METODOLOGÍA PARA LA GENERACIÓN AUTOMÁTICA DE REGLAS BORROSAS Y AJUSTE ADAPTATIVO DE FUNCIONES DE PERTENENCIA POR MEDIO DE UNA ARQUITECTURA DE RED NEURONAL NETFUZ 1.0

Presentado a La Maestría en ingeniería Área de informática y Ciencias de La Computación
UNIVERSIDAD INDUSTRIAL DE SANTANDER
Para la obtención del título de magíster en ingeniería

AUTOR
JUAN CARLOS REYES FIGUEROA
Ingeniero de Sistemas
Universidad Industrial de Santander

DIRECTOR
FERNANDO RUIZ DÍAZ
Master of Engineering

MAESTRÍA EN INGENIERÍA
ÁREA DE INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
UNIVERSIDAD INDUSTRIAL DE SANTANDER
BUCARAMANGA, AGOSTO DE 2007

RESUMEN

TÍTULO: METODOLOGÍA PARA LA GENERACIÓN AUTOMÁTICA DE REGLAS BORROSAS Y AJUSTE ADAPTATIVO DE FUNCIONES DE PERTENENCIA POR MEDIO DE UNA ARQUITECTURA DE RED NEURONAL -NETFUZ 1.0- *

AUTOR: REYES FIGUEROA Juan Carlos**

PALABRAS CLAVE: Red Neuronal Artificial, Sistema de Inferencia Borroso, Lógica Borrosa, Funciones de Membresía, Sistemas Neuro-Borrosos, COBOR 2.0.

DESCRIPCIÓN O CONTENIDO: En la generación de los Sistemas de Inferencia Borrosos, la tarea primordial es la extracción y el ajuste de las funciones de membresía y las reglas borrosas. Sin embargo, al usar los métodos tradicionales para realizar esta tarea, los resultados obtenidos no son los esperados y en la mayoría de casos se presentan graves inconvenientes. Esta propuesta de investigación presenta una metodología basada en Redes Neuronales Artificiales que permite extraer automáticamente las reglas borrosas y los parámetros de las funciones de membresía de un Sistema de Inferencia Borroso tipo Sugeno, partiendo de un conjunto de datos entrada-salida. Este trabajo favorecerá las aplicaciones de los sistemas neuro-borrosos en aplicaciones industriales, y en situaciones con alta complejidad matemática, que no presentan un modelo de solución, o en las cuales no es posible adoptar una estrategia convencional.

Se contempla el desarrollo de un software que facilitará la aplicación en el control de procesos, la predicción y la estimación de parámetros, el cual será diseñado e implementado en Delphi 7.0 y servirá de apoyo a los estudiantes de materias relacionadas con la Inteligencia Artificial.

*Trabajo de Investigación

**Facultad de Ciencias Físico-Mecánicas, Maestría en Ingeniería, ME. Fernando Ruiz Díaz.

SUMMARY

TITLE: METHODOLOGY FOR THE AUTOMATIC GENERATION OF FUZZY RULES AND TUNING OF ADAPTATIVE MEMBERSHIP FUNCTIONS FOR MEANS OF THE NEURAL NETWORKS ARCHITECTURE -NETFUZ 1.0- *

AUTHOR: REYES FIGUEROA Juan Carlos**

KEYWORDS: Artificial Neural Network, Fuzzy Inference Systems, Fuzzy Logic, Membership Functions, Fuzzy-Neural Systems, COBOR2.0

DESCRIPTION OR CONTENT: In the generation of the Fuzzy Inference Systems, the primordial task is the extraction and the tuning of the memberships functions and the fuzzy rules. However, when using the traditional methods to carry out this task, the obtained results are not the prospective ones and in most of cases serious inconveniences are presented. This article presents a methodological proposal base in Artificial Neural Networks that allows extracting the fuzzy rules and the parameters of the memberships functions of a Fuzzy Inference System type Sugeno automatically, leaving of a group of data input-output. This work will favor the applications of the neuro-fuzzy systems in industrial applications, and in situations with high mathematical complexity that they don't present a solution model, or in which it is not possible to adopt a conventional strategy.

The development of a software is contemplated that will facilitate the application in the control of processes, the prediction and the estimate of parameters, which will be designed and implemented in Delphi 7.0 and it will serve from support to the students of matters related with the Artificial Intelligence.

---

* Work of Research

** Faculty of Physical-Mechanics Engineering's, Master of Engineering, ME. Fernando Ruiz Díaz

AGRADECIMIENTOS

A todos aquellos que lo merecen

> Cuando pones la proa visionaria hacia una estrella y tiendes el ala hacia tal excelsitud inasible, afanoso de perfección y rebelde a la mediocridad, llevas en ti el resorte misterioso de un ideal. Es ascua sagrada, capaz de templarte para grandes acciones. Custódiala, si la dejas apagar no se reenciende jamás. Y si ella muere en ti, quedas inerte: fría bazofia humana. Sólo vives por esa partícula de ensueño que te sobrepone a lo real [...] Es de pocos esa inquietud de perseguir ávidamente una quimera, venerando a filósofos, artista y pensadores, que fundieron en síntesis supremas sus visiones del ser y de la eternidad, volando más allá de lo real. Los seres de tu estirpe, cuya imaginación se puebla de ideales y cuyo sentimiento polariza hacia ellos la personalidad entera, forman raza aparte de la humanidad: son **idealistas**.
>
> *Citado por José Ingenieros en el Hombre Mediocre.*

Este proyecto de investigación debe ser visto como el fruto de un arduo trabajo realizado, que se inició gracias a los sabios consejos recibidos a tiempo, los cuales permitieron un cambio radical en mi vida como estudiante y templanza en lo personal.

Entre las personas que me han acompañado y han sido un soporte fundamental en mi vida, ya sea con sus consejos o con su apoyo incondicional destaco:

A Dios, por regalarme el tesoro tan preciado de la vida, y rodearme de personas que me quieren y desean todo lo mejor para mi.

A mi madre Maria Antonia, que me ha enseñado que con trabajo duro, perseverancia y honestidad, se pueden lograr los ideales trazados. Gracias por su apoyo incondicional y por enseñarme los valores para convertirme en persona de bien e inculcar el mí el respeto por los demás.

Al profesor Fernando Ruiz, que gracias a sus sabios consejos me ayudó a cambiar mi vida, enseñándome que con trabajo y sacrificio podemos lograr nuestras metas. Gracias por los consejos y regaños recibidos a tiempo, yo se que fueron para mi bienestar.

A mis amigos Giovanni, Juan José, José Luís, Urbano, Sonia y Uriel, gracias por la amistad que me han ofrecido, han sido unas personas especiales para mi formación como

profesional y como persona, gracias por los momentos de alegría y felicidad que me han brindado, y espero que este logro personal, lo sea también de ustedes.

Al profesor Jorge Luís Chacón, Después del profesor Fernando, el profesor Jorge Luís me brindo su conocimiento y consejo de manera incondicional, que fueron de mucha ayuda para la consecución de los objetivos propuestos.

A todos aquellos que lo merecen, que de alguna u otra manera colaboraron y sin las cuales no hubiera sido posible la elaboración de este proyecto

JUAN CARLOS

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

CAPITULO 1

INTRODUCCIÓN

Los sistemas de control presentan gran importancia en el desarrollo tecnológico y la innovación científica, pero surgen inconvenientes al tratar de resolver problemas que conllevan un alto grado de realismo por medio de métodos convencionales, debido a que los resultados obtenidos por ellos exhiben un comportamiento que dista de la realidad. Este hecho fomentó una búsqueda incesante de nuevas vías que garanticen una solución apropiada cuando se afrontan este tipo de situaciones.

De esta manera se manifiesta una revolución de pensamiento, suceso que propició un cambio en la orientación de las investigaciones adelantadas por parte de reconocidos científicos, en pos del desarrollo de nuevas e innovadoras metodologías para la solución de problemas cercanos a la realidad en los cuales no es posible, hasta ahora, establecer un modelo que determine su comportamiento por medio de estrategias clásicas.

Por consiguiente surgieron nuevas formas para abordar problemas que presentan dificultad para ser descritos mediante un enfoque algorítmico tradicional, éstas emulan características propias de los sistemas biológicos humanos y son conocidas como: Soft Computing o Computación Blanda **[1]**, las cuales representan un nuevo criterio para el procesamiento de la información, capaz de manejar las imprecisiones e incertidumbres que aparecen cuando se trata de resolver problemas relacionados con el mundo real.

Partiendo de esta premisa, brotaron nuevas formas de computación soportadas en características del razonamiento humano y cualidades propias de los seres vivos. La Lógica Borrosa, sintetizada por el matemático Iraní Lofti Zadeh en al año de 1975 **[2]**, plantea una manera de formalizar el razonamiento basado en el lenguaje natural soportado bajo el concepto de los Conjuntos Borrosos **[3]**. Las Redes Neuronales Artificiales, propuestas inicialmente por McCulloch y Pitts en el año de 1943 **[4]** y desarrolladas a posteriori por investigadores como: Widrow, Rosemblatt, Kohonen, entre otros. Emulan cualidades de los sistemas neuronales biológicos y se aplican en problemas de alta complejidad y no linealidad.

De la misma manera Afloraron metodologías como los Algoritmos Genéticos, El Recocido Simulado y Los Agentes Inteligentes. Que igual a las mencionadas con antelación, se inspiraron en características inherentes a los seres humanos y presentan gran relevancia en el tratamiento de situaciones complejas, las cuales exhiben la particularidad de no ser solubles con las técnicas clásicas reconocidas hoy.

1.1 Contexto General

1.1.1 Descripción Del Problema

Las Tecnologías de Soft Computing han manifestado ser una alternativa válida para generar resultados que satisfagan las expectativas del experto en procesos ante situaciones de elevada complejidad y alto grado de realismo, donde es imposible establecer un modelo que determine su comportamiento por medio de estrategias convencionales, hasta ahora. Una de ellas se conoce como La Lógica Borrosa.

Es necesario recalcar que La Lógica Borrosa se basa en La Teoría de conjuntos borrosos la cual posibilita imitar el comportamiento de La Lógica Humana. El término "borroso" procede de la palabra inglesa "Fuzzy" que significa: confuso, difuso, indefinido o desenfocado. Hace parte de La Inteligencia Artificial y se funda en el concepto "Todo es cuestión de cuanto se cumple", lo cual permite manejar información vaga o de difícil especificación.

Por otra parte permite la imprecisión en la representación de un problema y aún así alcanza una solución favorable, maneja la incertidumbre, la ambigüedad y la vaguedad. Reconoce más que simples valores verdaderos y falsos por proposiciones que pueden ser representadas con grados de veracidad o falsedad.

Dentro de este contexto cabe señalar a Los Sistemas de Inferencia Borrosos como las aplicaciones más relevantes de La Lógica Borrosa, los cuales se basan en reglas de la forma " IF - THEN", donde los valores lingüísticos de la premisa y el consecuente están definidos por Conjuntos Borrosos. Hoy imperan tres tipos de Sistemas de Inferencia Borrosos: Mandami, Takagi-Sugeno-kang (TSK) y Tsukamoto, los cuales se diferencian, claramente, en la estructura de las reglas borrosas y los métodos de desemborronado empleados.

Los Sistemas de Inferencia Borrosos se utilizan para modelar el comportamiento de un sistema real por medio de un proceso de inferencia que consiste en extraer una conclusión a partir de ciertas premisas y un conjunto de reglas, además son una plataforma popular de cómputo basada en los conceptos de La Teoría de los Conjuntos, Reglas y Razonamiento borrosos. La estructura básica de los Sistemas de Inferencia Borrosos consta de cinco componentes: Emborronado, Base de Datos, Base de Reglas, Mecanismo de Inferencia y Desemborronado.

Lo cierto es que el diseño de Los Sistemas de Inferencia Borrosos implica habilidades por parte de los expertos en control, que de no ser así, acarrearían inconvenientes que irían en detrimento del buen funcionamiento del sistema como tal, las dificultades más frecuentes a los que se enfrenta el diseñador de los Sistemas de Inferencia Borrosos son condensadas en los siguientes interrogantes: ¿Cuántas Funciones de Membresía se deben utilizar?, ¿qué tipo de Función de Membresia se debe emplear?, ¿cuántas reglas borrosas se deben generar?

Por esta razón se hace imperiosa la necesidad de implementar estrategias que optimicen el diseño de los Sistemas de Inferencia Borrosos con respecto a los problemas citados con antelación, hoy se trabaja en metodologías hibridas que integran técnicas propias del Soft computing, una de ellas son los sistemas Neuro-Borrosos, que combinan la capacidad de aprendizaje de las Redes Neuronales Artificiales con la tolerancia a fallos, interpretabilidad y robustez de los SIB **[5]**.

En suma se pretende condensar una metodología que permita generar la base de reglas borrosas y ajustar los parámetros de las Funciones de Membresía de manera automática para un Sistema de Inferencia Borroso tipo TSK, a partir de un conjunto de datos entrada – salida, hecho que representa mayor facilidad para el diseño de los Sistemas de Inferencia y evita errores causados por los operadores humanos.

1.1.2 Objetivos

1.1.2.1 Objetivo General

Diseñar e implementar una metodología para favorecer el desarrollo de software en la escuela de ingeniería de sistemas (EISI), basado en técnicas de razonamiento aproximado, aplicado en la solución de problemas de ingeniería y como soporte a los cursos de inteligencia artificial.

1.1.2.2 Objetivos Específicos

- Realizar el estado del arte, para determinar que metodologías existen hoy, referentes a este tipo de técnicas y sus posibles aplicaciones, permitiendo la generación de nuevo conocimiento y asegurando la idoneidad de la investigación.

- Implementar un componente software basado en la metodología de Redes Neuronales Artificiales que permita:

    - ✓ Ajustar automáticamente las funciones de pertenencia de un sistema de inferencia borroso.
    - ✓ Generar automáticamente la base de reglas borrosas.

- Realizar un análisis comparativo entre un Sistema de Inferencia Borroso tradicional y el generado por el componente software, con el fin de determinar la confiabilidad y robustez del proyecto.

- Proponer los requerimientos y el diseño básico para el software a implementar, utilizando la metodología del prototipado.

1.1.3 Justificación

Los avances obtenidos por el empleo de las tecnologías blandas, en los diferentes campos de las ciencias e ingenierías, incrementó el interés por su uso, el cual se ve reflejado en las múltiples aplicaciones que han transformado drásticamente el pensamiento de los investigadores hoy.

Hasta ahora las aplicaciones desarrolladas han jalonado una nueva manera de abordar situaciones con alto grado de realismo, imposibles de tratar por medio de estrategias clásicas, con resultados que resultan abrumadores, debido a la precisión obtenida y a la sencillez del modelo implementado.

Aun así, el grupúsculo que maneja y entiende los fundamentos y operaciones básicas de las Tecnologías Blandas es bastante reducido, ya sea por el miedo a cambiar los paradigmas clásicos o debido a la actitud misoneísta de los actuales investigadores y hombres de ciencia.

Por esta razón no se ha alcanzado la plenitud en el desarrollo de metodologías como La Lógica Borrosa, Las Redes Neuronales Artificiales, los Algoritmos Genéticos y otras que hacen parte de las Tecnologías Blandas. Sólo un pequeño grupo ha intentado generar nuevas ideas y posibles combinaciones que fortalezcan estas tecnologías, en procura de desarrollar herramientas más completas que faciliten su aplicabilidad en campos inexplorados por las ciencias hoy.

La presente investigación, busca, esencialmente, crear conciencia en los investigadores acerca de las fortalezas de las tecnologías blandas, y promover su aplicabilidad, no en todos pero si, en una gran mayoría de situaciones tratadas, rutinariamente, en los centros de investigación y las universidades. Además de facilitar la interpretación de sus fundamentos y limitaciones.

CAPITULO 2

MARCO TEÓRICO

El fundamento teórico incluye la conceptualización necesaria para el desarrollo del presente trabajo de investigación. Se divide en dos partes: marco teórico general y marco teórico específico, la primera incluye aspectos como: historia de Los Sistemas de Inferencia Borrosos, La Lógica Borrosa y Las Redes Neuronales Artificiales, la segunda aborda el tema de Control Clásico y su evolución hasta los controladores inteligentes.

2.1 Lógica Borrosa

La Lógica Borrosa (LB) se basa en La Teoría de Conjuntos Borrosos la cual posibilita imitar el comportamiento de La Lógica Humana. El término "borroso" procede de la palabra inglesa "Fuzzy" que significa: confuso, difuso, indefinido o desenfocado. Hace parte de La Inteligencia Artificial y se funda en el concepto "Todo es cuestión de cuanto se cumple", lo cual permite manejar información vaga o de difícil especificación.

Es necesario recalcar que La LB es más flexible que La Lógica Clásica; define la realidad en grados de verdad siguiendo patrones de razonamiento similares a los del pensamiento humano. Permite "imprecisión" en la representación de un problema y aún así llega a una muy buena solución, maneja la incertidumbre, la imprecisión y reconoce más que simples valores verdaderos y falsos por proposiciones que pueden ser representadas con grados de veracidad o falsedad. Desde el punto de vista de La Inteligencia Artificial (IA), es un método de razonamiento de máquina similar al pensamiento humano que puede procesar información incompleta o incierta, característico de los sistemas expertos **[6]**.

Cabe señalar a Los Sistemas de Inferencia Borrosos (SIB) como las aplicaciones más relevantes de La LB, los cuales se basan en reglas de la forma " IF - THEN", donde los valores lingüísticos de la premisa y el consecuente están definidos por conjuntos borrosos. Un conjunto borroso A en X se define como un conjunto ordenado de pares.

$$A = \left\{ (x, \mu_A(x)) \middle| x \in X \right\} \tag{1}$$

$\mu_A(x)$ Se denomina pertenencia de x con respecto a A

A diferencia de los Sistemas Expertos las reglas involucradas en un SIB pueden ser desarrolladas con sistemas adaptativos, que aprenden al 'observar' como operan las personas los dispositivos reales o ser formuladas por un experto humano. En general estos sistemas de inferencia se aplican tanto a problemas de control como para modelar cualquier sistema continuo de Ingeniería, Física, Biología y Economía, se define así un modelo matemático que opera con funciones no lineales, que convierten las entradas en salidas acordes con los planteamientos lógicos que usan el razonamiento aproximado.

En resumidas cuentas La LB y los SIB tienen una gran variedad de aplicaciones que van desde la estimación de parámetros, toma de decisiones, sistemas mecánicos de control tales como el aire acondicionado o lavadoras automáticas, hasta el control de automóviles o casas "inteligentes". Las nociones como "más bien caliente" o "poco frío" pueden formularse matemáticamente y ser procesadas por computadoras. Permite además compaginar significancía y precisión (ver Figura 1) y representa un área fascinante de la investigación.

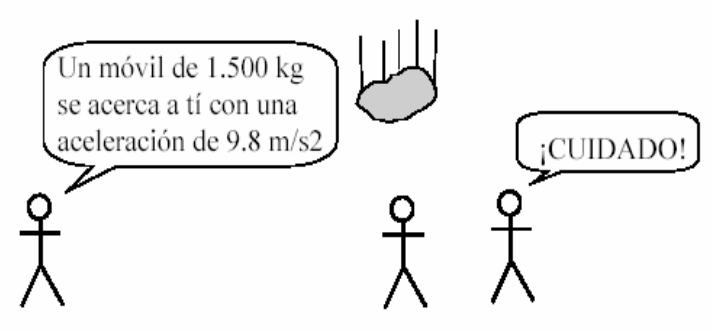


Fig. 1 Significancia y Precisión

2.1.1 Antecedentes De La Lógica Borrosa. Sorprende comprobar que en la actualidad se han incorporado nuevos sistemas de automatización, debido a la necesidad de rebajar costos y facilitar la tarea para el operador de procesos complejos, aplicados en situaciones de no linealidad en las cuales los métodos tradicionales proveen un resultado distante a la realidad, motivado por el carácter cualitativo de la información con que se cuenta.

En principio surgieron los controladores convencionales, utilizados aún en la industria, los cuales requieren de un modelo cuantitativo del problema; no siempre posible, especialmente en procesos complejos y donde exista no linealidad y en los cuales el experto juega un papel importante usando su intuición, habilidad heurística y experiencia. Es frecuente encontrar casos donde los controladores diseñados bajo los patrones típicos conocidos (clásicos, adaptables), no obtienen los resultados deseados o fallan del todo. Se trata de procesos de elevada complejidad y a pesar de ello, en la mayoría de los casos, operadores humanos diestros y experimentados “expertos” alcanzan resultados satisfactorios en su dirección. Cuando se investiga acerca de la forma como se manejan este tipo de procesos se deduce que la información acerca del estado; tanto como de las acciones de control se manejan de manera cualitativa.

Es típico para el proceder humano el dirigir procesos simples sólo con el uso de la intuición. ¿Cómo actúa un conductor de un automóvil? El nunca accionara sobre el timón en términos de radianes o grados, ni siquiera sobre el pedal del freno en milímetros o newton. El ante una esquina cerrada, gira el timón mucho, al estar cerca de la señal de pare frena un poco. Sin embargo, nadie pone en duda las excelentes virtudes del ser humano para conducir autos, las que por lo general llegan a la perfección.

Debido a los grandes inconvenientes presentados, el hombre en su afán de desarrollo implemento ciertas técnicas para solucionar este tipo de problemas, una de las más efectivas es La Teoría de los Conjuntos Borrosos que ha hecho posible establecer una modalidad llamada “Control Inteligente”.

El término borroso es introducido por primera vez gracias a Lofti Zadeh en 1962 **[7]**, en un trabajo que vincula la teoría de circuitos eléctricos con la de sistemas. Tres años más tarde Zadeh da luz a la llamada Teoría de los Conjuntos Borrosos que echa los cimientos de la síntesis lingüística, mostrando como pueden usarse planteos lógicos vagos para derivar inferencias (también vagas) a partir de datos inciertos.

La importancia de La LB desde el punto de vista de la teoría del control de procesos es la de proveer un soporte cuando se quiere traducir el conocimiento heuristico experimentado, para luego ser expresado en frases lingüísticas imprecisas e implementado en algoritmos numéricos.

Para describir mejor la diferencia entre la posibilidad y la probabilidad se utilizará un ejemplo sencillo, use plantea que “Jacinto y Ana tengan X hijos en su matrimonio” donde es posible asociar la distribución de posibilidades con X, interpretando a F(a) como el grado de facilidad con la que Jacinto y Ana tuvieran hijos en su matrimonio. También se puede asociar una distribución de probabilidades con X, interpretando a P(a) como la probabilidad de que Jacinto y Ana tuvieran hijos en su matrimonio, los valores de F(a) y P(a) se muestran en La Tabla 1:

X	1	2	3	4	5	6	7	8
F(X)	1	1	1	1	0.8	0.6	0.4	0.2
P(X)	0.1	0.8	0.1	0	0	0	0	0

Tabla1 Probabilidad Vs. Posibilidad

Se nota que mientras la posibilidad de tener 3 hijos es de 1, la probabilidad de que lo hiciera puede ser muy pequeña 0.1. De aquí que un alto grado de posibilidad no implica un alto grado de probabilidad ni un grado bajo de probabilidad implica un grado bajo de posibilidad. Por supuesto, si un evento es imposible queda circunscrito a ser improbable **[8]**.

2.2 Conjuntos Borrosos

Para internarse en el mundo de La LB, se deben dejar en claro los fundamentos en los cuales reposa y por medio de los que ha adquirido importancia. Por tal razón es necesario adentrarse en La Teoría de Los conjuntos borrosos que ha sido implementada en gran cantidad de aplicaciones y en las cuales consiguió resultados satisfactorios.

2.2.1 Historia. Los conjuntos borrosos (CB) son introducidos en el año de 1965, hecho que marcó el origen de La LB. La metodología de La LB es vista como un lenguaje que permite trasladar sentencias sofisticadas en lenguaje natural, a un lenguaje matemático formal. Aunque la motivación original fue apoyar el manejo de aspectos imprecisos del mundo real, la práctica temprana de La LB permitió el desarrollo de importantes aplicaciones.

En 1994, la teoría de Los CB se encontraba en la cumbre, pero esta idea no es nueva. Para muchos, estuvo bajo el nombre de LB durante 25 años, pero sus orígenes se remontan desde hace 2,500 años. Aristóteles consideraba que existían ciertos grados de veracidad, falsedad y grados de pertenencia.

En el siglo XVIII, el filósofo y obispo anglicano Irlandés George Berkeley y David Hume describieron que el núcleo de un concepto atrae conceptos similares. Hume en particular, creía en la lógica del sentido común, el razonamiento basado en el conocimiento que la gente adquiere en forma ordinaria mediante vivencias en el mundo. En Alemania, Emmanuel Kant, consideraba que sólo los matemáticos podían proveer definiciones claras y muchos principios contradictorios no tenían solución. Particularmente, la escuela americana de la filosofía llamada pragmatismo, fundada a principios de siglo por Charles Sanders Pierce, cuyas ideas se fundamentaron en estos conceptos, fue la primera en considerar "vaguedades", más que falso o verdadero, como forma de acercamiento al mundo y a la forma en que las personas funcionan.

La idea de que la lógica produce contradicciones fue popularizada por el filósofo y matemático británico Bertrand Russell a principios del siglo XX. Estudió las vaguedades del lenguaje, concluyendo con precisión que la vaguedad es un grado. El filósofo austriaco Ludwing Wittgenstein estudió las formas en las que una palabra puede ser empleada para muchas cosas que tienen algo en común. La primera lógica de vaguedades fue desarrollada en 1920 por el filósofo Jan Lukasiewicz. Visualizó los conjuntos con un posible grado de pertenencia con valores de 0 y 1, después los extendió a un número infinito de valores entre 0 y 1. En los años sesentas, Lotfi Zadeh inventó La LB, que combina los conceptos de la lógica y de los conjuntos de Lukasiewicz mediante la definición de grados de pertenencia **[9]**.

2.2.2 Notación de Los CB. Para describir un CB es necesario plantear el siguiente ejemplo**:** la Universidad Industrial de Santander (UIS), realiza un prueba para elegir a los aspirantes de las diferentes carreras ofrecidas en este ente universitario, con ella se catalogan a los candidatos en dos clases BAJO y ALTO, donde los que pertenezcan a la clase ALTO son admitidos, según La Lógica Clásica (LC) la representación de este conjunto se muestra en La Figura 2.

En primera instancia se define la variable puntaje obtenido para ingreso a la universidad, a continuación se crean los dos conjuntos a los que puede pertenecer un candidato que se presente a la prueba de admisión. Un aspirante pertenecería al conjunto nivel ALTO si obtiene un puntaje de 80 puntos o superior, en caso contrario pertenecería al conjunto BAJO ¿Qué sucedería con un aspirante que obtenga 79 puntos? Para La LC es obvio que estaría encasillado en el conjunto BAJO. Pero ¿Sería justo pensar que por un solo punto no pueda ser admitido?

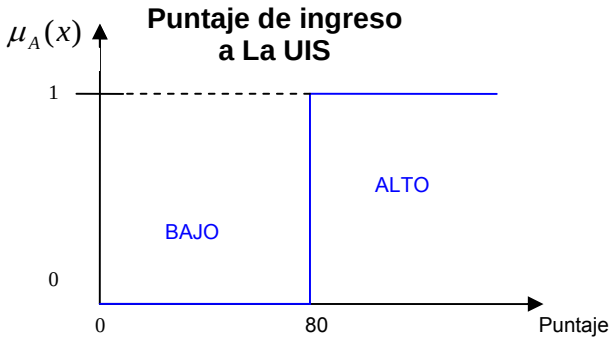


Fig. 2 Conjunto Clásico

La forma tradicional de representación de la pertenencia de un elemento a un conjunto se conoce como función de pertenencia o membresía y se denota de la siguiente manera:

$$\mu_A(x) \begin{cases} 1 \; si \; x \; pertenece \; al \; conjunto \; A \\ 0 \; si \; x \; no \; pertenece \; al \; conjunto \; A \end{cases} \tag{2}$$

Para los conjuntos clásicos X se define como una colección de elementos u objetos x, tal que para cada x exista una pertenencia o no de este elemento al conjunto denotado por A, esto se representa por medio de un conjunto de pares ordenados donde:

$$\begin{matrix} (x,0) & (x,1) \\ x \notin A & x \in A \end{matrix} \tag{3}$$

En La LB Las Funciones de Membresía no están definidas sólo por dos valores, existe una gama de grados de pertenencia de un elemento a un conjunto, luego es válido pensar que un CB no tiene fronteras definidas:

$$\mu_A(x) = X \rightarrow [0,1] \tag{4}$$

En los CB se tiene a X como una colección de objetos denotados por x, el CB $A \subseteq X$, se define como un conjunto de pares ordenados tales que:

$$A = \{(x, \mu_A(x)), x \in X\} \tag{5}$$

Donde $\mu_A(x)$ representa La Función de Membresía, estableciendo una relación de cada elemento del conjunto A, a un grado de pertenecía entre 0 y 1.

Los CB están definidos para el caso discreto de la siguiente manera:

$$A = \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + ... + \frac{\mu_A(x_n)}{x_n} \qquad \text{Luego } A = \sum_{i=1}^{n} \frac{\mu_A(x_i)}{x_i} \tag{6}$$

Y para el caso continuo:

$$A = \int_R \frac{\mu_A(x_i)}{x_i} \tag{7}$$

Del ejemplo anterior sobre de los puntajes de admisión en La UIS, se cuestionaba el hecho que un aspirante con un puntaje de 79 no fuera admitido, luego La LC no es muy flexible ante este tipo de situaciones. Es de amplio conocimiento que el tipo de problemas que se manejan en la vida real poseen este comportamiento, fenómeno que motivó el uso de una técnica apropiada para abordar casos de naturaleza similar, La LB afronta las situaciones de una manera diferente a la tradicional al dar un enfoque parecido al del razonamiento humano, necesario para obtener resultados aproximados en procesos imposibles de afrontar por estrategias convencionales, con alta confiabilidad y facilidad de uso. El conjunto anterior basado en la Teoría de los conjuntos borrosos quedaría como el presentado en La Figura 3:

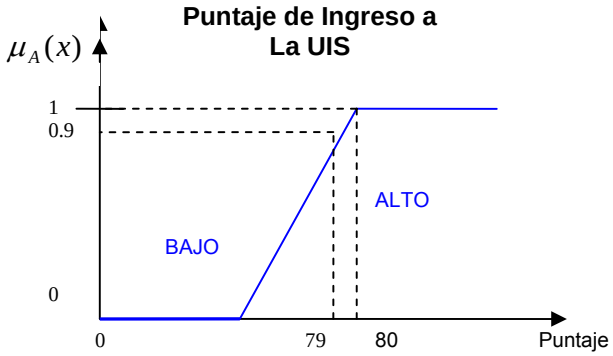


Fig. 3 Conjunto Borroso

En el caso de La LC el aspirante del ejemplo tendría un puntaje que lo colocaría en el nivel BAJO, en La LB pertenecería al nivel ALTO con un grado de pertenecía de 0.9 y al nivel BAJO con un grado de pertenencia de 0.1, algo que es más cercano a una situación real, con lo anterior se comprueba en parte las bondades ofrecidas por esta nueva metodología.

2.3 Terminología De Los CB

2.3.1 Soporte. El soporte de un CB A, es el conjunto de los puntos o elementos que estén contenidos en el universo de discurso, $x \in X$, tales que:

$$Soporte(A) = \{x \in X, \mu_A(x) > 0\} \quad (8)$$

2.3.2 Normalidad. Un CB A es normal si su núcleo no es un conjunto vació, quiere decir que siempre se encontrará un punto, $x \in X$, tal que $\mu_A(x) = 1$.

$$Normalidad(A) = \{x \in X, \mu_A(x) = 1\} \quad (9)$$

2.3.3 Puntos cross-over. Los puntos de transición de un CB A "cross-over" son aquellos donde la función de pertenencia equivale a 0.5.

$$Crossover(A) = \{x \in X, \mu_A(x) = 0.5\} \quad (10)$$

2.3.4 CB unitario "Singleton". Es un CB donde el soporte es un único punto del universo X con $\mu_A(x) = 1$ y se le conoce como CB unitario o "Singleton".

2.3.5 Núcleo. Es aquel donde los elementos $x \in X$ poseen un valor de $\mu_A = 1$.

$$Nuceo(A) = \{x \in X, \mu_A(x) = 1\} \quad (11)$$

2.3.6 Altura. La altura se define como el elemento de CB a, que posee la función de pertenencia más alta.

$$Altura(A) = SUP_{x \in X} \mu_A(x) \quad (12)$$

Para apreciar con más claridad las operaciones citadas con antelación, es necesario ver La Figura 4.

2.3.7 α Corte y α Corte intenso. Los conjuntos α Corte y α corte intenso son aquellos que contienen todos los elementos del universo de discurso X donde la función de pertenencia es mayor o igual o solo "mayor" que un valor específico de α y se definen como:

α Corte:

$$^{\alpha}A = \{x \in X, \mu_A(x) \geq 0\} \quad (13)$$

α Corte Intenso:

$$^{\alpha+}A = \{x \in X, \mu_A(x) > 0\} \quad (14)$$

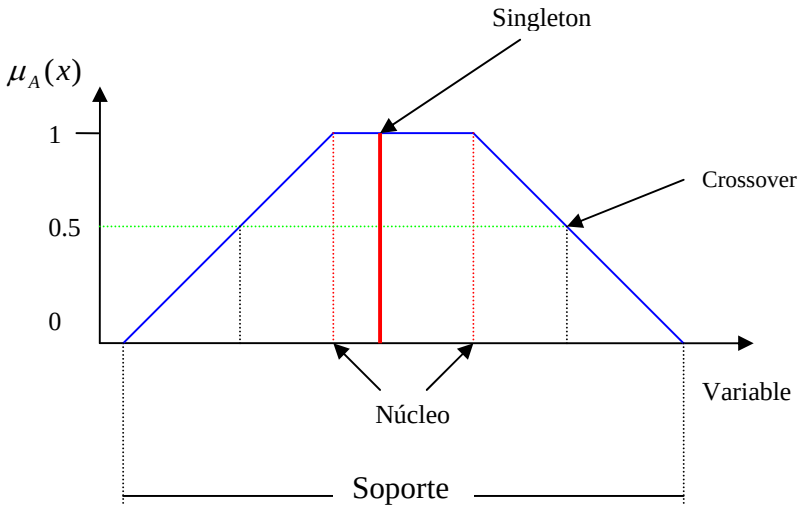


Fig. 4 Propiedades de los CB

Para entender mejor el concepto de los α Cortes, se da el siguiente ejemplo:
Se define la variable lingüística edad, luego se toman tres CB, de la siguiente manera:

$$A_1(x) = \begin{cases} 1;\ x \leq 20 \\ (35 - x)/15\ ;20 \leq x \leq 35 \\ 0\ ;\ x \geq 35 \end{cases}$$

$$A_2(x) = \begin{cases} 0;\ x \leq 45 \\ (x - 20)/15\ ;20 < x < 35 \\ (60 - x)/15\ ;45 < x < 60 \\ 1;35 \leq x \leq 45 \end{cases}$$

$$A_3(x) = \begin{cases} 0;\ x \leq 45 \\ (x - 45);\ 45 < x < 60 \\ 1;\ x \geq 60 \end{cases}$$

Al graficar estos conjuntos, que se definen en un universo de discurso $X\ [0\ 80]$ años de edad, ver Figura 5.

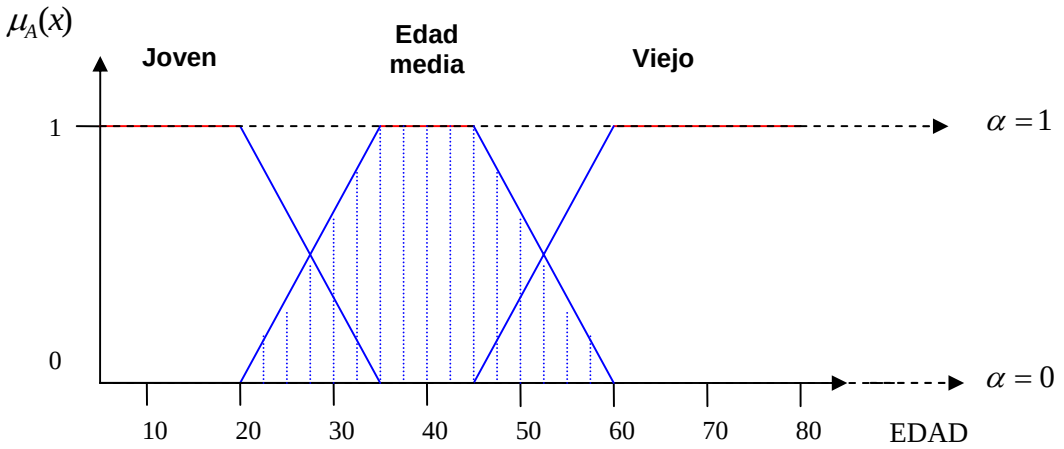


Fig. 5 α corte y α corte intenso

Luego se toma el conjunto edad media como referente para calcular las respectivas Funciones de Membresía en puntos representativos, los resultados se presentan en La Tabla 2:

X	$\mu_A(x)$
$x \notin \{22,24,\ldots.58\}$	0.00
$x \in \{22,58\}$	0.13
$x \in \{24,56\}$	0.27
$x \in \{26,54\}$	0.40
$x \in \{28,52\}$	0.53
$x \in \{30,50\}$	0.67
$x \in \{32,48\}$	0.80
$x \in \{34,46\}$	0.93
$x \in \{36,38,\ldots 44\}$	1.00

Tabla 2 Aproximación Discreta de La Función de Membresía Para el Conjunto A_2

Luego se determinan dos valores para α, 0 y 1, respectivamente, y se calcula su α Corte y α Corte intenso, donde:

${}^{0}A_1 = {}^{0}A_2 = {}^{0}A_3 = [0,80] = X$;

${}^{\alpha}A_1 = [0,35 - 15\alpha]$, ${}^{\alpha}A_2 = [15\alpha + 20,60 - 15\alpha]$, ${}^{\alpha}A_3 = [15\alpha + 45,80]$ Para todo $\alpha \in (0,1]$;

${}^{\alpha+}A_1$ = (0.35-15α), ${}^{\alpha+}A_2$ = (15α +20,60-15α), ${}^{\alpha+}A_3$ = (15α +45,80) para todo $\alpha \in [0,1)$;

${}^{1+}A_1 = {}^{\alpha+}A_2 = {}^{\alpha+}A_3 = 0$;

2.3.8 CB convexo. Un CB A es llamado convexo si y solo si, para cada $x_1, x_{2.....x_n} \in X$ Y para cualquier $\lambda \in [0,1]$, se tiene:

$$\mu_A(\lambda x_1 + (1-\lambda)x_2) \geq \min\{\mu_A(x_1), \mu_A(x_2)\} \quad (15)$$

2.3.9 Ancho de banda. Para un CB normal y convexo el ancho de banda o anchura es la distancia entre dos puntos extremos únicos, de tal manera que:

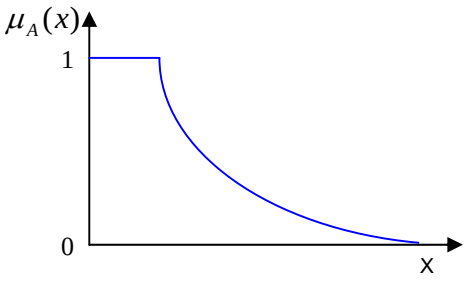
$$Ancho \quad (A) = |x_2 - x_1| \quad (16)$$
$$En\ donde: \ \mu_A(x_1) = \mu_A(x_2) = 0.5$$

2.3.10 Números borrosos. Un número borroso A, es un CB en el dominio real (R), si satisface las condiciones de normalidad y si es un conjunto convexo.

2.3.11 Simetría. Un CB es simétrico si su función de pertenencia es simétrica alrededor de determinado punto x = c, ósea que:

$$\mu_A(c + x) = \mu_A(c - x) \quad (17)$$

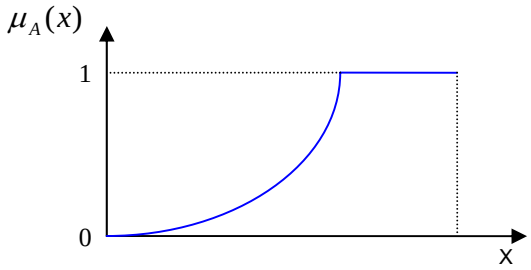
2.3.12 CB abierto a la izquierda. Un CB se dice abierto a la izquierda, si posee la siguiente trayectoria (ver Figura 6) y cumple la condición de:



$$\lim_{x \to -\infty} \mu_A(x) = 1 \ ; \lim_{x \to +\infty} \mu_A(x) = 0$$

Fig. 6 CB Abierto a La Izquierda

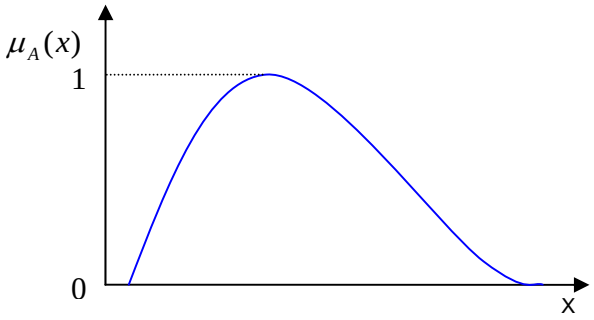
2.3.13 CB abierto a la derecha. Un CB se dice abierto a la derecha, si posee la siguiente trayectoria (ver Figura 7) y cumple la condición de:



$$\lim_{x \to -\infty} \mu_A(x) = 0 \ ; \lim_{x \to +\infty} \mu_A(x) = 1$$

Fig. 7 CB Abierto a La Derecha

2.3.14 CB cerrado. Un CB es cerrado, si posee la siguiente trayectoria (ver Figura 8) y cumple la condición de:



$$\lim_{x\to^{+}\infty} \mu_A(x) = \lim_{x\to^{+}\infty} \mu_A(x) = 0$$

Fig. 8 CB Cerrado

2.4 Operaciones Básicas De Los CB

Las operaciones básicas de los CB se derivan, principalmente, de los conjuntos clásicos o convencionales y, están claramente definidas, entre las que sobresalen: intersección, unión, inclusión y complemento.

2.4.1 Inclusión o subconjunto borroso. Un CB A esta contenido en un CB B, si y sólo si $\mu_A(x) \le \mu_B(x)$, para todo $x \in X$, luego se cumple que:

$$A \subseteq B \Leftrightarrow \mu_A(x) \le \mu_B(x) \tag{18}$$

2.4.2 Unión. La unión de dos CB A y B, es un CB denotado por C, que se describe de la siguiente manera $C = A \cup B$ o $C = A\,OR\,B$, y donde La Función de Membresía del conjunto C; esta relacionada por los conjuntos A y B:

$$\mu_C = Max\left(\mu_A(x), \mu_B(x)\right) = \mu_A(x) \vee \mu_B(x) \tag{19}$$

2.4.3 Intersección. La intersección de dos CB A y B, es un CB denotado por C, que se describe de la siguiente manera $C = A \cap B$ o $C = A\,AND\,B$ y, donde La Función de Membresía del conjunto C; esta relacionada por los conjuntos A y B:

$$\mu_C = Min\left(\mu_A(x), \mu_B(x)\right) = \mu_A(x) \wedge \mu_B(x) \tag{20}$$

2.4.4 Complemento o negación. El complemento de un CB A, se denomina ($\neg$A, NOT A) y se define como:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{21}$$

2.4.5 Producto Cartesiano. Sean A y B CB en "X" y "Y" respectivamente, el producto cartesiano de A y B denominado A x B, es un CB en el producto del espacio "X" y "Y", con Función de Membresía igual a:

$$\mu_{AxB}(x, y) = \min(\mu_A(x), \mu_B(x)) \tag{22}$$

2.4.6 Coproducto. De la misma manera que el producto cartesiano, A+B es un CB con Función de Membresía igual a:

$$\mu_{A+B}(x, y) = \max(\mu_A(x), \mu_B(x)) \tag{23}$$

Los anteriores operaciones de productos se caracterizan por tener una Función de Membresía en el espacio bidimensional, existen otras definiciones avanzadas de unión, intersección y complemento de CB como: las T-norms, y las T-conorms, que son generalizaciones de las anteriores operaciones, debido a que tanto las Funciones de Membresía de los CB como sus operaciones dependen del contexto y del experto que las define.

2.4.7 T-Norms. Generaliza las operaciones de intersección de los CB, en general es una operación binaria en el intervalo unitario, esto es una función de la forma:

$$T : [0,1]x[0,1] \rightarrow [0,1] \tag{24}$$

Para cada elemento x del conjunto universal, esta función asume el argumento del par consistente en los grados de pertenencia de los elementos en el conjunto A y el conjunto B, el grado de pertenencia del elemento en el conjunto constituye la intersección de A y B de la forma:

$$\mu_{A\cap B}(x) = T[\mu_A(x), \mu_B(x)] = \mu_A(x) * \mu_B(x) \tag{25}$$
$$Donde * representa\ el\ tipo\ de\ t-norm$$

El operador T-norm satisface los siguientes axiomas:

Condición de frontera:	$T(a,1) = a$
Monotonía:	$b \le d$ implica $T(a,b) \le T(a,d)$
Conmutativa:	$T(a,b) = T(b,a)$
Asociativa:	$T(a,T(b,d)) = T(T(a,b),d).$

Para todo $a, b, d \in [0,1]$

2.4.8 T-Conorms. Generaliza las operaciones de unión de los CB, en general es una operación binaria en el intervalo unitario, esto es una función de la forma:

$$S : [0,1]x[0,1] \rightarrow [0,1] \tag{26}$$

Para cada elemento x del conjunto universal, esta función toma el argumento del par consistente en los grados de pertenencia de los elementos en el conjunto A y el conjunto B, el grado de pertenencia del elemento en el conjunto constituye la unión de A y B de la forma:

$$\mu_{A\cup B}(x) = S[\mu_A(x), \mu_B(x)] = \mu_A(x) * \mu_B(x) \tag{27}$$
$$Donde * representa el tipo de t-conorm$$

El operador T-conorm satisface los siguientes axiomas:

Condición de frontera:	$S(a,1) = a$
Monotonía:	$b \le d$ implica $S(a,b) \le S(a,d)$.
Conmutativa:	$S(a,b) = S(b,a)$
Asociativa:	$S(a,S(b,d)) = S(S(a,b),d)$

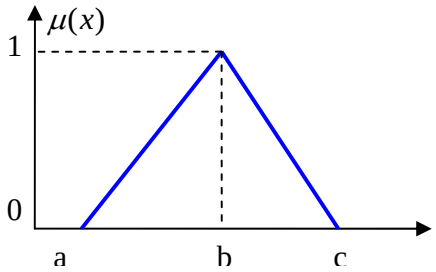
Para todo $a,b,d \in [0,1]$

2.5 Funciones De Membresía

Las funciones de membresía (FM) representan la manera gráfica para describir a un CB, son expresadas por medio de formulas matemáticas y definidas para el espacio unidimensional, bidimensional y multidimensional.

2.5.1 Funciones De Membresía Unidimensionales

2.5.1.1 FM triangular. Representada por medio de una figura triangular, utiliza tres parámetros (a, b, c) para hallar la pertenencia del elemento x (Figura 9).

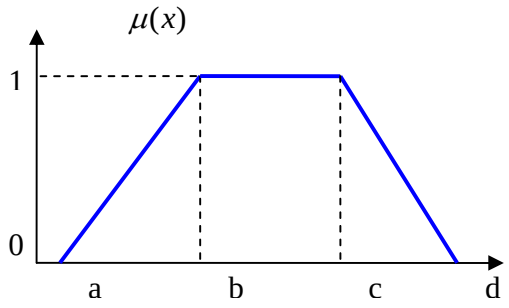


$$Triangular\,(x,a,b,c) = \begin{cases} 0 & x \le a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ \dfrac{c-x}{c-b} & b \le x \le c \\ 0 & c \le x \end{cases} \tag{28}$$

Fig. 9 FM Triangular

Los parámetros a, b, c determinan los tres ángulos de La FM triangular, con $(a,b,c) > 0 \quad y \quad a < b < c$.

2.5.1.2 FM trapezoidal. Representada por medio de una figura trapezoidal, utiliza cuatro parámetros (a, b, c, d) para hallar la pertenencia del elemento x (Figura 10).



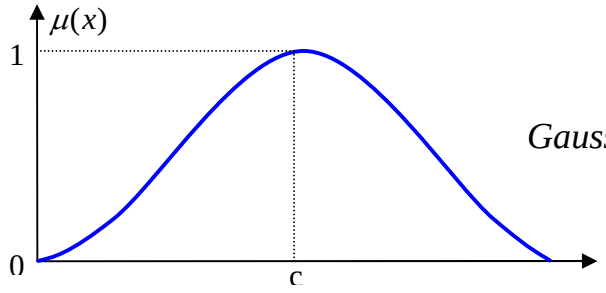
$$Trapezoida\ l\ (x,a,b,c,d) = \begin{cases} 0 & x \le a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ 1 & b \le x \le c \\ \dfrac{d-x}{d-c} & c \le x \le d \\ 0 & d \le x \end{cases} \qquad (29)$$

Fig. 10 FM Trapezoidal

Los parámetros (a, b, c, d) determinan las coordenadas de los cuatro ángulos de La FM trapezoidal, con $(a,b,c,d) > 0 \quad y \quad a < b < c < d$

Las FM triangular y trapezoidal se utilizan en mayor proporción en aplicaciones industriales reales, pero presentan la desventaja de no ser muy suaves en los puntos donde se forman los ángulos.

2.5.1.3 FM tipo campana de Gauss. Representada por medio de una Gaussiana, utiliza dos parámetros (c, σ) para hallar la pertenencia del elemento x (Figura 11).



$$Gaussiana(x,c,\sigma) = e^{-\left[\frac{x-c}{\sigma}\right]^2} \qquad (30)$$

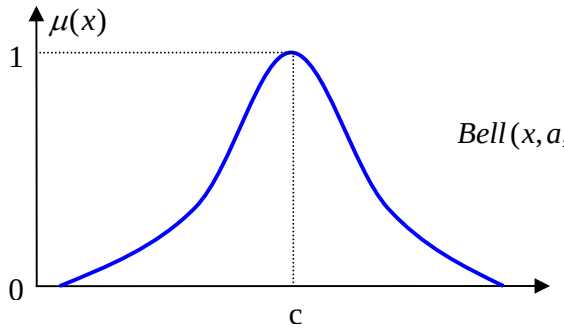
Fig. 11 FM Gaussiana

Las FM tipo campana de Gauss se determinan por medio de los parámetros **c** y σ, donde **c** representa el centro y σ representa el ancho de la FM.

2.5.1.4 FM tipo campana de Bell. Representada por medio de una campana de Bell, utiliza tres parámetros (a, b, c) para hallar la pertenencia del elemento x (Figura 12).

Esta FM es una generalización directa de la distribución de Cauchy utilizada en la teoría de las probabilidades, por eso se le conoce como FM de Cauchy, una FM particular de

Bell se obtiene a través de una selección correcta del conjunto de parámetros (a, b, c). Específicamente, se puede ajustar **c** y **a** para variar el centro y el ancho de la FM y utilizar **b** para controlar la inclinación de los puntos extremos.

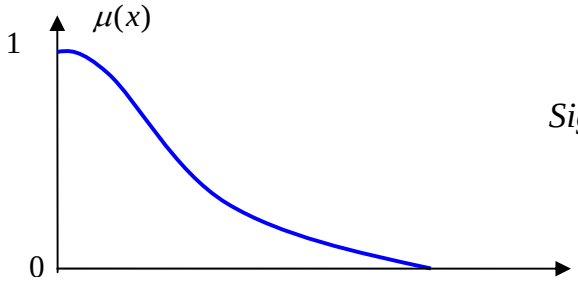


$$Bell(x,a,b,c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2*b}} \quad con\ b>0 \qquad (31)$$

Fig. 12 FM Tipo Campana de Bell

2.5.1.5 FM sigmoidal. Representada por medio de una figura Sigmoidal, utiliza dos parámetros (a, c) para hallar la pertenencia del elemento x (Figura 13).

El parámetro **a** controla la inclinación del punto de transición x = **c**, esta FM se utiliza en las Redes Neuronales Artificiales como función de activación y para simular el comportamiento de los SIB.



$$Sig\ (x,a,c) = \frac{1}{1+e^{[-a(x-c)]}} \qquad (32)$$

Fig. 20 FM Sigmoidal

2.5.2 FM bidimensionales. En ocasiones es ventajoso y necesario emplear FM con dos entradas cada una en un universo diferente, un método natural para extender las FM es mediante la extensión cilíndrica de las FM unidimensionales.

2.5.3 Extensión cilíndrica de un CB unidimensional. Si A es un CB en el universo X, entonces su extensión cilíndrica en XxY es un CB $C(A)$ definido por:

$$C\ (A)\ =\ \int_{XxY} \mu_{A}\ (x)\ /(\ x\ ,\ y\ ) \qquad (33)$$

2.5.3.1 Proyecciones de los CB. Sea R un CB bidimensional XxY, luego la proyección de R en X y Y se define como:

$$R_x = \int_X \left[\frac{\max \ \mu_R, (x, y)}{x} \right] \tag{34}$$

Y

$$R_y = \int_y \left[\frac{\max \ \mu_R, (x, y)}{y} \right] \tag{35}$$

2.5.4 Relaciones entre CB. Según los enfoques tradicionales, el diseño de un controlador se basa en un análisis cabal del proceso y, una vez que se posee un modelo cuantitativo de este, todas las decisiones se calculan usando algoritmos estrictamente numéricos. Es obvio que sólo a procesos "dóciles" al análisis cuantitativo le son aplicables técnicas de control de entidad numérica.

Por otra parte existe un número de procesos difíciles de controlar, automáticamente, en virtud de los obstáculos para su modelado, sin embargo, operadores humanos exhiben magníficos resultados en su dirección. Esto, junto con la aparición de La Teoría de los conjuntos borrosos, estimuló la investigación acerca de las estrategias de control de esos operadores, expresados por medio de reglas heurísticas para convertirlas en estrategias de control automático. El SIB resultante se basaría entonces en el modelo lingüístico de la estrategia del operador humano, ósea, en un modelo de decisión por parte del experto. La esencia del modelo es un programa basado en reglas. Por lo que se clasifica entre los llamados sistemas expertos.

2.5.4.1 Principio de extensión. El principio de extensión es el concepto básico de La LB y el cual suministra un procedimiento general para extender las expresiones matemáticas del dominio binario al dominio borroso (ver Figura 14). Este procedimiento generaliza el mapeo punto a punto de una función para establecer la relación entre dos CB.

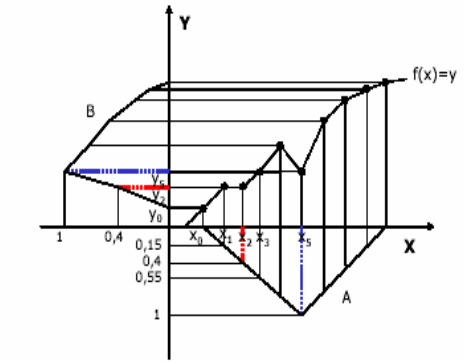


Fig. 14 Principio de Extensión Borrosa

Exactamente, se supone que f es una función entre $"X"$ y $"Y"$, y donde A es un CB definido en el espacio X como:

$$A = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + ... + \mu_A(x_n)/x_n \quad (36)$$

Por lo tanto el principio de extensión asevera que la imagen de un CB A bajo la acción de $f(.)$, se puede expresar como el CB B:

$$B = f(a) = \mu_A(x_1)/y_1 + \mu_A(x_2)y_2 + ... + \mu_A(x_n)/y \quad (37)$$

En donde, $Y_i = f(x_i)$ con $i = 1,2,...,n$ expresado de otra manera, el CB B se puede definir a través de los valores de $f(.)$ en $x_1.....x_n$. De una forma general:

$$\mu_B(y) = \max_{x=f^{-1}(y)} \mu_A(x) \quad (38)$$

2.5.4.2 Relaciones borrosas. Las relaciones borrosas binarias son CB en XxY los cuales relacionan cada elemento en XxY a un valor de pertenencia entre 0 y 1. En particular las relaciones borrosas unarias son CB con FM unidimensional; las relaciones borrosas binarias son CB con FM bidimensional y así sucesivamente.

2.5.4.3 Relación borrosa binaria. Sea "X" y "Y" dos universos de discurso, se concibe una relación borrosa binaria XxY de la forma:

$$\Re = \{((x, y), \mu_{\Re}(x, y)) \mid (x, y) \in XxY\} \quad (39)$$

Obsérvese que $\mu_{\Re}(x, y)$ es de hecho una FM bidimensional.

2.5.4.4 Composición Máx. – Min. Sean $\Re_1$ y $\Re_2$ dos relaciones borrosas definidas, respectivamente, en los espacios XxY y YxZ la composición máx.-min. De $\Re_1$ y $\Re_2$ es un CB definido por:

$$\Re_1 \cup \Re_2 = \{[(x, z), \max\min(\mu_{\Re 1}(x, y), \mu_{\Re 2}(y, z))] \mid x \in X, y \in Y, z \in Z \quad (40)$$

O su equivalente

$$\mu_{\Re_1 o \Re_2} = \max\min[\mu_{\Re_1}(x, y), \mu_{\Re_2}(y, z)] = \vee[\mu_{\Re_1}(x, y) \wedge \mu_{\Re_2}(y, z)] \quad (41)$$

Conociendo de antemano que $\vee$ y $\wedge$ representan las operaciones de máximo y mínimo, respectivamente, y donde $\Re_1$ y $\Re_2$ se expresan como relaciones en forma de matrices. El calculo $\Re_1$ o $\Re_2$ es una multiplicación de matrices, excepto que "x" y "+" se reemplazan

por $\vee\, y \wedge$. La composición máx.-min. es llamada producto máx.-min. El producto máx.-min. cumple propiedades comunes a las relaciones binarias como la asociativa, distributiva sobre la unión, distributiva débil sobre la intersección y la monotonía. El producto máx.-min. es difícil de analizar de manera matemática, este hecho induce el proponer una composición máx.-min. alternativa.

2.5.4.5 Composición producto – máx. La alternativa de la composición máx.-min., posee la siguiente estructura:

$$\mu_{\Re_1 \circ \Re_2}(x,z) = \max_y \left[\mu_{\Re_1}(x,y)\mu_{\Re_2}(y,z) \right] \tag{42}$$

2.6 Reglas Borrosas If - Then

2.6.1 Variable lingüística. Son variables cuyos valores se representan mediante términos lingüísticos y su significado se determina por medio de CB (ver Figura 15). Estas variables lingüísticas se caracterizan por poseer un conjunto de cinco elementos $(x, T(x), X, G, M)$ donde:

x	Representa el nombre de la variable
$T(x)$	Conjunto de términos lingüísticos de x
X	Universo de discurso de la variable x
G	Regla sintáctica para general los términos lingüísticos
M	Regla semántica que asigna a cada termino lingüístico t su significado $M(t)$, que es un CB en x.

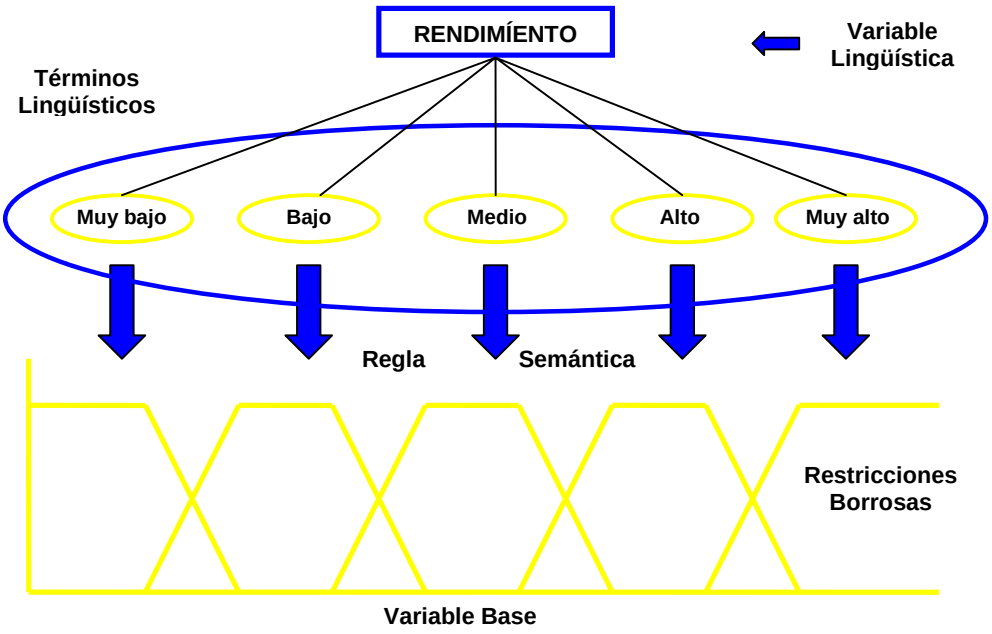


Fig. 15 Variable Lingüística General

2.6.2 Concentración y dilatación de valores lingüísticos. Sea A un valor lingüístico caracterizado por un CB con una FM $\mu_A(x)$ luego A^K se puede interpretar como una versión modificada del valor lingüístico original expresado como:

$$A^k = \int_x \left[\mu_A(x)\right]^k / x \tag{43}$$

Por lo tanto, la operación de concentración se define como:

$$CON\ (A) = A^2 \tag{44}$$

Y la dilatación se define como:

$$DIL(A) = A^{0.5} \tag{45}$$

Generalmente, se asume La $CON(A)$ y La $DIL(A)$, como el resultado de aplicar los modificadores "muy" y "más o menos" al termino lingüístico A. Sin embargo, se encuentran definiciones que se justifican de manera plena según la aplicación especifica.

Se interpreta al operador NOT y los conectores AND y OR de la siguiente manera:

$$NOT(A) = \neg A = \int_x \left[1 - \mu_A(x)\right] / x \tag{46}$$

$$A\,AND\,B = A \cap B = \int_x \left[\mu_A(x) \wedge \mu_B(x)\right] / x \tag{47}$$

$$A\,OR\,BA \cup B = \int_x \left[\mu_A(x) \vee \mu_B(x)\right] / x \tag{48}$$

Donde $A\,y\,B$ son dos valores lingüísticos cuyo significado esta definido por $\mu_A(.) y \mu_B(.)$.

Utilizando los $CON(.)\ y\ DIL(.)$ para los modificadores lingüísticos como "muy" y "más o menos", conjunto con las interpretaciones de los conectores $AND\,y\,OR$ es posible construir verdaderos "**términos lingüísticos compuestos"** como por ejemplo: no muy alto, no muy bajo, y alto pero no muy alto.

2.6.3 Intensificador de contraste. La operación de intensificación de contraste sobre un valor lingüístico A se define como:

$$INT(A) = \begin{cases} 2A^2 & Para\ 0 \le \mu_A(x) \le 0.5 \\ \neg 2(\neg A)^2 & Para\ 0.5 \le \mu_A(x) \le 1 \end{cases} \tag{49}$$

2.6.4 Conjunto ortogonal. Un conjunto de términos $T = t_1 ....., t_n$ de una variable lingüística x en el universo X es ortogonal si cumple la siguiente propiedad:

$$\sum_{t=1}^{n} \mu_{t_i}(x) = 1, \forall x \in X \tag{50}$$

En donde este CB es convexo y normal definido en X , además conforma el conjunto de términos T .

2.6.5 Reglas borrosas IF – THEN. La regla borrosa IF – THEN también conocida como implicación borrosa o declaración condicional borrosa, tiene la forma:

IF x es A THEN y es B

Donde $A\, y\, B$ son valores lingüísticos definidos en los universos de discurso $X\ y\ Y$ .por lo general el término $"x\, es\, A"$ se denomina antecedente o premisa y el término $"y\, es\, B"$ se denomina consecuente o conclusión.

Es normal encontrar ejemplos de expresiones lingüísticas como:

- Si la temperatura es alta, entonces aumente el caudal de agua.
- Si las ventas aumentan, entonces las ganancias son altas.

Es natural que la estructura de la regla borrosa "**If x es A then y es B**" se pueda describir de una manera abreviada como: $A \to B$. En esencia, esta expresión exhibe la relación entre las dos variables $"X"\, y"Y"$, lo que sugiere que la regla borrosa IF – THEN se defina como una relación borrosa binaria $\Re$ en el espacio del producto $X \times Y$.

Existen dos maneras de interpretar la regla borrosa $A \to B$

En primer lugar, se interpretar $A \to B$ como A acoplado con B luego:

$$R = A \to B = AxB = \int_{XxY} \mu_A(x) \,\overline{*}\, \mu_B(y)/(x, y) \tag{51}$$

En donde $\overline{*}$ es un operador T–Norm y $A \to B$ se utiliza de nuevo para representar la relación $\Re$.

En segundo lugar, se interpreta $A \to B$ como A implica a B , esta relación se escribe de cuatro maneras diferentes:

Implicación material:

$$R = A \to B = \neg A \cup B \tag{52}$$

Cálculo de proposiciones:

(53)

$$R = A \rightarrow B = \neg A \cup (A \cap B)$$

Cálculo de proposiciones extendido:

$$R = A \rightarrow B = (\neg A \cap B) \cup B \tag{54}$$

Modus Ponens Generalizado:

$$\mu_R(x, y) = \sup \left\{ c \middle| \mu_A(x) \overline{*} c \leq \mu_B(y) and \ 0 \leq c \leq 1 \right\} \tag{55}$$

En donde $R = A \rightarrow B$ y $\overline{*}$ es un operador T – Norm. Al basarse en estas dos interpretaciones y en los diferentes operadores T – Norm y T – Conorm, surge una gran variedad de métodos que son formulados en el cálculo de la relación borrosa $R = A \rightarrow B$. Es necesario resaltar que $\Re$, es visto como un CB el cual posee una FM bidimensional.

$$\mu_R(x, y) = f(\mu_A(x), \mu_B(y)) = f(a, b) \tag{56}$$

En donde $a = \mu_A(x) y, b \mu_B(y)$ cuando la función f se conoce como la función de implicación borrosa, esta realiza la transformación de los grados de pertenencia de $"x" en A \, y \, de" \, y" en \, B$, en los (x, y) de $A \rightarrow B$.

2.7 Razonamiento borroso. El razonamiento borroso o razonamiento aproximado extrae conclusiones a partir de un conjunto de reglas borrosas IF – THEN de hechos conocidos. Una herramienta necesaria para este proceso se conoce como La regla composicional de inferencia, que juega un papel importante para generar un razonamiento borroso válido.

2.7.1 Regla composicional de inferencia. El concepto de esta regla es similar a la composición máx.-min., de la relación matricial definida en (40). El principio de extensión es un caso especial de la regla composicional de inferencia (ver Figura 16).

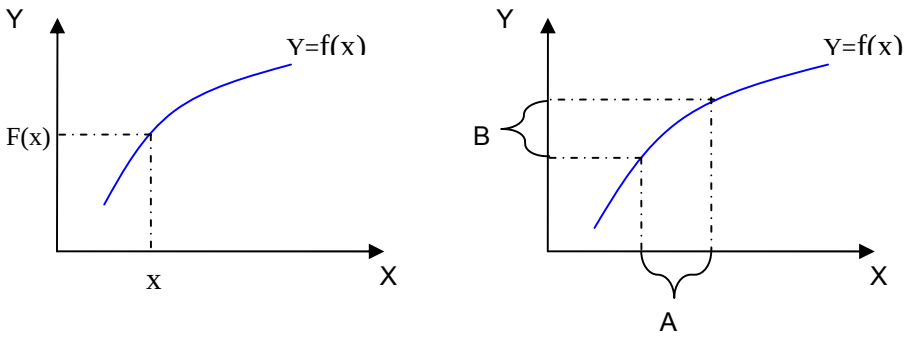


Fig. 16 Regla Composicional de Inferencia

La gráfica anterior representa la relación funcional entre dos variables:

- $x \to y$ donde $y = f(x)$
- $A \to B$ donde $B = \left[y \in Y \middle| y = f(x), x \in A \right]$

Sea la curva $y = f(x)$ que regula la relación entre $"X"$ y $"Y"$ cuando $X = x$, luego f(x) es una generalización de este proceso que permite tomar a $"x"$ como un intervalo y $f(x)$ como una función también de intervalo. Para construir el Intervalo $f(x)$ correspondiente al intervalo x, inicialmente, se construye la extensión cilíndrica de x y luego se halla la intersección con la curva intervalo.

La proyección de esta intersección en el eje Y produce $f(x)$. Al asumir F como una relación borrosa en $X \times Y$ y A como un CB de X. Para encontrar el CB resultante B, se construye una extensión cilíndrica $C(A)$ en base al conjunto A. La intersección de $C(A)$ y F forma un análogo de la intersección. Proyectando $c(A) \cap f$ sobre el eje $"Y"$ como un CB.

Con esta generalización obtenida por el reemplazo de las funciones características con sus correspondientes FM en la ecuación $\mu_B(y) \sup_{x \in X} \min\left[\mu_A(x), \mu_R(x, y)\right]$ se reduce la regla composicional de inferencia en una relación de matrices A y F que tienen un universo de discurso finito.

$$B = A \circ F \tag{57}$$

Donde "$\circ$ "denota el operador de composición.

Al utilizar la regla composicional de inferencia se formaliza la inferencia para el conjunto de reglas IF – THEN. Al que se denomina razonamiento aproximado o razonamiento borroso.

2.7.2 Razonamiento aproximado. Sean $A, A' y B$, CB con sus respectivos universos de discurso $X, X' y Y$. Si se asume que la implicación borrosa, se expresa como una relación borrosa $\Re$ sobre XxY. Entonces el conjunto B inferido por "**x es A**" y la regla borrosa "**si x es A entonces y es B**" se definen así:

$$\mu_{B'} = Max\ , Min\left[\mu_{A'}(x), \mu_R(x, y)\right] = \vee_x \left[\mu_{A'}(x) \wedge \mu_R(x, y)\right] \tag{58}$$

O en una forma equivalente:

$$B' = A' \circ R = A' \circ (A \to B) \tag{59}$$

Ahora se utiliza el procedimiento de inferencia o razonamiento borroso para obtener conclusiones, tendiendo en cuenta, que la relación borrosa de implicación $A \to B$ se define como una relación borrosa binaria apropiada. Las reglas If – Then y el razonamiento borroso son la espina dorsal de Los SIB.

El razonamiento borroso o razonamiento aproximado es un procedimiento que extrae conclusiones de un conjunto de reglas borrosas IF – THEN, sobre hechos conocidos. La regla básica de inferencia de La Lógica Binaria es el **Modus Ponens,** según la cual es posible inferir la verdad de la preposición B de la verdad A y de la implicación $A \to B$. Por ejemplo: si A se identifica con "el tomate es rojo" y B con "el tomate está maduro"; entonces, es cierto que si el "tomate es rojo"; también "el tomate está maduro".

Esto se ilustra a continuación:

Premisa1 (hecho)	$x \text{ es } A$
Premisa2 (regla)	$Si\ x \text{ es } A \text{ entonces } y \text{ es } B$
Consecuente (conclusión):	$y \text{ es } B$

Sin embargo, durante la mayor parte del razonamiento humano, el **Modus Ponens** se utiliza en forma aproximada.

Por ejemplo, se tiene la misma regla de implicación "Si el tomate es rojo, entonces el esta maduro", y es conocido que "el tomate es mas o menos rojo "; entonces es `posible inferir que "el tomate esta mas o menos maduro"; esto se ilustra a continuación:

Premisa1 (Hecho)	$x \text{ es } A'$
Premisa2 (Regla)	$Si\ x \text{ es } A \text{ entonces } y \text{ es } B$
Consecuente (Conclusión)	$y \text{ es } B'$

Donde A' esta cercano a A y B' esta cercano a B. Además A, A', B, B' son conjuntos borrosos con universos de discurso apropiados.

Ejemplo: "Sistema con una regla y un solo antecedente".

$IF\ x \text{ is } A\ THEN\ \ y \text{ is } B$	"Regla"
$x \text{ is } A'$	"Hecho"
B' = Conclusión	

$$\mu_{B'}(y) = Max_x Min\left[\mu_{A'}(x), \mu_R(x,y)\right] = \vee_x \left[\mu_{A'}(x), \mu_R(x,y)\right]$$

O su equivalente:

$$B' = A' \circ \left[A \to B\right] = A' \circ (A \to B)$$

$$\mu_{B'}(y) = \left[\vee_x \left(\mu_{A'}(x) \wedge \mu_A(x)\right)\right] \wedge \mu_B(y) \quad = \quad \mu_{B'}(y) = W \wedge \mu_B(y)$$

Donde W representa el grado de compatibilidad, certeza o de disparo de la regla, la gráfica de este sistema se muestra a continuación (ver Figura 17):

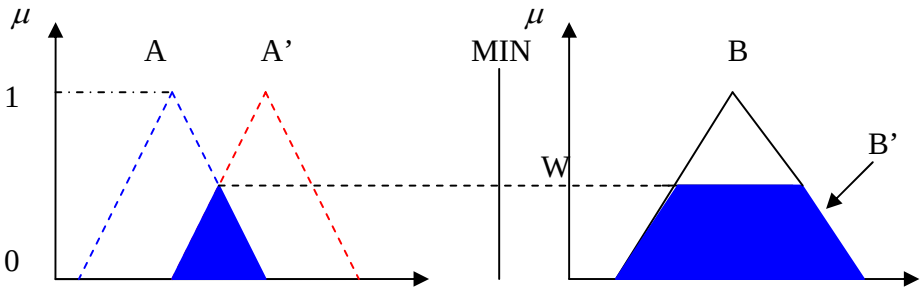


Fig. 17 Sistema con una Regla y un Antecedente

Por lo tanto, este procedimiento de inferencia se denomina razonamiento aproximado o razonamiento borroso, también se le conoce **Modus Ponens Generalizado (MPG).**

Ejemplo "Sistema de múltiples reglas y múltiples antecedentes".

La interpretación de varias reglas se toma como la unión de varias relaciones borrosas que corresponden a varias variables, por lo tanto el Modus Ponens Generalizado se describe como:

Premisa1 (Hecho)	$x \text{ is } A' \text{ AND } y \text{ is } B'$
Premisa2 (Regla)	$If \ x \text{ is } A_1 \text{ AND } y \text{ is } B_1 \text{ THEN } z \text{ is } C_1$
Premisa3 (Regla)	$If \ x \text{ is } A_2 \text{ AND } y \text{ is } B_2 \text{ THEN } z \text{ is } C_2$
Consecuente (Conclusión)	$z \text{ is } C'$

Luego:

$$C' = (A' \times B') \circ (R_1 \cup R_2) = [(A' \times B') \circ R_1] \cup [(A' \times B') \circ R_2]$$
$$C' = C_1' \cup C_2'$$

Que son los CB resultantes de las reglas 1 y 2, la gráfica de este sistema se muestra a continuación (ver Figura 18):

Cuando una regla borrosa tiene la forma $If \ x \text{ is } A \ OR \ y \text{ is } B \ THEN \ z \text{ is } C_2$, la intensidad del disparo se toma como el máximo grado de compatibilidad del antecedente para la condición dada.

Esta regla es equivalente a la unión de dos reglas borrosas $If \ x \text{ is } A \ THEN \ z \text{ is } C$ y $If \ y \text{ is } B \ THEN \ z \text{ is } C$.

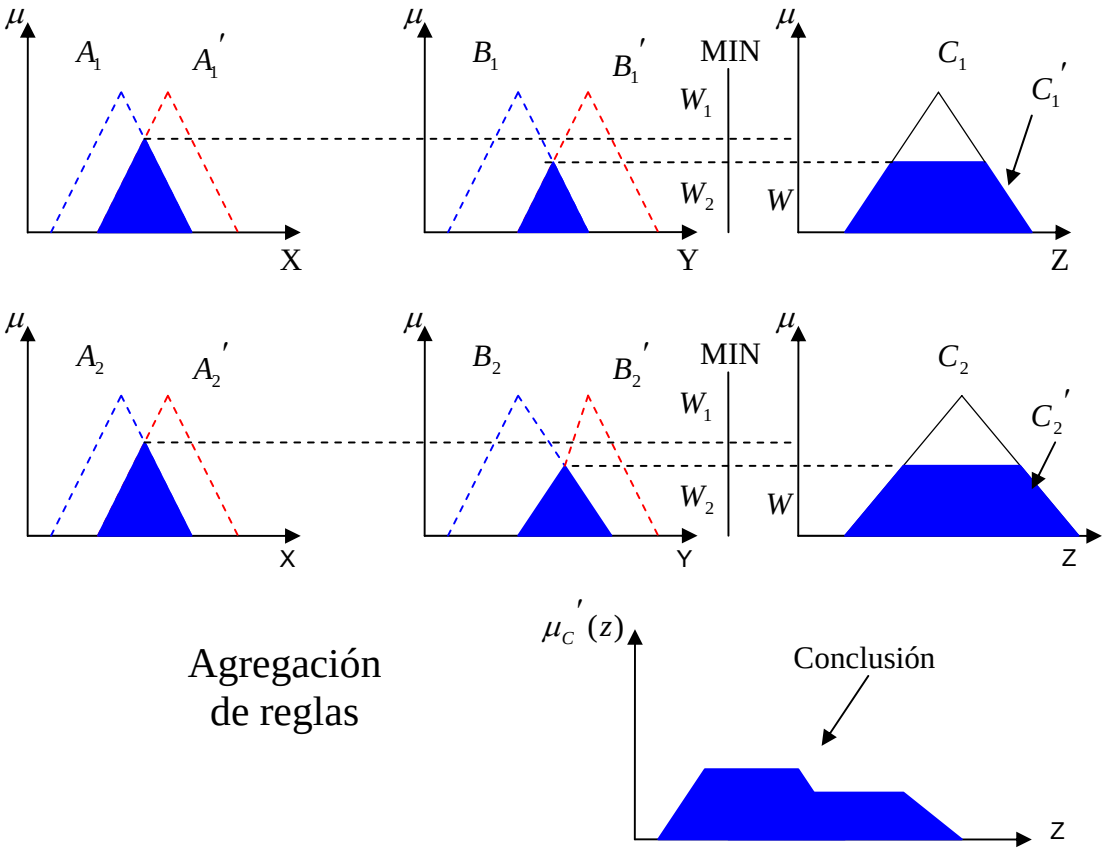


Fig. 18 Sistema de Múltiples Reglas y Múltiples Antecedentes

CAPITULO 3

SISTEMAS DE INFERENCIA BORROSOS

Los SIB permiten modelar el comportamiento de un sistema real, por medio de un proceso de inferencia que consiste en la extracción de conclusiones a partir de ciertas premisas y un conjunto de reglas borrosas, Los SIB son una plataforma popular de computo basada en los conceptos de La Teoría de los conjuntos borrosos, las reglas borrosas IF – THEN, y el razonamiento borroso. La estructura básica de un SIB consta de cinco componentes (ver Figura 19), que se deben implementar de forma ordenada y sucesiva:

La estructura básica de un SIB consiste de cinco componentes conceptuales,

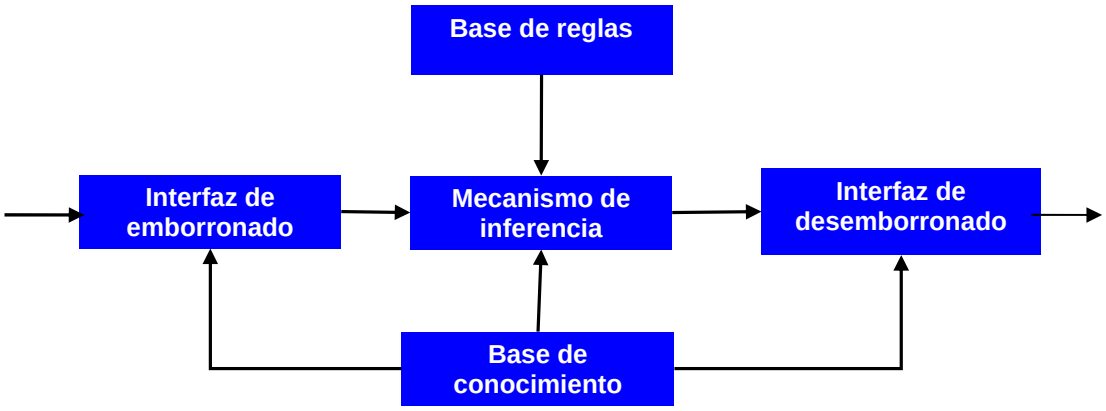


Fig. 19 Estructura de un SIB

3.1Emborronado. Convierte la entrada lógica o crisp en un valor borroso determinado por una FM **[10]**.

3.2 La Base De Datos. Define las FM que se utilizan en las reglas borrosas por medio de etiquetas lingüísticas, que son descritas por expertos en el proceso.

3.3 La Base De Reglas. Contiene el conjunto de acciones a realizar en función del estado. La composición de reglas correspondientes a la base es representada por medio de operaciones lógicas y asume la estructura ***I****F [condición] THEN [Acción****].***

3.4 Mecanismo De Inferencia. Realiza el procedimiento de inferencia sobre las reglas borrosas y los datos suministrados para producir una respuesta.

3.5 Desemborronado. Proceso que consiste en extraer u obtener un valor binario de un CB de salida como valor representativo, para ello se aplican los métodos de desemborronado que varían según el SIB empleado. Entre los métodos de Desemborronado más utilizados se encuentran **[11]**:

- **Centro de área**
- **Centro de sumas**
- **Centro de mayor área**
- **Primer máximo (Som), ultimo máximo (Lom) y media de máximos (Mom).**

Para ilustrar mejor los anteriores métodos se exhibe a continuación una figura con todos ellos (ver Figura 20).

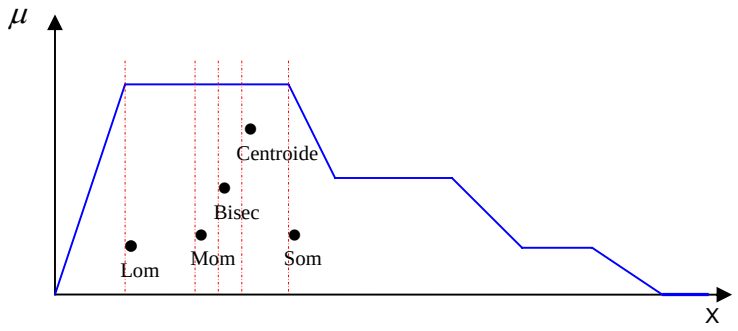


Fig. 20 Métodos de Desemborronado

3.6 Tipos de SIB. La importancia suministrada por La Teoría de los conjuntos borrosos generó un interés creciente en el desarrollo de aplicaciones basadas en sus fundamentos, hecho que provocó el nacimiento de los SIB, considerados las mayores aplicaciones de La LB. Hoy se cuentan tres tipos de SIB, que presentan diferencias puntuales, específicamente, en el tipo de reglas borrosas y los métodos de desemborronado utilizados.

3.6.1 SIB tipo Mandami. El SIB tipo Mandami fue propuesto en el año de 1975 por Ebrahim Mandami y un grupo de estudiantes del Colegio Queen Mary de Londres, como un primer intento para controlar un motor de vapor en combinación con una caldera, mediante un conjunto de reglas lingüísticas adquiridas de operadores humanos con experiencia, proceso del que se obtenía un CB de salida al que se aplicaba un método de desemborronado **[12]**.

Las reglas de un SIB tipo Mandami poseen la de la siguiente estructura:

$$si\ (x\ es\ A_1)\ Y\ (y\ es\ B_1)\ \ entonces\ z\ es\ C_1$$
$$si\ (x\ es\ A_2)\ Y\ (y\ es\ B_2)\ \ entonces\ z\ es\ C_2$$

3.6.1.1 Ventajas de un SIB tipo Mandami

- Facilidad par la derivación de las reglas
- Interpretabilidad de las reglas borrosas
- Conservan la esencia de La LB

3.6.1.2 Desventajas de un SIB tipo Mandami

- No garantizan la continuidad de la superficie de salida
- Menor eficiencia computacional

El esquema de inferencia para el SIB tipo Mandami se presenta en La Figura 21:

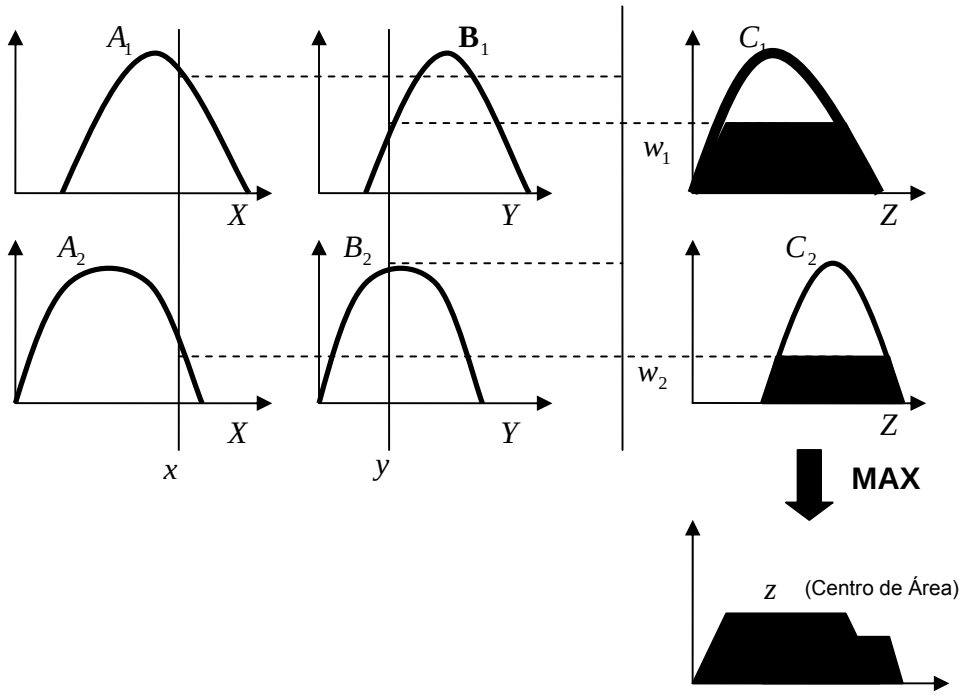


Fig. 21 Esquema de Inferencia Para un SIB Tipo Mandami

3.6.2 SIB tipo TSK. También conocido como el modelo de Sugeno, fue propuesto por Takagi, Sugeno y Kang en el año de 1985 como un esfuerzo para desarrollar un enfoque sistemático que genera las reglas borrosas a partir de un conjunto de datos entrada – salida **[13]**.

Una regla borrosa típica en un modelo de Sugeno tiene la forma:

$$si\ (x\ es\ A\ )\ Y\ (y\ es\ B\ )\ \ entonces\ x = f(x, y)$$

Donde $A\ y\ B$ son CB y $f(x, y)$ una función crisp; si la función $f(x, y)$ es polinomial se tiene un sistema de primer orden, en el caso de ser $f(x, y)$ una constante se tiene un sistema de orden cero.

3.6.2.1 Ventajas de un SIB tipo Sugeno

- Incrementan la precisión

- Mayor eficiencia computacional
- Facilidad para el análisis del sistema
- No necesitan interfaz de desemborronado
- Garantizan la continuidad en la superficie de salida

3.6.2.2 Desventajas de un SIB tipo Sugeno

- El consecuente es una fórmula matemática y no proporciona un marco natural para representar conocimiento humano.
- Limitan la representación de los principios de La LB.

El esquema de inferencia para el SIB tipo Sugeno se presenta en La Figura 22:

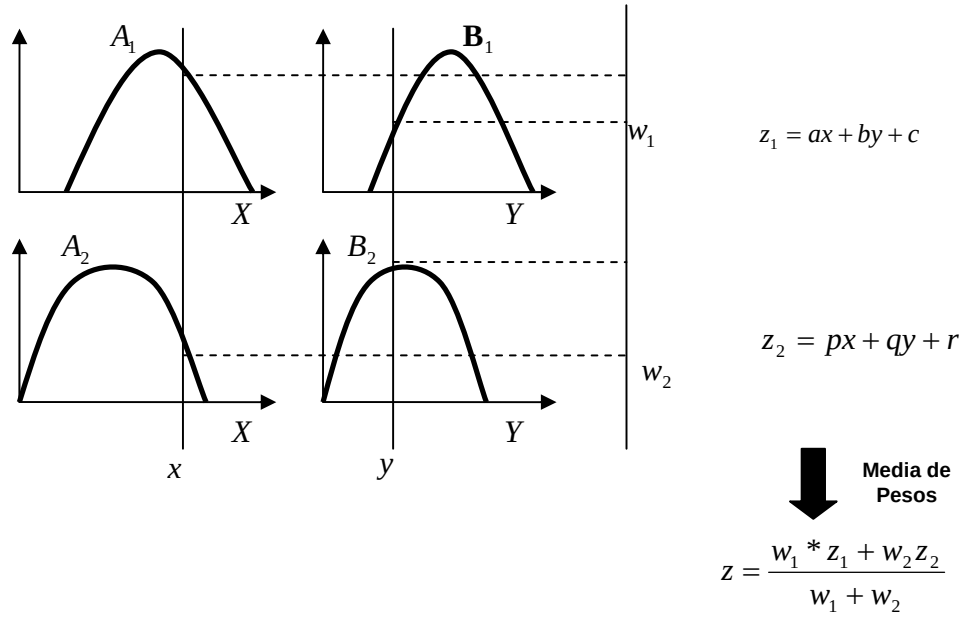


Fig. 22 Esquema de Inferencia Para un SIB Tipo Sugeno

3.6.3 SIB tipo Tsukamoto. En este modelo el consecuente de cada regla borrosa If – Then se representa por un CB con una FM monótona. Como consecuencia la salida de cada regla se define como un valor binario inducido por la fuerza del disparo de la regla. La salida total se toma como el peso promedio de salida de cada una de las reglas.
El modelo de Tsukamoto evita el tiempo de cálculo del desemborronado pero no es transparente como el de Mandami y el de Sugeno, por lo que es poco utilizado.

El esquema de inferencia para el SIB tipo Tsukamoto se presenta en La Figura 23:

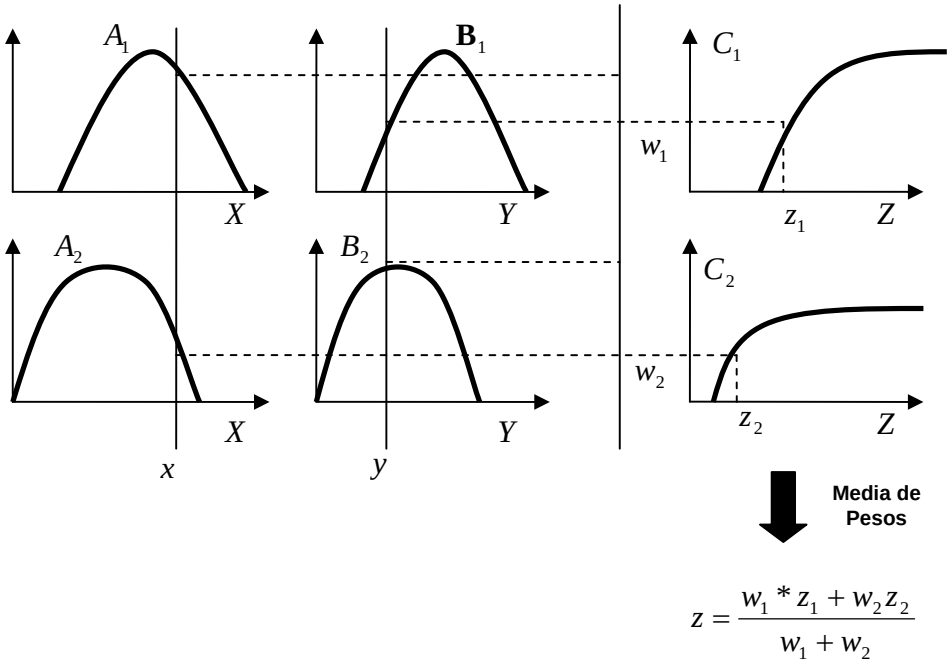


Fig. 23 Esquema de Inferencia Para un SIB Tipo Tsukamoto

CAPITULO 4

REDES NEURONALES ARTIFICIALES

Cuando la única herramienta de la que se dispone es un martillo, todos los problemas que uno encuentra tienden a parecer clavos
ANONIMO

4.1 Introducción. ¿Por qué no podemos construir un computador que piense? ¿Por qué no se puede esperar que máquinas capaces de realizar 100 millones de operaciones en coma flotante por segundo sean capaces de entender el significado de las formas en imágenes visuales, o incluso distinguir entres distintas clases de objetos similares? ¿Por qué no puede esa misma máquina *aprender* a partir de de la experiencia, en lugar de repetir, indefinidamente, un conjunto explícito de instrucciones generadas por un programador humano?

Estas son, solamente, unas pocas de las preguntas a las que se enfrentan los que diseñan computadores, los ingenieros y los programadores, todos los cuales se esfuerzan por crear sistemas de computadores "más inteligentes". La incapacidad de la generación actual de computadores para interpretar el mundo en general no indica, si embargo, que estas máquinas sean completamente inadecuadas. Hay muchas tareas que resultan, especialmente, adecuadas para ser resueltas mediante computadores convencionales: resolución de problemas matemáticos y científicos; creación, manipulación y mantenimiento de bases de datos; comunicaciones electrónicas; procesamiento de textos, gráficos y auto edición; incluso las sencillas funciones de control que dan inteligencia a los electrodomésticos y los hacen más fáciles de usar son gestionadas muy, eficientemente, por los computadores de hoy en día.

Por contraste, hay muchas aplicaciones que desearíamos automatizar, pero que no se han automatizado como consecuencia de las complejidades que implica la programación de un computador para llevar a cabo esas tareas. En un elevado porcentaje, los problemas no son irresolubles; lo que sucede es que son difíciles de resolver empleando sistemas de computadores secuenciales. Si la única herramienta de la que disponemos es un computador secuencial, entonces, de forma natural, intentamos resolver todos lo problemas en términos de algoritmos secuenciales. Hay problemas que no son adecuados para ser resueltos de esta manera, lo que hace que se inviertan grandes esfuerzos para desarrollar sofisticados algoritmos, e incluso que no se llegue a alcanzar una solución admisible **[14]**.

Esta dificultad de los sistemas de computo que trabajan bajo la filosofía de los sistemas secuenciales, difundidos por Von Neumann, ha hecho que un gran número de investigadores centre su atención en el desarrollo de nuevas formas para el tratamiento de la información, como el caso de Las Redes Neuronales Artificiales (RNA), que permitan solucionar problemas con alto grado de realismo tal como lo hace el cerebro humano.

En la tabla 3 se presenta un parangón entre las computadoras tipo Von Neumann y las RNA.

	Computadora	RNA
Procesador	• Complejo • Alta velocidad • Uno o pocos	• Simple • Baja velocidad • Un Gran número
Memoria	• Separada del procesador, localizada • No es direccionable por el contenido	• Integrada en el procesador, distribuida • Direccionable por el contenido
Computación	• Centralizado • Secuencial • Programas almacenados	• Distribuido paralelo • Auto - aprendizaje
Confiabilidad	• Muy vulnerable	• Robusta
Punto Fuerte	• Manipulación numérica y simbólica	• Problemas de percepción
Ambiente Operativo	• Bien definido • Bien limitado	• Pobremente definido • Ilimitado

Tabla 3 Parangón entre Computadoras Von Neumann y RNA

El cerebro humano, constituye una computadora notable, capaz de interpretar información imprecisa suministrada por los sentidos a un ritmo, increíblemente, veloz. Logra discernir un susurro en una sala ruidosa, un rostro en un callejón mal iluminado y leer entre líneas un discurso; lo más impresionante es que el cerebro aprende, sin instrucciones explícitas de ninguna clase, a crear las representaciones internas que hacen posibles estas habilidades.

4.1.1 Herramientas de extracción de información. El procesamiento de información de carácter redundante, imprecisa y distorsionada posee un papel primordial para la resolución de problemas reales de clasificación o de predicción en muchas áreas científicas.

Una de las metodologías con un mayor auge en la última década son los modelos de RNA **[15]**, que en esencia son estructuras formales de carácter matemático y estadístico con la propiedad del *aprendizaje*, es decir, la adquisición de conocimientos que en la mayoría de los casos se da a partir de ejemplos.

Este aprendizaje se produce mediante un estilo de computación[3] denominado, en *paralelo,* que intenta simular algunas de las capacidades que posee nuestro cerebro, (ver

[3] Los elementos de un modelo de computación son: Almacenamiento, Transmisión y Procesamiento.

Figura 23). Por esta razón se las definen como RNA para distinguirlas de los modelos biológicos.

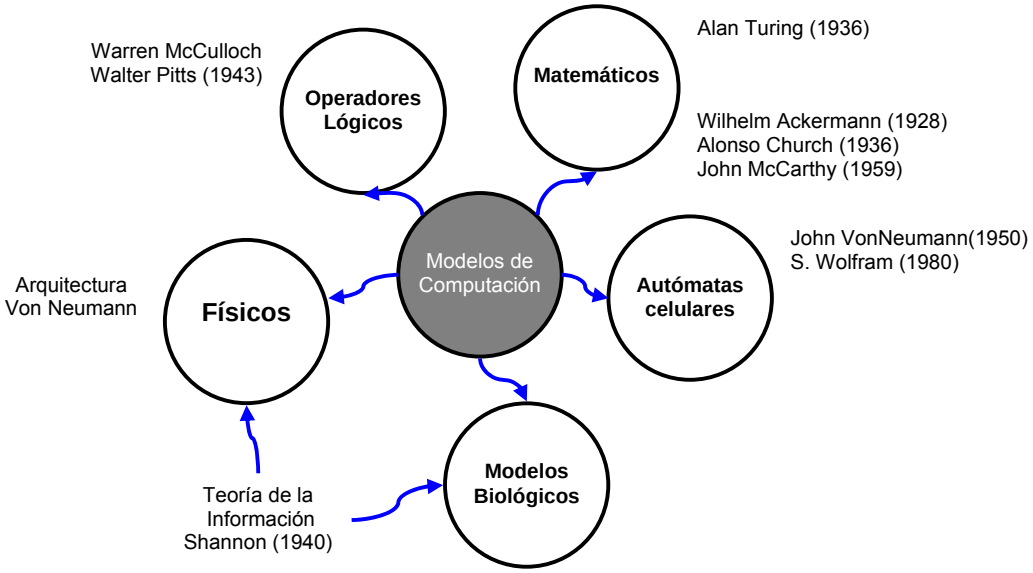


Fig. 23 Modelos Computacionales

Los tres elementos clave de los sistemas biológicos que pretenden emular los artificiales son: el *procesamiento en paralelo*, la *memoria distribuida* y la *adaptabilidad*.

4.1.2 Origen del paradigma de computación conexionista. La Inteligencia Artificial (IA), entendida como el modelado y la simulación de las actividades cognitivas complejas (percepción, memoria y solución de problemas) que caracterizan a los organismos avanzados y en particular a los seres humanos, se separó casi desde su inicio en dos ramas bien definidas.

- Modelar la actividad racional mediante sistemas formales de reglas y manipulación simbólica (por medio de sistemas lógicos), que se constituyó como la rama más conocida de La IA, que se denomina simbólico - deductiva (se postulan una serie de reglas y el sistema resuelve los problemas al realizar deducciones sobre las reglas existentes).
- Desarrollo de modelos computacionales inspirados en las redes neuronales biológicas, a los que se denomino inductivos o sub simbólicos, debido a que extraen la información necesaria para resolver un problema de un conjunto de ejemplos, sin necesidad de indicarle las reglas necesarias para resolverlo.

Es viable situar el origen de los modelos conexionistas con la definición de neurona, dada por McCulloch y Pitts en 1943 **[16]**, como un dispositivo binario con varias entradas y salidas. El psicólogo, D. O. Hebb**,** introdujo en 1949 dos ideas fundamentales que influyeron de manera decisiva en el campo de Las RNA: la primera define que una percepción o concepto se representan en el cerebro por medio de un conjunto de

neuronas activas simultáneamente, la segunda aclara que la memoria se localiza en las conexiones entre las neuronas (sinápsis).

Las hipótesis de Hebb, basadas en investigaciones psico - fisiológicas, presentan de manera intuitiva el modo en que las neuronas memorizan información y se plasman en la famosa regla de aprendizaje de Hebb, conocida también como regla del producto, la cual indica que las conexiones entre dos neuronas se refuerzan si ambas son activadas.

El diseño de las RNA, también llamadas sistemas de procesamiento en paralelo (*Parallel distributed processing systems* (PDPs)) o sistemas conexionistas (*Connectionist systems*), incorporan características biológicas generando un claro paralelismo entre estos y el modelo neuronal biológico. Así conceptos, como: *dendritas*, *axón*, *sinapsis* y cuerpo o *soma* que son de uso corriente en el entorno biológico, toman un papel relevante en los modelos matemáticos que replican la forma de transmisión de la señal eléctrica que por ellas circula. En 1901 *Santiago Ramón y Cajal* **[17],** postuló que la dirección de la transmisión de la información es determinada por la naturaleza en forma de red de las células nerviosas, a partir de los conocimientos sobre la naturaleza eléctrica de los impulsos nerviosos y de la velocidad de su transmisión.

4.2 Panorama Histórico. En 1956 se organizó en Dartmouth la primera conferencia sobre IA. En ella se discutió el uso potencial de las computadoras para simular "todos los aspectos del aprendizaje o cualquier otra característica de la inteligencia" y se presentó la primera simulación de una RNA, aunque todavía no se sabían interpretar los datos resultantes.

- En 1958, Rosemblatt publica los resultados de un ambicioso proyecto de investigación, denominado el Perceptrón.
- En 1960, Widrow publica una teoría sobre adaptación neuronal y los modelos inspirados en ella, el Adaline (Adaptative Linear Neuron) y el Madaline (Multiple Adaline).
- 1969, Minsky y Paper realizan una seria crítica del Perceptrón, revelando serias limitaciones como su incapacidad para representar la función XOR, debido a su naturaleza lineal.
- Anderson, Silverstein, Ritz & Jomnes (1977), estudian y desarrollan modelos de memorias asociativas.
- Kohonen (1978) continua el trabajo de Anderson y desarrolla modelos de aprendizaje competitivo basados en el principio de inhibición lateral.
- Grossberg (1978) realizó un importante trabajo teórico - matemático basado en principios fisiológicos; aportó importantes innovaciones con su modelo ART (Adaptative Resonance Theory) y, junto a Cohen, elabora un importante teorema sobre la estabilidad de las RNA recurrentes en términos de una función de energía.
- Rumelhart, McClelland & Hinton crean el grupo PDP, como resultado de los trabajos de este salieron los manuales (Rumelhart & McClelland 1986 y 1988) con más influencia desde la crítica de Minsky y Papert.
- Hopfield (1982) elabora un modelo de RNA consistente en unidades de proceso interconectadas que alcanzan mínimos energéticos, aplicando los principios de estabilidad desarrollados por Grossberg.

- Otros desarrollos destacables de esta década son la máquina de Boltzmann (Hinton & Sejnowski 1986) y los modelos BAM (Kosko 1987).

4.3 Definiciones De RNA. *Sistema de computación hecho por un gran número de elementos simples, elementos de proceso muy interconectados, los cuales procesan información por medio de su estado dinámico como respuestas a entradas externas [Hecht-Niesen 88].*

Las RNA son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico [Kohonen 88].

4.4 Ventajas De Las RNA. Debido a su constitución y a sus fundamentos, Las RNA presentan un gran número de características semejantes a las del cerebro. Por ejemplo: son capaces de aprender de la experiencia, de generalizar de casos anteriores a nuevos casos, de abstraer características esenciales a partir de entradas que representan información irrelevante, etc. Estas ventajas incluyen:

- *Aprendizaje adaptativo*
- *Autoorganización*
- *Tolerancia a fallos*
- Operación en tiempo real
- *Fácil inserción dentro de la tecnología existente*

A continuación, se detallan algunos principios computacionales[4], relacionados con pormenores de la organización del cerebro, que sirven de inspiración para el diseño de los modelos artificiales, ver tabla 4.

1	*Paralelismo masivo*: las operaciones de cálculo se realizan al mismo tiempo en diferentes regiones del *córtex* y en cada región funcional del mismo, el cálculo lo realizan simultáneamente todas las neuronas.
2	Alto grado de *complejidad en las conexiones* definidas entre las neuronas.
3	*Habilidad para el aprendizaje*: Existe una adaptación constante de las neuronas a partir de la experiencia acumulada.
4	*Estados binarios y variables continuas*: Las neuronas poseen dos estados, activado o desactivado y la información continua está codificada por la frecuencia de pulsos. (Pulsos por unidad de tiempo)
5	*Numerosos* tipos de neuronas y de señales.
6	Interacción entre neuronas de *carácter no lineal*.
7	El cerebro se organiza *descomponiendo* las conexiones en subredes.
8	Asociado al anterior aspecto cada subred tiene una *función* específica.

Tabla 4 Principios Computacionales

[4] Véase el paralelismo entre computación, aprendizaje y topologías neuronales en, Massón, E.; Wang, Y-J. (1990). **Introduction to computation and learning in artificial neural networks**, *European Journal of Operational Research*, 47, pp. 1-28.

4.5 Fundamentos de las RNA

4.5.1 El Modelo biológico. La teoría y modelado de Las RNA se inspira en la estructura y funcionamiento de los sistemas nerviosos, donde la neurona es el elemento fundamental. Existen neuronas de diferentes formas, tamaños y longitudes. Estos atributos son importantes para determinar la función y utilidad de la neurona. La clasificación de estas células en tipos estándar ha sido realizada por neuroanatomistas.

4.5.2 Estructura de las neuronas biológicas. El sistema nervioso humano, incluido el cerebro, consiste en la interconexión de millones y millones de unidades celulares[5] llamadas neuronas, sólo la corteza cerebral humana que es una capa larga y plana de neuronas de alrededor de 2 a 3 milímetros de ancho con un área superficial de aproximadamente $2200cm^2$, contiene alrededor de cien mil millones (10^{11}) de neuronas.

En cuanto a su fisiología, las neuronas constan de un cuerpo celular o soma, más o menos esférico de 5 a 10 micras de diámetro, donde se aloja el núcleo de la célula. Del cuerpo de la célula se desprenden ramificaciones de diversas fibras conocidas como dendritas y también una fibra más larga denominada axón. En su extremo, el axón se ramifica en filamentos y sub-filamentos mediante los que establece conexión con las dendritas y los cuerpos de las células de otras neuronas; estas conexiones son conocidas como sinapsis (ver Figura 24).

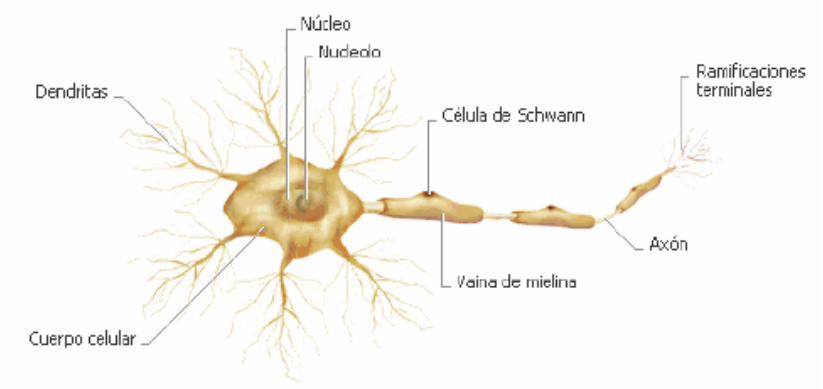


Fig. 24 Neurona Biológica

Las estructuras neuronales cambian durante toda la vida, estos cambios consisten en el refuerzo o debilitamiento de las uniones sinápticas; por ejemplo se cree que nuevas memorias son formadas por la modificación de esta intensidad entre sinápsis, así el proceso de recordar el rostro de un nuevo amigo, consiste en alterar varias sinápsis.

4.5.3 Naturaleza bioeléctrica de la neurona. Se utilizan señales con dos tipos de naturaleza: *Eléctrica y química.* La señal generada por la neurona transportada a lo largo

[5] Las *células nerviosas* son sistemas complejos con propiedades de autoorganización, que realizan el almacenamiento de la información en el proceso de sinápsis.

del axón es un impulso eléctrico, mientras que la señal que se transmite entre los terminales axónicos de una neurona y las dendritas de las neuronas siguientes es de origen químico; concretamente, se realiza mediante moléculas de sustancias transmisoras (*neurotransmisores*) que fluyen a través de unos contactos especiales, llamados *sinapsis,* que tienen la función de receptor y están localizados entre los terminales axónicos y las dendritas de la neurona siguiente (espacio sináptico, entre 50 y 200 angtroms).

La generación de las señales eléctricas está relacionada con la composición de la membrana celular. Existen procesos complejos relacionados con la generación de dichas señales; sin embargo, se simplifican del siguiente modo: la neurona, como todas las células, es capaz de mantener en su interior un líquido cuya composición difiere marcadamente de la composición del líquido exterior. La diferencia más notable se da en relación con la concentración de los iones *sodio* y *potasio.* El medio externo es 10 veces más rico en sodio que el interno, mientras que el medio interno es 10 veces más rico en potasio que el externo. Esta diferencia de concentración en iones sodio y potasio a cada lado de la membrana produce una diferencia de potencial de aproximadamente 70 milivoltios, negativa en el interior de la célula. Es lo que se llama *potencial de reposo* de la célula nerviosa ver Figura 25.

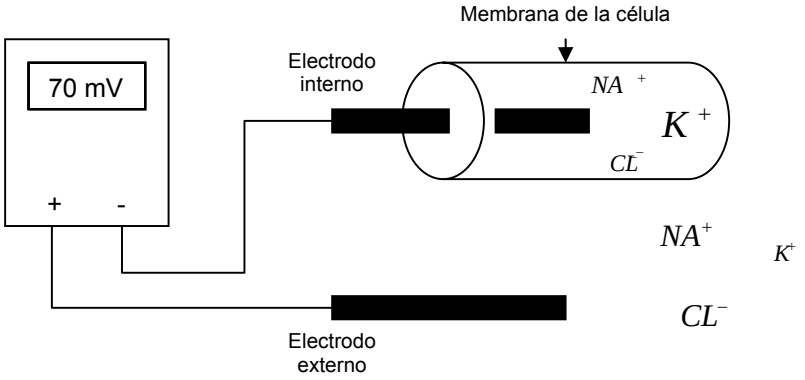


Fig. 25 Potencial de Reposo de Una Neurona

4.5.4 Elementos de una RNA. Los modelos neuronales asumen muchas simplificaciones del modelo biológico con el fin de facilitar su desarrollo matemático[6], así en esta línea, el primer modelo artificial fue diseñado por McCulloch-Pitts (1943), el cual utilizaba unidades de procesamiento denominadas *neuronas* que poseían dos estados discretos. Asociados a cada uno de ellos, se conseguía un *output* que se transmitía a lo largo de la estructura vinculada a la RNA, pero con la limitación que sólo permitían computar funciones booleanas. La Figura 26 compara una neurona biológica con el modelo matemático homólogo.

[6] Para conocer los fundamentos matemáticos y algebraicos de la metodología neuronal, véase Ellacott, S.; Bose, D. (1996). **Neural Networks: Deterministic Methods of Analysis**, International Thomson Computer Press.

