26 enero 2026

$$b_{k+1} \longleftarrow b_k - \eta \frac{\partial E_{in}(w_k, b_k)}{\partial b}$$

$$w_{j,k+1} \longleftarrow w_{j,k} - \eta \frac{\partial E_{in}(w_k, b_k)}{\partial w_j}$$

27 enero 2026

$$\theta_{k+1} \leftarrow \theta_k - \eta \nabla f(\theta_k)$$

$$h_\theta(x) = w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b$$

$$= [x, 1] \begin{bmatrix} w \\ b \end{bmatrix} = [x^T, 1]\, \theta$$

* - mejor

$$\theta^* = w^*, b^* = \arg \min_{\theta \in \mathbb{R}^{n+1}} \underbrace{\frac{1}{2M} \sum_{i=1}^{M} \Big(y^{(i)} - \overbrace{[x^T, 1]\,\theta}^{\hat{y}^{(i)}}\Big)^2}$$

Para w_j

$$\frac{\partial E_{in}(\theta)}{\partial w_j} = \frac{1}{\cancel{2}M} \sum_{i=1}^{M} -\cancel{2}\left(y^{(i)} - \hat{y}^{(i)}\right) \cdot x_j^{(i)}$$

$$\frac{\partial E_{in}(\theta)}{\partial b} = \frac{1}{2M} \sum_{i=1}^{M} -2\left(y^{(i)} - \hat{y}^{(i)}\right) \cdot$$

desc. grad error muestra

$$\nabla E_{in}(\theta_k) = \begin{bmatrix} \frac{1}{M} \sum_{i=1}^{M} -\left(y^{(i)} - \hat{y}^{(i)}\right) x_1^{(i)} \\ \frac{1}{M} \sum_{i=1}^{M} -\left(y^{(i)} - \hat{y}^{(i)}\right) x_2^{(i)} \\ \vdots \\ \frac{1}{M} \sum_{i=1}^{M} -\left(y^{(i)} - \hat{y}^{(i)}\right) x_n^{(i)} \\ \frac{1}{M} \sum_{i=1}^{M} -\left(y^{(i)} - \hat{y}^{(i)}\right) \end{bmatrix}$$

Podemos sacar $-\frac{1}{M}$

27 enero 2026

continuación

$$= -\frac{1}{M}\underbrace{\begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(m)} \\ \vdots & & & \\ x_n^{(1)} & x_n^{(2)} & \dots & x_n^{(m)} \\ 1 & 1 & \dots & 1 \end{bmatrix}}_{(n+1,\ m)} \underbrace{\begin{bmatrix} y^{(1)} - \hat{y}^{(1)} \\ y^{(2)} - \hat{y}^{(2)} \\ \vdots \\ \vdots \\ y^{(m)} - \hat{y}^{(n)} \end{bmatrix}}_{(m,1)}$$

$\to Xe^T$ $\to Y - Xe\,\theta$ $\to Xw + \vec{1}b$

$$\theta \longleftarrow \theta - \eta \nabla E_{in}(\theta)$$

$$\theta \longleftarrow \theta + \frac{\eta}{M} \underbrace{Xe^T(Y - Xe\,\theta)}$$

24:00 Programa en Python

- Necesito un programa que aprende y uno que predice (descenso gradiente de una lineal) con $XW + 1$ $lr = \eta$

epochs - iteraciones

e_tol - error tolerancia

X - matriz

np.square - vector con todos los cuadrados

_n es new

```
de dg_lin(X, Y, Wø, bø, lr, max-epochs, e_tol):

        w = wø.copy()   # las conds. iniciales
        b = bø.copy()   # no las modificamos
        hist = []       # historial
        for _ in range(max-epochs):
            Ŷ_est = X @ w+b          # @ es mult matricial
            Err = Y - Yest
            hist.append(np.square(err).mean())

# Y e Y ya deben exister
# X.shape = [m, n], Y.shape = [m, 1]
# w = np.zeros(X.shape[-1])    X.shape[-1] = n
# b = 0
# W_n, b_n, hist = dg_lin(X, Y, w, b, 0.1, 50,
                                          1e-4)
```

27 enero 2026

M = X.shape [0]

y.shape = m

derivada de b = d_b

#¿Aprendí suficiente? No. lo vuelvo a llamar. continua python.

```
    grad_w = -(1/M) X.T @ Err
    d_b = Err.mean()
    w -= lr · grad_w
    b -= lr · d_b
    if np.abs(grad_w).max() < e.tol:
        break
    return w, b, hist
```

29 enero 2026

Convertir de discreto a continuo

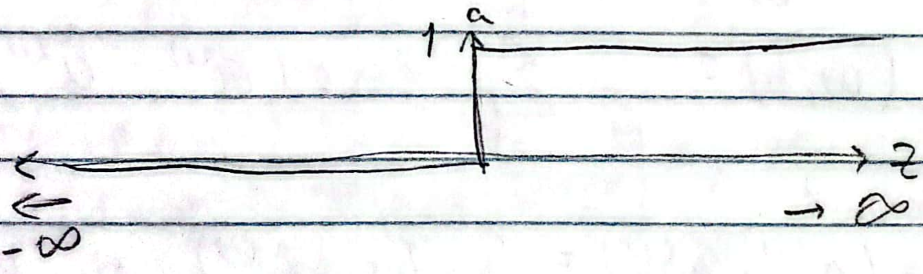
$Pr(Y=1 \mid \underset{(\text{conozco } X)}{X = x}; \theta) = \hat{a}$ Sálida de neurona (activación)

$\hat{y} = \begin{cases} 1 & \text{si } a > h_{umbral} \\ -1 & \text{en otro caso} \end{cases}$ - usamos humbral para poner un mínimo aceptable

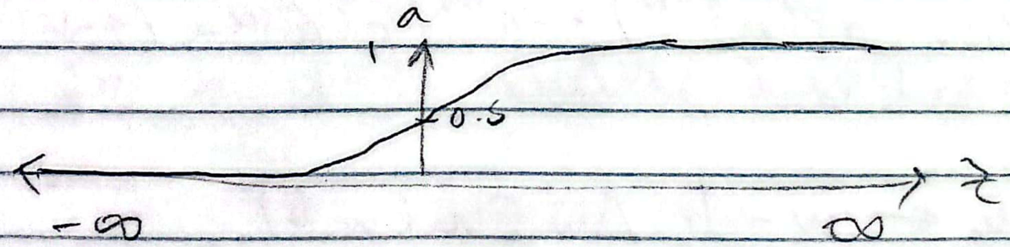
> Máximo a posteriori → $h = 0.5$

$\hat{a} = f(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b)$

$z = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b = X^T w + b = X_e^T \theta$



Si z es negativo, $a = -1$, si es positivo, 1. > Delta de dirac, función de pulso



$\sigma(z) = \dfrac{1}{1 + e^{-z}}$; sigmoid, función logística

$\hat{a} = Pr(Y = 1 \mid X = x; \theta) = \sigma(w^T x + b)$

> Regresión logística

$$\begin{bmatrix} x_1^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & \dots & x_n^{(2)} \\ \vdots & & \\ x_1^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \quad \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad \begin{bmatrix} a^{(1)} \\ a^{(2)} \\ \vdots \\ a^{(m)} \end{bmatrix}$$

$X \qquad Y \qquad A$ (probabilidad)

$y^{(i)} \in \{-1, 1\}$

pérdida de info

$$Ein(w,b) = \frac{1}{M} \sum_{i=1}^{m} loss(a^{(i)}, \hat{a}^{(i)})$$

> Proposición 1

$$Loss(a^{(i)}, \hat{a}^{(i)}) = \begin{cases} -\log(\hat{a}^{(i)}) & si \quad a^{(i)} = 1 \\ -\log(1-\hat{a}^{(i)}) & si \quad a^{(i)} = \emptyset \end{cases}$$

$$loss(a^{(i)}, \hat{a}^{(i)}) = -a^{(i)} \log(\hat{a}^{(i)} - (1-a^{(i)}) \log(1-\hat{a})$$

> Función de pérdida

$$w \leftarrow w - lr \, \Delta w \, Ein(w,b)$$

$$b \leftarrow b - lr \frac{\partial}{\partial b} Ein(w,b)$$

> Calcular der. parcial respecto a w_j

$$\frac{\partial}{\partial w_j} Ein(w,b) = \frac{\partial}{\partial w_j} \frac{1}{M} \sum_{i=1}^{M}$$

$$\log(\hat{a}^{(i)}) - (1-a^{(i)}) \log(1-\hat{a}^{(i)})$$

donde $\hat{a}^{(i)} = \frac{1}{1+e^{-z^{(i)}}}$, y $z^{(i)} = w_1 x_1^{(i)}$

$+ \dots + w_n x_n^{(i)} + b$

29 enero 2026

$$= \frac{1}{M} \sum_{i=1}^{m} -a^{(i)} \frac{\partial a^{(i)}}{\partial w_j} - \left(\frac{1-a^{(i)}}{1-a^{(i)}} \right) \frac{\partial a^{(i)}}{\partial w_j}$$

$$\boxed{\frac{d}{dx} \log\{v(x)\} = \frac{1}{v(x)} \frac{d}{dx} v(x)}$$

$$\frac{\partial a^{(i)}}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{1+e^{-z^{(i)}}}$$

$$= \frac{\partial}{\partial z^{(i)}} \frac{1}{1+e^{-z^{(i)}}} \frac{\partial z^{(i)}}{\partial w_j} = \frac{\partial}{\partial z^{(i)}} (1+e^{-z^{(i)}})^{-1} \frac{\partial z^{(i)}}{\partial w_j}$$

$$= (1+e^{-z^{(i)}})^{-2} e^{-z^{(i)}} x_j^{(i)}$$

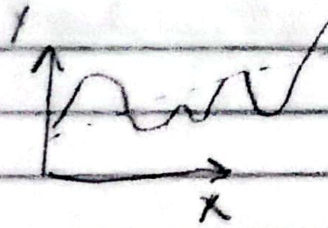
$$= \left(\frac{1+e^{-z^{(i)}}}{(1+e^{-z^{(i)}})^2} - \frac{1}{(1+e^{-z^{(i)}})^2} \right) x_j^{(i)}$$

$$= \left(\frac{1}{1+e^{-z^{(i)}}} - \frac{1}{1+e^{-z^{(i)}}} \right) x_j^{(i)}$$

$$\Downarrow$$

$$= \left(\hat{a}^{(i)} - \hat{a}^{(i)2} \right) = \hat{a}^{(i)} (1-\hat{a}^{(i)}) x_j^{(i)}$$

30 enero 2026



- Habría mucho error con regresión lineal

$\phi : \mathbb{R}^n \to \mathbb{R}^{n'}$ - Función generadora de características

phi $\quad$ R^n prima

$x \in \mathbb{R} \quad x' = \phi(x) = (x, x^2, x^3, x^4)$

$x' = \phi(x) = (\phi_1(x), \phi_2(x), \phi_3(x), \dots, \phi_n(x))$

> Expansión polinomial

$x = (x_1, x_2)$, $q(x) = (x_1, x_2, x_1^2, x_1 x_2, x_2^2)$

, $\phi(x) = (x_1, x_2, \underbrace{x_1^2, x_1 x_2, x_2^2}_{\text{segundo}}, x_1^3, x_1^2 x_2, x_1 x_2^2, x_2^3)$

$x = (x_1, x_2, x_3, x_4)$ Solo 2do orden

$\phi(x) = (x_1, x_2, x_3, x_4, x_1^2, x_1x_2, x_1x_3, x_1x_4, x_2^2, x_2x_3, x_2x_4, x_3^2, x_3x_4, x_4^2)$

$H = \{h\}$ tal que

> $h(x) = w_1x_1 + w_2x_1^2 + w_3x_1^3 + w_4x_1^4 + w_5x_1^5 + w_6x_1^6 + w_7x_1^7 + w_8x_1^8 + b$

> esto es bajo $w_j = 0$ si $j \geq 3$ ← restricciones duras

> $h(x) = w_1x_1 + w_2x_2 + b$

30 de enero 2026

> Las restricciones duras suelen ser excesivas para este tipo de aplicaciones

$h(x) = w^T \phi(x) + b \quad w, \phi(x) \in \mathbb{R}^n, \; b \in \mathbb{R}$

bajo $w_j = 0$ para $j \geq K$

Dim bc es K porque es $K-1+1$ (por lab)

¿Y si cambiamos a restricciones suaves?

$h(x) = w^T \phi(x) + b \quad w, \phi(x) \in \mathbb{R}^n, \; b \in \mathbb{R}$

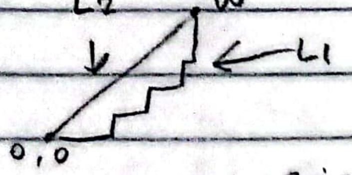
bajo $\underbrace{\sum_{j=1}^{n} w_j^2}_{w^T w \leq C} \leq C$ producto punto de w^T x w

> $\sqrt{C}$ es el radio

> La norma de un vector es a donde llega

$\|w\|_2 = \sqrt{w^T w}$ ← L2 norma 2

$\|w\|_1 = \sum_{j=1}^{n} |w_j|$ ← L1 (cuando solo puedes hacer 2 movimientos)



A esto se le llama regularización.

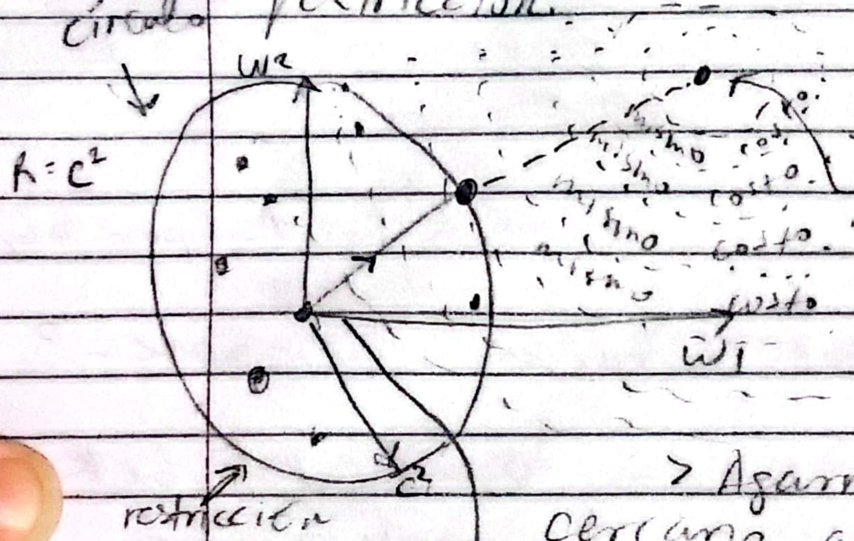
> Calcula la dimensión bc y si tenemos menos datos regularizamos.

30 enero 2026

Regularización lineal

N: w y b óptima que minimice error bajo cierta restricción.

> Tenemos de arga



$(w_1^*, w_2^*) = w^*$ ideal

> Cualquier punto fuera de la restricción no es bueno.

> Agarramos al punto más cercano al óptimo

Este vector es solución ortogonal