



INTRODUCCION A LAS REDES NEURONALES (II)

César Hervás-Martínez
Grupo de Investigación AYRNA

Departamento de Informática y
Análisis Numérico
Universidad de Córdoba
Campus de Rabanales. Edificio
Einstein.
Email: chervas@uco.es

2019-2020

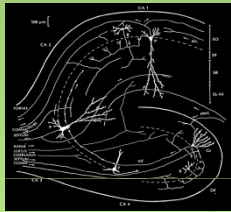
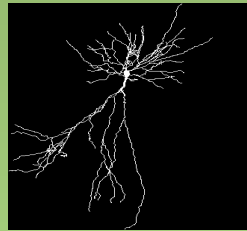
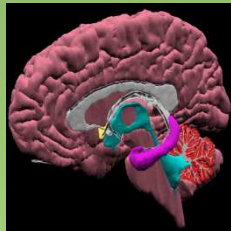


Modelado Biofísico y Neurociencia Computacional

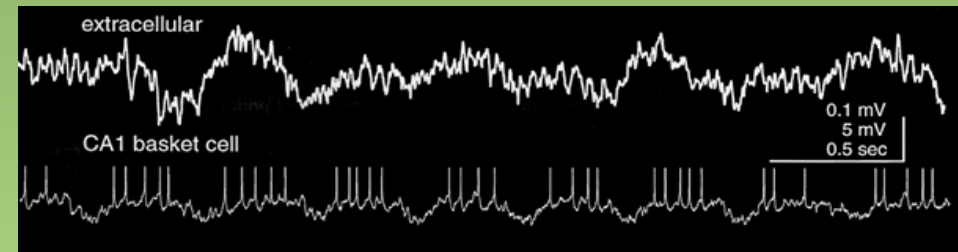


La metáfora

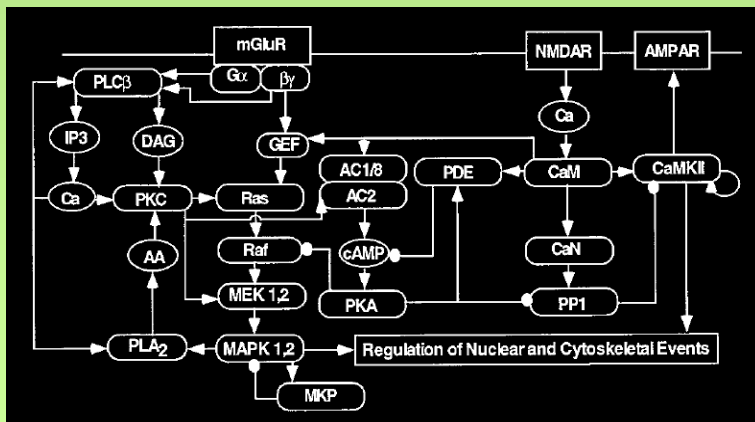
Anatomía



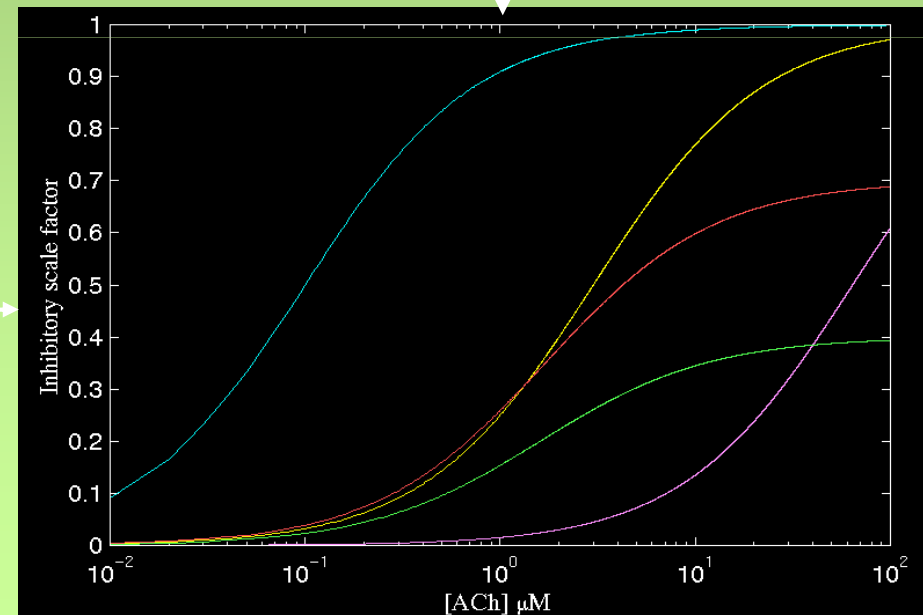
Fisiología



Bioquímica



Farmacología





COMPONENTES DE LA NEURONA



ESTRUCTURA ANATÓMICA

La metáfora

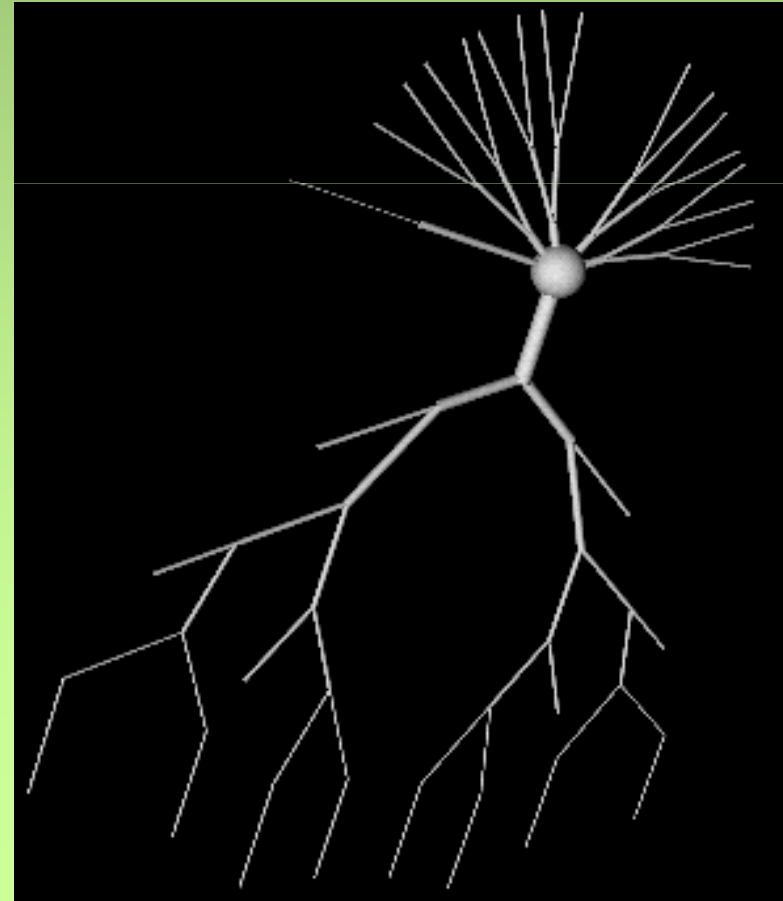
Morfología celular

celula piramidal biologica CA3
reconstruida por Turner *et al.*
(*J Comp Neurol*, 1993)



Representación geométrica

Modelo CA3 de célula piramidal de
Traub *et al.* (*J Physiol* 1994, 481:79-95)

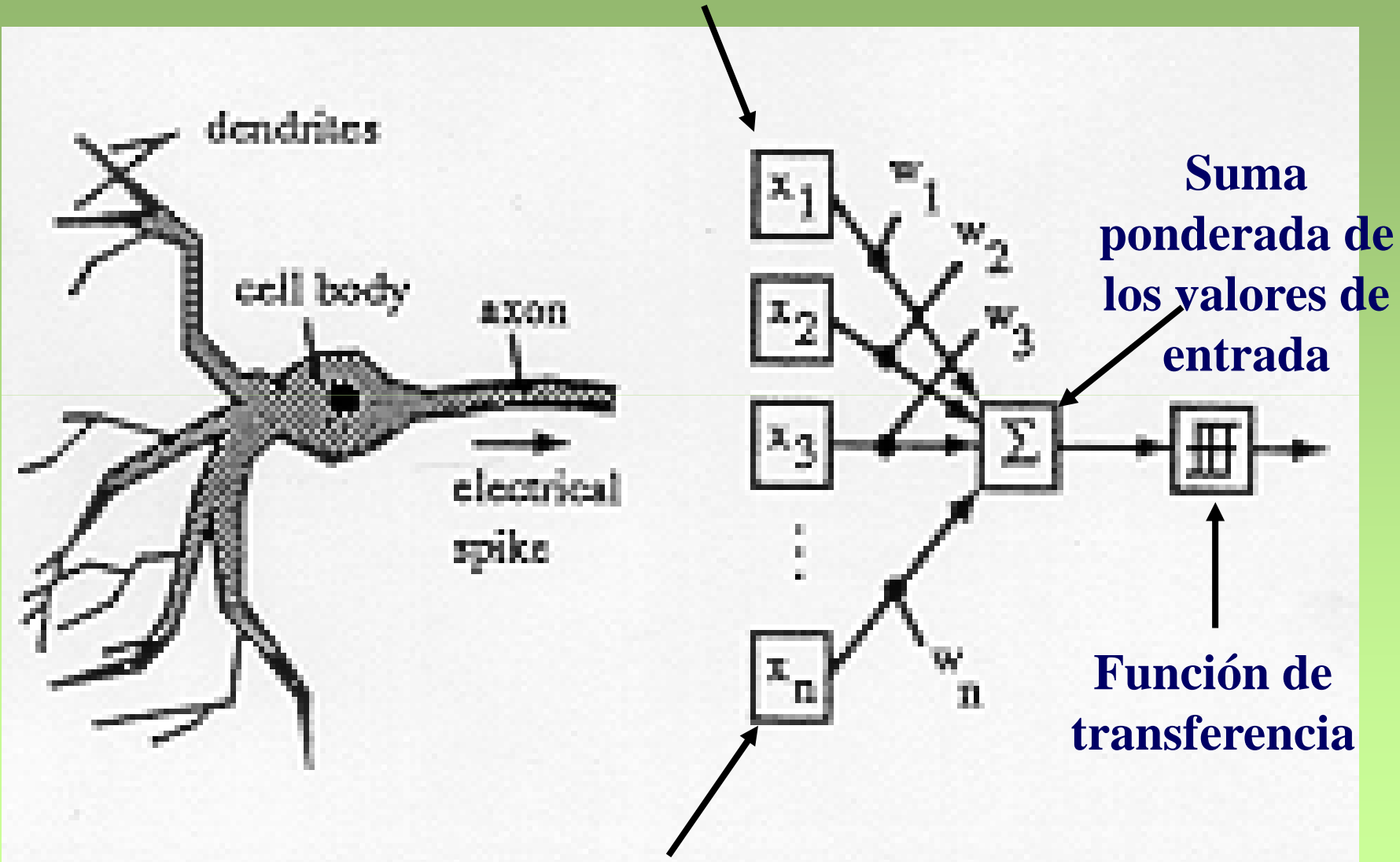




LA METÁFORA



Primer atributo de entrada



n-ésimo atributo de entrada



Que hace una red neuronal



- Las Redes Neuronales fijan hypersuperficies mediante funciones de base

□ Unidades Sigmoides

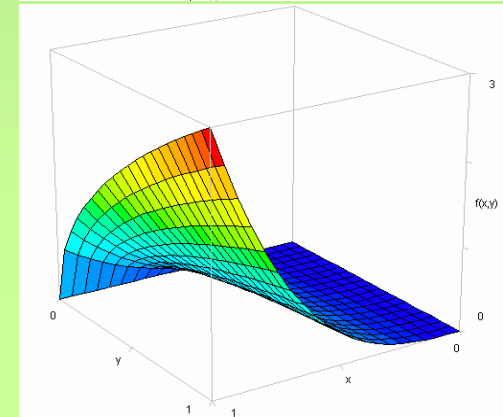
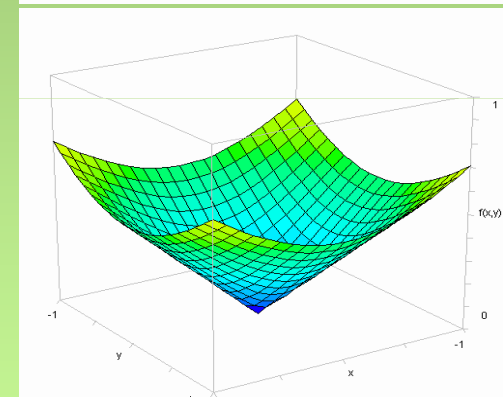
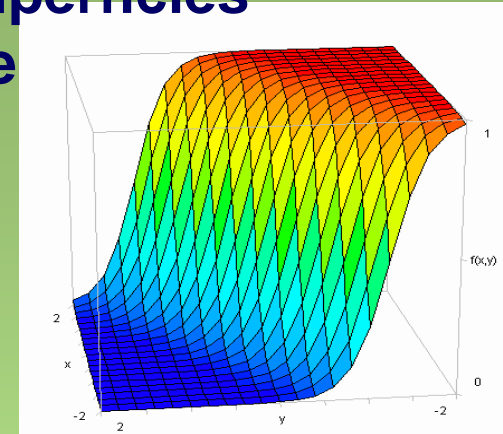
$$B_i(\mathbf{x}, \mathbf{W}) = \frac{1}{1 + e^{\left(\sum_{i=1}^k w_{ji} \cdot x_i + w_{j0} \right)}}$$

□ Unidades de Base Radial

$$B_j(\mathbf{x}; (\mathbf{c}_j \mid r_j)) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{r_j^2}\right)$$

□ Unidades Producto

$$B_i(\mathbf{x}, \mathbf{w}) = \prod_{i=1}^k x_i^{w_{ji}}$$

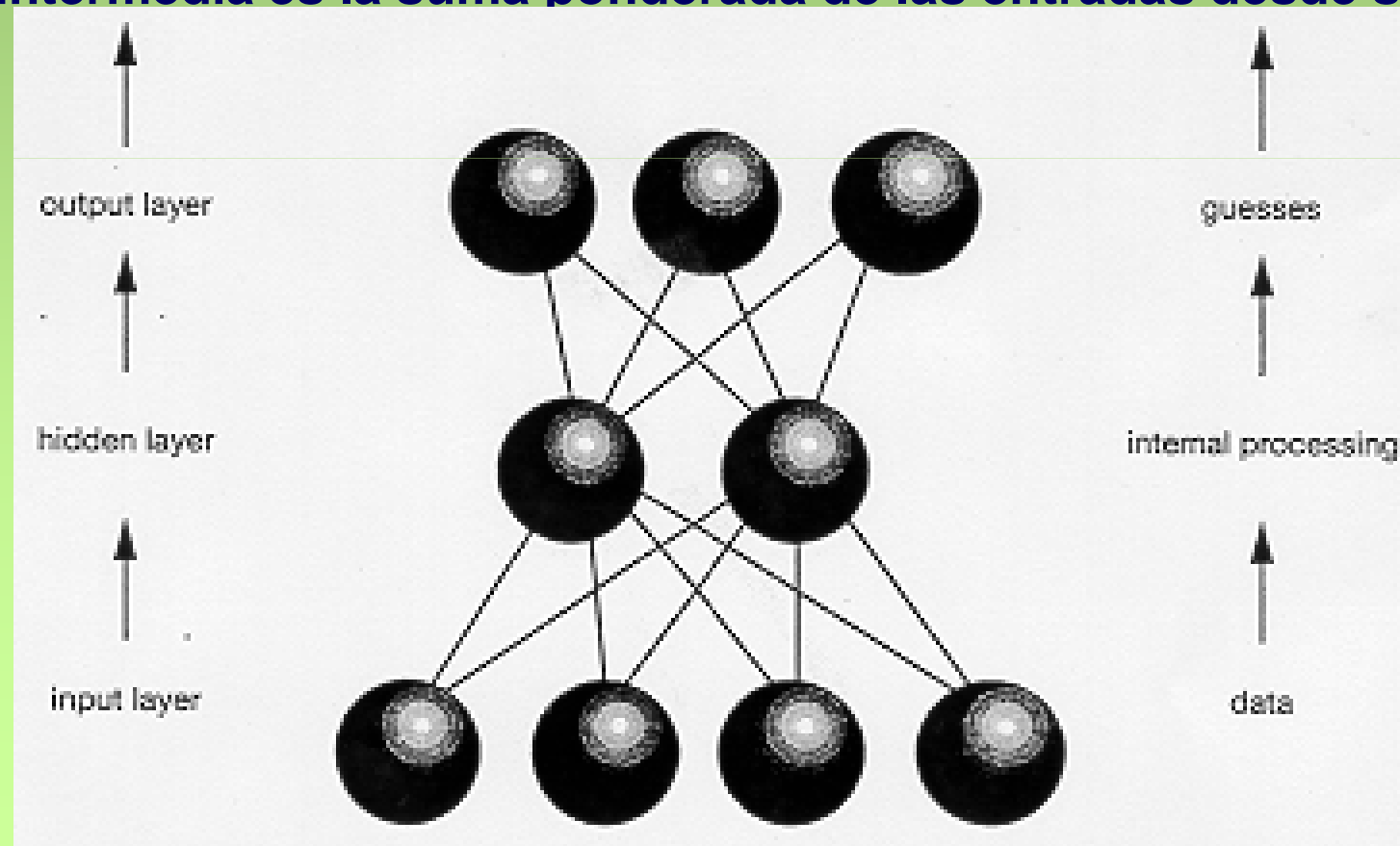




ESTRUCTURA DE UNA RED NEURONAL



- Las Redes Neuronales están compuestas de
 - **Nodos y Arcos**
- En cada arco se especifica un peso.
- Cada nodo contiene una **Función de Transferencia** la cual transforma la entrada en una salida. La entrada al nodo de la capa intermedia es la suma ponderada de las entradas desde sus arcos.





ESTRUCTURA DE UNA RED NEURONAL



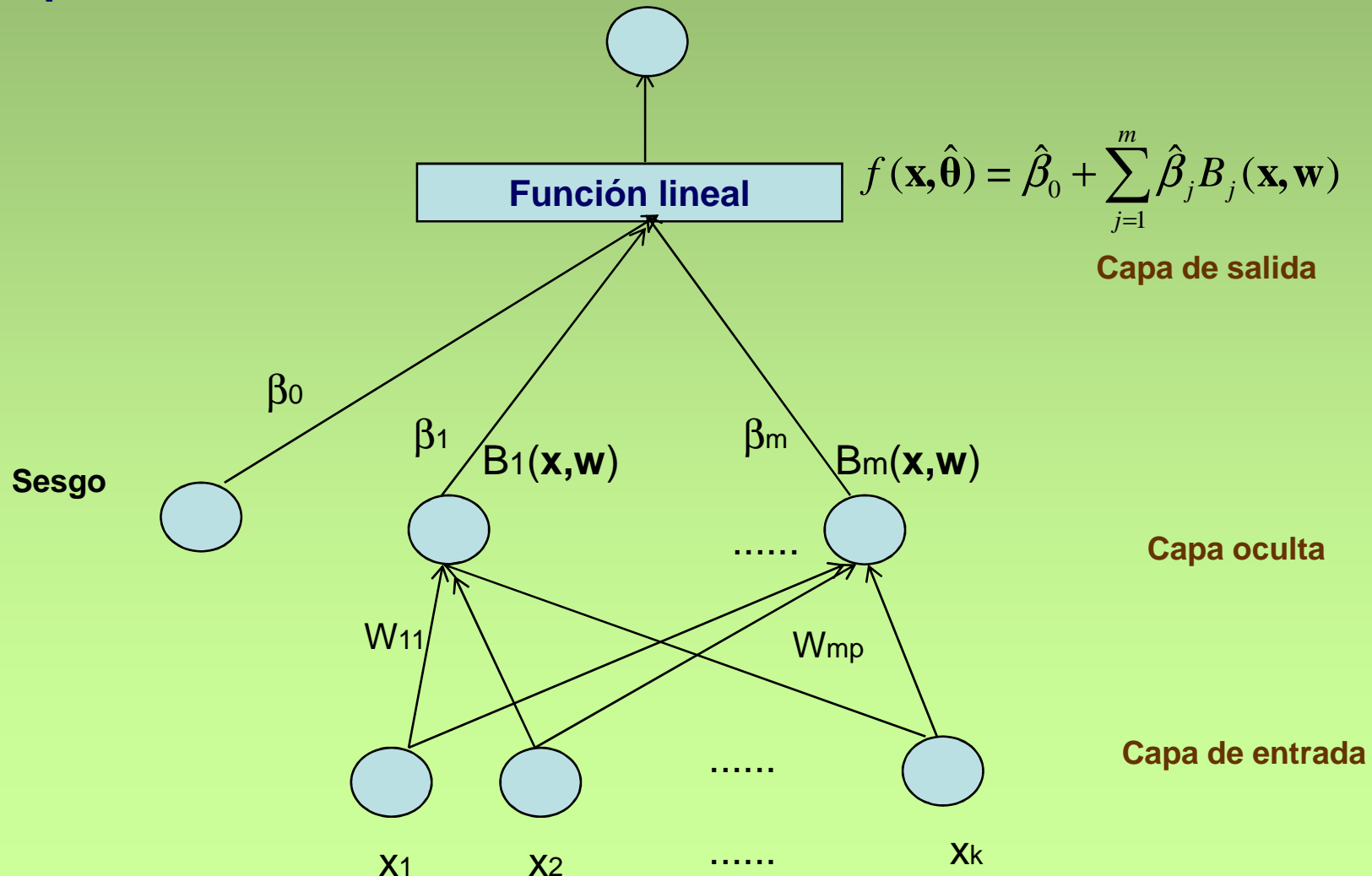
- Los nodos se agrupan en: una capa de entrada, una o más capas ocultas y una capa de salida.
- En la capa de entrada hay un nodo por cada atributo
- El número de nodos ocultos y de capas es configurable, es por ello un problema NP-completo.
- En la capa de salida hay un nodo por cada categoría o clase a predecir, a no ser que utilicemos salidas probabilísticas, esto es donde las salidas presentan la probabilidad de que un patrón pertenezca a esa clase, en este caso el número de salidas es el número de clases menos 1. Así, para un problema con dos clases con una salida es suficiente.



ESTRUCTURA DE UNA RED NEURONAL



- Las redes neuronales pueden también predecir salidas numéricas continuas (regresión) en cuyo caso el número de nodos coincide con el número de variables dependientes a predecir, una, en el caso más sencillo.





ESTRUCTURA DE UNA RED NEURONAL



Representación de un modelo de red para clasificación binaria

$$p_1(\mathbf{x}, \theta) = \frac{\exp(f(\mathbf{x}, \theta))}{1 + \exp(f(\mathbf{x}, \theta))}$$

$$p_2(\mathbf{x}, \theta) = \frac{1}{1 + \exp(f(\mathbf{x}, \theta))}$$

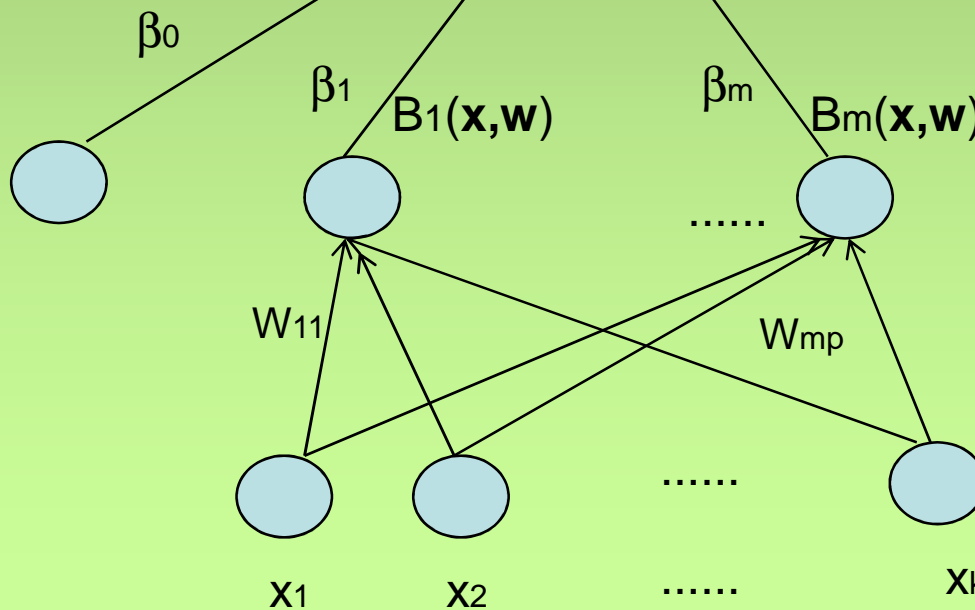
sobra la salida p_2
pues su valor es $1-p_1$

Función softmax

$$f(\mathbf{x}, \hat{\theta}) = \hat{\beta}_0 + \sum_{j=1}^m \hat{\beta}_j B_j(\mathbf{x}, \mathbf{w})$$

Capa de salida

Sesgo



Capa oculta

Capa de entrada

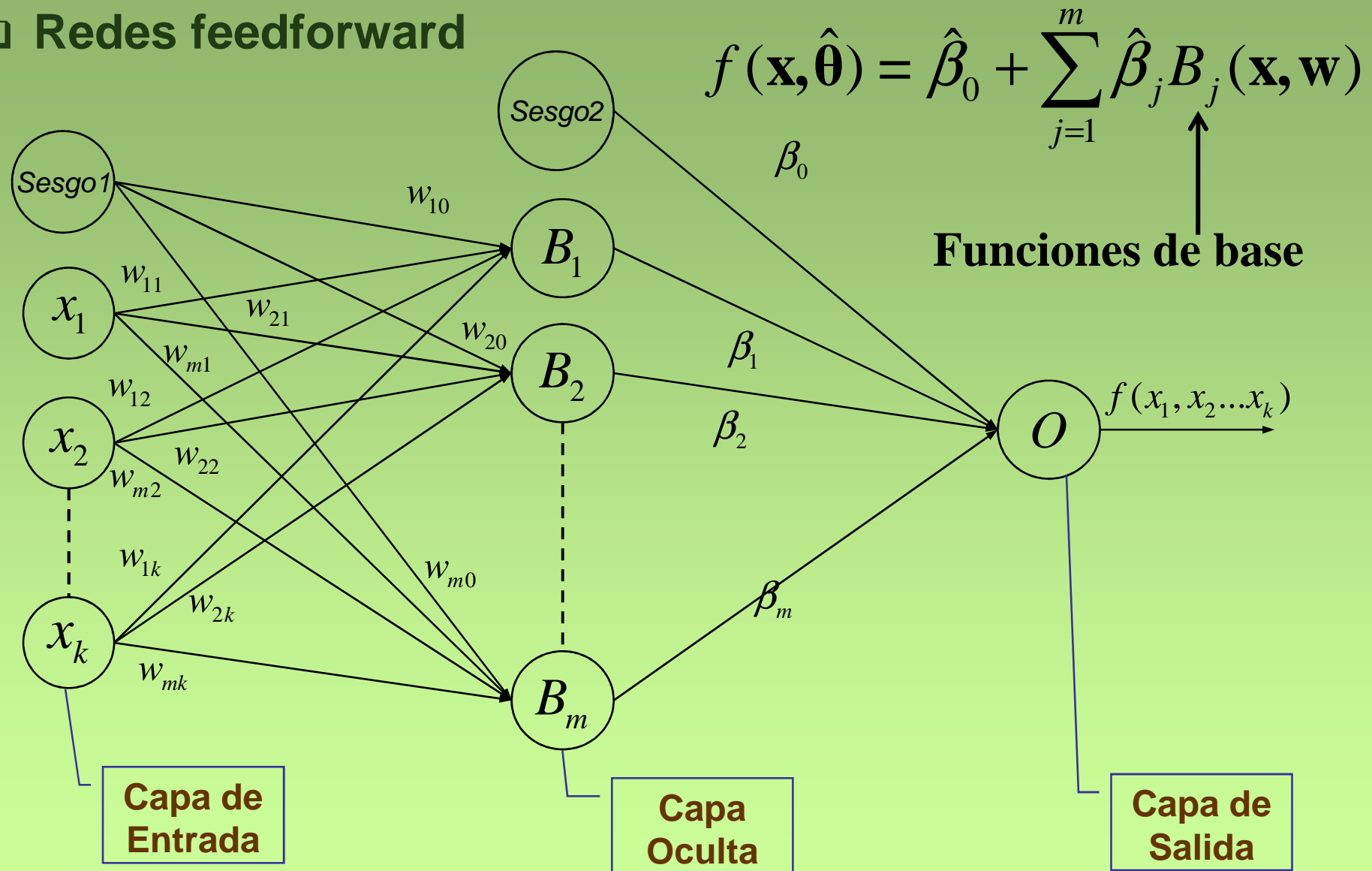


ESTRUCTURA DE UNA RED NEURONAL

ESTRUCTURA DE UN RED DE REGRESIÓN



Redes feedforward





MARGEN (PARA PREDICTORES CATEGÓRICOS, CLASIFICACIÓN)



- Para redes neuronales donde la salida (predicción) es categórica, hay un nodo de salida por cada categoría (clase) a ser predicha.
- La salida de cada nodo es por lo general la probabilidad de que un patrón de entrada pertenezca a dicha clase.
- El **margen** es la diferencia entre la probabilidad de pertenecer a la clase mas probable y la probabilidad de pertenecer a la segunda clase mas probable.



MARGEN (PARA PREDICTORES CATEGÓRICOS, CLASIFICACIÓN)



- Cuanto más grande sea el margen más grande será la confianza en la predicción.
- Por lo general si el margen es pequeño, se puede interpretar la red en el sentido de que “es incierto a que clase irá un nuevo patrón”.
- El margen se puede utilizar también para determinar la fuerza de las predicciones categóricas realizadas mediante modelos de regresión logística, árboles de decisión, máquinas de vectores soporte, modelos Naive Bayes, u otras metodologías de clasificación.



FUNCIONES DENTRO DE UN NODO: FUNCIONES DE TRANSFERENCIA



- Cada nodo contiene una función de transferencia, en el caso de la sigmoide, convierte la suma ponderada de las entradas como la salida de la función mediante una sigmoide

$$B_i(\mathbf{x}, \mathbf{W}) = \frac{1}{1 + e^{\left(\sum_{i=1}^k w_{ji} \cdot x_i + w_{j0} \right)}}$$



FUNCIONES DENTRO DE UN NODO: FUNCIONES DE TRANSFERENCIA

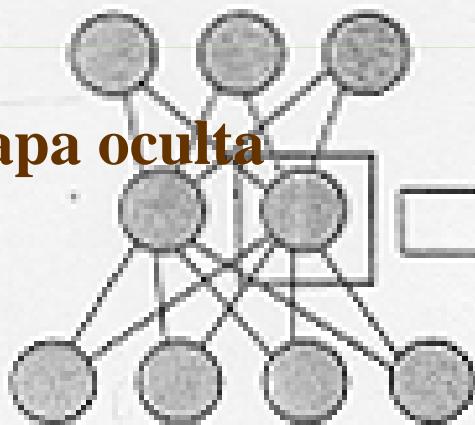


**Función de
transferencia
sigmoide**

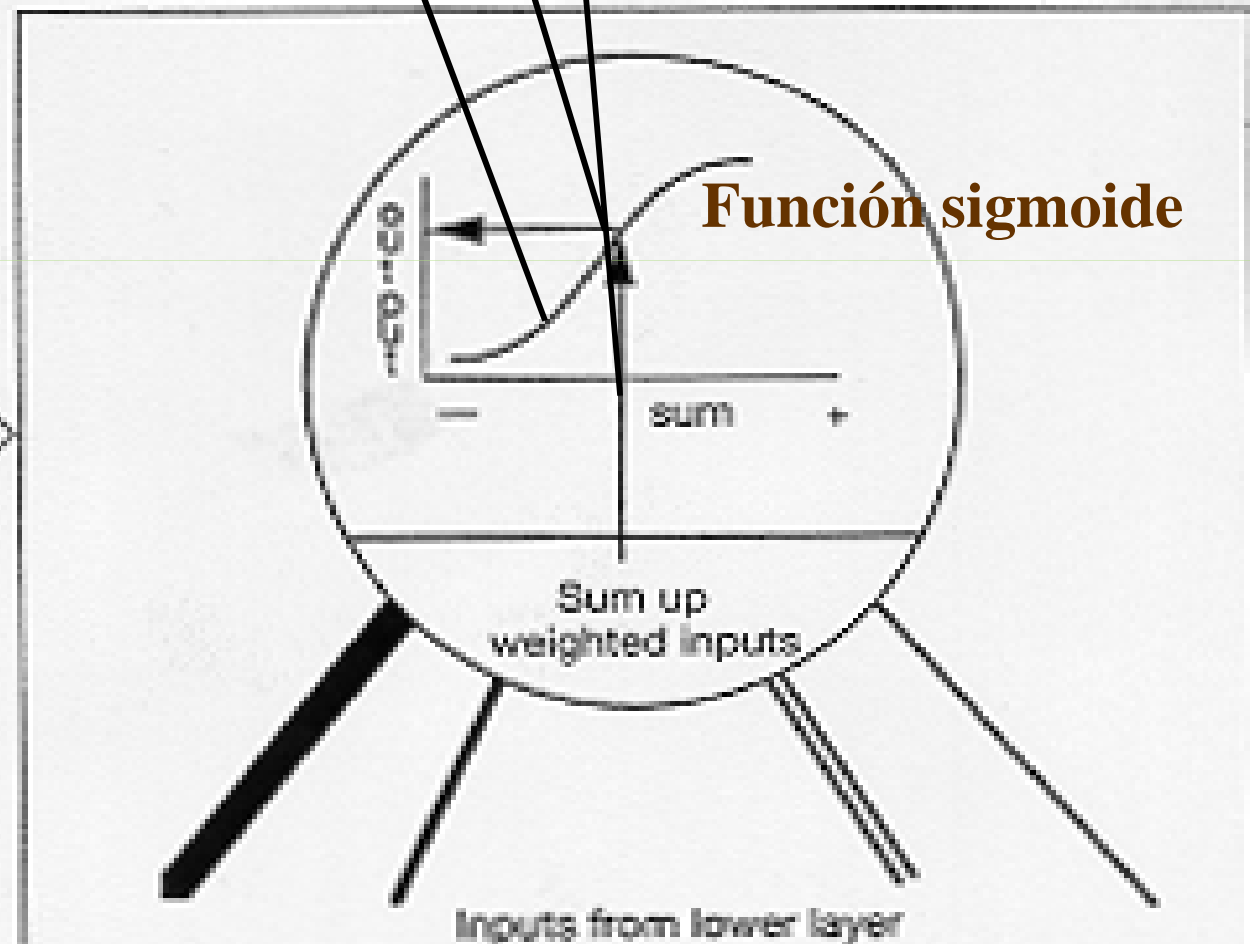
$$B_i(\mathbf{x}, \mathbf{W}) = \frac{1}{1 + e^{\left(\sum_{i=1}^k w_{ji} \cdot x_i + w_{j0} \right)}}$$

Capa de salida

Capa oculta



Capa de entrada

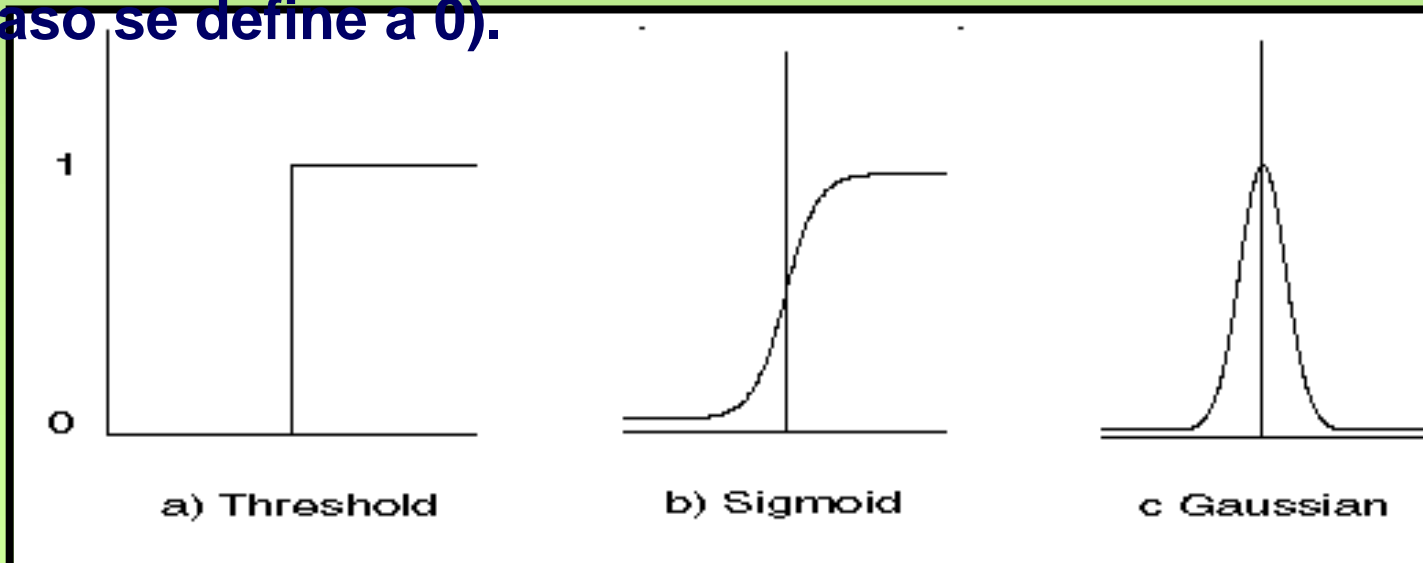




FUNCIONES DE TRANSFERENCIA



- Podemos utilizar una variedad de funciones de transferencia en los nodos de la capa oculta de la red.
 - Funciones de umbral (perceptron, figura a)
 - Funciones lineales y en rampa (para capa oculta en regresión, o en clasificación si luego se utiliza la función softmax)
 - Funciones sigmoides (e.g. Curvas logísticas, figura b)
 - Funciones de base radial (e.g. Gaussianas, figura c)
 - Estocásticas (el nodo de salida se define a 1 con probabilidad p dada por una función sigmoide, en otro caso se define a 0).





