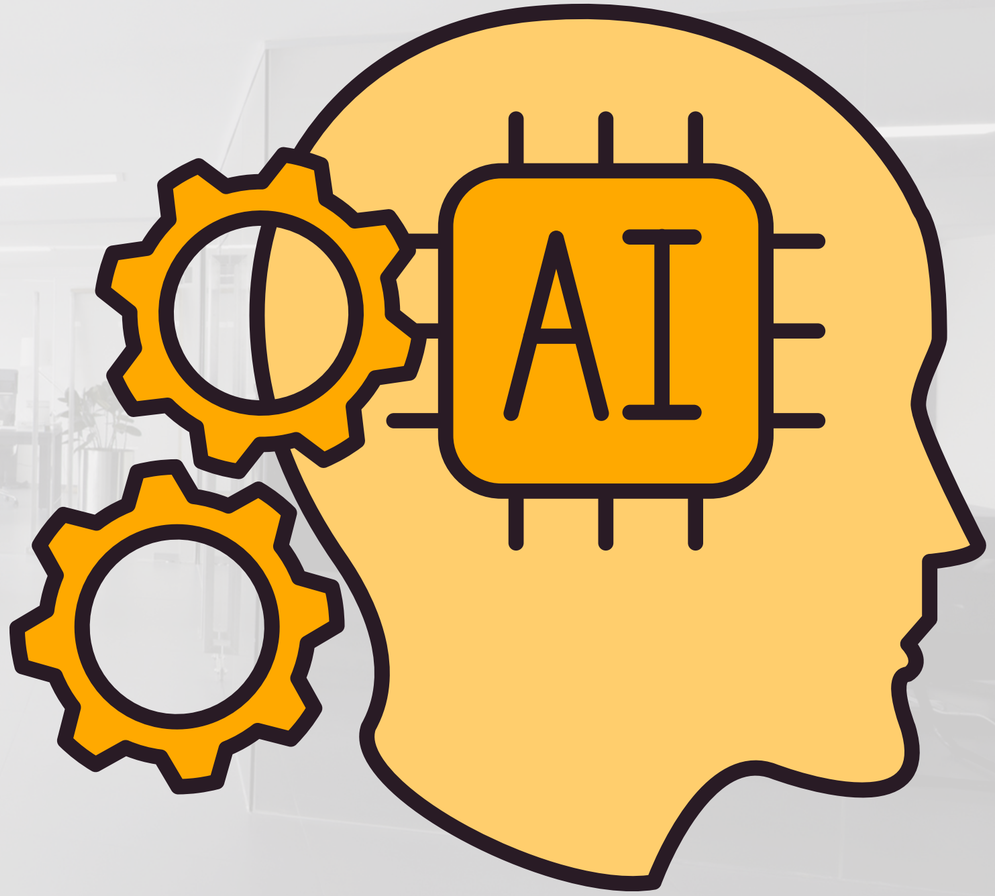
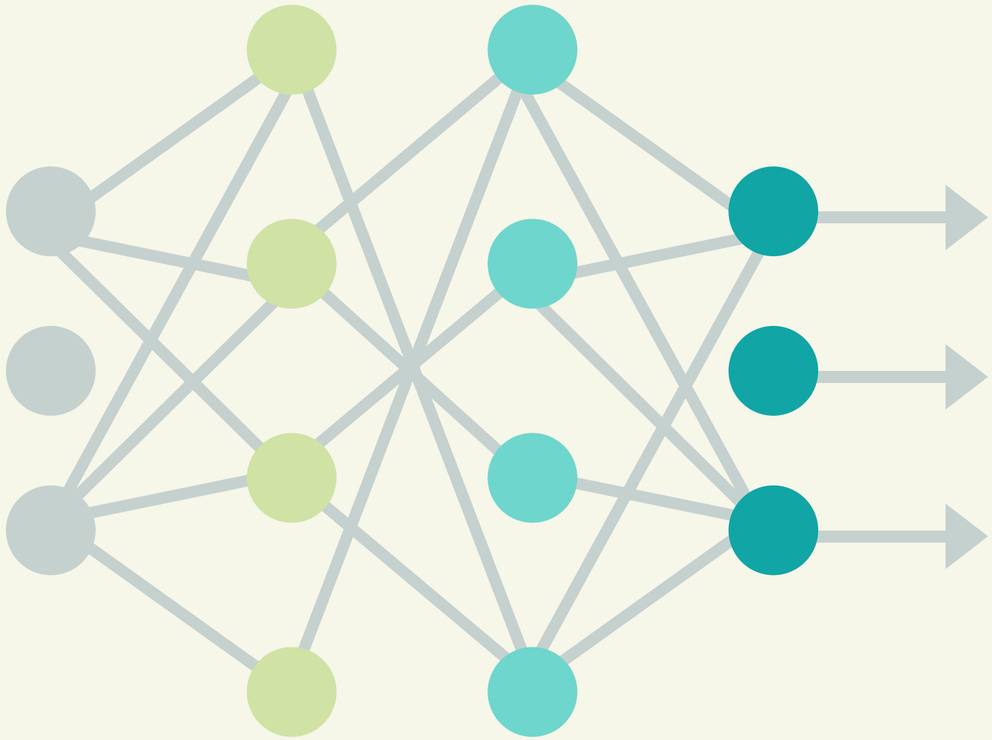
BACKPROPAGATION

