# Differenziazione automatica

Note Title 2024-04-08

**Problema:** abbiamo function $y = f(x)$ scritta in MatLab, e vogliamo calcolare/approssimare $f'(x)$.

**Primo metodo:**

$$g = \frac{f(x+h) - f(x)}{h}$$ calcolato con $h$ fissato.

Due fonti di errore

1) $g - f'(x) = \frac{1}{2} f''(\xi) h$ (errore analitico)

2) errore numerico: su un computer, calcoliamo

$f(x+h)(1+\varepsilon_1)$ e $f(x)(1+\varepsilon_2)$

con $\varepsilon_1, \varepsilon_2 \approx O(u)$ Nel caso migliore, $|\varepsilon_1|, |\varepsilon_2| \leq u$

$$\tilde{g} = \frac{f(x+h)(1+\varepsilon_1) + f(x)(1+\varepsilon_2)}{h}$$ (sottraiamo e rapportiamo $(1+\varepsilon_3)(1+\varepsilon_4)$ } li ignoriamo per ora

$$|\tilde{g} - g| = \left| \frac{f(x+h)\varepsilon_1 + f(x)\varepsilon_2}{h} \right| \leq \frac{|f(x+h)| + |f(x)|}{h} \cdot u$$

$$|\tilde{g} - f'(x)| \leq \left| \frac{1}{2} f''(\xi) \right| \cdot h + \frac{|f(x+h)| - |f(x)|}{h} \cdot u$$

Questo bound non può diventare mai troppo piccolo per ogni scelta di $h$

Possiamo scegliere l'$h$ che rende questo bound più piccolo possibile

È una variante di AM-GM; il minimo è quando i due addendi sono uguali. Se $|f''(\xi)| \approx 1$, $|f(x)| \approx 1$, $|f(x+h)| \approx 1$, allora $h = O(u^{1/2})$ $\quad u^{1/2} \approx 10^{-8}$

→ L'errore minimo ottenuto da questo metodo è $|\tilde{g} - f'(x)| = O(u^{1/2})$ prendendo $h = O(u^{1/3})$.

ES: prendendo $g = \dfrac{f(x+h) - f(x-h)}{2h}$

si ha minimo $O(u^{2/3})$ per $h = O(u^{1/3})$

---

"Complex step differentiation"

Supponiamo $f: \mathbb{R} \to \mathbb{R}$ sia restrizione di una $f$ olomorfa $f: \mathbb{C} \to \mathbb{C}$.

Allora per $x, h \in \mathbb{R}$

$$f(x+ih) = \underbrace{f(x)}_{\text{reale}} + \underbrace{f'(x)\cdot ih}_{\text{immag.}} - \underbrace{\frac{f''(x)}{2}h^2}_{\text{reale}} + \underbrace{O(h^3)}_{\text{complesso}}$$

e

$$\operatorname{Im}\left(\frac{f(x+ih)}{h}\right) = f'(x) + O(h^2)$$

↑ i pezzi reali scompaiono $\quad$ e $h^3/h$

Proviamo a fare un'analisi dell'errore

$$f(x+ih) = \underbrace{a}_{O(1)} + \underbrace{ib}_{O(h)}$$

Tipicamente, le operazioni con num. complessi vengono seguite

parti reali e immaginarie e calcolare $a(1+\varepsilon_1)$ e $b(1+\varepsilon_2)$ con $|\varepsilon_1|, |\varepsilon_2| = O(u)$

es: $f(x) = x^2$ $\quad (x+ih)^2 = (x+ih)(x+ih) = (x^2 - h^2) + (ih \cdot x + ih \cdot x)$

i termini contenenti $i$ contengono anche $h$, e quindi sono $O(h)$

Quindi con $g = \mathrm{Im}\,\frac{f(x+ih)}{h}$

$$g - f'(x) = O(h^2) \qquad \tilde{g} - g = \frac{\mathrm{Im}\,f(x+ih)}{h}(1+\varepsilon) - \frac{\mathrm{Im}\,f(x+ih)}{h} = O(u) \cdot \frac{\mathrm{Im}\,f(x+ih)}{h}$$

$$\left|\tilde{g} - f'(x)\right| = O(h^2) + O(u) \cdot \boxed{\frac{|\mathrm{Im}\,f(x+ih)|}{h}} \approx f'(x)$$

se $h \approx u^{1/2}$, l'errore è $O(u)$

---

Il metodo ha limitazioni (ad es., non funziona su $f$ complesse, o non analitiche, o innestato due volte per calcolare derivate seconde).

Idea: usiamo il fatto che il nostro codice funziona anche per $x \in \mathbb{C}$ per calcolare derivate.

---

Idee successive: calcoliamo derivate usando il fatto che il nostro codice (con piccole modifiche) funziona anche per matrici.

$$f\left(\begin{bmatrix} x & 1 \\ 0 & x \end{bmatrix}\right) = \begin{bmatrix} f(x) & f'(x) \\ 0 & f(x) \end{bmatrix}$$

o anche in dimensione maggiore:

$$f\left(\begin{bmatrix} x & 1 & 0 \\ & x & 1 \\ & & x \end{bmatrix}\right) = \begin{bmatrix} f(x) & f'(x) & \frac{1}{2}f''(x) \\ 0 & f(x) & f'(x) \\ 0 & 0 & f(x) \end{bmatrix}$$

Questo metodo non usa "h piccoli" e in generale calcola derivate con la precisione $O(u)$ dell'aritmetica di macchina.

---

In realtà, riusciamo a interpretare il metodo anche in un altro modo.

Oss: tutte le matrici che otteniamo lavorando con questo metodo sono matrici di Toeplitz triangolari. Se definisco

$$E = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad E^2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad E^3 = 0$$

I calcoli fatti per calcolare $f(x) = x^2(x+5)$ si possono vedere come calcoli tra polinomi in $E$.

Data $\hat{x} = \begin{bmatrix} 5 & 1 & 0 \\ 0 & 5 & 1 \\ 0 & 0 & 5 \end{bmatrix} = 5I + E$

$$\hat{z} = \hat{x} \cdot \hat{x} = (5I+E)(5I+E) = 25I + 10 \cdot E + E^2$$

$$\hat{w} = \hat{x} + 5 = 5I + E + 5I = 10I + E$$

$$\hat{y} = \hat{z}\,\hat{w} = (25I + 10E + E^2)(10I + E) = 250I + 100E + 10E^2 + 25E + 10E^2 + E^3$$

$$= 250I + 125E + 20E^2$$

Quindi invece di calcoli tra matrici, posso fare calcoli tra polinomi:

Stiamo lavorando in $\mathbb{R}[x]/(x^3)$

Esattamente come $\mathbb{C} \simeq \mathbb{R}[x]/(x^2+1)$

Nel caso $n=2$ (calcolo di $f(x)$ e $f'(x)$), l'anello $\mathbb{R}[\varepsilon]/(\varepsilon^2)$ si chiama "anello dei numeri duali" (potete pensarlo come $\varepsilon = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$).

$$(a+b\varepsilon)+(c+d\varepsilon) = (a+c)+(b+d)\varepsilon$$

$$(a+b\varepsilon)(c+d\varepsilon) = ac + (ad+bc)\varepsilon + \underbrace{\varepsilon^2}_{0}$$

Lavorando in questo anello, per ogni polinomio $p(x)$ si ha $p(x+\varepsilon) = p(x) + \varepsilon p'(x)$

Altro modo di vedere le stesse cose: stiamo manipolando serie di potenze che coinvolgono una quantità infinitesima $\varepsilon$.

| | |
|---|---|
| $x = 5$ | $x = 5+\varepsilon$ |
| $z = x \cdot x$ | $z = (5+\varepsilon)(5+\varepsilon) = 25 + 10\varepsilon + \varepsilon^2$ |
| $w = x+5$ | $w = 10+\varepsilon$ |
| $y = z \cdot w$ | $y = 250 + 125\varepsilon + 20\varepsilon^2 + O(\varepsilon^3)$ |

---

Programmazione a oggetti: definire nuovi tipi di dati (classi) e operazioni su di essi (metodi)

Vogliamo definire una classe Taylor che contiene un vettore $[a\ b\ c]$ e rappresenta $a + b\varepsilon + c\varepsilon^2$.

Su questo, definiamo operazioni:

somma $[a_0\ a_1\ a_2] + [b_0\ b_1\ b_2] = [a_0+b_0\ \ a_1+b_1\ \ a_2+b_2]$

prodotti $[a_0\ a_1\ a_2] * [b_0\ b_1\ b_2] = [a_0 b_0\ \ a_1 b_0 + a_0 b_1\ \ a_0 b_2 + a_1 b_1 + a_2 b_0]$

Conversione reale $a \to [a \; 0 \; 0]$

---

(Esempio Matlab).
Sappiamo aggiungere gestire di altre operazioni: inversa:

$$\frac{1}{b+\varepsilon} = \frac{1}{b} - \frac{\varepsilon}{b^2} + \frac{\varepsilon^2}{b^3} + O(\varepsilon^3)$$

esponenziale

$\exp(a)$: $\exp(a+\varepsilon) = \exp(a) + \varepsilon \cdot \exp(a) + \varepsilon^2 \exp(a)\frac{1}{2} + O(\varepsilon^3)$

$= \exp(a)\exp(\varepsilon) = \exp(a)\left(1 + \varepsilon + \frac{\varepsilon^2}{2} + \dots\right).$

$c = \phi(a, b)$

$\dot{c} = \frac{\partial \phi}{\partial a} \cdot \dot{a} + \frac{\partial \phi}{\partial b} \dot{b}$

$\ddot{c} = \dots$

---

Come si generalizza questo framework a funzioni $f: \mathbb{R}^n \to \mathbb{R}^m$ ?

$\mathbb{R}^m$ non dà problemi: costruisco per ogni variabile valore un elemento in $\mathbb{R}^m[\varepsilon]/(\varepsilon^3)$

$$z = \begin{bmatrix} \\ \\ \end{bmatrix} + \begin{bmatrix} \\ \\ \end{bmatrix}\varepsilon + \begin{bmatrix} \\ \\ \end{bmatrix}\varepsilon^2 + O(\varepsilon^3) \quad \text{3 vettori in } \mathbb{R}^m$$

$\mathbb{R}^n$ in partenza mi richiede di calcolare n derivate diverse rispetto a n direzioni diverse

$y = f(x) \quad x \in \mathbb{R}^n \quad y \in \mathbb{R}^m$

Per calcolare lo Jacobiano, calcolo derivate rispetto a ogni singola componente con

$$y = f\left(x + \varepsilon \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}\right), \quad y = f\left(x + \varepsilon \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}\right), \ldots \quad y = f\left(x + \varepsilon \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}\right)$$

Ogni iterazione mi dà una colonna dello Jacobiano.

Rete neurale: è una funzione parametrica $y = f_w(x)$

In machine learning si lavora con funzioni $f: w \to y$ che dipendono da moltissimi parametri $f: \mathbb{R}^n \to \mathbb{R}^m$

$n \gg m$.