Lezioni di Calcolo Numerico
Lezione 04: Approssimazione di dati e funzioni

Alberto Tibaldi

5 maggio 2018

Indice

2.4 Convergenza del polinomio di interpolazione

Ricordiamoci sempre che l'obiettivo di questa sezione del testo è cercare di ottenere, a partire da una funzione $y = f(x)$, un unico polinomio in grado di approssimarla in maniera accettabile su un intervallo $[a, b]$. Per fare questo, i coefficienti che definiscono questo polinomio devono essere ricavati imponendo il passaggio per un certo insieme di punti da noi scelti[1]. Nell'introduzione abbiamo però visto una situazione in cui, nonostante il polinomio intersecasse davvero la funzione in corrispondenza dei nodi, questo non assomigliava quasi in alcun modo alla funzione di partenza, in quanto presentava sovraelongazioni assolutamente innaturali rispetto al comportamento atteso. A questo punto il lettore, avendo visto questo e tenendo presente l'obiettivo finale, potrebbe chiedersi:

> «Abbiamo una qualche garanzia che non vengano fuori queste sovraelongazioni, e che quindi il nostro polinomio non sia solo interpolante, ma anche *rassomigliante* alla funzione di partenza?»

La risposta è no: per ciò che è stato scritto fino a questo momento, non sono state fornite garanzie di alcun tipo. Però, per poterlo dire, sarebbe opportuno cercare di dare un po' di formalità in più a queste affermazioni.

2.4.1 Errore di interpolazione

Fino a questo momento ci siamo solo concentrati sul cercare di soddisfare la condizione di interpolazione, sperando che come conseguenza naturale si avrebbe avuto un polinomio *rassomigliante* alla funzione di partenza. Per capire cosa sta succedendo, però, dobbiamo cercare di esprimere in *matematichese* questo concetto di *rassomigliare*, per poi poter quantificare davvero quanto due funzioni siano effettivamente simili. Partiamo dalla nostra $f(x)$ originale, e immaginiamo di approssimarla mediante un polinomio interpolante $p_n(x)$. Il fatto di approssimare qualcosa a qualcos'altro comporta **sbagliare**, commettere un **errore**; più piccolo è questo errore, più la funzione $f(x)$ e il polinomio $p_n(x)$ saranno somiglianti. Al fine di quantificare questo errore, definiamo la funzione **errore di interpolazione**

[1] ricordiamo sempre che questo problema è equivalente a richiedere il passaggio per coppie (x_i, y_i) che costituiscono un insieme di dati derivanti da un esperimento di un qualche genere

$$E_n(x) = f(x) - p_n(x). \tag{1}$$

Si può notare che questa funzione, nei nodi di interpolazione $\{x_i\}$, si annulli; infatti, dal momento che il nostro polinomio soddisfa le condizioni di interpolazione

$$p_n(x_i) = f(x_i),$$

allora certamente la funzione sarà zero[2]. Un altro dettaglio è: se la funzione $f(x)$ fosse un polinomio di grado minore o uguale a n, grado del polinomio interpolante, allora $E_n(x)$ sarebbe identicamente nulla. Infatti, **il polinomio interpolante di un polinomio è il polinomio stesso**: per il teorema di unicità esposto precedentemente, il polinomio è certamente unico.

2.4.2 Norma infinito e convergenza uniforme

Abbiamo definito l'errore di interpolazione (1) come una funzione ma, insomma, non è facilissimo *quantificare* questo errore. In effetti, questa funzione ci fornisce, per ogni punto della griglia *fine* in cui andiamo a valutare il polinomio di interpolazione, il valore dell'errore commesso. Immaginiamo per esempio di avere, per la stessa funzione, due polinomi di interpolazione di grado diverso, e di voler capire quale di questi sia *migliore* dell'altro. Non ci serve a nulla sapere punto per punto quale sia il migliore: ci servirebbe avere un unico indicatore che ci dia un'idea di come sia l'errore sull'intero intervallo!

In questo senso, dobbiamo cercare una maniera per ricavare, a partire dalla funzione $E_n(x)$, un singolo numero che *rappresenti* l'intera funzione. Ogni volta che vogliamo produrre un numero a partire da una funzione o da un vettore, si ricorre al concetto di **norma**[3]. Per questo caso, è utile definire la norma infinito di una funzione g su un intervallo $[a, b]$ come

$$\|g\|_\infty = \max_{x\in[a,b]} |g(x)|.$$

In altre parole, la norma infinito $\|.\|_\infty$ della funzione $g(x)$ è ottenuta cercando il massimo assoluto del modulo della stessa, nell'intervallo $[a, b]$. Volendo per esempio calcolare la norma infinito del vettore

$$\underline{b} = \begin{bmatrix} 1 & 3 & -10 & 9 & 9 \end{bmatrix},$$

questa sarà semplicemente il massimo dei valori assoluti delle componenti e, quindi, 10.

Cos'ha a che vedere tutta questa storia con il fatto che il polinomio interpolante $p_n(x)$ *assomigli* o meno alla nostra funzione? Beh, per capirlo, dovremmo semplicemente applicare questa idea della norma infinito alla nostra funzione errore $E_n(x)$. In effetti, se calcoliamo

$$\|E_n\|_\infty = \max_{x\in[a,b]} |f(x) - p_n(x)|,$$

stiamo valutando il **massimo errore** che commettiamo quando effettuiamo l'interpolazione. Ovviamente, se il **massimo** degli errori sull'intervallo è piccolo, a maggior ragione tutti gli altri saranno ancora più piccoli!!! Per questo motivo, più piccola sarà la norma infinito dell'errore, più simili saranno le funzioni!

Chiarito questo, propongo una domanda: se calcolassimo

$$\lim_{n\to\infty} \|E_n\|_\infty = \lim_{n\to\infty} \max_{x\in[a,b]} |f(x) - p_n(x)|,$$

questo tenderebbe a zero?

Prima di tutto, cosa significa quel limite? Beh, a ogni valore di n corrisponde un diverso polinomio interpolante $p_n(x)$ avente grado n. Per ottenerlo, però, è necessario utilizzare un diverso numero di nodi di interpolazione, dal momento che $p_n(x)$ è ottenuto interpolando esattamente $n + 1$ nodi. Quindi, l'idea di calcolare questo limite, è: faccio crescere i nodi di interpolazione, in

[2] in effetti, è possibile che in letteratura le procedure di interpolazione vengano anche chiamate *point matching*, ossia, soddisfare esattamente una condizione in un insieme di punti.

[3] il concetto di norma verrà introdotto con più calma e approfondito nelle prossime lezioni; per ora, si immagini semplicemente che esso costituisce una sorta di parametro che quantifica la *lunghezza* o l'*ampiezza* di un vettore/funzione

modo da *campionare* la curva in punti sempre più vicini tra loro, che tendono dunque ad avvicinarsi. Se faccio questo, il polinomio interpolante tende ad assomigliare di più alla curva di partenza, oppure no? E in particolare, se immaginiamo che $n \to \infty$, cioè, se immaginiamo ipoteticamente di poter far crescere fino all'infinito i nodi di interpolazione e il grado del polinomio interpolante, le due funzioni si assomiglieranno sempre di più, o no? **No. La risposta è NO.** Volendone una dimostrazione, l'Appendice (A.1) riporta un esempio di codice MATLAB®che, se eseguito con i parametri proposti, permette di vedere chiaramente che al crescere del grado del polinomio di interpolazione, l'errore non solo non tende a zero, ma tende a esplodere! Assurdamente, la funzione studiata in questo codice è $C^{(\infty)}$ e, di conseguenza, estremamente liscia; ciononostante, non si riesce a interpolare!

A rincarare ulteriormente la dose è il seguente teorema: data una **qualunque successione di nodi** distinti, tutti situati in $[a,b]$, esiste sempre una **funzione continua** $f(x)$ in $[a,b]$ che, interpolata su quei nodi, genera una successione di polinomi di interpolazione $\{p_n(x)\}$ **non uniformemente convergente** a $f(x)$ in $[a,b]$.

Di solito si parla di *teoremi costruttivi*, ma questo se vogliamo è **distruttivo**. Questo teorema garantisce che, se prendo un generico insieme di nodi, ci sarà sempre una qualche funzione continua tale per cui i polinomi interpolanti non le potranno mai assomigliare! Cioè, **non abbiamo garanzie!** E non stiamo parlando di funzioni assurde contenenti discontinuità, singolarità, salti: parliamo di funzioni **continue!** Notate inoltre che questo è proprio uno di quei problemi che a me vien da definire *filosofici*: questo non c'entra niente con la procedura con la quale stiamo valutando i coefficienti dei polinomi di interpolazione (che sia basata su `polyfit` o sui polinomi di Lagrange), quindi con ipotetici problemi di condizionamento: **questo è un limite dell'operazione di interpolazione!** È un limite intrinseco del tentare di interpolare, non c'entra nulla con i problemi di aritmetica, cancellazione o condizionamento: **è un problema filosofico!** Ma allora...

