



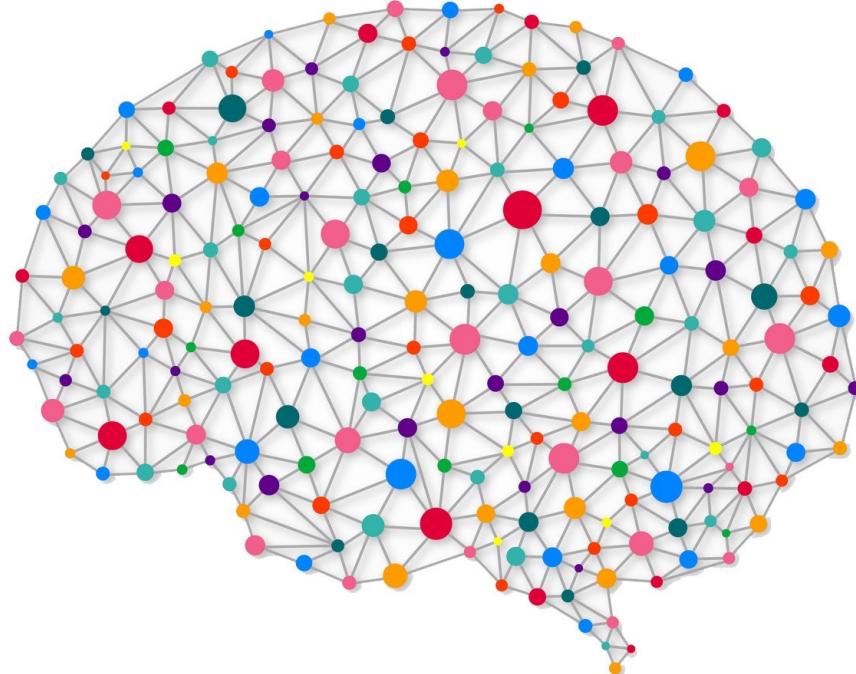
# *Aprendizaje Supervisado*

## Redes Neuronales

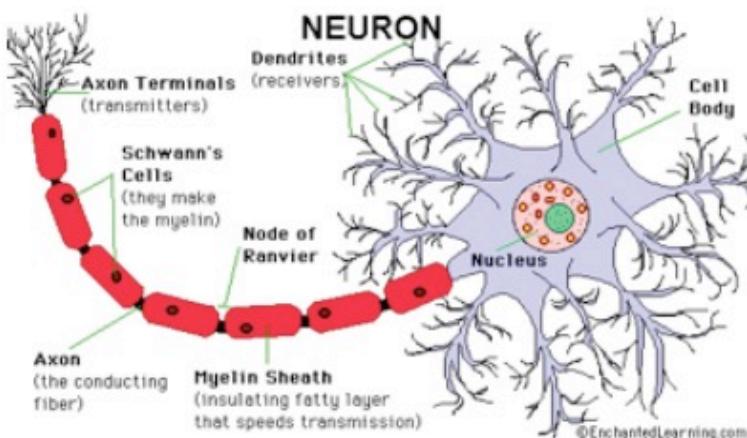
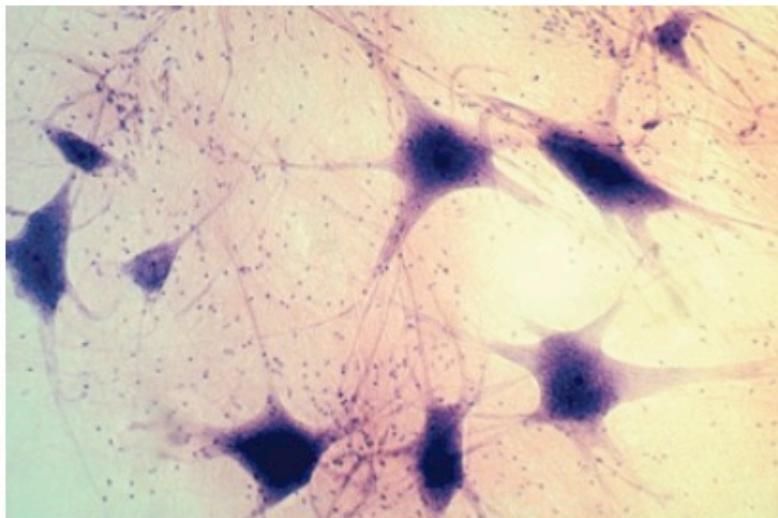
## Método del Perceptrón



# Analogía con el cerebro humano

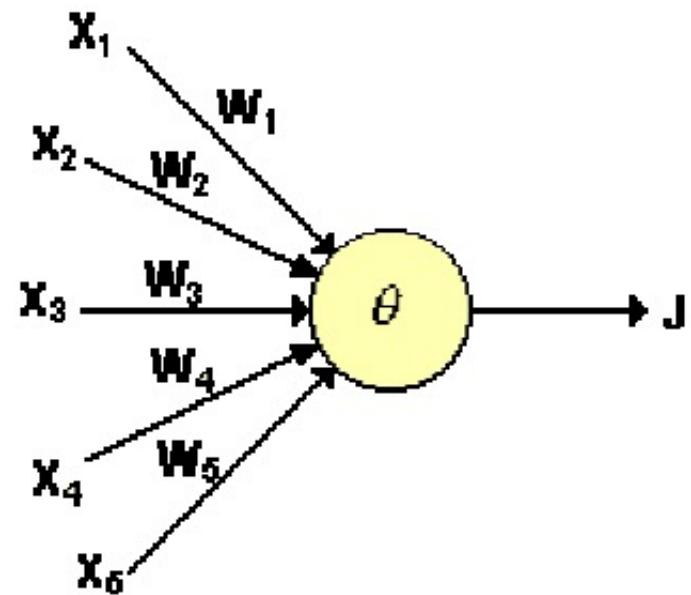
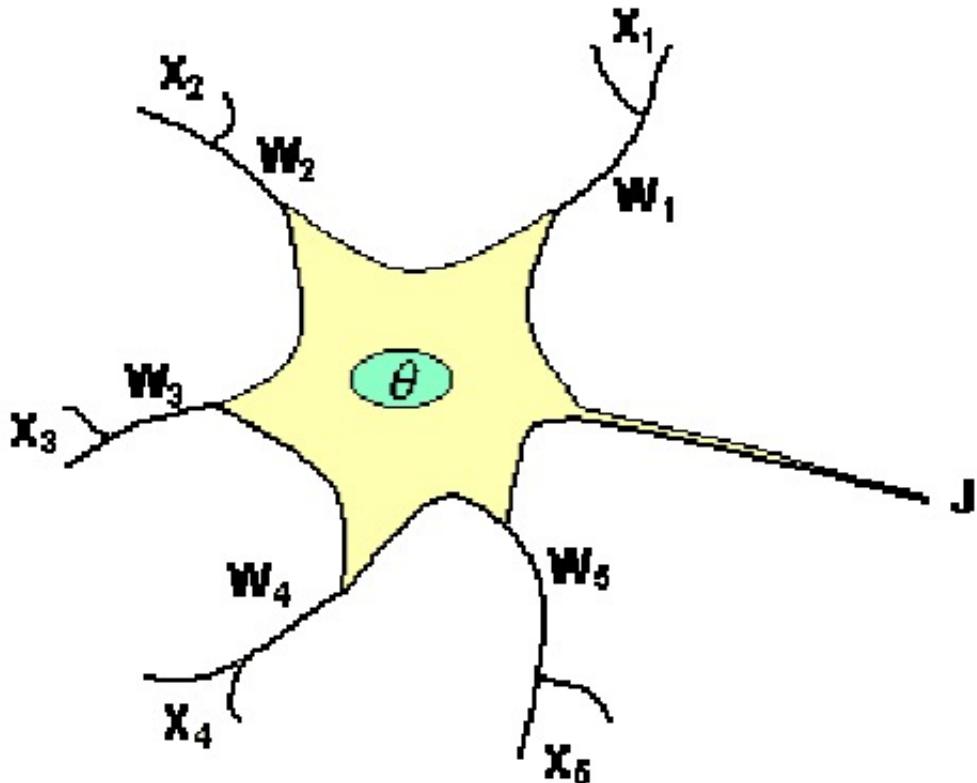


# Redes Neuronales - Perceptrón



- El cerebro humano está compuesto principalmente de células nerviosas llamada **Neuronas**.
- Estas neuronas están ligadas mediante unas fibras llamadas “**Axons**”.
- Una Neurona está conectada al Axón de otra Neorona mediante las **Dentritas**.
- En punto de contacto entre una Dentrita y el Axón se llama **Synapse**.
- Las **Redes Neuronales Artificiales** tratan de emular este esquema mediante **Nodos y Links**.

# Redes Neuronales - Perceptrón

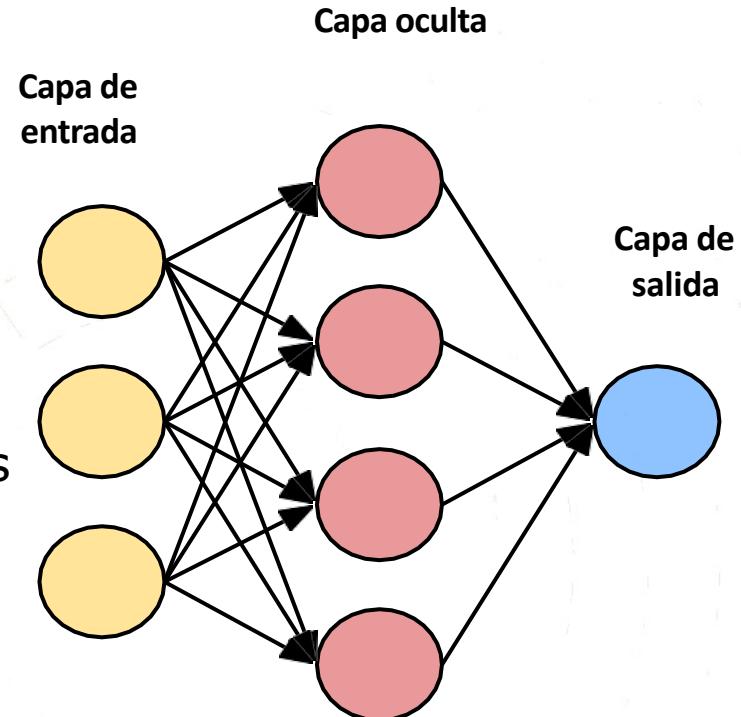


# Redes Neuronales

Comenzaremos con modelo de Redes totalmente conectadas:

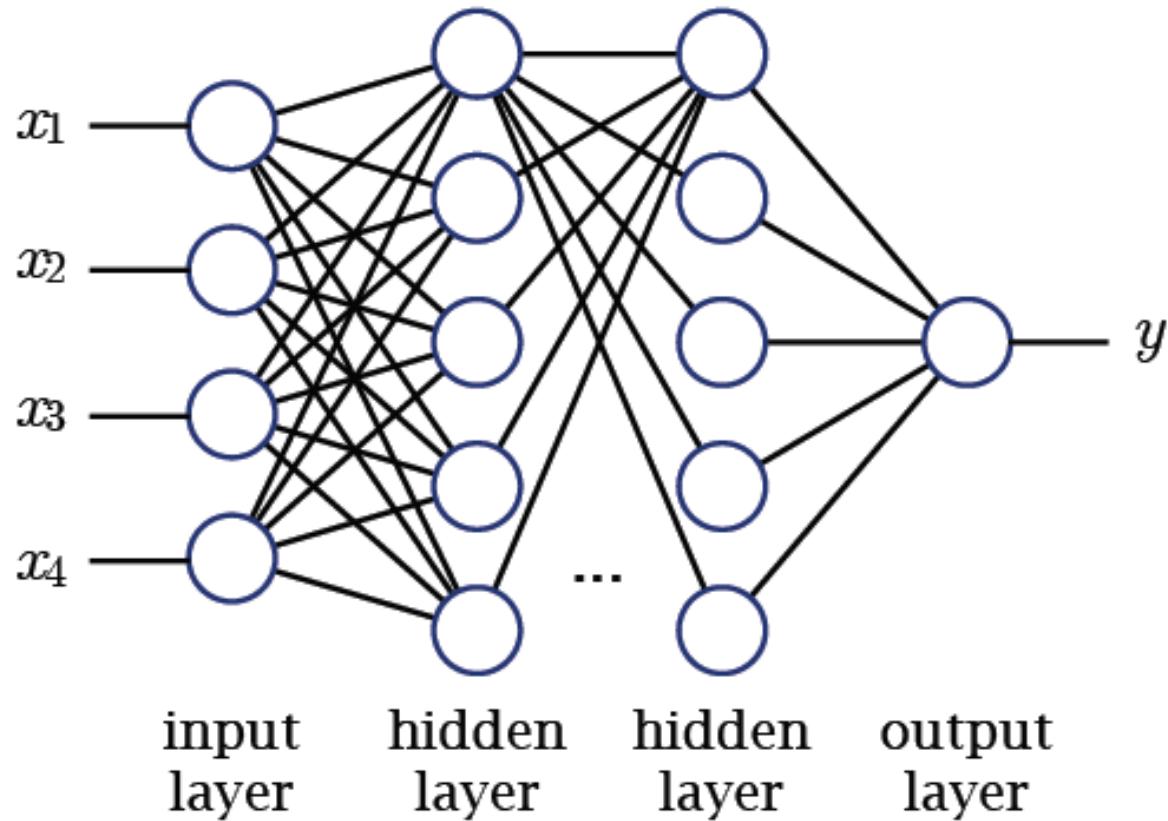
1. Una sola capa
2. Neurona única
3. Varias capas
4. Entrada y salida

Más adelante, cubriremos otros tipos de capas (convolución, max-pool, ...)



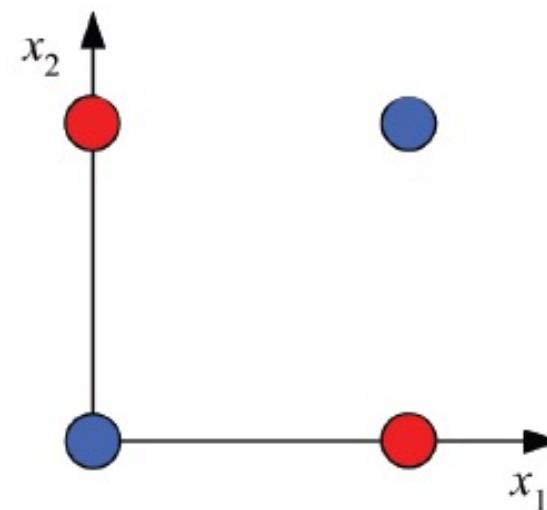
# Multi-layer perceptrons

feed-forward network

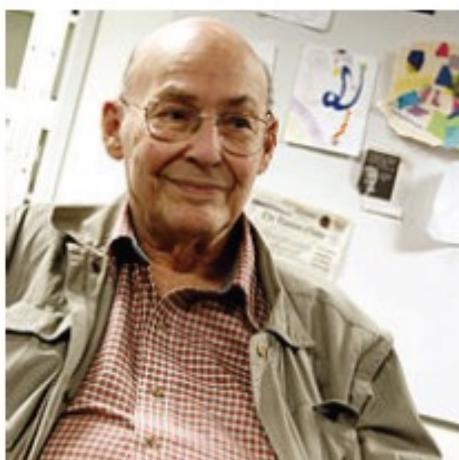


# Perceptrón - xor

$x_1$	$x_2$	$r$
0	0	0
0	1	1
1	0	1
1	1	0

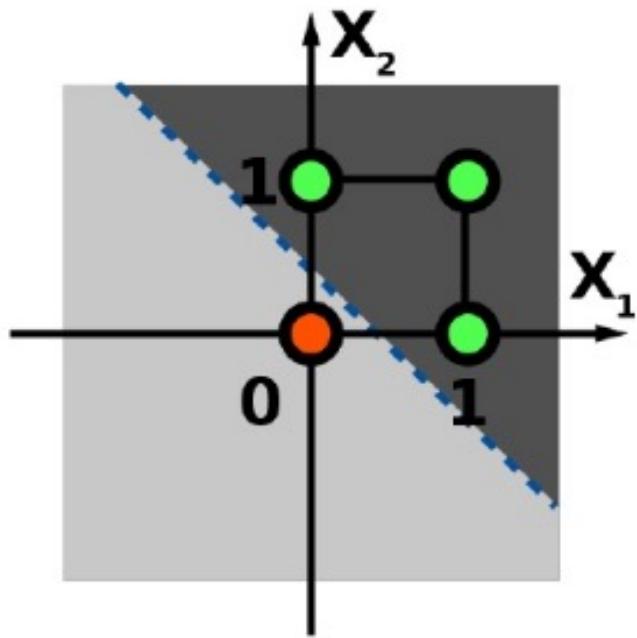


[Minsky and Papert, *Perceptrons*, 1969]



Marvin Minsky  
Turing Award 1969

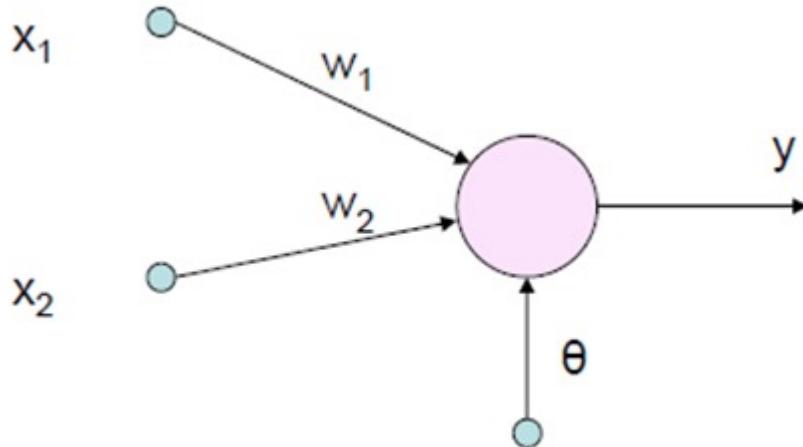
# Perceptrón - or



**OR ( $x_1 \cup x_2$ )**

X1	X2	OR
0	0	0
0	1	1
1	0	1
1	1	1

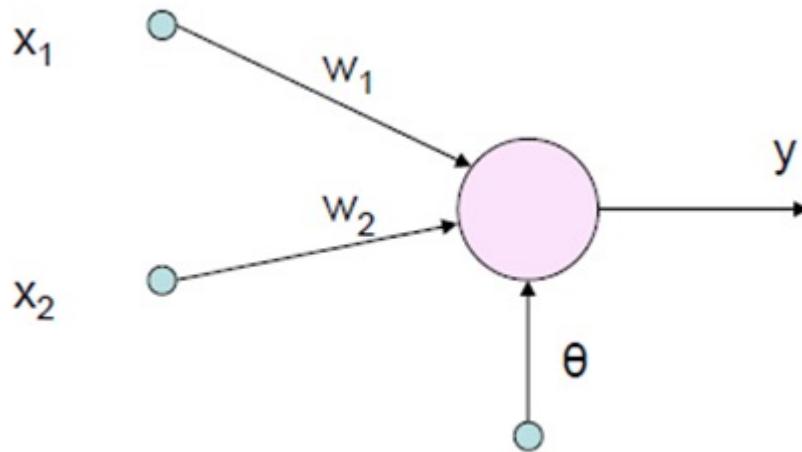
# Perceptrón - or



$$Y = I(W_1 X_1 + W_2 X_2 > \theta)$$

donde  $I(z) = \begin{cases} 1 & \text{Si } z \text{ es verdadero} \\ 0 & \text{En otro caso} \end{cases}$

# Perceptrón - or



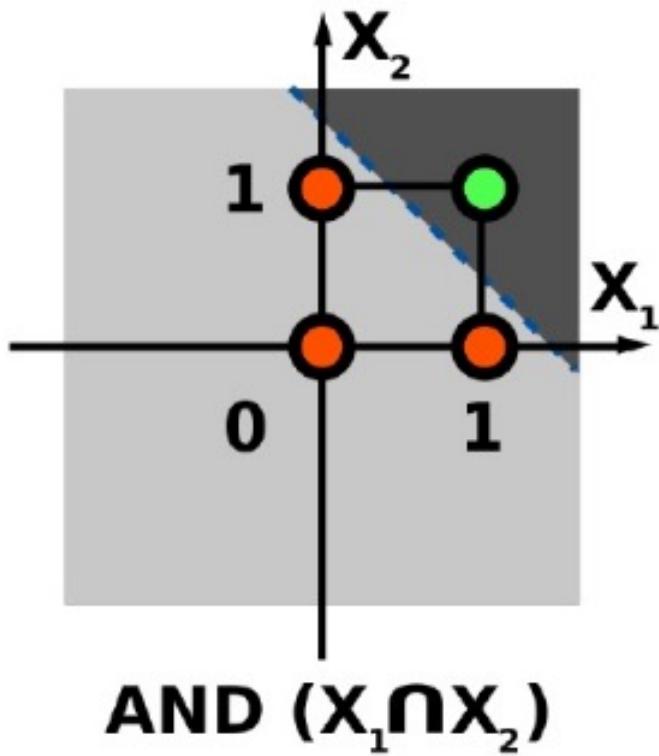
Si  $\theta = 0.5$ ,  $W_1 = 1$  y  $W_2 = 1$  entonces

$$Y = I(X_1 + X_2 > 0.5)$$

donde  $I(z) = \begin{cases} 1 & \text{Si } z \text{ es verdadero} \\ 0 & \text{En otro caso} \end{cases}$

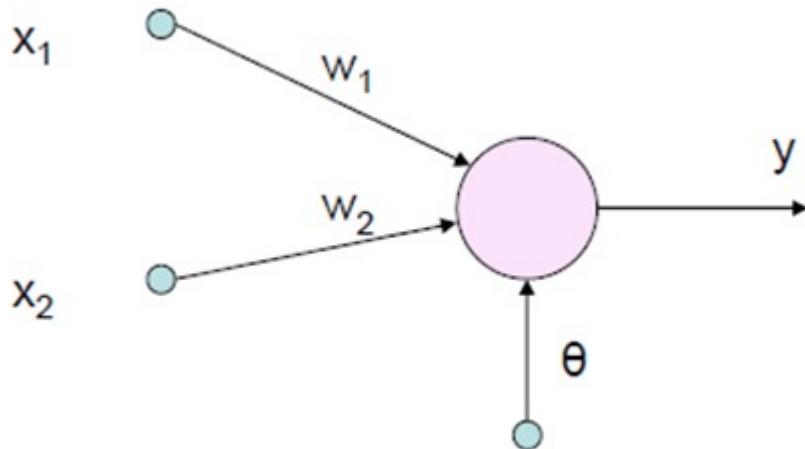
$x_1$	$x_2$	OR
0	0	0
0	1	1
1	0	1
1	1	1

# Perceptrón - and



X1	X2	AND
0	0	0
0	1	0
1	0	0
1	1	1

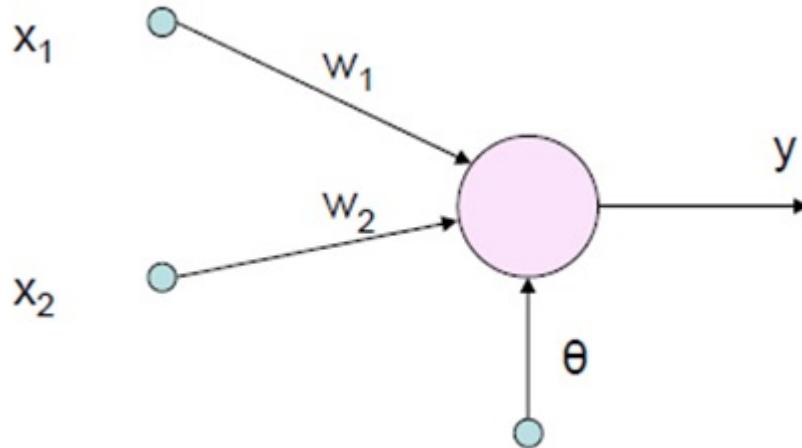
# Perceptrón - and



$$Y = I(W_1X_1 + W_2X_2 > \theta)$$

donde  $I(z) = \begin{cases} 1 & \text{Si } z \text{ es verdadero} \\ 0 & \text{En otro caso} \end{cases}$

