Departamento de Electrónica y Electromagnetismo

Universidad de Sevilla

Diseño e Implementación de Sistemas Neuro-Difusos Simplificados

Memoria presentada para optar al grado de Doctor por

Andrés A. Gersnoviez Milla

CENTRO NACIONAL DE MICROELECTRONICA IMSE

CSIC
Consejo Superior de Investigaciones Científicas

Departamento de Electrónica y Electromagnetismo

Universidad de Sevilla

Diseño e Implementación de Sistemas Neuro-Difusos Simplificados

Memoria presentada para optar al grado de Doctor por

Andrés A. Gersnoviez Milla

Dirigida por: Dra. MªIluminada Baturone Castillo

Sevilla, Junio 2015

CENTRO NACIONAL DE MICROELECTRONICA IMSE

CSIC
Consejo Superior de Investigaciones Científicas

Agradecimientos

Esta Tesis ha sido financiada, en parte, por el programa FPU de la Dirección General de Universidades del Gobierno Español y por los proyectos de investigación nacionales TEC2005-04359/MIC, TEC2008-04920, TEC2011-24319, y regionales EXC/2005/TIC-635, P09-TEP-4479, subvencionados todos con fondos FEDER.

Resumen

Esta Tesis se centra principalmente en la búsqueda y desarrollo de nuevas técnicas de simplificación para sistemas neuro-difusos, de modo que sistemas complejos puedan ser ejecutados de forma eficiente en software empotrado o hardware dedicado. Los algoritmos de extracción de reglas de tipo rejilla (*grid*) a partir de datos numéricos pueden dar lugar a sistemas que permitan una implementación eficiente y que estén dotados de interpretabilidad lingüística, pero, en general, podrán sufrir de *"la maldición de la dimensionalidad"*, por lo que son necesarios métodos que mitiguen el efecto de esa maldición.

La primera técnica desarrollada en la Tesis está basada en una simplificación tabular inspirada en el algoritmo de Quine-McCluskey de la lógica booleana. Se trata de un método bastante potente a la hora de disminuir el número de reglas que sirve, a su vez, para plantear una nueva técnica de clasificación de reglas. Esta nueva técnica da lugar a un método para el diseño de clasificadores neuro-difusos con bases de reglas incompletas, consiguiendo sistemas muy simples, que ofrecen un comportamiento mejor que el de otros sistemas con un número mayor de reglas.

La segunda técnica desarrollada en esta Tesis presenta un nuevo método de descomposición jerárquica de sistemas neuro-difusos, que consigue una simplificación bastante superior a la de otros métodos aparecidos en la literatura. Los sistemas resultantes no sólo son muy sencillos sino que, bajo ciertas condiciones, presentan un comportamiento lineal a tramos que permite aproximar comportamientos óptimos no lineales. El mayor obstáculo que deben superar los métodos de descomposición jerárquica es encontrar la estructura en la que descomponer el sistema. Sin embargo, en la Tesis se propone una metodología para encontrar dicha estructura, dando lugar a una técnica que consigue sistemas altamente simplificados, pero manteniendo los comportamientos de referencia.

Las distintas técnicas desarrolladas en la Tesis se prueban sobre campos de aplicación muy distintos, como son el de reconocimiento de patrones, el de control y el de robótica móvil autónoma, obteniendo resultados relevantes en cada uno de ellos.

Prólogo

El trabajo de investigación de esta Tesis se ha centrado en combinar las técnicas de la lógica difusa junto con las de sistemas neuronales para diseñar sistemas neuro-difusos. La mayoría de los trabajos reportados en la literatura sobre sistemas neuro-difusos no están centrados en su implementación, por lo que no llegan a realizaciones hardware o software eficientes. Por otro lado, la mayoría de los sistemas neuro-difusos pierden la interpretabilidad lingüística del paradigma difuso tras aplicar técnicas de aprendizaje neuronales. El objetivo fundamental en la realización de esta Tesis ha sido diseñar sistemas neuro-difusos lo más simples posibles para ser implementados eficientemente en hardware dedicado o software empotrado pero sin que la faceta *"neuro"* domine sobre la *"difusa"*, es decir, sin perder que el sistema resultante sea lingüísticamente interpretable. La interpretabilidad lingüística es una gran ventaja, ya que de ese modo se puede comprender con mayor facilidad lo que el sistema está haciendo en cada momento.

Para el diseño de sistemas neuro-difusos, un paso fundamental es el de dominar técnicas dedicadas al análisis y extracción de información a partir de datos numéricos. Por ello, durante el desarrollo de esta Tesis, se estudian a fondo estas técnicas, que pueden dividirse entre las basadas en métodos de agrupamiento (o *clustering*) y las basadas en particiones de tipo rejilla (o *grid*) de los universos de discurso de las variables de entrada del problema.

Dentro de las dos opciones, se ha optado por las técnicas de tipo grid porque permiten una implementación más eficiente y facilitan una mejor interpretabilidad lingüística (en particular si utilizan funciones de pertenencia lineales a tramos para describir los conjuntos difusos en vez de las funciones gaussianas que se emplean habitualmente en los métodos de clustering).

El principal problema surge cuando se pretenden solucionar problemas complejos, ya que una de las principales desventajas presentadas por las técnicas de tipo grid, es que el número de reglas que se extraen crece exponencialmente con el número de entradas. Este es uno de los mayores problemas que aparecen en el diseño de sistemas neuro-difusos y que es conocido como *"la maldición de la dimensionalidad"*. Por eso, uno de los objetivos de esta Tesis es la búsqueda de técnicas de simplificación de bases de reglas

y de funciones de pertenencia que reduzcan este problema.

En particular, se explora en esta Tesis una técnica que puede simplificar los sistemas en gran medida y que es objeto de gran interés en el contexto científico actual: el uso de sistemas jerárquicos. La idea fundamental de la jerarquía es descomponer el sistema neuro-difuso principal en subsistemas más pequeños, cada uno con un número menor de entradas y, por tanto, con menos reglas. El aplicar jerarquía puede generar sistemas mucho más simples, pero su utilización no es nada trivial, ya que, de plantearse la posibilidad de aplicar jerarquía, aparece el problema de cómo elegir la arquitectura de bloques adecuada para el sistema y cómo aplicar una metodología que sistematice su diseño.

Para comprobar la adecuación y efectividad de las distintas técnicas de simplificación investigadas, se plantean diversos tipos de aplicaciones en campos distintos, como son el del reconocimiento de patrones, el de control y el de la robótica móvil autónoma.

La memoria se estructura en cuatro capítulos. En el capítulo 1, se hace una revisión de distintos factores relacionados con los sistemas neuro-difusos que son cruciales para el desarrollo de las técnicas de simplificación, así como de su posterior implementación microelectrónica. Por ello, se revisan las distintas técnicas de extracción de información a partir de datos numéricos aplicadas a sistemas neuro-difusos, así como las condiciones bajo las cuales los sistemas neuro-difusos se comportan como sistemas polinómicos a tramos, para terminar con una revisión del estado del arte sobre realizaciones hardware/software de este tipo de sistemas.

En el capítulo 2 se revisan distintas técnicas de simplificación tanto de los tipos de las funciones de pertenencia, como de las reglas de estos sistemas. En este ámbito, se propone una técnica de simplificación tabular, basada en el método de Quine-McCluskey para la lógica binaria, que se generaliza para la lógica difusa. Esta técnica, a su vez, servirá para proponer un método de clasificación de reglas según unos índices denominados *índices de cubrimiento*, que permitirá una metodología para el diseño de clasificadores neuro-difusos con bases de reglas incompletas. Esta técnica se probará en el ámbito del reconocimiento de patrones ya que, debido a la imprecisión y ambigüedad de muchos patrones, el método propuesto resulta muy apropiado. El objetivo es conseguir una descripción lingüística de los patrones de una forma similar a la que podría aportar cualquier persona (lo que claramente puede redundar en una mejora en la interacción hombre-máquina). De este modo se pueden conseguir sistemas neuro-difusos con bases de reglas reducidas para describir patrones que, a su vez, pueden formar parte de sistemas de reconocimiento que aprovechen su gran interpretabilidad lingüística.

El capítulo 3 está centrado en la técnica de simplificación mediante descomposición jerárquica de sistemas neuro-difusos. Tras realizar una revisión de los trabajos desarrollados en este campo, se plantea una descomposición jerárquica que supera la simplificación de las arquitecturas reportadas hasta el momento. Tras analizar las propiedades que pre-

sentan estos nuevos sistemas, se plantea una metodología para encontrar la estructura más adecuada a la hora de descomponer un sistema bajo estudio. Para corroborar el buen funcionamiento de la técnica, se aplica al diseño de controladores predictivos basados en modelo. Estos controladores, que son estudiados ampliamente por su eficiencia, se obtienen tras resolver iterativamente la optimización del control de una planta en un horizonte finito. Debido a la complejidad de los problemas tratados, el diseño de estos controladores para operación en tiempo real es muy costoso. El objetivo es aplicar la técnica de descomposición jerárquica para diseñar e implementar sistemas neuro-difusos para control predictivo que sean lo suficientemente simples como para operar en tiempo real, prestando especial atención a los posibles problemas de inestabilidad resultantes de la simplificación. Con el diseño de sistemas jerárquicos se pretende conseguir sistemas que ofrezcan soluciones mucho más simples, a la vez que demuestren un mejor comportamiento que otros sistemas más complejos.

En el capítulo 4 se desarrolla una aplicación destinada al diseño de un sistema neuro-difuso encargado de controlar un robot móvil terrestre para que pueda navegar evitando obstáculos por entornos no estructurados ni conocidos. La idea fundamental consiste en combinar conocimiento heurístico aproximado junto con un análisis cinemático y dinámico (también aproximado) que proporcione datos numéricos con los que aplicar técnicas neuronales. El objetivo del capítulo es utilizar las distintas técnicas de simplificación desarrolladas y analizadas en esta Tesis (fundamentalmente relacionadas con el diseño de sistemas jerárquicos y técnica de simplificación tabular) para obtener un controlador de navegación eficiente, con un número muy reducido de reglas que, además, sean lingüísticamente interpretables.

En todos los capítulos se hace uso de distintas herramientas de CAD del entorno Xfuzzy 3, que permiten automatizar las técnicas propuestas.

La memoria de la Tesis finaliza recopilando los resultados obtenidos, tras todo el estudio, en el apartado de conclusiones finales.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción al diseño e implementación de sistemas neuro-difusos

En 1965, Lofti Zadeh introdujo en el trabajo [Zadeh 65] los principios de la lógica difusa, donde los conceptos de verdadero o falso de la lógica binaria clásica pasan a ser una cuestión de grado. Los sistemas difusos emulan los mecanismos de razonamiento aproximado utilizados por el cerebro humano, almacenando la base de conocimiento de forma simbólica, pero procesándola mediante técnicas numéricas [Zadeh 73] [Mizumoto 82]. Esta base de conocimiento está formada por una serie de reglas de tipo *"si-entonces"* como las que presentaría un sistema experto convencional, pero utilizando términos lingüísticos similares a los usados en el lenguaje natural.

A lo largo de todos estos años, los formalismos matemáticos de la lógica difusa han ido desarrollándose gracias a las contribuciones de muchos autores [Tsukamoto 79] [Wang 97] [Pedrycz 98]. Algunas de estas contribuciones están relacionadas con aspectos teóricos generales, mientras que otras van encaminadas a ámbitos de aplicación específicos [Mendel 95] [Patyra 96].

Otra rama importante de los sistemas inteligentes es la correspondiente a las redes o sistemas neuronales artificiales. Estos sistemas persiguen reproducir la potencia y flexibilidad del cerebro humano no emulando su forma de razonar, como en el caso de los sistemas expertos, sino su estructura y organización [Haykin 94]. El cerebro humano y, en general, los sistemas nerviosos de los seres superiores, están formados por una gran cantidad de neuronas altamente interconectadas y cuya conexión cambia con la experiencia y el aprendizaje. Imitando estas características, los sistemas de cómputo neuronales artificiales combinan una gran cantidad de elementos simples de procesado

altamente interconectados que realizan un procesado masivamente paralelo. Sus propiedades computacionales, que dependen de las interconexiones, se desarrollan mediante un proceso adaptativo de aprendizaje. Esta estructura contrasta con la de un computador convencional cuya operación es secuencial y centralizada en un microprocesador relativamente complejo. A diferencia de los sistemas expertos, las redes neuronales artificiales procesan el conocimiento de forma numérica, por lo que pueden implementarse mediante circuitos electrónicos y dar lugar a computadores neuronales. Sin embargo, su mecanismo de procesado no maneja información simbólica por lo que su operación no es transparente o comprensible para el ser humano.

La gran ventaja de los sistemas neuronales es que poseen métodos sistemáticos de aprendizaje para su diseño sin necesidad de que un experto humano les proporcione el conocimiento estructurado.

Los métodos de aprendizaje típicos de un sistema neuronal parten de una serie de datos numéricos de entrenamiento, es decir, de un conjunto de valores de entrada y sus correspondientes salidas. Esto significa que emplean información numérica sobre la relación entrada-salida que debe reproducir el sistema neuronal. Estos métodos se denominan *métodos* o *algoritmos de aprendizaje supervisado*. Su objetivo es minimizar el comportamiento erróneo del sistema neuronal, que se cuantifica mediante una función error (como el error cuadrático medio) entre la salida que proporciona el sistema y la salida que debe proporcionar, para cada patrón de entrada.

Aunque las causas y los objetivos que motivaron la aparición de los sistemas difusos fueron diferentes de los de los sistemas neuronales artificiales, lo cierto es que se ha detectado una enorme relación entre los comportamientos de ambos sistemas. Por ello, la línea que han seguido muchos investigadores es integrar ambos paradigmas en los denominados *sistemas neuro-difusos* para así combinar las ventajas de unos y otros. Un sistema neuro-difuso es, básicamente, un sistema difuso (por lo tanto con conocimiento estructurado) que emplea métodos de aprendizaje tomados de los sistemas neuronales para ajustar sus parámetros [Jang 97]. Por tanto, estos sistemas engloban las ventajas que ofrecen los sistemas difusos, como son la traducción de conocimiento heurístico expresado lingüísticamente y de forma aproximada y/o incierta, la rapidez del diseño, la facilidad de comprensión del proceso de razonamiento (válido para un no experto), la facilidad de test, la robustez y la suavidad en la respuesta. Por otro lado, se pueden emplear los métodos de aprendizaje de los sistemas neuronales y así poder ajustar un sistema definido a partir de conocimiento heurístico, o bien, extraer bases de reglas a partir de datos numéricos (sin conocimiento heurístico).

Dentro de las distintas propiedades que tienen los sistemas neuro-difusos, en este capítulo se hará una revisión de aquellas que están más relacionadas con los contenidos desarrollados en esta Tesis. En primer lugar, una de las propiedades ya comentadas y más utilizadas en este trabajo, es la habilidad de estos sistemas de aprender a partir

de datos numéricos, por lo que el apartado 1.1 se encargará de realizar una revisión de distintas técnicas de minería de datos (*data mining*) enmarcadas en este tipo de sistemas.

Otra propiedad importante que será explotada en capítulos posteriores es la equivalencia de este tipo de sistemas con los sistemas polinómicos a tramos. Por ello, el apartado 1.2 describirá cómo y bajo qué condiciones los sistemas neuro-difusos se comportan como sistemas polinómicos a tramos.

Por último, unos de los objetivos de esta Tesis es la simplificación de sistemas neuro-difusos para conseguir una implementación microelectrónica eficiente. Por esa razón, el capítulo finaliza con el apartado 1.3, donde se realiza una revisión de las distintas realizaciones hardware/software aparecidas en la literatura de este tipo de sistemas.

1.1. Extracción de bases de reglas a partir de datos numéricos

Las técnicas dedicadas al análisis y extracción de información a partir de datos numéricos son cada vez más necesarias porque son una forma de reducir y compactar información. Desde el punto de vista hardware, la realización de un sistema que respondiera con unas determinadas salidas frente a unas determinadas entradas siempre podría hacerse mediante una tabla de búsqueda (lookup table). Sin embargo, esta solución sólo tiene sentido para pocos datos (de otro modo, se dispara la complejidad de la tabla) y si no se requiere excesiva continuidad entre los valores que toma la variable de salida. En cualquier otro caso, siempre es mejor recurrir a interpoladores que reproduzcan los datos usando menos recursos de memoria. Dentro de los interpoladores, cobran gran interés los basados en lógica difusa, debido a su capacidad de extraer conocimiento lingüístico de fácil comprensión por personas no expertas.

Debido a la importancia de esta rama, se han desarrollado numerosas técnicas dedicadas a la extracción de bases de reglas difusas a partir de datos numéricos, dentro de las cuales se pueden distinguir las basadas en métodos de agrupamiento (o *clustering*) y las basadas en particiones de tipo rejilla (o *grid*) de los universos de discurso de las variables del problema.

1.1.1. Métodos de clustering

Las técnicas de *clustering* se basan en la idea de que, dada una serie de datos (o puntos, si se representan gráficamente), se puede realizar una división entre ellos, es decir, se puede realizar una serie de *"agrupamientos"* o colecciones de datos dentro de la nube de puntos que representan los datos. Aquellos datos que estén en el mismo grupo deben tener un cierto grado de cercanía o de características similares. A cada uno de esto grupos se les denomina *cluster*.

De este modo, cualquier dato puede pertenecer a un único cluster (teniendo un grado de pertenencia de 1 respecto a ese cluster) o a varios de ellos (teniendo un grado de pertenencia entre 0 y 1 respecto a cada uno de ellos). Cuando se impone la restricción de que cada dato pertenezca a un único cluster, se habla de clustering no-difuso o *hard clustering*, donde uno de los algoritmos más utilizados es el denominado Hard C-Means o K-Means. Por otro lado, si se permite que un mismo dato pueda pertenecer a varios clusters, se habla de clustering difuso (*fuzzy* o *soft clustering*). En este último caso, se puede cuantificar cuán difuso es cada cluster, es decir, mientras más difusos sean los clusters, cada dato podrá pertenecer a un número mayor de ellos al mismo tiempo.

La mayor ventaja que ofrecen las técnicas de clustering es que se pasa de manejar un gran número de datos a sólo unos pocos representativos (o prototipos) de ellos, que vienen a ser los centros de cada cluster.

A la hora de crear la base de reglas del sistema difuso, cada cluster obtenido por una técnica de clustering se transforma en una regla mediante su proyección en cada dimensión de las variables de entrada [Sugeno 93] [Klawonn 97]. 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, eliminando así el problema denominado *"la maldición de la dimensionalidad"* que aparece con las técnicas basadas en una partición de tipo grid, que consiste en que el número de reglas aumenta exponencialmente con el número de variables de entrada. Como contrapartida, las reglas obtenidas a partir de la proyección de los clusters no suelen tener 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) [Sugeno 93]. Otros autores se centran en simplificar las funciones de pertenencia obtenidas para conseguir un sistema difuso más transparente e interpretable [Setnes 98].

Las técnicas de clustering pueden obtener un número de clusters fijado de antemano o ir incrementando este número para que se ajuste mejor al modelo planteado por los datos. Así, las técnicas de clustering se pueden dividir en técnicas que se denominarán de *fixed clustering* (para el caso de un número de clusters prefijado) y técnicas que se denominarán de *incremental clustering* (para cuando el número de clusters varía para conseguir un determinado grado de adaptación a la información numérica).

1.1.1.1. Técnicas de *Fixed Clustering*

Considérese que se quiere obtener un sistema difuso con $p-1$ entradas y una salida. Supóngase, además, que el fichero de datos correspondiente para obtener dicho sistema tiene n patrones de datos, representados en forma matricial:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,p-1} & y_1 \\ x_{2,1} & x_{2,2} & \dots & x_{2,p-1} & y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,p-1} & y_n \end{bmatrix} \tag{1.1}$$

Estos n patrones se pueden representar, a su vez, de forma vectorial:

$$\begin{aligned} &\overrightarrow{x_1} = (x_{1,1}, x_{1,2}, \dots, x_{1,p-1}, y_1) \\ &\overrightarrow{x_2} = (x_{2,1}, x_{2,2}, \dots, x_{2,p-1}, y_2) \\ &\vdots \\ &\overrightarrow{x_n} = (x_{n,1}, x_{n,2}, \dots, x_{n,p-1}, y_n) \end{aligned} \tag{1.2}$$

Los algoritmos de clustering buscan minimizar una función objetivo. En este sentido, la función objetivo más utilizada para obtener un número prefijado de c clusters es la siguiente:

$$J[U, V; X] = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^m d_{ik}^2 \left(\overrightarrow{x_k}, \overrightarrow{v_i}\right) \tag{1.3}$$

Donde:

- $V = (\overrightarrow{v_1}, \overrightarrow{v_2}, \dots \overrightarrow{v_c})$, es la matriz formada por los centros de los c clusters. Cada uno de estos centros viene descrito por un vector de p elementos.
- X es la matriz formada por los datos numéricos de partida (1.1).
- U es la matriz de grados de pertenencia ($u_{ik} \in [0, 1]$), siendo u_{ik} el grado de pertenencia del dato $\overrightarrow{x_k}$ al cluster i.
- $d_{ik}\left(\overrightarrow{x_k}, \overrightarrow{v_i}\right)$ es la distancia entre el dato $\overrightarrow{x_k}$ y el prototipo $\overrightarrow{v_i}$.
- m es un exponente de ponderación que indica lo difuso que es el cluster. Cuando m tiende a 1, los clusters tienden a ser no-difusos (*hard clustering*), porque entonces los u_{ik} tienden a valer 1 ó 0. Por otro lado, si m tiende a infinito, los u_{ik} tienden a $\frac{1}{c}$, de forma que todos los datos pertenecen con el mismo grado a todos los clusters, es decir, que los clusters son totalmente difusos. La elección de m = 2 es ampliamente aceptada como buen valor para un clustering difuso [Hathaway 01].

Derivando la función objetivo J respecto a u_{ik} y $\overrightarrow{v_i}$ e imponiendo que tales derivadas se anulen para llegar a un mínimo local, se obtiene que:

$$\overrightarrow{v_i} = \frac{\sum_{k=1}^{n} (u_{ik})^m \cdot \overrightarrow{x_k}}{\sum_{k=1}^{n} (u_{ik})^m} \tag{1.4}$$

$$u_{ik} = \left[\sum_{j=1}^{c} \left(\frac{d_{ik}^2}{d_{jk}^2} \right)^{\frac{1}{m-1}} \right]^{-1} \tag{1.5}$$

Las técnicas de *fixed clustering* parten de unos prototipos $\overrightarrow{v_i}$ inicializados aleatoriamente o elegidos en base a unos determinados criterios. A partir de esos $\overrightarrow{v_i}$ se calculan los u_{ik} de acuerdo con la ecuación (1.5). Obtenidos los u_{ik} se vuelven a recalcular los $\overrightarrow{v_i}$ de acuerdo con la ecuación (1.4). Y así sucesivamente hasta que la diferencia entre los nuevos $\overrightarrow{v_i}$ y los antiguos sea inferior a un valor que fija la condición de parada del algoritmo.

Tres algoritmos muy conocidos que pertenecen a estas técnicas de *fixed clustering* son el algoritmo Fuzzy C-Means, el de Gustafson-Kessel y el de Gath-Geva.

El primero de ellos, propuesto por Bezdek [Bezdek 84], utiliza la distancia euclídea para $d_{ik}\left(\overrightarrow{x_k}, \overrightarrow{v_i}\right)$ por lo que genera clusters esféricos de aproximadamente el mismo tamaño. Este algoritmo es apropiado a la hora de extraer reglas de tipo Takagi-Sugeno de orden cero o con MaxLabel. Este último método de *defuzzification* (MaxLabel) aporta como salida el consecuente de la regla más activada y es utilizado en bases de reglas difusas que actúan como clasificadores. Otros métodos de *defuzzification*, como Fuzzy Mean (explicado posteriormente), se utilizan en bases de reglas que actúan como interpoladores. Los métodos de inferencia tipo Takagi-Sugeno [Takagi 85] también se explican más adelante.

El algoritmo de Gustafson-Kessel [Gustafson 79] emplea la distancia de Mahalanobis en vez de la euclídea:

$$d_{ik}^2\left(\overrightarrow{x_k}, \overrightarrow{v_i}\right) = (\rho_i \det C_i)^{\frac{1}{p}} \cdot \left(\overrightarrow{x_k} - \overrightarrow{v_i}\right)^T \cdot C_i^{-1} \cdot \left(\overrightarrow{x_k} - \overrightarrow{v_i}\right) \tag{1.6}$$

En donde se introducen los parámetros ρ_i y C_i, que representan el volumen del cluster y la matriz de covarianza del cluster i-ésimo, respectivamente. La matriz de covarianza viene descrita por la ecuación:

$$C_i = \frac{\sum_{k=1}^{n} (u_{ik})^m \cdot \left(\overrightarrow{x_k} - \overrightarrow{v_i}\right) \cdot \left(\overrightarrow{x_k} - \overrightarrow{v_i}\right)^T}{\sum_{k=1}^{n} (u_{ik})^m} \tag{1.7}$$

Al cambiar la distancia euclídea por la distanca de Mahalanobis, los cluster pasan de ser esféricos a ser hiperelipsoides. De este modo, el algoritmo de Gustafson-Kessel es apropiado a la hora de extraer reglas de tipo Takagi-Sugeno de orden uno.

En el caso del algoritmo de Gath-Geva [Gath 89] se utilizan los mismos parámetros que en el algoritmo de Gustafson-Kessel, salvo que el parámetro ρ_i pasa a ser la probabilidad a priori de que un dato pertenezca al cluster i:

$$\rho_i = \frac{1}{n} \sum_{k=1}^{n} u_{ik} \tag{1.8}$$

La otra diferencia entre ambos métodos es la sustitución de la distancia de Mahalanobis por una distancia basada en distribuciones normales:

$$d_{ik}^2 (\overrightarrow{x_k}, \overrightarrow{v_i}) = \frac{\sqrt{\det C_i}}{\rho_i} \cdot e^{\frac{(\overrightarrow{x_k} - \overrightarrow{v_i})^T \cdot C_i^{-1} \cdot (\overrightarrow{x_k} - \overrightarrow{v_i})}{2}} \tag{1.9}$$

El método de Gath-Geva, como el de Gustafson-Kessel, es preferible cuando los clusters tienen forma de hiperelipsoides. Mientras que el algoritmo de Gustafson-Kessel no funciona bien con clusters de distintos tamaños y orientaciones, el de Gath-Geva sí, porque puede variar el tamaño de los clusters, adaptándose a ellos. En cualquier caso, ambos algoritmos fallan cuando los datos están perfectamente alineados, ya que las matrices asociadas a los clusters tienen determinante cero, lo que impide que se puedan invertir.

Otra distancia utilizada para $d_{ik} (\overrightarrow{x_k}, \overrightarrow{v_i})$ en la expresión (1.3) es la distancia de Minkowski. Concretamente, en [Groenen 01], partiendo del trabajo de [Bobrowski 91], para luego presentar una forma más generalizada en [Groenen 07], sustituyen el término $d_{ik}^2 (\overrightarrow{x_k}, \overrightarrow{v_i})$ de la ecuación (1.3) por:

$$d_{ik}^{2\lambda} (\overrightarrow{x_k}, \overrightarrow{v_i}) = \left(\sum_{j=1}^{p} |x_{kj} - v_{ij}|^{\beta} \right)^{\frac{2\lambda}{\beta}} \tag{1.10}$$

Con $1 \leq \beta \leq \infty$, $0 \leq \lambda \leq 1$.

Estudiando los distintos valores de β y λ, analizan tanto las formas de los clusters obtenidos, como los valores que aportan las soluciones más robustas. De este modo, presentan un método que da buenos resultados para reconstruir clusters de datos perturbados por errores.

Un trabajo más actual relacionado con éstos es el de [Srivastava 11], donde la utilización de distancias de Minkowski se combina con algoritmos evolutivos para conseguir una mejor partición inicial del espacio.