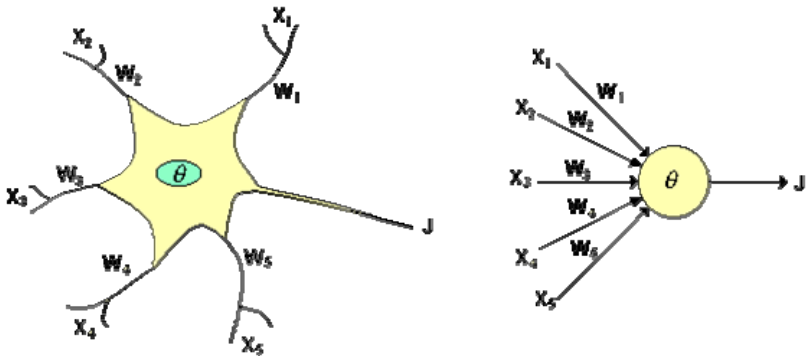


Fig. 26 Paralelismo Entre el Modelo Biológico y el Modelo McCulloch y Pitts

El modelo de McCulloch-Pitts posee las siguientes hipótesis. En primer lugar, el estado de una neurona en el tiempo, "$t+1$", depende sólo del estado que poseía en el período anterior, "t". En segundo lugar, una neurona estará activada o no si supera un umbral," θ", y en último lugar, se asume la sincronía entre *input* y *output* aún conociendo la asincronía en la neurona biológica.

La formalización del diseño del modelo de McCulloch-Pitts consiste en definir el estado de la entrada, "x_t" y luego definir la salida en el momento *t*, "y_t". La expresión que describe su funcionamiento es:

$$y_i^{t+1} = f\left[\sum_{j=1}^{m} w_{ij} x_j^t - \theta_i\right] = f(a) \tag{60}$$

$$f(a) = \begin{cases} 1 & si\ a \geq 0 \\ 0 & en\ otro\ caso \end{cases}$$

Donde la función $f(.)$, está descrita por $n+1$ parámetros, n - pesos $(w_1, w_2, ..., w_n)$ O eficacia sináptica y el umbral o tendencia $(\theta, "bias")$.

Las ventajas inherentes que poseen dichos modelos son: el aprendizaje, la generalización y la robustez frente al "ruido" de los datos. Es decir, la adquisición de la información es posible a partir del aprendizaje de la estructura interna de los datos, de forma que son las propias conexiones o pesos entre los diferentes elementos que constituyen una RNA donde se retiene el conocimiento. Es de gran importancia notar que no existe a priori una definición explicita de la forma del conocimiento, el propio algoritmo iterativo de estimación de los parámetros desconocidos o pesos, se encarga de extraer la presencia de regularidades en los datos[7].

[7] De ahí la expresión de *Aproximador Universal*, véase White, H. (1992). **Artificial Neural Networks. Aproximation and Learning Theory**, pp. 13-28, Blackwell Publishers.

4.5.5 Estado de activación. Adicionalmente, al conjunto de unidades, la representación necesita los estados del sistema en un tiempo t, esto se especifica mediante un vector de N números reales $A(t)$, que representa el estado de activación del conjunto de unidades de procesamiento. Cada elemento del vector representa la activación de una unidad en el tiempo t la activación de una unidad U_i en el tiempo t se designa por $a_i(t)$; es decir:

$$A(t) = \left(a_1(t), a_2(t),..., a_i(t),..., a_N(t)\right) \tag{61}$$

El procesamiento que realiza la RNA es visto como la evolución de un patrón de activación en el conjunto de unidades que la componen a través del tiempo.

Todas las neuronas que componen la RNA se hallan en cierto estado. En una visión simplificada, se dice que hay dos posibles estados, *reposo y excitado,* que se denominan *estados de activación*, y a cada uno de los cuales se le asigna un valor.

4.5.6 Función de salida o de transferencia. Entre las unidades o neuronas que forman una RNA existe un conjunto de conexiones que unen unas con otras. Cada unidad transmite señales a aquellas que estén conectadas con su salida. Asociada con cada unidad U_i hay una función de salida $f_i(a_i(t))$, que transforma el estado actual de activación $a_i(t)$ en una señal de salida $y_i(t)$; es decir:

$$y_i(t) = f_i(a_i(t)) \tag{62}$$

El vector que contiene las salidas de todas las neuronas en un instante t es:

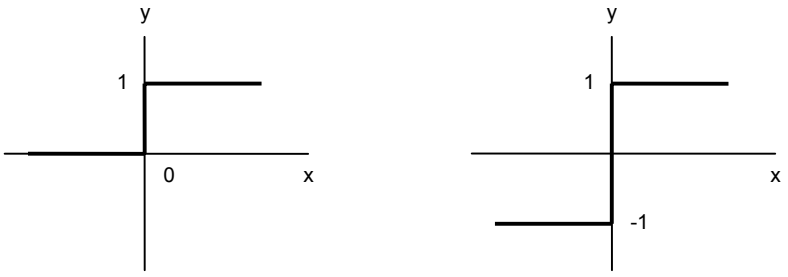
$$Y(t) = \left(f_1(a_1(t)), f_2(a_2(t)),..., f_i(a_i(t)),..., f_N(a_N(t))\right) \tag{63}$$

En algunos modelos, esta salida es igual al nivel de activación de la unidad, en cuyo caso la función f_i es la función identidad $f_i(a_i(t)) = a_i(t)$. A menudo f_i es de tipo sigmoidal, y suele ser la misma para todas las unidades.

Existen cuatro funciones de transferencia típicas que determinan los distintos tipos de neuronas:

- Función escalón
- Función lineal y mixta
- Sigmoidal
- Función gaussiana

4.5.6.1 Neurona de función escalón. La forma más fácil de definir la activación de una neurona es considerar que binaria. La función de transferencia escalón se asocia a neuronas binarias en las cuales, cuando la suma de las entradas es mayor o igual que el umbral de la neurona, la activación es 1; si es menor, la activación es 0 (o -1). Por otro lado, las RNA formadas por este tipo de neuronas son fáciles de implementar en hardware, pero sus capacidades son limitadas (ver Figura 27).



$$f(x)\begin{cases}1 \; si \; x \; es \geq 0 \\ 0 \; si \; x < 0\end{cases} \qquad f(x)\begin{cases}1 \; si \; x \; es \geq 0 \\ -1 \; si \; x < 0\end{cases}$$

En ambos casos se ha tomado que el umbral es cero; en caso de que no lo fuera, el escalón quedaría desplazado.

Fig. 27 Función de Transferencia Escalón

4.5.6.2 Neuronas de función lineal y mixta. La función lineal o identidad responde a la expresión $f(x) = x$. En las neuronas con función mixta, si la suma de las señales de entrada es menor que un límite inferior, la activación se define como 0 (o -1). Si dicha suma es mayor o igual que el límite superior, entonces la activación es 1. Si la suma de entrada está comprendida entre ambos limites, superior e inferior, entonces la activación se define como una función lineal de la suma de las señales de entrada.

Es posible representar la funciones de activación mixtas como se indica en la Figura 28, se toma el límite superior de la suma de todas las entradas de activación que afectan a la neurona durante el ciclo de operación (x) como c y el límite inferior como $-c$, y es la salida de activación de la neurona.

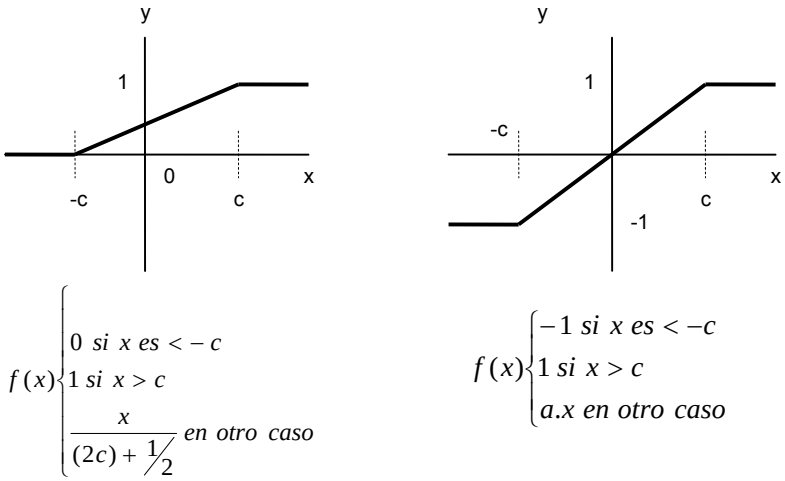


Fig. 28 Función de Activación Mixta

4.5.6.3 Neuronas de función continua. Una función definida en un intervalo de posibles valores de entrada, con un incremento monotónico y que tenga tanto límites superior como limite inferior, podrá realizar la función de activación o de transferencia de forma satisfactoria.

Sin embargo, la importancia de la función sigmoidal es que su derivada es positiva y cercana a cero para los valores grandes positivos o negativos; además, toma su valor máximo cuando x es 0. Esto permite usar reglas de aprendizaje definidas para las funciones escalón, con la ventaja, respecto a esta función, que la derivada está definida en todo el intervalo. La función escalón no podía definir la derivada en el punto de transición y esto no ayuda a los métodos de aprendizaje en los cuales se usan derivadas (ver Figura 29).

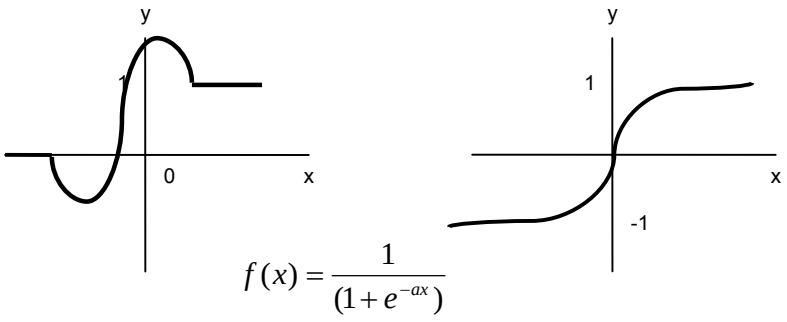


Fig. 29 Función de Activación Sigmoidal

4.5.6.4 Función de transferencia Gaussiana. Los centros y el ancho de estas funciones pueden ser adaptados, hecho que las hace más adaptativas que las funciones sigmoidales. Mapeos que suelen requerir dos niveles ocultos (neuronas de la RNA que se

encuentran entre las de entrada y las de salida) utilizando neuronas con funciones de transferencia sigmoidales; algunas veces se pueden realizar con un solo nivel en redes con neuronas de función gaussiana (ver Figura 30).

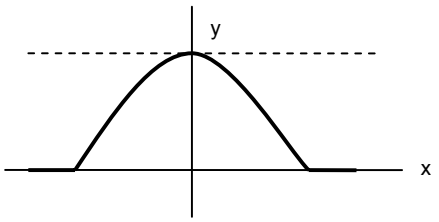


Fig. 30 Función de Transferencia Gaussiana

4.5.7 Conexiones entre neuronas. Las conexiones que unen a las neuronas que forman una RNA tiene asociado un peso, el que hace que la RNA adquiera conocimiento. Al considerar y_i como el valor de salida de una neurona i en un instante dado. Una neurona recibe un conjunto de señales que le presentan información del estado de activación de todas las neuronas con las que se encuentra conectada. Cada conexión (sinapsis) entre la neurona i y la neurona j está ponderada por un peso w_{ji}. Normalmente, se considera que el efecto de cada señal es aditivo, de tal forma que la entrada neta que recibe una neurona (potencial post sináptico) net_j es la suma del producto de cada señal individual por el valor de la sinapsis que conecta ambas neuronas:

$$net_j = \sum_i^N w_{ji} * y_i \tag{64}$$

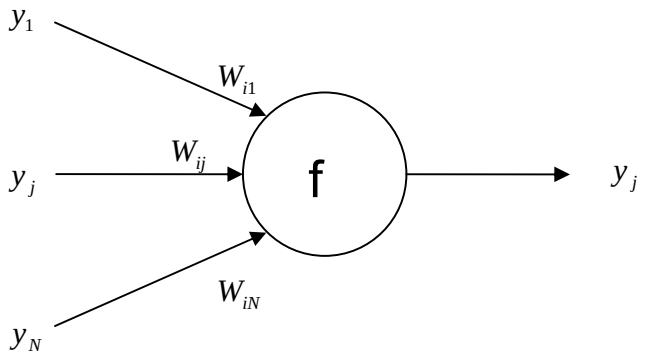
Esta regla muestra el procedimiento a seguir para combinar los valores de entrada a una unidad con los pesos de las conexiones que llegan esa unidad y es conocida como *regla de propagación*.

4.5.8 Función o regla de activación. Así como es necesaria una regla que combine las entradas a una neurona con los pesos de las conexiones, también se requiere una regla que combine las entradas con el estado actual de la neurona para producir un nuevo estado de activación. Esta función F produce un nuevo estado de activación en una neurona a partir del estado (a_i) que existía y la combinación de las entradas con los pesos de las conexiones (Net_i).

Dado el estado de activación $a_i(t)$ de la unidad U_i y la entrada total que llega a ella, (Net_i), el estado de activación siguiente, $a_i(t+1)$, se obtiene aplicando una función F llamada *función de activación.*

$$a_i(t+1) = F(a_i(t), Net_i) \tag{65}$$

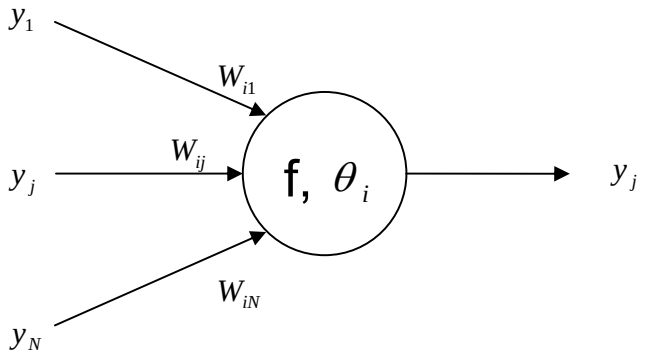
En al mayoría de los casos, F es *la función identidad*, por lo que el estado de activación de una neurona en $t+1$ coincidirá con el Net de la misma en t. En este caso, el parámetro que se le pasa a la función de salida, f, de la neurona será directamente el Net. El estado de activación anterior no se tiene en cuenta. Según lo anterior, la salida de una neurona $i(y_i)$ quedará de la siguiente manera (ver Figura 31):



$$y_i(t+1) = f(Net_i) = f\left(\sum_{j=1}^{N} w_{ij} y_j(t)\right) \tag{66}$$

Fig. 31 Salida de una Neurona Artificial

Por tanto, y en lo sucesivo, se considerara la función f, que será denominada de transferencia o de activación. Además, la función de activación no está centrada en el origen del eje que representa el valor de la entrada neta, sino que existe cierto desplazamiento debido a las características internas de la propia neurona y que no es igual en todas ellas. Este valor se denota como θ_i y representa el umbral de activación de la neurona i (ver Figura 32).



$$y_i(t+1) = f(Net_i - \theta) = f\left(\sum_{j=1}^{N} w_{ij} y_j(t) - \theta\right) \tag{67}$$

Fig. 32 Salida de una Neurona Artificial con Umbral

4.5.9 Regla de aprendizaje. Existen muchas definiciones del concepto general de aprendizaje, una de ellas es: *La modificación del comportamiento inducido por la interacción con el entorno y como resultado de experiencias, conducente al establecimiento de nuevos modelos de respuesta o estímulos externos.* Esta definición fue enunciada muchos años antes de que surgieran las RNA, sin embargo puede ser aplicada también a los procesos de aprendizaje de estos sistemas.

Biológicamente, se acepta que la información memorizada en el cerebro está relacionada con los valores sinápticos de las conexiones entre las neuronas que con ellas mismas; es decir, el conocimiento se encuentra en las sinapsis. En el caso de Las RNA, se puede considerar que el conocimiento se encuentra en los *pesos* de las conexiones entre neuronas. Todo proceso de aprendizaje implica cierto número de cambios en estas conexiones. En realidad, puede decirse que se *aprende* modificando los valores de los pesos de la RNA.

4.5.10 Estructura de una RNA. Los componentes más importantes de una RNA son:

- Unidades de procesamiento (la neurona artificial)
- Estado de activación de cada neurona
- Patrón de conectividad entre neuronas
- Regla de propagación
- Función de transferencia
- Regla de activación
- Regla de aprendizaje

Centrados en las características de cada nodo de la RNA (micro estructura), la forma como está organizada dicha RNA (meso-estructura) se presenta en función de:

- Número de niveles o capas
- Número de neuronas por nivel
- Patrones de conexión
- Flujo de información

4.5.11 Niveles o capas de neuronas. La distribución de neuronas dentro de la RNA se realiza formando niveles o capas de un número determinado de neuronas cada una. A partir de su situación dentro de la RNA, se pueden distinguir tres tipos de capas:

- Entrada: Es la capa que recibe directamente la información proveniente de las fuentes externas de la RNA.
- Ocultas: Son internas a la RNA y no tienen contacto directo con el entorno exterior. El número de niveles ocultos puede estar entre cero y un número elevado. Las neuronas de las capas ocultas pueden estar interconectadas de distintas maneras, lo que determina, junto con su número, las distintas tipologías de RNA.
- Salida: Transfieren información de la RNA hacia el exterior.

En La Figura 33 se muestra el esquema de la estructura de una posible RNA multicapa en la que cada nodo o neurona está conectada con neuronas de un nivel superior. Nótese

que existen más conexiones que nodos. En este sentido, se dice que una RNA es *totalmente conectada* si todas las salidas de un nivel llegan a todos y cada uno de los nodos del nivel siguiente.

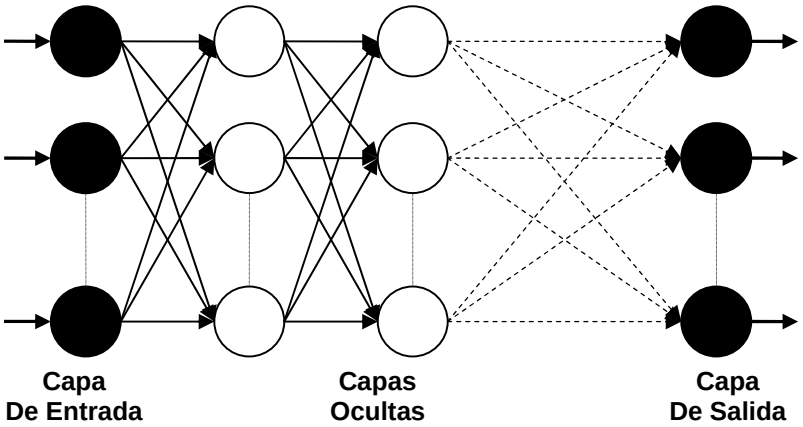


Fig. 33 Estructura de una Red Multinivel con conexiones Hacia delante

4.5.12 Formas de conexión entre neuronas. La conectividad entre los nodos de un RNA está relacionada con la forma en que las salidas de las neuronas están canalizadas para convertirse en entradas de otras neuronas. La señal de salida de un nodo puede ser una entrada de otro elemento de proceso, o incluso ser una entrada de sí mismo (conexión autorrecurente).

4.5.13 Características de Las RNA. La gran variedad de modelos de RNA existentes, en la actualidad, obliga en cierta medida a la realización de clasificaciones o taxonomías (ver Figura 34). Existen cuatro aspectos que caracterizan una RNA: Su topología, el mecanismo de aprendizaje, tipo de asociación realizada entre la información de entrada y de salida y la forma de representación de esas informaciones.

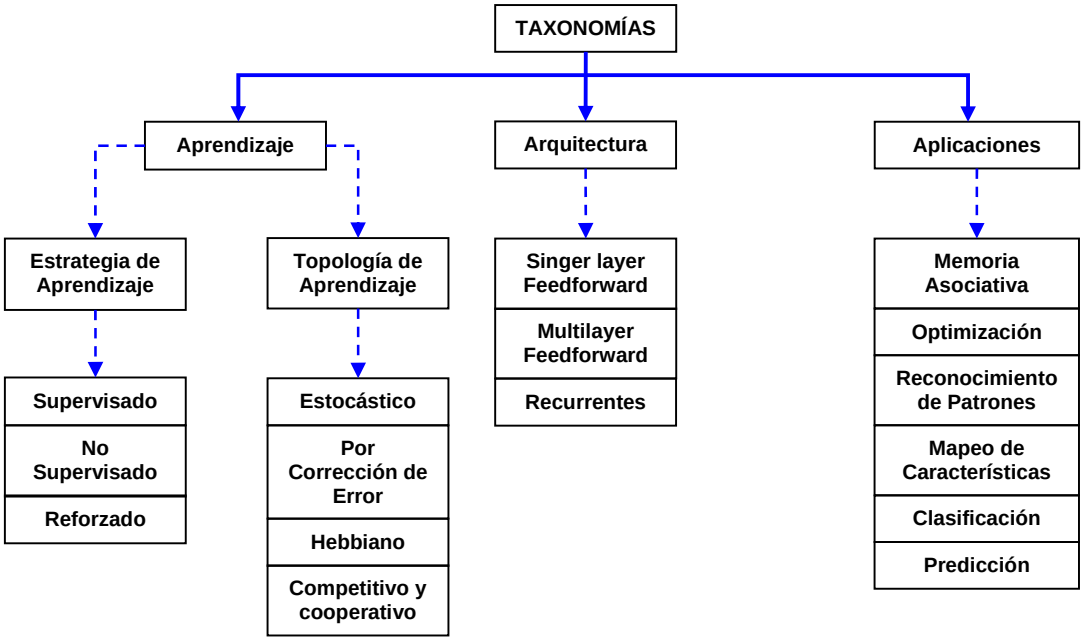


Fig. 34 Taxonomía de Las RNA

4.5.14 Topología de Las RNA. La topología o arquitectura de las RNA consiste en la organización y disposición de las neuronas en la RNA formando *capas* o agrupaciones de neuronas más o menos alejadas de la entrada y salida de la RNA. En este sentido los parámetros a tener en cuenta son:

- Número de capas
- Número de neuronas por capa
- Grado de conectividad
- Tipo de conexiones entre neuronas

Cuando se realiza una clasificación de las RNA en términos topológicos se distingue entre las RNA con una sola capa o nivel de neuronas y las RNA con múltiples capas (ver Figura 35).

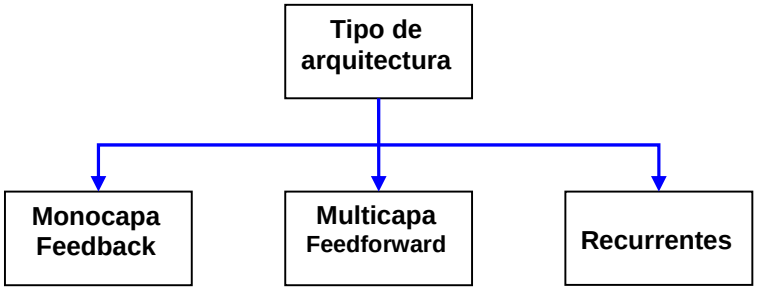


Fig. 35 Clasificación de las RNA Según la Topología

4.5.14.1 RNA monocapa (1 capa). En las RNA monocapa, coma la de Hopfield y la Brain-State-In-a-Box, se establecen conexiones laterales entre las neuronas que pertenecen a la única capa que constituye La RNA. También pueden existir *conexiones auto-recurrentes* (salida de una neurona conectada a su propia entrada), aunque en algún modelo, como el de Hopfield esta recurrencia no se utiliza.

4.5.14.2 RNA multicapa. Las RNA multicapa son aquellas que disponen de conjuntos de neuronas agrupadas en varios niveles o capas. En estos casos, una forma para distinguir la capa a la que pertenece una neurona, consistiría en fijarse en el origen de las señales que recibe a la entrada y el destino de la señal de salida.

Estas dos posibilidades permiten distinguir entre dos tipos de RNA con múltiples capas: las RNA con conexiones hacia delante o *feedforward*, y las RNA que disponen de conexiones tanto hacia delante como hacia atrás o *feedforward/feedback.*

4.5.14.2.1 RNA con conexiones hacia delante. En las RNA feedforward, todas las señales neuronales se propagan hacia delante a través de las capas de la RNA. No existen conexiones hacia atrás (ninguna salida de neuronas de una capa i se aplica a la entrada de neuronas de capas $i-1, i-2$), y tampoco auto-recurrentes (salida de una neurona aplicada a su propia entrada), ni laterales (salida de una neurona aplicada a la entrada de neurona de la misma capa).

4.5.14.2.2 RNA con conexiones hacia delante y atrás. En este tipo de RNA circula información tanto hacia delante *(forward*) como hacia atrás (backward) durante el funcionamiento de la RNA. Para que esto sea posible, existen conexiones *feedforward* y conexiones *feedback* entre las neuronas.

En general, excepto el caso particular de las RNA Cognitron y Neocognitron, suelen ser bicapa (dos capas), existiendo por tanto dos conjuntos de pesos: los correspondientes a las conexiones *feedforward* de la primera capa (capa de entrada) hacia la segunda (capa de salida) y los de las conexiones *feedback* de la segunda a la primera.

4.5.14.2.3 RNA recurrentes. Las RNA recurrentes se caracterizan porque se crean bucles en las neuronas de la RNA mediante el uso de las llamadas conexiones recurrentes, apareciendo conexiones de una neurona con ella misma, conexiones entre neuronas de una misma capa o conexiones de las neuronas de una capa a la capa anterior como se ilustra en la Figura 36.

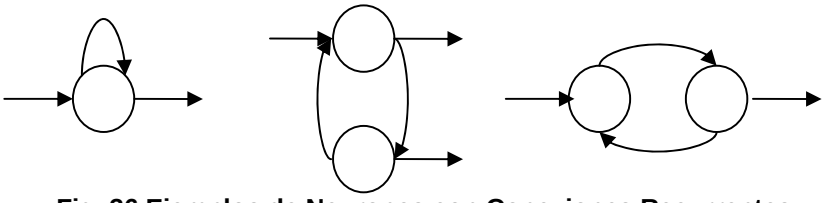


Fig. 36 Ejemplos de Neuronas con Conexiones Recurrentes

La consideración de las conexiones recurrentes en una RNA implica, un aumento del número de pesos o parámetros ajustables en al RNA, que permiten un aumento en la capacidad de representación, pues en las RNA la información se representa de manera

distribuida en los pesos de las conexiones y no en las propias neuronas. Sin embargo, el aumento de parámetros ajustables y la inclusión de éstos de manera recurrente complican el aprendizaje de Las RNA recurrentes. La Figura 37 presenta una clasificación de los modelos neuronales recurrentes.

4.5.15 Mecanismo de aprendizaje. El aprendizaje es el proceso por el cual una RNA modifica sus pesos en respuesta a una información de entrada. Los cambios que se producen durante el proceso de aprendizaje se reducen a la destrucción, modificación y creación de conexiones entre las neuronas. En los sistemas biológicos existe una continua creación y destrucción de conexiones. En los modelos de RNA, la creación de una nueva conexión implica que el peso de la misma pasa a tener un valor distinto de cero. De la misma forma, una conexión se destruye cuando su peso pasa a ser cero.

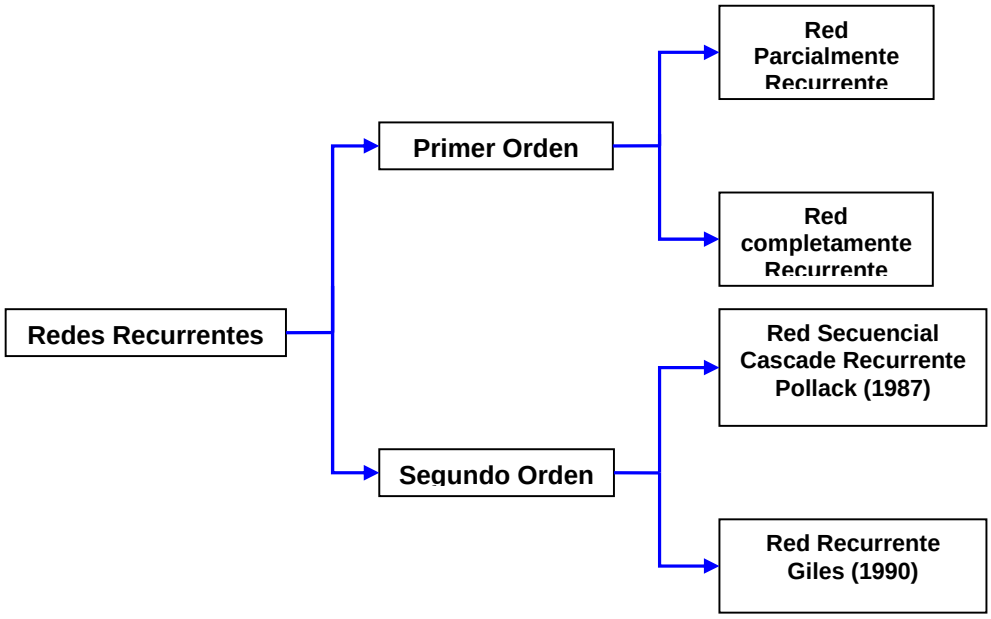


Fig. 37 Modelos de Redes Recurrentes

Durante el proceso de aprendizaje, los pesos de las conexiones de la RNA sufren modificaciones, por tanto se puede afirmar que este proceso ha terminado (*la RNA ha aprendido)* cuando los valores de los pesos permanecen estables $(\partial w_{ij} / \partial t = 0)$, una clasificación sobre los tipos de aprendizaje se detalla mejor en La Figura 38.

.

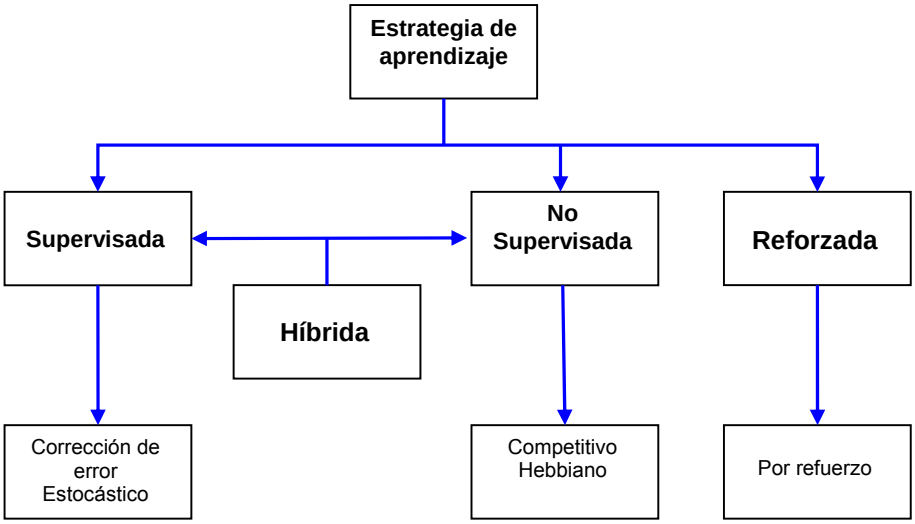


Fig. 38 Clasificación de los Mecanismos de Aprendizaje

Estos criterios determinan lo que se conoce como *la regla de aprendizaje* de la RNA. De forma general, se suelen considerar dos tipos de reglas: las que responden a lo que habitualmente se conoce como aprendizaje supervisado y las correspondientes a un aprendizaje no supervisado.

Es por ello que una de las clasificaciones que se realizan de las RNA obedece al tipo de aprendizaje utilizado por dichas redes. Así, se pueden distinguir:

- RNA con aprendizaje supervisado
- RNA con aprendizaje no supervisado

La diferencia fundamental entre ambos tipos estriba en la existencia o no de un agente externo (*supervisor*) que controle el proceso de aprendizaje de la RNA.

Otro criterio que se puede utilizar para diferenciar las reglas de aprendizaje se basa en considerar si La RNA puede *aprender* durante su funcionamiento habitual o si el aprendizaje supone la *desconexión* de La RNA; es decir su inhabilitación hasta que el proceso termine. En el primer caso, se trataría de un aprendizaje On-Line, mientras que el segundo es lo que se conoce como aprendizaje Off-Line.

Cuando el aprendizaje es Off Line, se distingue entre una *fase de aprendizaje o entrenamiento* y una *fase de operación o funcionamiento,* existiendo un conjunto de datos de entrenamiento y un conjunto de datos de test o prueba que serán utilizados en la correspondiente fase.

En las RNA con aprendizaje On-Line no se distingue entre la fase de entrenamiento y de operación, de tal forma que los pesos varían dinámicamente siempre que se presente una nueva información al sistema. En este tipo de RNA, debido al carácter dinámico de las mismas, el estudio de la estabilidad suele ser un aspecto fundamental.

4.5.15.1 RNA con aprendizaje supervisado. El aprendizaje supervisado se caracteriza porque el proceso de aprendizaje se realiza mediante un entrenamiento controlado por un agente externo (supervisor, maestro) que determina la respuesta que debería generar La RNA a partir de una entrada determinada. El supervisor comprueba la salida de La RNA y en el caso de que esta no coincida con la deseada, se procederá a modificar los pesos de las conexiones, con el fin de conseguir que la salida obtenida se aproxime a la deseada (ver Figura 39).

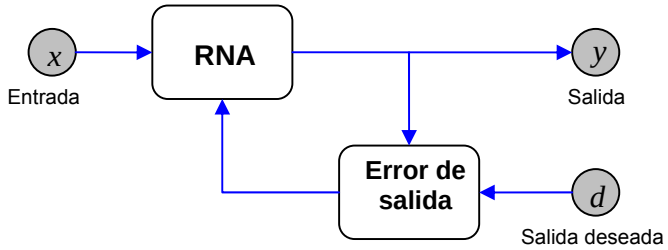


Fig. 39 Ciclo del Aprendizaje Supervisado

En este tipo de aprendizaje se suelen considerar, a su vez, tres formas para llevarlo a cabo, que dan lugar a los siguientes *aprendizajes* supervisados:

- Aprendizaje por corrección de error
- Aprendizaje por refuerzo
- Aprendizaje estocástico

4.5.15.2 RNA con aprendizaje no supervisado. Las RNA con aprendizaje no supervisado (también conocido como auto-supervisado) no requieren influencia externa para ajustar los pesos de las conexiones entre sus neuronas. La RNA no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta; por ello, suele decirse que estas RNA son capaces de *auto-organizarse* (ver Figura 40).

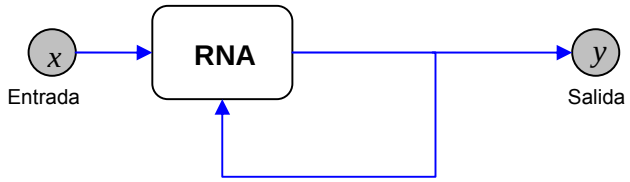


Fig. 40 Ciclo del Aprendizaje no Supervisado

Estas RNA deben encontrar las características, regularidades, correlaciones o categorías que se pueden establecer entre los datos que se presenten en su entrada. Puesto que no hay un supervisor que indique a la RNA que respuesta debe generar ante una entrada concreta, cabría preguntarse, precisamente, por lo que la RNA genera en estos casos. Existen varias posibilidades en cuanto a la interpretación de la salida de estas RNA, que dependen de su estructura y del algoritmo de aprendizaje empleado.

En cuanto a los algoritmos de aprendizaje no supervisado, en general se suelen considerar dos tipos que dan lugar a los siguientes aprendizajes:

- Aprendizaje Hebbiano
- Aprendizaje competitivo y cooperativo

En el primer caso, se pretende medir la familiaridad o extraer características de los datos de entrada, mientras que el segundo suele orientarse hacia la *clusterización* o clasificación de dichos datos.

4.5.16 Tipo de asociación entre las informaciones de entrada y salida. Las RNA son sistemas que almacenan cierta información aprendida; esta información se registra de forma distribuida en los pesos asociados a las conexiones entre neuronas. Por tanto, puede imaginarse una RNA como cierto tipo de memoria que almacena datos de forma estable, datos que se grabarán en dicha memoria como consecuencia del aprendizaje de la RNA y que podrán ser leídos a la salida como respuesta a cierta información de entrada, comportándose entonces la RNA como lo que habitualmente se conoce por memoria asociativa; es decir, cuando se aplica un estímulo (dato de entrada) la RNA responde con una salida asociada a dicha información de entrada.

Existen dos formas primarias de realizar esta asociación entre entrada-salida que se corresponden con la naturaleza de la información almacenada en la RNA. Una primera sería la denominada *heteroasociación,* que se refiere al caso en el que la RNA *aprende* parejas de datos $[(A_1, B_1), (A_2, B_2), ..., (A_N, B_N)]$, de tal forma que cuando que cuando se presente cierta información de entrada A_i, deberá responder generando la correspondiente salida asociada B_i. La segunda se conoce como *autoasociación,* donde La RNA *aprende* ciertas informaciones $A_1, A_2, ..., A_N$, de tal forma que cuando se le presenta una información de entrada realizará una autocorrelación, respondiendo con uno de los datos almacenados, el más parecido al de entrada.

4.5.17 Representación de la información de entrada y salida. Las RNA pueden también clasificarse en función de la forma en que se representan las informaciones de entrada y las respuestas o datos de salida. Así, en un gran número de RNA, tanto los datos de entrada como los de salida son de naturaleza analógica; es decir, son valores reales continuos, normalmente estarán normalizados y su valor absoluto será menor que la unidad. Cuando esto ocurre, las funciones de activación de las neuronas serán también continuas, del tipo lineal o sigmoidal.

4.6 Primeros Modelos Computacionales

4.6.1 Células de McCulloch-Pitts. El primer ejemplo que puede considerarse como una RNA, al menos estructuralmente, son las células de McCulloch-Pitts. Este primer modelo de neurona fue propuesto por Warren McCulloch y Walter Pitts en 1943[8]. En él modelaban una estructura y un funcionamiento simplificado de las neuronas del cerebro,

[8] W. S. McCulloch y W. A. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Boulletin of Mathematics and Biophysics*, 5, págs, 115-133, 1943.

considerándolas como dispositivos con sólo dos estados posibles: apagado (0) y encendido (1) (ver Figura 41).

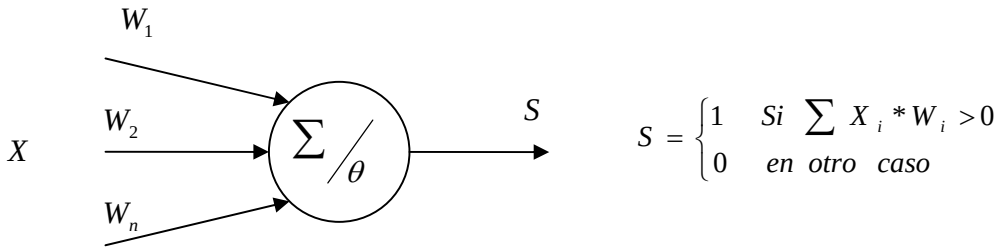


Fig. 41 Esquema de una célula McCulloch-Pitts

La célula de McCulloch-Pitts recibe como entrada un conjunto de n valores binarios, $X = \{x_1, x_2, ..., x_n\}$ procedente de las salidas de otras células, o de la entrada a la red; y produce una única salida también binaria s. Cada célula se caracteriza por $n+1$ valores reales, de los cuales n son los pesos de las conexiones (w_i) correspondientes a las entradas x_i, y el otro es un valor de umbral θ, que puede ser distinto para cada célula. La célula opera en lapsos discretos. La forma de procesar la entrada es la siguiente: la célula se activará y, por lo tanto, producirá un valor 1, si la suma de las entradas multiplicadas por los pesos supera al umbral θ.

$$S(t+1) = \begin{cases} 1 & \sum_i w_i x_i(t) > \theta \\ 0 & en\ otro\ caso \end{cases} \tag{68}$$

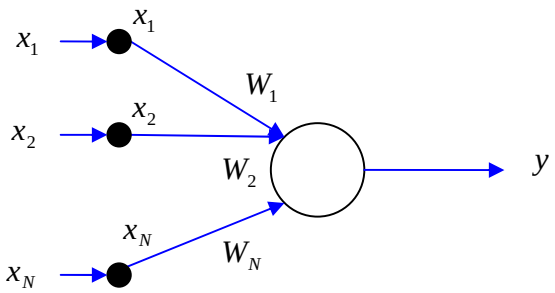
A partir del modelo de neurona de McCulloch-Pitts se define el primer modelo de RNA:

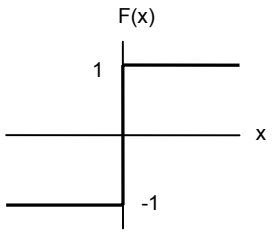
> Una RNA es una colección de neuronas de McCulloch-Pitts, todas con las mismas escalas de tiempos, donde sus salidas están conectadas a las entradas de otras neuronas.

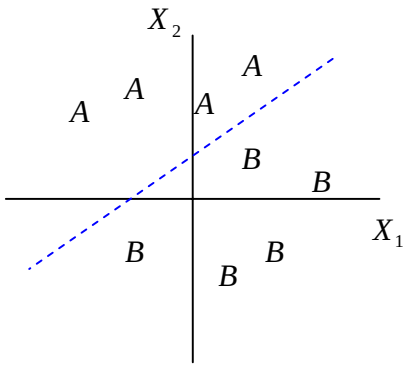
De este modo, una salida puede actuar sobre varias entradas, pero una entrada viene a lo sumo de una salida. La RNA tiene contacto con el exterior a través de las líneas de entrada y de salida. Las líneas de entrada de la RNA formarán parte de la entrada de alguna o de todas las neuronas de la RNA, Asimismo, las líneas de salida procederán de alguna o de todas las neuronas de la RNA.

4.6.2 El Perceptrón. El dispositivo conocido con el nombre de Perceptrón fue inventado por el Psicólogo Frank Rosenblatt a finales de los años cincuenta. En un intento de ilustrar algunas de las propiedades fundamentales de los sistemas inteligentes en general, sin entrar en demasía en ciertas condiciones especiales, y muchas veces desconocidas, que son válidas para organismos biológicos concretos. Rosenblatt creía que la conectividad que se desarrolla en las redes biológicas tiene un elevado porcentaje de aleatoriedad.

La única neurona de salida del Perceptrón realiza la suma ponderada de las entradas, resta el umbral y pasa el resultado a una función de transferencia de tipo escalón. La regla de decisión es responder $+1$ si el patrón presentado pertenece a la clase A, o -1 si el patrón pertenece a la clase B (ver Figura 42). La salida dependerá de la entrada neta (suma de las entradas x_i ponderadas) y del valor del umbral θ.







El Perceptrón puede clasificar un vector analógico de entrada en dos clases A y B.

$$y = f\left[\sum_{i=1}^{N} W_i X_i - \theta\right]$$

$$X_2 = \frac{-W_1}{W_2} X_1 + \frac{\theta}{W_2}$$

Fig. 42 El Perceptrón

Sin embargo, el perceptrón, al constar sólo de una capa de entrada y otra de salida con una única neurona, tiene una capacidad de representación bastante limitada. Este modelo sólo es capaz de discriminar patrones muy sencillos, linealmente separables. El caso más conocido es la imposibilidad del Perceptrón de representar la función Or-Exclusiva.

4.6.2.1 Regla de aprendizaje del Perceptrón. El algoritmo de aprendizaje del Perceptrón es de tipo supervisado, lo cual requiere que sus resultados sean evaluados y se realicen las oportunas modificaciones del sistema si fuera necesario. Desgraciadamente, sólo se pueden aprender clasificaciones fáciles (problemas de orden 1 en la terminología de Minsky y Papert[9]).

Para dar más claridad sobre el funcionamiento de la RNA Perceptrón es importante comprender el algoritmo de convergencia para ajustarlos pesos, el cual realiza el aprendizaje para esta RNA, específicamente, el aprendizaje por corrección de error) con N elementos procesales de entrada y un único elemento procesal de salida:

[9] M Minsky y S. Papert, “Perceptrons”; Ed. MIT Press, 1969.

1. Inicialización de los pesos y del umbral

Inicialmente se asignan valores aleatorios a cada uno de los pesos (w_i) de las conexiones y al umbral $-w_0 = \theta$.

2. Presentación de un nuevo par (Entrada, Salida esperada)

Presentar un nuevo patrón de entrada $X_P = (x_1, x_2, ..., x_N)$ junto con la salida esperada $d(t)$.

3. Cálculo de la salida actual

$$y(t) = f\left[\sum_i w_i(t)x_i(t) - \theta\right] \tag{69}$$

Siendo $f(x)$ la función de transferencia escalón.

4. Adaptación de los pesos

$$w_i(t+1) = w_i(t) + \alpha[d(t) - y(t)]x_i(t) \qquad (0 \le i \le N) \tag{70}$$

Donde $d(t)$ representa la salida deseada, y será 1 si el patrón pertenece a la clase A, y -1 si es de la clase B. En estas ecuaciones, α es un factor de ganancia en el rango de 0.0 a 1.0. Este factor debe ser ajustado de forma que satisfaga tanto los requerimientos de aprendizaje rápido como la estabilidad de las estimaciones de los pesos. Este proceso se repite hasta que el error que se produce para cada uno de los patrones (diferencia entre el valor de salida deseado y obtenido) es cero o bien menor que un valor preestablecido. Hay que observar que los pesos de no se cambian si la red ha tomado la decisión correcta.

5. Volver al paso 2

Este algoritmo es extensible al caso de múltiples neuronas en la capa de salida. El perceptrón será capaz de aprender a clasificar todas sus entradas, en un número finito de pasos, siempre y cuando el conjunto de los patrones de entrada sea linealmente separable.

4.6.3 RNA Perceptrón multicapa. El perceptrón multinivel o multicapa es una RNA de tipo feedforward compuesta de varias capas de neuronas entre la entrada y la salida de la misma (ver Figura 43). Esta RNA permite establecer regiones de decisión mucho más complejas que las de dos semiplanos, como hacía el Perceptrón de un solo nivel.

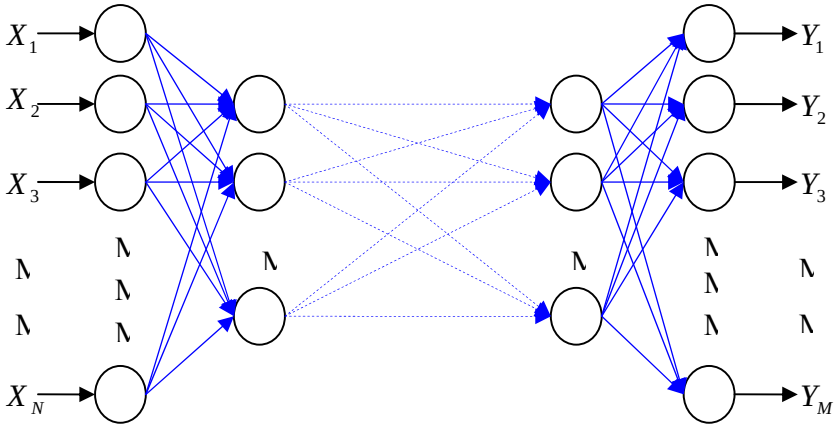


Fig. 43 Perceptrón Multicapa

Las capacidades del Perceptrón con dos, tres, y cuatro niveles o capas y con una única neurona en el nivel de salida, se muestran en la figura 44, en la segunda columna se exhibe el tipo de región de decisión que se puede formar con cada una de las configuraciones. En la siguiente columna se indica el tipo de región de decisión que se formaría para el problema de XOR. En las dos últimas columnas se muestran las regiones formadas para resolver el problema de clases con regiones mezcladas y las formas de regiones más generales para cada uno de los casos.

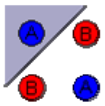
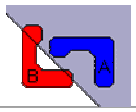
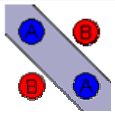
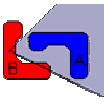
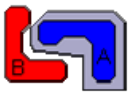
Estructura	Regiones de Decisión	Problema de la XOR	Clases con Regiones Mezcladas	Formas de Regiones más Generales
1 Capa 	Medio Plano Limitado por un Hiperplano			
2 Capas 	Regiones Cerradas o Convexas			
3 Capas 	Complejidad Arbitraria Limitada por el Número de Neuronas			

Fig. 44 Formas de Regiones Generadas por Un Perceptrón Multinivel

El Perceptrón básico de dos capas (la de entrada con neuronas lineales y la de salida con función de activación de tipo escalón) sólo puede establecer dos regiones separadas por una frontera lineal en el espacio de patrones de entrada. Un perceptrón con tres o más niveles de neuronas puede formar cualquier región convexa en este espacio.

4.6.3.1 Arquitectura del Perceptrón multicapa. La arquitectura del Perceptrón multicapa se caracteriza por que tiene sus neuronas agrupadas en capas de diferentes niveles. Cada una de las capas está formada por un conjunto de neuronas y se distinguen tres tipos de capas diferentes: la capa de entrada, las capas ocultas y la capa de salida.

Las conexiones del Perceptrón multicapa siempre están dirigidas hacia delante, es decir, las neuronas de una capa se conectan con las neuronas de la siguiente capa, de ahí que reciban el nombre de redes alimentadas hacia delante o redes "feedforward". Las conexiones entre las neuronas llevan asociado un número real, llamado peso de la conexión. Todas las neuronas de la red llevan también asociado un umbral, que en el caso del Perceptrón multicapa suele tratarse como una conexión más a la neurona, cuya entrada es constante e igual a 1.

4.6.3.2 Diseño de la arquitectura del Perceptrón multicapa. Cuando se aborda un problema utilizando el Perceptrón multicapa, uno de los primeros pasos a realizar es el diseño de la arquitectura de la RNA. Este diseño implica la determinación de la función de activación a emplear, el número de neuronas y el número de capas de la RNA.

La elección de la función de activación se suele hacer basándose en el recorrido deseado, y el hecho de elegir una u otra, no influye en la capacidad de la RNA para resolver el problema. En lo que respecta al número de neuronas y capas, algunos de estos parámetros vienen dados por el problema y otros deben ser elegidos por el diseñador. Así, por ejemplo, tanto el número de neuronas en la capa de entrada, como el número de neuronas de la capa de salida, vienen dados por las variables que definen el problema.

4.6.3.3 Algoritmo Backpropagation. La regla de aprendizaje del Perceptrón de Rosenblatt y el algoritmo LMS de Widrow y Hoff fueron diseñados para entrenar redes de una sola capa. Estas redes tienen la desventaja que solo pueden resolver problemas linealmente separables, fue esto lo que llevó al surgimiento de Las RNA multicapa para sobrepasar esta dificultad de Las RNA hasta entonces conocidas.

En 1986, Rumelhart, Hinton y Willians [Rumelhart 86], basándose en los trabajos de investigadores como: [Werbos 74] y [Parker 82], formalizaron un método para que una RNA aprendiera la asociación que existe entre los patrones de entrada a la misma y las clases correspondientes, utilizando más niveles de neuronas que los que utilizó Rosenblatt para desarrollar el Perceptrón. Este método, conocido como Backpropagation (propagación del error hacia atrás), está basado en la generalización de la regla delta y, a pesar de sus propias limitaciones, ha ampliado de forma considerable el rango de aplicaciones de las RNA.

4.6.3.4 La Regla delta generalizada. La regla propuesta por Widrow en 1960 (regla delta) ha sido extendida a redes con capas intermedias (regla delta generalizada) con conexiones hacia delante (feedforward) y cuyas células tienen funciones de activación

continuas (lineales o sigmoidales), dando lugar al algoritmo de retropropagación (Backpropagation). Estas funciones continuas son no decrecientes y derivables. La función sigmoidal pertenece a este tipo de funciones, a diferencia de la función escalón que se utiliza en el perceptrón, porque esta última no es derivable en el punto en el que se encuentra la discontinuidad.

Este algoritmo utiliza también una función o superficie de error asociada a la RNA, buscando el estado estable de mínima energía o de mínimo error a través del camino descendente de la superficie de error. Por ello, realimenta el error del sistema para realizar la modificación de los pesos en un valor proporcional al gradiente decreciente de dicha función de error.

4.6.3.5 Funcionamiento del Algoritmo. El método que sigue la regla delta generalizada para ajustar los pesos es el mismo que el de la regla delta utilizada en el Perceptrón y el Adaline, es decir, los pesos se actualizan de forma proporcional a la delta, o diferencia entre la salida deseada y la obtenida (δ = sal. Deseada - sal. Obtenida)

Dada una neurona (unidad U_i) y la salida que produce y_i (ver Figura 45), el cambio que se produce en el peso de la conexión que une la salida de dicha neurona con la unidad U_j (w_{ji}) para un patrón de aprendizaje p determinado es:

$$\Delta w_{ji}(t+1) = \alpha \delta_{pj} y_{pi} \tag{71}$$

En donde el subíndice p se refiere al patrón de aprendizaje concreto y α es la constante o tasa de aprendizaje.

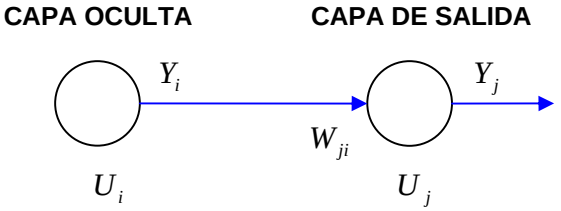


Fig. 45 Conexión Entre una Neurona de una Capa Oculta con una Neurona de Salida

El punto en el que difieren la regla delta generalizada de la regla delta es en el valor concreto de δ_{pj}. Por otro lado, en las redes multinivel, a diferencia de las redes sin neuronas ocultas, en principio no se puede conocer la salida deseada de las neuronas de las capas ocultas para determinar los pesos en función del error cometido. Sin embargo, inicialmente se puede conocer la salida deseada de las neuronas de salida. Según esto, si se considera la unidad U_j (ver Figura 105), entonces se define:

$$\delta_{pj} = (d_{pj} - y_{pj}) * f'(net_j) \tag{72}$$

Donde d_{pj} es la salida deseada de la neurona j para el patrón p y net_j es la entrada neta que recibe la neurona j.

Esta formula es como la de la regla delta, excepto en lo que se refiere a la derivada de la función de transferencia. Este término representa la modificación que hay que realizar en la entrada que recibe la neurona j. En el caso que dicha neurona no sea de salida, el error que se produce estará en función del error que se cometa en las neuronas que reciban como entrada la salida de dicha neurona. Esto es lo que se denomina procedimiento de propagación del error hacia atrás.

Según lo anterior, en el caso que U_j no sea una neurona de salida (ver Figura 46), el error que se produce está en función del error que se comete en las neuronas que reciben como entrada la salida de U_j:

$$\delta_{pj} = \left(\sum_k \delta_{pk} w_{kj} \right) * f'(net_j) \tag{73}$$

Donde el rango de k cubre todas aquellas neuronas a las que está conectada la salida de U_j. De esta forma, el error que se produce en una neurona oculta es la suma de los errores que se producen en las neuronas a las que está conectada la salida de ésta, multiplicando cada uno de ellos por el peso de la conexión.

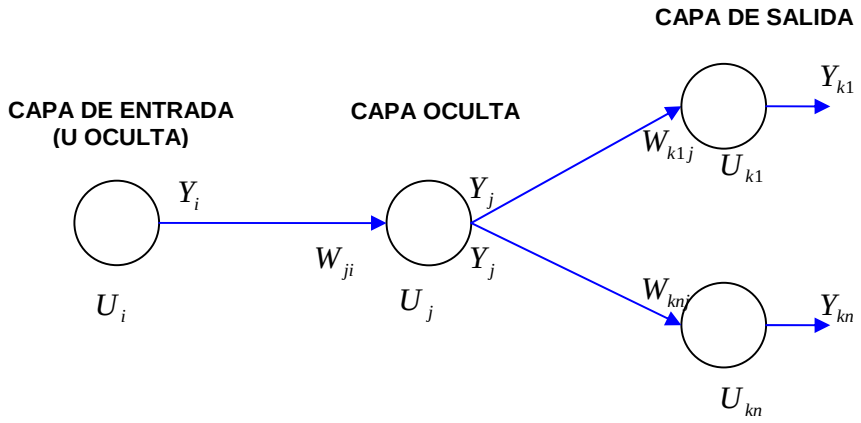


Fig. 46 Conexiones Entre Neuronas de la Capa Oculta con la Capa de Salida

4.6.3.6 Adición de un momento en la regla delta generalizada. El método de retropropagación del error, también conocido como del gradiente descendente, requiere

un gran número de cálculos para lograr el ajuste de los pesos de La RNA. En la implementación del algoritmo, se toma una amplitud de paso que viene dada por la tasa de aprendizaje α. A mayor tasa de aprendizaje, mayor es la modificación de los pesos en cada iteración, con lo que el aprendizaje será más rápido, pero por otro lado, puede dar lugar a oscilaciones. Rumelhart, Hinton Y Willians (1986), sugirieron que para filtrar estas oscilaciones se añada en la expresión del incremento de los pesos un término (momento) β , de manera que dicha expresión quede:

$$\begin{aligned} w_{ji}(t+1) = w_{ji}(t) + \alpha\delta_{pj} y_{pi} + \beta(w_{ji}(t) - w_{ji}(t+1)) = \\ \Delta w_{ji}(t+1) = \alpha\delta_{pj} y_{pi} + \beta\Delta w_{ji}(t) \end{aligned} \tag{74}$$

Donde β es una constante (momento) que determina el efecto en $t+1$ del cambio de los pesos en el instante t.

Con este momento se consigue la convergencia de la red en menor número de iteraciones, porque si en t el incremento de un peso era positivo y en $t+1$también, entonces el descenso por la superficie de error en $t+1$es mayor. Si embargo, si en t el incremento era positivo y en $t+1$ es negativo, el paso que se da en $t+1$ es más pequeño, lo cual es adecuado, debido a que eso significa que se ha pasado por un mínimo y que los paso deben ser menores para alcanzarlo.

Resumiendo el Algoritmo de Backpropagation queda finalmente:

$$w_{ji}(t+1) = w_{ji}(t) + [\Delta w_{ji}(t+1)] \tag{75}$$

$$w_{ji}(t+1) = w_{ji}(t) + [\alpha\delta_{pj} y_{pi} + \beta\Delta w_{ji}(t)] \tag{76}$$

Donde:

$$\delta_{pj} = (d_{pj} - y_{pj}) f'(net_j) \tag{77}$$

Si U_j es una neurona de salida y

$$\delta_{pj} = (\sum_k \delta_{pk} w_{kj}) f'(net_j) \tag{78}$$

Si U_j no es una neurona de salida.

4.6.3.7 Estructura y aprendizaje de una RNA con el algoritmo Backpropagation. En una RNA multinivel existe una capa de entrada con n neuronas y una capa de salida con m neuronas y al menos una capa oculta de neuronas internas. Cada neurona de una capa (excepto la de entrada) recibe entradas, de todas las neuronas de la capa anterior y envía su salida a todas las neuronas de la capa posterior (excepto las salidas). No hay conexiones hacia atrás feedback ni laterales entre neuronas de la misma capa.

A continuación se presentan, a modo de síntesis, los pasos y fórmulas a utilizar para aplicar el algoritmo de entrenamiento.

Paso 1

Inicializar los pesos de La RNA con valores pequeños aleatorios.

Paso 2

Presentar un patrón de entrada, $X_P : x_{p1}, x_{p2},..., x_{pN}$ y especificar la salida deseada que debe generar La RNA: $d_1, d_2,..., d_M$ (si la red se utiliza como un clasificador, todas las salidas deseadas serán cero, salvo una, que será la de la clase a la que pertenece el patrón de entrada).

Paso 3

Calcular la salida actual de la red, para ello se presentan las entradas a la red y se calcula la salida que presenta cada capa hasta llegar a la capa de salida ésta será la salida de la red $y_1, y_2,..., y_M$. Los pasos son los siguientes:

- Se calculan las entradas netas para las neuronas ocultas procedentes de las neuronas de entrada.

  Para una neurona j oculta:

$$net_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + \theta_j^h$$

  En donde el índice h se refiere a magnitudes de la capa oculta (hidden); el subíndice p, al p-ésimo vector de entrenamiento, y j a la j-ésima neurona oculta. El término θ puede ser opcional, pues actúa como una entrada más.

- Se calculan las salidas de las neuronas ocultas:

$$y_{pj} = f_j^h(net_{pj}^h)$$

- Se realizan los mismos cálculos para obtener las salidas de las neuronas de salida (capa o : output).

$$net_{pk}^{o} = \sum_{j=i}^{L} w_{kj}^{o} y_{pj} + \theta_{k}^{o}$$

$$y_{pk} = f_{k}^{o}(net_{pk}^{o})$$

Paso 4

Si la neurona k es una neurona de la capa de salida, el valor del delta es:

$$\delta_{pk}^{o} = (d_{pk} - y_{pk}) f_{k}^{o'}(net_{pk}^{o})$$

La función f debe cumplir el requisito de ser derivable, lo que implica la imposibilidad de utilizar una función escalón. En general se disponen de dos formas de función de salida para cumplir el requisito: la función lineal de salida ($f_k(net_{jk}) = net_{jk}$) y la función sigmoidal definida por la expresión:

$$f_k(net_{jk}) = \frac{1}{1+e^{-net_{jk}}}$$

La selección de la función de salida depende de la forma en que se decida representar los datos de salida: si se desea que las neuronas de salida sean binarias, se utiliza la función sigmoidal, puesto que esta función es casi biestable y, además, derivable. En otros casos es tan aplicable un función como otra.

Para la función lineal, se tiene $f_k^{o'} = 1$, mientras que la derivada de una función f sigmoidal es:

$$f_k^{o'} = f_k^{o}(1 - f_k^{o}) = y_{pk}(1 - y_{pk})$$

Por lo que los términos de error para las neuronas de salida quedarían:

$$\delta_{pk}^{o} = (d_{pk} - y_{pk})$$

Para la salida lineal, y

$$\delta_{pk}^{o} = (d_{pk} - y_{pk}) y_{pk} (1 - y_{pk})$$

Para la salida sigmoidal.

Si la neurona j no es de salida, entonces la derivada parcial del error no puede ser evaluada directamente. Por tanto, se obtiene el desarrollo a partir de valores que son conocidos y otros que pueden ser evaluados.

La expresión obtenida en este caso es:

$$\delta_{pj}^{h} = f_{j}^{h'}(net_{pj}^{h}) \sum_{k} \delta_{pk}^{o} w_{kj}^{o}$$

Donde se observa que el error en las capas ocultas depende de todos los términos de error de la capa de salida. De aquí surge el término de propagación hacia atrás. En particular, para la función sigmoidal:

$$\delta_{pj}^{h} = x_{pj} (1 - x_{pj}) \sum_{k} \delta_{pk}^{o} w_{kj}^{o}$$

Donde k se refiere a todas las neuronas de la capa superior a la de la neurona j. Así, el error que se produce en una neurona oculta es proporcional a la suma de los errores conocidos que se producen en las neuronas a las que está conectada la salida de está, multiplicando cada uno de ellos por el peso de la conexión. Los umbrales internos de las neuronas se adaptan de forma similar, considerando que están conectados con pesos desde entradas auxiliares de valor constante.

Paso 5

Actualización de los pesos

Para ello, se utiliza el algoritmo recursivo, comenzando por las neuronas de salida y trabajando hacia atrás hasta llegar a la capa de entrada, ajustado los pesos de la siguiente forma:

Para los pesos de las neuronas de la capa de salida:

$$w_{kj}^{o}(t+1) = w_{kj}^{o}(t) + \Delta w_{kj}^{o}(t+1);$$
$$\Delta w_{kj}^{o}(t+1) = \alpha \delta_{pk}^{o} y_{pj}$$

Y para los pesos de las neuronas de la capa oculta:

$$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) + \Delta w_{ji}^{h}(t+1);$$
$$\Delta w_{ji}^{h}(t+1) = \alpha \delta_{pj}^{h} x_{pi}$$

En ambos casos, para acelerar el proceso de aprendizaje, se puede añadir un término momento de valor: $\beta(w_{kj}^{o}(t) - w_{kj}^{o}(t-1))$ en el caso de la neurona de salida y $\beta(w_{ji}^{h}(t) - w_{ji}^{h}(t+1))$ cuando se trata de una neurona oculta.

Paso 6

El proceso se repite hasta que el término de error

$$E_p = \frac{1}{2}\sum_{k=1}^{M} \delta_{pk}^{2}$$

Resulta pequeño para cada uno de los patrones aprendidos.

4.6.3.8 Control de convergencia. En las técnicas de gradiente decreciente es conveniente avanzar por la superficie de error con incrementos pequeños de los pesos. Esto se debe a que tenemos una información local de la superficie y no se sabe lo lejos o lo cerca que se está del punto mínimo. Con incrementos grandes, se corre el riesgo de pasar por encima del punto mínimo sin conseguir estacionarse en él. Con incrementos pequeños, aunque se tarde más en llegar, se evita que ocurra esto.

El elegir un incremento paso adecuado influye en la velocidad con la que converge el algoritmo. Esta velocidad se controla a través de la constante de proporcionalidad o tasa de aprendizaje α, Normalmente, α debe ser un número pequeño (del orden de 0.05 a 0.25), para asegurar que la red llegue a asentarse en una solución. Un valor pequeño de α significa que la red tendrá que hacer un gran número de iteraciones. Si esa constante es muy grande, los cambios de pesos son muy grandes, avanzando rápido por la superficie de error, con el riesgo de saltar el mínimo y estar oscilando alrededor de él, pero sin poder alcanzarlo.

4.6.3.9 Dimensionamiento de la RNA, número de neuronas ocultas. No se pueden dar reglas concretas para determinar el número de neuronas o el número de capas de una red para resolver un problema concreto. Lo mismo ocurre a la hora de seleccionar el conjunto de vectores de entrenamiento. En todos estos casos, lo único que se puede dar son ideas generales deducidas de la experiencia de numerosos autores [Freeman 91].

Respecto al número de capas de la RNA, en general tres capas son suficientes (entrad-oculta-salida). Sin embargo, hay veces en que un problema es más fácil de resolver (la red aprende más de prisa) con más de una capa oculta. El tamaño de las capas, tanto de entrada como de salida, suele venir determinado por la naturaleza de la aplicación. En cambio, decidir cuántas neuronas debe tener la capa oculta no suele ser tan evidente. Un

comienzo de construcción puede ser el de asumir que el número de neuronas en la capa oculta sean menos de la mitad de la suma del número de entradas más el de las salidas.

4.6.4 RNA de base radial (RBFN). Son RNA multicapa con conexiones hacia delante, al igual que el Perceptrón multicapa, las RBFN se caracterizan porque están formadas por una única capa oculta y cada neurona de esta capa posee un carácter local, en el sentido que cada neurona oculta de la RNA se activa en una región diferente del espacio de patrones de entrada. Este carácter local viene dado por el uso de las llamadas funciones de base radial, generalmente, la función gaussiana, como funciones de activación. Las neuronas de la capa de salida de las redes de base radial realizan una combinación lineal de las activaciones de las neuronas ocultas.

La mayor contribución a la teoría, diseño y aplicaciones de las RNA de base radial se debe a Moody y darken [Moody and Darken 1989], Renals [Renals 1989] y a Poggio y Girossi [Poggio and Girossi 1990]. Uno de los objetivos iniciales de los autores era construir una red de neuronas que requiriera un menor tiempo de aprendizaje que el necesario por el Perceptrón multicapa y, de este modo, disponer de una red de neuronas que pudiera ser apropiada para aplicaciones en tiempo real. Esto se consiguió incorporando funciones de activación locales en las neuronas ocultas de la RNA, lo cual permitía que sólo unas pocas neuronas ocultas tuvieran que ser procesadas para nuevos patrones de entrada.

Al igual que el Perceptrón multicapa, Las RBFN son aproximadores universales, en el sentido que pueden aproximar cualquier función continua sobre un compacto de $\Re^n$.

Las funciones de base radial definen hiperesferas o hiperelipses que dividen el espacio de entrada. Por tanto, cada neurona oculta de La RBFN construye una aproximación local y no lineal en una determinada región de dicho espacio. Puesto que la salida de la red es combinación lineal de las funciones de base radial. Las aproximaciones que construyen las RBFN son combinaciones lineales de múltiples funciones locales y no lineales. De este modo, se suele decir que las RBFN aproximan relaciones complejas mediante una colección de aproximaciones locales menos complejas, dividiendo el problema en varios sub-problemas menos complejos.

4.6.4.1 Arquitectura de las RNA de base radial. Las RBFN están formadas por tres capas de neuronas: la capa de entrada, una única capa oculta y la capa de salida, como se ilustra en la Figura 47. La capa de entrada la componen un conjunto de neuronas que reciben las señales del exterior, transmitiéndolas a la siguiente capa sin realizar ningún procesado sobre dichas señales. Las neuronas de la capa oculta reciben las señales de la capa de entrada y realizan una transformación local y no lineal sobre dichas señales. Este carácter local es lo que las diferencia del Perceptrón multicapa, no sólo en cuanto a arquitectura, sino también en cuanto a comportamiento. Esta capa es la única que incluye componentes no lineales en las redes de base radial. Y, finalmente, la capa de salida que realiza una combinación lineal de las actividades de las neuronas ocultas, que actúa además como salida de la RNA.

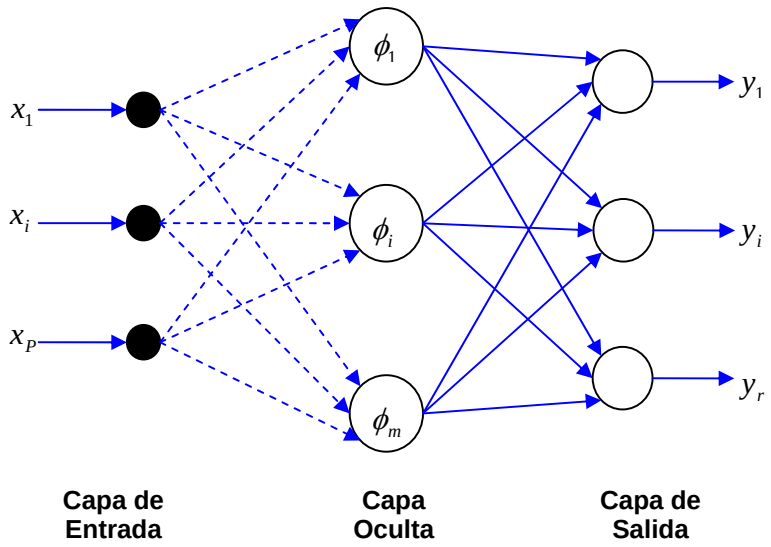


Fig. 47 Arquitectura de las RBFN

4.6.4.2 Activaciones de las Neuronas de la RBFN. Dada una RBFN con p neuronas en la capa de entrada, m neuronas en la capa oculta y r neuronas en la capa de salida, las activaciones de las neuronas de salida para el patrón de entrada $n, X(n) = (x_1(n), x_2(n), ..., x_p(n))$, denotadas como $y_k(n)$, vienen dadas por la siguiente expresión:

$$y_k(n) = \sum_{i=1}^{m} w_{ik}\phi_i(n) + u_k \qquad para\ k = 1,2,...,r \tag{79}$$

Donde w_{ik} representa el peso de la conexión de la neurona oculta i a la neurona de salida k, w_k es el umbral de la neurona de salida k y $\phi_i(n)$ son las activaciones de las neuronas ocultas para el patrón de entrada $X(n)$ en la anterior ecuación se observa que las neuronas de salida de la red utilizan la función de activación identidad, realizando una transformación lineal de las activaciones todas las neuronas ocultas.

Las funciones ϕ_i, también conocidas como funciones de base radial, determinan las activaciones de las neuronas ocultas de la red en función del vector de entrada a la red $X(n)$ y vienen dadas por la siguiente expresión:

$$\phi_i(n) = \phi\left(\frac{\|X(n) - C_i\|}{d_i}\right) \qquad para\ i = 1,2,...,m \tag{80}$$

Donde ϕ en una función de base radial; $C_i = (c_{i1},...,c_{ip})$ son vectores que representan los centros de la función de base radial; d_i son números reales que representa la desviación, anchura o dilatación de la función de base radial; y $\|\ \|$ es la distancia euclidiana del vector de entrada $X(n)$ al centro C_i, definida como:

$$\|X(n) - C_i\| = \left(\sum_{j=1}^{P} (x_j(n) - c_{ij})^2 \right)^{\frac{1}{2}} \tag{81}$$

Por tanto, la activación de una neurona oculta en las RBFN depende de la distancia del patrón de entrada $X(n)$ al centro C_i de la función de base radial. Estas funciones bases ϕ poseen un carácter local, pues son funciones que alcanzan un nivel cercano al máximo de su recorrido cuando el patrón de entrada $X(n)$ está próximo al centro de la neurona; a medida que el patrón se aleja del centro, el valor de función tiende al valor mínimo de su recorrido.

La función de base radial ϕ puede adoptar diferentes formas y expresiones, entre ellas se encuentran:

- Función Gaussiana

$$\phi(r) = e^{\left(\frac{-r^2}{2}\right)} \tag{82}$$

- Función Inversa Cuadrática

$$\phi(r) = \frac{1}{1+r^2} \tag{83}$$

- Función Inversa Multicuadrática

$$\phi(r) = \frac{1}{\sqrt{1+r^2}} \tag{84}$$

En el concepto de RBFN la más utilizada es la función gaussiana [Moody and Darken 89]. Por tanto, la activación de las neuronas ocultas de las RBFN viene dada, generalmente, por la siguiente expresión:

$$\phi_i(n) = \exp^{\frac{\|X(n)-C_i\|^2}{2d_i^2}} = \exp^{\frac{\sum_{j=1}^{p}(x_j(n)-c_{ij})^2}{2d_i^2}} \qquad para\ i = 1,2,..,m \tag{85}$$

Las salidas de las RBFN (79) son una combinación lineal de gaussianas, cada una de las cuales se activa para una determinada porción del espacio definido por los patrones de entrada.

4.6.4.3 Diseño de la Arquitectura de las RBFN. El número de entradas y salidas en una RBFN viene dado por el número de variables que definen el problema. Como ocurría cuando se utilizaba un Perceptrón multicapa, en algunas aplicaciones no hay lugar a duda sobre dichas variables. Sin embargo, existen aplicaciones en las que pudiera ser necesario llevar a cabo un análisis de las variables más relevantes y significativas que definen el problema.

4.6.4.4 Aprendizaje de las RBFN. El proceso de aprendizaje implica la determinación de todos los parámetros que intervine en la RBFN. Estos son: los centros y las desviaciones de las neuronas ocultas y los pesos de la capa oculta a la capa de salida, así como los umbrales de las neuronas de salida.

Debido a que las capas de neuronas en una red de base radial realizan tareas diferentes, es razonable separar el proceso de optimización de los parámetros de la capa oculta y los de la capa de salida mediante la utilización de diferentes técnicas.

Por tanto, uno de los mecanismos más usados para el aprendizaje de las RBFN es el llamado **método hibrido**, que combina dos fases: una fase no supervisada para la determinación de los centros y otra supervisada para la determinación de los pesos y los umbrales. Además del método anteriormente mencionado también se encuentra el método de aprendizaje totalmente supervisado, a continuación se describirá en detalle cada uno de estos métodos de aprendizaje, así como sus propiedades y características.

4.6.4.4.1 Método de aprendizaje hibrido. El método hibrido realiza el aprendizaje de las RBFN en dos fases:

- Fase no supervisada: determinación de los centros y amplitudes de las neuronas de la capa oculta.
- Fase supervisada: determinación de pesos y umbrales de la capa de salida.

El proceso de optimización en cada una de las fases se formula con un objetivo diferente y las técnicas para su resolución serán también, por tanto diferentes.

4.6.4.4.2 Fase no supervisada. Puesto que las neuronas ocultas de las RBFN se caracterizan porque representan zonas diferentes del espacio de patrones de entrada, los centros y las desviaciones de las funciones de base radial deben ser determinados con este objetivo, es decir, con el objetivo de clasificar el espacio de entrada en diferentes clases. El representante de cada clase será el centro de la función de base radial y la desviación vendrá dada por la amplitud de cada clase.

4.6.4.4.3 Determinación de los centros: algoritmo K-medias. Los centros de las funciones de base radial se determinan, por tanto, mediante un algoritmo de clasificación no supervisado que permita dividir el espacio de patrones de entrada en clases. El número de clases es el número de neuronas ocultas en la red de base radial. El método más utilizado es el algoritmo de K-medias, aunque es necesario destacar que cualquier algoritmo de clasificación no supervisado podría ser utilizado, como, por ejemplo, los mapas autoorganizados de Kohonen.

El algoritmo de K-medias [Lloyd 1982], [MacQueen 1967] es un algoritmo de clasificación no supervisado mediante el cual el espacio de patrones de entrada se divide en K clases o regiones. El representante de cada una de estas clases C_i, será el centro de la neurona oculta i. Dichos centros determinan con el objetivo de minimizar las distancias euclidianas entre los patrones de entrada y el centro más cercano, es decir:

$$J = \sum_{i=1}^{K} \sum_{n=1}^{N} M_{in} \left\| X(n) - C_i \right\| \tag{86}$$

Donde N es el número de patrones, $\| \ \|$ es la distancia euclidiana, $X(n)$ es el patrón de entrada n y Min es la función de pertenencia, que vale 1 si el centro C_i es el más cercano al patrón $X(n)$, y 0 en otro caso, es decir:

$$M_{in} = \begin{cases} 1 & si \left\| X(n) - C_i \right\| < \left\| X(n) - C_i \right\| \forall \ s \neq i, \ s = 1,2,...,K \\ 0 & En\, otro\, caso \end{cases} \tag{87}$$

Dado K el número de clases, $\{X(n) = (x_1(n), x_2(n),..., x_p(n))\}_{n=1...N}$ el conjunto de patrones de entrada y $\{C_i = (c_{i1}, c_{i2},..., x_{ip})\}_{i=1...K}$ los centros de las clases, los pasos para la aplicación del algoritmo son los siguientes:

Paso 1 Se inicializan los centros de las K clases. Pueden inicializarse a K patrones aleatorios del conjunto de patrones disponibles, o bien puede realizarse aleatoriamente, en cuyo caso es conveniente que se inicialicen dentro del rango de valores de los patrones de entrada.

Paso 2 Se asignan N_i patrones de entrada a cada clase i del siguiente modo: el patrón $X(n)$ pertenece a la clase i si $\left\| X(n) - C_i \right\| < \left\| X(n) - C_s \right\| \ \forall \ s \neq i \ con \ s = 1,2,...,K$. Por tanto, cada clase tendrá asociado un determinado número de patrones de entrada, aquellos más cercanos al centro de la clase.

Paso 3 Se calcula la nueva posición de los centros de las clases como la media de todos los patrones que pertenecen a su clase, es decir:

$$c_{ij} = \frac{1}{N_i}\sum_{n=1}^{N} M_{in} x_j(n) \;\; para \;\; j = 1,2,...,p \;,\; i = 1,2,...,K \tag{88}$$

Paso 4 Se repiten los pasos 2 y 3 hasta que las nuevas posiciones de los centros no se modifiquen respecto a si posición anterior, es decir, hasta que:

$$\left\| C_i^{nuevo} - C_i^{anterior} \right\| < \varepsilon \;\; \forall \; i = 1,2,...,K \tag{89}$$

Siendo ε un número real positivo próximo a cero que marca la finalización del algoritmo.

El algoritmo de K-medias es un método fácil de implementar y usar: suele ser un algoritmo bastante eficiente en problemas de clasificación, pues converge en pocas iteraciones hacia un mínimo de la función J dada por la ecuación (86), aunque podría tratarse de un mínimo local.

4.6.4.4.4 Determinación de las amplitudes. Una vez determinados los centros de las funciones de base radial, las amplitudes o desviaciones de dichas funciones deben calcularse de manera que cada neurona oculta se active en una región del espacio de entrada y de manera que el solapamiento de las zonas de activación de una neurona a otra sea lo más ligero posible, para suavizar así la interpolación.

Las amplitudes de cada función de base radial se pueden determinar usando heurísticas basadas en los /itp-vecinos más cercanos, tal y como lo propuso Moody [Moody and darken 1989], los cuales permiten que el solapamiento entre las neuronas ocultas sea lo más suave posible. Se pueden utilizar diferentes heurísticas, como, por ejemplo:

- Media uniforme de las distancias euclidianas del centro C_i a los p centros más cercanos:

$$d_i = \frac{1}{p}\sum_{p} \left\| C_i - C_p \right\| \tag{90}$$

- Otra opción bastante efectiva es determinar la amplitud de la función de base radial como la media geométrica de la distancia del centro a sus dos vecinos más cercanos:

$$d_i \sqrt{\left\| C_i - C_t \right\| \left\| C_i - C_s \right\|} \tag{91}$$

Siendo C_t y C_s los dos centros más cercanos al centro C_i

4.6.4.4.5 Fase supervisada. En esta fase se calculan los pesos y los umbrales de las neuronas de salida de la RBFN. En este caso, el objetivo es minimizar las diferencias entre las salidas de la red y las salidas deseadas. Por tanto, el proceso de aprendizaje está guiado por la minimización de una función error computada en la salida de la red:

$$E = \frac{1}{N}\sum_{n=1}^{N} e(n) \tag{92}$$

Donde N es el número de patrones o muestras y $e(n)$ es el error cometido por la red para el patrón $X(n)$, que viene dado generalmente por:

$$e(n) = \frac{1}{2}\sum_{k=1}^{r} (s_k(n) - y_k(n))^2 \tag{93}$$

Siendo $Y(n) = (y_1(n),...,y_r(n))$ y $S(n) = (s_1(n),...,s_r(n))$ los vectores de salida de la red y salida deseada para el patrón de entrada $X(n)$, respectivamente.

4.6.4.4.6 Mínimos cuadrados. Para resolver este problema de optimización se suele utilizar una técnica basada en la corrección del error. En la ecuación (79) se observa que las salidas de la RBFN dependen linealmente de los pesos y umbrales, por lo que un método bastante simple y eficiente es el algoritmo de los mínimos cuadrados. De este modo, los pesos y umbrales de la red se determinan mediante un proceso iterativo gobernado por la siguiente ley:

$$w_{ik}(n) = w_{ik}(n-1) - \alpha_1 \frac{\partial e(n)}{\partial w_{ik}} \tag{94}$$

$$u_k(n) = u_k(n-1) - \alpha_1 \frac{\partial e(n)}{\partial u_k} \tag{95}$$

$$Para\ k = 1,2,...,r \quad y\ para\ i = 1,..,m$$

Donde $e(n)$ es el error dado por la ecuación (93) y α_1 es la razón o tasa de aprendizaje.

Teniendo en cuenta la expresión del error y que el peso w_{ik} y el umbral u_k únicamente afectan a la neurona de salida k se obtiene que:

$$\frac{\partial e(n)}{\partial w_{ik}} = -(s_k(n) - y_k(n))\frac{\partial y_k(n)}{\partial w_{ik}} \tag{96}$$

$$\frac{\partial e(n)}{\partial u_k} = -(s_k(n) - y_k(n))\frac{\partial y_k(n)}{\partial u_k} \tag{97}$$

Derivando la salida $y_k(n)$ de la RBFN dada en la ecuación (79) respecto a los pesos y umbrales, se obtiene que:

$$\frac{\partial y_k(n)}{\partial w_{ik}} = \phi_i(n) \tag{98}$$

Donde $\phi_i(n)$ es la activación de la neurona oculta i para el patrón de entrada $X(n)$, y:

$$\frac{\partial y_k(n)}{\partial u_k} = 1 \tag{99}$$

Por tanto, las leyes dadas por las ecuaciones (95) y (96) para adaptar los pesos y los umbrales de la capa de salida de la red de base radial se pueden escribir de la siguiente forma:

$$w_{ik}(n) = w_{ik}(n-1) + \alpha_1(s_k(n) - y_k(n))\phi_i(n) \tag{100}$$

$$u_k(n) = u_k(n-1) + \alpha_1(s_k(n) - y_k(n)) \tag{101}$$

$$Para\ k = 1,2,...,r \quad y\ para\ i = 1,..,m$$

Cuando se calculan los pesos mediante la ley de aprendizaje dado por las ecuaciones (94) y (95), la convergencia es bastante rápida, consiguiendo una solución en un número pequeño de iteraciones o ciclos de aprendizaje.

CAPITULO 5

SISTEMAS DE CONTROL

Los sistemas de control automático presentan un desarrollo astronómico, debido a su gran aporte en el avance de las ingenierías y las ciencias, y como parte integral en los procesos modernos industriales y de manufactura. Estos sistemas se encuentran en máquinas – herramienta de la industria manufacturera, el diseño de sistemas de pilotos automáticos en la industria aeroespacial, diseño de automóviles y camiones de la industria automotriz, y en innumerable cantidad de procesos.

5.1 Panorama Histórico. El primer trabajo significativo en control automático fue el regulador de velocidad centrífuga de James Watt para el control de la velocidad de una máquina de vapor, en el siglo XVIII. Tiempo después Minorsky, Hazen y Nyquist aportaron trabajos importantes para el desarrollo de la teoría de control. En 1922, Minorsky Trabajo en los controladores automáticos para dirigir embarcaciones, donde se muestra que la estabilidad puede determinarse a partir de las ecuaciones diferenciales que describen el sistema.

En 1932 Nyquist diseño un procedimiento relativamente simple para determinar la estabilidad de sistemas en lazo cerrado, con base en la respuesta en lazo abierto en estado estable cuando la entrada aplicada es una senoidal. En 1934, Hazen quien introdujo el término de servomecanismos para los sistemas de control de posición, analizó el diseño de los servomecanismos con relevadores, capaces de seguir con precisión una entrada cambiante.

Durante la década de los cuarenta los métodos de la respuesta en frecuencia permitieron que los ingenieros diseñaran sistemas de control lineales en lazo cerrado que cumplieran con los requerimientos de desempeño. A finales de los años cuarenta y principios de los cincuenta, se desarrollo por completo el método del lugar geométrico de las raíces propuesto por Evans. Los métodos de respuesta en frecuencia y del lugar geométrico de las raíces, que forman el núcleo de la teoría de control clásica, conducen a sistemas estables que satisfacen un conjunto más o menos arbitrario de requerimientos de desempeño.

Durante los años comprendidos entre 1960 y 1980, se investigó a fondo el control óptimo, tanto como de sistemas determinísticos como estocásticos y el control adaptable, mediante el aprendizaje de sistemas complejos (ver Figura 48).

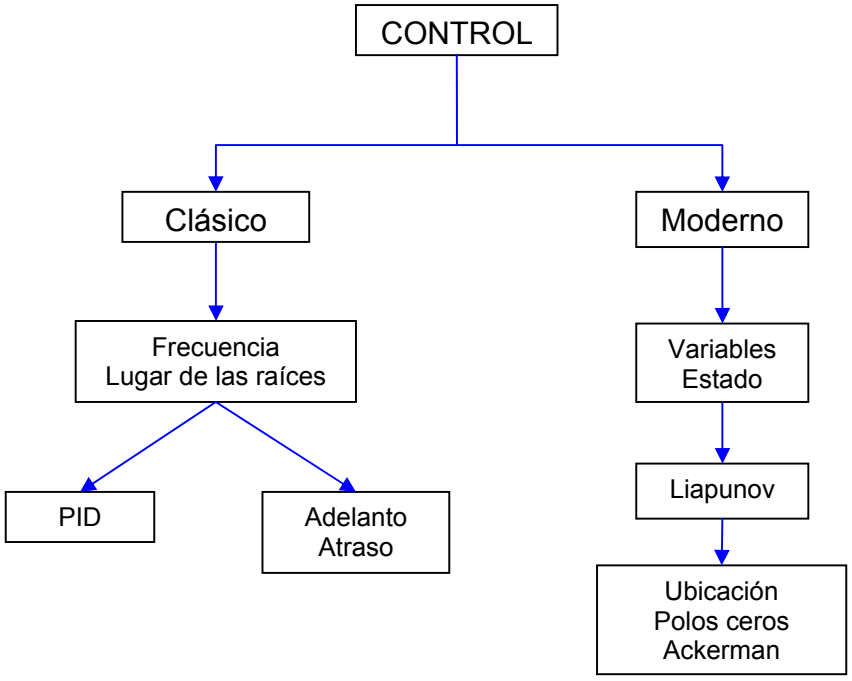


Fig. 48 Evolución del Control Automático

5.2 Sistemas de Control Realimentados. Un sistema que mantiene una relación prescrita entre la salida y la entrada de referencia, comparándolas y usando la diferencia como medio de control, se denomina *sistema de control realimentado*. Un ejemplo sería el sistema de control de temperatura de una habitación. Midiendo la temperatura real y comparándola con la temperatura de referencia (la temperatura deseada), el termostato activa o desactiva el equipo de calefacción o de enfriamiento para asegurar que la temperatura de la habitación se conserve en un nivel cómodo sin considerar las condiciones externas.

5.3 Sistemas De Control En Lazo Cerrado. Los sistemas de control realimentados se consideran también *sistemas de control en lazo cerrado*. En la práctica, los términos control realimentado y control en lazo cerrado se usan indistintamente. En un sistema de control en lazo cerrado, se alimenta al controlador la señal de error de actuación, que es la diferencia entre la señal de entrada y la señal de realimentación a fin de reducir el error y llevar la salida del sistema a un valor conveniente.

5.4 Sistemas de Control en Lazo Abierto. Los sistemas en los cuales la salida no afecta la acción de control se denominan *sistemas de control en lazo abierto.* En otras palabras, en un sistema de control en lazo abierto no se mide la salida ni se realimenta para compararla con la entrada.

Por este motivo a cada entrada de referencia le corresponde una condición operativa fija; como resultado, la precisión del sistema depende de la calibración. Ante la presencia de perturbaciones un sistema de lazo abierto no realiza la tarea deseada.

5.5 Función de Transferencia. La función de transferencia de un sistema, descrito mediante una ecuación diferencial lineal e invariante con el tiempo, se define como el cociente entre la transformada de Laplace de la salida (función de respuesta) y la transformada de Laplace de la entrada (función de excitación), bajo la suposición de que todas las condiciones iniciales son cero.

$$Función\ de\ transferencia = G(s) = \frac{L[salida]}{L[entrada]} \tag{102}$$

A partir del concepto de función de transferencia, es posible representar la dinámica de un sistema mediante ecuaciones algebraicas en s (ver Figura 49).

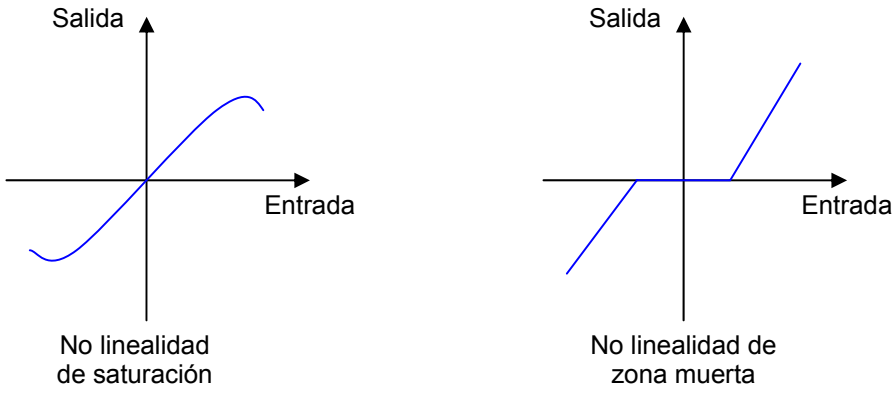


Fig. 49 Dinámica de un Sistema de Control

5.6 Diagramas de Bloques. Un diagrama de bloques de un sistema es una representación gráfica de las funciones que lleva a cabo cada componente y el flujo de señales (ver Figura 50). Tal diagrama muestra las relaciones existentes entre los diversos componentes. A diferencia de una representación matemática puramente abstracta, un diagrama de bloques tiene la ventaja de indicar en forma realista el flujo de las señales del sistema.

.Las ventajas de la representación mediante diagramas de bloques de un sistema estriban en que es fácil formar el diagrama de bloques general de todo el sistema con solo conectar los bloques de los componentes de acuerdo con el flujo de señales y que es posible evaluar la contribución de cada componente al desempeño general del sistema.

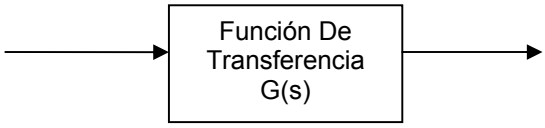


Fig. 50 Diagrama de Bloques

5.6.1 Punto suma. Un círculo con una cruz es el símbolo que indica una operación de suma. El signo de de más o menos en cada punta de flecha indica si la señal debe sumarse o restarse (ver Figura 51).

5.6.2 Punto de Ramificación. Un punto de ramificación es aquel que a partir del cual la señal de un bloque va de modo concurrente a otros bloques o puntos suma (ver Figura 51).

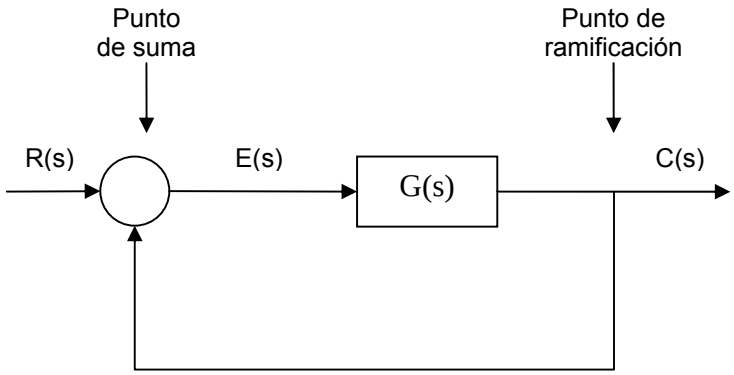


Fig. 51 Punto Suma y Punto de Ramificación

5.7 Función de Transferencia en Lazo Abierto y Función de Transferencia de la Trayectoria Directa. En la Figura 79 es posible observar que el cociente de la señal de realimentación B(s) entre la señal de error E(s) se denomina función de transferencia en lazo abierto.

$$\textit{Función de transferencia en lazo abierto} = \frac{B(s)}{E(s)} = G(s)H(s) \qquad (103)$$

El cociente entre la salida C(s) y la señal de error E(s) se denomina función de transferencia de la trayectoria directa, entonces:

$$\textit{Función de transferencia de la trayectoria directa} = \frac{C(s)}{E(s)} = G(s) \qquad (104)$$

Si la función de transferencia de la trayectoria de realimentación H(s) es la unidad, la función de transferencia en lazo abierto y la de trayectoria directa son iguales (ver Figura 52).

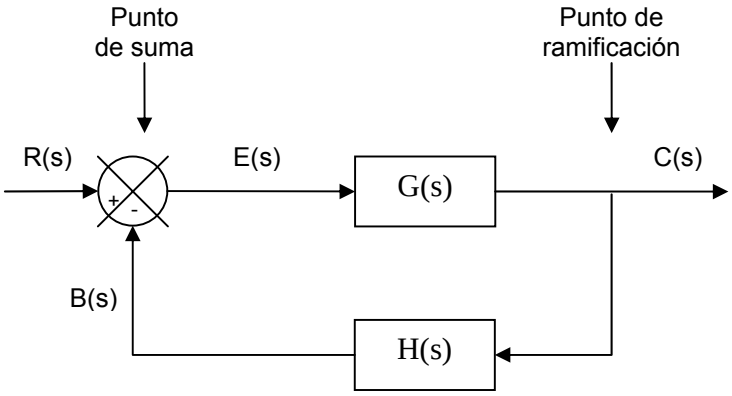


Fig. 52 Función de Transferencia en Lazo Abierto

5.8 Función De Transferencia En Lazo Cerrado. Para el sistema de la figura anterior, la salida C(s) y la entrada R(s) se relacionan del siguiente modo:

$$\begin{aligned} C(s) &= G(s)E(s) \\ E(s) &= R(s) - B(s) = R(s) - H(s)C(s) \end{aligned} \tag{105}$$

Eliminando E(s) de estas ecuaciones se obtiene:

$$C(s) = G(s)\left[R(s) - H(s)C(s)\right] \tag{106}$$

O bien:

$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)} \tag{107}$$

5.9 Teoría De Control Moderno. La tendencia moderna en los sistemas de ingeniería es hacia una mayor complejidad, debido a los requerimientos de las tareas complejas y a la elevada precisión. Los sistemas complejos tienen entradas y salidas múltiples que variar en el tiempo. Debido a la necesidad de alcanzar los requerimientos, cada vez más restrictivos en el desempeño de los sistemas de control, al aumento en la complejidad del sistema y aun acceso fácil a las computadoras de gran escala; desde 1960 se ha desarrollado la teoría de control moderna, que es un nuevo enfoque del análisis y diseño de sistemas de control complejos.

5.9.1 Análisis de la respuesta transitoria. Una vez obtenido el modelo matemático existen varios métodos para el análisis del desempeño del sistema. En la práctica, la señal de entrada para un sistema de control no se conoce con anticipación, pero es de naturaleza aleatoria, y la entrada instantánea no puede expresarse en forma analítica. En el análisis y diseño de sistemas de control, se debe tener una base de comparación del desempeño de diversos sistemas de control.

5.9.2 Señales de prueba típicas. Las señales de prueba que se usan son funciones escalón, rampa, parábola, impulso, senoidales. Con estas señales de prueba es posible realizar con facilidad análisis matemáticos y experimentales de sistemas de control, dado que las señales son funciones del tiempo muy simples. La forma de la entrada a la que el sistema estará sujeto con mayor frecuencia bajo una operación normal determina cuál de las señales de entrada típicas se debe usar para analizar las características del sistema.

5.9.3 Respuesta Transitoria y Respuesta en Estado Estable. La respuesta en el tiempo de un sistema de control consta de dos partes: la respuesta transitoria y la respuesta en estado estable. Por respuesta transitoria se conoce a la que va del estado inicial al estado final. Por respuesta en estado estable, se conoce a la manera en la cual se comporta la salida del sistema conforme t tiende a infinito.

5.9.4 Estabilidad Absoluta, Estabilidad Relativa, y Error en Estado Estable. Al diseñar un sistema de control se debe estar en la capacidad de predecir su comportamiento dinámico a partir del conocimiento de las componentes. La característica más importante del comportamiento dinámico de un sistema de control es la estabilidad absoluta. Es decir, si el sistema es estable o no. Un sistema está en equilibrio si, en ausencia de cualquier perturbación o entrada, la salida permanece en el mismo estado. Y es inestable si la salida diverge sin límite a partir de su estado de equilibrio cuando el sistema está sujeto a condiciones iniciales **[18]**.

5.9.6 Clasificación de los Controladores Industriales

- De dos posiciones (On/Off)
- Proporcionales
- Integrales
- Proporcionales-integrales
- Proporcionales-derivativos
- Proporcionales-integrales-derivativos

CAPITULO 6

CONTROL INTELIGENTE

La Teoría de Control se fundamenta en la búsqueda de mecanismos que permitan interactuar sobre los sistemas con el fin que ciertas variables observadas de éste se comporten según unas pautas prefijadas. Este objetivo se pretende lograr sobre cualquier sistema, independiente de su complejidad.

Las técnicas de control no lineal se han extendido poco a poco con el fin de contemplar todas las no linealidades que se hallan presentes en dichos sistemas y que lo alejan del modelo lineal de mayor sencillez. Sin embargo, estas herramientas analíticas han sido superadas por técnicas que tratan de obtener el control sobre el sistema a través de funciones que se inspiran en el razonamiento humano. A estas últimas se las ha denominado *Control Inteligente*, considerándose éste como un campo de investigación, que por necesidad se hallar ligado a otros enfoques clásicos de la teoría de control.

6.1 Evolución histórica de los sistemas de control inteligente. Hablar de control inteligente sólo puede ser entendido desde una perspectiva histórica de la evolución que han tenido estas propuestas, desde el punto de vista formal, de la experiencia obtenida por la aplicación de las mismas, y de la evolución de tecnologías subsidiarias, como lo es sin lugar a dudas los sistemas de computación.

Dos corrientes de pensamiento han coexistido desde este planteamiento: la “escuela simbólica” y la “escuela sub-simbólica”. La primera basa su planteamiento en los *sistemas expertos*. La LB ha aportado la posibilidad de trabajar con razonamientos e incertidumbre, y además ofrece una forma de convertir estos razonamientos en algoritmos numéricos. En este último enfoque se ha movido la “escuela sub-simbólica”, que basa la representación de la información, en la extracción de vectores de características que se derivan de la observación.

En muchos casos estas técnicas han tenido una inspiración biológica. Entre estas se encuentra la teoría de las RNA, que ha proporcionado un marco formal donde es posible la obtención de algoritmos de eficacia probada, y además ha proporcionado un nexo de unión con La LB y los sistemas expertos, denominados *sistemas* ***neuro-borrosos***.

6.2 Características de los sistemas inteligentes de control. Actualmente, están aceptados criterios que definen y delimitan las técnicas de control inteligente, entre los que se citan:

- Son aplicados a sistemas de difícil modelado, con el propósito de lograr el control del sistema en uno de los siguientes casos:
    - ✓ Sin necesidad del conocimiento del modelo matemático que define el comportamiento del sistema.

- ✓ A partir de un conocimiento vago o aproximado del comportamiento del sistema.
- ✓ A partir de esquemas de control clásico sobre un modelo aproximado del sistema que son ajustados o adaptados por medio de una supervisión llevada a cabo por el control inteligente.
- ✓ La información que manejan debe ser tanto numérica como simbólica, puede proceder de lecturas realizadas sobre el entorno a través de equipos de medida, y de razonamientos sencillos enunciados a través del lenguaje natural.

- Capacidad de aprendizaje. Los sistemas inteligentes deben proporcionar cierta *plasticidad* a su estructura que permita su adaptación o modificación en función del comportamiento observado.
- Capacidad de generalización. La razón principal que justifica el empleo de los sistemas inteligentes es la capacidad de generalización en la toma de decisiones, de manera que pueda actuar de forma adecuada ante situaciones nuevas, no previstas durante el diseño. De esta forma se lograr la plena autonomía del sistema inteligente.

6.3 Control directo y control indirecto. Cuando se aplican técnicas inteligentes para lograr el control de sistemas complejos, existen dos alternativas posibles:

- Control directo. El sistema inteligente sustituye en la cadena de regulación al regulador clásico, debido a que se dispone de un número mayor de entradas que permitan una observación más completa de las condiciones del entorno.
- Control indirecto. El control se realiza por alguna técnica clásica, cumpliendo el sistema inteligente la función de la sintonización de dicho controlador a partir de la observación del medio.

El esquema de control directo, por tanto, cede todo el control al sistema inteligente. El esquema de control indirecto opta, por el contrario, por emplear éste como un elemento de supervisión y ajuste. Este último enfoque es útil cuando los sistemas presentan un comportamiento local próximo al lineal, y que, por tanto, puede ser regulado por técnicas convencionales.

6.4 Sistemas borrosos. Sus principios fueron establecidos por Lofti Zadeh **[3]** a partir de los denominados *CB*. Con la LB, a diferencia de la lógica Clásica, es posible modelar los modos imprecisos de razonamiento que juegan un papel esencial en la habilidad del ser humano de tomar decisiones racionales en un entorno de incertidumbre e imprecisión.

6.4.1 Sistemas borrosos y reguladores borrosos. Los sistemas borrosos son aquellos que guardan una relación directa con los conceptos borrosos y La LB. Es decir, sistemas en los que el conocimiento viene descrito en términos borrosos. Generalmente, la información que se dispone de un sistema es la procedente de sensores: *información numérica* y la procedente de expertos: *información lingüística*. Si se desea aprovechar toda esta información, es necesario disponer de un medio para tratar con la información lingüística. En muchos casos, la información numérica puede ser empleada para proporcionar adaptación al sistema borroso, a partir del sistema, inicialmente, establecido por la información lingüística.

Para el análisis de los sistemas borrosos, es adecuado realizar consideraciones sobre las razones que hacen estos apropiados para el control. En primer lugar, se exponen algunas razones teóricas:

- Como regla general, cualquier problema en ingeniería debe ser afrontado aprovechando al máximo toda la información disponible. Si el modelo matemático es demasiado complejo de obtener, las principales fuentes de información serán: 1) los sensores, que proporcionarán medidas de las variables que se consideren más importantes, y 2) expertos que proporcionen descripciones lingüísticas tanto acerca del sistema, como de las acciones de control. Los SIB permiten incorporar de una forma eficaz y sistemática esta información lingüística.
- El control borroso no precisa de un modelo matemático del sistema que se desea controlar, lo cual lo hace muy adecuado para tratar con los problemas cada vez más complejos a los que se enfrenta el experto en control.
- El control borroso proporciona reguladores no lineales, suficientemente, generales como para generar cualquier acción de control no lineal deseada.

Desde un punto de vista práctico, existen también razones que justifican el uso de SIB:

- El control borroso es sencillo de entender. En contraposición a las herramientas matemáticas cada vez más complejas que se vienen empleando en la teoría convencional de control, el control borroso es fácil de entender para los no especialistas gracias a su habilidad para emular las estrategias de control utilizadas por los seres humanos.
- El control borroso es sencillo de implementar. Los sistemas borrosos permiten un grado muy alto de paralelismo. Muchos circuitos integrados VLSI borrosos han sido ya desarrollados. En Jamshidi **[19]** se puede encontrar una extensa relación de herramientas hardware/software de LB disponibles en el mercado.
- El control borroso tiene un desarrollo relativamente barato. Al ser sencillo de entender, su coste en términos de *software* es bajo. Por otra parte, al ser sencillo de implementar, su coste en términos de *hardware* es también relativamente bajo. A esto se une la gran disponibilidad de herramientas para el desarrollo de reguladores borrosos. El control borroso, tiene, por tanto, una relación costo/beneficio muy atractiva.

6.4.2 Sistemas con autoaprendizaje. Los sistemas borrosos adaptativos precisan de alguna técnica que permita el aprendizaje de las reglas que gobernarán este sistema. Existen un conjunto de sistemas que se han desarrollado con el fin de la extracción de un conocimiento o aprendizaje de la información disponible del entorno, como pueden ser Las RNA, los estimadores, etc. La sinergia entre estas teorías y La LB da lugar a los sistemas borrosos adaptativos.

La clasificación fundamental entre estos sistemas es aquella que distingue entre dos tipos de aprendizaje:

- *Aprendizaje supervisado*: el proceso de aprendizaje se realiza por medio de un entrenamiento controlado por un agente externo (supervisor, maestro) que

comprueba la salida de la red y en el caso en el que no coincida con la deseada, procede a modificar el sistema por medio de un algoritmo de adaptación de sus parámetros. Los aprendizajes más importantes de este tipo son el de corrección de error, por refuerzo y el estocástico.

- *Aprendizaje no supervisado*: en este caso, el sistema debe ser capaz de encontrar las relaciones íntimas que presentan los datos sin que se le indique qué respuesta debe generar ante una entrada concreta. Generalmente este aprendizaje lleva a una categorización de los datos de entrada. Los aprendizajes de este tipo principales son el hebbiano y el competitivo y cooperativo.

6.4.3 Identificación y control de sistemas por medio de RNA. Problemas habituales en los sistemas de control tales como no linealidades, retardos, saturaciones, parámetros variantes con el tiempo y otras complejidades son de difícil solución. Los avances logrados en las teorías de control robusto y adaptativo han mostrado utilidad en casos particulares, pero todavía existen situaciones en las que no ha sido posible hallar una solución al problema por medio de estas técnicas de control. Las RNA presentan la capacidad de generalización y no necesitan disponer de un modelo matemático del sistema que procesan.

6.4.4 Control de sistemas. Tal y como es conocido por la teoría de control adaptativo, la identificación y el control de un sistema están estrechamente ligados. Un identificador del sistema es conocido también como *modelo directo,* porque a partir de los estados anteriores del sistema, se determina la salida en el instante posterior. El modelo inverso, por el contrario, calcula la entrada adecuada al sistema cuando se le presenta la salida deseada en el instante posterior. Este último sistema cumple con la función de regulador o controlador y puede ser realizado, al igual que en el caso anterior por medio de una RNA.

El regulador decide en cada instante qué señal se debe introducir a la entrada del sistema para obtener a la salida la señal deseada. Este tipo de control se conoce generalmente como control feedforward o control con prealimentación. Generalmente, estos sistemas incluirán un bucle de realimentación adicional que asegure la estabilidad del sistema **[20]**. El paradigma del control adaptativo por modelo de referencia (MRAC) puede ser extendido al caso de las RNA **[21]**. Una RNA es empleada para identificar el sistema, mientras otra genera la entrada a la planta. El objetivo es lograr que el sistema siga la salida de un modelo de referencia. En la Figura 53 se ha representado la estructura básica de control de este modelo.

6.4.5 Sistemas borrosos adaptativos y sus aplicaciones al control. Existen un gran número de trabajos que combinan los sistemas de LB con alguna técnica que permita la adaptación de estos al entorno. El objetivo es aunar la posibilidad de La LB de procesamiento de información imprecisa con los sistemas con capacidad de aprendizaje. La potencia de un sistema borroso estriba en gran medida en las reglas borrosas proporcionadas por los expertos y que van a formar su base de conocimientos.

Estas reglas, dadas siempre de una forma vaga e imprecisa llegar a ser difíciles de ajustar de tal forma que describa, adecuadamente, el comportamiento a modelar. Sin embargo, por medio de sistemas con capacidad de aprendizaje, pueden ser refinadas o, lo que es más importante, es posible extraer de forma automática nuevas reglas del entorno que no

habían sido descritas por el experto. A lo que se denomina como: Sistema Borroso Adaptativo.

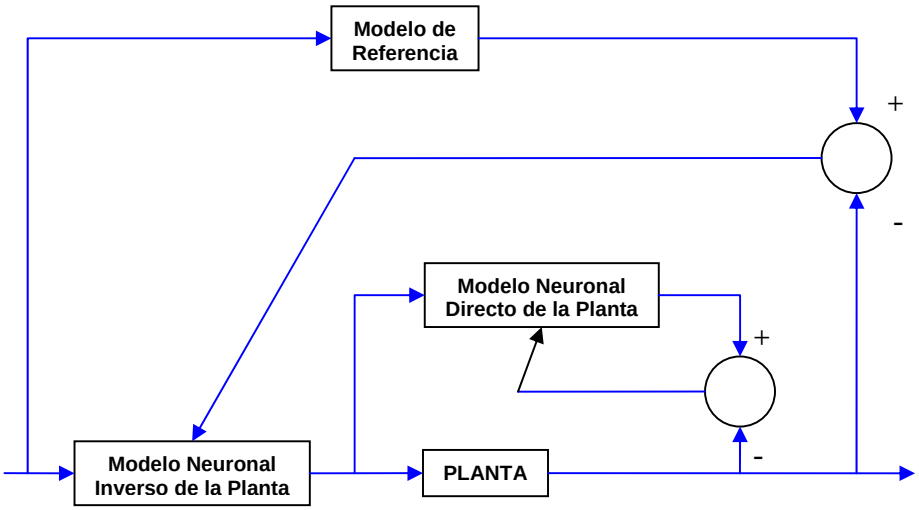


Fig. 53 Control Adaptativo por Modelo de Referencia Realizado con RNA

6.4.6 Paradigmas. Los paradigmas de los sistemas borrosos adaptativos son numerosos. Cada autor ha propuesto un modelo diferente de sistema, destacando en algunos casos el paralelismo con otros sistemas tales como Las RNA, mientras que en otros se enuncia un algoritmo y un modelo específico. Tratar de clasificar estos modelos es una tarea, prácticamente imposible, porque muchos de los sistemas propuestos se ajustan a diferentes categorías. A continuación, se destacan los principales paradigmas a partir de los cuales la mayoría de los autores han partido para diseñar el sistema borroso adaptativo.

a) Sistemas borrosos representados como una RNA *feedforward*. Clasificación según el método de ajuste o aprendizaje:

- Redes de dimensión estática: aprendizaje por retropropagación; mínimos cuadrados (FBF); table-lookup; otros algoritmos de optimización.
- Redes auto-organizativas: RBFN con aprendizaje competitivo; RNA FAM; otros algoritmos de aprendizaje con autoorganización

b) RNA que procesan señales *borrosas* y/o tienen pesos borrosos:

- RNA borrosas con entradas reales/borrosas, pesos reales/borrosos.
- Mapas cognitivos borrosos (FCM).

