

Redes Neuronales Artificiales

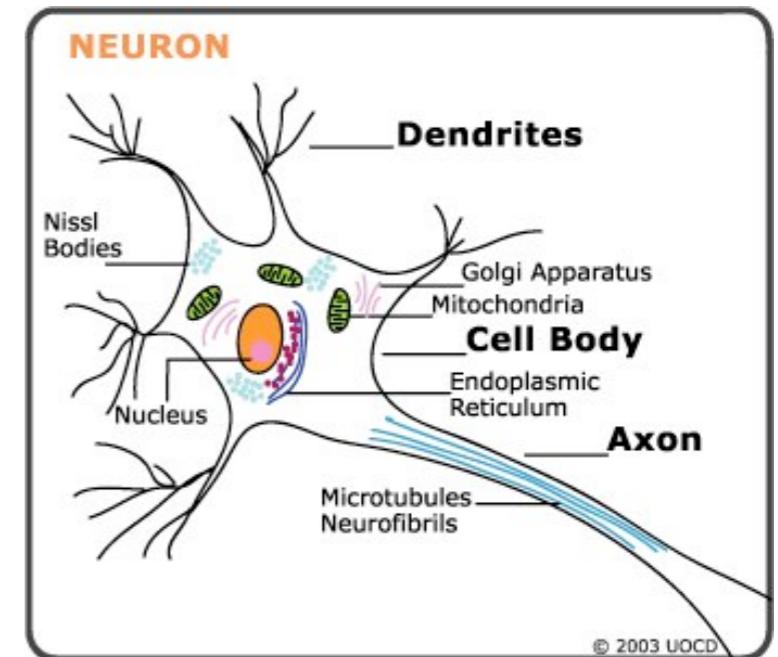


<https://github.com/biuse/2021 Neural Networks>

REDES NEURONALES ARTIFICIALES

Herramienta de *machine learning* que nació con dos objetivos:

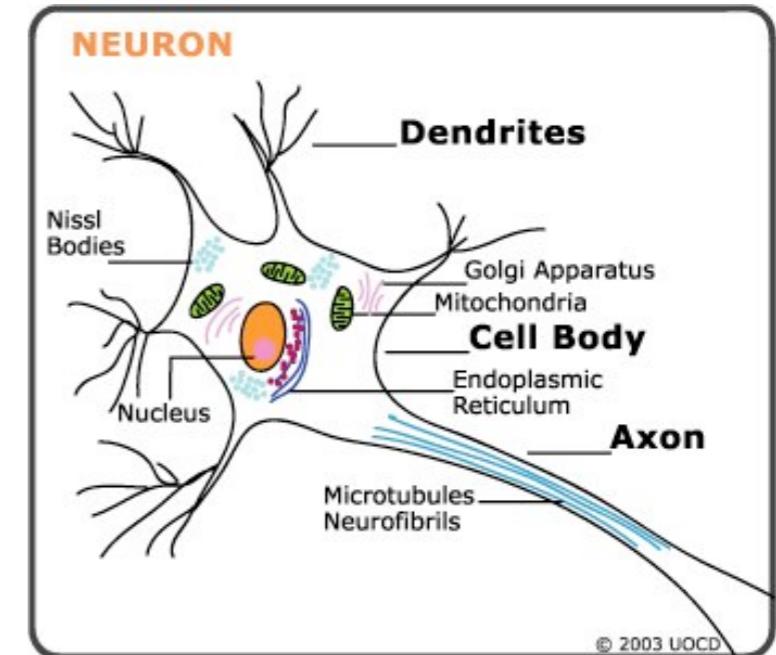
- Modelizar las neuronas biológicas
- Inteligencia artificial



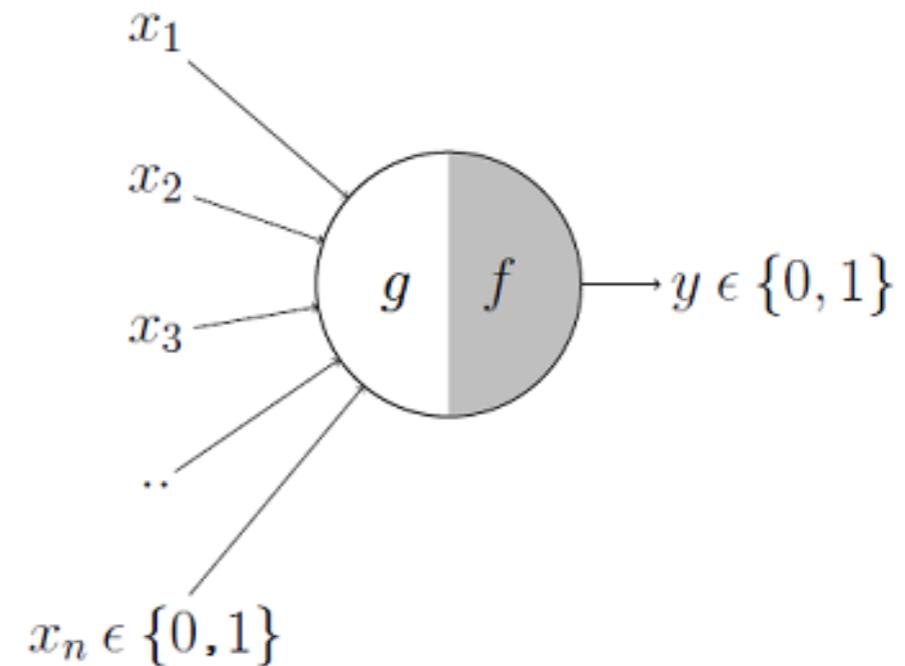
REDES NEURONALES ARTIFICIALES

Herramienta de *machine learning* que nació con dos objetivos:

- Modelizar las neuronas biológicas
- Inteligencia artificial



Neurona artificial recibe
una señal y devuelve 0 o 1

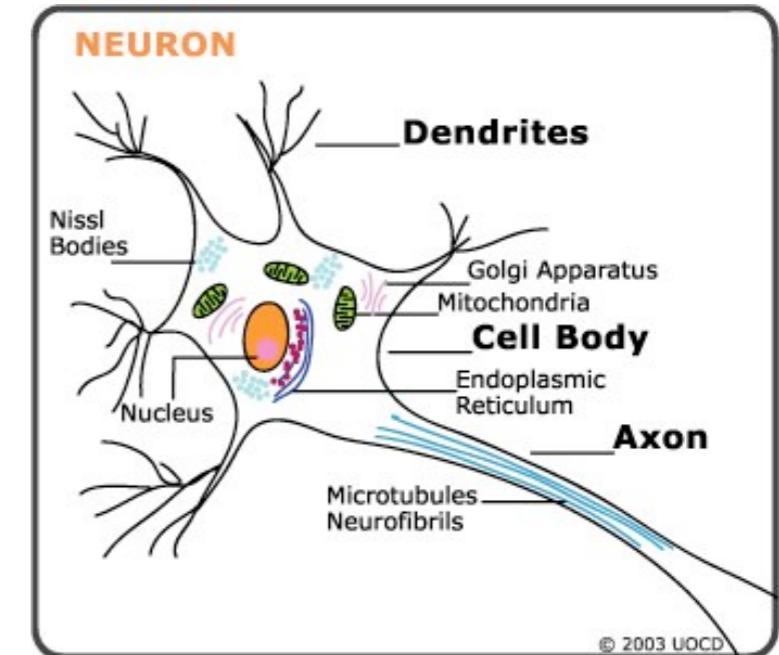


Modelo de McCulloch and Pitts 1943

REDES NEURONALES ARTIFICIALES

Herramienta de *machine learning* que nació con dos objetivos:

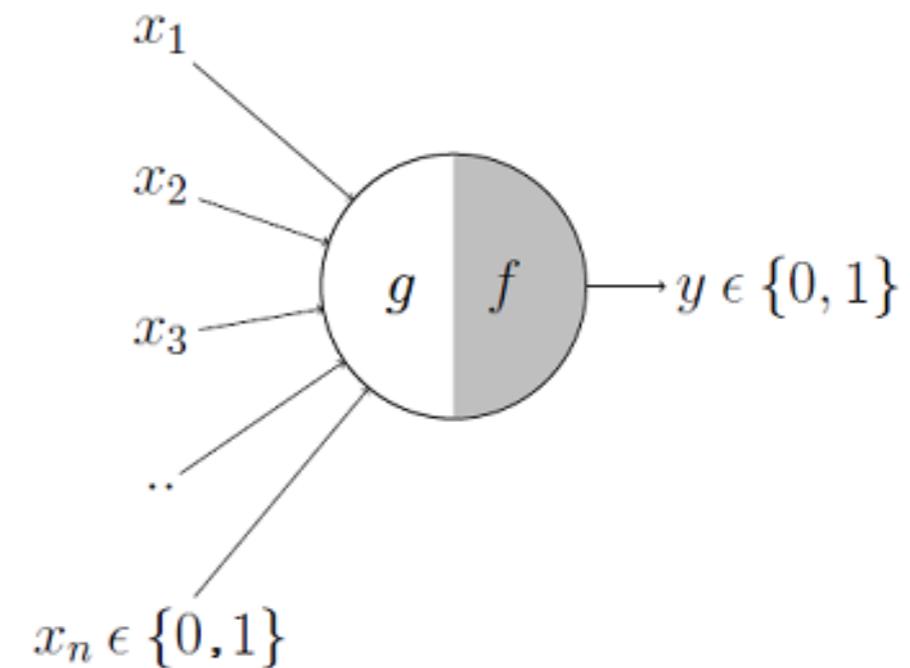
- Modelizar las neuronas biológicas
- Inteligencia artificial



Neurona artificial recibe
una señal y devuelve 0 o 1

$$\begin{aligned} f(g(\mathbf{x})) &= 1 \quad \text{if} \quad g(\mathbf{x}) \geq \theta \\ &= 0 \quad \text{if} \quad g(\mathbf{x}) < \theta \end{aligned}$$

$$g(\mathbf{x}) = \sum_{i=1}^n x_i$$

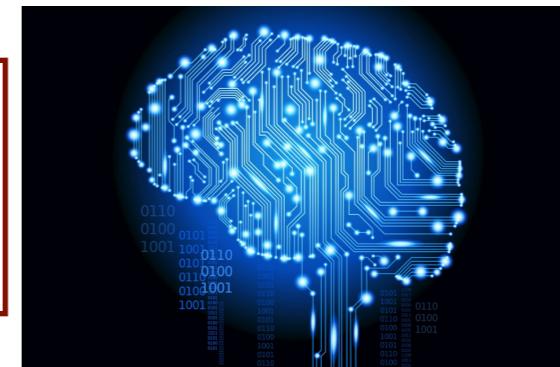


Modelo de McCulloch and Pitts 1943

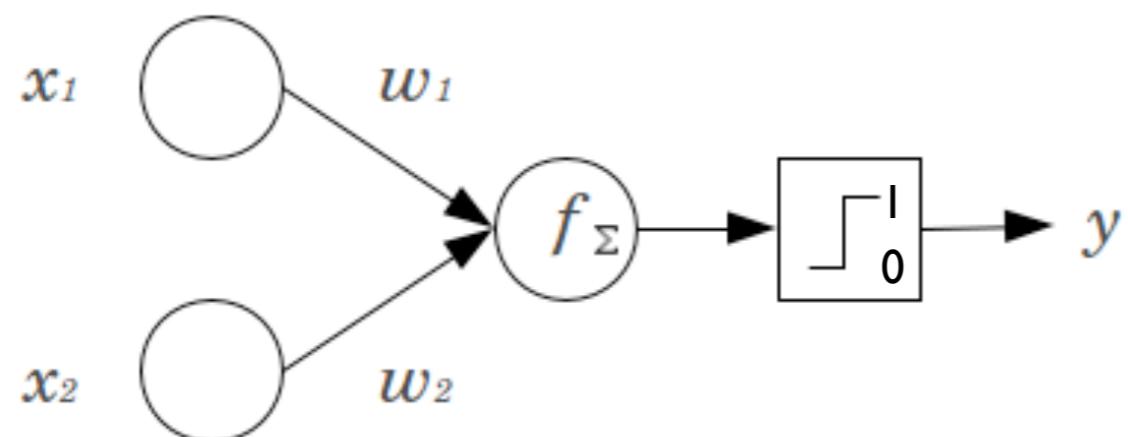
REDES NEURONALES

Herramienta de *machine learning* que nació con dos objetivos:

- modelizar las neuronas biológicas
- la inteligencia artificial



PERCEPTRON

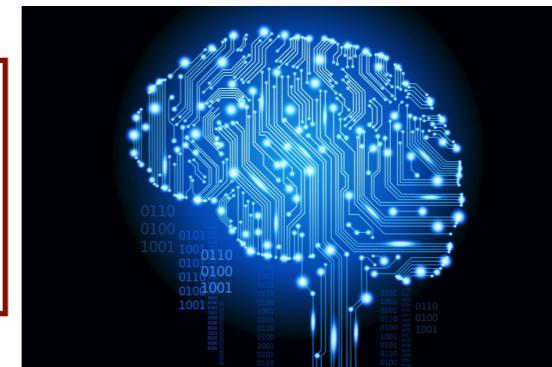


Rosenblatt 1958

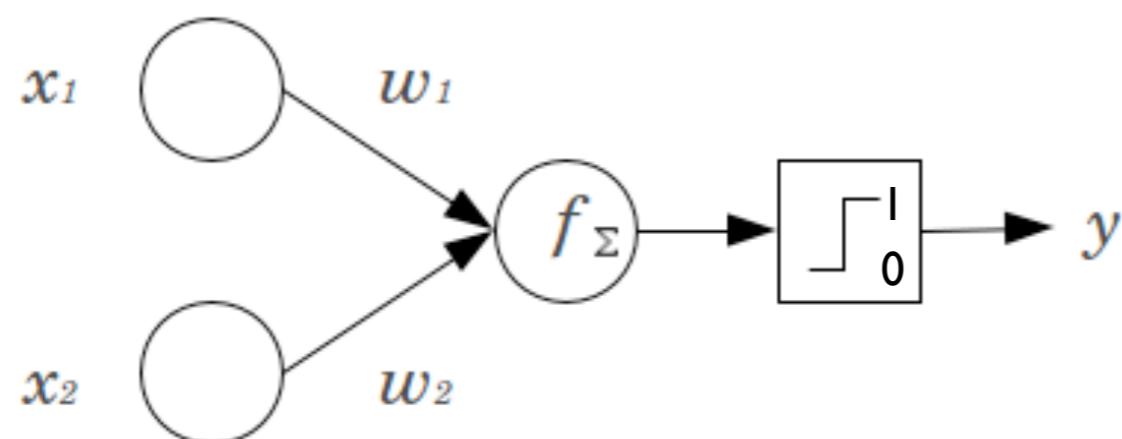
REDES NEURONALES

Herramienta de *machine learning* que nació con dos objetivos:

- modelizar las neuronas biológicas
- la inteligencia artificial



PERCEPTRON



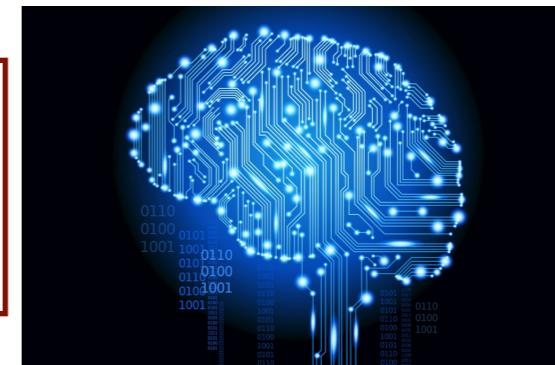
Rosenblatt 1958

Se añadieron unos **pesos**
que podían ajustarse.

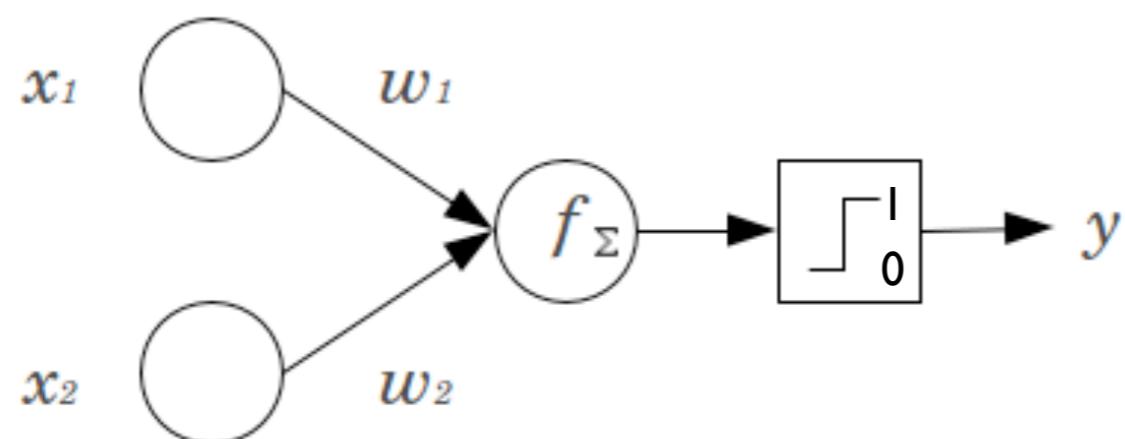
REDES NEURONALES

Herramienta de *machine learning* que nació con dos objetivos:

- modelizar las neuronas biológicas
- la inteligencia artificial



PERCEPTRON



Rosenblatt 1958

Se añadieron unos **pesos**
que podían ajustarse.

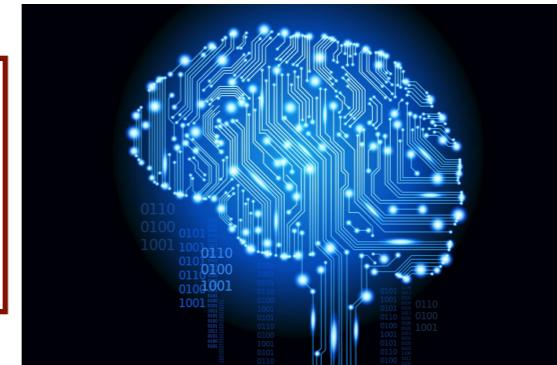
$$\begin{aligned} f(g(\mathbf{x})) &= 1 & \text{if} & \quad g(\mathbf{x}) \geq \theta \\ &= 0 & \text{if} & \quad g(\mathbf{x}) < \theta \end{aligned}$$

$$g(x) = \sum_i w_i x_i$$

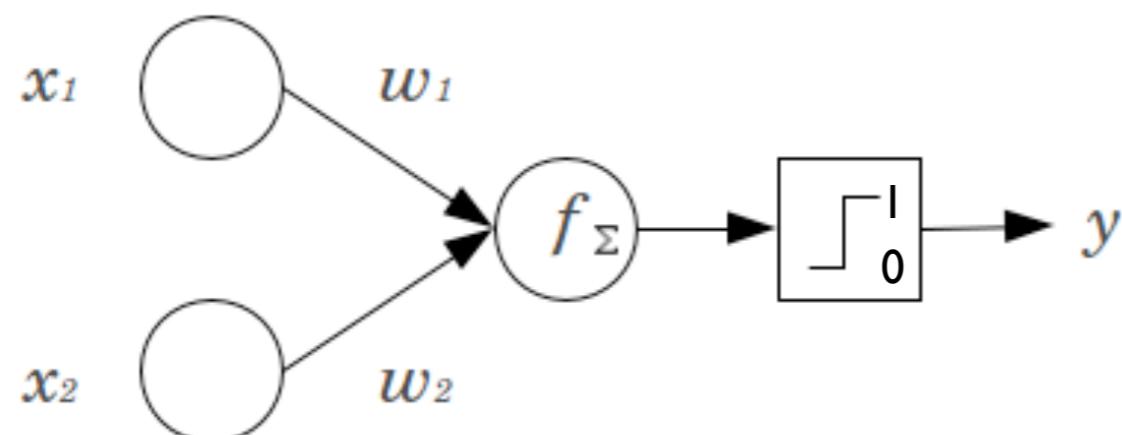
REDES NEURONALES

Herramienta de *machine learning* que nació con dos objetivos:

- modelizar las neuronas biológicas
- la inteligencia artificial



PERCEPTRON



Rosenblatt 1958

Se añadieron unos **pesos**
que podían ajustarse.

¡Y funcionó!

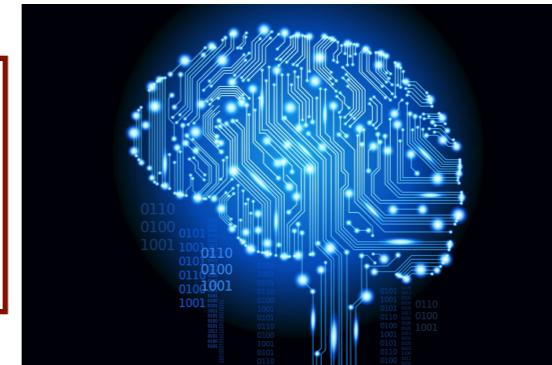
$$\begin{aligned} f(g(\mathbf{x})) &= 1 & \text{if} & \quad g(\mathbf{x}) \geq \theta \\ &= 0 & \text{if} & \quad g(\mathbf{x}) < \theta \end{aligned}$$

$$g(x) = \sum_i w_i x_i$$

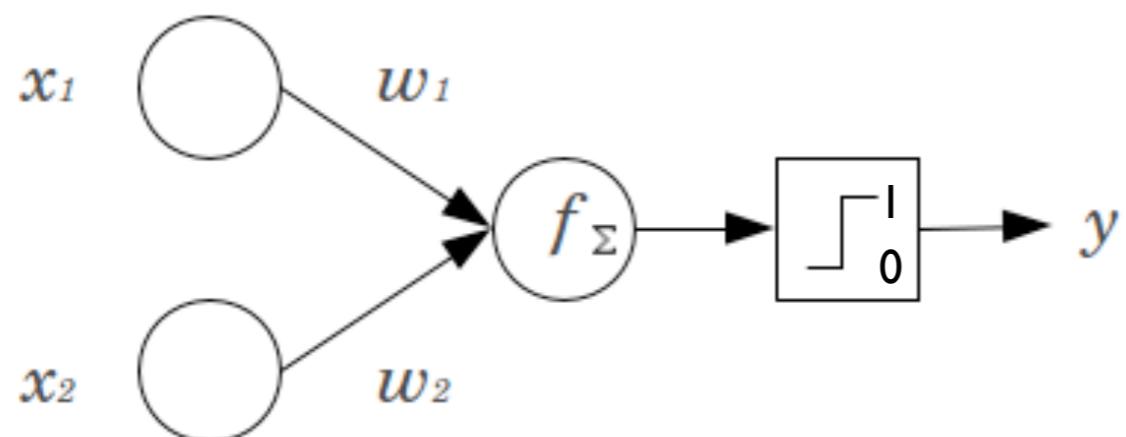
REDES NEURONALES

Herramienta de *machine learning* que nació con dos objetivos:

- modelizar las neuronas biológicas
- la inteligencia artificial



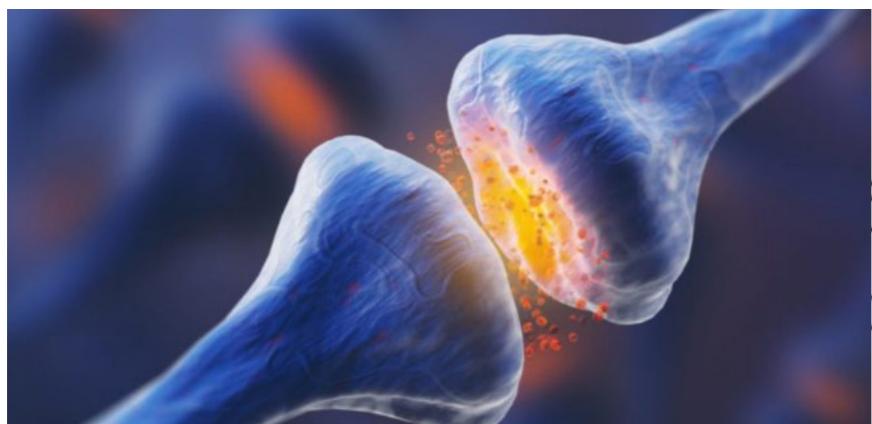
PERCEPTRON



Rosenblatt 1958

Se añadieron unos **pesos**
que podían ajustarse.

¡Y funcionó!

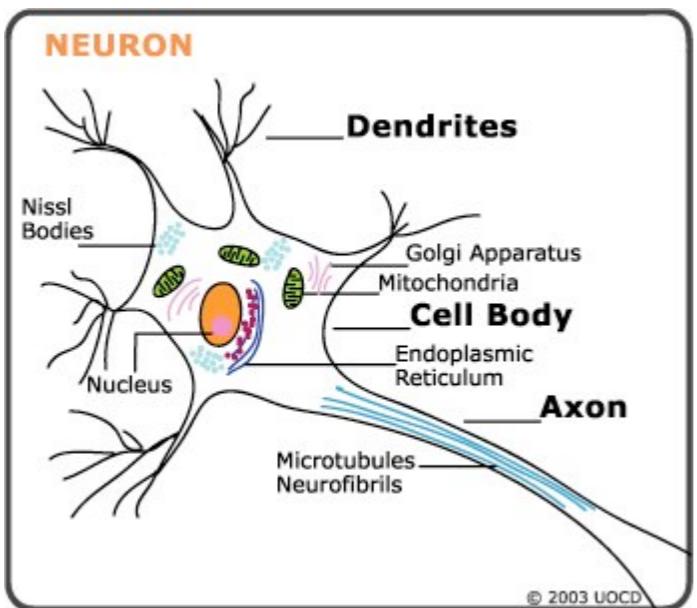


La **sinápsis**, la fortaleza de la conexión entre neuronas,
era fundamental para el aprendizaje

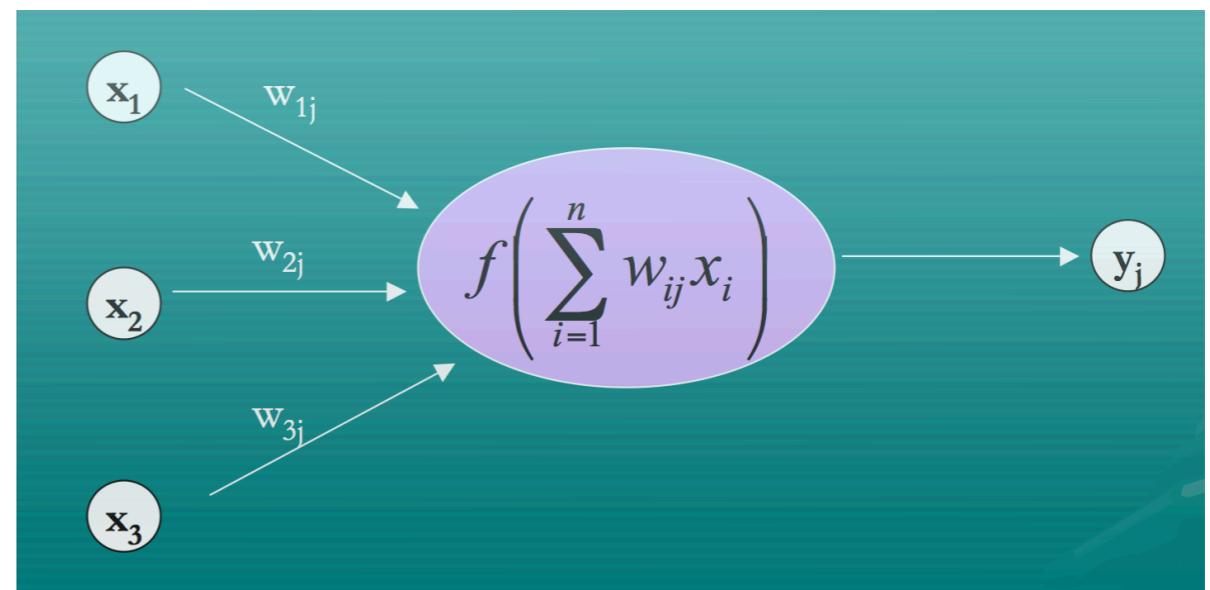
← ↑ ↗
 i

REDES NEURONALES

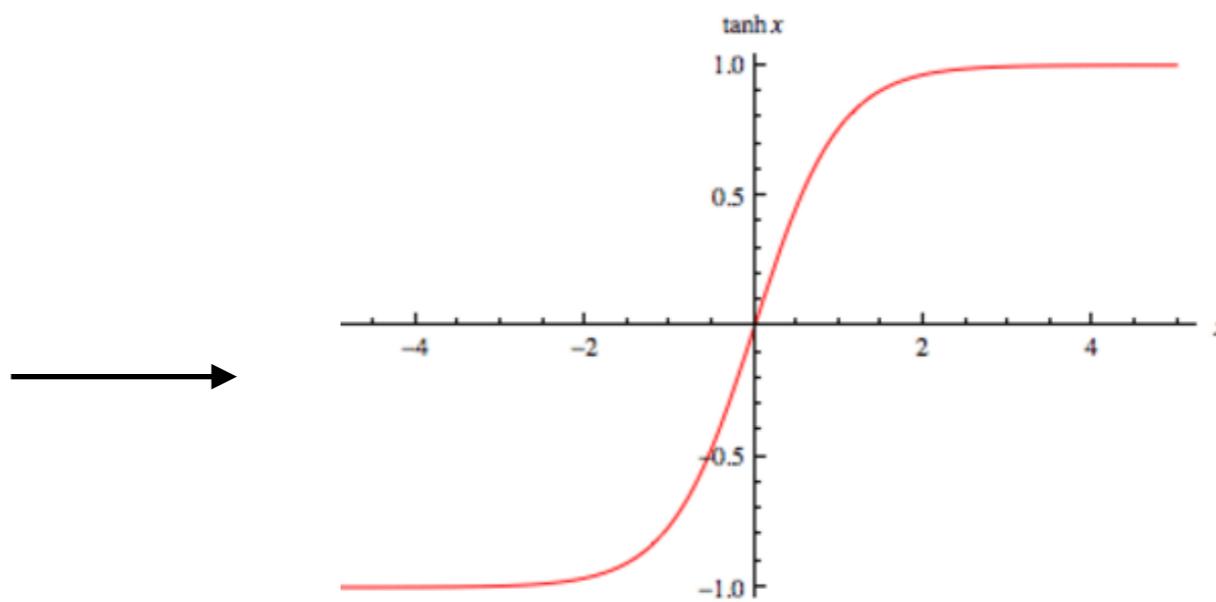
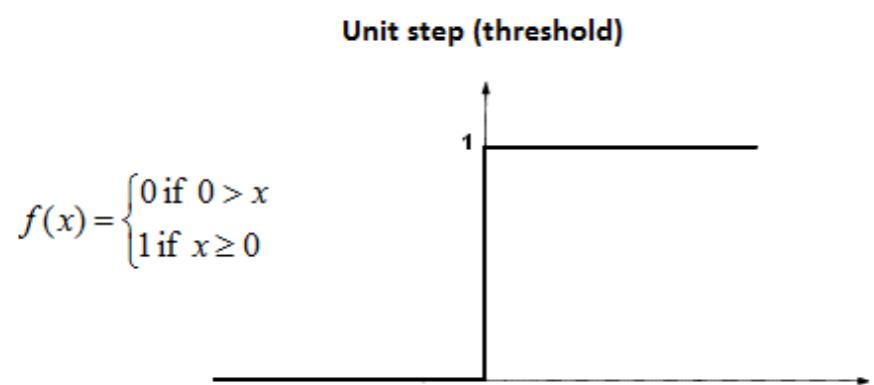
Neurona biológica (simplificación)



Neurona artificial

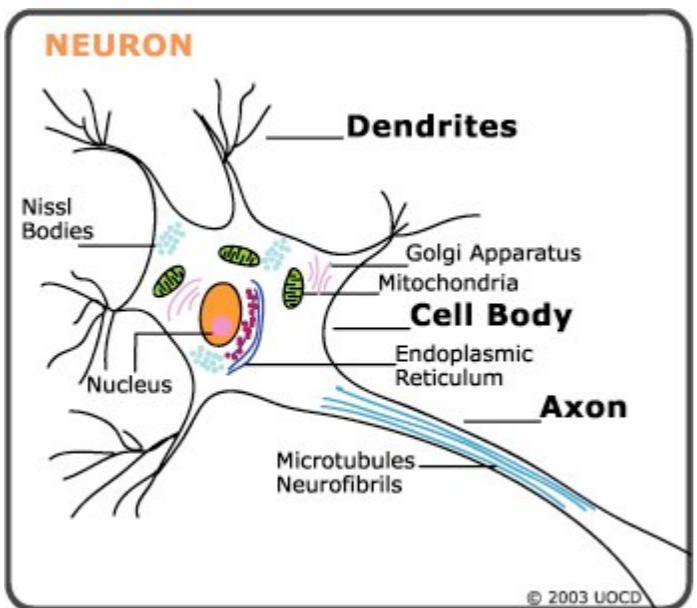


Se generalizó step function —> función de activación $f(x)$

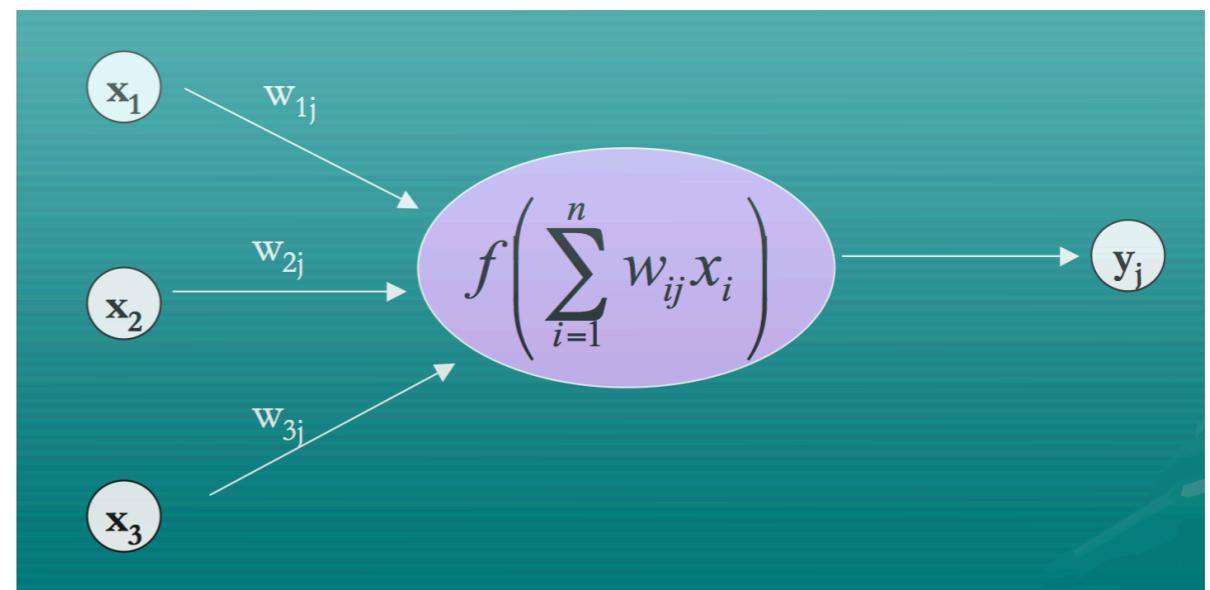


REDES NEURONALES

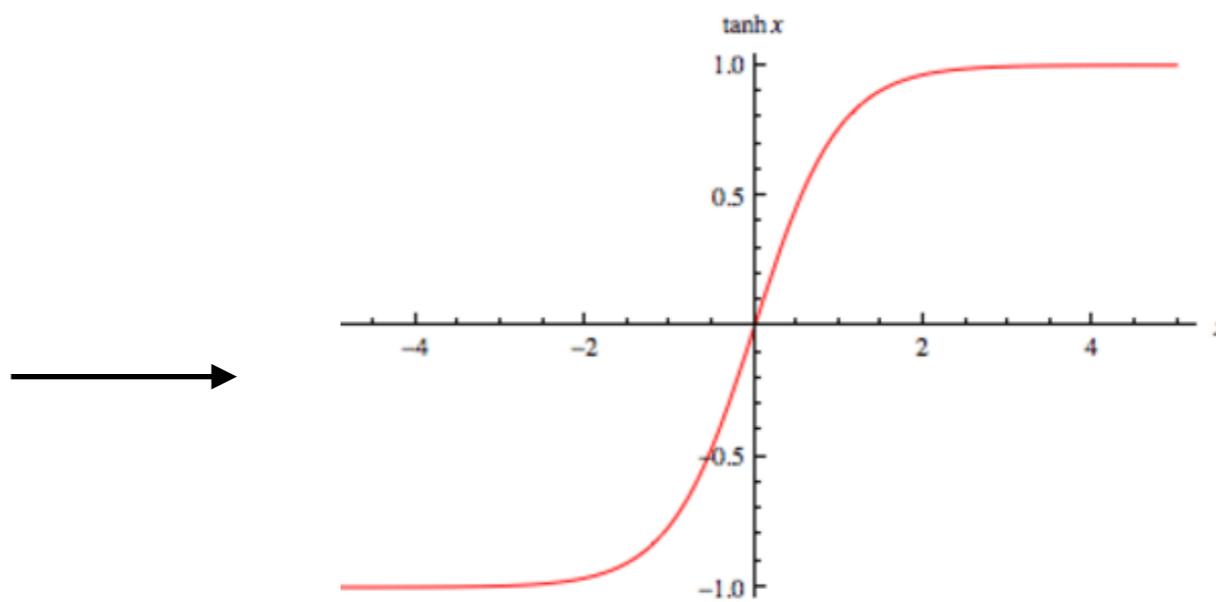
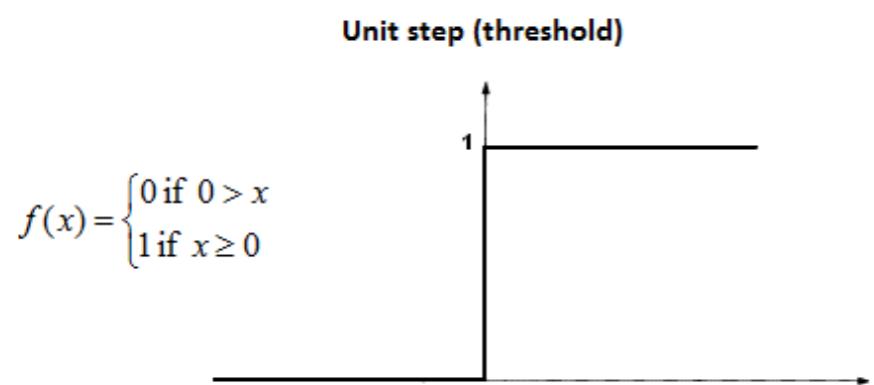
Neurona biológica (simplificación)



Neurona artificial

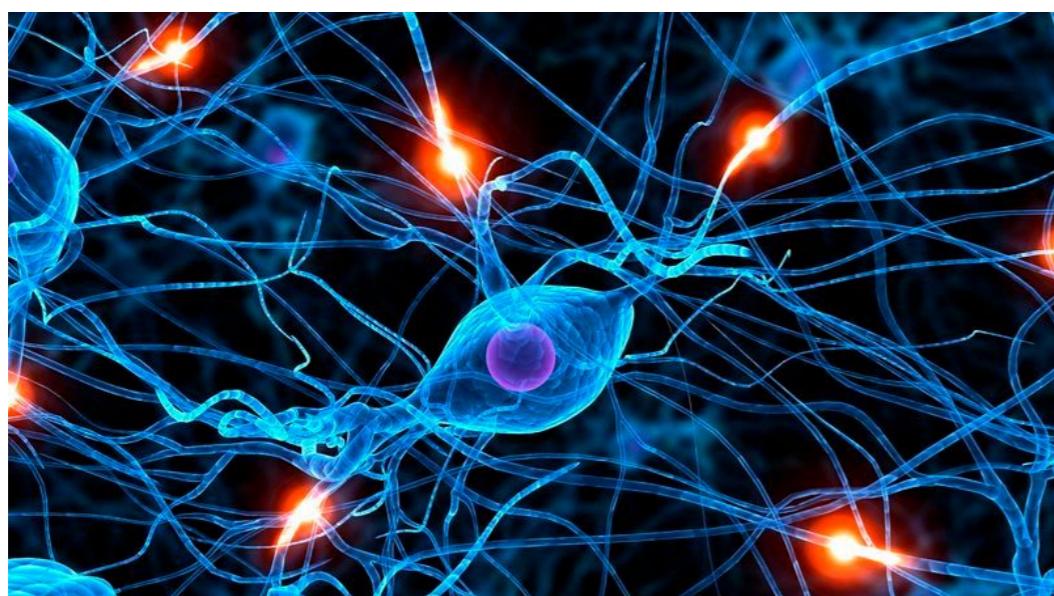


Se generalizó step function —> función de activación $f(x)$



REDES NEURONALES

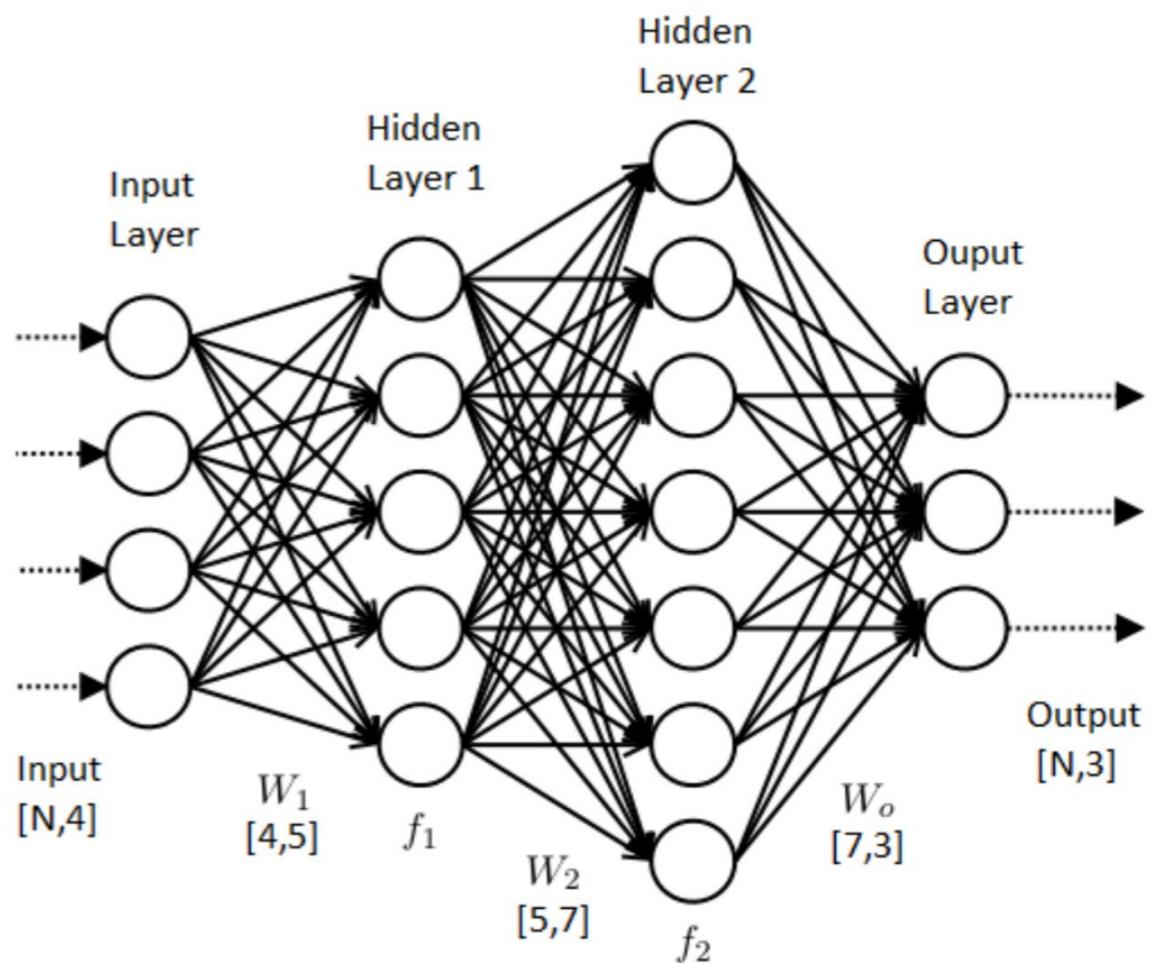
Red neuronal biológica



Una ANN son las neuronas conectadas entre sí.

Red neuronal artificial

Se organiza por capas de neuronas conectadas entre ellas.



REDES NEURONALES

TIPOS

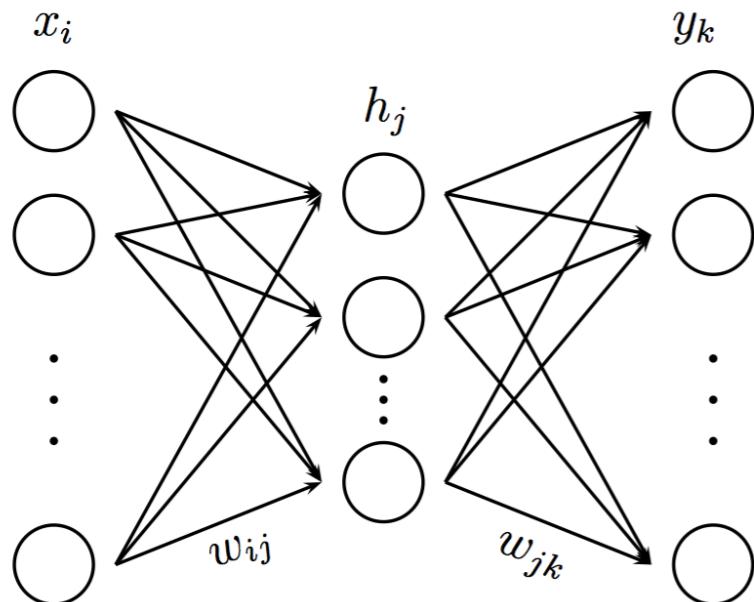
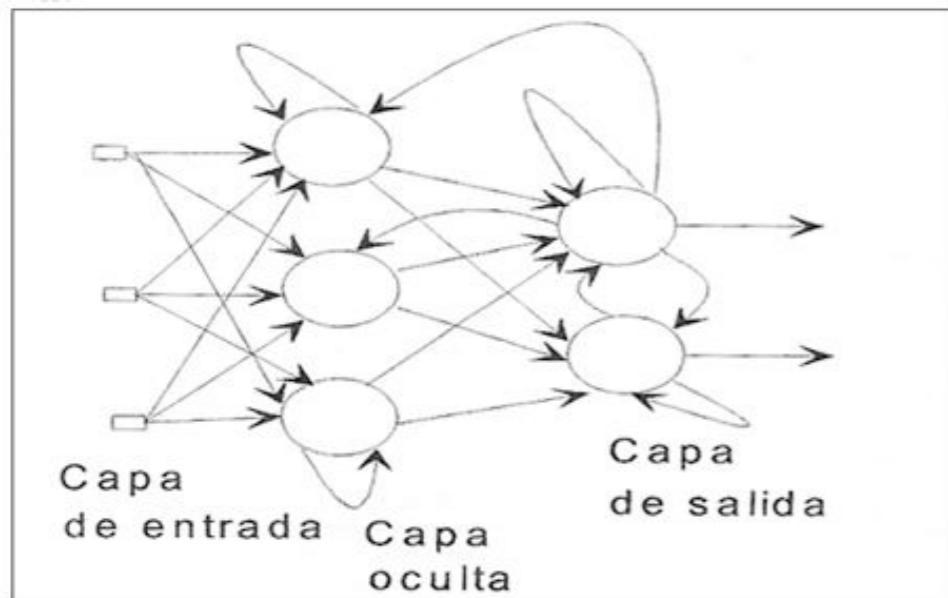
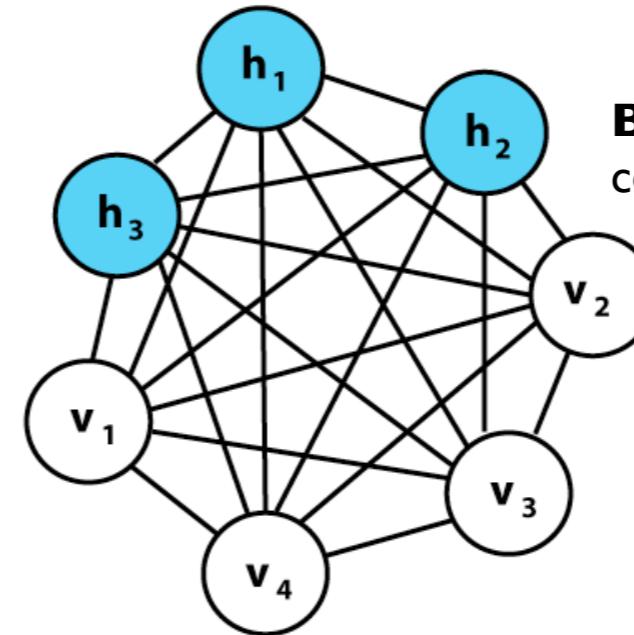


Figure 1. Schematic of a 3-layer feed-forward neural network.

Redes neuronales recurrentes. Pueden haber conexiones en varias direcciones



Feed-forward. Solo conexiones hacia delante



Boltzmann machine. Todas conectadas con todas

Redes neuronales convolucionales. deep learning

Images

REDES NEURONALES

TIPOS

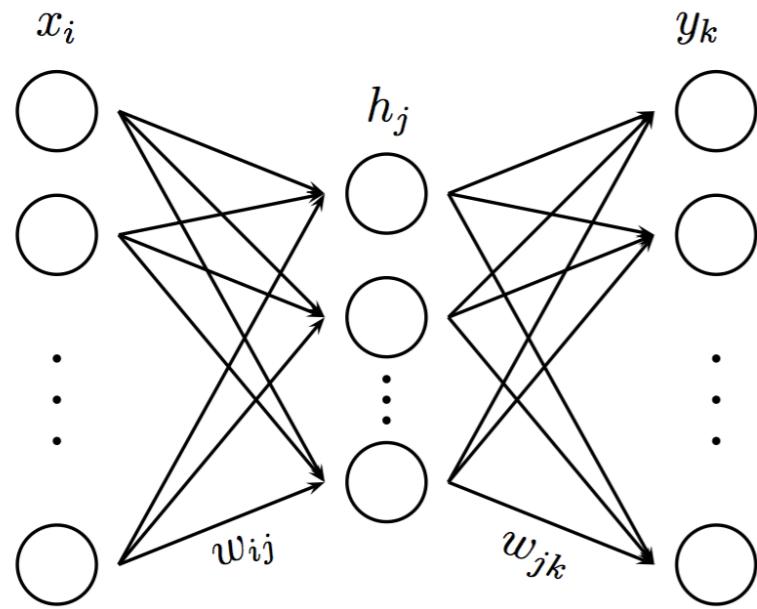
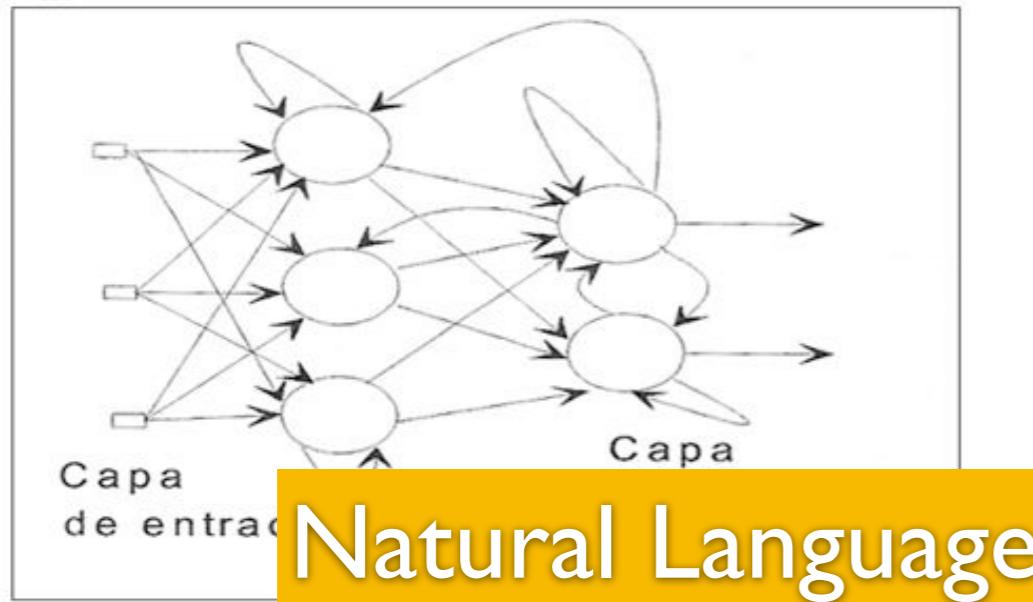
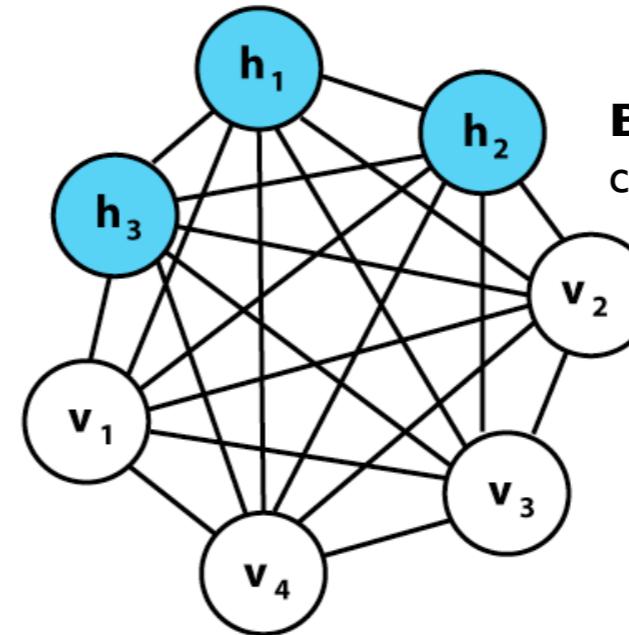


Figure 1. Schematic of a 3-layer feed-forward neural network.

Redes neuronales recurrentes. Pueden haber conexiones en varias direcciones



Feed-forward. Solo conexiones hacia delante



Boltzmann machine. Todas conectadas con todas

Redes neuronales convolucionales. deep learning

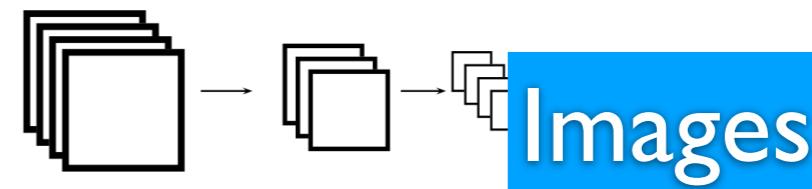


Figure 2. Schematic convolutional neural network example, with 2 hidden layers in the CNN part and 1 hidden layer in the FC part. In this example, the input depth is $k = 4$, the first layer depth $m_1 = 3$ and the second layer depth is $m_2 = 4$, the number of hidden nodes in the FC hidden layer is 4, and only one output is present.

REDES NEURONALES

TIPOS

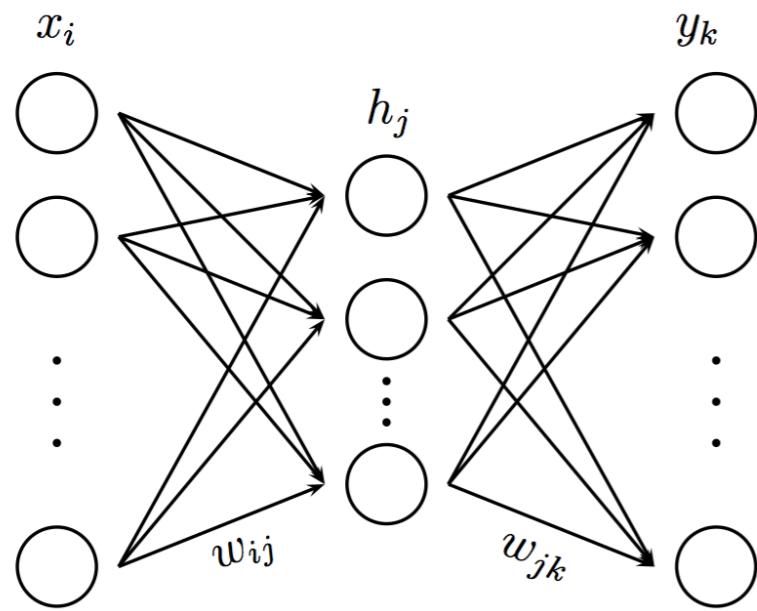
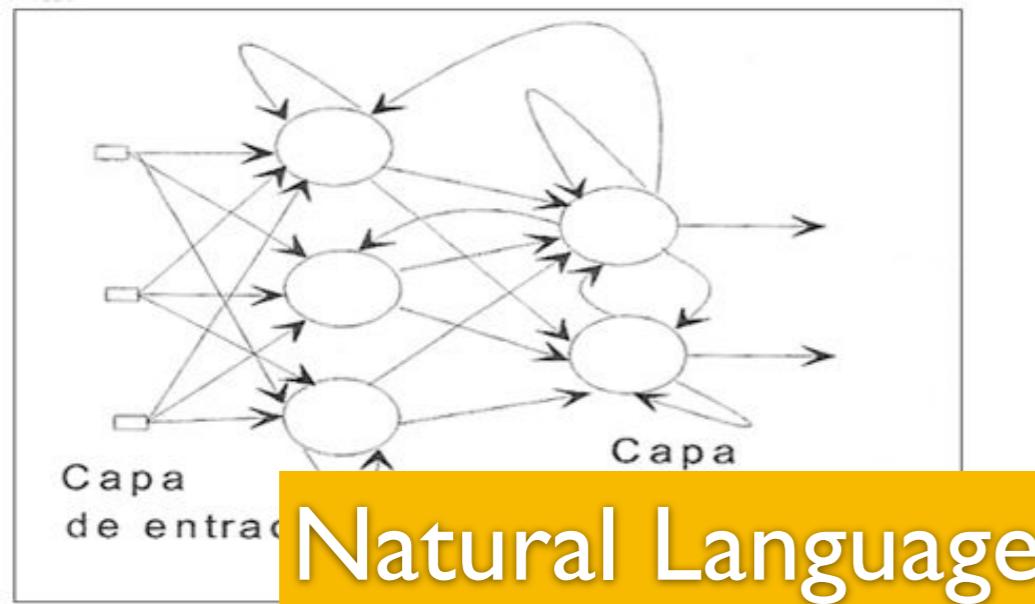


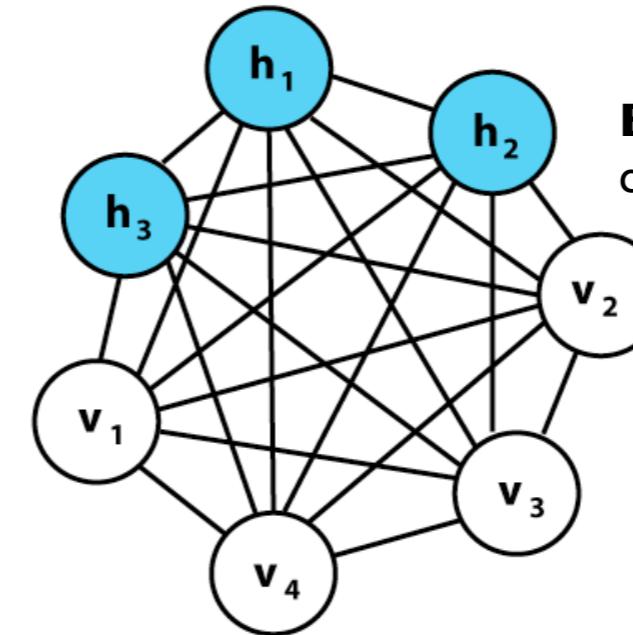
Figure 1. Schematic of a 3-layer feed-forward neural network.

Redes neuronales recurrentes. Pueden haber conexiones en varias direcciones



Natural Language

Feed-forward. Solo conexiones hacia delante



Boltzmann machine. Todas conectadas con todas

Redes neuronales convolucionales. deep learning

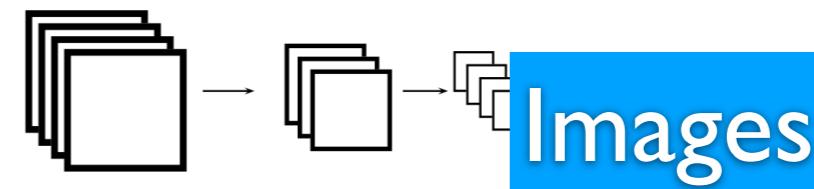
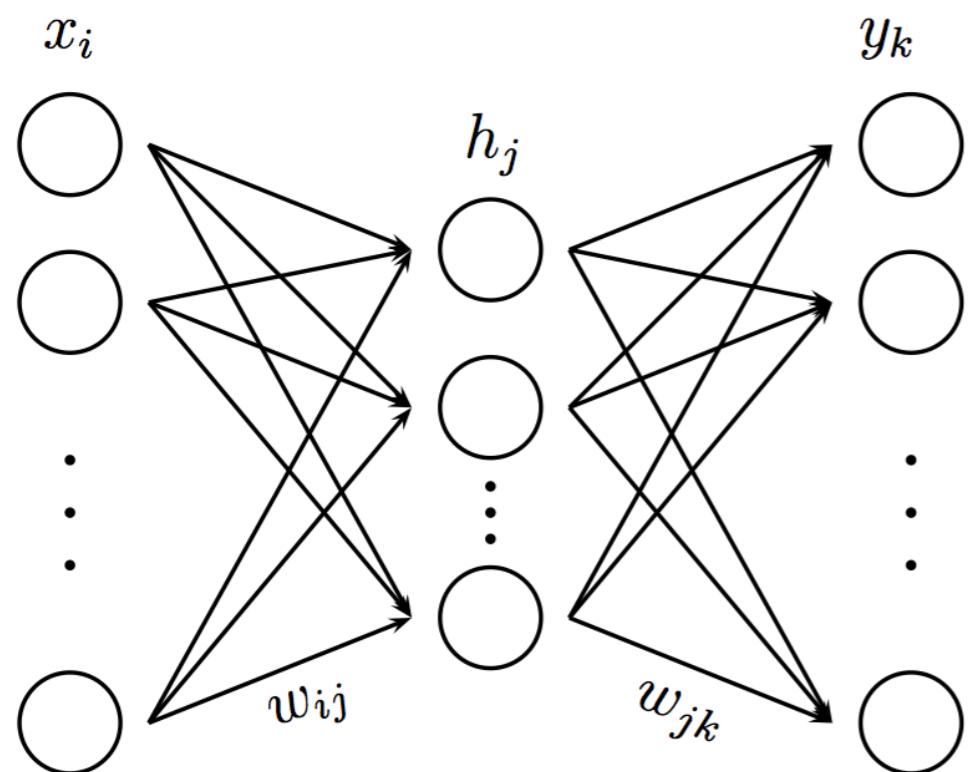


Figure 2. Schematic convolutional neural network example, with 2 hidden layers in the CNN part and 1 hidden layer in the FC part. In this example, the input depth is $k = 4$, the first layer depth $m_1 = 3$ and the second layer depth is $m_2 = 4$, the number of hidden nodes in the FC hidden layer is 4, and only one output is present.

REDES NEURONALES



Una red neuronal feed forward se organiza por capas conectadas las unas con las otras.

Figure 1. Schematic of a 3-layer feed-forward neural network.

REDES NEURONALES

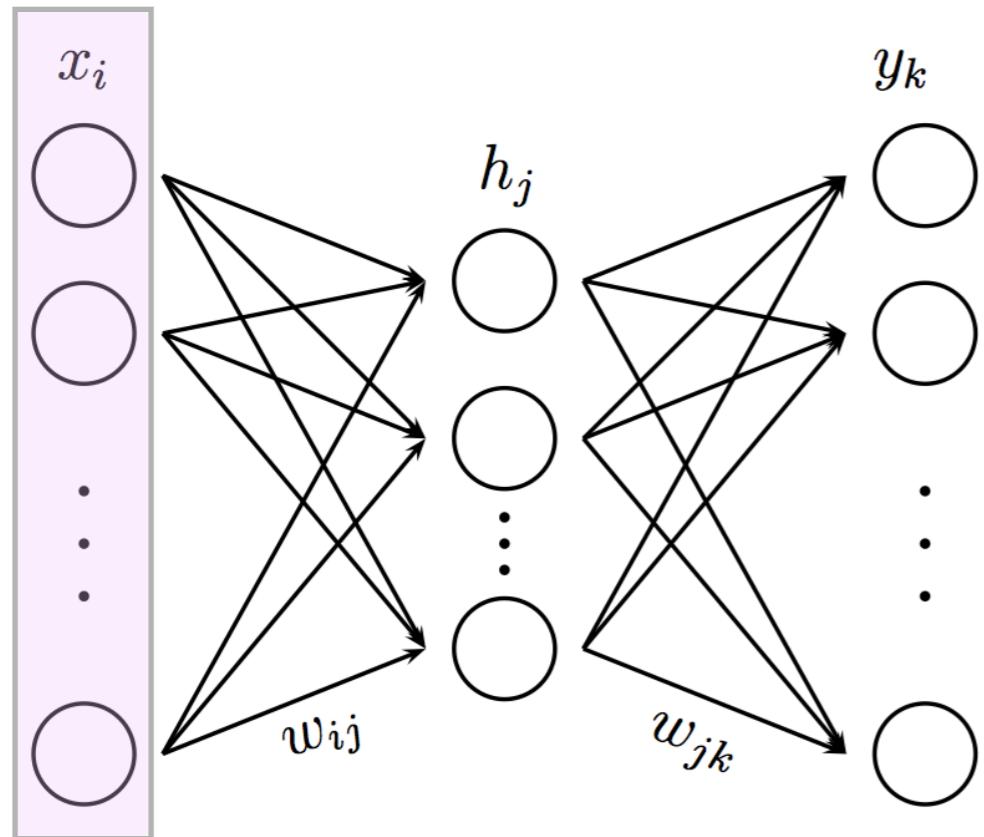


Figure 1. Schematic of a 3-layer feed-forward neural network.

Una red neuronal feed forward se organiza por capas conectadas las unas con las otras.

Están la capa inicial de **inputs** x

REDES NEURONALES

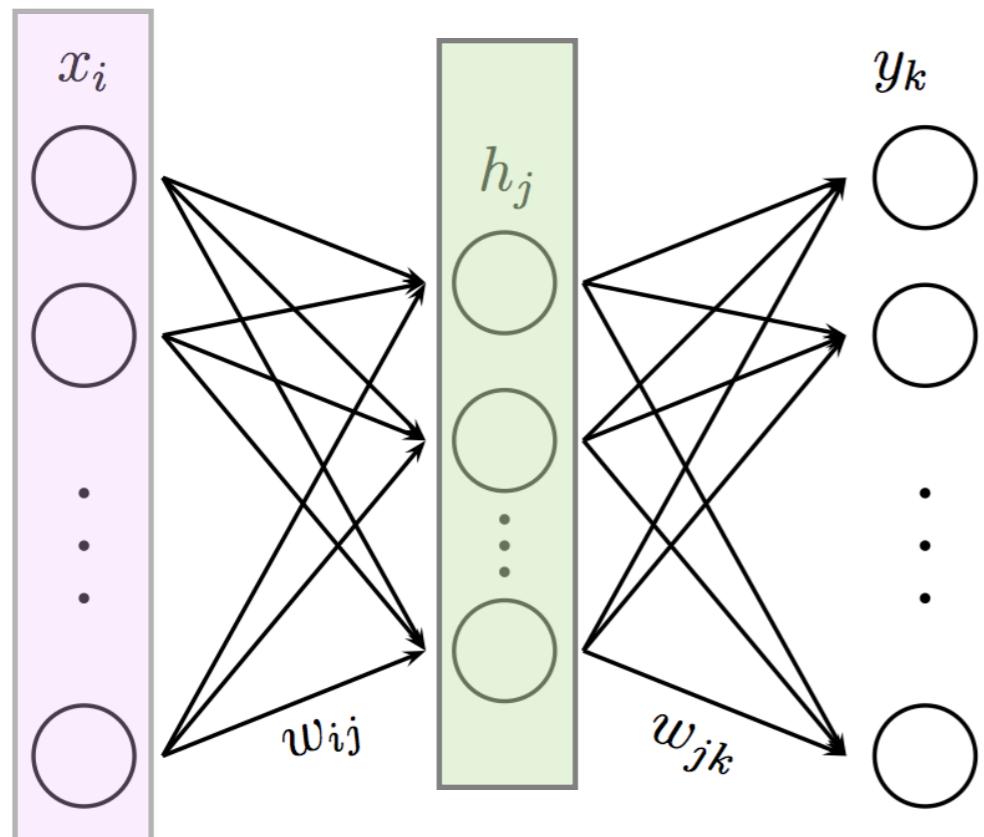


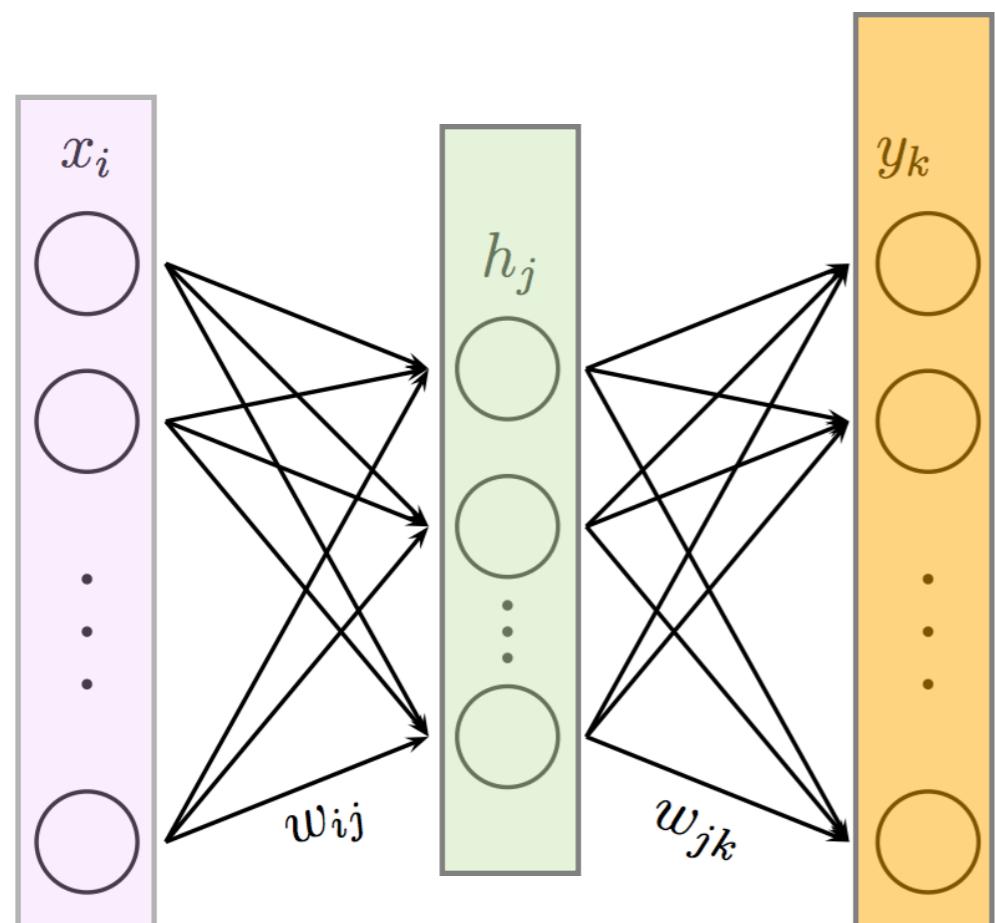
Figure 1. Schematic of a 3-layer feed-forward neural network.

Una red neuronal feed forward se organiza por capas conectadas las unas con las otras.

Están la capa inicial de **inputs** x

Las capas de neuronas (h) que se suelen llamar **capas ocultas**,

REDES NEURONALES



Una red neuronal feed forward se organiza por capas conectadas las unas con las otras.

Están la capa inicial de **inputs** x

Las capas de neuronas (h) que se suelen llamar **capas ocultas**,

La capa de **outputs** (y) con el resultado.

Figure 1. Schematic of a 3-layer feed-forward neural network.

REDES NEURONALES

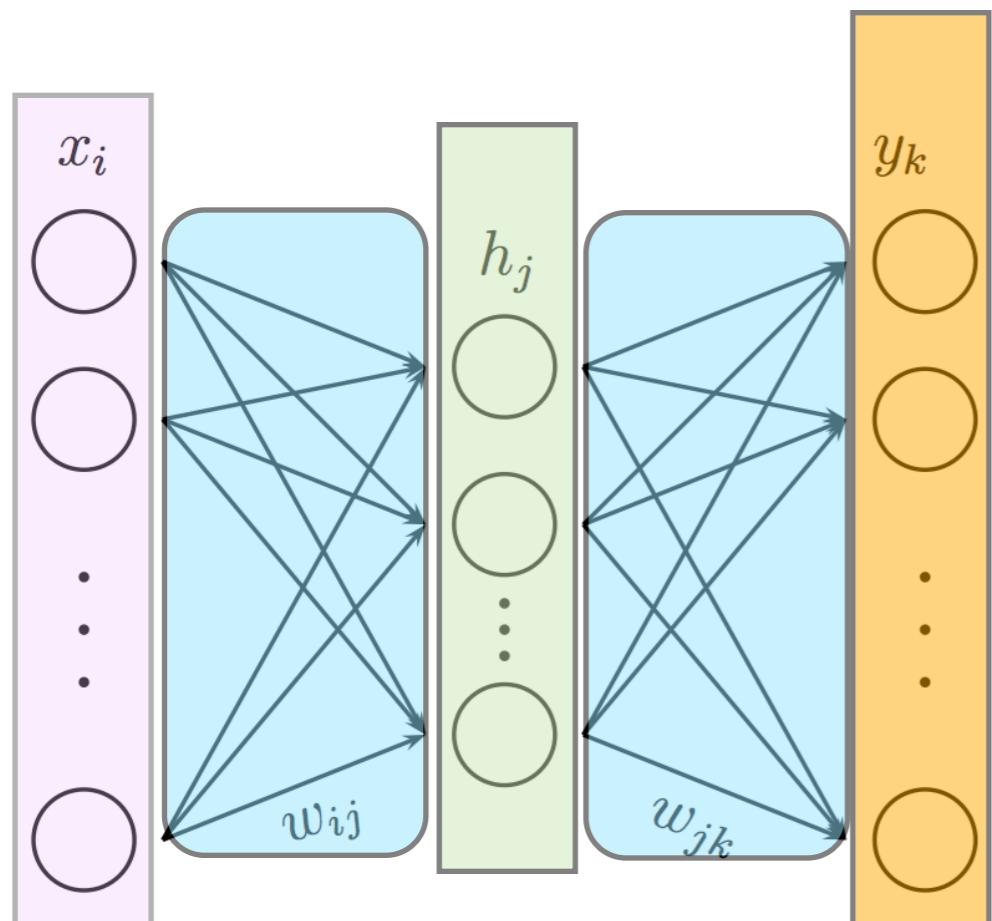


Figure 1. Schematic of a 3-layer feed-forward neural network.

Una red neuronal feed forward se organiza por capas conectadas las unas con las otras.

Están la capa inicial de **inputs** x

Las capas de neuronas (h) que se suelen llamar **capas ocultas**,

La capa de **outputs** (y) con el resultado.

Las conexiones que están representadas por pesos W

REDES NEURONALES

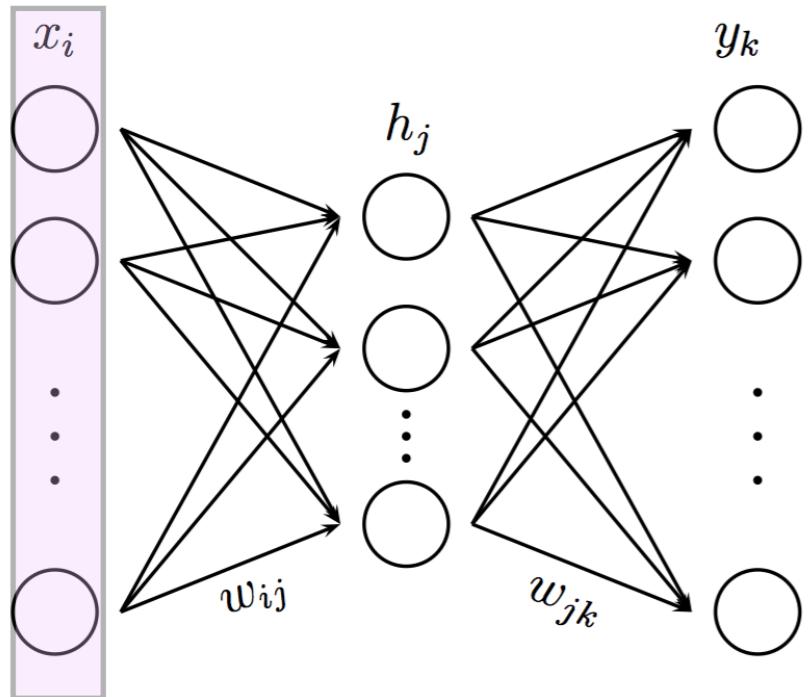


Figure 1. Schematic of a 3-layer feed-forward neural network.

En la capa de los inputs no pasa nada. Están ahí los valores de nuestras variables.

REDES NEURONALES

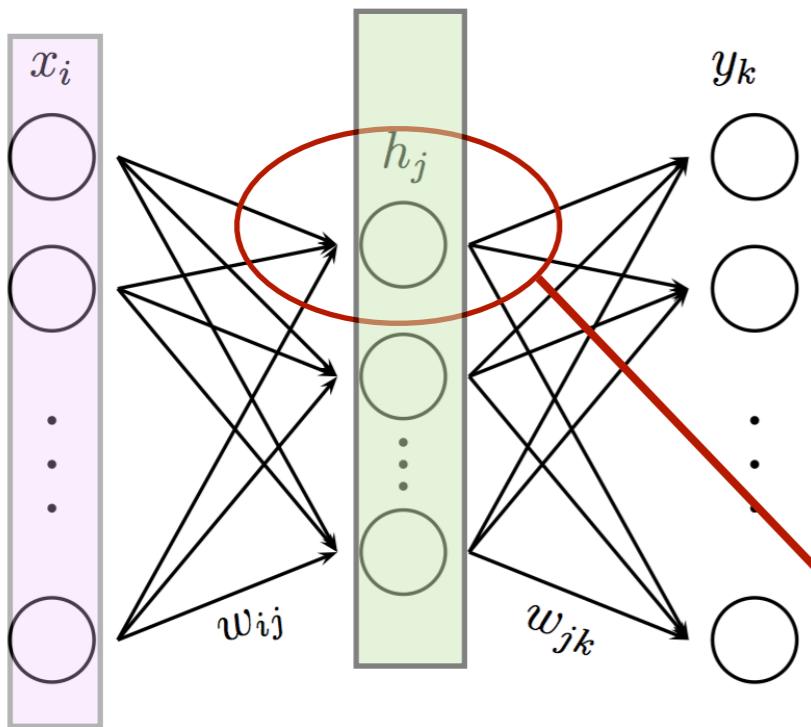
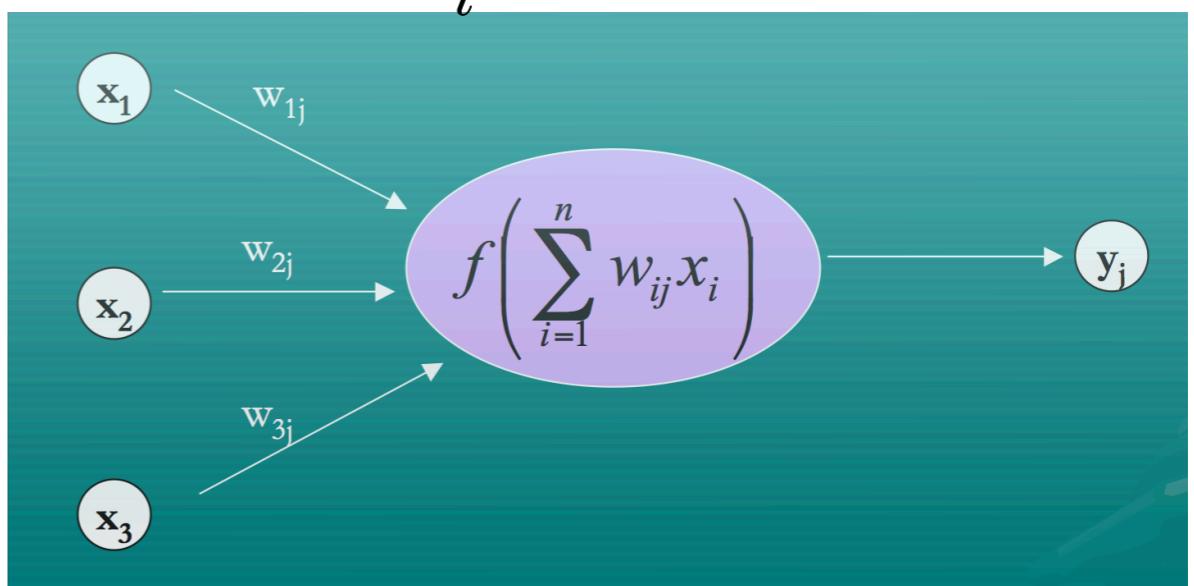


Figure 1. Schematic of a 3-layer feed-forward neural network.

En la capa de los inputs no pasa nada. Están ahí los valores de nuestras variables.

En la siguiente capa, en cada una de las neuronas se hace una operación matemática:

$$h_j = f\left(\sum_i w_{ij}x_i + \theta_j\right)$$



REDES NEURONALES

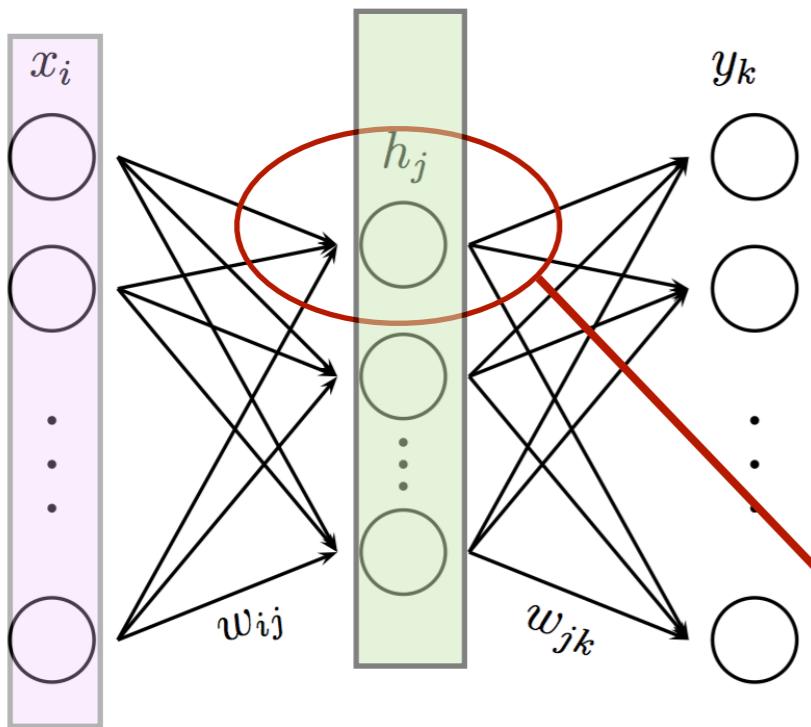


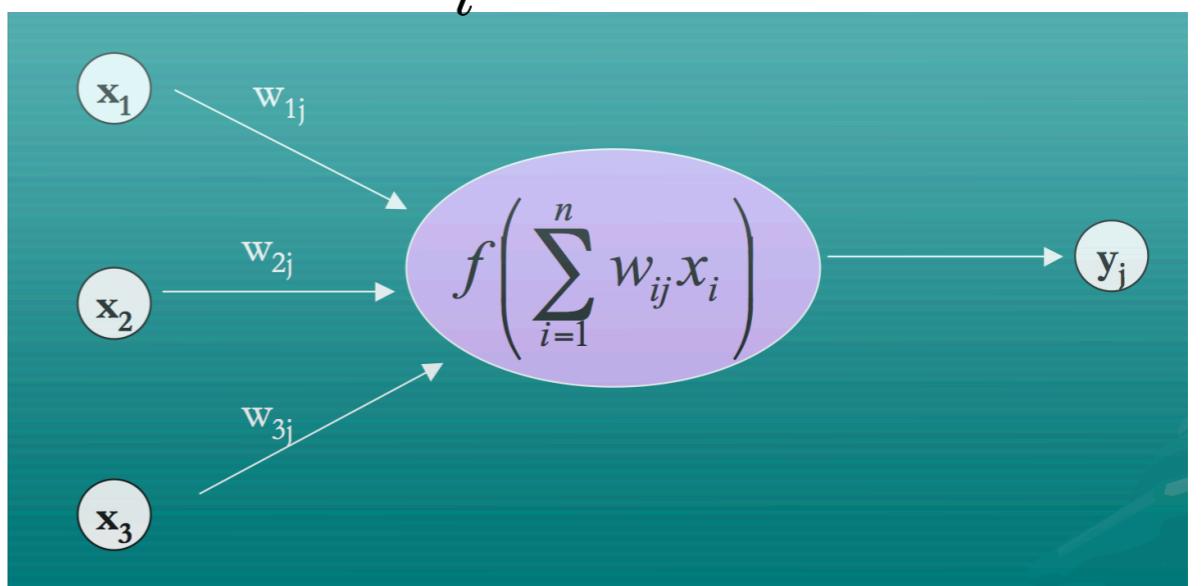
Figure 1. Schematic of a 3-layer feed-forward neural network.

Para cada neurona de la capa oculta hacemos este cálculo y los h_i serían los valores de entrada para la siguiente capa.

En la capa de los inputs no pasa nada. Están ahí los valores de nuestras variables.

En la siguiente capa, en cada una de las neuronas se hace una operación matemática:

$$h_j = f\left(\sum_i w_{ij}x_i + \theta_j\right)$$



REDES NEURONALES

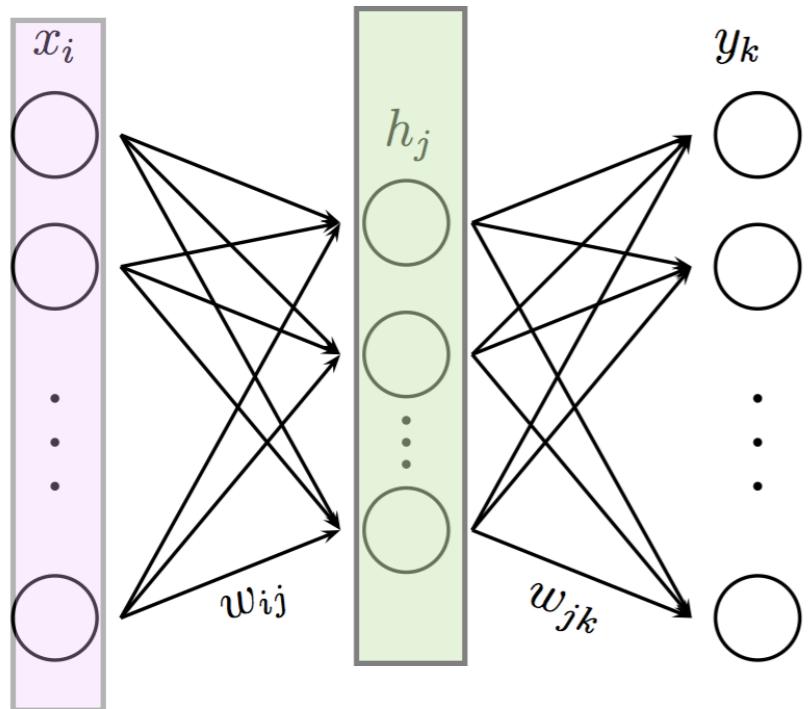


Figure 1. Schematic of a 3-layer feed-forward neural network.

Si hubiera una siguiente capa oculta vuelve a pasar lo mismo.

$$h_{j+1} = g(\sum_j w_{jj+1} h_j + \theta_{j+1})$$

La función que aplicamos no tiene porque ser la misma

REDES NEURONALES

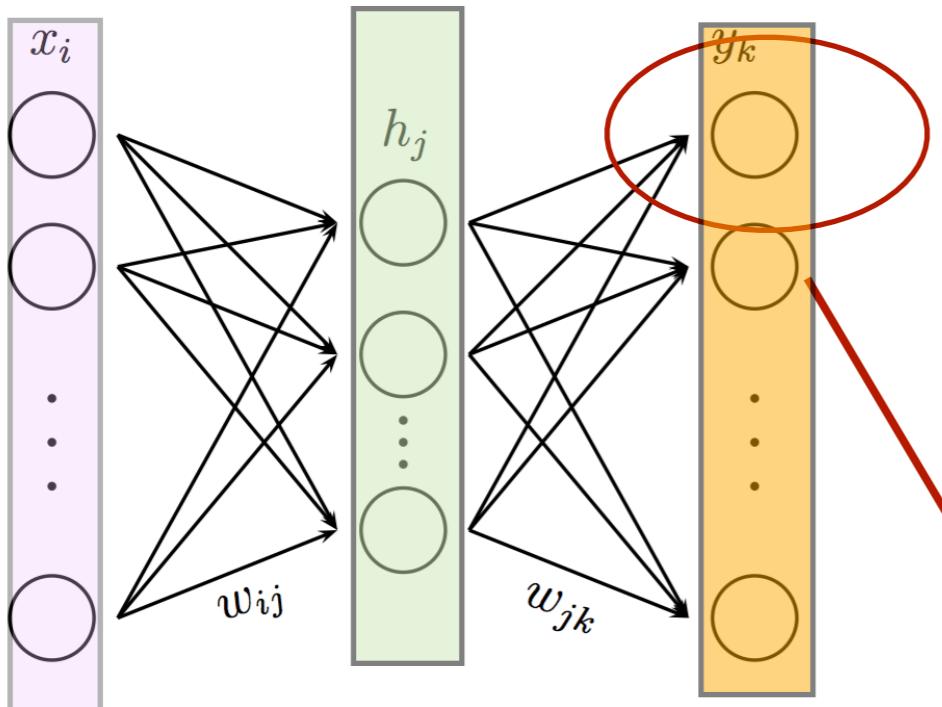


Figure 1. Schematic of a 3-layer feed-forward neural network.

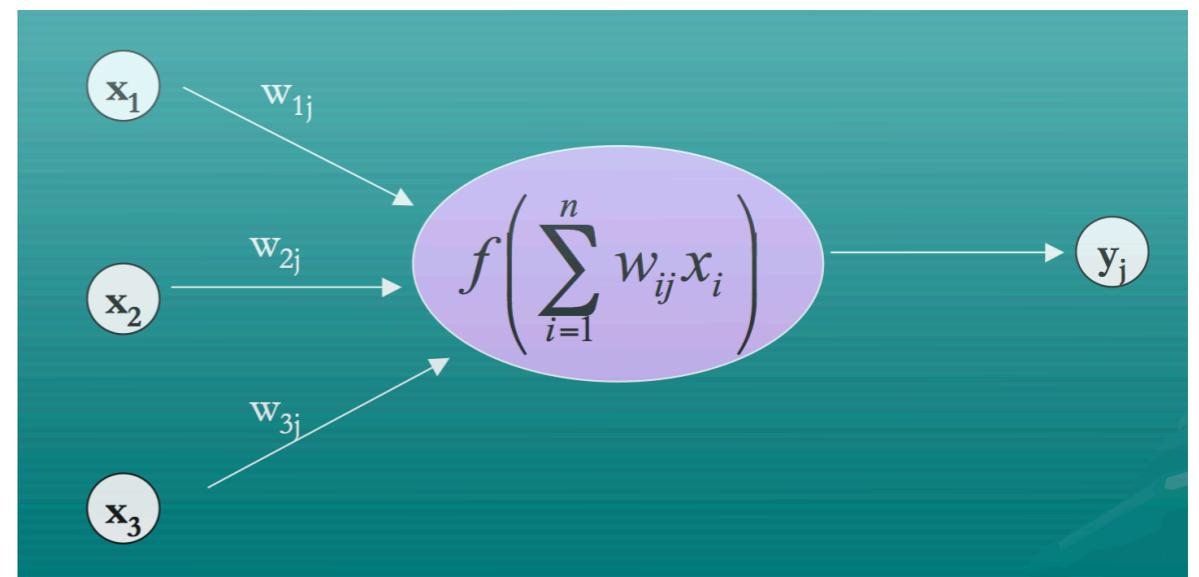
Y así hasta la última capa

$$y_k = f(\sum_k w_{kj+N} h_{j+N} + \theta_k)$$

Si hubiera una siguiente capa oculta vuelve a pasar lo mismo.

$$h_{j+1} = g(\sum_j w_{jj+1} h_j + \theta_{j+1})$$

La función que aplicamos no tiene porque ser la misma



REDES NEURONALES

Podemos poner los outputs en función de los inputs y de los pesos

Forma general:

$$y_k = g \left(w_{vk} + \sum_i w_{ij}x_i + \theta_j + w_v \right)$$

Necesitamos tener unos datos x,y y solo nos faltará conocer w, θ

REDES NEURONALES

Podemos poner los outputs en función de los inputs y de los pesos

Forma general:

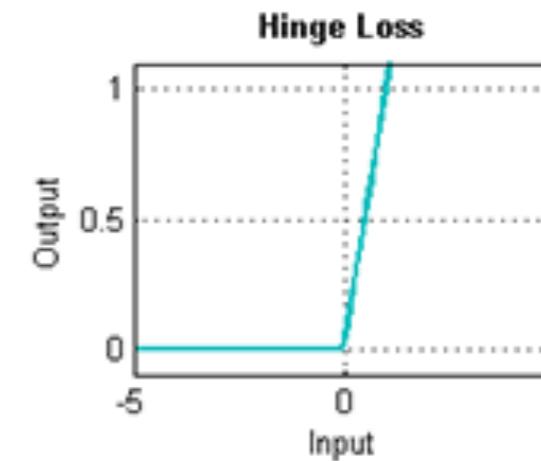
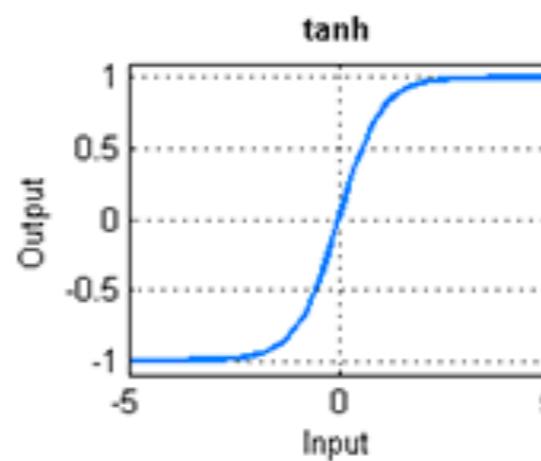
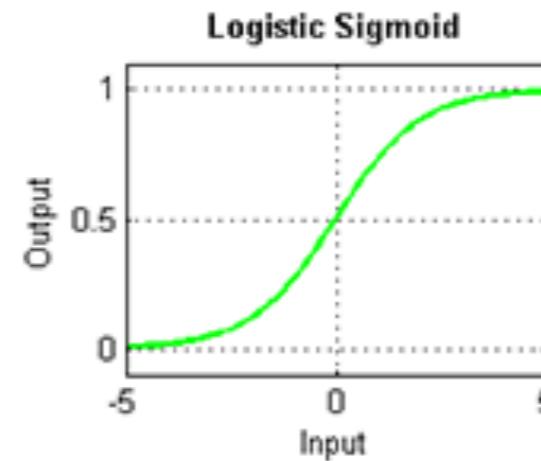
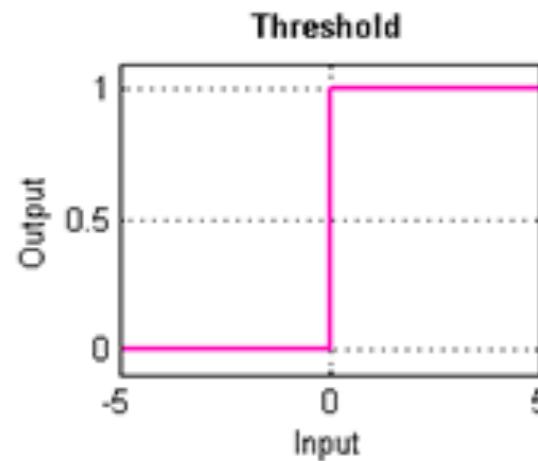
$$y_k = g \left(w_{vk} + \dots + f \left(\sum_i w_{ij} x_i + \theta_j \right) + w_v \right)$$

En el fondo estamos construyendo una familia de funciones con mucha flexibilidad.

Necesitamos tener unos datos x,y y solo nos faltará conocer w, θ

REDES NEURONALES

FUNCIONES DE ACTIVACIÓN



Función de Heaviside o función escalón.

$$H[n] = \begin{cases} 0, & n < 0, \\ 1, & n \geq 0, \end{cases}$$

Originalmente se creía que las neuronas biológicas se **activaban o no**. Por eso esta fue la función escogida en los primeros modelos.

Tangente hiperbólica o logistic sigmoid. Funciones derivables hacen que el problema sea más fácil de tratar matemáticamente. Siguen siendo funciones de activación, pero continuas, la más usada es tanh

$$f(x) = \frac{1}{1+e^{-x}} \quad f(x) = \frac{e^{2x}-1}{e^{2x}+1}$$

Recifier (ReLU). Se usa en problemas de clasificación, a veces se prefieren versiones más suaves.

Función lineal! Se suele usar en la capa final, así los outputs pueden tener cualquier valor.

$$f(x) = x$$

REDES NEURONALES

- Ejemplo

REDES NEURONALES

Ejemplo: predecir precio vivienda

Queremos predecir el precio al que se podría vender una casa dadas sus características intrínsecas y ambientales.

Training set: Datos de precio de venta de una casa de los últimos años dadas unas características.

Antigüedad	Tamaño	Contaminación	Conflictividad	Precio
20	180	130	0.1	120000
10	130	20	0.2	200000
25	200	75	0.5	130000
2	100	0	0.2	190000
...
5	200	60	0.3	?



REDES NEURONALES

Ejemplo: predecir precio vivienda

Queremos predecir el precio al que se podría vender una casa dadas sus características intrínsecas y ambientales.

Training set: Datos de precio de venta de una casa de los últimos años dadas unas características.

Antigüedad	Tamaño	Contaminación	Conflictividad	Precio
20	180	130	0.1	120000
10	130	20	0.2	200000
25	200	75	0.5	130000
2	100	0	0.2	190000
...
5	200	60	0.3	?



EMPEZAMOS A DEFINIR LA RED QUE VAMOS A USAR

- 1) Es una red de regresión porque queremos saber un valor.
- 2) Primera capa 4 neuronas última l.
- 3) Última capa función de activación lineal

REDES NEURONALES

Ejemplo: predecir precio vivienda

Acabamos de definir numero de neuronas y capas
Y tenemos una función

$$y_k = \sum_j w_{jk} \tanh(\sum_i w_{ij} x_i + \theta_j) + \theta_k$$



Escogemos la función que queremos minimizar
loss function.

$$E = \sum_n \frac{(t^n - y^n)^2}{N}$$

En el ajuste encontramos los parámetros del
modelo que son los pesos y los bias w_{ij}, θ_j

REDES NEURONALES

Ejemplo: predecir precio vivienda

Acabamos de definir numero de neuronas y capas
Y tenemos una función

$$y_k = \sum_j w_{jk} \tanh(\sum_i w_{ij} x_i + \theta_j) + \theta_k$$



Escogemos la función que queremos minimizar
loss function.

$$E = \sum_n \frac{(t^n - y^n)^2}{N}$$

En el ajuste encontramos los parámetros del
modelo que son los pesos y los bias w_{ij}, θ_j

REDES NEURONALES

Ejemplo: predecir precio vivienda

Acabamos de definir numero de neuronas y capas
Y tenemos una función

$$y_k = \sum_j w_{jk} \tanh(\sum_i w_{ij} x_i + \theta_j) + \theta_k$$



Escogemos la función que queremos minimizar
loss function.

$$E = \sum_n \frac{(t^n - y^n)^2}{N}$$

en este caso k=1, pero si hay más de un output, la función objetivo general quedaría (K número de outputs, N número set training) :

En el ajuste encontramos los parámetros del modelo que son los pesos y los bias w_{ij}, θ_j

REDES NEURONALES

Ejemplo: predecir precio vivienda

Una vez la red está entrenada tendremos un modelo que relaciona las características con el precio

Antigüedad	Tamaño	Tamaño jardín	Conflictividad	Precio
20	180	0	0.1	120000
10	130	100	0.2	200000
25	200	20	0.5	130000
2	100	0	0.2	190000
...
5	200	20	0.3	



$$y_k = \sum_j w_{jk} \tanh(\sum_i w_{ij} x_i + \theta_j) + \theta_k$$

REDES NEURONALES

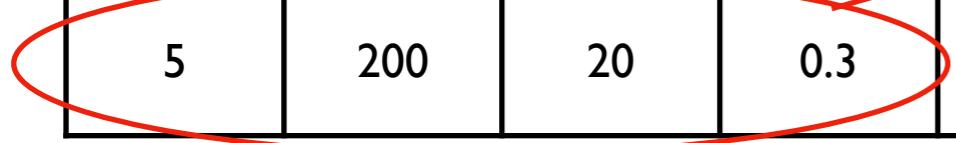
Ejemplo: predecir precio vivienda

Una vez la red está entrenada tendremos un modelo que relaciona las características con el precio

Antigüedad	Tamaño	Tamaño jardín	Conflictividad	Precio
20	180	0	0.1	120000
10	130	100	0.2	200000
25	200	20	0.5	130000
2	100	0	0.2	190000
...
5	200	20	0.3	



$$y_k = \sum_j w_{jk} \tanh(\sum_i w_{ij} x_i + \theta_j) + \theta_k$$



REDES NEURONALES

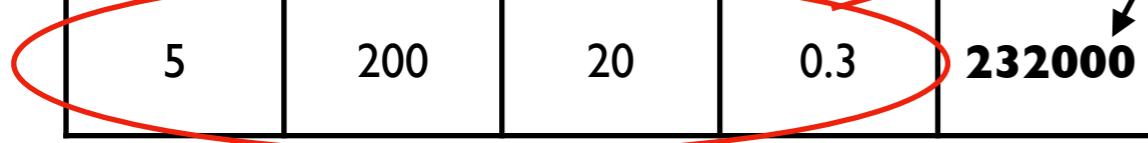
Ejemplo: predecir precio vivienda

Una vez la red está entrenada tendremos un modelo que relaciona las características con el precio

Antigüedad	Tamaño	Tamaño jardín	Conflictividad	Precio
20	180	0	0.1	120000
10	130	100	0.2	200000
25	200	20	0.5	130000
2	100	0	0.2	190000
...
5	200	20	0.3	232000



$$y_k = \sum_j w_{jk} \tanh(\sum_i w_{ij} x_i + \theta_j) + \theta_k$$



REDES NEURONALES

Entrenamiento

Una vez definimos la función podríamos usar `curve_fit` como antes.

Pero las ANN suele hacerse en problemas **multidimensionales**, con ajustes con **muchos parámetros** ($ninp \times nhid + nhid \times nout + nhid + nout$) y **muchos datos** por lo que se usa un proceso ligeramente más elaborado.

REDES NEURONALES

- Entrenamiento

REDES NEURONALES

ENTRENAMIENTO

Por temas de memoria y poder visualizar el desempeño de la red, el ajuste se suele hacer a trozos y se van actualizando los pesos:

Anemia_Egos.xls																																		
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL					
STU	PRO	NBV	NBR	NPB	MB	ENF	T	w	we	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20					
1	1	10	10	30	36	2	0	4	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26					
1	1	10	10	30	36	5	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22				
0	0	2	2	18	10	2	3	0	1	2	3	4	5	17	3	1	2	165	60	90	52	56	21	41	63	4	12	1	0	0	0			
1	0	11	20	3	3	2	0	1	4	6	7	8	9	10	11	12	13	14	15	16	17	18	19	61	1	3	165	50,4	54,4					
1	0	38	100	0	2	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
1	0	15	100	0	2	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
1	0	3	5	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0			
1	1	2	3	2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0			
1	0	3	3	4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0			
1	0	18	100	0	2	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
1	0	5	7	2	2	4	0	1	2	3	4	5	8	18	5	2	3	165	63,5	68,6	56,6	47,6	23,3	56,2	9	12	1	1	0	1	1	0	1	1
1	0	50	99	26	11	1	0	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24				
1	0	9	1	1	0	2	0	2	4	5	7	17	4	1	1	2	167	60	64	52	40	23	50	10	10	1	1	0	0	0	0	0		
1	0	25	100	0	2	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
1	0	6	1	2	1	1	0	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24				
1	1	45	25	2	2	3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
1	0	15	5	8	2	3	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22				
1	0	4	12	8	3	3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
1	0	3	4	1	2	3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
1	0	1	3	2	2	2	0	1	2	3	4	5	16	4	1	2	157	47,6	59,2	43,1	38,5	18,2	55	41	21	0	0	0	0	0	0	0		
1	0	9	3	1	0	4	0	1	2	3	4	5	7	1	1	4	153	51	63	51	40	22,4	40	10	2	1	0	0	0	0	0	0		
0	1	8	5	4	1	1	0	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24				
1	0	4	8	1	2	3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
0	1	15	100	5	1	3	0	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24				
1	1	300	700	10	3	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
1	0	5	7	3	2	4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
1	0	1	2	1	2	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
0	0	6	2	3	2	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
1	1	80	5	3	0	4	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24				
0	0	1	2	1	2	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
0	0	1	2	1	2	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
0	0	1	2	1	2	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
0	0	1	2	1	2	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
0	0	1	2	1	2	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
0	0	1	2	1	2	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				
0	0	1	2	1	2	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23				

REDES NEURONALES

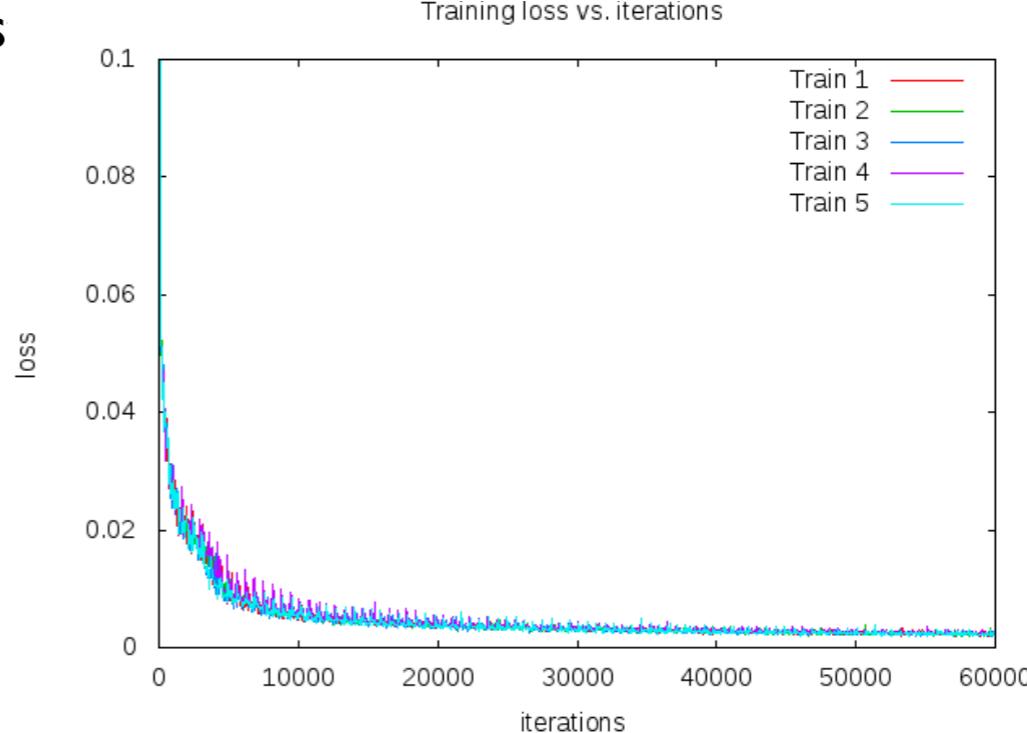
ENTRENAMIENTO

Por temas de memoria y poder visualizar el desempeño de la red, el ajuste se suele hacer a trozos y se van actualizando los pesos:

		100%		Zoom		Help																							
Undo	Redo	AutoSum	Sort A-Z	Sort Z-A	Gallery	Toolbox																							
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AJ	AK	AL	
STU	PRO	NBV	NBI	NBS	NPB	NPR	NPS	ENF	LN	WE	IN	R	A	D	O	TAI	PO	PAW	PM	PUS	AD	BONP	NC	NC	LE	L	W	C	
1	1	10	10	1	1	4	1	3	4	1	3	4	1	16	48	45	45.4	52.2	52.1	19.1	57.2	10	5.1	0	0	0	1		
1	1	10	10	1	1	4	1	3	4	1	3	4	1	16	48	45	45.4	52.2	52.1	19.1	57.2	10	5.1	0	0	0	1		
1	1	30	20	50	5	1	0	3	9	8	1	19	6	4	5	170	88	92	61.5	63	60	68	9	3	1	0	1	0	
1	0	4	5	3	5	5	0	3	9	8	1	19	9	1	7	151	154	165	104	45	53	68	9	3	1	0	1	1	
1	0	2	18	10	2	3	0	1	6	6	5	17	3	2	165	65	60	52	21.4	63	4	12	1	0	0	0	1		
1	0	35	100	10	2	2	0	1	7	3	7	4	4	3	167	124	108	145	165	50	60	7	16	1	0	0	0	1	
1	0	15	100	10	4	3	1	4	7	10	5	17	8	2	170	100	109	63	50	36	125	1	4	1	0	1	1		
1	0	2	5	1	3	0	1	2	11	3	15	5	1	3	152	99	65	59	45	52	5	2	1	0	0	0	1		
1	1	2	3	2	3	0	1	4	8	4	4	14	6	4	162	48.5	50	46	30	48	9	1	1	0	0	0	1		
1	1	3	30	1	1	9	0	2	1	1	1	19	2	1	4	21	44	44	17	18	44	44	4	1	0	0	1		
1	0	3	4	0	1	2	4	7	1	1	16	3	1	1	162	37	35	44	40	45	65	10	3	1	1	0	1		
1	0	15	2	10	3	0	2	3	4	4	2	157	50	49	48	39	21	52	27	0	1	0	0	0	0	1			
1	0	5	7	2	2	4	0	4	6	8	16	5	2	3	165	63.5	66.6	56.6	47.5	23.3	56.2	9	12	1	0	1	0	1	
1	0	50	99	20	1	0	1	3	6	2	13	16	4	1	160	40	57	36	34	15.9	47.3	30	14	1	0	0	0	1	
1	0	1	1	0	1	0	1	1	3	6	2	13	16	4	1	160	40	57	36	34	15.9	47.3	30	14	1	0	0	0	1
1	0	1	25	7	3	3	0	1	6	4	7	21	4	2	170	165	67	67	54	24.4	65	9	5	1	0	0	1		
1	0	6	10	2	1	1	0	2	4	5	6	18	4	2	160	52	61	51	46	20.4	54	5	4	1	0	0	1		
1	1	45	20	2	2	3	0	8	2	3	5	5	1	3	158	51	57	46.5	40	57	12	4	1	1	1	0	1		
1	0	10	5	8	2	3	0	1	5	6	5	140	25	20	23	24	42	43	6	5	1	0	0	0	1				
1	0	1	1	1	1	1	0	1	9	9	1	19	1	1	1	160	40	40	40	23.8	23.8	40	1	1	0	0	1		
1	0	3	70	6	3	6	0	1	7	11	9	6	1	9	161	77	101	44	31	34.8	46	3	3	1	1	0	0	1	
1	0	1	3	2	2	2	0	2	4	5	16	4	2	1	257	47.8	52.2	36.5	19.2	50	4	2	1	0	0	0	1		
1	0	9	3	1	4	0	1	3	7	7	7	4	1	153	51	63	51	45	22.4	40	10	2	1	0	0	1			
1	0	4	4	1	4	1	0	1	4	5	16	4	2	1	160	40	40	40	23.8	23.8	40	1	1	0	0	1			
1	0	1	4	8	3	1	0	3	5	4	9	15	6	1	3	172	65	75	60	50	70	6	5	1	0	0	1		
1	0	15	100	5	3	0	1	2	1	8	14	1	1	7	177	73	73	82.7	43	45	24.4	68	16	3	1	0	0	1	
1	1	300	700	10	3	2	0	1	5	6	9	6	6	3	158	49	50	44	40	19.4	64	31	21	0	0	0	0	1	
1	0	1	7	5	3	2	0	1	4	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1	
1	0	0	8	2	3	2	0	4	2	5	18	8	3	9	150	83	83	63	40	25.1	50	6	2	1	0	0	1		
1	1	8	20	5	5	3	0	1	4	11	16	4	2	1	255	65	81	60	58	75	4	4	1	1	1	1	1		
1	0	yes	yes	yes	1	0	1	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0	0	1		
1	0	0	yes	yes	yes	1	0	1	2	6	18	4	2	1	252	48.5	54	46	21.3	52	5	5	1	0	0				

Iteración. Cada pasamos a la siguiente submuestra se llama iteración.

Época. Cuando hemos hecho un ajuste con cada uno de los “trozos”. Es Training loss vs. iterations



REDES NEURONALES

ENTRENAMIENTO

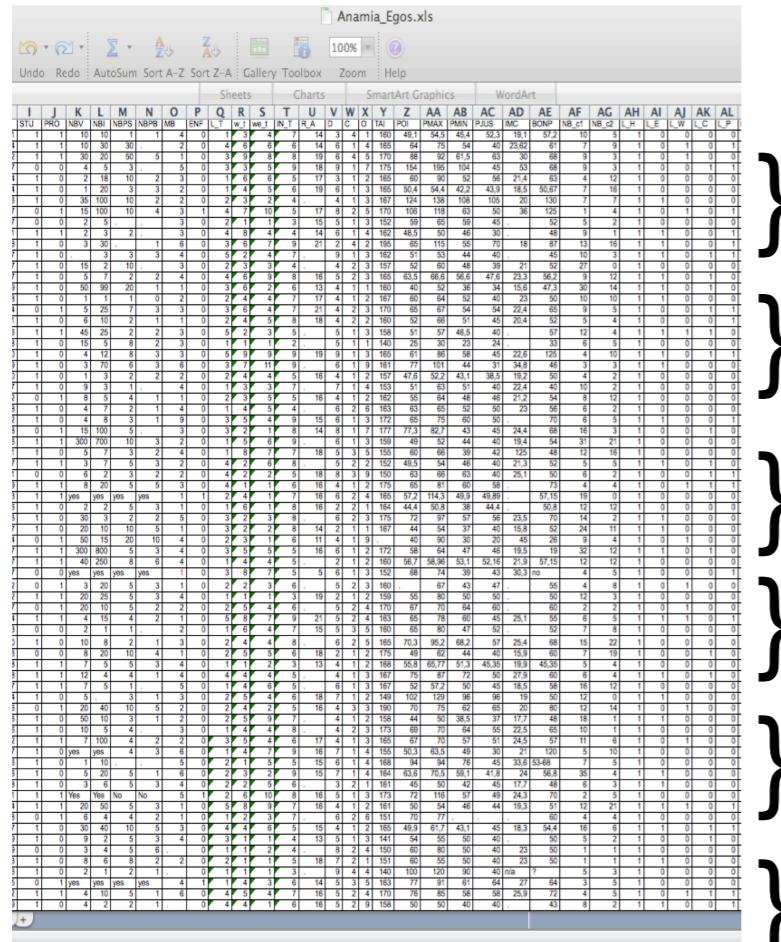
Por temas de memoria y poder visualizar el desempeño de la red, el ajuste se suele hacer a trozos y se van actualizando los pesos:

I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL							
T	STU	PRO	NBV	NBV	NBV	MB	NBV	MB	ENF	INT	R.A.	D	C	O	T.A.	PO	PMAX	PBM	PMS	MC	BONP	NB	c1	NB	c2	L.H	L.E.	L.W.	L.C.	L.P.						
T	1	1	10	10	10	1	1	4	0	3	4	7	14	3	1	160	49.1	54.5	45.4	52.3	16.1	37.2	1	5	1	0	0	0	0	0	0					
T	1	1	10	30	30	2	0	4	6	6	14	6	1	4	160	64	54	40	23.82	61	7	9	1	0	0	0	0	0	0	0						
T	0	0	30	20	50	5	1	0	3	7	8	10	6	4	5	170	88	92	61.5	63	30	68	9	3	1	0	0	0	0	0	0	0				
T	1	0	4	2	18	10	2	3	0	1	8	17	3	1	2	160	60	90	52	56	21.41	63	4	12	1	0	0	0	0	0	0	0				
T	1	0	11	20	3	3	2	0	1	4	6	19	6	1	3	160	90.4	54.4	42.2	43.9	18.5	50.67	7	16	1	0	0	0	0	0	0	0				
T	1	0	30	100	0	2	2	0	2	3	4	4	1	3	187	124	138	108	105	20	130	7	7	1	0	0	0	0	0	0	0					
T	0	0	15	100	10	10	4	0	3	7	17	20	2	3	15	5	59	59	45	36	17.5	52	9	1	0	0	0	0	0	0	0	0				
T	1	1	2	3	2	3	1	0	4	8	4	4	14	6	1	4	162	48.5	50	48	30	48	9	1	1	0	0	0	0	0	0	0	0			
T	1	0	3	3	3	3	4	0	2	7	1	1	4	1	3	182	51	53	40	45	10	3	1	0	0	0	0	0	0	0	0	0				
T	1	0	15	30	10	10	4	0	2	7	1	1	4	1	3	182	51	53	40	45	10	3	1	0	0	0	0	0	0	0	0	0				
T	1	0	5	7	2	2	4	0	4	8	16	5	2	3	165	63.5	68.6	56.6	47.6	23.3	58.2	9	12	1	0	0	0	0	0	0	0	0				
T	1	0	50	90	20	11	1	0	2	8	6	13	4	1	1	160	40	52	36	34	15.6	47.3	30	14	1	0	0	0	0	0	0	0	0			
T	1	0	1	1	1	1	0	2	4	7	17	4	1	1	2	167	60	64	52	40	23	50	10	10	1	0	0	0	0	0	0	0	0			
T	1	0	6	1	2	1	1	0	2	7	1	1	4	1	2	160	62	65	51	43	20.4	52	9	12	1	0	0	0	0	0	0	0	0			
T	1	1	45	25	2	2	3	0	2	7	1	1	5	1	3	158	51	57	40.5	40	57	12	4	1	1	1	0	0	0	0	0	0	0			
T	1	0	15	5	8	2	3	0	1	7	2	1	5	1	1	140	25	30	23	24	33	6	9	1	0	0	0	0	0	0	0	0	0			
T	1	0	4	12	8	3	3	0	1	7	1	1	5	1	3	160	61	86	58	45	22.6	725	4	10	1	0	0	0	0	0	0	0	0			
T	1	0	3	4	1	2	1	0	2	7	1	1	4	1	2	160	61	86	58	45	22.6	725	4	10	1	0	0	0	0	0	0	0	0			
T	1	0	1	3	2	2	2	0	2	4	5	16	4	1	2	157	47.6	52.2	43.1	38.5	18.2	55	4	2	1	0	0	0	0	0	0	0	0			
T	1	0	9	3	1	4	0	0	3	7	1	1	4	153	51	63	51	40	22.4	40	10	2	1	0	0	0	0	0	0	0	0	0				
T	0	1	8	5	4	1	1	0	2	7	1	1	4	1	2	162	55	64	48	46	21.2	54	8	12	1	0	0	0	0	0	0	0	0	0		
T	0	1	4	8	2	2	1	0	2	7	1	1	4	1	2	160	63	65	52	50	23	56	8	12	1	0	0	0	0	0	0	0	0	0		
T	0	1	15	100	5	1	3	0	2	7	1	1	4	14	8	1	7	177	77.3	82.7	43	45	24.4	68	16	31	1	0	0	0	0	0	0	0	0	0
T	1	1	300	700	10	10	3	2	0	1	5	9	6	1	3	159	49	52	44	40	18.4	54	31	21	1	0	0	0	0	0	0	0	0	0		
T	1	0	5	7	3	2	4	0	2	8	7	18	5	3	5	150	60	66	39	42	12.5	48	12	18	1	0	0	0	0	0	0	0	0	0		
T	0	0	6	2	3	2	2	0	2	7	1	5	18	5	2	152	46.4	54	49	42	21.2	52	12	18	1	0	0	0	0	0	0	0	0	0		
T	1	1	80	20	5	5	3	0	2	7	1	18	5	2	152	63	63	51	40	25.1	50	12	31	1	0	0	0	0	0	0	0	0	0			
T	1	1	yes	yes	yes	yes	yes	yes	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
T	0	0	1	20	10	10	5	1	0	2	7	18	14	2	1	1	167	44	54	37	40	15.8	52	24	11	1	1	0	0	0	0	0	0	0	0	0
T	0	1	50	15	20	10	4	0	2	7	1	11	4	11	4	1	160	58	64	47	46	18.5	19	32	12	1	1	0	0	0	0	0	0	0	0	0
T	0	0	1	20	20	20	5	4	0	2	7	1	1	5	1	1	152	56	68	74	70	20.3	70	35	12	1	1	0	0	0	0	0	0	0	0	0
T	0	0	1	30	5	3	1	0	2	2	6	2	3	2	6	1	1	160	51	67	43	47	55	4	81	1	0	0	0	0	0	0	0	0	0	0
T	1	1	20	25	5	3	4	0	1	7	1	3	18	2	1	1	160	55	60	50	50	50	12	31	1	0	0	0	0	0	0	0	0	0	0	
T	0	1	20	10	5	2	2	0	2	7	1	1	4	1	2	160	60	62	65	20	86	12	14	1	0	0	0	0	0	0	0	0	0	0		
T	1	0	50	10	3	1	2	0	2	7	1	1	4	1	2	158	44	50	36.5	37</																

REDES NEURONALES

ENTRENAMIENTO

Otras cosas que se tienen que escoger al hacer el entrenamiento:

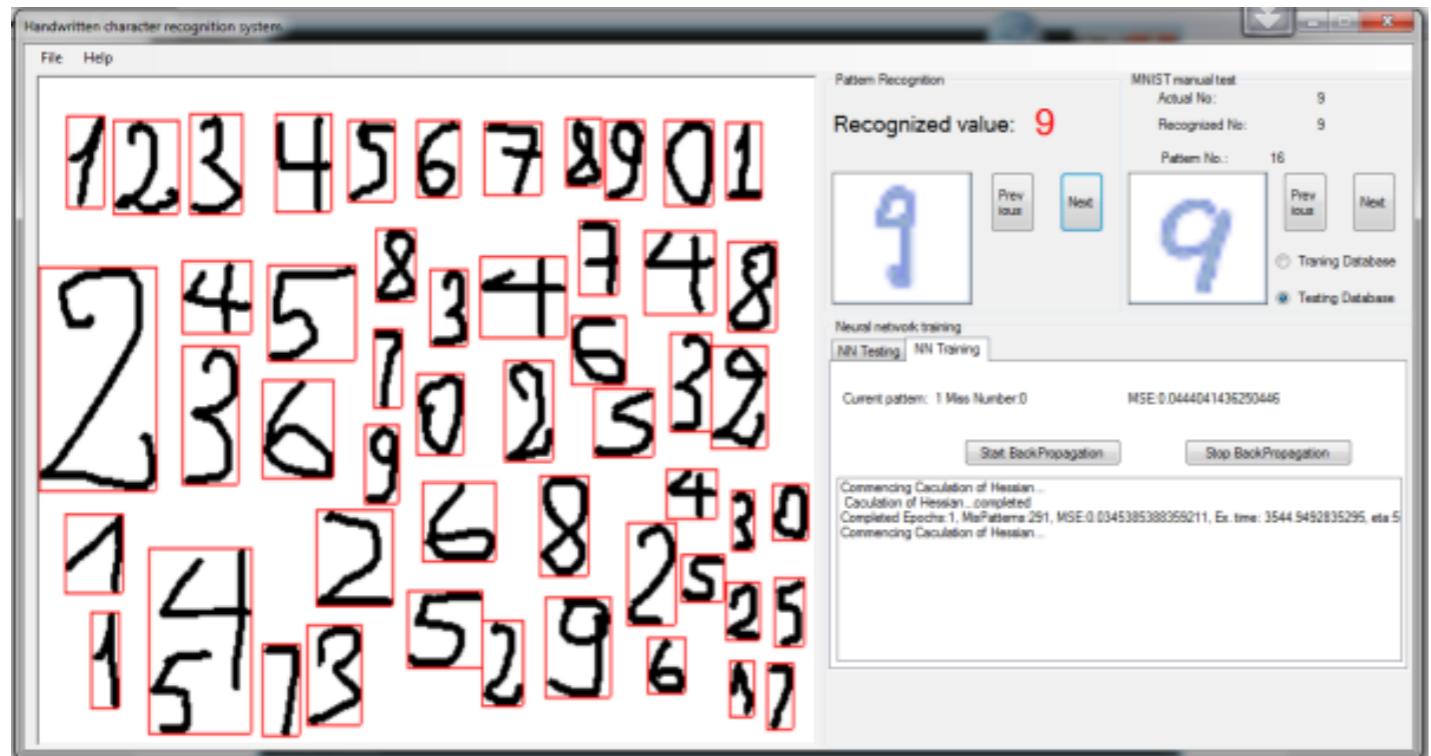


STU	PRO	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL		
1	1	10	10	1	1	4	0	1	3	4	7	14	3	4	1	1	160	49.1	54.5	45.4	52.3	16.1	37.2	5	1	0	0	0	0		
1	1	10	30	30	2	0	1	6	7	6	14	6	1	4	1	160	64	54	40	23.82	61	7	9	1	0	1	0	0	1		
1	1	30	20	50	5	1	0	1	3	7	8	10	6	4	5	170	88	92	61.5	63	30	68	9	3	1	0	1	0	0	1	
1	0	4	2	18	10	2	3	0	1	8	7	5	17	3	1	2	165	60	90	52	56	21.41	63	4	12	1	0	0	0	0	1
1	0	11	20	3	3	2	0	1	4	7	6	19	6	1	3	160	90.4	54.4	42.2	43.9	18.5	50.67	7	16	1	0	0	0	0	0	1
1	0	30	100	0	2	2	0	1	3	4	7	4	1	3	187	124	138	108	105	20	130	7	7	1	1	0	0	0	0	1	
1	0	15	100	10	0	4	0	1	7	7	10	2	1	2	1	160	90	52	36	17.5	10	12	1	0	0	0	0	0	1		
1	1	2	3	2	1	3	0	4	8	7	4	14	6	1	4	162	48.5	50	48	30	48	9	1	1	1	0	0	0	0	1	
1	0	31	30	1	1	6	0	1	6	7	9	21	2	4	2	195	65	115	55	70	18	87	13	16	1	1	0	0	0	1	
1	0	3	3	4	0	1	2	1	7	7	1	1	3	162	51	53	40	45	10	3	1	1	0	0	0	0	0	1			
1	0	18	100	100	0	2	1	0	1	7	7	1	1	2	1	160	90	52	44	21	24	27	1	0	0	0	0	0	1		
1	0	5	7	2	2	4	0	4	8	7	8	16	5	2	3	165	63.5	68.6	56.6	47.6	23.3	58.2	9	12	1	1	0	0	0	1	
1	0	50	99	26	11	1	0	2	8	7	6	13	4	1	1	160	40	52	36	34	15.6	47.3	30	14	1	1	0	0	0	1	
1	0	91	1	1	1	0	2	0	4	7	7	17	4	1	2	187	60	64	52	40	23	50	10	10	1	1	0	0	0	1	
1	0	20	20	100	0	2	1	1	7	7	1	1	2	1	1	160	90	52	44	22.4	40	10	2	1	0	0	0	0	1		
1	0	61	1	2	1	1	0	2	7	7	8	16	4	2	2	160	62	65	51	43	26.4	62	5	1	0	0	0	0	1		
1	1	45	25	0	2	3	0	3	2	7	7	5	1	3	158	91	57	40.5	40	57	12	41	1	1	1	0	0	0	1		
1	0	15	5	8	2	3	0	1	7	7	1	1	2	1	1	140	26	30	23	24	33	6	5	1	0	0	0	0	0	1	
1	0	41	12	8	3	3	0	1	7	7	9	1	3	160	61	86	58	45	22.6	72.5	4	10	1	1	0	0	0	0	1		
1	0	3	4	0	1	2	1	0	7	7	1	1	2	1	1	160	90	52	44	21	24	27	1	0	0	0	0	0	1		
1	0	1	3	2	2	0	1	2	4	5	5	16	4	1	2	157	47.6	52.2	43.1	38.5	18.2	55	4	2	1	0	0	0	0	1	
1	0	91	3	1	1	4	0	1	3	7	7	1	4	153	51	63	51	40	22.4	40	10	2	1	0	0	0	0	1			
0	1	81	5	4	1	1	0	2	7	7	5	16	4	1	2	162	55	64	48	46	21.2	54	8	12	1	0	0	0	0	1	
0	1	4	8	2	1	1	0	2	7	7	6	1	2	1	1	160	63	65	52	50	23	56	8	21	1	0	0	0	0	1	
0	1	4	8	1	1	1	0	2	7	7	5	16	4	1	2	160	65	66	53	51	23.5	57	8	21	1	0	0	0	0	1	
0	1	15	100	5	1	3	0	2	7	7	8	14	8	1	2	177	77.3	82.7	43	45	24.4	68	16	3	1	0	0	0	1		
1	1	300	700	10	3	2	0	1	5	7	9	6	1	3	159	49	52	44	40	18.4	54	31	21	1	0	0	0	0	1		
1	0	51	7	3	2	4	0	1	8	7	18	5	3	5	150	60	66	39	42	12.5	48	12	16	1	0	0	0	0	1		
1	0	6	2	3	2	0	1	2	7	7	5	18	8	9	1	2	152	46.4	54	49	21.2	52	5	1	0	0	0	0	1		
1	1	81	20	5	5	3	0	2	7	7	1	6	4	1	2	157	47.6	52.2	43.1	38.5	18.2	55	4	2	1	0	0	0	0	1	
1	0	1	yes	yes	yes	1	1	1	4	7	7	16	6	2	3	165	97.2	114.3	48.9	49.89	57.15	19	0	1	0	0	0	0	0	1	
0	0	2	2	3	2	0	1	2	7	7	8	16	4	1	2	179	44.1	50.8	38	44	50.8	50.8	12	1	0	0	0	0	0	1	
1	0	20	10	10	5	1	0	2	7	7	8	14	2	1	1	187	44.1	54	37	40	15.8	52	24	11	1	1	0	0	0	1	
0	1	50	15	26	10	4	0	2	7	7	8	14	2	1	1	160	58	64	47	46	18.5	69	32	12	1	1	0	0	0	1	
0	0	1	yes	yes	yes	0	1	5	7	2	1	5	1	3	152	68	74	35	43	20.3	70	12	12	1	0	0	0	0	1		
0	1	3	2	1	0	1	2	7	7	6	1	2	140	102	129	96	96	19	50	12	0	1	1	0	0	0	0	1			
1	1	20	25	5	3	4	0	1	7	7	19	2	1	2	159	55	80	50	50	50	12	3	1	0	0	0	0	0	1		
0	1	20	10	5	2	1	0	2	7	7	6	1	2	1	170	87	70	64	60	50	21	2	1	0	0	0	0	0	1		
1	0	4	2	1	1	0	2	7	7	8	17	2	1	2	159	73	79	44	44	25.1	77	12	1</								

REDES NEURONALES

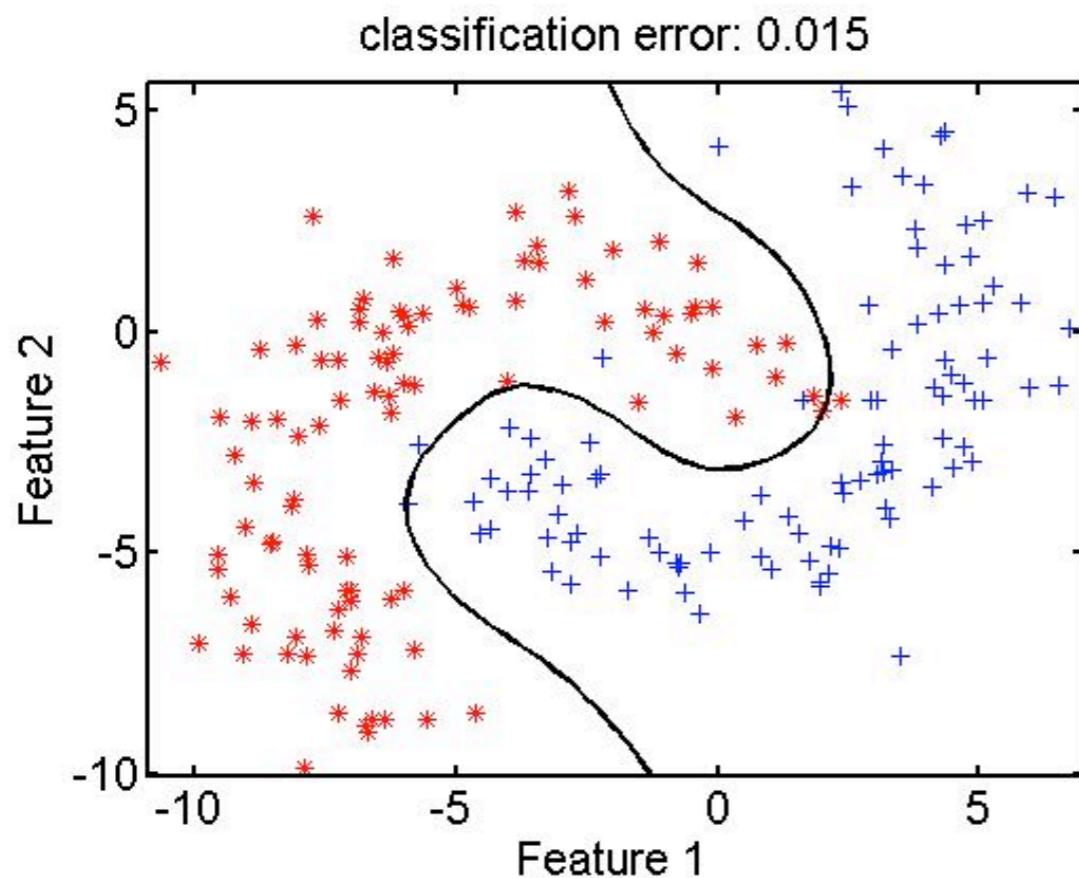
- Aplicaciones

Reconocimiento de patrones



1. Se pasa lenguaje escrito a imágenes de 0 y 1
2. Se entrena la red con muchos ejemplos
3. Una vez entrenada la red es capaz de leer algo escrito a mano (o una matrícula de coche, ...)

Clasificación



- Es capaz de separar objetos en distintas clases
- Problemas de muchas dimensiones y altamente no lineal

Clasificación

1. Graban sonido.
2. Lo clasifican en función del motivo del llanto
(clase 1, Hambre, clase 2 Dolor, clase 3 sueño,
clase 4 mamá...)
3. Entrenan la red
4. Grabas a tu hijo y la red te dirá que le pasa (en realidad te dirá la probabilidad de
que le pase 1, 2, 3 o 4,...)



Clasificación

[World Congress on Medical Physics and Biomedical Engineering 2018](#) pp 809-813 | [Cite as](#)

Baby Cry Recognition Using Deep Neural Networks

1. Graban sonido.

2. Lo clasifican en función del motivo del llanto
(clase 1, Hambre, clase 2 Dolor, clase 3 sueño,
clase 4 mamá...)

4. Entrenan la red

5. Grabas a tu hijo y la red te dirá que le pasa (en realidad te dirá la probabilidad de que le pase 1, 2, 3 o 4,...)



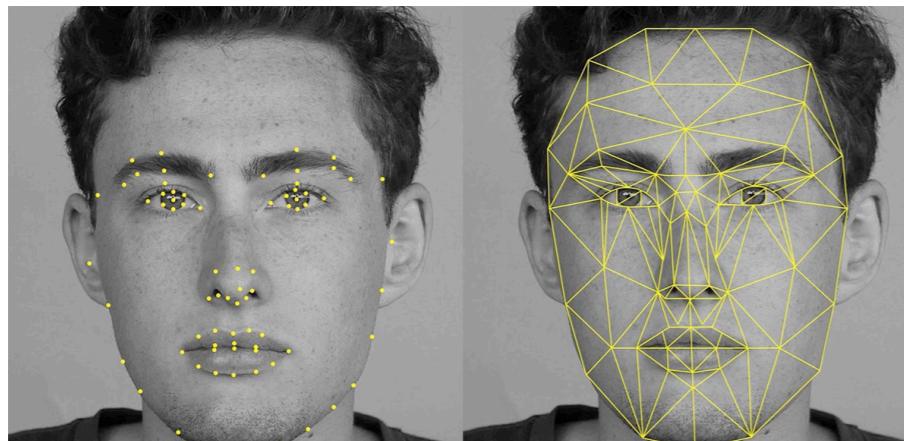
REDES NEURONALES

APLICACIONES

Reconocimiento imágenes. Una máquina lee una imagen como la intensidad en cada pixel.
Eso se puede dar como input a una red neuronal



Identificación de plantas, animales



Reconocimiento facial

Reconocimiento de etiquetas, texto,...



Clasificación de Galaxias

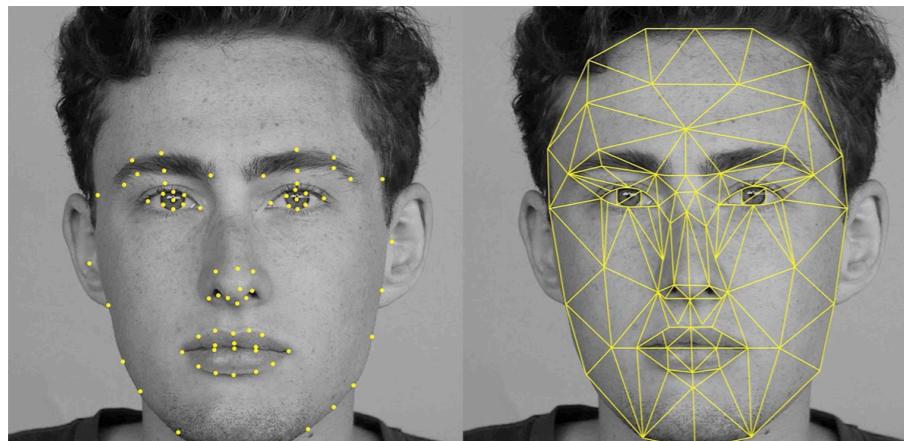
REDES NEURONALES

APLICACIONES

Reconocimiento imágenes. Una máquina lee una imagen como la intensidad en cada pixel.
Eso se puede dar como input a una red neuronal



Identificación de plantas, animales



Reconocimiento facial

Reconocimiento de etiquetas, texto,...



Clasificación de Galaxias

DEEP
LEARNING

REDES NEURONALES

- Validación y otras consideraciones

REDES NEURONALES

EVALUACIÓN: SET DE VALIDACIÓN

Durante el entrenamiento vamos evaluando con la loss función a cada iteración. Mínimos cuadrados para regresión y cross entropy para

Clasificación *loss function*:

$$H(p, q) = - \sum_{x \in \mathcal{X}} p(x) \log q(x)$$

Regresión *loss function*:

$$E = \sum_n \frac{(t^n - y^n)^2}{N}$$

REDES NEURONALES

EVALUACIÓN: SET DE VALIDACIÓN

Durante el entrenamiento vamos evaluando con la loss función a cada iteración. Mínimos cuadrados para regresión y cross entropy para

Clasificación *loss function*:

$$H(p, q) = - \sum_{x \in \mathcal{X}} p(x) \log q(x)$$

Regresión *loss function*:

$$E = \sum_n \frac{(t^n - y^n)^2}{N}$$

REDES NEURONALES

EVALUACIÓN: SET DE VALIDACIÓN

Durante el entrenamiento vamos evaluando con la loss función a cada iteración. Mínimos cuadrados para regresión y cross entropy para

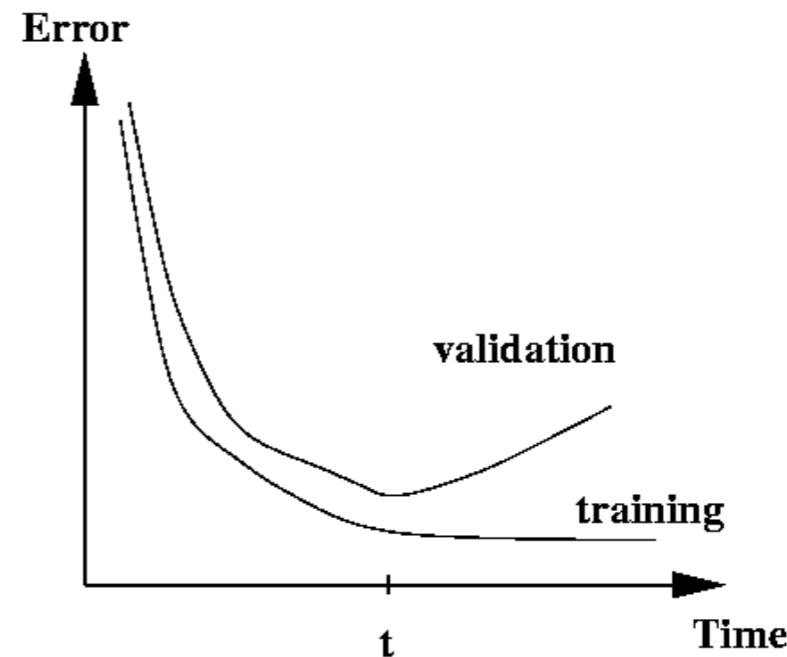
Clasificación *loss function*:

$$H(p, q) = - \sum_{x \in \mathcal{X}} p(x) \log q(x)$$

También calculamos la *loss function* para un **set de validación**, datos conocidos que no hemos usado en el training

Regresión *loss function*:

$$E = \sum_n \frac{(t^n - y^n)^2}{N}$$



REDES NEURONALES

EVALUACIÓN TEST VALIDACIÓN

Una vez terminado el entrenamiento se usan también otros indicadores sobre el set de validación para ver el desempeño de la red. Por ejemplo:

Regresión:

$$RelErr = \frac{1}{N} \sum_n \frac{|t^n - y^n|}{t^n}$$

$$\rho = \frac{\sigma_{ty}}{\sigma_t \sigma_y}$$

REDES NEURONALES

EVALUACIÓN TEST VALIDACIÓN

Una vez terminado el entrenamiento se usan también otros indicadores sobre el set de validación para ver el desempeño de la red. Por ejemplo:

Clasificación

Precisión (o TPR)

$$TPR = \frac{TP}{TP+FP}$$

Compleitud (o recall)

$$Rec = \frac{TP}{TP+FN}$$

Regresión:

$$RelErr = \frac{1}{N} \sum_n \frac{|t^n - y^n|}{t^n}$$

$$\rho = \frac{\sigma_{ty}}{\sigma_t \sigma_y}$$

REDES NEURONALES

ENTRENAMIENTO

Algunas observaciones:

- * Sólo podremos generalizar en el **rango del entrenamiento**. Siendo menos efectivo en los bordes (edge problem)
- * Hay que fijarse que tenemos **suficientes datos** para cubrir bien el rango del problema. Cuantos más datos mejor.
- * En ocasiones puede ser útil entrenar varias redes con training sets distintos y hacer promedio de los resultados.
- * En datos con muchas variables de magnitudes distintas es importante **estandarizar** los inputs y los outputs para que pesen por igual todos

$$x_i = \frac{x_i - \langle x_i \rangle}{\sigma_i}$$

REDES NEURONALES

ENTRENAMIENTO

Algunas observaciones:

* Sólo podremos generalizar en el **rango del entrenamiento**. Siendo menos efectivo en los bordes (edge problem)

* Hay que fijarse que tenemos **suficientes datos** del problema. Cuantos más datos mejor.

* En ocasiones puede ser útil entrenar varias redes hacer promedio de los resultados.

* En datos con muchas variables de magnitudes distintas **estandarizar** los inputs y los outputs para que permanezcan en la misma escala.

$$x_i = \frac{x_i - \langle x_i \rangle}{\sigma_i}$$

Antigüedad	Tamaño	Tamaño jardín	Conflictividad	Precio
20	180	0	0.1	120000
10	130	100	0.2	200000
25	200	20	0.5	130000
2	100	0	0.2	190000
...
5	200	20	0.3	?

REDES NEURONALES

ARQUITECTURA

¿Cuántas capas ocultas hay que poner?
¿Y cuantas neuronas en cada capa?

No hay muchas pistas, hay que probar bastantes configuraciones.

Pista 1. Universal approximation theorem (Cybenko 1989): Una red neuronal feed-forward de una **sola capa** con un número finito de neuronas puede aproximar cualquier función continua real.

Pista 2. Pocas neuronas pueden hacer que no lleguemos a trazar bien el problema y demasiadas puede ser que sobreajustemos (overfitting). Un número para empezar $(N_{inp}+N_{out})/2$, **pero no es una regla fiable!!**

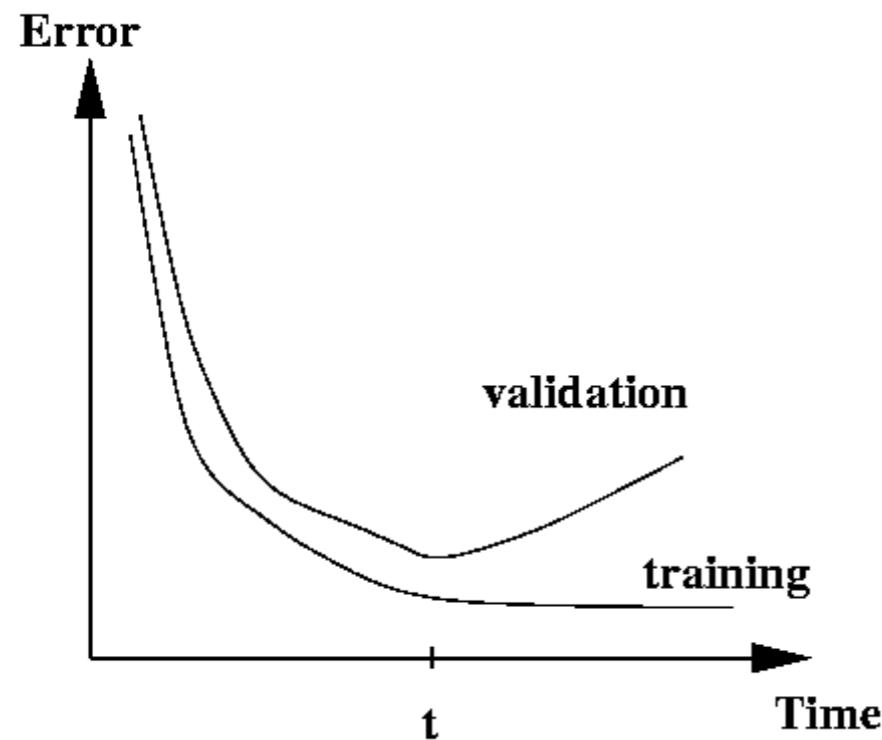
¡Prueba y error!

REDES NEURONALES

ENTRENAMIENTO

¡Vigilad con el overfitting!

Posibles soluciones:



- 1) Estamos usando demasiados parámetros:**
→ reducir capas o número de neuronas por capa
- 2) Early stopping** → Paramos entrenamiento en iteración donde empieza overfitting
- 3) Optimizer options:** Paramos entrenamiento en iteración donde empieza overfitting

REDES NEURONALES

GENERALIZACIÓN

Una vez entrenada la red y validada,
ponemos cualquier input x y nos dará el
resultado que predice la red con una
función matemática.

REDES NEURONALES

GENERALIZACIÓN

Una vez entrenada la red y validada, ponemos cualquier input x y nos dará el resultado que predice la red con una función matemática.

$$y_k = g\left(\left(\sum_j w_{jk} \dots f\left(\sum_i w_{ij} x_i\right)\right)\right)$$

REDES NEURONALES

PREDICTIVE ANALYSES

- Con la digitalización y la tecnología Big Data se dan las condiciones perfectas para sacar el máximo provecho de estas herramientas.
- con DEEP Learning (redes neuronales convolucionales) se obtienen resultados sorprendentemente buenos de clasificación, reconocimiento de imágenes, detección de objetos.



NN con Python

(vemos ejemplo con notebook)

sklearn
(otros Keras, pylearn,...)



mlp=MLPClassifier(hidden_layer_sizes=,activation=,batch_size=,verbose=...)

Objeto donde definimos arquitectura de la red para clasificación

mlp=MLPRegressor(hidden_layer_sizes=,max_iter=,verbose=True,...)

Objeto donde definimos arquitectura de la red para regresión

mlp.fit(x,y) Entrena la red

mlp.predict(xnew) Hace predicciones, nos da ynew para xnew

confusion_matrix(t,predictions) True positives and false positives de cada clase

classification_report(t,predictions) precision and recall for classification problems

REDES NEURONALES

Entrenamiento

Una red neuronal no es muy difícil de programar, pero hay paquetes que ya tienen todo implementado, con mucha flexibilidad y fáciles de usar:



REDES NEURONALES

Entrenamiento

Una red neuronal no es muy difícil de programar, pero hay paquetes que ya tienen todo implementado, con mucha flexibilidad y fáciles de usar:

