Clase 100 — Perceptrón, MLP y backpropagation

Parte: 2 — Deep Learning · Fuente: Géron, cap. 10 § From Biological to Artificial Neurons.

Duración estimada: 80 min.

Clase 100 — Perceptrón, MLP y backpropagation

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 10 § From Biological to Artificial Neurons. Duración estimada: 80 min.*

Objetivo

Que el alumno entienda la unidad fundamental del Deep Learning: la neurona artificial (perceptrón de Rosenblatt 1957), por qué un solo perceptrón no puede aprender XOR, cómo el MLP (Multi-Layer Perceptron) lo resuelve apilando capas con activaciones no lineales, y cómo backpropagation (regla de la cadena) permite calcular gradientes en cualquier grafo computacional — el algoritmo que destrabó el Deep Learning moderno.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Implementar a mano un perceptrón (y = step(w·x + b)) y mostrar por qué no separa XOR.
- Construir un MLP con keras.Sequential([Dense(...), Dense(...)]) y entrenarlo sobre un dataset simple.
- Explicar forward pass (compute layer by layer) y backward pass (propagar gradientes con la regla de la cadena).
- Calcular a mano los gradientes para un MLP de 2 capas con una sola muestra.
- Reconocer que la diferenciación automática (autograd) hace innecesario derivar a mano para modelos arbitrarios.

Temas

#	Tema	Por qué importa
1	Perceptrón clásico (Rosenblatt 1957)	La unidad atómica; ver sus límites motiva
2	El problema XOR	Por qué necesitamos no linealidad y profun
3	MLP: input → hidden → output	La arquitectura mínima útil.
4	Activación no lineal	Sin activación, N capas = 1 capa lineal.
5	Backpropagation (Rumelhart, Hinton & Willi	La regla de la cadena aplicada a un grafo.
6	Autograd / autodiff	Lo que hace que TF/PyTorch/JAX te liberen

Definiciones y características

- Perceptrón: $y = \text{step}(\Sigma\, w_i \cdot x_i + b)$. Regla de aprendizaje: $w_i \leftarrow w_i + \eta \cdot (y_{true} - y_{pred}) \cdot x_i$. Converge si los datos son linealmente separables.
- MLP (Multi-Layer Perceptron): red feedforward fully-connected con ≥ 1 capa oculta y activaciones no lineales (sigmoid, tanh, ReLU).
- Forward pass: $a^l = \sigma(W^l \cdot a^{l-1} + b^l)$. Se compute layer-by-layer hasta la salida.
- Loss function: L(y_pred, y_true) — MSE para regresión, cross-entropy para clasificación.
- Backward pass: aplica la regla de la cadena para calcular $\partial L/\partial W^l$ propagando el gradiente de la salida hacia la entrada.

- Autograd: TF (GradientTape), PyTorch (backward()), JAX (grad) — construyen el grafo y aplican backprop automáticamente.
- Universal Approximation Theorem (Cybenko 1989, Hornik 1991): un MLP con una sola capa oculta y "suficientes" neuronas puede aproximar cualquier función continua. En la práctica, profundo es más eficiente que ancho.

Dataset / recursos

- sklearn.datasets.make_moons(n_samples=500, noise=0.2) — clásico para mostrar no linealidad.
- XOR como dataset de 4 puntos (ejercicio motivacional).
- Librerías: tensorflow, keras (incluido), numpy, matplotlib.

Ejercicios

1. Perceptrón a mano: implementá un perceptrón en numpy y entrenalo en AND/OR (separables). Después probá con XOR; mostrá que no converge.
2. MLP con Keras: model = keras.Sequential([Dense(8, activation='relu', input_shape=(2,)), Dense(1, activation='sigmoid')]). Entrenalo sobre XOR; debería llegar a accuracy 1.0.
3. Visualización decision boundary: con make_moons, entrená un MLP [16, 8] y graficá la frontera con un meshgrid.
4. Backprop a mano: para un MLP con 1 input, 1 hidden (2 neuronas), 1 output, con MSE, calculá $\partial L/\partial w$ para una muestra y comparalo con tf.GradientTape.
5. ¿Y sin activación no lineal?: cambiá las activaciones a linear en el MLP de XOR y mostrá que ya no puede aprenderlo.

Homework verificable

Notebook que:

1. Carga make_moons(noise=0.2, random_state=42).
2. Entrena 2 modelos: regresión logística (Parte 1) vs MLP [16, 8] con ReLU.
3. Reporta accuracy en test para ambos.
4. Grafica las dos decision boundaries lado a lado.
5. Explica en 3 líneas por qué el MLP captura la curvatura y la regresión logística no.

Criterio de aceptación: el MLP debe lograr ≥ 0.95 accuracy y la frontera debe ser claramente curva; la regresión logística queda en ≈ 0.85 con frontera recta.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
MLP no aprende XOR aunque la arquitectura	Probablemente sin activación no lineal, o
Loss nan desde la primera época	Inicialización mala + activación + LR. Fix
Modelo sigmoid de salida con loss='mse'	Lento y mal calibrado. Fix: loss='binary_c
input_shape mal especificado: (2,) vs 2	Tiene que ser tupla. Fix: Dense(8, input_s
Modelo memoriza train y mal en test con XO	XOR son 4 puntos — no hay nada para genera

Preguntas frecuentes

¿Una sola neurona equivale a regresión logística?

Sí, exactamente: sigmoid(w·x + b) con cross-entropy = logistic regression. Por eso un MLP "es regresión logística apilada con no linealidades en medio".

¿Por qué no se podía entrenar redes profundas hasta los 2000s?

Por tres problemas: vanishing gradients (sigmoid satura), datasets chicos, hardware limitado. Lo resolvieron ReLU (2010), GPUs y ImageNet (2012). Lo verás en Clase 096.

¿Cuántas capas y neuronas pongo?

Empezá chico: 1-2 capas ocultas, 16-128 neuronas. Si underfittea, ensanchá o profundizá. Si overfittea, achicá o regularizá. La arquitectura se busca empíricamente con Keras Tuner (Clase 095) u Optuna.

¿Qué es el "Universal Approximation Theorem" en la práctica?

Que existe un MLP que aproxima tu función. No dice nada sobre cuán difícil es encontrarlo (el problema de optimización). Por eso "1 capa muy ancha" en teoría alcanza, pero "10 capas más angostas" en práctica converge mucho mejor.

¿Forward + backward = una época?

No. Una iteración (= un batch) es 1 forward + 1 backward + 1 update. Una época es haber visto todo el dataset una vez (= n_samples / batch_size iteraciones).

Referencias

- Géron, cap. 10 — Introduction to Artificial Neural Networks with Keras.
- Rumelhart, Hinton & Williams (1986), Learning representations by back-propagating errors, Nature — paper original de backprop.
- Goodfellow, Bengio & Courville (2016), Deep Learning, cap. 6 (online: <https://www.deeplearningbook.org/>).
- 3Blue1Brown — serie "But what is a neural network?" (4 videos, intuición visual de backprop).
- Keras docs — Sequential model.

Siguiente clase

Clase 101 — Regresión y clasificación con MLP

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb