Redes Neuronales Artificiales

Inspiración biológica

"Entender el cerebro y emular su potencia"

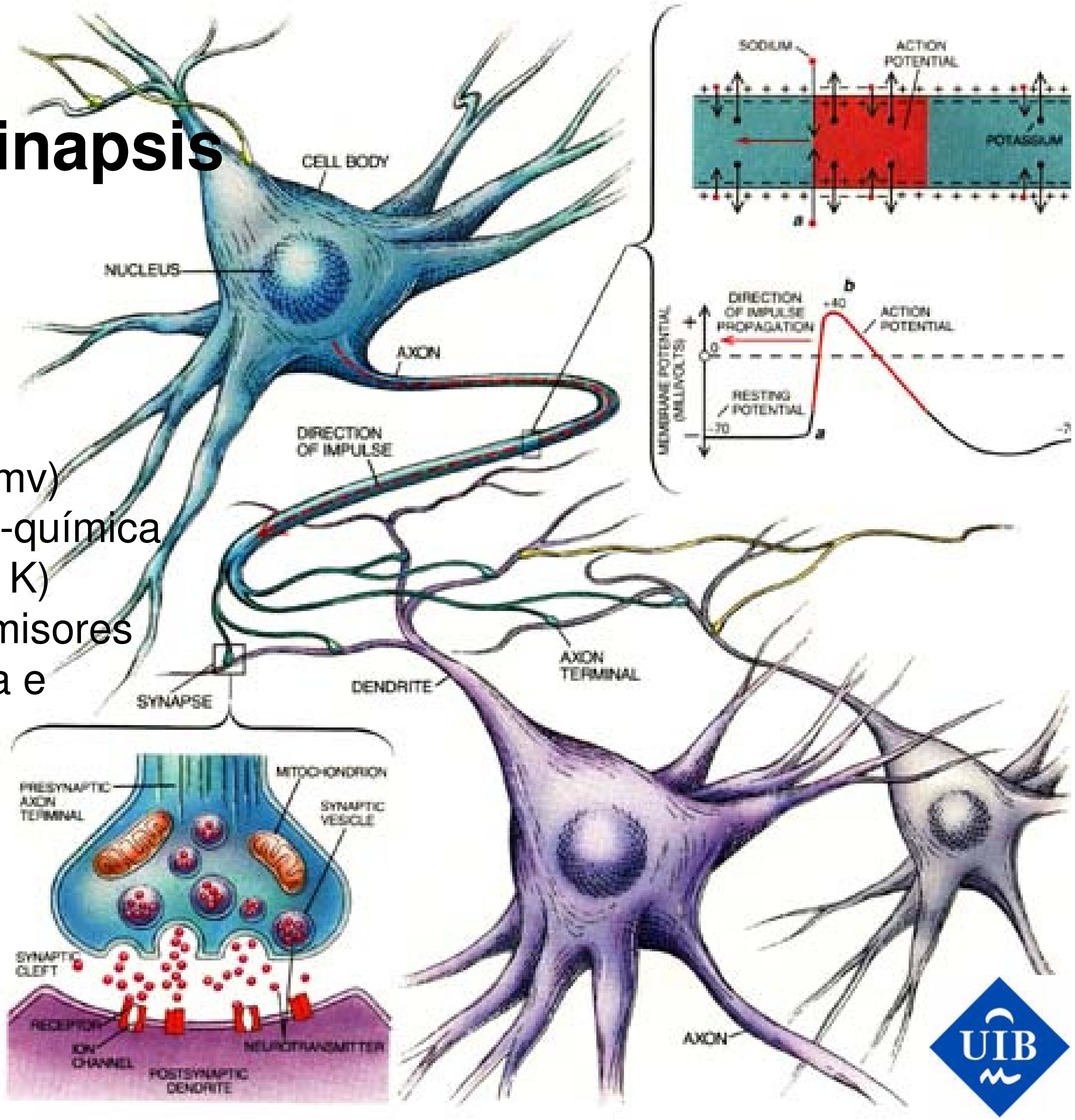
Cerebro

- Gran velocidad de proceso
- Tratamiento de grandes cantidades de información procedente de
  - Los sentidos
  - Memoria almacenada
- Capacidad de tratar situaciones nuevas
- Capacidad de aprendizaje

Redes Neuronales Artificiales

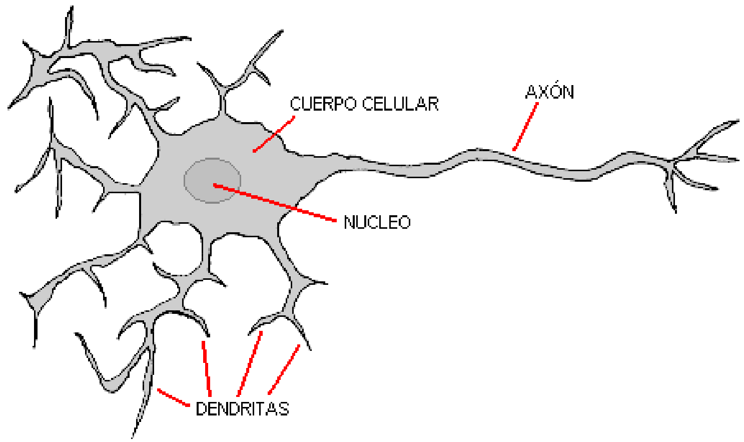
- Basado en el modelo computacional del aprendizaje humano que se realiza a través de las neuronas
- Se comportan de forma parecida a nuestro cerebro aprendiendo de la experiencia y del pasado, y aplicando tal conocimiento a la resolución de problemas nuevos (entrenamiento)

Neuronas y sinapsis



- Señal eléctrica (-70mv)
- Propagación electro-química (concentración Na y K)
- Sinapsis, neurotrasmisores químicos (excitadora e inhibidora)

La neurona biológica "básica"

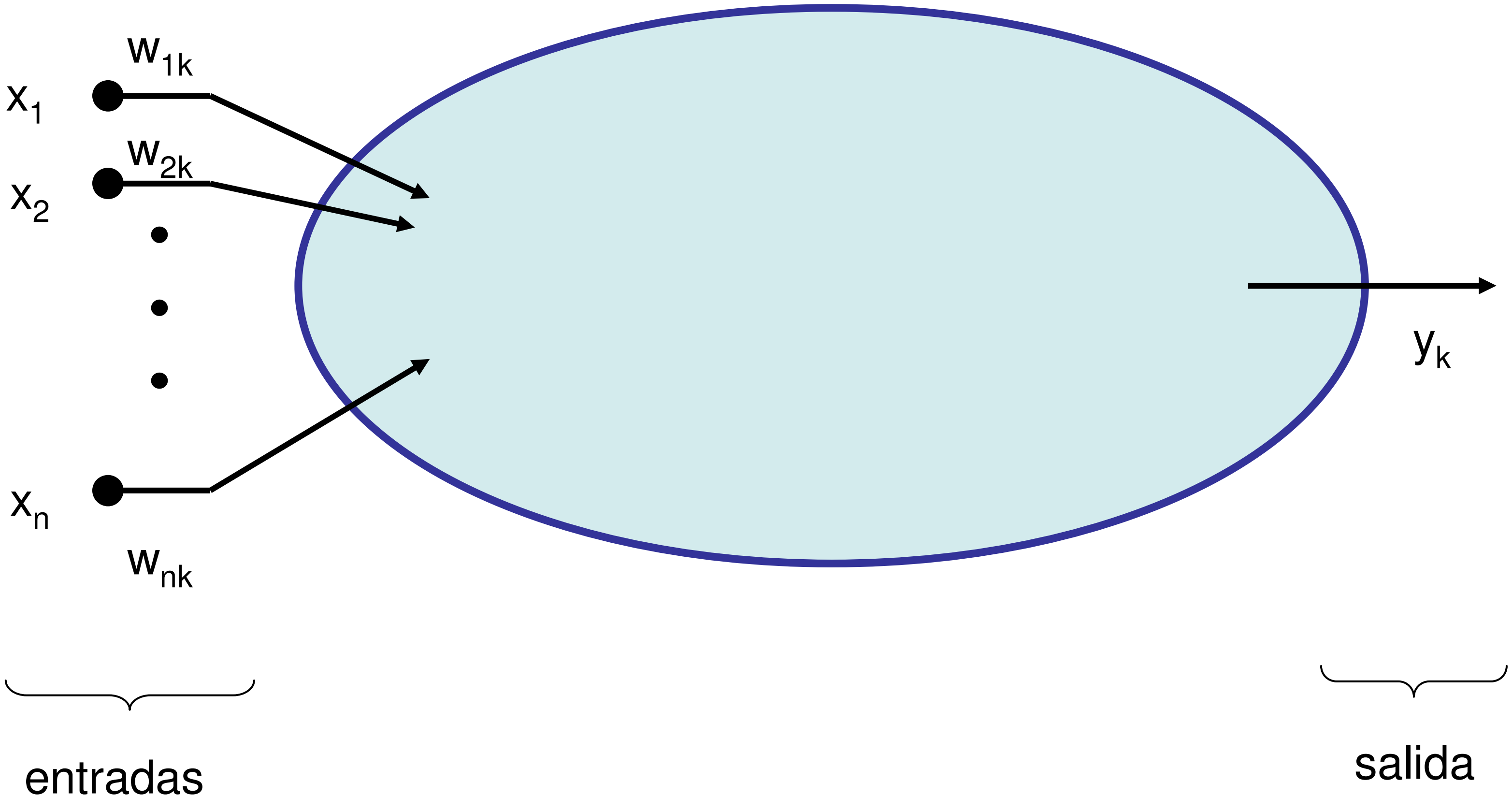


- El cuerpo y las dendritas actúan como superficie de entrada; el axón transporta las salidas.
- Los extremos de las ramas del axón forman sinapsis sobre otras neuronas.

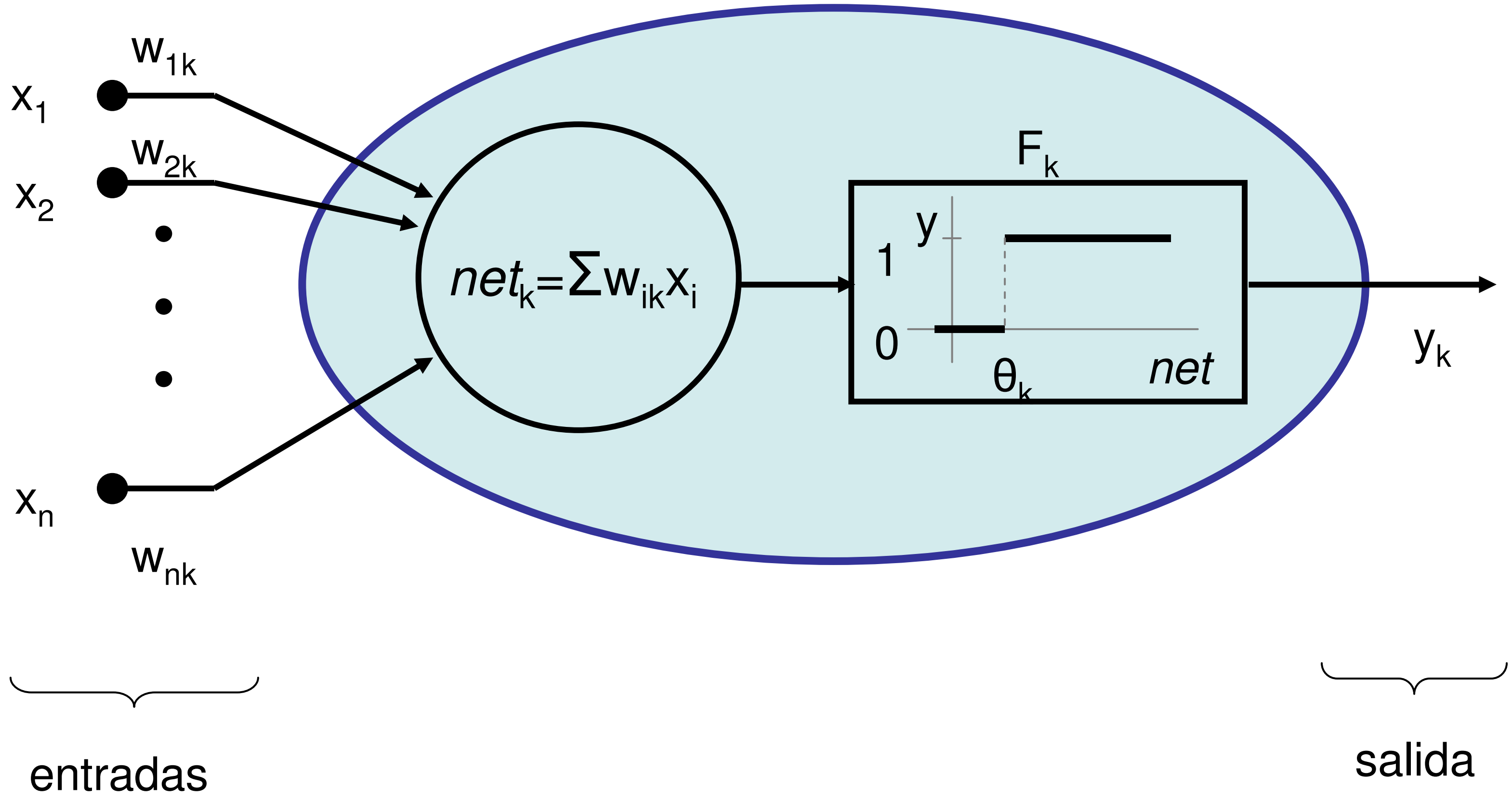
Neurona McCulloch-Pitts (1943)

- Conjunto de entradas x_i, $i = 1, 2, \ldots, n$
- Cada sinapsis desde la salida de una neurona i a la entrada de otra neurona k, tiene un peso asociado w_{ik}
  - si $w_{ik} > 0$, la sinapsis es excitadora
  - si $w_{ik} < 0$, la sinapsis es inhibidora
  - si $w_{ik} = 0$, no hay sinapsis
- Envía la salida y (0 o 1) a otras neuronas si sobrepasa el umbral

Neurona McCulloch-Pitts



Neurona McCulloch-Pitts



Neurona McCulloch-Pitts

- **Regla de propagación**

  Suma ponderada todas las entradas x_i de otras neuronas

  $$net_k = \Sigma^n{}_{i=1} w_{ik} x_i$$

- **Función de transferencia**

  F_k compara net_k con θ_k (o compara $net_k - \theta_k$ con 0) y envía la salida y (0 o 1) a otras neuronas

Notación vectorial

- Vector input:

$$\mathbf{X} = [x_0=1, x_1, x_2, \ldots. x_n]^t$$

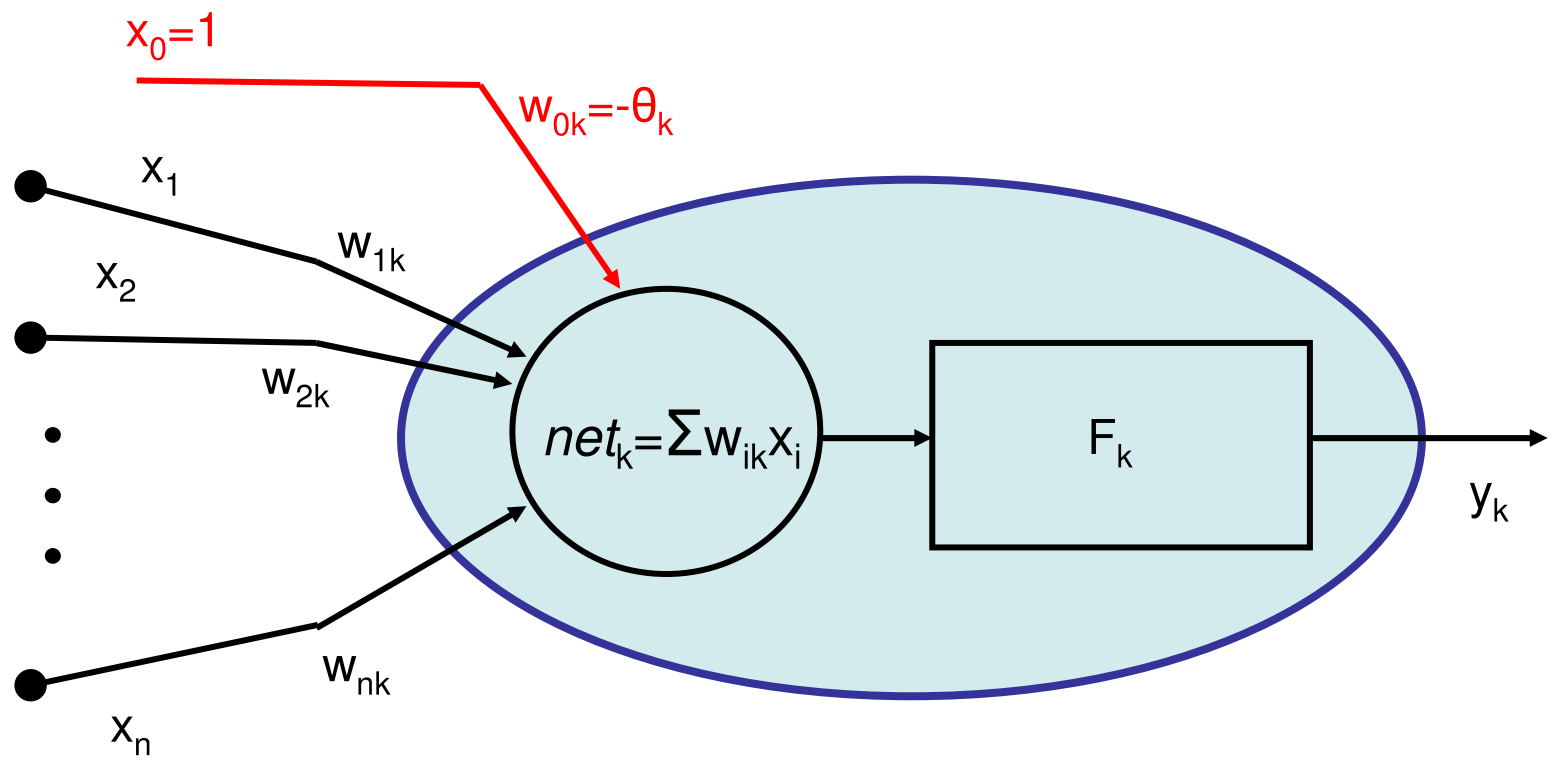
- Vector peso:

$$\mathbf{W_k} = [w_{0k} = -\theta_k, w_{1k}, w_{2k}, \ldots. w_{nk}]^t$$

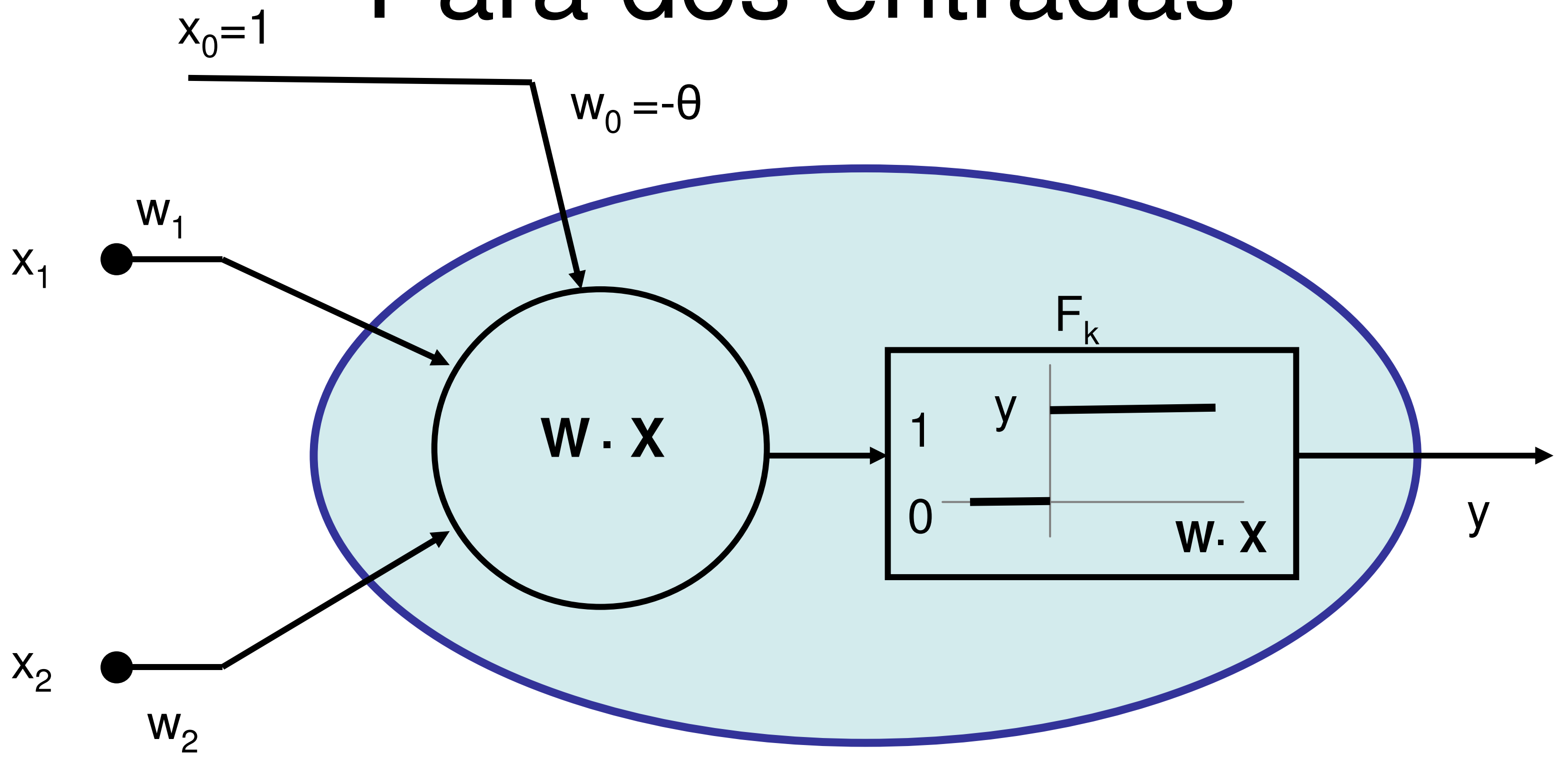
- Regla de propagación:

$$\mathbf{W^t_k \cdot X} = net_k - \theta_k$$

- Función de transferencia compara $\mathbf{W^t_k \cdot X}$ con 0.

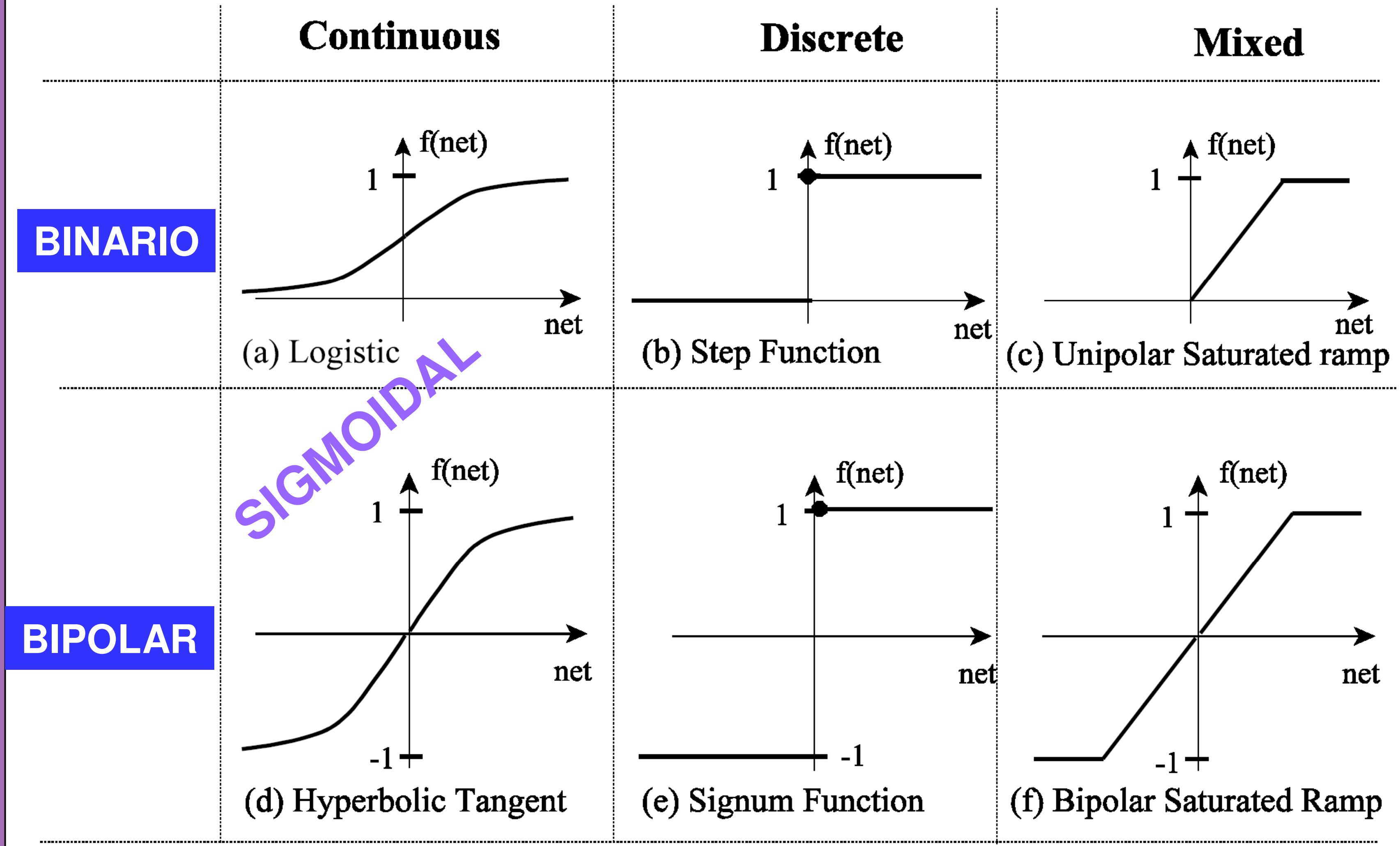

$x_0=1$
$w_{0k}=-\theta_k$
x_1
w_{1k}
x_2
w_{2k}
$net_k=\Sigma w_{ik}x_i$
F_k
y_k
w_{nk}
x_n

Para dos entradas



- Función de transferencia F_k: escalón binario
  - $y = 1$ si $\mathbf{W}^t \cdot \mathbf{X} \geq 0$ $(w_1x_1 + w_2x_2 - \theta \geq 0)$
  - $y = 0$ si $\mathbf{W}^t \cdot \mathbf{X} < 0$ $(w_1x_1 + w_2x_2 - \theta < 0)$

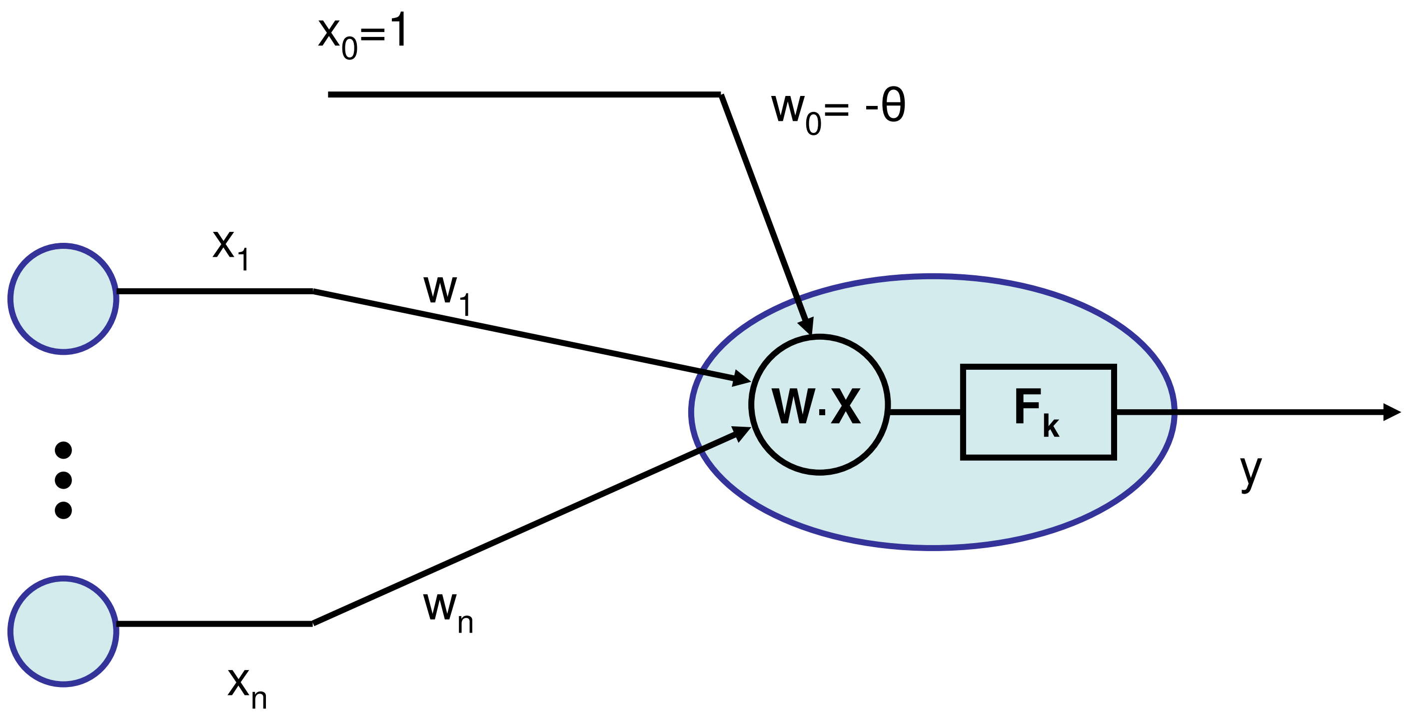
Otras funciones de transferencias



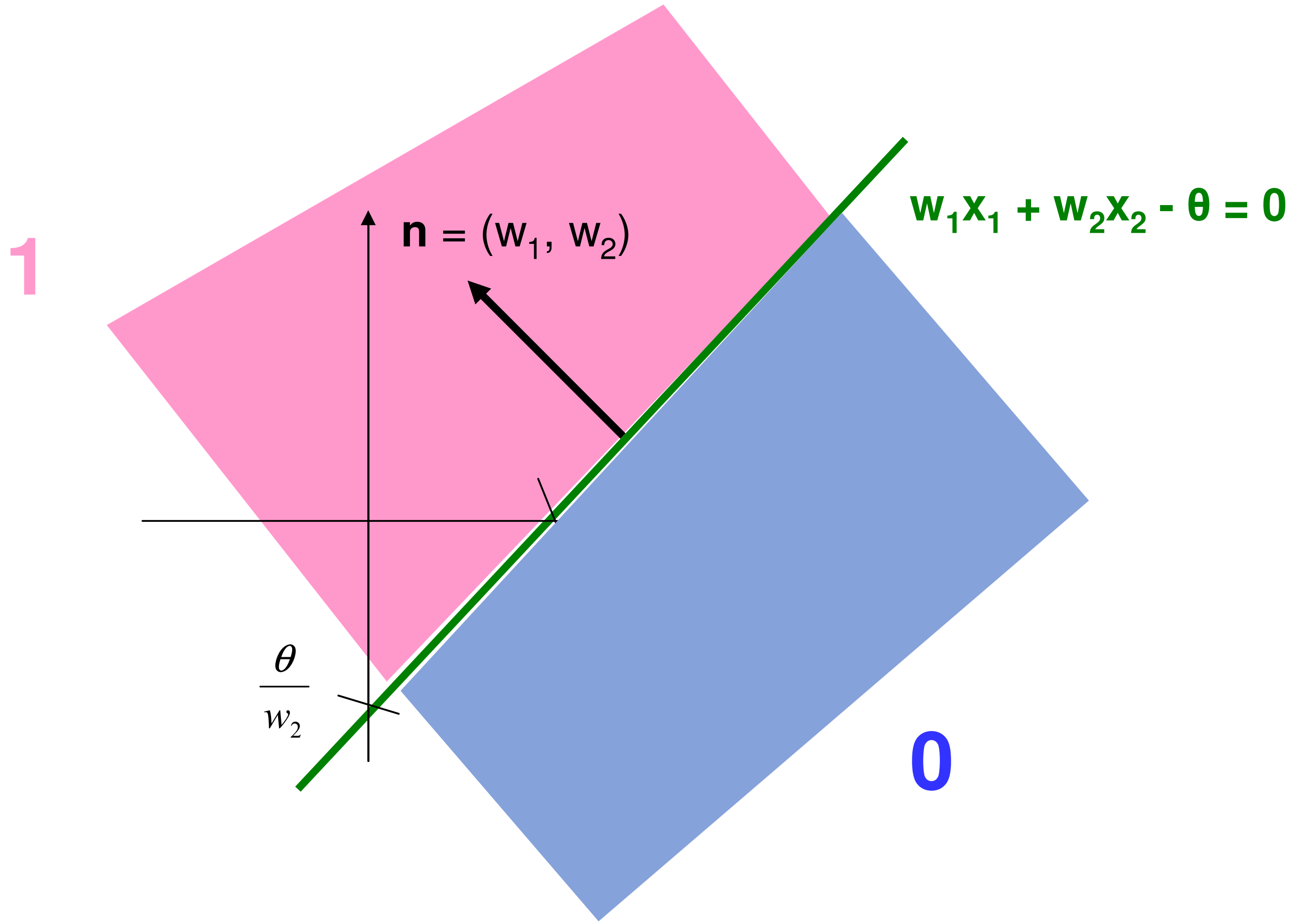
Perceptrón

- Red Neuronal Artificial más simple que consta de capa de entrada (n neuronas) y capa de salida (1 neurona), Rosenblatt 1958
- Vector entradas: $\mathbf{X} = [x_0=1, x_1, x_2, \ldots. x_n]^t$
  entradas binarias
- Vector peso: $\mathbf{W} = [w_0 = -\theta, w_1, w_2, \ldots. w_n]^t$
- Regla de propagación: $\mathbf{W^t \cdot X}$
- Función de transferencia: escalón binario o bipolar
  – $y = 1$ si $\mathbf{W^t} \cdot \mathbf{X} \geq 0$
  – $y = 0$ si $\mathbf{W^t} \cdot \mathbf{X} < 0$