Por otro lado, hay situaciones en las que los clusters no son fácilmente identificables (por ejemplo, un par de clusters que no sean linealmente separables en el espacio original, también llamado *espacio de características*). En este caso, algunos autores optan por transformar los datos del espacio de características a un espacio de mayor dimensionalidad (incluso infinita), llamado *espacio de kernel*, con la esperanza de encontrar

con mayor facilidad los clusters en este nuevo espacio [Schölkopf 98] [Müler 01]. Dentro de este ámbito, uno de los primeros trabajos propuestos para el clustering difuso es el de [Zhang 03], donde se presenta el algoritmo de Fuzzy C-Means basado en kernel, que utilizan para realizar clustering sobre datos incompletos.

A veces, un único kernel es insuficiente para representar los datos. Es por eso que algunos autores plantean la utilización de múltiples kernels a partir de un conjunto de *kernels base* para refinar resultados. Cada kernel se corresponderá con características distintas de los resultados, significando una transformación no lineal distinta cada uno de ellos. Dentro de este tipo de aproximaciones, aparece el trabajo de [Huang 12], donde se presenta una extensión de estos métodos para el clustering difuso.

En [Graves 10] se hace una comparación exhaustiva entre los algoritmos Fuzzy C-Means y Gustafson-Kessel, con distintos algoritmos de clustering basados en kernel. La conclusión a la que llegan es que no siempre un tipo de algoritmo es superior al otro. Esto es debido, en parte, a las desventajas que tienen los algoritmos basados en kernel, como es la dificultad en la selección de la función kernel, así como de los parámetros de optimización del kernel (por ejemplo, si se utiliza un kernel gaussiano, habría que especificar el valor de σ, la anchura de la gaussiana). Dependiendo de la selección de estos parámetros, el uso de kernels puede generar estructuras extrañas. Por tanto, el saber cuándo y cómo aplicar algoritmos basados en kernel de forma ventajosa, se convierte en un problema complejo [Pal 14].

Por último, la calidad de la solución propuesta por las técnicas de fixed clustering, como ocurre en la mayoría de los problemas de optimización no lineal, depende en gran medida de la elección de los valores iniciales del algoritmo, como, por ejemplo, del número de clusters prefijado y de los centros de los clusters iniciales. La elección del número de clusters influye mucho en el resultado, puesto que puede obligar a formar un número de grupos que no es el adecuado para la distribución de los datos. Determinar el número óptimo de clusters para un problema es muy importante, pero usualmente muy complicado, porque los datos pueden no responder a agrupaciones claras. Por esta razón, se han desarrollado las denominadas *técnicas de validación de clusters*, cuyo objetivo es determinar si el número de clusters encontrados puede ser considerado el mejor de entre los posibles. Han habido incontables estudios de medidas de validez directas e indirectas para métodos de clustering concretos (hard), probabilísticos y difusos. Los *índices de validación directos* suelen emplearse para evaluar clusters concretos, es decir, no difusos; mientras que, para evaluar clusters difusos y probabilísticos, se emplean *índices indirectos* [Ruspini 98]. Sin embargo, estas funciones de validación típicamente se utilizan para eliminar soluciones muy malas y, normalmente, como parte del proceso previo para una validación final realizada por un humano.

1.1.1.2. Técnica de *Incremental Clustering*

Para solucionar este tipo de problemas, Chiu [Chiu 94] propuso la técnica *Subtractive Clustering* (que ha sido referida en este capítulo como *Incremental Clustering*, para distinguirla de las técnicas que han sido denominadas como *Fixed Clustering*). Esta técnica es una modificación del método de estimación de clusters de Yager y Filev [Yager 94], en el cual se introducía por primera vez la idea de centro de cluster potencial o, simplemente, cluster potencial.

La técnica de Yager y Filev está basada en una partición de tipo rejilla (o grid) del espacio de datos y en la computación de un valor potencial para cada punto del grid basado en su distancia con el resto de datos. De este modo, un punto del grid con muchos puntos en su cercanía tendrá un alto valor de potencial. El punto del grid que tenga el mayor valor de potencial será elegido como el primer centro de cluster. La idea clave de este método es que, una vez elegido el primer centro de cluster, el potencial de todos los puntos del grid se reduce acorde a su distancia al centro del cluster. Por tanto, aquellos puntos que estén cerca del primer centro de cluster verán su potencial altamente reducido. El siguiente centro de cluster se eligirá como el punto del grid que tenga el siguiente potencial mayor. Estos pasos se repetirán de forma iterativa hasta que el potencial de todos los puntos del grid sean inferiores al de un valor umbral.

Aunque el procedimiento es sencillo y efectivo, tiene el problema de que la computación crece exponencialmente según la dimensión del problema (*maldición de la dimensionalidad*) como ocurre en todos los problemas que emplean una partición tipo grid.

La técnica *Incremental Clustering* evita este problema considerando cada dato del sistema, y no cada punto del grid, como un centro de cluster potencial. Usando esta idea, el número de *"puntos del campo"* efectivos a evaluar se simplifica al número de datos del sistema planteado, independientemente de la dimensión del problema. De esta forma se estudia la distancia de cada dato a los demás. El dato que tenga más datos cercanos, será el que presentará un valor potencial más alto, convirtiéndose en el primer centro de cluster potencial. Una vez seleccionado el primer centro de cluster, todos los datos cercanos a éste disminuirían su valor potencial de ser centros. De esta forma, el siguiente centro de cluster será aquel que presente el siguiente valor potencial mayor. Este proceso se realizará de forma iterativa hasta que el número de clusters sea el impuesto como límite en el algoritmo.

Este método, por lo tanto, puede proporcionar cuántos y cuáles son los centros de clusters con mayor potencialidad de serlos, para una serie de datos dados, lo cual se podría también aprovechar para realizar una inicialización más correcta de los algoritmos de *Fixed Clustering*.

1.1.2. Métodos de tipo grid

Los métodos de tipo rejilla o *grid* generan una partición de los espacios de las entradas y las salidas antes de crear la base de reglas. Las reglas se obtienen seleccionando las etiquetas más adecuadas para las entradas y las salidas respecto a la base de datos numéricos. Estos métodos consiguen reglas con mucho más significado lingüístico, pero con el coste de obtener un número de reglas mucho mayor. Concretamente, suponiendo un sistema de n variables de entrada y m etiquetas lingüísticas para cada una de ellas, el número de reglas del sistema difuso es de m^n. Este fenómeno, en el que la complejidad de un problema crece exponencialmente con el número de las variables de entrada, no es exclusivo de los sistemas difusos, y fue identificado por Bellman en [Bellman 61] como *"la maldición de la dimensionalidad"*.

Algunas soluciones reportadas en la literatura para evitar este problema han sido seleccionar las reglas más significativas (como proponen Nauck en [Nauck 00] y Senhadji en [Senhadji 02]), o mediante la descomposición jerárquica del sistema, pero estos temas serán abordados en mayor detalle en los capítulos siguientes.

El paso de los datos de entrada-salida a un sistema difuso, se puede hacer de una manera directa mediante el algoritmo de Wang y Mendel (al que se referirá como *Fixed Grid Partitioning*), o de manera iterativa con el algoritmo de Higgins-Goodman (al que se referirá como *Incremental Grid Partitioning*), que se describirán a continuación.

1.1.2.1. Algoritmo *Fixed Grid Partitioning* (algoritmo de Wang y Mendel)

El algoritmo de Wang y Mendel [Wang 92b] es una de las técnicas más conocidas para construir un sistema difuso mediante el estudio de un conjunto de datos y, de manera opcional, de la experiencia aportada por un experto humano.

Considerando un fichero de datos de la misma forma al utilizado en los apartados anteriores (ecuación (1.1)), el primer paso que realiza este algoritmo es el de definir una estructura grid para las n variables de entrada. Al mismo tiempo, se seleccionan el tipo de funciones de pertenencia para dichas variables, así como el número de etiquetas en el que se dividirá cada universo de discurso.

Una vez se ha generado la partición grid se analiza, para cada uno de los n patrones $\overrightarrow{x_i}$, cuáles son las etiquetas de cada universo de discurso a las que el patrón tiene un mayor grado de pertenencia. Y, a partir de esta información, se van generando las reglas difusas.

Teniendo en cuenta que cada patrón generará una regla y que el número de patrones es elevado, no es de extrañar que aparezcan reglas contradictorias, es decir, reglas que tienen el mismo antecedente, pero distinto consecuente. En pos de solucionar dicho problema, a cada una de las reglas generadas se le asigna un grado, el cual se determina como el producto de grados de pertenencia de las variables de entrada y salida, de la

forma:

$$D = \prod_{i=1}^{n} \mu_{IF}(x_i) \cdot \mu_{THEN}(y_i) \tag{1.11}$$

Calculado el grado de todas las reglas, si en el conjunto de reglas hay más de una regla con los mismos antecedentes, se eliminan las que tengan menor grado, resolviendo así los conflictos que se puedan dar.

1.1.2.2. Algoritmo *Incremental Grid Partitioning*

La partición del sistema difuso generada por el algoritmo de Wang y Mendel, tal y como acaba de verse, es fija. El algoritmo *Incremental Grid Partitioning* [Higgins 94] va un paso más allá, realizando una partición de los universos de discurso de manera gradual, comprobando en cada nueva partición el grado de acercamiento a los datos del modelo que se está estudiando. De este modo, se consigue un particionado gradual o incremental.

El sistema difuso generado por el algoritmo *Incremental Grid Partitioning* es de tipo singleton, es decir, utiliza como método de *defuzzification* el Fuzzy Mean (o método de la altura), obteniendo la salida del sistema de la forma:

$$y_o = \frac{\sum_r \alpha^r \cdot c^r}{\sum_r \alpha^r} \tag{1.12}$$

Donde α^r es el grado de activación y c^r la conclusión tipo singleton de la regla r-ésima. Cabe destacar que, mientras que el algoritmo de Wang y Mendel puede ser utilizado tanto en sistemas difusos para problemas de interpolación, como de clasificación (puede utilizar como métodos de *defuzzification* tanto Takagi-Sugeno, como Fuzzy Mean, como MaxLabel), el algoritmo de *Incremental Grid* sólo puede ser utilizado en sistemas difusos que actúen como interpoladores (ya que utiliza únicamente como método de *defuzzification* el Fuzzy Mean).

Dentro de este algoritmo, pueden encontrarse dos variantes: Incremental Grid Partitioning con y sin aprendizaje de los singletons.

La variante que no considera aprendizaje de los singletons es la comentada inicialmente. La que sí lo considera realiza un ajuste, cada vez que se crea una nueva partición, de los valores de los consecuentes mediante aprendizaje supervisado. Aunque pueda parecer a primera vista que esto sería siempre lo idóneo, no es así de forma práctica, ya que, a veces, el introducir aprendizaje en los pasos intermedios de generación de determinados sistemas lleva a caminos sin salida y/o a soluciones con mayor grado de cómputo y resultados incluso peores. Ejecutar el aprendizaje sólo al final sí que es siempre ventajoso, como se comentará a continuación.

1.1.3. Ajuste fino a los datos numéricos

En los sistemas difusos que son de tipo singleton (Takagi-Sugeno de orden cero) o Takagi-Sugeno de orden uno, la dependencia de la salida frente a los valores asociados con los consecuentes es lineal, con lo que el error cuadrático medio tiene una forma de hiperparaboloide respecto a ellos. No hay, por tanto, mínimos locales sino un solo conjunto de esos valores que aseguran el valor mínimo global del error cuadrático medio. Con lo cual, es aconsejable como paso final del proceso de extracción de reglas aplicar cualquier método de aprendizaje supervisado sobre los consecuentes.

Tras la experiencia obtenida con el desarrollo de los distintos sistemas relacionados con esta Tesis, se ha concluido que lo mejor es emplear un método de optimización numérica de tipo *Quasi-Newton*. Concretamente, el algoritmo más utilizado ha sido el de Levenberg-Marquardt [Battiti 92], que encuentra ese mínimo global en muy pocas iteraciones, porque calcula un valor muy eficiente para la velocidad de aprendizaje, sin tener que calcular la matriz Hessiana.

En los sistemas difusos que emplean como método de *defuzzification* el MaxLabel (se aplican a problemas de clasificación), el error cuadrático medio no se puede emplear muchas veces porque los consecuentes de las reglas son símbolos que, aún representándose por números, no tienen mucho sentido empleados en una medida de distancia numérica. En estos casos, se suele emplear como medida de error el *número de errores de clasificación* (CE), definido como:

$$CE = \frac{1}{n} \cdot \frac{1}{s} \cdot \sum_{i=1}^{n} \sum_{j=1}^{s} \delta_{ij} \tag{1.13}$$

$$\delta_{ij} = \begin{cases} 0 & \text{si} \quad y_{ij} = \widetilde{y}_{ij} \\ 1 & \text{si} \quad y_{ij} \neq \widetilde{y}_{ij} \end{cases} \tag{1.14}$$

Donde n es el número de patrones de entrenamiento; s es el número de variables de salida; y_{ij} es el valor de la salida j-ésima del sistema para las entradas del patrón i-ésimo; $\widetilde{y}_{ij}$ es el valor deseado de la salida j-ésima según el patrón i-ésimo; y δ_{ij} es 0 si el patrón ha sido clasificado correctamente ó 1 si ha sido clasificado de forma incorrecta.

Como esta medida de error no es derivable ni cuadrática, para aplicar el algoritmo de Levenberg-Marquardt, es mejor emplear un *error de clasificación cuadrático* (CSE) como el siguiente:

$$CSE = \frac{1}{n} \cdot \frac{1}{s} \cdot \sum_{i=1}^{n} \sum_{j=1}^{s} (\alpha_{ij} - \delta_{ij})^2 \tag{1.15}$$

Donde α_{ij} es el grado de activación de la etiqueta de salida.

1.2. Sistemas neuro-difusos como sistemas polinómicos a tramos

Los sistemas neuro-difusos proporcionan, en general, superficies de salida no lineales frente a los valores de las entradas. Dado un valor para las entradas, una parte del conjunto de reglas difusas tendrá un grado de activación superior a cero, que será la responsable del valor de salida del sistema. Como se verá a continuación, dependiendo de las funciones de pertenencia para los antecedentes y consecuentes de dichas reglas, así como de los operadores difusos empleados, un sistema neuro-difuso puede comportarse como un sistema polinómico a tramos.

Supóngase un sistema neuro-difuso de tipo Takagi-Sugeno con partición *grid*, n entradas $(x_1, \ldots, x_n)$, con M_i etiquetas para cada una de las x_i entradas, con un grado de solapamiento s. Dado un valor para las entradas $\overrightarrow{x_o}$ $(x_{1o}, \ldots, x_{no})$, el sistema neuro-difuso identifica la celda particular del grid (CG_o) a la que las entradas pertenecen. Localizada la celda, el sistema proporciona la salida correspondiente, $y(\overrightarrow{x_o})$, evaluando las s^n reglas activas de la forma:

$$y(\overrightarrow{x_o}) = \frac{\sum_{j=1}^{s^n} \alpha_{oj}(\overrightarrow{x_o}) \cdot c_{oj}(\overrightarrow{x_o})}{\sum_{j=1}^{s^n} \alpha_{oj}(\overrightarrow{x_o})} = y_o \tag{1.16}$$

Donde α_{oj} es el grado de activación de una de las s^n reglas activas y c_{oj} es la función del consecuente correspondiente a la misma.

Los parámetros que definen y_o son una parte del conjunto global de parámetros que definen el sistema neuro-difuso $y(\overrightarrow{x})$. En este sentido, un sistema neuro-difuso se puede ver como un interpolador local a tramos, que proporciona el valor y_o para cada celda CG_o que forma la estructura grid del sistema. La interpolación proporcionada puede ser constante a tramos, lineal a tramos, cuadrática a tramos y, en general, no lineal a tramos, dependiendo de que y_o sea constante, lineal, cuadrática o no lineal respecto a $\overrightarrow{x_o}$.

Supóngase que las funciones de pertenencia de las entradas son familias de B-splines de orden cero, uno o dos (la forma de estas funciones se puede apreciar en la Figura 1.1). Estas familias están normalizadas, es decir, la suma de los grados de pertenencia a las distintas etiquetas para cualquier entrada siempre es la unidad. Si se considera el caso particular en el que el sistema neuro-difuso es de tipo Takagi-Sugeno de orden cero y se elige como operador conectivo entre antecedentes el producto, se puede comprobar que el sistema neuro-difuso se comporta como un sistema constante a tramos, multilineal a tramos y multicuadrático a tramos si las funciones de pertenencia para los antecedentes son B-splines de orden cero, uno y dos respectivamente [Baturone 01].

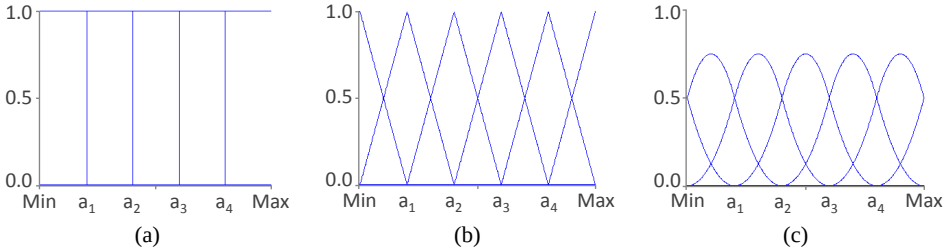

Figura 1.1: Familia de B-splines de orden: (a) cero; (b) uno; (c) dos

1.2.1. Sistemas neuro-difusos como sistemas constantes a tramos

Para el caso particular de un sistema de tipo Takagi-Sugeno de orden cero, cuyos antecedentes son familias de B-splines de orden cero (Figura 1.1a), es inmediato comprobar que el sistema se comporta como un sistema constante a tramos.

Cuando se utilizan B-splines de orden cero no hay solapamiento entre las funciones de pertenencia ($s = 1$), es decir, es un caso *"no difuso"*. Al no haber solapamiento entre las funciones de pertenencia de los antecedentes, sólo habrá una regla activa para un valor concreto de las entradas, cuyo grado de activación siempre será uno. Luego, una vez determinada la regla a la que pertenece el valor de las entradas, el sistema devolverá el consecuente c_{oj} correspondiente a dicha regla que, al ser un sistema de tipo Takagi-Sugeno de orden cero, es un valor constante.

1.2.2. Sistemas neuro-difusos como sistemas multilineales a tramos

Supóngase ahora el caso de un sistema tipo Takagi-Sugeno de orden cero en el que las funciones de pertenencia para los antecedentes son familias de B-splines de orden uno (Figura 1.1b). Se trata de funciones triangulares normalizadas con grado de solapamiento s igual a dos. Para un valor de una entrada dada, x_i, perteneciente al intervalo $[a_o^{(x_i)}, a_{o+1}^{(x_i)}]$, al ser s igual a dos, se activarán hasta un máximo de dos etiquetas. El valor de los grados de activación de estas etiquetas vendrá dado por:

$$\mu_{o1}^{(x_i)} = \frac{x_i - a_{o+1}}{a_o - a_{o+1}} \tag{1.17}$$

$$\mu_{o2}^{(x_i)} = 1 - \mu_{o1}^{(x_i)} \tag{1.18}$$

Donde la expresión (1.18) se consigue a partir de la normalización existente entre las funciones de pertenencia de los antecedentes.

Para simplificar el estudio, se puede suponer el caso particular de dos entradas,

(x_1, x_2). Llevando todo esto a la expresión (1.16), y sabiendo que su denominador es uno debido a la normalización de las funciones de pertenencia de los antecedentes, se tiene que:

$$y_o = \mu_{o1}^{(x_1)}\mu_{o1}^{(x_2)}c_{o11} + \mu_{o1}^{(x_1)}\mu_{o2}^{(x_2)}c_{o12} + \mu_{o2}^{(x_1)}\mu_{o1}^{(x_2)}c_{o21} + \mu_{o2}^{(x_1)}\mu_{o2}^{(x_2)}c_{o22} =$$
$$= \mu_{o1}^{(x_1)}\mu_{o1}^{(x_2)}c_{o11} + \mu_{o1}^{(x_1)}(1-\mu_{o1}^{(x_2)})c_{o12} + (1-\mu_{o1}^{(x_1)})\mu_{o1}^{(x_2)}c_{o21} + (1-\mu_{o1}^{(x_1)})(1-\mu_{o1}^{(x_2)})c_{o22} =$$
$$= c_{o22} + \mu_{o1}^{(x_1)}(c_{o12} - c_{o22}) + \mu_{o1}^{(x_2)}(c_{o21} - c_{o22}) + \mu_{o1}^{(x_1)}\mu_{o1}^{(x_2)}(c_{o11} - c_{o12} - c_{o21} + c_{o22})$$

Teniendo en cuenta que $\mu_{o1}^{(x_i)}$, a partir de la expresión (1.17), es lineal respecto a x_i, se tiene que la forma de y_o es del tipo:

$$y_o(x_1, x_2) = ax_1x_2 + bx_1 + cx_2 + d \tag{1.19}$$

Donde a, b, c y d son constantes relacionadas con los parámetros que definen las funciones de pertenencia de los antecedentes y de los valores singleton de los consecuentes. Se concluye, por tanto, que el sistema neuro-difuso se comporta como un sistema multilineal a tramos con respecto a las entradas.

Si, en vez de usar el producto como conectivo entre antecedentes, se utiliza el operador *meet* [Sasaki 94] [Rovatti 98]:

$$meet(\mu_1, \mu_2) = \frac{\text{mín}(\mu_1, \mu_2) + \text{máx}(\mu_1 + \mu_2 - 1, 0)}{2} \tag{1.20}$$

El sistema resultante no es multilineal a tramos, sino lineal a tramos, de forma que, para cada tramo, la salida es:

$$y_o(x_1, x_2) = ax_1 + bx_2 + c \tag{1.21}$$

Los sistemas lineales a tramos se conocen como sistemas PWA, por las siglas en inglés de *PieceWise Affine*.

1.2.3. Sistemas neuro-difusos como sistemas multicuadráticos a tramos

Un sistema neuro-difuso que se comporte como un sistema multicuadrático a tramos se puede obtener si los consecuentes son lineales respecto a las entradas del sistema (sistema Takagi-Sugeno de orden uno) y las funciones de pertenencia de los antecedentes son también lineales respecto a las entradas; o bien, que las funciones de pertenencia de los consecuentes sean constantes (Takagi-Sugeno de orden cero) y las de los antecedentes sean cuadráticas respecto a las entradas. Este último caso sucede cuando se usan familias de B-splines de orden dos (Figura 1.1c). El grado de solapamiento entre las funciones s es igual a tres, es decir, se pueden activar a la vez hasta tres funciones de pertenencia por cada entrada. El valor de estos grados de pertenencia, para la entrada x_i, viene dado

por:

$$\mu_{o1}^{(x_i)} = \frac{(x_i - a_o)^2}{(a_{o+1} - a_o) \cdot (a_{o+2} - a_o)} \tag{1.22}$$

$$\mu_{o2}^{(x_i)} = \frac{(x_i - a_{o+1})^2}{(a_{o+1} - a_o) \cdot (a_{o+1} - a_{o-1})} \tag{1.23}$$

$$\mu_{o3}^{(x_i)} = 1 - \mu_{o1}^{(x_i)} - \mu_{o2}^{(x_i)} \tag{1.24}$$

Donde la expresión (1.24) se consigue a partir de la normalización existente entre las funciones de pertenencia de los antecedentes.

Una vez más, con el objetivo de simplificar, se tiene en cuenta el caso particular de dos entradas (x_1, x_2). Analizando nuevamente la ecuación (1.16), en la que su denominador vuelve a valer la unidad por la normalización de los B-splines, se tiene que:
$y_o = \mu_{o1}^{(x_1)}\mu_{o1}^{(x_2)}c_{o11} + \mu_{o1}^{(x_1)}\mu_{o2}^{(x_2)}c_{o12} + \mu_{o1}^{(x_1)}\mu_{o3}^{(x_2)}c_{o13} + \mu_{o2}^{(x_1)}\mu_{o1}^{(x_2)}c_{o21} + \mu_{o2}^{(x_1)}\mu_{o2}^{(x_2)}c_{o22} + \mu_{o2}^{(x_1)}\mu_{o3}^{(x_2)}c_{o23} + \mu_{o3}^{(x_1)}\mu_{o1}^{(x_2)}c_{o31} + \mu_{o3}^{(x_1)}\mu_{o2}^{(x_2)}c_{o32} + \mu_{o3}^{(x_1)}\mu_{o3}^{(x_2)}c_{o33} = c_{o33} + \mu_{o1}^{(x_1)}(c_{o13} - c_{o33}) + \mu_{o2}^{(x_1)}(c_{o23} - c_{o33}) + \mu_{o1}^{(x_2)}(c_{o31} - c_{o33}) + \mu_{o2}^{(x_2)}(c_{o32} - c_{o33}) + \mu_{o1}^{(x_1)}\mu_{o1}^{(x_2)}(c_{o11} - c_{o13} - c_{o31} + c_{o33}) + \mu_{o1}^{(x_1)}\mu_{o2}^{(x_2)}(c_{o12} - c_{o13} - c_{o32} + c_{o33}) + \mu_{o2}^{(x_1)}\mu_{o1}^{(x_2)}(c_{o21} - c_{o23} - c_{o31} + c_{o33}) + \mu_{o2}^{(x_1)}\mu_{o2}^{(x_2)}(c_{o22} - c_{o23} - c_{o32} + c_{o33})$

Teniendo en cuenta las expresiones (1.22) y (1.23), y llevadas a la expresión anterior, se tiene que la forma de y_o es del tipo:

$$y_o(x_1, x_2) = ax_1^2x_2^2 + bx_1^2x_2 + cx_1x_2^2 + dx_1^2 + ex_2^2 + fx_1x_2 + gx_1 + hx_2 + i \tag{1.25}$$

Donde $a, \ldots, i$ son constantes relacionadas con los parámetros que definen las funciones de pertenencia de los antecedentes y de los valores singleton de los consecuentes. Por tanto, vista la expresión (1.25), se deduce que el sistema neuro-difuso estudiado se comporta como un sistema multicuadrático a tramos con respecto a las entradas del sistema.

Como ya se ha comentado, estos resultados se obtienen para sistemas de tipo Takagi-Sugeno de orden cero, junto con el operador producto como conectivo de antecedentes.

En el caso de usar el operador *meet*, el sistema es cuadrático a tramos, de forma que, para cada tramo, la salida es:

$$y_o(x_1, x_2) = ax_1^2 + bx_2^2 + cx_1x_2 + dx_1 + ex_2 + f \tag{1.26}$$

Donde $a, \ldots, f$ son constantes relacionadas con los parámetros que definen las funciones de pertenencia de los antecedentes y de los valores singleton de los consecuentes.

En las tablas 1.1 y 1.2 se resumen las equivalencias de los distintos sistemas neuro-

Tabla 1.1: Equivalencia entre sistemas neuro-difusos de tipo Takagi-Sugeno de orden cero con sistemas polinómicos a tramos, según el conectivo y función de pertenencia elegidos para los antecedentes

	Orden de la familia de B-splines para los antecedentes		
Conectivo	0	1	2
Meet	Const. a tramos	Lineal a tramos	Cuad. a tramos
Producto	Const. a tramos	Multilin. a tramos	Multicuad. a tramos

Tabla 1.2: Equivalencia entre sistemas neuro-difusos de tipo Takagi-Sugeno de orden uno con sistemas polinómicos a tramos, según el conectivo y función de pertenencia elegidos para los antecedentes

	Orden de la familia de B-splines para los antecedentes	
Conectivo	0	1
Meet	Lineal a tramos	Cuadrática a tramos
Producto	Lineal a tramos	Multicuadrática a tramos

difusos con el tipo de sistema polinómico a tramos correspondiente, según si el sistema es de tipo Takagi-Sugeno de orden cero o uno, el grado de los B-splines usados como funciones de pertenencia en los antecedentes, o el operador utilizado para conectar los antecedentes.

Analizando los distintos casos, el utilizar familias de B-splines de orden cero da como resultado una solución *"no difusa"* al no existir solapamiento; por otro lado, los B-splines de orden dos tienen baja interpretabilidad lingüística, debido a que la pertenencia total a un conjunto no es posible. No sólo eso, sino que, pensando en la progresión hacia una implementación microelectrónica, estos B-splines son más complejos que los de orden uno. Los B-splines de orden uno son los más adecuados para modelos lingüísticos y para ser integrados de forma eficiente en sistemas con hardware específico. Usando este tipo de funciones pueden conseguirse sistemas neuro-difusos que se comporten como sistemas lineales a tramos, multilineales a tramos, cuadráticos a tramos y multicuadráticos a tramos [Rovatti 98] [Rovatti 99].

1.3. Realizaciones Hardware/Software de sistemas difusos y neuro-difusos

Los primeros sistemas difusos que se desarrollaron (sobre todo en aplicaciones de control) fueron implementaciones software en las que todo el proceso de *fuzzification*, inferencia y *defuzzification* se realizaba mediante un programa que se ejecutaba sobre

una determinada plataforma. Muchos de estos desarrollos se basaban en la utilización de ordenadores personales (PC) o controladores lógicos programables (PLC) en los que el programa a ejecutar se codificaba usualmente en lenguaje de alto nivel [Cabrera 02] [Chang 02]. Las ventajas más significativas de este tipo de implementación radican, por una parte, en la gran flexibilidad de la que se dispone, puesto que permiten definir sistemas con un número arbitrario de reglas, sin limitación en el número y tipo de las funciones de pertenencia y con una amplia gama de operadores difusos a seleccionar; y, por otra, en su facilidad de desarrollo.

Muchas de las herramientas de diseño asistido por ordenador (CAD) existentes para el desarrollo de sistemas difusos ofrecen como opción de implementación la generación de un programa que realiza las tareas del módulo de inferencia difuso, simplificando así el proceso de desarrollo de este tipo de sistemas. Este tipo de solución es apropiado para aquellas aplicaciones en donde, sobre todo, los aspectos de peso, volumen, consumo de potencia, y coste no sean restrictivos, como es el caso de muchos controladores de procesos industriales.

Pese a la gran potencialidad de los PCs actuales, las alternativas de implementación software pueden verse limitadas en velocidad de respuesta. Esto puede deberse al inherente carácter secuencial de la ejecución de los programas (con excepción de las costosas implementaciones basadas en computadoras de arquitectura de ejecución paralela [Lee 01]), pero, fundamentalmente, se debe a la forma en que se programan los diferentes algoritmos, la eficiencia del compilador empleado y la complejidad de los operadores difusos utilizados. Es común encontrar desarrollos software de sistemas difusos que procesan toda la base de reglas y no únicamente aquellas que pueden estar activas, o que utilizan métodos de *defuzzification* tradicionales que requieren recorrer todo el universo de discurso de las salidas para obtener el resultado, con lo cual se enlentece considerablemente la operación del sistema [Baturone 00].

Las características de las operaciones difusas que motivaron en los años 90 el desarrollo de otras estrategias de realización, para superar las limitaciones impuestas por los procesadores de propósito general de la época, fueron el paralelismo de los algoritmos difusos y el uso de operaciones básicas como máximo y mínimo, que no podían ser ejecutadas eficientemente por estos procesadores. En aplicaciones masivas de control, donde el controlador difuso estaba integrado con el resto del sistema, una alternativa de implementación software que eliminaba muchas de las limitaciones anteriores se basó en la utilización de microcontroladores o procesadores digitales de señal, dispositivos de bajo coste y consumo de potencia, así como muy reducido peso y tamaño. Estos dispositivos poseen entornos de desarrollo que facilitan todas las fases de simulación, depuración y programación y, normalmente, pueden programarse desde lenguajes de alto nivel como C o BASIC gracias al desarrollo de compiladores específicos. Esta alternativa seguía presentando la ventaja de la flexibilidad de la implementación, aunque se veía limitada

por la disponibilidad de recursos de los procesadores utilizados, fundamentalmente los relativos a la unidad aritmética y lógica (ALU) y al espacio de memoria disponible. Además, teniendo en cuenta las limitaciones en cuanto a frecuencias de operación y arquitectura interna, la velocidad de inferencia podía ser considerablemente inferior a aquellas obtenidas en las soluciones basadas en PCs, por lo que era usual recurrir al uso de algunas restricciones similares a las que se utilizan en las implementaciones hardware con el objetivo de no degradar ostensiblemente la velocidad del módulo de inferencia [von Altrock 98].

Con este objetivo, fueron varias las empresas que desarrollaron herramientas para proporcionar una descripción del sistema difuso en código específico que optimizaba la implementación en sus microcontroladores. Optimizando el código generado se pueden conseguir tiempos de respuesta por debajo del milisegundo para sistemas difusos de complejidad media. Entre las compañías y productos más significativos cabe citar a Togai InfraLogic, que desarrolló y comercializó a principios de los 90 productos como *MicroFPL*, *TIL Shell*, *TILGen* y *FuzzyCLIPS*. Aptronix Inc. introdujo en 1992 el sistema *FIDE* (Fuzzy Inference Development Language) para generar código ensamblador para distintos microcontroladores de Motorola, Intel, Siemens y Omron. La empresa alemana Inform desarrolló una extensa gama de productos basados en el sistema *FuzzyTECH* para proporcionar tanto soluciones de propósito general, que generaban código C estándar, como soluciones específicas, que generaban código ensamblador para diversos microcontroladores, DSPs, coprocesadores difusos, procesadores de propósito general con soporte difuso y PLCs [Nijhuis 94]. En esta misma línea, la compañía Togai Infralogic comercializó a principios de los 90 el sistema *MicroFPL* para microcontroladores de Motorola, Oki, Intel, etc. [Yen 95]. Finalmente, la compañía Rigel Corporation, dedicada a la fabricación de sistemas de control empotrados, distribuyó el software *rFLASH*.

Para aumentar más la velocidad de respuesta se modificaron algunas arquitecturas de procesadores y se desarrollaron otras específicas a las cuales se les incorporó algún soporte hardware para facilitar la ejecución de algoritmos difusos. Esta última aproximación para aumentar la velocidad de procesado difuso sobre plataformas de propósito general tenía como objetivo incrementar la funcionalidad de un procesador estándar incluyendo instrucciones que aceleraban los mecanismos de inferencia. Las primeras propuestas para añadir soporte difuso en arquitecturas CISC y RISC se realizaron a principios de los 90 [Ungering 94] [Salapura 00], y se materializaron a nivel industrial con dos claros exponentes. En 1996 Motorola introdujo la familia de microcontroladores de 16 bits *68HC12*, que incluía una serie de instrucciones específicas para el procesado de algoritmos difusos que proporcionaban un incremento de velocidad de un orden de magnitud en relación a su predecesor, el *68HC11* [Bannatyne 96]. Uno de los desarrollos más interesantes lo constituyó la familia *ST FIVE*, de ST Microelectronics, que ofrecía una arquitectura de microcontrolador tradicional combinada con una arquitectura dedicada para algo-

ritmos difusos [Fortuna 03]. Salvo estas soluciones, las grandes compañías fabricantes de microprocesadores fueron abandonado el interés en este tipo de realizaciones. En la actualidad, el potente desarrollo alcanzado por los procesadores ha provocado que no tenga sentido ninguna de las soluciones que fueron adoptadas en esa época.

Otra forma de ampliar la funcionalidad de un procesador de propósito general consistió en el desarrollo de circuitos integrados capaces de acelerar las operaciones de un módulo de inferencia difuso (coprocesadores difusos), realizando total o parcialmente mediante hardware dichas operaciones. Estos coprocesadores, interconectados con un procesador convencional permitían obtener realizaciones híbridas de controladores difusos. La mayor parte de los *"chips difusos"* comercializados en la década de los 90 se encuadran en esta categoría. Entre aquellos que alcanzaron mayor popularidad se pueden mencionar los chips *FC110* y *VY86C570* de Togai InfraLogic, el *T/FC150* de Toshiba, las familias *SAE 81C99* y *81C991* [Eichfeld 98] de Siemens, los coprocesadores *FP1000*, *FP3000*, *FP5000* [Eichfeld 96] y el módulo de procesador difuso para PLCs *FZ001* de Omron, y la arquitectura *W.A.R.P.* de ST Microelectronics [Eichfeld 98]. Este tipo de soluciones eran adecuadas para aplicaciones que requerían tiempos de respuesta de cientos a decenas de microsegundo. Sin embargo, aunque la mayoría de los coprocesadores comerciales poseían una arquitectura orientada a la ejecución de algoritmos difusos, éstas no eran totalmente eficientes o carecían de las opciones de configuración necesarias para adaptarlos a diversas aplicaciones [Eichfeld 96]. Adicionalmente, la interconexión entre los circuitos del procesador y el coprocesador podía resultar un inconveniente.

El avance en velocidad de los procesadores de propósito general provocó que las principales compañías fabricantes de microprocesadores y microcontroladores abandonaran a finales de los 90 el desarrollo comercial de los coprocesadores difusos. Sin embargo, en la última década y dentro del ámbito académico, se han desarrollado algunos trabajos sobre coprocesadores difusos. Un ejemplo de esto lo constituye el trabajo presentado en [García 98] por García y De Pedro donde proponen una nueva versión del coprocesador *ORBEX*. Dentro de esta misma línea otro trabajo es el propuesto por Thareja y coautores en [Thareja 07] donde presentan el desarrollo de un coprocesador en Handel-C para su conexión con un procesador de arquitectura RISC. Por otra parte, Raychev y coautores proponen en [Raychev 05] el desarrollo de un coprocesador modelado en lenguaje VHDL. Finalmente, Di Stefano y Giaconia presentan en [Di Stefano 05] el diseño de un coprocesador que puede ser fácilmente conectado a un procesador de 32 bits. Los resultados experimentales recogidos en este trabajo demuestran un aumento de hasta 1.000 veces en la velocidad de ejecución de un algoritmo difuso adaptativo usando el coprocesador propuesto frente a una implementación software.

Finalmente, en aplicaciones de robótica, aeronáutica, procesado de voz e imagen, electrónica de automoción, etc., pueden ser necesarios tiempos de respuesta de unos cuantos microsegundos o por debajo del microsegundo, con consumos de área y potencia

restrictivos. En estas situaciones se requiere una implementación totalmente hardware del sistema; por este motivo en la literatura han sido propuestas un gran número de implementaciones hardware de sistemas difusos [Basterretxea 09] [Hernández 12] [Bosque 14]. Como queda reflejado en los anteriores trabajos, las mejores prestaciones en cuanto a tiempo de respuesta y consumo de área y potencia se consiguen mediante un circuito integrado de aplicación específica (ASIC). Dentro de las realizaciones de un ASIC difuso se pueden distinguir las basadas en técnicas analógicas y digitales. Las técnicas analógicas, aunque permiten realizaciones con un menor consumo de recursos y de potencia, tienen limitaciones en aspectos como la complejidad o la precisión que se puede conseguir. Otra importante limitación de las técnicas analógicas frente a las técnicas digitales se encuentra en la automatización del proceso de diseño. Mientras que para las técnicas digitales existen herramientas que permiten automatizar muchos pasos del proceso de diseño, reduciendo el tiempo de desarrollo y aumentando la fiabilidad, para las técnicas analógicas el desarrollo de estas herramientas es mucho menor. En cualquier caso, los ASICs difusos son las implementaciones menos flexibles (por ejemplo, son las que menos complejidad soportan en cuanto al número de reglas o interfaz de comunicaciones). Además, su coste es elevado en tiempos de desarrollo y en términos económicos si la producción no es masiva [Baturone 00].

Los continuos avances en las tecnologías de fabricación de circuitos integrados hacen hoy día posible la inclusión de todos los componentes de un sistema microelectrónico en un único chip de silicio, dando lugar a lo que se denomina comúnmente como *System on Chip* (SoC). Sin embargo, como consecuencia de la elevada complejidad de los sistemas actuales, la diferencia entre los recursos proporcionados por las tecnologías y la productividad alcanzada por los diseñadores se hace cada vez mayor. Para reducir este *"gap de productividad"* se han propuesto nuevas técnicas de diseño que explotan la reutilización de bloques básicos previamente diseñados, denominados módulos de propiedad intelectual o módulos-IP [Savage 00] [Reyneri 03]. El diseño basado en módulos-IP, en combinación con el uso de potentes herramientas de CAD que facilitan la realización de los sistemas, permite asimismo reducir la duración del ciclo de desarrollo de nuevos productos y acelerar su introducción en el mercado, lo que representa ventajas importantes desde el punto de vista económico. En este sentido, los diseños que actualmente pueden ser costosos como ASICs difusos pueden tener más sentido como *"hard IPs"*, es decir, como módulos IP específicos dentro de un SoC.

Otra gran línea de investigación y desarrollo es la basada en dispositivos lógicos programables complejos tipo FPGAs (Field Programmable Gate Arrays), como se pone de manifiesto por los numerosos trabajos propuestos en los últimos años que utilizan las FPGAs como plataforma final en sus diseños [Monmasson 07] [Rodríguez 07] [Monmasson 11]. Concretamente, dentro del ámbito de la lógica difusa, han surgido en la actualidad una gran cantidad de trabajos enfocados a la implementación de siste-

mas difusos sobre estos dispositivos para distintos campos de aplicación [Chowdhury 08] [Castro 09] [Sánchez 13]. Las últimas familias de FPGAs disponibles en el mercado son cada vez más complejas incorporando una elevada cantidad de recursos genéricos (en forma de bloques lógicos configurables), específicos (generadores de reloj, multiplicadores, memorias, etc.), procesadores y bloques para DSP. Esto ha motivado la aparición de arquitecturas eficientes de codiseño hardware/software basadas en combinar los bloques lógicos de la FPGA con un procesador empotrado sobre la misma FPGA, consiguiendo la implementación de un sistema completo sobre un chip programable (SoPC). En sistemas difusos y neuro-difusos se han presentado diferentes trabajos que utilizan estas metodologías de codiseño hardware/software. Algunas de estas propuestas utilizan procesadores *"hard-core"* como el procesador ARM incluido en la familia Excalibur de Altera [Echevarría 05] [Chowdhury 08] [del Campo 08]. Otras utilizan procesadores *"soft-core"* como los procesadores MicroBlaze de Xilinx [Castro 09] [Sánchez 13] y Nios de Altera [Huang 09] [Fu 10].

De forma paralela a la evolución experimentada por las FPGAs, en las metodologías de diseño para estos dispositivos también se han producido una serie de cambios significativos en los últimos años. Como alternativa a los flujos de diseño basados en esquemáticos y lenguajes de descripción de hardware [Millán 08] [Oliveira 10] [Zhang 10], recientemente han aparecido una serie de herramientas, como el entorno *System Generator* de Xilinx, que facilitan el diseño de algoritmos de sistemas de procesado digital de señal sobre FPGAs, y son cada vez más habituales de utilizar en la realización de hardware difuso [Lizárraga 08] [Sepúlveda 09] [Brox 13b].

Para automatizar el diseño de sistemas neuro-difusos se han desarrollado numerosas herramientas software. Comenzando por las ya nombradas *TIL Shell* de Togai Infralogic Inc., *FIDE* de Aptronix o *FuzzyTECH* de Inform, pasando por los módulos (toolboxes) de Matlab o Mathematica, se han desarrollado muchas herramientas más o menos generales o específicas. Entre ellas se encuentran las herramientas del entorno Xfuzzy 3 desarrollado en el Instituto de Microelectrónica de Sevilla [Xfuzzy]. Una revisión reciente sobre software para el diseño de sistemas difusos puede verse en [Alcalá 15].

1.4. Conclusiones

Los sistemas neuro-difusos pueden ser una solución para multitud de problemas en diversos campos de aplicación, porque combinan la capacidad de aprender de datos numéricos de los sistemas neuronales y de conceptos simbólicos de los sistemas difusos, la capacidad de procesar información de forma numérica a la vez que ambigua y aproximada y, además, la capacidad de representar el conocimiento de una forma estructurada, en forma de reglas *"si-entonces"*, que pueden ser entendidas por los usuarios de estos sistemas.

De entre la multitud de técnicas que permiten obtener sistemas neuro-difusos a partir de datos numéricos, en los capítulos siguientes se detallarán aquellas que permiten obtener bases de reglas con pocas reglas, pocas funciones de pertenencia por entrada y, en general, bases de reglas que puedan ser entendidas.

Puesto que, en una gran cantidad de casos, los sistemas neuro-difusos diseñados serán sistemas polinómicos a tramos, las operaciones necesarias serán la suma y el producto y el número de reglas activas simultáneamente no será muy elevado. En consecuencia, se verán realizaciones eficientes como software ejecutado en procesadores de propósito general de bajas prestaciones (un solo núcleo de procesado, arquitectura de 32 bits y sin unidad de punto flotante) y realizaciones como hardware dedicado que, fundamentalmente, harán uso de sumadores y multiplicadores con aritmética de punto fijo. Los ejemplos de implementación se ilustrarán sobre FPGAs que podrán ser de bajo coste. Los entornos de diseño empleados serán Xfuzzy 3, Matlab/Simulink e ISE Design Suite de Xilinx.

Capítulo 2

Simplificación de bases de reglas en sistemas neuro-difusos

Uno de los objetivos primordiales de esta Tesis es simplificar sistemas neuro-difusos en lo posible, sin afectar significativamente a su comportamiento, para facilitar su implementación. Para ello, se buscará la forma de reducir en gran medida el número de reglas y también el de las funciones de pertenencia de las variables involucradas. Además, para conseguir una implementación eficiente, las funciones de pertenencia deben ser simples (como triangulares) y normalizadas (que los grados de pertenencia a las distintas funciones sumen la unidad). Logrando que un sistema cumpla estos requisitos no sólo se consigue una implementación eficiente, sino que se facilita su interpretabilidad lingüística. Esto se debe a que, entre las distintas condiciones reportadas en la literatura para que un sistema sea lingüísticamente interpretable, se remarca que el número de reglas sea moderado y que las funciones de pertenencia sean distinguibles, estén normalizadas y cubran al completo el universo de discurso de las variables [Zhou 08] [Gacto 11].

Es habitual que un sistema neuro-difuso emplee funciones de pertenencia así como reglas *"si-entonces"* que puedan reducirse para obtener un sistema con similar, incluso mejor, comportamiento. Esto sucede, por ejemplo, cuando se extrae una base de reglas a partir de datos numéricos usando técnicas de clustering. Es habitual que, tras la proyección de los clusters, se obtengan funciones de pertenencia como las indicadas en la Figura 2.1a. Esto sucede también cuando se aplica aprendizaje supervisado a los consecuentes de las reglas, como se ilustra en la Figura 2.1b, se haya obtenido la base de reglas a partir de conocimiento heurístico o numérico. Y no sólo aparece esta redundancia de información en las funciones de pertenencia empleadas, sino también en las reglas.

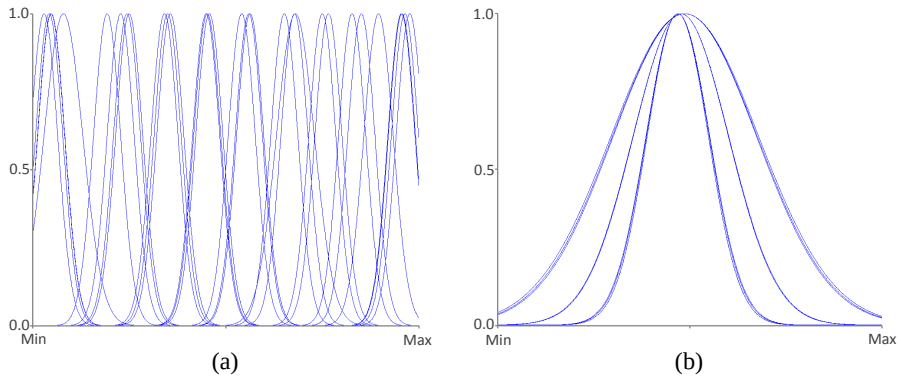

Figura 2.1: Ejemplos de funciones de pertenencia que presentan claras redundancias

Incluso empleando conocimiento heurístico es fácil expresar reglas lingüísticas que son redundantes.

2.1. Trabajos previos

2.1.1. Simplificación de funciones de pertenencia

La existencia de funciones de pertenencia muy parecidas y claramente simplificables en alguna de las variables de un sistema difuso conlleva la utilización de etiquetas lingüísticas innecesarias, pues denotan conceptos semánticamente equivalentes, como bien puede verse en la Figura 2.1. Por lo tanto, dificultan la interpretabilidad del sistema y, potencialmente, pueden también causar la existencia de reglas redundantes.

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

La simplificación por clustering busca un número reducido de funciones de pertenencia prototipo con las que representar todas las funciones originales. En general, conviene aplicar un algoritmo de clustering no-difuso o *hard clustering* (usualmente el Hard C-Means), porque cada función de pertenencia original será remplazada por una sola de las funciones prototipo encontradas. Este algoritmo aplica la minimización de una función objetivo como la ecuación (1.3), tal y como se vio en el capítulo anterior.

Los clusters se buscan en el espacio formado por los parámetros que definen las funciones de pertenencia. Un ejemplo es el de las funciones gaussianas, que se definen mediante sus centros y anchuras. Encontrar el número óptimo de funciones prototipo es complicado. Para ayudar a encontrar este número óptimo se utilizan, como ya se comentó en el capítulo anterior, los índices de validación directos. Ejemplos de estos índices se pueden encontrar en la siguiente lista:

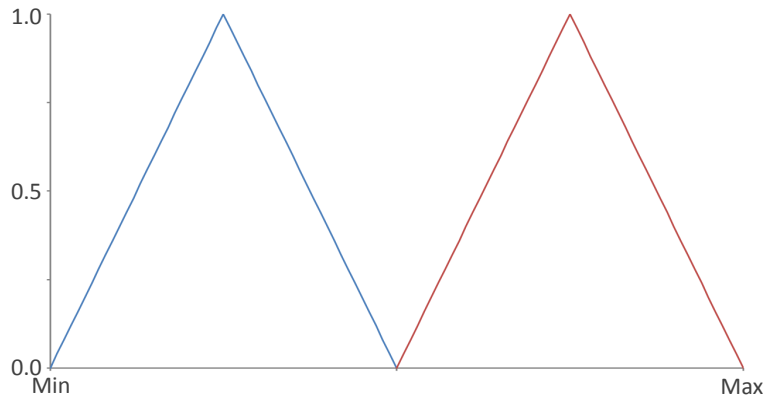

Figura 2.2: Conjuntos difusos con grado de similitud nulo

- Índice de Davies-Bouldin [Davies 79]. Es uno de los índices más empleados. Evalúa la proporción entre la suma del esparcimiento dentro de los clusters y la separación entre los mismos.

- Índice de Dunn [Dunn 73] [Dunn 74]. Junto al índice de Davies-Bouldin, es otro de los índices más clásicos. En este caso se trata de un índice geométrico que mide cómo de compactos y bien separados están los clusters en el conjunto de datos.

- Índices de Dunn generalizados [Bezdek 98]. Bezdek y Pal, considerando distintos tipos de medidas de la distancia entre clusters y de sus diámetros, consiguieron una familia de índices generalizados de Dunn, que dan buenos resultados para clusters volumétricos relativamente compactos y bien separados.

Otra forma de simplificar funciones de pertenencia es por similitud. Se define el concepto de *similitud* entre dos conjuntos difusos como el grado en el que dichos conjuntos difusos son iguales. Esta definición está relacionada con los conceptos que representan los conjuntos difusos. Por ejemplo, considerando los conjuntos difusos de la Figura 2.2, se puede comprobar que tienen la misma forma, pero representan conceptos completamente distintos. Su grado de similitud, en este caso, se considera nulo.

Por otro lado, se pueden considerar los dos conjuntos difusos de la Figura 2.3a, en los que se puede constatar que representan conceptos que se solapan en un alto grado, aunque la forma que los define sea diferente. En este caso, representan conceptos similares.

La simplificación por similitud consiste en identificar de dos en dos los conjuntos difusos que tengan un grado de similitud que está por encima de un valor umbral y sustituirlos por un conjunto difuso común representativo de los originales (Figura 2.3b). El proceso termina cuando ya se han evaluado todas las posibles parejas de funciones

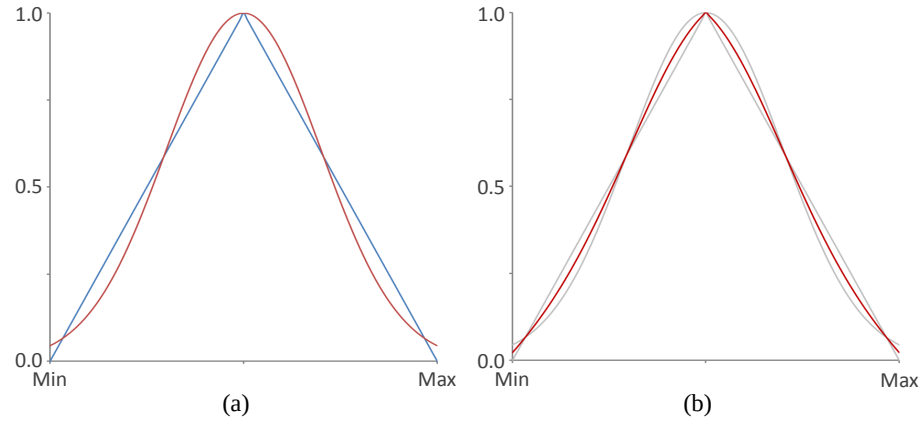

Figura 2.3: (a) Conjuntos difusos con un grado alto de similitud; (b) Conjunto difuso representativo de dos conjuntos difusos similares

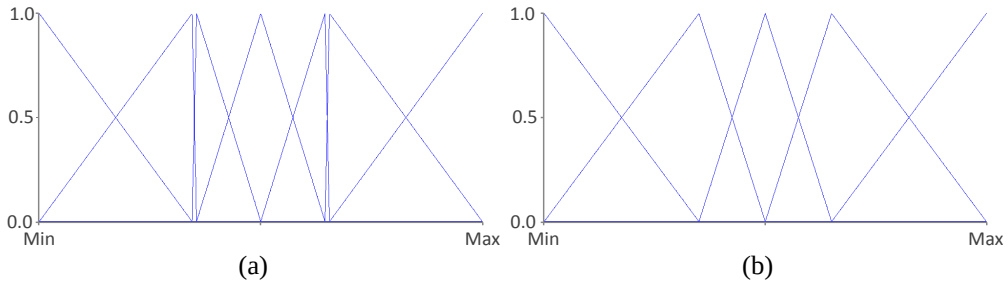

Figura 2.4: (a) Funciones de pertenencia triangulares con vértices cercanos; (b) Resultado de aplicar el algoritmo *FuZion*

a simplificar. Mientras el umbral sea más bajo, mayor será el número de funciones simplificadas [Setnes 98].

Otro algoritmo de simplificación de tipos, planteado en [Espinosa 00], es el algoritmo *FuZion*. En él, se presenta una rutina que fusiona funciones de pertenencia triangulares con vértices muy cercanos (Figura 2.4a). Determinada una distancia mínima aceptable, el algoritmo fusiona aquellas funciones de pertenencia cuyos vértices estén a una distancia menor (Figura 2.4b). Esta simplificación es apropiada para funciones de pertenencia vinculadas a variables de salida, pero no lo es si las funciones están vinculadas a variables de entrada.

Para conseguir un número reducido de tipos, en [Castellano 02] realizan clustering en dos niveles. Concretamente, identifican los centros de clusters tal y como se explicó en el apartado 1.1.1. La novedad consiste en que esos centros son proyectados en los espacios de cada una de las entradas, obteniendo prototipos unidimensionales en cada dimensión del espacio de las entradas. Llegados a este punto, vuelven a aplicar clustering sobre estas proyecciones de los prototipos, en cada una de las dimensiones del espacio de las

entradas, obteniendo un número de tipos final bastante reducido.

En [Zhou 06], consideran que la mayoría de procedimientos que fusionan tipos de las funciones de pertenencia en las entradas, no tienen en cuenta la información contenida en los datos de salida. Por ello, para conseguir un particionado del espacio de entrada con distinguibilidad, utilizan mediciones de entropía *"local"*. En el esquema que plantean, hacen balance entre aproximación y dinstinguibilidad (basándose en las medidas de entropía *"local"*), teniendo en cuenta la información aportada por los pares de datos entrada/salida.

Aunque la medida más común para cuantificar la distinguibilidad es la similitud aportada por [Setnes 98], el problema que tiene es el elevado coste computacional que suele conllevar. En este sentido, en [Mencar 07] proponen como alternativa utilizar una medida de posibilidad que, bajo ciertas condiciones, puede utilizarse para evaluar la distinguibilidad de una forma más rápida y eficiente que por similitud.

Por último, la simplificación por purga se trata de una simplificación trivial. Debido a procesos de simplificación de reglas, puede ocurrir que distintas funciones de pertenencia se vuelvan innecesarias, debido a que no son utilizadas en ninguna de las reglas de las bases de reglas del sistema. La simplificación por purga se limita a eliminar dichas funciones de pertenencia.

2.1.2. Simplificación de reglas

En el campo del diseño de sistemas difusos se han llevado a cabo enormes esfuerzos (especialmente en lo que concierne a algoritmos adaptativos de aprendizaje de reglas) para conseguir el mínimo conjunto de reglas difusas [Zhou 08] [Gacto 11]. En [Goodman 92], el compromiso entre un conjunto reducido de reglas y uno adecuado está inscrito en el contexto de la teoría de la información. Para aplicar ese enfoque en el contexto de la lógica difusa, se tiene que asumir la existencia de una expresión de coste para cada conjunto de reglas candidato que se trata de minimizar.

Una técnica, abordada por varios autores, es la de aplicar métodos de transformación ortogonal a la base de reglas, para así seleccionar las más importantes [Yen 99]. Los primeros en aplicar este tipo de técnicas, en el ámbito de la lógica difusa, fueron Wang y Mendel en [Wang 92a], proponiendo un algoritmo de mínimos cuadrados ortogonales. En ese trabajo, introdujeron el concepto de *"funciones de base difusas"* por el que, fijando ciertos parámetros, un sistema difuso puede representarse de forma equivalente por una expansión en serie de estas funciones base. A esta expansión en serie, le aplican el procedimiento clásico de ortogonalización de Gram-Schmidt, para determinar las funciones de base difusas más significativas. El hecho de utilizar funciones de base difusas en lugar de otras funciones base (como polinómicas, radiales, etc.), es porque cada una de las reglas del sistema se puede relacionar de forma natural con estas funciones base. Finalmente, el

método selecciona las reglas más importantes basándose en la contribución a la varianza de éstas sobre la varianza de la salida.

El problema que plantea este método es que, un valor de varianza bajo, no tiene por qué significar necesariamente que la regla correspondiente sea poco importante. Es por eso que, posteriormente, aparecieron trabajos para mejorar la aplicación del método de mínimos cuadrados ortogonales [Setnes 00] [Mastorocostas 01]. De estos trabajos, el más actual es el propuesto por Destercke y coautores en [Destercke 07], para conseguir sistemas difusos de tipo Takagi-Sugeno de orden cero a partir de datos, con una base de reglas interpretable. Para ello, aplican el algoritmo de mínimos cuadrados dos veces: la primera vez para seleccionar las reglas más importantes y la segunda para optimizar los consecuentes de las reglas. Después, mediante el algoritmo de Hard C-means, reducen el número de consecuentes.

Es habitual que los procesos de ajuste y simplificación de funciones de pertenencia den lugar a reglas similares que pueden ser eliminadas a continuación de la base de reglas. Las técnicas comúnmente empleadas para ello son las denominadas técnicas *"de podado"*. Consisten en llevar a cabo el tratamiento de una base de reglas para que, dado un conjunto de datos de entrada, determinar el grado de activación de las reglas para esas entradas y poder eliminar las n peores reglas (con grado de activación más bajo), seleccionar las n mejores (con grado de activación más alto) o bien imponer un umbral fijo que deba ser superado por el grado de activación de cada regla para que ésta permanezca en la base de reglas.

Varios autores han propuesto técnicas de extracción de reglas que clasifican las reglas mediante otros índices de comportamiento [Nauck 00] [Senhadji 02]. Por ejemplo, el índice propuesto en [Nauck 00] es:

$$P_r = \frac{1}{s} \sum_{p \in \mathcal{L}} \alpha_r(p) \cdot e_r(p) \tag{2.1}$$

$$e_r(p) = \begin{cases} 1 & \text{si } p \text{ está bien clasificado por } r \\ -1 & \text{en otro caso} \end{cases} \tag{2.2}$$

Donde $\mathcal{L}$ es el conjunto de datos del que se extraen las reglas; p es uno de los datos que pertenece a ese conjunto; $s = |\mathcal{L}|$ es la cardinalidad del conjunto de datos de entrenamiento $\mathcal{L}$; y $\alpha_r(p)$ es el grado de activación de la regla r para el patrón p. Se puede ver en la expresión (2.2) que es un índice aplicable a reglas para clasificación.

Definido este índice de comportamiento, desarrollan el algoritmo *NEFCLASS*, que se encarga de calcular el índice de comportamiento $P \in [-1, 1]$ para cada regla. Para $P = 1$ una regla se considera general y que clasifica todos los patrones de entrenamiento de forma correcta. Para $P = -1$, la regla clasifica todos los patrones de forma incorrecta. Para $P = 0$, bien el número de clasificaciones correctas y erróneas es igual, o bien la regla

no cubre ningún patrón. Por tanto, consideran únicamente las reglas con $P > 0$ como útiles. De forma análoga, para bases de reglas interpoladoras, en [Nauck 00] desarrollan el algoritmo *NEFPROX*.

Por otro lado, Senhadji y coautores desarrollan en [Senhadji 02] el algoritmo *NORFREA*. Este algoritmo parte de clasificar las reglas con el mismo índice de comportamiento propuesto en [Nauck 00]. La novedad consiste en que Senhadji y coautores se dieron cuenta que, tras seleccionar las n mejores reglas según el índice de comportamiento de las ecuaciones (2.1) y (2.2), si se aplicaba entrenamiento a ese resultado algunas reglas se volvían redundantes. Por tanto, proponen un método que detecta antes del entrenamiento cuáles reglas serán redundantes después del entrenamiento, siendo eliminadas y ocupando su lugar otras reglas que habían quedado fuera de las n mejores según [Nauck 00]. Con esta nueva selección de reglas, los resultados de *NORFREA* tras aplicar entrenamiento son mejores que los obtenidos por *NEFCLASS*.

2.2. Referencias bibliográficas del autor

- Arjona R., Gersnoviez A., Baturone I., *Fuzzy models for fingerprint description*, Lecture Notes in Computer Science, vol. 6857, pp. 228-235, 2011.
- Baturone I., Gersnoviez A., *Automatic extraction of linguistic models for image description*, in Proceedings of the 2010 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'10), pp. 1-8, 2010.
- Baturone I., Moreno-Velo F.J., Sánchez-Solano S., Barriga A., Brox P., Gersnoviez A., Brox M., *Using Xfuzzy environment for the whole design of fuzzy systems*, in Proceedings of the 2007 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'07), pp. 1-6, 2007.
- Baturone I., Moreno-Velo F.J., Gersnoviez A., *A CAD approach to simplify fuzzy system descriptions*, in Proceedings of the 2006 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'06), pp. 2392-2399, 2006.
- Barriga A., Sánchez-Solano S., Baturone I., Moreno-Velo F.J., Brox P., Montesino F., Hussein N.M., Brox M., Gersnoviez A., *Fuzzy logic activities at the Microelectronics Institute of Seville*, Lecture Notes in Computer Science, vol. 3931, pp. 157-162, 2006.
- Barriga A., Sánchez-Solano S., Baturone I., López D., Moreno-Velo F.J., Brox P., Montesino F., Hussein N.M., Brox M., Gersnoviez A., *New features of the fuzzy logic development environment Xfuzzy*, in Proceedings of the 2006 International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'06), pp. 17-20, 2006.

- Baturone I., Moreno-Velo F.J., Gersnoviez A., *Identifying fuzzy systems from numerical data with Xfuzzy*, in Proceedings of the 2005 Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT'05), pp. 1257-1262, 2005.

Capítulo 3

Simplificación de sistemas neuro-difusos mediante el uso de jerarquía

Como ya se explicó en el capítulo 1, uno de los mayores problemas de los sistemas neuro-difusos con partición tipo rejilla de los universos de discurso de entrada es la maldición de la dimensionalidad. La inviabilidad de implementar sistemas demasiado complejos, no sólo por la saturación de los elementos de memoria, sino por la ralentización debida a la elevada cantidad de reglas a procesar, lleva a pensar en diversos métodos de simplificación. En el capítulo anterior se abordó la estrategia de simplificación de bases de reglas, introduciendo un novedoso método de simplificación tabular. Aún así, no es la única manera de conseguir simplificar sistemas neuro-difusos, existiendo otros métodos que pueden ser utilizados de forma complementaria.

Un método que está en alza en los últimos años es el de dividir un único sistema neuro-difuso en varios subsistemas con un número menor de reglas. Esta estrategia fue introducida por primera vez por Raju y coautores en [Raju 91], denominando a esta descomposición de sistemas como *"sistemas jerárquicos"*. Una de las razones que hace tan atractiva esta técnica es que se consigue evitar el aumento exponencial de las reglas a medida que se incrementa el número de entradas, alcanzando un crecimiento lineal. Pero, como se verá a lo largo del capítulo, la mayor dificultad que plantea esta técnica es encontrar la estructura más adecuada que permita una mayor simplificación.

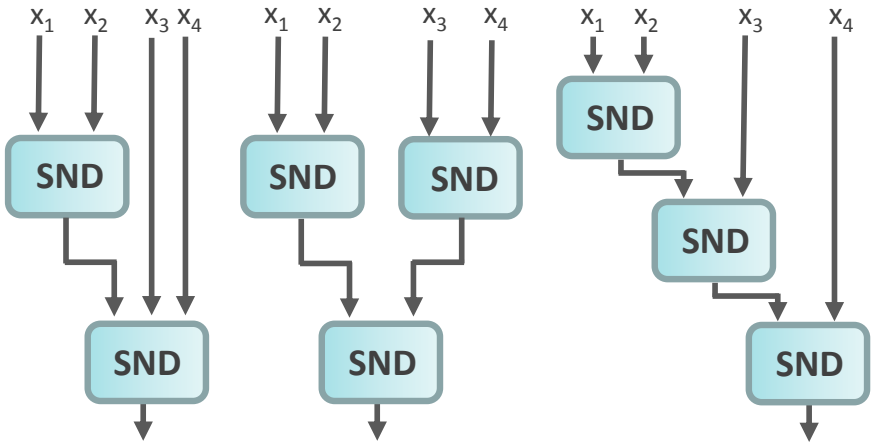

Figura 3.1: Distintos tipos de organización jerárquica

3.1. Trabajos previos

Desde la propuesta de Raju en 1991, no son muchos los autores que han estudiado teóricamente las propiedades de los sistemas jerárquicos. En este apartado se realizará una revisión de las contribuciones más relevantes desde el artículo de Raju hasta la actualidad.

Como ya se ha comentado, el uso de la jerarquía consiste en descomponer un sistema en una serie de módulos más simples. Dicha descomposición no es trivial, ya que dado un número de entradas, hay diversas arquitecturas posibles para descomponer el sistema [Torra 02], tal y como se puede ver en la Figura 3.1, aumentando considerablemente las posibilidades con el número de entradas. Otras opciones no reflejadas en la Figura 3.1 son que distintos módulos compartan algunas entradas, o que los módulos tengan más de una salida. Las distintas estructuras pueden clasificarse en tres tipos distintos: estructura en serie, en paralelo e híbrida [Aja 08]. En la estructura de tipo serie (también conocida como incremental) hay un único bloque por nivel de jerarquía, en el que entran la salida del bloque del nivel anterior, junto con variables de entrada del sistema (Figuras 3.1a y 3.1c). En la estructura de tipo paralelo, todas las variables de entrada del sistema acceden a los bloques que componen el primer nivel de jerarquía. Las salidas de los bloques del primer nivel, serán las entradas de los bloques que componen el segundo nivel, y así sucesivamente (Figura 3.1b). Por último, la estructura de tipo híbrida es una combinación de las dos anteriores.

Cuando se conoce la relación entre las distintas entradas, uno puede orientarse para

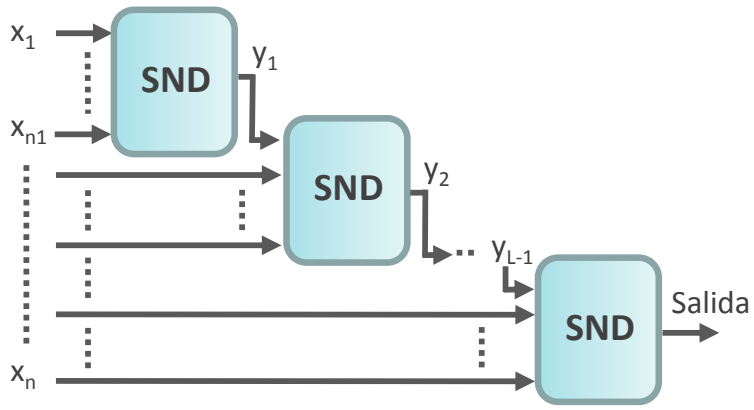

Figura 3.2: Tipo de jerarquía propuesta por Raju y coautores

encontrar la arquitectura más adecuada, pero en muchas ocasiones no se conoce esta relación, complicando en gran medida el problema. De hecho, no todas las funciones admiten una descomposición jerárquica (como, por ejemplo, algunas funciones Booleanas [Kikuchi 02]). Por lo tanto, uno de los grandes retos es hallar una estrategia que permita encontrar una estructura jerárquica adecuada para un sistema dado sin jerarquía, de modo que se mantenga su comportamiento, pero simplificándolo en la mayor medida posible.

3.1.1. Análisis de Raju, Zhou y Kisner

Supóngase un sistema con n entradas y que cada una de esas entradas tiene m etiquetas (se considera el mismo número de etiquetas para todas las entradas para simplificar la notación, obviamente no tiene por qué cumplirse). Si el sistema no es jerárquico, el número de reglas que puede llegar a tener es m^n. La propuesta de Raju y coautores es una descomposición jerárquica del sistema en módulos que no comparten entradas y tienen una única salida. Si el número de bloques jerárquicos es L, es fácil comprobar que el número total de reglas del sistema jerárquico será:

$$r_J = \sum_{i=1}^{L} m^{n_i} \tag{3.1}$$

Donde n_i es el número de entradas para el bloque i-ésimo.

El tipo de arquitectura propuesta es de tipo serie, como puede observarse en la Figura 3.2, en donde las variables más influyentes se agrupan en el primer nivel de jerarquía, las siguientes más influyentes se agrupan en el segundo nivel y así sucesivamente. Teniendo en cuenta que, de las n_i entradas del bloque i-ésimo, una de ellas es la salida del bloque

del nivel anterior (salvo para el caso del primer bloque), se debe cumplir la siguiente relación:

$$n_1 + \sum_{i=2}^{L} (n_i - 1) = n \tag{3.2}$$

Si se supone que n_i es un valor constante c ($\forall i = 1, \ldots, L$), sustituyendo en la ecuación (3.2), queda lo siguiente:

$$c + (L-1) \cdot (c-1) = n \Rightarrow L = \frac{n-c}{c-1} + 1 = \frac{n-1}{c-1} \tag{3.3}$$

Llevando el resultado de (3.3) a (3.1), se obtiene:

$$r_J = \sum_{i=1}^{L} m^{n_i} = \sum_{i=1}^{L} m^c = L \cdot m^c = \frac{n-1}{c-1} \cdot m^c \tag{3.4}$$

Como c y m son valores constantes, se puede apreciar que el crecimiento del número de reglas para el sistema jerárquico es una función lineal de n, en contraste con el crecimiento exponencial del sistema original.

Raju y coautores formulan, además, el siguiente teorema:

Teorema 3.1 *En la estructura jerárquica del sistema con n variables de entrada, $m \geq 2$ y $n_i \geq 2$, (a) el número total de reglas alcanzará el valor mínimo cuando $n_i = c = 2$ ($\forall i = 1, \ldots, L$), $L = n - 1$; y (b) alcanzará el máximo cuando $L = 1$ y, por tanto, $n_i = n_1 = n$.*

Para demostrarlo, considérese el nivel i-ésimo y sus n_i entradas y supóngase $n_i \geq 3$ (para poder dividirlo en bloques jerárquicos más pequeños). Este bloque, tal y como se ha comentado anteriormente, tendrá un total de m^{n_i} reglas.

Si se divide este nivel en dos bloques jerárquicos, de modo que uno de ellos sólo tenga dos entradas, el segundo bloque tendrá $n_i - 1$ (ya que serían las $n_i - 2$ entradas restantes del nivel i-ésimo, junto a la salida del bloque de dos entradas). Por tanto, el número total de reglas de este nuevo sistema jerárquico será:

$$r_{J_{2/(n_i-1)}} = m^2 + m^{n_i - 1} \tag{3.5}$$

Como se ha señalado que $n_i \geq 3$, se tiene que $n_i - 1 \geq 2$. Por tanto, llevando esto a la ecuación (3.5) junto a la relación $m \geq 2$:

$$r_{J_{2/(n_i-1)}} = m^2 + m^{n_i-1} \leq m^{n_i-1} + m^{n_i-1} = 2 \cdot m^{n_i-1} \leq m^{n_i} \tag{3.6}$$

Con esto se concluye que, cogiendo un sistema cualquiera, al dividirlo en dos bloques jerárquicos (siendo uno de esos bloques de dos entradas), el nuevo sistema tendrá un

número de reglas menor. Con lo cual, si se selecciona el bloque con más entradas del nuevo sistema y se vuelve a dividir siguiendo la misma estrategia una y otra vez, se puede llegar a la conclusión que el número de reglas alcanzará el valor mínimo cuando el sistema esté compuesto únicamente por bloques jerárquicos de dos entradas.

Este razonamiento es válido siempre y cuando se considere que los bloques tienen como mínimo dos entradas. Como ya se verá más adelante, existen estructuras jerárquicas con menos reglas si los bloques tienen una única entrada.

Siguiendo con el teorema, falta por demostrar el caso contrario, es decir, que el máximo se consigue cuando no hay descomposición jerárquica ninguna. Para ello considérense los bloques jerárquicos i-ésimo y j-ésimo, para luego combinarlos en un sistema único no jerárquico. Si el bloque i-ésimo tiene n_i entradas y el j-ésimo n_j, el nuevo sistema combinado tendrá un total de $n_i + n_j - 1$ entradas (ya que dentro de las n_j entradas se cuenta también la salida del bloque i-ésimo). Por tanto, el sistema no jerárquico tendrá un total de reglas:

$$r_{ij} = m^{n_i+n_j-1} \tag{3.7}$$

Teniendo en cuenta que los bloques tendrán como mínimo dos entradas, se tiene que $n_i \geq 2$ y, de la misma manera, $n_j - 1 \geq 1$. Manteniendo la relación $m \geq 2$ y suponiendo $n_i \geq n_j$, se puede partir del número de reglas total del sistema jerárquico inicial, tal que:

$$r_{i/j} = m^{n_i} + m^{n_j} \leq m^{n_i} + m^{n_i} = 2 \cdot m^{n_i} \leq m^{n_i+1} \leq m^{n_i+n_j-1} = r_{ij} \tag{3.8}$$

Por tanto, uniendo dos bloques jerárquicos cualesquiera en un sistema no jerárquico se obtiene un número de reglas mayor. Si este nuevo sistema, a su vez, se combina con otro bloque jerárquico, el sistema obtenido tendrá un número mayor de reglas. Continuando con este razonamiento, se llega a la conclusión que el número máximo de reglas se conseguirá cuando se hayan combinado todos los bloques jerárquicos, es decir, cuando se llega al sistema completo sin jerarquía.

3.1.2. Sistemas jerárquicos como aproximadores universales

El que se pueda conseguir altos grados de simplificación mediante el uso de jerarquía es bastante interesante, pero es fundamental que estos nuevos sistemas mantengan la propiedad de aproximación universal. Desde la presentación de los sistemas jerárquicos por parte de Raju y coautores, estudios teóricos posteriores se han centrado precisamente en las propiedades de aproximación de estos sistemas.

3.1.2.1. Teoría de la aproximación universal de Wang

Una aportación significativa a la hora de demostrar la propiedad de los sistemas jerárquicos como aproximadores universales la desarrolló Li-Xin Wang en sus trabajos [Wang 98] y [Wang 99].

Wang parte de la estructura planteada por Raju y coautores, pero utilizando sistemas difusos de tipo Takagi-Sugeno. Con esto, plantea el siguiente teorema de aproximación universal para el caso de tres variables, pudiéndose generalizar al caso de n variables:

Teorema 3.2 *Para cualquier función continua y diferenciable $g(x)$ en el conjunto compacto $U = \prod_{i=1}^{3} [\alpha_i, \beta_i]$ $(U \subset \mathbb{R}^3)$ y para cualquier $\epsilon > 0$, existe un sistema difuso jerárquico $f(x_1, x_2, x_3)$ de la forma:*

$$f(x_1, x_2, x_3) = f_2(f_1(x_1, x_2), x_3) \tag{3.9}$$

Tal que:

$$\|g(x) - f(x)\|_\infty < \epsilon \tag{3.10}$$

Donde f_1 y f_2 se corresponden con sistemas difusos de tipo Takagi-Sugeno y la norma infinito $\| * \|_\infty$ se define como $\|h(x)\|_\infty = \sup_{x \in U} |h(x)|$.

Para demostrar este teorema, Wang hace una demostración previa, desarrollada en [Wang 98]:

$$\|g(x) - f(x)\|_\infty \leq \sum_{i=1}^{3} \left(\left\| \frac{\partial g}{\partial x_i} \right\|_\infty + \left\| \frac{\partial f}{\partial x_i} \right\|_\infty \right) \cdot \left(\frac{\beta_i - \alpha_i}{m - 1} \right) \tag{3.11}$$

Siendo m el número de etiquetas para cada una de las entradas del sistema.

Como $g(x)$ y $f(x)$ son continuas y $\|\partial g/\partial x_i\|_\infty$, $\|\partial f/\partial x_i\|_\infty$, α_i y β_i son valores finitos, sólo hace falta elegir un número de etiquetas m lo suficientemente alto para que (3.11) sea menor que cualquier $\epsilon > 0$. Quedando demostrado el teorema de aproximación universal.

Aunque Wang consigue demostrar la capacidad de aproximación universal del modelo jerárquico que propone, no consigue escapar de la maldición de la dimensionalidad. Él mismo concluye que, aunque el número total de reglas en el sistema difuso jerárquico es mucho menor, cada regla es más complicada, porque la parte de los consecuentes se convierte en polinomios de orden m de las variables de entrada. Teniendo en cuenta que la base de la demostración de su teorema consiste en elegir un valor de m lo suficientemente alto, mientras más se quiera aproximar una función, más complejas serán las reglas necesarias. Por tanto, Wang sustituye un modelo no jerárquico con muchas reglas sencillas por un modelo jerárquico con menos reglas pero más complejas.

3.1.2.2. Teoría de la aproximación universal de Joo y Lee

Uno de los inconvenientes que preocupa a diversos autores, respecto a la estructura de la Figura 3.2, es la dificultad a la hora de otorgar interpretabilidad lingüística a las variables intermedias $(y_1, \ldots, y_{L-1})$. Si estas variables carecen de interpretabilidad lingüística y son usadas como variables de entrada en subsistemas siguientes, las reglas de éstos tendrán escaso significado lingüístico, implicando una dificultad añadida a la hora de diseñarlos.

Para solucionar este problema, una opción que surge es la de no utilizar estas variables intermedias en la parte de los antecedentes. En su lugar, estas variables modificarán las reglas en la parte de los consecuentes de los subsistemas en los que entren. De este modo, todos los antecedentes de todos los bloques estarán compuestos por variables principales del sistema, todas con interpretabilidad lingüística.

El aporte teórico más relevante utilizando sistemas jerárquicos de este tipo es el desarrollado por Joo y Lee en sus trabajos [Joo 02] y [Joo 05], en donde demuestran que su modelo tiene la propiedad de la aproximación universal.

La arquitectura planteada por Joo y Lee es la que se puede ver en la Figura 3.3, donde se aprecia una estructura de tipo híbrida. Según se puede ver en la figura, el sistema neuro-difuso SND-ij se refiere al sistema j-ésimo correspondiente a la capa i-ésima de jerarquía. Si $X = (x_1, x_2, \ldots, x_n)$ es el conjunto de variables de entrada del sistema, el subconjunto $X_{ij} = \left(x_1^{(ij)}, x_2^{(ij)}, \ldots, x_{n_{ij}}^{(ij)}\right)$, $X_{ij} \subset X$, representa aquellas entradas principales del sistema total que entran en el bloque SND-ij, formando íntegramente la parte de los antecedentes de dicho bloque. En este bloque, a su vez, también entrarán las salidas generadas por los bloques pertenecientes a la capa anterior de jerarquía, lo mismo que la salida generada por el bloque SND-ij (y_{ij}) entrará en los bloques pertenecientes a la capa siguiente. Como ya se comentó, estas variables intermedias modificarán las reglas en la parte de los consecuentes (lo cual se enfatiza en la Figura 3.3, entrando las variables principales del sistema por una zona distinta a la de las intermedias).

El número total de bloques jerárquicos será $\sum_{i=1}^{L} q_i$, donde q_i es el número de bloques jerárquicos que componen la capa i-ésima de jerarquía y L el número de capas.

Por último, en la jerarquía propuesta por Joo y Lee, una misma variable principal $x_k \in X$, puede entrar en distintos bloques de la jerarquía, cosa que no ocurría en las jerarquías usadas por Raju y Wang, donde cada variable principal entraba únicamente en un bloque jerárquico. Con esto, lo que pretenden Joo y Lee es conseguir reglas más simples, a costa de aumentar el número de reglas.

Para demostrar que su modelo tiene la propiedad de aproximación universal, Joo y Lee se basan en el trabajo desarrollado por Huwendiek y Brockmann [Huwendiek 99]. Huwendiek y Brockmann proponen un modelo muy específico de jerarquía (*Red de nodos difusos adaptativos - NetFAN*), y hacen uso del teorema de Stone-Weierstrass para

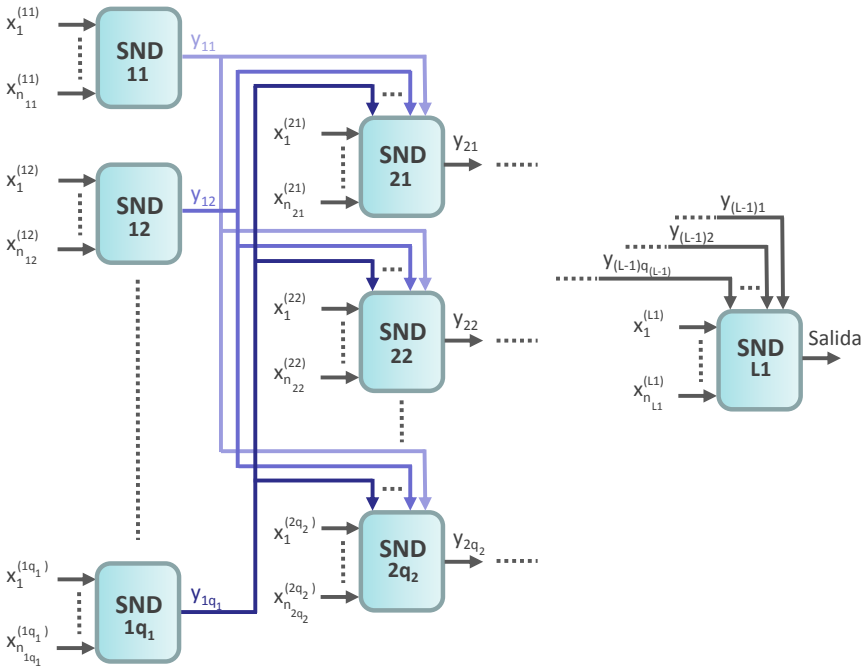

Figura 3.3: Tipo de jerarquía propuesta por Joo y Lee

demostrar que son aproximadores universales. Joo y Lee amplían la demostración de Huwendiek y Brockmann a su modelo contemplando, además, el uso de todo tipo de funciones de pertenencia para las variables de entrada (Huwendiek y Brockmann sólo lo demostraban para el caso de etiquetas trapezoidales).

El teorema de Weierstrass establece lo siguiente:

Teorema 3.3 (Teorema de aproximación de Weierstrass) *Suponiendo $g \in C(X)$, siendo $C(X)$ el conjunto de funciones reales continuas definidas en el conjunto compacto X. Para cualquier $\epsilon > 0$, existe un polinomio p sobre $C(X)$ tal que, para todo $x \in X$, se tiene:*

$$\|g(x) - p(x)\|_\infty < \epsilon \tag{3.12}$$

Más tarde, Stone se daría cuenta que el conjunto de polinomios $P(X) \subset C(X)$ no es el único conjunto de funciones pertenecientes a $C(X)$ que cumple el teorema de aproximación de Weierstrass. Concretamente, suponiendo un conjunto $H(X) \subset C(X)$, que cumple:

1. H es un álgebra. Es decir, el conjunto H es cerrado bajo la suma, el producto y el producto por un escalar.

2. H no desaparece en ningún punto de X. Esto es, $\forall x \in X, \exists h \in H | h(x) \neq 0$.

3. H separa puntos en X. Esto es, $\forall x, x' \in X, x \neq x', \exists h \in H | h(x) \neq h(x')$.

Entonces existe una función $h \in H$ tal que:

$$\|g(x) - h(x)\|_\infty < \epsilon \tag{3.13}$$

Esta última conclusión es la que establece el *Teorema de Stone-Weierstrass* [Rudin 76]. Como se puede observar, el conjunto de polinomios $P(X) \subset C(X)$ cumple todas las condiciones propuestas por Stone, siendo un caso particular incluido en el teorema de Stone-Weierstrass.

Dicho esto, lo que demuestran Huwendiek y Brockmann en [Huwendiek 99] es que el conjunto compuesto por todas las NetFAN cumple las condiciones de Stone. Por tanto, el modelo propuesto por ellos, cumple la propiedad de aproximación universal mediante el teorema de Stone-Weierstrass.

Esta misma estrategia es la que siguen Joo y Lee con sus respectivos modelos en [Joo 02] y [Joo 05]. Es decir, demuestran que el conjunto F, compuesto por las funciones de los sistemas difusos jerárquicos propuestos por ellos, cumplen las condiciones de Stone. El modelo propuesto en [Joo 02] sigue la estructura de la Figura 3.3, donde los sistemas son de tipo Takagi-Sugeno, con la parte de los consecuentes compuesta por múltiplos entre variables principales del sistema y las variables intermedias generadas en la capa jerárquica anterior. En [Joo 05] simplifican el modelo, consiguiendo que la parte de los consecuentes esté compuesta únicamente por múltiplos de las variables intermedias generadas en la capa jerárquica anterior.

Aunque la propuesta de Joo y Lee no deja de ser interesante, tiene sus inconvenientes. Es cierto que consiguen que la parte de los antecedentes sea simple y lingüísticamente interpretable, pero a costa de hacer la parte de los consecuentes más compleja y difícil de interpretar. Por otra parte, para conseguir una mayor interpretabilidad, aumentan el número de bloques jerárquicos respecto a la estructura de Raju y coautores. De hecho, para conseguir el orden de aproximación deseado, no se especifica cuántos bloques jerárquicos serán necesarios. Con lo que no se puede apreciar hasta qué punto se simplifica el sistema difuso en comparación con el sistema sin jerarquía.

3.1.2.3. Teoría de la aproximación universal de Zeng y Keane

Los estudios hasta ahora expuestos sobre aproximación universal hacen uso de sistemas difusos jerárquicos de tipo Takagi-Sugeno. El trabajo desarrollado por Zeng y Keane [Zeng 05b] difiere en esto último, ya que su estudio se basa en la jerarquía de los sistemas difusos de tipo Mamdani. No sólo es de gran relevancia este estudio, sino

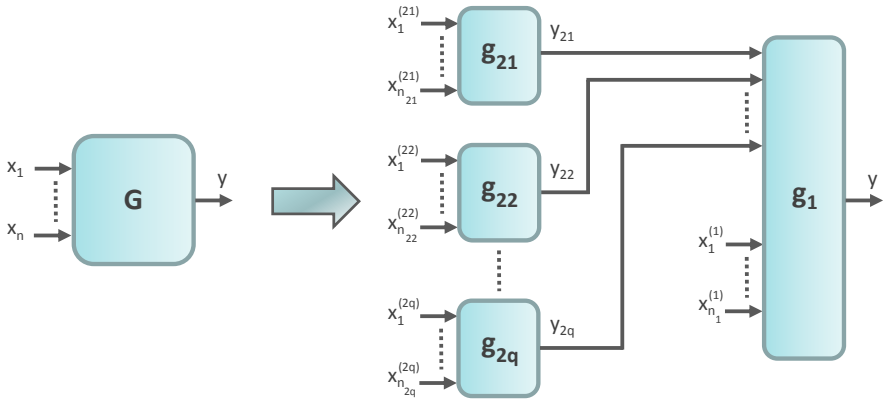

Figura 3.4: Función continua $G(x_1, \ldots, x_n)$ con estructura jerárquica natural

que en su trabajo introduce una serie de definiciones sobre la jerarquía de las funciones continuas que son de gran interés.

Concretamente, suponiendo $y = G(x_1, \ldots, x_n)$ una función continua con n entradas $X = (x_1, \ldots, x_n)$, definidas en el espacio $U = \prod_{i=1}^{n} U_i \subset \mathbb{R}^n$, generando la salida y definida en el espacio $V \subset \mathbb{R}$. Si existen $q + 1$ funciones continuas:

$$y = g_1 \left(y_{21}, \ldots, y_{2q}, x_1^{(1)}, \ldots, x_{n_1}^{(1)} \right) \tag{3.14}$$

$$y_{2j} = g_{2j} \left(x_1^{(2j)}, \ldots, x_{n_{2j}}^{(2j)} \right) \quad (j = 1, 2, \ldots, q) \tag{3.15}$$

cada una de ellas con un número de entradas menor que n, tal que:

$$G(x_1, \ldots, x_n) = g_1 \left[g_{21} \left(x_1^{(21)}, \ldots, x_{n_{21}}^{(21)} \right), \ldots, g_{2q} \left(x_1^{(2q)}, \ldots, x_{n_{2q}}^{(2q)} \right), \ldots, x_1^{(1)}, \ldots, x_{n_1}^{(1)} \right] \tag{3.16}$$

se dice que $G(x_1, \ldots, x_n)$ es una función continua con *estructura jerárquica natural* (Figura 3.4).

Supóngase ahora el subconjunto $X_1 = \left(x_1^{(1)}, \ldots, x_{n_1}^{(1)} \right) \subset X$ formado por las variables principales del sistema que entran en el primer nivel de jerarquía. A su vez, supóngase el subconjunto $X_{2j} = \left(x_1^{(2j)}, \ldots, x_{n_{2j}}^{(2j)} \right) \subset X$ formado por las variables principales del sistema que entran en el bloque j-ésimo del segundo nivel de jerarquía. Si se cumplen las condiciones:

$$X_1 \cap X_{2j} = \emptyset \quad (j = 1, 2, \ldots, q) \tag{3.17}$$

$$X_{2j} \cap X_{2k} = \emptyset \quad (j \neq k;\ j,k = 1,2,\ldots,q) \tag{3.18}$$

$$X_1 \cup X_{21} \cup X_{22} \cup \cdots \cup X_{2q} = X \tag{3.19}$$

se dice que $G(x_1,\ldots,x_n)$ es una función continua con *estructura jerárquica separable.*

Finalmente, supóngase la función continua $G(x_1,\ldots,x_n)$ y que, para cualquier partición arbitraria de X en $q+1$ subconjuntos disjuntos que cumplan las condiciones (3.17) - (3.19), existen funciones continuas g_1 y g_{2j} $(j = 1,\ldots,q)$ que satisfacen la ecuación (3.16). Entonces, se dice que $G(x_1,\ldots,x_n)$ es una función continua con *estructura jerárquica arbitrariamente separable.*

Todas las consideraciones hasta el momento son para una descomposición jerárquica de dos niveles. De todos modos, cualquier función g_{2j} puede, a su vez, tener cualquier tipo de estructura jerárquica. De esta forma, se generarían diversos niveles de jerarquía si la función lo permite.

Establecidas las definiciones anteriores, Zeng y Keane demuestran el siguiente teorema de aproximación para las funciones continuas que presentan estructura jerárquica natural:

Teorema 3.4 *Sea $G(x_1,\ldots,x_n)$ una función continua en $U = \prod_{i=1}^{n} [\alpha_i, \beta_i]$ $(U \subset \mathbb{R}^n)$ con estructura jerárquica natural* (3.16)*. Entonces, para cualquier $\epsilon > 0$, existe un sistema difuso jerárquico $F(x_1,\ldots,x_n)$ de tipo Mamdani con la misma estructura jerárquica que $G(x_1,\ldots,x_n)$:*

$$F(x_1,\ldots,x_n) = f_1[f_{21}(X_{21}),\ldots,f_{2q}(X_{2q}),X_1] \tag{3.20}$$

Tal que:

$$\|G - F\|_\infty < \epsilon \tag{3.21}$$

$$\|g_1 - f_1\|_\infty < \epsilon \tag{3.22}$$

$$\|g_{2j} - f_{2j}\|_\infty < \epsilon \quad (j = 1,2,\ldots,q) \tag{3.23}$$

Es decir, para cualquier función continua con estructura jerárquica natural, existe un sistema difuso jerárquico que se puede aproximar a ella con cualquier nivel de precisión, aproximándola bloque a bloque. Luego no sólo aproxima su comportamiento, sino que, además, su aproximación es a nivel estructural.

El teorema anterior se limita sólo al caso de funciones continuas con estructura jerárquica natural. De ahí que Zeng y Keane continúen con su desarrollo para conseguir un teorema más general.

Para dar el siguiente paso, hacen uso del teorema de Kolmogorov [Kolmogorov 57] [Kolmogorov 63] (solución del decimotercer problema de Hilbert), concretamente, de la versión simplificada que aparece en [Lorentz 66]:

Teorema 3.5 (Teorema de superposición de Kolmogorov) *Para una función continua cualquiera $G(x_1, \ldots, x_n)$ definida en $U = \prod_{i=1}^{n} [\alpha_i, \beta_i]$ $(U \subset \mathbb{R}^n)$, existen funciones continuas g y ψ_k $(k = 0, 1, \ldots, 2n)$ tal que:*

$$G(x_1, \ldots, x_n) = \sum_{k=0}^{2n} G_k(x_1, \ldots, x_n) = \sum_{k=0}^{2n} g\left[\sum_{i=1}^{n} \lambda_i \psi_k(x_i)\right] \tag{3.24}$$

siendo λ_i $(i = 1, \ldots, n)$ constantes.

En [Zeng 05b], Zeng y Keane llegan a demostrar que una función continua del tipo $G(x_1, \ldots, x_n) = g[\psi_1(x_1) + \cdots + \psi_n(x_n)]$ tiene estructura jerárquica arbitrariamente separable. Por lo tanto, a partir del teorema de Kolmogorov, se puede deducir que cualquier función continua puede representarse como la suma de $2n + 1$ funciones continuas con jerarquía arbitrariamente separable. Lamentablemente, aunque elegante, los autores se dan cuenta que la estructura obtenida por el teorema de Kolmogorov no suele ser ni la más simple, ni la mejor. Aún así, es el eslabón que les sirve para formular el siguiente teorema:

Teorema 3.6 *Para cualquier función continua $G(X)$ definida en $U = \prod_{i=1}^{n} [\alpha_i, \beta_i]$ $(U \subset \mathbb{R}^n)$, existen $Q + 1$ funciones continuas, tal que:*

$$G(X) = G_0[G_1(X), \ldots, G_Q(X)] \tag{3.25}$$

donde las funciones G_i $(i = 1, \ldots, Q)$ tienen estructura jerárquica natural.

Y, en particular, existen $2n + 1$ funciones continuas con estructura jerárquica arbitrariamente separable, tal que:

$$G(X) = \sum_{k=0}^{2n} G_k(X) \tag{3.26}$$

donde G_k $(k = 0, 1, \ldots, 2n)$ pueden ser de la misma forma que se presentan en (3.24).

Si se tiene $G(X)$ mediante la representación obtenida por el teorema de Kolmogorov, al estar compuesta por funciones continuas con estructura arbitrariamente separable,

mediante el teorema 3.4, pueden conseguirse sistemas difusos jerárquicos que aproximen cada una de las $2n+1$ funciones.

Si, por el contrario, la descomposición obtenida de G(X) es del tipo de (3.25), las funciones G_i $(i = 1, \ldots, Q)$, al tener estructura jerárquica natural, se les puede aplicar del mismo modo el teorema 3.4. $G_0(X)$, al ser una función continua, como los sistemas difusos son aproximadores universales, puede ser aproximada por un sistema difuso estándar. De este modo, demuestran que los sistemas difusos jerárquicos son aproximadores universales.

Falta por ver si el número de reglas de los sistemas difusos jerárquicos disminuye respecto al de los no jerárquicos. Para ello, supóngase un sistema difuso jerárquico con la estructura de la Figura 3.4. Supóngase, además, que $m_i^{(1)}$ es el número de etiquetas de la entrada $x_i^{(1)}$ $(i = 1, \ldots, n_1)$, $m_k^{(2j)}$ el número de etiquetas de la entrada $x_k^{(2j)}$ $(k = 1, \ldots, n_{2j};\ j = 1, \ldots, q)$ y $m_l^{(y)}$ el número de etiquetas de la variable intermedia y_l $(l = 1, \ldots, q)$. En este caso, se puede ver que el número de reglas total es $\left(\prod_{i=1}^{n_1} m_i^{(1)}\right) \cdot \left(\prod_{l=1}^{q} m_l^{(y)}\right) + \sum_{j=1}^{q} \left(\prod_{k=1}^{n_{2j}} m_k^{(2j)}\right)$. Si consideramos $M = \text{máx}\left\{\text{máx}_{i=1}^{n_1}\left(m_i^{(1)}\right), \text{máx}_{l=1}^{q}\left(m_l^{(y)}\right), \text{máx}_{j=1}^{q}\left[\text{máx}_{k=1}^{n_{2j}}\left(m_k^{(2j)}\right)\right]\right\}$, entonces puede acotarse superiormente el número total de reglas en el valor $M^{n_1+q} + \sum_{j=1}^{q} M^{n_{2j}}$, del mismo modo que puede acotarse el número de reglas del sistema difuso no jerárquico correspondiente en M^n. Como $n_1 + q < n$ y $n_{2j} < n$ $(\forall j = 1, \ldots, q)$:

$$\lim_{M \to \infty} \frac{M^{n_1+q} + \sum_{j=1}^{q} M^{n_{2j}}}{M^n} = 0 \tag{3.27}$$

Por tanto, el número de reglas del sistema difuso estándar es mayor que el número de reglas del sistema difuso jerárquico. De esta forma demuestran Zeng y Keane que los sistemas difusos jerárquicos son aproximadores universales, aproximando la función dada a nivel estructural, consiguiendo un número de reglas menor que el sistema difuso no jerárquico correspondiente.

Más adelante, Zeng y coautores continuarían su trabajo dentro del marco de los espacios discretos [Zeng 05a] [Zeng 08]. En estos trabajos amplían la teoría al campo de los sistemas neuro-difusos, consiguiendo demostrar que cualquier función definida en un espacio discreto, puede ser aproximada por un sistema difuso jerárquico con la estructura jerárquica deseada.

Aunque los resultados obtenidos por Zeng y Keane son de gran interés, tienen el problema de suponer que la descomposición (3.25) es conocida, cuando muchas veces no se conoce. De hecho, para el caso particular de la descomposición mediante el teorema de Kolmogorov, las funciones ψ_k de (3.24) no tienen una expresión explícita. Por tanto, sigue estando latente el problema de encontrar la estructura adecuada que simplifique un sistema en cuestión.

3.2. Referencias bibliográficas del autor

- Baturone I., Martínez-Rodríguez M.C., Brox P., Gersnoviez A., *Digital implementation of hierarchical piecewise-affine controllers*, in Proceedings of the 2011 IEEE International Symposium on Industrial Electronics (ISIE'11), pp. 1497-1502, 2011.
- Baturone I., Sánchez-Solano S., Gersnoviez A., Brox M., *An automated design flow from linguistic models to piecewise polynomial digital circuits*, in Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS'10), pp. 3317-3320, 2010.

Capítulo 4

Diseño e implementación de sistemas neuro-difusos aplicados a robótica móvil autónoma

Para comprobar la versatilidad y potencia de las técnicas de simplificación descritas en los capítulos anteriores, el presente capítulo se centra en otro de los ámbitos de aplicación destacados para los sistemas neuro-difusos, como es el de la robótica.

Los sistemas difusos han sido utilizados en diversas aplicaciones de control gracias a su capacidad para expresar lingüísticamente el conocimiento heurístico aportado por un experto. La capacidad de los sistemas difusos para manejar información lingüística facilita no sólo el desarrollo de los controladores, sino que, además, también facilita su mantenimiento y depuración de errores. En el caso de la robótica, los sistemas difusos ofrecen la ventaja de dotar al robot con la capacidad de procesar información simbólica, que es típica de los sistemas cognitivos. De hecho, una interfaz de lenguaje natural podría facilitar la interacción entre robots y usuarios. Por un lado, los usuarios podrían decidir qué debería hacer el robot, mientras que el robot sería capaz de explicar qué está haciendo y por qué.

Un punto débil en el proceso de diseño típico de controladores difusos es que los parámetros de los controladores se ajustan mediante métodos de *"ensayo y error"* que, aparte de ser una tarea tediosa, no consiguen comportamientos óptimos. Por ello, los métodos de *"ensayo y error"* han sido sustituidos en trabajos recientes por técnicas de aprendizaje automático, que van desde el uso de redes neuronales hasta métodos

evolutivos. El aprendizaje automático es puramente numérico y, por tanto, normalmente acaba transformando el sistema difuso y cognitivo inicial en un sistema emergente, la otra clase de sistemas cognitivos [Vernon 07]. El comportamiento del controlador mejora tras el aprendizaje, pero la información simbólica desaparece y, por tanto, también lo hace la interacción natural y directa para los humanos.

Por otro lado, el desarrollo de sistemas complejos para aplicaciones de control industrial demanda alta velocidad de respuesta, bajo consumo de potencia y área, además de bajo coste. Concretamente, aplicaciones típicas en el marco de la robótica actual, abordan sistemas multi-robot que sean capaces de ejecutar misiones cooperativas, sobre escenarios heterogéneos, de forma segura y fiable. Eso significa que los robots autónomos no sólo deben navegar de forma segura en tiempo real (bajo tiempo de procesamiento), sino que, además, deben comunicarse con otros robots y colaborar juntos en la resolución de tareas durante posibles largos períodos de tiempo (bajo consumo de potencia), llevados a cabo usando plataformas hardware que posiblemente tengan recursos limitados (como por ejemplo los microrobots o los robots aéreos no tripulados y con baja capacidad de carga).

Por tanto, el objetivo a alcanzar en este capítulo es triple: conseguir controlar un robot de forma que su comportamiento sea próximo al proporcionado por un controlador basado en análisis geométrico; que posea interpretabilidad lingüística; y que, a su vez, sea lo suficientemente simple como para ser implementado eficientemente en hardware dedicado o software empotrado en plataformas hardware de bajo coste.

La propuesta de diseño del controlador desarrollada en este capítulo consistirá en partir de una solución clásica: Reglas difusas de tipo *"si-entonces"* con interpretabilidad lingüística, que son definidas mediante conocimiento heurístico. La novedad que plantea esta técnica de diseño es que los símbolos son obtenidos exclusivamente mediante aprendizaje automático que mantiene la interpretabilidad lingüística. La información numérica necesaria para el aprendizaje se obtiene a partir de un análisis geométrico aproximado que tiene en cuenta las características cinemáticas y dinámicas del robot. El resultado de aplicar este análisis geométrico da lugar a un conjunto de ecuaciones que contienen, entre otras operaciones, divisiones, funciones trigonométricas directas (senos y cosenos) e inversas (arco cosenos) y raíces cuadradas. Para poder implementar este tipo de ecuaciones en hardware dedicado o software empotrado, una técnica habitual es aproximarlas mediante una tabla de búsqueda o una interpolación. En la técnica propuesta en este capítulo, en vez de aproximar estas operaciones, se aproxima la ecuación completa mediante un interpolador que, además, tenga una interpretación lingüística. Para conseguirlo, se aplicarán de forma combinada las técnicas de simplificación explicadas en los capítulos anteriores.

Todo esto se ilustrará mediante el desarrollo de un controlador para un robot móvil de tipo-coche, encargado de conseguir que éste navegue de forma segura, evitando posibles

obstáculos por entornos no estructurados ni conocidos, desde una configuración inicial a otra objetivo.

4.1. Aplicaciones en robótica móvil autónoma

Uno de los objetivos más importantes en robótica es crear robots autónomos que sean capaces de realizar tareas sin necesidad de una intervención humana. Un robot autónomo ha de realizar tres acciones básicas: explorar el entorno mediante sensores, procesar la información recopilada por éstos, y decidir sus acciones futuras controlando adecuadamente sus actuadores. Dentro de los problemas que se abordan en robótica móvil, uno fundamental es el de la navegación, que consiste básicamente en planificar y seguir una trayectoria entre una configuración inicial y otra objetivo. Una extensión de este problema es considerar restricciones no holónomas, es decir, ecuaciones no integrables que contienen las derivadas de los parámetros de configuración, lo que conlleva que trayectorias admisibles en el espacio de configuración puedan ser trayectorias no factibles para el robot.

En el marco de la navegación de robots no holónomos, los robots de tipo-coche han captado gran interés. El controlador de navegación de este tipo de robots debe considerar las restricciones no holónomas consistentes en que la dirección del movimiento sea siempre tangente a la trayectoria. Más aún, el radio de curvatura está mecánicamente limitado a un valor mínimo, que es equivalente a decir que la curvatura del vehículo está acotada superiormente. Entre los problemas más abordados en el control de movimiento de robots autónomos de tipo-coche destacan los de navegación con y sin obstáculos.

4.1.1. Navegación sin obstáculos

En ausencia de obstáculos, un problema típico es el de aparcamiento [Paromtichk 98] [Gómez 01] [Cuesta 04]. Éste es equivalente al problema de encontrar el camino más corto que une dos configuraciones dadas, una inicial y otra final. Para el caso de un robot tipo-coche, el camino más corto consiste en una secuencia finita de dos componentes elementales: arcos de circunferencia (con radio de curvatura mínimo) y segmentos de línea recta. Esto fue probado por Dubins para vehículos que se desplazaban únicamente hacia adelante [Dubins 97], y por Reeds y Shepp para vehículos que se mueven tanto hacia adelante como hacia atrás [Reeds 90]. En cualquier caso, el problema reside en que la curvatura es discontinua entre dos componentes elementales, con lo que estas trayectorias cortas no pueden ser seguidas de forma precisa sin detenerse en cada punto de discontinuidad para reorientar las ruedas delanteras. Para evitar estas paradas, diversos autores han propuesto planificadores de trayectorias de curvatura continua mediante el

uso de métodos de geometría diferencial. Dichos planificadores generan splines cúbicos, B-splines, β-splines, polinomios de orden 5, etc.

Otros autores han propuesto planificadores basados en lógica difusa, inspirados en emular el conocimiento heurístico de conductores expertos mediante reglas difusas y/o aprendiendo dichas reglas con la ayuda de datos de entrenamiento tomados del comportamiento humano del conductor [Sugeno 85] [Kong 92] [Jou 93]. Las ventajas de estos modelos difusos son su diseño simple, rápido, barato y de fácil mantenimiento, ya que las reglas pueden ser interpretadas de forma lingüística por un experto humano. Sin embargo, estos planificadores difusos no consideran restricciones no holónomas, y no son óptimos en el sentido de que no tienen en cuenta ni los caminos de longitud mínima ni los requisitos de control con bajo consumo de energía.

Los autores de [Chen 97] presentaron un controlador difuso para aparcar un camión con trayectorias de distancias sub-óptimas consistentes en arcos de circunferencia de radio mínimo conectados mediante curvas parabólicas que deben ser dadas por el controlador; Los autores de [Li 03a] plantean el problema de aparcamiento para un robot móvil autónomo de tipo-coche buscando trayectorias factibles de referencia mediante control en modo deslizante difuso; Y, más recientemente, los autores de [Demirli 09] resuelven el problema del aparcamiento utilizando un Mazda 6, usando un controlador neuro-difuso que aproxima las distancias compuestas por arcos de circunferencia de radio mínimo y líneas rectas mediante polinomios de orden 5, utilizando para el aprendizaje el método de *"substractive clustering"*.

Por otro lado, los autores de [Baturone 04b] describieron módulos difusos que proporcionan caminos de curvatura continua cercanos a los de longitud mínima, que se aproximan a las trayectorias formadas por arcos de circunferencia de radio de curvatura mínima y segmentos de línea recta, teniendo en cuenta restricciones no holónomas. Dichos módulos no necesitan saber los arcos o segmentos deseados (como en [Chen 97]), sino que los encuentran a partir de la configuración inicial del robot. Tampoco buscan trayectorias calculadas de forma analítica (como en [Li 03a]), sino que consiguen que el robot siga trayectorias factibles dadas en forma difusa, reduciendo de esta forma el coste computacional. Más aún, estos módulos están compuestos por un bajo número de reglas que, además, son lingüísticamente interpretables. La estrategia para conseguir dichos resultados fue utilizar algoritmos de identificación y aprendizaje supervisado, extrayendo las reglas a partir de archivos de datos de entrenamiento, conseguidos no de conocimiento heurístico ni tampoco de conductores expertos, sino de consideraciones geométricas.

4.1.2. Navegación con obstáculos

El problema básico de navegación puede complicarse con la existencia de obstáculos, debido a que el sistema de control debe considerar dos objetivos: evitar los obstáculos y alcanzar la configuración objetivo.

Para resolver el problema de la navegación con obstáculos para un robot autónomo, se han reportado diversos tipos de estrategias en la literatura. Estas soluciones se engloban en tres grandes grupos: estrategias de navegación deliberativa, estrategias de navegación reactiva y estrategias de navegación híbrida [Fernández 01].

4.1.2.1. Estrategias deliberativas

Las estrategias deliberativas fueron las primeras en utilizarse; están basadas en la ejecución cíclica de los procesos de percepción, planificación y ejecución [Nilsson 80], sin que haya conexión directa entre la percepción y la ejecución. El proceso de percepción interpreta la información proporcionada por los sensores obteniéndose un modelo del entorno. La ejecución permite navegar en un entorno real, generando consignas para los actuadores. Por último, la planificación coordina los procesos de percepción y actuación, de modo que, con el modelo generado por el proceso de percepción, produce un plan de actuación que es efectuado por el proceso de ejecución.

Una desventaja de estos métodos es que presentan problemas en la adaptación a entornos dinámicos parcialmente conocidos. Esto es debido a que es muy difícil contar con modelos completos del entorno donde la tarea va a ser llevada a cabo y que permitan asegurar la ejecución del plan generado. Por lo tanto, si aparece un objeto no recogido en el modelo es necesario reconstruir éste y generar un nuevo plan, pudiéndose producir una colisión si no da tiempo a actualizar el modelo y generar el nuevo plan. Para conseguir minimizar el tiempo de estos procesos suelen realizarse normalmente actualizaciones parciales del modelo y modificaciones del plan generado inicialmente (planificación local).

El proceso de ejecución, que realiza el seguimiento de las trayectorias, puede aplicar, entre otros, métodos como el de la persecución pura [Amidi 90] o los basados en control predictivo [Ollero 91]. Diversos autores han abordado como estrategia para la planificación de trayectorias la planificación en espacios espacio-temporales, obteniendo directamente los caminos a seguir y sus perfiles de velocidad asociados.

4.1.2.2. Estrategias reactivas

El segundo tipo de estrategias reportadas en la literatura son las reactivas. éstas surgen debido al problema presentado por las deliberativas de no reaccionar rápidamente ante variaciones del entorno. En los sistemas reactivos sí existe comunicación directa entre los procesos de percepción y ejecución.

El componente básico de este tipo de sistemas es un proceso orientado a la realización de una acción sencilla llamada comportamiento. El uso de estos elementos permite que el robot reaccione con gran flexibilidad en su entorno, comportándose de forma eficiente en entornos dinámicos donde no se posee un conocimiento a priori y preciso del mismo. Los esquemas de navegación reactiva se basan en que en cada intervalo de navegación el sistema sensorial adquiere información sobre el estado del entorno; en base a esta información extraída, se activan uno o varios comportamientos simples sumando sus actuaciones, de modo que, el comportamiento final del sistema, es una combinación de las acciones simples que han sido activadas.

Uno de los esquemas de navegación reactiva con mayor trascendencia ha sido el de inhibición de Brooks [Brooks 86] que se fundamenta en un sistema basado en flujo de datos donde las informaciones recogidas por los sensores son procesadas por los comportamientos que generan salidas hacia los actuadores; los comportamientos se agrupan en niveles de prioridad, donde la activación de un nivel inferior o de mayor prioridad, inhibe las consignas generadas por un nivel superior o de menor prioridad.

Una de las aplicaciones típicas en la navegación reactiva ha sido la de evitación de obstáculos. Entre las soluciones reportadas en la literatura destacan las que se basan en el seguimiento del contorno del obstáculo, en campos potenciales, en el empleo de lógica difusa y en optimización local. Todas ellas tienen en común no usar información previa del entorno ni realizar la planificación de un camino como se realizaba en las estrategias deliberativas; utilizan para generar la reacción la información sensorial que se recibe.

Aunque una solución muy simple para evitar un obstáculo es seguir su contorno, esta solución presenta problemas en el seguimiento de obstáculos con concavidades debido a que el robot puede quedar atrapado sin posibilidad de salir a causa de las restricciones cinemáticas del vehículo.

Otra estrategia ampliamente usada es la navegación basada en campos potenciales. En ella se considera al robot como una partícula dentro de un campo de fuerzas que desea llegar a una configuración destino. Se distinguen dos tipos de fuerzas: atractivas y repulsivas. Mientras que la configuración destino ejerce una fuerza atractiva sobre el robot, las configuraciones ocupadas por algún obstáculo ejercen una fuerza repulsiva que evitan que el robot colisione. El robot situado en una configuración determinada se dirigirá en la dirección de la fuerza resultante de la composición de las fuerzas repulsivas y atractivas. El problema de este método es que pueden aparecer mínimos locales en el potencial resultante, lo que puede provocar que el robot no llegue a la configuración objetivo. Connoly [Connoly 90] propuso resolver este problema mediante el uso de funciones potenciales armónicas, aunque éstas tienen un coste computacional muy elevado. Por otro lado, Borestein y Koren [Borestein 89] propusieron el método basado en histograma de vector campo. Hay otro tipo de estrategias basadas en la idea de los campos potenciales, que realizan el cálculo de las consignas de control con sistemas di-

fusos. Fabrizi [Fabrizi 99] evalúa el coste de la elección de una determinada dirección y velocidad en base a tres conjuntos difusos. Hay otros autores, como Vázquez y García [Vázquez 94] que también generan las consignas de reacción fusionando la información sensorial y el objetivo a alcanzar mediante lógica difusa.

La lógica difusa también ha sido utilizada directamente para la generación de reacciones de evitación de obstáculos pues es muy intuitivo plasmar el conocimiento humano en el problema de evitación de obstáculos. Entre los autores que han usado esta estrategia se encuentran Kubota y Hashimoto [Kubota 92], los cuales generan en cada instante las consignas necesarias para esquivar obstáculos fijos en función de reglas difusas cuyas entradas son la dirección y distancia a las que están situados los obstáculos. Para el caso de obstáculos móviles también son tenidas en cuenta la velocidad y dirección relativas del obstáculo al robot. Otros ejemplos que emplean reglas difusas extraídas de conocimiento heurístico pueden verse en [Kimiaghalam 01] [Zhang 05]. Normalmente son controladores simples, pero suelen ir acompañados de un tedioso ajuste del tipo prueba-error que no acaba obteniendo resultados óptimos. Por otro lado, se han reportado varios métodos basados en paradigmas difusos y de redes neuronales [Fukuda 99] [Driankov 01] [Meng 05], por ejemplo, para aprender las trayectorias proporcionadas por un experto [Rusu 03] [Wang 03]. Incluso en este caso, las trayectorias que lleva a cabo el robot no son, por lo general, tan óptimas como las obtenidas a partir del análisis geométrico del problema, y, además, los controladores resultantes no suelen ser simples y pierden cualquier significado lingüístico.

4.1.2.3. Estrategias de navegación híbridas

Las estrategias deliberativas presentan como problema la rigidez y como ventaja la capacidad de navegar de forma óptima. Las reactivas, por el contrario, se caracterizan por su flexibilidad y falta de capacidad de navegar de forma óptima. De ahí que se hayan usado esquemas mixtos que combinen ambas estrategias, para así conseguir las ventajas que ambas ofrecen y reducir los inconvenientes que ambas presentan. Este tipo de estrategias se conocen con el nombre de estrategias de navegación híbridas [Wooldridge 09].

Dentro de los esquemas de navegación híbridos, destacan aquellos que usan en los niveles inferiores mecanismos de control reactivo, mientras que en las capas superiores se sitúan mecanismos de planificación y deliberación. La ventaja de estas estrategias reside en la combinación de la planificación global de trayectorias, donde se tiene en cuenta toda la información conocida antes del tiempo de navegación, con la reacción tomada en base al conocimiento adquirido durante el seguimiento de la trayectoria inicial.

Kant y Zucker [Kant 86] desglosan el problema de la planificación de una trayectoria libre de obstáculos en dos subproblemas. El primero representa una primera planificación global, con todo el conocimiento que se tiene del entorno antes de comenzar la navega-

ción; el segundo representa la modificación de la trayectoria en tiempo de navegación cuando se detecta un obstáculo en la zona cercana al robot. El desvío de la trayectoria inicialmente generada se realiza utilizando campos potenciales, sin tener en cuenta la dinámica del robot ni considerar el movimiento del obstáculo.

Kyriakopoulos y Saridis [Kyriakopoulos 93] usan campos potenciales para evitar obstáculos considerando en detalle el modelo dinámico del robot para imponer los límites de velocidad. Debido a que se centra en la variación del perfil de velocidad del vehículo y no en modificar el camino que sigue, sólo se puede utilizar para la evitación de obstáculos conocidos a priori y obstáculos móviles inesperados. Para evitar obstáculos estáticos inesperados es necesario combinar esta estrategia con otras.

Griswold y Eem [Griswold 90] consideran que la evitación de obstáculos se puede realizar en dos fases: una primera en la que se genera un camino que evita todos los obstáculos estáticos y una segunda en tiempo de ejecución, donde se generan los cambios de velocidad necesarios para la evitación de obstáculos móviles detectados por el sistema sensorial. No consideran las limitaciones de velocidad ni características geométricas del obstáculo como puede ser el tamaño.

Las estrategias analizadas en este capítulo para el diseño de controladores neuro-difusos han sido las estrategias reactivas, como se verá en el apartado **??**.

4.2. Realizaciones hardware de sistemas difusos para robots móviles autónomos

La idea de un robot móvil autónomo es que lleve a bordo todo el hardware necesario para llevar a cabo sus tareas. Dentro de ellas, un grupo básico son las tareas de bajo nivel entre las que se incluyen las de acondicionamiento de las señales de los sensores y actuadores, y las de control de bajo nivel de los actuadores. El hardware estándar que se emplea para realizar estas tareas de bajo nivel se basa en microcontroladores comerciales, que poseen periféricos para recibir señales analógicas (como las que proporcionan muchos sensores tipo sónar) y digitales (como las de muchos codificadores rotatorios, giróscopos, GPSs y láseres) y periféricos para proporcionar señales analógicas y digitales (como las que se emplean en el control de motores). En este ámbito de tareas de bajo nivel se han propuesto muchos algoritmos basados en lógica difusa que mejoran las técnicas convencionales. Ejemplos de este tipo son los controladores difusos que extienden las posibilidades de los PID (proporcional-integral-derivativo) o de los controladores basados en la teoría *"sliding mode"* (modo deslizante). Estos controladores difusos de bajo nivel también se implementan normalmente sobre microcontroladores comerciales [Betin 07].

Otro grupo de tareas, que se pueden denominar de nivel intermedio, son las que incluyen la realización de comportamientos, como por ejemplo navegar hacia una con-

figuración objetivo, evitar obstáculos o perseguir a un móvil determinado. Tareas de nivel superior son las que combinan las tareas anteriores para cumplir misiones como, por ejemplo, el rescate de heridos o apagar incendios. También en este nivel superior se incluyen tareas de comunicación con otros robots (para realizar misiones cooperativas) o con seres humanos (por ejemplo, hospitales o centros de emergencia). Para este tipo de tareas intermedias y (sobre todo) para las superiores, el hardware que se ha empleado tradicionalmente han sido PCs industriales tipo PC104. Los PC104 son una versión de la arquitectura PC para aplicaciones empotradas e industriales donde el espacio, el consumo de energía y/o la fiabilidad son factores críticos. Por eso las tarjetas PC104 (90 x 96mm) son más pequeñas que las tarjetas ISA, ya que pueden apilarse unas sobre otras sin necesidad de placas base, consumen menos y están diseñadas para ser más robustas que los sistemas PC. Una gran ventaja es que pueden ser programados con las mismas herramientas que los PCs, lo que reduce la necesidad de conocimientos y el costo de un desarrollo personalizado.

La posibilidad de emplear una metodología que reduzca el tiempo y precio en el desarrollo del sistema de control es particularmente interesante en el campo de la robótica móvil autónoma, en la que los algoritmos de control deben ser validados y refinados en pruebas de campo con el robot real, porque es muy difícil que las simulaciones contemplen todos los posibles efectos de no linealidades, ruidos, interferencias, etc. Por eso otras opciones que también se están empleando mucho en este campo son tarjetas con un hardware más específico para tareas de control, pero que permiten un prototipado tan rápido y cómodo como es trabajar con un PC. En esta línea se encuentran, por ejemplo, tarjetas como la DS1104 comercializada por dSPACE, que incluyen un microprocesador PowerPC junto con un DSP esclavo de Texas Instruments, además de memoria, varios temporizadores, unidad de control de interrupciones, hardware para entradas y salidas tanto analógicas como digitales, interfaz para encoders, interfaz serie para comunicaciones e interfaz PCI para conectarse a un PC [dSpace]. Esta tarjeta se puede programar cómodamente desde el entorno Matlab/Simulink a través de las rutinas del Real-Time Workshop (RTW). La utilización de este tipo de tarjetas para implementar estrategias de navegación difusas basadas en comportamientos puede verse en [Cupertino 06].

Si el diseño de los algoritmos difusos se hace de forma adecuada, no es necesario el empleo de microprocesadores potentes para llevar a cabo tareas complejas en tiempo real. Así, por ejemplo, en [Wang 04] se describe cómo un algoritmo de navegación evitando obstáculos se puede simplificar empleando 32 en vez de 625 reglas para implementarse con éxito en una placa de control basada en el microcontrolador 86HC11 de Motorola. Otro trabajo en esta línea, que se describe en [Baturone 04a], es la implementación de un algoritmo difuso para navegar sin obstáculos en la tarjeta de control LF2407 EVM de Spectrum Digital [TMS320LF2407a]. Esta tarjeta sólo posee como procesador el TMS320LF2407, que es un DSP de punto fijo de Texas Instruments que combina la

capacidad de procesado de un núcleo de DSP con los periféricos de un microcontrolador. La CPU de este DSP contiene: una ALU de 32 bits, un acumulador de 32 bits, desplazadores para escalar datos, un multiplicador de 16 x 16 bits y lógica para generación de direcciones de datos y programa [TMS320LF2407b]. El controlador difuso se programó y descargó en el DSP utilizando el entorno de desarrollo Code Componer Studio de Texas Instruments. Los resultados experimentales obtenidos son tan buenos como los obtenidos con un PC en el que los algoritmos difusos fueron diseñados sin ningún tipo de restricciones [Baturone 04b].

Otro tipo de hardware que está siendo objeto de amplia investigación en el mundo de la robótica móvil autónoma es el basado en FPGAs. Típicamente las FPGAs se emplean como coprocesadores combinados con microcontroladores, DSPs o microprocesadores. En [Arroyabe 00] [Scolari 03] y [Sánchez 13] pueden verse ejemplos de este tipo de soluciones. En [Sánchez 13] el coprocesador difuso que se combina con un DSP consta de un *"soft core"* que contiene las tareas de control difuso de alto nivel y de un software empotrado sobre el MicroBlaze que implementa las tareas de comunicación con el DSP, por lo que se habla de una implementación hardware/software. Otros trabajos incluso incluyen las tareas de control de bajo nivel en la FPGA. En esta línea se puede citar el trabajo de [Islam 06] en el que un algoritmo de navegación evitando obstáculos que se desarrolla en Matlab es después traducido a VHDL y sintetizado como *"soft core"* en una FPGA APEX de Altera. Otro ejemplo es el trabajo de [Li 03b] en el que se describe cómo, tanto el algoritmo de navegación difuso como las tareas de comunicación y control de bajo nivel de motores, se escribieron en VHDL y se implementaron como *"soft cores"* en una FPGA Flex de Altera. En [Paiz 06] se explota la reconfigurabilidad que ofrecen las FPGAs para implementar un *"soft core"* difuso reconfigurable. Por último, en [[Tzafestas 10], resuelven el problema de seguimiento de caminos para un robot de tipo-coche, mediante un controlador difuso implementado como *"soft core"* en una FPGA Spartan-3 de Xilinx.

A la hora de diseñar sistemas de control neuro-difusos, el estudio desarrollado en este capítulo se ha centrado en plataformas basadas en DSPs y, sobre todo, FPGAs de bajo coste.

4.3. ROMEO 4R

ROMEO 4R es un robot desarrollado en el Departamento de Ingeniería de Sistemas y Automática de la Escuela Superior de Ingenieros de la Universidad de Sevilla (Figura 4.1). Toma su nombre de *RObot Móvil para Exteriores*, y es un prototipo resultado de la adaptación de un vehículo eléctrico convencional de cuatro ruedas a una plataforma de investigación y desarrollo, mediante la incorporación de diversos dispositivos y sensores. Desde el punto de vista de la estructura física, se trata de un vehículo muy similar a

Figura 4.1: ROMEO 4R

los coches de los campos de golf. El objetivo de esta plataforma es la experimentación de técnicas tanto de navegación autónoma como de control teleoperado. En cualquier caso, el vehículo ha sido diseñado para permitir una conducción tanto manual como automática. Se puede elegir entre uno u otro modo de funcionamiento mediante un cuadro de mandos fácilmente accesible tanto para el conductor como para el acompañante, pues se encuentra situado en la parte central justo a la derecha del volante.

A continuación se resumirán las diferentes partes que componen el vehículo completo, para tener una visión general del mismo y comprender cómo el controlador, que se desarrollará a lo largo del capítulo, interactuará con él.

4.3.1. Actuadores

El vehículo tiene instalados dos motores de corriente continua, uno aplicado a las ruedas motrices y otro a las ruedas directrices. El motor de tracción es de 36V y desarrolla 2CV de potencia. El motor de dirección es de 24V y 80W de potencia.

4.3.2. Codificadores (encoders)

ROMEO 4R dispone de varios codificadores ópticos, pero sólo se usan dos, ambos relativos o incrementales. Uno de ellos es un codificador situado en el eje de tracción, que permite determinar el desplazamiento del vehículo. A partir del mismo, se pueden obtener estimaciones de la velocidad para un intervalo de tiempo Δt tomando dos medidas consecutivas, una al principio del intervalo (x_i) y otra al final (x_f), y realizando las conversiones de unidades necesarias:

$$v = \frac{2\pi R}{K} \frac{x_f - x_i}{\Delta t} \tag{4.1}$$

K es el número de pulsos que da el codificador en una vuelta completa de rueda y R el radio de las ruedas.

El otro codificador es lineal y se encuentra situado en la parte delantera del vehículo, en un lugar protegido de posibles daños y suciedad. Se usa para medir el giro de las ruedas directrices. Para ello, mide el desplazamiento de la vara.

La lectura de los codificadores se realiza mediante la DCX PC-100, una tarjeta de control de motores que dispone de varias entradas para la lectura de encoders.

4.3.3. Sensores de navegación: Giróscopo y GPS

Con el objeto de determinar de forma precisa la orientación del vehículo, ROMEO incorpora un giróscopo del modelo *Autogyro Navigator Plus de KVH Industries*. Se trata de un interferómetro de fibra óptica de un solo eje, adecuado para sistemas de navegación terrestre. Proporciona medidas de la velocidad angular de giro con bastante precisión, con lo que se puede obtener una buena estimación de la orientación del vehículo a partir del mismo.

Empleando sólo información procedente de los codificadores y del giróscopo es habitual la acumulación de errores en la estimación de la posición. Por eso ha sido necesario dotar al vehículo de un sistema de posicionamiento global por satélite (GPS), pudiendo alcanzar de esta forma errores inferiores a unos pocos centímetros.

4.3.4. Sónares y láser 2D

El robot posee un total de diez sónares de la marca SIEMENS. De estos diez, cuatro son del modelo 3RG6125-3BF00 (corto alcance) y el resto son del modelo 3RG6124-3BF00 (largo alcance). El objetivo de estos sónares es la detección de posibles elementos cercanos al vehículo. El rango de detección de los sónares varía en función del modelo:

- *Corto alcance*: 400-3000mm.
- *Largo alcance*: 600-6000mm.

La salida que proporcionan estos sónares consiste en una corriente proporcional a la distancia que existe entre el sónar y el elemento detectado más cercano. Mediante una tarjeta de conversión de corriente a tensión, es posible realizar la lectura de los sónares a partir de una tarjeta con convertidores A/D de tensión.

En relación con el sensor láser 2D, se trata del modelo LMS220-30106 de la marca SICK (Figura 4.2), y se ubica en la parte delantera del vehículo. Consiste en un sistema de medición láser que realiza una detección de los alrededores en dos dimensiones. Entre sus características principales, debe destacarse que la distancia máxima de medida es de 80m, con un barrido de 180°. El interfaz de comunicaciones está basado en un puerto serie, admitiendo el protocolo RS232 o RS422.

Figura 4.2: Sensor láser 2D instalado en ROMEO 4R

4.3.5. Visión: Cámaras

El sistema de visión de ROMEO 4R está compuesto por un conjunto de dos cámaras instaladas en sendos *pan & tilt*, sobre el techo del vehículo. El hecho de poseer dos cámaras simultáneamente permite disponer de visión estéreo, lo que se puede aprovechar para conocer la distancia de un objeto al vehículo. Se dispone de varios modelos, entre ellas, las cámaras monocromas CV-M50 de la marca JAI alimentadas a 12V.

4.3.6. Sistema de control industrial

El sistema de control tipo industrial de ROMEO 4R dispone de dos equipos informáticos conectados entre sí mediante un cable de red de tipo par trenzado con conectores RJ-45. Se les conoce con los nombres de *romeo4a* y *romeo4b*. Ambos equipos se encuentran situados en la parte trasera del vehículo. *Romeo4a* es el equipo encargado de la recogida de información y posterior procesamiento de las imágenes capturadas por las cámaras. Este PC dispone de un conjunto de DSPs así como una tarjeta digitalizadora destinadas a la captura y procesamiento de las imágenes. Si así lo requiere la aplicación, este equipo puede emplearse para mandar información de control a *romeo4b*. *Romeo4b* se encarga del control de bajo nivel, lo que incluye fundamentalmente la lectura continuada de todos los dispositivos (a excepción de las cámaras), el procesamiento de la información recogida, la ejecución del algoritmo de control, así como llevar a cabo la actuación sobre los motores. Este controlador está constituido por un ordenador personal (PC) industrial que dispone de un procesador Pentium a 100Mhz, 64MB de memoria RAM, un disco duro de 20GB, una tarjeta gráfica VGA, una tarjeta de expansión de puertos serie (que añade dos más) y una tarjeta de red Ethernet, para la comunicación con *romeo4a* o bien con alguna red de área local. La alimentación del PC es a 24V de corriente continua. Asimismo, se encuentran insertados en *romeo4b* los siguientes dispositivos:

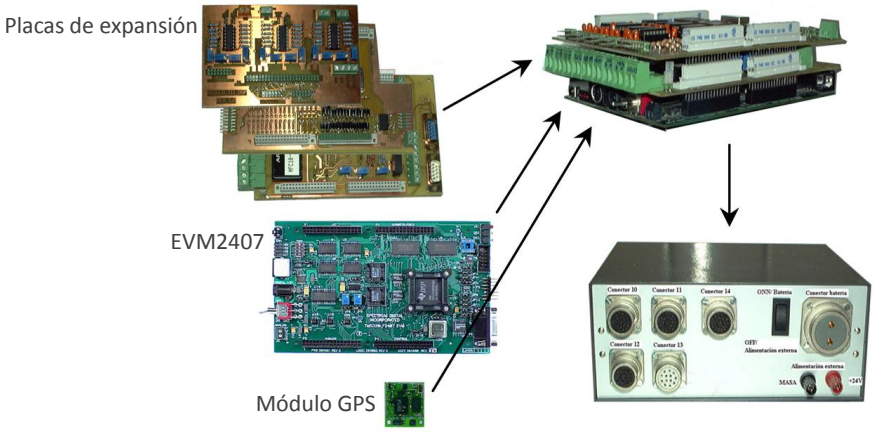

Figura 4.3: Controlador empotrado basado en DSP para ROMEO 4R

- Tarjeta de control de motores DCX-PC 100, de la marca Precision MicroControl Corporation. Actúa de interfaz entre el hardware de control y el PC, proporcionando medidas de los codificadores de los motores y permitiendo la actuación sobre los mismos. Esto es posible gracias a dos módulos de control de motores MC-200 con los que está equipada la tarjeta, cada uno responsable del control de un motor. Estos módulos realizan un control PID programable con realimentación directa de la lectura de los codificadores.

- Tarjeta de adquisición de datos de propósito general AX5411 de Axiom. Permite la lectura y escritura de entradas y salidas analógicas y digitales.

4.3.7. Sistema de control empotrado

También se ha desarrollado un controlador empotrado para ROMEO basado en el DSP de punto fijo TMS320LF2407 de Texas Instruments [Ferruz 03]. La Figura 4.3 muestra los componentes de este controlador: (a) una placa EVM2407 de Spectrum Digital [TMS320LF2407a] [Ferruz 03], que incluye el chip DSP, 128K de SRAM externa, 4 canales de convertidores D/A de 12 bits y circuitería de interfaz; (b) placas de expansión para acondicionamiento de señal, protección de sobre tensiones y 4 puertos serie adicionales; y (c) un módulo GPS compacto.

Los convertidores D/A de la placa EVM se comunican con las tarjetas de control de los motores de tracción y dirección. Uno de los puertos RS-232 de la placa de expansión se emplea para comunicación con el giróscopo. El hardware del DSP ofrece una interfaz directa para los encoders así como otro puerto RS-232, que se usa como enlace de comunicaciones con un PC externo para intercambiar, entre otros mensajes, posibles

situaciones de error. Se ha empleado el entorno Code Composer Studio para desarrollar y descargar sobre el DSP un entorno en tiempo de ejecución que incluye los drivers de bajo nivel para acceder a los sensores y actuadores del robot y una serie de módulos software encargados de leer la información de los sensores, pasar las señales de control a los actuadores implementando los PID de bajo nivel, así como llevar a cabo algoritmos para estimar la posición del robot y otras tareas de control básicas. Las tareas de control intermedio pueden realizarse como módulos software sobre el DSP [Baturone 05] o como módulos hardware o software en otras placas, por ejemplo basadas en una FPGA que, a su vez, se comuniquen con el DSP [Sánchez 07]. Para el problema de navegación tratado en este capítulo, será necesario conocer la posición del robot. Para esto se emplearán los encoders, el giróscopo y el GPS (para más precisión). También se necesitará conocer dónde hay obstáculos. Para ello se hará uso sólo del láser 2D. A diferencia del sistema de control industrial, el sistema de control empotrado sí impone restricciones de cómputo y memoria para implementar algoritmos de control. El controlador neuro-difuso, cuyo diseño se describirá a continuación, puede empotrarse sin problema como otro módulo software en el DSP de este sistema, o como un módulo hardware o software en una placa basada en FPGA.

4.4. Referencias bibliográficas del autor

- Baturone I., Gersnoviez A., Barriga A., *Neuro-fuzzy techniques to optimize an FPGA embedded controller for robot navigation*, Applied Soft Computing, vol. 21, pp. 95-106, 2014.
- Baturone I., Gersnoviez A., *A simple neuro-fuzzy controller for car-like robot navigation avoiding obstacles*, in Proceedings of the 2007 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'07), pp. 1-6, 2007.

Conclusiones finales

El objetivo primordial de esta Tesis ha sido desarrollar técnicas de simplificación para sistemas neuro-difusos, de modo que sistemas complejos puedan ser ejecutados de forma eficiente en software empotrado o hardware dedicado.

A la hora de extraer conocimiento a partir de datos numéricos, se ha optado por las técnicas de tipo grid, para así facilitar una implementación hardware eficiente. Concretamente, se han utilizado sistemas neuro-difusos de tipo Takagi-Sugeno con familias de triángulos como funciones de pertenencia para cubrir las entradas, siempre que sea posible. Estos sistemas, con un número de reglas y funciones de pertenencia reducido, no sólo son aptos para una implementación eficiente, sino que también facilitan su interpretabilidad lingüística. Sin embargo, estos sistemas presentan el fenómeno de *"la maldición de la dimensionalidad"*, por lo que se han propuesto una serie de técnicas para mitigar en lo posible dicho problema.

- Se ha desarrollado una técnica basada en una simplificación tabular inspirada en el algoritmo de Quine-McCluskey de la lógica booleana. El método no sólo demuestra ser bastante potente a la hora de reducir el número de reglas, sino que , al agrupar reglas, permite el diseño de clasificadores neuro-difusos cuyas regiones o fronteras de decisión no tienen por qué ser las típicas paralelas a los ejes de las variables de entrada sino que pueden ser no lineales, dependiendo de las t-normas, s-normas y funciones de pertenencia empleadas.

- Partiendo de la simplificación tabular, se ha propuesto un método que permite ordenar las reglas obtenidas según su grado de generalidad o particularidad, medido por un índice que ha sido denominado *índice de cubrimiento*. El método facilita el diseño de clasificadores neuro-difusos con bases de reglas incompletas, que se ha aplicado a diversos ejemplos de descripción de imágenes sencillas (patrones) para probar su eficacia. Con esto, se ha conseguido describir características de bajo nivel de imágenes, utilizando bases de reglas muy simples, empleadas en sistemas de reconocimiento de patrones, que llegaron a mostrar un mejor comportamiento que el de otros sistemas con un número mayor de reglas.

- Se ha revisado el estado del arte correspondiente a las técnicas de simplificación de sistemas difusos mediante el uso de jerarquía, analizando las propiedades demostradas por los trabajos previos, así como las desventajas que arrastraban. Los estudios demostraban que este tipo de simplificación conseguía disminuir las reglas de un sistema de forma drástica. Sin embargo, el mayor problema consistía en encontrar la forma de descomponer el sistema.

- Se ha propuesto un método de descomposición jerárquica que consigue el mayor nivel de simplificación hasta ahora reportado en la literatura.

- Los sistemas jerárquicos propuestos, como se comportan como sistemas lineales a tramos bajo ciertas condiciones, son muy eficientes a la hora de aproximar comportamientos óptimos no lineales.

- Tras estudiar las características de estos sistemas, se ha propuesto una metodología que permite evaluar si es posible o no encontrar una estructura jerárquica como la propuesta, y, en caso de ser posible, cómo encontrarla. La gran mayoría de trabajos previos sobre sistemas jerárquicos no propone ninguna metodología de diseño.

- La metodología se ha probado en el diseño de varios controladores predictivos basados en modelo. Comparando los controladores jerárquicos diseñados con otros controladores reportados en la literatura, se concluye que los controladores jerárquicos propuestos son los más sencillos y, por lo tanto, los más eficientes para ser implementados. En concreto, para implementaciones en FPGAs son los que menos slices ocupan, menos latencia y *throughput* presentan y menos recursos propios de la FPGA emplean. A pesar de su simplicidad, su eficacia en cuanto a control es elevada: son estables y su tiempo de establecimiento no se empeora significativamente, incluso se mejora en algunos casos.

- Como ejemplo de aplicación final para ilustrar la problemática compleja que surge a la hora de diseñar sistemas neuro-difusos, se ha abordado el diseño de un controlador para la navegación libre de obstáculos de un robot autónomo tipo-coche. Puesto que el primer paso a la hora de diseñar un sistema neuro-difuso es disponer de datos numéricos, el diseño partió de un análisis geométrico del problema, con consideraciones cinemáticas y dinámicas, para alcanzar el mejor comportamiento de respuesta. El resultado obtenido fue el habitual cuando se abordan este tipo de problemas: las ecuaciones obtenidas fueron demasiado complejas para ser implementadas en hardware dedicado y carecían de interpretabilidad lingüística.

- Al controlador anterior se le aplicaron las distintas técnicas desarrolladas en esta Tesis, obteniendo como resultado un controlador simple con un número de reglas

reducido con contenido lingüístico fácil de interpretar por un usuario no experto. La simplicidad del controlador hizo que la complejidad del cálculo fuera muy reducida, pudiendo ejecutarse a gran velocidad o con bajo consumo de potencia por sistemas basados en procesadores con punto fijo. Y, junto a estas ventajas, se añadió que el comportamiento alcanzado fue cercano al óptimo. Con lo cual, además de ser un controlador apto para una implementación hardware eficiente y de poseer interpretabilidad lingüística, el comportamiento ofrecido fue cercano al de un controlador descrito por ecuaciones geométricas.

- Todas las técnicas desarrolladas han demostrado una gran potencia a la hora de simplificar sistemas, consiguiendo sistemas de gran simplicidad muy aptos para una implementación hardware de muy bajo coste. A pesar de la complejidad que conllevan este tipo de técnicas, su desarrollo y pruebas se han llevado a cabo con facilidad gracias a la gran flexibilidad aportada por las herramientas de CAD del entorno de Xfuzzy 3. Con la ayuda de este entorno y la utilización de las técnicas desarrolladas, se pueden conseguir sistemas neuro-difusos muy simples de forma rápida y sencilla.

Bibliografía

[Aja 08] Aja-Fernández S., Alberola-López C., *Matrix modeling of hierarchical fuzzy systems*, IEEE Transactions on Fuzzy Systems, vol. 16, no. 3, pp. 585-599, 2008. 34

[Alcalá 15] Alcalá-Fernández J., Alonso J.M., *A survey of fuzzy systems software: Taxonomy, current research trends and prospects*, IEEE Transactions on Fuzzy Systems, dx.doi.org/10.1109/TFUZZ.2015.2426212. 22

[Alessio 09] Alessio A., Bemporad A., *A survey on explicit model predictive control*, chapter of Magni L., Raimondo D.M., Allgöwer F. (Editors), *Nonlinear model predictive control: Towards new challenging applications*, Lecture Notes in Control and Information Sciences, vol. 384, pp. 345-369, Springer Berlin Heidelberg, 2009.

[Amidi 90] Amidi O., *Integrated mobile robot control*, Carnegie Mellon University, Robotics Institute, 1990. 51

[Arroyabe 00] Arroyabe J.L., Aranguren G., Nozal L.A.L., Martin J.L., *Autonomous vehicle guidance with fuzzy algorithm*, in Proceedings of the 26^{th} Annual Conference of the IEEE Industrial Electronics Society (IECON'2000), vol. 3, pp. 1503-1508, 2000. 56

[Bannatyne 96] R. Bannatyne, *Motorola's 68HC12 an evolution from 8-bit to 16-bit*, Embedded-System Engineering, vol. 4, no. 4, pp. 32-33, 1996. 19

[Basterretxea 09] Basterretxea K., del Campo I., *Electronic hardware for fuzzy computation*, chapter of Laurent A., Lesot M.-J. (Editors), *Scalable fuzzy algorithms for data management and analysis: Methods and design*, Information Science Reference, Hershey-New York, USA, pp. 1-30, 2009. 21

[Battiti 92] Battiti R., *First- and second- order methods for learning: Between steepest descent and Newton's method*, Neural Computation, vol. 4, no. 2, pp. 141-166, 1992. 12

[Baturone 00] Baturone I., Barriga A., Sánchez-Solano S., Jiménez C.J., López D., *Microelectronic design of fuzzy logic-based systems*, CRC Press, Boca Ratón, FL, USA, 2000. 18, 21

[Baturone 01] Baturone I., Sánchez-Solano S., *Microelectronic design of universal fuzzy controllers*, Mathware and Soft Computing, vol. 8, no. 3, pp. 303-319, 2001. 13

[Baturone 04a] Baturone I., Moreno-Velo F.J., Sánchez-Solano S., Blanco V., Ferruz J., *DSP-based fuzzy controllers: application to parking an autonomous robot*, in

Proceedings of the 19^{th} Conference on Design of Circuits and Integrated Systems (DCIS'2004), 2004. 55

[Baturone 04b] Baturone I., Moreno-Velo F.J., Sánchez-Solano S., Ollero A., *Automatic design of fuzzy controllers for car-like autonomous robots*, IEEE Transactions on Fuzzy Systems, vol. 12, no. 4, pp. 447-465, 2004. 50, 56

[Baturone 05] Baturone I., Moreno-Velo F.J., Sánchez-Solano S., Blanco V., Ferruz J., *Embedded fuzzy controllers on standard DSPs*, in Proceedings of the 2005 IEEE International Symposium on Industrial Electronics (ISIE'05), vol. 3, pp. 1197-1202, 2005. 61

[Baturone 07] Baturone I., Moreno-Velo F.J., Sánchez-Solano S., Barriga A., Brox P., Gersnoviez A., Brox M., *Using Xfuzzy environment for the whole design of fuzzy systems*, in Proceedings of the 16^{th} IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'07), pp. 1-6, 2007.

[Baturone 08] Baturone I., Moreno-Velo F.J., Blanco V., Ferruz J., *Design of embedded DSP-based fuzzy controllers for autonomous mobile robots*, IEEE Transactions on Industrial Electronics, vol. 55, no. 2, pp. 928-936, 2008.

[Bayat 11] Bayat F., Johansen T.A., Jalali A.A., *Using hash tables to manage the time-storage complexity in a point location problem: Application to explicit model predictive control*, Automatica, vol. 47, no. 3, pp. 571-577, 2011.

[Bellman 61] Bellman R.E., *Adaptive control processes*, Princeton University Press, 1961. 10

[Bemporad 02a] Bemporad A., Borrelli F., Morari M., *Model predictive control based on linear programming - The explicit solution*, IEEE Transactions on Automatic Control, vol. 47, no. 12, pp. 1974-1985, 2002.

[Bemporad 02b] Bemporad A., Morari M., Dua V., Pistikopoulos E.N., *The explicit linear quadratic regulator for constrained systems*, Automatica, vol. 38, no. 1, pp. 3-20, 2002.

[Betin 07] Betin F., Sivert A., Yazidi A., Capolino G.-A., *Determination of scaling factors for fuzzy logic control using the sliding-mode approach: application to control of a DC machine drive*, IEEE Transactions on Industrial Electronics, vol. 54, no. 1, pp. 296-309, 2007. 54

[Bezdek 84] Bezdek J.C., Ehrlich R., Full W., *FCM: The fuzzy c-means clustering algorithm*, Computers & Geosciences, vol. 10, no. 2-3, pp. 191-203, 1984. 6

[Bezdek 98] Bezdek J.C., Pal N.R., *Some new indexes of cluster validity*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 28, no. 3, pp. 301-315, 1998. 27

[Bobrowski 91] Bobrowski L., Bezdek J.C., *C-means clustering with the l_1 and l_∞ norms*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, no. 3, pp. 545-554, 1991. 7

[Borestein 89] Borestein J., Koren Y., *Real time obstacle avoidance for fast mobile robots*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 19, no. 5, pp. 1179-1187, 1989. 52

[Bosque 14] Bosque G., del Campo I., Echanobe J., *Fuzzy systems, neural networks and neuro-fuzzy systems: A vision on their hardware implementation and platforms over two decades*, Engineering Applications of Artificial Intelligence, vol. 32, pp. 283-331, 2014. 21

[Breiman 93] Breiman L., *Hinging hyperplanes for regression, classification, and function approximation*, IEEE Transactions on Information Theory, vol. 39, no. 3, pp. 999-1013, 1993.

[Brooks 86] Brooks R.A., *A robust layered control system for a mobile robot*, IEEE Journal of Robotics and Automation, vol. 2, no. 1, pp. 14-23, 1986. 52

[Broomhead 88] Broomhead D.S., Lowe D., *Multivariate functional interpolation and adaptive networks*, Complex Systems, vol. 2, pp. 321-355, 1988.

[Brox 07] Brox P., Baturone I., Sánchez-Solano S., *A fuzzy edge-dependent interpolation algorithm*, chapter of Nachtegael M., van der Weken D., Kerre E.E., Philips W. (Editors), *Soft computing in image processing: Recent Advances*, Series Studies in Fuzziness and Soft Computing, Springer-Verlag, vol 210, pp. 155-185, 2007.

[Brox 13a] Brox P., Castro-Ramírez J., Martínez-Rodríguez M.C., Tena E., Jiménez C.J., Baturone I., Acosta A.J., *A programable and configurable ASIC to generate Piecewise-Affine functions defined over general partitions*, IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 60, no. 12, pp. 3182-3194, 2013.

[Brox 13b] Brox M., Sánchez-Solano S., del Toro E., Brox P., Moreno-Velo F.J., *CAD tools for hardware implementation of embedded fuzzy systems on FPGAs*, IEEE Transactions on Industrial Informatics, vol. 9, no. 3, pp. 1635-1644, 2013. 22

[Cabrera 02] Cabrera A., Sendhadji R., Sánchez-Solano S., Barriga A., Jiménez C.J., Llanes O., *Development of level controllers based on fuzzy logic*, in Proceedings of the 1^{st} International ICSC-NAISO Congress on Neuro-Fuzzy Technologies (NF'02), pp. 81-86, 2002. 18

[Castellano 02] Castellano G., Fanelli A.M., Mencar C., *Generation of interpretable fuzzy granules by a double-clustering technique*, Archives of Control Sciences, vol. 12, no. 4, pp. 397-410, 2002. 28

[Castro 09] Castro C.Y., Llanos C.H., de Britto Vidal Filho W., dos Santos Coehlo, L., *Fuzzy control for cyclist robot stability using FPGAs*, in Proceedings of the 2009 International Conference on Reconfigurable Computing and FPGAs (ReConFig'09), pp. 410-415, 2009. 22

[Chang 02] Chang S.-J., Li T.-H.S., *Design and implementation of fuzzy parallel-parking control for a car-type mobile robot*, Journal of Intelligent and Robotics Systems, vol. 34, no. 2, pp. 175-194, 2002. 18

[Chen 97] Chen G., Zhang D., *Back-driving a truck with suboptimal distance trajectories: a fuzzy logic control approach*, IEEE Transactions on Fuzzy Systems, vol. 5, no. 3, pp. 369-380, 1997. 50

[Chi 95] Chi Z, Wu J., Yan H., *Handwritten numeral recognition using self-organizing maps and fuzzy rules*, Pattern Recognition, vol. 28, no. 1, pp. 59-66, 1995.

[Chiu 94] Chiu S.L., *Fuzzy model identification based on cluster estimation*, Journal of Intelligent and Fuzzy Systems, vol. 2, no. 3, pp. 267-278, 1994. 9

[Chowdhury 08] Chowdhury S.R., Chakrabarti D., Saha H., *FPGA realization of a smart processing system for clinical diagnostic applications using pipelined datapath architectures*, Microprocessors and Microsystems, vol. 32, no. 2, pp. 107-120, 2008. 22

[Comaschi 12] Comaschi F., Genuit B.A.G., Oliveri A., Heemels W.P.M.H., Storace M., *FPGA implementations of piecewise affine functions based on multi-resolution hyperrectangular partitions*, IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 59, no. 12, pp. 2920-2933, 2012.

[Connoly 90] Connoly C.I., Burns J.B., Weiss R., *Path planning using Laplace's equation*, in Proceedings of the 1990 IEEE International Conference on Robotics and Automation (ICRA'90), 1990. 52

[Cormen 01] Cormen T.H., Leiserson C.E., Rivest R. L., Stein C., *Introduction to algorithms*, MIT Press, Cambridge, MA, USA, 2001.

[Cuesta 04] Cuesta F., Gómez-Bravo F., Ollero A., *Parking maneuvers of industrial-like electrical vehicles with and without trailer*, IEEE Transactions on Industrial Electronics, vol. 51, no. 2, pp. 257-269, 2004. 49

[Cupertino 06] Cupertino F., Giordano V., Naso D., Delfine L., *Fuzzy control of a mobile robot*, IEEE Robotics & Automation Magazine, vol. 13, no. 4, pp. 74-81, 2006. 55

[Davies 79] Davies D.L., Bouldin D.W., *A cluster separation measure*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 2, pp. 224-227, 1979. 27

[del Campo 08] del Campo I., Echanobe J., Bosque G., Tarela J.M., *Efficient hardware/software implementation of an adaptive neuro-fuzzy system*, IEEE Transactions on Fuzzy Systems, vol. 16, no. 3, pp. 761-778, 2008. 22

[Demirli 09] Demirli K., Khoshnejad M., *Autonomous parallel parking of a car-like mobile robot by a neuro-fuzzy sensor-based controller*, Fuzzy Sets and Systems, vol. 160, no.19, pp. 2876-2891, 2009. 50

[Desaulniers 96] Desaulniers G., *On shortest paths for a car-like robot maneuvering around obstacles*, Robotics and Autonomous Systems , vol. 17, no. 3, pp. 139-148, 1996.

[Destercke 07] Destercke S., Guillaume S., Charnomordic B., *Building an interpretable fuzzy rule base from data using ortogonal least squares - Application to a depollution problem*, Fuzzy Sets and Systems, vol. 158, no. 18, pp. 2078-2094, 2007. 30

[Di Stefano 05] Di Stefano A., Giaconia C., *An FPGA-based adaptive fuzzy coprocessor*, chapter of Cabestany J., Prieto A., Sandoval F. (Editors), *Computational intelligence and bioinspired systems*, Lecture Notes in Computer Science, vol. 3512, pp. 590-597, 2005. 20

[Driankov 01] Driankov D., Saffiotti A., eds., *Fuzzy logic techniques for autonomous vehicle navigation*, Springer-Physica Verlag, DE, 2001. 53

[dSpace] dSpace, GmbH, ds1104 Microcontroller Board Reference Manual, `http://www.dspace.com` 55

[Dubins 97] Dubins L.E., *On curves of minimal length with a constrait on average curvature and with prescribed initial and terminal positions and tangents*, American Journal of Mathematics, vol. 79, no. 3, pp. 497-516, 1997. 49

[Dunn 73] Dunn J.C., *A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters*, Journal of Cybernetics, vol. 3, no. 3, pp. 32-57, 1973. 27

[Dunn 74] Dunn J.C., *Well-separated clusters and optimal fuzzy partitions*, Journal of Cybernetics, vol. 4, no. 1, pp. 95-104, 1974. 27

[Echevarría 05] Echevarría P., Martínez M.V., Echanobe J., del Campo I., Tarela J.M., *Design and HW/SW implementation of a class of piecewise-linear fuzzy system*, en actas del XII Seminario Anual de Automática, Electrónica Industrial e Instrumentación (SAAEI'05), pp. 360-364, 2005. 22

[Eichfeld 96] Eichfeld H., Kunemund T., Menke M., *A 12b general-purpose fuzzy logic controller chip*, IEEE Transactions on Fuzzy Systems. vol. 4, no. 4, pp. 460-475, 1996. 20

[Eichfeld 98] Eichfeld H., Mertens A., Brandmeier T., Waidelich F., Graumann R., Schwab M., *Applications of SAE 81C99x fuzzy coprocessors*, in Proceedings of the 7^{th} IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'98), vol. 1, pp.19-24, 1998. 20

[Espinosa 00] Espinosa J., Vandewalle J., *Constructing fuzzy models with linguistic integrity from numerical data-AFRELI algorithm*, IEEE Transactions on Fuzzy Systems, vol. 8, no. 5, pp. 591-600, 2000. 28

[Fabrizi 99] Fabrizi E., Panzieri S., Ulivi G., *An integrated sensing-guidance system for a robotized wheelchair*, in Proceedings of the 14^{th} World Conference of the International Federation of Automatic Control (IFAC'99), 1999. 53

[Fernández 01] Fernández-Ramos R., *Navegación autónoma de robots móviles en entornos parcialmente conocidos*, Tesis Doctoral, Universidad de Málaga, 2001. 51

[Ferruz 03] Ferruz J., Blanco V., Ollero A., Acevedo J.V., *An embedded DSP-based controller for the ROMEO-4R vehicle*, in Proceedings of the 5^{th} IFAC International Symposium on Intelligent Components and Instruments for Control Applications (SICICA'03), pp. 101-106, 2003. 60

[Fortuna 03] Fortuna L., Lo Presti M., Vinci C., Cucuccio A., *Recent trends in fuzzy control of electrical drives: An industry point of view*, in Proceedings of the 2003 International Symposium on Circuits and Systems (ISCAS'03), vol. 3, pp. 459-461, 2003. 20

[Fu 10] Fu Y., Li H., Kaye M.E, *Hardware/software codesign for a fuzzy autonomous road-following system*, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 40, no. 6, pp. 690-696, 2010. 22

[Fukuda 99] Fukuda T., Kubota N., *An intelligent robotic system based on a fuzzy approach*, in Proceedings of the IEEE, vol.87, no. 9, pp. 1448-1470, 1999. 53

[Gacto 11] Gacto M.J., Alcalá R., Herrera F., *Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures*, Information Sciences, vol. 181, no. 20, pp. 4340-4360, 2011. 25, 29

[García 98] García-Rosa R., De Pedro T., *Modeling a fuzzy coprocessor and its programming language*, Mathware & Soft Computing, vol. 5, no. 2-3, pp. 167-174, 1998. 20

[Gath 89] Gath I., Geva A.B., *Unsupervised optimal fuzzy clustering*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 7, pp. 773-780, 1989. 7

[Gómez 01] Gómez-Bravo F., Cuesta F., Ollero A., *Parallel and diagonal parking in non-holonomic autonomous vehicles*, Engineering Applications of Artificial Intelligence, vol. 14, no. 4, pp. 419-434, 2001. 49

[Goodman 92] Goodman R.M., Higgins C.M., Miller J.W., Smyth P., *Rule-based neural networks for classification and probability estimation*, Neural Computation, vol. 4, no. 6, pp. 781-804, 1992. 29

[Graves 10] Graves D., Pedrycz W., *Kernel-based fuzzy clustering and fuzzy clustering: A comparative experimental study*, Fuzzy Sets and Systems, vol. 161, no. 4, pp. 522-543, 2010. 8

[Griswold 90] Griswold N.C., Eem J., *Control of mobile robots in the presence of moving objects*, IEEE Transactions on Robotics and Automation, vol. 6, no. 2, pp. 263-268, 1990. 54

[Groenen 01] Groenen P.J.F., Jajuga K., *Fuzzy clustering with squared Minkowski distances*, Fuzzy Sets and Systems, vol. 120, no. 2, pp. 227-237, 2001. 7

[Groenen 07] Groenen P.J.F., Kaymak U., van Rosmalen J., *Fuzzy clustering with Minkowski distance functions*, chapter of de Oliveira J.V., Pedrycz W. (Editors), *Advances in fuzzy clustering and its applications*, John Wiley & Sons, Ltd, pp. 53-68, 2007. 7

[Gustafson 79] Gustafson D.E., Kessel W.C., *Fuzzy clustering with a fuzzy covariance matrix*, in Proceedings of the 1978 IEEE Conference on Decision and Control including the 17^{th} Symposium on Adaptive Processes (CDC'78), pp. 761-766, 1978. 6

[Hathaway 01] Hathaway R.J., Bezdek J.C., *Fuzzy c-means clustering of incomplete data*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 31, no. 5, pp. 735-744, 2001. 5

[Haykin 94] Haykin S., *Neural networks: A comprehensive foundation*, IEEE Press Macmillan, 1994. 1

[Hernández 12] Hernández-Zavala A., Camacho-Nieto O., *Fuzzy hardware: A retrospective and analysis*, IEEE Transactions on Fuzzy Systems, vol. 20, no. 4, pp. 623-635, 2012. 21

[Higgins 94] Higgins C.M., Goodman R.M., *Fuzzy rule-based networks for control*, IEEE Transactions on Fuzzy Systems, vol. 2, no. 1, pp. 82-88, 1994. 11

[Huang 09] Huang H.-C., Tsai C.-C., *FPGA implementation of an embedded robust adaptive controller for autonomous omnidirectional mobile platform*, IEEE Transactions on Industrial Electronics, vol. 56, no. 5, pp. 1604-1616, 2009. 22

[Huang 12] Huang H.-C., Chuang Y.-Y., Chen C.-S., *Multiple kernel fuzzy clustering*, IEEE Transactions on Fuzzy Systems, vol. 20, no. 1, pp. 120-134, 2012. 8

[Huwendiek 99] Huwendiek O., Brockmann W., *Function approximation with decomposed fuzzy systems*, Fuzzy Sets and Systems, vol. 101, pp. 273-286, 1999. 39, 41

[Hybrid] *Hybrid Toolbox*, `http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox`

[Ishibuchi 01] Ishibuchi H., Nakashima T., *Effect of rule weights in fuzzy rule-based classification systems*, IEEE Transactions on Fuzzy Systems, vol. 9, no. 4, pp. 506-515, 2001.

[Islam 06] Islam M.S., Azim M.A., Jahan M.S., Othman M., *Design and synthesis of mobile robot controller using fuzzy*, in Proceedings of the 2006 IEEE International Conference on Semiconductor Electronics (ICSE '06), pp. 825-829, 2006. 56

[Jang 97] Jang J.-S.R., Sun C.-T., Mizutani E., *Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence*, Prentice Hall, Upper Saddle River, NJ, USA, 1997. 2

[Jiménez 00] Jiménez C.J., *Desarrollo en hardware digital de sistemas difusos con una arquitectura optimizada*, Tesis Doctoral, Universidad de Sevilla, 2000.

[Johansen 07] Johansen T.A., Jackson W., Schreiber R., Tøndel P., *Hardware synthesis of explicit model predictive controllers*, IEEE Transactions on Control Systems Technology, vol. 15, no. 1, pp. 191-197, 2007.

[Joo 02] Joo M.G., Lee J.S., *Universal approximation by hierarchical fuzzy system with constraints on the fuzzy rule*, Fuzzy Sets and Systems, vol. 130, no. 2, pp. 175-188, 2002. 39, 41

[Joo 05] Joo M.G., Lee J.S., *A class of hierarchical fuzzy systems with constraints on the fuzzy rules*, IEEE Transactions on Fuzzy Systems, vol. 13, no. 2, pp. 194-203, 2005. 39, 41

[Jou 93] Jou C.-C., Wang N.-C., *Training a fuzzy controller to back up an autonomous vehicle*, in Proceedings of the 1993 IEEE International Conference on Robotics and Automation (ICRA'93), vol. 1, pp. 923-928, 1993. 50

[Julián 99] Julián P., Desages A., Agamennoni O., *High-level canonical piecewise linear representation using a simplicial partition*, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 46, no. 4, pp. 463-480, 1999.

[Kant 86] Kant K., Zucker S.W., *Toward efficient trajectory planning: the path-velocity decomposition*, The International Journal of Robotics Research, vol. 5, no. 3, pp. 72-89, 1986. 53

[Kikuchi 02] Kikuchi H., Takagi N., *Hierarchical fuzzy modeling and jointly expandable functions*, International Journal of Intelligent Systems, vol. 17, pp. 515-529, 2002. 35

[Kimiaghalam 01] Kimiaghalam B., Homaifar A., Suttikulvet B., Sayyarrodsari B., *A multi-layered multi fuzzy inference systems for autonomous robot navigation and obstacle avoidance*, in Proceedings of the 10^{th} IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'01), vol. 1, pp. 340-343, 2001. 53

[Klawonn 97] Klawonn F., Kruse R., *Constructing a fuzzy controller from data*, Fuzzy Sets and Systems, vol. 85, no. 2, pp. 177-193, 1997. 4

[Kolmogorov 57] Kolmogorov A.N., *On the representation of continuous functions of many variables by superposition of continuous functions of one variables and addition*, in Doklady Akademii Nauk BSSR, vol. 144, pp. 679-681, 1957. 44

[Kolmogorov 63] Kolmogorov A.N., *On the representation of continuous functions of many variables by superposition of continuous functions of one variables and addition*, Transactions of American Math Society, vol. 2, no. 28, pp. 55-59, 1963. 44

[Kong 92] Kong S.-G., Kosko B., *Comparison of fuzzy and neural truck backer-upper control systems*, in Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence, Prentice-Hall, Englewood Cliffs, NJ, 1992. 50

[Kosko 91] Kosko B., *Neural networks and fuzzy systems: A dynamical systems approach to machine intelligence*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1991.

[Kubota 92] Kubota T., Hashimoto H., *A strategy for collision avoidance among moving obstacle for a mobile robot*, in Proceedings of the 11^{th} World Conference of the International Federation of Automatic Control (IFAC'92), 1992. 53

[Kyriakopoulos 93] Kyriakopoulos K.J., Saridis G.N., *An integrated collision prediction and avoidance scheme for mobile robots in non-stationary environments*, Automatica, vol. 29, no. 2, pp. 309-322, 1993. 54

[Lazzerini 00] Lazzerini B., Marcelloni F., *A linguistic fuzzy recogniser of off-line handwritten characters*, Pattern Recognition Letters, vol. 21, no. 4, pp. 319-327, 2000.

[Lee 01] Lee S.G., *Parallel fuzzy inference system for large volumes of remote sensing data*, in Proceedings of the 2001 IEEE International Symposium on Industrial Electronics (ISIE'01), vol.1, pp. 296-301, 2001. 18

[Li 03a] Li T.-H.S., Chang S.-J., *Autonomous fuzzy parking control of a car-like mobile robot*, IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, vol. 33, no. 4, pp. 451-465, 2003. 50

[Li 03b] Li T.-H.S., Chang S.-J., Chen Y.-X., *Implementation of human-like driving skills by autonomous fuzzy behavior control on an FPGA-based car-like mobile robot*, IEEE Transactions on Industrial Electronics, vol. 50, no. 5, pp. 867-880, 2003. 56

[Li 07] Li Q., Shi Z., Shi Z., *Linguistic expression based image description framework and its application to image retrieval*, chapter of Nachtegael M., van der Weken D., Kerre E.E., Philips W. (Editors), *Soft computing in image processing: Recent Advances*, Series Studies in Fuzziness and Soft Computing, vol. 210, pp. 97-120, 2007.

[Lin 92] Lin J.-N., Unbehauen R., *Canonical piecewise-linear approximations*, IEEE Transactions on Circuits and Systems I: Theory and Applications, vol. 39, no. 8, pp. 697-699, 1992.

[Lizárraga 08] Lizárraga G., Sepúlveda R., Montiel O., Castillo O., *Modeling and simulation of the defuzzification stage using Xilinx System Generator and Simulink*, chapter of Castillo O., Melin P., Kacprzyk J., Pedrycz W. (Editors), *Soft computing for hybrid intelligent systems*, Studies in Computational Intelligence, vol. 154, pp. 333-343, 2008. 22

[Lorentz 66] Lorentz G.G., *Approximation of functions*, Holt, Reinhart and Winston, 1966. 44

[Mahyuddin 09] Mahyuddin M.N., Wei C.Z., Arshad M.R., *Neuro-fuzzy algorithm implemented in Altera's FPGA for mobile robot's obstacle avoidance mission*, in Proceedings of the 2009 IEEE Region 10 Conference (TENCON'09), pp. 1-6, 2009.

[Martínez 15] Martínez-Rodríguez M.C., Brox P., Baturone I., *Digital VLSI implementation of piecewise-affine controllers based on lattice approach*, IEEE Transactions on Control Systems Technology, vol. 23, no. 3, pp. 842-854, 2015.

[Mastorocostas 01] Mastorocostas P.A., Theocharis J.B., Petridis V.S., *A constrained ortogonal least-squares method for generating TSK fuzzy models: Application to short-term load forecasting*, Fuzzy Sets and Systems, vol. 118, no. 2, pp. 215-233, 2001. 30

[Mayne 00] Mayne D.Q., Rawlings J.B., Rao C.V., Scokaert P.O.M., *Constrained model predictive control: Stability and optimality*, Automatica, vol. 36, no. 6, pp. 789-814, 2000.

[McCluskey 56] McCluskey E.J., *Minimization of boolean functions*, The Bell System Technical Journal, vol. 35, no. 6, pp. 1417-1444, 1956.

[Mencar 07] Mencar C., Castellano G., Fanelli A.M., *Distinguishability quantification of fuzzy sets*, Information Sciences, vol. 177, no. 1, pp. 130-149, 2007. 29

[Mendel 95] Mendel J.M., *Fuzzy logic systems for engineering: A tutorial*, in Proceedings of the IEEE, vol. 83, no. 3, p. 345-377, 1995. 1

[Meng 05] Meng J.E., Chang D., *Obstacle avoidance of a mobile robot using hybrid learning approach*, IEEE Transactions on Industrial Electronics, vol. 52, no. 3, pp. 898-905, 2005. 53

[Millán 08] Millán I., Montiel O., Sepúlveda R., Castillo O., *Design and implementation of a hybrid fuzzy controller using VHDL*, chapter of Castillo O., Melin P., Kacprzyk J., Pedrycz W. (Editors), *Soft computing for hybrid intelligent systems*, Studies in Computational Intelligence, vol. 154, pp. 437-446, 2008. 22

[Mizumoto 82] Mizumoto M., Zimmermann H.-J., *Comparison of fuzzy reasoning methods*, Fuzzy Sets and Systems, vol. 8, no. 3, pp. 253-283, 1982. 1

[MOBY-DIC] *MOBY-DIC Toolbox*, `http://ncas.dibe.unige.it/software/MOBY-DIC_Toolbox`

[Monmasson 07] Monmasson E., Cirstea M.N., *FPGA design methodology for industrial control systems - A review*, IEEE Transactions on Industrial Electronics, vol. 54, no. 4, pp. 1824-1842, 2007. 21

[Monmasson 11] Monmasson E., Idkhajine L., Cirstea M.N., Bahri I., Tisan A., Naouar M.W., *FPGAs in industrial control applications*, IEEE Transactions on Industrial Informatics, vol. 7, no. 2, pp. 224-243, 2011. 21

[Moreno 07] Moreno-Velo F.J., Baturone I., Barriga A., Sánchez-Solano S., *Automatic tuning of complex fuzzy systems with Xfuzzy*, Fuzzy Sets and Systems, vol. 158, no. 18, pp. 2026-2038, 2007.

[MPT] *Multi-Parametric Toolbox (MPT)*, `http://people.ee.ethz.ch/~mpt/3`

[Müler 01] Müler K.-R., Mika S., Rätsch G., Tsuda K., Schölkopf B., *An introduction to kernel-based learning algorithms*, IEEE Transactions on Neural Networks, vol. 12, no. 2, pp. 181-201, 2001. 8

[Nachtegael 00] Nachtegael M., Kerre E.E., *Classical and fuzzy approaches towards mathematical morphology*, chapter of Nachtegael M., Kerre E.E. (Editors), *Fuzzy techniques in image processing*, Studies in Fuzziness and Soft Computing, vol. 52, pp. 3-57, 2000.

[Nauck 00] Nauck D., *Data analysis with neuro-fuzzy methods*, Habilitation Thesis, University of Magdeburg, Germany, 2000. 10, 30, 31

[Naus 10] Naus G.J.L., Ploeg J., Van de Molengraft MJ.G., Heemels W.P.M.H., Steinbuch M., *Design and implementation of parameterized adaptive cruise control: An explicit model predictive control approach*, Control Engineering Practice, vol. 18, no. 8, pp. 882-892, 2010.

[Nijhuis 94] Nijhuis J., van Aartsen H., Barakova E., Jansen W., Spaanenburg B., *On the optimal mapping of fuzzy rules on standard micro-controllers*, Microprocessing and Microprogramming, vol. 40, no. 10-12, pp. 697-700, 1994. 19

[Nilsson 80] Nilsson N.J., *Principles of artificial intelligence*, Morgan Kaufmann, San Francisco, CA, USA, 1980. 51

[Oliveira 10] Oliveira D.N., de Souza Braga A.P., da Mota Almeida O., *Fuzzy logic controller Implementation on a FPGA using VHDL*, in Proceedings of the 2010 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS'10), pp.1-6, 2010. 22

[Oliveri 09] Oliveri A., Oliveri A., Poggi T., Storace M., *Circuit implementation of piecewise-affine functions base on a binary search tree*, in Proceedings of the 19^{th} European Conference on Circuit Theory and Design (ECCTD'09), pp. 145-148, 2009.

[Oliveri 11] Oliveri A., Naus G.J.L., Storace M., Heemels W.P.M.H., *Low-complexity approximations of PWA functions: A case study on adaptive cruise control*, in Proceedings of the 20^{th} European Conference on Circuit Theory and Design (ECCTD'11), pp. 669-672, 2011.

[Ollero 91] Ollero A., Amidi O., *Predictive path tracking of mobile robots. Applications to the CMU Navlab*, in Proceedings of the 5^{th} IEEE Internacional Conference on Advanced Robotics (ICAR'91), vol. 2, pp. 1081-1086, 1991. 51

[Ollero 01] Ollero A., *Robótica: Manipuladores y robots móviles*, Marcombo, 2001.

[Paiz 06] Paiz C., Chinapirom T., Witkowski U., Porrmann M., *Dynamically reconfigurable hardware for autonomous mini-robots*, in Proceedings of the 32^{nd} Annual Conference on IEEE Industrial Electronics (IECON'06), pp. 3981-3986, 2006. 56

[Pal 14] Pal N.R., Sarkar K., *What and when can we gain from the kernel versions of c-means algorithm?*, IEEE Transactions of Fuzzy Systems, vol. 22, no. 2, pp. 363-379, 2014. 8

[Paromtichk 98] Paromtichk I., Laugier C., Gusev S.V., Sekhavat S., *Motion control for parking an autonomous vehicle*, in Proceedings of the 5^{th} International Conference on Control, Automation, Robotics and Vision (ICARCV'98), vol. 1, pp. 136-140, 1998. 49

[Patyra 96] Patyra M.J., Mlynek D.M. (Editors), *Fuzzy logic: Implementation and application*, Wiley & Teubner, 1996. 1

[Pedrycz 93a] Pedrycz W., *Fuzzy neural networks and neurocomputations*, Fuzzy Sets and Systems, vol. 56, no. 1, pp. 1-28, 1993.

[Pedrycz 93b] Pedrycz W., *Fuzzy-set based models of neurons and knowledge-based networks*, IEEE Transactions on Fuzzy Systems, vol. 1, no. 4, pp. 254-266, 1993.

[Pedrycz 98] Pedrycz W., Gomide F., *An introduction to fuzzy sets: Analysis and design*, MIT Press, Cambridge, 1998. 1

[Poggi 10] Poggi T., Comaschi F., Storace M., *Digital circuit realization of piecewise-affine functions with nonuniform resolution: Theory and FPGA implementation*, IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 57, no. 2, pp. 131-135, 2010.

[Quine 52] Quine W.V., *The problem of simplifying truth functions*, The American Mathematical Monthly, vol. 59, no. 8, pp. 521-531, 1952.

[Raju 91] Raju G.V.S., Zhou J., Kisner R.A., *Hierarchical fuzzy control*, International Journal of Control, vol. 54, pp.1201-1216, 1991. 33

[Raychev 05] Raychev R., Mtibaa A., Abid M., *VHDL Modelling of a fuzzy co-processor architecture*, in Proceedings of the 2005 International Conference on Computer Systems and Technologies (CompSysTech'05), 2005. 20

[Reeds 90] Reeds J.A., Shepp R.A., *Optimal paths for a car that goes both forwards and backwards*, Pacific Journal of Mathematics, vol. 145, no. 2, pp. 367-393, 1990. 49

[Reyneri 03] Reyneri, L.M., *Implementation issues of neuro-fuzzy hardware: Going toward HW/SW codesign*, IEEE-Transactions on Neural Networks, vol. 14, no. 1, pp. 176-194, 2003. 21

[Rodríguez 07] Rodríguez-Andina J.J., Moure M.J., Valdes M.D., *Features, design tools, and application domains of FPGAs*, IEEE Transactions on Industrial Electronics, vol. 54, no. 4, pp. 1810-1823, 2007. 21

[Rovatti 98] Rovatti R., *Fuzzy piecewise multilinear and piecewise linear systems as universal approximators in Sobolev norms*, IEEE Transactions on Fuzzy Systems, vol. 6, no. 2, pp. 235-249, 1998. 15, 17

[Rovatti 99] Rovatti R., *High speed implementation of piecewise-quadratic Takagi-Sugeno systems*, in Proceedings of the 8^{th} IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'99), vol. 1, pp. 292-297, 1999. 17

[Rovatti 00] Rovatti R., Fantuzzi C., Simani S., *High-speed DSP-based implementation of piecewise-affine and piecewise-quadratic fuzzy systems*, Signal Processing, vol. 80, no. 6, pp. 951-963, 2000.

[Rudin 76] Rudin W., *Principles of mathematical analysis*. McGraw-Hill, 1976. 41

[Ruspini 98] Ruspini E.H., Bonissone P.P., Pedrycz W. (Editors), *Handbook of fuzzy computation*, Institute of Physics Publishing, Bristol, UK, and Philadelphia, PA, USA, 1998. 8

[Rusu 03] Rusu P., Petriu E.M., Whalen T.E., Cornell A., Spoelder H.J.W., *Behavior-based neuro-fuzzy controller for mobile robot navigation*, IEEE Transactions on Instrumentation and Measurement, vol. 52, no. 4, pp. 1335-1340, 2003. 53

[Salapura 00] Salapura V., *A fuzzy RISC processor*, IEEE Transactions on Fuzzy Systems, vol. 8, no. 6, pp. 781-790, 2000. 19

[Sánchez 07] Sánchez-Solano S., Cabrera A.J., Baturone I., Moreno-Velo F.J., Brox M., *FPGA implementation of embedded fuzzy controllers for robotic applications*, IEEE Transactions on Industrial Electronics, vol. 54, no. 4, pp. 1937-1945, 2007. 61

[Sánchez 13] Sánchez-Solano S., Brox M., del Toro E., Brox P., Baturone I., *Model-based design methodology for rapid development of fuzzy controllers on FPGAs*, IEEE Transactions on Industrial Informatics, vol. 9, no. 3, pp. 1361-1370, 2013. 22, 56

[Sasaki 94] Sasaki M., Ueno F., *A novel implementation of fuzzy logic controller using new meet operation*, in Proceedings of the 3^{rd} IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'94), vol. 3, pp. 1676-1681, 1994. 15

[Savage 00] Savage W., Chilton J., Camposano R., *IP reuse in the system on a chip era*, in Proceedings of the 13^{th} International Symposium on System Synthesis (ISSS'00), pp. 2-7, 2000. 21

[Schölkopf 98] Schölkopf B., Smola A., Müller K.-R., *Nonlinear component analysis as a kernel eigenvalue problem*, Neural Computation, vol. 10, no. 5, pp. 1299-1319, 1998. 8

[Scolari 03] Scolari A.G., Campestrini L., Fehlberg R., Pereira L.F.A., *An integrated hardware and software design of a mobile robot*, in Proceedings of the 29^{th} Annual Conference of the IEEE Industrial Electronics Society (IECON'03), vol. 1, pp. 877-881, 2003. 56

[Senhadji 02] Senhadji R., Sánchez-Solano S., Barriga A., Baturone I., Moreno-Velo F.J., *NORFREA: An algorithm for non redundant fuzzy rule extraction*, in Proceedings of the 2002 IEEE International Conference on Systems, Man and Cybernetics (SMC'02), vol. 1, pp. 604-608, 2002. 10, 30, 31

[Sepúlveda 09] Sepúlveda R., Montiel O., Lizágarra G., Castillo O., *Modeling and simulation of the defuzzification stage of a type-2 fuzzy controller using the Xilinx System Generator and Simulink*, chapter of Castillo O., Pedrycz W., Kacprzyk J. (Editors), *Evolutionary design of intelligent systems in modeling, simulation and control*, Studies in Computational Intelligence, vol. 257, pp. 309-325, 2009. 22

[Setnes 98] Setnes M., Babuška R., Kaymak U., van Nauta Lemke H.R., *Similarity measures in fuzzy rule base simplification*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 28, no. 3, pp. 376-386, 1998. 4, 28, 29

[Setnes 00] Setnes M., Hellendroon H., *Orthogonal transforms for ordering and reduction of fuzzy rules*, in Proceedings of the 9^{th} IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'00), vol. 2, pp. 700-705, 2000. 30

[Smeulders 00] Smeulders A.W.M., Worring M., Santini S., Gupta A., Jain R., *Content-based image retrieval at the end of the early years*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 12, pp. 1349-1380, 2000.

[Snoeyink 97] Snoeyink J., *Point location*, chapter of Goodman J.E., O'Rourke (Editors), *Handbook of discrete and computational geometry*, pp. 559-574, CRC Press, Boca Ratón, FL, USA, 1997.

[Srivastava 11] Srivastava V., Tripathi B.K., Pathak V.K., *An evolutionary fuzzy clustering with Minkowski distances*, chapter of Lu B.-L., Zhang L., Kwok J. (Editors), *Neural information processing*, Lecture Notes in Computer Science, vol. 7063, pp. 753-760, 2011. 7

[Storace 11] Storace M., Poggi T., *Digital architectures realizing piecewise-linear multivariate functions: Two FPGA implementations*, International Journal of Circuit Theory and Applications, vol. 39, no. 1, pp. 1-15, 2011.

[Sugeno 85] Sugeno M, Nishida M., *Fuzzy control of model car*, Fuzzy Sets and Systems, vol. 16, no. 2, pp. 103-113, 1985. 50

[Sugeno 93] Sugeno M., Yasukawa T., *A fuzzy-logic-based approach to qualitative modeling*, IEEE Transactions on Fuzzy Systems, vol. 1, no. 1, pp. 7-31, 1993. 4

[Takagi 85] Takagi T., Sugeno M., *Fuzzy identification of systems and its applications to modeling and control*, IEEE Transactions on Systems, Man, and Cybernetics, vol. SMS-15, no. 1, pp. 116-132, 1985. 6

[Tarela 99] Tarela J.M., Martínez M.V., *Region configurations for realizability of lattice piecewise-linear models*, Mathematical and Computer Modelling, vol. 30, no. 11-12, pp. 17-27, 1999.

[Thareja 07] Thareja V., Bolic M., Groza V., *Design of a fuzzy logic coprocessor using Handel-C*, in Proceedings of the 2^{nd} IEEE International Workshop on Soft Computing Applications (SOFA'07), pp. 83-88, 2007. 20

[TMS320LF2407a] TMS320LF2407 Evaluation Module. Technical Reference, Spectrum Digital, Inc., 2000. 55, 60

[TMS320LF2407b] TMS320LF2407 Reference Guides, Texas Instruments, Inc. 56

[Torra 02] Torra V., *A review of the construction of hierarchical fuzzy systems*, International Journal of Intelligent Systems, vol. 17, pp. 531-543, 2002. 34

[Tsukamoto 79] Tsukamoto, Y., *An approach to fuzzy reasoning method*, chapter of Gupta M.M., Ragade R.K., Yager R.R. (Editors), *Advances in fuzzy set theory and applications*, North-Holland, Amsterdam - New York - Oxford, pp. 137-149, 1979. 1

[Tzafestas 10] Tzafestas S.G., Deliparaschos K.M., Moustris G.P., *Fuzzy logic path tracking control for autonomous non-holonomic mobile robots: Design of System on a Chip*, Robotics and Autonomous Systems, vol. 58, pp. 1017-1027, 2010. 56

[Ungering 94] Ungering A.P., Bauer H., Goser K., *Architecture of a fuzzy-processor based on an 8-bit microprocessor*, in Proceedings of the 3^{rd} IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'94), vol. 1, pp. 297-301, 1994. 19

[Vázquez 94] Vázquez F, García E., *A local guidance method for low-cost mobile robots using fuzzy logic*, Annual Review in Automatic Programming, vol. 19, pp. 203-207, 1994. 53

[Vernon 07] Vernon D., Metta G., Sandini G., *A survey of artificial cognitive systems: implications for the autonomous development of mental capabilities in computational agents*, IEEE Transactions on Evolutionary Computation, vol. 11, no. 2, pp. 151-180, 2007. 48

[Virtex-II] Virtex-II Pro and Virtex-II Pro X FPGA User Guide, Xilinx, 2007.

[von Altrock 98] von Altrock C., *Adapting existing hardware for fuzzy computation*, in Handbook of Fuzzy Computation, Institute of Physics Publishing, 1998. 19

[Wang 92a] Wang L.-X., Mendel J.M., *Fuzzy basis functions, universal approximation, and orthogonal least-squares learning*, IEEE Transactions on Neural Networks, vol. 3, no. 5, pp. 807-814, 1992. 29

[Wang 92b] Wang L.-X., Mendel J.M., *Generating fuzzy rules by learning from examples*, IEEE Transactions on System, Man and Cybernetics, vol. 22, no. 6, pp. 1414-1427, 1992. 10

[Wang 97] Wang L.-X., *A course in fuzzy systems and control*, Prentice Hall, New York, NY, USA, 1997. 1

[Wang 98] Wang L.-X., *Universal approximation by hierarchical fuzzy systems*, Fuzzy Sets and Systems, vol. 130, no. 2, pp. 175-188, 2002. 38

[Wang 99] Wang L.-X., *Analysis and design of hierarchical fuzzy systems*, IEEE Transactions on Fuzzy Systems, vol. 7, no. 5, pp. 617-624, 1999. 38

[Wang 03] Wang X., Yang S.X., *A neuro-fuzzy approach to obstacle avoidance of a nonholonomic mobile robot*, in Proceedings of the 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM'03), vol. 1, pp. 29-34, 2003. 53

[Wang 04] Wang X., Yang S.X., Meng M.Q.-H., *Intelligent obstacle avoidance for an autonomous mobile robot*, in Proceedings of the 5^{th} World Congress on Intelligent Control and Automation (WCICA'04), vol. 5, pp. 4656-4660, 2004. 55

[Wen 09] Wen C., Ma X., Ydstie B.E., *Analytical expression of explicit MPC solution via lattice piecewise-affine function*, Automatica, vol. 45, no. 4, pp. 910-917, 2009.

[Wooldridge 09] Wooldridge M.J., *An introduction to multiagent systems*, John Wiley & Sons Ltd., 2009. 53

[Xfuzzy] *Xfuzzy: Fuzzy Logic Design Tools*, `http://www.imse.cnm.es/Xfuzzy` 22

[Yager 94] Yager R.R, Filev D.P., *Approximate clustering via the mountain method*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 24, no. 8, 1994. 9

[Yen 95] Yen J., Langari R., Zadeh L.A., *Industrial applications of fuzzy logic and intelligent systems*, IEEE Press, New York, NY, USA, 1995. 19

[Yen 99] Yen J., Wang L., *Simplifying fuzzy rule-based models using orthogonal transformation methods*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 29, no. 1, pp. 13-24, 1999. 29

[Zadeh 65] Zadeh L.A., *Fuzzy sets*, Information and Control, vol. 8, no. 3, pp. 338-353, 1965. 1

[Zadeh 73] Zadeh, L.A., *Outline of a new approach to the analysis of complex systems and decision processes*, IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-3, no. 1, pp. 28-44, 1973. 1

[Zeng 05a] Zeng X.-J., Goulermas J.Y., Keane J.A., Liatsis P., *Approximation capabilities of hierarchical neural-fuzzy systems for function approximation on discrete spaces*, International Journal of Computational Intelligence Research, vol. 1, no. 1, pp. 29-41, 2005. 45

[Zeng 05b] Zeng X.-J., Keane J.A., *Approximation capabilities of hierarchical fuzzy systems*, IEEE Transactions on Fuzzy Systems, vol. 13, no. 5, pp. 659-672, 2005. 41, 44

[Zeng 08] Zeng X.-J., Goulermas J.Y., Liatsis P., Wang D., Keane J.A., *Hierarchical fuzzy systems for function approximation on discrete input spaces with application*, IEEE Transactions on Fuzzy Systems, vol. 16, no. 5, pp. 1197-1215, 2008. 45

[Zhang 03] Zhang D.-Q., Chen S.-C., *Clustering incomplete data using kernel-based fuzzy c-means algorithm*, Neural Processing Letters, vol. 18, no. 3, pp. 155-162, 2003. 8

[Zhang 05] Zhang N., Beetner D., Wunsch D.C., Hemmelman B., Hasan A., *An embedded real-time neuro-fuzzy controller for mobile robot navigation*, in Proceedings of the 14^{th} IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'05), pp. 319-324, 2005. 53

[Zhang 10] Zhang N., Kamdem R., Ososanya E., Mahmoud W., Liu W., *VHDL implementation of the hybrid fuzzy logic controllers with FPGA*, in Proceedings of the 2010 International Conference on Intelligent Control and Information Processing (ICICIP'10), pp. 5-10, 2010. 22

[Zhou 06] Zhou S.-M., Gan J.Q., *Constructing accurate and parsimonious fuzzy models with distinguishable fuzzy sets based on an entropy measure*, vol. 157, no. 8, pp. 1057-1074, 2006. 29

[Zhou 08] Zhou S.-M., Gan J.Q., *Low-level interpretability and high-level interpretability: A unified view of data-driven interpretable fuzzy system modelling*, Fuzzy Sets and Systems, vol. 159, no. 23, pp. 3091-3131, 2008. 25, 29

Breve Curriculum Vitae

El autor recibió el título de Licenciado en Física en la Universidad de Córdoba en 2004 con la calificación de Premio Extraordinario. En 2005 obtuvo una beca FPU adscrita a la Escuela Técnica Superior de Ingeniería Informática de la Universidad de Sevilla, que disfrutó hasta 2007. En 2008 obtuvo el Diploma de Estudios Avanzados con la calificación de Sobresaliente en el Programa de Doctorado de Microelectrónica del Departamento de Electrónica y Electromagnetismo de la Universidad de Sevilla. Desde 2007 hasta la actualidad, es profesor a tiempo completo del Departamento de Arquitectura de Computadores, Electrónica y Tecnología Electrónica de la Universidad de Córdoba.

Publicaciones en revistas:

- Quiles F.J., Ortiz M., Gersnoviez A., Brox M., Olivares A., Glösekötter P., *Development of a wireless low power datalogger with high performance converter*, Elektronika ir Elektrotechnika, vol. 21, no. 3, `dx.doi.org/10.5755/j01.eee`, 2015.
- Baturone I., Gersnoviez A., Barriga A., *Neuro-fuzzy techniques to optimize an FPGA embedded controller for robot navigation*, Applied Soft Computing, vol. 21, pp. 95-106, 2014.
- Arjona R., Gersnoviez A., Baturone I., *Fuzzy models for fingerprint description*, Lecture Notes in Computer Science, vol. 6857, pp. 228-235, 2011.
- Barriga A., Sánchez-Solano S., Baturone I., Moreno-Velo F.J., Brox P., Montesino F., Hussein N.M., Brox M., Gersnoviez A., *Fuzzy logic activities at the Microelectronics Institute of Seville*, Lecture Notes in Computer Science, vol. 3931, pp. 157-162, 2006.

Participaciones en congresos internacionales:

- Baturone I., Martínez-Rodríguez M.C., Brox P., Gersnoviez A., *Digital implementation of hierarchical piecewise-affine controllers*, in Proceedings of the 2011 IEEE International Symposium on Industrial Electronics (ISIE'11), pp. 1497-1502, 2011.

- Baturone I., Gersnoviez A., *Automatic extraction of linguistic models for image description*, in Proceedings of the 2010 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'10), pp. 1-8, 2010.
- Baturone I., Sánchez-Solano S., Gersnoviez A., Brox M., *An automated design flow from linguistic models to piecewise polynomial digital circuits*, in Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS'10), pp. 3317-3320, 2010.
- Ortiz M., Brox M., Quiles F.J., Gersnoviez A., Moreno C., Montijano M., *Using soft processors for component design in SOC: A case-study of timers*, in Proceedings of the 2008 International Symposium on System-on-Chip (SOC'08), pp. 1-4, 2008.
- Baturone I., Gersnoviez A., *A simple neuro-fuzzy controller for car-like robot navigation avoiding obstacles*, in Proceedings of the 2007 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'07), pp. 1-6, 2007.
- Baturone I., Moreno-Velo F.J., Sánchez-Solano S., Barriga A., Brox P., Gersnoviez A., Brox M., *Using Xfuzzy environment for the whole design of fuzzy systems*, in Proceedings of the 2007 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'07), pp. 1-6, 2007.
- Baturone I., Moreno-Velo F.J., Gersnoviez A., *A CAD approach to simplify fuzzy system descriptions*, in Proceedings of the 2006 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'06), pp. 2392-2399, 2006.
- Barriga A., Sánchez-Solano S., Baturone I., López D., Moreno-Velo F.J., Brox P., Montesino F., Hussein N.M., Brox M., Gersnoviez A., *New features of the fuzzy logic development environment Xfuzzy*, in Proceedings of the 2006 International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'06), pp. 17-20, 2006.
- Baturone I., Moreno-Velo F.J., Gersnoviez A., *Identifying fuzzy systems from numerical data with Xfuzzy*, in Proceedings of the 2005 Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT'05), pp. 1257-1262, 2005.

Participaciones en congresos nacionales:

- Baturone I., Gersnoviez A., *Diseño de sistemas neuro-difusos para control de robots mediante Xfuzzy 3*, en Actas del IX Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica (TAEE'10), pp. 1-8, 2010.
- Gersnoviez A., Baturone I., *Controlador para navegación reactiva evitando obstáculos mediante técnicas neuro-fuzzy*, en Actas del XIV Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF'08), pp. 551-556, 2008.

- Brox M., Sánchez-Solano S., Brox P., Baturone I., Barriga A., Gersnoviez A., *Síntesis hardware de módulos de inferencia difusos mediante herramientas de diseño de DSP*, en Actas del VIII Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica (TAEE'08), pp. 152, 2008.

- Gersnoviez A., Baturone I., Moreno-Velo F.J., *Extracción de bases de reglas simples y lingüísticamente interpretables*, en Actas del XIII Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF'06), pp. 111-116, 2006.

- Brox M., Gersnoviez A., Sánchez-Solano S., Baturone I., *Controlador difuso para problemas de navegación en presencia de obstáculos fijos*, en Actas del XIII Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF'06), pp. 29-34, 2006.

- Brox M., Gersnoviez A., Sánchez-Solano S., Cabrera A., Baturone I., *Desarrollo de módulos-IP de controladores difusos para el diseño de sistemas empotrados sobre FPGAs*, en Actas del VII Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica, pp. 83-84, 2006.