Fundamentos de Data Science con Python

Sesión 4: Introducción al Deep Learning

Dr. Julio Lopez-Nunez

Diciembre, 2025

Objetivos de la sesión.

Comprender los fundamentos de las redes neuronales artificiales y su relación con el aprendizaje automático tradicional.

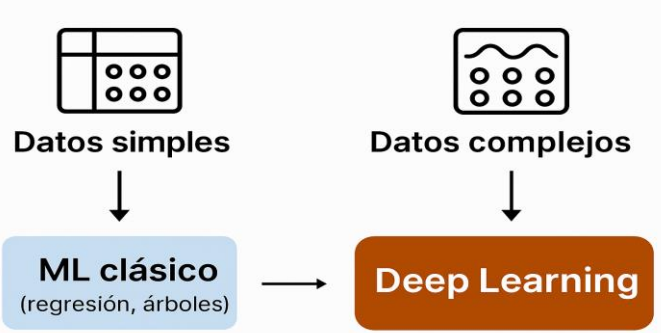
Conceptos clave de Deep Learning: neuronas, capas, activaciones.

Perceptrón.

Red neuronal multicapa (MLP).

¿Por qué necesitamos Deep Learning?

- Limitaciones del Machine Learning tradicional.
- Deep Learning como evolución natural
- **¿Cuándo DL es útil?**
- Cuando los datos son complejos (imágenes, texto, audio, señales).
- Cuando un modelo clásico no logra separar bien las clases o predecir con precisión.
- Cuando necesitamos capturar patrones profundos a varias capas.



Anatomía de una red neuronal

¿Cómo funciona una red neuronal?

Una neurona artificial

Recibe valores de entrada.

Multiplica cada valor por un peso.

Suma todo y agrega un bias.

Aplica una función de activación (ej.: ReLU, Sigmoid).

Capas conectadas

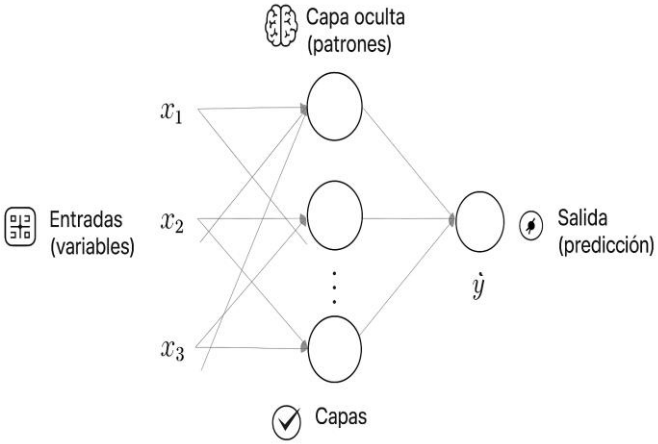
Capa de entrada:
recibe las variables del dataset.
Capas ocultas:
combinan pesos → capturan patrones cada vez más complejos.
Capa de salida:
produce la predicción (ej.: 0/1 para aprobado/reprobado).

Entrenamiento (backpropagation)

La red compara su predicción con la respuesta real.

Calcula el error → ajusta los pesos para mejorar.

Repite el proceso durante varias épocas.



El Perceptrón

Unidad básica de una red neuronal.

Practica: https://playground.tensorflow.org/

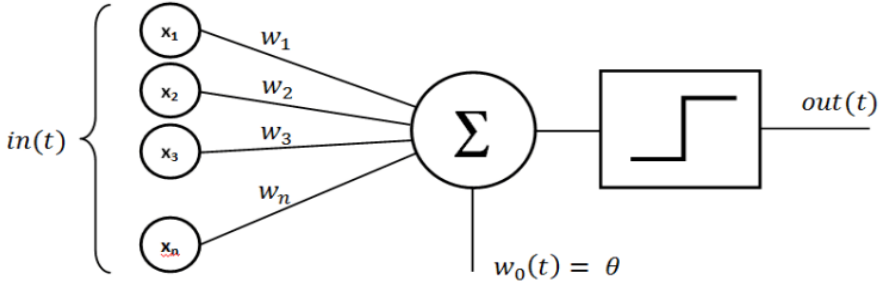
¿Qué es una neurona artificial?: Una función que recibe entradas ***(x_1, x_2, ...)***, las combina con pesos ***(w_i)***, suma un sesgo ***(b)*** y aplica una función de activación.

¿Para qué sirve la activación?: Para introducir no linealidad, permitiendo que la red aprenda patrones complejos.

Capas ocultas: Una red neuronal combina muchas neuronas en capas sucesivas, donde cada capa aprende representaciones más profundas del problema.

$$z = w_1 x_1 + w_2 x_2 + \cdots + b$$

$$\hat{y} = \text{activación}(z)$$



Funciones de activación.

Permite que la red aprenda patrones no lineales.

Sin activación, la red sería solo una regresión lineal.

ReLU (la más usada):

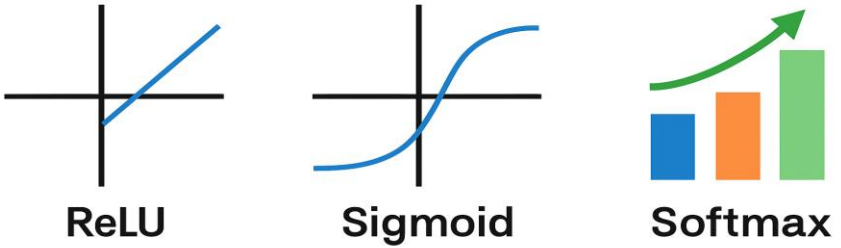
Rápida
Simple
Funciona muy bien con redes profundas
Se usa en capas ocultas

Sigmoid:

Útil para clasificación binaria
Produce valores entre 0 y 1
Interpretable como probabilidad

Softmax:

Transforma varias salidas en probabilidades que suman 1.
Se usa en redes que clasifican múltiples categorías
Ideal para MNIST (10 dígitos)



Ref.:
https://web.archive.org/web/20220817071122/https://mlfromscratch.com/activation-functions-explained/#/

¿Cómo “piensa” una red neuronal?

Ref.:
https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

MNIST

Un dataset de 70,000 imágenes de dígitos escritos a mano (0–9).

Cada imagen es en escala de grises, tamaño 28×28 píxeles.

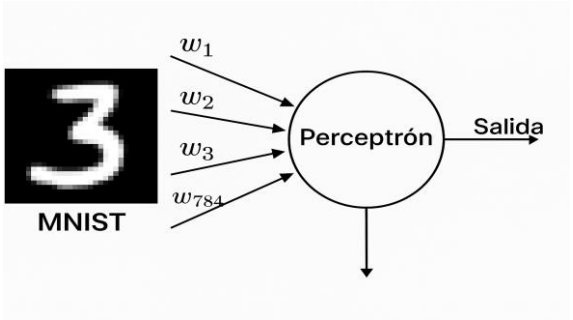
Muy usado para enseñar y probar modelos de Deep Learning, especialmente redes neuronales y convolucionales.

Las primeras capas detectan bordes y trazos simples.

Las capas posteriores combinan trazos → formas más completas.

La última capa decide el dígito → clasificación final.

Las redes aprenden representaciones



¿Cómo aprende una red neuronal?

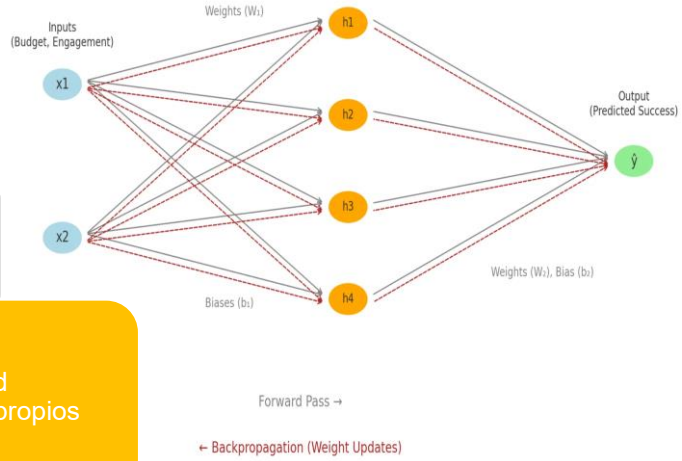
1) ¿Cómo decidir cuántas capas ocultas usar?
Problemas simples (datos tabulares): 1 capa oculta.
Problemas más complejos: 2–4 capas ocultas.
Problemas muy complejos (imágenes, audio, texto): muchas capas.
Más capas = más capacidad (riesgo de overfitting.)

Propagación hacia adelante (Forward Pass): La red toma los datos, los procesa capa por capa y produce una predicción.

Cálculo del error (Loss Function): Medimos que tan lejos estamos de la respuesta real.

Retropropagación (Backpropagation): La red aprende corrigiendo sus propios errores.

Repetición (Epochs):Cada vuelta hace que la red mejore un poco.



2) ¿Cuántas neuronas por capa?
Para datos tabulares:
Primera capa: entre 2× y 3× el número de entradas.
Capas siguientes: igual o menos neuronas.
Capa de salida: 1 neurona en regresión. n neuronas en clasificación.

Para imágenes (como MNIST):
La primera capa oculta suele tener muchas menos neuronas que entradas.
Capa de salida: 10 neuronas (una por cada dígito).
(La red aprende patrones, no pixeles individuales)

Métricas y Evaluación en Deep Learning.

Función de pérdida (Loss)

Indica qué tan bien o mal está aprendiendo la red.
Regresión → MSE (Error cuadrático medio)
Clasificación → Cross-entropy (entropía cruzada)
“La red intenta minimizar esta función.”

Accuracy

Proporción de predicciones correctas.
Ideal para clasificación (como MNIST).

Train vs Test

Train: qué tan bien aprende con los datos conocidos
Test: qué tan bien generaliza a datos nuevos
“Si el modelo va muy bien en train y mal en test → OVERFITTING.”

Overfitting & Underfitting

Overfitting: el modelo memoriza
Underfitting: aprende demasiado poco
Soluciones comunes: (Más datos, Regularización, Dropout, Menos capas / menos neuronas)

¿Cómo aprenden las redes neuronales?

Descenso del Gradiente (Gradient Descent)

El objetivo del entrenamiento es minimizar la función de pérdida (loss).
La red neuronal ajusta sus pesos y sesgos para reducir ese error.

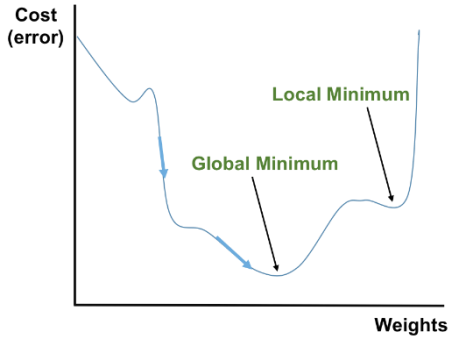
¿Cómo decide en qué dirección mover esos pesos?
Usando el gradiente, que indica “hacia dónde baja más rápido” la pérdida.

- Calcula el gradiente del error
- Actualiza los pesos
- Repite miles de veces

Gradiente: dirección de máximo aumento de la pérdida.

Elementos clave:
1.- El modelo se mueve en la dirección opuesta al gradiente.
2.- Cada actualización mejora ligeramente la predicción.

Es como bajar una montaña con niebla: avanzas paso a paso siguiendo la pendiente más pronunciada hacia abajo.



Ref.: https://youtu.be/IHZwWFHWa-w?si=cO7Wn2y3YyYOj3ig

Componentes del Descenso del Gradiente

(Learning Rate, Iteraciones y Backpropagation)

Learning Rate (α):

- Tamaño del paso que da el modelo al actualizar los pesos.
- Muy grande → el modelo "salta" y no aprende.
- Muy pequeño → aprende muy lento o queda atrapado.

Epochs / Iteraciones:

- Una epoch = el modelo ve todo el dataset una vez.
- El descenso del gradiente ocurre muchas veces durante el entrenamiento

Backpropagation (retropropagación):

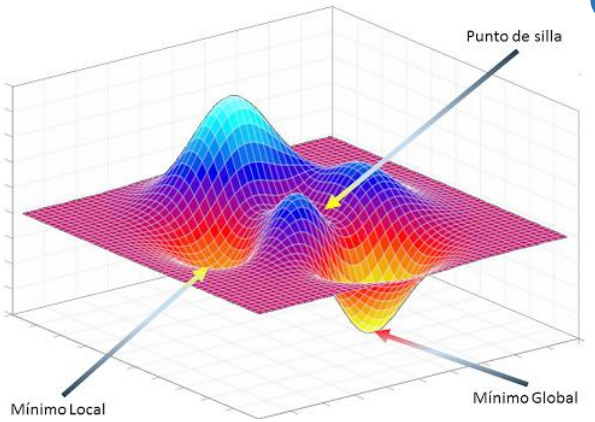
Algoritmo que calcula el gradiente desde la salida hacia las capas internas.

Permite saber qué peso debe ajustarse y cuánto.

Es el "motor matemático" que permite al gradiente funcionar.

No lo veremos en detalle matemático → solo la idea conceptual.

Entrenar una red: repetir miles de actualizaciones pequeñas usando el gradiente hasta que la pérdida se minimiza.



Cierre y próximos pasos.

Hoy aprendimos

1. Qué es una red neuronal
2. El perceptrón como unidad básica
3. Capas ocultas y funciones de activación
4. Forward pass, error y backpropagation
5. Cómo elegir capas y neuronas
6. Métricas de evaluación

Próxima sesión

1. Introducción a redes más profundas
2. Primer modelo práctico en Colab
3. Uso de MNIST o dataset educativo
4. Ajuste de hiperparámetros simples
5. Visualización durante entrenamiento

Preguntas de reflexión

¿Qué entendemos ahora por “aprendizaje”?

.

¿Por qué decimos que el proceso es “casi artesanal”?