Análisis Matemático para Inteligencia Artificial

Martín Errázquin (merrazquin@fi.uba.ar)

Especialización en Inteligencia Artificial

Clase 5

Análisis Matemático

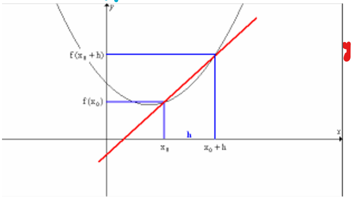
Repaso

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 $h \to 0$, ...

(x_0, y_0) (x_1, y_1) $\Delta y = f(x_0+h) - f(x_0)$ $y_1 := f(x_1)$

$x_1 = x_0 + h$

h



$y = \frac{\Delta y}{\Delta x} x + b$

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

$$\underbrace{lim_{h\to 0}}_{h \to 0} \frac{\overbrace{f(x_0 + h) - f(x_0)}^{\Delta y}}{\underbrace{h}_{\Delta x}}$$

Clasificación de funciones

$0 \leq t < 2\pi$ $\quad \vec{x}(t) = (\cos(t), \sin(t))$

$\mathbb{R} \to \mathbb{R}$

$\vec{x}'(t) = (-\sin(t), \cos(t))$

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, $n > 1$.

(in) n \ (out) m	esc (=1)	vect. (>1)
escalar (=1)	función escalar	función vectorial
vector (>1)	$\boxed{\text{campo escalar}}$	campo vectorial

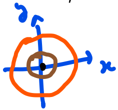
$h(\vec{x}) = \|\vec{x}\|$

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

$g(x,y) = (x+y, x-y)$

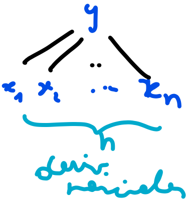
Conjuntos de Nivel Dada $f : D \subset \mathbb{R}^n \to \mathbb{R}$ el conjunto de nivel k de f, $L_k \subset \mathbb{R}^n$, definido por:

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



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.

- 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:



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

$$\vdots$$

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

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

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

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

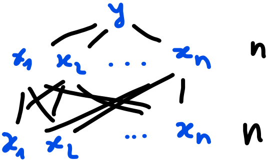
$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ & \vdots & \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} = \begin{pmatrix} -\nabla f_1- \\ \vdots \\ -\nabla f_m- \end{pmatrix} \in \mathbb{R}^{m\times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

$J_{i,j} = \frac{\partial f_i}{\partial x_j}$

Matriz Hessiana

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

bajo hip.



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

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

$\Rightarrow$ H simétrica

n^2 deriv. parc. segundas $\frac{\partial^2 f}{\partial x_i \partial x_j}$

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:

$x - a \in \mathbb{R}^n$

$\nabla f \in \mathbb{R}^n$

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

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

Regla de la Cadena en forma matricial

$J \qquad \nabla$

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

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

$x_1 : \mathbb{R}^2 \to \mathbb{R}$

$x_2 : \mathbb{R}^2 \to \mathbb{R}$

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

$\begin{pmatrix} s \\ t \end{pmatrix} \to \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \to f$

$\nabla f(s,t)$

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

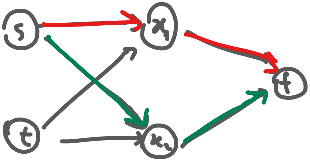
$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t}$$

Y luego

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

Recordemos reglas de derivación:

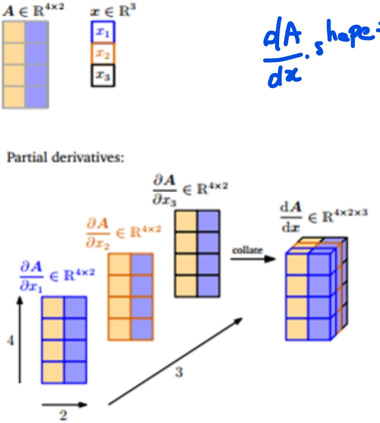
- $\frac{\partial (f+g)(s)}{\partial s} = \frac{\partial f}{\partial s} + \frac{\partial g}{\partial s}$
- $\frac{\partial (fg)(s)}{\partial s} = \frac{\partial f}{\partial s} g(s) + f(s)\frac{\partial g}{\partial s}$



Derivada de matrices

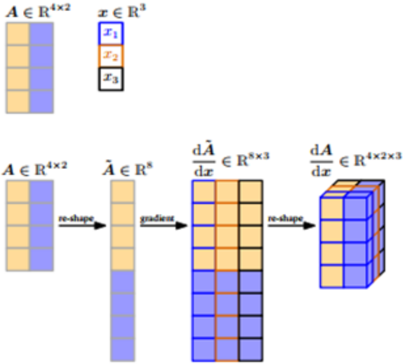
A.shape=(4,2)   x.shape=(3,)

$\frac{dA}{dx}$.shape=(4,2,3)



(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.

- descompongo el in en escalares
- (deriv)
- recompilo en nuevo eje (apilo)



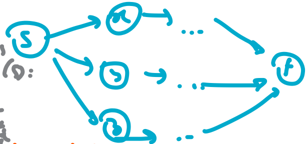
(b) Approach 2: We re-shape (flatten) $A \in \mathbb{R}^{4\times2}$ into a vector $\tilde{A} \in \mathbb{R}^{8}$. Then, we compute the gradient $\frac{d\tilde{A}}{dx} \in \mathbb{R}^{8\times3}$. We obtain the gradient tensor by re-shaping this gradient as illustrated above.

