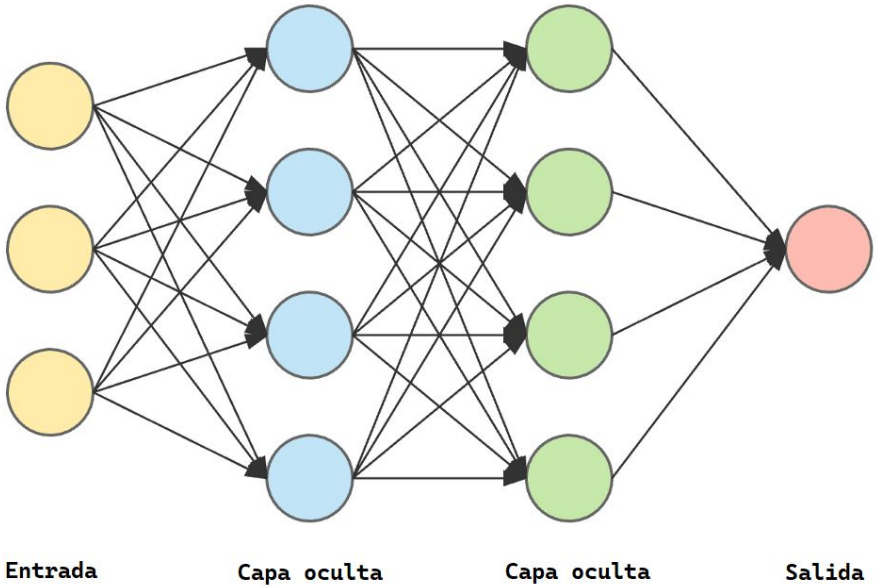
Redes neuronales

Redes Neuronales

Una red neuronal es un modelo simplificado que emula el modo en que el cerebro humano procesa la información: Funciona simultaneando un número elevado de unidades de procesamiento interconectadas que parecen versiones abstractas de neuronas.

Las unidades de procesamiento se organizan en capas. Hay tres partes normalmente en una red neuronal : **una capa de entrada**, con unidades que representan los campos de entrada; una o varias capas ocultas; y una capa de salida, con una unidad o unidades que representa el campo o los campos de destino. Las unidades se conectan con fuerzas de conexión variables (o ponderaciones). Los datos de entrada se presentan en la primera capa, y los valores se propagan desde cada neurona hasta cada neurona de la capa siguiente. al final, se envía un resultado desde la capa de salida.



Resumen

🧠 Estructura de la Red Neuronal

Nuestra red tiene **tres partes principales**:

1. **Capa de Entrada** (Input Layer)
   📌 Recibe las **tres variables de entrada**: Edad, Colesterol y Presión Arterial.
   📌 Cada dato se representa como un **número real**.

2. **Capas Ocultas** (Hidden Layers)
   📌 Transforman la información a través de **neuronas** que aplican **pesos y funciones de activación**.
   📌 Se usan funciones **ReLU** (Rectified Linear Unit) para aprender patrones no lineales.

3. **Capa de Salida** (Output Layer)
   📌 Una **única neurona** que usa la **función sigmoide** para producir una probabilidad entre **0 y 1**.
   📌 Si la salida es mayor a **0.5**, clasificamos como **"Enfermo"**, de lo contrario **"No Enfermo"**.

Comparación modelo anterior

- **Regresión Logística** → Trata de encontrar una **línea de separación** entre dos clases (enfermo / no enfermo).
- **Red Neuronal** → Aprende patrones **más complejos** con múltiples capas y neuronas.

Característica	Regresión Logística	Red Neuronal
Tipo de modelo	Modelo lineal	Modelo no lineal (con capas y neuronas)
Salida	Probabilidad (0 a 1)	Probabilidad (0 a 1) pero con más capas de procesamiento
Expresividad	Limitada a relaciones lineales	Puede capturar relaciones más complejas
Función de activación	Sigmoide	ReLU en capas ocultas, sigmoide en la salida
Aprendizaje	Algoritmo simple (gradiente descendente)	Algoritmo más complejo (backpropagation con optimizadores)

🚀 Paso a Paso: Cómo Procesa la Información

1. Propagación hacia adelante (Forward Propagation)

Cada **neurona** en la red toma las entradas, **las multiplica por pesos y suma un sesgo** (bias). Luego, la salida pasa por una **función de activación**.