> «Ma allora che senso ha studiare tutte queste cose, se poi non abbiamo la minima garanzia che funzionino?!?!?!???!?!»

In effetti, non solo non abbiamo la garanzia che funzionino, ma il teorema ci dice che abbiamo la garanzia che **non funzionino.** Tuttavia, se potessimo in qualche modo garantire delle ipotesi un po' più forti di quelle del teorema, potremmo pensare di ottenere di più. In questo senso, ci conviene agire su due fronti.

- Dobbiamo garantire qualcosina in più sulla regolarità della funzione di partenza. In particolare, se questa funzione fosse solo continua, ma con derivate discontinue, non potremmo fare nulla. Dobbiamo quindi chiedere che f sia almeno derivabile una volta.
- Non possiamo scegliere una **qualunque** successione di nodi: ci conviene scegliere delle **particolari** successioni di nodi. Abbiamo provato sulla nostra pelle che usare nodi equispaziati non sia una scelta particolarmente da pane e volpe.

Partiamo dal secondo punto. Una scelta decisamente più sagace è quella di utilizzare i nodi di Chebyshev-Lobatto

$$z_i = -\cos\left(\frac{(i-1)\pi}{n}\right) \in [-1,1], \qquad i = 1,\dots,n+1, \tag{2}$$

o di Chebyshev

$$z_i = -\cos\left(\frac{(2i-1)\pi}{2(n+1)}\right) \in [-1,1], \qquad i = 1,\dots,n+1. \tag{3}$$

Come sempre, la n utilizzata in queste espressioni è il grado del polinomio interpolante la funzione in $n+1$ nodi. Tuttavia, invece di essere equispaziati, saranno distribuiti in maniera non uniforme.

Detto questo, le noie non sono finite: se andiamo a valutare (2) o (3), vediamo che i nodi sono definiti nell'intervallo $z \in [-1,1]$: questi sono il risultato di un coseno con un certo argomento! Questo non va bene, perché, in generale, una funzione va studiata su un intervallo $x \in [a,b]$! Per poter usare questi nodi sul nostro intervallo, è necessario *mapparli* su di esso: ci serve definire una funzione $x = m(z)$ che trasformi i nodi nel formato $\{z_i\}$ al formato $\{x_i\}$. Inoltre, abbiamo

capito che la spaziatura relativa tra i vari nodi è fondamentale: equispaziati non van bene, ma con questa spaziatura *di Chebyshev* sì. Quindi, la trasformazione per passare da z a x deve essere tale da non perturbare la spaziatura relativa e quindi, l'unica possibilità, è **la retta**: la curva che mantiene uguali i rapporti, e quindi le spaziature relative.

Come mi insegnava la prof. della terza liceo, la retta passante per due punti che permette di trasformare l'intervallo $[z_1, z_2] = [-1, 1]$ nell'intervallo $[x_1, x_2] = [a, b]$ soddisfa l'equazione

$$\frac{x - x_1}{x_2 - x_1} = \frac{z - z_1}{z_2 - z_1} \Longrightarrow \frac{x - a}{b - a} = \frac{z - (-1)}{1 - (-1)} \Longrightarrow x - a = \frac{b - a}{2}(z + 1),$$

che ci permette di scrivere

$$x = \frac{b - a}{2} z + \frac{b + a}{2}. \tag{4}$$

Al fine di mostrare un esempio di programmazione, l'Appendice A.2 mostra l'implementazione dello stesso esempio di Appendice A.1, usando però nodi di Chebyshev. A questo punto, possiamo enunciare un altro teorema.

Sia $\{p_n(x)\}$ la successione dei polinomi interpolanti $f(x)$ nei nodi di Chebyshev-Lobatto oppure di Chebyshev. Se $f \in C^{(k)}([a, b])$, $k \geq 1$, allora

$$\|f - p_n\|_\infty = \mathcal{O}\left(\frac{\log n}{n^k}\right), \quad n \to \infty.$$

Cosa significa tutto questo? Se si usano i nodi di Chebyshev o di Chebyshev-Lobatto, l'errore in norma infinito tende a 0 e, più la funzione è regolare, ovvero più derivate continue essa ha, e più questo rapidamente errore decresce, al crescere del grado del polinomio interpolante. Quindi, l'uso di questi particolari nodi di interpolazione ha ripristinato l'intuizione per cui più una funzione è *liscia*, regolare, e più facile deve essere interpolarla. Un appunto è però doveroso: questo teorema non è in alcun modo in conflitto con il teorema *distruttivo* enunciato precedentemente. Infatti, le garanzie di convergenza che esso ci fornisce sono contenute nel dettaglio $k \geq 1$, ovvero, nella richiesta di avere a che fare con funzioni almeno derivabili. Se infatti la funzione fosse solamente continua, avremmo $k = 0$ e, quindi, un tasso[4] $\mathcal{O}(\log n)$ che, quindi, all'infinito diverge, esplode: tende a ∞.

A Convergenza uniforme del polinomio di interpolazione: esempi MATLAB

A.1 Esempio 1: nodi equispaziati

Viene ora riportato uno script MATLAB® che permette di generare i polinomi di interpolazione di vari gradi utilizzando nodi equispaziati. Questo permette chiaramente di dimostrare che, agendo in maniera ingenua, non si hanno garanzie sulla convergenza uniforme del polinomio interpolante. L'esempio studia sull'intervallo $[-5, 5]$ la funzione di Runge, che ha ha infinite derivate continue.

```
clear
close all
clc

nvet = [1 2 3 4 5 6 7 8 9 10 11 12]; % specificare i gradi dei polinomi di interpolazione che si ↩
    vogliono provare
a = -5; % estremo inferiore intervallo
b = +5; % estremo superiore intervallo
f = @(x)1./(1+x.^2); % definisco la funzione di partenza
z = linspace(a,b,10001); % definisco la griglia fine su cui valutare i polinomi di interpolazione

figure(1)
set(gcf,'Position',[381 175 1208 753])

for indn = 1:length(nvet)
```

[4] la notazione *ogrande* $\mathcal{O}$ indica il comportamento asintotico, quindi per argomento tendente a infinito, dell'oggetto di cui ci si sta occupando

```
    n = nvet(indn);
    x = linspace(a,b,n+1); % in questo esercizio definisco i nodi di interpolazione tra loro ↩
        equispaziati
    c = polyfit(x,f(x),n); % valuto i coefficienti del polinomio interpolante
    p = polyval(c,z); % valuto il polinomio interpolante
    %
    figure(1), clf
    hold on
    grid on
    box on
    plot(z,f(z),'b',x,f(x),'ko',z,p,'r') % blu, curva originale; cerchietti neri, nodi di ↩
        interpolazione; rosso, polinomio
    xlabel('x');
    ylabel('f(x) e p_n(x)')
    title(['Funzione di partenza (blu) e polinomio p_{',num2str(n),'}(x)'])
    errrel = norm(f(z)-p)/norm(f(z));
    disp(['n = ',num2str(n),', errore relativo norma infinito ',num2str(errrel)])
    pause

end
```

A.2 Esempio 2: nodi di Chebyshev

Viene ora riportato uno script MATLAB®che permette di generare i polinomi di interpolazione di vari gradi, dove però vengono usati nodi di Chebyshev anziché nodi equispaziati. L'esempio studia di nuovo la funzione di Runge sull'intervallo $[-5,5]$.

```
clear
close all
clc

nvet = [1 2 3 4 5 6 7 8 9 10 11 12]; % specificare i gradi dei polinomi di interpolazione che si ↩
    vogliono provare
a = -5; % estremo inferiore intervallo
b = +5; % estremo superiore intervallo
f = @(x)1./(1+x.^2); % definisco la funzione di partenza
z = linspace(a,b,10001); % definisco la griglia fine su cui valutare i polinomi di interpolazione

figure(1)
set(gcf,'Position',[381 175 1208 753])

for indn = 1:length(nvet)
    n = nvet(indn);
    i = 1:(n+1); % indici dei nodi di Chebyshev
    zi = -cos(((2*i-1)*pi)/(2*(n+1))); % nodi Cheb. su intervallo [-1,1]
    x = (b-a)/2*zi+(b+a)/2; % mappare nodi Cheb. su intervallo [a,b]
    c = polyfit(x,f(x),n); % valuto i coefficienti del polinomio interpolante
    p = polyval(c,z); % valuto il polinomio interpolante
    %
    figure(1), clf
    hold on
    grid on
    box on
    plot(z,f(z),'b',x,f(x),'ko',z,p,'r') % blu, curva originale; cerchietti neri, nodi di ↩
        interpolazione; rosso, polinomio
    xlabel('x');
    ylabel('f(x) e p_n(x)')
    title(['Funzione di partenza (blu) e polinomio p_{',num2str(n),'}(x)'])
    errrel = norm(f(z)-p)/norm(f(z));
    disp(['n = ',num2str(n),', errore relativo norma infinito ',num2str(errrel)])
    pause

end
```