- aplano A (flatten)
- (deriv)→) …
- recompongo en el orden inverso del flatten

Diferenciación Automática

Sean, para una función f:

ds=0
for i in hijos(s):
ds+= $\frac{df}{dx}$



- $x_1, \ldots, x_d$ 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

$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x} \cdot \frac{\partial x}{\partial s} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial s} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial s}$

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 f(x)= x**2
def df(x)= 2*x

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

- 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.

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

@ $x = 1$...

x

$a = \cos(x)$

$b = a^2$

$y = x^b$

- $\dfrac{dy}{dy} = 1$
- $\dfrac{dy}{db} = \dfrac{dy}{dy} \cdot \dfrac{dy}{db} = 1 \cdot 1^0 \cdot \log(1) = 0$

$\dfrac{d(x^b)}{db} = x^b \log(x)$

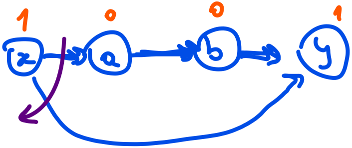
- $\dfrac{dy}{da} = \dfrac{dy}{db} \cdot \dfrac{db}{da} = 0 \cdot 2 \cdot 0 = 0$

$2a$

- $\dfrac{dy}{dx} = \dfrac{dy}{da} \cdot \dfrac{da}{dx} + \text{“}\dfrac{dy}{dx}\text{”} = 0 \cdot -\sin(1) + 0 = 0$

$-\sin(1)$

$\dfrac{d(x^b)}{dx} = b \cdot x^{b-1} = 0 \cdot 1^{-1} = 0$



$x = 1$

$a = \cos(x) = 0$

$b = 0^2 = 0$

$y = 1^0 = 1$

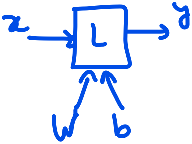
Backpropagation

¿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.

$\frac{\partial \mathcal{L}}{\partial W_1}, \frac{\partial \mathcal{L}}{\partial W_2}, \frac{\partial \mathcal{L}}{\partial W_3}, \frac{\partial \mathcal{L}}{\partial b_1}, \frac{\partial \mathcal{L}}{\partial b_2}, \frac{\partial \mathcal{L}}{\partial b_3}$

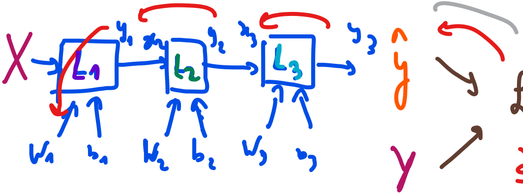


$y = L(x, W, b)$

$x \in \mathbb{R}^n$

$W \in \mathbb{R}^{k \times n}, \quad b \in \mathbb{R}^k$

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



$\mathcal{L}(\hat{y}, y) \in \mathbb{R}$

$$\frac{\partial \mathcal{L}}{\partial b_1} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial y_3}{\partial x_3} \cdot \frac{\partial y_2}{\partial x_2} \cdot \frac{\partial y_1}{\partial b_1}$$

Sea $X \in \mathbb{R}^n$, $W \in \mathbb{R}^{k\times n}$, $b \in \mathbb{R}^k$ y $g : \mathbb{R} \to \mathbb{R}$ no lineal, si se recibe $\frac{dJ}{dy} \in \mathbb{R}^k$ entonces:

$X = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$ $\quad W = \begin{pmatrix} 5 & 2 \\ 2 & 1 \\ 1 & 0 \end{pmatrix}$ $\quad b = \begin{pmatrix} 3 \\ 3 \\ 2 \end{pmatrix}$

$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \odot \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 1\cdot 3 \\ 2\cdot 4 \end{pmatrix} = \begin{pmatrix} 3 \\ 8 \end{pmatrix}$

$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 4 \end{pmatrix} = 1\cdot 3 + 2\cdot 4 = 11$

$$\frac{dJ}{db} = dY \odot g'(z)$$

$$\frac{dJ}{dW} = dY \odot g'(z) \cdot X^T$$

$$\frac{dJ}{dX} = W^T \cdot dY \odot g'(z)$$

$z = W\cdot X + b = \begin{pmatrix} 16 \\ 7 \\ 2 \end{pmatrix} + \begin{pmatrix} 3 \\ 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 19 \\ 10 \\ 4 \end{pmatrix}$

$g(z) = \text{máx}(z, 0)$

$y = g(z) = \begin{pmatrix} 19 \\ 10 \\ 4 \end{pmatrix}$

donde $z = W \cdot X + b \in \mathbb{R}^k$ e $y = g(z)$ con g aplicada elemento a elemento.
Luego como $y^{(m)} = x^{(m+1)}$, entonces $\frac{dJ}{dX^{(m+1)}} = \frac{dJ}{dy^{(m)}}$ que es lo que le pasamos a la capa anterior.

Redes neuronales (pre-II): 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
  ...
```

$dY \odot g'(z)$

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

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

clase 6