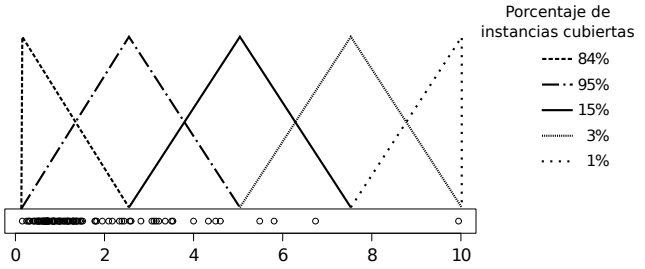

Figura 1. Particiones difusas con distribución equiespacial

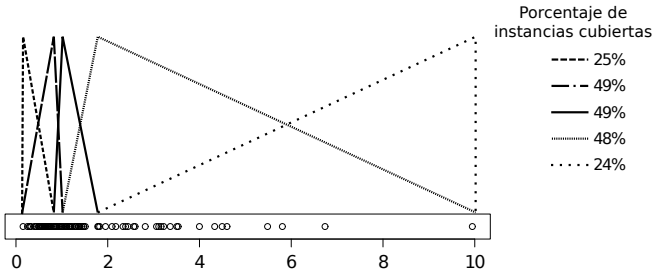

Figura 2. Particiones difusas con distribución equifrecuencial

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.

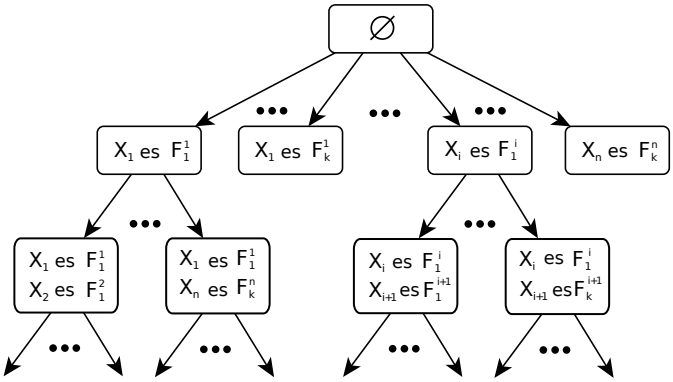

Figura 3. Estructura general del árbol de reglas candidatas

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

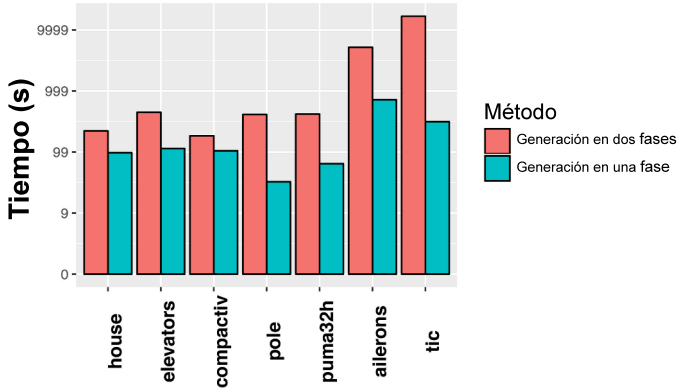

Figura 4. Comparativa de tiempos para ambos métodos en escala logarítmica

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.