FUNCIONES DE TRANSFERENCIA



Lineal: $f_i(S) = cS$ siendo $S = \sum_{j=1}^k w_{ji} x_{ji}$

Umbral o salto: $f_i(S) = \begin{cases} 0 & \text{si } S \leq T_0 \\ 1 & \text{si } S > T_0 \end{cases}$

Rampa: $f_i(S) = \begin{cases} 0 & \text{si } S < T_1 \\ \frac{S - T_1}{T_2 - T_1} & \text{si } T_1 \leq S \leq T_2 \\ 1 & \text{si } S > T_2 \end{cases}$

Sigmoide: $f_i(S) = \frac{1}{1 + e^{-cS}}$

Tangente Hiperbólica: $f_i(S) = \frac{1 - e^{-cS}}{1 + e^{-cS}}$

Gaussiana: $f_i(S) = e^{-\frac{S^2}{v}}$



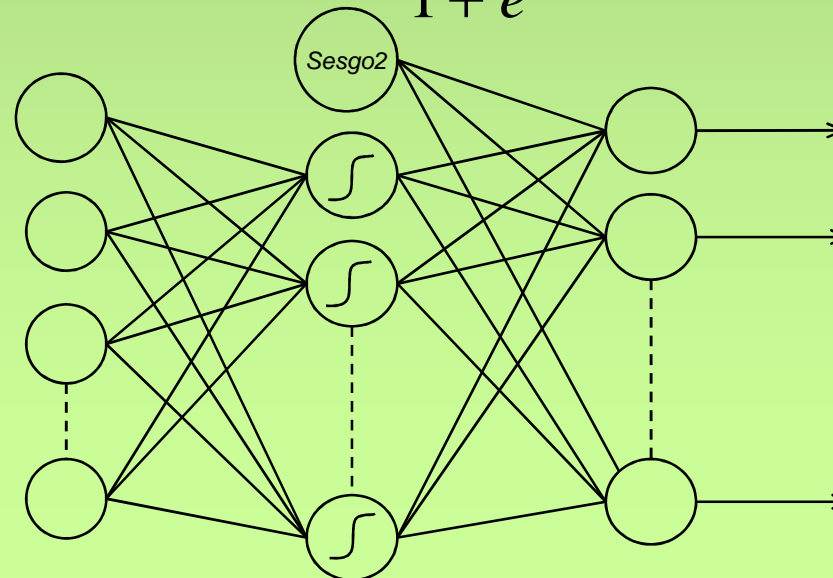
FUNCIONES DE BASE



- Distintas funciones de salida en capa oculta →
Distintos modelos Redes Neuronales

Unidad Sigmoidal (US)

$$H_j(x_1, x_2, \dots, x_k) = \frac{1}{1 + e^{-\left(\sum_{i=1}^k w_{ji} \cdot x_i + w_{j0}\right)}}$$





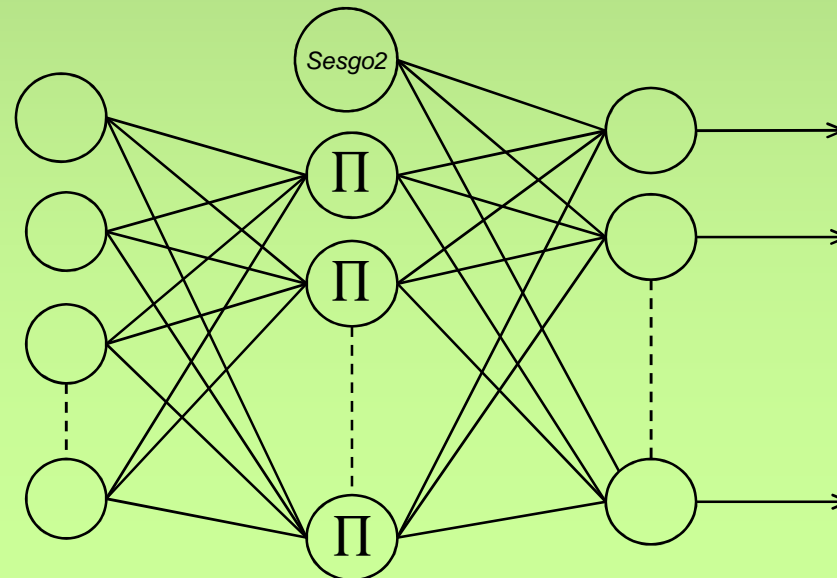
FUNCIONES DE BASE



- ❑ Distintas funciones de salida en capa oculta →
Distintos modelos de Redes Neuronales

Unidades Producto (UP)

$$H_j(x_1, x_2, \dots, x_k) = \prod_{i=1}^k x_i^{w_{ji}}$$





Función de base Unidad Producto



$$\mathbb{F} = \left\{ f : A \subset R^k \rightarrow R : f(x_1, x_2, \dots, x_k) \right\}$$

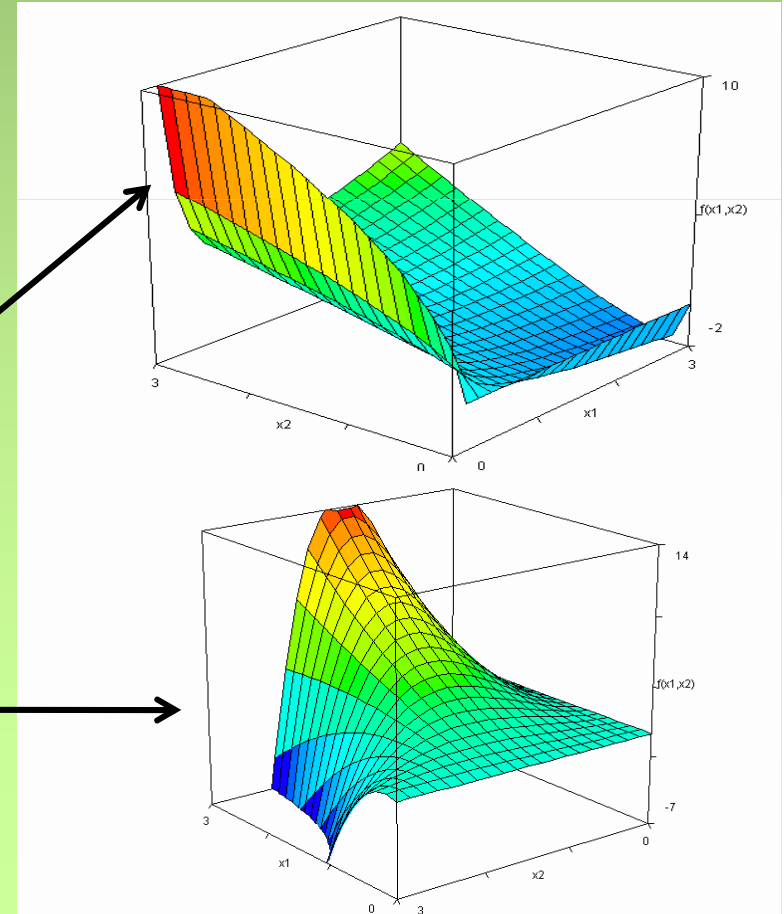
$$f(x_1, x_2, \dots, x_k) = \left(\sum_{j=1}^m \beta_j \left(\prod_{i=1}^k x_i^{w_{ji}} \right) \right) + t$$

$$A = \left\{ (x_1, x_2, \dots, x_k) \in R^k : 0 < x_j \right\}$$

Ejemplos:

$$f_1(x_1, x_2) = x_1^{-1} x_2^{0,5} - 2x_1^{0,6} x_2^{0,3} + x_1 x_2$$

$$f_2(x_1, x_2) = x_1 x_2 - x_1^2 x_2^3 + 2x_1^2 x_2^2$$





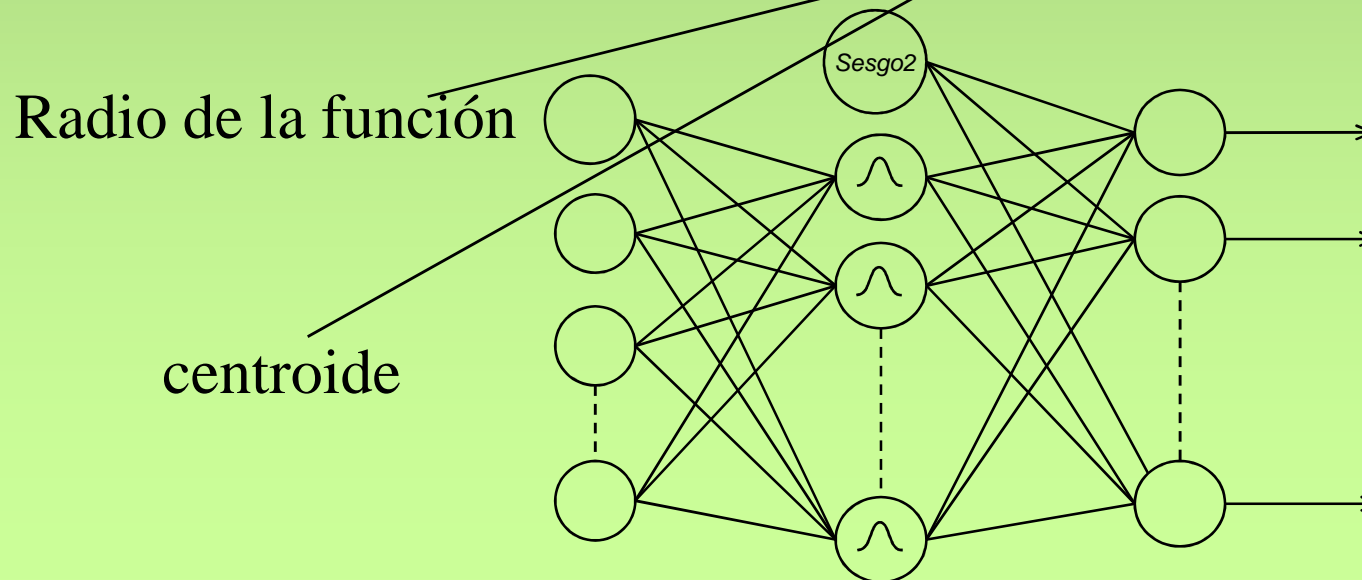
FUNCIONES DE BASE



- Distintas funciones de salida en capa oculta → Distintos modelos de Redes Neuronales

Funciones de Base Radial (RBF) Gaussiana

$$H_j(x_1, x_2 \dots x_k) = e^{-\frac{1}{2} \left(\frac{\| (x_1, x_2 \dots x_k) - (c_{j1}, c_{j2} \dots, c_{jk}) \|}{r_j} \right)^2}$$





FUNCIONES DE BASE RADIAL



Funciones de Base Radial (RBF) Gaussiana

$$\Phi(\|\mathbf{x}_i - \mathbf{c}_j\|) = e^{-\frac{1}{2} \left(\frac{\|(\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_k) - (\mathbf{c}_{j1}, \mathbf{c}_{j2} \dots, \mathbf{c}_{jk})\|}{r_j} \right)^2}, \quad i=1, \dots, n, \quad j=1, \dots, m$$

n , n° de patrones, m , n° de nodos en capa oculta, o n° de funciones de base radial

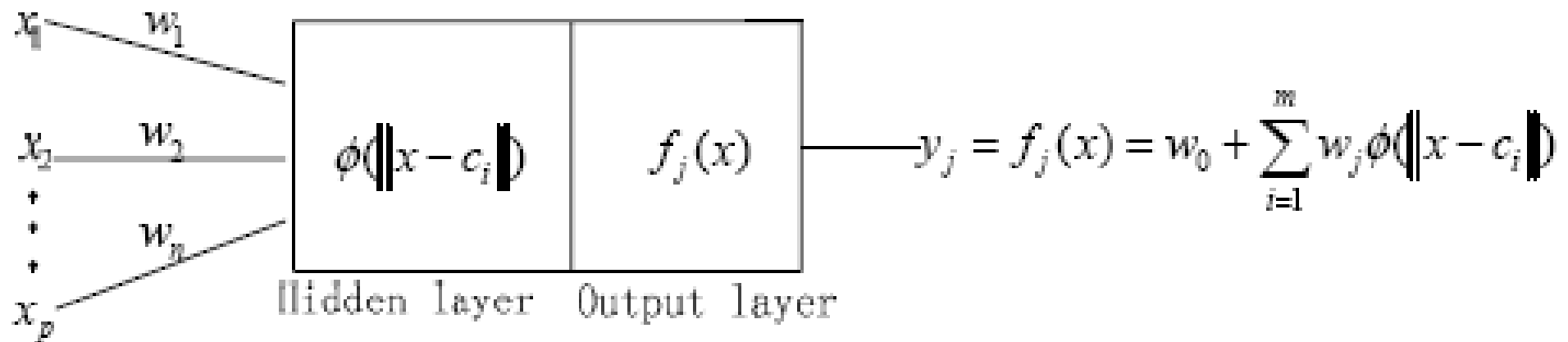
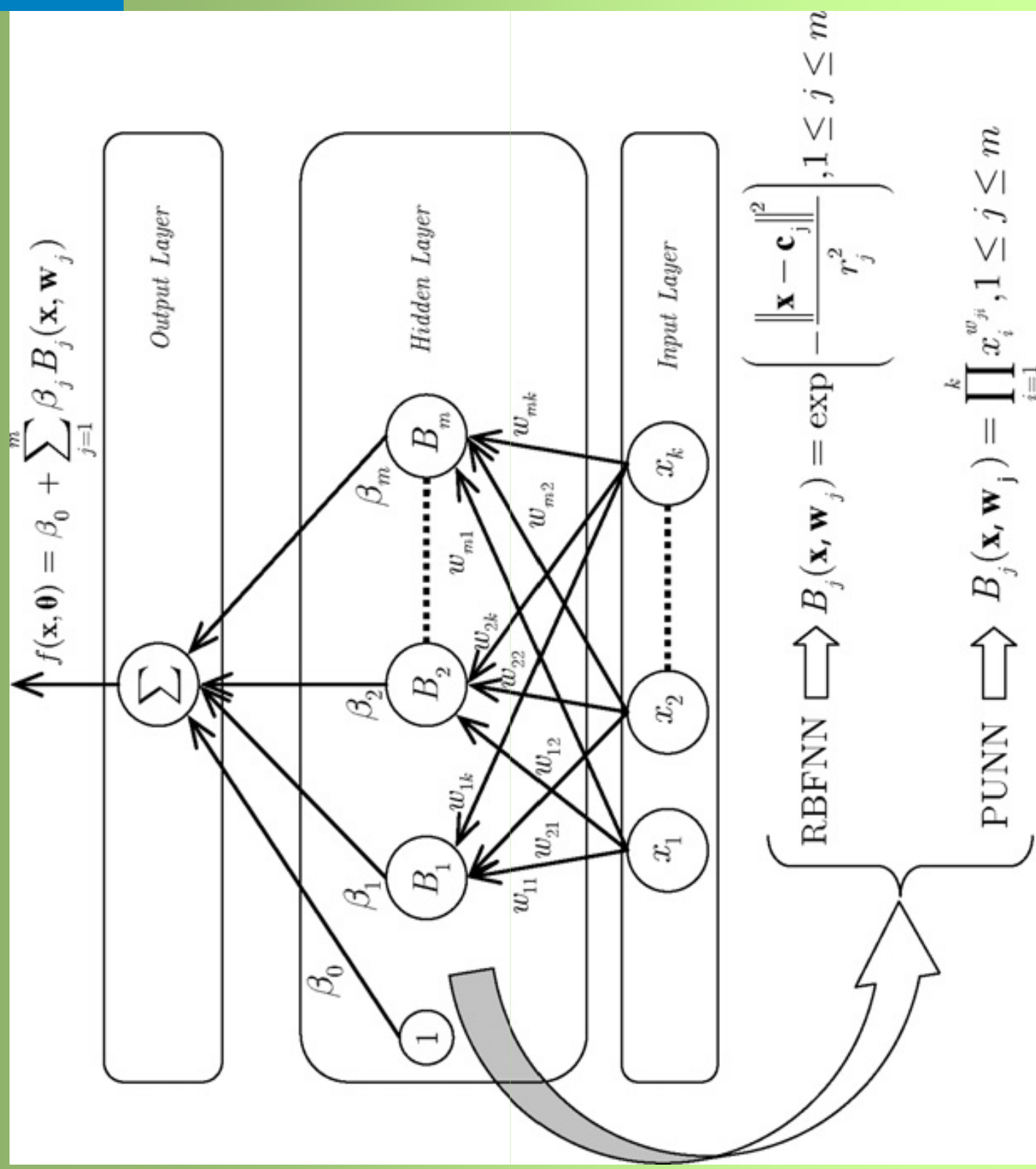


