63

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

CAMPUS ARAGÓN

PROPUESTA DE UN SISTEMA NEURO-DIFUSO PARA EL MICROCONTROLADOR AL220

TESIS

QUE PARA OBTENER EL GRADO DE:
INGENIERO MECÁNICO ELECTRICISTA

P R E S E N T A:
ULIANOV MONTAÑO JUÁREZ

ASESOR:
ING. BEATRIZ ESLAVA ARELLANES

MÉXICO, D.F. 2001

PROPUESTA DE UN SISTEMA NEURO-DIFUSO PARA EL MICROCONTROLADOR AL220

CONTENIDO

INTRODUCCIÓN

El fin de este trabajo es presentar los fundamentos teóricos y los primeros resultados de una investigación en sistemas inteligentes, específicamente en el campo de sistemas Neuro-Difusos. Las características de estos sistemas se aclarará a lo largo del texto; se considera pertinente, sin embargo, hacer una breve discusión en torno a la expresión *sistema inteligente*.

La inteligencia es una noción compleja a la cual se ha intentado clarificar de varios modos. Estos van desde definiciones operacionales hasta filosóficas.

Una forma típica de definición operacional es por ejemplo: Se dice que un sistema posee inteligencia cuando un observador (humano) es incapaz de determinar que su interlocutor no posee inteligencia. Esta definición no es de gran ayuda al momento de intentar diseñar un mecanismo que imite conductas inteligentes; además tiene reminiscencias del tipo de formulaciones psicológicas que eran comunes en la psicología conductista de los años cincuenta.

Por otra parte, la memoria, la razón y la imaginación son las facultades que definen al ser humano como una criatura racional. Esa ha sido una opinión ampliamente difundida entre los filósofos desde el siglo XVII. Pero considerando estas facultades *por separado* parece que tienen poco que decirnos acerca de la índole del ser humano como ser inteligente. Por separado, al parecer, no tienen mucho que ver con la actividad de un ser humano. Pensemos en la facultad de recordar; muchos animales poseen también esa habilidad. Podemos construir máquinas capaces de retener información y de disponer de ella a solicitud. Mecanismos tan sencillos como imprimir una cierta característica en un material plástico parecen suficientes para imitar la habilidad de la memoria.

La razón en el siglo XX ha pasado por un proceso análogo al de la memoria. Los psicólogos cognitivos han encontrado evidencia de que por lo menos ciertas clases de primates tienen capacidades de abstracción, que creíamos exclusivas del género humano. La capacidad de comparar y elaborar juicios puede en cierta medida mecanizarse construyendo un lenguaje formal, con reglas sintácticas y correspondencias semánticas. Al formalizar un lenguaje estamos en la posibilidad de elaborar mecanismos que puedan llevar a cabo las reglas de la sintaxis. Cuando G. Boole publicó el primer tratado acerca de lo que hoy conocemos como *álgebra booleana* pretendía haber descubierto las propiedades mas significativas de nuestra facultades de razonar y de emitir juicios: el quería que sus descubrimientos acerca de sintaxis formal descifraran por fin las *leyes del pensamiento*. Los teóricos de la inteligencia artificial de las décadas del 50 y 60 parecían estar contagiados de un entusiasmo similar al de Boole. Eso es comprensible; pues atarearse en la construcción de mecanismos cuyo propósito es la emisión de pensamientos es, acaso, una de las empresas mas fascinantes y complejas, aún si los fines a los que se destina son de índole utilitaria. Poco a poco fue haciéndose evidente que los esfuerzos en el área de la inteligencia artificial tenían limitaciones inherentes que el mero recurso de la sintaxis lógica no podría superar (es interesante

notar que en el campo de la teoría y metodología científica sucedió algo análogo: los componentes para emitir juicios acerca de teorías parecían necesitar de instrumentos más complejos que la lógica). Problemas como el reconocimiento de patrones o la expresión de matices y énfasis en la emisión de juicios, que para un animal o un ser humano resultan tareas triviales, resultaban sumamente difíciles con las herramientas de la sintaxis lógica. Estos problemas motivaron incluso el diseño de lógicas alternativas como lógicas modales o temporales. Algo que sí se puso de manifiesto es que la expresión del razonamiento humano como mera sintaxis es posible, pero resulta insuficiente para aprehender las formas de la razón humana.

Hacer una ilustración análoga a la de la memoria o la del razonamiento resulta difícil con la facultad de imaginar. Se puede pensar, no obstante, en los métodos de generación de patrones aleatorios. En lugares como MIT[1], Stanford o IRCAM[2] se trabaja desde hace varios años en técnicas de composición musical automática, mediante algoritmos de computadora. Por lo demás, es difícil imaginar a los animales superiores, sin la facultad de imaginar. Para sobrevivir un organismo ha de ser capaz no solo de reconocer su entorno, sino de proyectar posibles situaciones riesgosas, es decir ha de poseen en último término un una habilidad básica de imaginar.

El motivo de esta breve discusión es brindar una perspectiva de las motivaciones de este trabajo. En cierta medida hoy en día, tal vez sin este propósito explícito en mente, contemplamos la posibilidad de imitar habilidades que parecen centrales a la idea del ser humano como una especie diferenciada en ciertos aspectos del resto de los animales. No obstante no se pone mucho énfasis en esto; entre la comunidad de investigadores en Vida e Inteligencia Artificial hay una división entre aquellos que tiene un interés profundo en temas relacionados con la imitación de facultades humanas, y aquellos cuya preocupación es sobre todo la elaboración de mecanismo con fines prácticos. En realidad esta división es en alguna medida artificial, pues de hecho en la práctica unos y otros se ven constantemente confrontados con problemas de todo tipo. Aun si nuestro propósito es el diseño de un mecanismo con fines prácticos, no podemos mantenernos ajenos acerca de sus repercusiones en otros ámbitos, como el ético, el científico o el epistemológico[3].

Regresando a la discusión original, se debe tratar de clarificar por que las habilidades de recordar, razonar e imaginar no son ni especificas ni definitorias de un comportamiento inteligente. En realidad deberíamos pensar que es la unión de tales habilidades la que nos llevaría a sistemas con comportamientos inteligentes. Si suponemos esto, tendremos que enfrentar dos serios problemas: primero el del emergentismo: es decir no tenemos en realidad un mecanismo que explique como a partir de propiedades elementales surge un complejo mayor con propiedades complejas que no están presentes en sus elementos. La inteligencia, posee propiedades que no están presentes ni en los elementos constitutivos por separado, ni en el conjunto que forman la unión de las tres habilidades singulares. Segundo y mas importante, estas habilidades

---

[1] Massachussetts Institute of Technology

[2] Institut de Recherche et Coordination Acoustique/Musique : IRCAM es el centro de investigación y desarrollo acústico musical mas importante internacionalmente, esta ubicado en el Centro Poumpidou, Paris, Francia.

[3] La epistemología es la rama de la filosofía que se ocupa de los problemas de la cognición, es decir, es la teoría filosófica del conocimiento.

pueden de hecho ser conjuntadas en un sistema, por ejemplo una computadora lo suficientemente potente. Aún así la facultad de inteligencia solo podría pensarse en términos de una definición operacional.

La dificultad radica en al menos tres puntos que se han descuidado. El primero es que no basta la simple unión de capacidades, sino que se trata de una *integración* mas complicada. Lo segundo es que estas habilidades están presentes en diversos grados, la emulación de inteligencia no es cuestión de emergencia, sino de grados. Tercero, se ha olvidado una característica importante de las operaciones de la inteligencia, la facultad de crear, que es diferenciable de la facultad de imaginar.

Es aquí donde parece relevante detenerse en los sistemas de tipo Neuro-Difuso. Como se verá en los capítulos uno y dos, estos sistemas tienen sus raíces en dos tipos de modelos que se aproximan de manera diferente al procesamiento de información que los sistemas convencionales basados en lógica clásica. Por una parte los modelos de redes neuronales tienen como eje de articulación, en las etapas incipientes de su desarrollo, una idea estructuralista; se piensa que al copiar estructuras se pueden copiar procesos. Eso es ya de por si una idea interesante: en lugar de buscar algoritmos deterministas que imiten las propiedades externas de un sistema inteligente se opta por reproducir las estructuras profundas en las que estos se sustentan.

Por otra parte, los modelos de procesamiento por lógica difusa tratan de incorporar a un modelo sintáctico ciertas propiedades que se observan en los lenguajes naturales. Este es un acercamiento diferente al de las redes neuronales. Mientras los modelos de redes parten del principio de imitar estructuras subyacentes, los modelos de lógica difusa plantean modelar de una forma altamente formalizada estructuras muy complejas como los lenguajes naturales y los procesos simbólicos de comparación. Debe destacarse que en lógica difusa se conservan valores veritativos que admiten una cantidad infinita de estados intermedios entre la verdad y la falsedad.

Una red neuronal no puede entenderse como una suma de pequeñas partes interdependientes, se trata de un todo que posee propiedades que no poseen ni los elementos de procesamiento por separado ni su simple unión, mas bien se trata de un sistema dinámico, global y diacrónico(es decir que su historia repercute en su respuesta futura). Estas propiedades resultan interesantes si consideramos que tampoco los comportamientos inteligentes pueden entenderse como la unión de sus elementos constitutivos, como ya se ha mencionado responden sobre todo a procesos de integración de manera similar a los sistemas neuronales. También se ha dicho que los facultades no solo deben integrarse sino que deben hacerlo en cierto grado, debe haber propiedades con la suficiente intensidad. Estas nociones de grado están presentes también en la construcción de un sistema simbólico basado en lógica difusa. Por último, la integración de estas dos tendencias en apariencia muy diferentes, parece algo natural si consideramos que en la inteligencia hay un elemento creador además de la mera proyección de situaciones.

La integración de modelos neuro-difusos, responde entonces a la necesidad se sobrellevar problemas en el campo de sistemas inteligentes, que otrós modelos no pueden afrontar sin severas dificultades. Esta perspectiva no puede dejar de entusiasmar a quien quiera que este (por las razones que fueren) interesado en modelar sistemas que se conduzcan de una manera *inteligente*.

Resultado de este entusiasmo ha sido la investigación que se desarrolla en los

siguientes capítulos. Allí se expondrán los pormenores teóricos de los modelos neuronales y difusos para luego pasar a presentar algunas propuestas de sistemas neuro-difusos. Por último se presentará una propuesta de sistema neuro-difuso original que busca aprovechar recursos de hardware de bajo costo. Siendo los sistemas a base de lógica difusa y sobre todo los sistemas neuronales de reciente introducción en el país, queda justificado el hacer un examen detenido de estos modelos a nivel teórico, pues el material disponible para consulta no es de fácil acceso a nivel de licenciatura, y la comprensión de la estructura de los sistemas híbridos requiere aún mas atención. Así, entonces, los primeros capítulos de este trabajo pueden considerarse como material básico y se justifican por si mismos. Por estas mismas razones, la discusión de las propiedades formales de el nuevo sistema neuro-difuso propuesto queda fuera del alcance de este trabajo, pues es un tópico en el cual aún se esta trabajando. Lo mismo ocurre con la implementación en hardware y su puesta a prueba en planta que además requiere de recursos económicos adicionales. Debe enfatizarse entonces que este trabajo se propone exponer los fundamentos teóricos necesarios para la propuesta de un sistema original, únicamente. Esto se llevará a cabo en los siguientes capítulos atendiendo a las siguientes divisiones.

En el capítulo uno se exponen los fundamento de la lógica difusa, se le comparan con las propiedades de la lógica clásica también se presenta un modelo de sistema difuso que opera regularmente como sistema de control.

El capitulo dos se centra en los modelos de redes neuronales artificiales, estos modelos son mas antiguos que la lógica difusa, sin embargo han pasado por etapas largas de desinterés y actualmente, al menos en los ámbitos de ingeniería en México, gozan de menor difusión que los modelos difusos, por ello se insiste en una recapitulación histórica, seguida de una breve exposición de los modelos nerviosos biológicos de donde han sido derivados; para luego exponer sus componentes esenciales: arquitectura, función de activación y algoritmos de aprendizaje. También se hace un breve recuento de algunos puntos interesantes de su aplicación al área de control, área que se ha dado en denominar *neurocontrol*.

El capitulo tres plantea como se puede pensar en una unificación de los campos de la lógica difusa y las redes neuronales. Se exponen alguno ejemplos típicos de como se ha hecho esto y también el modelo de la Neo Neurona Difusa, que presenta propiedades que serán explotadas para plantear el nuevo modelo.

El capitulo cuarto y último hace una somera descripción de las propiedades de los problemas a los cuales esta enfocada la propuesta para después exponer las cualidades y como se relacionan esta con los modelos difusos y neuronales por separado, además se expone el hardware que se ha tenido en cuenta a la hora de proponer el diseño, como las características de aquel determinan la estructura de este. Por fin se expone la propuesta concreta refiriendo sus propiedades como red adaptativa : arquitectura; algoritmo de aprendizaje y funciones sinápticas.

CAPITULO I : LÓGICA DIFUSA[1]

ANTECEDENTES.

El desarrollo industrial, y por lo tanto el desarrollo de técnicas y herramientas, es la base del sistema económico actual. La evolución de la ciencia y los apremiantes requisitos de productividad del sistema han hecho que hoy en día estas técnicas, en todos los ámbitos, se desarrollen de manera vertiginosa.

Desde finales de la década pasada, una de las tecnologías emergentes más exitosa ha sido el *Control Inteligente*. Jamshidi [1] lo define como

> *la combinación de la teoría del control, investigación de operaciones e inteligencia artificial (IA).*

Entre las tecnologías basadas en IA, la lógica difusa es un área que se ha popularizado rápidamente a partir del anuncio del primer chip difuso en 1987. En los países desarrollados se invierten billones de dólares en el desarrollo y comercialización de productos que integran lógica difusa, la cual ha tenido éxito en aplicaciones tan delicadas como el diagnóstico médico. El éxito comercial de esta tecnología la hace atractiva para ser adaptada a otros sistemas a fin de reducir costos o aumentar la eficiencia.

CONJUNTOS DIFUSOS

En 1965, Lotfi A. Zadeh introduce la teoría de conjuntos difusos, cuyos objetos son conjuntos sin confines bien delimitados. La pertenencia a uno u otro conjunto no es cuestión de afirmación o negación, sino de grado.

El trabajo de Zadeh es un salto conceptual. Si x es un objeto y A es un conjunto difuso, la proposición "x es miembro de A" tendrá una veracidad que no necesariamente es *cierto* o *falso*, sino que será cierto o falso en el mismo grado en que x es miembro de A. Para representar los grados de pertenencia o de veracidad de las proposiciones asociadas, se utiliza comúnmente el intervalo de los números reales [0,1]. Los extremos de este intervalo representan la negación o afirmación total de la pertenencia del elemento, así como la veracidad o falsedad total de sus proposiciones asociadas.

La capacidad de los conjuntos difusos de expresar transiciones graduales entre la pertenencia y no pertenencia es de amplia utilidad. Puede servir para representar las incertidumbres inherentes a las mediciones o bien para representar de modo significativo la vaguedad de los conceptos expresados en el lenguaje natural.

Para ejemplificar el concepto de conjunto difuso se recurrirá al ejemplo de clasificar por edad a un grupo de personas. Pueden clasificarse por la cantidad exacta de

---

[1] *Fuzzy Logic*. El adjetivo *difuso* es al que con más frecuencia se recurre para traducir el inglés *fuzzy*. Lo que se intenta dar a entender es el carácter ambiguo de ciertas propiedades cuado se les examina en casos límite o de frontera.

años que hayan cumplido a la fecha. Pueden también ser calificados como *joven* o *maduro*. Estos últimos conceptos son más vagos, intuitivos, pero mas fáciles de manejar para un ser humano. El problema surge cuando es preciso discriminar un elemento en una zona ambigua: una persona de 35 años puede considerarse joven, pero alguien de 60 años, no. Debe aceptarse una edad intermedia que señale la transición entre joven y maduro, por ejemplo joven hasta 40 años. Sin embargo alguien de 41 años presentará diferencias tan sutiles con alguien de 40, que parece absurdo admitirlo en la otra categoría. Habrá, entonces, que aceptar que una diferencia de 1 año no hace cambiar una persona de una categoría a otra. Pero considerar esto hará que todas las personas se clasifiquen como jóvenes o como maduras, es decir que los intervalos se extiendan sobre todo el ámbito de edades. Para resolver este inconveniente es necesario introducir la cualidad de vaguedad por medio de una transición gradual entre las edades que pertenecen a una categoría. El concepto básico de conjunto difuso tiene estas últimas propiedades, es más simple e intuitivo para el ser humano y es un concepto más general que el de la teoría clásica de conjuntos.

En los conjuntos clásicos los objetos del *Universo del Discurso* (el conjunto tal que todo conjunto particular A está en relación de contensión con el primero)[2] se discrimina en miembros y no miembros de un conjunto. Existe una distinción inequívoca y precisa entre los miembros y los que no lo son. Sin embargo en muchos procesos de clasificación utilizamos conceptos que no son de este tipo: v.g. "números muy grandes", "mediciones precisas", "personas altas", "mucha contaminación", etc. En general usamos calificativos; calificamos los objetos del mundo real dentro de conjuntos definidos. El concepto de conjunto difuso nos ofrece herramientas para tratar con la vaguedad de las calificaciones.

Un conjunto difuso puede modelarse matemáticamente asignando a cada entidad perteneciente a un universo del discurso un valor que represente su grado de membresía a un conjunto particular dado. Este grado corresponde a la intensidad con que dicho elemento es similar o compatible con el concepto representado por el conjunto. Las entidades serán miembros del conjunto en mayor o menor grado según lo indique el valor asignado.

CONTROL INTELIGENTE

La Lógica difusa encontró, en un principio, un amplio campo de aplicación en los sistemas de tamaño y complejidad modesta. Uno de los primeros sistemas complejos en adoptarla exitosamente, en 1977, fue un horno de cemento, en Dinamarca. Actualmente la mayoría de estos hornos utiliza algún tipo de sistema experto difuso

Una de las áreas mas activas de la lógica difusa son los sistemas de control. Los controladores difusos son sistemas expertos que hacen suaves interpolaciones entre reglas bien delimitadas. Varias reglas son activadas simultáneamente por una variable de entrada en grados continuos, estas múltiples activaciones son finalmente combinadas para obtener el resultado o la acción final interpolada. Las bases para el control difuso son: El procesamiento de información con incertidumbre y el ahorro de energía valiéndose para ello del uso de reglas de "sentido común" y declaraciones lingüísticas. Existen muchas tareas que implican un *lazo de decisión*, que es la parte del sistema encargada de ejercer el control de acuerdo a los datos provenientes de los sensores. En muchos casos esta acción la realiza un operario humano basándose en la experiencia y el sentido común. Esta experiencia se codifica en forma de un conjunto de reglas lingüísticas condicionales, muchas veces inconscientes, del tipo *SI x cae en el caso a*

ENTONCES la acción a tomar es el caso b. Estas reglas no son del tipo *nítido* (bien definidas) sino, como todo lo humano, en base a juicios vagos. Tales problemas pueden ser vaciados a un conjunto de variables y reglas difusas, que estructuradas adecuadamente pueden decidir tan bien como lo haría el experto humano. Poco a poco, los sistemas artificiales se aproximan a las capacidades humanas, pero los sistemas tradicionales tienen limitaciones, los temas que se expondrán en este trabajo resumen algunos de los métodos que, en el momento actual, se plantean como vías alternas de solución

LÓGICA CLÁSICA Y LÓGICA DIFUSA

La intención, de este capítulo, es exponer los fundamentos teóricos de la *Lógica Difusa*. La lógica difusa es una herramienta que nos permite tratar con objetos conceptuales que poseen una vaguedad inherente. La principal motivación para el desarrollo y la aplicación de la lógica difusa es su tolerancia a la inexactitud y a la imprecisión.

Teoría de Conjuntos y Teoría de Conjuntos Difusos

La exposición de la Lógica como un sistema axiomatizado, se hace hoy en día introduciendo primero la Teoría de Conjuntos. Despues, por referencia a ésta se define la lógica. Aquí se hará algo similar: primero se describe la Teoría de Conjuntos Difusos y despues la Lógica Difusa.

La mejor forma de exponer la teoría de conjuntos difusos es por comparación y contraste con la teoria clásica de conjuntos. Ambas comparten gran cantidad de postulados, preceptos y axiomas, pero la teoria de conjuntos borrosos es más amplia que la clásica. Mientras que la teoría de conjuntos admite solo dos grados de membresía: los valores canónicos 0 y 1, la teoria de conjuntos difusos admite un número infinito de grados de membresía entre 0 y 1.

Un *conjunto difuso* es una colección de elementos, en un universo de información donde la frontera del conjunto contenido en el universo es ambigua, vaga o borrosa.

Teoría de Conjuntos

Se define U como el conjunto de todos los objetos con las mismas características, se le denomina *Universo del Discurso*.

Los elementos individuales del universo se designan usando *x*.

Un conjunto A es una colección o varias colecciones de elementos del universo U. El conjunto A es un *subconjunto* de U.

Se define el *número de cardinalidad* n_x, como el número total de elementos en el conjunto X.

Se usará la siguiente notación:

$x \in X$ *x* pertenece a X

$x \in A$ *x* pertenece a A

$x \notin A$ *x* no pertenece a A

Para los conjuntos A y B en U tenemos:

$A \subset B$ A esta contenido en B. De otra forma: $\forall x \in A$ entonces $x \in B$

$A \subseteq B$ A esta completamente contenido en B

$A = B$ $A \subseteq B$ y $B \subseteq A$

$\varnothing$ *conjunto vacio*. El conjunto que no contiene ningún elemento.

P(X) *Conjunto Potencia de X*. El conjunto conformado por todos los subconjuntos de X

Operaciones con Conjuntos

Sean dos conjuntos A y B en U. Se definen las siguientes operaciones:

$A \cup B$	Unión	$A \cup B = \{x \mid x \in A \text{ o } x \in B\}$
$A \cap B$	Intersección	$A \cap B = \{x \mid x \in A \text{ y } x \in B\}$
$\overline{A}$	Complemento	$\overline{A} = \{x \mid x \notin A \ , \ x \in U\}$
$A \setminus B$	Diferencia	$A \setminus B = \{x \mid x \in A \text{ y } x \notin B\}$

los diagramas de Venn representativos se muestran en seguida

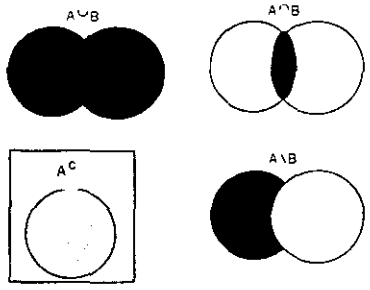

FIGURA 1.1 Diagramas de Venn para las 4 operaciones basicas

Propiedades de los Conjuntos

La manipulación matemática de conjuntos se facilita si se tienen presentes sus propiedades. Enseguida se muestran las más significativas para compararlas con las de los conjuntos difusos:

Conmutatividad	$A \cup B = B \cup A$
	$A \cap B = B \cap A$
Asociatividad	$A \cup (B \cup C) = (A \cup B) \cup C$
	$A \cap (B \cap C) = (A \cap B) \cap C$
Distributividad	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
Idempotencia	$A \cup A = A$
	$A \cap A = A$

Identidad $A \cup \varnothing = A$

$A \cap U = A$

$A \cap \varnothing = \varnothing$

$A \cup U = U$

Transitividad $A \subseteq B \subseteq C \Rightarrow A \subseteq C$

Involución $\overline{\overline{A}} = A$

Los conjuntos exhiben dos propiedades importantes: Las leyes del Tercero Exculido y las leyes de DeMorgan. Las leyes del Tercero Excluido son dos, la ley del tercero Excluido y la Ley de Contradicción. Estas leyes son relevantes porque son válidas sólo para conjuntos en la axiomatización clasica y no para los conjuntos difusos. Las Leyes de DeMorgan son útiles para probar tautologías y contradicciones lógicas.

Leyes del Tercero Excluido:

1. Ley del tercero excluido: $A \cup \overline{A} = U$
2. Ley de Contradicción $A \cap \overline{A} = \varnothing$

Leyes de DeMorgan:

- $\overline{A \cap B} = \overline{A} \cup \overline{B}$
- $\overline{A \cup B} = \overline{A} \cap \overline{B}$

Mapeo de Conjuntos a Funciones.

El mapeo es una forma más general de modelar conjuntos, además permite la introducción de los conjuntos difusos de manera fácil. Mapear es establecer relaciones entre elementos o subconjuntos de un conjunto en un Universo del Discurso con elementos de o subconjuntos de algún otro Universo.

Sean X y Y dos diferentes Universos del Discurso. Si un elemento x contenido en X corresponde a un elemento y contenido en Y ,esto se denomina un mapeo de X a Y , o bien: f: $X \rightarrow Y$. La *función característica* χ_A se define como el mapeo:

$$\chi_A \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$$

χ_A expresa "membresía" o pertenencia del elemento x del Universo respecto del conjunto A. La idea de membresía es un mapeo: a un elemento x en el universo del discurso X, corresponde alguno de los dos elementos, 0 o 1, del Universo Y.

Para cada conjunto A definido en U, existe una asignacion de verdad, V(A) bajo el mapeo de la función característica χ . Por convención, al cojunto vacio se le asigna un valor de membresía 0 ($\forall x \ x \notin \varnothing$); mientras que al conjunto universal se le asigna el valor

de membresía 1 ($\forall x \ x \in U$).

Operaciones en Términos de Funciones

Las operaciones con conjuntos pueden definirse apelando a relaciones entre las funciones características χ. Este enfoque es muy próximo a la manera como se definen las operaciones en conjuntos difusos.

Sean dos conjuntos A y B en U. Las operaciones, entonces, se definen así:

Unión	$A \cup B \rightarrow \chi_{A \cup B}(x) = \max(\chi_A(x), \chi_B(x)) = \chi_A(x) \vee \chi_B(x)$
Intersección	$A \cap B \rightarrow \chi_{A \cap B}(x) = \min(\chi_A(x), \chi_B(x)) = \chi_A(x) \wedge \chi_B(x)$
Complemento	$\overline{A} \rightarrow \chi_{\overline{A}}(x) = 1 - \chi_A(x)$
Contensión	$A \subseteq B \rightarrow \chi_A(x) \leq \chi_B(x)$

Conjuntos Difusos

En los conjuntos clasicos la transición entre membresia y no membresia es abrupta y bien definida de uno a otro elemento del Universo.Pero para un conjunto difuso en el mismo Universo, la transición de membresía a no membresía, es gradual, ello refleja el hecho de que las fronteras del conjunto difuso son ambiguas. La pertenencia o membresia de un elemento se mide por medio de una función que representa la vaguedad de las fronteras.

Un *Conjunto Difuso*, es un conjunto que contiene elementos cuya membresía a dicho conjunto puede ser más o menos intensa. Elementos que pueden pertenecer al conjunto solo en cierto grado. Esto contrasta con los elementos de los conjuntos clasicos, estos solo son o no son miembros del conjunto. Los elementos de un conjunto difuso pueden ser miembros de otro conjunto en el mismo Universo simultaneamente, y no necesariamente en el mismo grado.

Los elementos de un conjunto difuso se mapean a un Universo de *valores de membresía* usando una forma funcional. Se usa una función denominada *función de membresía*, que mapea los elementos de un conjunto difuso a un intervalo de números reales, generalmente el intervalo $[0,1]$. Si un elemento x en el universo, es miembro del conjunto difuso A, el mapeo esta dado como:

$$\mu_A(x) \in [0,1]$$

$$A = (x, \mu_A(x) | x \in U)$$

En varios autores se encuentra la notacion convencional siguiente, válida para un Universo del Discurso discreto y finito, para un conjunto difuso A

$$A = \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_1)}{x_1} + \cdots = \sum_i \frac{\mu_A(x_i)}{x_i}$$

Para un Universo continuo e infinito, el conjunto difuso A en U, se denota:

$$A = \int \frac{\mu_A(x)}{x}$$

La barra horizontal es solo un delimitador, no un indicador de cociente. Los simbolos + y $\sum$ indican la operación de unión, mientras que el simbolo de integración indica unión para variables continuas.

Operaciones con Conjuntos Difusos.

Sean tres conjuntos difusos A, B, C en U y para un elemento *x* de U se definen las siguientes operaciones en forma funcional.

Unión	$\mu_{A\cup B}(x) = \mu_A(x) \vee \mu_B(x)$
Intersección	$\mu_{A\cap B}(x) = \mu_A(x) \wedge \mu_B(x)$
Complemento	$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$

Los correspondientes diagrmas de Venn son:

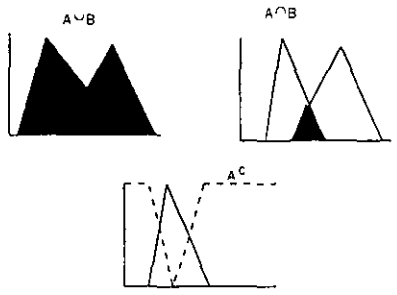

FIGURA 1.2 Diagramas de Venn para las 3 operaciones difusas.

Todo conjunto difuso en U es un subconjunto de U.

De manera similar a los conjuntos clásicos, el cojunto vacio $\varnothing$ asigna 0 a toda *x*. Al conjunto universal se le asigna 1 para toda *x*. De modo formal:

$$A \subseteq U \rightarrow \mu_A(x) \leq \mu_U(x)$$

$$\forall x \in U,\ \mu_{\varnothing}(x) = 0$$

$$\forall x \in U,\ \mu_U(x) = 1$$

La colección de todos los conjuntos y subconjuntos difusos conforma el Conjunto Potencia difuso *P(X)*. La cardinalidad de este conjunto difuso es infinita:

$n_{P(x)} = \infty$

Las leyes de DeMorgan son validas también para conjuntos difusos:

$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$

Las demás leyes enunciadas anteriormente para conjuntos calsicos valen también para los conjuntos difusos. Es importante notar que la extensión tiene dos excepciones importantes las Leyes del Tercero Excluido, es decir la ley de tercero excluido y la ley de contradicción:

$$A \cup \overline{A} \neq U$$

$$A \cap \overline{A} \neq \varnothing$$

Propiedades de los Conjuntos Difusos.

Los conjuntos difusos poseen las mismas propiedades que los conjuntos clasicos. Este hecho aunado al de que los valores de membresia de los conjuntos clasicos forman un subconjunto del intervalo $[0,1]$, nos permite decir que los conjuntos clasicos se consideran un caso particular de los conjuntos difusos. Las propiedades generales de los conjuntos difusos son:

Conmutatividad	$A \cup B = B \cup A$
	$A \cap B = B \cap A$
Asociatividad	$A \cup (B \cup C) = (A \cup B) \cup C$
	$A \cap (B \cap C) = (A \cap B) \cap C$
Distributividad	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
Idempotencia	$A \cup A = A$
	$A \cap A = A$
Identidad	$A \cup \varnothing = A$
	$A \cap U = A$
	$A \cap \varnothing = \varnothing$
	$A \cup U = U$
Transitividad	$A \subseteq B \subseteq C \Rightarrow A \subseteq C$
Involución	$\overline{\overline{A}} = A$

Si se revisan las anteriores expresiones para las características y operaciones y se comparan con las dadas para conjuntos clásicos, se reconoce inmediatamnete la similitud, más aún, se advierte que las expresiones para los conjuntos difusos son más generales.

Características de la Función de Membresía

Toda la información contenida en un conjunto difuso es descrita por su función de membresía, la función de membresía puede ser de cualquier tipo que susceptible de representar una necesidad concreta. La más comunes, por la facilidad que ofrecen de ser implementadas en computadoras digitales son las de forma triangular o trapezoidal. Para estas últimas resulta útil adoptar una terminología descriptiva de las características peculiares de esta función. La figura siguiente muestra los términos que se adoptan en este trabajo.

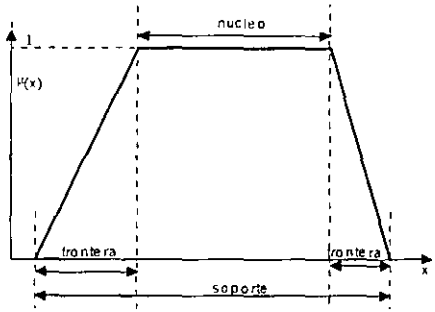

FIGURA 1.3 Soprte, Nucleo y Fronteras de las Funciones de Membresía.

Lógica Clásica.

La introducción de la teoría de conjuntos ha dispuesto las circunstancias propicias para la introducción de la lógica de modo formal. Aunque la teoría de conjuntos fija su génesis a finales del siglo XIX, mientras que las investigaciones en materia de lógica se remontan a la antigüedad clásica; hoy en día la forma convencional como se introduce la lógica, es con relación a la teoría de conjuntos. Se retoma aquí la exposición de la lógica difusa, siguiendo la misma pauta que en la exposición de la teoría de conjuntos difusos: exponiendo primero las bases de la lógica clasica para después extenderse a la lógica difusa.

En la lógica predicacional clásica se define una *proposición simple* P, como una declaración ligüistica contenida en un universo de proposiciones, que puede ser calificada como estrictamente falsa o estrictamente verdadera. A la veracidad de la proposición P se le asigna una cantidad binaria T(P), que se designa *valuacion de verdad* , tal como se

hizo con los elementos de un universo para representar su membresía a algún conjunto particular. Si P es verdadera se le asigna 1,si es falsa se le asigna 0. Tenemos entonces el mapeo:

$$T:U \rightarrow \{0,1\}$$

siendo U el universo de todas las proposiciones.

Sean P y Q proposiciones simples en U que pueden ser combinadas usando los siguientes conectores:

Disyunción $(\vee)$

Conjunción $(\wedge)$

Negación $(\overline{\ })$

Implicación $(\Rightarrow)$

Doble implicación $(\Leftrightarrow)$

La combinación forma nuevas expresiones, a partir de proposiciones simples.

Ahora, sean dos conjuntos A y B en el universo del discurso X, donde estos conjuntos representan ideas o pensamientos. Existirá, entonces, un *cálculo proposicional* para el caso de que la proposición P mida o estime la veracidad de la declaración de que un elemento *x*, en X, está contenido en el conjunto A y la veracidad de la declaración de que el mismo elemento *x* está contenido en el conjunto B, formalmente:

P: verdad que $x \in A$

Q: verdad que $x \in B$, la veracidad se mide en términos de la valuación de verdad, esto es:

Si $x \in A$, T(P)=1; De otro modo T(P)=0.

Si $x \in B$, T(Q)=1 de otro modo T(Q)=0.

Otra forma es usar la la función característicarepresentando 1 verdadero y 0 falso.

$$\chi_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$$

Se define una *proposición compuesta* como una proposición lógica formada por la conexión o de dos o más proposiciones simples mediante conectores lógicos. La lógica se ocupa de las reglas para determinar la veracidad de proposiciones compuestas. Determinar la veracidad de las proposiciones simples esta fuera del ámbito de la lógica. Las proposiciones compuestas, resultantes de usar los cinco conectores lógicos, para las proposiciones P y Q se definen:

$P: x \in A, \quad \overline{P}: x \notin A$

$P \vee Q \Rightarrow x \in A \text{ o } x \in B \quad$ por tanto $T(P \vee Q) = \max(T(P), T(Q))$

$P \wedge Q \Rightarrow x \in A \text{ y } x \in B \quad$ por tanto $T(P \wedge Q) = \min(T(P), T(Q))$

Si $T(P)=1$, *Entonces* $T(\overline{P})=0$; *Si* $T(P)=0$, *Entonces* $T(\overline{P})=1$

$P \Leftrightarrow Q \Rightarrow x \in A,B$

por tanto, $T(P \Leftrightarrow Q) = T(P) = T(Q)$

Implicación Clásica

Usando analogías con los conjuntos:

$P \Rightarrow Q \equiv \bar{A} \cup B$ *es verdad* ≡ *(no en A)* o *(en B)*

Así, tenemos $P \Rightarrow Q \Leftrightarrow (\bar{P} \vee Q)$

$T(P \Rightarrow Q) = T(\bar{P} \vee Q) = \max(T(\bar{P}), T(Q))$.

Las proposiciones P y Q pueden tomar dos valores de verdad: verdadero o falso. Los valores de verdad (V o 1 verdadero, F o 0 falso) para cada proposición compuesta con P, Q y los conectores ya mencionados se muestran en la siguiente tabla de verdad:

P	Q	$\bar{P}$	$P \vee Q$	$P \wedge Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
V	V	F	V	V	V	V
V	F	F	V	F	F	F
F	V	V	V	F	F	F
F	F	V	F	F	F	V

La implicación es especialmente útil en el diseño de sistemas expertos. En particular si la implicación involucra dos universos del discurso diferentes. P es una proposición descrita por el conjunto A, definido en el universo X; Q es un a proposición descrita por el conjunto B, definido en el universo Y. × indica el Producto Cartesiano, la relación R representa, en términos de la teoría de conjuntos la operacion $P \Rightarrow Q$:

$R = (A \times B) \cup (A \times Y) \equiv$ Si A, Entonces B

Si $x \in A$, donde $x \in X$, $A \subset X$

entonces $y \in B$, donde $y \in Y$, $B \subset Y$

Esta implicación es equivalente a la regla lingüística *si A, entonces B*. Y que en función de Teoría de Conjuntos es:

$P \Rightarrow Q$: Si, $x \in A$, entonces $x \in B$, o bien : $P \Rightarrow Q \equiv \bar{A} \cup B$

Para fines prácticos son interesantes tres casos en que aparecen proposiciones simples: Las *tautologías*, las *contradicciones* y las *inferencias deductivas*.

Una *tautología* es una proposición compuesta que siempre es verdad, siendo indiferente el valor de las proposiciones simples que la componen. Las formas más comunes de tautologías se listan a continuación, por el carácter de este trabajo se omiten las pruebas.

$\bar{B} \cup B \Leftrightarrow U$

$A \cup U, \ \bar{A} \cup U \Leftrightarrow U$

$(A \wedge (A \Rightarrow B)) \Rightarrow B$ *Modus Ponens*

$(\bar{B} \wedge (A \Rightarrow B)) \Rightarrow \bar{A}$ *Modus Tolens*

Por el contrario, una *contradicción* es una proposición compuesta que es siempre falsa, independientemente de los valores de verdad de las proposiciones simples que la componen. Los mas comunes son:

$\bar{B} \cap B$

$A \cap \varnothing; \ \bar{A} \cap \varnothing$

La *Inferencia Deductiva*: La tautología denominada Modus Ponens se usa como herramienta de inferencia en sistemas basados en reglas lógicas. Reglas del tipo *SI - ENTONCES*, se usan para determinar si un *antecedente* (causa o acción) infiere un *consecuente* (efecto o reacción).

Supongase la regla

Si A , Entonces B

Si se admite un nuevo antecedente A', se obtendrá una nueva regla de la forma

Si A' , Entonces B'

Ello se deduce usando la composición de relaciones, que en terminos de la Teoría de Conjuntos se expresa así:

$$B' = A' \circ R = A' \circ ((A \times B) \cup (\bar{A} \times Y))$$

donde $\circ$ denota composición de funciones.

La regla *Si A , Entonces B* (P definida en A, en el universo X y Q definida en B, en el universo Y), se define en terminos de funciones como:

$$P \Rightarrow Q: \ R = (A \times B) \cup (\bar{A} \times Y)$$

$$\chi_R(x,y) = \max\left[\left(\chi_A(x) \wedge \chi_B(y)\right), \left((1 - \chi_A(x)) \wedge 1\right)\right]$$

donde $\chi(\cdot)$ es la función caracteristica definida previamente en este trabajo.

Lógica Difusa.

Una *proposición difusa*, P, es una declaración que involucra algún concepto cuyas fronteras no estan estrictamente delimitadas. En general, los lenguajes naturales son de este tipo, involucran habitualmente terminos vagos e imprecisos (tales como numeros grandes, poco combustible, etc.). El valor de verdad asignado a la proposición difusa P, puede ser cualquiera en el intervalo $[0,1]$ Resulta entonces el mapeo:

$$T: U \rightarrow [0,1]$$

Si se asigna una proposición difusa a un conjunto difuso en el universo del discurso. P se asigna al conjunto A, entonces la valuación de verdad, T(P), de la

proposición es:

$$T(P) = \mu_A(x) \text{ donde: } 0 \leq \mu_A(x) \leq 1$$

El grado de verdad de la proposición $P: x \in A$ es igual al grado de membresía del elemento x en el conjunto difuso A.

Los conectores lógicos se definen también para proposiciones difusas, pero su valuación se obtiene mediante cálculos numericos y no solo simbólicos como es el caso de la lógica clasica.

Definase P en el conjunto difuso A, Q en el conjunto difuso B.

Las operaciones siguientes se definen asi:

Negación: $T(\bar{P}) = 1 - T(P)$

Disyunción: $P \vee Q \rightarrow x$ es A o B : $T(P \vee Q) = \max(T(P), T(Q))$

Conjunción: $P \wedge Q \rightarrow x$ es A y B : $T(P \wedge Q) = \min(T(P), T(Q))$

Implicación: $P \Rightarrow Q \rightarrow$ Si x es A, Entonces x es B:

$$T(P \Rightarrow Q) = T(\bar{P} \vee Q) = \max\left(T(\bar{P}), T(Q)\right)$$

La implicación puede modelarse en una forma a base de reglas:

$P \Rightarrow Q$ es: Si x es A, Entonces y es B

y equivale a la relación difusa R:

$$R = (A \times B) \cup (\bar{A} \times Y)$$

su función de membresía se expresa con la siguiente fórmula:

$$\mu_R(x, y) = \max\left[\left(\mu_A(x) \wedge \mu_B(y)\right), \left(1 - \mu_A(x)\right)\right]$$

Como puede apreciarse, la relación entre la lógica difusa y la teoría de los conjuntos difusos es la misma que entre lógica y teoria de conjuntos.

Controladores Difusos y Sistemas de Inferencia

El control de procesos, como ya se mencionó, fue la primera aplicación práctica de la lógica difusa. Con el tiempo esta se ha convertido en una herramienta muy útil cuando las leyes de control de un proceso son difíciles de obtener o de expresar en forma matemática. También cuando el control debe ser no lineal, o que este sometido a variables no medibles, pero susceptibles de estimación. La lógica difusa, pues, se presenta como un recurso útil para diseñar controladores de procesos o sistemas que involucran incertidumbre o fuertes no linealidades.

De manera formal, un sistema de control es una relación que se establece entre dos conjuntos de variables; las variables de entrada y de salida del sistema. En este sentido se interpreta a un sistema de control como el mapeo entre los dos conjuntos de variables, de manera tal que el sistema sea capaz de llevar el proceso particular controlado a los estados apropiados definidos por la aplicación específica. La lógica difusa permite crear sistemas de control a partir de la combinación de reglas y conjuntos difusos aún si no se tiene un conocimiento profundo del modelo matemático del sistema a controlar. Dada su capacidad de representar incertidumbres, les posible describir el proceso y las leyes de control por medio de reglas de "sentido común" que incluyan cantidades aproximadas. Con reglas de la forma SI . . . , ENTONCES se puede definir la estrategia de control adecuada de acuerdo con la experiencia de un experto. Conociendo la estrategia, se puede traducir en un algoritmo que incluya reglas y conjuntos difusos, es decir, las decisiones que se toman en función de los diferentes estados de un proceso, se toman a partir de datos cualitativos y no cuantitativos.

Variables Lingüísticas

De lo anterior, resulta que necesitamos un medio de manejar datos cualitativos, este medio son las *variables lingüísticas*. Estas son variables cuyos valores son palabras o sentencias expresadas en un lenguaje natural o formal. Las sentencias o palabras se representan con conjuntos difusos; así por ejemplo, la variable *temperatura* puede tomar los valores *frío, templado, caliente, muy caliente*, etc. A estos últimos valores se les representa como conjuntos difusos en un mismo universo del discurso. El número y el rango de los conjuntos difusos se eligen de manera que los representen adecuadamente según el criterio de diseño y la experiencia. Estos conjuntos cumplen todas las propiedades enunciadas en las secciones precedentes.

La estructura básica de un sistema de control difuso consta de tres etapas: etapa de *fuzzificación*, de *evaluación de reglas* y de *defuzzificación*. En estos sistemas de control, si bien el proceso se planifica de forma cualitativa, en la práctica se llevan a cabo apelando a las facultades de cálculo numérico que nos proveen los sistemas de cómputo actuales. En una sección anterior se ha mencionado ya esta facultad de la lógica difusa: es susceptible tanto de cálculo simbólico como numérico. Las funciones de membresía son los medios que relacionan las valores numéricos reales de una variable con los valores numéricos que representan la pertenencia a un determinado conjunto difuso y que son susceptible de manipulación de acuerdo a las reglas y las relaciones lógicas difusas

que se establecen. Las operaciones son simples y pueden ser ejecutadas por dispositivos como microcontroladores o pequeños microprocesadores.

La figura siguiente muestra la estructura de un sistema de control difuso.

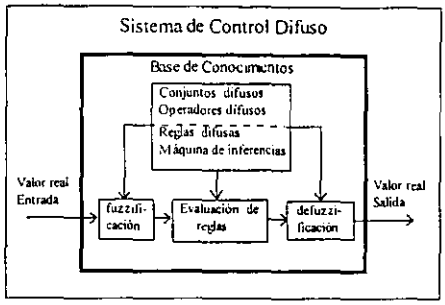

FIGURA 1.4 Sistema de Control Difuso.

Se presenta en seguida una descripción de cada etapa de un sistema de control difuso.

Fuzzificación

La primera etapa del sistema de control consiste en traducir los valores reales provenientes del sistema, en valores difusos. Es decir transformar las variables numéricas en variables lingüísticas. Esto se hace mediante las funciones de membresía. El valor de entrada se considera un elemento en un determinado universo del discurso, que es al mismo tiempo miembro de algún o algunos subconjuntos difusos del mismo universo. El grado en que es miembro de cada conjunto esta dado por su función de membresía correspondiente. Los valores de las funciones de membresía son suficientes para formar una representación de la variable lingüística. A cada variable de entrada se le representa con una variable lingüística, y en cada caso se obtienen todas las funciones de membresía, de acuerdo con los conjuntos difusos de que sea miembro en un grado mayor a cero.

El proceso se puede ejemplificar con la variable lingüística que se presenta en la figura 1.5:

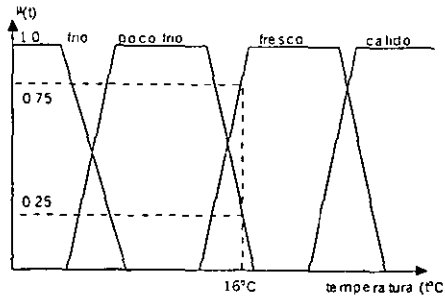

FIGURA 1.5 Ejemplo de fuzzificación.

El universo del discurso corresponde a la variable *temperatura*, los valores reales en el eje horizontal son los valores numéricos de la variable. La representación de los valores lingüísticos es mas complicada, el eje vertical representa el grado en que cada elemento del universo del discurso pertenece al conjunto difuso (o valor lingüístico) que se indica sobre este. El conjunto de pares o puntos que se forman las figuras constituyen las función de membresía. Así a cada elemento X corresponde por lo menos un punto que es el primer elemento de los pares ordenados en una función de membresía. Así para el elemento X=16°C, su valor numérico es 16 pero su valor difuso lo determinan las funciones de membresía de los conjuntos *poco frío* y *fresco*. Según la representación gráfica a cada punto en el eje horizontal corresponde un par ordenado $(X,\mu_i(X))$ donde i representa un valor lingüístico, μ la función de membresía. Para este caso los pares para las funciones de membresía en el punto dado son:

(16, 0.3)

(16, 0.7)

Kosko ha propuesto hipercubos como representación geométrica de los conjuntos difusos en un universo del discurso, sin embargo, para propósitos prácticos de control basta con definir el conjunto de pares ordenados de la forma $(i,\mu_i(X))$, para obtener una representación útil del valor, en un estado del sistema, de la variable lingüística en cuestión. Así que la variable lingüística, para este caso, se representa por el conjunto de pares ordenados cuyas funciones de membresía son mayores que cero:

(poco frío, 0.3)

(fresco, 0.7)

La obtención de todos los pares de este tipo para cada variable concluye la etapa de fuzzificación.

Evaluación de reglas

Con los datos de entrada ya en forma difusa, se pasa a la etapa que se encarga de decidir las acciones de acuerdo con la estrategia y el algoritmo establecido. Esta etapa involucra dos partes: una *base de reglas difusa* y una *máquina de inferencias difusa*.

Base de reglas

Esta parte del sistema de control es donde reside propiamente el algoritmo a ejecutar. Se elabora con el conocimiento de un experto en el área de conocimiento del problema a solucionar. El conocimiento de la estrategia a seguir se codifica en la forma de una serie de sentencias condicionales o reglas, expresadas de la forma: *SI un conjunto de condiciones se satisfacen, ENTONCES un conjunto de acciones se ejecutan.* La serie de sentencias implementa las estrategias de control a partir del conocimiento que se dispone acerca del proceso a controlar. A la parte condicional se le denomina antecedente y a la parte de acción se le denomina conclusión o consecuente de la regla. El antecedente de una regla de control difuso expresa el estado en que deben encontrarse las variables del proceso para activarla, mientras que su conclusión especifica los cambios que deben llevarse a cabo cuando se activa dicha regla.

Los antecedentes de cada regla se pueden combinar con los operadores lógicos básicos ya conocidos: conjunción disyunción o negación. Por ejemplo, para un sistema de dos entradas y una salida, una regla condicional puede ser: *SI X es* A *y Y es* B, *entonces Z es* C; donde existe un universo del discurso independiente para cada una de las variables *X*, *Y* y *Z*. En cada universo del discurso deben especificarse los valores lingüísticos que asumirá cada variable en forma de conjuntos difusos adecuados. Por ejemplo, para *X* los conjuntos serán $A_1, A_2, \ldots A_n$; para Y : $B_1, B_2, \ldots B_m$; y para Z: $C_1, C_2, \ldots C_p$. La distribución en el universo del discurso y la forma de la función de membresía se determinan de acuerdo a la experiencia, o bien usando técnicas de adaptación como las que se discutirán en un capítulo ulterior.

Con la estructura de las variables lingüísticas definidas, la construcción de reglas se realiza eligiendo combinaciones apropiadas de conjuntos difusos de entrada y salida, relaciones que constituyen una estimación difusa que expresa cómo la salida depende funcionalmente de la entrada sin requerir de una descripción matemática precisa. Una regla difusa como: *SI A ENTONCES B*, representa un mapeo, una relación difusa entre una parte del universo de la variable de entrada con una parte del universo de la variable de salida. Al antecedente A de la regla se le denomina *asociante de entrada*, mientras que a la conclusión B de la regla se le denomina *asociante de salida.* El antecedente expresa los diversos estados por los que transita la variable de control, mientras que la conclusión expresa el incremento o decremento que debe sufrir la variable a manipular para reducir la desviación o error.

Máquina de Inferencias

Es un interprete de la base de reglas. Su tarea consiste en calcular una conclusión numérica a partir de los valores de entrada, esto de acuerdo a la estructura de las variables, de la base de reglas y las operaciones definidas en la lógica difusa. Es, entonces la parte que se encarga de "razonar" de acuerdo al conocimiento explícito en la base de reglas. Para ello la máquina de inferencias utiliza un *mecanismo de inferencias difuso*, que es una lógica de toma de decisiones que emplea el conocimiento codificado en la base de reglas para obtener una conclusión difusa correspondiente a los valores que han ingresado al sistema de control y que han pasado, previamente, por la etapa de fuzzificación.

Las maneras de realizar inferencias en lógica difusa son dos: el *Modus Ponens Generalizado* y el *Modus Tollens Generalizado*, que son las formas difusas del Modus Ponens y el Modus Tollens. En seguida se muestra como se conforman:

Modus Ponens Generalizado:

premisa 1: X es A'

premisa 2: si X es A entonces Y es B

consecuencia: Y es B'.

Modus Tollens Generalizado:

premisa 1: Y es B'

premisa 2: si X es A entonces Y es B

consecuencia: X es A'.

Donde A, A', B y B' son los valores lingüísticos representados por conjuntos difusos de la variables lingüísticas X y Y. En los sistemas de control difuso el proceso de inferencias generalmente se realiza utilizando implicaciones del tipo Modus Ponens Generalizado.

Un ejemplo de la aplicación de este criterio para un sistema de dos entradas y una salida es:

entrada: X es A' y Y es B'

R_1 : si X es A_1 y Y es B_1 , entonces Z es C_1

o R_2 : si X es A_2 y Y es B_2 , entonces Z es C_2

o ... ...

o R_n : si X es A_n y Y es B_n , entonces Z es C_n

consecuencia: Z es C'

Donde X y Y son las variables lingüísticas de estado y Z es la variable lingüística de control. A_i, B_i, y C_i son los valores de las variables X, Y y Z, respectivamente, en diferentes universos del discurso U, V y W, con i=1, 2,...,n, y con todas las variables ligadas con el conector lógico de conjunción. Todo esto constituye la base de reglas.

El modo de implementar lógicamente una regla de control difuso de la forma *si X es A_i y Y es B_i ,entonces z es C_i*; es a través de una implicación que es una relación difusa R_i, que se define como:

$$\mu_{R_i} = \mu_{(A_i \wedge B_i \Rightarrow C_i)}(u, v, w) = \min\left(\mu_{(A_i)}(u), \mu_{(B_i)}(v)\right) \Rightarrow \mu_{(C_i)}(w)$$

donde:

$A_i \wedge B_i$ es un conjunto difuso en $A_i \times B_i$ en $U \times V$.

$R_i \equiv (A_i \wedge B_i) \Rightarrow C_i$ Es la relación difusa de implicación en $U \times V \times W$.

$\Rightarrow$ denota, en este trabajo, implicación difusa.

En general un sistema de control difuso (SCD) procesa en paralelo un número de reglas n: R_1, R_2,..., R_n. Así es que, para este caso ejemplificativo, para cada par de entradas A y B que ingresan al sistema se activa sistemáticamente cada regla almacenada en la base de conocimiento pero en grados diferentes. De tal manera que la salida del sistema , si la conclusión de cada regla es C_1, C_2,..., C_n; es igual a la suma de los grados parciales de verdad de cada regla, esto es:

$$C' = w_1 C_1 + w_2 C_2 + \ldots + w_n C_n$$

donde w_i refleja el grado de credibilidad, validez o influencia de la regla R_i en la conclusión difusa C'.

Para obtener el grado de verdad o grado de influencia w de cada regla sobre la conclusión difusa C', el mecanismo de inferencias dispara, a partir de los valores difusos que provienen de la etapa de fuzzificación, aquellas reglas que tengan en su antecedente un valor difuso cuyos pares ordenados incluyan algún valor de función de membresía mayor que cero.

Una vez obtenido el grado de influencia w para cada regla, se pueden utilizar dos tipos de razonamiento para obtener la conclusión difusa C': *correlación-mínimo* y *correlación-producto*. Para explicar en que consiste cada razonamiento considérese el siguiente ejemplo: se tiene una base de reglas con las dos siguientes reglas de control:

R_1 : si X es A_1 y Y es B_1 , entonces Z es C_1

R_2 : si X es A_2 y Y es B_2 , entonces Z es C_2

los grados de influencia w_1 y w_2 de dichas reglas en la conclusión C' se expresa de la siguiente manera:

$$w_1 = \mu_{A_1}(X_0) \wedge \mu_{B_1}(Y_0)$$

$$w_2 = \mu_{A_2}(X_0) \wedge \mu_{B_2}(Y_0)$$

donde $\mu_{A_1}(X_0)$, $\mu_{B_1}(Y_0)$, $\mu_{A_2}(X_0)$ y $\mu_{B_2}(Y_0)$ representan el grado en que X_0 y Y_0 pertenecen a lo conjuntos difusos A_1, B_1, A_2 y B_2 respectivamente.

Razonamiento correlación-mínimo

También llamado *mínimo de Mandani*, considera a cada regla como una implicación difusa R_i, en donde la i-ésima regla conduce a la conclusión:

$$\mu_{C_i'}(W) = w_1 \wedge \mu_{C_1}(W)$$

lo que implica que la función de membresía μ_C de la consecuencia inferida C esta dada por:

$$\mu_{C_1}(W) = \mu_{C_1'} \vee \mu_{C_2'} = \left(w_1 \wedge \mu_{C_1}(W)\right) \vee \left(w_2 \wedge \mu_{C_2}(W)\right)$$

Razonamiento correlación-producto

Llamado también *producto de Larsen*, considera a cada regla como una implicación difusa R_i, en este caso la i-ésima regla conduce a la conclusión:

$$\mu_{C'_i}(W) = w_i \cdot \mu_{C_i}(W)$$

lo que implica que la función de membresía μ_C de la consecuencia inferida C esta dada por:

$$\mu_C(W) = \mu_{C'_1} \vee \mu_{C'_2} = \left(w_1 \cdot \mu_{C_1}(W)\right) \vee \left(w_2 \cdot \mu_{C_2}(W)\right)$$

Etapa de defuzzificación

Los resultados de la etapa previa son de tipo difuso, por lo que para obtener un valor numérico utilizable en los transductores y actuadores convencionales, es necesario transformarlos mediante esta última etapa.

El proceso de defuzzificación consiste en transformar las salidas del sistema de control difuso de un espacio de accionés de control definidas dentro de un universo del discurso difuso a acciones definidas dentro de un universo del discurso no difuso. Es decir, es el proceso de combinar la conclusión de todas las reglas disparadas durante el proceso de evaluación de reglas, en una única acción con un valor específico y concreto que sale del sistema para ejecutar la acción de control.

Los tres métodos más comunes para realizar este proceso son: a) *criterio del máximo*, b) *promedio de máximos*, y c) *centro de gravedad*.

a) Método del criterio del máximo.

Este criterio selecciona como salida, de entre las reglas disparadas durante el proceso de evaluación de reglas, la conclusión de aquella que tenga el máximo grado de verdad o de influencia sobre la conclusión difusa C'

Este método es el más sencillo , pero también es el menos usado, debido a que sólo considera aquella regla que posee el máximo grado de verdad, ignorando la posible contribución de las demás reglas. Por otra parte, en el caso de que dos reglas posean un mismo valor w, y éste coincida con el máximo, se crea un conflicto que debe ser solucionado con otro criterio codificado en otra subrutina.

b) Método del promedio de máximos.

Esta técnica genera una acción de control que representa al valor promedio de todas las acciones de control locales dadas por aquellas reglas disparadas durante el proceso de evaluación de reglas. Su expresión matemática es la siguiente: $Z_0 = \frac{\sum_{i=1}^{n} w_i}{n}$

donde w_j es el grado de influencia asignado a cada regla, y n es el número de reglas tomadas en cuenta.

c) Método del centro de gravedad.

De estos tres métodos este es el mejor. Este considera la contribución y el grado de influencia de cada regla disparada durante el proceso de evaluación de reglas. Este

método consiste de varias etapas, las que se describen a continuación.

1.- Truncar cada uno de los conjuntos difusos de las funciones de membresia de salida en el punto dado por el grado de influencia de aquellas reglas disparadas durante el proceso de evaluación de reglas y que, por tanto, son tomadas en cuenta para realizar el proceso de restricción.

2.- Calcular el área correspondiente a cada uno de los conjuntos truncados en el paso previo.

3.- Calcular el punto en el eje x correspondiente al centro de gravedad de cada conjunto truncado.

4.- Calcular el centro de gravedad correspondiente al eje x con la expresión siguiente:

$$Z_0 = \frac{\sum_{i=1}^{n} A_i x_i}{\sum_{i=1}^{n} A_i}$$

donde:

Z_0 = centro de gravedad en el eje x (acción de control concreta)

A_i = área del conjunto difuso truncado i

x_i = componente x del par ordenado del centro de gravedad en el conjunto difuso truncado i

$i = 1,2,...,n$

n = número de los conjuntos difusos truncados

Existe una forma simplificada para obtener el centro de gravedad que se basa en el resultado demostrado por Bart Kosko, en el sentido de que se puede calcular el centro de gravedad global a partir de los centros de gravedad locales de cada conjunto difuso. El resultado se expresa como sigue:

$$Z_0 = \frac{\sum_{i=1}^{n} w_i C_i I_i}{\sum_{i=1}^{n} w_i I_i}$$

donde:

Z_0 = centro de gravedad global en el eje x (acción de control concreta)

w_i = grado de influencia para la regla i

C_i = centro de gravedad con respecto al eje x del conjunto difuso i

I_i = área del conjunto difuso i

$i = 1,2,...,n$

n = número de reglas disparadas durante el proceso de evaluación de reglas

En la siguiente figura se muestra la interpretación gráfica de los diversos métodos de defuzzificación para el caso de dos reglas disparadas durante el proceso de evaluación de reglas.

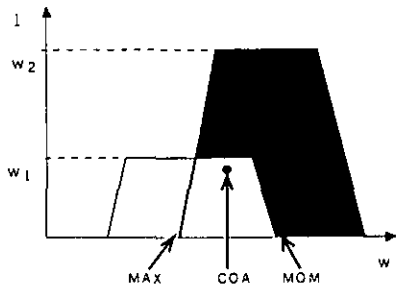

FIGURA 1.6 Interpretación grafica de los metodos de defuzzificación.

CAPITULO II : REDES NEURONALES

ANTECEDENTES

Historia

La investigación de sistemas de procesamiento paralelo de información datan de hace mas de 50 años. Las primeras ideas para tales mecanismo pretendían imitar el funcionamiento de la mente humana. El camino mas plausible parecía ser el imitar al órgano que la genera: el sistema nervioso central. Las investigaciones decayeron durante años y actualmente divergen en varios aspectos. Cabe aquí presentar un bosquejo del estado de las investigaciones en sus diversas etapas.

Década de 1940:

McCulloch y Pitts

Diseñaron lo que generalmente se considera como la primera red neuronal, reportándolo en su trabajo de 1943. Estos investigadores reconocieron que combinando muchas unidades básicas de procesamiento (*neuronas*) en sistemas reticulares, la capacidad de computo se incrementaba. Los *pesos* (intensidades de las conexiones entre nodos) de una sistema McCuloch-Pitts son ajustados de modo que la neurona ejecuta un función lógica simple y particular; y con diferentes neuronas ejecutando diversas funciones lógicas. Las neuronas pueden ser arregladas en una red de tal suerte que producen una salida que puede ser representada como una combinación de funciones lógicas. El flujo de información por la red se hace solo en tiempos discretos, en los intervalos las señales se desplazan de una unidad de proceso a la siguiente. Este retardo de tiempo les permitió modelar algunos proceso psicológicos (su investigación se enfocaba al modelado de la fisiología neuronal), como la percepción de diferencias de temperatura.

McCulloch y Pitts fueron los primeros en proponer e implementar la idea del *umbral de activación*, que es utilizado en la gran mayoría de las redes neuronales hoy en día. Esta idea consiste en definir un número (el umbral de activación), si la suma ponderada de entradas provenientes de la red es mayor que el umbral, entonces la neurona en cuestión se enciende o se activa. Las redes de McCulloch y Pitts se usaron principalmente como circuitos lógicos (binarios), una suerte de versión *sui generis* de los actuales circuitos lógicos secuenciales.

Los subsecuentes trabajos de McCulloch y Pitts señalaron a cuestiones que todavía son áreas importantes de investigación; tales como el reconocimiento invariante de patrones ante traslaciones y rotaciones.

Donald Hebb

Psicólogo de la Universidad de McGill; afiliado a la escuela conductista y precursor del cognitivismo, diseñó el primero método de aprendizaje para redes neuronales. Propuso que si dos neuronas se activan simultáneamente, entonces la conexión entre

éstas debe reforzarse. Posteriormente refinó la idea de modo que permitió hacer simulaciones en computadora.

Décadas de 1950 y 1960

John Von Neumann estuvo profundamente interesado en modelar el cerebro. Estuvo relacionado con los investigadores pioneros en el estudio de redes neuronales, e insinuó direcciones alternativas para el desarrollo de computadoras, aunque su conocido modelo terminó por imponerse.

Perceptrones

Frank Rosenblatt, junto con otros investigadores, introdujo y desarrolló una clase de redes neuronales artificiales llamadas *perceptrones*. El perceptrón más típico consistía de una capa de entrada (simulando la retina del ojo) conectada por caminos con pesos fijos que conducían a las neuronas asociadoras, los pesos en los caminos de conexión eran ajustables. El algoritmo de aprendizaje del perceptrón consiste en ajustes iterativos de los pesos sinápticos, y es mas poderosa que el del aprendizaje hebbiano (usando en general el error que comete la red respecto de una salida deseada, en lugar de solo reforzar conexiones donde la activación es mutua). Se ha probado que el aprendizaje del perceptrón converge al conjunto correcto de pesos si existe el conjunto que resuelva el problema en cuestión. El modelo del perceptrón incorpora la idea del umbral del modelo McCulloc-Pitts.

El éxito inicial del perceptrón condujo a un entusiasmo desmedido. Sin embargo la prueba matemática inicial de la convergencia del aprendizaje iterativo en perceptrones, fue seguida de una demostración, por Minsky y Papert en 1969, de sus limitaciones. Los resultados mostraban que la red tipo perceptrón puede aprender solo problemas linealmente separables, que son problemas de clasificación que pueden representarse por medio de combinaciones lineales de los pesos de la red y las entradas.

ADALINE

En 1960 Bernard Widrow y Marcian Hoff desarrollaron una regla de aprendizaje que se designa regla del *mínimo cuadrado medio (least mean square)* o *regla delta*, que está estrechamente relacionada a la regla de aprendizaje del perceptrón. La regla del perceptrón ajusta los pesos de conexión siempre que la respuesta de la red es incorrecta (la respuesta induce una clase a la que pertenece la entrada). La regla delta ajusta los pesos para reducir la diferencia entre la entrada de la red a la unidad de salida y la salida deseada . Esto da como resultado el *mínimo error medio.* La semejanza de los modelos desarrollados en psicología por Rosenblatt y en ingeniería eléctrica por Widrow y Hoff revela el carácter interdisciplinario del desarrollo de las redes neuronales. La diferencia en las reglas de aprendizaje, aunque tenues , condujeron a una mejora de la habilidad de generalización de la red. La regla de Widrow y Hoff para redes de una sola capa, es la precursora de la regla o algoritmo de retropropagación para redes multicapa.

El trabajo de Widrow y sus estudiantes se ha reportado, unas veces como "investigación en redes neuronales" y, otras como en sistemas lineales adaptativos. Este tipo de redes se denominan *ADALINE*, nombre que deriva del inglés *Adaptive Linear Neuron* o bien *Adaptive Line System.* Este modelo se ha aplicado a problemas de control, tales como enganchar remolques o balanceo de péndulos invertidos.

De este modelo deriva uno mas complejo y más poderoso, resultado de una arquitectura de capas múltiples, denominado MADALINE.

Década de 1970

La presentación de la demostración de Minsky y Papert acerca de las limitaciones del perceptrón, provocó una baja en el interés de los investigadores. En ésta década la investigación continuó, pero solitaria y silenciosa. No obstante, algunos de los investigadores ahora "clásicos" empezaron sus investigaciones durante esta época.

Kohonen

Teuvo Kohonen, de la Universidad Tecnológica de Helsinki, hizo sus primeros trabajos (1972) acerca de memorias asociativas con redes neuronales. En 1982 desarrollo mapas característicos auto-organizativos, que aprovechan una estructura topológica para unidades que funcionan formando cúmulos de patrones. Estas se han usado para problemas interesantes; como reconocimiento de voz o composición automática de música.

Anderson

James Anderson, de la Brown University, también comenzó investigando redes neuronales en memorias asociativas. Desarrolló sus ideas hasta `postular su modelo *brain-state-in-a-box*. Consiste en truncar la salida lineal de los primeros modelos de redes neuronales, para evitar que las iteraciones del algoritmo de aprendizaje conduzcan a una respuesta muy grande; logrando de ese modo que la red converja a una solución estable. Su modelo se ha aplicado al diagnóstico médico y al aprendizaje de tablas de relaciones.

Grossberg- Carpenter

Stephen Grossberg, es acaso el investigador de redes neuronales mas prolífico, sus muchas investigaciones se centran en la parte básica de la disciplina: matemáticas y biología.

Gail Carpenter, junto con Grossberg desarrolló la teoría de las redes neuronales auto-organizativas conocida como *Teoría de la Resonancia Adaptativa*: ART por sus iniciales en inglés (*Adaptive Resonance Theory*).

Década de 1980

Retropropagación

En la década de 1970, los modelos de procesamiento serial se verían beneficiados por los avances en la técnica de construcción de circuitos de estado sólido, mientras que la investigación en redes neuronales se experimento un desaliento por dos motivos. Primero; su incapacidad de resolver problemas que implican mapeos sencillos, como las función Or-Exlcusiva. Segundo; por la ausencia de un método general para entrenar redes con varias capas de neuronas.

Un método para propagar información acerca de los errores en las unidades de salida de vuelta a las unidades ocultas, fue descubierto en la Década anterior (P. Werbos en 1974), pero no era muy popular. Fue, además, redescubierto independientemente, antes de que fuera ampliamente conocido, por David Parker en 1985 y por LeCun en 1986. Es muy similar a un algoritmo, aún mas antiguo, usado en teoría de control óptimo

.El trabajo de Parker llamó la atención del grupo de investigación en procesamiento paralelo y distribuido, de la Universidad de California, San Diego; dirigido por el psicólogo David Rumelhart, este grupo, junto con James McClenlland de la Carneigie-Mellon University, lo refinaron y publicaron en 1988, desarrollando un método estándar de entrenamiento para redes de capas múltiples: el algoritmo de retropropagación.

Redes Hopfield

John Hopfield, del California Institute of Technology y premio Nobel de Física, jugó un papel importante en el resurgimiento del interés por las redes neuronales. Junto con David Tank, investigador de AT&T, e inspirado por su trabajo en materiales magnéticos; Hopfield desarrollo varias redes neuronales basadas en pesos sinápticos fijos y umbrales de activación adaptativas que pueden funcionar como memorias asociativas. La publicación de su trabajo en Scientific American (1987), así como la insistencia de Hopfield (Premio Novel de Fisica)en que si "se desea que las máquinas puedan hacer los que los humanos, es necesario estudiar la cognición humana"; ayudaron al resurgimiento del interés en esta área.

Hopfield llama la atención respecto del carácter mimético de los paradigmas neuronales, la siguiente sección bosqueja los rasgos de los sistemas neuronales biológicas que inspiraron las investigaciones resumidas aquí.

FUNDAMENTOS BIOLÓGICOS

Existe un analogía entre la estructura de una célula nerviosa o neurona y el elemento o unidad de procesamiento que constituye una Red Neuronal Artificial (RNA). La estructura de cada neurona individual varía mucho, dependiendo de su localización dentro del sistema nervioso y del organismo al que pertenezca, sin embargo, la *organización global* de dichas células muestra una menor variedad. Es precisamente esta estructura global la que es responsable (hay mucha evidencia a favor) de los complicados procesos en los sistemas nerviosos de los seres vivos, y es también lo mas interesante si se planea entender y repetir artificialmente estos procesos.

Pero para construir unidades de procesamiento artificiales, es instructivo conocer como operan sus exitosas contrapartes biológicas.

Una célula nerviosa tiene tres componentes principales que son cruciales para entender su operación y las de sus modelos artificiales: *dendritas* , *soma* y *axón*.

Las dendritas (del griego *dendron*, árbol) son ramificaciones de la célula nerviosa y son los elementos por los cuales una neurona recibe señales de otras mas. Las señales, dentro de una neurona, son flujos químicos, que dan como resultado la alteración del equilibrio iónico entre la parte interna y la externa de la neurona modificando el potencial eléctrico en la membrana celular.

Las señales se transmiten de una célula a otra, vía las uniones sinápticas por medio de un proceso químico, este proceso involucra la liberación de sustancias diversas (los neurotransmisores), desde la zona interna de la célula emisora (presináptica) en las sinápsis, hacia el exterior de la célula, donde posteriormente entran en contacto con ciertas zonas de la membrana de la célula receptora (célula postsináptica). Una vez allí desencadenan reacciones químicas que modifican de manera local la permeabilidad de la

membrana de la célula receptora, este cambio de permeabilidad permite que ciertos géneros de iones ingresen por difusión al interior de la neurona, provocando un cambio en el potencial de la membrana. Debido a las peculiaridades de los neurotransmisores, resulta que cada proceso de transmisión tiene una calidad específica. La acción del transmisor químico modifica la señal entrante (generalmente escalando la frecuencia de las señales recibidas), análogamente a la forma como se modifica las señales entrantes a una neurona artificial.

El *soma*, es el cuerpo de la célula (véase la figura 2.1). En la zona del soma donde se localiza la base del axón, denominada *montículo del axón*, tiene lugar un proceso de *suma* de los potenciales eléctricos resultado de las señales que entran. Cuando se recibe bastantes señales de intensidad suficiente, el potencial en el montículo del axón rebasa cierto umbral y se desencadena una reacción que conduce a una modificación de las características de la membrana celular de la neurona. Esta membrana se torna más permeable a los iones de sodio, que fluyen al interior de la célula provocando un cambio aún mayor en el potencial eléctrico, este último potencial se denomina *potencial de acción*. Se dice que la neurona se activa cuando esta presente éste potencial de acción. Debe hacerse notar aquí que este potencial de acción se establece como una diferencia con respeto al estado de homeóstasis de la célula, durante el cual se establece un potencial eléctrico entre la parte interna de la membrana celular y la parte externa, fijado por la permeabilidad de la membrana y denominado potencial de reposo. La magnitud de la diferencia no es fija y eso parece ser una característica relevante para la operación total del sistema. Pero, para un modelo simplificado, las células se pueden considerar como activadas o desactivadas en algún instante de tiempo. Esta simplificación permitió el desarrollo de los primeros modelos neuronales en los años cuarenta.

El *axón* es la parte por la cual una neurona difunde su estado de activación a las demás neuronas con las que tienen uniones sinápticas. La transmisión de la señal desde una neurona particular no se lleva a cabo por el fenómeno de conducción eléctrica, pues los materiales biológicos que la componen presentan poca conductibilidad, en general las células nerviosas son pésimos conductores. Lo que en realidad fluye por el axón son reacciones químicas desatadas en cadena. En la figura 2.1 puede notarse que el axón esta cubierto por una sustancia denominada mielina que forma vainas a su alrededor, el conjunto de esta sustancia conforma la materia blanca del cerebro. Estas vainas de mielina no son uniformes a lo largo de todo el axón, sino que presentan interrupciones, estas interrupciones son los nodos de Ranvier. Cuando una neurona se activa, la membrana celular en la base del axón pasa del potencial de reposo al potencial de acción, los nodo de Ranvier son particularmente sensibles a este cambio, cuando se presenta el potencial de acción, cada nodo reacciona modificando todavía mas su polarización, esta polarización afecta al siguiente nodo y este a su vez al próximo. De este modo el potencial de acción se transmite a lo largo del axón. Cuando los nodos de Ranvier han transmitido el potencial de acción, no pueden volver a su estado original sino después de cierto tiempo, de modo que no pueden excitarse ni reiniciarse la transmisión del potencial antes de 1 milisegundo, este intervalo se conoce como periodo refractario, y es el responsable de que la transmisión de señales en una neurona se limite a una frecuencia de 1000 impulsos nervios por segundo.

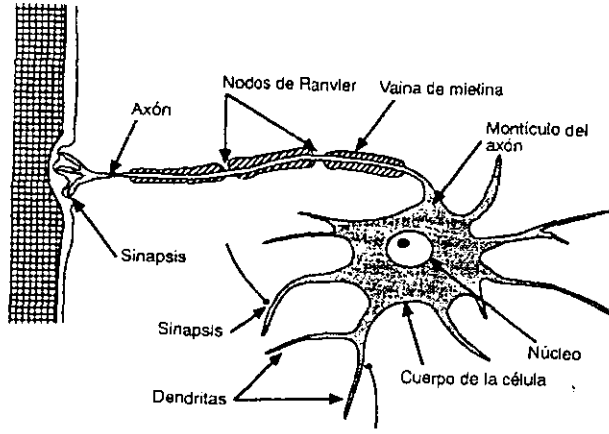

FIGURA 2.1 Representación esquemática de una neurona típica.

Los rasgos mas importantes de los elementos de procesamiento de las neuronas artificiales son tomadas de las propiedades de las neuronas biológicas. Las características que se toman de las células nerviosas para construir unidades básicas o nodos en redes neuronales artificiales son:

El elemento de procesamiento recibe diversas señales.

Las señales son modificadas por un peso o intensidad en la sinapsis receptora.

El elemento de procesamiento suma las entradas modificadas.

Cuando la suma de las entradas rebasa cierto umbral, la neurona transmite una salida única.

La salida de una neurona particular puede ir a muchas otras neuronas que se unen a esta por medio de las sinapsis en el axón.

El procesamiento de información es local.

La memoria es distribuida:

a) La memoria de largo plazo reside en las sinápsis de las neuronas.

b) La memoria de corto plazo corresponde a las señales enviadas por las neuronas.

La intensidad de las sinapsis puede ser modificada por la experiencia.

Los neurotransmisores para las sinapsis pueden ser excitatorios o inhibitorios.

Otra característica importante que comparten las neuronas biológicas con las artificiales es la tolerancia a fallas. Las neuronas biológicas son tolerantes en dos aspectos. Primero, los sistemas nerviosos naturales son capaces de reconocer muchas señales de entrada que son diferentes de cualquier señal que hayan experimentado antes. Segundo. el sistema nervioso es capaz de tolerar daños y destrucción parcial del sistema neurológico. Las células nerviosas tienen un número fijo desde el nacimiento y no

son remplazadas cuando mueren, sin embargo, el organismo es capaz de seguir aprendiendo aún cuando sobreviene la degradación fisiológica debido a la edad. Aún en casos de severo daño traumático, otras células son capaces de formar nuevos circuitos neuronales para suplir a los perdidos, y ser entrenadas de manera que aprendan las mismas funciones. De manera similar las RNA pueden ser diseñadas de modo que sean insensibles a pequeños "traumatismos" locales en la red; y la red puede ser reentrenada en caso de daño severo.

Aún cuando no se intente modelar sistemas nervioso biológicos por medio de RNA, el aproximar las redes a un modelo plausible puede conducir a mejorías en la capacidad de cómputo de la red. También se ha encontrado que los agrupamientos óptimos computacionalmente de RNA, corresponden a los aglomerados biológicos de neuronas. En el algoritmo de retropropagación (de que se hablará más adelante en este trabajo), se ha encontrado que separando la acción de retropropagación en la red en pequeñas piezas, para hacer el algoritmo más local (lo que es más plausible biológicamente), reditúa en un mayor poder de cómputo.

Si bien la fisiología neuronal fue la inspiración, no solo del particular cómputo paralelo de las redes neuronales artificiales, sino de casi todos los primeros intentos por lograr sistemas que pudieran realizar funciones lógicas, el enfoque actual se ha generalizado más y ahora se considera al sistema nervioso como un caso particular de modelos más generales.

Los desarrollos técnicos de cada época han estado siempre condicionados por el nivel y cantidad de conocimientos disponibles. Así, en la antigüedad las herramientas se diseñaban de conformidad con las experiencias e intuiciones de sus constructores: las válvulas flotantes y los relojes de agua son una verdadera hazaña del intelecto humano, si se considera el todavía limitado conocimiento empírico con que se contaba. Los reguladores de temperatura del siglo XVII, la máquina de vapor y las máquinas de combustión interna son progresivos desarrollos de la técnica todavía en parte empírica. Pero a partir del siglo XIX y sobre todo en el XX, el desarrollo de técnicas y herramientas esta cada vez más ligado al conocimiento científico. Con este conocimiento es posible proyectar y construir herramientas y dispositivos cada vez más complicados. La revolución industrial hizo preciso que se intensificarse el estudio de nociones mecánicas, lo que condujo a la introducción de una refinada noción de lo que conocemos como energía. Durante el presente siglo se introdujo un nuevo objeto de estudio: la información. Pronto se hizo evidente que la noción de información puede entenderse de modo independiente y que es susceptible de manipulación. Dispositivos para manipular información han sido proyectados y construidos desde mucho tiempo atrás, (Ramon Lulio, Blaise Pascal o Charles Babbage son ejemplos ilustres)

DESCRIPCIÓN DE LAS REDES NEURONALES ARTIFICIALES

Las redes neuronales se consideran, básicamente, modelos matemáticos de procesamiento de información. Al margen de diversos puntos de vista que las consideran según su actividad dentro de un sistema mayor como segmentos de software o elementos de hardware, cumpliendo funciones específicas. Estos modelos permiten implementar métodos para representar relaciones diversas.

Una *Red Neuronal Artificial* (RNA) es un sistema de procesamiento de información

que tiene ciertas características de funcionamiento comunes con las redes neuronales biológicas. Las RNA se desarrollaron como generalizaciones de los modelos matemáticos de la cognición humana. Se basan en los siguientes presupuestos:

- El procesamiento de la información se realiza en elementos muy simples denominados *neuronas* por su semejanza operativa con las células nerviosas, aunque también se les denomina *nodos, unidades* o *células*.
- Las señales se transmiten entre las neuronas a través de conexiones que las enlazan.
- Cada conexión tiene asociado un *peso* o *intensidad sináptica*, que representa la atenuación o amplificación de la señal, así como si esta señal excitará o inhibirá a la neurona receptora, típicamente se representa por la multiplicación por un factor adecuado.
- Cada neurona aplica una *función de activación* (que usualmente es una función no lineal) a la *entrada ponderada de la red* o *entrada de red* (la suma de las entradas escaladas por los pesos) para determinar su respuesta o señal de salida.

Tres son las características que determinan la índole de una red neuronal: 1) Su *arquitectura*, que es el patrón de conexiones entre sus neuronas. 2) Su *algoritmo de aprendizaje* o *algoritmo de entrenamiento*, que es el método por el cual determinan los pesos o intensidades sinápticas que comunican unas neuronas con otras. 3) Su función de activación.

Una red neurona consiste de una gran cantidad de elementos simples o neuronas. Cada neurona se conecta a otras por medio de eslabones de comunicación que tienen asociada un factor de escala que determina el grado en que tal conexión es excitatoria o inhibitoria; y a la que se denomina intensidad o peso sináptico. El conjunto estructurado y organizado de las neuronas y sus pesos adecuados por medio de la experiencia de la red, representa la información que usa ésta para resolver un problema.

Cada neurona posee un estado interno, denominado *activación* o *nivel de actividad*, el cual es función de las entradas que ha recibido. Habitualmente, cada neurona envía su nivel de activación como señal a las neuronas que mantienen sinápsis con ésta. La neurona envía una única señal en cierto instante, aunque esta señal se divulga a muchas otras neuronas.

En la siguiente figura se ejemplifica la formación de la señal de entrada a la neurona y_i.

x

$$y_i = w_1 x_1 + w_2 x_2 + w_3 x_3$$

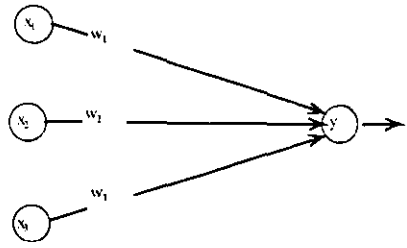

FIGURA 2.2 Entrada a una neurona *y*.

La activación *y* de la neurona *Y* está dada por una función sobre su entrada desde la red: $y = f(y_i)$, en general esta función es no lineal y sigmoide, ya que de otra forma, según la experiencia empírica, la red posee poca capacidad de computo, una función así es por ejemplo:

$$f(x) = \frac{1}{1+e^{-x}}$$

La neurona Y puede estar conectada a otras neuronas, por medio de sinapsis con sus respectivas intensidades asociadas, los cuales escalan las señales enviadas por *y*.

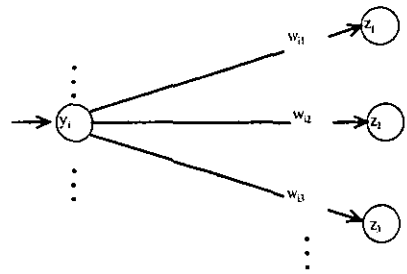

FIGURA 2.3 Distribución de la señal de la neurona y_i.

Este tipo de estructura, con uno o varios nodos intermedios (las neuronas *y* entre las *x* y las z), se conoce como multicapa o de capa oculta; aunado a la función de activación no lineal, le da a la red la capacidad de resolver problemas que no tiene una red con solo capas de entrada y salida (neuronas *x* y z), pero por otro lado su algoritmo de aprendizaje es más complicado.

CLASIFICACIÓN DE LAS RNA

Como se ha dicho arriba, tres son las características que definen una RNA. En cada uno de estos rasgos se han experimentado con modificaciones que pretenden adaptar la red a ciertas aplicaciones, o mejorar su rendimiento respecto de otros modelos.

Cada nueva modificación representa un nuevo intento, un nuevo individuo dentro de la especie de las RNA. El modelo particular cambia casi de aplicación en aplicación y cada investigador pretende mejorar e interpretar sus resultados, a fin de colaborar con el desarrollo de una teoría sólida para interpretarlos bajo un esquema general. En este trabajo se exponen solo las bases fundamentales para comprender el paradigma general de las RNA; pues una exposición completa y pormenorizada de cada especie implicaría un trabajo mucho más amplio que el pretendido aquí. Se expondrán los principios de acuerdo a las tres características fundamentales: Arquitectura, Algoritmo de aprendizaje y función de activación. Luego de esto se expone un modelo concreto que es ampliamente usado para aplicaciones de control: el perceptrón multicapa, que aprende por medio del algoritmo de retropropagación.

Arquitectura

Las neuronas se consideran ordenadas en estratos o capas. Cada capa esta conformada por las neuronas que reciben las señales de una fuente o fuentes comunes, como los nodos de entrada de una red o las activaciones de las neuronas de otra capa. Generalmente, las neuronas de una misma capa se comportan de manera similar, se observan reacciones uniformes para los mismos estímulos. Las factores determinantes de estas reacciones son su función de activación y el patrón de intensidades sinápticas por las cuales recibe y emite señales. En una sola capa, usualmente las neuronas poseen la misma función de activación y el mismo patrón de conexiones sinápticas. La manera como se conectan las neuronas es en general, la interconexión completa o la ausencia total de conexión. Es decir, una neurona en una capa no se conecta con la de otra, o bien, si se conecta con una neurona , se conecta con todas las neuronas de esa capa (salvo en algunos modelos de memorias autoasociativas o por crecimiento evolutivo).

El arreglo de las neuronas en capas y el patrón de conexión entre estas son los que constituye la arquitectura de la red. La figura 2.4 muestra una red que consiste de unidades de entrada, una unidad oculta y unidades de salida.

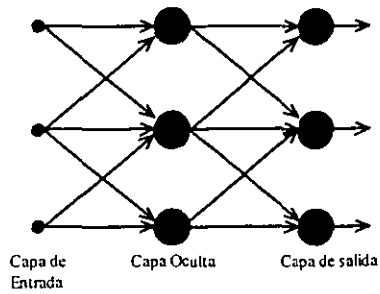

FIGURA 2.4 Representación de una red neuronal con una capa oculta.

Las redes en las cuales las señales fluyen desde las unidades de entrada a las de

salida, en una sola dirección, hacia adelante, se denominan redes de alimentación hacia adelante (*feedforward net*), la red de la figura anterior es un ejemplo. Las redes *recurrentes* (*recurrente* net) , son redes competitivas completamente interconectadas, y en las cuales los caminos de las señales forman un lazo de retroalimentación de una unidad con ella misma (en ciertos modelos la interconexión directa de una neurona con ella misma se suprime).

Red multicapa

Es una red con una o mas capas de nodos o neuronas(las llamadas unidades o capas ocultas). Usualmente hay una capa de pesos entre dos niveles adyacentes de unidades o capas (entrada, oculta, salida). Estas redes pueden resolver problemas más complicados que las redes de una sola capa , aunque entrenarlas es mas difícil. La creación de algoritmos eficientes para este tipo de redes dio un nuevo impulso a la investigación y aplicación del paradigma conectivista, que se había estancado durante años.

Entrenamiento: modos y algoritmos de aprendizaje

Como ya se ha dicho, una de las razones por las que los modelos biológicos de procesamiento de información resultan atractivos, es su capacidad de aprender y de asimilar experiencia continuamente; lo que los hace sumamente adaptables a su entorno.

El conocimiento se adquiere en base a experiencia, es decir, en base a situaciones y estímulos que se presentan al organismo; éste posee la capacidad de retener, a nivel celular, rasgos sobresalientes que representan aproximadamente la situación externa y la reacción interna del organismo, esta representación es el conocimiento que va fijándose en el sistema nervioso. Además existe la capacidad de modificar esta representación de acuerdo a los nuevos acontecimientos, de manera que las respuestas disponibles del organismo se tornen óptimas. El conocimiento en una RNA, se deposita también en base a las experiencias a que se somete el sistema, la experiencia se representa en una forma adecuada para ingresar a la red; habitualmente consiste en un conjunto de ejemplos representativos del problema; denominado *conjunto de entrenamiento*. Cada conjunto varia según las características de la red y de lo que desea que esta aprenda. El conocimiento que la red adquiere de su experiencia se representa como el patrón de conexiones y las intensidades o pesos de las sinapsis correspondiente en la totalidad de la red.

Conjuntamente con la arquitectura de la red, el método para ajustar los valores de los pesos, es una importante característica que distingue las diversas redes neuronales. Se distinguen dos tipos de entrenamiento: *entrenamiento supervisado* y *entrenamiento no supervisado*. Además existen redes cuya estructura, distribución y propiedades sinápticas son fijas y no se ajustan por un proceso iterativo de entrenamiento, pero estas tiene aplicaciones muy focalizadas, como modelado de órganos de percepción o generación de tablas.

Entrenamiento supervisado

Este es el modo usual como se entrenan la mayor parte de las RNA. El entrenamiento se lleva a cabo presentando una secuencia de ejemplos o patrones,

generalmente bajo la forma de vectores, cada uno de los cuales cuenta con vector asociado llamado *vector objetivo* o simplemente *objetivo* que representa la respuesta deseable para su correspondiente estímulo de entrada. Con el conjunto de vectores se procede a ajustar los pesos sinápticos de la red mediante un *algoritmo de aprendizaje*, el algoritmo de aprendizaje es el método por el cual los pesos sinápticos de la red se ajustan de manera que la salida se aproxima cada ves más a la deseada, sin que haya necesidad de un cálculo directo o de una determinación especulativa. Ya que las respuestas a cada estímulo se conocen de antemano, este tipo de entrenamiento se denomina *entrenamiento supervisado*.

Algunas de las más simples redes, están diseñadas para clasificar patrones, es decir, determinar si un vector de entrada pertenece o no a una categoría dada. En general se trata de una clasificación binaria , es decir, la red responde con una señal si el patrón es en efecto miembro de la categoría y con otra distinta si no lo es. Las redes tienen, entonces una única salida y se someten a un entrenamiento supervisado; la arquitectura dependerá de las características del problema, algunos problemas denominados *problemas linealmente separables*[1], pueden resolverse usando redes de capa sencilla mientras que los que no tienen esta características requieren de al menos una capa oculta. El algoritmo de aprendizaje se elige o se diseña de acuerdo a al complejidad del problema y de la arquitectura. En un apartado posterior se mostrará como se implementa el algoritmo de retropropagación.

La asociación de patrones es una forma especial de mapeo, en esta la respuesta esperada de la red no es binaria, sino que consiste en un patrón o vector. La RNA se entrena de tal modo que asocie un conjunto de vectores de entrada con un conjunto correspondiente de vectores de salida; a tal red se le denomina *memoria asociativa*. Si el patrón de salida deseado corresponde al mismo patrón de entrada, la red es un *memoria autoasociativa*. Si los vectores de entrada y los vectores objetivo son diferentes, se trata de una *memoria heteroasociativa*. Luego del entrenamiento, una memoria asociativa puede "recordar" un patón almacenado cuando a la entrada se le presenta un estímulo que se asemeje lo suficiente al vector que aprendió durante el entrenamiento.

Las redes de arquitectura multicapa pueden entrenarse para que ejecuten mapeos multidimensionales no lineales, dese un espacio n-dimensional de vectores de entrada a un espacio m-dimensional de vectores de salida.

Entrenamiento no supervisado

Las redes neuronales *autoorganizativas* agrupan vectores de entrada similares, sin usar datos de entrenamiento para determinar el aspecto que un miembro del grupo debe alcanzar (es decir sin usar vectores objetivo); para determinar a que grupo pertenece cada vector, ni la cantidad o calidad de los diversos grupos o categorías. Se le provee a la red con vectores de entrada, pero no de vectores objetivo. La red, valiéndose de un algoritmo adecuado, ajusta los pesos de manera que las entradas más similares se asignan a la misma salida o unidad de acumulación (*cluster unit*). La red producirá un vector representativo o *ejemplar* (*exemplar*), el término es usado por otros autores en sustitución de *target vector*. (vector objetivo) para cada *cúmulo (cluster)* formado.

---

[1]Problemas que pueden representar todos sus estados deseables con una relación de precedencia respecto de una función lineal en ese mismo espacio.

Redes de sinapsis o pesos fijos.

Otros tipos de redes pueden resolver problemas de optimización con restricciones (*constrained optimization problems*). Tales redes se desempeñan satisfactoriamente con problemas que causan dificultades a las técnicas tradicionales, problemas como los que incluyen restricciones de conflictos (no todas las condiciones pueden satisfacerse al mismo tiempo). En tales casos, frecuentemente, una solución próxima a lo óptimo es satisfactoria. Una RNA puede encontrar tal solución.

FUNCIONES DE ACTIVACIÓN

La operación básica de una unidad de procesamiento o neurona artificial, consiste, primero en la suma de sus entradas ponderadas o señales post-sinápticas (salvo en el caso de la capa de entrada) , luego, la aplicación, sobre esta suma, de una función de activación y, finalmente, la divulgación en la salida del resultado de esta función. Para las unidades de entrada, esta función es la función identidad $f(x)=x$. Típicamente, la misma función de activación es usada para todas la neuronas en una misma capa de la red. sin embargo esto no es estrictamente necesario. En la mayoría de los casos se usa una función de activación no lineal. Para explotar las ventajas que ofrecen las redes multicapa, comparadas con las limitadas posibilidades de las redes de capa simple, la funciones de activación deben ser forzosamente no lineales. Los resultados de alimentar una señal a través de dos o más capas de procesamiento lineal no difieren de los que se pueden obtener usando una sola capa con el elemento de escala adecuado.

Funciones de activación mas comunes.

Para la capa de entrada, como se menciono se utiliza la Función identidad

$f(x) = x$ para toda x.

Las redes de capa sencilla frecuentemente usan la función escalón para convertir la entrada de la red, que es una variable continua, a una unidad de salida que es una señal binaria o bipolar. En este caso es necesario usar un valor de umbral, si la entrada (la suma ponderada) a la neurona es menor que dicho umbral, la neurona permanece "apagada", es decir en estado bajo (sea binario o bipolar), pero si la entrada es superior al umbral la neurona se "enciende", es decir cambia al estado alto. La función escalón se conoce también como *función umbral* (*threshold function*), y se define así:

Función umbral , con umbral θ.

$$f(x) = \begin{cases} 1 & \text{si } x \geq \theta \\ 0 & \text{si } x \leq \theta \end{cases}$$

Las funciones sigmoides, son como su nombre lo indica curvas con forma de S. Son las funciones de activación mas útiles. La función logística y la tangente hiperbólica son las más comunes. Tiene la peculiaridad ventajosa de ser derivables en todo punto, lo que es condición necesaria en algoritmos como el de retropropagación.

La función logística es una función sigmóide que se extiende en el rango de 0 a 1. es usada como función de activación en redes cuyos valores de salida deseados son binarios o en el intervalo entre 0 y 1. A veces se le nombra sigmoide binaria y se define:

Sigmoide Binaria

$$f(x) = \frac{1}{1+e^{-\sigma x}}$$ (σ es un valor de escala)

$$f'(x) = \sigma f(x)\left[1 - f(x)\right]$$

La función logística puede ser escalada para abarcar cualquier rango de valores que sea apropiado a un determinado problema. El rango mas común es el que va de -1 a 1; si se escala de esta manera, la función se denomina sigmoide bipolar

Sigmoide bipolar:

$$g(x) = 2f(x) - 1 = \frac{2}{1+e^{-\sigma x}} - 1 = \frac{1-e^{-\sigma x}}{1+e^{-\sigma x}}$$

$$g'(x) = \frac{\sigma}{2}\left[1 + g(x)\right]\left[1 - g(x)\right]$$

Las tres características aquí expuestas son comunes a toda red neuronal, sin embargo la elección de las cualidades de la red depende mucho de lo que se pretenda hacer con esta. No hay un modelo de RNA que se pueda señalar como paradigma universal, como se ha dicho, una red neuronal es más un modelo abstracto, derivado de los sistemas nerviosos biológicos.

Cuando se está frente a problemas de control es más importante inspeccionar que relación puede establecerse entre los modelos neuronales y la teoría de control. Eso se discute en una sección posterior. Se explicará, así mismo, como trabaja un enfoque particular de las RNA, el perceptrón multicapa usando el algoritmo de retropropagación del error como método de entrenamiento.

EL PERCEPTRÓN MULTICAPA Y EL ALGORITMO DE RETROPROPAGACIÓN

El algoritmo de aprendizaje basado en la retropropagación del error es uno de los que con mayor frecuencia se ha usado para atacar problemas de control. Es muy extendido y es un buen ejemplo de la separación que existe entre el diseño de sistemas artificiales de procesamiento y su modelo biológico. Algunos autores han señalado que no es plausible una red nerviosa biológica que opere bajo los principios del algoritmo de retropropagación estándar; otros han propuesto modificaciones en la arquitectura y el algoritmo a fin de aproximarlo a los sucesos que son factibles en un sistema biológico. Sin embargo, este algoritmo ha demostrado su utilidad en aplicaciones de control. Aunque no puede reclamar el título del algoritmo más eficiente, su generalidad, su relativa sencillez para implementarse en código de computadora y su flexibilidad para adaptarse a mejorías en su algoritmo, establecen un compromiso atractivo para ser explotado en aplicaciones de ingeniería.

Arquitectura

El algoritmo se emplea con redes del tipo perceptrón con capas múltiples. En la gráfica se muestra una red de este tipo con dos capas ocultas. La red es *completamente conectada* (*fully connected*), cada neurona en una capa de la red esta conectada a todas las neuronas de la capa previa. La señal fluye desde los nodos de entrada a los de salida, pasando y modificándose en cada capa de la red.

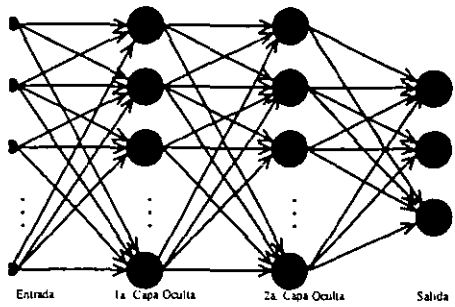

FIGURA 2.5 Representación de la arquitectura de un perceptrón con dos capas ocultas

El entrenamiento o memorización por el cual ha de pasar la red para adaptarse a una aplicación específica involucra tres fases (aunque algunos autores las reducen a dos): La alimentación de la señal de entrada o estímulo desde los nodos de entrada hasta los de salida, pasando por las capas intermedias; el cálculo del error asociado y la difusión de éste en sentido contrario (retropropagación) desde los nodos de salida a los de entrada; finalmente, el ajuste de los pesos o intensidades de las sinapsis. Después de que la red ha terminado su fase de entrenamiento, se utiliza propagando el estímulo hacia adelante.

Existen, por tanto dos tipos de señales que se propagan por la red:

a) *Señales de Función*: es el estímulo de entrada que se propaga por las capas ocultas y emerge en los nodos de salida como la respuesta de la red. Se le llama *señal de función* porque representa propiamente la respuesta, es decir la función de la red y, además, porque en cada neurona por la que la señal pasa, la señal es recalculada como función de las entradas y sinapsis asociadas que se aplican a la neurona.

b) *Señales de error*: La señal de error se origina en cada una de las neuronas o nodos de salida de la red. Y se propaga por capa(estrato) oculta hasta los nodos de entrada de la red. El cálculo de que cada neurona ejecuta involucra una función dependiente del error que se genera en la salida respecto de la respuesta esperada o deseada.

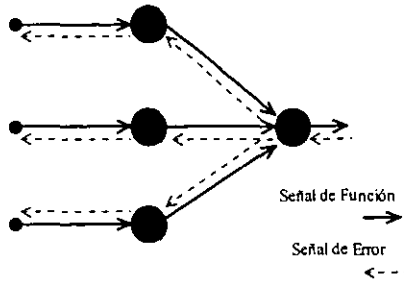

FIGURA 2.6 Direcciones del flujo de señales, difusión progresiva de las señales de función y propagación hacia atrás de las señales de error.

Las neuronas de salida constituyen la capa de salida de la red. Las neuronas restantes conforman las capas ocultas de la red, estas no forman parte ni de la entrada ni de la salida, de ahí que se les designe ocultas. La primera capa oculta es alimentada por las neuronas de entrada, conformada por unidades sensoras o de percepción ; las salidas resultantes de la primera capa oculta se aplican a las neuronas de la siguiente capa oculta (si existe), este proceso se repite hasta alcanzar la capa de salida.

Cada neurona en alguna capa oculta o en la capa de salida ejecuta dos operaciones:

1) Calcula la función de señal que aparece a la salida de la neurona, que se expresa como una función no lineal continua de las señales de entrada y sus intensidades sinápticas asociadas.
2) Calcula una estimación instantánea del *vector gradiente* formado por los gradientes de la superficie de error con respecto de los pesos o intensidades sinápticas conectadas a las entradas de la neurona. Este vector es necesario para la fase de retropropagación del error.

El algoritmo se obtiene mediante un procedimiento algo complicado, la siguiente notación facilitará la exposición:

i, j, k: Índices que refieren diferentes neuronas en la red, con la señal difundiéndose desde la entrada a la salida. Cuando la neurona *j* es una unidad de proceso en una capa oculta, *j* yace más próxima a la salida que la neurona *i*, y la neurona *k* más próxima a la salida que la neurona *j*.

n: Número de iteración, denota el n-ésimo patrón de entrenamiento presentado a la red.

E(n): Símbolo que se refiere a la suma instantánea de los errores al cuadrado durante la iteración *n*.

E_{prom}: Promedio del error cuadrado, es el promedio de los valores *E(n)* para todos los valores *n*, es decir para todos los patrones de entrenamiento.

$e_j(n)$:se refiere al error o señal de error en la salida de la neurona *j* durante la iteración *n*.

$d_j(n)$:se refiere a la respuesta deseada para la neurona *j* y se usa para calcular $e_j(n)$.

$y_j(n)$: se refiere a la función de señal que aparece en la salida de la neurona *j* en la iteración *n*.

$w_{ji}(n)$: se refiere al peso o intensidad sináptica que conecta la salida de la neurona i a la entrada de la neurona *j* en la iteración *n*.

$\Delta w_{ji}(n)$: denota la corrección aplicada a $w_{ji}(n)$.

$v_j(n)$: denota el nivel interno de actividad de la red de la neurona *j* en la iteración *n*. Constituye la señal aplicada a la no linealidad asociada a dicha neurona.

$\varphi_j(\bullet)$: es la función de activación que describe la relación no lineal entrada-salida de la neurona *j*.

θ_j : denota el umbral de la neurona *j*. Su efecto se representa por el peso sináptico $w_{j0} = \theta_j$ conectado a una entrada fija igual a -1, esta entrada tiene la función de fijar el umbral de activación.

$x_i(n)$: representa el i-ésimo elemento del vector o patrón de entrada.

$o_k(n)$: representa el k-ésimo elemento de del vector o patrón de salida.

η: denota el parámetro que determina el índice o tasa de aprendizaje.

Algoritmo de aprendizaje

El algoritmo se deriva de la siguiente manera: La señal de error en la salida de la neurona *j* durante la iteración *n* (es decir, durante la presentación a la red del patrón de entrenamiento *n*), se define:

$$e_j(n) = d_j(n) - y_j(n), \text{ la neurona } j \text{ es un nodo de salida.} \qquad (2.1)$$

Y el valor instantáneo del error cuadrado en la neurona *j* como $\frac{1}{2}e_j^2(n)$.

El valor instantáneo *E(n)* de la *suma instantánea de los errores cuadrados* se obtiene sumando los $\frac{1}{2}e_j^2(n)$ de todas la neuronas en la capa de salida. La *suma instantánea de los errores cuadrados* de la red se expresa:

$$E(n) = \frac{1}{2}\sum_{j \in C}\frac{1}{2}e_j^2(n) \qquad (2.2)$$

donde el conjunto C incluye todas las neuronas en la capa de salida . *N* denota el número total de patrones contenidos en el conjunto de patrones de entrenamiento (conjunto de entrenamiento). El *promedio del error cuadrado* se obtiene sumando *E(n)* para toda *n* y normalizándolo respecto a *N*., como se muestra:

$$E(n) = \frac{1}{N}\sum_{j \in C}^{N}\frac{1}{2}e_j^2(n) \qquad (2.3)$$

La suma instantánea de los errores cuadrados y las demás expresiones son función de los parámetros libres pesos sinápticos y umbrales de la red. Para un conjunto de patrones de entrenamiento, E_{prom} representa la *función de costo*, que representa la medida del desempeño de la red en el aprendizaje del conjunto de patrones de entrenamiento. El objetivo del proceso de aprendizaje es ajustar los parámetros libres a fin de minimizar E_{prom}. El método usado por este algoritmo se basa en un ajuste de los pesos sinápticos patrón por patrón. El ajuste de los pesos se hace de acuerdo con los respectivos errores calculados para cada patrón presentado a la red. El promedio aritmético de estos cambios individuales en los pesos sobre el conjunto de patrones es, por tanto, una estimación del verdadero cambio que resultaría de la modificación de los pesos basándose en la minimización de la función de costo, sobre todo el conjunto de entrenamiento.

En la siguiente figura se muestra la neurona *j* siendo alimentada por un conjunto de señales de función producidas por la capa previa. El nivel de actividad interno de la red $v_j(n)$ producido en la entrada de la función de activación de la neurona *j* es, por tanto:

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n) \tag{2.4}$$

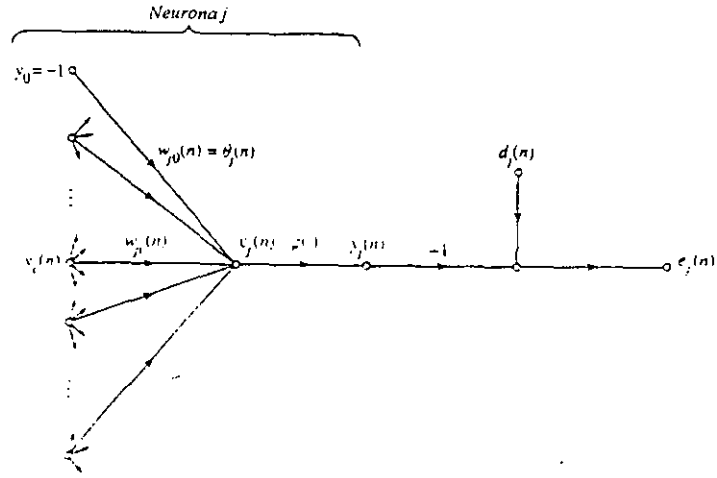

FIGURA 2.7 Gráfica del flujo de señales en la neurona *j*.

donde *p* es el número total de entradas (excluyendo los umbrales) aplicadas a la neurona *j*.

El peso sináptico w_{j0} (correspondiente a al entrada fija y_0=-1) es igual al umbral θ_j aplicada a la neurona *j* . Por tanto la señal de función $y_j(n)$ a al salida de la neurona *j* en la iteración *n* es:

$$y_j(n) = \varphi\big(v_j(n)\big) \tag{2.5}$$

El algoritmo de retropropagación aplica un ajuste $\Delta w_{ji}(n)$ al peso sináptico $w_{ji}(n)$, proporcional al gradiente instantáneo $\frac{\partial E(n)}{\partial w_{ji}(n)}$. De acuerdo a la regla dela cadena el gradiente puede expresarse como:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{2.6}$$

el gradiente $\frac{\partial E(n)}{\partial w_{ji}(n)}$ representa un *factor de sensibilidad*, que determina la dirección de

la búsqueda en el espacio de pesos sinápticos $w_{ji}(n)$.

Diferenciando la ecuación 2.2 respecto a $e_j(n)$ tenemos:

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \tag{2.7}$$

Diferenciando la ecuación 2.1 respecto a $y_j(n)$ tenemos:

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{2.8}$$

Diferenciando la ecuación 2.5 respecto a $v_j(n)$ tenemos:

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi'_j\left(v_j(n)\right) \tag{2.9}$$

(el signo ' indica diferenciación con respecto al argumento).

Diferenciando la ecuación 2.4 respecto a $w_{ji}(n)$ tenemos:

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{2.10}$$

Sustituyendo las ecuaciones 2.7 a 2.10 en 2.6 tenemos:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi'_j\left(v_j(n)\right)y_i(n) \tag{2.11}$$

La *regla delta* define el ajuste como :

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{2.12}$$

Donde η es una constante que define la tasa de aprendizaje y se denomina *parámetro de tasa de aprendizaje* (*learning-rate parameter*) del algoritmo de retropropagación. El uso del signo menos en 2.12 asegura el *descenso del gradiente* o bien el *gradiente descendente* en el espacio de pesos. Usando 2.11 en 2.12 tenemos:

$$\Delta w_{ji}(n) = -\eta \delta_j(n) y_i(n) \tag{2.13}$$

donde el *gradiente local* δ_j se define:

$$\delta_j(n) = -\frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -e_j(n)\varphi'_j\left(v_j(n)\right) \tag{2.14}$$

De la ecuaciones previas, es evidente que el error a la salida de cada neurona es el factor clave para determinar el ajuste de los pesos sinápticos. Sin embargo, dependiendo del lugar en la red que ocupe la neurona *j*, pueden identificarse dos casos distintos. Un caso es en el que la que la neurona *j* se encuentra en la capa de salida de la red. El otro caso es cuando dicha neurona esta en una capa oculta, aunque dichas neuronas no son directamente accesibles, comparten cierta responsabilidad en la generación del error de salida, la dificultad para determinar como serán ajustados los

parámetro de dichas neuronas provocó desconfianza y, luego, un letargo en la investigación de las RNA, el algoritmo de retropropagación resuelve este problema y es responsable del renovado interés en la investigación en este campo. En seguida se examina como se resuelve el problema para los dos casos.

Primer caso

Cuando la neurona *j* se localiza en la capa de salida de la red, se tiene al alcance su propia respuesta. Puede usarse 2.1 para determinar el error $e_j(n)$ asociado a la neurona *n*. Una vez determinado éste , el gradiente se calcula el gradiente local $\delta_j(n)$ con 2.14.

Segundo caso

Si la neurona *j* se encuentra en una capa oculta, no disponemos de la respuesta especifica que debe manifestar para minimizar el error. Así pues, la señal de error para dicha neurona será determinada de manera recursiva en términos de las señales de error de todas las neuronas a las cuales dicha neurona esta conectada directamente. La figura 2.8 muestra una neurona en una capa oculta de la red.

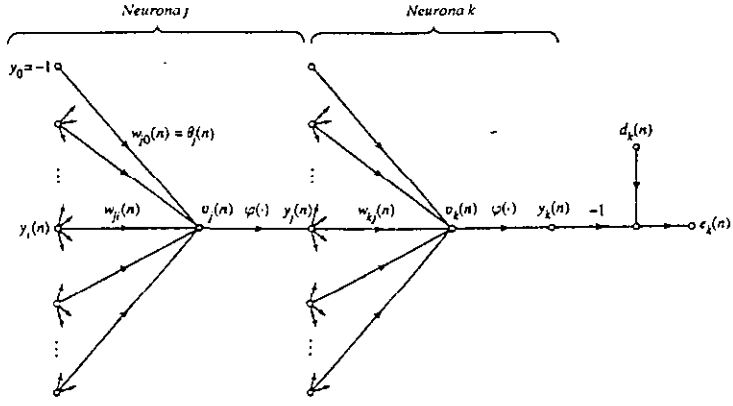

FIGURA 2.8 Gráfica del flujo de señales de la neurona *j* a la neurona *k*.

Según 2.14, el gradiente local se redefine, usando 2.9, para la neurona oculta *j*, como:

$$\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial y_j(n)}\varphi'_j\left(v_j(n)\right) \qquad (2.15)$$

Para calcular $-\frac{\partial E(n)}{\partial y_j(n)}\varphi'_j\left(v_j(n)\right)$ se procede así: De la figura 2.8 se observa que:

$$E(n) = \frac{1}{2}\sum_{j\in C}\frac{1}{2}e_k^2(n) \quad (k \text{ representa un nodo de salida}) \tag{2.16}$$

que es 2.2, salvo por el uso de k en lugar de j, para evitar confusiones con la neurona oculta j. Diferenciando la ecuación 2.16 respecto a $y_j(n)$ tenemos:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \tag{2.17}$$

Usando la regla de la cadena para obtener la derivada parcial $\frac{\partial e_k(n)}{\partial y_j(n)}$, y reescribiendo 2.17 en forma equivalente:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{2.18}$$

Y de la figura (2)(146) se advierte que:

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi'_k\left(v_k(n)\right) \tag{2.19}$$

por tanto:

$$\frac{\partial v_k(n)}{\partial y_j(n)} = -\varphi'_k\left(v_k(n)\right) \tag{2.20}$$

Se nota también que para la neurona k, el nivel interno de actividad de la red es:

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n) \tag{2.21}$$

donde q es el número total de entradas (excluyendo los umbrales) aplicadas a la neurona k. Diferenciando la ecuación 2.21 respecto a $y_j(n)$ tenemos:

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{2.22}$$

Usando 2.20 y 2.22 en 2.18 tenemos:

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k e_k(n)\varphi'_k\left(v_k(n)\right) w_{kj}(n) = -\sum_k \delta_k(n) w_{kj}(n) \tag{2.23}$$

Usando 2.23 en 2.15 obtenemos el gradiente local para la neurona oculta j:

$$\delta_j(n) = \varphi'_j\left(v_j(n)\right)\sum_k \delta_k(n) w_{kj}(n) \quad (j \text{ es una neurona oculta}) \tag{2.24}$$

Tenemos, de los resultados anteriores, que el algoritmo de retropropagación en el perceptrón multicapa da lugar a las siguientes relaciones:

Primero: la señal de corrección $\Delta w_{ji}(n)$ aplicada al peso sináptico que conecta la neurona i a la neurona j, se define:

(corrección del peso)=(taza de aprendizaje)(gradiente local)(señal de entrada de la neurona j), es decir:

$(\Delta w_{ji}(n))=(\eta)(\delta_j(n))(y_i(n))$

Segundo: el gradiente local depende de si la neurona j es un nodo de salida o uno oculto, hay entonces dos casos

1. Si la neurona j es un nodo de salida, el gradiente local es igual al producto de la derivada $\varphi'_j\left(v_j(n)\right)$ y la señal de error $e_j(n)$.

2. Si j es una neurona oculta, el gradiente local es igual al producto de la derivada asociada $\varphi'_j\left(v_j(n)\right)$ y la suma ponderada de los gradientes locales calculados para las neuronas en la oculta siguiente que esta conectada a la neurona j.

Las tres fases del algoritmo

Come se mencionó anteriormente, el algoritmo consta de tres fases. Durante la primera fase se denomina propagación hacia adelante, la segunda propagación hacia atrás o retropropagación y la tercera consiste en el ajuste de los pesos o intensidades sinápticas.

En la primera fase, los pesos sinápticos permanecen inalterados. Las señales de función se calculan neurona por neurona, la señal de función que aparece e la salida de la neurona *j* se calcula usando:

$$y_j(n) = \varphi\left(v_i(n)\right) \tag{2.26}$$

$v_j(n)$ es el nivel interno de actividad de la red de la neurona *j* definido:

$$v_i(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n) \tag{2.27}$$

p es el número total de entradas aplicadas a la neurona *j*, $w_{ji}(n)$ es el peso sináptico de la conexión de la neurona *i* a la neurona *j* y $y_i(n)$ es la señal de función que aparece a la salida de la neurona *i* y a la entrada de la neurona *j*. Si ésta última esta en la primera capa oculta de la red, entonces:

$$y_i(n) = x_i(n) \tag{2.28}$$

Si, por otra parte, la neurona *j* se encuentra en la capa de salida de la red:

$$y_j(n) = o_j(n) \tag{2.29}$$

Esta salida se compara con la respuesta deseada $d_j(n)$, y obteniendo la señal de error $e_j(n)$ para dicha neurona.

La segunda fase o fase de retropropagación del error, comienza en la capa de salida, propagando las señales de error hacia las capas detrás de esta, calculando recursivamente el gradiente local para cada neurona.

La tercera fase, comienza ya con todos los gradientes calculados, lo cual permite aplicar los ajustes a los pesos sinápticos de acuerdo con la ecuación 2.25. Para una neurona localizada en la capa de salida, el gradiente local es simplemente igual a la señal

de error multiplicada por la primera derivada de su función de activación. Dados los gradientes locales de la capa de salida, se usa a continuación 2.24 para calcular los gradientes de la penúltima capa y por tanto los ajustes a los pesos de todas las conexiones que la alimentan. El cálculo recursivo continúa capa por capa propagando los ajustes sinápticos que se hayan hecho.

Los patrones o vectores de entrenamiento se mantienen fijos durante la totalidad de las tres fases del proceso de entrenamiento. El proceso se repite usando cada vez diversos patrones de los disponibles en el conjunto de patrones de entrenamiento o conjunto de entrenamiento.

El algoritmo se detiene cuando se ha alcanzo un nivel de error que se considere adecuado para la aplicación a que se destina; de otro modo el algoritmo prosigue.

Así pues tenemos las siguientes características de este modelo que puede aplicarse como controlador:

Arquitectura : perceptrón con capas múltiples y alimentación de la señal hacia adelante.

Algoritmo: retropropagación del error, aquí existen señales que van hacia adelante y hacia atrás, y hay un modo de determinarlas.

Funciones de activación: diferenciables, habitualmente funciones sigmoides, y las misma para todas las capas.

Este es uno de los varios modelos disponibles, es muy apropiado para el aprendizaje supervisado, pero puede adaptarse a otros tipos.

NEUROCONTROL

Las RNA exhiben propiedades que son deseables y las hacen viables como sistemas de control Inteligente, se han mencionado ya, pero para el caso de control las mas interesantes son:

Aprenden por experiencia, en vez de modelar o ser programadas.

Pueden generalizar, es decir, mapean entradas similares entre sí a salidas similares entre sí.

Pueden formar mapeos arbitrarios continuos y no lineares.

Su arquitectura es distribuida, inherentemente paralela y puede, por ello, cumplir sus tareas en tiempo real

Para que una red pueda ser implementada como sistema de control o neurocontrolador, requiere de estas propiedades adicionales:

Estabilidad en el tiempo, la habilidad de absorber nueva información (plasticidad) mientras retiene el conocimiento previamente codificado a través de la red (estabilidad).

Adaptación en tiempo real o aprendizaje en respuesta a las variaciones de la

planta.

Condiciones conocidas o probadas de la convergencia del aprendizaje necesarias para procesar la predicción de su comportamiento en lazo cerrado (para certificar el neurocontrolador).

Para el neurocontrol, las redes que pueden ejecutar aproximaciones de funciones son las más útiles Si las variables a ser estimadas por la red son valores futuros (representando, por ejemplo, el proceso como una serie no lineal en el tiempo), entonces la red es un predictor. Si, por el contrario las variables de salida estimadas se relacionan directamente con las variables de entrada, entonces la red se usa para modelado de la planta. Así mismo si un modelo de la planta ha sido derivado ya, entonces un controlador puede ser sintetizado a partir de muestras representativas de la función de control.

La red neuronal artificial mas popular en aplicaciones del neurocontrol es el Perceptron Multi-Capa (PMC), usando, para su entrenamiento, el algoritmo de retropropagación.

Este tipo de redes presentan, desafortunadamente, algunos inconvenientes; son relativamente lentas, la convergencia no puede establecerse de modo general y, por último, si el espacio de estados cubierto por una red se incrementa, la totalidad de la red habrá de ser re-entrenada. Un PMC utiliza una versión simple del gradiente descendiente usado en identificación y control clásicos. Los modelos no lineares utilizan habitualmente métodos de optimización no lineales, tales como la optimización por gradiente descendiente, ya que aseguran convergencia hacia mínimos locales que contienen el vector inicial en el fondo de su cuenca de atracción. El perceptrón multicapa ajusta sus pesos o intensidades sinápticas de manera altamente no lineal , generando un compleja función de costo en su espacio pesos, con muchos mínimos locales donde las reglas de gradiente descendente puede quedar atrapado.

La poca rapidez de convergencia de los PMC ha sido superada en parte usando diversas técnicas, las cuales son especificas y no se discuten aquí con amplitud[1].

Las memorias asociativas implementadas por redes neuronales forman primero una transformación no lineal del espacio de entradas, por lo general a un espacio de muchas mas dimensiones, seguido de una capa lineal en los pesos ajustables para formar una salida que es una combinación ponderada de la entrada transformada. La primera capa se fija por el diseñador y un conocimiento previo acerca de la planta puede ser incorporado (bajo la forma de una estructura sinaptica), mientras que la capa lineal de salida contiene los pesos que son ajustables por alguno de los varios métodos de realimentación del error de salida; lo que resulta en una simple y convexa función de costo en el espacio error-pesos, para el cual existe generalmente un mínimo global único, a diferencia del PMC, donde los minimos locales son múltiples.

---

[1]Entre estas son dignas de mención: La *red de capa de vínculos funcionales* (Functional Layer Link Network), que utiliza un conocimiento a priori de las relaciones entrada-salida para ejecutar un proceso previo a la entrada de la información en la red. Otro enfoque ha sido el de usar memorias asociativas, tales como la de capa simple usada en los modelos de *controlador de articulación de modelo cerebelar* (Albus Cerebellar Model Articulation Controller, CMAC) , redes de *funciones de base radial* (Radial Basis Functions) y la red *Brown B-Spline*. Todas satisfacen, en alguna medida, las condiciones funcionales para un neurocontrolador citadas arriba.

Clasificación de los Neurocontroladores

Además de las clasificaciones en que se dividen regularmente las redes neuronales (arquitectura, algoritmo de aprendizaje y función de activación), en el contexto de los neurocontroladores se pueden clasificar de otras maneras.

Por su modo de Generalización

Una forma de clasificarlas es dividiendo entre redes neuronales que generalizan localmente o globalmente. La generalización de la red es local si uno o mas de los pesos de la red pueden afectar las salida de la red para todo punto de el espacio de entradas. El perceptrón multicapa es un ejemplo de redes de generalización global, esta propiedad le confiere capacidad de generalización y lo hace un controlador robusto, aunque a expensas de un aprendizaje lento y de interferencias en el aprendizaje que se amplían por toda la red. Mientras que la generalización local ocurre en redes para las cuales unos pocos pesos afectan la respuesta de salida de la red para un punto dentro una región local del espacio de entrada; las memorias asociativas tales como CMAC , las de funciones de base radial y la red *Brown B-Spline* son ejemplos de redes de generalización local, en las cuales la interferencia en el aprendizaje se minimiza y , consecuentemente, el aprendizaje es relativamente rápido, ello debió al pequeño número de pesos a ser actualizados.

Además de esta clasificación, Werbos (1993) clasifica a los neurocontroladores según su diseño en cinco tipos principales: Control Supervisado, Control Inverso, Control Neuronal Adaptativo, Retropropagación de la Utilidad en el Tiempo y lo que denomina *Adaptive Critics*.

Algunos autores entusiastas han señalado al las redes neuronales como las soluciones universales a cualquier problema de control. Pensando en ellas como especies de cajas milagrosas capaces de resolver cualquier problema con tan solo adaptar su tamaño y reducir su tiempo de entrenamiento (mejoras que son hoy en día la tendencia de los equipos de computo seriales). Paul Werbos (1992), no obstante ha señalado que los diseños básicos de RNA con que se cuenta hoy en día pueden clasificarse en alguna de cinco categorías de neurocontroladores. Afirma también que todas estas pueden ser entendidas dentro del contexto de la teoría de control existente. con una sola excepción, se trata de métodos genéricos , cuya principal ventaja es su efectividad computacional; que pueden ser usados en cualquier red neuronal de funciones de activación diferenciable. Cuando se cuente con el tiempo suficiente y un buen conocimiento para especificar las funciones en detalle, se puede recurrir a los métodos tradicionales, aunque, en general el trabajo será mas costoso. Cuando no hay mucho tiempo o se tiene un conocimiento insuficiente de las funciones, se pueden explotar los teoremas de los aproximadores universales, que explican la capacidad de mapeo de las redes neuronales.

La aplicaciones practicas incluyen, por lo general, sistemas cuya dinámica es solo parcialmente conocida, así que los óptimo en estos caso es lograr una acercamiento a la soluciones que implique un compromiso entre las teoría de control clásicas, y las aproximaciones conexionistas, o demás técnicas disponibles (como sistemas difusos).

Consideraciones: Si una tarea puede realizarse con la misma eficacia usando métodos convencionales o redes neuronales, existen ventajas al utilizar las últimas, como la plasticidad y la robustez del control resultante.

Los cinco método específicos

Las aplicaciones específicas a que se destina una determinada RNA determina su arquitectura y demás características. Cada diseñador trata de adaptar la red de modo que sea especialmente eficiente para su tarea particular, esto hace que cada aplicación desarrolle redes muy elaboradas. Detrás de esta aparente complejidad, existen solamente cinco diseños genéricos usados para construir redes neuronales artificiales, dedicadas directamente al control de actuadores o dispositivos activos de salida de algún tipo. Existen diseños abocados al reconocimiento de patrones y al identificación de sistemas, que forman parte orgánica de los sistemas de control, pero no son necesariamente considerados como parte de sistema que directamente controla, sino como la parte sensorial, por ello no se discuten en este trabajo.

En el *Control No Supervisado*, una red neuronal aprende el mapeo de las entradas sensoras a las acciones deseada, adaptándose a un conjunto de entrenamiento que contiene ejemplos de lo que al red neuronal debe hacer. Esta puede imitar a un experto humano, o puede aprender operaciones o actividades básicas y generalizarlas para aplicarlas a situaciones mas exigentes. Su entrenamiento permite a los sistemas responder a las entradas sensoriales y se aplican ahora ampliamente en el desarrollo de robots manipuladores.

En el *Control Inverso Directo*, una RNA aprende la dinámica inversa de un sistema, de modo que puede hacer que el sistema siga una trayectoria deseada. Esta técnica se usa en el diseño de controladores para robots navegadores.

En el *Control Neuronal Adaptativo*, los mapeos lineales usados en diseños estándar como el *Control Adaptativo Referido A Modelo*, son remplazados por redes neuronales, teniendo como resultado un sistema más robusto y mayor habilidad para manejar no-linealidades.

La *Retropropagación de la Utilidad*, involucra un contraflujo de información a través del tiempo. Adapta un controlador óptimo , resolviendo , esencialmente, un problema de cálculo de variaciones. Habitualmente, se resuelve para los pesos o parámetros óptimos del controlador, en vez de para un catálogo de acciones. Pero las investigaciones en redes neuronales han conducido a hacerlo de ambos modos; siendo el último el mas exacto y el primero más robusto en la mayoría de las aplicaciones. Para resolver problemas como los del tipo de seguimiento de trayectorias, este diseño se ha comportado bien en situaciones donde el control por dinámica inversa es inaplicable. como con el cálculo de variaciones, este método requiere de un modelo del sistema a controlar; que puede, así mismo, ser una red neuronal, y permite muy poco ruido.

Control Supervisado

El aprendizaje supervisado, no involucra un plan de largo plazo. Algunos investigadores arguyen que el control supervisado es algo trivial, ya que solo trabaja si se conoce de antemano como ejecutar la tarea deseada. No obstante, otros , argumentan que este tipo de sistemas pueden ser en verdad de mucha utilidad. Los consideran como herramientas para transferir información de un experto humano a un sistema artificial. Este último se interpretaría como una suerte de "experto neuronal"; el cual, a fin de aprender sus tareas, enfoca su atención en lo que los expertos hacen en realidad, en lugar de lo que ellos declaran que hacen, su objetivo es, basándose en experiencias

reales y conocidas, alcanzar una imitación aceptable, sin recurrir a soluciones simbólicas explícitas, que, en muchos casos, hacen abstracción de fenómenos subsidiarios como el ruido, la incertidumbre, etc (los sistemas difusos discutidos en el capítulo 1 son una variedad de soluciones simbólicas, pero que incorporan la incetidumbre). Además, si el operador humano solo puede ejecutar la tarea lentamente, en la simulación el controlador neuronal puede aprender a imitarlo y luego ejecutar aquella más rápidamente.

Para implementar el control supervisado , se comienza por escoger uno de los varios métodos disponibles en la literatura de RNA (memorias asociativas, perceptron etc.). En todos estos, el usuario vincula de modo adecuado una historia de entradas y objetivos deseados, los cuales suministra durante la fase de aprendizaje y el algoritmo de aprendizaje adapta, entonces, la red neuronal (algunos algoritmos permite que esto se ejecute en tiempo real. (entre otros métodos de aprendizaje se pueden nombrar CMAC, retropopagación o las variedades de memorias asociativas).

Los sistemas de control supervisado pueden ser vistos como los apéndices mecánicos, a los cuales se les suministraba las mismas entradas que un operario disponía a un controlador de robots, y este último las registraba en una localidad destinada a ello. Sin embargo, la diferencia estriba en que, estos graban solo una registro fijo de movimientos, mientras que las redes neuronales almacenan las relaciones que existen entre las entradas y las salidas, y pueden en general adaptarse mejor.)

Control Inverso (*Direct Inverse Control*)

Este tipo de control se basa también en el aprendizaje supervisado. De nuevo, cualquier método de aprendizaje puede ser usado. Los objetivos son las señales de los actuadores (ángulos), y las entradas son las coordenadas del brazo del robot (o algún otro sistema). la historia de las entradas y los objetivos son el resultado de dejar que el sistema deambule por su entorno, memorizando o aprendiendo las posiciones y ángulos reales.

Qué método de aprendizaje se elegirá; y la forma en que se aplica al problema específico varía mucho para cada problema y cada investigador, los procesos a que debe someterse la información antes de ingresarla al controlador son, también, diversos. Proponer un ejemplo particular sería ilustrativo; pero se correría el riesgo de restringir una teoría naturalmente general y abierta, la investigación en RNA es algo vivo y cambiante y en donde se debe buscar soluciones a veces osadas.

Además, en general para cada aplicación particular han de tenerse las bases suficientes para determinar la arquitectura y el algoritmo más conveniente al problema. Así, si se pretende modelar un sistema nervioso biológico, habrá que sustentarse en un conocimiento suficiente sobre neurofisiología, si se intenta reconocer patrones, comprimir datos o controlar un actuador mecánico, habrá que enfocarse sobre las características del sistema, más bien que algún problema particular.

Kawato y Jordan [17] hacen énfasis en las limitaciones del control inverso directo, en los casos en que no existe un mapeo de uno a uno de las coordenadas espaciales a las señales del actuador, por ejemplo cuando hay más que coordenadas, tal mapeo uno a uno no existe.

Control Neuronal Adaptativo (*Neural Adaptive Control*)

El área de investigación acerca de estos sistemas es muy reciente y es una con grandes expectativas, se relaciona con áreas como el control de ecosistemas, controles y

reguladores biológicos etc, y esta en relación con los diseños de control adaptativo (como los sistemas adaptativos de control difuso). que proveen de nuevas líneas de investigación e ideas.

Retropropagación de la Utilidad en el Tiempo (*Backpropagation Through Time: BBT*)

Este diseño puede ser usado para resolver problemas de optimización en el tiempo porque:

Permite al usuario o diseñador tomar cualquier función de utilidad, medida de desempeño, o función de costo para maximizar o minimiza.

El método da cuenta precisamente del impacto de las acciones presentes sobre la futura utilidad.

BBT es básicamente cálculo de variaciones. La diferencia esencial es que BBT incluye un modo de calcular las derivadas de la utilidad, que son más fáciles de calcular cuando se trabaja con grandes sistemas dispersos (*sparses systems*). Tal como el cálculo de variaciones, este método manifiesta dos desventajas

Requiere un modelo de su ámbito externo , que, en principio, deberá ser exacto y estar libre de ruido.

Requiere cálculos usando estados previos en el tiempo (*backwards truogh time*) en el cálculo de las derivadas, lo que no es consistente con un verdadero aprendizaje en tiempo real.

Aún , se puede imitar algo similar al aprendizaje en tiempo real; dividiendo la experiencia en diversos experimentos o cadenas y actualizando los pesos después de que cada experimento sea analizado.

Adaptive Critics

Los diseños *Adaptive Critic* incluyen una amplia familia de métodos. Como BBT, ellos son capaces de maximizar cualquier función o medida de reforzamiento sobre el tiempo. Son menos exactos que *BBT*, ya que utilizan una aproximación (la red crítica), para representar el efecto de las acciones corrientes en la utilidad futura. Por otra parte, pueden ser derivados como aproximaciones de la programación dinámica , que es un método exacto para tratar ruido y con modelos ambientales estocásticos.

Este es la categoria donde existe mayor libertad y divergencia entre un enfoque y otro, incluye muchos métodos y algunos se han probado experimentalmente, con resultados aceptables. El que un método o no pueda ser clasificado como perteneciente o no a esta categoria depende de como sea interpretado. Como se ha dicho es el área que más estrechamente se relaciona con el modelado de sistemas de control biológicos.

CAPITULO III: SISTEMAS *NEURO-DIFUSOS*

INTRODUCCION: ANTECEDENTES E HISTORIA

LOS SISTEMAS DE INFERENCIA DIFUSOS COMO APROXIMADORES UNIVERSALES

En el capitulo segundo de este trabajo se ha expuesto que una RNA del tipo Perceptrón Multicapa, puede realizar un mapeo con una exactitud arbitraria. Existen varios modelos que poseen la propiedad de aproximadores universales, las Máquinas de Turing[1] son un caso especial, y algunos autores argumentan que las RNA pueden reducirse a una máquina de Turing. Los sistemas de inferencia difusos son también aproximadores universales. Esta afirmación es importante porque implica que ambos modelos pueden ser utilizados para tareas de propósito general intercambiándose, cooperando o sintetizando nuevos modelos que incorporen, por un lado las propiedades de los SCD; que le permiten usar descripciones y reglas lingüísticas que pueden luego ser tratados a nivel simbólico, y por otro, las cualidades de adaptación y aprendizaje por medio de ejemplos presentes en las redes neuronales.

Bart Kosko ha comparado los sistemas difusos adaptivos y las redes neuronales con los sistemas clásicos de control [20]. Él enfatiza la similitud de ambos sistemas con la habilidad de un operador humano, quien es capaz de control un proceso pobremente definido, valiéndose solo de la experiencia y sin conocimiento de la dinámica subyacente al sistema. Considera a estos sistemas de control como estimadores libres de modelos (*Modelo Libre* es la expresión mas común) o estimadores sin modelo (*model-free estimators*); es decir que estiman una función sin requerir de una descripción matemática de la dependencia funcional de la salida respecto de la entrada, pues aprenden a partir de ejemplos.

Una de las criticas más persistentes hacia los controladores basados en lógica difusa, es que su efectividad no ha sido comprobada categóricamente, Hoy, sin embargo, se cuentan con investigaciones que han concluido con algunas de estas pruebas; sin bien aún existen dudas por resolver. Los primeros intentos por justificar teóricamente la flexibilidad y amplitud de alcances de los controladores difusos eran solo descripciones cualitativas. Wang [22] fue el primero que presentó, en 1992, una prueba matemática de que ciertos tipos de controladores difusos poseen la capacidad de aproximar una función real y continua en un conjunto compacto con exactitud arbitraria, es decir son aproximadores universales. Buckley [23] presentó una prueba similar para una modificación de los controladores tipo Sugeno. Por último J. L. Castro [24], obtuvo una

[1] A grandes rasgos, una Máquina de Turing es un sistema formal que ejecuta secuencialmente operaciones finitas en tiempos discretos sobre las entradas del sistema. Adicionalmente se especifica que las operaciones son de tipo lógico y aritmetico y que se dispone de un tiempo infinito. Los modelos de computadoras secuenciales como el de J. Von Neumann, se basan en modelos de maquinas de Turing con recursos finitos.

prueba mas general, aplicable a una inmensa variedad de controladores.

La estrategia de la prueba de Castro, comienza como sigue:

Él afirma que son aproximadores universales los SCD que pertenecen a clases con:

Un solo tipo de funciones de membresía, que pueden ser de forma triangular , trapezoide, etc.

Un operador de conjunción difuso modelado por una función arbitraria del tipo T-norma.

La implicación difusa debe satisfacer una propiedad muy común en los SCD.

El método de defuzzificación debe satisfacer una condición también frecuente en los SCD.

Castro considera a la máquina de inferencias difusas como dos etapas independientes. Considera, por tanto, que un SCD se compone de estos cuatro elementos:

Base de Reglas Difusa

Interfaz de Fuzzificación (etapa de fuzzificación)

Maquina de Inferencias Difusa

Interfaz de Defuzzificación (etapa de defuzzificación)

Las propiedades de cada una de estas etapas corresponden a las explicadas en el capítulo uno, salvo por la separación explicita que propone dentro de la Maquina de inferencias.

Esta sección de la prueba de Castro puede resultar ineresante para este trabajo y se resume en el Apéndice II, mientras que el Apéndice I presenta una prueba para las redes neuronales. No obstante se omitirá una descripción completa aquí. El trabajo está publicado en: Castro, J. L. *Fuzzy Logic Controllers Are Universal Approximators.* IEEE Transactions on Systems, Man, and Cybernetics. Vol 25. No 4. Abril 1995. pp 629-635.

Este resultado es una de las primeras pruebas generales acerca de la capacidad de aproximación de los SCD. Junto con la prueba análoga para las redes neuronales, muestra que dichos modelos comparten propiedades básicas, y que la convergencia hacia un modelo mas general es posible.

REDES ADAPTIVAS

Las tentativas de mejorar los sistemas libres de modelo, según Kosko, y los resultados teóricos como el descrito en la sección anterior han conducido a postular modelos más generales basados en los disponibles hasta ahora. Las redes adaptivas son el conjunto que contiene todos los tipos de modelos de redes neuronales con capacidad de aprendizaje supervisado. Entre este tipo de redes existen algunas que pueden ser interpretadas como un conjunto organizado de reglas difusas 'Si-Entonces', este tipo de redes son en las que se enfocara la siguiente sección.

Arquitectura

Un *Red Adaptiva* es una estructura en forma de red cuyo comportamiento global entrada-salida esta determinada por los valores de una colección de parámetros modificables. Especificamente, la configuración de una red adaptiva se compone de un conjunto de nodos conectados por medio de enlaces con direcciones determinadas, donde cada nodo es una unidad de proceso que ejecuta una función estática sobre las señales que fluyen hacia dentro, para generar una salida única en el nodo. Cada enlace especifica la dirección del flujo de las señal de un nodo a otro. Usualmente, la función del nodo es una función parametrizada cuyos parámetros son variables; cambiando estos parámetros se cambian tanto la función de nodo como el comportamiento general de la red adaptiva.

En el caso general, una red adaptiva es heterogénea; cada nodo puede tener una función diferente, además cada enlace tiene una función únicamente direccional, solo definen la dirección de propagación de la señal; no existen los enlaces ponderados ni se asocian parámetros con cada enlace. La figura siguiente muestra un red adaptiva típica con dos entradas y dos salidas

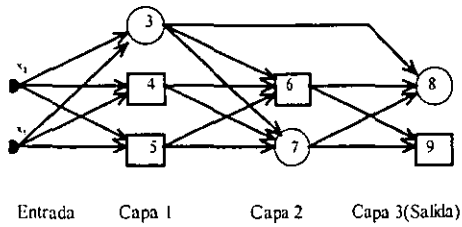

FIGURA 3.1 Representación de una red adaptiva.

Los parámetros de una red adaptiva están distribuidas en los nodos de la red, así que cada nodo tiene un conjunto de parámetros locales. La unión de estos conjuntos de parámetros locales es el conjunto de parámetros globales de la red. Si el conjunto de parámetro de un nodo es no vacío , entonces su función depende de estos parámetros; el cuadrado representa este tipo de *nodo adaptivo*. Si un nodo tiene un conjunto de parámetros vacío, entonces su función es fija, el circulo en la figura representa tales nodos de función fija.

Las redes adaptivas se clasifican en dos categorías, dependiendo de los tipos de conexión que poseen: *alimentadas hacia adelante (feedforward)* y *recurrentes*. Las redes alimentadas hacia adelante propagan la salida de cada nodo desde el lado de entrada hasta el lado de salida de la red, únicamente. Las redes recurrentes poseen al menos una conexión que forma un camino circular dentro de la red , la figura 3.2 muestra una red recurrente, mientras que la figura 3.1 muestra una de alimentación hacia adelante.

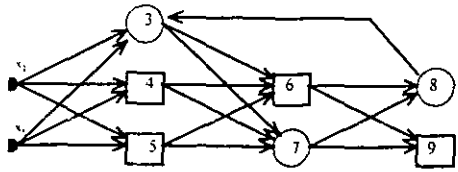

FIGURA 3.2 Una red adaptativa recurrente.

A nivel conceptual, una red adaptiva alimentada hacia adelante es un mapeo estático entre su espacio de entrada y su espacio de salida, mapeo que puede ser una función lineal simple o una no lineal muy complicada, dependiendo de la estructura de la red.

El objetivo al construir redes adaptivas es lograr un mapeo no lineal que no es regido por un modelo conocido, sino por un conjunto de datos consistente de cierto número de pares formados por estados de entrada y de salida que se desean del sistema. Como en las redes neuronales, esta información se denomina *conjunto de entrenamiento* o *conjunto de datos de entrenamiento*. El procedimiento para ajustar los parámetros de la red adaptiva de modo que su desempeño mejore respecto de los objetivos deseados se denomina, también como en las redes neuronales, *algoritmo de aprendizaje*. Usualmente el desempeño del sistema se mide como la discrepancia entre la salida deseada y la salida verdadera de la red para una y la misma entrada. Por lo general, el algoritmo de aprendizaje deriva de la aplicación de una técnica específica de optimización para una medida de error, y dependerá tanto de la estructura de la red como de la representación y medida del error.

Las redes neuronales son un ejemplo de redes adaptivas donde las entradas, salidas, errores y medidas son de tipo numérico, sean reales o racionales, escalares o vectoriales. El algoritmo de aprendizaje se adapta a esta representación, usando herramientas matemáticas como el cálculo y el álgebra para implementar la técnica de minimización del error que mejor convenga.

Sin embargo, si la representación de las entradas, salidas o señales intermedias de la red cambia, será necesario adaptar nuevos algoritmos para el tipo de representación. Se ha visto que la estructura de un SCD es muy similar a la de un sistema experto o a los árboles de decisión usados en controles digitales y computadoras. Las manipulaciones que se llevan a cabo dentro de un SCD son simbólicas, similares a las de la lógica clásica, pero implementadas por medio de los operadores y relaciones definidos en la lógica difusa. Se ha dicho también que existen en el SCD dos tipos de operaciones, las propiamente simbólicas o lógicas; definidas en la base de reglas e implementadas en la máquina de inferencias y aquellas de representación de información que Castro llama apropiadamente interfaces, que son las etapas de fuzzificación y defuzzificación.

De estas puntualizaciones es claro que las redes adaptativas ni tiene porque

restringirse a las representaciones reales o vectoriales de la información. Del mismo modo los sistemas expertos de proceso simbólico fueron modificados introduciendo cambios en la representación y el proceso de información mediante conjuntos y lógica difusa, las redes adaptivas incluyen modelos que son la aplicación de representaciones u operadores difusos a los modelos de las redes neuronales. La siguiente sección tratará de uno de los modelos donde se ha logrado la integración los procesos difusos a las redes neuronales.

EL ALGORITMO HÍBRIDO DE APRENDIZAJE

Dentro del contexto de las redes adaptivas existen algoritmos que puede ser aplicados a redes cuya representación es, hasta cierto grado, irrelevante, y puede usarse tanto en modelos convencionales de redes neuronales, como en sistemas con representaciones difusas. Se ha observado que si la salida de una red adaptiva o su transformación es lineal en algunos de los parámetros, entonces pueden identificarse estos parámetros lineales por el método de optimización lineal de mínimo cuadrado ampliamente utilizado en problemas de optimización. Esta observación condujo al diseño de una algoritmo de aprendizaje híbrido, que será denominado aquí *Algoritmo Híbrido de Aprendizaje*. Este algoritmo combina el método de gradiente descendente (que es el método de optimización usado en el algoritmo de retropropagación descrito en el capitulo dos), con el estimador de mínimo cuadrado; LSE por sus denominación en inglés *Least Square Estimator*, para la identificación rápida de parámetros. Dentro del algoritmo existen los siguientes tipos de aprendizaje:

1) Aprendizaje por lotes

Aquí se hace uso de los resultados de toda una época de aprendizaje, es decir de los errores generados por todos los elementos del conjunto de entrenamiento. Por simplicidad se asume que la red adaptiva considerada tiene una sola salida, ya que al igual que los sistemas difusos, los sistemas de múltiples salidas pueden analizarse como una colección de sistemas MISO (multiples entradas, una salida). La salida única está dada por:

$$salida = F(\vec{I}, S) \qquad (3.1)$$

Donde $\vec{I}$ es el vector de variables de entrada y S es el conjunto de parámetros. Si existe una función H tal que la composición de funciones $H \circ F$ es lineal en algunos de los elementos de S, entonces estos elementos pueden ser identificados por el método de mínimos cuadrados. Formalmente, si el conjunto de parámetros S puede ser descompuesto en dos subconjuntos S_1 y S_2

$$S = S_1 \oplus S_2$$

(Donde $\oplus$ representa suma directa), tal que $H \circ F$ es lineal para los elementos de S_2, entonces, aplicando H a

$$salida = F(\vec{I}, S)$$

Si la matriz de pesos sinapticos de la red adaptiva en cuestión es A, y B es la matriz formada por los vectores de entrenamiento del conjunto de entrenamiento para

una época de aprendizaje, tenemos:

$$H(salida) = A \circ F(\vec{I}, S) \qquad (3.2)$$

el cual es lineal en los elementos de S_2. Ahora dando valores a los elementos de S_1, se pueden 'conectar' P datos de entrenamiento en la expresión anterior, obteniendo la ecuación matricial:

$$A\theta = B \qquad (3.3)$$

donde θ es un vector desconocido cuyos elementos son parámetros en S_2. Esta ecuación representa el problema lineal estándar de mínimo cuadrado y la mejor solución para θ, que minimiza $\|A\theta - B\|^2$, es el estimador de mínimo cuadrado (LSE) θ^*:

$$\theta^* = (A^T A)^{-1} A^T B \qquad (3.4)$$

donde A^T es la matriz traspuesta de A y $(A^T A)^{-1} A^T$ es la Pseudo-inversa de A si A^T es no singular. Ahora, específicamente, hágase a_i^T igual al i-esimo renglón de la matriz A, definido por $A\theta = B$. Hágase b_i^T igual al i-ésimo elemento de B. Entonces θ puede ser calculado iterativamente como sigue:

$$\left.\begin{aligned} \theta_{i+1} &= \theta_i + S_{i+1} a_{i+1} (b_{i+1}^T - a_{i+1}^T \theta_i) \\ S_{i+1} &= S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{1 + a_{i+1}^T S_i a_{i+1}}, \\ & i = 0,1,\ldots,P-1 \end{aligned}\right\} \qquad (3.5)$$

donde el estimador de mínimos cuadrados θ^* es igual a θ_p.

P es el número de patrones de entrada que conforman una época.

Las condiciones iniciales necesitadas para asegurar esta expresión son θ_0 y $S_0 = \gamma I$.

Donde γ es un número grande positivo, I es la matriz identidad de dimensión $M \times M$ (M es la dimensión del vector de parámetros óptimos que se obtenga). Cuando se trata de redes de salida múltiple aún se aplica la expresión (3.5) excepto en que b_i^T es la i-esima columna de B.

Ahora pueden combinarse el método del gradiente y el LSE para ajustar los parámetros en una red adaptiva. Para aplicar el método híbrido de aprendizaje por lotes, cada época se considera compuesta de una propagación hacia adelante y de una hacia atrás. En la propagación hacia adelante, luego de que un vector de entrada ha sido presentado, se calculan las salidas de nodo en la red, capa por capa, hasta que un renglón correspondiente en la matrices A y B en la expresión $A\theta = B$ sea obtenido. Este proceso se repite para todas las entradas de datos y para la totalidad de los componentes de A y B; entonces los parámetros en S_2 son identificados por la fórmula de la Pseudo-inversa en $\theta^* = (A^T A)^{-1} A^T B$ o por las fórmulas recursivas de la expresión (3.5). Luego que los parámetros en S_2 han sido identificados, puede calcularse el la medida de error para cada entrada de datos. En la parte de propagación hacia atrás, las señales de error

(calculado según el método de gradiente descendente del algoritmo de retropropagación) se propagan desde la parte de salida de la red hasta la parte de la entrada; el vector gradiente se acumula con cada entrada de datos . Al final de la propagación hacia atrás para todos los datos de entrenamiento, los parámetros en S_1 son ajustados según las expresiones del algoritmo de retropropagación (capitulo dos).

Para valores dados de los parámetros en S_1, los parámetros en S_2 así encontrados garantizan ser un punto optimo global (no solo un óptimo local) en el espacio de parámetros de S_2, debido a la elección de la medida de error cuadrada. Este algoritmo de aprendizaje reduce la dimensión del espacio de búsqueda en el método de gradiente descendente, pero además, generalmente reduce substancialmente el tiempo que el método consume para converger.

No debe perderse de vista que al usar el método de mínimo cuadrado en los datos sometidos a la transformación $H(\bullet)$, los parámetros obtenidos son óptimos en términos de la medida de error cuadrado transformado, en lugar de la forma original. En la práctica, esto no representa problemas en tanto que $H(\bullet)$ sea monotónicamente creciente y los datos de entrada no sean excesivamente ruidosos.

2) Aprendizaje por Patrón

Si los parámetros se ajustan inmediatamente después de la presentación de un vector individual, se tiene un esquema de aprendizaje por patrón o aprendizaje *en línea*. Esta estrategia de aprendizaje es sumamente útil cuando se usan sistemas con características cambiantes y donde es crucial la identificación de parámetros con cada cambio en los datos de entrada. Para llevar a cabo esta estrategia de aprendizaje es necesario contar con un método de gradiente descendente que, así mismo, se pueda ajustar con cada patrón o vector de entrada, este método existe, pero su descripción no ha sido incluida en este trabajo. Esta segunda forma del método no resulta relevante para la discusión de los sistemas neuro-difusos pretendida aquí, pero puede consultarse en Jang y Sun 1995 [25].

LA CLASE ANFIS *DE REDES ADAPTIVAS*

La clase de redes adaptivas que actúa como marco para sistemas de inferencias difusos, es denominada ANFIS por las iniciales de su denominación en inglés: *Adaptive Network-Based Fuzzy Iference System*; que son sistemas de inferencia adaptivos neuro-difusos. Se describirá aquí, análogamente al caso de los sistemas de control difuso, un sistema ANFIS típico.

A) Arquitectura

Asúmase que el sistema en consideración tiene dos entradas x y y; y una salida z. Un conjunto de reglas Si-Entonces típico se puede expresar:

Regla 1 : Si x es A_1 y y es B_1,

entonces $f_1 = p_1 x + q_1 y + r_1$,

Regla 2 : Si x es A_2 y y es B_2,

entonces $f_2 = p_2 x + q_2 y + r_2$,

La figura 3.3 ilustra el mecanismo de razonamiento para este modelo.

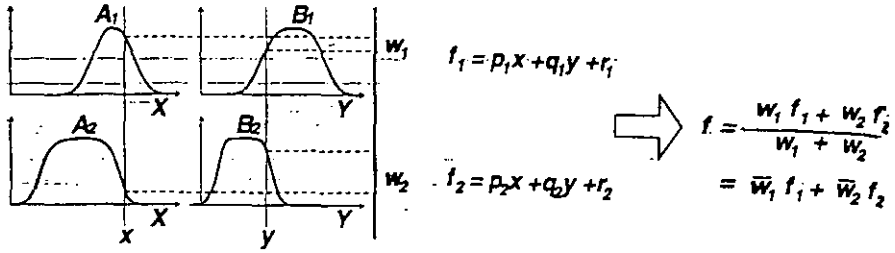

FIGURA 3.3 El Razonamiento Aproximado representado representado en terminos de conjuntos y lógica difusa.

La estructura ANFIS equivalente se muestra en la figura 3.4, donde nodos de la misma capa tienen funciones similares, como se describirá en seguida. (el nodo de salida y en la capa *l* se denota $O_{l,i}$.

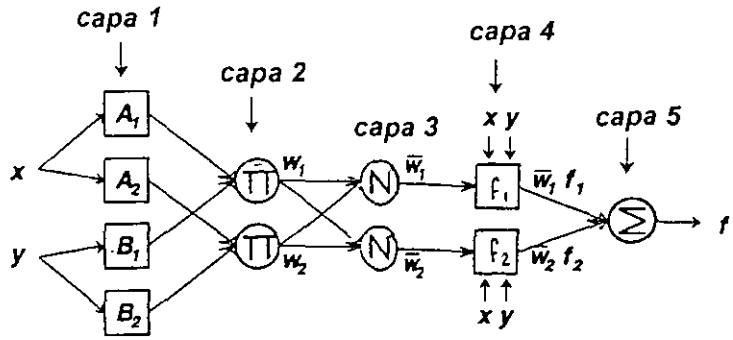

FIGURA 3.4 Equivalencia de la figura 3.3 en terminos de una red adaptativa ANFIS.

Capa 1

Cada nodo *i* en esta capa es un nodo adaptivo con una salida de nodo definida por

$O_{1,i} = \mu_{A_i}(x)$ para $i = 1,2$, o

$O_{1,i} = \mu_{B_{i-2}}(x)$ para $i = 3,4$

donde x (o y) es la entrada al nodo y A_i (o B_{i-2}) es un conjunto difuso asociado con este

nodo. Por tanto, las salidas de esta capa son los valores de membresía de la parte de la premisa. Aquí las funciones de membresía de los conjuntos A_i y B_i, pueden ser cualesquiera funciones parametrizadas. Por ejemplo A_i puede caracterizarse por función de forma de campana

$$\mu_{A_i}(x) = \frac{1}{1+\left[\left(\frac{x-c_i}{a_i}\right)\right]^{b_i}} \qquad (3.6)$$

donde:

$\{a_i, b_i, c_i\}$ es conjunto de parámetros.

Los parámetros en esta capa se denominan *parámetros de premisa.*

Capa 2

Cada nodo en esta capa es un nodo fijo denominado $\prod$, el cual multiplica la señal entrante y entrega el producto. Por ejemplo

$$O_{2,i} = w_i = \mu_{A_i}(x) \times \mu_{B_i}(x), \quad i = 1,2 \qquad (3.7)$$

Cada salida de nodo representa la intensidad o fuerza de activación de cada regla. Debe aclararse que cualquier otro operador T-norma que lleven a cabo operaciones difusas AND, puede ser utilizada como función de nodo en esta capa.

Capa 3

Todo nodo en esta capa es un nodo fijo denominado N. El i-ésimo nodo calcula la razón de la intensidad de activación dela i-ésima regla a la suma de todas las fuerzas o intensidades de activación de todas las reglas

$$O_{3,i} = \overline{w}_i = \frac{w_i}{w_1 + w_2} \quad i = 1,2 \qquad (3.8)$$

Por conveniencia, las salidas de esta capa se denominarán *intensidades de activación normalizadas.*

Capa 4

Cada nodo y en esta capa es un nodo adaptivo con una función de nodo

$$O_{4,i} = \overline{w}_i f_i = \overline{w}_i (p_i x + q_i y + r_i) \qquad (3.9)$$

donde

$\overline{w}_i$ es la salida de la capa 3 y $\{a_i, b_i, c_i\}$ es el conjunto de parámetros.

Los parámetros de esta capa se denominan *parámetros de consecuente.*

Capa 5

El nodo único en esta capa es un nodo fijo, denominado $\sum$, que calcula la salida global como la suma de todas sus señales entrantes:

$$O_{5,1} = \text{salida global} = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \qquad (3.10)$$

La arquitectura descrita constituye una red adaptiva que tiene exactamente la misma función que un sistema de inferencias difuso. La arquitectura no es única; las capas 3 y 4 pueden combinarse para obtener una red equivalente de solo cuatro capas. Así mismo, puede ejecutarse una normalización de pesos en la última capa para obtener una red ANFIS de cuatro capas como se muestra en la figura 3.5.

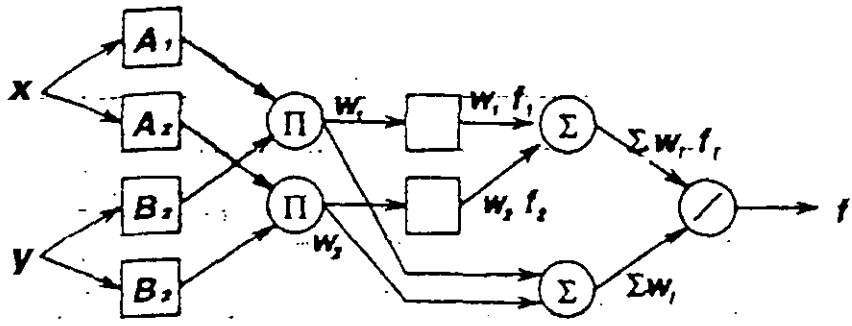

FIGURA 3.5 Red ANFIS eqwuivalente a la de la figura 3.3, de solo cuatro capas y salida normalizada.

B) Algoritmo Híbrido de Aprendizaje

De la arquitectura del modelo ANFIS descrito y en referencia a la figura 3.3, se observa cuando los valores de los parámetros de premisa están fijos, la salida global puede expresarse como una combinación lineal de los parámetros de consecuente. La salida f en la figura citada puede ser reescrita como:

$$f = \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2$$

$$= \overline{w}_1 f_1 + \overline{w}_2 f_2$$

$$= \left(\overline{w}_1 x\right) p_1 + \left(\overline{w}_1 y\right) q_1 + \left(\overline{w}_1\right) r_1$$

$$+\left(\overline{w}_2 x\right)p_2 + \left(\overline{w}_2 y\right)q_2 + \left(\overline{w}_2\right)r_2 \qquad (3.11)$$

Esta expresión es lineal respecto de los parámetros de consecuente:

p_1, q_1, r_1, p_2, q_2 y r_2.

Por lo tanto el algoritmo híbrido de aprendizaje presentado anteriormente en este mismo capitulo puede aplicarse a este caso. Mas específicamente, durante la propagación hacia adelante de las señales del algoritmo híbrido , las salidas de nodo avanzan hasta la capa 4 y los parámetros de consecuente se identifican con los del método de mínimos cuadrados. En la propagación hacia atrás del algoritmo las señales de error retroceden y los parámetros de premisa se modifican por el método del gradiente descendente. La siguiente tabla sintetiza las actividades durante ambas fases del algoritmo.

	Propagación hacia delante	Propagación hacia Atrás
Parámetros de Premisa	Fijos	Gradiente Descendiente
Parámetros de Consecuente	Estimación por Mínimos Cuadrados	Fijos
Señales	Salidas de Nodo	Señales de Error

Las ecuacion (3.10) muestra que existe semejanza entre los modelos neuronales, y el modelo de redes adaptivas ANFIS. Jang y Sun [25], consideran que en realidad estos dos modelos no son mas que dos marcos computacionales para una misma función. Argumentan que bajo ciertas condiciones puede probarse que es así en el caso general.

EL MODELO DE LA NEO-NEURONA DIFUSA

El modelo presentado en la sección anterior no es único; se presentó por dos motivos. Primero, el modelo puede interpretarse como un sistema de inferencia difuso que adquiere características de adaptabilidad al aplicarse en este los conceptos de las redes adaptivas; o bien, puede pensarse como una red adaptiva donde las representaciones dejan de ser puramente numéricas y vectoriales, y pasa a compartirse la interpretación difusa de la información (aunque a nivel de computo básico las variables siguen siendo puramente numéricas). El hecho de que el algoritmo híbrido de aprendizaje pueda aplicarse a ambos modelos acentúa la cercanía de estos modelos.

El segundo motivo, es que la aproximación aqui mostrada es representativa de una gran parte de los enfoques que se han hecho de estos sistemas. En los últimos años se han presentado varios modelos neuro-difusos que han resultado exitosos. En general, estos sistemas interpretan un sistema difuso en términos de un red neuronal , de forma

que cada paso en el proceso es equivalente a una capa, por lo menos, de una red neuronal. De este modo la mayoría de las arquitecturas tienen al menos cuatro capas correspondientes a las operaciones de fuzzificación, intersección (que en nuestro caso correspondería a la identificación de las reglas activadas), implicación (la inferencia difusa, en nuestro caso el *Modus Ponens Generalizado*) y defuzzificación. Además estas arquitecturas difieren las redes neuronales en términos de la uniformidad en los nodos de procesamiento; las RNA, con el fin de ser uniformes, usan en general la misma función de activación o funciones de características similares en todos lo nodos, los sistemas neuro-fuzzy, usan en general, la función adecuada para la etapa que representa la capa. También difieren en las estrategias de interconexión; las RNA en general usan conexiones totales entre capas (salvo en las redes que pueden modificar sus conexiones, y no solo sus intensidades sinápticas); mientras que los sistemas neuro-difusos las interconexiones se definen por el tipo de reglas usadas. Estas diferencias provocan, así mismo, que los algoritmos de entrenamiento se modifiquen o se generalicen, como en el caso del algoritmo híbrido de entrenamiento.

Existen sin embargo, modelos que no se ciñen a la aproximación de las redes adaptivas. Mucho de aquellos modelos pretenden mejorar el desempeño de sistemas difusos conocidos usando estructuras adaptables tomadas del campo de las redes adaptivas, sin embargo, conforme se consolida el paradigma neuro-fuzzy surgen modelos que toman aspectos generales de las redes adaptivas difusas para lograr modelos independientes de sus predecesores difusos o neuronales. En particular, Takeshi Yamakawa ha presentado un modelo que se aparta del paradigma clásico, que presenta caracteristicas que lo hacen atractivo para aplicaciones reales mediante hardware.

Esta sección presenta el modelo de la Neo-Neurona Difusa, para posteriormente definir la forma de implementarse en hardware. Se seguirá aquí la exposición que hace Yamakawa *et al.* [26], una exposición más reciente con aplicación al filtrado y restauración de señales puede encontrarse en Ruan [27].

En el trabajo aludido se describe un modelo de neurona difusa (*fuzzy neuron*) que se denomina *Neo Neurona Difusa* (*neo fuzzy neuron*), cuya característica de transferencia sináptica (las funciones asociadas a los vínculos entre neuronas) es no lineal y cuyo soma (el cuerpo de la neurona, su función de nodo) se modela por una operación de suma. La exposición es breve y concisa y será útil presentarla íntegramente, en seguida se expone la parte del trabajo, correspondiente a la exposición del modelo de neo neurona difusa, en el cual:

> *... cada sinapsis no lineal se caracteriza por un conjunto de reglas de implicación con pesos* (o intensidades) *en forma de singletos en los consecuente...*
>
> prosigue el trabajo con la
>
> *ESTRUCTURA Y ALGORITMO DE APRENDIZAJE DE UNA NEO NEURONA DIFUSA*
>
> *La estructura de una neo neurona difusa se muestra en la figura 3.6, donde las características de cada sinapsis se representan por una función no lineal f_i y el soma no presenta ninguna función sigmoidal. La agregación las señales sinápticas se logra mediante una suma algebraica. De modo que la salida de esta neo neurona difusa se puede representar por la siguiente ecuación;*

$$\hat{y} = f_1(x_1) + f_2(x_2) + \ldots + f_m(x_m)$$

$$= \sum_{i=1}^{m} f_i(x_i) \qquad (3.12)$$

La estructura de las sinapsis no lineales se muestra en la figura 3.6 . El espacio de entrada x_i *se divide en varios* segmentos difusos *que son caracterizados por funciones de membresía* $\mu_{i1}, \mu_{i1}, \ldots, \mu_{ij}, \ldots \mu_{in}$ *dentro del mismo rango entre* x_{min} *y* x_{max} *como se muestra en la figura , 1,2,...,j,...,n son números asignados a las etiquetas* (denominación lingüística) *de los segmentos difusos. Las funciones de membresía son seguidas pesos variables* $w_{i1}, w_{i1}, \ldots, w_{ij}, \ldots w_{in}$.

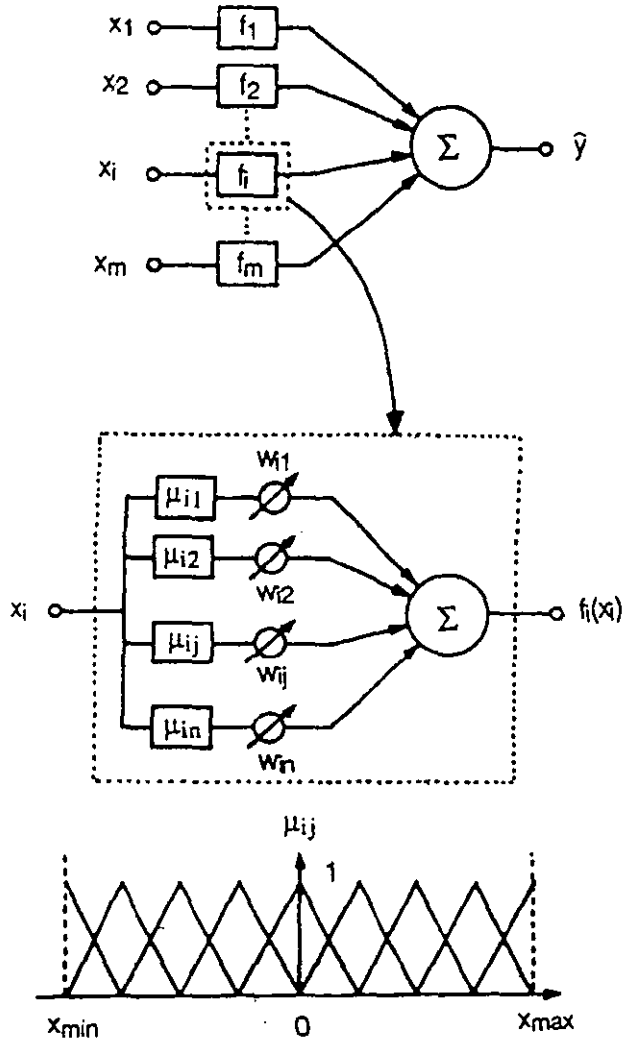

FIGURA 3.6 El esquema superior corresponde a la estrusctura de una Neo Neurona Difusa, cada característica sináptica está representada por una función no lineal f_i. El diagrama al centro es la estructura de dicha sinapsis no lineal, que se describe en terminos de conjuntos y logica difusa. El diagrama inferior representa el espacio de entrada con sus funciones de membresía asignadas.

El mapeo de x_i *a* $f_i(x_i)$ *esta determinado por las inferencias difusas y la defuzzificación. La inferencia difusa adoptadas la de consecuente singleton, es decir que cada peso* w_{ij} *es un valor determinístico* (sic) *tal como 0.3. Debe enfatizarse que cada función de membresía en el antecedente es triangular y complementario (así nombrado por los autores) con sus vecinos. En otras palabras, una señal de entrada activa solo dos funciones de membresía simultáneamente y la suma de los grados de estas dos funciones de membresía vecinas denominadas k y k+1 es siempre igual a 1, es decir*

$$w_{i,k} + w_{i,k+1} = 1 \qquad (3.13)$$

Así que la defuzzificación tomando el centro de gravedad no precisa una división y la salida de la neo neurona puede ser representada por la siguiente ecuación:

$$f_i(x_i) = \frac{\sum_{j=1}^{n} \mu_{ij}(x_i) \cdot w_{ij}}{\sum_{j=1}^{n} \mu_{ij}(x_i)} = \frac{\mu_{ik}(x_i) \cdot w_{ik} + \mu_{i,k+1}(x_i) \cdot w_{ik+1}}{\mu_{ik}(x_i) + \mu_{i,k+1}(x_i)}$$

$$= \mu_{ik}(x_i) \cdot w_{ik} + \mu_{i,k+1}(x_i) \cdot w_{ik+1} \qquad (3.14)$$

Esta ecuación puede ser llevada a cabo por la arquitectura mostrada en la figura.

Los pesos w_{ij}[1] *son asignados por aprendizaje , cuya regla (algoritmo) es descrito por n reglas Si-Entones, como se muestra en seguida;*

Regla Número j

> *SI* x_i *yace en el segmento difuso* μ_{ij} *, entonces el peso correspondiente* w_{ij} *debe ser incrementado en forma directamente proporcional al error de salida* $(y - \hat{y})$*, ya que el error es causado por el peso.*

Esta proposición puede ser representada por la siguiente ecuación

$$W_{ij}(t+1) = W_{ij}(t) + \alpha_i \mu_{ij}(x_i)(y - \hat{y}) \qquad (3.15)$$

donde α_i *es coeficiente de aprendizaje para la i-ésima sinapsis no lineal.*

Este es el algoritmo de aprendizaje para esta neo neurona difusa. Después del aprendizaje , esta neo neurona difusa facilita la extrapolación de datos.

Este sistema, propuesto por Yamakawa, difiere en varios aspectos de las redes adaptivas. Esto muestra la diversidad de enfoques que se han desarrollado, unos tratando de reunir características deseables de ambos sistemas en uno más general, y otros de modo mas heurístico que pueden introducir mejoras en algún aspecto, como la facilidad de implementación o la transparencia del sistema (como se vio, el paradigma

---

1 Nótese la diferencia con los paradigmas ANFIS, donde el algoritmo es una modificacion de los ya disponibles.

neuronal postula una caja negra que aprende, de ahí la denominación de Kosko "model-free"). Este modelo particular presenta conexiones entre capas con funciones no lineales; de tipo difuso, recuérdese que las redes adaptivas poseen conexiones con la sola función direccional. Las funciones de nodo son homogéneas y es simplemente una función de agregación, mientras que las redes adaptivas concentran la totalidad del proceso no lineal y de agregación en los nodos. Por último, el algoritmo de aprendizaje consiste en un conjunto de reglas, donde el conjunto de parámetros del modelo las redes adaptivas no tiene límites definidos estrictamente, el conjunto de parámetros pasa a ser un conjunto difuso con elementos difusos, sometidos por tanto a las estipulaciones de la teoría de conjuntos difusos y a manipulaciones mediante lógica difusa. Sin embargo este modelo presenta algunas ventajas sobre el ANFIS, sobre todo en la reducción de la complejidad de su arquitectura, y lo torna, por tanto mas atractivo para implementarse en hardware.

CAPITULO IV : DISEÑO DEL SISTEMA

DEFINICIÓN DEL PROBLEMA

La intención de los capítulos previos de este trabajo fue exponer los fundamentos teóricos para necesarios para proponer un esquema de controlador Neuro-Difuso de propósito general. En éste capitulo se concretará la propuesta. Debe enfatizarse que aunque el modelo se ha inspirado en el de la Neo Neurona Difusa, la necesidad de adaptarlo a los recursos disponibles motivó el diseño de un modelo original, y como tal debe proseguirse en la fundamentación teórica y las pruebas empíricas. Dicha tarea excede los límites propuestos en esta tesis, por lo que se limitará a la descripción de los pormenores del diseño.

Aquí se propone un esquema de controlador de propósito general que pueda aplicarse a situaciones diversas donde existan problemas que involucren variables con las siguientes características:

Las variables externas o ambientales son de tipo aleatorio (por ejemplo el efecto que sobre un motor tienen la presión y temperatura del aire).

No son fácilmente medibles o en todo caso resulta costoso.

Las mediciones de dichas variables poseen un alto grado de incertidumbre.

Las relaciones entre variables son no lineales.

Estas características son comunes a muchos de los problemas que en la actualidad se atacan mediante técnicas del llamado *cómputo suave* (que se entiende como la unión de los campos de redes neuronales, lógica difusa y razonamiento probabilístico). Las características de las redes neuronales y de la lógica difusa que son deseables para enfrentar problemas de este tipo son estas: tolerancia al ruido, mapeos de funciones no lineales, generalización de respuesta (es decir la capacidad de responder de un modo aproximado al óptimo en situaciones que son similares pero que no son fácilmente reconocibles como tales) y la ausencia de modelo matemático del sistema a controlar.

En muchas circunstancias el desempeño de un sistema depende de las circunstancias inmediatas prevalecientes al interior y también del estado en que se ejerce determinada acción (un motor de combustión es un buen ejemplo de ello, pues el combustible debe inyectarse en el momento en que se considera que el estado es óptimo para la combustión). Tampoco es poco común que las variables involucradas requieran, para ser monitoreadas, de procedimientos y herramientas muy refinadas y por lo tanto costosas. En muchos casos el monitoreo de estas variables no asegura que se pueda controlar efectivamente al sistema.

Ruido

Además del hecho de la dificultad para acceder las variables internas debe agregarse que muchas de estas variables están sometidas a incertidumbres inherentes.

Para controlar el momento de ejecutar una acción se requerirá un información precisa del estado del sistema y la dinámica del sistema introducirá en las mediciones, forzosamente, una cantidad de ruido que obligará a que la señal sea filtrada. En cuanto las variables físicas como la temperatura, es bien conocido que estas variables poseen incertidumbre asociada, y la teoría que enmarca sus relaciones se basa en la estadística y la teoria de la probabilidad.

No linealidad

También se da el caso de que las relaciones de dependencia de las variables de un sistema son muy complicadas , dependen tanto de variables ambientales como de variables internas; o incluso existe interdependencia de unas variables con otras. Además de esto, los modelos matemáticos o numéricos de estas relaciones son muy difíciles de determinar debido a las dificultades de medición y las incertidumbres involucradas en cada una de ellas. Para sistemas complicados muchos de los modelos, que han podido determinarse a costa de numerosas y largos experimentos, son de tipo numérico (es decir que no hay una función definida) y son de poca precisión, en muchos casos definen solo las curvas extremas de una familia que se espera cubran los casos mas comunes. Las curvas de comportamiento pueden variar para un mismo sistema durante su funcionamiento o si se presentan cambios en el ambiente. Cuando estas curvas pueden ser modeladas matemáticamente, el modelo esta compuesto por funciones no lineales o resultan en ecuaciones diferenciales de orden superior al primero.

Las dificultades escuetamente resumidas aquí tornan sumamente complicado el diseño de un controlador óptimo usando los modelos clásicos del control. La emergencia de modelos alternativos ha ensanchado el repertorio de modelos de solución para problemas como este. Las dificultades características de algunos sistemas señaladas arriba pueden ser solventadas por modelos que permitan un procesamiento de la información de forma mas abierta, considerando las incertidumbres inherentes y con la capacidad de generalización que le permita adaptarse a cambios aleatorios en el comportamiento del sistema. En los tres primeros capítulos de este trabajo se han examinado tres modelos de sistemas de procesamiento de la información y su aplicación al diseño de controladores. El diseño de un control de propósito general usando sistemas Neuro-Difusos, en particular el modelo de la Neo Neurona Difusa, se justifica como la primera fase para el diseño de un controlador destinado a este tipo de problemas.

Generalización:

Las capacidades de generalización de los sistemas neuronales le permiten tratar con sistemas que pueden variar su modelo dependiendo de ciertos parámetros que están fuera del control del diseñador, como son las variables ambientales que afectan a un sistema abierto.

Ausencia de modelo matemático:

La cualidad de poder diseñar un sistema en ausencia de un modelo matemático y a partir de ejemplos en el caso neuronal; o de reglas expresadas lingüísticamente en el caso difuso; torna innecesario el monitorear con precisión todas o la mayoría de las

variables no determinantes[1] involucradas en el desempeño de un sistema complejo.

Tratamiento del ruido y la incertidumbre:

El diseño y el procesamiento distribuido de los sistemas neuronales le confieren la capacidad para tratar con señales ruidosas, además la representación difusa de la información permite que porciones importantes de las cualidades de los datos, como su vaguedad, se preserven durante el procesamiento.

No linealidad del modelo:

Por último, el problema de la no linealidad del modelo puede ser solventada, ya sea mediante el diseño lingüístico de un controlador difuso (como se verá la Neo Neurona Difusa no se ajusta al modelo canónico del controlador difuso), o bien mediante el entrenamiento a partir de datos conocidos. La adaptabilidad de los sistemas neuronales y la representación difusa de la información en las sinapsis, pueden aprovechar los modelos numéricos disponibles de ciertos sistemas. En el caso de que no existan descripciones numéricas, aún se puede someter al sistema a ejemplos directos en la práctica.

No obstante sería muy prematuro aplicar el modelo de Yamakawa tal como él lo presenta. Se pretende, en primer lugar, una implementación en hardware tal que aproveche recursos de bajo costo. El diseño usando microcontroladores se presenta como una buena opción, los microcontroladores capaces de tratar directamente procesos y reglas difusas, particularmente, se presentan como la elección natural. Lo que resta de este trabajo se dedicará al análisis y diseño de un modelo basado en la Neo Neurona Difusa e implementado en un microcontrolador difuso.

Problema típico de evaluación

Para el diseño del controlador de propósito general se seguirá el esquema básico de la Neo Neurona desarrollado por Yamakawa. Este modelo se presento en el capitulo tercero, según su publicación en *Yamakawa et al* [29]. , y puede consultarse allí mismo. Para probarlo se utilizará un problema que se utiliza frecuentemente para evaluar el comportamiento y las prestaciones generales de un modelo dado. Dentro de el conjunto de problemas típicos de evaluación (*benchmarking problems*) se encuentran problemas prácticos como el balanceo de un péndulo invertido o mas generales como la identificación y el modelado de sistemas dinámicos. Precisamente el problema general de la identificación de modelos dinámicos será el que se aborde aquí como punto de partida para el diseño propuesto.

Yamakawa plantea que una neo neurona difusa puede ser combinada con una serie de elementos de retardo para obtener un sistema de procesamiento de una entrada y una salida, como se muestra en la siguiente figura.

---

[1]Las variables principales o determinantes se deben monitorear, sin embargo puede diseñarse un sistema neuronal cuya función sea obtener un modelo aproximado para determinar la importancia de cada variable de acuerdo a los objetivos deseados.

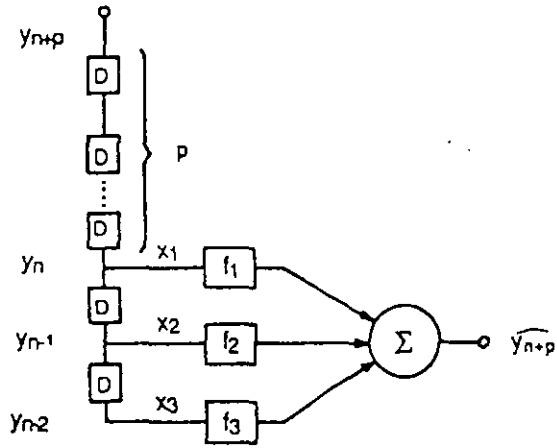

FIGURA 4.1 Configuración del sistema para predecir un numero arbitrario *p* de pasos adelante en un modelo dinámico.

Este sistema puede lograr la identificación de un sistema dinámico aun si este es caótico.

A fin de examinar la factibilidad de la identificación de sistemas de la Neo Neurona Difusa y los elementos de retardo, Yamakawa genera una señal de prueba que es muy caótica y cuyo comportamiento es muy difícil de predecir. La señal fue generada por un sistema dinámico definido por la siguiente fórmula recurrente:

$$\begin{aligned} y_{n+1} &= f_1(x_1)+f_2(x_2)+f_3(x_3) \\ &= f_1(x_n)+f_2(x_{n-1})+f_3(x_{n-2}) \\ &= \frac{5y_n}{1+y_n^2} - 0.5y_n - 0.5y_{n-1} + 0.5y_{n-2} \\ &= \frac{5y_n}{1+y_n^2} - 0.5y_n - 0.5y_{n-1} + 0.5y_{n-2} \quad (4.1) \end{aligned}$$

Los valores iniciales utilizados fueron :

y_1=0.733

y_2=0.234

y_3=0.973

Yamakawa examinó neo neuronas difusas de una sola sinapsis, dos sinapsis, tres,

cuatro y cinco sinápsis para estudiar que tan precisamente pueden identificar el sistema en cuestión.

El numero de segmentos difusos en cada espacio de entrada x_i es 12 y estas doce funciones de membresía son asignadas a cada sinapsis. Cada peso correspondiente a la función de membresía se ajusta de acuerdo a la siguiente ecuación:

$$W_{ij}(t+1) = W_{ij}(t) + \alpha_i \mu_{ij}(x_i)\left(y_{n+1} - \hat{y}_{n+1}\right) \quad (4.2)$$

donde

α_i es el coeficiente de aprendizaje

$\hat{y}_{n+1}$ es la respuesta real del sistema

Para obtener este sistema en forma de dispositivo en hardware, pueden utilizarse dispositivos como microcontroladores o procesadores digitales de señales. En el Departamento de Electrónica de la Facultad de Ingeniería se cuenta con un microcontrolador de reciente aparición en el mercado, el AL220, especializado en el procesamiento y diseño de sistemas a base exclusivamente de lógica difusa. Sin embargo debido a las características de este dispositivo , como se verá, el modelo propuesto de Neo Neurona Difusa no puede implementarse sin alteraciones. En su momento se introducirán y justificarán las modificaciones, pero el modelo de prueba se mantendrá similar al descrito por Yamakawa. Los modelos de cuatro y cinco sinápsis son los que mejor aprovecharán los recursos disponibles en el AL220, y estos serán la base para el diseño.

CARACTERÍSTICAS DE LA NEO NEURONA DIFUSA

En este apartado se mostrarán las diferencias y similitudes del Modelo de la Neo Neurona Difusa con las redes neuronales artificiales y con los Sistemas de control difuso convencionales. Esto con el fin de comprender mas a fondo el modelo y poder posteriormente hacer las adaptaciones que serán precisas para implementarlo aprovechando las características del AL220.

Para hacer mas claros los contrastes de la Neo Neurona Difusa con los sistemas de control difuso y las redes neuronales artificiales, se comparará primero la función sináptica de la Neo Neurona Difusa con los bloques constructivos de un SCD: etapas de fuzzificación etapa de evaluación de reglas y etapa de defuzzificación.

Luego se contrastaran los tres componentes de esta, su arquitectura, función de activación y algoritmo de aprendizaje con su contraparte puramente neuronal. Una comparación detallada con sistemas de redes adaptivas se excluye, solo se hará observaciones pertinentes cuando se compare con las redes neuronales. Las redes adaptivas, en particular el modelo ANFIS, y la neo neurona difusa son dos aproximaciones diferentes a la integración de las redes neuronales y los sistemas difusos. Como ya se ha dicho, las redes adaptivas fueron en un principio un excelente marco teórico que permitió la unificación coherente de estos dos sistemas que ya desde antes exhibían similitudes evidentes, sin embargo el marco teórico no siempre es suficiente para explicar nuevas propuesta, como es el caso de la neo neurona y difusa.

Aspecto difuso

Etapa de fuzzificación

La entrada de las sinapsis de la neo neurona pasan por una etapa en que se determina su grado de pertenencia a los diferentes conjuntos difusos definidos en su universo. Esta etapa es similar ala encontrada en un SCD, por lo general, esta etapa se mantiene inalterable casi en todos los sistemas que utilizan información con representaciones difusas, pues si bien no todos los sistemas difusos hacen uso de las funciones lógicas definidas en la lógica difusa, todos usan las representaciones que surgen de la teoría de conjuntos difusos. Como se verá mas adelante, la neo neurona difusa no utiliza operadores lógicos en la forma canónica descrita en la lógica difusa.

Etapa de evaluación de reglas

El modelo de la neo neurona difusa introduce las características de adaptabilidad mediante pesos variables asociados a cada función de membresía, este hecho modifica las funciones lógicas que han de aplicarse, ello deriva en dos diferencias en la fase de proceso difuso de la neo neurona difusa respecto de la fase de evaluación de reglas de un SCD.

Primero: no se puede identificar una base de reglas que especifique un proceso simbólico derivado de los conocimientos y los métodos organizados de un experto para cumplir una meta global especifica, y no hay por tanto, una correspondencia transparente entre las acciones locales de cada proceso difuso y la tarea global a la que se destina el sistema. Hay, entonces, una pérdida de transparencia general en el sistema. Tal como en las RNA, no es evidente como un proceso local desemboca en el comportamiento global del sistema.

Segundo: Debido a las relaciones geométricas establecidas en los universos del discurso de las entradas, la máquina de inferencias es sustituida por el mismo proceso de defuzzificación, se desvanece con esto el proceso de "inferencia" del sistema y el proceso se reduce a operaciones mas elementales aún que las operaciones lógicas. La introducción de pesos w_{ij} intermedios sustituye las funciones AND entre diversas funciones de membresía y las funciones OR funcionan como elementos de agregación de la información. Con esto, el sistema pierde otra característica que hace a los sistemas de control difuso transparentes; además de las reglas lingüísticas explícitas, tampoco se encuentran procesos lógicos, es decir, las sinapsis no tratan con elementos que puedan identificarse como símbolos, sino con estimaciones representadas en su modalidad difusa.

Etapa de defuzzificación:

En esta etapa, se presentan las siguientes dos diferencias:

Primera: la etapa de defuzzificación en un SCD no puede delimitarse claramente en el modelo de la neo neurona difusa. La defuzzificación esta integrada en el procedimiento mismo de procesamiento (funciones

producto y suma), la geometría de los espacios de entrada hacen innecesario la normalización de la salida y esta disponible al término del proceso suma-producto

Segunda : la salida de la etapa de defuzzificacion no corresponde a la acción de control concreta, sino solo a la función sináptica que ha de ingresar al soma para una etapa final de agregación.

De estas observaciones es patente que la arquitectura de la neo neurona difusa carece de dos de las características más atractivas de los SCD; una es que sus reglas son explícitas lingüísticamente, y por tanto familiares, y la otra que sus reglas son transparentes; las acciones a seguir por el controlador son claras con solo observar las variables y las reglas.

No obstante se preserva una característica valiosa que ya se ha señalado en el capitulo previo: la representación de la información de forma difusa, hecho que permite que el sistema trate los datos al mismo tiempo que su incertidumbre (o mas específicamente, su vaguedad). Esta arquitectura, por tanto, saca provecho de una característica ya apuntada de los sistemas difusos, la separación entre la representación de la información y el procesamiento de esta. Puede considerarse como un caso extremo, ya que no hay aquí un procesamiento por medio de operadores lógicos, sino solo una transformación difusa de la entrada a la salida. Esta transformación se usa como función sináptica de una red neuronal difusa. Las operaciones clásicas de un sistema experto desaparecen y se sustituyen por el procesamiento local encontrado en las redes neuronales, pero la representación de la información mantiene su vaguedad. Esto pone de manifiesto las posibilidades que existen en los sistemas difusos al poder separar e incluso sustituir la etapa de procesamiento con la etapa de representación de la información, lo que no es posible en sistemas basados en lógica bivalente, como el caso de los programas convencionales de computadora.

Aspecto neuronal

Con respecto a las diferencias y similitudes del modelo de la neo neurona con las redes adaptivas, particularmente las redes neuronales artificiales se puntualizará en las tres características que definen una red neuronal en particular: su arquitectura, su función de activación y su algoritmo de aprendizaje.

Arquitectura

Dentro de la arquitectura de la neo neurona hay varios puntos sobresalientes que la diferencian de las redes neuronales ordinarias.

Primero: la representación de la información a través de las sinapsis es difusa. Específicamente, cada entrada en las diferentes sinapsis se representa con dos grados de pertenencia relacionados de una manera especifica (son funciones complementarias), sin embargo, las cualidades distintivas de los procesos neuronales, que consisten en representar, procesar y almacenar la información de manera local y distribuida se mantienen, pues la salida global de las neuronas afecta a cada sinapsis de manera particular.

Segundo: no figuran capas ocultas en la red, solo existen los nodos de entrada y salida, además la conexión es total y no hay conexiones recursivas. Cuando las funciones de activación son lineales, las capas ocultas resultan innecesarias, pues la función global se puede expresar como función lineal de la entrada. En este caso la función del soma de la neo neurona consiste en una suma aritmética, cabría pensar, entonces, que las capas ocultas resultan innecesarias. Esto sin embargo no es correcto, ya que los enlaces sinápticos operan sobre la entrada con una función mas complicada. Esto nos lleva al siguiente punto.

Tercero: A diferencia de las redes adaptivas, las conexiones sinápticas, además de su función direccional, cumplen con una función de excitación o inhibición. Esta función de excitación o inhibición es diferente de la encontrada en las redes neuronales comunes. En la neo neurona difusa la operación de las sinapsis no consiste solamente en un escalamiento de la señal, sino que es precisamente en las sinapsis donde se lleva a cabo la mayor parte del procesamiento y donde, a causa del proceso de fuzzificación se introducen las funciones no lineales (en las RNA convencionales la función somática o función de activación es donde se introducen las funciones no lineales). Si se pretendiera diseñar un sistema multicapas, este último hecho tornaría mas difícil su diseño, pues no es claro como afectarían el desempeño del sistema las capas ocultas. Este tema es, de primera instancia, demasiado complicado como para seguirse tratando aquí y se opta solo por usar el sistema con una sola capa.

Cuarto: No hay presentes salidas múltiples, el sistema no presenta procesos que relacionen un espacio n dimensional de entrada con un espacio de salida de m dimensiones, para el problema propuesto aquí es suficiente un mapeo del tipo $R \rightarrow R^n$, sin embargo, para aplicaciones como clasificación, reconocimiento de patrones o control de varias variables, el sistema puede extenderse sin demasiada dificultad agregando neo neuronas difusas en paralelo, usando las mismas entradas, la misma arquitectura y el mismo algoritmo de aprendizaje. El ajuste de los pesos asociados a los conjuntos difusos en los espacios de entrada se haría de acuerdo a el conjunto de entrenamiento apropiado para cada neurona.

Función de activación

En términos estrictos la función de activación en el soma es la función identidad $f(x)=x$. En el soma se ejecuta, al igual que en una red neuronal una función de suma aritmética de las señales ponderadas provenientes de las sinapsis. La diferencia, como se ha mencionado ya, radica en la función sináptica; mientras en las RNA la función de las sinapsis consiste en escalar la señal de entrada, es decir aplicar la función $f(x)=xw_{ij}$. En el modelo de la neo neurona difusa, esta función sináptica consiste en un proceso difuso similar a la inferencia difusa usando operadores aritméticos. La función aplicada es

$$f_i(x_i)= \mu_{i,k}(x_i)w_{i,k}(x_i)+\mu_{i,k+1}(x_i)w_{i,k+1}(x_i) \qquad (4.3)$$

donde k es el índice de la primera función μ_{ij}

La distribución de dicho proceso difuso en cada sinapsis compensa la ausencia de función de activación no lineal en el soma.

Algoritmo de aprendizaje

La neo neurona posee la capacidad de adaptar sus respuestas, según éstas se aproximen o no a las respuestas deseadas. Esto se logra modificando las sinápsis, como en la RNA, pero, la mayor complejidad de las funciones sinápticas en la neo neurona obligan a un mayor refinamiento en el ajuste. Los parámetros que se ajustan durante la etapa de aprendizaje son los pesos w_{ij} asociados con cada función de membresía en los universos de entrada. Los pesos se ajustan según la siguiente regla:

Si x_i yace en el segmento difuso μ_{ij},

Entonces su correspondiente peso asociado w_{ij}, debe ser incrementado de manera directamente proporcional al error de salida $y_{n+1} - \hat{y}_{n+1}$

Esta regla se expresa en términos numéricos con la siguiente expresión:

$$W_{ij}(t+1) = W_{ij}(t) + \alpha_i \mu_{ij}(x_i)\left(y_{n+1} - \hat{y}_{n+1}\right) \quad (4.4)$$

donde α_i es el coeficiente de aprendizaje.

Como en la mayoría de los algoritmos de aprendizaje en las memorias asociativas, el objetivo es la minimización del error de salida.

El hecho de que la información en la neo neurona difusa se represente en forma difusa, obliga a usar una regla difusa para ajustar el sistema. El antecedente de esta regla presenta una condición que puede tener diferentes grados de veracidad, pues el hecho de que x_i yazca en el segmento μ_{ij}, no es una condición de cierto o falso, pues como puede verse en la ecuación del algoritmo de aprendizaje, el ajuste se hace según el valor de la función de membresía μ_{ij}, es decir el ajuste es mayor cuanto mayor sea la veracidad de la proposición simple "x_i es μ_{ij}", donde "μ_{ij}" es la etiqueta del conjunto con función de membresía μ_{ij}.

El algoritmo de aprendizaje puede adaptarse para llevar a cabo aprendizaje supervisado o no supervisado. La única diferencia es la forma en que se determinará el error del sistema.

El algoritmo de aprendizaje presentaría problemas en el caso de que se aumentaran capas ocultas de neo neuronas difusas. Pues, al basarse el ajuste de los pesos en una condición difusa y una numérica (el valor del error), no es inmediato como habría de modificarse esta regla para ajustarse a un error difuso intermedio cometido por la primera capa oculta. Esta dificultad, junto con las discutidas anteriormente, ponen de manifiesto que el diseño de un sistema multicapas a base de neuronas difusa requiere de un investigación que está fuera del alcance de este trabajo. Se opta aquí, entonces, definitivamente por el diseño de un sistema de capa simple basada en una Neo Neurona

Difusa.

Los siguientes apartados describirán como se llevará al ámbito del AL220 este modelo teórico. Para ello se discutirán las propiedades de los elementos de procesamiento elegidos para implementar físicamente el modelo de estímulo-respuesta y para diseñar la simulación en computadora del algoritmo de aprendizaje.

CARACTERÍSTICAS DEL AL220

El AL220 es un microcontrolador a base de lógica difusa; posee cuatro entradas analógicas con sendos convertidores A/D de ocho bits y cuatro salidas analógicas, lo que permite conectarlo directamente a sensores mediante acoplamientos adecuados. Este microcontrolador permite controlar las salidas basándose en la combinación y procesamiento difuso de la información de entrada. La salidas pueden fijarse a un valor determinado o ser incrementadas o decrementadas por valores establecidos en un rango de 0 a 255 o por los valores presentes en los registros de entrada y salida.

Los parámetros de la funciones y las reglas difusas se almacenan en memorias de tipo ROM y EEPROM. La información requerida para su operación; el conjunto de parámetros, o, hablando en términos familiares; la programación para una aplicación reside en la EEPROM y se introduce al sistema usando la interfaz y el sistema de desarrollo en Software *INSiGHT IIe*. La programación se hace con declaraciones SI ... ENTONCES..., que constituyen lo que se denomina la base de reglas.

OPERACIÓN DEL AL220

Como se explicó en el primer capítulo de este trabajo, el procesamiento de datos en un sistema de control difuso puede dividirse en al menos tres etapas. El AL220 sigue estas tres etapas y agrega además las de conversión A/D Y D/A. Primero se muestrea la información presente en la entrada y se almacena en su respectivo *latch*. En seguida el *Comparador* (o fuzzificador) compara el contenido de los latches de entrada con las *Ventanas de Comparación* (funciones de membresía), a fin de encontrar valores digitales para los *Términos* (variables lingüísticas). El comparador (fuzzificador) ejecuta, también, las operaciones Máximo de Mínimos (funciones AND y OR), cálculo que determina la regla ganadora. Por último, el *Modo de Salida* (defuzzificador) determina el valor de la acción a seguir dispuesta por la regla ganadora, este valor es enviado a un latch de salida y está disponible para su realimentación interna.

Comparador (fuzzificador)

El Comparador, que corresponde a la etapa de fuzzificación, compara los datos almacenados en cada latch de entrada con las Ventanas de Comparación, que representan las funciones de membresía, para calcular el grado de pertenencia, sin embargo este grado de pertenencia es, para propósitos prácticos, una cantidad entre 0 y 63, representando 0 y 100% de la escala respectivamente. cuando la operación Mínimo ha sido ejecutada sobre todos los Términos o variables lingüísticas en una regla, el valor resultante representa la regla y este es almacenado en todas las reglas que hacen referencia a la entrada en cuestión. Una vez con todas las reglas evaluadas, el dispositivo procede a aplicar la operación Máximo , cuando esta operación se ha ejecutado sobre

todas las reglas que hacen referencia a una misma salida, el valor resultante corresponde al valor de la acción de la regla ganadora, y este se destina al Modo de salida, que dentro del microcontrolador corresponde a la etapa de defuzzificación.

Las reglas se evalúan en el orden en que son dispuestas durante la programación del dispositivo. Cualquier regla puede hacer referencia a cualquier salida y además, las salidas pueden ser referidas repetidamente en un conjunto dado de reglas.

Cuando una regla o un grupo de reglas que afectan a una salida ha sido procesada y la siguiente regla hace referencia a otra salida, el compilador inserta automáticamente el código para la Ultima Regla causando que la salida sea actualizada con el valor de la acción de la regla ganadora. Por lo tanto, durante un ciclo de procesamiento, una salida puede ser actualizada el mismo número de veces que el número de grupos separados de reglas que hagan referencia a la regla ganadora.

Modos de salida (defuzzificador)

Los valores resultantes de las operaciones min-max representan los consecuentes de las implicaciones, se denominan para propósitos prácticos *valores de la acción de las reglas ganadoras* , además de estas, en el AL220 existe un dato de *Modo*, juntos pasan al bloque de *Modo de Salida,* que corresponde, con ciertas reservas, a la etapa de defuzzificación. Dependiendo del valor de la acción y de el modo de salida seleccionado, se tiene la información que ha de pasar al latch de salida. Además se dispone de esta información internamente como posibles entradas a las reglas. Si todas las reglas en un grupo que hacen referencia a la misma salida se evalúan en cero, entonces la salida permanece inalterada.

Métodos de salida

Las diferencias en el defuzzificador son de particular interés para nuestro propósito, pues la defuzzificación en la neo neurona difusa, como se ha visto, es diferente la que se encuentra en los SCD estándar, aquí se presentan las características de la defuzzificación en el AL220, y se comparan con los métodos convencionales de defuzzificación. Como ya se ha dicho antes, la etapa de defuzzificación debe interpretarse en este dispositivo con ciertas reservas.

La defuzzzificación es referida en la documentación del AL220 como Modificación de Salida, esta causa que el Valor de Acción de la regla ganadora modifique una salida, usando uno de los dos métodos disponibles en el dispositivo. Estos dos Métodos de Salida son el inmediato y el de acumulación, que difieren de los métodos habituales de defuzzificación. Ambos se refieren métodos pueden apreciarse en la siguientes figuras, y pueden ser elegidos arbitrariamente para cada regla.

Modo inmediato

Cuando este modo es seleccionado dentro de una regla, se ejecuta el método se salida inmediato, que funciona como una tabla de valores asociados, donde el Valor de Acción asignado a al regla ganadora durante la programación es aplicado a una salida.

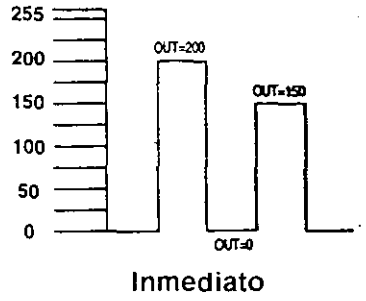

FIGURA 4.2 El modo inmediato en el AL220.

Es decir, este método colocará un valor fijo asociado a la regla que resulte ganadora después de la operación máximo de mínimos, independientemente del estado actual de la salida.

Modo acumulación

Si este modo es seleccionado dentro de una regla, se ejecuta el método se salida por acumulación. Este método incrementa o decrementa la salida existente de acuerdo al Valor de Acción de la regla ganadora. La salida es, por tanto, una función de la acción actual y la salida previa.

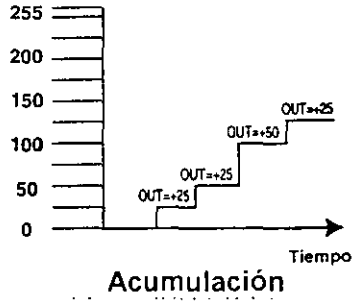

FIGURA 4.3 El modo Acumulación en el AL220.

DIVERGENCIAS DEL AL220 RESPECTO DE LOS SCD ORDINARIOS

Como puede advertirse de la precedente descripción, el AL220 difiere en algunos aspectos de un sistema de control difuso como se describió en el capítulo uno de este trabajo.

Primero: Fuzzificación

El método de fuzzificación corresponde aproximadamente a los métodos usuales de fuzzificación, salvo que solo hay 64 posibles valores de verdad (seis bits de resolución).

Segundo: Evaluación de reglas

La etapa de Evaluación de Reglas se ejecuta, en el AL220 al mismo tiempo que la etapa de fuzzificación, la evaluación de reglas se lleva a cabo mediante operadores min y max para implementar las funciones lógicas AND y OR. Esto acelera el proceso en su conjunto, pero al no disponerse de otros tipos de funciones t-norma y t-conorma, el diseño debe ceñirse a estas particularidades. El resultado final consiste en un solo término resultado de las funciones max-min y de ser necesario de una discriminación entre términos similares de acuerdo al orden en que aparecen en la base de reglas. De esta forma, tampoco se dispone de términos ponderados como en el caso de los SCD convencionales, el término resultante del proceso de inferencia esta asociado a una única regla que se denomina Regla Ganadora, los demás datos asociados a esta regla se utilizan en el proceso de defuzzificación.

Tercero: Defuzzificación

La diferencia más patente reside en el método de defuzzificación, esta diferencia afecta la forma en que las reglas de la Base de Reglas se conforman. Mientras que en una base de reglas difusas los términos involucrados tanto en el antecedente como en el consecuente son variables lingüísticas; en el AL220 el consecuente consiste de la referencia a una salida, de un modo de salida y de un valor de acción. Si durante la etapa de evaluación de reglas, el proceso da como resultado alguno de los términos que forman parte una regla especifica; su conjunto de parámetros, aunado a la información presente en el latch de salida determinan el valor que se presentará en la salida en la siguiente acción.

De los comentarios anteriores es evidente que la arquitectura del AL220 se ajusta solo en cierta medida a la arquitectura de un SCD convencional. La parte que más se aleja del diseño normal es la parte de la defuzzificación, sin embargo, como se ha señalado en un apartado anterior, el modelo que se pretende implementar - la neo neurona difusa - tampoco se ajusta exactamente al modelo común de SCD, difiriendo de este sobre todo en la etapa de evaluación de reglas y en la de defuzzificación, esta última cualidad en coincidencia con el diseño del AL220.

Estos hechos muestran que:

- La etapa de fuzzificación de la información en la neo neurona difusa puede ser cumplida por el AL220 definiendo adecuadamente las funciones de membresía o Ventanas del Comparador dentro del conjunto de parámetros con que se programa el AL220. En este caso las funciones se ingresaran de modo que cumplan con la condición de ser todos Adyacentes Complementarios.
- La etapa de evaluación de reglas, tanto en el modelo de la Neo Neurona

como de en AL220 se lleva a cabo durante el mismo proceso de fuzzificación, en este caso habrán de ajustarse de manera que se obtenga la expresión definida en el modelo de neo neurona difusa, estas modificaciones se explicarán en la sección de diseño del sistema.

- El proceso de defuzzificación se puede omitir del AL220 y ser ejecutado externamente por sumadores analógicos, o bien usar una rutina interna.

PARÁMETROS DE PROGRAMACIÓN DEL AL220

Para comenzar el diseño del dispositivo final es necesario conocer los parámetros que regulan el AL220, pues a diferencia de un microcontrolador convencional, el AL220 no recibe una serie de instrucciones que determinan su comportamiento en cada instante, sino que recibe un conjunto de parámetros asociados al modelo difuso que se requiere implementar. El AL220 posee una memoria EEPROM de 256x8 donde los parámetros que regulan las acciones del controlador son almacenadas. Estos parámetros se generan por medio del sistema de desarrollo INSiGHT IIe, especialmente diseñado para este fin. La programación de estos parámetros se hace mediante interfaces gráficas y etiquetas lingüísticas para definir las variables de entrada, los conjuntos de reglas que rigen cada salida y las acciones a seguir en cada caso.

Para la programación del AL220 existen tres subconjuntos de parámetros que definen las aciones: Las definiciones de variables de entrada y salida. La definición de las funciones de membresía o Ventanas del Comparador y por último la Definición de Reglas.

Definición de variables de entrada y salida.

Cada una de las cuatro entradas analógicas del AL220 puede corresponder a un universo del discurso independiente. Para hacer esto, deben asignársele a cada entrada una etiqueta lingüística diferente, etiqueta que se usará para definir los conjuntos en su universo del discurso. A las cuatro salidas se les puede asignar, así mismo, etiquetas lingüísticas. Cuando la parte consecuente de una regla hace referencia a la etiqueta lingüística de una salida, la acción a seguir, dado por el Valor de Acción se ejecutará, en caso que así se determine, en la salida asociada.

De este modo tenemos solo dos parámetros asociados para cada entrada:

(ETIQUETA LINGÜÍSTICA DE ENTRADA , ENTRADA ASOCIADA)

y

(ETIQUETA LINGÜÍSTICA DE SALIDA , SALIDA ASOCIADA)

En este trabajo, este conjunto de parámetros será resumido en una tabla como la siguiente:

Entrada física	Etiqueta de Entrada	Salida física	Etiqueta de salida

Definición de las funciones de membresía

Este conjunto de parámetros tiene dos funciones, la primera, consiste en organizar el universo del discurso de una variable de entrada en grupos de entradas que corresponden a las funciones de membresía. La segunda es definir automáticamente los términos disponibles para ser usados en la parte antecedente de las reglas.

Las funciones de membresía se definen, dentro del AL220 como triadas asociadas de parámetros, consistentes en lo siguientes: Amplitud (*width*), Centro (*center*), y Tipo (*type*). El parámetro Centro coloca la función de membresía una locación particular dentro del universo del discurso que, como se vio consta de 256 valores discretos posibles, correspondientes a los valores posibles permitidos por la resolución de ocho bits del convertidor. El parámetro Ancho determina que tan similar puede ser una entrada al valor del parámetro Centro para ser todavía considerado como miembro del conjunto definido por la función de membresía, en otras palabras es el soporte de la función de membresía, en el cual el grado de membresía se calcula mediante una función lineal.

Tanto el valor del parámetro Centro como el de Amplitud pueden ser valores fijos determinados entre 0 y 255 o bien, pueden hacerse "flotantes" al obtener su correspondiente valor de entre los disponibles en los latches de entrada o de salida, los valores flotantes hacen, pues, referencia a los valores presentes en las entradas o salidas durante ese preciso ciclo de procesamiento, a esta función se le denomina en la documentación del AL220 como *función de membresía flotante*.

El parámetro Tipo asocia una de las seis formas de función de membresía disponibles en el AL220. Estas formas son simplemente funciones de membresía de tipo trapezoidal y de acuerdo a su características se denominan:

Izquierda Inclusiva (II),
Derecha Inclusiva (DI),
Simétrica Inclusiva (SI),
Izquierda Exclusiva (IE),
Derecha Exclusiva (DE),
Simétrica Exclusiva (SE).

Esta formas de función pueden ser combinadas en un mismo universo del discurso, mas no en una misma función.

Este conjunto de parámetros debe asociarse a una etiqueta lingüística diferente a las de entrada y salida, a fin de dar una noción de las características de la entrada que representara, etiquetas como *pequeño* o *muypequeño* pueden resultar convenientes.

De modo que una función de membresía queda totalmente definida por estos cuatro parámetros asociados:

ETIQUETA DE CONJUNTO, (Centro, Amplitud, Tipo).

Además, dentro de esta subconjunto de parámetros debe asociarse cada función de membresía con una entrada analógica, ello se hace asociando la etiqueta de entrada con la etiqueta de función mediante una predicación, es decir:

ETIQUETA DE ENTRADA es ETIQUETA DE CONJUNTO.

Tal como se explicó en el capitulo primero, la proposición A es B, en el marco de

la lógica difusa, puede tener muchos grados de veracidad dependiendo de los parámetros asociados a la función de membresía. La predicación se realiza automáticamente dentro del entorno del sistema de desarrollo, solo se seleccionan, en los campos destinados para ello, las etiquetas disponibles (previamente definidas) para la entrada y la función de membresía. Estas proposiciones predicativas son fácilmente comprendidas por el diseñador y resumen de manera transparente las asociaciones de parámetros que se han especificado. Cada una de estas proposiciones constituye un término que puede ser usado para formar los antecedentes de las reglas.

En este trabajo, este conjunto de parámetros será resumido en una tabla como la siguiente:

Variable	Etiqueta de conjunto	Centro	Amplitud	Tipo	Término resultante

Definición de reglas

Las reglas en el AL220 combinan uno o mas de los términos generados por la asociación referida en los párrafos anteriores; términos del tipo A es B, donde A es un etiqueta de entrada y B es una etiqueta de función. La combinación de estos términos se hace mediante operadores AND, que en el AL220 se lleva a cabo por medio de funciones min, esta combinación constituye el antecedente de una regla. El antecedente así formado se asocia con una parte consecuente, este consecuente contiene una acción o respuesta que es preciso definir de acuerdo al comportamiento que se desea del controlador. La asociación resulta en proposiciones compuestas *Si...Entonces.* Cada salida activa debe asociarse con al menos una parte consecuente de una regla en la base de reglas. La parte consecuente consiste entonces de una etiqueta de salida; que le indica la salida sobre la que habrá de manifestarse la respuesta, la otra parte del consecuente es precisamente la acción a tomar

La acción asociada al consecuente de una regla puede tomar una de dos formas. La acción en *modo inmediato*, que se representa en el sistema de desarrollo como "="; copia un valor absoluto al latch de salida. Este valor puede provenir de otra entrada, salida o constante fija en memoria y definida de antemano. La segunda forma es la acción en *modo de acumulación*, representada como "+"; esta agrega o sustrae su valor de la salida presente, de la misma forma que el modo inmediato, este valor puede provenir de otra entrada, salida o constante fija.

De modo que las reglas se construyen a partir de los siguientes parámetros:

(TÉRMINOS DEL ANTECEDENTE, ETIQUETA DE SALIDA, ACCIÓN)

donde

"TÉRMINOS DEL ANTECEDENTE" esta constituido por los términos:

T1, T2, ..., Tn.

"ETIQUETA DE SALIDA" es una de las etiquetas asignadas a las salidas, y sobre la cual se aplicará la acción de la regla.

"ACCIÓN" esta constituido por los siguientes parámetros:

modo (+ o =), valor de acción.

Si en la parte de modo aparece un símbolo = se refiere al modo inmediato,

mientras que un + se refiere al modo de acumulación. El valor de acción se representa con un numero entre 0 y 255 si se elige una acción fija, pero si se elige una salida flotante aparecerá la etiqueta de entrada o salida que se haya asignado.

De esta descripción puede observarse que en la parte consecuente de las reglas no figuran conjuntos difusos, sino que se especifican las acciones concretas a seguir. Es decir que, diversamente de los SCD ordinarios, el AL220 integra también la etapa de defuzzificación a la etapa de evaluación de reglas.

En este trabajo, este conjunto de parámetros será resumido en una tabla como la siguiente:

Términos del antecedente	Salida	Modo de salida	Acción	Orden de ejecución

El orden de ejecución se muestra explícitamente porque no es trivial, como se mostrará en el ejemplo de aplicación del siguiente apartado.

En este conjunto se agregará una tabla mas para hacer más claras las reglas lingüísticas resultantes:

Orden de Ejecución	Antecedente SI	Consecuente ENTONCES

CARACTERÍSTICAS DE TEMPORIZACIÓN DEL AL220 Y SU APLICACIÓN COMO ELEMENTO DE RETRASO

La fase de prueba, como se ha dicho al principio de este capitulo, se llevará a cabo con un problema general de evaluación: la identificación de un sistema dinámico, esta prueba precisará que una misma señal de entrada se presente varias veces pero cada vez con un cierto retardo respecto de la señal original.

A pesar de que el AL220 esta diseñado específicamente para procesos difusos, estos no se ejecutan todos simultáneamente, sino en sucesión, de allí que deba tenerse cuidado en el orden en que se ejecutará el proceso. En aplicaciones como la generación de señales retrasadas puede programarse en un primer AL220 como elemento generador de cuatro retardos analógicos externos. Para esto deben tomarse en cuenta las características de temporización del dispositivo, las cuales se resumen en seguida, de acuerdo con la documentación del fabricante.

Temporización

Las tres partes de la arquitectura del dispositivo que influyen en la temporización son el convertidor A/D, el procesador de reglas y el convertidor D/A de salida.

La velocidad de proceso depende tanto de la frecuencia del reloj y el número de

ciclos de reloj que son precisos para completar un ciclo de procesamiento, que para el caso del AL220 es 1024. El ciclo consiste en muestrear los datos de entrada, procesarlos de acuerdo a las reglas almacenadas en memoria. La frecuencia del reloj puede variar entre 1Mhz y 10Mhz.

Conversión de la entrada

Las entradas son muestreadas y almacenadas internamente en su latch correspondiente, en periodos sucesivos de 256 ciclos de reloj cada uno. por tanto, se requiere de un ciclo de procesamiento completo para completar las cuatro entradas, después de lo cual el ciclo se repite. Con el dispositivo trabajando a la máxima velocidad, la frecuencia de muestreo es 10Mhz/(4(256ciclos))=10^6/1024= 9.765khz, El teorema dell muestreo nos dice que para muestrear sin perdidas una señal de frecuencia *f* es necesario una frecuencia de muestreo *2f*. Asi que el ancho de banda del dispositivo es de (9.765khz)/2= 4.882 khz.

Temporización del procesamiento

El primer ciclo de procesamiento comienza después de que un primer ciclo, destinado a adquirir los datos de entrada, ha sido completado. Los ciclos de procesamiento constan de 1024 ciclos de reloj sin importar el numero de variables lingüísticas o reglas empleadas.

Realimentación interna

Cuando los datos en los latch de salida son integradas a un bucle interno para ser usados como entradas, se demoran respecto de las entradas analógicas por los primeros 1024 pulsos de reloj que derivan del ciclo de muestreo inicial, después de eso, como los latches de salida son actualizados durante el procesamiento, los datos realimentados son usados como entradas.

Operación en modo *prescale*

El AL220 posee un contador de preescala de 8 bits que permite al dispositivo permanecer inactivo durante 1024 ciclos de reloj, luego de los cuales el dispositivo permanece activo durante otros 1024 ciclos (suficientes para completar el muestreo de las cuatro entradas). Esta cualidad puede aprovecharse para reducir la velocidad de operación a la mitad.

Para generar señales retardadas, pueden usarse las retroalimentaciones internas: Para ello basta con asignar un conjunto difuso que abarque todo el universo de entrada de la señal a retardar como antecedente en el primer conjunto de reglas que controlan la primera salida, como el proceso no se lleva a cabo sino posteriormente al ciclo de procesamiento inicial, pueden asignarse la primera salida como entrada al segundo conjunto de reglas de procesamiento que controlan la segunda salida, de modo que la señal pase integra a la salida dos, solo hasta después de que se halla completado el ciclo de procesamiento. Esta, a su vez, se asigna como entrada al tercer conjunto de reglas que controlan la tercera salida. Con la cuarta salida se hace algo similar.

El AL220 posee la cualidad de actualizar un latch de salida cada vez que una regla ganadora hace referencia a esta, sin importar el estado del proceso dentro del ciclo de

procesamiento, para evitar perdidas de información debe cuidarse el orden en que se ejecutan las reglas, como ya se ha dicho, las reglas se ejecutan en el orden original en que fueron programadas.

Si se ejecuta las reglas en orden de referencia a la entradas, se tendrá durante el primer ciclo solo latches conteniendo ceros, con lo cual la salidas para el ciclo de procesamiento siguiente se actualizarán con valores cero y en cada ciclo subsiguiente se actualizaran todas con el mismo valor de entrada presente en tal ciclo. Para evitar esto la reglas deben disponerse de manera inversa al orden en que se desea que la señal retrasada se presente en cada salida. Es decir, si se quiere que la salida uno sea la primera en exhibir la señal retardada por un tiempo de 1024 ciclos de reloj, la primera regla que debe aparecer es la que controla la cuarta salida , luego las que controlan la tercera la segunda y la primera.

De este modo el AL220 debe ser programado con los siguientes parámetros para generar un retardo de 1024 ciclos de reloj. Adviértase que el retardo puede ser incrementado usando el modo *prescale* comentado arriba.

Parámetros para generar cuatro retardos de una señal de entrada en las salidas del AL220

ENTRADAS Y SALIDAS

Entrada física	Etiqueta de Entrada	Salida física	Etiqueta de salida
1	ENTRADA	1	RETARDO1
2	-	2	RETARDO2
3	-	3	RETARDO3
4	-	4	RETARDO4

FUNCIONES DE MEMBRESÍA

Variable	Etiqueta de conjunto	Centro	Amplitud	Tipo	Término resultante
ENTRADA	UNIVERSO	0	63	DI	ENTRADA ES UNIVERSO
RETARDO1	UNIVERSO	0	63	DI	RETARDO1 ES UNIVERSO
RETARDO2	UNIVERSO	0	63	DI	RETARDO2 ES UNIVERSO
RETARDO3	UNIVERSO	0	63	DI	RETARDO3 ES UNIVERSO
RETARDO4	UNIVERSO	0	63	DI	RETARDO4 ES UNIVERSO

PARÁMETROS DE LAS REGLAS

Términos del antecedente	Salida	Modo de salida	Acción	Orden de ejecución
RETARDO4 ES UNIVERSO	RETARDO4	=	RETARDO3	1
RETARDO3 ES UNIVERSO	RETARDO3	=	RETARDO2	2
RETARDO2 ES UNIVERSO	RETARDO2	=	RETARDO1	3
RETARDO1 ES UNIVERSO	RETARDO1	=	ENTRADA	4

REGLAS RESULTANTES

Orden de Ejecución	Antecedente SI	Consecuente ENTONCES
1	RETARDO4 ES UNIVERSO	RETARDO4 = RETARDO3
2	RETARDO3 ES UNIVERSO	RETARDO3 = RETARDO2
3	RETARDO2 ES UNIVERSO	RETARDO3 = RETARDO2
4	RETARDO1 ES UNIVERSO	RETARDO1 = ENTRADA

ADAPTACIÓN DE LA NEO NEURONA DIFUSA AL AL220

Se ha señalado en los apartados anteriores, las diferencias del modelo de la neo neurona difusa y de el AL220 con el modelo ordinario de controlador difuso expuesto en el primer capitulo. Se han mencionado también los diferencias entre el funcionamiento del AL220 y el modelo de la neo neurona difusa. Se advirtió que el AL220 integra en el mismo procedimiento de evaluación de reglas, la etapa de defuzzificación; mientras que en la neo neurona se utiliza un método simplificado gracias a la geometría de los universos de entrada; lo que elimina la necesidad de normalizar la salida de la defuzzificación. Este hecho puede ser aprovechado por el AL220; como se verá más adelante, pues como se ha dicho, en este no se realiza un proceso normal de defuzzificación, sino que se asigna uno de dos modos de salida: inmediato y por adición. La etapa de procesamiento sináptico en la neo neurona (funciones f_i) difieren en dos aspectos del procesamiento de reglas del AL220, que se resumen en las siguientes

Consideraciones.

Primera: las operaciones lógicas se llevan acabo utilizando diferentes variantes de operadores t-norma y t-conorma. La neo neurona difusa emplea los operadores aritméticos canónicos (*,+), mientras que el AL220 utiliza funciones de discriminación mínimo y máximo (min, max).

Segunda: cada función de membresía posee un peso asociado previamente a la función de agregación. Este peso, como se ha explicado, es determinado mediante aprendizaje y es lo que le confiere al sistema su adaptabilidad. Sin embargo, en el Al 220, debido a su rígida estructura de procesamiento, dicho peso no puede ser introducido antes de la función de agregación.

Este último aspecto es el que parece disponer los mayores obstáculos para adaptar el modelo de la neo neurona a la arquitectura del AL220, sin embargo, las característica apuntada en la observación primera puede ser utilizada ventajosamente para resolver este inconveniente. Para hacer esto tómese en cuenta que:

El peso asociado a cada función de membresía es de un valor que permanece fijo durante la operación de respuesta a estimulo de la neo neurona, solo se modifica durante el proceso de aprendizaje.

La salida de la función de agregación estará influenciada sobre todo por la intensidad o el grado en que la entrada pertenezca a un determinado

conjunto y el valor de su peso asociado.

Al disponer el AL220 solo de funciones de discriminación min-max, puede representarse la influencia arriba descrita, asignando el peso asociado al conjunto como el valor de acción correspondiente a la regla de la que forma parte el conjunto difuso; cada regla constara de un solo conjunto como término del antecedente y su peso asociado como valor de acción en modo inmediato. De este modo, el valor del peso se verá reflejado en la salida siempre y cuando este asociado al conjunto o función de membresía de la que la entrada sea miembro con mayor intensidad o en mayor grado, que es lo que se busca en las funciones f_i del modelo sináptico de la neo neurona difusa pero usando operadores aritméticos producto-suma.

Para conseguir esto el AL220 deberá programarse con los siguientes parámetros, para una sinapsis modificada:

Parámetros para implementar las funciones fi del modelo de neo neurona difusa

Se especificarán los parámetros para una sola sinapsis, las demás se diseñan en forma similar cambiando los subíndices.

ENTRADAS Y SALIDAS

Entrada física	Etiqueta de Entrada	Salida física	Etiqueta de salida
1	ENTRADA1	1	F1
2	-	2	-
3	-	3	-
4	-	4	-

FUNCIONES DE MEMBRESÍA

Se divide el espacio de entrada en conjuntos con soportes similares, se usan 12 conjuntos en cada universo para probarlo como identificador de sistemas dinámicos. El universo tiene solo una resolución de 256 niveles y por tanto el soporte de cada conjunto es de 42 unidades, excepto en los conjuntos primero y decimosegundo, que tiene un soporte de 23 unidades y forma trapezoide, a fin de abarcar todo el espacio de entrada.

Variable	Etiqueta de conjunto	Centro	Amplitud	Tipo	Término resultante
ENTRADA1	M101	0	23	II	ENTRADA1 ES M101
ENTRADA1	M102	23	21	SI	ENTRADA1 ES M102
ENTRADA1	M103	44	21	SI	ENTRADA1 ES M103
ENTRADA1	M104	65	21	SI	ENTRADA1 ES M104
ENTRADA1	M111	232	21	SI	ENTRADA1 ES M111

ENTRADA1	M112	253	23	DI	ENTRADA1 ES M112

PARÁMETROS DE LAS REGLAS

Cada regla consta de un solo termino en el antecedente y el consecuente es el peso asociado w_{ij}, que se obtendrá mediante aprendizaje, lo que será el objeto de una sección ulterior.

Términos del antecedente	Salida	Modo de salida	Acción	Orden de ejecución
ENTRADA1 ES M101	F1	=	w_{11}	1
ENTRADA1 ES M102	F1	=	w_{12}	2
ENTRADA1 ES M103	F1	=	w_{13}	3
ENTRADA1 ES M111	F1	=	w_{111}	11
ENTRADA1 ES M112	F1	=	w_{112}	12

REGLAS RESULTANTES

Orden de Ejecución	Antecedente SI	Consecuente ENTONCES
1	ENTRADA1 ES M101	F1 = w_{11}
2	ENTRADA1 ES M102	F1 = w_{12}
3	ENTRADA1 ES M103	F1 = w_{13}
11	ENTRADA1 ES M111	F1 = w_{111}
12	ENTRADA1 ES M112	F1 = w_{112}

Agregación de las sinapsis

La agregación propia del soma, es la última etapa de la neo neurona. Como no existe función de activación a nivel del soma de la neurona, la función es simplemente una suma de entradas, estas entradas son las salidas provenientes del AL220; salidas analógicas y por tanto función del soma puede ser llevada a cabo por un sumador lineal de entradas múltiples a base de op amp, o bien, en el AL220 puede agregarse un conjunto final de reglas que usen el modo de salida por adición asignando a los valores de acciones flotantes la misma salida, usando los valores de salida presentes en las salidas restantes.

Esta última opción será la utilizada para el caso de cuatro sinapsis. El conjunto de

parámetros y de reglas que deberán aparecer en la parte final del listado, actuando solo sobre la salida 4 para disminuir el tiempo de actualización de las salidas, pues esta es la última salida que se actualiza durante la etapa de sinapsis. Los parámetros se listan en seguida.

ENTRADAS Y SALIDAS

Entrada física	Etiqueta de Entrada	Salida física	Etiqueta de salida
1	ENTRADA1	1	-
2	-	2	-
3	-	3	-
4	-	4	F4

FUNCIONES DE MEMBRESÍA

Variable	Etiqueta de conjunto	Centro	Amplitud	Tipo	Termino resultante
ENTRADA1	UNIVERSO	0	63	DI	ENTRADA1 ES UNIVERSO

PARÁMETROS DE LAS REGLAS

Términos del antecedente	Salida	Modo de salida	Acción	Orden de ejecución
ENTRADA1 ES UNIVERSO	F4	+	F3	1
ENTRADA1 ES UNIVERSO	F4	+	F2	2
ENTRADA1 ES UNIVERSO	F4	+	F1	3

REGLAS RESULTANTES

Orden de Ejecución	Antecedente SI	Consecuente ENTONCES
1	RETARDO4 ES UNIVERSO	F4+F3
2	RETARDO4 ES UNIVERSO	F4+F2

3	RETARDO4 ES UNIVERSO	F4+F1

Para el caso de 5 o mas sinapsis será necesario usar otro AL22O trabajando en paralelo usando además una de sus entradas como entrada solo a la etapa de agregación. Para esto, el segundo AL220 deberá incluir un regla que sume la salida del primer AL220.

ENTRADAS Y SALIDAS

Entrada física	Etiqueta de Entrada	Salida física	Etiqueta de salida
1	ENTRADA5	1	F5
2	-	2	-
3	-	3	-
4	F4	4	SUMA

FUNCIONES DE MEMBRESÍA

Variable	Etiqueta de conjunto	Centro	Amplitud	Tipo	Término resultante
SUMA	UNIVERSO	0	63	DI	SUMA ES UNIVERSO

PARÁMETROS DE LAS REGLAS

Términos del antecedente	Salida	Modo de salida	Acción	Orden de ejecución
SUMA ES UNIVERSO	SUMA	=	F4	1
SUMA ES UNIVERSO	SUMA	+	F5	2

REGLAS RESULTANTES

Orden de Ejecución	Antecedente SI	Consecuente ENTONCES
1	SUMA ES UNIVERSO	SUMA = F4
2	SUMA ES UNIVERSO	F4+F5

Algoritmo de aprendizaje

Se han introducido algunos cambios en la arquitectura original del modelo de la

neo neurona difusa. Específicamente, los cambios fueron estos:

En el modelo de la neo neurona difusa, las operaciones que jugarían el papel de operaciones lógicas en las sinapsis, como ya se mencionó en un apartado anterior, no pueden ponerse en correspondencia transparente con operaciones lógicas o lingüísticas, es decir que han perdido su transparencia. Han sido interpretadas aquí como simples operadores sinápticos difusos sin implicar relaciones de tipo lógico. Estas operaciones han sido modificadas mediante la sustitución de los operadores (*,+) por los (min, max).

El peso ajustable que figura en el proceso difuso de las sinapsis, ha sido colocado como valor de acción de una regla en lugar de formar parte de los productos previos a la agregación de las sinapsis.

Estas modificaciones deben tomarse en cuenta en le momento del aprendizaje , y por tanto, se verán reflejadas en el algoritmo de aprendizaje (al menos en la parte de estimulación-respuesta).

Por cuestiones de flexibilidad y de economía se decidió implementar el algoritmo de aprendizaje en una simulación por computadora. Se ha considerado que un dispositivo especializado en generar los valores de los parámetros adecuados para el dispositivo requeriría de una inversión innecesaria, pues nada asegura que se obtendrán ventajas sobre las simulaciones en software.

El algoritmo de aprendizaje original de Yamakawa *et al.* es este:

Si x_i yace en el segmento difuso μ_{ij} difuso.

Entonces su correspondiente peso asociado w_{ij}, debe ser incrementado de manera directamente proporcional al error de salida $y_{n+1} - \hat{y}_{n+1}$.

Esta regla se expresa en términos numéricos con la siguiente expresión:

$$W_{ij}(t+1) = W_{ij}(t) + \alpha_i \mu_{ij}(x_i)\left(y_{n+1} - \hat{y}_{n+1}\right). \quad (4.4)$$

donde α_i es el coeficiente de aprendizaje.

Del modelo original de la neo neurona difusa se ve que los únicos valores por ajustar son los pesos w_{ij}, que en el caso del nuevo diseño propuesto corresponden a los Valores de Acción en el AL220.

EL MODELO FINAL MODIFICADO

Lo que resta es sólo modificar el modelo estímulo-respuesta que se usará en la simulación, este modelo es el especificado en la parte de diseño de la arquitectura. Como se ha visto si se modifican los operadores aritméticos por los min-max, puede lograrse que el AL220 lleve a cabo operaciones que resultaran en un comportamiento similar al de la neo neurona difusa. Estas modificaciones hacen que el modelo implementado en el

AL220 sea similar, pero no estrictamente equivalente al modelo original de Yamakawa. A este modelo se le denominará en adelante como neo Neurona Difusa Modificada: NNDM.

La formalización del modelo queda fuera del ámbito de este trabajo, pero aquí pueden puntualizarse las siguientes consideraciones heurísticas, tomando en cuenta los casos extremos que se pueden presentar.

a) La entrada a una sinapsis coincide casi completamente con el centro de un conjunto difuso μ_{ij}. En la neo neurona difusa, el valor de la función de membresía correspondiente $\mu_{ij}(x_i)$ será casi 1, el valor de su único conjunto adyacente con valor de membresía $\mu_{ij+1}(x_i)$ será muy próximo a cero. Por lo tanto el valor de la suma en la sinapsis en cuestión será casi w_{ij}

$$\begin{aligned} &\mu_{ij}(x_i) \approx 1, \\ &\mu_{ij}(x_i)W_{ij} \approx W_{ij}, \\ &\mu_{ij+1}(x_i) \approx 0, \\ &\mu_{ij+1}(x_i)W_{ij+1} \approx 0; \\ &\mu_{ij}(x_i)W_{ij} + \mu_{ij+1}(x_i)W_{ij+1} \approx W_{ij} \end{aligned} \qquad (4.5)$$

b) La entrada coincide casi por completo con el punto medio entre los centros de dos conjuntos difusos adyacentes. En tal caso cada conjunto tendrá un valor aproximado de 0.5, es decir los valores de $\mu_{ij}(x_i)$ y $\mu_{ij+1}(x_i)$ serían similares entre sí. Como el algoritmo de aprendizaje depende de estos valores para ajustar los pesos, significa que los pesos serán ajustados en montos similares si la condición persiste, en tal caso el valor final de los w_{ij} serán también similares entre sí. En tal caso, la suma en las sinapsis resultará en un valor parecido a w_{ij}

$$\begin{aligned} &\mu_{ij}(x_i) \approx 0.5, \\ &\mu_{ij+1}(x_i) \approx 0.5, \\ &\mu_{ij}(x_i) \approx \mu_{ij+1}(x_i) \approx 0.5 \\ &\mu_{ij}(x_i)W_{ij} + \mu_{ij+1}(x_i)W_{ij+1} \approx W_{ij} \approx W_{ij+1} \end{aligned} \qquad (4.6)$$

Estas observaciones insinúan una equivalencia entre la neo neurona difusa y la NNDM, y una primera vía de investigación para su formalización.

Así, pues, el modelo final de la NNDM en el AL220 queda como sigue:

Arquitectura

El sistema es en realidad una memoria heteroasociativa consistente de una capa sencilla y no existen conexiones recursivas. Las funciones sinápticas son, a diferencia del modelo original, reglas que emplean el *modus ponens* y su método de composición es del tipo min-max. La defuzzificación es simplemente el modo de salida inmediato del AL220, asignado al valor de acción el valor del peso w_{ij} obtenido mediante aprendizaje.

El modelo final modificado se muestra en seguida:

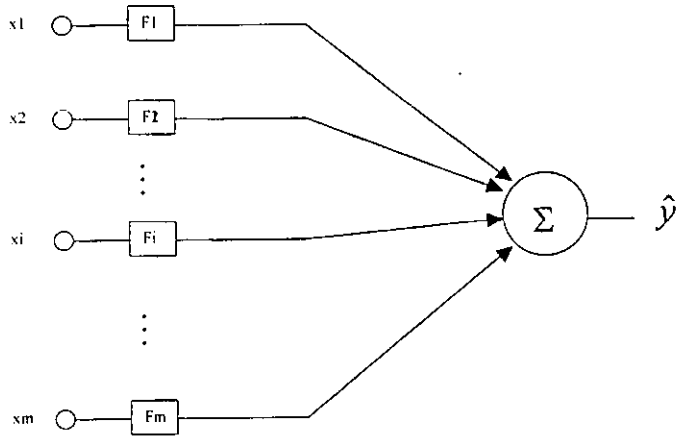

FIGURA 4.3 Arquitectura de la NNDM

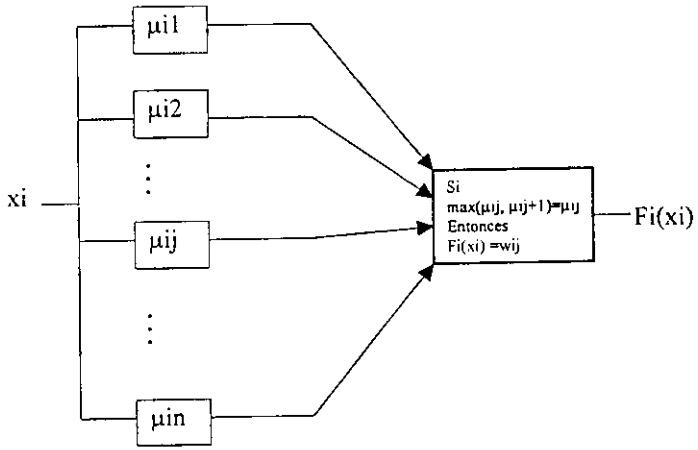

FIGURA 4.4 Funcion sinaptica Fi para la NNDM

Función de activación:

Se utiliza, afín al modelo original, una función de activación lineal. La agregación en el soma se implementa mediante un conjunto de reglas que aplican el modo de adición sobre funciones sinápticas presentes en los latch de salida del AL220.

Algoritmo de aprendizaje:

Las reglas del algoritmo son las mismas que en el modelo original, salvo que el modelo en software usará las representaciones internas del AL220 así como sus reglas, para determinar la salida y el error.

La regla de ajuste es:

Si x_i yace en el segmento difuso μ_{ij}

Entonces su correspondiente peso asociado w_{ij}, debe ser incrementado de manera directamente proporcional al error de salida

$$(y - \hat{y}) \qquad (4.8)$$

Esta regla se expresa en términos numéricos con la siguiente expresión:

$$W_{ij}(t+1) = W_{ij}(t) + \alpha_i \mu_{ij}(x_i)(y - \hat{y}) \qquad (4.9)$$

donde α_i es el coeficiente de aprendizaje.

Simulación

Para completar la simulación de la NNDM y su algoritmo de aprendizaje deben considerarse las siguientes etapas:

Paso 0: Inicializar los pesos w_{ij} con valores aleatorios.

Paso 1: Entradas y salidas, formato y generación del conjunto de entradas.

Paso 2: Etapa de sinapsis; conjuntos de aprendizaje y reglas lógicas.

Paso 3: Etapa del soma; salida del sistema.

Paso 4: Cálculo del error, aplicación del criterio de parada.

Paso 5: Ajuste de los pesos sinápticos mediante al algoritmo de aprendizaje.

Paso 6: Repetir desde el paso 1.

Para generar los estímulos primarios de entrada será necesario un conjunto de aprendizaje, consistente de pares asociados de entrada-salida. Este conjunto puede provenir de mediciones hechas en un sistema real, o bien puede construirse a partir de su modelo matemático $M(\bar{x})$. En tal caso deberá tomarse un conjunto finito de entradas $\bar{x}_i$, y construir a partir de ellas los pares entrada-salida, es decir los vectores $(\bar{x}_i, M(\bar{x}_i))$. Para el caso del modelo general propuesto, se usará el mismo modelo utilizado por Yamakawa a fin de contrastar el comportamiento del modelo modificado.

$$y_{n+1} = f_1(x_1) + f_2(x_2) + f_3(x_3)$$

$$= \frac{5y_n}{1+y_n^2} - 0.5y_n - 0.5y_{n-1} + 0.5y_{n-2} \qquad (4.10)$$

con los valores iniciales:

y_1=0.733

y_2=0.234

y_3=0.973

Cada elemento del conjunto de entrenamiento resulta:

$$(\bar{y}_i, y_{i+1})$$

donde el vector de entrada $\bar{y}_i$ se forma con la entrada y_i y sus retardos:

$$\bar{y}_i = (y_i, y_{i+1}, \ldots, y_{i+n})$$

donde n es el numero de retardos.

Se eligió un conjunto de entrenamiento consistente solo de los pares correspondientes a y_0 , y_1 , y_2 , ..., y_{49}. Este conjunto fue obtenido empíricamente por Yamakawa y este trabajo lo utilizará también, para contrastar los resultados de la NNDM con el modelo original.

La etapa del procesamiento sináptico se llevan a cabo obteniendo los valores de las funciones de membresía $\mu_{ij}(x_i)$ y aplicando los función max sobre todos los pares de $\mu_{ij}(x_i)$ y $\mu_{ij+1}(x_i)$ de una sinapsis. La función min se excluye pues los pesos asociados pasan directamente al valor de acción. El valor de acción asociado a cada $\mu_{ij}(x_i)$ pasará a la variable de salida solo en el caso de que el $\mu_{ij}(x_i)$ sometido a la función max coincida con el valor estudiado en ese momento. De esta forma se simulan las evaluaciones que tienen lugar en el AL220.

Este procedimiento se repite para cada sinapsis en el sistema.

Cuando se haya completado el proceso sináptico en todas las sinapsis, los valores de salida de cada sinapsis f_i estos se suman y normalizan.

En seguida se calcula el error

$$\text{error} = y_{n+1} - \hat{y}_{n+1} \qquad (4.11)$$

y se verifica si cumple la condición de parada. Para este caso se elige una condición de error suficientemente pequeño:

$$\sqrt{\overline{error^2}} < criterio \qquad (4.12)$$

donde

$\sqrt{\overline{error^2}}$ es la estimación al cabo de una época de aprendizaje.

Las épocas de aprendizaje varian según el número de sinapsis utilizadas.

criterio es un valor empírico que variará según las sinapsis usadas y la resolución

de la simulación.

Si la condición se cumple el algoritmo se detiene y los resultados se almacenan en un archivo de datos.

Si la condición no se cumple se prosigue con el algoritmo.

El siguiente paso es ajustar los pesos, que en la NNDM son los valores de acción asociados a las reglas. La regla es similar a la expuesta por Yamakawa y se usa la expresión:

$$W_{ij}(t+1) = W_{ij}(t) + \alpha_i \mu_{ij}(x_i)\left(y_{n+1} - \hat{y}_{n+1}\right) \qquad (4.13)$$

con α_i =0.20 para todas las sinapsis, pues estos resultaron, experimentalmente, en una mayor estabilidad y velocidad del algoritmo.

RESUMEN DE RESULTADOS.

Yamakawa ha reportado los mejores resultados para su sistema usando cuatro y cinco sinapsis. Se tomarán sus resultados como referencia para evaluar los primeros resultados de la Neo Neurona Difusa Modificada.

Para el caso de cinco sinapsis la etapa de entrenamiento puede ajustarse fácilmente modificando las variables en el programa, sin embargo la implementación física en el AL220 requerirá del uso de un microcontrolador conectando a la última salida del primer dispositivo.

Se dividirán los resultados de las primeras pruebas en software que se presentaran aquí, en tres rubros que resultan relevantes para la evaluación del sistema; a saber: velocidad relativa de convergencia, precisión de predicción y evaluación de desempeño general.

Velocidad Relativa de Convergencia

La velocidad de convergencia puede medirse en términos absolutos como el número de épocas de aprendizaje necesarias para alcanzar un error de salida deseado. Yamakawa no es especifico en cuanto al valor que toma como criterio, solo habla de un error *razonablemente* pequeño, sin embargo, debe tenerse en cuenta que un error demasiado pequeño resultará en un sistema con poca adaptabilidad hacia la incertidumbre de las mediciones, lo cual no es una característica deseable en sistemas neuronales o difusos. De modo que en la expresión (4.12) la variable *criterio* para detener el aprendizaje el proceso de aprendizaje, en estas primeras pruebas toma el valor: 0.125. Este valor permite observar el proceso de convergencia del aprendizaje en un tiempo relativamente corto de simulación.

La información de la velocidad debe contrastarse con los parámetros que la afectan, el más importante, debido a que se puede monitorear y controlar fácilmente es el coeficiente de aprendizaje. Los ciclos de aprendizaje usan 45 conjuntos de cinco datos (5 sinapsis), 100 ciclos de aprendizaje constituyen una *época de aprendizaje*, luego de cada época de aprendizaje se calcula el error. Las siguientes gráficas muestran los resultados obtenidos.

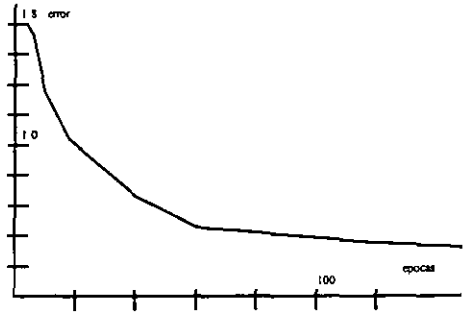

FIGURA 4.5 Relación del error de salida del sistema al número de epocas de entrenamiento para un coeficiente de aprendizaje de 0.10.

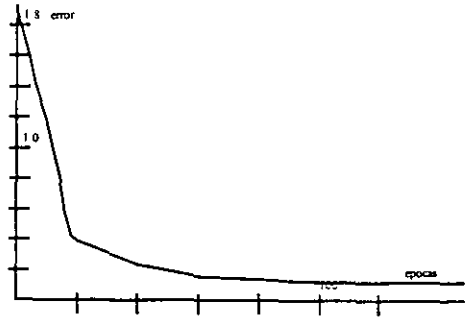

FIGURA 4.6 Relación del error de salida del sistema al número de epocas de entrenamiento para un coeficiente de aprendizaje de 0.20.

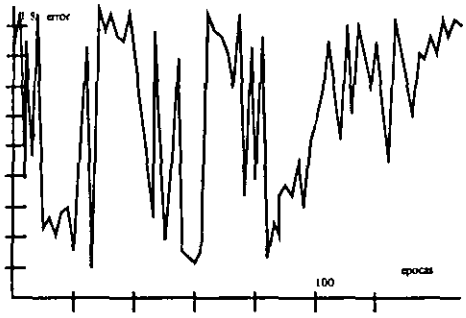

FIGURA 4.7 Relación del error de salida del sistema al número de epocas de entrenamiento para un coeficiente de aprendizaje de 0.30.

Como puede observarse, la velocidad de convergencia es sensible a las variaciones del coeficiente de aprendizaje. A partir de un coeficiente mayor a 0.3 el sistema no converge hacia un determinado error, es decir deja de aprender y se torna inestable. Esto contrasta con los resultados de Yamakawa, el cual mantiene una convergencia aún con un coeficiente de 0.56.

Precisión de Predicción.

La Neo Neurona Difusa Modificada de cinco sinapsis, fue entrenada para predecir secuencialmente el comportamiento del modelo que se representa en la expresión (4.10).

Los resultados encontrados para el sistema entrenado con un coeficiente aprendizaje de 0.2 es el siguiente:

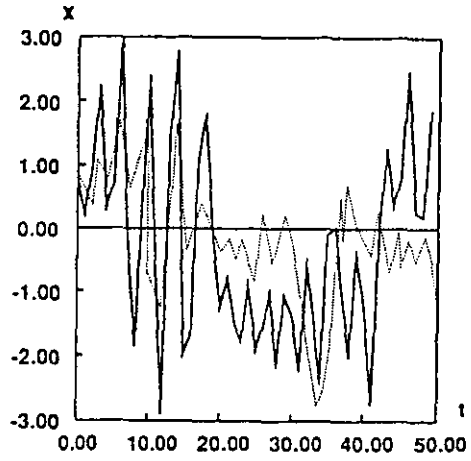

FIGURA 4.8 Predicción secuencial del comportamiento caotico de la función en la equación (4.10), usando una NNDM con 5 sinapsis. La linea sólida representa la función, mientras que la punteada corresponde a la función de predicción de la NNDM.

Los resultados de Yamakawa pueden hacer predicciones solo hasta los 25 primeros valores discretos del sistema, con un error *razonable* (según palabras de Yamakawa). Este sistema puede mantenerse cerca del comportamiento esperado hasta los 30 primeros valores, pero en cambio el aprendizaje es mas lento y el error es mayor aunque consistente (es decir el comportamiento de la NNDM es similar (*imita*) al del sistema original).

Evaluación de Desempeño general.

Usando los resultados anteriores puede compararse este sistema con el sistema original de Yamakawa. Para ello es necesario diseñar un criterio que resuma las diferencias de comportamiento de los sistemas, considerando las principales caracteristicas que comparten los sistemas, de modo que el resultado refleje los efectos de las caracteristicas modificadas.

Criterios de este tipo pueden ser las razones de las velocidades relativas y la precisiónes de predicción.

Se define la *precisión de predicción* como:

1-error de predicción.

Una pendiente 0 en el valor 1 representará un comportamiento de la NNDM

indistinguible del comportamiento del sistema de Yamakawa, mientras que pendientes mayores y valores menores que 1 representan menor desempeño con relación al sistema original.

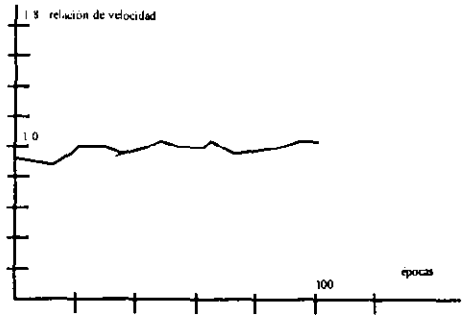

FIGURA 4.9 Razón de la velocidade relativa de la NNDM a la de la Neo Neurona de original.

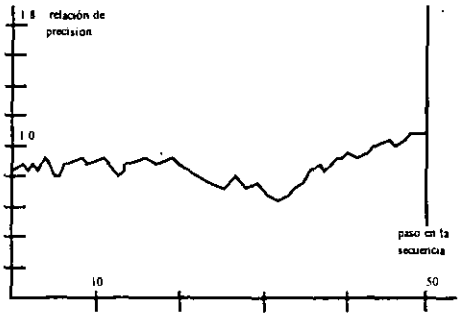

FIGURA 4.10 Razon de la precisiónes de predicción de la NNDM a la de la Neo Neurona de original.

Puede observarse que el desempeño de la NNDM es en general mas pobre que el sistema original, sin embargo debe tomarse en cuenta que los resultado obtenidos se obtuvieron con un diseño que considera las limitaciones del microcontrolador AL220, las cuales pueden considerarse como un compromiso con su bajo costo.

Esto concluye la propuesta de la Neo Neurona Difusa Modificada y con ella la tarea propuesta para este trabajo, una descripción de los procedimientos de prueba en planta excede el marco de este trabajo, pues la intención aquí es comenzar el estudio de los sistemas de control neuro-difuso y proponer la primera etapa de esta propuesta original. Solo resta concluir con algunas puntualizaciones propicias para comprender a este sistema en el marco de los sistemas Neuro-Difusos.

CONCLUSIONES

Se ha hecho aquí un recuento de las teorias que sustentan dos modelos alternativos a los modelos clásicos de control. También se presentaron los modelos híbridos que resultan de hacer fusiones entre los modelos neuronal y difuso, en el marco de una teoria mas amplia: las redes adaptivas o redes adaptativas. Por último se ha propuesto una nueva estructura neuro-difusa no disponible en la literatura existente, que se ha denominado Neo Neurona Difusa Modificada. Se analizó con algún detalle las discrepancias y coincidencias de este modelos con los modelos simples de sistema de control difuso y las estructuras neuronales. De particular importancia resulta que este nuevo modelo plantea un tipo de procesamiento no lineal que establece una diferencia con los modelos corrientes en las redes neuronales y la redes adaptativas canónicas, consistente en que el procesamiento de la señal es sobre todo un procesamiento que ocurre en las sinapsis del sistema. El objetivo de este trabajo ha sido presentar una base teórica (que puede servir también como material de consulta) para los sistemas neuro-difusos y hacer el planteamiento de un nuevo sistema. En el capitulo cuatro se ha visto como se puede implementar un sistema neuro-difuso que, debido a su procesamiento sinaptico, puede aprovechar recursos de hardware difuso como el AL220. Se presentó también el principio de una justificación matemática de la incertidumbre para este sistema. El implementar estos sistemas no es tan costoso como el diseño de un procesador neuronal complicado. Desde luego que el sistema posee algunas limitaciones, al ser pensado para procesadores de 8 bits y con procesamiento difuso no se pretende que puedan implementar funciones con una exactitud arbitraria. Su propósito es, sobre todo, lograr un procesamiento inteligente a bajo costo.

Aún así, el esquema puede fácilmente adaptarse a otros tipos de dispositivos de hardware, que pueden ser desde procesadores puramente neuronales hasta diseños VLSI dedicados.

Hay todavía muchos puntos por discutir, pues el modelo presentado aquí es la propuesta a nivel teórico y actualmente se sigue trabajando sobre los fundamentos teóricos del modelo y en las pruebas tanto en simulación como en planta. Estas no se presenta aqui porque caen fuera de lo límites trazados para este trabajo. Sin embargo no se han dejado simplemente de lado, sino que se prosigue la investigación.

Además de problemas de índole técnica, los sistemas neuro-difuso, el computo suave y el área de inteligencia y vida artificial plantean problemas de orden científico y hasta filosófico.

En cuanto al área de lenguajes formales, por ejemplo, los sistemas neuronales y la teoría de conjuntos difusos plantean modelos de representación que no necesariamente pueden clasificarse como relaciones "semánticas".

Así mismo la teoría de la representación hoy en día tiene que lidiar con aspectos complicados de construcción de mapas en sistemas autónomos de navegación y en

general de sistemas de memoria heteroasociativa.

Tópicos como estos son también temas complicados e interesantes que no se han podido discutir en este texto, pero que no se deben abandonar. Un deseo implícito es que este trabajo, además de exponer temas técnicos de áreas muy interesantes como sistemas neuro-difuso y de hacer una propuesta propia, tenga como efecto lateral el despertar el interés en la investigación desde cualquier punto de vista sea técnico, científico o filosófico.

APÉNDICE I: LAS RNA COMO APROXIMADORES UNIVERSALES

Un uso común de las RNA, y que se explota en los neurocontroladores, es aproximar un mapeo continuo f . Existen mapeos muy simples que una red de capa sencilla no puede representar, pero una red multicapa puede hacer mapeos muy complicados. La validez de esta proposición se basa el teorema de existencia del mapeo para redes neuronales de Kolmogrov; el cual afirma que una red neuronal de alimentación hacia adelante con tres capas de neuronas (unidades de entrada, unidades ocultas y unidades de salida) puede representar cualquier función continua exactamente. Las siguiente exposición de los teoremas Kolmogorov y Specher se basan en la presentación hecha por Funahashi en 1989. El teorema de Hecht-Nielsen, difunde el teorema de Sprecher en la terminología de redes neuronales.

Teorema de Kolmogorov

Cualquier función continua $f(x_1, ..., x_n)$ de varias variables definida en I^n con $n \geq 2$, donde $I = [0,1]$, puede ser representada en la forma:

$$f(x) = \sum_{j=1}^{2n+1} \chi_j \left(\sum_{j=1}^{n} \psi_{ij}(x_i) \right)$$

donde χ_j y φ_j son funciones continuas de una variable y ψ_{ij} son funciones monótonas que no dependen de f.

Teorema de Sprecher

Para cada entero $n \geq 2$, existe una función real, monótonamente creciente $\psi(x)$. ψ: $[0,1] \rightarrow [0,1]$, dependiendo de n y teniendo la siguiente propiedad: Para cada número preasignado $\delta > 0$, existe un número racional $\varepsilon > 0$, $\varepsilon > \delta > 0$, tal que toda función real y continua de n variables, f(x), definida en I^n, puede ser representada como:

$$f(x) = \sum_{j=1}^{2n+1} \chi \left(\sum_{i=1}^{n} \lambda \psi \left(x_i + \varepsilon(j-1) + j - 1 \right) \right)$$

donde la función χ es real y continua, y λ es una constante independiente de f.

Teorema de Hecht-Nielsen

Dada una función continua $f : I^n \rightarrow R^m$, donde I es el intervalo cerrado [0,1] , f puede ser representado exactamente por una red neuronal alimentada hacia adelante, teniendo n unidades de entrada, 2n+1 unidades ocultas y m unidades de salida.

Las unidades de entrada reparten la señal de entrada a las unidades ocultas. La función de activación de la j-ésima neurona oculta es $z_j = \left(\sum_{i=1}^{n} \lambda \psi \left(x_i + \varepsilon(j-1) + j \right) \right)$, donde la constante real λ y la función continua y monótonamente creciente ψ, son

independientes de f (aunque dependen de n) y la constante ε satisface las condiciones del teorema de Sprecher. La función de activación para las unidades de salida es: $y_k = \sum_{j=1}^{2n+1} g_k z_j$, donde la función g_k es real , continua y dependiente de f y de ε.

Hornik, Stichcombe y White, extendieron los resultado precedentes, añadiendo que la función de activación de al menos una de las unidades depende de la función que está siendo aproximada, para mostrar que las redes multicapa alimentadas hacia adelante con funciones arbitrariamente *apretadas*, pueden aproximar casi cualquier función de interés (específicamente, cualquier función mensurable, de un espacio de dimensión finita a otro espacio de dimensión finita). Una función *apretada*; es una función no decreciente f(x) tal que $0 \leq f(x) \leq 1$ para toda x, $f(x) \to 0$ cuando $x \to -\infty$ y $f(x) \to 1$ cuando $x \to \infty$. Estos resultados requieren un número suficientemente grande de unidades ocultas, pero los autores no manifiestan el número de unidades necesarias (que a la fecha es un tema de investigación y discusión). Los mismos autores, en 1990, mostraron que con adecuadas suposiciones y un poco de trabajo adicional, una red neuronal puede aproximar tanto una función como su derivada o su derivada generalizada. Esto ha sido útil para aplicaciones tales como el aprendizaje de movimientos suaves en robots.

Se ha mostrado, también que los pesos necesarios para lograr una aproximación pueden ser aprendidos, es decir que la probabilidad de que el error de la red excediendo cualquier nivel especificado, tiende a cero conforme el tamaño del conjunto de entrenamiento se incrementa. Además, la complejidad de la red aumenta con el tamaño del conjunto de entrenamiento.

Ha habido gran interés en determinar los tipos de funciones de activación requeridos para asegurar que una red neuronal multicapa pueda aproximar una función arbitraria con una exactitud especifica. Kreinovich [17] ha mostrado que una red neuronal conformada por neuronas lineales y neuronas con una sola función de activación no lineal puede representar cualquier función hasta una precisión especificada diferente de cero. Sin embargo, el asume un número ilimitado de capas ocultas.

Geva y Sitte [19] han demostrado un método constructivo para aproximar funciones de variables múltiples usando redes neuronales multicapa. Lo hacen combinando dos funciones sigmoides (funciones en forma de s) para producir una función de activación que es similar a una Función de potencial gaussiana, *Radial Basis Function* (que es un modelo en que las neuronas responden solo a información local, proveniente de una vecindad de radio determinado).

La facultad de aproximar una función arbitraria, le da a las RNA, en específico a los perceptrones multicapa, un enorme potencial de aplicación, en teoría de control se cuentan con métodos similares (las diversas transformaciones que se usan para representar una función), pero en general requieren de un conocimiento a priori; la ventaja de una red neuronal, estriba en que un modelo no es necesario, bastan ejemplos, y pueden, así mismo, usarse para obtener dicho modelo. La cualidad de aproximador universal la comparten los sistemas difusos.

APÉNDICE II: LOS SISTEMAS DE INFERENCIA DIFUSOS COMO APROXIMADORES UNIVERSALES

Se ha mencionado ya que debido a la incipiencia de estos modelos de control, no existe una notación ni denominación normalizada para el campo de la lógica difusa, Castro utiliza, como se vió en el capitulo 3, una designación un tanto divergente de la corriente en este trabajo, pero por respeto a su trabajo se reproducirá aquí sin hacer cambios sustanciales.

Se consideran solo los sistemas tipo MISO (entradas múltiples y salida única, por sus iniciales en inglés *Multiple Input Single Output*) ya que un sistema de salidas múltiples puede descomponerse en una colección de sistemas de salida única, es decir sistemas que poseen funciones

$U \subseteq R^n \rightarrow R$

La base de reglas es un conjunto constituido por declaraciones lingüísticas de la forma:

R_j: Si x_1 es A_1^j y ... y x_n es A_n^j. Entonces y es B^j .....

donde

x_i $(i=1...n)$ son las entradas de la base de reglas

y es la salida el sistema

A_i^j y B^j $(j = 1...k)$ son los términos lingüísticos

k es el número de reglas difusas.

relacionando cada término lingüístico con una función de membresía, se especifica el significado de la regla.

En la *interfaz de fuzzificación* el considera dos tipos principales

Fuzzificación de Punto (*Point fuzzification*):

$$F(x) = \begin{cases} 1 & \text{Si } x = x_0 \\ 0 & \text{En otro caso} \end{cases}$$

Fuzzificación Aproximada (*Approximate fuzzification*): $F(x) \neq 0$ si y solo si $|x - x_0| < \delta$ (este tipo es el que se expuso en el capitulo 1)

La lógica usada por la máquina de inferencias (separada de la base de reglas, de acuerdo a Castro). La regla general de inferencia de una regla es del tipo

SI X es A entonces Y es B,

que no es mas que el Modus Ponens Generalizado, expuesto en el capitulo uno:

Si X es A entonces Y es B

X es A' (antecedente)

Y es B' (consecuente)

y la relación esta dada por:

$$B'(y) = \sup_x T'\left(A'(x), I(A(x), B(y))\right)$$

sup es una función de máximo

T' es un función t-norma

I es una función de implicación

de esto depende B'(y)

Se utiliza una conjunción difusa para reglas de entrada múltiple:

Si X_1 es A_1 y X_2 es A_2 y ... X_n es A_n , entonces Y es B

X es A' (antecedente)

Y es B' (consecuente)

donde

$A(\bar{x}) = T'\left(A_1(x_1), A_2(x_2), \ldots, A_n(x_n)\right)$ depende de una t-notma T.

Los métodos de defuzzificación son los generales expuestos en el primer capitulo. Castro define a los métodos de defuzzificación en general como un mapeo de los subconjuntos difusos en V en V, siendo este último el universo de la variable de salida y.

Asi que se puede asociar una función a cada SCD como sigue:

Fuzzificación: Un conjunto difuso $A' = Fuzzy(x_0)$ se asocia a al entrada x_0 .

Inferencia difusa: Una salida aproximada se obtiene por razonamiento difuso (reglas y relaciones definidas por la teoría de los conjuntos difusos). B' = inferencia difusa de A'

Defuzzificación: El valor de salida se obtiene a partir de la salida aproximada (que no necesariamente es difusa, dependiendo de las relaciones y funciones que se use en la inferencia):

$$y_0 = Defuzz(B')$$

Castro expone su resultado así:

> *Considérese un tipo de SCD; un método de fuzzificación, un método de inferencias difusas, un método de defuzzificación y una clase de reglas difusas RUL, siendo todo fijo y organizado. Dada una función real arbitraria* f *en un compacto* $U \subset R^n$ *en cierto* $\varepsilon > 0$, *¿es posible encontrar un conjunto de reglas difusas en RUL tal que el SCD asociado aproxime a f hasta un nivel* ε *?*

Castro responde positivamente y arguye que la aproximación es posible para casi todo tipo de SCD, para ello presenta dos casos:

1. SCD con consecuentes (en las reglas) difusos.

2. SCD con consecuentes no difusos.

Este trabajo se ha enfocado a los SCD puramente difusos, así que solo se muestra la prueba para el caso 1; la prueba del caso 2 se puede consultar en Castro 1995.

Castro procede a demostrar así:

Para cada $a < b \in R$ *sea* $\mu(a,b): R \to R$ *una función de membresía tal que* $\mu(a,b)(x) \neq 0$ *si y solo si* $x \notin (a,b)$. *Sean T y T' dos t-normas, I una implicación difusa y T' una t-conorma.*

Los SCD con consecuentes difusos son aproximadores universales.

Sea $S_1 = S_1(T', T, I, T^*, \mu(a,b))$ *la familia de todos los SCD donde:*

I) El método de fuzzificación es la fuzzificación de punto.

II) La base de reglas se compone de un número finito de reglas de la forma:

Si X_1 *es* A_1 *y* X_2 *es* A_2 *y ...* X_n *es* A_n *, entonces Y es B*

donde la función de membresía para cada A_{ij} *es* $\mu(a_{ij}^1, a_{ij}^2)$

para algún $a_{ij}^1 < a_{ij}^2 \in R$, *es decir.*

$A_{ij}(x_i)=(a_{ij}^1, a_{ij}^2)(x_i)$

y la función de membresía para cada B_j *es* $\mu(b_{ij}^1, b_{ij}^2)$ *para algún* $b_{ij}^1 < b_{ij}^2 \in R$, *es decir:*

$B_j(y)= \mu(b_j^1, b_j^2)(y)$

III) La inferencia difusa se hace con:

-T como conjunción difusa.

-El Modus Ponens Generalizado construido con T' e I.

-La combinación de todas las inferencias difusas se hace por medio de T.*

Así, la inferencia difusa se hace con:

a) Una regla

R_j: *Si* x_1 *es* A_{1j} *y ...* x_n *es* A_{nj} *, entonces Y es B'*

será aplicad si y solo si la entrada $\bar{x} = \bar{x}^0$ *coincide con el antecedente es decir, si y solo si* $A_j(\bar{x}^0) \neq 0$ *, siendo* $A_j(\bar{x}) = T(A_{1j}(x_1), A_{2j}(x_2), \ldots, A_{nj}(x_n))$.

b) si la entrada $\bar{x}^0$ *coincide con el antecedente, la inferencia es*

x_1 *es* A_1 *y* x_2 *es* A_2 *y . . .* x_n *es* A_n, *entonces y es B*

$\bar{x}$ *es A'*

y es B'

$$B'(y) = \max\left\{T'\left(A'(\bar{x}), I\left(A(\bar{x}), B(y)\right)\right) / \bar{x} \in R^n\right\}$$

$$A(\bar{x}) = \max\left\{T'\left(A'(\bar{x}), I\left(A(\bar{x}), B(y)\right)\right) / \bar{x} \in R^n\right\},$$

y como la entrada es un punto (vector) $\bar{x} = \bar{x}^0$

$$A'(x) = \begin{cases} 1 & \text{si } \bar{x} = \bar{x}^0 \\ 0 & \text{en otro caso} \end{cases}$$

el resultado será troducido en:

$$B'(y) = T'\left(1, I\left(A(\bar{x}^0), B(y)\right)\right) = I\left(A(\bar{x}^0), B(y)\right)$$

c) En general, puede expresarse la inferencia de la regla R_j *cuando la entrada es* $\bar{x} = \bar{x}^0$, *por:*

$$B'_j(y) = \begin{cases} 0 & \text{si } A_j(\bar{x}^0) = 0 \\ I\left(A_j(\bar{x}^0), B_j(y)\right) & \text{en cualquier otro caso} \end{cases}$$

la cual en el caso de una t-norma, la implicación es la expresión general

$$B'_j(y) = I\left(A_j(\bar{x}^0), B_j(y)\right)$$

d) La combinación de todas las inferencias difusas se hace por medio de T:*

$$B'(y) = T^*\left(\left\{B'_j(y)\right\}_j\right)$$

IV)El método de defuzzificación verifica la propiedad de producir un punto en el soporte del conjunto difuso original,

$$y^0 = S(\bar{x}^0) \in \text{Soporte}(B')$$

donde $S \in S_1$ *es el mapeo global de entrada-salida.*

Los únicos parámetros no especificados en esta clase son el número de reglas k, y los coeficientes determinantes de las funciones de membresía, $a_{ij}^1, a_{ij}^2, b_{ij}^1, b_{ij}^2 \quad (i = 1,\ldots,n \ , j = 1,\ldots,k)$.

TEOREMA 1

Sea $f: U \subseteq R^n \to R$ *una función continua definida en un compacto U. Si I(a,0)=0 siendo* $a \neq 0$, *entonces para cada* $\varepsilon > 0$ *existe un* $S_\varepsilon \in S_1$ *tal que*

$$\max\left\{\left|f(\bar{x}) - S_\varepsilon(\bar{x})\right| / x \in U\right\} \leq \varepsilon$$

Lema 1

Bajo las condiciones del Lema 1 existe un $S_\varepsilon \in S_1$ *tal que*

APENDICE III: CÓDIGO FUENTE PARA LA SIMULACIÓN DE LA NNDM,Y SU ETAPA DE APRENDIZAJE

El Codigo para simular y entrenar la neo neurona difusa de 5 sinapsis escrito para lenguaje C consiste basicamente de las siguientes lineas.

```
01.c [1]
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
/* VARIABLES POR UTILIZAR */
int  i, j, epoca, ej;
float s[50], x[5], m[5][12], w[5][12], wg[5], sigma, y, yd, alfa[5], e[50];
float ecm, alfa[5], alto;
main()
{
        printf("INICIALIZACION DE LAS VARIABLES \n" );
        printf("LLAMADA A inicio() \n");
inicio();
epoca=1;
        printf(" epoca=1  \n");
        printf("APRENDIZAJE POR EPOCAS, EN CICLO INFINITO
HASTA LA CONVERGENCIA  \n");
    do
    {
    printf("COMIENZA AHORA LA EPOCA NUMERO \n");
    printf("NUMERO %d \n", epoca);
        printf(" LLAMADA A epoc()  \n");
            for(ej=0; ej<50; ej++) {epoc();}
    e_c_m();
```

[1] Las marcas .C corresponden a los archivos que contienen las diferentes funciones llamadas por el programa.

```
                epoca++;
                }
                 while (ecm>alto);

                        printf("LLAMADA A LA RUTINA DE ESCRITURA EN DISCO
\n");
                        printf("escribir() \n");
        escribir();
        }
        /* termina main  */
        aleator.c
        #include <stdio.h>
        #include <stdlib.h>
        #include <time.h>
        /* VARIABLES POR UTILIZAR */
        extern int  i, j, epoca, ej;
        extern float s[50], x[5], m[5][12], w[5][12], wg[5], sigma, y, yd, alfa[5], e[50];
        extern float ecm, alfa[5], alto;
        /* GENERAR LOS PESOS ALEATORIoS */
        aleatoreo()
        {
        float menor, mayor;
        printf("ESTA ES LA RUTINA DE GENERACION INICIAL DE PESOS
aleatoreo() \n");
        randomize();
        printf("RANDOMIZE() \n");
        menor=mayor=128;
        for(i=0; i<5; i++)
                {
                for(j=0; j<12; j++)
                        {
                        w[i][j]=rand();
                        w[i][j]=w[i][j] / 32768;
                        w[i][j]=w[i][j] * 255;
```

```
                    if(w[i][j]<1){ w[i][j]= 1; }
                    printf("W %d %d %s %f", i, j, "=", w[i][j]);
                        if(w[i][j]<menor) {menor=w[i][j]; }
                        if(w[i][j]>mayor) {mayor=w[i][j]; }
                    }
                }
        printf(" ha terminado el ajuste a los intervalo de 0 a 255 \n");
        printf(" menor valor es %f \n", menor);
        printf(" mayor valor es %f \n", mayor);
        printf(" ha terminado el ajuste a los intervalo de 0 a 255 \n");
        printf("FIN DE LA FUNCION ALEATOREO () \n");
        }
        e_c_m.c
        #include <stdio.h>
        #include <stdlib.h>
        #include <time.h>
        /* VARIABLES POR UTILIZAR */
        extern int  i, j, epoca, ej;
        extern float s[50], x[5], m[5][12], w[5][12], wg[5], sigma, y, yd, alfa[5], e[50];
        extern float ecm, alfa[5], alto;
        /* LLAMADA AL CALCULO DEL ERROR CUADRATICO MEDIO PARA LA
EPOCA PRESENTE */
        /* FUNCION DE GENERACION DEL ERROR CUADRATICO MEDIO */
        e_c_m()
        {
         printf("FUNCION DE GENERACION DEL ERROR CUADRATICO MEDIO
\n");
         printf("PRINCIPIO DE LA FUNCION e_c_m() \n");
        ecm=0;
        for(ej=0; ej<50; ej++) {ecm = ecm + (e[ej] * e[ej]);  printf(" ecm= %f", ecm); }
        ecm = ecm / 2;
        printf(" ecm= %f  \n", ecm);
         printf(" FIN DE LA FUNCION e_c_m() \n");
        }
```

```
/* FIN DE LA GENERACION DEL ERROR CUADRATICO MEDIO */
epoc--.c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
/* VARIABLES POR UTILIZAR */
extern int  i, j, epoca, ej;
extern float s[50], x[5], m[5][12], w[5][12], wg[5], sigma, y, yd, alfa[5], e[50];
extern float ecm, alfa[5], alto;
epoc()
{
                    printf("COMIENZA AHORA EL EJEMPLO NUMERO
%d \n", ej);
          vector();     printf("vector() ACTUALMENTE CORRE EL
EJEMPLO NUMERO %d \n", ej);
          fuzzi(); printf("fuzzi() ACTUALMENTE CORRE EL EJEMPLO
NUMERO %d \n", ej);
                    printf("y_err()   ACTUALMENTE   CORRE   EL
EJEMPLO NUMERO %d \n", ej);
                    printf("ajuste()  FINALIZA  AHORA  EL  EJEMPLO
NUMERO %d \n", ej);
}
escrib.c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
/* VARIABLES POR UTILIZAR */
extern int  i, j, epoca, ej;
extern float s[50], x[5], m[5][12], w[5][12], wg[5], sigma, y, yd, alfa[5], e[50];
extern float ecm, alfa[5], alto;
/* RUTINA DE ESCRITURA EN DISCO DE LA MATRIZ DE PESOS */
escribir()
{
FILE *fp;
printf("principio DE LA FUNCION escribir() \n");
```

```
fp = fopen("pesos.dat", "wt");
for(i=0; i<5; i++)
        {
        for(j=0; j<12; j++)
                {
                fprintf(fp, "%s %d %d %3d %s ", "w", i, j, w[i][j], " ");
                }
        fprintf(fp," %s \n", "--");
        }
fclose(fp);
 printf("FIN DE LA FUNCION escribir() \n");
}
```

inicio.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
/* VARIABLES POR UTILIZAR */
extern int  i, j, epoca, ej;
extern float s[50], x[5], m[5][12], w[5][12], wg[5], sigma, y, yd, alfa[5], e[50];
extern float ecm, alfa[5], alto;
inicio()
{
alto = 0.2;
for(i=0; i<5; i++) {alfa[i] = 0.2; printf("alfa %d %s %f \n", i, "=", alfa[i]);}
printf("LLAMADA A LA INICIALIZACION DE LA SERIE DE ENTRADAS");
printf("serie() \n");
serie();
printf("LLAMADA A LA INICIALIZACION DE LOS PESOS \n");
printf("aleatoreo() \n");
aleatoreo();
}
```

serie.c

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <time.h>
/* VARIABLES POR UTILIZAR */
extern int  i, j, epoca, ej;
extern float s[50], x[5], m[5][12], w[5][12], wg[5], sigma, y, yd, alfa[5], e[50];
extern float ecm, alfa[5], alto;
/* GENERAR LAS ENTRADAS s[ej] PARA CADA EJEMPLO ej */
serie()
{
float F1, F2, F3, F11, F12, menor, mayor;
int imen, imay;
printf("esta es la función serie \n");
s[0] = 0.733;
s[1] = 0.234;
s[2] = 0.973;
menor=s[1];
mayor=s[2];
printf("s[0] = 0.733  s[1] = 0.234   s[2] = 0.973  \n");

for (ej=2; ej<49; ej++)
        {
/* ecuacion */
F11= (s[ej] * 5)/(1+(s[ej] * s[ej]));
F12= s[ej] * 0.5;
F1= F11 - F12;
F2= s[ej-1] * 0.5;
F3= s[ej-2] * 0.5;
s[ej+1]= F1 - F2 + F3;
/*termina ecuacion */
        printf("s %d %s %f %s", ej+1,"=", s[ej+1],"    ");
        if(s[ej+1]<menor){ menor=s[ej+1]; imen=ej+1;}
        if(s[ej+1]>mayor) {mayor=s[ej+1]; imay=ej+1;}
        }
printf(" ha terminado la generacion de la serie de acuero ala formula \n");
        printf(" menor valor es s[ %d %s %f \n", imen,"]=", menor),
```

```
        printf(" mayor valor es s[ %d %s %f \n", imay,"]=", mayor);
/* ajuste a un intervalo de 0 a 255 */
printf(" ajuste a un intervalo de 0 a 255 \n");
menor=128;
for (ej=0; ej<50; ej++)
        {
        s[ej] =( (s[ej] + 3)/6 ) * 255;
        printf("s[%d%s%f%s", ej,"]=", s[ej],"  ");
        if(s[ej]<menor) {menor=s[ej]; imen=ej;}
        if(s[ej]>mayor) {mayor=s[ej]; imay=ej;}
        }
printf(" ha terminado el ajuste a los intervalo de 0 a 255 \n");
printf(" menor valor es s[ %d %s %f \n", imen,"]=", menor);
printf(" mayor valor es s[ %d %s %f \n", imay,"]=", mayor);
printf(" ha terminado el ajuste a los intervalo de 0 a 255 \n");
/* la serie de entradas esta dada en s[ej] con ej de 0 a 49 */
printf("la serie de entradas esta dada en s[ej] con ej de 0 a 49   \n");
printf(" FIN DE LA FUNCIÓN SERIE() \n");
}
```

vector.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
/* VARIABLES POR UTILIZAR */
extern int  i, j, epoca, ej;
extern float s[50], x[5], m[5][12], w[5][12], wg[5], sigma, y, yd, alfa[5], e[50];
extern float ecm, alfa[5], alto;
/* GENERAR LA ENTRADA EN LA SINAPSIS UNO Y SUS RETARDOS
PARA EL EJEMPLO ACTUAL ej */
/* SE PRECISA DE UN CICLO FOR PARA LAS 5 SINAPSIS */
vector()
{
printf("PRINCIPIA LA FUNCIÓN retardos() \n");
```

```
        for (i=0; i<5; i++)
                {x[i] = s[ej - i]; printf("x %d %s %f \n", i, "=", x[i]);}
    printf("FIN DE ACTUALIZACION DE ENTRADAS PARA CADA EJEMPLO
ej \n");
    printf("GENERACION DE LA SALIDA DESEADA yd \n");
    yd = s[ej+1];
    printf(" yd= %f  \n", yd);
    }
    /* GENERACION DEL VETOR OBJETIVO CONCLUIDA */
    FUZZIFI1.c
    /* EVALUACION DE x EN CADA CONJUNTO LOS PARAMETROS DE LA
RECTA SON m=-1/23  b=VFINAL(1/23) */
    #include <stdio.h>
    #include <stdlib.h>
    #include <time.h>
    /* VARIABLES POR UTILIZAR */
    extern int  i, j, epoca, ej;
    extern float s[50], x[5], m[5][12], w[5][12], wg[5], sigma, y, yd, alfa[5], e[50];
    extern float ecm, alfa[5], alto;
    /* intervalo=23 y sobran tres */
    /* nota : para cada else hay que ajustar los intervalos y la ecuacion de la
recta */
    fuzzi()
    {
    float INI, FIN;  printf(" PRINCIPIA LA FUNCION fuzzi() \n");

    for(i=0; i<5; i++)
    {
        printf("PRIMERO SE INICIALIZAN LOS m a cero \n");
        for (j=0; j<12; j++) {m[i][j]=0; printf("m %d %d %s %f \n", i, j, "=",
m[i][j]);}
        printf("EVALUACION DE x EN CADA CONJUNTO \n");
        if(x[i] >=0 && x[i] <=1) {m[i][j]=1; m[i][j+1]=0;  printf("m[ %d %s %d
%s %f \n", i, "][", j, "]=", m[i][j]);  printf("m[ %d %s %d %s %f \n", i, "][", j+1, "]=",
m[i][j+1]);}
                else
```

```
                    {
                    INI=1;
                    for(j=0; j<12; j++)
                            {
                            INI = 1 + (j * 23);
                            FIN = INI + 23;
                            if(x[i] > INI  && x[i] <= FIN)  {m[i][j]= (FIN-x[i])/23 ;
m[i][j+1]=1-m[i][j];  printf("m[ %d %s %d %s %f \n", i, "][", j, "]=", m[i][j]); printf("m[
%d %s %d %s %f \n", i, "][", j+1, "]=", m[i][j+1]);}
                            }
                    }
        if(x[i]>254 && x[i]<=255){m[i][j]=1;  printf("m[ %d %s %d %s %f \n", i, "][", j,
"]=", m[i][j]); }
                printf("TERMINA LA FUNCION  fuzzi() \n");
        }
        }
        /* TERMINA LA ETAPA DE FUZZIFICADO */
```

APÉNDICE IV: PROGRAMACIÓN DE UNA CADENA DE RETARDOS ANALÓGICOS EN EL SISTEMA DE DESARROLLO PARA EL AL220.

La siguiente es una impresión de los parámetros en el sistema de desarrollo *INSiGHT IIe* para llevar a cabo la cadena de retardos propuesta en el capitulo cuarto.

La salida generada por la emulación del AL220 con los anteriores parámetros y una entrada sinoidal recortada son los siguientes:

La salida generada por la emulación del AL220 con los anteriores parámetros y una entrada con armónicos sinoidales son los siguientes:

GLOSARIO

Axón	Estructura de la célula nerviosa en forma de larga prolongación, donde se reciben las señales provenientes de otras células y transmitidas por medio de las sinapsis.
Control Inteligente	La convergencia en la práctica y el estudio con fines del modelado y control de procesos y sistemas de la teoría del control, la investigación de operaciones y la inteligencia artificial.
Defuzzificación	En los sistemas de control difuso, es el proceso mediante el cual los resultados de las inferencia que tienen formas topológicas complejas son transformadas a números reales con valores concretos, de modo que puedan ser tratadas como señales de salida.
Dendrita	En las neuronas artificiales, se trata de la estructura que se prolonga hacia el exterior en forma de ramificaciones; su función es transmitir hacia otras células las señales generadas en el cuerpo de la neurona.
Función de Membresía	Una relación que asocia un conjunto de cardinalidad infinita con los grados de verdad de un sentencia.
Fuzzificación	En un sistema de control difuso es la transformación de los valores concretos de las variables de entrada en formas vectoriales que representan la forma lingüística de las mismas variables.
Incertidumbre	En el sentido técnico en la práctica científica (que es el usado en este trabajo), es el rango ponderado de error de una variable.
Lógica	Sistema de símbolos (lenguaje), junto con ciertos axiomas y reglas de inferencia, que permiten construir relaciones y argumentos correctos a modo de preservar relaciones de verdad entre términos y oraciones que ocurren bajo determinadas formas en el lenguaje.
Lógica Difusa	Conjunto de axiomas y reglas de inferencia que permiten la construcción de sistemas simbólicos de inferencia que integran vaguedad y valores graduales de verdad para sus sentencias. La sintaxis y la inferencia de la lógica difusa es similar a la de la lógica simbólica de primer orden, pero sus términos y sus inferencias incorporan incertidumbres; articuladas en la forma de las funciones de Membresía de cada termino y en los métodos de combinación en los argumentos.
Memoria Autoasociativa	Un tipo de memoria que asocia varios patrones consigo mismos.
Memoria heteroasociativa	Un tipo de memoria que asocia patrones de diversa índole.
Paradigma	Palabra griega y latina para denominar un modelo o un patrón ejemplar. La utilización amplia del termino ha sido promovida por el trabajo de Tomas S. Kuhn, en este autor la palabra tiene una acepción de "marco conceptual" o de modo general de resolución de problemas.
Soma	Cuerpo de la neurona, allí se suman las diversas señales entrantes y se emite una señal de salida si tal suma alcanza cierto umbral.

BIBLIOGRAFÍA

CAPITULO 1

[1] Mohamad Jamshidi, Nader Vadee, Timothy J. Ross, editores. *Fuzzy Logic and Control. Software and Hardware Applications.* Prentice Hall; New Jersey 1993.

[2] Stephen Pollard. *Philosophical Introduction to Set Theory.* University of Notre Dame Press. 1990.

[3] George J. Klir, Bo Yuan. *Fuzzy Sets and Fuzzy Logic. Theory and Applications.* Prentice Hall. 1995.

[4] Wang, Li-Xi, *Adaptive Fuzzy Systems and Control.* PTR Prentice Hall, 1994.

[5] Robinson, G.; *Computers, Minds and Robots.* Temple University Press. 1992.

[6] Stephen Pollard. *Philosophical Introduction to Set Theory.* University of Notre Dame Press. 1990.

[7] Turksen, I.B., Dubois D., Prade, H., Yager, R.R., Guest Editors; *Fuzzy Sets and Systems,* Special Memorial Volume-First Issue: Foundations of Fuzzy Reasoning; North-Holland-Amsterdam 1991.

[8] Kandel, Abraham, Editor; *Fuzzy Expert Systems.* CRC Press, Florida 1992.

[10] Li, Han-Xiong, Gatland, H. B. *Conventional Fuzzy Control and Its Enhancement.* IEEE Transactions on Systems, Man, and Cybernetics. Part b: Cybernetics, Vol 26. No 5. October 1996. pp 791-797.

CAPITULO 2

[11] Fuasett, Laurene V. ; *Fundamentals of Neural Networks: Architectures, Algorithms and Applications.* Florida Institute of Technology, Prentice Hall, New Jersey; 1994.

[12] Arbib, Michael A.; *The Handbook of Brain Theory and Neural Networks.* Cambridge, Mass. MIT.; 1995.

[13] Hecht-Nielsen, Robert; *Neurocomputing*; Addison-Wesley Publishing Company, San Diego Ca; 1991.

[14] Mehra, Pankaj and Wah, Benjamin W. ; *Artificial Neural Networks: Concepts and Theory* ; IEEE Computer Society Press; Los Alamitos Cal. 1997.

[15] Page, G. F., Gomm, J. B., Williams, D. ; *Application of Neural Networks to Modelling and Control* ; Chapman and Hall, London 1993.

[16] Gupta, Madan M., Rao, Dandina H.; *Neuro-Control Systems: Theory and Applications* ; IEEE Press, New York, 1994.

[17] Rao, Vemury V. ; *Artificial Neural Networks: Concepts and Control Applications* ; IEEE Press, Los Alamitos Cal., 1992.

[18] Haykin, Simon. *Neural Networks. A comprehensive Foundation.* Macmillan College Publishing Company, New York 1995.

[19] Cui, Xiangzhhong, Shin, Kang G. *Direct Control and Coordination Using Neural Networks.* IEEE Transactions on Systems, Man, and Cybernetics. Vol 23. No 3. May-Jun 1993. pp 686-697.

CAPITULO 3.

[20] Kosko, Bart. *Neural Networks and Fuzzy Systems.* New Jersey: Prentice-Hall, 1992

[21] Wang, Li-Xi, *Adaptive Fuzzy Systems and Control.* PTR Prentice Hall, 1994.

[22] Wang, L. X.. *Fuzzy systems are Universal Approximators.* Proc. of IEEE Intern. Conf. on Fuzzy Systems. San Diego, EEUU 1992. pp 1162-1170.

[23] Buckley, J.J. . *Sugeno Type Controllers are Universal Controllers.* Fuzzy Sets and Systems, vol. 53, 1993. pp 299-304.

[24] Castro, J. L. *Fuzzy Logic Controllers Are Universal Approximators.* IEEE Transactions on Systems, Man, and Cybernetics. Vol 25. No 4. Abril 1995. pp 629-635.

[25] Jang, J.-S.R.; Sun, Chuen-Tsai ; *Neuro-Fuzzy Modeling and Control* ; Proceedings of the IEEE, Vol. 83 No. 3, March 1995, pp 378 - 406.

[26] Yamkawa, T., Uchino, E., Miki, T., Kusanagi, H.. *A Neo Fuzzy Neuron and Its Applications to System Identification and Prediction of the System Behavior.* Proc. of the 2nd Int. Conf. on Fuzzy Logic and Neural Networks. Iizuca, Japon, Julio 17-22. 1992, pp 477-483.

[27] Uchino, E., Yamakawa, T., Soft Computing Based Signal Prediction , Restoration, and Filtering. En Ruan, Da. Intelligent Hybrid Systems: Fuzzy Logia, Neural Networks, and Genetic Algorithms. SKC-CEN 1997. pp 331-352.

[28] Ruan, Da (Editor). Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks, and Genetic Algorithms. SKC-CEN 1997..

CAPITULO 4

[29] Yamkawa, T., Uchino, E., Miki, T., Kusanagi, H.. *A Neo Fuzzy Neuron and Its Applications to System Identification and Prediction of the System Behavior.* Proc. of the 2nd Int. Conf. on Fuzzy Logic and Neural Networks. Iizuca, Japon, Julio 17-22. 1992 pp 477-483.

[30] AL220 Analog Micro Controler Preliminary Data ;Adaptive Logic Inc.; 1996

[31] INSiGHT Iie Development System for the Adaptive Logic Family of Fuzzy Controllers: User's Guide ; Adaptive Logic Inc.; San Jose Ca. 1996