

# Redes neuronales

Scribo

- Es una familia de algoritmos potentes con los que podemos modelar comportamientos inteligentes.
- Su comportamiento emerge de la interacción de muchas partes trabajando conjuntamente. A cada parte de la red neuronal se le denomina neurona.

## Parte 1: Neuronas

Es la parte básica de la red neuronal, tienen conexiones de entrada en la que a través reciben estímulos externos, que son valores de entrada y con ellos realizará un cálculo interno y generará un valor de salida.

Ejemplo:

Variables binarias:

VR:  $x_1$

Noche:  $x_2$

entrada

Price night:  $y_i$

salida

0,1

\* Se analiza el valor de la  
regresión lineal y si lo supera  
el umbral es  $y=1$   
si es inferior  $y=0$

Por lo tanto, el valor de la salida dependerá de si el resultado de la neurona es mayor o menor que 0.

BIAS = -UMBRAL

$$w_1x_1 + b \leq 0 \rightarrow y=0$$

$$w_1x_1 + b > 0 \rightarrow y=1$$

→ Se prueban las diferentes combinaciones para obtener un resultado positivo.

→ La combinación de parámetros tiene que marcar una frontera, se va ajustando los valores hasta encontrar lo que estamos buscando.

→ Al combinar varios neuronas se da solución a problemas más complejos.

# Capítulo 10: Redes Neuronales

## Parte 2: Red neuronal

- Con dos neuronas en una misma capa, estas recibirán la misma información de entrada, los cálculos los pasan a la siguiente.
- Capa de entrada → capas ocultas → capas de salida →
- Secuencia: una neurona recibe la información de otras, y pueden jerarquizarla y así adquirir más conocimiento. La profundidad es la cantidad de capas, y es lo que da nombre al Deep learning (aprendizaje profundo).
- Para lograr esto se deben concatenar, pero al sumar muchas líneas rectas de otra linea recta o puede colapsar, tiene que ocurrir una operación que las distorsione. Pasar el valor de salida por la función de activación  $f(x)$  y distorsionará el valor de salida para poder concatenar.

### Ejemplo: función de activación

$f(wx) \rightarrow Y = \{0, 1\}$  es la función escalonada porque el cambio de valor se produce instantáneamente.

#### • Función de activación sigmoidal

$$f(x) = \sigma(x) = \frac{1}{e^x + 1}$$

Conseguimos deformación y

representar probabilidades de 0 al 1

#### • Función de activación TANH

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{(e^x + e^{-x})}$$

Rango varia de -1 a 1

#### • Función de activación RELU

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

Es lineal cuando es positiva

y es 0 cuando el valor de entrada es negativo.

Ejemplo 2: ¿Qué ocurre en una red neuronal?

El trabajo de la función de activación es distorsionar el plano generado por la neurona.

Sigmoid   
TANH   
RELU 

Una posible solución es colocar una función sigmoid

- Colocar cuatro neuronas con una orientación diferente. Con una nueva neurona se construye la combinación de cuatro figuras geométricas, obteniendo como resultado una superficie plana. Esta sería la solución porque la intersección del plano con la montaña produce la frontera circular que buscaban.

## Parte 2.5 Jugando con redes neuronales

Herramientas: Neural Network

Variables de entrada

$X_1$

$X_2$

Variables de salida → Resultado que asigna a los ejes.

- ¿Qué ocurre en la red neuronal?

- + las neuronas son una neurona gigante y no funcionan

- Tendríanos millones

- + Anadir 3ra neurona → se genera una estructura en el espacio multidimensional que puede ser intersectada con un plano y puede ser y genera la frontera de separación buscada.

- Número de variables de entrada

- + la red encuentra solución

- + lo función de activación uniforme funciona más rápido al entrenar la red.

- + con los dos redes puede encontrar esa red perfecta

### Parte 3: ¿Qué es una red neuronal? → Backpropagation

1986 → se publicó un artículo científico de redes neuronales

1950 → se conoció como perceptrón la primera red neuronal → solo resolvía problemas lineales  
→ No se sabía cómo entrenarlas.

- Antes se evaluaba el error del modelo en el punto en el que se encontraba, se calculaban las derivadas parciales en dicho punto, se obtiene el vector que indicaba la pendiente de la función, es decir hacia donde se iba el error (el gradiente) y por eso nos iban a la dirección contraria por donde lo disminuían.

Regresión simple  $y = w_0 + w_1 x$  (dos parámetros)

• Cómo varía el costo ante un cambio del parámetro  $w$ ?

$\frac{\partial C}{\partial w} \rightarrow$  función

$\frac{\partial w}{\partial w} \rightarrow$  parámetros

- Descenso del gradiente → optimizar la función de custo usando el gradiente

↳ Backpropagation

- Si encontramos la neurona que es responsable del error

• El error de las capas anteriores depende directamente del error de las posteriores

• Responsabilizamos las neuronas → reparto de dichos errores

• Una vez que se asume el error se va llevando el error hacia atrás, de una por una.

• Estos errores se usan para calcular las derivadas parciales de cada parámetro de la red, y es lo que necesitamos para minimizar el error con el gradiente.