1️⃣ **Primera capa oculta (16 neuronas con activación ReLU):**

$$H_1 = \text{ReLU}(w_{11}X_1 + w_{12}X_2 + w_{13}X_3 + b_1)$$

$$H_2 = \text{ReLU}(w_{21}X_1 + w_{22}X_2 + w_{23}X_3 + b_2)$$

ReLU convierte valores negativos en **cero** y mantiene los positivos.

Captura **patrones no lineales** en los datos.

🚀 Paso a Paso: Cómo Procesa la Información

Segunda capa oculta (8 neuronas con ReLU):

$$H_1' = \text{ReLU}(w_1' H_1 + w_2' H_2 + b')$$

Capa de salida (1 neurona con Sigmoide):

- La salida es la probabilidad de que el paciente esté enfermo:

$$P(\text{Enfermo} = 1) = \frac{1}{1 + e^{-(w_o + w_1 H_1' + w_2 H_2')}}$$

🔹 **2. Función de Activación**

📌 **ReLU (Rectified Linear Unit)** → Se usa en capas ocultas para introducir **no linealidad**.
📌 **Sigmoide** → Convierte la salida en una **probabilidad entre 0 y 1**.

Supongamos un paciente nuevo

- **Edad**: 50 años
- **Colesterol**: 220
- **Presión Arterial**: 130

📌 **Predicción con Regresión Logística:**

1. Calculamos Z:

$$Z = (-8) + (0.02 \times 50) + (0.03 \times 220) + (0.01 \times 130) = 0.9$$

2. Aplicamos sigmoide:

$$P = \frac{1}{1 + e^{-0.9}} = 0.71$$

Conclusión: El paciente tiene **71% de probabilidad** de estar enfermo → **"Enfermo"**.

📌 **Predicción con la Red Neuronal:**

1. La información pasa por **capas ocultas** donde los pesos son ajustados dinámicamente.
2. La capa de salida aplica sigmoide y da una probabilidad:

$$P = 0.85$$

Conclusión: La red neuronal estima una **85% de probabilidad** → **"Enfermo"**.

Redes Neuronales

Ahora vamos a construir un **ejemplo de una red neuronal** para **clasificar si una persona tiene una enfermedad o no**, utilizando **TensorFlow y Keras**.

🚀 ¿Qué Haremos?

1. **Generamos datos sintéticos** con variables como edad, colesterol y presión arterial.
2. **Construiremos una red neuronal** con TensorFlow/Keras.
3. **Entrenaremos el modelo** y lo evaluaremos.
4. **Haremos predicciones para nuevos pacientes**.

Collab

https://drive.google.com/file/d/1ImfRs23I60g9T2FqXQcfod-P0pWR3NMi/view?usp=sharing

Backpropagation (retropropagación) y **Gradient Descent (descenso del gradiente)**

Forward Propagation (Propagación hacia adelante)

- Calculamos la salida de la red neuronal multiplicando los datos de entrada por los pesos y aplicando funciones de activación.

Error Calculation (Cálculo del Error)

- Comparamos la predicción con la salida real (etiqueta) y calculamos el error con una función de pérdida (Ej: **Binary Crossentropy** en clasificación binaria).

Backward Propagation (Retropropagación)

- El error se distribuye hacia atrás en la red para ajustar los pesos.
- Se usan derivadas parciales para calcular **cómo cada peso contribuyó al error**.

