ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

INGENIERÍA INFORMÁTICA

Aplicación de un método de interpolación, basado en índices de solapamiento, a la detección de riesgos ambientales

Autor: Mikel PINTOR
Tutores: Humberto BUSTINCE
Fco. Javier FERNÁNDEZ
Pamplona,23 de julio de 2014

Índice general

Índice de tablas

Índice de figuras

Resumen

En este trabajo se implementa y evalúa un nuevo método de inferencia difusa basado en índices de solapamiento, presentado por Bustince et al. en [1]. Este método es una generalización del método de interpolación clásico para sistemas difusos basados en reglas y su principal característica es la utilización de índices de solapamiento [2] para evaluar las entradas del sistema y compararlas con los antecedentes de las reglas difusas.

Para evaluar este nuevo método de interpolación, se desarrolla también un caso práctico: la detección y determinación de riesgos ambientales, especialmente incendios forestales, presentado originalmente en [3]. El problema a resolver es la determinación del riesgo de incendio forestal, dadas unas mediciones realizadas por una red de sensores inalámbricos. Esta red de sensores mide magnitudes tales como la temperatura, la humedad relativa, la luminosidad etc. A partir de estas entradas, el sistema difuso presentado en este trabajo es capaz de determinar el riesgo de incendio, basándose en un conjunto de reglas previamente definido. En la resolución de este caso práctico se utilizan tanto el nuevo método de interpolación como el método clásico de Mamdani [4] y se realiza una comparación de los resultados obtenidos con ambos.

Introducción y objetivos

La teoría de conjuntos difusos ha tenido una gran importancia en el campo de la inteligencia artificial, desde que fuera introducida por L.A. Zadeh en 1965 [5]. La principal ventaja de los conjuntos difusos frente a los clásicos es que permiten modelar conceptos vagos o imprecisos. Es precisamente esta propiedad la que ha hecho que los conjuntos difusos se hayan convertido en una pieza clave para multitud de métodos y técnicas de razonamiento aproximado.

Los conjuntos difusos constituyen la base de la lógica difusa que, en palabras del propio Zadeh, pretende emular la habilidad de la mente humana para razonar en términos difusos e imprecisos. Utilizando conjuntos difusos se pueden modelar conceptos vagos como *"Temperatura Alta"* o *"Distancia cercana"*, que no son posibles en la lógica clásica. Estos conceptos además pueden ser utilizados para realizar razonamiento aproximado y control.

Dentro del ámbito de la lógica difusa, una clase muy importante de métodos son los llamados *sistemas difusos basados en reglas*. La principal característica de estos sistemas es que en ellos se modela el conocimiento sobre el problema a resolver utilizando reglas del tipo *SI-ENTONCES*. En este tipo de métodos las entradas del sistema se evalúan contra estas reglas para producir los resultados adecuados. Una ventaja de este tipo de métodos es que el conjunto de reglas puede ser dado directamente por un experto, por lo que no es necesaria la fase de entrenamiento tan característica de otros sistemas como las redes neuronales.

El principal objetivo de este proyecto es la implementación y evaluación de un nuevo sistema difuso basado en reglas, introducido por Bustince et al. [1], que tiene como principal característica la utilización de índices de solapamiento para determinar cómo de parecidas son las entradas a las condiciones (antecedentes) de las reglas difusas. Este nuevo método es una generalización del método de interpolación.

En este proyecto se van a aplicar estos métodos de logica difusa a un caso práctico: la detección y determinación de riesgos ambientales, particularmente incendios forestales [3]. En este caso práctico, el sistema difuso recibirá una serie de magnitudes medidas por una red de sensores inalámbricos, tales como: temperatura, luminosidad, humedad, etc. y deberá determinar el riesgo del incendio forestal, utilizando un conjunto de reglas previamente definido.

Este documento se divide en cuatro capítulos. En el capítulo 1 se introduce la teoría de conjuntos difusos básica para el desarrollo posterior de los métodos estudiados. En este capítulo se introducen conceptos importantes y muy utilizados como las variables lingüísticas, operadores de agregación, t-normas,etc. En este capítulo además se define el concepto de índice de solapamiento y se proporciona un método para su construcción [1].

En el capítulo 2 se introduce el concepto de lógica difusa y se definen las características y propiedades de los sistemas difusos basados en reglas. En este capítulo se introducen además el método clásico de Mamdani [4] y el método de interpolación basado en índices de solapamiento [1], que serán aplicados al caso práctico de detección de riesgos ambientales.

En el capítulo 3 se presenta el caso práctico de la detección y evaluación de riesgo de incendios forestales [3]. En este capítulo se definirán las características del problema a resolver y se aplicarán los métodos estudiados en el capítulo 2, realizando un estudio comparativo entre ellos.

Por último, en el capítulo 4 se presenta una implementación en MATLAB de los métodos estudiados y su aplicación al caso práctico de detección de riesgos de incendios forestales.

1 Teoría de conjuntos difusos

En este primer capítulo se introduce la teoría relacionada con conjuntos difusos necesaria para el desarrollo del proyecto. Los conjuntos difusos forman la base de los métodos estudiados en este proyecto y han tenido una importancia capital en el desarrollo de la inteligencia artificial en las últimas decadas, desde que fueran introducidos por L.A. Zadeh en 1965 [5]. Por esta razón, es necesario definirlos de forma apropiada y estudiar algunas de sus propiedades fundamentales.

1.1. Conjuntos difusos

En esta sección se define el concepto de conjunto difuso (sec. 1.1.1), así como las operaciones más básicas que se pueden realizar sobre ellos (sec. 1.1.3). En la sección 1.1.2 se presenta el concepto de variable lingüística, una de las aplicaciones más importantes de los conjuntos difusos y clave para el desarrollo de la lógica difusa.

1.1.1. Definición y propiedades

Definición 1.1. *Sea* U *un conjunto clásico de objetos, denominado* universo*, cuyos elementos son denotados como* x*. La pertenencia a un subconjunto clásico* A *de* U *se expresa normalmente por medio de una función de pertenencia* $\mu_A : U \to \{0, 1\}$ *tal que:*

$$\mu_A(x) = \begin{cases} 1 & \text{iff } x \in A, \\ 0 & \text{iff } x \notin A \end{cases} \tag{1.1}$$

(se utiliza iff *como abreviatura de "si y sólo si")*

De esta forma, en un conjunto clásico, se dice que un elemento x pertenece al conjunto A ($\mu_A(x) = 1$) o no pertenece ($\mu_A(x) = 0$).

Ejemplo 1.1. Un ejemplo de conjunto clásico es el conjunto de todos los números enteros pares, cuya función de pertenencia $\mu_A : \mathbb{Z} \to \{0, 1\}$ podría expresarse como:

$$\mu_A(x) = \begin{cases} 1 & \text{iff x } \mathrm{mód}\ 2 = 0, \\ 0 & \text{iff x } \mathrm{mód}\ 2 \neq 0 \end{cases} \tag{1.2}$$

Si el valor de salida de la función de pertenencia puede ser cualquier número en el intervalo [0,1] (y no sólo los valores discretos {0,1}) se dice que el conjunto A es *difuso* y que la función $\mu_A(x)$ es el *grado de pertenencia* de x a A. En este tipo de conjuntos, cuanto más próximo sea el valor de $\mu_A(x)$ a 1, más pertenece x a A.

El concepto de conjunto difuso fue introducido por L.A. Zadeh en 1965 [5] y se define como:

Definición 1.2. *Dado un conjunto de referencia (o universo)* U*, un conjunto difuso* A *sobre* U *es un conjunto tal que:*

$$\{(u_i, \mu_A(u_i))|u_i \in U\} \tag{1.3}$$

donde μ_A : U $\rightarrow [0, 1]$ *es la* función de pertenencia *(o* grado de pertenencia*) de* A.

Es habitual también referirse a la función de pertenencia de un conjunto A utilizando la notación $A(u_i)$ (en vez de $\mu_A(u_i)$). En ese caso A denota el conjunto difuso A y $A(u_i)$ el valor de la función de pertenencia de dicho conjunto para el elemento u_i. En este trabajo se van a utilizar ambas notaciones indistintamente.

Definición 1.3. *Dado un conjunto de referencia (o universo) U, denotamos como $FS(U)$ al conjunto de todos los conjuntos difusos definidos sobre U.*

Ejemplo 1.2. Supongamos que el universo *U* está compuesto por los valores razonables de altura (medida en centímetros) de una persona adulta, tal que:

$$U = \{150, 151, \ldots, 229, 230\} \tag{1.4}$$

Se puede definir sobre *U* el conjunto difuso *VT*, para representar el concepto de altura *"Muy alta"*, especificando su función de pertenencia μ_{VT}, de forma que los valores de altura μ_i considerados como *"muy altos"* hagan que el valor de dicha función de pertenencia $\mu_{VT}(\mu_i)$ sea más próximo a 1 (pertenecen más al conjunto).

Se puede elegir cualquier tipo de función para definir al conjunto difuso, siempre que se ajuste a las necesidades del problema que se pretende resolver. En este ejemplo se va a definir la función de pertenencia μ_{VT} como una función sigmoidal tal que:

$$\mu_{VT}(\mu_i) = \frac{1}{1 + e^{-0,4(\mu_i - 190)}} \tag{1.5}$$

La función de pertenencia $\mu_{VT}(\mu_i)$ se representa en la figura 1.1. Se puede observar que alturas que no podrían considerarse muy altas como 150, 160, 170,etc. hacen que la función de pertenencia μ_{VT} sea cero (no pertenencen al conjunto). Algunos valores del conjunto difuso *VT*:

$$VT = \{(150, 0), (160, 0), (170, 0), \ldots, (190, 0{,}5), (195, 0{,}88), (200, 1), (210, 1), \ldots\} \tag{1.6}$$

1.1.2. Variables lingüísticas

Una variable lingüística se puede definir de manera informal como una variable que puede tomar palabras del lenguaje natural como valores. Para formular palabras y utilizarlas como valores de variables lingüísticas se utilizan conjuntos difusos (por medio de sus correspondientes funciones de pertenencia). Así pues, una variable lingüística se puede definir como [6]:

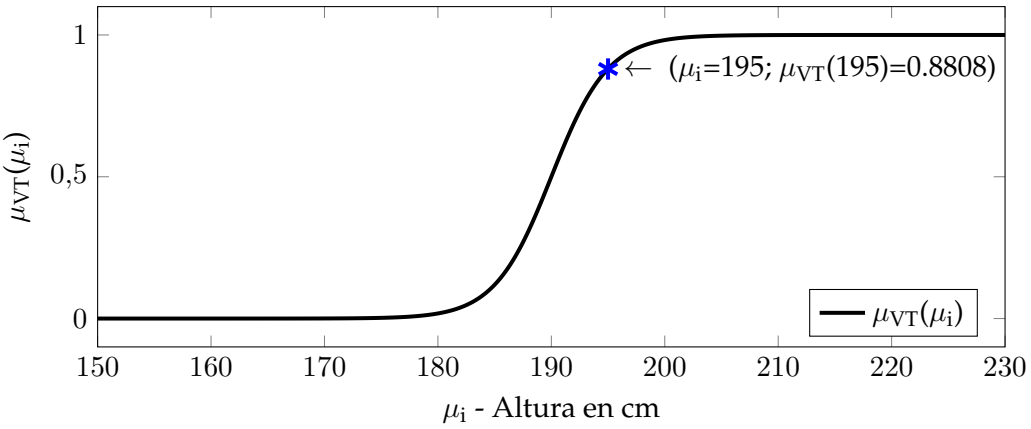


Figura 1.1: Función de pertenencia del conjunto difuso *"Muy Alto"* ($\mu_{VT}(\mu_i)$).

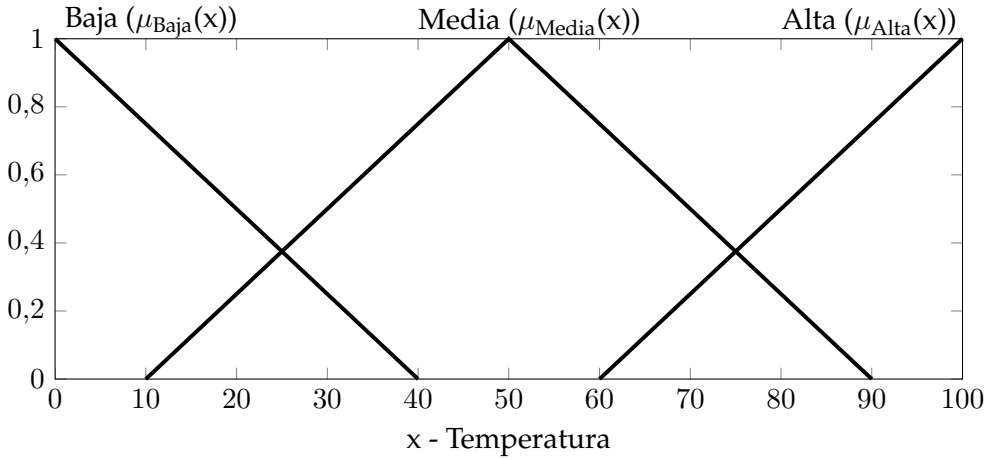


Figura 1.2: Ejemplo 1.3: Funciones de pertenencia para los valores *"baja"* (μ_{Baja}),*"media"* (μ_{Media}) y *"alta"* (μ_{Alta}) de la variable lingüística *Temperatura*.

Definición 1.4. *Una* variable lingüística *es un tipo de variable que puede tomar palabras como valores. Las palabras son caracterizadas por conjuntos difusos definidos en el mismo universo en el que la variable es definida.*

Ejemplo 1.3. Supongamos que la temperatura medida por un termómetro en grados celsius (ºC) es una variable x que puede tomar valores enteros en el intervalo $[0, 100]$. Este intervalo de valores posibles es lo que se define como el *universo de discurso* o *universo de referencia* U. Podemos definir sobre U los conjuntos difusos *"baja"*,*"media"* y *"alta"* para modelar los conceptos de temperatura alta, media y baja respectivamente. Si consideramos que x es una variable lingüística entonces x puede tomar los valores *"baja"*,*"media"* y *"alta"*. Es decir, podemos decir que: *"x es baja"*,*"x es media"* o *"x es alta"*, o lo que es lo mismo *"Temperatura es baja"*,*"Temperatura es media"* o *"Temperatura es alta"*. En la figura 1.2 se representan las funciones de pertenencia para los conjuntos difusos *"Temperatura baja"*,*"Temperatura media"* y *"Temperatura alta"*.

Una definición más formal de *variable lingüística* (equivalente a la definición 1.4), dada por Zadeh [7], es la siguiente:

Definición 1.5. Una *variable lingüística* se caracteriza por la tupla (X, T, U, M), donde:

- X es el nombre de la variable lingüística. En el ejemplo 1.3,"Temperatura en ºC".

- T es el conjunto de valores lingüísticos que X puede tomar. En el ejemplo 1.3, T = {baja, media, alta}

- U es el universo de referencia del que cada valor lingüístico toma sus valores cuantitativos (escalares). En el ejemplo 1.3, $U = [0, 100]$

- M es la regla semántica que relaciona cada valor lingüístico en T con un conjunto difuso en U. En el ejemplo 1.3, M relaciona los valores "baja", "media" y "alta" con las funciones de pertenencia representadas en la figura 1.2.

1.1.3. Operaciones sobre conjuntos difusos

En esta sección se definen algunas operaciones básicas sobre conjuntos difusos. Un conjunto difuso compuesto a partir de dos conjuntos difusos sobre el mismo universo *U* puede definirse por su función de pertenencia tal que:

Definición 1.6. *Dada una función* F : $[0, 1]^2 \to [0, 1]$ *y dos conjuntos difusos* A *y* B *definidos sobre el mismo universo* U, $A, B \in FS(U)$, *denotamos como* $F(A, B)$ *el conjunto difuso sobre* U *cuya función de pertenencia viene dada por:*

$$\mu_{F(A,B)}(u_i) = F(A(u_i), B(u_i)) \tag{1.7}$$

De esta forma, se pueden definir las operaciones de unión ($\cup$) y de intersección ($\cap$) clásicas sobre los subconjuntos difusos *A* y *B* sobre *U* [8] tal que:

$$\forall \mu_i \in U, \quad \mu_{A \cup B}(u_i) = max(\mu_A(u_i), \mu_B(u_i)) \tag{1.8}$$

$$\forall \mu_i \in U, \quad \mu_{A \cap B}(u_i) = min(\mu_A(u_i), \mu_B(u_i)) \tag{1.9}$$

donde $\mu_{A \cup B}$ y $\mu_{A \cap B}$ son las funciones de pertenencia de $A \cup B$ (unión) y $A \cap B$ (intersección) respectivamente.

Definición 1.7. *El* complementario $\overline{A}$ *de un conjunto difuso* A *sobre* U *puede definirse por su función de pertenencia [5], tal que:*

$$\forall \mu_i \in U, \quad \mu_{\overline{A}}(\mu_i) = 1 - \mu_A(\mu_i) \tag{1.10}$$

1.2. T-normas y operadores de agregación

Dada una función $F : [0, 1]^2 \to [0, 1]$ y dos conjuntos difusos *A* y *B* definidos sobre el mismo universo *U*, $A, B \in FS(U)$, denotamos como $F(A, B)$ el conjunto difuso sobre *U* cuya función de pertenencia viene dada por:

$$\mu_{F(A,B)}(u_i) = F(A(u_i), B(u_i)) \tag{1.11}$$

Una clase importante de este tipo de funciones son las llamadas t-normas y t-conormas (normas triangulares) [9].

Definición 1.8. *Una t-norma es una operación binaria* T *en el intervalo* $[0, 1]$ *que es conmutativa, asociativa, monótona y tiene el valor* 1 *como elemento neutro. Es decir, una función* T : $[0, 1]^2 \to [0, 1]$ *tal que* $\forall x, y, z \in [0, 1]$*:*

(T1) $T(x, y) = T(y, x)$ *(Conmutatividad)*

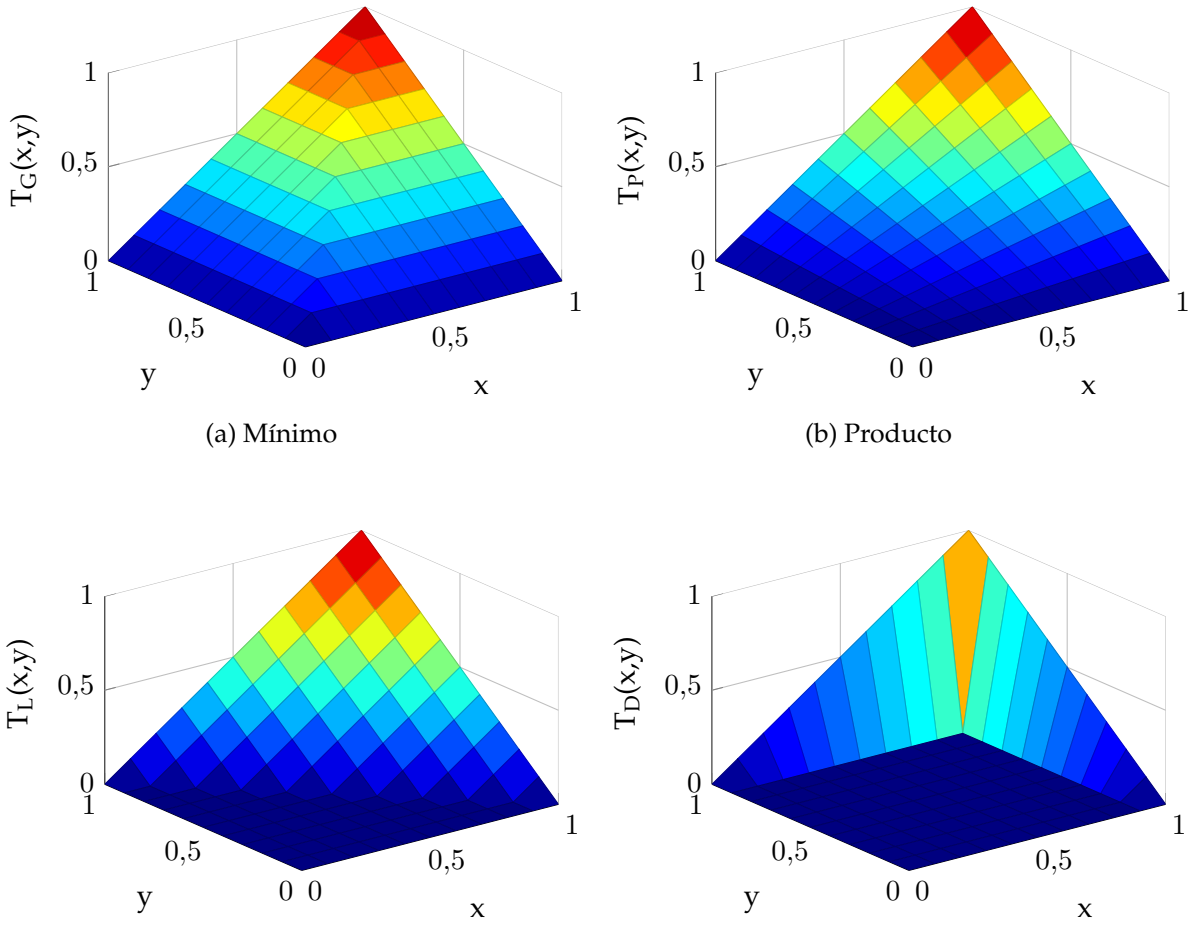
(T2) $T(x, T(y, z)) = T(T(x, y), z)$ *(Asociatividad)*

(T3) $T(x, y) \leq T(x, z)$ *cuando* $y \leq z$ *(Monotonía)*

(T4) $T(x, 1) = x$ *(Elemento neutro)*

Estas propiedades son suficientes para garantizar que las t-normas generalizan la conjunción clásica ($x \wedge y$) cuando se aplican a valores booleanos ($T(0,0) = 0$,$T(0,1) = T(1,0) = 0$ y $T(1,1) = 1$). Algunos ejemplos prominentes de t-normas son los siguientes:

- **Mínimo:** $T_G(x, y) = min\{x, y\}$
- **Producto:** $T_P(x, y) = xy$
- **Łukasiewicz:** $T_L(x, y) = max\{x + y - 1, 0\}$
- **T-norma drástica:** $T_D(x, y) = \begin{cases} y & if \quad x = 1, \\ x & if \quad y = 1, \\ 0 & \text{en cualquier otro caso.} \end{cases}$



(a) Mínimo

(b) Producto

(c) Łukasiewicz

(d) T-norma drástica

Figura 1.3: Representación gráfica de las t-normas *Mínimo* (1.3a), *Producto* (1.3b), *Łukasiewicz* (1.3c) y la *t-norma drástica* (1.3d).

Definición 1.9. *Una t-conorma es una operación binaria* S *en el intervalo* $[0, 1]$ *que es conmutativa, asociativa, monotona y tiene el valor* 0 *como elemento neutro. Es decir, una función* $\mathrm{S} : [0, 1]^2 \to [0, 1]$ *tal que* $\forall x, y, z \in [0, 1]$*:*

(S1) $S(x, y) = S(y, x)$ *(Conmutatividad)*

(S2) $S(x, S(y, z)) = S(S(x, y), z)$ *(Asociatividad)*

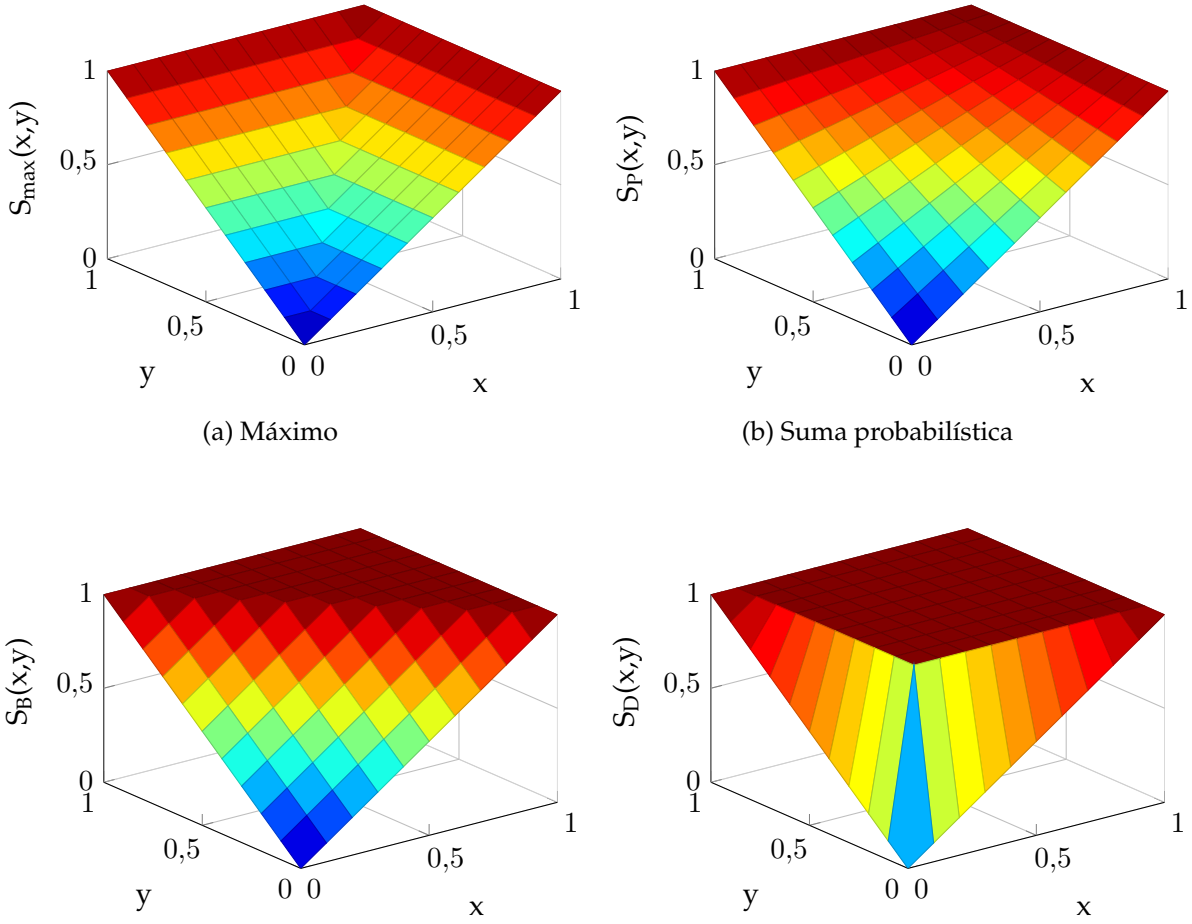
(S3) $S(x, y) \leq S(x, z)$ *cuando* $y \leq z$ *(Monotonía)*

(S4) $S(x, 0) = x$ *(Elemento neutro)*

De la misma forma que las t-normas (definición 1.8) generalizan la conjunción clásica, las t-conormas generalizan la disyunción ($x \vee y$). A continuación se incluyen algunos ejemplos de t-conormas:

- **Máximo:** $S_{max}(x, y) = max\{x, y\}$
- **Suma probabilística:** $S_P(x, y) = x + y - xy$

- **Suma acotada:** $S_B(x,y) = min\{x+y, 1\}$
- **T-conorma drástica:** $S_D(x,y) = \begin{cases} y & if \quad x = 0, \\ x & if \quad y = 0, \\ 1 & \text{en cualquier otro caso.} \end{cases}$



(a) Máximo

(b) Suma probabilística

(c) Suma acotada

(d) T-conorma drástica

Figura 1.4: Representación gráfica de las t-conormas *Máximo* (1.4a), *Suma probabilística* (1.4b), *Suma acotada* (1.4c) y la *t-conorma drástica* (1.4d).

Otra clase importante de funciones son los *operadores de agregación* [10, 11]. Los operadores de agregación son funciones que toman uno o varios parámetros y devuelven un único valor, que de alguna manera representa a los parámetros de entrada agregados.

Definición 1.10. *Una función* $\mathrm{M} : [a,b]^n \to [a,b]$ *es un operador de agregación si es monótona y no decreciente en cada una de sus componentes y además cumple que* $M(a,a,\cdots,a) = a$ *y* $M(b,b,\cdots,b) = b$.

Definición 1.11. *Una función de agregación* M *se dice que es una media si cumple que:*

$$min(x) = min(x_1,\cdots,x_n) \leq \mathrm{M}(x_1,\cdots,x_n) \leq max(x_1,\cdots,x_n) = max(x) \tag{1.12}$$

A continuación se incluyen algunos de los operadores de agregación más comunes:

- **Media aritmética:** $M(x_1, x_2, \cdots, x_n) = \frac{1}{n} \sum_{i=1}^{n} x_i$
- **Media geométrica:** $M(x_1, x_2, \cdots, x_n) = (\prod_{i=1}^{n} x_i)^{\frac{1}{n}}$
- **Media ponderada:** $M_{w_1,w_2,\cdots,w_n}(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n} (w_i \cdot x_i)$ tal que $0 \leq w_i \leq 1$ y $\sum_{i=1}^{n} w_i = 1$.
- **Mediana:** Se toma el elemento central del conjunto ordenado de argumentos.
- **Máximo:** $M(x_1, x_2, \cdots, x_n) = max(x_1, x_2, \cdots, x_n)$
- **Mínimo:** $M(x_1, x_2, \cdots, x_n) = min(x_1, x_2, \cdots, x_n)$

1.3. Funciones de solapamiento

En esta sección se presenta el concepto de *función de solapamiento* [2, 12, 13, 14], que tendrá una gran importancia en el desarrollo de este trabajo, ya que a partir de estas funciones se pueden construir índices de solapamiento.

1.3.1. Definición de función de solapamiento y teorema de construcción.

Definición 1.12. *Una función de solapamiento* $G_O : [0,1]^2 \to [0,1]$ *cumple:*

(G1) $G_O(x, y) = G_O(y, x) \ \ \forall\, x, y \in [0, 1]$

(G2) $G_O(x, y) = 0$ *si y sólo si* $x \cdot y = 0$

(G3) $G_O(x, y) = 1$ *si y sólo si* $x \cdot y = 1$

(G4) G_O *es creciente*

(G5) G_O *es continua*

Las funciones de solapamiento son casos particulares de los operadores de agregación sin divisores por cero o divisores por uno. Algunos ejemplos de funciones de solapamiento son:

$G_O(x, y) = min(x, y)$
$G_O(x, y) = \sqrt{xy}$
$G_O(x, y) = min(x^k y, xy^k), k \in]0, 1[$

En [2] se presenta un teorema que proporciona tanto una caracterización como un método de construcción para funciones de solapamiento:

Teorema 1.1. *Una función* $G_O : [0,1]^2 \to [0,1]$ *es una función de solapamiento si y sólo si puede ser expresada como:*

$$G_O(x,y) = \frac{f(x,y)}{f(x,y) + h(x,y)} \tag{1.13}$$

para cualesquiera $f, h : [0,1]^2 \to [0,1]$ *tales que:*

(1) f *y* h *son simétricas;*

(2) f *es no decreciente y* h *es no creciente;*

(3) $f(x,y) = 0$ *si y sólo si* $min(x.y) = 0$*;*

(4) $h(x,y) = 0$ *si y sólo si* $min(x.y) = 1$*;*

(5) f *y* h *son continuas*

Ejemplo 1.4. *A continuación se presentan algunos ejemplos de construcción de funciones de solapamiento utilizando el teorema 1.1:*

(1) Si $f(x,y) = min(x,y)$ *y* $h(x,y) = max(1-x, 1-y)$ *entonces:*

$$G_O(x,y) = min(x,y) \tag{1.14}$$

es una función de solapamiento.

(2) Si $f(x,y) = \sqrt{x \times y}$ *y* $h(x,y) = max(1-x, 1-y)$ *entonces:*

$$G_O(x,y) = \frac{\sqrt{x \times y}}{\sqrt{x \times y} + max(1-x, 1-y)} \tag{1.15}$$

es una función de solapamiento.

(3) Si $f(x,y) = \sqrt{x \times y}$ *y* $h(x,y) = 1 - x \times y$ *entonces:*

$$G_O(x,y) = \frac{\sqrt{x \times y}}{\sqrt{x \times y} + 1 - x \times y} \tag{1.16}$$

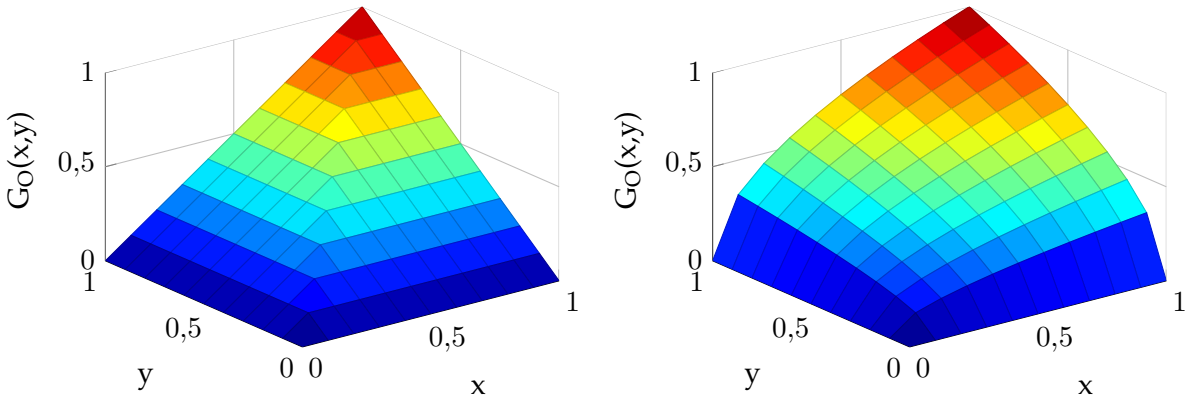
es una función de solapamiento.

En la figura 1.5 se pueden ver las representaciones gráficas de las funciones de solapamiento del ejemplo anterior.

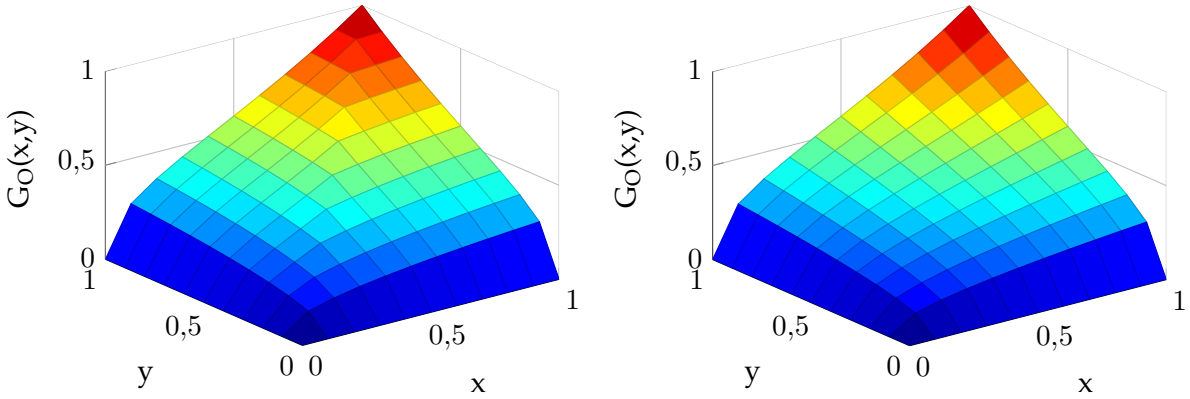
1.3.2. Caso particular: t-normas.

Por la definición 1.8 sabemos que una t-norma es una función conmutativa, asociativa, y creciente $T : [0,1]^2 \to [0,1]$ tal que $T(x,1) = x$ para todo $x \in [0,1]$. Bajo ciertas condiciones, las t-normas también cumplen la definición de función de solapamiento [2].

Teorema 1.2. *Si una t-norma* T *es una función de solapamiento, entonces* T *es uno de los siguientes 3 tipos:*



(a) $G_O(x,y) = min(x,y)$ (b) $G_O(x,y) = \sqrt{x \cdot y}$



(c) $G_O(x,y) = \frac{\sqrt{x \times y}}{\sqrt{x \times y} + max(1-x, 1-y)}$ (d) $G_O(x,y) = \frac{\sqrt{x \times y}}{\sqrt{x \times y} + 1 - x \times y}$

Figura 1.5: Representación gráfica de algunas funciones de solapamiento.

(1) $T = \text{mín}$

(2) T es estricta (continua y estrictamente monótona)

(3) T es la suma ordinal de la familia $\{([a_m, b_m], T_m)\}$*, siendo* T_m *todas las t-normas continuas y arquimedeanas tales que si para algún* m_0 *tenemos que* $a_{m_0} = 0$ *entonces* T_{m_0} *es necesariamente una t-norma estricta.*

Proposición 1.1. *Sea* G_O *una función de solapamiento tal que* $G_O(x, G_O(y,z)) = G_O(y, G_O(x,z))$*, entonces* T *es una t-norma [1].*

1.4. Índices de solapamiento

En esta sección se presenta la definición de índice de solapamiento y se estudian algunas de sus propiedades más importantes. Los índices de solapamiento tienen una importancia vital en el desarrollo de este trabajo, ya que son la base del método de inferencia difusa estudiado.

1.4.1. Definición y propiedades

El concepto de solapamiento aplicado a conjuntos difusos fue introducido por Zadeh en 1978 [15]. A grandes rasgos, un índice de solapamiento es una función que permite determinar cuánto se solapan dos conjuntos difusos, es decir, cómo de coincidentes son dichos conjuntos.

Definición 1.13. *Sea $A, B \in FS(U)$. La consistencia entre A y B se define como:*

$$O_Z(A, B) = sup_{i=1}^{n}(min(A(u_i), B(u_i))) \tag{1.17}$$

Para conjuntos de referencia finitos, el supremo es en realidad el máximo elemento del conjunto.

En 1982 Dubois y Prade [16] presentan la siguiente definición para el índice de solapamiento:

Definición 1.14. *Un índice de solapamiento es una función $O : FS(U) \times FS(U) \to [0, 1]$ tal que:*

(O1) $O(A, B) = 0$ *si y sólo si A y B son completamente disjuntos.*

(O2) $O(A, B) = 1$, *si* ($A(u_i) = 0$ *o* $B(u_i) = 0$) *o* ($A(u_i) = 1$ *o* $B(u_i) = 1$)

(O3) $O(A, B) = O(B, A)$

(O4) Si $B \leq C$, *entonces* $O(A, B) \leq O(A, C)$

La condición (O2) en esta definición presenta la ventaja de que, si A no es difuso, entonces $O(A, A) = 1$.

Por esta razón Dubois, Ostasiewicz y Prade imponen en [16] las siguientes condiciones:

(1) Para todos los conjuntos difusos tales que $A(u_i) \leq 1$ y $A(u_i) \leq 1$ para cualquier $u_i \in U$, (O2) debe ignorarse.

(2) El índice ROC ([16]) no satisface (O2).

Debido a estas consideraciones, normalmente sólo se imponen las condiciones (O1), (O3), (O4) de la definición 1.14 a los índices de solapamiento. En [1] se propone la siguiente definición de índice de solapamiento:

Definición 1.15. *Un índice de solapamiento es una función $O : FS(U) \times FS(U) \to [0, 1]$ tal que:*

(O1) $O(A, B) = 0$ *si y sólo si A y B tienen soportes disjuntos, es decir,* $A(u_i)B(u_i) = 0$ *para todo* $u_i \in U$

(O2) $O(A, B) = O(B, A)$

(O3) Si $B \leq C$, *entonces* $O(A, B) \leq O(A, C)$

Un índice de solapamiento normal es un índice O *tal que:*

(O4) Si existe un $u_i \in U$ *tal que* $A(u_i) = B(u_i) = 1$, *entonces* $O(A, B) = 1$

1.4.2. Construcción de índices de solapamiento

En esta sección se presenta el método de construcción de índices de solapamiento propuesto por [1] que se basa en operadores de agregación (sección 1.2) y funciones de solapamiento (sección 1.3).

Teorema 1.3. *Sea $M : [0,1]^2 \to [0,1]$ una función de agregación tal que $M(x_1, \cdots, x_n) = 0$ si y sólo si $x_1 = \cdots = x_n = 0$. Sea $G_O : [0,1]^2 \to [0,1]$ una función de solapamiento. Entonces, la función $O : F(U) \times F(U) \to [0,1]$ definida como:*

$$O(A,B) = M(G_O(A(u_1), B(u_1)), \cdots, G_O(A(u_n), B(u_n))) \tag{1.18}$$

es un índice de solapamiento en el sentido de la definición 1.15. Recíprocamente, si G_O es una función de solapamiento y $M : [0,1]^n \to [0,1]$ es un operador de agregación tal que O definido por la ecuación 1.18 es un índice de solapamiento, entonces $M(x_1, \cdots, x_n) = 0$ si y sólo si $x_1 = \cdots = x_n = 0$.

Ejemplo 1.5. Utilizando el método de construcción del teorema 1.18 se pueden construir los siguientes índices de solapamiento:

- Si $M =$ Media aritmética y $G_O = x \cdot y$:

$$O_\pi(A,B) = \frac{1}{n} \sum_{i=1}^{n} \mu_A(x_i) \cdot \mu_B(x_i) \tag{1.19}$$

- Si $M = \text{máx}(x_1, \cdots, x_n)$ y $G_O = \text{mín}(x,y)$:

$$O_Z(A,B) = \max_{i=1}^{n}(\text{mín}(\mu_A(x_i), \mu_B(x_i))) \tag{1.20}$$

- Si $M =$ Media aritmética y $G_O = \sin(\frac{\pi}{2}(x \cdot y)^{\frac{1}{4}})$:

$$O_{sin}(A,B) = \frac{1}{n} \sum_{i=1}^{n} \sin(\frac{\pi}{2}(\mu_A(x_i) \cdot \mu_B(x_i))^{\frac{1}{4}}) \tag{1.21}$$

2 Lógica difusa

En esta sección se introducen las bases de la lógica difusa y los mecanismos de inferencia difusa. Además se describen algunos de los métodos clásicos de inferencia difusa basada en reglas.

2.1. ¿Qué es la lógica difusa?

La lógica difusa fue introducida por Lofti A. Zadeh en 1965. Desde entonces se ha aplicado en multitud de escenarios tales como control de procesos industriales, medicina, electrónica y otros tipos de sistemas expertos. En general, la mayoría de aplicaciones de la lógica difusa son el área del control.

La motivación para el desarrollo de una lógica difusa fue expresada por Zadeh (1984, [17]) de la siguiente manera:

> *"La habilidad de la mente humana para razonar en términos difusos es realmente una gran ventaja. Aunque una gran cantidad de información es captada por los sentidos en una situación determinada, de alguna manera la mente humana es capaz de descartar la mayoría de esa información y concentrarse sólo en aquello que es relevante."*

El objetivo principal de la lógica difusa es intentar dotar a las máquinas de un sistema de razonamiento aproximado similar al de los humanos, que pueda lidiar con imprecisiones y términos inexactos. Para ello, la lógica difusa se basa en la utilización de variables lingüísticas expresadas en lenguaje natural, que forman la base del razonamiento aproximado.

Por medio del uso de variables lingüísticas se pueden modelar conceptos como *"muy alto"* o *"bastante caliente"* y establecer relaciones entre ellos, expresadas de forma matemática y algorítmica. Los sistemas de lógica difusa tratan la imprecisión de las entradas mediante el uso de variables lingüísticas (expresadas como conjuntos difusos), que transforman dichos valores de entrada en números difusos.

2.2. Razonamiento aproximado

El razonamiento aproximado utiliza conjuntos difusos y lógica difusa para modelar la forma de discurrir y razonar del ser humano [18]. El razonamiento aproximado no posee la precisión y exactitud de la lógica clásica, pero es más efectivo a la hora de lidiar con imprecisiones, conceptos vagos o sistemas complejos y/o pobremente definidos.

En este proyecto se va a implementar una forma de razonamiento aproximado que utiliza un mecanismo de inferencia difusa basado en el modus ponens generalizado, que es una extensión del modus ponens clásico.

2.2.1. Modus ponens clásico

El modus ponens clásico es una proposición compuesta muy conocida en la lógica clásica. Tiene la forma:

$$((p \wedge (p \rightarrow q)) \rightarrow q) \tag{2.1}$$

Para inferir el valor de verdad de q a partir del de p utilizando el modus ponens clásico, se utiliza una *regla* de inferencia, expresada de forma simbólica utilizando el *silogismo*:

$$\begin{array}{rl} \text{Premisa 1:} & p \\ \text{Premisa 2:} & p \rightarrow q \\ \hline \text{Conclusión:} & q \end{array} \tag{2.2}$$

que puede expresarse como: si la proposición p es verdadera (premisa 1) y la proposición $p \rightarrow q$ es verdadera (premisa 2), entonces q es verdadera (conclusión). Esto permite obtener el valor de verdad de q a partir del de p utilizando una regla de inferencia. Es decir, por medio de esta regla de inferencia sabemos que, si p es verdadero, entonces q es también verdadero.

La regla de inferencia 2.2 puede expresarse de manera más detallada como:

$$\begin{array}{rl} \text{Premisa 1:} & x \text{ es } A \\ \text{Premisa 2:} & \text{IF } x \text{ es } A \text{ THEN } y \text{ es } B \\ \hline \text{Conclusión:} & y \text{ es } B \end{array} \tag{2.3}$$

donde $p = x$ es A y $q = y$ es B.

Ejemplo 2.1. *Un ejemplo de la regla de inferencia 2.3 es la regla de divisibilidad por 3:* "Ùn número entero es divisible por 3 si la suma de sus dígitos es múltiplo de 3"*. Puede expresarse mediante el siguiente silogismo:*

$$\begin{array}{rl} \text{Premisa 1:} & \textit{La suma de los dígitos de un número es múltiplo de 3} \\ \text{Premisa 2:} & \text{IF } \textit{La suma de los dígitos de un número es múltiplo de 3} \text{ THEN } \textit{El número es divisible por 3} \\ \hline \text{Conclusión:} & \textit{El número es divisible por 3} \end{array} \tag{2.4}$$

En este ejemplo se utilizan conceptos exactos (lógica clásica). Un número pertenece o no pertence al conjunto de números cuya suma de cifras es múltiplo de 3. Por lo tanto la proposición "La suma de los dígitos de un número es múltiplo de 3" *sólo puede tomar los valores* Verdadero *(1) o* Falso *(0).*

2.2.2. Modus ponens generalizado

El *modus ponens generalizado* es, como su nombre índica, una generalización del modus ponens clásico (definido en la sección anterior) utilizado en la lógica difusa. Tiene la forma:

$$\begin{array}{rl} \text{Premisa 1:} & p' \\ \text{Premisa 2:} & p \rightarrow q \\ \hline \text{Conclusión:} & q' \end{array} \tag{2.5}$$

o lo que es lo mismo:

$$\begin{array}{rl} \text{Premisa 1:} & x \text{ es } A' \\ \text{Premisa 2:} & \text{IF } x \text{ es } A \text{ THEN } y \text{ es } B \\ \hline \text{Conclusión:} & y \text{ es } B' \end{array} \tag{2.6}$$

En este caso las proposiciones $p' = x$ es A', $p = x$ es A, $q = y$ es B, $q' = y$ es B' vienen caracterizadas por los conjuntos difusos A',A,B y B',que representan conceptos difusos. Esta regla de inferencia puede interpretarse de la siguiente manera: si $p \rightarrow q$ y tenemos p' (aproximadamente p) entonces tenemos q' (aproximadamente q). El modus ponens generalizado permite, por tanto, utilizar conceptos difusos y obtener el "grado de verdad" de la conclusión a partir de las premisas.

En la siguiente sección se define el concepto de *regla difusa* (p.e. IF x es A THEN y es B), que constituye una de las premisas en el modus ponens generalizado. En las secciones 2.5 y 2.6 se definen los métodos de inferencia difusa que se van a estudiar en este proyecto. Estos métodos utilizan reglas difusas y el modus ponens generalizado para obtener las salidas (B') a partir de las premisas (A').

2.3. Reglas difusas

Una *regla difusa IF-THEN*[1] puede definirse de manera informal como:

Definición 2.1. *Una regla difusa IF-THEN es una estructura condicional de la forma:*

$$\textit{IF [antecedentes] THEN [consecuentes]} \tag{2.7}$$

, donde [antecedentes] y [consecuentes] son proposiciones difusas.

Las proposiciones difusas pueden ser *simples* o *compuestas*. Una proposición difusa simple es un predicado único formado por una variable lingüística *x* y un valor *A* de dicha variable (un conjunto difuso). Es decir, *A* es un conjunto difuso (valor lingüístico) definido sobre el mismo universo de referencia *U* de *x*. Por ejemplo si la variable *x* es la *"Temperatura medida en ºC"* del ejemplo 1.3, entonces las siguientes son proposiciones difusas sobre *x*:

[1] Podría traducirse como SI-ENTONCES.

$$x \text{ es B} \tag{2.8}$$

$$x \text{ es M} \tag{2.9}$$

$$x \text{ es A} \tag{2.10}$$

donde *B*, *M* y *A* son los valores lingüísticos "baja", "media" y "alta" respectivamente. Estos valores lingüísticos vienen definidos por conjuntos difusos con funciones de pertenencia μ_{Baja}, μ_{Media} y μ_{Alta}.

Las proposiciones difusas simples se pueden combinar para formar predicados más complejos. Para ello se utilizan normalmente los conectivos *AND* (intersección difusa) y *OR* (unión difusa). Por ejemplo, se puede definir la condición *"Temperatura es Alta Ó Temperatura es Media"* con la siguiente proposición difusa:

$$x \text{ es } A \text{ OR } x \text{ es } M \tag{2.11}$$

donde A y M son los valores lingüísticos *"alta"* (μ_{Alta}) y *"media"* (μ_{Media}) respectivamente. En la misma proposición difusa se pueden utilizar variables diferentes (generalmente es así) definidas normalmente sobre universos de referencia diferentes. El consecuente generalmente suele ser una proposición difusa simple formado por una variable *y* sobre *V*. Por ello, se puede definir el concepto de *regla difusa IF-THEN* de manera más formal como:

Definición 2.2. *Una* regla difusa IF-THEN *es una estructura condicional de la forma:*

$$\textit{IF } x_1 \textit{ es } A_1 \odot x_2 \textit{ es } A_2 \odot \ldots \odot x_n \textit{ es } A_n \textit{ THEN } y \textit{ es } B \tag{2.12}$$

donde $x_1 \in U_i, \ldots, x_n \in U_n$,$y \in V$ son variables lingüísticas y $\odot$ algún conectivo difuso (AND,OR,AND NOT,etc.).

2.4. Sistemas difusos basados en reglas

Un enfoque muy utilizado a la hora de modelar un sistema de lógica difusa es la utilización de *reglas* expresadas en un lenguaje muy próximo al natural, mediante variables lingüísticas. Estas reglas habitualmente son formuladas por un experto humano aunque pueden ser derivadas también de datos numéricos. A este tipo de sistemas difusos se les denomina *sistemas difusos basados en reglas*. Los métodos estudiados en este trabajo pertenecen a este tipo de sistemas.

Existen principalmente 3 tipos de sistemas difusos [6]: sistemas difusos puros, sistemas difusos tipo Takagi-Sugeno-Kang (TKG) y sistemas difusos con fusificador y defusificador. A continuación se realiza una breve descripción de las características de estos sistemas.

En la figura 2.1 se representa el esquema básico de un sistema difuso puro. En los sistemas difusos puros tanto las entradas como la salida del sistema son conjuntos difusos. El sistema difuso está dotado de un *conjunto de reglas difusas IF-THEN*, que forman la base de conocimiento sobre el dominio del problema. El *sistema de inferencia difusa* combina estas reglas *IF-THEN*

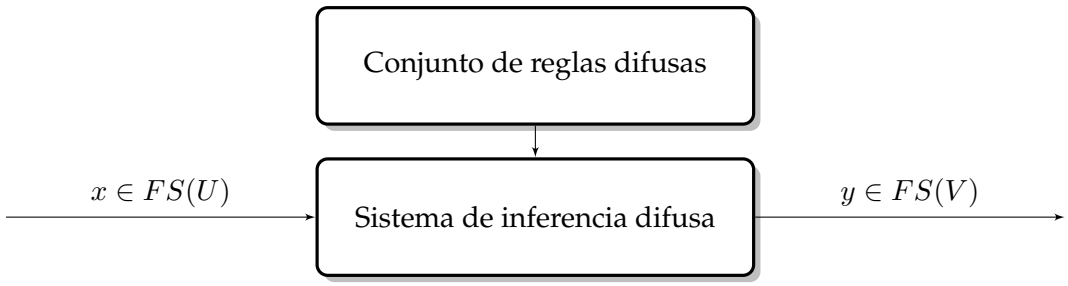


Figura 2.1: Diagrama de un sistema difuso puro.

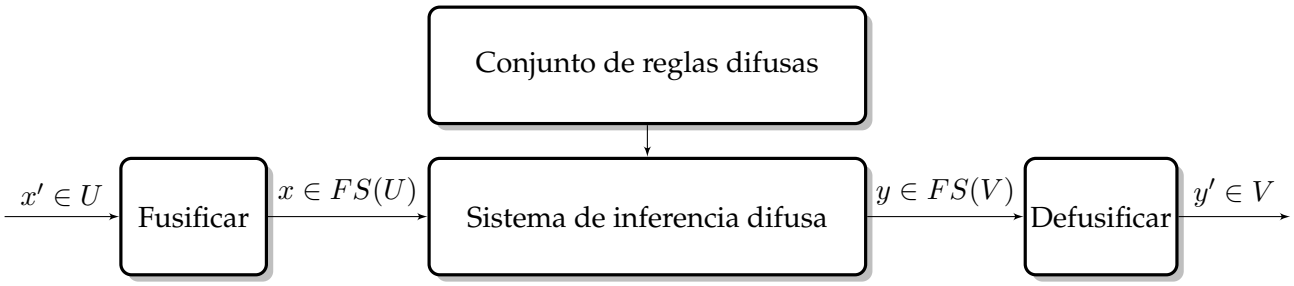


Figura 2.2: Diagrama de un sistema difuso con fusificador y defusificador.

y transforma los conjuntos difusos de entrada sobre el universo $U \subset R^n$ $(x_1 \ldots, x_n)$ en un conjunto difuso sobre $V \subset R$ (y), utilizando principios de lógica difusa (modus ponens).

El principal problema de los sistemas difusos puros es que tanto las entradas como las salidas son conjuntos difusos, mientras que en aplicaciones de control industrial se utilizan generalmente valores escalares reales. Para solucionar este problema, Takagi, Sugeno y Kang propusieron un nuevo método simplificado que utiliza entradas y salidas escalares [19][20].

El método de Takagi-Sugeno-Kang utiliza reglas de la forma:

$$R_i : \text{IF } x_1 \text{ es } A_{1i}, x_2 \text{ es } A_{2i} \text{ , } \ldots, x_n \text{ es } A_{ni} \text{ THEN } y = f_i(x_1, x_2, \ldots, x_n) \tag{2.13}$$

donde A_{ij} representa las funciones de pertenencia asociadas a las variables lingüísticas utilizadas en los antecedentes y $f_i(x_1, x_2, \ldots, x_n)$ representa los consecuentes. Normalmente f_i es un polinomio en las variables de entrada, pero puede ser cualquier tipo de función mientras pueda describir la salida del modelo de forma apropiada, teniendo en cuenta las entradas.

Como se puede ver, la principal diferencia entre un sistema difuso puro y el sistema de Takagi-Sugeno-Kang es que el consecuente de la regla (*THEN*) es sustituido por una simple fórmula matemática. Este cambio supone que el método para combinar las diferentes reglas para proporcionar una salida es más sencillo. De hecho, en el método de Takagi-Sugeno-Kang se utiliza la media ponderada de los consecuentes obtenidos de las diferentes reglas.

El principal problema del método de Takagi-Sugeno-Kang es que el consecuente de las reglas es una función matemática y por lo tanto se pierde cierta capacidad de modelar el sistema utilizando el lenguaje natural. Para solucionar estos problemas, se puede modificar el sistema difuso puro, añadiendo un fusificador a la entrada de las variables y un defusificador a la salida.

En la figura 2.2 se representa el esquema básico de un sistema difuso con fusificador y defusificador. El fusificador transforma las entradas escalares en U del sistema en conjuntos difusos. Una vez realizada la inferencia difusa se obtiene un conjunto difuso como salida. El defusificador transforma el conjunto difuso de salida en una variable escalar en V. Existen multitud de métodos para realizar estas transformaciones y en siguientes secciones se detallarán algunos de ellos. El sistema difuso presentado en este trabajo se basa en este último tipo de sistema.

2.4.1. Fusificadores

En la sección 2.4 se ha definido un tipo de sistema difuso que utiliza un concepto llamado *fusificador* para transformar las entradas escalares del sistema en conjuntos difusos. Un *fusificador* puede definirse formalmente como:

Definición 2.3. *Un* fusificador *es una función $F : \mathbb{R} \to FS(U)$ que toma un valor escalar real x^* en el universo U, $x^* \in U \subset \mathbb{R}$, y lo transforma en un conjunto difuso A' sobre U.*

Para que el conjunto de salida represente de manera lo más fiel posible al valor de entrada, es necesario imponer algunas restricciones a la hora de construir el fusificador. La principal es que la función de pertenencia de A' debe tener un valor alto en el punto x^* (generalmente máximo, $\mu_{A'}(x^*) = \max(\mu_{A'}(\mu_i)), \forall \mu_i \in U$). Además, si se considera que existe ruido o imprecisión en los valores entrada (por ejemplo imprecisiones provocadas por instrumentos de medida) se puede utilizar una función de pertenencia cuyo valor sea alto en los puntos próximos a x^* y bajo (o incluso cero) en los más alejados.

Algunos de los fusificadores más utilizados son los siguientes [6]:

- **Fusificador singleton:** El *fusificador singleton* (fig. 2.3a) transforma un valor escalar real $x^* \in U$ en un conjunto difuso A' en U, de forma que la función de pertenencia de A' tiene valor 1 en el punto x^* y 0 en el resto de puntos de U:

$$\mu_{A'}(x) = \begin{cases} 1 & \text{si x} = x^*, \\ 0 & \text{en cualquier otro caso} \end{cases} \tag{2.14}$$

  Este tipo de fusificador tiene la ventaja de ser simple y facilitar los cálculos realizados en el motor de inferencia. Sin embargo, tiene la desventaja de que no puede lidiar con ruido o imprecisiones en los valores de entrada.

- **Fusificador gaussiano:** El *fusificador gaussiano* (fig. 2.3b) transforma un valor escalar real $x^* \in U$ en un conjunto difuso A' en U, utilizando una función de pertenencia Gaussiana:

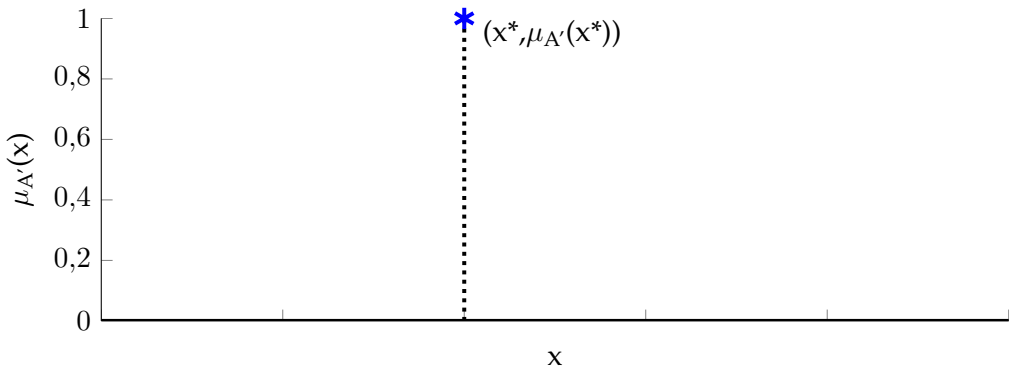
$$\mu_{A'}(x) = e^{-(\frac{x_1 - x_1^*}{a_1})^2} \star \ldots \star e^{-(\frac{x_n - x_n^*}{a_n})^2} \tag{2.15}$$

  donde a_i son parámetros positivos y $\star$ es una t-norma, generalmente el producto algebraico o el mínimo. Este tipo de fusificador es adecuado cuando existe ruido o imprecisiones en los datos de entrada.

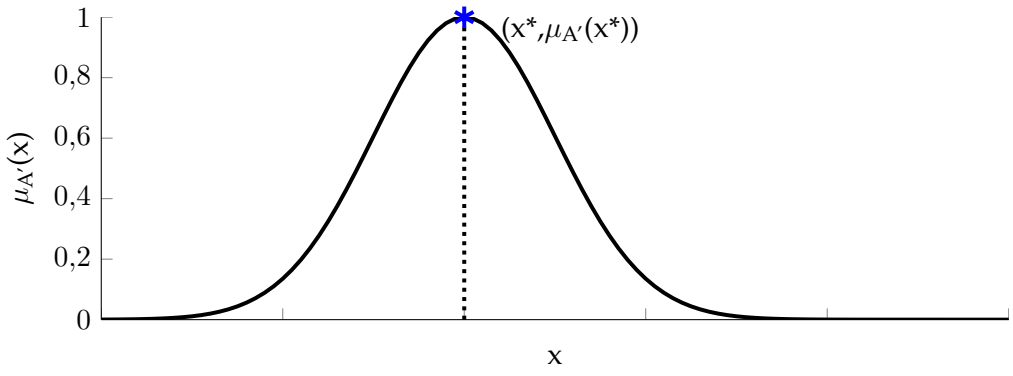
- **Fusificador triangular:** El *fusificador triangular* (fig. 2.3c) transforma un valor escalar real $x^* \in U$ en un conjunto difuso A' en U, utilizando una función de pertenencia triangular:

$$\mu_{A'}(x) = \begin{cases} (1 - \frac{|x_1 - x_1^*|}{b_1}) \star \ldots \star (1 - \frac{|x_n - x_n^*|}{b_n}) & \text{si } |x_i - x_i^*| \leq b_i, i = 1, 2, \ldots, n, \\ 0 & \text{en cualquier otro caso} \end{cases} \tag{2.16}$$

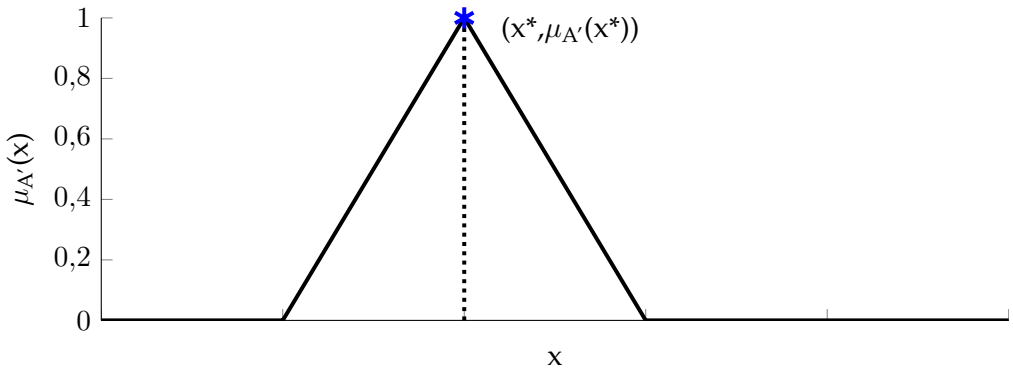
donde b_i son parámetros positivos y $\star$ es una t-norma, generalmente el producto algebraico o el mínimo. Al igual que el fusificador gaussiano este tipo de fusificador es adecuado cuando existe ruido o imprecisiones en los datos de entrada.



(a) Fusificador singleton.



(b) Fusificador gaussiano.



(c) Fusificador triangular.

Figura 2.3: Representación de las funciones de pertenencia obtenidas con los fusificadores *singleton* (2.3a), *gaussiano* (2.3b) y *triangular* (2.3c).

2.4.2. Defusificadores

El último paso en el sistema difuso definido en 2.4 es la *defusificación*, es decir, transformar el conjunto difuso, obtenido al aplicar las reglas de inferencia, en un valor escalar. Puede decirse que un defusificador realiza la operación complementaria a un fusificador (descritos en la sección 2.4.1). Así pues, un defusificador puede definirse como [6]:

Definición 2.4. *Un* defusificador *es una función que transforma un conjunto difuso B' en $V \subset \mathbb{R}$ (la salida del sistema de inferencia difusa) en un valor escalar $y^* \in V$.*

La misión de un defusificador es obtener el punto y^* en V que mejor representa al conjunto difuso B'. En este sentido, los defusificadores cumplen un rol similar a los operadores de agregación (por ejemplo la media aritmética). Se pueden utilizar diversas funciones como defusificadores, aunque generalmente se eligen aquellas que satisfacen las siguientes condiciones [21]:

- **Continuidad:** Un cambio pequeño en la entrada del sistema difuso no debería causar una gran variación en la salida.
- **Desambiguación:** El método de defusificación debe devolver un único valor para y^*. Es decir, no debe existir ambigüedad a la hora de seleccionar el valor para y^*.
- **Plausibilidad:** El punto y^* debe representar al conjunto difuso B' de forma intuitiva. Es decir, el punto y^* debe estar aproximadamente en el centro del soporte de B' y tener un grado de pertenencia alto en B'.
- **Simplicidad computacional:** La función debe ser lo más sencilla posible de calcular. Esto es especialmente importante en controladores difusos que operan en tiempo real.
- **Método de ponderado:** El método seleccionado debe ponderar los diferentes conjuntos de salida. Este criterio depende del ámbito del problema.

A continuación se presentan algunos de los métodos de defusificación más utilizados:

- **Centroide:** También conocido como *centro de gravedad*, el método del *centroide* fue propuesto por Sugeno en 1985 y es uno de los métodos más utilizados [22]. Este método obtiene el valor y^* como el centro del área cubierta por la función de pertenencia de B'. El método del centro de gravedad puede definirse como:

$$y^* = \frac{\sum y \mu_{B'}(y)}{\sum \mu_{B'}(y)} \tag{2.17}$$

  Este método tiene la ventaja de que proporciona un valor plausible de forma intuitiva.

- **Bisector:** El método del bisector obtiene el punto por el que pasa la línea que divide la región delimitada por la función de pertenencia de B' y el eje de abscisas, en dos subsecciones de igual área:

$$y^* \text{ tal que: } \sum_{y=y_0}^{y=y^*} \mu_{B'}(y) = \sum_{y=y^*}^{y=y_n} \mu_{B'}(y) \tag{2.18}$$

El valor obtenido con este método coincide en ocasiones con el método del *centroide* y también es computacionalmente costoso.

- **Máximo:** El defusificador del máximo obtiene el punto y^* en el que la función de pertenencia $\mu_{B'}(y)$ alcanza su máximo valor. Es decir:

$$y^* \text{ tal que: } \mu_{B'}(y^*) = \max(\mu_{B'}(y_i)) \qquad , \forall y_i \in V \tag{2.19}$$

Este método tiene la ventaja de ser fácil de implementar y poco costoso de calcular. Sin embargo, puede producir resultados ambiguos, puesto que pueden existir varios puntos en los que $\mu_{B'}(y)$ alcanza un valor máximo. En el caso de que existan varios puntos con valores máximos es necesario imponer algún criterio para seleccionar uno de ellos. Si se construye el conjunto:

$$hgt(B') = \{y \in V | \mu_{B'}(y) = \sup_{y \in V} \mu_{B'}(y)\} \tag{2.20}$$

es decir, el conjunto de todos los puntos en V donde $\mu_{B'}(y)$ alcanza su valor máximo. El defusificador del máximo obtiene un elemento arbitrario y^* del conjunto $hgt(B')$, es decir:

$$y^* = \text{ cualquier punto en } hgt(B') \tag{2.21}$$

- **Menor de máximos (SOM, *smallest of maximum*):** Dado el conjunto $hgt(B')$ con todos los puntos donde $\mu_{B'}(y)$ alcanza su valor máximo, el defusificador *menor de máximos* obtiene el menor punto y^* del conjunto tal que:

$$y^* = \text{ínf}\{y \in hgt(B')\} \tag{2.22}$$

- **Mayor de máximos (LOM, *largest of maximum*):** Dado el conjunto $hgt(B')$ con todos los puntos donde $\mu_{B'}(y)$ alcanza su valor máximo, el defusificador *mayor de máximos* obtiene el mayor punto y^* del conjunto tal que:

$$y^* = \sup\{y \in hgt(B')\} \tag{2.23}$$

- **Media de máximos (MOM, *mean of maximum*):** Dado el conjunto $hgt(B')$ con todos los puntos donde $\mu_{B'}(y)$ alcanza su valor máximo, el defusificador *media de máximos* obtiene el punto medio y^* del conjunto tal que:

$$y^* = \frac{\text{ínf}\{y \in hgt(B')\} + \sup\{y \in hgt(B')\}}{2} \tag{2.24}$$

En la figura 2.4 se presenta una función de pertenencia y los resultados de aplicar los defusificadores definidos anteriormente. En este caso, los resultados obtenidos con el método del *centroide* y el *bisector* coinciden, aunque no siempre tiene por qué ser así.

2.5. El controlador de Mamdani

Uno de los métodos de inferencia difusa más utilizado es el llamado *controlador de Mamdani* [4], propuesto por Mamdani y Assilian en 1975 para realizar el control de un motor de vapor a partir de un conjunto de reglas obtenidas de operadores humanos experimentados.

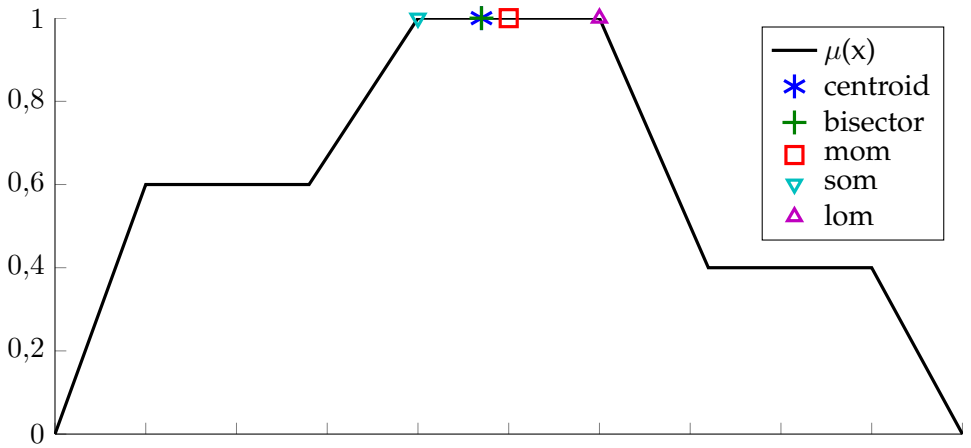


Figura 2.4: Resultado de aplicar los defusificadores *centroide* (∗), *bisector*(+), *media de máximos* (□), *menor de máximos*(▽) y *mayor de máximos*(△).

El método de Mamdani utiliza reglas difusas de la forma:

$$R_i : \text{IF } \chi_1 \text{ es } A_{i1}, \chi_2 \text{ es } A_{i2}, \ldots, \chi_n \text{ es } A_{im} \text{ THEN } y \text{ es } B_i \tag{2.25}$$

donde $\chi_1, \ldots, \chi_m$ son variables lingüísticas cuyos valores son conjuntos difusos $A_{ij} \subset FS(U_j)$ con $j \in \{1, \ldots, m\}$. El consecuente y es también una variable lingüística sobre Y ($B_i \subset FS(Y)$). Las entradas del sistema son valores escalares $x_1 \in U_1, \ldots, x_m \in U_m$.

Cada regla se evalúa por separado y se obtiene el grado de pertenencia de cada entrada al conjunto difuso de su correspondiente antecedente, utilizando el operador mínimo para realizar la conjunción AND de los antecedentes ($k_i = \text{mín}(\mu_{Ai1}(x_1), \ldots, \mu_{Aim}(x_m))$). El resultado de evaluar cada regla es un conjunto difuso sobre Y (B'_i), que se calcula "truncando" el consecuente con el valor k_i ($B'_i = \{(y, \text{mín}(B_i(y), k_i)) | y \in Y\}$). En el caso de múltiples reglas, el conjunto difuso de salida se calcula combinando los conjuntos difusos obtenidos en cada regla utilizando el máximo ($B'(y) = \max_{i=1}^{n}(B'_i)$).

Algoritmo 1: Método de Mamdani

Input: Un conjunto de reglas R_i ($i \in \{1, \ldots, n\}$) con varios antecedentes A_{ij} ($j \in \{1, \ldots, m\}$) y entradas escalares $x_1, \ldots, x_m$.

Output: B'.

1 **for** $i \in \{1, \ldots, n\}$ **do**
2 Calcular $k_i = \text{mín}(\mu_{Ai1}(x_1), \ldots, \mu_{Aim}(x_m))$
3 Calcular $B'_i = \{(y, \text{mín}(B_i(y), k_i)) | y \in Y\}$
4 **end**
5 Construir $B' = \{(y, B'(y)) | y \in Y\}$ dado por:

$$B'(y) = \max_{i=1}^{n}(B'_i).$$

6 **return** B'

La principal ventaja del método de Mamdani es que es relativamente sencillo de implementar y computacionalmente eficiente. El método de Larsen es similar, pero utilizando el producto en vez del mínimo como operador de implicación [23].

2.6. Método de interpolación basado en índices de solapamiento

En esta sección se introduce un nuevo método para resolver sistemas basados en reglas que generaliza el método clásico de interpolación , utilizando para ello índices de solapamiento [1].

Históricamente el método más utilizado para resolver sistemas basados en reglas, es decir, para calcular el consecuente B', era el *método de interpolación*, desarrollado por Kóczy en 1993 [24]. En este método se utiliza la consistencia de Zadeh O_Z [15], dada en la ecuación 1.17. Los pasos a seguir son:

Algoritmo 2: Método de interpolación

Input: Un conjunto de reglas R_j, con $j \in \{1, \ldots, n\}$, un hecho A' y el índice de consistencia O_Z (ec. 1.17).
Output: B'.

1 **for** $j \in \{1, \ldots, n\}$ **do**
2 | Calcular $O_Z(A', A_j) = \underset{x \in X}{\text{máx}}(\text{mín}(A'(x), A_j(x)))$
3 **end**
4 Construir $B' = \{(y, B'(y)) | y \in Y\}$ dado por:

$$B'(y) = \overset{n}{\underset{j=1}{\text{máx}}}(\text{mín}(B_j(y), O_Z(A', A_j))).$$

5 **return** B'

Dado que el índice de consistencia O_Z es un índice de solapamiento que cumple la definición 1.14, el algoritmo 2 se puede generalizar para utilizar cualquier índice de solapamiento:

Algoritmo 3: Método de interpolación generalizado

Input: Un conjunto de reglas R_j, con $j \in \{1, \ldots, n\}$ y un hecho A'.
Output: B'.

1 Seleccionar un operador de agregación M_1, una función de solapamiento G_O y un índice de solapamiento O.
2 **for** $j \in \{1, \ldots, n\}$ **do**
3 | Calcular $O(A', A_j)$
4 **end**
5 Construir $B' = \{(y, B'(y)) | y \in Y\}$ dado por:

$$B'(y) = \overset{n}{\underset{j=1}{M_1}}(G_O(B_j(y), O(A', A_j))).$$

6 **return** B'

Si en el algoritmo 3 utilizamos $M_1 = \text{máx}$, $G_O = \text{mín}$ y $O = O_Z$ entonces se recupera el algoritmo 2.

En el caso de que cada regla tenga varios antecedentes, se puede generalizar el algoritmo 3 de la siguiente manera:

Como en el algoritmo anterior, se toma un conjunto de reglas, en este caso con varios antecedentes cada una, y un hecho A'. Se debe seleccionar también un operador de agregación M, una t-norma T y un índice de solapamiento O. En caso de que las reglas tengan más de un antecedente, el hecho A' debe tener un valor para cada uno de ellos $(A'_1, \cdots, A'_n)$.

Algoritmo 4: Método de interpolación generalizado para reglas con varios antecedentes

Input: Un conjunto de reglas R_j con varios antecedentes, con $j \in \{1, \ldots, n\}$ y un hecho A'.
Output: B'.

1 Seleccionar un operador de agregación M, una t-norma T y un índice de solapamiento O.
2 **for** $i = 1 \to n$ **do**
3 | Calcular $O(A'_1, A_{i1}), \ldots, O(A'_m, A_{im})$
4 | Calcular $k_i = T(O(A'_1, A_{i1}), \ldots, O(A'_m, A_{im}))$
5 | Construir sobre el universo de referencia Y el conjunto $K_i = \{(y, k_i) | y \in Y\}$
6 **end**
7 Construir $B' = \{(y, B'(y)) | y \in Y\}$ dado por:

$$B'(y) = \underset{i=1}{\overset{n}{M}}(\text{mín}(K_i, B_i)).$$

8 **return** B'

Cada regla se evalúa por separado y para cada una se calculan los índices de solapamiento de cada valor del hecho A' con su correspondiente antecedente en la regla (línea 3). Estos índices de solapamiento se agregan utilizando la t-norma T, que modela la conjunción AND (línea 4). El valor k_i se utiliza para construir sobre el universo de salida Y el conjunto difuso K_i, tal que todos los elementos $y \in Y$ tienen como grado de pertenencia dicho valor k_i (línea 5).

Por último, para cada regla se calcula el conjunto de salida como el mínimo entre su correspondiente conjunto K_i (calculado en pasos previos) y el conjunto difuso de salida de la regla B_i. Los conjuntos de salida obtenidos para cada regla se agregan utilizando el operador de agregación M para obtener el conjunto de salida B' (línea 7). El conjunto B' es el resultado final de aplicar el método de inferencia a la entrada A'.

3 Detección de incendios forestales

En este capítulo se presenta una aplicación práctica de los métodos de inferencia difusa basados en reglas, descritos en los capítulos anteriores. El objetivo es definir un sistema basado en reglas capaz de determinar el riesgo de incendios forestales a partir de mediciones de algunas magnitudes tales como la temperatura, el humo, la humedad etc. Estas magnitudes constituirán las entradas del sistema de inferencia, que las utilizará para dar una estimación cuantitativa del riesgo del incendio forestal.

3.1. Red de sensores inalámbricos

El primer paso para la determinación del riesgo de un incendio forestal es la medición y toma de datos ambientales relacionados con dicho incendio. Para ello se puede desplegar una red de sensores inalámbricos (*Wireless Sensor Networks*, WSN), que han sido desarrolladas y utilizadas en una gran variedad de aplicaciones en áreas tales como automoción [25], defensa, medicina, agricultura [26][27], etc.

Algunas aplicaciones de este tipo de sistemas relacionadas con los riesgos ambientales incluyen, por ejemplo, el control de movimientos de personas y ganado, monitorización de factores ambientales que afectan a la calidad de los cultivos, detección de incendios forestales, mediciones meteorológicas y detección de inundaciones etc. [28]

Las redes de sensores inalámbricos están diseñadas para monitorizar y controlar eventos en lugares que presentan riesgos ambientales tales como bosques, terrenos montañosos etc. Por esta razón se diseña este tipo de sistemas de forma que sean lo más autónomos posible. Esto incluye, por ejemplo, la utilización de energías renovables (típicamente energía solar), que posibilitan que los sensores desplegados en el terreno sean totalmente auto-suficientes.

El diseño y despliegue de una red de sensores inalámbricos queda totalmente fuera del alcance de este proyecto y por tanto no se va a entrar en ningún tipo de detalle técnico sobre estos sistemas. El objetivo de este proyecto es evaluar algoritmos de inferencia difusa sobre los valores que serían entregados al sistema de decisión en una aplicación real .

Las mediciones realizadas por la red de sensores constituyen las entradas del sistema de inferencia y decisión. Estas entradas son, originalmente, valores escalares que son transformados por el sistema de lógica difusa en valores difusos (por medio de variables lingüísticas). Esto permite realizar, sobre estas mediciones, un proceso de razonamiento aproximado que puede tratar mejor las imprecisiones, en comparación con utilizar dichos valores escalares directamente.

3.2. Magnitudes medidas y variables lingüísticas

Las entradas del sistema difuso son valores escalares de magnitudes que describen el incendio forestal, tales como la temperatura o la luminosidad, medidas por la red de sensores. Para cada una de estas magnitudes se define una variable lingüística con tres valores posibles (definidas por sus correspondientes funciones de pertenencia). Así pues, los valores escalares de entrada serán transformados en conjuntos difusos que posteriormente serán comparados con las variables lingüísticas según las reglas definidas, para obtener como salida el riesgo de incendio (que es también una variable lingüística). Estas variables lingüísticas así como sus correspondientes universos (rangos de valores posibles) son:

(χ_1) **Temperatura:** medida en grados centígrados (0ºC a 120ºC).

(χ_2) **Humo:** medida en partes por millón (0 a 100ppm).

(χ_3) **Luz:** medida en lux (0 a 1000 lux).

(χ_4) **Humedad:** medida en partes por millón (0 a 100ppm).

(χ_5) **Distancia:** medida en metros (0 a 80m).

Para cada una de estas variables lingüísticas se definen tres valores posibles: *Baja* (L,*Low*), *Media* (M, *Medium*) y *Alta* (H, *High*). En el caso de la distancia, estos valores tienen el sentido de *Cerca* (C, *Close*), *Media* (M, *Medium*) y *Lejos* (F, *Far*) respectivamente.

La salida del sistema viene dada por la variable lingüística $y =$ *"Riesgo de incendio"*, que determina el riesgo de incendio forestal en una escala del 0 al 100 (%). Esta variable lingüística puede tomar los valores: *Muy bajo* (VL, *Very Low*), *Bajo* (L, *Low*), *Medio* (M, *Medium*), *Alto* (H, *High*) y *Muy Alto* (VH, *Very High*).

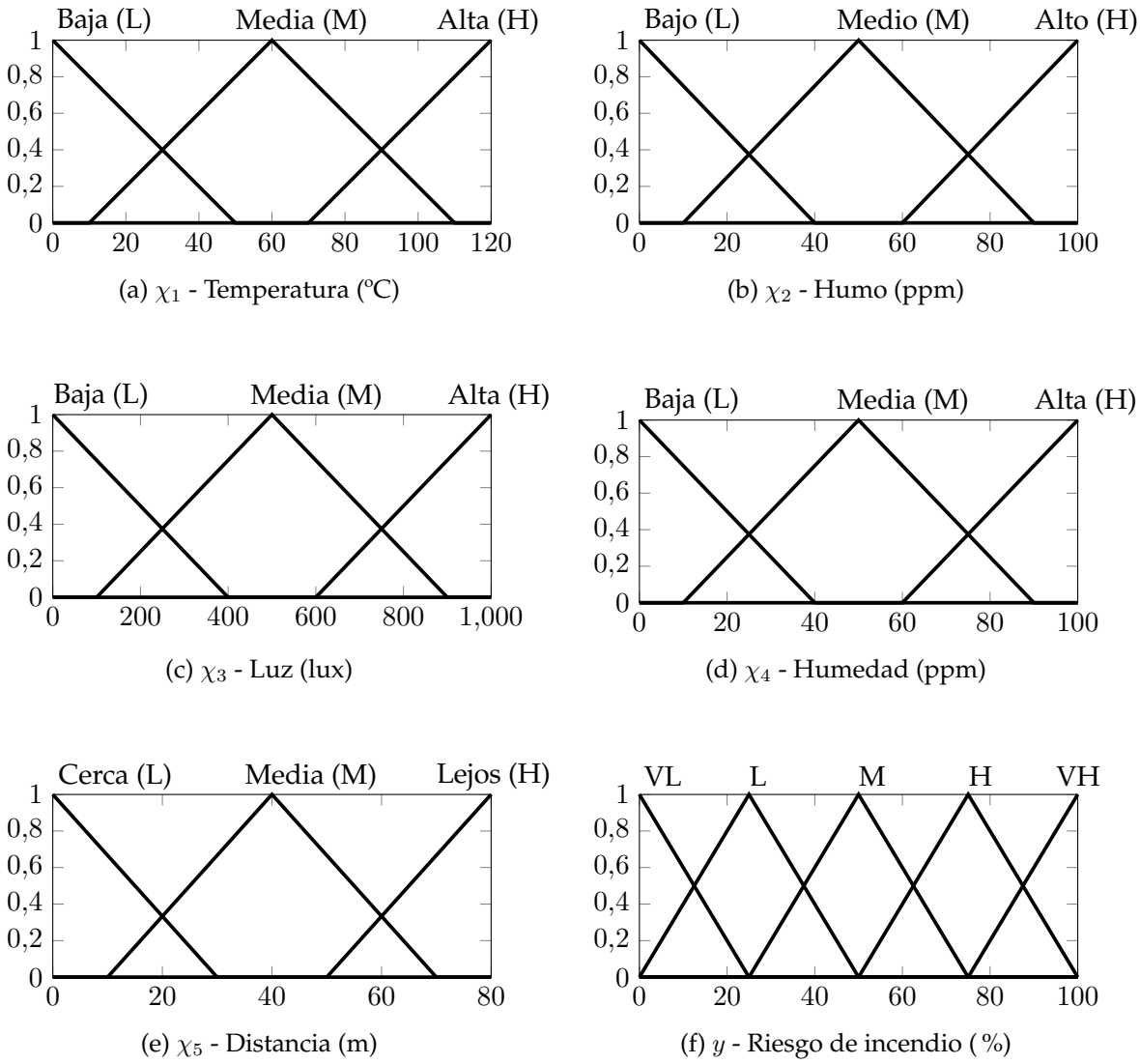
En la figura 3.1 se representan las variables lingüística descritas anteriormente. Como se puede ver, se han elegido funciones de pertenencia lineales para modelar los valores de las variables lingüísticas.

3.3. Conjunto de reglas

Una vez definidas las variables lingüísticas, es necesario construir un conjunto de reglas, que forman la base de conocimiento sobre el problema. Este conjunto de reglas permite, mediante el mecanismo de inferencia, transformar las entradas en salidas. En este caso particular las reglas tienen la forma:

$$\text{IF } \chi_1 \text{ es } A_1 \text{ AND } \chi_2 \text{ es } A_2 \text{ AND } \chi_3 \text{ es } A_3 \text{ AND } \chi_4 \text{ es } A_4 \text{ AND } \chi_5 \text{ es } A_5 \text{ THEN } y \text{ es } B \quad (3.1)$$

Donde $\chi_1, \ldots, \chi_n$ son las variables de entrada del sistema difuso (definidas en la sección anterior) e y es la salida (riesgo de incendio). Todas las reglas definidas en este sistema harán uso de las 5 variables lingüísticas conectadas mediante el operador de conjunción (AND).



(a) χ_1 - Temperatura (ºC)

(b) χ_2 - Humo (ppm)

(c) χ_3 - Luz (lux)

(d) χ_4 - Humedad (ppm)

(e) χ_5 - Distancia (m)

(f) y - Riesgo de incendio (%)

Figura 3.1: Variables lingüísticas utilizadas en la determinación de riesgo de incendios.

$\chi_1 =$ Temp.	$\chi_2 =$ Humo	$\chi_3 =$ Luz	$\chi_4 =$ Humedad	$\chi_5 =$ Distancia	$y =$ Riesgo
L	L	L	H	H	VL
L	L	L	H	M	VL
L	L	L	H	L	VL
L	L	L	M	H	VL
L	L	L	M	M	VL
L	L	L	M	L	L
L	L	L	L	H	VL
L	L	L	L	M	L
L	L	L	L	L	L
L	L	M	H	H	VL

L	L	M	H	M	L
L	L	M	H	L	L
L	L	M	M	H	VL
L	L	M	M	M	L
L	L	M	M	L	L
L	L	M	L	H	VL
L	L	M	L	M	L
L	L	M	L	L	L
L	L	H	H	H	L
L	L	H	H	M	L
L	L	H	H	L	L
L	L	H	M	H	L
L	L	H	M	M	L
L	L	H	M	L	M
L	L	H	L	H	L
L	L	H	L	M	M
L	L	H	L	L	M
L	M	L	H	H	VL
L	M	L	H	M	VL
L	M	L	H	L	L
L	M	L	M	H	VL
L	M	L	M	M	L
L	M	L	M	L	L
L	M	L	L	H	L
L	M	L	L	M	L
L	M	L	L	L	M
L	M	M	H	H	L
L	M	M	H	M	L
L	M	M	H	L	M
L	M	M	M	H	L
L	M	M	M	M	L
L	M	M	M	L	M
L	M	M	L	H	L

L	M	M	L	M	L
L	M	M	L	L	M
L	M	H	H	H	L
L	M	H	H	M	M
L	M	H	H	L	M
L	M	H	M	H	L
L	M	H	M	M	M
L	M	H	M	L	M
L	M	H	L	H	L
L	M	H	L	M	M
L	M	H	L	L	H
L	H	L	H	H	L
L	H	L	H	M	L
L	H	L	H	L	M
L	H	L	M	H	L
L	H	L	M	M	L
L	H	L	M	L	M
L	H	L	L	H	L
L	H	L	L	M	M
L	H	L	L	L	M
L	H	M	H	H	L
L	H	M	H	M	M
L	H	M	H	L	M
L	H	M	M	H	L
L	H	M	M	M	M
L	H	M	M	L	M
L	H	M	L	H	L
L	H	M	L	M	M
L	H	M	L	L	M
L	H	H	H	H	M
L	H	H	H	M	M
L	H	H	H	L	M
L	H	H	M	H	M

L	H	H	M	M	M
L	H	H	M	L	H
L	H	H	L	H	M
L	H	H	L	M	H
L	H	H	L	L	H
M	L	L	H	H	VL
M	L	L	H	M	VL
M	L	L	H	L	L
M	L	L	M	H	VL
M	L	L	M	M	L
M	L	L	M	L	L
M	L	L	L	H	L
M	L	L	L	M	L
M	L	L	L	L	M
M	L	M	H	H	L
M	L	M	H	M	L
M	L	M	H	L	M
M	L	M	M	H	L
M	L	M	M	M	L
M	L	M	M	L	M
M	L	M	L	H	L
M	L	M	L	M	L
M	L	M	L	L	M
M	L	H	H	H	L
M	L	H	H	M	M
M	L	H	H	L	M
M	L	H	M	H	L
M	L	H	M	M	M
M	L	H	M	L	M
M	L	H	L	H	L
M	L	H	L	M	M
M	L	H	L	L	H
M	M	L	H	H	L

M	M	L	H	M	L
M	M	L	H	L	M
M	M	L	M	H	L
M	M	L	M	M	L
M	M	L	M	L	M
M	M	L	L	H	L
M	M	L	L	M	M
M	M	L	L	L	M
M	M	M	H	H	L
M	M	M	H	M	M
M	M	M	H	L	M
M	M	M	M	H	L
M	M	M	M	M	M
M	M	M	M	L	M
M	M	M	L	H	M
M	M	M	L	M	M
M	M	M	L	L	M
M	M	H	H	H	M
M	M	H	H	M	M
M	M	H	H	L	H
M	M	H	M	H	M
M	M	H	M	M	M
M	M	H	M	L	H
M	M	H	L	H	M
M	M	H	L	M	H
M	M	H	L	L	H
M	H	L	H	H	L
M	H	L	H	M	M
M	H	L	H	L	M
M	H	L	M	H	L
M	H	L	M	M	M
M	H	L	M	L	M
M	H	L	L	H	M

M	H	L	L	M	M
M	H	L	L	L	H
M	H	M	H	H	M
M	H	M	H	M	M
M	H	M	H	L	M
M	H	M	M	H	M
M	H	M	M	M	M
M	H	M	M	L	H
M	H	M	L	H	M
M	H	M	L	M	H
M	H	M	L	L	H
M	H	H	H	H	M
M	H	H	H	M	M
M	H	H	H	L	H
M	H	H	M	H	M
M	H	H	M	M	H
M	H	H	M	L	H
M	H	H	L	H	H
M	H	H	L	M	H
M	H	H	L	L	VH
H	L	L	H	H	L
H	L	L	H	M	L
H	L	L	H	L	M
H	L	L	M	H	L
H	L	L	M	M	L
H	L	L	M	L	M
H	L	L	L	H	L
H	L	L	L	M	M
H	L	L	L	L	M
H	L	M	H	H	L
H	L	M	H	M	M
H	L	M	H	L	M
H	L	M	M	H	L

H	L	M	M	M	M
H	L	M	M	L	M
H	L	M	L	H	L
H	L	M	L	M	M
H	L	M	L	L	M
H	L	H	H	H	M
H	L	H	H	M	M
H	L	H	H	L	M
H	L	H	M	H	M
H	L	H	M	M	M
H	L	H	M	L	H
H	L	H	L	H	M
H	L	H	L	M	H
H	L	H	L	L	H
H	M	L	H	H	L
H	M	L	H	M	M
H	M	L	H	L	M
H	M	L	M	H	L
H	M	L	M	M	M
H	M	L	M	L	M
H	M	L	L	H	M
H	M	L	L	M	M
H	M	L	L	L	H
H	M	M	H	H	M
H	M	M	H	M	M
H	M	M	H	L	H
H	M	M	M	H	M
H	M	M	M	M	H
H	M	M	M	L	H
H	M	M	L	H	M
H	M	M	L	M	H
H	M	M	L	L	H
H	M	H	H	H	H

H	M	H	H	M	H
H	M	H	H	L	H
H	M	H	M	H	H
H	M	H	M	M	H
H	M	H	M	L	VH
H	M	H	L	H	H
H	M	H	L	M	VH
H	M	H	L	L	VH
H	H	L	H	H	M
H	H	L	H	M	M
H	H	L	H	L	H
H	H	L	M	H	M
H	H	L	M	M	M
H	H	L	M	L	H
H	H	L	L	H	H
H	H	L	L	M	H
H	H	L	L	L	H
H	H	M	H	H	M
H	H	M	H	M	H
H	H	M	H	L	H
H	H	M	M	H	H
H	H	M	M	M	H
H	H	M	M	L	H
H	H	M	L	H	H
H	H	M	L	M	H
H	H	M	L	L	VH
H	H	H	H	H	H
H	H	H	H	M	H
H	H	H	H	L	VH
H	H	H	M	H	H
H	H	H	M	M	VH
H	H	H	M	L	VH
H	H	H	L	H	VH

H	H	H	L	M	VH
H	H	H	L	L	VH

Tabla 3.1: Conjunto de reglas del sistema difuso de detección de incendios forestales.

3.4. Resultados y comparación de métodos

En esta sección se presentan los resultados obtenidos al aplicar los métodos descritos en secciones anteriores y se realiza una comparativa de los mismos, usando diferentes t-normas e índices de solapamiento (en el caso del algoritmo 4).

3.4.1. Método de Mamdani

El primer método estudiado es el *método de Mamdani*, presentado en la sección 2.5 (algoritmo 1). En la figura 3.2 y en la tabla 3.2 se presentan los resultados de aplicar el método de Mamdani, utilizando el conjunto de reglas definidas en la tabla 3.1 a diferentes entradas. Las entradas del sistema son valores escalares para cada una de las variables lingüísticas:

Fig.	**Temp.**	**Humo**	**Luz**	**Hum.**	**Dist.**	**Riesgo (%)**				
						cent. (∗)	**bis. (+)**	**som (▽)**	**mom (□)**	**lom (△)**
3.2a	25	0	200	20	70	20	18	0	6	12
3.2b	30	20	500	50	40	35	33	10	25	40
3.2c	40	50	500	30	40	43	45	38	50	62
3.2d	80	80	700	20	30	67	69	63	75	87
3.2e	100	90	900	10	20	80	82	84	92	100
3.2f	120	100	1000	10	10	91	92	92	96	100

Tabla 3.2: Resultados de aplicar el método de Mamdani a la detección de incendios forestales. Las gráficas correspondientes se muestran en la figura 3.2.

Los valores de salida (riesgo (%)) se obtienen aplicando los defusificadores definidos en la sección 2.4.2 al conjunto difuso obtenido al aplicar el método de Mamdani sobre las entradas (representados en la figura 3.2).

Se puede comprobar que los resultados obtenidos con el método de Mamdani parecen ser intuitivamente correctos. En el primer ejemplo (fig. 3.2a) las entradas sugieren un riesgo de incendio bajo, debido a la baja temperatura medida, el humo inexistente y la baja luminosidad. Aplicando el método de Mamdani efectivamente se obtiene un nivel de riesgo bajo (entre 0 %

y 20 %).

Como se puede ver, en general, los valores obtenidos con el defusificador *centroide* y el *bisector* son similares y pueden diferir ampliamente de los obtenidos con los métodos basados en el máximo. En este ejemplo se puede comprobar además que los defusificadores basados en el máximo pueden no dar resultados intuitivamente acertados, ya que el método del menor de los máximos da un resultado del 0 % de riesgo en el primer ejemplo, a pesar de que las entradas no son mínimas.

A medida que aumentan la temperatura, el humo y la luz y disminuyen la humedad y la distancia el riesgo de incendio aumenta (figuras 3.2c y 3.2e), lo cual es coherente con lo expresado en el conjunto de reglas.

3.4.2. Método de interpolación basado en índices de solapamiento

Para ejecutar el algoritmo 4 es necesario seleccionar una t-norma(T), un operador de agregación(M) y un índice de solapamiento (O). En este estudio se va a utilizar la media aritmética como operador de agregación. En cuanto a las t-normas, se van a utilizar las siguientes:

- **Mínimo:** $T_{min}(x_1, x_2, \cdots, x_n) = \min\{x_1, x_2, \cdots, x_n\}$
- **Producto:** $T_{prod}(x_1, x_2, \cdots, x_n) = x_1 * x_2 * \cdots * x_n$
- **Media geométrica:** $T_{geo}(x_1, x_2, \cdots, x_n) = (\prod_{i=1}^{n} x_i)^{\frac{1}{n}}$
- **Media armónica:** $T_{harm}(x_1, x_2, \cdots, x_n) = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$
- **Seno:** $T_{sin}(x_1, x_2, \cdots, x_n) = \frac{1}{n} \sin(\frac{\pi}{2}(x_1 * x_2 * \ldots * x_n))^{\frac{1}{4}}$
- **Einstein:** $T_{einstein}(x_1, x_2, \cdots, x_n) = \frac{(x_1 * x_2 * \cdots * x_n)}{1+(1-x_1)*\ldots*(1-x_n)}$

Además se van a utilizar los siguientes índices de solapamiento:

- **Media de productos:** $O_{\pi}(A, B) = \frac{1}{n} \sum_{i=1}^{n} \mu_A(x_i) * \mu_B(x_i)$
- **Media de mínimos:** $O_{avgmin}(A, B) = \frac{1}{n} \sum_{i=1}^{n} \min(\mu_A(x_i), \mu_B(x_i))$
- **Máximo de mínimos:** $O_Z(A, B) = \max_{i=1}^{n}(\min(\mu_A(x_i), \mu_B(x_i)))$
- **Media de raíces cuadradas:** $O_{\surd}(A, B) = \frac{1}{n} \sum_{i=1}^{n} \sqrt{\mu_A(x_i) * \mu_B(x_i)}$
- **Media del seno:** $O_{sin}(A, B) = \frac{1}{n} \sum_{i=1}^{n} \sin(\frac{\pi}{2}(\mu_A(x_i) * \mu_B(x_i))^{\frac{1}{4}})$

Así por ejemplo, si tomamos M = Media aritmética, $T = T_{min}$ y $O = O_Z$ y aplicamos el método de interpolación(algoritmo 4) a las entradas de la tabla 3.2 se obtienen los resultados mostrados en la tabla 3.3 y la figura 3.3:

Fig.	Temp.	Humo	Luz	Hum.	Dist.	Riesgo (%)				
						cent. (∗)	bis. (+)	som (▽)	mom (□)	lom (△)
3.3a	25	0	200	20	70	18	15	8	10	12
3.3b	30	20	500	50	40	30	30	32	36	40
3.3c	40	50	500	30	40	41	41	38	42	45
3.3d	80	80	700	20	30	69	69	63	66	68
3.3e	100	90	900	10	20	85	87	84	90	95
3.3f	120	100	1000	10	10	91	92	92	96	100

Tabla 3.3: Resultados de aplicar el método de interpolación con M = Media aritmética, $T = T_{min}$ y $O = O_Z$ a la detección de incendios forestales. Las gráficas correspondientes se muestran en la figura 3.3.

En este caso, los resultados son bastante similares a los obtenidos con el método de Mamdani, a pesar de que se obtienen conjuntos de salida con características diferentes. Los valores obtenidos con los defusificadores *centroide* y *bisector* se asemejan bastante a los obtenidos con el método de Mamdani, mientras que en los defusificadores basados en el máximo(*som*, *mom* y *lom*) las diferencias son más notables.

La ventaja de este método es que permite seleccionar diferentes combinaciones de t-normas e índices de solapamiento y evaluar las salidas obtenidas con cada uno de ellos. En la tabla 3.4 y en las figuras 3.4-3.9 se presentan los resultados obtenidos al aplicar el método de interpolación utilizando diferentes combinaciones de t-normas e índices de solapamiento a las entradas: $Temperatura = 30$ ºC, $Humo = 20$ ppm, $Luz = 500$ lux, $Humedad = 50$ ppm y $Distancia = 40$ m.

T-norma	Índice de solapamiento	Riesgo (%)				
		cent. (∗)	bis. (+)	som (▽)	mom (□)	lom (△)
T_{prod}	O_π	29	29	26	38	49
	O_{avgmin}	29	29	26	38	49
	O_Z	29	29	28	36	44
	$O_{\surd}$	30	30	26	38	49
	O_{sin}	31	31	26	38	49
T_{min}	O_π	31	31	26	38	49
	O_{avgmin}	31	31	26	38	49
	O_Z	30	30	32	36	40
	$O_{\surd}$	31	31	26	38	49
	O_{sin}	31	31	26	38	49
T_{geo}	O_π	31	31	26	38	49
	O_{avgmin}	31	31	26	38	49
	O_Z	31	30	32	32	32
	$O_{\surd}$	31	31	26	38	49
	O_{sin}	31	31	26	38	49
T_{harm}	O_π	31	31	26	38	49

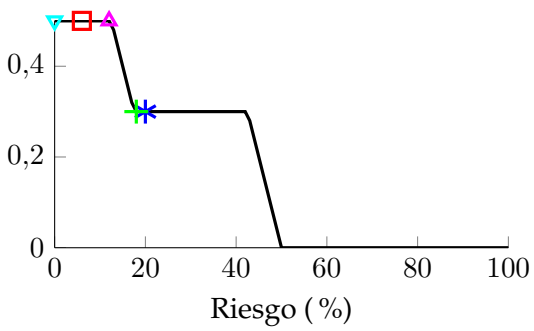
	O_{avgmin}	31	31	26	38	49
	O_Z	31	30	33	33	33
	$O_{\surd}$	31	31	26	38	49
	O_{sin}	31	31	26	38	49
T_{sin}	O_π	31	31	26	38	49
	O_{avgmin}	31	31	26	38	49
	O_Z	31	31	28	37	46
	$O_{\surd}$	31	31	26	38	49
	O_{sin}	31	31	26	38	49
$T_{einstein}$	O_π	29	29	26	38	49
	O_{avgmin}	29	29	26	38	49
	O_Z	29	29	28	36	44
	$O_{\surd}$	30	30	26	38	49
	O_{sin}	31	31	26	38	49

Tabla 3.4: Resultados de aplicar el método de interpolación con diferentes combinaciones de t-normas e índices de solapamiento.

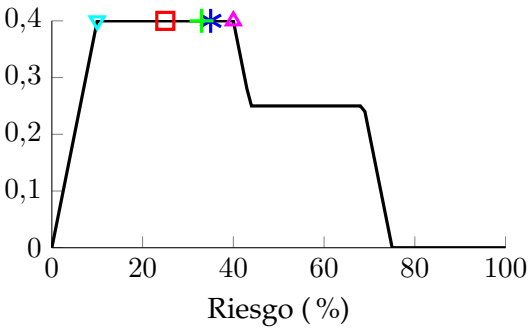
Como se puede ver en la tabla, los resultados finales (defusificados) al aplicar diferentes t-normas y operadores de agregación no varían significativamente (al menos con estos datos de entrada). En las gráficas se puede apreciar que los conjuntos de salida sí presentan algunas diferencias entre sí, especialmente cuando se utiliza el índice de solapamiento del máximo de mínimos (O_Z).

También existen diferencias notables en la escala de la función de pertenencia de los conjuntos de salida. El ejemplo más claro se puede ver en las gráficas obtenidas con la t-norma producto. Los valores son del orden de 10^{-14}, mientras que al utilizar otras t-normas como el mínimo se obtienen resultados "más grandes" (de órdenes entre 10^{-3} y 10^{-5}). Esto se debe a que los índices de solapamiento entre las entradas y los antecedentes son valores en el rango [0,1] y por tanto al realizar el producto de estos valores se obtienen números cada vez más pequeños.

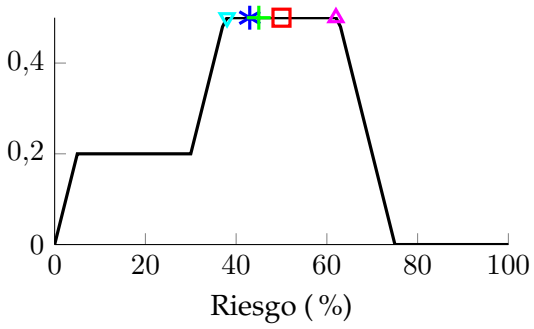
Es importante tener esto en cuenta a la hora de elegir una t-norma puesto que en el caso de reglas con muchos antecedentes se podrían producir problemas de precisión al trabajar con números tan pequeños. Un caso similar es el de la t-norma de Łukasiewicz, que no se puede aplicar de forma efectiva a reglas con más de dos antecedentes, puesto que el resultado tiende a cero muy rápidamente.



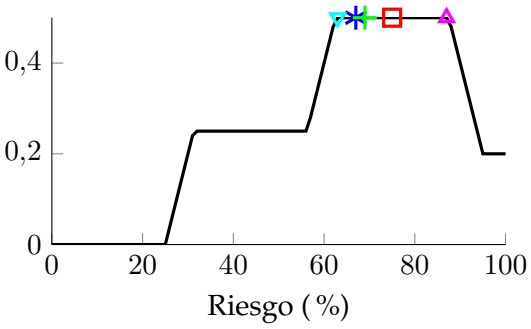
(a) Riesgo bajo.



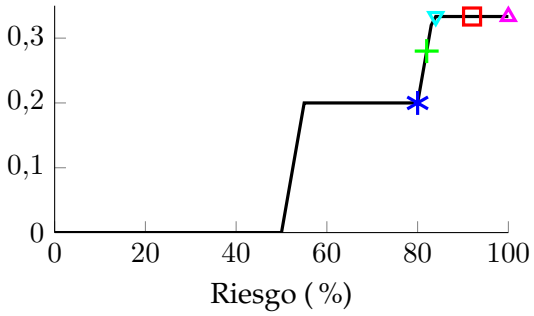
(b) Riesgo bajo-medio.



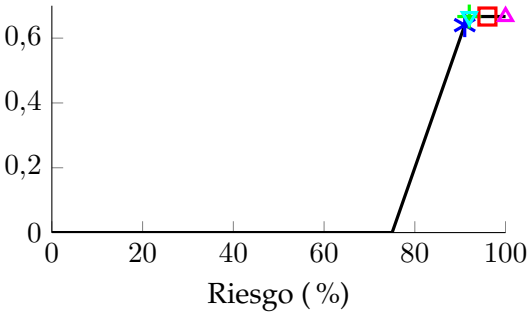
(c) Riesgo medio.



(d) Riesgo medio-alto.

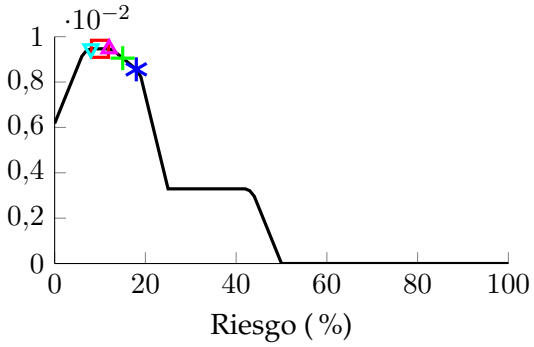


(e) Riesgo alto.

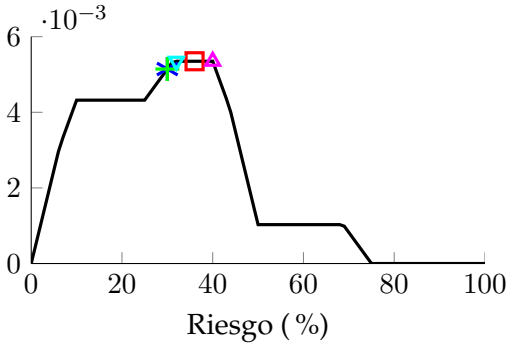


(f) Riesgo muy alto.

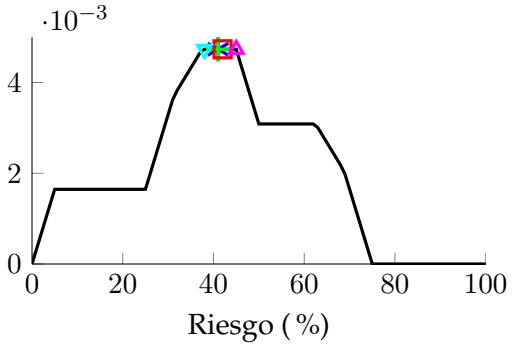
Figura 3.2: Método de Mamdani aplicado a la detección de incendios forestales.



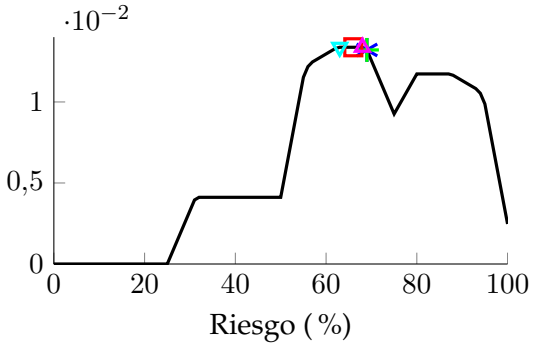
(a) Riesgo bajo.



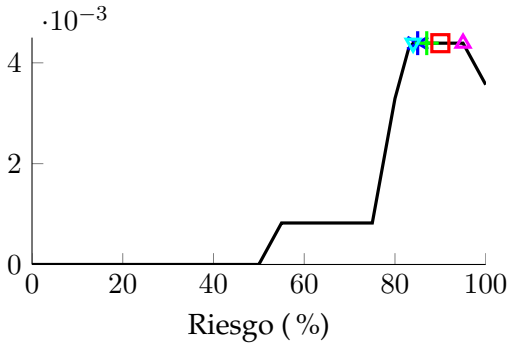
(b) Riesgo bajo-medio.



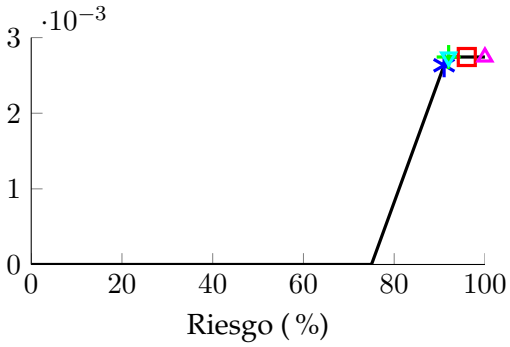
(c) Riesgo medio.



(d) Riesgo medio-alto.



(e) Riesgo alto.



(f) Riesgo muy alto.

Figura 3.3: Método de interpolación aplicado a la detección de incendios forestales.

T-norma: Producto (T_{prod})

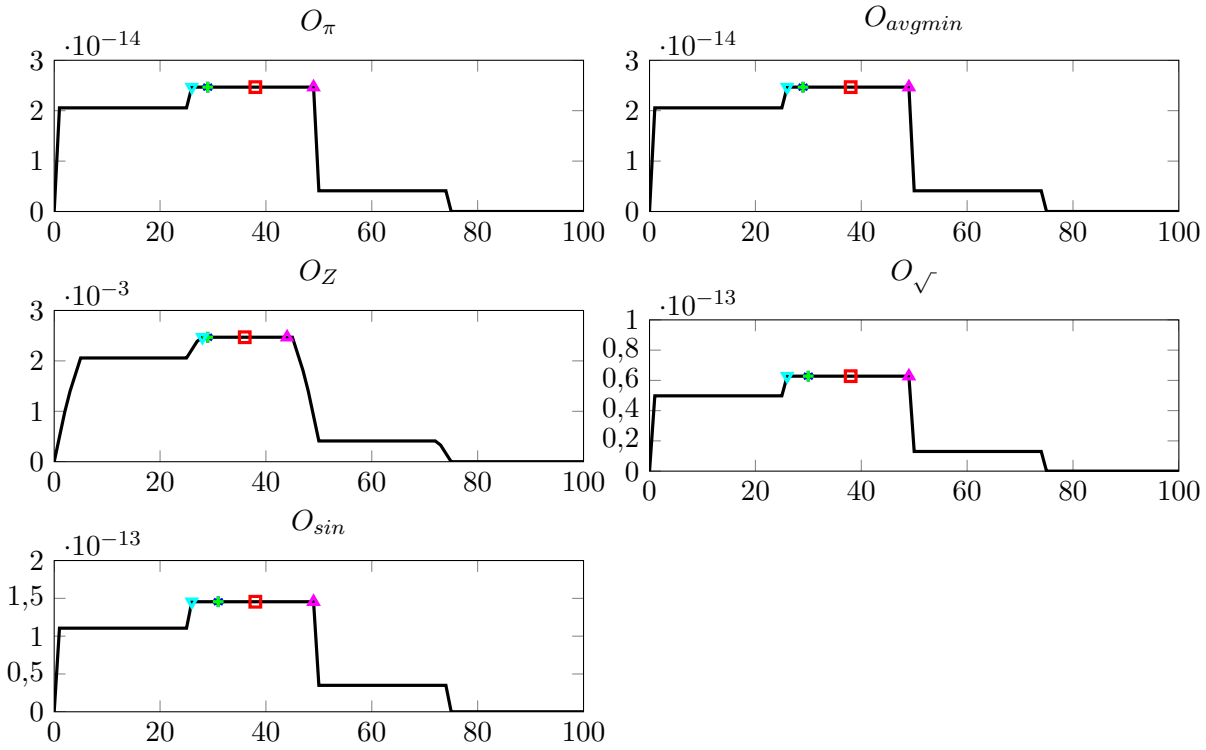


Figura 3.4: T-norma: Producto (T_{prod})

T-norma: Mínimo (T_{min})

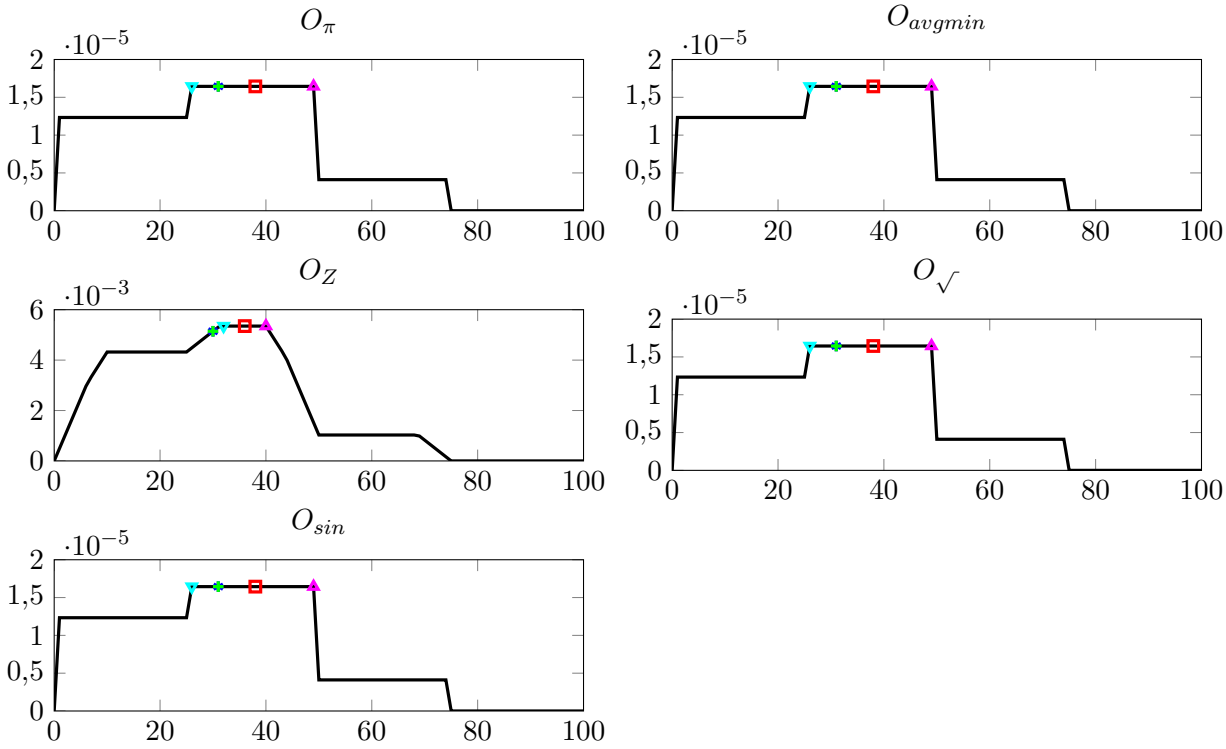


Figura 3.5: T-norma: Mínimo (T_{min})

T-norma: Media geométrica (T_{geo})

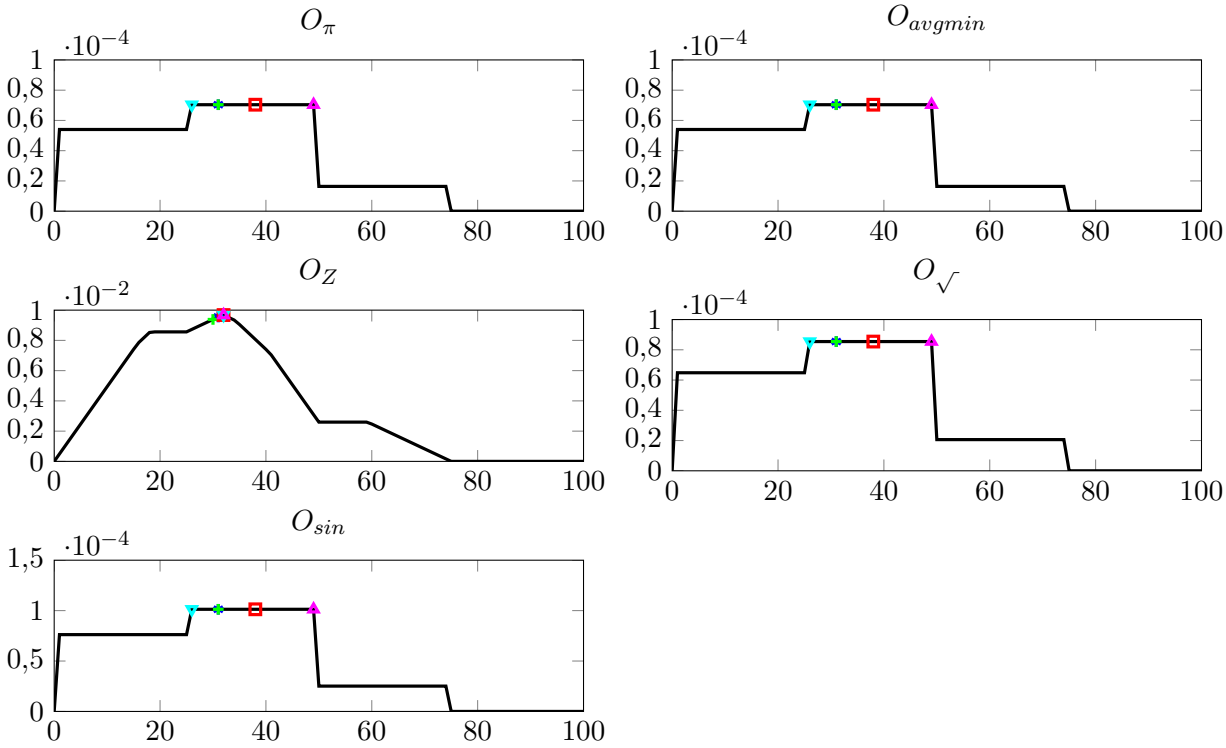


Figura 3.6: T-norma: Media geométrica (T_{geo})

T-norma: Media armónica (T_{harm})

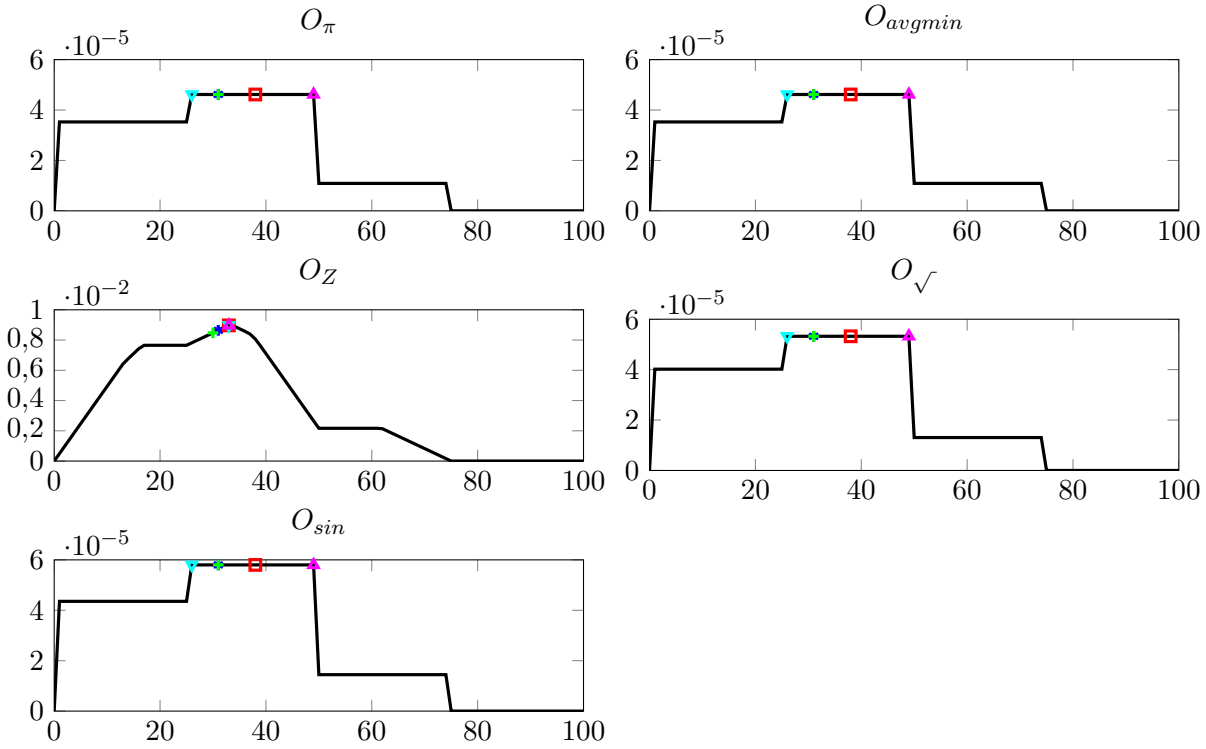


Figura 3.7: T-norma: Media armónica (T_{harm})

T-norma: Seno (T_{sin})

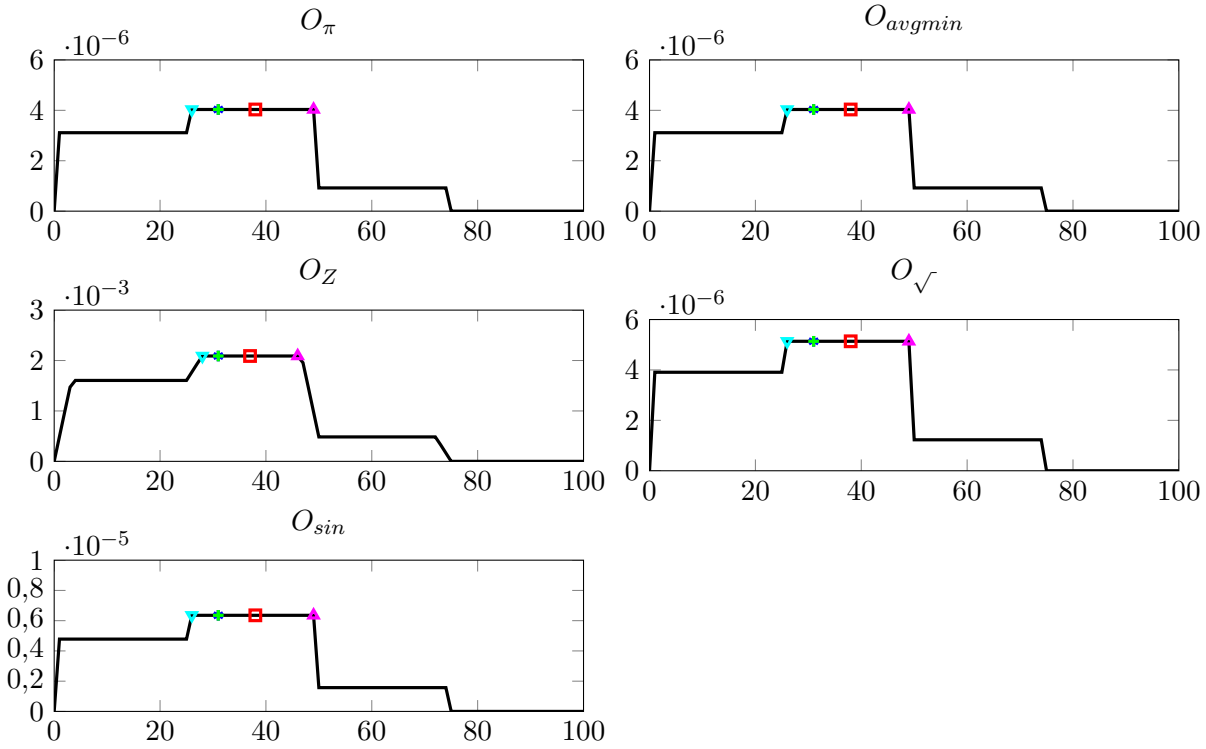


Figura 3.8: T-norma: Seno (T_{sin})

T-norma: Einstein ($T_{einstein}$)

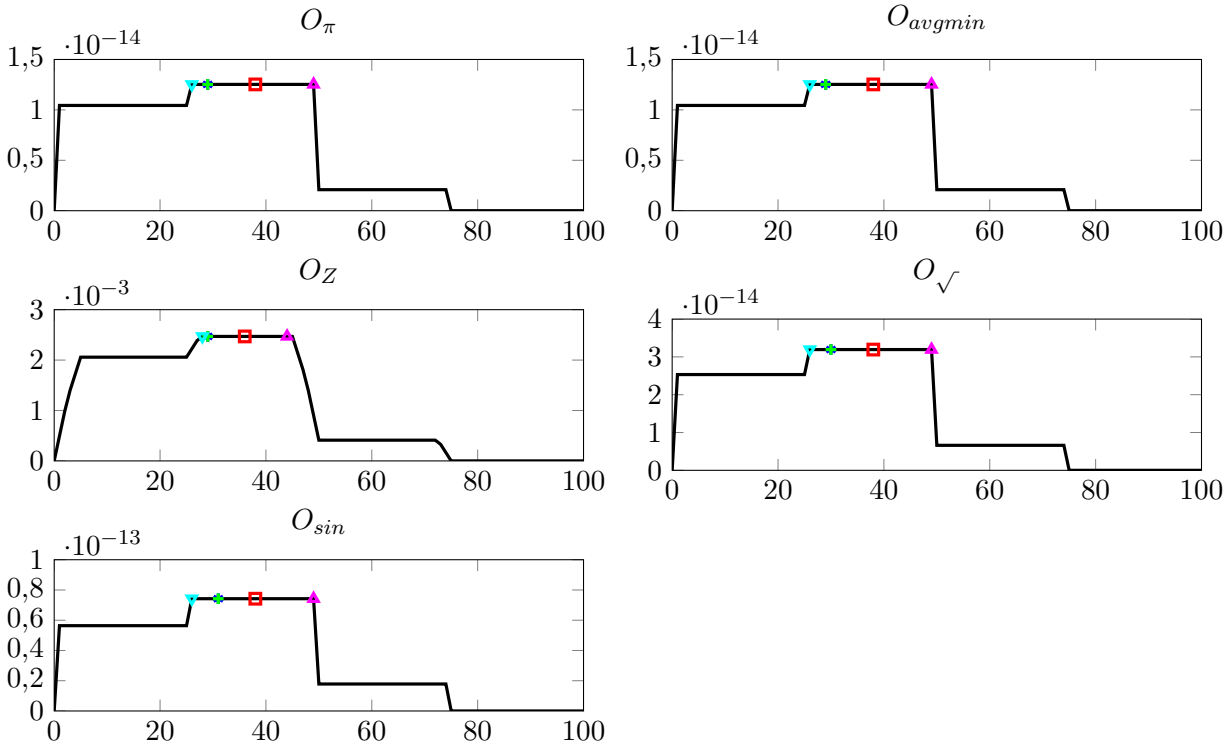


Figura 3.9: T-norma: Einstein ($T_{einstein}$)

4 Implementación en MATLAB

En este capítulo se incluye el código fuente de la implementación en MATLAB de los sistemas de lógica difusa estudiados en este trabajo.

4.1. General

En esta sección se incluyen algunas funciones de propósito general que han sido implementadas en el desarrollo de este proyecto.

4.1.1. T-normas y operadores de agregación

MATLAB dispone de una colección importante de funciones predefinidas para calcular t-normas y operadores de agregación. En este proyecto se han utilizado las siguientes:

- Media aritmética: `mean()`.
- Máximo: `max()`.
- Producto: `prod()`.
- Mínimo: `min()`.
- Media geométrica: `geomean()`.
- Media armónica: `harmmean()`.

Además se han implementado sendas funciones para calcular las t-normas del seno y de Einstein:

Código 4.1: T-norma del seno (`functions/sinmean.m`)

```
1 function s = sinmean( values )
2
3 if ~isempty(values)
4     s = sin(((pi/2)*prod(values))^(1/4))/length(values);
5 else
6     s = 0;
7 end
8
9 end
```

Código 4.2: T-norma de Einstein (`functions/einsteinmean.m`)

```
function e = einsteinmean( values )

e = prod(values)/(1+prod(arrayfun(@(x)(1-x),values)));

end
```

Como se puede ver, en la función `einsteinmean()` se ha utilizado una función anónima:

```
@(x)(1 - x)
```

que es una funcionalidad muy útil de MATLAB, inspirada en los lenguajes funcionales, que permite declarar funciones *lambda* o *anónimas* (sin nombre) y utilizarlas como argumentos de otras funciones (en este caso de `arrayfun()`, que aplica dicha función anónima a todos los elementos de un vector). Esta característica del lenguaje de MATLAB se utilizará también a la hora de definir índices de solapamiento.

4.1.2. Construcción de índices de solapamiento

Los índices de solapamiento (1.4) juegan un papel crucial en el desarrollo de este proyecto, al ser la base del método de interpolación estudiado (2.6). En la sección 1.4.2 se ha presentado un método para construir índices de solapamiento a partir de operadores de agregación y funciones de solapamiento.

Este mismo concepto de "construir" índices de solapamiento a partir de otras funciones se puede implementar en MATLAB. Para ello se va a utilizar una técnica de programación llamada *clausura*[1], que junto con las funciones anónimas, es muy utilizada en los lenguajes funcionales. A grandes rasgos, una clausura es una función de primer orden (puede tratarse como cualquier otra variable de tipo básico) que mantiene el estado de las variables del ámbito en el que fue creada. Así pues, el objetivo es definir una función que tome como parámetros una función de agregación y una de solapamiento y devuelva una nueva función que sea el índice de solapamiento construido a partir de estas funciones recibidas como parámetros.

Para implementar la construcción de índices de solapamiento se ha definido la función:

```
make_overlap_index( M, Go )
```

en `functions/make_overlap_index.m`, que toma como parámetros una función de agregación `M` y una de solapamiento `Go` y devuelve una nueva función que es el índice de solapamiento construido a partir de `M` y `Go` (según el teorema 1.4.2):

Código 4.3: Construcción de índices de solapamiento (`functions/make_overlap_index.m`).

```
% Construcción de índice de solapamiento
% M: Una función de agregación.
% Go: Una función de solapamiento
% O: Valor devuelto, el índice de solapamiento construido a partir de M y
```

[1] Conocida generalmente por su término inglés "closure".

```
% Go.
function O = make_overlap_index( M , Go )

    function o = overlap_index( A , B )
        o = M(arrayfun(Go,A,B));
    end

    O = @overlap_index;
end
```

El resultado de llamar a esta función es una nueva función que calcula el índice de solapamiento entre dos conjuntos pasados como parámetros `A` y `B`. Por ejemplo, si queremos construir el índice de solapamiento de Zadeh O_Z (ecuación 1.17) debemos pasar esta función un puntero a la función máximo (`@max`) como función de agregación y un puntero a la función mínimo (`@min`) como función de solapamiento:

```
Oz = make_overlap_index(@max,@min);
```

La función `Oz()` ahora puede utilizarse para calcular el índice de solapamiento entre dos conjuntos `A` y `B` pasados como parámetros:

```
overlap_index = Oz(A, B);
```

La construcción de los índices de solapamiento utilizados en la comparativa entre el método de interpolación y Mamdani (sec. 3.4.2) se ha implementado en la clase `O` (`functions/O.m`) utilizando métodos estáticos:

Código 4.4: Índices de solapamiento (`functions/O.m`).

```
% Índices de solapamiento
classdef O
    methods (Static)
        function O = pi()
            O = make_overlap_index(@mean,@(x,y)(x*y));
        end

        function O = avgmin()
            O = make_overlap_index(@mean,@min);
        end

        function O = maxmin()
            O = make_overlap_index(@max,@min);
        end

        function O = sqrt()
            O = make_overlap_index(@mean,@(x,y)(sqrt(x*y)));
        end

        function O = sin()
            O = make_overlap_index(@mean,@(x,y)(sin((pi/2)*(x*y)^(1/4))));
        end
    end
end
```

De esta forma se puede obtener el índice de solapamiento ya construido llamando a su función correspondiente. Por ejemplo, para obtener la consistencia de Zadeh:

```
1 Oz = O.maxmin();
```

4.2. Sistemas difusos basados en reglas

En esta sección se incluyen las funciones implementadas relacionadas con los sistemas difusos basados en reglas estudiados a lo largo del proyecto.

4.2.1. Conjuntos de reglas

Los conjuntos de reglas juegan un papel fundamental en los sistemas difusos basados en ellas, dado que expresan el conocimiento sobre el dominio del problema y tienen la misión de definir cómo se transforman las entradas del sistema difuso en salidas. Los métodos de Mamdani e interpolación implementados en este proyecto reciben el conjunto de reglas como un parámetro, de forma que pueden ser utilizados con cualquier conjunto de reglas y aplicados a cualquier tipo de problema.

Existen multitud de formas de implementar el conjunto de reglas. En este caso se ha optado por utilizar un vector de estructuras, debido a que el código resultante es sencillo de entender y de modificar. Una estructura en MATLAB es un tipo de variable compuesta que puede tener un número arbitrario de campos con nombre (similar a las estructuras de lenguajes como C). En el caso del conjunto de reglas, cada regla es una estructura `R` con los siguientes campos:

- `R.n`: Nº de regla.
- `R.A`: Vector con las funciones de pertenencia de los valores (punteros a funciones) de cada uno de los antecedentes de la regla.
- `R.B`: Función de pertenencia del valor del consecuente.

Por ejemplo, para definir una regla para el caso de la detección de incendios forestales tal que:

IF *Temperatura* es "Baja" AND *Humo* es "Alto" AND *Luz* es "Baja"
AND *Humedad* es "Alta" AND *Distancia* es "Media" THEN *Riesgo* es "Bajo" (4.1)

hay que construir una estructura de la siguiente forma:

```
1 R.n = 1;
2 R.A = {@temp.low, @smoke.high, @llight.low, @humidity.high, @distance.medium};
3 R.B = @threat.low;
```

Para definir un conjunto con varias reglas basta con añadir índices:

```
1 R(1).n = 1;
2 R(1).A = {@temp.low, @smoke.high, @llight.low, @humidity.high, ...
      @distance.medium};
3 R(1).B = @threat.low;
4
5 R(2).n = 2;
6 R(2).A = {@temp.low, @smoke.high, @llight.low, @humidity.high, @distance.high};
7 R(2).B = @threat.low;
```

La variable `R` resultante es un vector de estructuras y se puede obtener la regla *i-ésima* haciendo uso del índice (como cualquier otro vector): `R(i)`.

4.2.2. Fusificadores

Otro componente importante de los sistemas difusos basados en reglas son los fusificadores. Un fusificador es una función que transforma una entrada escalar del sistema en un conjunto difuso. Existen diferente tipos de fusificadores, tal y como se ha estudiado en la sección 2.4.1. En este proyecto, sin embargo, sólo se ha utilizado el fusificador *singleton* cuya implementación en MATLAB es la siguiente:

Código 4.5: Fusificador singleton (`functions/singleton_fuzzifier.m`)

```
1  % Difusificador singleton.
2  % Parámetros:
3  % x: universo de referencia.
4  % value: valor escalar de entrada
5  %
6  % Valor devuelto:
7  % f: conjunto difuso donde el valor "value" tiene grado de pertenencia 1 y
8  % el resto de valores 0.
9
10 function f = singleton_fuzzifier(x, value)
11 f.v(1,:) = x;
12 f.v(2,:) = zeros(length(f.v(1,:)),1);
13 f.v(2,f.v(1,:)==value) = 1;
14 end
```

4.2.3. Defusificadores

El último componente de un sistema difuso basado en reglas es el *defusificador*. Un defusificador es una función que transforma el conjunto difuso de salida en un valor escalar. En la sección 2.4.2 se han establecido las propiedades que debe tener un defusificador y se han definido algunos de los más utilizados.

MATLAB, en su toolbox *Fuzzy Logic,* ya tiene implementados los defusificadores más utilizados por medio de la función `defuzz(x, mf, type)`. Esta función toma como parámetros el universo de referencia del conjunto `x`, los grados de pertenencia para los valores del universo de referencia `mf` y el tipo de defusificador a utilizar `type`. Los tipos de defusificadores posibles son:

- `centroid`: Centroide.
- `bisector`: Bisector.
- `mom`: Media de máximos.
- `som`: Menor de máximos.
- `lom`: Mayor de máximos.

4.2.4. Método de Mamdani

El primero de los métodos estudiados en este proyecto es el método de Mamdani (algoritmo 1). La implementación de este método es una función que toma como parámetros:

- `R`: Conjunto de reglas tal y como se ha definido en la sección 4.2.1.
- `fact`: Premisa o valores escalares de entrada al sistema. Un vector con tantas posiciones como antecedentes tienen las reglas (variables lingüísticas), de forma que el valor para el antecedente *i-ésimo* debe estar en `fact(i)`.
- `y`: Universo de salida (vector).

Código 4.6: Método de Mamdani (`functions/mamdani.m`)

```
1  % Método de Mamdani.
2  % Parámetros:
3  % R: Conjunto de reglas.
4  % fact: Premisa. Vector de valores de entradas escalares. El valor para el
5  % antecedente i-ésimo debe estar en la posición fact(i).
6  % y: Universo de salida
7  %
8  % Devuelve:
9  % B = Conjunto de salida.
10
11 function B = mamdani( R, fact, y)
12
13 n = length(R);
14 B = zeros(length(R), length(y));
15
16  % Evaluar cada regla
17 for i=1:n
18    % Evaluar cada antecedente
19    u = zeros(1,length(R(i).A));
20    for j=1:length(R(i).A)
21        u(j) = R(i).A{j}(fact(j));
22    end
23
24    % Obtener el valor del conjunto difuso de salida para esta regla truncando ...
          el consecuente con
25    % el mínimo de los grados de pertenencia de cada antecedente.
26    B(i,:) = min(arrayfun(R(i).B, y),min(u));
27 end
28
```

```
29 % Obtener el conjunto de salida agregando todos los consecuentes obtenidos ...
       utilizando el máximo
30 B = max(B,[],1);
31
32 end
```

4.2.5. Método de interpolación basado en índices de solapamiento

El método más importante estudiado en este proyecto es el método de interpolación basado en índices de solapamiento (algoritmo 4). Este método se ha implementado en la función

```
interpolation(R, fact, y, O, T, M)
```

definida en `functions/interpolation.m` y que toma como parámetros:

- `R`: Conjunto de reglas tal y como se ha definido en la sección 4.2.1.
- `fact`: Premisa. Vector de conjuntos difusos con tantas posiciones como antecedentes tienen las reglas (variables lingüísticas), de forma que el valor para el antecedente *i-ésimo* debe estar en `fact(i)`.
- `y`: Universo de salida (vector).
- `O`: Índice de solapamiento. Función que puede construirse utilizando el método implementado en 4.3.
- `T`: T-norma. Utilizada para calcular el grado de activación de cada regla agregando los índices de solapamiento de cada antecedente.
- `M`: Operador de agregación.

El método toma estos parámetros y devuelve el conjunto difuso de salida, resultado de aplicar el conjunto de reglas (`R`) a la entrada (`fact`).

Código 4.7: Método de interpolación basado en índices de solapamiento (`functions/interpolation.m`)

```
1  % Método de interpolación basado en reglas.
2  % Parámetros:
3  % R: Conjunto de reglas.
4  % fact: Premisa. Vector de conjuntos difusos que componen la premisa. El ...
       valor para el
5  % antecedente i-ésimo debe estar en la posición fact(i).
6  % y: Universo de salida
7  % O: Índice de solapamiento utilizado.
8  % T: T-norma utilizada.
9  % M: Operador de agregación utilizado.
10 %
11 % Devuelve:
12 % Bp = Conjunto de salida.
13
```

```
14 function Bp = interpolation( R, fact, y, O , T, M )
15
16 n = length(R);
17 k=zeros(1,n);
18 K=zeros(n,length(y));
19 B=K;
20
21 for i=1:n
22     % Calcular el grado de activación de cada regla
23     k(i) = matching_degree(R(i), fact, O, T);
24     K(i,:) = k(i);
25
26     % Calcular el conjunto de salida B para esta regla
27     B(i,:) = arrayfun(R(i).B, y);
28 end
29
30 % Calcular el conjunto de salida agregando los conjuntos obtenidos en cada
31 % regla.
32 Bp = cellfun(M,num2cell(min(K,B),1));
33
34 end
```

Como se puede ver, en la línea 23 4.7 se utiliza la función:

```
matching_degree(R, fact, O, T)
```

definida en `functions/matching_degree.m`, para calcular el grado de activación de cada regla. Esta función toma como parámetros:

- `R`: Regla.
- `fact`: Premisa.
- `O`: Índice de solapamiento.
- `T`: T-norma.

Esta función calcula el índice de solapamiento (`O`) de cada antecedente de la regla `R` con su correspondiente valor en la premisa `fact` y agrega los valores obtenidos utilizando la t-norma `T`:

Código 4.8: Cálculo del grado de activación basado en índices de solapamiento (`functions/matching_degree.m`)

```
1  % Cálculo del grado de activación de una regla utilizando índices de
2  % solapamiento.
3  % Parámetros:
4  % R: Regla (estructura).
5  % fact: Premisa.
6  % O: Índice de solapamiento utilizado.
7  % T: T-norma utilizada.
8  %
9  % Devuelve:
10 % m = grado de activación
11
12 function m = matching_degree(R, fact, O, T)
```

```
13
14 overlaps = zeros(length(R.A),1);
15 for i=1:length(R.A)
16     overlaps(i) = O(fact(i).v(2,:) , arrayfun(R.A{i},fact(i).v(1,:)) );
17 end
18 m = T(overlaps);
19
20 end
```

Estos métodos, a pesar de ser correctos y devolver los resultados esperados, son mejorables en lo que a rendimiento se refiere. El principal problema de esta implementación es que se recalculan los conjuntos difusos de los antecedentes y el consecuente en cada regla (línea 27 en 4.7 y línea 16 en 4.8). Sin embargo, este cálculo sólo es necesario hacerlo la primera vez para cada valor de cada variable lingüística y después reutilizarlo.

Por ello, se ha implementado una nueva versión de estos métodos que utiliza una caché de conjuntos difusos ya calculados, que se almacenan en una especie de *tabla hash*. La tabla hash se ha implementado utilizando una clausura, de forma que se devuelve una función que mantiene la estructura de la tabla hash en el ámbito y la utiliza en llamadas posteriores. La función

```
fuzzyset_hash()
```

definida en `functions/fuzzyset_hash.m` se ha implementado de la siguiente manera:

Código 4.9: Función para construir una tabla hash de conjuntos difusos

```
1  % Hash de conjuntos difusos
2  % Devuelve:
3  %
4  % fh: Puntero a una función que implementa una tabla hash de conjuntos
5  % difusos con parámetros:
6  %   mf: Función de pertenencia del conjunto.
7  %   x: Universo de referencia.
8  % Si el conjunto se ha calculado previamente, se devuelve dicho valor
9  % precalculado. En caso contrario, se aplica la función de pertenencia mf
10 % al universo de referencia x.
11 function fh = fuzzyset_hash()
12     hash = struct;
13
14     function fs = get_fuzzy_set( mf , x )
15         f_key = strrep(func2str(mf),'.','');
16         if(isfield(hash, f_key) == 0)
17             fs = arrayfun(mf, x);
18             hash.(f_key) = fs;
19         else
20             fs = getfield(hash, f_key);
21         end
22     end
23
24     fh = @get_fuzzy_set;
25 end
```

El resultado de llamar a esta función es un puntero a una nueva función (`get_fuzzy_set(mf, ... x)`) que calcula los grados de pertenencia de un conjunto difuso dados su función de pertenencia `mf` y su universo de referencia `x`. En caso de que este cálculo haya sido realizado previa-

mente, devuelve el resultado anterior, para evitar trabajo innecesario.

Esta función hace uso de la estructura `hash` (línea 12), que es una variable que ha quedado en su ámbito al crearse la función. Esta variable permite guardar los cálculos realizados en llamadas anteriores a la función y mantener el estado de la tabla hash mientras la función exista. La tabla hash de conjuntos se indexa utilizando el nombre de la función de pertenencia (obtenido mediante la función `func2str()`).

Una vez implementada esta función, se puede utilizar en los métodos `interpolation()` y `matching_degree()` para ganar rendimiento:

Código 4.10: Método de interpolación basado en índices de solapamiento (`functions/hashed_interpolation.m`)

```
1  % Método de interpolación basado en reglas.
2  % Parámetros:
3  % R: Conjunto de reglas.
4  % fact: Premisa. Vector de conjuntos difusos que componen la premisa. El ...
       valor para el
5  % antecedente i-ésimo debe estar en la posición fact(i).
6  % y: Universo de salida
7  % O: Índice de solapamiento utilizado.
8  % T: T-norma utilizada.
9  % M: Operador de agregación utilizado.
10 %
11 % Devuelve:
12 % Bp = Conjunto de salida.
13
14 function Bp = hashed_interpolation( R, fact, y, O , T, M )
15
16 % Inicializar la tabla hash de conjuntos
17 hash = fuzzyset_hash();
18
19 n = length(R);
20 k=zeros(1,n);
21 K=zeros(n,length(y));
22 B=K;
23
24 for i=1:n
25     % Calcular el grado de activación de cada regla.
26     k(i) = hashed_matching_degree(hash, R(i), fact, O, T);
27     K(i,:) = k(i);
28
29     % Calcular el conjunto de salida B para esta regla. Utilizar la función
30     % de hash para evitar cálculos innecesarios.
31     B(i,:) = hash(R(i).B, y);
32 end
33
34 % Calcular el conjunto de salida agregando los conjuntos obtenidos en cada
35 % regla.
36 Bp = cellfun(M,num2cell(min(K,B),1));
37
38 end
```

Código 4.11: Cálculo del grado de activación basado en índices de solapamiento (`functions/hashed_matching_degree.m`)

```
1  % Cálculo del grado de activación de una regla utilizando índices de
2  % solapamiento.
```

```
3  % Parámetros:
4  % hash: Tabla hash de conjuntos difusos.
5  % R: Regla (estructura).
6  % fact: Premisa.
7  % O: Índice de solapamiento utilizado.
8  % T: T-norma utilizada.
9  %
10 % Devuelve:
11 % m = grado de activación
12
13 function m = hashed_matching_degree(hash, R, fact, O, T)
14
15 overlaps = zeros(length(R.A),1);
16 for i=1:length(R.A)
17     overlaps(i) = O(fact(i).v(2,:) , hash(R.A{i},fact(i).v(1,:)) );
18 end
19 m = T(overlaps);
20
21 end
```

A pesar de que esta función es muy simple, la ganancia de rendimiento al utilizar esta técnica es notable.

4.3. Detección de incendios forestales

4.3.1. Variables lingüísticas

Como se ha definido en 1.5, una variable lingüística es una variable que puede tomar conjuntos difusos como valores. Así mismo, un conjunto difuso viene definido por su función de pertenencia. Por tanto, se debe definir una función para cada valor de cada variable lingüística utilizada.

Generalmente, en MATLAB, cada función se define en un fichero separado que únicamente contiene el código de dicha función. En este caso y dado que se utilizan 5 variables lingüísticas para los antecedentes con 3 valores posibles cada una y una variable de salida para el consecuente con 5 valores posibles, habría que crear 20 ficheros para definir estas funciones. Aunque es perfectamente posible y válido hacerlo de esta manera, resulta tedioso manejar tantos ficheros.

Una solución adecuada al problema es definir cada variable lingüística en un fichero en el que se implementan las funciones de pertenencia de sus posibles valores. Para ello se puede crear una clase para cada variable lingüística y definir la función de pertenencia de cada valor como un método estático de la misma. Esta forma de definir las variables lingüísticas tiene la ventaja de que todo el código asociado a la variable está en un mismo fichero. Además la forma de utilizar las funciones resulta muy cómoda y expresiva.

Por ejemplo, para la variable lingüística χ_1 = *Temperatura* se ha definido la clase `temp` en el fichero `lang_variables/temp.m`. Esta clase tiene los siguientes métodos estáticos:

- `get_x()` : Devuelve el universo de referencia para la variable (en este caso el vector $[0, 100]$).

- `low(t)` : Función de pertenencia para el valor "Bajo".
- `medium(t)` : Función de pertenencia para el valor "Medio".
- `high(t)` : Función de pertenencia para el valor "Alto".

De esta forma, para obtener el grado de pertenencia de un valor `t` al conjunto *"Temperatura Media"* basta con hacer:

```
1 v = temp.medium(t);
```

Además es posible obtener un puntero a estas funciones utilizando el operador `@`:

```
1 fh = @temp.medium;
2 v = fh(t);
```

La variable `fh` es un puntero a la función `temp.medium()` y puede utilizarse de la misma forma que ésta (línea 2). Además la variable `fh` se puede utilizar de forma similar a otros tipos de variables, es decir, se pueden crear vectores de punteros de funciones, pasarlos como parámetros a otras funciones etc. Esta propiedad se utilizará de forma intensiva a la hora de implementar los métodos estudiados en este proyecto y el conjunto de reglas.

A continuación se incluyen las definiciones de las variables lingüísticas utilizadas en la aplicación práctica de detección de incendios forestales:

Código 4.12: χ_1 - Temperatura (`lang_variables/temp.m`)

```
1  % Variable lingüística: Temperatura (ºC)
2  classdef temp
3      methods (Static)
4          % Universo
5          function x = get_x()
6              x = 0:120;
7          end
8
9          % Valores
10         function u = low(temp)
11             u = lineal(0,50,temp,'r');
12         end
13
14         function u = medium(temp)
15             u = max([lineal(10,60,temp),lineal(60,110,temp,'r')]);
16         end
17
18         function u = high(temp)
19             u = lineal(70,120,temp);
20         end
21     end
22 end
```

Código 4.13: χ_2 - Humo (`lang_variables/smoke.m`)

```
1  % Variable lingüística: Humo (ppm)
```

```
classdef smoke
    methods (Static)
        % Universo
        function x = get_x()
            x = 0:100;
        end

        % Valores
        function u = low(smoke)
            u = lineal(0,40,smoke,'r');
        end

        function u = medium(smoke)
            u = max([lineal(10,50,smoke),lineal(50,90,smoke,'r')]);
        end

        function u = high(smoke)
            u = lineal(60,100,smoke);
        end
    end
end
```

Código 4.14: χ_3 - Luz (`lang_variables/llight.m`)

```
% Variable lingüística: Luz (lux)
classdef llight
    methods (Static)
        % Universo
        function x = get_x()
            x = 0:1000;
        end

        % Valores
        function u = low(light)
            u = lineal(0,400,light,'r');
        end

        function u = medium(light)
            u = max([lineal(100,500,light),lineal(500,900,light,'r')]);
        end

        function u = high(light)
            u = lineal(600,1000,light);
        end
    end
end
```

Código 4.15: χ_4 - Humedad (`lang_variables/humidity.m`)

```
% Variable lingüística: Humedad (ppm)
classdef humidity
    methods (Static)
        % Universo
        function x = get_x()
            x = 0:100;
        end

        % Valores
        function u = low(humidity)
            u = lineal(0,40,humidity,'r');
        end
```

```
13
14          function u = medium(humidity)
15              u = max([lineal(10,50,humidity),lineal(50,90,humidity,'r')]);
16          end
17
18          function u = high(humidity)
19              u = lineal(60,100,humidity);
20          end
21      end
22  end
```

Código 4.16: χ_5 - Distancia (`lang_variables/distance.m`)

```
1   % Variable lingüística: Distancia (m)
2   classdef distance
3       methods (Static)
4           % Universo
5           function x = get_x()
6               x = 0:80;
7           end
8
9           % Valores
10          function u = close(distance)
11              u = lineal(0,30,distance,'r');
12          end
13
14          function u = medium(distance)
15              u = max([lineal(10,40,distance),lineal(40,70,distance,'r')]);
16          end
17
18          function u = far(distance)
19              u = lineal(50,80,distance);
20          end
21      end
22  end
```

Código 4.17: y - Riesgo (`lang_variables/threat.m`)

```
1   % Variable lingüística: Riesgo del incendio (%C)
2   classdef threat
3       methods (Static)
4           % Universo
5           function x = get_x()
6               x = 0:100;
7           end
8
9           % Valores
10          function u = very_low(threat)
11              u = lineal(0,25,threat,'r');
12          end
13
14          function u = low(threat)
15              u =  max([lineal(0,25,threat),lineal(25,50,threat,'r')]);
16          end
17
18          function u = medium(threat)
19              u = max([lineal(25,50,threat),lineal(50,75,threat,'r')]);
20          end
21
22          function u = high(threat)
23              u = max([lineal(50,75,threat),lineal(75,100,threat,'r')]);
```

```
24          end
25
26          function u = very_high(threat)
27              u = lineal(75,100,threat);
28          end
29      end
30  end
```

La función `lineal()` es una función muy simple que permite calcular el valor de una función de pertenencia lineal entre los puntos pasados en los dos primeros parámetros. Esta función toma los siguientes parámetros:

- `first`: Primer punto del rango (eje x).
- `last`: Último punto del rango (eje x).
- `value`: Punto en el que se quiere calcular el valor de la función de pertenencia (eje x).
- `r`: Parámetro opcional. Si tiene valor 'r' se calcula la función de pertenencia "inversa".

Este método construye la función de pertenencia lineal que pasa entre los puntos (`first`, 0) y (`last`, 1) o (`first`, 1) y (`last`, 0) en caso de que se pase el último parámetro con valor `'r'` y obtiene el grado de pertenencia del punto `value` para esa función. En caso de que el valor a calcular esté fuera del rango [`first`, `last`], el grado de pertenencia devuelto es 0:

Código 4.18: Función de pertenencia lineal (`functions/lineal.m`)

```
1  % Función de pertenencia lineal.
2  % Parámetros:
3  % first: Primer punto del rango (eje x).
4  % last: Último punto del rango (eje x).
5  % value: Punto en el que se quiere calcular el valor de la función de ...
       pertenencia (eje x).
6  % r: Parámetro opcional. Si tiene valor 'r' se calcula la función de
7  % pertenencia "inversa" (de 1 a 0).
8  %
9  % Valor devuelto:
10 % u = valor de la función de pertenencia en el punto "value"
11
12 function u = lineal(first, last, value, r)
13     if value >= first && value <= last
14         u = (value - first)/(last - first);
15         if(nargin == 4 && r == 'r')
16             u = 1 - u;
17         end
18     else
19         u = 0;
20     end
21 end
```

4.3.2. Conjunto de reglas

El conjunto de reglas se ha definido en la función:

```
fire_detection_rules()
```

en `fire_detection_rules.m`, de forma que sea sencillo obtener y utilizar este conjunto en otros scripts y funciones. Esta función devuelve un vector de estructuras tal y como se ha definido en la sección 4.2.1.

Este tipo de estructura tiene la ventaja de que es sencilla de implementar y de entender. Sin embargo, para conjuntos de reglas grandes (como es el caso) resulta tedioso tener que indicar el índice para cada regla y repetir una y otra vez el nombre del conjunto (`R`) o de los campos.

Por esta razón se ha optado por una forma de definir el conjunto de reglas más concisa y clara. El conjunto de reglas se define como una matriz, en la que cada fila corresponde a una regla y cada columna es un puntero a la función de pertenencia del valor de la variable lingüística que ocupa esa posición. Las primeras $n-1$ columnas de cada fila corresponden a los antecedentes de la regla, y la última al consecuente.

En la función `fire_detection_rules` (código 4.19) se define el conjunto de reglas en la matriz `rules` (líneas 2-335) de esta forma. Dado que el resto de funciones implementadas esperan un conjunto de reglas definido como un vector de estructuras, se transforma esta matriz en dicho vector en las líneas 340-344. De esta forma, el conjunto de reglas es muy sencillo de leer y modificar y se asemeja mucho a la tabla 3.1.

Código 4.19: Conjunto de reglas (`fire_detection_rules.m`)

```
1  function R = fire_detection_rules()
2  rules = {
3  %temp.low y smoke.low
4  @temp.low,@smoke.low,@llight.low,@humidity.high,@distance.far,@threat.very_low;
5  @temp.low,@smoke.low,@llight.low,@humidity.high,@distance.medium,@threat.very_low;
6  @temp.low,@smoke.low,@llight.low,@humidity.high,@distance.close,@threat.very_low;
7
8  @temp.low,@smoke.low,@llight.low,@humidity.medium,@distance.far,@threat.very_low;
9  @temp.low,@smoke.low,@llight.low,@humidity.medium,@distance.medium,@threat.very_low;
10 @temp.low,@smoke.low,@llight.low,@humidity.medium,@distance.close,@threat.low;
11
12 @temp.low,@smoke.low,@llight.low,@humidity.low,@distance.far,@threat.very_low;
13 @temp.low,@smoke.low,@llight.low,@humidity.low,@distance.medium,@threat.low;
14 @temp.low,@smoke.low,@llight.low,@humidity.low,@distance.close,@threat.low;
15
16 @temp.low,@smoke.low,@llight.medium,@humidity.high,@distance.far,@threat.very_low;
17 @temp.low,@smoke.low,@llight.medium,@humidity.high,@distance.medium,@threat.low;
18 @temp.low,@smoke.low,@llight.medium,@humidity.high,@distance.close,@threat.low;
19
20 @temp.low,@smoke.low,@llight.medium,@humidity.medium,@distance.far,@threat.very_low;
21 @temp.low,@smoke.low,@llight.medium,@humidity.medium,@distance.medium,@threat.low;
22 @temp.low,@smoke.low,@llight.medium,@humidity.medium,@distance.close,@threat.low;
23
24 @temp.low,@smoke.low,@llight.medium,@humidity.low,@distance.far,@threat.very_low;
25 @temp.low,@smoke.low,@llight.medium,@humidity.low,@distance.medium,@threat.low;
26 @temp.low,@smoke.low,@llight.medium,@humidity.low,@distance.close,@threat.low;
27
28 @temp.low,@smoke.low,@llight.high,@humidity.high,@distance.far,@threat.low;
29 @temp.low,@smoke.low,@llight.high,@humidity.high,@distance.medium,@threat.low;
30 @temp.low,@smoke.low,@llight.high,@humidity.high,@distance.close,@threat.low;
31
32 @temp.low,@smoke.low,@llight.high,@humidity.medium,@distance.far,@threat.low;
```

```
@temp.low,@smoke.low,@llight.high,@humidity.medium,@distance.medium,@threat.low;
@temp.low,@smoke.low,@llight.high,@humidity.medium,@distance.close,@threat.medium;

@temp.low,@smoke.low,@llight.high,@humidity.low,@distance.far,@threat.low;
@temp.low,@smoke.low,@llight.high,@humidity.low,@distance.medium,@threat.medium;
@temp.low,@smoke.low,@llight.high,@humidity.low,@distance.close,@threat.medium;

%temp.low y smoke.medium
@temp.low,@smoke.medium,@llight.low,@humidity.high,@distance.far,@threat.very_low;
@temp.low,@smoke.medium,@llight.low,@humidity.high,@distance.medium,@threat.very_low;
@temp.low,@smoke.medium,@llight.low,@humidity.high,@distance.close,@threat.low;

@temp.low,@smoke.medium,@llight.low,@humidity.medium,@distance.far,@threat.very_low;
@temp.low,@smoke.medium,@llight.low,@humidity.medium,@distance.medium,@threat.low;
@temp.low,@smoke.medium,@llight.low,@humidity.medium,@distance.close,@threat.low;

@temp.low,@smoke.medium,@llight.low,@humidity.low,@distance.far,@threat.low;
@temp.low,@smoke.medium,@llight.low,@humidity.low,@distance.medium,@threat.low;
@temp.low,@smoke.medium,@llight.low,@humidity.low,@distance.close,@threat.medium;

@temp.low,@smoke.medium,@llight.medium,@humidity.high,@distance.far,@threat.low;
@temp.low,@smoke.medium,@llight.medium,@humidity.high,@distance.medium,@threat.low;
@temp.low,@smoke.medium,@llight.medium,@humidity.high,@distance.close,@threat.medium;

@temp.low,@smoke.medium,@llight.medium,@humidity.medium,@distance.far,@threat.low;
@temp.low,@smoke.medium,@llight.medium,@humidity.medium,@distance.medium,@threat.low;
@temp.low,@smoke.medium,@llight.medium,@humidity.medium,@distance.close,@threat.medium;

@temp.low,@smoke.medium,@llight.medium,@humidity.low,@distance.far,@threat.low;
@temp.low,@smoke.medium,@llight.medium,@humidity.low,@distance.medium,@threat.low;
@temp.low,@smoke.medium,@llight.medium,@humidity.low,@distance.close,@threat.medium;

@temp.low,@smoke.medium,@llight.high,@humidity.high,@distance.far,@threat.low;
@temp.low,@smoke.medium,@llight.high,@humidity.high,@distance.medium,@threat.medium;
@temp.low,@smoke.medium,@llight.high,@humidity.high,@distance.close,@threat.medium;

@temp.low,@smoke.medium,@llight.high,@humidity.medium,@distance.far,@threat.low;
@temp.low,@smoke.medium,@llight.high,@humidity.medium,@distance.medium,@threat.medium;
@temp.low,@smoke.medium,@llight.high,@humidity.medium,@distance.close,@threat.medium;

@temp.low,@smoke.medium,@llight.high,@humidity.low,@distance.far,@threat.low;
@temp.low,@smoke.medium,@llight.high,@humidity.low,@distance.medium,@threat.medium;
@temp.low,@smoke.medium,@llight.high,@humidity.low,@distance.close,@threat.high;

%temp.low y smoke.high
@temp.low,@smoke.high,@llight.low,@humidity.high,@distance.far,@threat.low;
@temp.low,@smoke.high,@llight.low,@humidity.high,@distance.medium,@threat.low;
@temp.low,@smoke.high,@llight.low,@humidity.high,@distance.close,@threat.medium;

@temp.low,@smoke.high,@llight.low,@humidity.medium,@distance.far,@threat.low;
@temp.low,@smoke.high,@llight.low,@humidity.medium,@distance.medium,@threat.low;
@temp.low,@smoke.high,@llight.low,@humidity.medium,@distance.close,@threat.medium;

@temp.low,@smoke.high,@llight.low,@humidity.low,@distance.far,@threat.low;
@temp.low,@smoke.high,@llight.low,@humidity.low,@distance.medium,@threat.medium;
@temp.low,@smoke.high,@llight.low,@humidity.low,@distance.close,@threat.medium;

@temp.low,@smoke.high,@llight.medium,@humidity.high,@distance.far,@threat.low;
@temp.low,@smoke.high,@llight.medium,@humidity.high,@distance.medium,@threat.medium;
@temp.low,@smoke.high,@llight.medium,@humidity.high,@distance.close,@threat.medium;

@temp.low,@smoke.high,@llight.medium,@humidity.medium,@distance.far,@threat.low;
@temp.low,@smoke.high,@llight.medium,@humidity.medium,@distance.medium,@threat.medium;
```

```
96  @temp.low,@smoke.high,@llight.medium,@humidity.medium,@distance.close,@threat.medium;
97
98  @temp.low,@smoke.high,@llight.medium,@humidity.low,@distance.far,@threat.low;
99  @temp.low,@smoke.high,@llight.medium,@humidity.low,@distance.medium,@threat.medium;
100 @temp.low,@smoke.high,@llight.medium,@humidity.low,@distance.close,@threat.medium;
101
102 @temp.low,@smoke.high,@llight.high,@humidity.high,@distance.far,@threat.medium;
103 @temp.low,@smoke.high,@llight.high,@humidity.high,@distance.medium,@threat.medium;
104 @temp.low,@smoke.high,@llight.high,@humidity.high,@distance.close,@threat.medium;
105
106 @temp.low,@smoke.high,@llight.high,@humidity.medium,@distance.far,@threat.medium;
107 @temp.low,@smoke.high,@llight.high,@humidity.medium,@distance.medium,@threat.medium;
108 @temp.low,@smoke.high,@llight.high,@humidity.medium,@distance.close,@threat.high;
109
110 @temp.low,@smoke.high,@llight.high,@humidity.low,@distance.far,@threat.medium;
111 @temp.low,@smoke.high,@llight.high,@humidity.low,@distance.medium,@threat.high;
112 @temp.low,@smoke.high,@llight.high,@humidity.low,@distance.close,@threat.high;
113
114 %temp.medium y smoke.low
115 @temp.medium,@smoke.low,@llight.low,@humidity.high,@distance.far,@threat.very_low;
116 @temp.medium,@smoke.low,@llight.low,@humidity.high,@distance.medium,@threat.very_low;
117 @temp.medium,@smoke.low,@llight.low,@humidity.high,@distance.close,@threat.low;
118
119 @temp.medium,@smoke.low,@llight.low,@humidity.medium,@distance.far,@threat.very_low;
120 @temp.medium,@smoke.low,@llight.low,@humidity.medium,@distance.medium,@threat.low;
121 @temp.medium,@smoke.low,@llight.low,@humidity.medium,@distance.close,@threat.low;
122
123 @temp.medium,@smoke.low,@llight.low,@humidity.low,@distance.far,@threat.low;
124 @temp.medium,@smoke.low,@llight.low,@humidity.low,@distance.medium,@threat.low;
125 @temp.medium,@smoke.low,@llight.low,@humidity.low,@distance.close,@threat.medium;
126
127 @temp.medium,@smoke.low,@llight.medium,@humidity.high,@distance.far,@threat.low;
128 @temp.medium,@smoke.low,@llight.medium,@humidity.high,@distance.medium,@threat.low;
129 @temp.medium,@smoke.low,@llight.medium,@humidity.high,@distance.close,@threat.medium;
130
131 @temp.medium,@smoke.low,@llight.medium,@humidity.medium,@distance.far,@threat.low;
132 @temp.medium,@smoke.low,@llight.medium,@humidity.medium,@distance.medium,@threat.low;
133 @temp.medium,@smoke.low,@llight.medium,@humidity.medium,@distance.close,@threat.medium;
134
135 @temp.medium,@smoke.low,@llight.medium,@humidity.low,@distance.far,@threat.low;
136 @temp.medium,@smoke.low,@llight.medium,@humidity.low,@distance.medium,@threat.low;
137 @temp.medium,@smoke.low,@llight.medium,@humidity.low,@distance.close,@threat.medium;
138
139 @temp.medium,@smoke.low,@llight.high,@humidity.high,@distance.far,@threat.low;
140 @temp.medium,@smoke.low,@llight.high,@humidity.high,@distance.medium,@threat.medium;
141 @temp.medium,@smoke.low,@llight.high,@humidity.high,@distance.close,@threat.medium;
142
143 @temp.medium,@smoke.low,@llight.high,@humidity.medium,@distance.far,@threat.low;
144 @temp.medium,@smoke.low,@llight.high,@humidity.medium,@distance.medium,@threat.medium;
145 @temp.medium,@smoke.low,@llight.high,@humidity.medium,@distance.close,@threat.medium;
146
147 @temp.medium,@smoke.low,@llight.high,@humidity.low,@distance.far,@threat.low;
148 @temp.medium,@smoke.low,@llight.high,@humidity.low,@distance.medium,@threat.medium;
149 @temp.medium,@smoke.low,@llight.high,@humidity.low,@distance.close,@threat.high;
150
151 %temp.medium y smoke.medium
152 @temp.medium,@smoke.medium,@llight.low,@humidity.high,@distance.far,@threat.low;
153 @temp.medium,@smoke.medium,@llight.low,@humidity.high,@distance.medium,@threat.low;
154 @temp.medium,@smoke.medium,@llight.low,@humidity.high,@distance.close,@threat.medium;
155
156 @temp.medium,@smoke.medium,@llight.low,@humidity.medium,@distance.far,@threat.low;
157 @temp.medium,@smoke.medium,@llight.low,@humidity.medium,@distance.medium,@threat.low;
158 @temp.medium,@smoke.medium,@llight.low,@humidity.medium,@distance.close,@threat.medium;
```

```
159
160 @temp.medium,@smoke.medium,@llight.low,@humidity.low,@distance.far,@threat.low;
161 @temp.medium,@smoke.medium,@llight.low,@humidity.low,@distance.medium,@threat.medium;
162 @temp.medium,@smoke.medium,@llight.low,@humidity.low,@distance.close,@threat.medium;
163
164 @temp.medium,@smoke.medium,@llight.medium,@humidity.high,@distance.far,@threat.low;
165 @temp.medium,@smoke.medium,@llight.medium,@humidity.high,@distance.medium,@threat.medium;
166 @temp.medium,@smoke.medium,@llight.medium,@humidity.high,@distance.close,@threat.medium;
167
168 @temp.medium,@smoke.medium,@llight.medium,@humidity.medium,@distance.far,@threat.low;
169 @temp.medium,@smoke.medium,@llight.medium,@humidity.medium,@distance.medium,@threat.medium;
170 @temp.medium,@smoke.medium,@llight.medium,@humidity.medium,@distance.close,@threat.medium;
171
172 @temp.medium,@smoke.medium,@llight.medium,@humidity.low,@distance.far,@threat.medium;
173 @temp.medium,@smoke.medium,@llight.medium,@humidity.low,@distance.medium,@threat.medium;
174 @temp.medium,@smoke.medium,@llight.medium,@humidity.low,@distance.close,@threat.medium;
175
176 @temp.medium,@smoke.medium,@llight.high,@humidity.high,@distance.far,@threat.medium;
177 @temp.medium,@smoke.medium,@llight.high,@humidity.high,@distance.medium,@threat.medium;
178 @temp.medium,@smoke.medium,@llight.high,@humidity.high,@distance.close,@threat.high;
179
180 @temp.medium,@smoke.medium,@llight.high,@humidity.medium,@distance.far,@threat.medium;
181 @temp.medium,@smoke.medium,@llight.high,@humidity.medium,@distance.medium,@threat.medium;
182 @temp.medium,@smoke.medium,@llight.high,@humidity.medium,@distance.close,@threat.high;
183
184 @temp.medium,@smoke.medium,@llight.high,@humidity.low,@distance.far,@threat.medium;
185 @temp.medium,@smoke.medium,@llight.high,@humidity.low,@distance.medium,@threat.high;
186 @temp.medium,@smoke.medium,@llight.high,@humidity.low,@distance.close,@threat.high;
187
188 %temp.medium y smoke.high
189 @temp.medium,@smoke.high,@llight.low,@humidity.high,@distance.far,@threat.low;
190 @temp.medium,@smoke.high,@llight.low,@humidity.high,@distance.medium,@threat.medium;
191 @temp.medium,@smoke.high,@llight.low,@humidity.high,@distance.close,@threat.medium;
192
193 @temp.medium,@smoke.high,@llight.low,@humidity.medium,@distance.far,@threat.low;
194 @temp.medium,@smoke.high,@llight.low,@humidity.medium,@distance.medium,@threat.medium;
195 @temp.medium,@smoke.high,@llight.low,@humidity.medium,@distance.close,@threat.medium;
196
197 @temp.medium,@smoke.high,@llight.low,@humidity.low,@distance.far,@threat.medium;
198 @temp.medium,@smoke.high,@llight.low,@humidity.low,@distance.medium,@threat.medium;
199 @temp.medium,@smoke.high,@llight.low,@humidity.low,@distance.close,@threat.high;
200
201 @temp.medium,@smoke.high,@llight.medium,@humidity.high,@distance.far,@threat.medium;
202 @temp.medium,@smoke.high,@llight.medium,@humidity.high,@distance.medium,@threat.medium;
203 @temp.medium,@smoke.high,@llight.medium,@humidity.high,@distance.close,@threat.medium;
204
205 @temp.medium,@smoke.high,@llight.medium,@humidity.medium,@distance.far,@threat.medium;
206 @temp.medium,@smoke.high,@llight.medium,@humidity.medium,@distance.medium,@threat.medium;
207 @temp.medium,@smoke.high,@llight.medium,@humidity.medium,@distance.close,@threat.high;
208
209 @temp.medium,@smoke.high,@llight.medium,@humidity.low,@distance.far,@threat.medium;
210 @temp.medium,@smoke.high,@llight.medium,@humidity.low,@distance.medium,@threat.high;
211 @temp.medium,@smoke.high,@llight.medium,@humidity.low,@distance.close,@threat.high;
212
213 @temp.medium,@smoke.high,@llight.high,@humidity.high,@distance.far,@threat.medium;
214 @temp.medium,@smoke.high,@llight.high,@humidity.high,@distance.medium,@threat.medium;
215 @temp.medium,@smoke.high,@llight.high,@humidity.high,@distance.close,@threat.high;
216
217 @temp.medium,@smoke.high,@llight.high,@humidity.medium,@distance.far,@threat.medium;
218 @temp.medium,@smoke.high,@llight.high,@humidity.medium,@distance.medium,@threat.high;
219 @temp.medium,@smoke.high,@llight.high,@humidity.medium,@distance.close,@threat.high;
220
221 @temp.medium,@smoke.high,@llight.high,@humidity.low,@distance.far,@threat.high;
```

```
222 @temp.medium,@smoke.high,@llight.high,@humidity.low,@distance.medium,@threat.high;
223 @temp.medium,@smoke.high,@llight.high,@humidity.low,@distance.close,@threat.very_high;
224
225 %temp.high y smoke.low
226 @temp.high,@smoke.low,@llight.low,@humidity.high,@distance.far,@threat.low;
227 @temp.high,@smoke.low,@llight.low,@humidity.high,@distance.medium,@threat.low;
228 @temp.high,@smoke.low,@llight.low,@humidity.high,@distance.close,@threat.medium;
229
230 @temp.high,@smoke.low,@llight.low,@humidity.medium,@distance.far,@threat.low;
231 @temp.high,@smoke.low,@llight.low,@humidity.medium,@distance.medium,@threat.low;
232 @temp.high,@smoke.low,@llight.low,@humidity.medium,@distance.close,@threat.medium;
233
234 @temp.high,@smoke.low,@llight.low,@humidity.low,@distance.far,@threat.low;
235 @temp.high,@smoke.low,@llight.low,@humidity.low,@distance.medium,@threat.medium;
236 @temp.high,@smoke.low,@llight.low,@humidity.low,@distance.close,@threat.medium;
237
238 @temp.high,@smoke.low,@llight.medium,@humidity.high,@distance.far,@threat.low;
239 @temp.high,@smoke.low,@llight.medium,@humidity.high,@distance.medium,@threat.medium;
240 @temp.high,@smoke.low,@llight.medium,@humidity.high,@distance.close,@threat.medium;
241
242 @temp.high,@smoke.low,@llight.medium,@humidity.medium,@distance.far,@threat.low;
243 @temp.high,@smoke.low,@llight.medium,@humidity.medium,@distance.medium,@threat.medium;
244 @temp.high,@smoke.low,@llight.medium,@humidity.medium,@distance.close,@threat.medium;
245
246 @temp.high,@smoke.low,@llight.medium,@humidity.low,@distance.far,@threat.low;
247 @temp.high,@smoke.low,@llight.medium,@humidity.low,@distance.medium,@threat.medium;
248 @temp.high,@smoke.low,@llight.medium,@humidity.low,@distance.close,@threat.medium;
249
250 @temp.high,@smoke.low,@llight.high,@humidity.high,@distance.far,@threat.medium;
251 @temp.high,@smoke.low,@llight.high,@humidity.high,@distance.medium,@threat.medium;
252 @temp.high,@smoke.low,@llight.high,@humidity.high,@distance.close,@threat.medium;
253
254 @temp.high,@smoke.low,@llight.high,@humidity.medium,@distance.far,@threat.medium;
255 @temp.high,@smoke.low,@llight.high,@humidity.medium,@distance.medium,@threat.medium;
256 @temp.high,@smoke.low,@llight.high,@humidity.medium,@distance.close,@threat.high;
257
258 @temp.high,@smoke.low,@llight.high,@humidity.low,@distance.far,@threat.medium;
259 @temp.high,@smoke.low,@llight.high,@humidity.low,@distance.medium,@threat.high;
260 @temp.high,@smoke.low,@llight.high,@humidity.low,@distance.close,@threat.high;
261
262 %temp.high y smoke.medium
263 @temp.high,@smoke.medium,@llight.low,@humidity.high,@distance.far,@threat.low;
264 @temp.high,@smoke.medium,@llight.low,@humidity.high,@distance.medium,@threat.medium;
265 @temp.high,@smoke.medium,@llight.low,@humidity.high,@distance.close,@threat.medium;
266
267 @temp.high,@smoke.medium,@llight.low,@humidity.medium,@distance.far,@threat.low;
268 @temp.high,@smoke.medium,@llight.low,@humidity.medium,@distance.medium,@threat.medium;
269 @temp.high,@smoke.medium,@llight.low,@humidity.medium,@distance.close,@threat.medium;
270
271 @temp.high,@smoke.medium,@llight.low,@humidity.low,@distance.far,@threat.medium;
272 @temp.high,@smoke.medium,@llight.low,@humidity.low,@distance.medium,@threat.medium;
273 @temp.high,@smoke.medium,@llight.low,@humidity.low,@distance.close,@threat.high;
274
275 @temp.high,@smoke.medium,@llight.medium,@humidity.high,@distance.far,@threat.medium;
276 @temp.high,@smoke.medium,@llight.medium,@humidity.high,@distance.medium,@threat.medium;
277 @temp.high,@smoke.medium,@llight.medium,@humidity.high,@distance.close,@threat.high;
278
279 @temp.high,@smoke.medium,@llight.medium,@humidity.medium,@distance.far,@threat.medium;
280 @temp.high,@smoke.medium,@llight.medium,@humidity.medium,@distance.medium,@threat.high;
281 @temp.high,@smoke.medium,@llight.medium,@humidity.medium,@distance.close,@threat.high;
282
283 @temp.high,@smoke.medium,@llight.medium,@humidity.low,@distance.far,@threat.medium;
284 @temp.high,@smoke.medium,@llight.medium,@humidity.low,@distance.medium,@threat.high;
```

```
285 @temp.high,@smoke.medium,@llight.medium,@humidity.low,@distance.close,@threat.high;
286
287 @temp.high,@smoke.medium,@llight.high,@humidity.high,@distance.far,@threat.high;
288 @temp.high,@smoke.medium,@llight.high,@humidity.high,@distance.medium,@threat.high;
289 @temp.high,@smoke.medium,@llight.high,@humidity.high,@distance.close,@threat.high;
290
291 @temp.high,@smoke.medium,@llight.high,@humidity.medium,@distance.far,@threat.high;
292 @temp.high,@smoke.medium,@llight.high,@humidity.medium,@distance.medium,@threat.high;
293 @temp.high,@smoke.medium,@llight.high,@humidity.medium,@distance.close,@threat.very_high;
294
295 @temp.high,@smoke.medium,@llight.high,@humidity.low,@distance.far,@threat.high;
296 @temp.high,@smoke.medium,@llight.high,@humidity.low,@distance.medium,@threat.very_high;
297 @temp.high,@smoke.medium,@llight.high,@humidity.low,@distance.close,@threat.very_high;
298
299 %temp.high y smoke.high
300 @temp.high,@smoke.high,@llight.low,@humidity.high,@distance.far,@threat.medium;
301 @temp.high,@smoke.high,@llight.low,@humidity.high,@distance.medium,@threat.medium;
302 @temp.high,@smoke.high,@llight.low,@humidity.high,@distance.close,@threat.high;
303
304 @temp.high,@smoke.high,@llight.low,@humidity.medium,@distance.far,@threat.medium;
305 @temp.high,@smoke.high,@llight.low,@humidity.medium,@distance.medium,@threat.medium;
306 @temp.high,@smoke.high,@llight.low,@humidity.medium,@distance.close,@threat.high;
307
308 @temp.high,@smoke.high,@llight.low,@humidity.low,@distance.far,@threat.high;
309 @temp.high,@smoke.high,@llight.low,@humidity.low,@distance.medium,@threat.high;
310 @temp.high,@smoke.high,@llight.low,@humidity.low,@distance.close,@threat.high;
311
312 @temp.high,@smoke.high,@llight.medium,@humidity.high,@distance.far,@threat.medium;
313 @temp.high,@smoke.high,@llight.medium,@humidity.high,@distance.medium,@threat.high;
314 @temp.high,@smoke.high,@llight.medium,@humidity.high,@distance.close,@threat.high;
315
316 @temp.high,@smoke.high,@llight.medium,@humidity.medium,@distance.far,@threat.high;
317 @temp.high,@smoke.high,@llight.medium,@humidity.medium,@distance.medium,@threat.high;
318 @temp.high,@smoke.high,@llight.medium,@humidity.medium,@distance.close,@threat.high;
319
320 @temp.high,@smoke.high,@llight.medium,@humidity.low,@distance.far,@threat.high;
321 @temp.high,@smoke.high,@llight.medium,@humidity.low,@distance.medium,@threat.high;
322 @temp.high,@smoke.high,@llight.medium,@humidity.low,@distance.close,@threat.very_high;
323
324 @temp.high,@smoke.high,@llight.high,@humidity.high,@distance.far,@threat.high;
325 @temp.high,@smoke.high,@llight.high,@humidity.high,@distance.medium,@threat.high;
326 @temp.high,@smoke.high,@llight.high,@humidity.high,@distance.close,@threat.very_high;
327
328 @temp.high,@smoke.high,@llight.high,@humidity.medium,@distance.far,@threat.high;
329 @temp.high,@smoke.high,@llight.high,@humidity.medium,@distance.medium,@threat.very_high;
330 @temp.high,@smoke.high,@llight.high,@humidity.medium,@distance.close,@threat.very_high;
331
332 @temp.high,@smoke.high,@llight.high,@humidity.low,@distance.far,@threat.very_high;
333 @temp.high,@smoke.high,@llight.high,@humidity.low,@distance.medium,@threat.very_high;
334 @temp.high,@smoke.high,@llight.high,@humidity.low,@distance.close,@threat.very_high;
335 };
336
337 [m,n] = size(rules);
338
339 %Inicializar el conjunto de reglas
340 for i=1:m
341     R(i).n = i;
342     R(i).A = rules(i,1:n-1);
343     R(i).B = rules{i,n};
344 end
345 end
```

4.3.3. Método de Mamdani aplicado a la detección de incendios forestales

Con el código implementado hasta ahora ya es posible aplicar el método de Mamdani a la detección y determinación de riesgos de incendios forestales. Para ello se ha definido la función:

```
fire_detection_mamdani(temp, smoke, light, humidity, distance)
```

que recibe como parámetros las entradas escalares de temperatura, humo , etc. y devuelve el riesgo de incendio resultante al aplicar el método de Mamdani. Esta función además devuelve los valores defusificados de este conjunto de salida:

Código 4.20: Método de Mamdani aplicado a la determinación de riesgo de incendios forestales (`fire_detection_mamdani.m`)

```
% Método de Mamdani aplicado a la determinación de riesgos de incendios.
% Parámetros:
% temp: Temperatura (°C).
% smoke: Humo (ppm).
% light: Luz (lux).
% humidity: Humedad (ppm).
% distance: Distancia (m).
%
% Valores devueltos:
% B: Conjunto de salida.
% dc: Valor defusificado centroide.
% db: Valor defusificado bisector.
% dm: Valor defusificado media de máximos.
% ds: Valor defusificado menor de máximos.
% dl: Valor defusificado mayor de máximos.

function [B, dc, db, dm, ds, dl] = fire_detection_mamdani(temp, smoke, ...
    light, humidity, distance)
    % Añadir dependencias al path
    addpath('./lang_variables');
    addpath('../functions');

    % Universo de salida
    x_threat = threat.get_x();

    % Reglas
    R = fire_detection_rules();

    % Premisa
    fact = [temp,smoke,light,humidity,distance];

    % Aplicar método de Mamdani
    B = mamdani(R, fact, x_threat);

    % Defusificar
    dc = round(defuzz(x_threat, B, 'centroid'));
    db = round(defuzz(x_threat, B, 'bisector'));
    dm = round(defuzz(x_threat, B, 'mom'));
    ds = round(defuzz(x_threat, B, 'som'));
    dl = round(defuzz(x_threat, B, 'lom'));
end
```

Utilizando esta función se puede definir una nueva función:

```
fire_detection_mamdani_plot(temp, smoke, light, humidity, distance)
```

que acepta los mismos argumentos (y algunos parámetros opcionales más) y que además de aplicar el método de Mamdani sobre las entradas, genera una gráfica con los resultados obtenidos:

Código 4.21: Método de Mamdani y gráfica con resultados (`fire_detection_mamdani_plot.m`)

```
% Método de Mamdani aplicado a la determinación de riesgos de incendios.
% Esta función pinta la gráfica con los resultados y opcionalmente genera
% un fichero .tikz con dicha gráfica.
% Parámetros:
% temp: Temperatura (°C).
% smoke: Humo (ppm).
% light: Luz (lux).
% humidity: Humedad (ppm).
% distance: Distancia (m).
% varargin: Parámetros opcionales (ver fire_detection_plot_input_parser()).
%
% Valores devueltos:
% B: Conjunto de salida.
% dc: Valor defusificado centroide.
% db: Valor defusificado bisector.
% dm: Valor defusificado media de máximos.
% ds: Valor defusificado menor de máximos.
% dl: Valor defusificado mayor de máximos.

function [B, dc, db, dm, ds, dl] = fire_detection_mamdani_plot( temp, smoke, ...
    light, humidity, distance, varargin )
addpath('./lang_variables');
addpath('../functions');
addpath('../matlab2tikz');

% Parser de parámetros opcionales
p = fire_detection_plot_input_parser();
parse(p,varargin{:});

% Obtener los resultados con el método de Mamdani
[B, dc, db, dm, ds, dl] = fire_detection_mamdani(temp, smoke, light, ...
    humidity, distance);

% Pintar gráfico
alw = 0.75;     % AxesLineWidth
fsz = 9;       % Fontsize
lw = 1.2;       % LineWidth
msz = 9;        % MarkerSize

figure;
set(gca, 'FontSize', fsz, 'LineWidth', alw);
hold on;
hl = plot(threat.get_x(),B,'-k','LineWidth',lw,'MarkerSize',msz);
hdc = plot(dc,B(dc+1),'*b','LineWidth',lw,'MarkerSize',msz);
hdb = plot(db,B(db+1),'+g','LineWidth',lw,'MarkerSize',msz);
hdm = plot(dm,B(dm+1),'sr','LineWidth',lw,'MarkerSize',msz);
hds = plot(ds,B(ds+1),'vc','LineWidth',lw,'MarkerSize',msz);
hdl = plot(dl,B(dl+1),'^m','LineWidth',lw,'MarkerSize',msz);
hasbehavior(hl,'legend',false);
xlabel('Riesgo (%)');
```

```
49 if(p.Results.showTitle == true)
50     title( strcat('Temp: ', sprintf('%d',temp), ', Humo: ', ...
           sprintf('%d',smoke), ', Luz: ', sprintf('%d',light), ', Humedad: ', ...
           sprintf('%d',humidity), ', Distancia: ', sprintf('%d',distance)));
51 end
52 if(p.Results.showLegend == true)
53     legend(strcat('centroid: ',sprintf('%d',dc)),strcat('bisector: ...
           ',sprintf('%d',db)),strcat('mom: ',sprintf('%d',dm)),strcat('som: ...
           ',sprintf('%d',ds)),strcat('lom: ',sprintf('%d',dl)));
54 end
55 if(p.Results.exportTikz == true)
56     matlab2tikz( strcat('./output/mamdani/mamdani-','T', ...
           sprintf('%d',temp),'_S', sprintf('%d',smoke),'_L', ...
           sprintf('%d',light),'_H', sprintf('%d',humidity),'_D', ...
           sprintf('%d',distance),'.tikz'),'showInfo', false,'standalone', ...
           false,'height', '\figureheight', 'width', '\figurewidth');
57 end
58
59 end
```

Este método acepta los siguientes parámetros opcionales:

- `'showTitle'` (true/false): Mostrar el título en la gráfica generada.
- `'showLegend'` (true/false): Mostrar la leyenda en la gráfica generada.
- `'exportTikz'` (true/false): Exportar la gráfica generada a tikz.

Para obtener y validar estos parámetros opcionales se utiliza un parser de entradas, que es el método estándar de MATLAB. Este parser es creado por la función:

```
fire_detection_plot_input_parser()
```

(utilizada en la línea 26):

Código 4.22: Parser de parámetros opcionales (`fire_detection_plot_input_parser.m`)

```
1  % Parser de parámetros opcionales para las funciones ...
       fire_detection_mamdani_plot y fire_detection_interpolation_plot.
2  % Parámetros opcionales:
3  % showTitle = true/false: Mostrar el título de la gráfica
4  % showLegend = true/false: Mostrar la leyenda de la gráfica.
5  % exportTikz = true/false: Exportar la gráfica en formato tikz.
6
7  function p = fire_detection_plot_input_parser()
8  p = inputParser;
9  defaultShowTitle = true;
10 defaultShowLegend = true;
11 defaultExportTikz = false;
12
13 addOptional(p,'showTitle',defaultShowTitle);
14 addOptional(p,'showLegend',defaultShowLegend);
15 addOptional(p,'exportTikz',defaultExportTikz);
16 end
```

Esta función define el parser y los parámetros opcionales que acepta, así como sus valores por defecto.

La función `fire_detection_mamdani_plot()` se puede utilizar por ejemplo para generar las gráficas de la figura 3.2 ejecutando el script `fire_detection_mamdani_examples.m`:

Código 4.23: Ejemplos del método de Mamdani (`fire_detection_mamdani_examples.m`)

```
clearvars;
addpath('./lang_variables');
addpath('../functions');
addpath('../matlab2tikz');

% Parámetros para generar gráficas
showTitle = false;
showLegend = false;
exportTikz = true;

%Riesgo bajo
fire_detection_mamdani_plot(25,0,200,20,70, 'showTitle', showTitle, ...
    'showLegend', showLegend, 'exportTikz', exportTikz);

% Riesgo bajo-medio
fire_detection_mamdani_plot(30,20,500,50,40, 'showTitle', showTitle, ...
    'showLegend', showLegend, 'exportTikz', exportTikz);

% Riesgo medio
fire_detection_mamdani_plot(40,50,500,30,40, 'showTitle', showTitle, ...
    'showLegend', showLegend, 'exportTikz', exportTikz);

%Riesgo medio-alto
fire_detection_mamdani_plot(80,80,700,20,30, 'showTitle', showTitle, ...
    'showLegend', showLegend, 'exportTikz', exportTikz);

%Riesgo alto
fire_detection_mamdani_plot(100,90,900,10,20, 'showTitle', showTitle, ...
    'showLegend', showLegend, 'exportTikz', exportTikz);

%Riesgo muy alto
fire_detection_mamdani_plot(120,100,1000,10,10, 'showTitle', showTitle, ...
    'showLegend', showLegend, 'exportTikz', exportTikz);
```

4.3.4. Método de interpolación basado en índices de solapamiento aplicado a la detección de incendios forestales

Con las funciones implementadas también se puede aplicar el método de interpolación basado en índices de solapamiento a la determinación de riesgo de incendios forestales. Para ello se ha definido la función:

```
fire_detection_interpolation(O, T, M, t, s, l, h, d)
```

similar a la ya definida para Mamdani. Esta función, además de las entradas escalares de temperatura, humo, etc. (`t, s, l, h, d`) recibe el índice de solapamiento `O`, la t-norma `T` y la función de agregación `M` utilizadas en el método. Al igual que el método de Mamdani, esta función

devuelve el valor de riesgo como un conjunto difuso y sus correspondientes valores defusificados:

Código 4.24: Método de interpolación basado en índices de solapamiento aplicado a la determinación de riesgo de incendios forestales (`fire_detection_interpolation.m`)

```
% Método de interpolación basado en índices de solapamiento aplicado a la ...
    determinación de riesgos de incendios.
% Parámetros:
% O: Índice de solapamiento.
% T: T-norma.
% M: Función de agregación.
% t: Temperatura (ºC).
% s: Humo (ppm).
% l: Luz (lux).
% h: Humedad (ppm).
% d: Distancia (m).
%
% Valores devueltos:
% B: Conjunto de salida.
% dc: Valor defusificado centroide.
% db: Valor defusificado bisector.
% dm: Valor defusificado media de máximos.
% ds: Valor defusificado menor de máximos.
% dl: Valor defusificado mayor de máximos.

function  [B, dc, db, dm, ds, dl] = fire_detection_interpolation(O, T, M, t, ...
    s, l, h, d)
addpath('./lang_variables');
addpath('../functions');
addpath('../matlab2tikz');

% Universos de referencia
x_temp     = temp.get_x();
x_smoke    = smoke.get_x();
x_light    = llight.get_x();
x_humidity = humidity.get_x();
x_distance = distance.get_x();
x_threat   = threat.get_x();

% Reglas
R = fire_detection_rules();

% Premisa (difusificar entradas escalares)
fact(1) = singleton_fuzzifier(x_temp, t);
fact(2) = singleton_fuzzifier(x_smoke, s);
fact(3) = singleton_fuzzifier(x_light, l);
fact(4) = singleton_fuzzifier(x_humidity, h);
fact(5) = singleton_fuzzifier(x_distance, d);

% Aplicar el método de interpolación
B = hashed_interpolation( R, fact, x_threat, O , T, M);

% Defusificar
dc = round(defuzz(x_threat, B, 'centroid'));
db = round(defuzz(x_threat, B, 'bisector'));
dm = round(defuzz(x_threat, B, 'mom'));
ds = round(defuzz(x_threat, B, 'som'));
dl = round(defuzz(x_threat, B, 'lom'));

end
```

Al igual que en el método de Mamdani, se puede definir una función que utilice esta función para generar gráficas con los resultados. La función:

```
fire_detection_interpolation_plot(O, T, M, t, s, l, h, d)
```

se ha implementado en `fire_detection_interpolation_plot.m` de la siguiente forma:

Código 4.25: Método de interpolación y generación de gráficas con resultados (`fire_detection_interpolation_plot.m`)

```
% Método de interpolación basado en índices de solapamiento aplicado a la ...
    determinación de riesgos de incendios.
% Esta función pinta la gráfica con los resultados y opcionalmente genera
% un fichero .tikz con dicha gráfica.
% Parámetros:
% O: Índice de solapamiento.
% T: T-norma.
% M: Función de agregación.
% t: Temperatura (ºC).
% s: Humo (ppm).
% l: Luz (lux).
% h: Humedad (ppm).
% d: Distancia (m).
% varargin: Parámetros opcionales (ver fire_detection_plot_input_parser()).
%
% Valores devueltos:
% B: Conjunto de salida.
% dc: Valor defusificado centroide.
% db: Valor defusificado bisector.
% dm: Valor defusificado media de máximos.
% ds: Valor defusificado menor de máximos.
% dl: Valor defusificado mayor de máximos.

function [B, dc, db, dm, ds, dl] = fire_detection_interpolation_plot( O, T, ...
    M, t, s, l, h, d, varargin )
addpath('../matlab2tikz');

% Parser de parámetros opcionales
p = fire_detection_plot_input_parser();
parse(p,varargin{:});

% Obtener los resultados con el método de interpolación
[B, dc, db, dm, ds, dl] = fire_detection_interpolation(O.f, T, M, t, s, l, ...
    h, d);

% Pintar gráfico
alw = 0.75;     % AxesLineWidth
fsz = 9;       % Fontsize
lw = 1.2;       % LineWidth
msz = 9;        % MarkerSize

figure;
set(gca, 'FontSize', fsz, 'LineWidth', alw);
hold on;
hl = plot(threat.get_x(),B,'-k','LineWidth',lw,'MarkerSize',msz);
hdc = plot(dc,B(dc+1),'*b','LineWidth',lw,'MarkerSize',msz);
hdb = plot(db,B(db+1),'+g','LineWidth',lw,'MarkerSize',msz);
hdm = plot(dm,B(dm+1),'sr','LineWidth',lw,'MarkerSize',msz);
hds = plot(ds,B(ds+1),'vc','LineWidth',lw,'MarkerSize',msz);
hdl = plot(dl,B(dl+1),'^m','LineWidth',lw,'MarkerSize',msz);
```

```
48  hasbehavior(hl,'legend',false);
49  xlabel('Riesgo (%)');
50  if(p.Results.showTitle == true)
51      title(strcat('Temp: ',sprintf('%d',t),', Humo: ',sprintf('%d',s),', Luz: ...
            ',sprintf('%d',l),', Humedad: ',sprintf('%d',h),', Distancia: ...
            ',sprintf('%d',d)));
52  end
53  if(p.Results.showLegend == true)
54      legend(strcat('centroid: ',sprintf('%d',dc)),strcat('bisector: ...
            ',sprintf('%d',db)),strcat('mom: ',sprintf('%d',dm)),strcat('som: ...
            ',sprintf('%d',ds)),strcat('lom: ',sprintf('%d',dl)));
55  end
56  if(p.Results.exportTikz == true)
57      matlab2tikz(strcat('./output/interpolation/interpolation-', 'O-', ...
            O.name, '_T', func2str(T), '_M', func2str(M), '--','T', ...
            sprintf('%d',t),'_S', sprintf('%d',s), '_L', sprintf('%d',l), '_H', ...
            sprintf('%d',h), '_D', sprintf('%d',d),'.tikz'),'showInfo', ...
            false,'standalone', false,'height', '\figureheight', 'width', ...
            '\figurewidth');
58  end
59
60  end
```

Los parámetros opcionales de este método son los mismos que en el método de Mamdani ya que, de hecho, utiliza el mismo parser de entradas (código 4.22). Esta función se puede utilizar por ejemplo para generar las gráficas de la figura 3.3 ejecutando el script:

Código 4.26: Ejemplos del método de interpolación (`fire_detection_interpolation_examples.m`)

```
1   clearvars;
2   addpath('./lang_variables');
3   addpath('../functions');
4   addpath('../matlab2tikz');
5
6   % Parámetros para generar gráficas
7   showTitle = false;
8   showLegend = false;
9   exportTikz = true;
10
11  % Índice de solapamiento
12  Os.f = O.maxmin();
13  Os.name = 'Oz';
14
15  %Riesgo bajo
16  fire_detection_interpolation_plot(Os, @min, @mean, 25, 0, 200, 20, 70, ...
        'showTitle', showTitle, 'showLegend', showLegend, 'exportTikz', exportTikz);
17
18  % Riesgo bajo-medio
19  fire_detection_interpolation_plot(Os, @min, @mean, 30, 20, 500, 50, 40, ...
        'showTitle', showTitle, 'showLegend', showLegend, 'exportTikz', exportTikz);
20
21  % Riesgo medio
22  fire_detection_interpolation_plot(Os, @min, @mean, 40, 50, 500, 30, 40, ...
        'showTitle', showTitle, 'showLegend', showLegend, 'exportTikz', exportTikz);
23
24  %Riesgo medio-alto
25  fire_detection_interpolation_plot(Os, @min, @mean, 80, 80, 700, 20, 30, ...
        'showTitle', showTitle, 'showLegend', showLegend, 'exportTikz', exportTikz);
26
27  %Riesgo alto
```

```
28 fire_detection_interpolation_plot(Os, @min, @mean, 100, 90, 900, 10, 20, ...
       'showTitle', showTitle, 'showLegend', showLegend, 'exportTikz', exportTikz);
29
30 %Riesgo muy-alto
31 fire_detection_interpolation_plot(Os, @min, @mean, 120, 100, 1000, 10, 10, ...
       'showTitle', showTitle, 'showLegend', showLegend, 'exportTikz', exportTikz);
```

4.3.5. Comparativa de resultados obtenidos con el método de interpolación

En la sección 3.4.2 se ha presentado una comparativa de los resultados obtenidos al aplicar el método de interpolación basado en índices de solapamiento a unas entradas concretas utilizando diferentes combinaciones de t-normas e índices de solapamiento.

Para obtener estos resultados se ha implementado la función:

```
fire_detection_interpolation_comparison(t, s, l, h, d)
```

que aplica el método de interpolación con cada combinación de t-normas e índices de solapamiento a las entradas `t, s, l, h, d`:

Código 4.27: Comparativa de resultados obtenidos con el método de interpolación (`fire_detection_interpolation_comparison.m`)

```
1  % Comparativa de resultados obtenidos con diferentes combinaciones de
2  % t-normas e índices de solapamiento utilizando el método de interpolación
3  % aplicado a las entradas recibidas como parámetros.
4  %
5  % Parámetros:
6  % t: Temperatura (ºC).
7  % s: Humo (ppm).
8  % l: Luz (lux).
9  % h: Humedad (ppm).
10 % d: Distancia (m).
11 %
12 % Valor devuelto:
13 % Esta función no devuelve ningún valor, pero genera una tabla LaTeX con
14 % los resultados obtenidos y un fichero .tikz con las gráficas por cada t-norma,
15 % combinada con cada índice de solapamiento.
16 function fire_detection_interpolation_comparison( t, s, l, h, d)
17 addpath('./lang_variables');
18 addpath('../functions');
19
20 % Parámetros gráficas
21 alw = 0.75;     % AxesLineWidth
22 fsz = 9;       % Fontsize
23 lw = 1.2;       % LineWidth
24 msz = 9;        % MarkerSize
25
26 % Universo de salida
27 x_threat   = threat.get_x();
28
29 % Operador de agregación
30 M = @mean;
31
32 % Índices de solapamiento
```

```
33 Os(1).f = O.pi();      Os(1).name = 'Opi';       Os(1).latex_name = '$O_{\pi}$';
34 Os(2).f = O.avgmin(); Os(2).name = 'Oavgmin';  Os(2).latex_name = ...
       '$O_{avgmin}$';
35 Os(3).f = O.maxmin(); Os(3).name = 'Omaxmin';  Os(3).latex_name = '$O_{Z}$';
36 Os(4).f = O.sqrt();   Os(4).name = 'Osqrt';     Os(4).latex_name = ...
       '$O_{\sqrt{\text{ }}}$';
37 Os(5).f = O.sin();    Os(5).name = 'Osin';      Os(5).latex_name = '$O_{sin}$';
38
39 % T-normas
40 Ts(1).f = @prod; Ts(1).name = 'Producto';               Ts(1).latex_name = ...
       '$T_{prod}$';
41 Ts(2).f = @min;  Ts(2).name = 'Mínimo';                 Ts(2).latex_name = ...
       '$T_{min}$';
42 Ts(3).f = @geomean;  Ts(3).name = 'Media geométrica';   Ts(3).latex_name = ...
       '$T_{geo}$';
43 Ts(4).f = @harmmean;  Ts(4).name = 'Media armónica';    Ts(4).latex_name = ...
       '$T_{harm}$';
44 Ts(5).f = @sinmean;  Ts(5).name = 'Sinmean';           Ts(5).latex_name = ...
       '$T_{sin}$';
45 Ts(6).f = @einsteinmean;  Ts(6).name = 'Einstein mean'; Ts(6).latex_name = ...
       '$T_{einstein}$';
46
47 % Abrir fichero
48 fName = strcat('table-interpolation-comparison--', 'T', sprintf('%d',t), ...
       '_S', sprintf('%d',s), '_L', sprintf('%d',l), '_H', sprintf('%d',h), ...
       '_D', sprintf('%d',d), '.tex');
49 fid = fopen(fName,'w');
50 if fid<0
51     error 'No se ha podido abrir el fichero';
52 end
53
54 % Imprimir la cabecera de la tabla
55 fprintf(fid,'%s\r\n','\begin{longtable}{| c | c | c | c | c | c | c |}');
56 fprintf(fid,'%s\r\n','\hline');
57 fprintf(fid,'%s\r\n',' \multirow{2}{*}{\textbf{T-norma}} & ...
       \multirow{2}{*}{\textbf{Índice de solapamiento}} &  ...
       \multicolumn{5}{|c|}{\textbf{Riesgo (\%)}} \\ \cline{3-7}');
58 fprintf(fid,'%s\r\n','& & \textbf{cent.  (\textasteriskcentered)} & ...
       \textbf{bis. (+)} & \textbf{som ($\triangledown$)} & \textbf{mom ...
       ($\square$)} & \textbf{lom ($\vartriangle$)}  \\ \hline');
59 for j=1:length(Ts)
60     figure('name',strcat('T-norma: ', Ts(j).name));
61     set(gca, 'FontSize', fsz, 'LineWidth', alw);
62     for i=1:length(Os)
63         [Y,dc,db,dm,ds,dl] = fire_detection_interpolation(Os(i).f , Ts(j).f, ...
               M, t, s, l, h, d);
64
65         subplot(3,2,i);
66         plot(x_threat, Y, '-k', dc, Y(dc+1), '*b', db, Y(db+1), '+g', dm, ...
               Y(dm+1), 'sr', ds, Y(ds+1), 'vc', dl, Y(dl+1), '^m', 'LineWidth', ...
               lw);
67         %legend('Threat', strcat('centroid:', sprintf('%d', dc)), ...
               strcat('bisector:', sprintf('%d', db)), strcat('mom:', ...
               sprintf('%d', dm)), strcat('som:', sprintf('%d', ds)), ...
               strcat('lom:', sprintf('%d', dl)));
68         title(Os(i).name);
69
70         % Escribir fila
71         str = '';
72         if(i==1)
73             str = strcat(str,'\multirow{',sprintf('%d', ...
                   length(Os)),'}{*}{',Ts(j).latex_name,'} ');
74         end
```

```
75          str = strcat(str, '& ', Os(i).latex_name, ' & ', sprintf('%d',dc), ' ...
                & ', sprintf('%d',db), ' & ', sprintf('%d',ds), ' & ', ...
                sprintf('%d',dm), ' & ', sprintf('%d',dl),' \\ ');
76          if(i==length(Os) && j<length(Ts))
77              str = strcat(str, ' \hhline{|=|=|=|=|=|=|=|} ');
78          elseif(i==length(Os) && j==length(Ts))
79              str = strcat(str, ' \hline ');
80          else
81              str = strcat(str, ' \cline{2-7} ');
82          end
83          fprintf(fid,'%s\r\n',str);
84      end
85      matlab2tikz(strcat( ...
            './output/interpolation/interpolation-comparison-Tnorma-', ...
            func2str(Ts(j).f), '--T', sprintf('%d',t), '_S', sprintf('%d',s), ...
            '_L', sprintf('%d',l), '_H', sprintf('%d',h), '_D', sprintf('%d',d), ...
            '.tikz'),'showInfo', false, 'standalone', false, 'height', ...
            '\figureheight', 'width', '\figurewidth');
86  end
87  fprintf(fid,'%s\r\n','\end{longtable}');
88  fclose(fid);
89
90  end
```

Esta función genera también gráficas con los resultados obtenidos, que se imprimen por pantalla y en ficheros .tikz que se pueden importar después en LaTeX. Además vuelca los resultados en formato de tabla LaTeX, tal y como se puede ver en la sección 3.4.2. De hecho, para generar esos resultados es tan sencillo como ejecutar la siguiente orden desde la línea de comandos:

```
1  fire_detection_interpolation_comparison(30, 20, 500, 50, 40);
```

5 Conclusiones

En este trabajo se ha presentado un nuevo método de inferencia difusa basado en índices de solapamiento. En los sistemas difusos basados en reglas, las entradas del sistema se comparan con los valores de las variables lingüísticas definidas como antecedentes en dichas reglas. De alguna forma, hay que calcular "cómo de coincidentes" son las entradas a dichos antecedentes. Los índices de solapamiento dan, precisamente, una medida del solapamiento entre dos conjuntos difusos y, por tanto, basar un método de inferencia en ellos resulta algo natural.

La principal ventaja de este método, frente a otros métodos clásicos como el controlador de Mamdani, es que es una generalización del método de interpolación y permite la utilización y combinación de diferentes índices de solapamiento, t-normas y operadores de agregación. Una gran ventaja de este método es que está construido a partir de funciones bien conocidas y ampliamente estudiadas como las t-normas, los operadores de agregación y las funciones de solapamiento.

En la sección 3.4 se ha presentado una comparativa de los resultados obtenidos con el método basado en índices de solapamiento y el método clásico de Mamdani aplicados al caso práctico de la detección y determinación de riesgos ambientales. Como se puede comprobar en dicha comparativa, los resultados obtenidos con ambos métodos son intuitivamente correctos y en general bastante similares entre sí.

A pesar de ello, se puede comprobar que, dependiendo de las diferentes t-normas e índices de solapamiento elegidos, los resultados obtenidos pueden variar ligeramente. En general, el método de interpolación basado en índices de solapamiento produce mejores resultados, dado que la combinación convexa de índices de solapamiento es también un índice de solapamiento. En una aplicación real, el método de interpolación basado en índices de solapamiento tiene la ventaja de que permite probar diferentes combinaciones de funciones y utilizar aquellas que mejores resultados ofrezcan (comparando con los resultados esperados por un experto). Gracias al teorema de construcción de índices de solapamiento (ec. 1.18), se pueden probar multitud de índices diferentes, construidos a partir de funciones muy conocidas y estudiadas como los operadores de agregación y las funciones de solapamiento.

Otra ventaja del método de interpolación basado en índices de solapamiento es que, a pesar de proporcionar una gran flexibilidad y potencia, es un método relativamente sencillo de entender e implementar en cualquier lenguaje de programación (en este caso se ha implementado en MATLAB).

6 Líneas futuras

En este trabajo se ha presentado un nuevo método de interpolación basado en índices de solapamiento, propuesto en [1]. En ese mismo artículo se propone también un algoritmo similar basado en índices de solapamiento para problemas de clasificación. En dicho algoritmo se utilizan diferentes operadores de agregación para agregar ciertos conjuntos difusos obtenidos en el proceso de clasificación. Para obtener un resultado único, se utilizan funciones penalty [29] para determinar qué combinación de operadores de agregación produce resultados más similares a los valores agregados.

Esta misma idea se puede aplicar al método de interpolación basado en índices de solapamiento. Como se puede ver en la línea 7 del algoritmo 4, se utiliza un operador de agregación para agregar los conjuntos difusos obtenidos en cada regla. En este paso se podrían utilizar diferentes operadores de agregación y determinar mediante funciones penalty qué operador o combinación de operadores de agregación produce los resultados más similares a los conjuntos agregados. Sería interesante comprobar si esta aplicación de funciones penalty produce mejores resultados y compararlos con el método de interpolación "básico" presentado en este trabajo.

En el ámbito de los sistemas difusos basados en reglas existen otros métodos y técnicas dignas de mencionar y que sería interesante estudiar en futuros trabajos. Uno de los campos más importante e interesante es el *aprendizaje de reglas difusas*. Habitualmente los conjuntos de reglas se construyen utilizando conocimiento experto sobre el problema. Sin embargo, también es posible generar dicho conjunto de reglas mediante procesos de aprendizaje, similares a los de las redes neuronales. En estos procesos de aprendizaje se utilizan ejemplos numéricos en los que se indican las entradas y sus correspondientes salidas. A partir de estos ejemplos es posible generar reglas difusas que transforman las entradas del sistema en las salidas esperadas.

En [30] y [31] se presentan algunos métodos y técnicas para el aprendizaje de reglas difusas. Otro tipo de técnicas interesantes relacionadas con el aprendizaje de reglas son aquellos relacionadas con la reducción y optimización de los conjuntos de reglas aprendidos. Una propiedad de estos métodos de aprendizaje es que tienden a generar conjuntos de reglas "sobredimensionados", es decir, con reglas redundantes que pueden degradar el rendimiento del sistema difuso. Por ello existen multitud de estudios y trabajos relacionados con la reducción y optimización de conjuntos de reglas que pueden ser interesantes de cara a futuros trabajos.

Bibliografía

[1] H. Bustince, E. Hüllermeier, R. Mesiar, N. Pal, and A. Pradera, "Construction of overlap indexes from overlap functions and fuzzy rule-based systems." April 2013.

[2] H. B. Sola, J. Fernandez, R. Mesiar, J. Montero, and R. Orduna, "Overlap index, overlap functions and migrativity.," in *IFSA/EUSFLAT Conf.* (J. P. Carvalho, D. Dubois, U. Kaymak, and J. M. da Costa Sousa, eds.), pp. 300–305, 2009.

[3] P. Bolourchi and S. Uysal, "Forest fire detection in wireless sensor network using fuzzy logic," in *Computational Intelligence, Communication Systems and Networks (CICSyN), 2013 Fifth International Conference on*, pp. 83–87, June 2013.

[4] E. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1 – 13, 1975.

[5] L. Zadeh, "Fuzzy sets," *Information Control*, vol. 8, pp. 338–353, 1965.

[6] L. Wang, *A Course in Fuzzy Systems and Control*. Prentice Hall PTR, 1997.

[7] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning," *Journal of Information Science*, p. 199, 1975.

[8] D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*. Mathematics in science and engineering, Academic Press, 1980.

[9] E. Klement, R. Mesiar, and E. Pap, *Triangular Norms*. Trends in logic, Studia logica library, Springer, 2000.

[10] T. Calvo, G. Mayor, and R. Mesiar, *Aggregation Operators: New Trends and Applications*. Studies in Fuzziness and Soft Computing, Physica-Verlag HD, 2002.

[11] G. Beliakov, A. Pradera, and T. Calvo, *Aggregation Functions: A Guide for Practitioners*. Studies in fuzziness and soft computing, Springer London, Limited, 2007.

[12] H. Bustince, J. Fernandez, R. Mesiar, J. Montero, and R. Orduna, "Overlap functions," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 72, no. 3–4, pp. 1488 – 1499, 2010.

[13] H. Bustince, M. Pagola, R. Mesiar, E. Hullermeier, and F. Herrera, "Grouping, overlap, and generalized bientropic functions for fuzzy modeling of pairwise comparisons," *Fuzzy Systems, IEEE Transactions on*, vol. 20, pp. 405–415, June 2012.

[14] A. Jurio, H. Bustince, M. Pagola, A. Pradera, and R. Yager, "Some properties of overlap and grouping functions and their application to image thresholding," *Fuzzy Sets and Systems*, vol. 229, no. 0, pp. 69 – 90, 2013. Theme: Computer Science.

[15] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets Syst.*, vol. 100, pp. 9–34, Apr. 1999.

[16] D. Dubois, W. Ostasiewicz, and H. Prade, "Fuzzy sets: History and basic notions," in *Fundamentals of Fuzzy Sets* (D. Dubois and H. Prade, eds.), vol. 7 of *The Handbooks of Fuzzy Sets Series*, pp. 21–124, Springer US, 2000.

[17] L. Zadeh, "Making computers think like people: The term "fuzzy thinking" is pejorative when applied to humans, but fuzzy logic is an asset to machines in applications from expert systems to process control," *Spectrum, IEEE*, vol. 21, pp. 26–32, Aug 1984.

[18] G. Bojadziev and M. Bojadziev, *Fuzzy Sets, Fuzzy Logic, Applications*. Advances in fuzzy systems - applications and theory, World Scientific, 1995.

[19] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. SMC-15, pp. 116–132, Jan 1985.

[20] M. Sugeno and G. Kang, "Structure identification of fuzzy model," *Fuzzy Sets and Systems*, vol. 28, no. 1, pp. 15 – 33, 1988.

[21] H. Hellendoorn and C. Thomas, "Defuzzification in fuzzy controllers," *Journal of Intelligent and Fuzzy Systems*, vol. 1, no. 2, pp. 109–123, 1993.

[22] C.-C. Lee, "Fuzzy logic in control systems: fuzzy logic controller. ii," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 20, pp. 419–435, Mar 1990.

[23] P. M. Larsen, "Industrial applications of fuzzy logic control," *International Journal of Man-Machine Studies*, vol. 12, no. 1, pp. 3 – 10, 1980.

[24] L. Kóczy and K. Hirota, "Approximate reasoning by linear rule interpolation and general approximation," *International Journal of Approximate Reasoning*, vol. 9, no. 3, pp. 197 – 225, 1993.

[25] H.-M. Tsai, O. Tonguz, C. Saraydar, T. Talty, M. Ames, and A. Macdonald, "Zigbee-based intra-car wireless sensor networks: a case study," *Wireless Communications, IEEE*, vol. 14, pp. 67–77, December 2007.

[26] T. Chi, M. Chen, and Q. Gao, "Implementation and study of a greenhouse environment surveillance system based on wireless sensor network," in *Embedded Software and Systems Symposia, 2008. ICESS Symposia '08. International Conference on*, pp. 287–291, July 2008.

[27] J. Hwang, C. Shin, and H. Yoe, "Study on an agricultural environment monitoring server system using wireless sensor networks," *Sensors*, vol. 10, no. 12, pp. 11189–11211, 2010.

[28] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393 – 422, 2002.

[29] H. Bustince, E. Barrenechea, T. Calvo, S. James, and G. Beliakov, "Consensus in multi-expert decision making problems using penalty functions defined over a cartesian pro-

duct of lattices," *Information Fusion*, vol. 17, no. 0, pp. 56 – 64, 2014. Special Issue: Information fusion in consensus and decision making.

[30] R. Alcalá, J. Casillas, O. Cordón, F. Herrera, and S. J. I. Zwir, "Techniques for learning and tuning fuzzy rule-based systems for linguistic modeling and their application," in *In C. Leondes (Ed.), Knowledge*, pp. 889–941, Academic Press, 1999.

[31] M. Serrurier, D. Dubois, H. Prade, and T. Sudkamp, "Learning fuzzy rules with their implication operators," *Data & Knowledge Engineering*, vol. 60, no. 1, pp. 71 – 89, 2007. Intelligent Data Mining.