

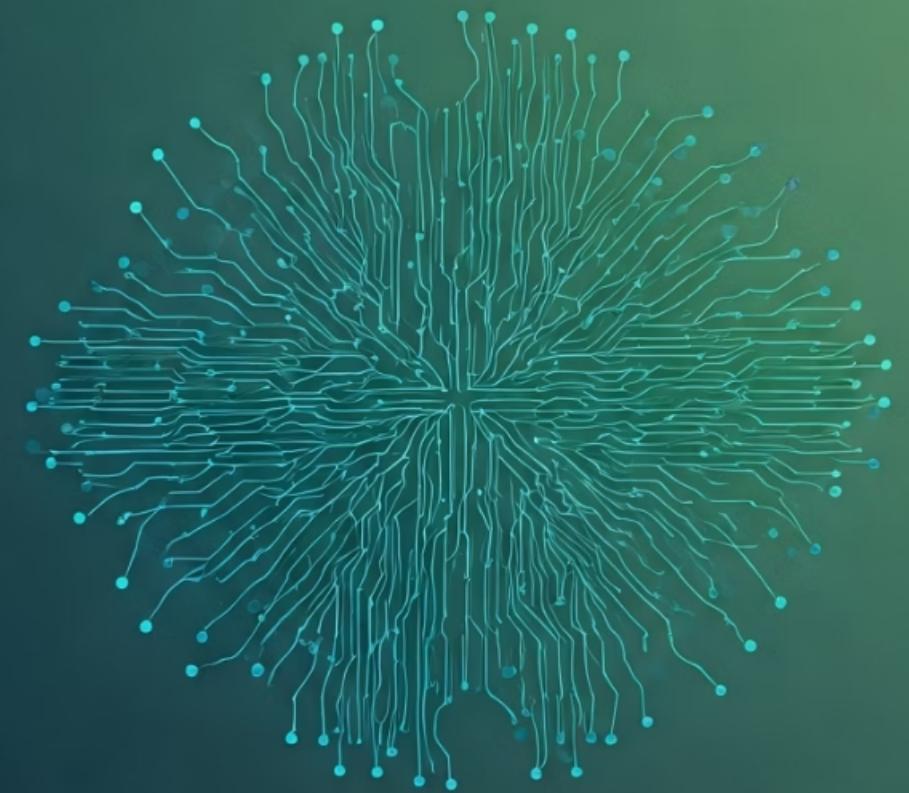
El algoritmo de backpropagation

El algoritmo de backpropagation es una de las piedras angulares del aprendizaje de redes neuronales.

El backpropagation revolucionó el campo de la inteligencia artificial al permitir el entrenamiento efectivo de redes neuronales multi capa, sentando las bases para el desarrollo del deep learning moderno que conocemos hoy.

A lo largo de esta presentación, exploraremos sus orígenes históricos, diferentes analogías para comprender su funcionamiento y sus fundamentos matemáticos.

Clase impartida por Andrea Alonso





Evolución histórica del backpropagation

Antecedentes (1940s-1960s)

McCulloch y Pitts proponen el primer modelo de neurona artificial (1943). Frank Rosenblatt desarrolla el Perceptrón (1957), pero solo podía resolver problemas linealmente separables. Minsky y Papert demuestran estas limitaciones en *Perceptrons* (1969).

1

El renacimiento (1980s)

Paul Werbos desarrolla las bases matemáticas del backpropagation (1974-1982). Rumelhart, Hinton y Williams lo popularizan en su trabajo *Learning representations by back-propagating errors* (1986).

2

El "Invierno de la IA" (1970s)

Las limitaciones del perceptrón provocaron un período de desinterés en las redes neuronales. La comunidad científica se centró en otros enfoques de inteligencia artificial.

3

¿Qué es backpropagation y para qué sirve?

Definición simple

Es un algoritmo que permite a las redes neuronales "aprender de sus errores" ajustando automáticamente sus pesos para mejorar sus predicciones.

El problema que resuelve

Cuando una red neuronal hace predicciones incorrectas, el backpropagation determina qué pesos deben ajustarse y en qué cantidad para mejorar el rendimiento.

La función de backpropagation

Calcula el error comparando la predicción con el valor real, propaga el error hacia atrás por toda la red, y ajusta los pesos en la dirección que reduce el error.

Podemos entenderlo con la **analogía del chef perfeccionista**: prueba el plato (predicción), nota que está muy salado (error), identifica que echó demasiada sal (backpropagation), reduce la sal en la próxima preparación (ajuste de pesos) y repite hasta perfeccionar la receta (entrenamiento).

Ejemplos y analogías intuitivas



La Cadena de Responsabilidad

Como una empresa rastreando un error desde el CEO hasta el operario que necesita ajustar su máquina.



El Efecto Dominó Inverso

Un dominó cae al final (error) y miramos hacia atrás para entender la contribución de cada pieza anterior.



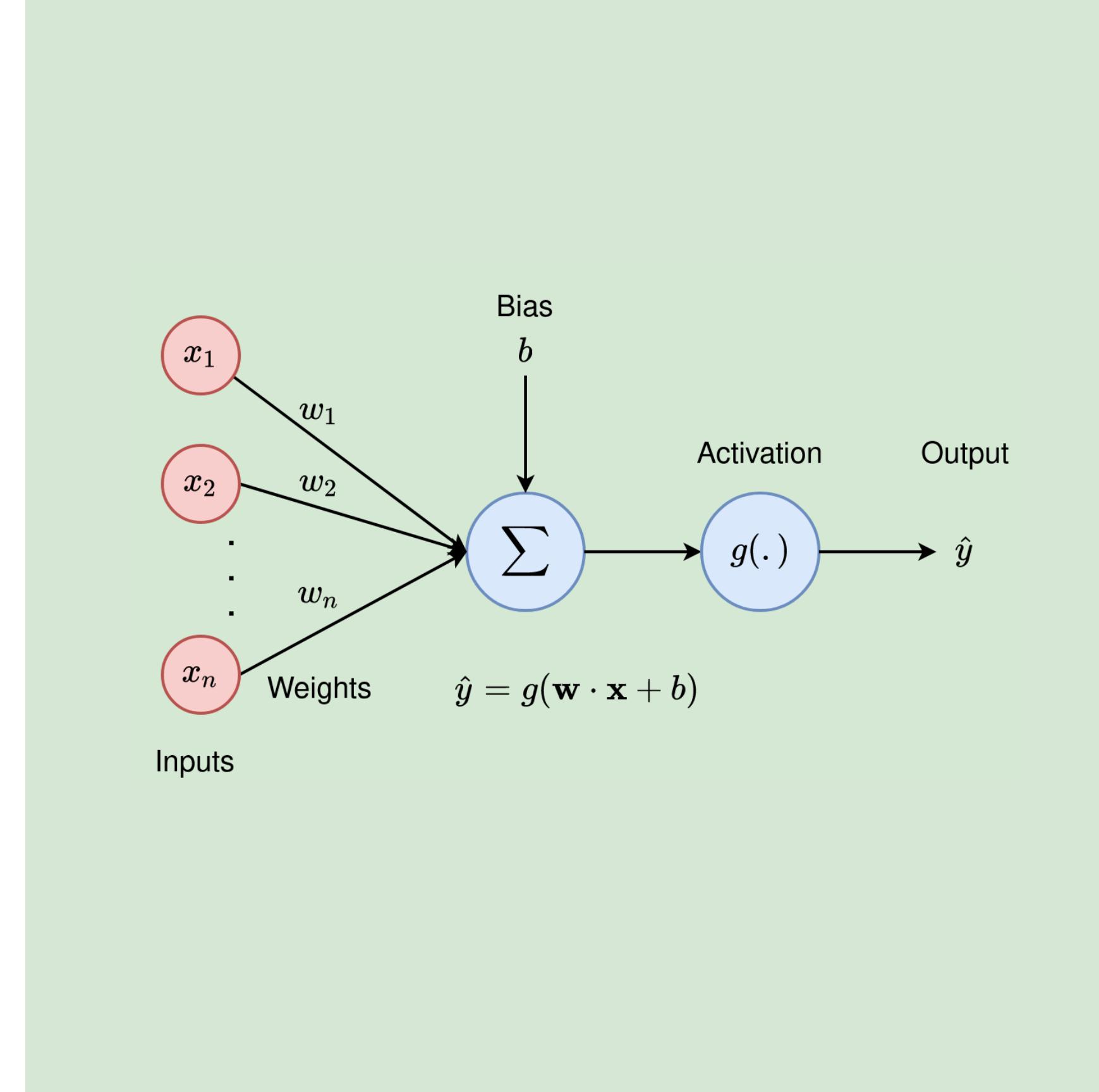
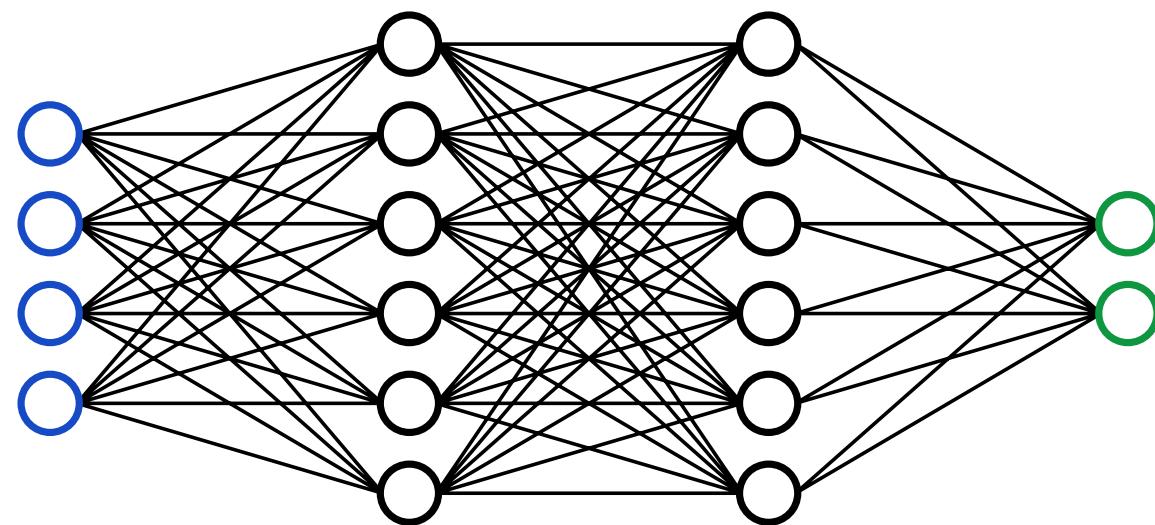
El Teléfono Descompuesto

Propaga el "mensaje de error" hacia atrás, informando a cada persona (neurona) sobre su contribución a la distorsión.

Estas analogías nos ayudan a comprender cómo backpropagation rastrea el error desde la salida hasta encontrar exactamente qué "operarios" (pesos) necesitan ajustarse, calculando la contribución precisa de cada elemento al resultado final.



La red neuronal



Relación con el descenso del gradiente



***Chain rule:** es una herramienta fundamental del cálculo que permite **calcular derivadas de funciones compuestas**. Si visualizamos a la red neuronal como una cadena de funciones, para saber cómo afecta un peso al error final, tendremos que encadenar derivadas usando la **regla de la cadena**.



Funcionamiento paso a paso (conceptual)



Propagación hacia adelante

La información entra por las neuronas de entrada, se procesa capa por capa hasta llegar a la salida, generando una predicción.

Cálculo del error

Se compara la predicción con el valor real y se calcula una medida del error (ej.: error cuadrático medio).

Propagación hacia atrás

Usando derivadas y la chain rule, el error se propaga desde la salida hacia las capas anteriores. Cada neurona recibe información sobre su contribución al error.

Repetición

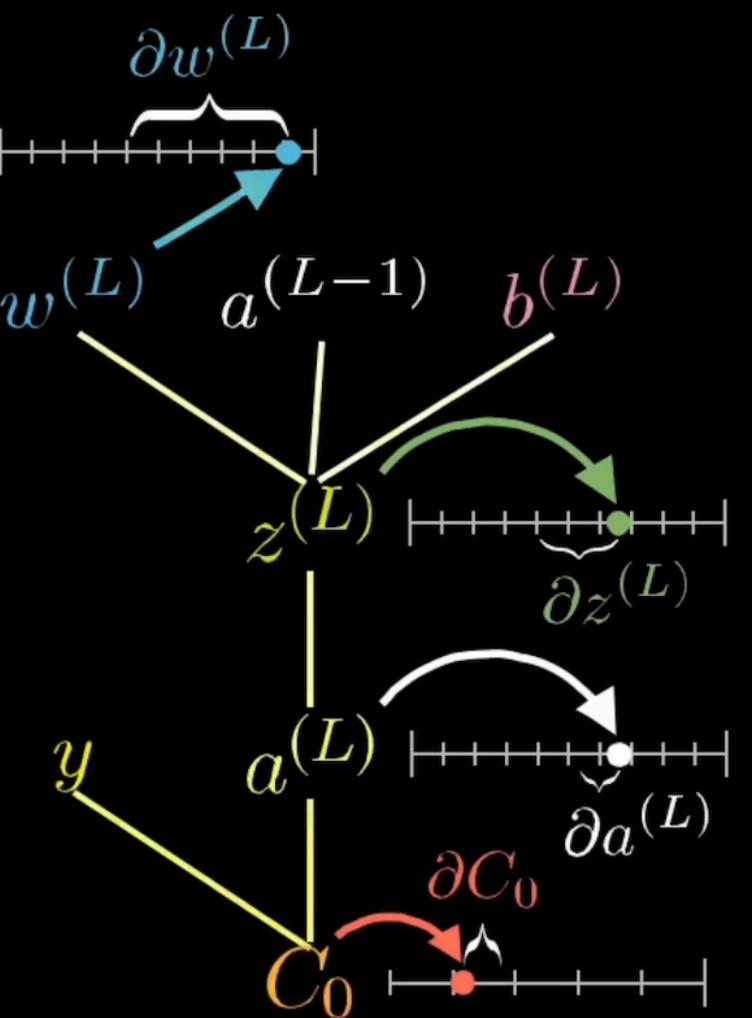
Tas calcular las derivadas, se utiliza el descenso del gradiente para modificar los pesos, y después el proceso se repite miles de veces hasta que la red aprende a hacer predicciones precisas.

Fundamentos matemáticos

Notación básica

- w_{jk}^l : Representa el peso que conecta la neurona k de la capa l-1 con la neurona j de la capa l
- b_j^l : El bias de la neurona j en la capa l
- a_j^l : La activación (salida) de la neurona j en la capa l
- z_j^l : La entrada ponderada de la neurona j en la capa l, antes de aplicar la función de activación
- σ : La función de activación (como sigmoide, tanh, o ReLU)
- C : Nuestra función de costo que queremos minimizar
- δ_j^l : El error de la neurona j en la capa l (esto es central en backpropagation)

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}}$$



Fundamentos matemáticos

Ecuaciones fundamentales

1. Propagación Hacia Adelante

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l$$

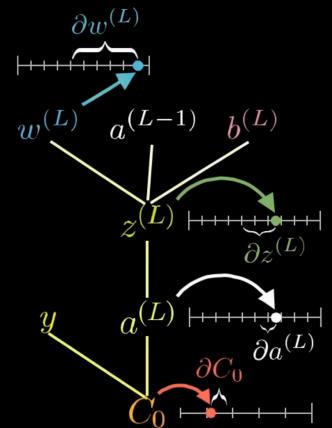
$$a_j^l = \sigma(z_j^l)$$

2. Función de Costo (Error Cuadrático Medio)

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

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}}$$

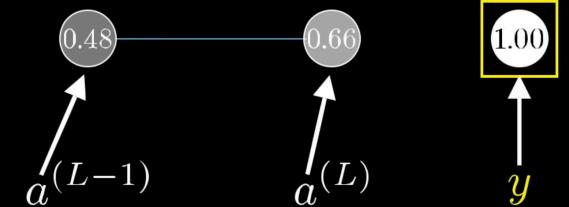
$$C_0(\dots) = (a^{(L)} - y)^2$$



$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$$

$$a^{(L)} = \sigma(z^{(L)})$$

Desired output



$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}}$$

$$C_0 = (a^{(L)} - y)^2$$

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$$

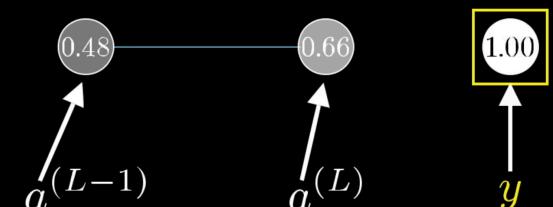
$$\frac{\partial C_0}{\partial a^{(L)}} = 2(a^{(L)} - y)$$

$$a^{(L)} = \sigma(z^{(L)})$$

$$\frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma'(z^{(L)})$$

$$\partial w^{(L)}$$

$$\frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)}$$



Las cuatro ecuaciones de backpropagation

Error en la capa de salida

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \times \sigma'(z_j^L)$$

Interpretación: El error en la última capa depende de cuánto afecta esa neurona al costo total y de qué tan sensible es la función de activación.

Propagación del error

$$\delta_j^l = \left(\sum_k w_{kj}^{l+1} \delta_k^{l+1} \right) \times \sigma'(z_j^l)$$

Interpretación: El error de una neurona es la suma ponderada de los errores de las neuronas de la siguiente capa, multiplicada por la derivada de la función de activación.

Gradientes

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \times \delta_j^l$$

Interpretación: El gradiente del sesgo es directamente el error de esa neurona. El gradiente del peso es el producto de la activación de entrada y el error de salida.

Recursos de aprendizaje

Videos

Más intuitivos:

- [¿Qué está haciendo realmente la retropropagación?](#), 3Blue1Brown
- [Qué es una red neuronal | Backpropagation](#), Dot CSV

Más matemáticos:

- [Cálculo de retropropagación](#), 3Blue1Brown
- [¿Qué es una Red Neuronal? | Las Matemáticas de Backpropagation](#), Dot CSV

Artículos

- [Blog: Cómo funciona el algoritmo de backpropagation](#), David Díaz S., Medium