Fig. 1. Estructura de una red RBF





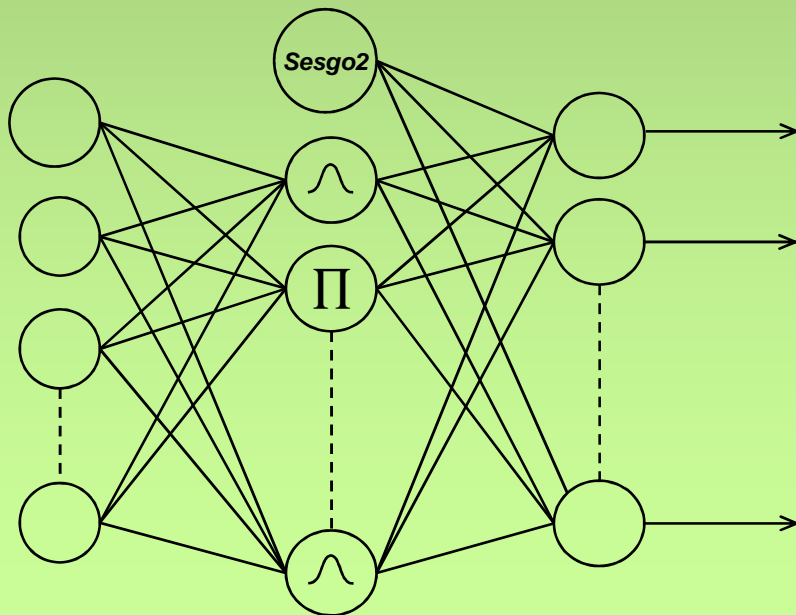
FUNCIONES DE BASE



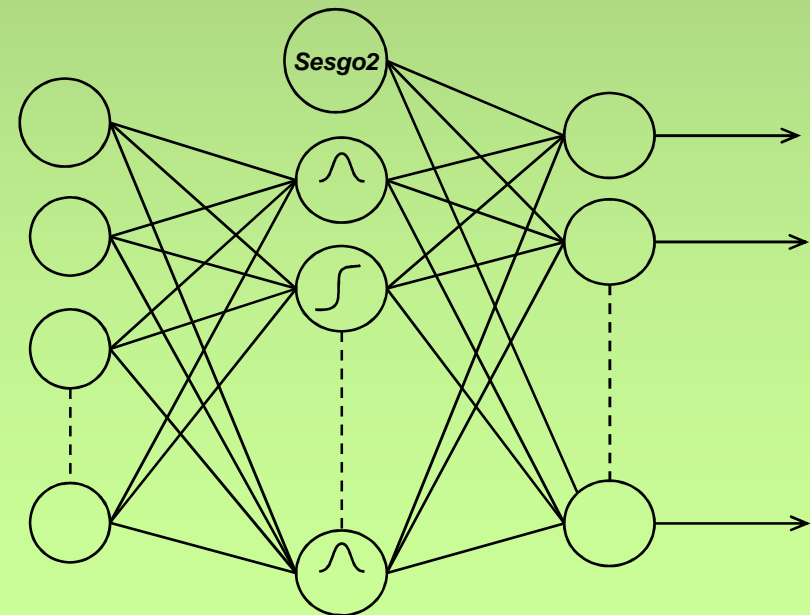
- Distintas funciones de salida en capa oculta →
Distintos modelos de Redes Neuronales

Modelos de híbridos de funciones de base

Modelos Híbridos RBF+UP



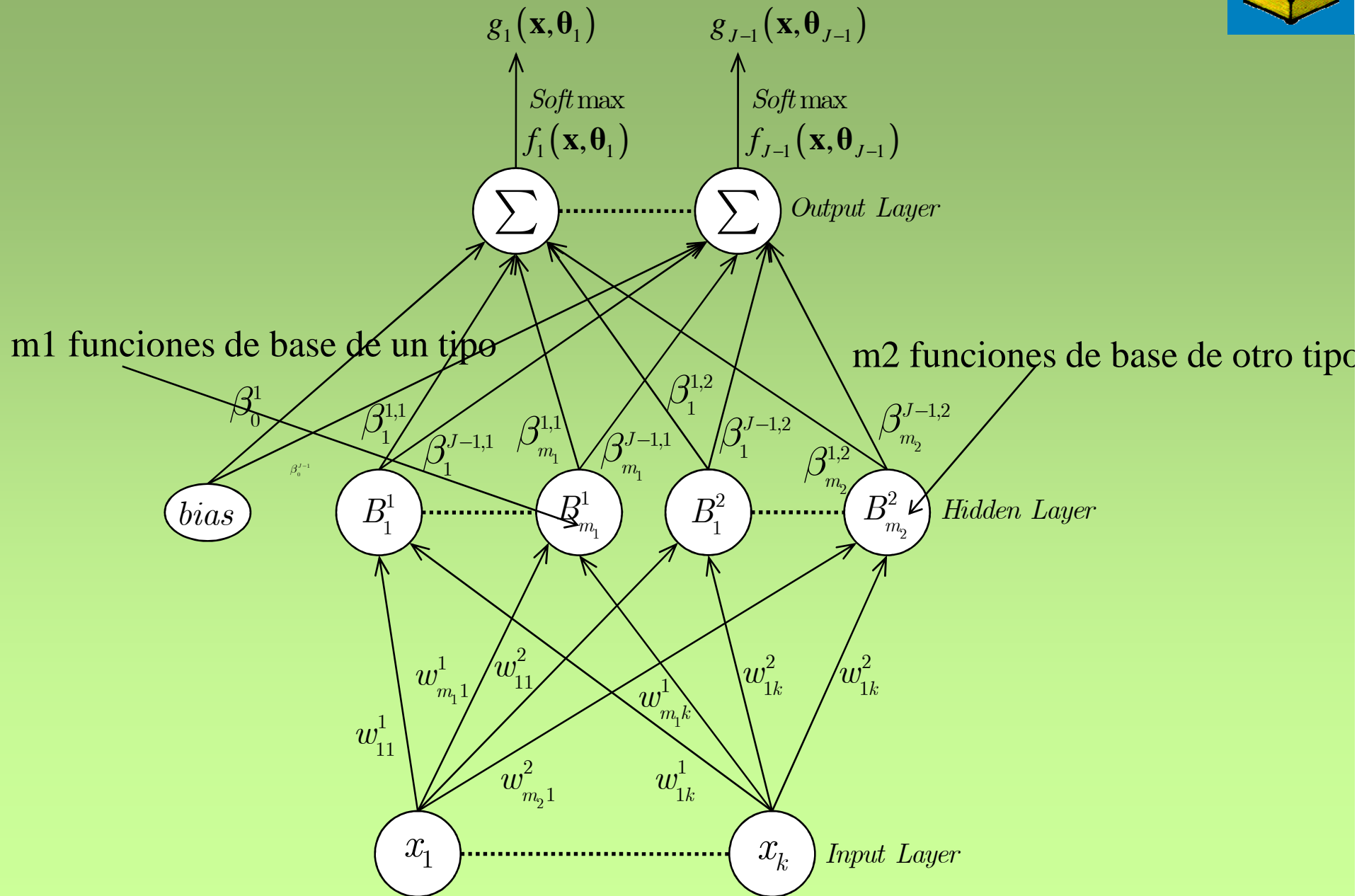
Modelos Híbridos RBF+US





ESTRUCTURA DE UNA RED NEURONAL HIBRIDA

ESTRUCTURA DE UN RED DE CLASIFICACIÓN MULTICLASE





UTILIZANDO LA RED



- Se introduce un ejemplo (patrón) en la red: el valor de cada atributo del ejemplo se introduce en un nodo de entrada de la red.
- Los múltiples valores de entrada se transmiten a través de cada arco (x) mediante el peso (w) definido sobre cada arco.
- La suma de valores ponderados se transmiten en cada nodo.
- Se obtiene la salida (h) a partir de cada nodo aplicando la función de transferencia (f) a la suma ponderada de las entradas al nodo de la capa oculta. Continuandose hacia adelante a la salida a través de la red.
- Este proceso se conoce como activación hacia adelante “*feed-forward*”.

$$y_i = f_i\left(\sum_{j=1}^k w_{ji} x_{ji}\right) = f_i\left([w_{1i} \dots w_{ki}] \begin{bmatrix} x_{1i} \\ \dots \\ x_{ki} \end{bmatrix}\right)$$

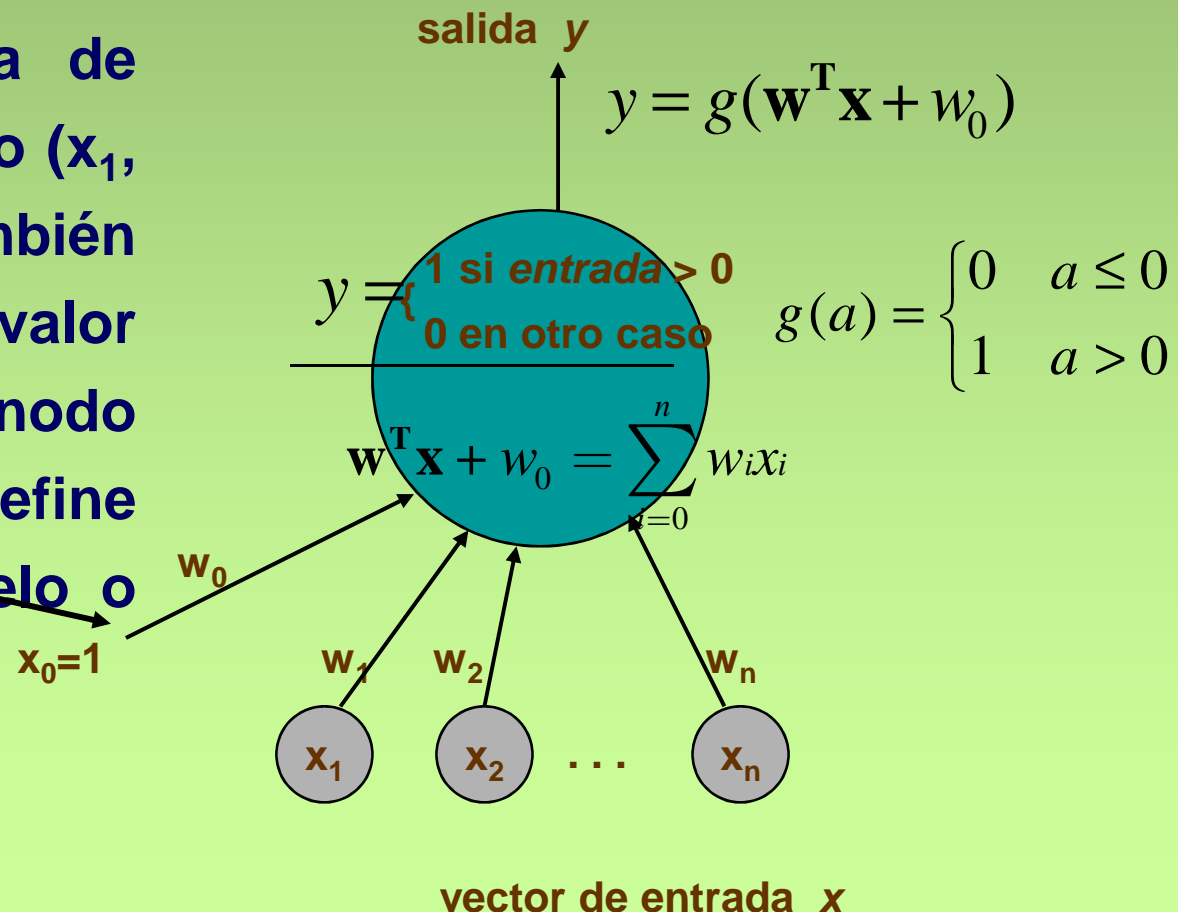


EL PERCEPTRON: BASADO EN UN SENCILLO UMBRAL



Un perceptron es una red neuronal sin capas ocultas, donde se define una función de transferencia basada en un umbral.

Observamos que además de tener un nodo en capa de entrada para cada atributo (x_1, x_2, \dots, x_n), podemos también definir una neurona de valor constante asociada a un nodo de entrada (x_0), que se define como el sesgo del modelo o 'bias'.



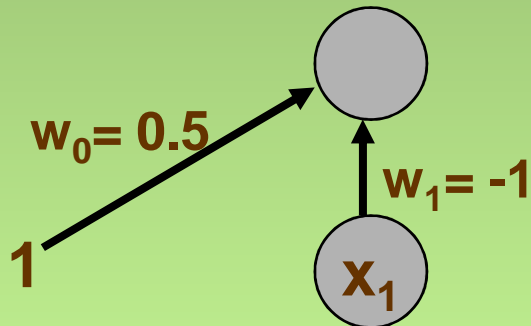


UN SIMPLE PERCEPTRON BASADO EN UMBRAL



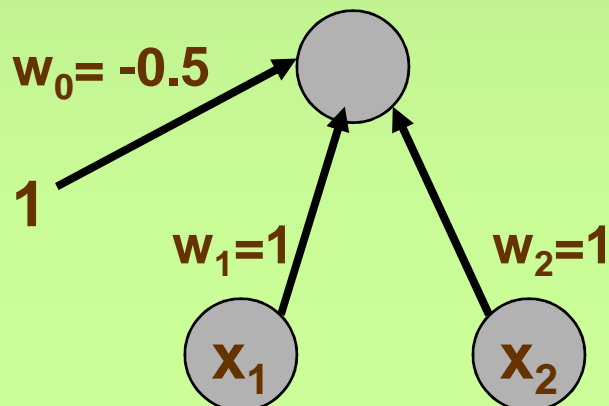
Si consideramos el perceptron definido en la transparencia anterior podemos producir pesos para modelar el bit de inversión y las funciones Booleanas OR y AND.

- Bit de Inversion

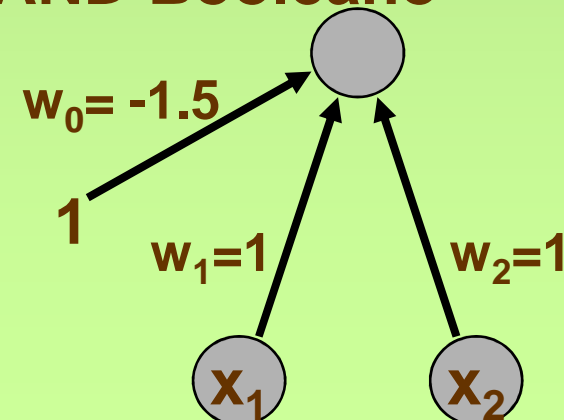


entrada x_1	entrada x_2	inversion x_1	OR	AND
0	0	1	0	0
0	1	1	1	0
1	0	0	1	0
1	1	0	1	1

- OR Booleano



- AND Booleano

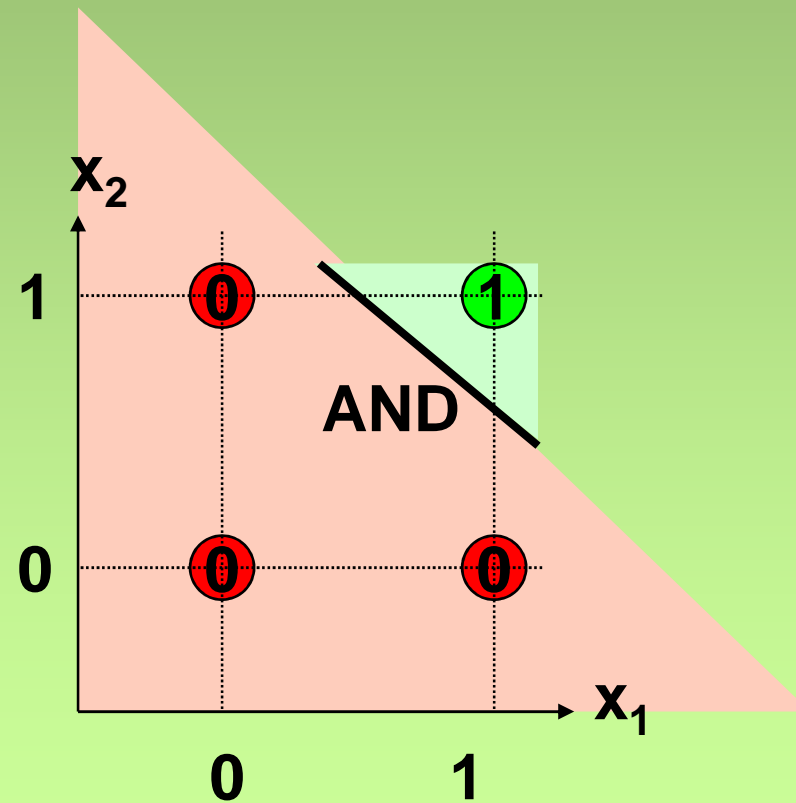
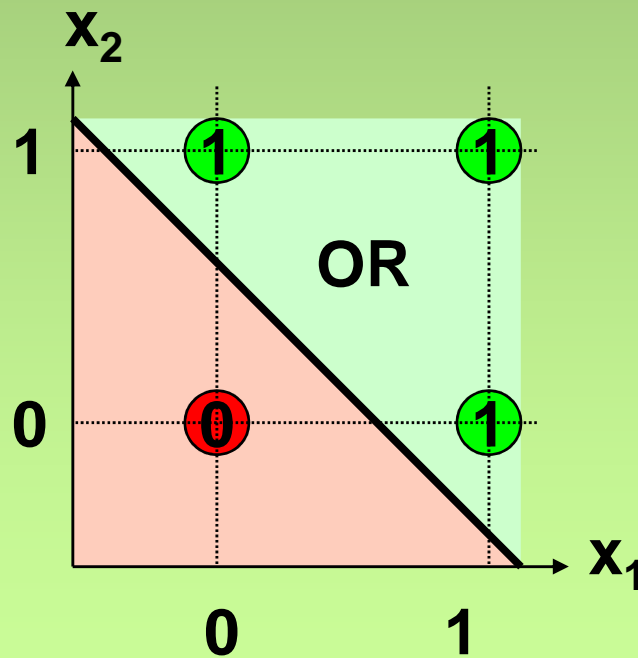




PERCEPTRON BASADO EN UMBRAL



Sin embargo si queremos discriminar la función Booleana XOR, esto no se puede obtener mediante un simple perceptron lineal, puesto que las funciones OR y AND son linealmente separables, pero la función lógica XOR no lo es.



