

Tema 9: Aprendizaje

Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga



Contenido

- [**1. Introducción**](#)
- [**2. Neurona Artificial**](#)
- [**3. Perceptrón simple**](#)
- [**4. Perceptrón multicapa**](#)
- [**5. Conclusiones**](#)

1. Introducción



1. Introducción Aprendizaje

- Aprendizaje: utiliza percepciones para actuar y mejorar la habilidad del agente en el futuro.
 - Agente: observa, memoriza experiencias, crea teorías científicas.
- Agente es diseñado:
 - **Elemento de acción:** decide qué acciones llevar a cabo.
 - **Elemento de aprendizaje:** modifica el **elemento de acción** para poder tomar mejores decisiones.
- Elemento de aprendizaje:
 - Componente
 - Realimentación: factor más importante
 - **Supervisado:** aprender una función a partir de ejemplos de sus entradas y sus salidas, par $(x, f(x))$.
 - **No supervisado:** aprender de patrones de entrada de los que no se especifica la salida, x .
 - **Refuerzo:** aprende por recompensa, suele incluir aprender cómo se comporta el entorno.
 - Representación:
 - Polinomios lineales con peso para representar funciones de utilidad (programas de teoría de Juegos).
 - Sentencias en Lógica Proposicional/Lógica de Primer Orden.
 - Descripciones probabilísticas para las componentes de inferencia de un agente basado en la teoría de la decisión.

1. Introducción

Inspiración biológica: el cerebro

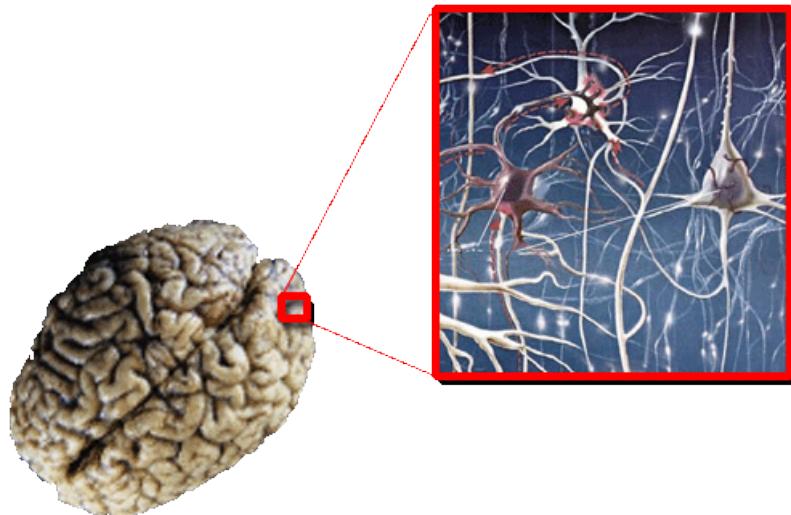
10^{11} neuronas

+20 tipos neuronas

10^{14} sinapsis

Tiempo de respuesta

- 1-10 ms

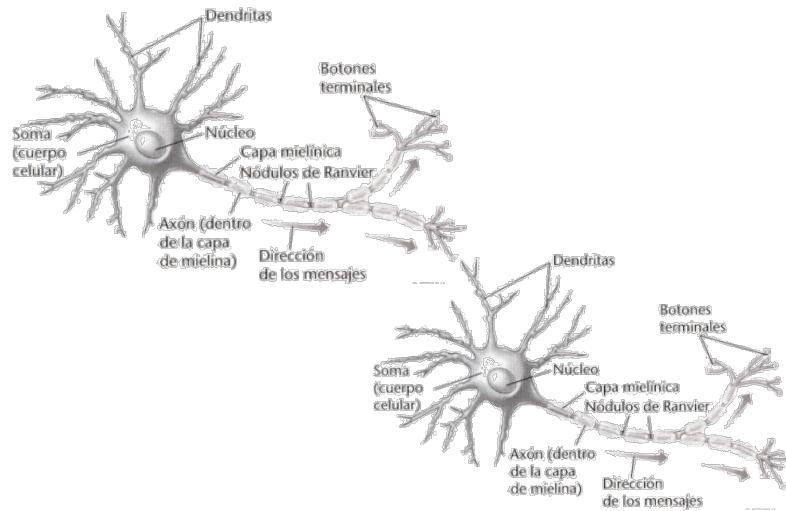


- Órgano responsable de la cognición, las emociones, la memoria y el aprendizaje.
- No se entiende el funcionamiento completo del cerebro.
- Sólo algunas partes son esenciales para el procesamiento de la información.

1. Introducción

Inspiración biológica: la neurona

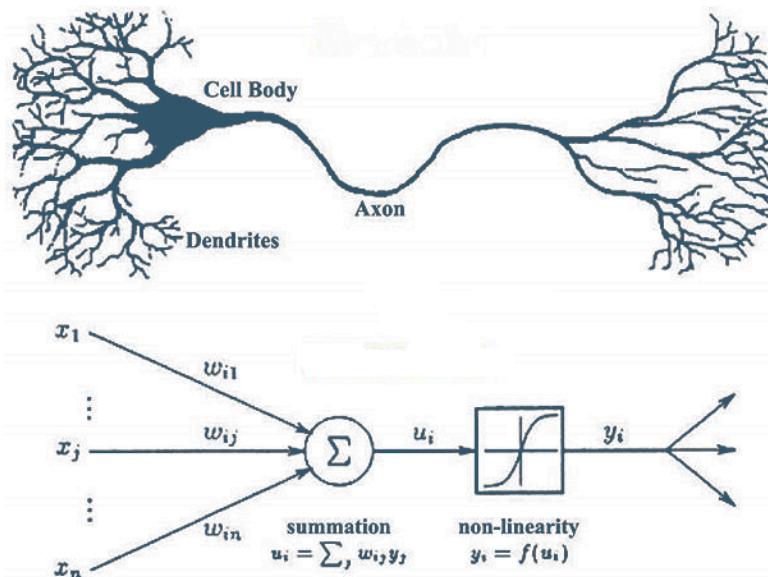
- Unidad de procesamiento elemental.
- Las señales recibidas (a través de las **dendritas**) de otras neuronas mediante las **sinapsis** se combinan en el **soma** produciendo un potencial en el cuerpo celular.
- Las señales emitidas (a través del **axón**) dependerán del potencial alcanzado.



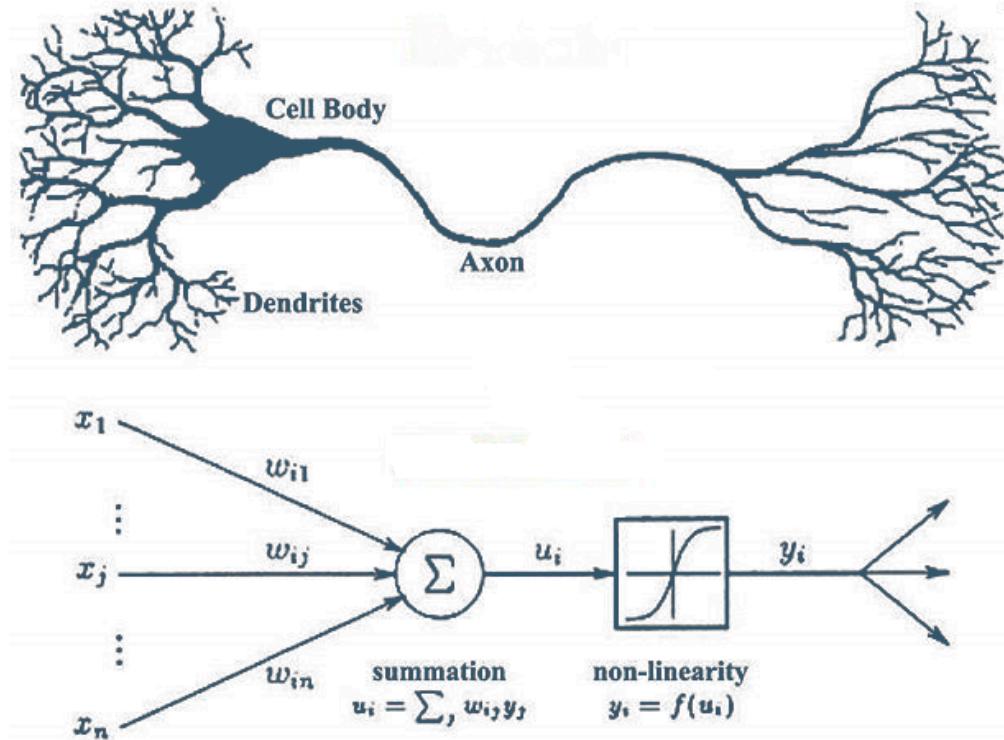
1. Introducción

Inspiración biológica: la neurona

- Unidad de procesamiento elemental.
- Las señales recibidas (a través de las **dendritas**) de otras neuronas mediante las **sinapsis** se combinan en el **soma** produciendo un potencial en el cuerpo celular.
- Las señales emitidas (a través del **axón**) dependerán del potencial alcanzado.



2. Neurona Artificial



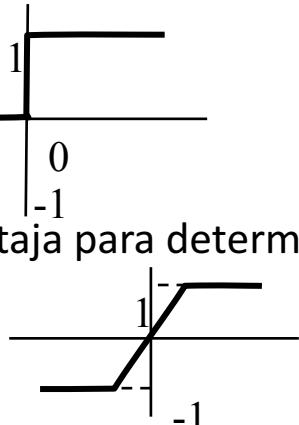
2. Neurona Artificial

Definición

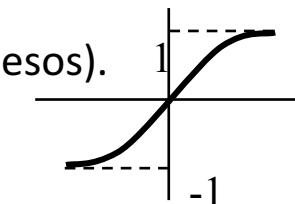
- Pesos sinápticos
- Umbral de activación
- Función de activación
 - Activa la neurona cuando las entradas son correctas
 - Desactiva la neurona cuando las entradas son incorrectas.
 - Función no lineal.

- Tipos:

- Escalón o paso [0,1]
- Signo [-1,1]
- Sigmoide: diferenciable (ventaja para determinar los pesos).
- Rampa
- ...

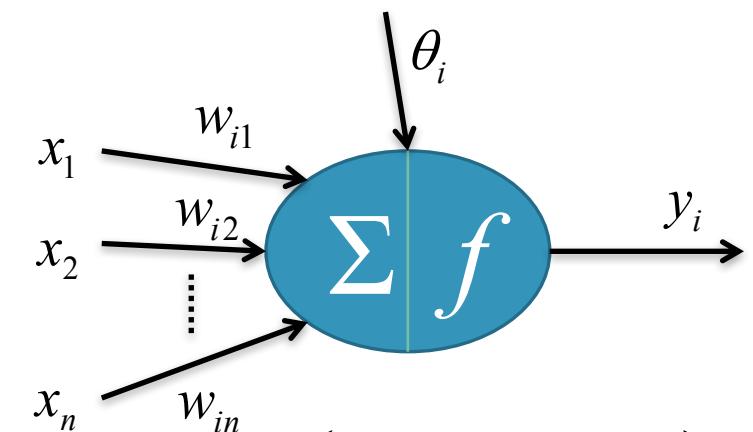


- Estado de la neurona:
 - Discreto: binario (0,1), bipolar (-1,1)...
 - Continuo



Función de activación

McCulloch & Pitts, 1943



$$y_i = f\left(\sum_{j=1}^n w_{ij}x_j - \theta_i\right)$$

Potencial sináptico o
Potencial de activación

2. Neurona Artificial

Ejemplo. Suma ponderada

- ❖ Genera la salida de esta neurona:

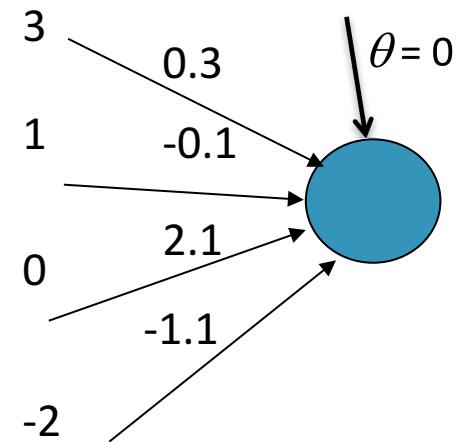
$$y_i = f\left(\underbrace{\sum_{j=1}^n w_{ij}x_j}_{\text{Potencial de activación: suma ponderada}} - \theta_i \right)$$

Potencial de activación: suma ponderada - θ

1.- Suma ponderada:

Entrada: $(3, 1, 0, -2)$

Suma ponderada: $3(0.3) + 1(-0.1) + 0(2.1) + -1.1(-2) = 0.9 + (-0.1) + 2.2 = 3$



2. Neurona Artificial

Ejemplo. Función de activación

2.- Aplicar Función de Activación:

$$y_i = f\left(\sum_{j=1}^n w_{ij}x_j - \theta_i\right)$$

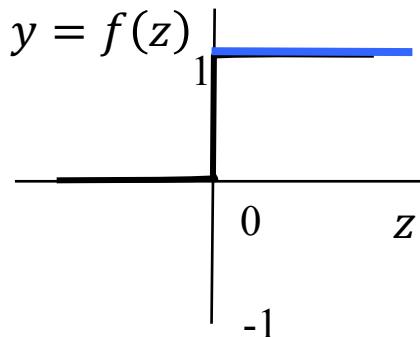
Potencial de activación: z

Elegimos una Función de Activación:

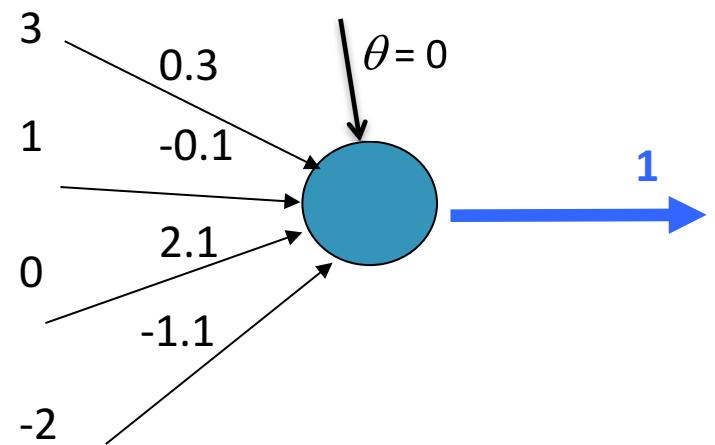
- neurona en estado binario.

$$f(z) = \begin{cases} 1 & \text{si } z \geq 0 \\ 0 & \text{si } z < 0 \end{cases}$$

Función paso



$y = 1$ La neurona se activa.



2. Neurona Artificial

Salida de la neurona: Función de activación

2.- Aplicar Función de Activación:

$$y_i = f\left(\sum_{j=1}^n w_{ij}x_j - \theta_i\right)$$

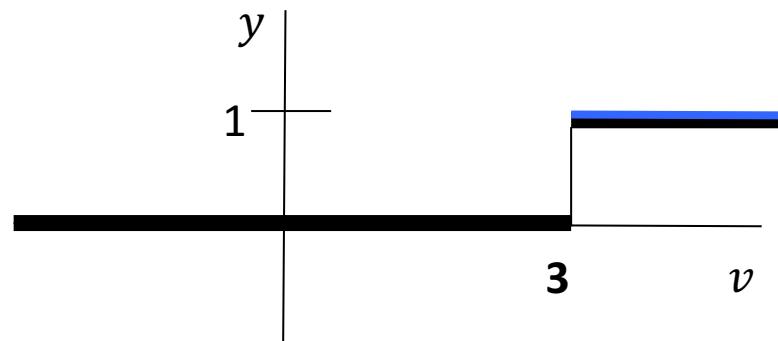
Potencial de activación: $v - \theta$

Elegimos una Función de Activación:

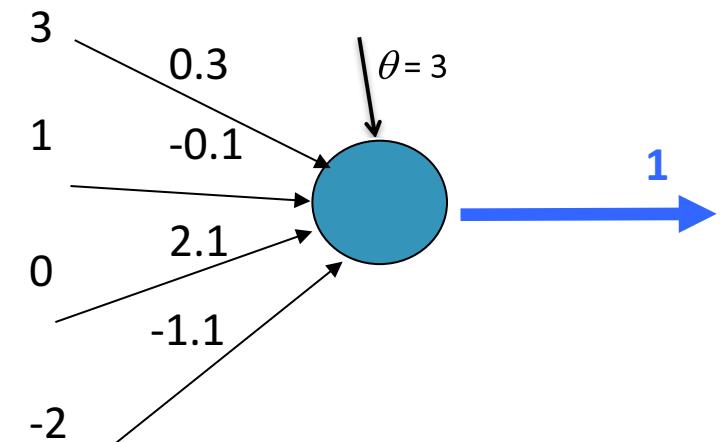
- neurona en estado binario.

$$f(v - \theta) = \begin{cases} 1 & \text{si } v \geq \theta \\ 0 & \text{si } v < \theta \end{cases}$$

Función paso



Para estos valores de entrada ($v=3$), cuando $\theta = 3$ la neurona se dispara.



Ejercicio 1. Enunciado

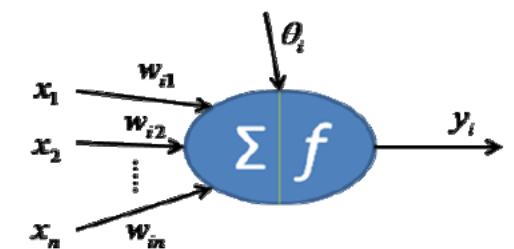
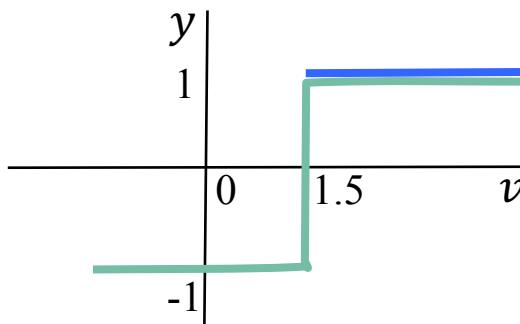
Diseñar un perceptrón simple para clasificar a los alumnos que pueden acceder a una beca de investigación. Los alumnos aptos de una beca de investigación deberán tener una nota media superior a 1,5 de las 40 asignaturas cursadas por los alumnos. Las calificaciones de las asignaturas se puntuán con 0 (suspenso), 1 (aprobado), 2 (notable), 3 (sobresaliente) y 4 (matrícula de honor). Determinar el número de neuronas, así como los pesos sinápticos y umbrales de la red neuronal propuesta.

Ejercicio 1. Solución

Diseñar un perceptrón simple para clasificar a los alumnos que pueden acceder a una beca de investigación. Los alumnos aptos de una beca de investigación deberán tener una nota media superior a 1,5 de las 40 asignaturas cursadas por los alumnos. Las calificaciones de las asignaturas se puntuán con 0 (suspenso), 1 (aprobado), 2 (notable), 3 (sobresaliente) y 4 (matrícula de honor). Determinar el número de neuronas, así como los pesos sinápticos y umbrales de la red neuronal propuesta.

Necesitamos un perceptrón simple con 40 entradas ($n=40$) y una única neurona de salida, donde los pesos sinápticos tendrán un valor de $1/40$ y el umbral de 1,5. La función de activación es la función signo, con lo que:

$$y = \text{sgn}\left(\sum_{i=1}^{40} \frac{1}{40} x_i - 1.5\right)$$



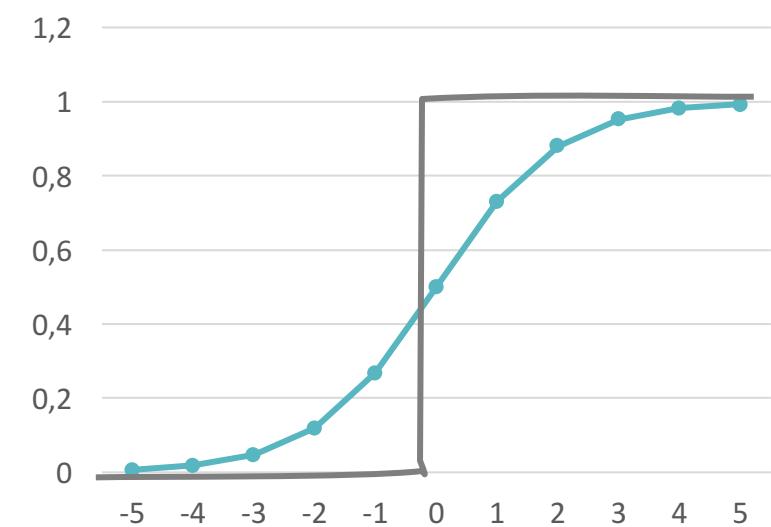
2. Neurona Artificial Otra Función Activación: Sigmoide

- Algunas redes neuronales necesitan que la función de activación sea **continuamente derivable**.
- La **Función Sigmoide (Función Logística)** es a menudo utilizada para aproximar la función escalón.

$$y_i = f\left(\sum_{j=1}^n w_{ij}x_j - \theta_i\right)$$

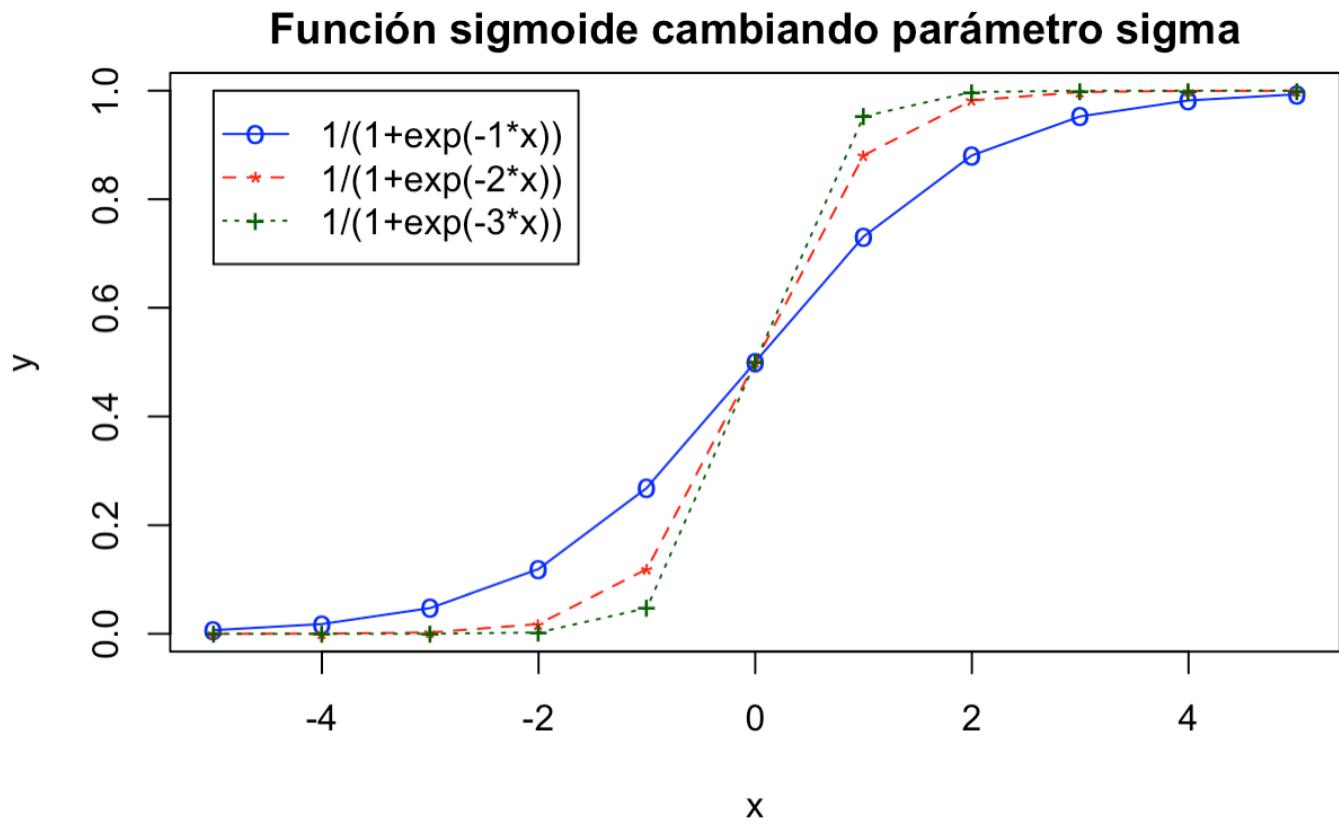
Función de activación: Función Sigmoide

$$f(z) = \frac{1}{1 + e^{-\sigma \cdot z}}$$



2. Neurona Artificial Otra Función Activación: Sigmoide

$$y = f(x) = \frac{1}{1 + e^{-\sigma \cdot x}}$$



2. Neurona Artificial Ejemplo Sísmoide

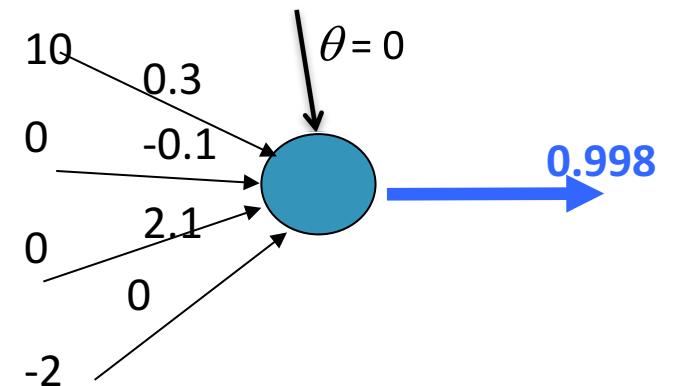
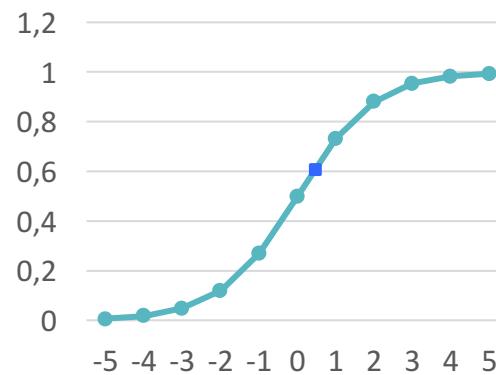
- ❖ Dada la siguiente neurona con sus entradas y pesos. Calcular el valor de activación de la neurona cuando el sesgo es 0, utilizando como función de activación la Función Sísmoide.