# Perceptrón - and



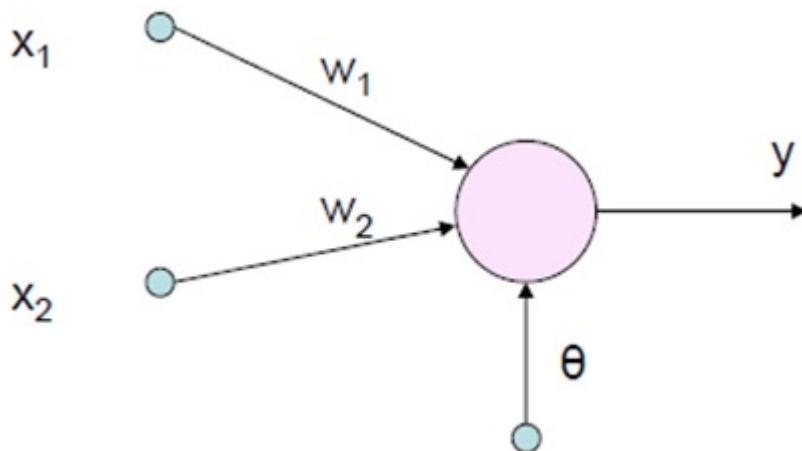
Si  $\theta = 1.5$ ,  $W_1 = 1$  y  $W_2 = 1$  entonces

$$Y = I(X_1 + X_2 > 1.5)$$

donde  $I(z) = \begin{cases} 1 & \text{Si } z \text{ es verdadero} \\ 0 & \text{En otro caso} \end{cases}$

$x_1$	$x_2$	AND
0	0	0
0	1	0
1	0	0
1	1	1

# Perceptrón - and



$\theta$  se llama Umbral

$W_1$  y  $W_2$  se llaman los Pesos

$I(Z)$  es la regla de decisión

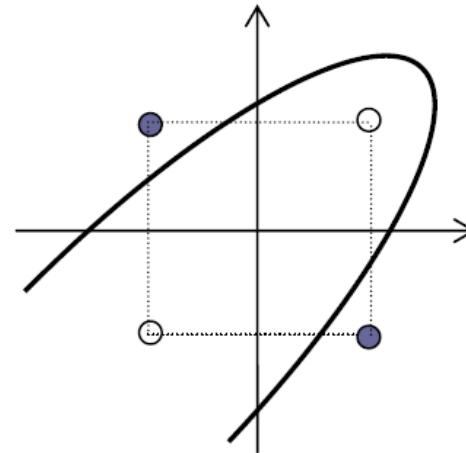
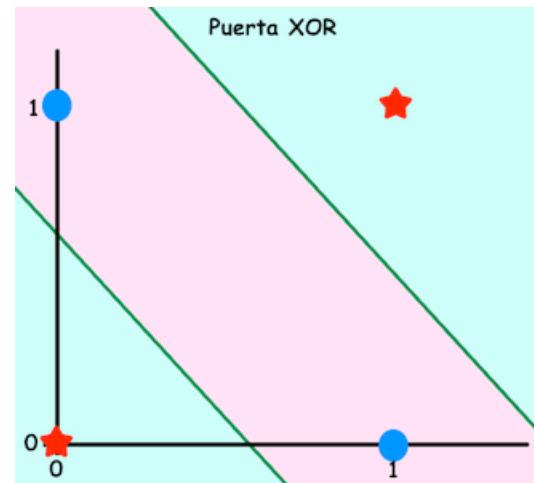
La Función de Activación usada

en todos los casos anteriores fue la identidad

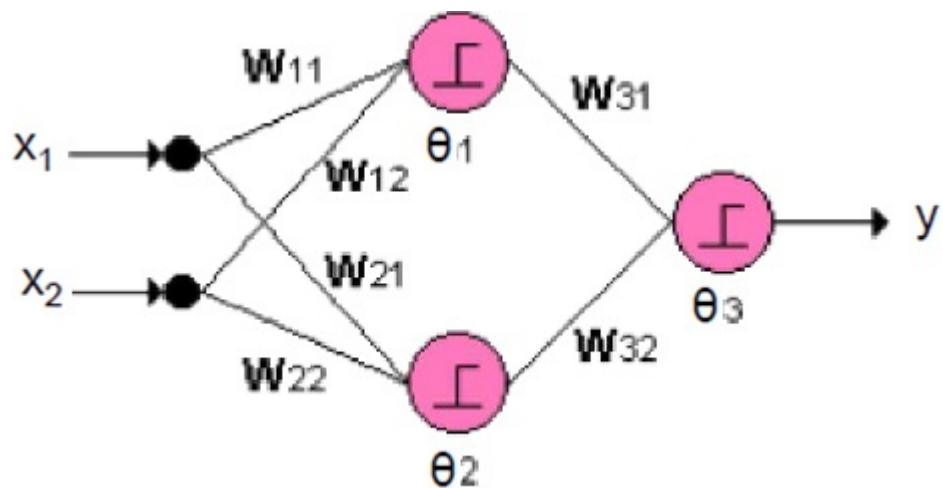
# Perceptrón - xor

XOR

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0



# Perceptrón multicapa - xor



XOR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

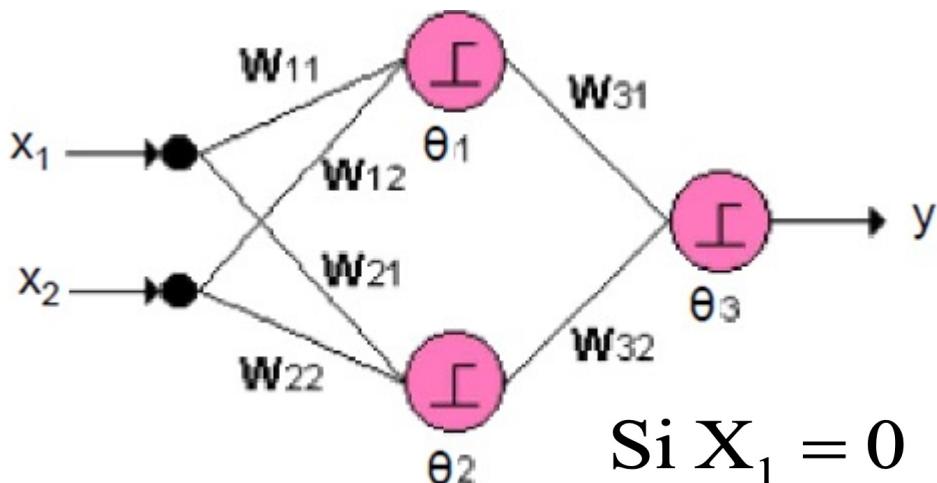
$$w_{11}=1 \quad w_{12}=1$$

$$w_{21}=1 \quad w_{22}=1$$

$$w_{31}=1 \quad w_{32}=-1.5$$

$$\theta_1=0.5 \quad \theta_2=1.5 \quad \theta_3=0.5$$

# Perceptrón multicapa - xor



$$\begin{aligned} w_{11} &= 1 & w_{12} &= 1 \\ w_{21} &= 1 & w_{22} &= 1 \\ w_{31} &= 1 & w_{32} &= -1.5 \\ \theta_1 &= 0.5 & \theta_2 &= 1.5 & \theta_3 &= 0.5 \end{aligned}$$

Si  $X_1 = 0$  y  $X_2 = 0$  entonces

$$I(W_{11}X_1 + W_{12}X_2 = 0 > 0.5) = 0$$

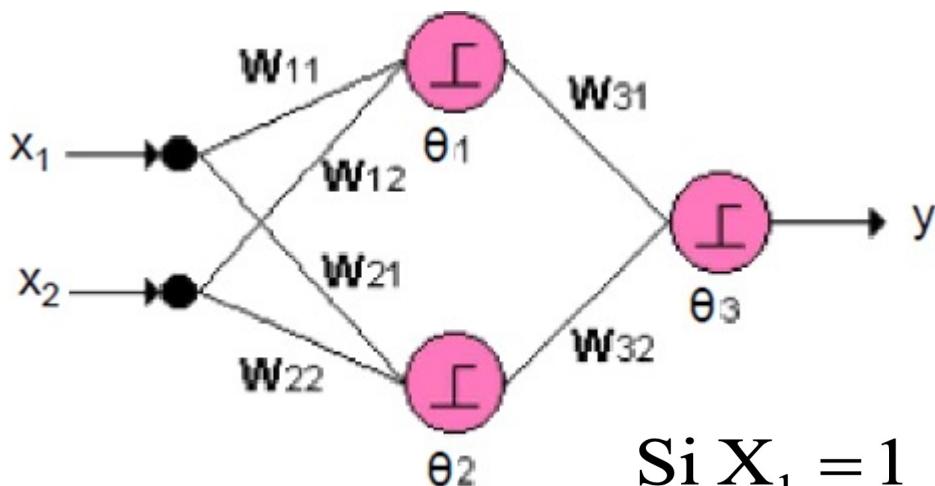
$$I(W_{21}X_1 + W_{22}X_2 = 0 > 1.5) = 0$$

$$Y = I(W_{31} * 0 + W_{32} * 0 > 0.5) = 0$$

donde  $I(z) = \begin{cases} 1 & \text{Si } z \text{ es verdadero} \\ 0 & \text{En otro caso} \end{cases}$

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

# Perceptrón multicapa - xor



$$\begin{aligned} w_{11} &= 1 & w_{12} &= 1 \\ w_{21} &= 1 & w_{22} &= 1 \\ w_{31} &= 1 & w_{32} &= -1.5 \\ \theta_1 &= 0.5 & \theta_2 &= 1.5 & \theta_3 &= 0.5 \end{aligned}$$

**XOR**

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Si  $X_1 = 1$  y  $X_2 = 0$  entonces

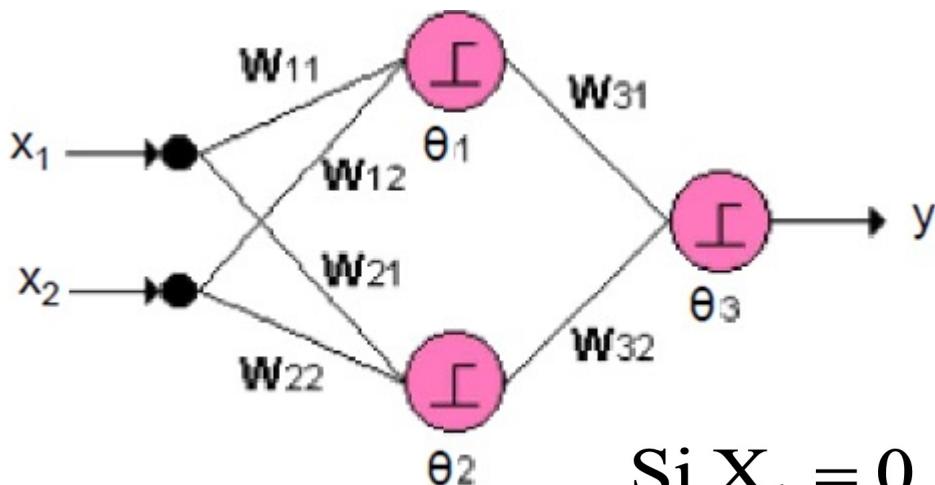
$$I(W_{11}X_1 + 0 = 1 > 0.5) = 1$$

$$I(W_{21}X_1 + 0 = 1 > 1.5) = 0$$

$$Y = I(W_{31} * 1 + 0 = 1 > 0.5) = 1$$

donde  $I(z) = \begin{cases} 1 & \text{Si } z \text{ es verdadero} \\ 0 & \text{En otro caso} \end{cases}$

# Perceptrón multicapa - xor



$$\begin{aligned} w_{11} &= 1 & w_{12} &= 1 \\ w_{21} &= 1 & w_{22} &= 1 \\ w_{31} &= 1 & w_{32} &= -1.5 \\ \theta_1 &= 0.5 & \theta_2 &= 1.5 & \theta_3 &= 0.5 \end{aligned}$$

Si  $X_1 = 0$  y  $X_2 = 1$  entonces

$$I(0 + W_{12}X_2 = 1 > 0.5) = 1$$

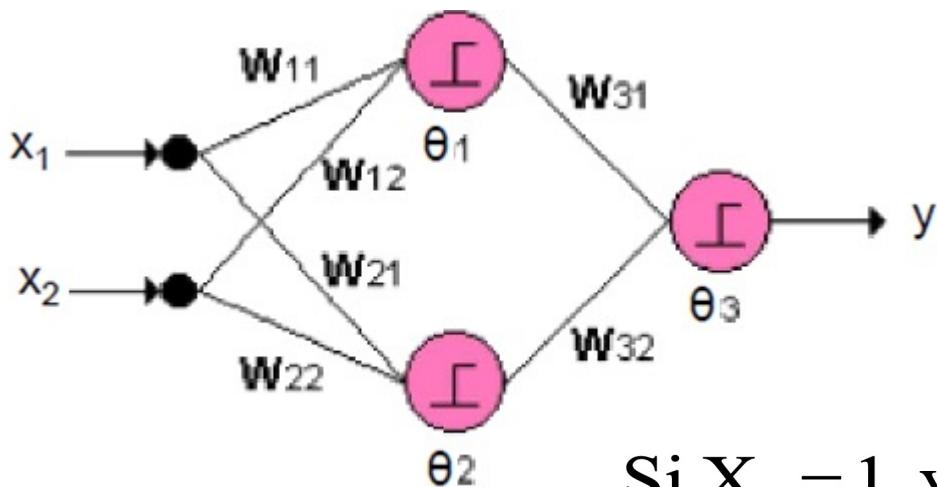
$$I(W_{22}X_2 + 0 = 1 > 1.5) = 0$$

$$Y = I(W_{31}X_1 + W_{32} * 0 = 1 > 0.5) = 1$$

donde  $I(z) = \begin{cases} 1 & \text{Si } z \text{ es verdadero} \\ 0 & \text{En otro caso} \end{cases}$

