Deep Learning para Business Analytics

Día 02 · Tema 1 — Fundamentos rápidos

Del perceptrón a la red profunda, con números a la vista.

Eduardo C. Garrido-Merchán

Profesor Propio Adjunto, Métodos Cuantitativos, ICADE

ecgarrido@comillas.edu

Madrid, Día 02

BBVA · Telefónica · NETFLIX · amazon · INDITEX · Iberdrola

ejemplos del día

COMILLAS UNIVERSIDAD PONTIFICIA ICAI ICADE CIHS

Lo que vamos a cubrir hoy.

1. Repaso rápido de Machine Learning

2. La neurona y el MLP

3. Backpropagation, con números

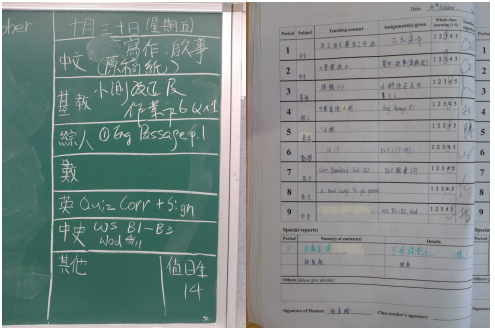
4. Optimizadores modernos

5. Caso del día: churn telco

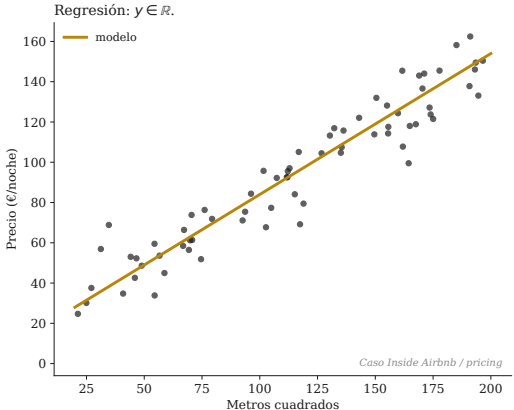
Lo que vas a saber hacer al final del día:

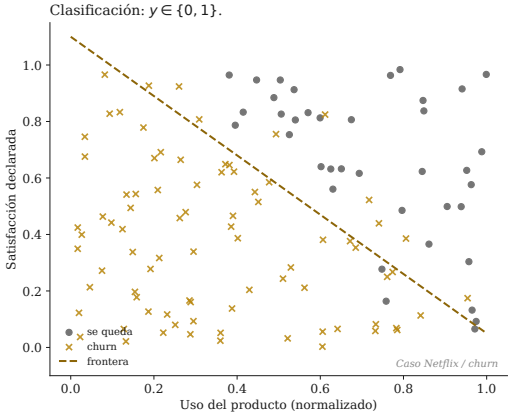
1. Distinguir cuándo es regresión y cuándo clasificación frente a una hoja de cálculo de negocio.
2. Calcular el forward y el backward de una neurona *a mano*, con números.
3. Elegir entre SGD y Adam y argumentarlo en un mail al director técnico.

Repaso rápido de Machine Learning

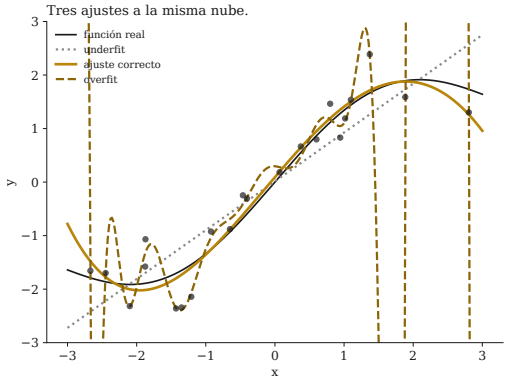


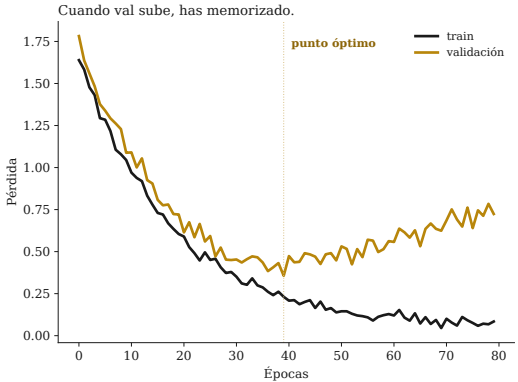
Regresión vs. clasificación, con dos casos de marca.





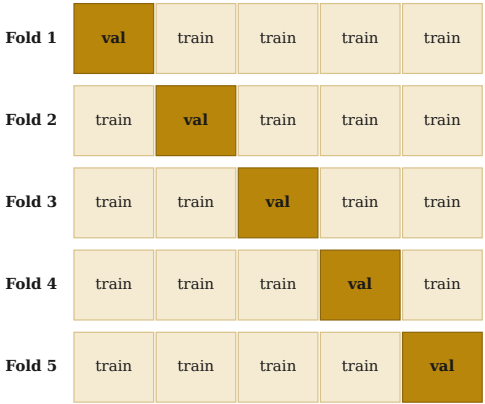
Overfitting: el enemigo número uno.





Validación cruzada k-fold: escudo contra suerte muestral.

Validación cruzada k=5: cada fold es validación una vez.



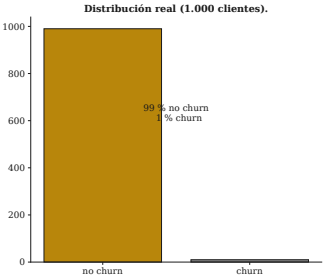
Para qué. 🛡️ Estimar la métrica con menos sensibilidad a la partición concreta.

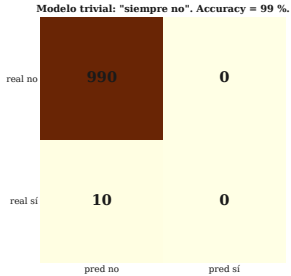
Cómo. ⚙️ El dataset se divide en k trozos. Cada trozo es la validación en su turno; los demás $k-1$ son entrenamiento. La métrica final es la media de las k ejecuciones.

Cuándo no. ⚠️ En series temporales (rolling window), en datasets enormes (holdout único basta).

Idea clave. Una sola partición train/test es ruidosa: úsala solo en gran volumen.

La trampa del accuracy en problemas desbalanceados.





La trampa. ⚠️ Si el 99 % son no-churn, un modelo trivial que siempre prediga "no"tiene 99 % de accuracy y **cero valor de negocio**.

Mejor. 📊 AUC, recall sobre la clase rara, F1 sobre la clase rara, expected value.

Aún mejor. 🎯 Definir antes el KPI de negocio (€ de pérdida evitada por TP, € de coste por FP) y optimizar la decisión contra eso.

Idea clave. Antes de elegir métrica, escribe la función de coste del negocio.

Concepto → aplicación: el ML clásico que ya sabéis.

Concepto	Aplicación de negocio	Entradas	KPI
Regresión lineal	Pricing dinámico de alquiler vacacional	Tamaño, ubicación, fecha	Error medio (€/noche)
Regresión logística	Scoring crediticio en banca minorista	Histórico de pagos, ingresos	AUC, Gini, € de mora evitada
k-NN / árboles	Recomendación de productos similares	Clicks, categoría, precio	Click-through rate
Random Forest	Detección de fraude en transacciones	Importe, hora, geo	Recall fraude, FP por día
Gradient Boosting	Forecast de demanda producto a producto	Histórico, promo, precio comp.	MAPE, días sin stock

Idea clave. Antes de saltar a redes, comprueba si una de estas resuelve ya tu problema.

Parte 2 / 5

La neurona y el MLP

Netflix predice lo que verás esta noche con una red neuronal.

Netflix: flujo simplificado de recomendación.

80 % del contenido visto proviene de la recomendación.

El negocio. 🚀 Netflix usa MLPs (entre otras arquitecturas) para ordenar su catálogo por usuario. El 80 % del contenido que se consume llega a través de la recomendación, no de la búsqueda.

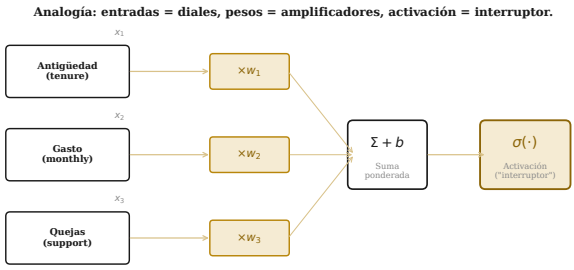
Impacto. 💰 Se estima que el sistema de recomendación evita **$1 B/año** en cancelaciones de suscripción.

NETFLIX Caso Netflix

La pregunta. 🧠 ¿Qué hay dentro de esa "caja negra"? Una neurona. Vamos a abrirla.

Idea clave. Una red neuronal es el motor detrás de cada recomendación que ves.

Una neurona decide con matices, no con sí/no.



Entradas. ⚙️ Cada entrada (x_i) es un dato medible del cliente: su antigüedad, su gasto mensual, sus llamadas al soporte.

Pesos. ⚖️ Cada peso (w_i) dice cuánto importa esa entrada. Un peso grande amplifica; un peso cercano a cero silencia.

Sesgo. 🎯 El sesgo (b) es el umbral base: "sin ninguna señal, ¿cuál es mi predicción por defecto?"

Activación. 🧠 La función $\sigma(\cdot)$ transforma la suma en algo no lineal: aplasta valores extremos, introduce curvas. Sin ella, la red sería una regresión lineal disfrazada.

Idea clave. Primero la intuición, luego la ecuación. Así se entiende cada símbolo.

Qué hace una neurona, formalmente.

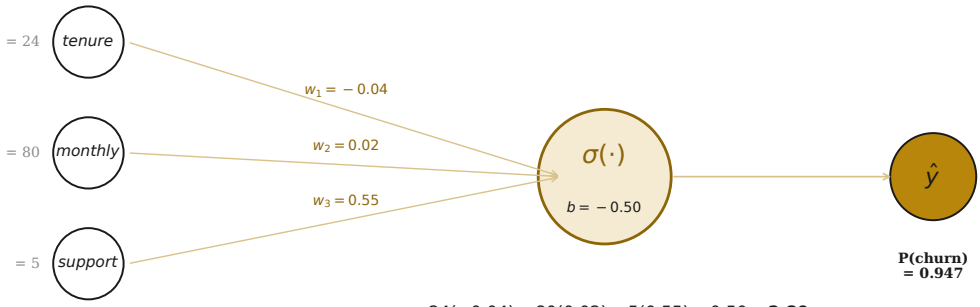
$$y = \sigma\left(\sum_{i=1}^{d} w_i\, x_i + b\right)$$

x_i	entradas (features).
w_i	pesos aprendidos durante el entrenamiento.
b	sesgo, también aprendido.
$\sigma(\cdot)$	activación no lineal (ReLU, sigmoide…).
y	salida de la neurona.

Idea clave. La no linealidad es lo que da poder; sin ella, una pila de capas se colapsa a una lineal única.

Ejemplo a mano: forward sobre un cliente real.

Una neurona, un cliente, predicción de churn.



$$z = 24(-0.04) + 80(0.02) + 5(0.55) - 0.50 = \mathbf{2.89}$$

$$y = \text{sigmoide}(2.89) = \mathbf{0.947}$$

Forward paso a paso, en texto.

Cliente. 🏦 Tres atributos observados: tenure = 24 meses, monthly = 80 €, support_calls = 5. La empresa quiere saber si churneará en el próximo trimestre.

Paso 1 — combinación lineal.

$$z = w_1x_1 + w_2x_2 + w_3x_3 + b = (-0{,}04)(24) + (0{,}02)(80) + (0{,}55)(5) + (-0{,}50) = \mathbf{2.89}.$$

Paso 2 — activación.

$$\hat{y} = \text{sigmoide}(z) = \frac{1}{1 + e^{-2{,}89}} = \mathbf{0.947}.$$

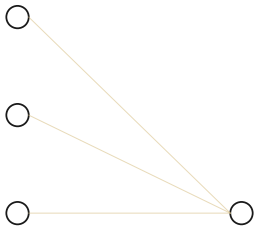
Paso 3 — interpretación de negocio. 💰 El modelo le asigna 94,7 % de probabilidad de churn. Con umbral 0,5 lo marcamos como cliente prioritario para una campaña de retención.

Idea clave. El forward de una red entera es exactamente esto, repetido capa a capa.

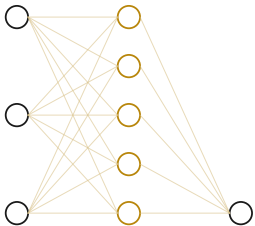
De una neurona a una red profunda.

De una neurona a millones: misma idea, más capas.

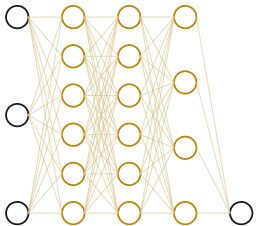
Perceptrón (1958)
3 → 1



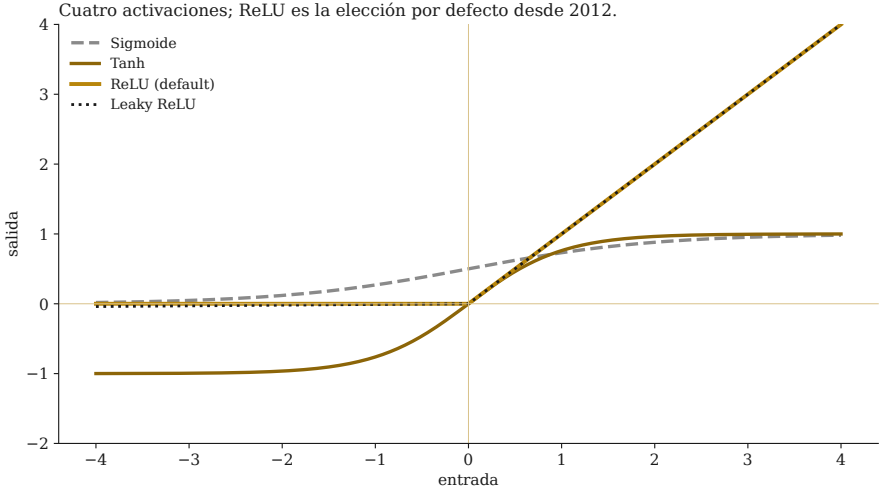
MLP (años 80)
3 → 5 → 1



Red profunda (2012-)
3 → 6 → 6 → 4 → 1



Cuatro activaciones; ReLU es la moderna por defecto.



Parte 3 / 5

Backpropagation, con números

El bucle del entrenamiento, en un diagrama.

Forward: predicción → pérdida.

Backward: gradientes → actualizar pesos.

El backward es una receta fija: siempre los mismos tres factores.

Los seis pasos, y dónde se aplica cada uno.

1. Haz el forward y guarda los valores.
Necesitarás cada z y cada a que has calculado.

2. Calcula el error en la neurona de salida.
$\delta = \hat{y} - y$

3. Gradiente de un peso de esa neurona.
$\frac{\partial \mathcal{L}}{\partial w} = \delta \cdot x$
x es lo que entra por ese peso.

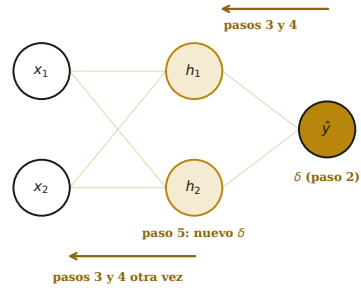
4. Gradiente del sesgo de esa neurona.
$\frac{\partial \mathcal{L}}{\partial b} = \delta \cdot 1 = \delta$
El sesgo se suma solo, no multiplica a ninguna entrada: su x vale 1.

5. Pasa el error a una neurona de la capa anterior.
$\delta \leftarrow \delta \cdot w \cdot f'(z)$
w es el peso por el que baja; z, el de esa neurona anterior.

6. Cuando ya tienes todos los gradientes, actualiza.
$w \leftarrow w - \eta \frac{\partial \mathcal{L}}{\partial w}$ y $b \leftarrow b - \eta \frac{\partial \mathcal{L}}{\partial b}$

Repite 3, 4 y 5 hacia atrás hasta la primera capa. El paso 6, al final y una sola vez.

Si una neurona alimenta a varias, en el paso 5 suma un $\delta \cdot w$ por cada una.



No cambias de fórmula: cambias de sitio.

gradiente $= \delta \times$ (lo que ese parámetro multiplicaba)

el peso multiplica a x; el sesgo, a 1

δ: el error que llega a esa neurona, o sea $\partial \mathcal{L}/\partial z$.

$f'(z)$: derivada de la activación (ReLU: 1 si $z > 0$, si no 0).

η: el learning rate.

Ejemplo a mano: backward sobre el mismo cliente.

Backpropagación a mano: cada derivada sale de la función de la izquierda.

1. La BCE (entropía cruzada binaria).

$$\mathcal{L} = -[\,y \log \hat{y} + (1-y)\log(1-\hat{y})\,]$$

De los dos sumandos sólo sobrevive el de la clase verdadera:

$y=1$: $\mathcal{L} = -\log \hat{y}$	$y=0$: $\mathcal{L} = -\log(1-\hat{y})$
$\frac{\partial \mathcal{L}}{\partial \hat{y}} = -\frac{1}{\hat{y}}$	$\frac{\partial \mathcal{L}}{\partial \hat{y}} = +\frac{1}{1-\hat{y}}$

Los dos casos a la vez, en una sola expresión:

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}} = \frac{\hat{y}-y}{\hat{y}(1-\hat{y})}$$

2. La sigmoide.

$$\frac{\partial \hat{y}}{\partial z} = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}} = \hat{y}\,(1-\hat{y})$$

porque $e^{-z}/(1+e^{-z}) = 1-\hat{y}$

3. Al componerlas se cancela el denominador.

$$\frac{\partial \mathcal{L}}{\partial z} = \frac{\hat{y}-y}{\hat{y}(1-\hat{y})} \cdot \hat{y}\,(1-\hat{y}) = \hat{\mathbf{y}} - \mathbf{y}$$

Es el error de predicción, sin más. Vale igual para $y=0$ y para $y=1$.

1. Salida del forward

$\hat{y} = 0.947, \quad y = 1$ (el cliente sí churnea)

2. Pérdida BCE, con $y=1$

$\mathcal{L} = -\log \hat{y} = -\log(0.947) = \mathbf{0.054}$

3. Gradiente respecto a $\hat{y}$ — aquí $y=1$

$\frac{\partial \mathcal{L}}{\partial \hat{y}} = -\frac{1}{\hat{y}} = -\frac{1}{0.947} = \mathbf{-1.056}$

4. Gradiente respecto a z — regla de la cadena

$\frac{\partial \mathcal{L}}{\partial z} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \hat{y}\,(1-\hat{y})$

$= (-1.056) \cdot (0.947)(0.053) = \hat{y} - y = \mathbf{-0.053}$

Lectura: el gradiente en z es el error de predicción, nada más.

5. Gradiente respecto a w_1 (tenure) — regla de la cadena

$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial z} \cdot \frac{\partial z}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial z} \cdot x_1$

$= (-0.053) \cdot 24 = \mathbf{-1.264}$

Lectura: cada peso hereda ese error escalado por su propia entrada.

6. Actualización SGD con $\eta = 0.01$

$w_1 \leftarrow w_1 - \eta \frac{\partial \mathcal{L}}{\partial w_1} = -0.04 + 0.0126 = \mathbf{-0.027}$

Por qué no derivamos a mano más allá del ejemplo: autograd.

Diferenciación automática. 🔥 Cada operación tensorial registra su jacobiano local. Cuando llamas a `loss.backward()`, PyTorch encadena la regla de la cadena por ti y rellena los gradientes.

Tu trabajo. ⚙️ Definir modelo (forward), pérdida y optimizador. El backward es de la biblioteca.

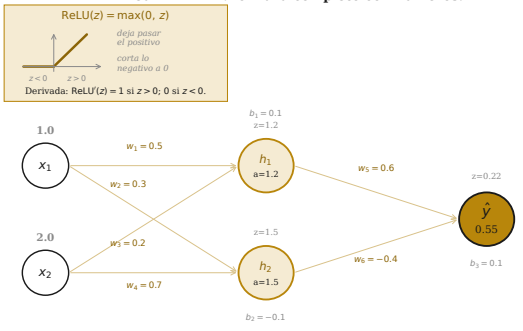
Idea clave. Lo que aprendiste a mano es lo que ocurre por dentro de `loss.backward()`, ni más ni menos.

```
import torch.nn as nn, torch.optim as optim
model = nn.Sequential(nn.Linear(20, 32),
nn.ReLU(),
     nn.Linear(32, 1))
opt = optim.Adam(model.parameters(), lr=1e-3)
loss_fn = nn.BCEWithLogitsLoss()

for xb, yb in train_loader:
  opt.zero_grad()
  pred = model(xb)
  loss = loss_fn(pred, yb)
  loss.backward()
  opt.step()
```

Ejemplo resuelto: forward y backward en una red $2 \to 2 \to 1$.

Red 2 → 2 → 1: forward completo con números.



Forward. 🚀

$z_1 = 0{,}5 \cdot 1 + 0{,}3 \cdot 2 + 0{,}1 = \mathbf{1.2}$

$a_1 = \mathrm{ReLU}(1{,}2) = \mathbf{1.2}$

$z_2 = 0{,}2 \cdot 1 + 0{,}7 \cdot 2 - 0{,}1 = \mathbf{1.5}$

$a_2 = \mathrm{ReLU}(1{,}5) = \mathbf{1.5}$

$z_o = 0{,}6 \cdot 1{,}2 - 0{,}4 \cdot 1{,}5 + 0{,}1 = \mathbf{0.22}$

$\hat{y} = \sigma(0{,}22) = \mathbf{0.555}$

Pérdida (BCE, $y = 1$). 🎯

$\mathcal{L} = -\log(0{,}555) = \mathbf{0.589}$

Backward. 🔥

$\delta_{\text{out}} = \hat{y} - y = \mathbf{-0.445}$

$\frac{\partial \mathcal{L}}{\partial w_5} = \delta \cdot a_1 = -0{,}445 \cdot 1{,}2 = \mathbf{-0.534}$

$\frac{\partial \mathcal{L}}{\partial w_6} = -0{,}445 \cdot 1{,}5 = \mathbf{-0.668}$

$\delta_{h_1} = w_5 \cdot \delta \cdot \mathrm{ReLU}'(z_1) = 0{,}6 \cdot (-0{,}445) \cdot 1 = -0{,}267$

(activa: $z_1 = 1{,}2 > 0 \Rightarrow \mathrm{ReLU}'(z_1) = 1$)

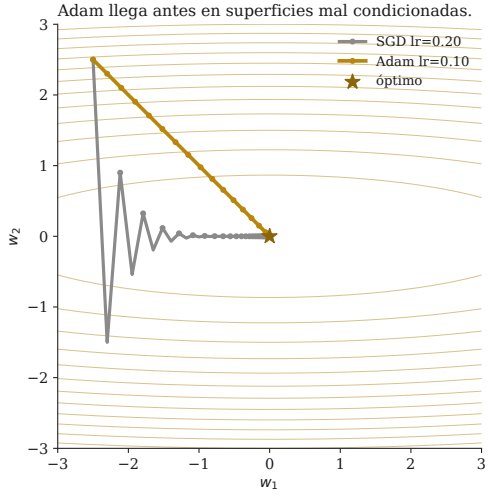
$\frac{\partial \mathcal{L}}{\partial w_1} = \delta_{h_1} \cdot x_1 = \mathbf{-0.267}$

Idea clave. Cada gradiente dice: "mueve este peso así para reducir la pérdida."

Parte 4 / 5

Optimizadores modernos

SGD vs Adam: dos formas de bajar por la misma colina.



Cómo elegir optimizador, learning rate y batch size.

Optimizador por defecto. 🚀 **AdamW** con lr = 10^{-3} y weight_decay = 10^{-4}. Si el modelo es tabular pequeño y el dataset también, prueba SGD con momentum.

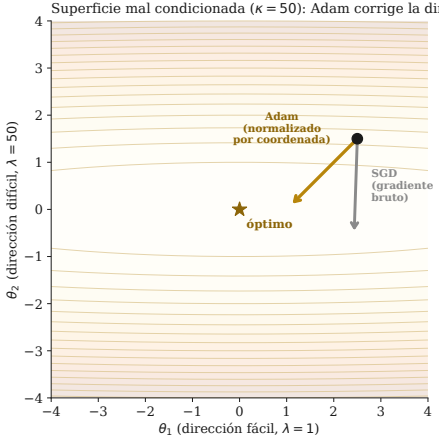
Learning rate. 🎯 El hiperparámetro más importante. Si la pérdida explota, bájalo; si baja muy lento, súbelo.

Batch size. ⚙️ 32, 64 o 128 son los clásicos. Batch grande → entrenamiento estable pero menos exploración; pequeño → más ruido pero a veces mejor generalización.

Diagnóstico rápido. 📊 La pérdida de train debe bajar consistente. Oscila violenta ⇒ lr alto. Baja muy lenta ⇒ lr bajo. Train baja, val sube ⇒ overfitting.

Idea clave. El 80 % de los entrenamientos fallan por un lr mal puesto. Aprended a corregirlo en vivo.

Por qué Adam normaliza cada coordenada: geometría de la pérdida.



El problema. ⚠️ Si la pérdida cambia rápido en θ_2 y lento en θ_1, el gradiente bruto apunta casi perpendicular al óptimo.

Condicionamiento. 🧠 $\kappa = \lambda_{\text{máx}}/\lambda_{\text{mín}}$. Cuando $\kappa \gg 1$, SGD zigzaguea en la dirección empinada y avanza poco en la suave.

Adam corrige. 🚀 Divide cada coordenada por $\sqrt{v_t}$ (varianza acumulada del gradiente). En el límite:

$$\Delta\theta_i \;\approx\; -\eta\,\text{sign}\left(\frac{\partial \mathcal{L}}{\partial \theta_i}\right)$$

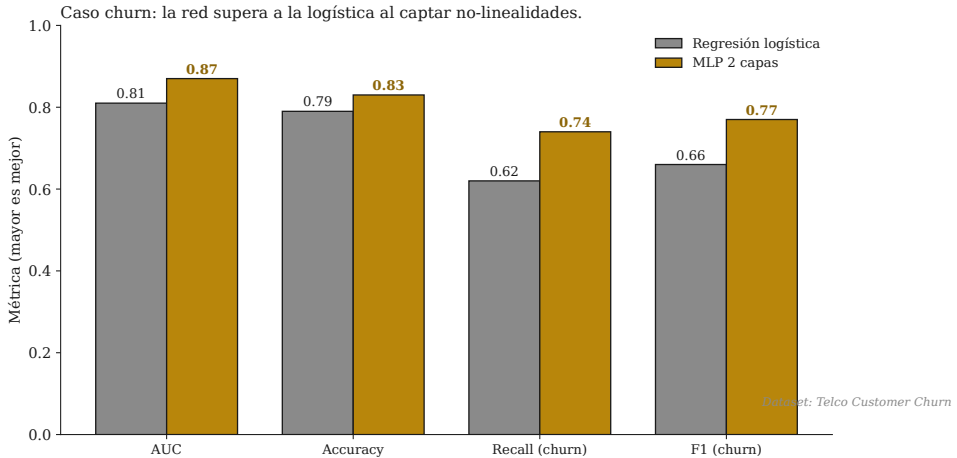
Paso igual en cada dirección, independiente de la curvatura.

Idea clave. Adam aproxima un Newton "barato": normaliza por curvatura sin la Hessiana.

Parte 5 / 5

Caso del día: churn telco

La red supera a la logística cuando hay no-linealidades.



Lo que el alumno se lleva a casa de este día.

Concepto 1. 🧠 Un MLP es una composición de capas lineales y activaciones no lineales; el forward es solo multiplicar matrices y aplicar funciones.

Concepto 2. 🔥 El backward es la regla de la cadena aplicada al grafo de operaciones. PyTorch lo automatiza con `loss.backward()`.

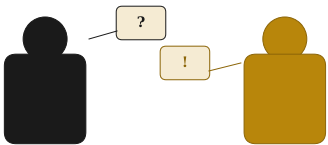
Concepto 3. 🚀 **AdamW** con lr = 10^{-3} es punto de partida razonable para casi todo problema tabular o de imagen pequeña.

Concepto 4. 📊 La elección de métrica condiciona la solución; un 99 % de accuracy no significa nada si la clase importante es la rara.

Para la práctica. Vas a entrenar tu primer MLP sobre Telco Customer Churn. Rúbrica en `enunciado_practica_02.md`.

Próximo día. Día 03: regularización, learning rate schedules y dashboards.

Pausa para debatir: ¿Defenderías un 99 % de accuracy ante el CFO?



La pregunta. ⚠️ Tu modelo de fraude detecta 99 % de transacciones legítimas y 0 % del fraude. ¿Lo despliegas?

Posición A. ✅ Sí: 99 % es 99 %, el sistema reduce manualmente la carga.

Posición B. 🎯 No: 0 % de recall sobre la clase importante es un fracaso de KPI, no un éxito.

Reglas. 5 minutos, dos turnos de palabra por bando, móviles boca abajo. Yo modero, no participo.

Hasta el Día 03.

Regularización, LR schedules y dashboards.

Eduardo C. Garrido-Merchán

ecgarrido@comillas.edu