El valor del potencial de activación sería 3.

$$y_i = f\left(\sum_{j=1}^n w_{ij}x_j - \theta_i\right)$$

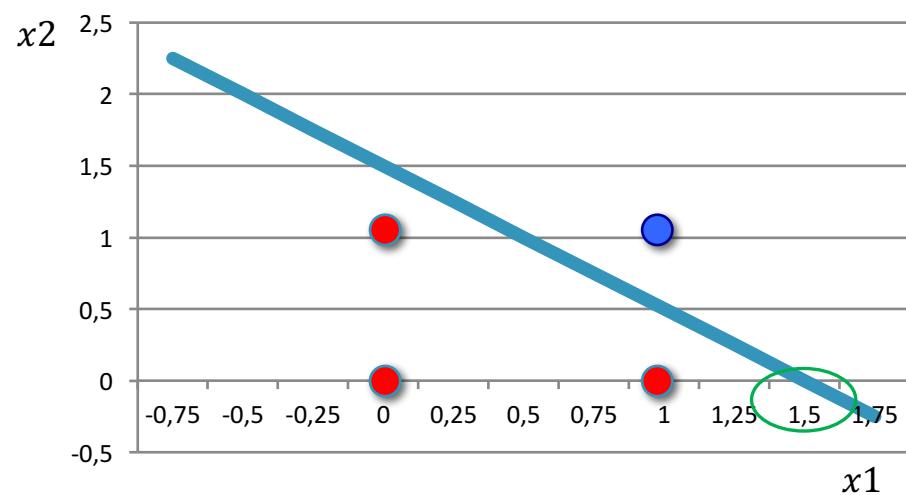
Función Sísmoide:

$$f(3) = \frac{1}{1 + e^{-\sigma \cdot 3}} = 0.998$$



2. Neurona Artificial Ejemplo AND

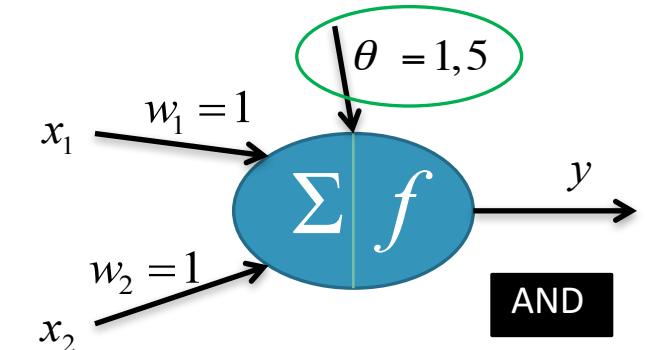
Función de activación: Paso. Neurona en estado binario.



$$y_i = f\left(\sum_{j=1}^n w_{ij}x_j - \theta_i\right)$$

Igualando a 0
se obtiene la recta (hiperplano)
que separa el espacio.

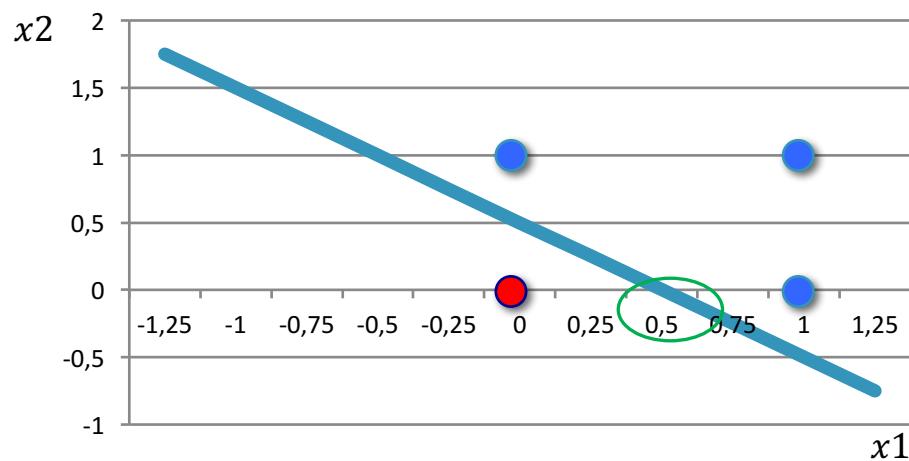
x_1	x_2	y AND	y	e	w_1	w_2	x_0	BIAS t
0	0	0	0	0	1	1	1	1,5
0	1	0	0	0				
1	0	0	0	0				
1	1	1	1	0				



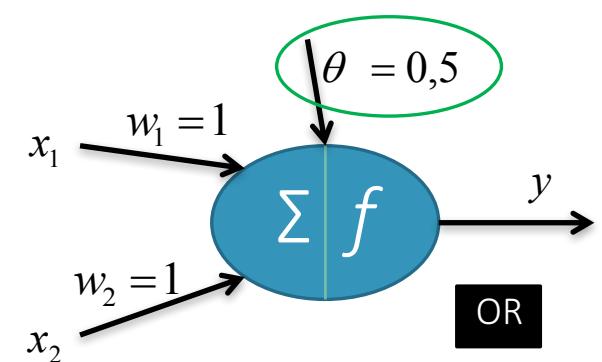
Igualando a 0 y despejando x_2 :
 $(\theta - W_1 x_1 X_1) / W_2$

2. Neurona Artificial Ejemplo OR

Función de activación: Paso. Neurona en estado binario.



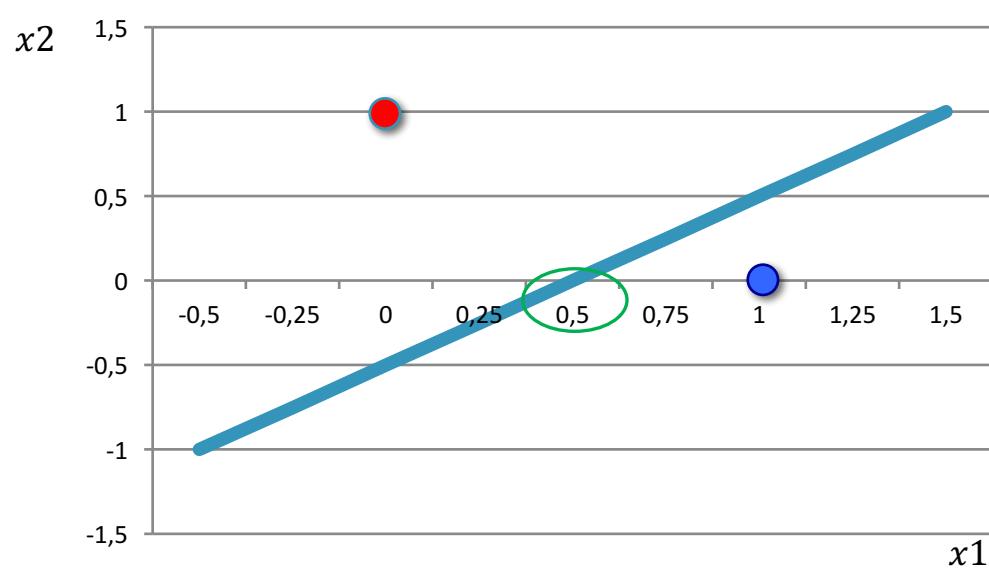
$$y_i = f\left(\sum_{j=1}^n w_{ij}x_j - \theta_i\right)$$



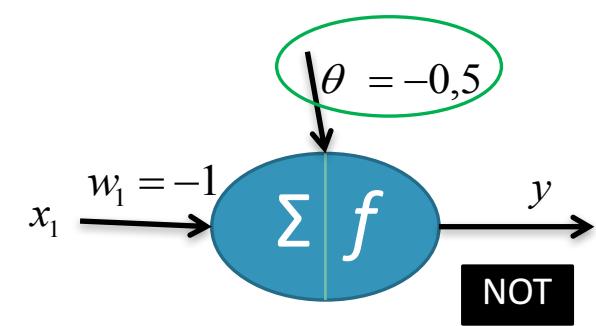
x_1	x_2	y OR	y	e	w_1	w_2	x_0	BIAS t
0	0	0	0	0			1	0,5
0	1	1	1	0				
1	0	1	1	0				
1	1	1	1	0				

2. Neurona Artificial Ejemplo NOT

Función de activación: Paso; Neurona en estado binario.



$$y_i = f\left(\sum_{j=1}^n w_{ij}x_j - \theta_i\right)$$

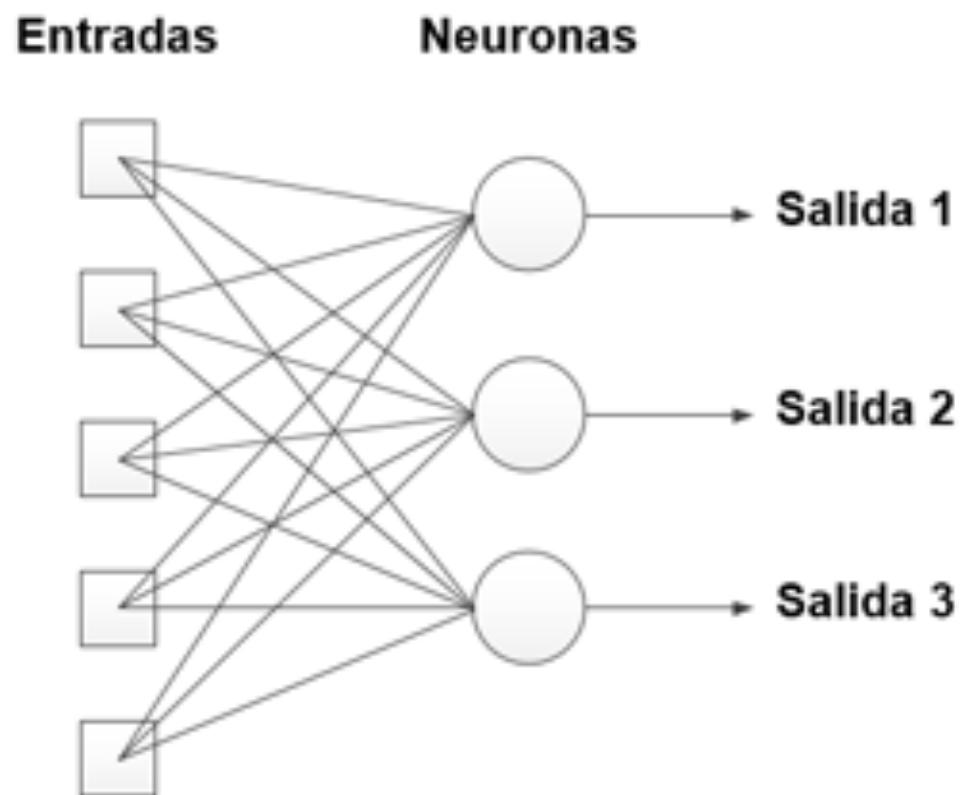


El peso W_1 negativo invierte la pendiente de la recta.

x_1	y NOT	y	e	w_1	x_0	BIAS t
0	1	0	0	-1	1	-0,5
1	0	1	0			

3. Perceptrón simple

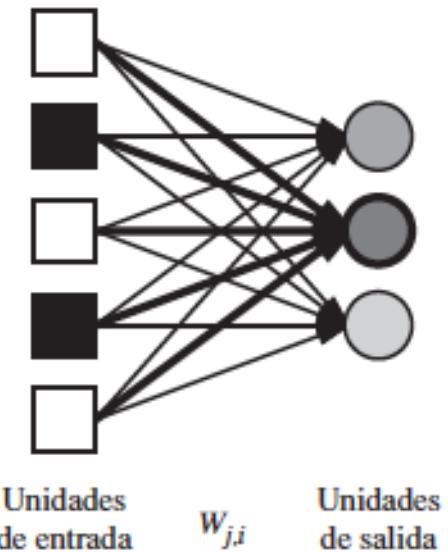
Red con alimentación hacia delante
~~Red cíclica o recurrente~~



3. Perceptrón simple

Definición

- Perceptrón o red de una sola capa: Frank Rosenblatt, 1958
 - Neuronas de entrada directamente conectada a las neuronas de salida.
 - Neurona recibe entrada de las neuronas de la capa que la precede.
- Aplicación:
 - Clasificación:
 - dicotómica (enfermo/sano), politómica (nada/leve/moderado/grave), etc.
 - Regresión.
- Aprendizaje basado en la corrección del error.

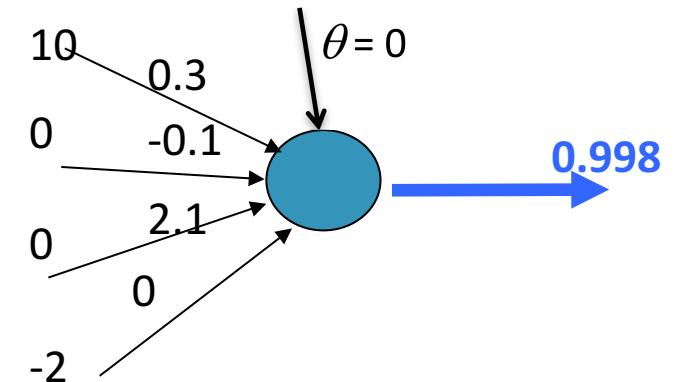
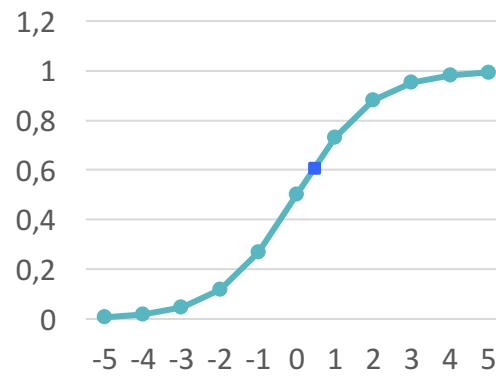


2. Neurona Artificial Ejemplo Sigmoidal

- Dada la siguiente neurona con sus entradas y pesos. Calcular el valor de activación de la neurona cuando el sesgo es 0, utilizando como función de activación la Función Sigmoidal.

Función Sigmoidal:

$$f(3) = \frac{1}{1 + e^{-\sigma \cdot 3}} = 0.998$$



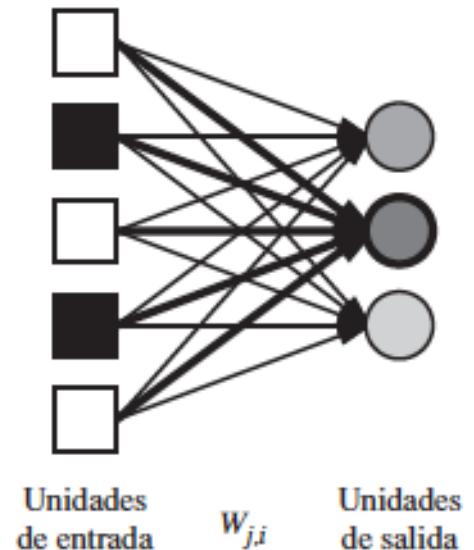
- La red neuronal devuelve la probabilidad de pertenencia a una clase. Podemos interpretarlo como una probabilidad sin más, o utilizar la probabilidad para clasificar:
 - Si es menor que un umbral decimos que no perteneces a esa clase, y si es mayor que el umbral consideramos que perteneces a esa clase.

3. Perceptrón simple

Definición

- Perceptrón o red de una sola capa: Frank Rosenblatt, 1958
 - Neuronas de entrada directamente conectada a las neuronas de salida.
 - Neurona recibe entrada de las neuronas de la capa que la precede.
- Aplicación:
 - Clasificación:
 - dicotómica (enfermo/sano), politómica (nada/leve/moderado/grave), etc.
 - Regresión.
- Aprendizaje basado en la corrección del error.

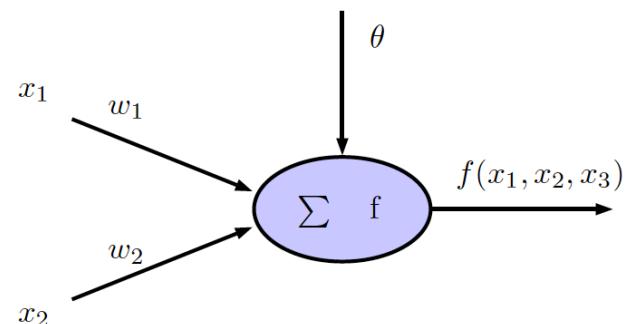
$$\Delta w_j(k) = \eta [z(k) - y(k)] x_j(k)$$



Ejercicio 7.8. Enunciado

Dados los pesos sinápticos y el umbral de un perceptrón simple con la arquitectura que se muestra mas abajo, debe determinarse la función booleana que implementa, teniendo en cuenta que la función de activación es la función signo. Los pesos sinápticos y el umbral son:

Entrada 1	2.58
Entrada 2	-2.31
Sesgo	3.18



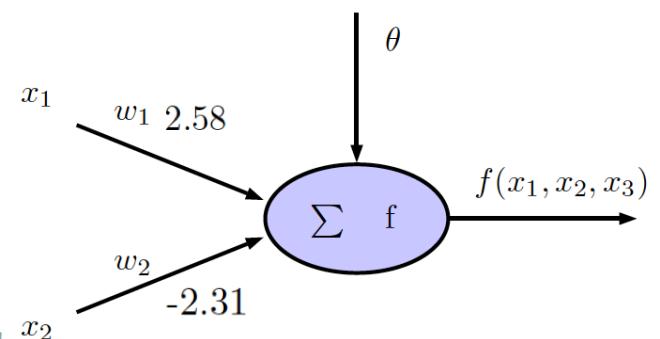
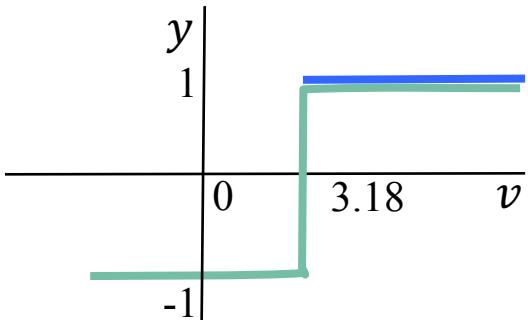
Ejercicio 7.8. Solución

Dados los pesos sinápticos y el umbral de un perceptrón simple con la arquitectura que se muestra mas abajo, debe determinarse la función booleana que implementa, teniendo en cuenta que la función de activación es la función signo. Los pesos sinápticos y el umbral son:

Entrada 1	2.58
Entrada 2	-2.31
Sesgo	3.18

g = potencial sináptico de la neurona de salida.

f = resultado de la neurona de salida.

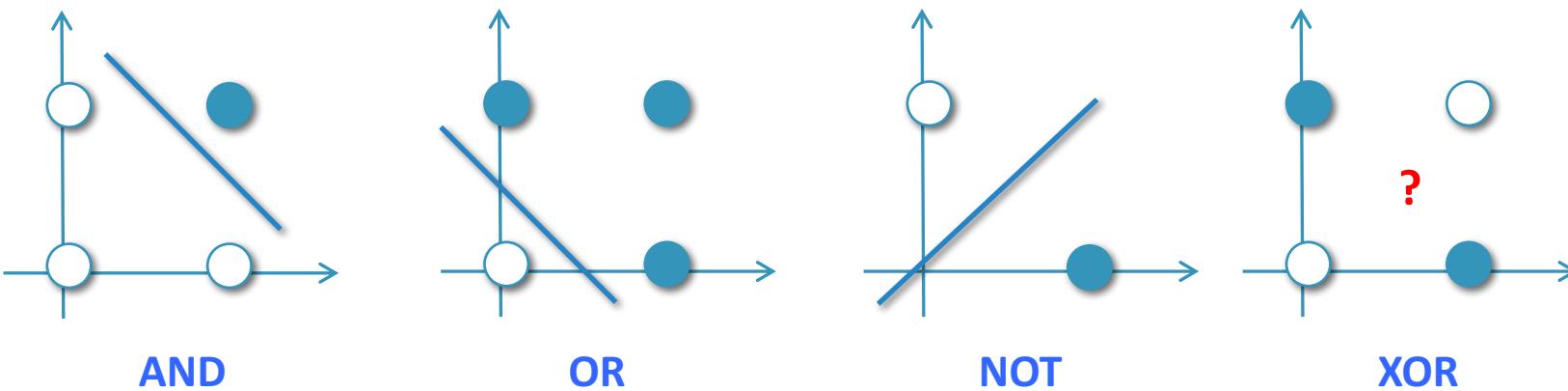


Entrada 1	Entrada 2	$g: x_1w_1 + x_2w_2 - \theta = v - \theta$	f
-1	-1	$2.58(-1) + (-2.31)(-1) - 3.18 = -0.27 - 3.18 = -3.45$	-1
-1	1	$2.58(-1) + (-2.31)(1) - 3.18 = -4.89 - 3.18 = -8.07$	-1
1	-1	$2.58(1) + (-2.31)(-1) - 3.18 = 4.89 - 3.18 = 1.71$	1
1	1	$2.58(1) + (-2.31)(1) - 3.18 = 0.27 - 3.18 = -2.91$	-1

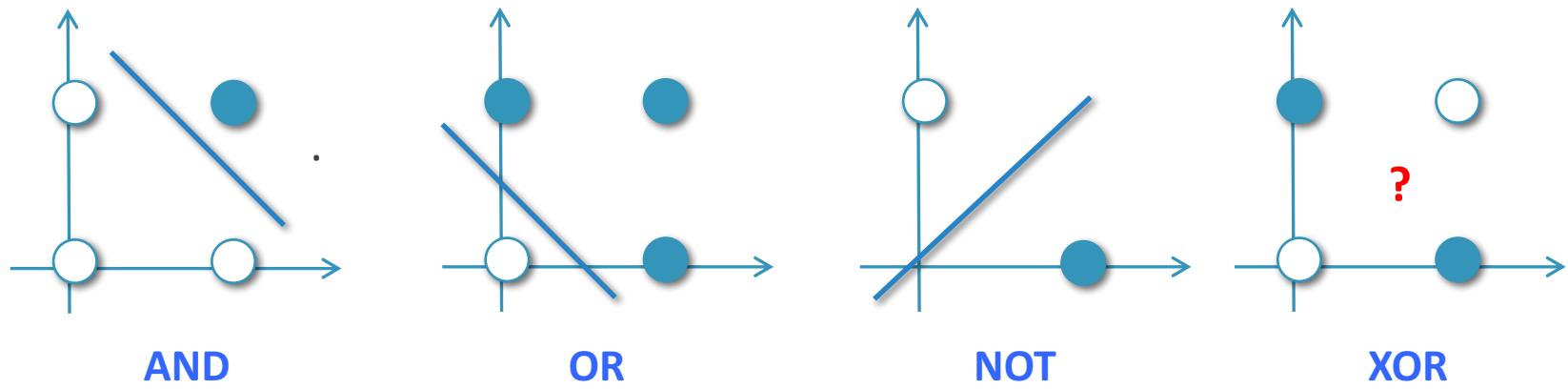
$$f(v - \theta) = \begin{cases} 1 & \text{si } v \geq \theta \\ -1 & \text{si } v < \theta \end{cases}$$

3. Perceptrón simple Inconvenientes

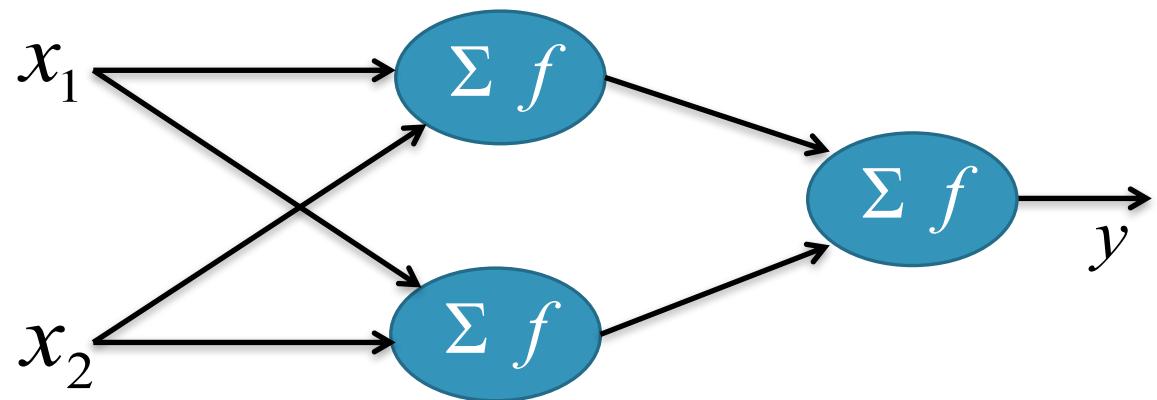
- Sólo clasifica conjuntos linealmente separables.
- No es posible Implementación XOR.
- Solución: añadir capas de neuronas.



4. Perceptrón mult capa



Entrada Capa oculta Capa salida

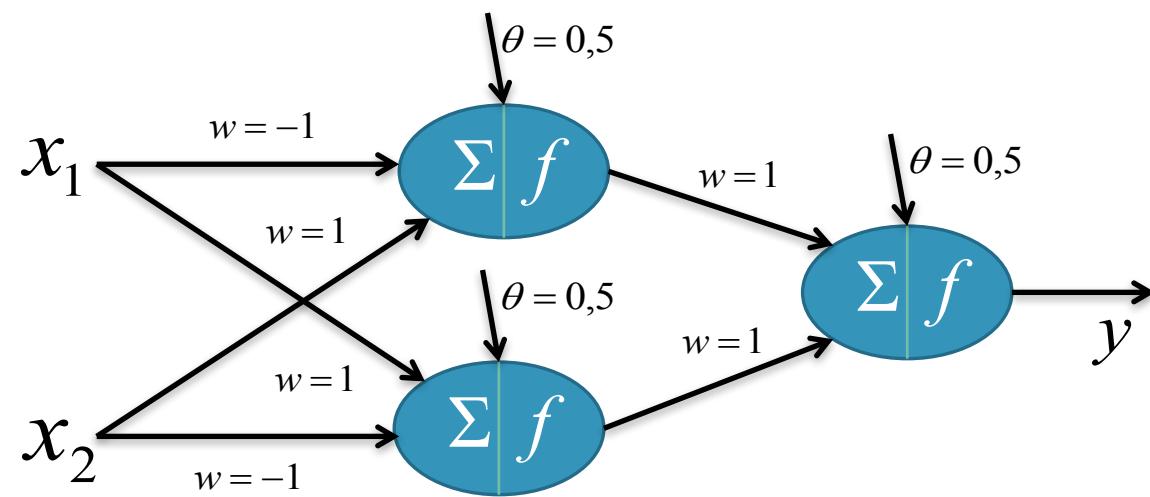


4. Perceptrón multicapa

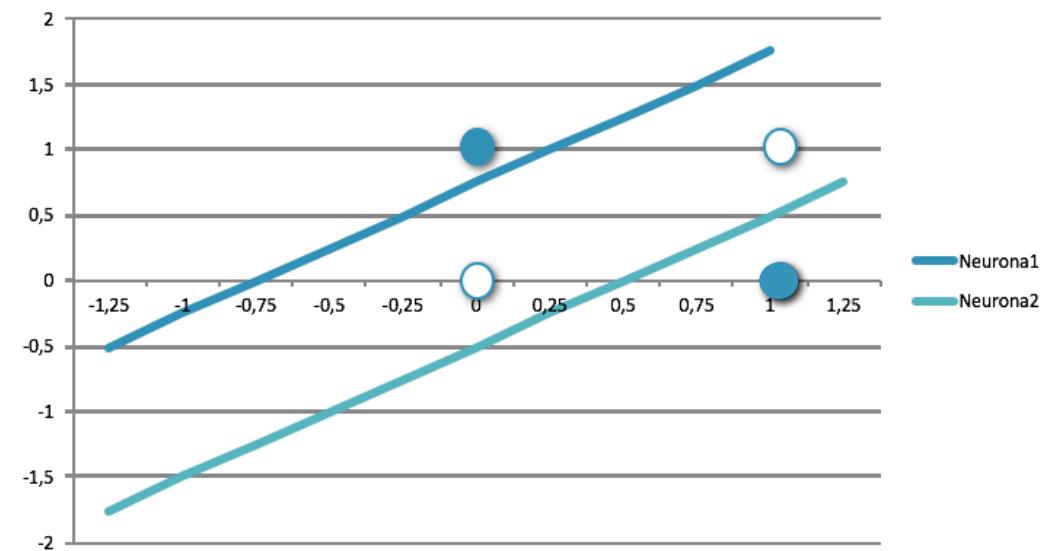
Ejemplo puerta XOR

- Implementación XOR

Perceptrons: Minsky y Papert, 1969



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

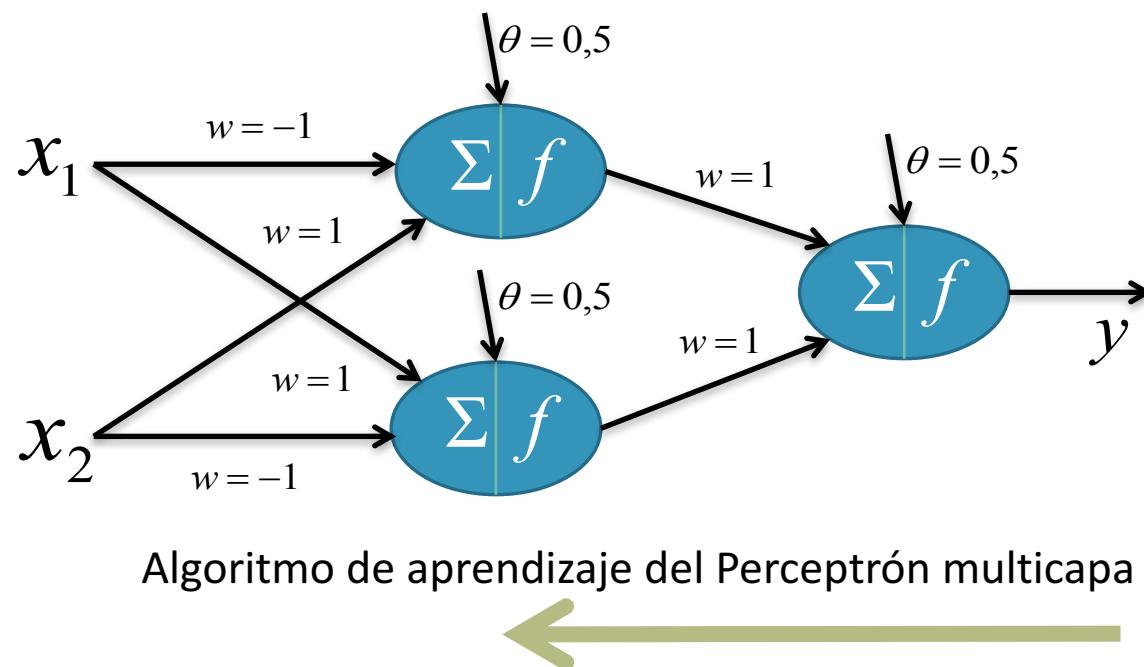


4. Perceptrón multicapa

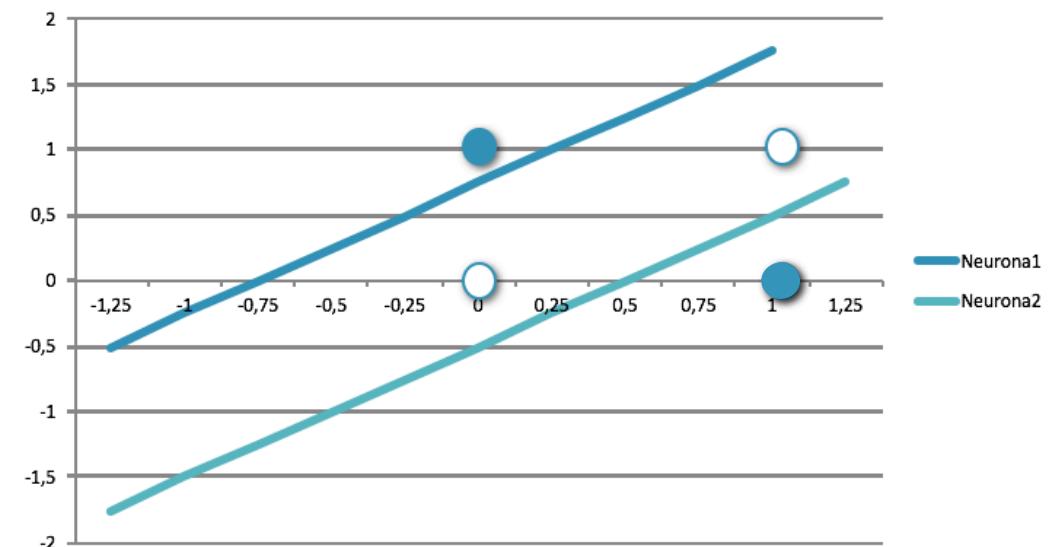
Ejemplo puerta XOR

- Implementación XOR

Parallel distributed processing, 1986



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

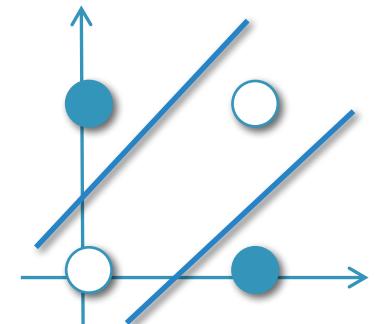
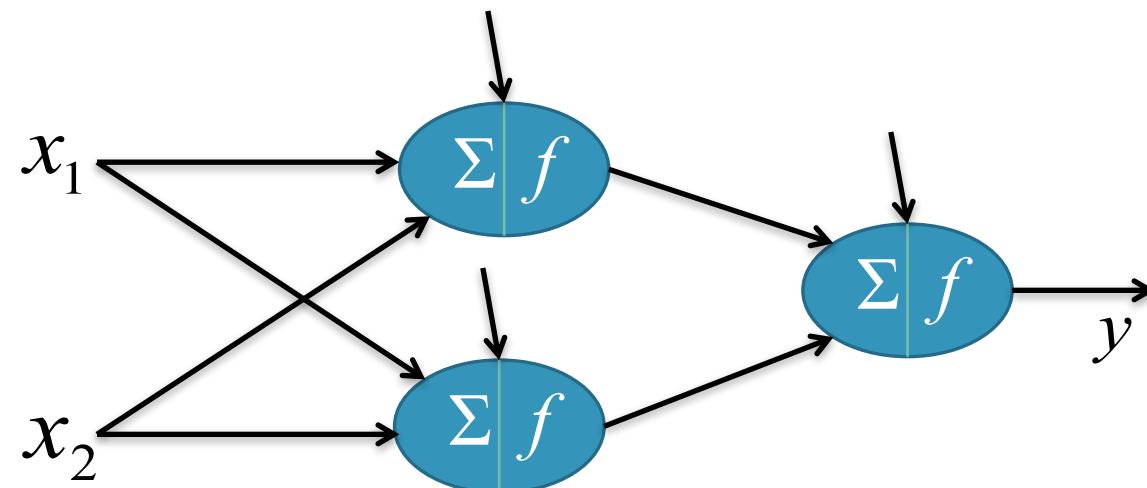


4. Perceptrón multicapa

Ejemplo puerta XOR

- Perceptrón aprende a dividir el espacio:
 - Zona donde se activa.
 - Zona donde NO se activa.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

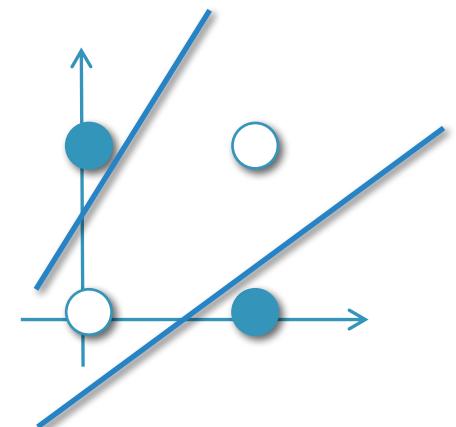
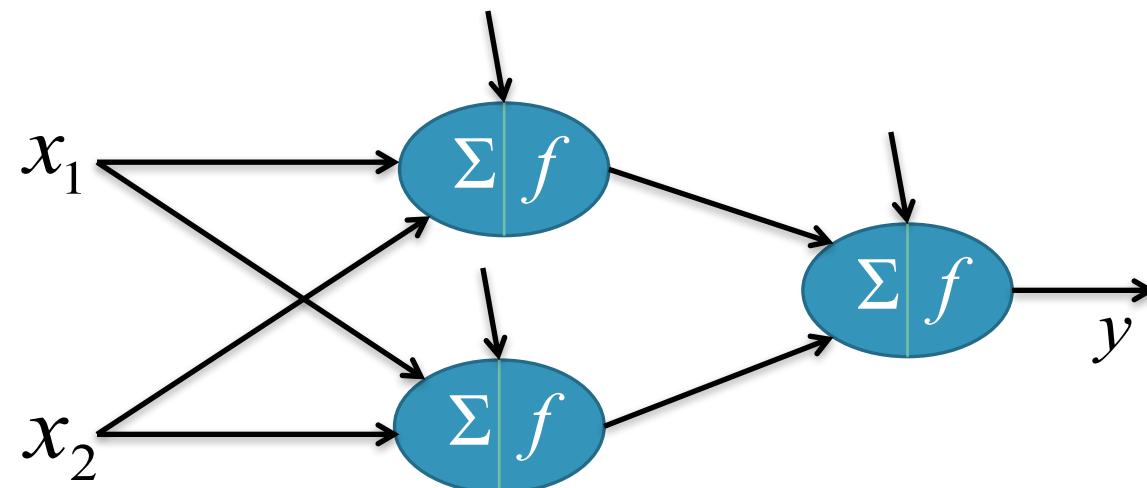


4. Perceptrón multicapa

Ejemplo puerta XOR

- Perceptrón aprende a dividir el espacio:
 - Zona donde se activa.
 - Zona donde NO se activa.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

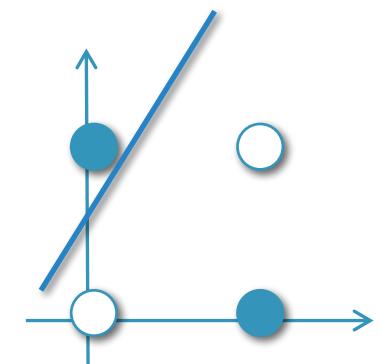
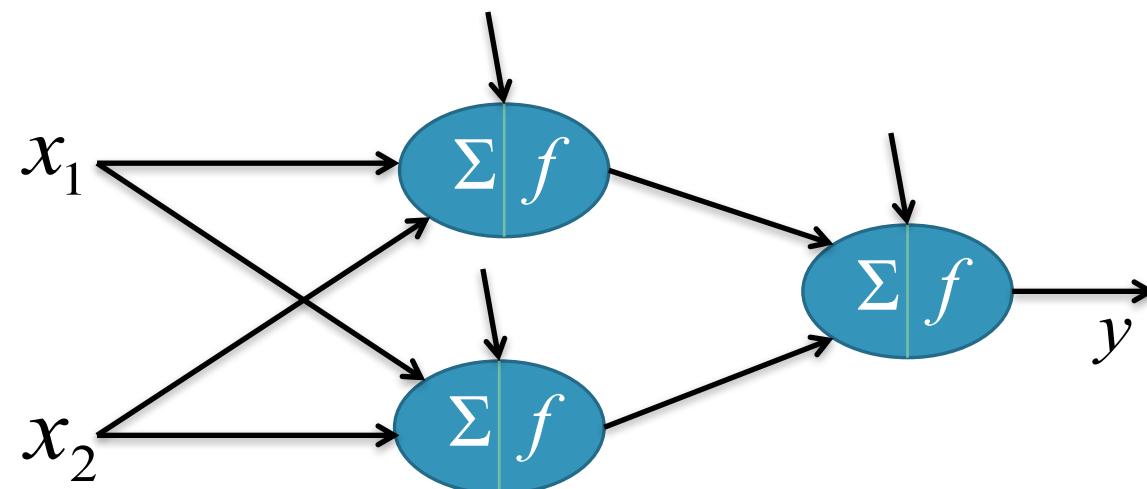


4. Perceptrón multicapa

Ejemplo puerta XOR

- Perceptrón aprende a dividir el espacio:
 - Zona donde se activa.
 - Zona donde NO se activa.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

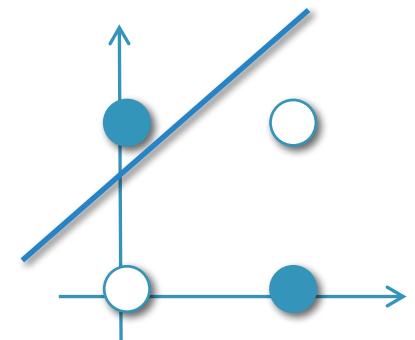
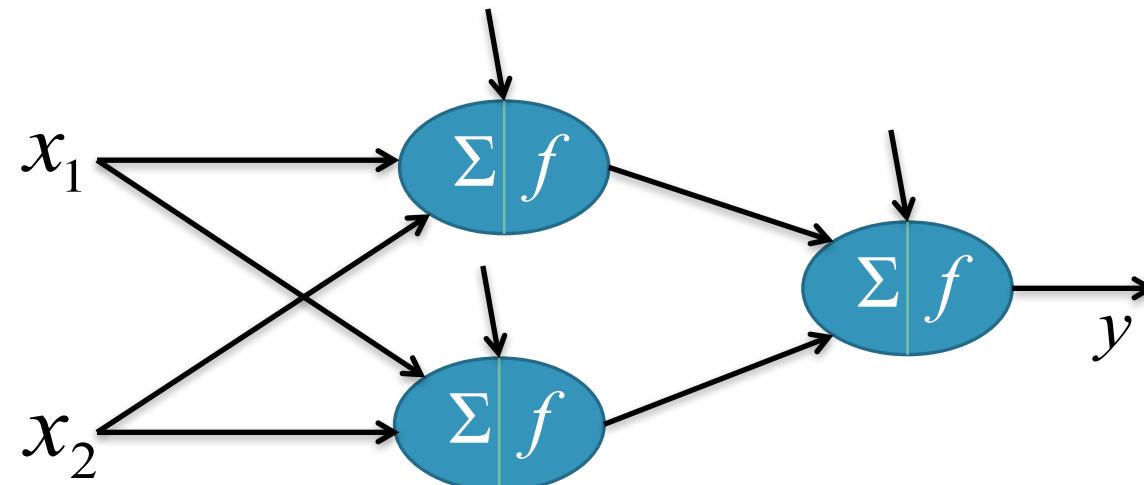


4. Perceptrón multicapa

Ejemplo puerta XOR

- Perceptrón aprende a dividir el espacio:
 - Zona donde se activa.
 - Zona donde NO se activa.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

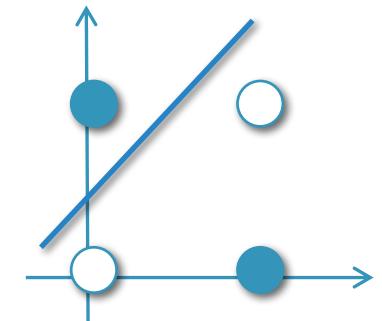
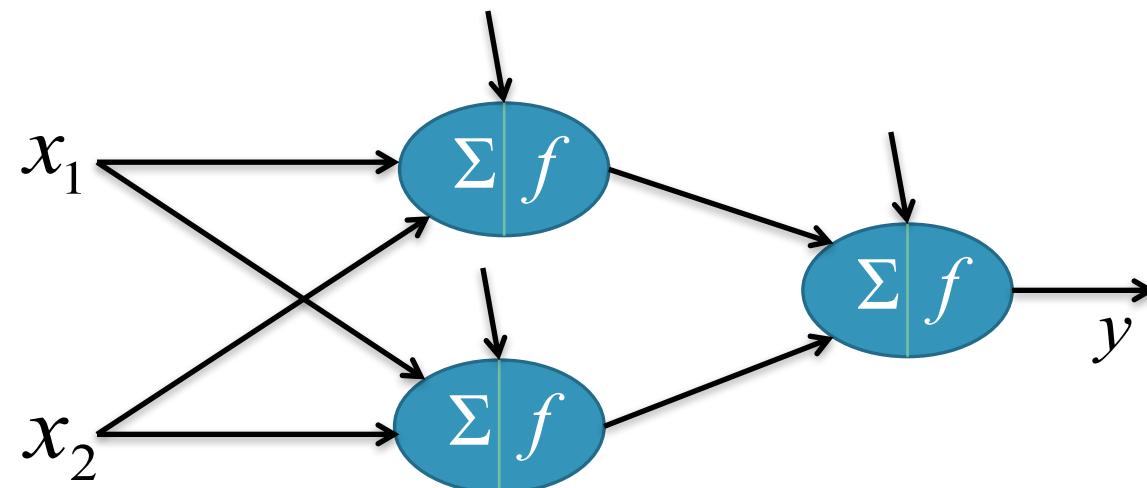


4. Perceptrón multicapa

Ejemplo puerta XOR

- Perceptrón aprende a dividir el espacio:
 - Zona donde se activa.
 - Zona donde NO se activa.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

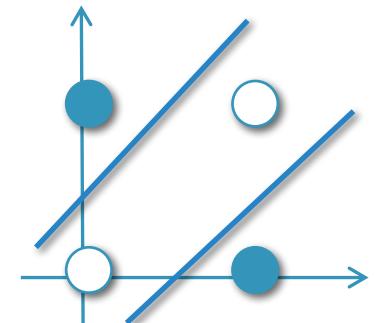
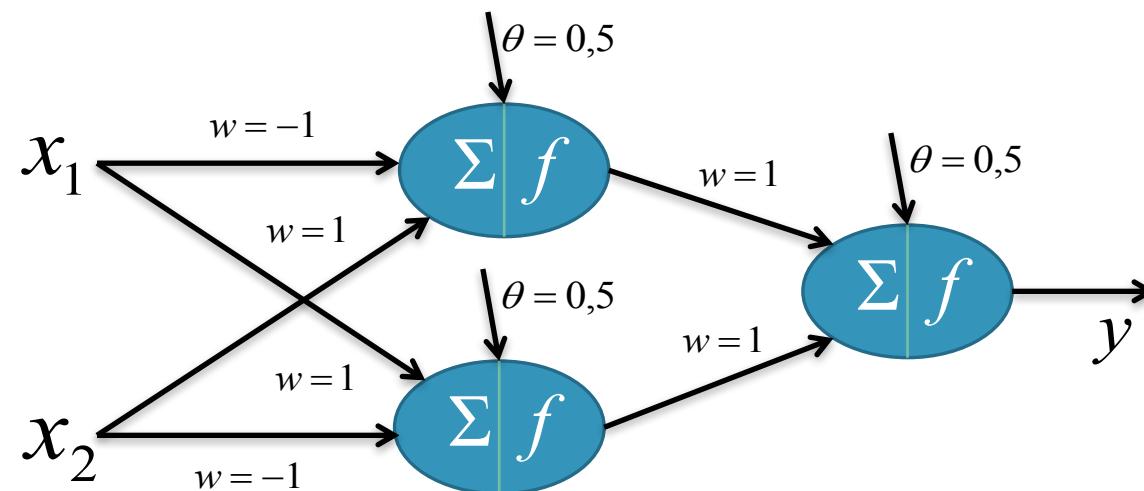


4. Perceptrón multicapa

Ejemplo puerta XOR

- Perceptrón aprende a dividir el espacio:
 - Zona donde se activa.
 - Zona donde NO se activa.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

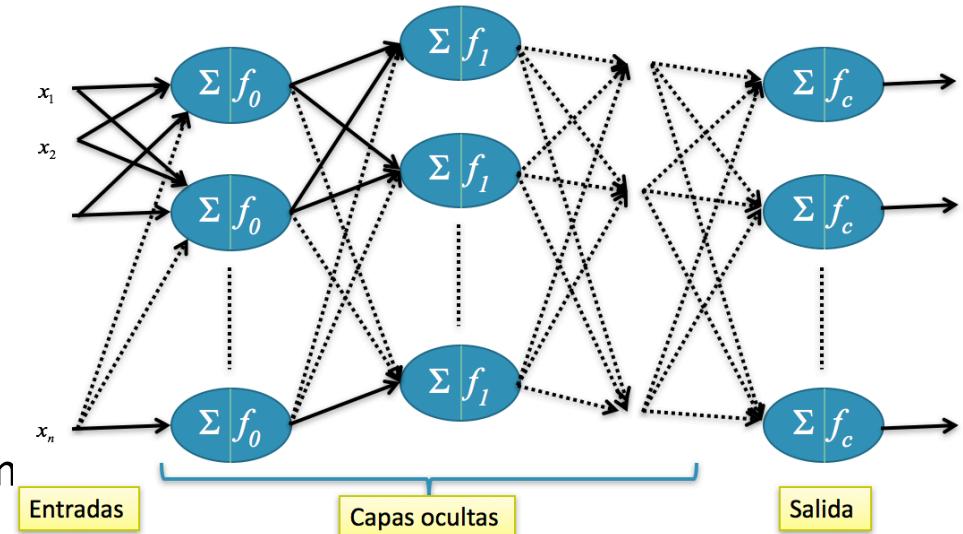


4. Perceptrón multicapa

Topología

- Modelo del Aprendizaje Computacional (Machine Learning): Red neuronal Artificial
 - Más conocido: Perceptrón multicapa + Retro propagación de Errores (método de cálculo: descenso del gradiente).
- Mapear¹ conjunto de **datos de entrada** en un conjunto de **datos de salida** deseado.

edad	tam	grado	gang	feno	quim	horm	recid
51	2.3	G1	0	Luminal A	No	Yes	Leve
46	3.4	G3	4	Basal like	Yes	No	Grave
49	2.1	G2	1	HER2 enriched	Yes	Yes	Moderado
31	1.7	G2	1	Basal like	Yes	No	Leve
36	1.5	G3	0	Basal like	Yes	No	Grave
59	3.8	G3	2	Luminal B	Yes	Yes	No



- Arquitectura de red: entrada-capas ocultas-capa salida
- Las conexiones sinápticas modifican sus valores (W) para m
- Funciones de activación:
 - capa salida: función logística.
 - capa oculta: tangente hiperbólica.

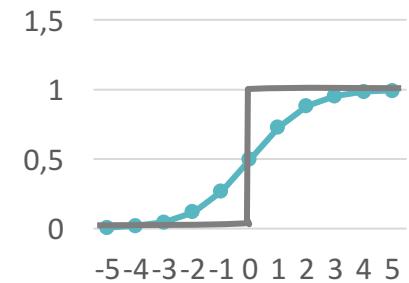
¹Mapear: localizar y representar gráficamente la distribución relativa de las partes de un todo.

4. Perceptrón multicapa

Regla de retropropagación

- Función de activación debe ser diferenciable¹. Utilizamos la sigmoidal porque se deriva fácil:

$$f(x) = \frac{1}{1 + e^{-x}}; \quad \frac{df(x)}{dx} = f(x) * (1 - f(x))$$

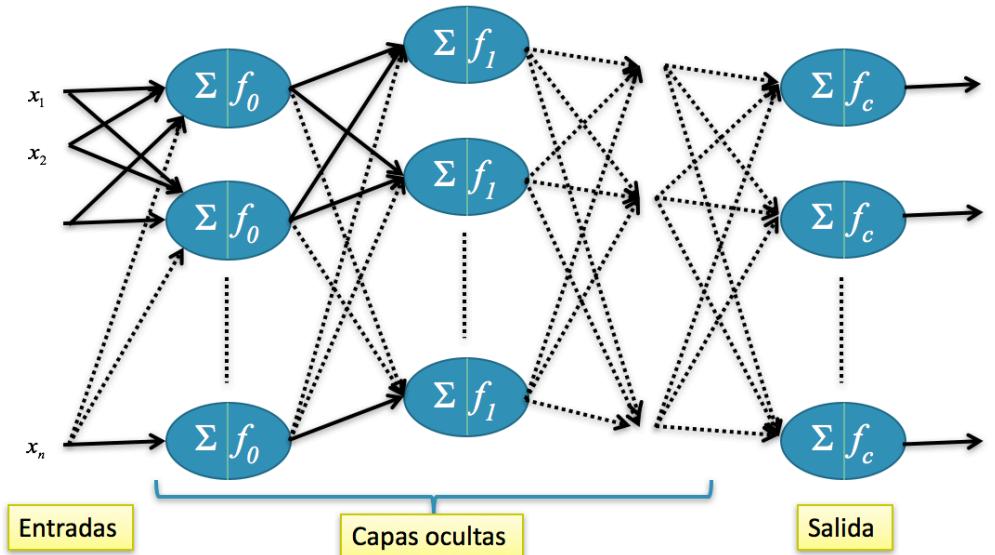
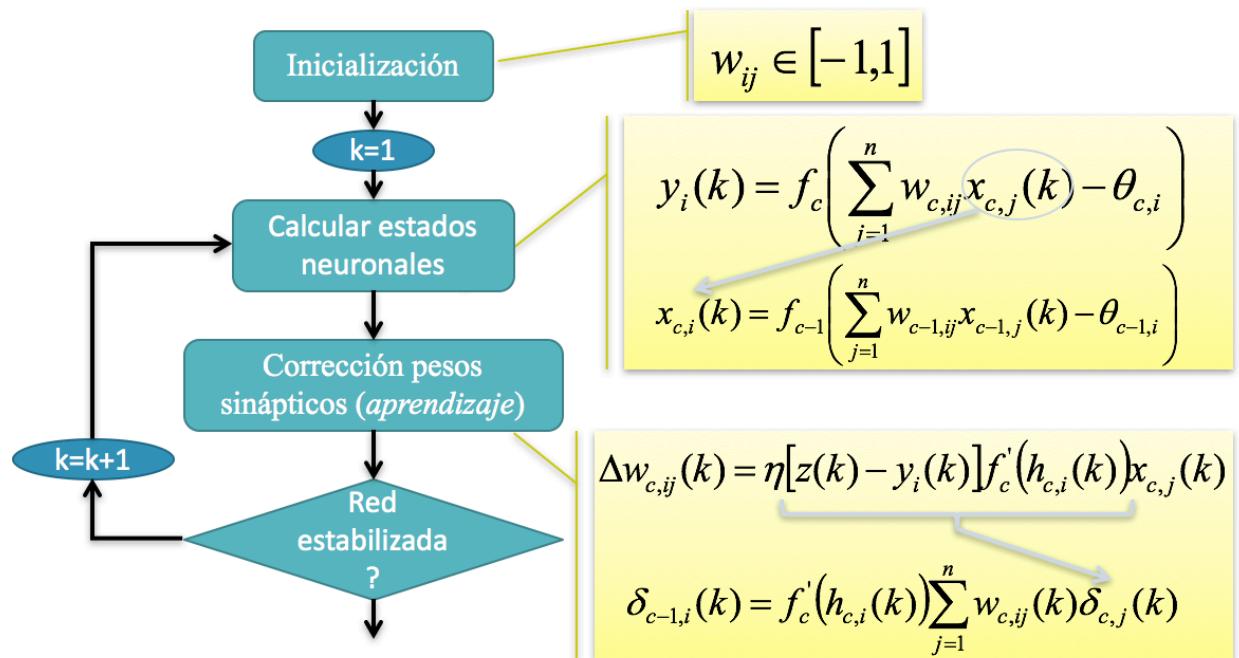


- Inicialmente los valores son calculados hacia delante:
 - Pesos sinápticos
 - Salidas de neuronas de las capas oculta.
 - Salidas de las neuronas de la capa de salida.
- Posteriormente los errores son calculados desde capa salida hasta el inicio:
 - Con la retropropagación del error se reajustan los pesos sinápticos.

¹ f es diferenciable en \mathbb{R} por ser una función con derivadas parciales continuas en \mathbb{R} (condición suficiente para la diferenciabilidad).

4. Perceptrón multicapa

Regla de retropropagación



4. Perceptrón multicapa

Regla de retropropagación

- El error es calculado: $e_k = d_k - y_k$
- Gradiente para las salidas:

$$\delta_k = \frac{dy_k}{dx_k} * e_k = y_k * (1 - y_k) * e_k$$

- Y para cada nodo j en las capas ocultas:

$$\delta_j = y_j * (1 - y_j) * \sum_{k=1}^n w_{ik} \delta_k$$

- Cada peso en la red es actualizado acorde la siguiente fórmula:

$$w_{ij} = w_{ij} + \alpha * x_i * \delta_j$$

$$w_{jk} = w_{jk} + \alpha * y_j * \delta_k$$

4. Perceptrón multicapa

Regla de retropropagación

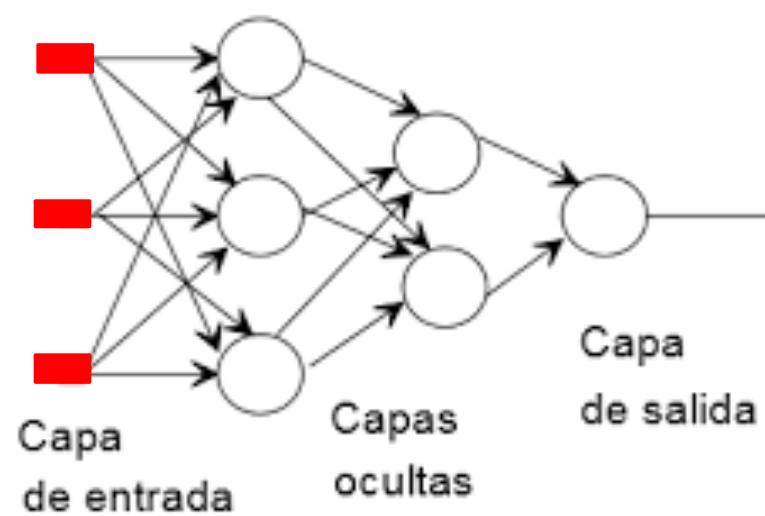
```
function BACK-PROP-LEARNING(examples, network) returns a neural network
    inputs: examples, a set of examples, each with input vector x and output vector y
            network, a multilayer network with  $L$  layers, weights  $w_{i,j}$ , activation function  $g$ 
    local variables:  $\Delta$ , a vector of errors, indexed by network node

repeat
    for each weight  $w_{i,j}$  in network do
         $w_{i,j} \leftarrow$  a small random number
    for each example  $(\mathbf{x}, \mathbf{y})$  in examples do
        /* Propagate the inputs forward to compute the outputs */
        for each node  $i$  in the input layer do
             $a_i \leftarrow x_i$ 
        for  $\ell = 2$  to  $L$  do
            for each node  $j$  in layer  $\ell$  do
                 $in_j \leftarrow \sum_i w_{i,j} a_i$ 
                 $a_j \leftarrow g(in_j)$ 
        /* Propagate deltas backward from output layer to input layer */
        for each node  $j$  in the output layer do
             $\Delta[j] \leftarrow g'(in_j) \times (y_j - a_j)$ 
        for  $\ell = L - 1$  to  $1$  do
            for each node  $i$  in layer  $\ell$  do
                 $\Delta[i] \leftarrow g'(in_i) \sum_j w_{i,j} \Delta[j]$ 
        /* Update every weight in network using deltas */
        for each weight  $w_{i,j}$  in network do
             $w_{i,j} \leftarrow w_{i,j} + \alpha \times a_i \times \Delta[j]$ 
    until some stopping criterion is satisfied
return network
```

Figure 18.23 The back-propagation algorithm for learning in multilayer networks.

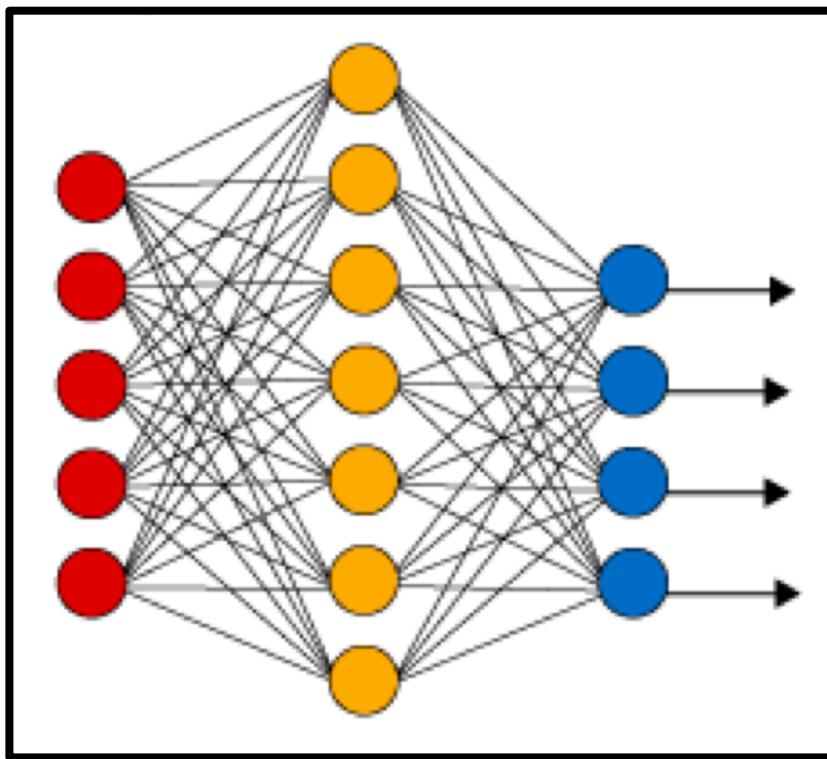
4. Perceptrón multicapa

Aplicabilidad. Clasificación dicotómica



edad	tam	grado	gang	feno	quim	horm	recid
51	2.3	G1	0	Luminal A	No	Yes	NO
46	3.4	G3	4	Basal like	Yes	No	SI
49	2.1	G2	1	HER2 enriched	Yes	Yes	NO
31	1.7	G2	1	Basal like	Yes	No	NO
36	1.5	G3	0	Basal like	Yes	No	NO
59	3.8	G3	2	Luminal B	Yes	Yes	NO

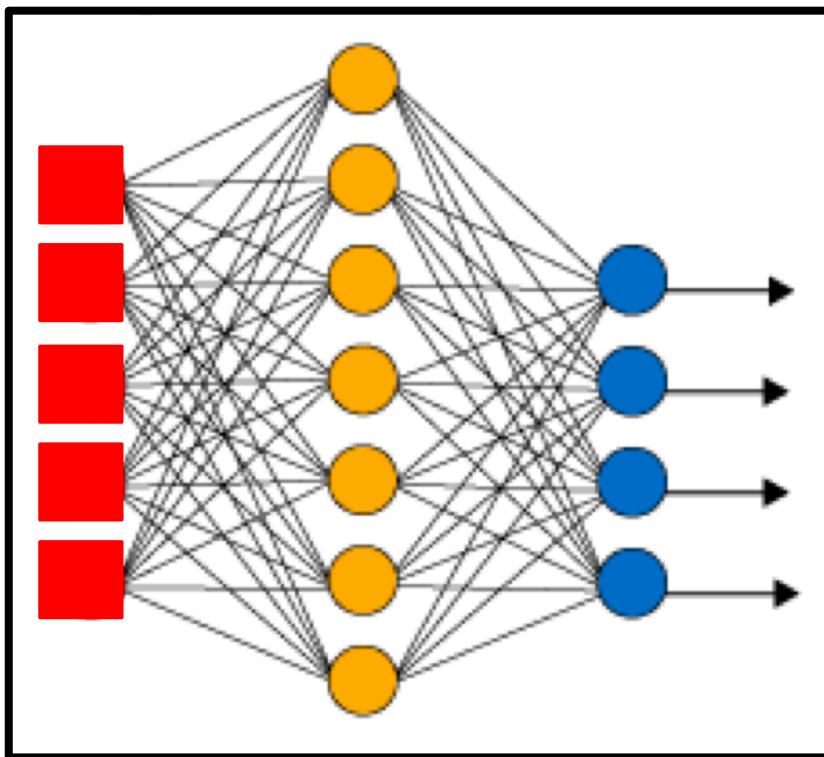
4. Perceptrón multicapa Aplicabilidad. Clasificación polítómica



edad	tam	grado	gang	feno	quim	horm	recid
51	2.3	G1	0	Luminal A	No	Yes	Leve
46	3.4	G3	4	Basal like	Yes	No	Grave
49	2.1	G2	1	HER2 enriched	Yes	Yes	Moderado
31	1.7	G2	1	Basal like	Yes	No	Leve
36	1.5	G3	0	Basal like	Yes	No	Grave
59	3.8	G3	2	Luminal B	Yes	Yes	No

4. Perceptrón multicapa

Aplicabilidad

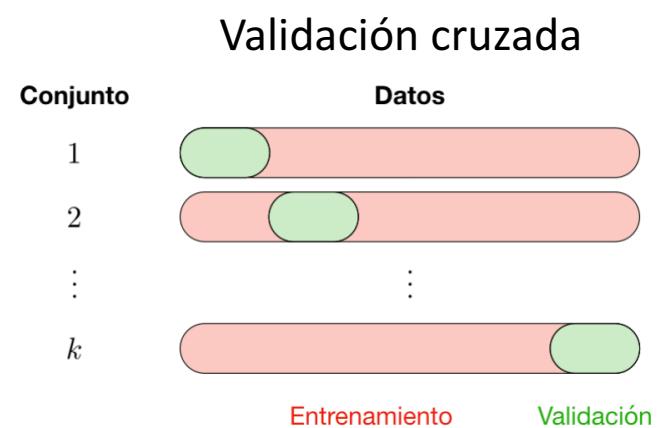
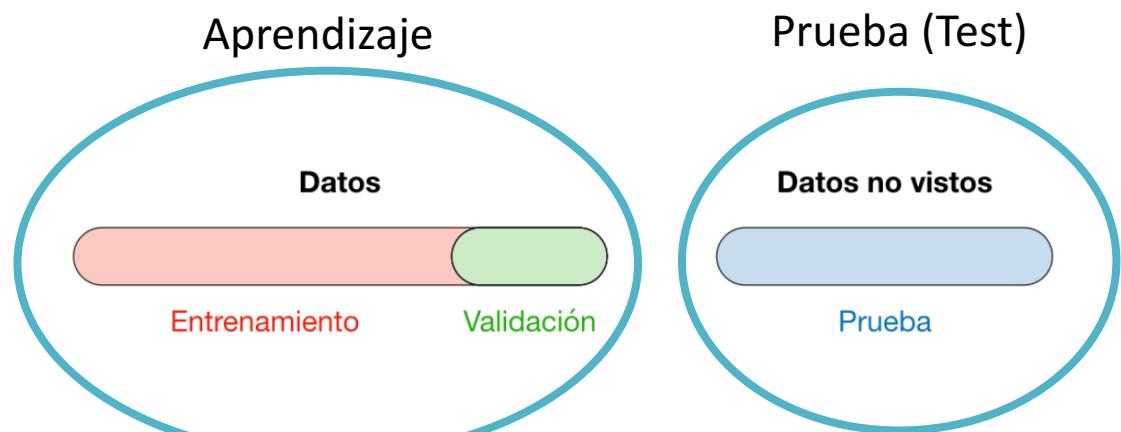


edad	tam	grado	gang	feno	quim	horm	recid
51	2.3	G1	0	Luminal A	No	Yes	Leve
46	3.4	G3	4	Basal like	Yes	No	Grave
49	2.1	G2	1	HER2 enriched	Yes	Yes	Moderado
31	1.7	G2	1	Basal like	Yes	No	Leve
36	1.5	G3	0	Basal like	Yes	No	Grave
59	3.8	G3	2	Luminal B	Yes	Yes	No

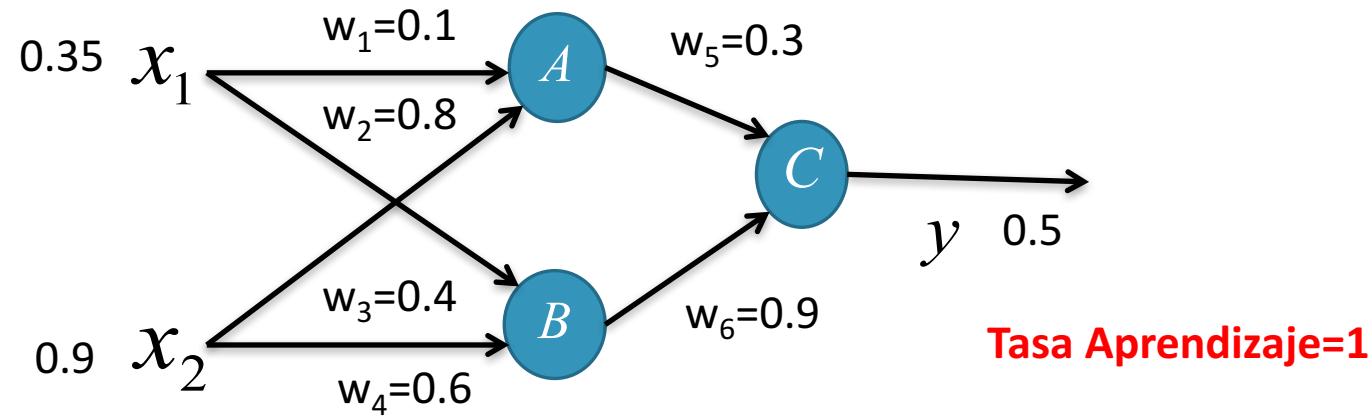
4. Perceptrón multicapa

Validación Cruzada

- Aprendizaje: entrenamiento y validación.
- Prueba (test): evaluar el modelo.
- Validación:
 - Validación Cruzada (Cross Validation) con K cajas (K-folds).
 - Evitar sobreajuste (overfitting) del modelo



❖ Ejercicio: perceptrón multicapa y retro-propagación. La función de activación es sigmoide.



Entrada Neurona A = $(0.35 \times 0.1) + (0.9 \times 0.8) = 0.755$.

Salida = $1/(1+e^{-0.755}) = 0.68$.

Entrada Neurona B = $(0.9 \times 0.6) + (0.35 \times 0.4) = 0.68$.

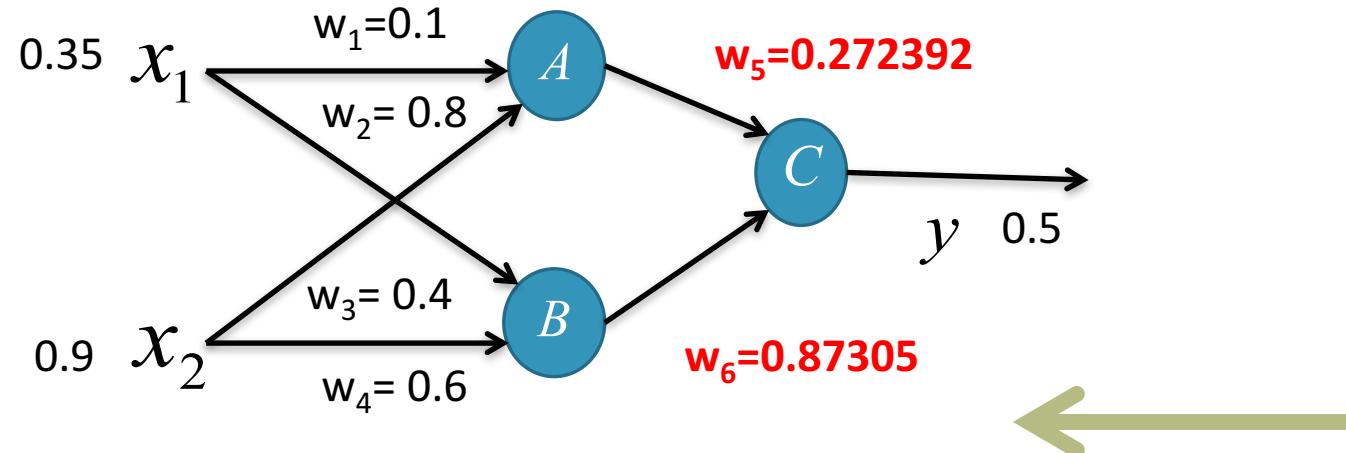
Salida = $1/(1+e^{-0.68}) = 0.6637$.

Entrada Neurona C = $(0.3 \times 0.68) + (0.9 \times 0.6637) = 0.80133$.

Salida = $1/(1+e^{-0.80133}) = 0.69$.

Error salida $0.5 - 0.69 = -0.19$

❖ Ejercicio: perceptrón multicapa y retro-propagación. La función de activación es sigmoide.



Gradiente Salida $\delta = \text{out} (1 - \text{out}) (t - \text{out}) = 0.69 (1-0.69) (0.5- 0.69) = -0.0406$.

$$\delta_k = \frac{dy_k}{dx_k} * e_k = y_k * (1 - y_k) * e_k$$

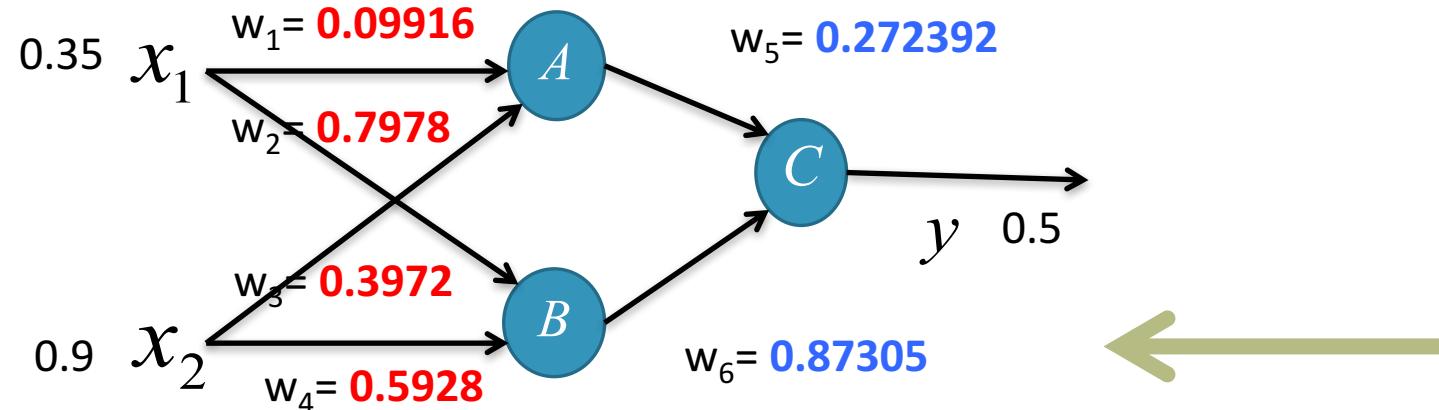
Nuevos pesos para la capa salida

$$w_5 = w_5 + (\delta \times \text{input}) = 0.3 + (-0.0406 \times 0.68) = 0.272392.$$

$$w_6 = w_6 + (\delta \times \text{input}) = 0.9 + (-0.0406 \times 0.6637) = 0.87305.$$

$$w_{jk} = w_{jk} + \alpha * y_j * \delta_k$$

❖ Ejercicio: perceptrón multicapa y retro-propagación. La función de activación es sigmoide.



Gradientes capas ocultas:

$$\delta_1 = \text{out} \times (1-\text{out}) \times \delta \times w_1 = 0.68 \times 0.32 \times -0.0406 \times 0.272392 = -0.0024$$

$$\delta_2 = \text{out} \times (1-\text{out}) \times \delta \times w_2 = 0.6637 \times 0.3363 \times -0.0406 \times 0.87305 = -0.0079$$

Nuevos pesos:

$$w_1 = 0.1 + (0.35 \times -0.0024) = 0.09916.$$

$$w_2 = 0.8 + (0.9 \times -0.0024) = 0.7978.$$

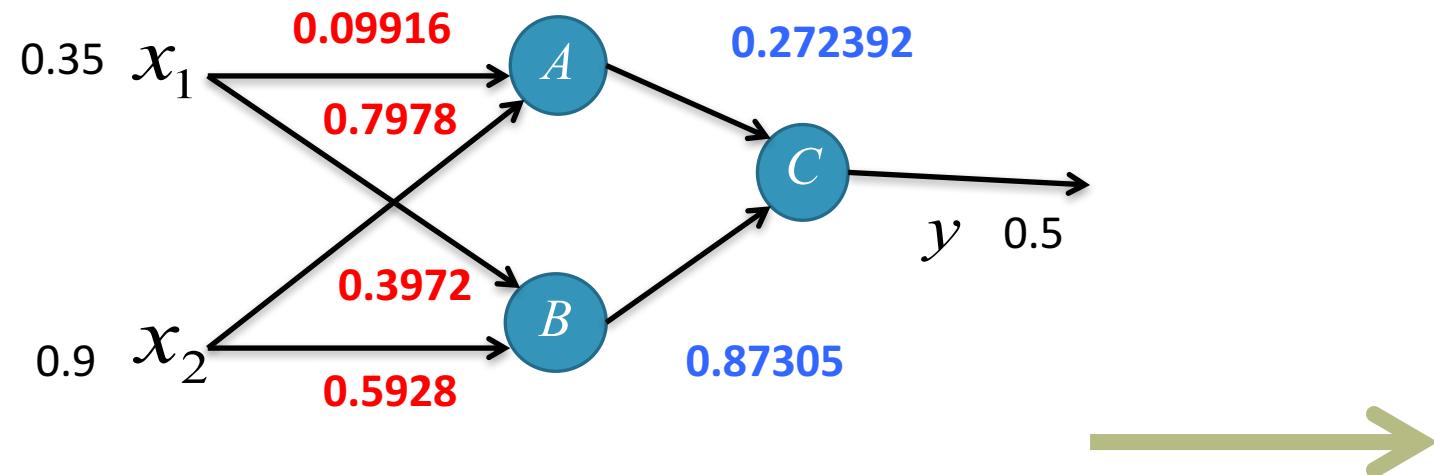
$$w_3 = 0.4 + (0.35 \times -0.0079) = 0.3972.$$

$$w_4 = 0.6 + (0.9 \times -0.0079) = 0.5928.$$

$$\delta_j = y_j * (1 - y_j) * \sum_{k=1}^n w_{ik} \delta_k$$

$$w_{ij} = w_{ij} + \alpha * x_i * \delta_j$$

❖ Ejercicio: perceptrón multicapa y retro-propagación. La función de activación es sigmoide.



$$\text{Entrada Neurona A} = (0.35 \times \textcolor{red}{0.09916}) + (0.9 \times \textcolor{red}{0.7978}) = 0,75275.$$

$$\text{Salida} = 1/(1+e^{-0,75275}) = \textcolor{green}{0,67977}.$$

$$\text{Entrada Neurona B} = (0.9 \times \textcolor{red}{0.5928}) + (0.35 \times \textcolor{red}{0.3972}) = 0,67260.$$

$$\text{Salida} = 1/(1+e^{-0,67260}) = \textcolor{green}{0,66208}.$$

$$\text{Entrada Neurona C} = (0.3 \times \textcolor{green}{0,67977}) + (0.9 \times \textcolor{green}{0,66208}) = 0,76312.$$

$$\text{Salida} = 1/(1+e^{-0,76312}) = \textcolor{blue}{0,682}.$$

$$\text{Error salida } 0.5 - 0.68 = -0.18 \text{ (Antes 0.69)}$$

Ejercicios propuestos

👉 Relación de ejercicios con las soluciones en el campus virtual:

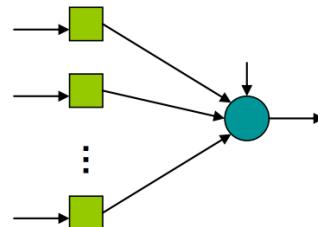


UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y Ciencias de la Computación

SISTEMAS INTELIGENTES.

RELACIÓN DE PROBLEMAS DE
REDES NEURONALES

Ejercicio 1: Dados los pesos sinápticos y los sesgos de un perceptrón simple con la arquitectura mostrada mas abajo, calcula la función booleana que implementa.



4. Conclusiones

- Las **redes neuronales artificiales** están inspiradas en el funcionamiento de las **neuronas biológicas**, pero no pretenden simularlas sino ser útiles para la **resolución de problemas** de ingeniería.
- Pueden representar funciones **no lineales** complejas mediante una red de unidades sencillas.
- El **perceptrón multicapa** puede resolver problemas insolubles para el perceptrón simple.
- El algoritmo de **retro-propagación** busca valores de los pesos sinápticos que minimicen el error.
- El **perceptrón multicapa con retro-propagación** se usa para problemas reales de **regresión** (número de neuronas de salida debe coincidir con el número de salidas de la función, ej: colores en una imagen (rojo, azul, amarillo) o **clasificación** (binaria: 1 neurona, multiclase: 1 neurona por categoría).