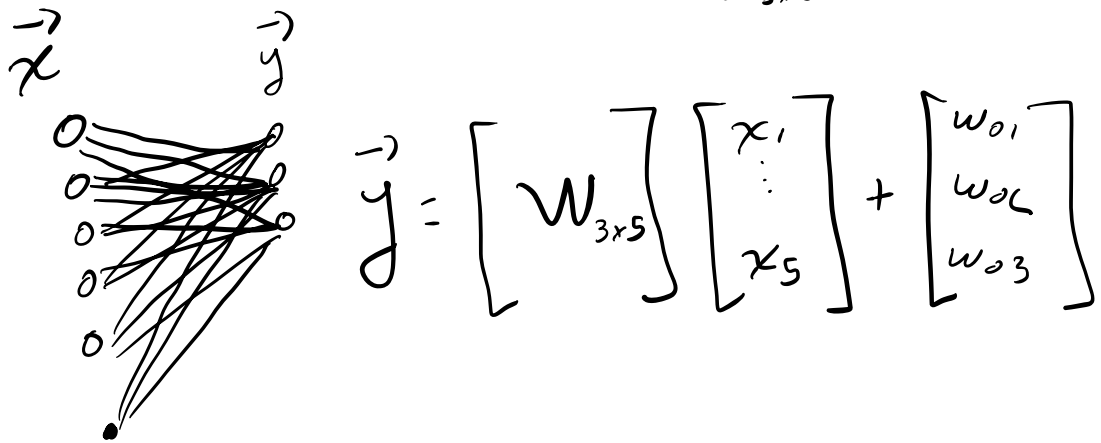


Intro a redes neuronales.

Capas

- Linear (a, b) (\vec{x}) = $\underset{\substack{\downarrow \\ W_{b \times a}}}{W} \vec{x}_{a \times 1} + \vec{b}_{b \times 1}$



- Linear (1, 1) (x) = $w_1 x + w_0$

- Sigmoid (\cdot) (x) = $\frac{1}{1 + \exp(-x)}$

Sequential (

lista de capas

)

Regresión
lineal

Linear (1, 1)

Regresión
logística

Sequential (

Linear (1, 1),

Sigmoidal()

)

$$\text{Linear}(1, 50)(x) = \begin{bmatrix} \vdots \\ \end{bmatrix}_{50 \times 1} x + \begin{bmatrix} \vdots \\ \end{bmatrix}_{50 \times 1}$$

tamaño de entrada
tamaño de salida.

$$= \begin{bmatrix} w_1^1 x + w_0^1 \\ w_1^2 x + w_0^2 \\ \vdots \\ w_1^{50} x + w_0^{50} \end{bmatrix}$$

$$\text{Tanh}(\quad) = \begin{bmatrix} \text{Tanh}(w_1^1 x + w_0^1) \\ \vdots \\ \text{Tanh}(w_1^{50} x + w_0^{50}) \end{bmatrix}$$

$$\text{Linear}(50, 1)(h) = \tilde{W}_{1 \times 50} []_{50 \times 1} + b$$

L_1

$$\begin{bmatrix} \tilde{w}_1 & \dots & \tilde{w}_{50} \end{bmatrix} \begin{bmatrix} \text{Tanh}(w_1^1 x + w_0^1) \\ \vdots \\ \text{Tanh}(w_1^{50} x + w_0^{50}) \end{bmatrix}$$

$$+ \tilde{w}_0$$

$$\tilde{w}_0 + \tilde{w}_1 \text{Tanh}(w_1^1 x + w_0^1) + \dots \\ + \tilde{w}_{50} \text{Tanh}(w_1^{50} x + w_0^{50})$$

$\text{Tanh}()$

$\text{Linear}(1, 50)$

Tanh

Teorema de aproximación universal (Cybenko, 1989)

Una función del estilo

Sequential (

Linear (a, b)

Tanh () - - - - -

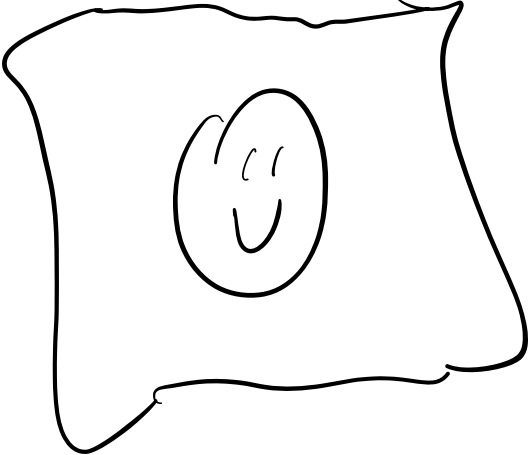
Linear (b, c)

)

"una función
sigmoide"
↳ Tanh
sigmoide
⋮
funciones de
activación.

Aproxima Cualquier función^{continua}
entre compactos de $\mathbb{R}^d \rightarrow \mathbb{R}^C$

$\begin{bmatrix} \text{Edad} \\ \text{Peso} \\ \vdots \end{bmatrix} \rightarrow \begin{bmatrix} \% \text{ cáncer} \\ \% \text{ diabetes} \end{bmatrix}$

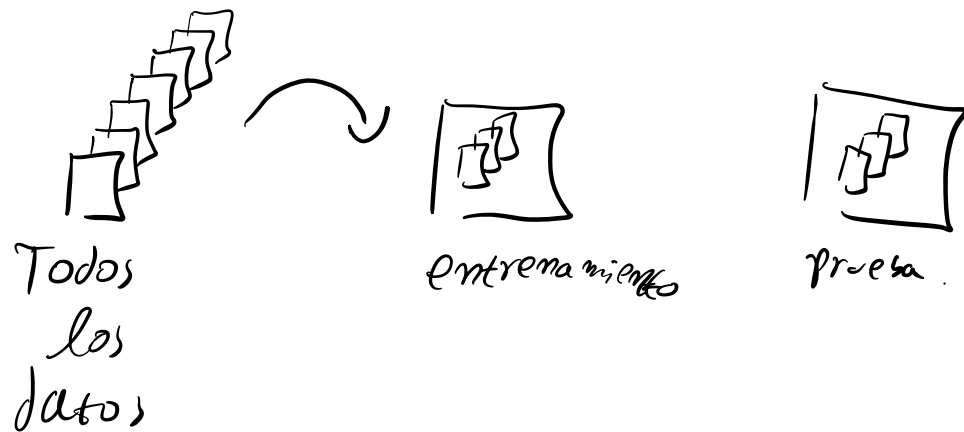
 $\rightarrow \begin{bmatrix} \text{Está} \\ \text{Sonriendo} \end{bmatrix}$

$\begin{bmatrix} \text{Una Obra} \\ \text{de} \\ \text{Arte} \end{bmatrix} \rightarrow \begin{bmatrix} \text{Prob. dadais} \\ \vdots \end{bmatrix}$

Pasos para entrenar una red.

1. Preparar los datos.

↳ Outcome: data loaders.



2. Minimizar la función

$$\mathcal{E}(\bar{\omega})$$

usando los datos de
entrenamiento.

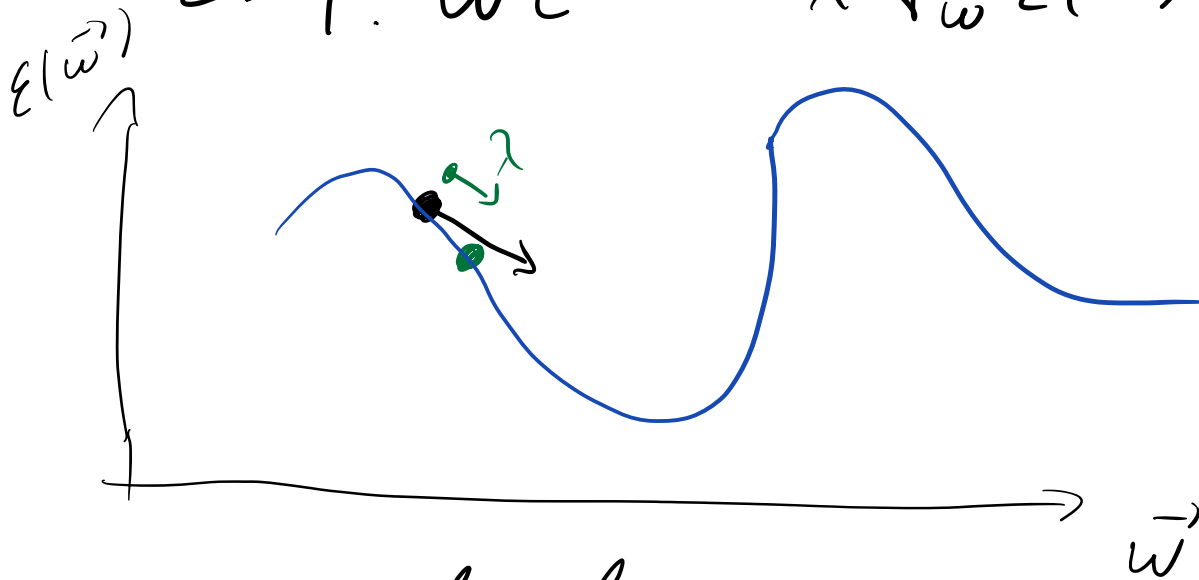
2.1. Definir $\mathcal{E}(\bar{\omega})$

2.2. Definir $\mathcal{J}(x; \bar{\omega})$

2.3 Computar

$$\nabla_{\vec{w}} \mathcal{E}(\vec{w}; D)$$

$$2.4. \vec{w} \leftarrow \vec{w} - \lambda \nabla_{\vec{w}} \mathcal{E}(\vec{w})$$



λ : el learning rate.

$$\lambda \approx 10^{-3}$$

El paso 2.4 se conoce como
descender por el gradiente

(GD)

Casi nunca podemos
computar $\mathcal{E}(\bar{\omega}; D)$ para
toda D .

Entonces la computamos
para

$$S \subseteq D$$

$|S|$ pequeño \subset el tamaño
del batch

batch size
 $\approx 128, 64, 2$

$$\mathcal{E}(\bar{\omega}; D) = \frac{1}{2} \sum_n^N (y(x_n; \bar{\omega}) - t_n)^2$$

SGD: Stochastic gradient descent.

ADAM: adaptive momentum.

En resumen.

- $\mathcal{E}(\bar{w})$, $-J(\bar{x}, \bar{w})$,

- Optimización

1. Preparamos los datos

2. Minimizar $\mathcal{E}(\bar{w})$

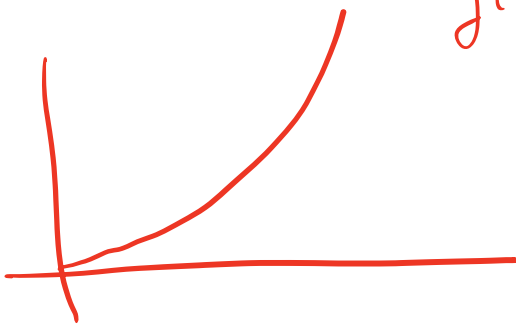
2.1. Evaluamos $\nabla_{\bar{w}} \mathcal{E}(\bar{w})$

2.2 tomamos "un paso" en

nuestro algoritmo de
optimización.

Tarea:

1. Implementar una función
que devuelva
 $y(x) = x^2$



$(x, y(x))$

n_{datos} números al
azar en
 $[0, 5]$

$y(x) = x^2 + \epsilon$
 \downarrow
 $\mathcal{N}(0, 0.1)$

2. Gráficos.