Redes neuronales

Clase #4

11/06/2025

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

$$\sigma(w \cdot x + b) = \frac{1}{1+e^{-(w \cdot x + b)}}$$

Empezamos c/ la primera red neuronal.

w: Determina la rapidez con que se activa la neurona (pendiente).

bias: Determina la ubicación de la activación

Diseño de la primera red neuronal

objetivo: Clasificación de dígitos escritos a mano

X: Imágenes de dígitos escritos a mano

1 2 3 4 5

6 7 8 9 0

one hot encoding

vector de un solo valor "1"

y los demás.

clasificación one hot

$$\left. 1 := \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} ; \quad 2 := \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} ; \quad 3 := \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \end{pmatrix} ; \quad \dots \quad 0 := \begin{pmatrix} 0 \\ \vdots \\ 1 \end{pmatrix} \right\} \text{vector de 10 entradas}$$

otra clasificación posible (Pero no conveniente)

Binaria:

$$0 := \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} ; \quad 1 := \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} ; \quad 2 := \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} ; \quad 3 := \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} ; \dots$$

Se empleará la base de datos MNIST

- contiene 60 000 imágenes para entrenar
- contiene 10 000 imágenes para probar (20%)
- Las imágenes son de 28x28 : 784 pixeles
  La capa de entrada será de 784 componentes.
- La capa de salida será un vector de 10 componentes ($y(x)$)

- La intensidad de cada pixel se escala entre 0 y 1

- Definimos la función de costo:

$$C(w,b) = \frac{1}{2n} \sum_{x} \|y(x) - a\|^2$$

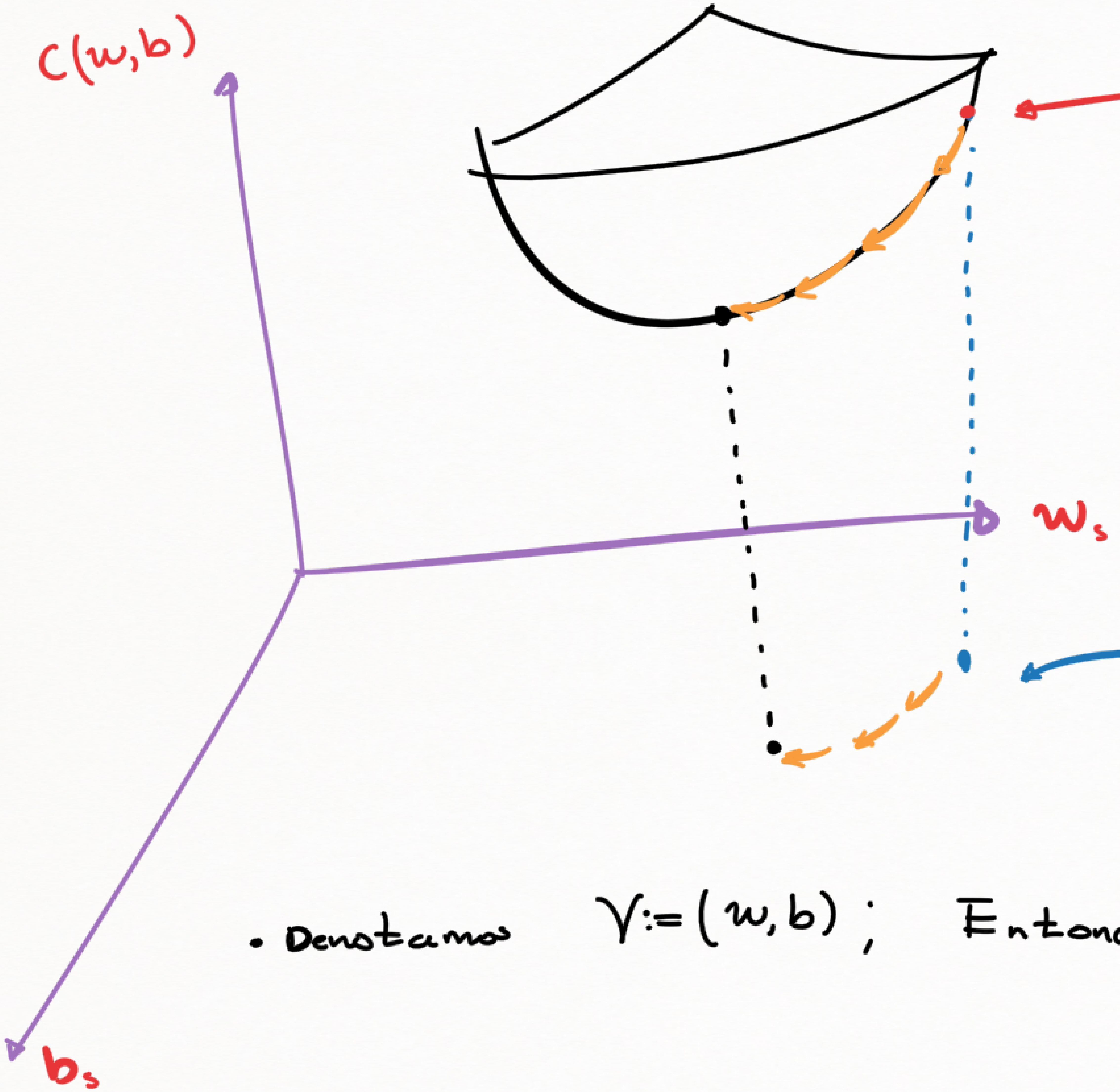
Datos

w = colección de pesos

b = colección de bias

a = Activación de salida $\hat{y}(x)$

El objetivo es minimizar $C(w,b)$ como función de w y b



$\vec{\nabla}_{w,b} C =$

- El algoritmo para encontrar el mínimo se llama:

<u>Gradient descent</u>

Valor inicial de w's y b's

Learning rate = Tasa de aprendizaje

- Denotamos $V := (w,b)$; Entonces proponemos

↳ $\Delta V = -\eta \vec{\nabla}_V C$

cada iteración de "mi entrenamiento" será

$$V \longrightarrow V' = V - \eta \vec{\nabla}_{\ell} C$$

con...

$$\vec{\nabla} C = \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}, \dots, \frac{\partial C}{\partial v_m} \right)$$

o bien

$$w_k \longrightarrow w_k' = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l \longrightarrow b_l' = b_l - \eta \frac{\partial C}{\partial b_l}$$

* Como la función de costo tiene la forma:

$$C = \frac{1}{n} \sum_x C_x$$

suma sobre datos de entrenamiento

• Podemos aproximar su gradiente tomando sólo un subconjunto de los datos en cada iteración.

- A este subconjunto se le llama mini-batch

- Y a este algoritmo se le llama "Stochastic Gradient Descent"

De modo que:

$$w_k \longrightarrow w_k' = w_k - \frac{\eta}{m_j} \sum^{m_j} \frac{\partial C_{x_j}}{\partial w_k}$$

$$b_l \longrightarrow b_l' = b_l - \frac{\eta}{m_j} \sum^{m_j} \frac{\partial C_{x_j}}{\partial b_l}$$

$m_j \longrightarrow$ Número de elementos en el mini-batch

* cuando se terminan los datos de entrenamiento debido a que se fueron tomando en los mini-batches se le conoce como una época (epoch).

- NO es nada práctico usar diferencias finitas. i.e. → método de Euler
  ↳ Picard.

↳ Derivación automática.

• Métodos para realizar derivadas.

1:- Derivación numérica

2:- Derivación simbólica

3:- Diferenciación automática

La Derivación simbólica emplea las reglas de derivación, por ejemplo:

a) $$\frac{d}{dx}\left(f(x)+g(x)\right) = \frac{df}{dx} + \frac{dg}{dx}$$

b) $$\frac{d}{dx}\left(f(x)\,g(x)\right) = \frac{df(x)}{dx}\,g(x) + f(x)\,\frac{dg(x)}{dx}$$

c) $$\frac{d}{dx} f(g(x_0)) = \frac{df(g(x_0))}{dg} \cdot \frac{dg(x_0)}{dx}$$

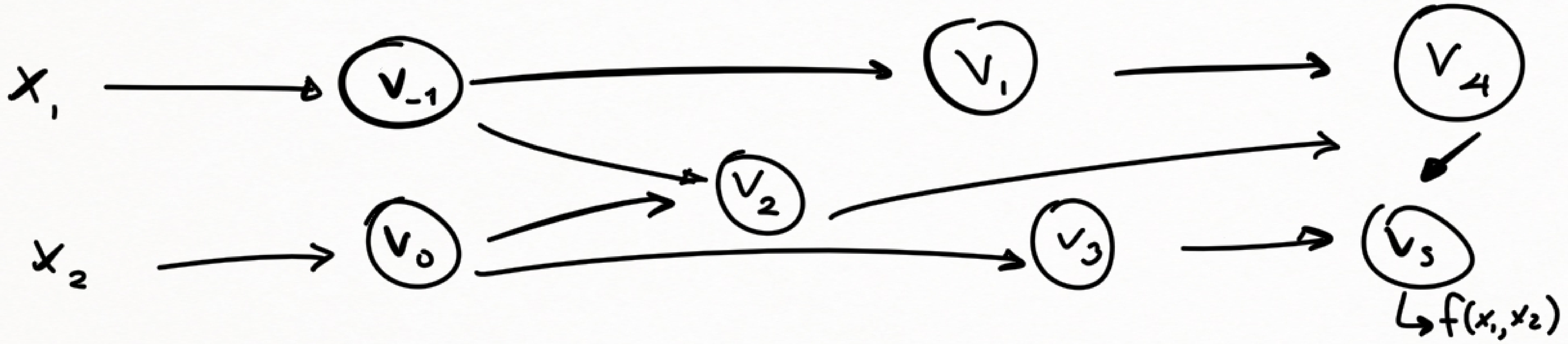
— Derivación numérica:

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x+h e_i) - f(x)}{h}$$

Article → "Back propagation"

Ejemplo: $f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \operatorname{sen}(x_2)$

Derivación automática



Evaluación hacia adelante

(Ejemplo)

$v_{-1} = x_1 \qquad = 2$

$v_0 = x_2 \qquad = 5$

Evaluación hacia adelante (Ejemplo)

"Lazy-evaluation"

$v_{-1} = x_1 \quad = 2$

Sig.

$v_0 = x_2 \quad = 5$

$v_1 = \ln v_{-1} \quad = \ln 2$

$v_2 = v_{-1} \times v_0 \quad = 2 \times 5$

$y = v_5 \quad = 11.652$

$v_3 = \operatorname{sen} v_0 \quad = \operatorname{sen} 5$

$v_4 = v_1 + v_2 \quad = 0.693 + 10$

$v_5 = v_4 - v_3 \quad = 10.693 + 0.959$