EXTRACCIÓN DE BASES DE REGLAS SIMPLES Y LINGÜÍSTICAMENTE INTERPRETABLES

A. Gersnoviez[1], I. Baturone[1], F. J. Moreno-Velo[2]

[1] Instituto de Microelectrónica de Sevilla - Centro Nacional de Microelectrónica - CSIC
Avda. Reina Mercedes s/n, (Edif. CICA), E-41012, Sevilla, Spain.

[2] Dpto. Ingeniería Electrónica, de Lenguajes Informáticos y Automática,
Universidad de Huelva, Huelva, Spain.

Proc. XIII Congreso Español de Tecnologías y Lógica Fuzzy (ESTYLF 2006), pp. 111-116, Ciudad Real, Septiembre 20-22, 2006.

EXTRACCIÓN DE BASES DE REGLAS SIMPLES Y LINGÜÍSTICAMENTE INTERPRETABLES

Andrés A. Gersnoviez[1] **Iluminada Baturone**[1] **Francisco J. Moreno-Velo**[2]
[1] Instituto de Microelectrónica de Sevilla, (IMSE-CNM) y Dept. Electrónica y Electromagnetismo, Univ. de Sevilla (España).
e-mail: {andres, lumi}@imse.cnm.es
[2] Dept. Ing. Electrónica, de Lenguajes Informáticos y Automática, Univ. de Huelva (España).
e-mail: velo@diesia.uhu.es

Resumen

Este artículo presenta una técnica basada en la lógica difusa para extraer bases de reglas a partir de datos numéricos. Permite obtener bases de reglas interpretables lingüísticamente a la vez que simples en cuanto a número de reglas, sencillez en las partes antecedentes y consecuentes y facilidad de implementación hardware/software. Los pasos más significativos de esta técnica son los siguientes: (1) extracción de la base de reglas empleando particiones granulares de las variables del problema, (2) ajuste de las funciones de pertenencia para las variables de salida y posterior simplificación, (3) simplificación tabular de la base de reglas y (4) simplificación de las funciones de pertenencia para las variables de entrada. La técnica puede aplicarse de forma automática mediante las herramientas de CAD integradas en el entorno Xfuzzy 3. Se incluye un ejemplo de aplicación en robótica móvil para ilustrar las ventajas de la técnica propuesta.

Palabras Clave: Extracción de conocimiento, simplificación, interpretabilidad lingüística.

1. INTRODUCCIÓN

Las técnicas dedicadas al análisis y extracción de información a partir de datos numéricos son cada vez más necesarias. Dentro de estas técnicas, cobran gran interés las de lógica difusa, debido a su capacidad de extraer conocimiento lingüístico de fácil comprensión por personas no expertas. Existen dos tipos de estrategias a la hora de extraer información de una base de datos numérica: una es mediante métodos de agrupamiento o *clustering* y otra mediante métodos basados en particiones granulares (tipo *grid*) de las variables del problema.

Los métodos de clustering organizan los datos numéricos en clusters y usan éstos para crear las reglas. Cada cluster se transforma en una regla mediante su proyección en cada dimensión de las variables de entrada [9]-[5]. Por tanto, toda la descripción de la base de reglas se obtiene simultáneamente. La extracción de información mediante clustering tiene la ventaja de conseguir un número bajo de reglas, pero como contrapartida, sus reglas no tienen apenas significado lingüístico. Diversas aproximaciones han sido propuestas en la literatura para incrementar el significado lingüístico de estas bases de reglas. Una de ellas es el uso de modificadores lingüísticos, que permiten reducir tanto el número de funciones de pertenencia (muchas de ellas pueden ser obtenidas como modificaciones de otras) como el de las reglas (algunas de ellas pueden combinarse en otra más genérica) [9]. Otros autores se centran en simplificar las funciones de pertenencia obtenidas para conseguir un sistema difuso más transparente e interpretable [8].

Por otro lado, los métodos tipo grid generan una partición de los espacios de las entradas y las salidas antes de crear la base de reglas [7]. Las reglas se obtienen seleccionando las etiquetas más adecuadas para las entradas y las salidas respecto a la base de datos numérica. Estos métodos consiguen reglas con mucho más significado lingüístico, pero con el coste de obtener un número de reglas mucho mayor. Para evitar este número elevado de reglas, algunas soluciones reportadas en la literatura han sido seleccionar las reglas más significativas [1]-[11] o aplicar algoritmos de minimización inspirados en el diseño Booleano [10].

Otra diferencia significativa entre ambos métodos es que los de clustering suelen emplear funciones de pertenencia gaussianas, lo cual dificulta sobremanera su síntesis en hardware dedicado o software empotrado, no siendo así para los métodos tipo grid, que pueden utilizar funciones trapezoidales y triangulares mucho más fáciles de implementar.

La técnica que se presenta en este artículo trata de obtener bases de reglas con las ventajas de las dos técnicas anteriormente comentadas, es decir, que sean fácilmente interpretables de forma lingüística a la vez que simples, en el sentido de que el número de reglas sea bajo, las partes antecedentes y consecuentes de las reglas sean sencillas y de que su implementación sea eficiente en un hardware o software con pocos recursos. Para ello se emplean técnicas tipo grid como primer paso para extraer bases de reglas interpretables y se les aplican métodos de simplificación que permiten reducir no sólo el número de reglas sino también el de las funciones de pertenencia que se contemplan en las distintas variables.

El artículo se estructura de la siguiente manera. La sección 2 resume métodos de simplificación de bases de reglas difusas y explica los métodos que se emplean en la técnica propuesta: simplificación tabular de la base de reglas y simplificación de las funciones de pertenencia de antecedentes y consecuentes. La sección 3 describe brevemente cómo se puede realizar automáticamente la técnica propuesta mediante las herramientas de CAD del entorno Xfuzzy 3. En la sección 4 se muestra un ejemplo de aplicación en robótica móvil. Por último, las conclusiones se incluyen en la sección 5.

2. TÉCNICA PROPUESTA

En la técnica que proponemos el primer paso es extraer una base de reglas lingüísticamente interpretable empleando particiones granulares de las variables del problema. A continuación se simplifican las funciones de pertenencia para las variables de salida para proceder a una simplificación tabular de la base de reglas (tanto más sencilla cuanto menor sea el número de funciones de pertenencia de los consecuentes). Por último, se simplifican las funciones de pertenencia para las variables de entrada. Los métodos de simplificación tabular y de funciones de pertenencia se describen a continuación.

2.1. SIMPLIFICACIÓN TABULAR

El método de simplificación que empleamos en nuestra técnica es una expansión del algoritmo de Quine-McCluskey del diseño Booleano [3]. Este método ya ha servido de inspiración a algunos autores para obtener implementaciones hardware eficientes [10]. En nuestro caso, hemos expandido este método para obtener bases de reglas más simples (por lo tanto, también más fáciles de implementar en hardware) y con más contenido lingüístico. Para ilustrar nuestra propuesta consideremos la base de reglas tipo grid de dos entradas ilustrada en la Figura 1.

Lo primero a tener en cuenta es que los consecuentes de una base de reglas difusa no son bi-valuados, sino que toman varios valores (N, NS, Z, PS y P en nuestro ejemplo). Por tanto, la simplificación tabular se aplica a cada función de pertenencia de los consecuentes que aparezcan en la base de reglas (aunque, para el caso de r funciones, pueden hacerse r-1 simplificaciones usando la condición "en otro caso" para la r-ésima función). Por eso interesa haber simplificado con anterioridad el número de funciones de pertenencia de los consecuentes. Para cada función de pertenencia de consecuentes, los pasos a seguir son los siguientes:

x1 \ x2	(1) MP	(2) P	(3) M	(4) G	(5) MG
(1) MP	Z	Z	Z	Z	Z
(2) P	PS	PS	Z	NS	NS
(3) M	Z	Z	Z	Z	Z
(4) G	N	N	Z	P	P
(5) MG	N	N	Z	P	P

Figura 1: Ilustración de la simplificación tabular.

