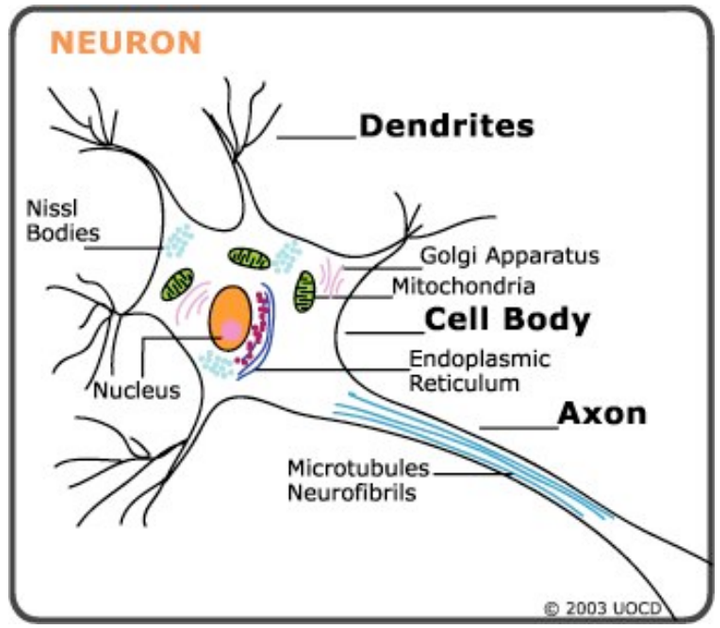
Redes Neuronales Artificiales

https://github.com/biuse/2021NeuralNetworks

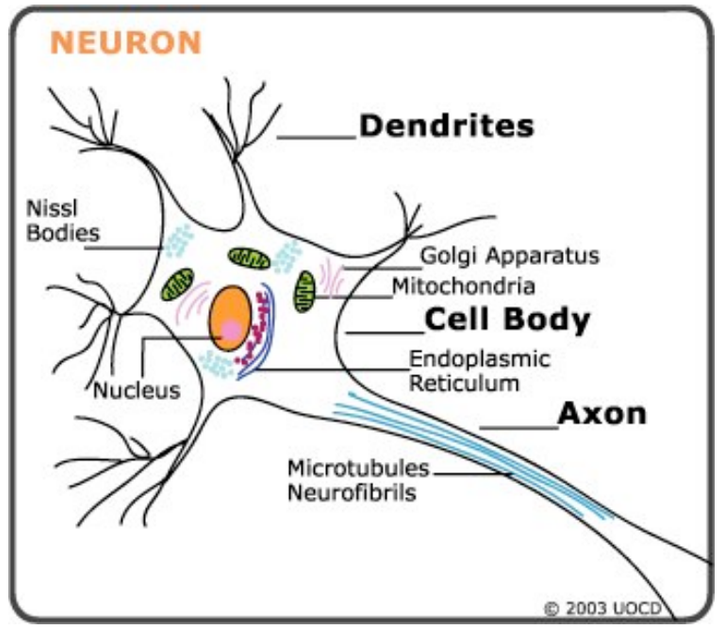
REDES NEURONALES ARTIFICIALES

Herramienta de *machine learning* que nació con dos objetivos:
- Modelizar las neuronas biológicas
- Inteligencia artificial

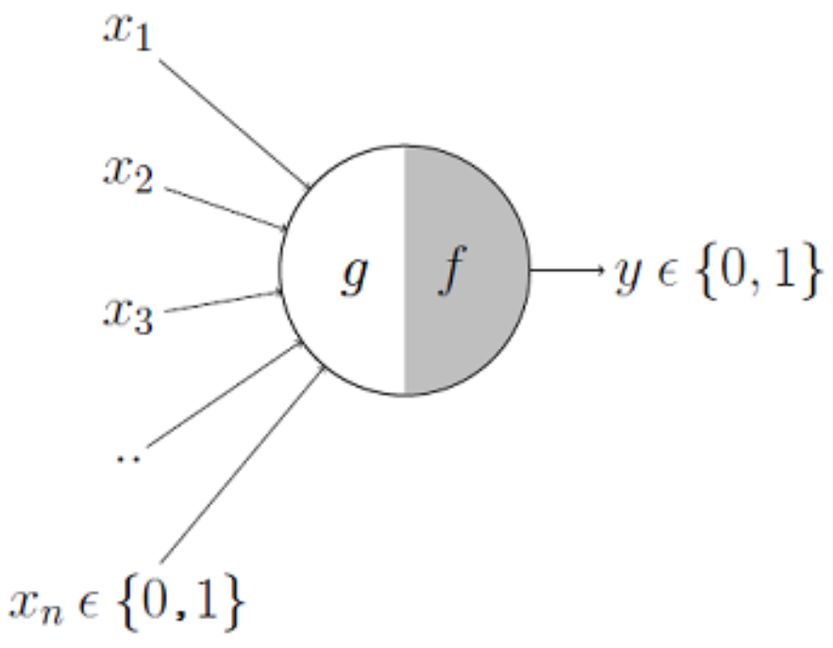


REDES NEURONALES ARTIFICIALES

Herramienta de *machine learning* que nació con dos objetivos:
- Modelizar las neuronas biológicas
- Inteligencia artificial



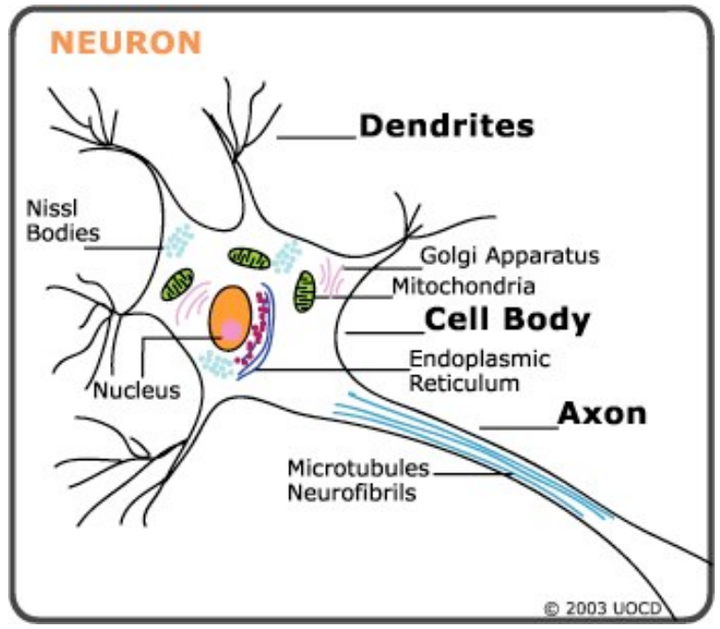
Neurona artificial recibe una señal y devuelve 0 o 1



Modelo de McCullcoch and Pitts1943

REDES NEURONALES ARTIFICIALES

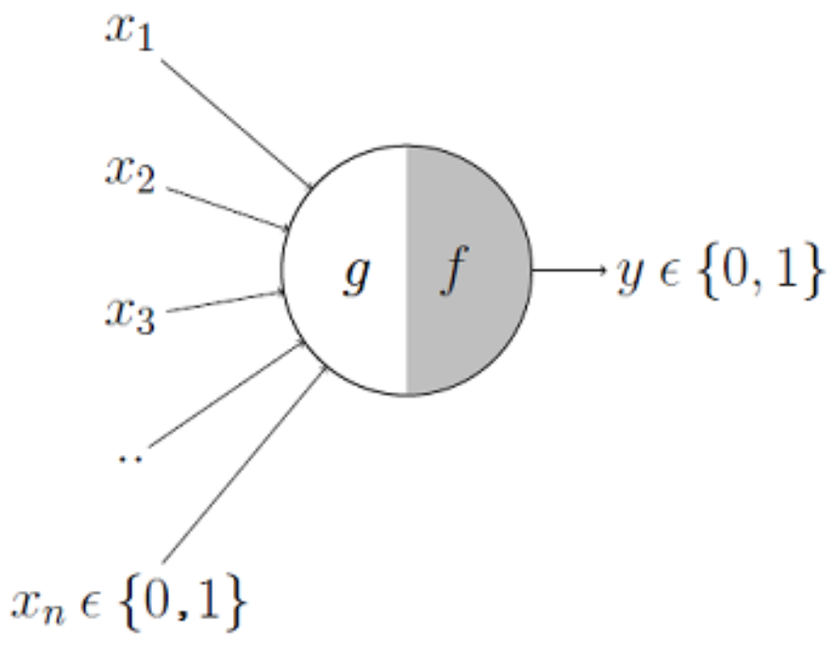
Herramienta de *machine learning* que nació con dos objetivos:
- Modelizar las neuronas biológicas
- Inteligencia artificial



Neurona artificial recibe una señal y devuelve 0 o 1

$$f(g(\mathbf{x})) = 1 \quad if \quad g(\mathbf{x}) \geq \theta$$
$$= 0 \quad if \quad g(\mathbf{x}) < \theta$$

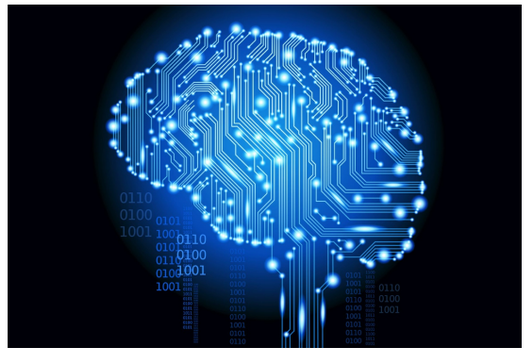
$$g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$



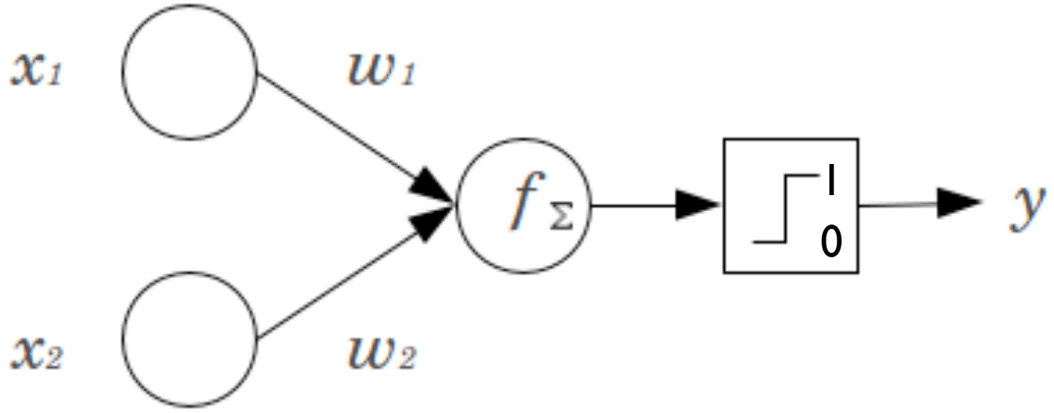
Modelo de McCullcoch and Pitts1943

REDES NEURONALES

Herramienta de *machine learning* que nació con dos objetivos:
-modelizar las neuronas biológicas
-la inteligencia artificial



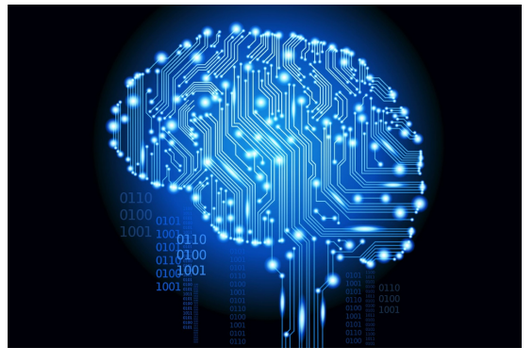
PERCEPTRON



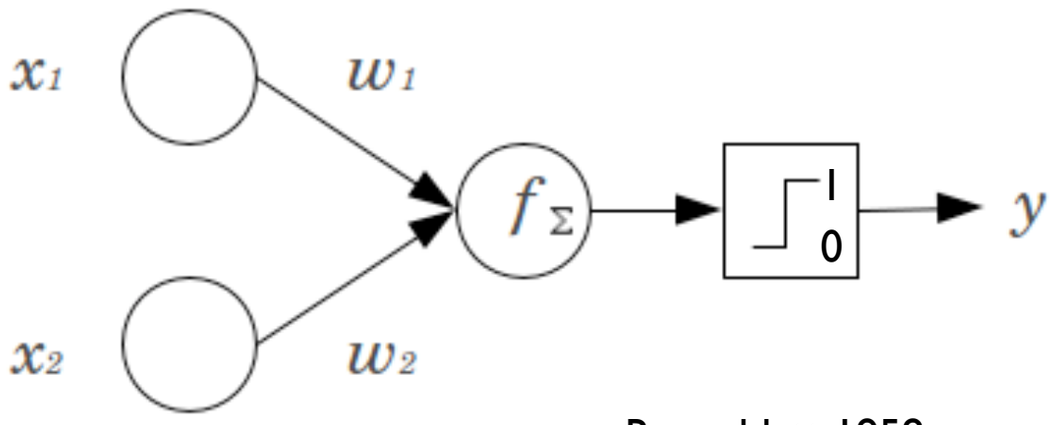
Rosenblatt 1958

REDES NEURONALES

Herramienta de *machine learning* que nació con dos objetivos:
- -modelizar las neuronas biológicas
- -la inteligencia artificial



PERCEPTRON

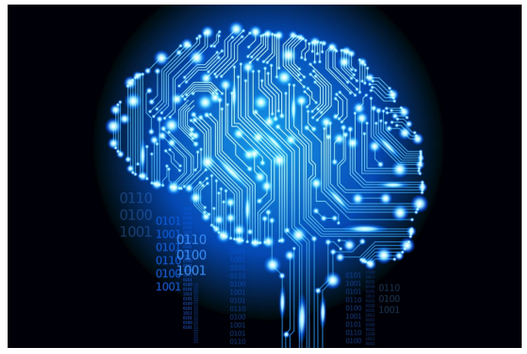


Rosenblatt 1958

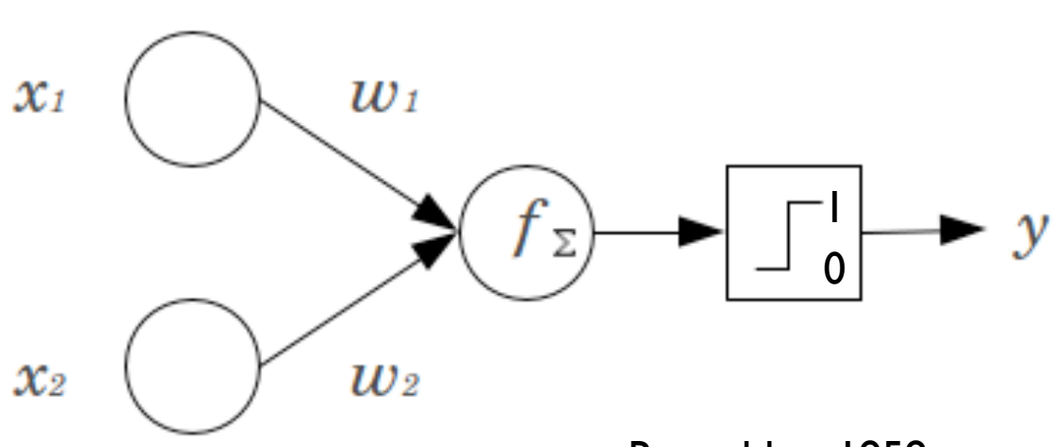
Se añadieron unos **pesos** que podían ajustarse.

