ESTUDIO DEL GRUPO SOBRE TÉCNICAS EMERGENTES (Año 2007)

Resumen

Las redes neuronales y los sistemas basados en lógica borrosa son las nuevas herramientas de uso más difundido en la actualidad, encontrándose complementadas con los algoritmos genéticos, formando todas ellas el denominado conjunto de técnicas emergentes.

Todas se encuentran basadas en las soluciones que la naturaleza ha dado a lo largo de su evolución a los problemas derivados de información masiva y distorsionada por parte del entorno, dichas soluciones fueron copiadas por sistemas artificiales con la finalidad de resolver problemas relacionados al reconocimiento del habla, del lenguaje, del control inteligente, del procesamiento de señales, etc.

Para introducirnos en el tema en este trabajo presentaremos los conceptos teóricos básicos de:

- redes neuronales artificiales, que simulan la estructura del cerebro, mediante un hardware concreto, con la finalidad de reproducir sus capacidades específicas y
- sistemas borrosos, que tratan de manejar conceptos no precisos, como los utilizados en la vida cotidiana, mediante variables lingüísticas, permitiendo incorporar "inteligencia" en los sistemas de control.

Ambas herramientas nos permitirán descubrir de una manera automática relaciones de entrada-salida, o de rasgos característicos, partiendo de datos concretos y mediante un proceso de aprendizaje a partir de ejemplos. El aporte extra que producen los sistemas borrosos es la incorporación del conocimiento de los expertos.

Para finalizar, es de fundamental importancia destacar que estas técnicas, y otras complementarias, deberán integrarse para obtener mejores resultados finales.

REDES NEURONALES ARTIFICIALES

1. Introducción

El desarrollo de las redes neuronales artificiales (RNA) está basado en la teoría de grafos y son utilizadas cada vez con mayor frecuencia para la solución de diversos problemas relacionados con la Ingeniería, la Estadística, la computación, la Economía y la Administración, entre otros.

Las RNA son sistemas de procesamiento de información cuyo funcionamiento y modelización se basan en la estructura de los sistemas neurobiológicos. Básicamente las RNA intentan modelizar el cerebro humano, desarrollando incluso actividades tan complejas como el aprendizaje, intentando poseer cierta inteligencia, es decir son capaces de aprender de la experiencia a partir de señales o datos provenientes del

exterior dentro de un marco de computación. Es por eso que son utilizadas básicamente en modelos de aprendizaje y análisis de datos, así como también como predictores de series cronológicas.

No obstante lo anterior, *"la inteligencia de la RNA se encuentra todavía en fuerte debate. Por un lado, las neuronas artificiales son capaces de procesar amplia cantidad de datos y efectuar extremadamente precisas predicciones, pero por otro lado, muchos autores argumentan que a pesar de su capacidad para predecir con precisión, las neuronas artificiales no son inteligentes en el sentido humano, ya que no son capaces de imaginar"* (Yochanan Shachmurove).

Entre las principales aplicaciones de las RNA podemos mencionar: las series cronológicas, la minería de datos, minería de datos con mapas autoorganizados, comportamiento colectivo, modelizaciones de respuestas de mercados, flujos de telecomunicaciones, valuación de bienes raíces, optimización, relaciones entre los fenómenos económicos y financieros, entre otras.

Cabe advertir por último, que muchos de los conceptos que modelizan las RNA ya tienen un tratamiento específico y de buenos resultados dentro del ámbito de la Estadística, por lo que muchos de los resultados que con ellas se pueden obtener deberían ser comparados con los arrojados por las metodologías estadísticas desarrolladas con anterioridad (Joekes, S. y Yacci, M.R., 2003).

2. La neurona biológica

El sistema nervioso está compuesto por una red de células individuales ampliamente interconectadas entre sí: las neuronas. Éstas constituyen procesadores de información que funcionan de la siguiente manera: ingresa información por las dendritas, se procesa en la soma y se transmite hacia el exterior por el axón. A su vez, en las denominadas "neuronas intermedias" el axón se conecta a otras dendritas, mientras que las llamadas "neuronas receptoras" reciben información desde el exterior.

En la siguiente figura se presenta una neurona biológica:

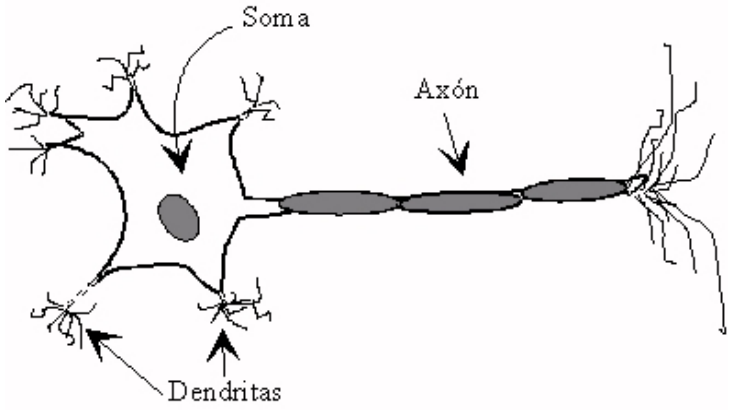


Figura 1. Neurona biológica típica

Se denomina sinapsis a la unión entre dos neuronas, siendo su comunicación direccional, es decir, fluye la información en un único sentido; además se puede afirmar que las señales nerviosas se pueden transmitir de forma eléctrica (en el interior, mediante descargas que se producen en el cuerpo celular y se propagan por el axón) o químicamente (fuera de la neurona, fundamentándose en el intercambio de neurotransmisores).

La comunicación más habitual es la química, mediante la cual la neurona presináptica libera sustancias químicas complejas (como la adrenalina), que atraviesan el vacío sináptico; entonces, si la neurona postsináptica es sensible a esas sustancias químicas aceptará el estímulo permitiendo el paso de determinados iones a través de su membrana, la que se comporta como un capacitor que se carga al recibir impulsos eléctricos provenientes de los iones. Las corrientes iónicas provocan potenciales postsinápticos excitadores (positivos) o inhibidores (negativos), que se manifiestan mediante la existencia de un umbral de disparo. Para ver esquemáticamente este proceso presentamos la siguiente figura:

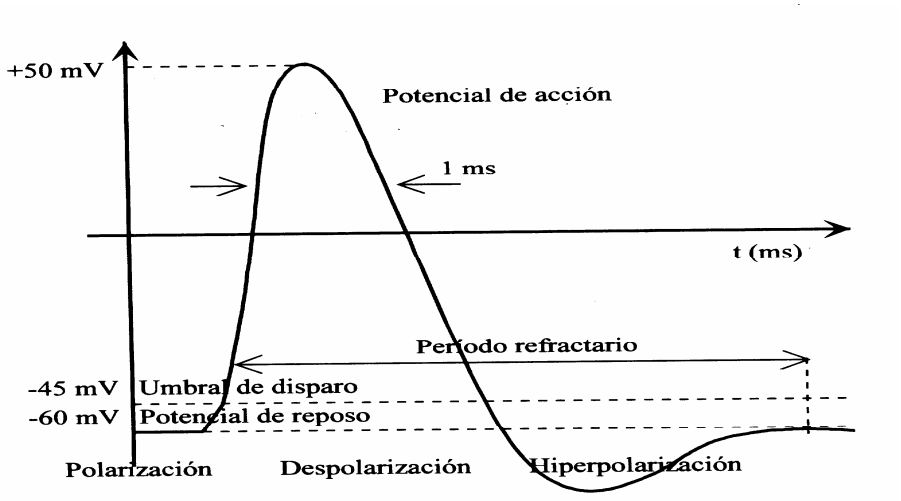
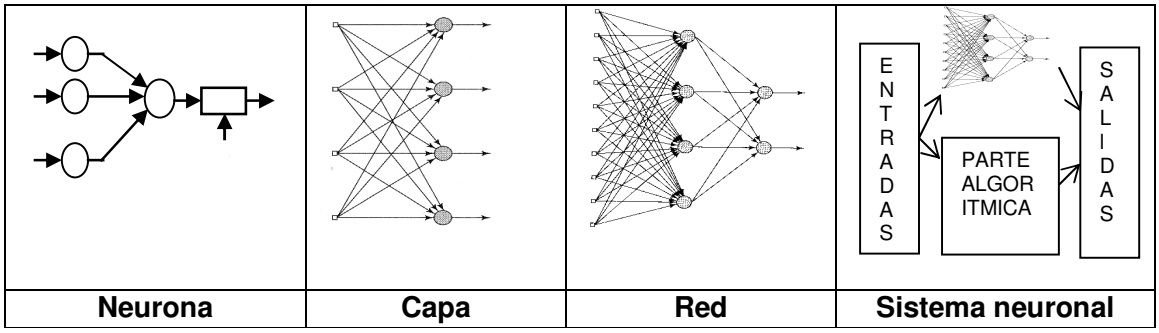


Figura 2. Potencial de acción

Se debe aclarar que este esquema presentado resulta una simplificación, pues la realidad en mucho más compleja.

3. Estructura de una red de neuronas artificiales

Recordando que las RNA imitan la estructura del sistema nervioso con la finalidad de construir sistemas de procesamiento de información para que tengan cierto comportamiento inteligente, podemos graficar jerárquicamente su estructura mediante el siguiente cuadro:



Neurona	Capa	Red	Sistema neuronal

Se puede hablar de redes neuronales como sistemas:

- de procesamiento paralelo, donde se trabaja sobre una imagen para reconocerla, pudiéndose: acentuar contrastes, extraer contornos, etc.
- de memoria distribuida, por las sinapsis de la red, donde si una neurona resulta dañada no se pierde más que una pequeña parte de la información
- de adaptabilidad, es decir con rápida adaptación al entorno, modificando las sinapsis, aprendiendo de la experiencia; esto lleva a la propiedad de poder generalizar a través de ejemplos

4. Modelo de neurona artificial

Una neurona artificial puede entenderse como un procesador elemental, es decir un dispositivo simple de cálculo, que a partir de un vector de entrada (procedente del exterior o de otra neurona), produce una respuesta única.

Tendremos entonces, un conjunto de inputs, un conjunto de ponderadores, una función de activación, un umbral, y un output de la neurona.

Lo que se podría graficar de la siguiente manera:

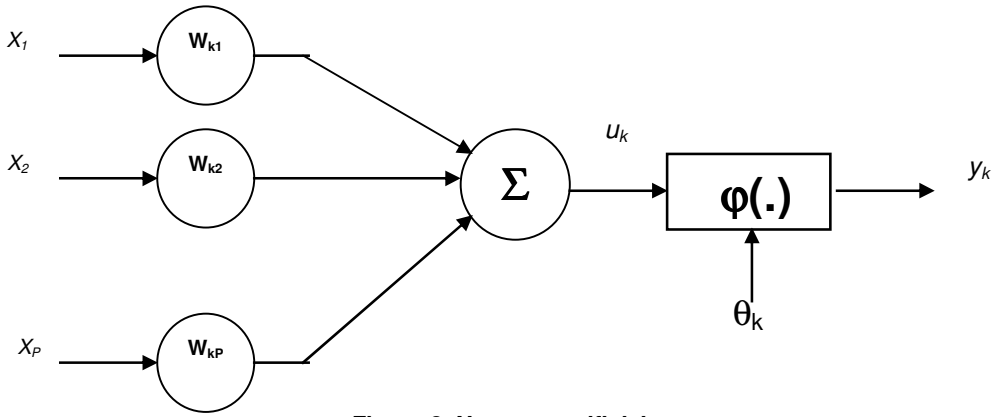


Figura 3. Neurona artificial

Donde encontramos:

- Conjunto de entradas $\rightarrow x_j(k)$;
- Pesos sinápticos para cada neurona $i \rightarrow w_{ij}$;
- Regla de propagación $\rightarrow u_i(k) = \varphi\left(w_{ij}, x_j(k)\right)$ que proporciona el valor potencial postsináptico;
- Función de activación $\rightarrow y_i = f\left(u_i(k-1)\right)$ que proporciona el estado de activación actual $u_i(k)$ y
- Umbral $\rightarrow \theta_k$ que responde a un parámetro adicional que modifica la función de activación de la siguiente manera: $y_i = f\left(\sum w_{ij} x_j - \theta_i\right)$, y representará el nivel mínimo que debe alcanzar el potencial postsináptico para que la neurona se dispare o active.

5. Conceptos que intervienen en una neurona artificial

5.1. Peso sináptico

Define la intensidad de interacción entre la neurona de entrada *j* y la de salida *i*, donde si el peso es positivo tenderá a excitar a la postsináptica, mientras que si es negativo tenderá a inhibirla.

5.2. Reglas de propagación

Permiten obtener a partir de las entradas y los pesos el valor postsináptico $u_i(k)$, cuya función más habitual es la suma ponderada, con la siguiente simbología:

$$u_i(k) = \sum_{j=1}^{p} w_{ij} x_j \ .$$

Otra regla no lineal sería la siguiente:

$$u_i(k) = \sum_{j=1}^{p} w_{ij1j2\cdots jp} x_{j1} x_{j2} \cdots x_{jp} \ ,$$

que implica una interacción de tipo multiplicativo entre las entradas de las neuronas, determinando una neurona de orden superior.

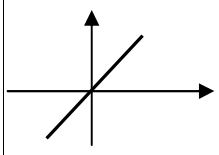
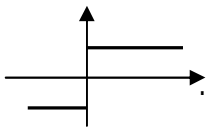
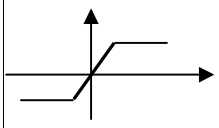
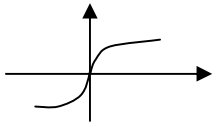
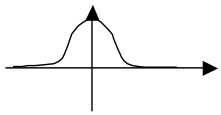
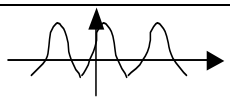
Finalmente encontramos como regla la diferencia euclídea, con la siguiente simbología:

$$u_i^2(k) = \sum_{j=1}^{p} \left(x_j - w_{ij} \right)^2 ,$$

que representa la diferencia entre el vector de entradas y el de pesos.

5.3. Función de activación o de transferencia

Esta función proporciona el estado de activación actual, a partir del potencial postsináptico y del propio estado de activación anterior, siendo sus formas más habituales las que podemos observar en el siguiente cuadro:

	Función	**Rango**	**Gráfica**
Identidad	$y = x$	$[-\infty, \infty]$	
Escalón	$y = sign(x)$	$\{-1,1\}$ $\{0,1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si \quad x \prec -1 \\ x, & si \quad 1 \le x \le -1 \\ 1, & si \quad x \succ 1 \end{cases}$	$[-1,1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$	$[0,1]$ $[-1,1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,1]$	
Sinusoidal	$y = Asen(\varpi x + \varphi)$	$[-1,1]$	

Para observar prácticamente como se comportan estos cálculos supongamos un ejemplo de aplicación:

- hay 4 dendritas de entrada cuyos potenciales son:

$x_1 = 0.3$ $x_2 = 0.8$ $x_3 = 0.5$ $x_4 = 0.9$

- y sus respectivas ponderaciones son:
  $w_1 = 0.5$ $w_2 = 0.7$ $w_3 = 0.1$ $w_4 = 0.4$
- la función de activación en este caso se calculará:

$$u_i(k) = \sum_{j=1}^{p} w_{ij} x_j = (0.3 * 0.5) + (0.8 * 0.7) + (0.5 * 0.1) + (0.9 * 0.4) = 1.12$$

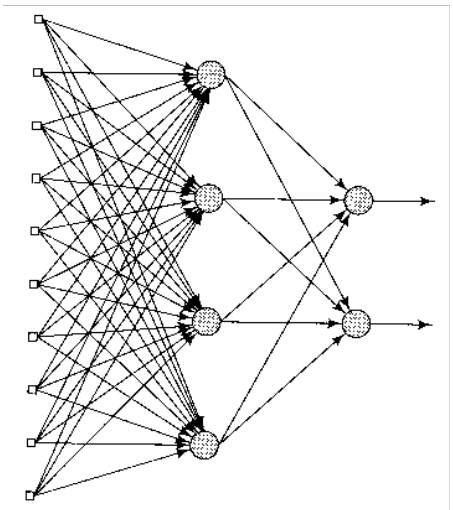
- si el umbral para este ejemplo fuera $\theta_i=1$, las dendritas de salida tendrán un potencial de 0.12, dado que el umbral es menor al potencial calculado como suma ponderada, si por el contrario el umbral hubiera sido, por ejemplo, de 1.2, las dendritas de salida no tendrían potencial.

Una vez calculado el output de la neurona, puede pasarse a otra neurona de la red. La interpretación del output de la neurona depende del problema que se está considerando, pero si se tiene, por ejemplo, valores de salida 0 o 1, dependiendo de la función de activación, podría representar el 1 a la pertenencia a una clase y el 0 a la no pertenencia (lógica booleana), mientras que si el potencial de salida pertenece al intervalo [0,1], nos indicaría el grado de pertenencia a la clase (lógica difusa).

6. Arquitectura de redes neuronales

La arquitectura de conexiones sinápticas determina el comportamiento de la red, de las que se puede decir que son direccionales y cuyas neuronas se agrupan en unidades estructurales que se denominan capas; precisamente el conjunto de una o más capas constituyen la red neuronal. Si se quisiera definir una red utilizando el concepto de grafo, tendríamos que consiste: en un conjunto de nodos, más un conjunto de conexiones establecidas entre ellos.

Encontramos tres tipos de capas como puede observarse en la siguiente figura:



Capa de entrada Capa oculta Capa de salida

Figura 4. Arquitectura unidireccional

que pueden definirse como:

- de entrada, reciben datos o señales procedentes del entorno,
- de salida, proporciona la respuesta de la red neuronal y
- oculta, no tiene una conexión directa con el entorno.

6.1. Algunas consideraciones sobre la estructura

Con relación a las **conexiones** existen:

- **intracapa**, tienen lugar entre las neuronas pertenecientes a una misma capa y
- **intercapa**, se producen entre neuronas de diferentes capas.

Con respecto a la **cantidad de capas** puede clasificarse en:

- **monocapa**, compuesta por una única capa de neuronas y
- **multicapa**, son aquellas cuyas neuronas se organizan en varias capas.

Atendiendo al **flujo** de datos encontramos:

- **unidireccionales**, donde la circulación es en un solo sentido y
- **recurrentes**, la información puede circular entre las capas en cualquier sentido.

Considerando la **memoria asociativa** de la red tenemos:

- **heteroasociativa**, mediante la cual ante un determinado patrón de entrada responde con otro patrón de salida, siempre el mismo y
- **autoasociativa**, responde a la red entrenada para asociar un determinado patrón consigo mismo.

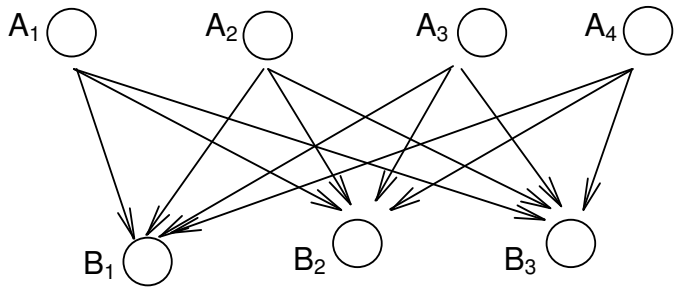
Finalmente, si se tiene en cuenta la **dinámica de actualización** del estado de las neuronas, tenemos:

- **sincrónica**, los estados se actualizan en función de un cierto reloj común,
- **asincrónica**, cada neurona actualiza su estado sin atender a cuándo lo hacen las demás y
- **no determinista**, la salida de la neurona tiene carácter probabilístico.

7. Funcionamiento de las RNA (Kauffman, A. y Gil Aluja, J., 1995). Caso práctico

7.1. Neuronas con estado en {0, 1}

Supongamos una red que contiene sólo dos capas: la capa A, compuesta por cuatro neuronas, y la capa B, que contiene tres neuronas. Las neuronas de la capa A envían señales x_i ponderadas por w_{ij} a las neuronas de la capa B, tal como lo muestra la figura siguiente:



Dado que cada dendrita de salida de cada neurona A_i de la capa A, que a su vez es una dendrita de entrada en una neurona B_j de la capa B, tiene un potencial y un peso de ponderación, la señal que va a recibir cada neurona B_j es $\beta_j = \sum_{i=1}^{4} x_i w_{ij}$, tomando j los valores 1, 2 o 3.

Supongamos que el estado inicial de las neuronas de la capa A es el siguiente:

	A_1	A_2	A_3	A_4
A	1	0	0	1

Como lo indica el subtítulo de este parágrafo, estamos trabajando bajo es supuesto que el estado de una neurona sólo pueda tomar los valores 0 o 1.

Representaremos el peso de ponderación de cada arco con la siguiente notación matricial:

	B_1	B_2	B_3
A_1	0,9	-1,5	1,1
A_2	0,5	2,1	-0,2
A_3	1,4	-0,9	2,1
A_4	-1,6	1,3	3,4

w_{ij}

De manera que el peso del arco que sale de la neurona A_1 hacia la neurona B_1 es de 0,9, el peso de la dendrita que sale de la neurona A_3 y llega a la neurona B_2 es –0,9 y así sucesivamente, hasta que la conexión entre la neurona A_4 y la neurona B_3 posee un peso ponderador de 3.4.

Las β_j se calcularán, entonces, mediante la convolución suma-producto, como se muestra en los siguientes cuadros:

	A_1	A_2	A_3	A_4
A	1	0	0	1

$\bigcirc$

	B_1	B_2	B_3
A_1	0,9	-1,5	1,1
A_2	0,5	2,1	-0,2
A_3	1,4	-0,9	2,1
A_4	-1,6	1,3	3,4

$=$

B_1	B_2	B_3
-0,7	-0,2	4,5

β_j

Estos valores β_j son los que se compararán, dentro de cada una de las neuronas B_j de la capa B, con el umbral θ_j, que supondremos toma los siguientes valores:

B_1	B_2	B_3
0,3	-0,5	2,1

θ_j

De manera que comparamos este umbral con los valores obtenidos de la convolución suma-producto y si éste último valor para una neurona es mayor a su umbral, dicha neurona tendrá un valor final de 1, en caso contrario tomará el valor 0. En nuestro ejemplo será:

	B_1	B_2	B_3
B	0	1	1

En este caso se ha tomado, para simplificar, una red neuronal con sólo dos capas. Si existieran más capas, con la última matriz hallada, que corresponde a los valores que toma cada una de las neuronas de la capa B, deberá efectuarse una convolución suma-producto con la matriz de los pesos ponderadores de los arcos que van de cada una de las neuronas de la capa B a cada una de las neuronas de la capa siguiente (C), los valores obtenidos se comparan con el umbral para cada una de las neuronas de esta tercer capa y así se determina el estado final de estas últimas neuronas, es decir, se repite el mismo mecanismo que ya se explicó. Si hubiera una cuarta capa o más, se continúa el proceso.

Como comentario final se puede decir que, si trabajáramos bajo el supuesto de que el estado de una neurona puede indicarse con los valores –1 y 1, en lugar de con 0 y 1, como lo hicimos en este apartado, la única diferencia sería que en caso de no superar cada valor resultante de convolucionar el potencial de cada dendrita con su respectiva ponderación el valor del umbral, se asignará a la neurona de llegada el estado –1 en lugar de 0.

7.2. Redes con señales en [0, 1]

En el punto anterior se propuso el caso en que el estado de una neurona podía representarse sólo con dos valores posibles, el 0 o el 1 (o el –1 y el 1). En este caso, las situaciones que podían darse eran dos: el valor ponderado de cada potencial superaba o

no al umbral y se asignaba 1 o 0 (o –1) a la neurona de llegada, respectivamente, sin importar si dicho valor superaba por mucho o era muy inferior al umbral.

En este apartado incluimos borrosidad en el análisis, dejando de ser ahora dicotómica la asignación de valor. Trabajaremos con una función sigmoide, de manera que el valor que indica el estado de una neurona será:

Si $\beta_j \leq \theta_j$, entonces $b_j = 0$

Si $\beta_j > \theta_j$, entonces $b_j = \frac{1}{1+e^{-\beta_j}}$

Esto nos indica que por debajo del umbral la neurona tendrá estado 0, como en el caso anterior, pero si se sobrepasa el umbral, el estado de la neurona de llegada no tendrá un único valor sino que éste será tanto mayor cuanto más se sobrepase al umbral. Incluso, los umbrales que fijamos en el apartado anterior eran un número preciso. Si introducimos incertidumbre también en la fijación de los umbrales, podríamos fijar dos umbrales, de manera que el estado de la neurona de llegada tomará los siguientes valores:

Si $\beta_j \leq \theta_{1j}$, entonces $b_j = 0$

Si $\theta_{1j} < \beta_j \leq \theta_j$, entonces $b_j = \frac{1}{1+e^{-\beta_j}}$

Si $\beta_j > \theta_j$, entonces $b_j = 1$

Se propone a continuación, con el objeto de aclarar los conceptos recién vertidos, un ejemplo numérico, utilizando una red de tres capas: la primera tendrá tres neuronas, la segunda dos y la tercera también tres.

El estado inicial de las neuronas de la capa A son:

A_1	A_2	A_3
0,8	0	1

A su vez, las ponderaciones de las dendritas son:

	B_1	B_2
A_1	-1,5	1,2
A_2	0,6	-0,7
A_3	0,1	-0,5

Si hacemos la convolución suma-producto entre ambas matrices, obtenemos los siguientes valores:

B_1	B_2
-1,1	0,46

Estos valores deberán ser comparados con los umbrales para luego determinar el estado final de cada neurona de la capa B.

A tal efecto, utilizaremos dos umbrales, θ_{1j} y θ_{2j}, siendo el valor del primero el límite inferior del intervalo de confianza que presentaremos y el segundo el límite superior de cada intervalo.

B_1	B_2
[-1,3; 1]	[0,25; 0,75]

Ahora sí podemos comparar los valores antes calculados con estos umbrales. Dado que tanto para B_1 como para B_2 el valor correspondiente se encuentra entre los umbrales respectivos, el estado final de las neuronas de la capa B no será ni 0 ni 1, sino que debemos usar la función sigmoide, de la siguiente manera:

$$b_{j1} = \frac{1}{1+e^{-\beta_j}} = \frac{1}{1+e^{1,1}} = 0,2497$$
$$b_{j2} = \frac{1}{1+e^{-\beta_j}} = \frac{1}{1+e^{-0,46}} = 0,6130$$

Por lo tanto, el estado final de las neuronas de la capa B será:

B_1	B_2
0,2497	0,6130

Con estos valores se puede repetir el proceso para obtener el estado final de las neuronas de la tercera y última capa de esta red neuronal artificial.

Si los pesos de ponderación son:

	C_1	C_2	C_3
B_1	-2,8	1,3	0,7
B_2	1,4	2,9	-0,2

La convolución suma-producto brinda los siguientes resultados:

C_1	C_2	C_3
0,1590	2,1023	0,0522

Si los umbrales son:

C_1	C_2	C_3
[0,5; 1,7]	[0,3; 1,6]	[-1,5; 0,75]

Entonces, el estado final de cada neurona de la tercera capa es: para C_1, como el valor obtenido de la convolución (0,1590) es menor al primer umbral (θ_{11}=0,5), entonces el estado final de dicha neurona será 0; para C_2, como el valor obtenido de la convolución es mayor al segundo umbral (θ_{22}=1,6), la neurona en cuestión tendrá un estado final 1; C_3 posee un valor que se encuentra entre los umbrales propuestos, por lo que su estado final será: $b_{j3} = \frac{1}{1+e^{-\beta_j}} = \frac{1}{1+e^{-0,0522}} = 0,5130$.

Con lo cual, tendremos:

C_1	C_2	C_3
0	1	0,5130

7.3. Redes con señales en [-1, 1]

En el parágrafo anterior se trabajó con neuronas que podían adoptar cualquier valor, siempre y cuando no fuera ni superior a 1 ni inferior a 0. En este punto, se respetará el mismo valor superior que en el caso anterior, pero cambiaremos el límite inferior a –1, de manera que tendremos, para la alternativa incluida anteriormente consistente en trabajar con dos umbrales, los siguientes valores a adoptar por cada neurona de llegada:

Si $\beta_j \leq \theta_{1j}$, entonces $b_j = -1$

Si $\theta_{1j} < \beta_j \leq \theta_{2j}$, entonces $b_j = \frac{1-e^{-\beta_j}}{1+e^{-\beta_j}}$

Si $\beta_j > \theta_{2j}$, entonces $b_j = 1$

Así, para el siguiente ejemplo consistente en una red de tres capas de neuronas, tendremos que, si las neuronas de la primera capa tienen los siguientes estados iniciales:

A_1	A_2	A_3
0,4	0.97	-0.8

A su vez, las ponderaciones de las dendritas son:

	B_1	B_2
A_1	-0.9	-0.4
A_2	0,2	0,7
A_3	-0.6	-0,3

Si hacemos la convolución suma-producto entre ambas matrices, obtenemos los siguientes valores:

B_1	B_2
0.31	0,76

Ahora bien, para conocer el estado final de las neuronas de la segunda capa, debemos comparar los dos valores recién obtenidos con los umbrales que a continuación proponemos y, luego, aplicar la regla presentada al comenzar el apartado.

Los umbrales para cada neurona son:

B_1	B_2
[-1,6; 1.3]	[0,95; 1,35]

Como puede apreciarse, la primera neurona de la segunda capa presenta un estado que se encuentra comprendido entre los valores inferior y superior del umbral respectivo, por lo que para determinar el valor que le corresponde como estado final a dicha neurona habría que hacer:

$$b_j = \frac{1-e^{-\beta_j}}{1+e^{-\beta_j}} = \frac{1-e^{-0.31}}{1+e^{-0.31}} = 0.15$$

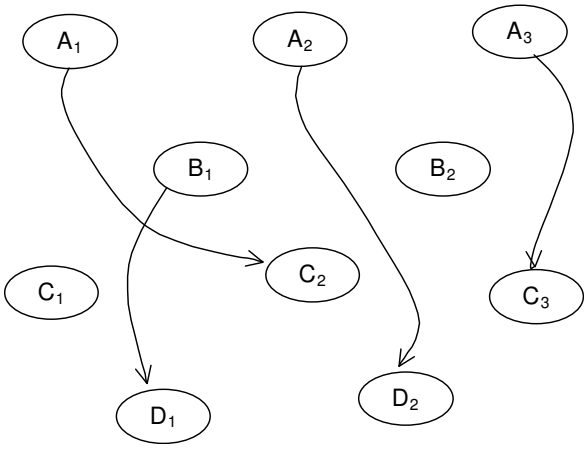
Por su parte, la segunda neurona presenta un estado inferior al menor valor del umbral correspondiente, por lo que el estado final de dicha neurona será –1.

Así, le segunda capa de neuronas quedará con los siguientes valores finales:

B_1	B_2
0.15	-1

7.4. Función ordinal en redes neuronales con capas

Supongamos una red neuronal de cuatro capas donde cada neurona de una capa envía una dendrita a cada neurona de la capa siguiente pero, además, existen dendritas que saltean una capa, tal como se presenta en el gráfico que sigue (obviamos las conexiones entre neuronas de capas inmediatas):



Si las neuronas de la primer capa tienen los siguientes estados:

	A_1	A_2	A_3
A	0.5	0.8	1

y los ponderadores de las dendritas que van desde cada neurona de la capa A a cada neurona de la capa B son:

	B_1	B_2
A_1	0.1	1
A_2	0.9	0.5
A_3	0	-2

tendremos las siguientes entradas:

	A_1	A_2	A_3
A	0.5	0.8	1

$\bigcirc$

	B_1	B_2
A_1	0.1	1
A_2	0.9	0.5
A_3	0	-2

$=$

B_1	B_2
0.77	0.7

Si consideramos los umbrales siguientes:

B_1	B_2
[0.7, 1.5]	[1, 1.75]

y trabajamos con la regla ya presentada, según la cual:

Si $\beta_j \leq \theta_{1j}$, entonces $b_j = 0$

Si $\theta_{1j} < \beta_j \leq \theta_j$, entonces $b_j = \frac{1}{1+e^{-\beta_j}}$

Si $\beta_j > \theta_j$, entonces $b_j = 1$

se obtiene:

	B_1	B_2
B	0.6835	0

ya que $\frac{1}{1+e^{-0.77}} = 0.6835$

Hasta ahora se ha calculado el estado que tendrán las neuronas de la segunda capa. Para calcular los estados de las neuronas de la tercer capa, debemos considerar que las neuronas C_2 y C_3 reciben información de las neuronas A_1 y A_3, respectivamente.

Dada esta situación, precisamos, además de los ponderadores de las conexiones entre las B_j y las C_j, los propios de las conexiones entre A_1 y C_2 y A_3 y C_3. Si tales ponderadores son:

	C_1	C_2	C_3
B_1	0.7	1.7	-0.5
B_2	-2.1	-1.1	0

y

	C_1	C_2	C_3
A_1	0	-2.3	0
A_2	0	0	0
A_3	0	0	1.8

y recordando que

A_1	A_3
0.5	1

y

B_1	B_2
0.6835	0

tenemos que:

$$C_1 = 0.6835 * 0.7 + 0 * (-2.1) = 0.4785$$
$$C_2 = 0.6835 * 1.7 + 0 * (-1.1) + 0.5 * (-2.3) = 0.012$$
$$C_3 = 0.6835 * (-0.5) + 0 * 0 + 1 * 1.8 = 1.4583$$

Es decir:

C_1	C_2	C_3
0.4785	0.012	1.4583

Si tenemos los siguientes umbrales para cada una de las neuronas de esta tercer capa:

C_1	C_2	C_3
[0.3, 1.2]	[0.8, 2.1]	[0.5, 1.4]

Quedarán los siguientes valores para cada neurona:

C_1	C_2	C_3
0.6174	0	1

Siguiendo con la misma metodología y sabiendo que los ponderadores de las relaciones entre las neuronas de la tercer y cuarta capa son:

	D_1	D_2
C_1	0.1	0.5
C_2	-1.2	0.7
C_3	2.1	-0.9

mientras que los ponderadores de las conexiones entre la primera capa y la última y entre la segunda y la última son:

	D_1	D_2
A_1	0	0
A_2	0	0.4
A_3	0	0

	D_1	D_2
B_1	1.4	0
B_2	0	0

tenemos que:

$$D_1 = 0.6174 * 0.1 + 0 * (-1.2) + 1 * 2.1 + 0.6835 * 1.4 = 3.1186$$
$$D_2 = 0.6174 * 0.5 + 0 * 0.7 + 1 * (-0.9) + 0.8 * 0.4 = -0.2713$$

Si los umbrales son:

D_1	D_2
[0.8, 2.1]	[-1.5, 0.6]

Los estados finales de las neuronas de la cuarta y última capa serán:

D_1	D_2
1	0.4326

En el presente epígrafe se ejemplificó el funcionamiento de redes neuronales por capas cuando algunas neuronas saltan una capa para conectarse con otra de una capa posterior a la inmediata.

Para ello se utilizó el supuesto se señales en [0, 1], pero de manera análoga puede trabajarse si el supuesto es de señales en {0, 1}, {-1, 1} o [-1, 1].

8. Modos de operación: recuerdo y aprendizaje

8.1. Aprendizaje o entrenamiento

Al construir un sistema neuronal se debe definir el tipo de neurona y la estructura, para lo cual se deberán fijar los primeros pesos, que pueden ser nulos o aleatorios. Para lograr buenos resultados es necesario entrenar la red, lo que se denomina modo de aprendizaje, que se puede desarrollar a dos niveles.

Lo más usual consiste en modificar los pesos sinápticos siguiendo cierta regla de aprendizaje, que pretende optimizar la función de error que mide la eficacia actual de la red.

El proceso es iterativo, actualizando una y otra vez hasta alcanzar el rendimiento deseado, para ello se utiliza la siguiente expresión:

$$\Delta w_{ij}\left(k+1\right) = w_{ij}\left(k\right) + \Delta w_{ij}\left(k\right)$$

Existen diferentes tipos de aprendizaje:

- **supervisado**, se presenta a la red un conjunto de patrones, junto con la salida deseada y de manera iterativa ajusta sus pesos hasta obtener lo deseado;
- **no supervisado o autoorganizado**, se presenta a la red una multitud de patrones sin presentar la salida deseada, la red a partir de reglas de aprendizaje reconocerá regularidades, extraerá datos o agrupará por similitud;
- **híbrido**, coexisten los dos anteriores en diferentes capas y
- **reforzado**, es un intermedio de los dos primeros, como en el supervisado se utiliza información sobre el error pero que representa un índice de rendimiento global de la red (establece sin detalles lo bien o mal que la red actúa).

El aprendizaje se realiza mediante algoritmos que tratan de minimizar una función de error, lo que puede dar lugar a problemas con convergencia.

Es preciso distinguir entre el error alcanzado por la fase de aprendizaje de los datos de entrenamiento, con el error mediante el cual la red entrenada mide su capacidad de generalizar.

8.2. Recuerdo o ejecución

Entrenada una red, suponiendo que se llegó a la salida deseada, se congelan pesos y estructura y se dispone a procesar datos, al no existir realimentación se puede decir que se tiene estabilidad en los resultados.

9. Aplicación de redes neuronales artificiales

El modo más habitual de realizar una red es mediante la simulación en una PC, mediante programas de alto nivel especializados; la utilización a problemas concretos prácticos están alcanzando excelentes resultados por su: capacidad de aprendizaje, robustez, no linealidad y tolerancia a la incertidumbre del entorno.

Las **principales aplicaciones** actuales pueden resumirse de la siguiente manera:

a) para redes lineales
- telecomunicaciones,
- anulación de ruido y vibraciones y
- alineación de haces de partículas

b) para redes no lineales
- clasificación de patrones, como por ejemplo: reconocimiento del habla, reconocimiento de caracteres manuscritos, control de calidad, etc.
- predicción y análisis financiero, como por ejemplo: concesión de préstamos, análisis de mercados, reservas de vuelos, etc.
- control y optimización,
- aplicaciones militares, por ejemplo en defensa,
- predicción,
- economía,
- bioingeniería, etc.

Las siguientes nos darán las **características que debe cumplir el problema** para ser tratado con RNA:

- no se dispone de un conjunto de reglas que lo defina completamente,
- sí se dispone de muchos ejemplos o casos,
- los datos son imprecisos, incoherentes o con ruido,
- con elevada dimensionalidad,
- la solución es más rápida que mediante el uso de técnicas convencionales y
- ante nuevos cambios puede reentrenarse la red con los nuevos datos.

Las redes neuronales **no son útiles** cuando: existe un algoritmo que resuelve totalmente el problema, no se dispone de un número adecuado de casos o cuando se requiera una solución siempre predecible y explicable.

Para resolver un problema mediante RNA se deben realizar los siguientes **pasos**:

1. planteamiento del problema, descripción detallada con expresa indicación de algún aspecto que podrá ser resuelto mediante una red;
2. requerimientos del sistema, se deben definir concretamente las especificaciones a cumplir, por ejemplo: el error, el tipo de formulación, la forma de disposición de los datos, el tiempo de respuesta requerido, etc.;
3. revisión bibliográfica, tratando de indagar sobre aplicaciones parecidas;
4. elección del modelo más apropiado;
5. datos disponibles y selección de variables relevantes, para los primeros se debe definir si se dispone de todos al mismo tiempo, en este caso se podrá realizar un entrenamiento por lotes, mientras que si se reciben en tiempo real deberá procederse con un entrenamiento en línea; con relación a las variables, se debe tener en cuenta las de entrada y las de salida, eligiéndolas cuidadosamente para obtener los mejores resultados, recordando: pocas variables restringen el espacio de búsqueda, puede viciar la arquitectura y llegar a una pobre generalización y muchas variables pueden conducir a un error de generalización elevado;
6. elección de los conjuntos de aprendizaje y test, siempre considerando que el número debe ser lo suficientemente representativo del fenómeno que se quiere modelizar;
7. preprocesamiento, es el tratamiento previo de los datos para poder incorporarlos a la red, se puede cambiar el tipo de distribución aplicando una transformación, siendo las más frecuentes el empleo de algoritmos, raíces cuadradas y cúbicas, si se pretende modificar el rango se debe aplicar un escalado (puede ser: por patrones –filas– o por entradas –columnas–), el que mejores resultados proporciona es el realizado por columnas, que consiste en: estandarizar ($x' = \frac{x - \overline{x}}{\sigma}$) y luego se escalan esos valores al intervalo $[0,1]$ ó $[-1,1]$;
8. proceso de entrenamiento, que puede modificar: los pesos iniciales, el ritmo de aprendizaje (sabiendo que un ritmo pequeño evita escapar de mínimos locales y disminuye la velocidad de convergencia; mientras que uno alto puede conducir a la inestabilidad), el número de neuronas ocultas (no existen recetas para su determinación, pudiéndose recurrir por ejemplo a: prueba y error) y la parada (no se pretende extender indefinidamente el sistema, considerando que a partir de cierto punto se pierde la facultad de generalizar); finalmente
9. evaluación de los resultados, es decir cuando se llega al ideal se aplica la propuesta a casos nuevos, con la finalidad de medir la eficacia de la red de forma completamente objetiva.

Los principales **inconvenientes** que presentan las RNA pueden resumirse de la siguiente manera:

- constituyen métodos de resolución demasiado creativos, es decir dadas las especificaciones del problema, se desconoce la topología de red que va a solucionar de forma más eficiente, hay que utilizar prueba y error,

- una vez entrenada es difícil interpretar su funcionamiento, no es fácil asegurar con qué acierto responderá ante casos nunca vistos,
- es difícil averiguar por qué la red no es capaz de ajustar los datos y
- los modelos neuronales requieren elevados requisitos de cómputo.

La aplicación de las RNA puede fundamentarse con las siguientes **conclusiones**:

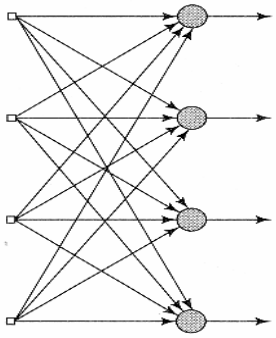
- para resolver eficientemente un problema, éste debe comprenderse bien;
- para ello se debe introducir explícitamente su conocimiento apriorístico;
- es conveniente separar el problema en partes;
- para seleccionar el mejor método se deben ensayar con distintos, realizando comparaciones objetivas, siendo la tendencia actual integrar diversas soluciones (neuronales, borrosas, estadísticas, etc.);
- debe huirse de las aplicaciones a ciegas, pues no se llegará a la solución más eficaz;
- es posible que se deba crear un software específico para la aplicación, con incluso un hardware específico para lo cual deberá tenerse en cuenta no sólo el rendimiento del equipo, sino una perspectiva global que comprenda: aspectos económicos, velocidad, robustez, fiabilidad, etc.

10. Clasificación de RNA. SUPERVISADAS

10.1. Redes unidireccionales

Recordamos que consisten en encontrar una función a partir de un conjunto de entradas, que proporciona la salida deseada al problema planteado, por ejemplo en el caso concreto de las predicciones bursátiles, se trataría de encontrar la función que permita establecer la cotización de determinada empresa con respecto a las diversas variables de entrada (cotizaciones previas, tipos de Interés, inflación, etc.). Analizaremos algunas muy puntuales como pueden ser:

10.1.1. El perceptrón simple (ROSENBLATT, 1959)



Capa de entrada Capa de salida

Figura 5. Perceptrón simple

Es un modelo compuesto por dos capas de neuronas, como puede verse en la figura 5, que consta de:

- una de entrada, que no realiza ningún cómputo sólo envía información y
- otra de salida, que se activa con una función de tipo escalón.

Su utilización más común es como clasificador, responde 0 si no pertenece a la clase que representa y 1 si pertenece, solamente permite discriminar entre dos clases linealmente separadas, como puede observarse en la siguiente figura:

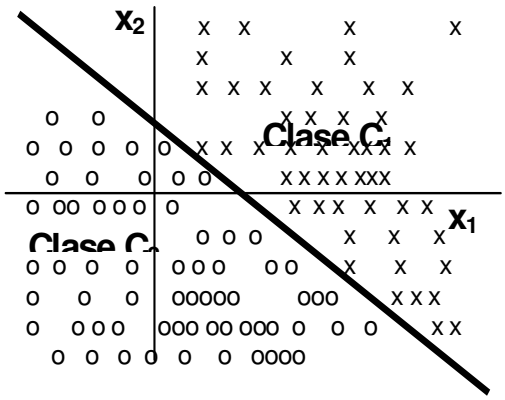


Figura 6. Clasificación

Justamente esta separación lineal representa la principal dificultad del perceptrón, por lo tanto su aprendizaje puede resultar tan exitoso como totalmente fallido de acuerdo al problema que esté intentando resolver.

Su importancia radica en el algoritmo de aprendizaje que le permite determinar automáticamente los pesos sinápticos que clasifican un conjunto de patrones etiquetados, este tipo de algoritmos se lo denomina por corrección de errores, ya que ajustan los pesos en proporción a la diferencia existente entre la salida actual y la deseada con la finalidad de minimizar el error actual.

Por otra parte se debe equilibrar la fijación del valor del ritmo de aprendizaje ya que uno pequeño puede implicar lentitud y uno alto puede ocasionar oscilaciones en el entrenamiento por producir variaciones en los pesos muy significativas.

10.1.2. La adalina (WIDROW, 1961)

Utiliza una neurona similar al perceptrón pero de respuesta lineal, con entradas que pueden ser continuas e incorpora un parámetro adicional, denominado *bias*, que si bien funciona como un umbral, no tiene la finalidad del disparo sino que proporciona un grado de libertad adicional.

Su utilidad es muy limitada por tratarse de un sistema lineal, es decir solamente podrá separar patrones de comportamiento lineal independiente.

El método consiste en proponer una función error y mediante un procedimiento de optimización (generalmente minimizar), generar una configuración de pesos, de tal forma que los modifique iterativamente hasta obtener el óptimo de la red.

El método de optimización más utilizado es el denominado descenso por el gradiente, es decir que partiendo en *t=0*, con cierta configuración de pesos sinápticos: *w(0)* y con una función de error *E(w)*, se calcula el sentido de la máxima variación de la función error.

Matemáticamente se expresa:

$$w(t+1) = w(t) - \varepsilon \nabla E(w),$$

donde ε indica el tamaño del peso tomado en cada iteración, que de manera ideal debería ser infinitesimal.

Gráficamente sería:

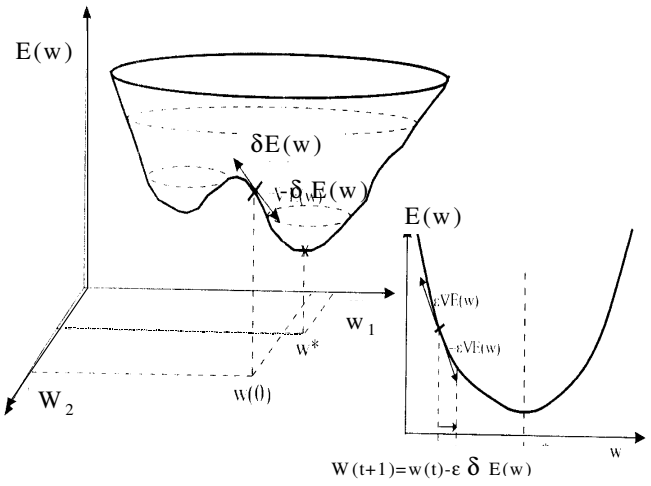


Figura 7. Superficie de error *E(w)*

10.1.3. El perceptrón multicapa (GRUPO PDP, 1986)

Consiste en agregar las capas ocultas al perceptrón simple, que presenta la siguiente figura:

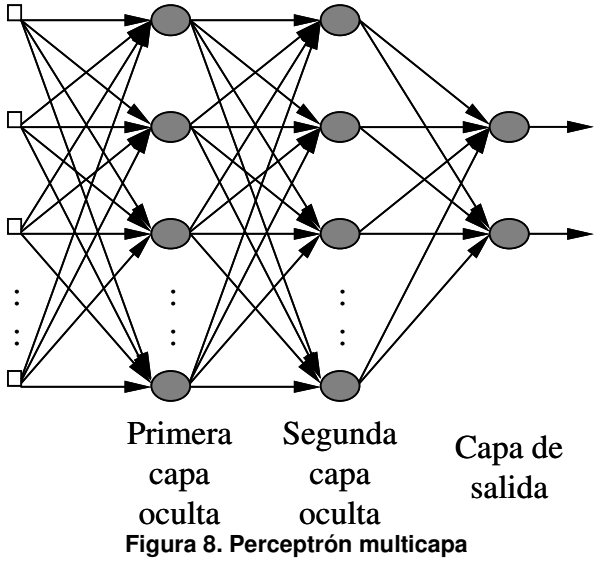


Figura 8. Perceptrón multicapa

Su principal ventaja consiste en la capacidad de representar complejas clasificaciones de manera eficaz y relativamente simple.

Utiliza como algoritmo de aprendizaje la retropropagación por errores, con funciones de activación de tipo sigmoideo, con minimización por el descenso por el gradiente (habrá uno para la capa de salida y otro para las capas ocultas), con actualización de umbrales y calculo final de la señal de error, proporcional al error de salida de la red actual.

El procedimiento podría resumirse de la siguiente manera:

- establecer de manera aleatoria los pesos y umbrales iniciales,
- para cada patrón del conjunto de aprendizaje se debe:
  - ejecutar para obtener la respuesta de la red,
  - calcular las señales de error asociadas y
  - calcular el incremento parcial de los pesos y umbrales
- calcular el incremento total actual de los pesos y umbrales, teniendo en cuenta todos los patrones,
- actualizar pesos y umbrales y
- calcular el error actual, volviendo al primer paso hasta obtener el resultado más satisfactorio.

Para este esquema, propuesto para el descenso por el gradiente, es importante comenzar con pesos aleatorios y llevar a cabo una fase de ejecución para todos y cada uno de los patrones del conjunto de entrenamiento, de tal forma de calcular la variación de pesos debida a cada patrón, acumularlos y finalmente actualizar, denominándose aprendizaje por lotes. Una variante es el aprendizaje en serie, que consiste en actualizar todos los patrones, en vez de acumular y después actualizar, en este caso el orden de ingreso de los patrones debe ser aleatorio para evitar el vicio a favor del último patrón si el orden fuera siempre el mismo.

En cualquiera de las dos propuestas anteriores, este método de gran generalidad, tiene una ventaja principal que es la de dar solución a múltiples problemas con buenos resultados y en corto plazo; justamente su principal desventaja es su lentitud a la convergencia, otro problema que plantea es que puede producir un sobreajuste, que está directamente relacionado con la generalización que debe producir toda red, pudiéndose solucionar mediante el cálculo del error de generalización como se verá en el punto siguiente.

10.2. Generalización

Se entiende por este término la capacidad que tiene la red de proporcionar una respuesta correcta ante patrones que no han sido entrenados.

En todo proceso de entrenamiento se puede calcular:

- el **error en el aprendizaje**, como el error cuadrático medio de los resultados de la red para el conjunto de patrones entrenados y
- el **error en la generalización**, que proviene del uso de nuevos patrones no entrenados en la red.

La representación de los errores en el test (validación) y en el aprendizaje (entrenamiento), como lo muestra la siguiente figura, nos mostraría el valor mínimo de la curva del error de validación que representa el error de generalización mínimo, es decir a partir de allí no sería necesario seguir entrenando la red si la finalidad es obtener la mejor generalización.

A partir de ese momento la red no ajusta correctamente los pesos, sino que memoriza los patrones del conjunto de entrenamiento, constituyendo lo que se denomina sobreaprendizaje, significando que la red está aprendiendo demasiado.

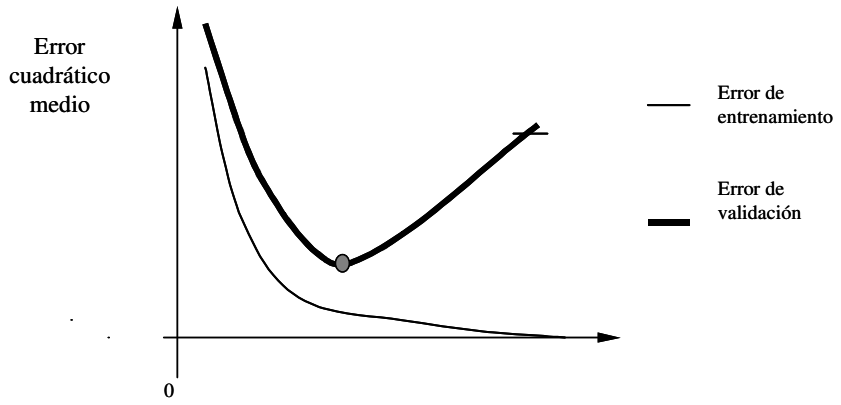


Figura 9. Evolución del error de aprendizaje y del error de generalización

También podría realizarse el entrenamiento y la validación en el mismo momento, para detenerse en el óptimo, a esta mecánica se la denomina validación cruzada y es muy utilizada en las redes neuronales supervisadas.

Finalmente cabe aclarar que si se deseara medir de manera totalmente objetiva la eficacia de estos procedimientos, el conjunto de patrones de partida debería dividirse en tres grupos:

- un conjunto de aprendizaje,
- uno para test, para la validación cruzada (evitando el sobreajuste) y
- un tercer grupo, también para testear, pero que se mantenga totalmente al margen del proceso de entrenamiento, permitiendo de esta forma validar objetivamente la red desarrollada.

11. Redes AUTOORGANIZADAS

La principal aplicación de estos modelos es a los problemas de clasificación, visualización de datos y representación de densidades de probabilidad.

11.1. Modelos neuronales NO SUPERVISADOS

En estos modelos no existe ningún parámetro externo que indique si la red está operando de manera correcta, pues no se dispone de una salida previa hacia la cual se deba llegar, debe descubrir por sí misma rasgos comunes, regularidades, correlaciones o categorías entre los datos de entrada e incorporarlos a su estructura interna de conexiones (pesos).

Los modelos no supervisados suelen son próximos a la biología, simples, tipo monocapa y con algoritmos rápidos y sencillos.

11.1.1. Modelo de mapas autoorganizados (KOHONEN, 1982)

Este modelo trata de reproducir el comportamiento de la información en la corteza cerebral, que con frecuencia aparece organizada espacialmente como por ejemplo el área sensorial, tiene una estructura unidireccional de dos capas como lo muestra la siguiente figura:

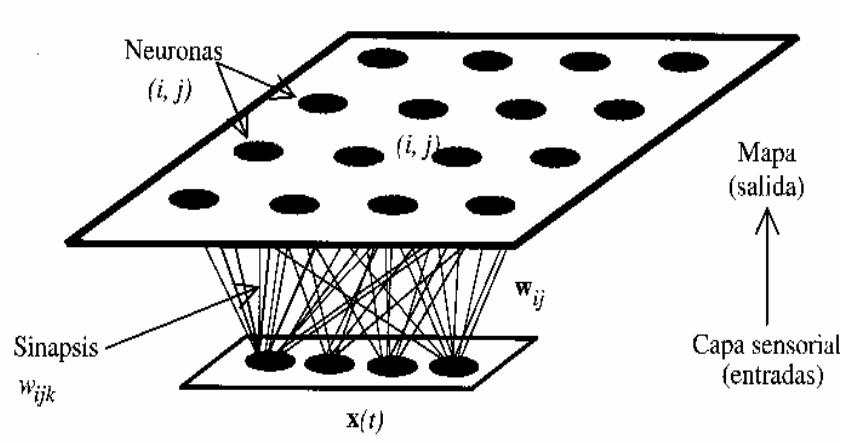


Figura 10. Arquitectura del modelo de mapas autoorganizados

La primera es la capa de entrada, que consiste en *m* neuronas, una por cada variable de entrada, su función es la de distribuir la información hacia la segunda capa, en ésta se realiza el procesamiento, es una estructura rectangular de neuronas que actúan en paralelo, que puede describirse como una matriz de procesadores en dos dimensiones que almacena un vector de pesos sinápticos, como puede verse en la siguiente figura:

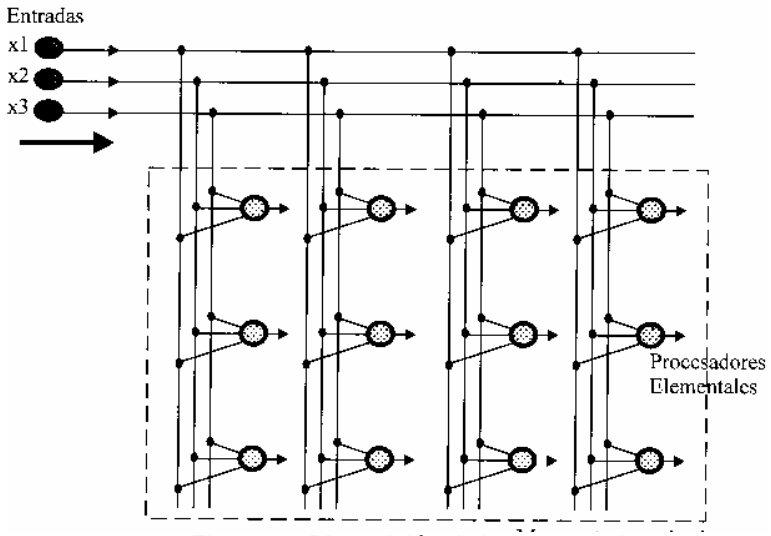


Figura 11. Disposición de los procesadores

En el modo de **operación normal** los pesos permanecen fijos, en primer lugar cada neurona calcula la similitud entre el vector de entrada y su propio vector de pesos, a través de cierta distancia una de ellas de declara vencedora por tener la menor distancia, es decir: ser la más similar, de esta forma la neurona actúa como detector de rasgos específicos indicando el tipo de rasgo en el vector de entrada.

En el **entrenamiento** cada neurona sintoniza con diferentes rasgos el espacio de entradas, tras la presentación y procesamiento del vector de entradas la ganadora

modifica sus pesos de manera de ser más intensa en futuros procesos ante el mismo patrón de entrada.

Esto parece ser un esquema competitivo clásico, pero la novedad de Kohonen es la incorporación de las relaciones entre las neuronas próximas en el mapa y lo hace mediante una función de vecindad, que de forma sencilla define un entorno alrededor de la neurona ganadora, reconociendo interacciones laterales y cuyo efecto principal es la actualización de los pesos de la ganadora, como también los pertenecientes a su vecindad.

En conclusión, lo que se realiza es una proyección no lineal de un espacio multidimensional de entrada sobre un espacio discreto de salida, representado por la capa de neuronas.

11.1.2. Algoritmo de aprendizaje

Al comenzar la vecindad comprende una amplia región del mapa, permitiendo un ordenamiento global de los pesos, con las futuras iteraciones el tamaño se va reduciendo hasta que finalmente sólo se actualiza el peso de la ganadora.

El algoritmo más habitual funciona de la siguiente manera:

- se parte de la definición de los pesos, que pueden ser nulos, aleatorios de poco valor o predeterminados;
- en cada iteración se presenta un patrón $x(t)$, definido de acuerdo a la función de distribución del espacio sensorial de entrada $p(x)$;
- cada neurona calcula la similitud (entre sus pesos, w_{ij} y los correspondientes al vector de entrada, x), una medida muy usada es la distancia euclídea de la siguiente manera:
  $d^2\left(w_{ij}, x\right) = \sum_{k=1}^{n}\left(w_{ijk} - x_k\right)^2$;
- determinar la neurona ganadora, con la menor distancia;
- actualizar los pesos de la ganadora y sus vecinas, la regla más empleada es la siguiente:
  $w_{ijk}(t+1) = w_{ijk}(t) + \alpha(t) \cdot h(|i-g|, t) \cdot \left(x_k(t) - w_{ijk}(t)\right)$,
  donde $\alpha(t)$ representa el ritmo de aprendizaje y la función $h(.)$ indica qué neuronas son consideradas vecinas, donde ***i*** es cualquier neurona y ***g*** la ganadora, dando un valor 0 si la neurona no pertenece a la vecindad y un número positivo si pertenece, además puede observarse que la vecindad es un conjunto centrado en la ganadora;
- si se ha alcanzado el máximo de iteraciones fijado el proceso termina con la última actualización de pesos, caso contrario se repiten nuevamente todos los pasos definidos.

El siguiente cuadro nos muestra los diferentes modelos de neuronas que podemos encontrar en este tipo de procesamiento, y sus respectivos algoritmos de entrenamiento:

Modelo de neurona	Función distancia	Regla de aprendizaje
1) Manhattan	$d\left(\mathbf{w}_{ij},\mathbf{x}\right)=\sum_{k=1}^{n}\left\lvert w_{ijk}-x_k\right\rvert$	$\Delta w_{ijk}(t)=\alpha(t)\cdot sign\left(x_k(t)-w_{ijk}(t)\right)$
2) Producto escalar		
2a) $\left\lVert\mathbf{x}\right\rVert=cte=1$	$d\left(\mathbf{w}_{ij},\mathbf{x}\right)=\sum_{k=1}^{n}w_{ijk}x_k$	$\Delta w_{ijk}(t)=\alpha(t)\cdot\left(x_k(t)-w_{ijk}(t)\right)$
2b) $\left\lVert\mathbf{x}\right\rVert\neq constante$	$d\left(\mathbf{w}_{ij},\mathbf{x}\right)=\sum_{k=1}^{n}w_{ijk}x_k$	$w_{ijk}(t+1)=\frac{w_{ijk}(t)+\alpha(t)\cdot x_k(t)}{\left\lVert\mathbf{w}_{ijk}(t)+\alpha(t)\cdot\mathbf{x}_k(t)\right\rVert}$
2c) $\left\lVert\mathbf{x}\right\rVert\neq constante$	$d\left(\mathbf{w}_{ij},\mathbf{x}\right)=\sum_{k=1}^{n}w_{ijk}x_k\equiv y_{ij}$	$\Delta w_{ijk}(t)=\alpha(t)\cdot\left(x_k-y_{ij}(t)\cdot w_{ijk}(t)\right)$
2d) $\left\lVert\mathbf{x}\right\rVert\neq constante$	$d\left(\mathbf{w}_{ij},\mathbf{x}\right)=\frac{\sum_{k=1}^{n}w_{ijk}x_k}{\left\lVert w_{ij}\right\rVert\cdot\left\lVert x\right\rVert}$	$\Delta w_{ijk}(t)=\alpha(t)\cdot\left(x_k(t)-w_{ijk}(t)\right)$
3) Euclídea	$d^2\left(\mathbf{w}_{ij},\mathbf{x}\right)=\sum_{k=1}^{n}\left(w_{ijk}-x_k\right)^2$	$\Delta w_{ijk}(t)=\alpha(t)\cdot\left(x_k(t)-w_{ijk}(t)\right)$

12. Otros modelos de redes neuronales

Simplemente para mencionarlos podemos encontrar los modelos híbridos y los realimentados, los primeros complementan entrenamientos supervisados con autoorganizados, el más habitual es el denominado **funciones de base radial**, como desarrollaremos más adelante; en cambio para los segundos se puede mencionar a los **modelos Hopfield discretos** (con entradas y salidas digitales), utilizado para generar memorias asociativas y en la eliminación de ruidos en imágenes y los **continuos** (de entradas y salidas analógicas), utilizado para problemas de optimización, que por ser apropiado para problemas contable-administrativos lo desarrollaremos brevemente.

12.1. Aplicaciones del modelo de Hopfield analógico. Optimización

Los problemas de optimización a los que puede aplicarse este modelo consisten en encontrar la mejor solución cumpliendo con ciertas condiciones restrictivas, las aplicaciones más usuales son: resolución de problemas de programación lineal, problemas de viajantes de comercio, entrenamiento de otra red, etc.

El método consiste en que cada neurona de salida representa una variable del problema a resolver, las condiciones o restricciones se incorporan como términos de la función de energía, donde cada mínimo de esta función representa una solución

localmente óptima del problema, es decir si se deja evolucionar una red construida con parámetros (pesos y corrientes) que conformen una función de energía, se alcanzará finalmente un mínimo local que representará una solución localmente óptima del problema.

12.2. Las funciones de base radial

Su estructura cuenta con tres capas de neuronas, de entrada, oculta y de salida; las de entrada como en la mayoría de los modelos únicamente envían información del exterior; las de salida son lineales esencialmente, calculan mediante la suma ponderada de las salidas que proporciona la capa oculta y la mayor dificultad se encuentra en la definición del modo de operar de las neuronas de la capa oculta, éstas operan sobre la base de la distancia que separa el vector de entradas con el vector sináptico que cada neurona almacena, denominado centroide, aplicando para este calculo una función radial con forma gaussiana, como puede observarse en la siguiente figura:

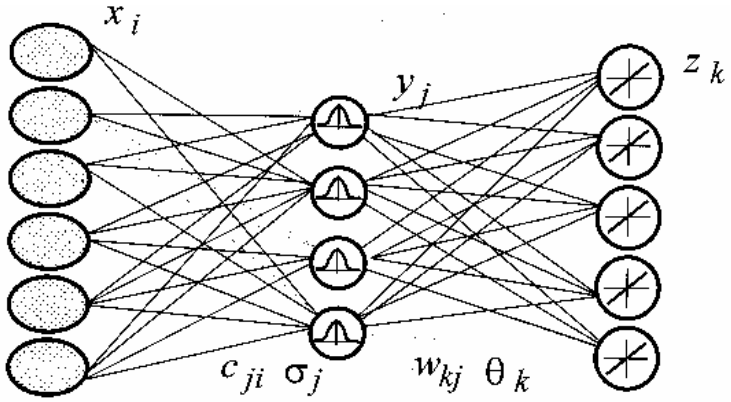


Figura 12. Arquitectura de una red de base radial y sus funciones de activación

12.2.1. El entrenamiento en las funciones de base radial

La **primera** cuestión que se plantea es cómo elegir la cantidad de nodos adecuados que cubran el espacio de entrada, como puede verse en la siguiente figura:

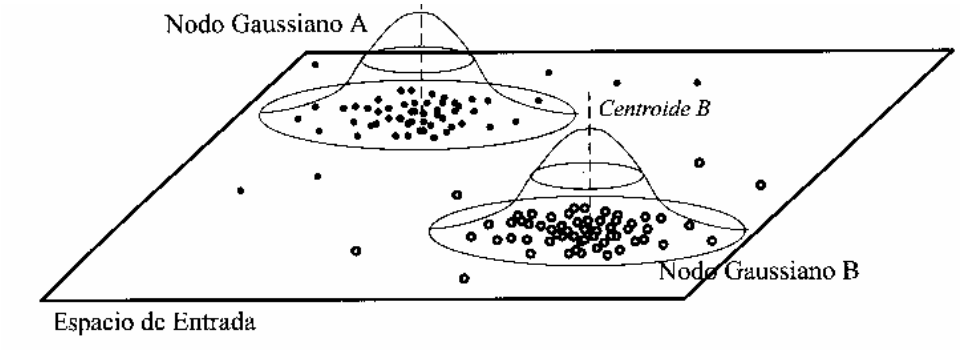


Figura 13. Dos nodos gaussianos cubren el espacio de entrada

Propuesto el número de nodos, se procede con el **segundo** paso que es un proceso que consiste en:

- elegir los valores de partida de cada nodo, este valor no es muy relevante para la salida final;
- en cada iteración se reparten todos los patrones de aprendizaje ***x***, entre todos los nodos, asignándolo a la neurona cuyo centroide dista menos;
- se definen nuevos centroides, como el promedio de los patrones de aprendizaje (N_j) del paso anterior de la siguiente manera:
  $c_{ij} = N_j \sum_{\mathbf{x} \in \text{ neurona } j} \mathbf{x}$ y
- finalmente si los nuevos centroides no han cambiado el algoritmo ha convergido, caso contrario comenzará nuevamente la iteración.

Por **último** se entrenan las neuronas de salida, completando el proceso con los umbrales definidos que se actualizan siguiendo el mismo esquema, considerando que se trata de pesos con entradas de valor –1.

13. Comparación con otras técnicas

Los siguientes cuadros tratan de comparar las RNA *con* la *Inteligencia Artificial* (IA), que las incluye y es entendida como ciencias de la computación –aprendizaje de la máquina ('machine learning')– y *con* la *Estadística*, las cuales no deben entenderse como excluyentes, sino más bien como complementarias:

13.1. Redes neuronales e inteligencia artificial

	Redes neuronales	**Inteligencia artificial**
Origen: ambas tratan de entender y emular la inteligencia humana	tienen su inspiración en la biología	tienen su inspiración en la psicología

Productos más conocidos	sistemas expertos con: • motor de inferencia, donde se programan las manipulaciones genéricas de tipo lógico a aplicar • base de conocimiento, donde se almacena la información a emplear *los razonamientos se obtienen aplicando las reglas almacenadas en la base de conocimiento, haciendo uso del motor de inferencia*	inteligencia computacional *el conocimiento emerge de las interconexiones de la estructura de la red de procesadores elementales*
Desarrollo	requiere esfuerzos en tiempo y dinero *debe recopilar las reglas sobre la base del conocimiento de expertos y posteriormente codificarlas*	trabajan en paralelo
Respuesta	es lenta, debe atravesar por frondosos árboles de decisión	en tiempos extremadamente pequeños
Resultados en problemas complejos, con pocas reglas de descripción y con un proceso masivo de información	incapaces de dar buenos resultados	grandes éxitos, con reducción de tiempo y costos
Resumiendo **operan**	de arriba hacia abajo (enfoque **descendente**) *con reglas, conceptos y cálculos secuenciales, tal como lo hace la parte izquierda de nuestro cerebro*	de abajo hacia arriba (enfoque **ascendente**) *interpretando de manera intuitiva y paralela los estímulos que llegan del exterior, como lo hace la parte derecha de nuestro cerebro*

13.2. RNA y Estadística

Se podría decir que las primeras tienen una serie de ventajas que pueden resumirse de la siguiente manera:

- resultan relativamente más fáciles de emplear,
- la interpretación de sus resultados resulta más factible para usuarios no expertos en ellas,
- normalmente no parten de restricciones respecto de los datos de partida,
- no suelen imponer presupuestos (como la distribución gaussiana por ejemplo),
- son de respuesta más rápida y
- la más importante es que producen errores menores.

	Redes neuronales	**Estadística**
Comprende	*un conjunto de técnicas de ajuste estadístico inspiradas en la biología*	*un conjunto de métodos que sirven para recoger, organizar, resumir y analizar datos, extrayendo conclusiones y tomando decisiones razonables*
Relaciones de aplicación entre ambas herramientas	• Perceptrón simple (nodo tipo umbral) • Perceptrón simple (nodo sigmoideo) • Adalina • Perceptrón multicapa • Aprendizaje no supervisado • Red simple de Kohonen • Funciones de base radial • Mapas de Kohonen	• Análisis discriminante • Regresión logística • Regresión lineal • Regresión no lineal simple y multivariada • Análisis de componentes principales • Análisis cluster mínimos cuadrados • Métodos de kernel regression • Escalas multidimensionales

SISTEMAS BORROSOS

1. Lógica borrosa o difusa *(fuzzy logic)*

Esta lógica nos permite operar con información imprecisa, como pueden ser las variables: temperatura, estatura y otras, seguidas por los atributos: alta, baja, etc. que permite expresarlas, en lo que se denominan, conjuntos borrosos.

Por otra parte estos conjuntos se basan en reglas para producir acciones, del tipo: *Si la temperatura es alta, entonces enfría mucho*; por ejemplo, si esto está definido para un sistema de control se producen uno o varios valores de salida mediante la incorporación de variables de entrada definidas en términos difusos.

Los sistemas basados en esta lógica surgen con Zadeh, en 1965, siendo su idea base la denominada *ley de incompatibilidad*, que expresa que *cuando la complejidad aumenta, las sentencias precisas pierden significado y las sentencias significativas pierden precisión*, como lo grafica la siguiente figura:

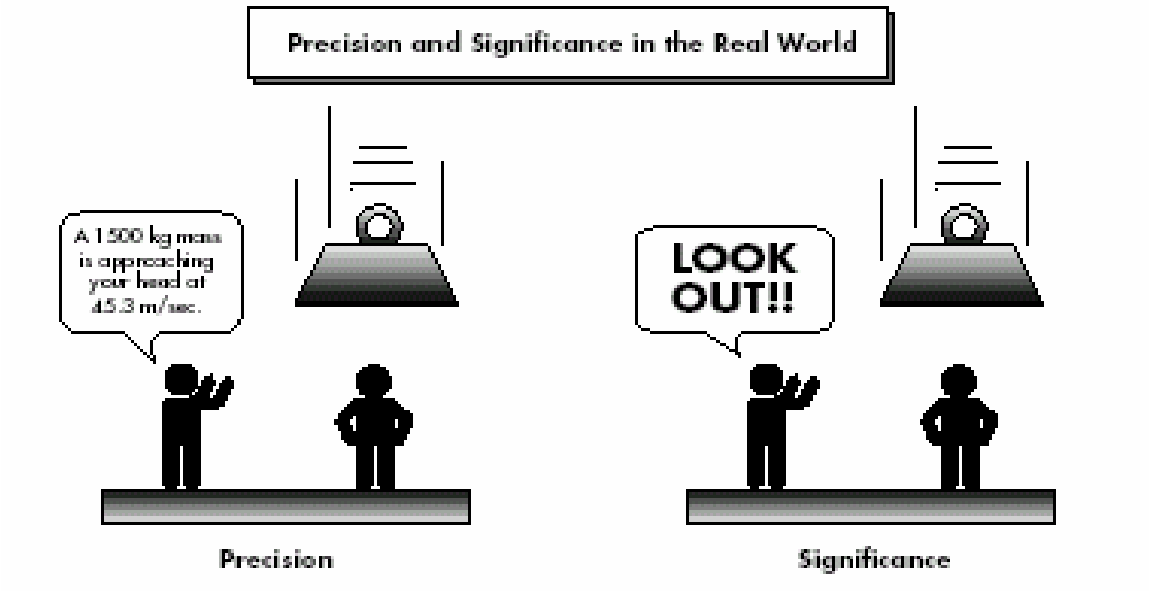


Figura 14. Diferencia entre sentencias precisas y significativas

Por supuesto, esta lógica puede aplicarse a los mismos problemas determinados para las redes, obteniendo mejores resultados en aquellos no lineales o no bien definidos, permitiendo aprender de los datos mediante el uso de algoritmos de aprendizaje.

La gran **diferencia con las redes** es la **incorporación del conocimiento de los expertos** en un tema determinado para el problema a resolver, ya sea como punto de partida para la optimización de un conocimiento ambiguo o directamente.

2. Introducción

Esta lógica en lugar de utilizar ecuaciones matemáticas complejas introduce el concepto de conjunto (o subconjunto) borroso, asociándolo a un determinado valor

lingüístico, que puede estar definido con una palabra, un adjetivo o con lo que se denomina etiqueta.

Además se define una función de pertenencia $\mu_A(x)$, que indica el grado en que la variable *X* está incluida en el concepto etiquetado con el nombre *A*, la siguiente figura muestra las distintas etiquetas que puede tomar una variable, las que se nombrarán conforme a las necesidades:

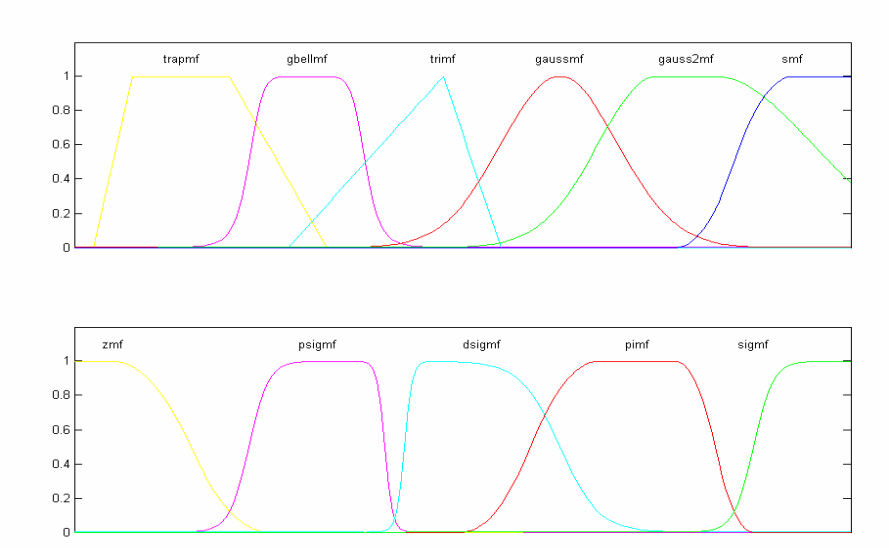


Figura 15. Etiquetas lingüísticas de la variable *X*

Siendo sus principales **beneficios**:

- la simplicidad, mediante la cual se elimina la dificultad del modelo matemático, permitiendo la descripción de entradas, salidas y reglas, en lengua natural;
- la utilización de menos códigos y memoria, considerando que puede definirse un sistema con unas pocas reglas;
- la mejor performance, ya que permite cálculos rápidos sin requerimiento de procesadores importantes;
- son de fáciles y rápidos desarrollos, donde el usuario se puede concentrar en su objetivo más que en el conocimiento de la matemática;
- confiables, considerando que a pesar del gran tamaño las reglas funcionan autónomamente, eliminando la disyuntiva entre robustez y sensibilidad; finalmente
- permite nuevas aplicaciones consideradas hasta ahora muy complejas.

Y las **desventajas** observadas son:

- la necesidad de adoptar una nueva forma de plantear los problemas;
- el desconocimiento inicial del comportamiento de variables de entrada-salida;
- la carencia por sí mismas de mecanismos de optimización del modelo y finalmente
- la definición de las reglas no es siempre obvia.

3. Conjuntos borrosos

En un conjunto tradicional algo está incluido totalmente en él o no lo está en absoluto, esta situación se puede indicar asignando el 1 a los individuos que pertenecen al conjunto y 0 a aquellos que no pertenecen, llamándose a esta asignación función de pertenencia o inclusión, graficándose como lo muestra la siguiente figura:

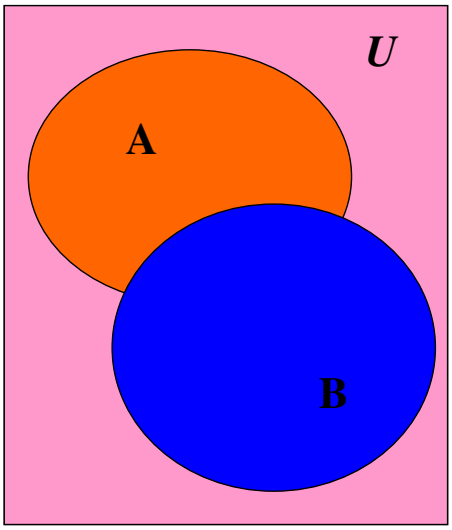


Figura 16. Conjuntos clásicos

En un conjunto borroso *F,* definido en el universo *U*, queda caracterizado por la función de pertenencia μ_F, que toma valores en el intervalo $[0,1]$ y puede representarse de la siguiente manera:

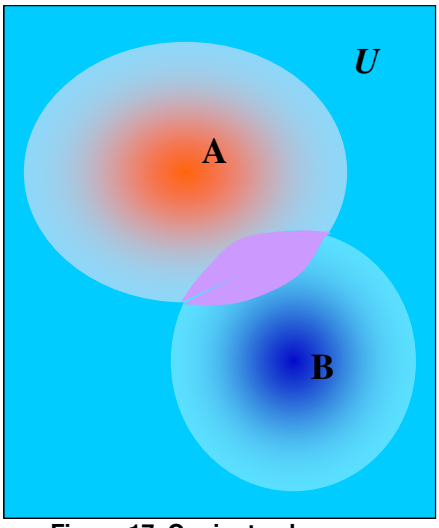


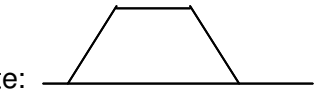
Figura 17. Conjuntos borrosos

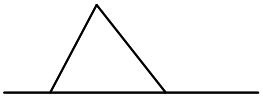
Además dado un conjunto borroso *F*, se definen los siguiente conceptos:

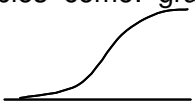
- **Conjunto soporte**, es el tradicional que contiene todos los valores de *U* con función de pertenencia diferente de cero
- **Conjunto α-corte**, en simbología F_α y contiene todos los puntos de *F* con función de pertenencia mayor que *α*
- **Conjunto normalizado**, es aquel conjunto borroso cuya mayor función de pertenencia es 1

3.1. Funciones de inclusión de conjuntos borrosos

Las funciones de inclusión o de pertenencia consisten en un conjunto de pares ordenados si la variable es discreta y de una función continua si no lo es, encontramos las siguientes:

- **de tipo trapezoidal**, definida por cuatro puntos: *a*, *b*, *c* y *d*, siendo cero para los menores valores *a* y los mayores valores *d*; es 1 entre los valores *b* y *c* y finalmente toma valores del intervalo $[0,1]$ entre *a* y *b*, y entre *c* y *d*. Su representación gráfica es la siguiente:  y su utilización es para sistemas borrosos sencillos, porque permite definir conjuntos con pocos datos y cálculos (por ejemplo para variables como: adulto, normal, adecuada, etc.)

- **de tipo triangular**, partiendo de la anterior es adecuada para modelar propiedades con valor de pertenencia diferente de cero para un rango de valores alrededor de un solo punto, es decir cuando *b*=*c* y su representación sería: 

- **de tipo S**, también parte de la trapezoidal haciendo: *c*=*d*=*max(U)*; es adecuada para modelizar variables como: grande, mucho, positivo, etc., y se representa con la siguiente figura: 

- **de tipo singleton**, para la cual todos los valores de la trapezoidal son iguales

3.2. Variable lingüística

Se denomina variable lingüística aquella que toma valores del lenguaje natural, como puede ser: mucho, positivo, caliente, etc. que juegan el papel de etiquetas para cada conjunto borroso.

Formalmente la podemos definir como: $(A,T(A),U,G,M)$, donde asociándolo al término *temperatura*, a título de ejemplo, cada elemento representa:

- A el nombre de la variable, *A=temperatura*;
- $T(A)$ el conjunto de términos que nombran los valores de *X que puede tomar* A, que son conjuntos borrosos en el universo U, serán los términos necesarios para describir en un problema la temperatura: *muy fría, fría, templada, cálida, etc.*;
- U el universo de discurso de la variable *X*, si bien esta variable puede tomar valores desde el cero absoluto hasta el infinito, habrá que limitarla al problema específico, definiendo un rango, por ejemplo: *0 °a 50 °*;
- G una regla sintáctica para la generación de los nombres de X y
- M una regla semántica para asocial el significado a cada valor.

3.3. Particiones borrosas

Una partición de A es uno de los subconjuntos borrosos que pueden formarse con los elementos de $T(A)$ y que se identifican con una etiqueta, por ejemplo: (Bajo, Medio, Alto) y con una función de pertenencia propia, por ejemplo: $\{\mu_{Bajo}(t),\mu_{Medio}(t),\mu_{Alto}(t)\}$, bajo estas condiciones hablamos de una partición completa si los conjuntos definidos cubren todo el U y decimos que los conjuntos están solapados si su intersección no es nula.

Los nombres de las particiones suelen identificarse con sus iniciales, así por ejemplo (Negativo Grande, Negativo Chico, Cero, Positivo Chico, Positivo Grande), sería (NG, NC, C, PC, PG).

3.4. Medidas borrosas

Para todo conjunto borroso pueden definirse ciertas medidas, la principal es la borrosidad, que justamente mide lo borroso que es un conjunto e implica determinar la distancia de A al conjunto discreto de los valores de X en los que $\mu_A(x) \succ 0$ y que podríamos denominar C.

Las medidas más concretas para determinar la distancia entre dos conjuntos borrosos son y se calculan de la siguiente manera:

- **Hamming**, $d(A)=\sum\left|\mu_A(x)-\mu_C(x)\right|$
- **Euclídea**, $d(A)=\left(\sum\left(\mu_A(x)-\mu_C(x)\right)^2\right)^{1/2}$

- **Minkowski**, $d(A)=\left(\sum\left(\mu_A(x)-\mu_C(x)\right)^w\right)^{1/w}$ con $w\in[1,\infty]$

Otra forma de cálculo es la **similitud**, la cual mide cuán parecido es un conjunto a otro y la **entropía** que informa sobre cuánta información aporta el conjunto a la descripción de la variable y se calcula como:

$$d(A)=-\sum\left\{\mu_A(x)\log\mu_A(x)+\left[1-\mu_A(x)\right]\log\left[1-\mu_A(x)\right]\right\}$$

Por último tenemos el **agrupamiento borroso**, se basa en la medición de las distancias euclídeas entre vectores y se utiliza para determinar reglas borrosas que definen un sistema desconocido o caja negra, uno de los métodos más comunes de agrupamiento es el denominado *k-medias*.

3.5. Operaciones borrosas

Se debe dejar constancia que la teoría clásica de conjuntos puede considerarse un caso particular de la borrosa, por lo tanto las operaciones básicas pueden definirse y calcularse como se muestra en el siguiente cuadro:

	Calculo	Rango
Igualdad	$\mu_A(x)=\mu_B(x)$	$x\in U$
Unión	$\mu_{A\cup B}(x)=\max\left[\mu_A(x),\mu_B(x)\right]$	$\forall x\in U$
Intersección	$\mu_{A\cap B}(x)=\min\left[\mu_A(x),\mu_B(x)\right]$	$\forall x\in U$
Complemento	$\overline{\mu_A}(x)=1-\mu_A(x)$	$x\in U$
Norma	$\mu_{Norma(A)}(x)=\dfrac{\mu_A(x)}{\max\left[\mu_A(x)\right]}$	$x\in U$
Concentración	$\mu_{Conc(A)}(x)=\left(\mu_A(x)\right)^2$	$x\in U$
Dilatación	$\mu_{Dilat(A)}(x)=\left(\mu_A(x)\right)^{0.5}$	$x\in U$

De estas operaciones vale la pena recordar que la concentración y dilatación se conocen como modificadores, que aplicados sobre el lenguaje común significa utilizar el *"muy" o "más o menos"*, concretamente para el conjunto borroso *F* en *U*, si *F=joven*, el conjunto *muy joven* sería: $\mu_{nuy\ F}(u)=\left(\mu_F(u)\right)^2$ y el conjunto definido como *más o menos joven* se calcularía: $\mu_{mas\ o\ menos\ F}(u)=\left(\mu_F(u)\right)^{0.5}$. Gráficamente puede observarse en la siguiente figura:

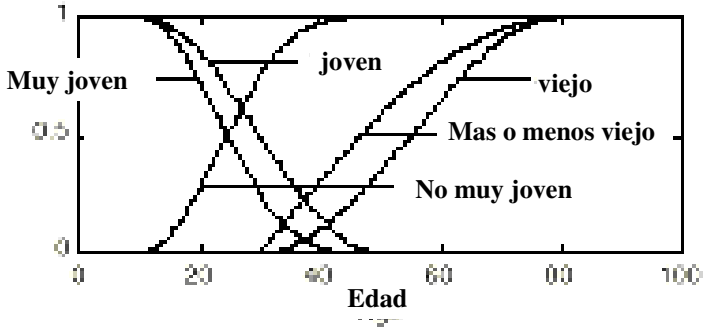


Figura 18. Modificadores de pertenencia

3.6. Inferencia borrosa

Como la lógica clásica, la borrosa se ocupa del razonamiento formal, siendo su principal ventaja que puede tomar valores intermedios entre verdadero y falso.

Utiliza reglas del tipo Si-Entonces (If-then), mediante las cuales se expresa una relación borrosa, que si bien no proporciona un razonamiento preciso sino aproximado, implica que se puede inferir, de una regla, una conclusión, aunque la premisa no se cumpla totalmente.

Existen dos métodos básicos de inferencia:

- *Modus ponens* generalizado, conocido como razonamiento directo, donde:

  (Conocimiento) *Si x es A Entonces y es B*
  (Hecho) *x es A'*
  (Consecuencia) *y es B'*

  Si *A, A', B y B'* son conjuntos borrosos se pueden expresar las siguientes consecuencias:

	x es A'	*y es B'*
criterio 1	*x es A*	*y es B*
criterio 2-1	*x es muy A*	*y es muy B*
criterio 2-2	*x es muy A*	*y es B*
criterio 3-1	*x es mas o menos A*	*y es mas o menos B*
criterio 3-2	*x es mas o menos A*	*y es B*
criterio 4-1	*x no es A*	*y es desconocido*
criterio 4-2	*x no es A*	*y no es B*

- *Modus tolens* generalizado, conocido como razonamiento inverso y se resume como:

(Conocimiento) *Si x es A Entonces y es B*
(Hecho) *y es B'*
(Consecuencia) *x es A'*

Siendo las posibles consecuencias:

	y es B'	*x es A'*
criterio 5	*y no es B*	*x no es A*
criterio 6	*y no es muy B*	*x no es muy A*
criterio 7	*y no es mas o menos B*	*x no es mas o menos A*
criterio 8-1	*y es B*	*x es desconocido*
criterio 8-2	*y es B*	*x es A*

3.7. Implicación borrosa

Se definen dos conjuntos borrosos A y B en U y V respectivamente, una implicancia borrosa de A en B se indica $A \rightarrow B$ y puede venir definida por alguna de las siguientes funciones de inclusión:

- **Conjunción borrosa** $\mu_{A\rightarrow B}(u,v) = \mu_A(u) * \mu_B(v)$
- **Disyunción borrosa** $\mu_{A\rightarrow B}(u,v) = \mu_A(u) \dot{+} \mu_B(v)$
- **Implicación material** $\mu_{A\rightarrow B}(u,v) = \mu_{\bar{A}}(u) \dot{+} \mu_B(v)$
- **Cálculo proposicional** $\mu_{A\rightarrow B}(u,v) = \mu_{\bar{A}}(u) \dot{+} \mu_{A*B}(v)$
- **Modus ponens generalizado** $\mu_{A\rightarrow B}(u,v) = \sup\{c \in [0,1] \mid \mu_A(u) * c \leq \mu_B(v)\}$
- **Modus tolens generalizado** $\mu_{A\rightarrow B}(u,v) = \inf\left\{c \in [0,1] \mid \mu_B(v) \dot{+} c \leq \mu_A(u)\right\}$

4. Reglas borrosas

Las reglas combinan uno o más conjuntos de entrada con una salida, permitiendo expresar el conocimiento disponible de una relación, generalmente se necesitan varias reglas formando la denominada *base de reglas*, que representan las relaciones conocidas entre entradas y salidas.

Esta base puede confeccionarse como una tabla o como memoria asociativa, que permite representar mediante una clara gráfica la relación entre dos variables de entrada y la correspondiente salida, como se visualiza en la siguiente figura.

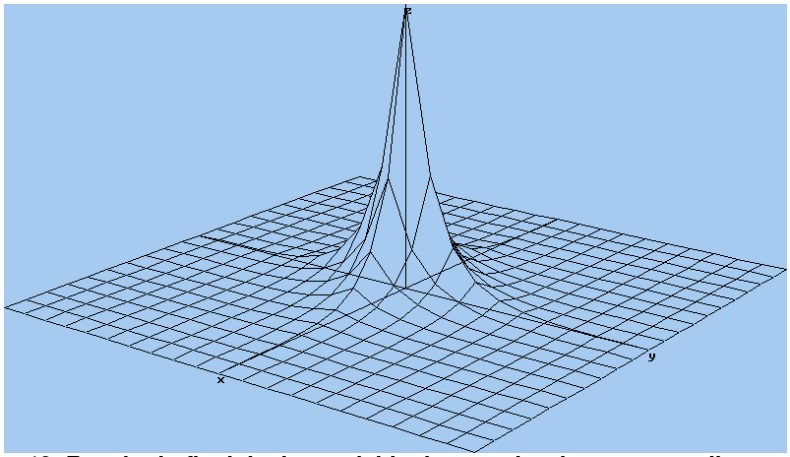


Figura 19. Resultado final de dos variable de entrada y la correspondiente salida

Esta base de reglas se la conoce como $R^{(l)}$ y pueden ser de dos tipos:

- **de tipo Mamdani**, con la siguiente estructura:

  $R^{(l)}$: *si* x_1 *es* F_1^l *y* ... *y* x_n *es* F_n^l *entonces y es* G^l, donde:

  F_i^l *y* G^l son conjuntos borrosos definidos en $U_i \subset \Re$ *y* $V \subset \Re$, respectivamente y

  $x = (x_1,\ldots,x_n)^{\mathrm{T}} \in U_1 \times \ldots \times U_n$ e $y \in V$ son variables lingüísticas

  estas reglas permiten expresar el conocimiento previo disponible sobre el problema, incluso el adquirido en un proceso de optimización

- **de tipo Sugeno**, donde la función de salida es una combinación lineal (o en caso más general, una función genérica) de las variables de entrada, siendo su estructura:

  $R^{(l)}$: *si* x_1 *es* F_1^l *y* ... *y* x_n *es* F_n^l *entonces* $y^l = f^l(x)$

  estas reglas simplifican los cálculos de salida, pero no resultan tan adecuadas para expresar conocimiento de expertos

4.1. Dispositivos de inferencia borrosa

Son aquellos dispositivos que tratan de interpretar las reglas con la finalidad de obtener un valor de salida a partir de las variables lingüísticas de entrada, si para simplificar tenemos que:

$$F_1^l \times \ldots \times F_n^l \equiv A \text{ y } G^l \equiv B$$

podemos expresar las implicancias de la ecuación general con las siguientes relaciones:

- **por la regla del mínimo** $\mu_{A\to B}(x,y)=\min\left[\mu_A(x),\mu_B(y)\right]$
- **por la regla del producto** $\mu_{A\to B}(x,y)=\mu_A(x)\mu_B(y)$
- **por la regla aritmética** $\mu_{A\to B}(x,y)=\min\left[1,1-\mu_A(x)+\mu_B(y)\right]$
- **por la regla del max-min** $\mu_{A\to B}(x,y)=\max\left\{\min\left[\mu_A(x),\mu_B(y)\right],1-\mu_A(x)\right\}$
- **por la regla booleana** $\mu_{A\to B}(x,y)=\max\left\{1-\mu_A(x),\mu_{\ B}(y)\right\}$
- **por la regla de Goguen** $\mu_{A\to B}(x,y)=\begin{cases}1 & \mu_A(x)\le\mu_B(y)\\ \mu_B(y)/\mu_A(x) & \mu_A(x)\succ\mu_B(y)\end{cases}$

4.1.1. Mecanismos de inferencia

Estos mecanismos permiten decidir qué reglas son disparadas, de manera independiente de la base de reglas, pudiéndose ser implementados en cualquier lenguaje de programación.

Los más tradicionales pueden definirse como:

- *encadenamiento hacia atrás* (forward chaining), es decir en control se parte de un conjunto de datos de entrada para obtener una salida, disparándose todas las reglas involucradas obteniéndose varias salidas y se tiene la necesidad de agrupar las distintas salidas, Respuesta (salida) → Datos (entrada) y
- *encadenamiento hacia adelante* (backward chaining), Respuesta (salida) ← Datos (entrada).

En conclusión, la salida final puede ser:

- M conjuntos borrosos que son los resultantes de aplicar la entrada A' a cada una de las reglas de la base,
- un único conjunto borroso que es la unión de los M conjuntos borrosos, o
- M escalares, si las reglas son del tipo Sugeno, cada uno de los cuales es el resultado de aplicar de entrada A' a cada una de las reglas de la base.

4.2. Borrosificador o fuzzificación

Establece una relación entre puntos de entrada al sistema no borrosos, en símbolos: $x=(x_1,\ldots,x_n)^{\mathrm{T}}$ y sus correspondientes conjuntos borrosos A en U, es decir todas las variables exteriores deberán borrosificarlas previamente y la forma de hacerlo puede ser:

- **singleton**, es el más utilizado y para cada valor de entrada x, se define un conjunto A' que lo soporta con su correspondiente función de pertenencia $\mu_A(x')$, de modo

que $\mu_A(x)=1, \quad (x'=x)$ y $\mu_A(x')=0$, para todos los otros $x' \in U$ en los que $x' \neq x$

- **no singleton**, se utiliza una función exponencial con forma de campana del tipo:

$$\mu_{A'}(x') = a \cdot \exp\left[-\left(\frac{x'-x}{\sigma}\right)^2\right]$$

4.3. Desborrosificador o defuzzificación

Es la función que transforma un conjunto borroso definido en V, salida de un dispositivo de inferencia, en un valor no borroso, para lo cual existen diferentes métodos, como por ejemplo: *máximo*, *media de centros* o *centro de áreas*, cuando se utilizan reglas de tipo Mamdani; en cambio si las reglas son tipo Sugeno el valor de salida no borroso se utiliza como *media ponderada* de las salidas de cada regla de la base.

4.4. Estructura de un sistema borroso

En resumen, la estructura de un sistema borroso puede representarse esquemáticamente mediante la siguiente figura:

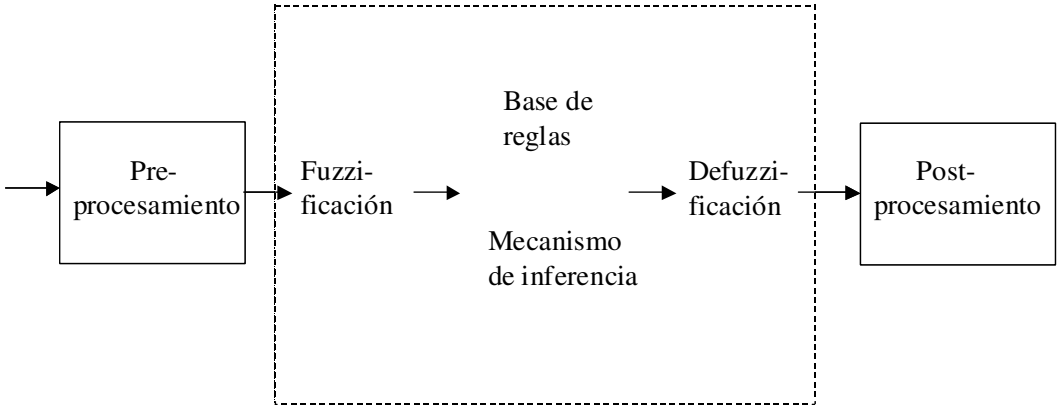


Figura 20. Estructura de un sistema borroso

4.4.1. Desarrollo de sistemas borrosos

Existen muchas posibilidades de aplicación de estos sistemas pero en términos generales son ideales para:

- problemas complejos, con muchas variables lingüísticas o muchas reglas, necesitando eficiencia computacional, siendo preferibles para estos casos la

utilización de funciones triangulares o trapezoidales, con funciones de tipo singleton para las salidas (aunque son más sensibles al ruido de las entradas) y

- en aplicaciones en las que se requiere que el sistema pueda desarrollar un aprendizaje, es decir se necesita facilidad de adaptación, utilizando la función de salida $y = f(x)$, con preferencia de funciones de inclusión exponenciales.

5. Borrosidad y probabilidad

No debe confundirse la función de pertenencia de un conjunto borroso con la función de densidad de probabilidad, ya que la primera indica hasta qué punto cierto valor de una variable puede ser incluido en un conjunto borroso, mientras que la segunda indica la frecuencia con que los diversos valores de una variable se presentan.

Por ejemplo: una probabilidad 0.20 de *clientes satisfechos* por el servicio de pensión de un hotel implica que 20 de cada 100 pasajeros se encuentran satisfechos con los servicios de pensión recibidos, mientras que una pertenencia de 0.20 al conjunto de *clientes satisfechos* significa que nuestro pasajero, en una escala de 0 (totalmente insatisfecho) a 1 (totalmente satisfecho), percibe su sensación de satisfacción en un 0.20 (lo cual podría interpretarse como muy baja).

6. Sistemas de control borroso

6.1. Características de los sistemas

Estos sistemas de control tienen su mayor aplicación en los electrodomésticos, procesos industriales y robots, siendo su principal característica que la entrada no necesariamente tiene precisión numérica, puede ser: vaga, ambigua, imprecisa, ruidosa o incompleta. Su estructura básica puede esquematizarse de la siguiente manera:

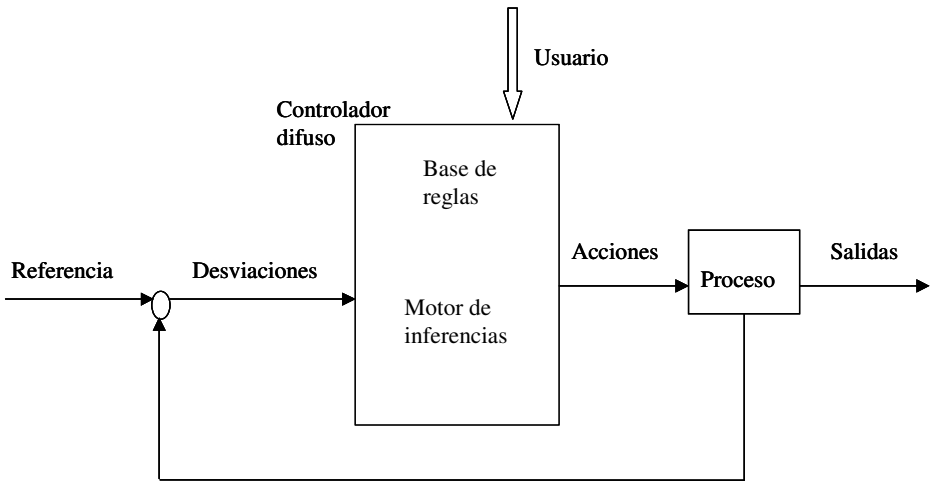


Figura 21. Estructura de un sistema de control borroso

6.2. Definiciones de un sistema de control borroso

Para establecer un sistema de control es necesario:

- definir el objetivo del controlador, para lo cual habrá que contestar algunas preguntas como por ejemplo: ¿qué trato de controlar?, ¿qué tengo que hacer para controlarlo?, ¿qué clase de respuesta espero?, ¿qué posibles fallas debo considerar?, etc.;
- determinar las variables de entrada/salida, estableciendo: los nombres, que deberán estar expresados con sustantivos; y la cantidad mínima de variables, recordando que el aumento complica el sistema y no necesariamente lo hace más eficiente;
- fijar los términos difusos para cada variable, es decir los adjetivos a ser aplicados y su cantidad mínima, por ejemplo: 3;
- establecer las reglas de control, aplicando los términos borrosos;
- de ser necesario también determinar los pre y post procesamientos de las variables y finalmente
- testear el sistema, verificando si el resultado es satisfactorio, en caso contrario producir el ajuste de las funciones de pertenencia y controlando la validez de las reglas.

6.3. Tipos de sistemas de control

6.3.1. Controladores borrosos directos sin optimización

Los controladores directos permiten realizar control de sistemas utilizando una descripción lingüística de las reglas, que han de obtenerse del conocimiento que se dispone de ella o de procedimientos heurísticos, un primer bloque realiza el preprocesado de las variables de entrada para generar las entradas al controlador borroso.

6.3.2. Controladores borrosos directos con optimización

Cuando se precisa una solución más eficiente, o no se dispone de conocimiento previo se utilizan controladores borrosos directos con optimización.

Un primer bloque realiza la evaluación del funcionamiento del controlador, para generar posibles modificaciones en un segundo bloque, denominado de evaluación o de ajuste que permite el cálculo de la sobreoscilación del sistema en torno al punto de trabajo o el tiempo que tarda en estabilizarse tras una variación.

Según el tipo de ajuste nos encontramos con controladores:

- **autoorganizados**, estos sistemas son capaces de modificar automáticamente y sin intervención humana su base de conocimiento, a partir de discrepancias medidas con criterios prefijados, estableciendo los ajustes a la base de reglas;

- **con autoaprendizaje**, en estos sistemas no se modifican los términos lingüísticos sino que se modifican los parámetros escalares, tanto del controlador como de los mecanismos de pre o postproceso y finalmente
- **basados en modelado borroso**, que precisan una serie de simplificaciones sobre los parámetros relativos a antecedentes y consecuentes, básicamente actúan como modeladores de la dinámica inversa del proceso.

6.3.3. Controladores borrosos híbridos

Estos sistemas están formados por dos controladores interconectados, uno convencional que se encarga básicamente del control garantizando un comportamiento estable, otro borroso que actúa en paralelo introduciendo el componente heurístico en el proceso, también puede emplearse para el ajuste de los parámetros del controlador, orientándose a mejorar cierta característica (por ejemplo, la reducción de oscilaciones, o la mejora del tiempo de establecimiento, entre otras).

7. Aprendizaje en sistemas borrosos

Existen numerosos algoritmos de aprendizaje aplicables a estos sistemas, en general todos los vistos para redes neuronales complementándose de una manera muy eficiente, por eso usualmente a estos sistemas se los denomina neuroborrosos y son actualmente un campo de intenso trabajo, entre ellos encontramos:

- **retropropagación**,
- **algoritmos genéticos**
- **mínimos cuadrados ortogonales**
- **tablas de búsqueda**
- **agrupamiento**

7.1. Retropropagación

Se lo utiliza en los sistemas con borrosificador singleton, funciones gaussianas, implicación por producto y desborrosificador por media de centros, se lo puede representar como una red unidireccional y aprovecha la capacidad de aprendizaje de la red neuronal para optimizar su funcionamiento.

Por otra parte, estos mecanismos de aprendizaje podrían emplearse para extraer las reglas que una red neuronal ha encontrado en el entrenamiento, eliminando uno de los grandes problemas de las RNA, su operación como caja negra.

7.2. Algoritmos genéticos

Están inspirados en los métodos de solución de problemas que ha utilizado la naturaleza para hacer evolucionar a los seres vivos, adaptándolos a los diferentes

entornos, consiste principalmente en la codificación de las características de los seres vivos en el genoma y su evolución a través de la reproducción sexual y las mutaciones.

El proceso de optimización comienza con la inicialización, que consiste en generar una población de individuos, para los cuales existen dos interpretaciones:

- la que entiende por población un conjunto de individuos, donde cada individuo será un controlador completo, en este caso definimos como: *genoma*, al parámetro que define a todos y cada uno de los individuos de la población; *genotipo*, a la parte del genoma que describe a un individuo concreto; *fenotipo*, a la expresión del genotipo y *gen*, a cada uno de los parámetros que describen a un individuo
- la que consiste en considerar como individuo las habilidades de un controlador, de forma que la población representa el conjunto de todas las habilidades, siendo este concepto el utilizado en robótica

Siguiendo la primer interpretación de individuo, a partir de la etapa de **inicialización** se puede generar una población, de manera aleatoria o partiendo de soluciones desarrolladas por otros procedimientos.

La siguiente fase, denominada **evaluación**, es la que más tiempo requiere y consiste en dejar que cada individuo de la población actúe controlando el sistema, a partir de ella se decide qué comportamientos no se han de potenciar, cuáles sí y en qué medida.

A continuación sigue la fase de **selección**, es decir establecer qué individuo trasmitirá su genotipo a la generación siguiente, existiendo diversas alternativas:

- **sólo el mejor**, se selecciona el individuo que tiene la mejor evaluación y únicamente su genoma es trasmitido a la próxima generación, permitiendo una rápida convergencia a la solución, pero puede estancarse en un mínimo local o tender a especializarse en las cualidades del individuo seleccionado;
- **sólo los mejores**, en este método se evita la especialización mencionada anteriormente ya que se seleccionan, los *n* individuos mejor evaluados y finalmente
- **todos** los individuos son seleccionados, mecánica utilizada cuando se pretende buscar soluciones a problemas con un gran espacio de búsqueda y para los que no se conoce una buena aproximación.

De esta forma con el nuevo genoma final, se expresa un fenotipo, reconstruyendo cada uno de los controladores que forman la población y se procede a una nueva evaluación, así sucesivamente hasta un número finito de veces o hasta que la evaluación se estabilice.

7.2.1. Qué optimizar

Es importante definir qué se pretende optimizar con este procedimiento, encontrando entre otros: **coeficientes (**ganancias de entrada y salida, particiones de

variables de entrada y salida), **reglas**, **estructura de base de reglas** o **completa** (codificación desordenada).

La optimización de coeficientes consiste en seleccionar un conjunto inicial de parámetros escalares y limitar la optimización a encontrar la mejor combinación de estos valores sin modificar la base de reglas, de esta forma el espacio de las entradas queda dividido en tantos subespacios borrosos como valores lingüísticos se defina en la base y cada conjunto borroso de una partición de entrada afecta al comportamiento de una amplia región.

En el caso de optimización de las reglas, la influencia es menor ya que supone modificar las premisas de las reglas sin cambiar la definición de las particiones, lo que permite ajustar el comportamiento en cada una de las regiones controladas.

Cuando hablamos de optimizar la estructura consiste en optimizar el comportamiento del controlador y finalmente la completa implica una optimización simultánea de todos los aspectos vistos, para lo cual es de especial significación el tipo de codificación empleada.

8. Aplicaciones de los sistemas borrosos

Estos sistemas permiten utilizar el conocimiento que disponen los expertos sobre un tema, bien directamente, bien como punto de partida para una optimización automática, es decir permite formalizar tanto el conocimiento ambiguo de un experto como el sentido común de una forma tecnológicamente realizable.

Una importante ventaja de estos sistemas es que, por la simplicidad de sus cálculos, normalmente pueden ejecutarse en sistemas económicos y rápidos, no obstante las ventajas o desventajas que presentan comparativamente los diferentes sistemas, la tendencia actual es la combinación de distintas técnicas para resolver los problemas complejos.

Finalmente, queda por decir que la lógica borrosa tiene una corta historia, pero un crecimiento rápido, en virtud de su capacidad para resolver problemas relacionados con la incertidumbre de la información o del conocimiento de los expertos.

Bibliografía

- (Kauffman, A. y Gil Aluja, J., 1995)
- (del Brío, B.M. y Sanz Molina, A., 1997), *Redes neuronales y sistemas borrosos, Introducción teórica y práctica*, RA-MA Editorial