Análisis Matemático para Inteligencia Artificial

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

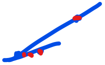
Especialización en Inteligencia Artificial

Clase 5

Análisis Matemático

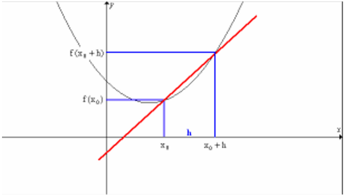
Repaso

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 $h \to 0$, ...



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

$x_2 = x_1+h$



... esto nos conduce a la definición de derivada de f en x_0:

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

Clasificación de funciones

$(x,y,z) \to (x\cos(z), y\,\mathrm{sen}(z))$

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

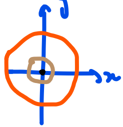
- Si $m = 1$ diremos que es una función
  - escalar, si $n = 1$,
  - campo escalar, $n > 1$.
- Si $m > 1$ diremos que es una función
  - vectorial, si $n = 1$,
  - campo vectorial, $n > 1$.

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

$\mathbb{R}^n \to \mathbb{R}$

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

$f: \mathbb{R}^2 \to \mathbb{R}$



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

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.

@mapa_fisico $f: (\mathrm{long}, \mathrm{lat}) \to h$

Derivando campos ...

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

n deriv. parciales

$$\frac{\partial f}{\partial x_1} = lim_{h\to 0}\frac{f(x_1+h, x_2, ..., x_n) - f(x_1, x_2, ..., x_n)}{h} \quad \in \mathbb{R}$$

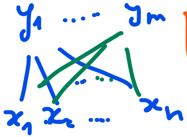
$$\vdots \qquad \vdots$$

$$\frac{\partial f}{\partial x_n} = lim_{h\to 0}\frac{f(x_1, x_2, ..., x_n+h) - f(x_1, x_2, ..., x_n)}{h} \quad \in \mathbb{R}$$

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

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

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



n · m deriv. parciales

$$J_{f_{ij}} = \frac{\partial f_i}{\partial x_j}$$

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ & \vdots & \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} = \begin{pmatrix} -\nabla f_1- \\ \vdots \\ -\nabla f_m- \end{pmatrix} \in \mathbb{R}^{m \times n}$$

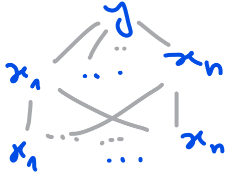
Matriz Hessiana

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

$f : \mathbb{R}^n \to \mathbb{R}$ ← campo escalar

$\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ ← campo vectorial

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



n

n

n^2

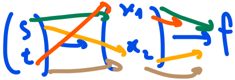
$$H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

Una aplicación común del Hessiano es el polinomio de Taylor de orden 2 para campos escalares. Sea f un campo escalar $f : \mathbb{R}^n \to \mathbb{R}$, asumiendo que posee derivadas parciales de todo orden en un entorno de un punto $a \in \mathbb{R}^n$, se define el polinomio de Taylor de orden 2:

$$P_2(x) = f(a) + \nabla_f(a)^T(x-a) + \frac{1}{2}(x-a)^T H_f(a)(x-a) \quad \in \mathbb{R}$$

Regla de la Cadena en forma matricial

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



$\mathbb{R}^2 \to \mathbb{R}^2$ $\quad$ $\mathbb{R}^2 \to \mathbb{R}$

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

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

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

Y luego

$$\nabla f_{(x_1,x_2)} \cdot J_x(s,t)$$

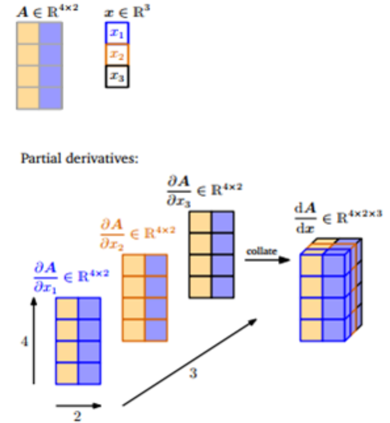
$$\frac{df}{d(s,t)} = \frac{df}{dx}\frac{dx}{d(s,t)} = \underbrace{\begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix}}_{1\times 2} \underbrace{\begin{bmatrix} \frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial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

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

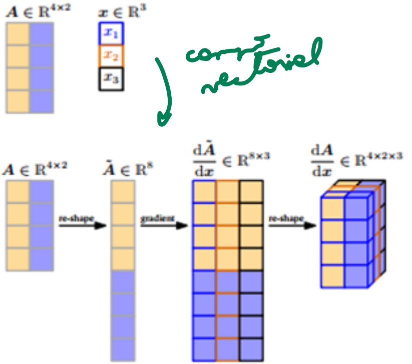
$f: \mathbb{R}^{a\times b\times c} \to \mathbb{R}^{x\times y} \Rightarrow \frac{\partial f}{\partial x} \in \mathbb{R}^{(a\times b\times c\times x\times y)}$



(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.

plantear $A(x_i)$ $\forall i$, calcula $\frac{\partial A}{\partial x_i}$

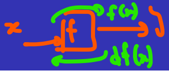
cambio vectorial



(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.

1) flatten A
2) op. matriciales
3) recomponer el tensor

Diferenciación Automática

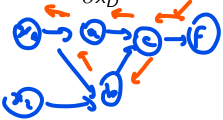



```
class Square:
    def fw(x):
        return x**2
    def bw(x):
        return 2*x
```

Sean, para una función f:

- $x_1, \ldots, x_d$ las variables de entrada
- $x_{d+1}, \ldots, x_{D-1}$ 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:



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

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial c} \cdot \frac{\partial c}{\partial a}$$

- 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: idea gráfica

$y = x^{\cos^2(x)}$

$\frac{\partial y}{\partial b} = x^b \log(x)$

sin el fw no podría evaluar b

- $\frac{\partial y}{\partial y} = 1$
- $\frac{\partial y}{\partial b} = \frac{\partial y}{\partial y} \cdot \frac{\partial y}{\partial b} = 1 \cdot 2^{0.9987} \cdot \log(2) =$
- $\frac{\partial y}{\partial a} = \frac{\partial y}{\partial b} \cdot \underbrace{\frac{\partial b}{\partial a}}_{2a} = \cdot 2 \cdot 0.9993 =$
- $\boxed{\frac{\partial y}{\partial x}} = \frac{\partial y}{\partial a} \cdot \frac{\partial a}{\partial x} + \frac{\partial y}{\partial y} \cdot \underbrace{\frac{\partial y}{\partial x}}_{b \cdot x^{b-1}} =$

$= \cdot -\sin(2) + 1 \cdot 0.9987 \cdot 2^{-0.0013}$

x

$a = \cos(x)$

$b = a^2$

$y = x^b$

$$\begin{tikzcd} x \arrow[r] \arrow[rrr, bend right] & a \arrow[r] & b \arrow[r] & y \end{tikzcd}$$

en $x = 2$

$a = \cos(2) = 0.9993$

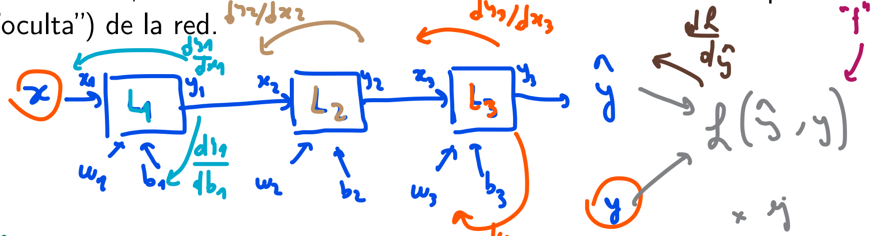
$b = 0.9993^2 = 0.9987$

$y = 2^{0.9987} = 1.9981$

¿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.



$$\frac{d\ell}{db_1} = \frac{d\ell}{d\hat{y}} \frac{dy_3}{dx_3} \frac{dy_2}{dx_2} \frac{dy_1}{db_1}$$

$$\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

$$\frac{d\ell}{d\hat{y}} = \hat{y} - y$$

Redes neuronales (0): Bosquejo de código

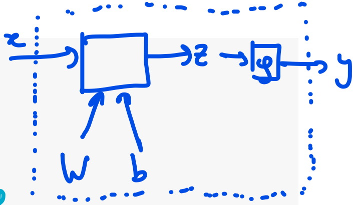
```python
class Layer:
  ...
  def forward(self, X):
    self.last_x = X
    self.last_z = self.W @ X + self.b
    self.last_y = self.g.f(self.last_z)
    return self.last_y

  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
  ...
```

```python
class Sigmoid:
  def f(self, x):
    return 1/(1+exp(-x))
  def df(self, x):
    s= 1/(1+exp(-x))
    return s*(1-s)
```

$\sigma(x)$

$\sigma'(x)$

accum_df $\frac{d\ell}{dy_n}$



$z = W \cdot x + b$

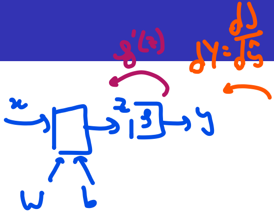
$y = g(z)$

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

$dW = db \cdot X^T$ $\quad dX = W^T \cdot db$

paso de optim

devuelvo el accum-df $\frac{d\ell}{dy_{n-1}}$



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

$$\frac{\partial \hat{y}}{\partial z} = g'(z)$$

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

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

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

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

$z = w_1x_1 + w_2x_2 + \dots + w_nx_n + b$

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

$dY \cdot g'(z) \cdot (x_1, \dots, x_n)$

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

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

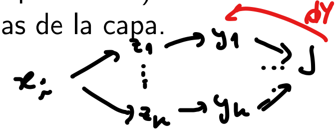
$$\frac{\partial J}{\partial w_i} = dY \cdot g'(z) \cdot x_i$$

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.

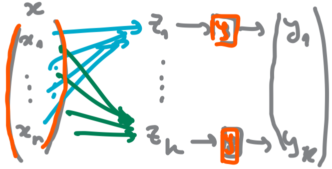
Nuevamente tenemos:

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



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?



$y \in \mathbb{R}^k$

$W = \begin{pmatrix} — w_1 — \\ — w_2 — \\ \vdots \\ — w_k — \end{pmatrix}$ $b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{pmatrix}$

$w_1 \in \mathbb{R}^{1 \times n}$

$b_k \in \mathbb{R}$

Redes neuronales (IV)

$$\begin{cases} dY_1 \cdot g'(z_1) \cdot 1 \\ \vdots \\ dY_k \cdot g'(z_k) \cdot 1 \end{cases}$$

En el caso de b es simple:

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

Ahora para cada elemento de W tenemos:

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

$$= \left(\begin{pmatrix} dY_1 \\ \vdots \\ dY_k \end{pmatrix} \odot \begin{pmatrix} g'(z_1) \\ \vdots \\ g'(z_k) \end{pmatrix} \right) x^T = \underbrace{\overset{\mathbb{R}^k}{dY} \odot \overset{\mathbb{R}^k}{g'(z)}}_{dz \in \mathbb{R}^{k \times 1}} \cdot \overbrace{x^T}^{\mathbb{R}^{1 \times n}} \in \mathbb{R}^{k \times n}$$

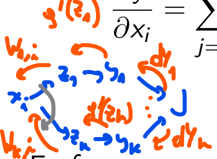
Redes neuronales (V): Cerrando el ciclo

¿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.*

col i de W

$$\frac{\partial \hat{y}}{\partial x_i} = \sum_{j=1}^{k} \overbrace{dY_j}^{\mathbb{R}} \cdot \overbrace{g'(z_j)}^{\mathbb{R}} \cdot \overbrace{W_{j,i}}^{\mathbb{R}} = \langle \begin{pmatrix} dY_1 \cdot g'(z_1) \\ \vdots \\ dY_k \cdot g'(z_k) \end{pmatrix}, \begin{pmatrix} W_{1,i} \\ \vdots \\ W_{k,i} \end{pmatrix} \rangle =$$

$dz \in \mathbb{R}^k$



$$= \langle \underbrace{dY \odot g'(z)}_{dz}, W_{:,i} \rangle = \underbrace{W_{i,:}^T}_{\text{fila } i \text{ de } W^T} \cdot (dY \odot g'(z)) \quad dz \in \mathbb{R}^k$$

$\in \mathbb{R}$

En forma vectorizada:

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

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