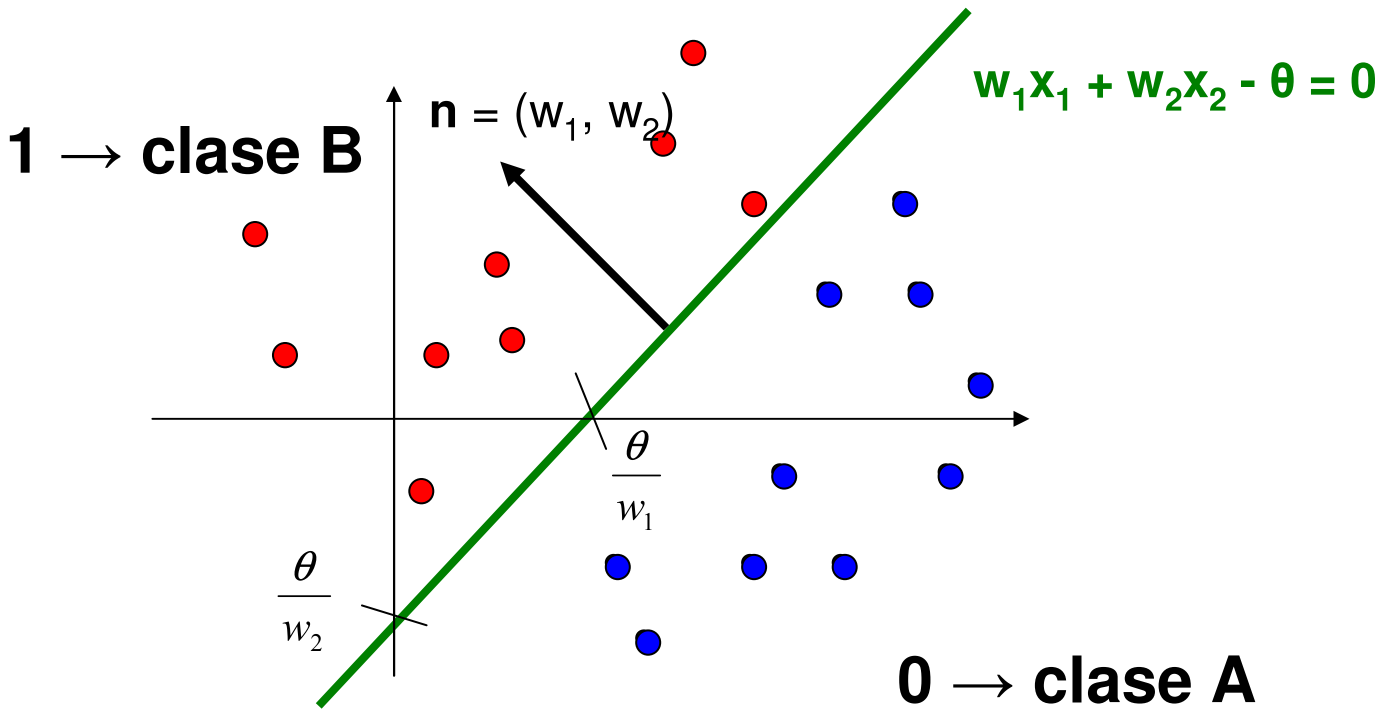
Perceptrón



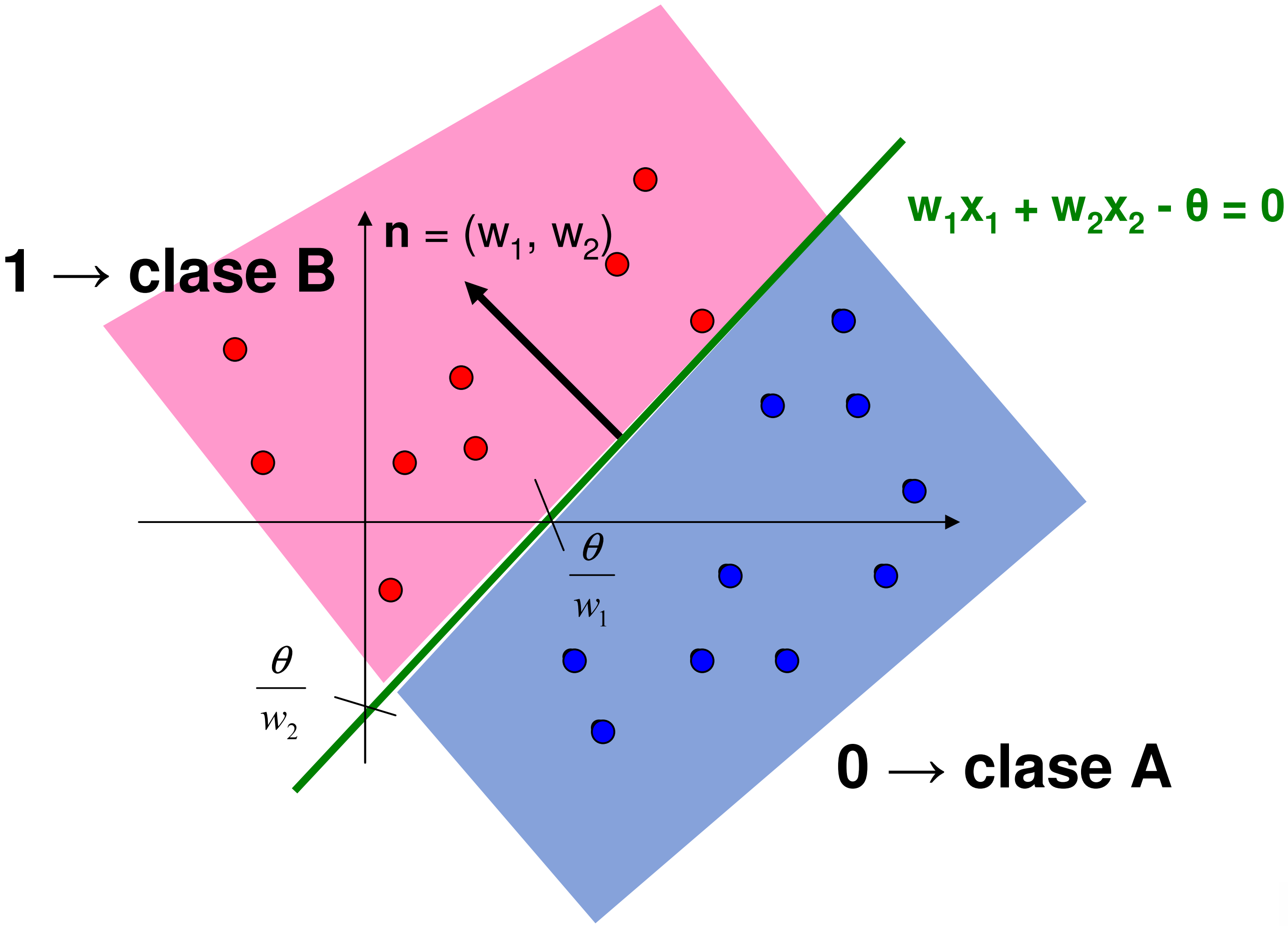
$$y = 1 \quad \text{si } w_1x_1 + w_2x_2 - \theta \geq 0$$
$$y = 0 \quad \text{si } w_1x_1 + w_2x_2 - \theta < 0$$



$$y = 1 \quad \text{si } w_1x_1 + w_2x_2 - \theta \geq 0$$
$$y = 0 \quad \text{si } w_1x_1 + w_2x_2 - \theta < 0$$



$$y = 1 \quad \text{si } w_1x_1 + w_2x_2 - \theta \geq 0$$
$$y = 0 \quad \text{si } w_1x_1 + w_2x_2 - \theta < 0$$

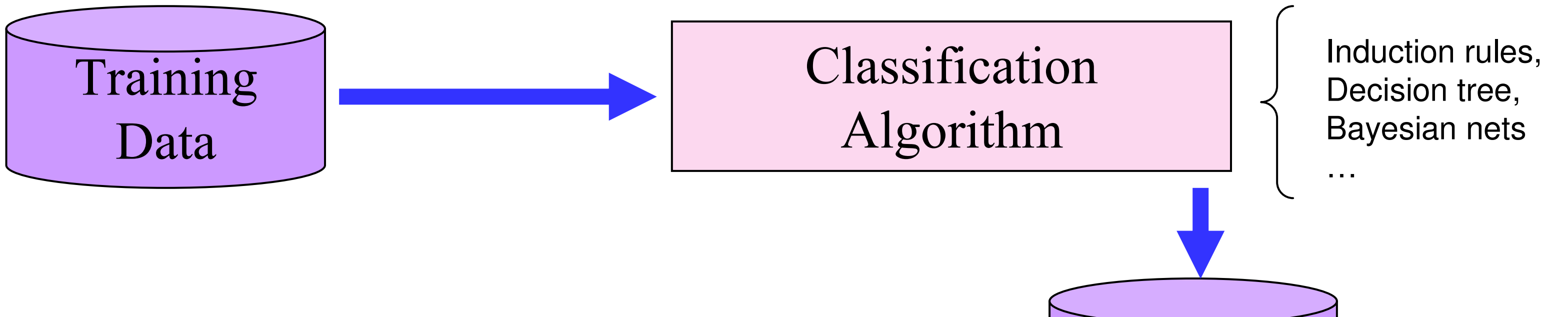


El perceptrón: clasificación

- El perceptrón es capaz de realizar tareas de clasificación de forma automática.
- A partir del conjunto de entrenamiento determinamos los valores de los pesos y el umbral.
- Al final del proceso, el perceptrón es capaz de determinar a que clase pertenece un ejemplo nuevo.

Clasificación

- A partir de un conjunto de registros
  - Cada registro está formado por un conjunto de atributos, uno de ellos es la clase.
- Encontrar un modelo para los valores de la clase como función de los valores de los demás atributos
- Objetivo: asignar la clase a registros nuevos (lo más correctamente posible)
- Aplicaciones
  - Concesión de tarjetas de crédito/hipotecas
  - Diagnóstico médico, ……..



CLASS

Age	Car	RISK
20	Combi	high
18	Sport	high
40	Sport	high
50	Family	low
35	Minivan	low
30	Combi	high
32	Family	low
40	Combi	low

Class labels: low, high

28 Combi **HIGH**

8 instances
3 attributes

Clasificador

- Es una aplicación desde el espacio de atributos $(x_1, x_2, \ldots, x_n)$ en el conjunto de m etiquetas de la clase
- Un clasificador particiona el espacio en m regiones de decisión
- La decisión de contorno (superficie discriminante) separa las clases mediante una línea o curva
  - Si n>2, hablamos de planos, superficies o hiperplanos

Clasificador lineal

- Es una aplicación desde el espacio de atributos (x_1, x_2,…, x_n) en el conjunto de m etiquetas de la clase mediante una función lineal
- El perceptrón construye una superficie discriminante lineal que separa las clases
  - En dimensión 2 mediante una recta
  - En dimensión 3 mediante un plano
  - En dimensión n mediante un hiperplano

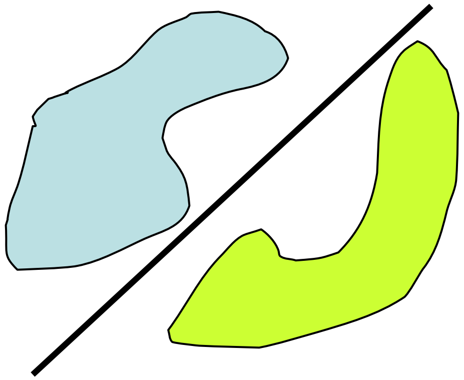
Clasificación vs. Predicción

- **Clasificación:** Para predecir el valor de un atributo categórico (discreto o nominal)
- **Predicción:** Para modelar funciones que toman valores continuos (esto es, predecir valores numéricos desconocidos)

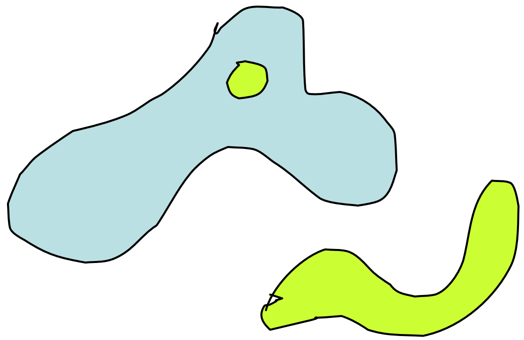
Limitaciones del perceptrón

Un perceptrón puede dividir al hiperplano en dos clases siempre y cuando éstas sean sean linealmente separables.

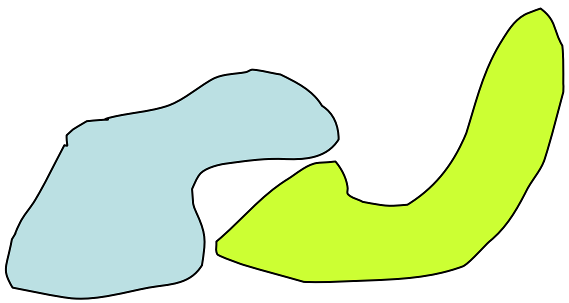
En 2D:



Si es separable



No es separable



No es linealmente separable

Aprendizaje del perceptrón

- Objetivo: determinar los valores de **W** (pesos y umbral) que clasifican los datos en dos clases.
- Entrenamiento: el perceptrón se expone a un conjunto de ejemplos de entrenamiento y el vector peso es ajustado de tal manera que al final del entrenamiento se obtienen las salidas esperadas para cada uno de los ejemplos
- Conclusión: Dada una nueva entrada, éste es clasificado correctamente.

Aprendizaje del perceptrón

- Dado el conjunto de entrenamiento:

salida deseada = etiqueta de la clase

$$[\mathbf{X}^z, d^z], \quad z = 1, 2, \ldots q$$

- Diseñamos un perceptrón que clasifique correctamente a partir de un vector peso **W** tal que el producto $\mathbf{W}^t \cdot \mathbf{X} = 0$, define un hiperplano que clasifica los ejemplos.

Algoritmo de aprendizaje

La corrección del error se realiza por el método incremental

- Se asignan valores arbitrarios a los pesos

  $\mathbf{W}^{t\,(1)} = [w_0 = -\theta, w_1, w_2, \ldots.\, w_n]^t$

- Repetimos para cada ejemplo del conjunto de entrenamiento [$\mathbf{X}^z$, d^z], hasta que todos los registros queden clasificados correctamente por el mismo vector peso
  - Presentación del ejemplo [$\mathbf{X}^z$, d^z]
  - Calculamos la salida actual $y^z = F(\mathbf{W}^{t\,(k)} \cdot \mathbf{X}^z)$
  - Calculamos el error cometido $e = d^z - y^z$
  - Adaptamos los pesos (corrección del error)

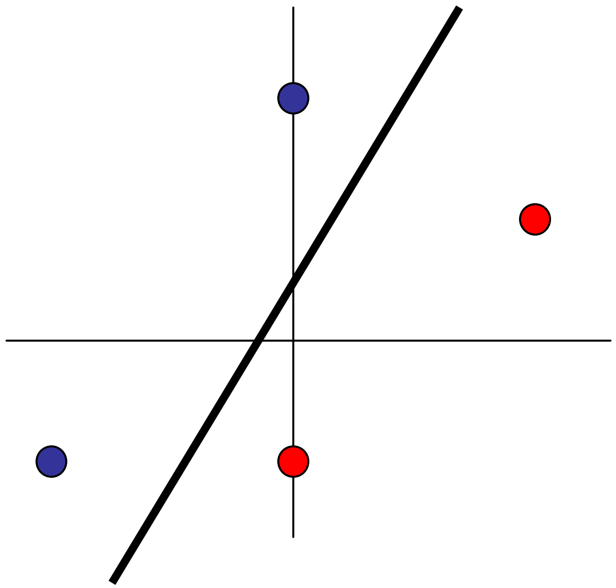
$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + \lambda(d^z - y^z)\,\mathbf{X}^z$$

λ = factor de aprendizaje

Ejemplo

- Entrena el perceptrón a partir del conjunto de entrenamiento dado. Utiliza la función escalón <u>bipolar</u> como función de transferencia y toma $\lambda = 1$

x_1	x_2	d
2	1	1
0	-1	1
-2	-1	-1
0	2	-1



1ª iteración

$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + \lambda(d^z - y^z)\,\mathbf{X}^z$$

	entrada			salida deseada	peso inicial				salida	error	peso final		
	x_0	x_1	x_2	d	w_0	w_1	w_2	**W·X**	y	e	w_0	w_1	w_2
1	1	2	1	1	0,5	-0,7	0,2	-0,7	-1	2	2,5	3,3	2,2
	1	0	-1	1									
	1	-2	-1	-1									
	1	0	2	-1									

2ª iteración

$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + \lambda(d^z - y^z)\,\mathbf{X}^z$$

	entrada			salida deseada	peso inicial				salida	error	peso final		
	x_0	x_1	x_2	d	w_0	w_1	w_2	**W·X**	y	e	w_0	w_1	w_2
1	1	2	1	1	0,5	-0,7	0,2	-0,7	-1	2	2,5	3,3	2,2
	1	0	-1	1									
	1	-2	-1	-1									
	1	0	2	-1									
2	1	2	1	1	2,5	3,3	2,2	11,3	1	0			
	1	0	-1	1	2,5	3,3	2,2	0,3	1	0			
	1	-2	-1	-1	2,5	3,3	2,2	-6,3	-1	0			
	1	0	2	-1	2,5	3,3	2,2	6,9	1	-2	0,5	3,3	-1,8

3ª iteración

$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + \lambda(d^z - y^z)\,\mathbf{X}^z$$

	entrada			salida deseada	peso inicial				salida	error	peso final		
	x_0	x_1	x_2	d	w_0	w_1	w_2	**W·X**	y	e	w_0	w_1	w_2
1	1	2	1	1	0,5	-0,7	0,2	-0,7	-1	2	2,5	3,3	2,2
	1	0	-1	1									
	1	-2	-1	-1									
	1	0	2	-1									
2	1	2	1	1	2,5	3,3	2,2	11,3	1	0			
	1	0	-1	1	2,5	3,3	2,2	0,3	1	0			
	1	-2	-1	-1	2,5	3,3	2,2	-6,3	-1	0			
	1	0	2	-1	2,5	3,3	2,2	6,9	1	-2	0,5	3,3	-1,8
3	1	2	1	1	0,5	3,3	-1,8	5,3	1	0			
	1	0	-1	1	0,5	3,3	-1,8	2,3	1	0			
	1	-2	-1	-1	0,5	3,3	-1,8	-4,3	-1	0			
	1	0	2	-1	0,5	3,3	-1,8	-3,1	-1	0			

$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + \lambda(d^z - y^z)\,\mathbf{X}^z$$

	entrada			salida deseada	peso inicial				salida	error	peso final		
	x_0	x_1	x_2	d	w_0	w_1	w_2	**W·X**	y	e	w_0	w_1	w_2
1	1	2	1	1	0,5	-0,7	0,2	-0,7	-1	2	2,5	3,3	2,2
	1	0	-1	1									
	1	-2	-1	-1									
	1	0	2	-1									
2	1	2	1	1	2,5	3,3	2,2	11,3	1	0			
	1	0	-1	1	2,5	3,3	2,2	0,3	1	0			
	1	-2	-1	-1	2,5	3,3	2,2	-6,3	-1	0			
	1	0	2	-1	2,5	3,3	2,2	6,9	1	-2	0,5	3,3	-1,8
3	1	2	1	1	0,5	3,3	-1,8	4,8	1	0			
	1	0	-1	1	0,5	3,3	-1,8	2,3	1	0			
	1	-2	-1	-1	0,5	3,3	-1,8	-4,3	-1	0			
	1	0	2	-1	0,5	3,3	-1,8	-3,1	-1	0			

SOLUCION: w_0 = 0,5; w_1 = 3,3; w_2 = -1,8;

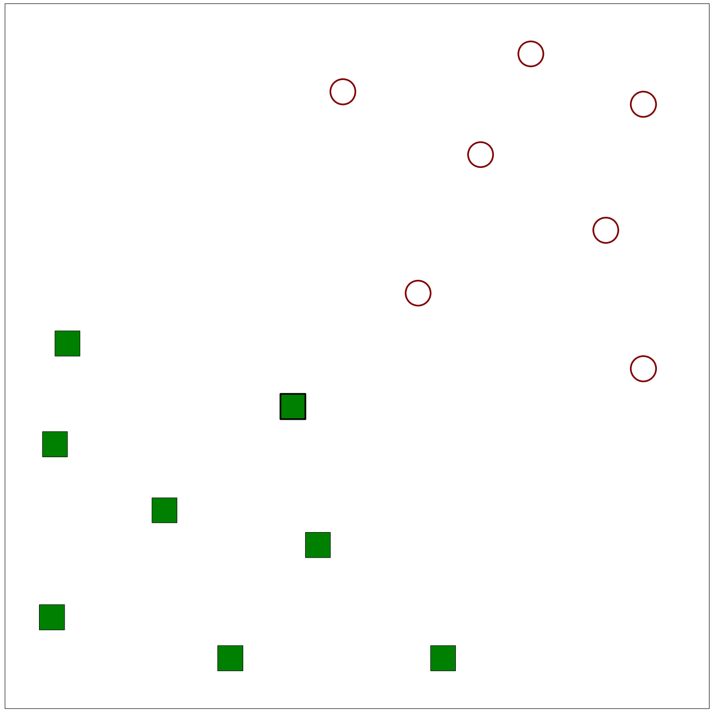
¿Cómo clasificarías la entrada **X** =(1 0 0)?

- Para **W** = (0,5 3,3 -1,8), calculamos la entrada neta y aplicamos la función de transferencia

$$F(\mathbf{W}\cdot\mathbf{X}) = F(0{,}5) = 1$$

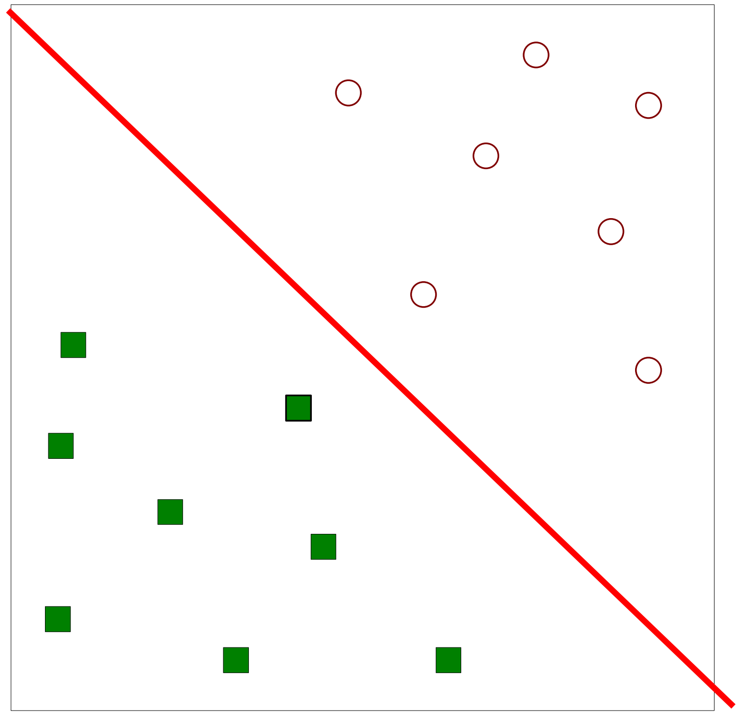
Máquinas de soporte vectorial

- Encontrar el hiperplano lineal (decisión de contorno) que separe los datos



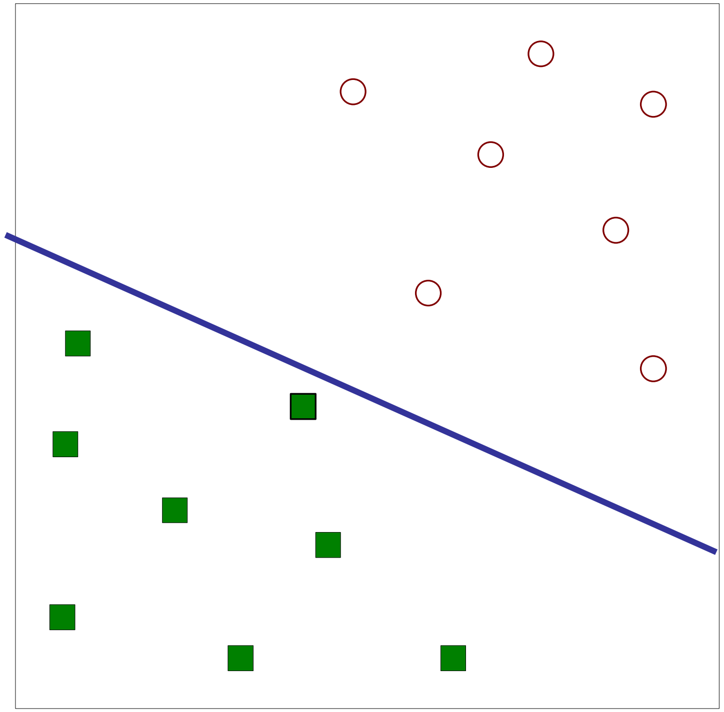
Máquinas de soporte vectorial

- Una posible solución



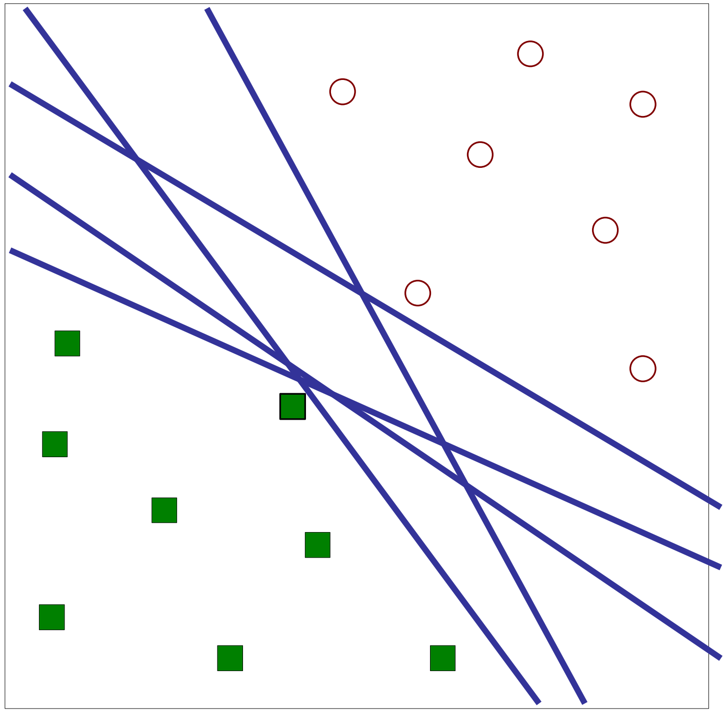
Máquinas de soporte vectorial

- Otra solución



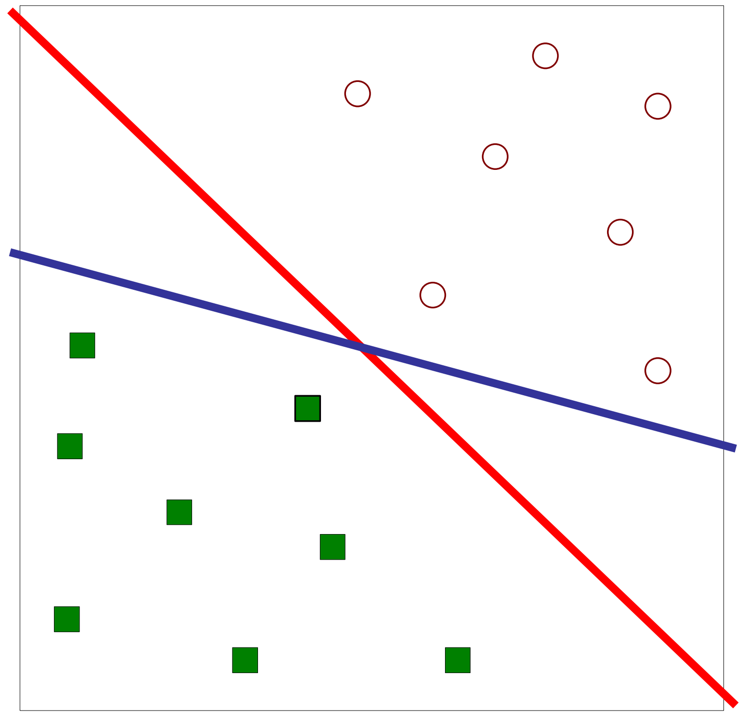
Máquinas de soporte vectorial

- Otras soluciones



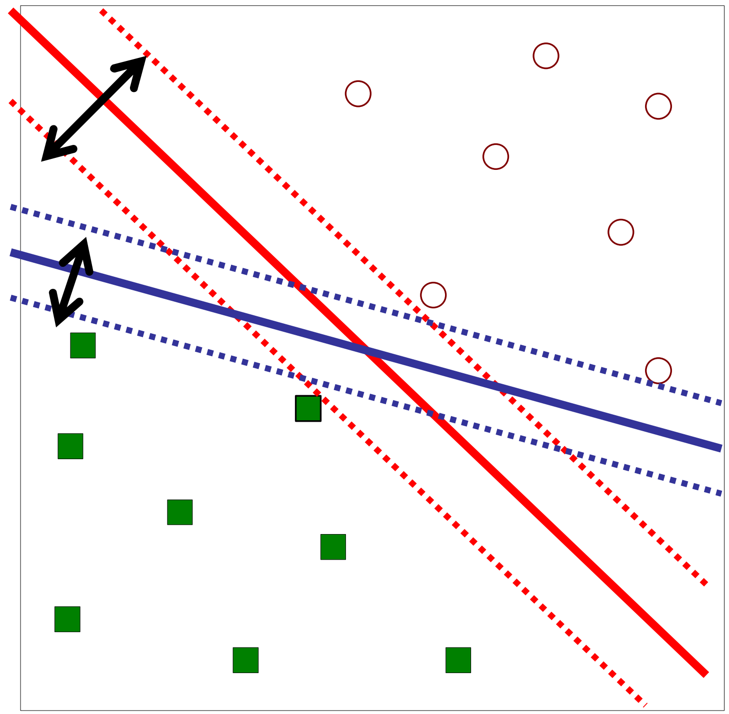
Máquinas de soporte vectorial

- ¿Cuál es mejor?
- ¿Cómo defines "mejor"?



Máquinas de soporte vectorial

- “Mejor”: hiperplano que maximiza el margen

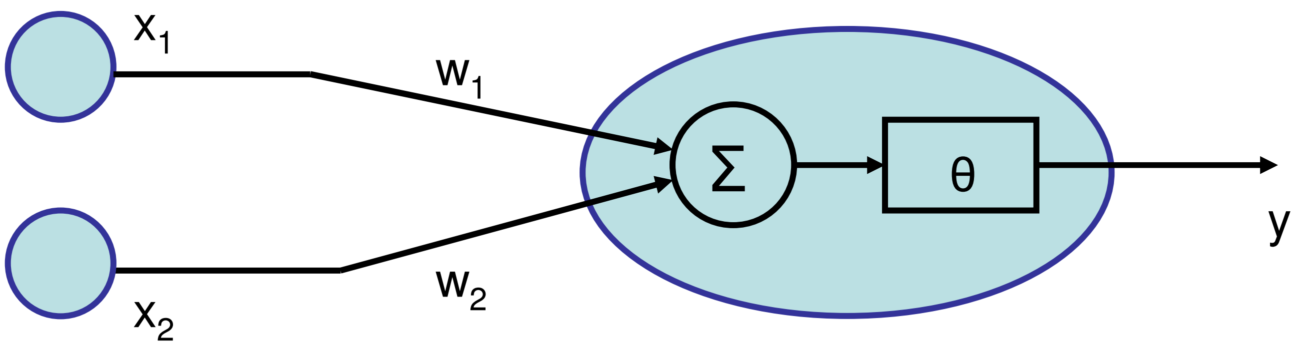


- ROJO es “mejor” que AZUL

Problema de optimización con restricciones

Perceptrón: AND, OR y NOT

- Dos entradas binarias
- Función de transferencia: escalón binario
  - $y = 1$ si $w_1x_1 + w_2x_2 - \theta \geq 0$
  - $y = 0$ si $w_1x_1 + w_2x_2 - \theta < 0$



- Hay que determinar los pesos y el umbral

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

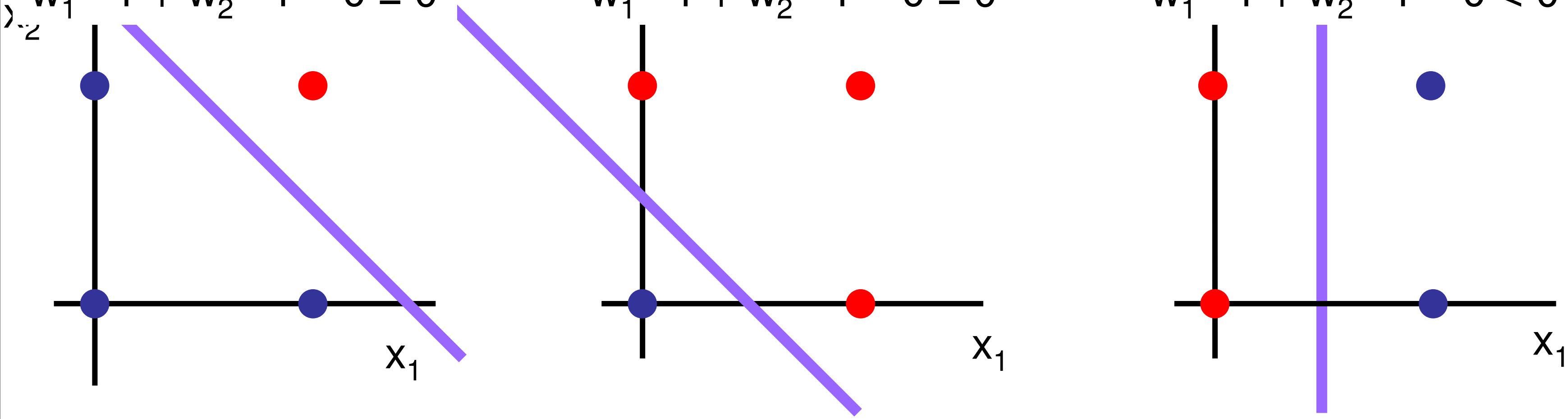
$$w_1 \cdot 0 + w_2 \cdot 0 - \theta < 0$$
$$w_1 \cdot 0 + w_2 \cdot 1 - \theta < 0$$
$$w_1 \cdot 1 + w_2 \cdot 0 - \theta < 0$$
$$w_1 \cdot 1 + w_2 \cdot 1 - \theta \geq 0$$

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

$$w_1 \cdot 0 + w_2 \cdot 0 - \theta < 0$$
$$w_1 \cdot 0 + w_2 \cdot 1 - \theta \geq 0$$
$$w_1 \cdot 1 + w_2 \cdot 0 - \theta \geq 0$$
$$w_1 \cdot 1 + w_2 \cdot 1 - \theta \geq 0$$

NOT		
x_1	x_2	y
0	0	1
0	1	1
1	0	0
1	1	0

$$w_1 \cdot 0 + w_2 \cdot 0 - \theta \geq 0$$
$$w_1 \cdot 0 + w_2 \cdot 1 - \theta \geq 0$$
$$w_1 \cdot 1 + w_2 \cdot 0 - \theta < 0$$
$$w_1 \cdot 1 + w_2 \cdot 1 - \theta < 0$$



AND		
x_1	x_2	y

$$w_1 \cdot 0 + w_2 \cdot 0 - \theta < 0$$
$$w_1 \cdot 0 + w_2 \cdot 1 - \theta < 0$$
$$w_1 \cdot 1 + w_2 \cdot 0 - \theta < 0$$
$$w_1 \cdot 1 + w_2 \cdot 1 - \theta \geq 0$$

1	1	1

OR		
x_1	x_2	y

$$w_1 \cdot 0 + w_2 \cdot 0 - \theta < 0$$
$$w_1 \cdot 0 + w_2 \cdot 1 - \theta \geq 0$$
$$w_1 \cdot 1 + w_2 \cdot 0 - \theta \geq 0$$
$$w_1 \cdot 1 + w_2 \cdot 1 - \theta \geq 0$$

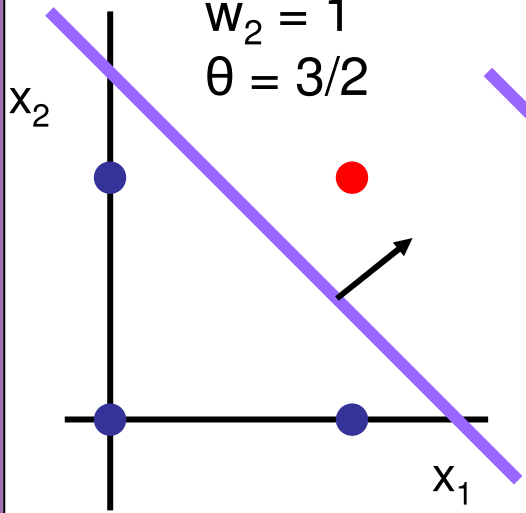
1	1	1

NOT		
x_1	x_2	y

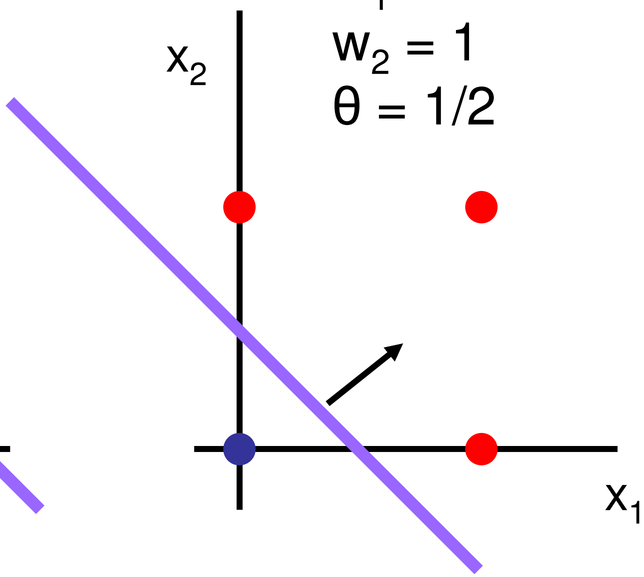
$$w_1 \cdot 0 + w_2 \cdot 0 - \theta \geq 0$$
$$w_1 \cdot 0 + w_2 \cdot 1 - \theta \geq 0$$
$$w_1 \cdot 1 + w_2 \cdot 0 - \theta < 0$$
$$w_1 \cdot 1 + w_2 \cdot 1 - \theta < 0$$

1	1	0

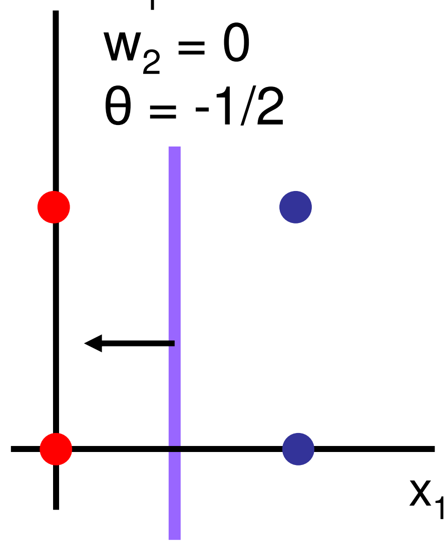
$w_1 = 1$
$w_2 = 1$
$\theta = 3/2$



$w_1 = 1$
$w_2 = 1$
$\theta = 1/2$



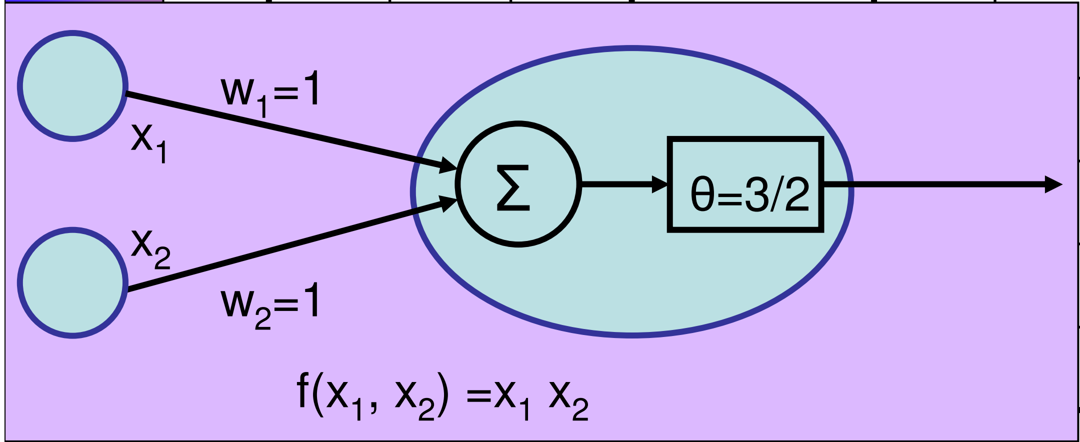
$w_1 = -1$
$w_2 = 0$
$\theta = -1/2$



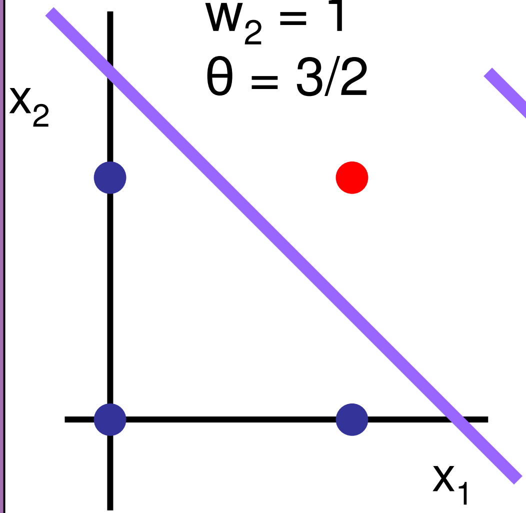
AND		

OR		
		y
		0
		1
		1
		1

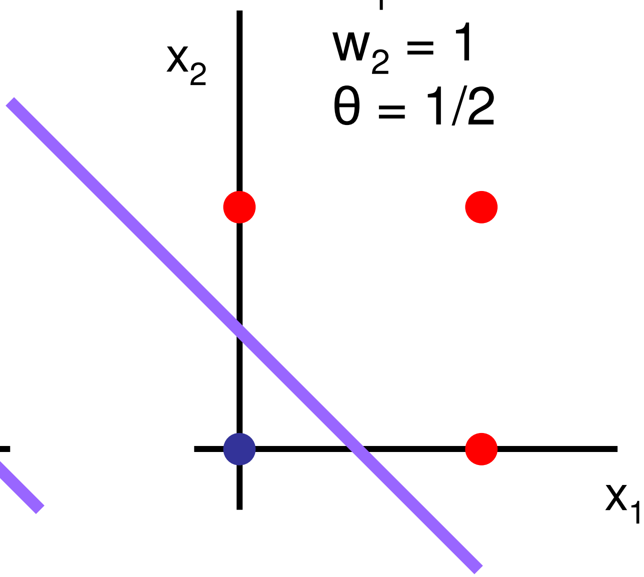
NOT		
x_1	x_2	y
0	0	1
0	1	1
1	0	0
1	1	0



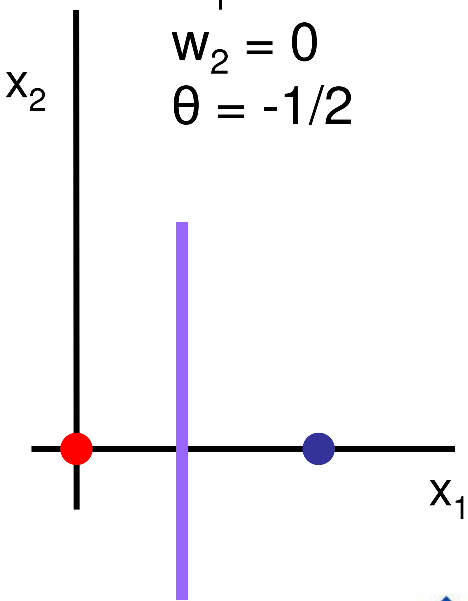
$w_1 = 1$
$w_2 = 1$
$\theta = 3/2$



$w_1 = 1$
$w_2 = 1$
$\theta = 1/2$



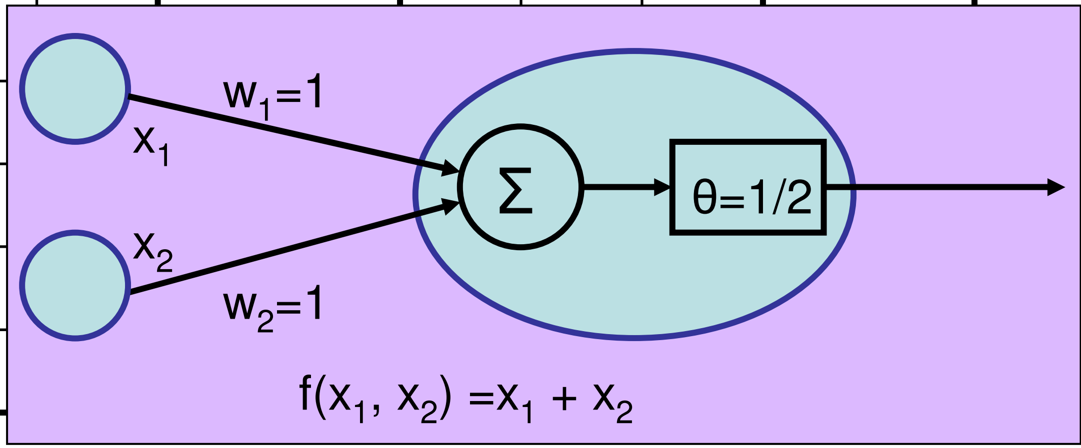
$w_1 = -1$
$w_2 = 0$
$\theta = -1/2$



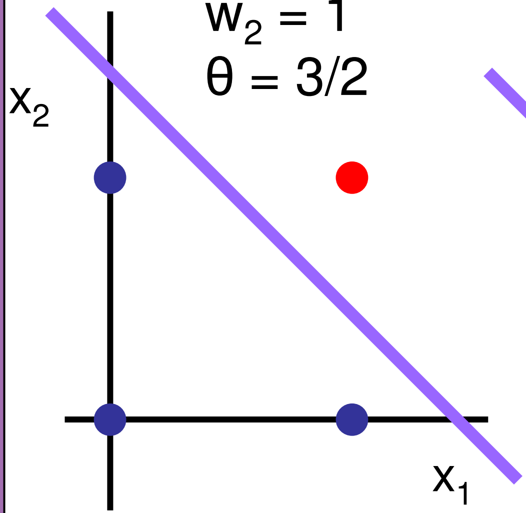
AND	
x_1	x_2
0	0
0	1
1	0
1	1

OR

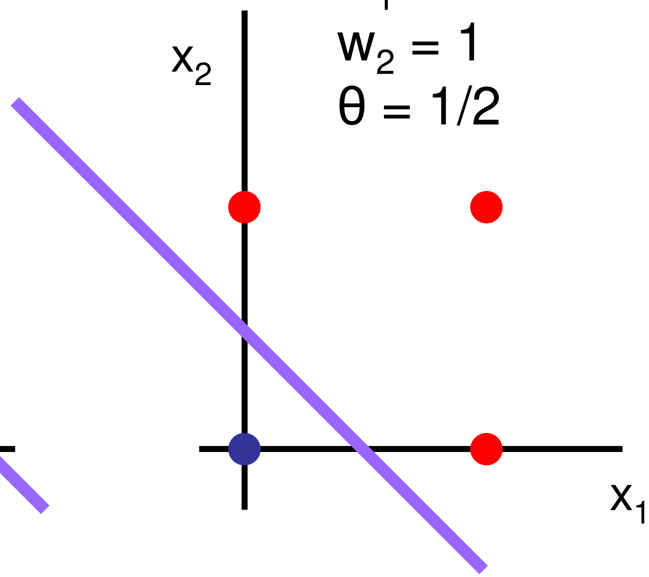
NOT	
x_2	y
0	1
1	1
0	0
1	0



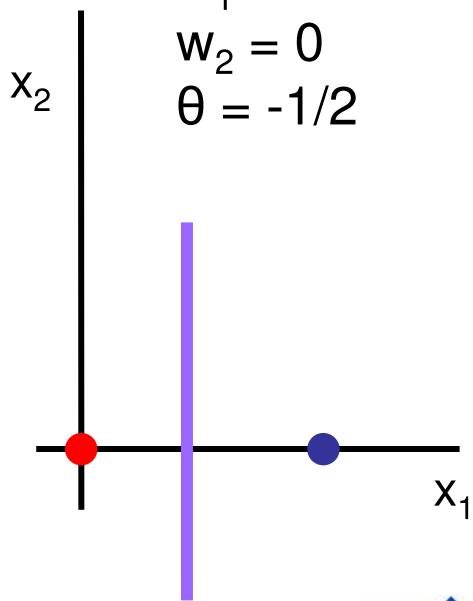
$w_1 = 1$
$w_2 = 1$
$\theta = 3/2$



$w_1 = 1$
$w_2 = 1$
$\theta = 1/2$



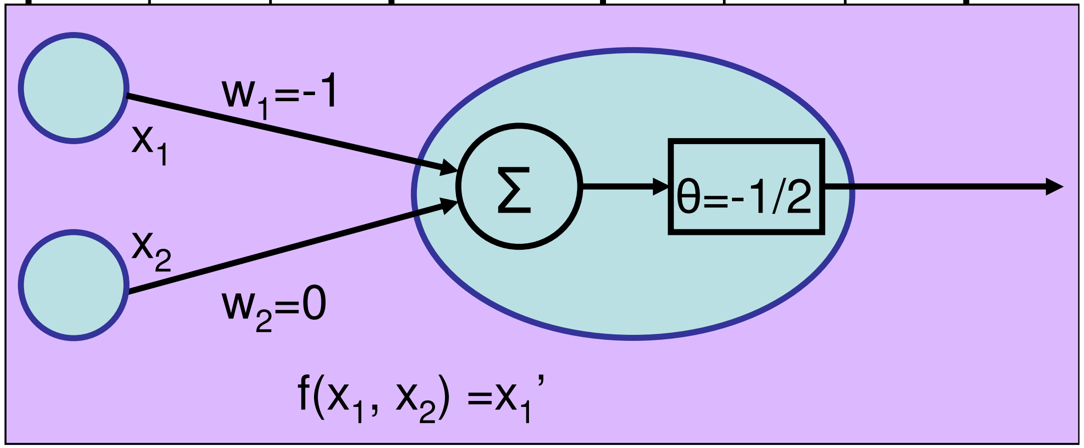
$w_1 = -1$
$w_2 = 0$
$\theta = -1/2$



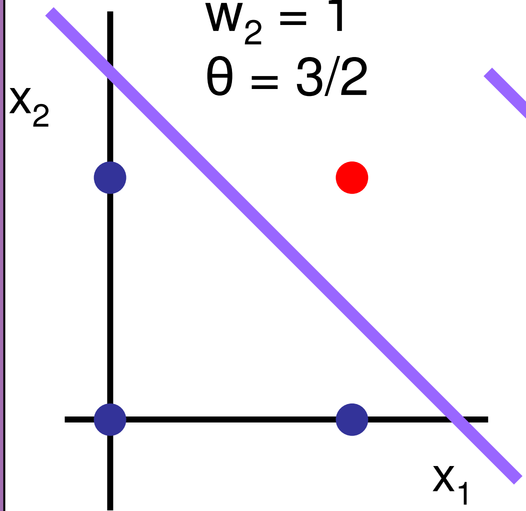
AND		
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

OR

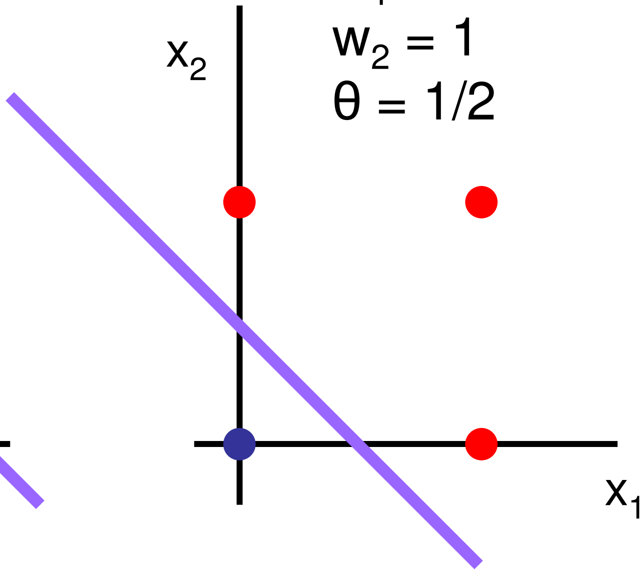
NOT



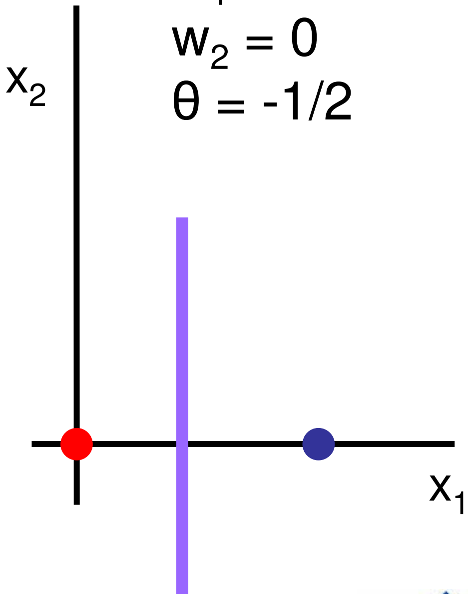
$w_1 = 1$
$w_2 = 1$
$\theta = 3/2$



$w_1 = 1$
$w_2 = 1$
$\theta = 1/2$



$w_1 = -1$
$w_2 = 0$
$\theta = -1/2$



- Generalizamos para n entradas

$f = x_1 x_2 \ldots x_n$

$w_1 =$

$w_2 =$

……..

$w_n =$

$\theta =$

$f = x_1 + x_2 + \ldots + x_n$

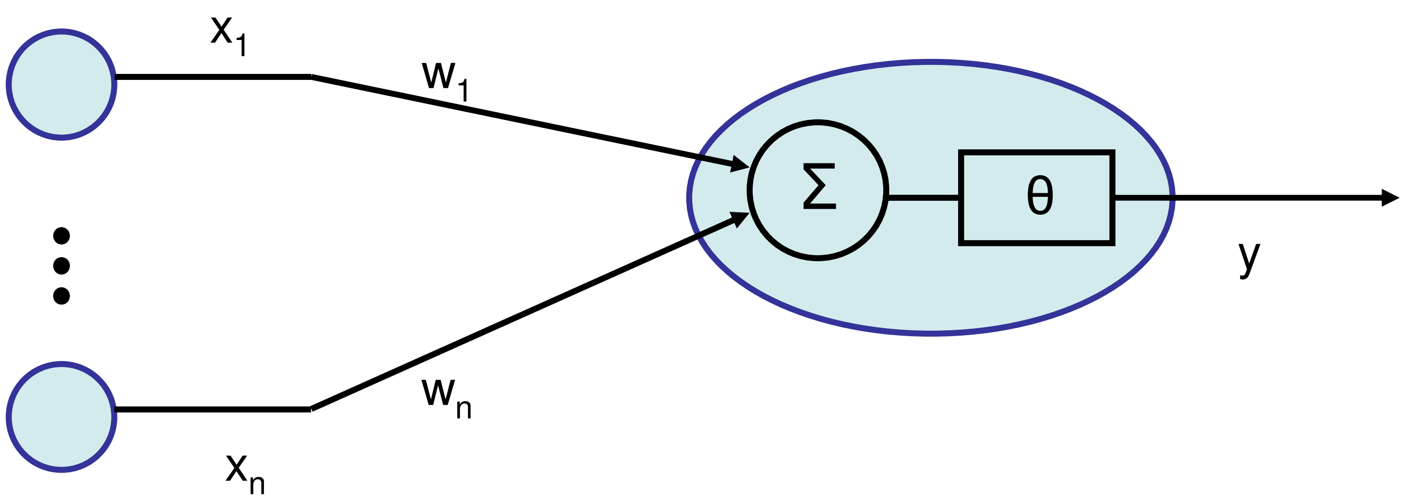
$w_1 =$

$w_2 =$

……..

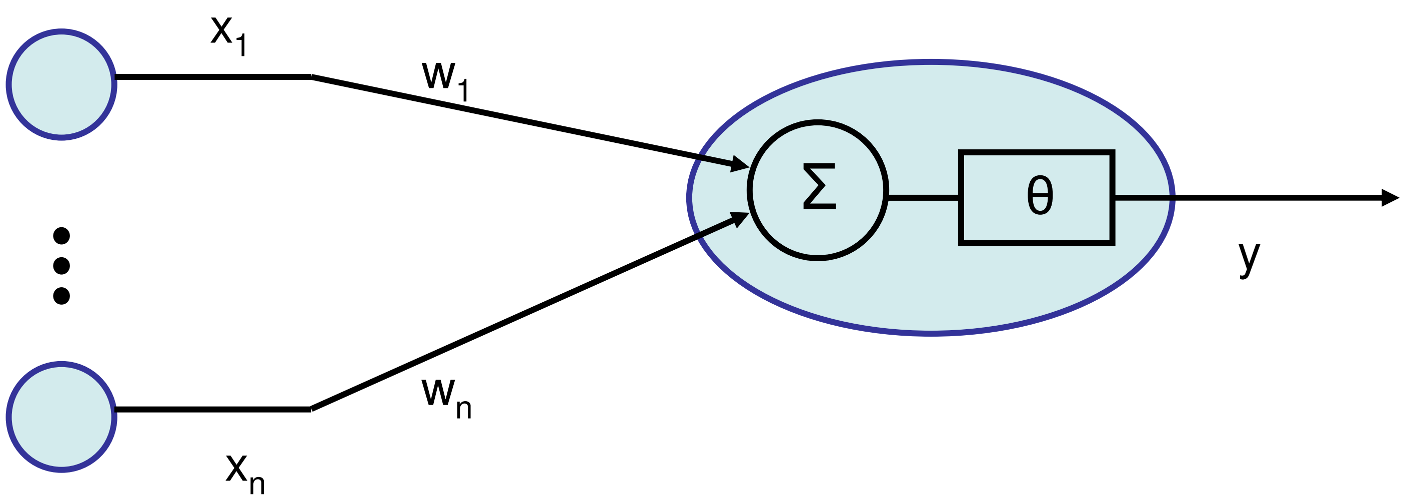
$w_n =$

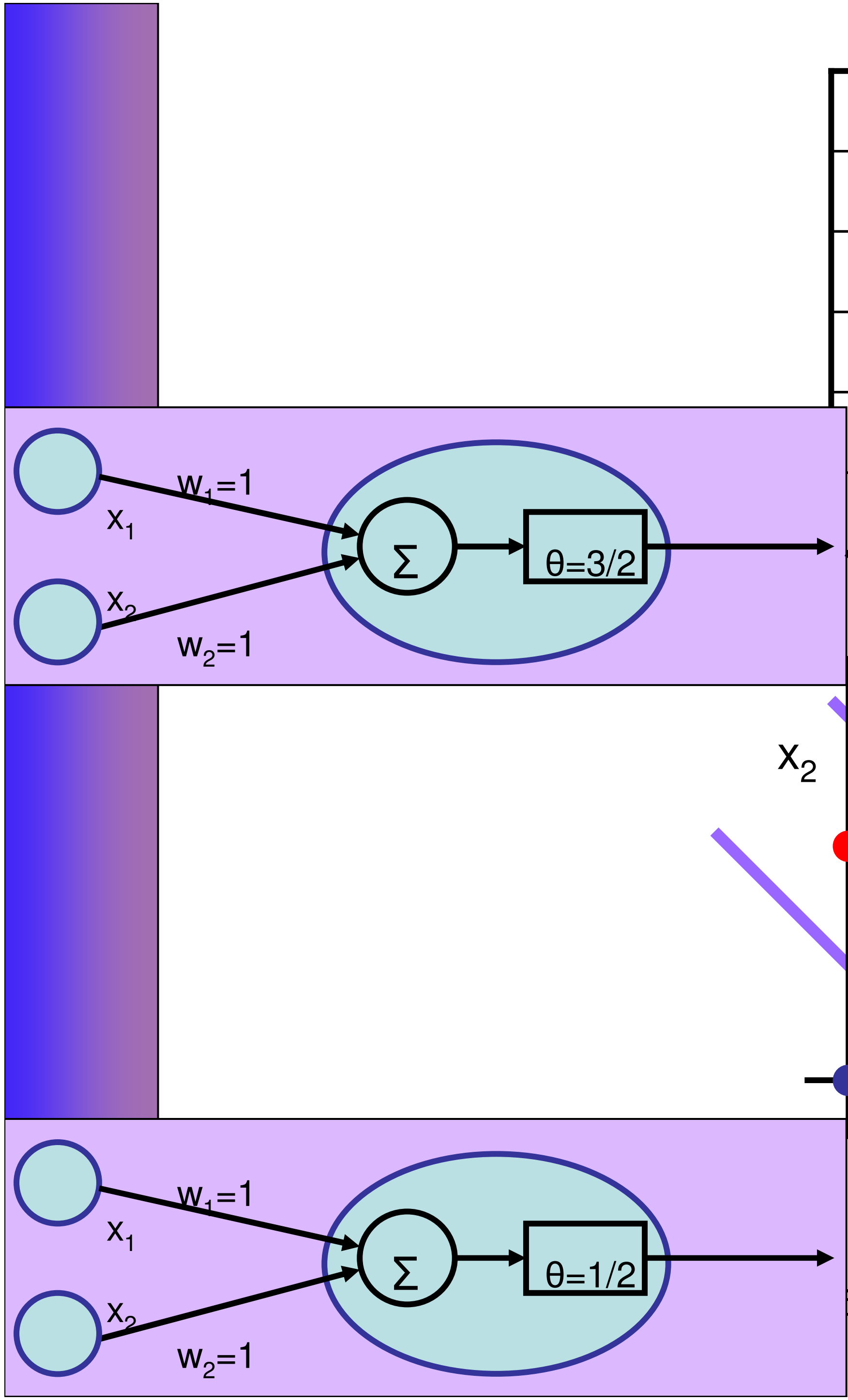
$\theta =$



- Generalizamos para n entradas

$f = x_1 x_2 \ldots x_n$	$f = x_1 + x_2 + \ldots + x_n$
$w_1 = 1$	$w_1 = 1$
$w_2 = 1$	$w_2 = 1$
……..	……..
$w_n = 1$	$w_n = 1$
$\theta = n-1/2$	$\theta = 1/2$

