Día 02 · Tema 1 — Fundamentos: neurona, MLP y backpropagation

Eduardo C. Garrido-Merchán · ecgarrido@comillas.edu Universidad Pontificia Comillas · ICADE

1. Una neurona, simbólicamente

La unidad de cómputo elemental de toda red es la neurona artificial. Dado un vector de entradas $x \in \mathbb{R}^d$, una neurona calcula

$$y \;=\; \sigma\Big(\sum_{i=1}^{d} w_i\, x_i \;+\; b\Big) \;=\; \sigma\big(w^\top x + b\big). \tag{1}$$

- $x_i \in \mathbb{R}$ i-ésima feature de entrada
- $w_i \in \mathbb{R}$ peso aprendible asociado a x_i
- $b \in \mathbb{R}$ sesgo (bias) aprendible
- σ activación no lineal: ReLU, sigmoide, GELU…
- y salida escalar de la neurona

La activación más usada hoy es **ReLU**, $\sigma(z) = \max(0, z)$, porque no satura para $z > 0$ y se deriva trivialmente.

2. Forward sobre un cliente real (ejemplo a mano)

Sea un cliente telco con *tenure* 24, *monthly* 80 € y *support_calls* 5. Con pesos $w = (-0{,}04,\ 0{,}02,\ 0{,}55)$ y $b = -0{,}50$, la combinación lineal y la activación sigmoide dan

$$z = -0{,}04 \cdot 24 + 0{,}02 \cdot 80 + 0{,}55 \cdot 5 + (-0{,}50) = 2{,}89,$$
$$\hat{y} = \sigma(z) = \tfrac{1}{1+e^{-2{,}89}} \approx 0{,}947.$$

La salida 0,947 se interpreta como probabilidad de *churn*: con umbral 0,5 el cliente se marca como prioritario para retención.

3. El MLP como composición de capas

Apilando L neuronas en capas obtenemos un *multilayer perceptron*:

$$f_\theta(x) \;=\; h_L \circ h_{L-1} \circ \cdots \circ h_1(x), \qquad h_\ell(z) = \sigma_\ell\big(W_\ell z + b_\ell\big). \tag{2}$$

- $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ matriz de pesos de la capa ℓ
- $b_\ell \in \mathbb{R}^{d_\ell}$ sesgos de la capa ℓ
- d_ℓ anchura de la capa ℓ
- L profundidad (número de capas)

Sin la no-linealidad σ_ℓ la composición colapsa a una afín única: la no-linealidad es lo que da expresividad.

4. Pérdida y descenso de gradiente

Para clasificación binaria usamos la cross-entropy:

$$\ell(\hat{y}, y) \;=\; -\big[\, y \log \hat{y} + (1-y)\log(1-\hat{y})\,\big]. \tag{3}$$

El optimizador actualiza θ con

$$\theta_{t+1} \;=\; \theta_t \;-\; \eta_t\, \nabla_\theta \hat{\mathcal{R}}_n(\theta_t), \qquad \hat{\mathcal{R}}_n(\theta) = \tfrac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(x_i), y_i). \tag{4}$$

- η_t tasa de aprendizaje (learning rate) en el paso t
- ∇_θ gradiente respecto a todos los parámetros
- $\hat{\mathcal{R}}_n$ riesgo empírico sobre n ejemplos

5. Backpropagation, regla de la cadena

Backprop calcula $\nabla_\theta \hat{\mathcal{R}}_n$ aplicando recursivamente la regla de la cadena. Definiendo $z_\ell = W_\ell a_{\ell-1} + b_\ell$ y $a_\ell = \sigma_\ell(z_\ell)$ (con $a_0 = x$), y la *señal de error* $\delta_\ell := \partial \ell / \partial z_\ell$, se cumple

$$\delta_L = \nabla_{a_L} \ell \;\odot\; \sigma_L'(z_L), \qquad \delta_\ell = (W_{\ell+1}^\top \delta_{\ell+1}) \;\odot\; \sigma_\ell'(z_\ell). \tag{5}$$

Y entonces $\partial \ell / \partial W_\ell = \delta_\ell \, a_{\ell-1}^\top$ y $\partial \ell / \partial b_\ell = \delta_\ell$.

- $\odot$ producto componente a componente (Hadamard)
- σ'_ℓ derivada de la activación, evaluada en z_ℓ
- a_ℓ activación de la capa ℓ
- δ_ℓ gradiente de la pérdida respecto a z_ℓ

Idea clave. En PyTorch nunca se escribe (5) a mano: se llama a `loss.backward()` y *autograd* aplica la cadena automáticamente.

6. Adam vs SGD: el optimizador moderno

SGD vanilla actualiza con la regla (4). **Adam** mantiene medias móviles del gradiente m_t y del cuadrado v_t, y corrige sesgo:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\, g_t, & v_t &= \beta_2 v_{t-1} + (1-\beta_2)\, g_t^2, \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t}, & \hat{v}_t &= \frac{v_t}{1-\beta_2^t}, \\ \theta_{t+1} &= \theta_t - \eta \, \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\varepsilon}. \end{aligned} \tag{6}$$

- g_t gradiente estocástico instantáneo $\nabla_\theta \ell$ del minibatch
- β_1, β_2 factores de decaimiento (0,9, 0,999)
- ε término de estabilidad numérica (10^{-8})

AdamW es la variante que el curso usa por defecto: añade *weight decay* desacoplado del gradiente, $\theta \leftarrow \theta - \eta\lambda\theta$, lo que regulariza sin distorsionar las medias móviles.

7. Métricas con clase desbalanceada

En *churn* y fraude, $1-2\,\%$ de positivos hacen que el accuracy mienta. Las métricas relevantes son

$$\text{Recall} = \frac{TP}{TP+FN}, \quad \text{Precision} = \frac{TP}{TP+FP}, \quad F_1 = \frac{2\,\mathrm{P}\cdot\mathrm{R}}{\mathrm{P}+\mathrm{R}}, \tag{7}$$

y **AUC-ROC** (área bajo TPR(FPR)). Antes de elegir modelo, fijar la métrica: si lo crítico es no perder churners, optimizar recall a costa de precisión.