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

# Perceptrón multicapa - xor



$$\begin{aligned} w_{11} &= 1 & w_{12} &= 1 \\ w_{21} &= 1 & w_{22} &= 1 \\ w_{31} &= 1 & w_{32} &= -1.5 \\ \theta_1 &= 0.5 & \theta_2 &= 1.5 & \theta_3 &= 0.5 \end{aligned}$$

**XOR**

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Si  $X_1 = 1$  y  $X_2 = 1$  entonces

$$I(W_{11}X_1 + W_{12}X_2 = 2 > 0.5) = 1$$

$$I(W_{21}X_1 + W_{22}X_2 = 2 > 1.5) = 1$$

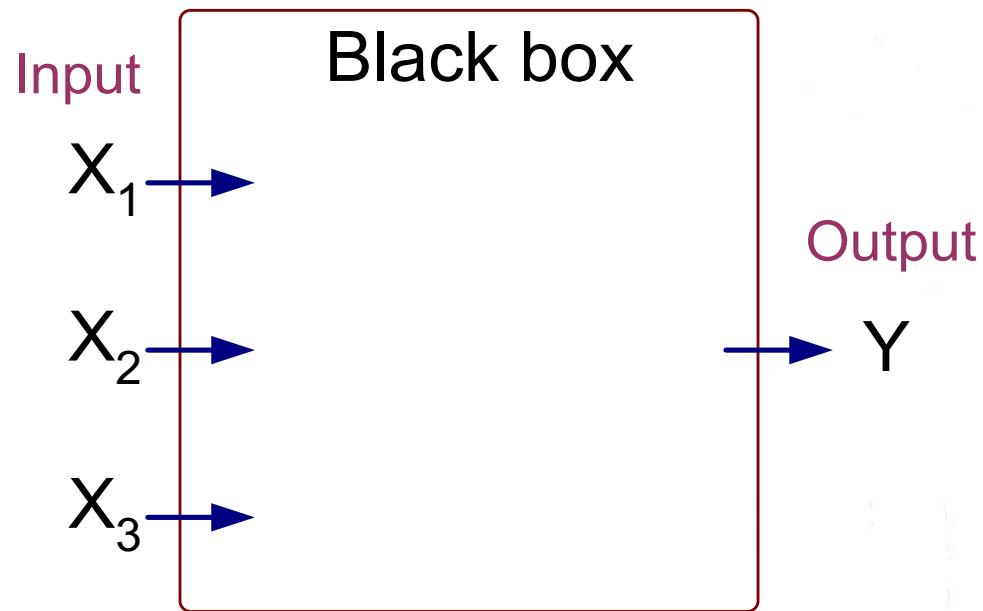
$$Y = I(W_{31} * 1 + W_{32} * 1 = -0.5 > 0.5) = 0$$

donde  $I(z) = \begin{cases} 1 & \text{Si } z \text{ es verdadero} \\ 0 & \text{En otro caso} \end{cases}$

# Redes Neuronales

## Perceptrón

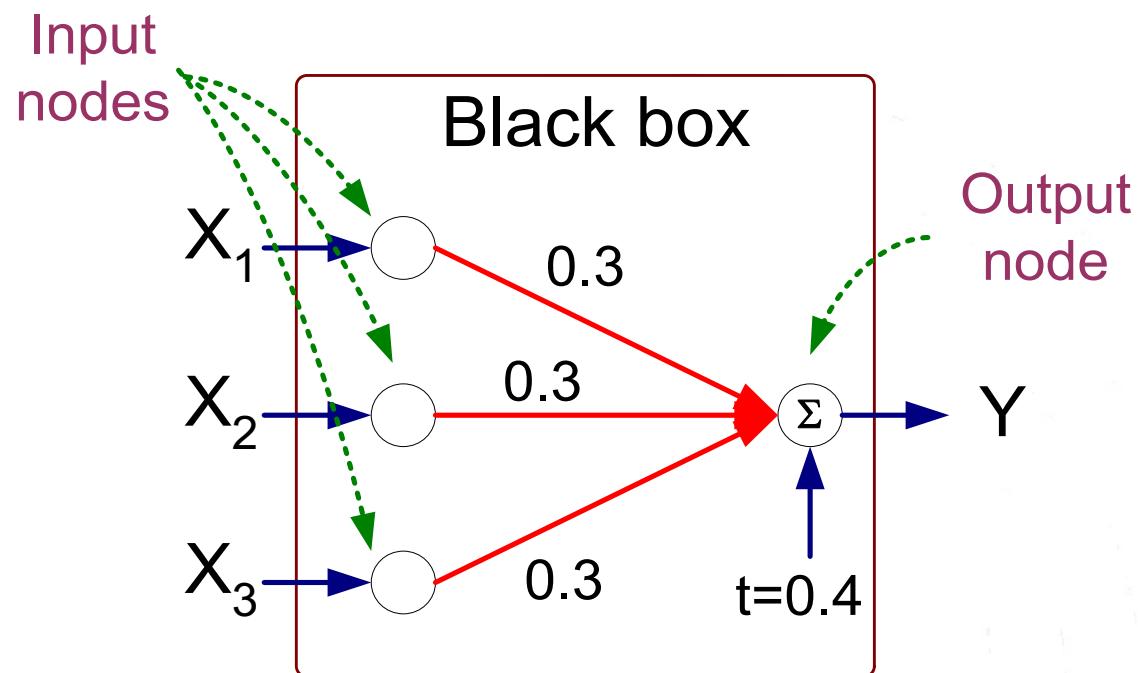
$X_1$	$X_2$	$X_3$	$Y$
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



La salida  $Y$  es 1 si al menos 2 de las 3 entradas son iguales a 1.

# Redes Neuronales

$X_1$	$X_2$	$X_3$	$Y$
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0

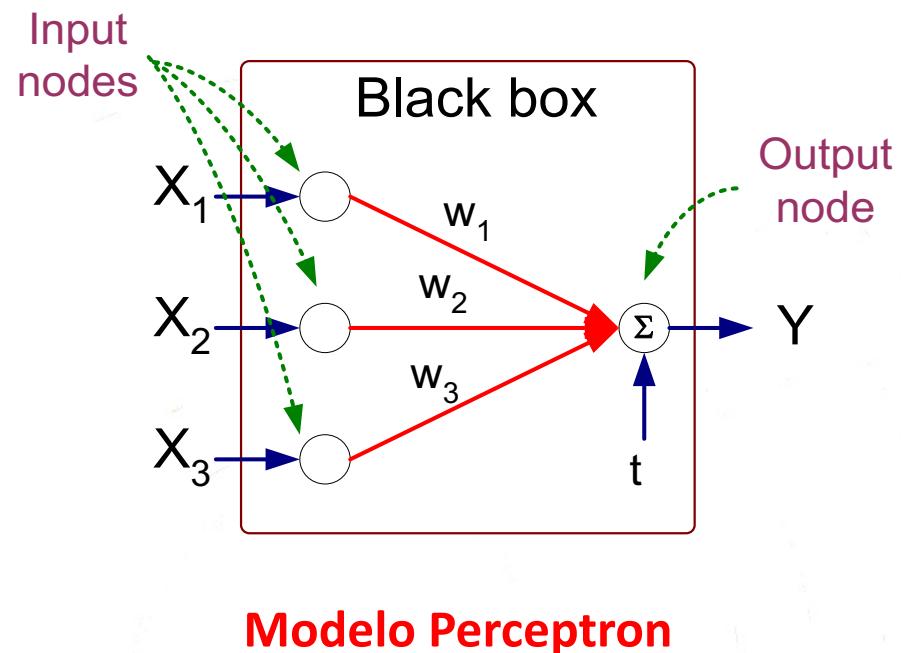


$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 > 0.4)$$

donde  $I(z) = \begin{cases} 1 & \text{Si } z \text{ es verdadero} \\ 0 & \text{En otro caso} \end{cases}$

# Redes Neuronales

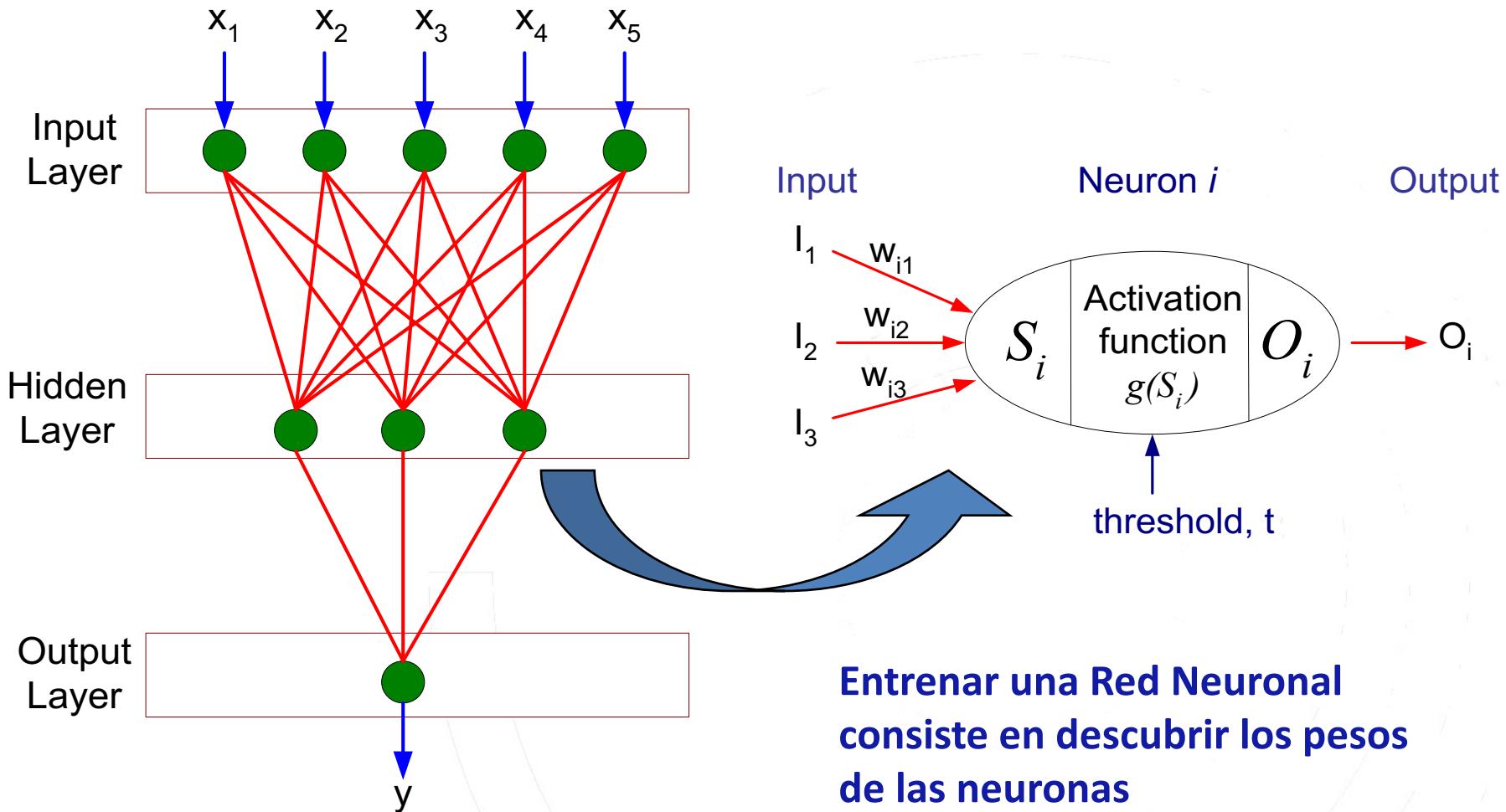
- Modelo es un conjunto de nodos interconectados y enlaces ponderados
- Nodo de salida suma cada uno de su valor de entrada de acuerdo a los pesos de sus vínculos
- Comparar nodo de salida contra un **umbral t**



$$Y = I\left(\sum_i w_i X_i - t\right) \quad o$$

$$Y = sign(\sum_i w_i X_i - t)$$

# Estructura General de una Red Neuronal



# Algunas funciones de Activación

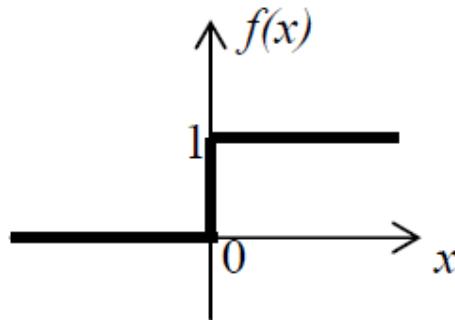
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = \max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