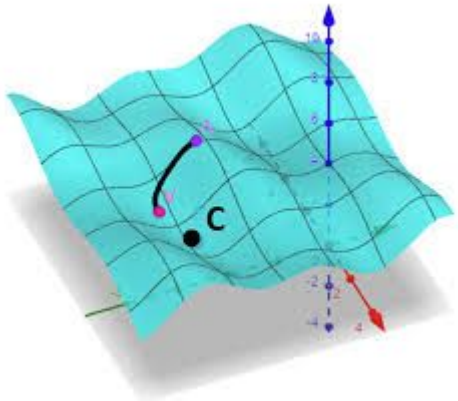
Descenso del gradiente

Gradient Descent (Descenso del Gradiente)

- Se ajustan los pesos en la dirección que **minimiza** el error.
- **Fórmula de actualización de pesos:**

$$W_{\text{nuevo}} = W_{\text{viejo}} - \alpha \cdot \frac{\partial J}{\partial W}$$

- W son los pesos de la red.
- α es la tasa de aprendizaje.
- $\frac{\partial J}{\partial W}$ es la derivada del error respecto al peso.



Ejemplo Numérico Sencillo

Supongamos una red neuronal **con 1 neurona** y **1 entrada**.

- **Entrada**: $X = 2$
- **Peso inicial**: $W = 0.5$
- **Salida esperada**: $Y_{\text{real}} = 1$
- **Función de activación**: Sigmoide
- **Función de pérdida**: Error cuadrático medio (MSE)

$$L = \frac{1}{2}(Y_{\text{real}} - Y_{\text{pred}})^2$$

Paso 1: Forward Propagation

1. **Calcular la salida de la neurona:**

$$Z = W \cdot X = 0.5 \times 2 = 1$$

2. **Aplicamos la función sigmoide:**

$$Y_{\text{pred}} = \frac{1}{1 + e^{-1}} = 0.73$$

Paso 2: Calcular el error

$$L = \frac{1}{2}(1 - 0.73)^2 = 0.0364$$

Paso 3: Calcular el error

Derivamos la función de pérdida con respecto al peso:

$$\frac{\partial L}{\partial W} = (Y_{\text{pred}} - Y_{\text{real}}) \cdot X \cdot f'(Z)$$

Donde $f'(Z)$ es la derivada de la sigmoide:

$$f'(Z) = Y_{\text{pred}}(1 - Y_{\text{pred}}) = 0.73 \times (1 - 0.73) = 0.197$$

$$\frac{\partial L}{\partial W} = (0.73 - 1) \times 2 \times 0.197 = -0.107$$

Paso 4: Gradient Descent (Actualizar el Peso)

Si usamos una tasa de aprendizaje $\alpha = 0.1$:

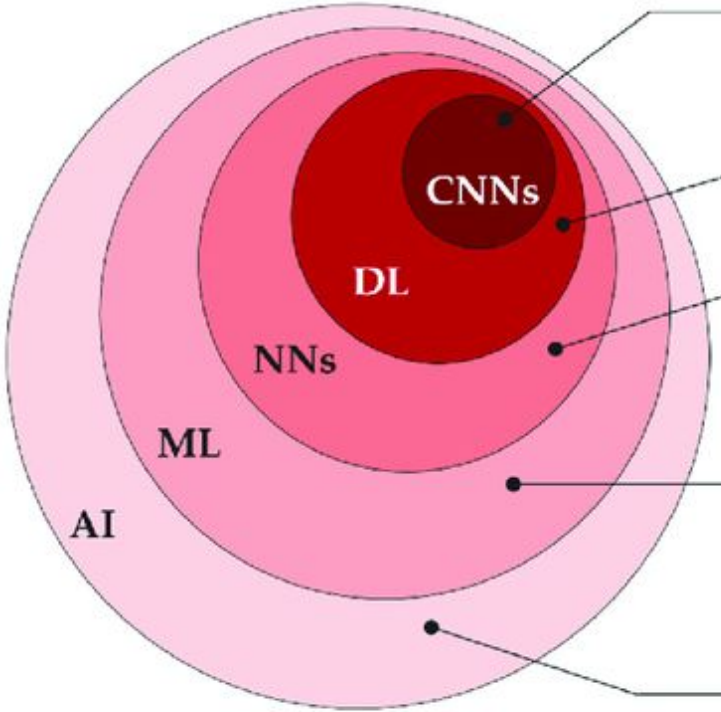
$$W_{\text{nuevo}} = W - \alpha \times \frac{\partial L}{\partial W}$$

$$W_{\text{nuevo}} = 0.5 - 0.1 \times (-0.107) = 0.51$$

🔹 **El peso aumentó de 0.5 a 0.51 porque el error nos decía que debíamos aumentarlo para mejorar la predicción.**

Deep Learning

Deep Learning



Convolutional Neural Networks (CNNs):
Deep learning models that use convolutional and pooling layers.

Deep Learning (DL):
Neural networks that use multiple hidden layers.

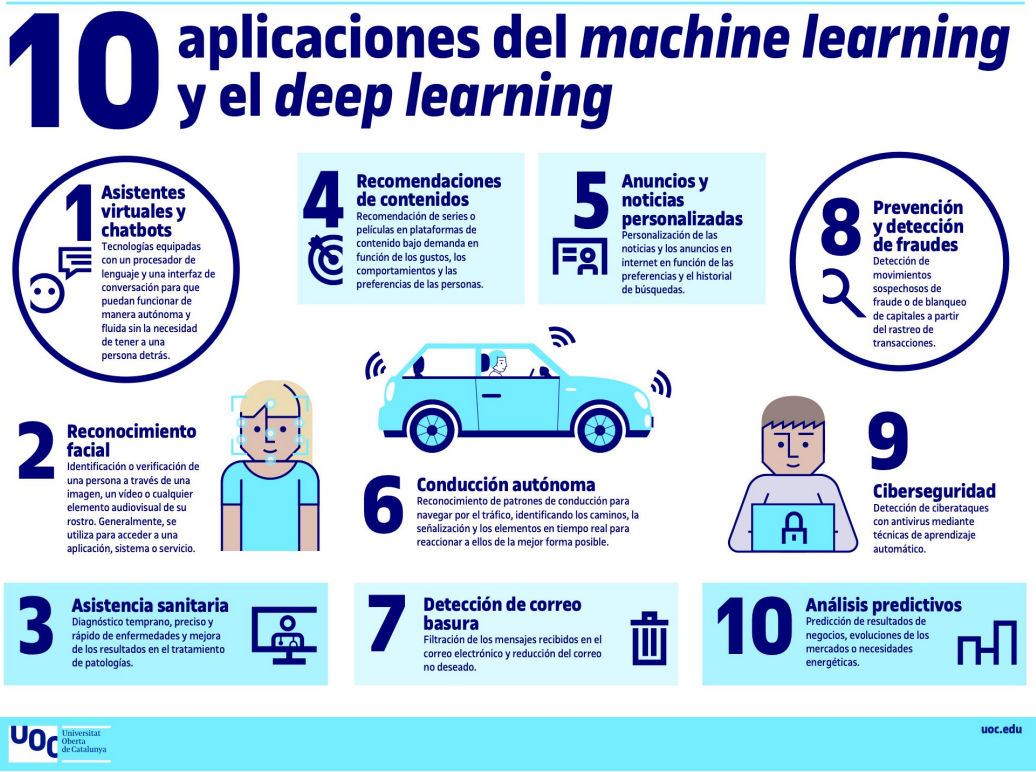
Neural Networks (NNs):
A subset of machine learning.
Network mimicking the human brain through a set of algorithms.

Machine Learning (ML):
A subset of artificial intelligence.
Computer uses statistical methods and computer science to predict outcomes without being explicitly programmed.

Artificial Intelligence (AI):
Broad term for machines, that mimic human intelligence.
Visual perception, speech recognition etc.

Deep Learning

La principal diferencia entre el deep learning y el machine learning es la estructura de la arquitectura de red neuronal subyacente. Los modelos tradicionales de machine learning “no profundos” utilizan redes neuronales simples con una o dos capas computacionales. Los modelos de deep learning utilizan tres o más capas, pero normalmente cientos o miles de capas, para entrenar los modelos.



Collab

https://colab.research.google.com/drive/1A4Xp8gva_S2_YXOuBkjHftOv7e52TFNd?usp=sharing