Paso 1: Se listan en una columna todos los antecedentes de las reglas con esa función de pertenencia en su consecuente. Los antecedentes tampoco son bi-valuados, sino que están relacionados normalmente con varios conjuntos difusos (MP, P, M, G y MG, en nuestro ejemplo). En lugar de trabajar con códigos binarios, asignamos números naturales ordenados a los conjuntos difusos de cada variable de entrada (por ejemplo, 1 a MP, 2 a P, 3 a M, 4 a G y 5 a MG). Los antecedentes que sumen el mismo valor se agrupan y se listan en orden creciente de sumas. Esto se muestra en la Figura 2 para nuestro ejemplo. En la columna situada más a la izquierda (primera lista), las 13 reglas (en forma tabular) asociadas con el consecuente Z están ordenadas en 7 grupos (aquellos cuyos antecedentes suman 2, 3, 4, 5, 6, 7 y 8).

Paso 2: Se realiza una búsqueda exhaustiva entre los grupos vecinos para combinar los mintérminos adyacentes: aquellos cuya suma difiere en la unidad y comparten las mismas funciones de pertenencia para cada variable de entrada excepto una que es consecutiva. Los implicantes obtenidos engloban u+1 funciones de pertenencia, siendo u el número de entradas. Esto se muestra en la segunda lista de la Figura 2. Se repite el proceso para cada lista, combinando implicantes con u+1 funciones para obtener implicantes con u+2 y así sucesivamente hasta que no puedan combinarse más implicantes. El resultado final es una lista de implicantes primos (quinta lista de la Figura 2, que se corresponden con los grupos sombreados en la Figura 1).

Paso 3: Se selecciona un número mínimo de implicantes primos que cubran todos los mintérminos de la función a

x_1	x_2		x_1	x_2		x_1	x_2		x_1	x_2		x_1	x_2
1	1	√	1	1,2	√	1	1,2,3	√	1	1,2,3,4	√	1	1,2,3,4,5
1	2	√	1	2,3	√	1	2,3,4	√	1	2,3,4,5	√	3	1,2,3,4,5
1	3	√	1	3,4	√	1	3,4,5	√	3	1,2,3,4	√	1,2,3,4,5	3
3	1	√	1,2	3	√	1,2,3	3	√	1,2,3,4	3	√		
1	4	√	1	4,5	√	2,3,4	3	√	2,3,4,5	3	√		
2	3	√	2,3	3	√	3	2,3,4	√	3	2,3,4,5	√		
3	2	√	3	2,3	√	3	3,4,5	√					
1	5	√	3	3,4	√	3,4,5	3	√					
3	3	√	3,4	3	√								
3	4	√	3	4,5	√								
4	3	√	4,5	3	√								
3	5	√											
5	3	√											

Figura 2: Tabla de minimización que ilustra la simplificación tabular.

simplificar, es decir, se busca la regla más simple asociada con un consecuente. Para ello se sigue el siguiente procedimiento: (1) inicializar a cero el conjunto de mintérminos cubiertos y seleccionar la lista situada más a la derecha (la lista de implicantes primos) de la tabla de minimización. (2) De la lista seleccionada, elegir los implicantes que cubran el mayor número de mintérminos no cubiertos para formar la regla más simple, borrarla de la lista, y pasar a (3). Si no se encuentras implicantes, entonces pasar a (4). (3) Eliminar aquellos implicantes (que están aún seleccionados para formar la regla más simple) que están cubiertos por el nuevo implicante que se ha incluido. Entonces pasar a (2). (4) Seleccionar de la tabla de minimización la lista inmediatamente anterior a la última analizada y pasar a (2). El procedimiento finaliza cuando todos los mintérminos asociados con el consecuente están cubiertos. En el ejemplo de las Figuras 1 y 2, se puede ver cómo los 3 implicantes primos de la última lista cubren todos los mintérminos.

Paso 4: Se usan modificadores lingüísticos para expresar la regla resultante de una forma simple y expresiva. En particular, usamos los modificadores lingüísticos 'mayor o igual' y 'menor o igual'. Por ejemplo, para un implicante x1(1), x2(2,3,4), la parte del antecedente se expresaría como sigue:

Si (x1 *es* MP *y* (x2 es *mayor o igual que* P *o menor o igual que* G)) (3)

Si todas las funciones de pertenencia de una variable están cubiertas por el implicante primo que aparece en la regla, esa variable se elimina de la parte del antecedente porque su valor no importa. Por ejemplo, para un implicante x1(1), x2(1,2,3,4,5), la parte del antecedente se expresaría como sigue:

Si (x1 *es* MP) (4)

Si todas las funciones de pertenencia de una variable están cubiertas por el implicante primo excepto una, se usa el modificador lingüístico 'distinto de'. Por ejemplo, para el implicante x1(1), x2(1,2,4,5), la parte del antecedente se expresaría como:

Si (x1*es* MP *y* x2 *es distinto de* M) (5)

Si el límite más bajo (más alto) de una agrupación es la primera (última) función de pertenencia, dicha condición puede ser eliminada de la parte antecedente. Por ejemplo, para el implicante x1(4,5), x2(1,2), la parte del antecedente asociado se expresa como:

Si (x1 es *mayor o igual que* G *y* x2 es *menor o igual que* P) (6)

Por tanto, las 25 reglas (en forma tabular) mostradas en la Figura 1, quedan simplificadas en las siguientes 5 reglas:

Si (x1 *es* MP *o* x1 *es* M *o* x2 *es* M), *entonces* y = Z;
Si (x1 *es* P *y* x2 es *menor o igual que* P), *entonces* y = PS;
Si (x1 *es* P *y* x2 *es mayor o igual que* G), *entonces* y = NS; (7)
Si (x1 *es mayor o igual que* G *y* x2 *es menor o igual que* P), *entonces* y = N;
Si (x1 *es mayor o igual que* G *y* x2 *es mayor o igual que* G), *entonces* y = P;

2.2. SIMPLIFICACIÓN DE FUNCIONES DE PERTENENCIA

Los métodos más ampliamente usados para simplificar funciones de pertenencia son los de clustering, unión por similitud y purga.

Los métodos de clustering buscan un número reducido de clusters (prototipos de funciones de pertenencia) en los

que agrupar varias funciones originales. Los clusters se encuentran en el espacio de los parámetros que definen las funciones de pertenencia. El número óptimo de clusters puede ser encontrado automáticamente aplicando índices de validez, como son el Índice de Separación de Dunn, el Índice de Davies-Bouldin y los Índices Generalizados de Dunn [2].

Los procesos de unión por similitud buscan el par de funciones más similares y las remplaza por una función única si el grado de similitud está por encima de un valor umbral. El proceso termina cuando no pueden unirse más funciones. Mientras el umbral sea más bajo, mayor será el número de funciones unidas [8].

En nuestra técnica aplicamos métodos de clustering o similitud para simplificar las funciones de pertenencia de los consecuentes, puesto que, al aplicar métodos de ajuste sobre ellas, lo normal es que queden varias muy similares. Este paso es previo a la simplificación tabular.

El mecanismo de simplificación por purga busca aquellas funciones de pertenencia que no están siendo usadas en la base de reglas y las elimina. En nuestra técnica aplicamos este mecanismo tras realizar la simplificación tabular para simplificar las funciones de pertenencia de los antecedentes. En el ejemplo de la Figura 1, este mecanismo reduce las funciones de pertenencia de la variable x1 a 4 (MP, P, M, G) y las de x2 a 3 (P, M, G).

3. AUTOMATIZACIÓN DE LA TÉCNICA

La técnica propuesta puede realizarse automáticamente mediante las herramientas de CAD del entorno Xfuzzy 3, desarrollado en el Instituto de Microelectrónica de Sevilla [4]. La extracción de la base de reglas tipo grid puede realizarse con la herramienta *xfdm* [6] y el ajuste de las funciones de pertenencia de los consecuentes aplicando técnicas de aprendizaje supervisado puede llevarse a cabo mediante la herramienta *xfsl* [4]. Para aplicar el método de simplificación tabular y los métodos de simplificación de funciones de pertenencia se ha desarrollado la herramienta *xfsp*. Dicha herramienta permite aplicar métodos de simplificación por purga, clustering y unión por similitud para las funciones de pertenencia (de antecedentes o consecuentes) (Figura 3) y métodos de compresión-expansión y podado, además de simplificación tabular, para las bases de reglas.

4. EJEMPLO PRÁCTICO

Aunque la técnica se ha probado con bases de datos para aprendizaje automático disponibles en el repositorio UCI [12], nuestro interés actual se centra en utilizar esta técnica para diseñar sistemas de control empotrado en

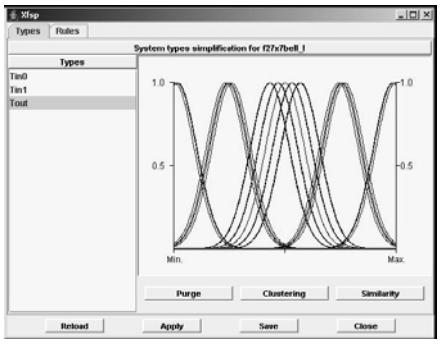

Figura 3: Ventana principal de *xfsp* para simplificar funciones de pertenencia.

robótica móvil. Por eso el ejemplo que incluimos está relacionado con el problema de navegación evitando obstáculos de un robot autónomo de tipo "coche". Para la detección del obstáculo, el robot posee un láser en su parte delantera que hace barridos de 0 a 180º. Dentro de los numerosos bloques destinados a las distintas tareas de control del robot, uno de ellos estudia si el obstáculo en cuestión es evitable, con lo cual se procede a los cambios pertinentes en la trayectoria para evitarlo, mientras que si es inevitable, el robot debe parar.

Utilizando conocimiento heurístico, lo normal es que aplicáramos sólo dos reglas difusas: "Si el obstáculo está muy cerca, detente" y "Si el obstáculo está muy lejos, sigue adelante". Pero estas reglas son demasiado simples para una navegación eficiente.

Otra opción es estudiar cinemáticamente el problema, estando centrado en el eje trasero del coche el origen de coordenadas referidas al vehículo, y añadiendo restricciones debidas a los círculos de curvatura máxima que pueden llevarse a cabo en una dirección u otra. La ecuación más simplificada a la que se ha llegado para describir el comportamiento de un obstáculo inevitable es:

$$h^2 + 2h\left[d\left|\sin(\phi_m)\right| + R_{\min}\left|\cos(\phi_m)\right|\right] < \\ < 4\left(R_{\min}+1\right) - d^2 \qquad (7)$$

Donde h es la distancia desde el punto más cercano del obstáculo al láser y ϕ_m es el ángulo que marca el láser para dicho punto. R_{min} es el radio mínimo perteneciente a la circunferencia de curvatura máxima del robot, mientras que d es la distancia entre el láser y el eje trasero del coche (Figura 4).

Con el análisis cinemático no tenemos apenas información lingüística sobre lo que tendría que hacer el robot, es decir, la ecuación (7) no explica lingüísticamente lo que debe hacer el robot. Nuestro objetivo ha sido traducir esa ecuación a una base de reglas simple y lingüísticamente interpretable. Para ello hemos generado

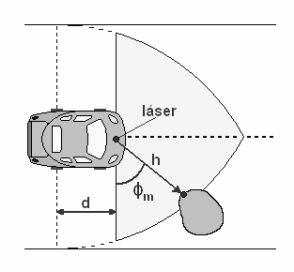

Figura 4: Zona de obstáculo inevitable.

un fichero con los datos numéricos que representan el análisis cinemático. Aplicando nuestra técnica, hemos empleado la herramienta *xfdm* incluida en Xfuzzy 3 para extraer una base de reglas tipo grid. En concreto, hemos elegido el método de Wang y Mendel, con 20 funciones de pertenencia para cada entrada, con lo que se genera un sistema con 400 reglas. Las funciones de pertenencia cubren las entradas como se muestra en la Figura 5, siendo el rango de valores para *h* entre 0 y 3m, mientras que para ϕ_m es de 0 a 180º. El comportamiento del sistema se puede visualizar mediante la herramienta *xfplot* de Xfuzzy 3 tal y como puede apreciarse en la Figura 6, donde la zona más oscura corresponde a los obstáculos inevitables (y, por tanto, la orden de control es parar el coche) y la más clara a los obstáculos evitables. En este problema, los consecuentes de las reglas sólo toman dos valores 'parar' (v = cero) o 'seguir adelante' (v = uno), por lo que no es necesario aplicar ajuste ni simplificación de consecuentes.

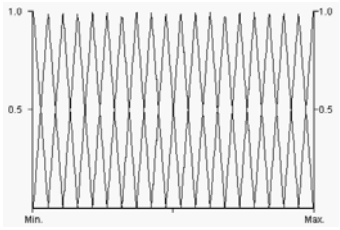

Figura 5: Entradas del sistema difuso.

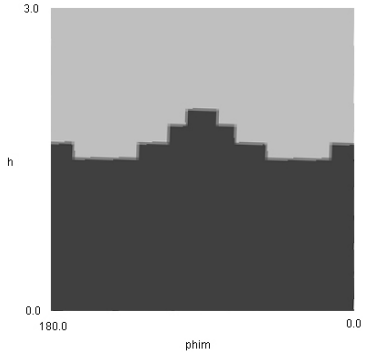

Figura 6: Salida del sistema difuso con 400 reglas.

El siguiente paso es aplicar sobre dicho sistema el método de simplificación tabular con la ayuda de *xfsp*. El resultado da lugar a un sistema de dos reglas expresadas de forma comprimida: una regla para los puntos de obstáculo inevitable (v = cero), y otra para los puntos de obstáculo evitable (v = uno). Aprovechando que el problema está formado por una base de reglas completa y, además, es bi-valuado, podemos quedarnos con la regla correspondiente a los puntos de obstáculo inevitable y su negada. Hecho esto, si expandimos la base de reglas obtenemos como resultado lo expuesto en la Tabla 1. Se puede apreciar cómo de un problema de 400 reglas se pasa a otro de 7, mucho más simple.

Lo siguiente es aplicar el método de purga sobre las funciones de pertenencia de las entradas. Las entradas *h* y

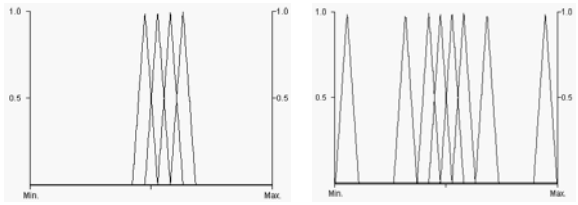

Figura 7: Entradas *h* y ϕ_m tras la simplificación.

Tabla 1: Reglas obtenidas tras la simplificación del sistema difuso

Si	h *es menor o igual que* aprox143cm	v = cero
Si	ϕ_m *es menor o igual que* aprox9grad y h *es menor o igual que* aprox158cm	v = cero
Si	ϕ_m *es mayor o igual que* aprox170grad y h *es menor o igual que* aprox158cm	v = cero
Si	ϕ_m *es mayor o igual que* aprox57grad y *es menor o igual que* aprox123grad y h *es menor o igual que* aprox158cm	v = cero
Si	ϕ_m *es mayor o igual que* aprox76grad y *es menor o igual que* aprox104grad y h *es menor o igual que* aprox174cm	v = cero
Si	ϕ_m *es mayor o igual que* aprox85grad y *es menor o igual que* aprox95grad y h *es menor o igual que* aprox189cm	v = cero
Si no		v = uno

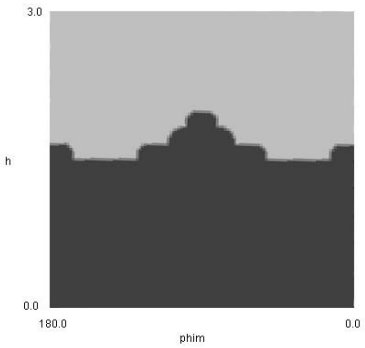

Figura 8: Salida del sistema difuso simplificado

ϕ_m pasan de tener 20 funciones de pertenencia cada una, a tener 5 y 8 respectivamente, tal y como puede verse en la Figura 7. El comportamiento del sistema final simplificado se puede apreciar en la Figura 8, dando lugar a un resultado más suave que el original.

Analizando las reglas de la Tabla 1 y comparando con lo observado en la Figura 8, podemos comprobar la interpretación lingüística de la base de reglas obtenida:
- Si el obstáculo está cerca, párate.
- Si el obstáculo está un poco más lejos, pero está muy a la izquierda, o muy a la derecha o en gran parte de la zona central frente al robot, párate.
- Si el obstáculo está bastante más lejos pero bastante centrado frente al robot, párate.
- Si el obstáculo está mucho más lejos pero situado justo frente al robot, entonces párate.
- En otro caso, sigue adelante.

Utilizando una técnica de clustering para extraer la base de reglas (por ejemplo el algoritmo Incremental Clustering [6]) y simplificando las funciones de pertenencia, el número de reglas final, así como el número de funciones de pertenencia totales y el comportamiento del sistema, son similares a los obtenidos por la técnica propuesta, salvo que con el clustering se pierde toda interpretabilidad lingüística y el sistema obtenido es más costoso de implementar.

5. CONCLUSIONES

Partiendo de una base de reglas tipo grid, gracias a la gran eficacia del método de simplificación tabular y los métodos de simplificación de funciones de pertenencia, se obtienen sistemas con un número de reglas y un número de funciones de pertenencia equiparables a los que se obtienen con métodos de clustering, pero manteniendo las ventajas en cuanto a interpretabilidad lingüística de las bases de reglas tipo grid. Si se emplean funciones de pertenencia triangulares o trapezoidales el sistema resultante es, además, muy eficiente para ser implementado en hardware dedicado o software empotrado.

Referencias

[1] D. Nauck, “Data Analysis with Neuro-Fuzzy Methods”, PhD. Dissertation, Univ. of Magdeburg, Faculty of Computer Science, Alemania, 2000.

[2] E. H. Ruspini, P. P. Bonissone, W. Pedrycz, Eds., “Handbook of Fuzzy Computation”, Institute of Physics Pub., 1998

[3] E. J. McCluskey Jr., “Introduction to the Theory of Switching Circuits”, McGraw-Hill Book Co., Nueva York, 1965.

[4] F. J. Moreno-Velo, I. Baturone, F. J. Barrio S. Sánchez-Solano, A. Barriga, “A Design Methology for Complex Fuzzy Systems”, Proc. 3rd European Symp. on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems, EUNITE’2003, Oulu, Julio 2003.

[5] F. Klawonn, R. Kruse, “Constructing a Fuzzy Controller from Data”, *Fuzzy Sets and Systems*, 85, pp. 177-193, 1997.

[6] I. Baturone, F. J. Moreno-Velo, A. Gersnoviez, “Identifying fuzzy systems from numerical data with Xfuzzy”, Proc. 4th Conference of the European Society for Fuzzy Logic and Technology, EUS-FLAT’2005, pp. 1257-1262, Barcelona , Sept. 2005.

[7] L. Wang, J. M. Mendel, “Generation Rules by Learning from Examples”. *Proc. Int Symp. on Intelligent Control*, pp. 263-268, 1991.

[8] M. Setnes, R. Babuska, U. Kaymak, H. R. van Nauta Lemke, “Similarity measures in fuzzy rule base simplification”, *IEEE Trans. on Systems, Man and Cybernetics, part B*, vol. 28, no. 3, pp. 376-386, Junio 1998.

[9] M. Sugeno, T. Yakusawa, “A Fuzzy-Logic-Based Aproach to Qualitative Modeling”, *IEEE Trans. On Fuzzy Systems*, Vol. 1, Nº 1, pp.7-31, Feb. 1993.

[10] R. Rovatti, R. Guerreri, G. Baccarani, “An enhanced two-level Boolean synthesis methology for fuzzy rules minimization”, IEEE Trans. on Fuzzy Systems, vol. 3, no 3, pp. 288-299, Aug. 1995.

[11] R. Senhadji, S. Sánchez-Solano, A. Barriga, I. Baturone, F. J. Moreno-Velo, “NORFREA: An Algorithm for non-redundant fuzzy rule extraction”, *Proc. IEEE SMC’2002*, vol. 1, pp. 604-608, Tunisia, Oct. 2002.

[12] UCI Repository of Machine Learning Databases, http://www.ics.uci.edu/pub/machine-learning-databases.