REDES NEURONALES

Herramienta de *machine learning* que nació con dos objetivos:
- -modelizar las neuronas biológicas
- -la inteligencia artificial



PERCEPTRON



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

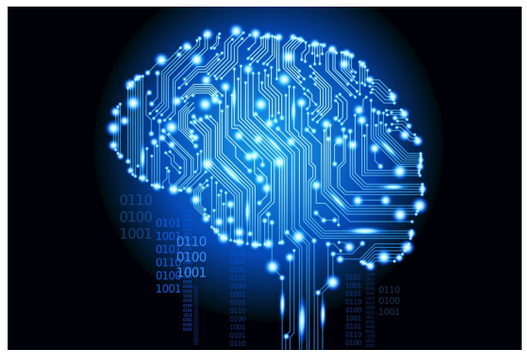
Rosenblatt 1958

$$f(g(\mathbf{x})) = 1 \quad if \quad g(\mathbf{x}) \geq \theta$$
$$= 0 \quad if \quad g(\mathbf{x}) < \theta$$

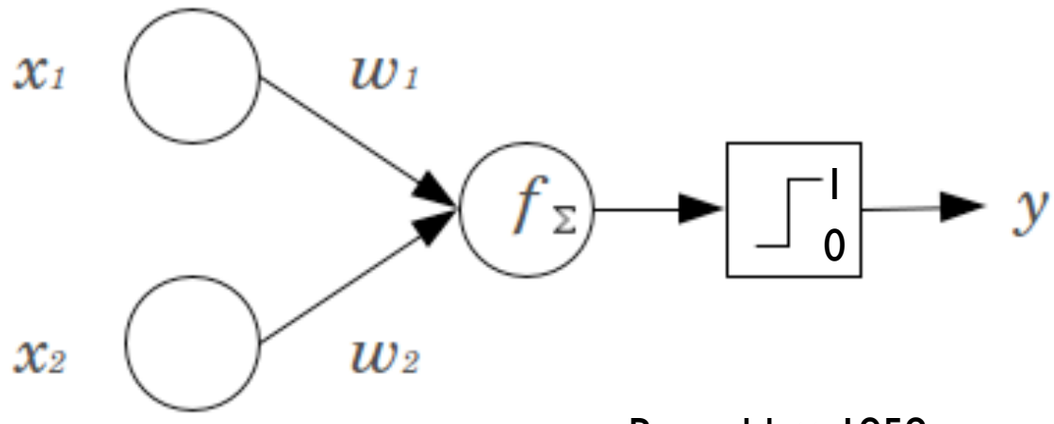
$$g(x) = \sum_i w_i x_i$$

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.

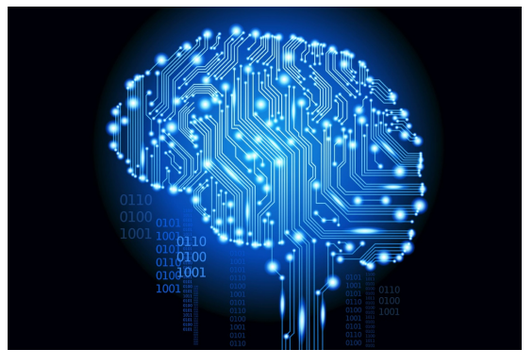
¡Y funcionó!

$$f(g(\mathbf{x})) = 1 \quad if \quad g(\mathbf{x}) \geq \theta$$
$$= 0 \quad if \quad g(\mathbf{x}) < \theta$$

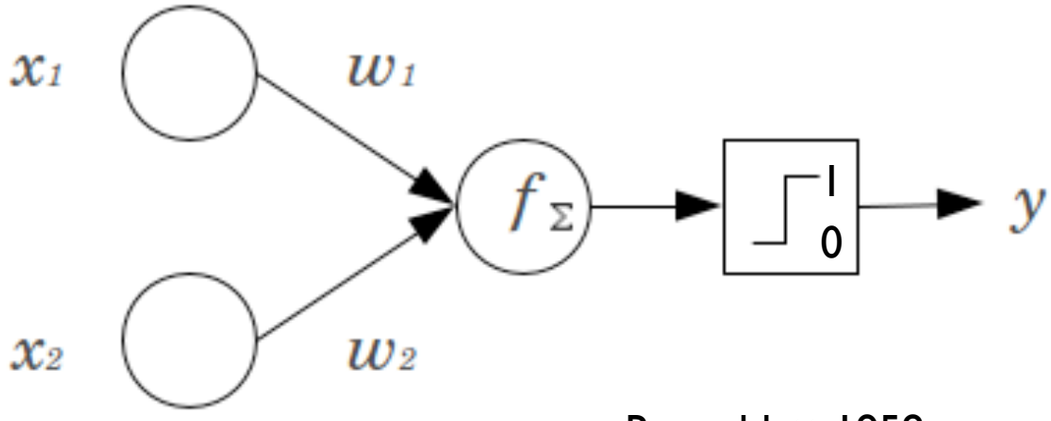
$$g(x) = \sum_i w_i x_i$$

REDES NEURONALES

Herramienta de *machine learning* que nació con dos objetivos:
-modelizar las neuronas biológicas
-la inteligencia artificial



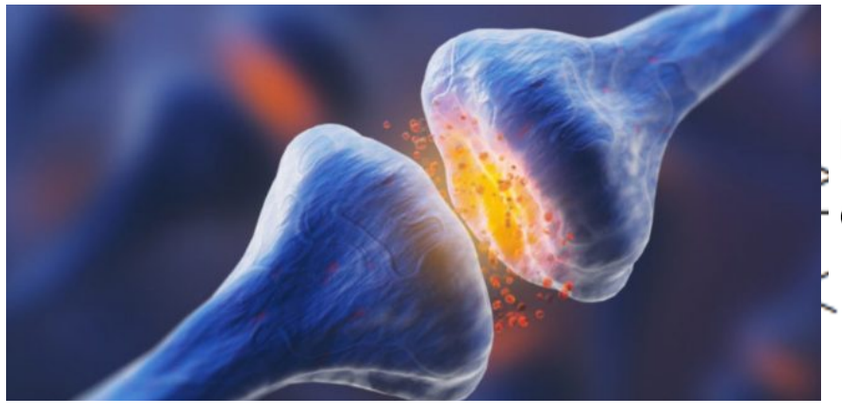
PERCEPTRON



Rosenblatt 1958

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

¡Y funcionó!

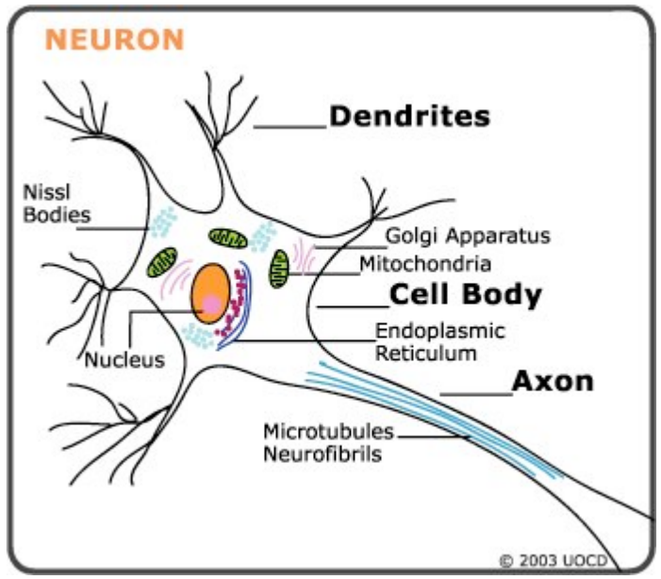


La **sinápsis**, la fortaleza de la conexión entre neuronas, era fundamental para el aprendizaje

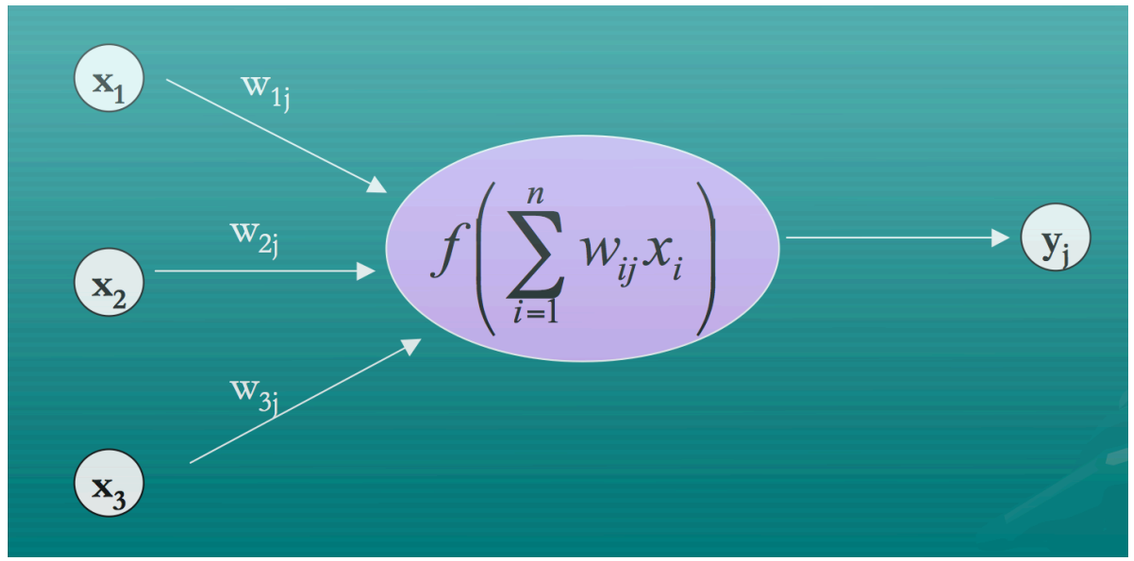
$$\sum_i$$

REDES NEURONALES

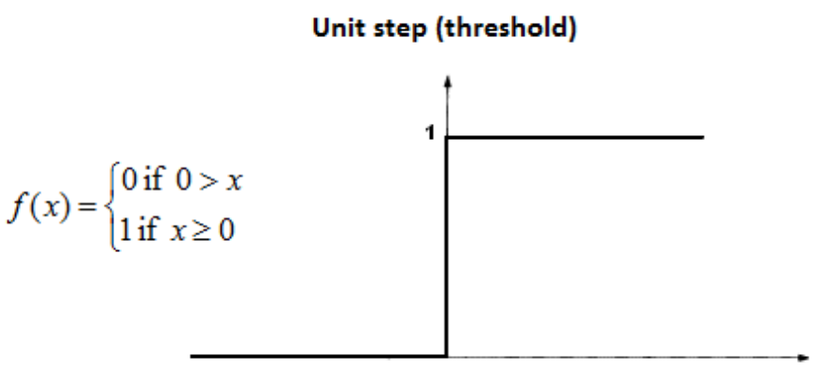
Neurona biológica (simplificación)

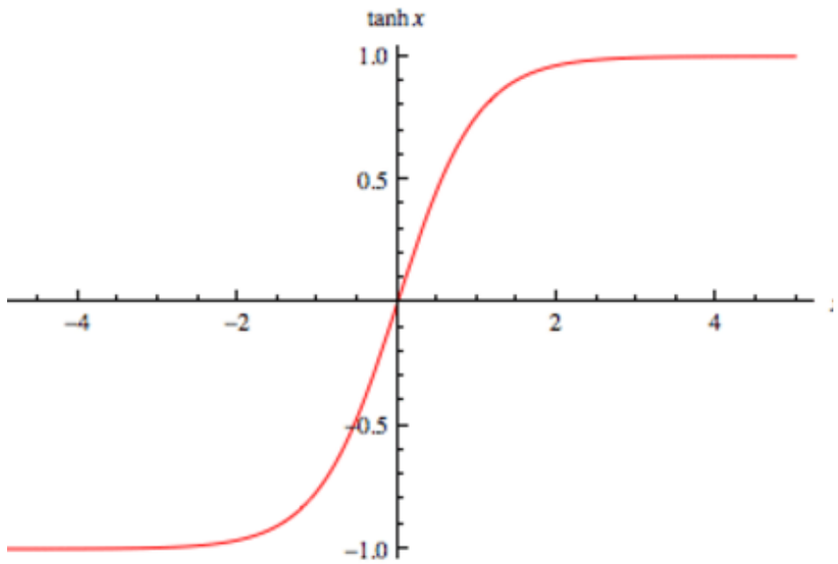


Neurona artificial



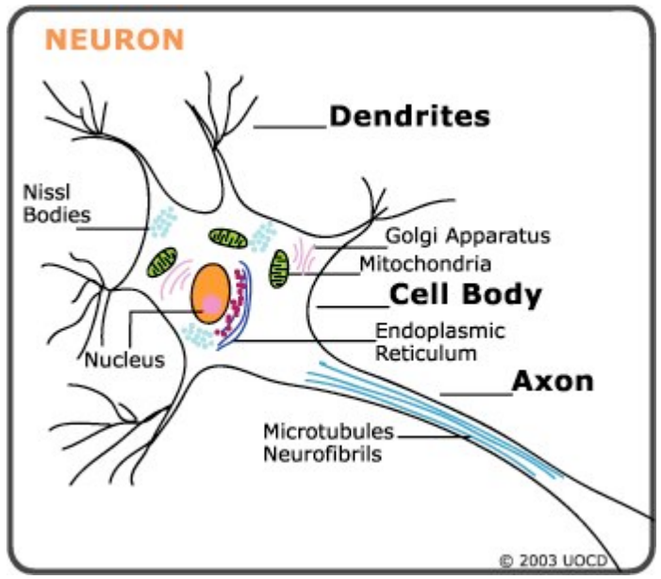
Se generalizó step function —> función de activación f(x)



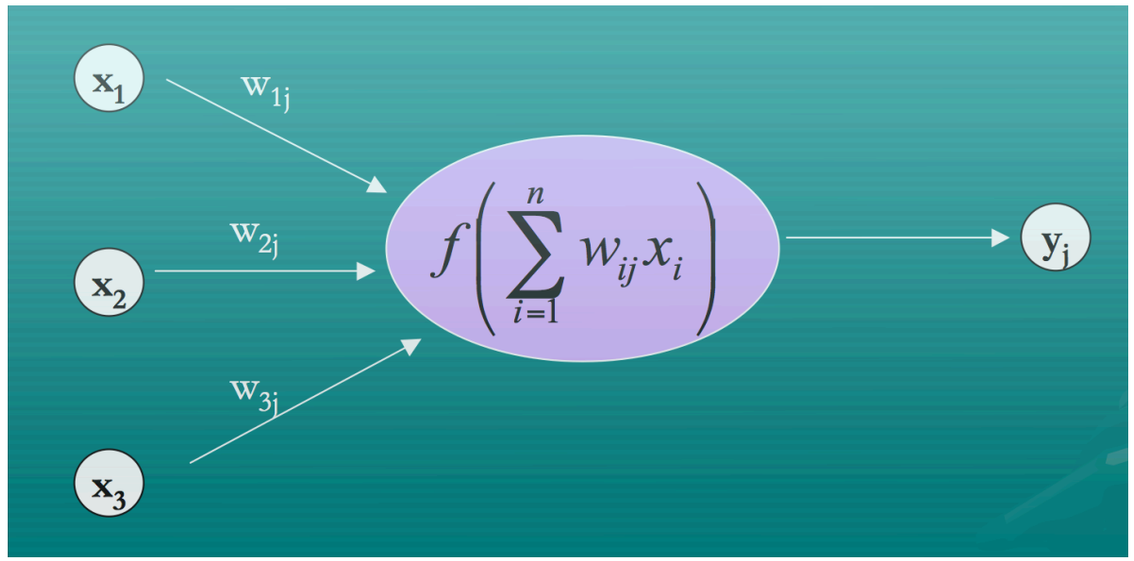


REDES NEURONALES

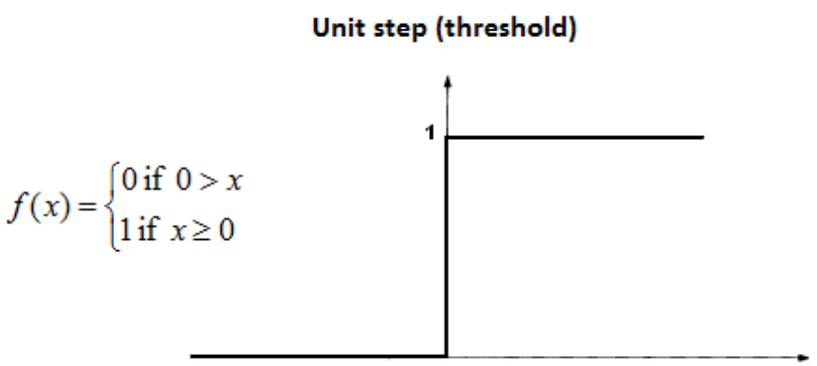
Neurona biológica (simplificación)

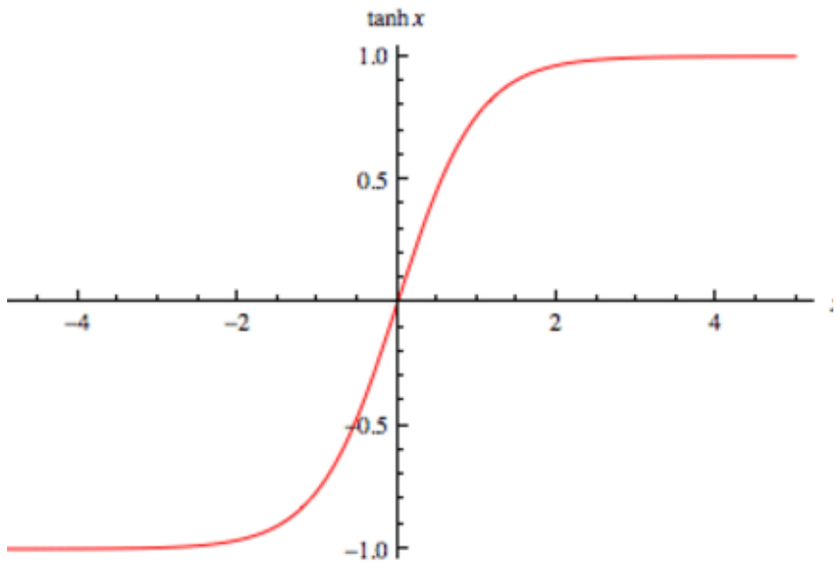


Neurona artificial



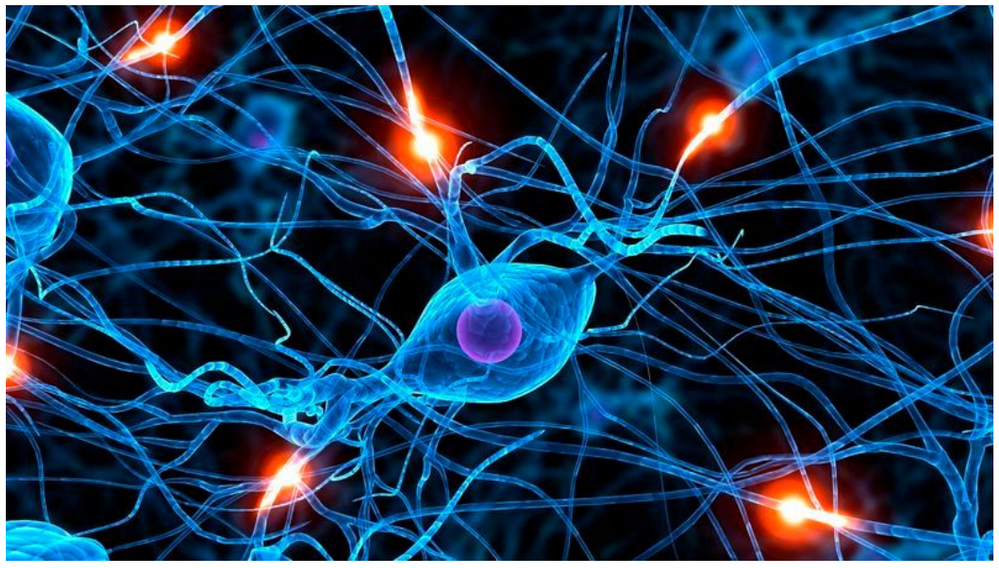
Se generalizó step function —> función de activación f(x)





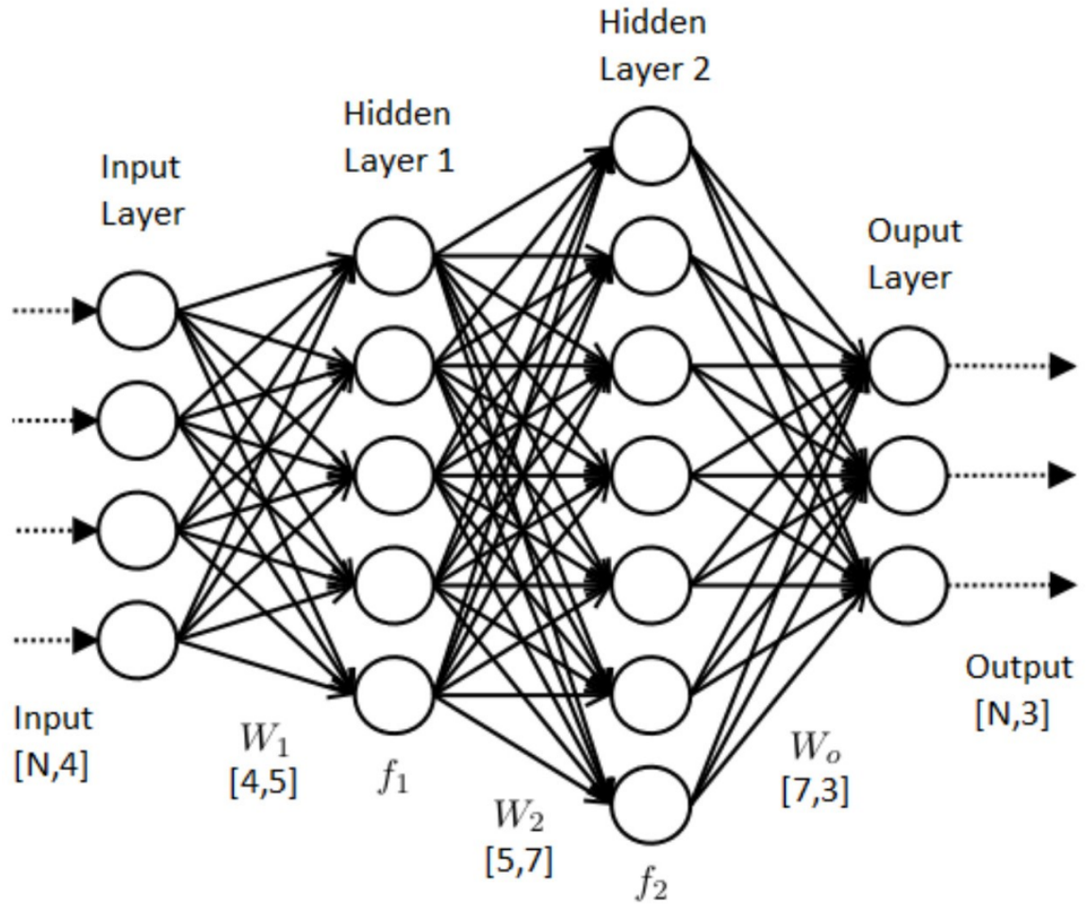
REDES NEURONALES

Red neuronal biológica



Red neuronal artificial

Se organiza por capas de neuronas conectadas entre ellas.



Una ANN son las neuronas conectadas entre sí.

REDES NEURONALES

TIPOS

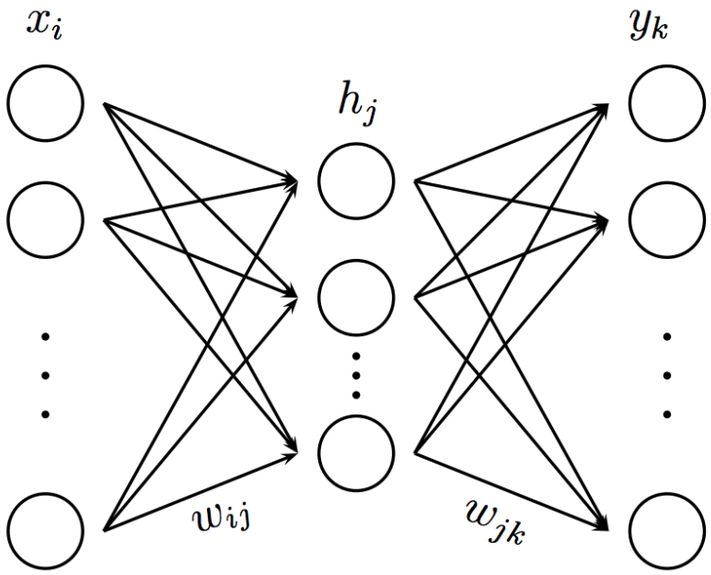
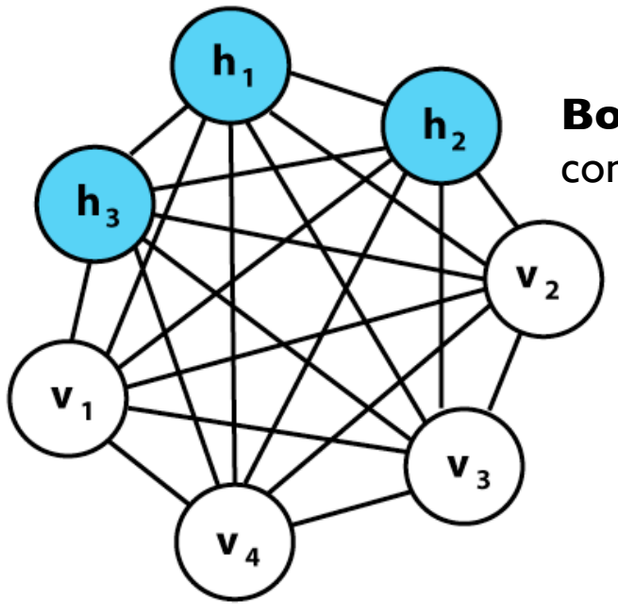


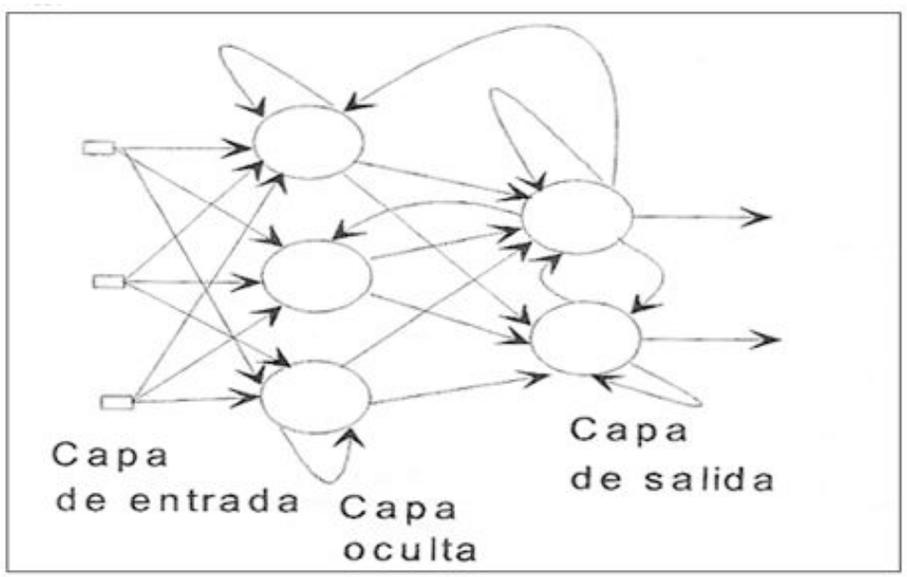
Figure 1. Schematic of a 3-layer feed-forward neural network.

Feed-forward. Solo conexiones hacia delante



Boltzmann machine. Todas conectadas con todas

Redes neuronales recurrentes. Pueden haber conexiones en varias direcciones



Redes neuronales convolucionales. deep learning

Images

REDES NEURONALES

TIPOS

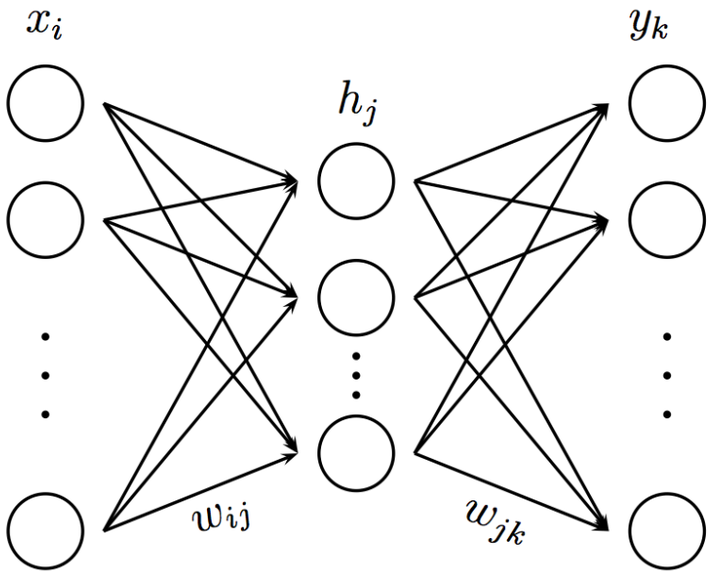
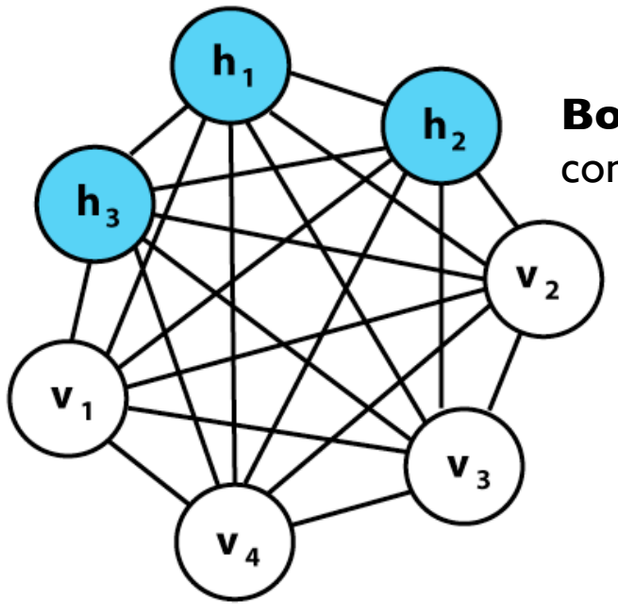


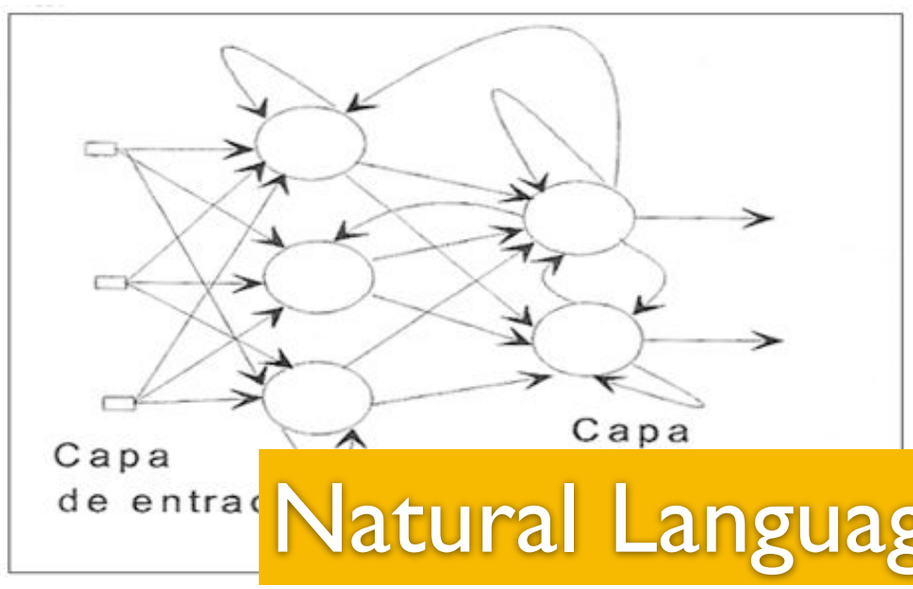
Figure 1. Schematic of a 3-layer feed-forward neural network.

Feed-forward. Solo conexiones hacia delante



Boltzmann machine. Todas conectadas con todas

Redes neuronales recurrentes. Pueden haber conexiones en varias direcciones



Redes neuronales convolucionales. deep learning

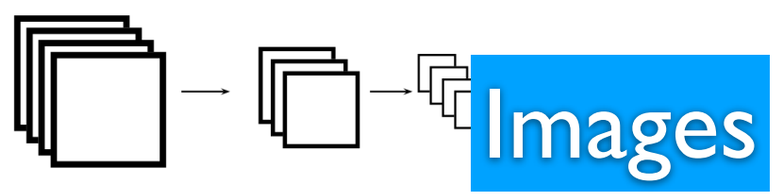


Figure 2. Schematic convolutional neural network example, with 2 hidden layers in the CNN part and 1 hidden layer in the FC part. In this example, the input depth is $k = 4$, the first layer depth $m_1 = 3$ and the second layer depth is $m_2 = 4$, the number of hidden nodes in the FC hidden layer is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

REDES NEURONALES

TIPOS

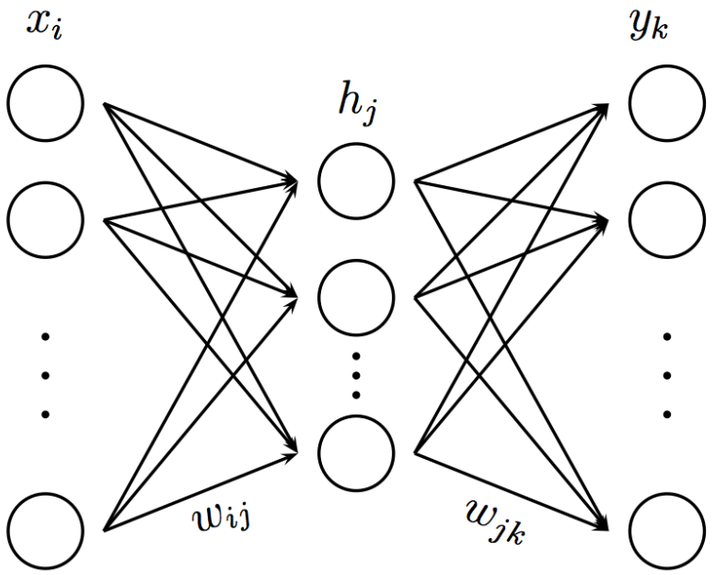
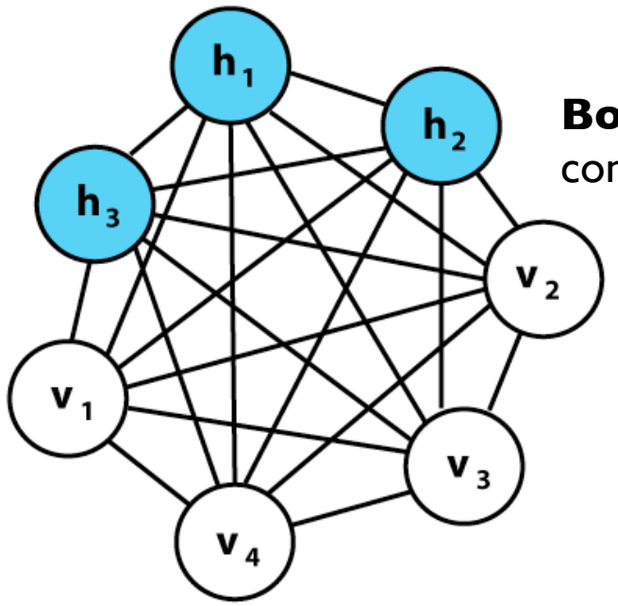


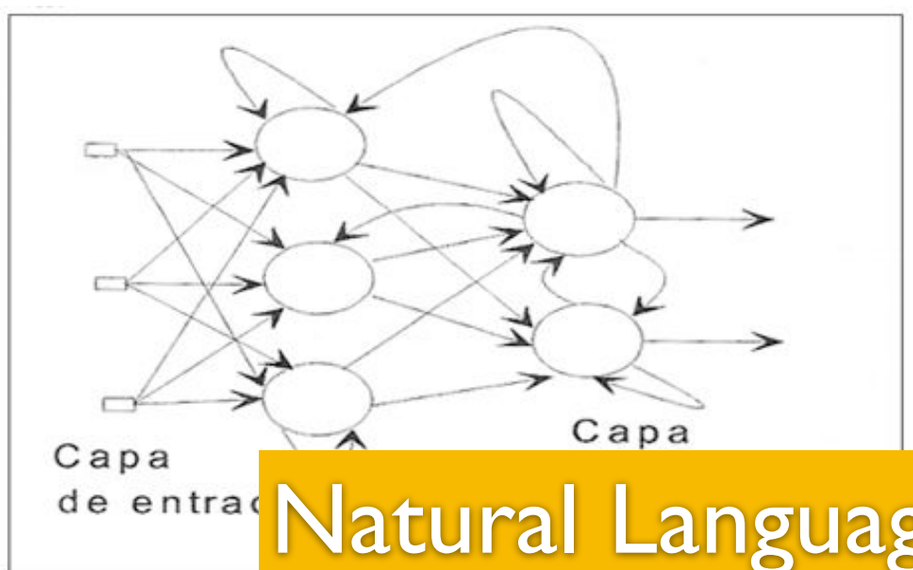
Figure 1. Schematic of a 3-layer feed-forward neural network.

Feed-forward. Solo conexiones hacia delante



Boltzmann machine. Todas conectadas con todas

Redes neuronales recurrentes. Pueden haber conexiones en varias direcciones



Redes neuronales convolucionales. deep learning

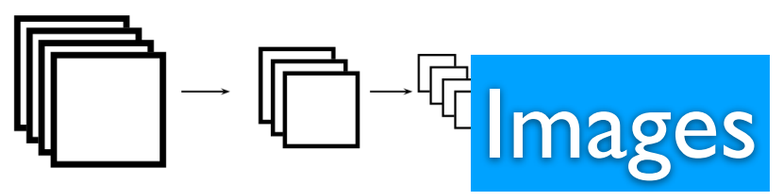


Figure 2. Schematic convolutional neural network example, with 2 hidden layers in the CNN part and 1 hidden layer in the FC part. In this example, the input depth is $k = 4$, the first layer depth $m_1 = 3$ and the second layer depth is $m_2 = 4$, the number of hidden nodes in the FC hidden layer is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

REDES NEURONALES

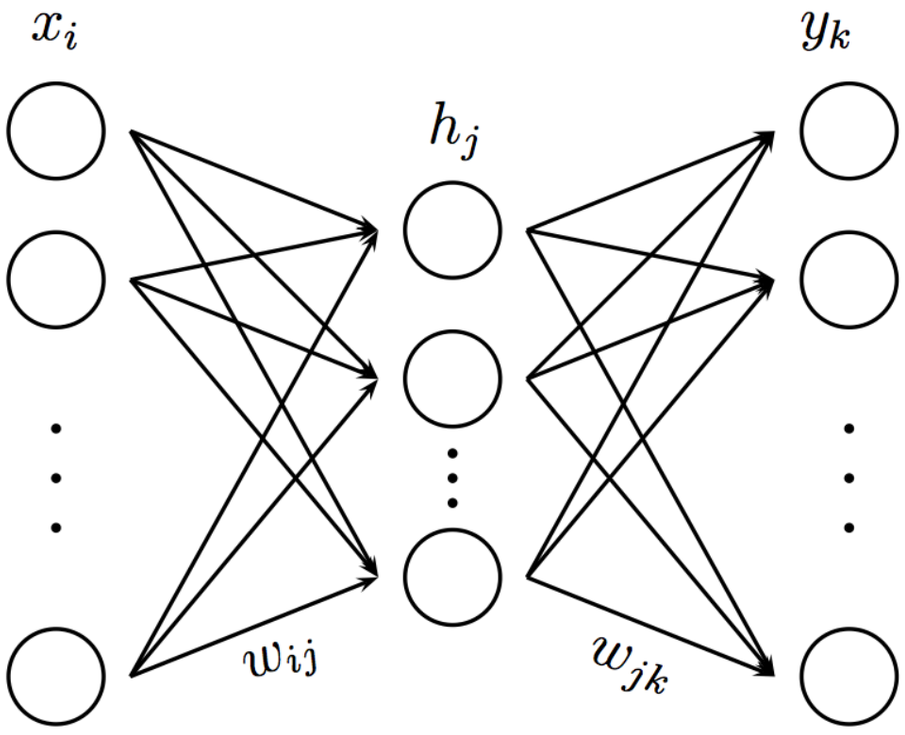


Figure 1. Schematic of a 3-layer feed-forward neural network.

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

REDES NEURONALES

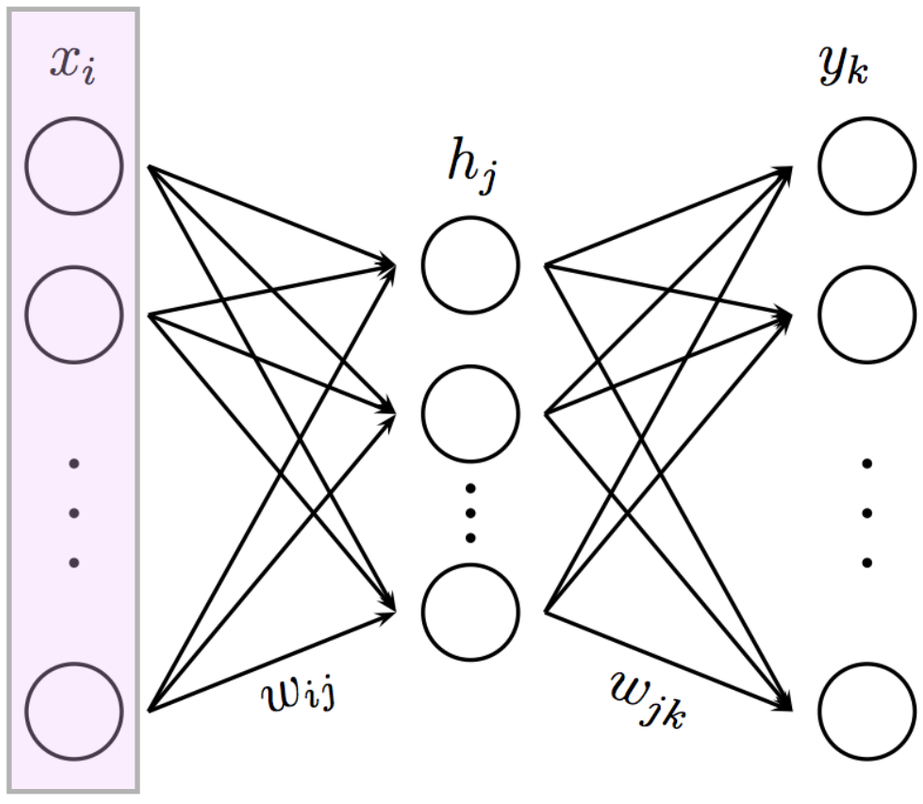


Figure 1. Schematic of a 3-layer feed-forward neural network.

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

Están la capa inicial de **inputs** x

REDES NEURONALES

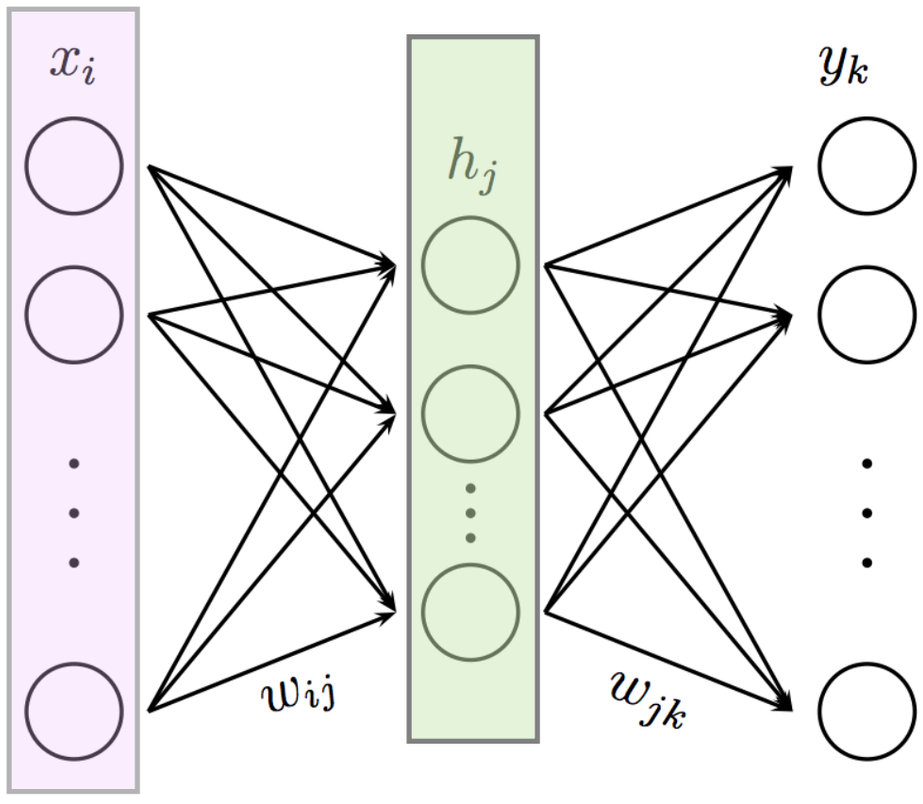


Figure 1. Schematic of a 3-layer feed-forward neural network.

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

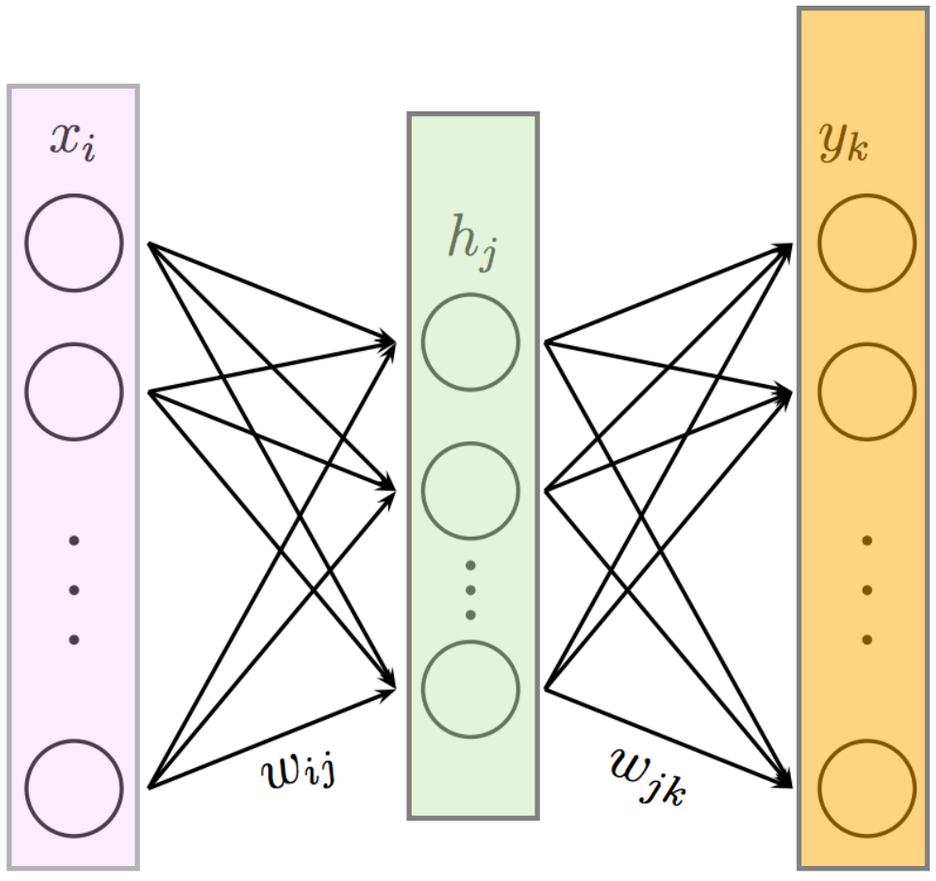


Figure 1. Schematic of a 3-layer feed-forward neural network.

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.

REDES NEURONALES

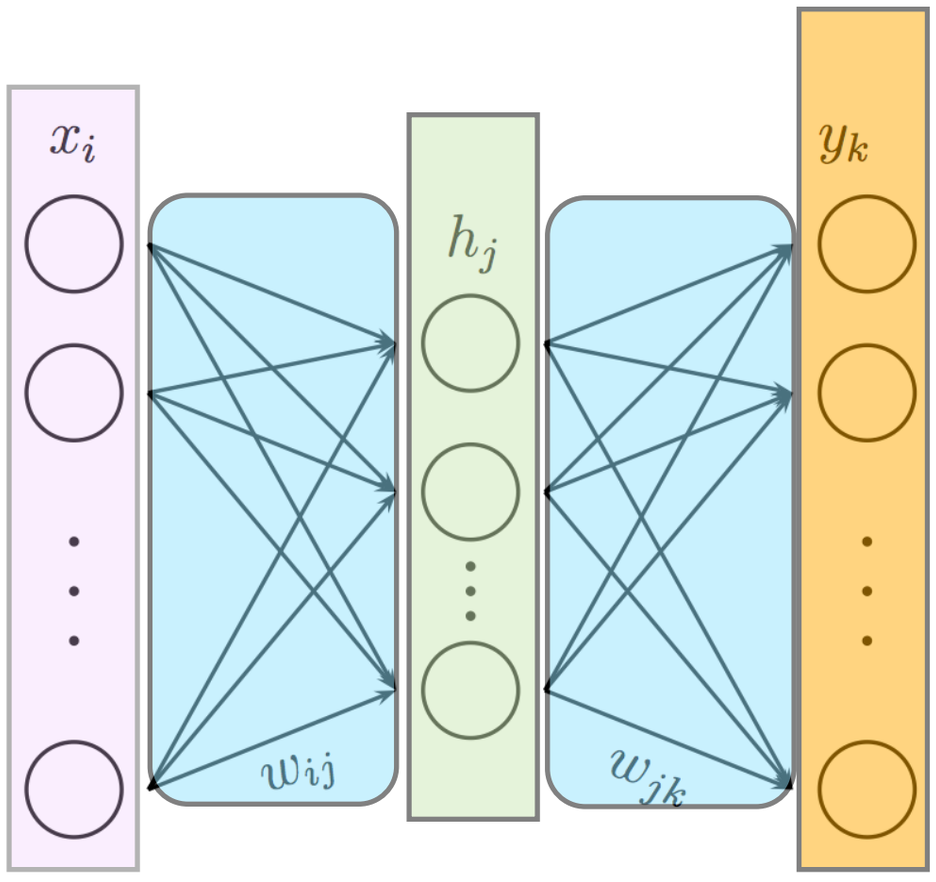


Figure 1. Schematic of a 3-layer feed-forward neural network.

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

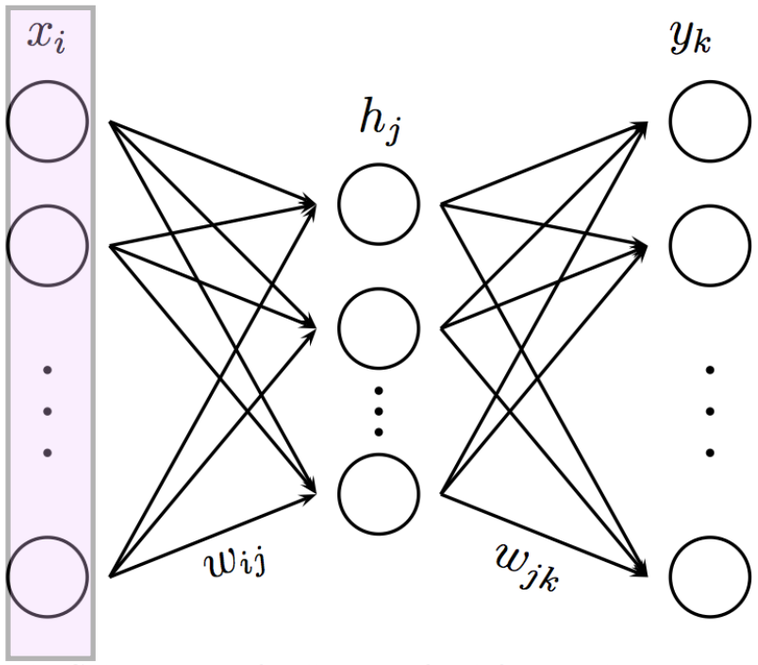


Figure 1. Schematic of a 3-layer feed-forward neural network.

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

REDES NEURONALES

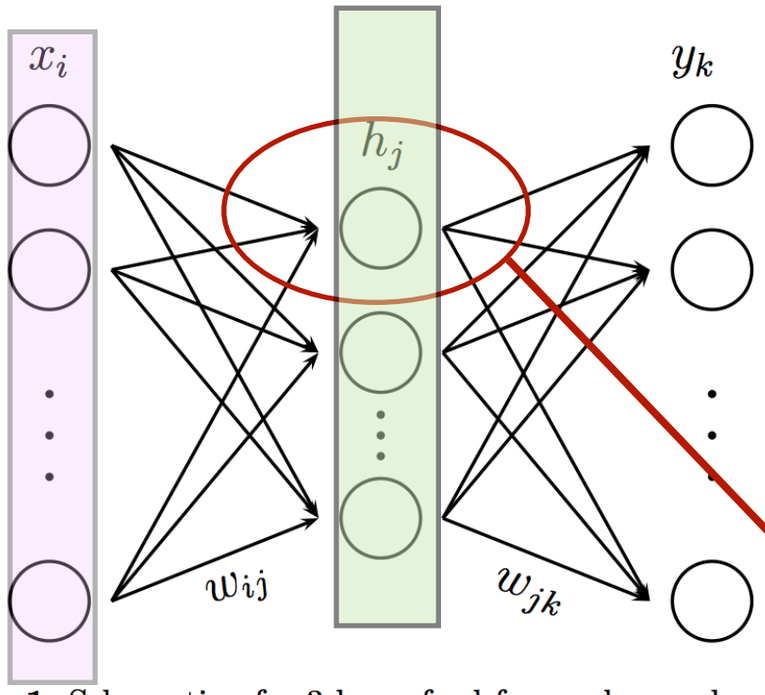
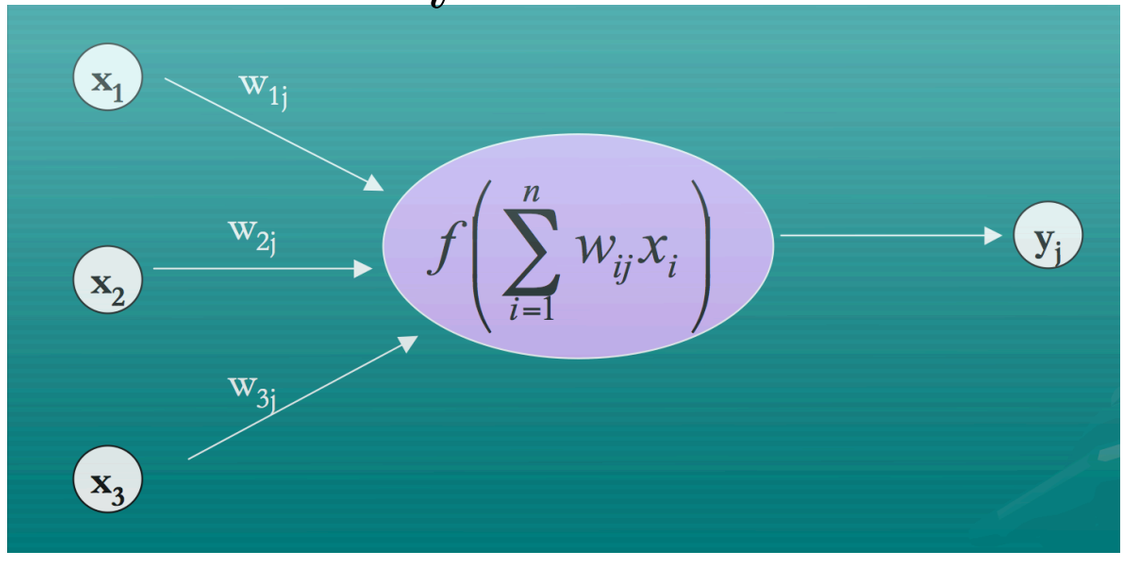


Figure 1. Schematic of a 3-layer feed-forward neural network.

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

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

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



REDES NEURONALES

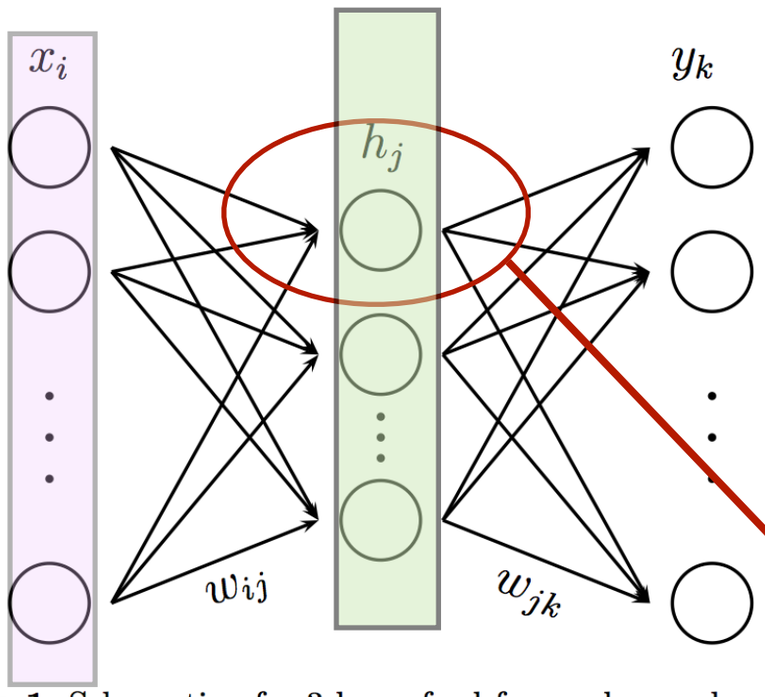


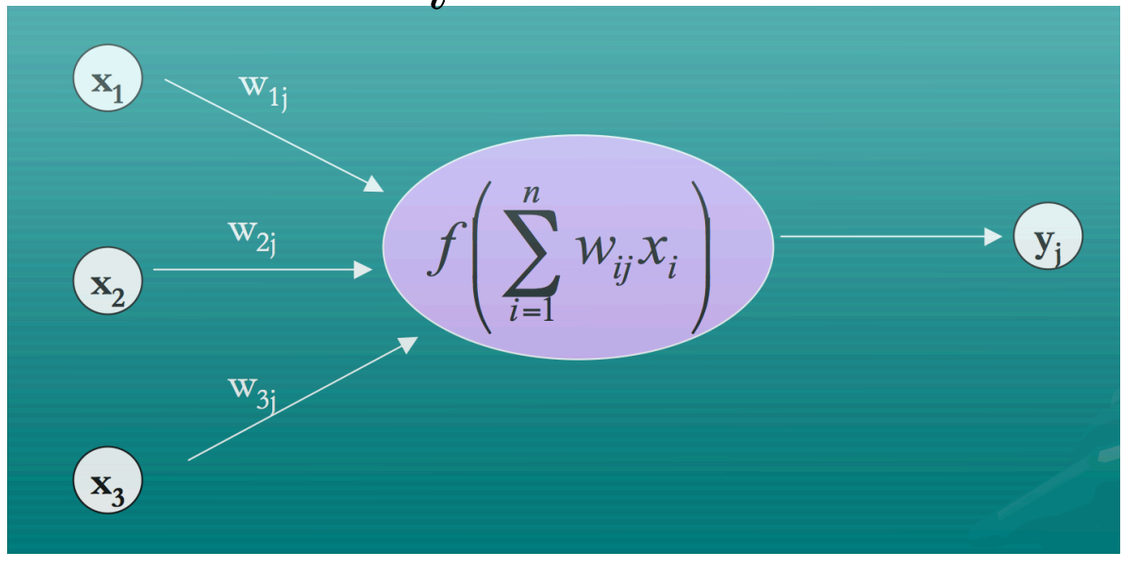
Figure 1. Schematic of a 3-layer feed-forward neural network.

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

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

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

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



REDES NEURONALES

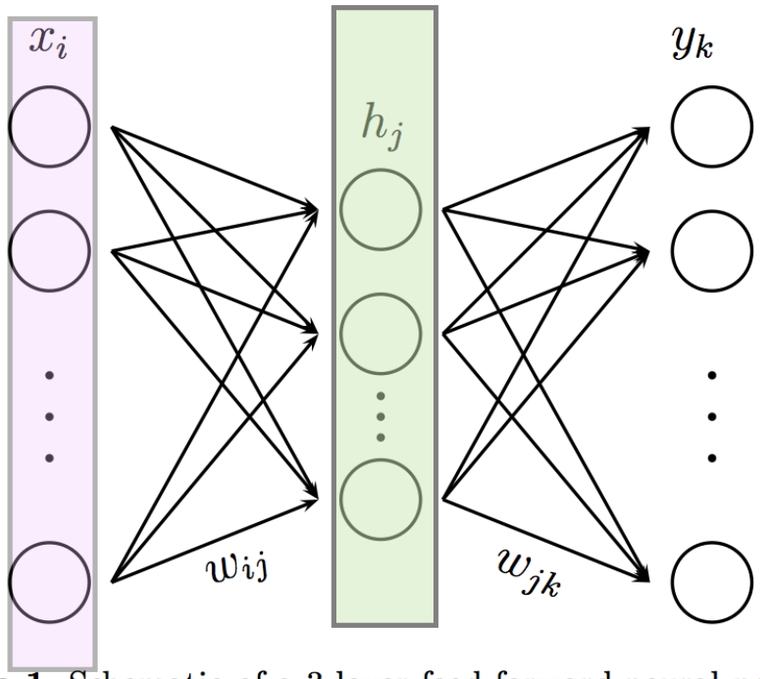


Figure 1. Schematic of a 3-layer feed-forward neural network.

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

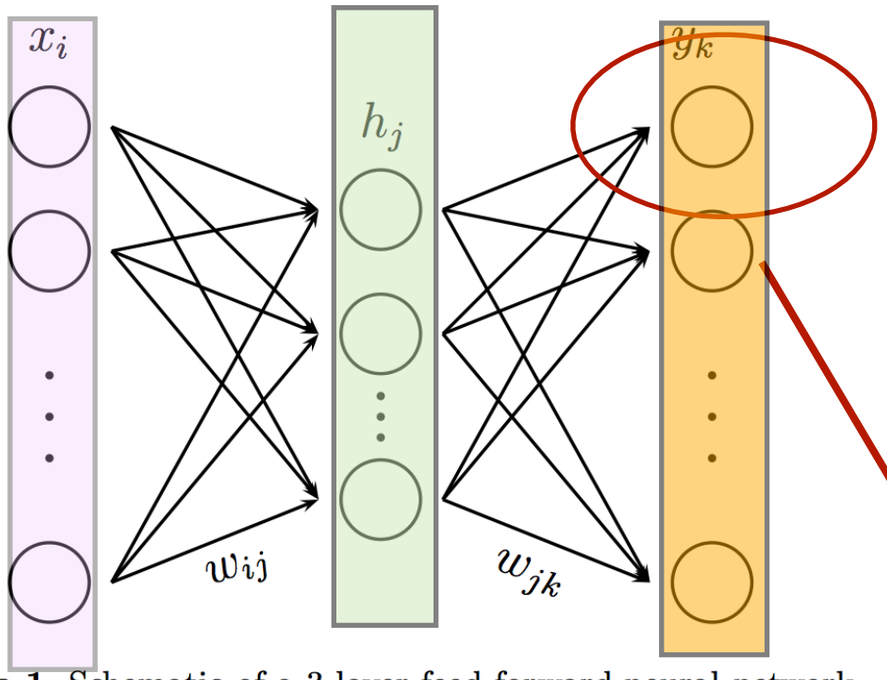


Figure 1. Schematic of a 3-layer feed-forward neural network.

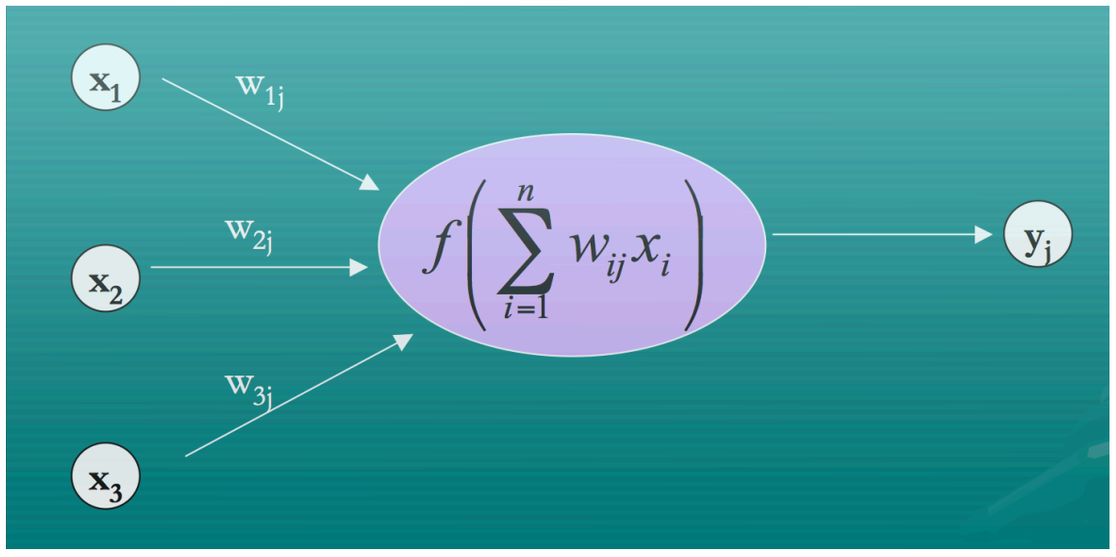
Si hubiera una siguiente capa oculta vuelve a pasar lo mismo.

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

La función que aplicamos no tiene porque ser la misma

Y así hasta la última capa

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



REDES NEURONALES

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

Forma general:

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

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:

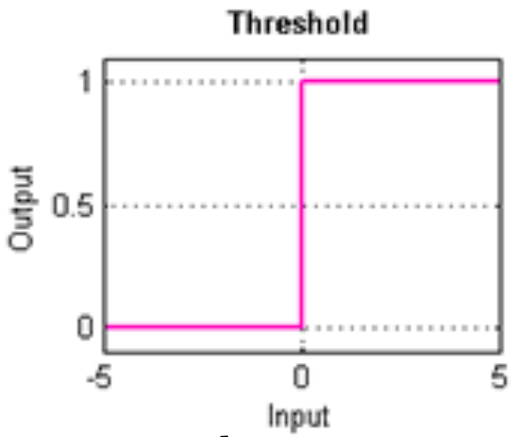
$$y_k = g\left(w_{vk} \cdot \ldots \cdot f\left(\sum_i w_{ij} x_i + \theta_j \right) + w_v \right)$$

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

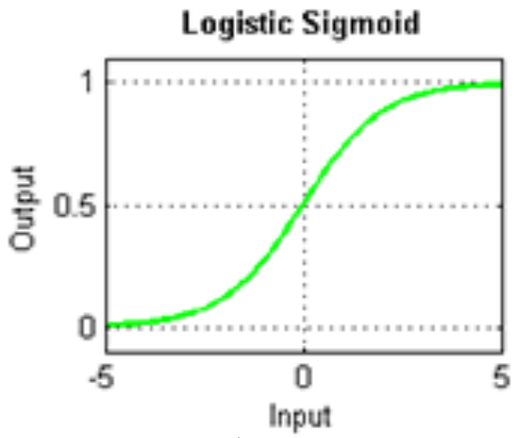
FUNCIONES DE ACTIVACIÓN

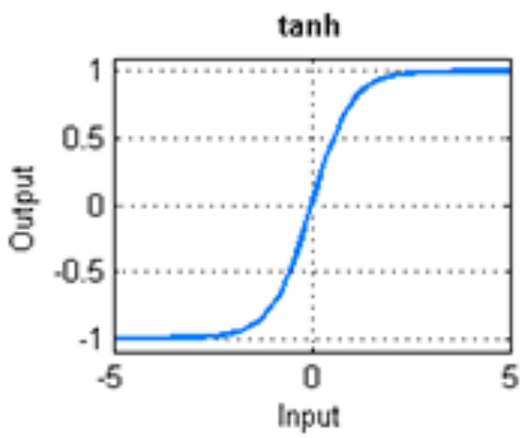


Función de Heaviside o función escalón.

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

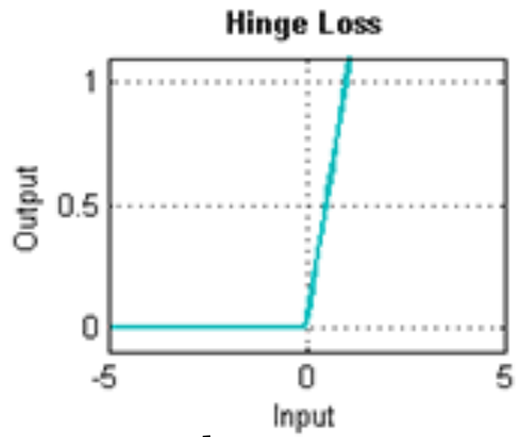
Originalmente se creía que las neuronas biológicas se **activaban o no**. Por eso esta fue la función escogida en los primeros modelos.





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

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



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 porqué queremos saber un valor.

2) Primera capa 4 neuronas última 1.

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 x nhid + nhid x nout + nhid + nout) y **muchos datos** por lo que se usa un proceso ligeramente más elaborado.

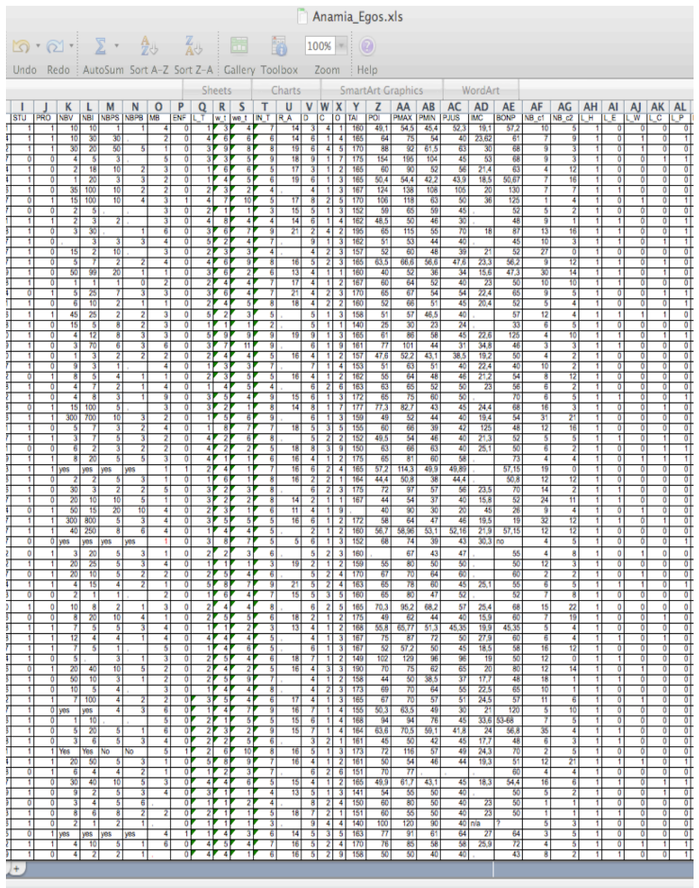
REDES NEURONALES

- 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:



Minimizamos loss function

w_{ij}^{0} = random numbers

$\longrightarrow w_{ij}^{iter1}$

Minimizamos E

$w_{ij}^{0} = w_{ij}^{iter1}$

$\longrightarrow w_{ij}^{iter2}$

Minimizamos E

$w_{ij}^{0} = w_{ij}^{iterN-1}$

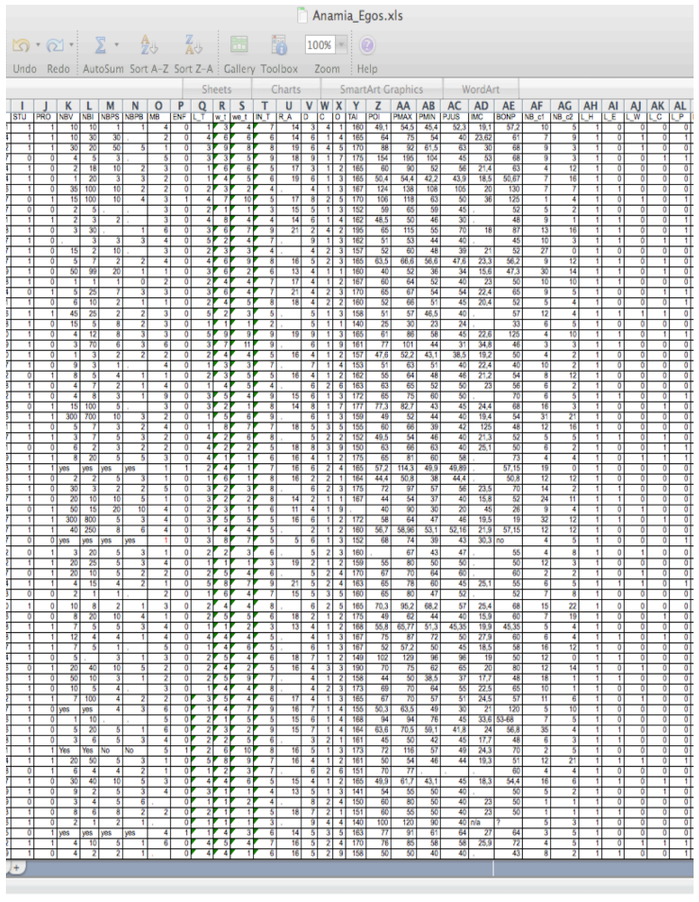
$\longrightarrow w_{ij}^{iterN}$

Batch_size =tamaño de la muestra

REDES NEURONALES

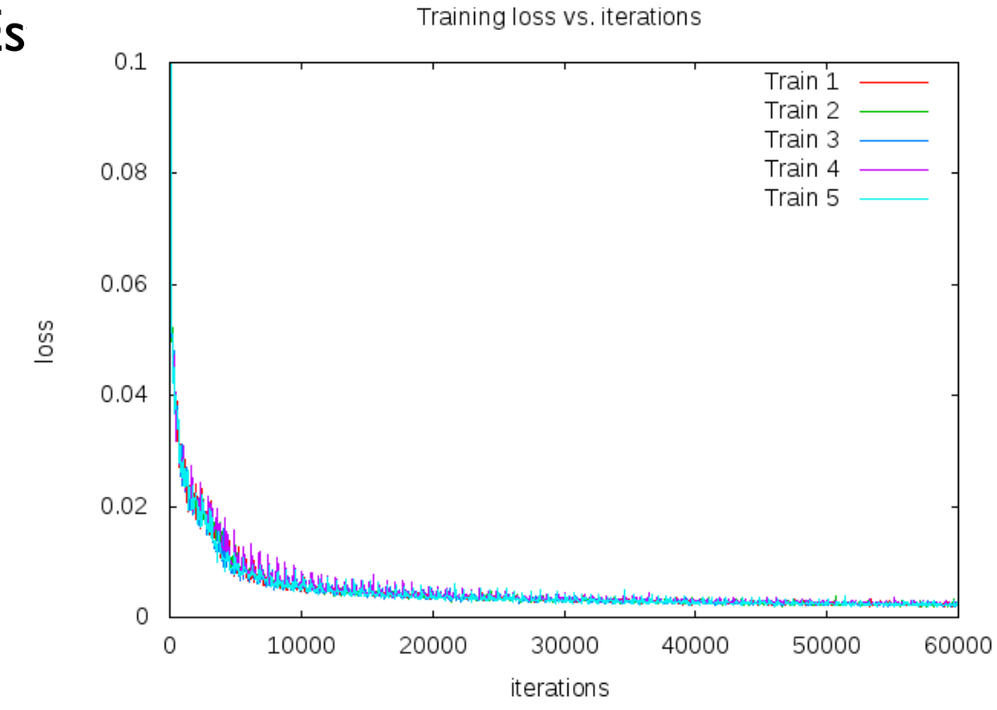
ENTRENAMIENTO

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:



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

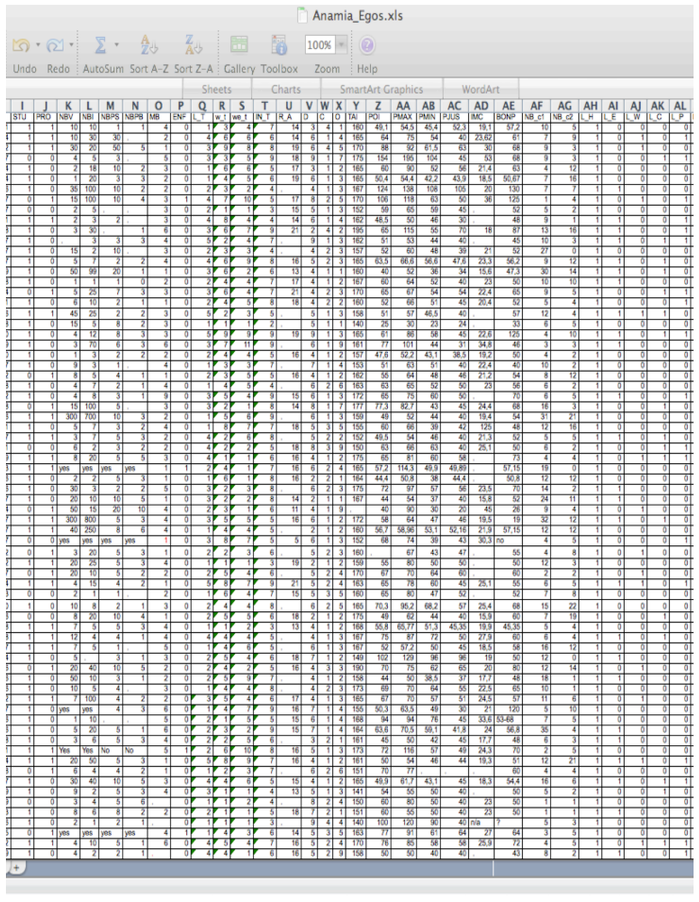
Época. Cuando hemos hecho un ajuste con cada uno de los "trozos" Es



REDES NEURONALES

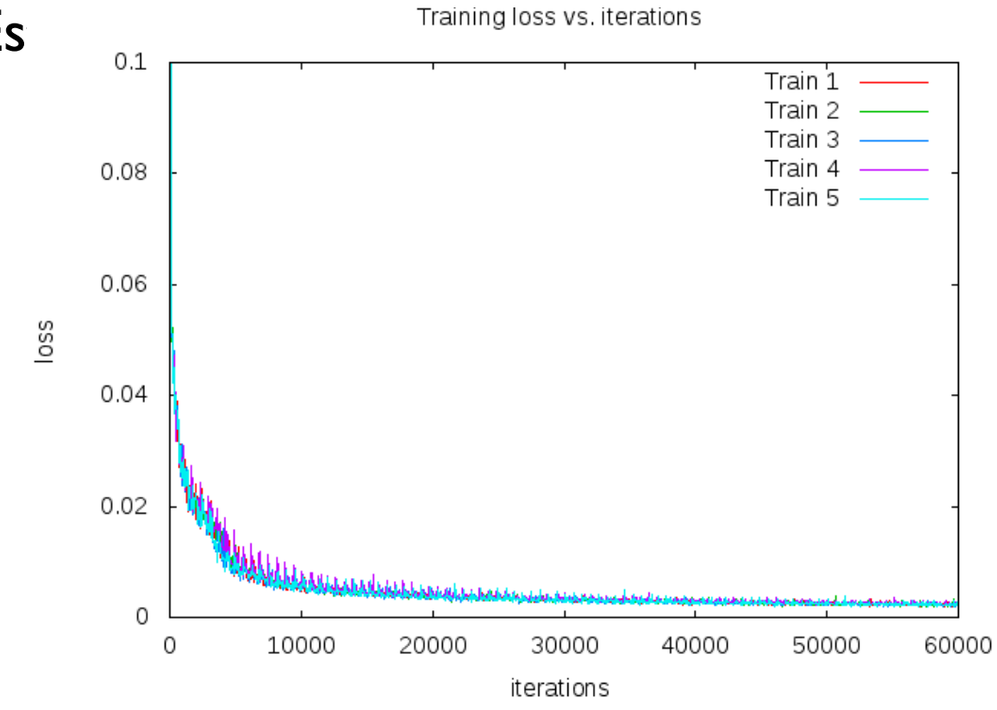
ENTRENAMIENTO

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:



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

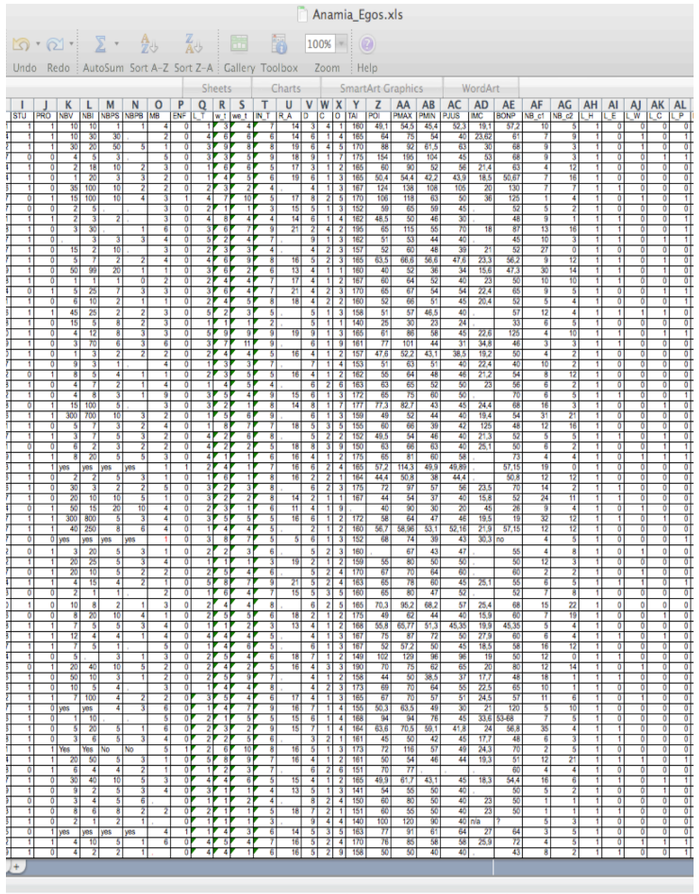
Época. Cuando hemos hecho un ajuste con cada uno de los "trozos" Es



REDES NEURONALES

ENTRENAMIENTO

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



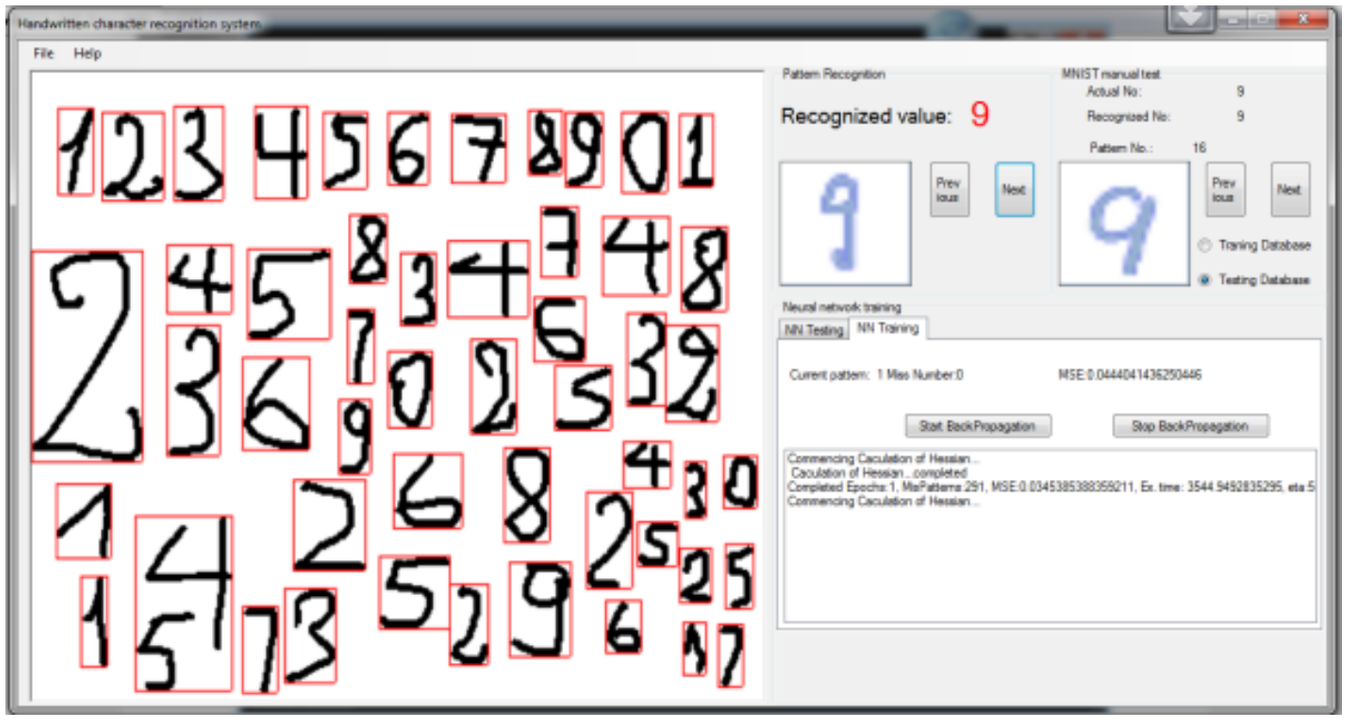
Optimizer: Algoritmo que usaremos para minimizar la función.

Loss function: Dependiendo del problema nos puede interesar distintas funciones de distancia para minimizar

REDES NEURONALES

- Aplicaciones

Reconocimiento de patrones

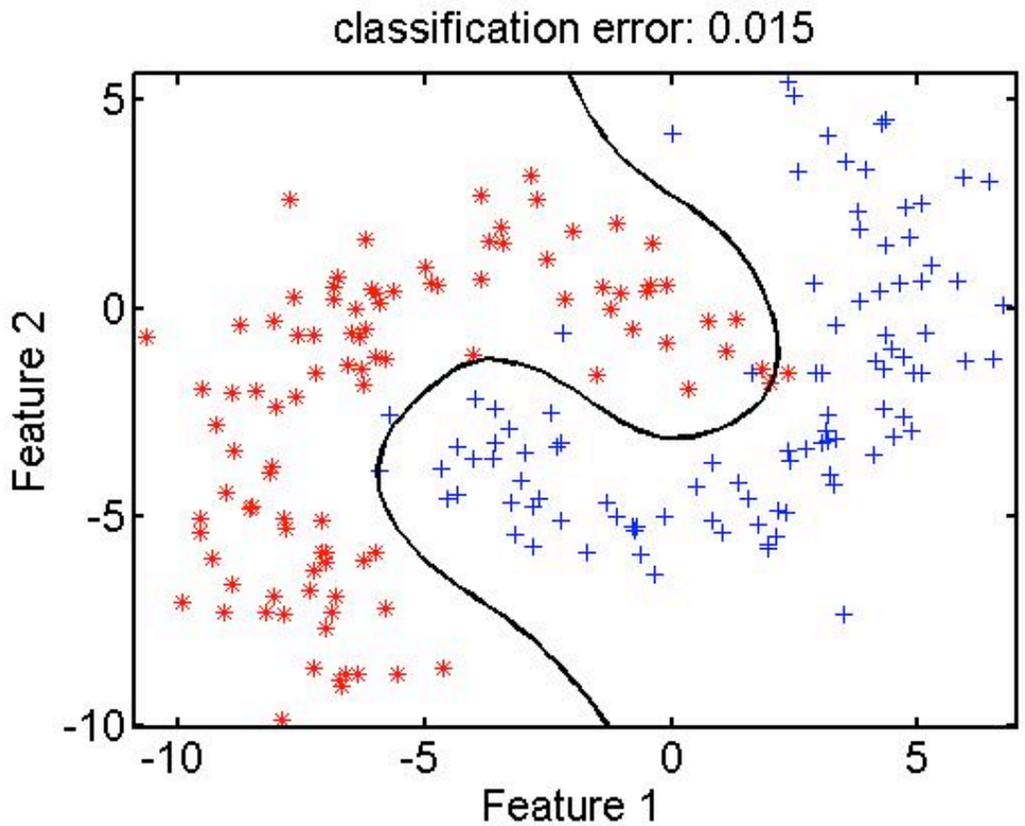


1. Se pasa lenguaje escrito a imagenes de 0 y 1

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...)

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,...)

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

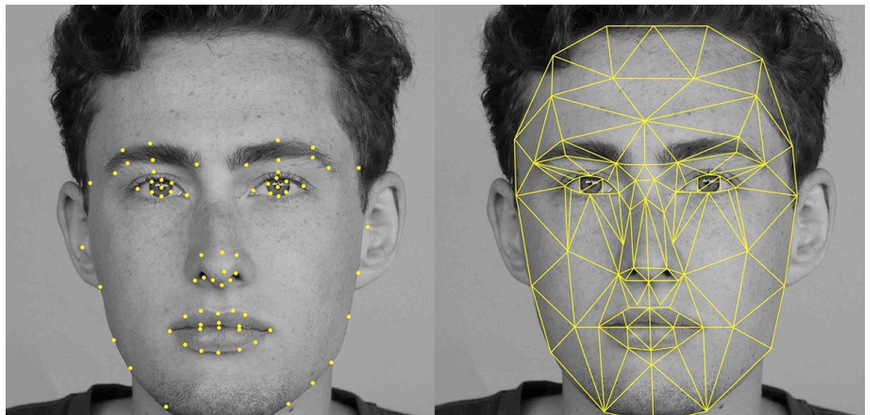
APLICACIONES

Reconocimiento de etiquetas, texto,...

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

Clasificación de Galaxias



Reconocimiento facial

REDES NEURONALES

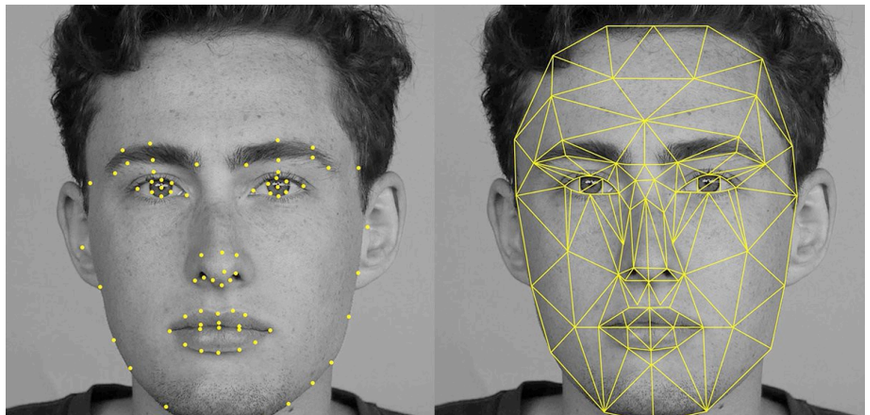
APLICACIONES

Reconocimiento de etiquetas, texto,...

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

Clasificación de Galaxias



Reconocimiento facial

DEEP LEARNING

REDES NEURONALES

- Validación y otras consideraciones

REDES NEURONALES

EVALUACiÓN: SET DE VALIDACIÓN

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

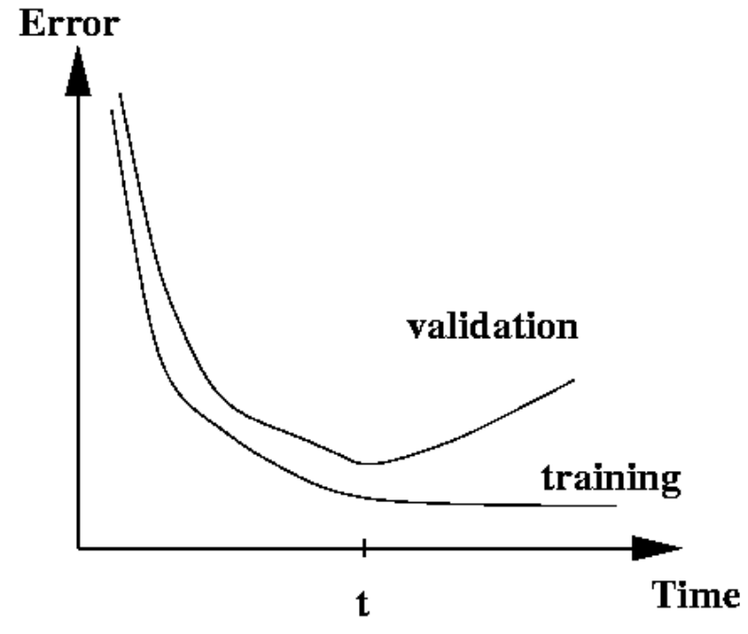
Clasificación *loss function*:

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

Regresión *loss function*:

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

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



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)

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

Completitud (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

$$x_i = \frac{x_i - < x_i >}{\sigma_i}$$

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 dis **estandarizar** los inputs y los outputs para que pe

$$x_i = \frac{x_i - < x_i >}{\sigma_i}$$

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.

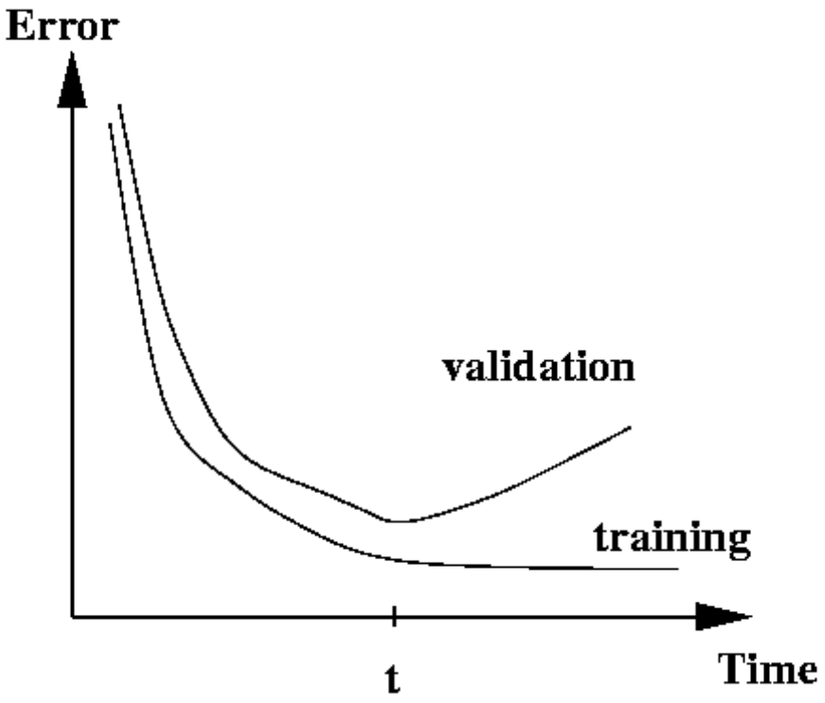
Pista 2. Pocas neuronas pueden hacer que no lleguemos a trazar bien el problema y demasiadas puede ser que sobreajustems (overfitting). Un número para empezar (Ninp+Nout)/2, **pero no es una regla fiable!!**

¡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.

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

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 Leaning (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 clasificacion

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: