II. SISTEMAS NEURO-DIFUSOS

Los sistemas Neuro-difusos forman parte de una nueva tecnología de computación llamada computación flexible (*soft-computing*), la cual tiene su origen en la teoría de la Inteligencia Artificial. La computación flexible engloba un conjunto de técnicas que tienen en común la robustez en el manejo de la información imprecisa e incierta que existe en los problemas relacionados con el mundo real (por ejemplo: reconocimiento de formas, clasificación, toma de decisiones, etc.). En algunos casos, las técnicas de computación flexible pueden ser combinadas para aprovechar sus ventajas individuales; algunas de estas técnicas son:

- La Lógica difusa.
- Las Redes Neuronales.
- Los Algoritmos Evolutivos o Algoritmos Genéticos.
- La Teoría del Caos.
- La Teoría del Aprendizaje.
- El Razonamiento Aproximado.

En los siguientes apartados se explicarán brevemente las técnicas de Redes Neuronales, Lógica Difusa y la combinación de ambas, por ser las de principal interés para el desarrollo de esta tesis.

2.1 Introducción a las Redes Neuronales

Las Redes Neuronales Artificiales (RNAs) son la implementación en hardware y / o software de modelos matemáticos idealizados de las neuronas biológicas. Las neuronas artificiales son interconectadas unas a otras y son distribuidas en capas de tal forma que emulan en forma simple la estructura neuronal de un cerebro. Cada modelo de neurona es capaz de realizar algún tipo de procesamiento a partir de estímulos de entrada y ofrecer una respuesta, por lo que las RNA en conjunto funcionan como redes de computación paralelas y distribuidas similares a los sistemas cerebrales biológicos. Sin embargo, a diferencia de las computadoras

convencionales, las cuales son programadas para realizar tareas específicas, las redes neuronales artificiales, tal como los sistemas cerebrales biológicos, deben ser entrenadas.
En la siguiente tabla se presenta una comparación entre las computadoras Von Neumann, y las redes neuronales artificiales[1]:

Tabla 2.1 - Computadora Von Neumann vs Sistema Neuronal

	Computadora	**Red Neuronal Artificial**
Procesador	- Complejo - Alta velocidad - Uno o pocos	- Simple - Baja velocidad - Un gran número
Memoria	- Separada del procesador - Localizada - No es direccionable por el contenido.	- Integrada en el procesador distribuida - Direccionable por el contenido
Computación	- Centralizado - Secuencial - Programas Almacenados	- Distribuido Paralelo - Auto-Aprendizaje
Confiabilidad	- Muy vulnerable	- Robusta
Punto fuerte	- Manipulaciones numéricas y simbólicas	- Problemas de percepción
Ambiente operativo	- Bien definido - Bien limitado	- Pobremente definido - Ilimitado

2.1.1 La Neurona Biológica

El sistema nervioso humano, incluido el cerebro, consiste en la interconexión de millones y millones de pequeñas unidades celulares llamadas neuronas[2], por ejemplo, sólo la corteza cerebral humana, una capa larga y plana de neuronas de alrededor 2 a 3 milímetros de ancho con un área superficial de aproximadamente 2,200 cm^2, contiene alrededor de cien mil millones (10^{11}) de neuronas[1].

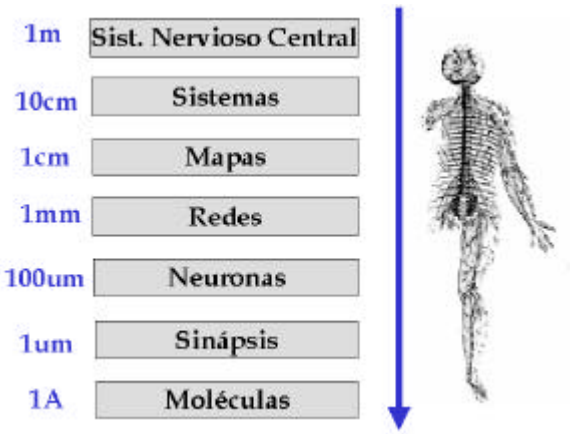

Figura 2.1 - Sistema Nervioso

En cuanto a su fisiología, las neuronas constan de un cuerpo celular o soma, en donde se aloja el núcleo de la célula. Del cuerpo de la célula salen ramificaciones de diversas fibras conocidas como dendritas y sale también una fibra más larga denominada axón. En su extremo, el axón también se ramifica en filamentos y sub-filamentos mediante los que establece conexión con las dendritas y los cuerpos de las células de otras neuronas; estas conexiones son conocidas como sinapsis. La comunicación entre neuronas ocurre como resultado de la liberación, por parte de la neurona pre-sináptica, de sustancias químicas con propiedades eléctricas llamadas neurotransmisores y la sub-secuente absorción de esas sustancias por parte de la neurona post-sináptica.

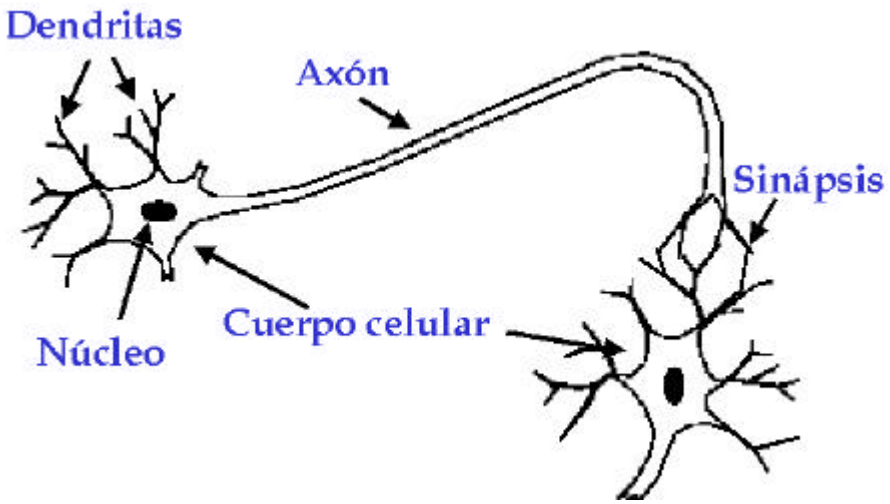

Figura 2.2 - Neurona biológica

Aunque la operación de una neurona biológica no está totalmente esclarecida, la mayor parte de los biólogos concuerdan en que la neurona después de recibir entradas a través de sus dendritas, las procesa en su soma, transmite el resultado del cálculo a través de su axón y utiliza las conexiones sinápticas para transportar la señal desde el axón a miles de dendritas pertenecientes a otras neuronas[3].

2.1.2 Componentes de una Red Neuronal Artificial

Ciertamente existen computadoras que tienen la habilidad de realizar procesamiento paralelo, pero el gran número de procesadores que serían necesarios para imitar el cerebro no puede ser alcanzado con la tecnología de hardware actual; además, la estructura interna de una computadora no puede ser cambiada mientras realiza una tarea[4]. Estos hechos conducen a la utilización de un modelo idealizado de neurona biológica elemental para propósitos de simulación.

Aunque existen diferentes tipos de Redes Neuronales Artificiales todos ellos tienen casi los mismos componentes elementales. Como en el sistema nervioso biológico, una red neuronal artificial está constituida por neuronas que están unidas entre sí a través de conexiones, a las

cuales se les asignan valores numéricos o pesos que representan el conocimiento de la Red Neuronal. Al cambiar los valores de los pesos se consigue imitar el cambio en la estructura de las conexiones sinápticas que ocurre durante el proceso de aprendizaje en la red neuronal biológica.

La siguiente figura muestra una neurona artificial idealizada.

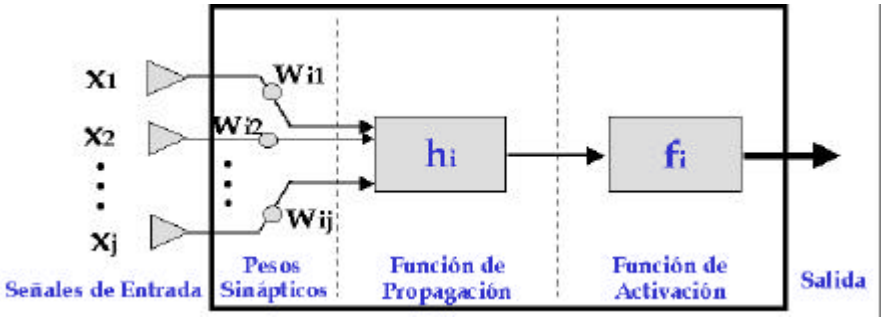

Figura 2.3 – Neurona Artificial

Como se puede observar en la figura 2.3 la neurona artificial luce similar a su contraparte biológica.

La información que conforma un conjunto de entradas Xj(t) es enviada a la neurona a través de sus conexiones con pesos sinápticos Wij, donde el subíndice i representa a la neurona i. Esta entrada es procesada por una función de propagación (por ejemplo: hi(t) = (Wij . Xj)). El resultado es comparado con un valor umbral θ por la función de activación yi(t) = fi(hi(t)) que representa simultáneamente la salida de la neurona y su estado de activación. Sólo si la entrada excede el valor umbral, la neurona se activará, en caso contrario se inhibirá.

En una RNA las neuronas suelen estar agrupadas en capas. Se conoce como capa o nivel a un conjunto de neuronas cuyas entradas provienen de la misma fuente, y cuyas salidas tienen el mismo destino. Usualmente cada neurona de una capa está conectada a todas las neuronas de las capas anterior y posterior (excepto en la capa de entrada y en la capa de salida)[5].

Las Redes Neuronales permiten resolver problemas que no pueden ser solucionados usando algoritmos convencionales. Tales son usualmente problemas de clasificación u optimización. Los diferentes dominios en los que las redes neuronales son utilizadas, incluyen:

- Asociación de patrones.
- Clasificación de patrones.
- Procesamiento de imágenes
- Reconocimiento de voz.

- Problemas de optimización.
- Simulación.

2.1.3 Tipos de Redes Neuronales

Existen diferentes tipos de redes neuronales y cada uno tiene características especiales, por lo que cada tipo de problema tiene su propio tipo de red neuronal para solucionarlo. Las redes neuronales pueden ser clasificadas según el tipo de aprendizaje, el tipo de aplicación y la arquitectura de la conexión. La siguiente tabla, presenta ésta clasificación.

Tabla 2.2 – Tipos de Redes Neuronales

Clasificación por el tipo de Aprendizaje	- Redes Neuronales con Aprendizaje Supervisado. - Redes Neuronales con Aprendizaje no-Supervisado. - Redes Neuronales con Aprendizaje Híbrido. - Redes Neuronales con Aprendizaje Reforzado. - Redes Neuronales con Aprendizaje Competitivo
Clasificación por el tipo de aplicación	- Redes Neuronales Aproximadoras de Funciones. - Redes Neuronales Asociativas o Memorias Asociativas. - Redes Neuronales Clasificadoras.
Clasificación por la Arquitectura de la conexión	- Redes Neuronales Monocapa. - Redes Neuronales Multicapa. - Redes Neuronales Realimentadas.

2.1.4 Arquitectura de Redes Neuronales Artificiales

Diferentes tipos de interconexión implican diferentes comportamientos de la red. Por ejemplo, las redes que tienen flujo de datos unidireccional (*feedforward*) son estáticas, es decir, producen sólo un grupo de valores de salida en lugar de una secuencia de valores de salida para una

entrada dada, además sus salidas no dependen de los valores anteriores de la red. Por otro lado las redes neuronales recurrentes o realimentadas son sistemas dinámicos[1]. Según la arquitectura de la conexión las redes neuronales se pueden clasificar, entre otras, como: Red Neuronal Monocapa, Red Neuronal Multicapa y Red Neuronal Realimentada.

Red Neuronal Monocapa

Las redes monocapa están formadas sólo por una capa de neuronas, y suelen utilizarse frecuentemente en tareas relacionadas con la regeneración de información incompleta o distorsionada que se presenta a la red.

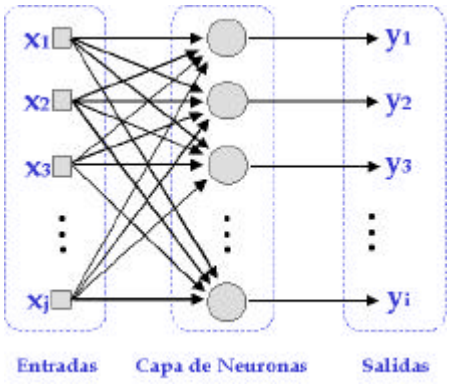

Figura 2.4 - Red Neuronal Monocapa

Red Neuronal Multicapa

Son las estructuras más comunes; como se puede apreciar en la figura 2.5, además de la capas de entrada y salida, poseen un número de capas intermedias u ocultas que mejoran su desempeño.

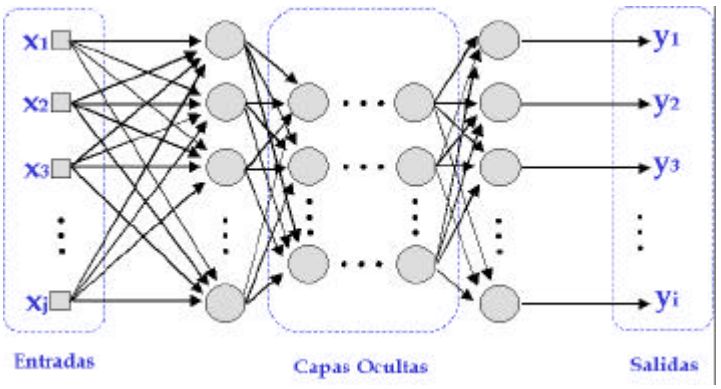

Figura 2.5 - Red Neuronal Multicapa

Red Neuronal Realimentada

Se caracteriza porque sus salidas pueden ser utilizadas como entradas. La estabilidad de la red es un importante factor a considerar en este tipo de arquitectura.

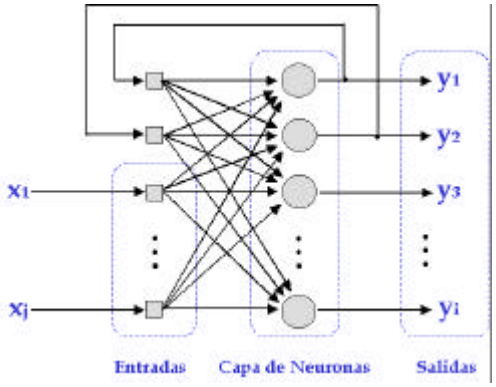

Figura 2.6 - Red Neuronal Realimentada

2.1.5 El Proceso de Aprendizaje

Como ya se mencionó, biológicamente se suele aceptar que la información memorizada en el cerebro está más relacionada con los valores sinápticos de las conexiones entre las neuronas que con las neuronas mismas, es decir, el conocimiento se encuentra en las sinapsis y todo proceso de aprendizaje consiste en la creación, modificación y destrucción de estas conexiones entre las neuronas. De forma similar el aprendizaje en las RNA consiste en determinar un conjunto de pesos sinápticos que permita a la red realizar correctamente el tipo de procesamiento deseado, esto se logra a través del entrenamiento de la red. Una vez que la red neuronal ha sido correctamente entrenada será capaz de determinar la salida deseada para las entradas que se le presenten.

2.1.5.1 Tipos de Aprendizaje

Los tipos de aprendizaje son los métodos utilizados para entrenar las RNAs, algunos de ellos son: el aprendizaje supervisado y el aprendizaje competitivo.

Aprendizaje Supervisado

En este caso un agente supervisor externo presenta a la red un conjunto de patrones característicos de entrada junto con la salida que se desea obtener e iterativamente la red ajusta sus pesos hasta que su salida tiende a ser la deseada; para realizar esta tarea la red utiliza información acerca del error que comete en cada paso de entrenamiento.

Aprendizaje Competitivo

En el aprendizaje competitivo las neuronas de salida compiten entre ellas para alcanzar el estado de activación. Como resultado sólo una unidad de salida estará activa en algún momento dado. Este procedimiento es conocido como WTA (*Winner-Take-all*).

2.1.5.2 Algoritmos de Aprendizaje

El tipo de algoritmo de aprendizaje depende esencialmente del tipo de aplicación de la red, así como de su topología. A continuación se describen los algoritmos de Corrección de Error y de Retro-propagación del Error.

Algoritmo de Corrección de Error

En el aprendizaje supervisado, se le da a la red una asociación de salidas-entradas. Durante el proceso de aprendizaje, la salida 'y' generada por la red puede no ser igual a la salida deseada 'd'. El aprendizaje por corrección de error consiste pues, en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores obtenidos de la red y los valores deseados, es decir, en función del error obtenido en la salida. Un algoritmo simple de aprendizaje por corrección de error podría ser el siguiente.

$$Wij = Wij + \alpha \, . \, x_j \, (d_i - y_i) \qquad (2.1)$$

Donde d_i es la salida deseada, y_i es la salida real de la neurona 'i' obtenida una iteración antes, x_j es una entrada j-ésima a la neurona 'i' y α es la tasa de aprendizaje de la red o constante de velocidad de aprendizaje de la red, el resultado Wij de este cálculo es el nuevo valor que será asignado al peso en la siguiente iteración. El principio esencial de los algoritmos de corrección

del error es usar la señal de error (d-y) para modificar los pesos en las conexiones y reducir gradualmente el error.

Algoritmo de Aprendizaje de Retro-propagación del Error

El algoritmo de aprendizaje que usa una Red Neuronal Multicapa es la Retro-propagación del Error. La importancia de este algoritmo radica en su capacidad de modificar los pesos de las neuronas de las capas intermedias de la red durante el entrenamiento.

La idea central de la retro-propagación del error es calcular los errores para las unidades de las capas ocultas a partir de los errores en las unidades de salida, para luego propagarlos capa tras capa hacia atrás hasta llegar a la capa de entrada, modificando los pesos de las neuronas en cada paso. El algoritmo debe ajustar los parámetros de la red para calcular el gradiente de error y minimizar el error medio cuadrático entre la salida deseada y la salida de la red[6].

La siguiente explicación del algoritmo de Retro-propagación se encuentra en la referencia [7] de la bibliografía, algunos de los subíndices han sido modificados para hacerlos congruentes con los utilizados en esta tesis.

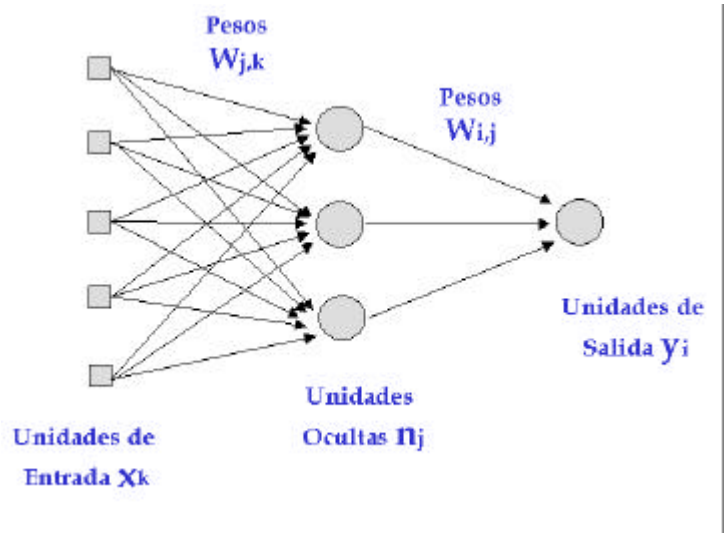

Figura 2.7 – Red Neuronal Multicapa

Existen dos diferencias con respecto al algoritmo de corrección de error visto anteriormente: en vez de un valor de entrada se utiliza la activación de la unidad oculta n_j como se muestra en la figura 2.7, y la ecuación contiene un término para el gradiente de la función de activación. Si E_{rri} es el error $(d_i - y_i)$ del nodo de salida, donde d_i es la salida deseada e y_i es la salida real, entonces la ecuación de actualización de los pesos del vínculo entre la unidad j y la unidad i es:

$$Wij = Wij + \alpha \cdot n_j \cdot E_{rri} \cdot f'(ent_i) \qquad (2.2)$$

En donde f' es la derivada de la función de activación h con entrada ent_i. Si se define el nuevo término de error Δ_i como: $\Delta_i = E_{rri} \cdot f'(ent_i)$. La ecuación de actualización de los pesos se convierte entonces en:

$$Wij = Wij + \alpha \cdot n_j \cdot \Delta_I \qquad (2.3)$$

Para actualizar las conexiones entre las unidades de entrada y las ocultas, hay que definir una cantidad análoga al término de error de los nodos de salida. Es en este momento cuando se realiza la propagación inversa del error. La idea es que el nodo oculto j es responsable de una parte del error Δ_i en cada uno de los nodos de salida con los que se conecta. Por lo tanto, los valores son Δ_i divididos de acuerdo con la intensidad de la conexión entre el nodo oculto y el nodo de salida, y se propagan hacia atrás para proporcionar los valores Δ_j del estado oculto. La regla de propagación de los valores Δ es la siguiente:

$$\Delta_j = f'(ent_j) \cdot \Sigma_i \, Wij \cdot \Delta_i \qquad (2.4)$$

Ahora bien, la regla de actualización de pesos correspondiente a los pesos que están entre las entradas y el nivel oculto es casi idéntica a la regla de actualización del nivel de salida:

$$Wj,k = Wj,k + \alpha \cdot x_k \cdot \Delta_j \qquad (2.5)$$

2.2 Introducción a los Sistemas Difusos

Los fundamentos de los sistemas difusos se encuentran en la lógica difusa. La lógica difusa o borrosa es una técnica de computación flexible que le permite a un computador clasificar información del mundo real en una escala infinita acotada por los valores falso y verdadero; tiene por objetivo proporcionar un soporte matemático formal al razonamiento basado en el lenguaje natural, el cual se caracteriza por tratarse de un razonamiento de tipo aproximado que hace uso de proposiciones que expresan información de carácter impreciso.

Algunas características de la lógica difusa que la hacen de tanto interés son[8]:

- Es fácil de entender, los conceptos matemáticos son bastante sencillos.
- Es flexible, su escalamiento es sencillo.
- Es tolerante a datos imprecisos.
- Puede modelar funciones no-lineales de complejidad arbitraria.
- Puede ser construida sobre la información de la experiencia de los operarios que manejan el sistema que se desea automatizar.
- Puede ser complementaria a las técnicas de control convencionales.
- Está basada en el lenguaje utilizado por los humanos.

La lógica difusa debe ser distinguida de la incertidumbre en el sentido en que la lógica difusa describe la ambigüedad de un evento, mientras que la incertidumbre la ocurrencia de un evento. Específicamente, el concepto incerteza se refiere a la incerteza estocástica, que se caracteriza por referirse a eventos bien definidos, por ejemplo,

La probabilidad de que el motor falle es de 80%

La lógica difusa trata con incertezas lingüísticas del tipo:

El motor falla constantemente.

Ambas sentencias son muy similares, sin embargo, existe una significativa diferencia en cuanto a la forma de expresar la probabilidad. Mientras en el caso de incerteza estocástica, la probabilidad es expresada en un sentido matemático, en una incerteza lingüística la probabilidad es percibida correctamente sin que haya sido cuantificada.

A continuación se explican algunos conceptos que son de gran utilidad para el estudio de la lógica difusa y la teoría de los conjuntos difusos:

Universo de Discurso

El universo de discurso denotado por U contiene todos los elementos que pueden ser tomados bajo consideración para asignar valores a las variables del sistema difuso. Los conjuntos difusos toman sus elementos del universo de discurso, por ejemplo el conjunto de gente joven podría tener a todos los seres humanos del mundo como su universo.

Función de Pertenencia

Todos los elementos dentro del universo de discurso U son miembros de algún conjunto difuso en cierto grado. La función de pertenencia es la curva que define con que grado cada elemento está incluido en el conjunto difuso. Para la definición de las funciones de pertenencia se utilizan formas estándar como la función triangular, trapezoidal, S, exponencial, singleton y π.

Variables Lingüísticas

Las variables lingüísticas son elementos fundamentales de cualquier sistema de lógica difusa. En ellas se combinan múltiples categorías subjetivas que describen el mismo concepto, así, para el caso de la variable *altura* existirán las categorías: *bajo*, *mediano*, *alto* y *muy alto*, que son llamadas términos lingüísticos y representan los posibles valores de una variable lingüística. En un lenguaje más formal, una variable lingüística se caracteriza básicamente por tres parámetros (x, T(x), U) donde x es el nombre de la variable, T(x) es el conjunto de términos lingüísticos de x, y U es el universo de discurso.

Grado de Pertenencia

Es el grado con el cual una entrada bien definida es compatible con una Función de Pertenencia, puede tomar valores entre 0 y 1. Por ejemplo, el Grado de Pertenencia de x al conjunto difuso *alto* (A) es representado por la función $\mu_A(x)$, donde x es un valor numérico de altura dentro del universo U ($x \in U$). El rango de μ es cualquier valor entre 0 y 1, según represente algún valor entre ningún o total grado de pertenencia al conjunto difuso.

Término

Es una categoría subjetiva de una variable lingüística, y consecuentemente, es el nombre descriptivo usado para identificar una función de pertenencia. Tal como las variables algebraicas toman valores numéricos, las variables lingüísticas toman como valores términos lingüísticos.

Entradas bien definidas (Entradas *Crisp*)

Son los diferentes valores discretos de la variable del sistema, por ejemplo las alturas medidas de un grupo de personas: 1.60m, 1.75m, 1.80m, etc. En oposición al concepto de difuso, lo *crisp*, definido, nítido o preciso no representa ninguna incerteza o imprecisión.

Rango/Dominio

Es el intervalo sobre el cual se define una Función de Pertenencia. Por ejemplo, una función de pertenencia Alto podría tener un dominio de 1.60 a 1.9 m y su rango sería de 0.3 m.

La figura 2.8 muestra gráficamente los conceptos que se acaban de revisar líneas arriba.

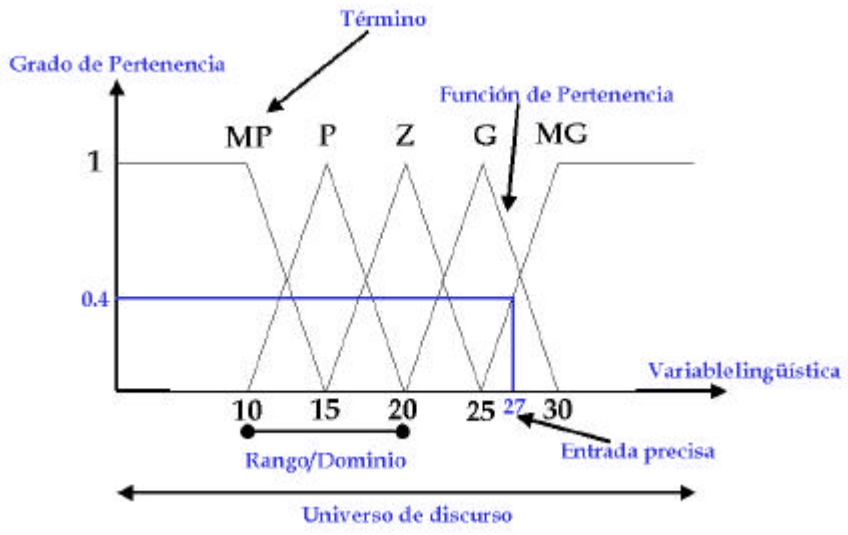

Figura 2.8 – Conceptos de lógica difusa

En las siguientes secciones se presentan algunos conceptos resaltantes de la teoría de los conjuntos difusos y la lógica difusa que son necesarios para analizar un Sistema Lógico Difuso (SLD).

2.2.1 Conjuntos Difusos

En teoría clásica de conjuntos, un conjunto tiene unos límites nítidos definidos (límites *crisp*). Por ejemplo, el conjunto A de los números más grandes que 8 se representa como:

$$A = \{ x / x > 8 \} \tag{2.6}$$

Sin embargo, según la teoría de los Conjuntos Difusos las vaguedades inherentes a los conceptos manejados por el ser humano, la transición desde "pertenecer a un conjunto" hasta "no pertenecer a un conjunto" es gradual. Así, un conjunto difuso (un conjunto sin un límite definido), contiene elementos sólo con un cierto grado de pertenencia.

A diferencia de los conjuntos clásicos que pueden ser caracterizadas ya sea por sus funciones de pertenencia, una descripción de sus elementos o un listado de sus elementos, los conjuntos difusos sólo pueden ser caracterizados por sus funciones de pertenencia; la única condición que una función de pertenencia debe satisfacer es que debe variar entre 0 y 1; la función en sí misma puede ser una curva arbitraria cuya forma la podemos definir como una función que sea agradable desde el punto de vista de simplicidad, conveniencia, velocidad y eficiencia[8]. Una forma particular para la función de pertenencia puede ser determinada sólo en el contexto de una aplicación particular pero resulta que muchas aplicaciones no son muy sensibles a las variaciones en la forma de sus funciones de pertenencia; en tales casos, es conveniente usar una forma simple, tal como la forma triangular[9]. La figura 2.9 presenta algunas funciones de pertenencia frecuentemente utilizadas:

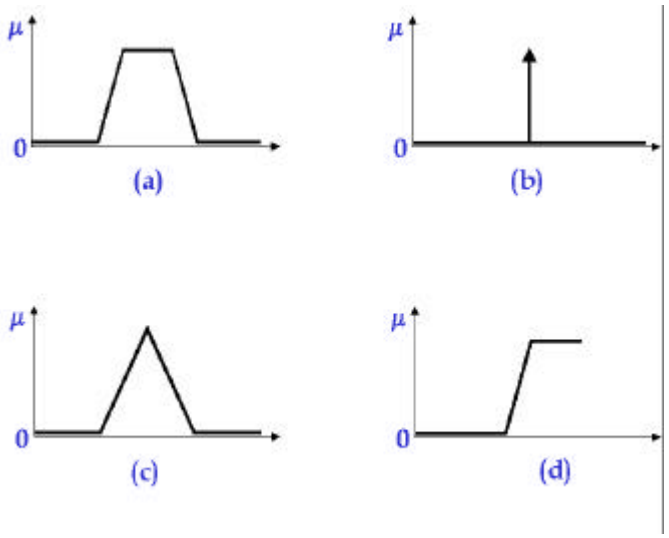

Figura 2.9 – Funciones de pertenencia. (a) Tipo trapezoidal. (b) Tipo singleton. (c) Tipo triangular. (d) Tipo z.

A los conjuntos difusos se les pueden aplicar determinados operadores, o bien puede realizarse operaciones entre ellos de la misma forma como se hace con los conjuntos crisp o clásicos. En general las funciones que definen las operaciones entre conjuntos difusos son arbitrarias en un grado sorprendente, lo que significa que uno puede crear sus propios operadores. Los operadores que califican como intersecciones difusas y uniones difusas son referidas como T-normas y T-conormas respectivamente.

Por ejemplo, sean A y B dos conjuntos difusos en U con funciones de pertenencia μ_A y μ_B respectivamente, es usual definir la T-norma min, la T-conorma max, y el complemento de la siguiente manera:

- T – norma min:

$$A \cap B = \mu_A(u) \min \mu_B(u)$$

- T – conorma max:

$$A \cup B = \mu_A(u) \max \mu_B(u)$$

- Complemento de A:

$$\mu_A = 1 - \mu_A(u)$$

Como lo demostró Zimmerman, los pares de T-normas y T-conormas satisfacen las propiedades conmutativas, asociativas y distributivas de los conjuntos clásicos así como las leyes de Morgan[10].

La bibliografía sobre lógica difusa contiene abundante información en cuanto a la Teoría de los Conjuntos Difusos, donde se abordan temas como relación y composición difusa como extensión de sus contrapartes clásicas.

2.2.2 Lógica Difusa

Los Sistemas Lógicos Difusos son sistemas basados en reglas, las cuales son expresadas como implicaciones lógicas, es decir, en forma de sentencias **SI-ENTONCES**. La implicación refleja la relación que guarda un hecho derivado de otro y pertenece a una rama de las matemáticas conocida como lógica.

La lógica, en su variante proposicional, tiene su elemento base en la proposición o "hecho". Una proposición es una sentencia que involucra términos definidos y debe ser apropiadamente llamada "verdadera" o "falsa", es decir hay hechos que se cumplen o no. Sin embargo,

haciendo una extensión de la teoría de conjuntos difusos, se puede decir que hay hechos que no son necesariamente ciertos o falsos, sino que tienen un cierto grado de verdad. Por ejemplo, "hoy hace calor" podría ser una oración válida en grado 0.6; así, lo que existe en el mundo no es un hecho, sino el grado de verdad de un hecho, y los seres humanos tiene un grado de certeza del hecho que puede variar entre 0 y 1(falso y verdadero).

Concretamente, se puede decir que una lógica consta de lo siguiente[7]:

- Un sistema formal para describir lo que está sucediendo en un momento determinado, y que consta de:
    - La sintaxis del lenguaje, que explica como construir oraciones
    - La semántica del lenguaje, a través de la cual cada oración expresa algo relacionado con el mundo.

- Una Teoría de demostración, que agrupe un conjunto de reglas para deducir las implicaciones de un conjunto de oraciones, y que especifique los pasos de razonamiento confiables.

Con el objetivo de mostrar la relación existente entre lógica proposicional y la lógica difusa, se presentan los conceptos referentes a sintaxis, semántica e inferencia relacionados a cada una de ellas.

Sintaxis

La sintaxis de la lógica proposicional está conformada por las constantes lógicas V (verdadero) y F (falso), los símbolos de las proposiciones tales como p y q, y los operadores lógicos conjunción ($\wedge$), disyunción (v), negación (~), equivalencia ($\leftrightarrow$) e implicación ($\rightarrow$). En lógica difusa, sin embargo, en vez de las constantes lógicas V y F se utilizan grados de verdad (grados de pertenencia a los conjuntos difusos) y los operadores lógicos son la negación difusa, las T-normas y las T-conormas.

Tabla 2.3 – Sintaxis de la lógica proposicional y de la lógica difusa

Elemento de Sintaxis	Lógica Clásica	Lógica Difusa
Constantes Lógicas	Verdadero (V) Falso (F)	Grado de verdad que varía entre 0 y 1
Símbolos de Proposiciones	p, q, r, etc.	p, q, r, etc.
Operadores Lógicos	Conjunción Disyunción Negación Implicación Equivalencia	T - normas T - conormas Negación difusa

Semántica

Al igual que en lógica proposicional, en lógica difusa, la semántica se define especificando la interpretación de los símbolos de proposición y de las constantes y especificando el significado de los conectores lógicos; a continuación se presenta una tabla de los valores semánticos de los elementos de la lógica difusa y de la lógica proposicional.

Tabla 2.4 – Semántica de la lógica proposicional y de la lógica difusa

Elemento	Significado Semántico	
	Lógica proposicional	**Lógica difusa**
p, q, r, etc	Un hecho. Ejm: Lima es una ciudad grande	Un grado de verdad. Ejm: Lima es una ciudad grande en un grado 0.6
Constantes lógicas	Verdadero: El hecho de la verdad Falso: No existe en el mundo	Certeza en cierto grado de verdad
Operadores lógicos	Y, no, o, etc.	Y, no, o, etc.

Inferencia

La inferencia lógica es un proceso mediante el cual se implanta la relación de implicación que existe entre oraciones[7]; es decir, si se cuenta con determinadas premisas y una posible conclusión, se puede averiguar si la conclusión es verdadera. Para ello, en lógica proposicional se construye una tabla de verdad para la oración *Premisas* → *conclusión* y se procede a verificarla.

El procedimiento que permite cerciorarse de la confiabilidad de una inferencia mediante tablas de verdad, podría emplearse a clases enteras de inferencias pues existen ciertos patrones que se presentan una y otra vez, lo que permite establecer a simple vista su confiabilidad. De esta manera se aprehende el patrón respectivo en algo que se conoce como regla de inferencia. Una vez establecida la regla, se le puede emplear para hacer más inferencias sin pasar por el tedioso procedimiento de la construcción de tablas de verdad. Se considera confiable una regla de inferencia si la conclusión es verdadera en todos aquellos casos en los que las premisas también son válidas. Algunas reglas de inferencia muy conocidas son: el Modus Ponens, el Modus Tolens, la Resolución, y la O-Introducción.

Si se desea aplicar la lógica proposicional a la ingeniería, la regla de inferencia que se ha de utilizar es el Modus Ponens, ya que satiface la Ley de Causa y Consecuencia[10].

Así:

Premisa 1 (Hecho): "x es A"
Premisa 2 (Conocimiento): "Si x es A, entonces y es B"
Conclusión: "y es B"

En lógica difusa se utiliza el Modus Ponens Generalizado, una extensión del Modus Ponens:

Premisa 1 (Hecho): "u es A*"
Premisa 2 (Conocimiento): "Si u es A, entonces v es B"
Consecuencia: "v es B*"

Al comparar el Modus Ponens y el Modus Ponens Generalizado para determinar sus principales diferencias, en este último A* no es necesariamente igual al antecedente A y B* no es necesariamente igual al consecuente B. En lógica proposicional una regla será disparada sólo si la primera premisa es exactamente igual al antecedente de esa regla, y el resultado de tal activación es el consecuente real de la regla; en lógica difusa sin embargo, una regla es activada mientras no haya un grado cero de similitud entre la premisa y el antecedente de la regla, y el resultado de tal activación es un consecuente que tiene un grado de similitud diferente de cero con respecto al consecuente de la regla. Este proceder puede ser entendido como razonamiento aproximado.

2.2.3 Sistemas Lógicos Difusos

En general, un Sistema Lógico Difuso (SLD) es un mapeo no-lineal de un vector de datos de entrada con una salida escalar, es decir mapea números con números[8]. La teoría de los conjuntos difusos y la lógica difusa establecen las especificaciones de este mapeo no-lineal único, en cuanto es capaz de manejar datos numéricos y conceptos lingüísticos simultáneamente. Los SLD han sido aplicados exitosamente en campos tales como el control automático, clasificación de datos, análisis de decisiones, sistemas expertos y visión por computadora.

2.2.3.1 Etapas de un Sistema Lógico Difuso

Un SLD consta de tres etapas:

- Fusificación
- Reglas de Evaluación
- Defusificación

El esquema se muestra en la figura 2.10.

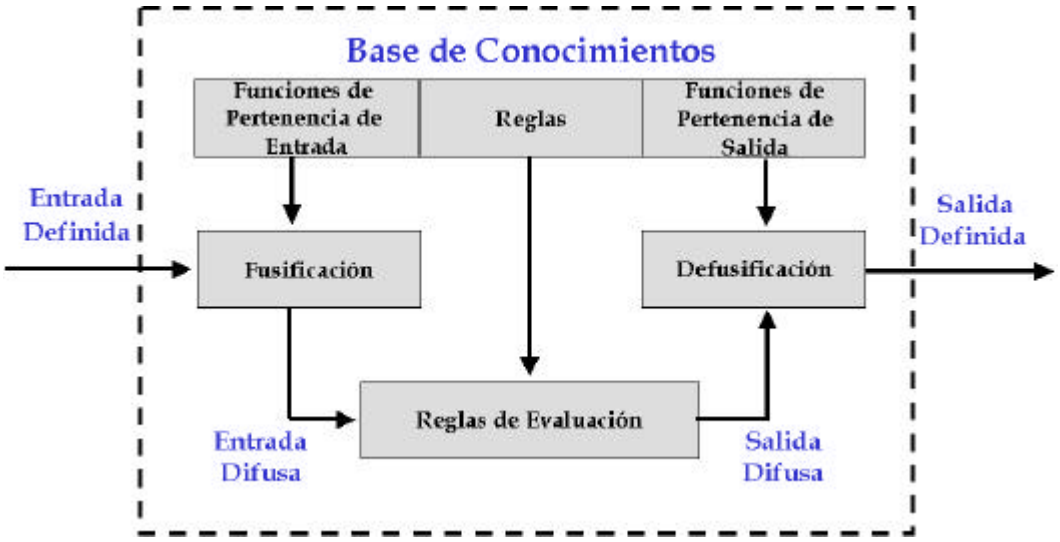

Figura 2.10 - Sistema Lógico Difuso

Fusificación

Los elementos fundamentales en esta etapa son las Funciones de Pertenencia de Entrada. La variable del proceso (entrada definida, no-difusa o *crisp*) intersecta las Funciones de Pertenencia generando las entradas difusas. Mediante este procedimiento, el fusificador establece una relación entre los puntos de entrada no difusos y sus correspondientes conjuntos difusos en el universo de discurso U.

Reglas de Evaluación

Las Reglas son sentencias **SI-ENTONCES** que describen las condiciones (antecedentes) y las acciones (consecuentes) que deben existir para tomar una decisión. La sintaxis de las reglas es la siguiente:

SI Antecedente 1 **Y** Antecedente 2 . . . **ENTONCES** Consecuente 1 **Y**. . .

El antecedente de una regla puede tener muchas partes, en tal caso todas las partes del antecedente son calculadas simultáneamente y transformadas en un número utilizando los

operadores lógicos descritos en la sección precedente. El consecuente de una regla puede también tener partes múltiples y todos los consecuentes son afectados de la misma manera por el antecedente.

En el caso de lógica proposicional, evaluar las reglas **SI-ENTONCES** es sencillo. Si la premisa es verdadera, la conclusión es verdadera, por el contrario, se genera un problema cuando el antecedente es difuso, pues no se sabe como se verá afectado el consecuente. Sin embargo, de acuerdo a la teoría revisada en la sección 2.2.2**,** si el antecedente es verdadero en cierto grado, entonces el consecuente es verdadero en cierto grado.

Así, las reglas difusas permiten expresar el conocimiento que se tiene acerca de la relación entre los antecedentes y los consecuentes en un cierto grado de verdad. Para expresar este conocimiento de forma completa normalmente se precisa de varias reglas, con pesos asociados, que se agrupan formando una base o bloque de reglas.

Mediante la inferencia los sistemas difusos interpretan las reglas de tipo IF-THEN contenidas en su base de conocimientos, con el fin de obtener los valores de salida a partir de los valores que tienen las variables lingüísticas de entrada al sistema. Una vez que las entradas crisp han sido convertidas a variables de valores lingüísticos (fusificación), se utiliza la inferencia difusa para identificar las reglas de tipo **SI-ENTONCES** que se aplican a la situación actual y se calculan los valores lingüísticos de salida.

La inferencia es un cálculo que consiste en tres pasos: agregación de las variables lingüísticas de entrada, composición o implicación y agregación del resultado.

- **Agregación de las variables lingüísticas de entrada**

  Es el primer paso que se sigue para realizar una inferencia difusa, la agregación determina el grado en el que se cumple la parte **SI** de la regla. Si la regla contiene varias premisas, estas suelen estar relacionadas por operadores lógicos difusos como T-normas y T-conormas.

- **Composición**

  Es el segundo paso que se lleva a cabo para realizar la inferencia, y es conocida también como implicación difusa. Mediante la composición se comprueba la validez de la conclusión de una regla al relacionar el grado con que se cumple el antecedente de la regla, con el peso de la misma.

- **Agregación del Resultado**

  Ya que las decisiones están basadas en la prueba de todas las reglas que forman un sistema de inferencia difuso los consecuentes de las reglas deben ser combinados de alguna manera para tomar una decisión. La agregación es el proceso a través del cual los conjuntos difusos que representan las salidas de las reglas son combinados en un único conjunto difuso.

Defusificación

La salida del proceso de inferencia es hasta ahora un conjunto difuso que indica la posibilidad de realizar una acción de control. Sin embargo, las aplicaciones de los sistemas difusos no pueden interpretar los valores lingüísticos obtenidos, por lo que funciones de pertenencia de salida son utilizadas para retransformar los valores difusos nuevamente en valores definidos o *crisp* mediante la defusificación.

Algunos métodos de defusificación existentes son:

- Procedimiento Máximo
- Media Ponderada
- Singleton
- Centro de Masa
- Centro de Área

El método de defusificación utilizado en el controlador Neuro-difuso que propone esta tesis es el Centro de Area. Este método consiste en hallar para cada conjunto difuso su área (A) y centro (x); se suma el producto entre ellos y el resultado se divide entre la sumatoria total de las áreas como se muestra en la siguiente ecuación.

$$\text{Valor defusificado} = \bar{x} = \frac{\sum A_i \bar{x}_i}{\sum A_i} \tag{2.7}$$

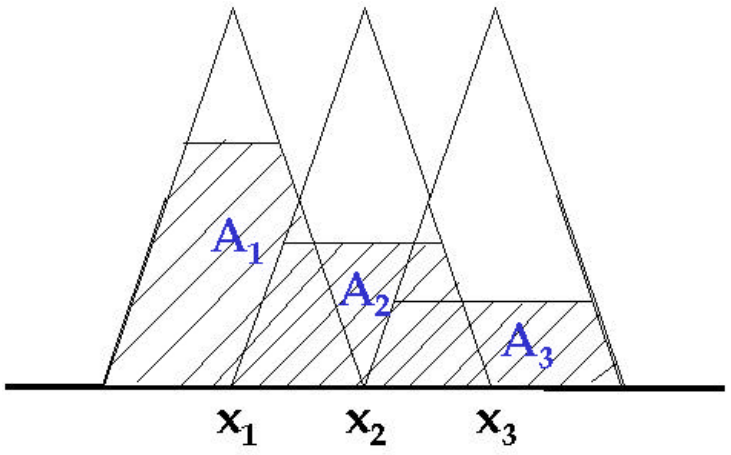

Figura 2.11 – Defusificación por Centro de Area

2.2.4 Sistemas Difusos aplicados al Control de Procesos

Desde las primeras investigaciones de Mamdani sobre una aplicación práctica para el control lógico difuso, los sistemas expertos de control difuso basados en reglas conocidos como controladores lógicos difusos o FLC (*Fuzzy Logic Controllers*) se han convertido sin duda en la aplicación más extendida de la lógica difusa[2].

El control lógico difuso provee un mecanismo para convertir una estrategia lingüística de control basada en el conocimiento de un operador experto en una estrategia de control automático. En este sentido, el método de control lógico difuso puede ser considerado como una aproximación entre la matemática convencional precisa y las tomas de decisiones de los humanos. Así, el lenguaje del mundo real usado en control difuso permite a los programadores incorporar la lógica ambigua de los humanos dentro de la computadora. Además, el uso de modelos lingüísticos en lugar de modelos matemáticos mejora grandemente la transparencia del sistema y facilita las potenciales modificaciones. Los sistemas difusos pueden ser utilizados en combinación con los controladores clásicos de la Teoría de Control para lo cual existen

diversos métodos; así, además del controlador difuso directo modelado líneas arriba, algunos métodos son[11]:

- Controlador Difuso Supervisorio
- Controlador Difuso Adaptativo
- Controlador Difuso de Intervención

2.3 Introducción a los Sistemas Neuro-difusos

Como se ha visto a lo largo del presente trabajo, la Lógica Difusa y las Redes Neuronales tienen propiedades computacionales particulares que las hacen adecuadas para ciertos problemas particulares y no para otros. Por ejemplo, mientras las redes neuronales ofrecen ventajas como el aprendizaje, adaptación, tolerancia a fallas, paralelismo y generalización, no son buenas para explicar como han alcanzado sus decisiones. En cambio los sistemas difusos, los cuales razonan con información imprecisa a través de un mecanismo de inferencia bajo incertidumbre lingüística, son buenos explicando sus decisiones pero no pueden adquirir automáticamente las reglas que usan para tomarlas.

Tabla 2.5 – Lógica Difusa y Redes Neuronales

Lógica Difusa	Redes Neuronales
Permite utilizar el conocimiento disponible para optimizar el sistema directamente.	No existe un método sencillo que permita modificar u optimizar la red, ya que esta se comporta como una "caja negra"
Permite describir el comportamiento de un sistema a partir de sentencias "si – entonces"	La selección del modelo apropiado de red y el algoritmo de entrenamiento requiere de mucha experiencia
Permite utilizar el conocimiento de un experto	Permite hallar soluciones a partir de un conjunto de datos
El conocimiento es estático	Son capaces de aprender y auto-adaptarse
Existen muchas aplicaciones comerciales	Su aplicación es mayormente académica
Permiten encontrar soluciones sencillas con menor tiempo de diseño	Requieren un enorme esfuerzo computacional

Los sistemas Neuro-Difusos combinan la capacidad de aprendizaje de las RNAs con el poder de interpretación lingüística de los sistemas de inferencia difusos[12], obteniéndose los siguientes resultados:

- Aplicabilidad de los algoritmos de aprendizaje desarrollados para redes neuronales.
- Posibilidad de promover la integración de conocimiento (implícito que puede ser adquirido a través del aprendizaje y explícito que puede ser explicado y entendido).
- La posibilidad de extraer conocimiento para una base de reglas difusas a partir de un conjunto de datos.

Existen sistemas de desarrollo que han logrado unir la Lógica Difusa con las Redes Neuronales, por ejemplo se tiene:

ANFIS

ANFIS (*Adaptive Neuro Fuzzy Inference System*) es un método que permite sintonizar o crear la base de reglas de un sistema difuso, utilizando el algoritmo de entrenamiento de retro-propagación a partir de la recopilación de datos de un proceso. Su arquitectura es funcionalmente equivalente a una base de reglas tipo Sugeno[13].

FSOM

FSOM (*Fuzzy Self-Organizing Maps*) consiste en un sistema difuso optimizado a partir de la técnica de los mapas auto-organizados de Kohonen.

NEFCLASS

El algoritmo NEFCLASS está basado en la estructura del perceptrón multicapa cuyos pesos son modelados por conjuntos difusos. Así, se preserva la estructura de una red neuronal, pero se permite la interpretación del sistema resultante por el sistema difuso asociado, es decir, la RNA deja de ser una "caja negra".

FuzzyTech

Es un software que propone un método de desarrollo de sistemas Neuro-difuso similar a ANFIS. FuzzyTECH es el sistema de desarrollo utilizado en esta tesis.

2.3.1 Aprendizaje en un Sistema Neuro-difuso

El Sistema Neuro-difuso consiste de un sistema difuso tradicional (ver figura 2.13) excepto que cada etapa, puede ser representada por una capa de neuronas a las que se puede proveer capacidades de aprendizaje de Redes Neuronales para optimizar el conocimiento del sistema como muestra la figura 2.12.

Figura 2.12 – Diagrama de bloques de un Sistema Neuro-difuso

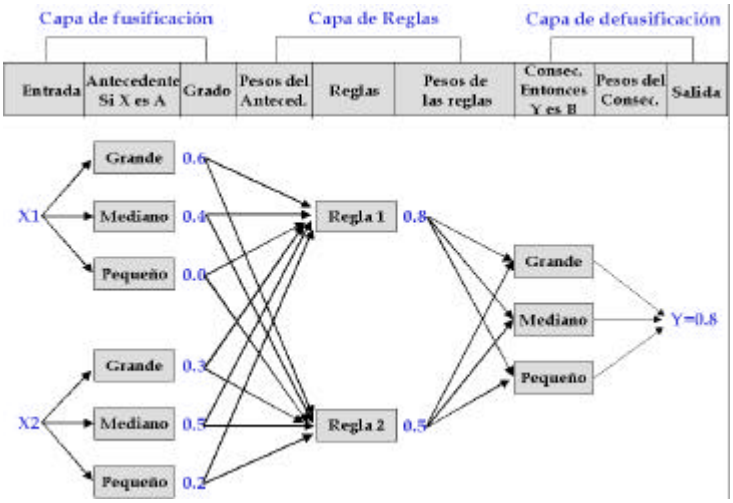

Figura 2.13 – Estructura de un Sistema Neuro-difuso

En la figura 2.13 cada pequeño rectángulo representa una neurona.

En la capa de fusificación, cada función de pertenencia de entrada del antecedente de una regla difusa representa una neurona. Los parámetros de estas neuronas, como los vértices de las funciones de pertenencia, pueden ser entrenados para determinar la forma final y la ubicación de las funciones de pertenencia.

En la figura 2.13 el grado de pertenencia que indica la certeza de "X1 es grande" es 0.6, "X1 es mediano" es 0.4, y "X1 es pequeño" es 0.0. Las salidas de estas neuronas funciones de pertenencia son conectadas a la capa de reglas difusas como lo especifiquen las reglas difusas y a través de enlaces con pesos que representan el proceso de agregación de las variable lingüísticas de entrada.

La capa de reglas difusas representa la base de reglas difusas; cada neurona representa una regla difusa de tipo SI-ENTONCES. Las salidas de las neuronas están conectadas a la capa de defusificación a través de enlaces con pesos; los pesos de estos enlaces representan la significancia relativa de las reglas asociadas con las neuronas. Sus valores pueden ser asignados de acuerdo al conocimiento a priori o inicializados como 1.0 y luego entrenadas para reflejar su importancia real para las funciones de pertenencia de salida contenidas en la capa de defusificación.

La función de la capa de defusificación es la evaluación de las reglas; en este cada consecuente "Entonces Y es B" como función de pertenencia de salida representa una neurona; la certeza de cada consecuente es calculada, y es considerada como lo bien que se ajustan las reglas que tienen el mismo consecuente (proceso de agregación del resultado). Los pesos de cada enlace de salida de estas neuronas representan los centros de área de cada función de pertenencia del consecuente y son entrenables, la salida final es entonces calculada usando algún método de defusificación.

Para realizar el entrenamiento de los sistemas neuro-difusos la estructura de la figura 2.13 puede ser configurada con valores iniciales obtenidos del conocimiento a priori, y luego, sintonizados utilizando un algoritmo de entrenamiento tal como Retro-propagación del Error, de la siguiente manera:

Paso 1: Presentar una muestra de entrada, y computar la salida correspondiente

Paso 2: Computar el error entre la salida y el valor objetivo

Paso 3: Se ajustan los pesos de conexión y las funciones de pertenencia

Paso 4: Si el error es mayor que la tolerancia, volver al paso 2, si no es así, el entrenamiento ha sido finalizado.

2.3.2 FuzzyTECH

Fuzzy TECH es un entorno de desarrollo para sistemas difusos basado en MS-Windows que permite el diseño de un sistema difuso gráficamente y la optimización del mismo utilizando un entrenamiento basado en redes neuronales. El desarrollo de un sistema Neuro-difuso en fuzzy TECH sigue los siguientes pasos[14]:

Paso 1: Obtener datos de entrenamiento, utilizar algún algoritmo de agrupamiento de datos si es necesario.

Paso 2: Crear un sistema lógico difuso vacío.

Paso 3: Ingresar en el sistema todo el conocimiento a priori existente para la solución.

Paso 4: Abrir los componentes del sistema lógico difuso que deberán ser entrenados.

Paso 5: Establecer los parámetros del modo Neurofuzzy

Paso 6: Entrenar con los datos muestra.

Paso 8: Evaluar la performance del sistema

Paso 9: Optimización manual.

A continuación se describen brevemente algunas características y herramientas que ofrece este programa pues es el entorno de desarrollo utilizado en la presente tesis.

2.3.2.1 Agrupamiento de Datos (*Clustering Module*)

La tecnología Neuro-difusa provee un método poderoso para convertir datos experimentales en reglas difusas. Sin embargo, en muchos casos, los datos de entrenamiento deben ser agrupados previamente, esto es necesario por dos razones:

- Eliminar los datos redundantes para alcanzar la convergencia rápidamente.
- Resolver conflictos en los datos en casos en que el conjunto de ellos sea inconsistente.

El agrupamiento o clustering es una clasificación de datos en grupos (clusters) de acuerdo a un cierto criterio de similitud[15]; por lo general, esto se logra maximizando la similitud intra-grupal y minimizando la similitud extra-grupal[16]. La similitud de los datos está especificada por la precisión (*accuracy*) de cada variable[14].

Con el fin de ilustrar la función de un algoritmo de agrupamiento (*clustering*) de datos, se presenta la Figura 2.14 en la que se grafican los valores de datos en el caso de una entrada y una única salida ya que es mucho más fácil identificar clusters en espacios de datos de dos dimensiones que en espacios dimensionales más grandes.

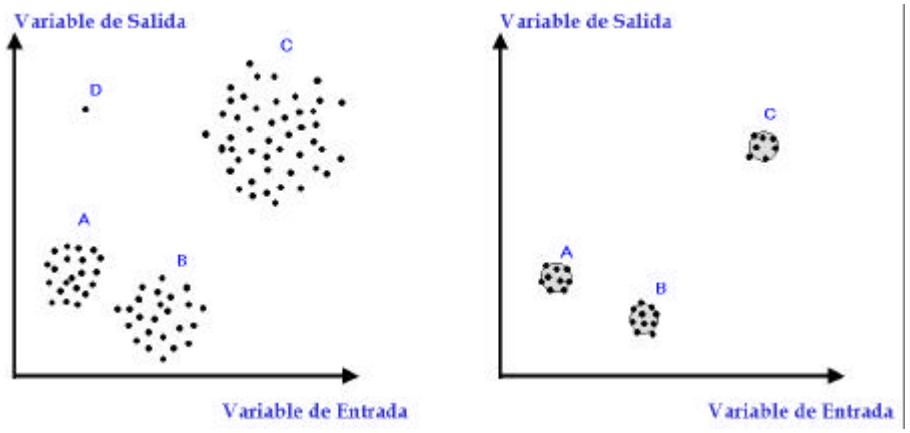

Figura 2.14 – Clusterización o agrupamiento de datos

Como se muestra en la figura 2.14, la mayor parte de los datos se agrupan con otros en tres 'manchas'; estas manchas, son los clusters o grupos. Un dato en particular del cluster puede reemplazar al grupo para disminuir el número de datos; este es llamado dato característico o representativo del grupo; así, como resultado se obtiene un conjunto de valores característicos, que representan al conjunto de datos inicial.

FuzzyTECH utiliza dos métodos para realizar el agrupamiento de datos, uno es el IsodataCluster, y el otro es el FuzzyCluster, ambos se describen a continuación.

IsodataCluster

El algoritmo Isodata es una técnica de agrupamiento de datos muy popular. Está basado en la elección aleatoria de k centros de cluster iniciales que son actualizados de tal forma que después de un número de ciclos representan al resto de los datos tan bien como puede ser posible. Isodata es iterativo, y sigue los siguientes pasos:

- Se inicia con la asignación aleatoria de k centros de clusters iniciales
- Asigna cada dato al cluster más cercano
- Se calculan los nuevos centros de los clusters que pueden ser un número diferente a k. El cálculo se hace a partir de todos los datos individuales del cluster
- El segundo y tercer paso son repetidos hasta que el "cambio" entre las iteraciones sea pequeño. El cambio puede ser definido de varias formas, ya sea midiendo las distancias que el centro del cluster ha cambiado de una iteración a otra, o por el porcentaje de datos que han cambiado entre las iteraciones.

La función objetivo del algoritmo Isodata que debe ser minimizada es la suma de las distancias cuadradas entre cada dato y su centro de cluster asignado para minimizar la variabilidad dentro del cluster.

$$SS_{dist} = \sum_{\forall x} [x - C(x)]^2 \qquad (2.8)$$

donde C(x) es el centro del cluster al que el dato x está asignado. El algoritmo Isodata es muy sensible a los valores centrales iniciales de los clusters, así, es posible que dos clasificaciones del mismo grupo de datos que se inicien con valores iniciales diferentes terminen siendo clasificaciones diferentes.

FuzzyCluster

El algoritmo FuzzyCluster, como todo algoritmo difuso, no considera límites definidos (*crisp*) para determinar si un dato pertenece a un cluster o no; asocia cada variable con una función de pertenencia que determina que tan similares son dos puntos de datos, dependiendo de los

valores de los parámetro delta y épsilon de la función de pertenencia (ver figura 2.15). A diferencia del Isodata cluster, la similitud de dos puntos de datos para una variable v es asignada como un grado de verdad μv. La distancia d_{xy} entre dos puntos es entonces computada como el mínimo negativo de las similitudes ($d_{xy} = -\min_v \{\mu v\}$). Los dos puntos que tienen el menor d_{xy} son considerados cercanos[14].

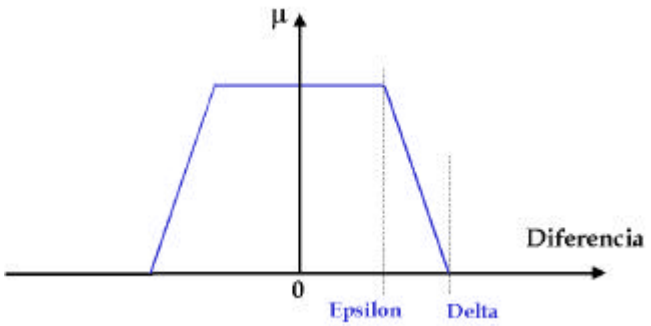

Figura 2.15 – Función de pertenencia para expresar la similaridad entre datos[14]

La interpretación de epsilon y delta en la figura 2.15 es la siguiente:

- Epsilon es la más grande diferencia en valor de dos puntos de datos que todavía se consideran similares.
- Delta es la diferencia más pequeña en el valor de dos punto de datos que no son considerados de ninguna manera similares.

2.3.2.2 Creación de Sistemas Lógicos Difusos *(Fuzzy Tools)*

Fuzzy TECH permite la edición gráfica de las variables lingüísticas para cada una de las variables del sistema. Para esto, se pueden utilizar gran variedad de funciones de pertenencia,

como las de tipo S, Z, triangular y PI. Posee también un editor gráfico de reglas, con un formato similar al de una hoja de cálculo; se pueden asociar pesos individuales a las reglas que pueden ajustarse de forma automática utilizando un entrenamiento basado en Redes Neuronales.

Todo sistema difuso desarrollado en fuzzyTECH está compuesto de tres tipos de objetos:

Texto

El texto es un objeto opcional utilizado para hacer transparente la estructura del sistema lógico difuso y describir la estructura del sistema. No tienen ninguna influencia computacional.

Variables

Las variables pueden ser de entrada, intermedias o de salida, y están ligadas cada una a una interface. Las interfaces son mostradas como pequeñas cajas que muestran el nombre de la variable y un ícono que representa el método computacional de fusificación o defusificación elegido. Las variables intermedias no están ligadas con ninguna interface y tienen influencia sólo en el bloque de reglas al que pertenecen.

Bloques de Reglas

En fuzzyTECH, las reglas individuales son contenidas dentro de bloques de reglas (BR), que conforman motores de inferencia difusos. Cada BR contiene las reglas para un conjunto de variables e implementan el proceso de inferencia difusa.

En las interfaces de entrada, ligadas a las variables lingüísticas de entrada, se permiten las siguientes herramientas para llevar a cabo la fusificación:

- Compute MBF, es el método tradicional de fusificación, el valor crisp intersecta la función de pertenencia y se determina un cierto grado de validez.

- Look up MBF, se suele usar en microcontroladores, computa las funciones de pertenencia completamente y almacena los valores en una tabla en el código generado.

En las interfaces de salida, ligadas a las variables lingüísticas de salida, se pueden utilizar las siguientes herramientas de defusificación:

- Center–of–Maximum, computa la salida crisp utilizando un promedio ponderado de los valores máximos de los términos de la variable según los resultados de la inferencia.
- Mean–of–Maximum, computa la salida sólo para el término con mayor grado de validez.
- Center–of–Area, utiliza un promedio ponderado de los valores de las áreas de los términos de las variables.

fuzzy TECH utiliza tres familias de operadores de agregación de entrada que implementan las T-normas y T-conormas (operadores lógicos difusos) del sistema:

- Min–Max (Se puede seleccionar una de ellas o una combinación de ambas).
- Min–Avg (Se puede seleccionar una de ellas o una combinación de ambas).

fuzzy TECH sólo permite el operador de composición PROD para realizar la inferencia difusa.

fuzzy TECH permite dos métodos de agregación de resultado.

- Max (*Maximum*), selecciona la regla con el mayor peso de todas las que tienen alguna relación con el término.
- Bsum (*Bounded Sum*), realiza una suma ponderada de los resultados de las reglas.

Para ilustrar el funcionamiento de las herramientas de diseño de sistemas difusos con fuzzyTECH se presenta el siguiente ejemplo de inferencia:

Supóngase que se tiene una base de conocimientos de dos reglas, y que se ha seleccionado la herramienta de agregación de entradas MAX-MIN con un parámetro de 0 (MIN), que el operador de composición es PROD y que la herramienta de agregación de salidas es MAX.

Regla 1: Si A es grande y B es frío entonces C es azul. (peso = 0.8)
Regla 2: Si A es pequeño y B es caliente entonces C es azul. (peso = 0.6)

Si el grado de verdad del antecedente A es grande es 0.2, A es pequeño es 0.8, B es frío es 0.6 y B es caliente es 0.4; entonces, el grado de validez de las premisas, obtenido mediante la agregación de las entradas utilizando el operador MIN, en la regla 1 es 0.2 y en la regla 2 es 0.4. Utilizando el operador PROD para determinar la validez del consecuente se obtiene 0.16

(0.2*0.8) para la regla 1 y 0.24 (0.6*0.4), y como resultado de la agregación de las salidas (operador MAX), se obtiene que la salida difusa tiene un grado de validez 0.24 azul.

2.3.2.3 Entrenamiento de un Sistema Difuso *(Neurofuzzy Módule)*

El módulo Neurofuzzy de fuzzyTECH contiene herramientas para entrenar un sistema Neuro-difuso; en general, todas ellas combinan el algoritmo de entrenamiento de Retro-propagación con la idea de aprendizaje competitivo, con el fin de evitar que todas las neuronas activas que producen error sean modificadas, ya que esto podría incrementar el error del sistema al ingresar nuevas muestras.

Para realizar el entrenamiento, las herramientas de aprendizaje deben ser configuradas previamente; el módulo Neuro-difuso de fuzzy TECH permite el ajuste de los siguientes elementos: parámetros de aprendizaje, herramientas de aprendizaje, modo de selección de datos y criterios para detener el entrenamiento.

Parámetros de Aprendizaje

Los parámetros que se deben especificar antes de iniciar el aprendizaje son el ancho de paso (Step Width), y el número de neuronas ganadoras. El ancho de paso de entrenamiento es la tasa de aprendizaje del algoritmo de Retro-propagación del Error y determina que valor es añadido o sustraido a los pesos de la neurona ganadora; la determinación de este parámetro depende de la velocidad con que se desea alcanzar una solución, y de la convergencia del sistema. El

número de neuronas ganadoras determina cuantas neuronas son actualizadas dentro de cada paso de iteración; mientras más neuronas ganadoras haya, es menor la protección frente al cambio de resultados de entrenamiento satisfactorios alcanzados por muestras anteriores.

Herramientas de Aprendizaje

El módulo Neurofuzzy contiene herramientas de entrenamiento estándar (REAL – METHOD y RANDOM – METHOD) y "batch" (BATCH – LEARN y BATCH – RANDOM) que utilizan el algoritmo de Retropropagación del error. La primera de cada una de estas herramientas utiliza la tasa de aprendizaje ingresada como paso de entrenamiento constante, mientras que la

segunda utiliza pasos definidos de forma aleatoria a partir del intervalo [0... tasa de aprendizaje] para la actualización; además, los métodos estándar, efectúan un paso de optimización por cada muestra, mientras que los métodos Batch, calculan los errores y gradientes para todas las muestras, y actualizan los parámetros del sistema después de cada iteración completa[14].

Modo de Selección de datos

Los datos de muestra puede ser proveídos al sistema en forma secuencial o en forma aleatoria; la elección y la secuencia de las muestras influencia en el éxito del entrenamiento pues es posible que el resultado de un paso de entrenamiento, o una serie de pasos de entrenamiento, destruya el entrenamiento de los pasos de entrenamiento anteriores y vice-versa.

Criterios para detener el entrenamiento

FuzzyTECH permite tres tipos de criterios para finalizar el proceso de entrenamiento; estos son: (a) por el número máximo de iteraciones (Max. Steps), es decir detener el entrenamiento después de un número fijo de iteraciones; (b) por la desviación promedio (Avg. Dev.), criterio alcanzado si el promedio de los errores durante una iteración completa es menor que un valor umbral definido; y (c) por la máxima desviación (Max. Dev.), que compara el error de la peor muestra con un umbral de error, en este caso, las muestras con un error por debajo del umbral de error no son consideradas en el entrenamiento.

2.3.2.4 Implementación de sistemas Neuro-difusos

La implementación de sistemas difusos y neuro-difusos en fuzzyTECH se puede llevar a cabo de dos maneras: autónoma y servidor-cliente. En el primer caso el sistema neuro-difuso opera independientemente del entorno de desarrollo, para lo cual se pueden utilizar los generadores de código de fuzzyTECH, los cuales permiten generar código fuente C, Cobol, M, así como código fuente para microcontroladores, PLCs y DSPs.

De otro lado, la implementación tipo servidor-cliente permite que el kernel de fuzzyTECH trabaje como servidor de datos para MS Excel, MS VisualBasic u otras aplicaciones de MS Windows, a través de las interfaces DDE, RCU-API (DLL) y *Serial Link*.