Copyright © Sebastian Raschka 2016  
(<http://sebastianraschka.com>)

# Algunas funciones de Activación:

En todos los ejemplos anteriores hemos usado la función Escalón de Heaviside como función de activación:

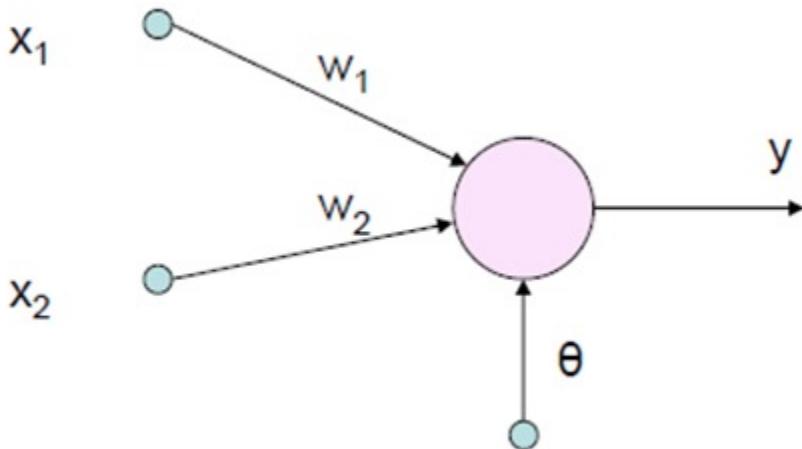
Función paso o  
De Heaviside



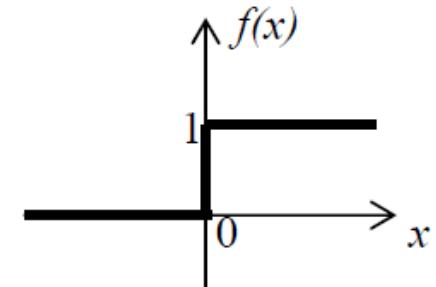
$$H(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}$$

Otra forma de ver Perceptron del OR que es equivalente a la vista anteriormente se muestra en la siguiente filmina:

# Función de activación $H(x)$ =escalón Perceptrón del OR



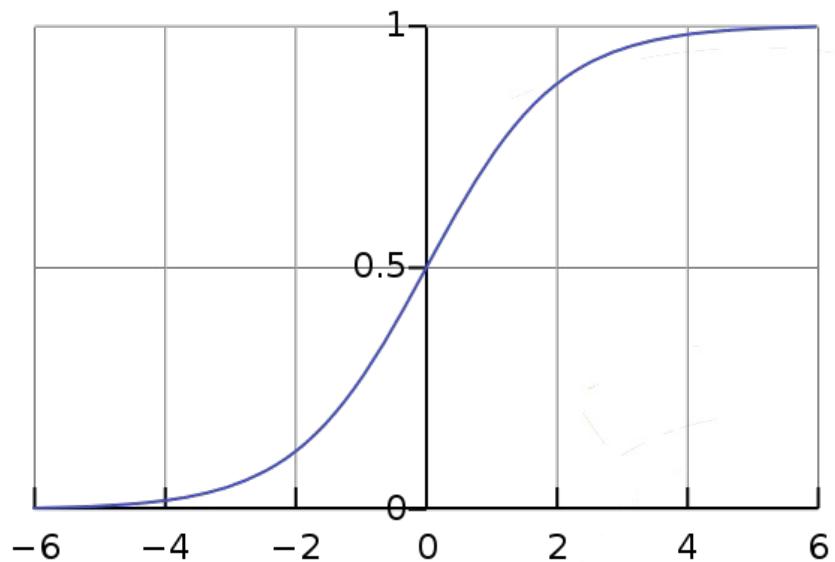
*Función paso o  
De Heaviside*



$$Y = I(z) = H(z) = H(W_1 X_1 + W_2 X_2 - \theta)$$

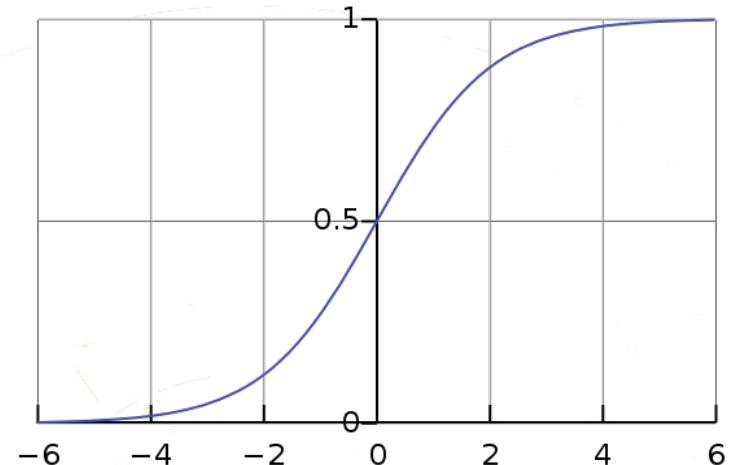
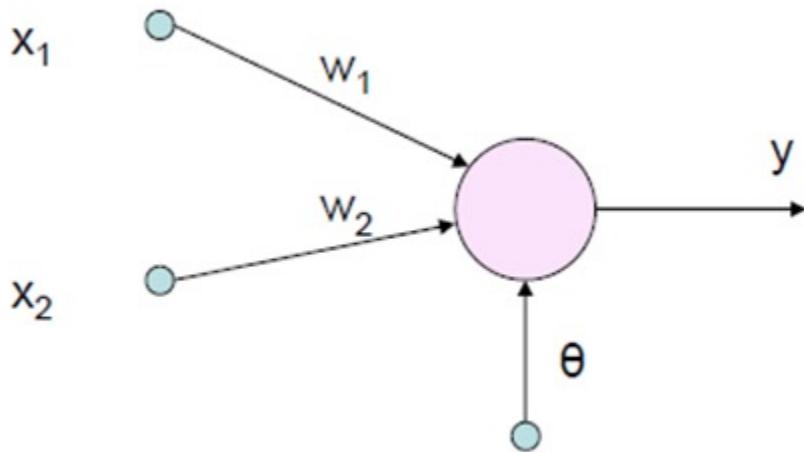
donde  $H(z) = \begin{cases} 1 & \text{Si } z \geq 0 \\ 0 & \text{En otro caso} \end{cases}$

# Función de activación $S(x)$ =Sigmoidea



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

# Función de activación $S(x)$ =Sigmoidea en el Perceptrón del or



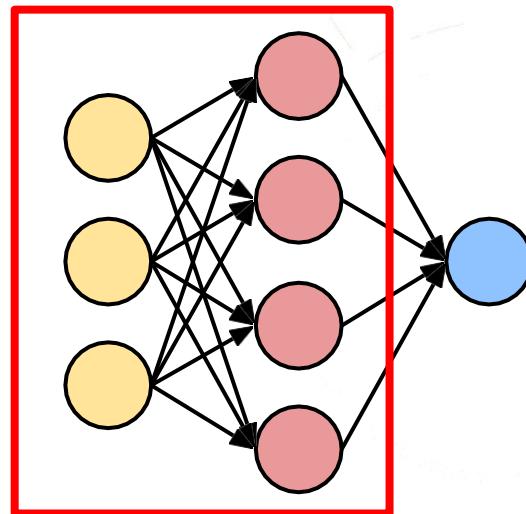
$$S(z) = S(W_1X_1 + W_2X_2 - \theta) = \frac{1}{1 + e^{-(W_1X_1 + W_2X_2 - \theta)}}$$

Como este valor  $S(z)$  queda entre 0 y 1,  $I(S(z))$  asigna 1 si  $S(z) \geq 0.5$  y 0 en caso contrario

# ¿Matricialmente qué hace una sola capa en una red neuronal?

$$g(\mathbf{W}\vec{x} + \vec{b})$$

$g$  es la función de activación



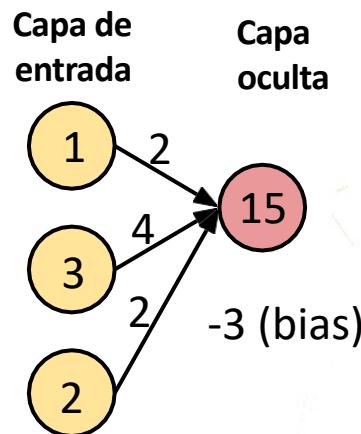
# ¿Matricialmente qué hace una sola capa en una red neuronal?

En primer lugar, implica algunas operaciones de matriz...

$$\begin{pmatrix} 2 & 4 & 2 \\ 1 & 3 & 7 \\ 0 & 7 & 3 \\ 3 & 2 & 5 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix} + \begin{pmatrix} -3 \\ 0 \\ 2 \\ -1 \end{pmatrix} = \begin{pmatrix} 15 \\ 24 \\ 29 \\ 18 \end{pmatrix}$$

$\mathbf{W}$        $\xrightarrow{x}$        $\xrightarrow{b}$

# ¿Matricialmente qué hace una sola capa en una red neuronal?



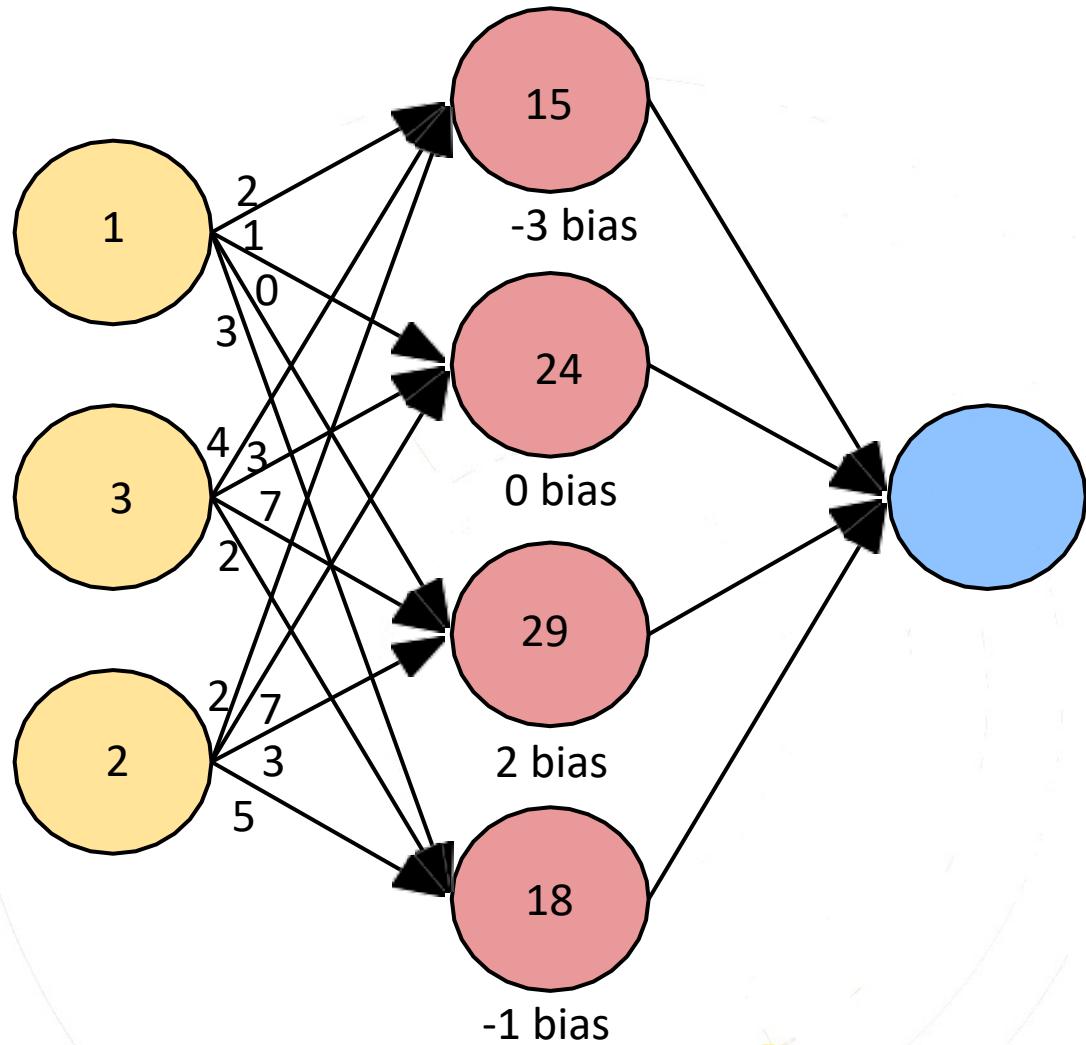
- Cada círculo es un elemento de una capa de entrada u oculta.
- Cada flecha es un peso.

$$g(\vec{w}_1^\top \vec{x} + b_1)$$

# ¿Matricialmente qué hace una sola capa en una red neuronal?

$$g(\mathbf{W}\vec{x} + \vec{b})$$

$g$  es la función de activación, en este caso la identidad.



# ¿Matricialmente qué hace una sola capa en una red neuronal?

Si  $g$  no es la identidad:

$$g \left( \begin{pmatrix} 2 & 4 & 2 \\ 1 & 3 & 7 \\ 0 & 7 & 3 \\ 3 & 2 & 5 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix} + \begin{pmatrix} -3 \\ 0 \\ 2 \\ -1 \end{pmatrix} \right) = g \begin{pmatrix} 15 \\ 24 \\ 29 \\ 18 \end{pmatrix}$$

$\xrightarrow{W}$        $\xrightarrow{x}$        $\xrightarrow{b}$

# ¿Matricialmente qué hace una sola capa en una red neuronal?

Si  $g$  no es la identidad:

$$g \begin{pmatrix} 2 & 4 & 2 \\ 1 & 3 & 7 \\ 0 & 7 & 3 \\ 3 & 2 & 5 \end{pmatrix} W + \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix} b = g \begin{pmatrix} 15 \\ 24 \\ 29 \\ 18 \end{pmatrix}$$

Matriz de Pesos  $W$

Vector de entradas  $x$

Vector bias  $b$

The diagram shows the mathematical operation of a single layer neural network. It consists of three main components: a weight matrix  $W$ , an input vector  $x$ , and a bias vector  $b$ . The weight matrix  $W$  is circled in red, and the input vector  $x$  is circled in blue. The bias vector  $b$  is also circled in red. Red arrows point from the labels 'Matriz de Pesos' and 'Vector bias' to their respective circled components. A blue arrow points from the label 'Vector de entradas' to the circled input vector  $x$ . The final result is the output vector, which is the result of applying the activation function  $g$  to the sum of the weighted inputs and the bias.

# ¿Matricialmente qué hace una sola capa en una red neuronal?

¿Qué se puede entrenar?

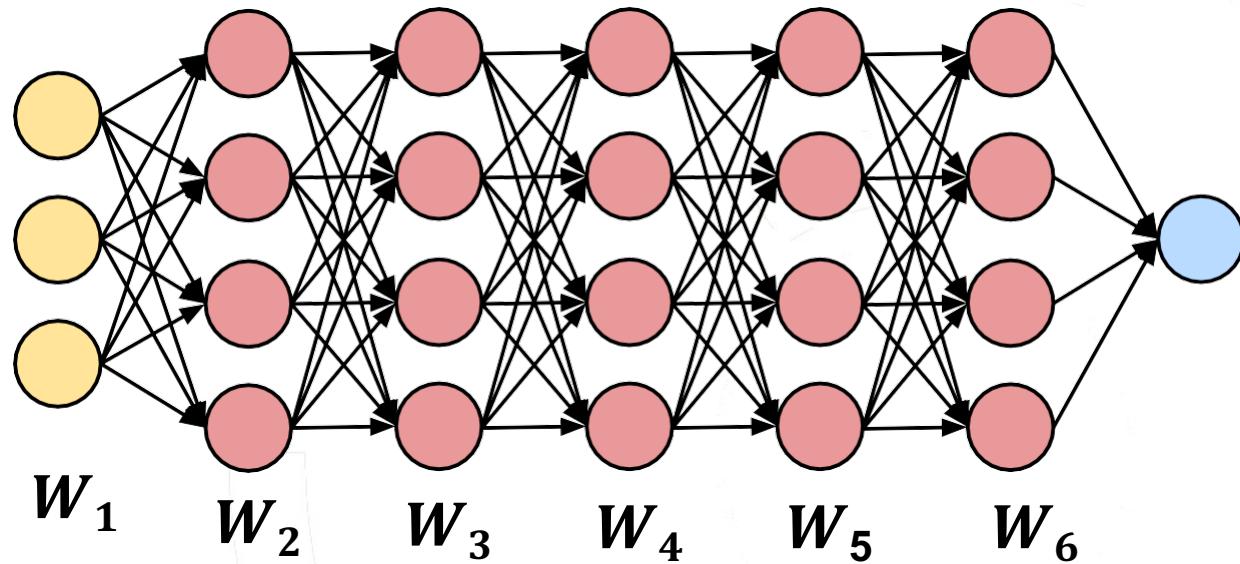
$$g \left( \begin{pmatrix} 2 & 4 & 2 \\ 1 & 3 & 7 \\ 0 & 7 & 3 \\ 3 & 2 & 5 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix} + \begin{pmatrix} -3 \\ 0 \\ 2 \\ -1 \end{pmatrix} \right)$$

$W$        $\xrightarrow{x}$        $\xrightarrow{b}$

Parámetros entrenables

# Apilamiento de varias capas

*Se convierte en una red neuronal profunda (Deep) cuando se utilizan muchas capas*



## Pesos para cada neurona

# Algoritmo de Aprendizaje

- Inicializar los pesos  $w=(w_0, w_1, \dots, w_k)$
- Ajustar los pesos de tal manera que la salida de la red Neuronal sea consistente con etiquetas de clase en los casos de entrenamiento:
- Función Objetivo:  $E(w) = \sum_i [Y_i - f(w_i, X_i)]^2$ 
  - Encuentra el peso  $w_i$ 's de que minimizan la función objetivo anterior (Error Cuadrático)
  - Ej. Algoritmo “backpropagation”

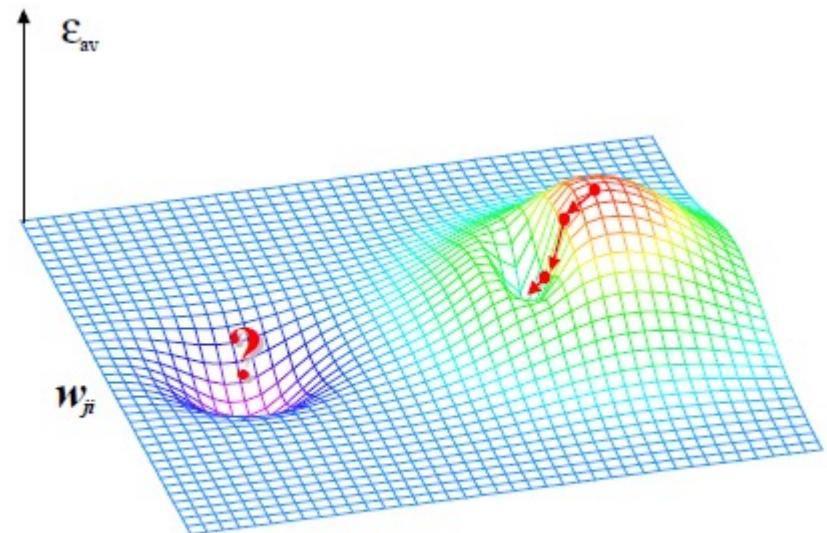
# Descenso del Gradiente y Mínimos locales

Se debe minimizar la función :

$$E(w) = \sum_i [Y_i - f(w_i, X_i)]^2$$

Usualmente se hace utilizando el descenso del gradiente :

$$w_j \leftarrow w_j - \lambda \frac{\partial E(w)}{\partial (w_j)}$$



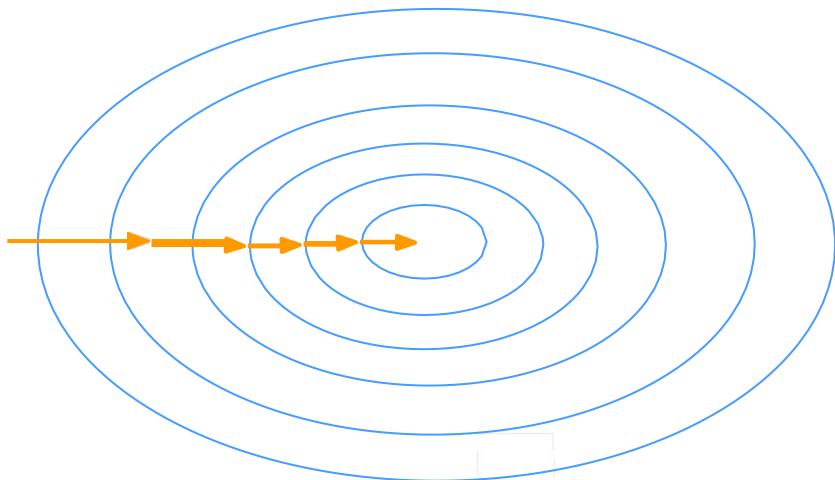
Peligro de caer en mínimo local.

# Descenso estocástico del gradiente

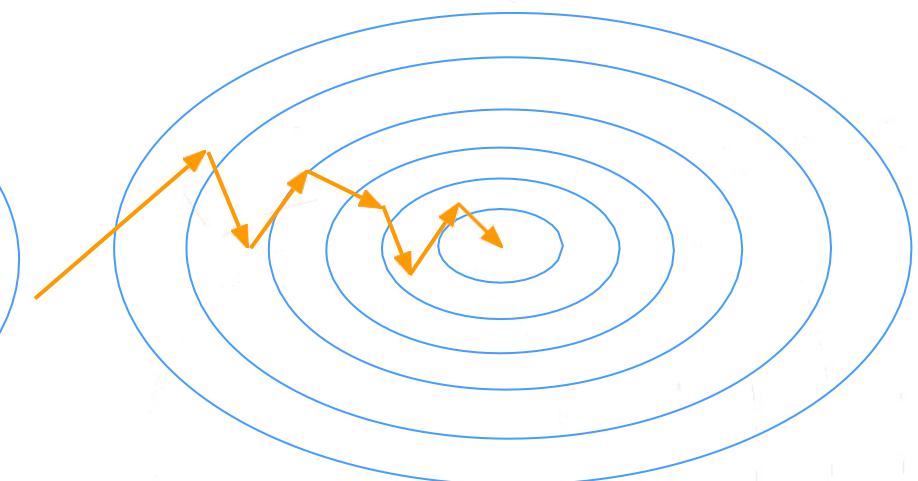
- En el método del gradiente clásico se estima el gradiente evaluándolo solo en un pequeño subconjunto de los ejemplos de entrenamiento.
- El tamaño de este subconjunto es el **batch size**, que se convierte en otro hiper-parámetro.
- En Descenso Estocástico del gradiente se usa un subconjunto diferente para cada paso, con el fin de ir a través de todos (la mayoría) los casos de entrenamiento. Pasar por todo el conjunto de entrenamiento una vez se llama uno **epoch** de entrenamiento.

# Descenso estocástico del gradiente

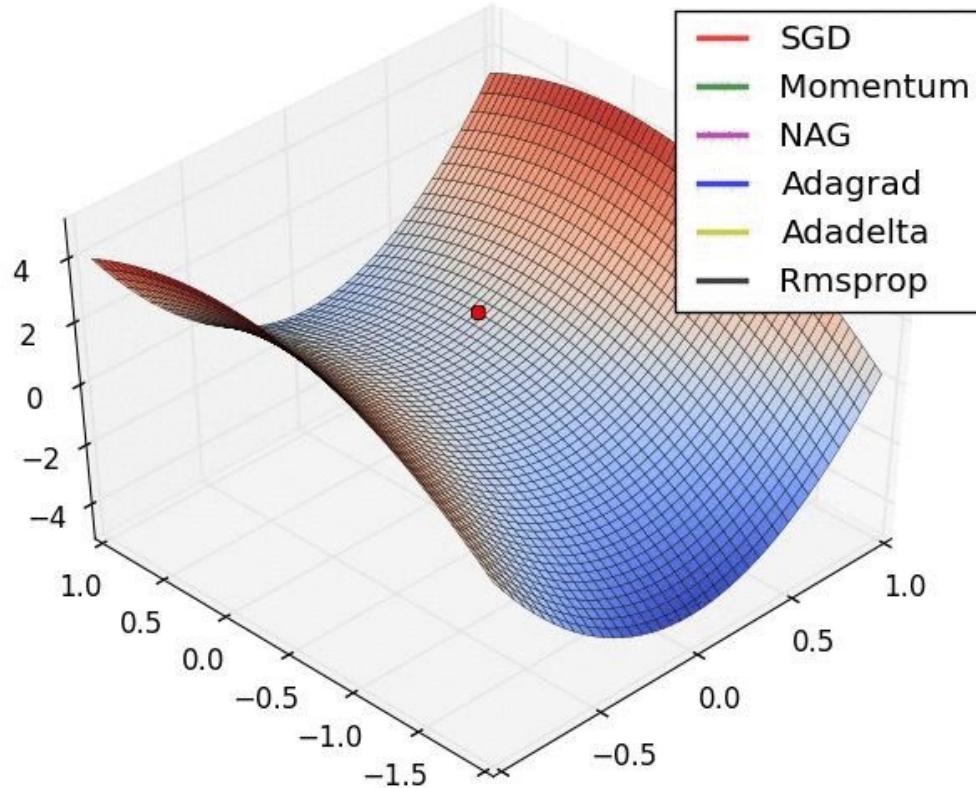
Descenso de gradiente



Descenso estocástico del gradiente



# Otros algoritmos de optimización



- “Adam” es un optimizador muy comúnmente utilizado.
- Esto es un parámetro más del modelo.

# ¿Cuándo dejar de entrenar la red neuronal?

- Hasta la convergencia, cuando la pérdida o función de costo ya no cambia mucho de un paso al otro .
- En la práctica, a menudo para un número fijo de epochs.

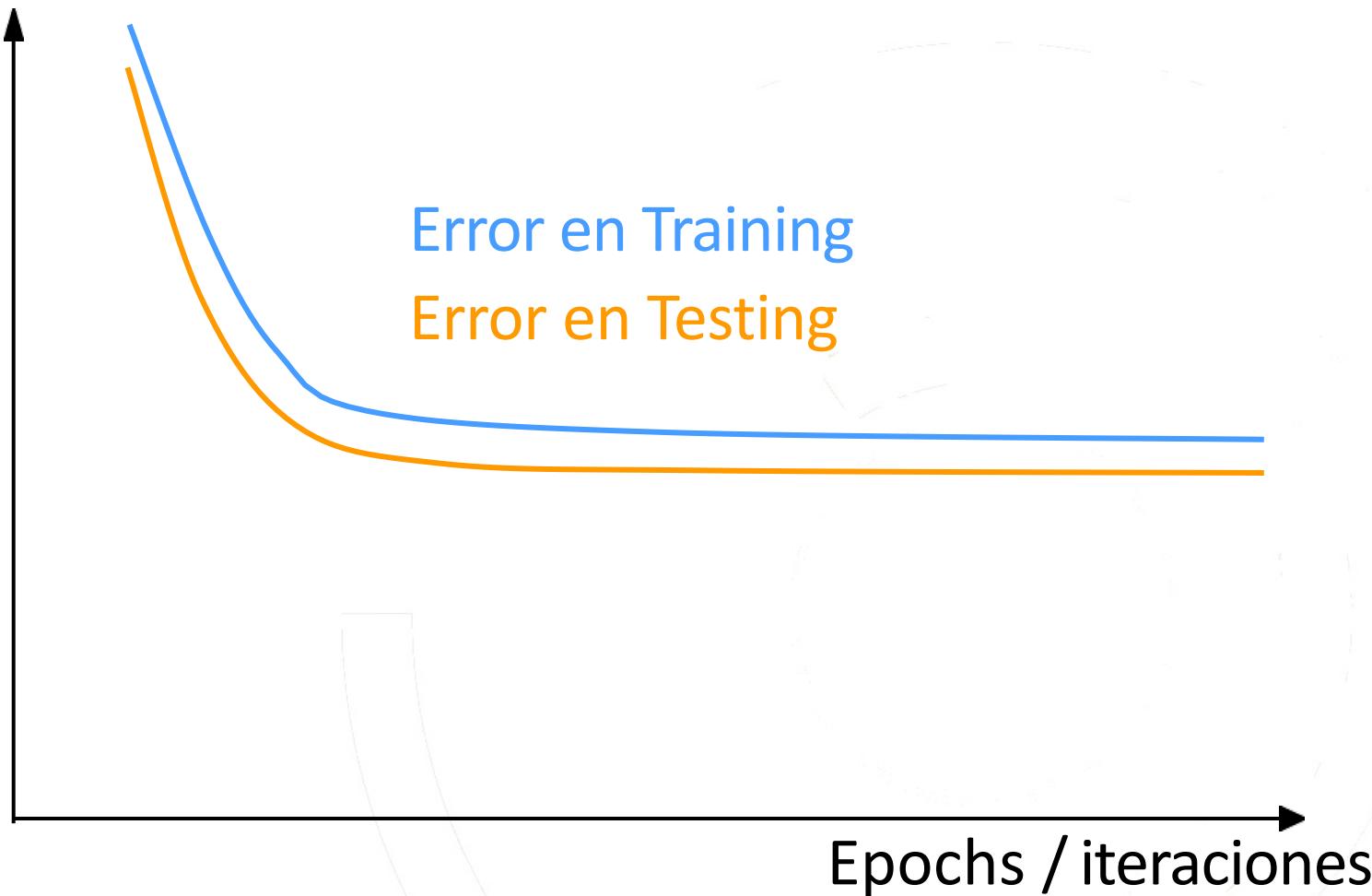
# Overfitting y underfitting

## (Sobreajuste y subajuste)

- **Underfitting:** Se da cuando el modelo no es lo suficientemente complejo como para capturar la estructura en los datos. La capacidad del modelo es insuficiente. Tanto el error en la tabla de entrenamiento como en la tabla de testing son altos.
- **Overfitting:** Se da cuando el modelo tiene tanta capacidad que memoriza los datos de entrenamiento. Normalmente indica una cantidad insuficiente de datos de entrenamiento en relación con la complejidad del modelo.

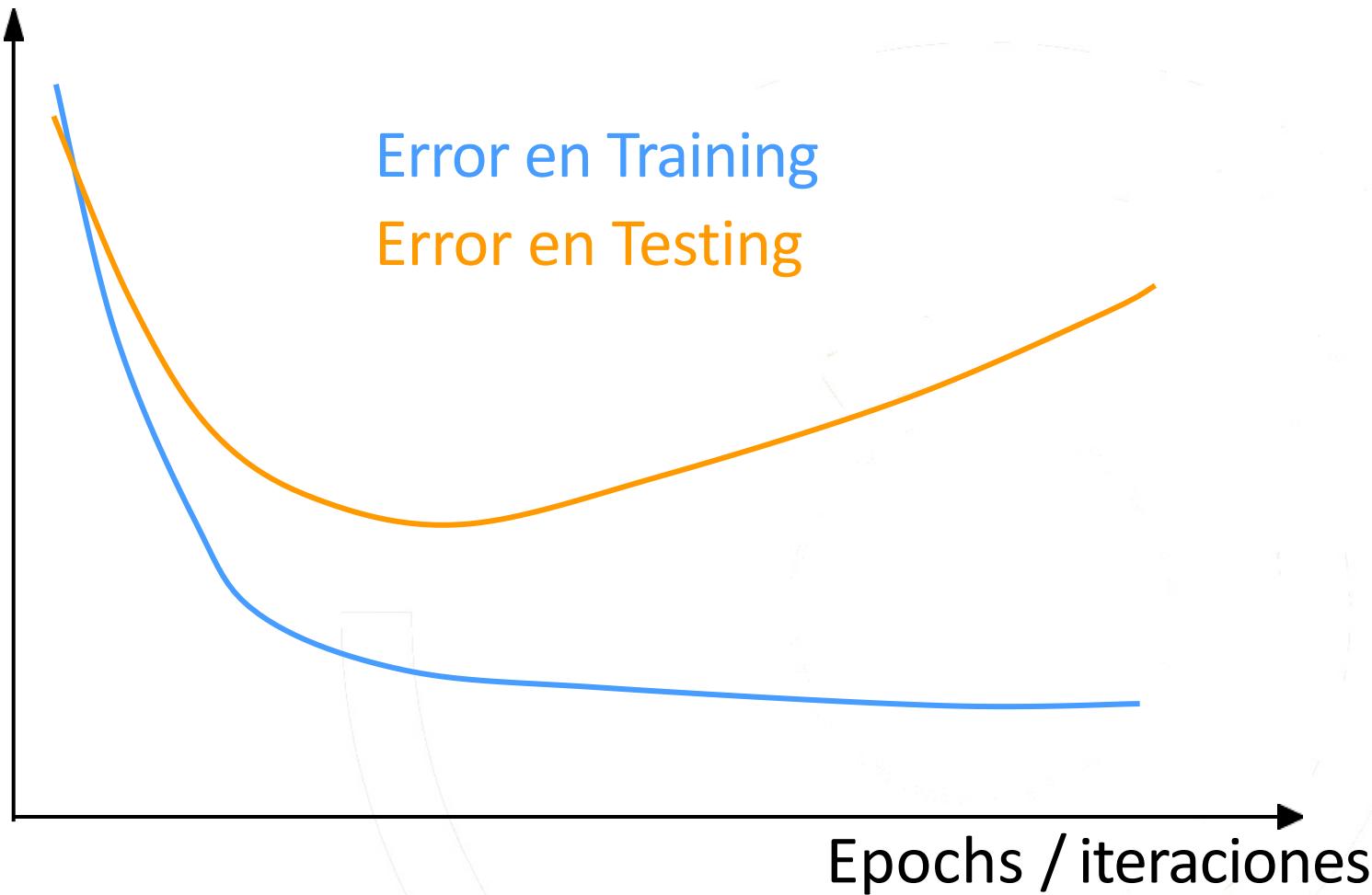
# Underfitting

Error

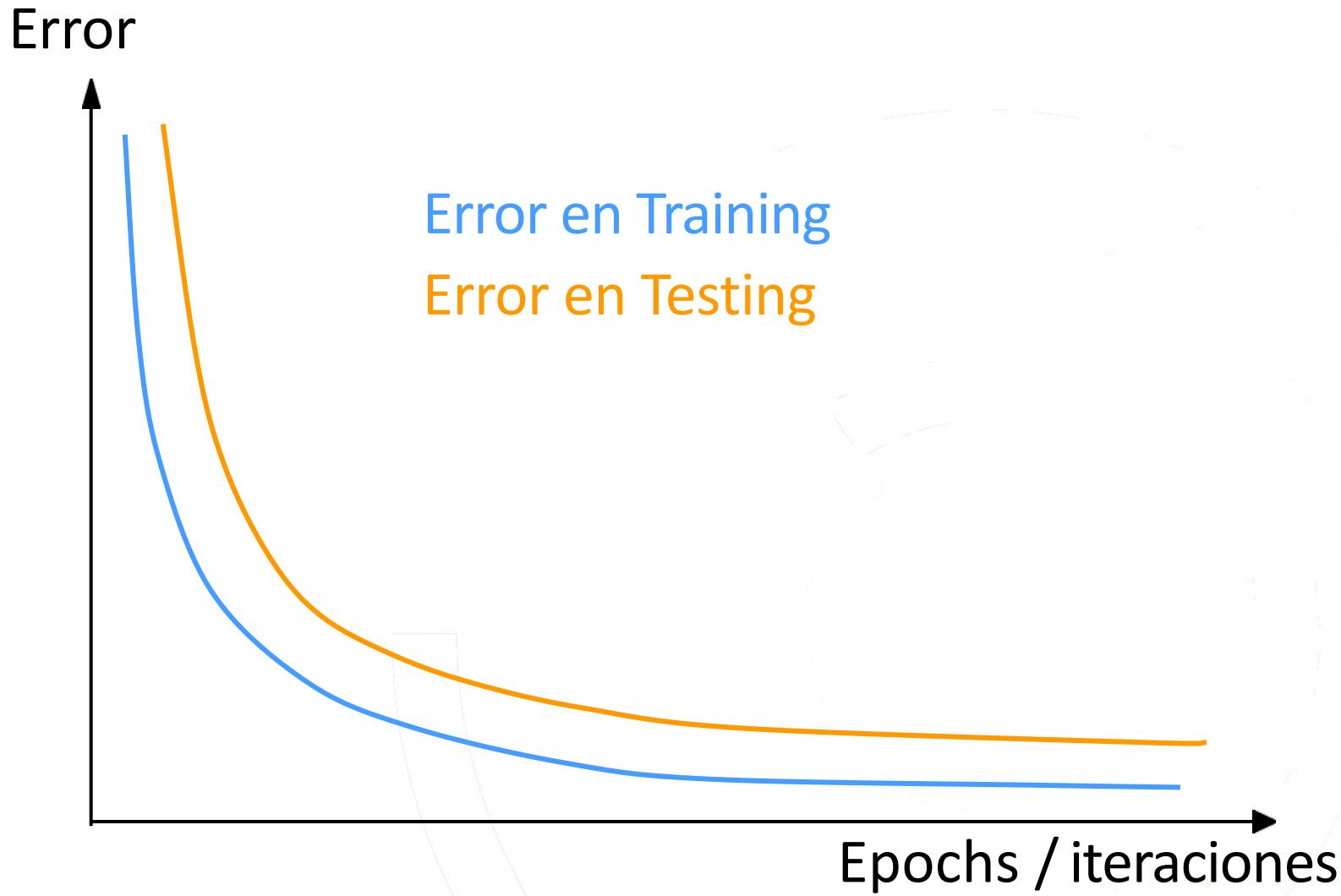


# Overfitting

Error

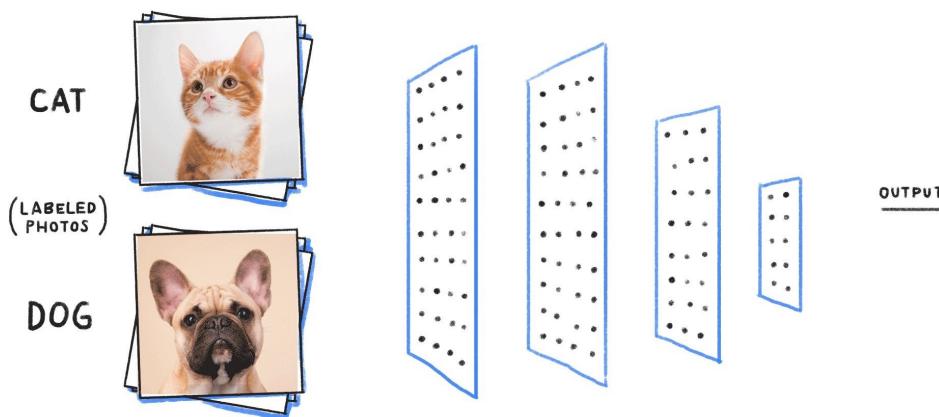


# Entrenamiento adecuado



# Definición del problema

- Si  $y$  es una salida discreta, puede utilizar la codificación “one-hot”.
- Si  $y$  es una variable continua, puede ser salida por la red directamente.
- Clases = {cat, dog}.



$$y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

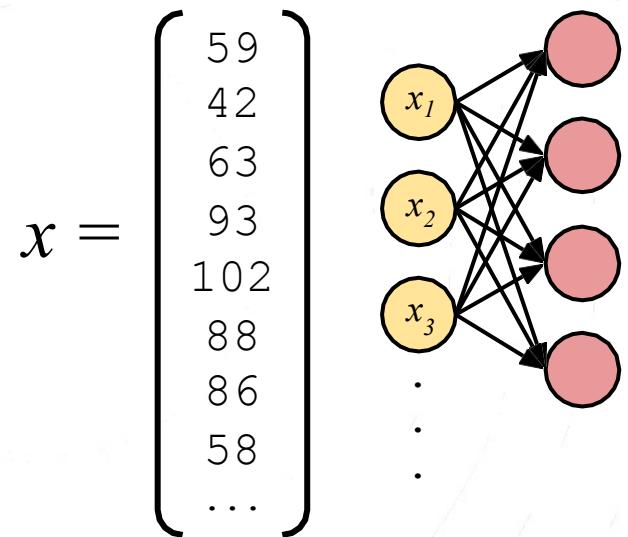
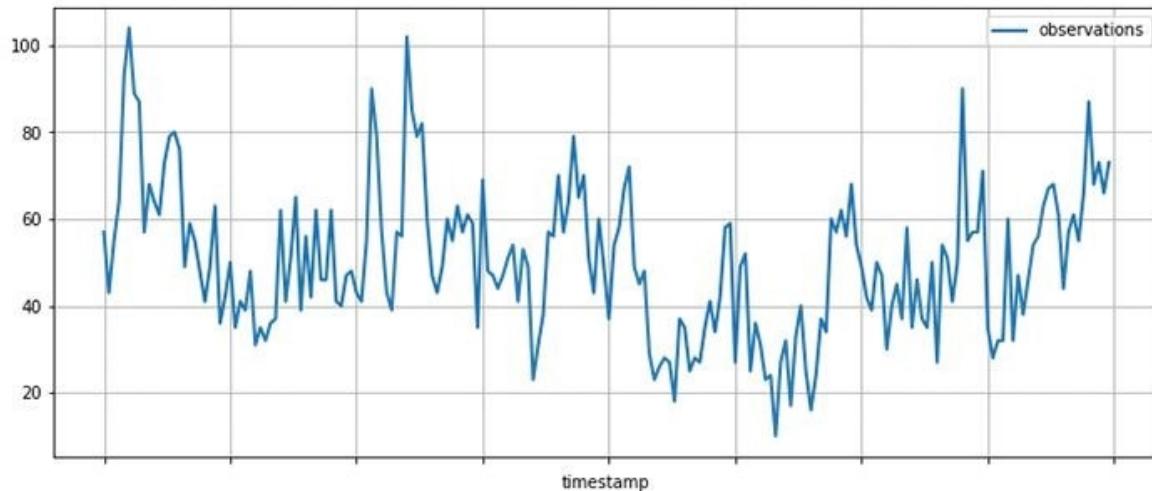
"cat"

$$y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

"dog"

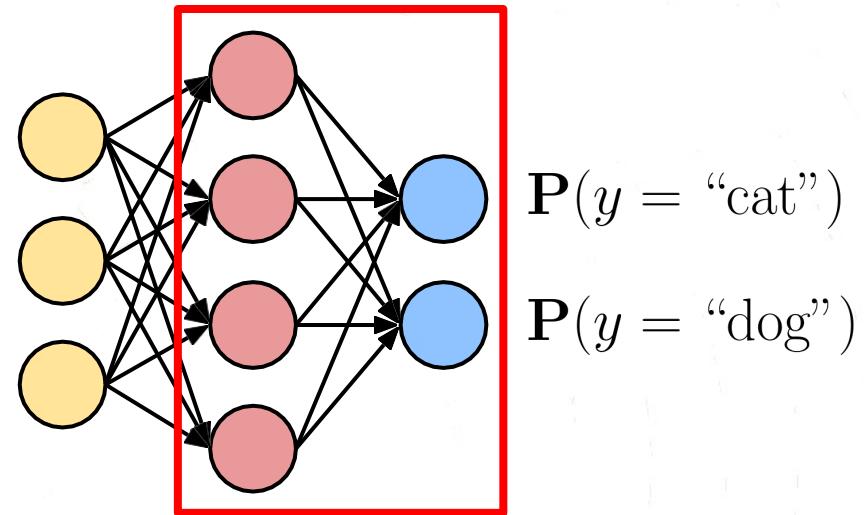
# Definición del problema

1. Si  $x$  es una salida discreta, puede utilizar la codificación “one-hot”.
2. Si  $x$  es una variable continua, se puede introducir directamente en la red.



# Función de activación en la última capa

- Para la clasificación, cada neurona en la salida representa una clase de salida.
- La salida representa la probabilidad de que la entrada pertenezca a cada clase.
- ¿Qué restricciones debe satisfacer la salida?



# Función de activación en la última capa

*¿Qué restricciones debe satisfacer la salida?*

- 1. Las probabilidades son positivas.*
- 2. Las probabilidades suman 1.*

$$\begin{aligned}\mathbf{P}(y = \text{"cat"}) &\geq 0 \\ \mathbf{P}(y = \text{"dog"}) &\geq 0\end{aligned}$$

$$\mathbf{P}(y = \text{"cat"}) + \mathbf{P}(y = \text{"dog"}) = 1$$

# Ventajas y Desventajas de las Redes Neuronales

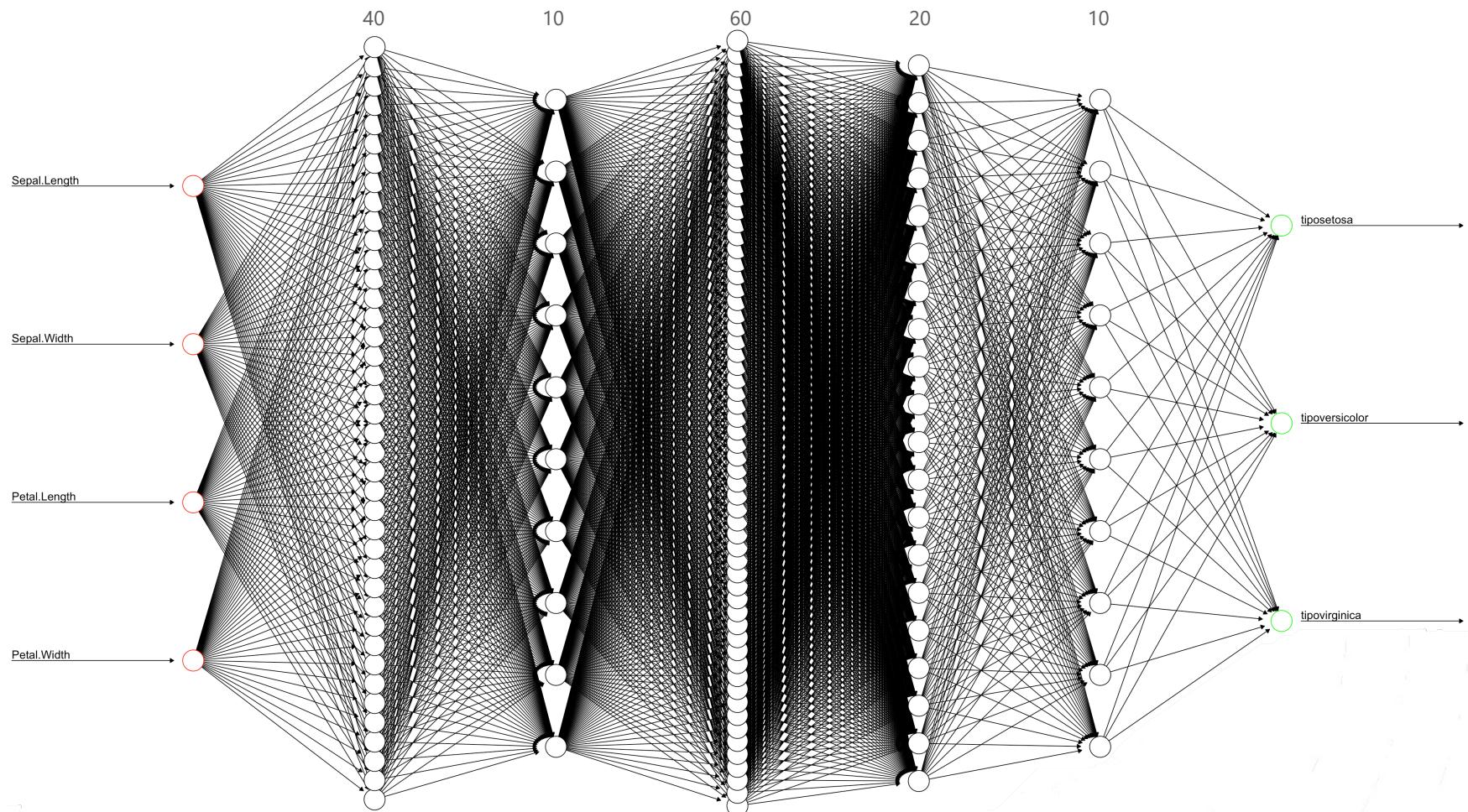
- **Ventaja:** Frontera de decisión no lineal
- **Desventajas:**
  - Convergencia muy lenta
  - Muchos óptimos locales, lo cual hace el método muy inestable
  - La mejor estructura de la red es desconocida
  - Difícil de manejar variables nominales



# Variables Cualitativas

- Por defecto estos modelos no aceptan variables cualitativas.
- Lo correcto es pasarlas a *variables dummy* antes de aplicar los métodos.

# Red Neuronal IRIS





# oldemar rodíguez

CONSULTOR en M1NER14 D3 D4T0S

*Gracias....*