Introducción a las Técnicas de Computación Inteligente

Aguilar Castro, José L.
Cerrada Lozada, Mariela
Colina Morles, Eliezer
Hidrobo Torres, Francisco
Rivas Echeverría, Francklin
Rodríguez Graterol, Wladimir

Universidad de Los Andes
Mérida, Venezuela

Abril, 2000

Editor:
José L. Aguilar Castro

Diagramación:
Mayerlin Y. Uzcátegui S.

Portada:
XXXXXX

Impresión:
XXXXXX

Depósito de Ley:
LF XXXXXXX

ISBN:
XXXXXX

A mis seis soles:
Kristell, Vanessa, Anyo, Ana, Mamá y Papá
por esa luz que me irradian permanentemente
José L. Aguilar Castro

A Cristian y Antonio... los amo
Mamá y Papá... fuente de apoyo incondicional
Mariela Cerrada Lozada

Índice General

Índice de Figuras

Introducción a la Computación Inteligente

José L. Aguilar Castro
CEMISID
Departamento de Computación
Facultad de Ingeniería
Universidad de Los Andes
Mérida 5101, Venezuela
E-mail: aguilar@ula.ve

Eliezer Colina Morles
Departamento de Sistemas de Control
Facultad de Ingeniería
Universidad de Los Andes
Mérida 5101, Venezuela
E-mail: ecolina@ula.ve

Capítulo 1

Introducción a la Computación Inteligente

El propósito de este capítulo es presentar el concepto y las ideas centrales de la Computación Inteligente, así como diferentes aspectos esenciales de esta área de estudio. En particular, se bosquejan técnicas que normalmente son consideradas como partes de la Computación Inteligente, las cuales serán objeto de una presentación mas exhaustiva en el resto del libro. Por otro lado, en este capítulo se darán definiciones preliminares de los principales términos usados en el libro.

1.1 Computación Inteligente

La necesidad de desarrollar sistemas inteligentes se ha incrementado en los últimos años debido a que el conocimiento se ha convertido en un recurso estratégico para ayudarnos en tareas complejas [1, 2, 7, 10]. Esto también ha impulsado estudios teóricos dirigidos, fundamentalmente, a lograr una mejor comprensión de los mecanismos de procesamiento de información en los humanos. La comunidad científica computacional ha respondido a estas necesidades a través del área computación inteligente y, en particular, a través de los mecanismos de integración de las diferentes técnicas inteligentes para conformar los sistemas híbridos inteligentes [1, 4, 6, 7, 8, 9]. En el primer congreso sobre Computación Inteligente realizado en 1994 con auspicios del IEEE, Bezdek [1] presentó la relación entre los componentes de un sistema inteligente (ver figura 1.1).

En la figura 1.1 se caracteriza la complejidad de los sistemas inteligentes, aumentado dicha complejidad de izquierda a derecha y de abajo a arriba. En esta concepción, la computación inteligente tiene mas que ver con la implementación computacional que con el modelado del paradigma. El nivel superior representa la inteligencia biológica y los dos niveles siguientes la inteligencia de la máquina. Este modelo ha recibido numerosas críticas (en [1] se encuentra un análisis de todas las críticas hechas a dicho modelo).

La Computación Inteligente la definiremos como una metodología de cálculo que tiene habilidades para adaptarse a nuevas situaciones, posee atributos de razonamiento, tales como: generalización, descubrimiento, asociación y abstracción; además, sus respuestas son predicciones o tomas de decisiones [1, 2, 4, 6, 7]. De esta manera, la Computación Inteligente envuelve conceptos, paradigmas y algoritmos adaptativos, los cuales permiten generar apropiadas acciones en ambientes complejos y cambiantes

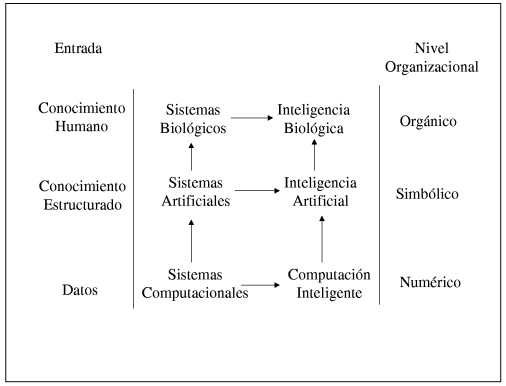

Figura 1.1: Relación entre los componentes de los Sistemas Inteligentes

(comportamiento inteligente). Los modelos computacionales usados tienen analogía biológica, basados en lo que se ha logrado comprender y concebir como inteligencia (cuando se hace referencia a la inteligencia normalmente se piensa en la relación inteligencia-biología, debido a que esta ha sido la que se ha estudiado con mayor amplitud; sin embargo, se podría pensar en inteligencia no biológica).

La computación inteligente se ha desarrollado desde dos ángulos [1, 2, 3, 4, 5, 6]: a partir de la implementación de aplicaciones basadas en sus paradigmas para resolver problemas particulares, o a partir de la definición de modelos de los diferentes mecanismos de procesamiento de información de los humanos. En particular, las investigaciones ligadas en como hibridizar las técnicas inteligentes existentes, con el fin de subsanar las debilidades de cada una de ellas y aumentar el uso potencial de las mismas, han tenido un papel fundamental en el desarrollo de la computación inteligente.

Inspirados en los mecanismos de procesamiento de información de los humanos, se podría hacer una primera clasificación de las técnicas inteligentes usadas en computación inteligente. En este caso, se consideraran solamente aspectos cognitivos y estructurales. Siguiendo este razonamiento, dos escuelas aparecen: una basada en los trabajos de Newell y Simon [1972] [1, 2, 3, 5, 8, 9, 10], conocida como la escuela clásica de inteligencia artificial. La otra, basada en los trabajos de McClelland y otros [1986] [1, 2], sigue un esquema de procesamiento distribuido-paralelo inspirado en el comportamiento del cerebro. En la escuela clásica se afirma que la mente humana es esencialmente una máquina de procesamiento secuencial, donde la información es procesada siguiendo un esquema de producción de reglas. En ella se definen protocolos de análisis para modelar los procesos mentales característicos de los sistemas de procesamiento de los humanos. En la segunda escuela se propone que el proceso

inteligente consiste en un gran número de elementos de procesamiento simples, los cuales operan en paralelo y se comunican entre si (el procesamiento de información es realizado a través de la interacción entre ellos). Estos modelos son conocidos como esquemas conexionistas. Este último enfoque es evolucionario por naturaleza, ya que las interacciones en el sistema están constantemente cambiando. Estas dos escuelas representan los principales espectros a nivel de los mecanismos de procesamiento de información de los humanos. Uno de los espectros representa un procesamiento basado en un comportamiento analítico-interpretativo (sistemas simbólicos), mientras que, el otro representa un comportamiento intuitivo, no interpretativo-automático (redes conexionistas). Mientras el primer espectro funciona en unidades de tiempo de minutos o segundos (nivel macroestructural), el otro en unidades de tiempo de milisegundos (nivel microestructural).

1.2 Conceptos Preliminares

El nuevo diccionario Colegial Webster [1, 11] define *inteligencia* como "la habilidad para aprender o entender, o para tratar nuevas situaciones". En el diccionario REASON es presentada otra definición [1]: "la habilidad para aplicar conocimiento para manipular un ambiente". Otra definición de Inteligencia, quizás más adaptada a este libro, es dada por David Fogel [1, 2]: "es la capacidad de un sistema para adaptar su comportamiento, de tal manera de alcanzar sus objetivos en ambientes variables".

Pero, para comprender que es inteligencia, es necesario entender como se adquiere el conocimiento, como éste es almacenado, como el comportamiento inteligente es generado, como motivos, emociones y prioridades son usadas y desarrolladas, como señales sensoras son transformadas en símbolos, como símbolos son manipulados lógicamente para razonar sobre el pasado y planificar para el futuro, y como el mecanismo de inteligencia produce los fenómenos de ilusión, creencia, esperanzas, sueños y temores. Para entender esas funciones se deben conjugar estudios en muchas áreas, que van desde la neurofisiología, la biología genética hasta las matemáticas.

La Inteligencia Artificial tiene como objetivo desarrollar máquinas que puedan realizar tareas tan bien como los humanos, o incluso mejor, dándoles un comportamiento inteligente a dichas máquinas. Comportamiento inteligente involucra percepción, razonamiento, aprendizaje, comunicación y actuación, en ambientes complejos. Otro objetivo de la Inteligencia Artificial consiste en entender que clase de comportamiento ocurre en los humanos, animales y máquinas. Así, la Inteligencia Artificial tiene objetivos científicos e ingenieriles, con el fin de desarrollar sistemas que puedan llevar a cabo actividades complejas como las realizadas por el humano, tales como comprensión del lenguaje natural, reconocimiento de imágenes, razonamiento simbólico, etc.

Ciertos grupos de científicos agrupan los trabajos de la inteligencia artificial en dos categorías: los enfoques basados en procesamiento simbólico, apoyados por los trabajos de Newell y Simon, los cuales conforman el enfoque clásico de inteligencia

artificial cuyo principio consiste en el uso de operadores lógicos aplicados a bases de conocimientos declarativas para realizar un proceso de razonamiento lógico. Estos enfoques, en todas sus variantes, se han agrupado en los llamados enfoques basados en conocimiento. La segunda categoría, llamada conexionista, hace énfasis en el desarrollo de símbolos como respuesta a un comportamiento emergente de las interacciones con el ambiente. Las redes neuronales representan el enfoque mas conocido de esta área. Pero, como ya se dijo y veremos mas adelante, esta clasificación de la inteligencia artificial no tiene un consenso dentro de la comunidad científica, ya que ciertos autores no incluyen las redes neuronales artificiales en el área de inteligencia artificial, sino que lo consideran perteneciente al área de vida artificial [2].

La Inteligencia Artificial Distribuida es un subcampo de la Inteligencia artificial que distribuye el cómputo a la hora de resolver problemas [2, 7, 9, 10]. La Inteligencia Artificial Distribuida se diferencia de la inteligencia artificial clásica en que individualiza el concepto de inteligencia, centrando más su interés en las interacciones de los elementos y menos en el modo individual de razonamiento de ellos. El área de la Inteligencia artificial Distribuida comprende [1, 2, 9, 10]: los mecanismos de resolución de problemas distribuidos, los sistemas multiagentes y la Inteligencia artificial Paralela. Esta área ha mostrado ser capaz de obtener un comportamiento colectivo complejo, sin que los elementos individualmente tengan facultades impresionantes, a partir de mecanismos de cooperación y de coordinación de acciones.

La Vida Artificial busca comprender y modelar sistemas provistos de vida, es decir, capaces de sobrevivir, de adaptarse y de reproducirse, en medios usualmente hostiles [1, 2, 9]. El área vida artificial fue definido por C. Langton como "el area que estudia la vida como ella podría ser, y la vida como ya es" [9]. Así, se trata de abstraer principios subyacentes a las organizaciones vivientes y de implementarlos en un computador para probarlos y estudiarlos. Contraria al área clásica de inteligencia artificial que considera que la inteligencia procede de una manipulación eficiente de símbolos, ella se concentra en el comportamiento, la autonomía, y sobre todo, la problemática de la vitalidad de los seres vivientes: "para que un ser viva, es necesario que se encuentre en el interior de su dominio de vitalidad" [9].

Una *Red Neuronal Artificial* es un paradigma inteligente basado en el modelo masivamente paralelo del cerebro. Las redes neuronales artificiales están inspiradas en el funcionamiento de las células nerviosas en el cerebro. Estos modelos simulan una estructura computacional paralela altamente interconectada compuesta de muchos, relativamente simples, elementos de procesamiento. Entre las debilidades mas grandes de esta técnica esta la no existencia de una representación estructural del conocimiento, la dificultad para interactuar con bases de datos convencionales, y su imposibilidad para explicar las razones de una conclusión alcanzada. Una presentación mas extensa de esta técnica será hecha en el capítulo 4.

Los Sistemas Expertos, también conocidos como sistemas basados en conocimiento, tienen como objetivo desarrollar programas que puedan emplear experiencia y conocimiento humano para tratar con problemas que usualmente requieren razonar y pensar [1, 7]. Es una de las técnicas mas populares del área Inteligencia Artificial. Entre sus

debilidades esta lo lento y difícil para capturar el conocimiento, su imposibilidad para tratar con información imprecisa o incompleta, la explosión combinatoria de las reglas para ciertos problemas, su imposibilidad para operar bajo restricciones de tiempo, y sus problemas para recuperar y aprovechar situaciones pasadas. Una presentación más extensa de esta técnica será hecha en el capítulo 3.

La Lógica Difusa es una técnica de "razonamiento aproximado", la cual comprende operaciones sobre conjuntos difusos, tales como unión, intersección, etc. [1, 7]. Los conjuntos difusos modelan propiedades de imprecisión, aproximación y vaguedad en la información. La lógica difusa puede ser vista como una generalización de la lógica convencional (basada en dos valores). En lógica convencional, un elemento es miembro o no de un conjunto, de tal forma que cada elemento tiene un grado de pertenencia de 0 o 1 a ese conjunto. En los conjuntos difusos, valores entre 0 y 1 reflejan el grado de pertenencia de cada elemento a ese conjunto. Entre las desventajas de los sistemas difusos esta el problema de adquisición de conocimiento, y algunas limitaciones para adaptarse y aprender. Una presentación mas extensa a esta técnica será hecha en el capítulo 2.

La Computación Evolutiva esta compuesta por todos los paradigmas inspirados en los mecanismos de la evolución biológica, tales como la biología genética y la selección natural. Esta área abarca a los Algoritmos Genéticos, la Programación Genética y Evolucionaria, y las Estrategias Evolutivas. La técnica más conocida son los algoritmos genéticos. Una de las desventajas mas importantes de estas técnicas es que son computacionalmente costosas. Una presentación mas extensa a estas técnicas será hecha en el capítulo 5.

1.3 Adaptación vs Aprendizaje

El concepto de *adaptación* es fundamental en computación inteligente. Según el nuevo diccionario Colegial Webster, adaptación es [1, 11]: "1. el acto o proceso de adaptarse, 2. el ajuste a condiciones ambientales: ajuste de un órgano sensitivo a la intensidad o calidad del estímulo, o modificación de un organismo o sus partes, para ser más apta su existencia a las condiciones de su ambiente". De la misma fuente, adaptar es: "hacer apto algo (tanto para un específico, o un nuevo uso o condición), frecuentemente a través de modificaciones". Estas definiciones encajan perfecto en las características esenciales del área Computación Inteligente.

Por otro lado, la palabra *aprendizaje* puede ser definida como "conocimiento o experiencia adquirida por instrucciones o estudio". Además, aprender es definido como "ganancia en conocimiento, en comprensión, o en experiencia, derivada de estudios, instrucciones o experiencias" (nuevo diccionario Colegial Webster [11]).

En la figura 1.2 se muestran todos los elementos de un sistema artificial inteligente, los cuales intercambian información a través del modelo del mundo. El modelo del mundo se comporta como el medio de almacenamiento, categorizando los diferentes tipos de información posibles de manipular. Aprendizaje aplica a todo el sistema, pero

adaptación aplica solamente al área de computación inteligente. La adaptación debe sobrepasar muchas barreras, incluyendo óptimos locales y no linealidades. El sistema adaptativo responde a retos que se le van presentando, modificando sus estructuras a través de diferentes mecanismo durante su evolución. En resumen, la adaptación es el más apropiado término para referirse a lo que hace la computación inteligente.

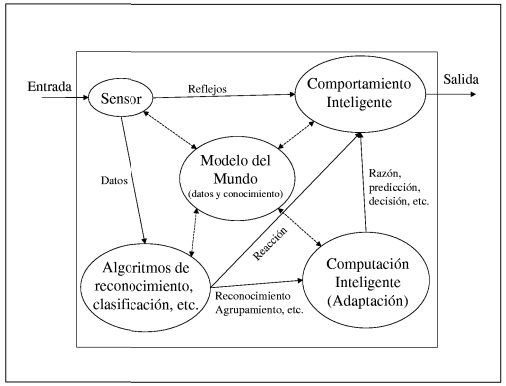

Figura 1.2: Sistema Artificial Inteligente

1.4 Procesos Caóticos y Estocásticos

Los procesos estocásticos juegan un rol fundamental en los paradigmas computacionales inteligentes. Ejemplos de esto los encontramos, en la inicialización aleatoria de los pesos para algunos algoritmos de entrenamiento neuronal, o en el uso estocástico de los operadores involucrados en las técnicas evolutivas, entre otros.

La caracterización de *estocástico* que hacemos de algún proceso físico, proviene de observaciones de su comportamiento *dinámico-caotico*. Una definicion suscinta de caos es la de "determinismo desordenado". Entre los atributos de los sistemas caóticos, tenemos: no predictividad, evolución temporal de la incertidumbre, dificultad creciente para acceder información sobre sus condiciones iniciales en la medida que el tiempo crece, y comportamiento no periódico.

El rol del caos en la biología es un tema de estudio de interés actual. Los patrones climáticos y muchos otros sistemas físicos caóticos poseen un gran efecto en la evolución y determinación de características humanas (ver trabajos de Lorenz y otros [1]). Esta afirmación pone en relieve la importancia de estos sistemas en el comportamiento inteligente, y a su vez, nos lleva a concluir que el uso de funciones caóticas, en vez de

funciones pseudo-aleatorias, en las implementaciones de computación inteligente permitiría modelar de manera mas precisa el comportamiento o la ocurrencia de eventos en sistemas biológicos.

Bibliografía

[1] R. Berhart, P. Simpson, R. Dobbins, "Computational Intelligence: PC Tools", *AP Professional Academic Press*, 1996

[2] J. Ferber, *Les Systèmes Multi-Agents vers Intelligence Collective*, *Inter Editions*, 1995.

[3] J. Ferdinder, A. Moreno, "La Vie Artificielle", *Editions Francaise*, 1992

[4] S. Goonatilake, S. Khebbak, "Intelligent Hybrid System", *John Wiley and Son*, 1998.

[5] J. Heudin, "La Vie Artificiel", *Hermes Editions*, 1994

[6] L. Jain, R. Jain (Ed.), "Hybrid Intelligent Engineering Systems", *1997 World Scientific Publishing*, 1997

[7] R. Khosla, T. Dillon, "Engineering Intelligent Hybrid Multi-Agent Systems", *Kluwer Academic Publishers*, 1997

[8] C. Langton (Ed), *Artificial Life III*, *Addison Wesley*, 1994.

[9] C. Langton (Ed.), "Artificial Life", *MIT Press* , 1995

[10] N. Nilsson, "Artificial Intelligence", *Morgan Kaufmann Publisher INC.*, 1998

[11] WEBSTER'S New Collegiate Dictionary, *C.Merriam Company*, 1975.

Lógica Difusa

Mariela Cerrada Lozada
CEMISID
Departamento de Sistemas de Control
Facultad de Ingeniería
Universidad de Los Andes
Mérida 5101, Venezuela
E-mail: cerradam@ula.ve

Wladimir Rodríguez Graterol
Departamento de Computación
Facultad de Ingeniería
Universidad de Los Andes
Mérida 5101, Venezuela
E-mail: wladimir@ula.ve

Capítulo 2

Lógica Difusa

2.1 Introducción

En la naturaleza y los procesos creados por el hombre, ciertos comportamientos son entendidos basándose en razonamientos cualitativos más que cuantitativos. Una de las razones se debe al carácter impreciso e incierto de los fenómenos a describir. Por otro lado, la forma del pensamiento humano, así como las asociación de ideas, tienen una connotación más imprecisa que exacta. Por ejemplo, la discriminación por parte de la mente de humana de las personas "altas", "medias" ó "bajas", dentro de un grupo de personas, obedece más a una clasificación basada en criterios subjetivos que en criterios exactos, respecto de lo que se piensa sobre la medida límite que sirva como referencia para decidir sobre si una persona es realmente alta ó no. Así como este ejemplo sencillo, existen otros fenómenos más complejos donde la forma de caracterizarlos ó describirlos responde a criterios "difusos".

El desarrollo de la Lógica Difusa fue motivado, en gran medida, por la necesidad de crear un marco conceptual para la representación flexible del conocimiento, dentro de un ambiente impreciso e incierto [1]. Su importancia radica en el hecho de que los modos del razonamiento humano, por su naturaleza, son aproximados.

El concepto central de la Lógica Difusa es el Conjunto Difuso, el cual es considerado como una extensión de los conjuntos ordinarios, en el trabajo desarrollado por L. Zadeh en 1965 [2]. En la teoría sobre Conjuntos Difusos, la pertenencia o nó de un elemento del universo a un conjunto no esta enmarcada dentro de una lógica bivaluada de 0 ó 1, para indicar la no pertenencia o pertenencia, respectivamente, al conjunto dado; sino que la pertenencia de un elemento determinado a un conjunto dado obedece a una lógica multivaluada dentro del intervalo [0,1], que asigna al elemento un "grado de pertenencia"que puede ir desde la no pertenencia (0) hasta la pertenencia total (1). Así pues, dentro de esta idea y utilizando el ejemplo anterior, la "altura"de una persona es considerada una "variable difusa", que puede estar clasificada por de tres "conjuntos difusos"etiquetados como "bajo", "medio", "alto", donde la transición de la pertenencia de un conjunto a otro, de una persona con una altura x, es gradual.

En Lógica Difusa, la proposición *María es Alta* es una restricción difusa sobre la altura de María, y su grado de veracidad depende del grado en que la altura de María pertenenezca al conjunto difuso "alto". Así pues, en Lógica Difusa todo es cuestión de grado, donde el razonamiento exacto es pensado como un caso límite del

razonamiento aproximado. El conocimiento, dentro de esta lógica, es interpretado como una colección de proposiciones difusas, expresadas en lenguaje natural, sobre un conjunto de variables, y sometidas a una proceso de inferencia para determinar su veracidad.

Por otro lado, es posible introducir conectores lógicos de la forma Y, O y reglas de producción de la forma SI (antecedente) ENTONCES (consecuente), las cuales han demostrado ser poderosas para la toma de desiciones, para formular relaciones entre un conjunto de variables para generar ciertas conclusiones. De esta manera, algunas proposiciones como *Temperatura es Alta*, *Humedad es Alta*, *Día es Caluroso* pueden relacionarse en una proposición de la forma:

SI (Temperatura es Alta) Y (Humedad es Alta) ENTONCES (Día es Caluroso).

donde la veracidad de la proposición difusa del consecuente dependerá de la veracidad de las proposiciones difusas del antecedente.

Combinando los conceptos sobre conjuntos difusos con reglas de producción de la forma SI-ENTONCES, y permitiendo introducir cierta impresición o inexactitud, es posible formular modelos lógico difusos para describir fenómenos complejos difíciles de modelar con métodos formales, haciendo de la Lógica Difusa una herramienta poderosa para la conceptualización de sistemas. El propósito de estos modelos es "capturar"el funcionamiento de los sistemas bajo estudio [3], a través de reglas difusas con antecedentes asociados a las variables de entrada al sistema y consecuentes asociados a sus variables de salida. Estos modelos son llamados "modelos basados en reglas", "modelos basados en conocimiento" simplemente "modelos difusos".

El proceso de obtener un valor de salida a partir de un valor de entrada al modelo difuso es llamado "mecanismo de inferencia o de razonamiento". Si además los valores de las variables de entrada no son difusos sino puntuales o exactos, debe existir un proceso de difusificación de estas variables para poder aplicarse el modelo difuso. Finalmente, siendo que la salida del proceso de inferencia es un valor difuso, es necesario un proceso de desdifusificación para transformar los valores difusos de la variables de salida en valores exactos.

En este capítulo revisaremos los aspectos fundamentales sobre el modelado difuso. En la primera sección revisaremos los conceptos básicos sobre la teoría de conjuntos difusos. En la segunda sección mostraremos algunos conceptos sobre la teoría del razonamiento aproximado, la cual es la base de los mecanismos de inferencia de modelos linguísticos. En la tercera sección estableceremos una clasificación sobre los tipos de modelos difusos existentes, así como los mecanismos de inferencia asociados. En la cuarta sección hablaremos sobre los mecanismos de difusificación y desdifusificación. En la última sección, mostraremos las aplicaciones del modelado difuso en el diseño de controladores, detección y diagnóstico de fallas, etc.

2.2 Teoría de Conjuntos Difusos. Conceptos Básicos

El concepto de conjunto difuso fué introducido por L. Zadeh como una generalización de la idea sobre los conjuntos ordinarios [2]. Recordemos, que para un conjunto ordinario definido sobre un dominio determinado, la pertenencia o nó de un elemento de dicho dominio al conjunto ordinario esta perfectamente definida. Así, un elemento cualquiera a pertenece o no pertenece a algún conjunto de elementos. Para los conjuntos difusos, la pertenencia o nó de un elemento del dominio a dicho conjunto tiene un carácter multivaluado, es decir, no existe una frontera entre la pertenencia o no de un elemento al conjunto ó dicho de otra manera, la transición entre pertenencer on nó pertenecer a un conjunto, es gradual.

A continuación mostraremos algunas definiciones fundamentales sobre conjuntos difusos, las cuales, en algunos casos, son extensiones de las definiciones dadas en los conjuntos ordinarios, sin embargo, existen definiciones propias para los conjuntos difusos que no tienen sentido para los conjuntos ordinarios, en virtud de la caracterización misma de un conjunto difuso [3]. Particularmente, las operaciones de unión e intersección en conjuntos ordinarios, así como las propiedades de conmutatividad, idempotencia, asociatividad y distributividad relativas a estas operaciones, son aplicadas de igual manera a los conjuntos difusos.

2.2.1 Definiciones

Recordemos que un conjunto ordinario A esta definido sobre otro conjunto ordinario X. La función característica asociada a A es un mapa:

$$\mu_A : X \to \{0, 1\} \tag{2.1}$$

que asigna a cada elemento $x \in X$ el valor de 1 si x es un elemento de A, o asigna un 0 si x no es un elemento de A. Esta función característica se ilustra en la figura 2.1.

Un conjunto difuso también es definido sobre un dominio ordinario llamado el *universo de discurso*, pero extiende el rango de la función característica sobre todo el intervalo unitario.

Definición 1. Sea $X = \{x_1, x_2, ..., x_n\}$ un universo de discurso. Luego, D es un conjunto difuso sobre X, con una función característica:

$$\mu_D : X \to [0, 1] \tag{2.2}$$

que asocia a cada $x_i \in X$ un valor sobre el intervalo unitario $I = [0, 1]$

Usualmente, en el marco de la teoría de conjuntos difusos, la función característica de un conjunto difuso es llamada *Función de Membresía*, e indica el *grado* con el que un elemento $x_i \in X$ pertence a dicho conjunto difuso, comunmente llamado *grado de membresía*. Se debe notar que valores del grado de membresía cercanos a 1 indican

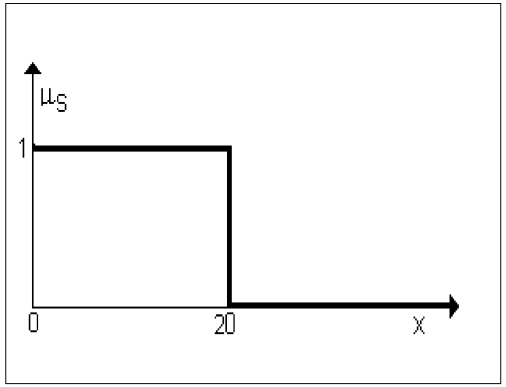

Figura 2.1: Función Característica de un Conjunto Ordinario

una pertenencia fuerte al conjunto difuso, mientras que valores cercanos a 0 denotan una pertenencia débil a dicho conjunto. Es importante resaltar que el valor del grado de membresía depende del contexto bajo el cual se esta considerando la definición de conjuntos difusos

En la figura 2.2, se muestra un conjunto difuso sobre la línea real X, definido por su función de membresía. Un par de funciones de membresía muy utilizadas son las funciones triangulares y trapezoidales, las cuales son mostradas en las figuras 2.3 y 2.4 respectivamente.

Si el universo de discurso es la línea real, las funciones de membresía pueden ser representadas en forma funcional, por ejemplo, $\mu_D(x) = \frac{1}{1+x^2}$. Otra forma de expresar el grado de membresía es explícitamente, a través de una representación de conjunto, es decir, un conjunto difuso D sobre X esta dado por el siguiente conjunto:

$$D = \{0.7/x_1, 0.3/x_2, 1/x_3, 0/x_4, ..., v_n/x_n\}$$

Los términos de la forma v_i/x_i, indican que el elemento x_i de X tiene grado de membresía v_i en el conjunto difuso D.

Definición 2. Un conjunto difuso D de X es llamado *normal* si existe al menos un elemento $x_i \in X$ tal que $\mu_D(x_i) = 1$. Un conjunto difuso que no es normal se le llama *subnormal*

Nótese que, según esta definición, cualquier conjunto ordinario excepto el conjunto nulo, es un conjunto normal. El conjunto ordinario nulo, claramente, generaliza la subnormalidad.

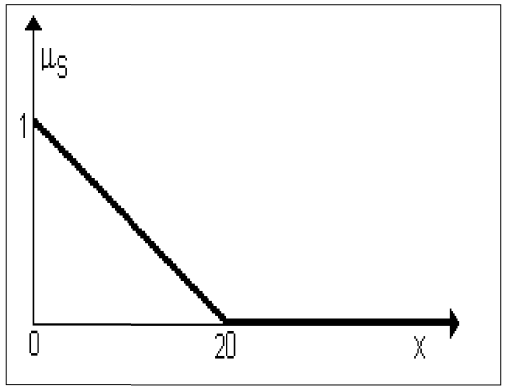

Figura 2.2: Función de Membresía de un Conjunto Difuso

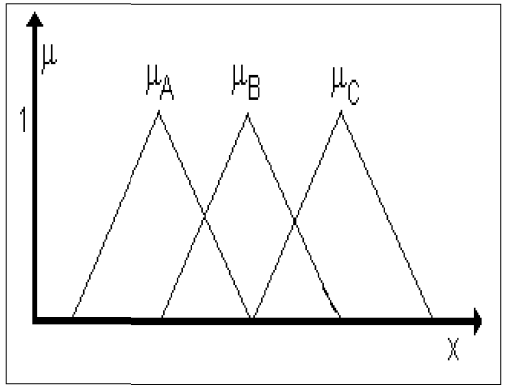

Figura 2.3: Funciones de Membresía Triangulares

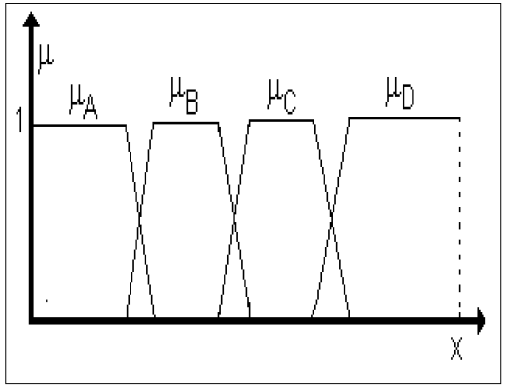

Figura 2.4: Funciones de Membresía Trapezoidales

Definición 3. El *Peso* de un conjunto difuso D es el mayor grado de membresía dado para un elemento en D, denotado como $Peso(D)$. Luego,

$$Peso(D) = Max_x \mu_D(x) \tag{2.3}$$

Nótese que un conjunto difuso *normal* tiene *peso* igual a 1.

Definición 4. Sea D un conjunto difuso sobre X. El *soporte* de D, denotado por $Soporte(D)$, es el subconjunto ordinario de X cuyos elementos tienen todos grados de membresía diferentes de cero en D. Es decir,

$$Soporte(D) = \{x \,|\mu_D(x) > 0, \forall x \in X\} \tag{2.4}$$

Definición 5. Sea D un conjunto difuso sobre X. El *núcleo* de D, denotado por $Nucleo(D)$, es el conjunto ordinario de X que consiste de todos los elementos de D con grado de membresía igual a 1. Es decir,

$$Nucleo(D) = \{x \,|\mu_D(x) = 1, \forall x \in X\} \tag{2.5}$$

Nótese, que un conjunto difuso *normal* tiene un *núcleo* no nulo, mientras que un conjunto difuso *subnormal* tiene un *núcleo* nulo.

Definición 6. Sean D y E conjuntos difusos sobre X. D es un *subconjunto* de E, denotado por $D \subset E$, si $\mu_E(x) \geq \mu_D(x)$ para cada $x \in X$.

Definición 7. Sean D y E conjuntos difusos sobre X. D y E son *iguales*, denotado por $D = E$, si $D \subset E$ y $E \subset D$. Luego, $D = E$ si y solo si $\mu_D(x) = \mu_E(x)$, para $x \in X$.

Definición 8. El *conjunto difuso nulo* de X, denotado por ϕ, el conjunto difuso tal que $\mu_\phi(x) = 0$ para cada $x \in X$.

Nótese, que para cualquier conjunto difuso D en X, $\phi \subset D$.

Definición 9. Sea D un conjunto difuso sobre X. La *cardinalidad difusa* de D, denotada por $P(D)$ es definida como:

$$P(D) = \sum_{i=1}^{n} \mu_D(x_i),\ \forall x_i \in X \tag{2.6}$$

2.2.2 Operaciones Sobre Conjuntos Difusos

A continuación definiremos algunas operaciones sobre los conjuntos difusos. Como se mencionó anteriormente, alguna de estas operaciones son usualmente definidas para conjuntos ordinarios, por lo tanto, mantendremos la misma notación. Otras operaciones son solo definidas para conjuntos difusos estrictamente.

Definición 10. Sean D y E conjuntos difusos sobre X. La *unión* de estos conjuntos difusos, denotada por $D \cup E$, es otro conjunto difuso F de X, tal que:

$$\begin{aligned} \mu_F(x) &= Max\left[\mu_D(x), \mu_E(x)\right] \\ &= \mu_D(x) \vee \mu_E(x),\ \forall x \in X \end{aligned} \tag{2.7}$$

donde $\vee$ denota al operador Max (máximo). Esta operación es llamada *Unión Estándar*. En la figura 2.5 se ilustra la aplicación de esta operación.

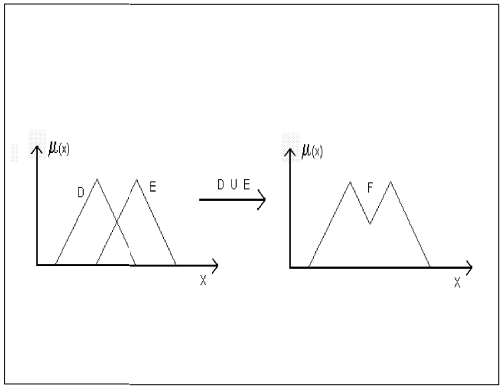

Figura 2.5: Operación de Unión Difusa Estándar

Ahora bien, la operación de *unión* puede ser especificada formalmente como una función U que devuelve el grado de membresía del elemento en el conjunto $D \cup E$. Luego:

$$\mu_{(D \cup E)}(x) = U[\mu_D(x), \mu_E(x)],\ \forall x \in X \tag{2.8}$$

Las propiedades que debe cumplir una función U para ser aceptada como una función de unión difusa, son las mismas propiedades de las funciones que son conocidas como *t-conormas*. Una función *t-conorma* es una operación binaria sobre el intervalo unitario que satisface, al menos, los siguientes axiomas para todo $a, b, d \in [0, 1]$:

- **Axioma 1.** $U(a, 0) = a$ (condición de frontera)
- **Axioma 2.** $b \leq d$ implica $U(a, b) \leq U(a, d)$ (monoticidad)
- **Axioma 3.** $U(a, b) = U(b, a)$ (conmutatividad)
- **Axioma 4.** $U(a, U(b, d)) = U(U(a, b), d)$ (asociatividad)
- **Axioma 5.** U es una función contíua
- **Axioma 6.** $U(a, a)) < a$ (superidempotencia)
- **Axioma 7.** $a_1 < a_2$ y $b_1 < b_2$ implica $U(a_1, b_1) < U(a_2, b_2)$ (monoticidad estricta)

A continuación se enumeran algunas de las funciones t-conormas frecuentemente usadas como funciones de unión difusa:

1. **Suma Algebraica.** $U(a, b) = a + b - ab$
2. **Suma Acotada.** $U(a, b) = min(1, a + b)$
3. **Unión Drástica.**

$$U(a, b) = \begin{cases} a & si \quad b = 0 \\ b & si \quad a = 0 \\ 1 & si \quad otrocaso \end{cases} \tag{2.9}$$

Definición 11. Sean D y E conjuntos difusos sobre X. La *intersección* de estos conjuntos difusos, denotada por $D \cap E$, es otro conjunto difuso G de X, tal que:

$$\begin{aligned} \mu_G(x) &= Min\left[\mu_D(x), \mu_E(x)\right] \\ &= \mu_D(x) \wedge \mu_E(x),\ \forall x \in X \end{aligned} \tag{2.10}$$

donde $\wedge$ denota al operador Min (mínimo). Esta operación es llamada *Intersección Estándar*. En la figura 2.6 se ilustra la aplicación de esta operación.

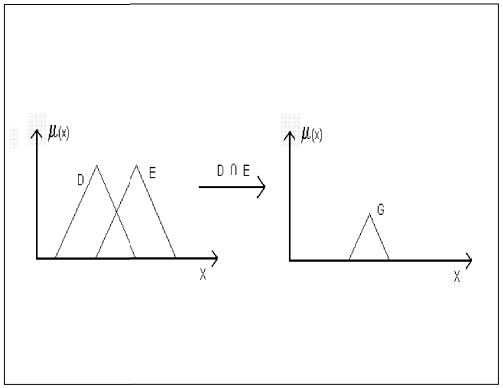

Figura 2.6: Operación de Intersección Difusa Estándar

Análogamente que en la operación de unión difusa, la operación de *intersección* puede ser especificada formalmente como una función I que devuelve el grado de membresía del elemento en el conjunto $D \cup E$. Luego:

$$\mu_{(D \cap E)}(x) = I[\mu_D(x), \mu_E(x)], \; \forall x \in X \tag{2.11}$$

Las propiedades que debe cumplir una función I para ser aceptada como una función de intersección difusa, son las mismas propiedades de las funciones que son conocidas como *t-normas*. Una función *t-norma* es una operación binaria sobre el intervalo unitario que satisface, al menos, los siguientes axiomas para todo $a, b, d \in [0, 1]$:

- **Axioma 1.** $I(a, 1) = a$ (condición de frontera)
- **Axioma 2.** $b \leq d$ implica $I(a, b) \leq I(a, d)$ (monoticidad)
- **Axioma 3.** $I(a, b) = I(b, a)$ (conmutatividad)
- **Axioma 4.** $I(a, I(b, d)) = I(I(a, b), d)$ (asociatividad)
- **Axioma 5.** I es una función contíua
- **Axioma 6.** $I(a, a)) < a$ (subidempotencia)
- **Axioma 7.** $a_1 < a_2$ y $b_1 < b_2$ implica $I(a_1, b_1) < I(a_2, b_2)$ (monoticidad estricta)

A continuación se enumeran algunas de las funciones t-normas frecuentemente usadas como funciones de intersección difusa:

1. **Producto Algebraico.** $I(a, b) = ab$
2. **Diferencia Acotada.** $I(a, b) = max(0, a + b - 1)$
3. **Intersección Drástica.**

$$I(a, b) = \begin{cases} a & si \quad b = 1 \\ b & si \quad a = 1 \\ 1 & si \quad otrocaso \end{cases} \tag{2.12}$$

En un sentido general, las operaciones de intersección y unión sobre conjuntos difusos pueden verse como una *agregación* de conjuntos difusos, es decir, la obtención de otro conjunto difuso, cuya función de membresía está definida por algún tipo de operación sobre las funciones de membresía de un cierto número de conjuntos difusos [16, 15].

Teorema 1. Sean D y E conjuntos difusos sobre X. Si $F = D \cup E$ y $G = D \cap E$, entonces:

1. $G \subset F$
2. $D \subset F$ y $E \subset F$
3. $G \subset D$ y $G \subset E$

Prueba. Se deja como ejercicio al lector.

Los conjuntos difusos cumplen propiedades de conmutatividad, idempotencia, asociatividad, distributividad, entre otras, que son propiedades relativas a las operaciones de unión e intersección definidas originalmente sobre conjuntos ordinarios. A continuación, se listan estas propiedades, considerando que A, B y C son subconjuntos difusos sobre X.

1. **Conmutatividad**
   $A \cup B = B \cup A$
   $A \cap B = B \cap A$
2. **Idempotencia**
   $A \cup A = A$
   $A \cap A = A$
3. **Asociatividad**
   $A \cup (B \cup C) = (A \cup B) \cup C = A \cup B \cup C$
   $A \cap (B \cap C) = (A \cap B) \cap C = A \cap B \cap C$

4. **Distributividad**
   $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
   $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

5. $A \cup \phi = A \qquad A \cup X = X$
   $A \cap \phi = \phi \qquad A \cap X = A$

6. $A \subset A \cup B$
   $A \supset A \cap B$
   $A \cap B \subset A \cup B$

7. Si $A \subset B$ entonces
   $B = A \cup B$
   $A = A \cap B$

8. Si $A \subset B$ y $B \subset C$ entonces $A \subset C$

Definición 12. Sean D y E conjuntos difusos sobre X. El *complemento relativo* de E con respecto de D, denotado por $D - E$, se define como el conjunto F de X donde:

$$\mu_F(x) = Max\left[0, \mu_D(x) - \mu_E(x)\right], \forall x \in X \tag{2.13}$$

Definición 13. Sea D un conjunto difuso sobre X. El *complemento o negación* de D, denotado por $\overline{D}$, se define como el conjunto difuso $\overline{D} = X - D$, tal que:

$$\mu_{\overline{D}}(x) = 1 - \mu_D(x), \forall x \in X \tag{2.14}$$

En la figura 2.7 se ilustra la aplicación de esta operación.

A continuación, se dan algunas definiciones que no son usadas en los conjuntos ordinarios.

Definición 14. Sea D un conjunto difuso sobre X y α cualquier número nonegativo. Luego se denota por D^α al conjunto difuso B de X, tal que:

$$\mu_B(x) = (\mu_D(x))^\alpha \tag{2.15}$$

Nótese que si $\alpha > 1$, entonces $D^\alpha \subset D$, y si $\alpha < 1$, entonces $D^\alpha \supset D$. Si $\alpha > 1$ se denomina operación de *concentración* y si $\alpha < 1$ se denomina operación de *dilatación*. Por otro lado, si D es un conjunto ordinario, entonces $D^\alpha = D$.

En la figura 2.8 se ilustra el efecto de estas operaciones sobre un conjunto difuso.

Definición 15. Sea D un conjunto difuso sobre X y α cualquier número sobre el intervalo unitario. Se define al conjunto difuso $F = \alpha D$ tal que $\mu_F(x) = \alpha\mu_D(x)$

Definición 16. Sea D un conjunto difuso sobre X. El *conjunto α-corte* de D, denotado por D_α, es un conjunto ordinario de X que consiste de todos los elementos de X para los cuales $\mu_A(x) \geq \alpha$. De aquí:

$$D_\alpha = \{x \,|\mu_D(x) \geq \alpha, \forall x \in X\} \tag{2.16}$$

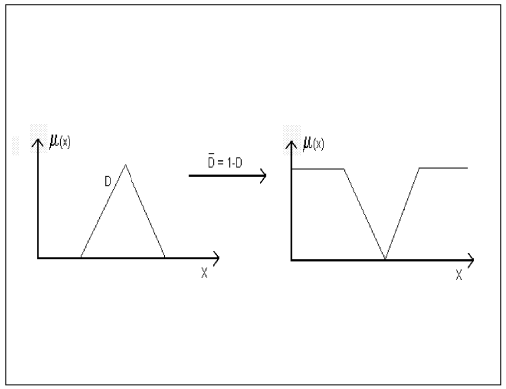

Figura 2.7: Operación de Complemento o Negación Difusa

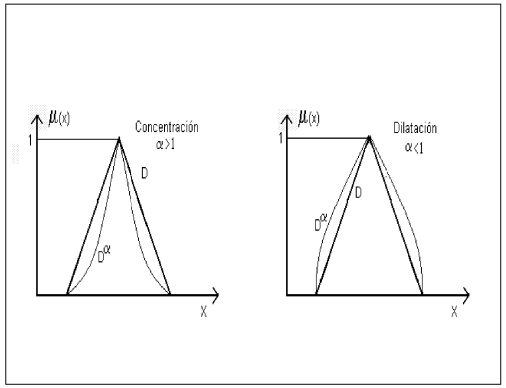

Figura 2.8: Operaciones de Concentración y Dilatación

Nótese que los conjuntos α-corte permite la representación de cualquier conjunto difuso en un conjunto ordinario.

2.2.3 Relaciones Difusas

Recordemos que si X y Y son dos conjuntos ordinarios, entonces su producto Cartesiano, denotado por $X \times Y$, consiste de todos los pares (x, y) donde $x \in X$ y $y \in Y$. A continuación, extenderemos el concepto de relación entre conjuntos ordinarios, definiendo una relación difusa.

Definición 17. Una *relación difusa* sobre el par X,Y, es definida como un conjunto difuso sobre el espacio de producto Cartesiano $X \times Y$.

En forma más general, si $X_1, X_2, ..., X_n$ son una colección de conjuntos, una relación difusa es un conjunto difuso definido sobre el espacio de su producto Cartesiano $X_1 \times X_2 \times ... \times X_n$. El concepto clásico de relación no es mas que un caso particular de le definición arriba dada, donde los grados de membresía son ceros ó unos.

Este concepto es sumamente útil para de capturar imprecisiones dadas en ciertas relaciones. Por ejemplo, si X representa un conjunto de personas y Y representa un conjunto de tamaños, entonces la relación entre las personas que pueden tener un tamaño particular puede capturarse a través de una relación difusa R sobre $X \times Y$, donde $\mu_R(x, y)$ indica el grado con el cual una persona x tiene cierto tamaño y.

Puesto que una relación difusa no es mas que un conjunto difuso, entonces todos los mecanismos para manejar conjuntos difusos pueden utilizarse para manipular relaciones difusas. A continuación, se define un mecanismo para obtener relaciones difusas a partir de conjuntos difusos, extendiendo la noción del producto Cartesiano.

Definición 18. Sean D Y E conjuntos difusos sobre X y Y respectivamente. Luego, su *producto cruz o producto Cartesiano* $D \times E$, es una relación difusa R sobre el conjunto $X \times Y$, donde:

$$\mu_R(x, y) = Min\left[\mu_D(x), \mu_E(y)\right] \tag{2.17}$$

2.2.4 Valores Linguísticos

Una de las características de los conjuntos difusos es la habilidad para la representación de algunos conceptos, a través de la definición de variables linguísticas. Por ejemplo, el concepto de *frío* puede ser definido como un subconjunto difuso sobre el conjunto de temperaturas entre 0 y 100 grados C. Si se tiene una variable V que puede tomar valores en el conjunto X, entonces podemos representar información acerca de esta variable a través de sentencias de la forma V *es* x, donde x es algún valor linguístico sobre el conjunto X. Así, si la variable V representa la temperatura entonces V *es fría* es una información acerca de V representada a través de una sentencia, donde *fría* es un valor linguístico, el cual puede ser representado como un conjunto difuso.

Consideremos como ejemplo, entonces, a la variable linguística temperatura sobre un rango de 0 a 100 grados C. Definimos para esta variable cuatro valores linguísticos como *Frío*, *Templado*, *Caliente* y *Muy Caliente* representados por conjuntos difusos sobre el dominio de valores de esta variable. Entonces, esta variable linguística no es mas que una variable difusa con un universo de discurso sobre los posibles valores de la temperatura, donde se definen conjuntos difusos dados por sus funciones de membresía, como se muestra en la figura 2.9:

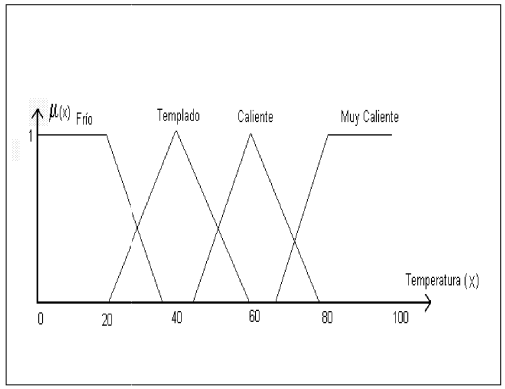

Figura 2.9: Partición difusa sobre el dominio de la variable Temperatura

Un valor linguístico es, en esencia, la asociación de un conjunto difuso con el valor de una variable, lo cual introduce cierta incertudumbre respecto al conocimiento que se tenga sobre el valor actual de la variable.

Si una premisa envuelve un valor definido x^* de X, tal como V *es* x^*, entonces la certeza de la expresión V *es* B, siendo B un conjunto difuso sobre X, viene dada por el *grado de verdad* definido como el grado de membresía de x^* en B, $\mu_B(x^*)$. Sin embargo, si una premisa envuelve un valor difuso V *es* A, el determinar el grado de certeza de la sentencia V *es* B no es sencillo. Por ejemplo, si tenemos el conocimiento de que *la temperatura esta bajo los 30 grados* no podemos tener certeza sobre la sentencia *La temperatura es fría*, pues posiblemente el valor actual de la temperatura esté sobre los 20 grados.

Si una premisa envuelve un valor difuso, una medida de confirmación es la *posibilidad*. Otra medida de confirmación es la *certeza*. Estas ideas de incertidumbre fue introducida por Zadeh [13] y desarrollada luego otros investigadores [14].

2.3 Generalidades Sobre la Teoría del Razonamiento Aproximado

La teoría del razonamiento aproximado, basada en el uso de los conjuntos difusos, fue introducida por Zadeh [3] para dar un marco al razonamiento a partir de información incierta. El concepto central de esta teoría es asignar conjuntos difusos como valores de variables y representar este par variable-valor difuso en sentencias o proposiciones linguísticas. El sistema de razonamiento llamado *Razonamiento Aproximado***RA** ofrece un mecanismo para modelar y hacer inferencias a partir de relaciones funcionales impresisas y constituye la base para derivar algunos de los mecanismos de inferencia conocidos (recuerde, como se mencionó en la primera sección, que el mecanismo de inferencia es el procedimiento seguido para obtener una conclusión a partir de una entrada dada a un modelo basado en reglas difusas). Esta teoría ha tenido grandes contribuciones por los trabajos desarrollados por Dubois, Prade y Yager [3].

2.3.1 Elementos Básicos de un Sistema RA

Los elementos básicos de un sistema RA son la coleccion de variables simples $V_1, ..., V_n$, y una colección de conjuntos $X_1, ..., X_n$, donde cada X_i es el *conjunto base* o el universo de discurso de la variable V_i. El conjunto X_i contiene los valores permisibles que puede tomar la variable V_i.

Una *variable conjunta* es alguna colección de una o más variables simples. Por ejemplo, V_a, V_b y V_c son variables conjuntas definidas por $V_a = (V_1, V_2)$, $V_b = (V_1, V_4, V_5)$ y $V_c = V_1$. Una *proposición* es una sentencia de la forma $V\ es\ M$, donde V es una variable conjunta, M es un conjunto difuso sobre el espacio del producto cartesiano de los conjuntos base de las variables simples que conforman a la variable conjunta. El producto cartesiano X de los conjuntos base de la variables V_i está definido sobre el universo de V. Así, M es una *relación difusa* sobre X. Una proposición de la forma dada es llamada una *sentencia canónica*.

Por ejemplo, si V_1 representa la altura de una persona y V_2 representa el peso, entonces la variable conjunta $V = (V_1, V_2)$ tiene como conjunto base los pares dados por la altura y el peso.

Así pues, definidos los elementos básicos de un sistema RA, definamos algunos conceptos que juegan un rol importante en la descripción de la incertidumbre contenida en las proposiciones.

Definición 19. Sean V_a y V_b dos variables conjuntas sobre las bases X y Y respectivamente. Sean $P_1 = V_a\ es\ M$ y $P_2 = V_b\ es\ N$ dos proposiciones. Su *conjunción* $P_1\ yP_2$, denotada como $P_1 \times P_2$ es la proposición:

$$V\ es\ P$$

donde V es una variable conjunta que consiste en la unión de las variables simples que confoman a V_a y V_b, y P es un conjunto difuso sobre el dominio de V, tal que para z en el dominio de V:

$$\mu_P(z) = \mu_M(x) \wedge \mu_N(x) \tag{2.18}$$

En 2.18, x es el elemento en X que concuerda con z sobre el dominio que tienen en común. Analogamente, y es el elemento en Y que concuerda con z sobre el dominio que tienen en común.

Por ejemplo, sean V_1, V_2 y V_3 son variables simples sobre los conjuntos base X, Y y Z, respectivamente. Si definimos dos variables conjuntas $V_a = (V_1, V_2)$ y $V_b = (V_2, V_2)$ y un par de proposiciones $V_a esM$ y $V_b esN$ donde M y N son conjuntos difusos dados por:

$$M = \left\{ \frac{0.7}{(x_1,y_1)}, \frac{0.9}{(x_1,y_2)}, \frac{1}{(x_2,y_1)}, \frac{0.2}{(x_2,y_2)}, \right\}$$
$$N = \left\{ \frac{0.8}{(y_1,z_1)}, \frac{1}{(y_1,z_2)}, \frac{0.4}{(y_2,z_1)}, \frac{1}{(y_2,z_2)}, \right\}$$

La conjunción de estas proposiciones es la proposición $V\ es\ P$, donde P es un conjunto difuso dado por:

$$P = \left\{ \frac{0.7}{(x_1,y_1,z_1)}, \frac{0.7}{(x_1,y_1,z_2)}, \frac{0.4}{(x_1,y_2,z_1)}, \frac{0.9}{(x_1,y_2,z_2)}, \right.$$
$$\left. \frac{0.8}{(x_2,y_1,z_1)}, \frac{1}{(x_2,y_1,z_2)}, \frac{0.2}{(x_2,y_2,z_1)}, \frac{0.2}{(x_2,y_2,z_2)} \right\}$$

Nótese que si las variables en la operación conjunción son las mismas, la operación de reduce a la operación *intersección* entre conjuntos difusos. Es decir:

$$V_a\ es\ M \cap V_a\ es\ N = V_a\ es\ M \cap N \tag{2.19}$$

Si V_a y V_b no tienen variables comunes, la operación conjunción resulta en el clásico producto cartesiano. Esto es:

$$V_a\ es\ M \cap V_b\ es\ N = (V_a, V_b)\ es\ M \times N \tag{2.20}$$

A continuación introducimos la definición de proyección de una proposición, el cual es otra operación importante dentro de la teoría de RA.

Definición 20. Sea P= $V_a\ es\ M$ una proposición y V_b alguna variable. La *proyección de* $V_a\ es\ M$ *sobre* V_b, denotada como $Proy_{V_b}[P]$, es la proposición $V_b\ es\ N$ donde:

$$\mu_N(z) = Max_Q\left[\mu_M(x)\right] \tag{2.21}$$

donde Q es el conjunto de todos los x que concuerdan con z sobre las variables que V_a y V_b tienen en común.

La manipulación del conocimiento a través de proposiciones difusas en un sistema de RA, están basadas fundamentalmente en las operaciones de *conjunción* y *proyección*. Recuerde, que la operación de conjunción es, esencialmente, una operación de producto cartesiano generalizado, y resulta útil para combinar proposiciones.

2.3.2 Mecanismo de Inferencia en un Sistema RA

En algunas aplicaciones complejas, especialmente en ambientes industriales, se establecen relaciones funcionales entre variables que pueden manejarse en un ambiente de razonamiento aproximado. De esta manera, una relación funcional $V = f(U)$ donde $U \in X$ y $V \in Y$, puede ser expresada como una relación F sobre el espacio del producto cartesiano $X \times Y$, contenida en una proposición difusa de la forma *(U,V) es F*. Esta proposición puede verse como la unión (o disjunción) de una colección de n proposiciones básicas, es decir:

$$F = \cup_{i=1}^{n} F_i \tag{2.22}$$

Esta proposiciones básicas son de la forma *Si U es A_i entonces V es B_i* donde A_i y B_i son conjuntos difusos de X y Y. Entonces:

$$(U,V) \; es \; F_i = U \; es \; A_i \times V \; es \; B_i \tag{2.23}$$

El mecanismo de inferencia basado en RA, sigue un proceso deductivo llamado *modus ponens generalizado* [17].

Definición 21. Sean U y V dos variables linguísticas y A', A, B', B conjuntos difusos. Entonces, el procedimiento de inferencia *modus ponens generalizado* se define como:

Premisa 1: *U es A'* (proposición de entrada)
Premisa 2: *Si U es A entonces V es B* (proposición que contiene la relación entre x y y
Consecuencia: *V es B'* (proposición inferida)

La proposición inferida se obtiene a partir del mecanismo propuesto por la teoría RA, formando la conjunción de las premisas P_1 y P_2 y formando la proposición:

$$(U,V) \; es \; G = (U,V) \; es \; F \times U \; es \; A' \tag{2.24}$$

donde $G = F \cap A'$.

Considerando que $F = \cup_{i=1}^{n} F_i$, donde $F_i = A_i \cap B_i$, entonces:

$$\begin{aligned} G &= (\cup_i F_i) \cap A' \\ &= \cup_i (A_i \cap B_i) \cap A' \\ &= \cup_i (A_i \cap A') \cap B_i \end{aligned} \tag{2.25}$$

De esta manera:

$$\begin{aligned} B' &= Proy_V[\cup_i (A_i \cap A') \cap B_i] \\ &= \cup_i Proy_V[(A_i \cap A') \cap B_i] \end{aligned} \tag{2.26}$$

Nótese que el conjunto difuso B' no es mas que la agregación, vía el operador unión, de todos las soluciones S_i provenientes de cada proposición P_i del conjunto de proposiciones $P_1, ...P_n$ La función de pertenencia del conjunto difuso B' viene dada por:

$$\mu_{B'}(y) = Max_i(Max_x[(A_i \wedge A') \wedge B_i] \tag{2.27}$$

La obtención del conjunto difuso B' en la proposición inferida a partir de la unión o disjunción de las relaciones F_i de cada proposición, es un procedimiento constructivo. En la siguiente sección mostraremos en detalle este método constructivo aplicado a cada uno de los posibles modelos difusos basados en reglas.

2.4 Clasificación de los Modelos Difusos

La complejidad del mundo real ha creado ciertas dificultades en el momento de establecer afirmaciones exactas sobre su comportamiento. El propósito de Zadeh, en el trabajo desarrollado en [4], de modelar el mecanismo del pensamiento humano con valores linguísticos más que numéricos, conduce a la introducción de vaguedad en la teoría de sistemas, dando lugar al desarrollo de sistemas difusos, como resultado de la difusificación del sistema convencional. Un modelo difuso es, entonces, una representación de la características de un sistema basado en la herramientas ofrecidas por la teoría de conjuntos difusos. Una característica de los sistemas difusos es la codificación difusa de la información, pues estos sistemas no operan bajo valores numéricos sino difusos, siendo estos valores difusos mas poderosos que la información que pueda brindar un número. De esta manera, la imprecisión o incertidumbre que pueda estar presente en los sistemas puede ser atrapada en la generalización de la infomación permitida por su codificación difusa.

Básicamente, podemos distinguir dos clases de modelos difusos, caracterizados por su habilidad para representar diferentes tipos de información. Una primera clase engloba a los modelos linguísticos basados en una colección de reglas SI-ENTONCES, con proposiciones difusas, que usa un razonamiento aproximado en el proceso de inferencia. La segunda categoría de modelos difusos usa reglas lógicas con antedecentes difusos y consecuentes funcionales (modelos híbridos). Esta clase de modelos fue formulada por Sugeno y sus colaboradores [5, 6], cuyo metodo de razonamiento integra la habilidad de los modelos linguísticos para la representación cualitativa del conocimiento con un potencial para expresar también información cuantitativa.

2.4.1 Modelos Linguísticos

Una de las características de los sistemas difusos es que pueden modelarse linguísticamente [4, 7, 8], usando reglas de decisión de la forma SI-ENTONCES. La habilidad en la toma de desiciones depende, entonces, de la existencia de una base de reglas y un mecanismo de razonamiento difuso. A continuación, presentaremos las estructuras genéricas de estas reglas, asi como los diferentes mecanismos de razonamiento para inferir el conocimiento a partir de dichas reglas.

1. *Reglas con una entrada y una salida*

   En este caso, el antecedente y el consecuente de la regla están constituídos por una sola variable. De esta manera, la estructura generica de esta regla es:

   $$Si\ U\ entonces V \tag{2.28}$$

   donde U es la variable de entrada del sistema difuso y V es la variable de salida. Si los universos de las variables difusas U y V tienen particiones dadas por conjuntos difusos B_i y D_i, respectivamente, etiquetados con valores linguísticos, entonces el siguiente conjunto de reglas son las diferentes **instancias** de la regla genérica 2.28:

   Si U es B_1 entonces V es D_1
   Tambien
   Si U es B_2 entonces V es D_2
   También
   ⋮
   También
   Si U es B_m entonces V es D_m

   Los conjuntos difusos B_i y D_i son considerados los parámetros del modelo linguístico y el número de reglas determinan su estructura, caracterizando la habilidad del modelo para la generalización. Cada instancia de la regla asocia a una región difusa de la variable de entrada con una región difusa en la variable de salida [3]. A este modelo linguístico se le denomina SISO, por su denotación en ingles *Single Input-Single Output.*

2. *Reglas con multiples enirada y una salida*

   En este caso el antecedente de la regla contempla multiples variables de entrada. Así, la estructura genérica de la regla es:

   $$Si\ U_1\ y\ U_2\ y....\ U_r\ entonces\ V \tag{2.29}$$

   Si consideramos que cada variable difusa de entrada U_i tiene tiene particiones en su universo de discurso determinadas por conjuntos difusos B_{ij}, donde i=(1,m); j=(1,r), asociados a variables linguísticas, entonces definen diferentes instancias de la regla genérica 2.29, mostradas a continuación:

   Si U_1 es B_{11} y U_2 es B_{12} y... y U_r es B_{1r} entonces V es D_1
   También
   Si U_1 es B_{21} y U_2 es B_{22} y... y U_r es B_{2r} entonces V es D_2
   También
   ⋮
   También
   Si U_1 es B_{m1} y U_2 es B_{m2} y... y U_r es B_{mr} entonces V es D_2

   El operador "Y"en el antecedente de la regla es interpretado como una intersección difusa de los conjuntos difusos B_{ij} sobre el espacio de entrada. A este modelo linguístico se le denomina MISO, por su denotación en ingles *Multiple Input-Single Output.*

3. *Reglas con multiples enirada y múltiples salidas*

   Consideremos ahora el caso mas general, donde tanto el antecedente como el consecuente tienen multiples variables. La estructura generica de esta regla es de la forma:

   $$Si\ U_1\ y\ U_2\ y....\ U_r\ entonces\ V_1; V_2; ...; V_r \tag{2.30}$$

   donde las variables difusas U_i son la variables difusas de entrada y las variables V_i son las variables difusas de salida. Si se considera una partición difusa de las estas variables definiendo conjuntos difusos B_{ij} y D_{ip}, donde i=1,...,m; j=1,...,r; p=(1,...,s),sobre el universo de discurso de las variables de entrada y salida respectivamente, entonces las instancias de la regla con las estructura dada en 2.30 vienen dadas por el siguiente conjunto de reglas:

   Si U_1 es B_{11} y U_2 es B_{12} y... y U_r es B_{1r} entonces V_1 es D_{11}; V_2 es D_{12};...; V_r es D_{1s}
   También
   Si U_1 es B_{21} y U_2 es B_{22} y... y U_r es B_{2r} entonces V_1 es D_{21}; V_2 es D_{22};...; V_r es D_{2s}
   También
   ⋮
   También
   Si U_1 es B_{m1} y U_2 es B_{m2} y... y U_r es B_{mr} entonces V_1 es D_{m1}; V_2 es D_{m2};...; V_r es D_{ms}

   A este modelo linguístico se le denomina MIMO, por su denotación en ingles *Multiple Input-Multiple Ouiput.* Una forma alternativa de los modelos linguísticos

MIMO es aquella donde se considera a los consecuentes difusos en 2.30 conectados con el operador "Y", es decir, la estructura genérica de la reglas es:

$$Si\ U_1\ y\ U_2\ y ... U_r\ entonces\ V_1\ y\ V_2\ y ... y\ V_r \tag{2.31}$$

Según esta estructura genérica, las diferentes instancias del modelo linguístico 2.31 estarán representadas por el siguiente conjunto de reglas:

Si U_1 es B_{11} y U_2 es B_{12} y... y U_r es B_{1r} entonces V_1 es D_{11} y V_2 es D_{12} y ... y V_r es D_{1s}
También
Si U_1 es B_{21} y U_2 es B_{22} y... y U_r es B_{2r} entonces V_1 es D_{21} y V_2 es D_{22} y... y V_r es D_{2s}
También
$\vdots$
También
Si U_1 es B_{m1} y U_2 es B_{m2} y... y U_r es B_{mr} entonces V_1 es D_{m1} y V_2 es D_{m2} y... y V_r es D_{ms}

En esta forma alternativa, el operador "Y"en el consecuente de la regla es interpretado como una intersección difusa de los conjuntos difusos D_{ij} sobre el espacio de salida.

2.4.2 Mecanismos de Razonamiento o Inferencia para Modelos Linguísticos

Se define al mecanismo de razonamiento o inferencia como el proceso de obtener un conjunto de valores (en principio difusos) para las variables de salida de un sistema, a partir de un conjunto de valores (en principio difusas) de la variables de entrada sobre la base del conocimiento del modelo difuso (con alguna de las estructuras definidas en la sección anterior), que explica el comportamiento del sistema bajo estudio.

En general, en virtud de la teoría de razonamiento aproximado, cada regla de la forma:

$$\text{Si } U \text{ es } B \text{ entonces } V \text{ es } D$$

define una relación difusa R sobre el universo del producto Cartesiano $X \times Y$. Esta relación difusa define un conjunto difuso con función de membresía denotada por $\mu_R(x, y)$. Para un valor difuso A de la variable de entrada U, se obtiene también otra relación difusa G sobre el universo del producto Cartesiano $X \times Y$ que define un conjunto difuso dado por la intersección difusa $A \cap R$, cuya función de membresía viene dada por:

$$\mu_G(x, y) = \mu_A(x) \wedge \mu_R(x, y) \tag{2.32}$$

La salida inferida por la regla, dada como entrada el valor difuso A, no es mas que un subconjunto difuso F del universo de la variable de salida V, obtenido como una proyección del conjunto difuso G sobre el espacio de salida Y. Es decir,

$$F = Proy_y G \tag{2.33}$$

La función de membresía de este conjunto F no es mas que:

$$\mu_F(y) = \vee_x \left(\mu_A(x) \wedge \mu_R(x, y)\right) \tag{2.34}$$

A este procedimiento de inferencia se le denomina la *Regla de Inferencia Max-Min.* Esta regla de inferencia no es mas que un mapeo que define una transformación del valor difuso de entrada en un valor difuso de salida.

A continuación, mostraremos la aplicación de esta regla para los diferentes modelos linguísticos revisados en la sección anterior, donde cada una de las instancias del modelo es considerada como una relación difusa R definidas sobre universo del producto cartesiano sobre los espacios de entrada y salida.

Mecanismo de Razonamiento Tipo Mamdani

Inspirado en Zadeh, A. Mamdani y sus colaboradores [9] combinaron las ideas de los sistemas basados en reglas con el uso de parámetros difusos para construir un controlador basado en el razonamiento de un operador humano. El método de razonamiento propuesto por Mandani para obtener la salida de un modelo basado en reglas estrictamente difusas esta basado en la superposición de las salidas de cada una de las reglas que componen el modelo difuso para una determinada entrada, dándole de éste modo una naturaleza constructiva al modelo.

Cada regla de la forma:

$$\text{Si } U \text{ es } B_i \text{ entonces } V \text{ es } D_i$$

es expresada como una relación difusa R_i la cual es interpretada como una intersección de los conjuntos difusos B_i y D_i, como se muestra a continuación:

$$R_i = B_i \cap D_i \tag{2.35}$$

donde R_i es definida sobre el espacio del producto Cartesiano $X \times Y$. La función de membresía de este este conjunto difuso R_i esta definida por:

$$\mu_{R_i}(x, y) = \mu_{B_i}(x) \wedge \mu_{D_i}(y) \tag{2.36}$$

Nótese que la relación difusa R_i forma una región rectangular en el espacio del producto Cartesiano $X \times Y$, como se ilustra en el figura 2.10.

Según el método de Mamdani, la agregación de las reglas(debido al conectivo lógico También), se lleva a cabo obteniendo la unión de las relaciones difusas individuales, esto es:

$$R = \cup_{i=1}^{m} R_i \tag{2.37}$$

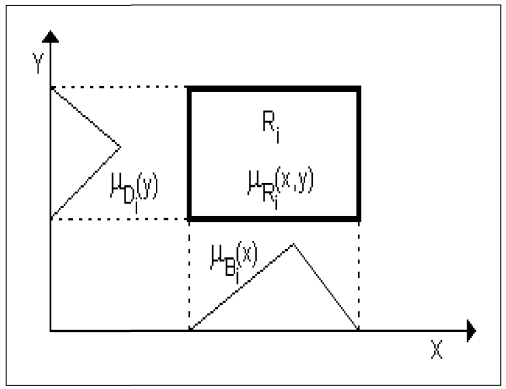

Figura 2.10: Relaciones Difusas

De este modo, la función de pertenencia del conjunto difuso R esta dado por la siguiente expresión:

$$\mu_R(x,y) = \vee_{i=1}^{m} \mu_{R_i}(x,y) = \vee_{i=1}^{m} (\mu_{B_i}(x) \wedge \mu_{D_i}(y)) \tag{2.38}$$

Así, dado un conjunto difuso A de entrada, la función de membresía del conjunto difuso de salida F, según la regla de inferencia *max-min*, es:

$$\begin{aligned} \mu_F(y) &= \vee_x [\mu_A(x) \wedge \mu_R(x,y)] \\ &= \vee_x [\vee_{i=1}^{m} (\mu_A(x) \wedge \mu_{R_i}(x,y))] \\ &= \vee_{i=1}^{m} [\vee_x (\mu_A(x) \wedge \mu_{B_i}(x) \wedge \mu_{D_i}(y))] \\ &= \vee_{i=1}^{m} [(\vee_x [\mu_A(x) \wedge \mu_{B_i}(x)]) \wedge \mu_{D_i}(y)] \\ &= \vee_{i=1}^{m} [\tau_i \wedge \mu_{D_i}(y)] \end{aligned} \tag{2.39}$$

donde:

$$\tau_i = \vee_x [\mu_A(x) \wedge \mu_{B_i}(x)] \tag{2.40}$$

denota la posibilidad condicional de B_i dado A. El valor de τ_i representa el *grado de disparo* o la *fuerza del disparo* de la i-esima regla y es una medida de la *relevancia* de la misma.

El procedimiento descrito para obtener el conjunto difuso de salida inferido por un modelo linguístico SISO, usando el método de Mamdani, a partir de un conjunto difuso de entrada, se resume en el siguiente algoritmo:

1. Para cada regla del modelo difuso:

   (a) Calcule el grado de disparo τ_i de la i-ésima regla, esto es:

$$\tau_i = \vee_x [\mu_A(x) \wedge \mu_{B_i}(x)] \tag{2.41}$$

   si la entrada es un conjunto difuso A.

   (b) Encuentre el conjunto difuso F_i, dado como salida por la i-ésima regla, según la siguiente equación:

$$F_i(y) = \tau_i \wedge \mu_{D_i}(y) \tag{2.42}$$

2. Obtenga la agregación F(y) de cada salida F_i, esto es:

$$F(y) = \vee_{i=1}^{m} F_i(y) \tag{2.43}$$

En la figura 2.11 se ilustra la aplicación de este algoritmo.

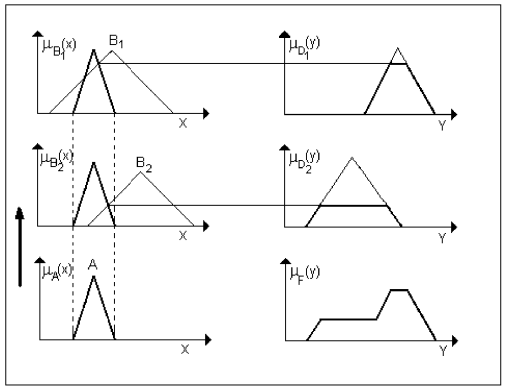

Figura 2.11: Mecanismo de Razonamiento tipo Mamdani

Geométricamente, el conjunto de reglas que conforman el modelo linguístico conforman una región difusa sobre el producto cartesiano $X \times Y$. En el método constructivo propuesto por Mamdani, esta relación R del modelo es obtenida a partir de la unión de todas las relaciones difusas R_i dada por cada regla que conforman el modelo, como se ilustra en la figura 2.12.

Otros métodos de razonamiento asigna distribuciones de posibilidad alternativos a la relación difusa R. Por ejemplo, el método de inferencia basado en razonamiento de tipo lógico, obtiene la salida considerando como conjuntos difusos del antecedente a aquellos dados por los complementos de los conjuntos difusos originales, además de considerar a las relaciones difusas asociadas a cada regla como una unión difusa entre

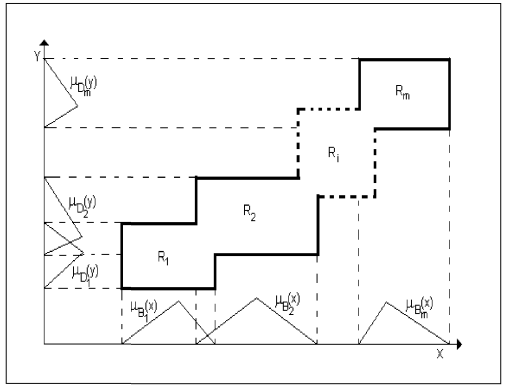

Figura 2.12: Unión de Relaciones Difusas

el conjunto difuso de entrada y salida (a diferencia del método constructivo, cada relación difusa se trata como una intersección difusa). Esto es:

$$R_i = B_i \cup D_i \tag{2.44}$$

Si a partir de esta relación así definida aplicamos el razonamiento aproximado sobre el cual se basa en forma general cualquier mecanismo de inferencia, es fácil obtener que el conjunto de salida inferida por el modelo, esta determinado por:

$$F(y) = \wedge_{i=1}^{m} [\tau_i \vee \mu_{D_i}(y)] \tag{2.45}$$

donde:

$$\tau_i = \vee_x [\mu_A(x) \wedge \mu_{B_i}(x)] \tag{2.46}$$

Mas detalles sobre este procedimiento pueden ser revisados en [3]. A continuación, sólo trataremos el método constructivo propuesto por Mamdani para desarrollar los procedimientos de inferencia para modelos MISO y MIMO, los cuales no son mas que una extensión del caso SISO utilizado para ilustrar el dicho mecanismo de inferencia.

Mecanismo de Razonamiento Tipo Mamdani Aplicado a Modelos MISO

En los modelos MISO dados por X, el operador "Y"en los antecedentes de las reglas se interpreta como una intersección difusa. Así, la relación difusa R_i asociada a cada regla no es mas que una intersección difusa entre los conjuntos difusos del antecedente y del consecuente. Esto es:

$$R_i = B_{i1} \cap ... \cap B_{ir} \cap D_i \tag{2.47}$$

donde R_i es definida sobre el espacio del producto Cartesiano $X_1 \times ... \times X_r \times Y$. La función de membresía de este conjunto difuso R_i esta definida por:

$$\mu_{R_i}(x_1, ..., x_r, y) = \mu_{B_{i1}}(x_1) \wedge \mu_{B_{ir}}(x_r) \wedge \mu_{D_i}(y) \tag{2.48}$$

La agregación de las relaciones difusas R_i, $\forall i = 1, ..., m$, se obtiene a partir de la unión difusa de las relaciones difusas individuales, esto es:

$$R = \cup_{i=1}^{m} R_i = \cup_{i=1}^{m} (B_{i1} \cap ... \cap B_{ir} \cap D_i) \tag{2.49}$$

Según esto, la función de membresía de la relación difusa R esta dado por:

$$\begin{aligned} \mu_R(x_1, ..., x_r, y) &= \vee_{i=1}^{m} \mu_{R_i}(x_1, ..., x_r, y) \\ &= \vee_{i=1}^{m} \mu_{B_{i1}}(x_1) \wedge \mu_{B_{ir}}(x_r) \wedge \mu_{D_i}(y) \end{aligned} \tag{2.50}$$

Ahora bien, dados los conjuntos difusos de entrada $A_1, ..., A_r$ asociados a cada variable difusa de la entrada U_i, el conjunto difuso de la salida F, inferido por el modelo, es obtenido a partir de una extensión del procedimiento desarrollado para el caso SISO, exhibe la función de membresía:

$$F(y) = \vee_{i=1}^{m} [\tau_i \wedge \mu_{D_i}(y)] \tag{2.51}$$

donde τ_i denota el *grado de disparo de la i-esima regla*, y viene dado por:

$$\tau_i = (\vee_{x_i} [\mu_{A_1}(x_1) \wedge \mu_{B_{i1}}(x_1)]) \wedge ... \wedge (\vee_{x_i} [\mu_{A_r}(x_r) \wedge \mu_{B_{ir}}(x_r)]) \tag{2.52}$$

Un algoritmo similar al dado en el caso SISO, resume el mecanismo de inferencia para modelos MISO, esto es:

1. Para cada regla del modelo difuso, y dado un conjuntos de valores difusos para las variables de entrada:
   (a) Calcule el grado de disparo τ_i de la i-ésima regla esto es:
   $$\tau_i = (\vee_{x_i} [\mu_{A_1}(x_1) \wedge \mu_{B_{i1}}(x_1)]) \wedge ... \wedge (\vee_{x_i} [\mu_{A_r}(x_r) \wedge \mu_{B_{ir}}(x_r)]) \tag{2.53}$$
   si las entradas son conjuntos difusos $A_1, ..., A_r$.
   (b) Encuentre el conjunto difuso F_i, dado como salida por la i-ésima regla, según la siguiente ecuación:
   $$F_i(y) = \tau_i \wedge \mu_{D_i}(y) \tag{2.54}$$
2. Obtenga la agregación F(y) de cada salida F_i, esto es:
   $$F(y) = \vee_{i=1}^{m} F_i(y) \tag{2.55}$$

Mecanismo de Razonamiento Tipo Mamdani Aplicado a Modelos MIMO

Consideremos ahora el caso general dado por el modelo MIMO de la expresión 2.30. Si las variable de salida $V_1, ...V_s$ son independientes, el modelo MIMO puede ser descompuesto en una colección de s modelos MISO, de la siguiente forma:

Si U_1 es B_{i1} y U_2 es B_{i2} y... y U_r es B_{ir} entonces V_t es D_{it},
$\forall t = 1, ..., s$

cada uno consistiendo de m reglas de la forma:

Si U_1 es B_{i1} y U_2 es B_{i2} y... y U_r es B_{ir} entonces V_t es D_{it},
$\forall i = 1, ..., m$

De lo anterior se observa que se tienen s subsistemas MISO para cada variable de salida. De los resultados obtenidos para el caso MISO, los s conjuntos de salida $F^t(y_t)$ inferidos por cada subsistema MISO, dados los conjuntos difusos de entrada $A_1, ..., A_r$, tienen una función de membresía dada por la siguiente ecuación:

$$F^t(y_t) = \vee_{i=1}^{m} \left[\tau_i \wedge \mu_{D_{it}}(y_t)\right], \forall\, t = 1, ..., s \tag{2.56}$$

donde τ_i es el *grado de disparo* de la i-esima regla del t-esimo subsistema.

Claramente el mismo conjunto de reglas asociada con una salida particular exhibe el mismo *grado de disparo*, pudiéndose calcular dichos grados de disparos en una forma conjunta o paralela.

Si consideramos el caso alternativo para los modelos MIMO, dado por la estructura genérica 2.31, la relación difusa R asociada al conjunto de instancias del modelo difuso, esta definida sobre el espacio del producto Cartesiano $X_1, ..., X_r, Y_1, ..., Y_s$, y viene dada por:

$$R = \cup_{i=1}^{m} R_i = \cup_{i=1}^{m} B_{i1} \cap ... \cap B_{ir} \cap D_{i1} \cap ... \cap D_{is} \tag{2.57}$$

La función de membresía del conjunto difuso R es:

$$\begin{aligned} \mu_R(x_1, ..., x_r, y_1, ..., y_s) &= \vee_{i=1}^{m} \mu_{R_i}(x_1, ..., x_r, y_1, ..., y_s) \\ &= \vee_{i=1}^{m} \mu_{B_{i1}}(x_1) \wedge ... \wedge \mu_{B_{ir}}(x_r) \\ \wedge \mu_{D_{i1}}(y_1) \wedge ... \wedge \mu D_{is}(y_s) & \end{aligned} \tag{2.58}$$

Luego, para un conjunto de conjuntos difusos de entrada $U_1 = A_1, ..., U_r = A_r$, este modelo linguístico produce un conjunto de salida F en el espacio del producto Cartesiano $Y = Y_1 \times, ..., \times Y_s$. Este conjunto difuso F está definido por la proyección de la intersección difusa $R \cap A_1 \cap ... \cap A_r$ sobre el espacio de salida Y, y sus función de membresía viene dado por:

$$\mu_F(y) = \vee_x \left[\mu_R(x_1, ..., x_r, y_1, ..., y_s) \wedge \mu_{A_1}(x_1) \wedge ... \wedge \mu_{A_r}(x_r)\right] \tag{2.59}$$

La obtención de cada una de las salidas individuales $F^1, ...F^s$, el conjunto difuso F debe ser proyectado sobre los universos de discursos asociados a cada salida particular, esto es:

$$\mu_{F^i}(y_i) = \vee_{y_1, ...y_s} \mu_F(y_1, ..., y_s), \forall s \neq i \tag{2.60}$$

El cálculo de los conjuntos difusos de salida no es una tarea sencilla, además de requerir cierto esfuerzo computacional. En el trabajo desarrollado por Gupta y sus colaboradores [10], se encuentra que una aproximación bastante razonable para el cálculo de los conjuntos de salida F^i, es la dada por la expresión 2.56

2.4.3 Modelos Difusos Takagi-Sugeno-Kang (TSK)

Este tipo de modelos, propuesto por Sugeno y sus colaboradores [5, 6], permite incorporar un conocimiento objetivo sobre el sistema bajo estudio, en vez de enmarcar dicho conocimiento dentro de conjuntos difusos, como lo hacen los modelos linguísticos revisados en la sección anterior. El método de razonamiento de TSK está asociado con una base de reglas en un formato particular, dónde los consecuentes no son expresados de forma difusa, sino de forma funcional, como muestra la siguiente expresión:

Si U_1 es B_{11} y U_2 es B_{12} y... y U_r es B_{1r}
entonces $Y_1 = b_{10} + b_{11}u_1 +, ..., +b_{1r}u_r$
También
Si U_1 es B_{21} y U_2 es B_{22} y... y U_r es B_{2r}
entonces $Y_2 = b_{20} + b_{21}u_1 +, ..., +b_{2r}u_r$
También
$\vdots$
También
Si U_1 es B_{m1} y U_2 es B_{m2} y... y U_r es B_{mr}
entonces $Y_m = b_{m0} + b_{m1}u_1 +, ..., +b_{mr}u_r$

donde $B_{ij},\ j = 1, ..., r,\ i = 1, ..., m$ son etiquetas linguísticas asociadas a los conjuntos difusos definidos sobre los espacios de entrada $X_1, X_2, ..., X_r$ de un sistema MISO, y $U_1, U_2, ..., U_r$ son los valores de las variables de entrada. Nótese que las funciones que aparecen en los consecuentes son lineales, con salidas puntuales y_i, entradas puntuales u_j y parámetros $b_{ij},\ j = 0, ..., r$.

La ventaja de este tipo de modelos reside en su capacidad para descomponer fenómenos complejos en subsistemas simples. Si para un fenómeno complejo es posible descomponer regiones disjuntas sobre el espacio de estado, pudiéndo luego identificar las dinámicas del sistema sobre esas regiones a través de modelos lineales o no lineales, cada uno con su propia estructura y sus propios parámetros, la caracterización de este sistema resulta en una colección de subsistemas combinados sobre la base de una "conmutación" lógica.

Ahora bien, en la realidad esa partición disjunta no es siempre posible, bien sea por la no existencia de fronteras entre estas regiones o por el conocimiento parcial sobre

el comportamiento de estos sistemas. Luego, el modelo TSK permite representar una partición difusa de las regiones sobre el espacio de estado del sistema, y reemplazar la conmutación lógica por un mecanismo de razonamiento interpolativo. De esta manera, la dinámica del sistema en cualquier punto del espacio de estado puede verse como la combinación de las dinámicas de una o mas regiones a donde dicho puntos pertenece. Este tipo de modelo puede resultar bastante útil cuando cuando las regiones asociadas a diferentes condiciones de operación pueden ser descritas a través de etiquetas linguísticas.

Bajo el método de razonamiento propuesto por TSK, la salida puntual y inferida por este modelo híbrido esta definida por el promedio de las salidas individuales y_i de cada regla dado un conjunto de valores puntuales de las variables de entrada U_j. Esto es:

$$y = \sum_{i=1}^{m} \frac{\tau_i}{\sum_{j=1}^{m} \tau_j} y_i = \sum_{i=1}^{m} \frac{\tau_i}{\sum_{j=1}^{m} \tau_j} (b_{i0} + b_{i1}u_1 +, ..., +b_{ir}u_r) \tag{2.61}$$

donde τ_i denota el *grado de disparo de la i-ésima regla*, dado por:

$$\tau_i = \mu_{B_i\ 1}(u_1) \wedge ... \wedge \mu_{B_i\ r}(u_r) \tag{2.62}$$

Nótese, que siendo $u_j,\ j = 1, ..., r$ valores puntuales de las variables de entrada, $\mu_{B_{ij}}(u_j)$ denota el grado de membresía del valor u_j, al conjunto difuso B_{ij}.

Para el caso general, las funciones de los consecuentes de las reglas, pueden venir dadas por funciones no lineales de las variables de entrada, esto es $y_i = f(u_1, ..., u_r)$. En este caso, la expresión 2.61 también aplica para calcular la salida y.

2.5 Los Mecanismos de Difusificación y Desdifusificación

En algunos casos, los valores de las variables de entrada a los modelos difusos provienen de procesos que generan valores puntuales o "crisp"(como se le considera en inglés). Las aplicaciones en el área de control de procesos son ejemplos clásicos de sistemas con esta característica. En estos casos, es necesario someter a dichos valores puntuales a un *proceso de difusificación*, con el fin de darle una conotación difusa a dichos valores. Análogamente, en algunos casos puede resultar de inutilidad práctica la utilización de los valores difusos que generan como salida los modelos linguísticos, y es necesario someter a dichos valores difusos a un *proceso de desdifusificación*, con el fin de generar valores puntuales de dichas variables de salida. Estos procesos mencionados se ejecutarían antes y después, respectivamente, del mecanismo de inferencia difusa revisado en la sección anterior, como muestra la figura 2.13.

2.5.1 Mecanismo de Difusificación

Dado un valor puntual $x^* \in X$, el mecanismo de difusifucación considerado consiste en crear un conjunto difuso A cuya función de pertenencia es entendida como aquella

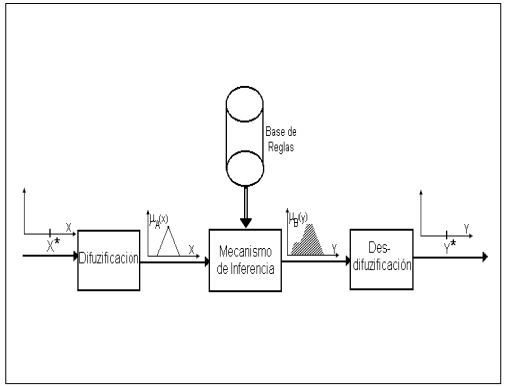

Figura 2.13: Módulos de Difusificación y Desdifusificación

donde para cualquier $x \in X$ su valor es cero, excepto en el valor x^*, donde toma el valor de 1. Esto es:

$$\mu_A(x) = \begin{cases} 0 & si \quad x \neq x^* \\ 1 & si \quad x = x^* \end{cases} \tag{2.63}$$

Nótese que este conjunto difuso así considerado no es mas que un conjunto ordinario con un único elemento dado por x^* (Difusificación Sencilla).

Esta manera de difusificar el valor de entrada al modelo difuso, simplifica el cálculo del *grado de disparo* τ_i de cada regla del modelo difuso. Para el caso de modelo SISO dados por el conjunto de reglas 2.28, cada τ_i viene dado por:

$$\tau_i = \mu_{B_i}(x^*) \tag{2.64}$$

Claramente, τ_i no es mas que el grado de membresía del valor puntual x^* de la variable X, al conjunto difuso B_i asociado a dicha variable.

De esta manera, el algoritmo de inferencia para modelos SISO mostrado en la sección anterior, dados valores puntuales de las variables de entrada, se redefine como:

Algoritmo de inferencia con difusificación.

1. Para cada regla del modelo difuso:

   (a) Calcule el grado de disparo τ_i de la i-esima regla, esto es:

$$\tau_i = \mu_{B_i}(x^*) \tag{2.65}$$

(b) Encuentre el conjunto difuso F_i, dado como salida por la i-esima regla, según la siguiente ecuación:

$$F_i(y) = \tau_i \wedge \mu_{D_i}(y) \tag{2.66}$$

2. Obtenga la agregación F(y) de cada salida F_i, esto es:

$$F(y) = \vee_{i=1}^{m} F_i(y) \tag{2.67}$$

En el caso de los modelos MISO, dados por la estructura 2.29, si se tienen valores puntuales de entrada $x_1^*, ..., x_r^*$, asociados a cada variable de entrada U_i, en virtud de lo condiderado en el caso SISO, es fácil deducir que el *grado de disparo* τ_i para cada regla del modelo es definido por la siguiente ecuación:

$$\tau_i = \mu_{B_{i1}}(x_1^*) \wedge ... \wedge \mu_{B_{ir}}(x_r^*) \tag{2.68}$$

La expresión anterior indica que τ_i, no es mas que el mínimo valor entre los grados de membresía de cada valor puntual x_j de la variable X_j, al conjunto difuso B_{ij} asociado a dicha varible, para todo $j = 1, ..., r$.

Así, el algoritmo de inferencia dado en la sección anterior, considerando ahora valores puntuales de las variable de entrada, es:

1. Para cada regla del modelo difuso

   (a) Calcule el grado de disparo τ_i de la i-ésima regla esto es:

   $$\tau_i = \mu_{B_{i1}}(x_1^*) \wedge ... \wedge \mu_{B_{ir}}(x_r^*). \tag{2.69}$$

   (b) Encuentre el conjunto difuso F_i, dado como salida por la i-ésima regla:

   $$F_i(y) = \tau_i \wedge \mu_{D_i}(y) \tag{2.70}$$

2. Obtenga la agregación $F(y)$ de cada salida F_i, esto es:

$$F(y) = \vee_{i=1}^{m} F_i(y) \tag{2.71}$$

En la figura 2.14 se ilustra el algoritmo de inferencia con difusificación.

Para el caso de los modelos MIMO, siendo que son considerados como un conjunto de subsistemas MISO, el cálculo de τ_i para la i-ésima regla del cada subsistema es exactamente igual al dado en la ecuación 2.68.

En el modelo propuesto por Takagi-Sugeno-Kang en la sección anterior, el mecanismo de inferencia propone el grado de disparo de cada regla híbrida dado por la ecuación 2.62, el cual proviene de considerar la difusificación del valor puntual $u_j, j = 1, ..., r$, por el mecanismo de difusificación sencilla.

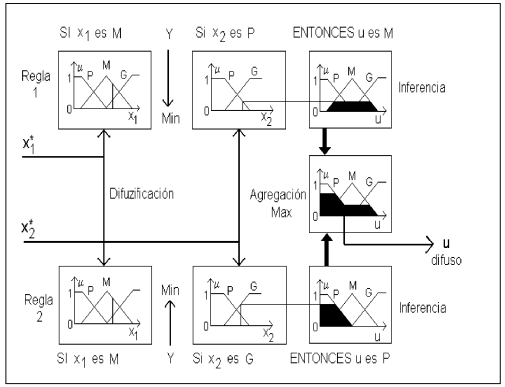

Figura 2.14: Algoritmo de Inferencia con Difusificación

2.5.2 Mecanismo de Desdifusificación

Ya hemos visto que las salidas obtenidas de los modelos linguísticos son conjuntos difusos definidos sobre un conjunto de soporte Y. Así, para un conjunto difuso de salida F definido sobre Y, cada $F(y_i) = w_i$, para todo $y_i \in Y$, es una medida de cuán bueno es el valor y_i como valor de salida proveniente del modelo difuso, para la entrada en particular. El mecanismo de difusificación usa la información sobre F, es decir su función de membresía, para determinar cuál es el mejor valor y^* para ser considerado como salida.

Este proceso de selección del mejor valor puntual representativo de la salida difusa, tiene una naturaleza probabilística. Consideremos una variable V definida sobre un universo de discurso $X = \{x_1, x_2, ..., x_n\}$, y sea $A = \{1/x_1, 1/x_2, ..., 1/x_m, 0/x_{m+1}, ..., 0/x_n\}$ un conjunto difuso de X, donde $m \leq n$. Claramente, cualquier x_i, para $1 \leq i \leq m$ podría ser un elemento representativo de X, entonces el problema es decidir cual x_i debería ser seleccionado.

Existe una La variedad de métodos para obtener el valor puntual mas representativo de un conjunto difuso. A continuación, presentamos un método de desdifusificación básico, el cual se encuentra parametrizado por un cierto valor α, pudiendo generar, entonces, infinitas expresiones para calcular este valor representativo X^*.

Metodo de desdifusificación BADD

Considere un conjunto difuso A sobre un universo de discurso discreto $X = \{x_1, ..., x_n\}$, con función de membresía $\mu_A(x_i) = w_i$ para $i = 1, ..., n$. La tranformación para selec-

cionar un valor x^* representativo de un conjunto difuso es:

$$x^* = \frac{\sum_{i=1}^{n} w_i^{\alpha} x_i}{\sum_{i=1}^{n} w_i^{\alpha}} \tag{2.72}$$

Este método es llamado el *Método de Desdifusificación Basado en la Distribución de Desdifusificación Básica* (Método BADD, por sus siglas en inglés) . Si en 2.72, se toma $\alpha = 1$ (asociado a una confianza *normal* sobre el conjunto difuso de salida obtenido del modelo), entonces el método de desdifusificación recibe el nombre de *Método del Centro de Area ó Centroide*. Si se toma $\alpha \to \infty$ (asociado a una alta confianza en el conjunto difuso obtenido por el modelo), entonces el método de desdifusificación recibe el nombre de *Método de la Media del Máximo*.

Ambos métodos no son mas que casos particulares del método básico general y son los de uso más comúm. El valor de x^* obtenido para $\alpha = 0$ en el método básico, es el valor más alto sobre el universo X para el conjunto A, mientras que el Método de la Media del Máximo ofrece el valor más bajo que se pueda obtener a través del método básico. Nótese además que el valor desdifusificado para $\alpha = 0$. no es mas que la media aritmética de los valores pertenecientes al universo X.

La parametrización de este método de desdifusificación a través de α nos provee de un potencial aprendizaje sobre cual es el mejor método de desdifusificación para el problema bajo estudio en particular, sin embargo, puede señalarse como desventaja de este métddo, que el cálculo de x^* para $\alpha \neq 1$ puede ser complicado. Por otro lado, la dependencia no-lineal entre el valor de x^* y el parámetro α dificulta el aprendizaje del valor apropiado de dicho parámetro.

Existen otros métodos de desdifusificación que superan las deventajas mencionadas para el método básico [3], sin embargo estos no serán estudiados en este manuscrito.

2.5.3 Un Ejemplo de Aplicación de Sistemas de Inferencia Difusos.

A continuación, mostraremos un ejemplo donde se ilustra el mecanismo de inferencia de un sistema difuso con difusificación y desdifusificación.

Una de las primeras aplicaciones industriales que tuvo la lógica difusa, fue el modelado de controladores a partir de la información dada por los expertos. Consideremos, entonces, el controlador difuso difuso tipo Mamdani, cuyo comportamiento es análogo a un controlador Proporcional-Integral clásico. Las variables de entrada al modelo difuso que describe al controlador son el *error en el instante de tiempo k (e(k))* y *el cambio del error en el instante de tiempo k* ($\triangle e(k) = e(k) - e(k-1)$). La variable de salida del modelo difuso es *el cambio del control en el instante tiempo k* ($\triangle u(k) = u(k) - u(k-1)$)

Las variables difusa $e(k)$, $\triangle e(k)$ y $\triangle u(k)$ tienen asociadas tres valores difusos con las etiquetas linguísticas *Negativo, Cero y Positivo*. Las funciones de membresía que definen a estos conjuntos difusos se muestran en la figura 2.15.

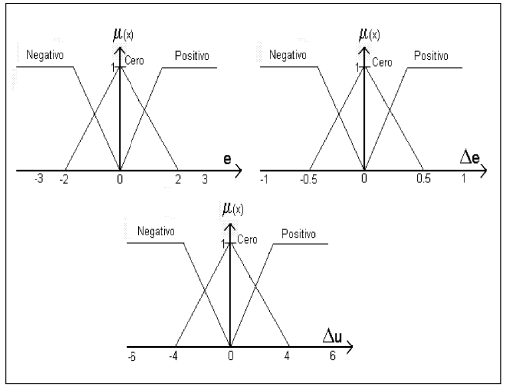

Figura 2.15: Funciones de Membresía de los Conjuntos Difusos $e(k), \triangle e(k), \triangle u(k)$

Con el fin de aplicar posteriormente el mecanismo de desdifusificación discreta propuesto en la ecuación 2.72, hacemos una discretización equidistante del universo de discurso de la variable $\triangle u(k)$, mostrada a continuación:

$$\triangle u(k) = [-6, -4.5, -3, -1.5, 0, 1.5, 3, 4.5, 6]$$

De esta manera, obtenemos las funciones de membresía discretas de cada conjunto difuso de la variable de salida, de la siguiente forma:

$\mu_N(\triangle u(k)) = [1, 1, 1, 0.5, 0, 0, 0, 0, 0]$
$\mu_C(\triangle u(k)) = [0, 0, 0, 0.5, 1, 0.5, 0, 0, 0]$
$\mu_P(\triangle u(k)) = [0, 0, 0, 0, 0, 0.5, 1, 1, 1]$

La base de reglas que conforman al modelo difuso del controlador se enumeran a continuación:

1. Si e(k) es Negativo y $\triangle$e(k) es Negativo entonces $\triangle$u(k) es Negativo.
2. Si e(k) es Negativo y $\triangle$e(k) es Zero entonces $\triangle$u(k) es Negativo.
3. Si e(k) es Negativo y $\triangle$e(k) es Positivo entonces $\triangle$u(k) es Zero.
4. Si e(k) es Zero y $\triangle$e(k) es Negativo entonces $\triangle$u(k) es Negativo.
5. Si e(k) es Zero y $\triangle$e(k) es Zero entonces $\triangle$u(k) es Zero.
6. Si e(k) es Zero y $\triangle$e(k) es Positivo entonces $\triangle$u(k) es Positivo.

7. Si e(k) es Positivo y $\triangle$e(k) es Negativo entonces $\triangle$u(k) es Zero.
8. Si e(k) es Positivo y $\triangle$e(k) es Zero entonces $\triangle$u(k) es Positivo.
9. Si e(k) es Positivo y $\triangle$e(k) es Positivo entonces $\triangle$u(k) es Positivo.

Conocida la caracterización difusa del controlador, obtenemos el valor de salida $\triangle$u(k) a partir de un par de valores de e(k) y $\triangle$e(k), aplicando el mecanismo de inferencia difusa con difusificación y aplicándole posteriormente al conjunto difuso de salida obtenido un método de desdifusificación.

Consideremos el par de valores puntuales e(k)=-2.1 y $\triangle$e(k)=0.5. El primer paso del algoritmos de inferencia es calcular de *grado de disparo* de cada regla, según la ecuación 2.65. De esta manera, los grados de disparo para cada regla son:

1. $\tau_1 = 0$
2. $\tau_2 = 0$
3. $\tau_3 = 1$
4. $\tau_4 = 0$
5. $\tau_5 = 0$
6. $\tau_6 = 0$
7. $\tau_7 = 0$
8. $\tau_8 = 0$
9. $\tau_9 = 0$

Nótese que para este par de valores sólo es activada la regla 3, con un grado de disparo de 1. El siguiente paso es encontrar el conjunto difuso de salida inferido por cada regla, según la ecuación 2.66. Los grados de membresía de dichos conjuntos difusos de salida son:

1. $\mu_{F_1}(\triangle u(k)) = [0,0,0,0,0,0,0,0,0]$
2. $\mu_{F_2}(\triangle u(k)) = [0,0,0,0,0,0,0,0,0]$
3. $\mu_{F_3}(\triangle u(k)) = [0,0,0,0,5,1,0.5,0,0,0]$
4. $\mu_{F_4}(\triangle u(k)) = [0,0,0,0,0,0,0,0,0]$
5. $\mu_{F_5}(\triangle u(k)) = [0,0,0,0,0,0,0,0,0]$
6. $\mu_{F_6}(\triangle u(k)) = [0,0,0,0,0,0,0,0,0]$

7. $\mu_{F_7}(\triangle u(k)) = [0,0,0,0,0,0,0,0,0]$
8. $\mu_{F_8}(\triangle u(k)) = [0,0,0,0,0,0,0,0,0]$
9. $\mu_{F_9}(\triangle u(k)) = [0,0,0,0,0,0,0,0,0]$

Del resultado anterior se observa que, efectivamente, la regla 3 es la única que genera un conjunto difuso de salida. El último paso de algoritmo de inferencia es obtener la unión de todos los conjuntos de salida, según la ecuación 2.67. Evidentemente, siendo F_3 el único conjunto difuso con grados de membresía no nulos, el conjunto difuso de salida F propuesto por el modelo es F_3.

Para obtener el valor puntual de $\triangle u(k)$ aplicamos el método de desdifusificación del Centroide, tomado a $\alpha = 1$ en la ecuación 2.72. De esta manera:

$$\triangle u(k)^* \frac{=0.5*(-1.5)+1(0)+0.5*(1.5)}{0.5+1+0.5} = 0$$

En términos reales, este valor inferido por el controlador difuso significa que dados este par de valores para $e(k)$ y $\triangle e(k)$, no debe modificarse el actual valor del control $u(k)$ puesto que la variación $\triangle u(k)$ es nula (0).

Si aplicamos el mismo procedimiento para el par de valores $e(k) = -0.9$ y $\triangle e(k) = 0.2$, verificaremos que, en este caso, se activan las reglas 2,3,5 y 6, en consecuencia tendremos cuatro conjuntos difusos de salida ingeridos por cada una de estas reglas. La unión difusa de estos conjuntos genera un único conjunto difuso de salida cuyo valor puntual más representativo, obtenido a través de Método del Centroide, es $\triangle u(k) = -1.2348$

2.6 Aplicaciones de Lógica Difusa en Control de Procesos

El grado de vaguedad en incertidumbre en sistemas complejos, ha hecho de la Lógica Difusa una herramienta importante tanto para el tratamiento como para el procesamiento de la informacón contenida en dichos sistemas. La información cuantitativa y cualitativa presente en procesos complejos han servido de base para el desarrollo de sistemas basados en reglas para la solución de interrogantes planteadas respecto al comportamiento de estos procesos. Así, los conceptos de la lógica difusa han sido ampliamente utilizado en el desarrollo de sistemas expertos, programación matemática, procesamiento de imágenes y señales, modelado de interacciones hombre-máquina, sistemas para la toma de desiciones entre otras. En esta sección revisaremos, particularmente, las protenciales aplicaciones de la Lógica Díffusa en el área de control de procesos [18].

Un sistema de control, en el sentido más basico, es un proceso descrito por una cantidad de variables, algunas de ellas manipuladas y otras controladas, que contempla una serie de tareas generalmente separadas en niveles de jereraqía. En la figura

2.16 se muestra el esquema básico de un sistema de control multinivel En el nivel más bajo toma lugar tareas de medición y manipulación de señales; en el siguiente nivel, denominado nivel de control, se ejecutan la tareas de control local para ajustar las variables del proceso a sus valores de referencia; en el nivel de supervisión se ejecutan tareas de monitoreo de señales con el fín de indicar estados no deseados del proceso y tomar acciones; finalmente, en los niveles superiores se ejecutan tareas de optimización, coordinación y administración general de las tareas.

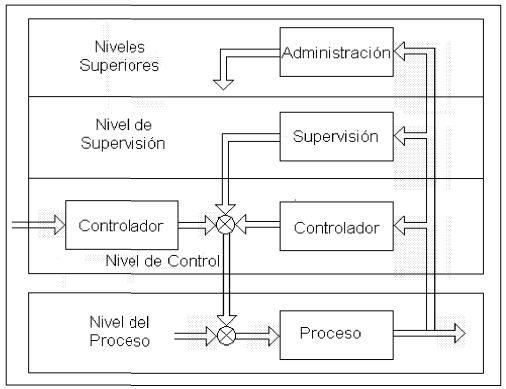

Figura 2.16: Sistema de Control Multinivel

La incertidumbre presente en las variables manejadas en cada nivel así, así como en la descripción del proceso, ha dado lugar al desarrollo de sistemas lógico difusos orientados a alcanzar la funciones de cada nivel del proceso de control. Inspirados en los trabajos de L. Zadeh, Mandani y Assilian propusieron controladores difusos que describen el comportamiento de un operador humano en una forma linguística [19]. Basicamente, la estructura de un controlador difuso es un modelo basado en reglas SI-ENTONCES donde las variables de entrada son el error y la variación del error de la variable controlada respecto a su valor de referencia, y la salida es el valor de la señal de control que debe ser aplicada, como se muestra en la figura 2.17.

Sin embargo, la lógica difusa no solo es aplicada para el diseño de controladores linguísticos, sino que pueden diseñarse reglas para el entonamiento de los parámetros de controladores clásicos, tales como PID, basados en el conocimiento del proceso. La utilización de la lógica disfusa en estos términos podría suponer un esquema adaptativo híbrido para la adaptación en línea de los parametros de controladores difusos o clásicos, tomando en cuenta la presencia de perturbaciones en el proceso, según muestra la figura 2.18. Otra aplicación híbrida es el uso de controladores difusos de respaldo que se incorporan al sistema de control cuando, en presencia de condiciones

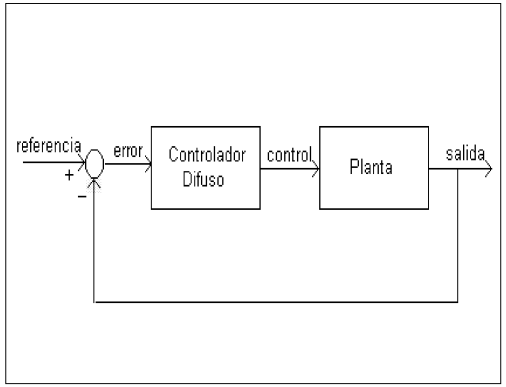

Figura 2.17: Controladores Difusos

anormales, el controlador clásico no tiene buen desempeño. En este caso el controlador difuso puede manipular una señal controlada adicional y otra señal de control, como muestra la figura 2.19

A pesar de que las aplicaciones de controladores difusos a nivel local están aún bajo investigación, se pueden establecer algunos lineamientos para ayudar a la decisión sobre su uso en los niveles de control directo:

- En procesos donde no existe una descripción o conocimiento de su compotamiento que permita la utilización de esquemas clásicos de control. Hay que recordar que en varios caso, el conocimiento de un modelo rudo del proceso permite la aplicación, con éxito, de técnicas clásicas de control. En procesos complejos, el sólo conocimiento qualitativo del comportamiento de procesos hace de los controladores difusos una buena alternativa para alcanzar los objetivos deseados.

- La operación manual de cualquier procesos complejo puede ser traducida a reglas difusas que contengan la información manejada por el operador experto.

Algunas de las desventajas sobre el uso de controladores difusos apuntan hacia el entonamiento de sus parámetros, que puede resultar en una tarea engorrosa y de largo tiempo; por otro lado, aspectos esenciales en la ingeniería de control como el estudio de la **estabilidad** del proceso no es una tarea trivial cuando se usan controladores difusos debido a la diversidad de estructuras que puede tener dicho controlador. Algunos esfuerzos en esta dirección han sido desarrollados en [17].

En los niveles de supervisión, la lógica difusa puede aplicarse para el desarrollo de detectores de fallas basados en información cualitativa del proceso. Ciertos valores característicos de las variables de proceso, tomados de las observaciones de los

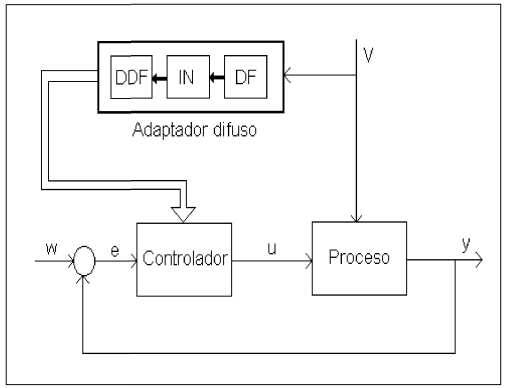

Figura 2.18: Esquema de Control Híbrido Adaptativo

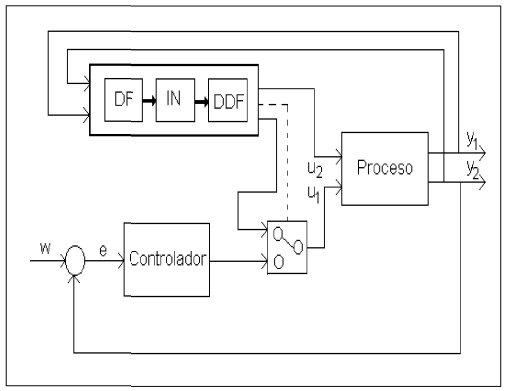

Figura 2.19: Controlador Difuso de Respaldo

operadores, tales como ruidos especiales, colores, vibraciones, etc., pueden producir información que almacenada en modelos linguísticos indiquen la existencia de fallas en el proceso. Otra aplicación es la interpretación basada en reglas difusas de los valores de ciertos parámetros indicativos de fallas obtenidos con procedimientos clásicos basados en modelos. De esta forma, se establecen *umbrales difusos* respecto a los cuales los valores de estos parámetros indicativos apuntan hacia la existencia de fallas, como se muestra en la figura 2.20.

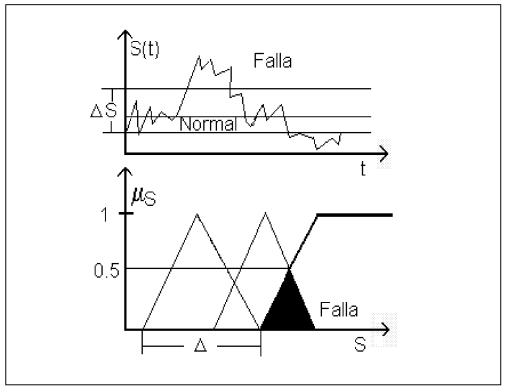

Figura 2.20: Esquema de Detección de Fallas Difuso

El diagnóstico de una falla también puede lograrse con sistemas logico difusos donde el secuenciamiento de eventos disparados a partir de la información proveniente de un detector de fallas, puede colocarse en forma de reglas que apunten hacia un sistema experto de diagnóstico, basado en la información cualitativa sobre el comportamiento del proceso en presencia de una determinada falla.

En general, el uso de la Lógica Difusa en las tareas de control ofrece un amplio espectro de aplicaciones, con diferentes esquemas, que responden a las necesidades del procesos donde las características del mismo exija el uso de esta técnica. De hecho, la descripción misma del proceso puede obtenerse a partir de modelos basados en reglas difusas. Una característica particular de los sistemas de control, es que la vaguedad y la incertidumbre aumenta en la medida que ascendemos en los niveles funcionales, por lo tanto los usos potenciales de la lógica difusa van en aumento también, no solo como técnica aislada, sino en combinación con otras técnicas que incorporen a los modelos difusos capacidades de aprendizaje y adaptación a las variaciones del medio donde están inmersos.

Bibliografía

[1] *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, R. Yager and L. Zadeh, Eds., London: Kluwer Academic Publishers, 1992.

[2] L.A. Zadeh, "Fuzzy sets", *Information and Control,* vol. 8, pp. 338-353, 1965.

[3] R. Yager and D. Filev, "Essentials of Fuzzy Modeling and Control", N.Y.: John Wiley, 1994.

[4] L. Zadeh, "Outline of a new approach to the analisys of complex systems and decision processes", *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, pp. 28-44, 1973.

[5] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control", *IEEE Trans. Syst., Man, Cybern*, vol. 15, pp. 116-132, 1985.

[6] M. Sugeno and G.T. Kang, "Fuzzy modeling and control of multilayer incinerator", *Fuzzy Sets and Systems*, vol. 18, pp. 329-346, 1986.

[7] R.M. Tong, "Sinthesis of fuzzy models for industrial processes-Some recent results", *International Journal of General Systems*, vol. 4, pp. 143-163, 1978.

[8] W. Pedrycz, *Fuzzy Control and Fuzzy Systems*, New York: Jhon Wiley, 1989.

[9] A. Kandel, *Fuzzy Mathematical Techniques with Applicactions*, Reading, MA: Addison Wesley, 1986.

[10] M.M. Gupta, J.B. Kiszka and SG.J. Trojan, "Multivariable structure of fuzzy control system", *IEEE Trans. Syst., Man, Cibern.*, vol. SMC-16, pp. 638-656, 1986.

[11] G.J. Klir, "Probability-possibility conversion", in *Proc. 3rd. IFSA Congress*, Seattle, 1989, pp. 408-411.

[12] S. Moral, "Construction of a probability distribution from a fuzzy information", in *Fuzzy Set Theory and Applications*, A. Jones, A. Kauffman, H.J. Zimmermann, Eds., Reidel: Dordrecht, pp. 51-60, 1986.

[13] L.A. Zadeh, "Fuzzy sets as a basis for a theory of possibility", *Fuzzy Sets and Systems*, vol. 1, pp. 3-28, 1978.

[14] D. Dubois and H. Prade, *Possibility Theory: An Approach to Computerized Processing of Uncertainty*, New York: Plenun Press, 1988.

[15] D. Dubois and H. Prade, "A review of fuzzy sets agregation connectives", *Information Sciences*, vol. 36, pp. 85-121, 1985.

[16] R. Yager, "Connectives and quantifiers in fuzzy sets", *Fuzzy Sets and Systems* vol. 40, pp. 39-76, 1985.

[17] L. Wang, *Adaptive Fuzzy Systems and Control. Design and Stability*, New Jersey: Prentice Hall, 1994.

[18] R. Isermann, "On fuzzy logic applications for automatic control, supervision, and fault diagnosis", *IEEE Trans. Syst., Man, Cybern.-Part A: Systems and Humans*, vol. 28, no. 2, March 1998.

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

Sistemas Expertos

Wladimir Rodríguez Graterol
Departamento de Computación
Facultad de Ingeniería
Universidad de Los Andes
Mérida 5101, Venezuela
E-mail: wladimir@ula.ve

Francklin Rivas Echverría
Departamento de Sistemas de Control
Facultad de Ingeniería
Universidad de Los Andes
Mérida 5101, Venezuela
E-mail: rivas@ula.ve

Capítulo 3

Sistemas Expertos

3.1 Introducción

En este capítulo se dará una introducción al campo de la Ingeniería del Conocimiento, empezando por una breve historia de la Inteligencia Artificial y de los Sistemas Basados en el Conocimiento; posteriormente se definirán lo que son los Sistemas Basados en el Conocimiento y los Sistemas Expertos. Finalizando con una descripción de lo que es la Ingeniería del Conocimiento, incluyendo sus componentes y aspectos metodológicos.

3.1.1 Historia de la Inteligencia Artificial

Los Sistemas Basados en el Conocimiento (Sistemas Expertos) son una de las aplicaciones de la Inteligencia Artificial. La Inteligencia Artificial es el área de la Informática que intenta lograr una utilización de los computadores en campos en los cuales el ser humano es superior. La Inteligencia Artificial se compone de un conjunto de técnicas que permiten a los computadores emular a la mente humana. Las investigaciones de la Inteligencia Artificial están orientadas a descubrir la forma de cómo hacer pensar y razonar a un computador como una persona. La idea de que un computador estaría en capacidad para razonar viene del hecho de ver a un computador como un procesador de símbolos, y que estos símbolos pueden ser números, texto o conceptos. Basado en lo anterior H. Simon presenta en 1981 la "hipótesis de los símbolos físicos"(Figura 3.1). Los principios básicos de esta son:

- Los pensamientos corresponden al lenguaje.
- El lenguaje puede ser representado con símbolos.
- Mediante la manipulación de símbolos en el computador se puede simular el proceso de pensar.

En la tabla que se muestra a continuación se da una visión general de los principales desarrollos en la Inteligencia Artificial.

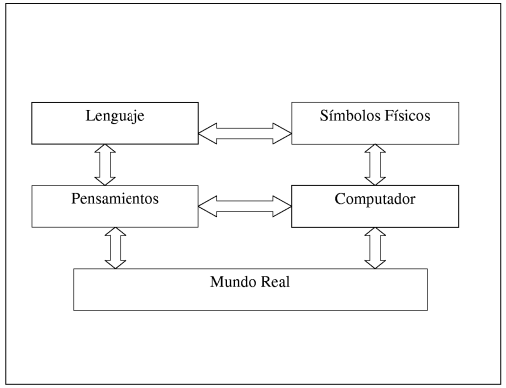

Figura 3.1: Hipótesis de los Símbolos Físicos

Período	Eventos Principales
Raíces	Lógica Formal Psicología Cognitiva
Pre - IA (1945 - 1954)	Desarrollo de los computadores H. Simon, "Administrative Behavior" N. Wiener, Cibernética M. Turing, "Computer Machinery and Intelligence" Conferencia sobre Cibernética en Macy
Inicio de la Investigación en IA (1955 - 1960)	Mayor disponibilidad de Computadores IPL-1 (Information Processing Language-1) Seminario de sobre IA en Dartmouth, verano de 1956 GPS (General Problem Solver) Psicología del procesamiento de información
A. Búsqueda de Solucionadores Generales de Problemas (1961 - 1970)	Newell y H. Simon, "Human Problem Solving" LISP Búsqueda de Heurísticas Robótica Programas de Ajedrez DENDRAL (Stanford)
El descubrimiento de los Sistemas Basados en el Conocimiento (1971 - 1980)	MYCIN (Stanford) HEARSAY II (Carnagie-Mellon) MACSYMA (MIT) Ingeniería del Conocimiento EMYCIN (Stanford) GUIDON (Stanford) PROLOG
	Hebert Simon, Premio Novel
Comercialización de la IA	PROSPECTOR (SRI) Proyecto Japones de la Quinta Generación Compañías de IA Desarrollo comercial de los Sistemas Basados en el conocimiento

Tabla 1. Desarrollo de la Inteligencia Artificial

La historia de los Sistemas Basados en el Conocimiento esta relacionada con los últimos tres períodos de la historia de la Inteligencia Artificial.

- Periodo de Iniciación (1961 - 1970)
- Período de Experimentación y Desarrollo (1971 - 1980)
- Período de Comercialización (1981 -)

A continuación se presentaran algunos ejemplos de Sistemas Basados en el Conocimiento que se desarrollaron en cada uno de estos períodos.

Período de Iniciación (1961 - 1970):

- DENDRAL: Muchos le consideran el primer sistema experto. Desarrollado en la Universidad de Stanford, DENDRAL analiza compuestos químicos que no están identificados. Después de interpretar los datos provenientes del espectrógrafo de masas y del espectrógrafo de resonancia magnética nuclear, DENDRAL realiza modelos topológicos de la estructura molecular de los compuestos.
- MACSYMA: resuelve problemas matemáticos tanto numéricos como simbólicos. Utiliza técnicas heurísticas de reconocimiento de patrones para resolver muchos tipos de problemas matemáticos, como son la diferenciación, integración, ecuaciones polinomiales y matrices.
- HEARSAY: interpreta en lenguaje natural un subconjunto del idioma.

Período de Experimentación y Desarrollo (1971 - 1980):

- MYCIN: Sistema Experto para ayudar en el diagnóstico y tratamiento de infecciones bacterianas de la sangre. Desarrollado en la Universidad de Stanford. Emplea reglas de producción y métodos de inferencia con encadenamiento regresivo. Introdujo el concepto de factor de certeza, permitiendo el razonamiento con incertidumbre. Un factor de certeza es un número que indica el grado de confianza que se puede tener en una determinada respuesta.
- PROSPECTOR: se desarrollo a principios de los años setenta en SRI International. Construido sobre la tecnología de MYCIN. Es uno de los sistemas expertos más estudiados debido a su gran éxito. Especialmente conocido desde que con su ayuda se descubrió un importante depósito de molibdeno, valorado en más de 100 millones de dólares, en el estado de Washington. Es un sistema experto diseñado para ayudar a los geólogos a encontrar yacimientos importantes. Dándole datos acerca de un área particular, PROSPECTOR puede estimar las probabilidades de encontrar distintos tipos de depósitos minerales..
- CASNET: Sistema Experto para el diagnóstico del glaucoma.

- INTERNIST: Sistema de medicina interna que puede diagnosticar hasta 500 enfermedades.

Período de Comercialización (1981 -):

- DRILLING ADVISOR: utilizado para diagnosticar problema en la perforación de pozos petroleros e interpretación de datos geológicos.
- XCON desarrollado en la Universidad de Carnegie-Mellon a finales de los años setenta y revisado en DEC a principios de los ochenta. Es un sistema experto diseñado para ayudar a los técnicos de Digital Equipment Corporation (DEC) a configurar sistemas de miniordenadores. XCON genera automáticamente la configuración deseada a partir de los requisitos del cliente seleccionando entre la amplia gama de la popular serie de los VAX de DEC.
- MOLGEN: ayuda a planificar experimentos en genética molecular.
- DELTA: diseñado para ayudar al personal de mantenimiento de trenes a mantener las locomotoras diesel-eléctricas de General Electric.

3.1.2 Areas de la Inteligencia Artificial

Una de las metas de la Inteligencia Artificial ha sido la replicación de las funcionalidades del cerebro humano, por lo que han surgido diferentes subcampos de la IA orientados a copiar las diferentes facultades del ser humano. A continuación se describirán algunos de estos subcampos:

- **Visión Artificial**: este campo esta orientado a lograr la construcción de sistemas de reconocimiento de patrones que funcionen como lo hace el sistema de visión del ser humano.
- **Robótica**: campo orientado a la producción de dispositivos mecánicos capaces de movimiento controlado.
- **Procesamiento del Habla**: campo que persigue el reconocimiento y síntesis del lenguaje hablado.
- **Procesamiento de Lenguaje Natural**: campo que estudia la compresión y producción de lenguaje natural escrito.
- **Prueba de Teoremas**: campo que intenta la prueba automática de teoremas en matemática y lógica.
- **Solucionadores de Problemas Generales**: orientado a la solución de clases de problemas generales expresados en un lenguaje formal.
- **Reconocimiento de Patrones**: orientado al reconocimiento y clasificación de patrones.

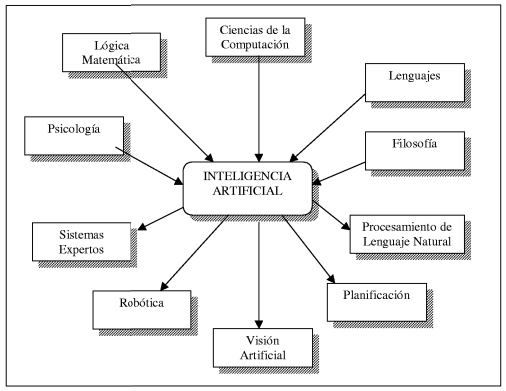

Figura 3.2: Campos de la Inteligencia Artificial

- **Juegos**: orientado a la construcción de programas de juegos competitivos.
- **Aprendizaje de Máquina**: orientado a la construcción de máquinas capaces de acumular conocimiento mediante la observación de ejemplos.

En la figura 3.2 se muestra la relación existente entre los Sistemas Basados en el Conocimiento (Sistemas Expertos) y algunos de los campos de la Inteligencia Artificial.

3.2 Sistemas Expertos

Los sistemas expertos son un producto del área de la ciencia en inteligencia artificial. Estos sistemas emulan el comportamiento de un especialista humano al enfrentar un problema complejo que requiere de conocimientos especializados.

En la actualidad existe una gran cantidad de sistemas de esta especie. Son especialmente conocidos los que se desarrollaron para el área médica, entre todos, probablemente el más famoso, es MYCIN desarrollado en la Universidad de Stanford, California en la década del setenta cuyo objeto es el diagnóstico tratamiento de formas complejas de meningitis y de infecciones bacterianas.

3.2.1 ¿Qué es un Sistema Experto?

Se considera que un sistema experto es la incorporación en un computador de un componente basado en el conocimiento que se obtiene a partir de la habilidad de un experto, de forma tal que el sistema pueda dar consejos inteligentes o tomar decisiones inteligentes. Una característica adicional, es que el sistema sea capaz, bajo demanda,

de justificar su propia línea de razonamiento de una forma inmediatamente inteligible para el que los usa.

Se trata de programas que tienen la capacidad para ayudar, aconsejar, analizar, consultar y determinar la pertenencia de un objeto, fenómeno o situación a una clase dada. Son herramientas para tratar problemas que generalmente exigen para su solución total o parcial de la presencia de un especialista. Se diferencian de los programas más convencionales por el uso de procedimientos deductivos y por la posibilidad de realizar "razonamientos aproximados o inciertos. Esto les dota de la capacidad para resolver problemas insuficientemente definidos o pobremente estructurados.

3.2.2 Definiciones

A continuación se presentan algunas definiciones de los Sistemas Expertos:

- Un sistema experto es un sistema informático que incorpora, en forma operativa, el conocimiento de una persona experimentada, de forma que es capaz tanto de responder como esta persona, como de explicar y justificar sus respuestas.
- Un sistema experto (SE) es un sistema computacional que manipula conocimientos organizados sobre algún campo específico de expertos humanos y es programado para trabajar convincentemente como un asesor en el campo dado con capacidad para explicar su razonamiento si éste es solicitado.
- Una definición que expone de otra forma la filosofía de los SE es la que Naylor propone: "Un SE es un sistema computacional compuesto de una base de conocimiento extraída de uno(s) experto(s) de tal forma que el sistema pueda ofrecer un consejo inteligente o pueda tomar una decisión inteligente acerca del desarrollo de una actividad. Este sistema deberá justificar su propia línea de razonamiento lógico de una forma directamente inteligible por el usuario".

Un sistema experto debe ser capaz de realizar:

- Resolver problemas muy difíciles tan bien o mejor que un experto humano.
- Razonar heurísticamente, utilizando reglas que los expertos humanos consideran eficaces.
- interactuar eficazmente y en lenguaje natural con las personas.
- Manipular descripciones simbólicas y razonar sobre ellas.
- Funcionar con datos erróneos y reglas imprecisas.
- Contemplar simultáneamente múltiples hipótesis alternativas.
- Explicar por qué plantea sus preguntas.
- Justificar sus conclusiones.

3.2.3 Diferencia entre un experto humano y un sistema experto

Experto Humano	Sistema Experto
No Perdurable	Permanente
Dificil de transferir	Facil de transferir
Dificil de documentar	Facil de documentar
Impredecible	Consistente
Caro	Alcanzable
Creativo	No inspirado
Adaptativo	Necesita ser enseñado
Experiencia personal	Entrada simbólica
Enfoque amplio	Enfoque cerrado
Conocimiento del sentido común	Conocimiento técnico

Tabla 2. Diferencias entre un Experto humano y un Sistema Experto

3.2.4 Elementos de un Sistema Experto

Los elementos de un Sistema Experto son:

- **Base de conocimientos**: contiene una gran cantidad de información sobre un tema específico, generalmente introducida por un experto en dicho tema, sobre el cual se desarrolla la aplicación Depósito de las primitivas del conocimiento. Su función consiste en almacenar los conocimientos relativos al área del S. E. y, por lo tanto, depende del modelo de representación del conocimiento que se utilice.
  - Representación basada en Reglas de Producción.
  - Representación basada en Lógica de Predicados.
  - Representación basada en Redes Semánticas.
  - Representación basada en Marcos.
  - Representación basada en Restricciones.

  El diseño de este esquema de representación del conocimiento afecta al diseño del motor de inferencia, al proceso de actualización del conocimiento, al proceso de explicación y a la eficacia global del sistema.

- **Motor de inferencia**: utiliza la base de conocimientos y la base de hechos, los fusiona y los combina realizando de esta manera una serie de razonamientos acerca del problema que se le ha presentado. El paradigma del motor de inferencia es la estrategia de búsqueda para producir el conocimiento demandado. Varios paradigmas se emplean en un Sistema Experto, pero la mayoría de ellos se basan en dos conceptos fundamentales:

  - encadenamiento hacia atrás que es un proceso de razonamiento descendente, que se inicia a partir de los objetivos deseados y trabaja hacia atrás en dirección a las condiciones iniciales.
  - encadenamiento hacia adelante que es un proceso de razonamiento ascendente que se inicia con las condiciones conocidas y trabaja hacia adelante para alcanzar los objetivos deseados.

- **Sistema de explicación**: son módulos de interacción con el usuario, o sea, posibilitan la fácil comunicación entre la máquina y el operador. Mantiene una memoria temporal de los conocimientos empleados en el procesamiento para poder dar una explicación al usuario. El sistema debe acceder a un registro de los conocimientos que se emplearon en el procesamiento, basándose en el esquema de representación de la base de conocimientos y traducirlo a una forma que sea aceptable por el usuario. Este hecho constituye esencialmente un subconjunto del problema del procesamiento del lenguaje natural. Las facilidades de explicación de algunos sistemas actuales se limitan a listar simplemente las reglas que se utilizaron durante la ejecución.

3.2.5 Areas de aplicación de los Sistemas Expertos

- Interpretación: infieren la descripción de situaciones por medio de sensores de datos. Estos Sistemas Expertos usan datos reales, con errores, con ruidos, incompletos etc. Ejemplos: medición de temperatura, reconocimiento de voz, análisis de señales etc.

- Predicción: infieren probables consecuencias de situaciones dadas. Algunas veces usan modelos de simulación para generar situaciones que puedan ocurrir. Ejemplos: predecir daños a cosechas por algún tipo de insecto.

- Diagnostico: infieren las fallas de un sistema basándose en los síntomas. Utilizan las características de comportamiento, descripción de situaciones o conocimiento sobre el diseño de un componente para inferir las causas de la falla. Ejemplos: diagnostico de enfermedades basándose en síntomas, encontrar componentes defectuosos o fallas en circuitos.

- Diseño: configuración de objetos. Utilizan un conjunto de limitaciones y restricciones para configurar objetos. Utilizan un proceso de análisis para construir un diseño parcial y una simulación para verificar o probar las ideas. Ejemplos: configuración de equipos de oficina, de equipos de computo.

- Planeación: diseñan un curso completo de acción, se descompone la tarea en un subconjunto de tareas. Ejemplo: transferir material de un lugar a otro, comunicaciones, enrutamiento, planificación financiera.

- Monitoreo: comparan observaciones del comportamiento del sistema con el comportamiento estándar, se compara lo actual con lo esperado. Ejemplo: asistir a un paciente de cuidados intensivos, trafico aéreo, uso fraudulento de tarjetas de créditos.
- Depuración: ("debugging") sugieren remedios o correcciones de una falla. Ejemplo: sugerir el tipo de mantenimiento a cables dañados, la prescripción medica a un paciente.
- Reparación: sigue un plan para administrar un remedio prescrito. Poco se ha hecho, requiere planeación, revisión y diagnostico.
- Instrucción: diagnostican, revisan y reparan el comportamiento de un estudiante. Ejemplo: educar a un estudiante de medicina, usa un modelo del estudiante y planea la corrección de deficiencias.
- Control: gobierna el comportamiento del sistema. Requieren interpretar una situación actual, predecir el futuro, diagnosticar las causas de los problemas que se pueden anticipar, formular un plan para remediar estas fallas y monitorear la ejecución de este.

3.3 Representación del Conocimiento

El conocimiento puede definirse como el conjunto de hechos y principios acumulados por una clase de acto, hecho o estado del conocimiento. Sin embargo, el conocimiento es mucho más que esto, ya que incluye la familiaridad en el lenguaje, conceptos, procedimientos, reglas, ideas, abstracciones, lugares, costumbres, hechos y asociaciones, junto con la capacidad de utilizar todos estos conceptos eficientemente para modelar diferentes aspectos del mundo.

Con el objeto de resolver los problemas complejos encontrados en la Inteligencia Artificial, es necesario contar con una gran cantidad de conocimiento y algunos mecanismos para manipularlo para generar soluciones a nuevos problemas. Así, la forma en la que se represente dicho conocimiento determina la eficiencia en su manipulación. Un sistema para la representación del conocimiento debe contar con las siguientes cuatro propiedades:

- **Suficiencia de Representación**: la capacidad de representar todos los tipos de conocimiento que son necesarios en el dominio.
- **Suficiencia de Inferencia**: la capacidad de manipular las estructuras de representación de tal forma que se deriven nuevas estructuras correspondientes al nuevo conocimiento inferido.
- **Eficiencia de Inferencia**: es la habilidad de incorporar a la estructura del conocimiento información adicional que pueda guiar al mecanismo de inferencia para obtener mejores resultados.

- **Eficiencia de Adquisición**: es la capacidad de adquirir nuevo conocimiento.

No existe alguna representación lo suficientemente completa que incluya estas cuatro propiedades. Sin embargo, algunas de las que se mencionarán a continuación son una parte representativa de la investigación que se ha realizado en este rubro.

3.3.1 Tipos de Representación

La representación del conocimiento puede dividirse en declarativo (representando hechos y aserciones) y procedural (guardando acciones o consecuencias). Una forma de caracterizarlos es la de **saber qué** (declarativo) *versus* **saber cómo** (procedural).

Dentro de los esquemas declarativos se encuentran aquellos basados en la lógica y los relacionales. Los modelos relacionales pueden representar el conocimiento por medio de árboles, grafos, tablas, marcos, guiones, redes semánticas, o gráficas conceptuales. Las representaciones lógicas incluyen el uso de la lógica proposicional y, más importante aún, la lógica de predicados.

Los modelos procedurales almacenan el conocimiento de forma de cómo hacer las cosas. Se caracterizan por el uso de gramáticas formales implementadas por sistemas de producción basados en reglas o en procedimientos.

Conocimiento Relacional Simple

Es una forma sencilla para representar hechos declarativos como un conjunto de relaciones del mismo tipo utilizados en sistemas de bases de datos. Una de las desventajas es que no facilita la capacidad de inferencia. Sin embargo, puede servir como el conocimiento de entrada a mecanismos de inferencia más poderosos. Un ejemplo de este tipo de representación se encuentra en la Figura 3.3.

Conocimiento Heredable

Con el modelo relacional se cuenta con un conjunto de atributos y un conjunto de valores que describen a los objetos en la base de conocimientos. Para facilitar la inferencia el conocimiento se representa como heredable, es decir, donde los elementos de ciertas clases heredan atributos y valores de clases más generales en las que se encuentran incluidos. Este tipo de organización se conoce como *Redes Semánticas*. Las redes semánticas son un esquema para representar relaciones abstractas entre objetos en el dominio del problema, como miembros en una clase. Consisten de un grupo de nodos que se encuentran ligados para formar las relaciones entre objetos. Los nodos representan conceptos. Un ejemplo se ilustra en la Figura 3.4.

Algunas de las ventajas de esta representación son:

- Flexibilidad para agregar, modificar, o eliminar nodos y arcos
- Capacidad de heredar relaciones de otros nodos

Código	Materia	Calificación
IS2345	Sistemas Experos	19
IS6578	Inteligencia Artificial	16
IS8790	Tesis	19

Figura 3.3: Representación Relacional

- Fácil de dibujar inferencias sobre la jerarquía de herencia

Sin embargo, una fuerte desventaja de este tipo de representación es la falta de una estructura formal definitiva.

Otro tipo de representación heredable son los *Marcos*. Estos consisten de un grupo de *ranuras* que contienen atributos para describir un objeto, una clase de objetos, una situación, una acción, o un evento. Difieren de las *Redes Semánticas* en que los *Marcos* contienen un subconjunto de elementos que pueden representarse en una *Red Semántica*. En una *Red Semántica*, la información referente a un objeto puede encontrarse en cualquier parte de la base de conocimientos, mientras que en un *Marco* la información se agrupa en una sola unidad. Un ejemplo se ilustra en la Figura 3.5.

Algunas de las ventajas de utilizar *Marcos* son:

- Los Marcos están organizados jerárquicamente tal que pueden heredar relaciones de otros Marcos.
- Facilitan la búsqueda rápida de la base de conocimientos a través de la representación compacta de la información.
- Permiten la representación de relaciones de herencia entre objetos.

Los *Guiones (scripts)* son un tipo especial de marcos. Describen una secuencia de eventos en un contexto particular. Presentan la secuencia esperada de eventos y la información asociada a ellos en una serie de marcos ligados en el tiempo. Ejemplo de un guión se ilustra en la Figura 3.6. Cabe notar que en este tipo de representación es necesario especificar primitivas de acción como son: ATTEND (Poner atención a un objeto), PTRANS (Cambiar físicamente de un lugar a otro), MOVE (Mover una parte

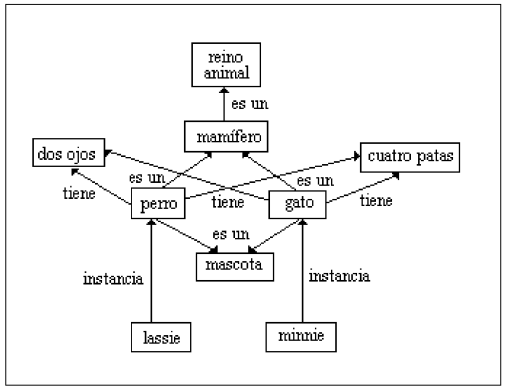

Figura 3.4: Red Semántica

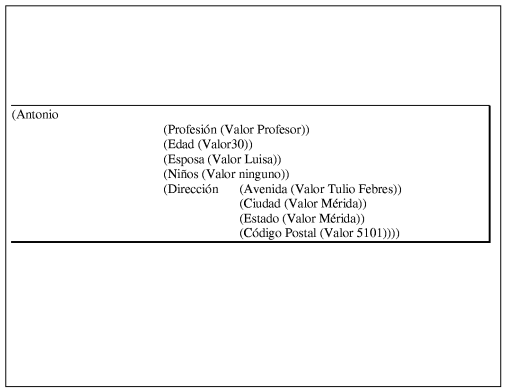

Figura 3.5: Marcos

del cuerpo), SPEAK (Hablar), GRASP (Sostener un objeto), etc. que nos permitan describir las escenas del guión.

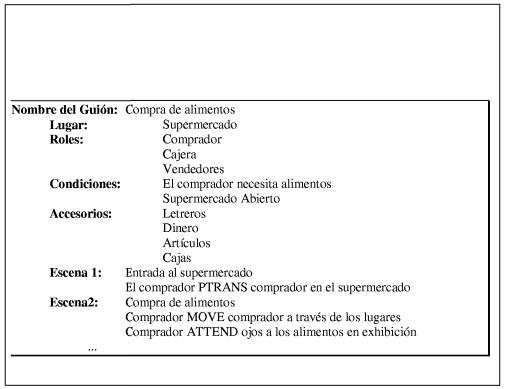

Figura 3.6: Ejemplo de un Guíon

Los *Grafos Conceptuales* son otra forma de representación. Los nodos del grafo son conceptos o relaciones conceptuales. La diferencia con las redes semánticas, que son un tipo especial de grafos, es que los arcos no están etiquetados; en lugar de ello, los nodos de relaciones conceptuales representan las relaciones entre los conceptos. Así, los nodos que representan conceptos sólo pueden tener arcos a nodos que representan relaciones, y viceversa. La Figura 3.7 ilustra un ejemplo de una grafo conceptual.

Conocimiento Inferenciable

Una herencia apropiada es una poderosa forma de inferencia, pero no es la única forma. Algunas veces, es necesario todo el poder de la lógica tradicional para describir las inferencias que se necesitan. La representación lógica es el resultado de filósofos y matemáticos por buscar caracterizar los principios del razonamiento correcto. Se busca llegar a la representación formal de los lenguajes con reglas de inferencia completas. Consecuentemente, la semántica del cálculo de predicados enfatiza las operaciones de mantenimiento de la verdad en fórmulas bien formadas.

En el siguiente ejemplo se ilustra el uso de la lógica de primer orden para representar conocimiento.

$$\forall x : [Romano(x) \wedge conoce(x, Marcos)] \rightarrow \quad (3.1)$$

$$[odia(x, Cesar) \vee (\forall y : \exists z : odia(y, z)] \rightarrow esta\_loco(x, y)) \quad (3.2)$$

Sin embargo, este conocimiento es poco útil a menos que exista un procedimiento de inferencia que pueda explotarlo. Este procedimiento debe de implementar las reglas

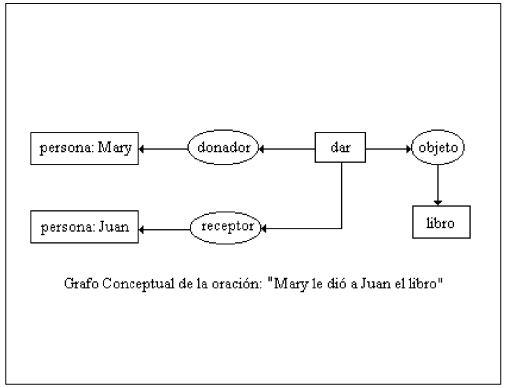

Figura 3.7: Grafo conceptual

de inferencia de la lógica estándar. Existen varios de estos procedimientos, algunos razonan hacia adelante partiendo de los hechos hacia las conclusiones, otros razonan hacia atrás, de las conclusiones hacia los hechos. Una de las aplicaciones más comunes es la *resolución*, que utiliza pruebas por medio de contradicción.

El cálculo proposicional es un sistema elemental de la lógica formal que se utiliza para determinar si una proposición es verdadera o no. El cálculo de predicados agrega, además, la capacidad de especificar relaciones y hacer generalizaciones sobre las proposiciones convirtiendo los argumentos en variables.

Las ventajas de la lógica de predicados son:

- Simplicidad en la notación que permite que las descripciones sean comprensibles.
- La modularidad permite que se agreguen, eliminen, o modifiquen enunciados sin afectar otros enunciados en la base de conocimientos.
- La consistencia es una ventaja porque cada hecho tiene que representarse sólo una vez.
- Las técnicas de Demostración de Teoremas pueden utilizarse para generar nuevos hechos de otros hechos previos.

Las desventajas son:

- Dificultad para representar conocimiento procedural y heurístico.
- Dificultad para manejar grandes bases de conocimiento debido a su estructura organizacional restringida.
- Procedimientos limitados de manipulación de datos.

Conocimiento Procedural

Este tipo de conocimiento especifica qué hacer y cuándo. Se puede codificar en diferentes tipos de programas utilizando varios lenguajes como LISP, C, Pascal, Fortran, etc. La técnica más utilizada para representar el conocimiento procedural es por medio de *reglas de producción*. Esto se ilustra a continuación:

- **Si** clima = frío, y mes = julio, y hora = 6:00 p.m., y cielo = con nubes
  Entonces: sacar paraguas

Podemos ver que la estructura básica de una regla es: **SI** *premisa*, **ENTONCES** *conclusión*. Donde la premisa se refiere a los hechos que deben cumplirse para poder realizar la conclusión asociada a ella. Las ventajas de este tipo de representación son:

- Las reglas pueden agregarse, eliminarse o actualizarse fácilmente.
- La representación del conocimiento es directa y es fácil de interpretar.
- Están estructuradas de forma similar a cómo las personas razonan para resolver problemas.
- Son útiles para representar la interacción entre conocimiento declarativo y procedural.

Una de las desventajas es que se necesita de un mecanismo de búsqueda eficiente para encontrar las reglas adecuadas dependiendo del problema que se intenta resolver.

En resumen, en el cálculo de predicados y las redes semánticas podemos representar objetos utilizando símbolos sencillos. Las representaciones con marcos y guiones nos permiten definir estructuras complejas por medio del encapsulamiento de características y atributos. Además, el cálculo de predicados se ocupa de las asignaciones con valor de *verdad* pero no de las relaciones semánticas entre los componentes de un dominio. Las representaciones mencionadas dentro del conocimiento heredable son un caso típico de grafos.

Podemos observar que la elección de la forma en la que se va a representar el conocimiento es determinante para el problema que se requiere enfrentar. Cada tipo de representación tiene ventajas pero también sus desventajas. Así, es necesario analizar cada una de ellas para hacer la mejor elección y adaptarla al problema a solucionar.

3.4 Manejo de Incertidumbre e Inconsistencia

3.4.1 Introducción

Todo sistema que tenga por objetivo simular un comportamiento inteligente entre componentes inteligentes y autónomos (ej. Sistemas Multiagentes) o utilizando un sólo componte pero que termine con éxito lo que se conoce como Prueba de Turing

(ej. Sistema Experto), debe de considerar dentro de sus restricciones el manejo de incertidumbre e inconsistencia. Los formalismos de la lógica clásica no permiten utilizar representaciones reales del mundo en el que vivimos. Por el contrario, los seres inteligentes continuamente necesitan tomar decisiones bajo el velo de la incertidumbre.

La incertidumbre puede venir de una gran variedad de fuentes. Por ejemplo, la información puede estar incompleta o puede proceder de fuentes poco confiables. Detalles y hechos importantes pueden no tomarse en cuenta o cambiar rápidamente. Además, los hechos que están disponibles pueden ser imprecisos, vagos o difusos. Así, mucha de la información puede ser contradictoria o poco creíble. Sin embargo, diariamente nos enfrentamos a este tipo de situaciones y, generalmente, llegamos a soluciones razonables. De lo contrario, no seríamos capaces de adaptarnos a un mundo dinámico.

Se han desarrollado diferentes métodos para el manejo de conocimiento incierto, difuso, y cambiante. Explicaremos a continuación dos propuestas:

- *Razonamiento no monotónico*: en éste, los axiomas y/o reglas de inferencia se complementan para permitir el razonamiento con información incompleta. Sin embargo, estos sistemas conservan la propiedad de que, en cualquier momento, dada una afirmación, ésta se cree verdadera, falsa, o ninguna de las anteriores.
- *Razonamiento estadístico*: en éste, la representación se complementa para permitir algún tipo de medida numérica de certeza (en lugar de sólo VERDADERO o FALSO) asociada a cada afirmación o enunciado.

Los sistemas convencionales de razonamiento, como la lógica de predicados de primer orden, está diseñada para trabajar con información que tiene tres propiedades importantes:

- Es completa con respecto al dominio de interés. Es decir, todos los hechos que son necesarios para resolver un problema están presentes en el sistema o pueden derivarse de los que se encuentren por medio de las reglas convencionales de la lógica de primer orden.
- Es consistente.
- La única forma en que puede cambiar es que pueden agregarse nuevos hechos. Si estos nuevos hechos son consistentes con los hechos previos, nada se cambiará del conjunto de hechos que se conoce que son verdaderos. Esto se conoce como conocimiento *monotónico*.

Si alguna de estas propiedades no se cumple, estos sistemas de razonamiento fallan. Los sistemas de razonamiento no monotónico se diseñaron para resolver problemas en los que una o todas de estas propiedades pueden no cumplirse.

3.4.2 Razonamiento No Monotónico

La lógica clásica descansa sobre la premisa de que las deducciones que se obtienen a partir de ella son válidas y permanecen así. Al agregar nuevos axiomas se incrementa la cantidad de conocimiento en la base de conocimientos. Entonces, el conjunto de hechos e inferencias en estos sistemas sólo pueden crecer, no reducirse; esto es, crecen monotónicamente. El razonamiento no monotónico contempla el hecho de que nueva información puede cambiar las creencias o deducciones previas.

Con el objeto de tratar de modelar mundos más reales, así como el razonamiento de sentido común, los investigadores han propuesto extensiones y alternativas a la lógica tradicional. Estas extensiones incorporan diferentes formas de incertidumbre y no monotocidad. Explicaremos a continuación algunos de los modelos creados:

3.4.3 Razonamiento Probabilístico

Enfoque Bayesiano

Utilizando la teoría de probabilidades es posible generalizar observaciones sobre eventos para llegar a afirmaciones sobre poblaciones de objetos, o viceversa, de poblaciones llegar a eventos específicos. El método Bayesiano utiliza el Teorema de Bayes para manejar incertidumbre en el proceso de inferencia sobre objetos o eventos. El Teorema de Bayes dice:

Sea $\{B_1, B_2, \ldots, B_n\}$ un conjunto de eventos que forman una partición del espacio muestra S, donde $P(B_i) \neq 0$, para $i = 1, 2, \ldots, n$. Sea A cualquier evento de S tal que $P(A) \neq 0$. Entonces, para $K = 1, 2, \ldots, n$, se tiene

$$P(B_k|A) = \frac{P(B_k \cap A)}{\sum_{i=1}^{n} P(B_i \cap A)} = \frac{P(B_k)P(A|B_k)}{\sum_{i=1}^{n} P(B_i)P(A|B_i)} \tag{3.3}$$

En la Figura 3.8 el área sombrada representa el evento A y los eventos etiquetados B_i son los eventos sobre los cuales se realiza la inferencia. El Teorema de Bayes permite calcular la probabilidad de tener un evento B_i dado que el evento A ha ocurrido.

Ejemplo: Si se conoce que el 2

Dado el hecho:$P(T) = 0.02$

Las Variables Definidas: $P(X|T), P(X|Not - T), P(T|X)$

$P(X|T)$ = probabilidad de que los rayos-X de una persona con tuberculosis sean positivos.

$P(X|Not - T)$ = probabilidad de que los rayos -X de una persona saludable sean positivos. $P(T|X)$ = probabilidad de que una persona con rayos-X positivos tenga tuberculosis. y dada la información de que $P(X|T) = 0.90$ y $P(X|Not - T) = 0.01$ Calcular $P(T|X)$. Utilizando el Teorema de Bayes, se puede calcular $P(T|X)$ de la siguiente manera:

$$P(T|X) = \frac{P(T)P(X|T)}{P(T)P(X|T) + P(Not - T)P(X|Not - T)} \tag{3.4}$$

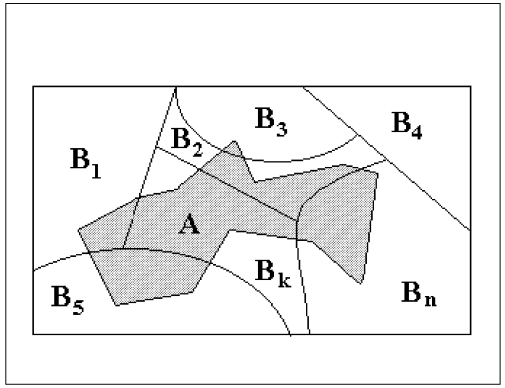

Figura 3.8: Teorema de Bayes

$$= \frac{(0.02)(0.90)}{(0.02)(0.90) + (0.98)(0.01)} = 0.648 \tag{3.5}$$

Sin embargo, aunque las técnicas para aplicar métodos probabilísticos están bien desarrolladas, existen varias razones por las que el análisis probabilístico convencional no ha sido muy popular en implementaciones prácticas (ej. Sistemas Expertos). Algunas de estas razones son:

1. Las probabilidades *a priori*, $P(A|B_i)$, deben ser conocidas. Debido a que los métodos más heurísticos para la solución de problemas utilizan el juicio experto más que hechos matemáticos, estas probabilidades *a priori* generalmente no están disponibles.

2. El razonamiento Bayesiano puede caer en una explosión combinatoria del análisis.

3. Los usuarios no familiarizados con probabilidades pueden mal interpretar los resultados de un análisis probabilístico.

Así, los métodos estadísticos para razonamiento pueden ser *óptimos* sólo desde un punto de vista teórico.

Redes Bayesianas

El razonamiento, utilizando sentido común, generalmente es *incierto*. Sin embargo, es posible construir y utilizar un método probabilístico para este tipo de razonamiento. Para utilizar la teoría de probabilidades se debe de razonar acerca de las probabilidades en el conocimiento incierto o encontrar alguna forma de representar una distribución

de probabilidad completa sobre los posibles eventos. Las redes Bayesianas proveen este último punto, ya que generan un modelo formal de algunas formas de razonamiento utilizando sentido común y, más importante aun, proveen un formalismo para razonar sobre las probabilidades en un amplio rango de aplicaciones.

Una Red Bayesiana modela la estructura causal de un proceso no determinista. Es un grafo acíclico dirigido, $G = V, E$, donde el conjunto de vértices representa variables y cada arco las relaciones de causalidad que existen entre ellas. La variable al final del arco es dependiente de la variable que se encuentra al inicio del mismo. Los padres de un nodo son aquellos que tienen arcos dirigidos a él. Cada nodo tiene una tabla de probabilidad condicional que cuantifica el efecto de sus padres sobre él. Un ejemplo ese ilustra en la Figura 3.9.

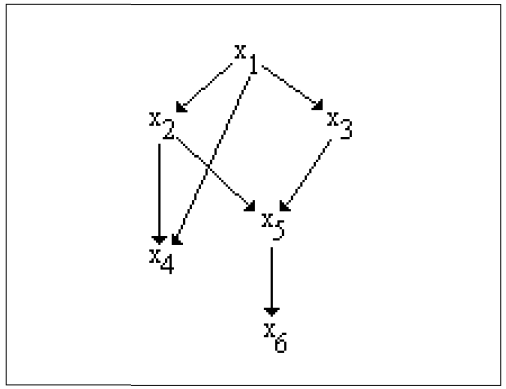

Figura 3.9: Red Bayesiana

Así, para representar relaciones causales entre las variables $x_1, \ldots, x_6$ de la figura anterior, se puede utilizar la probabilidad conjunta como un producto de las probabilidades condicionales de la siguiente manera:

$$P(x_1, \ldots, x_6) = P(x_6|x_5)P(x_5|x_3, x_2)P(x_4|x_1, x_2)P(x_3|x_1)P(x_2|x_1)P(x_1) \quad (3.6)$$

es decir:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|Padres(x_i)) \quad (3.7)$$

En las Redes Bayesianas, una variable toma sus valores de un grupo de estados mutuamente exclusivos y exhaustivos. Una variable puede ser discreta, teniendo un número finito de valores, o puede ser continua. La elección de los valores por sí misma representa un trabajo interesante. Cuando se observa el valor de cada variable en un

conjunto X, llamamos a este conjunto de observaciones una instancia de X. El espacio de unión de un conjunto de variables U es el conjunto de todas las instancias de U. La distribución de probabilidad conjunta sobre U es la distribución de probabilidad sobre el espacio de unión de U.

El problema general de calcular probabilidades de interés a partir de una distribución de probabilidad conjunta se conoce como inferencia probabilística. Todos los algoritmos exactos de inferencia probabilística en Redes Bayesianas utilizan la independencia condicional. Pero, a pesar de que se pueden utilizar las aseveraciones de independencia condicional en una Red Bayesiana para la inferencia probabilística, una **inferencia exacta** en una Red Bayesiana cualquiera es un problema **NP-duro**.

Sin embargo, una de las ventajas de las Redes Bayesianas es que no es necesario utilizar una tabla de probabilidades conjuntas completa en la que se listen las probabilidades de todas las posibles combinaciones de los eventos que se puedan concebir. La mayor parte de los eventos son *condicionalmente independientes* de otros, por lo cual sus interacciones no deben ser consideradas. En lugar de esto, es posible utilizar una representación local en la cual se describan agrupamientos de eventos que interactúan.

La tarea básica de un sistema probabilístico de inferencia es calcular la distribución de probabilidades *a posteriori* de un conjunto de variables dados ciertos valores de las evidencias.

Las Redes Bayesianas pueden realizar cuatro tipos de inferencia:

1. Diagnóstico. De Efectos a Causas
2. Causales. De Causas a Efectos
3. Inferencias Intercausales. Entre Causas de un Efecto común
4. Inferencias Mixtas. Combinación de dos o más de los anteriores

Por ejemplo, si tenemos la Red Bayesiana ilustrada en la Figura 3.10:

Una inferencia por Diagnóstico es: Dado que Arturo llamó, inferir la probabilidad de que hubo un asalto: P(Asalto Arturo llamó). Una inferencia Causal es: Dado que hubo un asalto, inferir la probabilidad de que Arturo llame : P(Arturo llamó Asalto). Una inferencia Intercausal es: Dado que la alarma sonó, tenemos que P(Asalto Alarma)= 0.376. Pero si se añade la evidencia de que hubo un terremoto, entonces P(Asalto Alarma y Terremoto)= 0.003. Aun cuando los asaltos y los terremotos son independientes, la presencia de alguno de ellos provoca que el otro sea menos probable. Una inferencia Mixta es: Dado que Arturo llama es verdadero y que Terremoto es falso entonces: P(Alarma Arturo llama y $\neg$ Terremoto)= 0.03. Esta es una inferencia de Diagnóstico y Causal. También, P(Asalto Arturo llama y $\neg$ Terremoto)= 0.017. Esta es una combinación de inferencias Intercausal y de Diagnóstico.

3.4.4 Teoría de Dempster-Shafer

La Teoría de Dempster-Shafer es otro método para manejar incertidumbre. Hace la distinción entre ignorancia e incertidumbre ya que deben tratarse diferente. El

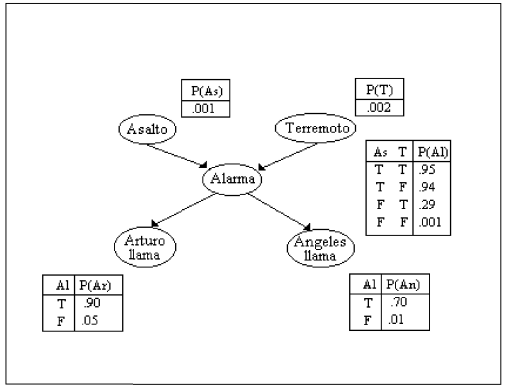

Figura 3.10: Ejemplo de Red Bayesiana

no conocer el valor de una variable no necesariamente significa que está sujeta a la incertidumbre. Con la teoría clásica de probabilidades, es necesario considerar *conocer y desconocer* como opuestos. Esto es, si A, B, y C son los únicos tres eventos en un espacio muestra (S) y sabemos que $P(A) = 0.2$ y $P(B) = 0.5$, entonces necesariamente $P(C) = 0.3$, ya que $P(S) = 1.0$ y $P(A) + P(B) + P(C) = P(S)$. Sin embargo, esto no puede ser exacto en la representación del razonamiento humano porque es posible que una persona crea o no crea tres diferentes cosas con la misma probabilidad. La probabilidad de C puede no tener nada que ver con incertidumbre o probabilidades. El hecho es que tal vez ignoremos el valor de C. Así, si sabemos los valores de A y B no necesariamente implica que podemos inferir la probabilidad de C.

Dempster, en 1968, propuso una teoría generalizada de la incertidumbre *versus* la ignorancia. Este teoría fue extendida por Shafer en 1976 y está basada en la noción de que probabilidades separadas pueden asignarse a todos los subconjuntos del universo de discurso en lugar de sólo a miembros únicos; como se requiere en la teoría tradicional de probabilidad. Como resultado, la teoría de Dempster-Shafer permite la siguiente inequidad:

$$P(A) + P(B) \leq 1.0 \tag{3.8}$$

A continuación veremos un ejemplo de cómo aplicar ésta teoría:

Definiremos un universo de hipótesis mutuamente excluyentes. Este universo se llamará *Marco de Discernimiento* y lo denotaremos por Ω. Así, en un problema de diagnóstico, Ω puede consistir de {alergia, gripe, resfrío, neumonía }. La meta es determinar alguna medida de credibilidad para los elementos de Ω. Sin embargo, no toda la evidencia confirma a un sólo elemento. Es decir, si tenemos como evidencia *fiebre*, ésta puede confirmar {gripe, resfrío, neumonía}.

Para trabajar con esto, la teoría de Dempster-Shafer utiliza una función de densidad de probabilidad que denotaremos por m. La función m está definida no sólo para los elementos de Ω sino también para todos sus subconjuntos. El valor $m(p)$ mide la cantidad de creencia que se asigna al subconjunto p de la hipótesis. Si Ω tiene n elementos, entonces existen 2^n subconjuntos de Ω. Se deben asignar un valor a m tal que la suma de todos los valores de m asignados a los subconjuntos de Ω sea 1.

A continuación se ilustrará un ejemplo de cómo funciona m en un problema de diagnóstico. Asumamos que no tenemos información sobre cómo elegir entre las cuatro hipótesis al inicio del diagnóstico. Entonces m debe definirse como:

$$\Omega \quad (1.0) \tag{3.9}$$

Todos los demás valores de m son, entonces, 0. Esto significa que la respuesta al diagnóstico se encuentra al alguna parte de todo el conjunto Ω. Ahora, supongamos que contamos con cierta información que, con un valor de 0.6, nos dice que el diagnóstico está en el conjunto gripe, resfrío, neumonía. Entonces el valor de m debe actualizarse como sigue:

$$gripe, resfrio, neumonia \quad (\quad 0.6) \tag{3.10}$$

$$\Omega \quad (\quad 1.0) \tag{3.11}$$

Así asignamos el valor de certeza adecuado a {gripe, resfrío, neumonía}. El resto de nuestra creencia permanece en el conjunto Ω. Cabe hacer la aclaración de que no se afirma que el resto debe asignarse al complemento de {gripe, resfrío, neumonía}.

Ahora bien, supongamos que recibimos más información que nos determina lo siguiente:

$$alergia, gripe, resfrio \quad (\quad 0.8) \tag{3.12}$$

$$\Omega \quad (\quad 0.2) \tag{3.13}$$

Entonces debe de calcularse la combinación de ambas informaciones para sacar un nuevo valor de m. Esto se realizará con ayuda de la siguiente tabla:

		{A,G,R}	(0.8)	Ω	(0.2)
{G,R,N}	(0.6)	{G,R}	(0.48)	{G,R,N}	(0.12)
Ω	(0.4)	{A,G,R}	(0.32)	Ω	(0.08)

Tabla 3. Cálculo del valor de m

La tabla anterior ilustra la aplicación de la siguiente fórmula:

$$m_3(Z) = \frac{\sum_{X \cap Y = Z} m_1(X) m_2(Y)}{1 - \sum_{X \cap Y} m_1(X) m_2(Y)} \tag{3.14}$$

donde X es el conjunto de subconjuntos de Ω a los que m_1 asigna un valor diferente de 0 y Y es el correspondiente conjunto para m_2.

Así, los cuatro conjuntos generados al tomar todas las intersecciones de un elemento de X con un elemento de Y se muestran en la tabla 3. El valor para m_3 se calcula al multiplicar los valores de m_1 y m_2 asociados con los elemento de donde se derivaron. Entonces, m_3 queda de la siguiente manera:

$$gripe, resfrio \quad (\quad 0.48) \tag{3.15}$$
$$alergia, gripe, resfrio \quad (\quad 0.32) \tag{3.16}$$
$$gripe, resfrio, neumonia \quad (\quad 0.12) \tag{3.17}$$
$$\Omega \quad (\quad 0.08) \tag{3.18}$$

3.4.5 Factores de Certeza

Es la forma más común de representar pesos heurísticos. Junto con la lógica difusa, los factores de certeza es un método heurístico que utiliza técnicas pseudo-probabilísticas para manejar la incertidumbre.

En este método, los números mayores a 0 se utilizan para una evidencia positiva y números menores a 0 se utilizan para una evidencia negativa. Los factores de certeza indican el grado de certeza con el que se cree que cada regla o hecho es verdadero. Fueron desarrollados por Shortliffe en los 70's y aplicados en el sistema MYCIN, que intentaba recomendar terapias para pacientes con infecciones bacteriales.

Los factores de certeza se utilizan principalmente en los sistemas expertos, donde a cada regla se le asocia un valor de credibilidad a la conclusión.

En la Figura 3.11 se muestra un ejemplo de combinar una serie de factores de certeza para un parámetro. El método ilustrado en la figura utiliza niveles de certeza entre 0 y 1. El factor de certeza se calcula con la siguiente fórmula:

$$CCF = (CF\ Inicial) + (CF\ Subsecuente)(1 - CF\ Inicial) \tag{3.19}$$

Esta fórmula se aplica repetidamente en caso de que existan más de dos factores de certeza en serie. La falla en este método es que, si el primer factor de certeza encontrado para el parámetro es 1, entonces todos los valores de certeza subsecuentes para el parámetro no tendrán efecto alguno en la instanciación previa. Por ello, este método regresa a la lógica monotónica.

La Figura 3.12 muestra cómo el nivel de certeza de una premisa induce un nivel de certeza en la conclusión de la regla. La Figura 3.13 muestra el efecto de combinar una premisa incierta y una regla incierta. Se siguen investigando mecanismos que puedan reflejar más adecuadamente el proceso de razonamiento humano cuando enfrenta información incierta. A pesar de sus deficiencias, los métodos *ad hoc* para combinar los factores de certeza han sido más utilizados que los métodos formales. Esto se debe a que los métodos formales son difíciles de implementar.

Un análisis de sensibilidad realizado por Buchanan *et al* en 1984 muestra que los métodos *ad hoc*, aunque no son óptimos, satisfacen las necesidades básicas en la mayoría de los problemas.

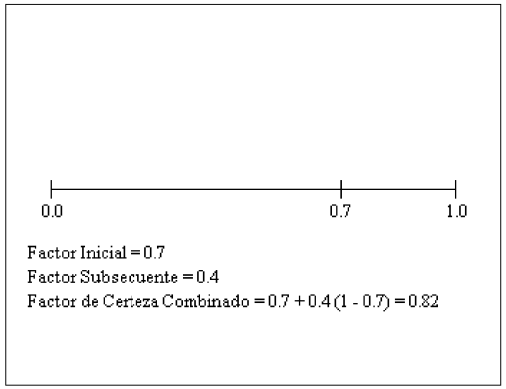

Figura 3.11: Combinación de factores de certeza para un parámetro

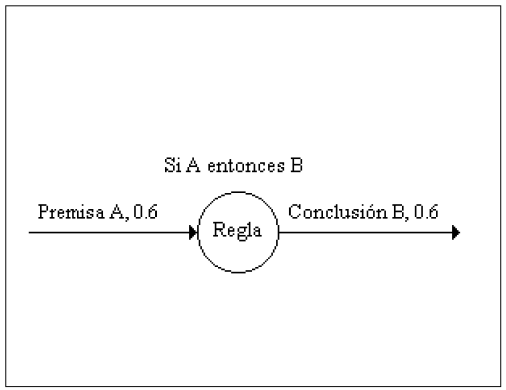

Figura 3.12: Indución del nivel de certeza de la premisa sobre la conclusión

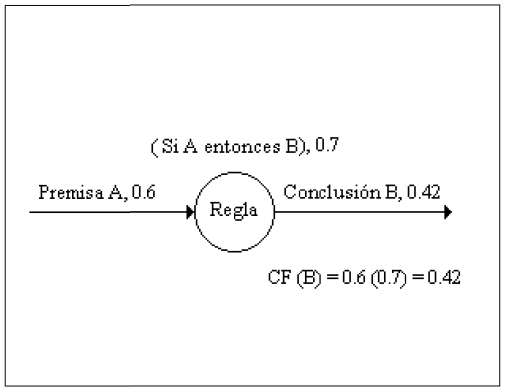

Figura 3.13: Combinación de una premisa incierta con una regla incierta

Ahora bien, un factor de certeza ($CF[h, e]$) se define en términos de dos componentes:

- $MB[h, e]$ - es la medida (entre 0 y 1) de credibilidad en la hipótesis h dada la evidencia e. MB mide el grado en el que la evidencia confirma la hipótesis.
- $MD[h, e]$ - es la medida (entre 0 y 1) de incredulidad de la hipótesis h dada la evidencia e. MD mide el grado en el que la evidencia confirma negativamente la hipótesis.

Con estas dos medidas, el factor de certeza se define como:

$$CF[h, e] = MB[h, e] - MD[h, e] \tag{3.20}$$

Con esto, concluimos la información referente al manejo de incertidumbre e inconsistencias. La elección de alguno de los métodos expuestos (o la combinación de dos o más) dependerá del tipo de problemas que se deseen modelar.

3.5 Metodología para el desarrollo de Sistemas Expertos

Esta metodología surge de la integración de enfoques, técnicas y otras metodologías provenientes de diversas áreas vinculadas con los sistemas basados en conocimiento e Ingeniería de Software. La ventaja del método planteado es que considera la naturaleza de herramienta computacional de los sistemas basados en conocimiento y busca el

mayor aprovechamiento de los recursos disponibles en el área donde se desarrollará el Sistema Experto. La estructuración en etapas, fases y pasos ayuda al proceso de análisis de requerimientos y diseño conceptual. La descripción de la metodología propuesta se presenta a continuación:

Etapa 1: Análisis y descripción del problema.

Determina las características del problema y evalúa la factibilidad de realizar un Sistema Experto. En esta etapa se pretende determinar la naturaleza del problema y los objetivos precisos que indique exactamente cómo se espera que el sistema experto contribuya a la solución de los problemas. Estudia los diferentes recursos con que se cuenta y verifica la posibilidad de su aprovechamiento en las fases de desarrollo e implantación. Existirá una interacción entre el experto humano y el ingeniero de conocimiento. Cuando el experto en el dominio muestre distintos escenarios, el ingeniero del conocimiento desarrollará una "primera"descripción del problema. En caso de que el experto humano no sienta que se representa el problema en su totalidad, entonces el ingeniero reformulará la descripción. Esta actividad continua hasta que ambos estén de acuerdo en la descripción.

- Fase 1.1.- Descripción General del Problema:
  - 1.1.1.- Familiarización con el proceso sobre el cual se desea realizar el Sistema Experto.
  - 1.1.2.- Familiarización con los ambientes computacionales donde se encuentran los datos a ser utilizados.
  - 1.1.3.- Definición detallada del problema que motiva el desarrollo del Sistema Experto.
- Fase 1.2.- Análisis de Factibilidad para el desarrollo del Sistema Experto: En esta fase se estudia si el sistema cumple con las condiciones para realizar un Sistema Experto tomando en cuenta los siguientes criterios:
  - 1.2.1.- La tarea a desarrollar requiere del conocimiento manejado por un experto.
  - 1.2.2.- Disponibilidad del experto o equipo de expertos.
  - 1.2.3.- La experticia es requerida en varios lugares simultaneamente.
  - 1.2.4.- El sistema requiere del manejo de incertidumbre y aplicación de juicios personales.
  - 1.2.5.- Existe un grupo potencial de usuarios.
  - 1.2.6.- Se dispone del tiempo para desarrollar el Sistema Experto.
- Fase 1.3.- Análisis de datos: Verificación de la ubicación y forma de representación de los datos a ser manejados por el sistema experto, considerando el tipo de base de datos (industrial, relacional, orientada a objeto, etc), plataforma computacional (Windows, DOS, UNIX, VMS, etc).

- Fase 1.4.- Elección de la fuente de conocimiento: Es necesario contar con un experto o un grupo de ellos que estén dispuestos a colaborar con el proyecto. Los expertos deben ser reconocidos como tal por el grupo de usuarios.

Etapa 2: Especificación de requerimientos.

En esta etapa se estudia los requerimientos globales del sistema experto a desarrollar, considerando hacia quién estará dirigido el sistema, restricciones de acceso, nivel de detalle requerido en sus explicaciones, requerimientos funcionales, formatos deseados por los usuarios finales. Es importante discutir con los potenciales usuarios del sistema y tomar en cuenta sus aspiraciones y comentarios para garantizar que posteriormente estén dispuestos a utilizar el sistema desarrollado.

- Fase 2.1.- Estimación del perfil de los usuarios finales del Sistema Experto.
- Fase 2.2.- Verificación de los requerimientos con el usuario.
- Fase 2.3.- Determinación de los requerimientos de información: Se especifica la información que debe producir el Sistema Experto y sus atributos tales como el formato de presentación, la frecuencia de salida, sus usuarios directos y su interconexión con otros programas.
- Fase 2.4.- Determinación de los requerimientos funcionales: Consiste en la definición de las funciones generales que debe satisfacer el Sistema Experto.
- Fase 2.5.- Determinación de los requerimientos de entrada de datos:
  - 2.5.1.- Selección de las posibles fuentes de entrada al Sistema Experto.
  - 2.5.2.- Identificación de las fuentes de datos.
  - 2.5.3.- Especificación de los procesos de adquisición de datos.
  - 2.5.4.- Especificación de los procesos de generación de parámetros.
  - 2.5.5.- Caracterización de la interoperabilidad entre las bases de datos que se requieren en la implantación.
- Fase 2.6.- Definición de los requerimientos de hardware y software para la implantación del Sistema Experto:
  - 2.6.1.- Especificación de la plataforma de hardware que se utilizará para el desarrollo y operación del Sistema Experto.
  - 2.6.2.- Determinación, análisis y selección de las herramientas de software disponibles en el mercado para el desarrollo de Sistemas Expertos.

Etapa 3: Análisis de costos, tiempo y recursos.

En esta etapa se realiza un estimado de los costos del desarrollo del sistema experto incluyendo equipos, programas y honorarios profesionales. Igualmente se realiza un cronograma de las actividades a desarrollarse. Generalmente en esta etapa terminaría

el estudio de factibilidad de realizar el sistema experto, ya que hasta esta etapa se ha verificado la necesidad y pertinencia del desarrollo del sistema experto, se ha estudiado los requerimientos y se ha estimado el tiempo, recursos y costos involucrados en el desarrollo e implantación del Sistema.

- Fase 3.1.- Elaboración del plan de actividades de desarrollo e implantación.
- Fase 3.2.- Estimación del tiempo requerido para el desarrollo del Sistema Experto.
- Fase 3.3.- Estimación de los recursos computacionales (hardware-software) requeridos para el desarrollo del Sistema Experto.
- Fase 3.4.- Estimación de los costos de desarrollo.

Etapa 4: Ingeniería del Conocimiento.

En esta etapa se busca designar estructuras para organizar el conocimiento. Después de haber determinado el problema en toda su magnitud, sin haberse referido a técnicas de programación o a indagar solo en los métodos que son exitosos en inteligencia artificial, es en esta etapa donde el ingeniero del conocimiento selecciona estructuras apropiadas a este sistema experto en particular. Es decir, que dan solución total o parcial al problema analizado en las etapas precedentes, una de las responsabilidades principales del ingeniero del conocimiento es analizar situaciones tipo y a partir de ellas extraer las reglas que describan el conocimiento del experto en el dominio

- Fase 4.1.- Adquisición del Conocimiento: Es la parte más importante de un Sistema Experto, ya que es donde el Ingeniero del Conocimiento interactúa con el experto para obtener la información sobre la solución de los problemas, así como las estrategias utilizadas para la obtención de cada solución.
- Fase 4.2.- Estructuración del Conocimiento: En esta fase, el Ingeniero del Conocimiento debe llevar a una base de conocimiento la información proporcionada por el experto. El conocimiento puede ser de carácter superficial o profundo dependiendo de la estructura interna y de las interacciones entre sus componentes.

Etapa 5: Diseño preliminar del Sistema Experto.

En esta etapa se realiza la ingeniería de detalle del Sistema Experto, se diseña cuidadosamente cada una de las módulos que comprenderán la herramienta, se hace la selección final de los componentes a utilizar y se diseña los protocolos necesarios para las interrelaciones con otros programas y/o equipos requeridos.

- Fase 5.1.- Diseño preliminar de la arquitectura del Sistema Experto.
- Fase 5.2.- Selección de la herramienta computacional de acuerdo a los requerimientos surgidos en la etapa de Ingeniería del Conocimiento.

- Fase 5.3.- Diseño preliminar de procesos de adquisición y almacenamiento de datos.
- Fase 5.4.- Diseño preliminar de procesos de interconexión.
  - 5.4.1.- Integración Interna.
  - 5.4.2.- Integración Externa.
  - 5.4.2.- Selección de software auxiliar.
- Fase 5.5.- Verificación del diseño preliminar del Sistema Experto.

Etapa 6: Desarrollo e Implantación del Sistema Experto.

Esta es la etapa final del desarrollo del sistema experto, aquí, se construye, implanta, prueba y depura el sistema. Posterior a la finalización de la implantación comienza la fase de mantenimiento y actualización que durará durante la vida del sistema, ya que se busca mantenerlo operativo en las mejores condiciones e incorporando conocimiento y/o recursos nuevos al sistema según los requerimientos tecnológicos para su vigencia.

- Fase 6.1.- Construcción del prototipo.
- Fase 6.2.- Validación del prototipo.
- Fase 6.3.- Construcción del modelo operacional
- Fase 6.4.- Prueba y depuración: Consiste en plantearle situaciones al Sistema Experto y al experto humano y verificar si ambos generan la misma solución además de seguir las mismas estrategias. En caso de existir discordancia entre el experto humano y el Sistema Experto se procede a la revisión y modificación de la base de conocimiento.
- Fase 6.5.- Mantenimiento y actualización.

3.6 Conclusión

La pregunta sigue cerniéndose sobre nosotros, ¿es realmente inteligente la máquina? Debemos admitir que muestra dos rasgos muy importantes de la inteligencia, la capacidad de aprender y la capacidad de razonar, pero aún así no alcanza el grado de perfección del ser humano, no es consciente del proceso que esta llevando a cabo.

Los computadores son procesadores en serie, avanzan de un punto a otro estando los futuros pasos determinados por los resultados presentes. El cerebro humano además piensa en paralelo, razón por la cual la capacidad humana es hasta ahora inalcanzable. Otra ventaja del humano sobre la máquina es que el humano posee una visión global de lo que nos rodea y el sentido común, mientras la máquina ni siquiera puede acercarse a estos principios.

De todas maneras al comienzo del desarrollo de los programas de inteligencia artificial salieron a la luz todas estas cuestiones, sobre todo la interrogante de que si el programa sabía o no lo que estaba ocurriendo, pero hoy esto ha dejado de ser un problema, lo que importa es que funcionan y facilitan una gran cantidad de tareas al hombre. Luego de la derrota de Kasparov por la computadora Deep Blue, muchos se preguntan cuál será el futuro de la raza humana. ¿Serán las máquina mejores que los hombres y lo reemplazaran en sus tareas?, ¿Que pasará con los desocupados por las máquinas?. Nunca ninguna máquina podrá reemplazar al hombre en cuanto a su creatividad y su gran inteligencia, las máquinas están creadas como herramientas e instrumentos del hombre y siempre deben ser tomadas como eso, herramientas. El fin de toda máquina es servir al hombre y por eso nunca deben estar en detrimento de la calidad de vida del hombre.

Bibliografía

[1] B. G. Buchanan and E. A. Feigenbaum, "Dendral and Meta-Dendral: Their Applications Dimension," *Artificial Intelligence, Vol.11, No.1, pp.5-24, 1978.*

[2] B. G. Buchanan and E. H. Shortlifee, Rule based Expert System, *Addison Wesley, 1984.*

[3] E. A. Feigenbaum, Handbook of Artificial Intelligence, *Heuris Tech Press/William Kaufman, Inc., 1981.*

[4] J. Giarratano and G. Riley, Expert Systems: Principles and Programming, *PWS Publishing Company, 1993.*

[5] P. Harmon and D. King, Expert Systems, *John Wiley, 1985.*

[6] F. Hayes-Roth, D. A. Waterman and D. B. Lenat, Building Expert Systems, *Addison Wesley, 1983.*

[7] R. Keller, Expert system technology: development and applications, *New Jersey, Prentice Hall, 1987*

[8] P. Klahr, Expert Systems: techniques, tools and applications, *Massachusetts, Addison-Wesley, 1986*

[9] J. Luis Maté Hernández, Ingeniería del conocimiento: diseño y construcción de sistemas expertos, *Ed SEPA, 1988.*

[10] F. Rivas-Echeverría, E. Colina Morles and C. Rivas-Echeverría, "Expert Systems Methodology for Management Processes", in Proceedings of IASTED International Conference on Software Engineering, *USA, 1998.*

[11] J. P. Sánchez y Beltrán, Sistemas Expertos: Una Metodología de Programación, *Macrobit, 1990.*

Redes Neuronales Artificiales

Eliezer Colina Morles
Departamento de Sistemas de Control
Facultad de Ingeniería
Universidad de Los Andes
Mérida 5101, Venezuela
E-mail: ecolina@ula.ve

Francklin Rivas Echverría
Departamento de Sistemas de Control
Facultad de Ingeniería
Universidad de Los Andes
Mérida 5101, Venezuela
E-mail: rivas@ula.ve

Capítulo 4

Redes Neuronales Artificiales

4.1 Introducción

Las redes neuronales artificiales constituyen una rama de la Inteligencia Artificial, por medio de la cual se busca emular el funcionamiento de las Redes Neuronales Biológicas en lo relativo al aprendizaje y procesamiento de información. Su desarrollo ha tenido un impacto favorable tanto para el área de la computación y sus aplicaciones tecnológicas, como para otras áreas como la fisiología y neurología, con las cuales se ha creado una interrelación muy provechosa en el uso de modelos del funcionamiento cerebral y en la interpretación de procesos asociados con las capacidades de aprendizaje.

Hoy en día, el campo de aplicación de las redes neuronales se ha visto ampliamente acrecentado, siendo utilizadas en tareas como: Modelado e identificación de sistemas, simulación, control de procesos, predicción, manejo de fallas, reconocimiento de patrones, diagnóstico médico, diseño de sensores virtuales, etc.

Este capítulo, presenta un estudio de las Redes Neuronales Artificiales tanto en su diseño como en algunas de sus aplicaciones.

4.1.1 Redes Neuronales Biológicas

Las Redes Neuronales Biológicas son células nerviosas que constituyen los elementos primordiales del sistema nervioso central. El cerebro humano se distingue por poseer billones de tales células nerviosas (alrededor de 10^{11}) y un mayor número de interconexiones (aproximadamente 10^4 conexiones por cada neurona). El proceso de transmisión de información entre las neuronas es de carácter bioquímico y consiste en la manipulación de niveles de Sodio o Potasio. En general, las neuronas son capaces de recibir y procesar señales provenientes de otras neuronas, generar y conducir pulsos nerviosos y posteriormente retransmitirlos a otras neuronas. Morfológicamente hablando, las neuronas son células que poseen un cuerpo en forma piramidal, esférica o variable de acuerdo a las necesidades fisico-químicas y constan de las siguientes partes:

- **Dendritas:** Son extensiones tubulares de fibras nerviosas que sirven como receptores de los estímulos externos o de la señales emitidas por otras neuronas.
- **Cuerpo Celular:** También llamado "Soma", es donde se almacena todo el contenido recibido por las dendritas. Si el nivel alcanzado por las señales de entrada

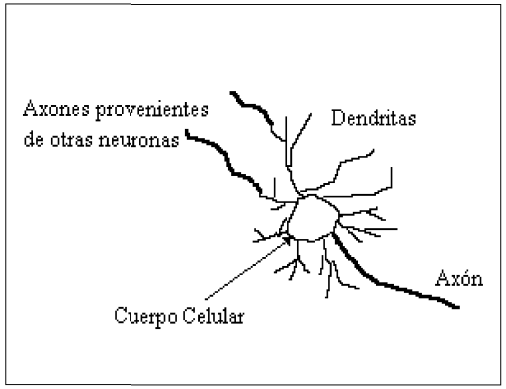

Figura 4.1: Ilustración de una neurona biológica

combinadas es suficientemente grande, entonces la neurona se activará generando una señal de salida (niveles de Sodio o Potasio), la cual será suministrada a las otras neuronas por medio de axón.

- **Axón:** Son estructuras cilíndricas recubiertas por mielina, que contienen en su interior partes de fibras terminales de respuesta de alta intensidad. Su forma es parecida a la de las dendritas, aunque de mayor grosor y su función es la de suministrar a las demás neuronas los niveles de fluido neuro-transmisor en los momentos en que la neurona se encuentra en estado de excitación.

La figura 4.1 ilustra la morfología de una neurona biológica.

A nivel de conocimiento es importante destacar que las neuronas como elementos de procesamiento no guardan información. Es la intensidad de las interconexiones entre las neuronas las que determinan las acciones y asociaciones realizadas por la red neuronal. Estas interconexiones entre las salidas de unas neuronas (Axones) y las entradas de otras neuronas (Dendritas) son llamadas "Sinapsis". Entonces, el proceso de aprendizaje puede ser visto como el ajuste apropiado de las interconexiones sinápticas para guardar el conocimiento y/o asociaciones deseadas a través de la experiencia. Esto explica la relación inversa que existe entre el número de neuronas y el conocimiento que poseen los seres humanos cuando nacen.

4.1.2 Redes Neuronales Artificiales

Las redes neuronales artificiales tratan de emular el comportamiento funcional de las redes biológicas. Para ello, se emplean modelos sencillos del comportamiento neuronal. La figura 4.2 presenta un esquema que contempla la artificialización de

las partes fundamentales de la neurona biológica. En este modelo se puede ver un conjunto de entradas que provienen de otras neuronas o de algún estímulo externo, un conjuntos de pesos que indican las fuerzas de las conexiones sinápticas entre las entradas y las dendritas de la neurona; el cuerpo celular de la neurona cuya función es acumular todas las señales ponderadas recibidas; una función "Γ", llamada función de activación, que determina el cambio de estado inhibitorio a excitatorio de la neurona; y por último, el axón encargado de transmitir las salidas de la neurona activada a otras neuronas.

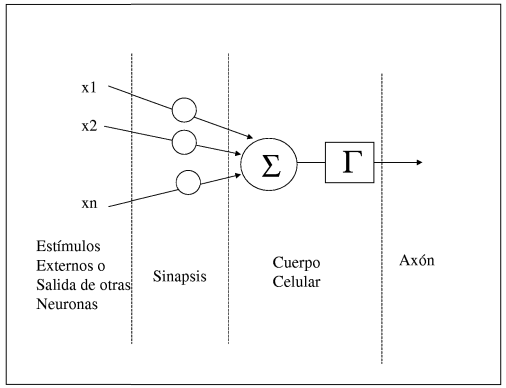

Figura 4.2: Modelo de Neurona Artificial a partir de neurona biológica

Una función de activación sencilla, que permite asociar un estado inhibitorio o excitatorio de la neurona, es la función signo, descrita como:

$$Sign(x) = \begin{cases} 1 & Si\ x \geq 0 \\ -1 & Si\ x < 0 \end{cases} \tag{4.1}$$

En general, la función de activación es una función no lineal de su argumento.

La figura 4.3 ilustra un modelo operacional de una neurona artificial. De allí se pueden obtener las siguientes expresiones:

$$N = \sum_{i=1}^{n} w_i x_i + w_{n+1} b \tag{4.2}$$

$$S = \Gamma(N) \tag{4.3}$$

Nótese que en el modelo artificial ha sido incorporado, por medio del término $w_{n+1}b$, el fenómeno de la predisposición a la inhibición, presente en las neuronas biológicas. Para efectos de aplicación práctica de las Redes Neuronales Artificiales, la predisposición

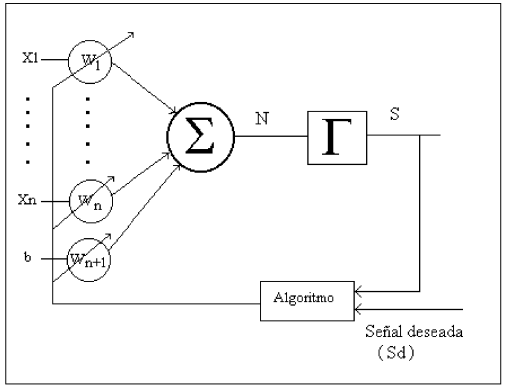

Figura 4.3: Neurona Artificial considerando predisposición a la inhibición

antes mencionada no es un valor conocido de antemano y por ende se deja como un término adicional de ajuste de la red neuronal llamado sesgo, desviación o umbral.

Usando notación vectorial para simplificar las expresiones que describen el modelo matemático neuronal, se puede definir un vector X de entradas ampliadas, -(incluye las entradas y un valor constante que define el sesgo o desviación (b) incorporado para modelar la predisposición a la inhibición de la neurona)-, como:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ b \end{bmatrix}, \tag{4.4}$$

y un vector de pesos ampliados W, que contiene las intensidades de las interconexiones sinápticas ante todas las entradas incluyendo la asociada al valor de sesgo o desviación.

$$W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ w_{n+1} \end{bmatrix} \tag{4.5}$$

De ese modo, el modelo matemático resultante es el siguiente:

$$N = W^T X \tag{4.6}$$

$$S = \Gamma(N) \tag{4.7}$$

Funciones de activación

En la etapa inicial del desarrollo de las redes neuronales artificiales, se utilizó funciones de activación como las funciones de umbral lógico que permitían un funcionamiento neuronal caracterizado por dos estados: inhibitorio y excitatorio. Paulatinamente se fue introduciendo otros tipos de funciones de activación que facilitaron la redefinición de los valores del estado de operación neuronal y permitieron, en algunos casos, el uso de técnicas matemáticas basadas en derivadas para la corrección del error de aprendizaje. Entre estas funciones de activación se encuentran las funciones Sigmoide Unipolar, Sigmoide Bipolar, Tangente Hiperbólica y Lineal. A continuación se describen gráficamente cada una de estas funciones.

- Función Sigmoide Unipolar:

  La expresión matemática de una Sigmoide Unipolar es la siguiente:

  $$\Gamma(x) = \frac{1}{1 + e^{-\lambda x}} \tag{4.8}$$

  donde "λ"es un parámetro que permite moldear la forma de la curva ilustrada en la figura 4.4. Se ve que para valores negativos del argumento x la función tiende al cero, mientras que para valores positivos tiende al uno, emulando los dos estados de las neuronas.

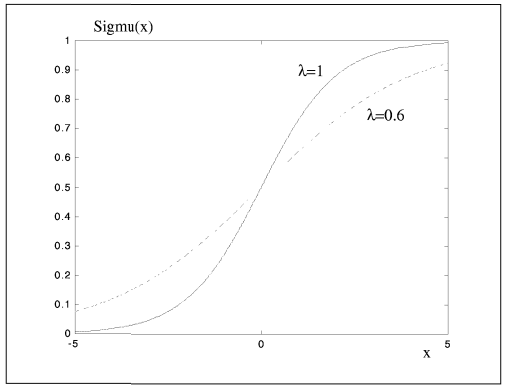

Figura 4.4: Función Sigmoide Unipolar

- Función Sigmoide Bipolar:

  Esta función tiene por expresión matemática la siguiente:

  $$\Gamma(x) = \frac{2}{1 + e^{-\lambda x}} - 1 \tag{4.9}$$

donde λ cumple el mismo rol descrito para la función anterior.

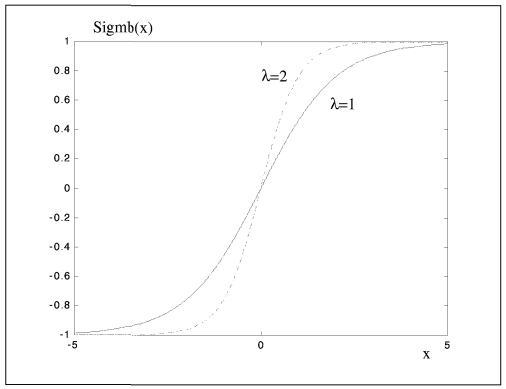

Figura 4.5: Función Sigmoide Bipolar

Su comportamiento puede verse en la figura 4.5. Esta función tiende a -1 para valores negativos del argumento x, mientras que para valores positivos tiende a 1.

- Función Tangente Hiperbólica:

  En este caso, la expresión matemática de la función está descrita por:

  $$\Gamma(x) = Tanh(x) \tag{4.10}$$

  y su comportamiento puede apreciarse en la figura 4.6.

- Función Lineal:

  La función lineal generalmente utilizada es la descrita por la ecuación:

  $$\Gamma(x) = x \tag{4.11}$$

  cuyo comportamiento se ilustra en la figura 4.7. En varias configuraciones neuronales las funciones de activación lineales se emplean en las neuronas de la capa de salida.

4.1.3 Antecedentes Históricos

Una de las tareas científicas que han realizado los neurólogos a lo largo de la historia, es la de estudiar las características "estímulo-respuesta"de las neuronas biológicas

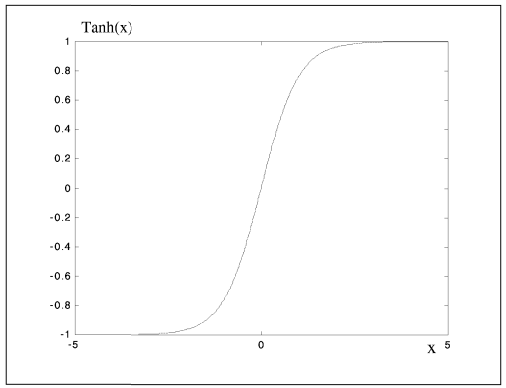

Figura 4.6: Función Tangente Hiperbólica

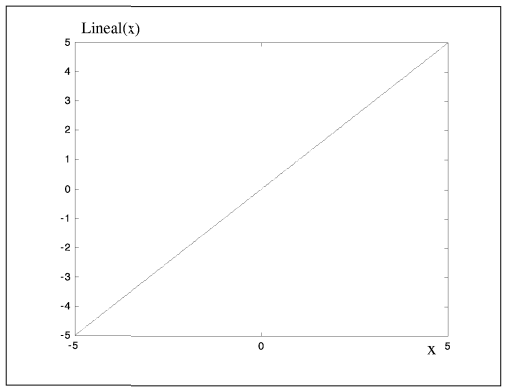

Figura 4.7: Función Lineal

en forma individual y colectiva. Algunos psicólogos por su lado, han estudiado las funciones del cerebro a fin de elaborar modelos de la conducta humana en función del aprendizaje; lo que ha permitido disponer de un enfoque poderoso para evaluar el procesamiento mental. Por otra parte, grupos de científicos de la computación han centrado su esfuerzo en el diseño de novedosos dispositivos de procesamiento paralelo que superen el desempeño de las actuales máquinas convencionales de procesamiento serial.

Es la hibridización de todas estas corrientes científicas, conjuntamente con enfoques de la teoría de sistemas que abarcan áreas del diseño, el desarrollo, la producción y las matemáticas, la que ha dado como resultado emergente la disciplina de las Redes Neuronales Artificiales (RNA) que se conoce hoy en día.

La primera versión de la neurona artificial se le atribuye a W. McCulloch y W. Pitts en 1943 [27], quienes propusieron un modelo simple donde la función Γ de la Figura 4.3 era una función de umbral lógico. Esta neurona artificial inicial era capaz de realizar varias operaciones lógicas sencillas.

En 1948 N. Wiener introduce el término "Cibernética"[39], para describir un área de estudio que abarcaba la teoría de control y la comunicación hombre-máquina.

Un trabajo de D. Hebb, publicado en 1949 [13], establece un mecanismo de aprendizaje según el cual las sinapsis (conexiones) entre neuronas que son activadas al mismo tiempo tienden a fortalecerse; mientras que aquellas que no lo son tienden a debilitarse. Este enfoque del aprendizaje ha tenido un gran impacto en el entrenamiento de ciertas configuraciones topológicas de neuronas artificiales.

Los resultados presentados por McCulloch y Pitts, el entonces reciente enfoque cibernético y el trabajo de Hebb, tuvieron influencia en el trabajo de Rosenblatt, quien en 1957 [29] introdujo un nuevo enfoque para el reconocimiento de patrones usando neuronas artificiales (perceptrones) dotadas con un algoritmo de entrenamiento, que le permitieron resolver problemas de clasificación. Paralelamente al trabajo de Rosenblatt, B. Widrow y su equipo desarrollaron el termino "Adaline"[37] para hacer referencia a una neurona lineal adaptativa, que fue exitosamente utilizada en problemas de procesamiento de señales.

La era inicial de las RNA llegó a su fin con la publicación en 1969 del libro "Perceptrons"[25] de M. Minsky y S. Papert . En este libro los autores demostraron las limitaciones de los perceptrones organizados en una sola capa o nivel para resolver el problema del "O-Exclusivo". Como resultado de esta publicación, la mayoría de los investigadores abandonaron el área en procura de otras alternativas en el campo de la Inteligencia Artificial.

Durante la década de los 70 sin embargo, hubo interesantes trabajos particularmente en el diseño de memorias de contenido direccionable realizados por Kohonen [18]y Anderson [2]. El trabajo de Grossberg [10], sobre el problema general del aprendizaje, es digno de ser mencionado.

A partir de 1980 ocurrió un renacimiento en el interés por estudiar las Redes Neuronales, posiblemente motivado por una mejor comprensión sobre el procesamiento de información en sistemas vivos, los avances en la tecnología de la computación y la

disponibilidad de nuevas teorías de algoritmos adaptativos. El redescubrimiento del algoritmo de retropropagación por D. Parker [28] y Y. LeCum [22] en 1985 y posteriormente divulgado extensamente por D. Rumelhart, G. Hinton y R, Williams [30] en 1986, definitivamente vigorizaron el campo de las RNA al punto de que podemos ahora conseguir en el mercado toda una gama de aplicaciones prácticas que van desde dispositivos para el reconocimiento de patrones y el procesamiento de señales, hasta sistemas especializados para la identificación, el diagnóstico, la detección de fallas, la predicción y el diseño de controladores en sistemas dinámicos de alta complejidas, sensores virtuales, etc.

4.2 Modelos Neuronales

Partiendo de definir a una red neuronal como un ensamblaje de elementos de procesamiento, llamados neuronas, entonces es posible formular una variedad de modelos neuronales atendiendo a características estructurales, como el arreglo y la naturaleza de las interconexiones entre neuronas y a características asociadas a los algoritmos de entrenamiento, como la forma de ponderar las interconexiones neuronales para lograr un objetivo predefinido. Así, en atención a las características estructurales, las redes neuronales pueden ser clasificadas en dos grandes grupos o modelos: Modelos de redes en cascada y modelos de redes retroalimentadas. Por otro lado, dependiendo de la forma de ponderar las interconexiones neuronales, las redes neuronales pueden ser categorizadas en redes entrenadas con supervisión y redes entrenadas sin supervisión.

4.2.1 Clasificación de las Redes Neuronales por su estructura

La estructura de una red neuronal la conforma su esqueleto o configuración topológica. El diseño de varias estructuras neuronales ha sido realizado emulando aquellas que se consiguen en el sistema nervioso alrededor de los centros receptores de los sentidos. Otras estructuras han sido diseñadas para explotar ciertas funcionalidades asociadas a las capacidades de aprendizaje deseables para el arreglo neuronal. En general, atendiendo a las configuraciones topológicas, los modelos neuronales pueden ser clasificados en modelos en cascada y modelos retroalimentados.

Modelos neuronales en cascada

En estas configuraciones las neuronas del arreglo son ordenadas formando capas, cuyas entradas provienen de las salidas de las neuronas de la capa precedente (exceptuando las entradas a las neuronas de la primera capa, que corresponden a los patrones o señales provenientes el fenómeno o proceso bajo consideración), y cuyas salidas van a las neuronas de la capa siguiente. En los modelos de redes en cascada no existen interconexiones entre neuronas pertenecientes a una misma capa ni interconexiones de retroalimentación.

En general, una red neuronal en cascada proporciona una transformación instantánea entre sus entradas y sus salidas; lo cual se traduce en la construcción de mapas estáticos entre un espacio de entrada y uno de salida.

Ejemplos importantes de modelos de redes neuronales en cascada son las redes perceptrónicas multicapas [11], las redes de funciones de base radial [33] y el modelo de control de articulación cerebelar de Albus [1], entre otros. Las Figuras 4.8 y 4.9 ilustran un modelo de red perceptrónica multicapa y un modelo de red de funciones de base radial, respectivamente.

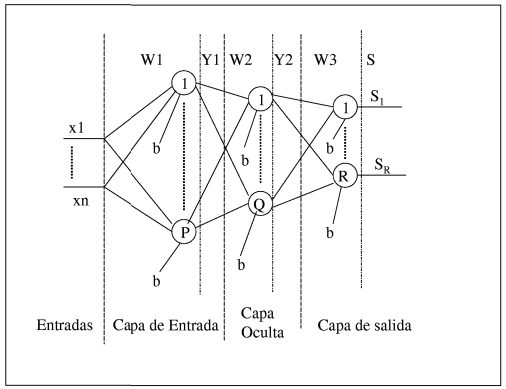

Figura 4.8: Modelo de red perceptrónica multicapas

En la Figura 4.8 las líneas que llegan y las que salen de cada neurona representan los pesos de las interconexiones sinápticas y son denotados por $W1_{ij}$, $W2_{ki}$ y $W3_{rk}$ para referirse a los pesos de las capas de entrada, de la capa oculta y de la capa de salida, respectivamente. Los subíndices identifican los puntos de llegada y de partida de cada peso (p.e. $W1_{ij}$ indica al peso que llega a la neurona i de la capa de entrada y que proviene de la entrada x_j). Note que $W1$, $W2$ y $W3$ son matrices cuyas dimensiones son $P \times (n+1)$, $Q \times (P+1)$ y $R \times (Q+1)$ respectivamente, donde n es el número de entradas, P es el número de neuronas en la primera capa, Q es el número de neuronas en la capa oculta y R es el número de salidas de la red neuronal.

La línea que parte de "b"y llega a cada neurona corresponde al valor de umbral o desviación. Las entradas al modelo corresponden al vector $X = [x1, x2, \ldots, xn, b]^T$. Las salidas de la capa de entrada se denotarán con el vector $Y1 = [y1_1, y1_2, \ldots, y1_P]^T$, las salidas de la capa oculta se denotarán con el vector $Y2 = [y2_1, y2_2, \ldots, y2_Q]^T$ y las salidas de la capa de salida se identifican con el vector $S = [s_1, s_2, \ldots, s_R]^T$.

Note que

$$S = \Gamma_3(W3\ \Gamma_2(W2\ \Gamma_1(W1\ X))) \tag{4.12}$$

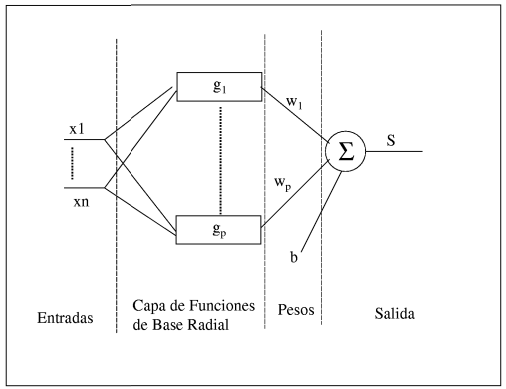

Figura 4.9: Modelo de red de funciones de base radial

donde Γ_1, Γ_2 y Γ_3 son las funciones de activación de la capa de entrada, oculta y de salida respectivamente.

Consideremos ahora la figura 4.9. Sea $X = [x1, x2, \ldots, xn]^T$ el patrón de entrada y sea $W = [w1, w2, \ldots, wp]^T$ el vector de pesos entre las "p"funciones de base radial y la neurona de salida. Se ha asumido que la función de activación de la neurona de salida es lineal. Note que

$$S(X) = \sum_{i=1}^{p} w_i g_i(X) + w_{p+1} b \tag{4.13}$$

En ésta formulación $\{\ g_i(X),\quad i = 1, 2, \ldots, p\}$ representa un conjunto de funciones bases que son linealmente independientes entre si. Por ejemplo

$$g_i(X) = G(\|x - c_i\|),\quad i = 1, 2, \ldots, p \tag{4.14}$$

donde el conjunto de centros c_i, $\quad i = 1, 2, \ldots, p$, deben ser seleccionados apropiadamente.

Modelo Neuronales Retroalimentados

En los modelos retroalimentados las salidas de las neuronas en una capa pueden estar interconectadas a las entradas de las neuronas de la misma capa o a entradas de neuronas en capas precedentes. Este hecho le proporciona al arreglo neuronal características de procesamiento dinámico en el sentido de que las salidas de la red dependen no solo de sus entradas en un instante dado, sino también de sus entradas y salidas en instantes anteriores. Así, las transformaciones de un espacio de entrada

en otro de salida realizadas con modelos de redes retroalimentadas son de carácter dinámico. Dos ejemplos típicos de modelos de redes retroalimentadas son las redes de Hopfield [12] y las redes de Elman [7].

Las figuras 4.10 y 4.11 representan modelos neuronales de Hopfield y Elman respectivamente.

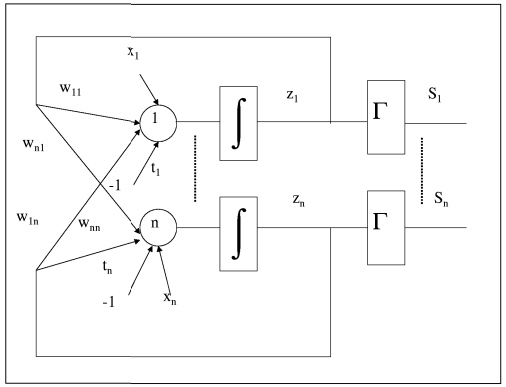

Figura 4.10: Modelo de red de Hopfield

De la figura 4.10 se puede obtener:

$$\frac{dZ}{dt} = \left[\frac{dz_1}{dt}, \frac{dz_2}{dt}, \ldots, \frac{dz_n}{dt}\right]^T \tag{4.15}$$

$$\frac{dz_i}{dt} = \sum_{j=1, j\neq i}^{n} w_{ij} z_j + x_i - t_i \tag{4.16}$$

$$S_i = \Gamma(z_i);\quad i = 1, 2, \ldots, n \tag{4.17}$$

En relación a la figura 4.11, las salidas $S_i(k),\ i = 1, \ldots, q$, de la red de Elman, son agrupadas en el vector $S(k) = [S_1(k), \ldots, S_q(K)]^T$. Las salidas de los nodos ocultos pueden ser representados por el vector $Y2(k) = [y2_1(k), \ldots, y2_p(k)]^T$. En forma análoga, las salidas de los nodos de la primera capa y de los nodos de contexto pueden ser representados por los vectores: $Y1(k) = [y1_1(k), \ldots, y1_n(k)]^T$ y $Y1^C(k) = [y1_1^C(k), \ldots, y1_p^C(k)]^T$, respectivamente.

Las matrices $W1$, $W2$, $W2^C$ y $W3$ representan los valores de interconexión entre los patrones de entrada y los nodos de la primera capa, las salidas de la primera capa y las entradas de la capa oculta, la salida de los nodos de contexto y las entradas

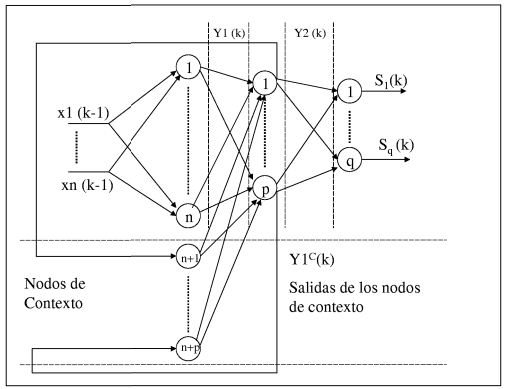

Figura 4.11: Modelo de red de Elman

de la capa oculta y las salidas de la capa oculta y las entradas de la capa de salida, respectivamente.

Nóte que:

$$S(k) = \Gamma(W3\ Y2(k)) \quad (4.18)$$
$$Y2(k) = \Gamma(W2\ Y1(k) + W2^C\ Y1^C(k)) \quad (4.19)$$
$$Y1(k) = \Gamma(W1\ X(k-1)) \quad (4.20)$$
$$Y1^C(k) = Y2(k-1) \quad (4.21)$$

Otros modelos neuronales

Cabe destacar que desde el punto de vista del arreglo neuronal existe una variedad de modelos que son el resultado de combinar en una sola estructura arquitecturas de redes en cascada con arquitecturas de redes retroalimentadas. Una característica distintiva de esos modelos neuronales la constituye sus algoritmos de entrenamiento, particularmente por el uso de un proceso de auto-organización que, aparte de los valores de entrada a la red, toma en cuenta parámetros estadísticos de las mismas. Este tipo de modelos se les denomina redes auto-organizables y ejemplos particulares de éstos son los mapas de Kohonen [18] y las redes de teoría de resonancia adaptativa de Carpenter - Grossberg (ART-1), [4]. Las figuras 4.12 y 4.13 representan un mapa de Kohonen y una red ART-1, respectivamente.

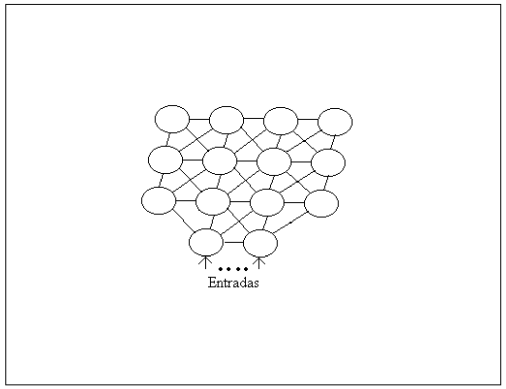

Figura 4.12: Modelo de red auto-organizable de Kohonen

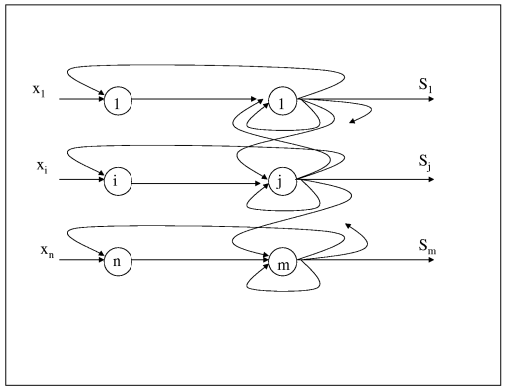

Figura 4.13: Modelo de red de Carpenter-Grossberg (ART-1)

4.2.2 Clasificación de las Redes Neuronales por su entrenamiento

Desde el punto de vista de los algoritmos de entrenamiento, las redes neuronales pueden ser categorizadas entre aquellas que son entrenadas con supervisión y las que no lo son. Esta categorización es válida irrespectivamente de que los patrones de entrenamiento sean señales contínuas o discretas.

Una categoría diferente de redes neuronales puede hacerse para referirse a aquellas que son entrenadas usando reforzamiento [1]; donde en vez de un supervisor, se emplea un "crítico"para evaluar el aprendizaje de la red.

Las figuras 4.14 y 4.15 ilustran las categorías esenciales de redes entrenadas con y sin supervisión, respectivamente.

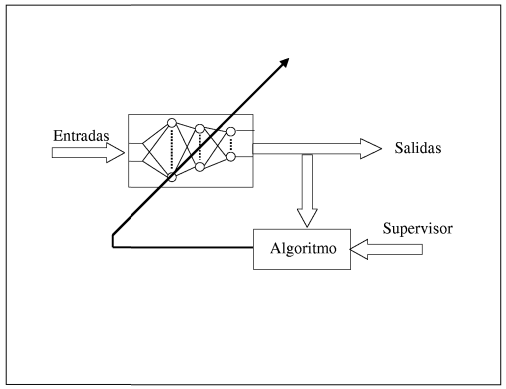

Figura 4.14: Red Neuronal entrenada con supervisión

Los algoritmos de entrenamiento pueden ser agrupados dependiendo de los fundamentos en los cuales basan su construcción a objeto de lograr el aprendizaje de la red neuronal. Por ejemplo, el aprendizaje inspirado en fundamentaciones neuro-biológicas puede ser logrado usando la regla de Hebb o la regla de aprendizaje competitivo [13]. Por otro lado, también es posible forzar el aprendizaje en una red neuronal fundamentándolo en consideraciones matemáticas sobre una función de error [40] o realizando analogías con ideas tomadas de la termodinámica y la teoría de la información, como en el aprendizaje de Boltzmann [14]. La figura 4.16 ilustra una clasificación de los algoritmos de entrenamiento dependiendo de su fundamentación conceptual.

Indiferentemente del ambiente de aprendizaje utilizado y de su fundamentación conceptual, la mayoría de los algoritmos de entrenamiento pueden ser geométricamente interpretados en términos de producir una adaptación de los pesos de interconexión de la red en la dirección del patrón de entrenamiento presentado a la entrada de la

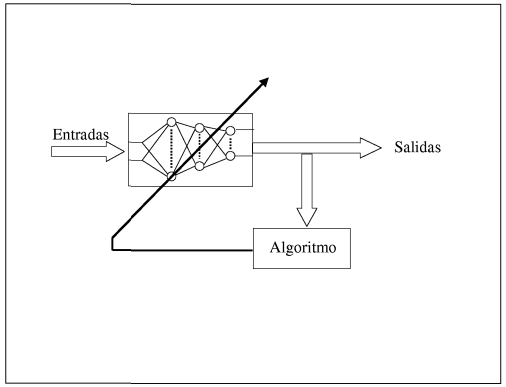

Figura 4.15: Red Neuronal entrenada sin supervisión

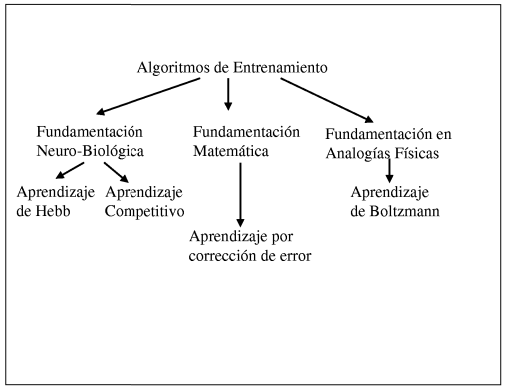

Figura 4.16: Clasificación de los Algoritmos de Entrenamiento basados en su fundamentación conceptual

misma. La figura 4.17 ilustra geométricamente la idea de adaptación de los pesos de interconexión de la red.

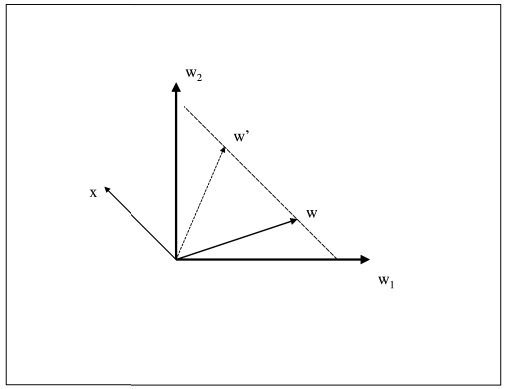

Figura 4.17: Ilustración geométrica de la idea de adaptación de pesos de interconexión

Esta idea de adaptación puede ser formulada en términos matemáticos en la forma siguiente [11], [30]: Sea $W_i = [W_{i1}, W_{i2}, \ldots, W_{in}]^T$ el vector de pesos que llegan a la i-ésima neurona y $X = [x_1, x_2, \ldots, x_n]^T$ el patrón de entrada, entonces el ajuste del vector W_i que debe ser realizado en el k-ésimo paso de entrenamiento puede ser formulado como:

$$\triangle W_i(k) = \alpha f_a(W_i(k), X(k), Sd_i(k))X(k) \tag{4.22}$$

donde $\triangle W_i(k) = W_i(k+1) - W_i(k)$ y α es un parámetro de diseño que determina la tasa de aprendizaje. Por otro lado, $f_a(W_i(k), X(k), Sd_i(k))$ representa una función de aprendizaje que depende de los valores presentes del vector de pesos W_i, del patrón de entrada X y, en caso de existir, de la i-ésima señal de supervisión Sd_i.

La versión en tiempo contínuo de la fórmula (4.22) es la siguiente:

$$\frac{dW_i(t)}{dt} = \alpha f_a(W_i(t), X(t), Sd_i(t))X(t) \tag{4.23}$$

A continuación se incluyen algunos algoritmos de entrenamiento de uso frecuente en el diseño de redes neuronales.

Entrenamiento con Fundamentación Neuro-Biológica y Psicológica

La primera fundamentación biológica para el aprendizaje artificial fué propuesto por Donald Hebb en 1949 [13] y constituye un enfoque auto-organizativo que puede ser

resumido en los términos siguientes: "Los valores de interconexión que parten de la neurona A hacia la neurona B se incrementarán durante el entrenamiento en la medida en la que la neurona B es sensibilizada por la neurona A". Este enfoque original tiene la desventaja de no considerar interconexiones inhibitorias entre las neuronas interactuantes, lo cual significaría la imposibilidad de tener valores negativos para las interconexiones neuronales.

Otro enfoque de inspiración psicológica, es el establecido por Cohen-Grossberg fundamentado en los experimentos de condicionamiento psicológico realizado por Pavlov [10].

Como resultado del enfoque de Grossberg existe un algoritmo de entrenamiento que incluye una corrección al postulado de Hebb para realizar la normalización de los valores de interconexión de la red, forzándolos a crecer o decrecer dentro de ciertos límites.

En forma análoga, el aprendizaje de Cohen-Grossberg es posible implantarlo por medio de los algoritmos del "Vencedor Toma Todo (Instar)"y de "Explosión (Outstar)". A continuación se presenta una descripción de cada uno de los algoritmos mencionados.

Algoritmo de Hebb y Hebb modificado

En relación a la estructura general para los algoritmos de entrenamiento representado por la ecuación (4.22), en el algoritmo de Hebb la función de aprendizaje está definido como:

$$f_a(W_i, X, Sd_i) = f_a(W_i, X) = S_i \tag{4.24}$$

donde S_i es la salida de la i-ésima neurona. De este modo, el incremento de los valores de interconexión de los pesos que llegan a la i-ésima neurona está dado por:

$$\triangle W_i(k) = \alpha S_i(k)X(k) \tag{4.25}$$

Note que la j-ésima interconexión del vector $W_i(k)$ puede expresarse como:

$$\triangle w_{ij}(k) = \alpha S_i(k)x_j(k) \tag{4.26}$$

Una modificación del algoritmo de Hebb propuesta por Grossberg, que permite la normalización de los valores de interconexión en cada presentación de un patrón de entrenamiento es la siguiente:

$$\triangle W_i(k) = -\beta W_i(k) + \alpha S_i(k)X(k) \tag{4.27}$$

o bien, en forma particularizada

$$\triangle w_{ij}(k) = -\beta w_{ij}(k) + \alpha S_i(k)x_j(k) \tag{4.28}$$

donde β es un término de memoria cuyos valores están circunscritos al intervalo (0,1).

La versión a tiempo contínuo del algoritmo modificado de Hebb es la siguiente:

$$\frac{dW_i(t)}{dt} = -\beta W_i(t) + \alpha S_i(t)X(t) \tag{4.29}$$

Algoritmo del Vencedor toma todo

Este algoritmo permite forzar el aprendizaje de características estadísticas de los patrones de entrada a la red. Constituye un tipo de aprendizaje competitivo que puede ser implantado en los términos siguientes:

La función de aprendizaje "f_a"es de la forma

$$f_a(W_i, X, Sd_i) = f(W_i, X) = 1 \tag{4.30}$$

El incremento de los valores de inteconexión que llegan a la i-ésima neurona está dado por:

$$\triangle W_i(k) = -\beta W_i(k) + \alpha X(k) \tag{4.31}$$

o bien en forma particularizada, la j-ésima interconexión está dada por:

$$\triangle w_{ij}(k) = -\beta w_{ij}(k) + \alpha x_j(k) \tag{4.32}$$

La i-ésima neurona es considerada como "ganadora"entre el conjunto de p neuronas del arreglo toda vez que la siguiente condición es satisfecha:

$$W_i^T(k)X(k) = \max_{j=1,2,\ldots,p}(W_j^T(k)X(k)) \tag{4.33}$$

La figura 4.18 ilustra el proceso de competencia para la adaptación de los valores de interconexión de la neurona ganadora.

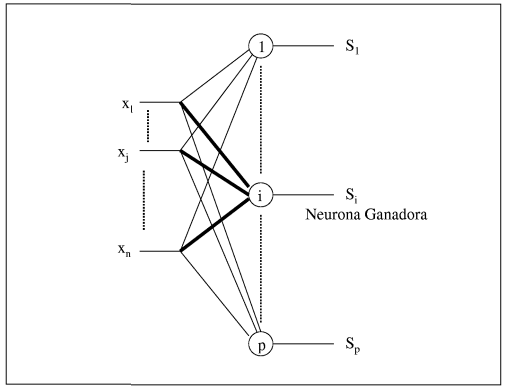

Figura 4.18: Ilustración del algoritmo el Vencedor toma todo

Algoritmo de Explosión (Outstar)

El objetivo en este algoritmo es adaptar los valores de interconexión en la red de modo que sus salidas correspondan a un valor deseado. Esto significa que el algoritmo opera en un ambiente con supervisión. Al finalizar el entrenamiento en forma exitosa, la red podrá extraer características estadísticas de los parámetros de entrenamiento.

En este algoritmo la función de aprendizaje f_a se expresa como:

$$f_a(W_i, X, Sd_i) = Sd_i \tag{4.34}$$

El ajuste de los valores de interconexión es calculado utilizando la formula:

$$\triangle W_i(k) = -\beta W_i(k) + \alpha Sd(k) \tag{4.35}$$

lo que equivale a realizar ajustes de cada componente del vector W_i por medio de la ecuación:

$$\triangle w_{ji}(k) = -\beta w_{ji}(k) + \alpha Sd_j(k) \quad , \quad j = 1, 2, \ldots, p \tag{4.36}$$

Note que el valor de "p"corresponde al número de neuronas en un arreglo como el ilustrado en la figura 4.19 y que los valores de interconexión ajustados son aquellos que parten de la i-ésima componente del vector de entrada.

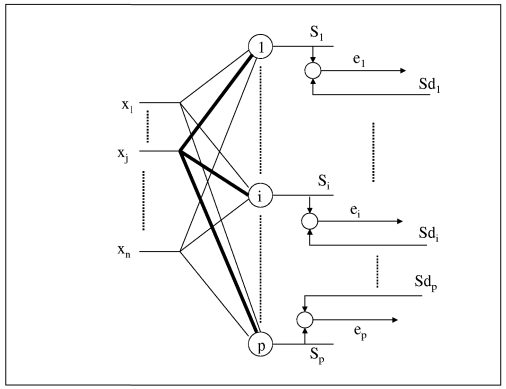

Figura 4.19: Ilustración del algoritmo de Explosión

Entrenamiento con fundamentación matemática

Existe un número de algoritmos de entrenamiento neuronal donde la adaptación de los valores de interconexión de la red es realizada resolviendo un problema de optimización

sobre una función o funcional de error entre la salida deseada (Sd) y la salida del arreglo neuronal (S). Vale decir que estos algoritmos de entrenamiento operan en un ambiente supervisado. Partiendo desde la regla del perceptrón de Rosenblatt [29] y pasando por el algoritmo de Widrow-Hoff [37] hasta llegar a la regla Delta Generalizada, todos estos algoritmos procuran disminuir el error de aprendizaje de la red por adaptación de los valores de interconexión en la presencia de patrones de entrenamiento. En la actualidad han sido reportados en la literatura un sin número de modificaciones y versiones variadas de implantar aprendizaje artificial recurriendo a la idea de retropropagar el error de salida de la red [22], [30],[28],[40].

Las siguientes subsecciones sumarizan los aspectos esenciales de la regla del perceptrón de Rosenblatt, la regla de Widrow-Hoff, la Regla Delta y la Regla Delta Generalizada o Algoritmo de Retropropagación del error.

Perceptrón de Rosenblatt

En el algoritmo para el perceptrón diseñado por Rosenblatt las señales de entrada y de supervisión de la red son de naturaleza bipolar, es decir, sus valores son iguales a 1 o -1. La función de aprendizaje f_a es igual a:

$$f_a(W_i, X, Sd) = \frac{1}{2}(Sd - S) \tag{4.37}$$

y la adaptación de los valores de interconexión se realiza por medio de la ecuación:

$$\triangle W_i(k) = \frac{1}{2}\alpha(Sd(k) - S(k))X(k) \tag{4.38}$$

o correspondientemente:

$$\triangle w_{ij}(k) = \frac{1}{2}\alpha(Sd_i(k) - S_i(k))x_j \quad , \quad j = 1, 2, \ldots, R \tag{4.39}$$

Es de hacer notar que por tratarse de señales bipolares binarias el valor de la función de aprendizaje será igual a ± 2, excepto cuando Sd_i coincide con S_i, cuyo valor será igual a cero (por ende no será necesario realizar cambio en el valor de las interconexiones). Cuando la función de activación "Γ"de la figura 4.3 es igual a "$\Gamma(N) = Sign(N)$"se obtiene el modelo del perceptrón de Rosenblatt.

En la sección 4.3 se desarrolla en detalle los algoritmos para entrenamiento de las redes perceptrónicas por ser las más utilizadas.

Algoritmo de Widrow-Hoff

En el algoritmo de entrenamiento de Widrow-Hoff, la adaptación de los valores de interconexión se realiza resolviendo el problema de minimizar un índice igual al cuadrado de las desviaciones entre la salida deseada "Sd"y la suma lineal de las entradas ponderadas a la neurona. Así, la función de aprendizaje "f_a"queda definida como:

$$f_a(W_i, X, Sd) = Sd - W_i^T X \tag{4.40}$$

mientras que la adaptación de los valores de interconexión se realiza usando la ecuación

$$\triangle W_i(k) = \alpha(Sd_i - W_i^T(k)X(k))X(k) \tag{4.41}$$

En forma individual, cada peso es adaptado por medio de la fórmula

$$\triangle w_{ij}(k) = \alpha(Sd_i - W_i^T(k)X(k))x_j(k) \qquad j = 1, 2, \ldots, R. \tag{4.42}$$

El algoritmo de Widrow Hoff también es conocido como la regla de aprendizaje por mínimos cuadrados.

La Regla Delta

El aprendizaje neuronal usando la regla Delta constituye una generalización del algoritmo de Widrow-Hoff, en el sentido de que la adaptación de los valores de interconexión se realiza resolviendo el problema de minimizar las desviaciones cuadráticas de la salida deseada "Sd_i"y la salida no lineal de la neurona (la salida de la función de activación). De este modo, el índice o funcional de costos a ser minimizado está dado por:

$$J(e(k)) = \frac{1}{2}[Sd_i - \Gamma(W_i^T(k)X(k)]^2 \tag{4.43}$$

La minimización de este índice puede realizarse en forma recursiva utilizando un descenso en la dirección contraria al gradiente de $J(e(k))$ con respecto al vector $W_i(k)$. Esto es:

$$\triangle W_i(k) = -\alpha(\nabla_W J(e(k)) \tag{4.44}$$

En estas circunstancias, la función de aprendizaje "f_a"resultante está dada por:

$$f_a(W_i, X, Sd_i) = -[Sd_i - \Gamma(W_i^T X)]\frac{\partial \Gamma(W_i^T X)}{\partial W_i} \tag{4.45}$$

En definitiva, la adaptación de los valores de interconexión se obtiene por medio de la ecuación

$$\triangle W_i(k) = \alpha[Sd_i - \Gamma(W_i^T X)]\frac{\partial \Gamma(W_i^T X)}{\partial W_i^T X}X(k) \tag{4.46}$$

En forma individualizada, cada peso es actualizado usando la fórmula

$$\triangle w_{ij}(k) = \alpha[Sd_i - \Gamma(W_i^T X)]\frac{\partial \Gamma(W_i^T X)}{\partial W_i^T X}x_j(k) \qquad j = 1, 2, \ldots, R \tag{4.47}$$

La Regla Delta Generalizada

La extensión de la Regla Delta para realizar el entrenamiento de modelos neuronales arreglados en capas, sin retroalimentación entre sus elementos constituyentes, es llamada la Regla Delta Generalizada o Algoritmo de Retropropagación del error. Su fundamentación matemática es la misma de la Regla Delta: Adaptar los valores de interconexión de la red realizando descenso por gradiente sobre un índice cuadrático del error de aprendizaje de la red. El enunciado anterior, significa que, dependiendo del número de capas neuronales, será necesario establecer reglas de adaptación para los valores de interconexión que llegan a cada una de las capas. Así, en un arreglo de dos capas como el ilustrado en la figura 4.20, los valores de interconexión etiquetados con las expresiones Wm_{ij} y Wn_{kl} deberán ser adaptados usando las siguientes ecuaciones:

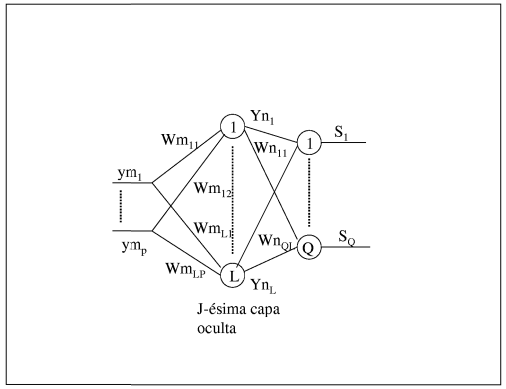

Figura 4.20: Arreglo de dos capas de Neuronas

$$\triangle Wn(k) = -\alpha_1 \frac{\partial J(e)}{\partial Wn(k)} \tag{4.48}$$

$$\triangle Wm(k) = -\alpha_2 \frac{\partial J(e)}{\partial Wm(k)} \tag{4.49}$$

donde

$$J(e) = \frac{1}{2}\sum_{i=1}^{Q}(Sd_i - S_i)^2 \tag{4.50}$$

En la ecuación (4.50) "S_i"representa la i-ésima salida del arreglo neuronal, mientras que "Sd_i"corresponde a la i-ésima señal de supervisión.

Realizadas las operaciones de derivación sugeridas en las ecuaciones (4.48) y (4.49) y tomando en cuenta funciones de activación de tipo sigmoidal, es relativamente sencillo corroborar que las funciones de aprendizaje "$f_a(Wn, Yn, Sd)$"y "$f_a(Wm, Ym, Sd)$"son los siguientes:

$$f_a(Wn, Yn, Sd) = \frac{1}{2}(Sd_k - S_k)(1 - S_k^2) \quad , \quad k = 1, 2, \ldots, Q \tag{4.51}$$

$$f_a(Wm, Ym, Sd) = \frac{1}{2}(1 - ym_j^2)\sum_{i=1}^{Q} f_a(Wn, Yn, Sd)Wn_{ij} \quad , \quad j = 1, 2, \ldots, L \tag{4.52}$$

En definitiva, las ecuaciones para la adaptación de los valores de interconexión resultantes son las siguientes:

$$\triangle Wn(k) = \alpha_1 f_a(Wn, Yn, Sd)Yn^T(k) \tag{4.53}$$

$$\triangle Wm(k) = \alpha_2 f_a(Wm, Ym, Sd)Ym^T(k) \tag{4.54}$$

Los valores de α_1 y α_2, que definen la tasa de aprendizaje, deben ser seleccionados pertenecientes al intervalo (0,1).

Entrenamiento con Fundamentación Física

En el entrenamiento con fundamentación en leyes físicas, se realiza una analogía entre un fenómeno físico y el algoritmo de adaptación de los valores de interconexión de la red. Por ejemplo, en el estudio de la física de la materia condensada [12], el uso de termodinámica estadística es utilizada como herramienta para el análisis estadístico de átomos en muestras de materias líquidas o sólidas. Este análisis permite establecer el comportamiento más probable del sistema de átomos en equilibrio térmico, a una temperatura dada. Según fue reconocido por Von Newmann [35], existen indicativos de que la información es similar a la entropía y que los procesos degenerativos de la entropía son paralelos a los procesos degenerativos en el procesamiento de la información.

Tanto la ecuación de distribución de Boltzmann [14], como la ecuación de Fokker-Planck para el estudio del movimiento Browniano de partículas [12] han servido de bases para la formulación de algoritmos de entrenamiento neuronal. A continuación se presenta una breve descripción del algoritmo de entrenamiento de Boltzmann.

Algoritmo de Entrenamiento de Boltzmann

La figura 4.21 ilustra el arreglo neuronal correspondiente a una máquina de Boltzmann que posee nodos de entrada, ocultos y de salida. En estas máquinas el estado de sus neuronas es caracterizado en forma binaria bipolar, (1 o -1) y sus valores de interconexión entre neuronas son siempre simétricos. En su entrenamiento, el estado

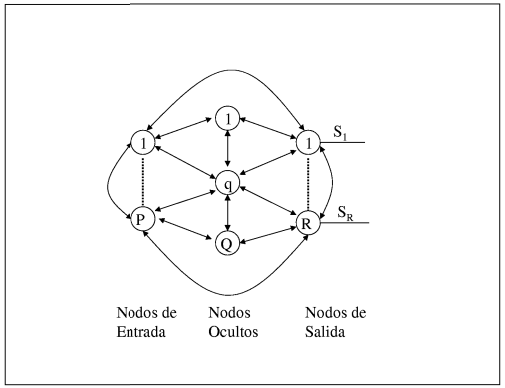

Figura 4.21: Máquina de Boltzmann

de la j-ésima neurona (e_j), seleccionada aleatoriamente es conmutado (p.e. de 1 a -1) a una temperatura "T"con una probabilidad dada por:

$$Prob(e_j \ conmute \ a \ -e_j) = \frac{1}{1 + exp(-\triangle E_j/T)} \tag{4.55}$$

donde "$\triangle E_j$"es el cambio de energía requerido pra realizar la conmutación. La función de energía "E"utilizada es la siguiente

$$E = -\frac{1}{2}\sum_i \sum_{j \ i \neq j} w_{ji} e_j e_i \tag{4.56}$$

donde "w_{ji}"representa el valor de interconexión entre la neurona i y la neurona j.

Las neuronas de una máquina de Boltzmann pueden ser divididas en dos grupos funcionales:

- Las neuronas visibles, que proveen una interfaz entre la red y el ambiente.
- Las neuronas ocultas, las cuales operan en forma libre.

Las condiciones de operación de las máquinas de Boltzmann son las siguientes:

- **Condición de enclave**, donde el estado de las neuronas visibles se encuentra "enclavado"en un modo específico determinado por el ambiente.
- **Condición de operación libre**, en la cual todas las neuronas de la máquina operan libremente.

Denotando por "ρ^*_{ji}"la correlación entre los estados de las neuronas i y j, condicionado a que la red se encuentre en una condición de "enclave"y por "ρ_{ji}"la correlación no condicionada entre los estados de las neuronas i y j; entonces la adaptación del valor de interconexión entre la neurona i y la neurona j se realiza por medio de la fórmula:

$$\triangle w_{ji} = \alpha(\rho^*_{ji} - \rho_{ji}); \quad i \neq j \tag{4.57}$$

donde α es la tasa de aprendizaje y las expresiones para evaluar "ρ^*_{ji}"y "ρ_{ji}"son las siguientes:

$$\rho^*_{ji} = \sum_\gamma \sum_\sigma P^*_{\gamma\sigma} e_{j/\gamma\sigma} e_{i/\gamma\sigma} \tag{4.58}$$

$$\rho_{ji} = \sum_\gamma \sum_\sigma P_{\gamma\sigma} e_{j/\gamma\sigma} e_{i/\gamma\sigma} \tag{4.59}$$

donde "$P^*_{\gamma\sigma}$"es la probabilidad condicional de que las neuronas visibles se encuentren en el estado "γ"y que las neuronas ocultas se encuentren conjuntamente en el estado "σ"dado que la máquina se encuentre en una condición de enclave, y "$P_{\gamma\sigma}$"es la probabilidad condicional de que las neuronas visibles se encuentren en el estado "γ"y las neuronas ocultas se encuentren en el estado "σ"dado que la máquina se encuentra operando libremente. "$e_{i/\gamma\sigma}$"representa el estado de la neurona i dado que las neuronas visibles de la máquina están en un estado "γ"y las neuronas ocultas están en estado "σ".

4.3 Clasificación de patrones usando Redes Neuronales

Una de las principales áreas donde se han usado las redes neuronales es en el reconocimiento y clasificación de patrones. Un patrón es la descripción cuantitativa de un objeto, evento o fenómeno. La clasificación puede involucrar patrones espaciales o temporales.

Ejemplos de patrones espaciales son: cuadros, imágenes de video, mapas, huellas digitales, letras, etc. Ejemplos de patrones temporales incluyen señales de voz, electrocardiogramas, electroencefalogramas, sismogramas, algunas señales generadas por sensores de procesos, etc. Los patrones temporales generalmente involucran secuencias ordenadas de datos en el tiempo.

El objetivo de la clasificación de patrones es el asignar el patrón presentado al sistema a una de las clases o categorías pre-especificadas.

Una de las definiciones más amplias de clasificación de patrones se puede plantear en los siguientes términos: "El problema de clasificación de patrones puede ser visto como la discriminación de datos de entrada que pertenecen a una población objeto por la vía de buscar atributos invariantes dentro de los miembros de la población".

EL "Reconocimiento de patrones"consiste en la asignación de una clase para patrones de entrada que no son idénticos a los patrones usados para el entrenamiento del clasificador.

4.3.1 Reconocimiento de patrones linealmente separables (Redes Perceptrónicas Discretas)

El caso más sencillo de reconocimiento de patrones se presenta cuando la ubicación espacial de los datos que pertenecen a cada una de las diferentes categorías se pueden separar utilizando superficies planas. En éste caso se habla de patrones linealmente separables, los cuales deben cumplir con las siguientes condiciones algebráicas:

Si $\exists\ W \in \Re^n / W^T Y > 0\ \forall\ Y \in$ *clase 1 y* $W^T Y < 0\ \forall\ Y \in$ *clase 2, entonces las clases 1 y 2 de patrones son linealmente separables. En caso contrario serán no linealmente separables.*

Para la resolución de éste tipo de problemas se suele utilizar las funciones discriminantes [2], [17],[23] las cuales son funciones matemáticas que describen las superficies planas y son generadas a partir de un conjunto de observaciones realizadas sobre datos que pertenecen a las diferentes categorías.

Las redes neuronales discretas pueden ser utilizadas para resolver problemas de clasificación de patrones linealmente separables.

Perceptrón Discreto

Sea el problema de clasificar correctamente un conjunto de elementos que pertenecen a dos clases y considere el modelo de una neurona discreta (Ver Figura 4.3) con función de activación *Signo*. Note que las siguientes expresiones son válidas:

$$N = x_1 * w_1 + x_2 * w_2 + \ldots + x_n * w_n + b * w_{n+1} \quad (4.60)$$

$$S = Signo(N) = Signo(x_1 * w_1 + x_2 * w_2 + \ldots + x_n * w_n + b * w_{n+1}) \quad (4.61)$$

El argumento de la función *Signo* (la suma de entradas ponderadas) constituye una ecuación de una superficie en el espacio n-dimensional, que puede ser tomada como función de decisión; donde los elementos que se encuentren por arriba de la superficie formarán parte de una de las clases y los elementos por debajo de la superficie de decisión pertenecerán a la otra clase. Utilizando un solo perceptrón discreto el número máximo de clases que se pueden discriminar es dos. Esto es debido a que la salida de una neurona discreta solo posee dos estados, 1 o -1.

En la figura 4.22 se puede observar patrones linealmente separables que pertenecen a dos posibles clases. En éste caso, basta con una línea recta como superficie de decisión, para determinar si un patrón cualquiera pertenece a alguna de las dos clases. Para efectos de utilizar un perceptrón discreto en éste problema de clasificación, se debe en primera instancia seleccionar el código numérico asociado a cada clase, es decir, a cada una de las dos clases asignarle una de las posibles salidas de la función signo. Por ejemplo, los patrones pertenecientes a la clase mostrada con asteriscos($*$), se les asignará como salida deseada un 1 y a los patrones de la clase ilustrada por círculos ($\circ$) se le asignará como salida deseada el valor -1.

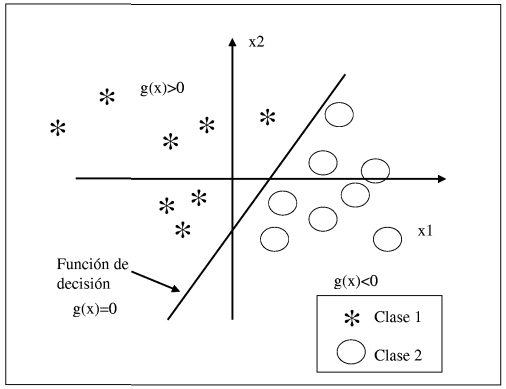

Figura 4.22: Clasificación de dos patrones linealmente separables

4.3.2 Algoritmo de entrenamiento para el perceptrón discreto

Cuando se utiliza un perceptrón discreto para realizar clasificación entre dos clases, pueden ocurrir las siguientes situaciones.

1. Que el patrón pertenezca a la clase 1 (Salida deseada 1) y que el perceptrón genere como salida clase 1 (Salida Neuronal 1).

2. Que el patrón pertenezca a la clase 1 (Salida deseada 1) y que el perceptrón genere como salida clase 2 (Salida Neuronal -1).

3. Que el patrón pertenezca a la clase 2 (Salida deseada -1) y que el perceptrón genere como salida clase 1 (Salida Neuronal 1).

4. Que el patrón pertenezca a la clase 2 (Salida deseada -1) y que el perceptrón genere como salida clase 2 (Salida Neuronal -1).

El error de clasificación está dado por la diferencia entre la salida deseada y la salida neuronal ($e = Sd - S$). Así que a nivel de error puede ocurrir que sea cero en los casos 1 y 4, que sea 2 en el caso 2 o que sea -2 en el caso 3.

Para efectos de esta explicación, se asumirá que el espacio aumentado de los patrones de entrenamiento, Y, lo conforma el patrón de entrada "X"mas el elemento de desviación "b". Esto es $Y = [X\ b]^T$. Así, el vector aumentado de los pesos es de la forma $W = [w_1, \ldots, w_{n+1}]^T$. La figura 4.23 ilustra la superficie de decisión para un patrón de entrenamiento Y en el espacio aumentado de los pesos.

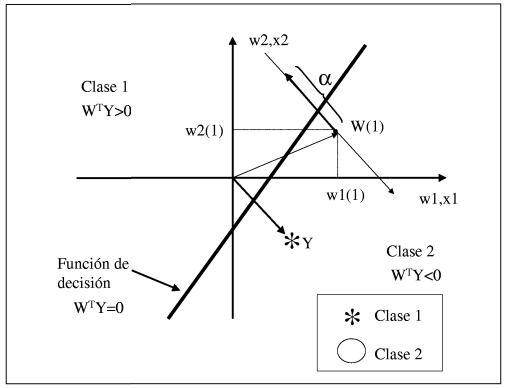

Figura 4.23: Superficie de decisión para dos clases linealmente separables

Asumamos que el valor inicial de los pesos es $W(1)$ y que el patrón Y pertenece a la clase 1 (codificada con salida deseada 1). En la figura se puede observar que el patrón está siendo mal clasificado, ya que se encuentra en la región del espacio que la superficie de decisión determina para la clase 2 (codificada con salida deseada -1). Es decir, que con el valor inicial escogido para los pesos $W(1)$ resultaría una función discriminante con valor negativo ($(W(1)^T Y_1 < 0$). A fin de modificar convenientemente esta situación, los pesos de la neurona deberán ser adaptados de modo que la nueva función discriminante ubique el patrón seleccionado en la región del espacio que le corresponde. Para ello, se ajusta los pesos en la dirección de máximo aumento, es decir, en la dirección del gradiente de $W^T Y$ con respecto a W. Esto es:

$$\nabla_W (W^T(K) Y(K)) = Y(K) \tag{4.62}$$

De modo que cuando el patrón de la clase 1 sea mal clasificado, el ajuste de los pesos debe realizarse según la siguiente expresión:

$$W(K+1) = W(K) + \alpha Y(K) \tag{4.63}$$

donde la constante $\alpha > 0$ es llamada el factor de aprendizaje o incremento de corrección y $W(K+1)$ representa el nuevo vector de pesos despúes de la corrección realizada en la iteración K.

Consideremos a continuación el otro posible error de clasificación ilustrado en la figura 4.24. En este caso nuevamente existe una mala clasificación del patron $Y(K)$, ya que inicialmente $W^T(1)Y(K) > 0$ (lo cual corresponde a patrones pertenecientes a la clase 1), así, que nuevamente se debe modificar el valor de la función discriminante por medio de la corrección de los pesos. Para ello nuevamente se utiliza la técnica del

máximo descenso que se corresponde con la dirección negativa del patrón Y. Es decir, los pesos deben ser ajustados según:

$$W(K+1) = W(K) - \alpha Y(K) \tag{4.64}$$

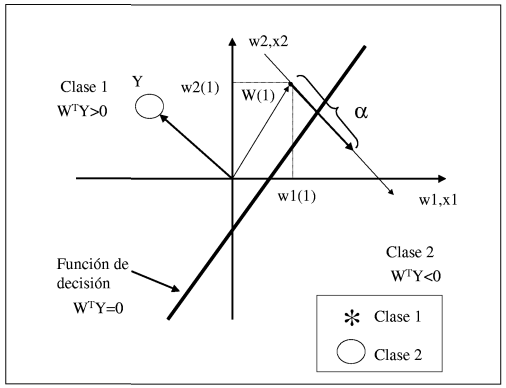

Figura 4.24: Error de clasificación para la clase 2

En resumen, para los casos donde ocurre una mala clasificación del patrón de entrada presentado ($Y(K)$), el siguiente procedimiento de adaptación de pesos (entrenamiento) debe realizarse:

- Incrementar el valor de los pesos W por $\alpha Y(K)$ si el patrón de la clase 1 no es clasificado apropiadamente.
- Decrementar el valor de los pesos W por $\alpha Y(K)$ si el patrón de la clase 2 no es clasificado apropiadamente.

Considerando los errores que se obtienen en cada caso de error de clasificación, el algoritmo de entrenamiento se puede sintetizar en una sola expresión de la forma:

$$W(K+1) = W(K) + \frac{1}{2}\alpha e(K) Y(K) \tag{4.65}$$

Ejemplo

A objeto de ilustrar la operación del algoritmo de entrenamiento para un perceptrón discreto, se considerará la emulación de operador "O"lógico. Este operador posee dos entradas binarias (x_1, x_2) y proporciona una salida binaria según se muestra en la tabla 1.

x_1	x_2	x_1 OR x_2
-1	-1	-1
-1	1	1
1	-1	1
1	1	1

Tabla 1. Operador "O"lógico

Aprecie que la emulación de este operador lógico es equivalente a resolver el problema de clasificar correctamente, en una de las dos clases (1 o -1), las cuatro combinaciones posibles de los valores de los patrones de entrada (problema del dicotomizador).

La figura 4.25 describe gráficamente el problema de clasificación considerado y pone en relieve que existe una infinita cantidad de superficies de decisión que pueden ser utilizadas para resolverlo.

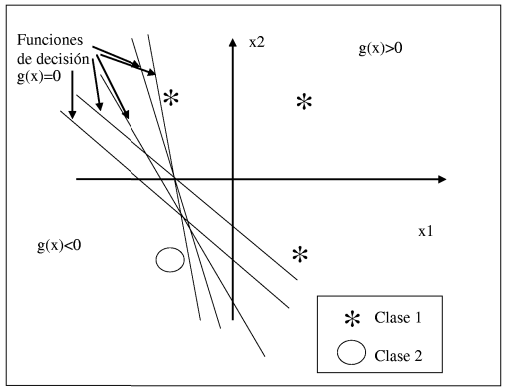

Figura 4.25: Espacio de soluciones para el operador lógico "O"

Para resolver este problema de clasificación, se procederá a entrenar un perceptrón discreto, que además de las dos entradas x_1 y x_2, posea una entrada de desviación igual a 1. Así, el entrenamiento del perceptrón discreto permitirá obtener una función de decisión que proporcione una solución apropiada al problema.

La figura 4.26 ilustra al perceptrón discreto usado en este problema. Para efectos del algoritmo de entrenamiento, se asumirá que el valor de la tasa de aprendizaje es igual a 1 ($\alpha = 1$), y que el valor inicial de los pesos del perceptrón están dados como $W(1) = [1\ 2\ 4]^T$. Finalmente, note que la señal deseada "Sd"tiene como valores los que aparecen en la última columna de la tabla 1 (x_1 o x_2).

La tabla 2 incluye los cuatro patrones posibles con sus correspondientes valores deseados.

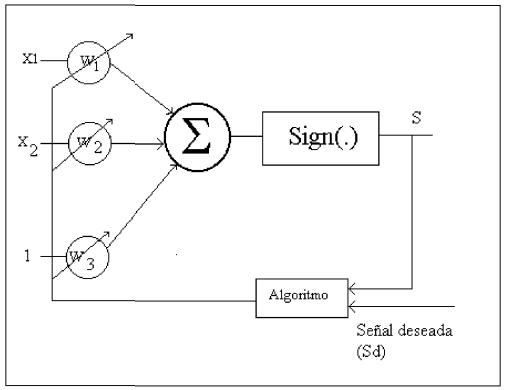

Figura 4.26: Perceptrón discreto usado para resolver el problema del operador lógico "O"

	x_1	x_2	b	Sd
Patrón 1	-1	-1	1	-1
Patrón 2	-1	1	1	1
Patrón 3	1	-1	1	1
Patrón 4	1	1	1	1

Tabla 2. Patrones de entrenamiento con sus correspondientes salidas deseadas

El proceso de entrenamiento es descrito a continuación:

Paso 1: Presentación del patrón 1

$$Y(1) = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \quad , \quad Sd(1) = -1 \tag{4.66}$$

La salida del perceptrón discreto $S(1)$ es entonces:

$$S(1) = Sign(W^T(1)Y(1)) = Sign\left([1\ 2\ 4]\begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}\right) = Sign(-1-2+4) = Sign(1) = 1 \tag{4.67}$$

El correspondiente error de clasificación resulta ser

$$e(1) = Sd(1) - S(1) = -1 - 1 = -2 \tag{4.68}$$

Los pesos del perceptrón deben ser actualizados por medio del algoritmo:

$$W(2) = W(1) - Y(1) => W(2) = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix} - 1 \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 3 \end{bmatrix} \tag{4.69}$$

Paso 2: Presentación del patrón 2

$$Y(2) = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} \quad , \quad Sd(2) = 1 \tag{4.70}$$

La salida del perceptrón discreto $S(2)$ es la siguiente:

$$S(2) = Sign(W^T(2)Y(2)) = Sign\left([2\ 3\ 3] \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} \right) = Sign(-2+3+3) = Sign(4) = 1 \tag{4.71}$$

El error de clasificación para este patrón es:

$$e(2) = Sd(2) - S(2) = 1 - 1 = 0 \tag{4.72}$$

En vista de que el error resultante fue igual a cero, los pesos del perceptrón discreto permanecerán inalterados. Esto es:

$$W(3) = W(2) = \begin{bmatrix} 2 \\ 3 \\ 3 \end{bmatrix} \tag{4.73}$$

Paso 3: Presentación del patrón 3

$$Y(3) = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \quad , \quad Sd(3) = 1 \tag{4.74}$$

La salida del perceptrón discreto $S(3)$ resulta ser:

$$S(3) = Sign(W^T(3)Y(3)) = Sign\left([2\ 3\ 3] \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \right) = Sign(2-3+3) = Sign(2) = 1 \tag{4.75}$$

El error de clasificación es entonces:

$$e(3) = Sd(3) - S(3) = 1 - 1 = 0 \tag{4.76}$$

Nuevamente, los pesos del perceptrón permanecerán iguales con la presentación de este patrón:

$$W(4) = W(3) = \begin{bmatrix} 2 \\ 3 \\ 3 \end{bmatrix} \tag{4.77}$$

Paso 4: Presentación del patrón 4

$$Y(4) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad , \quad Sd(4) = 1 \tag{4.78}$$

La salida resultante del perceptrón discreto $S(4)$ está dada por:

$$S(4) = Sign(W^T(4)Y(4)) = Sign\left([2\ 3\ 3] \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right) = Sign(2+3+3) = Sign(8) = 1 \tag{4.79}$$

El error de clasificación correspondiente es:

$$e(4) = Sd(4) - S(4) = 1 - 1 = 0 \tag{4.80}$$

De modo que al finalizar el primer ciclo de presentación de patrones, los valores de los pesos de interconexión del perceptrón resultan ser:

$$W(5) = W(4) = \begin{bmatrix} 2 \\ 3 \\ 3 \end{bmatrix} \tag{4.81}$$

Fin del primer ciclo de entrenamiento.

A partir de este momento, resulta sencillo verificar que las presentaciones sucesivas de los patrones de entrenamiento (p.e. denominando $Y(5) = Y(1)$, $Y(6) = Y(2)$, $Y(7) = Y(3)$, $Y(8) = Y(4)$) proporcionarán un error de clasificación nulo. Esto permite concluir que la línea descrita por la ecuación

$$g(x_1, x_2) = 2x_1 + 3x_2 + 3 = 0 \tag{4.82}$$

representa una función de decisión implantada por el perceptrón discreto y que permite resolver el problema de clasificación en forma satisfactoria. Note que los parámetros de la función de decisión corresponden a los valores de los pesos, encontrados a través del algoritmo de entrenamiento, del perceptrón discreto.

Resumen del algoritmo de entrenamiento para el perceptrón discreto

Dados "P"patrones de entrenamiento

$[X(1), Sd(1); X(2), Sd(2); \ldots; X(P), Sd(P)]$ donde $X(i) \in \Re^n$ y $Sd(i) \in \Re$; $i = 1, 2, \ldots, P$.

Tenemos que el vector de entrada aumentado está dado por:

$$Y(i) = \begin{bmatrix} X(i) \\ 1 \end{bmatrix} \tag{4.83}$$

Se denotará por "Kª"l paso de entrenamiento y por "pª un contador que lleva el número de patrones evaluados en cada ciclo.

1. Seleccionar el factor de aprendizaje $\alpha > 0$
2. Inicializar los pesos W con valores aleatorios pequeños. Recordar que $W \in \Re^{n+1}$.
3. Inicializar los contadores y una variable de acumulación del error. $K \leftarrow 1$; $p \leftarrow 1$; $E \leftarrow 0$.
4. Ciclo de entrenamiento. Se presenta las entradas y se calcula la salida del perceptrón.
   - $Y \leftarrow Y(p)$
   - $Sd \leftarrow Sd(p)$
   - $S = Sign(W^T Y)$
5. Actualización de los pesos
   - $W \leftarrow W + \frac{1}{2}\alpha(Sd - S)Y$
6. Calcular el error del ciclo
   - $E \leftarrow E + \frac{1}{2}(Sd - S)^2$
7. Si p=P entonces ir al paso 8, de lo contrario:
   - $p \leftarrow p + 1$
   - Ir al paso 4.
8. Fin del ciclo de entrenamiento.

   Si $E = 0$ entonces Fin de la sesión de entrenamiento. Se obtiene los valores de los pesos (W) y del número de ciclos (K), de lo contrario:
   - $E \leftarrow 0$
   - $p \leftarrow 1$
   - $K \leftarrow K + 1$
   - Ir al paso 4.

4.3.3 Clasificación usando el perceptrón Contínuo

La principal diferencia entre el perceptrón discreto y contínuo radica en que la función de activación de éste último no es de carácter binario sino que utiliza una función contínua y derivable, permitiendo el uso de técnicas basadas en el descenso por gradiente sobre una función o funcional de error y empleando las derivadas de las funciones de activación. De hecho, el problema de adaptación de los pesos es resuelto por medio de la minimización de una funcional de costo escalar, usando procedimientos de máximo descenso por técnicas de gradientes.

El procedimiento de descenso por gradiente consiste en lo siguiente: A partir de un vector de pesos (W) seleccionado arbitrariamente, se calcula el gradiente en la dirección de los pesos de la funcional de costo utilizada $(\nabla_W J(E(W)))$. El ajuste de los pesos se realiza moviéndose en la dirección negativa del gradiente calculado sobre la superficie de error multidimensional. Esta dirección negativa del gradiente será la que genere el máximo descenso. El algoritmo para la actualización de los pesos puede ser expresado de la siguiente manera:

$$W(K+1) = W(K) - \alpha \nabla_W J(E) \tag{4.84}$$

donde "α"es una constante positiva llamada el factor o tasa de aprendizaje.

Definamos la funcional de costo en el K-ésimo paso de entrenamiento como la diferencia cuadrática media entre el valor de la salida deseada (Sd) y la salida generada por el perceptrón contínuo (S).

La funcional a ser minimizada es entonces:

$$J(E) = \frac{1}{2}(Sd - S)^2 \tag{4.85}$$

Sustituyendo la salida neuronal por su expresión, se obtiene:

$$J(E) = \frac{1}{2}[Sd - f(W^T Y)]^2 \tag{4.86}$$

El objetivo del entrenamiento consiste en lograr la minimización de la funcional $J(E(W))$ en el espacio $(n+1)$-dimensional de los pesos.

Se puede observar que el gradiente de la funcional de costo en la dirección de W puede ser expresado como:

$$\nabla_W J(E) = \frac{1}{2}\nabla_W [Sd - f(W^T Y)]^2 \tag{4.87}$$

El vector gradiente, $(n+1)$-dimensional, es definido como:

$$\nabla_W J(E) = \begin{bmatrix} \frac{\partial J(E)}{\partial W_1} \\ \frac{\partial J(E)}{\partial W_2} \\ \vdots \\ \frac{\partial J(E)}{\partial W_{n+1}} \end{bmatrix} \tag{4.88}$$

Utilizando la regla de la cadena para el cálculo de las derivadas se obtiene:

$$\nabla_W J(E) = -(Sd - S) f'(W^T Y) \begin{bmatrix} \frac{\partial (W^T Y)}{\partial W_1} \\ \frac{\partial (W^T Y)}{\partial W_2} \\ \vdots \\ \frac{\partial (W^T Y)}{\partial W_{n+1}} \end{bmatrix} \tag{4.89}$$

Note que:

$$\frac{\partial(W^T Y)}{\partial w_i} = x_i, \quad i = 1, 2, \ldots, n+1 \tag{4.90}$$

Así, el gradiente de la funcional de costo es de la forma:

$$\nabla_W J(E) = -(Sd - S) f'(W^T Y) Y \tag{4.91}$$

Para el cálculo de los nuevos pesos usando esta regla, se necesita la especificación del valor de la tasa de aprendizaje (α) y de la función de activación y su derivada.

Resumen del Algoritmo de Entrenamiento para el perceptrón contínuo

Dados "P"patrones de entrenamiento
$[X(1), Sd(1); X(2), Sd(2); \ldots; X(P), Sd(P)]$ donde $X(i) \in \Re^n$ y $Sd(i) \in \Re$; $i = 1, 2, \ldots, P$.

Tenemos que el vector de entrada aumentado está dado por:

$$Y(i) = \begin{bmatrix} X(i) \\ 1 \end{bmatrix} \tag{4.92}$$

Se denotará por "K$^{\text{a}}$l paso de entrenamiento y por "p$^{\text{a}}$ un contador que lleva el número de patrones evaluados en cada ciclo.

1. Seleccionar la tasa de aprendizaje $\alpha > 0$
2. Inicializar los pesos W con valores aleatorios pequeños. Recordar que $W \in \Re^{n+1}$.
3. Inicializar los contadores, la variable de acumulación del error y el error cuadrático medio máximo permitido por ciclo de entrenamiento. $K \leftarrow 1$; $p \leftarrow 1$; $E \leftarrow 0$; $E_{max} > 0$
4. Ciclo de entrenamiento. Se presenta las entradas y se calcula la salida del perceptrón.
   - $Y \leftarrow Y(p)$
   - $Sd \leftarrow Sd(p)$
   - $S = f(W^T Y)$
5. Actualización de los pesos
   - $W \leftarrow W + \frac{1}{2}\alpha(Sd - S) f'(W^T Y) Y$
6. Calcular el error del ciclo

   - $E \leftarrow E + \frac{1}{2}(Sd - S)^2$
7. Si p=P entonces ir al paso 8, de lo contrario:
   - $p \leftarrow p + 1$
   - Ir al paso 4.
8. Fin del ciclo de entrenamiento.

   Si $E < E_{max}$ entonces Fin de la sesión de entrenamiento. Se obtiene los valores de los pesos (W) y del número de ciclos (K), de lo contrario:
   - $E \leftarrow 0$
   - $p \leftarrow 1$
   - $K \leftarrow K + 1$
   - Ir al paso 4.

4.3.4 Redes Perceptrónicas Unicapas para Clasificación Múltiple

Hasta el momento hemos visto la clasificación para dos clases linealmente separables, usando los perceptrones tanto discretos como contínuos. Ahora utilizaremos las técnicas de corrección de error para realizar clasificación de multiples categorías que son linealmente separables. Este tipo de problemas de clasificación se caracteriza por el hecho de que existen "R"funciones discriminantes lineales tales que si:

$$g_i(X) > g_j(X), \quad para \quad i, j = 1, 2, \ldots, R \quad i \neq j \tag{4.93}$$

entonces el patrón X pertenecerá a la clase "i".

A continuación aparece un procedimiento para la clasificación de "R"categorías linealmente separables.

En este caso se necesita diseñar una red perceptrónica de una capa que constará de "R"neuronas. Así, se tendrá una matriz de pesos que será de dimensión $(n+1) \times R$ debido a las n entradas, el umbral y las R neuronas de salida. Las salidas se deben interpretar de acuerdo a la codificación binaria que se haya adoptado para cada clase. Se asumirá que la clase a la que pertenece un patrón será reflejada por aquella neurona del arreglo que proporcione el mayor valor de salida.

Ya que en este caso se tiene una matriz de pesos, se debe ajustar el algoritmo de entrenamiento para éste caso general quedando de la siguiente manera:

$$w_{ij}(K+1) = w_{ij}(K) + \frac{1}{2}\alpha(Sd_i(K) - S_i(K)) Y_j(K) \quad para\ i = 1, 2, \ldots, n+1 \quad j = 1, 2, \ldots, R \tag{4.94}$$

donde el subindice i denota la neurona asociada al peso a ser ajustado y el subindice j indica la entrada conectada con el peso.

4.3.5 Resumen del algoritmo de entrenamiento para clasificación múltiple usando redes perceptrónicas unicapas

Dados "P"patrones de entrenamiento
$[X(1), Sd(1); X(2), Sd(2); \ldots; X(P), Sd(P)]$ donde $X(i) \in \Re^n$ y $Sd(i) \in \Re^R$; $i = 1, 2, \ldots, P$.

Tenemos que el vector de entrada aumentado está dado por:

$$Y(i) = \begin{bmatrix} X(i) \\ 1 \end{bmatrix} \tag{4.95}$$

Se denotará por "K^{a}l paso de entrenamiento y por "p^{a} un contador que lleva el número de patrones evaluados en cada ciclo.

1. Seleccionar el factor de aprendizaje $\alpha > 0$
2. Inicializar los pesos W con valores aleatorios pequeños. Recordar que $W \in \Re^{n+1} \times \Re^R$.
3. Inicializar los contadores y una variable de acumulación del error. $K \leftarrow 1$; $p \leftarrow 1$; $E \leftarrow 0$.
4. Ciclo de entrenamiento. Se presenta las entradas y se calcula la salida del perceptrón.
   - $Y \leftarrow Y(p)$
   - $Sd \leftarrow Sd(p)$
   - $S(i) = Sign(W_i^T Y)$ para $i = 1, 2, \ldots R$.
5. Actualización de los pesos
   - $w_{ij} \leftarrow w_{ij} + \frac{1}{2}\alpha(Sd_i - S_i)x_j$
6. Calcular el error del ciclo
   - $E \leftarrow E + \frac{1}{2}\sum_{i=1}^{R}(Sd_i - S_i)^2$
7. Si p=P entonces ir al paso 8, de lo contrario:
   - $p \leftarrow p + 1$
   - Ir al paso 4.
8. Fin del ciclo de entrenamiento.

   Si $E = 0$ entonces Fin de la sesión de entrenamiento. Se obtiene los valores de los pesos (W) y del número de ciclos (K), de lo contrario:

   - $E \leftarrow 0$
   - $p \leftarrow 1$
   - $K \leftarrow K + 1$
   - Ir al paso 4.

4.4 Algoritmo de Retropropagación

El Algoritmo de retropropagación es el más utilizado para el entrenamiento de las redes neuronales en cascada. Dicho algoritmo permite la generación de mapas entrada-salida a partir de datos experimentales. En él los patrones de entrenamiento son presentados en forma secuencial. Si un patrón es presentado a la red y su clasificación o asociación resulta errónea, los pesos de la red serán ajustados de manera de reducir una funcional de costos generada a partir del error cuadrático medio existente entre la salida deseada y la salida neuronal. A este error se le llama "error de aprendizaje". Generalmente éste ajuste de los pesos se llevará a cabo hasta que el error cuadrático medio de todos los patrones se encuentre por debajo de un nivel máximo fijado por el usuario, de acuerdo a los niveles de precisión requeridos según la aplicación a desarrollar.

Durante la fase de entrenamiento la red realiza dos tareas: Una tarea de presentación de los patrones en forma de cascada directa, con la cual se obtienen las salidas neuronales a partir de las entradas presentadas a la red neuronal. La otra tarea es la de actualización de los pesos, la cual se realiza desde atrás hacia adelante, ajustando primero los pesos de la capa de salida, pasando por las capas ocultas y por último los pesos de la capa de entrada. Este proceso se hace de esta manera debido a la dependencia que tienen los pesos de todas las capas sobre las salidas de la red neuronal. Ya que el algoritmo de ajuste de los pesos está basado en minimización de una funcional de costo que depende del error de aprendizaje, y para ello se utiliza técnicas basadas en derivadas; entonces al aplicar la regla de la cadena sucesivamente se puede obtener los gradientes de los pesos de cada capa sobre la funcional de costo.

4.4.1 Regla Delta para un nivel de múltiples perceptrones contínuos

El caso más sencillo del algoritmo de entrenamiento de retropropagación lo constituye la llamada "'Regla Delta", la cual es la generalización del algoritmo para perceptrones contínuos arreglados en una capa.

En este caso se considerará una red perceptrónica de una capa que constará de "R"neuronas. Así, nuevamente se tendrá una matriz de pesos que será de dimensión $(n + 1) \times R$ debido a las n entradas, la desviación y las R neuronas de salida.

Utilizando notación matricial se obtiene que la salida neuronal S puede ser expre-

sada como:

$$S = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_R \end{bmatrix} = \Gamma[WY] \tag{4.96}$$

donde:

$$Y = [x1, x2, \ldots, xn, b]^T \tag{4.97}$$

$$W = \begin{bmatrix} w_{11} & w_{12} & \ldots & w_{1n+1} \\ w_{21} & w_{22} & \ldots & w_{2n+1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{R1} & w_{R2} & \ldots & w_{Rn+1} \end{bmatrix} \tag{4.98}$$

y el operador de funciones de activación $\Gamma[.]$ es la matriz diagonal

$$\Gamma[.] = \begin{bmatrix} f(.) & 0 & \ldots & 0 \\ 0 & f(.) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & f(.) \end{bmatrix} \tag{4.99}$$

La salida deseada será el vector:

$$Sd = \begin{bmatrix} Sd_1 \\ Sd_2 \\ \vdots \\ Sd_R \end{bmatrix} \tag{4.100}$$

La expresión para la funcional de costo será generalizada para incluir los errores cuadráticos de todas las salidas. Esto es, para un patrón específico se obtiene:

$$J(E_p) = \frac{1}{2}\sum_{i=1}^{R}(Sd_i - S_i)^2 = \frac{1}{2}\|Sd - S\|^2 \tag{4.101}$$

El algoritmo general para el ajuste de la matriz de pesos será:

$$w_{ij}(K+1) = w_{ij}(K) + \frac{1}{2}\alpha(Sd_i - S_i)f'(WY)x_j; \quad para\ i = 1, 2, \ldots, n+1; \quad j = 1, 2, \ldots, R \tag{4.102}$$

donde i es el número de la neurona a la cual está conectada el peso a ser ajustado y j es el número de la entrada conectada con el peso.

Resumen del Algoritmo de Entrenamiento para una capa de múltiples perceptrones

Dados "P"patrones de entrenamiento
$[X(1), Sd(1); X(2), Sd(2); \ldots; X(P), Sd(P)]$ donde $X(i) \in \Re^n$ y $Sd(i) \in \Re$; $i = 1, 2, \ldots, P$.

Tenemos que el vector de entrada aumentado está dado por:

$$Y(i) = \begin{bmatrix} X(i) \\ 1 \end{bmatrix} \tag{4.103}$$

Se denotará por "K^{a}l paso de entrenamiento y por "p^{a} un contador que lleva el número de patrones evaluados en cada ciclo.

1. Seleccionar el factor de aprendizaje $\alpha > 0$
2. Inicializar los pesos W con valores aleatorios pequeños. Recordar que $W \in \Re^{(n+1)\times R}$.
3. Inicializar los contadores, la variable de acumulación del error y el error cuadrático medio máximo permitido por ciclo de entrenamiento. $K \leftarrow 1$; $p \leftarrow 1$; $E \leftarrow 0$; $E_{max} > 0$
4. Ciclo de entrenamiento. Se presenta las entradas y se calcula la salida del perceptrón.
   - $Y \leftarrow Y(p)$
   - $Sd \leftarrow Sd(p)$
   - $S = f(W^T Y)$
5. Actualización de los pesos
   - $w_{ij} \leftarrow wij + \frac{1}{2}\alpha(Sd_i - S_i)f'(W^T Y)x_j$
6. Calcular el error del ciclo
   - $E \leftarrow E + \frac{1}{2}\sum_{i=1}^{R}(Sd_i - S_i)^2$
7. Si p=P entonces ir al paso 8, de lo contrario:
   - $p \leftarrow p + 1$
   - Ir al paso 4.
8. Fin del ciclo de entrenamiento.

   Si $E < E_{max}$ entonces Fin de la sesión de entrenamiento. Se obtiene los valores de los pesos (W) y del número de ciclos (K), de lo contrario:

- $E \leftarrow 0$
- $p \leftarrow 1$
- $K \leftarrow K + 1$
- Ir al paso 4.

4.4.2 Regla Delta Generalizada

Este algoritmo de entrenamiento es la generalización para redes multicapas de la "Regla Deltaüsada para el entrenamiento de una capa de perceptrones contínuos. Está basado igualmente en un método iterativo que busca minimizar una funcional de costo formada por la suma instantánea de los errores cuadráticos y definida por:

$$J(e) = \frac{1}{2}\sum_{j=1}^{R} e_j^2 = \frac{1}{2}\sum_{j=1}^{R}(Sd_j - S_j)^2 \tag{4.104}$$

Así, que se deben actualizar los pesos de todas las capas para minimizar la funcional de costo. El ajuste de los pesos de las interconexiones sinápticas se hace para cada patrón de entrenamiento presentado a la red y utilizando una expresión en sentido opuesto al gradiente de la funcional en la dirección de cada uno de los pesos. Esta regla se expresa de la siguiente manera:

$$\triangle wi_{jk} = -\alpha\frac{\partial J(e)}{\partial wi_{jk}} \tag{4.105}$$

donde α es la tasa de aprendizaje, i indica la capa donde se van a actualizar los pesos, j y k representan el número de la neurona a la que llega el peso a modificar y la entrada de la cual proviene, respectivamente.

Se puede observar de la ecuación (4.105) que la corrección de los pesos es proporcional al gradiente. Usando la regla de la cadena consecutivamente, se obtiene la siguiente expresión:

$$\frac{\partial J(e)}{\partial wi_{jk}} = \frac{\partial J(e)}{\partial e_j}\frac{\partial e_j}{\partial Y_j}\frac{\partial Y_j}{\partial (Wi^T Y_k)}\frac{\partial (Wi^T Y_k)}{\partial wi_{jk}} \tag{4.106}$$

donde Wi_{jk} es el peso que conecta la k-ésima señal de entrada con la j-ésima neurona de la i-ésima capa.

Derivando la ecuación (4.104) se obtiene:

$$\frac{\partial J(e)}{\partial e_j} = e_j \tag{4.107}$$

mientras que la derivada del error $e_j = Sd_j - S_j$ con respecto a la salida S_j es:

$$\frac{\partial e_j}{\partial S_j} = -1 \tag{4.108}$$

La derivada de la salida de la capa j con respecto a la suma ponderada de los pesos por las entradas a la misma capa es:

$$\frac{\partial Y_j}{\partial (Wi^T Y_k)} = f_j'(Wi^T Y_k) \tag{4.109}$$

y por último, la derivada de la suma ponderada con respecto a los pesos que se buscan ajustar es:

$$\frac{\partial (Wi^T Y_k)}{\partial wi_{jk}} = Y_k \tag{4.110}$$

sustituyendo las expresiones obtenidas en la ecuación (4.106) se obtiene:

$$\frac{\partial J(e)}{\partial wi_{jk}} = -e_j * f_j'(Wi^T Y_k) * Y_k \tag{4.111}$$

Con lo que se puede definir la *Regla Delta* de la siguiente forma:

$$\triangle Wi_{jk} = Wi(K+1) - Wi(K) = \alpha * \delta_j * Y_k \tag{4.112}$$

donde "δ_j"queda definida dependiendo de la ubicación de la capa a la cual se le va a actualizar los pesos.

- Si la j-ésima neurona pertenece a la capa de salida entonces:

$$\delta_j = e_j * f_j'(Wi^T Y_k) \tag{4.113}$$

- Si la j-ésima neurona está en las capas intermedias o de entrada, entonces:

$$\delta_j = f_j'(Wi^T Y_k) * \sum_L (\delta_L * wi_{Lj}) \tag{4.114}$$

  donde L es el número de neuronas de la siguiente capa con las que está conectada la j-ésima neurona.

El proceso de entrenamiento utilizando la regla delta generalizada consiste en presentar un patrón a la red y calcular las salidas de las neuronas de la primera capa, posteriormente calcular las salidas de las neuronas de las capas escondidas hasta por último obtener las salidas de las neuronas de la capa de salida. En este momento se inicia la fase de actualización de los pesos. Para ello, se calculan las sensibilidades δ_j de cada neurona de la última capa, posteriormente las sensibilidades de la penúltima capa y así sucesivamente hasta obtener las sensibilidades de las neuronas de la capa de entrada. Una vez obtenidas todas las sensibilidades, se utiliza la ley de actualización para los pesos de cada capa.

Resumen del Algoritmo de Entrenamiento usando la Regla Delta generalizada

Dados "P"patrones de entrenamiento
$[X(1), Sd(1); X(2), Sd(2); \ldots; X(P), Sd(P)]$ donde $X(i) \in \Re^n$ y $Sd(i) \in \Re$; $i = 1, 2, \ldots, P$.

Tenemos que el vector de entrada aumentado está dado por:

$$Y(i) = \begin{bmatrix} X(i) \\ 1 \end{bmatrix} \tag{4.115}$$

Se denotará por "K^{a}l paso de entrenamiento y por "p^{a} un contador que lleva el número de patrones evaluados en cada ciclo.

1. Seleccionar el factor de aprendizaje $\alpha > 0$
2. Inicializar los pesos $W1, W2, \ldots, WH$ con valores aleatorios pequeños. Recordar que Wi son matrices de pesos cuyas dimensiones dependen del número de entradas y salidas de cada capa.
3. Inicializar los contadores, la variable de acumulación del error y el error cuadrático medio máximo permitido por ciclo de entrenamiento. $K \leftarrow 1$; $p \leftarrow 1$; $E \leftarrow 0$; $E_{max} > 0$
4. Ciclo de entrenamiento. Se presenta las entradas y se calcula la salida del perceptrón.
   - $Y \leftarrow Y(p)$
   - $Sd \leftarrow Sd(p)$
   - $S = f(WH^T f(WH - 1...f(W1Y))...)$
5. Calcular las sensibilidades en cada capa. Iniciando con la capa de salida.
   - Si la j-ésima neurona pertenece a la capa de salida entonces:
   $$\delta_j = e_j * f_j'(Wi^T Y_k) \tag{4.116}$$
   - Si la j-ésima neurona está en las capas intermedias o de entrada, entonces:
   $$\delta_j = f_j'(Wi^T Y_k) * \sum_L (\delta_L * wi_{Lj}) \tag{4.117}$$
   donde L es el número de neuronas de la siguiente capa con las que está conectada la j-ésima neurona.
6. Actualización de los pesos

   - $wi_{jk}(K+1) = wi_{jk}(K) + \alpha * \delta_j * Y_k$
7. Calcular el error del ciclo
   - $E \leftarrow E + \frac{1}{2} \sum_{i=1}^{R} (Sd_i - S_i)^2$
8. Si p=P entonces ir al paso 8, de lo contrario:
   - $p \leftarrow p + 1$
   - Ir al paso 4.
9. Fin del ciclo de entrenamiento.
   Si $E < E_{max}$ entonces Fin de la sesión de entrenamiento. Se obtiene los valores de los pesos $(W1, W2, \ldots, WH)$ y del número de ciclos (K), de lo contrario:
   - $E \leftarrow 0$
   - $p \leftarrow 1$
   - $K \leftarrow K + 1$
   - Ir al paso 4.

4.4.3 Retropropagación con factor de momento

Este método es una variación del algoritmo de retropropagación, el cual está basado en técnicas empíricas para mejor el ajuste de los pesos. Este algoritmo incorpora un término llamado el "Factor de Momento"cuya función es incorporar memoria al aprendizaje, considerando los ajuste previos que se hayan realizado a los pesos de la red neuronal. Este algoritmo acelera la convergencia de los pesos a sus valores ideales cuando la trayectoria de los parámetros se mueve en una dirección apropiada. Igualmente, evita el estancamiento en mínimos locales de la funcional de costo.

La expresión para la actualización de los pesos es la siguiente:

$$\triangle wi_{jk}(K+1) = \gamma \triangle wi_{jk}(K) + \alpha * \delta_j * Y_k \tag{4.118}$$

donde γ es el factor de momento y sus valores suelen oscilar entre 0 y 1 (generalmente se propone valores entre 0.95-0.98).

4.4.4 Retropropagación con tasa de aprendizaje variable

El algoritmo de retropropagación con tasa de aprendizaje variable es otra de las llamadas variaciones empíricas de la regla delta generalizada. Su funcionamiento está basado en la búsqueda de la aceleración en la convergencia de los pesos a sus valores ideales tomando en cuenta la forma de la superficie espacial generada por la funcional de costo basada en el error de aprendizaje. La velocidad de convergencia será mayor

si la tasa de aprendizaje es mas grande en las regiones planas lo cual lleva a actualizaciones mas rápidas, mientras que en las regiones irregulares, la actualización de los pesos deben llevarse a cabo con una tasa de aprendizaje pequeña.

Para el uso de éste algoritmo de entrenamiento, se utiliza la siguiente funcional de costo:

$$J(e) = \sum_{j=1}^{P} \left[\sum_{k=1}^{R} e_k^2 \right] = \sum_{j=1}^{P} \left[\sum_{k=1}^{R} (Sd_k - S_k)^2 \right] \tag{4.119}$$

donde P es el número de patrones de entrenamiento y R es el número de salidas neuronales.

El ajuste de los pesos se realiza usando la regla delta generalizada. Después de obtenidos los nuevos pesos se verifica el valor de la funcional de error con los nuevos pesos. Si el nuevo valor de la funcional es mayor que un porcentaje pre-establecido (generalmente entre 1% y 5%), entonces se mantiene los pesos anteriores desechando los recién calculados, se decrementa el factor de aprendizaje multiplicándolo por un factor ρ entre 0 y 1 y se fija en cero el factor de momento (γ). En caso de que la evaluación de la funcional de costo genere un valor al usar los nuevos pesos menor que el anterior, entonces se restituye el valor original del factor de momento (en caso de que hubiese sido llevado a cero) y se incrementa la tasa de aprendizaje multiplicándola por un factor $\phi > 1$.

Ya que este método solo va a modificar los pesos en caso de que mejoren la respuesta del sistema, entonces sus pesos convegerán mas rápidamente a sus valores ideales que con la regla delta generalizada o el algoritmo de retropropagación con factor de momento. Su contraparte está en la cantidad de parámetros que deben ser seleccionados por el usuario.

4.5 Metodología para el diseño de aplicaciones usando Redes Neuronales

Esta metodología surge de la integración de enfoques, técnicas y otras metodologías provenientes de diversas áreas vinculadas con los sistemas basados en conocimiento e Ingeniería de Software y puede ser considerada en diferentes campos de desarrollo de aplicaciones computacionales; de hecho coincide ampliamente con la presentada en el capítulo de Sistemas Expertos, encontrándose diferencias únicamente en las descripciones particulares de cada técnica.

La metodología presentada a continuación está dividida en etapas, fases y pasos para una mejor comprensión de las actividades a desarrollar

Etapa 1: Análisis y descripción del problema.

Determina las características del problema y evalúa la factibilidad de utilizar una red neuronal. En esta etapa se pretende determinar la naturaleza del problema y los objetivos precisos que indiquen exactamente cómo se espera que la red neuronal contribuya a la solución del problema. Estudia los diferentes recursos de información, datos

e infraestructura con que se cuenta y verifica la posibilidad de su aprovechamiento en las fases de desarrollo e implantación.

- Fase 1.1.- Descripción General del Problema:
  - 1.1.1.- Familiarización con el proceso en el cual se desea utilizar una red neuronal.
  - 1.1.2.- Familiarización con los ambientes computacionales donde se encuentran los datos a ser utilizados.
  - 1.1.3.- Definición detallada del problema que motiva el diseño de una aplicación que utiliza una red neuronal.
- Fase 1.2.- Análisis de Factibilidad para la utilización de una red neuronal: En esta fase se estudia si el sistema cumple con las condiciones para utilizar una red neuronal, tomando en cuenta aspectos como la cantidad y calidad de datos que se poseen
- Fase 1.3.- Análisis de datos: Verificación de la ubicación y forma de representación de los datos a ser utilizados por la red neuronal, considerando el tipo de base de datos y la plataforma computacional.

Etapa 2: Especificación de requerimientos.

En esta etapa se estudia los requerimientos globales del sistema a desarrollar, considerando hacia quién estará dirigido el sistema, restricciones de acceso, requerimientos funcionales, formatos deseados por los usuarios finales. Es importante discutir con los potenciales usuarios del sistema y tomar en cuenta sus aspiraciones y comentarios para garantizar que posteriormente estén dispuestos a utilizar el sistema desarrollado.

- Fase 2.1.- Estimación del perfil de los usuarios finales del sistema a desarrollar.
- Fase 2.2.- Verificación de los requerimientos con el usuario.
- Fase 2.3.- Determinación de los requerimientos de información: Se especifica la información que debe producir la red neuronal y sus atributos tales como el formato de presentación, la frecuencia de salida y su interconexión con otros programas.
- Fase 2.4.- Determinación de los requerimientos funcionales: Consiste en la definición de las funciones generales que debe satisfacer el sistema que utiliza la red neuronal.
- Fase 2.5.- Determinación de los requerimientos de entrada de datos:
  - 2.5.1.- Selección de las posibles fuentes de entrada a la red neuronal.
  - 2.5.2.- Identificación de las fuentes de datos.

  - 2.5.3.- Especificación de los procesos de adquisición de datos.
  - 2.5.4.- Especificación de los procesos de generación de parámetros.
  - 2.5.5.- Caracterización de la inter-operabilidad entre las bases de datos que se requieren en la implantación.

- Fase 2.6.- Definición de los requerimientos de hardware y software para la implantación de la red neuronal:
  - 2.6.1.- Especificación de la plataforma de hardware que se utilizará para el entrenamiento y operación de la red neuronal.
  - 2.6.2.- Determinación, análisis y selección de las herramientas de software disponibles en el mercado para el desarrollo de aplicaciones con redes neuronales.

Etapa 3: Análisis de costos, tiempo y recursos.

En esta etapa se realiza un estimado de los costos del desarrollo del sistema que utiliza la red neuronal incluyendo equipos, programas y honorarios profesionales. Igualmente se realiza un cronograma de las actividades a ser desarrolladas. Generalmente en esta etapa terminaría el estudio de factibilidad de realizar el sistema computacional deseado.

- Fase 3.1.- Elaboración del plan de actividades de desarrollo e implantación.
- Fase 3.2.- Estimación del tiempo requerido para el diseño y entrenamiento de la red neuronal.
- Fase 3.3.- Estimación de los recursos computacionales (hardware-software) requeridos para el desarrollo del Sistema.
- Fase 3.4.- Estimación de los costos de desarrollo.

Etapa 4: Diseño de la red neuronal.

- Fase 4.1.- Análisis y procesamiento de datos: En esta fase se debe seleccionar el conjunto de datos que serán utilizados para entrenar la red neuronal y los datos que se usarán para probar su funcionamiento.
- Fase 4.2.- Selección de la topología de la red neuronal: La descripción de una red neuronal puede venir dada en término de los siguientes elementos:
  - Número de entradas a la red neuronal.
  - Número de salidas.
  - Número de capas intermedias u ocultas.
  - Número de neuronas en cada capa.

  - Funciones de activación utilizadas en cada capa.
  - Parámetros de entrenamiento (depende del algoritmo utilizado).
  - Algoritmo de entrenamiento.

El número de entradas y salidas de la red neuronal se obtienen del problema a resolver, ya que dependiendo de las variables que se estén manejando y de la finalidad del uso de la red se puede definir el número de entradas y salidas.

El número de capas intermedias y el número de neuronas en cada capa intermedia son parámetros de diseño que suelen ser escogidos por ensayo y error, aunque existe un teorema que demuestra que las redes neuronales con una sola capa intermedia puede ser utilizada como "aproximador universal"[15], ya que siempre logra reproducir cualquier función no lineal por compleja sea. Este concepto es de carácter matemático y deja de ser útil a la hora de implementaciones reales, ya que el número de neuronas requeridas (en algunos casos) puede ser tan grande que haga al problema inmanejable computacionalmente. Para efectos prácticos se suele probar en primera instancia una red con un capa escondida y se aumenta paulatinamente el número de neuronas en dicha capa hasta verificar si se logra resultados satisfactorios. En caso de que se haya aumentado demasiado el número de neuronas y no se logre buenos resultados se procede a la incoporación de otra capa intermedia y así sucesivamente.

Las funciones de activación de la capa de salida se escogen de acuerdo a las salidas deseadas para la red neuronal. Generalmente para el caso de clasificación se suelen utilizar funciones como las sigmoides o tangentes hiperbólicas, mientras que para casos donde la salida de la red estime alguna variable física, por lo general se usan funciones lineales.

Etapa 5: Diseño preliminar del sistema computacional que utiliza la red neuronal.

En esta etapa se realiza la ingeniería de detalle del Sistema, se diseña cuidadosamente cada una de las módulos que comprenderán la herramienta, se hace la selección final de los componentes a utilizar y se diseña los protocolos necesarios para las interrelaciones con otros programas y/o equipos requeridos.

- Fase 5.1.- Diseño preliminar de la arquitectura del Sistema computacional.
- Fase 5.2.- Selección de la herramienta computacional de acuerdo a los requerimientos surgidos en la etapa de diseño de la red neuronal.
- Fase 5.3.- Diseño preliminar de procesos de adquisición y almacenamiento de datos.
- Fase 5.4.- Diseño preliminar de procesos de interconexión:
  - 5.4.1.- Integración Interna.

  - 5.4.2.- Integración Externa.
  - 5.4.2.- Selección de software auxiliar.

- Fase 5.5.- Verificación del diseño preliminar de la red neuronal.

Etapa 6: Diseño e Implantación del sistema computacional que utiliza la red neuronal.

Esta es la etapa final del desarrollo del sistema computacional que utiliza una red neuronal, aquí, se construye, implanta, prueba y depura el sistema. Posterior a la finalización de la implantación comienza la fase de mantenimiento y actualización que durará durante la vida del sistema, ya que se busca mantenerlo operativo en las mejores condiciones e incorporando conocimiento y/o recursos nuevos al sistema según los requerimientos tecnológicos para su vigencia.

- Fase 6.1.- Construcción del prototipo.
- Fase 6.2.- Validación del prototipo.
- Fase 6.3.- Construcción del modelo operacional
- Fase 6.4.- Prueba y depuración
- Fase 6.5.- Mantenimiento y actualización.

Bibliografía

[1] J. Albus, "A theory of cerebellar function", *Mathematical Biosciences*, Vol 10, pp. 25-61, 1971.

[2] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis*, USA: John Wiley & Sons Inc., 1984.

[3] N.K. Bose and P. Liang, *Neural Networks Fundamental with Graphs, Algorithms, and Applications*, USA:McGraw-Hill, 1996.

[4] G. Carpenter and S. Grossberg, "A massivelly parallel architecture for a self organizing neural pattern recognition machine", *Computer Vision, Graphics and Image Processing*,Vol 37, pp. 54-115, 1987.

[5] E. Colina-Morles and N. Mort, "Neural network-based adaptive control design,"*Journal of Systems Engineering Vol. 1, No. 1, pp. 9-14, 1993.*

[6] J. Dayhoff, Neural Network Architectures*, USA:Van Nostrand Reinhold, 1990.*

[7] J. Elman, "Finding Structure in Time". Cognitive Science*, No. 14, pp. 179-211, 1990.*

[8] J. Freeman and D. Skapura, Redes Neuronales: Algoritmos, aplicaciones y técnicas de Programación*, USA:Addison-Wesley/Diaz de Santos, 1993.*

[9] S.I. Gallant, Neural Network Learning and Expert Systems*, USA:MIT Press, 1995.*

[10] S. Grossberg, "Adaptive Patern Classification and Universal Recording: Parallel Developmen and Coding of Neural Feature Detectors", Biological Cybernetics,*No. 23, pp. 121-134, 1976.*

[11] M. Hagan, H. Demuth and M. Beale, Neural Networks Design*. USA: PWS Publishing Company, 1996*

[12] S. Haykin, Neural Network A comprehensive Foundation*, USA: IEEE Press, 1994.*

[13] D. Hebb, Organization of Behaviour: A Neuropsycological Theory*, USA:John Wiley & Sons Inc., 1949.*

[14] G. Hinton and T. Senjnowski, "Learning and relearning in Boltzmann Machines"in Parallel Distributed Procesing: Explorations in the microstructure of Cognition *, USA:MIT Press, 1986.*

[15] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators", Neural Networks,*Vol. 2, pp. 359-366, 1989.*

[16] E. Ikonen, Algorithms for process modelling using Fuzzy Neural Networks*, Finland:OULU University Press, 1996.*

[17] R. Jonhson and D. Wichern, Applied Multivariate Statistical Analysis*, USA: Prentice Hall, 1992.*

[18] T. Kohonen et al*, "Associative Recall of Images",* Biological Cybernetics,*No. 22, pp. 159-168, 1976.*

[19] B. Kosko, Neural Networks and Fuzzy Systems*, USA:Prentice Hall, 1992.*

[20] J. G. Kuschewski, S. Hui and S. H. Żak, "Application of feedforward networks to dynamical system identification and control,". IEEE Transactions on Control Systems Technology*, Vol. 1, No. 1, pp. 37-49, 1993.*

[21] C. Lau, Neural Networs: Theoretical Foundations and Analysis*, USA:IEEE Press, 1992.*

[22] Y. Le Cun, "Une procedure d'apprentissage pour reseau a seuil assymetrique", Cognitiva,*Vol 85, pp. 277-286, 1985.*

[23] L. Lebart, A. Morineau and K. Warwick, Multivariate Descriptive Statistical Analysis*, USA: John Wiley & Sons , 1984.*

[24] G. Marchandani, "On Hidden Nodes for Neural Nets", IEEE Trans. on Circuits and Systems*, Vol 36, pp. 121-134, 1989.*

[25] M. Minsky and S. Paperi, Perceptrons: An Introductions to computational Geometry*, USA:MIT Press, 1969.*

[26] D.F. Morrison, Multivariate Statistical Methods*, USA: McGraw-Hill, 1990.*

[27] W. Mc Culloch and W. Pitts, "A logical calculus of the ideas immanent in Nervious Activities", Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133, 1943.*

[28] D. Parker, "Learning-logic: Casting the cortex of the human brain in silicon", Technical Report TR-47, Center for Computational Research in Economics and Management Science, MIT, 1985.

[29] F. Rosenblatt, "The Perceptron: A Perceiving and Recognizing Automation", Technical Report 85-460-1. Cornel Aeron. Lab.*, 1957.*

[30] D. Rumelhart, G. Hinton and R. Williams, "Learning Internal Representation by error propagation"in Parallel Distributed Procesing: Explorations in the microstructure of Cognition *,Vol. 1, pp.318-362, USA:MIT Press, 1986.*

[31] H. Sira-Ramírez and E. Colina-Morles, "A sliding mode strategy for adaptive learning in adalines". IEEE Transactions on Circuits and Systems -I: Fundamental Theory and Applications*, Vol. 42, No. 12, 1995.*

[32] H. Sira-Ramírez and S. H. Żak, "The adaptation of perceptrons with applications to inverse dynamics identification of unknown dynamic systems". IEEE Transactions on Systems, Man and Cybernetics*, Vol. 21, No. 3, 634-643,1991.*

[33] L. H. Tsoukalas and R. E. Uhrig, Fuzzy and Neural Approaches in Engineering*, USA:John Wiley & Sons Inc., 1997.*

[34] M. Vidyasagar, A Theory of Learning and Generalization with applications to Neural Networks and Control Systems*, Great Britain:Springer-Verlag, 1997.*

[35] J. Von Newmann, Papers of John Von Newmann on Computing and Computer Theory*, USA: Edited by W. Aspray & A. Burks, 1986.*

[36] P. Wasserman, Neural Computing: Theory and Practice*, USA:Van Nostrand Reinhold, 1989.*

[37] B. Widrow, Generalization and Information storage in Networks of Adalines Neurons*, USA:Spartan Books, 1962.*

[38] B. Widrow and M.A. Lehr, "30 years of adaptive neural networks: perceptron, madaline, and backpropagation". Proceedings IEEE*, Vol. 78, 1415-1442, 1990.*

[39] N. Wiener, Cybernetics*, USA:MIT Press, 1948.*

[40] J. M. Zurada, Introduction to Artificial Neural Systems*, USA:West Publishing Company, 1992.*

Computación Evolutiva

Francisco Hidrobo Torres
Laboratorio SUMA
Facultad de Ciencias
ULAM
E-mail: hidrobo@ula.ve

José L. Aguilar Castro
CEMISID
Departamento de Computación
Facultad de Ingeniería
Universidad de Los Andes
Mérida 5101, Venezuela
E-mail: aguilar@ula.ve

Capítulo 5

Computación Evolutiva

El propósito de este capítulo es dar una introducción a la Computación Evolutiva. Se presentaran las ideas centrales de la Computación Evolutiva, así como los diferentes aspectos esenciales de esta área de estudio. Después, se bosquejan las técnicas que normalmente son consideradas como partes de la Computación Evolutiva: los Algoritmos Genéticos, los Programas Evolutivos, las Estrategias Evolutivas, la Programación Evolutiva y la Programación Genética. Los Algoritmos Genéticos son objeto de una presentación mas exhaustiva por ser la técnica de mayor uso de esta área.

5.1 Generalidades

Observar a los seres vivos y estudiar las complejas transformaciones que los han llevado a lo que son, ha sido una tarea atractiva para muchos científicos. La teoría de la evolución desarrollada por Darwin representa el trabajo más importante en esa área. No es nuestra intención detallar los planteamientos de dicha teoría, ni explicar el trasfondo de la misma; lo que queremos es relacionar algunos conceptos expresados en la teoría de la evolución con un conjunto de técnicas usadas para resolver computacionalmente algunos problemas, enmarcadas en lo que se ha denominado Computación Evolutiva [3, 5, 16, 17, 25, 26, 43, 54]. Así, la Computación Evolutiva (**CE**) abarca los modelos computacionales que usan como elemento clave algún mecanismos de la teoría de la **evolución** para su diseño e implementación.

El principal aporte de la CE a la metodología de resolución de problemas ha sido el uso de mecanismos de selección de soluciones potenciales y de construcción de nuevos candidatos por recombinación de características de otros ya existentes, de modo similar a como ocurre en la evolución de organismos naturales [9, 10, 22, 36, 38, 39, 45, 46, 50]. El fundamento principal es aprovechar ciertas ideas genéricas que hay detrás de los procesos evolutivos de las especies, mas que tratar de reproducirlos, con el objetivo de abordar problemas complejos de búsqueda y aprendizaje.

Las implementaciones concretas en el área de la CE han sido denominadas Algoritmos Evolutivos [5, 16, 17, 25, 26, 43]. El propósito genérico de los algoritmos evolutivos es guiar una búsqueda estocástica haciendo evolucionar un conjunto de estructuras, seleccionando de modo iterativo las más adecuadas. Tales algoritmos se caracterizan, en general, por ser muy simples desde el punto de vista biológico, pero

lo suficientemente complejos para proporcionar robustos y potentes mecanismos de búsqueda.

Un algoritmo evolutivo se ejecuta sobre una población de individuos que representan a un conjunto de soluciones candidatas a un problema, la cual es sometida a ciertas transformaciones (a través de *Operadores Genéticos*) para actualizar la búsqueda, y después, a un proceso de selección que favorece a algunos individuos (habitualmente los mejores). Cada ciclo de *transformación + selección* representa una generación.

5.2 Bases biológicas de la Computación Evolutiva

Desde el punto de vista biológico, la CE se basa en tres teorías [8, 9, 10, 13, 22, 36, 37, 38, 39, 45, 50, 51]: la de *evolución de Darwin*, la de *seleccionismo de Weismann* y la de *genética de Mendel*. Cada una ha aportado elementos fundamentales para el desarrollo de los modelos asociados a la CE.

La teoría de la evolución biológica nace a mediados del siglo XIX, cuando Darwin propuso un mecanismo que permite explicar el cambio de las especies a lo largo del tiempo [8, 9, 10, 13, 22, 39, 50, 51]. Desde el principio, el evolucionismo ha tenido como meta responder la pregunta de cómo se produce esa lenta evolución biológica. Darwin y la mayoría de los evolucionistas que lo sucedieron, proponen a la *selección natural* como uno de los mecanismos fundamentales capaces de explicar la génesis de nuevos tipos específicos.

A continuación presentaremos los postulados de la selección natural que conforman las bases biológicas de la CE:

1. No todos los individuos de una misma especie son iguales. Entre ellos pueden encontrarse pequeñas variaciones que serán transmitidas hereditariamente a sus descendientes.

2. Algunas de estas variaciones podrán resultar favorables en determinados ambientes, lo que facilitará la supervivencia de sus portadores.

3. Todas las poblaciones tienden a aumentar su número geométricamente. Sin embargo, en condiciones reales, en cada generación el número permanece aproximadamente constante. Esta poca variación en el número de individuos en cada generación se da por el hecho de que no todos sobreviven, ya que existe una lucha por la supervivencia donde **triunfa** el que es portador de las características más favorables en el medio donde vive.

4. Los individuos mejor adaptados tendrán más posibilidades de dejar descendientes a los cuales transmitirán sus características. De esta forma, se irá incrementando la cantidad de individuos portadores de las características más favorables respecto del resto de la población.

5. Como las características favorables se van acumulando (agregando), con el tiempo surgen grandes diferencias entre el grupo original y los individuos que poseen dichas características. Finalmente, este último grupo se aparta tanto del tipo original (especie preexistente), que se forma una nueva especie.

Este mecanismo, conocido como selección natural, ha tenido muchas críticas, desde aquella época hasta el presente; como también, ha sido objeto de varias correcciones. Más importante que las correcciones ha sido la utilización de la teoría de Darwin para la generación de nuevas corrientes, entre las que destaca la NeoDarwinista; la cual combina la teoría de Darwin con los aportes de la Genética, conocida también con el nombre de Teoría Sintética de la Evolución [18, 22]. Esta teoría postula un principio que es altamente relevante para la CE: *La unidad sobre la que actúa la evolución no es el individuo, sino otra de orden superior: la población.*

Además de los aspectos relacionados a la evolución, los biólogos han buscado descubrir las reglas que explican cómo las características de los descendientes se relacionan con las de sus padres y las de los padres de sus padres; esto es, cómo ocurre la *Herencia Genética* (otro punto relevante en la CE). De entre las teorías formuladas para explicar cómo se heredan las características, una merece especial mención. Esta es la de Mendel, que proporcionó el fundamento sobre el cual se ha basado toda la investigación genética posterior [8, 12, 13, 22, 45].

Para explicar los resultados obtenidos en sus experimentos, Mendel formuló una serie de hipótesis. No se trataba de observaciones ni de hechos, sino de afirmaciones que, de ser verdaderas, proporcionarían una explicación de los resultados obtenidos. Las hipótesis formuladas por Mendel fueron las siguientes:

1. En cada organismo existe un par de factores que regulan la aparición de una cierta característica (Hoy en día, a estos factores los denominamos genes).

2. El organismo obtiene tales factores de sus padres, un factor por cada padre.

3. Cada uno de estos factores se transmite como una unidad discreta **inmodificable**.

4. Cuando las células reproductivas (espermatozoides u óvulos) están formadas, los factores se separan y se distribuyen a los gametos en forma de unidades independientes. Esta afirmación se conoce comúnmente con el nombre de primera ley de Mendel, o ley de la segregación.

5. Si un organismo posee dos factores diferentes para una característica dada, uno de ellos debe expresarse y excluir totalmente al otro. Hoy en día, los biólogos usan el término *alelo* para describir las formas alternativas de un gen que controla la aparición de una característica dada.

Por su parte, la teoría de Weismann ha tenido implicaciones cruciales para la evolución [8, 12, 22, 36, 38, 45, 46, 53]: Demostró que los cambios en el cuerpo, debido

al uso o el desuso de partes del mismo, no se reflejan en el plasma del germen, y por consiguiente, no pueden ser heredados. A propósito, realizó experimentos cortando las colas de los ratones padres, y demostró que la descendencia de los ratones todavía tenían colas, aun después de muchas generaciones. De esta forma, probó que el cuerpo del padre transmite meramente su germen plasmático, invalidando así la teoría Lamarkista.

5.3 Algoritmos Evolutivos

En esta sección presentaremos los aspectos más resaltantes de los algoritmos evolutivos.

5.3.1 Operadores Genéticos

Los operadores genéticos, llamados también operadores evolutivos, son los que realizan los cambios de la población durante la ejecución de un algoritmo evolutivo. Clásicamente, existen dos operadores o procesos genéticos [11, 24, 29, 42]: mutación y cruce. Sin embargo, han sido propuestos otros operadores que se acoplan a problemas particulares [11, 24, 25, 40, 43, 53]. Más aún, los algoritmos evolutivos pueden incorporar operadores basados en otras técnicas (p.e: en el temple simulado) [1, 25, 43]. A continuación describiremos los operadores genéticos básicos y algunos otros operadores.

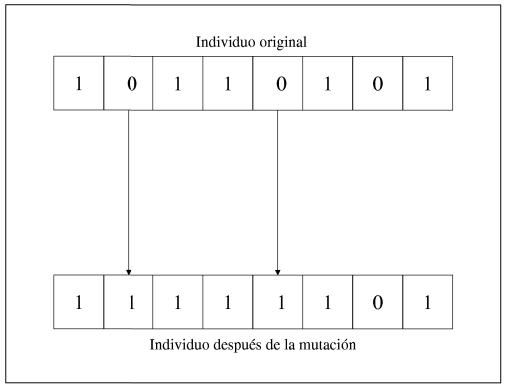

Figura 5.1: Aplicación del Operador Mutación

Mutación

El operador de mutación es un operador unario que simula el proceso evolutivo que ocurre en los individuos cuando cambian su estructura genética. Cuando sobre algún individuo se aplica mutación, se seleccionan aleatoriamente componentes (genes) de dicho individuo para ser modificados, también de forma aleatoria. Esto hace que a través del operador de mutación se puedan producir cambios en la estructura de un individuo que hagan que éste tenga un grado de adaptabilidad significativamente distinto (sea mejor o peor). La figura 5.1 muestra un ejemplo de la aplicación del operador mutación sobre los genes 2 y 5 de un individuo codificado de forma binaria. En el ejemplo que mostramos en la figura 5.1 se mutan dos genes; sin embargo, en la práctica el número de genes a mutar puede ser cualquiera, e incluso variar durante la ejecución del algoritmo evolutivo.

El operador de mutación que acabamos de describir representa la denominada *mutación clásica*, a la cual se le ha dado importancia teórica secundaria, manifestada a través de probabilidades de aplicación bajas respecto al cruce. Sin embargo, en la práctica es un operador imprescindible. Algunas variantes al operador clásico de mutación son [11, 24, 25, 40, 43, 53]:

- *Mutaciones sobre genes*: Cuando un gen puede tomar mas de dos valores (llamados también *alelos*), la mutación se realiza cambiando el valor actual por otro más o menos parecido.
- *Mutaciones no estacionarias*: En algunas ocasiones es conveniente reducir la tasa de mutación (probabilidad de mutación) según el programa progresa. La forma de implementar tal reducción depende en gran medida del problema; una manera sencilla es multiplicando la probabilidad de mutación por un factor de reducción cada cierto número de generaciones.
- *Mutaciones no uniformes*: Se dan distintas probabilidades de mutación a cada gen (o a cada bit dentro de un gen), dependiendo de su significado.

Cruce

El cruce o mezcla es un operador, normalmente binario, que permite expresar el proceso de apareamiento natural usando operaciones sencillas. Mediante el operador de cruce se toman diversos componentes de distintos individuos para generar con ellos un nuevo individuo, de tal manera de que herede características de sus padres. Su grado de adaptabilidad dependerá de la combinación de esas características. La figura 5.2 ilustra la aplicación del cruce entre dos individuos, con codificación binaria, en la cual se ha tomado como punto de cruce el cuarto gen (cuarto bit).

Desde el punto de vista práctico, el operador de cruce nos ofrece amplia variedad en cuanto al número de puntos de cruce y la selección de su ubicación. Así, se han propuesto diversas alternativas para el cruce [11, 24, 25, 40, 43, 53]:

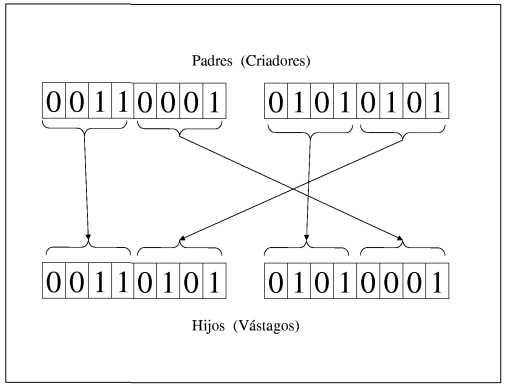

Figura 5.2: Aplicación del Operador cruce

- *Cruce multipunto*: Propone el cruce de los individuos entorno a dos o más puntos. Con este cruce se amplia las capacidades de intercambio de información, pero se pierde velocidad de convergencia.
- *Cruce segmentado*: Es un tipo de cruce multipunto donde el número de puntos es variable. El número fijo de puntos de cruce se sustituye por una probabilidad de segmentación. Por ejemplo, si la probabilidad de segmentación es de 0.2 el promedio esperado de puntos de cruce será de $0.2 * l$ (l = longitud del código) en cada par de progenitores.
- *Cruce uniforme*: Para cada bit del primer descendiente se decide, según una cierta probabilidad uniforme, de qué progenitor heredará su valor esa posición. El segundo descendiente recibe el correspondiente gen del otro progenitor.

Operadores Avanzados

En algunas circunstancias particulares los operadores genéticos básicos no son suficientes para hallar una solución satisfactoria a un problema dado, bien sea porque conducen a una convergencia prematura del algoritmo, o porque no pueden ser utilizados por el tipo de codificación de los parámetros seleccionado. También, a veces se desean explorar otros mecanismos de diversificación de la población. Para estos casos se han planteado operadores avanzados cuya aplicación depende de los problemas particulares. Algunos de estos operadores son [11, 24, 25, 40, 43, 53]:

- **Dominación**. Determina algunos componentes (genes) que son preponderantes en el individuo para su valor de aptitud, y da preferencia a esos componentes en otros operadores genéticos.

- **Segregación**. Es lo contrario a la dominación, en este caso los componentes segregados no influyen de manera determinante en el grado de adaptabilidad del individuo.
- **Translocación**. Se intercambian trozos del código genético de un individuo.
- **Inversión**. Este operador modifica la información genética de un individuo tomando dos componentes (genes) del mismo e intercambiando los valores de estos; es decir, al primero le asigna la información del segundo y al segundo la información del primero.
- **Duplicación**. Copia exactamente la información de un individuo a otro.

5.3.2 Mecanismos de Selección

Los mecanismos de selección se refieren a las políticas empleadas para la conformación de los *progenitores* [11, 24, 25, 53]. Mediante la selección se toman, a través de algún criterio de muestreo, k individuos de la población para ser usados como progenitores. Los criterios específicos de muestreo dependen del problema y de la experticia del implementador; sin embargo, existen tres criterios de uso común:

Muestreo por sorteo

Se asume que cada individuo de la población tiene una cierta puntuación, normalmente relacionada a su valor de aptitud, con la cual se calcula la probabilidad p_i que tiene cada individuo i para formar parte de la muestra (ver ecuación 5.1).

$$p_i = \frac{f_i}{\sum_{j=0}^{n} f_j} \tag{5.1}$$

donde:

n : tamaño de la población

f : función de aptitud

De esta manera, se dispone de un mecanismo simple para implementar el muestreo:

1. Se calculan las probabilidades acumulativas (q_i):
   $q_0 = 0$
   $q_i = q_{i-1} + p_i \ (\forall i = 1, ..., n)$
2. Se genera un número aleatorio α entre 0 y 1
3. Se elige el individuo i que cumpla con la condición:
   $q_{i-1} < \alpha < q_i$

Este algoritmo simple puede modificarse para seleccionar k individuos, para esto, se generan k números aleatorios en el paso 2 $(\alpha_1, ..., \alpha_k)$ y luego se ejecuta el paso 3 k-veces, tomando en la iteración i el valor de α_i correspondiente. Este mecanismo podría ocasionar que algunas individuos sean seleccionados varias veces, mientras que otros nunca sean seleccionados.

Muestreo universal

Este mecanismo también es conocido como la ruleta. La idea en este método es similar a la del muestreo por sorteo para k individuos, con la diferencia que en el muestreo universal sólo se genera un número aleatorio α y se utiliza éste para crear los números:

$$\alpha_i = \frac{\alpha + i - 1}{k} \quad (\forall i = 1, ...k)$$

Una vez obtenidos estos números, se aplica k-veces el paso 3 del muestreo por sorteo.

Muestreo por torneos

En este caso, se toman al azar m individuos de la población, y formará parte de la muestra el mejor de estos m individuos. Para seleccionar k individuos, se repite el proceso k-veces. En el muestreo por torneos no es necesario la asignación de puntuaciones ni el cálculo de probabilidades. No obstante, es necesario asignar un valor adecuado al parámetro m.

5.3.3 Mecanismos de Reemplazo

Los mecanismos de reemplazo permiten mantener la población constante [11, 24, 25, 53]. Estos mecanismos especifican la forma de "sumar"(esto no es una suma cartesiana) los n individuos existentes en la población con los s individuos producto de la reproducción, de manera de obtener una nueva población de n individuos. Entre los mecanismos de reemplazo, los de mayor uso y aceptación son:

Reemplazo directo

Los progenitores son reemplazados directamente por sus descendientes. Obviamente, este criterio solo puede usarse si $s < n$.

Reemplazo por similitud

Cada descendiente reemplaza aquel miembro de la población que más se le asemeja. Al igual que en el caso directo, debe cumplirse que $s < n$.

Reemplazo por inserción (,)

Este criterio puede usarse con variación según la relación entre s y n:

Si $s \leq n$: Se escogen s individuos de la población, los cuales serán sustituidos por los s descendientes. Esta escogencia se hace a través de algún criterio de muestreo, normalmente se toman los peores.

Si $s > n$: Se toma una muestra (con algún criterio) de tamaño n de los s descendientes. Bajo este esquema, cada individuo vive sólo por una generación.

Reemplazo por inclusión (+)

Se unen los n miembros de la población con los s descendientes, conformando una población auxiliar de tamaño $s + n$. Luego, de esta población auxiliar, se toma una muestra de tamaño n según algún criterio de selección (por ejemplo, los n mejores).

5.3.4 Esquema de Funcionamiento

El esquema general que describe el proceso de funcionamiento de los algoritmos evolutivos, puede ser representado a través del siguiente macroalgoritmo:

1. Generar una población inicial. Se genera aleatoriamente los individuos de la población.
2. Evaluar los individuos. Se evalúa la aptitud de todos los individuos de la población.
3. Seleccionar, a través de algún mecanismo, ciertos individuos de la población.
4. Modificar los **genes** de los padres seleccionados usando los operadores genéticos.
5. Evaluar los nuevos individuos.
6. Generar una nueva población con la existente y los individuos generados en el paso 4.
7. Verificar el criterio de convergencia, o regresar al paso 3.

Las principales ventajas de los algoritmos evolutivos son:

- Requieren poca información específica para funcionar; es suficiente un criterio de aptitud que ni siquiera debe ser expresado explícitamente (e.j un procedimiento), ni ser exacto.
- Es posible incorporar información específica; con lo cual se logra, aún con poca información, incrementar el rendimiento sin perjudicar seriamente la generalidad.

- Son métodos de aproximación sucesiva, pudiendo parar en cualquier momento para proporcionar buenas soluciones, no necesariamente óptimas.
- Al trabajar con una *población*, se hace posible el uso de criterios distintos a la *aptitud*.
- Son algoritmos de complejidad casi lineal, es decir, el número de operaciones crece proporcionalmente con el tamaño del problema. Por lo tanto, se pueden resolver problemas con un gran número de variables usando una potencia computacional modesta.

Sobre la base del esquema descrito anteriormente, se han desarrollado cinco paradigmas [5, 16, 17, 25, 43]:

- **Algoritmos Genéticos**. Se tiene una población de cadenas de caracteres o enteros binarios que evolucionan a través de transformaciones, unitarias y binarias, y de un proceso de selección.
- **Programas Evolutivos**. Evoluciona una población de estructuras de datos, sometiéndolas a una serie de transformaciones específicas y a un proceso de selección.
- **Estrategias Evolutivas**. Se hace evolucionar una población de números reales que codifican las posibles soluciones de un problema numérico y los tamaños de salto.
- **Programación Evolutiva**. La población esta constituida por máquinas de estados finitos que se someten a transformaciones unitarias.
- **Programación Genética**. La población esta constituida por programas que resuelven un problema. El objetivo no es encontrar la mejor solución, sino el mejor programa para resolverlo.

5.4 Algoritmos Genéticos

Los Algoritmos Genéticos (**AGs**) son algoritmos de búsqueda que emulan la evolución biológica en el computador, siguiendo un proceso *evolutivo inteligente* sobre individuos, basado en la utilización de operadores genéticos. Los AGs fueron propuestos por John Holland a mediados de los años 70 [11, 23, 24, 28, 29, 40, 41, 42, 53]. Desde entonces, Holland y sus colaboradores han realizado diversos estudios en la Universidad de Michigan, con dos objetivos principales: abstraer y explicar rigurosamente los procesos adaptativos de los sistemas naturales, y diseñar sistemas de software que conserven los mecanismos importantes de los sistemas naturales.

La idea principal de Holland fue usar características propias del proceso evolutivo natural de las especies sobre un algoritmo programable, para resolver Problemas

Difíciles (NP-Completos). La base fundamental de los AGs es simular la evolución de los **individuos** de cierta población, con el objetivo de conseguir mejores individuos.

El procedimiento general de todo AG consiste en mantener una población de potenciales soluciones a través de las generaciones; los individuos de esa población son representados por *cromosomas*; los cuales son, básicamente, cadenas de caracteres que se transforman a través de la evolución. En la practica, es posible implementar este modelo genético de computación codificando los individuos como cadenas de *bits* o caracteres. Así, las operaciones de mutación, cruce, etc., pueden implementarse como simples manipulaciones de *bits*.

El objetivo final de los AGs es encontrar la mejor solución posible, partiendo de soluciones escogidas aleatoriamente. Para eso, se utilizan operadores evolutivos como mecanismos de búsqueda de nuevas, quizás mejores, soluciones. El procedimiento continua hasta cumplirse algún criterio de convergencia o parada, tal que el algoritmo quede "atrapado"en un óptimo local.

Los campos de aplicación de los AGs cubren diversas áreas, principalmente:

- Búsqueda y Optimización.
- Toma de decisiones.
- Clasificación.
- Aprendizaje Genético.

Entre los cinco paradigmas fundamentales de la CE, los AGs son los que ocupan el principal lugar. Entre las razones para esto pueden mencionarse [11, 23, 24, 28, 29, 41, 42, 53]:

- Reúnen, de manera natural, todas las ideas fundamentales de la CE.
- Poseen la mayor base teórica, la cual es sencilla y con mayores posibilidades de ampliación.
- Requieren menor conocimiento específico del problema, lo que les da mayor versatilidad.
- Tal vez, como consecuencia de las anteriores, son el paradigma más usado de la CE, y junto con las Redes Neuronales Artificiales, los más usados de toda el área de Computación Inteligente.

Esto ha conllevado a que muchas veces al hablar de los Algoritmos Evolutivos, por omisión se este hablando de AGs, y que los avances de la CE se planteen teniendo en mente este paradigma. Por estas razones, nos extenderemos más en la explicacioón de los AGs.

5.4.1 Características de los AGs

Los AGs poseen características que determinan su funcionamiento y su aplicabilidad en determinados problemas. Estas son [11, 23, 24, 28, 29, 41, 42, 53]:

Búsqueda Ciega. No disponen de ningún conocimiento específico del problema, de esta manera, la búsqueda esta orienta únicamente por los valores de la función objetivo.

Codificación de Parámetros. Se trabaja con la codificación de un conjunto de parámetros (o un subconjunto de ellos) y no con los parámetros en sí. Los AGs requieren que el conjunto de parámetros del problema sean codificados en una cadena de longitud finita sobre algún alfabeto. Dicha codificación determina la representación de los individuos. Típicamente, este alfabeto es binario. Es posible que para un mismo problema se tengan múltiples formas de codificar los parámetros.

Búsqueda Basada en Población. Se busca desde una población de puntos y no desde uno solo. Parten de un conjunto de soluciones (población de soluciones) y no de una solución inicial única, de esta manera, la probabilidad de encontrar un óptimo local malo se reduce.

Función Objetivo como Elemento a Optimizar. Se usa una función objetivo o función de evaluación para definir lo que se desea optimizar, sin requerir más información derivada o auxiliar. Esta función permite evaluar la calidad de los individuos en cada generación, partiendo de los valores codificados de los parámetros (genera algún valor cuantitativo). Sin embargo, en la mayoría de los casos se transforma la función de evaluación en una función de aptitud (*"fitness"*) para controlar la diversidad de la población y hacer que las evaluaciones siempre tomen valores positivos. La construcción de la función de aptitud se logra por desplazamiento y/o escalamiento de la función de evaluación [11, 24, 41, 53].

Operadores Aleatorios. Se hace uso de operadores estocásticos para las transformaciones, en vez de reglas de transición determinísticas.

Estas características convierten a los AGs en métodos de búsqueda robustos; esto es, logran mayor eficiencia sin perdida de generalidad. Otra característica de los AGs es su capacidad para intercambiar, estructuradamente, información en paralelo, lo cual puede denominarse paralelismo implícito. Explicaremos esto de la siguiente manera: los AGs procesan externamente cadenas de códigos; no obstante, internamente procesan similitudes entre cadenas, así que al procesar cada cadena realmente están procesando todos los patrones similares que contiene esa cadena. Esta característica hace a los AGs mucho más eficiente que otros métodos de búsqueda ciega.

5.4.2 Implementando AGs

En esta parte daremos una guía de los pasos a seguir y los problemas que deben resolverse para implementar un AG para un problema dado.

Selección de la Codificación. Como ya dijimos, los AGs emplean normalmente una codificación basada en cadenas binarias. En esta etapa, debe hacerse corresponder a cada punto del espacio de soluciones con una cadena binaria. Cada cadena representará a un individuo. Si suponemos que un individuo Y posee m genes (o características), se requieren para la codificación l bits, donde l está determinado por la expresión: $l = \sum_{i=0}^{m} L_i$, con $L_i = log_2 a_i$, sabiendo que el gen i tiene a_i posibles valores (denominados alelos). Cabe resaltar la diferencia entre la estructura de los individuos y el contenido de cada uno, por cuanto la estructura se deriva de la representación (es igual para todos los individuos); mientras que el contenido es derivado de la codificación (el valor binario que tiene cierto individuo).

Elección de la población inicial. Consiste en determinar los individuos que formarán parte de la población en la primera generación. En general, la población inicial debe ser lo más variada posible, con una distribución de aptitudes suficientemente uniforme para no tener convergencia prematura. Aunque algún conocimiento específico del problema podría ayudar a la selección de una población inicial factible y cercana al óptimo, casi siempre ocurre que la misma es escogida al azar por falta de dicha información.

Criterio de parada. Deben definirse las condiciones bajo las cuales consideramos que el AG a finalizado; bien porque a dado una solución aceptable o porque la búsqueda a fracasado y no es necesario continuar. El mecanismo habitualmente utilizado es el de número de iteraciones (generaciones). Este mecanismo resulta demasiado arbitrario y poco efectivo en muchos casos, por lo cual también se usan como criterios de parada:

- *La similitud de las estructuras.* Se refiere al parecido entre los individuos de la población.
- *La similitud de contenido.* Se refiere a la evolución del algoritmo, es decir, si después de cierto número de iteraciones el AG ha propuesto una mejor solución.

Funciones de Evaluación. Es necesario especificar la función de evaluación que usará el AG para medir la aptitud de cada individuo en un problema dado. De manera ideal, la función de evaluación coincide con el objetivo a maximizar o minimizar. Sin embargo, en algunos casos no se puede concretar la función matemática que define el objetivo. En estos casos, se hace necesario replantear el objetivo del problema para encontrar algún esquema equivalente que lo represente y que si pueda ser llevado a una función de evaluación implementable.

Esto último también es válido en problemas donde el objetivo puede especificarse bien matemáticamente, pero su implementación computacional es muy difícil.

Selección de los operadores genéticos. La influencia de los operadores genéticos es fundamental en el funcionamiento de los AGs; por lo tanto, es importante hacer una buena selección de los mismos. Habitualmente, se utilizan dos operadores, el de cruce y el de mutación. No obstante, cuando se tiene conocimiento específico del problema es posible incorporar otros operadores genéticos o modificar operadores ya existentes con el fin de mejorar el funcionamiento del AG, tal que se pueda lograr mejores soluciones y/o para disminuir el tiempo de ejecución.

Criterios de Selección de padres. Con el objetivo de dirigir el proceso de búsqueda, la selección debe favorecer a los individuos de mejor aptitud. Para cumplir este objetivo existen diversos mecanismo de selección. Entre estos mecanismos destaca el elitesco, el cual basa la selección en el valor de aptitud de cada individuo, dando mayor probabilidad de selección a los más aptos. Otros mecanismos son: aleatorio, por sorteo, por restos, por ruleta (ver sección 1.3.2). Todos los mecanismos admiten modificaciones para dar oportunidades a todos los individuos, o excluir algunos, o asegurarse que no se repita una selección.

Criterios de reemplazo. No siempre los criterios con que se seleccionan los padres son los mismos a los usados para seleccionar a los sobrevivientes; por lo tanto, debemos especificarlos por separado. Entre estos criterios destacan: el directo, por similitud , por inserción y por inclusión (explicados en la sección 5.3.3).

Manejo de los individuos no factibles. En algunos problemas no es posible establecer un relación punto a punto entre todos los individuos generados por el AG y las soluciones factibles (puntos del espacio de soluciones). Esto implica que existirán individuos que representan soluciones no válidas para el problema; por lo cual, será necesario establecer mecanismos para su distinción y tratamiento. Estos mecanismos, necesariamente incorporan conocimiento específico a los AGs. Para aclarar este aspecto, mencionaremos algunas técnicas que permiten el manejo de individuos no factibles:

- *Penalización*: A los individuos que resultan ser no factibles se les reduce, significativamente, su aptitud.
- *Reparación o corrección*: Cuando se encuentre un individuo no factible, se ejecuta un procedimiento que lo corrige. La mayor dificultad de esta técnica es que se necesita un procedimiento de corrección para cada problema.
- *Decodificación*: Se busca una codificación que haga no probable la aparición de individuos no factibles. Adicionalmente, se deben modificar los operadores genéticos para que siempre generen individuos factibles.

Parámetros de entonación. Además de los pasos y criterios mencionados anteriormente, los AGs requieren otra serie de parámetros para su funcionamiento, tales como tamaño de la población, probabilidades de uso de los operadores genéticos, número de iteraciones (sí es este el criterio de parada), etc.

Para ilustrar el mecanismo de implementación de un AG para un problema dado; especificaremos a continuación un ejemplo con el tratamiento de cada uno de los pasos especificados anteriormente.

Supongamos que se desea maximizar la función:

$$F(x) = Sen\left(\frac{x\pi}{256}\right)$$

en el intervalo [0,255]. En este problema, lo que realmente se desea es encontrar el valor de x que maximiza la función, puesto que el máximo valor de la función es conocido.

- **Codificación**. Como las soluciones varían entre 0 y 255, se pueden utilizar cadenas binarias de longitud 8. Por ejemplo, un individuo se verá así:

  00010100

- **Función de evaluación**. La función de evaluación es la misma función dada, y se utilizará como función de aptitud la diferencia entre el valor para la función dada y el valor máximo (uno):

  $$g(x) = 1 - F(x)$$

  De esta manera, el problema se transforma en un problema de minimización de $g(x)$. Para poder realizar la evaluación de un individuo es necesario calcular su valor decimal. La tabla 5.1 muestra los individuos de una población de tamaño 8, con sus valores para x y $f(x)$.

Individuo	x	$f(x)$
10111101	189	.733
11011000	216	.471
01100011	99	.937
11101100	236	.243
10101110	174	.845
01001010	74	.788
00100011	35	.416
00110101	53	.650

Tabla 5.1: Población para el problema de maximización de la función $Sen\left(\frac{x\pi}{255}\right)$

5.4.3 Fundamentos Matemáticos de los AGs

Con las secciones anteriores daremos por satisfechos los requerimientos de los lectores que solo necesitan una visión global de los AGs y alguna guía para su implementación en un problema específico. Sin embargo, para aquellos que deseen entender con mayor detalle el funcionamiento y las bases matemáticas de los AGs, presentaremos a continuación los resultados que se han obtenido en el estudio de los AGs desde el punto de vista matemático [11, 23, 24, 28, 29, 41, 53].

Normalmente, los AGs codifican los individuos como cadenas binarias, por lo tanto se utiliza el alfabeto $V = \{0,1\}$. Si extendemos el alfabeto V por la agregación del símbolo '$*$', tendremos el alfabeto $V+ = \{0,1,*\}$, donde el símbolo '$*$' representa el símbolo comodin o caracter *no importa*, por lo cual puede valer 0 o 1. Así, definiremos un patron H como una secuencia de símbolos del alfabeto $V+$ y A como una secuencia de símbolos del alfabeto V.

Para una codificación de 8 bits, por ejemplo, tendríamos que un valor posible para H sería $1*100**1$ y la cadena $A = 10100111$ representa una instancia del patron H. Las siguientes dos propiedades de los patrones serán de interés para definiciones posteriores:

- **Orden**. El orden de un patron H, denotado como $o(H)$, es el número de posiciones del patron que están ocupadas por un 1 o por un 0. En el ejemplo presentado anteriormente, donde $H = 1*100**1$, $o(H) = 5$.
- **Longitud**. La longitud de un patron se denota por $\delta(H)$, y representa la distancia entre el primer y último caracter de la cadena con valor de 0 o 1. Para el ejemplo $H = 1*100**1$, $\delta(H) = 8 - 1 = 7$

Los patrones y sus propiedades permiten clasificar y analizar las similitudes entre cadenas de caracteres. Además, sirven de base para estudiar los efectos de la reproducción y de los operadores genéticos en los bloque de información contenidos en la población. A continuación presentaremos esos dos efectos:

Efectos de la reproducción

Definición 1. *Para valores promedio de aptitud altos de un patron, el número de instancias de dicho patron en la población crecerá; mientras que para valores bajos disminuirá.*

Llamemos H a un determinado patron, $\mathbf{A}(t)$ a la población en un tiempo t y, $m(H,t)$ al número de instancia del patron H en $\mathbf{A}$ en el momento t. En el proceso de reproducción, una cadena A_i participará (formará parte de los padres) con una probabilidad p_i, definida por la ecuación 5.1.

Luego de la reproducción y el reemplazo, en una población de tamaño n, tendremos que $m(H,t+1)$ (la cantidad de instancia del patron H en el tiempo $t+1$) estará dada por la siguiente ecuación [11, 23, 24, 28, 29, 41, 53]:

$$m(H, t+1) = m(H, t)(f(H)/\bar{f}) \tag{5.2}$$

donde:

$f(H)$= la aptitud promedio de las cadenas que son instancias de H en el tiempo t.

$\bar{f} = \frac{\sum f_j}{n}$ (El promedio de aptitud de toda la población)

Si en la ecuación 5.2 suponemos que para los patrones que tienen valores promedios de aptitud alto, tal promedio es superior al de la población en una cantidad $c\bar{f}$, siendo c una constante positiva, podemos reescribir dicha ecuación de la siguiente manera:

$$m(H, t+1) = m(H, t)(\bar{f} + c\bar{f})/\bar{f} = (1+c).m(H, t) \tag{5.3}$$

De la ecuación 5.3 tenemos que:

$$m(H, t) = m(H, 0)(1+c)^t \tag{5.4}$$

Esta última ecuación muestra que la reproducción incrementa exponencialmente el número de instancias de patrones que tienen promedio de aptitud altos. Por otro lado, para patrones que tienen promedio de aptitud bajos, el termino $f(H)/\bar{f}$ en la ecuación 1.2 va tomando valores menores que 1, lo que hace disminuir el número de instancias para esos patrones (q.e.d).

Efecto de los operadores genéticos

Si únicamente se incrementa el número de instancias de patrones que tienen promedio de aptitud altos, sin explorar nueva información, tendríamos AGs que simplemente copian estructuras sin explorar regiones nuevas del espacio de búsqueda. Es en este último aspecto donde intervienen los operadores genéticos.

Definición 2. *El número de instancias de un patron en la población, cuando se usa el operador de cruce, es directamente proporcional a la probabilidad de supervivencia p_s del patron.*

Para darle sentido a esta definición, daremos la descripción de la probabilidad de supervivencia de un patron. La probabilidad de supervivencia de un patron H con longitud $\delta(H)$, en el caso de usarse el operador de cruce, viene dada por la expresión [11, 23, 24, 28, 29, 41, 53]:

$$p_s = 1 - \frac{\delta(H)}{l-1} \tag{5.5}$$

donde:
l es la longitud del código binario utilizado
$l-1$ representa las posibles selecciones para el punto de cruce.

Si el operador de cruce es usado con una probabilidad p_c, entonces la expresión 5.5 quedaría de la siguiente forma:

$$p_s \leq 1 - p_c.\frac{\delta(H)}{l-1} \tag{5.6}$$

Con las expresiones 5.6 y 5.2 obtendremos una expresión que nos permite estimar el efecto combinado de la reproducción y el cruce sobre el número de instancias de un patron. Esta expresión viene dada de la siguiente manera:

$$m(H, t+1) \geq m(H, t)\frac{f(H)}{\bar{f}}[1 - p_c.\frac{\delta(H)}{l-1}] \tag{5.7}$$

Definición 3. *Para que un patrón H sobreviva, en el caso de usarse el operador de mutación, todas las posiciones que son $1s$ o $0s$ en H deben sobrevivir.*

La definición 3 implica que la probabilidad de sobrevivencia de un patrón esta en función de su orden $o(H)$. Si suponemos que un bit (alelo) es mutado con probabilidad p_m; entonces podremos decir que la probabilidad de sobrevivencia de un alelo es $1-p_m$; por lo tanto, la probabilidad de sobrevivencia de un patron usando el operador de mutación viene dada como $o(H)$ veces el producto de $1 - p_m$, esto es:

$$(1 - p_m)^{o(H)} \tag{5.8}$$

Dicha probabilidad puede ser aproximada al valor $1 - o(H).p_m$, para valores pequeños de p_m ($p_m \ll 1$).

De esta manera, podemos concluir que el efecto combinado de la reproducción, el cruce, y la mutación, sobre el número de instancias que tendrá un patron en la próxima generación esta dado por la siguiente expresión (suponiendo efectos mínimos de los productos cruzados):

$$m(H, t+1) \geq m(H, t)\frac{f(H)}{\bar{f}}[1 - p_c.\frac{\delta(H)}{l-1} - o(H).p_m] \tag{5.9}$$

5.5 Programas Evolutivos

Los Programas Evolutivos (**PEs**), o programas de evolución, son una estrategia de **optimización estocástica** similar a los AGs, pero con énfasis específico en los operadores genéticos tales como se observan en la naturaleza y en la estructura de datos que utiliza adaptada al problema [35, 40, 43]. Por esto, a diferencia de los AGs, los PEs no son restrictivos en cuanto a la representación del problema. Mientras en los AGs se hace necesario una codificación de las soluciones del problema; en los PEs, tal representación se hace de forma directa.

La propuesta de los PEs fue hecha por Michalewicz en 1994 [40]; cuando propuso incorporar conocimiento específico del problema a resolver en las estructuras de datos. Así, los PEs son, en una buena medida, métodos que incorporan directamente conocimiento específico a los AGs; puesto que permiten la utilización de estructuras de datos naturales. Esto permite, a su vez, la utilización de operadores genéticos sensibles al contexto para mejorar la eficiencia del algoritmo de búsqueda sin una gran perdida de la propiedad de generalización. Además de perder un poco de generalización, con la incorporación de conocimiento específico también se pierde el paralelismo implícito (basado en alfabetos de símbolos mínimo). Sin embargo, esta pérdida es compensada con el procesamiento de información más útil.

Desde el punto de vista práctico, veremos a los PEs como una modificación de los AGs que comparten exactamente su estructura básica, con las distinciones siguientes:

1. La representación es más natural. Por lo cual, esta más cercana al dominio del problema. Este aspecto puede lograrse en las implementaciones, usando las estructuras de datos más adecuadas para cada problema (arreglos de números reales, pilas, listas, conjuntos, árboles, grafos, etc.)

2. Los operadores genéticos son específicos al contexto del problema. Por lo tanto, pueden estar adaptados al dominio con el fin de generar resultados que mantengan la estructura de los individuos.

5.5.1 PEs en optimización paramétrica

La utilización de AGs en problemas de optimización paramétrica da resultados mediocres. Por una parte, la codificación binaria requiere individuos de tamaño muy grande cuando se desea alta precisión en la representación; y por otra parte, los operadores genéticos no permiten realizar ajustes finos en las etapas finales de la evolución [43]. De esta manera, se requería un esquema que permitiese representar a los individuos como arreglos de números reales donde cada uno de esos números sería un gen. Desde del punto de vista de representación, el uso de arreglos de números reales en punto flotante nos proporciona tres ventajas prácticas:

1. La notación de punto flotante cuantifica el espacio de búsqueda mediante una rejilla logarítmica, con lo que se logra incrementar la capacidad de ajuste fino.

2. Se simplifica el manejo de las restricciones, puesto que es más natural.

3. Se facilita la incorporación de conocimiento específico a través de la utilización de operadores apropiados para el dominio del problema.

5.5.2 PEs en optimización combinatoria

Hasta ahora hemos mencionado a los PEs como métodos eficaces para la resolución de problemas de optimización numérica. Sin embargo, existen algunos problemas que se caracterizan por la existencia de soluciones que se componen secuencialmente, es decir, dado un conjunto de elementos se pueden obtener diferentes arreglos ordenados de estos, permitiendo una basta cantidad de posibilidades; los cuales se denominan problemas de optimización combinatoria [19, 6]. El entendimiento de los problemas de optimización combinatoria es bastante sencillo, más no así su solución debido a que algunos de ellos son problemas de tipo NP-completos. Los problemas NP-completos tienen dos propiedades, la primera de ellas es que su complejidad es NO polinómica (el tiempo de ejecución crece exponencialmente con el tamaño del problema), y la segunda es el hecho de que el problema puede ser transformado en otro problema NP-completo [27, 35, 40, 43].

Desde el punto de vista práctico, los PEs permiten implementar buenas soluciones a problemas de optimización combinatoria con las ventajas de su generalidad y la facilidad de implementaciones similares para distintos problemas. A continuación presentaremos un ejemplo clásico de un problema de optimización combinatoria y mostraremos el esquema general de solución basado en un PE.

Agente Viajero

El problema del agente viajero es un problema clásico de optimización combinatoria que puede describirse como [27]:

Dadas N ciudades, el viajero de comercio debe visitar cada ciudad una vez teniendo en cuenta que el costo total del recorrido debe ser mínimo.

Una de las instancias más comunes del problema del agente viajero es el *problema euclidiano*, donde el costo del recorrido viene dado por la suma ordenada de las distancias entre las ciudades visitadas. Este problema puede ser expresado de la siguiente manera:

$$G = (N, A)$$

donde:
$N = \{1, ..., n\}$: conjunto de n nodos (vértices),
$A = \{a_{ij}\}$: matriz de adyacencia.

Además, se define la matriz de distancias D:

$$\{d_{ij}\} = \begin{cases} \infty & \text{Si } a_{ij} = 0 \\ l_{ij} & \text{Si } a_{ij} = 1 \end{cases}$$

donde:
l_{ij}: distancia entre las ciudades i y j

Suponiendo que las ciudades son numeradas desde 1 hasta n, una solución al problema puede expresarse a través de una matriz de estado E que indica el orden en que son visitadas las ciudades. Esta matriz tendrá en las filas el orden de visita de las ciudades y en las columnas las ciudades.

$$\{e_{ij}\} = \begin{cases} 1 & \text{Si la ciudad j fue la } i\text{-esima ciudad visitada} \\ 0 & \text{En otro caso} \end{cases}$$

La matriz de estado E permitirá verificar la validez de una solución, de tal manera de asegurar que todas las ciudades son visitadas una y solo una vez. Esto se hace por la verificación de las siguientes restricciones:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} e_{ij} = n \tag{5.10}$$

$$\sum_{i=1}^{n} e_{ij} = 1; \quad \forall j \in \{1,\ldots,n\} \tag{5.11}$$

$$\sum_{j=1}^{n} e_{ij} = 1; \quad \forall i \in \{1,\ldots,n\} \tag{5.12}$$

Así mismo, la matriz E permite definir un arreglo unidimensional V de dimensión n, dicho arreglo contendrá en cada posición la ciudad que fue visitada en dicha posición.

$$v_j = i \text{ (Si la ciudad } i \text{ fue la } j\text{-esima ciudad visitada)}$$

Por último, se propone una función objetivo compuesta por dos partes, una correspondiente a los costos relativos a las distancias recorridas y la otra relativa al grado de validez de la solución[1]. Tales funciones serían:

$$F_1 = \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{n} l_{ik}e_{ij}e_{kj+1} \tag{5.13}$$

$$F_2 = C\left(\left|\sum_{i=1}^{n}\sum_{j=1}^{n} e_{ij} - n\right| + \sum_{i=1}^{n}\left|\sum_{j=1}^{n} e_{ij} - 1\right| + \sum_{j=1}^{n}\left|\sum_{i=1}^{n} e_{ij} - 1\right|\right) \tag{5.14}$$

[1]Este componente representa el manejo de soluciones No-factibles

$$F_C = F1 + F2 \tag{5.15}$$

donde:

$$C = n* \text{ Máx}(l_{ik}) \text{ factor de penalización.}$$

A su vez, este factor de penalización representará el valor mínimo de F_C para una solución No-Factible; con lo cual se simplifica su identificación.

El problema consiste en encontrar el recorrido por las ciudades que minimice el valor de la función de costo F_C. A continuación presentamos los aspectos de mayor relevancia que deben tomarse en cuenta para la resolución de esta instancia del problema del agente viajero usando un PE:

1. **Representación del problema**. Para la representación computacional del grafo de las ciudades, se recomienda el uso de una matriz simétrica de distancias para almacenar los valores en unidades de distancia que separan las ciudades. Esta representación involucra el manejo de arreglos bidimensionales, la cual se logra con una implementación sencilla y directa en la mayoría de los lenguajes de programación.

2. **Codificación de los individuos**. En el caso del agente viajero, una solución viene dada como un recorrido que cumple con las restricciones del problema. De esta manera, la forma de representación de los individuos (soluciones) se puede realizar por medio del arreglo unidimensional V de longitud n (número de ciudades) presentado antes. Dicho arreglo contiene el recorrido ordenado, es decir, en la posición 1 irá la primera ciudad visitada, en la posición 2 la segunda, y así sucesivamente. Para utilizar esta representación se supone que las ciudades son numeradas de 1 a n.

3. **Función Objetivo**. La definición de la función objetivo influye de forma directa en el tiempo de ejecución del PE. Se recomienda el uso de la función (5.13) puesto que tiene el mismo poder de representación que la función (5.15), y su complejidad es menor. En este caso, la verificación de las restricciones para el problema deben realizarse a través de los operadores genéticos utilizados.

4. **Operadores genéticos**. Si se utiliza la función (5.13), los operadores genéticos deben realizar el tratamiento de soluciones no factibles. Así, los operadores cumplen dos funciones: la de evolución genética y la de generar soluciones válidas. Un ejemplo de un posible operador genético a utilizar es el operador de inversión:

   - Inversión. En este caso, se toma aleatoriamente una posición (j) (con un valor de ciudad c_1) y se genera un nuevo valor que indica la nueva ciudad (c_2) que será visitada en esa posición, también de forma aleatoria. Luego, se busca este valor c_2 de ciudad en el arreglo (el cual ocupa una posición i), y a esa posición i se le asigna el valor c_1.

5.6 Estrategias Evolutivas

Las Estrategias Evolutivas (**EEs**) son, en esencia, métodos estocásticos con paso adaptativo, que permiten resolver problemas de optimización paramétrica [2, 3, 4, 34, 44]. A estos métodos se le han incorporado procedimientos propios de la CE que los han convertido en un paradigma más de dicha metodología. Con tal mezcla, las EEs pueden definirse como algoritmos evolutivos enfocados hacia la optimización paramétrica que utilizan una representación a través de vectores reales, una selección determinística y operadores específicos de mutación y cruce. La idea original de este esquema fue desarrolla por Rechenberg y Schwefel en Alemania [2, 3, 4, 34, 44].

Las estrategias evolutivas pueden dividirse en simples y múltiples; en las siguientes secciones explicaremos cada una.

5.6.1 EEs Simples

Las EE simples son consideradas como procedimientos estocásticos de optimización paramétrica con paso adaptativo; esta característica las hace, de cierto modo, similares al *temple simulado* [2]. En este caso, se hace evolucionar un sólo individuo usando únicamente un operador genético, la mutación. Son relativamente sencillas, y se denominan también, EEs de dos miembros. Debido a que evoluciona solamente un individuo, no son consideradas estrictamente como métodos evolutivos. A pesar de ser muy sencillas, presentan gran utilidad práctica y han sido utilizadas, con algunas mejoras, para resolver problemas reales en diversas áreas.

En términos de representación, las EEs simples se denotan como $(1+1)-$EE, ya que se utiliza un reemplazo por inclusión. Pueden ser descritos de la siguiente manera:

El individuo se representa a través de un par de vectores reales:

$$v := (x; s)$$

El vector x representa una solución (un punto en el espacio de búsqueda), y s es un vector de desviaciones típicas usado para la mutación. Como tenemos un único posible progenitor no habrá selección, y como ya dijimos, el criterio de reemplazo se basa en sustitución simple del progenitor en caso de que el descendiente tenga mejor *aptitud*.

En términos prácticos, la evolución se implementa a través de la siguiente expresión:

$$x[t+1] = \begin{cases} x[t] + N(0, s[k]) \; ssi \; x[t+1] \succ x[t] \; \wedge x[t+1] \in \chi \\ x[t] \;\; de\, lo\, contrario \end{cases} \tag{5.16}$$

[2] Temple simulado: Método de optimización combinatoria que utiliza conceptos físicos de "temperatura"y "energía"para representar y resolver problemas de optimización. Esta basado en el método de Monte Carlo, simulando el comportamiento de ciertos sistemas físicos, tales como el gas, en presencia de baños calientes (ver [1, 31] para más detalles)

donde:

$N(0, s)$: Representa un vector de números aleatorios gaussianos independientes con medias 0 y un vector de desviaciones típicas s (puede usarse cualquier otra distribución, por ejemplo, la *lognormal*), donde cada uno de los elementos del vector s es la desviación típica a usar por cada uno de los elementos del vector x.

χ: Representa el espacio de soluciones.

Las mejoras propuestas a las EEs simples se han enfocado al operador de mutación y al manejo de las desviaciones típicas.

5.6.2 EEs Múltiples

Las EEs múltiples surgen como respuesta a las debilidades de las EEs simples, las cuales tienden a converger hacia subóptimos. En las EEs múltiples existen múltiples individuos (población), y se producen en cada generación varios nuevos individuos, usando tanto mutación como cruce [44]. Se denotan como $(\lambda \dagger \mu)$-EE, donde λ es el tamaño de la población y μ es el tamaño de la descendencia. Los símbolos $\dagger$ representan los dos posibles mecanismos de reemplazo: de tipo por inclusión $(+)$ o de tipo por inserción $(,)$.

Como mencionamos anteriormente, en las EEs Múltiples se incorpora el operador de cruce (también puede usarse cualquier otro operador). Se usa normalmente el cruce promedio, el cual genera un único descendiente de dos padres, promediando los valores de estos. Si suponemos que tenemos dos progenitores:

$$P_1 = [(x_1, \ldots, x_m); (s_1, \ldots, s_m)]$$

$$P_2 = [(x'_1, \ldots, x'_m); (s'_1, \ldots, s'_m)]$$

Con el operador de cruce promedio obtendremos un descendiente de la siguiente manera:

$$\left[\left(\frac{1}{2}(x_1 + x'_1), \ldots, \frac{1}{2}(x_m + x'_m)\right); \left(\sqrt{(s_1)^2 + (s'_1)^2}, \ldots, \sqrt{(s_m)^2 + (s'_m)^2}\right)\right]$$

En cuanto a los criterios de reemplazo, siempre se usa un esquema determinista, pudiéndose utilizar una estrategia de inserción o de inclusión. Debido a esto, existen dos tipos de EEs Múltiples:

- (λ, μ)-**EE** (Reemplazo por inserción):
  - Cuando $\mu \leq \lambda$. Los μ descendientes reemplazan a los μ peores miembros de la población.

  - Cuando $\mu > \lambda$. La nueva población será formada por los λ mejores miembros de los descendientes.

- $(\lambda + \mu)$**-EE** (Reemplazo por inclusión):
  Se forma una población con los μ descendientes y los λ progenitores para luego seleccionar de esta los mejores λ individuos.

5.6.3 Relación de las EEs con los AGs

Los puntos de inicio de las EEs y los AGs son diferentes, puesto que parten de enfoques distintos. No obstante, los desarrollos realizados en cada área las ha ido acercando de manera significativa, y ahora podemos decir que las EEs son un tipo especial de AGs, o viceversa. Presentaremos a continuación las diferencias más significativas entre ambos métodos:

- Los AGs son mucho más generales que las EEs; como consecuencia de esto las EEs son más fuertes por la incorporación de conocimiento específico.
- El operador fundamental en las EEs es la mutación (la alteración de propiedades), mientras que en los AGs es el cruce (intercambio de información).
- Los AGs realizan búsqueda global, presentando dificultades para el ajuste fino; mientras que en las EEs se presenta exactamente lo contrario.

5.7 Programación Evolutiva

A principios de la década de los 60, L. Fogel propuso un mecanismo que hacía evolucionar un conjunto de *Máquinas de Estados Finitos*, el cual se denominó Programación Evolutiva [14, 15, 16, 17]. Antes de continuar la presentación de la Programación Evolutiva, aclararemos los aspectos básicos de las Máquinas de Estados Finitos.

Una Máquina de Estados Finitos es un transductor que puede ser estimulado por un alfabeto finito de símbolos de entrada, que puede responder a un alfabeto finito de símbolos de salida, y que posee algún número finito de estados internos diferentes. La figura 5.3 muestra una Máquina de Estados Finitos de tres estados, donde los números representan símbolos de entrada, las letras griegas símbolos de salida, y los caracteres en mayúscula representan los estados (tomada de [16]).

Para la máquina mostrada en la figura 5.3, mostramos en la tabla 5.2 un ejemplo de respuesta para una secuencia de símbolos de entrada dada, partiendo del estado **C**.

En la programación evolutiva, un conjunto de máquinas de estado finito (son los individuos) se expone a un ambiente, esto es, a una secuencia de símbolos observados en un momento dado (por ejemplo, a la secuencia de símbolos de entrada (0,1,1,1,0,1)). Para cada máquina de Estados Finitos, por cada símbolo de entrada que se le presenta se determina el símbolo de salida que da y se compara con el del ambiente (también

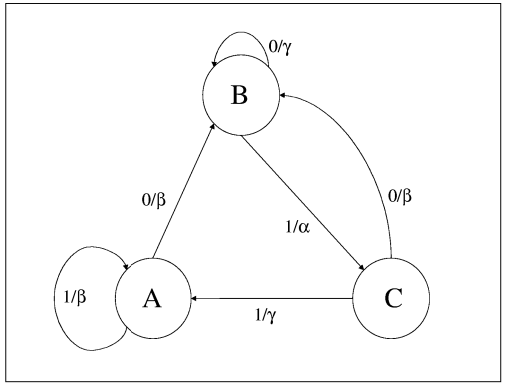

Figura 5.3: Máquina de Estados Finitos de tres estados (A, B, C), dos símbolos de entrada (0,1) y tres símbolos de salida (α, β, γ)

Estado Actual	C	B	C	A	A	B
Símbolo de entrada	0	1	1	1	0	1
Próximo estado	B	C	A	A	B	C
Símbolo de salida	β	α	γ	β	β	α

Tabla 5.2: Respuesta de la máquina de estados finitos para la cadena de entrada (0,1,1,1,0,1)

puede compararse su próximo estado con el que corresponde en el ambiente). Así, la predicción de cada máquina de Estados Finitos es medida con respecto al ambiente, como una función error (error absoluto, error cuadrático, etc.). Después que se hace la última predicción, un error promedio para cada máquina es calculado para indicar la aptitud de ella. Las máquinas que presenten el mejor resultado (mínimo error) se retienen para ser padres en la próxima generación.

En cuanto al esquema de generación de descendientes (reproducción), por cada máquina padre se obtiene un descendiente aplicando sobre esta una mutación aleatoria. Clásicamente, existen cinco tipos de mutaciones aleatorias:

- Cambiar un símbolo de salida.
- Cambiar un estado de transición (par símbolo de entrada y de salida para ir de un estado a otro).
- Agregar un estado.
- Eliminar un estado (si la máquina tiene mas de un estado).

- Cambiar el estado inicial (si la máquina tiene mas de un estado).

Tanto el tipo de mutación a aplicar como el número de padres a usar para generar nuevos descendientes se escogen bajo un esquema probabilístico. Los esquemas de mutación presuponen que existe un conjunto de símbolos de entrada/salida, así como de estados, para realizar sus operaciones.

La programación evolutiva se usó en problemas de reconocimiento de símbolos. Luego de un cierto período de estancamiento, ha retomando auge como método de solución en problemas que puedan ser naturalmente representados como máquinas de estados finitos.

5.8 Programación Genética

La Programación Genética (**PG**) surge como una propuesta de **J.R. Koza** que consiste en utilizar la metodología de CE no para encontrar soluciones a problemas, sino para encontrar el mejor procedimiento para resolverlos [30, 32, 33, 35, 49]. Es decir, las estructuras que se someten a evolución no codifican las posibles soluciones a un problema dado, sino los algoritmos o programas que, al ser ejecutados, determinan dichas soluciones.

Con la PG, lo que se busca es la respuesta a uno de los grandes dilemas de la ciencia de la computación: *Como pueden los computadores aprender a resolver problemas que no hayan sido explícitamente programados*, o lo que es lo mismo, *como pueden construirse computadores que hagan lo que necesitan hacer sin decirles exactamente como hacerlo.*

En la PG, la representación de los individuos se hace a través de unas estructuras de datos jerárquicamente enlazados que reciben el nombre de *árboles de análisis*. La codificación utiliza un conjunto de símbolos de funciones y otro de símbolos terminales (átomos) adecuados para la solución del problema dado, es decir, en su dominio de interés. Así, el espacio de búsqueda es el espacio de todos los posibles programas de computación compuestos de funciones y terminales apropiados al dominio del problema. Las funciones pueden ser operaciones aritméticas, funciones matemáticas, funciones lógicas, o funciones específicas del dominio.

En la práctica, tanto las funciones como los símbolos pueden implementarse de manera sencilla usando las *'expresiones-S'* de LISP. Por ejemplo, la siguiente expresión:

$$(+ 1\ 2\ (\ IF\ (\ >\ TIME\ 10\ )\ 3\ 4\ )\ )$$

contiene los átomos: 1, 2, 10, 3, 4, TIME; y las funciones: +, IF, >. El respectivo árbol se muestra en la figura 5.4.

5.8.1 Funcionamiento de los operadores genéticos

Como en la mayoría de los paradigmas de CE, en la PG los operadores de mayor uso son el de cruce y el de mutación. Usaremos un ejemplo en cada caso para ilustrar el

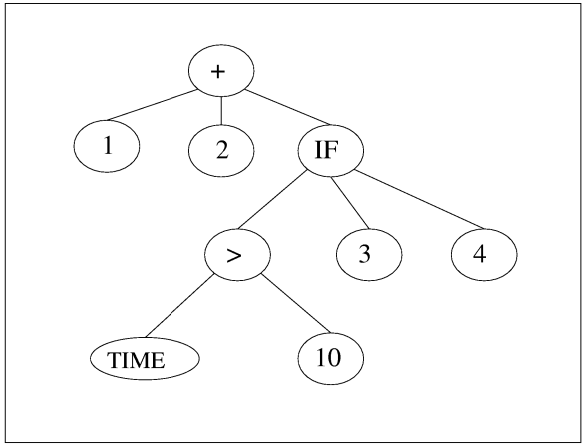

Figura 5.4: Ejemplo de un árbol de análisis

funcionamiento de estos operadores.

Cruce en PG

Supongamos que se han seleccionado los árboles padres de la figura 5.5 (Etiquetados como a y b). Si se toma la raíz del árbol como punto de cruce, el hijo 1 contendrá las ramas izquierdas de los padres; mientras que el hijo 2 las derechas. Así, se obtendrán los hijos mostrados en la figura 5.6.

Mutación en PG

El operador de mutación tiene una implementación más intuitiva, puesto que permite cambiar símbolos terminales por funciones (crear nuevas ramas), o modificar funciones (cambiar ramas ya existente, ya sea por otras ramas o por símbolos terminales). Por ejemplo, si al árbol de la figura 5.4 se le cambia el símbolo terminal 1 por una nueva rama (por ejemplo, por la rama + 1 4), se obtendría un árbol como el de la figura 5.7.

5.8.2 Resolviendo problemas con PG

Para la aplicación de la PG a un problema dado, existen cinco pasos principales a cubrir, cada uno los cuales involucra la determinación de:

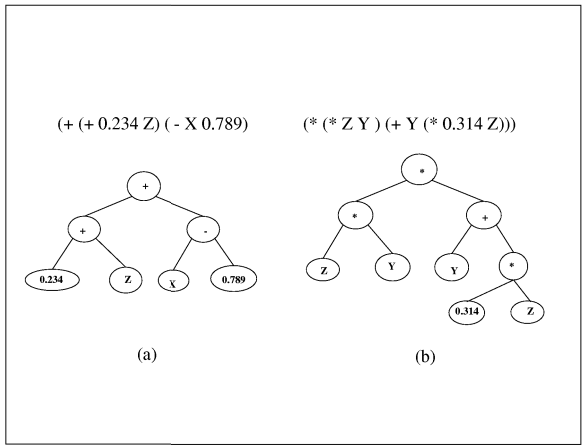

Figura 5.5: Arboles padres para el operador de cruce

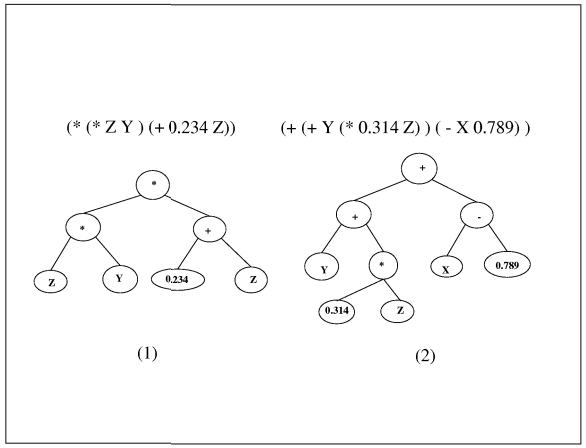

Figura 5.6: Arboles hijos obtenidos por el cruce

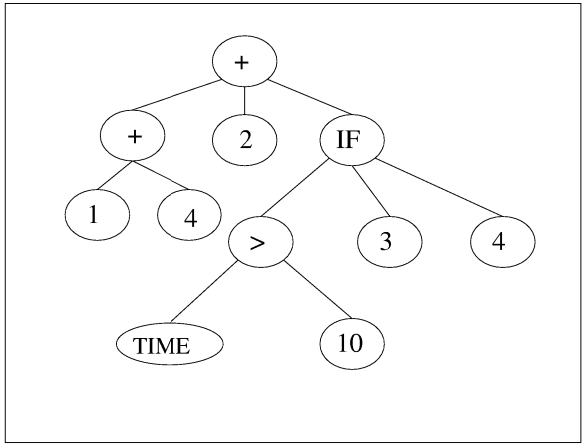

Figura 5.7: Arbol obtenido por mutación de un árbol padre

- El conjunto de los símbolos terminales (o simplemente terminales): Representan las entradas a los programas y el resto de los elementos atómicos que lo compondrán (constantes, variables, etc.). Son usados por la PG para intentar construir el programa de computación que resuelva, o aproximadamente resuelva, el problema dado.

- El conjunto de las funciones primitivas: Estas funciones se usan para generar las expresiones matemáticas que intentarán ser ajustadas, según los requerimientos del problema. Dichas funciones usan como parámetros los elementos que conforman el conjunto de símbolos terminales.

- La medida de rendimiento (función de adaptabilidad): En muchos problemas, la adaptabilidad se mide en función del error del programa. Así, mientras mas cercano sea el error a cero mejor será el programa. En otros problemas, por ejemplo en control óptimo, la adaptabilidad se mide en función de la cantidad de tiempo (o energía, o dinero) que se toma para llevar el sistema al estado deseado. De manera más general, podemos usar una función multiobjetivo que incorpore una combinación de diversos factores (exactitud, eficiencia, etc).

- Los parámetros de control de la ejecución (número de generaciones, criterio de parada, etc.).

- El mecanismo para interpretar los resultados.

Los primeros dos pasos corresponden a los pasos de especificación de la representación del problema en un AG clásico. Cada programa de computador (ej. expresiones matemáticas, etc.) es representado por la composición de funciones, pertenecientes al conjunto de funciones, y terminales, del conjunto de terminales. Dicha representación es hecha a través de árboles de análisis. Los últimos tres pasos en un PG están en concordancia directa con los pasos de un AG convencional.

5.8.3 Definición Automática de Funciones

La definición automática de funciones (en inglés ADFs) consiste en una extensión de la PG mediante la cual se descubren nuevas unidades funcionales de manera automática, esto permite resolver problemas más complejos [30, 32, 33, 35, 49]. El fundamento de definición automática de funciones es la programación modular, en la cual los programadores escriben funciones o subrutinas que son invocadas una o varias veces durante la ejecución de un programa.

Desde el punto de vista del diseño la programación modular, involucra tres pasos:

1. Descomposición del problema en subproblemas.
2. Resolución de los subproblemas
3. Recomposición de las soluciones parciales para construir la solución global.

Para implementar la definición automática de funciones en la PG, se establece una estructura sintáctica de los programas basada en la figura 5.8 [32, 30].

En la figura 5.8 se muestran ocho (8) diferentes tipos de nodos en el programa genérico (árbol de análisis). Los seis nodos que aparecen sobre la línea punteada no varían. Los otros dos nodos son los árboles de análisis que definen, tanto al cuerpo del programa como al de la función. En general, los ocho nodos mencionados son:

1. La raíz del árbol: etiqueta que indica que es el nodo inicial del árbol.
2. Inicio de la rama de definición de la función: etiqueta que indica el inicio del árbol de definición de la función.
3. Nombre del ADF: nodo con el nombre de la función.
4. Lista de argumentos: nombre de los argumentos de entrada que usara la función a definir.
5. Los valores a retornar por la ADF: nombre de los argumentos de salida de la función a definir.
6. Variables que almacenan el resultado de la llamada al ADF
7. Cuerpo del ADF: árbol de análisis que define la función.

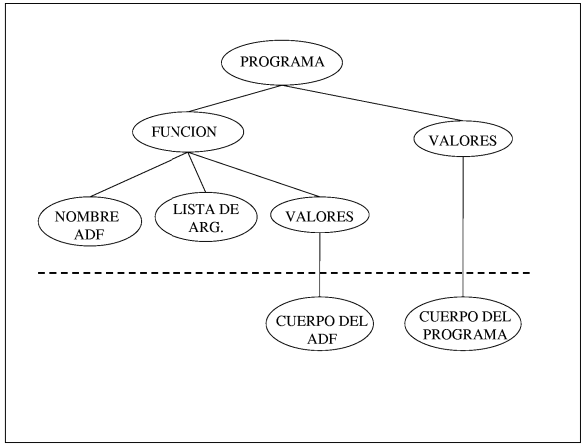

Figura 5.8: Programa genérico con una rama para la función definida automáticamente y otra rama para el cuerpo del programa (LISP)

8. Cuerpo del programa: árbol de análisis que define el programa.

Antes de usar ADFs, es necesario definir:

- Número de ADFs que se requieren.
- Número de argumentos de cada uno de los ADF.
- Referencias jerárquicas, si es el caso, entre las ADFs.
- El conjunto de terminales y funciones para cada ADF y el programa principal.

Así como se ha elaborado una estructura específica para la definición automática de funciones, se ha hecho lo mismo para otros tipos de estructuras posibles de definir automáticamente, tales como las de repetición (ADL), etc., (ver [30, 32, 33, 35, 49] para más detalles).

Bibliografía

[1] J. Aguilar "A Hybrid Approach based on Genetic Algorithms and Simulated Annealing for Combinatorial Optimization Problems", In *Proc. XXI Latinoamerican Informatics Conference*, 1994, pp. 1401-1412.

[2] T. Bäck, F. Hoffmeister, H. Schwefel, "A Survey of Evolution Strategies", In *1991 Int. Conf. on Genetic Algorithms*, 1991, pp. 2-9.

[3] T. Bäck, H. Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization", *Evolutionary Computation*, vol. 1, no. 1, pp. 1-23. 1993.

[4] T. Bäck,G. Rudolph, H. Schwefel, "Evolutionary Programming and Evolution Strategies: Similarities and Differences", In *Int. Conf. on Evolutionary Programming*, 1993, pp. 11-22.

[5] T. Bäck, U. Hammel, H. Schwefel, "Evolutionary computation: Comments on the history and current state", *IEEE Trans. Evolutionary Computation*, vol. 1, no. 1, pp. 3-17, 1997.

[6] K. Bogart, *Introductory Combinatorics*. Harcourt Brace, San Diego, 1990.

[7] G. Box, "Evolutionary operation: a method of increasing industrial productivity", *Applied Statistics*, vol. 6, pp. 81-101, 1957.

[8] A. Cairns-Smith, *Seven Clues to the Origin of Life*, *Cambridge Univ. Press*, 1985.

[9] C. Darwin, *The Origin of Species (original de su trabajo de 1859). Penguin Classics*, 1985.

[10] C. Darwin, *Teoría de la Evolución. Península*, 1972.

[11] L. Davis, *Handbook of Genetic Algorithms. Van Nostrand Reinhold*, 1991.

[12] R. Dawkins, *The Extended Phenotype: The Gene as a Unit of Selection*, *Oxford University Press*, 1982.

[13] R. Fisher, *The Genetic Theory of Natural Selection*, *Dover*, 1958.

[14] L. Fogel, A. Owens and M. Walsh, *Artificial Intelligence through Simulated Evolution. Joth Wiley*, 1966.

[15] D. Fogel, "An introduction to simulated evolutionary optimization", *IEEE Trans. Neural Networks*, vol. 5, no. 1, pp. 3-14, 1994.

[16] D. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press*, 1995.

[17] D. Fogel, *Evolutionary Computation: The Fossil Record*, *IEEE Press*, 1998.

[18] E. Ford, *Genetics and adaptation. Edward Arnold*, 1976.

[19] L.R. Foulds, *Combinatorial Optimization for Ungraduates. Springer-Verlag*, New York, 19984.

[20] A. Fraser, "Simulation of genetic systems by automatic digital computers", *Australian Journal of Biological Sciences*, vol. 10, pp. 484-491, 1957.

[21] G. Friedman, "Digital simulation of an evolutionary process", *General Systems Yearbook*, vol. 4, pp. 171-184, 1959.

[22] D. Futuyma,*Evolutionary Biology*, *Sinauer Assoc.*, 1986.

[23] D. Goldberg, "The Genetic Algorithm: Who, How, and What Next?". In *Adaptive and Learning Systems (S. Kumpati ed.)*, Plenum, 1986.

[24] D. Golberg, *Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley Co.*, 1989.

[25] D. Golberg, "Genetic and Evolutionary Algorithms Come of Age", *Communications of the ACM*, vol. 37, no. 3, pp. 113–119, 1996.

[26] J. Heitkotter, D. Beasley, *The Hitch-Hiker's Guide to Evolutionaty Computation*, *Heitkötter and Beasly Ed.*, 1998.

[27] F. Hidrobo, J. Aguilar, "Algoritmos Genéticos Paralelos en Problemas de Optimización Combinatoria", *Revista Técnica de Ingeniería, LUZ*, vol. 21, no. 1, pp.47-58, 1998

[28] J. Holland, *Adaptation in Natural and Artificial Sytems, 2nd ed. MIT Press*, 1992.

[29] J. Holland, "Genetic Algorithms", *Scientific American*, vol. 267, no. 1, pp. 66-72, 1992.

[30] K. Kinnear (Ed), *Advances in Genetic Programming. MIT Press*, 1994.

[31] S. Kirkpatric, "Optimization by simulated annealing: quantitative studies", *Journal Statis. Phys.*, vol. 34, pp. 974-997, 1984.

[32] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, *MIT Press*, 1992.

[33] J. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, *MIT Press*, 1994.

[34] F. Kursawe, F. "Evolution strategies: Simple models of natural processes?", *Revue Internationale de Systémique*, 1994.

[35] W. Langdon, *Genetic Programming and Data Structures. Kluwer*, 1998.

[36] R. Lewontin, *The Genetic Basis of Evolutionary Change*, *Columbia Univ. Press*, 1974.

[37] S. Maynard, "Optimization Theory in Evolution", *Annual Review of Ecology and Systematics*, vol. 9, pp. 31-56, 1978.

[38] S. Maynard, Maynard Smith, *Evolutionary Genetics*, *Oxford Univ. Press*, 1989.

[39] E. Mayr, *Animal Species and Evolution*, *Harvard Univ. Press*, 1963.

[40] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag*, 1994.

[41] M. Mitchell, S. Forrest, "Genetic Algorithms and Artificial Life", Artificial Life, vol. 1 no. 1, 1993.

[42] M. Mitchell, *An Introduction to Genetic Algorithms. MIT Press*, 1996.

[43] A. Pérez, *Introducción a la Computación Evolutiva. http://www.etsimo.uniovi.es/pub/EC/EA/papers/intro-spanish.ps.gz*

[44] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (Evolution Strategy: Optimization of technical systems by means of biological evolution), 2nd ed.*, {*it Fromman-Holzboog*, 1993.

[45] M. Ridley, *The Problems of Evolution*, *Oxford Univ. Press*, 1985.

[46] R. Rosen, *Optimality Principles in Biology*, *Butterworths*, 1967.

[47] H. Schwefel, *Numerical Optimization of Computer Models*, *Wiley*. 1981.

[48] H. Schwefel, "Collective Phaenomena in Evolutionary Systems", In it Proc. 31st Ann. Meet. Int. Soc. for General System Research, 1987, pp. 1025-1033.

[49] L. Spector, W. Langdon, U. O'Reilly, P. Angeline, *Advances in Genetic Programming, Vol. 3*, *IEEE Press*, 1999.

[50] C. Stern, *El origen de la genética. Alhambra*, 1973.

[51] P. Tort (ed.), *Dictionary of Darwinism and of Evolution*, *Presses Universitaires de France*, 1996.

[52] J. Watson, N. Hopkins, J. Roberts, J. Steitz, A. Weiner, *Molecular Biology of the Gene (4th ed.)*, *Benjamin*, 1987.

[53] P. Wayner, "Genetic Algorithms: Programming takes a valuable tip from nature", BYTE, January, pp. 361–368, 1991.

[54] G. Williams, *Adaptation and Natural Selection*, *Princeton Univ. Press*, 1966.

Nuevas Técnicas Inteligentes

José L. Aguilar Castro
CEMISID
Departamento de Computación
Facultad de Ingeniería
Universidad de Los Andes
Mérida 5101, Venezuela
E-mail: aguilar@ula.ve

Francisco Hidrobo Torres
Laboratorio SUMA
Facultad de Ciencias
ULAM
E-mail: hidrobo@ula.ve

Capítulo 6

Nuevas Técnicas Inteligentes

Este capítulo presenta tres aspectos que estos últimos años se han desarrollado entorno al área de Computación Inteligente. Antes que nada, haremos una presentación del área Vida Artificial. Después, presentaremos unos de los esquemas inteligentes que ha tenido uno de los desarrollos mas impresionantes estos últimos años, los Sistemas Artificiales de Hormigas. Es bueno resaltar que actualmente están en plena evolución otras técnicas Artificiales Inteligentes, las cuales no serán presentadas en este libro, tales como los Sistemas Artificiales Inmunológicos, la computación basada en Colonias de Abejas, la Computación ADN, los Algoritmos Culturales, etc. Finalmente, la última parte de este capítulo se dedicará a presentar los Sistemas Híbridos Inteligentes, vistos como mecanismos integradores de todas las técnicas estudiadas en este libro.

6.1 Vida Artificial

La Vida Artificial como área de investigación no existe sino desde hace unos quince años. El objetivo de la Vida Artificial es ambicioso, recrear artificialmente la vida [3, 2, 8, 14, 15]. Así, la vida artificial la definiremos como el área que diseña y desarrolla sistemas artificiales que poseen características de los seres vivientes. Es así como aparecen términos tales como ambientes artificiales, ecosistemas artificiales, creadores artificiales y biología artificial.

La Vida Artificial, mas que formular teorías a partir de la observación de sistemas naturales, trata de comprender la vida elaborando programas capaces de genera estructuras emergentes de apariencia biológica. La Vida Artificial parte de abstracciones generadas artificialmente, las cuales no deben ser irrealistas para que puedan tener un paralelo con la manera de actuar de los sistemas naturales. Una vez esto, las manifestaciones de los comportamientos artificialmente elaborados van a proveer nuevos hechos, e inclusos fenómenos biológicos, posibilitando evaluar teorías hasta ahora especulativas sobre muchos de esos fenómenos biológicos. De esta manera, la Vida Artificial permite alargar el concepto mismo de experimentabilidad en la biología.

La Vida Artificial esta creando un mundo extraño de criaturas donde sus comportamientos y aptitudes son imprevisibles y emergentes, incluso para sus creadores. Pero, lo mas impresionante son sus capacidades para interpretar esos escenarios emergentes en términos funcionales en el interior mismo del universo artificial donde aparecen. La

Vida Artificial es capaz de generar por primera vez en la historia de la ciencias, un universo de criaturas artificiales que se autoreproducen, evolucionan, aprenden, e incluso se organizan colectivamente. La validez de esos organismos y biosistemas virtuales es medido por el hecho que a partir de especificaciones simples y locales que definen las reglas de juego, se hacen emerger estructuras y comportamientos comparables a los que existen en el mundo real biológico.

La Vida Artificial sigue las trazas de un área más antigua, la Inteligencia Artificial. La Vida Artificial presenta muchos de los problemas que se encuentran en Inteligencia Artificial, pero tiene características propias que le han permitido desarrollarse como un área distinta. Por ejemplo, los sistemas de Vida Artificial son mas complicados que los de Inteligencia Artificial, ya que la inteligencia (facultad cognitiva) representa solamente un aspecto de la vida. Es decir, la vida artificial es más natural ya que intenta reproducir todo el sistema artificialmente, y no solo una parte. En particular, la inteligencia artificial tiene el problema de que numerosas funciones no están directamente ligadas a la inteligencia (son hechas por otros subsistemas de los organismos biológicos), por consiguiente, deben ser generadas desde el exterior (limitando su autonomía). Por el contrario, en la vida artificial todas las funciones deben ser incluidas en el sistema artificial, lo que minimiza la dependencia de operadores externos.

Pero al igual que en Inteligencia Artificial, donde la primera pregunta tiene que ver con la definición de la inteligencia, en Vida Artificial tiene que ver con la definición de la vida. Esta área ha conllevado a debates interesantes sobre el concepto de la vida. La vida es un fenómeno del cual creemos tener conocimiento intuitivo, pero que por su complejidad su definición es uno de los problemas difíciles a resolver. El concepto de vida que se utiliza es mas amplio que el que nosotros conocemos: se trata de la vida como es, como podría ser en otros planetas, con otros materiales, así como en el universo abstracto de los computadores.

De una manera empírica, la biología a desarrollado la noción de vida. A partir de ciertas apreciaciones se han podido generalizar principios que han permitido reconocer a seres vivientes de otras variedades que se han podido observar. El problema es saber si tenemos criterios para distinguir entre lo que nosotros conocemos por vida y tal como ella podría ser. Según Maynard-Smith (1986) [14], la vida debe definirse por la posesión de propiedades que son necesarias para asegurar la evolución por selección natural. En este caso, la vida aparece como una red recursiva de unidades autónomas, lo que asegura el desarrollo de sistemas adaptativos. Pero para adaptarse a cambios rápidos e irregulares del ambiente, se debe agregar una condición adicional: cada organismo logra individualmente establecer una coordinación funcional entre los estados del ambiente y los estados metabólicos-motores que el dispone.

A continuación presentamos un conjunto de propiedades de la vida aceptadas por la comunidad científica:

- La vida implica una estructura espacio-temporal.
- La vida requiere de procesos de auto-reproducción.

- Los seres vivientes guardan información de su propia descripción (esta contenida en el ADN).
- Todo ser viviente posee un metabolismo, que convierte energía y materia del ambiente en elementos y energías utilizable por las diferentes partes de su organismo.
- Todo ser viviente interactua con su ambiente.
- Hay una interdependencia entre todas las partes de un ser viviente.
- La vida evoluciona.

Esta lista podría alargarse, pero todas estarían en relación con dos características fundamentales en todo ser viviente: su autonomía (crear sus propios controles, leyes y procesos de autorregulación, para orientar su comportamiento) y su facultad para procesar información.

Uno de los aspectos que habrá que tener presenten en la vida artificial es que los seres vivientes son organismos, mientras que las abstracciones a nivel de modelos o simulaciones que lo hacen posible son mecanismos. En ese sentido, la vida artificial hace aportes importantes a la biología teórica, ya que permite estudiar la vida tal como la conocemos, pero también tal como ella podría ser en cualquier universo (no solamente en la tierra). Las criaturas virtuales que se emulan en los computadores aseguran las características de los seres vivientes tal como se conocen en la tierra (tienen un metabolismo, se reproducen, mueren, se organizan colectivamente, aprenden y evolucionan).

Dentro del área Vida Artificial existen tres líneas de trabajos:

- Los que consideran a la Vida Artificial como una extensión del universo formal del dominio de las ciencias biológicas.
- Los que consideran a la Vida Artificial como una metodología (a través del modelado y de la simulación) válida para estudiar organismos naturales.
- Los que consideran a la Vida Artificial como un medio para obtener organismos artificiales en el mismo universo que el de los organismos naturales.

Lo mas interesante del área vida artificial, es que los sistemas que se desarrollan son vistos como sistemas autoadaptativos, lo que los diferencia de los esquemas clásicos de computación (ver figura 6.1).

La vida artificial intentó en sus inicios extraer la vida de los sistemas vivientes, del ambiente concreto donde ellos se realizan, sin poder lograrlo por tres razones [1, 3, 8, 14, 15]:

- La primera viene del carácter autoreferencial del código genético: en particular, su naturaleza auto-informacional, la cual esta relacionada con sus mecanismos moleculares.

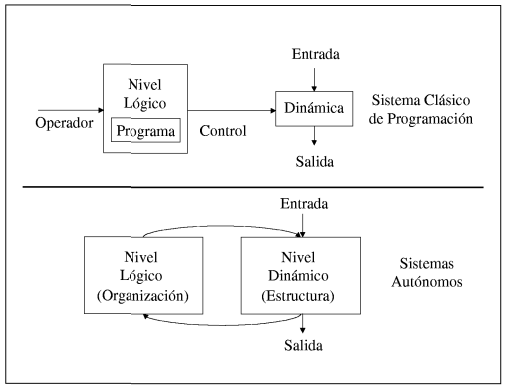

Figura 6.1: Comparación entre Sistemas Programados

- La segunda es la interrelación entre la lógica y la dinámica en el proceso celular. En otras palabras, la lógica de los procesos no se separa de sus bases físicas. Así, la lógica de los sistemas vivientes conocidos, esta profundamente ligada a las especifidades de sus materiales, por más que gran parte de la información este precodificada.
- La última es la composición de la información en los sistemas biológicos. Si bien es cierto que la organización biológica esta basada en la información genética, su expresión final (fenotipo del organismo) es el resultado de un conjunto complejo de procesos dinámicos auto-organizativos dependientes del contexto. Esto demuestra que la organización biológica real esta basada en información genética y en sus estructuras internas como respuesta al entorno donde existe. Así, la organización viviente esta basada en las interacciones que se dan entre el conjunto de instrucciones simbólicas del ADN y el universo físico.

Pero para contornar estos problemas, Langton, en su articulo "Programme de la Vie Artificielle"[14, 15], propone la independencia entre la lógica de la vida y su realización concreta. De aquí, la idea de reproducir la organización de la base de la vida en otros medios, en particular, informáticos. Esto conlleva a orientar las investigaciones hacia la construcción de sistemas artificiales que busquen desarrollar realizaciones en concreto que reproduzcan las capacidades fundamentales de los seres vivientes (autoreproducción, autoreparación, etc.), y no sistemas que simulen esas capacidades.

Ciertamente que continua la duda sobre la idea de la vida tal como ella podría ser, en oposición a la vida tal como nosotros la conocemos. La teoría realmente capaz de explicar que es la vida (tanto la que conocemos, así como de otras formas), debe

conducir a la posibilidad de articular programas de investigación que conduzcan a su fabricación. El proyecto de ciencia basado en la noción de vida virtual imponiendo su hegemonía a la que estudia la vida natural no nos parece razonable. Así, la Vida Artificial no sustituirá la biología clásica, sino que colabora con ella, contribuirá a la creación de una nueva ciencia biológica que estudiará organismos tanto naturales como artificiales.

6.2 Sistemas Artificiales de Hormigas

Este paradigma computacional se basa en una analogía con el funcionamiento de las colonias de hormigas, de tal manera de proponer un enfoque estocástico de resolución de problemas de optimización combinatoria [3, 2, 4, 5, 6, 7, 9, 11, 16, 17]. Las principales características de este modelo son *la retroalimentación positiva* y *el procesamiento distribuido*, a través de un esquema *heurístico constructivo*. La *retroalimentación positiva* permite un rápido descubrimiento de buenas soluciones, la *computación distribuida* evita una convergencia prematura, y la *heurística constructiva* ayuda al procedimiento de búsqueda de soluciones aceptables. Este enfoque presenta las siguientes propiedades [3, 2, 4, 5, 11]:

- Versatilidad:
  Puede ser aplicado a versiones similares del mismo problema. Por ejemplo, a diferentes versiones del problema del Viajero de Comercio.

- Robustez:
  Puede ser aplicado, con pequeños cambios, a otros problemas de optimización combinatoria (Por ejemplo, la versión que se utiliza para resolver el problema del Viajero de Comercio se puede utilizar para resolver el problema de Asignación Cuadrática).

- Basado en un esquema poblacional:
  Esto hace fácil su implementación paralela, posibilitando una explotación eficiente de la retroalimentación positiva como mecanismo de búsqueda.

En este enfoque se distribuye la actividad de búsqueda en unos elementos llamados "hormigas", es decir, agentes con simples capacidades, los cuales emulan en cierta forma el comportamiento de las hormigas reales. Así, las investigaciones sobre el comportamiento de las hormigas reales han inspirado esta área [2, 4, 11]. En particular, el problema estudiado es entender como animales ciegos como las hormigas pueden establecer el camino mas corto entre su colonia y una fuente de alimentos. En esos estudios se determinó que el medio usado por las hormigas para comunicarse información con respecto al camino a seguir, consistía en trazas de *pheromone* (sustancia química que emiten las hormigas). En general, las hormigas van dejando cantidades de pheromone (en diferente proporción) en sus movimientos, marcando los caminos

recorridos con esa sustancia. Las observaciones mostraron que mientras una hormiga aislada esencialmente hacía un movimiento aleatorio, una hormiga que encontraba trazas previas podía detectarlas y decidir con alta probabilidad seguirlas, de tal manera de reforzar dicha traza con su propio pheromone. El comportamiento colectivo que emerge es una forma de comportamiento *autocatalítico* (proceso que se refuerza a si mismo, de tal manera de generar una rápida convergencia), donde entre más hormigas decidan seguir una traza es más atractiva seguir a esa traza. Este último proceso es una *retroalimentación positiva*, tal que la probabilidad que una hormiga escoja un camino aumenta según el número de hormigas que previamente han escogido el mismo camino.

Como se indico en el párrafo anterior, la metáfora natural en la que se basan los algoritmos artificiales de hormigas es la de las colonias de hormigas, que puede resumirse de la siguiente manera []: las hormigas reales son capaces de encontrar el camino más corto desde una fuente de alimentos a su nido, sin usar señales visuales, sino explotando la información del pheromone. Para eso, mientras las hormigas caminan ellas van depositando pheromone, el cual se acumula al pheromone previamente dejado por otras hormigas. Suponga ahora que una hormiga llega a un punto donde debe escoger si cruzar a la derecha o a la izquierda. Si ellas no tienen ninguna razón para definir cual es la mejor escogencia, ellas escogerán aleatoriamente, lo cual nos hará suponer que la mitad cruzara a la derecha y la otra mitad a la izquierda. La ruta escogida será fortalecida en la misma proporción del pheromene dejado por las hormigas que han pasado por ese camino. Al haber un camino mas corto que otro, eso redundará que uno será mas visitado en promedio y acumulará más pheromene en su camino. Después de un tiempo transitorio, la cantidad diferente de pheromone entre los dos caminos será lo suficientemente grande para tener una influencia en las nuevas hormigas, tal que ellas prefieran con una alta probabilidad el camino más corto (ya que perciben una mayor cantidad de pheromone). Después de cierto tiempo, todas las hormigas seguirán el camino más corto.

Esto ha inspirado a los sistemas artificiales de hormigas, algoritmo en el cual un grupo de hormigas cooperan entre si para solucionar un problema intercambiando información a través del pheromone depositado en los arcos de un grafo. Así, cada hormiga construye una solución (es decir, existe una búsqueda paralela de soluciones) mezclando la explotación de la información ganada por experiencias pasadas con una heurística egoísta (en este tipo de heurística solo se trata de seguir a la mejor solución). La *experiencia pasada* es basada en el pheromone depositado por las hormigas, mientras que la *heurística egoísta* es simplemente basada en la longitud de los arcos del grafo. La principal idea innovadora introducida por este enfoque es el uso de la cooperación entre agentes relativamente simples, los cuales se comunican a través de una memoria distribuida (pheromones depositados en los arcos).

Para explicar matemáticamente el fenómeno estudiado, considere el ejemplo de la figura 6.2 que muestra un camino a través del cual las hormigas están caminando (desde una fuente de alimentos A al nido E y viceversa (ver figura 6.2.a)). En algún momento, un obstáculo aparece de tal manera que el camino previo es obstaculizado.

Ahora, las hormigas que iban de A a E caminaran desde A hasta B (o hasta D las que iban en sentido opuesto) y al llegar a ese punto deben decidir si tomar hacia la derecha o izquierda (ver figura 6.2.b). La escogencia es influenciada por la intensidad de la traza de pheromone dejada por precedentes hormigas. Un alto nivel de pheromone de un camino es para una hormiga un fuerte estimulo, y así, una alta probabilidad para seguir ese camino. Las primeras hormigas que llegan a B (o D) tienen la misma probabilidad de cruzar a la izquierda o a la derecha (ya que no hay nada de pheromone). Como el camino BCD es mas corto que BHD, la hormiga que siga el camino BCD llegará a D antes que la hormiga que siga el otro camino (ver figura 6.2.c). El resultado es que las hormigas viniendo desde ED encontraran una traza más fuerte en el camino DCB, causado por la mitad de todas las hormigas que han decidido irse por el camino DCBA y por las que se vinieron por el camino ABCD. Entonces, ellas preferirán (por probabilidad) el camino DCB al camino DHB (esto también ocurrirá en sentido inverso). Es decir, el número de hormigas que seguirán el camino BCD (o DCB) por unidad de tiempo será más alto que el número de hormigas que seguirán el camino BHD (o DHB). Esto causará que la cantidad de pheromone sobre el camino más corto crecerá más rápido que sobre el más largo, de tal manera que la probabilidad con la cual una simple hormiga escoja el camino a seguir tenderá a ser hacia el más corto. El resultado final es que rápidamente todas las hormigas tenderán a escoger el camino más corto.

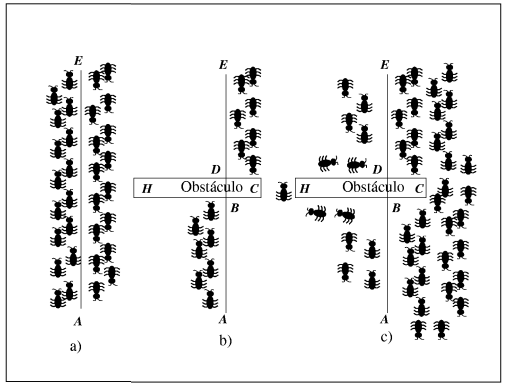

Figura 6.2: Sistema de Hormigas Modelado

Veamos la misma explicación anterior, pero ahora con números: consideremos el grafo de la figura 6.3.a. En la figura esta indicada las distancia (d) entre los nodos (es un grafo simétrico). Supongamos que a intervalos regulares de tiempo t, 30 nuevas hormigas van desde E hacia A y 30 desde A hacia E. Además, cada hormiga camina a una velocidad de una unidad de distancia por unida de tiempo dejando

una traza de pheromone de intensidad 1 mientras camina, y el pheromone dejado por una hormiga se evapora después de una unidad de tiempo. En t=1 no hay trazas, pero las 30 hormigas que llegan a B y D deben escoger hacia donde ir, lo cual harán aleatoriamente. En promedio, 15 hormigas se irán de cada nodo inicial B o D hacia H o C (ver figura 6.3.b). Para t=2, las nuevas 30 hormigas que van de E a A encontraran una traza de intensidad 15 en el camino que pasa por H, dejadas por las 15 hormigas previas que van de E, y una traza de intensidad 30 sobre el camino a C, obtenida de la suma de las trazas de las 15 hormigas previas que vienen de A y las 15 previas que van de E a A a través de C (ver figura 6.3.c). Así, la probabilidad de escoger un camino es modificada, tal que el número de hormigas que irán hacia C será el doble de aquellas que van hacia H (20 versus 10, respectivamente). Lo mismo ocurrirá para las 30 hormigas que vienen de A a E. Para t=3, la traza por el camino BCD será de 40, mientras que por BHD será de 10. Esto hace que en promedio 24 de las nuevas hormigas que van de B a D irán hacia C y 6 hacia H, lo mismo ocurre en la dirección de D a B. Para t=4, las trazas serán 48 contra 6, y las nuevas hormigas, en promedio, se irán 27 contra 3 por cada camino en cada sentido. Esto sigue así, hasta que al final, eventualmente todas se irán por el camino más corto.

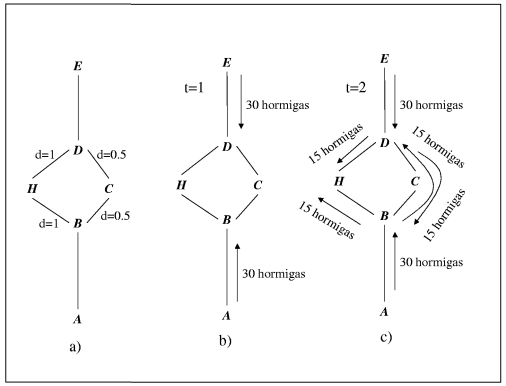

Figura 6.3: Sistema de Hormigas representado usando Grafos

6.2.1 Modelo

En esta sección, se presentará el Sistema Artificial de Hormigas. Para describirlo, usaremos el problema del viajero de comercio como herramienta (ya que el sistema artificial de hormigas puede hacer una representación directa de ese problema).

El problema del viajero de comercio consiste en, dada n ciudades, encontrar el camino de recorrido más corto por todas las ciudades, tal que se pase por cada una

de ellas una sola vez. Nosotros llamaremos d_{ij} la distancia entre las ciudades i y j. Una instancia de ese problema puede ser representada por un grafo (N,E), donde N es el conjunto de ciudades y E el conjunto de caminos entre las ciudades.

Si $b_i(t)(\forall i = 1, ..., n)$ es el número de hormigas en la ciudad i en el momento t, $m = \sum_{i=1}^{n} b_i(t)$ será el número de hormigas en el sistema. Cada hormiga tendrá las siguientes características:

- Ella escogerá la ciudad a visitar según una probabilidad que esta en función de la distancia entre las ciudades y la cantidad de pheromone presente en los arcos.
- Para que las hormigas hagan recorridos legales, transiciones a ciudades ya visitadas son prohibidas.
- Cuando se completa un recorrido, la hormiga deja una sustancia llamada pheromone (traza) en cada uno de los arcos visitados.

En los Sistemas Artificiales de Hormigas se usa un grafo como modelo. Dicho grafo contiene dos costos, el costo del arco (ligado a la distancia entre sitios) y el costo de que tan deseable es un arco (ligado al pheromone, el cual es actualizado por cada hormiga que pasa por él). Usando ese grafo, se actúa de la siguiente forma:

Cada hormiga genera un recorrido completo usando una probabilidad llamada *regla de transición de estado*, de tal manera que las hormigas se muevan por los arcos más cortos con alta cantidad de pheromone. Una vez que todas las hormigas han realizado un recorrido completo, se aplica una regla de actualización global de pheromone. Dicha actualización consiste en que una porción del pheromone actual en cada arco se evapora, y cada hormiga deposita una cantidad de pheromone por cada arco que pertenezca a su recorrido, proporcional a que tan corto fue su recorrido (en pocas palabras, arcos que pertenezcan al recorrido más corto recibirán una mayor cantidad de pheromone). Este proceso es iterativo, tal que una iteración (t,t+1) del algoritmo de Colonia de Hormigas consiste en los n movimientos realizados por las m hormigas. Así, cuando se ha ejecutado n veces el algoritmo las m hormigas han completado un recorrido (paseo por todas las ciudades).

Ahora pasamos a definir las expresiones matemáticas necesarias al modelo. Como se dijo antes, después de cada iteración (recorrido completo) la intensidad de la traza es actualizada. Esta se actualiza según la fórmula siguiente:

$$\gamma_{ij}(t+n) = \alpha\gamma_{ij}(t) + \triangle\gamma_{ij} \tag{6.1}$$

donde,
$\gamma_{ij}(t)$ es la intensidad de la traza depositada en el arco (i,j) en el momento t.
α es un coeficiente tal que (1-α) representa la tasa de evaporación de la traza entre el intervalo de tiempo t y t+n.
Además,

$$\triangle\gamma_{ij} = \sum_{k=1}^{m} \triangle\gamma_{ij}^{k}$$

donde,
$\triangle\gamma_{ij}^{k}$ es la cantidad de traza (pheromone) por unidad de longitud dejada en el arco (i,j) por la k^{th} hormiga entre el intervalo t y t+n, la cual es calculada como:

$$\triangle\gamma_{ij}^{k} = \begin{cases} Q/L_k \; si \; la \; hormiga \; k \;\; pasa \; por \; el \; arco \; (i,j) \\ \qquad en \; su \; recorrido \; (entre \; t \; y \; t+n) \\ 0 \; de \; lo \; contrario \end{cases} \tag{6.2}$$

donde,
Q es una constante,
L_k es la longitud del recorrido hecho por la hormiga k.

El coeficiente α debe ser menor que 1 para evitar acumular cantidades de traza ilimitadamente. Por otro lado, la intensidad de la traza en el momento $t = 0$ ($\gamma_{ij}(0)$), puede ser escogida aleatoriamente.

Para asegurar que se cumpla la restricción de que una hormiga visita n ciudades, se usa una estructura de datos tipo lista que guarda las ciudades ya visitadas por ella hasta el momento t ($LISTA_k$(t), $\forall k = 1, ...m$), de tal manera de que la hormiga no vuelva a visitarlas hasta que el recorrido sea completado (iteración n). Así, $LISTA_k(s)$ indicará la ciudad visitada por la hormiga k en la posición (momento) s. Cuando un recorrido es completado, entonces el algoritmo pasa a calcular el valor de dicho recorrido (la solución que propone esa hormiga).

Nosotros llamamos visibilidad (v_{ij}) la cantidad $1/d_{ij}$, la cual es el inverso de la distancia. Esta cantidad no es modificada durante la ejecución del algoritmo, a diferencia de la traza, la cual cambia según la ecuación 6.1. La regla de transición de estado (o función de transición), la cual indica la probabilidad con la cual la hormiga k escogerá moverse desde i hasta j, es dada por la ecuación siguiente:

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\gamma_{ij}(t)]^{\theta}+[(v_{ij})]^{\beta}}{\sum_{l \in AUN_k(t)}[\gamma_{il}(t)]^{\theta}+[(v_{il})]^{\beta}} \; sij \in AUN_k(t) \\ 0 \; de \; lo \; contrario \end{cases} \tag{6.3}$$

donde,
$AUN_k(t)$ es el complemento del conjunto $LISTA_k(t)$. Es decir, son las ciudades aun no visitadas por la hormiga k hasta el momento t (para generar soluciones validas).
θ y β son parámetros que determinan la importación relativa del pheromone vs. la distancia.

Así, la probabilidad de transición combina la visibilidad (indica que ciudades deben ser escogidas con alta probabilidad, de tal manera de implementar una heurística egoísta constructiva), con la intensidad de la traza en el momento t (indica que si un arco (i,j) ha tenido un gran tráfico, entonces este es altamente deseable, de tal manera de implementar un proceso autocatalítico).

La actualización del pheromone tiende a asignar una gran cantidad de pheromone a los caminos más cortos. Este esquema es similar al *aprendizaje reforzado*, en el cual,

mejor es una solución más se refuerzan los elementos que caracterizan dicha solución. El pheromone depositado en los arcos tiene el rol de una memoria distribuida de largo término. Esta memoria no es guardada localmente en cada individuo, sino distribuida a través de los arcos del grafo. Esto permite una forma indirecta de comunicación. Algunas características de las diferentes versiones algorítmicas de este modelo, son las siguientes:

- Algunas veces, la regla de actualización global es aplicada solamente sobre los arcos que pertenecen al mejor recorrido (mejor hormiga). En otros versiones, todas las hormigas pueden actualizar los arcos que recorrió, proporcionalmente a la calidad de su recorrido.
- Algunas veces, mientras una hormiga va construyendo su solución, una regla de actualización local se va aplicando.

Estas versiones se verán en detalle más adelante.

6.2.2 Algoritmos

Dada las definiciones anteriores, el algoritmo clásico de los sistemas artificiales de hormigas es definido como sigue: en t=0 se realiza una fase de inicialización, durante la cual las hormigas son colocadas en las diferentes ciudades y un valor inicial $\gamma_{ij}(0)$ es colocada como la intensidad inicial de las trazas en cada arco. El primer elemento de $LISTA_k$ ($LISTA_k(1)$), para cada hormiga k, es la ciudad inicial donde ella es colocada ($\forall k = 1, ..., m$). A partir de eso, cada hormiga se mueve de una ciudad i a otra j, escogiendo la ciudad para donde moverse según la función de probabilidad definida por la ecuación 6.3 (si $\theta = 0$, el nivel de traza no es considerado y se tendrá un algoritmo egoísta estocástico con múltiples puntos de inicio). Después de n iteraciones, todas las hormigas han completado un recorrido y $LISTA_k$, ($\forall k = 1, ..., m$), estará llena. En ese momento, para cada hormiga k se calcula la longitud de su recorrido (L_k), y $\triangle\gamma_{ij}^k$ es actualizado según la ecuación 6.2. Entonces, el camino más corto propuesto por las diferentes hormigas es seleccionado y guardado ($min_k(L_k)$, $\forall k = 1, ..., m$). Finalmente, $\gamma_{ij}(t + n)$ es actualizado usando la ecuación 6.1, según la versión del algoritmo que se este usando.

Este proceso es repetido hasta un máximo número de ciclos (cada ciclo es un grupo de n iteraciones de las m hormigas), o hasta que todas las hormigas empiecen a hacer los mismos recorridos (esto se llama comportamiento estancado, ya que el algoritmo no busca nuevas soluciones). El algoritmo general es el siguiente:

1. Inicializar todos los valores ($\gamma_{ij}(0)$, $\triangle\gamma_{ij}$, número de ciclos, etc.).
2. Repetir para $k = 1, m$
   (a) Colocar cada hormiga k en un sitio inicial r escogido aleatoriamente entre 1 y n ($LISTA_k(1) = r$)

3. Repetir para s=2, n
   (a) Repetir para k=1, m
      i. Escoger nodo j donde deberá moverse la hormiga k, a sabiendas que esta actualmente en el nodo i, según probabilidad $p_{ij}^k(t)$ (ver 6.3)
      ii. Mover hormiga k al nodo j.
      iii. Insertar nodo j en $LISTA_k(s)$.
4. Repetir para k=1, m
   (a) Calcular el costo del recorrido descrito por $LISTA_k$ y guardar en L_k.
   (b) Calcular $\triangle\gamma_{ij}^k$, según ecuación 6.2.
   (c) $\triangle\gamma_{ij} = \triangle\gamma_{ij} + \triangle\gamma_{ij}^k$.
5. $L_{Optimo} = min_k(L_k)$, $\forall k = 1, m$
6. Calcular, por cada arco(i,j), $\gamma_{ij}(t + n)$ según ecuación 6.1.
7. t=t+n
8. reinicializar $\triangle\gamma_{ij} = 0$ para todos los arcos.
9. Determinar si actual recorrido más corto (L_{Optimo}) es mejor que anterior (L'_{Optimo}). Si es el caso: $L'_{Optimo} = L_{Optimo}$.
10. Si número ciclos es igual al máximo o existe comportamiento estancado
    (a) Imprimir recorrido mas corto (L'Optimo). de lo contrario
    (b) limpiar la lista.
    (c) ir a 2.
    (d) número-ciclos=número-ciclos + 1

Este algoritmo es de complejidad ($numero-ciclos_{maximo}.n.m$). Varias extensiones (versiones) a este algoritmo han sido hechas. A continuación las presentaremos.

Las primeras extensiones a este algoritmo fueron llamadas *densidad-hormigas y cantidad-hormigas*. En esos modelos, cada traza es actualizada en cada paso, sin esperar que el recorrido se complete. Ellas difieren entre si en la manera como las trazas son actualizadas. En el modelo *densidad-hormigas* se deja una cantidad de traza Q por cada unidad de longitud en cada arco (i,j), cada vez que una hormiga pase por dicho arco:

$$\triangle\gamma_{ij}^k = \begin{cases} Q \text{ si la hormiga } k \text{ pasa por el arco } (i,j) \\ \quad \text{en su recorrido (entre } t \text{ y } t+1) \\ 0 \text{ de lo contrario} \end{cases} \tag{6.4}$$

En el modelo *cantidad-hormigas*, una hormiga que va de i a j deja una cantidad Q/d_{ij} de traza por unidad de longitud entre dichos arcos:

$$\triangle\gamma_{ij}^{k} = \begin{cases} Q/d_{ij} \; si \; la \; hormiga \; k \; pasa \; por \; el \; arco \; (i,j) \\ \qquad en \; su \; recorrido (entre \; t \; y \; t+1) \\ 0 \; d\epsilon \; lo \; contrario \end{cases} \tag{6.5}$$

Como se ve, en el modelo densidad-hormigas la intensidad de traza que se deja en el arco (i,j) cuando una hormiga va de i a j es independiente de la distancia, mientras que en el otro modelo es inversamente proporcional a d_{ij} (caminos más corto son más deseados).

Ambos modelos han dado peores resultados que el algoritmo original. Esto es debido al tipo de información de retroalimentación usada para dirigir el proceso de búsqueda. En el algoritmo clásico se usa información global, es decir, cada hormiga deja una cantidad de traza que es proporcional a que tan buena solución es (hormigas que producen camino más corto contribuyen con una cantidad de traza más grande que las hormigas cuyo recorrido fue muy pobre). En las otras dos extensiones del modelo original se usa información local para dirigir el proceso de búsqueda (por ejemplo, el modelo cantidad-hormiga considera solamente la distancia del arco actualmente recorrido). En estos modelos, la búsqueda no es dirigida por ninguna medida del resultado final alcanzado, por consiguiente, no es sorprendente los pobres resultados que se logran con ellos (para más detalles, ver [2, 5]).

Otra versión algorítmica de los Sistemas Artificiales de Hormigas, es la siguiente [5, 6, 7, 17]): m hormigas son inicialmente colocadas en n ciudades aleatoriamente. Cada hormiga construye un recorrido aplicando una regla de transición de estado, la cual provee un balance entre la exploración de nuevas áreas y la explotación del conocimiento acumulado (ver ecuación 6.6). A medida que se construye la solución, se va modificando la cantidad de pheromone en cada uno de los arcos visitados usando una regla de actualización local. Una vez que todas las hormigas han terminado su recorrido, la cantidad de pheromone es nuevamente modificada en los arcos que conforman sus recorridos (aplicando una regla de actualización global sobre los arcos que pertenecen al mejor recorrido de hormiga). En esta versión, la regla de transición de estado, tal que una hormiga k posicionada en i escoja moverse a la ciudad j, es la siguiente:

$$j = \begin{cases} max_{(l \in AUN_k(t))}(\gamma_{il}(t)^{\theta} * v_{il}^{\beta}) \; si \; q \leq q_0 \; (explotacióon) \\ S \; de \; lo \; contrario \; (exploración) \end{cases} \tag{6.6}$$

donde,
q es un número aleatorio uniformemente distribuido entre 0 y 1,
q_0 es un parámetro,
S es una variable aleatoria seleccionada según la distribución de probabilidad definida por la ecuación 6.3

Esta regla sigue favoreciendo las transiciones a los nodos conectados por arcos cortos con gran cantidad de pheromone. El parámetro q_o determina el equilibrio entre

los dos objetivos: explotación vs exploración.

El papel de la regla de actualización local es mezclar los recorridos, tal que sitios tempranamente visitados por un recorrido de una hormiga sean explorados más tarde por otras hormigas. Es decir, el efecto de esta regla es estar cambiando dinámicamente la condición de deseable, tal que cada vez que una hormiga usa un arco este es menos deseable (este pierde algo de su pheromone). Sin este esquema de actualización local, todas las hormigas buscarían en la vecindad del mejor recorrido (caso de las versiones anteriores). Así, este esquema de actualización local (disminuyendo la cantidad de pheromone en los arcos visitados, haciéndolos menos deseables en el futuro) permite hacer búsquedas de nuevos recorridos, posiblemente mejores, en todo el espacio de soluciones (no en la vecindad del mejor recorrido). Esto hace que las hormigas nunca converjan en un camino común, lo que es deseable para explorar diferentes caminos, aumentado la probabilidad de que uno de ellos encuentre la mejor solución. De esta manera, ha medida que se va construyendo la solución las hormigas van modificando los arcos que van visitando según la regla de actualización local:

$$\gamma_{ij}(t+1) = \delta\gamma_{ij}(t) + (1-\delta)\triangle\gamma_{ij} \tag{6.7}$$

donde,
δ es un coeficiente tal que (1-δ) representa la tasa de evaporación de la traza entre el intervalo de tiempo t y t+1.

Con respecto a $\triangle\gamma_{ij}$, se ha experimento con tres valores [9]: el primero fue inspirado por el esquema de aprendizaje llamado Q-learning, desarrollado para resolver problemas de aprendizaje reforzado (tales problemas consisten en agentes que deben aprender la mejor acción a realizar en cada estado en el cual se encuentren, usando como única información de aprendizaje un número escalar que representa la evaluación de la realización de cada una de las posibles acciones a ejecutar en cada estado). Este enfoque permite que la hormiga aprenda a que sitio moverse en función de su actual ubicación, lo cual implica aplicar recursivamente la regla de la ecuación 6.7. Así, se puede definir como $\triangle\gamma_{ij} = max_{(z \in AUN_k(t)-j)}(\gamma_{jz}(t))$ la cual es exactamente la misma fórmula aplicada en Q-learning. Los otros dos posibles valores son: $\triangle\gamma_{ij} = \gamma_{ij}(0)$ que es el nivel inicial de pheromone y $\triangle\gamma_{ij} = 0$. Se han obtenido mejores resultados para las dos primeras expresiones matemáticas (similares entre ellos), pero son más fáciles los cálculos para la segunda expresión.

Como se dijo antes, en esta versión solamente la hormiga globalmente mejor (la hormiga que construyo el camino más corto) puede depositar pheromone (regla de actualización global). Esto conlleva a una búsqueda mas dirigida, tal que las hormigas busquen en la vecindad del mejor recorrido conseguido en la última iteración del algoritmo. Esta actualización global, por consiguiente, es realizada después que todas las hormigas han completado su recorrido. Al usarse este esquema, el nivel de pheromone es actualizado según la ecuación:

$$\gamma_{ij}(t+n) = \alpha\gamma_{ij}(t) + (1-\alpha)\triangle\gamma_{ij} \tag{6.8}$$

tal que,

$$\triangle\gamma_{ij} = \begin{cases} 1/L_0 \; si\; el\; arco\; (i,j)\; pertenece\; al\; mejor\; recorrido\; global \\ 0\; de\; lo\; contrario \end{cases} \quad (6.9)$$

donde,
L_0 es la longitud del mejor recorrido global.

La ecuación 6.9 estipula que solo aquellos arcos que pertenecen al mejor recorrido serán reforzados. Otra versión de este último esquema de actualización global es el enfoque *iteración-mejor*, (opuesto al global-mejor anterior), el cual usa el parámetro L_0^s (longitud del mejor recorrido hasta la s-esima ciudad actualmente visitada), tal que los arcos que son reforzados son los pertenecientes al mejor parcial recorrido hasta el actual movimiento. Se ha demostrado que la diferencia entre las dos versiones es mínima. Para esta versión, el algoritmo general es el siguiente:

1. Inicializar todos los valores ($\gamma_{ij}(0)$, $\triangle\gamma_{ij}$, número de ciclos, etc.).
2. Repetir para k= 1, m
   (a) Colocar cada hormiga k en un sitio inicial r escogido aleatoriamente entre 1 y n ($LISTA_k(1) = r$)
3. Repetir para s=2, n
   (a) Repetir para k=1, m
      i. Escoger nodo j donde deberá moverse la hormiga k, a sabiendas que esta actualmente en el nodo i, según ecuación 6.6.
      ii. Mover hormiga k al nodo j.
      iii. Insertar nodo j en $LISTA_k(s)$.
      iv. Insertar arco (i,j) en $RECORRIDO_k(s)$.
   (b) Repetir para k=1, m
      i. Actualizar $\gamma_{RECORRIDO_k(s)}(t+1))$ según ecuación 6.7
   (c) t=t+1
4. Repetir para k=1, m
   (a) Calcular el costo del recorrido descrito por $LISTA_k$ y guardar en L_k.
5. $L_{Optimo} = min_k(L_k)$, $\forall k = 1, m$
6. Actualizar arcos del recorrido de L_{Optimo} según ecuación 6.9.
7. Determinar si recorrido más corto actual (L_{Optimo}) es mejor que anterior (L'_{Optimo}). Si es el caso: $L'_{Optimo} = L_{Optimo}$.
8. Si número ciclos es igual al máximo o existe comportamiento estancado

   (a) Imprimir recorrido mas corto ($L'Optimo$). de lo contrario
   (b) limpiar la lista.
   (c) ir a 2.
   (d) número ciclos=número ciclos+1

6.2.3 Análisis Experimentales

Pasaremos a analizar la eficiencia del Sistema Artificial de Hormigas al no considerar la parte heurística ($\beta = 0$) o el nivel de pheromone ($\theta = 0$). Para eso, usaremos como ejemplo los resultados mostrados por Dorigo et al. en [5, 7]. En la gráfica 6.4 se muestran los resultados de dicho trabajo, en la cual se puede observar que en ambos casos los resultados son peores que en el estándar (que usa ambas partes), e incluso, el caso que no usa pheromone es aun peor (esto confirma la importancia de la cooperación). Esto se debe a que el caso que no usa pheromone es un algoritmo heurístico egoísta con varios puntos de inicio, mientras que en el caso en que no se usa la parte heurística el sistema es guiado por el reforzamiento provisto por las reglas de actualización del pheromone. Estos resultados indican la efectividad de los Sistemas Artificiales de Hormigas para explotar la cooperación mediante el pheromone (intercambiando información a través del pheromone). Se puede observar que se llega a mejores soluciones (incluso a la óptima) si hay cooperación, mientras que en los otros casos, puede que incluso no se lleguen a obtener buenas soluciones.

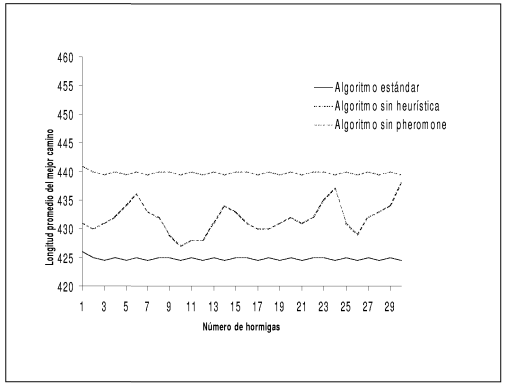

Figura 6.4: Comparación del algoritmo estándar de colonias de hormigas, con otros sin la parte heurística y sin la parte de Pheromone

El número de hormigas, según la literatura [5, 7], debe estar entorno al valor de 10.

Además, según resultados previos [5, 7] las hormigas inicialmente deben ser uniformemente distribuidas a través de todos los nodos. Otro aspecto importante a considerar es el "comportamiento estancado", situación en la que todas las hormigas siguen el mismo recorrido, lo cual significa que el sistema ha cesado de explorar nuevas posibilidades y no se consiguen mejores recorridos. Según ciertos parámetros del algoritmo, después de ciertos ciclos de esté, todas las hormigas siguen el mismo recorrido sin importar la naturaleza estocástica del algoritmo debido al alto contenido de trazas en los arcos del recorrido óptimo con respecto a los otros. El valor alto del pheromone de esos arcos evita que las hormigas escojan arcos no pertenecientes al recorrido rápido. Este comportamiento depende del parámetro α, un alto valor de α significa que la traza es muy importante y que las hormigas tenderán a escoger arcos escogidos por otras hormigas en el pasado. Por otro lado, un valor bajo de α hace de este algoritmo un proceso clásico estocástico egoísta.

Esta técnica ha sido comparada con otras técnicas, particularmente en el problema del viajero de comercio en todas sus versiones (casos simétricos y asimétricos, etc.). Además, recientemente se han hecho esfuerzos para extenderlo a otros tipos de problemas de optimización combinatoria.

En general, para poder ser aplicado a cualquier problema de optimización combinatoria, se deben seguir los siguientes pasos:

- Definir el problema como un grafo de búsqueda con múltiples agentes. Para eso se debe mapear el espacio de soluciones del problema en el ambiente de las colonias de hormigas (grafos cuyos nodos representan los sitios que deben recorrer las hormigas).
- Definir el mecanismo de autocatálisis, es decir, el proceso de actualización del pheromone,
- Diseñar la heurística que permite una definición constructiva de las soluciones y,
- Definir el método para generar soluciones válidas.

6.2.4 Comentarios Finales

En todo Sistema Distribuido, uno de los temas más importantes a considerar es el protocolo de comunicación. En el caso de las colonias de hormigas, las hormigas se comunican a través de una estructura de datos global: después de cada recorrido, la traza dejada por cada recorrido de hormiga cambiará la probabilidad con la cual la misma decisión será tomada en el futuro. Una heurística también guía a las hormigas en las primeras fases del proceso computacional, cuando aún no existe experiencia acumulada en la estructura. Esta heurística pierde importancia (recordarse del coeficiente de evaporación) cuando la hormiga empieza a ganar experiencia.

La idea general es que una población de agentes es guiada por un proceso autocatálitico distribuido, además de cierta fuerza egoísta. Si un agente esta solo, tanto el proceso como la fuerza tienden a hacer converger al agente en una solución *suboptima*. Si hay agentes que interactuan, la fuerza egoísta puede dar correctas sugestiones al proceso autocatalitico, lo que puede llevar al sistema a converger en muy buenas, frecuentemente óptimas, soluciones. Este comportamiento es debido a que los agentes adquieren información durante el proceso de búsqueda que les permite modificar la representación del problema, de tal manera de reducir el espacio considerado por el proceso de búsqueda. Las principales contribuciones de esta técnica son:

- Se introduce un mecanismo de retroalimentación como una herramienta poderosa de búsqueda y optimización. La idea es que si una hormiga debe escoger una opción y tiene información de escogencias previas, entonces ella deberá escoger aquella que sea la mejor.
- La efectividad de la búsqueda hecha por hormigas cooperativas es mayor que si estas actuaran independientemente (efecto sinergético de los sistemas distribuidos).

Los trabajos más relacionas con esta área de investigación son:

- Los estudios del comportamiento social de los animales. De las metáforas naturales, entre las características más importantes explotadas, se podrían nombrar: el paralelismo implícito, la naturaleza estocástica y adaptativa, y la autocatálisis.
- Las investigaciones sobre algoritmos heurísticos naturales.
- La optimización estocástica.

Queda por evaluar el uso de esta técnica en problemas tales como reconocimiento de patrones, máquinas de aprendizaje, etc. Finalmente, otras versiones del algoritmo deberán ser evaluadas: por ejemplo, cuantas hormigas deben contribuir en la regla de actualización global? (a mayor número quizás habrá menos posibilidad de ser atrapado en un óptimo local, sin que sea necesario dejar a todas actualizar las trazas); Por que no definir una regla de actualización global o local negativa? (si una hormiga produce un recorrido muy malo, entonces sustraer pheromone de los arcos que componen su recorrido); etc.

6.3 Sistemas Híbridos Inteligentes

6.3.1 Generalidades

Una de las características más resaltantes de los humanos es que realizan un procesamiento de información que proviene de fuentes híbridas. Nuestras acciones son

gobernadas por una combinación de información genética e información adquirida a través del aprendizaje. La información de nuestros genes ha sobrevivido y ha sido probada durante muchos años. El aprendizaje humano procesa un conjunto de información compleja derivada de su interacción con el ambiente. A partir de la combinación de esos tipos de información es que los humanos tienen la habilidad para actuar en ambientes complejos, rápidamente cambiantes.

Estos tipos de sistemas que procesan información híbrida están siendo replicados en máquinas adaptativas. En el corazón de estas máquinas adaptativas están las técnicas de computación inteligente, y en particular, los Sistemas Híbridos Inteligentes. Los sistemas híbridos inteligentes consisten en combinar dos o mas técnicas inteligentes, las cuales pueden ser combinadas con sistemas computacionales convencionales, tales como bases de datos y paquetes estadísticos. Se podrían resumir el porque crear sistemas híbridos inteligentes, en las siguientes tres razones [10, 12, 13]:

- *Debilidad de las técnicas:* el objetivo de los Sistemas Híbridos Inteligentes es combinar las técnicas de tal manera que las debilidades de unas sean cubiertas por las habilidades de las otras y viceversa. Así, las técnicas con ciertas debilidades en una propiedad, son combinadas con otras que son buenas en esa propiedad.

- *Multiplicidad de tareas:* una técnica puede no ser aplicable a los muchos subproblemas en los cuales se puede descomponer un problema.

- *Multifuncionalidad:* la motivación es crear sistemas con capacidades de múltiples mecanismos de procesamiento dentro de su misma arquitectura. Es decir, se requiere diseñar sistemas cuyos componentes realicen tipos diferente de procesamiento de información.

En particular, el desarrollo de los Sistemas Híbridos Inteligentes en el área de Computación Inteligente es bastante deseable, debido a que cada una de las técnicas tienen un conjunto de propiedades computacionales que las hacen susceptibles para ciertos problemas y no para otros. Por ejemplo, las redes neuronales son buenas para reconocer patrones, pero no tienen la capacidad de explicar como ellos alcanzan una decisión. Los sistemas de inferencia difusos pueden razonar con información imprecisa, pero ellos no pueden adquirir automáticamente las reglas que usarán para su proceso de toma de decisiones. Esto último también ocurre en los sistemas expertos.

6.4 Propiedades de los Sistemas Inteligentes

En esta parte presentamos las propiedades de las Técnicas Inteligentes desde el punto de vista de sus capacidades para el procesamiento de información [10, 12, 13].

6.4.1 Adquisición del conocimiento

La adquisición del conocimiento es una etapa fundamental para el desarrollo de sistemas inteligentes. Esta etapa envuelve capturar, interpretar y representar el conocimiento. En el caso de los sistemas expertos este proceso es largo, costoso, y muchas veces, irrealizable. Otras técnicas, tales como las Redes Neuronales Artificiales y los Algoritmos Genéticos, tienen ciertas ventajas porque pueden aprender desde los datos. Los sistemas de inducción de reglas son otro ejemplo de técnicas que han automatizado el proceso de adquisición de conocimiento.

6.4.2 Robustez

Un sistema no es robusto si responde apropiadamente solamente en ciertos dominios y requieren una sustancial intervención humana para compensar sus debilidades en otros dominios. Un tipo de problema de un sistema no robusto es su imposibilidad para actuar con conocimiento incompleto, inconsistente o inexacto. Las causas pueden ser una inadecuada representación de las estructuras cognitivas o de los mecanismos de razonamiento. Este es un problema que ocurre clásicamente en los sistemas expertos. En las otras técnicas, esto es resuelto de diferentes formas. En los Algoritmos Genéticos manteniendo poblaciones de soluciones, en la lógica difusa a través de su representación de conocimiento y método de razonamiento, y en las Redes Neuronales a través de la representación distribuida del conocimiento y razonamiento.

6.4.3 Alto y Bajo nivel de razonamiento

Según la teoría cognitiva existen dos niveles de razonamiento, un alto nivel, en el cual realizamos tareas cognitivas secuencialmente, y un bajo nivel, en el cual realizamos tareas paralelas de reconocimiento. Por ejemplo, los sistemas expertos tienen habilidades para realizar tareas cognitivas de alto nivel , tales como razonamiento en base a objetivos. Por el contrario, las Redes Neuronales ofrecen habilidades para realizar tareas de reconocimiento de patrones (bajo nivel), tales como procesamiento visual, etc.

6.4.4 Capacidad de Explicar

La habilidad para proveer a los usuarios con explicaciones sobre el proceso de razonamiento es importante en ciertos tipos de aplicaciones. Dichas explicaciones son importantes para aceptar soluciones generadas por ellos, o para clarificar el proceso de razonamiento. En los sistemas expertos y en los sistemas de inducción de reglas, las explicaciones son típicamente provistas por las cadenas de inferencia durante el proceso de razonamiento. Los sistemas de lógica difusa toman decisiones finales a partir de la agregación de decisiones. Por el contrario, en las Redes Neuronales es difícil dar una explicación, ya que ellos no tienen una implícita representación del conocimiento.

Esto dificulta encontrar una cadena de inferencia la cual pueda ser usada para dar explicaciones.

6.4.5 Clasificación

Existen varias clasificaciones en la literatura. La primera que daremos esta caracterizada por su funcionalidad, arquitectura de procesamiento y requerimientos computacionales [10, 12].

- Reemplazando Funciones:
  En esta clase de hibridación, ciertas actividades (funciones) de una técnica son realizadas por otra técnica inteligente. La motivación para realizar esto puede ser para aumentar la eficiencia de la técnica. Algunos ejemplos de estos sistemas son:
  - Reemplazar el mecanismo de aprendizaje de una Red Neuronal (modificación de los pesos) por un Algoritmo Genético.
  - Usar las Redes Neuronales para construir controladores lógicos difusos
  - Definir automática las funciones de membrecia usando Algoritmos Genéticos
  - Usar un Algoritmo Genético o una Red Neuronal para definir un sistema basado en reglas difusas
  - Reemplazar el operador de cruce de un Algoritmo Genético por una Red Neuronal

- Intercomunicando Técnicas Inteligentes:

  Consiste en interconectar módulos inteligentes de procesamiento, los cuales intercambian información y ejecutan funciones separadas. Esto es útil en problemas que se puedan descomponer en tareas, tal que uno pueda usar módulos independientes para ir resolviendo cada tarea. Estos módulos independientes que colectivamente resuelven el problema, deben ser coordinados por un mecanismo de control. Por ejemplo, si un problema necesita reconocer patrones y optimizar ciertos parámetros se podrían usar Redes Neuronales y Algoritmos Genéticos para cada una de esas tareas. Algunos ejemplos de estos sistemas son:
  - Métodos de pre-procesamiento difuso que son introducidos a una Red Neuronal para tareas de diagnóstico
  - Reconocimiento de patrones de Datos a través de la siguiente secuencia: análisis estadísticos, reglas de inducción de patrones y mecanismos de agrupamiento (que pueden ser implementados usando Redes Neuronales o Algoritmos Genéticos).

- Generando Sistemas polimórficos:

  Estos son sistemas donde a partir de una única arquitectura de procesamiento se realizan todas las funcionalidades de las diferentes técnicas de procesamiento. La motivación de estas técnicas es poder realizar tareas multifuncionales en una única arquitectura computacional. En este grupo son muy pocos los trabajos encontrados en la literatura.

Dillon y Khosla [13] presentan otra clasificación, la cual agrupa los sistemas híbridos inteligentes en cuatro categorías:

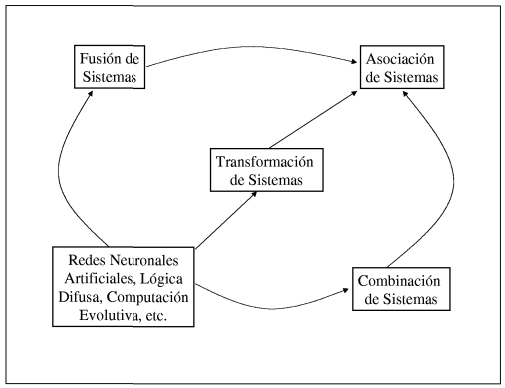

Figura 6.5: Clases de Sistemas Híbridos Inteligentes según Dillon y Khosla

- Fusión de Sistemas:

  En esta categoría, los mecanismos de procesamiento y representación de la información de una técnica inteligente son fusionados con la estructura de representación de otra técnica inteligente, es decir, el conocimiento y/o razonamiento de una técnica es fusionado en la otra. Desde un ángulo práctico, puede ser visto como la manera en que una técnica inteligente trata de resolver sus debilidades y explotar sus fortalezas para resolver un problema en particular. Esto es realizado usando otra técnica inteligente dentro de ella misma. Lo que es importante resaltar es que la resolución del problema es basada en la técnica sobre la cual se hace la fusión. Algunos de los ejemplos de estos sistemas son los siguientes:
  - Modelar el Razonamiento simbólico usando Redes Neuronales.
  - Implementar operadores difusos usando Redes Neuronales.

  - Concebir sistemas difusos usando Redes Neuronales o Algoritmos Genéticos.
  - Optimizar los pesos de las Redes Neuronales usando Algoritmos Genéticos.
  - Determinar la estructura de las Redes Neuronales usando Algoritmos Genéticos.

- Transformación:

  En estos sistemas, una técnica inteligente transforma la forma de representación de la información en otra forma, para poder ser usada por otra técnica inteligente. Desde el punto de vista práctico, estos sistemas se usan cuando el conocimiento o información requerida por la Técnica Inteligente a usar, no esta disponible en la forma requerida por ella. Así, para poder resolver el problema se necesitan las dos representaciones definidas por las dos técnicas inteligentes que se estén usando. Algunos de los ejemplos de aplicación de estos sistemas, son los siguientes:

  - Fusificar la data de entrada para una Red Neuronal.
  - Optimizar la data de entrada a una Red Neuronal, usando Algoritmos Genéticos (por ejemplo, para disminuir la dimensión de la data de entrada).

- Combinación de Sistemas:

  Este enfoque consiste en combinar dos o mas técnicas inteligentes, bajo la premisa que no hay una técnica inteligente que abarque todo el espectro referente a las capacidades cognitivas y de procesamiento de los humanos a la hora de querer resolver un problema. En otras palabras, como cada técnica inteligente refleja diferentes aspectos del proceso cognitivo de los humanos, combinándolas se puede desarrollar un mejor y mas completo sistema de procesamiento de información, y por consiguiente, una mas poderosa estrategia de resolución de problemas.

  Por consiguiente, en este caso se hace una hibridización explícita, es decir, se asocian los diferentes niveles de procesamiento de información y de la inteligencia con la técnica inteligente que mejor lo haga, reteniendo la identidad de cada técnica inteligente de una manera clara (eso no sucede en los sistemas fusionados). Esto implica una estructura modular en el sistema. Algunos de los ejemplos de aplicación de estos sistemas, son los siguientes:

  - Sistemas de reconocimiento neuro-difusos
  - Controladores híbridos neuro-difusos o neuro-genético-difusos, para tareas de control de plantas, de movimiento de robots, de manejo de fallas, etc.

- Asociación de Sistemas

  Estos sistemas intentan asociar las categorías presentadas anteriormente, de manera de maximizar las capacidades de procesamiento y de eliminar las restricciones a nivel de representación del conocimiento. Así, las arquitecturas híbridas asociativas incorporan conceptos de las tres arquitecturas vistas anteriormente, lo que permite solventar los problemas presentes en cada uno de estos sistemas. Por ejemplo, en la fusión de sistemas normalmente se hace una conversión del conocimiento explícito en conocimiento mecanismo, o viceversa, lo que hace perder aspectos declarativos del problema bajo estudio. Las arquitecturas de transformación tienen problemas de escalabilidad cuando aumenta la complejidad de las tareas a realizar, incluso, generando ruido o perdida de datos. En el caso de la combinación de sistemas, estos tiene problemas a nivel del proceso de transferencia de conocimiento o de información entre módulos que representan técnicas inteligentes diferentes, por consiguiente, sufriendo la calidad de los datos que son transferidos.

  Este tipo de sistema híbrido no ha sido muy utilizado en la resolución de problemas. Uno de los trabajos pioneros en esta área es el presentado por Dillon y Khosla en el libro [13]. En dicho trabajo, ellos presentan una aplicación de esta arquitectura en el diseño y desarrollo de un sistema de procesamiento de alarmas en tiempo real.

6.4.6 Metodología

En esta sección se presenta un ciclo clásico de desarrollo de Sistemas Híbridos. Este enfoque procedimental pueden llevar a reducir los costos y tiempos de desarrollo. Este ciclo esta compuesto por las siguientes fases (ver figura 6.6):

- Análisis del problema: Esta fase envuelve dos pasos
  - Identificación de las tareas existente dentro del problema a tratar: Esto no siempre es el caso, ya que hay problemas que no pueden ser descompuestos en tareas.
  - Identificación de los requerimientos del problema (y si fuera el caso, de sus tareas). Es decir, definir los mecanismos de razonamiento, adquisición del conocimiento,tipo de información a usar, etc.

- Análisis de la relación entre las propiedades de las técnicas inteligentes y los requerimientos del problema:

  Consiste en hacer la correspondencia entre las propiedades de las técnicas inteligentes disponibles con los requerimientos identificados en la fase anterior. Para eso, se puede usar la tabla siguiente:

	Propiedades				
Técnicas	Adquisición de conocimiento	Robustez	Razonamiento explicito	Razonamiento implicito	Explicativo
Sistemas expertos	*	*	*****	*	*****
Sistemas Difusos	*	*****	***	*****	****
Redes Neuronales	*****	*****	*	*****	*
Algoritmos Genéticos	*****	***	***	***	***

Tabla 6.1: Relación entre técnicas inteligentes y propiedades

Esta tabla provee un esquema para determinar los grados de competencia de cada técnica inteligente para la realización de cada una de las tareas requeridas.

- Selección del tipo de sistema a utilizar:

  En esta fase se selecciona el sistema requerido para resolver el problema. Si el problema puede ser resuelto por una sola técnica (es la que mejor clasifica en la tabla anterior para todos los requerimientos), obviamente no se escoja usar un sistema híbrido. Si el problema esta descompuesto en tareas que no se solapan y si existen técnicas diferentes que claramente hagan juego con cada una de esas tareas, lo ideal será usar un sistema de intercomunicación o combinación. Si existen tareas que no pueden ser resueltas usando solamente una técnica inteligente, escoja usarse un enfoque híbrido de reemplazo de funciones, de fusión o de transformación. Finalmente, un sistema polimórfico o asociativo es válido en situaciones donde la funcionalidad cambia dinámicamente, lo que requiere de la habilidad de cambiar de un estilo de procesamiento a otro.

- Implementación:

  En esta fase se necesita definir las herramientas de programación y los ambientes necesarios para implementar el sistema. Por ejemplo, usando la programación orientada por objetos se pueden representar las diferentes técnicas de procesamiento y mezclar diferentes estilos de procesamiento dentro de la misma aplicación.

- Validación:

  Esta fase es usada para probar y verificar la funcionalidad de los componentes de la aplicación, y del sistema como un todo. Si un funcionamiento erróneo es descubierto, entonces toda la aplicación debe ser reconcebida o refinada, dependiendo de la magnitud del problema.

- Mantenimiento:

  Consiste en, periódicamente, evaluar el rendimiento del sistema, y refinarlo en los casos necesarios. El mantenimiento es fundamental en los componentes adaptativos (Redes Neuronales, Algoritmos Genéticos) cuando se esta en ambientes rápidamente cambiantes, donde los rendimientos pueden fácilmente degradarse si no hay un continuo reentrenamiento con datos recientes.

En las tres primeras fases es imprescindible la participación de un experto del área bajo estudio. Para todo el desarrollo es fundamental la participación de un experto en técnicas inteligentes. Para las últimas tres fases, un experto en programación es requerido.

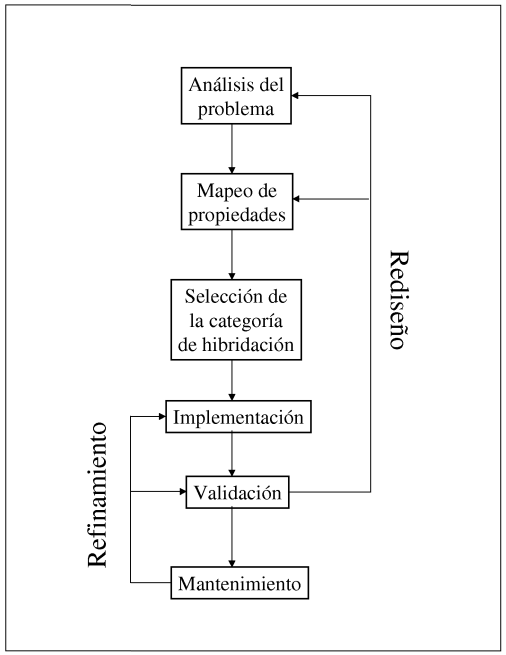

Figura 6.6: Metodología para desarrollar Sistemas Híbridos Inteligentes

Bibliografía

[1] E. Bonabeau, M. Dorigo, G. Theraulaz, "From Natural to Artificial Swarm Intelligence", *Oxford University Press*, 1999.

[2] A. Coloni, M. Dorigo, V. Maniezzo, "Distributed Optimization by Ant Colonies", in *Proc. European Conference on Artificial Life*, 1991, pp. 134-142.

[3] D. Corne, M. Dorigo, F. Glove, "New ideas in Optimization", *Mc Graw Hill*, 1999.

[4] M. Dorigo, *Optimization, Learning and Natural Algorithms*, Ph.D Thesis, Plitecnico de Milano, Italy, 1992.

[5] M. Dorigo, V. Maniezzo, A. Coloni, "The ant system: Optimization by a colony of cooperating agents", *IEEE Trans. Syst. Man, Cybern.*, vol. 26, no. 2, pp. 29-41, 1996.

[6] M. Dorigo, L. Gambardella, "A study of some propierties of Ant-Q", in *Proc. 4th Int. Conf. Parallel Problem Solving from Nature*, 1996, pp. 656-665.

[7] M. Dorigo, L. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", *IEEE Trans. on Evolutionary Computation*, vol. 1, no. 1, pp. 53-66, 1997.

[8] J. Ferber, *Les Systèmes Multi-Agents vers Intelligence Collective*, *Inter Editions*, 1995.

[9] L. Gambardella, M. Dorigo, "Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem", In *Proc. of 12th Int. Conf. on Machine Learning*, 1995, pp. 252-260.

[10] S. Goonatilake, S. Khebbak, "Intelligent Hybrid System", *John Wiley and Son*, 1998.

[11] B. Holldobler, E. Wilson, *The ants*, *Springer-Verlag*, 1990.

[12] L. Jain, R. Jain (Ed.), "Hybrid Intelligent Engineering Systems", *1997 World Scientific Publishing*, 1997

[13] R. Khosla, T. Dillon, "Engineering Intelligent Hybrid Multi-Agent Systems", *Kluwer Academic Publishers*, 1997

[14] C. Langton (Ed), *Artificial Life III*, *Addison Wesley*, 1994.

[15] C. Langton (Ed.), "Artificial Life IV", *MIT Press* , 1995

[16] R. Schoonderwoerd, O. Holland, J. Bruten, L. Rothkrantz, "Ant-based Load Balancing in Telecommunications Networks", *Adaptive Behavior*, vol. 5, no. 2, pp. 169-207, 1997.

[17] Y. Stutzle, H. Hoos, "The Max-Min Ant System and Local Search for the Traveling Salesman Problem", In *Proc. 4th Int. Conf. on Evolutionary Computation*, 1997, pp. 309-314.