PRESENTADO POR:

- DANIEL FELIPE SORACIPA
- JUAN JOSE MEDINA
- CRISTIAN DANIEL MONTAÑEZ
- SERGIO ALEJANDRO RUIZ



ARQUITECTURA DE UNA RED NEURONAL



Una red neuronal artificial se compone de capas de unidades de procesamiento llamadas neuronas. Estas capas están organizadas de manera secuencial: la entrada pasa por una o varias capas ocultas antes de llegar a la salida. Cada neurona transforma la información que recibe aplicando una combinación lineal de sus entradas seguida por una función de activación no lineal. La arquitectura define cómo fluye la información a través de la red y determina su capacidad de aprendizaje.

1. Capa de entrada

2. Capas ocultas

3. Capa de salida

CAPAS DE UNA RED NEURONAL

Capa de entrada

- Recibe directamente las variables de entrada (x_1, x_2, ..., x_n).
- Cada nodo representa una característica del conjunto de datos.
- No realiza transformaciones, solo entrega los datos a la siguiente capa.

Capas ocultas

- Una o varias capas intermedias entre la entrada y la salida.
- Cada neurona realiza el siguiente cálculo:

$z = \Sigma(w \cdot a) + b$

$a = \sigma(z$

Donde:

- w: pesos
- a: activaciones de la capa anterior
- b: sesgo (bias)
- σ(z): función de activación (ReLU, Sigmoid, tanh, etc.)

Capas de salida

- Entrega la predicción de la red ($\hat{y}$).
- Su forma depende del tipo de tarea:

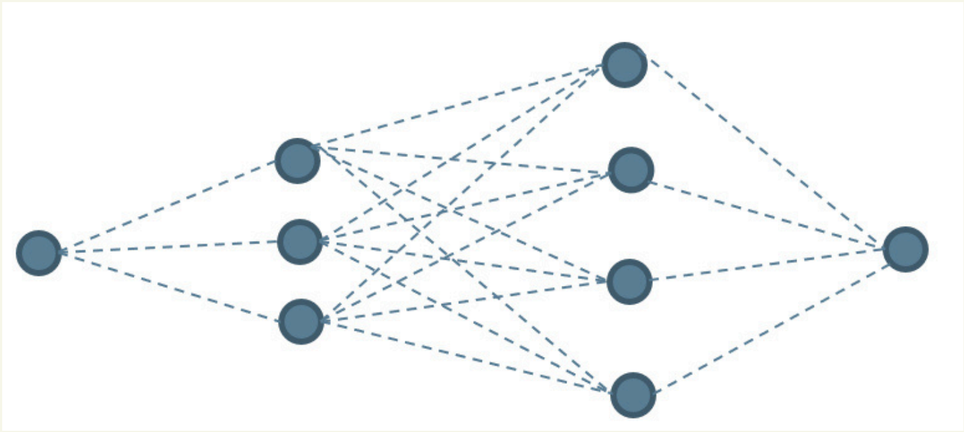
- Regresión: 1 neurona con activación lineal

- Clasificación binaria: 1 neurona + Sigmoid

- Clasificación multiclase: softmax sobre n neuronas

1. FORWARD PASS

calcular las salidas de la red y el costo (función de pérdida) para un ejemplo de entrenamiento.



Pre-activacion :

$$z^L_j = \sum_k w^L_{jk} \cdot a^{l-1}_k + b^L_j$$

Activacion :

$$a^L_j = \sigma(z^L_j)$$

donde σ es una función de activación (sigmoide, ReLU, etc.)

Ejemplo:

con una sola neurona de salida (coste cuadrático medio):

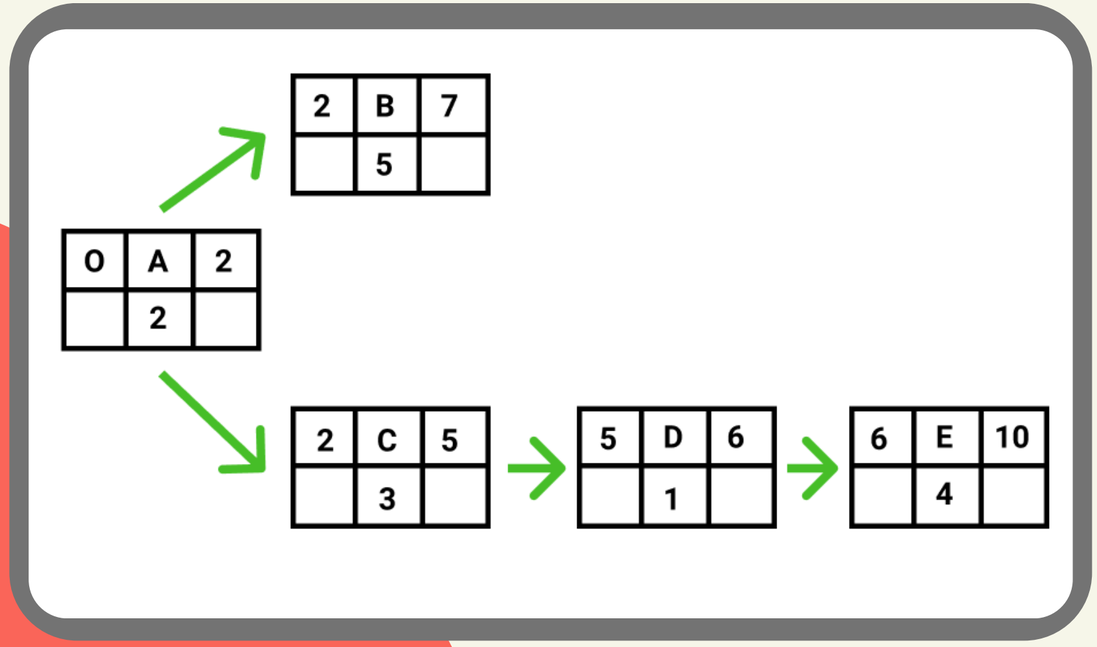
Coste:

$$C = \frac{1}{2}(aL - y)^2$$

- Calculamos z y luego a capa por capa.
- Al llegar a la salida, se evalúa el costo (error) con respecto al valor deseado y.

2. BACKWARD PASS

Determinar cuánto afectó cada neurona de la capa de salida al error final.



FORMULAS

Error en la capa de salida :

$$\delta^L = \partial C/\partial a^L \cdot \sigma'(z^L)$$

Costo cuadratico medio

$$\delta^L = (a^L - y) \cdot \sigma'(z^L)$$

- δ^L mide la “culpa” de cada neurona de salida en el error total.
- Es el punto de partida para propagar errores hacia atrás.

3. COSTO (FUNCIÓN DE PÉRDIDA)

Error cuadrático medio para una sola muestra

$$C = \frac{1}{2}\sum_j (a_j^L - y_j)^2$$

Cross-Entropy

$$C = -\sum_j [y_j \log a_j^L + (1 - y_j) \log(1 - a_j^L)]$$

BACKWARD PASS (RETROPROPAGACIÓN)

Objetivo:

Calcular

- $\frac{\partial C}{\partial w_{jk}^{l}}$
- $\frac{\partial C}{\partial b_{j}^{l}}$

Usando la regla de la cadena de cálculo diferencial.

Paso 1: Error de la capa de salida

Definimos el "error" δjL de la neurona j en la última capa L:

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

Este es el inicio del flujo hacia atrás. Se mide cuánto contribuye cada salida al error final.

Paso 2: Error en capas ocultas

Para capas l = L-1, L-2, ..., 2l=L-1,L-2,...,2:

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

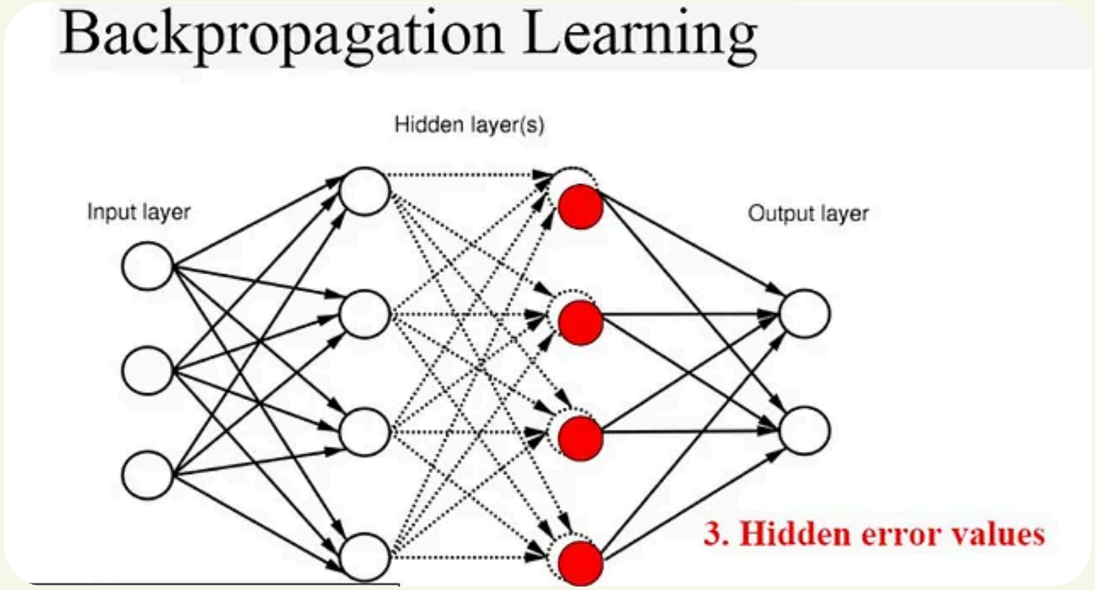
Se propaga el error hacia atrás usando los pesos y la derivada de la activación.

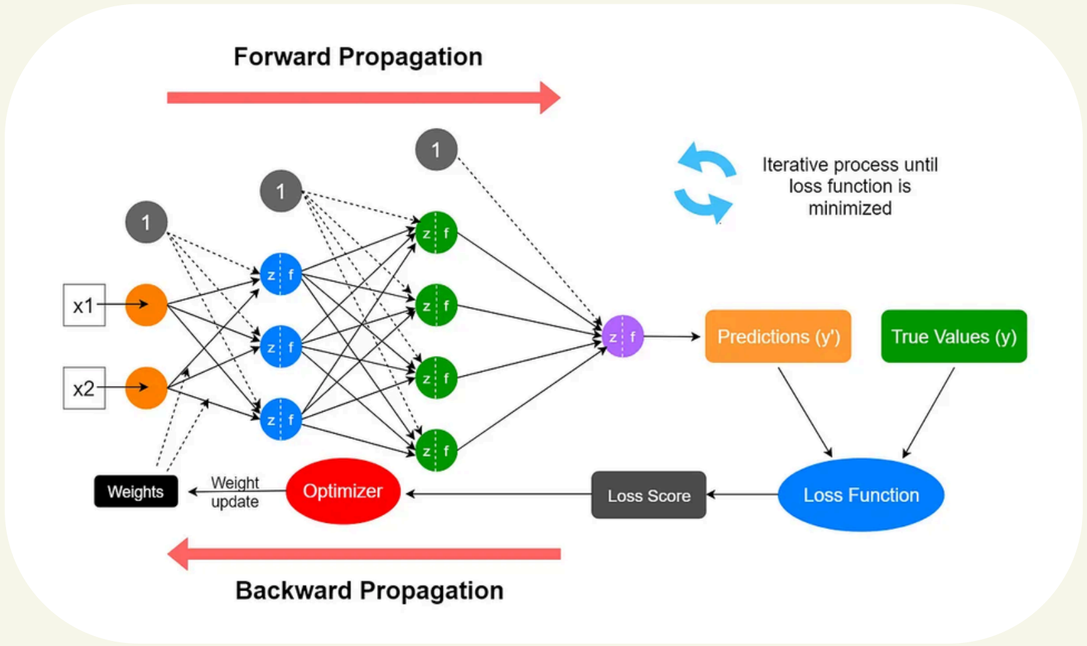
Gradientes para pesos y biases

$$\frac{\partial C}{\partial w_{jk}^{l}} = a_k^{l-1} \cdot \delta_j^l \quad , \quad \frac{\partial C}{\partial b_j^l} = \delta_j^l$$

5. FLUJO COMPLETO DEL ALGORITMO

Resumen esquemático del funcionamiento interno de una red neuronal durante su entrenamiento. Divide el proceso en dos fases:





Con esto podemos entender y visualizar:

- Cómo los errores se propagan hacia atrás (para saber cómo ajustar los pesos).
- Qué variables se calculan en cada paso.
- Qué ecuaciones se deben aplicar.

Este flujo nos permite calcular gradientes que usamos para actualizar los pesos y los sesgos durante el entrenamiento.

5. FLUJO COMPLETO DEL ALGORITMO

Forward Pass (Propagación hacia adelante):

$a^0 \rightarrow z^1 \rightarrow a^1 \rightarrow z^2 \rightarrow a^2 \rightarrow \ldots \rightarrow z^{\wedge}L \rightarrow a^{\wedge}L \rightarrow C$

- a^0: entrada (input).
- z^l: suma ponderada de la entrada en la capa l ($z^l = w^l \cdot a^{l-1} + b^l$).
- a^l: activación de la capa l ($a^l = \sigma(z^l)$, donde σ es la función de activación como sigmoid, ReLU, etc.).
- C: coste (error) del output a^L respecto al valor esperado y.

Esto representa cómo se propagan los datos de entrada por la red para producir una salida.

Backward Pass (desde L hasta 1)

$\partial C/\partial a^{\wedge}L \rightarrow \delta^{\wedge}L$

$\delta^{\wedge}L \rightarrow \delta^{\wedge}\{L-1\} \rightarrow \ldots \rightarrow \delta^1$

$\delta^{\wedge}l \rightarrow \partial C/\partial w^{\wedge}l$ y $\partial C/\partial b^{\wedge}l$

- $\partial C/\partial a^{\wedge}L$: sensibilidad del costo con respecto a la salida (output).
- $\delta^{\wedge}L$: "error" en la capa de salida (también se llama delta), que es:

$\delta^{\wedge}L = \partial C/\partial a^{\wedge}L \odot \sigma'(z^{\wedge}L)$

- $\delta^{\wedge}l$: error en una capa intermedia, propagado hacia atrás:

$\delta^{\wedge}l = (w^{\wedge}\{l+1\})^T \cdot \delta^{\wedge}\{l+1\} \odot \sigma'(z^{\wedge}l)$

EJEMPLO NUMÉRICO SENCILLO (UNA SOLA NEURONA EN CADA CAPA)

Supongamos:

- $a^{L-1} = 0.6$
- $w^L = 0.8$
- $b^L = 0.1$
- $\sigma(z) = \text{sigmoid}(z)$
- $y = 1$

Entonces:

1. $z^L = w^L a^{L-1} + b^L = 0.8 \cdot 0.6 + 0.1 = 0.58$
2. $a^L = \sigma(0.58) \approx 0.64$
3. $C = (a^L - y)^2 = (0.64 - 1)^2 \approx 0.1296$
4. $\delta^L = (a^L - y) \cdot \sigma'(z^L)$

$$\sigma'(z^L) = \sigma(z^L)(1 - \sigma(z^L)) = 0.64 \cdot (1 - 0.64) \approx 0.2304$$

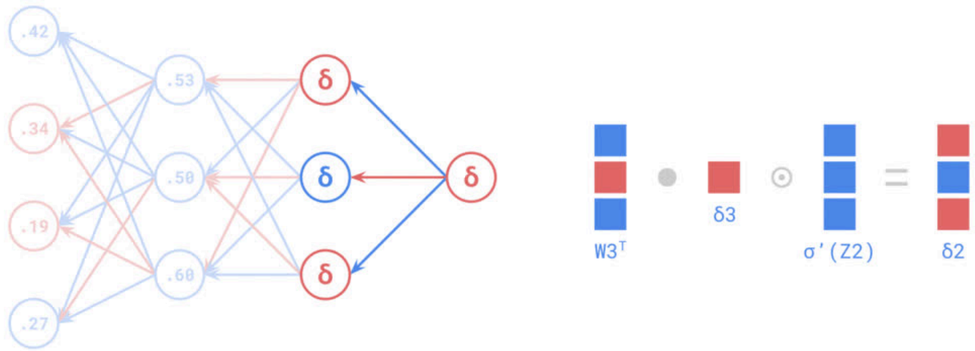
$$\delta^L = (0.64 - 1)(0.2304) \approx -0.0833$$

Su gradiente de peso seria:

$$\frac{\partial C}{\partial w^L} = a^{L-1} \cdot \delta^L = 0.6 \cdot (-0.0833) \approx -0.05$$

NOTACIÓN GENERAL

Símbolo	Significado
z_j^l	Entrada ponderada a la neurona j en capa l
a_j^l	Activación de la neurona j en capa l
w_{jk}^l	Peso de neurona k de capa $l-1$ a neurona j de capa l
b_j^l	Bias de la neurona j en capa l
δ_j^l	Error local en la neurona j de capa l
$\sigma(z)$	Función de activación



https://medium.com/the-feynman-journal/what-makes-backpropagation-so-elegant-657f3afbbd

CONCLUSIÓN

El algoritmo de backpropagation:

- Usa la regla de la cadena para descomponer cómo los pesos afectan el costo final.
- Flujo:
  - Hacia adelante para calcular activaciones y el costo.
  - Hacia atrás para propagar el error y calcular gradientes.
- Es la base del aprendizaje por descenso de gradiente en redes neuronales.

GRACIAS