REDES NEURONALES ARTIFICIALES (ANN)

Sergi Ramírez

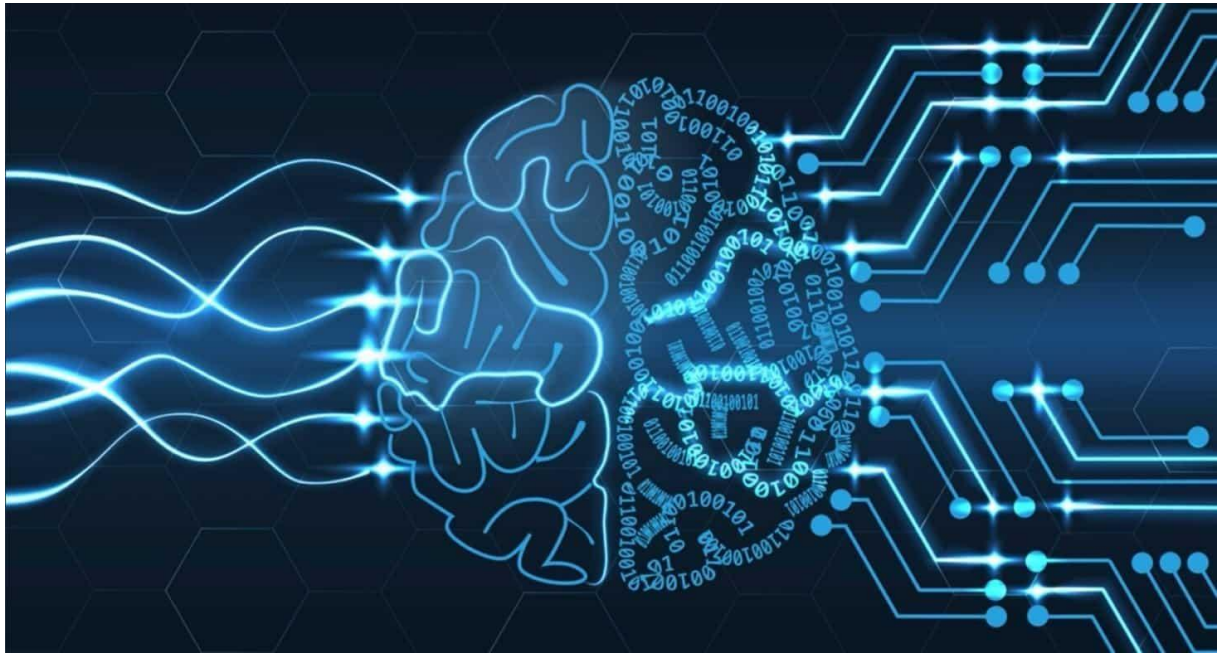
12 de Diciembre de 2024

Universidad de Barcelona - Universidad Politécnica de Cataluña

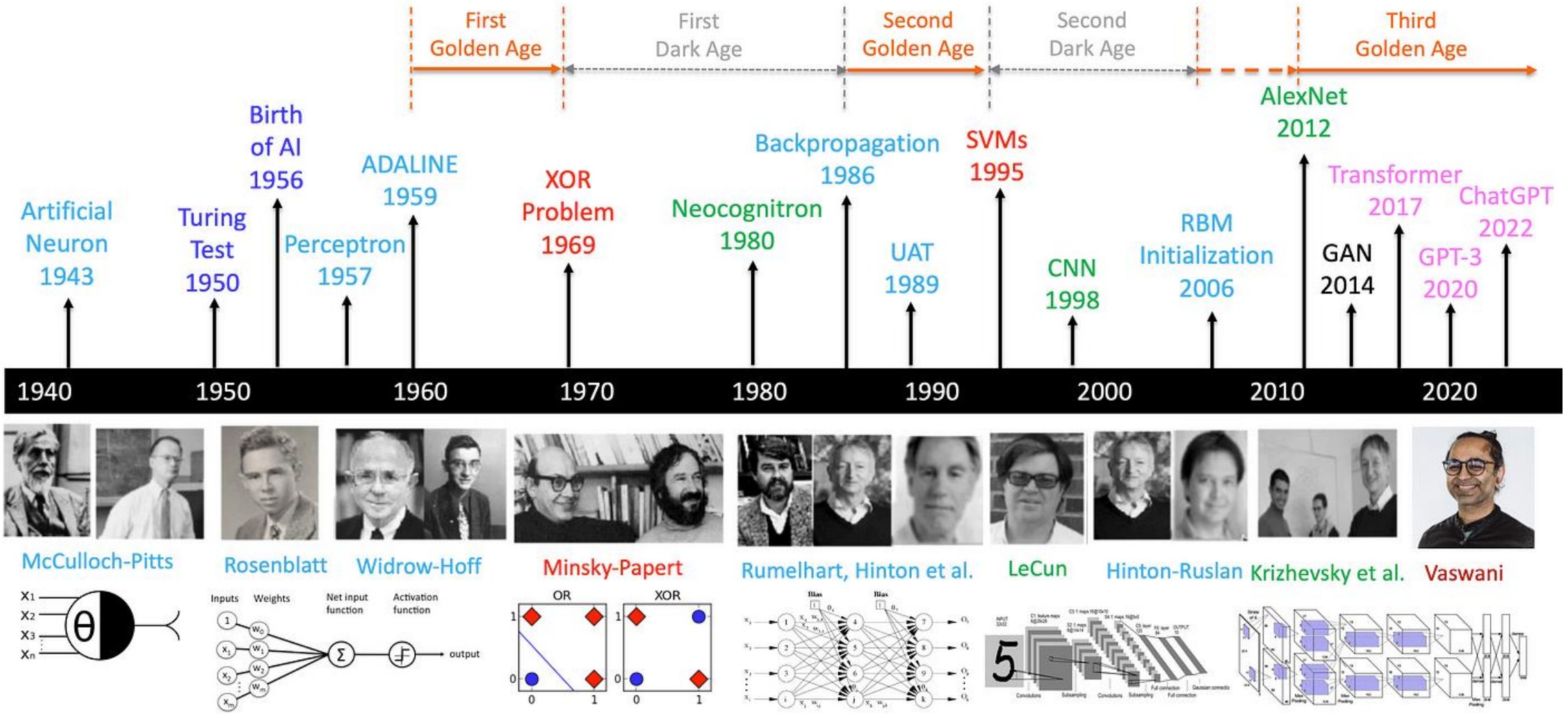
UNIVERSITAT DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Índice

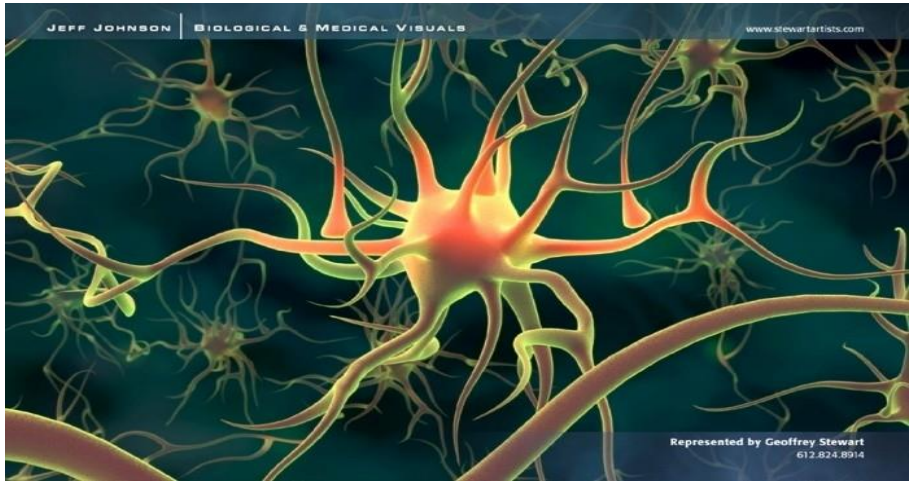


Un poco de historia

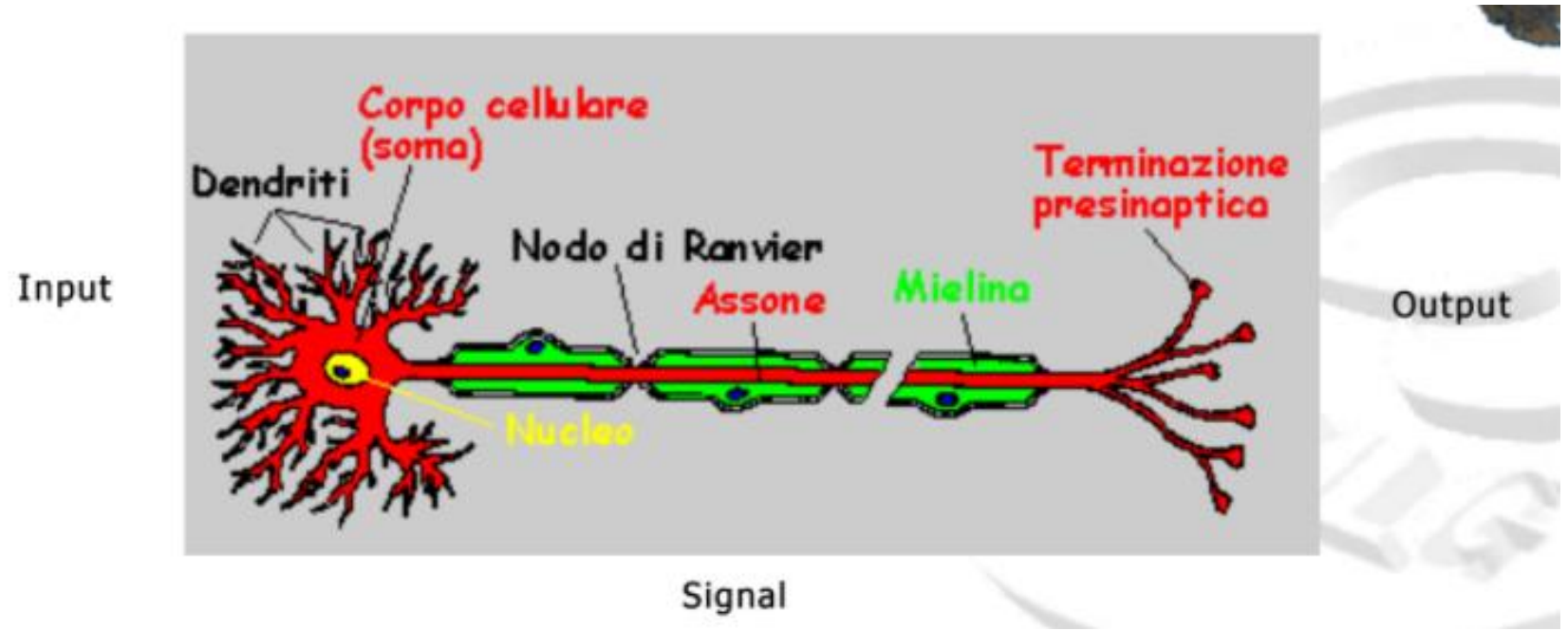


El modelo biológico

El cerebro humano contiene 10 billones de neuronas aproximadamente. Cada una está conectada con otras 10.000 neuronas.



El modelo biológico



Neuronas, dendritas y axones.

- Paralelismo
- Tolerancia a los errores

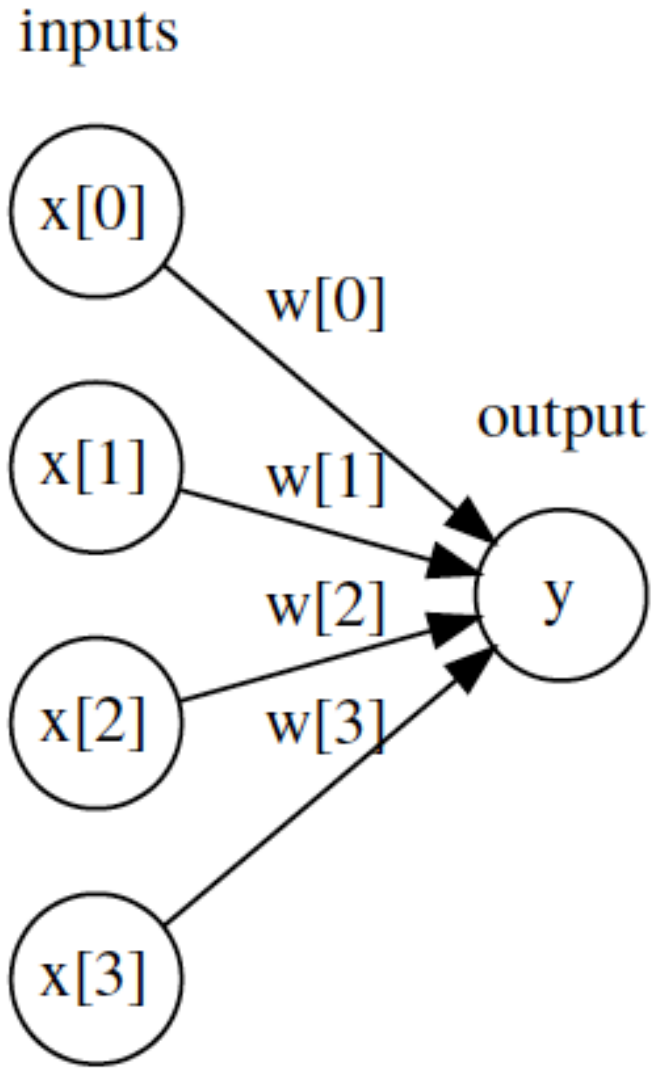
La señal se transmite a través de las neuronas

Artificial Neural Networks (ANN)

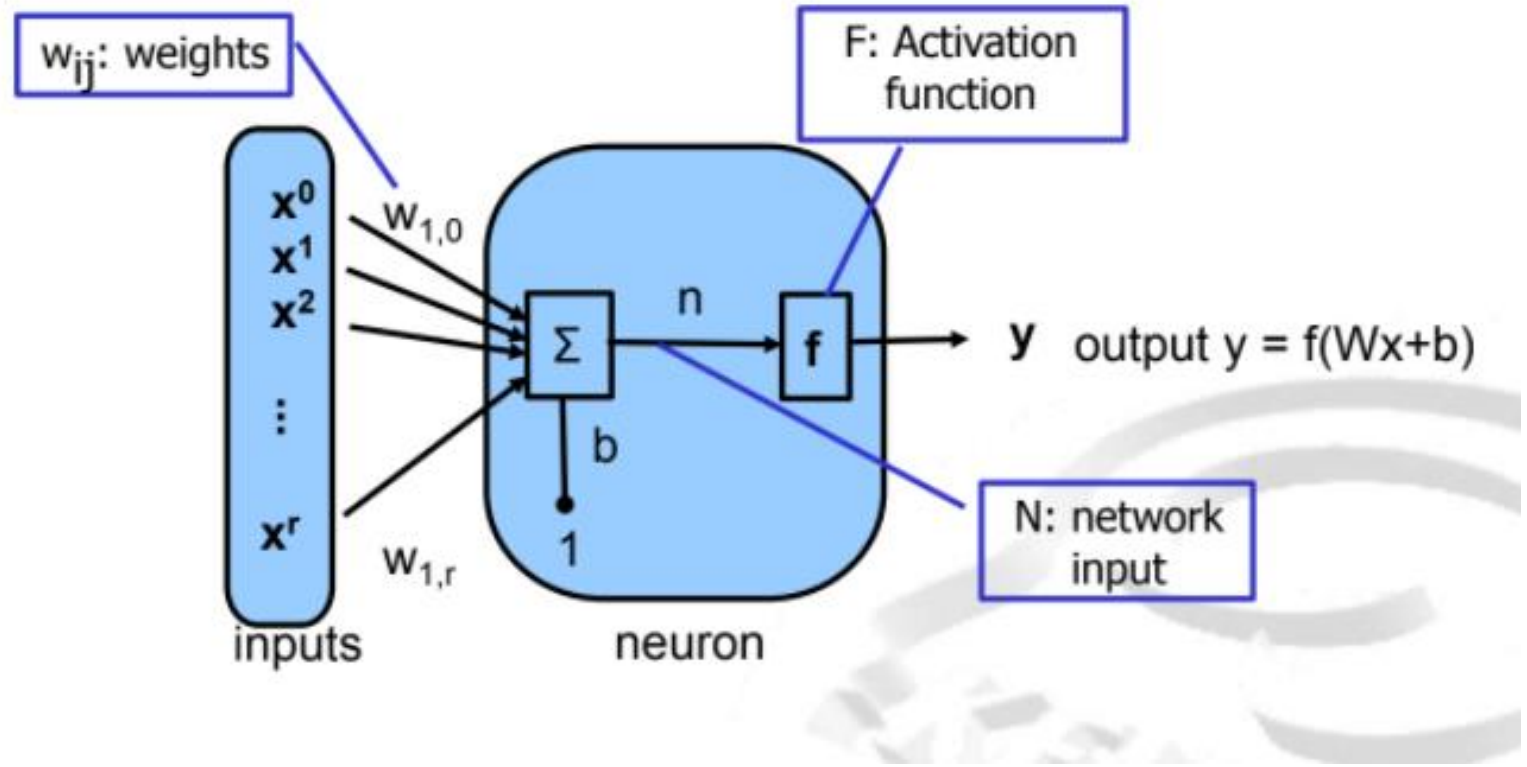
Recta de regresión lineal

$$y = w_i x_i + \cdots + w_d x_d + b$$

Cada neurona de la capa intermedia tiene un valor de bias.



Artificial Neural Networks (ANN)

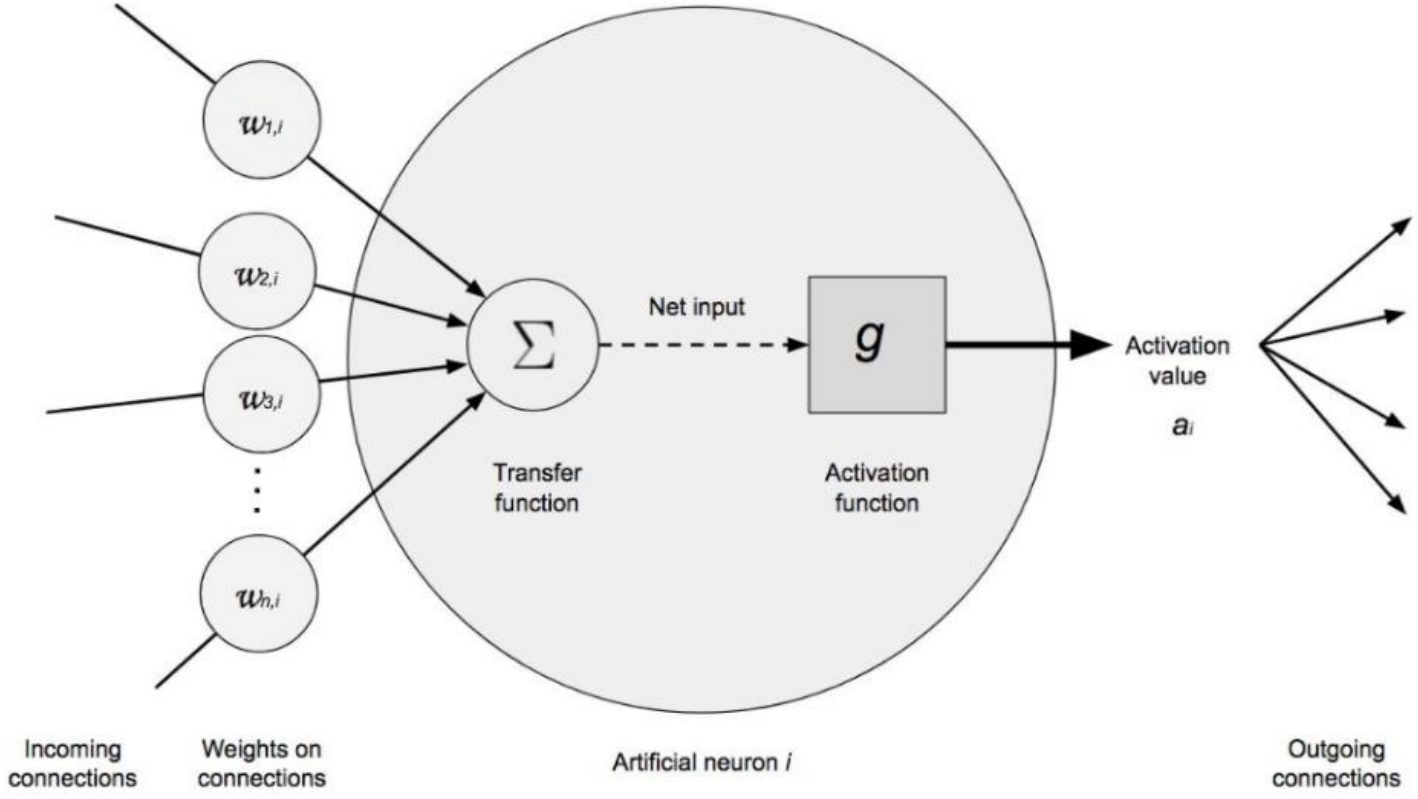


$$y_i = f(w_o + \sum_{\{j=1\}}^{\{P\}} W_j x_{ij} + \varepsilon_i$$

$$net_i = \sum_{\{i=0\}}^{\{P\}} W_j x_{ij}$$

La neurona

La neurona es la unidad funcional de los modelos de redes. Dentro de cada neurona ocurren simplemente dos operaciones: la suma ponderada de sus entradas y la aplicación de una función de activación.



La neurona

Para la capa de entrada la función de activación es la unidad. En la capa de salida la función de activación utilizada suele ser:

- **Identidad** para problemas de regresión

- **Softmax** para clasificación

Función de activación

- Las funciones de activación controlan qué información se propaga desde una capa a la siguiente (*forward propagation*).

- Estas funciones convierten el valor de entrada (combinación de los *input*, pesos y bias) en un nuevo valor. Gracias a combinar funciones de activación no lineales con múltiples capas, son capaces de aprender relaciones no lineales.

- Convierten el valor de entrada en un valor dentro del rango (0, 1) o (-1, 1).

- Cuando el valor de activación de una neurona es cero, se dice que la neurona está inactiva,

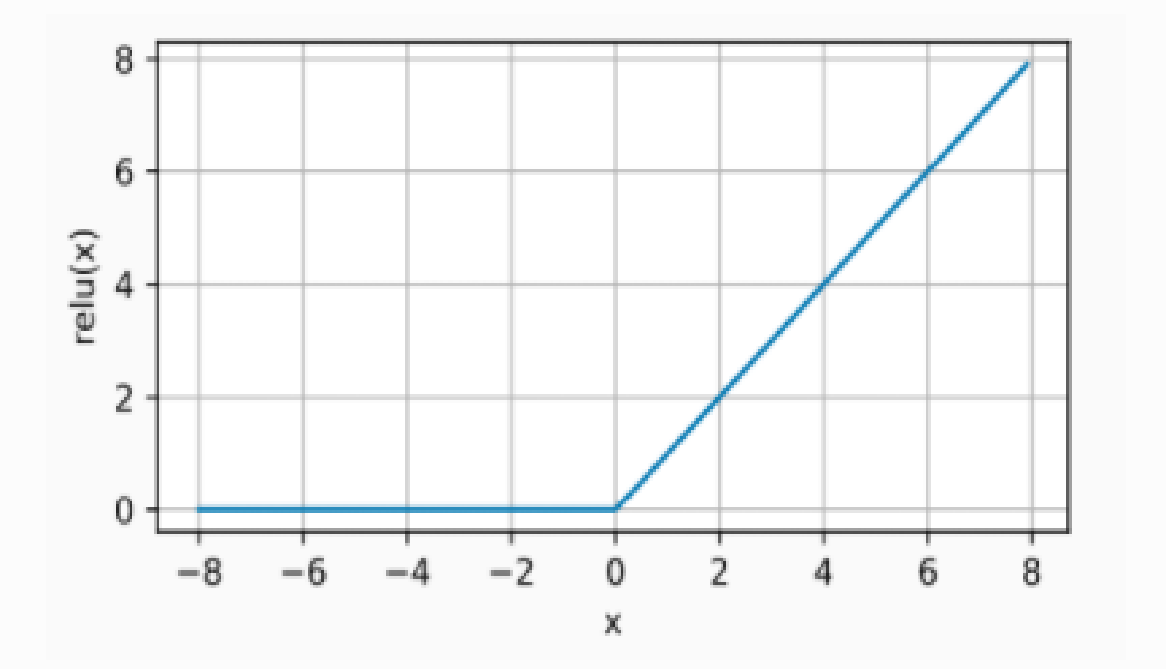
Función de activación

Las funciones de activación más empleadas son las siguientes:

Rectified linear unit (ReLU)

Aplica una transformación que activa la neurona solo si el *input* está por encima de cero. Mientras el valor de entrada está por debajo de cero, el valor de salida es cero, pero cuando es superior, el valor de salida aumenta de forma lineal con el de entrada.

$$\boldsymbol{ReLU(x) = \max(0, x)}$$



Función de activación

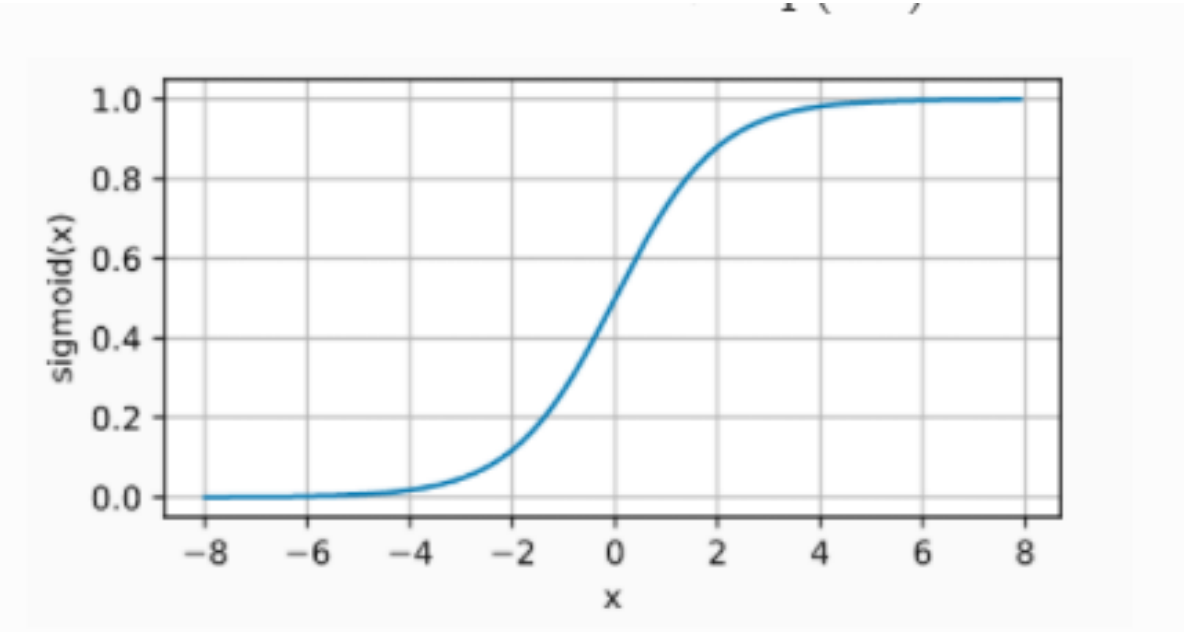
Las funciones de activación más empleadas son las siguientes:

Sigmoide

La función sigmoide transforma valores en el rango de (-inf, +inf) a valores en el rango (0, 1).

Un caso en el que la función de activación sigmoide sigue siendo la función utilizada por defecto es en las neuronas de la capa de salida de los modelos de clasificación binaria, ya que su salida puede interpretarse como probabilidad.

$$\boldsymbol{sigmoide(x) = \frac{1}{1 + exp(-x)}}$$



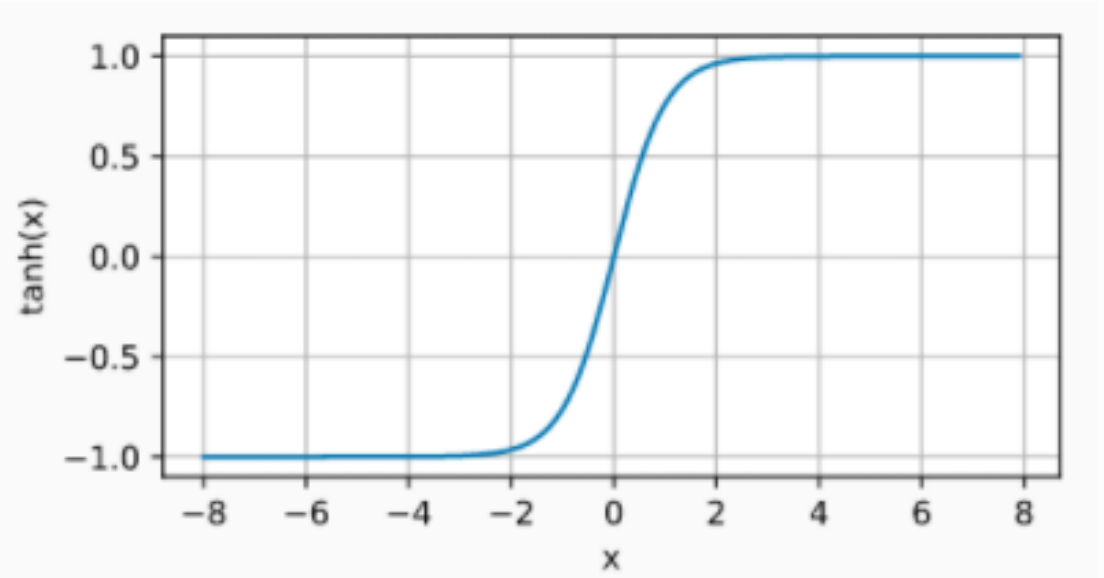
Función de activación

Las funciones de activación más empleadas son las siguientes:

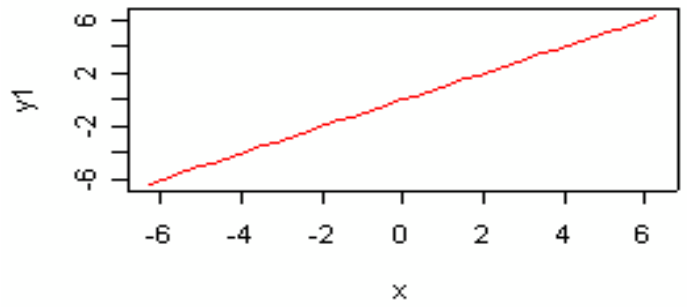
Tangente hiperbólica (Tanh)

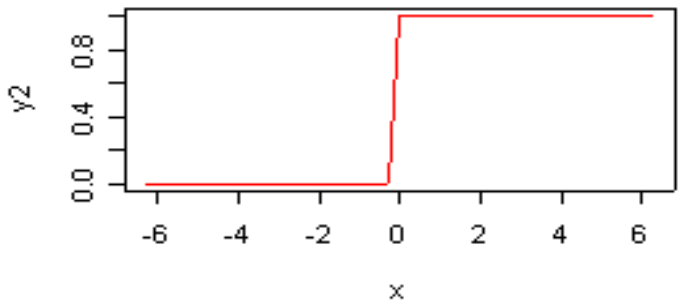
La función de activación su salida está acotada en el rango (-1, 1).

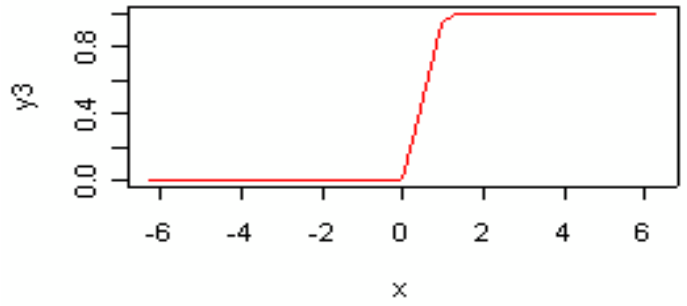
$$tanh(x) = \frac{1 - exp(-2x)}{1 + exp(-2x)}$$

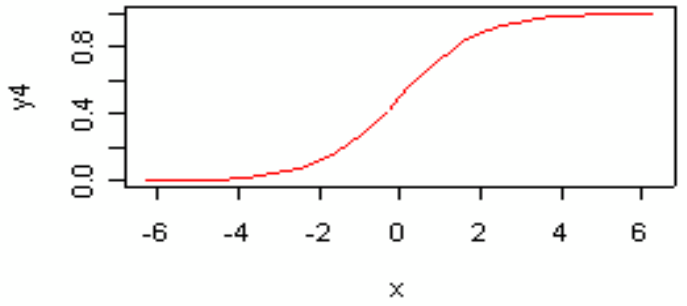


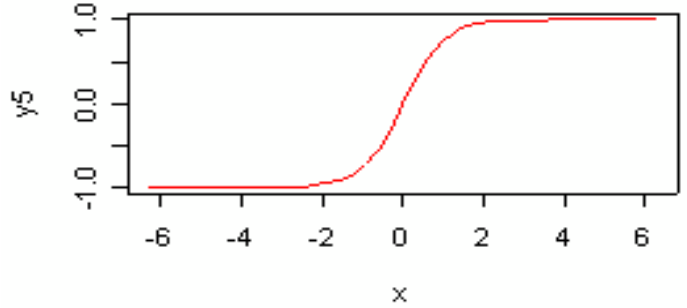
Sin las funciones de activación, las redes neuronales solo pueden aprender relaciones lineales.











Funciones de activación

Función de coste (loss function)

- La función de coste o función de pérdida es la encargada de cuantificar la distancia entre el valor real y el valor predicho por la red, mide cuánto se equivoca la red al realizar predicciones.

- Cuanto más próximo a cero es el valor de coste, mejor son las predicciones de la red (menor error), siendo cero cuando las predicciones se corresponden exactamente con el valor real.

- La función de coste puede calcularse para
  - una única observación
  - un conjunto de datos (entrenamiento de los modelos)
- Dependiendo del tipo de problema es necesario utilizar una función de coste u otra.
  - **Regresión**: error cuadrático medio y el error absoluto medio.
  - **Clasificación**: *log loss (logistic loss)* o *cross-entropy loss*.

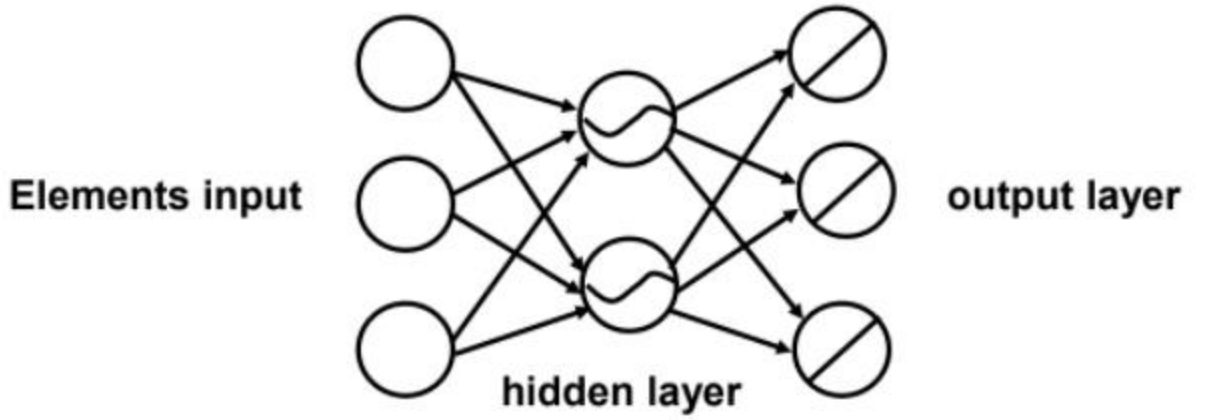
Función de coste (loss function)

Log loss, logistic loss o cross-entropy loss

En problemas de clasificación, la capa de salida utiliza como función de activación la función *softmax*. Gracias a esta función, la red devuelve una serie de valores que pueden interpretarse como la probabilidad de que la observación predicha pertenezca a cada una de las posibles clases.

Artificial Neural Networks (ANN)

Estructura de una red multi-capas

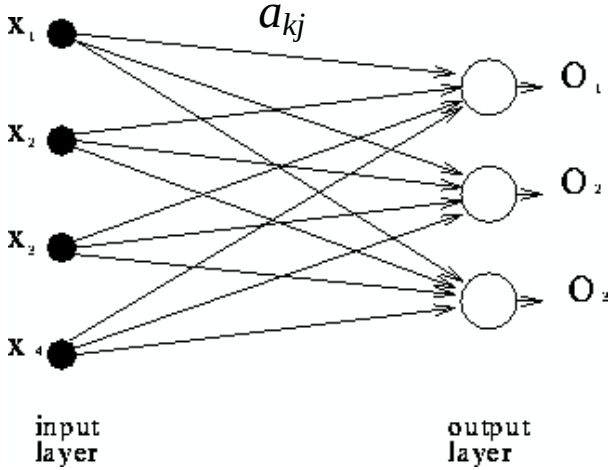


- Propiedades del cerebro emuladas por las ANN:
  - Computación paralela y distribuida
  - Conexión densa de unidades básicas
  - Las conexiones se pueden modificar a través de la experiencia.
  - Aprendizaje constante

El perceptrón (Rosemblatt, 1957)

$$y_{ik} = \sum_{j=1}^{retina} a_{kj} x_{ij}$$

Reconocimiento de 26 letras a partir de su imagen en la retina (400 pixeles)



Output $o_{ik} = g(y_{ik})$

Función de coste $$E_i = \frac{1}{2} \sum_{k=1}^{letters} (g(a'x) - t_{ik})^2$$

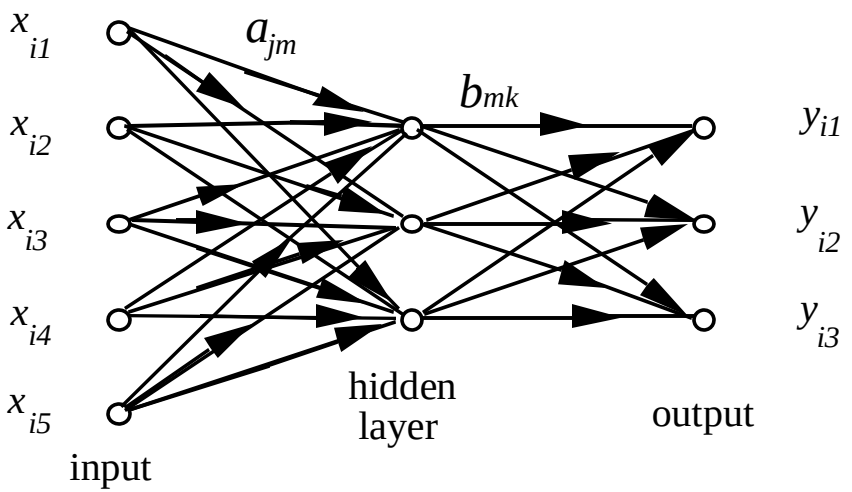
Estimación de los pesos $$a_{kj}^{t+1} = a_{kj}^{t} - \eta \frac{\partial E_i}{\partial a_{kj}}$$

$$\frac{\partial E_i}{\partial a_{kj}} = (o_{ik} - t_{ik}) g'(y_{ik}) \; x_{ik}$$

El perceptrón es como un sistema de ecuaciones de regresión simultáneas con función de activación no lineal.
El perceptrón (red neuronal con una capa) sólo puede discriminar regiones que sean linealmente separables.

A supervised multilayer perceptron

Perceptrón con una capa oculta (red neuronal de dos capas)



Input : **X** (*n*,*p*) (*p* variables de entrada –normalizada-)

Output : **Y** (*n*,*q*) (q variables de salida, binarias o continuas)

Pesos:

A designa la matriz (*p*, *c*) de pesos entre las neuronas de entrada y las de la capa oculta,

B la matriz (*c*, *q*) de pesos entre las neuronas de la capa oculta y las de salida.

Entrenamiento

El proceso de entrenamiento de una red neuronal consiste en ajustar el valor de los pesos y sesgo de tal forma que, las predicciones que se generen, tengan el menor error posible.

El entrenamiento de la red sigue los siguientes pasos:

1. Iniciar la red con valores aleatorios
2. Para cada observación calcular el error que comete la red al hacer la predicción y al final calcular el promedio de errores.
3. Identificar en que punto se ha cometido los errores
4. Modificar los pesos y el sesgo de la red en la dirección correcta
5. Repetir los pasos anteriores hasta que la red sea buena

Para realizar este proceso recurrimos a la retropropagación (backpropagation) y la optimización por descenso del gradiente (gradient descent).

Entrenamiento

Backpropagation

Es el algoritmo que permite cuantificar la influencia que tiene cada peso y sesgo en las predicciones de la red. Para ello, hace uso de la regla de la cadena para calcular el gradiente (vector formado por las derivadas parciales de una función). Por lo tanto mide cuánta “responsabilidad” ha tenido ese parámetro en el error cometido. Gracias a esto, se puede identificar qué pesos de la red hay que modificar para mejorarla.

Descenso del gradiente

Es el algoritmo de optimización que permite minimizar una función haciendo actualizaciones de sus parámetros en la dirección del valor negativo de su gradiente. Suele utilizarse el gradiente estocástico que consiste en dividir el conjunto de entrenamiento en lotes (minibatch o batch) y actualizar los parámetros de la red con cada uno.

Preprocesado (transformaciones)

One hot encoding (Binarización)

Consiste en crear nuevas variables dummy con cada uno de los niveles de las variables cualitativas.

Descenso del gradiente

Cuando los predictores son numéricos, la y la magnitud de su varianza pueden en el modelo. Si no se igualan de alguna forma los predictores, aquellos que se midan en una escala mayor o que tengan más varianza dominarán el modelo aunque no sean los que más relación tienen con la variable respuesta. Existen principalmente 2 estrategias para evitarlo:

- **Centralizar**: Consiste en restar a cada valor la media del predictor
- **Normalizar**

Preprocesado (transformaciones)

Normalización o Estandarización

Consiste en transformar los datos de forma que todos los predictores estén en la misma escala.

- **Normalización**: Dividir cada predictor entre su desviación tipo después de centrarlo

- **Estandarización**: Transforma los datos para incorporarlo dentro del rango [0, 1]

Hiperparámetros

Número y tamaño de capas

El número de capas y el número de neuronas determinan la complejidad del modelo y su potencial capacidad de aprendizaje.

La capa de entrada tiene tantas neuronas como predictores y la capa de salida tiene una neurona en problemas de regresión y tantas como clases en problemas de clasificación.

Cuantas más neuronas y capas, mayor la complejidad de las relaciones que puede aprender el modelo.

OJOOO!! Cómo cada neurona está conectada por pesos al resto de neuronas de las capas adyacentes, el número de parámetros a aprender aumenta y con ello el tiempo de entrenamiento.

Hiperparámetros

Learning rate

Cómo de rápido pueden cambiar los parámetros de un modelo a medida que se optimiza (depende de los datos e interacciona con el resto de hiperparámetros)

Si el learning rate es :

- muy grande, el modelo no será capaz de aprender.
- es muy pequeño, el proceso tardará demasiado y no llegar a completarse.

Recomendaciones:

- Utilizar un learning rate lo más pequeño posible siempre

- Utilizar valores mayores al inicio y pequeños al final.

Hiperparámetros

Algoritmo de optimización

Aunque existen multitud de adaptaciones, suele recomendarse:

- Conjuntos de datos pequeños: l-bfgs
- Conjuntos de datos grandes: adam o rmsprop

La elección del algoritmo de optimización puede tener un impacto notable en el aprendizaje de los modelos, sobre todo en *deep learning.*

Hiperparámetros

Regulación

Los métodos de regularización tienen el objetivo de reducir el sobreajuste (*overfitting*) de los modelos. Un modelo con sobreajuste memoriza los datos de entrenamiento pero es incapaz de predecir correctamente nuevas observaciones.

Los modelos de redes neuronales son modelos sobre parametrizados, por lo tanto, las regularizaciones son fundamentales. De entre las muchas que existen, destacan la regularización L1 y L2 (*weight decay*) y el *dropout*.

.

Hiperparámetros

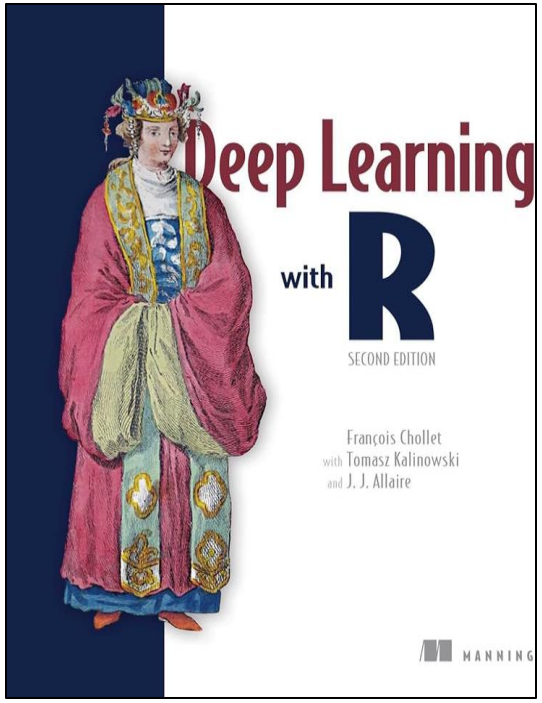
Regulación L1 y L2 (*weight decay*)

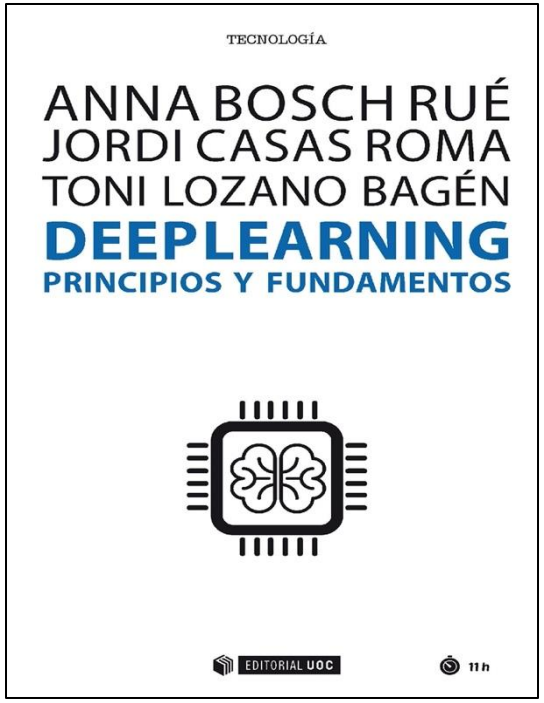
El objetivo es evitar que los pesos tomen valores excesivamente elevados. De esta forma, se evita que unas pocas neuronas dominen el comportamiento de la red y se fuerza a que las características poco informativas (ruido) tengan pesos próximos o iguales a cero.

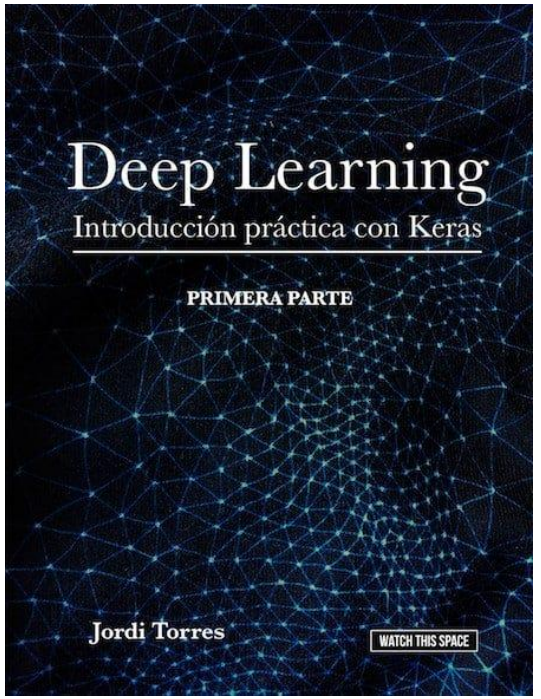
Dropout

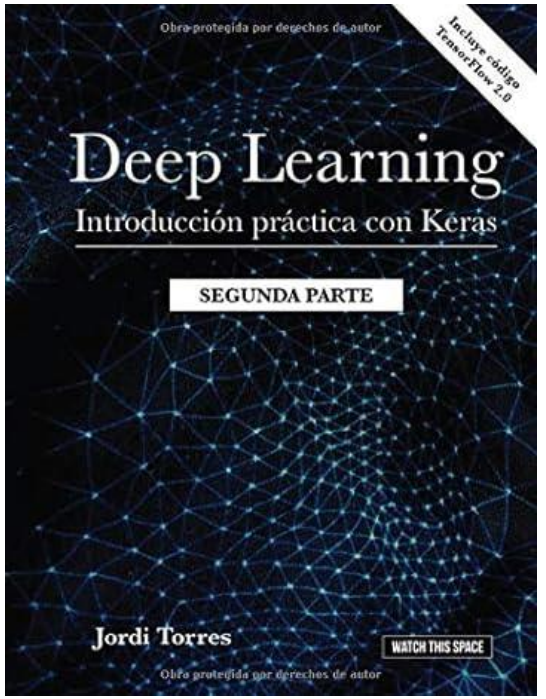
Este proceso consiste en desactivar aleatoriamente una serie de neuronas durante el proceso de entrenamiento. En concreto, durante cada iteración del entrenamiento, se ponen a cero los pesos de una fracción aleatoria de neuronas por capa. El porcentaje de neuronas que suele desactivarse por capa (dropout rate) suele ser un valor entre 0.2 y 0.5.

Bibliografia









https://torres.ai/

Deep Learning An MIT Press book

UNIVERSITAT DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH