

# Análisis Matemático para Inteligencia Artificial

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

Especialización en Inteligencia Artificial

Clase 5

# 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  $h \rightarrow 0$ , ...

$$(x_1, y_1) ; (x_2, y_2) \quad \frac{y_2 - y_1}{x_2 - x_1}$$



... 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**,  $n > 1$ .

		$m$	
$n$	$= 1$	$= 1$	$> 1$
	$= 1$	función escalar	función vectorial
	$> 1$	campo escalar	campo vectorial

**Conjuntos de Nivel** Dada  $f : D \subset \mathbb{R}^n \rightarrow \mathbb{R}$  el conjunto de nivel  $k$  de  $f(x)$   $L_k \subset \mathbb{R}^n$ , 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.

# 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:  $\nabla f = \left( \frac{\partial f}{\partial x_1} \dots \frac{\partial f}{\partial x_n} \right)$ .  $\mathbb{R}^n \rightarrow \mathbb{R}^n$

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:



$$J_f(i, j) = \frac{\partial f_i}{\partial x_j}$$

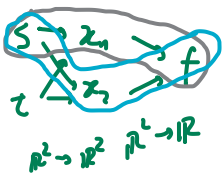
$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

$$= \begin{pmatrix} -\nabla f_1 - \\ \vdots \\ -\nabla f_m - \end{pmatrix}$$

$$\in \mathbb{R}^{m \times n}$$

# Regla de la Cadena en forma matricial

Sea  $f(x_1(s, t), x_2(s, t)) : \mathbb{R}^2 \rightarrow \mathbb{R}$



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

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

Y luego

$$\nabla f(x_1, x_2) \quad J_{x_1, x_2}(s, t)$$

$$\nabla f(s, t) = \frac{df}{d(s, t)} = \frac{df}{dx} \frac{dx}{d(s, t)} = \underbrace{\left[ \frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \right]}_{\mathbb{R}^{1 \times 2}} \underbrace{\begin{bmatrix} \frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t} \end{bmatrix}}_{\mathbb{R}^{2 \times 2}} \in \mathbb{R}^{1 \times 2}$$

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

$$f: \mathbb{R}^{n \times m} \rightarrow \mathbb{R}^{a+b}$$

$$\frac{df}{dx} \in \mathbb{R}^{a+b \times n \times m}$$

$$A \in \mathbb{R}^{4 \times 2} \quad x \in \mathbb{R}^3$$

$$f(x) = y$$

$$f: \mathbb{R}^3 \rightarrow \mathbb{R}^{4 \times 2}$$

desorden  
el in

$$A \in \mathbb{R}^{4 \times 2} \quad x \in \mathbb{R}^3$$

flatten del  
out

! (4x2)  
8x3

Partial derivatives:

$$\frac{\partial A}{\partial x_1} \in \mathbb{R}^{4 \times 2}, \quad \frac{\partial A}{\partial x_2} \in \mathbb{R}^{4 \times 2}, \quad \frac{\partial A}{\partial x_3} \in \mathbb{R}^{4 \times 2}$$

collate

$$\frac{dA}{dx} \in \mathbb{R}^{4 \times 2 \times 3}$$

(4x2)x3

(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.

$$A \in \mathbb{R}^{4 \times 2} \quad \tilde{A} \in \mathbb{R}^8$$

re-shape

gradient

re-shape

$$\frac{d\tilde{A}}{dx} \in \mathbb{R}^{8 \times 3}, \quad \frac{dA}{dx} \in \mathbb{R}^{4 \times 2 \times 3}$$

(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.

escalar  $\rightarrow$  rango 0  
vector  $\rightarrow$  " "  
matriz  $\rightarrow$  " "  
"cubo"  $\rightarrow$  " "

tensor

"rango superior"





# Diferenciación Automática: ejemplo

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

$$\frac{\partial e}{\partial x} = y, \frac{\partial e}{\partial y} = x, \frac{\partial f}{\partial x} = 3, \frac{\partial g}{\partial x} = 2x, \frac{\partial h}{\partial x} = \cos(x)$$



$$\frac{\partial s}{\partial x}, \frac{\partial s}{\partial y} \quad (3, 4)$$

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

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

$$\frac{\partial a_3}{\partial a_1} = \frac{\partial e}{\partial x}(a_1, a_2) = a_2 = 12$$

$$\frac{\partial a_3}{\partial a_1} = \dots = 9$$

$$\frac{\partial a_4}{\partial a_2} = \frac{\partial h}{\partial x}(a_2) = \cos(1) \approx 0.978$$

$$\frac{\partial a_1}{\partial x} = \frac{\partial f}{\partial x}(x) = 2 \cdot 3 = 6$$

$$\frac{\partial a_1}{\partial y} = \frac{\partial f}{\partial y}(y) = 3$$

$$\frac{\partial s}{\partial a_1} = \frac{\partial s}{\partial a_3} \cdot \frac{\partial a_3}{\partial a_1} = -0.5966 \cdot 12 = -7.15 \dots$$

$$\frac{\partial s}{\partial a_2} = \frac{\partial s}{\partial a_1} \cdot \frac{\partial a_1}{\partial a_2} + \frac{\partial s}{\partial a_4} \cdot \frac{\partial a_4}{\partial a_2} = 108 \cdot 12 + (-0.5966) \cdot 0.978 \approx 100.255$$

$$\frac{\partial s}{\partial x} = \frac{\partial s}{\partial a_1} \cdot \frac{\partial a_1}{\partial x} = -7.15 \dots \cdot 6 = -42.9 \dots$$

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

$$x=3$$

$$y=4$$

$$a_1 = g(x)$$

$$a_2 = f(y)$$

$$a_3 = e(a_1, a_2)$$

$$a_4 = h(a_2)$$

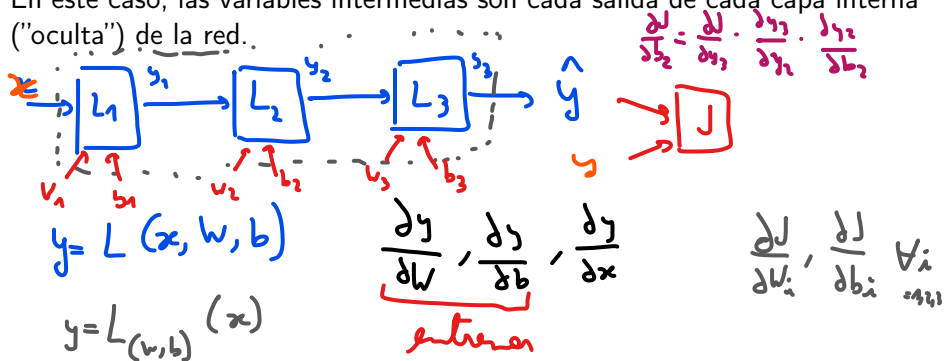
$$s = e(a_3, a_4)$$

# 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:

$$\hat{y} = g(\overbrace{w \cdot x + b}^z) \quad \text{f. escalar}$$

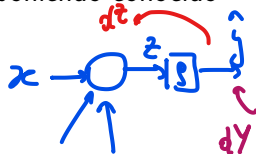
donde en su forma más simple  $x, y, w, b \in \mathbb{R}$  y  $g : \mathbb{R} \rightarrow \mathbb{R}$  es una función no lineal como puede ser la sigmoidea  $\sigma(z) = \frac{1}{1+e^{-z}}$ .

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

$$\frac{dJ}{d\hat{y}} = dY \in \mathbb{R}:$$

$$\frac{\partial \hat{y}}{\partial z} = g'(z) \in \mathbb{R}$$
$$\frac{\partial J}{\partial W} = \frac{dJ}{d\hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial W} = dY \cdot g'(z) \cdot x \in \mathbb{R}$$

$$\frac{\partial J}{\partial b} = \frac{dJ}{d\hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial b} = dY \cdot g'(z) \cdot 1 \in \mathbb{R}$$



$$dz = dY \cdot g'(z)$$

## Redes neuronales (II)

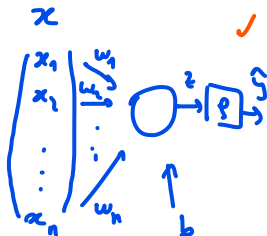
Si ahora consideramos múltiples entradas, es decir  $x \in \mathbb{R}^n$ ,  $W \in \mathbb{R}^{1 \times n}$ :

$$\hat{y} = g(\underbrace{W \cdot x}_{1 \times n \cdot n \times 1} + \underbrace{b}_{1}) \quad \text{with } z \in \mathbb{R} = y\left(\underbrace{x_1 \cdot w_1 + x_2 \cdot w_2 + \dots + x_n \cdot w_n}_{1 \times n \cdot n \times 1} + b\right)$$

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

$$\frac{\partial J}{\partial W} = \nabla_J(W) = \left( \overbrace{dY \cdot g'(z)}^{dJ/dw_1} \cdot x_1, \dots, dY \cdot g'(z) \cdot x_n \right) = \overbrace{dY}^1 \cdot \overbrace{g'(z)}^1 \cdot \overbrace{x^T}^{1 \times n} \in \mathbb{R}^{1 \times n}$$

$$\frac{\partial J}{\partial b} = dY \cdot g'(z) \cdot 1 \in \mathbb{R} \quad \checkmark$$



## 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:

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

Handwritten annotations:  $\mathbb{R}^k$  (green) for  $\hat{y}$ ,  $\mathbb{R}^n$  (orange) for  $x$ ,  $\mathbb{R}^k$  (green) for  $z$ . Dimensions are indicated below:  $k \times n$  for  $W$ ,  $n \times 1$  for  $x$ , and  $k \times 1$  for  $b$ .

$$W = \begin{pmatrix} -w_1 \\ -w_2 \\ \vdots \\ -w_k \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_k \end{pmatrix}$$
$$g(z) = \begin{pmatrix} g(z_1) \\ \vdots \\ g(z_k) \end{pmatrix}$$

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

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

$$\hat{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{pmatrix} \quad dy \in \mathbb{R}^k$$

# Redes neuronales (IV)

En el caso de  $b$  es simple:  $\overbrace{k+1}$

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

Ahora para cada elemento de  $W$  tenemos:

$$\begin{aligned} \frac{\partial J}{\partial W} &= \begin{pmatrix} \frac{\partial J}{\partial W_{1,1}} & \cdots & \frac{\partial J}{\partial W_{1,n}} \\ \frac{\partial J}{\partial W_{2,1}} & \cdots & \frac{\partial J}{\partial W_{2,n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial J}{\partial W_{k,1}} & \cdots & \frac{\partial J}{\partial W_{k,n}} \end{pmatrix} = \begin{pmatrix} \nabla_J(W_{1,:}) \\ \vdots \\ \nabla_J(W_{k,:}) \end{pmatrix} = \begin{pmatrix} dY_1 \cdot g'(z_1) \cdot x^T \\ \vdots \\ dY_k \cdot g'(z_k) \cdot x^T \end{pmatrix} = \\ &= \left[ \begin{pmatrix} dY_1 \\ \vdots \\ dY_k \end{pmatrix} \odot \begin{pmatrix} g'(z_1) \\ \vdots \\ g'(z_k) \end{pmatrix} \right] \cdot x^T = \underbrace{dY}_{k \times 1} \odot \underbrace{g'(z)}_{k \times 1} \cdot \underbrace{x^T}_{1 \times n} \in \mathbb{R}^{k \times n} \end{aligned}$$

## 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.

$$\frac{\partial \hat{y}}{\partial x_i} = \sum_{j=1}^k \overbrace{dY_j}^{\frac{dJ}{d\hat{y}_j}} \cdot \overbrace{g'(z_j)}^{\frac{dz_j}{d\hat{y}_j}} \cdot \overbrace{W_{j,i}}^{\frac{dz_j}{dx_i}} = \left\langle \begin{pmatrix} dY_1 \cdot g'(z_1) \\ \vdots \\ dY_k \cdot g'(z_k) \end{pmatrix}, \begin{pmatrix} W_{1,i} \\ \vdots \\ W_{k,i} \end{pmatrix} \right\rangle =$$
$$= \langle dY \odot g'(z), W_{:,i} \rangle = \underbrace{W_{i,:}^T}_{i\text{-ésima fila de } W^T} \cdot [dY \odot g'(z)]$$

En forma vectorizada:

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

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
...
```

$z = W \cdot x + b$

$y = g(z)$

$dY \odot g'(z) = db$

$dW = (dY \odot g'(z)) \cdot X^T$

$dX = W^T \cdot (dY \odot g'(z))$

$\downarrow$  update de los  $W, b$

$\hookrightarrow$  es la  $dY$  de la capa anterior