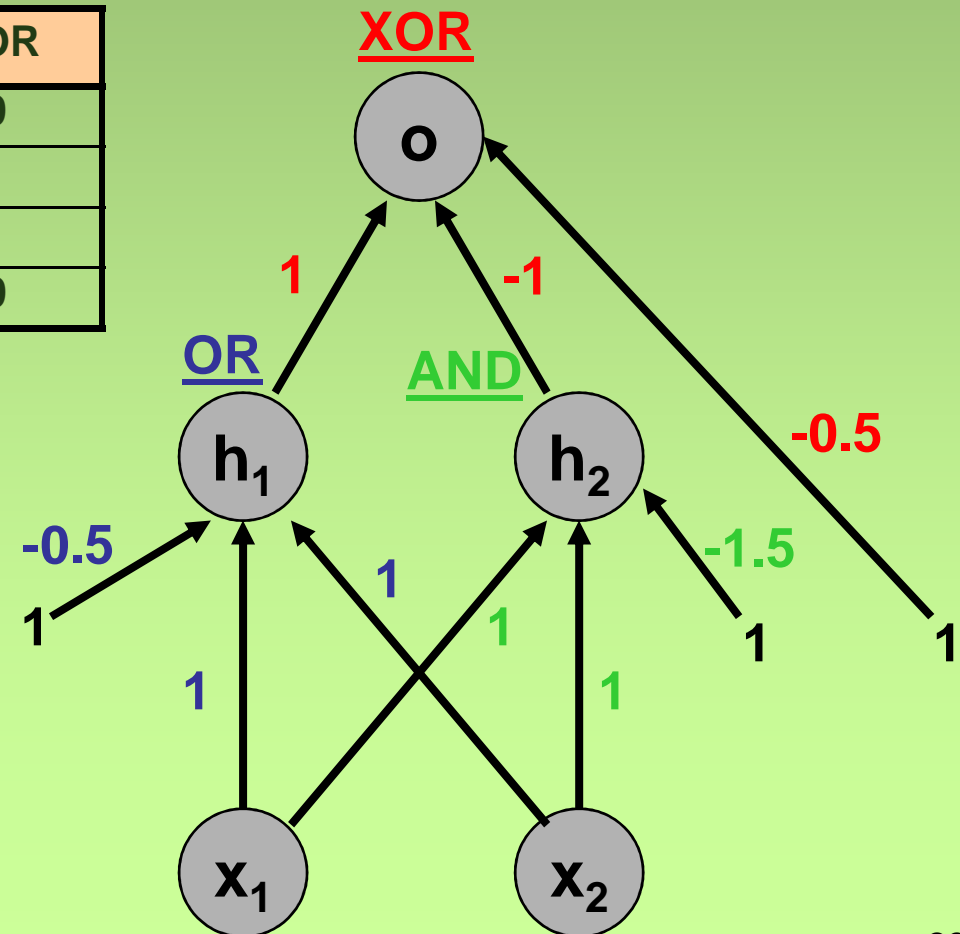
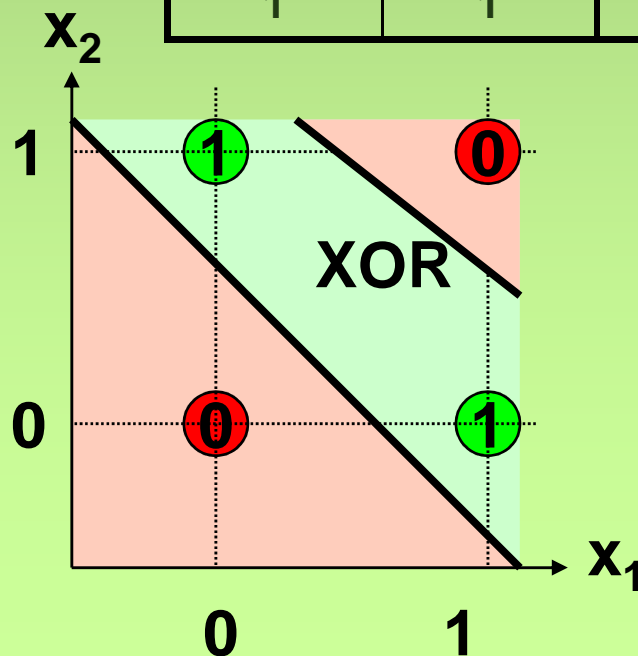


PERCEPTRON BASADO EN UMBRAL



XOR se modela mediante una red neuronal más compleja. Cada nodo de la capa oculta y cada nodo de la capa de salida, utilizarán la función de umbral definida anteriormente.

Entrada x_1	Entrada x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

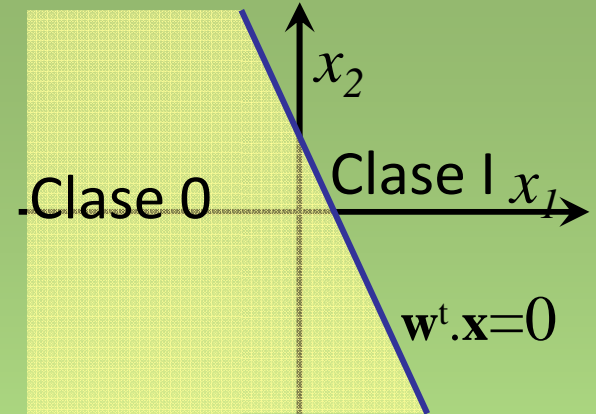




PERCEPTRON BASADO EN UMBRAL: RESUMEN

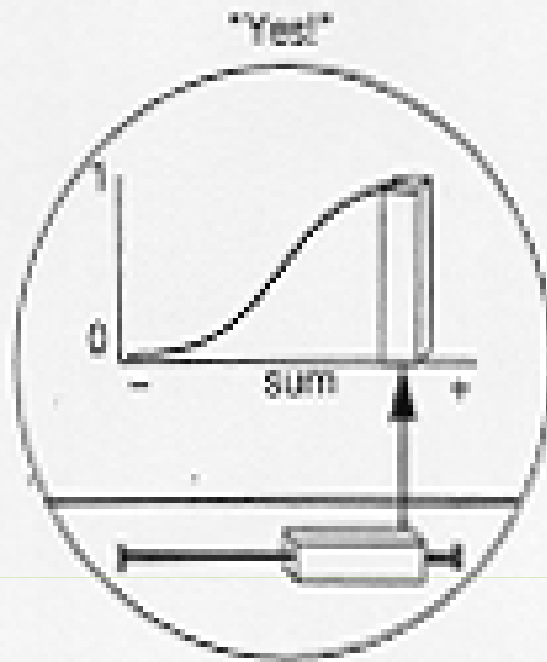


- Separa el espacio con un hiperplano
- $y = f (w_1 x_1 + w_2 x_2 + \dots + w_n x_n),$
- $f(s) = \{ 1 \text{ si } s \geq 0, 0 \text{ si } s < 0 \}$
- Puede incluir sesgo o “bias” w_0 .
- Importante históricamente
 - Estudiado muy detalladamente (Minsky y Papert ‘69)
- Es un clasificador lineal en 2 clases.
 - Resuelve problemas con patrones linealmente separables
 - No resuelve el problema XOR.
- Análogo a un clasificador de Bayes Gaussiano.
 - minimiza la probabilidad de error
 - clasificador denominado paramétrico

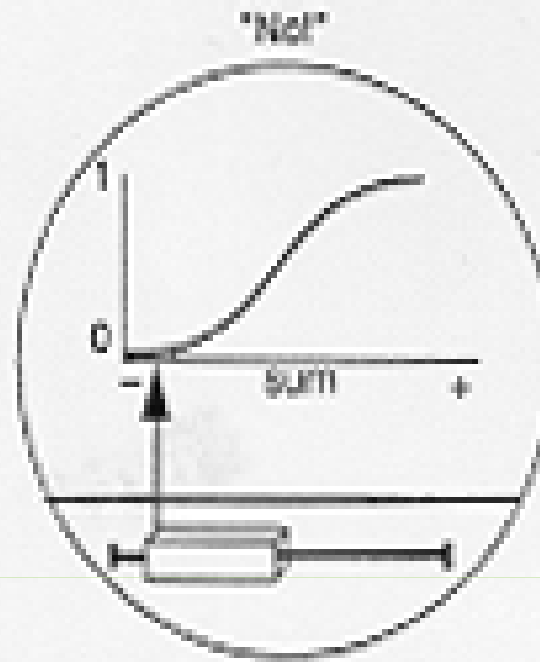




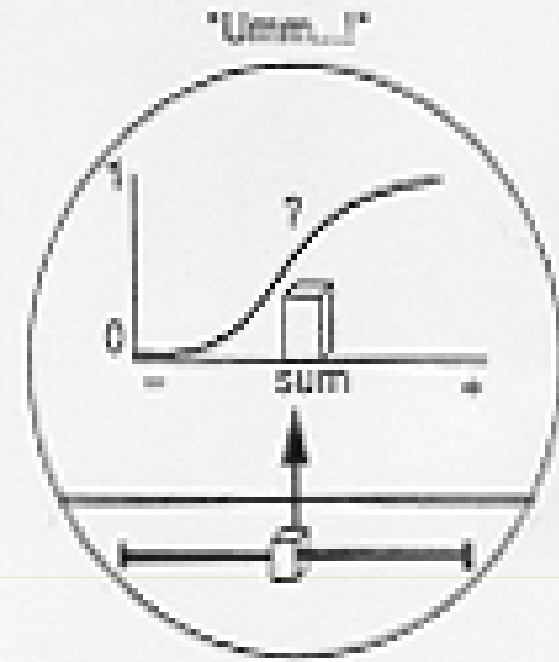
INHIBIDORES Y EXCITADORES



Large positive sums (inputs)
result in high output from nodes

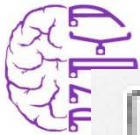


Large negative sums
result in low output from nodes

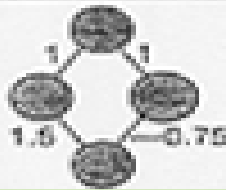
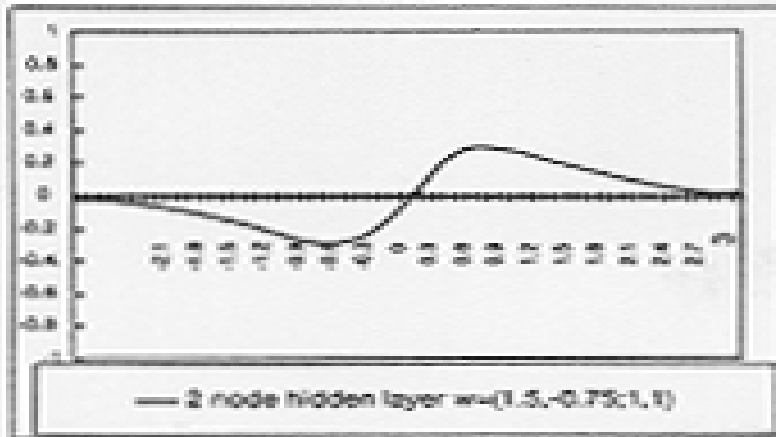


Sums near zero
result in mediocre output from nodes

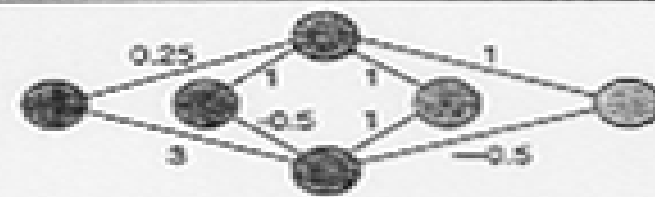
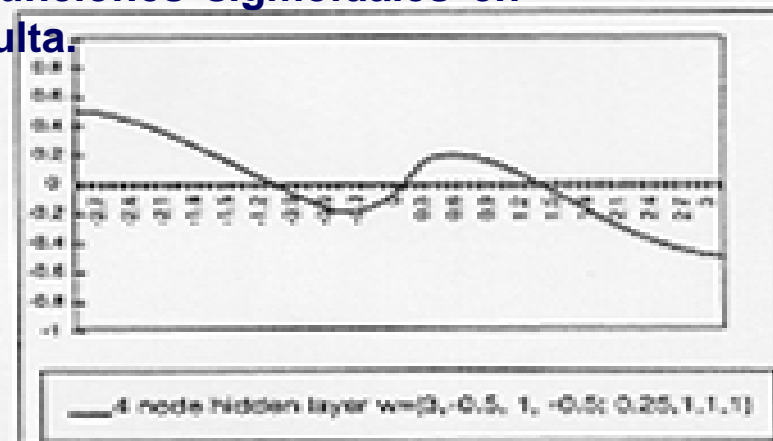
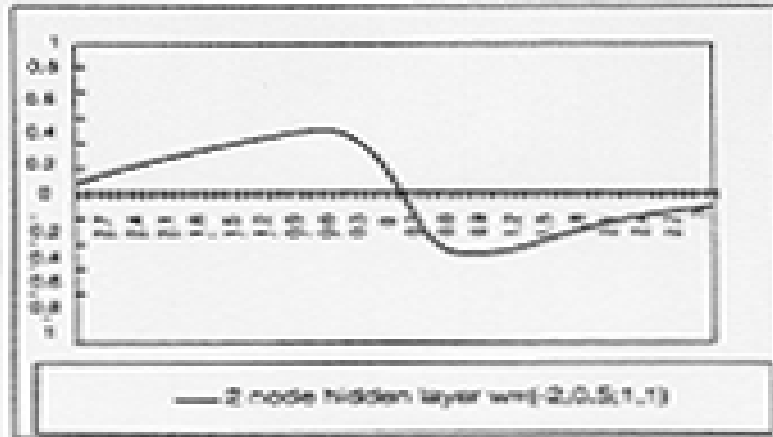
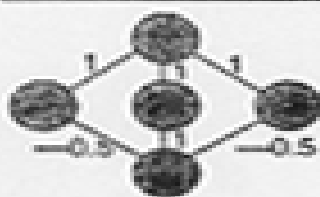
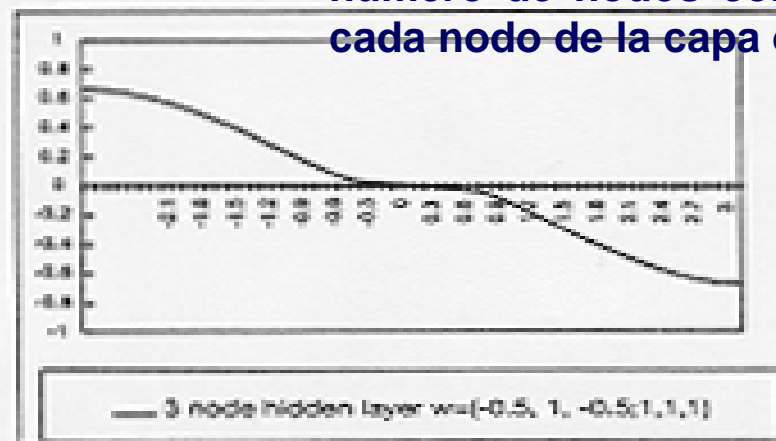
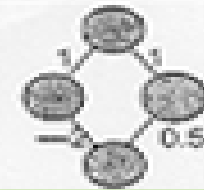
<p>Valores de la suma grandes y positivos de entrada a la neurona (o nodo) de la capa oculta producen valores de salida altos de esa neurona</p>	<p>Valores de la suma grandes y negativos de entrada a la neurona (o nodo) de la capa oculta producen valores de salida bajos de esa neurona</p>	<p>Valores de la suma cercanos a cero de entrada a la neurona (o nodo) de la capa oculta producen valores de salida mediocres (cercanos a 0.5) de esa neurona</p>
--	--	---



Ejemplo de una red sencilla con conexiones hacia adelante



- Las redes neuronales pueden aproximar funciones complejas.
- Este ejemplo muestra diferentes funciones que se pueden obtener mediante diferentes pesos y numero de nodos con funciones sigmoidales en cada nodo de la capa oculta.





