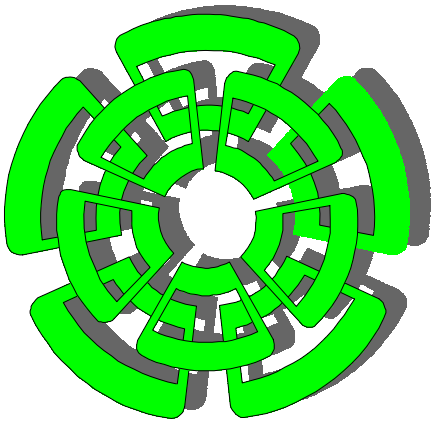

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL
INSTITUTO POLITÉCNICO NACIONAL

DEPARTAMENTO DE CONTROL AUTOMÁTICO

Modelado difuso neuronal con algoritmo de aprendizaje estable

TESIS QUE PRESENTA
Carlos Alejandro Villaseñor Lozano

PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS

EN LA ESPECIALIDAD DE
CONTROL AUTOMÁTICO

DIRECTORES DE TESIS:
Dr. Wen Yu Liu
Dra. Xiaoou Li Zhang

México, D.F., abril del 2003.

Índice general

Capítulo 1

Introducción

Las redes neuronales y la lógica difusa son estimadores universales, pueden aproximar cualquier función no lineal a cualquier precisión que se pida con la condición de que estén disponibles suficientes neuronas ocultas o reglas difusas. Resultados recientes muestran que el procedimiento de fusión de dos tecnologías diferentes parece ser muy efectivo para una amplia categoría de sistemas complejos no lineales, cuando no tenemos la información completa de la planta[1][4][13].

Los algoritmos de aprendizaje de gradiente descendiente y el de propagación hacia atrás (BP por sus siglas en inglés, Back Propagation) se usan siempre para ajustar los parámetros de las funciones de membresía (conjuntos difusos) y los pesos de desdifusificación (redes neuronales). La baja velocidad de aprendizaje y el mínimo local son desventajas de estos algoritmos de aprendizaje [14]. Algunas modificaciones fueron derivadas [3] sugiriendo un aprendizaje robusto BP para resistir el efecto del ruido y evitar la tendencia al error durante la aproximación. [23] usó funciones de membresía B-spline para minimizar la función de objeto robusta, la velocidad de convergencia se mejoró también. Se aplicaron redes neuronales RBF (por sus siglas en ingles, Radial Base Function) en [20] para determinar la estructura y los parámetros de sistemas difuso neuronales.

Para los ingenieros es muy importante asegurar la estabilidad en la teoría antes de aplicar técnicas de modelado difuso neuronal al sistema real. Es bien sabido que algoritmos normales de identificación (por ejemplo gradiente descendiente y mínimos cuadrados) son estables para condiciones ideales. En la presencia de dinámicas no modeladas estos procedimientos adaptivos pueden desestabilizarse fácilmente. La falta de robustez de la identificación de parámetros fue demostra-

da en [5] y se volvió un asunto de interés en 1980, año en el cual se sugirieron algunas técnicas de modificación robusta para identificación adaptable [8]. La actualización de los pesos del modelo neuro-difuso es un tipo de identificación de parámetros. El aprendizaje normal de gradiente descendiente y el algoritmo BP son estables, si el modelo difuso neuronal puede seguir a la planta exactamente. Generalmente debemos hacer algunas modificaciones a estos algoritmos para que el proceso de aprendizaje sea estable. El operador proyección es muy efectivo para asegurar los parámetros acotados por el modelado difuso [22]. Técnicas de proyección también usadas por muchos sistemas difuso neuronales [11]. Otro método generalizado es usar técnicas de modificación robusta [8] en modelado difuso neuronal. Por ejemplo, [24] modificación-σ, para prevenir la tendencia de los parámetros.

El modelado difuso neuronal es en el sentido de aproximación de caja negra. Todas las incertidumbres pueden ser estimadas como parte de la caja negra, esto es, las dinámicas no modeladas pueden ser consideradas dentro del modelo, no como incertidumbres estructuradas. Por lo tanto las técnicas comunes de robustificación no son necesarias. En [21], los autores sugirieron velocidades de aprendizaje óptimos y estables sin modificación robusta. La velocidad óptima fue difícil de encontrar, para lo cuál el algoritmo de búsqueda genética fue usado. Usando teoría de pasividad, aprobamos satisfactoriamente que el algoritmo de gradiente descendiente sin modificación robusta es estable y robusto a cualquier incertidumbre acotada para redes neuronales de tiempo continuo [25] e identificación neuronal de tiempo continuo [26].

Aquí se aplica el acercamiento de estabilidad entrada-a-estado (ISS por sus siglas en inglés), para obtener nuevas leyes de aprendizaje para modelado neuronal tipo Mamdani y Takagi-Sugeno-Kang. El acercamiento ISS para sistemas difuso neuronales aún no ha sido aplicado en la literatura. Comparado con [21], también usamos velocidades de aprendizaje de tiempo variable, pero esta velocidad de aprendizaje puede ser calculada directamente de los datos de entrada-salida. El algoritmo de aprendizaje es más simple que [21]. Aquí se discuten dos casos: las funciones de membresía de la parte consecuencia (la parte ENTONCES) son desconocidas donde se usa el aprendizaje gradiente descendiente, y las funciones de membresía de la parte premisa (la parte SI) son desconocidas donde se usa el algoritmo BP.

1.1. Objetivo y motivación de la tesis

El objetivo de ésta tesis es el de encontrar un método que elimine la necesidad de usar métodos robustos al la ley de gradiente descendiente y el algoritmo Back Propagation usando un acercamiento de estabilidad entrada-a-estado.

1.2. Organización de la tesis

En el primer capítulo se muestra una pequeña introducción de lo que son las redes neuronales y las redes difusas, así como de lo que es la estabilidad entrada-a-estado, en el segundo capítulo se muestra una explicación más detallada de lo que son y cómo funcionan las redes neuronales, se muestran algunos tipos de redes neuronales y sus ventajas y desventajas, en el tercer capítulo se muestra a detalle cómo funcionan las redes difusas, sus distintas partes y la implantación con las redes neuronales para en éste caso tener las redes difuso neuronales, en el cuarto capítulo se muestra el método propuesto de red difuso neuronal al conocerse la función de membresía y finalmente se muestra el método propuesto al no conocerse la función de membresía en el capítulo 5.

1.3. Publicaciones

Parte de esta tesis se presentó como el artículo titulado "Modelado de sistemas no lineales vía redes difuso neuronales con algoritmo de aprendizaje estable"en el *X Congreso Latinoamericano de Control Automático* celebrado en Guadalajara, Jal. del 3 al 6 de diciembre de 2002.

1.4. Estabilidad entrada-a-estado

La estabilidad entrada-a-estado (ISS por sus siglas en ingles, Input-to-State Stability) es otro acercamiento elegante para analizar la estabilidad además del método de Lyapunov. Puede llevar a conclusiones generales en estabilidad usando características de la entrada y del estado. Considere el siguiente sistema en tiempo discreto no lineal

$$\begin{aligned} x(k+1) &= f\left[x\left(k\right), u\left(k\right)\right] \\ Y(k) &= h\left[x\left(k\right)\right] \end{aligned} \tag{1.1}$$

donde $u(k) \in \Re^m$ es el vector de entrada, $x(k) \in \Re^n$ es el vector de estado, y $Y(k) \in \Re^m$ es el vector de salida. f y h son funciones generales suaves no lineales $f, h \in C^\infty$. Recordemos las siguientes definiciones.

Definición 1.1 *Un sistema (1.1) se dice que tiene estabilidad global entrada-a-salida si existe una función-$\mathcal{K}$, $\gamma(\cdot)$ (continua y estricta creciente $\gamma(0) = 0$) y una función-$\mathcal{KL}$ $\beta(\cdot)$ (función-$\mathcal{K}$ y $\lim_{s_k \to \infty} \beta(s_k) = 0$), tal que, para cada $u \in L_\infty$, i.e., $\sup\{\|u(k)\|\} < \infty$, y cada estado inicial $x^0 \in R^n$, esto mantiene que*

$$\left\|x\left(k, x^0, u(k)\right)\right\| \leq \beta\left(\left\|x^0\right\|, k\right) + \gamma\left(\|u(k)\|\right) \tag{1.2}$$

Definición 1.2 *Una función suave $V : \Re^n \to \Re \geq 0$ es llamada una función suave ISS-Lyapunov para el sistema (1.1) si:*

(a) existe una función-$\mathcal{K}_\infty$ (función-$\mathcal{K}$ y $\lim_{s_k \to \infty} \alpha_i(s_k) = \infty$) $\alpha_1(\cdot)$ y $\alpha_2(\cdot)$ tal que

$$\alpha_1(s) \leq V(s) \leq \alpha_2(s), \quad \forall s \in \Re^n \tag{1.3}$$

(b) existe una función-$\mathcal{K}_\infty$ $\alpha_3(\cdot)$ y una función-$\mathcal{K}$ $\alpha_4(\cdot)$ tal que

$$V_{k+1} - V_k \leq -\alpha_3(\|x(k)\|) + \alpha_4(\|u(k)\|), \quad \text{para todo } x(k) \in \Re^n, u(k) \in \Re^m \tag{1.4}$$

Teorema 1.1 *Para un sistema de tiempo discreto no lineal, los siguientes son equivalentes [9]*

- *Es entrada-a-estado estable (ISS).*
- *Es robustamente estable.*
- *Admite una función suave ISS-Lyapunov.*

Propiedad. Si un sistema no lineal es ISS, el comportamiento del sistema se mantiene acotado cuando sus entradas son acotadas.

De (1.1) tenemos

$$\begin{gathered} Y(k) = h[x(k)] := F_1[x(k)], \quad Y(k+1) = h[f[x(k), u(k)]] := F_2[x(k), u(k)] \\ Y(k+n-1) := F_n[x(k), u(k), u(k+1), ..., u(k+n-2)] \end{gathered} \tag{1.5}$$

Denotando

$$\begin{aligned} \overline{Y}(k) &= [Y(k), Y(k+1), ..., Y(k+n-1)]^T \\ U(k) &= [u(k), u(k+1), ..., u(k+n-2)]^T \end{aligned} \tag{1.6}$$

entonces $\overline{Y}(k) = F[x(k), U(k)]$, $F = [F_1 \cdots F_n]^T$. Si $\frac{\partial \overline{Y}}{\partial x}$ es no singular en $x = 0$, $U = 0$, (1.5) puede ser expresado como $x(k+1) = g\left[\overline{Y}(k+1), U(k+1)\right]$. Esto nos lleva al modelo NARMA multivariable [2]

$$\begin{aligned} Y(k) &= h[x(k)] = \Psi[y(k-1), y(k-2), ..., u(k-1), u(k-2), ...] \\ &= \Psi[X(k)] \end{aligned} \tag{1.7}$$

donde

$$X(k) = [Y(k-1), Y(k-2), ..., u(k-d), u(k-d-1), ...]^T \tag{1.8}$$

$\Psi(\cdot)$ es una ecuación diferencial no lineal representando las dinámicas de la planta , $u(k)$ y $y(k)$ son entradas y salidas escalares medibles, d es el retraso de tiempo. Uno puede ver que de la *Definition 1,2* y el *Theorem 1* no depende de la expresión exacta del sistema no lineal. Aplicaremos ISS al modelo NARMA (1.7).

Capítulo 2

Redes neuronales

Las redes neuronales constituyen una poderosa herramienta para modelar sistemas, especialmente no lineales, sean dinámicos o estáticos.

El cerebro humano es un sistema muy complejo formado por muchas células llamadas neuronas; se estima que existen entre 10^{10} y 10^{11} células en el cerebro. Las redes neuronales artificiales emulan la arquitectura y capacidades de sistemas neuronales biológicos.

Un esquema simplificado de una neurona se muestra en la figura 2.1.

En el cuerpo celular se realizan la mayoría de las funciones lógicas de la neurona. El axón es el canal de salida final de la neurona. Las dentritas reciben las señales de entrada de los axones de otras neuronas y se conectan al cuerpo celular por medio de las sinapsis.

En la figura 2.2 se observa la estructura de una neurona artificial con múltiples entradas.

En esta estructura, se tiene

$$u = \sum w_i x_i \tag{2.1}$$

donde:

w_i son los pesos de la neurona (sinapsis)

x_i son las entradas a la neurona

n es el número de entradas a la neurona

$$y = f(u) = f\left(\sum w_i x_i - \theta\right) \tag{2.2}$$

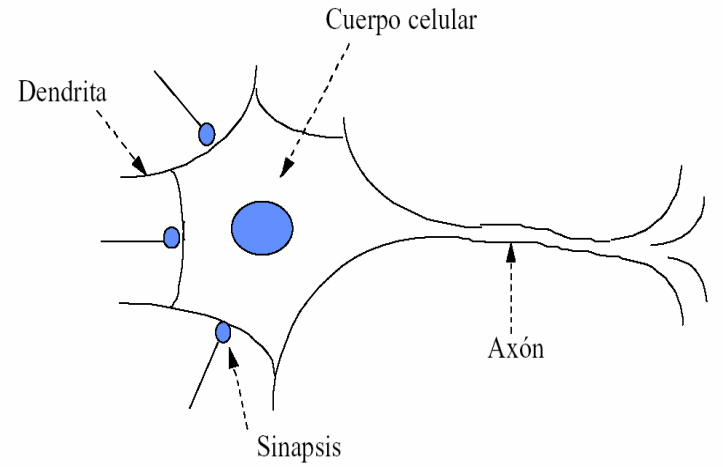

Figura 2.1: Esquema simplificado de una neurona

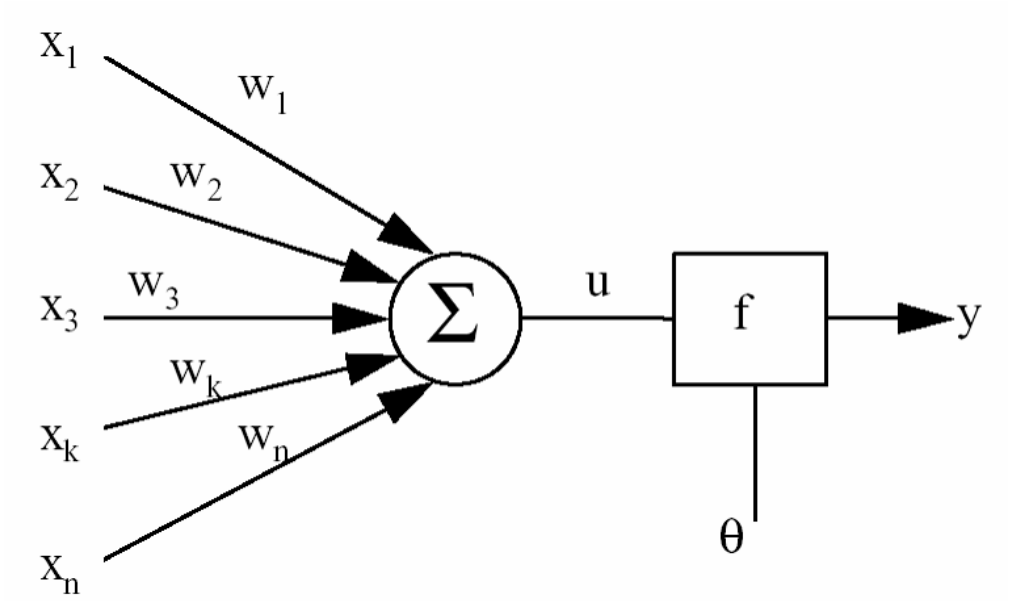

Figura 2.2: Estructura de una neurona artificial con múltiples entradas

donde:

y es la salida de la neurona (axón)

f es la función de activación, correspondiente, en general, a una función no lineal (cuerpo celular)

θ es el sesgo

En general se utilizan las siguientes funciones de activación mostradas en la figura 2.3:

Las redes neuronales son estructuras de procesamiento formadas por una gran cantidad de neuronas, que operan en paralelo.

Además, los distintos tipos de redes neuronales se generan a partir de la interconexión de neuronas.

Las principales redes neuronales que se utilizan para modelado no lineal son:

· Redes perceptrón multicapa

· Redes recurrentes

· Redes de funciones de base radiales (RBFN)

2.1. Ventajas de las redes neuronales

Las redes neuronales deben su capacidad de procesamiento de información a su estructura distribuida y paralela, a su capacidad de aprendizaje y por tanto de generalización.

Tareas

- Reconocimiento de patrones

- Memorias asociativas

- Aproximación funcional

- Etc.

Propiedades

- No linealidad. Las neuronas son elementos de proceso generalmente no lineales. La interconexión de estos elementos genera estructuras de transformación de datos donde este carácter no lineal queda distribuido a lo largo y ancho de la red.

- Modelado de relaciones de entrada/salida.

- Adaptabilidad. Las redes neuronales son por definición estructuras adaptables capaces de ajustar sus pesos y por tanto su función de transferencia a cambios en su entorno.

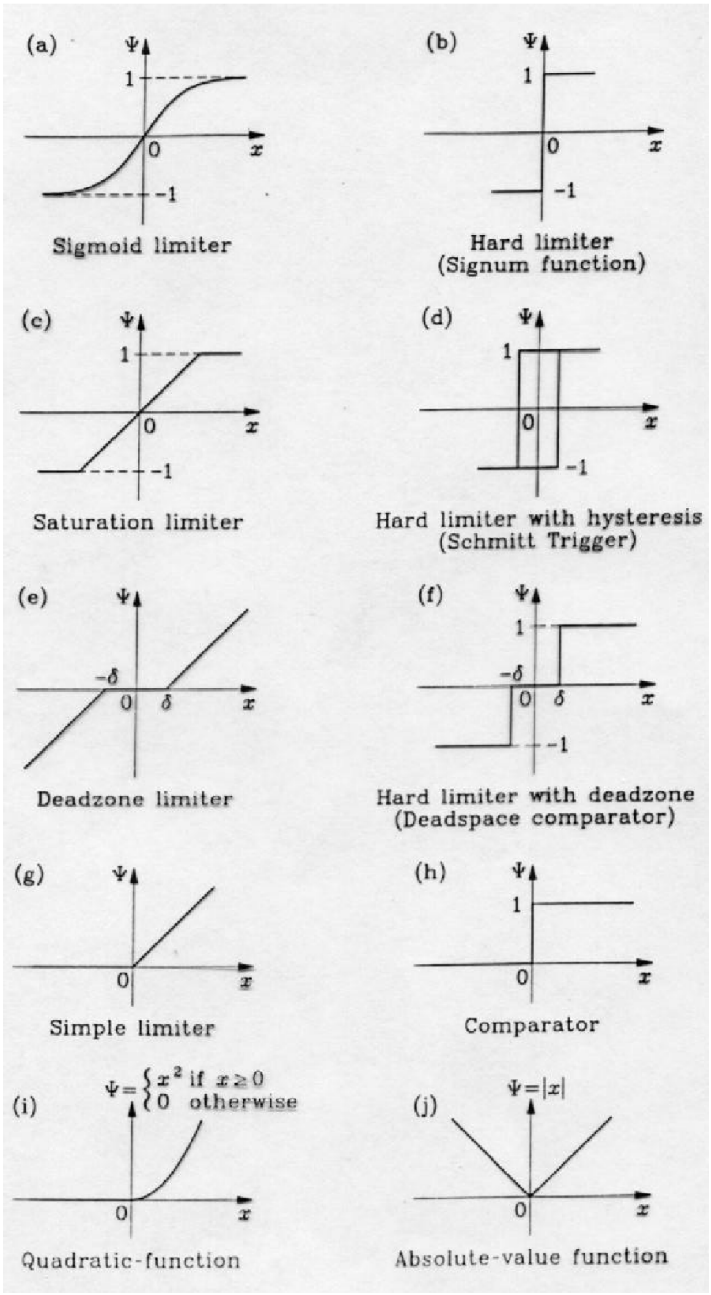

Figura 2.3: Distintos tipos de funciones de activación

- Tolerancia ante fallos. Una red neuronal tiene la capacidad de seguir respondiendo de forma no catastrófica cuando parte de su estructura está dañada. Esto es debido al tratamiento distribuido de la información y a la redundancia implícita en su estructura.

2.2. Tipos de redes neuronales

2.2.1. Perceptrón multicapa

El perceptrón multicapa es una estructura jerárquica que consiste en varias capas de neuronas totalmente interconectadas, que admiten como entradas las salidas de los elementos de proceso (neuronas) de la capa anterior.

En las redes perceptrón multicapa se distinguen tres tipos de capas:

- Capa de entrada. Esta formada por n unidades (siendo n el número de entradas externas) que se limitan a distribuir las señales de entrada a la capa siguiente.
- Capas ocultas. Están formadas por neuronas que no tienen contacto físico con el exterior. El número de capas ocultas es variable, pudiendo incluso ser nulo.
- Capa de salida. Está formado por m neuronas (siendo m el número de salidas externas) cuyas salidas constituyen el vector de salidas externas del perceptrón multicapa.

Los modelos dinámicos neuronales están dados por:

$$y(t) = N(y(t-1), ..., y(t-ny), u(t-1), ..., u(t-nu)) \tag{2.3}$$

donde N es la red neuronal que puede ser un perceptrón multicapa, como se muestra en la figura 2.4.

Aplicaciones
- Aproximación de funciones
- Reconocimiento de patrones
- Filtrado de señales
- Eliminación de ruido
- Segmentación de imágenes y señales

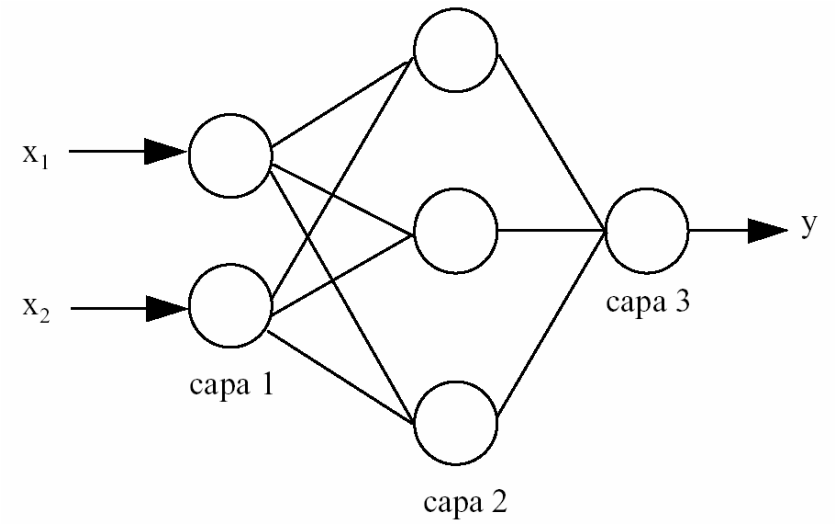

Figura 2.4: Perceptrón multicapa

- Control adaptable

- Compresión de datos

- Etc.

Ventajas

- Capacidad de representación funcional universal. Gran rapidez de procesamiento. Genera buenas representaciones internas de las características de los datos de entrada. Ampliamente estudiada. Es la red neuronal más aplicada en la práctica

Desventajas

- Tiempo de aprendizaje elevado para estructuras complejas

2.2.2. Redes recurrentes

Estos modelos son capaces de representar sistemas realimentados dinámicos no lineales (Narendra, 1990).

Además, se debe mencionar que existen diversos modelos neuronales que son combinaciones de las redes perceptrón multicapa y redes recurrentes.

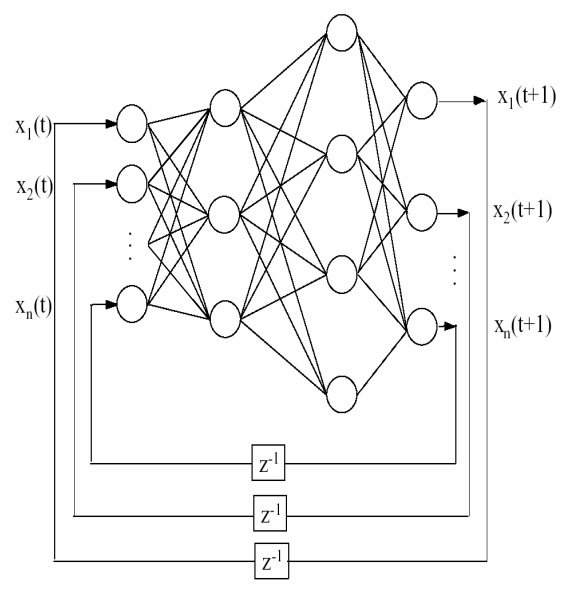

Figura 2.5: Red neuronal tipo recurrente

2.2.3. Redes de funciones de base radiales

Las redes de funciones de base radiales (RBFN Radial Basis Function Networks) consisten en dos capas (Jang, 1993). Los modelos dinámicos basados en las redes RBFN están dados por (2.3):

donde N es una red neuronal como se muestra en la figura 2.6 con $n = ny + nu$.

La capa oculta esta compuesta por n unidades radiales totalmente conectadas al vector de entrada. Las funciones de transferencia de la capa oculta son similares a una función de densidad Gaussiana, es decir:

$$a_i = \exp\left(-\frac{\|x - r_1\|^2}{\sigma_i^2}\right) \tag{2.4}$$

donde (2.3):

es el vector de entradas de la red

r_i son los centros de las unidades radiales

σ_i representan los anchos.

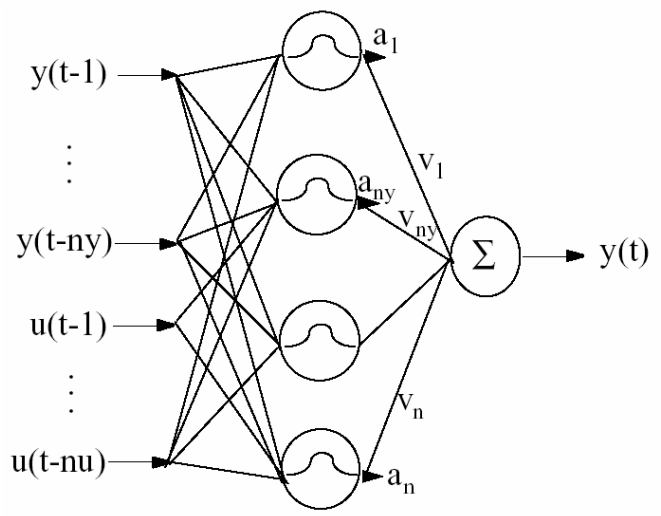

Figura 2.6: Red neuronal tipo de funciones de base radiales

La salida de la red está dada por:

$$y\left(t\right) = \sum v_i a_i \tag{2.5}$$

donde v_i son los pesos de las unidades radiales.

Aplicaciones

- Aproximación de funciones

- Reconocimiento de patrones

Ventajas

Capacidad de representación funcional universal. La estructura de esta red tiene interpretación directa, lo que permite realizar una buena inicialización de los pesos de la red, y extraer conocimiento de las estructuras ajustadas. La buena inicialización de los pesos acelera el proceso de aprendizaje.

Desventajas

El procesamiento realizado es algo más complejo que en el caso del perceptrón multicapa.

2.2.4. Otros tipos de redes neuronales

Adaline. Estas neuronas tienen capacidad de aprendizaje debido a que sus pesos son cambiados adaptivamente de acuerdo a un algoritmo adaptable. Sus aplicaciones principales son: filtrado adaptable de señales y reconocimiento de patrones. Son fácilmente implantables en hardware debido a su sencillez y homogeneidad, sin embargo sólo son capaces de resolver problemas de clasificación linealmente separables y llevar a cabo transformaciones lineales.

Mapas autoorganizativos de Kohonen. En este caso, las neuronas están ordenadas topológicamente. Frente a la presentación de un patrón n-dimensional de entrada, compiten lateralmente hasta que sólo una de ellas queda activa. El objetivo es que patrones de entrada con características parecidas queden asociados a neuronas topológicamente cercanas. Sus principales aplicaciones son: agrupación y representación de datos, compresión de datos y optimización.

2.3. Aprendizaje de redes neuronales

Se entiende por aprendizaje el cálculo de pesos y sesgos de manera que la red se comporte de una manera deseada. De acuerdo al tipo de aprendizaje, las redes se pueden subdividir en dos grandes grupos:

- Redes con aprendizaje supervisado. Estas redes se entrenan presentando para cada combinación de entradas, las salidas que se espera ellas produzcan. Los algoritmos de aprendizaje calculan pesos y sesgos nuevos a manera de minimizar el error entre la salida deseada y la obtenida realmente.

- Redes no supervisadas. Los algoritmos de aprendizaje calculan nuevos pesos libremente. Estas redes se utilizan como clasificadores pues se caracterizan por asociar una combinación de entradas específica con una sola salida.

2.3.1. Algoritmo de aprendizaje back propagation

El algoritmo de aprendizaje back propagation (BP) se utiliza para ajustar los pesos y sesgos de una red con el fin de minimizar la suma del cuadrado de los errores de la red (ver figura 2.7).

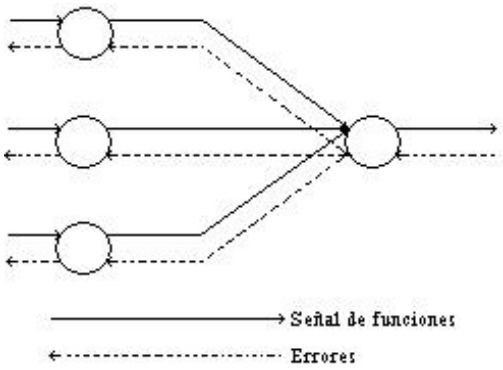

Figura 2.7: Ilustración de las direciones de dos señales de flujo en un perceptrón multicapa.

El algoritmo BP es un método iterativo de optimización de descenso según el gradiente, cuyos detalles se presentan a continuación.

Para una neurona j en una capa oculta o en la salida, la señal de salida es:

$$o_j = f\left(w_{ij} o_i - b_j\right) \tag{2.6}$$

donde f es la función de activación de la neurona

w_{ij} son los pesos de las conexiones entre la neurona considerada, j, y la neurona i, perteneciente a la capa precedente.

o_i es la salida de la neurona i de la capa precedente

b_j es el sesgo de la neurona j

En este caso, se considera funciones de activación sigmoide logarítmicas.

Además, se define:

$$net = \sum w_{ij} o_i - b_j \tag{2.7}$$

La salida de la neurona j, entonces, está dada por:

$$o_j = f\left(net_j\right) = \frac{1}{1 + e^{-net_j}} \tag{2.8}$$

Para el aprendizaje, el valor $-bj$ se considera como un peso correspondiente a la conexión de la neurona j con una supuesta neurona de la capa precedente cuya salida es constante e igual a uno.

El algoritmo de BP permite ajustar los pesos de la red neuronal con el fin de minimizar el error cuadrático sobre un conjunto de entradas y salidas asociadas (patrones) que la red debe ser capaz de aprender para luego realizar generalizaciones a partir de ellas.

Además, se define como superficie de error a la función multivariable generada por la expresión del error de ajuste en términos de los pesos y sesgos de las neuronas de la red.

El algoritmo BP permite determinar los valores de los pesos para los cuales la función de error es mínima. Esto no siempre se logra, convergiendo muchas veces el algoritmo a mínimos locales, no al mínimo global buscado, o simplemente no convergiendo.

Se considera una red con M neuronas en la capa de salida y suponiendo que se dispone de un conjunto de aprendizaje con P patrones, uno de los cuales, denominado p, tiene salidas dadas por:

$$t_p = [t_{p1}, t_{p2}, ..., t_{pM}] \tag{2.9}$$

el error cuadrático tiene, para ese patrón, la siguiente expresión

$$E_p = \frac{1}{2} \sum \left(t_{pi} - o_{pi}\right)^2 \tag{2.10}$$

que corresponde al error tomado para derivar la regla de optimización.

Los valores t_{pi} representan las salidas deseadas ante las entradas correspondientes al patrón p. Cuando dicho patrón es presentado a la red, los pesos se modifican según una regla iterativa derivada del método de optimización según el gradiente, con lo cual el peso w_{ij} según la ecuación es:

$$w_{ij}\left(h\right) = w_{ij}\left(h-1\right) + \Delta w_{ij}\left(h\right) \tag{2.11}$$

donde h corresponde al contador dentro de una iteración.

El valor de $\Delta w_{ij}\left(h\right)$ se calcula como:

$$\Delta w_{ij}\left(h\right) = \eta \left(-\frac{\partial E_p}{\partial w_{ij}}\right) = \eta \left(-\frac{\partial E_p}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}}\right) \tag{2.12}$$

donde η es la tasa de aprendizaje (constante de proporcionalidad) $(0 < \eta < 1)$

En general, los pesos se inicializan entre cero y uno aleatoriamente.

Se define el parámetro δ_j como:

$$\delta_j = -\frac{\partial E_p}{\partial net_j} = -\frac{\partial E_p}{\partial o_j}\frac{\partial o_j}{\partial net_j} \tag{2.13}$$

En las expresión siguientes, el subíndice p se ha omitido por simplicidad.

Para calcular las derivadas es necesario tener en cuenta que la función de activación escogida es una sigmoide logarítmica, cuya derivada es:

$$\frac{df\left(x\right)}{dx} = \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \frac{1}{1+e^{-x}}\left(1-\frac{1}{1+e^{-x}}\right) = f\left(x\right)\left(11-f\left(x\right)\right) \tag{2.14}$$

Para una neurona j en la capa de salida se tiene entonces,

$$\delta_j = \left(t_j - o_j\right) o_j \left(1 - o_j\right) \tag{2.15}$$

Para una neurona en la capa oculta o en la capa de entrada, se tiene:

$$\delta_j = o_j \left(1 - o_j\right) \sum \left(\delta_k w_{jk}\right) \tag{2.16}$$

donde el contador k cubre las neuronas de la capa posterior a la j.

Entonces, la corrección de los pesos se comienza por la capa de salida y se propaga hacia atrás hasta llegar a la capa de entrada.

Con esto, el término (2.12) se puede expresar como:

$$\Delta w_{ij} = \eta \delta_j o_i \tag{2.17}$$

Ahora bien, normalmente no se emplea sólo esta expresión sino que se agrega un término denominado momentum, que corresponde al cambio anterior en el peso ponderado por el coeficiente de momentum. Entonces, se tiene:

$$\Delta w_{ij} = \eta \delta_j o_i + \alpha \Delta w_{ij}\left(h-1\right) \tag{2.18}$$

donde α es el coeficiente de momento. Este término permite suavizar la convergencia del método y ayuda a que la convergencia de los pesos no se vea demasiado afectada por irregularidades en la superficie de error.

Considerando los P patrones de que se dispone y con los cuales se realizará el aprendizaje, la expresión para el error total, o error de ajuste, es la siguiente:

$$E = \sum E_p = \sum \left(\frac{1}{2} \sum \left(t_{pi} - o_{pi} \right)^2 \right) \tag{2.19}$$

En general, el aprendizaje se considera acabado cuando el valor de E es menor o igual que un límite preestablecido.

En resumen, la actualización de pesos secuencial es el método preferido para la implantación en-línea. Para este modo de operar, el algoritmo a seguir es:

1. *Inicialización.* Asumiendo que no se dispone de ninguna información a priori se escogen los pesos sinápticos y los límites de una distribución uniforme la cual su media es cero y la varianza se escoge de tal forma que la desviación estandar de los campos locales inducidos de las neuronas esté en la transición entre la parte lineal y la parte saturada de la función de activación sigmoide.

2. *Presentación de los ejemplos de aprendizaje.* Presentar la red con una época de los ejemplos de aprendizaje. Para cada ejemplo en el conjunto, ordenado de alguna manera, desarrollar la secuencia de computación hacia delante y hacia atras descritos en los puntos 3 y 4.

3. *Cómputo hacia delante.* Denotemos una muestra de aprendizaje en la época como $(x(k), d(k))$, con el vector de entrada $x(k)$ aplicado a la capa de entrada de nodos sensores y el vector de respuesta deseado $d(k)$ presentado a la capa de salida de nodos computacionales. Computar los campos locales inducidos y las señales de funciones de la red procediendo hacia delante en la red, capa por capa. El campo local inducido $v_j^{(l)}(k)$ para la neurona j en la capa l es

$$v_j^{(l)}(k) = \sum_{i=0}^{m_o} w_{ji}^{(l)}(k)\, y_i^{(l-1)}(k)$$

donde $y_i^{(l-1)}(k)$ es la señal de salida de la neurona i en la capa anterior $l-1$ en la iteración k y $w_{ji}^{(l)}(k)$ es el peso sináptico de la neurona j en la capa l que es alimentada de la neurona i en la capa $l-1$. Para $i = 0$ tenemos $y_0^{(l-1)}(k) = +1$ y $w_{j0}^{(l)}(k) = b_j^{(l)}(k)$ es aplicado a la neurona j en la capa l. Asumiendo el uso de una función sigmoide, la señal de salida de la neurona j en la capa l es

$$y_j^{(l)} = \varphi_j\left(v_j(k)\right)$$

Si la neurona j se encuentra en la primer capa oculta, usar

$$y_j^{(0)}(k) = x_j(k)$$

donde $x_j(k)$ es el $j-$ésimo elemnto del vector de entrada $\mathbf{x}(k)$. Si la neurona j está en la capa de salida, usar

$$y_j^{(L)} = o_j(k)$$

Computar la señal de error

$$e_j(k) = d_j(k) - o_j(k)$$

donde $d_j(k)$ es el j-ésimo elemento del vector de respuesta deseado $\mathbf{d}(k)$.

4. *Cómputo hacia atrás.* Computar los δs (gradientes locales) de la red definidos como

$$\delta_j^{(l)}(k) = \begin{cases} e_j^{(L)}(k)\,\varphi_j'\left(v_j^{(L)}(k)\right) & \text{para la neurona } j \text{ en la capa de salida } L \\ \varphi_j'\left(v_j^{(L)}(k)\right)\sum_m \delta_m^{(l+1)}(k)\,w_{kj}^{(l+1)}(k) & \text{para la neurona } j \text{ en la capa oculta } l \end{cases}$$

donde el apòstrofe en φ_j' denota diferenciación con respecto al argumento. Ajustar los pesos sinápticos de la red en la capa l de acuerdo a la regla delta generalizada:

$$w_{ji}^{(l)}(k+1) = w_{ji}^{(l)}(k) + \alpha\left[w_{ji}^{(l)}(k-1)\right] + \eta\delta_j^{(l)}(k)\,y_i^{(l-1)}(k)$$

donde η es el parámetro de la velocidad de aprendizaje y α es la constante momentum.

5. *Iteración.* Iterar los computos hacia delante y hacia atrás de los puntos 3 y 4 presentando nuevas épocas de ejemplos de aprendizaje a la red hasta que se consiga el criterio de paro.

Capítulo 3

Sistema difuso neuronales

3.1. Fundamentos de lógica difusa

La lógica difusa asocia incertidumbre a la estructura de un conjunto de datos (Zadeh, 1965). Los elementos de un conjunto difuso son pares ordenados que indican el valor del elemento y su grado de pertenencia.

Para un conjunto difuso $A = \{(x, \mu_A(x)) / x \in X\}$, se tiene que el elemento x pertenece al conjunto A con un grado de pertenencia $\mu_A(x)$, que puede variar entre 0 y 1. Por lo tanto, una variable puede ser caracterizada por diferentes valores lingüísticos, cada uno de los cuales representa un conjunto difuso.

Operaciones básicas de lógica difusa

Dados dos conjuntos difusos A y B en el mismo universo X, con funciones de pertenencia μ_A y μ_B respectivamente, se pueden definir las siguientes operaciones básicas:

Unión. La función de pertenencia de la unión de A y B se define como:

$$\mu_{A \cup B} = \text{máx} \{(\mu_A(x), \mu_B(x))\} \tag{3.1}$$

Intersección. La función de pertenencia de la intersección de A y B es:

$$\mu_{A \cap B} = \text{mín} \{(\mu_A(x), \mu_B(x))\} \tag{3.2}$$

Complemento. La función de pertenencia del complemento de A se define como:

$$\mu_A - x = 1 - \mu_A(x) \tag{3.3}$$

Producto cartesiano. Dados los conjuntos difusos $A_1, ..., A_n$ con universos $X_1, ..., X_n$ respectivamente, se define el producto cartesiano como un conjunto difuso en $X_1 \times ... \times X_n$ con la siguiente función de pertenencia:

$$\mu_{A_1 \times ... \times A_n}(x_1, ..., x_n) = \text{mín}\left\{\mu_{A_1}(x_1), ..., \mu_{A_n}(x_n)\right\} \tag{3.4}$$

según Mamdani (1974) y

$$\mu_{A_1 \times ... \times A_n}(x_1, ..., x_n) = \mu_{A_1}(x_1) \cdot ... \cdot \mu_{A_1}(x_n) \tag{3.5}$$

según Larsen (1980).

3.2. Variables lingüísticas y reglas difusas SI-ENTONCES

3.2.1. De variables numéricas a variables lingüísticas

En nuestra vida diaria las palabras se usan a menudo para describir variables, por ejemplo: "Hoy hace calor" o el equivalente " La temperatura del día de hoy es alta", usamos la palabra "alta" para describir la variable "La temperatura de hoy", esto es, la variable "La temperatura de hoy" toma la palabra "alta" como su valor. Claramente, la variable "La temperatura de hoy" también puede tomar los números $25°C$, $19°C$, etc. como su valor. Cuando una variable toma un número como su valor tenemos un marco matemático de trabajo bien establecido para formularlo, pero cuando una variable toma una palabra como su valor no tenemos un marco de trabajo formal para formularlo en teoría de matemática clásica. Para proveer dicho marco de trabajo se introdujo el concepto de variable lingüística, hablando normalmente, si una variable puede tomar palabras en lenguaje natural como su valor se le llama *variable lingüística*. Ahora la pregunta es ¿Como formular las palabras en un lenguaje matemático? Aquí usamos conjuntos difusos para caracterizar las palabras, así tenemos la siguiente definición.

Definición 3.1 *Si una variable puede tomar palabras en lenguaje natural como su valor, a esa variable se le llama variable lingüística, donde las palabras son caracterizadas por conjuntos difusos*

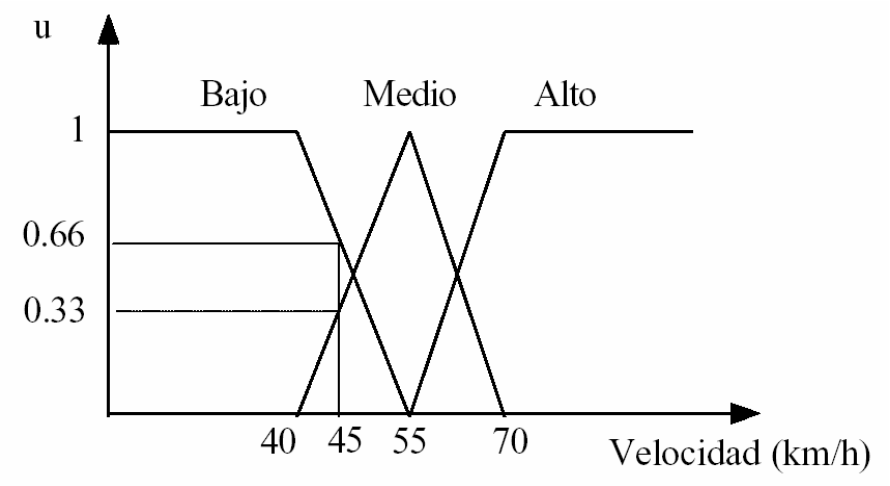

Figura 3.1: Velocidad de un automóvil como una variable lingüística

definidos en el universo en el cual la variable es definida.

Ejemplo 3.1 *La velocidad de un automóvil es una variable x que toma valores en el intervalo $[0, V_{\text{máx}}]$ donde $V_{\text{máx}}$ es la velocidad máximo del automóvil. Ahora definimos tres conjuntos difusos "Bajo", "Medio" y "Alto" en , $[0, V_{\text{máx}}]$ como se muestra en la figura 3.1. Si vemos a x como una variable lingüística entonces puede tomar a "Bajo", "Medio" y "Alto" como sus valores. Esto es, podemos decir "x es bajo", "x es medio" o "x es alto". Por supuesto x también puede tomar números en el intervalo $[0, V_{\text{máx}}]$ como su valor, por ejemplo, $x = 50km/hr$, $x = 35km/hr$, etc.*

La definición anterior es una definición simple e intuitiva para variables lingüísticas. En la literatura de teoría difusa se usa una definición más formal de variable lingüística, esta es,

Definición 3.2 *Una variable lingüística es caracterizada por (X, T, U, M), donde:*

- *X es el nombre de la variable lingüística; en el ejemplo anterior, X es la velocidad del automóvil.*
- *T es el conjunto de valores lingüísticos; en el ejemplo anterior, $T = \{Bajo, Medio, Alto\}$.*

- *U es el dominio físico actual en el cual la variable lingüística X toma sus valores cuantitativos; en el ejemplo anterior, $U = [0, V_{\text{máx}}]$*
- *M es la regla semántica que relaciona cada valor lingüístico en T con un conjunto difuso en U; en el ejemplo anterior, M relaciona "Bajo", "Medio" y "Alto" con las funciones de membresía mostradas en la figura anterior.*

Comparando las dos definiciones anteriores, nos damos cuenta de que son esencialmente equivalentes. La primera es más intuitiva y la segunda más formal. De estas definiciones podemos ver que las variables lingüísticas son extensiones de variables numéricas en el sentido de que se les permite tomar conjuntos difusos como sus valores.

3.2.2. Hedges lingüísticos

Con el concepto de variable lingüística podemos tomar palabras como valores (lingüísticos) de variables. En nuestra vida diaria a veces usamos más de una palabra para describir una variable. Por ejemplo, si vemos la velocidad de un automóvil como una variable lingüística puede ser "no lento", "un poco rápido", "muy rápido", etc. En general, el valor de una variable lingüística es un término compuesto, $x = x_1, x_2, ..., x_n$ que es una concatenación de términos atómicos $x_1, x_2, ..., x_n$. Estos *términos atómicos* pueden clasificarse en tres grupos.

- Términos primarios, los cuales son etiquetas de conjuntos difusos; como serían "Bajo", "Medio" y "Alto".
- Complemento "no" y conexiones "y" y "o".
- Hedges tales como "muy", "apenas" "más o menos", etc.

Aunque en su uso diario la palabra hedge no tiene un significado definido en esencia actúa como un intensificador. Así damos la siguiente definición para los dos hedges más comunes, *muy* y *más o menos*.

Definición 3.3 *Sea A un conjunto difuso en U, entonces muy A es definido como un conjunto difuso en U con la función de membresía*

$$\mu_{muy\ A}(x) = [\mu_A(x)]^2 \tag{3.6}$$

y más o menos A es un conjunto difuso en U con la función de membresía

$$\mu_{mas\ o\ menos\ A}(x) = [\mu_A(x)]^{1/2} \tag{3.7}$$

3.2.3. Reglas difusas SI-ENTONCES

Una regla difusa SI-ENTONCES es una expresión condicional expresada como:

$$\text{SI } \langle proposición\ difusa \rangle \text{ ENTONCES } \langle proposición\ difusa \rangle \tag{3.8}$$

De ahí que para entender que es una regla difusa primero debemos saber que es una proposición difusa.

3.2.4. Proposiciones difusas

Hay dos tipos de proposiciones difusas, *proposiciones difusas atómicas* y *proposiciones difusas compuestas*. Una *proposiciones difusa atómica* es un expresión singular

$$x \text{ es } A \tag{3.9}$$

donde x es una variable lingüística y A es un valor lingüístico de x (estos es, A es un conjunto difuso definido en el dominio físico de x). Una *proposición difusa compuesta* es una composición de proposiciones difusas atómicas usando los conectivos "y", "o" y "no" los cuales representan intersección difusa, unión difusa y complemento difuso respectivamente.

Nótese que en una proposición difusa compuesta las proposiciones difusas atómicas son independientes, esto es, cada proposición difusa atómica puede usar una variable x distinta. Actualmente, las variables lingüísticas en una proposición difusa compuesta son en general distintas.

Las proposiciones difusas compuestas deberían ser entendidas como *relaciones difusas*.

Definición 3.4 *Una relación difusa es un conjunto difuso definido en el producto cartesiano de conjuntos normales U_1,U_2, ..., U_n. Con el esquema representativo*

$$A = \{(x, \mu_A(x)) \mid x \in U\}$$

una relación difusa Q en $U_1 \times U_2 \times ... \times U_n$ es definida como el conjunto difuso

$$Q = \left\{ \left((u_1, u_2, ..., u_n), \mu_Q (u_1, u_2, ..., u_n) \right) \mid (u_1, u_2, ..., u_n) \in U_1 \times U_2 \times ... \times U_n \right\}$$

donde $\mu_Q : U_1 \times U_2 \times ... \times U_n \rightarrow [0, 1]$.

¿Cómo determinar las funciones de membresía de estas relaciones difusas?

- *Para conectivos "y" use intersecciones difusas.* Específicamente, sean x y y variables lingüísticas en los dominios físicos U y V, A y B conjuntos difusos en U y V respectivamente, entonces la proposición difusa compuesta

$$x \text{ es } A \ y \ y \text{ es } B \tag{3.10}$$

  es interpretado como la relación difusa $A \cap B$ en $U \times V$ con función de membresía

$$\mu_{A \cap B} (x, y) = t \left[\mu_A (x), \mu_B (y) \right] \tag{3.11}$$

  donde $t : [0, 1] \times [0, 1] \rightarrow [0, 1]$ es cualquier $norma - t$.

- *Para conectivos "o" use uniones difusas.* Específicamente la proposición compuesta

$$x \text{ es } A \ o \ y \text{ es } B \tag{3.12}$$

  es interpretada como la relación difusa $A \cup B$ en $U \times V$ con función de membresía

$$\mu_{A \cup B} (x, y) = s \left[\mu_A (x), \mu_B (y) \right] \tag{3.13}$$

  donde $s : [0, 1] \times [0, 1] \rightarrow [0, 1]$ es cualquier $norma - s$.

- *Para conectivos "no" use complementos difusos.* Estos es, reemplace *no* A por $\overline{A}$, el cual es definido de acuerdo al operador complemento.

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Figura 3.2: Tabla de verdad

3.2.5. Interpretación de las reglas difusas SI-ENTONCES

Como las proposiciones difusas son interpretadas como relaciones difusas la pregunta clave es ¿Cómo interpretar la operación SI-ENTONCES? En cálculo proposicional clásico, Si p ENTONCES q se escribe como $p \rightarrow q$ con la implicación $\rightarrow$ descrita como en la tabla de la figura 3.2, donde p y q son variables proposicionales cuyos valores son verdaderos (V) o falsos (F). Entonces $p \rightarrow q$ es equivalente a:

$$\overline{p} \vee q \tag{3.14}$$

y

$$(p \wedge q) \vee \overline{p} \tag{3.15}$$

en el sentido de que comparten la misma tabla de verdad $p \rightarrow q$, donde, $\overline{*}$, $\vee$ y $\wedge$ representan operaciones lógicas (clásicas) "no", "o" y "y", respectivamente.

Como las reglas difusas SI-ENTONCES pueden ser vistas como reemplazando p y q con proposiciones difusas, podemos *interpretar* las reglas difusas SI-ENTONCES reemplazando los operadores $\overline{*}$, $\vee$ y $\wedge$ con complemento difuso, unión difusa e intersección difusa respectivamente. Como hay una gran variedad de complementos difusos, uniones difusas e intersecciones difusas en la literatura se proponen un número distinto de reglas difusas SI-ENTONCES. A continuación se listan algunas.

- **Implicación Dienes-Rescher**: Si reemplazamos los operadores lógicos $\overline{*}$ y $\vee$ de la ecuación por el complemento difuso básico y por la unión difusa básica respectivamente, obtenemos la llamada *implicación Dienes-Rescher*. Específicamente, la regla difusa SI-ENTONCES SI $\langle FP_1 \rangle$ ENTONCES $\langle FP_2 \rangle$ se interpreta como una relación difusa Q_D en $U \times V$ con función de membresía

$$\mu_{Q_D}(x,y) = \text{máx}\left[1 - \mu_{FP_1}(x), \mu_{FP_2}(y)\right] \tag{3.16}$$

- **Implicación Lukasiewicz**: Si usamos la norma-s Yager con $w = 1$ para el operador $\vee$ y complemento difuso básico para el operador $\overline{*}$ obtenemos la *Implicación Lukasiewicz*. Específicamente, la regla difusa SI-ENTONCES SI $\langle FP_1 \rangle$ ENTONCES $\langle FP_2 \rangle$ se interpreta como una relación difusa Q_L en $U \times V$ con función de membresía

$$\mu_{Q_L}(x,y) = \text{mín}\left[1, 1 - \mu_{FP_1}(x) + \mu_{FP_2}(y)\right] \tag{3.17}$$

- **Implicación Zadeh**: Aquí la regla difusa SI-ENTONCES SI $\langle FP_1 \rangle$ ENTONCES $\langle FP_2 \rangle$ se interpreta como una relación difusa Q_Z en $U \times V$ con función de membresía

$$\mu_{Q_Z}(x,y) = \text{máx}\left[\text{mín}\left(\mu_{FP_1}(x), \mu_{FP_2}(y)\right), 1 - \mu_{FP_1}(x)\right] \tag{3.18}$$

  Claramente, la ecuación se obtiene usando complemento difuso, unión difusa básica e intersección difusa básica para $\overline{*}$, $\vee$ y $\wedge$ respectivamente.

- **Implicación Gödel**: La implicación Gödel es una implicación bien conocida en lógica clásica. Por generalización a proposiciones difusas obtenemos lo siguiente: La regla difusa SI-ENTONCES SI $\langle FP_1 \rangle$ ENTONCES $\langle FP_2 \rangle$ se interpreta como una relación difusa Q_G en $U \times V$ con función de membresía

$$\mu_{Q_G}(x,y) = \begin{cases} 1 & \text{si } \mu_{FP_1}(x) \leq \mu_{FP_2} \\ \mu_{FP_2}(y) & \text{si } \mu_{FP_1}(x) > \mu_{FP_2} \end{cases} \tag{3.19}$$

Es interesante explorar la relación entre estas implicaciones. El siguiente lema muestra que la implicación Zadeh es menor que la implicación Dienes-Rescher, la cual es menor que la implicación Lukasiewicz.

Lemma 3.1 *Para todo* $(x, y) \in U \times V$ *lo siguiente es verdad*

$$\mu_{Q_Z}(x, y) \leq \mu_{Q_D}(x, y) \leq \mu_{Q_L}(x, y) \tag{3.20}$$

Conceptualmente, podemos reemplazar los operadores $\overline{*}$, $\vee$ y $\wedge$ en (3.14) y (3.15) por cualquier complemento difuso, norma-s y norma-t respectivamente para obtener una interpretación particular.

Cuando p y q son proposiciones normales, $p \rightarrow q$ es una implicación *global* en el sentido de que la tabla de la figura 3.2 cubre todos los casos posibles. Sin embargo, cuando p y q son proposiciones difusas, $p \rightarrow q$ solamente puede ser una implicación *local* en el sentido de que $p \rightarrow q$ tiene valores de verdad amplios solo cuando ambos p y q tienen valores de verdad amplios. Por ejemplo, cuando decimos, "SI la velocidad es alta ENTONCES la resistencia es alta" esto es solo una proposición local en el sentido de que esta regla no nos dice nada acerca de la situación cuando "La velocidad es baja" o "La velocidad es media". De aquí que la regla difusa SI-ENTONCES

$$\text{SI } \langle FP_1 \rangle \text{ ENTONCES } \langle FP_2 \rangle \tag{3.21}$$

debería interpretarse como

$$\text{SI } \langle FP_1 \rangle \text{ ENTONCES } \langle FP_2 \rangle \text{ DE LO CONTRARIO } \langle NADA \rangle \tag{3.22}$$

donde $NADA$ significa que esta regla no existe. En términos lógicos

$$p \rightarrow q = p \wedge q \tag{3.23}$$

Usando *min* o *producto algebraico* para $\wedge$ en (3.23), obtenemos la *implicación Mamdani.*

- **Implicación Mamdani**: La regla difusa SI-ENTONCES se interpreta como una regla difusa Q_{MM} o Q_{MP} en $U \times V$ con función de membresía

$$Q_{MM}(x, y) = \text{mín}\left[\mu_{FP_1}(x), \mu_{FP_2}(y)\right] \tag{3.24}$$

o

$$Q_{MP}(x, y) = \mu_{FP_1}(x) \cdot \mu_{FP_2}(y) \tag{3.25}$$

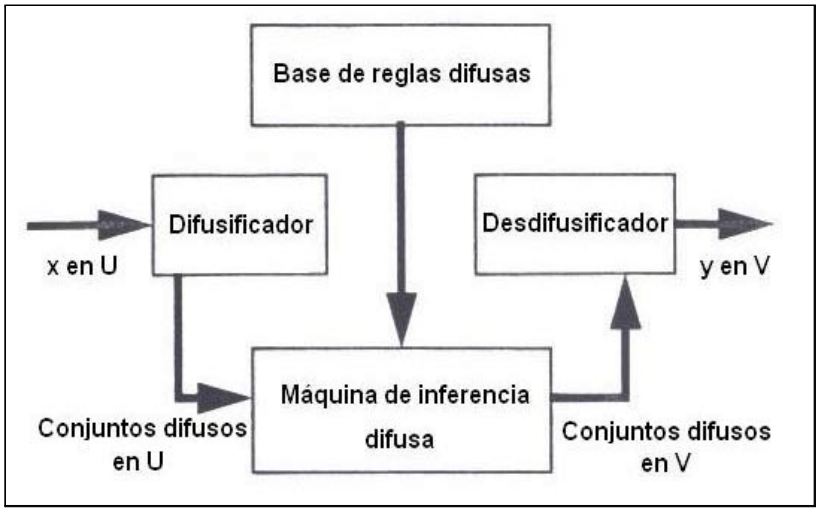

Figura 3.3: Estructura de los sistemas difusos

Las implicaciones Mamdani son las implicaciones más ampliamente usadas en sistemas difusos y control difuso. Se soportan por el argumento de que las reglas SI-ENTONCES son locales. Sin embargo, uno puede no estar de acuerdo con este argumento. Por ejemplo, uno puede discutir que cuando decimos "SI la velocidad es alta ENTONCES la resistencia es alta" nosotros *implícitamente* decimos que "SI la velocidad es lenta ENTONCES la resistencia es baja". En este sentido las reglas difusas SI-ENTONCES son no locales. Este tipo de debate indica que cuando representamos conocimiento humano en términos de reglas difusas SI-ENTONCES, diferentes personas tienen diferentes interpretaciones. Consecuentemente, se necesitan diferentes implicaciones para lidiar con la diversidad de interpretaciones. Por ejemplo, si los expertos humanos piensan que sus reglas son locales entonces deben usarse las implicaciones Mamdani; de lo contrario deben considerarse las implicaciones globales.

3.3. Estructura de los sistemas difusos

A continuación se explican las partes que forman un sistema difuso como se muestra en la figura 3.3

3.3.1. Fusificadores

Un fusificador es definido como un mapeo de un punto con valor-real $\mathbf{x}^* \in U \subset R^n$ a un conjunto difuso A' en U. El fusificador debe considerar el hecho de que la entrada está en el punto no-difuso $\mathbf{x}^*$, si la entrada al sistema difuso es corrompido por ruido es deseable que el fusificador ayude a suprimir dicho ruido y finalmente el fusificador debería ayudar a simplificar el cómputo envuelto en la máquina de inferencia difusa, el computo más complicado en la máquina de inferencia difusa es $\sup_{\mathbf{x}\in\mathbf{U}}$ por lo que nuestro objetivo es simplificar el computo envuelto en $\sup_{\mathbf{x}\in\mathbf{U}}$.

- Fusificador **singleton**: El difusificador singleton mapea un punto con valor-real $\mathbf{x}^* \in U$ a un singleton difuso A' en U el cual tiene valor de membresía $\mathbf{1}$ en $\mathbf{x}^*$ y 0 en todos los otros puntos en U, esto es,

$$\mu_{A'}(\mathbf{x}) = \begin{cases} 1 \text{ si } \mathbf{x} = \mathbf{x}^* \\ 0 \text{ si } \mathbf{x} \neq \mathbf{x}^* \end{cases} \tag{3.26}$$

- Fusificador **Gaussiano**: El fusificador Gaussiano mapea $\mathbf{x}^* \in U$ a un conjunto difuso A' en U el cual tiene la siguiente función de membresía Gaussiana:

$$\mu_{A'}(\mathbf{x}) = e^{-\left(\frac{x_1 - x_1^*}{a_1}\right)^2} \star ... \star e^{-\left(\frac{x_n - x_n^*}{a_n}\right)^2} \tag{3.27}$$

  donde a_i son parámetros positivos y la norma-t $\star$ es usualmente escogida como producto algebraico o mín.

- Fusificador **triangular**: El fusificador triangular mapea $\mathbf{x}^* \in U$ a un conjunto difuso A' en U el cual tiene la siguiente función de membresía triangular:

$$\mu_{A'}(\mathbf{x}) = \begin{cases} \left(1 - \frac{|x_1 - x_1^*|}{b_1}\right) \star ... \star \left(1 - \frac{|x_n - x_n^*|}{b_n}\right) & \text{si } |x_i - x_i^*| \leq b_i,\ i = 1, 2, ..., n \\ 0 & \text{si } |x_i - x_i^*| > b_i,\ i = 1, 2, ..., n \end{cases} \tag{3.28}$$

  donde b_i son parámetros positivos y la norma-t $\star$ es usualmente escogida como producto algebraico o mín.

3.3.2. Defusificadores

El defusificador está definido como un mapeo de conjunto difuso B' en $V \subset R$ (que es la salida de la máquina de inferencia difusa) a un punto no-difuso $y^* \in V$. La tarea del defusificador es especificar un punto en V que mejor represente al conjunto difuso en B'. Esto es similar al valor promedio de una variable al azar, sin embargo, como B' es construido de una forma especial tenemos un número de opciones para representar este punto de representación. Los criterios a considerar al escoger un defusificador son:

Plausibilidad: El punto y^* debería representar B' desde un punto de vista intuitivo; por ejemplo, puede estar aproximadamente en medio del soporte de B' o tener un alto grado de membresía en B'.

Simplicidad computacional: Este criterio es particularmente importante para control difuso porque los controladores difusos operan en tiempo real.

Continuidad: Un pequeño cambio en B' no debería resultar en un gran cambio en y^*.

- El *defusificador de centro de gravedad* especifica a y^* como el centro del área cubierta como la función de membresía de B', esto es

$$y^* = \frac{\int_V y \mu_{B'}(y)\, dy}{\int_V \mu_{B'}(y)\, dy} \tag{3.29}$$

  donde $\int_V$ es la integral convencional. La figura 3.4 muestra esta operación gráficamente.

- Como el conjunto difuso B' es la unión o intersección de M conjuntos difusos, una buena aproximación de (3.29) es el promedio de los pesos de los centros de los M conjuntos difusos, con los pesos igual a las alturas de los correspondientes conjuntos difusos. Específicamente, sea y^{-1} el centro del l-ésimo conjunto difuso y sea w_l su peso, el *defusificador de centro promedio* determina y^* como

$$y^* = \frac{\sum_{l=1}^{M} y^{-1} w_l}{\sum_{l=1}^{M} w_l} \tag{3.30}$$

  La figura 3.5 ilustra esta operación gráficamente para un simple ejemplo con $M = 2$.

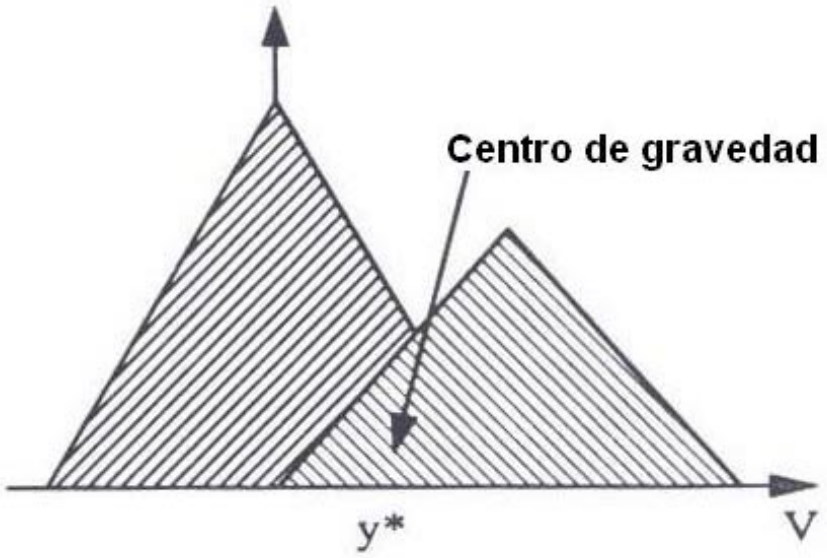

Figura 3.4: Difusificador de centro de gravedad

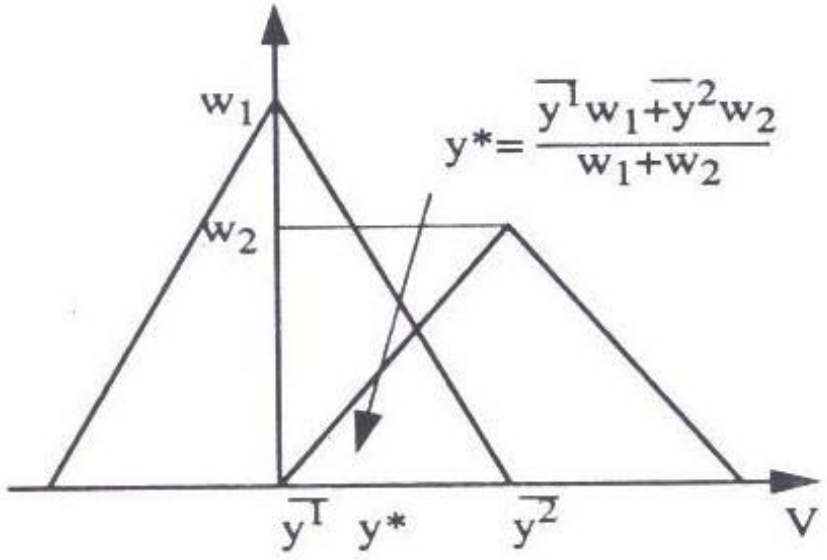

Figura 3.5: Difusificador de centro promedio

- Conceptualmente, el *defusificador máximum* escoge a y^* como el punto en V en el cual $\mu_{B'}(y)$ alcanza su valor máximo, definiendo el conjunto

$$hgt\left(B'\right) = \left\{ y \in V \mid \mu_{B'}(y) = \sup_{y \in V} \mu_{B'}(y) \right\} \tag{3.31}$$

esto es, $hgt(B')$ es el conjunto de todos los puntos en V en los cuales $\mu_{B'}(y)$ alcanza su máximo valor. El *difusificador máximum* define a y^* como un elemento arbitrario en $hgt(B')$, esto es

$$y^* = \text{cualquier punto en } hgt\left(B'\right) \tag{3.32}$$

Si $hgt(B')$ contiene solo un punto entonces y^* es únicamente definido. Si $hgt(B')$ contiene más de un punto entonces todavía podemos usar (3.32) o usar el ínfimo de los máximos, el supremo de los máximos o el promedio de los máximos. Específicamente, el *ínfimo de los defusificadores máximos* da

$$y^* = \inf\left\{ y \in hgt\left(B'\right) \right\} \tag{3.33}$$

el *supremo de los defusificadores máximos* da

$$y^* = \sup\left\{ y \in hgt\left(B'\right) \right\} \tag{3.34}$$

y el *promedio de los defusificadores máximos* da

$$y^* = \frac{\int_{hgt(B')} y dy}{\int_{hgt(B')} dy} \tag{3.35}$$

donde $\int_{hgt(B')}$ es la integración usual para la parte continua de $hgt(B')$ y su sumatoria para su parte discreta.

3.3.3. Regla composicional de inferencia

La regla composicional de inferencia es una generalización del siguiente procedimiento (referirse a la figura 3.7): suponiendo que tenemos una curva $y = f(x)$ de $x \in U$ a $y \in V$ y dados $x = a$, entonces de $x = a$ y $y = f(x)$ podemos inferir que $y = b = f(a)$.

Generalizando el procedimiento anterior al asumir que a es un intervalo y $f(x)$ es una función valor-intervalo como se muestra en la figura 3.8. Para encontrar el intervalo b el cual es inferido de

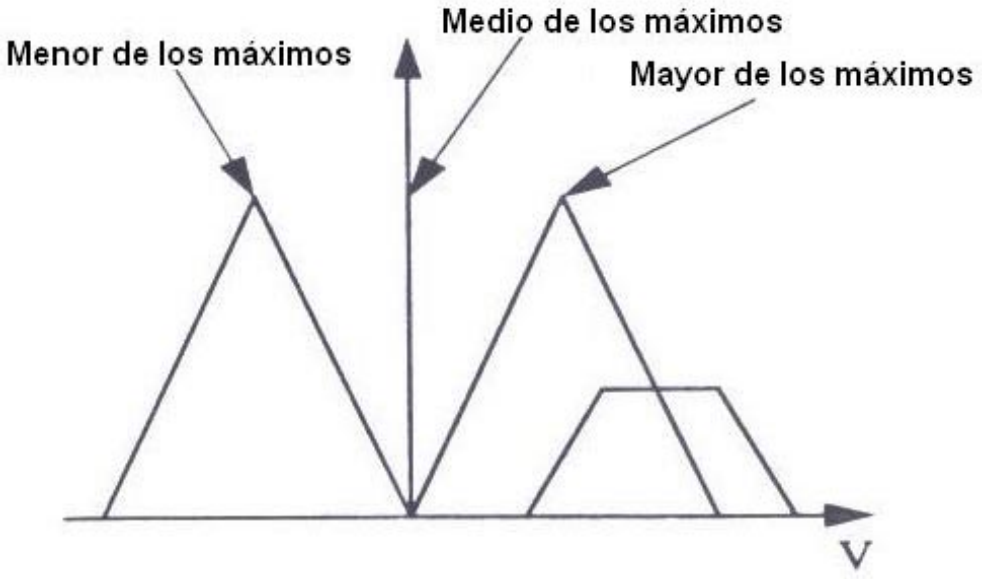

Figura 3.6: Difusificador máximum

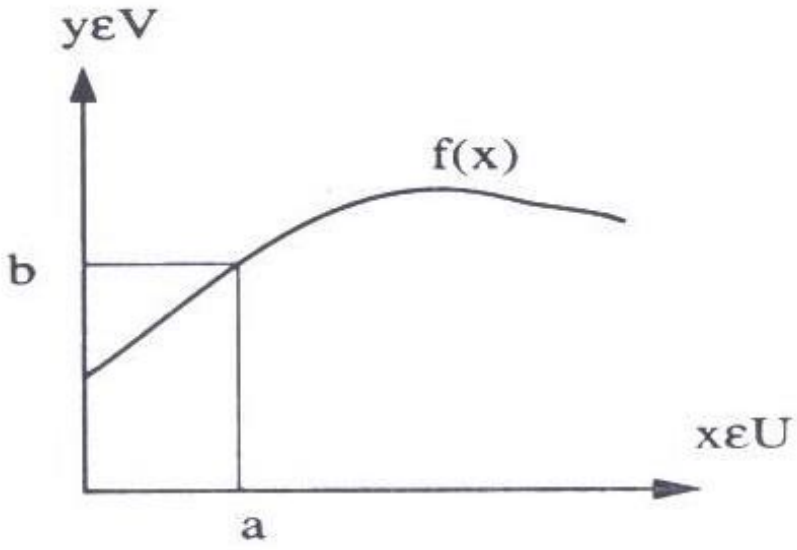

Figura 3.7: Infiriendo y=b de x=a y y=f(x)

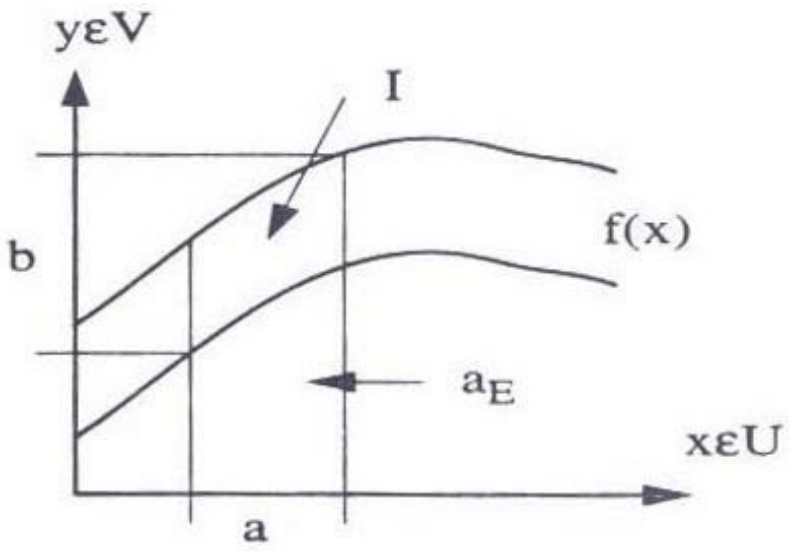

Figura 3.8: Infiriendo el intervalo b del intervalo a y la función valor-intervalo f(x)

a y $f(x)$ primero construimos un conjunto cilíndrico a_E con base a y encontramos su intersección I con la curva valor-intervalo. Entonces proyectamos I en V produciendo el intervalo b.

Llendo un paso adelante en nuestra cadena de generalización asumiendo que A' es un conjunto difuso en U y Q es una relación difusa en $U \times V$. Formando de nuevo una extensión cilíndrica A'_E de A' e interceptando con la relación difusa Q (ver figura 3.9) obtenemos un conjunto difuso $A'_E \cap Q$ el cual es análogo a la intersección I en la figura 3.8. Entonces, proyectando $A'_E \cap Q$ en el eje-y obtenemos el conjunto difuso B'.

Específicamente, dados $\mu_{A'}$ y $\mu_Q(x,y)$ tenemos

$$\mu_{A'_E}(x,y) = \mu_{A'}(x) \tag{3.36}$$

y consecuentemente

$$\begin{aligned} \mu_{A'_E \cap Q}(x,y) &= t\left[\mu_{A'_E}(x,y), \mu_Q(x,y)\right] \\ &= t\left[\mu_{A'}(x), \mu_Q(x,y)\right] \end{aligned} \tag{3.37}$$

Finalmente obtenemos B', la proyección de $A'_E \cap Q$ en V, como

$$\mu_{B'}(y) = \sup_{x \in U} t\left[\mu_{A'}(x), \mu_Q(x,y)\right] \tag{3.38}$$

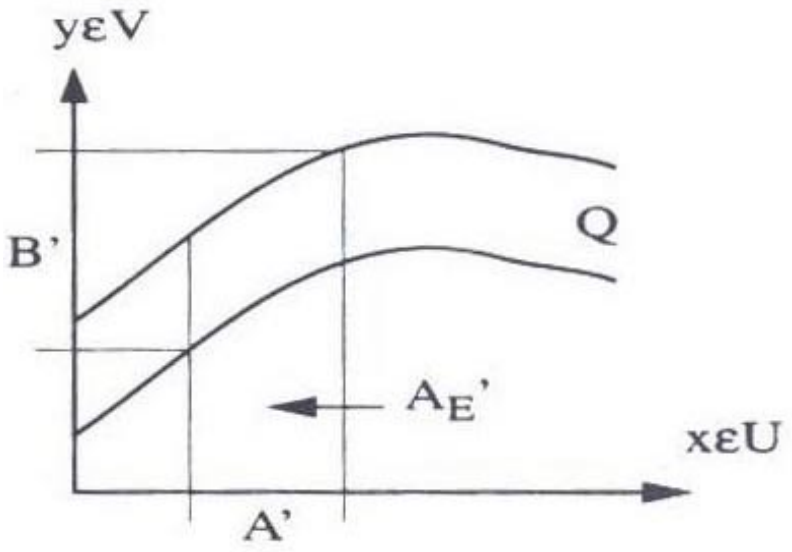

Figura 3.9: Infiriendo el conjunto difuso B' del conjunto difuso A' y la relación difusa Q

a (3.38) se le llama *regla composicional de inferencia*. El símbolo $\star$ es usado para representar un operador norma-t, entonces se puede reescribir

$$\mu_{B'}(y) = \sup_{x \in U} t\left[\mu_{A'}(x) \star \mu_Q(x, y)\right] \tag{3.39}$$

- **Modus Ponens generalizado**: Dado el conjunto difuso A' (que representa la parte premisa, x es A') y la relación difusa $A \rightarrow B$ en $U \times V$ (que representa la parte premisa SI x es A ENTONCES y es B), un conjunto difuso B' en V (representando la conclusión y es B') es inferido como

$$\mu_{B'}(y) = \sup_{x \in U} t\left[\mu_{A'}(x), \mu_{A \rightarrow B}(x, y)\right] \tag{3.40}$$

- **Modus Tollens generalizado**: Dado el conjunto difuso B' (que representa la parte premisa, y es B') y la relación difusa $A \rightarrow B$ en $U \times V$ (que representa la parte premisa SI x es A ENTONCES y es B), un conjunto difuso A' en U (representando la conclusión x es A') es inferido como

$$\mu_{A'}(y) = \sup_{y \in V} t\left[\mu_{B'}(y), \mu_{A \rightarrow B}(x, y)\right] \tag{3.41}$$

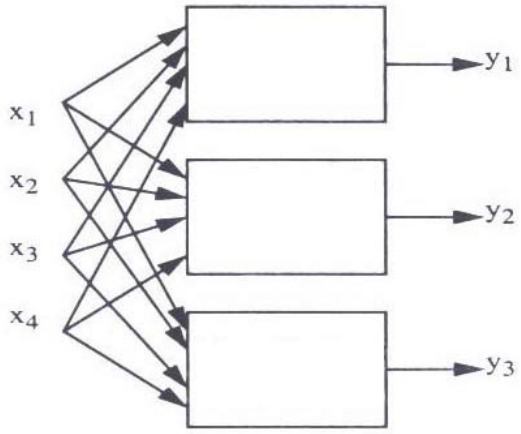

Figura 3.10: Un sistema difuso multi-entrada-multi-salida puede ser descompuesto en una colección de sistemas difusos multi-entrada-única-salida

- **Silogismo hipotético generalizado**: Dada la relación difusa $A \to B$ en $U \times V$ (que representa la parte premisa SI x es A ENTONCES y es B) y la relación difusa $B' \to C$ en $V \times W$ (que representa la parte premisa, SI y es B' ENTONCES z es C) una relación difusa $A \to C'$ en $U \times W$ (representando la conclusión SI x es A ENTONCES z es C') es inferido como

$$\mu_{A \to C}(x, z) = \sup_{y \in V} t\left[\mu_{A \to B}(x, y), \mu_{B' \to C}(y, z)\right] \tag{3.42}$$

3.3.4. Base de reglas difusas

Considere el sistema difuso mostrado en la figura 3.10. donde $U = U_1 \times U_2 \times ... \times U_n \subset R^n$ y $V \subset R$. Consideramos solo el caso multi-entrada-única-salida porque un sistema multi-salida siempre puede ser descompuesto en una colección de sistemas de una-salida.

Una *base de regla difusa* consiste en un conjunto de reglas difusas SI-ENTONCES, es el corazón del sistema difuso en el sentido de que todos los otros componentes son usados para implantar estas reglas de una manera rápida y eficiente. Específicamente, la base de reglas difusas contiene las siguientes reglas difusas SI-ENTONCES:

$$Ru^{(l)}\text{: SI } x_1 \text{ es } A_1^l \text{ y ... y } x_n \text{ es } A_n^l \text{ ENTONCES } y \text{ es } B^l \tag{3.43}$$

donde A_1^l y B^l son conjuntos difusos en $U_i \subset R$ y $V \subset R$, respectivamente, y $x = (x_1, x_2, ..., x_n)^T \in U$ y $y \in V$ son las variables (lingüísticas) de entrada y de salida respectivamente. Sea M el número de reglas en la base de reglas difusas; o sea, $l = 1, 2, ..., M$ en (3.43). Llamamos a las reglas en la forma de (3.43) *reglas difusas canónicas SI-ENTONCES* porque incluyen muchos otros tipos de reglas difusas y proposiciones difusas como casos especiales como se muestra en el siguiente lema.

Lemma 3.2 *Las reglas difusas canónicas SI-ENTONCES en la forma de (3.43) incluyen lo siguiente como casos especiales:*

1. *"Reglas parciales":*

$$SI\ x_1\ es\ A_1^l\ y\ ...\ y\ x_m\ es\ A_m^l\ ENTONCES\ y\ es\ B^l \tag{3.44}$$

   donde $m < n$.

2. *"Reglas o":*

$$SI\ x_1\ es\ A_1^l\ y\ ...\ y\ x_m\ es\ A_m^l\ o\ x_{m+1}\ es\ A_{m+1}^l\ y\ ...\ y\ x_n\ es\ A_n^l\ ENTONCES\ y\ es\ B^l \tag{3.45}$$

3. *Proposición difusa única*

$$y\ es\ B^l \tag{3.46}$$

4. *"Reglas graduales", por ejemplo:*

$$Entre\ más\ pequeña\ x,\ más\ grande\ y \tag{3.47}$$

5. *Reglas no difusas (o sea, reglas convencionales de producción)*

Definición 3.5 *Un conjunto de reglas difusas SI-ENTONCES es completo si para cualquier* $\mathbf{x} \in U$ *existe por lo menos una regla de base difusa, digamos regla* $Ru^{(l)}$ *en la forma de (3.43) tal que*

$$\mu_{A_i^l}(x_i) \neq 0 \tag{3.48}$$

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

Intuitivamente, lo *completo* de un conjunto de reglas significa que en cualquier punto en el espacio de entrada hay por lo menos una regla que "dispara"; esto es, el valor de membresía de la parte SI de la regla en este punto es no-cero.

Definición 3.6 *Un conjunto de reglas difusas SI-ENTONCES es consistente si no hay reglas con la misma parte SI pero diferentes partes ENTONCES.*

Para producción de reglas no-difusas, la consistencia es un requerimiento importante porque es difícil continuar la búsqueda si hay reglas en conflicto. Para reglas difusas, sin embargo, la consistencia no es crítica porque si hay reglas en conflicto la máquina de inferencia difusa y el difusificador los promediará automáticamente para producir un resultado. Por supuesto, siempre es mejor tener una regla de base difusa consistente.

Definición 3.7 *Un conjunto de reglas difusas SI-ENTONCES es continua si no existen vecindades de reglas las cuales la parte de conjuntos difusos ENTONCES tenga intersección vacía.*

Intuitivamente, continuidad significa que el comportamiento de entrada-salida del sistema difuso debe ser suave.

En una *máquina de inferencia difusa* se usan principios de lógica difusa para combinar las reglas difusas SI-ENTONCES en la base de reglas difusas en un mapeo de conjuntos difusos de un conjunto A' en U a un conjunto difuso en B' en V. Ya sabemos que una regla difusa SI-ENTONCES es interpretada como una relación difusa en el espacio de productos $U \times V$ de entrada-salida. Si la base de reglas difusas consiste solamente en una regla entonces se especifica el mapeo de el conjunto difuso A' en U al conjunto difuso B' en V, como cualquier base de reglas difusas práctica constituye en más de una regla, la pregunta clave es ¿Cómo inferir con un conjunto de reglas? Hay dos maneras de inferir con un conjunto de reglas: inferencia basada en composición e inferencia basada en regla-individual, las cuales se discuten a continuación.

En la Inferencia basada en composición todas las reglas en la base de reglas difusas son combinadas en una relación difusa única en $U \times V$, la cual es vista como una regla difusa SI-ENTONCES única. Para desarrollar esta combinación debemos primero entender lo que significa intuitivamente un conjunto de reglas y entonces podremos usar los operadores lógicos apropiados para combinarlos.

Hay dos argumentos opuestos para lo que debería significar un conjunto de reglas. El primero ve las reglas como proposiciones condicionales independientes, si aceptamos este punto de vista, un

operador razonable para combinar las reglas es *unión*. El segundo ve las reglas como proposiciones condicionales fuertemente unidas tal que se deben satisfacer las condiciones de todas las reglas para que todo el conjunto de reglas tenga un impacto, si adaptamos este punto de vista deberíamos usar el operador *intersección* para combinar las reglas. El segundo argumento puede parecer extraño pero para algunas implicaciones, por ejemplo la implicación Gödel, tiene sentido como veremos más adelante. Ahora veamos los detalles de estos dos argumentos.

Sea $Ru^{(l)}$ una relación difusa en $U \times V$ la cual representa la regla difusa SI-ENTONCES; o sea, $Ru^{(l)} = A_1^l \times ... \times A_n^l \rightarrow B^l$. Sabemos que $A_1^l \times ... \times A_n^l$ es una relación difusa en $U = U_1 \times ... \times U_n$ definida por

$$\mu_{A_1^l \times ... \times A_n^l}(x_1, ..., x_n) = \mu_{A_1^l}(x_1) \star ... \star \mu_{A_n^l}(x_n) \tag{3.49}$$

donde $\star$ representa a cualquier operador de norma-t. La implicación $\rightarrow$ en $Ru^{(l)}$ está definida de acuerdo a varias implicaciones (3.16)-(3.19), (3.24) y (3.25). Si aceptamos el primer punto de vista de un conjunto de reglas entonces las M reglas en la forma de (3.43) son interpretadas como relación difusa única Q_M en $U \times V$ definido por

$$Q_M = \bigcup_{l=1}^{M} Ru^{(l)} \tag{3.50}$$

Esta combinación es llamada la *combinación Mamdani*. Si usamos el símbolo $\dot{+}$ para representar las normas-s entonces puede ser reescrito como

$$\mu_{Q_M}(\mathbf{x}, y) = \mu_{Ru^{(1)}}(\mathbf{x}, y) \dot{+} ... \dot{+} \mu_{Ru^{(M)}}(\mathbf{x}, y) \tag{3.51}$$

Para el segundo punto de vista de un conjunto de reglas las M reglas difusas SI-ENTONCES de (3.43) son interpretadas como una relación difusa Q_G en $U \times V$ definida como

$$Q_G = \bigcap_{l=1}^{M} Ru^{(l)} \tag{3.52}$$

o equivalentemente

$$\mu_{G_M}(\mathbf{x}, y) = \mu_{Ru^{(1)}}(\mathbf{x}, y) \star ... \star \mu_{Ru^{(M)}}(\mathbf{x}, y) \tag{3.53}$$

donde $\star$ denota norma-t. Esta combinación es llamada la *combinación Gödel*.

Sea A' un conjunto difuso arbitrario en U y sea la entrada a la máquina de inferencia difusa, entonces, viendo a Q_M o Q_G como una regla difusa SI-ENTONCES única y usando el modo generalizado Ponens (3.40), obtenemos la salida de la máquina de inferencia difusa como

$$\mu_{B'}(y) = \sup_{\mathbf{x}\in U} t\left[\mu_{A'}(\mathbf{x}), \mu_{Q_M}(\mathbf{x}, y)\right] \tag{3.54}$$

si usamos la combinación Mamdani o como

$$\mu_{B'}(y) = \sup_{\mathbf{x}\in U} t\left[\mu_{A'}(\mathbf{x}), \mu_{Q_G}(\mathbf{x}, y)\right] \tag{3.55}$$

si usamos la combinación Gödel.

En *inferencia basada en regla individual* cada regla en la base de regla difusa determina un conjunto difuso de salida y la salida de toda la máquina de inferencia difusa es la combinación de las M conjuntos difusos individuales, la combinación puede ser hecha por unión o por intersección.

3.4. Modelos basados en redes difuso neuronales con aprendizaje de gradiente descendiente

3.4.1. Redes difuso neuronales tipo Mamdani

Los sistemas de inferencia difusos (control) han sido usados satisfactoriamente en una amplia variedad de problemas prácticos, especialmente para aplicaciones industriales. Una pregunta teórica fundamental acerca de los sistemas de inferencia difusos se mantiene sin respuesta, esta es, ¿Porque un sistema de inferencia difuso muestra tan excelente desempeño para tal variedad de aplicaciones? Las respuestas existentes son cualitativas, por ejemplo, "Los sistemas de inferencia difusos pueden utilizar información lingüística de humanos expertos", "Los sistemas de inferencia difusa pueden simular el procedimiento humano del pensamiento" y "Los sistemas de inferencia difusa capturan la aproximación, la naturaleza inexacta del mundo real". Consideremos la clase de sistemas de inferencia difusa cuyo conjunto de salida difuso de cada regla lógica difusa es un singleton. Se ha probado que esta clase de sistema de inferencia difusa es un aproximador universal [**?**, Jou]; o sea, es capaz de aproximar cualquier función real continua en un conjunto compacto a una exactitud arbitraria, si se tienen las suficientes reglas lógicas disponibles. A continuación discutiremos la prueba de este resultado fundamental el cual se basa en el teorema de Stone-Weierstrass.

Un esquema de inferencia difusa, el cual se basa en el modo generalizado modus Ponens (3.40), se puede describir esquemáticamente de la siguiente forma. Aquí consideramos sistemas difusos multiples-entradas-una-salida (MISO multi-input-single-output), $f : \Re^n \rightarrow \Re$, porque un sistema multi-salidas se puede siempre descomponer en una colección de sistemas única-salida.

Hecho:

$$x_1 \text{ ES } A'_1 \text{ Y } x_2 \text{ ES } A'_2 \text{ Y } ... \text{ Y } A'_n \text{ ES } A'_n \tag{3.56}$$

Regla 1:

$$\text{SI } x_1 \text{ ES } A^1_1 \text{ Y } x_2 \text{ ES } A^1_2 \text{ Y } ... \text{ Y } A^1_n \text{ ES } A^1_n, \text{ ENTONCES } y \text{ ES } B^1 \text{ SI NO} \tag{3.57}$$

Regla 2:

$$\text{SI } x_1 \text{ ES } A^2_1 \text{ Y } x_2 \text{ ES } A^2_2 \text{ Y } ... \text{ Y } A^2_n \text{ ES } A^2_n, \text{ ENTONCES } y \text{ ES } B^2 \text{ SI NO} \tag{3.58}$$

Regla m:

$$\text{SI } x_1 \text{ ES } A^m_1 \text{ Y } x_2 \text{ ES } A^m_2 \text{ Y } ... \text{ Y } A^m_n \text{ ES } A^m_n, \text{ ENTONCES } y \text{ ES } B^m \tag{3.59}$$

Conclusión:

$$y \text{ ES } B' \tag{3.60}$$

La n-ésima variable $x = (x_1, x_2, ..., x_n)^T$ denota la entrada y la variable y denota la salida. Para cada variable lingüística x_i, U_i es el universo de colección de posibles patrones; $x_i \in U_i$, $i = 1, 2, ..., n$. Sea U un producto cartesiano de universos $U = U_1 \times U_2 \times ... \times U_n$. En consecuencia, Sea V el universo de y. Hay m funciones membresía para cada x_i, $i = 1, 2, ..., n$. y y produciendo medidas de membresía para cada variable con respecto a los conjuntos difusos A^j_i y B^j, respectivamente, y $\mu_{A^j_i}(x_i) : U_i \rightarrow [0, 1]$ y $\mu_{B^j}(y) : V \rightarrow [0, 1]$, $j = 1, 2, ..., m$. Nótese que no hay restricciones en la forma de las funciones de membresía; pueden ser lineales o no lineales.

Cada una de las reglas lógicas difusas o implicaciones difusas asociando n conjuntos de entradas difusas con un conjunto de salidas difusas puede ser representado por una relación difusa $R^j =$

$\left(A_1^j \text{ Y } A_2^j \text{ Y } ... \text{ Y } A_n^j\right) \rightarrow B^j$ definido en $U_1 \times ... \times U_n \times V$. Aquí se considera el operador producto para la relación difusa que es

$$\mu_{R^j}(x_1, ..., x_n, y) = \mu_{A_1^j}(x_1) ... \mu_{A_n^j}(x_n) \cdot \mu_{B^j}(y) \tag{3.61}$$

Las relaciones difusas R^j, $j = 1, 2, ..., m$, codificando las reglas lógicas difusas pueden ser agregadas para formar la relación completa R interpretando ENTONCES como uniones difusas; o sea,

$$\mu_R(x_1, ..., x_n, y) = \bigvee_{j=1}^{m} \mu_{R^j}(x_1, ..., x_n, y) \tag{3.62}$$

donde $\bigvee$ denota el operador máx binario. Si x_i toma el conjunto difuso de entrada A_i', $i = 1, 2, ..., n$, entonces el conjunto difuso salida o consecuencia B' puede ser deducido usando la operación de composición difusa como sigue:

$$B' = \left(A_1' \text{ Y } A_2' \text{ Y } ... \text{ Y } A_n'\right) \circ R \tag{3.63}$$

donde $\circ$ denota la regla de inferencia del producto máx composicional. Explícitamente, la función de membresía de la parte consecuente B'es

$$\mu_{B'}(y) = \bigvee_{x_1,...,x_n} \left[\left(\prod_{i=1}^{n} \mu'_{A_i}(x_i) \right) \cdot \left(\bigvee_{j=1}^{m} \left(\prod_{i=1}^{n} \mu_{A_i^j}(x_i) \right) \cdot \mu_{B^j}(y) \right) \right] \tag{3.64}$$

Como la ecuación (3.64) mapea conjuntos difusos a conjuntos difusos, define un mapeo difuso $F(A_1', A_2', ..., A_n') = B'$. En la práctica, especialmente en aplicaciones de control, el conjunto de entrada difuso A_i' iguala a un dato numérico a_i'. En este caso, el proceso de difusificación puede ser hecho por un singleton difuso; esto es, $\mu'_{A_i}(x_i) = 1$ si $x_i = a_i'$ y $\mu'_{A_i(x_i)} = 0$ si $x_i \neq a_i'$. En este caso la consecuencia B' en la ecuación (3.64) se convierte en

$$\mu_{B'}(y) = \bigvee_{j=1}^{m} \left[\left(\prod_{i=1}^{n} \mu_{A_i^j}(x_i) \right) \cdot \mu_{B^j}(y) \right] \tag{3.65}$$

La información en el conjunto resultante B' obtenida de cualquiera de las ecuaciones (3.64) y (3.65) reside mayormente en los valores relativos de los grados de membresía. Podemos *desdifusificar* el conjunto de salida difusa $B\prime$ para producir una salida numérica la cual es un punto representativo

de B'. Usando el método de desdifusificación de centroide, la salida numérica final y^* inferida de las reglas lógicas difusas puede ser únicamente determinada de el conjunto de salidas difusas B' como sigue:

$$y^* = \frac{\int_V \mu_{B'}(y)\, y dy}{\int_V \mu_{B'}(y)\, dy} \tag{3.66}$$

Al introducir los procesos de difusificación singleton y desdifusificación centroide en el sistema de inferencia, hemos transformado el mapeo difuso en la ecuación (3.65) a el de la ecuación (3.66), la cual simplemente mapea n entradas numéricas x_i, $i = 1, 2, ..., n$ a un valor de salida y. Este tipo de sistemas de inferencia difusos han sido ampliamente usados en el área de control difuso.

Solo hemos revisado el proceso de inferencia de un sistema difuso típico. A continuación discutiremos un modelo simplificado en el cual se obtendrá la prueba de aproximador universal. Consideremos que el conjunto de salida difusa B^j en la ecuación de las reglas son singletones β^j, esto es, $\mu_{B^j}(y) = 1$ si $y = \beta^j$ y $\mu_{B^j}(y) = 0$ si y toma otro valor, $j = 1, 2, ..., m$. Además, vemos las reglas difusas con los mismos singletones de salida ($\beta^l = \beta^k$ para $l \neq k$) como distintas reglas difusas, y todas ellas contribuyen igualmente a la conclusión final inferida. En otras palabras, el operador max $\bigvee$ es removido. Con esta simplificación la ecuación se convierte en

$$\mu'_B(y) = \begin{cases} \prod_{i=1}^n \mu_{A_i^j}(x_i) & \text{si } y = \beta^j \\ 0 & \text{si } y \neq \beta^j \end{cases} \tag{3.67}$$

donde $x_i = 1, 2, ..., n$ son singletones difusos. Como el conjunto resultante B' es discreto, podemos reemplazar el radio de la integral en la ecuación de y^* con un radio de sumas discretas simples y obtener puntos representativos de B' como sigue

$$y^* = \frac{\sum_{j=1}^m \beta^j \left(\prod_{i=1}^n \mu_{A_i^j}(x_i) \right)}{\sum_{j=1}^m \prod_{i=1}^n \mu_{A_i^j}(x_i)} \tag{3.68}$$

A continuación mostraremos que el sistema difuso inferido simplificado puede usarse para aproximar cualquier función arbitraria en $C(\Re^n)$ a cualquier grado de exactitud deseado, donde $C(\Re^n)$ denota el conjunto de todas las funciones continuas en $\Re^n$. Para ser precisos acerca de la clase de sistemas de inferencia difusa simplificados en consideración, las siguientes notaciones y definiciones son necesarias. El *soporte* de una función con valores-reales f en $\Re^n$ es el cierre del conjunto de

todos los puntos $\mathbf{x}$ en $\Re^n$ en donde $f(\mathbf{x}) \neq 0$. Una colección $\{V_\alpha\}$ de subconjuntos abiertos de $\Re^n$ es una *cubierta abierta* de un conjunto E de $\Re^n$ si $E \subset U_\alpha V_\alpha$. Supongamos que U es un subconjunto compacto de $\Re^n$ y $\{V_\alpha\}$ es una cubierta abierta de U, entonces para funciones $\psi_1, \psi_2, ..., \psi_m$ en $C(\Re^n)$, $\{\psi_j\}$ es llamada una *partición de unidad* si (i) $\psi_j \in [0,1]$ para $j = 1, 2, ..., m$; (ii) cada ψ_j tiene soporte en algún V_α; y (iii) $\sum_{j=1}^m \psi_j(\mathbf{u}) = 1$ para cada $\mathbf{u}$ en U. Usando la notación de partición de unidad, podemos reescribir el sistema de inferencia difuso simplificado como sigue. Sea $\psi_j(x)$ la función de entrada $\mathbf{x} = (x_1, x_2, ..., x_n)$:

$$\psi_j(\mathbf{x}) = \frac{\prod_{i=1}^n \mu_{A_i^j}(x_i)}{\sum_{k=1}^m \prod_{i=1}^n \mu_{A_i^k}(x_i)}, \quad j = 1, 2, ..., m \tag{3.69}$$

Asumiendo que el denominador $\sum_{k=1}^m \prod_{i=1}^n \mu_{A_i^j}(x_i) \neq 0$ para cada $\mathbf{x}$ [o sea, $\prod_{i=1}^n \mu_{A_i^j}(x_i) \neq 0$ para algún $k \in \{1, 2, ..., m\}$ o, equivalentemente, cada ψ_j tiene soporte en algún V_α], entonces $\sum_{j=1}^m \psi_j(\mathbf{x}) = 1$ para cada $\mathbf{x}$ y por lo tanto las funciones ψ_j, $j = 1, 2, ..., m$, forman una partición de unidad. Podemos definir la clase de sistemas de inferencia difusos como una familia de funciones $f : \Re^n \to \Re$ en la forma de

$$f(\mathbf{x}) = \sum_{j=1}^m \beta^j \psi_j(\mathbf{x}) \tag{3.70}$$

para cada $\mathbf{x} \in \Re^n$, $\beta^j \in \Re$ y m es un número finito de reglas difusas. Denotemos la familia de funciones f de la ecuación (3.70) como F^n. La ecuación anterior tiene exactamente la misma forma que la ecuación de y^* donde $\mathbf{x}$ corresponde a la entrada, la función ψ_j corresponde a la activación de la j-ésima regla lógica difusa y el escalar β^j corresponde a la j-ésima salida singleton. Como F^n es de hecho la clase de sistemas de inferencia difusos en consideración. En la representación de la ecuación anterior nos referimos a ψ_j como un conjunto de *funciones base*. Este modelo hace uso de conjuntos de entrada difusos teniendo un campo receptivo local con una curva sensitiva que cambia como una función de la distancia de un punto en particular en el espacio de entrada. El uso de funciones de membresía de conjuntos de entrada difusos con campos receptivos de superposición claramente proveen un tipo de interpolación y extrapolación.

Se puede mostrar que la clase de sistemas de inferencia difusos F^n puede ser usada para aproximar cualquier función real y continua en un conjunto compacto usando el teorema de Stone-Weierstrass

Teorema 3.1 ***Teorema Stone-Weierstrass****. Sea A un conjunto de funciones reales y continuas en un conjunto compacto U. Si (a) A es un álgebra, (b) A posee puntos separados en U y (c) A no se desvanece en ningún punto de U entonces el cierre uniforme de a consiste de todas las funciones reales y continuas en U.*

3.4.2. Redes difuso neuronales tipo Takagi-Sugeno

Estos modelos se caracterizan por relaciones basadas en reglas difusas donde las premisas de cada regla representan subespacios difusos y las consecuencias son una relación lineal de entrada-salida [18].

Las variables de entrada en las premisas de cada regla son relacionadas por operadores "y" y la variable de salida es una combinación lineal de las variables de estado. Por lo tanto, las reglas del modelo tienen la siguiente forma:

R_i :Si $x_1$es A_{1_i} y ... y x_k es A_{k_i} entonces

$$Y_i = p_0^i + p_1^i x_1 + ... + p_k^i x_k \tag{3.71}$$

donde:

$x_1, ..., x_k$ son las variables de entrada o premisas de las reglas

$A_{1_i}, ..., A_{k_i}$ son los conjuntos difusos asociados a las variables de entrada

$p_0^i, ..., p_k^i$ son los parámetros de la regla i

Y_i es la salida de la regla i.

Por lo tanto, la salida del modelo, Y, se obtiene ponderando la salida de cada regla por su respectivo grado de cumplimiento W_i, es decir:

$$Y = \frac{\sum (W_i Y_i)}{\sum (W_i)} \tag{3.72}$$

donde M es el número de reglas del modelo y W_i se calcula según el operador intersección.

3.4.3. Formato de implicación difusa y algoritmo de razonamiento

Denotemos la función de membresía de un conjunto A como $A(x)$, $x \in X$. Todos los conjuntos difusos están asociados con funciones de membresía lineales. Por lo tanto, una función de membresía

está caracterizada por dos parámetros dando el grado máximo 1 y el mínimo 0. El valor de verdad de una proposición "x es A y y es B" se expresa como:

$$|x \text{ es } A \text{ y } y \text{ es } B| = A(x) \wedge B(y) \tag{3.73}$$

Formato de implicaciones

Sugerimos que una implicación difusa R sea de la forma

$$R : \text{Si } f(x_1 \text{ es } A_1, ..., x_k \text{ es } A_k) \text{ entonces } y = g(x_1, ..., x_k) \tag{3.74}$$

donde y es la variable de la consecuencia cuyo valor es inferido, $x_1, ..., x_k$ son las variables de la parte premisa que aparecen también en la parte consecuencia, $A_1, ..., A_k$ son conjuntos difusos con funciones de membresía lineales representando a un subespacio difuso en el cual la implicación R puede ser aplicada para razonamiento, f es la función lógica que conecta las proposiciones en la parte premisa y g es la función que implica el valor de y cuando $x_1, ..., x_k$ satisface la parte premisa.

Si la parte premisa A_i es igual a X_i para algún i donde X_i es el universo de discordia de x_i, este término se omite; x_i es no condicionado.

Algoritmo de razonamiento

Supongamos que tenemos implicacionesR^i $(i = 1, ..., n)$ de la forma anterior. Cuando nos es dado $\left(x_1 = x_1^0, ..., x_1 = x_k^0\right)$ donde $x_1^0, ..., x_k^0$ son singletones, el valor de y es inferido en los siguientes pasos

1) Para cada implicación R^i, y^i se calcula con la función g^i en la parte consecuencia

$$y^i = g^i\left(x_1^0, ..., x_1^0\right) = p_0^i + p_1^i x_1^0, ..., p_k^i x_k^0 \tag{3.75}$$

2) El valor verdadero de la proposición $y = y^i$ es calculado por la ecuación

$$\left|y = y^i\right| = \left|\left(x_i^0 \text{ es } A_1^i \text{ y } ... \text{ y } x_k^0 \text{ es } A_k^i\right)\right| \wedge \left|R^i\right| = \left(A_1^i(x_1^o) \wedge ... \wedge A_k^i(x_k^o)\right) \wedge \left|R^i\right| \tag{3.76}$$

donde $|*|$ denota el valor de verdad de la proposición $*$ y $\wedge$ denota el operador min y $\left|x^0 \text{ es } A\right| = A\left(x^0\right)$es el grado de membresía de x^0.

Por simplicidad asumimos $\left|R^i\right| = 1$, entonces el valor de verdad de la parte consecuencia obtenido es

$$\left|y = y^i\right| = A_1^i\left(x_1^o\right) \wedge ... \wedge A_k^i\left(x_k^o\right) \tag{3.77}$$

3) La salida final y inferida de n implicaciones es dada como el promedio de todos los y^i con los pesos $\left|y = y^i\right|$:

$$y = \frac{\sum \left|y = y^i\right| \times y^i}{\sum \left|y = y^i\right|} \tag{3.78}$$

Identificación de parámetros de la parte consecuencia

Ahora mostraremos como determinar los parámetros óptimos de la parte consecuencia para minimizar el índice de desempeño, cuando las variables premisas y los parámetros son conocidos. El índice de desempeño se ha definido como el promedio de la raíz cuadrada de los errores de salida, lo que significa, las diferencias entre los datos de salida del sistema original y los del modelo.

Representemos un sistema por las siguientes implicaciones:

$$\begin{aligned} R^1 \text{ Si } x_1 \text{ es } A_1^1, \text{ ... , y } x_k \text{ es } A_k^1 \text{ entonces } y = p_0^1 + p_1^1 \cdot x_1 + ... + p_k^1 \cdot x_k \\ \vdots \\ R^n \text{ Si } x_1 \text{ es } A_1^n, \text{ ... , y } x_k \text{ es } A_k^n \text{ entonces } y = p_0^n + p_1^n \cdot x_1 + ... + p_k^n \cdot x_k \end{aligned} \tag{3.79}$$

Entonces la salida y para la entrada $(x_1, ..., x_k)$ es obtenida como

$$y = \frac{\sum_{i=1}^n \left(A_1^i\left(x_1\right) \wedge ... \wedge A_n^i\left(x_n\right)\right) \cdot \left(p_0^i + p_1^i \cdot x_1 + ... + p_k^i \cdot x_k\right)}{\sum_{i=1}^n \left(A_1^i\left(x_1\right) \wedge ... \wedge A_n^i\left(x_n\right)\right)} \tag{3.80}$$

Sea

$$\beta_i = \frac{A_1^i\left(x_1\right) \wedge ... \wedge A_n^i\left(x_n\right)}{\sum_{i=1}^n \left(A_1^i\left(x_1\right) \wedge ... \wedge A_n^i\left(x_n\right)\right)} \tag{3.81}$$

entonces

$$y = \sum_i^n \beta_i \left(p_0^i + p_1^i \cdot x_1 + ... + p_k^i \cdot x_k\right) = \sum_{i=1}^n \left(p_0^i \cdot \beta_i + p_1^i \cdot x_1 \cdot \beta_i + ... + p_k^i \cdot x_k \cdot \beta_i\right) \tag{3.82}$$

Cuando un conjunto de datos de entrada-salida $x_{1j}, x_{2j}, ..., x_{kj} \rightarrow y_i$ $(j = 1, ..., m)$ es dado, podemos obtener los parámetros consecuencia $p_0^i, p_1^i, ..., p_k^i \, (i = 1, ..., n)$ por el método de mínimos cuadrados usando la ecuación (3.82).

Sea X (matriz de $m \times n(k+1)$), Y (vector m) y P (vector de $n(k+1)$) entonces

$$Y = PX,\ X = \begin{vmatrix} \beta_{11}, ..., & \beta_{n1}, x_{11} \cdot \beta_{11}, ..., & x_{11} \cdot \beta_{n1}, ... & \\ & \cdots & x_{k1} \cdot \beta_{11}, ..., & x_{k1} \cdot \beta_n \\ \vdots & & & \\ \beta_{1m}, ..., & \beta_{nm}, x_{1m} \cdot \beta_{1m}, ..., & x_{1m} \cdot \beta_{1m}, ... & \\ & \cdots & x_{k1} \cdot \beta_{1m}, ..., & x_{k1} \cdot \beta_{nm} \end{vmatrix} \tag{3.83}$$

donde

$$\beta_{ij} = \frac{A_{i1}(x_{1j}) \wedge ... \wedge A_{ik}(x_{kj})}{\sum_j A_{i1}(x_{1j}) \wedge ... \wedge A_{ik}(x_{kj})} \tag{3.84}$$

$$Y = [y_1, ..., y_m]^T \tag{3.85}$$

$$P = \left[p_0^1, ..., p_0^n, p_1^1, ..., p_1^n, ..., p_k^1, ..., p_k^n\right]^T \tag{3.86}$$

Entonces el vector de parámetros P se calcula:

$$P = \left(X^T X\right)^{-1} X^T Y \tag{3.87}$$

Nótese que el método propuesto es consistente con el método de razonamiento. En otras palabras, este método de identificación nos permite obtener los mismos parámetros que como en el sistema original, si tenemos un suficiente número de datos de salida libres de ruido para la identificación.

El vector de parámetros P es calculado por un filtro Kalman estado-estable. El llamado filtro Kalman estado-estable es un algoritmo para calcular los parámetros de una ecuación lineal algebraica que dan los errores de mínimos cuadrados. Aquí lo aplicamos para calcular el vector de parámetros P.

Sea x_i el i-ésimo vector fila de la matriz X definida y y_i el i-ésimo elemento de Y. Entonces P es recursivamente calculado por las siguientes ecuaciones donde S_i es una matriz $(n \cdot (k+1)) \times (n \cdot (k+1))$.

$$P_{i+1} = P_i + S_{i+1} \cdot x_{i+1} \cdot (y_{i+1} - x_{i+1} \cdot P_i) \tag{3.88}$$

$$S_{i+1} = S_i - \frac{S_i \cdot x_i + x_{i+1} \cdot S_i}{1 + x_{i+1} \cdot S_i \cdot x_{i+1}^T} \tag{3.89}$$

$$P = P_m \tag{3.90}$$

donde los valores iniciales de P_0 y S_0 son los conjuntos siguientes.

$$P_0 = 1 \tag{3.91}$$

$$S_0 = \alpha \cdot I \ (\alpha = \text{número muy grande}) \tag{3.92}$$

donde I es la matriz identidad.

Capítulo 4

Modelado difuso neuronal con función de membresía conocida con algoritmo de aprendizaje estable

4.1. Modelado difuso neuronal

Usamos $l(i = 1, 2 \cdots l)$ reglas difusas SI-ENTONCES para desarrollar un mapeo de un vector de entrada lingüística $X = [x_1 \cdots x_n] \in \Re^n$ a un vector de salida lingüística $\widehat{Y}(k) = [\widehat{y}_1 \cdots \widehat{y}_m]^T$. $A_{1i}, \cdots A_{ni}$ y $B_{1i}, \cdots B_{mi}$ son conjuntos difusos standard [22]. Para una variable de entrada x_i hay l_i conjuntos difusos. En el caso de una conexión completa, $l = l_1 \times l_2 \times \cdots l_n$. De [22] sabemos que, al usar el difusificador inferencia de productos, difusificador centro-promedio y función de activación singleton, la p-ésima salida del sistema lógico difuso puede ser expresado como

$$\begin{aligned} \widehat{y}_p &= \left(\sum_{i=1}^{l} w_{pi} \left[\prod_{j=1}^{n} \mu_{A_{ji}} \right] \right) / \left(\sum_{i=1}^{l} \left[\prod_{j=1}^{n} \mu_{A_{ji}} \right] \right) \\ &= \sum_{i=1}^{l} w_{pi} \phi_i \end{aligned} \tag{4.1}$$

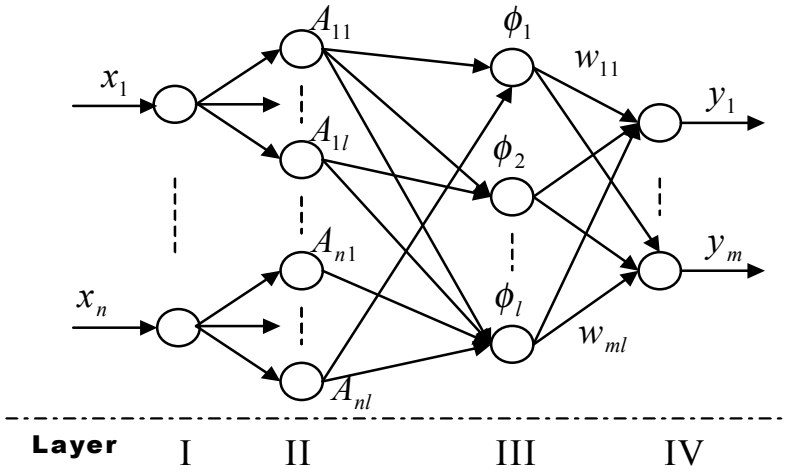

Figura 4.1: Arquitectura del sistema difuso neuronal

donde $\mu_{A_{ji}}$ es la función de membresía del conjunto difuso A_{ji}, w_{pi} es el punto al cuál $\mu_{B_{pi}} = 1$. Si definimos

$$\phi_i = \prod_{j=1}^{n} \mu_{A_{ji}} / \sum_{i=1}^{l} \prod_{j=1}^{n} \mu_{A_{ji}} \tag{4.2}$$

(5.2) puede ser expresado en la forma matricial

$$\widehat{Y}(k) = W(k)\,\Phi[X(k)] \tag{4.3}$$

donde el parámetro $W(k) = \begin{bmatrix} w_{11} & & w_{1l} \\ & \ddots & \\ w_{m1} & & w_{ml} \end{bmatrix}$, y el vector de datos $\Phi[X(k)] = [\phi_1 \cdots \phi_l]^T$.
La estructura del sistema difuso neuronal se muestra en la figura 4.1. Esta red difuso neuronal de cuatro capas se ha discutido en muchos artículos [7], [11], [13] y [21]. La capa I acepta vectores lingüísticos $X(k)$. Cada nodo de la capa II representa el valor de la función de membresía de la variable lingüística. Los nodos de la capa III representan reglas difusas. La capa IV es la capa de salida, las conexiones entre las capas III y IV son completamente conectadas por la matriz de pesos $W(k)$. Las capas I y II son la parte premisa. Las capas III y IV son la parte consecuencia.

4.2. Aprendizaje sin modificación robusta

Cuando tenemos alguna información a priori de la planta a identificar, podemos construir reglas difusas como (4.1) y

$$R^i : \text{SI } x_1 \text{ es } A_{1i} \text{ y } x_2 \text{ es } A_{2i} \text{ y } \cdots \text{ y } x_n \text{ es } A_{ni} \text{ ENTONCES } \widehat{y}_j = p^i_{j0} + p^i_{j1}x_1 + \cdots p^i_{jn}x_n$$

En esta sección asumiremos que las funciones de membresía $A_{1i} \cdots A_{ni}$ se conocen a priori, i.e., se conoce $\phi_i = \prod_{j=1}^{n} \mu_{A_{ji}} / \sum_{i=1}^{l} \prod_{j=1}^{n} \mu_{A_{ji}}$ (ver que los modelos [4], [20], [24] Mamdani (5.4) y TSK [18, Takagi] tienen la misma forma porque $\Phi[X(k)]$ es conocida, la única diferencia es la definición de $W(k)$).El objetivo del modelado difuso neuronal es encontrar los valores centrales de $B_{1i} \cdots B_{mi}$ (los pesos entre las capas III y IV en la figura 4.1), de tal forma que la salida $\widehat{Y}(k)$ de la red difuso neuronal (4.3) pueda seguir a la salida $Y(k)$ de la planta no lineal (1.7).

Definamos el error de identificación como

$$e(k) = \widehat{Y}(k) - Y(k) \tag{4.4}$$

El modelado difuso neuronal discutido es un tipo de identificación en-línea, esto es, usaremos el error de modelado $e(k)$ para entrenar la red difuso neuronal (5.4) en-línea tal que $\widehat{Y}(k)$ pueda aproximarse a $Y(k)$. De acuerdo a la teoría de aproximación de funciones de lógica difusa [21] y redes neuronales [6], el proceso no lineal identificado (1.7) puede representarse como

$$Y(k) = W^* \Phi[X(k)] - \mu(k) \tag{4.5}$$

donde W^* son los pesos desconocidos los cuales pueden minimizar la dinámica no modelada $\mu(k)$. El error de identificación puede representarse como en (4.4) y (4.5)

$$e(k) = \widetilde{W}(k) \Phi[X(k)] + \mu(k) \tag{4.6}$$

donde $\widetilde{W}(k) = W(k) - W^*$. En esta tesis solo estamos interesados en identificación de lazo abierto, podemos asumir que la planta (1.7) es estable acotada a la entrada y acotada a la salida, *i.e.*, $y(k)$ y $u(k)$ en (1.7) son acotados. Por la cota de la función de membresía Φ podemos asumir que $\mu(k)$ en (4.5) es acotada. El siguiente teorema da un algoritmo de gradiente descendiente estable para modelado difuso neuronal.

Teorema 4.1 *Si usamos las redes difuso neuronales (5.4) para modelar plantas no lineales (1.7), el siguiente algoritmo de gradiente descendiente con velocidad de aprendizaje variable puede hacer al error de identificación* $e(k)$ *acotado*

$$W(k+1) = W(k) - \eta_k e(k) \Phi^T [X(k)] \tag{4.7}$$

donde $\eta_k = \dfrac{\eta}{1 + \|\Phi [X(k)]\|^2}$, $0 < \eta \leq 1$. *El error de identificación normalizado satisface el siguiente desempeño promedio*

$$J = \lim_{T\to\infty} \sup \frac{1}{T} \sum_{k=1}^{T} e_N^2(k) \leq \overline{\mu} \tag{4.8}$$

donde $\overline{\mu} = \max_k \left[\mu^2(k)\right]$.

Demostración. Escogemos una matriz definida positiva L_k como

$$L_k = \left\| \widetilde{W}(k) \right\|^2 \tag{4.9}$$

la ley de actualización (4.7), tenemos

$$\widetilde{W}(k+1) = \widetilde{W}(k) - \eta_k e(k) \Phi^T [X(k)] \tag{4.10}$$

Usando las desigualdades

$$\|a - b\| \geq \|a\| - \|b\|, -2\|ab\| \leq a^2 + b^2 \tag{4.11}$$

para cualquier a y b. Usando (4.6) y $0 \leq \eta_k \leq \eta \leq 1$, tenemos

$$\begin{aligned}
\Delta L_k &= L_{k+1} - L_k = \left\| \widetilde{W}(k) - \eta_k e(k) \Phi^T(X) \right\|^2 - \left\| \widetilde{W}(k) \right\|^2 \\
&= \eta_k^2 e^2(k) \|\Phi [X(k)]\|^2 - 2\eta_k \|e(k)\| \left\| \Phi^T(X) \widetilde{W}(k) \right\| \\
&= \eta_k^2 e^2(k) \|\Phi [X(k)]\|^2 - 2\eta_k \|e(k) [e(k) - \mu(k)]\| \\
&\leq \eta_k^2 e^2(k) \|\Phi [X(k)]\|^2 - 2\eta_k e^2(k) + \eta_k e^2(k) + \eta_k \mu^2(k) \\
&= -\eta_k e^2(k) \left[1 - \eta_k \left\| \Phi^T(X) \right\|^2\right] + \eta \mu^2(k)
\end{aligned} \tag{4.12}$$

Como

$$\begin{aligned}
&1 - \eta_k \|\Phi [X(k)]\|^2 = 1 - \frac{\eta}{1 + \|\Phi [X(k)]\|^2} \|\Phi [X(k)]\|^2 \\
&\geq 1 - \eta \frac{\max_k \left(\|\Phi[X(k)]\|^2\right)}{1 + \max_k \left(\|\Phi[X(k)]\|^2\right)} \geq 1 - \frac{\max_k \left(\|\Phi[X(k)]\|^2\right)}{1 + \max_k \left(\|\Phi[X(k)]\|^2\right)} \\
&= \frac{1}{1 + \max_k \left(\|\Phi[X(k)]\|^2\right)}
\end{aligned} \tag{4.13}$$

y $\eta_k \leq \eta$

$$\Delta L_k \leq -\pi e^2(k) + \eta\mu^2(k) \tag{4.14}$$

donde π se define como

$$\pi = \frac{\eta}{1 + \max_k \left(\|\Phi[X(k)]\|^2 \right)} \tag{4.15}$$

Porque

$$n\,\min\left(\widetilde{w}_i^2\right) \leq L_k \leq n\,\max\left(\widetilde{w}_i^2\right) \tag{4.16}$$

donde $n\,\min\left(\widetilde{w}_i^2\right)$ y $n\,\max\left(\widetilde{w}_i^2\right)$ son funciones-$\mathcal{K}_\infty$, y $\pi e^2(k)$ es una función-$\mathcal{K}_\infty$, $\eta\mu^2(k)$ es una función-$\mathcal{K}$. De (4.6) y (4.9) sabemos que L_k es la función de $e(k)$ y $\mu(k)$, entonces L_k admite una función suave ISS-Lyapunov como en *Definition 2*. Del *Theorem 1*, la dinámica de la identificación del error es entrada-a-salida estable. La "ENTRADA" es correspondiente al segundo término de (4.14), *i.e.*, el error de modelado $\mu(k)$, el "ESTADO" es correspondiente al primer término de (4.12), *i.e.*., el error de identificación $e(k)$. Porque la "ENTRADA" $\mu(k)$ es acotada y la dinámica es ISS, el "ESTADO" $e(k)$ es acotado.

(4.12) puede ser reescrito como

$$\Delta L_k \leq -\eta \frac{e^2(k)}{1 + \max_k \left(\|\Phi[X(k)]\|^2 \right)} + \eta\mu^2(k) \leq -\eta \frac{e^2(k)}{1 + \max_k \left(\|\Phi[X(k)]\|^2 \right)} + \eta\overline{\mu} \tag{4.17}$$

Resumiendo (4.17) de1 hasta T, y usando $L_T > 0$ y L_1 constante, obtenemos

$$\begin{array}{c} L_T - L_1 \leq -\eta \sum_{k=1}^{T} e_N^2(k) + T\eta\overline{\mu} \\ \eta \sum_{k=1}^{T} e_N^2(k) \leq L_1 - L_T + T\eta\overline{\mu} \leq L_1 + T\eta\overline{\mu} \end{array} \tag{4.18}$$

(4.8) es establecida. ■

Comentario 4.1 *Si las redes difuso neuronales (5.4) pueden seguir plantas no lineales (1.7) exactamente ($\mu(k) = 0$), i.e., podemos encontrar la mejor función de membresía $\mu_{A_{ji}}$ y W^* tal que el sistema no lineal puede ser reescrito como $Y(k) = W^*\Phi\left[\mu_{A_{ji}}\right]$. Debido a $e^2(k) > 0$, el error de identificación es asintóticamente estable,*

$$\lim_{k\to\infty} e(k) = 0 \tag{4.19}$$

Comentario 4.2 *En general el modelado difuso neuronal puede no seguir sistemas no lineales exactamente. No podemos hacer converger los parámetros del modelo difuso neuronal a sus valores óptimos, solo queremos forzar la salida de redes difuso neuronales para que siga la salida de la planta. Aunque los parámetros pueden no converger a sus valores óptimos, (4.8) muestra que el error de identificación normalizado convergerá a un cierto radio* $\overline{\mu}$.

4.3. Simulación

A continuación los algoritmos de aprendizaje estable sugeridos son evaluados como función de aproximación y sistema de identificación.

4.3.1. Función de aproximación de dos dimensiones

Queremos usar el algoritmo estable para aproximar la siguiente función

$$f(x_1, x_2) = 0{,}52 + 0{,}1x_1 + 0{,}28x_2 - 0{,}6x_1x_2 \tag{4.20}$$

Este ejemplo se tomo de [22], los cuales usaron el siguiente sistema difuso

$$\widehat{f} = \frac{\sum_{x_1}\sum_{x_2} f(x_1, x_2)\mu_{A_1}\mu_{A_2}}{\sum_{x_1}\sum_{x_2}\mu_{A_1}\mu_{A_2}} \tag{4.21}$$

para aproximarlo. Usaremos redes difuso neuronales (5.4).

La entrada

$$X(k) = [x_1(k), x_2(k)]. \tag{4.22}$$

Los conjuntos difusos de A_i son los mismos que en [22]. El número de conjuntos difusos para cada variable de entrada es 11. Hay 121 reglas difusas. Las funciones de membresías para x_1 y x_2 son funciones triangulares en $[-1, 1]$, ver figura 4.2.

Como en la figura 4.1, $n = 2$, $l_1 = l_2 = 11$, $l = 121$, $m = 1$. $x_1(k)$ y $x_2(k)$ se escogen como

$$x_1(k) = -1 + \frac{2k}{T}; \quad x_2(k) = 1 - \frac{2k}{T} \tag{4.23}$$

donde $k = 1, 2, \cdots T$. En esta simulación usaremos $T = 600$. El algoritmo de aprendizaje es como en (4.7), con $\eta = 1$. Los resultados de la identificación se muestran en la figura 4.3.

Para evaluar la efectividad del algoritmo, encontramos que después de $\eta \geq 2{,}7$, el proceso de aprendizaje se vuelve inestable. El identificación resultante con $\eta = 2{,}7$ se muestra en la figura 5.2. El *Teorema 2* da una condición **necesaria** de η para un aprendizaje estable, $\eta \leq 1$. En este ejemplo, la estabilidad límite puede agrandarse hasta $\eta < 2{,}7$. Para una aplicación real podemos escoger $\eta = 1$, El *Teorema 2* asegura que el error de modelado sea estable. Si queremos acelerar el proceso de aprendizaje, podemos escoger un η un poco mayor, tal vez el error de identificación también sea estable.

La neuro identificación discutida es en-línea, no estudiamos la convergencia de los parámetros, nos interesamos por el error de identificación $e(k)$. Los parámetros no convergen a algunas constantes o valores óptimos, ver figura 4.4.

Definamos ahora el error medio cuadrático para tiempo finito

$$J_1(N) = \frac{1}{2N}\sum_{k=1}^{N} e^2(k) \tag{4.24}$$

En la fase de aprendizaje $J_1(300) = 0{,}0016$, en la fase de prueba $J_1(300) = 0{,}0018$. Como en el remark 5, el algoritmo de identificación en-línea no puede hacer converger los pesos a sus valores óptimo después de un cierto tiempo de aprendizaje. Aún para esta simple función no lineal los resultados de prueba no son muy alentadores.

1) En este ejemplo encontramos que el límite de estabilidad para η está cerca de 2. Este límite se puede cambiar con parámetros modelo tales como la condición inicial de W_{x_1,x_2} el número de reglas difusas, *etc.*. Aunque$1 < \eta < 2$ puede acelerar el proceso de entrenamiento no podemos garantizar la estabilidad para cualquier condición y para todo el proceso de aprendizaje. El Teorema 2 asegura que error en el período de modelado es estable para cualquier condición cuando $0 < \eta \leq 1$.

Los errores de modelado $\mu(k)$ en (4.5) y $\zeta(k)$ en (5.19) dependen de la complejidad del modelo particular escogido y que tan cerca está de la planta. En este ejemplo, si escogemos $l_1 = l_2 = 8$ todas las otras condiciones no cambian. El resultado del aprendizaje se muestra en la figura 4.4. El error de identificación se agranda a $J_1(300) = 0{,}0021$, el peor resultado debido a las reglas difusas redundantes. Desde un punto de vista de identificación es porque el modelo no está cercano a la planta, debemos mencionar que la estructura del modelo influencia al error de modelado pero no destruye la estabilidad del proceso de identificación.

2) A continuación se muestran los resultados al usando 6 reglas difusas (figura 4.6)con velocidad

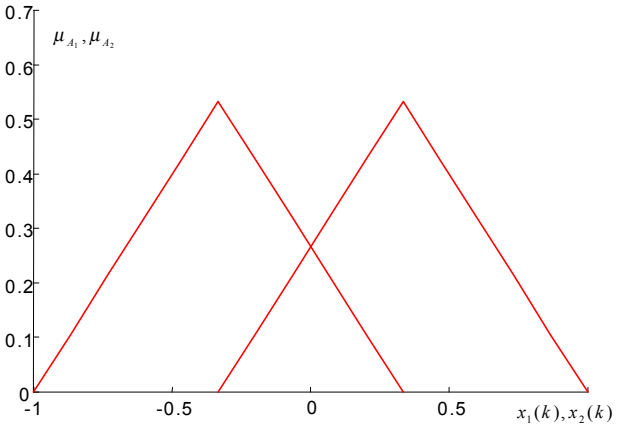

Figura 4.2: Funciones de membresía

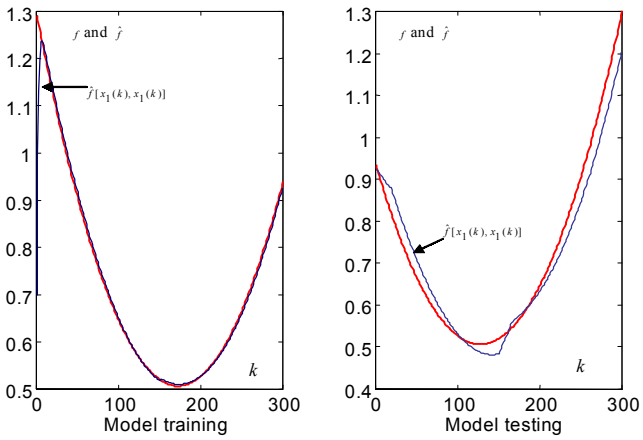

Figura 4.3: Función de aproximación

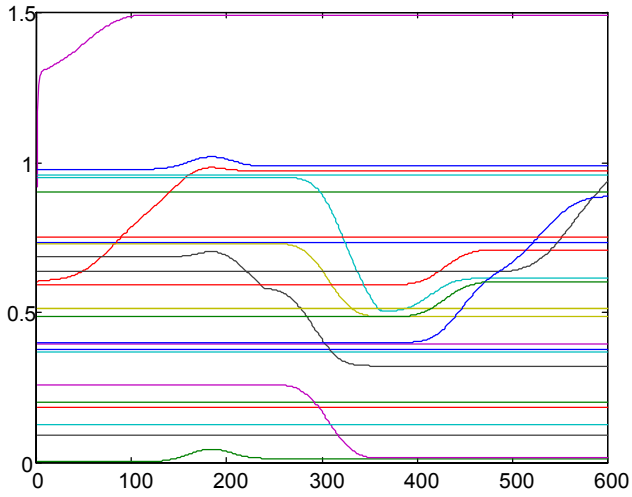

Figura 4.4: Pesos

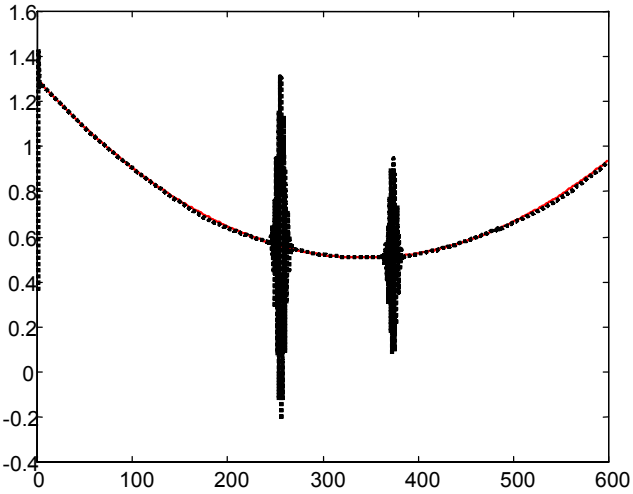

Figura 4.5: Inestable

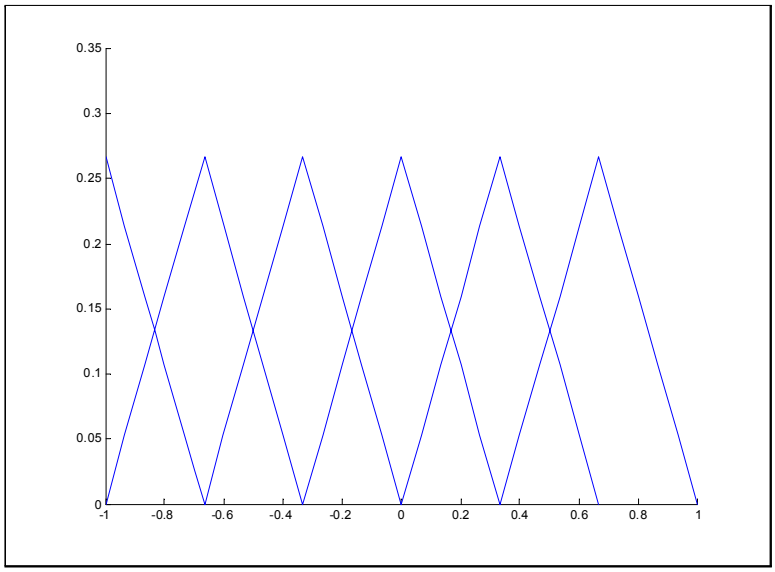

Figura 4.6: Funciones de membresía

de aprendizaje de 0,5 el seguimiento no es muy bueno, si aumentamos la velocidad de aprendizaje se obtienen mejores resultados, podemos ver que con velocidad de aprendizaje de 3 se obtiene un buen resultado como se puede ver en la figura 4.8 y si lo aumentamos todavía más se mantiene la estabilidad aunque afecta en el tiempo que tarda en seguir a la planta (figura 4.9).3) Usando ahora 11 funciones de membresía (figura 4.10) podemos observar que ya no se mejora mucho (ver figura 4.11), por lo que no siempre es bueno aumentar el número de reglas difusas, se puede obtener el mismo resultado con menos funciones de membresía y en un tiempo menor.

4) Usando las mismas condiciones que para la función (4.20) usaremos el mismo algoritmo para aproximar el sistema no lineal

$$y(k) = 2 * sin(6{,}28 * x_1) * x_2$$

y al usar 11 reglas difusas (figura 4.10) se obtuvieron los siguientes resultados:

Con una velocidad de aprendizaje baja no se obtuvieron resultados tan buenos como la ecuación anterior, al aumentar la velocidad de aprendizaje se volvía más lento el proceso de identificación, pero se mantenía la estabilidad como se muestra en las figuras 4.12 y 4.13

pero al usar 2 funciones de membresía el algoritmo estable logra una aproximación muy buena

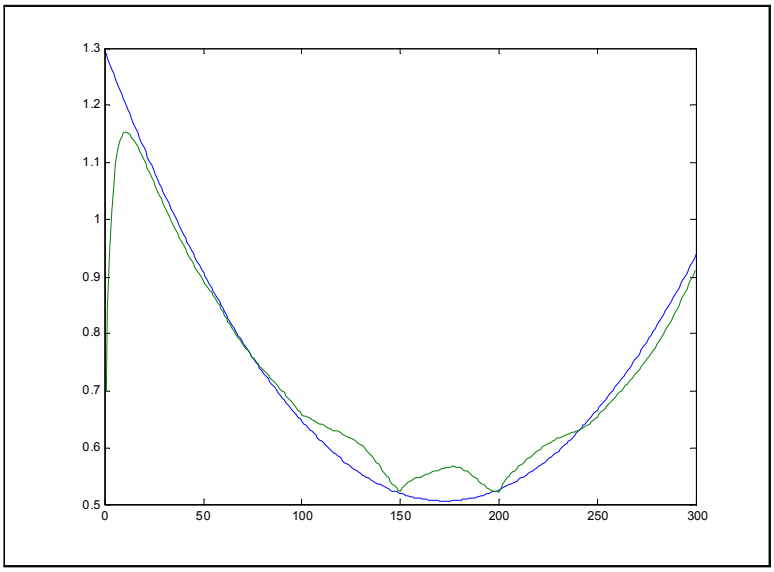

Figura 4.7: 6 reglas difusas con velocidad de aprendizaje de 0.5

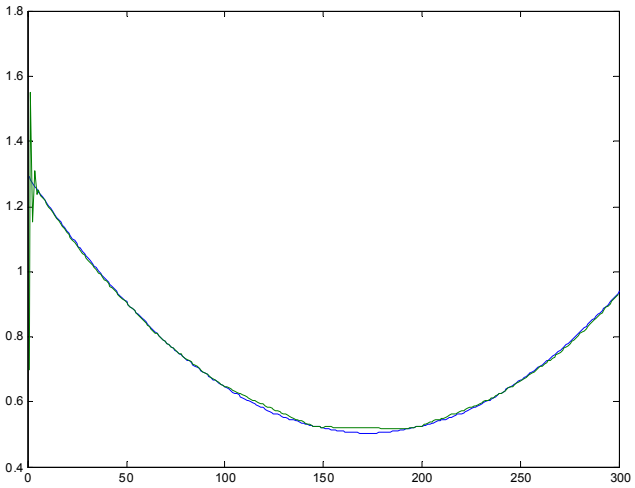

Figura 4.8: 6 reglas difusas con velocidad de aprendizaje de 3

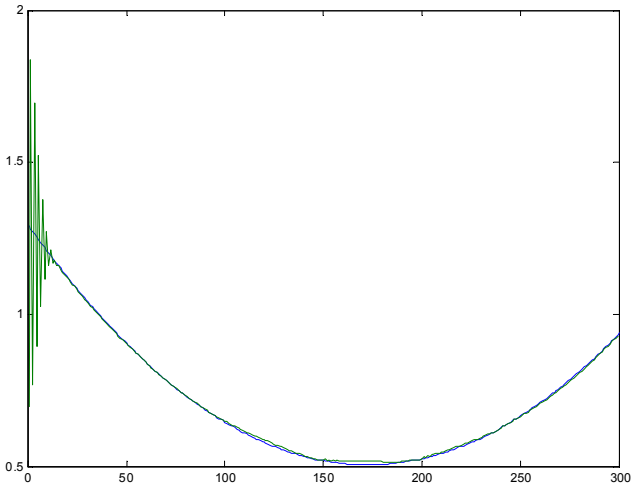

Figura 4.9: 6 reglas difusas con velocidad de aprendizaje de 4

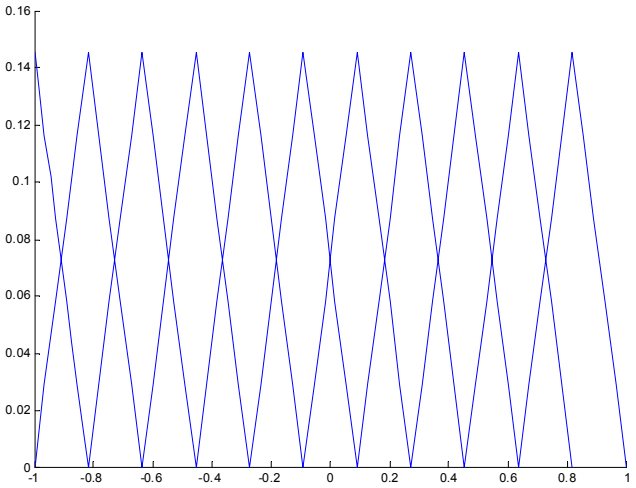

Figura 4.10: Funciones de membresía

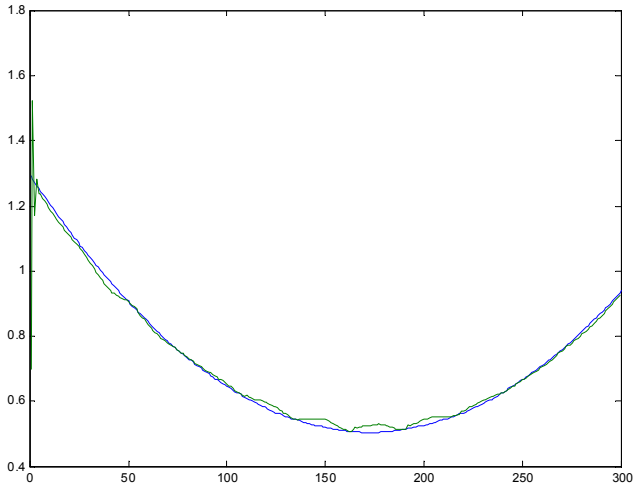

Figura 4.11: 11 reglas difusas con velocidad de aprendizaje de 3

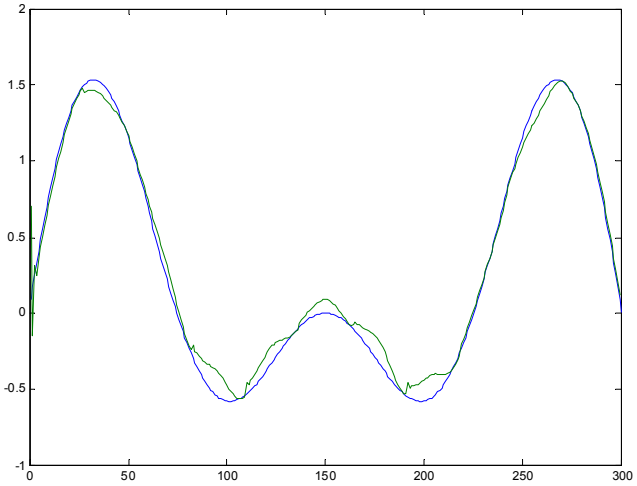

Figura 4.12: 11 reglas difusas con velocidad de aprendizaje de 3

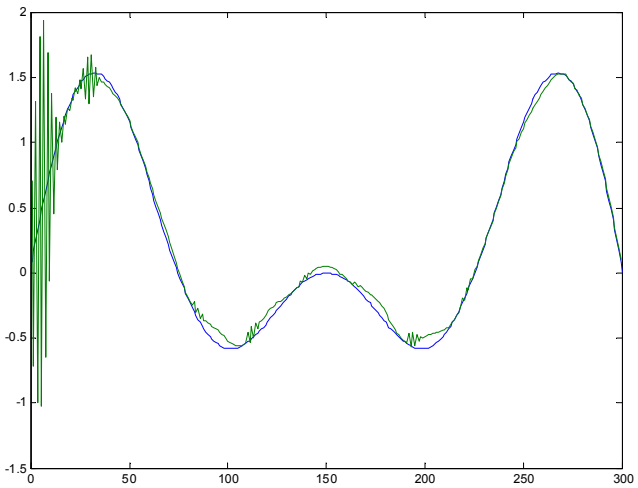

Figura 4.13: 11 reglas difusas con velocidad de aprendizaje de 5

(ver figura 4.14), una vez más observamos que al aumentar el número de reglas difusas no mejoramos la aproximación, sino que la hacemos más lenta y torpe debido a las reglas difusas redundantes

aún con 6 reglas (ver figura 4.15) difusas la aproximación no es tan buena como con 2, por lo que podemos ver que el algoritmo es estable y no requiere de muchas reglas difusas para funcionar bien por lo que no tiene ningún problema para usarse en-línea

4.3.2. Identificación de sistema no lineal

Primero usaremos un sistema no lineal para ilustrar el aprendizaje de gradiente descendiente (4.7), el algoritmo BP (5.20) y la estabilidad. La planta a identificar es el *Ejemplo 2* de [17] el cual también fue discutido en [10][16][21].

$$y(k) = \frac{y(k-1)\,y(k-2)\,[y(k-1)+2{,}5]}{1+y(k-1)^2+y(k-2)^2} + u(k-1) \tag{4.25}$$

La señal de entrada se escoge como

$$\sin u(k) = (\frac{2\pi}{25}k) \tag{4.26}$$

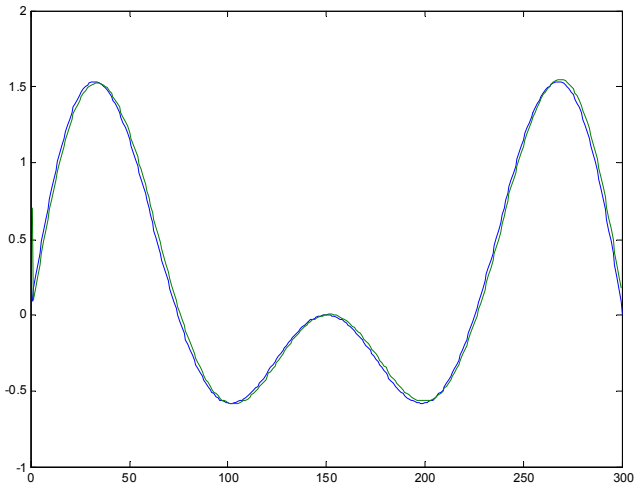

Figura 4.14: 2 reglas difusas con velocidad de aprendizaje de 2

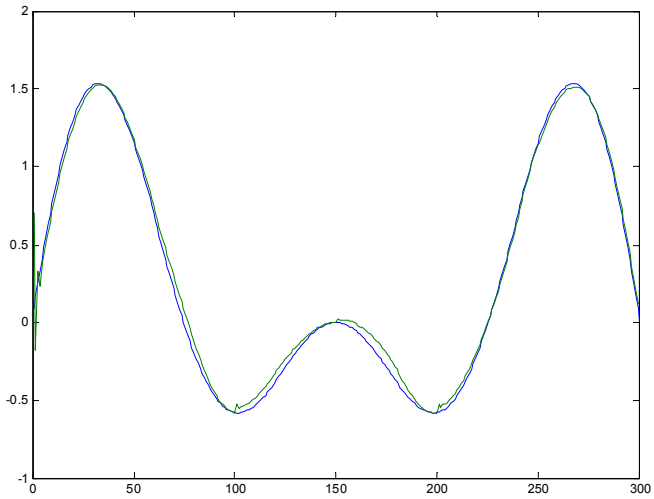

Figura 4.15:

Usaremos una red difuso neuronal (5.8) para identificar (4.25),

$$X(k) = [y(k-1), y(k-2), u(k-1)]^T \tag{4.27}$$

Para evitar la sobrecarga computacional, Usamos conexión simple, *i.e.*, la entrada por ϕ_k es solamente $A_{1k},\ A_{2k}, \cdots A_{nk}\ (k = 1 \cdots l)$. Entonces $l = 10,\ m = 1$. Las funciones de membresía se escogen como funciones Gaussianas

$$\mu_{A_{ji}}(k) = \exp\left(-\frac{(x_j - c_{ji})^2}{\sigma_{ji}^2}\right) \tag{4.28}$$

· Primero, asumiremos que se conoce la función de membresía A_{ji}. Escogeremos que el centro c_{ji} y el ancho σ_{ji} son constantes aleatoria entre $[0, 1]$. Las condiciones iniciales para W_k son aleatorias de $[0, 1]$. $\eta = 9{,}5$. Tal vez exista un mejor η, pero la simulación se tiene que hacer muchas veces para encontrar un buen η.

La complejidad del modelo es importante en el contexto de identificación de sistema, el cuál corresponde al número de reglas difusas del modelo difuso neuronal. En esta simulación tratamos de probar un diferente número de reglas difusas, encontramos que después de que el número de reglas difusas es 20, la precisión de identificación no mejora mucho. El resultado de la identificación se muestra en la figura 4.16.

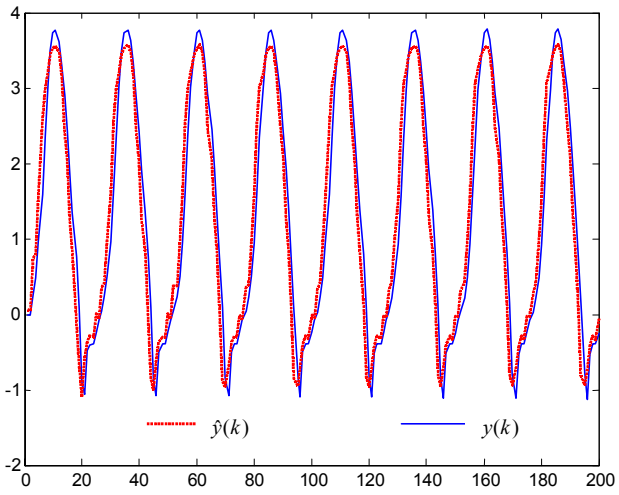

Figura 4.16: 10 reglas difusas con velocidad de aprendizaje de 1

Capítulo 5

Modelado difuso neuronal con aprendizaje de función de membresía premisa estable

5.1. Modelado difuso neuronal sin modificación robusta

5.1.1. Redes difuso neuronales Tipo Mamdani

Un esquema de inferencia difusa, el cual se basa en el modo generalizado modus Ponens, se puede describir esquemáticamente de la siguiente forma. Aquí consideramos sistemas difusos multiples-entradas-una-salida (MISO multi-input-single-output), $f : \Re^n \to \Re$, porque un sistema multi-salidas se puede siempre descomponer en una colección de sistemas única-salida.

$$\mathrm{R}^i\text{: SI } x_1 \text{ es } A_{1i} \text{ y } x_2 \text{ es } A_{2i} \text{ y } \cdots \ x_n \text{ es } A_{ni} \text{ ENTONCES } \widehat{y}_1 \text{ es } B_{1i} \text{ y} \cdots \widehat{y}_m \text{ es } B_{mi} \tag{5.1}$$

Usamos $l(i = 1, 2 \cdots l)$ reglas difusas SI-ENTONCES para desarrollar un mapeo de un vector de entrada lingüística $X = [x_1 \cdots x_n] \in \Re^n$ a un vector de salida lingüística $\widehat{Y}(k) = [\widehat{y}_1 \cdots \widehat{y}_m]^T$. $A_{1i}, \cdots A_{ni}$ y $B_{1i}, \cdots B_{mi}$ son conjuntos difusos standard [22]. Para una variable de entrada x_i hay l_i conjuntos difusos. En el caso de una conexión completa, $l = l_1 \times l_2 \times \cdots l_n$. De [22] sabemos que, al usar el difusificador inferencia de productos, difusificador centro-promedio y función de activación

singleton, la p-ésima salida del sistema lógico difuso puede ser expresado como

$$\begin{aligned}\widehat{y}_p &= \left(\sum_{i=1}^{l} w_{pi}\left[\prod_{j=1}^{n}\mu_{A_{ji}}\right]\right) / \left(\sum_{i=1}^{l}\left[\prod_{j=1}^{n}\mu_{A_{ji}}\right]\right) \\ &= \sum_{i=1}^{l} w_{pi}\phi_i\end{aligned} \tag{5.2}$$

donde $\mu_{A_{ji}}$ es la función de membresía del conjunto difuso A_{ji}, w_{pi} es el punto al cuál $\mu_{B_{pi}} = 1$. Si definimos

$$\phi_i = \prod_{j=1}^{n}\mu_{A_{ji}} / \sum_{i=1}^{l}\prod_{j=1}^{n}\mu_{A_{ji}} \tag{5.3}$$

(5.2) puede ser expresado en la forma matricial

$$\widehat{Y}(k) = W(k)\,\Phi[X(k)] \tag{5.4}$$

donde el parámetro $W(k) = \begin{bmatrix} w_{11} & & w_{1l} \\ & \ddots & \\ w_{m1} & & w_{ml} \end{bmatrix}$, y el vector de datos $\Phi[X(k)] = [\phi_1 \cdots \phi_l]^T$. La estructura del sistema difuso neuronal se muestra en la figura 4.1. Esta red difuso neuronal de cuatro capas se ha discutido en muchos artículos [7][11][13][21]. La capa I acepta vectores lingüísticos $X(k)$. Cada nodo de la capa II representa el valor de la función de membresía de la variable lingüística. Los nodos de la capa III representan reglas difusas. La capa IV es la capa de salida, las conexiones entre las capas III y IV son completamente conectadas por la matriz de pesos $W(k)$. Las capas I y II son la parte premisa. Las capas III y IV son la parte consecuencia.

5.1.2. Redes difuso neuronales Tipo Takagi-Sugeno-Kang

$$\mathrm{R}^i\text{: SI } x_1 \text{ es } A_{1i} \text{ y } x_2 \text{ es } A_{2i} \text{ y } \cdots \text{ y } x_n \text{ es } A_{ni} \text{ ENTONCES } \widehat{y}_j = p^i_{j0} + p^i_{j1}x_1 + \cdots p^i_{jn}x_n \tag{5.5}$$

donde $j = 1 \cdots m$. La p-ésima salida del sistema lógico difuso puede ser expresada como

$$\widehat{y}_p = \sum_{i=1}^{l}\left(p^i_{p0} + p^i_{p1}x_1 + \cdots p^i_{pn}x_n\right)\phi_i \tag{5.6}$$

donde ϕ_i es definido como en (5.3). (5.6) puede también ser representado en la forma de tipo-Mamdani (5.4),

$$\widehat{Y}(k) = W(k)\,\sigma[x(k)]$$

donde $\widehat{Y}(k) = [\widehat{y}_1 \cdots \widehat{y}_m]^T$

$$W(k) = \begin{bmatrix} p_{10}^1 \cdots p_{10}^l & p_{11}^1 \cdots p_{11}^l & \cdots & p_{1n}^1 \cdots p_{1n}^l \\ \vdots & \vdots & & \vdots \\ p_{m0}^1 \cdots p_{m0}^l & p_{m1}^1 \cdots p_{m1}^l & \cdots & p_{mn}^1 \cdots p_{mn}^l \end{bmatrix}$$

$$\sigma[x(k)] = \begin{bmatrix} \phi_1 \cdots \phi_l & x_1\phi_1 \cdots x_1\phi_l & \cdots & x_n\phi_1 \cdots x_n\phi_l \end{bmatrix}^T$$

Usaremos funciones de membresía Gaussianas para identificar reglas difusas, las cuales se definen como

$$\mu_{A_{ji}} = \exp\left(-\frac{(x_j - c_{ji})^2}{\sigma_{ji}^2}\right) \tag{5.7}$$

la $q-$*ésima* salida del modelo difuso neuronal puede expresarse como

$$\widehat{y}_q = \sum_{i=1}^{l} w_{qi} \prod_{j=1}^{n} \exp\left(-\frac{(x_j - c_{ji})^2}{\sigma_{ji}^2}\right) / \left[\sum_{i=1}^{l}\prod_{j=1}^{n} \exp\left(-\frac{(x_j - c_{ji})^2}{\sigma_{ji}^2}\right)\right] \tag{5.8}$$

Definamos

$$z_i = \prod_{j=1}^{n} \exp\left(-\frac{(x_j - c_{ji})^2}{\sigma_{ji}^2}\right), \qquad a_q = \sum_{i=1}^{l} w_{qi} z_i, \qquad b = \sum_{i=1}^{l} z_i \tag{5.9}$$

Entonces

$$\widehat{y}_q = \frac{a_q}{b} \tag{5.10}$$

Similar a (4.5), el proceso de identificación no lineal (1.7) puede representarse como

$$y_q = \sum_{i=1}^{l} w_{qi}^* \prod_{j=1}^{n} \exp\left(-\frac{\left(x_j - c_{ji}^*\right)^2}{\sigma_{ji}^{*2}}\right) / \left[\sum_{i=1}^{l}\prod_{j=1}^{n} \exp\left(-\frac{\left(x_j - c_{ji}^*\right)^2}{\sigma_{ji}^{*2}}\right)\right] - \mu_q \tag{5.11}$$

donde w_{qi}^*, c_{ji}^* y σ_{ji}^{*2} son parámetros desconocidos los cuales pueden minimizar la dinámica no modelada μ_q.

En el caso de tres variables independientes, la fórmula de Taylor tiene una función suave f como

$$f\left(x_1,x_2,x_3\right)=\sum_{k=0}^{l-1}\frac{1}{k!}\left[\left(x_1-x_1^0\right)\frac{\partial}{\partial x_1}+\left(x_2-x_2^0\right)\frac{\partial}{\partial x_2}+\left(x_3-x_3^0\right)\frac{\partial}{\partial x_3}\right]_0^k f+R_l \tag{5.12}$$

donde R_l es lo que sobra de la fórmula de Taylor. Si x_1,x_2,x_3 corresponder a w_{pi}^*, c_{ji}^* y σ_{ji}^{*2}, x_1^0,x_2^0,x_3^0 corresponde w_{pi}, c_{ji} y σ_{ji}^2,

$$y_q+\mu_q=\widehat{y}_q+\sum_{i=1}^{l}\left(w_{qi}^*-w_{qi}\right)z_i/b+\sum_{i=1}^{l}\sum_{j=1}^{n}\frac{\partial}{\partial c_{ji}}\left(\frac{a_q}{b}\right)\left(c_{ji}^*-c_{ji}\right)+\sum_{i=1}^{l}\sum_{j=1}^{n}\frac{\partial}{\partial \sigma_{ji}}\left(\frac{a_q}{b}\right)\left(\sigma_{ji}^*-\sigma_{ji}\right)+R_{1q} \tag{5.13}$$

Usando la regla de la cadena tenemos

$$\begin{array}{l}\frac{\partial}{\partial c_{ji}}\left(\frac{a_q}{b}\right)=\frac{\partial}{\partial z_i}\left(\frac{a_q}{b}\right)\frac{\partial z_i}{\partial c_{ji}}=\left(\frac{1}{b}\frac{\partial a_q}{\partial z_i}+\frac{\partial}{\partial z_i}\left(\frac{1}{b}\right)a_q\right)\left(2z_i\frac{x_j-c_{ji}}{\sigma_{ji}^2}\right)\\ =\left(\frac{w_{qi}}{b}-\frac{a_q}{b^2}\right)\left(2z_i\frac{x_j-c_{ji}}{\sigma_{ji}^2}\right)=2z_i\frac{w_{qi}-\widehat{y}_q}{b}\frac{x_j-c_{ji}}{\sigma_{ji}^2}\end{array} \tag{5.14}$$

$$\frac{\partial}{\partial \sigma_{ji}}\left(\frac{a_q}{b}\right)=\frac{\partial}{\partial z_i}\left(\frac{a_q}{b}\right)\frac{\partial z_i}{\partial \sigma_{ji}}=2z_i\frac{w_{qi}-\widehat{y}_q}{b}\frac{\left(x_j-c_{ji}\right)^2}{\sigma_{ji}^3} \tag{5.15}$$

En forma matricial

$$y_q+\mu_q=\widehat{y}_q-\widetilde{W}_qZ\left(k\right)-D_{Zq}\overline{C}_kE-D_{Zq}\overline{B}_kE \tag{5.16}$$

donde

$$\begin{array}{c}Z\left(k\right)=\left[z_1/b\cdots z_l/b\right]^T,\qquad W_q=\left[w_{q1}\cdots w_{ql}\right],\qquad \widetilde{W}_q=W_q-W_q^*\\ D_{Zq}=\left[2z_1\frac{w_{q1}-\widehat{y}_q}{b},\cdots,2z_l\frac{w_{ql}-\widehat{y}_q}{b}\right],\qquad E=\left[1,\cdots 1\right]^T\\ \overline{C}_k=\begin{bmatrix}\frac{x_1-c_{11}}{\sigma_{11}^2}\left(c_{11}-c_{11}^*\right) & & \frac{x_n-c_{n1}}{\sigma_{n1}^2}\left(c_{n1}-c_{n1}^*\right)\\ & \ddots & \\ \frac{x_1-c_{1l}}{\sigma_{1l}^2}\left(c_{1l}-c_{1l}^*\right) & & \frac{x_n-c_{nl}}{\sigma_{nl}^2}\left(c_{nl}-c_{nl}^*\right)\end{bmatrix}\\ \overline{B}_k=\begin{bmatrix}\frac{\left(x_1-c_{11}\right)^2}{\sigma_{11}^3}\left(\sigma_{11}-\sigma_{11}^*\right) & & \frac{\left(x_n-c_{n1}\right)^2}{\sigma_{n1}^3}\left(\sigma_{n1}-\sigma_{n1}^*\right)\\ & \ddots & \\ \frac{\left(x_1-c_{1l}\right)^2}{\sigma_{1l}^3}\left(\sigma_{1l}-\sigma_{1l}^*\right) & & \frac{\left(x_n-c_{nl}\right)^2}{\sigma_{nl}^3}\left(\sigma_{nl}-\sigma_{nl}^*\right)\end{bmatrix}\end{array} \tag{5.17}$$

Definamos el error de identificación como

$$\begin{array}{c}e_q=\widehat{y}_q-y_q\\ e_q=Z\left(k\right)\widetilde{W}_q+D_{Zq}\overline{C}_kE+D_{Zq}\overline{B}_kE+\mu_q-R_{1q}\end{array} \tag{5.18}$$

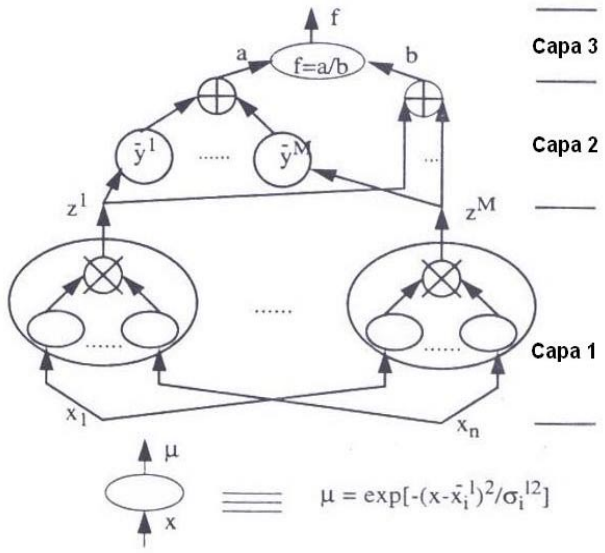

Figura 5.1: Representación de la red del sistema difuso

donde R_{1q} es el error de aproximación de segundo orden de la serie de Taylor, $q = 1 \cdots m$. Definamos

$$\begin{aligned} e(k) &= [e_1 \cdots e_m]^T \\ e(k) &= \widetilde{W}_k Z(k) + D_z(k) \overline{C}_k E + D_z(k) \overline{B}_k E + \zeta(k) \end{aligned} \tag{5.19}$$

donde $\widetilde{W}_k = \begin{bmatrix} w_{11} - w_{11}^* & & w_{m1} - w_{m1}^* \\ & \ddots & \\ w_{1l} - w_{1l}^* & & w_{ml} - w_{ml}^* \end{bmatrix}$, $D_z(k) = \begin{bmatrix} 2z_1 \frac{w_{11} - \widehat{y}_1}{b} & & 2z_l \frac{w_{1l} - \widehat{y}_1}{b} \\ & \ddots & \\ 2z_1 \frac{w_{m1} - \widehat{y}_m}{b} & & 2z_l \frac{w_{ml} - \widehat{y}_m}{b} \end{bmatrix}$. $\zeta(k) = \mu - R_1$, $\mu = [\mu_1 \cdots \mu_m]^T$, $R_1 = [R_{11} \cdots R_{1m}]^T$.

Por la cota de la función Gaussiana ϕ podemos asumir que μ en (5.11) es acotada, también R_1 es acotada. Entonces $\zeta(k)$ en (5.19) es acotada. El siguiente teorema da un algoritmo BP estable para redes difuso neuronales en tiempo discreto.

Teorema 5.1 *Si usamos una red difuso neuronal (5.8) para identificar una planta no lineal (1.7),*

el siguiente algoritmo de BP puede identificar el error $e(k)$ *acotado*

$$\begin{aligned} W_{k+1} &= W_k - \eta_k e(k) Z(k)^T \\ c_{ji}(k+1) &= c_{ji}(k) - 2\eta_k z_i \frac{w_{pi}-\widehat{y}_p}{b} \frac{x_j - c_{ji}}{\sigma_{ji}^2} (\widehat{y}_q - y_q) \\ \sigma_{ji}(k+1) &= \sigma_{ji}(k) - 2\eta_k z_i \frac{w_{pi}-\widehat{y}_p}{b} \frac{(x_j - c_{ji})^2}{\sigma_{ji}^3} (\widehat{y}_q - y_q) \end{aligned} \tag{5.20}$$

donde $\eta_k = \dfrac{\eta}{1 + \|Z\|^2 + 2\|D_z\|^2}$, $0 < \eta \leq 1$. *El promedio del error de identificación satisface*

$$J = \lim_{T\to\infty} \sup \frac{1}{T} \sum_{k=1}^{T} e^2(k) \leq \frac{\eta}{\pi} \overline{\zeta} \tag{5.21}$$

donde $\pi = \dfrac{\eta}{1+\kappa}\left[1 - \dfrac{\kappa}{1+\kappa}\right] > 0$, $\kappa = \max_k \left(\|Z\|^2 + 2\|D_z\|^2\right)$, $\overline{\zeta} = \max_k \left[\zeta^2(k)\right]$

Demostración. Definamos $\widetilde{c}_{ji}(k) = c_{ji}(k) - c_{ji}^*$, $\widetilde{b}_{ji}(k) = \sigma_{ji}(k) - \sigma^*(k)$, el elemento de $\widetilde{C}_k$ es expresado como $\widetilde{c}_{ji}(k) = \left[\widetilde{C}_k\right]$. Entonces

$$\left[\widetilde{C}_{k+1}\right] = \left[\widetilde{C}_k\right] - 2\eta_k z_i \frac{w_{qi} - \widehat{y}_q}{b} \frac{x_j - c_{ji}}{\sigma_{ji}^2} (\widehat{y}_q - y_q) \tag{5.22}$$

Escogemos una matriz definida positiva L_k como

$$L_k = \left\|\widetilde{W}_k\right\|^2 + \left\|\widetilde{C}_k\right\|^2 + \left\|\widetilde{B}_k\right\|^2 \tag{5.23}$$

Por la ley de actualización (5.20), tenemos

$$\widetilde{W}_{k+1} = \widetilde{W}_k - \eta_k e(k) Z(k)^T \tag{5.24}$$

Usando (5.19) tenemos

$$\begin{aligned} \Delta L_k &= \left\|\widetilde{W}_k - \eta_k e(k) Z(k)^T\right\|^2 + \left\|\widetilde{C}_k - \left[2\eta_k z_i \frac{w_{qi}-\widehat{y}_q}{b} \frac{x_j - c_{ji}}{\sigma_{ji}^2} (\widehat{y}_q - y_q)\right]\right\|^2 \\ &+ \left\|\widetilde{B}_k - \left[2\eta_k z_i \frac{w_{qi}-\widehat{y}_q}{b} \frac{(x_j - c_{ji})^2}{\sigma_{ji}^3} (\widehat{y}_q - y_q)\right]\right\|^2 - \left\|\widetilde{W}_k\right\|^2 - \left\|\widetilde{C}_k\right\|^2 - \left\|\widetilde{B}_k\right\|^2 \\ &= \eta_k^2 e^2(k) \left(\left\|Z(k)^T\right\|^2 + 2\left\|D_z^T\right\|^2\right) - 2\eta_k \|e(k)\| \left\|\widetilde{W}_k Z(k)^T + D_z^T \overline{C}_k E + D_z^T \overline{B}_k E\right\| \\ &= \eta_k^2 e^2(k) \left(\|Z\|^2 + 2\|D_z\|^2\right) - 2\eta_k \|e(k)[e(k) - \zeta(k)]\| \\ &\leq -\eta_k e^2(k) \left[1 - \eta_k \left(\|Z\|^2 + 2\|D_z\|^2\right)\right] + \eta\zeta^2(k) \\ &\leq -\pi e^2(k) + \eta\zeta^2(k) \end{aligned} \tag{5.25}$$

donde π se define como

$$\pi = \frac{\eta}{1 + \max\limits_{k} \left(\|Z\|^2 + 2 \|D_z\|^2 \right)} \tag{5.26}$$

Porque

$$n \left[\text{mín} \left(\widetilde{w}_i^2 \right) + \text{mín} \left(\widetilde{c}_{ji}^2 \right) + \text{mín} \left(\widetilde{b}_{ji}^2 \right) \right] \leq L_k \leq n \left[\text{máx} \left(\widetilde{w}_i^2 \right) + \text{máx} \left(\widetilde{c}_{ji}^2 \right) + \text{máx} \left(\widetilde{b}_{ji}^2 \right) \right] \tag{5.27}$$

donde $n \left[\text{mín} \left(\widetilde{w}_i^2 \right) + \text{mín} \left(\widetilde{c}_{ji}^2 \right) + \text{mín} \left(\widetilde{b}_{ji}^2 \right) \right]$ y $n \left[\text{máx} \left(\widetilde{w}_i^2 \right) + \text{máx} \left(\widetilde{c}_{ji}^2 \right) + \text{máx} \left(\widetilde{b}_{ji}^2 \right) \right]$ son funciones-$\mathcal{K}_\infty$, y $\pi e^2(k)$ es una función-$\mathcal{K}_\infty$, $\eta \zeta^2(k)$ es una función-$\mathcal{K}$. De (5.19) y (5.23) sabemos que V_k es la función de $e(k)$ y $\zeta(k)$, entonces L_k admite una función suave ISS-Lyapunov como está definido en *Definición 2*. Del *Teorema 1*, la dinámica del error de identificación es entrada-a-salida estable. ¨Porque la "ENTRADA" $\zeta(k)$ es acotada y la dinámica es ISS, el "ESTADO" $e(k)$ es acotado.

(5.25) puede ser reescrito como

$$\Delta L_k \leq -\pi e^2(k) + \eta \zeta^2(k) \leq \pi e^2(k) + \eta \overline{\zeta} \tag{5.28}$$

Resumiendo (5.28) de 1 hasta T, y usando $L_T > 0$ y L_1 constante, obtenemos

$$\begin{array}{c} L_T - L_1 \leq -\pi \sum_{k=1}^{T} e^2(k) + T\eta\overline{\zeta} \\ \pi \sum_{k=1}^{T} e^2(k) \leq L_1 - L_T + T\eta\overline{\zeta} \leq L_1 + T\eta\overline{\zeta} \end{array} \tag{5.29}$$

(5.21) es establecida. ■

Comentario 5.1 *Para modelos difuso neuronales tipo Takagi-Sugeno-Kang (5.5) con funciones de membresía Gaussianas en A_{ji}, la $q-$ésima salida del sistema lógico difuso puede ser expresado como*

$$\widehat{y}_q = \sum_{i=1}^{l} \left(\sum_{k=0}^{n} p_{qk}^i x_k \right) \prod_{j=1}^{n} \exp \left(-\frac{(x_j - c_{ji})^2}{\sigma_{ji}^2} \right) / \left[\sum_{i=1}^{l} \prod_{j=1}^{n} \exp \left(-\frac{(x_j - c_{ji})^2}{\sigma_{ji}^2} \right) \right] \tag{5.30}$$

donde $x_0 = 1$. La parte $\sum\limits_{i=1}^{l} \left(w_{qi}^* - w_{qi} \right) z_i / b_q$ (5.13) es cambiada a

$$\sum_{i=1}^{l} \left(\sum_{k=0}^{n} \left(p_{qk}^{i*} - p_{qk}^i \right) x_k \right) z_i / b \tag{5.31}$$

Si usamos el siguiente algoritmo

$$\begin{aligned} p_{qk}^{i}(k+1) &= p_{qk}^{i}(k) - \eta_k (\widehat{y}_q - y_q) \tfrac{z_i}{b} x_k \\ c_{ji}(k+1) &= c_{ji}(k) - 2\eta_k z_i \tfrac{w_{pi}-\widehat{y}_p}{b} \tfrac{x_j - c_{ji}}{\sigma_{ji}^2} (\widehat{y}_q - y_q) \\ \sigma_{ji}(k+1) &= \sigma_{ji}(k) - 2\eta_k z_i \tfrac{w_{pi}-\widehat{y}_p}{b} \tfrac{(x_j - c_{ji})^2}{\sigma_{ji}^3} (\widehat{y}_q - y_q) \end{aligned} \tag{5.32}$$

El error de identificación $e(k)$ es acotado. La prueba es igual a la del Teorema 3.

Comentario 5.2 *Normalizando las velocidades de aprendizaje η_k en (4.7) y (5.20) son tiempo-variables para asegurar la estabilidad del error de identificación. Estas ganancias de aprendizaje son más fáciles de decidir que [21], y no se requiere ninguna información a priori, por ejemplo podemos escoger $\eta = 1$. La contradicción en la rápida convergencia y el aprendizaje estable se puede evitar. Se pueden encontrar velocidades de aprendizaje de tiempo-variable en algunos temas standard adaptables [8]. Pero también usaron modificación robusta para garantizar la estabilidad de procesos de identificación.*

Comentario 5.3 *Si escogemos η como una función zona-muerta:*

$$\begin{cases} \eta = 0 & Si \ |e(k)| \leq \overline{\zeta} \\ \eta = \eta_0 & Si \ |e(k)| > \overline{\zeta} \end{cases} \tag{5.33}$$

(5.20) es la misma que en [22]. Su un término de modificación-σ o un término regla-δ modificado se agrega en $_k$ en (4.7) o (5.20), se vuelve la misma que en [24] o [12]. Pero todos ellos necesitan la cota máxima del error de modelado $\overline{\zeta}$. Y el error de identificación se agranda por la modificación robusta [8].

Comentario 5.4 *Como asumimos que las redes difuso neuronales no pueden seguir sistemas no lineales exactamente, no podemos hacer que los parámetros (w_{ji}, c_{ji} y σ_{ji}) convergan, nos gustaría simplemente forzar la salida de la red difuso neuronal para seguir a la planta de la salida de la planta, i.e. el error de identificación es estable. Sin embargo los pesos no pueden converger a sus valores óptimos, (5.21) muestra que el error de identificación convergerá al radio $\frac{\eta}{\pi}\overline{\zeta}$. Aunque la entrada esté persistentemente excitando, el error de modelado $\zeta(k)$ no hará converger a los pesos a sus valores óptimos. Es posible que el error de salida converga, pero los errores de los pesos son muy altos cuando las reglas difusas no son bien definidas. La relación del error de salida y el error*

de los pesos se muestra en (4.6) y (5.19). Un caso más simple es que usemos (4.6) y las redes difuso neuronales podrán seguir a la planta no lineal exactamente

$$\begin{aligned} &\textit{planta: } y = W^{*}\Phi\left[X\left(k\right)\right] \\ &\textit{redes difuso neuronales: } \widehat{y} = W\left(k\right)\Phi\left[X\left(k\right)\right] \\ &\textit{error de salida: } \left(y-\widehat{y}\right) = \left(W^{*}-W\left(k\right)\right)\Phi\left[X\left(k\right)\right] \end{aligned} \tag{5.34}$$

Si $\Phi\left[X\left(k\right)\right]$ *es muy grande, un error de salida pequeño* $\left(y-\widehat{y}\right)$ *no significa una buena convergencia del error de los pesos* $W^{*}-W\left(k\right)$.

5.2. Diseño de sistemas difusos usando el aprendizaje de gradiente descendiente

- **Paso 1. Determinación de la estructura y configuración de los parámetros iniciales.** Escójase el sistema difuso inicial en la forma de la figura y determinar M. Una M mayor resulta en más parámetros y en más cómputo requerido pero da una mejor aproximación de exactitud. Especificar los parámetros iniciales $y^{-1}\left(0\right)$, $x_i^{-1}\left(0\right)$ y $\sigma_i^{l}\left(0\right)$. Estos parámetros iniciales pueden ser determinados de acuerdo a las reglas lingüísticas de expertos humanos o pueden ser escogidos de manera que las correspondientes funciones de membresía cubran uniformemente los espacios de entrada y de salida. Para aplicaciones particulares se pueden usar métodos especiales.

- **Paso 2. Presentar la entrada y calcular la salida del sistema difuso.** Para un par entrada-salida dado $\left(x_0^{p}, y_0^{p}\right)$, $p = 1, 2, ...$ y al q-ésimo punto de aprendizaje $q = 0, 1, 2, ...$, presentar x_0^{p} a la capa de entrada del sistema difuso en la figura 5.1 y computar las salidas de las capas 1-3. Esto es

$$z^{l} = \prod_{i=1}^{n} \exp\left(-\left(\frac{x_{0i}^{p} - x_i^{-lp}\left(q\right)}{\sigma_i^{l}\left(q\right)}\right)^{2}\right) \tag{5.35}$$

$$b = \sum_{l=1}^{M} z^{l} \tag{5.36}$$

$$a = \sum_{l=1}^{M} y^{-1}(q)\, z^l \tag{5.37}$$

$$f = a/b \tag{5.38}$$

- **Paso 3.** Actualizar parámetros. Usar el algoritmo de aprendizaje para computar y actualizar los parámetros $y^{-1}(q+1)$, $x_i^{-1}(q+1)$ y $\sigma_i^l(q+1)$, donde $y = y_0^p$, y z^l, b, a y f igual a esos computados en el paso 2.

- **Paso 4.** Repetir desde el paso 2 con $q = q+1$ hasta que el error $|f - y_0^p|$ sea menor que un número preespecificado ϵ o hasta que q iguale a un número preespecificado.

- **Paso 5.** Repetir desde el paso 2 con $p = p+1$; estos es, actualizar los parámetros usando el siguiente par de entrada salida $\left(x_0^{p+1}, y_0^{p+1}\right)$.

- **Paso 6.** Si se es deseable y factible, hacer $p = 1$ y hacer los pasos 2-5 otra vez hasta que el sistema difuso diseñado sea satisfactorio. Para control en-línea e identificación de sistemas este paso no es factible porque los pares de entrada-salida se proveen uno-a-uno en tiempo-real. Para problemas de reconocimiento de patrones donde los pares de entrada-salida se proveen fuera-de-línea, este paso es comúnmente deseable.

5.3. Simulación

Identificación de sistemas no lineales.

1) Usamos un sistema no lineal para ilustrar el algoritmo BP (5.20). La planta identificada es el *Ejemplo 2* de [17] el cual también fue discutido por [10][16][21].

$$y(k) = \frac{y(k-1)\, y(k-2)\, [y(k-1) + 2{,}5]}{1 + y(k-1)^2 + y(k-2)^2} + u(k-1) \tag{5.39}$$

La señal de entrada se escoge como un número aleatorio en el intervalo $[0, 1]$. Usamos la red difuso neuronal (5.8) para identificar (4.25),

$$X(k) = [y(k-1), y(k-2), u(k-1)]^T$$

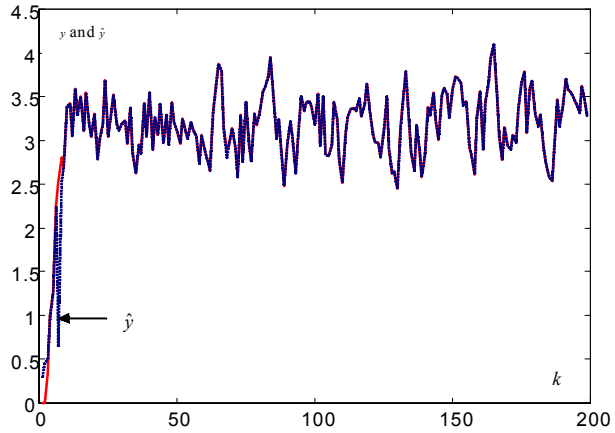

Figura 5.2: Identificación en-línea de un sistema no lineal

Para evitar la carga computacional usamos conexión singular, esto es, la entrada para ϕ_k es solo A_{1k}, $A_{2k}, \cdots A_{nk}$ $(k = 1 \cdots l)$, $l = 6$, $m = 1$. Las funciones de membresía se escogen como funciones Gaussianas

$$\mu_{A_{ji}}(k) = \exp\left(-\frac{(x_j - c_{ji})^2}{\sigma_{ji}^2}\right)$$

Asumimos que las funciones de membresía A_{ji} son desconocidas. Usamos (5.20) con $\eta = 1$ para actualizar W_k, c_{ji} y σ_{ji}. Las condiciones iniciales para W_k, c_{ji} y σ_{ji} son aleatorias de 0 a 1. Los resultados de la identificación en-línea se muestran en la figura 5.2.

Ahora comparamos nuestro algoritmo con un algoritmo normal BP [17] y aprendizaje óptimo [21]. Usamos la misma red neuronal multicapa como en [17], es $\Pi_{3,20,10,1}$ (dos capas ocultas con 20 y 10 nodos), y una velocidad de aprendizaje fijo $\eta = 0{,}05$. En esta simulación encontramos que después de $\eta > 0{,}1$ la red neuronal se vuelve inestable. También repetimos la simulación del *Ejemplo 3* de [21]. La comparación del desempeño se puede realizar por los errores medios cuadráticos (4.24). Los resultados comparados se muestran en la figura 5.3.

Podemos ver que el aprendizaje óptimo para redes difuso neuronales [21] es el mejor con respecto al error de identificación $J_1(200) = 0{,}0064$. Pero es difícil verlo porque tenemos que resolver una

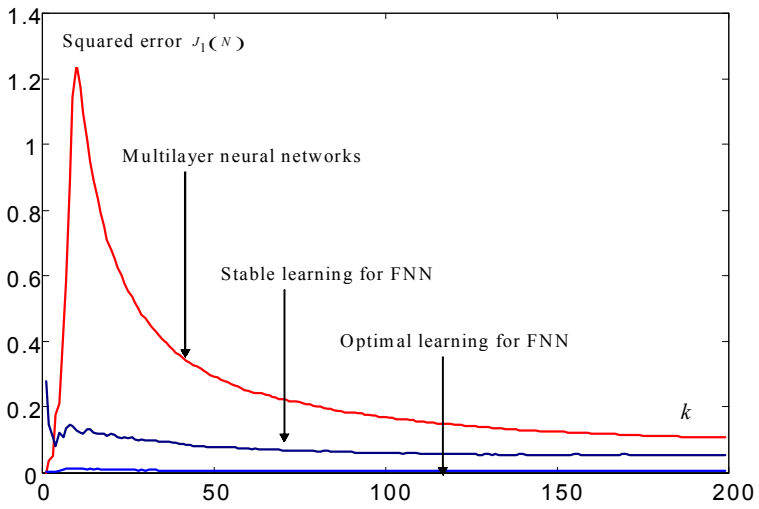

Figura 5.3: Comparación de desempeños

ecuación $A\beta^2 + B\beta = 0$ o usar algoritmos genéticos para encontrar la velocidad de aprendizaje óptima. El algoritmo estable propuesto en ésta tesis tiene casi la misma velocidad de convergencia que el aprendizaje óptimo. Aunque el error de identificación es mayor $J_1\,(200) = 0{,}051$ es simple y fácil de implantar. Algoritmos normales BP para redes neuronales multicapa tienen una velocidad de convergencia lenta y un error de identificación mayor $J_1\,(200) = 0{,}1078$.

2) Ahora veamos cómo se comporta el algoritmo al usar 20 reglas difusas y cómo se comporta una red neuronal con regla de aprendizaje Back Propagation.

En la figura 5.4 vemos que la red difusa neuronal con el algoritmo de aprendizaje estable logra una identificación muy buena, lo podemos constatar viendo el índice de desempeño mostrado en la figura 5.5

ahora veamos como se comporta la red neuronal usando 2 capas ocultas de 10 y 5 neuronas respectivamente, el resultado se puede ver en la figura 5.6. Podemos ver que la red difuso neuronal con algoritmo de aprendizaje estable hace un mucho mejor trabajo que la red neuronal.

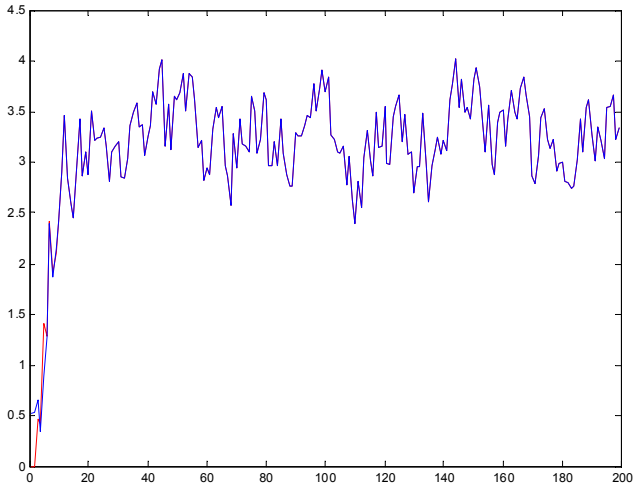

Figura 5.4: 20 reglas difusas con velocidad de aprendizaje de 1

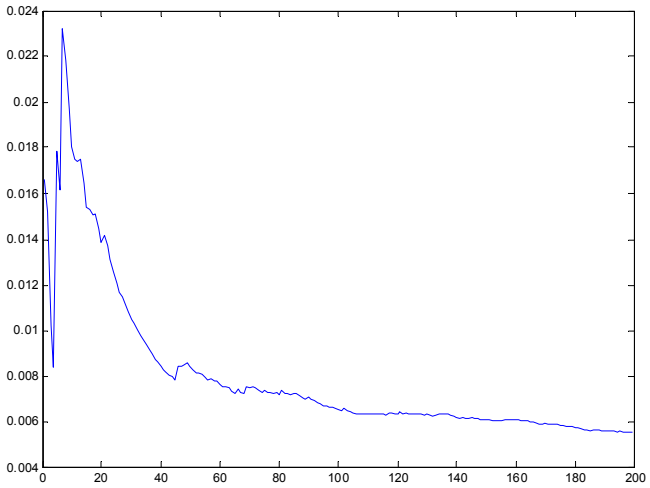

Figura 5.5: Indice de desempeño

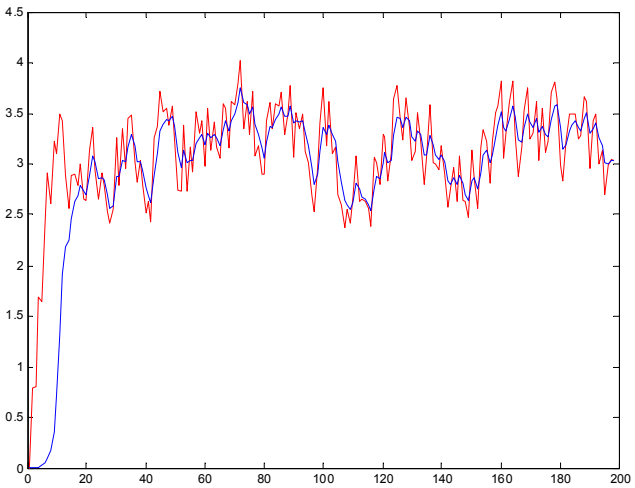

Figura 5.6: Identificación usando redes neuronales

3) Para la ecuación (5.39) usaremos ahora una entrada

$$u(k) = sin(6{,}28 * k/25)$$

todas las demás condiciones se mantienen iguales, usamos el mismo algoritmo de aprendizaje estable para actualizar W_k, c_{ji} y σ_{ji}. Para 20 reglas difusas usando una velocidad de aprendizaje de 5 el algoritmo tarda un poco en aproximarse como podemos ver en la figura 5.7, si se usa una velocidad de aprendizaje de 1 se aproxima más rápidamente como podemos ver en la figura 5.8

Si reducimos el número de reglas difusas a 2 el algoritmo de aprendizaje estable propuesto hace un mejor trabajo (figura 5.9), cabe notar que en ninguno de los casos se perdió la estabilidad aunque solo se haya asegurado para una velocidad de aprendizaje de 1.

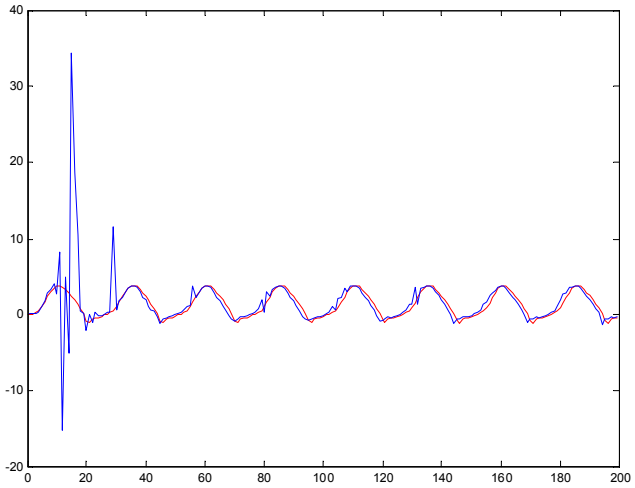

Figura 5.7: 20 reglas difusas con velocidad de aprendizaje de 5

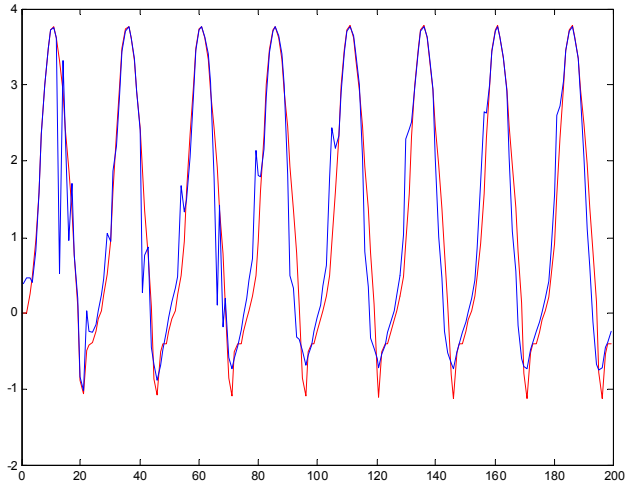

Figura 5.8: 20 reglas difusas con velocidad de aprendizaje de 1

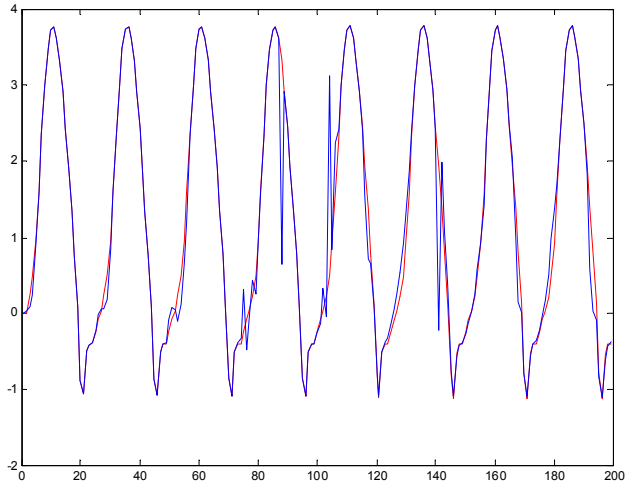

Figura 5.9: 2 reglas difusas con velocidad de aprendizaje de 1

Capítulo 6

Conclusión

En esta tesis se discutieron redes difuso neuronales tipo Mamdani y tipo Takagi-Sugeno-Kang. Se propuso una nueva ley de aprendizaje estable en la cuál usando el acercamiento de entrada-a-estado estable concluimos que las técnicas de robustificación normalmente usadas en el modelado difuso neuronal tales como proyección y zona-muerta no son necesarias para la ley de gradiente descendiente y el algoritmo BP. Dicha ley de aprendizaje estable se puede implantar para su uso en línea y es robusta ante cualquier incertidumbre acotada.

Podemos concluir que al usar una red difuso neuronal con algoritmo de aprendizaje estable llegamos a mejores aproximaciones que con los algoritmos comúnmente utilizados; es posible observar que la red difuso neuronal con algoritmo de aprendizaje estable tiene una mejor aproximación que una red neuronal con algoritmo de aprendizaje Back Propagation y tiene una carga computacional mucho menor. Se puede observar que con una mayor cantidad de reglas difusas no se obtiene una mejor aproximación, ésto es debido a la redundancia de reglas difusas que en vez de mejorar el resultado lo entorpece y a su vez lo vuelve más lento. El algoritmo de aprendizaje estable mantiene la estabilidad ante cualquier incertidumbre acotada e incluso puede funcionar con una velocidad de aprendizaje mayor a la que se propuso en el Teorema 2. Por su baja carga computacional se cumple el objetivo de usarse en-línea, se proponen como trabajos futuros el desarrollo de ésta técnica como controlador, así como su desarrollo como observador.

Bibliografía

[1] M.Brown, C.J.Harris, *Neurofuzzy Aadaptive Modelling and Control*, Prentice Hall: New York , 1994.

[2] W.C.Chan, C.W.Chan, K.C.Cheung and Y.Wang, Modeling of nonlinear stochastic dynamical systems using neurofuzzy networks, *38th IEEE Conference on Decision and Control*, 2643-2648, Phoenix, USA, 1999.

[3] D.S.Chen and R.C.Jain, A robust back propagation learning algorithm for fuunction approximation, *IEEE Trans. Neural Networks*, Vol.5, No.3, 1994.

[4] M.Y.Chen and D.A.Linkensm, A systematic neuro-fuzzy modeling framework with application to material property prediction, *IEEE Trans. Syst., Man, Cybern. B*, Vol.31, 781-790, 2001.

[5] B.Egardt, *Stability of Adaptive Controllers*, Lecture Notes in Control and Information Sciences, Vol.20, Springer-Verlag, Berlin, 1979

[6] S.Haykin, *Neural Networks- A Comprehensive Foundation*, Macmillan College Publ. Co., New York, 1994.

[7] S.I.Horikawa, T.Furuhashi and Y.Uchikawa, On fuzzy modeling using fuzzy neual networks with the back-propagation algorithm, *IEEE Trans. Neural Networks*, Vol.3, No.5, 801-806, 1992.

[8] P.A.Ioannou and J.Sun, *Robust Adaptive Control*, Prentice-Hall, Inc, Upper Saddle River: NJ, 1996.

[9] Z.P.Jiang and Y.Wang, Input-to-State Stability for Discrete-Time Nonlinear Systems, *Automatica*, Vol.37, No.2, 857-869, 2001.

[10] C.F.Juang, A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithm, *IEEE Trans. Fuzzy Syst.*, Vol.10, 155-170, 2002.

[11] Y.G.Leu, T.T.Lee and W.Y.Wang, Observer-based adaptive fuzzy-neural control for unknown nonlinear dynamical systems, *IEEE Trans. Syst., Man, Cybern. B*, Vol.29, 583-591, 1999.

[12] F.L.Lewis, A.Yesildirek and K.Liu, Multilayer Neural-Net Robot Controller with Guaranteed Tracking Performance, *IEEE Trans. Neural Networks*, Vol.7, No.2, 388-399, 1996.

[13] C.T.Lin and G.Lee, *Neural fuzzy systems: A neural-fuzzy synergism to intelligent systems*, Prentice-Hall Inc., NJ, 1996.

[14] C.T.Lin, A neual fuzzy control system with structure and parameter learning, *Fuzzy Sets anc Systems.*, Vol.70, 183-212, 1995.

[15] E.H.Mamdani, Application of fuzzy algorithms for control of simple dynamic plant, *IEE Proceedings - Control Theory and Applications*, Vol.121, No.12, 1585-1588, 1976.

[16] P.A.Mastorocostas and J.B.Theocharis, A recurrent fuzzy-neural model for dynamic system identification, *IEEE Trans. Syst., Man, Cybern. B*, Vol.32, 176-190, 2002.

[17] K.S.Narendra and S.Mukhopadhyay, Adaptive Control Using Neural Networks and Approximate Models, *IEEE Trans. Neural Networks*, Vol.8, No.3, 475-485, 1997.

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

[19] H.H.Tsai and P.T.Yu, On the optimal design of fuzzy neural networks with robust learning for function approximation, *IEEE Trans. Syst., Man, Cybern. B*, Vol.30, 217-223, 2000.

[20] S.Wu and M.J.Er, Dynamic fuzzy neural networks- a novel approach to function approximation, *IEEE Trans. Syst., Man, Cybern. B*, Vol.30, 358-364, 2000.

[21] C.H.Wang, H.L.Liu and C.T.Lin, Dynamic optimal learning rates of a certain class of fuzzy neural networks and its applications with genetic algorithm, *IEEE Trans. Syst., Man, Cybern. B*, Vol.31, 467-475, 2001.

[22] L.X.Wang, *Adaptive Fuzzy Systems and Control*, Englewood Cliffs NJ: Prentice-Hall, 1994.

[23] W.Y.Wang, T.T.Lee, C.L.Liu and C.H.Wang, Function approximation using fuzzy neural networks with robust learning algorithm, *IEEE Trans. Syst., Man, Cybern. B*, Vol.27, 740-747, 1997.

[24] W.Y.Wang, Y.G.Leu and C.C.Hsu, Robust adaptive fuzzy-neural control of nonlinear dynamical systems using generalized projection updated law and variable structure controller, *IEEE Trans. Syst., Man, Cybern. B*, Vol.31, 140-147, 2001.

[25] W.Yu and X. Li, Some stability properties of dynamic neural networks, *IEEE Trans. Circuits and Systems, Part I*, Vol.48, No.1, 256-259, 2001.

[26] W.Yu and X. Li, Some new results on system identification with dynamic neural networks, *IEEE Trans. Neural Networks*, Vol.12, No.2, 412-417, 2001.