

# Análisis Matemático para Inteligencia Artificial

Verónica Pastor (vpastor@fi.uba.ar),  
Martín Errázquin (merrazquin@fi.uba.ar)

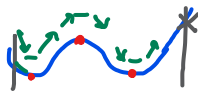
Especialización en Inteligencia Artificial

30/05/2023

# Análisis Matemático - Optimización

# Repaso

- 1 En los videos de repaso definimos funciones de cuyo dominio y codominio eran los reales, la gráfica de la función se representa en  $\mathbb{R}^2$ .
- 2 Toda función  $f$  describe el cambio de una magnitud (v. dependiente) en términos de otra (v. independiente), cuando esta variable se mueve en cierto intervalo  $[x_0, x_0 + h]$  la variación total se mide como  $f(x_0 + h) - f(x_0)$ .
- 3 Mientras que la variación media es  $\frac{f(x_0 + h) - f(x_0)}{(x_0 + h) - x_0}$ . Geométricamente, podemos ver la variación media como la pendiente de la recta secante.
- 4 Cuando hacemos que  $h \rightarrow 0$ , ...



$$\frac{y_2 - y_1}{x_2 - x_1}$$

$$\frac{f(x_1 + h) - f(x_1)}{x_1 + h - x_1}$$

... esto nos conduce a la definición de derivada de  $f$  en

$x_0$ :

$$\lim_{h \rightarrow 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

# Clasificación de funciones

Dada  $f : D \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$ .

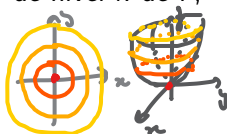
- Si  $m = 1$  diremos que es una función
  - **escalar**, si  $n = 1$ ,
  - **campo escalar**,  $n > 1$ .
- Si  $m > 1$  diremos que es una función
  - **vectorial**, si  $n = 1$ ,
  - **campo vectorial**,  $n > 1$ .

$in \rightarrow \underline{dim=1} \Rightarrow \text{función}$   
 $\rightarrow \underline{dim>1} \Rightarrow \text{campo}$

$out \rightarrow \underline{dim=1} \Rightarrow \text{escalar}$   
 $\rightarrow \underline{dim>1} \Rightarrow \text{vectorial}$

**Conjuntos de Nivel** Dada  $f : D \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$  el conjunto de nivel  $k$  de  $f$ ,  $L_k \subset \mathbb{R}^n$ , definido por:

$$L_k = \{x \in \mathbb{R}^n / x \in D \wedge f(x) = k\}$$

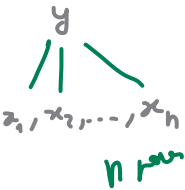



La representación geométrica de  $L_k$  se obtiene identificando gráficamente los puntos del dominio de la función para los cuales el valor de  $f$  es igual a  $k$ , para graficar no es necesario agregar un eje.

$\hookrightarrow (lat, long) \rightarrow h$  mapas con relieve

# Derivando campos ...

- escalares: Sea  $f : D \subset \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $(x_1, \dots, x_n)^T \mapsto f((x_1, \dots, x_n)^T)$ , se definen las **derivadas parciales** como:


$$\frac{\partial f}{\partial x_1} = \lim_{h \rightarrow 0} \frac{f(x_1 + h, x_2, \dots, x_n) - f(x_1, x_2, \dots, x_n)}{h}$$

$$\frac{\partial f}{\partial x_n} = \lim_{h \rightarrow 0} \frac{f(x_1, x_2, \dots, x_n + h) - f(x_1, x_2, \dots, x_n)}{h}$$


Se define el **gradiente** como:  $\nabla f = \left( \frac{\partial f}{\partial x_1} \dots \frac{\partial f}{\partial x_n} \right) \in \mathbb{R}^n$

Importante: El gradiente apunta en la dirección de máximo crecimiento.

- vectoriales: Sea  $f : D \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  
 $(x_1, \dots, x_n)^T \mapsto (f_1((x_1, \dots, x_n)^T), \dots, f_m((x_1, \dots, x_n)^T))$ , se define el **jacobiano** como:

$y_1, y_2, \dots, y_m$

$x_1, x_2, \dots, x_n$

$n \cdot m$

$$J_f(x_i) = \frac{\partial f_i}{\partial x_j}$$

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

$$= \begin{pmatrix} -\nabla f_1 - \\ -\nabla f_2 - \\ \vdots \\ -\nabla f_m - \end{pmatrix} \in \mathbb{R}^{m \times n}$$

# Regla de la Cadena en forma matricial

Sea  $f(x_1(s, t), x_2(s, t))$



Y luego

$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial s} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial s}$$

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial t}$$

$$\frac{d g \circ f(x)}{d x} = \frac{d g}{d f} \cdot \frac{d f}{d x}$$

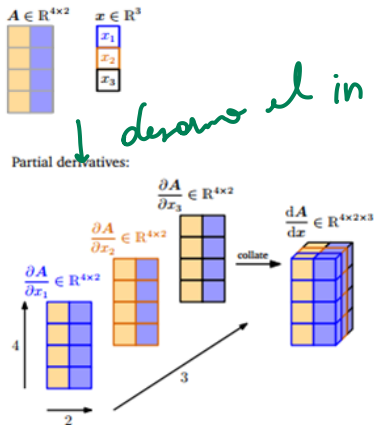
$$\nabla f_{(s,t)} = \frac{df}{d(s, t)} = \frac{df}{dx} \frac{dx}{d(s, t)} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t} \end{bmatrix}$$

Handwritten notes:  $\mathbb{R}^2 \rightarrow \mathbb{R}^1$ ,  $\mathbb{R}^{1 \times 2}$ ,  $\mathbb{R}^{2 \times 2}$ ,  $\mathbb{R}$ .

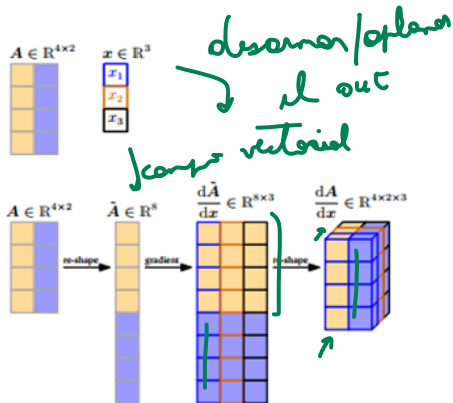
Recordemos reglas de derivación:

- $\frac{\partial (f+g)(s)}{\partial s} = \frac{\partial f}{\partial s} + \frac{\partial g}{\partial s}$
- $\frac{\partial (fg)(s)}{\partial s} = \frac{\partial f}{\partial s} g(s) + f(s) \frac{\partial g}{\partial s}$

# Derivada de matrices



(a) Approach 1: We compute the partial derivative  $\frac{\partial A}{\partial x_1}$ ,  $\frac{\partial A}{\partial x_2}$ ,  $\frac{\partial A}{\partial x_3}$ , each of which is a  $4 \times 2$  matrix, and collate them in a  $4 \times 2 \times 3$  tensor.



(b) Approach 2: We re-shape (flatten)  $A \in \mathbb{R}^{4 \times 2}$  into a vector  $\tilde{A} \in \mathbb{R}^8$ . Then, we compute the gradient  $\frac{d\tilde{A}}{dx} \in \mathbb{R}^{8 \times 3}$ . We obtain the gradient tensor by re-shaping this gradient as illustrated above.

`np.zeros(shape=(2,2,3,4))`

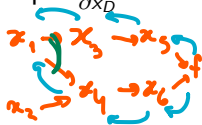
# Diferenciación Automática

Sean, para una función  $f$ :

- $x_1, \dots, x_d$  las variables de entrada
- $x_{d+1}, \dots, x_{D-1}$  las variables intermedias
- $x_D$  la variable de salida
- $g_i$  funciones elementales
- $Hij(x_i)$  el conjunto de nodos hijos de cada  $x_i$

Así queda definido un **grafo de cómputo**. Recordando que  $f = D$ , tenemos que  $\frac{\partial f}{\partial x_D} = 1$ . Para las otras variables  $x_i$  aplicamos la regla de la cadena:

```
class Square:
    def forward(self, x):
        return x2
    def backwards(self, x):
        return 2x
```



$$\frac{\partial f}{\partial x_i} = \sum_{x_j \in Hij(x_i)} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i} = \sum_{x_j \in Hij(x_i)} \frac{\partial f}{\partial g_j} \frac{\partial x_j}{\partial x_i}$$

$$\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial x_3} \cdot \frac{\partial x_3}{\partial x_1} + \frac{\partial f}{\partial x_5} \cdot \frac{\partial x_5}{\partial x_1}$$

- La diferenciación automática se puede utilizar siempre que la función pueda representarse como un grafo de cómputo.
- La gran ganancia de este mecanismo está en que cada función sólo precisa saber cómo derivarse a sí misma, permitiendo OOP.



# Diferenciación Automática: ejemplo

Sean  $e(x, y) = xy$ ,  $f(x) = 3x$ ,  $g(x) = x^2$ ,  $h(x) = \sin(x)$

$$\frac{\partial e}{\partial x} = y, \frac{\partial e}{\partial y} = x, \frac{df}{dx} = 3, \frac{dg}{dx} = 2x, \frac{dh}{dx} = \cos(x)$$

$$\begin{aligned} \frac{\partial a_1}{\partial x} &= \frac{df}{dx}(x) \\ &= 2 \cdot 3 = 6 \end{aligned}$$



$$\frac{\partial s}{\partial a_3} = \frac{\partial e}{\partial x}(a_3, a_4) = a_4 = -0.5966$$

$$\frac{\partial s}{\partial a_4} = \frac{\partial e}{\partial y}(a_3, a_4) = a_3 = 108$$

$$\frac{\partial a_3}{\partial a_1} = \frac{\partial e}{\partial x}(a_1, a_2) = a_2 = 12 \rightarrow \frac{\partial s}{\partial a_1} = \frac{\partial s}{\partial a_3} \cdot \frac{\partial a_3}{\partial a_1} = -0.5966 \cdot 12 = -7.1592$$

$$\begin{aligned} \frac{\partial a_3}{\partial a_2} &= \frac{\partial e}{\partial y}(a_1, a_2) = a_1 = 9 \\ \frac{\partial a_4}{\partial a_2} &= \frac{dh}{dx}(a_2) = \cos(12) \approx 0.979 \end{aligned}$$

$$\begin{aligned} \frac{\partial s}{\partial a_2} &= \frac{\partial s}{\partial a_3} \cdot \frac{\partial a_3}{\partial a_2} + \frac{\partial s}{\partial a_4} \cdot \frac{\partial a_4}{\partial a_2} \\ &= -0.5966 \cdot 9 + 108 \cdot 0.979 \\ &= -5.369 + 105.624 \approx 100.255 \end{aligned}$$

$$\frac{\partial s}{\partial x}, \frac{\partial s}{\partial y} ? \quad \frac{\partial a_2}{\partial y} = \frac{dg}{dx}(y) = 3$$

$$\frac{\partial s}{\partial x} = \frac{\partial s}{\partial a_1} \cdot \frac{\partial a_1}{\partial x} = 6 \cdot -7.1592 = \dots$$

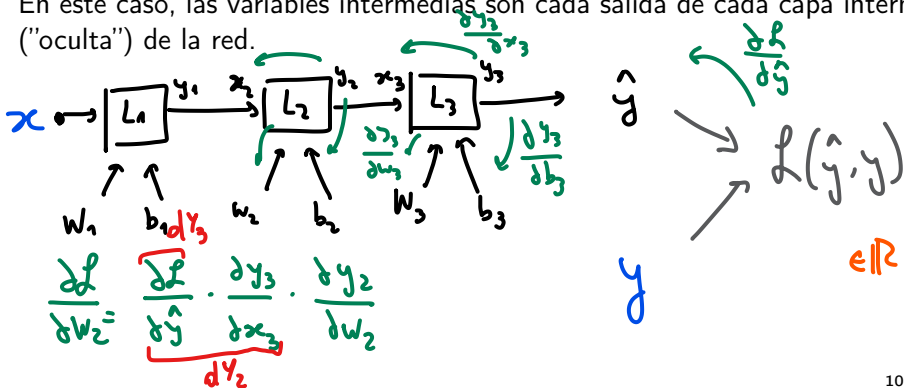
$$\frac{\partial s}{\partial y} = \frac{\partial s}{\partial a_2} \cdot \frac{\partial a_2}{\partial y} = 100.255 \cdot 3 = \dots$$

# Backpropagation

¿Dónde se aplica la diferenciación automática? En **Backpropagation** (o simplemente Backprop), el algoritmo utilizado para entrenar redes neuronales.

¿Qué función cumple? La de computar las derivadas de la función de error/costo respecto de *cada* parámetro de la red neuronal.

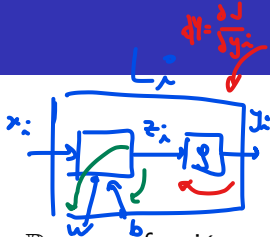
En este caso, las variables intermedias son cada salida de cada capa interna ("oculta") de la red.



# Redes neuronales (I)

Un perceptrón/neurona es un estimador de la forma:

$$\hat{y} = g(w \cdot x + b)$$



donde en su forma más simple  $x, y, w, b \in \mathbb{R}$  y  $g : \mathbb{R} \rightarrow \mathbb{R}$  es una función no lineal como puede ser la sigmoidea  $\sigma(z) = \frac{1}{1+e^{-z}}$ .

Si se define la función  $J(W, b)$  de error respecto de los parámetros  $W$  y  $b$  se puede comprobar que, definiendo  $z = w \cdot x + b$  y suponiendo conocido  $\frac{dJ}{d\hat{y}} = dY \in \mathbb{R}$ :

$$\begin{aligned} \frac{\partial \hat{y}}{\partial z} &= g'(z) \\ \frac{\partial J}{\partial W} &= \frac{dJ}{d\hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial W} = \underbrace{dY}_{\mathbb{R}} \cdot \underbrace{g'(z)}_{\mathbb{R}} \cdot \underbrace{x}_{\mathbb{R}} \in \mathbb{R} \quad \checkmark \\ \frac{\partial J}{\partial b} &= \frac{dJ}{d\hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial b} = dY \cdot g'(z) \cdot \underbrace{1}_{\mathbb{R}} \in \mathbb{R} \quad \checkmark \end{aligned}$$

## Redes neuronales (II)

Si ahora consideramos múltiples entradas, es decir  $x \in \mathbb{R}^n$ ,  $W \in \mathbb{R}^{1 \times n}$ :

$$\hat{y} = g(\overbrace{W}^{1 \times n} \cdot \overbrace{x}^{n \times 1} + \overbrace{b}^1) = g\left(\sum_{i=1}^n w_i \cdot x_i + b\right)$$

Entonces ahora para cada elemento de  $W = (w_1, \dots, w_n)$  vale lo anterior, y por tanto se puede comprobar que

$$\frac{\partial J}{\partial W} = \nabla_J(W) = (dY \cdot \underbrace{g'(z)}_{\frac{\partial z}{\partial w_1}} \cdot x_1, \dots, dY \cdot \underbrace{g'(z)}_{\frac{\partial z}{\partial w_n}} \cdot x_n) = \overbrace{dY}^{\mathbb{R}} \cdot \overbrace{g'(z)}^{\mathbb{R}} \cdot \overbrace{x^T}^{1 \times n} \in \mathbb{R}^{1 \times n}$$

$$\frac{\partial J}{\partial b} = dY \cdot g'(z) \in \mathbb{R}$$

$$\frac{\partial J}{\partial x_i} = \frac{\partial J}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial x_i} = dY \cdot g'(z) \cdot w_i$$

## Redes neuronales (III)

Una capa en una red neuronal se define como un vector de  $k$  neuronas en paralelo. Una propiedad atractiva de este formato es que se puede considerar a la salida de una capa  $y \in \mathbb{R}^k$  como simplemente el  $x$  de la capa siguiente. Por convención (y eficiencia computacional) se suele utilizar la misma no-linealidad  $g$  para todas las neuronas de la capa.

Nuevamente tenemos:

$$\hat{y} = g(\overbrace{W}^{k \times n} \cdot \overbrace{x}^{n \times 1} + \overbrace{b}^{k \times 1})$$

donde  $x \in \mathbb{R}^n$ ,  $W \in \mathbb{R}^{k \times n}$ ,  $b \in \mathbb{R}^k$  y se conviene  $g(z) = \begin{pmatrix} g(z_1) \\ \vdots \\ g(z_k) \end{pmatrix}$

¿Y ahora cómo se calculan las derivadas para  $W$  y  $b$ ?

# Redes neuronales (IV)

En el caso de  $b$  es simple:

$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial b} = \begin{pmatrix} dY_1 \\ \vdots \\ dY_k \end{pmatrix} \odot \begin{pmatrix} g'(z_1) \\ \vdots \\ g'(z_k) \end{pmatrix} \odot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = dY \odot g'(z) \in \mathbb{R}^k \checkmark$$

*Handwritten notes:*  
 - Above the first vector:  $\frac{\partial}{\partial y_{i_1}}$   
 - Above the second vector:  $\frac{\partial J}{\partial b_i} = dY_i \cdot g'(z_i)$   
 - Above the third vector:  $\text{neurona de la capa}$   
 - Under the first vector:  $\in \mathbb{R}^k$   
 - Under the second vector:  $\in \mathbb{R}^k$

Ahora para cada elemento de  $W$  tenemos:

$$\frac{\partial J}{\partial W} = \begin{pmatrix} \frac{\partial J}{\partial W_{1,1}} & \cdots & \frac{\partial J}{\partial W_{1,n}} \\ \frac{\partial J}{\partial W_{2,1}} & \cdots & \frac{\partial J}{\partial W_{2,n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial J}{\partial W_{k,1}} & \cdots & \frac{\partial J}{\partial W_{k,n}} \end{pmatrix} = \begin{pmatrix} \nabla_J(W_{1,:}) \\ \vdots \\ \nabla_J(W_{k,:}) \end{pmatrix} = \begin{pmatrix} dY_1 \cdot g'(z_1) \cdot x^T \\ \vdots \\ dY_k \cdot g'(z_k) \cdot x^T \end{pmatrix} =$$

$$= \begin{bmatrix} dY_1 \\ \vdots \\ dY_k \end{bmatrix} \odot \begin{bmatrix} g'(z_1) \\ \vdots \\ g'(z_k) \end{bmatrix} \cdot x^T = dY \odot g'(z) \cdot x^T \in \mathbb{R}^{k \times n} \checkmark$$

*Handwritten notes:*  
 - Above the first vector:  $\mathbb{R}$   
 - Above the second vector:  $\mathbb{R}$   
 - Above the third vector:  $\mathbb{R}^{n \times n}$   
 - Under the first vector:  $\mathbb{R}^{k \times 1}$   
 - Under the second vector:  $\mathbb{R}^{n \times n}$   
 - Under the third vector:  $\mathbb{R}^{k \times n}$

## Redes neuronales (V): Cerrando el ciclo

¿Cómo se encadena esto? Nosotros estamos dando por conocida la derivada del error respecto de la salida de la capa,  $dY = \frac{dJ}{d\hat{y}}$ , pero en realidad no tenemos idea si estamos en una capa intermedia o no.

$$\begin{aligned}\frac{\partial \hat{y}}{\partial x_i} &= \sum_{j=1}^k dY_j \cdot g'(z_j) \cdot W_{j,i} = \left\langle \begin{pmatrix} dY_1 \cdot g'(z_1) \\ \vdots \\ dY_k \cdot g'(z_k) \end{pmatrix}, \begin{pmatrix} W_{1,i} \\ \vdots \\ W_{k,i} \end{pmatrix} \right\rangle = \\ &= \langle dY \odot g'(z), W_{:,i} \rangle = W_{i,:}^T \cdot (dY \odot g'(z))\end{aligned}$$

En forma vectorizada:

$$dX = \frac{\partial J}{\partial x} = \begin{pmatrix} \frac{\partial J}{\partial x_1} \\ \vdots \\ \frac{\partial J}{\partial x_n} \end{pmatrix} = \begin{pmatrix} W_{1,:}^T \cdot dY \odot g'(z) \\ \vdots \\ W_{n,:}^T \cdot dY \odot g'(z) \end{pmatrix} = \overbrace{W^T}^{n \times k} \cdot \underbrace{(dY \odot g'(z))}_{\in \mathbb{R}^{k \times 1}} \quad \checkmark$$

*Handwritten notes:  $n \times k$ ,  $\in \mathbb{R}^{k \times 1}$ ,  $\in \mathbb{R}^n$*

Y ese  $dX$  no es otra cosa que el  $dY$  de la capa anterior!

# Redes neuronales (VI): Bosquejo de código

```
class Layer:
    ...
    def forward(self, X):
        self.last_x = X
        self.last_z = self.W @ X + self.b
        self.last_y = self.g.f(self.last_z)
        return self.last_y

    def backwards(self, dY):
        db = dY * self.g.df(self.last_z)
        dW = db @ self.last_x.T
        dX = self.W.T @ db
        (self.W, self.b = self.optimizer.step(self.W, self.b, dW, db)

    return dX
    ...
```

Handwritten notes:

- $z = W \cdot X + b$
- $\hat{y} = g(z)$
- $!!$
- $db = dY \cdot \frac{dY}{dz} \cdot g'(z)$
- $dW = \frac{dY}{dz} \cdot g'(z) \cdot X^T$
- $dX = W^T \cdot \frac{dY}{dz} \cdot g'(z)$
- $dX_i = dY_{i-1}$

option