

Análisis Matemático para Inteligencia Artificial

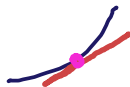
Verónica Pastor (vpastor@fi.uba.ar),
Martín Errázquin (merrazquin@fi.uba.ar)

Especialización en Inteligencia Artificial

18/11/2022

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 $f(x_0 + h) - f(x_0)$.



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

- 3 Mientras que la variación media es x_0 :

$$\frac{f(x_0+h)-f(x_0)}{(x_0+h)-x_0}.$$

Geométricamente, podemos ver la variación media como la pendiente de la recta secante.

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

- 4 Cuando hacemos que $h \rightarrow 0$, ...

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, $n > 1$.

$f: \mathbb{R} \rightarrow \mathbb{R}, f(x) = x^2 \Rightarrow f'(x) = 2x$

$f: \mathbb{R}^2 \rightarrow \mathbb{R}, f(x, y) = x^2 + y^2$



$f: \mathbb{R} \rightarrow \mathbb{R}^m, f(t) = (t^2, t+2) \Rightarrow f'(t) = (2t, 1)$

$f: \mathbb{R}^2 \rightarrow \mathbb{R}^3, f(x, y) = (x^2, 0, x\sqrt{y})$

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:



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

$f(x, y) = x^2 + y^2$

$L_1: x^2 + y^2 = 1$

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.

$L_0: x^2 + y^2 = 0$ $L_2: x^2 + y^2 = 2 = (\sqrt{2})^2$ $L_{-1}: x^2 + y^2 = -1$

$L_{-1} = \emptyset$



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:

$$f(x,y) = x^2 + y^2$$

$$\frac{\partial f}{\partial x}(x,y) = 2x$$

$$\frac{\partial f}{\partial y}(x,y) = 2y$$

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

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

Se define el **gradiente** como: $\nabla f_{(x_1, \dots, x_n)} = \left(\frac{\partial f}{\partial x_1} \dots \frac{\partial f}{\partial x_n} \right)$ **vector**

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:

$$f(x,y) = (x^2, 0, x\sqrt{y})$$
$$Jf = \begin{bmatrix} 2x & 0 \\ 0 & 0 \\ \sqrt{y} & x/2\sqrt{y} \end{bmatrix}$$

MATRIZ

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

m x n

$\partial f_i / \partial x_j$

Regla de la Cadena en forma matricial

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



Y luego

$$\frac{df}{d(s, t)} = \frac{df}{dx} \frac{dx}{d(s, t)} = \underbrace{\left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \right]}_{\nabla f} \underbrace{\begin{bmatrix} \frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t} \end{bmatrix}}_{J(x_1, x_2)(s, t)}$$

Recordemos reglas de derivación:

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

$$f(x) = \overline{\sin(2x^3)}$$

Composición de funciones

$$g(y) = \sin(y)$$

$$h(z) = 2z$$

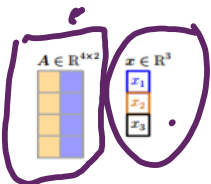
$$m(n) = n^3$$

$$f(x) = g(h(m(x)))$$

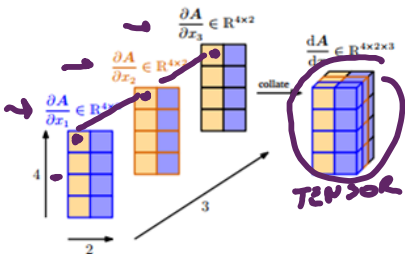
$$f' = g'(y) h'(z) \cdot m'(x) \cdot 1$$

$$f'(x) = \cos(2x^3) \cdot 6x^2$$

Derivada de matrices



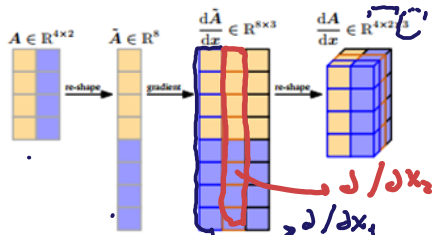
Partial derivatives:



(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 4×2 matrix, and collate them in a $4 \times 2 \times 3$ tensor.



$p_2^{big} \sim \mathbb{R}^3$
 ev. dim 3 \rightarrow ev. dim 3
 $p(x) = x^2 + 2$ $r = (1, 0, 2)$



(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.

Matriz Hessiana

$$f(x) = x^2 \rightarrow f'(x) = 2x \stackrel{\downarrow \text{PUNTO CRÍTICO}}{=} 0 \rightarrow f''(x) = 2 > 0 \stackrel{\text{MIN}}{\circlearrowright}$$

La **matriz Hessiana** es aquella cuyas derivadas de orden 2 de f respecto a $x \in \mathbb{R}^n$ se ubican:

$$f: \mathbb{R}^n \rightarrow \mathbb{R}$$

$$\frac{\partial}{\partial x_1} \left(\frac{\partial f}{\partial x_1} \right) = \frac{\partial^2 f}{(\partial x_1)^2}$$

$$\frac{\partial}{\partial x_n} \left(\frac{\partial f}{\partial x_1} \right) = \frac{\partial^2 f}{\partial x_n \partial x_1}$$

$$H = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

TEOREMA

Si $f \in C^2$ (tiene deriv. parciales cont. y derivables) entonces
 LAS DERIVADAS CRUZADAS SON IGUALES \Rightarrow **Simétrica**

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}$$

$$\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

derivar primero
resp. a x_1

$$\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y}$$

$$f(x,y) = x^2 + y^2 \quad \nabla f(2x, 2y)$$

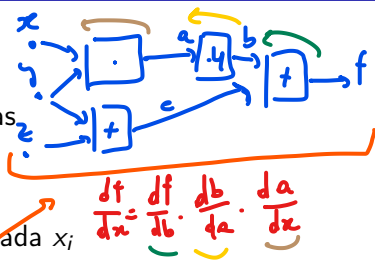
$$\frac{\partial^2 f}{\partial x} (x,y) = 2 \quad \frac{\partial^2 f}{\partial y \partial x} = 0$$

$$\frac{\partial^2 f}{\partial y^2} (x,y) = 2 \quad \frac{\partial^2 f}{\partial x \partial y} (x,y) = 0$$

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:

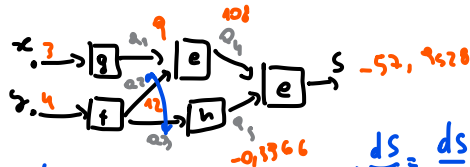
$$\frac{\partial f}{\partial x_i} = \sum_{x_j \in Hij(x_i)} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i} = \sum_{x_j \in Hij(x_i)} \frac{\partial f}{\partial g_j} \frac{\partial x_j}{\partial x_i}$$

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

$$\frac{de}{dx} = y, \quad \frac{de}{dy} = x, \quad \frac{df}{dx} = 3, \quad \frac{dg}{dx} = 2x, \quad \frac{dh}{dx} = \cos(x)$$



$$\frac{ds}{de} = 1$$

$$\frac{ds}{da_5} = a_5 = -0.5966$$

$$\frac{ds}{da_4} = a_4 = 108$$

$$\frac{ds}{da_1} = \frac{ds}{da_5} \cdot \frac{da_5}{da_1} = a_5 \cdot a_2 = -7.1592$$

$$\begin{aligned} \frac{ds}{da_2} &= \frac{ds}{da_5} \cdot \frac{da_5}{da_2} + \frac{ds}{da_4} \cdot \frac{da_4}{da_2} \\ &= a_5 \cdot a_1 + a_4 \cdot \cos(a_2) \end{aligned}$$

$$\frac{ds}{dx} = \frac{ds}{da_1} \cdot \frac{da_1}{dx} = a_5 \cdot a_2 \cdot 2x = -7.1592 \cdot 3$$

$$\frac{ds}{dy} = \frac{ds}{da_2} \cdot \frac{da_2}{dy} = (a_5 \cdot a_1 + a_4 \cdot \cos(a_2)) \cdot 3$$

$$x=3, y=4$$

$$a_1 = g(x)$$

$$a_2 = f(y)$$

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

$$a_4 = h(a_3)$$

$$a_5 = s$$

$$s = e(a_4, a_5)$$

$$s.backward()$$

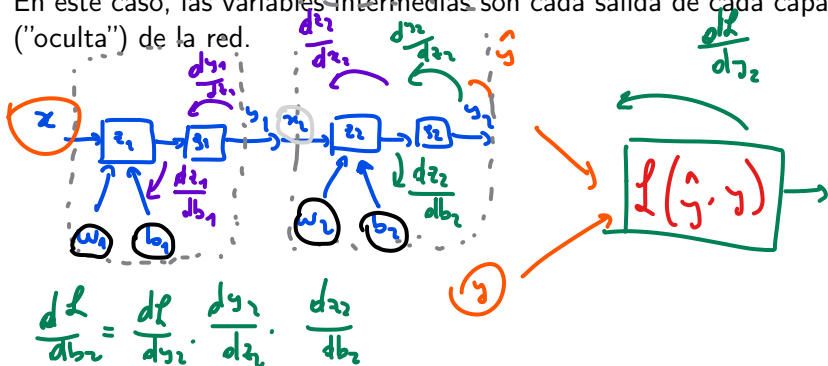
$$\frac{ds}{dx}, \frac{ds}{dy}$$

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.

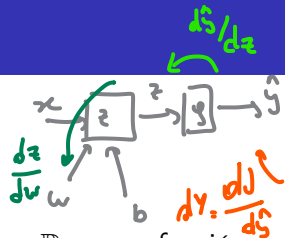
En este caso, las variables intermedias son cada salida de cada capa interna ("oculta") de la red.



Redes neuronales (I)

Un perceptrón/neurona es un estimador de la forma:

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



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

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

$$\begin{aligned} \left[\frac{\partial \hat{y}}{\partial z} = g'(z) \right] \\ \frac{\partial J}{\partial W} = \frac{dJ}{d\hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial W} = \overbrace{dY}^1 \cdot \overbrace{g'(z)}^1 \cdot \overbrace{x}^1 \in \mathbb{R}^1 \\ \frac{\partial J}{\partial b} = \frac{dJ}{d\hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial b} = \overbrace{dY}^1 \cdot \overbrace{g'(z)}^1 \cdot \overbrace{1}^1 \in \mathbb{R}^1 \end{aligned}$$

Redes neuronales (II)

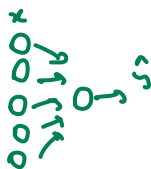
Si ahora consideramos múltiples entradas, es decir $x \in \mathbb{R}^n$, $W \in \mathbb{R}^{1 \times n}$:

$$\hat{y} = g(W \cdot x + b) \quad \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

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

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

$$\frac{\partial J}{\partial b} = \underbrace{dY}_{1} \cdot \underbrace{g'(z)}_1 \in \mathbb{R}^1$$



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:

$$W = \begin{pmatrix} -w_1 \\ \vdots \\ -w_k \end{pmatrix}$$

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

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:

element-wise Prod.

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

Ahora para cada elemento de W tenemos:

$k \times n$

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

Redes neuronales (V): Backpropagation

¿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.

$$\frac{\partial \hat{y}}{\partial x_i} = \sum_{j=1}^k \underbrace{\frac{\partial J}{\partial y_j}}_{dY_j} \cdot \underbrace{\frac{dy_j}{dx_i}}_{g'(z_j)} \cdot W_{j,i} = \left\langle \begin{pmatrix} dY_1 \cdot g'(z_1) \\ \vdots \\ dY_k \cdot g'(z_k) \end{pmatrix}, \begin{pmatrix} W_{1,i} \\ \vdots \\ W_{k,i} \end{pmatrix} \right\rangle =$$

$$= \langle dY \odot g'(z), W_{:,i} \rangle = W_{i,:}^T \cdot dY \odot g'(z)$$

En forma vectorizada:

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

Handwritten notes: $\angle a, b \rangle = a^T \cdot b = b^T \cdot a$

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