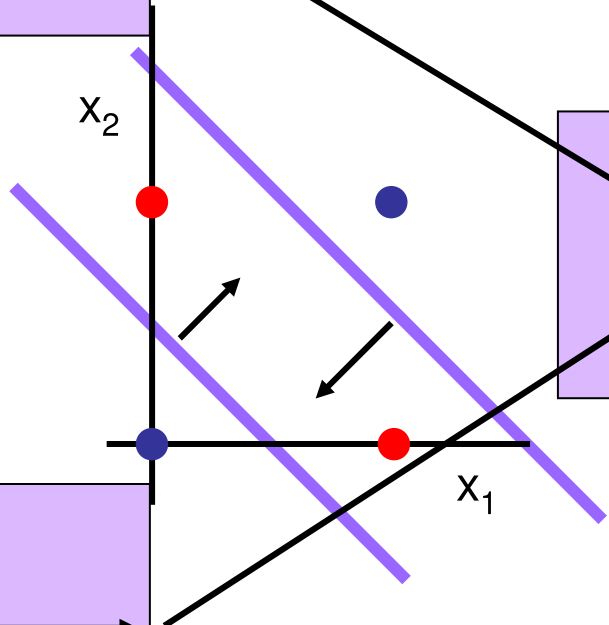


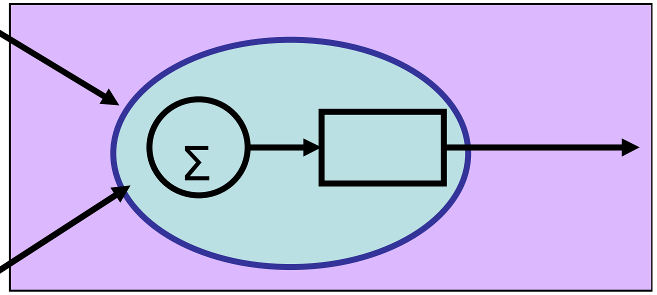


XOR		
x_1	x_2	y
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

$w_1 \cdot 0 + w_2 \cdot 0 - \theta < 0$
$w_1 \cdot 0 + w_2 \cdot 1 - \theta \geq 0$
$w_1 \cdot 1 + w_2 \cdot 0 - \theta \geq 0$
$w_1 \cdot 1 + w_2 \cdot 1 - \theta < 0$

$\theta > 0$
$w_1 \geq \theta$
$w_2 \geq \theta$
$w_1 + w_2 < \theta$

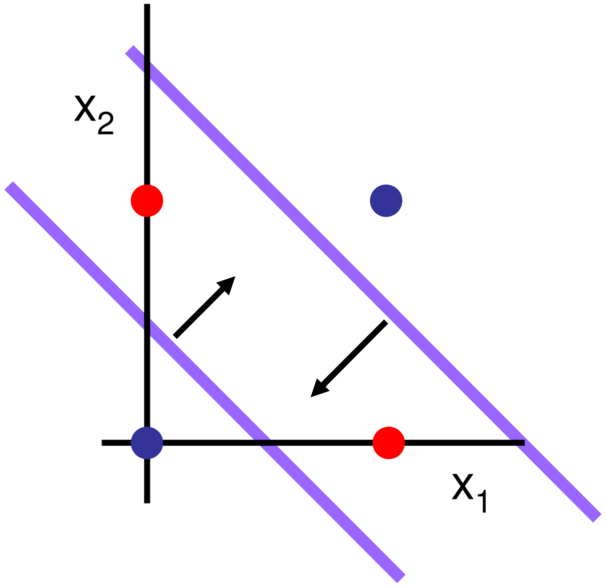


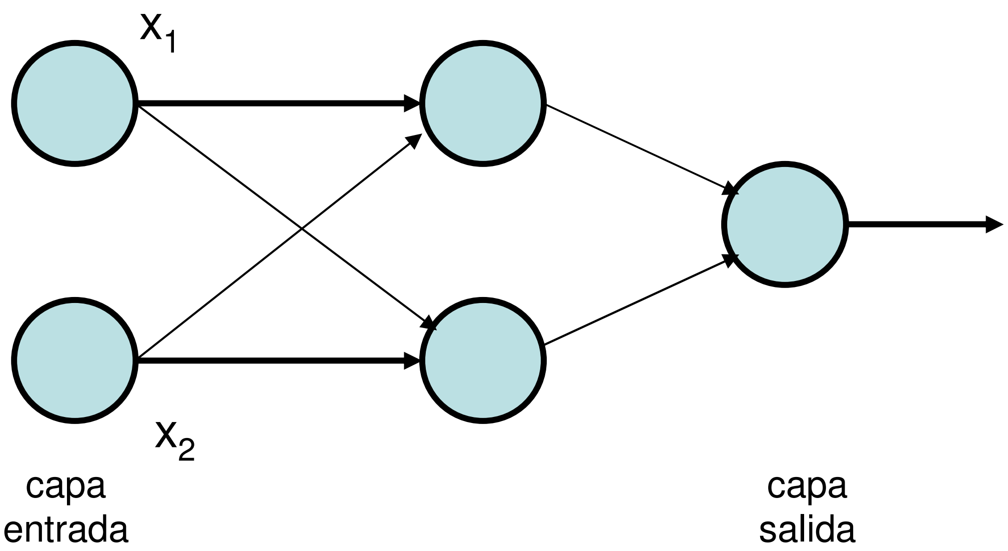


XOR		
x_1	x_2	y
0	0	0
0	1	1

$$w_1 \cdot 0 + w_2 \cdot 0 - \theta < 0$$
$$w_1 \cdot 0 + w_2 \cdot 1 - \theta \geq 0$$
$$w_1 \cdot 1 + w_2 \cdot 0 - \theta \geq 0$$
$$w_1 \cdot 1 + w_2 \cdot 1 - \theta < 0$$

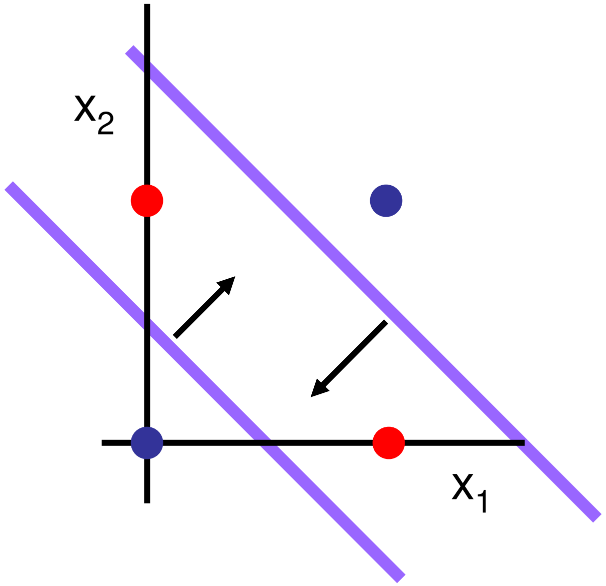
Con un perceptrón multicapa podemos modelar cualquier expresión lógica, basta con diseñar la red y determinar w_i y θ

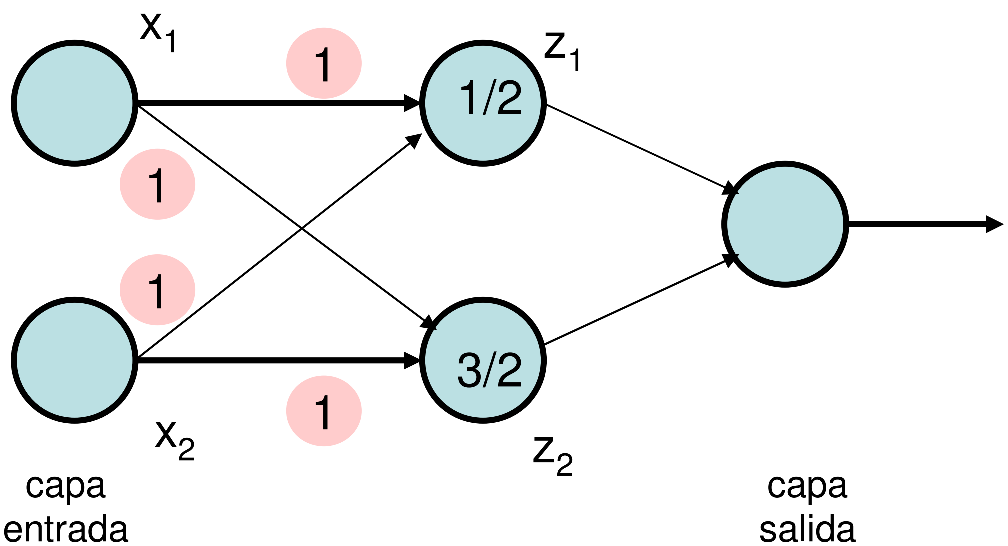




XOR		
x_1	x_2	y
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

$$w_1 \cdot 0 + w_2 \cdot 0 - \theta < 0$$
$$w_1 \cdot 0 + w_2 \cdot 1 - \theta \geq 0$$
$$w_1 \cdot 1 + w_2 \cdot 0 - \theta \geq 0$$
$$w_1 \cdot 1 + w_2 \cdot 1 - \theta < 0$$

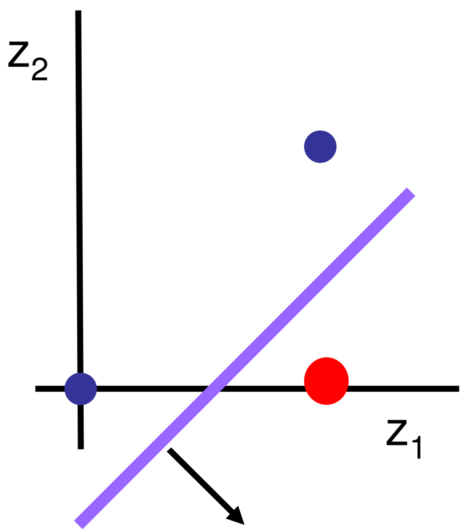


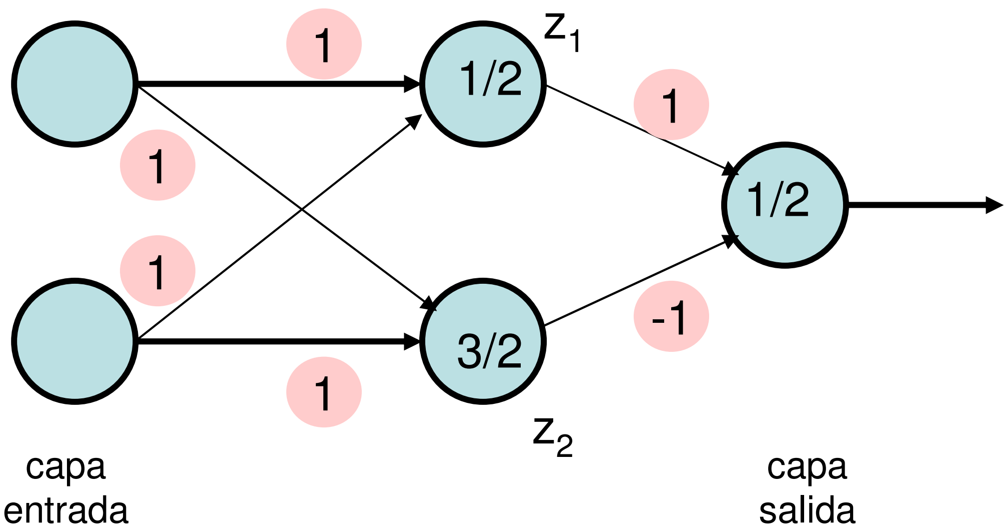


$z_1 = F(1x_1+1x_2-1/2)$
$z_2 = F(1x_1+1x_2-3/2)$

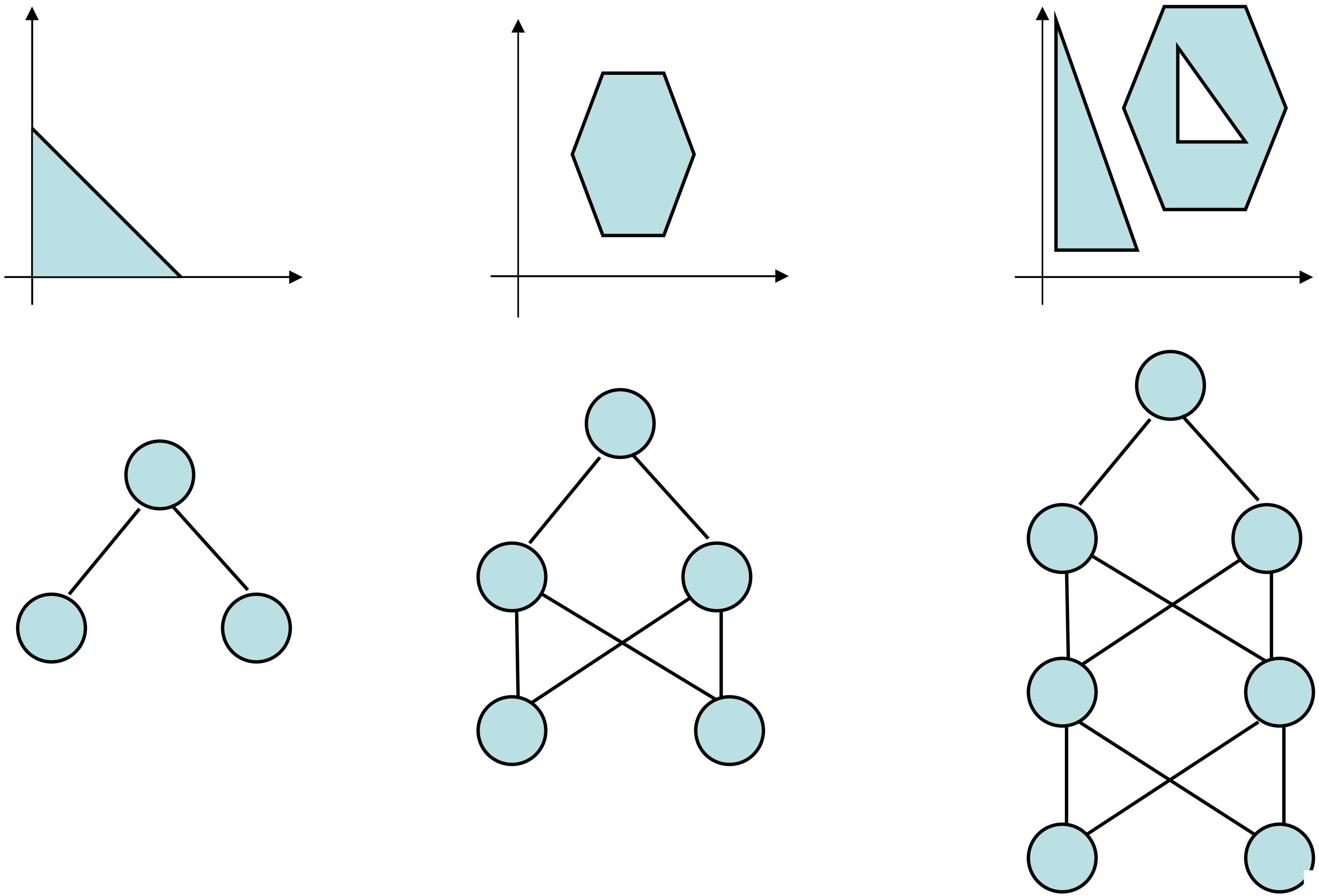
XOR				
x_1	x_2	y	z_1	z_2
0	0	**0**	0	0
0	1	**1**	1	0
1	0	**1**	1	0
1	1	**0**	1	1

En el espacio (z_1, z_2) el problema XOR es linealmente separable





Interpretación de las capas



primera capa contornos lineales

segunda capa combina contornos lineales

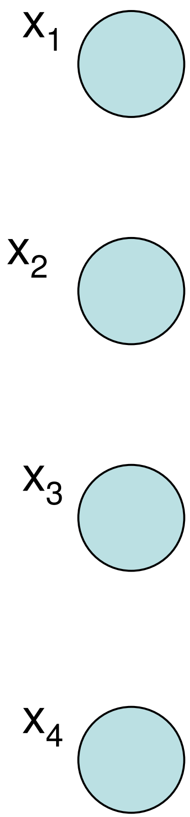
tercera capa contornos más complejos

Podemos modelar cualquier expresión lógica

$$f(x_1, x_2, x_3, x_4) = x_1 x_2' x_3 + x_1' x_2 x_3' x_4$$

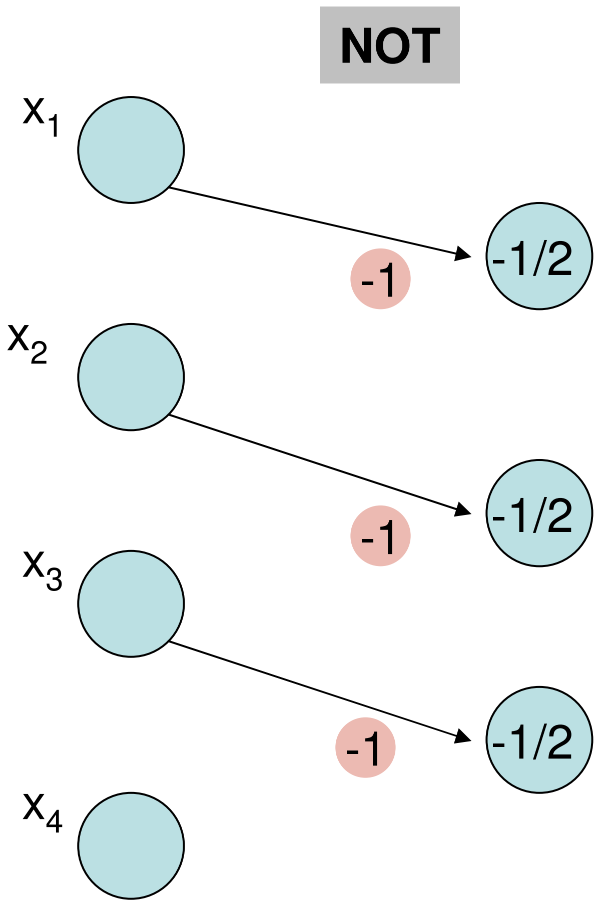
Podemos modelar cualquier expresión lógica

$$f(x_1, x_2, x_3, x_4) = x_1 x_2' x_3 + x_1' x_2 x_3' x_4$$



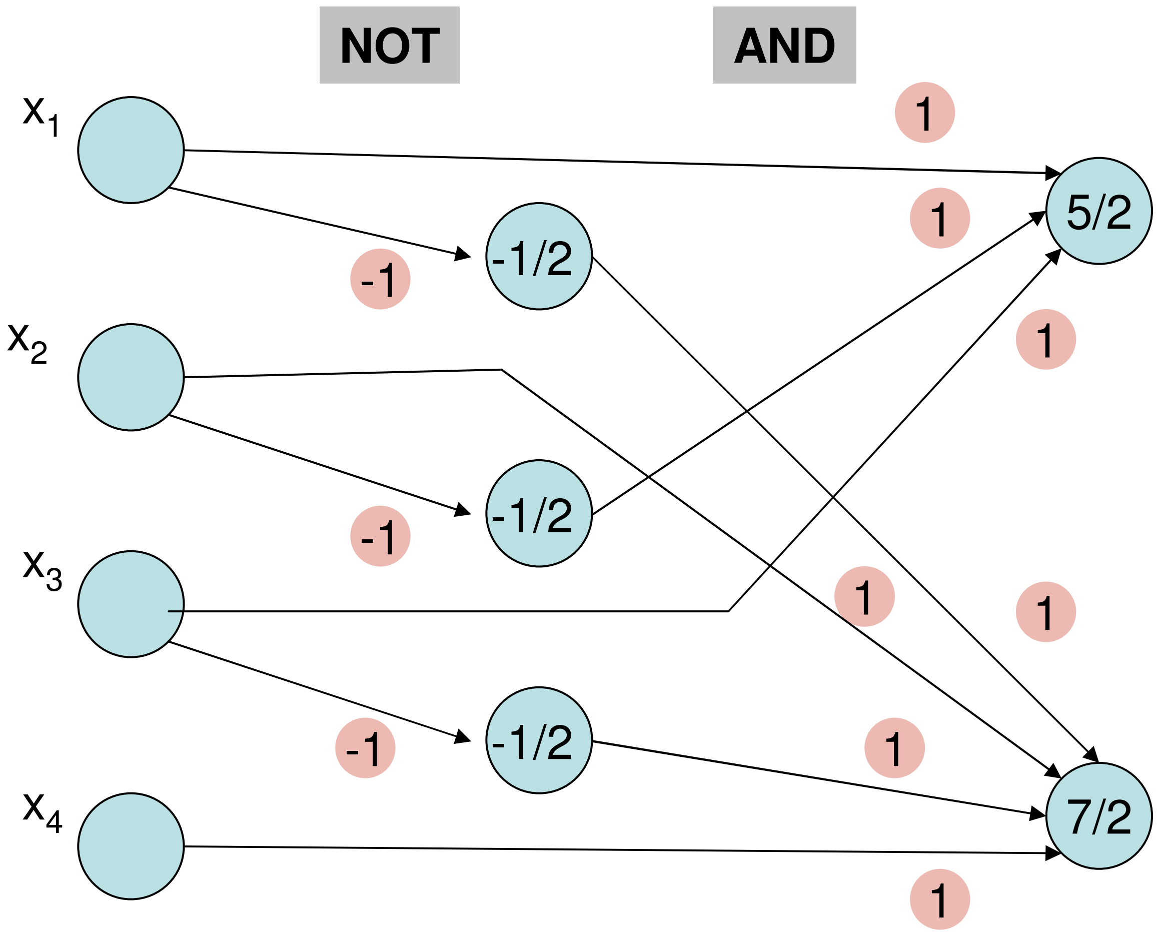
Podemos modelar cualquier expresión lógica

$$f(x_1, x_2, x_3, x_4) = x_1 x_2' x_3 + x_1' x_2 x_3' x_4$$



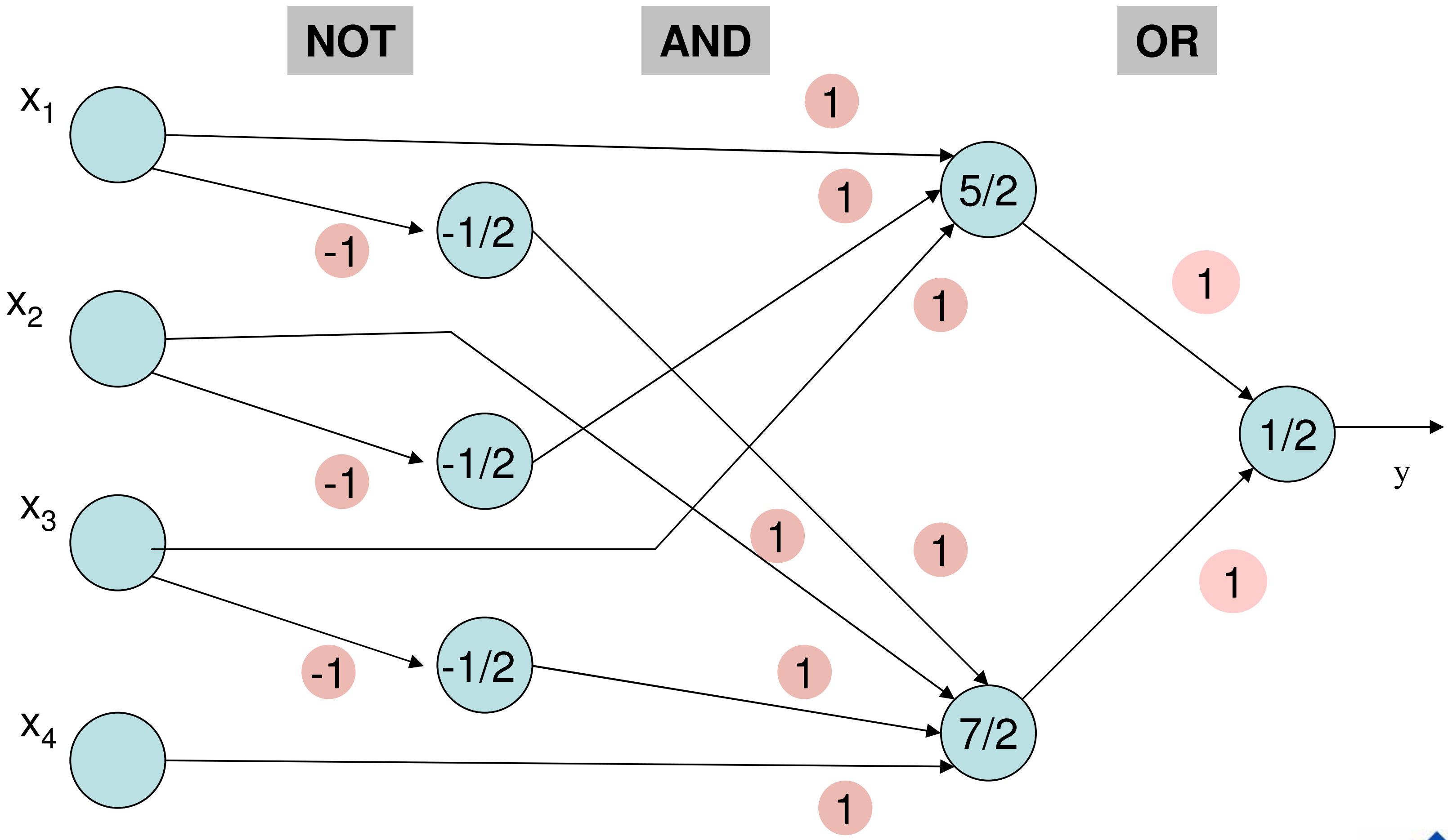
Podemos modelar cualquier expresión lógica

$$f(x_1, x_2, x_3, x_4) = x_1 x_2' x_3 + x_1' x_2 x_3' x_4$$



Podemos modelar cualquier expresión lógica

$$f(x_1, x_2, x_3, x_4) = x_1 x_2' x_3 + x_1' x_2 x_3' x_4$$



Redes lineales adaptativas

- El perceptrón, con su aprendizaje basado en ejemplos, es capaz de realizar tareas de clasificación
- Debido a su función de transferencia tipo escalón, sólo codifica salidas binarias
- Si las salidas fueran números reales, el perceptrón podía resolver problemas más generales
- Además, la regla de aprendizaje del perceptrón no “mide” el error en la salida

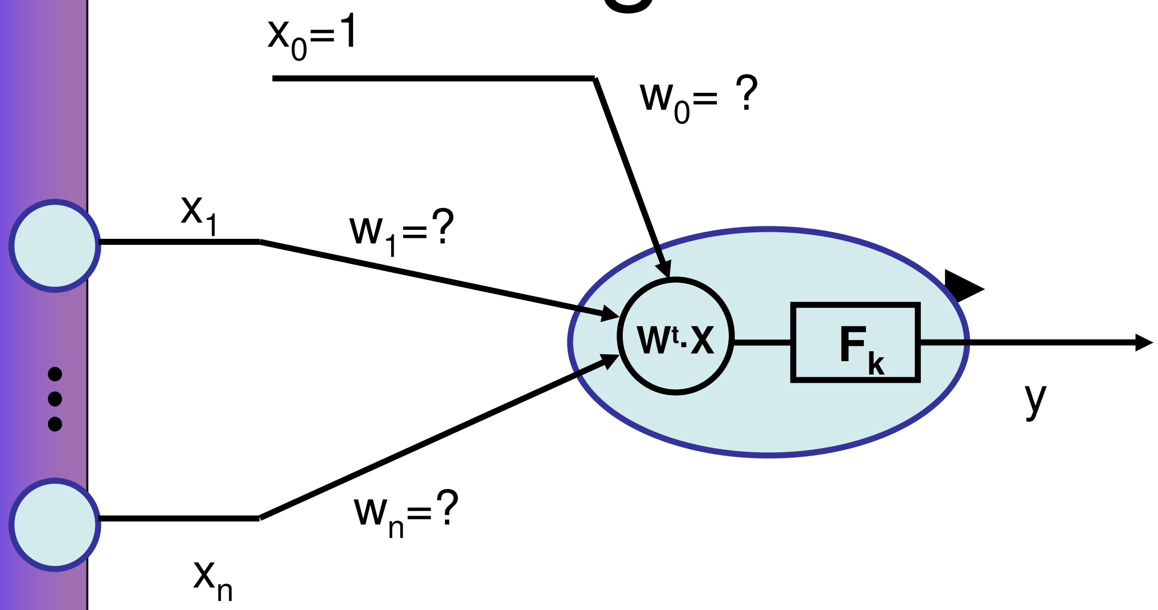
Redes lineales adaptativas

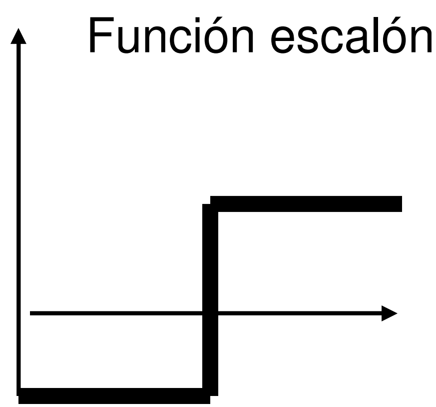
- Widrow-Hoff (1960) proponen un sistema de aprendizaje que tiene en cuenta el error producido
- Adaline (**Ada**ptive **L**inear **Ne**uron)
  - La estructura es idéntica al perceptrón, recibe un conjunto de entradas y las combina para producir una salida
- El aprendizaje, basado en el algoritmo LMS (Least Mean Square) incluye una ponderación de la diferencia:

$|d^p - y^p|$ d^p salida esperada

y^p valor actual del output

Asignación del error

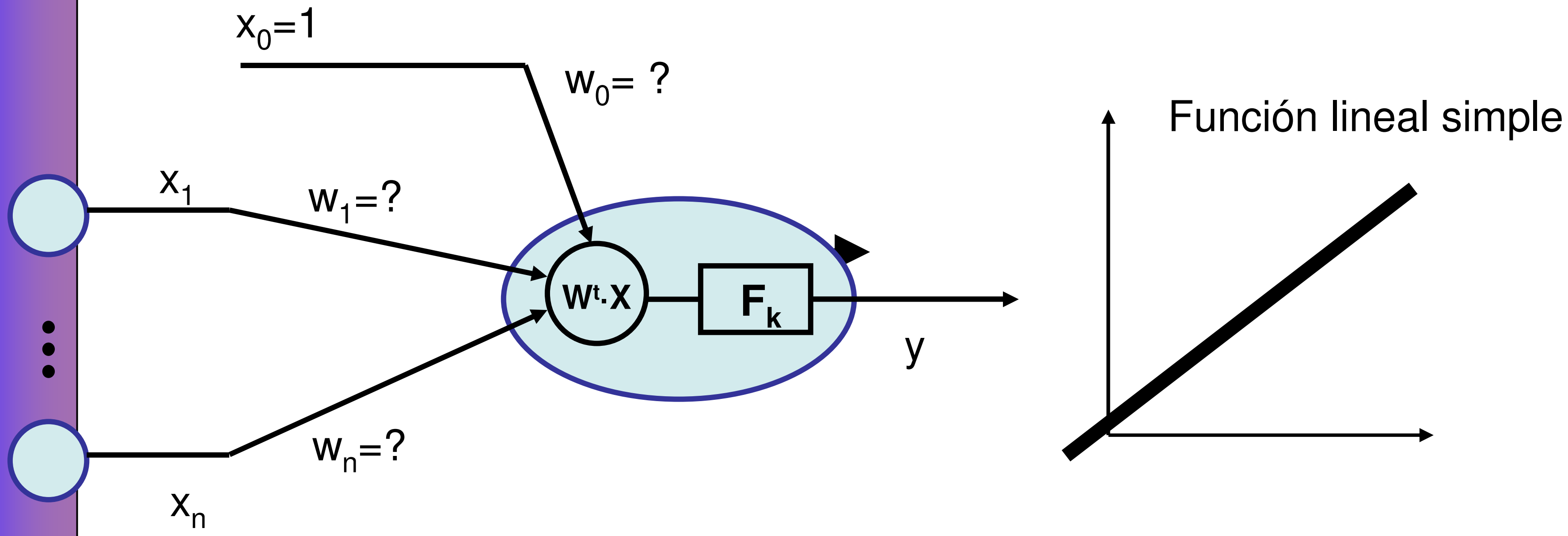




- A partir del output, resulta difícil decir que contribuye al error
- Esto dificulta el aprendizaje en redes con varias capas

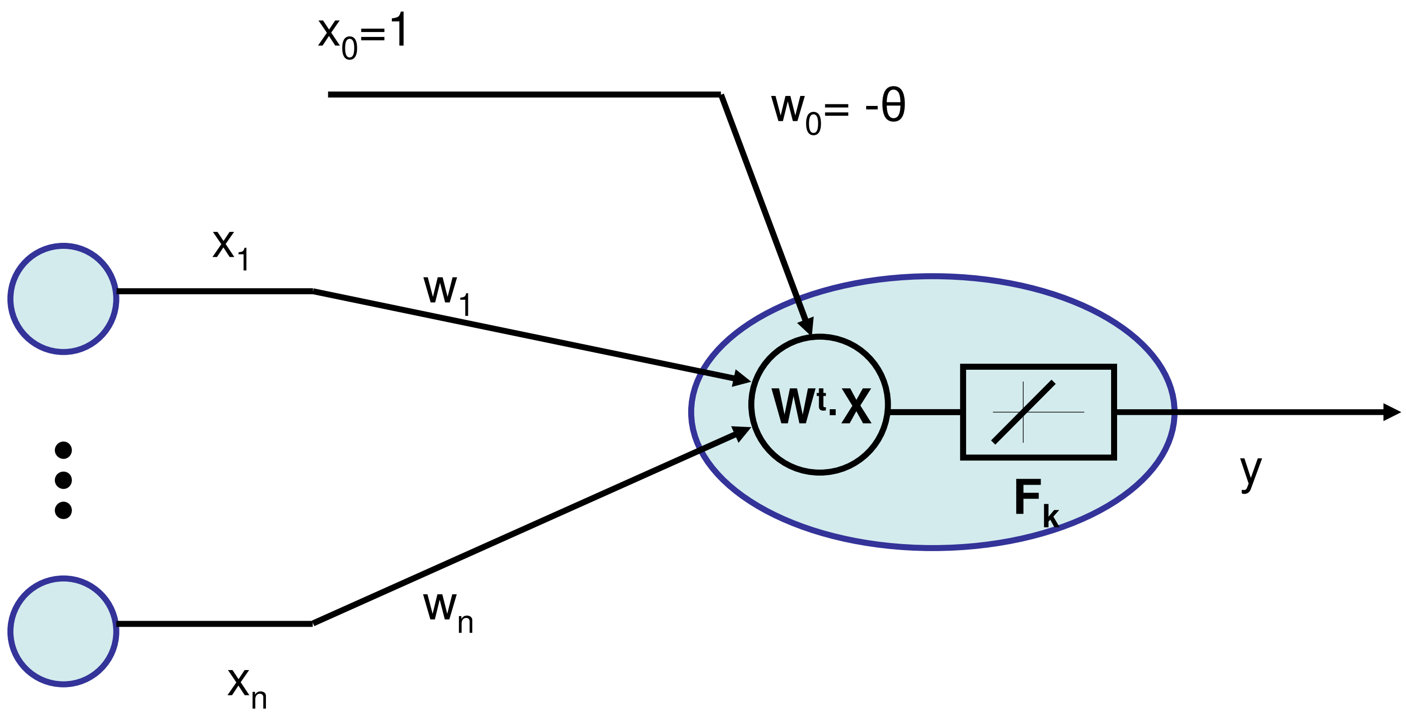
$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + \lambda(d^k - y^k)\mathbf{X}^k$$

Solución: función continua



- Una variación pequeña en el input crea un cambio perceptible en el output

Adaline



Adaline

- La diferencia está en la manera de utilizar la salida. La regla Delta utiliza directamente la salida de la red, sin pasarla por ninguna función de transferencia (o pasándola por la función de transferencia lineal: $y = W^t \cdot X$)
- El objetivo es obtener los w_i, $i = 0, 1, 2, \ldots, n$ para los que $y^z = d^z$ para el vector de entrada $\mathbf{X_i}$ que minimiza la desviación producida por la red para la totalidad de q registros del conjunto de entrenamiento

Adaline

- Se usa como medida global del error, el error cuadrático medio

$$E = \sum_{k=1}^{q} E^k = \frac{1}{2}\sum_{k=1}^{q}(d^k - y^k)^2$$

donde q es el número de neuronas de la última capa

Adaline tiene un único output

$$E = \frac{1}{2}(d - y)^2$$

- Se desea minimizar E para todos los registros del conjunto de entrenamiento

Aprendizaje del adaline

- Objetivo: determinar los valores de **W** (pesos y umbral) que clasifican los datos en dos clases.
- Entrenamiento: el adaline se expone a un conjunto de ejemplos de entrenamiento y el vector peso es ajustado de tal manera que al final del entrenamiento se obtienen las salidas esperadas para cada uno de los ejemplos
- Conclusión: Dada una nueva entrada, éste es clasificado correctamente.

Regla Delta para Adaline

- Dado el conjunto de entrenamiento y salidas deseadas

$$[\mathbf{X}^z, d^z], \quad z = 1, 2, \ldots k$$

- Buscamos el vector peso **W** que minimice la función del error para cada registro

$$E = \frac{1}{2}(d - y)^2 = \frac{1}{2}(d - \underbrace{F(\sum_{i=0}^{n} w_i x_i)}_{\text{F es la identidad}})^2 = \frac{1}{2}(d - \sum_{i=0}^{n} w_i x_i)^2$$

Regla Delta para Adaline

- Lo hacemos por la regla del descenso del gradiente, en cada iteración modificamos el vector pesos

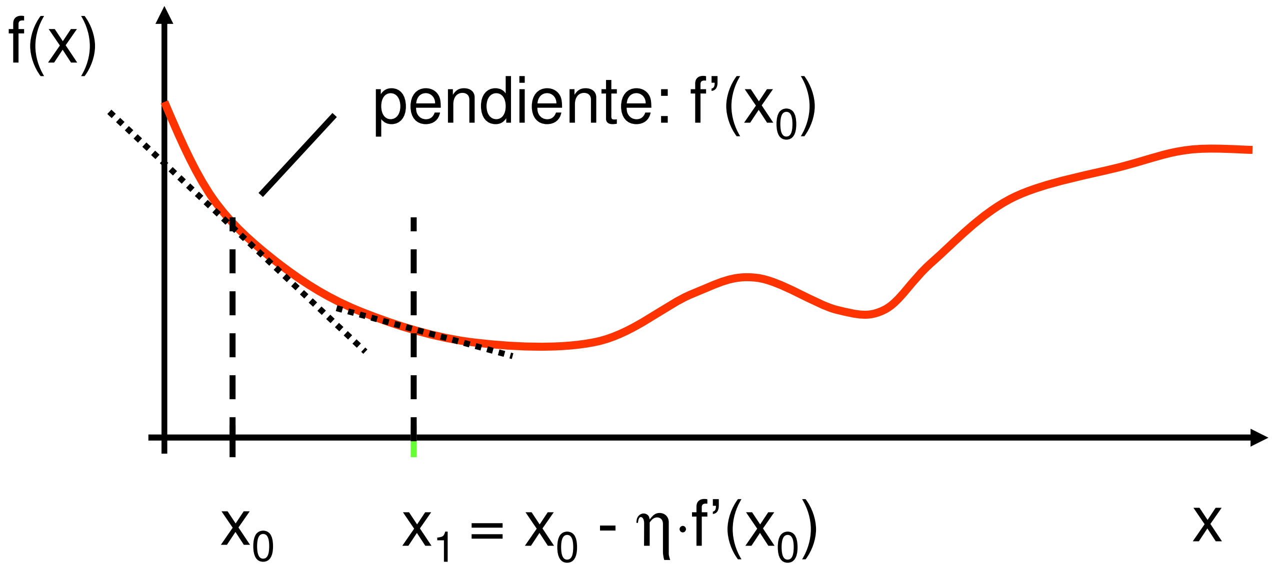
$$W^{(k+1)} = W^{(k)} + \alpha \cdot \Delta W$$

donde ΔW es el opuesto del vector gradiente de la función de error en $W^{(k)}$

El cambio en cada peso es proporcional a la derivada del error (registro actual) respecto del peso

Descenso del gradiente

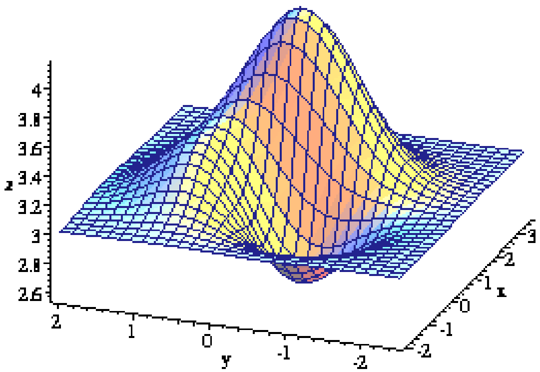
Encontrar el mínimo de la función de error f(x)

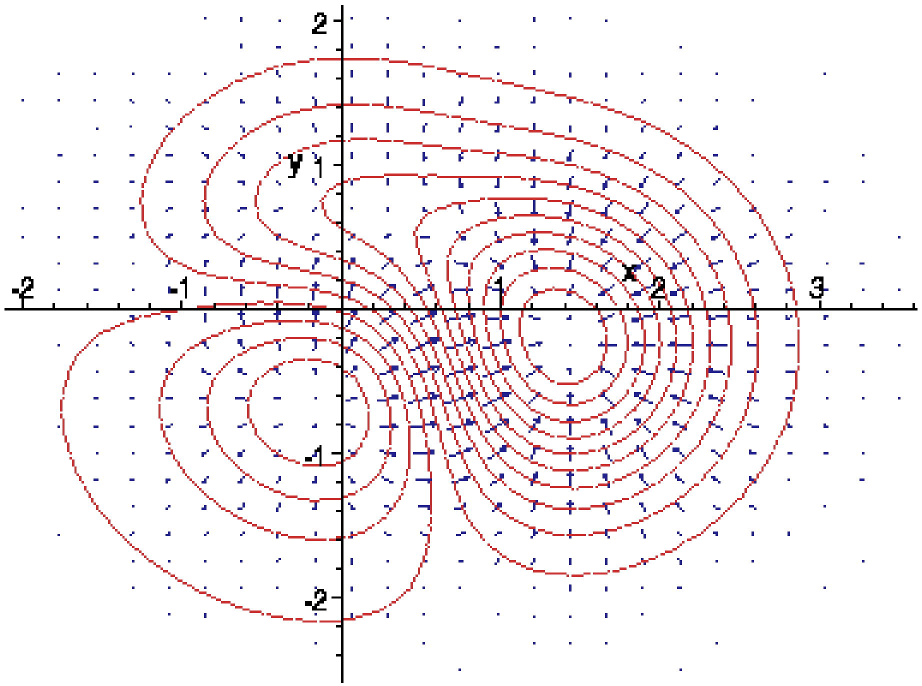


Repetir iterativamente hasta que para x_i, $f'(x_i)$ está suficientemente próxima a 0

Descenso del gradiente

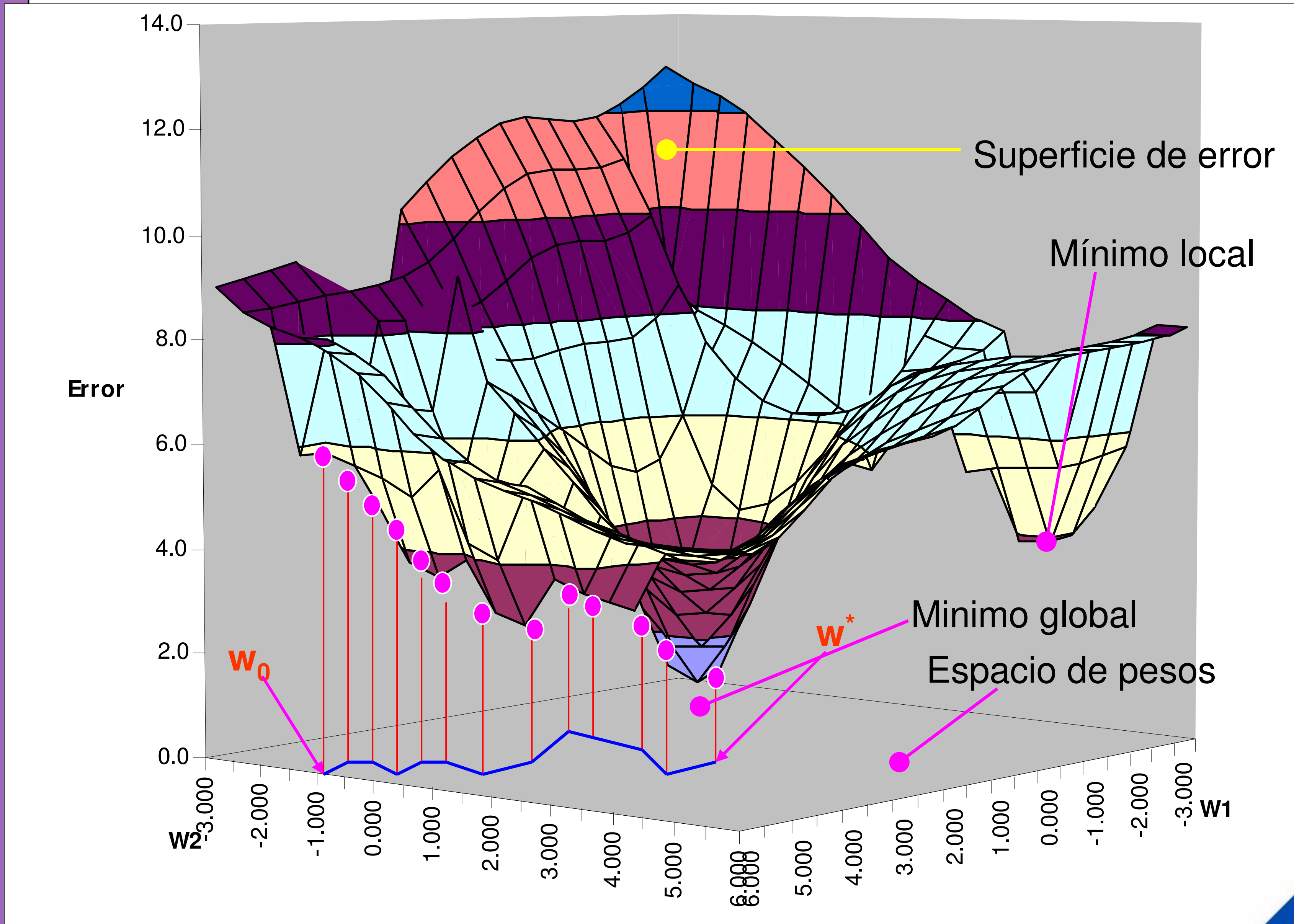
En dos dimensiones





A la derecha están las líneas de contorno, las flechas indican el gradiente de la función en distintos puntos. El gradiente siempre señala la dirección de pendiente máxima de la función. Para encontrar el mínimo hay que mover en contra del gradiente

Superficie de error



Regla Delta para Adaline

- El vector gradiente de la función de error es

$$\frac{dE}{dW} = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_n} \right)^t$$

donde

$$\frac{\partial E}{\partial w_i} = -(d - y) \cdot x_i$$

los nuevos pesos son

$$w_i^{(k+1)} = w_i^{(k)} + \alpha \cdot (d - y) \cdot x_i$$

Regla Delta para Adaline

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\left(\frac{1}{2}(d - \sum_{i=0}^{n} w_i x_i)^2\right)$$

$$= -(d - y)\frac{\partial(\sum_{i=1}^{n} w_i \cdot x_i)}{\partial w_i} = -(d - y)\cdot x_i$$

$$\Delta w_i = (d - y)\cdot x_i$$

Algoritmo de aprendizaje

La corrección del error se realiza por el método de descenso del gradiente

- Se asignan valores arbitrarios a los pesos (pequeños)

  $\mathbf{W}^{(1)} = [w_0 = -\theta, w_1, w_2, \ldots w_n]^t$

- Repetimos para cada ejemplo del conjunto de entrenamiento $[\mathbf{X}^z, d^z]$, hasta que todos los registros queden clasificados correctamente por el mismo vector peso
  - Presentación del ejemplo $[\mathbf{X}, d]$
  - Calculamos la salida actual y y lo comparamos con la deseada, calculamos d - y
  - Para todos los pesos, multiplicamos la diferencia por la entrada correspondiente, ponderamos
  - Adaptamos los pesos

α = factor de aprendizaje

$$w_i^{(k+1)} = w_i^{(k)} + \alpha \cdot (d - y) \cdot x_i$$

Ejemplo

- Decodificador de binario a decimal, tomar $\alpha = 0,3$

x_1	x_2	x_3	d
0	0	1	**1**
0	1	0	**2**
0	1	1	**3**
1	0	0	**4**
1	0	1	**5**
1	1	0	**6**
1	1	1	**7**

$$w_i^{(k+1)} = w_i^{(k)} + \alpha \cdot x_i \cdot (d - y)$$

1ª iteración

					Peso inicial							Peso final		
	x_1	x_2	x_3	d	w_1	w_2	w_3	y	$\alpha \cdot \Delta w_1$	$\alpha \cdot \Delta w_2$	$\alpha \cdot \Delta w_3$	w_1	w_2	w_3
1	0	0	1	**1**	0,84	0,394	0,783							
	0	1	0	**2**										
	0	1	1	**3**										
	1	0	0	**4**										
	1	0	1	**5**										
	1	1	0	**6**										
	1	1	1	**7**										

1ª iteración

$$w_i^{(k+1)} = w_i^{(k)} + \alpha \cdot x_i \cdot (d - y)$$

					Peso inicial							Peso final		
	x_1	x_2	x_3	d	w_1	w_2	w_3	y	$\alpha \cdot \Delta w_1$	$\alpha \cdot \Delta w_2$	$\alpha \cdot \Delta w_3$	w_1	w_2	w_3
1	0	0	1	**1**	0,84	0,394	0,783	0,783	0	0	0,065	0,84	0,394	0,848
	0	1	0	**2**										
	0	1	1	**3**										
	1	0	0	**4**										
	1	0	1	**5**										
	1	1	0	**6**										
	1	1	1	**7**										

$0{,}3 \cdot 1 \cdot (1-0{,}783) = 0{,}065$

1ª iteración

$$w_i^{(k+1)} = w_i^{(k)} + \alpha \cdot x_i \cdot (d - y)$$

					Peso inicial							Peso final		
	x_1	x_2	x_3	d	w_1	w_2	w_3	y	$\alpha \cdot \Delta w_1$	$\alpha \cdot \Delta w_2$	$\alpha \cdot \Delta w_3$	w_1	w_2	w_3
1	0	0	1	**1**	0,84	0,394	0,783	0,783	0	0	0,065	0,84	0,394	0,848
	0	1	0	**2**	0,84	0,394	0,848	0,394	0	0,482	0	0,84	0,876	0,848
	0	1	1	**3**										
	1	0	0	**4**										
	1	0	1	**5**										
	1	1	0	**6**										
	1	1	1	**7**										

0,3(2-0,394)1=0,482

1ª iteración

$$w_i^{(k+1)} = w_i^{(k)} + \alpha \cdot x_i \cdot (d-y)$$

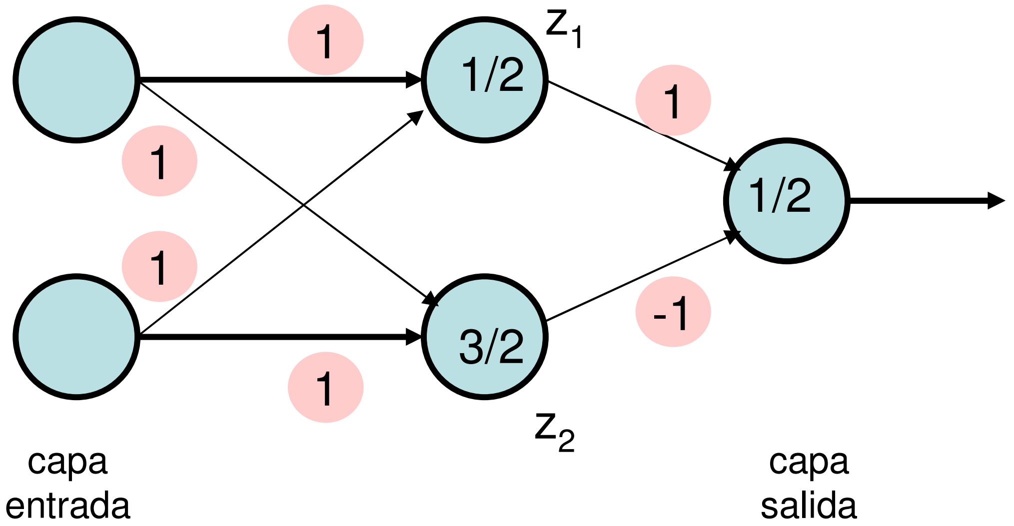
					Peso inicial							Peso final		
	x_1	x_2	x_3	d	w_1	w_2	w_3	y	$\alpha \cdot \Delta w_1$	$\alpha \cdot \Delta w_2$	$\alpha \cdot \Delta w_3$	w_1	w_2	w_3
1	0	0	1	**1**	0,84	0,394	0,783	0,783	0	0	0,065	0,84	0,394	0,848
	0	1	0	**2**	0,84	0,394	0,848	0,394	0	0,482	0	0,84	0,876	0,848
	0	1	1	**3**										
	1	0	0	**4**										
	1	0	1	**5**										
	1	1	0	**6**								3,090	1,966	1,825
	1	1	1	**7**	3,09 0	1,966	1,825	6,881	0,036	0,036	0,036	3,126	2,002	1,861

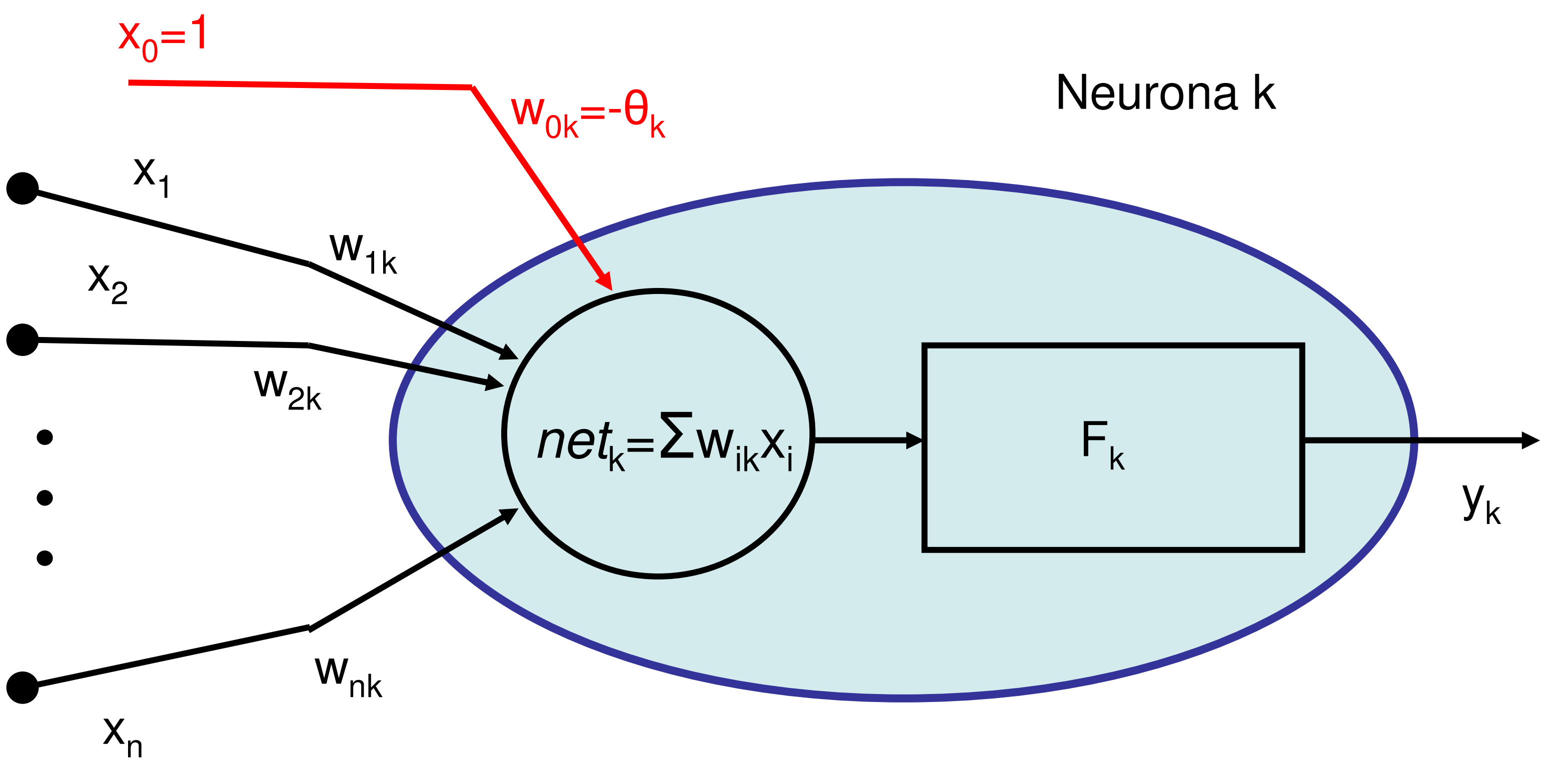
0,3(7-6,881)1=0,036

Perceptrón multicapa

- Hemos visto como el perceptrón multicapa permite clasificar datos no separados linealmente

XOR





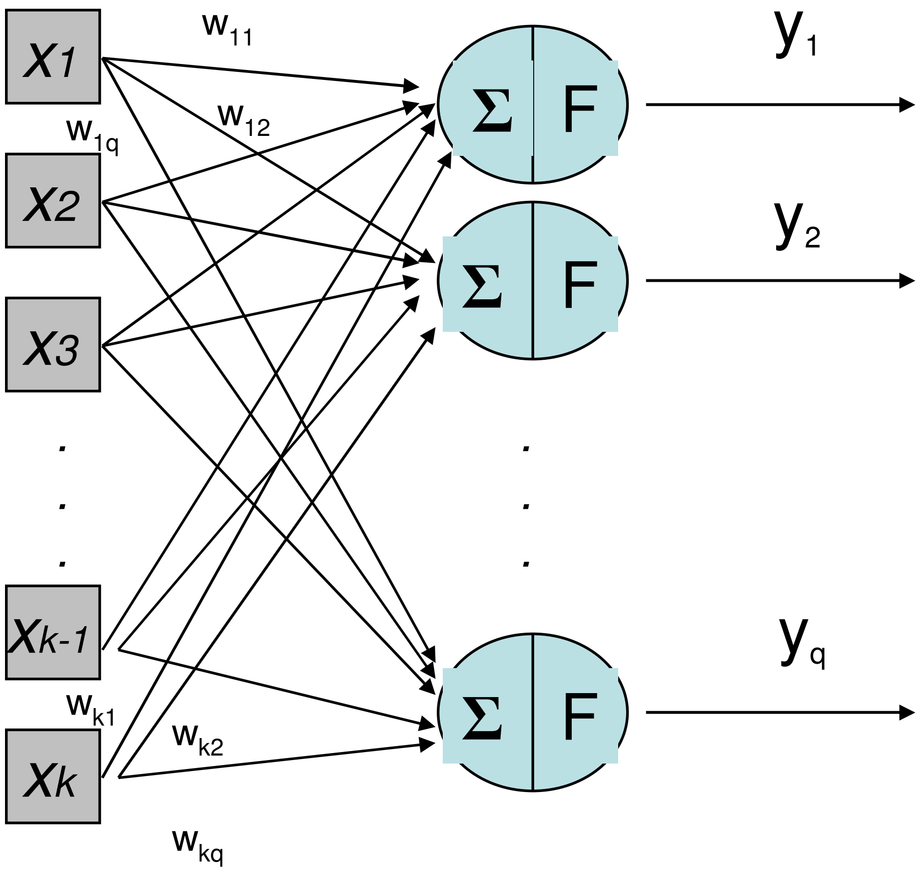
$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad ; \quad W_k = \begin{bmatrix} w_{0k} \\ w_{1k} \\ w_{2k} \\ \vdots \\ w_{nk} \end{bmatrix}$$

F_k compara $\mathbf{W_k}^{\mathbf{t}} \cdot \mathbf{X}$ con 0

$y_k = F_k(\mathbf{W_k}^{\mathbf{t}} \cdot \mathbf{X})$

Red neuronal artificial

Red neuronal de una capa con múltiples entradas y salidas



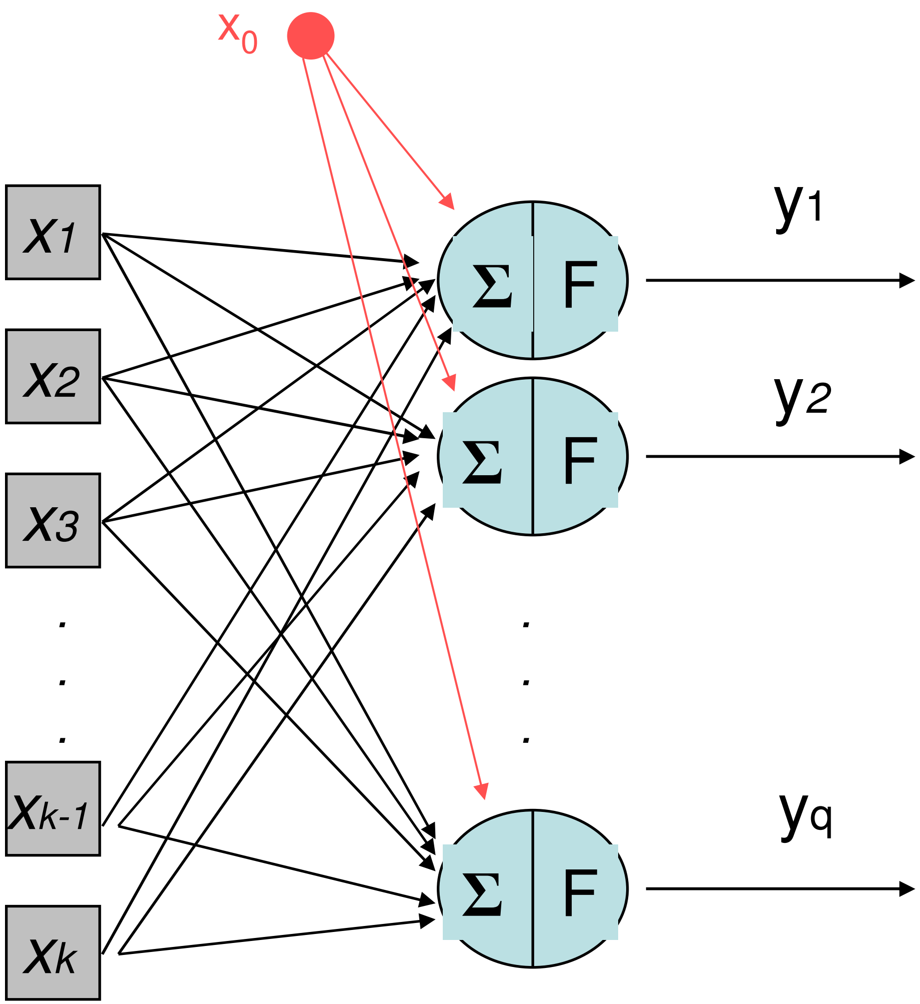
$$y_h = F_h(\Sigma w_{ih} \cdot x_i)$$

misma F para todas las neuronas de la capa

$$\mathbf{Y} = F(\mathbf{W}^t \cdot \mathbf{X})$$

Matriz de pesos $\mathbf{W = [W_1, W_2, \ldots W_q]}$

Red neuronal artificial



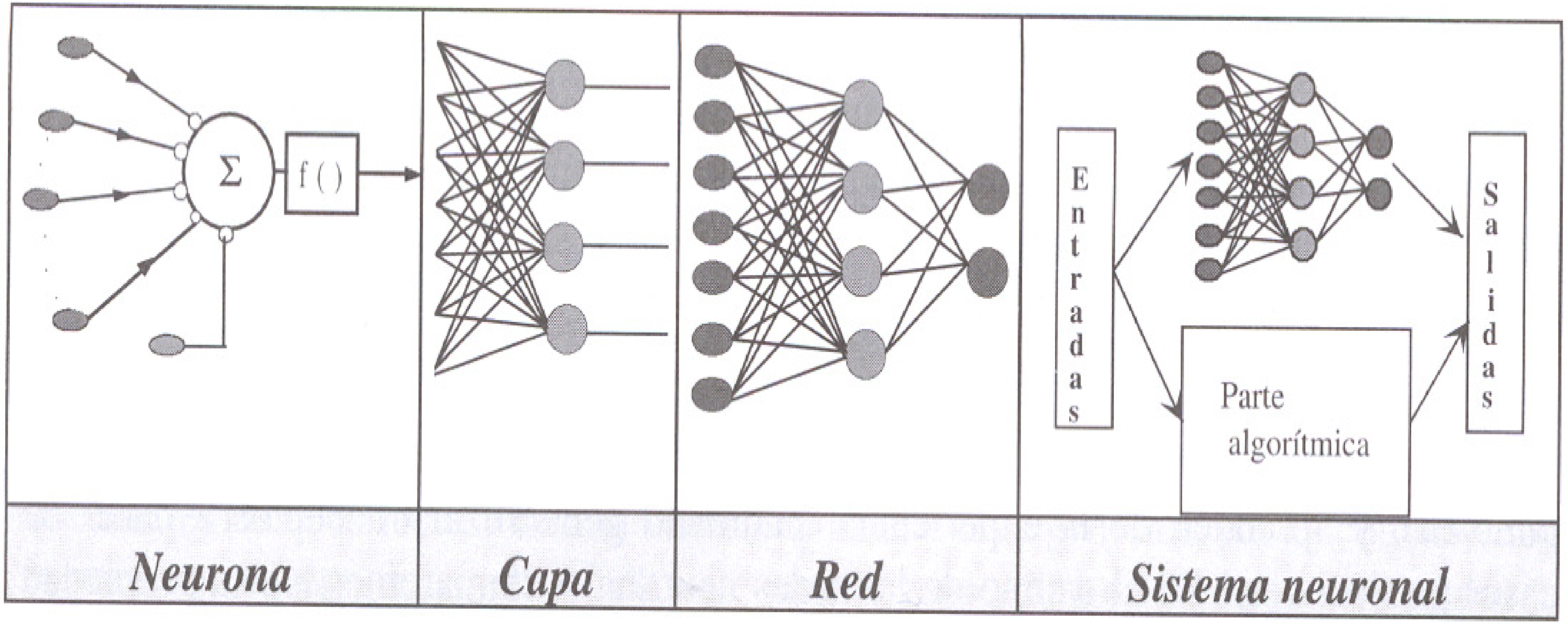
$$y_h = F_h(\Sigma w_{ih} \cdot x_i)$$

misma F para todas las neuronas de la capa

$$\mathbf{Y} = F(\mathbf{W}^t \cdot \mathbf{X})$$

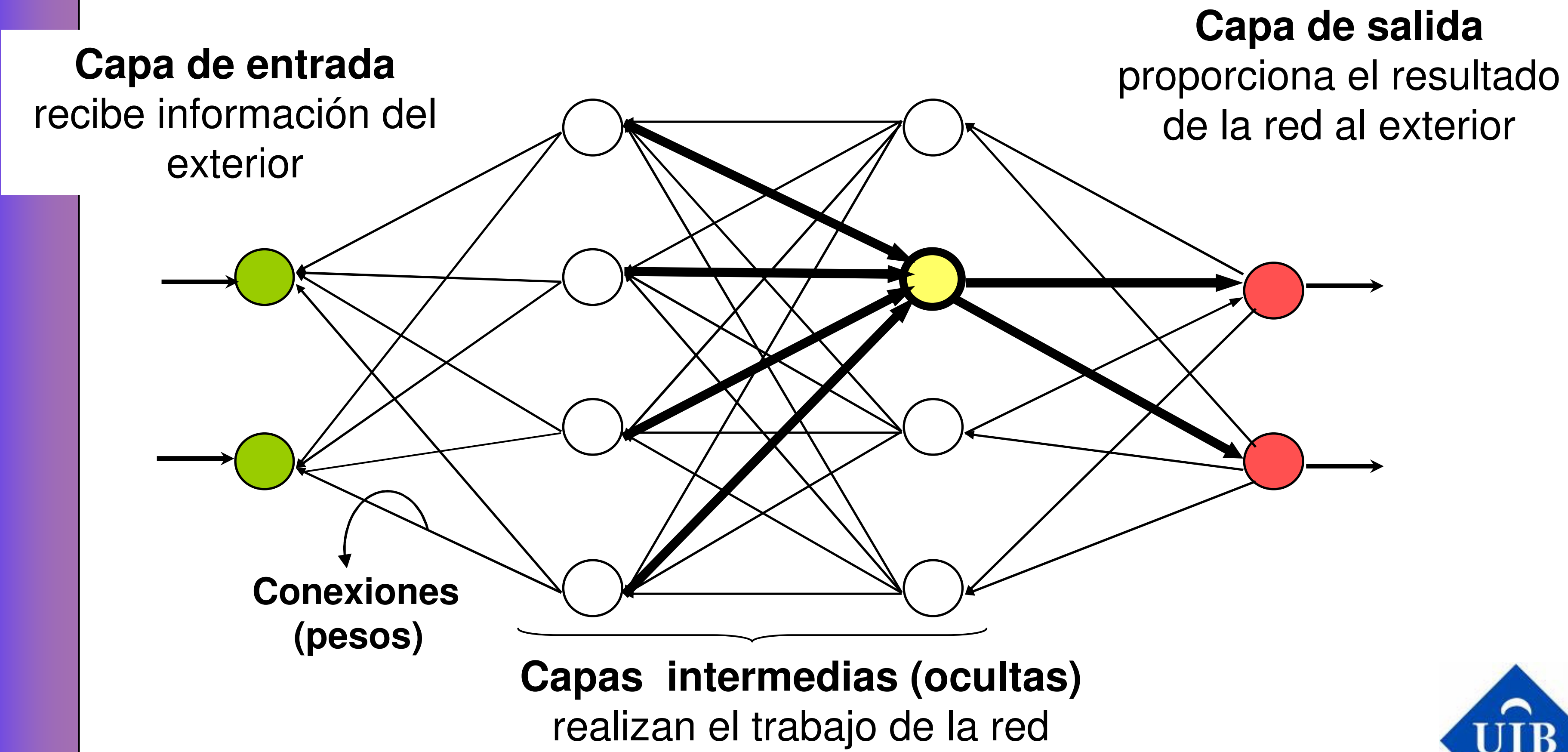
Matriz de pesos $\mathbf{W} = [\mathbf{W_1}, \mathbf{W_2}, \dots \mathbf{W_q}]$

Sistemas neuronales



Red neuronal artificial

- La estructura de la red neuronal puede ser compleja
- RNA organizadas en capas, feedforward, totalmente conectadas

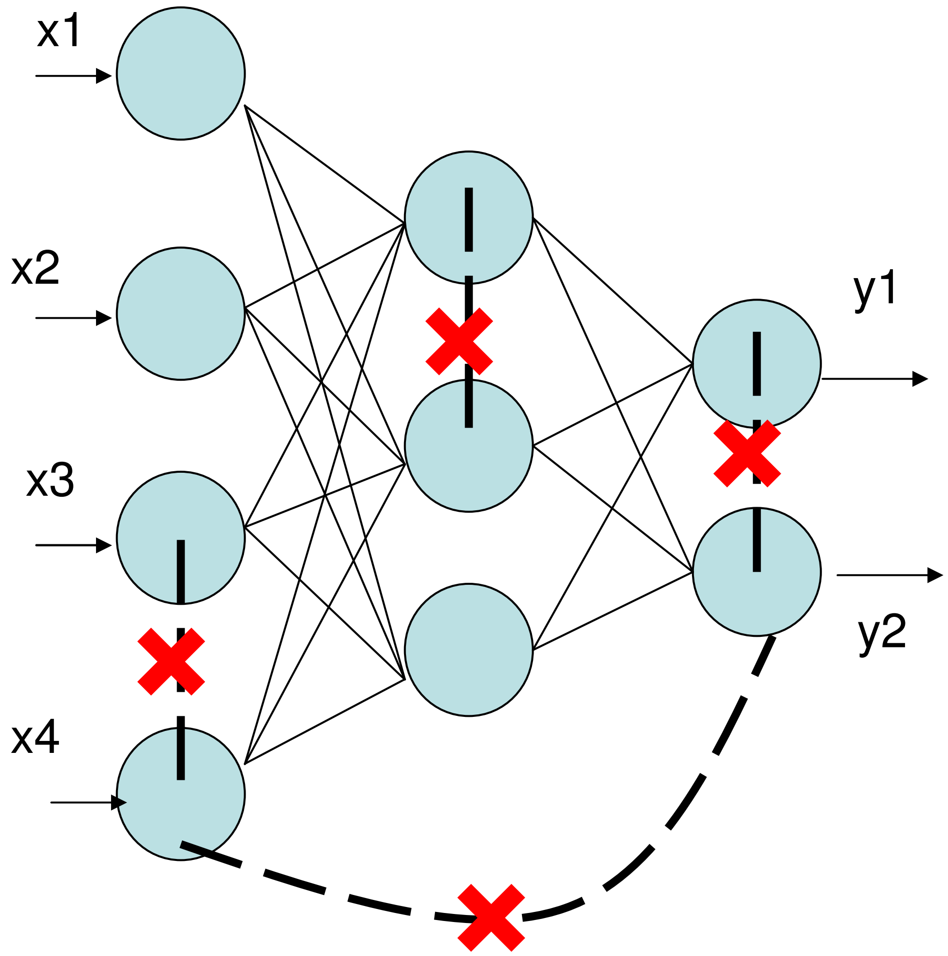


Red neuronal artificial

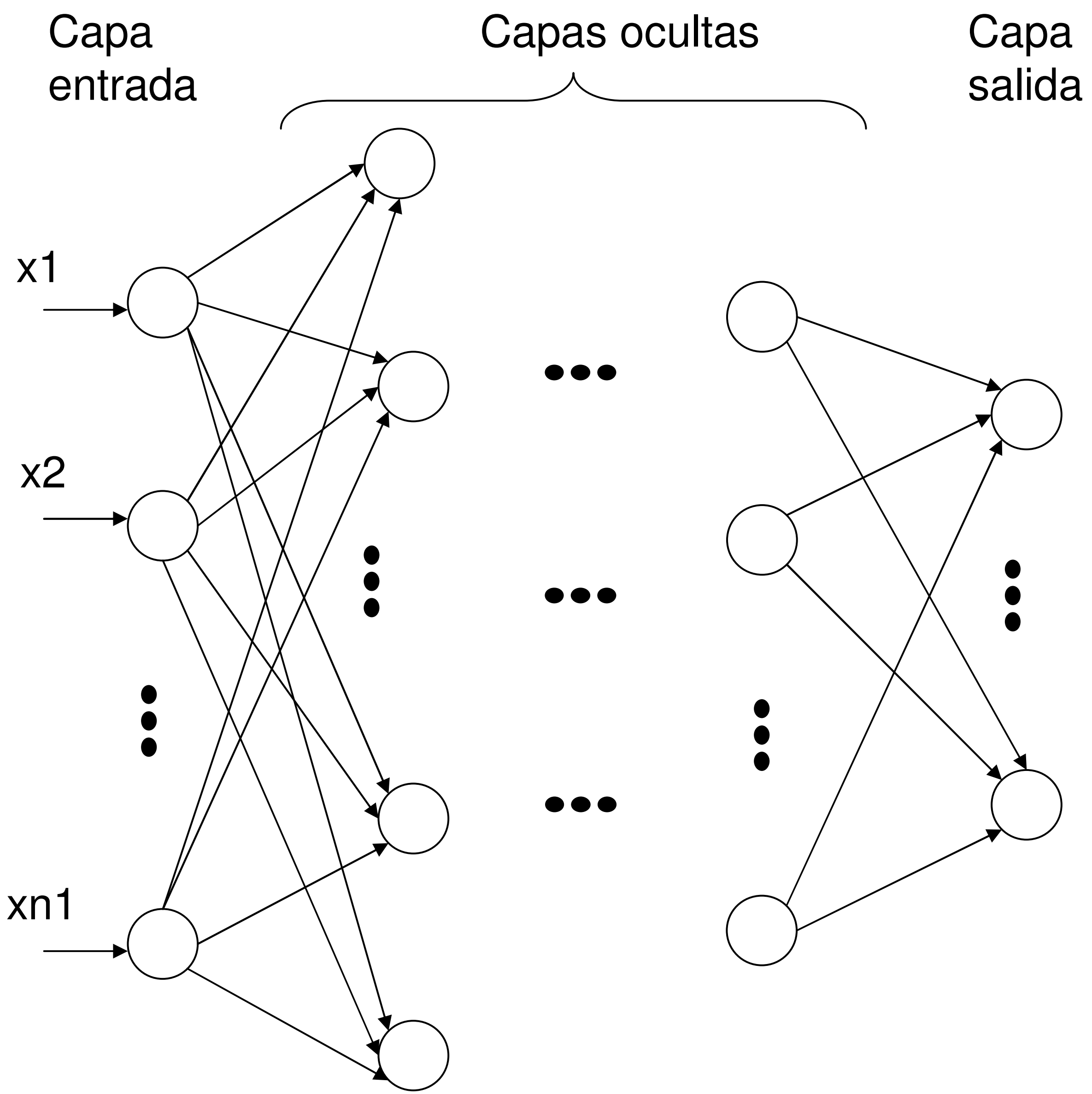
Arquitectura de la red

- Sin conexiones en la misma capa
- Sin conexiones directas entre capa de entrada y salida
- Las neuronas de una capa sólo están conectadas con las neuronas de la siguiente capa (feed forward)
- Completamente conectada entre capas
- Generalmente más de 3 capas
- El número de neuronas de la capa de salida puede ser diferente al número de neuronas de la capa de entrada
- El número de neuronas por capa oculta no tiene que coincidir con el número de neuronas de la capa de entrada, ni de salida
- El número de capas ocultas y el número de neuronas de cada capa depende del problema a resolver

Red neuronal artificial

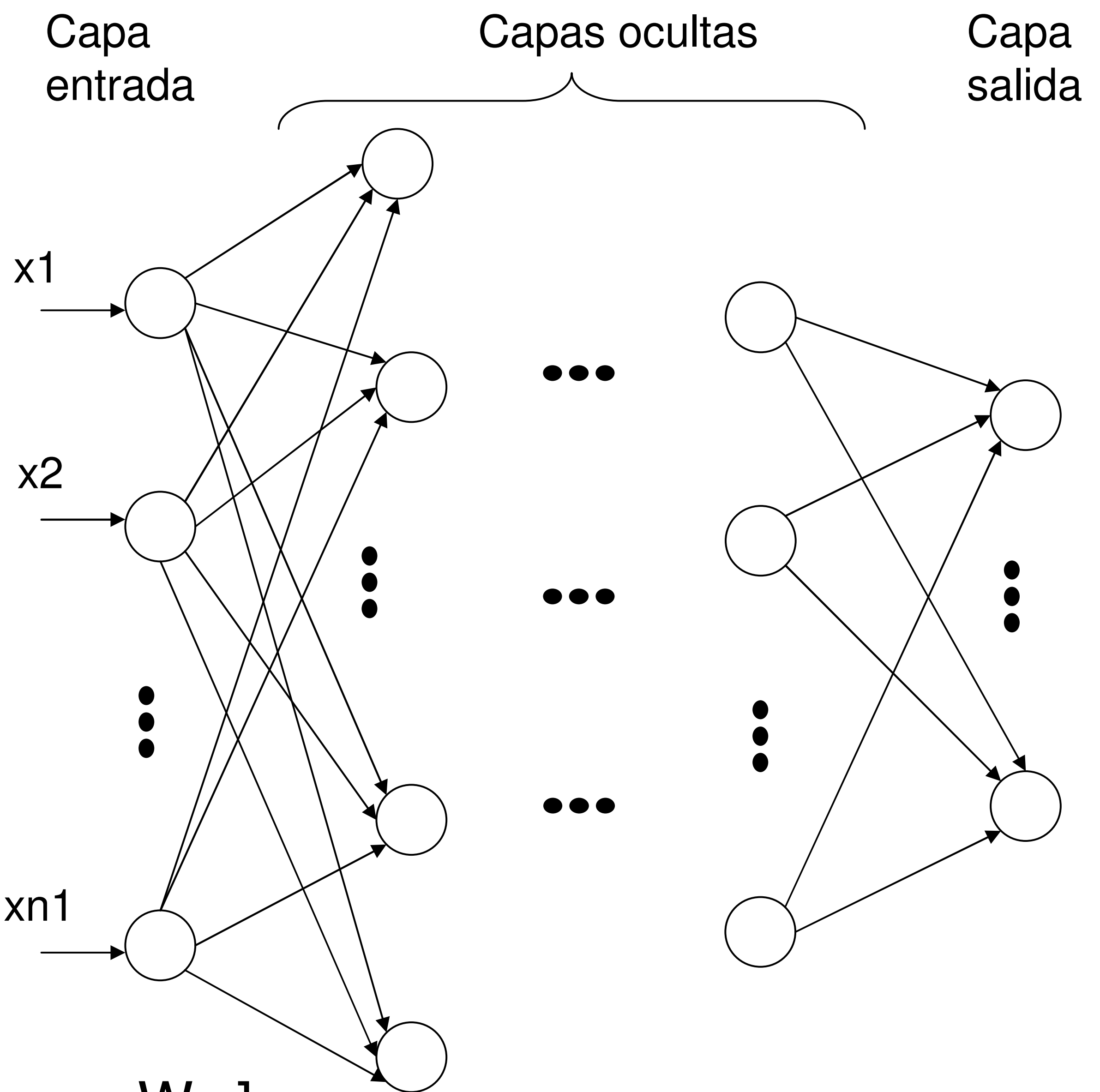


Red neuronal de varias capa con múltiples entradas y salidas



Red neuronal de varias capa con múltiples entradas y salidas

$2 \leftarrow 1$



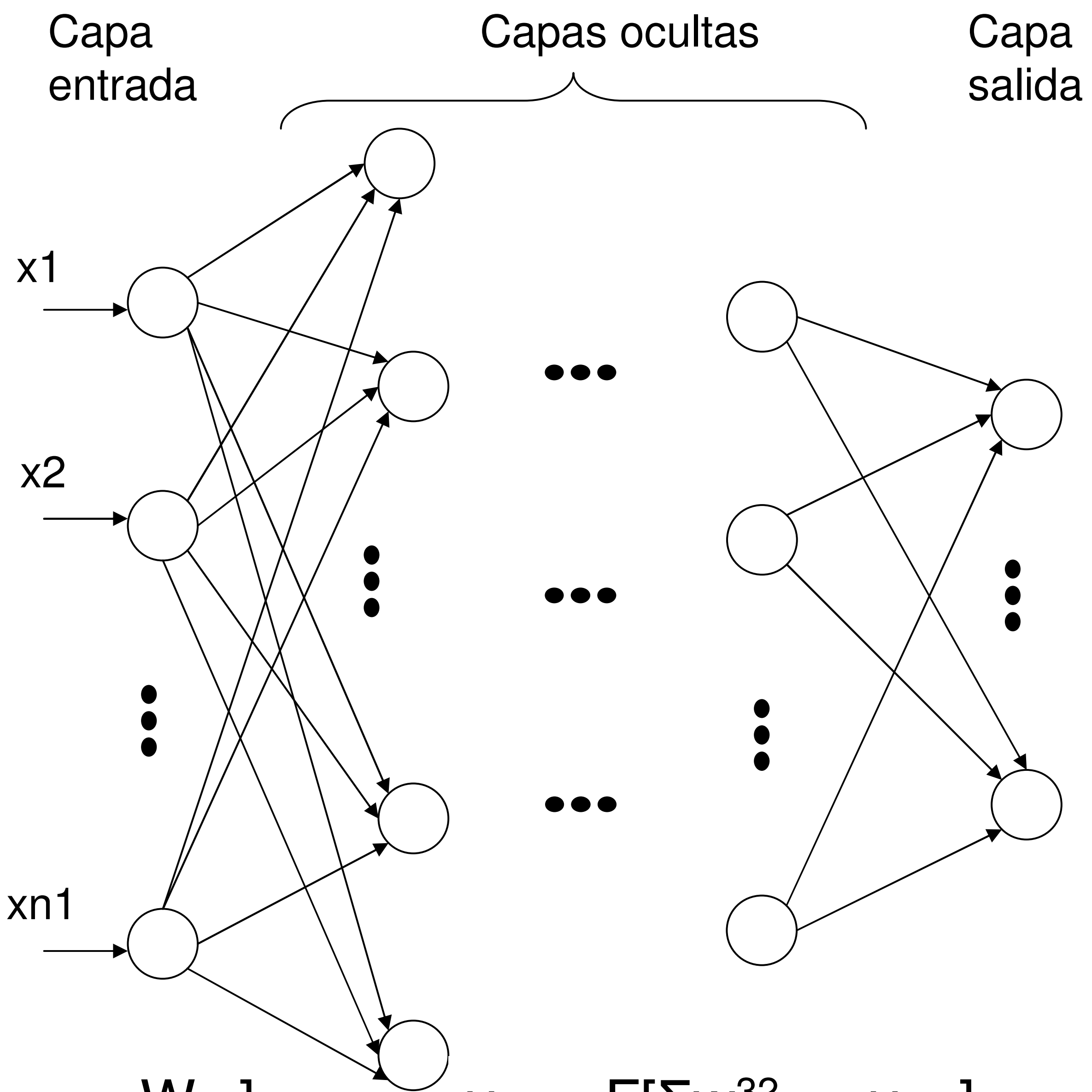
$W^{21} = [W_1, W_2, \ldots, W_{n1}]$

$\mathbf{Y_1} = F(\mathbf{W^{21\,t}} \cdot \mathbf{X_1})$

$y_{1h} = F(\Sigma w^{21}{}_{hi} \cdot x_{1i})$

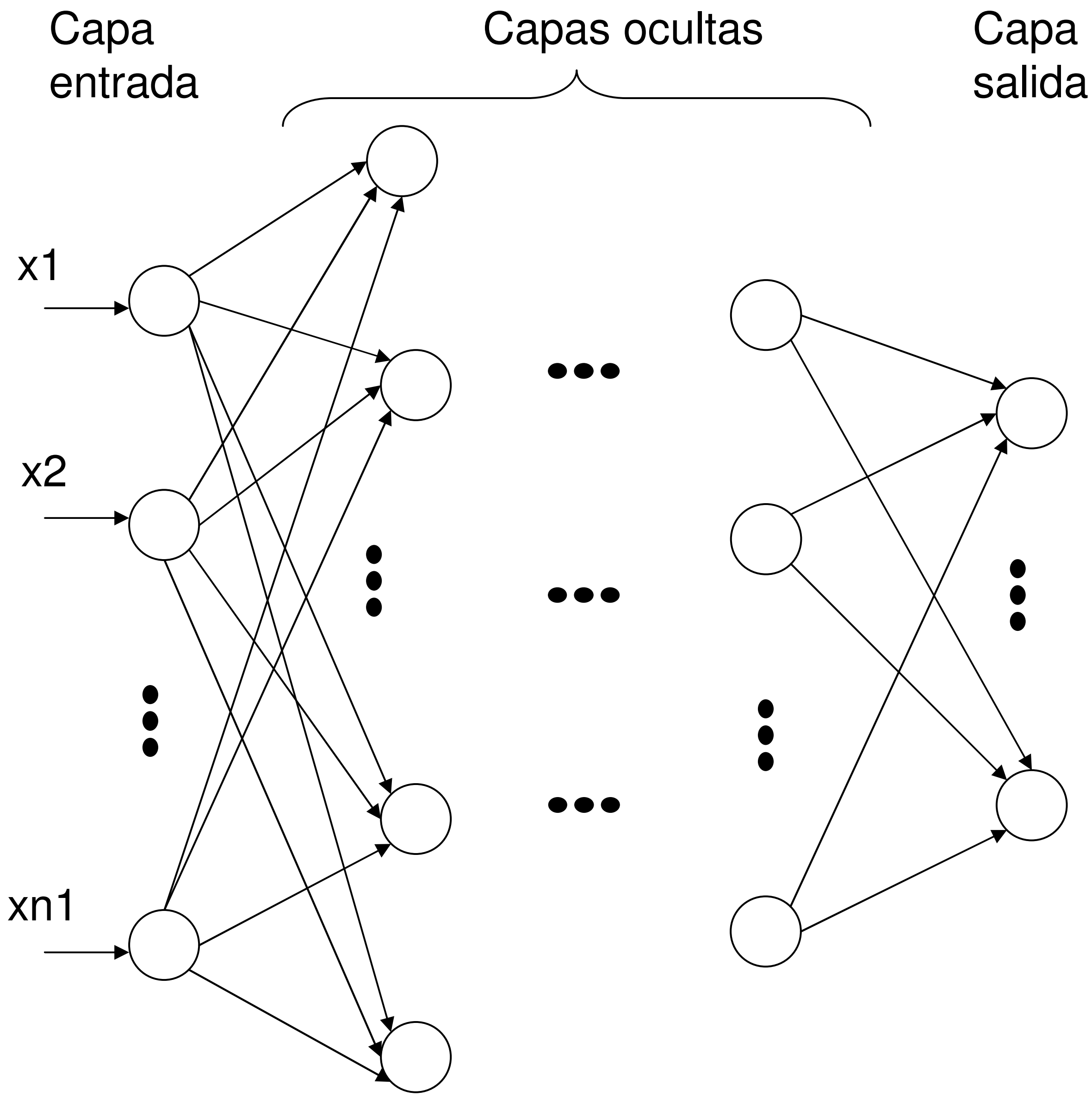
Red neuronal de varias capa con múltiples entradas y salidas

$3 \leftarrow 2$



$$W^{32} = [W_1, W_2, \ldots, W_{n2}] \qquad y_{2k} = F[\Sigma w^{32}_{hk} \cdot y_{1h}]$$

$$\mathbf{Y_2} = F(\mathbf{W^{32\,t}} \cdot \mathbf{X_2}) = F(\mathbf{W^{32\,t}} \cdot \mathbf{Y_1}) \qquad = F[\Sigma w^{32}_{hk} \cdot (F(\Sigma w^{21}_{hi} \cdot x_{1i}))]$$



Hasta llegar a la última capa

Función de transferencia derivable

- Función sigmoidal

$$F_S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

- Función tangente hiperbólica

$$F_H(x) = \frac{1-e^{-x}}{1+e^{-x}} = \frac{e^x-1}{e^x+1}$$

$$F_H(x) = 2F_S(x) - 1$$

Función de transferencia derivable

- Derivada de la función sigmoidal

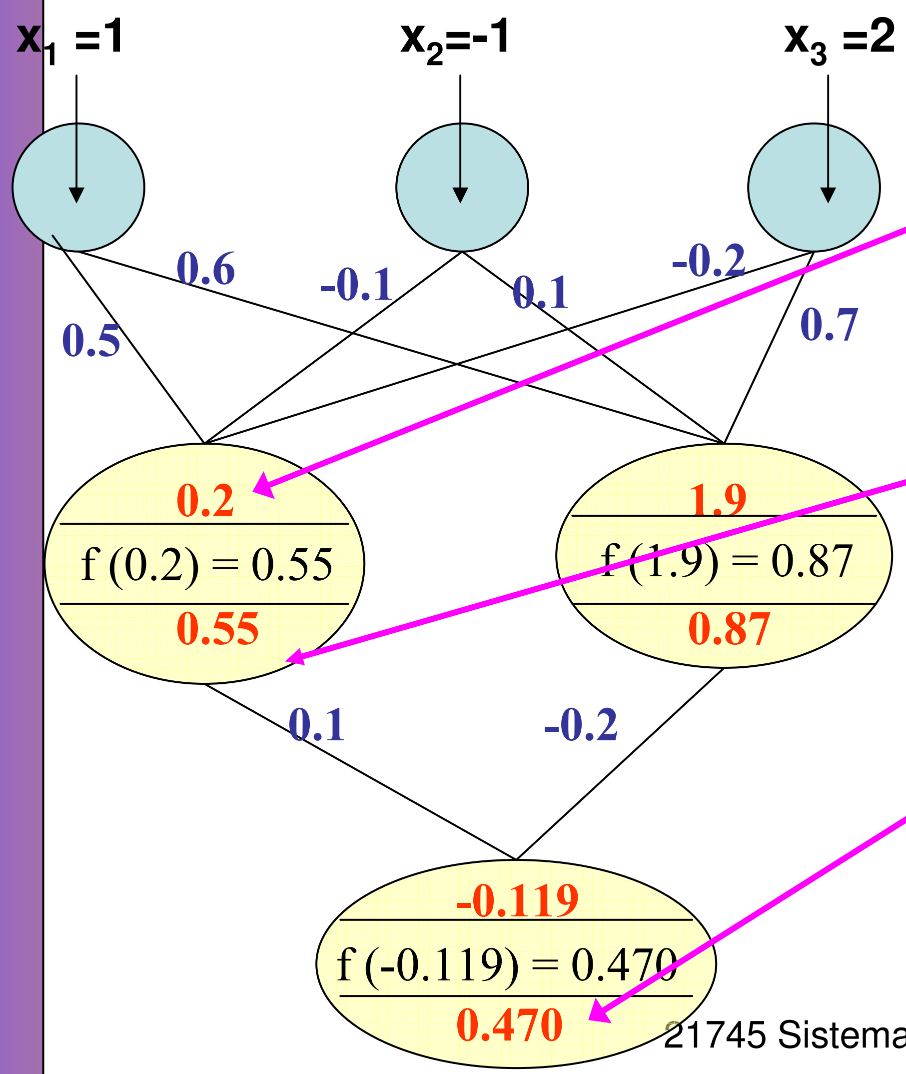
$$F'_S(x) = F_S(x)(1 - F_S(x))$$

- Derivada de la función tangente hiperbólica

$$F'_H(x) = 1 - F_H(x)^2$$

Predicción de la salida para la red neuronal de la figura, usar la función de transferencia sigmoidal

Input: $x_1\ x_2\ x_3$ **Output:** y **Model:** $y = f(x_1\ x_2\ x_3)$



$0.2 = 0.5 * 1 - 0.1*(-1) - 0.2 * 2$

$f(x) = e^x / (1 + e^x)$
$f(0.2) = e^{0.2} / (1 + e^{0.2}) = 0.55$

salida y = 0.470

si d = 2
entonces el error en la predicción es
E = (2-0.47) =1.53

- ¿Cómo mejoramos la red?
- Aprendizaje basado en regla delta
- Buscar pesos que minimicen la función de error

$$E=\frac{1}{2}(d-y)^2=\frac{1}{2}(d-F\left[\sum_{i=0}^{n} w_i x_i\right])^2$$

$$W^{(k+1)}=W^{(k)}+\alpha\cdot\Delta W$$

Regla Delta para Adaline

- El vector gradiente de la función de error es

$$\frac{dE}{dW} = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_n} \right)^t$$

$$y = F\left(\sum_i w_i \cdot x_i \right) = \sum_i w_i \cdot x_i$$

$$\frac{\partial E}{\partial w_i} = (d - y) \frac{\partial (d - y)}{\partial w_i} =$$

$$= -(d - y) \frac{\partial \left(\sum_{i=1}^{n} w_i \cdot x_i \right)}{\partial w_i} = -(d - y) \cdot x_i$$

Regla Delta generalizada

$$\frac{\partial E}{\partial w_i} = (d-y)\frac{\partial (d-y)}{\partial w_i} =$$

$$F(g(x_i))$$

$$= -(d-y)\frac{\partial (y)}{\partial w_i} = -(d-y)\frac{\partial \left(F(\sum_i x_i w_i) \right)}{\partial w_i}$$

Para derivar aplicamos la regla de la cadena

F(g(x_0))' = F'(g(x_0))g'(x_0)

Regla Delta generalizada

$$\frac{\partial\,[F(\sum_{i=1}^{n} x_i w_i)]}{\partial w_i} = \frac{\partial F(g(x_i))}{\partial g} \cdot \frac{\partial g}{\partial w_i} =$$

$$= \frac{\partial F(g(x_i))}{\partial g} \cdot x_i = F'(g) \cdot x_i$$

$$\frac{\partial E}{\partial w_i} = -(d - y) \cdot F'(\sum) \cdot x_i$$

$$\Delta w_i = (d - y) F'(\sum) x_i$$

Si usamos la función sigmoidal como función de transferencia

$$\Delta w_i = (d-y)F_S(\Sigma)(1-F_S(\Sigma))x_i$$

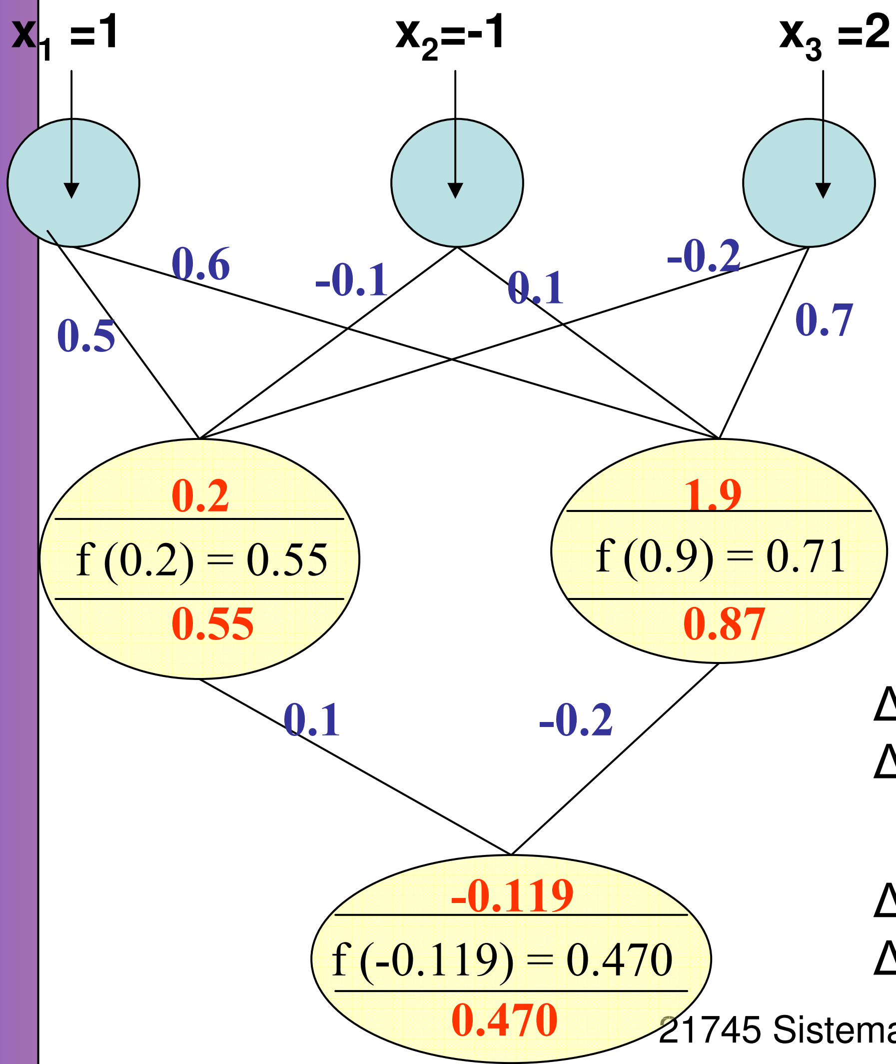
$$w_i^{(k+1)} = w_i^{(k)} + \alpha(d-y)F_S(\Sigma)(1-F_S(\Sigma))x_i$$

donde

$$\Sigma = \sum_i x_i w_i = net\,y$$

Adaptación de pesos usando la regla delta generalizada y función de transferencia sigmoidal

Input: $x_1\ x_2\ x_3$ **Output:** y **Model:** $y = f(x_1\ x_2\ x_3)$



$\Delta w_1 = (2\text{-}0{,}47)F(-0{,}119)([1-F(-0{,}119)]0{,}55$
$\Delta w_1 = 0{,}210$

$\Delta w_2 = (2\text{-}0{,}47)F(-0{,}119)([1-F(-0{,}119)]0{,}87$
$\Delta w_2 = 0{,}332$

- Usamos la regla delta generalizada para modificar los pesos que llevan al output

$$w_i^{(k+1)} = w_i^{(k)} + \alpha(d-y)F_S(\Sigma)(1-F_S(\Sigma))x_i$$

$$w_1^{(2)} = 0{,}1 + 0{,}5(2-0{,}47)F_S(-0{,}119)(1-F_S(-0{,}119))0{,}55 =$$

$$= 0{,}1 + 0{,}5 \cdot 1{,}53 \cdot 0{,}470 \cdot 0{,}53 \cdot 0{,}55 = 0{,}205$$

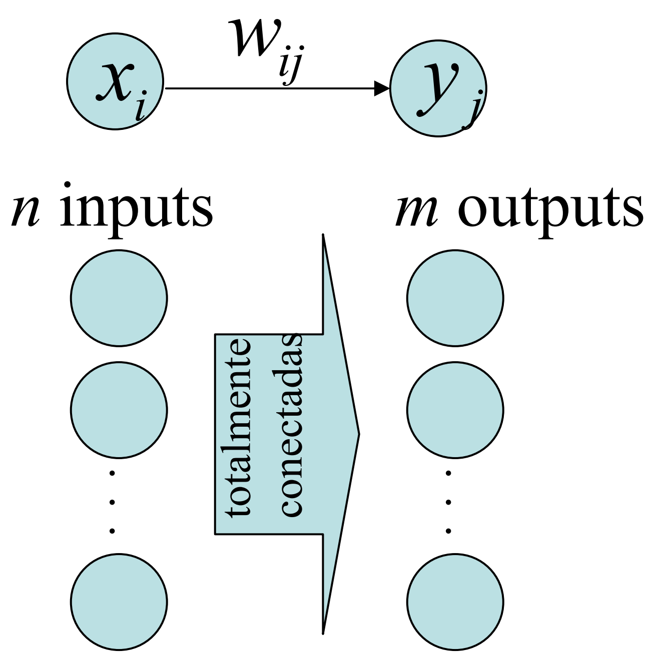
$$w_2^{(2)} = -0{,}2 + 0{,}5(2-0{,}47)F_S(-0{,}119)(1-F_S(-0{,}119))0{,}87 =$$

$$= -0{,}2 + 0{,}5 \cdot 1{,}53 \cdot 0{,}470 \cdot 0{,}53 \cdot 0{,}87 = -0{,}034$$

Extensión a múltiples salidas

- Hasta ahora sólo hemos visto una única neurona en la capa de salida
- Generalizamos para múltiples neurona en la capa de salida

$$\Delta w_{ij} = (d_j - y_j) x_i F'(net\, y_j)$$



w_{ij} peso desde input i a output j

d_j salida deseada de output j

y_j salida actual desde output j

net y_j suma ponderada a output j

x_i entrada i

$$net\, y_j = \sum_{i=1}^{n} x_i w_{ij}$$ entes

Extensión a múltiples salidas

$$x_i \xrightarrow{w_{ij}} y_j$$

$$y_j = F\left(\sum_{i=0}^{n} x_i w_{ij}\right)$$

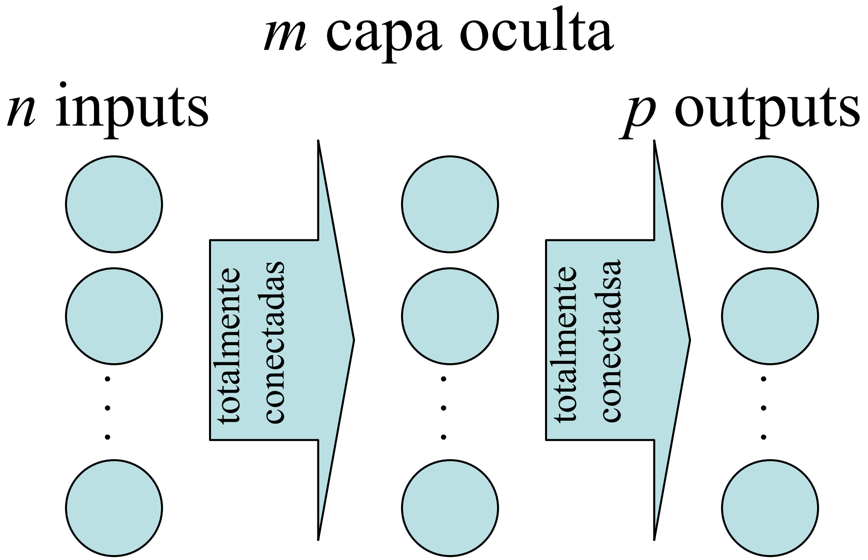
$$E = \frac{1}{2}\sum_{j=1}^{m}(d_j - y_j)^2 = \frac{1}{2}\sum_{j=1}^{m}\left(d_j - F\left(\sum_i x_i w_{ij}\right)\right)^2$$

$$\frac{\partial E}{\partial w_{ij}} = -(d_j - y_j)\frac{\partial}{\partial w_{ij}}\left(F\left(\sum_i x_i w_{ij}\right)\right) =$$

$$= -(d_j - y_j)x_i F'(net\, y_j)$$

$$w_{ij}^{(k+1)} = w_{ij}^{(k)} + \alpha(d_j - y_j)x_i F'(net\, y_j)$$

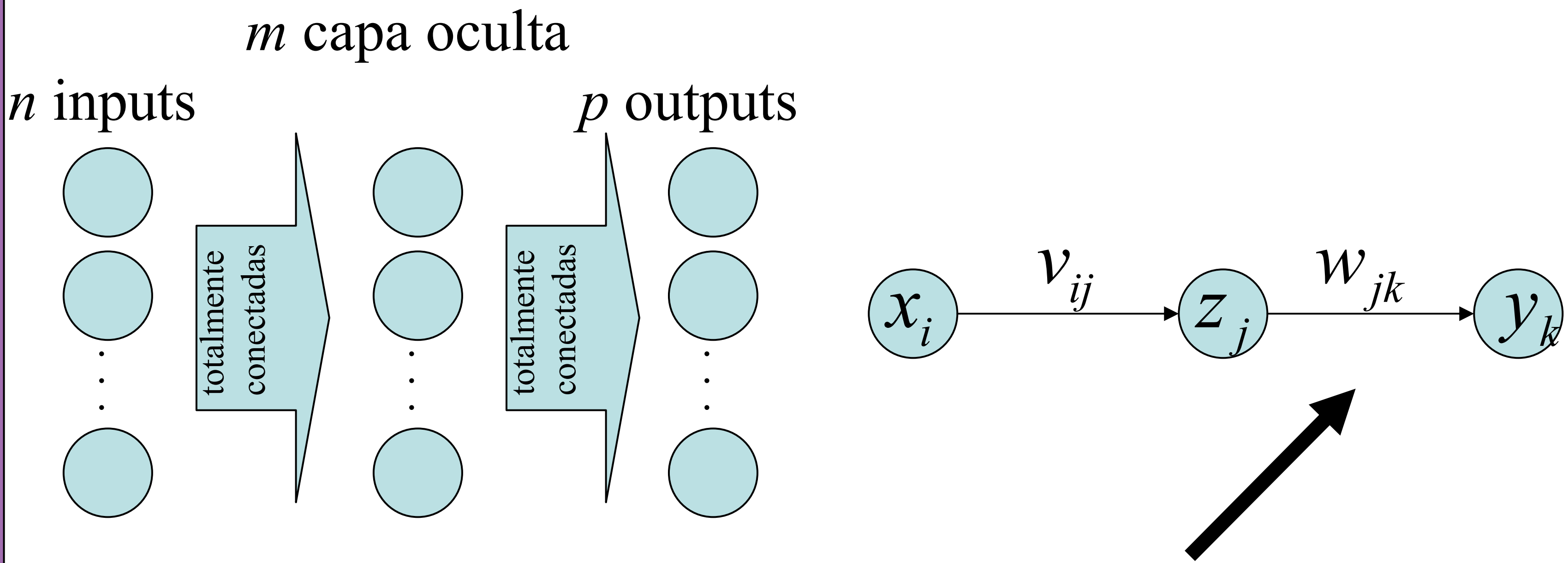
Extensión a múltiples capas



$$x_i \xrightarrow{v_{ij}} z_j \xrightarrow{w_{jk}} y_k$$

$$z_j = F\left(\sum_i x_i v_{ij}\right) \qquad y_k = F\left(\sum_j z_j w_{jk}\right)$$

Extensión a múltiples capas



- Calcular modificaciones en pesos para última capa con regla delta generalizada

$$\Delta w_{jk} = (d_k - y_k) z_j F'(net\, y_k)$$

- También hay que considerar los pesos entre capa de entrada y capa oculta

$$x_i \xrightarrow{v_{ij}} z_j \xrightarrow{w_{jk}} y_k$$

- En regla Delta generalizada usamos la derivada

$$\frac{\partial E}{\partial w_{jk}}$$

- Ahora necesitamos

$$\frac{\partial E}{\partial v_{ij}}$$

$$E = \frac{1}{2}\sum_{k=1}^{p}(d_k - y_k)^2$$

$$\frac{\partial E}{\partial v_{ij}} = -\sum_{k=1}^{p}[d_k - y_k]\frac{\partial y_k}{\partial v_{ij}} = -\sum_{k=1}^{p}[d_k - y_k]\frac{\partial F(\sum_j z_j w_{jk})}{\partial v_{ij}}$$

$$= -\sum_{k=1}^{p}\underbrace{[d_k - y_k]F'(\sum_j z_j w_{jk})}_{\delta_k}\frac{\partial \sum_j z_j w_{jk}}{\partial v_{ij}}$$

donde

$$z_j = F(\sum_i x_i v_{ij})$$

$$\frac{\partial E}{\partial v_{ij}} = -\sum_{k=1}^{p} \delta_k \frac{\partial \sum_j z_j w_{jk}}{\partial z_j} \frac{\partial z_j}{\partial v_{ij}} = -\sum_{k=1}^{p} \delta_k w_{jk} \frac{\partial z_j}{\partial v_{ij}}$$

$$z_j = F(\sum_i x_i v_{ij})$$

$$\frac{\partial E}{\partial v_{ij}} = -\sum_{k=1}^{p} \delta_k x_i F'(net z_j) w_{jk} =$$

$$= -x_i F'(net z_j) \sum_{k=1}^{p} \delta_k w_{jk}$$

$$v_{ij}^{(k+1)} = v_{ij}^{(k)} + \alpha \Delta v_{ij}$$

$$\Delta v_{ij} = x_i F'(net z_j) \sum_{k=1}^{p} \delta_k w_{jk} =$$

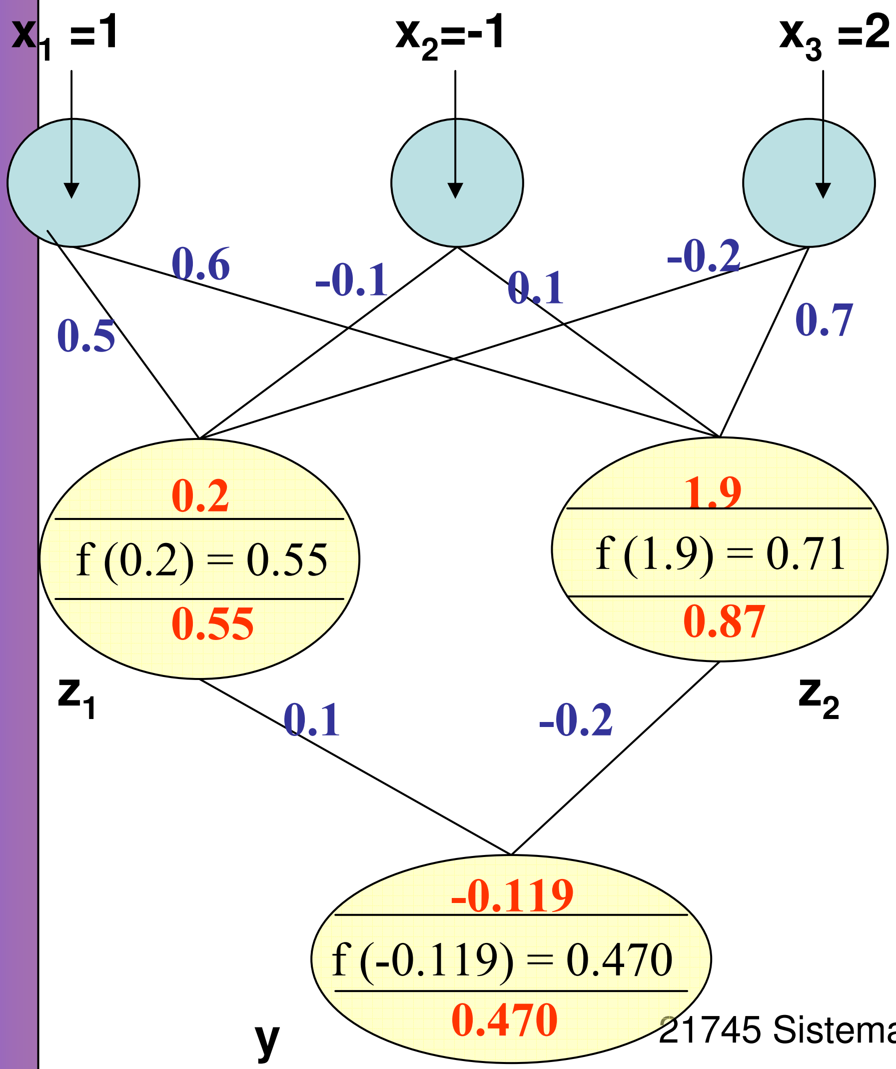
$$= x_i F'(net z_j) \sum_{k=1}^{p} (d_k - y_k) F'(net y_k) w_{jk}$$

Si sólo hay una neurona en la capa de salida

$$\Delta v_{ij} = x_i F'(net z_j)(d - y) F'(net y) w_j$$

Adaptación de los pesos usando propagación del error

Input: $x_1\ x_2\ x_3$ **Output:** y **Model:** $y = f(x_1\ x_2\ x_3)$



$$\Delta v_{ij} = x_i F'(net z_j)(d-y)F'(net y)w_j$$

Δv_{11} = 1F'(0.2) (2-0.47)F'(-0,119) 0,1

Δv_{12} = 1F'(1.9) (2-0.47)F'(-0,119)(- 0,2)

Δv_{21} =(-1)F'(0.2) (2-0.47)F'(-0,119) 0,1

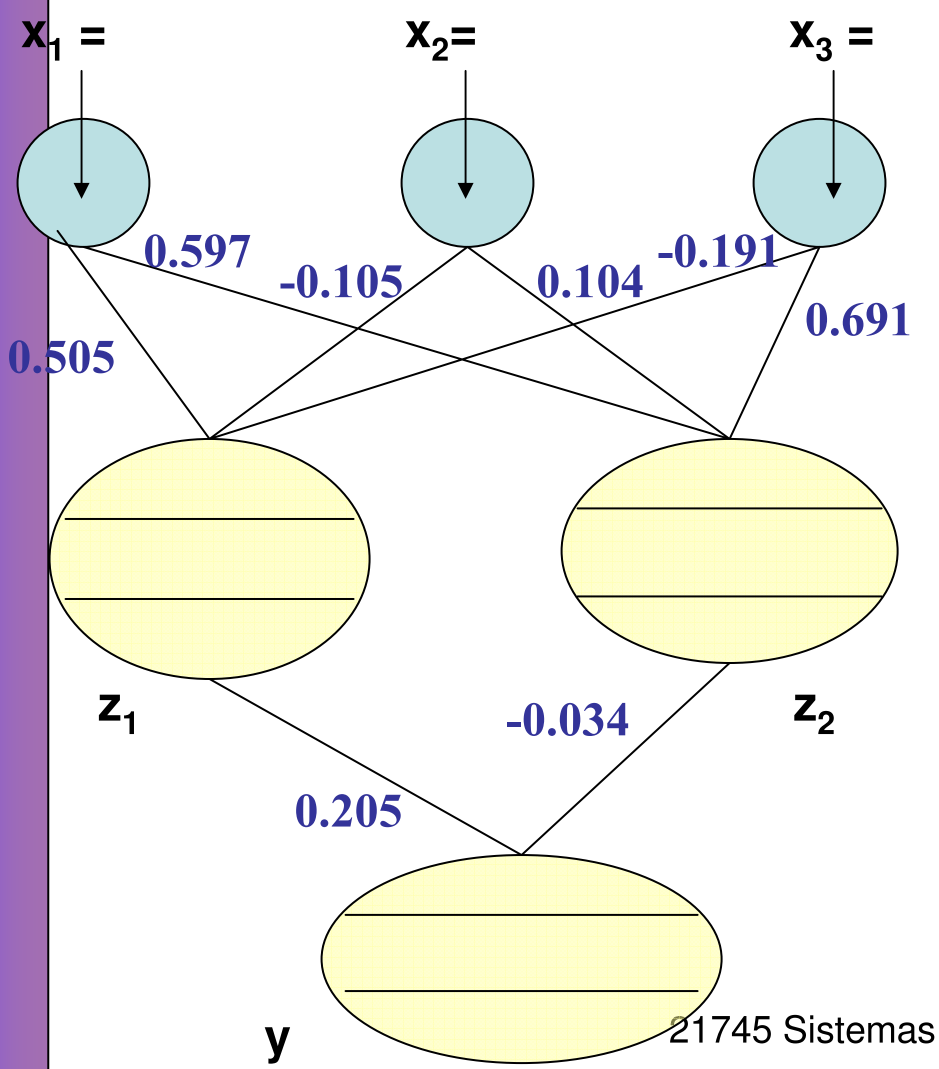
Δv_{22} = (-1)F'(1.9) (2-0.47)F'(-0,119)(- 0,2)

Δv_{31} =2F'(0.2) (2-0.47)F'(-0,119) 0,1

Δv_{32} = 2F'(1.9) (2-0.47)F'(-0,119)(- 0,2)

Una vez modificados los pesos, presentamos un nuevo ejemplo

Input: $x_1\ x_2\ x_3$ **Output:** y **Model:** $y = f(x_1\ x_2\ x_3)$

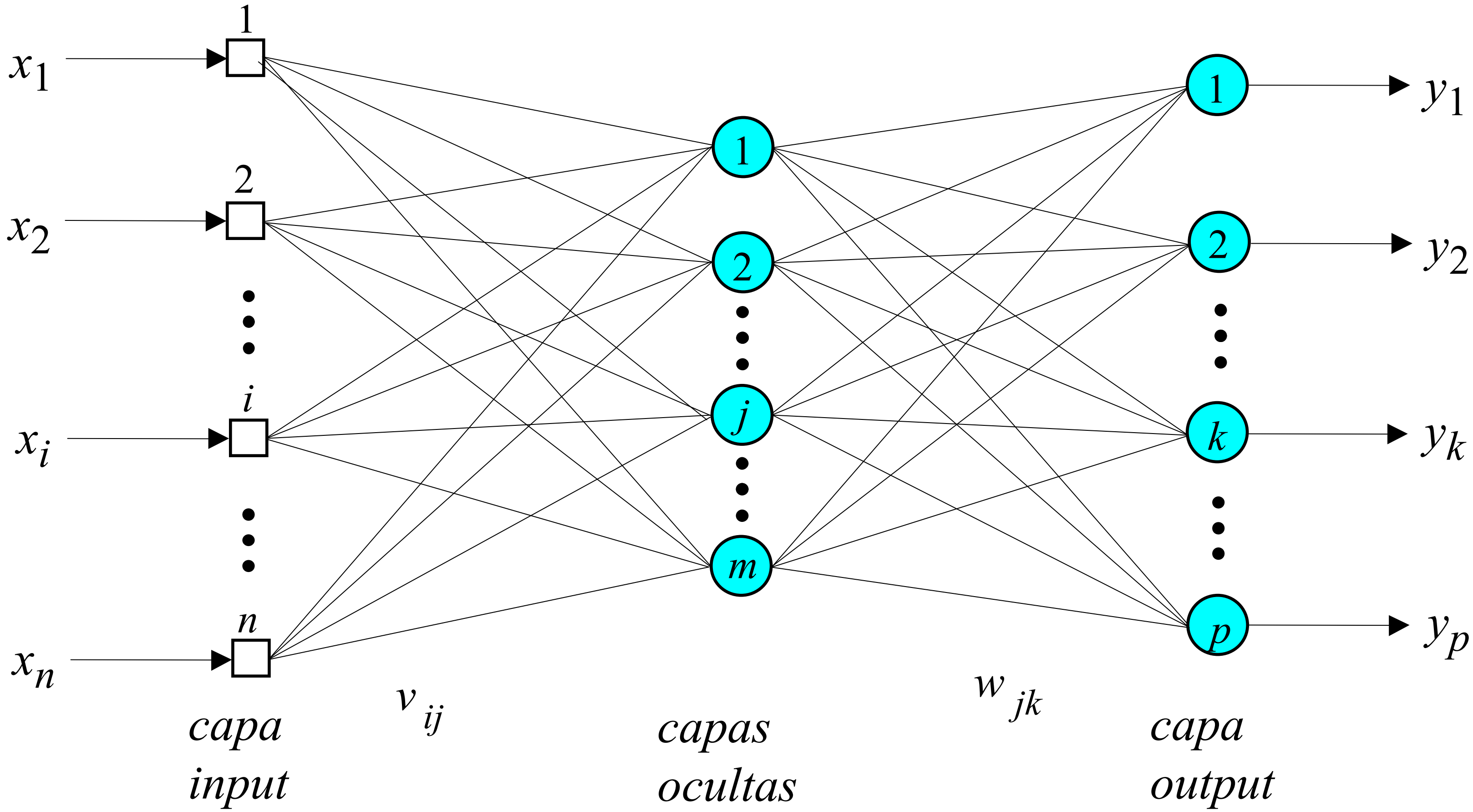


Hay que actualizar todos los pesos para cada ejemplo del conjunto de entrenamiento para cada iteración

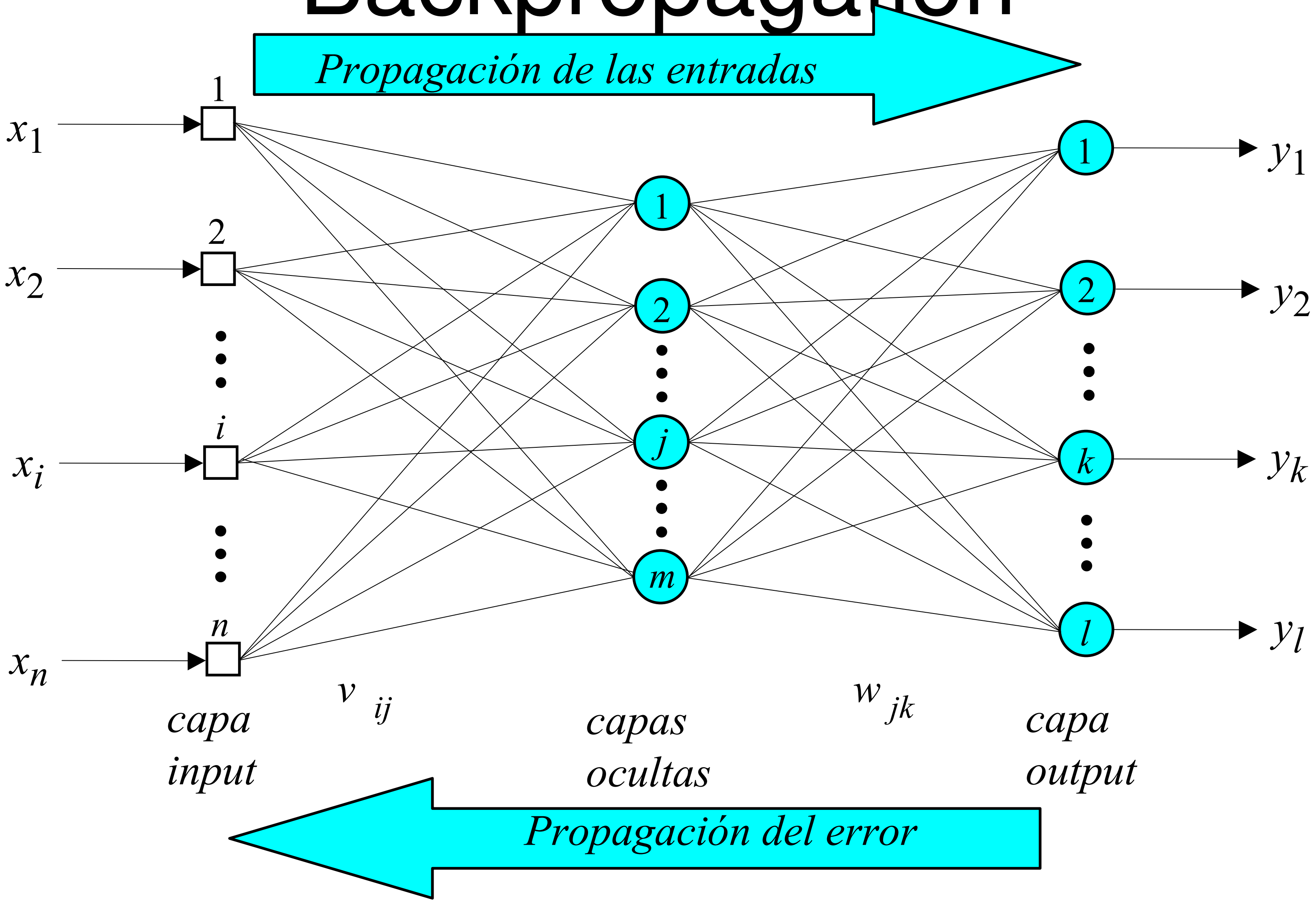
Backpropagation

- Método de aprendizaje en una red multicapa hacia adelante
- Problema para establecer el valor deseado en capas que no sean capa de salida
- Método para modificar pesos
  - Empezar en la capa de salida
  - Ir para atrás modificando pesos hasta llegar a la capa de entrada
  - Modificar pesos en cada capa

Backpropagation

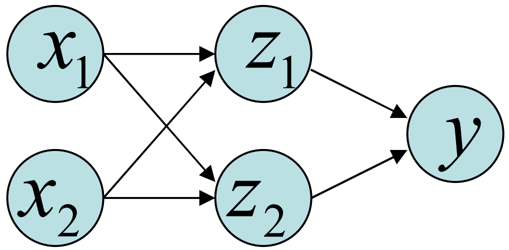


Backpropagation



Aplicación de backpropagation

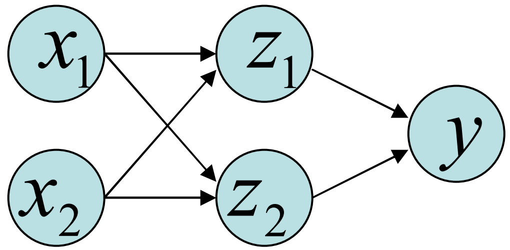
- Hemos visto las ecuaciones generales para redes con una capa oculta
- La modificación de los pesos depende de la arquitectura de la red neuronal
- Vamos a encontrar las modificaciones de pesos para la siguiente red neuronal



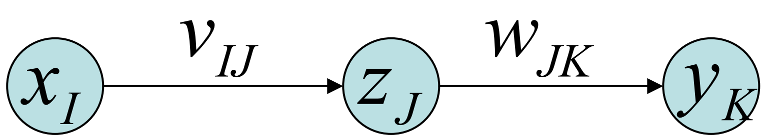
$$\Delta w_{jk} = (d_k - y_k) z_j F'(nety_k)$$

$$\Delta v_{ij} = x_i F'(netz_j) \sum_k \delta_k w_{jk}$$

$$\delta_k = (d_k - y_k) F'(nety_k)$$



Pesos z-y:

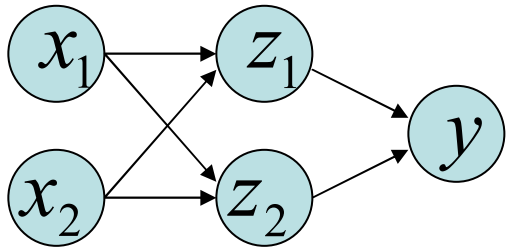


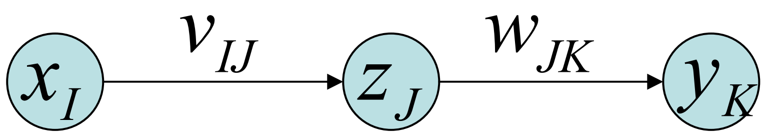
Pesos x-z:

$$\Delta w_j = (d - y) z_j F'(net\, y)$$

$$\Delta v_{ij} = x_i F'(net\, z_j)\delta\, w_j$$

$$\delta = (d - y) F'(net\, y)$$





Pesos z-y:

$$\Delta w_1 = (d - y) F'(net\, y) z_1$$

$$\Delta w_2 = (d - y) F'(net\, y) z_2$$

Pesos x-z:

$$\Delta v_{11} = F'(net\, z_1) x_1 (d - y) F'(net\, y) w_1$$

$$\Delta v_{12} = F'(net\, z_2) x_1 (d - y) F'(net\, y) w_2$$

$$\Delta v_{21} = F'(net\, z_1) x_2 (d - y) F'(net\, y) w_1$$

$$\Delta v_{22} = F'(net\, z_2) x_2 (d - y) F'(net\, y) w_2$$

Actualizar pesos por registro y cada iteración

Concepto de aprendizaje

- Aprendizaje: Capacidad para modificar el comportamiento mediante la experiencia
- Aprendizaje Automático: Disciplina encargada de estudiar y desarrollar programas informáticos que mejoran con la experiencia
- Aprendizaje de Redes Neuronales: Proceso por el que los parámetros libres de una red neuronal son adaptados de acuerdo con los estímulos de su entorno
- Algoritmo de aprendizaje: algoritmo para ajustar los pesos de una RNA

Aprendizaje del perceptrón

La corrección del error se realiza por el método incremental

- Se asignan valores arbitrarios a los pesos

  $\mathbf{W}^{t\,(1)} = [w_0 = -\theta, w_1, w_2, \ldots\, w_n]^t$

- Repetimos para cada ejemplo del conjunto de entrenamiento [$\mathbf{X}^z$, d^z], hasta que todos los registros queden clasificados correctamente por el mismo vector peso
  - Presentación del ejemplo [$\mathbf{X}^z$, d^z]
  - Calculamos la salida actual $y^z = F(\mathbf{W}^{t\,(k)} \cdot \mathbf{X}^z)$
  - Calculamos el error cometido $e = d^z - y^z$
  - Adaptamos los pesos (corrección del error)

$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + \lambda(d^z - y^z)\,\mathbf{X}^z$$

λ = factor de aprendizaje

Aprendizaje del perceptrón

TEOREMA:

Si el conjunto de datos $\mathcal{D} = \{[\mathbf{X}^z, d^z], z = 1, 2, \ldots p\}$ es finito y linealmente separable, el algoritmo anterior encuentra una solución en un tiempo finito (converge)

La regla delta

Basada en la minimización del error:

$$E = \frac{1}{2}(d-y)^2 = \frac{1}{2}(d - \sum_{i=0}^{n} w_i x_i)^2$$

- Se asignan valores arbitrarios a los pesos

  $\mathbf{W}^{\mathbf{t}\ \mathbf{(1)}} = [w_0 = -\theta, w_1, w_2, \ldots. w_n]^t$

- Repetimos para cada ejemplo del conjunto de entrenamiento [$\mathbf{X}^z$, d^z], hasta que todos los registros queden clasificados correctamente por el mismo vector peso
  - Presentación del ejemplo [$\mathbf{X}^z$, d^z]
  - Calculamos la salida actual $y^z = F(\mathbf{W}^{\mathbf{t}\ (k)} \cdot \mathbf{X}^z)$
  - Calculamos el error cometido $e = d^z - y^z$
  - Adaptamos los pesos

$$w_i^{(k+1)} = w_i^{(k)} + \alpha(d-y)F_S(net\,y)(1-F_S(net\,y))x_i$$

Backpropagation

Las salidas de una capa son las entradas de la siguiente; propagar hacia atrás el error

Esquema iterativo en dos etapas:

- Propagación hacia adelante: Evaluar el nivel de activación de las neuronas y calcular el error de la red
- Propagar el error hacia atrás, capa a capa, modificando los pesos

Backpropagation

Se asignan valores arbitrarios a los pesos

$\mathbf{W}^{t\ (1)} = [w_0 = -\theta, w_1, w_2, \ldots. w_n]^t$

- Repetimos para cada ejemplo del conjunto de entrenamiento [$\mathbf{X}^z$, d^z], completando iteraciones hasta alcanzar condición de parada
  - Presentación del ejemplo [$\mathbf{X}^z$, d^z] y propagación hacia la salida
  - Calculamos E
  - Aplicamos regla delta generalizada para adaptar los pesos

$$\Delta w_{jk} = (d_k - y_k) z_j F'(net\,y_k)$$

$$\Delta v_{ij} = x_i F'(net\,z_j) \sum_k \delta_k w_{jk}$$

$$\delta_k = (d_k - y_k) F'(net\,y_k)$$

Condiciones de parada

No se puede demostrar la convergencia del BP: Criterios heurísticos

- Gradiente cero.
  - Si w es extremo $\Rightarrow$ gradE(w) = 0
  - Parar cuando se alcance gradE(w) = 0
- Estado estacionario. Parar cuando el cambio en la función de error E sea suficientemente pequeño
- Gasto computacional fijo (nº epochs)

Evaluación

La evaluación de un algoritmo de clasificación se puede realizar atendiendo a distintos aspectos del modelo creado o del proceso utilizado para crearlo:

- **Precisión:** porcentaje de casos clasificados correctamente
- **Eficiencia:** tiempo necesario para construir/usar el clasificador
- **Robustez:** frente a ruido y valores nulos
- **Escalabilidad:** utilidad en grandes bases de datos

Evaluación

La evaluación de un algoritmo de clasificación se puede realizar atendiendo a distintos aspectos del modelo creado o del proceso utilizado para crearlo:

- **Precisión:** porcentaje de casos clasificados correctamente
- **Eficiencia:** tiempo necesario para construir/usar el clasificador
- **Robustez:** frente a ruido y valores nulos
- **Escalabilidad:** utilidad en grandes bases de datos
- **Interpretabilidad**
  (el clasificador, ¿es sólo una caja negra?)
- **Complejidad**
  (del modelo de clasificación) $\rightarrow$ Navaja de Occam

En igualdad de condiciones
la solución más sencilla es probablemente la correcta

Estimación de la precisión del modelo

- Antes de construir el modelo de clasificación, se divide el conjunto de datos disponible en un conjunto de **entrenamiento** (para construir el modelo) y un conjunto de **verificación** (para evaluar el modelo).
- Una vez construido el modelo, se usa para clasificar los datos del conjunto de verificación: comparando los casos etiquetados del conjunto de verificación con el resultado de aplicar el modelo, se obtiene un **porcentaje de clasificación**.
- Si la precisión del clasificador es aceptable, podremos utilizar el modelo para clasificar nuevos casos de los que desconocemos su clase.

- Matriz de confusión

		PREDICCIÓN	
		C_1	C_2
CLASE REAL	C_1	**a**	**b**
	C_2	**c**	**d**

- Matriz de confusión

		PREDICCIÓN	
		C_1	C_2
CLASE REAL	C_1	**a**	**b**
	C_2	**c**	**d**

Etiquetas correctas

Error tipo I

Error tipo II

- Matriz de confusión

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="2">PREDICCIÓN</td></tr>
<tr><td>C_1</td><td>C_2</td></tr>
<tr><td rowspan="2">CLASE REAL</td><td>C_1</td><td>a</td><td>b</td></tr>
<tr><td>C_2</td><td>c</td><td>d</td></tr>
</table>

Etiquetas correctas

$$precisión = \frac{a+d}{a+b+c+d}$$

$$porcentaje \quad de \quad error = \frac{b+c}{a+b+c+d}$$

Evaluación

	C_1	C_2
C_1	a	b
C_2	c	d

clases	buy_computer = yes	buy_computer = no	total	recognition(%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
total	7366	2634	10000	95.52

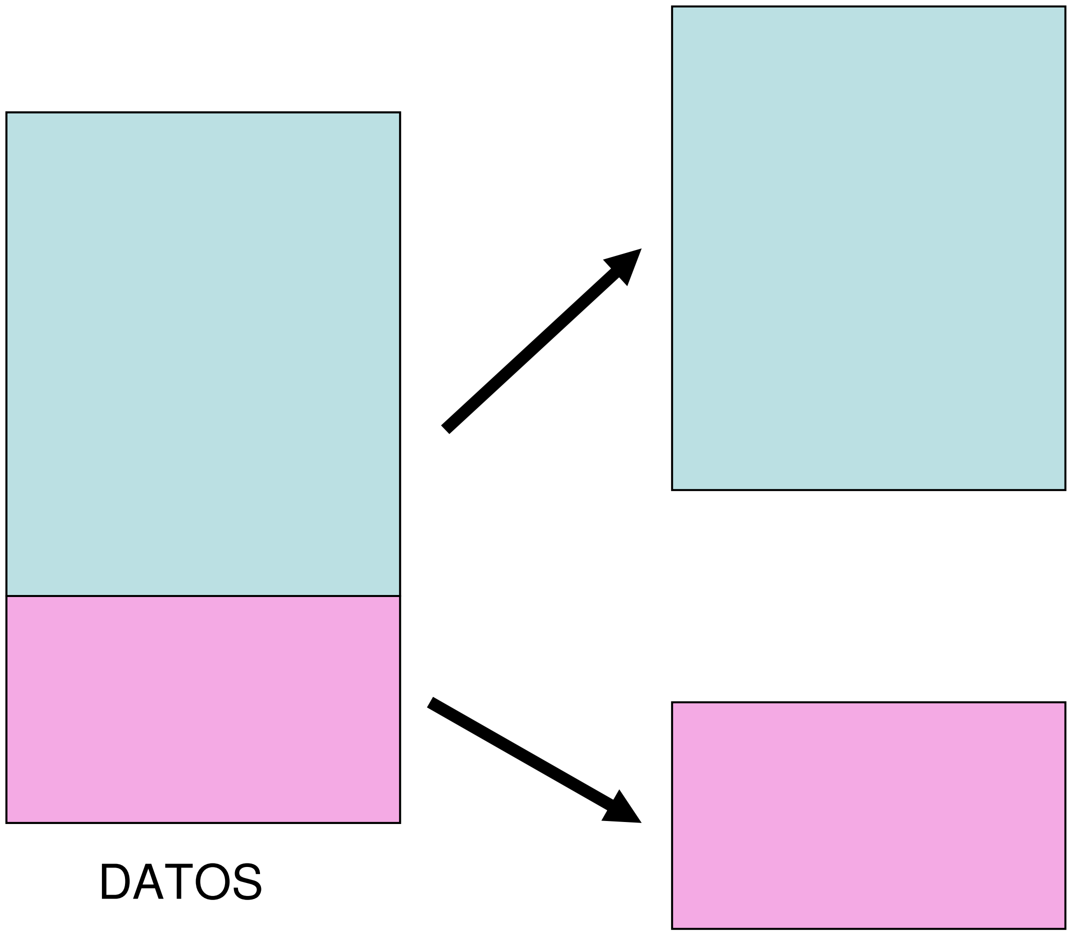
- Precisión = (6954+2588)/10000 = 0.9542
- Porcentaje de error = 1 – 0.9542 = 0.0458
- Otras medidas (diagnóstico del cáncer))

  sensitividad = a/(a+c)

  especificidad = d/(b+d)

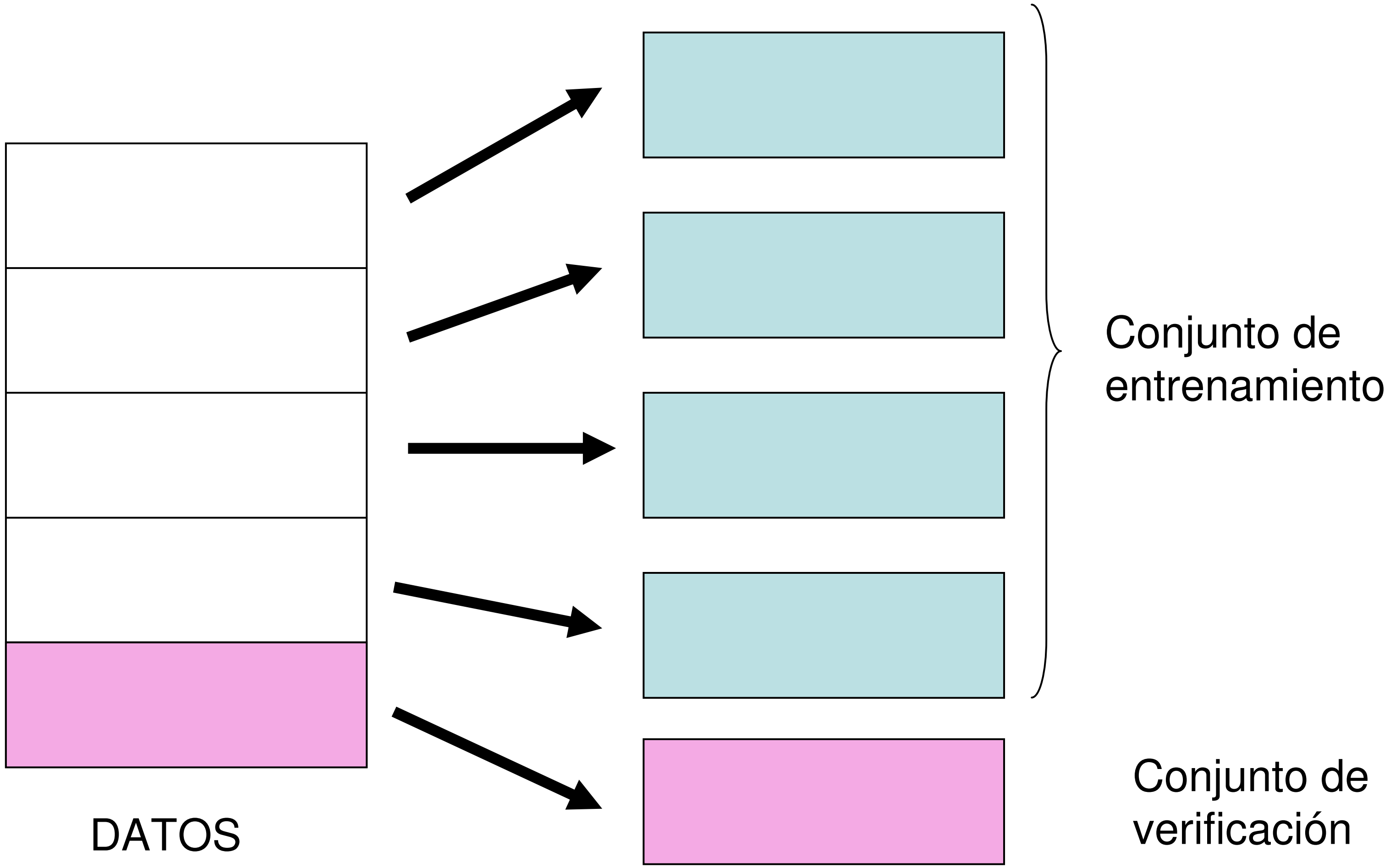
- El conjunto de verificación debe ser independiente del conjunto de entrenamiento.
- El error de clasificación en el conjunto de entrenamiento **NO** es un buen estimador de la precisión del clasificador.

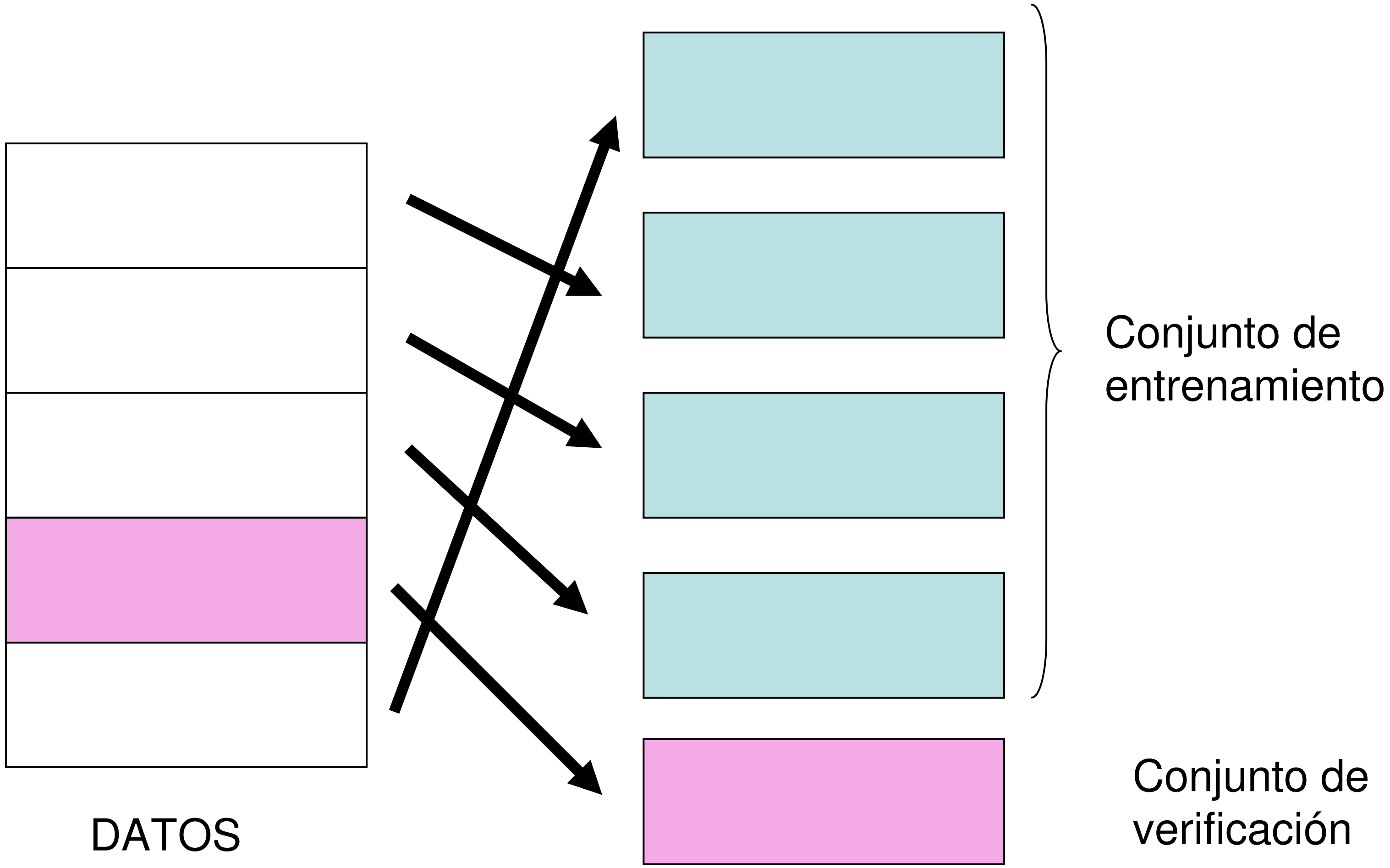
- **Partición:** se utilizan 2/3 de los registros como conjunto de entrenamiento y el 1/3 restante como conjunto de verificación para estimar la precisión del clasificador

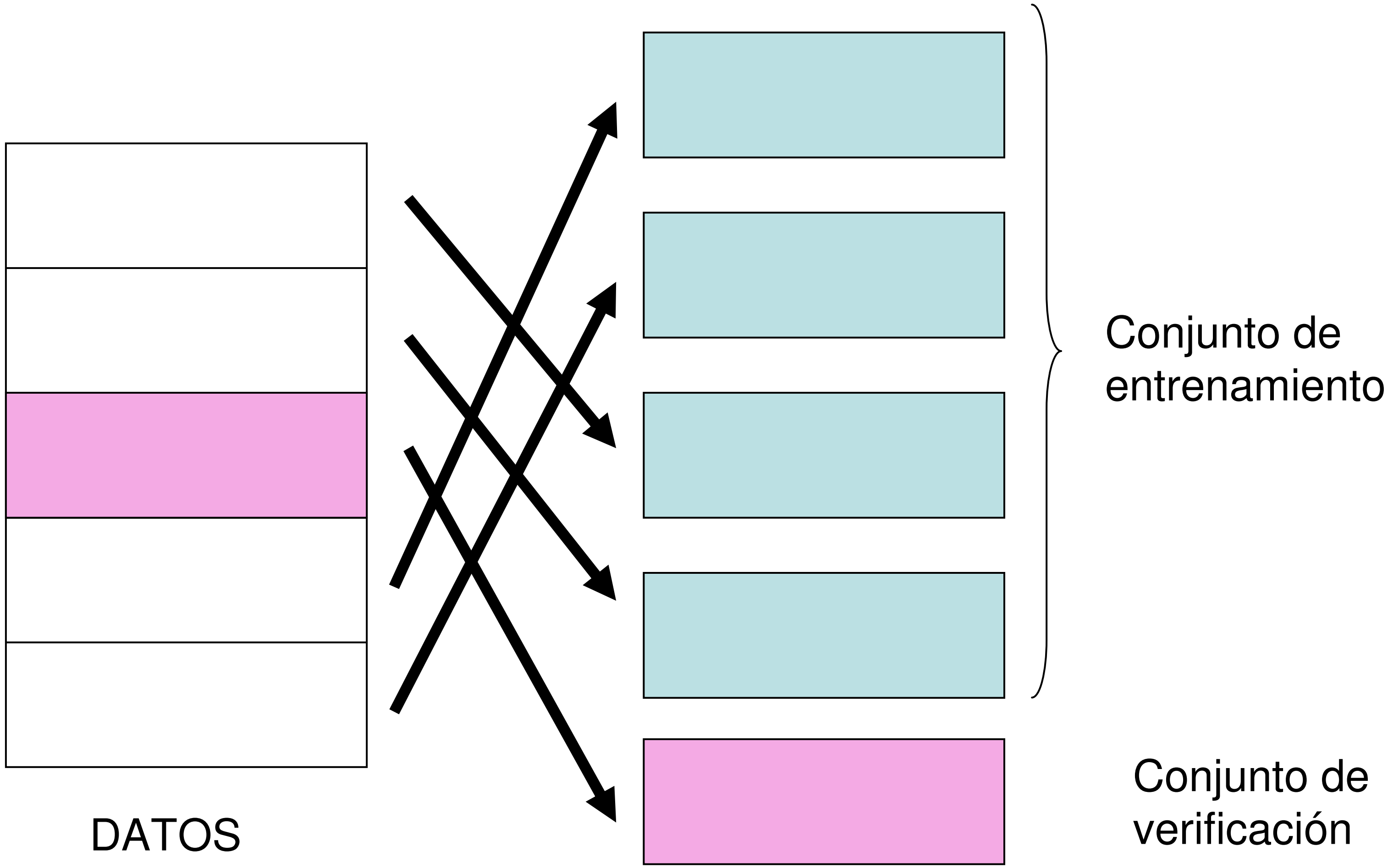

DATOS
Conjunto de entrenamiento
Conjunto de verificación

- **Partición:** se utilizan 2/3 de los registros como conjunto de entrenamiento y el 1/3 restante como conjunto de verificación para estimar la precisión del clasificador

- **Validación cruzada (**k-fold Cross-Validation)
  - Se divide aleatoriamente el conjunto de datos en k subconjuntos de intersección vacía (más o menos del mismo tamaño). Típicamente, k=10.
  - En la iteración i, se usa el subconjunto i como conjunto de verificación y los k-1 restantes como conjunto de entrenamiento.
  - Como medida de evaluación del método de clasificación se toma la media aritmética de las k iteraciones realizadas.


Conjunto de entrenamiento
Conjunto de verificación
DATOS


Conjunto de
entrenamiento
Conjunto de
verificación
DATOS


DATOS
Conjunto de entrenamiento
Conjunto de verificación

Condiciones de parada

- Parada temprana. Dividir el conjunto de datos en
  - entrenamiento: usado para ajustar los pesos
  - validacion: usado para valorar la capacidad de generalizacion
- Se mide el nivel de error en entrenamiento y en validación
- Parar cuando empiece a crecer el error en validación

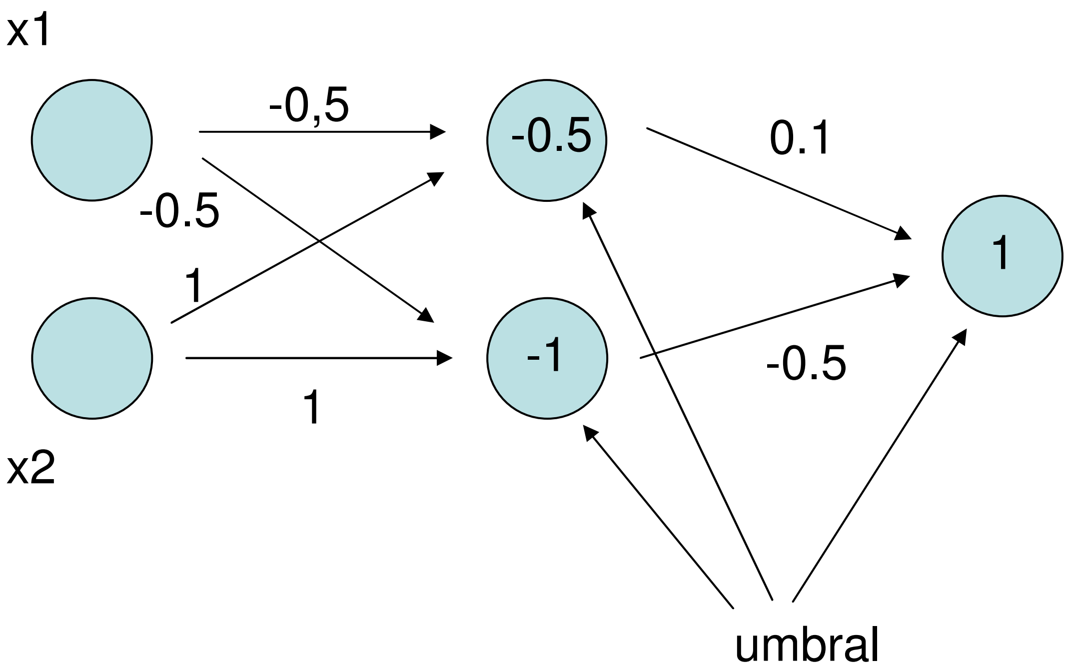
Limitaciones del BP

- Presencia de mínimos locales
- Elección de la función de error
- Sobreajuste
- Lentitud
- Sin fundamento biológico

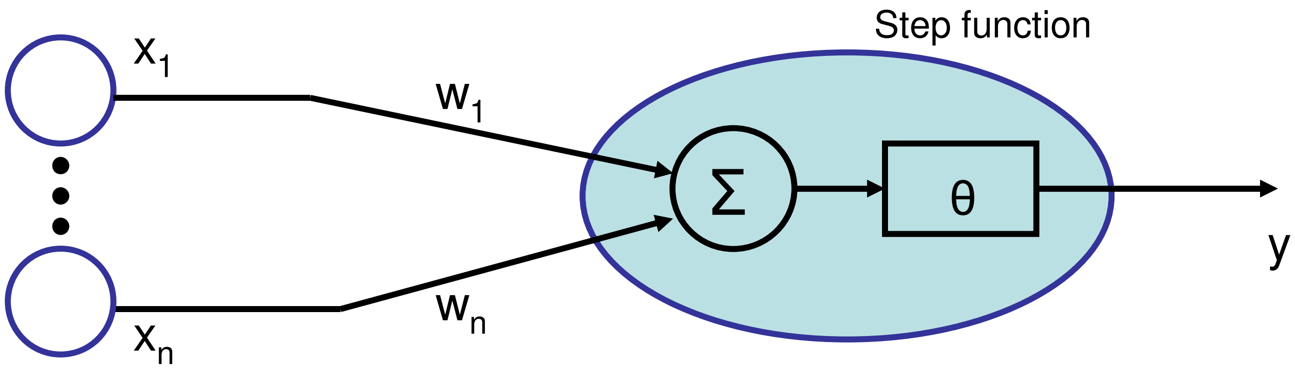
Ingeniera de RNA

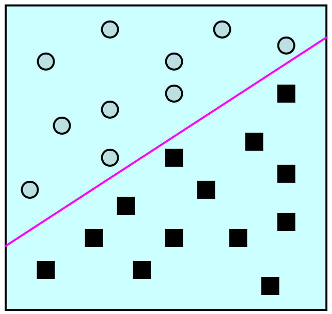
- Seleccionar el conjunto de datos. Entradas, salidas, tipo
- Establecer el modelo. Arquitectura, parámetros de aprendizaje
- Entrenar la red con el conjunto de datos
- Validar la red
- Aplicarla

x1	x2	d
0	0	0.687349
0	1	0.667459
1	0	0.698070
1	1	0.676727

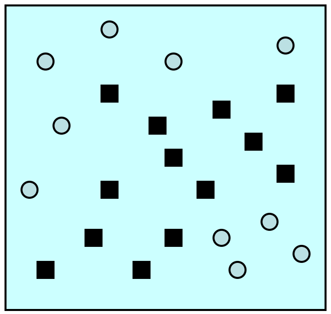


Perceptron





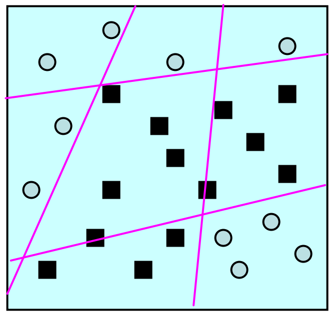
Linear partition
is possible



Linear partition
is NOT possible

Multilayered net

- Simplest case, step activation function, the number of internal units *k* defines the number of *borders*

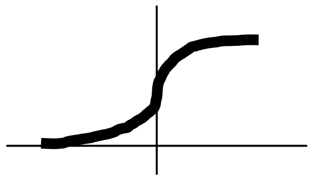


Poligonal partition
(4 internal units)

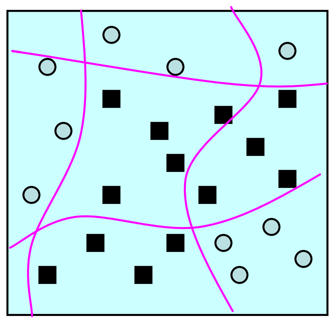
Artificial Neural Net

- Sigmoidal transfer function:

$$F(x) = \frac{1}{1+e^{-x}}$$

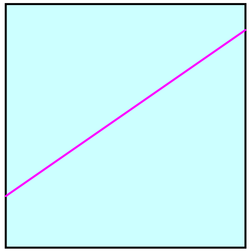


- Allows non-linear partitions:

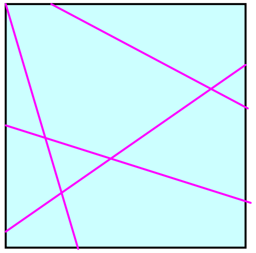


Non-linear multiple partition
(4 internal units)

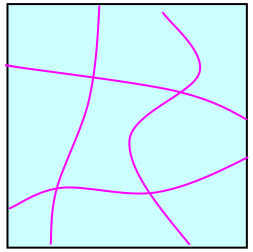
Comparison



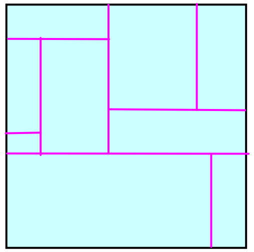
Perceptron



Multilayered perceptron
(F step function)

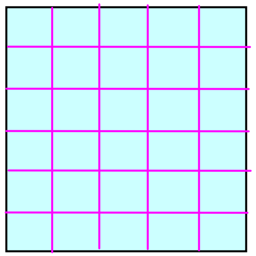


Multilayered neural net
(F sigmoidal)

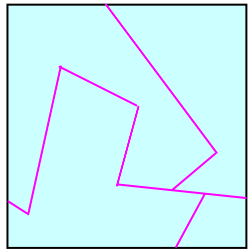


Decision tree

ID3



Naïve Bayesian



k-NN