

King of Dumbs

el rey de los tontos

Nivel: ♠ Técnico

¿Regresión logística o red neuronal? Es la misma neurona

Una regresión logística es una red neuronal de una sola neurona. Qué añade el deep learning, cuándo compensa y cuándo basta lo clásico, en tres niveles.

Español Publicado 4 de octubre de 2026



Escanea para abrir la lección interactiva

<https://kingofdumbs.com/es/lecciones/regresion-logistica-o-red-neuronal>

Qué vas a aprender

- Regresión logística
- Redes neuronales
- Función sigmoide
- Gradiente que se desvanece
- Elegir entre machine learning clásico y deep learning

Símbolos que aparecen

E_a subíndice: la etiqueta de abajo (♠) · Σ sumatorio: súmalos todos (♠) · x_i índice i: el elemento número i (♠) · Π productorio: multiplícalos todos (♠) · $\approx$ aproximadamente igual (♠) · $\sim$ del orden de / se reparte como (♠) · $\leq \geq$ menor o igual / mayor o igual (♠) · $|x|$ valor absoluto: sin signo (♠) · 10^{-3} exponente negativo: dividir (♠) · $h^{(1)}$ superíndice entre paréntesis: no es potencia (♠) · e el número e ($\approx 2,718$) (♠) · $\log$ logaritmo: ¿cuántas veces multiplico? (♦♠) · σ sigma: depende de la lección (♠) · λ lambda: un número que estira (♠) · δ delta minúscula: un error pequeño (♠) · $\|x\|$ norma: la longitud de la flecha (♠) · x^T traspuesta: tumbar la columna (♠) · $\in$ pertenece a (♠) · $\mathbb{R} \mathbb{Z}$ conjuntos de números (♠) · dE/dt derivada: el ritmo de cambio (♠) · ∂ derivada parcial: cambiar una sola cosa (♠) · ∇ gradiente: hacia dónde sube más (♠)

La primera vez que vi una red neuronal dibujada, con sus circulitos y sus flechas, pensé que era otra cosa que la estadística de toda la vida. Luego me tocó programar una regresión logística a mano y me hice la pregunta tonta de esta lección: **¿una regresión logística es una red neuronal?**

La respuesta corta es que sí. Una regresión logística es una red neuronal de **una sola neurona**. No se parecen ni son primas: son el mismo modelo con dos nombres. Si es la misma neurona, la pregunta interesante cambia: ¿qué añaden las redes grandes, y cuándo merece la pena pagarlas?

Elige tu carta: ♣ si empiezas de cero, ♦ si sabes qué es una función o una media, ♠ si quieres la notación. Puedes cambiarla en cualquier momento.

Una neurona, dos nombres

♦ → ♠ Del intermedio al técnico

En ♦ decíamos	En ♠ se escribe
la puntuación	$z = w^T x + b$
los pesos; el término fijo	$w \in \mathbb{R}^d$; b (sesgo)
la sigmoide, que aplasta entre 0 y 1	$\sigma(z) = 1/(1 + e^{-z})$
la probabilidad	$p = \sigma(z) = P(y = 1 \mid x)$
una moneda trucada	$y \sim \text{Bernoulli}(p)$; enlace logit (GLM)
el castigo (pérdida, log-loss)	$\text{BCE} = -[y \log p + (1 - y) \log(1 - p)]$
hacia dónde baja el castigo	$-\nabla L = -X^T(p - y)/n$
un solo fondo	convexa: $\nabla^2 L = X^T S X / n \geq 0$
un freno que encoge los pesos	regularización L2; en sklearn, $C = 1/\lambda$



$$\nabla L = X^T (p - y) / n$$

$$\nabla^2 L = X^T S X / n \geq 0$$

$$S = \text{diag}(p \odot (1 - p))$$

⇒ **convexa: un solo mínimo**

La regresión logística es un modelo lineal generalizado: $y \sim \text{Bernoulli}(p)$ con enlace canónico logit, así que $\text{logit}(p) = w^T x + b$ y $p = \sigma(w^T x + b)$. La sigmoide no es una elección de diseño: es la inversa del enlace canónico de la Bernoulli. Una neurona con activación σ entrenada con entropía cruzada binaria (BCE) es exactamente ese modelo, con la misma verosimilitud.

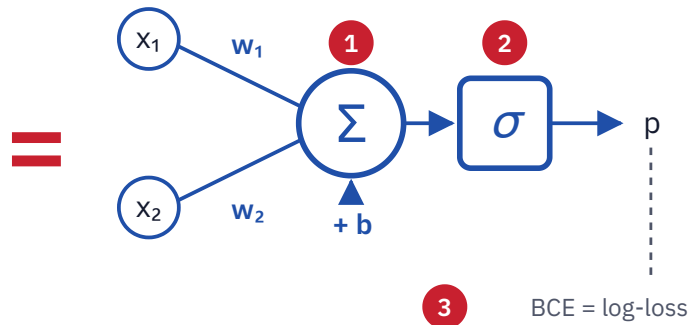
Regresión logística

como se escribe en estadística

- 1 Score lineal
 $z = w_1 x_1 + w_2 x_2 + b$
- 2 Probabilidad
 $p = \sigma(z)$
- 3 Pérdida (log-loss)
 $-[y \cdot \log p + (1-y) \cdot \log(1-p)]$

Una neurona

como se dibuja en deep learning



Las mismas tres piezas con dos notaciones. Cambia el dibujo, no el modelo.

Con la pérdida media $L(w) = (1/n) \sum_i \text{BCE}(y_i, \sigma(w^T x_i))$, el gradiente es $\nabla L = X^T(p - y)/n$ y la Hessiana, $\nabla^2 L = X^T S X/n$ con $S = \text{diag}(p_i(1 - p_i))$. Como S tiene diagonal positiva, $X^T S X \geq 0$: la pérdida es convexa, y estrictamente convexa si X tiene rango completo. Por eso el descenso de gradiente, Newton o IRLS llegan al mismo óptimo. Con datos separables, el óptimo no existe ($\|w\| \rightarrow \infty$) y hace falta regularizar.

Lo comprobé con scikit-learn sin regularización ($C=\text{np.inf}$) frente a una neurona entrenada con descenso de gradiente:

```
import numpy as np
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression

X, y = make_classification(n_samples=2000, n_features=5,
                          flip_y=0.1, random_state=0)
clf = LogisticRegression(C=np.inf).fit(X, y)

w, b = np.zeros(5), 0.0
for _ in range(20000):
    # una neurona: lineal + sigmoide
    p = 1 / (1 + np.exp(-(X @ w + b)))
    # gradiente de la log-loss
    w -= 0.5 * X.T @ (p - y) / len(y)
    b -= 0.5 * (p - y).mean()

# los mismos pesos
print(clf.coef_[0].round(3), w.round(3))
```

Con scikit-learn 1.9.1, los pesos coinciden hasta el tercer decimal. No es casualidad de este conjunto de datos: es el mismo problema convexo resuelto por dos caminos.

La logística de scikit-learn no es la del libro

`LogisticRegression()` aplica por defecto una regularización L2 con $C=1.0$ ($C = 1/\lambda$). Los coeficientes salen encogidos y los odds ratios, sesgados hacia 1. Si quieres interpretar coeficientes, usa `C=np.inf` (o `statsmodels`). Si quieres predecir, ajusta C con validación cruzada.

El Rey de los Tontos dice:

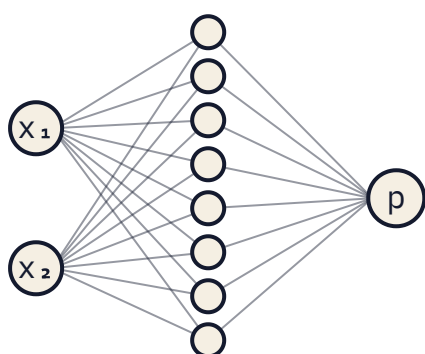
Yo me pasé una semana creyendo que tenía que elegir entre estadística y deep learning. Resulta que llevaba años haciendo deep learning de una neurona sin saberlo.



Quién fabrica las features

♦ → ♠ Del intermedio al técnico

En ♦ decíamos	En ♠ se escribe
las capas ocultas	$h = \tanh(W^{(1)}x + b^{(1)})$
las features que fabrican	$h \in \mathbb{R}^8$: representación aprendida
la neurona del final	$p = \sigma(W^{(2)T}h + b^{(2)})$
el producto de las dos columnas	$x_1 x_2$
cientos de árboles que suman su voto	$p = \sigma(\sum_k f_k(x))$, gradient boosting
un modelo ya entrenado con datos de medio mundo	embedding preentrenado + fine-tuning
la puntuación sin aplastar	logit z ; <code>BCEWithLogitsLoss</code>



features aprendidas

$$h = \tanh(W^{(1)}x + b^{(1)})$$

$$p = \sigma(W^{(2)T}h + b^{(2)})$$

$$W^{(1)}: 8 \times 2, W^{(2)}: 8$$

$$|\theta| = 2 \cdot 8 + 8 + 8 + 1$$

$$|\theta| = 33 \text{ vs } 4$$

Una red binaria de L capas es $p = \sigma(w^T \varphi(x) + b)$, con $\varphi = h^{(L-1)} \circ \dots \circ h^{(1)}$ aprendida conjuntamente con w . Condicionada a φ , la última capa es una regresión logística sobre las features h . Lo mismo vale para casi todo clasificador binario actual: en boosting binario (XGBoost binary:logistic, LightGBM binary), $p = \sigma(\sum_k f_k(x))$ con f_k árboles; en un modelo preentrenado afinado, φ es el embedding.



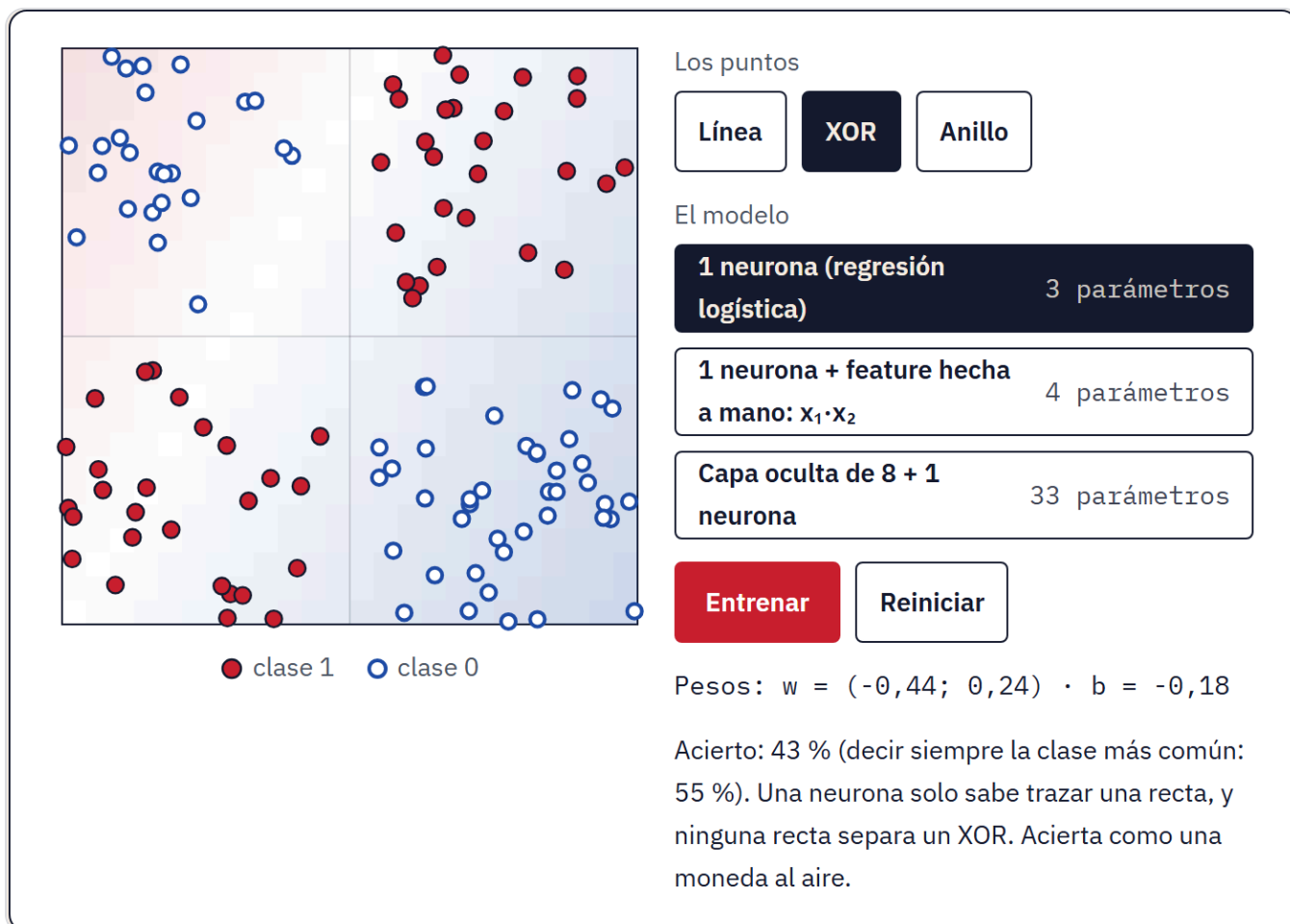
En boosting binario, la salida también es una sigmoide aplicada a la suma de los árboles.

En el XOR, con $x \in \{-1, 1\}^2$, la clase es $y = 1$ si $x_1 x_2 > 0$: ningún hiperplano en (x_1, x_2) la separa, pero en $(x_1, x_2, x_1 x_2)$ basta el hiperplano $x_1 x_2 = 0$. La capa oculta del laboratorio (8 neuronas tanh) llega a una frontera equivalente sin que nadie le diga qué producto buscar, a cambio de $|\theta| = 33$ en lugar de 4, de perder la convexidad y de que $W^{(1)}$ ya no se lea.

No pongas la sigmoide dos veces

En PyTorch, la red devuelve el logit z sin sigmoide y la pérdida es BCEWithLogitsLoss, que calcula $\log \sigma(z)$ con el truco log-sum-exp de forma estable. Poner `nn.Sigmoid()` al final y además BCELoss funciona hasta que un $|z|$ grande da $\log(0)$. La sigmoide se aplica solo al predecir.

Pruébalo. El XOR y el anillo son los dos ejemplos clásicos de puntos que una recta no puede separar. Elige un conjunto y un modelo, y entrénalo:



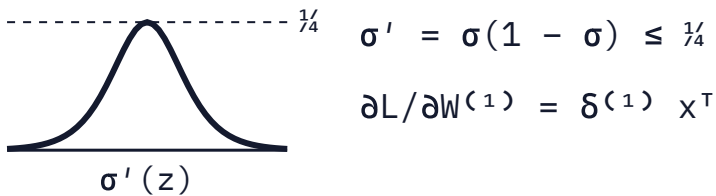
Widget interactivo. Pruébalo en la web: <https://kingofdumbs.com/es/lecciones/regresion-logistica-o-red-neuronal>

Lo que más me gusta de este experimento es la fila del medio. La misma neurona que fracasa con el XOR acierta en cuanto le das hecho a mano el producto de las dos coordenadas. La capa oculta llega al mismo sitio sin que nadie le diga qué buscar, a cambio de 33 números que ajustar en lugar de 4 y de que ya no puedas leer qué ha aprendido.

Por qué la sigmoide salió de las capas ocultas

♦ → ♠ Del intermedio al técnico

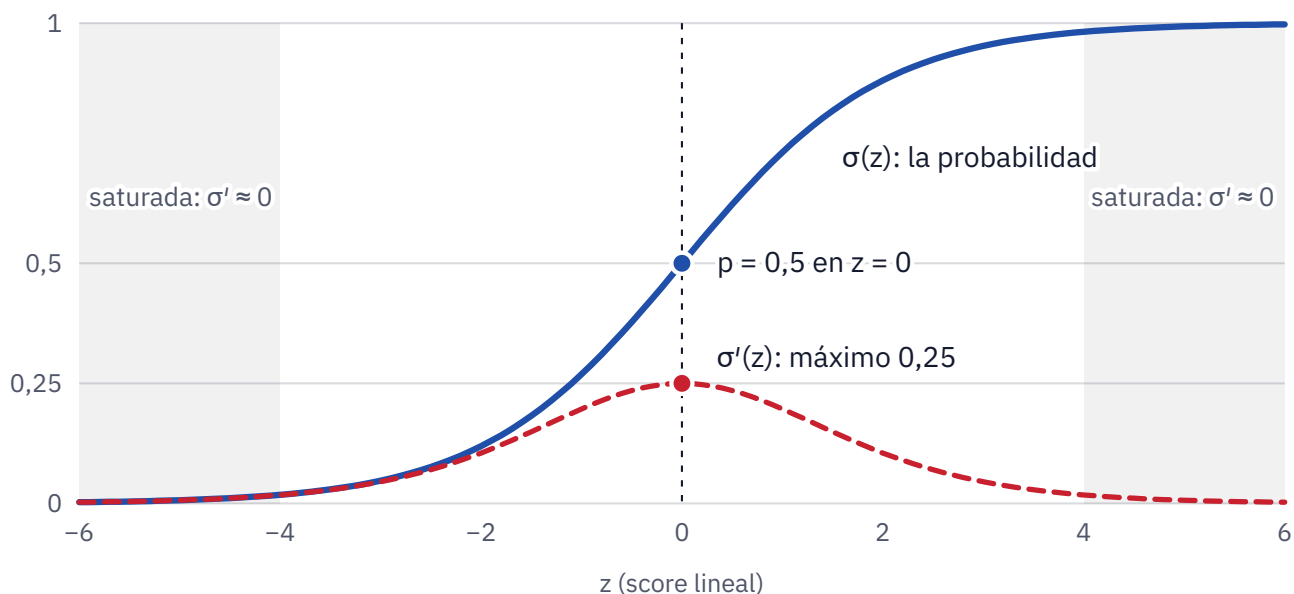
En ♦ decíamos	En ♠ se escribe
la pendiente de la sigmoide	$\sigma'(z) = \sigma(z)(1 - \sigma(z))$
como mucho 0,25	$\sigma' \leq \frac{1}{4}$, con igualdad en $z = 0$
el aviso de cuánto y hacia dónde mover cada peso (gradiente)	$\delta^{(1)} = \partial L / \partial z^{(1)}; \partial L / \partial W^{(1)} = \delta^{(1)} h^{(1-1)\top}$
multiplicar por la pendiente de cada capa (retropropagación)	$\delta^{(1)} = \text{diag}(\sigma'(z^{(1)})) W^{(1+1)\top} \delta^{(1+1)}$
0,25 multiplicado por sí mismo una vez por capa	$(\frac{1}{4})^n$
lo que estiran o encogen los pesos	$\ W^{(1)}\ $ (norma espectral)
una inicialización mejor	Glorot/Xavier: $\text{Var}(W) = 2/(n_{in} + n_{out})$
la ReLU	$\text{ReLU}(z) = \max(0, z)$
una puerta	$g = \sigma(Wx + b) \odot v$



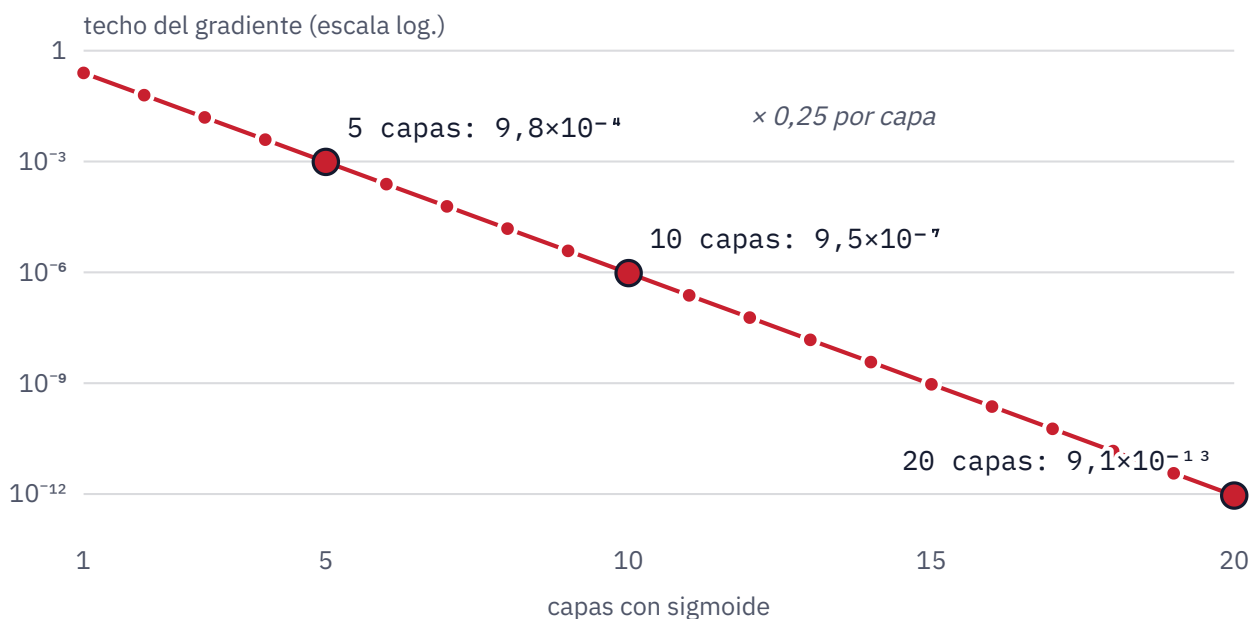
$$\delta^{(1)} = \text{diag}(\sigma'(z^{(1)})) W^{(1+1)\top} \delta^{(1+1)}$$

$$\|\delta^{(1)}\| \leq (\frac{1}{4})^n \prod_i \|W^{(i)}\| \cdot \|\partial L / \partial h^{(n)}\|$$

Con $h^{(i)} = \sigma(z^{(i)})$ y $z^{(i)} = W^{(i)}h^{(i-1)} + b^{(i)}$, la retropropagación da $\delta^{(i)} = \text{diag}(\sigma'(z^{(i)})) W^{(i+1)\top} \delta^{(i+1)}$, y el gradiente de la primera capa es $\partial L / \partial W^{(1)} = \delta^{(1)} x^\top$. Como $\sigma' = \sigma(1 - \sigma) \leq \frac{1}{4}$, con n capas sigmoides $\|\delta^{(1)}\| \leq (\frac{1}{4})^n \prod_i \|W^{(i)}\| \cdot \|\partial L / \partial h^{(n)}\|$. Salvo que las normas de los pesos compensen el factor 4 por capa, el gradiente decae exponencialmente con la profundidad (Hochreiter, 1991 y 1998). Si los $|z|$ están en las colas, $\sigma' \approx 0$ y la cota es mucho más baja.



Calculadas, no dibujadas a mano. La discontinua es σ' : su máximo, $\frac{1}{4}$, es el techo de la cota.



El techo $(\frac{1}{4})^n$ con n de 1 a 20, en escala logarítmica: una recta de pendiente $\log_{10} \frac{1}{4} \approx -0,6$ por capa.

Glorot y Bengio (AISTATS 2010, Tabla 1) lo midieron con 5 capas ocultas en Shapese-3×2: error de test del 82,61 % con sigmoide, 27,15 % con tanh y 15,60 % con tanh e inicialización normalizada, $\text{Var}(W) = 2/(n_{\text{in}} + n_{\text{out}})$, hoy llamada Xavier o Glorot. Además, σ no está centrada en cero: las activaciones medias de 0,5 sesgan las actualizaciones y saturan las capas altas. La ReLU, $\max(0, z)$, tiene derivada 1 en $z > 0$ y rompe el producto de factores menores que $\frac{1}{4}$.

Fuera de la cadena profunda, σ sigue en dos papeles: como enlace de salida, $p = \sigma(z)$, en binario y en multitiqueta (una σ por clase en lugar de softmax), y como puerta multiplicativa. En una LSTM, la puerta de entrada es $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$. En SwiGLU (Shazeer, 2020), $\text{FFN}(x) = (\text{Swish}(xW) \odot x)W_2$ con $\text{Swish}(u) = u \cdot \sigma(\beta u)$: LLaMA y PaLM la usan en cada bloque del transformer.

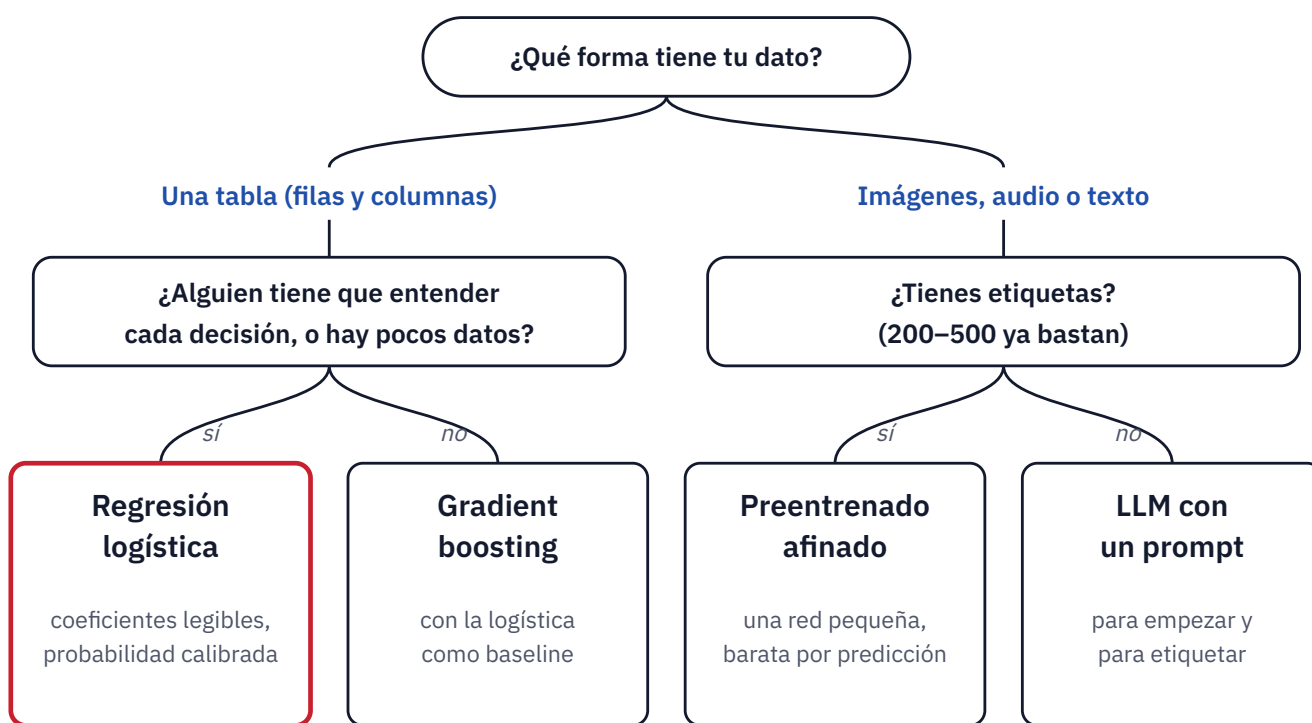
El Rey de los Tontos dice:

La sigmoide es como yo: la echaron de las capas ocultas y acabó de portera en todos los modelos de lenguaje.



Cuándo usar cada cosa

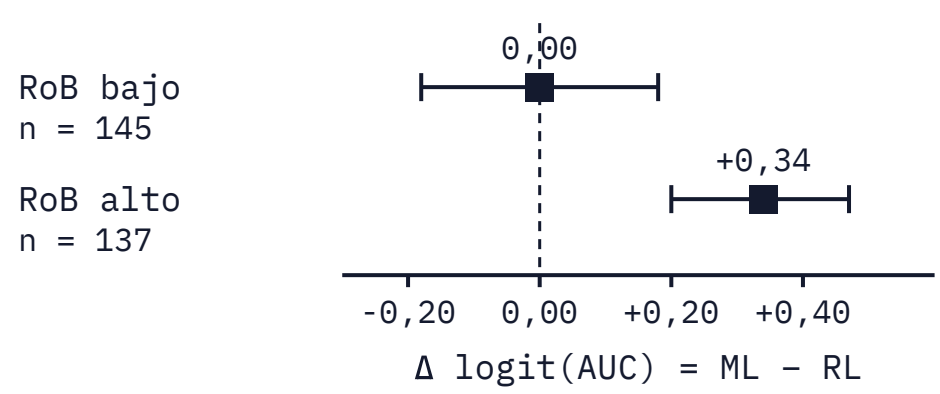
La decisión casi nunca va de “lo complejo que es el problema”. Va de qué forma tiene tu dato, cuántos ejemplos con la respuesta correcta tienes y quién tiene que entender cada decisión. Este árbol vale para los tres niveles:



Para predecir una etiqueta. Bajar de complejidad es barato; subir, caro: empieza por la izquierda.

◆→♠ Del intermedio al técnico

En ◆ decíamos	En ♠ se escribe
el acierto medido con el AUC	$AUC = P(s(x^+) > s(x^-))$
la diferencia fue cero	$\Delta \text{logit}(AUC) = 0,00$, IC 95 % $[-0,18, 0,18]$
riesgo de sesgo	RoB (risk of bias)
decir siempre la clase más común	baseline de clase mayoritaria
afinados; sin ejemplos	fine-tuning; zero-shot
árboles que suman su voto	GBDT (XGBoost, LightGBM)
bien calibrada	$P(y = 1 \mid \hat{p} = q) \approx q$; ECE
un solo ajuste	temperature scaling: $p = \sigma(z/T)$



Christodoulou et al. (*J Clin Epidemiol*, 2019): 71 estudios y 282 comparaciones. Con RoB bajo (145), $\Delta \text{logit}(AUC) = 0,00 [-0,18, 0,18]$; con RoB alto (137), $+0,34 [0,20, 0,47]$. La ventaja aparente del ML se concentra donde la validación es peor. Solo leí el resumen. En un banco de 45 conjuntos tabulares, Grinsztajn et al. (2022) encuentran que los modelos de árboles siguen siendo el estado del arte con datos de tamaño medio (unas 10.000 filas): las redes sufren con features no informativas y con funciones objetivo irregulares.

En texto, Bucher y Martini (2024) comparan el mejor modelo afinado con el mejor LLM generativo zero-shot (GPT-3.5, GPT-4 y Claude 3 Opus) en cuatro tareas. En dos de ellas, el zero-shot queda por debajo del baseline de clase mayoritaria (0,73 y 0,50). Según los autores (§5.5), el fine-tuning rinde bien con entre 200 y 500 etiquetas.

Que salga entre 0 y 1 no la hace una probabilidad

La logística maximiza la verosimilitud de una Bernoulli, así que sale bien calibrada si el modelo está bien especificado. Las redes profundas modernas, no: Guo et al. (*ICML 2017*, Tabla 1) midieron un ECE del 16,53 % en una ResNet-110 sobre CIFAR-100, que bajó al 1,26 % con

temperature scaling, un único parámetro T ajustado en validación: $p = \sigma(z/T)$, o $\text{softmax}(z/T)$ con varias clases. Si la salida alimenta una decisión con coste, recalibra.



Reto final

1. Una regresión logística y una neurona con sigmoide, entrenadas con el mismo castigo, ¿qué son?

- a. Primas: se parecen, pero dan resultados distintos
- b. El mismo modelo con dos nombres
- c. La neurona siempre acierta más, porque es deep learning

2. ¿Por qué dejó de usarse la sigmoide en las capas ocultas?

- a. Porque es lenta de calcular
- b. Porque el aviso del error se encoge en cada capa y las primeras casi no aprenden
- c. Porque los números entre 0 y 1 no sirven para nada

3. Tienes una tabla de 5.000 filas (edad, importe, visitas) y alguien tiene que entender cada decisión. ¿Por dónde empiezas?

- a. Un modelo de lenguaje con un prompt
- b. Una red profunda de diez capas
- c. Una regresión logística



En tres frases

1. Una regresión logística es una red neuronal de una sola neurona: la misma suma con pesos, la misma curva en S (la sigmoide) y el mismo castigo.

2. Lo que añade el deep learning son las capas de antes, que fabrican las features; la sigmoide salió de esas capas porque apaga el aviso del error, y hoy trabaja en la salida y en las puertas.
3. Si tus datos son una tabla con columnas que ya significan algo, empieza por la logística, que tarda minutos y te dice cuánto tiene que ganar lo complejo; si son píxeles, audio o texto, alguien tiene que fabricar las features.



Fuentes

- Regresión logística como modelo lineal generalizado y sigmoide como enlace canónico: [Stanford CS229, apuntes](#). Convexidad de la log-loss: Boyd y Vandenberghe, [Convex Optimization](#), §7.1.1.
- Valores por defecto de LogisticRegression (L2, C=1.0): [documentación de scikit-learn](#).
- Gradiente que se desvanece y derivada máxima de la sigmoide: Hochreiter, tesis de 1991 ([PDF](#), en alemán) y artículo de 1998 ([PDF](#)).
- Error de test con sigmoide, tanh y tanh con inicialización normalizada: Glorot y Bengio, *AISTATS* 2010, [Tabla 1](#).
- SwiGLU: Shazeer, [arXiv:2002.05202](#); su uso en [LLaMA](#) y [PaLM](#). Puertas de LSTM: [Hochreiter y Schmidhuber, 1997](#).
- Machine learning frente a regresión logística en predicción clínica: Christodoulou et al., *J Clin Epidemiol* 110, 12 (2019), [doi:10.1016/j.jclinepi.2019.02.004](#). Leído el resumen.
- Árboles frente a redes en datos tabulares: Grinsztajn et al., [arXiv:2207.08815](#).
- Modelos afinados frente a LLM sin ejemplos: Bucher y Martini, [arXiv:2406.08660](#), Tablas 1 a 4 y §5.5.
- Calibración de redes modernas: Guo et al., *ICML* 2017, [arXiv:1706.04599](#), Tabla 1.
- BCEWithLogitsLoss: [documentación de PyTorch](#).

Cómo está hecha esta lección: a partir de mis notas de machine learning, con ayuda de Claude para ordenarlas y redactar los tres niveles. Las cifras se comprobaron en las tablas originales de cada artículo el 4 de octubre de 2026. El laboratorio de la neurona entrena en tu navegador con descenso de gradiente, igual que el código del nivel ♠. Los dibujos son propios.

Soluciones

1. **b)** El mismo modelo con dos nombres. Eso es. Mismas tres piezas, mismos pesos al final.
2. **b)** Porque el aviso del error se encoge en cada capa y las primeras casi no aprenden. Exacto: como mucho 0,25 por capa. Con diez capas no llega ni un susurro.
3. **c)** Una regresión logística. Bien. Las columnas ya significan algo y la decisión se tiene que poder explicar: la logística es el punto de partida.

Lección de King of Dumbs, kingofdumbs.com. Versión para leer sin conexión; la original, con widgets que se tocan, está en: <https://kingofdumbs.com/es/lecciones/regresion-logistica-o-red-neuronal>

Actualizada el 4 de octubre de 2026.