Redes Neuronales Artificiales

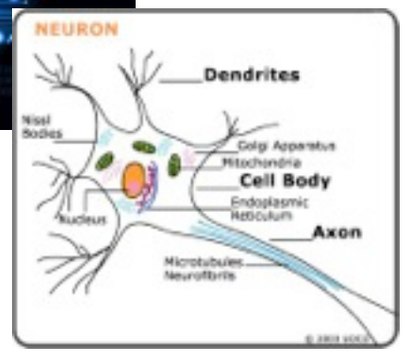
BIG DATA with PTYHON

INTRODUCTION TO
PYTHON FOR BIG DATA

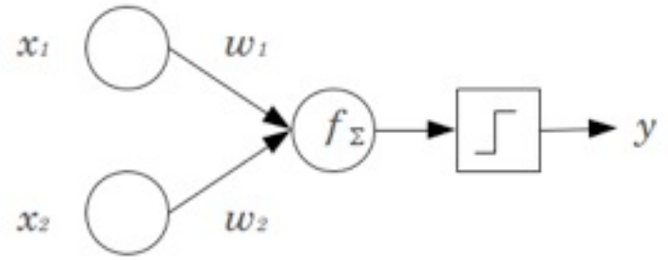
UC UNIVERSIDAD DE CANTABRIA · Grupo 9 · CANTABRIA CAMPUS INTERNACIONAL · GOBIERNO de CANTABRIA

REDES NEURONALES

Herramienta de *machine learning* que nació con dos objetivos:
-modelizar las neuronas biológicas
-la inteligencia artificial (McCullcoch and Pitts1943)



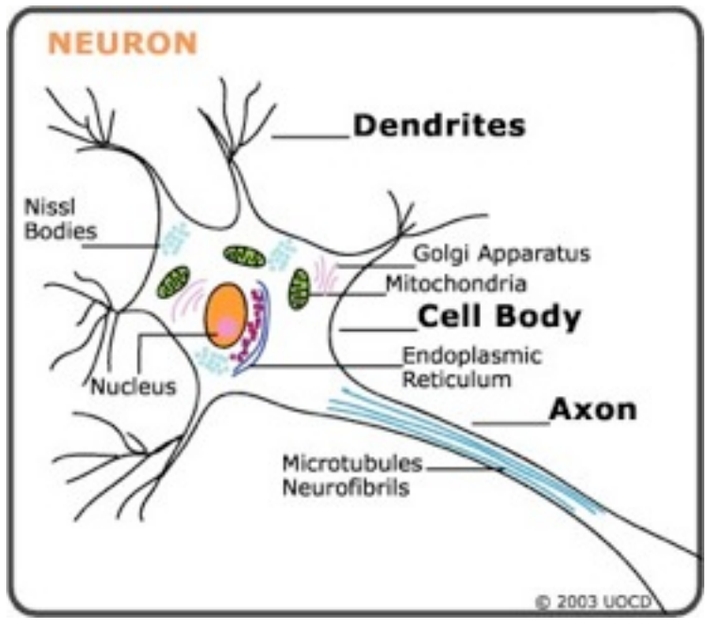
Perceptron. Red neuronal sencilla, sentó las bases de las redes neuronales actuales. **(**Rosenblatt 1958)

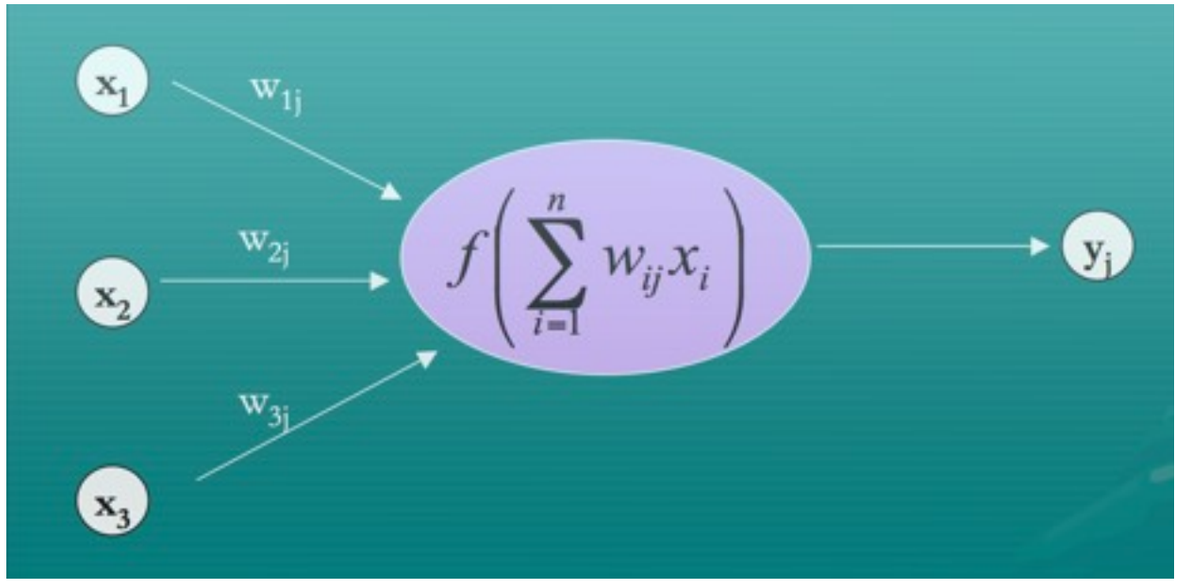


La idea era desarrollar un algoritmo que fuera capaz de aprender, emulando el comportamiento del cerebro humano

Millones de neuronas altamente interconnectadas

REDES NEURONALES





Neurona biológica (simplificación)
La información llega a las neuronas por las **dendritas**

En el **núcleo** de la célula hay el **proceso biológico** que procesa la señal.

Axon: Prolongación de la célula que lleva la información procesada por el núcleo hasta la dendrita de otra neurona

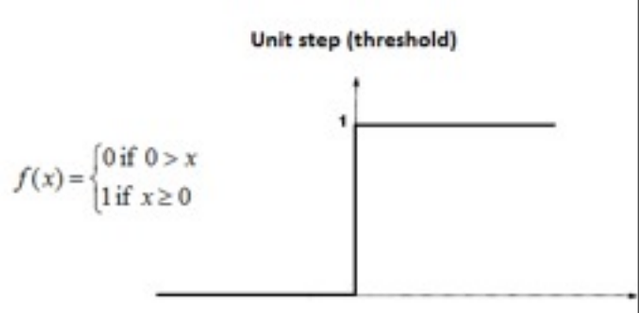
La conexión entre un axon y una dentrita se llama **sinapsis**. La fortaleza o debilidad de esta conexión es crucial para el aprendizaje.

Neurona artificial
Las dendritas vendrían representadas por los **enlaces** de llegada a la neurona.

Los **pesos** w_1,w_2, w_3, definen la fortaleza de la conexión entre una neurona y otra.

El proceso biológico sería la **función de activación** f(x,w) y el axon sería el **enlace** a la siguiente fila de neuronas.

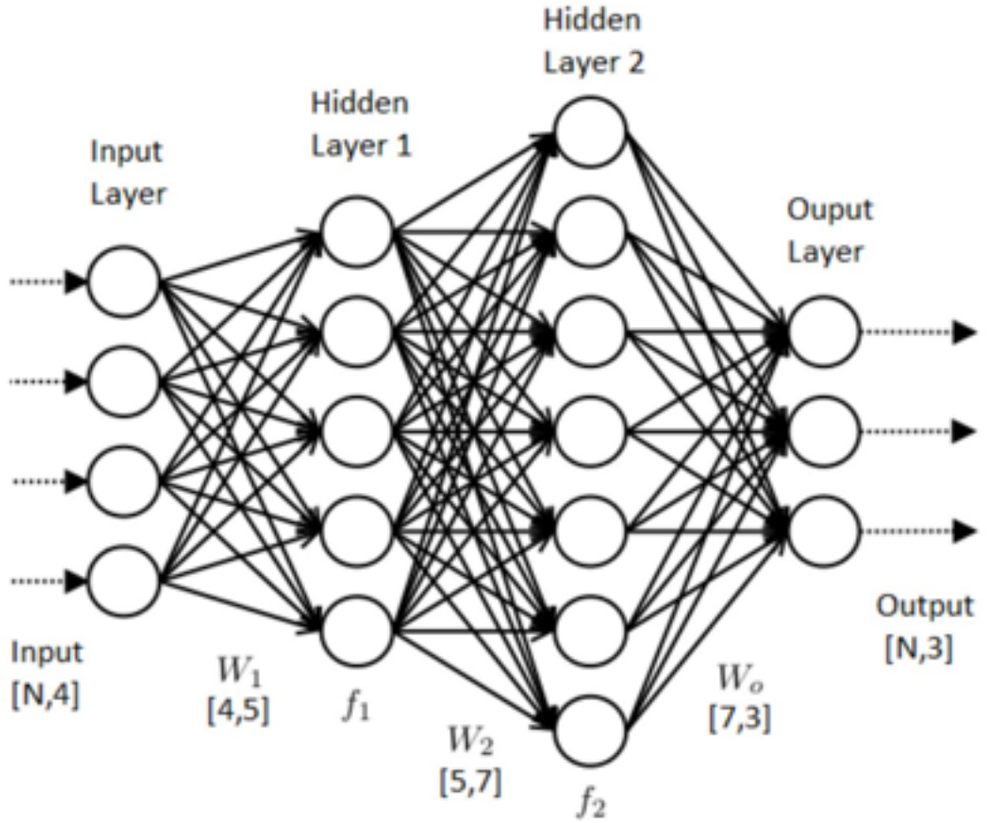
Perceptron: Red neuronal de una sola neurona, donde la función de activación es una función escalón



REDES NEURONALES

Red neuronal artificial

Se organiza, en general, por capas de neuronas, conectadas entre ellas.



Red neuronal biológica

Un cerebro humano tiene alrededor de 10^{11} neuronas, cada una de ellas con unos pocos de miles de conexiones.

Analogía

Los inputs de una red neuronal biológica serían los estímulos que le llegan a través del sistema nervioso. Millones de inputs.

Y los outputs serían la reacción a ese impulso.

Una red neuronal biolígica es muy compleja, la ANN que presentamos aquí tiene sus limitaciones, pero es muy eficiente en algunos problemas.

REDES NEURONALES

TIPOS

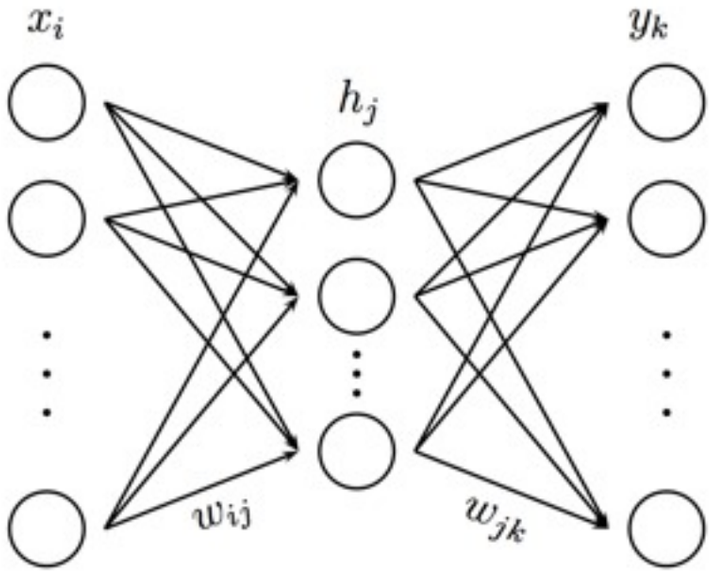
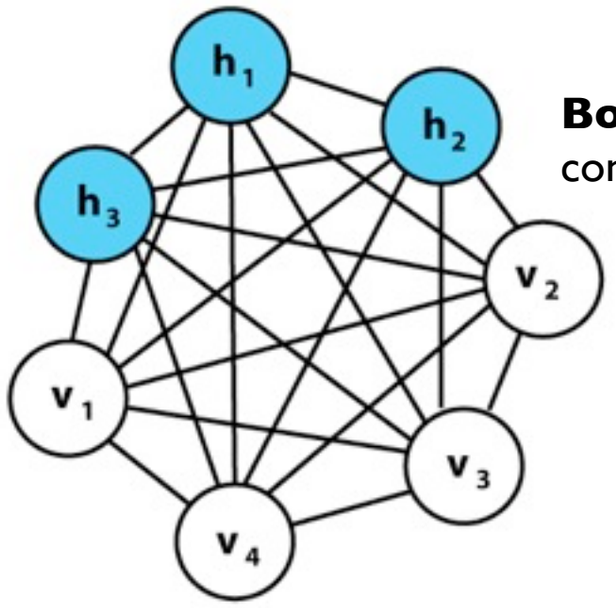


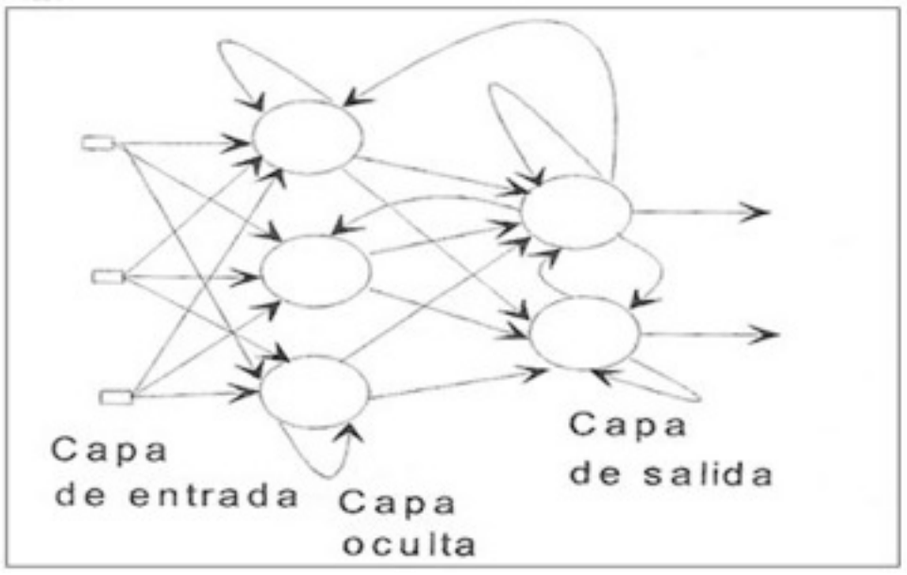
Figure 1. Schematic of a 3-layer feed-forward neural network.

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

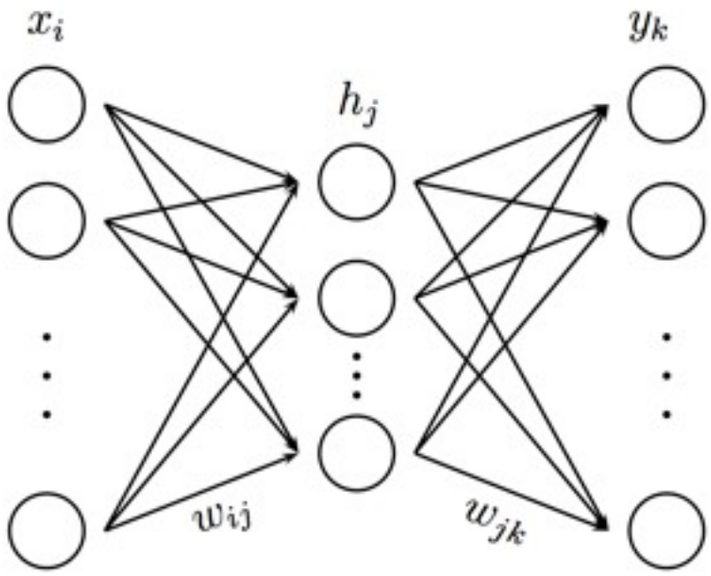
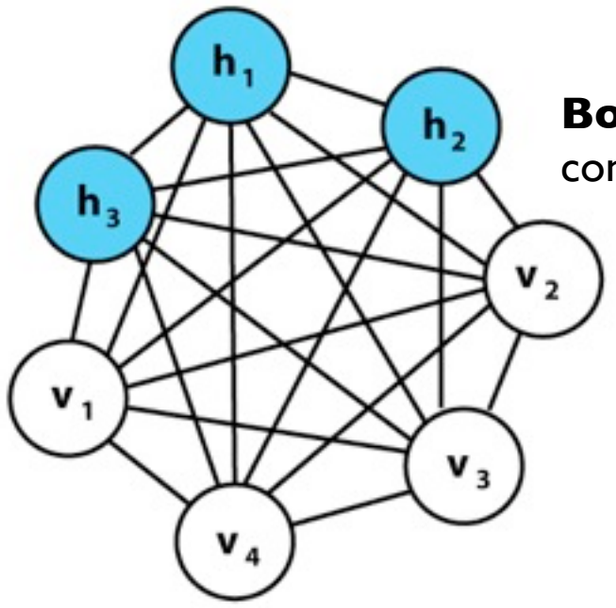


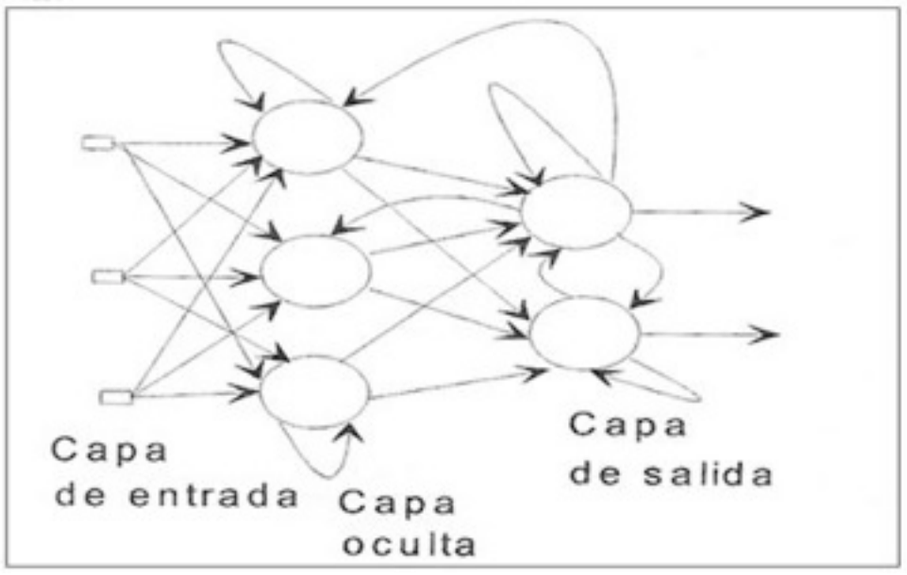
Figure 1. Schematic of a 3-layer feed-forward neural network.

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

NEURONA

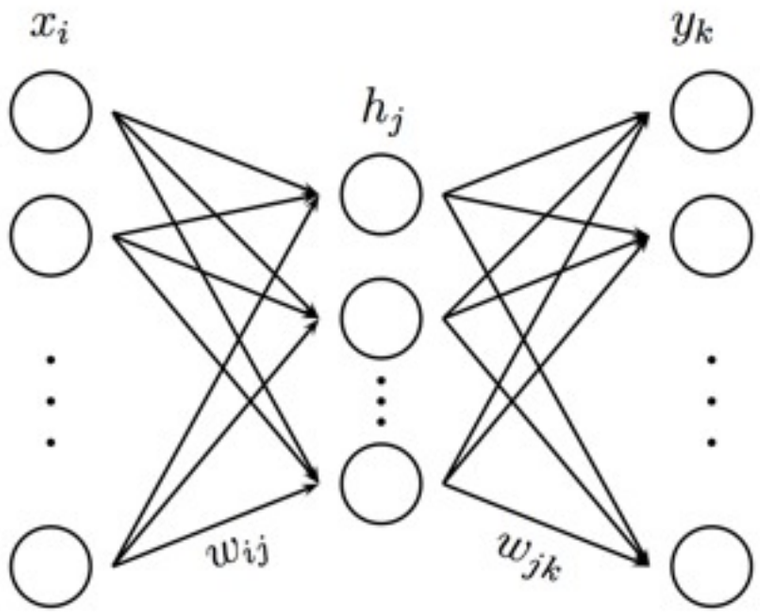


Figure 1. Schematic of a 3-layer feed-forward neural network.

Una red neuronal feed forward, se organiza por capas. Están la capa inicial de **inputs** x, las capas de neuronas (h) que se suelen llamar **capas ocultas**, que tienen una función de activación (para todas la misma), y la capa de **outputs** (y) que tiene en general una función de activación lineal.

En cada neurona se calcula un número, que depende de la función de activación, y de una combinación lineal de los pesos y los inputs que llegan a esta neurona.

$$h_1 = f(w_{11}x_1 + w_{21}x_2 + w_{31}x_3)$$

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

REDES NEURONALES

NEURONA

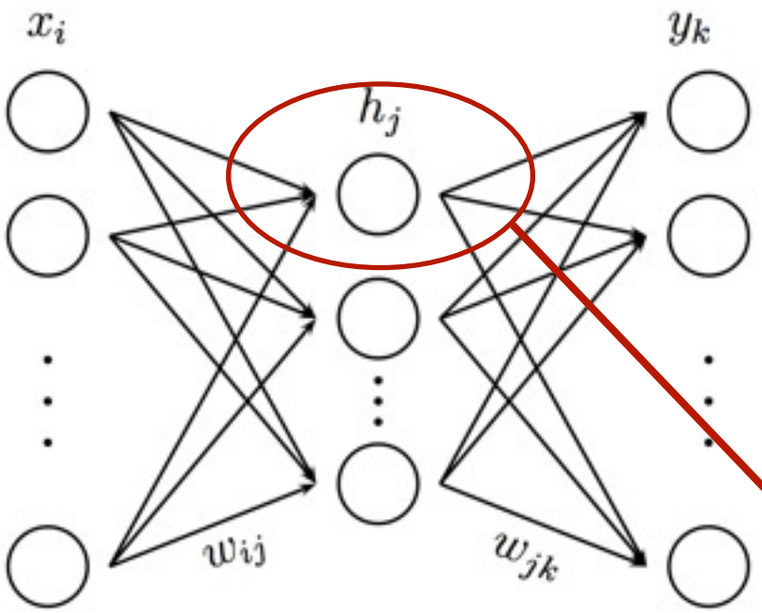


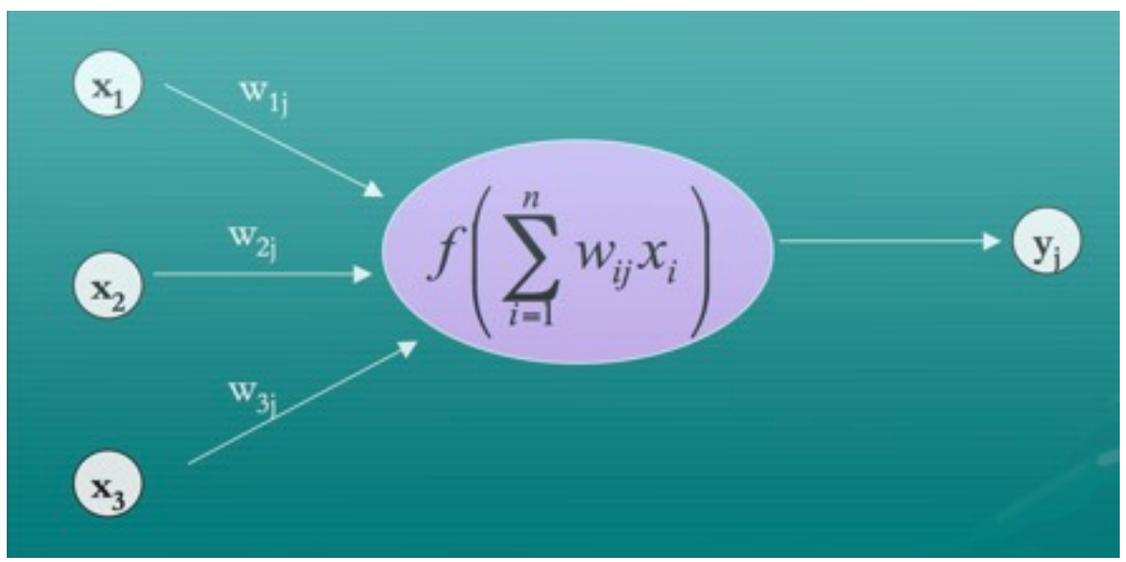
Figure 1. Schematic of a 3-layer feed-forward neural network.

Una red neuronal feed forward, se organiza por capas. Están la capa inicial de **inputs** x, las capas de neuronas (h) que se suelen llamar **capas ocultas**, que tienen una función de activación (para todas la misma), y la capa de **outputs** (y) que tiene en general una función de activación lineal.

En cada neurona se calcula un número, que depende de la función de activación, y de una combinación lineal de los pesos y los inputs que llegan a esta neurona.

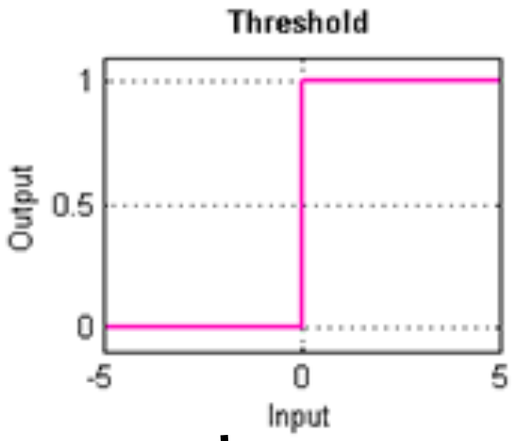
$$h_1 = f(w_{11}x_1 + w_{21}x_2 + w_{31}x_3)$$

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



REDES NEURONALES

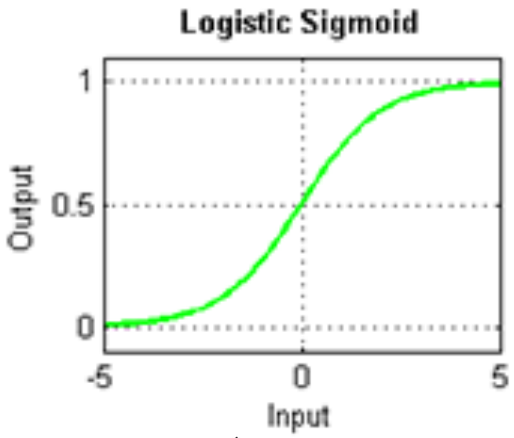
FUNCIONES DE ACTIVACIÓN

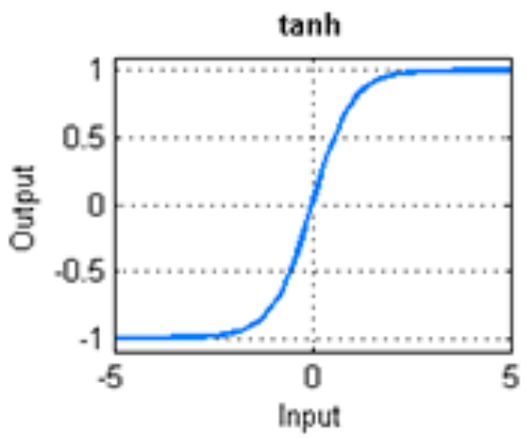


Función de Heaviside o función escalón.

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

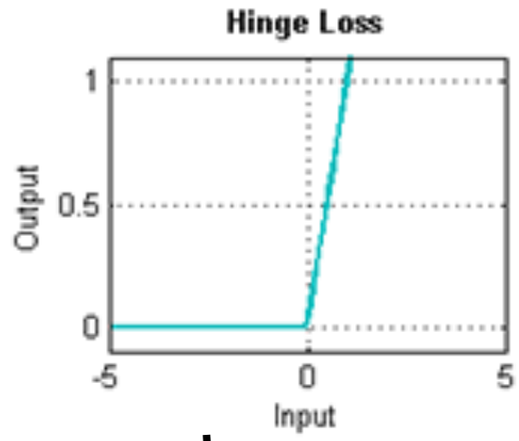
Originalmente se creía que las neuronas biológicas se **activaban o no**. Por eso esta fue la función escogida en los primeros modelos.





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

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



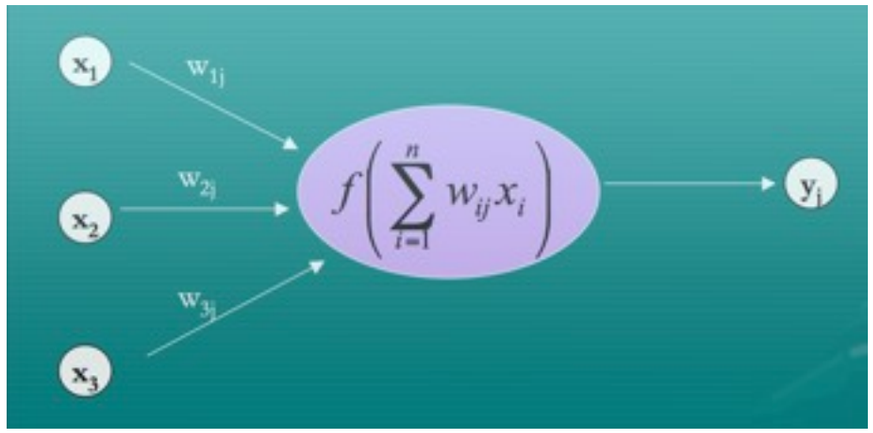
Hinge Loss (or RELU). Se usa en problemas de clasificación, a veces se prefieren versiones más suaves.

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

$$f(x) = x$$

REDES NEURONALES

UNA NEURONA



Ejemplo sencillo:

inputs --> x=(1,1,3)

pesos --> w_{i1}=(0,2,1)

función de activación --> tanh

h=tanh(0+2+3)=0.999

En una red completa de 1 capa:

capa oculta:

$$h_j = \tanh\left(\sum w_{ij}x_i\right)$$

última capa (suponemos f(a) lineal para esta capa):

$$y_k = \sum_j w_{jk}h_j$$

Podemos poner los outpus en función de los inputs y de los pesos →

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

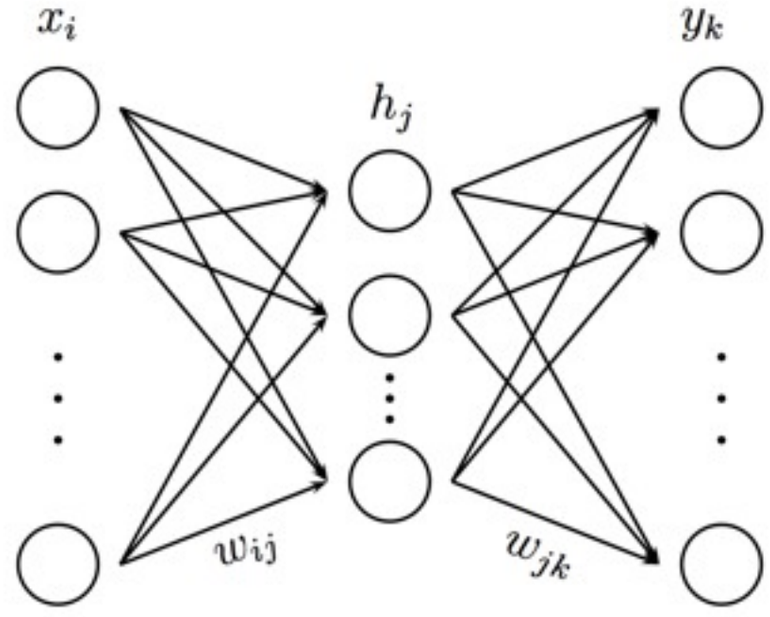


Figure 1. Schematic of a 3-layer feed-forward neural network.

REDES NEURONALES

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

Forma general ⟶

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

Pero ¿como sacamos los pesos? Necesitamos unos datos aquí le llamaremos muestra (o **set) de entrenamiento.**

x_1	x_2	t
0	1	1
10	-2	8
4	12	16
5	1	6
...	...	...

Dados unos inputs, comparamos los valores reales (t) con el valor estimado con la red (**modelo**), con una **función objetivo**, típicamente el error cuadrático y resolvemos con cualquier **algoritmo de optimización** que minimce esa función para encontrar los pesos w

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

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) :

$$E = \sum_n \sum_k \frac{(t_k^n - y_k^n)^2}{K \times N}$$

REDES NEURONALES

¿¿Mucha información??

Parece complicado pero es más sencillo de lo que parece y tiene muchísimas aplicaciones. Si el set de datos es bueno suelen ser muy eficientes.

Resumen:

1. Una red neuronal artifical es un conjunto de neuronas organizadas por capas.

2. La primera capa son los inputs (datos conociodos) y la última los outputs (resultados que queremos)

3. Entre medio hay una o más capas ocultas con una o más neuronas cada una

4. En cada una de las neuronas se hace una combinación lineal de los pesos y los inputs que le llegan y se aplica una función

5. El valor que se obtiene en esa neurona es el que se pasa a la siguiente capa como input

6. Esto nos permite tener una relación entre los outputs y los inputs, donde, dado un set de entrenamiento, la única incógnita son los pesos: $y_k(\vec{x}, \vec{w})$

7. **Minimizamos la diferencia cuadrática** entre los datos conocidos y el modelos y sacamos los pesos (parámetros del modelo) como ayer.

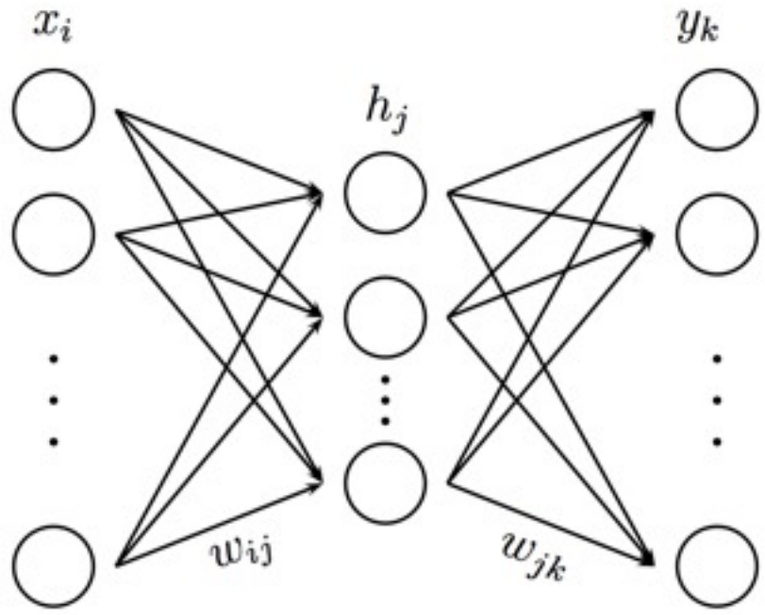


Figure 1. Schematic of a 3-layer feed-forward neural network.

REDES NEURONALES

¿¿Mucha información??

Parece complicado pero es más sencillo de lo que parece y tiene muchísimas aplicaciones. Si el set de datos es bueno suelen ser muy eficientes.

Resumen:

1. Una red neuronal artifical es un conjunto de neuronas organizadas por capas.

2. La primera capa son los inputs (datos conociodos) y la última los outputs (resultados que queremos)

3. Entre medio hay una o más capas ocultas con una o más neuronas cada una

4. En cada una de las neuronas se hace una combinación lineal de los pesos y los inputs que le llegan y se aplica una función

5. El valor que se obtiene en esa neurona es el que se pasa a la siguiente capa como input

6. Esto nos permite tener una relación entre los outputs y los inputs, donde, dado un set de entrenamiento, la única incógnita son los pesos: $y_k(\vec{x}, \vec{w})$

7. **Minimizamos la diferencia cuadrática** entre los datos conocidos y el modelos y sacamos los pesos (parámetros del modelo) como ayer.

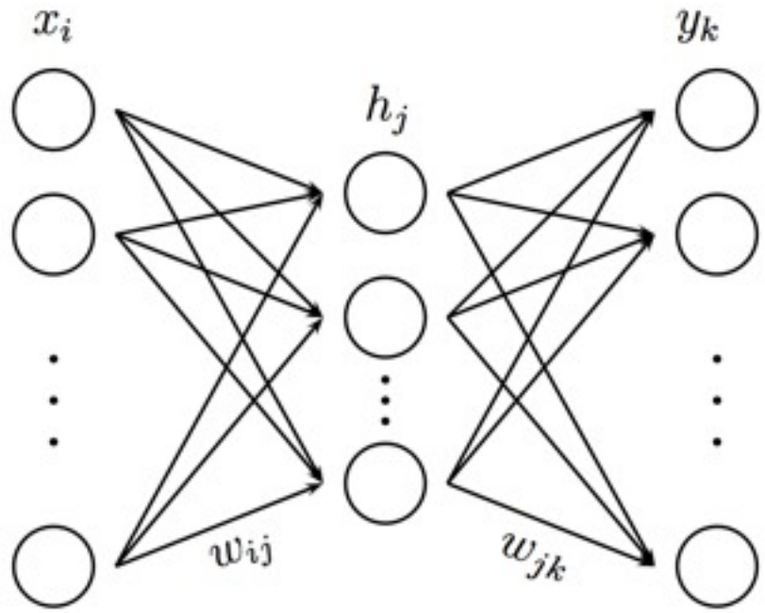


Figure 1. Schematic of a 3-layer feed-forward neural network.

Una vez la red este entrenada y los pesos fijados. Tendremos un modelo que nos perimte obtener el resultado de cualquier nuevo input

REDES NEURONALES

TIPOS DE PROBLEMAS

- Regresión:

Los conocemos previamente. Ej. diagnóstico médico.
Bienestar de una persona a los 60, dados una serie de datos
a los 30. Si el hospital tiene registrado los datos de los pacientes a los 30,
puede a los 60 valorar si bienestar,

nivel colesterol	capacidad respiratoria	presión	sensación bienestar	...	Probabilidad problemas cardiovascul
120	1	2	3		0.90
210	3.4	3	5		0.3
400	12	8	10		0.95
180	1	9	1		0.2
...	...	...	...		0

Generalización. Después de entrenar una red, puedo medir las mismas variables en un paciente nuevo de 30 años y hacer una predicción de la probabilidad de padecer enfermedad cardiovascular a los 60.

Big Data: En la época de la digitalización es sencillo encontrar training sets datos

REDES NEURONALES

TIPOS DE PROBLEMAS

- Clasificación:

Las redes neuronales también se pueden usar para clasificar:
Es parecido a antes, pero ahora los target es a que clase pertence.

color	tamaño	acidez	Clase
0.1	0.9	0.7	1
0.9	0.8	0.3	2
0.3	1	0.4	2
0.5	0.3	0.3	1
...	...	...	

Ejemplo:
Una red que clasifique manzanas y naranjas.
Escogemos caracterísitcas de cada una de ellas: color, tamaño, acidez (algo cuantificable), y a la red le decimos si es o no una manzana, en este caso 1 manzana y 2 es naranja.

Generalización:
Una vez entrenada la red, podremos poner las caracterísitcas de una nueva fruta (tiene que ser o manzana o naranja, ¡no vale darle una pera!) y nos dirá la probabilidad de que sea una naranja o una manzana.

REDES NEURONALES

TIPOS DE PROBLEMAS

- Clasificación:

Las redes neuronales también se pueden usar para clasificar:
Es parecido a antes, pero ahora los target es a que clase pertence.

color	tamaño	acidez	Clase
0.1	0.9	0.7	1
0.9	0.8	0.3	2
0.3	1	0.4	2
0.5	0.3	0.3	1
...	...	...	

Ejemplo:
Una red que clasifique manzanas y naranjas.
Escogemos caracterísitcas de cada una de ellas: color, tamaño, acidez (algo cuantificable), y a la red le decimos si es o no una manzana, en este caso 1 manzana y 2 es naranja.

Si queremos clasificar peras también hay que añadirlas en el set de entrenamiento y añadir la clase 3

REDES NEURONALES

TIPOS DE PROBLEMAS

- Clasificación:

Las redes neuronales también se pueden usar para clasificar:
Es parecido a antes, pero ahora los target es a que clase pertence.

color	tamaño	acidez	Clase
0.1	0.9	0.7	1
0.9	0.8	0.3	2
0.3	1	0.4	2
0.5	0.3	0.3	1
...	...	...	

Ejemplo:
Una red que clasifique manzanas y naranjas.
Escogemos caracterísitcas de cada una de ellas: color, tamaño, acidez (algo cuantificable), y a la red le decimos si es o no una manzana, en este caso 1 manzana y 2 es naranja.

Si queremos clasificar peras también hay que añadirlas en el set de entrenamiento y añadir la clase 3

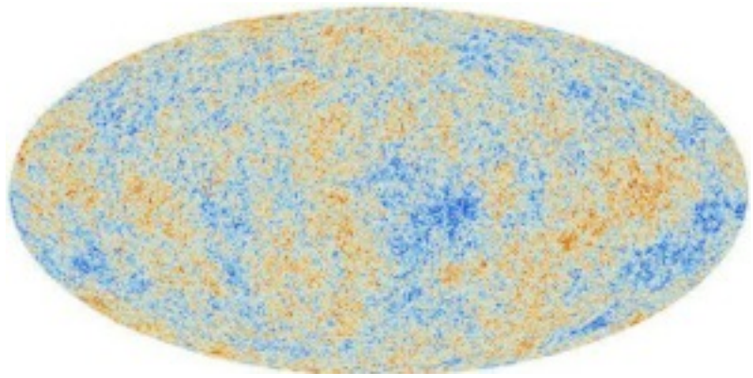
Función de activación en la capa de salida para un problema de clasificación tiene que ir de 0 a 1, en vez de ser lineal o bien se tranforman los outputs para que vayan de 0 a 1. También se suele cambiar la función objetivo por otra que compare probabilidades.

REDES NEURONALES

ENTRENAMIENTO

Y si no tenemos datos? ¿¿De donde sacamos el set de entrenamiento??

- **Simulaciones (si se puede):**
En ciencia muchas veces sabemos el comportamiento, no nos hace falta un modelo predictivo, pero en problemas altamente no lineales, o donde hay cálculos muy costosos, las redes neuronales son muy útiles. Como sabemos el comportamiento lo podemos simular



Ej. sencillo: aprender a sumar

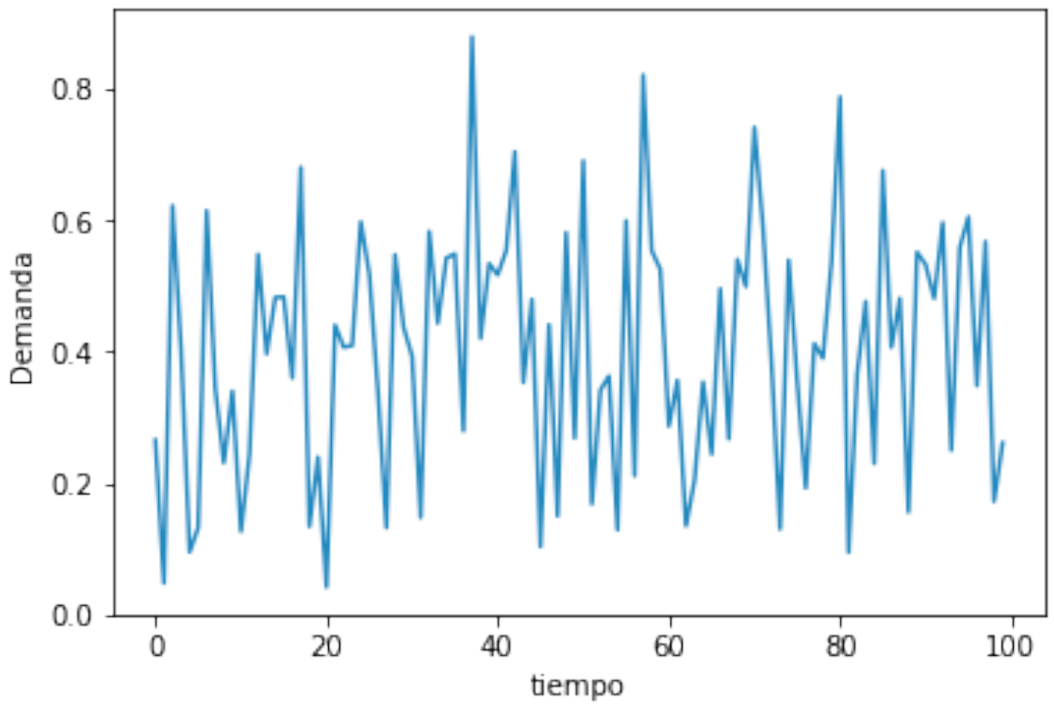
x_1	x_2	t
0	1	1
10	-2	8
4	12	16
5	1	6
...	...	...

Ej. realista: Obtención parámetros cosmológicos (Edad universo, cantidad materia oscura, velocidad de expansión,...)
Simulo una serie de mapas con unos valores de esos parámetros, y entreno la red.

inputs	target
pix mapa1	parametros 1
pix mapa2	parametros 2
pix mapa3	parametros 3
pix mapa4	parametros 4
...	...

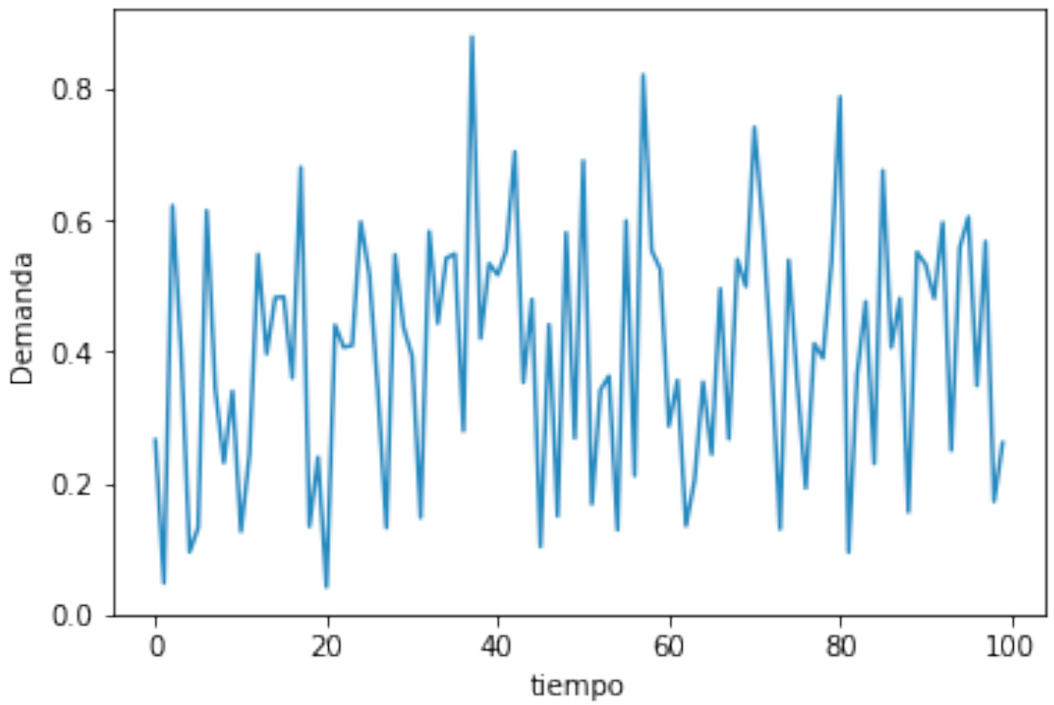
Ejemplo Regresión

Imaginad que tenemos como datos la **demanda** de algún producto con el tiempo



Ejemplo Regresión

Imaginad que tenemos como datos la **demanda** de algún producto con el tiempo

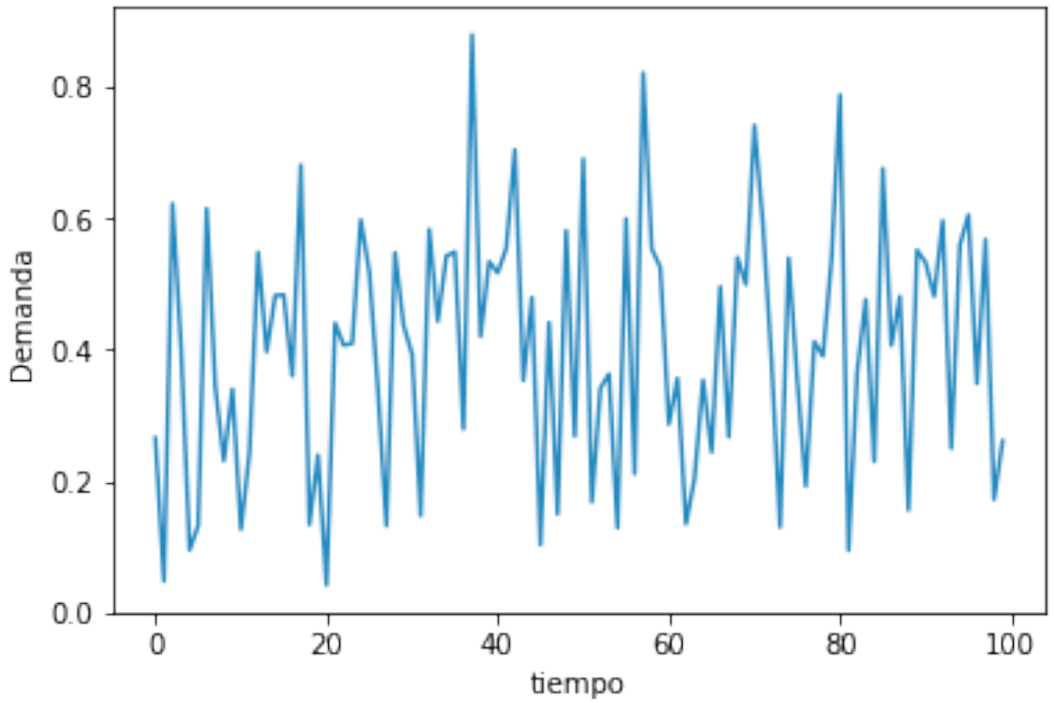


1. Definimos **número de capas, número de neuronas** en cada capa y las **funciones de activación** que vamos a usar y tendremos un "modelo"

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

Ejemplo Regresión

Imaginad que tenemos como datos la **demanda** de algún producto con el tiempo

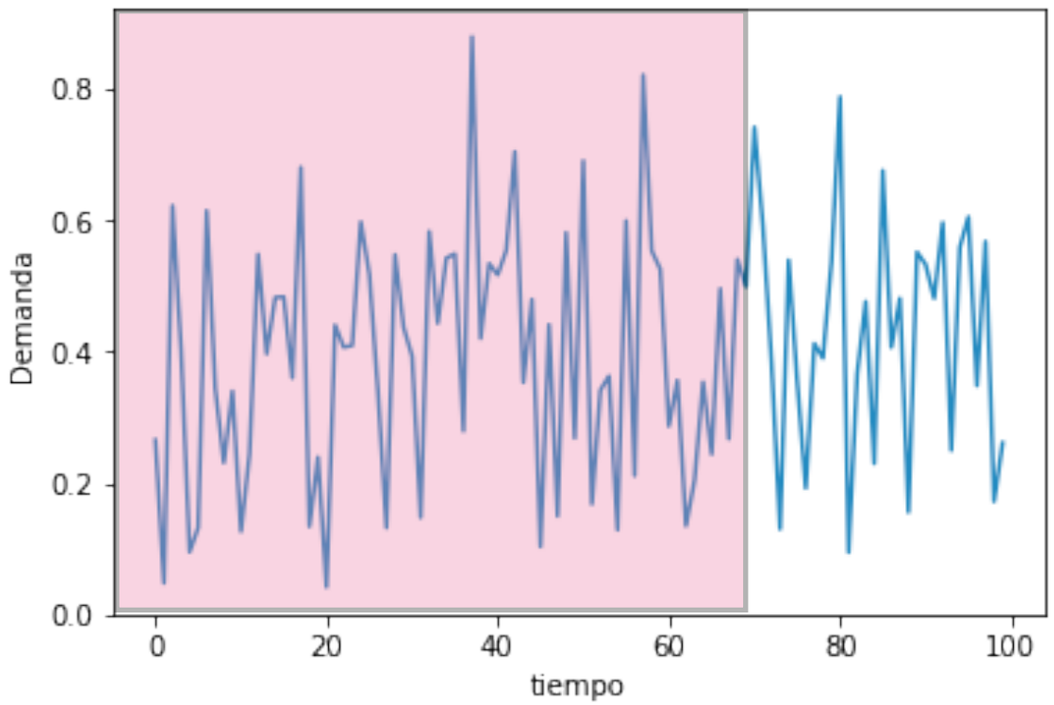


1. Definimos **número de capas, número de neuronas** en cada capa y las **funciones de activación** que vamos a usar y tendremos un "modelo"

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

Ejemplo Regresión

Imaginad que tenemos como datos la **demanda** de algún producto con el tiempo



training

1. Definimos **número de capas, número de neuronas** en cada capa y las **funciones de activación** que vamos a usar y tendremos un "modelo"

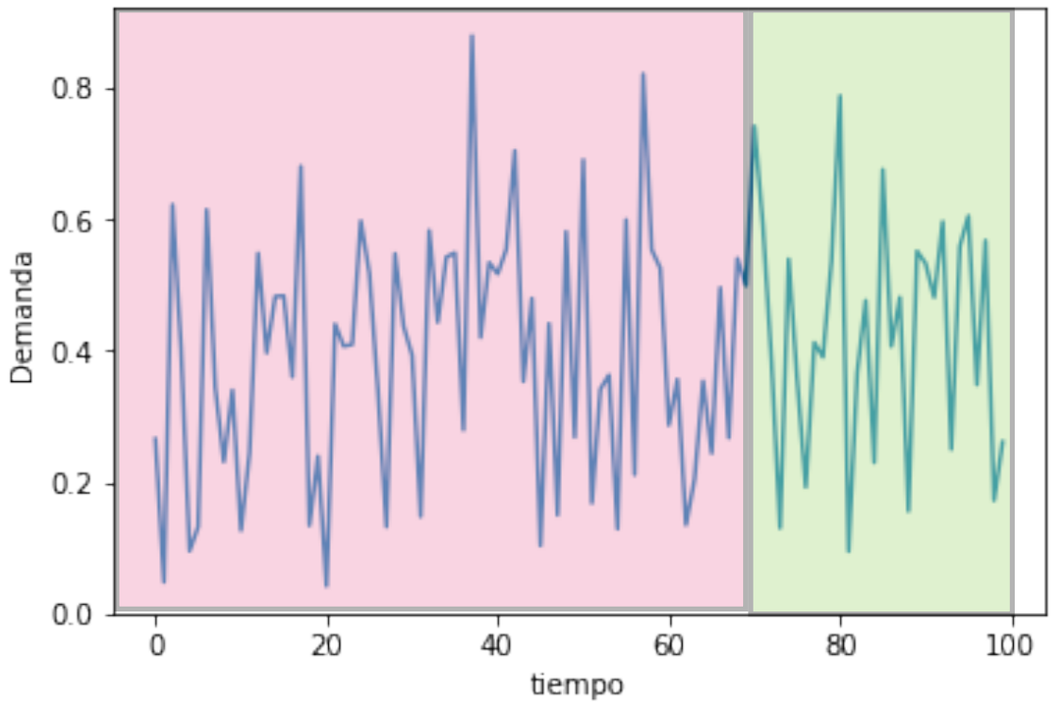
$$y_k = g\Big((\sum_j w_{jk} \dots f(\sum_i w_{ij} x_i)\Big)$$

2. Ajustamos el modelo a una parte de los datos

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

Ejemplo Regresión

Imaginad que tenemos como datos la **demanda** de algún producto con el tiempo



training testing

1. Definimos **número de capas, número de neuronas** en cada capa y las **funciones de activación** que vamos a usar y tendremos un "modelo"

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

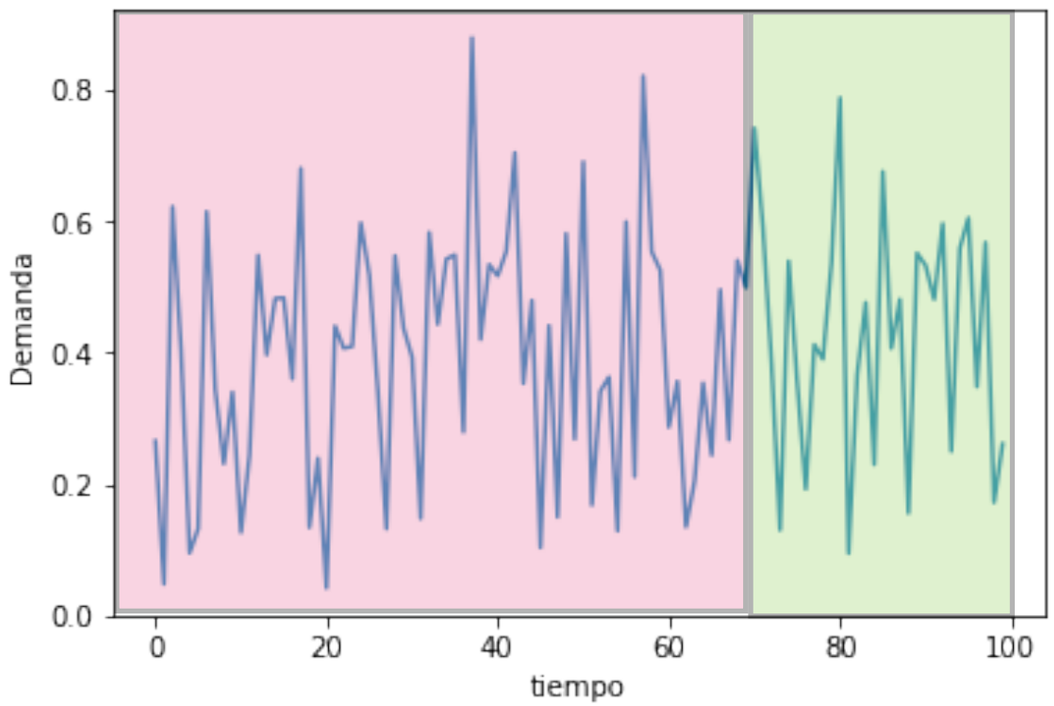
2. Ajustamos el modelo a una parte de los datos

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

3. Calculamos los valores predichos por la red para un set nuevo, pero del que **conocemos** el resultado $y(x_{test})$. (¡Los pesos los hemos fijado antes!)

Ejemplo Regresión

Imaginad que tenemos como datos la **demanda** de algún producto con el tiempo



Una vez nos convencen las predicciones con los datos que tenemos, podemos calcular la demanda en un tiempo futuro (mañana, la semana que viene...)

1. Definimos **número de capas, número de neuronas** en cada capa y las **funciones de activación** que vamos a usar y tendremos un "modelo"

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

2. Ajustamos el modelo a una parte de los datos

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

3. Calculamos los valores predichos por la red para un set nuevo, pero del que **conocemos** el resultado $y(x_{test})$. (¡Los pesos los hemos fijado antes!)

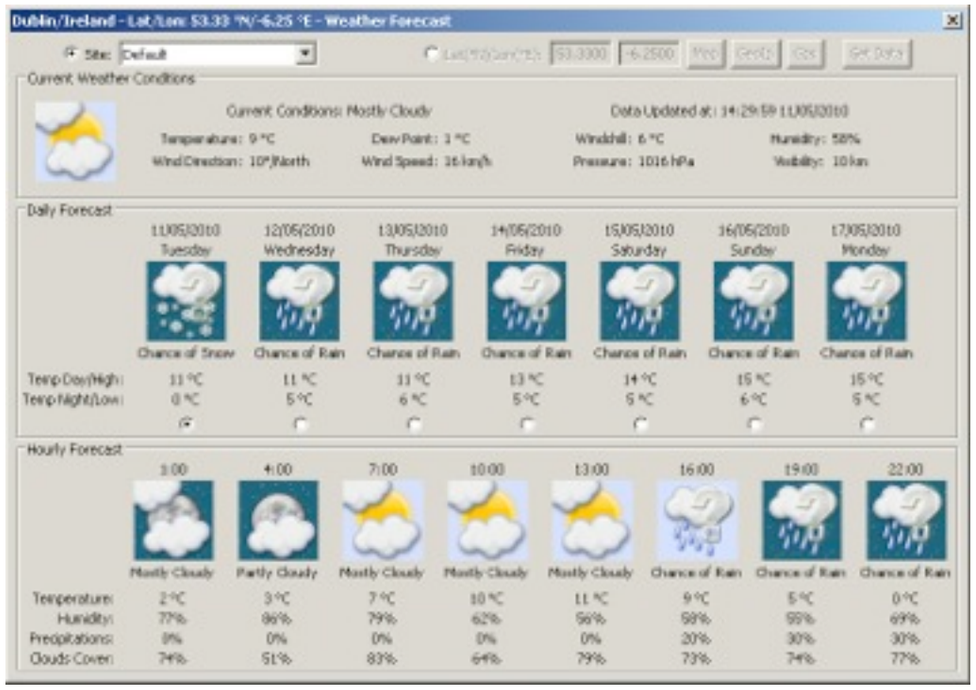
REDES NEURONALES

APLICACIONES

Modelos temporales. Dados una serie temporal de datos predecir lo que pasará en el futuro

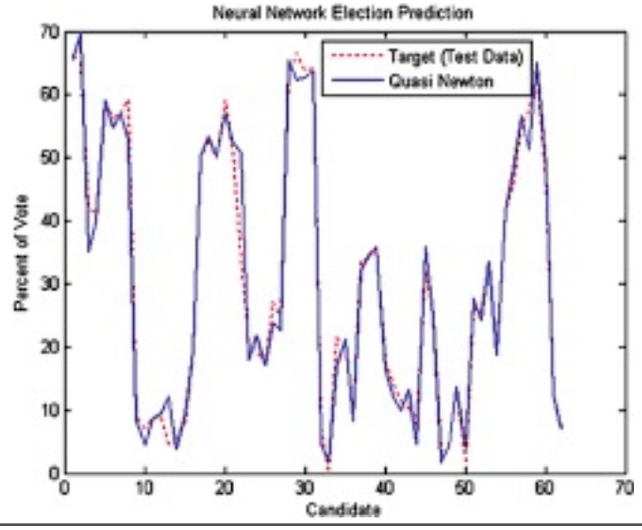
Movimientos Financieros

Predicción meteorológica



Ventas, variación precio de algo,...

Diagnósticos médicos

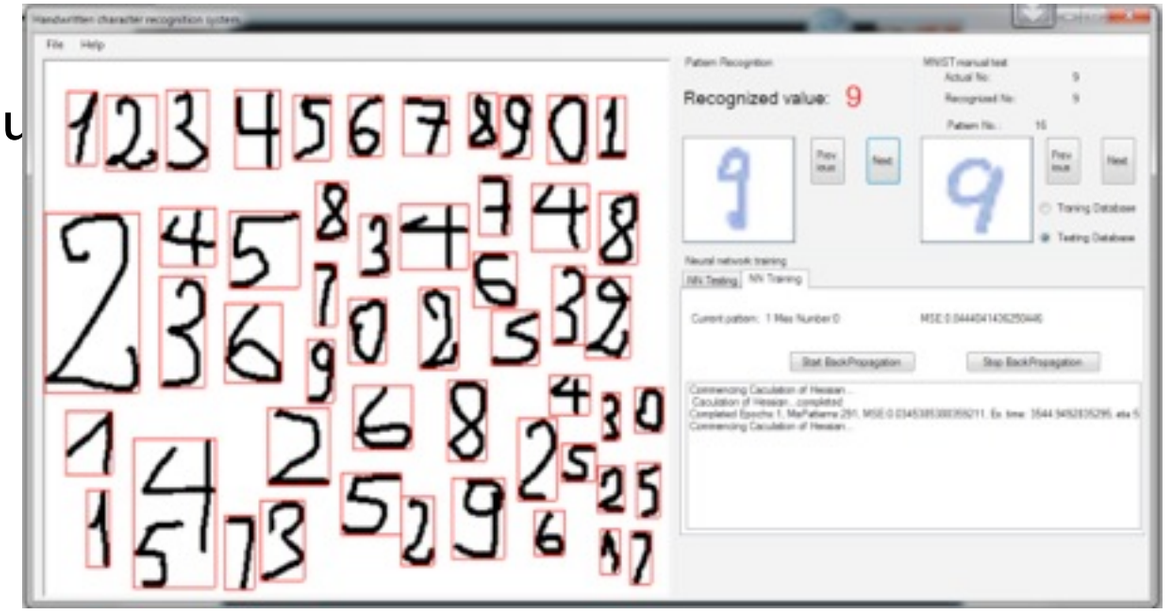


Política, sociología, ...

REDES NEURONALES

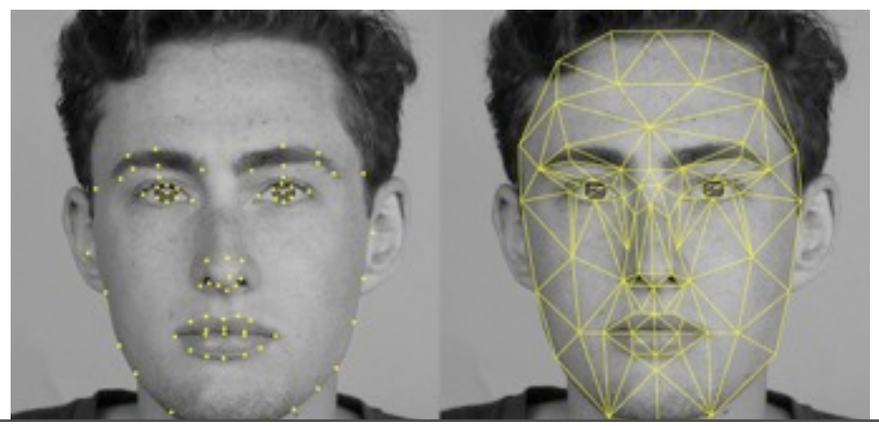
APLICACIONES

Reconocimiento de texto, matrículas,...



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

Identificación de plantas, animales



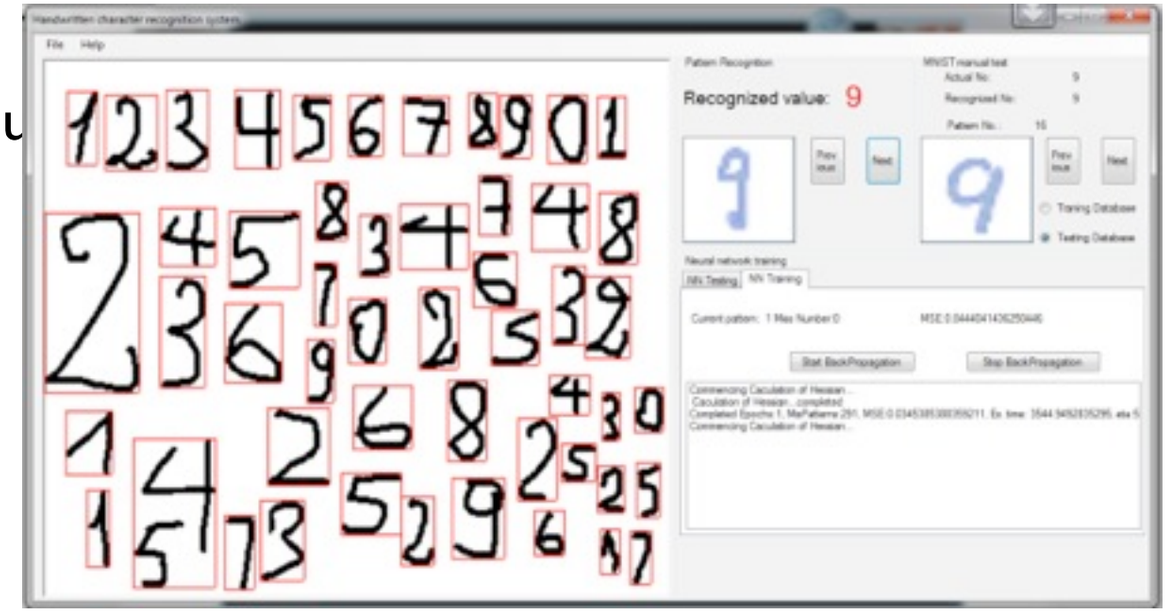
Reconocimiento facial

Clasificación de Galaxias

REDES NEURONALES

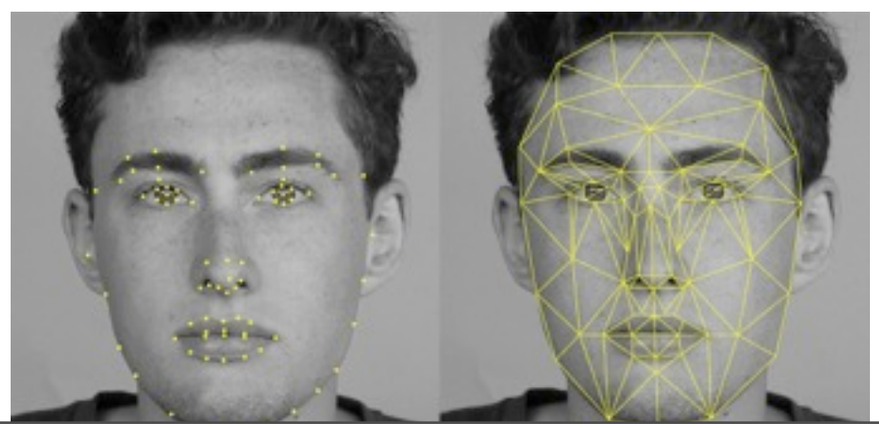
APLICACIONES

Reconocimiento de texto, matrículas,...



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

Identificación de plantas, animales



Reconocimiento facial

Clasificación de Galaxias

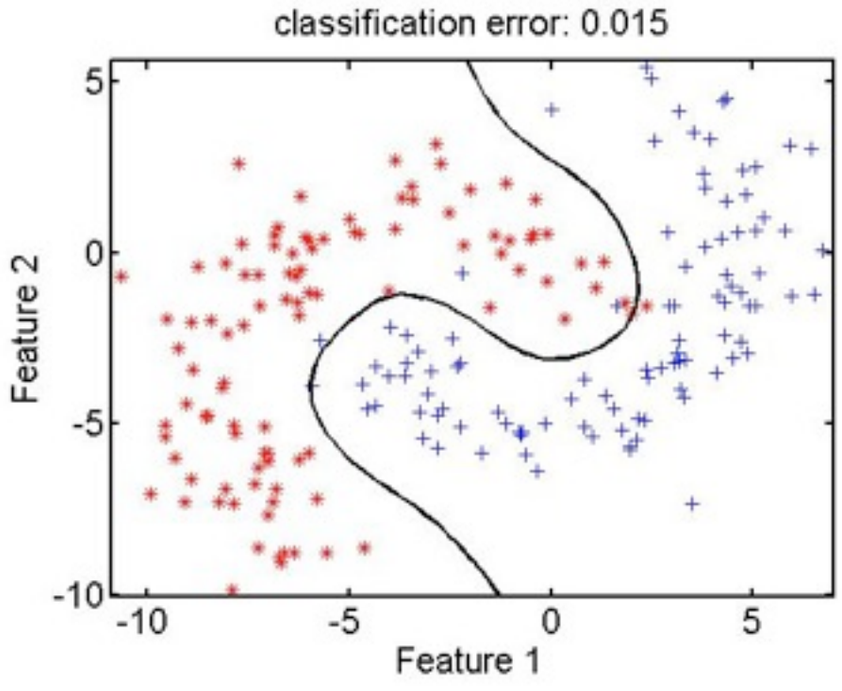
DEEP LEARNING

REDES NEURONALES

APLICACIONES

Problemas Altamente no-Lineales o de muchas dimensiones. A veces aunque sepamos resolver algo de otra manera, es muy lento o complejo y las redes son útils también.

Clasificación no-lineal en multidimensiones



Modelado de Galaxias

Problemas con muchas variables. Ej. evitar inversión de matrices

$$l(\theta|x^o, x^m) = \sum_{i=1}^{n} \left[\frac{n}{2} \log 2\pi + \frac{1}{2} \log |\Sigma| \right.$$
$$- \frac{1}{2}(x_i^o - \mu^o)^T \Sigma^{-1,oo}(x_i^o - \mu^o)$$
$$- (x_i^o - \mu^o)^T \Sigma^{-1,om}(x_i^m - \mu^m)$$
$$\left. - \frac{1}{2}(x_i^m - \mu^m)^T \Sigma^{-1,mm}(x_i^m - \mu^m) \right]. \quad (11)$$

REDES NEURONALES

ENTRENAMIENTO

Una vez tenemos el set de entrenamiento, podemos poner en funcionamento el aprendizaje: Básicamente minimizar la función, para encontrar los pesos. Pero lo hacemos a trozos:

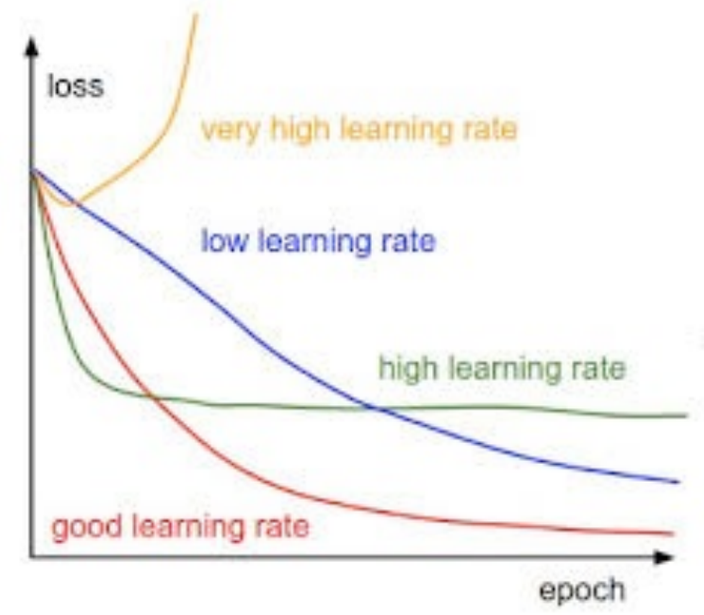
1. Empezamos el entrenamiento con en un putno inicial de pesos (aleatorio) con un pequeño subgrupo del set (**batch size**). Por ejemplo=50. Encontramos los pesos que minimizan la función objetivo. A esto le llamamos **iteración.**

2. Cogemos los 50 siguientes del set del entrenamiento y con estos pesos de punto de partida minimizamos la función objetivo.

3. Esto lo hacemos hasta que hayamos pasado por todo el set de entrenamiento. A la vuelta entera se le llama **época.**

4. Volvemos a empezar, y repetimos tantas veces como épocas le digamos o hasta que converja.

Este proceso nos permite ir evaluando la función durante el entrenamiento



REDES NEURONALES

ARQUITECTURA

- Sabemos seguro cuantos nodos hay en la capa de entrada y la de salida.

pero ...¿Cuantas capas 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!!**

Siempre hay que construir varias redes y probar. Primero se empieza con configuraciones sencillas y se va aumentando la complejidad, primero añadir neuronas, luego añadir capas. (aunque más capas no implica más pesos!)

REDES NEURONALES

GENERALIZACIÓN

¿Tamaño mínimo del set de entrenamiento?

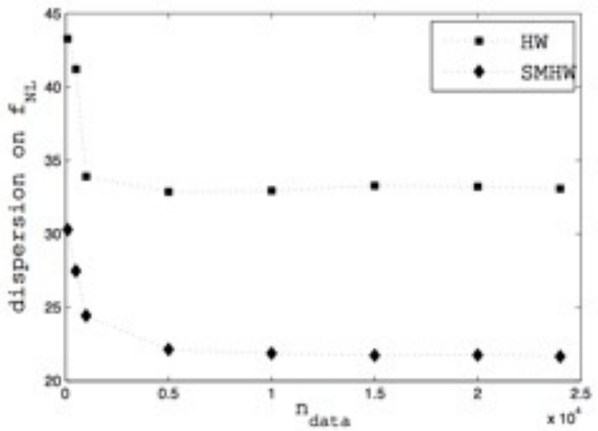
El tamaño del set de entrenamiento tiene que ser lo suficientemente grande para trazar todo el rango que nos interesa y se superior al número de parámetros de la red.

El número de parámetros que queremos estimar, (los pesos, mediante mínimos cuadrados) es en general alto.

número de pesos = $n_{in} \times n_{hid1} + n_{hid1} \times n_{hid2} + \ldots + n_{hidn} \times n_{out}$

Se deben hacer pruebas para ver cuando ha convergido, en general cuando se llega al resultado ótpimo agrandar el set de entrenamiento no mejora el resultado.

número de pesos = $n_{in} \times n_{hid1} + n_{hid1} \times n_{hid2} + \ldots + n_{hidn} \times n_{out}$



y ...¿función de activación y objetivo?

En general, en un problema de **regresión**, tanh para capas ocultas y lineal para los outputs y función objetivo error cuadrático.

Para **clasificación,** se usa más relu o y funciones como **cross-entropy** para la función objetivo.

Aquí haremos tests con las que tenga sklearn por default.

REDES NEURONALES

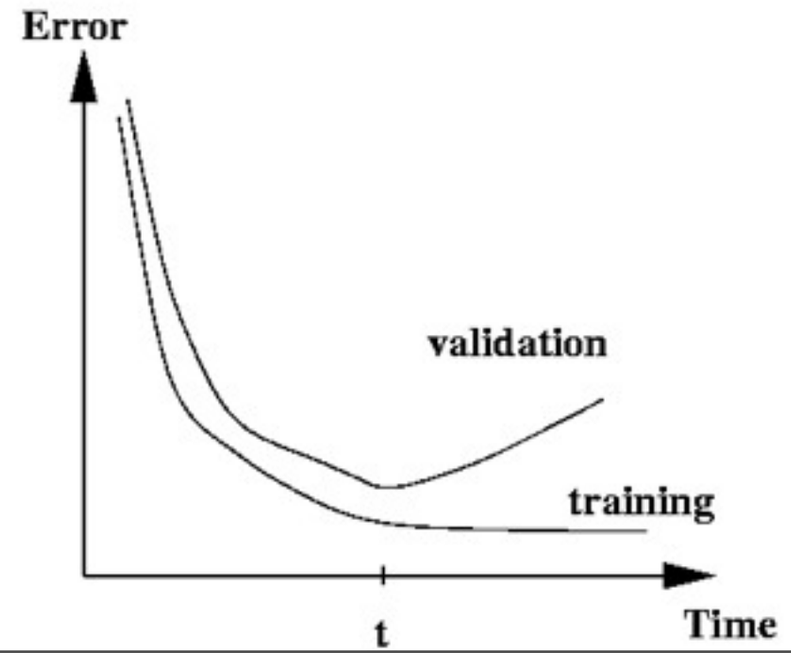
ARQUITECTURA

- Una vez decidimos las caracterísitcas de la red (número de capas y número de neuronas, función de activación, función objetivo) la entrenamos...

y ¿Como sabemos que lo está haciendo bien?

- Vamos viendo el resultado con algún estadístico:
  - regresión: residuos mínimos, coeficiente de correlación
  - clasifiación: TPR (true positive rate=numero de clasificados bien / total)

- Con un set independiente de datos al que llamamos **testing set**, que va calculando los mismos estadísticos a la vez. Importante para detectar overfitting



REDES NEURONALES

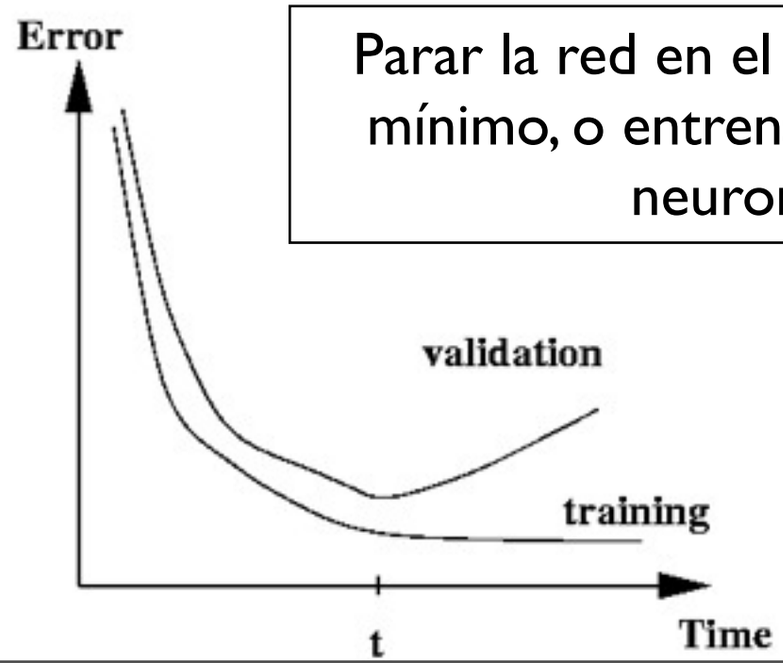
ARQUITECTURA

- Una vez decidimos las caracterísitcas de la red (número de capas y número de neuronas, función de activación, función objetivo) la entrenamos...

y ¿Como sabemos que lo está haciendo bien?

- Vamos viendo el resultado con algún estadístico:
  - regresión: residuos mínimos, coeficiente de correlación
  - clasifiación: TPR (true positive rate=numero de clasificados bien / total)

- Con un set independiente de datos al que llamamos **testing set**, que va calculando los mismos estadísticos a la vez. Importante para detectar overfitting



Parar la red en el punto del test mínimo, o entrenar con menos neuronas

REDES NEURONALES

GENERALIZACIÓN

- Una vez hemos entrenado la red satisfactoriamente, tenemos una función analítica "sencilla" y(x). Podemos coger cualquier nuevo input x y obtendremos el resultado mediante la función analítica y(x).

Esto es una de las grandes ventajas de las NN, que una vez entrenado, sacar el resultado para un nuevo objeto es inmediato.

Pero los nuevos valores tienen que estar representado por el test de entrenamiento.

set entrenamiento

nuevo valor →

x_1	x_2	t
0	1	1
10	-2	8
4	12	16
5	1	6
-20000	120399	???

REDES NEURONALES

FUTURO

- Las redes neuronales están siendo usadas en muchas aplicaciones y se van a seguir usando cada vez más
- con DEEP Leaning (redes neuronales convolucionales) se obtienen resultados sorprendentemente buenos de clasificación, reconocimiento de imágenes, detección de objetos.
- Con la digitalización y la tecnología Big Data se dan las condiciones perfectas para sacar el máximo provecho de estas herramientas.

¿Qué nuevas aplicaciones veremos?
¿Qué cosas podrá predecir? ¿Hasta qué punto será importante en nuestras vidas?

REDES NEURONALES

LAB

Programar una red neuronal no es complicado (con tiempo, se puede programar).

Pero ya hay varios paquetes de python que lo hacen muy fácil. Os muestro un ejemplo con sklearn para luego hacer ejercicios.

También hay Pylearn2, Keras, y más , miradlos

REDES NEURONALES

LAB

Ejercicio 2

- El ejemplo que hemos visto está en NNlab.ipynb, y los datos en el repositorio que os pasé
- Probad varias configuraciones (número neuronas, número de capas ocultas, funciones de activación y ver que funciona mejor)
- Haced el mismo ejercicio pero usando un set de datos real para clasificar vinos según su cultivador, (wine.data, se tiene que leer con pandas, read_csv) (datos reales sacados de https://archive.ics.uci.edu/)

Ejercicio 3

- Entrenad una red para que aprenda a multiplicar 4 números (tenéis ejemplo para que sume 3 números en NNExampleRegression.ipynb)