2022

03 / 11 / 2022

Resumen sesión teoría 03/11

SISTEMAS INTELIGENTES

ADRIAN UBEDA TOUATI 50771466R

UNIVERSIDAD DE ALICANTE | Grupo 1 teoría

Contenido

Redes neuronales 2
Introducción 2
¿Como funcionan? 3
NAND 3
Rectas 4
Plano 4
Capas 5
¿Qué hace una red neuronal en cada capa? 5
Ejemplo MNIST 6
Error cuadrático de la red 9

Redes neuronales

Introducción

Las redes neuronales intentan asemejarse a las redes neuronales orgánicas, pero están muy lejos de ser alcanzadas, el cuerpo humano tiene 100 mil millones de neuronas.

Son muy útiles para ciertas aplicaciones en cada campo, pero no son la solución a todo.

Reconocimiento de imagen

Reconocimiento del habla

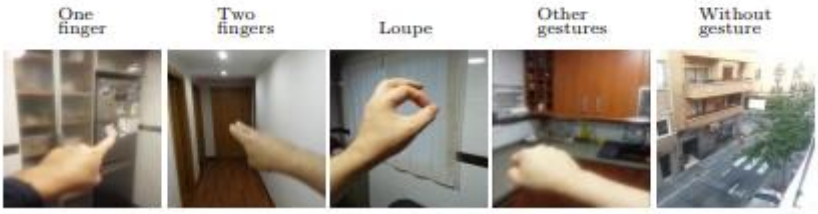
Procesamiento del lenguaje natural

Conducción autónoma

Diagnosis médica

...

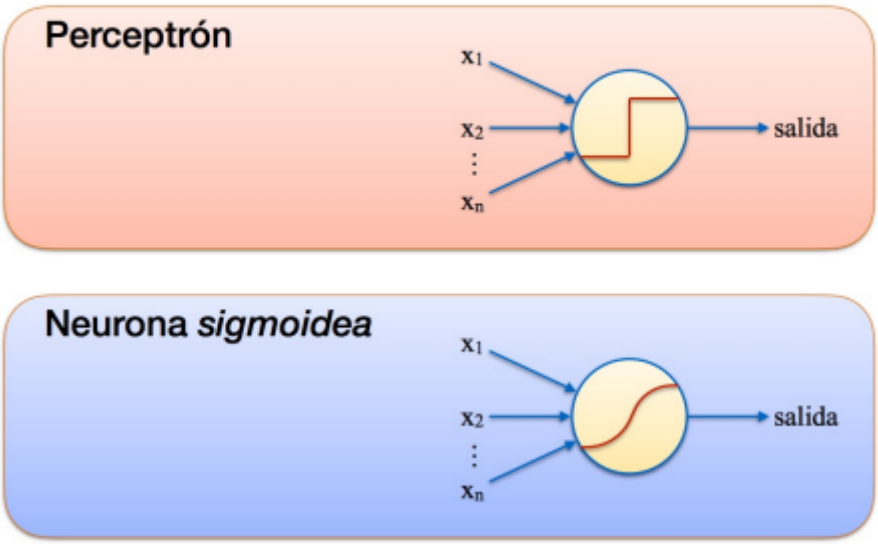
Ejemplo: Reconocimiento de gestos



Las redes neuronales deben ser entrenadas, deben tener una serie de matrices de pesos. Y que gracias a estas matrices sea capaz de predecir cosas y que den el resultado esperado.

Hay 2 tipos de neuronas artificiales

Neuronas artificiales



Con una función más estricta o progresiva

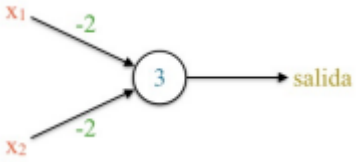
Pueden construir cosas que matemáticamente son muy complicadas, lleno a imposibles.

¿Como funcionan?

Cada entrada tiene un peso, y el perceptrón toma la decisión, peso*entrada…

Y dependiendo del umbral si es alcanzado o no, será 0 o 1

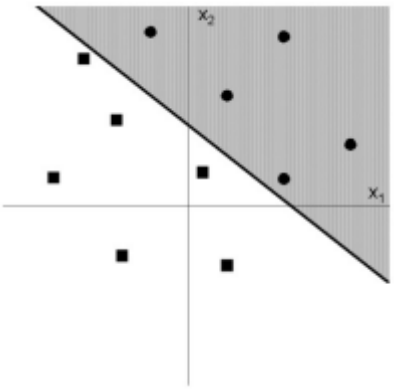
NAND



x_1	x_2	$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} -2 \\ -2 \end{bmatrix} + 3$	Salida
0	0	3	1
0	1	1	1
1	0	1	1
1	1	-1	0

Par calcularlo, si hay 2 bloques separadas, el objetivo es conseguir una ecuación de la recta que separe esas 2 regiones. Hay infinitas soluciones

Rectas



Una recta suele tener 2 planos, uno positivo respecto de la recta y otro negativo respecto de la recta

Para saber donde esta cada zona, lo primero es coger un punto que sepamos en que región esta, la sustituimos en la recta, si uno es negativo todos los puntos son negativos, y el plano complementario o será el positivo.

Si en la recta cogemos un punto de la recta nos devolverá 0

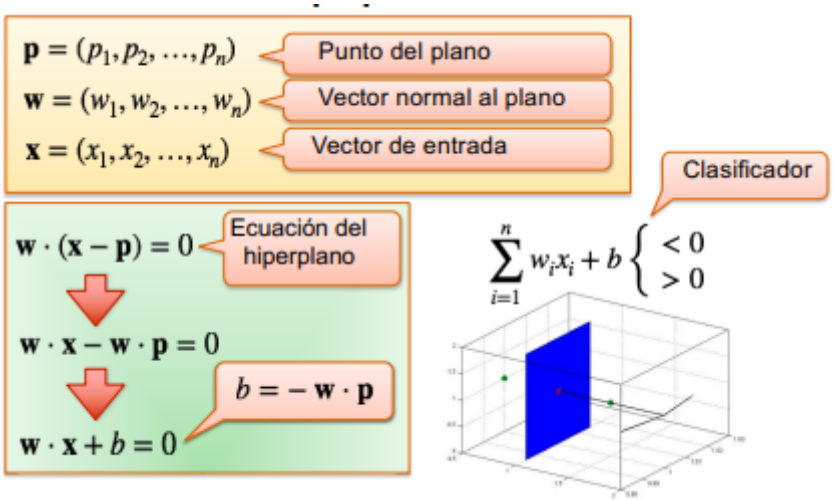
Cada plano dependerá de su recta

Plano

Se puede usar tantas dimensiones como queramos

Con una dimensión menos, creamos un hiperplano plano

Y así de esta forma seguir utilizando las redes neuronales



Para aquellos casos mas complicados donde no podremos separar en 2 los bloques, tendremos que combinar varios perceptrones

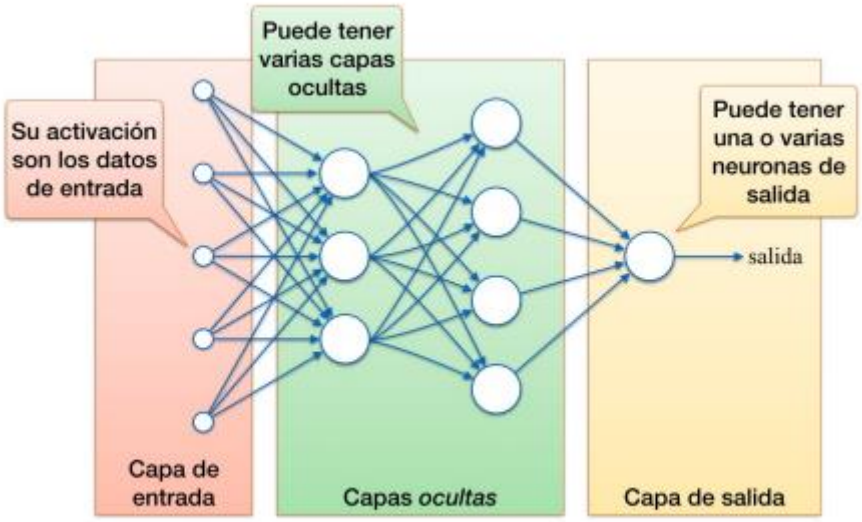
Capas

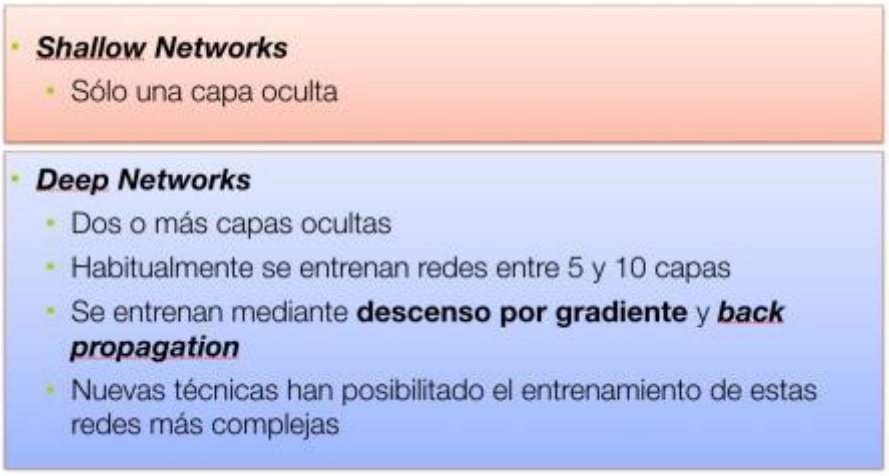
En cada capa podemos poner todas las capas que queramos

Cada neurona de la capa anterior de conecta a las neuronas de la siguiente capa y así sucesivamente se conectan todas las neuronas.

Haremos un calculo matricial para poder conseguir el resultando.

Tipos de capas:





¿Qué hace una red neuronal en cada capa?

En la primera capa separa en semi planos

En el 2 nivel formas elaboradas

Y por cada capa generalizaremos más.

Un numero excesivo de capas internas o insuficiente, puede empeorar el aprendizaje.

Se ajusta con prueba y error

- **Capa de entrada**
  - Tenemos en cuenta cómo podemos descomponer los datos que recibimos como diferentes neuronas de entrada
- **Capa de salida**
  - Tenemos en cuenta cómo podemos codificar el resultado que queremos obtener como salida
- El diseño de las **capas ocultas** no es trivial

El diseño de las capas oculta no es lineal.

Ejemplo MNIST

Reconocimiento de números en manuscrito, se ha guardado miles de personas en una base de datos con su forma de escribir los números entre 0 1

- **Capa de entrada**
  - Imágenes de 28x28 píxeles
  - Niveles de gris de [0, 1]
  - ***784 neuronas*** de entrada con valores [0, 1]

- **Capa de salida**
  - Número del 0 al 10
  - ***10 neuronas*** con valores {0, 1}

- **¿Cómo ajustamos los pesos?** **¡Puede ser entrenada!**

El entrenamiento, se efectúa con un conjunto de ejemplos ya resueltos.

Los dividiremos en 2 conjuntos:

- **Conjunto de entrenamiento (*training*)**
  - Ejemplos con los que ajustamos los pesos de la red
- **Conjunto de prueba (*test*)**
  - Ejemplos para validar los resultados se clasificación de la red

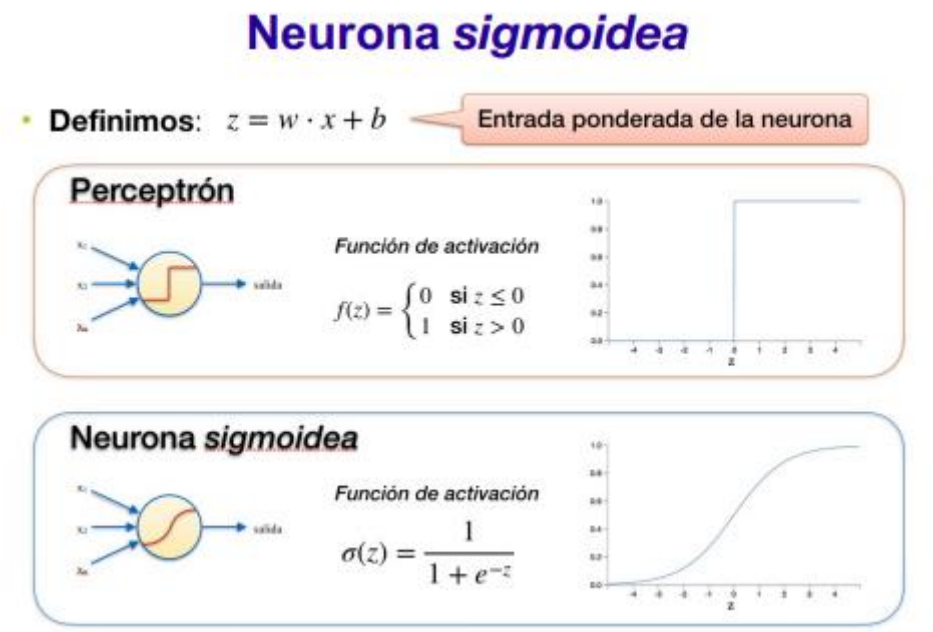
Cuando creemos que la maquina ya esta entrenada, pasamos el conjunto de pruebas y si la da los resultados esperados, ya estará lista

Si no tenemos los suficientes ejemplos, no tiene sentido realizar ningún entrenamiento.

Como entrenamos redes neuronales

Mecanismo de retro comparación, cojo cada ejemplo y veo que error se genera, y gracias por retro propagación voy modificando los pesos en función de esa discrepancia y vamos reajustando los separadores hasta que nos lleve a un estado de estabilidad en la red

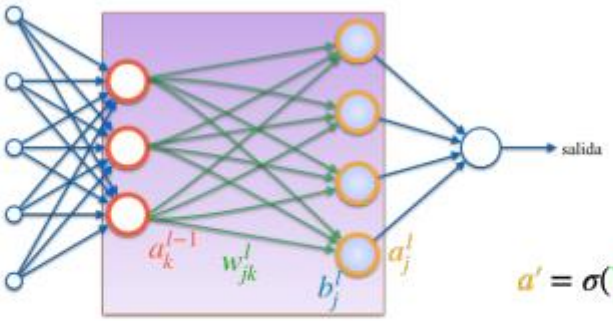
Con el perceptrón, no podremos hacer cambio gradualmente, pero en la neurona sigmoidea, podemos hacer pequeñas variaciones en la variable y así en la función, y es la más usada, es derivable y en el límite actúa como un escalón



Es la función de activación más usada

Las redes neuronales a nivel de implementación se realizan con matrices.

Cálculo de la activación en cada capa



$$a' = \sigma(Wa + b)$$

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \quad a' = \begin{bmatrix} a'_1 \\ a'_2 \\ a'_3 \\ a'_4 \end{bmatrix}$$

$$\begin{bmatrix} a'_1 \\ a'_2 \\ a'_3 \\ a'_4 \end{bmatrix} = \sigma\left(\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \right)$$

Entrada:

Datos de entrada: x

Matrices de pesos de cada capa: $w = [W^1, W^2, \ldots, W^L]$

Tuplas de *biases* de cada capa: $b = [b^1, b^2, \ldots b^L]$

Algoritmo FeedForward

$a \leftarrow x$

Para cada capa $l \in [1,\ldots,L]$

$a \leftarrow \sigma(W^l a + b^l)$

Devolver a

El algoritmo FeedForward, asocia la entrada a una salida

Mediante pruebas se consigue la aplicación de convergencia

Error cuadrático de la red

a: **salida real** de la red para la entrada x, pesos w y biases b

$y(x)$: **salida esperada** de la red para la entrada x

n: **Número total de ejemplos** de entrenamiento (*training*)

Podemos definir una **función de coste** C que evalúe el error cuadrático medio (MSE) de clasificación de la red:

$$C(w,b) = \frac{1}{2n}\sum_x \|y(x) - a\|^2$$

Debemos minimizar su valor