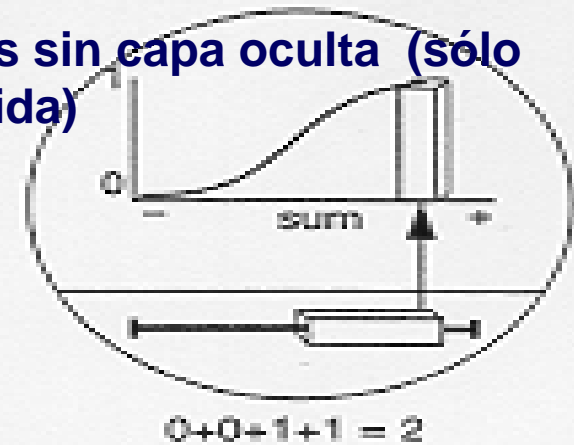
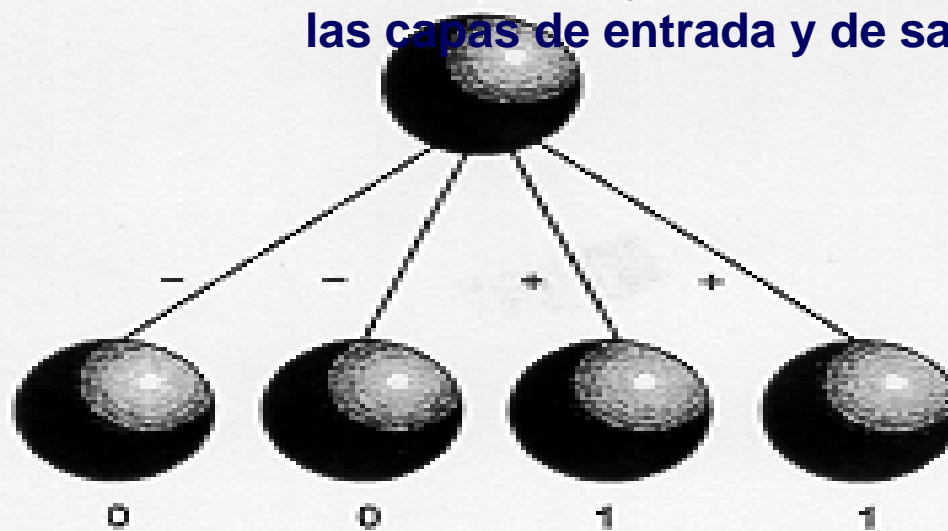
Ejemplo de Redes sencillas Feed-Forward



- Si suponemos que estamos trabajando con redes sin capa oculta (sólo las capas de entrada y de salida)

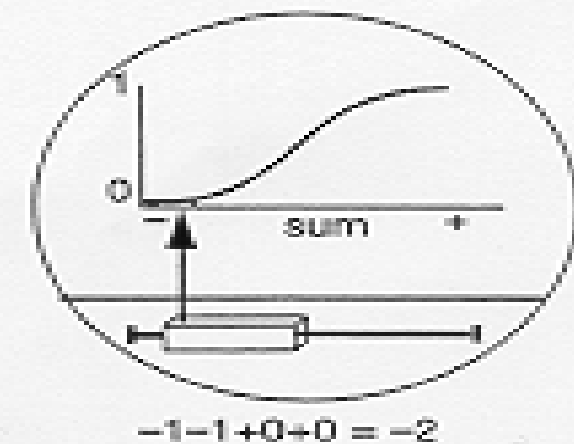
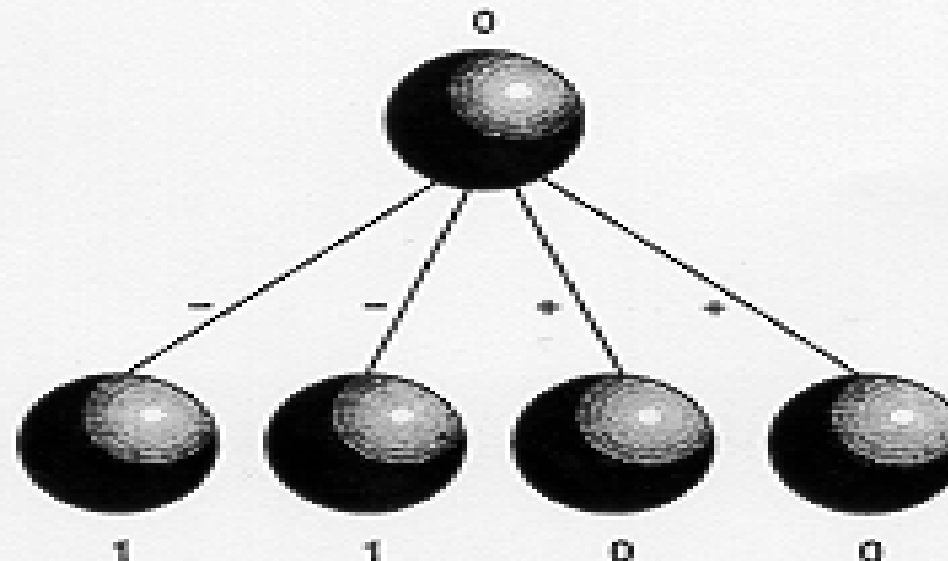
output layer

input layer



output layer

input layer





Ejemplo de Redes sencillas Feed-Forward



- Si suponemos que estamos trabajando con redes sin capa oculta (sólo las capas de entrada y de salida)
- Suponemos dos patrones de entrenamiento:
 - '0011' está en la clase 1
 - '1100' está en la clase 0

(Observemos que nuestros ejemplos tienen cuatro atributos binarios, con valor 0 o 1. De esta forma la red tiene cuatro nodos de entrada. La salida especifica si el ejemplo esta o no en una determinada clase. Así, la red tiene un sólo nodo de salida.)

- Los pesos correctos sobre cada uno de los arcos pasan a ser -1, -1, +1, y +1. Esto nos da la clase correcta para cada ejemplo, como se muestra en la gráfica.



Entrenamiento de una Red Neuronal Artificial



- ❑ **Algoritmo de retropropagación del error “backpropagation” para el entrenamiento de una red neuronal.**
- ❑ **1: Se selecciona un patrón del conjunto de entrenamiento y se introduce en la red neuronal.**
- ❑ **2: Los valores de las entradas se propagan a lo largo de la red hacia la capa de salida.**
- ❑ **3: Se calcula el error cuadrático de la salida (la salida deseada menos la salida del modelo de red, elevado al cuadrado).**
- ❑ **4: El error se propaga hacia atrás a través de la red de forma tal que se puedan cambiar los pesos.**
- ❑ **5: Ir a la etapa 1**



Entrenamiento de una Red Neuronal Artificial



- ❑ Algoritmo de retropropagación del error “backpropagation” para el entrenamiento de una red neuronal.
- 6.- Se repite el proceso hasta que haya sido procesado cada patron del conjunto de entrenamiento. A continuación la red resultante se chequea con el conjunto de validación (si esta red es mejor que una red previa mejor, entonces se salva)
- 7.- Estas etapas se repiten hasta que se descubra una ANN que no mejore en CCR a la mejor red obtenida hasta entonces.



Dinámica del aprendizaje:

- Calcula en primer lugar, los cambios de los pesos sinápticos de la neurona de salida;
- Calcula los cambios hacia atrás empezando por la capa $p-1$, y propaga hacia atrás los términos del error local.
- El método es relativamente complicado pero menos que el problema original de optimización



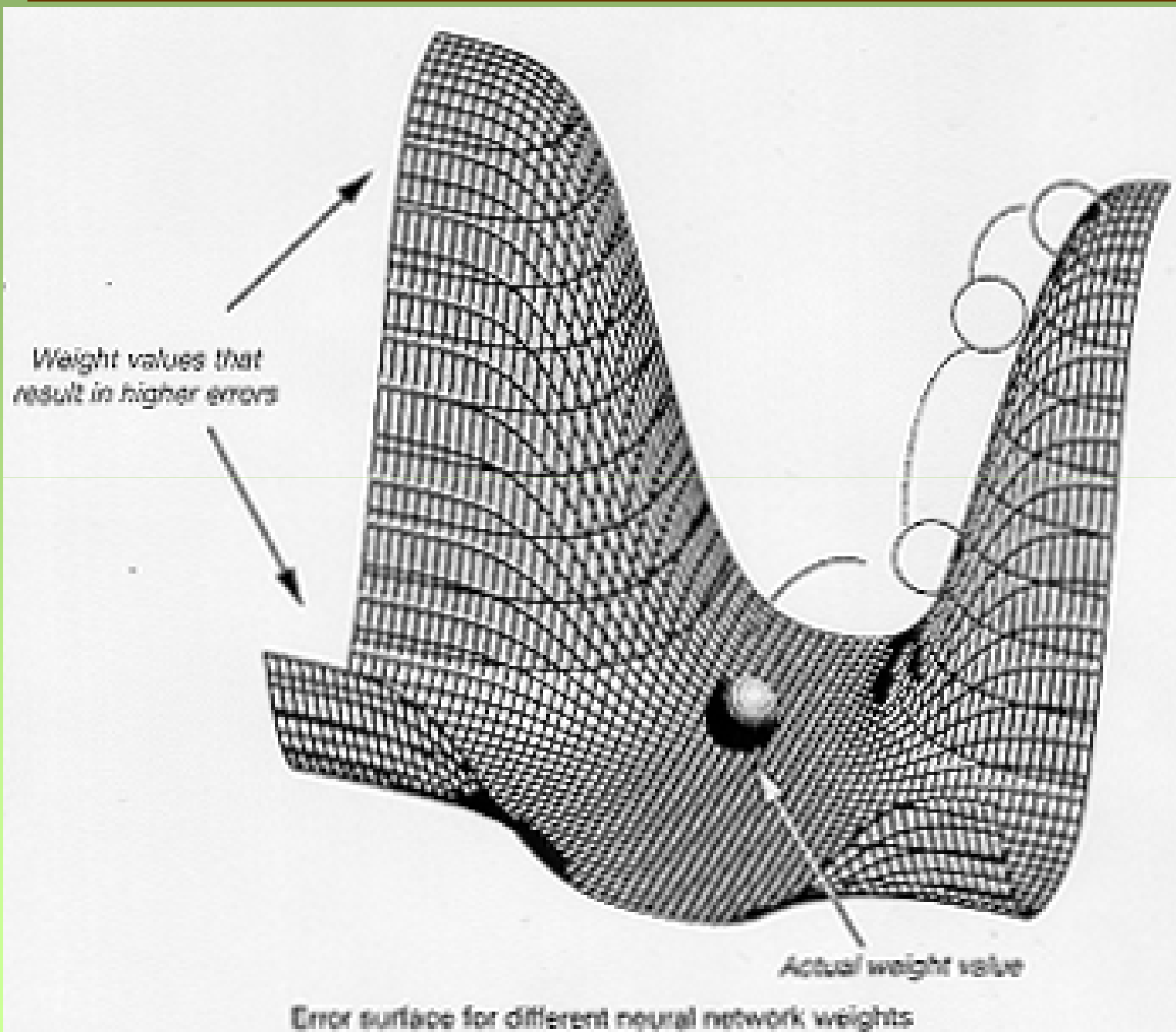
ALGORITMO DE RETROPROPAGACIÓN DEL ERROR



- **Back-propagation** es un procedimiento para ajustar los pesos de la red con el fin de minimizar el error de salida. La técnica empleada es un algoritmo de búsqueda local de **gradiente descendente**: se calcula la derivada de las funciones de transferencia y, para cada peso, se decide si añadir o restar una pequeña cantidad con el objetivo de reducir el error total.
- Supongamos que podemos ajustar dos pesos (w_1 y w_2). La superficie siguiente es un ejemplo de como puede variar el error como resultado de la combinación de esos pesos diferentes.
- Observamos que la dirección (positiva o negativa) y las etapas de las pendientes se pueden obtener utilizando derivadas parciales del error con respecto a w_1 y w_2 .



ALGORITMO DE RETROPROPAGACIÓN DEL ERROR

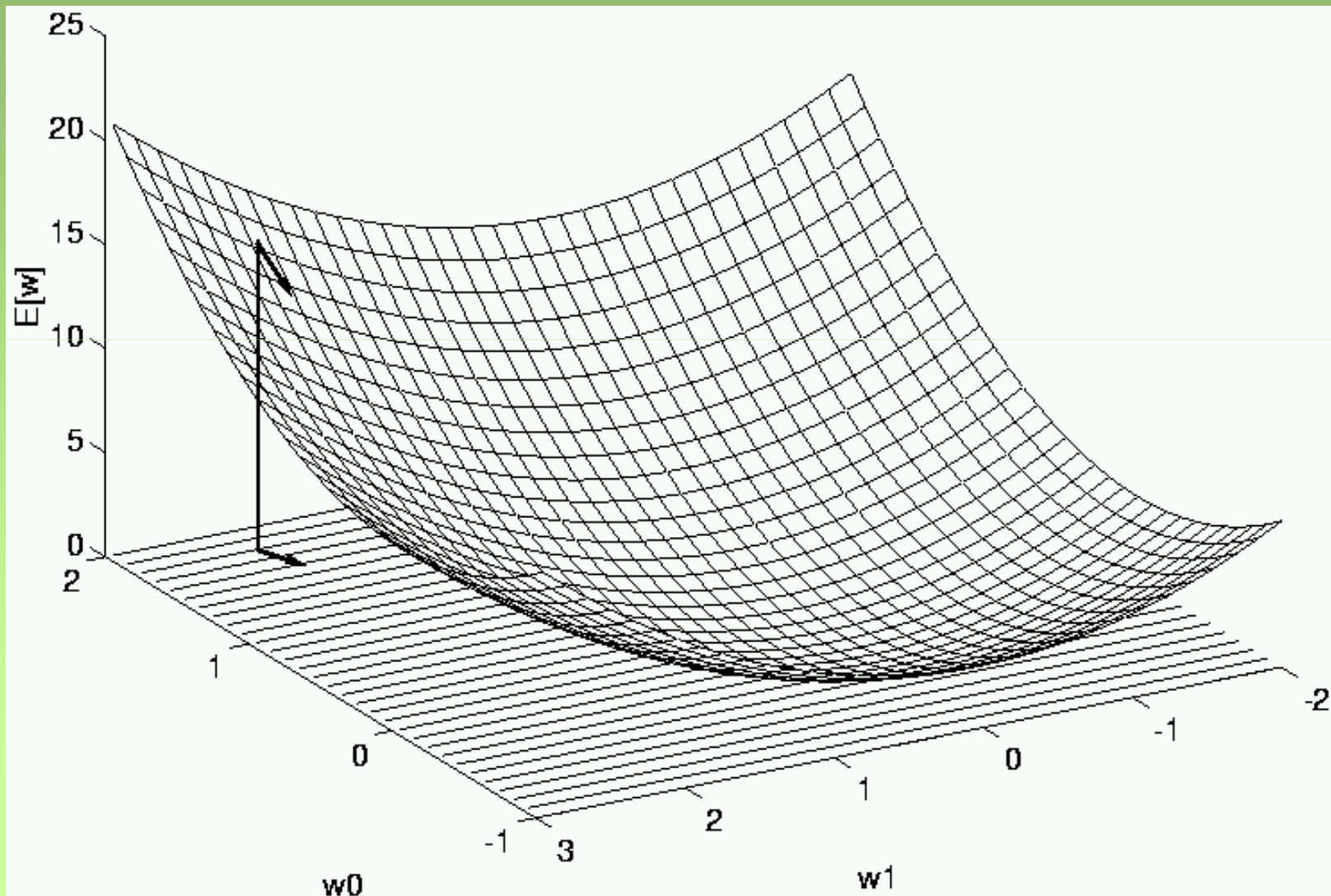




GRADIENTE DESCENDENTE



Aqui mostramos otra gráfica para ver como el error de predicción puede cambiar cuando cambian los valores de los pesos:





Experimentos



Primer Experimento: Datos Simulados



- ❑ **Datos simulados del conjunto de entrenamiento.**
- ❑ **Se introduce un 1% o un 2% de ruido Gaussiano a los patrones**
- ❑ **Datos igualmente simulados para el conjunto de generalización**

Nº de patrones: 100

Rango de los Exponentes: $[-5,5]$

Rango de los Coeficientes: $[-10,10]$

Nº de redes en la población : 1000



Resultados. Una Variable

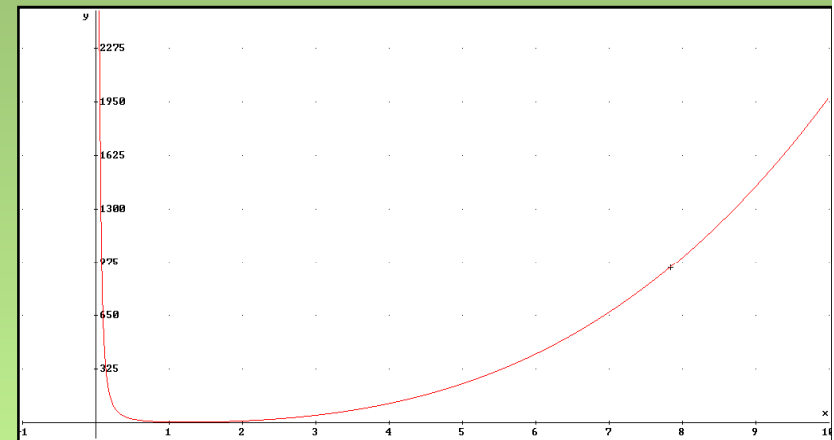
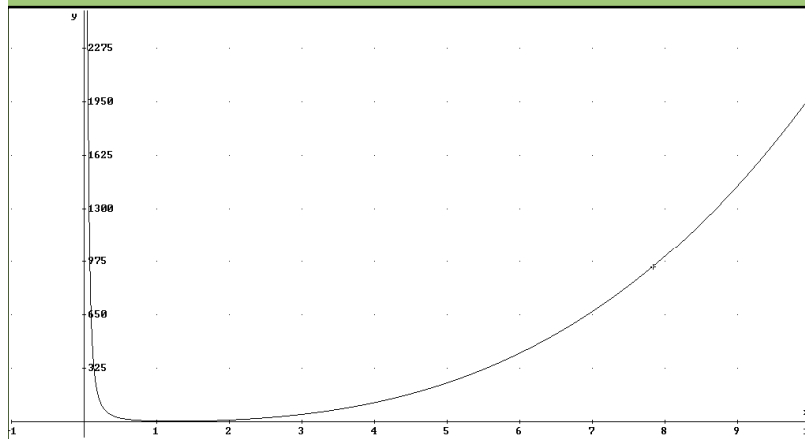


Función estimada

Función objetivo

$$f(x) = 2x^3 + x\sqrt{x} - 5x + \frac{6}{x^2}$$

$$f_p(x) = 2,0433x^{2,9933} - \frac{4,0219}{x^{2.2494}} - 3.8345x^{0,9159} + \frac{9,7519}{x^{2.1309}}$$



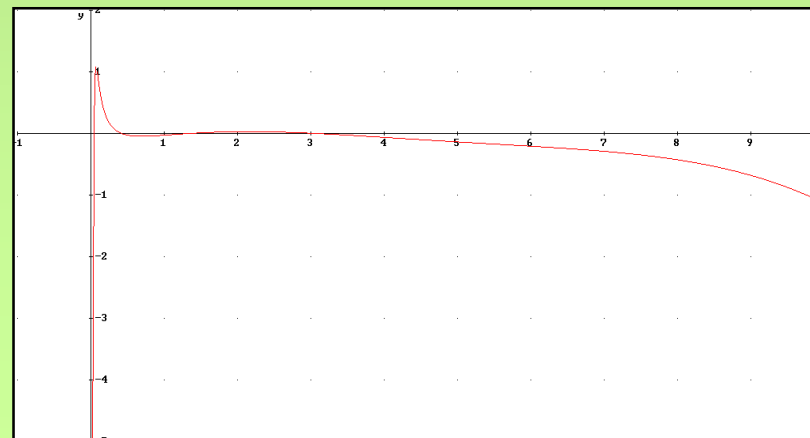
Superficie de error

de Generaciones: 2181

R2=0,9998

Error de los datos= 1%

suma de Residuos=679



$$f(x) - f_p(x)$$



Resultados. Dos Variables

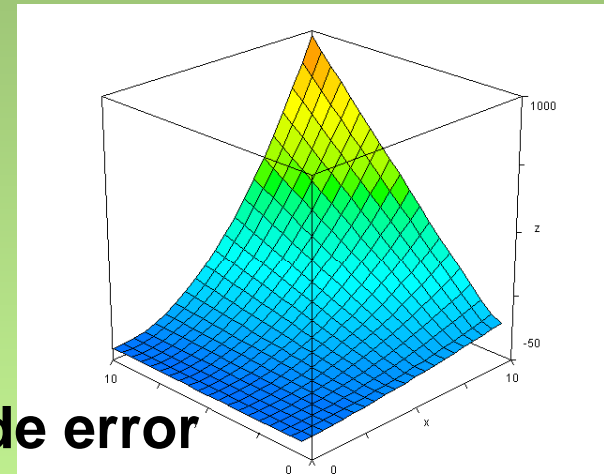
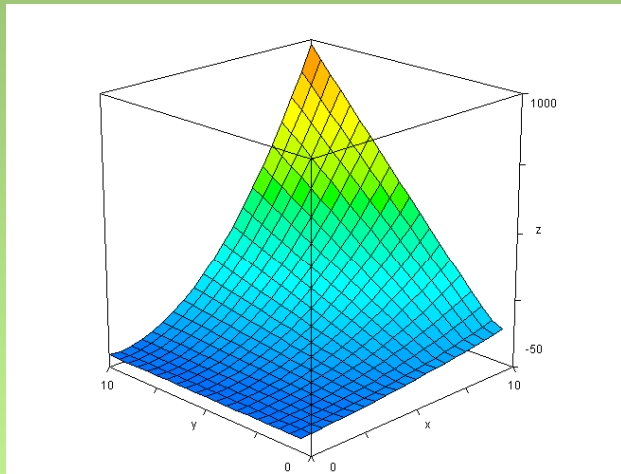


Función objetivo

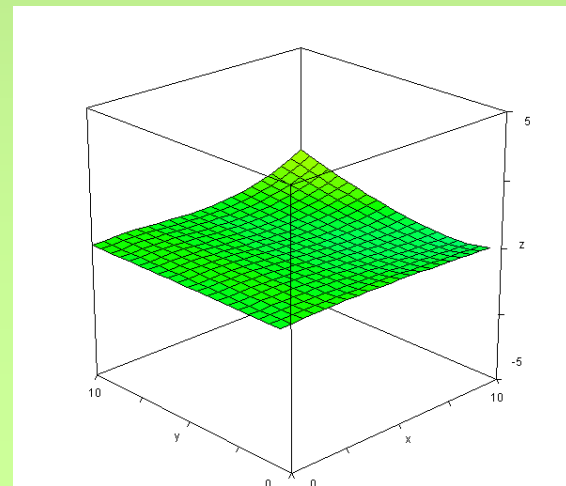
$$f(x, y) = -2\sqrt{xy} + x^2y + \frac{x}{y^2}$$

Función estimada

$$f_p(x, y) = -2,0011x^{0,521}y^{0,5099} + 1,0161x^{1,9954}y^{0,9977} \\ + 0,9852x^{1,0048}y^{-2,0033}$$



Superficie de error



Nº de Generaciones: 2000

R2=0,9995

Error de los datos= 1%

Suma de Residuos=50,09



Resultados. Tres Variables



Función estimada

$$f_p(x, y, z) = \frac{1,3313x^{1,9385}y^{0,9782}}{z^{2,0195}} - \frac{0,3884x^{1,6294}y^{0,8114}}{z^{2,1265}}$$

Función objetivo

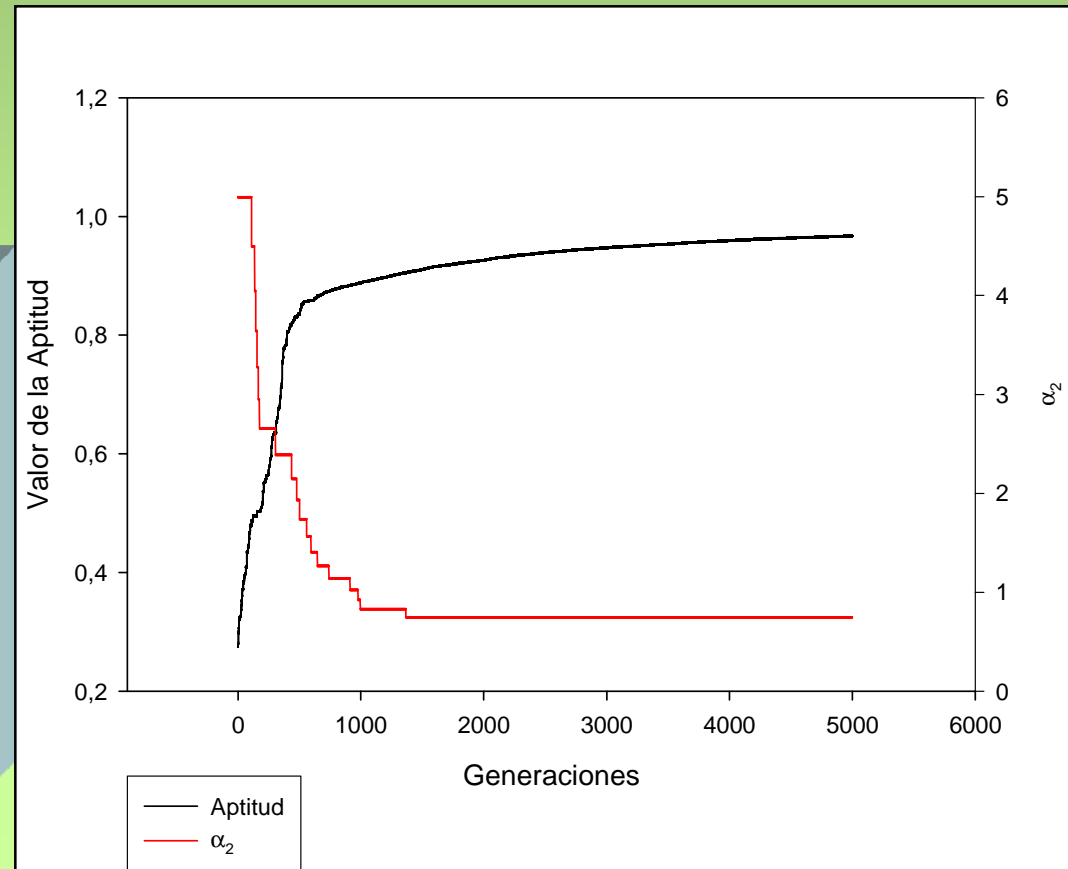
$$f(x, y, z) = \frac{x^2y}{z^2} - 3\frac{zy^3}{x^2} + 2\sqrt{xyz}$$
$$- \frac{2,9462y^{3,0123}z^{0,9954}}{x^{2,0037}} + 1,9054x^{0,5513}y^{0,4816}z^{0,4765}$$

Nº de Generaciones: 5000

$R^2=0,999$

Error de los datos= 1%

Suma de Residuos=138699





Aplicaciones. Modelos de Crecimiento Microbiano



Regresión en Microbiología Predictiva



Conocimiento detallado de la respuesta de crecimiento de los microorganismos en los alimentos frente a los factores ambientales que les afectan y a partir de los datos resultantes, predecir lo que sucederá durante el almacenamiento, distribución, etc.



Estimar la vida comercial
y la seguridad
microbiológica



Implantar medidas
para limitar el
desarrollo microbiano

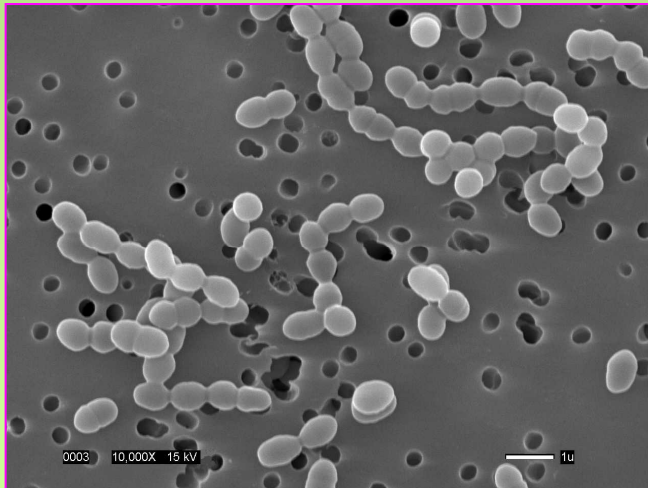


Regresión en Microbiología Predictiva



Leuconostoc mesenteroides

- ➔ Microorganismo responsable de alteración de diferentes alimentos
- ➔ Ampliamente distribuido: plantas, lácteos y productos alimenticios
- ➔ Incluido en Bacterias ácido lácticas (BAL)



- ▣ Gram +
- ▣ Coccoide (agar)/ Bacilar (caldo)
- ▣ Anaerobio facultativo
- ▣ No móvil
- ▣ No esporas
- ▣ Rango de Tª: 10-37°C



Crecimiento Microbiano



Variables independientes (4)

- Temperatura
- Concentración NaNO_2
- Concentración de NaCl

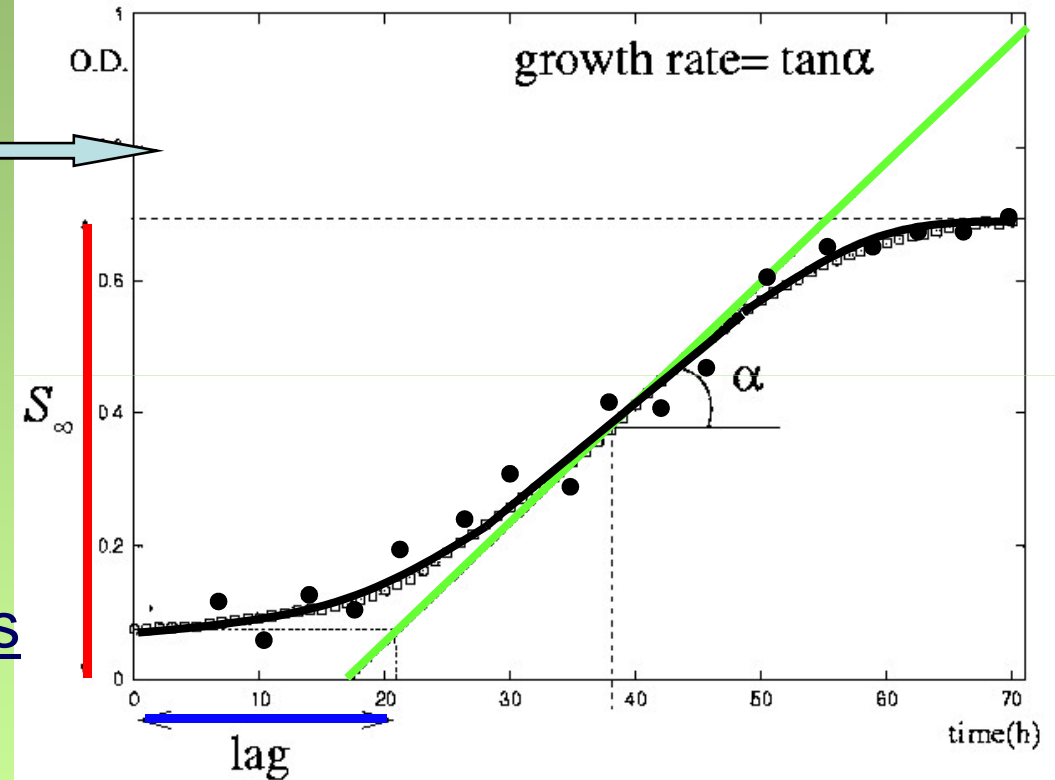
• pH

Variables dependientes

- Lag
- Growth rate
- Yend

Modelo
Primario

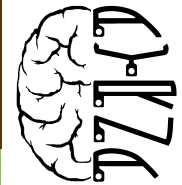
Modelo
Secundario



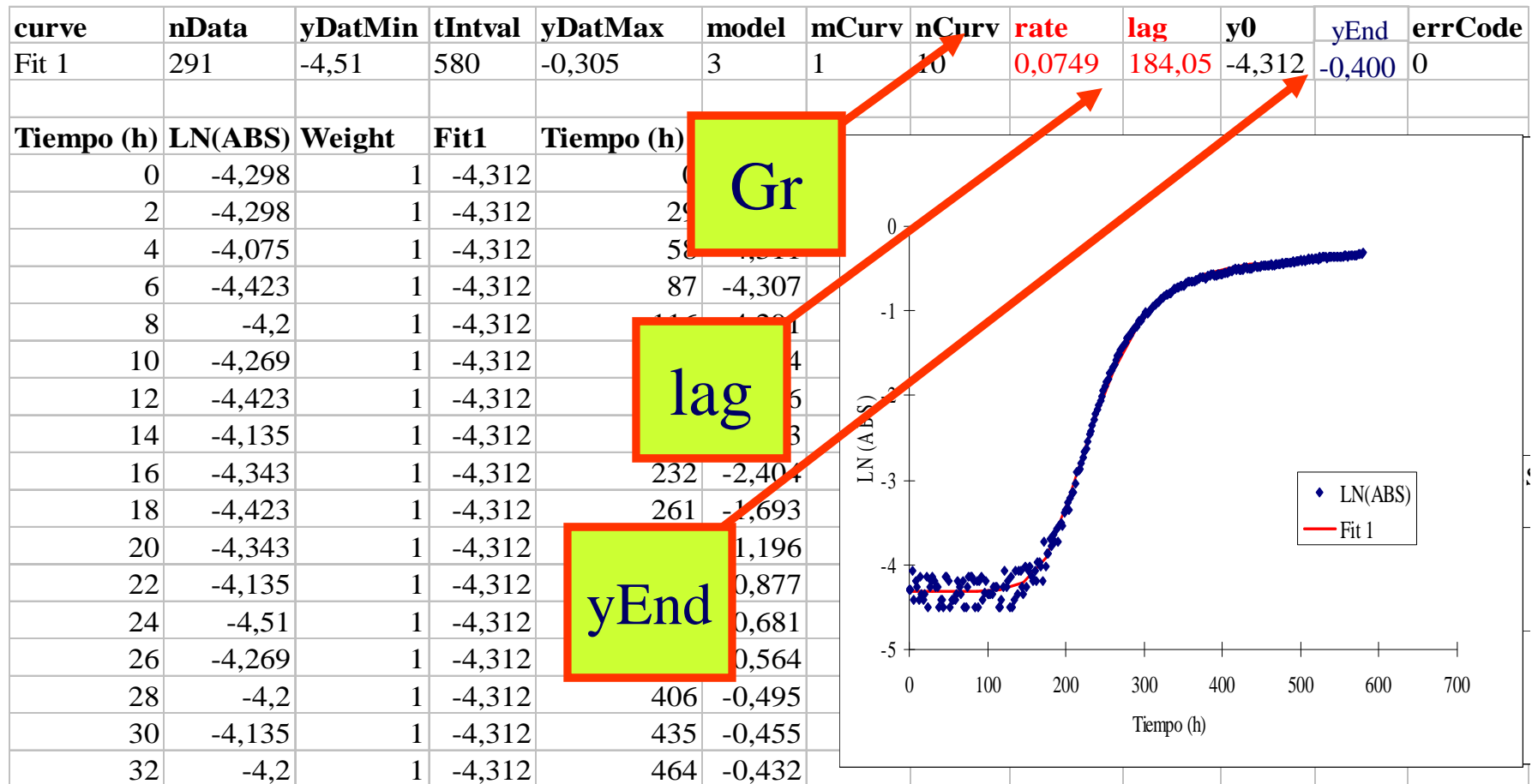
Ecuación Baranyi y Roberts



Crecimiento Microbiano



LN (ABS) vs tiempo $\xrightarrow{\text{DMFit}}$ Baranyi y Roberts, 1994 \rightarrow MODELO PRIMARIO





Metodología



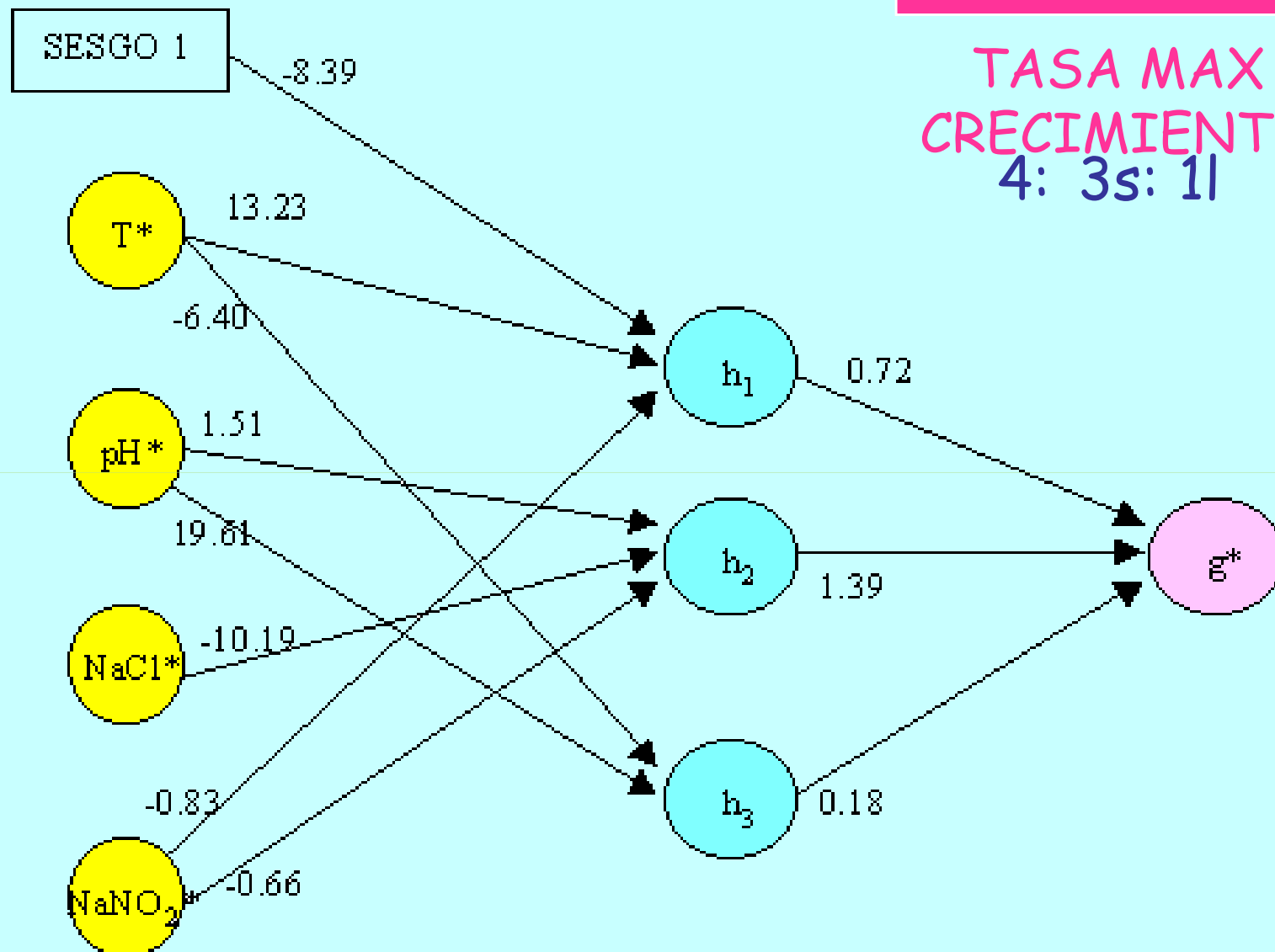
- ❑ Se utilizaron 150 curvas para entrenar y 60 para generalizar
- ❑ Las variables independientes se normalizan en el intervalo [0,1,1,1]
- ❑ Las variables dependientes se normalizan en el intervalo [1,2]
- ❑ Comparación con algoritmo genético con redes de unidades sigmoides
- ❑ Misma topología en redes de unidades producto UP y sigmoides US 4:4:1
- ❑ Se ejecutó el algoritmo 30 veces consecutivas
- ❑ Medida de comparación. SEP

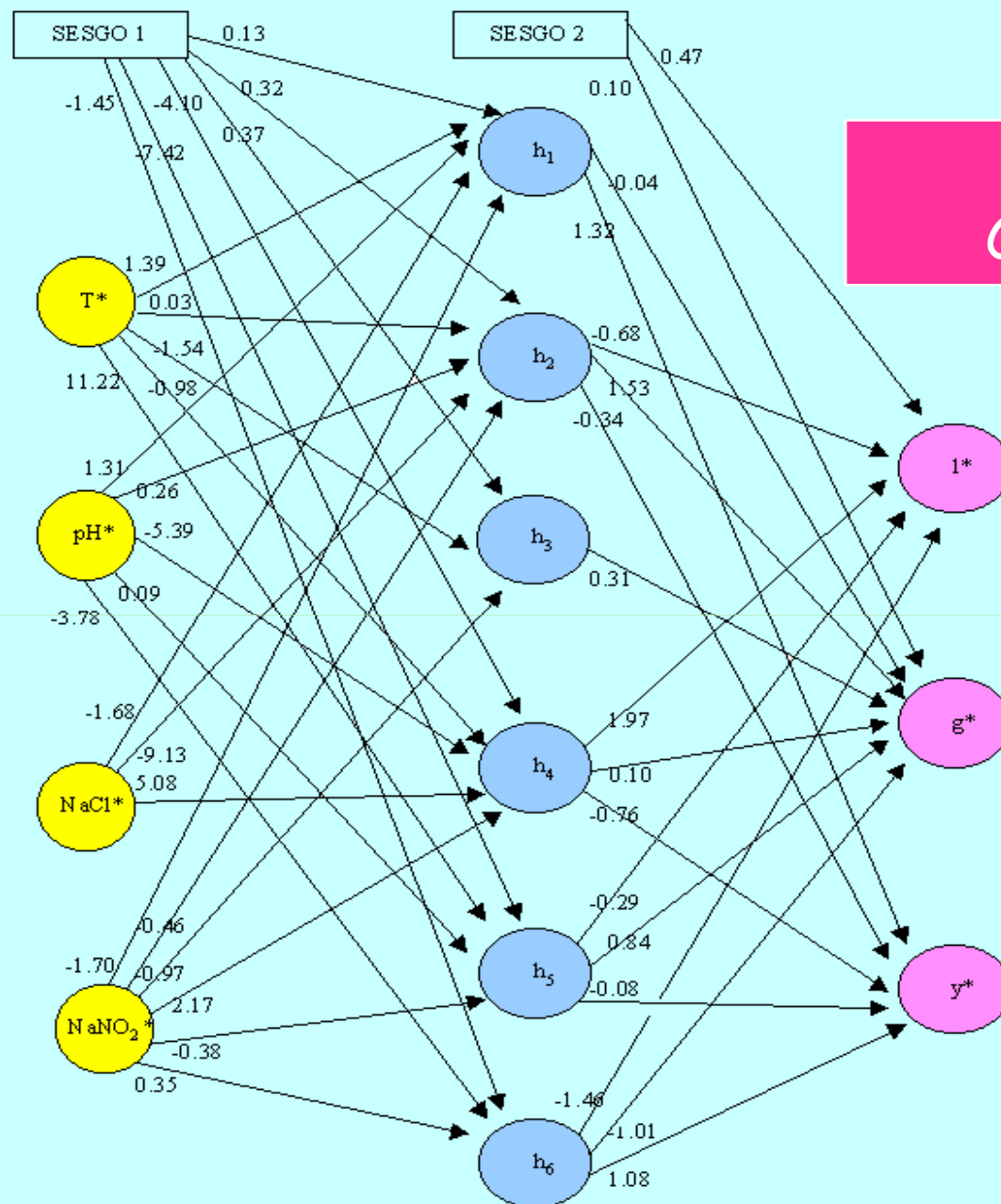
$$SEP = \frac{100}{|\bar{y}|} \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}}$$



Gr

TASA MAX
CRECIMIENTO
4: 3s: 1l





MODELO
CONJUNTO



Crecimiento Microbiano.

Ejemplo de interpretación de modelos



$$Grate^* = 2.9118(T^*)^{1.912}$$

~~$$+ 6.0509(T^*)^{6.548}(NaCl^*)^{5.246}$$~~

~~$$+ 1.8178(T^*)^{0.167}(pH^*)^{0.127}(NaCl^*)^{0.112}$$~~

~~$$- 0.2551(T^*)^{0.107}(pH^*)^{1.548}(NaCl^*)^{0.398}$$~~

~~$$- 4.3718(T^*)^{1.817}(pH^*)^{0.555}(NaNO_2^*)^{0.042}$$~~

$$pH^* \leq 0.1$$



$$Grate^* = 2.9118(T^*)^{1.912}$$

$$+ 6.0509(T^*)^{6.548}(NaCl^*)^{5.246}$$

$$+ 1.8178(T^*)^{0.167}(pH^*)^{0.127}(NaCl^*)^{0.112}$$

$$- 4.3718(T^*)^{1.817}(pH^*)^{0.555}(NaNO_2^*)^{0.042}$$

Término Básico

$$T^* \leq 0.35 \wedge NaCl^* \leq 0.35$$

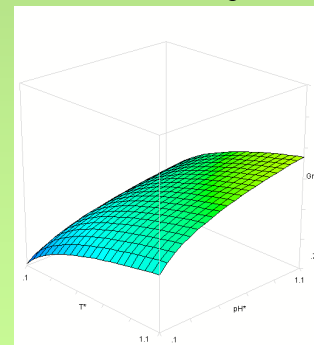
$$S_3 = 1.8178(T^*)^{0.167}(pH^*)^{0.127}(NaCl^*)^{0.112}$$

$$Grate^* = 2.9118(T^*)^{1.912}$$

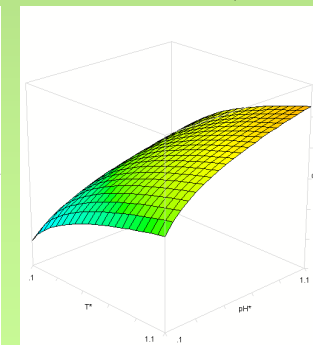
$$+ 1.8178(T^*)^{0.167}(pH^*)^{0.127}(NaCl^*)^{0.112}$$

$$- 0.2551(T^*)^{0.107}(pH^*)^{1.548}(NaCl^*)^{0.398}$$

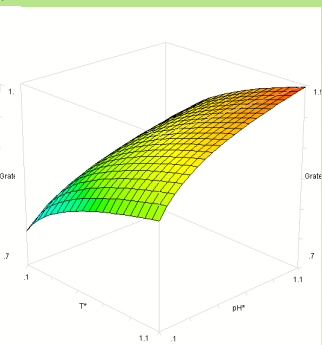
$$- 4.3718(T^*)^{1.817}(pH^*)^{0.555}(NaNO_2^*)^{0.042}$$



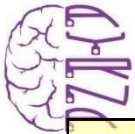
(NaCl*=0.1)



(NaCl*=0.6)



(NaCl*=1.1)



Ejemplo modelo de red neuronal

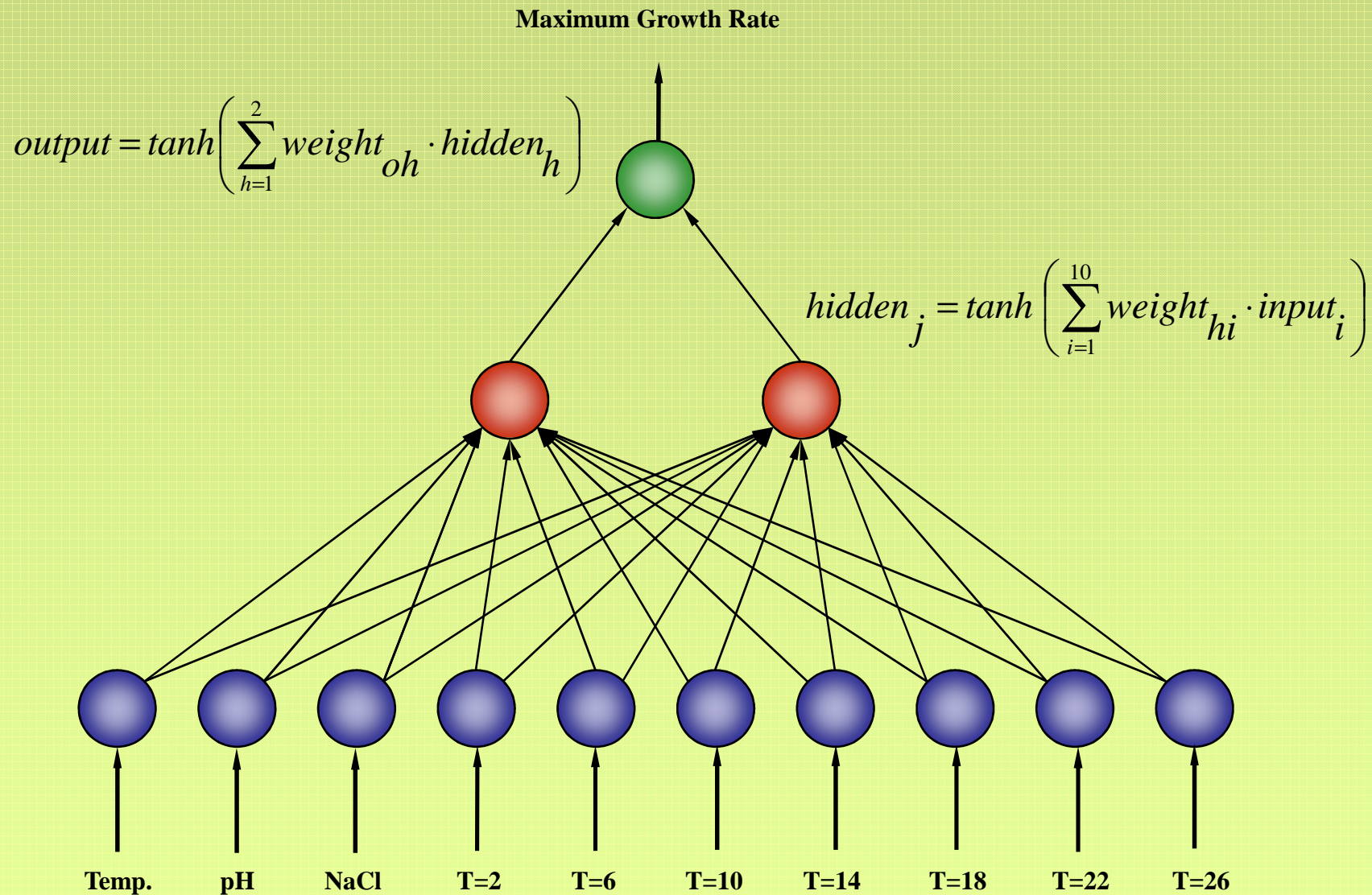


Figura: Red Neuronal predictiva para modelos de crecimiento microbiano



Resultados Base de datos indias Pima



Mejor modelo BBDD PIMA con unidades producto PU con 2, 4, 5 nodos en capa oculta

Iteration 2 - Generation 30 - Methodology EP

$-3.900512587963034 * (x_1^{-0.13684235372038445} * x_2^{0.8947057849519509} * x_3^{-0.16177688100613585} * x_4^{-0.22211012576697753} * x_6^{0.4913432471220496} * x_7^{0.23623427912780293})$

$+4.879092963310238 * (x_1^{0.11285972138120698} * x_2^{-1.3619173861726994} * x_3^{0.9628706545339946} * x_5^{0.05405779792187547} * x_6^{-1.5084101778277188})$

$+6.132961230744894 * (x_1^{-1.409023891889148} * x_2^{-0.7106040581439343} * x_3^{-0.5403167304727159} * x_4^{-0.28968502343526836} * x_5^{0.555395799909222} * x_7^{-0.4734870655117997})$

$-0.11286664464330956 * (x_1^{0.8983631146643081} * x_2^{-1.7527372637244092} * x_6^{0.8171140881579099} * x_7^{0.2618757435172636} * x_8^{-1.956517944452171})$

$+2.5127160041976815 * (1)$

Fitness: 0.010812206809437544

Number of hidden neurons: 4 Number of effective links: 27



Resultados base de datos indias Pima



Train CCR: 76.5625				75.8680556	17
Train Confussion Matrix				76.5625000	27
Pred.	0	1	CCR	75.8680556	13
Target				76.7361111	21
0	325	40	0.8904	76.9097222	21
1	95	116	0.5497		
				76.3888889	19.8000000
				0.4910464	5.2153619
Test CCR: 82.2916				78.1250000	17
Test Confussion Matrix				82.2916667	27
Predicted	0	1	CCR	79.1666667	13
Target				79.1666667	21
0	123	12	0.9111	79.1666667	21
1	22	35	0.6140		
Train Log: 0.4912				79.5833333	19.8000000
TrainChisq: 91.488				1.5797657	5.2153619



Resultados Base de datos Grate



Mejor modelo con PU para Grate 1, 2, 5 numero de nodos. 1000 individuos
Iteration 4 - Generation 400 - Methodology EP

$$\begin{aligned} &1.3892874334462106 * (x1^{1.1585166569836731} * x4^{-} \\ &0.1051251125212078) \\ &+1.7863616190220453 * (x1^{-4.087978726705221} * x3^{-4.69547483185112} \\ &* x4^{0.18663881866740797}) \\ &+0.1956211641178407 * (x1^{-3.7369045659863787} * \\ &x3^{2.4068525108311905} * x4^{0.18625837668354234}) \\ &-0.17422043783551888 * (x1^{0.12967881626147726} * x2^{-} \\ &3.1639588170762014 * x4^{0.3236642700125189}) \\ &+0.3228931353820062 * (x1^{0.8356221719188421} * x3^{-} \\ &4.572180685852902) \\ &-1.0603249747825179 * (1) \end{aligned}$$

Fitness: 0.9961592282738658

Number of hidden neurons: 5 Number of effective links: 19

Train MSE: 3.948114052385393E-4

Test MSE: 3.771368707920658E-4

Train SEP: 9.28527137842698

Test SEP: 9.109535824243359



INTRODUCCION A LAS REDES NEURONALES (II)

César Hervás-Martínez
Grupo de Investigación AYRNA

**Departamento de Informática y Análisis
Numérico**
Universidad de Córdoba
Campus de Rabanales. Edificio Einstein.
Email: chervas@uco.es

2019-2020