6.4.7 Reguladores borrosos adaptativos. La aplicación más sobresaliente de estos sistemas es el control. Los SIB han sido ideados para trabajar en situaciones en las que existe una gran incertidumbre o variaciones desconocidas en los parámetros de la planta y en su estructura. Generalmente, el objetivo básico del control adaptativo es mantener un

comportamiento consistente del sistema en presencia de estas incertidumbres. Por tanto, los sistemas borrosos avanzados deben ser adaptativos.

El modelo de control adaptativo más conocido es el denominado control por modelo de referencia (MRAC). La clasificación habitual de estos sistemas de control adaptativo se dividen en dos categorías: reguladores adaptativos directos e indirectos.

- **Regulador adaptativo directo**: los parámetros del regulador son ajustados de manera que se reduzca algún tipo de norma del error de salida entre la planta y el modelo de referencia.
- ***Regulador adaptativo indirecto***: se estiman los parámetros de la planta y el regulador se ajusta suponiendo que estos parámetros estimados son los reales.

6.4.8 Métodos Específicos de La LB. Tong **[22]** propuso un método para establecer las reglas borrosas de control a partir de los datos de entrada-salida del proceso, por medio de la "identificación borrosa". Este método resultaba, excesivamente heurístico, subjetivo, manual y difícil de adaptar en sistemas multivariables. El primer regulador borroso auto-organizativo (SOFC) fue descrito por Procyk **[23]**. Se trataba de un sistema con una estructura jerárquica basada en dos reglas base. Su mecanismo de aprendizaje se rige por un índice que evalúa el comportamiento de éste.

6.5 Métodos Derivados de la Teoría de RNA. La gran similitud entre la estructura de un sistema borroso y la de una RNA ha dado lugar a una extensísima literatura sobre esta materia. El *controlador* ***neuro-borroso*** aparece en Lee **[24]**. Algunos de los modelos más importantes son: el propuesto por Lin **[25]**; en Horikawa **[26]** y Wang **[27]**, que plantea un paradigma de sistema borroso-neuronal basado en las funciones sigmoidales con el algoritmo backpropagation; y el propuesto por Simpson **[28]** que indica cómo extraer reglas borrosas para clasificadores, por medio de regiones borrosas variables. Para ello se vale de un conjunto de hipercubos dinámicos que pertenecen a cada clase de entrada.

Ishibuchi en el año 1993 **[29]**, propone una RNA donde el problema de la clasificación viene dado, tanto por patrones como por un conjunto de reglas borrosas. Kuo en el año 1993 **[30]**, plantea un modelo de clasificador borroso basado en una RNA de cuatro capas, con método de clasificación cimentado en la distancia euclidiana ponderada. Lin en el año 1994 **[31]**, presenta un modelo de SIB adaptado a la estructura de una RNA. Para la construcción de la RNA se propone: a) un algoritmo de aprendizaje en dos fases: una no supervisada y otra supervisada por el método del gradiente descendiente; b) un algoritmo de aprendizaje on-line de la estructura y parámetros, empleando para ello un índice de "similitud borrosa".

6.6 Control Inteligente Basado en Modelos Locales. Cada vez más se trabaja con sistemas de gran complejidad. Tanto las plantas industriales, como los dispositivos empleados precisan de un control que satisfaga las demandas estrictas motivadas por exigencias económicas y de protección medioambiental.

La estrategia que proponen los sistemas de control basados en modelos locales (Johansen **[32]**), consiste en dividir un problema de gran complejidad en sub-problemas más sencillos que son analizados de forma individual. El éxito de este método consiste en la elección de los parámetros adecuados que permitan realizar dicha partición.

CAPITULO 7

MARCO TEÓRICO ESPECÍFICO

7.1 Sistemas Neuro-Borrosos. Son sistemas híbridos que combinan técnicas de RNA y SIB. Asociando de este modo la capacidad de aprendizaje de las RNA con la tolerancia a fallos, interpretabilidad y robustez de los SIB. Además permiten la integración de conocimiento, siendo posible extraer el conocimiento incluido en la RNA en formato de reglas borrosas (por eso son considerados modelos de "caja gris").

Actualmente, existe una gran cantidad de propuestas de combinación de la capacidad de aprendizaje de Las RNA y del procesamiento de información imprecisa de La LB. Esta combinación se refiere a aspectos como el uso de neuronas borrosas, cuyo funcionamiento se describe en función de operadores borrosos (producto y suma), en lugar de aritméticos; la adaptación borrosa de algoritmos de aprendizaje de RNA conocidas, como el Perceptrón multicapa; la simulación de RNA de sistemas expertos dinámicos que integran opiniones borrosas de varios especialistas; el almacenamiento y evaluación con RNA de las reglas utilizadas en sistemas de control borroso; la generación y adaptación de FM a CB mediante RNA; etc.

La LB y Las RNA, presentan propiedades computacionales particulares que las hacen adecuadas para cierto tipo de problemas. Mientras las RNA ofrecen ventajas como: el aprendizaje, adaptación, tolerancia a fallos, paralelismo y generalización, no son buenas para explicar como obtuvieron sus decisiones. Caso contrario con los sistemas borrosos, los cuales razonan con información imprecisa a través de un mecanismo de inferencia bajo incertidumbre lingüística, son buenos al explicar sus decisiones, pero no pueden adquirir de manera automática las reglas que usan para tomarlas. Para esclarecer las diferencias, en la Tabla 5 se explican algunas de ellas.

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

Tabla 5. Diferencias entre LB y RNA

Algunos sistemas que han logrado unir con éxito características de La LB y las RNA son descritos a continuación:

7.1.1 ANFIS (Adaptive Neuro Fuzzy Inference System). Es un método que permite sintonizar o crear las reglas de un SIB, al usar el algoritmo de entrenamiento de retropropagación a partir de la recopilación de datos de un proceso. Su arquitectura es funcionalmente equivalente a una base de reglas tipo Sugeno.

7.1.2 FSOM (Fuzzy Self-Organizing Map). Consiste en un Sistema Borroso optimizado a partir de la técnica de los mapas auto-organizados de Kohonen.

7.1.3 NEFCLASS. El algoritmo NEFCLASS está basado en la estructura del Perceptrón Multicapa (MLP) cuyos pesos son modelados por CB. Así se preserva la estructura de una RNA, pero se permite la interpretación del sistema resultante por el sistema borroso asociado.

7.2 Aprendizaje de un Sistema Neuro-Borroso. Un sistema Neuro-Borroso consiste en esencia en un sistema borroso tradicional, excepto que cada etapa, se representa por una capa de neuronas que son provistas por capacidades de aprendizaje para optimizar el conocimiento del sistema (ver Figura 54).

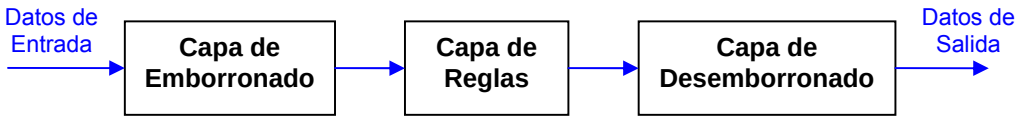


Fig. 54 Diagrama de Bloques de un Sistema Neuro-Borroso

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

En la Figura 55 el grado de pertenencia que indica la certeza de " X_1 es grande" es 0.6, " X_1 es mediano" es 0.4, y " X_1 es pequeño", es 0.0. Las salidas de estas neuronas, que equivalen a FM, son conectadas a la capa de reglas borrosas, como lo especifiquen las reglas borrosas y a través de enlaces con pesos que representan el proceso de agregación de las variables lingüísticas del proceso de entrada.

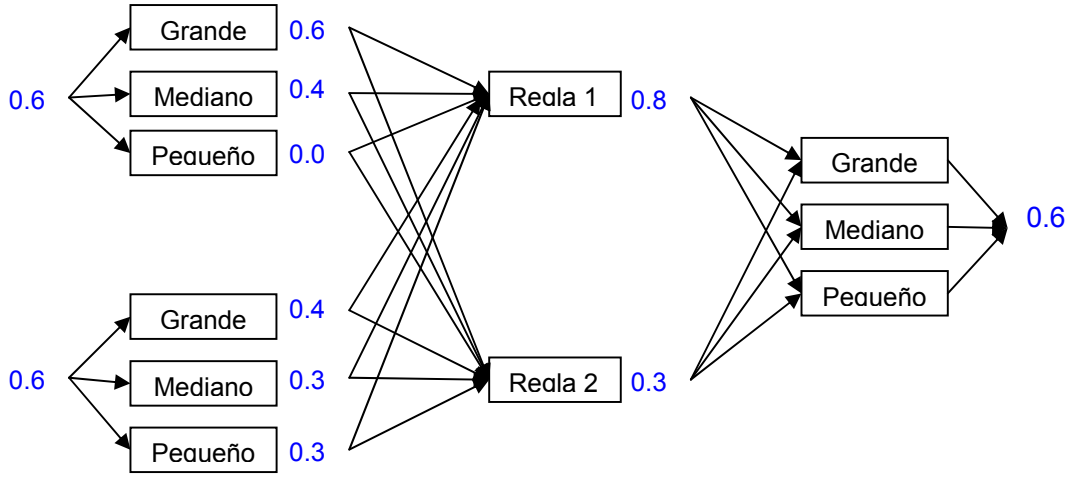


Fig. 55 Estructura de un Sistema Neuro-Borroso

La capa de reglas borrosas representa la base de reglas borrosas; cada neurona representa una regla borrosa del tipo "If-Then". Las salidas de las neuronas están conectadas a la capa de desemborronado a través de enlaces con pesos; los pesos de estos enlaces representan la significancía relativa de las reglas asociadas con las neuronas. Sus valores pueden ser asignados, de acuerdo al conocimiento a priori o inicializados como 1.0, y luego entrenadas donde se refleja la importancia real sobre Las FM de salida contenidas en la capa de desemborronado.

La función de la capa de emborronado es la evaluación de las reglas; en cada consecuente "Entonces Y es B" como FM de salida representa una neurona. La certeza de cada consecuente es calculada, según se ajustan las reglas que tienen el mismo consecuente. Los pesos de cada enlace de salida de estas neuronas representan los centros de área de cada FM del consecuente y se pueden entrenar, la salida final es calculada usando algún método de desemborronado.

Para realizar el entrenamiento de los sistemas Neuro-Borrosos la estructura mostrada en la Figura 56 es configurada con valores iniciales, obtenidos del conocimiento a priori, y luego ajustados utilizando un algoritmo de entrenamiento tal como el de Backpropagation, de la siguiente manera:

1. Presentar una muestra de entrada, y calcular la salida correspondiente.
2. calcular el error entre la salida y el valor deseado.
3. Ajustar los pesos de las conexiones y las FM.
4. si el error es mayor que la tolerancia, volver al paso 2, en otro caso el entrenamiento ha finalizado.

7.3 Limitaciones De Los Sistemas Neuro–Borrosos. Entre las principales limitaciones de los sistemas Neuro-Borrosos se encuentran:

- Número pequeño de entradas (curso de la dimensionalidad: crecimiento geométrico de la complejidad según el número de entradas).

- Dificultad para aprender la estructura de las reglas. Generalmente, sólo aprenden la forma de las FM y los coeficientes del consecuente (en TSK).
- Dificultad para tratar funciones no diferenciables.
- Problemas de convergencia: caída en óptimos locales.
- Problemas de sobre aprendizaje: Error de aproximación (conjunto de entrenamiento) mucho menor que el de generalización (conjunto de validación).

7.4 RNA Adaptativas: Arquitecturas y Algoritmos de Aprendizaje. Las arquitecturas y los tipos de aprendizaje de las RNA adaptativas, son de hecho un superconjunto de todas las clases de RNA tipo Feedforward, con capacidad de aprendizaje supervisado. Este tipo de redes contiene una estructura conformada por nodos los cuales están interconectados. Más aún todos o parte de los nodos son adaptativos y dependen de parámetros iniciales (S). La regla de aprendizaje que aplica sobre estos nodos debe ser dinámica en procura de minimizar el error global.

La regla básica de aprendizaje de las redes adaptativas se basa en el gradiente descendente y la regla de la cadena, que fue propuesta por Werbos en el año de 1970. Debido a que la regla de aprendizaje básica se basa en el método del gradiente, es notoria la lentitud y la facilidad para quedar atrapado en un mínimo local, se hace imperiosa la necesidad de implementar una regla de aprendizaje hibrido, que acelere el proceso de aprendizaje.

7.4.1 Arquitectura y Regla de Aprendizaje Básica de una Red Adaptativa. Una RNA adaptativa (ver Figura 56), es una RNA multicapa con conexiones hacia delante, en la cual cada nodo lleva a cabo una función particular sobre las señales que llegan a él. Las funciones dependen de un conjunto de parámetros propios de cada nodo.

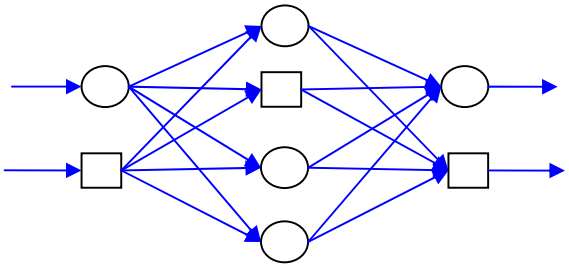


Fig. 56 Arquitectura de una RNA Adaptativa

Para reflejar las capacidades adaptativas de los nodos, se utilizaron círculos y cuadrados para diferenciarlos, dentro de las RNA adaptativas. El cuadrado es un nodo adaptativo que posee parámetros, mientras que el círculo es un nodo fijo y por ende no posee parámetros. El conjunto de parámetros de una RNA adaptativa es la unión de los parámetros de todos los nodos adaptativos.

Los parámetros de los nodos adaptativos se derivan del conjunto de datos de entrenamiento y son actualizados por medio de un aprendizaje basado en el gradiente, este proceso es descrito a continuación:

Suponga que se tiene una RNA adaptativa con L capas y la $k-th$ capa tiene $\#(k)\ nodos$. Es posible denotar el nodo de la $i-th$ posición en la $k-th$ capa por (k,i), y la función de nodo (o salida del nodo) por O_i^k. Donde la salida del nodo depende de las señales de entrada y el conjunto de parámetros propios del nodo:

$$O_i^k = O_i^k(O_i^{k-1},...,O_{\#(k-1)}^{k-1}, a, b, c,...) \tag{108}$$

Donde $a,b,c...$ son los parámetros pertenecientes a este nodo. Asumiendo que el conjunto de datos de entrenamiento dado tiene P entradas, es posible definir el error medio (o función energía) para la $p-th$ $(1 \le p \le P)$ entrada de los datos de entrenamiento, como la suma de los errores cuadrados:

$$E_p = \sum_{m=1}^{\#L} (T_{m.p} - O_{m.p}^L)^2 \tag{109}$$

Donde $T_{m.p}$ es el $m-th$ componente de $p-th$ vector de salida deseada, y $O_{m,p}^L$ es el $m-th$ del actual vector de salida producido por la presentación del $p-th$ vector de entrada. El error medio total es dado por $E = \sum_{p=1}^{P} E_p$.

Siguiendo el orden de desarrollo del procedimiento de aprendizaje, se debe implementar el método del gradiente descendente en E sobre el espacio de parámetros; primero se calcular la rata de error $\frac{\partial E_p}{\partial O}$ para el $p-th$ dato de entrenamiento y para cada nodo de salida O. La rata de error para un nodo de salida como (L,i) puede ser calculada directamente con la ecuación (109):

$$\frac{\partial E_p}{\partial O_{i,p}^L} = -2(T_{i,p} - O_{i,p}^L) \tag{110}$$

Para los nodos internos como (k,i), la rata de error es derivada por la regla de la cadena:

$$\frac{\partial E_p}{\partial O_{i,p}^k} = \sum_{m=1}^{\#(k+1)} \frac{\partial E_p}{\partial O_{m.p}^{k+1}} \frac{\partial O_{m.p}^{k+1}}{\partial O_{i,p}^k} \tag{111}$$

Donde $1 \le k \le L-1$, esto asegura que la rata de error de un nodo interno puede ser expresada como una combinación lineal de las ratas de error de los nodos de la siguiente capa. Por lo tanto para todo $1 \le k \le L$ y $1 \le i \le \#(k)$ es posible encontrar $\frac{\partial E_p}{\partial O_{i,p}^k}$, con las

ecuaciones (110) y (111). Ahora si α es un parámetro de la RNA adaptativa dada, entonces:

$$\frac{\partial E_p}{\partial \alpha} = \sum_{O^* \in S} \frac{\partial E_p}{\partial O^*} \frac{\partial O}{\partial \alpha} \tag{112}$$

Donde S es el conjunto de nodos cuya salida depende de α. Entonces la derivada del error medio total con respecto a α es:

$$\frac{\partial E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha} \tag{113}$$

Por consiguiente, la formula de actualización para el parámetro genérico α es:

$$\Delta \alpha = -\eta \frac{\partial E}{\partial \alpha} \tag{114}$$

En la cual η es la rata de aprendizaje que puede ser expresada como:

$$\eta = \frac{k}{\sqrt{\sum_{\alpha} \left(\frac{\partial E}{\partial \alpha} \right)^2}} \tag{115}$$

Donde k es la medida del paso, para variar la velocidad de convergencia del gradiente es necesario aumentar o disminuir el paso.

Actualmente, existen dos paradigmas de aprendizaje para redes adaptativas, en el aprendizaje por lotes (o aprendizaje Off-Line), la formula de actualización para los parámetros α se basan en la ecuación (113) y la acción de actualización toma lugar sólo después de que el conjunto de datos de entrenamiento completo ha sido presentado. Por otro lado si se quieren actualizar los parámetros de manera inmediata después de cada par entrada-salida presentado, la formula de actualización es basada en la ecuación (112) y esta se refiere a un aprendizaje por patrones (o aprendizaje On-Line).

7.4.2 Regla de Aprendizaje Hibrido: Aprendizaje por Lotes (Off-Line). Lo común es aplicar el método del gradiente para identificar los parámetros en una RNA adaptativa, pero este método, generalmente, es lento y queda atrapado con facilidad en un mínimo local. Lo que se propone es una regla de aprendizaje hibrido **[44],** que combina al método del gradiente con el estimador de mínimos cuadrados (LSE) para identificar los parámetros de la RNA adaptativa.

Por simplicidad se asume que la RNA adaptativa bajo consideración tiene una salida:

$$output = F(\vec{I}, S) \tag{116}$$

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

$$S = S_1 \oplus S_2 \tag{117}$$

(Donde $\oplus$ representa la suma directa) tal que $H \circ F$ es lineal en los elementos de S_2, entonces sobre H se aplica la ecuación (116), que define:

$$H(output) = H \circ F(\vec{I}, S) \tag{118}$$

Que es lineal en los elementos de S_2. Luego se proporcionan valores de elementos de S_1, ahora es posible presentar los datos de entrenamiento P en la ecuación (118) y obtener una formula matricial:

$$AX = B \tag{119}$$

Donde X es un vector desconocido cuyos elementos son parámetros de S_2. Al permitir que $|S_2| = M$, entonces las dimensiones de $A, X \ y \ B$ son: $P \times M, M \times 1 \ y \ P \times 1$ respectivamente. Desde que P (número de pares de datos de entrenamiento) sea mayor que M (número de parámetros lineales), este problema se convierte en sobredeterminado y no es posible encontrar una solución exacta a la ecuación (119). Es preferible utilizar el Estimador de Mínimos Cuadrados (LSE) de X, X^*, si se busca minimizar el error cuadrado $\|AX - B\|^2$, al final queda convertido en un problema estándar que es formulado en regresión lineal, filtros adaptativos y procesamiento de señales. La formula más conocida para hallar X^* es usar la seudo inversa de X:

$$X^* = (A^T A)^{-1} A^T B \tag{120}$$

Donde A^T es la transpuesta de A, y $(A^T A)^{-1} A^T$ es la seudo inversa de A, si $(A^T A)$ es no singular. Mientras la ecuación es concisa en la notación, esta es muy costosa en cálculos computacionales, más si se trata de encontrar la inversa de la matriz, y sobre todo si $(A^T A)$ es singular. Para evitar este proceso se propone emplear formulas secuenciales para computar los LSE de X.

El método secuencial de LSE es más eficiente (especialmente cuando M es pequeño) y puede ser modificado con facilidad en una versión On-Line para sistemas con

características cambiantes. Específicamente, la $i-th$ fila de la matriz A definida en la ecuación (119) es a_i^T y el $i-th$ elemento de B es b_i^T, entonces X puede ser calculada de forma iterativa usando formulas secuenciales adoptadas ampliamente en la literatura **[34], [35], [36]**:

$$X_{i+1} = X_i + S_{i+1}a_{i+1}(b_{i+1}^T - a_{i+1}^T X_i)$$
$$S_{i+1} = S_i - \frac{S_i a_{1+1} a_{i+1}^T S_i}{1 + a_{i+1}^T S_i a_{i+1}} \quad (121)$$

Donde S_i es a menudo llamada matriz de covarianza y el estimador de mínimos cuadrados X^* es igual a X_p. Las condiciones iniciales de la ecuación (121) son: $X_o = 0$ y $S_o = \gamma I$, donde γ es un número positivo grande y I es la matriz identidad de dimensión $M \times M$. Cuando se trabaja con RNA adaptativas multi – salida, (el *output* en la ecuación (116) se convierte en un vector columna), es posible aplicar la ecuación (121), excepto cuando b_i^T sea el $i-th$ fila de la matriz B.

De lo anterior se concluye que es probable combinar el método del gradiente con el estimador de mínimos cuadrados en una RNA adaptativa. Cada época de este procedimiento de aprendizaje hibrido es compuesta por un paso hacia atrás y un paso hacia delante. En el paso hacia delante, son suministrados los datos de entrada y las señales funcionales, para calcular la salida de cada nodo hasta que las matrices $A\ y\ B$ son obtenidas, y los parámetros en S_2 son identificados por la formula de mínimos cuadrados secuenciales (121). Después de identificar los parámetros en S_2, las señales funcionales se propagan hacia delante hasta que el error medio es calculado. En el paso hacia atrás, la rata de error es propagada desde la salida hasta la entrada, y los parámetros de S_1 son actualizados por el método del gradiente según la ecuación (114).

7.4.3 Regla de Aprendizaje Hibrido: Aprendizaje por patrones (On-Line). Si los parámetros son actualizados después de cada presentación de datos, se tiene un aprendizaje por patrones o aprendizaje en línea. Este paradigma de aprendizaje es primordial para la identificación de parámetros en línea en sistemas con características cambiantes. Es necesario que decaiga el efecto de los viejos pares de datos cuándo los nuevos pares de datos se presenten. Un método simple para solucionar el problema es adicionar un factor λ a la formula secuencial original, que le de prioridad a los nuevos datos presentados, según su relevancia dentro del sistema:

$$X_{i+1} = X_i + S_{i+1}a_{i+1}(b_{i+1}^T - a_{i+1}^T X_i)$$
$$S_{i+1} = \frac{1}{\lambda}[S_i - \frac{S_i a_{1+1} a_{i+1}^T S_i}{\lambda + a_{i+1}^T S_i a_{i+1}}] \quad (122)$$

Donde el valor de λ está entre 0 y 1.

CAPITULO 8

METODOLOGÍA PROPUESTA

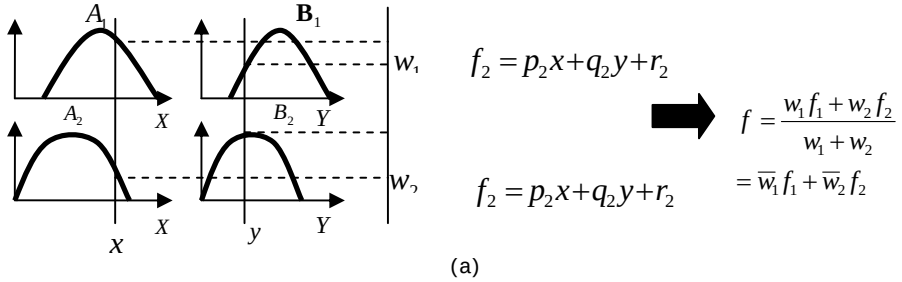
Dado el alcance obtenido por las Tecnologías Adaptativas en el ámbito nacional e internacional, surgió la necesidad de su estudio e implementación, para incrementar el interés por generar aplicaciones y herramientas que permitieran el desarrollo de estás teorías y su aplicabilidad.

Netfuz 1.0 surge como una herramienta basada en RNA adaptativas, descritas con antelación, que aprovechan su carácter flexible para implementar una equivalencia con los SIB tipo Sugeno, y cuyo propósito es la generación automática de reglas borrosas y el ajuste de los parámetros en las FM. Apoyando su fundamento en la regla de aprendizaje hibrido, y tomando de referencia autores como: Kosko, Zadeh y Jang, que determinaron los principios básicos de los sistemas Neuro-Borrosos, los cuales combinan las bondades propias de las RNA y la LB **[37]**.

8.1 Arquitectura Netfuz 1.0. Por simplicidad, se asume el SIB bajo consideración con dos entradas x, y y una salida z. Se supone que la base de reglas contiene dos reglas borrosas If-Then, del modelo Sugeno:

$$si\ x\ es\ A_1\ Y\ y\ es\ B_1\ entonces\ f_1 = p_1x + q_1y + r_1$$
$$si\ x\ es\ A_2\ Y\ y\ es\ B_2\ entonces\ f_2 = p_2x + q_2y + r_2$$

Se asumen los consecuentes de las reglas con tres parámetros lineales. El razonamiento borroso tipo Sugeno es ilustrado en la Figura 57(a), y su equivalencia en la arquitectura **Netfuz 1.0** es exhibido en la Figura 57(b).



(a)

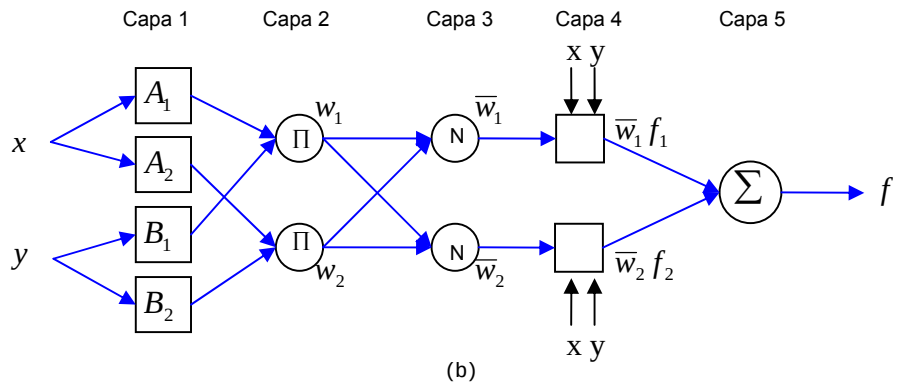


Fig. 57 (a) Razonamiento Borroso Tipo Sugeno; (b) equivalente con Netfuz 1.0

A continuación se definen las funciones y procedimientos a realiza según cada una de las capas que componen la arquitectura.

Capa 1: cada nodo i de esta capa es un nodo cuadrado con una salida de la forma:

$$O_i^1 = \mu_{A_i}(x) \tag{123}$$

Donde x es la entrada al nodo i, y A_i es la etiqueta lingüística (baja, media, etc.) asociada con la salida del nodo. En otras palabras O_i^1 es la FM de A_i. Usualmente, se escoge para $\mu_{A_i}(x)$ una función Bell-shaped, con máximo igual a 1 y mínimo igual a 0, tal que la función bell quede generalizada de la siguiente forma:

$$\mu_{A_i}(x) = \frac{1}{1+\left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}} \tag{124}$$

Donde $\{a_i, b_i, c_i\}$ es el conjunto de parámetros. Como los valores de estos parámetros cambian, las funciones Bell-shaped por consiguiente varían, de esta manera se exhiben varias formas de funciones de membresía para las etiquetas lingüísticas A_i. Los parámetros de esta capa se denominan parámetros de la premisa y son no-lineales.

Capa 2: Los nodos de esta capa son nodos circulares y se denotan por el símbolo $\prod$, en ellos se multiplican las señales de entrada y la salida del nodo es un producto, como el mostrado en la siguiente ecuación:

$$w_i = \mu_{A_i}(c) \times \mu_{B_i}(y), i = 1,2 \tag{125}$$

La salida de los nodos de la capa 2 representa la fuerza de disparo de la regla o la implicación.

Capa 3: Los nodos de esta capa son nodos circulares y se denotan por el símbolo N, en el se calcula la suma de todas las fuerzas de disparo de las reglas de la capa anterior:

$$\overline{w}_i = \frac{w_i}{w_1 + w_2} \qquad i = 1,2 \tag{126}$$

Por conveniencia, en las salidas de esta capa son normalizadas las fuerzas de disparo de las reglas.

Capa 4: Cada nodo i de esta capa es un nodo cuadrado con una salida de nodo descrita como:

$$O_i^4 = \overline{w}_i f_i = \overline{w}_i (q_i x + p_i y + r_i) \tag{127}$$

Donde $\overline{w}_i$ es la salida de la capa 3, y $\{p_i, q_i, r_i\}$ es el conjunto de parámetros. Los parámetros de esta capa se denominan parámetros del consecuente y son lineales.

Capa 5: El único nodo de esta capa es un nodo circular denotado por el símbolo $\sum$ aquí se computa la salida total, como la suma de todas las señales de entrada:

$$O_i^5 = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{128}$$

De esta manera se puede construir una RNA adaptativa que equivale en funcionamiento a un SIB tipo Sugeno, las extensiones para los SIB tipo mandami y Tsukamoto son viables, pero su complejidad es mayor.

8.2 Algoritmo de Aprendizaje Híbrido Netfuz 1.0. De la arquitectura propuesta en la figura 88(a), se presentan los valores de los parámetros de la premisa como parámetros de las FM, la salida total puede ser expresada como una combinación lineal de los parámetros del consecuente. Para ser más precisos, la salida f de la Figura 88 puede ser reescrita como:

$$\begin{aligned} f &= \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2 \\ &= \overline{w}_1 f_1 + \overline{w}_2 f_2 \end{aligned} \tag{129}$$

$$= (\overline{w}_1 x) p_1 + (\overline{w}_1 y) q_1 + (\overline{w}_1) r_1 + (\overline{w}_2 x) p_2 + (\overline{w}_2 y) q_2 + (\overline{w}_2) r_2$$

Que es lineal para los parámetros del consecuente $(p_1, q_1, r_1, p_2, q_2, r_2)$, como resultado se obtiene que:

$S =$ Conjunto total de parámetros

$S_1 =$ Conjunto de parámetros de la premisa

$S_2 =$ Conjunto de parámetros del consecuente

De la ecuación (118); $H(.)$ es una función identidad y $F(.,.)$ es una función propia de los SIB. Por lo tanto en algoritmo de aprendizaje hibrido desarrollado con antelación puede ser aplicado directamente. En el paso hacia delante, después que el vector de entrada es presentado, se calcula la salida de los nodos pertenecientes a La RNA adaptativa, capa por capa, `hasta que las correspondientes filas de las matrices A y X de las ecuación (119) sean obtenidas.

Este proceso se repite para todos los pares de datos de entrenamiento de forma que se completen las matrices A y X; luego los parámetros de consecuente S_2 son identificados, ya sea por el método de la formula de la pseudos inversa, ecuación (120), o el método de mínimos cuadrados recursivo, ecuación (121). Luego que los parámetros de S_2 son identificados, se computa el error obtenido por cada par de datos de entrenamiento. En el paso hacia atrás, las señales de error se propagan desde la salida final hacia la entrada inicial, el vector gradiente es acumulado para cada dato de entrenamiento de entrada. Al final del paso hacia atrás, luego de pasar por todos las datos de entrenamiento, los parámetros en S_1 son actualizados por el gradiente descendente, según ecuación (114). La Tabla 6 resume las actividades por cada paso:

	Paso adelante	**Paso atrás**
Parámetros de la premisa	Fijos	Gradiente descendente
Parámetros consecuente	LSE	Fijos
Señales	Salida de los nodos	Rata de error

Tabla 6 Pasos del Aprendizaje Hibrido

Los sistemas con Múltiples salidas son un caso especial para tratar, aunque es posible implementar la metodología propuesta en ellos, los resultados no son muy halagadores, esto se hace más evidente al aumentar la complejidad de problema planteado.

A continuación, se muestran algunas limitaciones de **Netfuz 1.0**:

- Apto sólo para sistemas Sugeno de orden 0 (el consecuente es un número real) ó 1 (el consecuente es un polinomio de primer grado).

- Tener una salida única, cuyo desemborronado será obtenido por el método de la media de pesos, todos los consecuentes de las reglas deben ser del mismo tipo ya sea de tipo lineal o de tipo constante.
- El número de reglas del SIB es igual al producto de las FM de las entradas del sistema.
- Todas las reglas deben tener un peso igual a 1.
- El número de reglas del SIB no debe ser mayor al número de datos de entrenamiento presentado.

CAPITULO 9

RESULTADOS DE LA INVESTIGACIÓN

Para corroborar la potencialidad de la metodología propuesta, se presenta a continuación dos ejemplos de aplicación a los cuales se les hará un análisis para determinar que grado de exactitud presenta la herramienta construida.

9.1 Ejemplo de Aplicación. Para probar la herramienta se tomaron 25 datos experimentales de entrada/salida que validan la operación $z = \sin(x) * \cos(x)$ definida para el intervalo $\left[-\pi/2, \pi/2\right]$, y se muestran en La Tabla 7.

iteración	X	Y	Z
1	-1.57079	-1.57079	0
2	-1.57079	-0.94247	-0.587785
3	-1.57079	-0.314159	-0.951056
4	-1.57079	0.314159	-0.951056
5	-1.57079	0.94247	-0.587785
6	-0.94247	-1.57079	0
7	-0.94247	-0.94247	-0.475528
8	-0.94247	-0.314159	-0.769420
9	-0.94247	0.314159	-0.769420
10	-0.94247	0.94247	-0.475528
11	-0.314159	-1.57079	0
12	-0.314159	-0.94247	-0.181635
13	-0.314159	-0.314159	-0.293892
14	-0.314159	0.314159	-0.293892
15	-0.314159	0.94247	-0.181635
16	0.314159	-1.57079	0
17	0.314159	-0.94247	0.181633
18	0.314159	-0.314159	0.293892
19	0.314159	0.314159	0.293892
20	0.314159	0.94247	0.181635
21	0.94247	-1.57079	0

22	0.94247	-0.94247	0.475528
23	0.94247	-0.314159	0.769420
24	0.94247	0.314159	0.769420
25	0.94247	0.94247	0.475528

Tabla 7 Datos Experimentales

El ejercicio de aplicación contiene dos entradas y una salida, las entradas poseen tres particiones borrosas: {baja, media, alta}. Al utilizar la metodología propuesta se obtienen los parámetros de las FM para el universo de discurso (o rango), además de las reglas borrosas con sus respectivos consecuentes.

9.1.1 FM obtenidas. La FM para el intervalo {bajo} de la variable X es:

$$e^{-\frac{(x+1.561)^2}{0.5422}}$$

La FM para el intervalo {medio} de la variable X es:

$$e^{-\frac{(x+0.312)^2}{0.521}}$$

La FM para el intervalo {alto} de la variable X es:

$$e^{-\frac{(x-0.932)^2}{0.5475}}$$

La FM para el intervalo {bajo} de la variable Y es:

$$e^{-\frac{(x+1.575)^2}{0..5355}}$$

La FM para el intervalo {medio} de la variable Y es:

$$e^{-\frac{(x+0.316)^2}{0.547}}$$

La FM para el intervalo {alto} de la variable Y es:

$$e^{-\frac{(x-0.949)^2}{0.53}}$$

9.1.2 Reglas Borrosas. Las reglas borrosas obtenidas para el ejercicio surgen como la combinatoria de las FM de las entradas**,** y se presentan a continuación:

$$1\, si\ X\ es\, bajo\, y\, Y\, es\, bajo\, entonces Z = 0.06321$$
$$2\, si\ X\ es\, medio\, y\, Y\, es\, bajo\, entonces\ Z = -1.201$$

$3\ si\ X\ es\ alto\ y\ Y\ es\ bajo\ entonces\ Z = -0.6405$
$4\ si\ X\ es\ bajo\ y\ Y\ es\ medio\ entonces\ Z = 0.02196$
$5\ si\ X\ es\ medio\ y\ Y\ es\ medio\ entonces\ Z = -0.4172$
$6\ si\ X\ es\ alto\ y\ Y\ es\ medio\ entonces\ Z = -0.2225$
$7\ si\ X\ es\ bajo\ y\ Y\ es\ alto\ entonces\ Z = -0.0524$
$8\ si\ X\ es\ medio\ y\ Y\ es\ alto\ entonces\ Z = 0.9952$
$9\ si\ X\ es\ alto\ y\ Y\ es\ alto\ entonces\ Z = 0.5309$

Para el ejemplo planteado, los resultados obtenidos por **Netfuz 1.0** al utilizar la regla de aprendizaje hibrido, generaron un error mínimo en un número de iteraciones bajo, superando los resultados obtenidos con el aprendizaje por el método del gradiente descendente, los resultados se muestran en la tabla 8.

MÉTODO	ERROR	ÉPOCAS
Hibrido	0.052838	3
Gradiente Des.	0.45035	10
Gradiente Des.	0.004056	100

Tabla 8 resultados obtenidos

A partir de los resultados se evidencia que al usar la regla de aprendizaje hibrido, se reduce, ostensiblemente, el error final en un número de épocas muy pequeño, resultados que contrastan con el método del gradiente que alcanzó un error aceptable pero, en un número de iteraciones alto.

La metodología de **Netfuz 1.0** presenta complicaciones cuando la complejidad del problema aumenta, como el caso de cuatro entradas o más, luego se recomienda para ejercicios con dos, y la herramienta esta diseñada para situaciones de dos entradas y una salida.

Los campos de aplicación de la herramienta **Netfuz 1.0**, van desde la identificación de sistemas no lineales, la predicción de parámetros, el pronóstico y los sistemas de control, ya sean en línea o no.

CAPITULO 10

DISEÑO Y SISTEMA NETFUZ 1.0

10.1 Diseño. El diseño del software se realizó utilizando el lenguaje Unificado de Modelado (Unified Modeling Language, UML), donde UML es un lenguaje o forma estandarizada de desplegar y escribir la estructura de un software por medio de conceptos orientados a objetos. Puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra gran cantidad de software. A continuación se presentan los diferentes elementos de diseño del proyecto Netfuz 1.0.

10.1.1 Diagrama De Clases Del Proyecto Netfuz 1.0. Un diagrama es una presentación gráfica de un conjunto de elementos, que la mayoría de las veces se dibuja como un grafo conexo de nodos (elementos) y arcos (relaciones). Las Clases son la base para la construcción de un sistema orientado a objetos. Para la realización del proyecto Netfuz 1.0 se elaboraron cuatro prototipos en los cuales se fueron agregando progresivamente las clases correspondientes para añadir las funcionalidades o componentes requeridos.

10.1.1.1 Prototipo No 1. El diagrama de clases del prototipo No 1 se muestra a continuación (Figura 58):

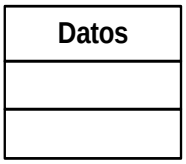


Fig. 58 Diagrama de Clases del Prototipo No 1

En este prototipo se implementaron los métodos de:

- Entrada de datos.
- Manipulación de datos permitiendo Agregar, Eliminar y Modificar conjuntos y reglas.
- Manipular subconjuntos de datos, con las funcionalidades dadas por el sistema.

El sistema Netfuz 1.0 se desarrolló con una estructura de bases de datos PARADOX, como un manejo de archivos Indexados, en la figura 59 se presenta la estructura de las tablas que se diseñaron:

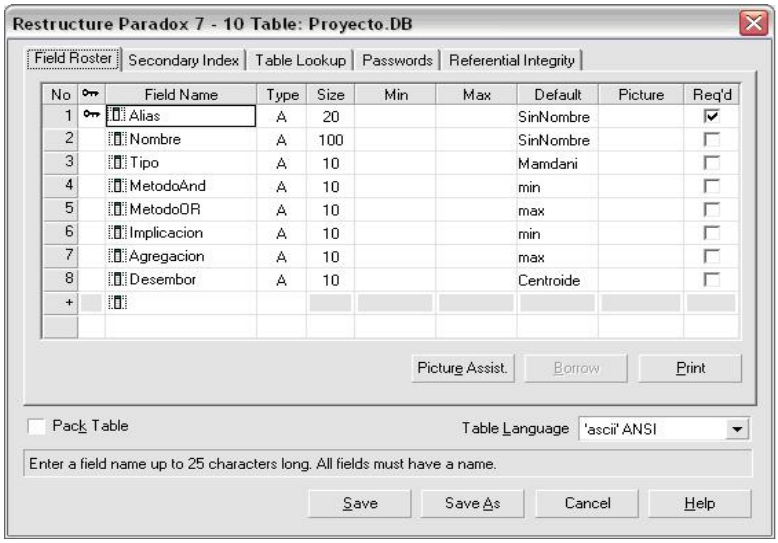


Fig. 59 Tablas de la Base de Datos

10.1.1.2 Prototipo No 2.

El diagrama de clases de este prototipo se aprecia en la figura 60:

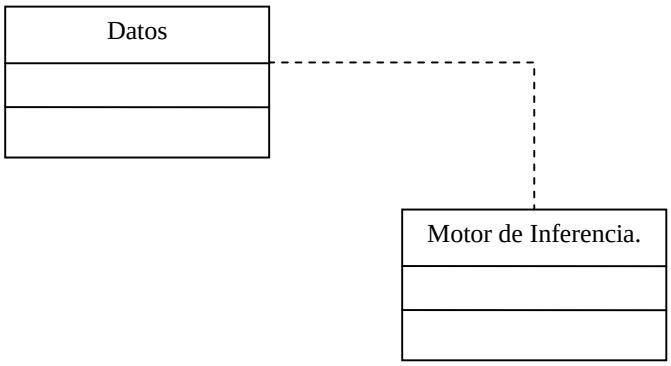


Fig. 60 Diagrama de Clases del Prototipo No 2

En este prototipo se agrego la clase Motor de inferencia la cual permite la fácil creación de la base de reglas borrosas con todas las características que se tienen en cuenta para el normal funcionamiento de un SIB.

10.1.1.3 Prototipo No 3. El diagrama de clases de este prototipo se aprecia en la figura 61:

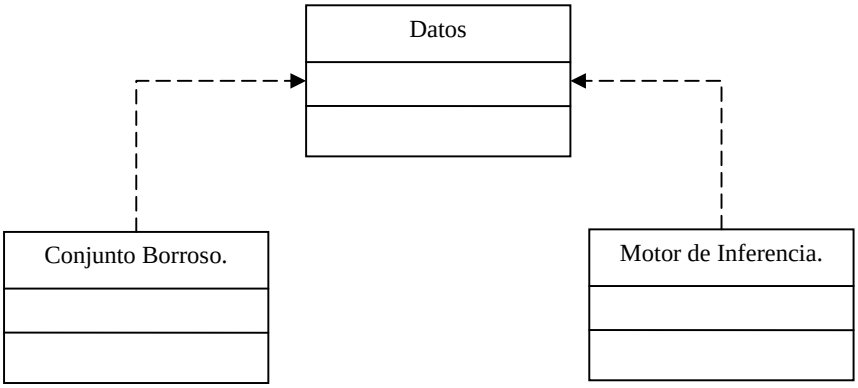


Fig. 61 Diagrama de Clases del Prototipo No 3

En este prototipo se ha agregado la clase Conjunto Borroso en la que se maneja toda la información de los SIB, como lo son CB y las FM.

En este prototipo también se implementaron los siguientes métodos:

- Entrada de información especifica.
- Emborronado, que nos permite transformar la información cuantitativa en información cualitativa, por medio de variables y modificadores lingüísticos.
- Buscar reglas activas.
- Operar las reglas activas.
- Operadores de implicación.
- Agregación de reglas.

10.1.1.4 Prototipo No 4. El diagrama de clases de este prototipo se aprecia en la figura 62:

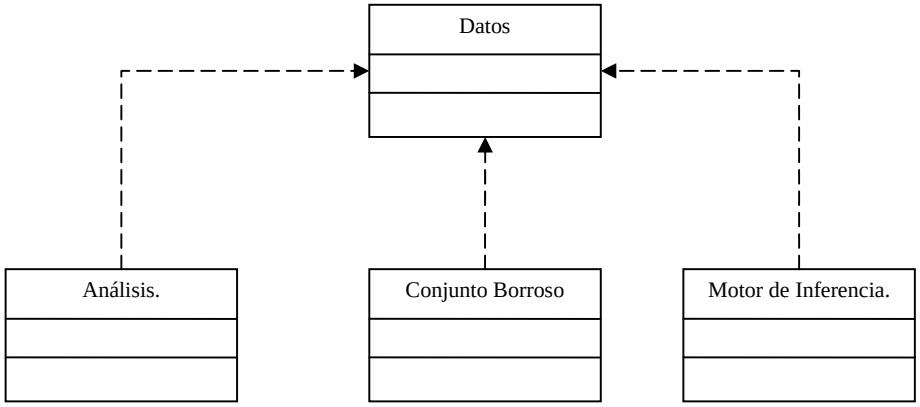


Fig. 62 Diagrama de Clases del Prototipo No 4

En este prototipo se agrego la clase Análisis que permite la toma de decisiones del usuario usando como referencia los resultados obtenidos por la herramienta, ya sea en forma gráfica o analítica.

10.1.1.5 Prototipo No 5. El diagrama de clases de este prototipo se aprecia en la figura 63:

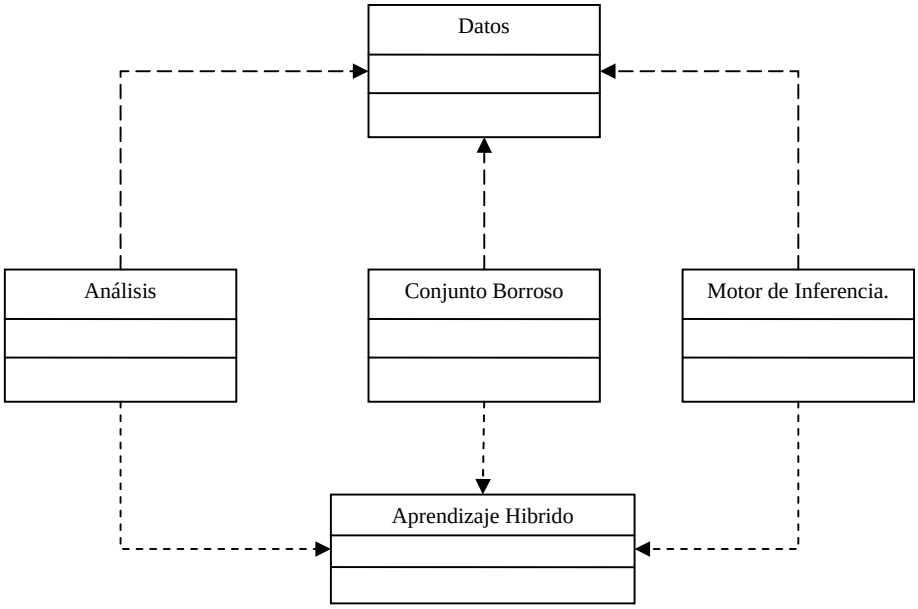


Fig. 63 Diagrama de Clases del Prototipo No 5

En este prototipo se agrego la clase Aprendizaje Híbrido que permite generar, automáticamente, las reglas borrosas y los parámetros de las funciones de membresía, basado en la metodología propuesta.

10.1.1.2 Diagrama De Clases Final. El diagrama de clases final se aprecia en la figura 64:

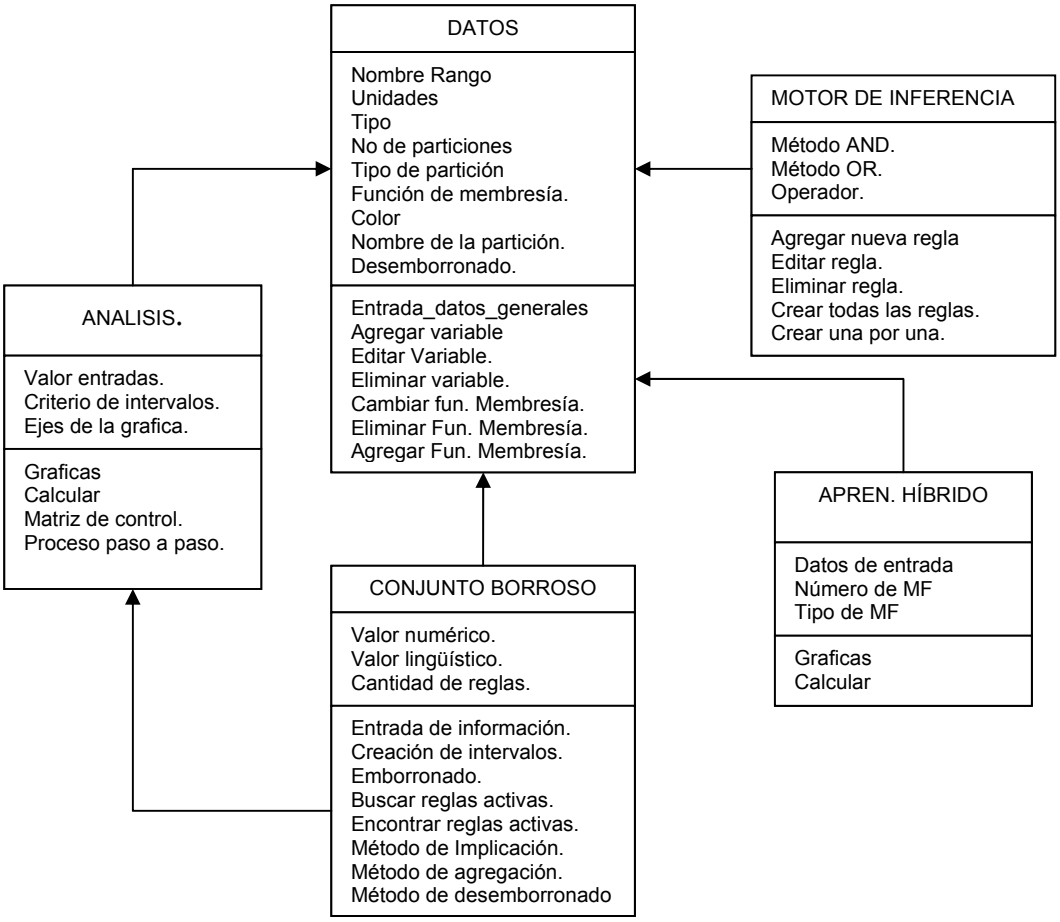


Fig. 64 Diagrama de Clases Final

10.2 SISTEMA

10.2.1 Arquitectura De Netfuz 1.0. El sistema se construyó en el ambiente de desarrollo DELPHI versión 7.0, que pertenece a la casa Borland y se presenta como una alternativa para el desarrollo de aplicaciones; Además, se destaca la velocidad, robustez y eficiencia de los compiladores que desarrolla.

Esto con el fin de desarrollar una herramienta que esté en capacidad de enfrentarse a problemas de gran amplitud como son los SIB, debido a que pueden existir sistemas con múltiples entradas y salidas.

10.2.2 Características Del Sistema. Netfuz 1.0 es una herramienta para el diseño y manejo de controladores inteligentes basados en LB, debido a la gran aceptación que estos, han tenido últimamente, se implementó un sistema capaz de proveer apoyo al experto de los procesos.

Después de tener las variables de entrada y salida definidas se continúa el proceso normal para la construcción de un SIB, como es la base de reglas, los métodos de desemborronado, y otras características que la herramienta brinda al usuario como:

- Generación de los conjuntos borrosos
- Elección de la función de membresía
- Construcción de las reglas borrosas
- Método de desemborronado
- Matriz de control
- Impresión de los datos y el reporte del caso en estudio
- Adquisición de datos
- Ajuste, automático, de los parámetros de las MF
- Generación de las reglas borrosas para un SIB TSK

10.2.3 Características De Los Componentes. Cada uno de los componentes tiene funcionalidades que permiten ayudar a todo el proceso global para la creación de los SIB.

10.2.3.1Componente de carga de datos. Este componente esta formado por la clase Datos permitiendo la carga a partir de archivos planos ya sean propios del sistema o en el formato de Netfuz 1.0, tomando esa información recibida para su manipulación por medio de las leyes de La LB, esto con el fin de generar las soluciones requeridas por los usuarios, después de los procesos propios de un SIB como son:

- Emborronado.
- Motor de inferencia.
- Desemborronado.

10.2.3.2 Componente de creación de los conjuntos borrosos. Este componente brinda facilidades para la construcción de los conjuntos borrosos de entrada y salida que son la base para la creación del SIB, permitiendo la manipulación de las FM, los implicadores, la construcción de la base de reglas y otras características propias del sistema.

10.2.3.3 Componente de visualización de resultados. Para este componente se utilizó un objeto grafico que permite la visualización, para una fácil comprensión y asociación de los resultados, realizando todo esto por métodos implementados durante el desarrollo del proyecto.

10.2.3.4 Motor e Interfaz a Usuarios. En el motor del sistema se centralizan o agrupan instancias de los diferentes componentes base del sistema, permitiendo la realización del proceso en forma global. La interfaz a usuarios comprende todos los componentes y formularios gráficos con los cuales se interactúa, construida de forma amigable y de fácil manejo.

10.2.3.5 Componente Para el Aprendizaje Híbrido. Este componente permite generar las reglas borrosas y los parámetros de las FM, de un SIB tipo TSK, partiendo de un conjunto de datos entrada - salida.

10.2.4 Parámetros Específicos. Para la construcción de los SIB se requiere una serie de parámetros que se explican a continuación:

- Mínimo numero de variables: la herramienta exige como mínimo la inclusión de dos variables de entrada y una variable de salida al proceso ya que es la única restricción para operar la herramienta.
- Máximo numero de variables: La herramienta da un rango máximo de variables al controlador que es de quince (15).
- Base de reglas: la base de reglas tendrá la opción para su construcción de generar las reglas una por una, según sea la elección del experto.
- Funciones de membresía: Las funciones de membresía que se implementaron fueron: Triangular, trapezoidal, gaussiana, bell y sigmoidal. Que son las más utilizadas por los expertos en LB, debido a su sencillez y buena transformación de los datos numéricos.
- Método de desemborronado: Los métodos de desemborronado implementados son los siguientes, centro de área, Media de máximos, Primer máximo, y ultimo máximo. Son los más eficientes computacionalmente hablando, y además los que tienen más exactitud en los resultados obtenidos, se da la posibilidad de implementar otros métodos.
- Para la metodología implementada se dala posibilidad de dos FM, sigmoidal y trapezoidal.

10.2.5 Requerimientos de Hardware y Software. Para el desarrollo del sistema Netfuz 1.0 se utilizó un equipo con las siguientes características:

- Procesador Pentium IV de 2.8 Mhz.
- 512 MB en RAM.

Para la ejecución del sistema Netfuz 1.0 se requieren como mínimo 256 MB de memoria RAM y, tener procesador Pentium III o superior. Esta herramienta funciona en sistemas operativos Windows series 9X, NT, 2000, XP.

CAPITULO 11

CONCLUSIONES Y TRABAJOS FUTUROS

11.1 Conclusiones

- La simbiosis entre la LB y las RNA, deja al descubierto un gran potencial que se debe explotar en todos los campos de la Ciencia y la Ingeniería, debido a los resultados obtenidos por Netfuz 1.0 en experimentos llevados a cabo, además de las posibles aplicaciones en las cuales es viable implementar esta metodología.

- Los sistemas neuro-borrosos proveen una estrategia a tomar en cuenta, para abordar situaciones imprecisas y de alta complejidad, generando resultados halagadores teniendo en cuenta que, métodos tradicionales no aplican para este tipo de situaciones.

- La metodología propuesta facilita la implementación de los SIB tipo Sugeno, y acrecienta el interés por parte de los investigadores para generar soluciones efectivas que conlleven el uso de Tecnologías Adaptativas, erigiéndose como pilar fundamental en la aplicabilidad de estas técnicas en pro de forjar soluciones en campos inexplorados.

- Este trabajo plasma los conocimientos adquiridos durante el transcurso de la maestría, hecho que se refleja en los objetivos alcanzados, que dan testimonio del arduo trabajo levado a cabo, producto de las exigencias por parte de profesores y directores, que orientaron mi trabajo en pos de aportar una pizca de ingenio e innovación al gigantesco mundo de la investigación.

11.2 Recomendaciones y Trabajos Futuros

- Los resultados obtenidos por Netfuz 1.0, debido al aprendizaje híbrido utilizado colmaron las expectativas, inicialmente contempladas, aunque es posible la implementación de otro tipo de aprendizaje, que deriva la investigación de nuevas arquitecturas neuronales.

- Ampliar la aplicabilidad de Netfuz 1.0, no sólo a los Sistemas de Inferencia Borrosos tipo TSK, sino a los tipo Mandami, que presentan relevancia en las investigaciones y son utilizados en gran número de aplicaciones; hecho que permite la evolución de la herramienta computacional.

- Se recomienda probar nuevas combinaciones con metodologías como: Los Algoritmo Genéticos, El Recocido Simulado y Los Agentes Inteligentes, para incrementar las eficiencia y eficacia de los Sistemas Híbridos, hecho que potenciaría este tipo de simbiosis lo que deriva en un aumento en el número de aplicaciones que integrar Tecnologías Adaptativas.

- Se recomienda probar la herramienta Netfuz 1.0 en situaciones complejas, que cumplan con los requerimientos y limitaciones de esta, para incrementar la aceptación de estas metodologías y de paso cambiar los conceptos errados que se tiene con las Tecnologías Adaptativas.

ANEXO A

MANUAL DEL USUARIO PARA MANEJO DEL SOFTWARE NETFUZ 1.0

A.1 Generalidades del Software. Netfuz 1.0 es una herramienta para la generación automática de reglas borrosas y el ajuste de los parámetros de las FM. Basado en una metodología que integra la Lógica Borrosa y las Redes Neuronales Artificiales.

El sistema se construyó en el ambiente de desarrollo DELPHI versión 7.0 y el almacenamiento de datos de la herramienta a través de la base de datos PARADOX.

A.2 Instalación del Software. Para la correcta instalación del software se debe tener en cuenta los siguientes pasos:

a. Inserte el CD de instalación que contiene el software Netfuz 1.0 en la unidad de CD.
b. Bajo el explorador de Windows ubicar la unidad de CD, donde se pueden visualizar los diferentes archivos que componen el software.
c. Abrir el archivo ejecutable SetupNetfuz1.0.exe, el cual inicia la instalación.
d. El ejecutable despliega una pantalla indicando los pasos a seguir, hasta la correcta instalación.
e. El programa crea una carpeta denominada Netfuz 1, ubicada en la siguiente dirección: c:\archivos de programa\netfuz1 donde quedará instalado.
f. Así mismo el instalador crea un acceso directo al programa en el menú inicio del escritorio de Windows.
g. Debido a que los rangos de números con los que trabaja Netfuz 1.0 tienen al punto (.) como configuración del símbolo decimal y separador de miles, se debe revisar la *configuración regional* del equipo de manera que este así.
h. En los computadores que no tengan instalados programas de la casa Borland a la que pertenece Delphi 7.0, es necesario instalar la librería ***BdeInst.dll***, esta librería se adjunta dentro del CD de instalación. El registro de esta librería se realiza de esta forma: crear una carpeta en la unidad C:\ llamada **netfuz1** y copiar allí esta librería. La dirección quedaría así: C:\netfuz1\BdeInst.dll.

Una vez efectuados los pasos anteriores se puede acceder al software ya sea a través del menú inicio o por al acceso directo creado.

A.3 Uso de la Herramienta y Presentación de los Menús. Al abrir el programa aparece el entorno principal de trabajo. Visualizamos de una forma gráfica todo el entorno de Netfuz 1.0, que es de fácil lectura y permite empezar el desarrollo del trabajo.

En este momento puedo Crear un proyecto o seleccionar un proyecto ya existente.

El software Netfuz1.0 presenta el siguiente entorno de trabajo (figura 65):

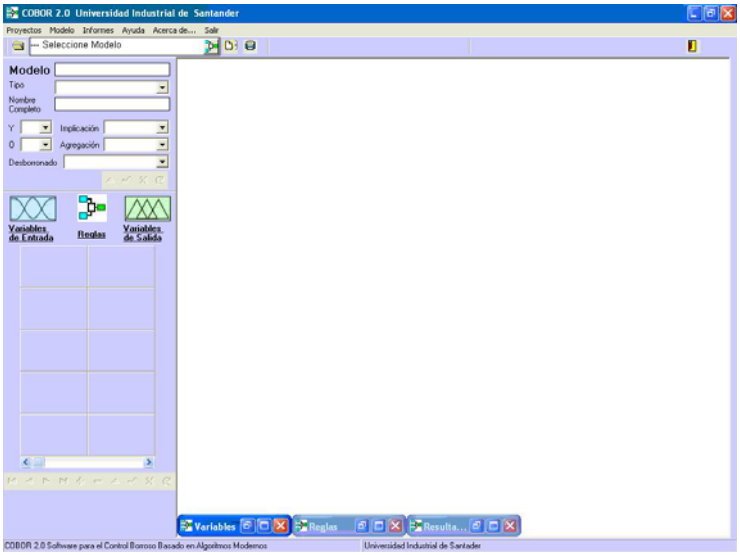


Fig. 65 Entorno de Trabajo Netfuz 1.0

El menú Principal presenta las siguientes opciones:

1. Proyectos
2. Modelo
3. Informes
4. Ayuda
5. Acerca de
6. Salir

A.3.1Abrir. Es la opción que permite seleccionar la carpeta en donde se almacena un proyecto creado (figura 66).

También puede activarse la opción abrir, a través del correspondiente botón de acceso directo ubicado en la pantalla inicial.

Nota: Si existe un proyecto abierto y se va a abrir otro, se debe primero salir del que esta abierto, debido a que el software no acepta dos proyectos, simultáneamente, y luego si activar la función abrir proyecto.

Fig. 66 Ventana Buscar Carpeta

A.3.2 Salvar. Es la opción que me permite guardar todo el proyecto que he creado o modificado, lo ubica en la pantalla principal y hay que indicarle la carpeta en donde se desee salvar la información.

A.3.3 Salir. Es la opción que termina la ejecución del programa.

A.4 Modelo. Es la opción del menú principal que nos permite visualizar las ventanas que se trabajan en el software Netfuz 1.0 Al entrar a ésta opción del menú presenta tres alternativas: Variables, Reglas y Resultados. En la figura 67 se exhibe esta descripción.

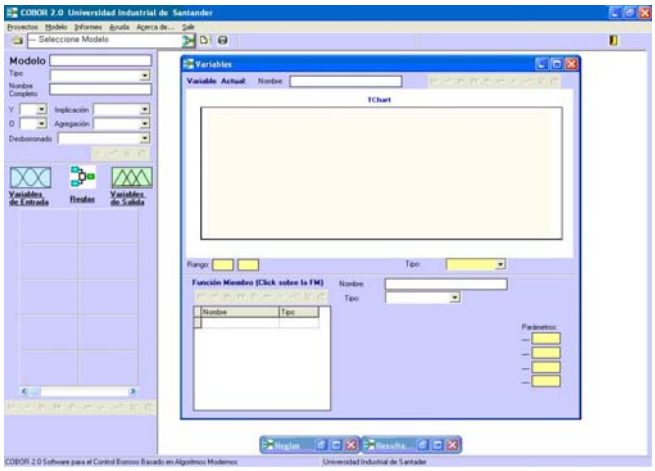


Fig. 67 Ventana Variables del Proyecto

Se visualiza la ventana de las Variables, sean tipo salida o tipo entrada y sus respectivas especificaciones.

También es posible visualizar la ventana de las Reglas borrosas, importante en el SIB, allí es posible implementar la matriz, con las opciones que el experto desee utilizar (figura 68).

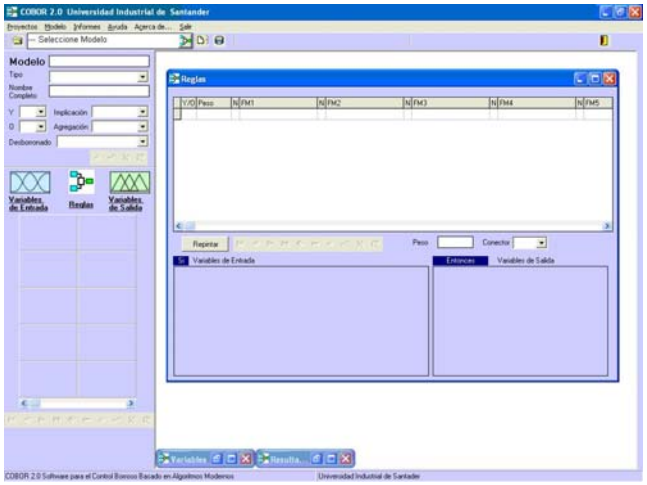


Fig. 68 Ventana Reglas Borrosas del proyecto

Visualizamos la ventana en donde se generan los resultados de forma gráfica.

En ésta ventana se generan los procesos que nos permiten observar a través de gráficas todo el proyecto que se ha generando (figura 69).

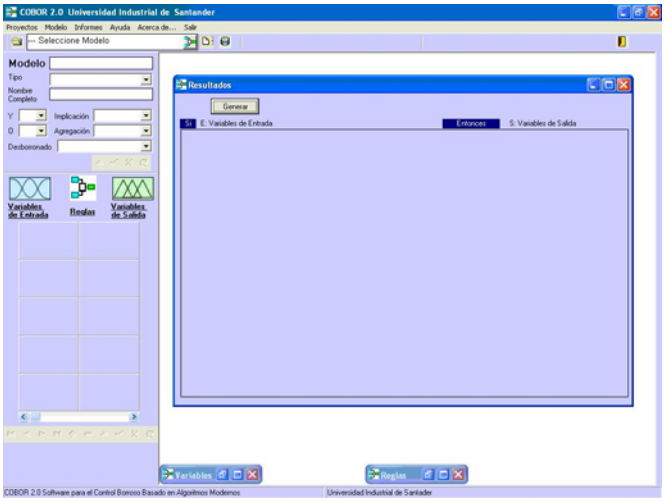


Fig. 69. Ventana Resultados del Proyecto

A.5 Informes. Es la opción del menú que nos presenta el informe detallado de las variables y reglas que conforman el proyecto (figura 70).

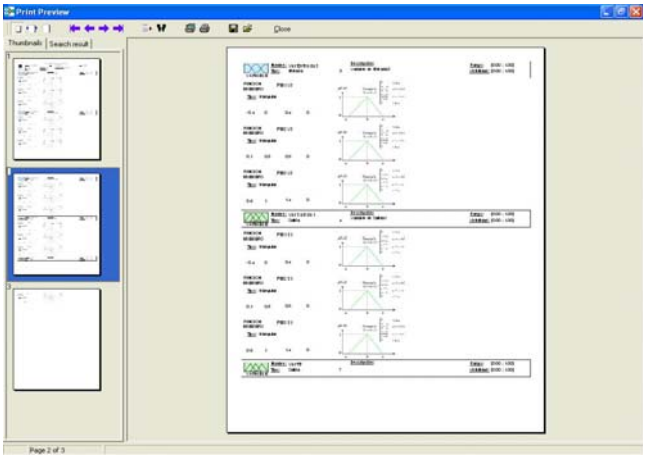


Fig. 70 Ventanas Informes del Proyecto

Se visualiza el informe detallado del proyecto que se ha venido desarrollando. Dentro de la visualización del informe, se activa un menú grafico que contiene entre otras las siguientes opciones: pasar a siguiente pagina, imprimir, guardar, etc.

A.6 Ayuda. Esta opción permite visualizar la ayuda al usuario del software de manera que se pueda consultar aspectos relacionados de su funcionamiento.

A.7 Acerca de. Es la opción del menú que presenta los datos del autor del software Netfuz 1.0

A.8 Salir. Es la opción que termina la ejecución del software

BIBLIOGRAFÍA

[1] Azvine B., Azarmi N., and Nauch D., Intelligent Systems and Soft Computing, Springer, 2000.

[2] Zadeh L., Fuzzy logic and approximate reasoning, *Syntheses*, 30(1), pp. 407-428, 1975.

[3] Zadeh L., Fuzzy and sets, inf. control, Vol. 8, pp 338-353, 1965.

[4] McCulloch W. and Pitts W., A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, 5, pp. 115-133, 1943.

[5] Jang J., Sun C., Mizutani E., Neuro Fuzzy and Soft Computing, A Computational Approach to Learning and Machine Intelligence, Prentice Hall, New York, 1999.

[6] Corredor, Martha Vitalia, Principios de Inteligencia Artificial y Sistemas Expertos, Ediciones UIS. 1982.

[7] Zadeh L., From circuit theory to systems theory, IRE Proc., 50, pp. 856-865, 1962.

[8] Haber R., Introducción al control borroso, En: 1er seminario internacional sobre control inteligente de procesos, 1995.

[9] Dubois D., and Prade H., Fuzzy sets and Systems: Theory and applications, academic press, New Cork, 1980.

[10] Terano T., Asai K., M. Sugeno, Theory and its applications, 1991.

[11] Ross T., Fuzzy Logic with Engineering Applications, McGraw –Hill, 1997.

[12] Mandami E., Applications of fuzzy logic to approximate reasoning using linguistic Systems, *IEEE Trans. On Systems, Man, and Cybernetics,* 26(12), pp. 1182-1191, 1977.

[13] Takagi T., and Sugeno M., Fuzzy Identification of System and its Application to Modelling and Control. IEEE Trans on SMC, 15(1), 116-132, 1985.

[14] Kosko B., “Neural Networks and Fuzzy Systems.”, Prentice Hall, Englewood, New Jersey, 1992.

[15] Hilera J., y Martínez V., Redes Neuronales Artificiales, Fundamentos, modelos y aplicaciones, Alfa omega Ra-Ma, 2000.

[16] McCulloch W., Pitts W., A logical calculus of the ideas imminent in nervous activity, *Bull. Math.Biophys*. 5:115-133. pp. 96-104. Referenciado en Hastie, T; Tibshirani, R., Friedman J. (2001).

[17] Ramón y Cajál, S., Histologie du système nerveux de l'homme et des vertèbres. París: Maloine; Edition Francaise Revue: Tome I, 1952; Tome II, 1955; Madrid: Consejo Superior de Investigaciones Científicas, 1911.

[18] Ogata K., Ingeniería de control moderna, Prentice Hall, 1998.

[19] Jamshidi M., On Software and Hardware Applications of Fuzzy Logic, in R. R. Yager and L. A. Zadeh Ed., *Fuzzy Sets, Neural Networks and Soft Computing*. New York: Van Nostrand Reinhold, 1994.

[20] Cembrano G., Wells G., Neural Networks for Control, pp. 383 - 402, 1992.

[21] Camacho E., Aracil M., Neural Network Based Adaptive Control, IFAC AIRTC'94, Valencia, 15-26, 1994.

[22] Tong, R., Synthesis of fuzzy models for industrial processes, *Int. Gen. Syst.* 4, 143-162, 1978.

[23] Procyk T., Mandani E., A linguistic self-organizing process controller, Automatica, vol. 15, pp. 15-30, 1979.

[24] Lee C., Berenji H., An intelligent controller based on approximate reasoning and reinforcement learning, *Proc. IEEE Intell. Mach*, 200-205, 1989.

[25] Lin C., Lee, C., Neural-network-based fuzzy logic control and decision system, *IEEE Trans. Computer*, vol. 40, no. 12, pp. 1320-1336, 1991.

[26] Horikawa S., Furuhashi T., Uchikawa Y., Tagawa, T. A study on fuzzy modeling using fuzzy neural networks, *Proc. Int. Fuzzy Eng. Symp.*, 1991.

[27] Wang L., Mendel J., Fuzzy basis functions, universal approximation, and orthogonal least-squares learning, *IEEE Trans. Neural Networks*, vol. 3, pp. 807-814, 1992.

[28] Simpson P., Fuzzy min-max neural networks. Part 1: Classification, *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 776-786, Sept. 1992.

[29] Ishibuchi H., Fujioka R., Tanaka H., Neural Networks that Learn from Fuzzy If-Then Rules, *IEEE Trans. Fuzzy Systems*, vol. 1, pp. 85-97, 1993.

[30] Kuo Y., Chen, J., A Fuzzy Neural Network Model and Its Hardware Implementation, *IEEE Trans. Fuzzy Systems*, vol. 1, no. 3, Aug. 1993.

[31] Lin C., Lee C., Supervised and Unsupervised Learning with Fuzzy Similarity for Neural Network-Based Fuzzy Logic Control Systems, in R. R. Yager and L. A. Zadeh Ed., *Fuzzy Sets, Neural Networks and Soft Computing*. New York: Van Nostrand Reinhold, 1994.

[32] Johansen T., Murray-Smith R., The operating regime approach to nonlinear modelling and control, en *Multiple Model Approaches to Modelling and Control*. Ed. R. Murray-Smith and T.A. Johansen, London: Taylor & Francis, 1997.

[33] Jang J., Fuzzy modeling using generalized neural networks and Kalman filter algorithm. In Proc.bof the Ninth National Conference on Artificial Intelligence (AAAI-91), pp. 762–767, July 1991.

[34] GOODWIN G. and SIN S., Adaptive filtering prediction and control, Prentice-Hall, Englewood Cliffs, N.J., 1984.

[35] LJUNG L., System identification: theory for the user, Prentice-Hall, Englewood Cliffs, N.J., 1987.

[36] STROBACH P., Linear prediction theory: a mathematical basis for adaptive systems, Springer-Verlag, 1990.

[37] KOSKO B., Neural Nets y Fuzzy System, prentice Hall, 1992.