Progetto di fine corso

Christian Mancini

20 settembre 2023

Sommario

L'obbiettivo del progetto è quello di riprodurre a scopo didattico tutte le procedure necessarie per approssimare dati in 2D con una curva nel piano tramite le B-Spline utilizzando i minimi quadrati. Verranno anche fornite le implementazioni delle B-Spline e delle HB-Spline in Python.

Tutti i codici sorgenti mostrati in questa relazione sono reperibili alla seguente repository GitHub al seguente link: `https://github.com/cMancio00/B-Spline`

1 Introduzione

L'approssimazione è il processo di costruzione di una curva che coincida il più possibile con dei dati soggetti ad errore casuale. Queste tecniche vengo usate come alternativa all'interpolazione, dove si vuole un'esatta corrispondenza con alcuni punti dati. Ci sono delle situazioni in cui è conveniente usare tecniche di approssimazione, ad esempio:

1. visualizzazione dei dati
2. rappresentazione di una funzione dove non sono disponibili dati
3. sintetizzare relazioni tra variabili

Ci sono vari campi che utilizzano tecniche di approssimazione ad esempio la *modellazione statistica*, *machine learning* e *statistical learning*, ma con obbiettivi diversi. Il nostro obbiettivo è quello di fornire un'approssimazione il più possibile precisa di dati generati da funzioni generatrici, soggetti ad errori casuali. Non ci interessa quindi l'interpretazione dei risultati ottenuti.

Nel seguito useremo i seguenti dati generati casualmente:

2 Approssimazione

Spesso ci ritroviamo a dover risolvere un sistema di equazioni lineari **sovradeterminato**, ovvero con più equazioni che incognite, in cui la matrice dei coefficienti ha rango massimo. Ciò che vogliamo risolvere è quindi:

$$A\underline{x} = \underline{b} \qquad A \in \mathbb{R}^{m \times n} \qquad m \gg n \equiv rank(A) \tag{1}$$

Figura 1: $y = x + \varepsilon$

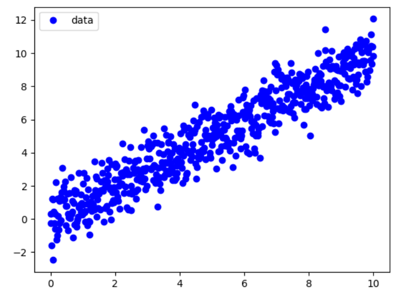
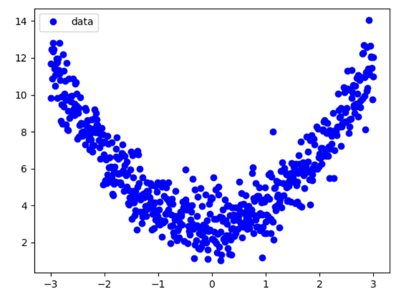


Figura 2: $3 + x^2 + \varepsilon$



Questo sistema lineare ammette soluzione se e solo se $\underline{b} \in range(A)$. Dato che $\underline{b} \in \mathbb{R}^m$, mentre $dim(range(A)) = rank(A) = n < m$, allora non ammette soluzione in senso classico. Possiamo però ricercare il vettore $\underline{x}$, in modo che minimizzi il sequente vettore detto *residuo*:

$$\underline{r} = \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix} = A\underline{x} - \underline{b} \tag{2}$$

Per fare ciò dobbiamo quindi ricercare $\underline{\mathrm{x}}$ che minimizzi la seguente quantità:

$$\sum_{i=1}^{m} = \|\underline{r}\|_2^2 = \|A\underline{x} - \underline{b}\|_2^2 \tag{3}$$

Questa è la soluzione ai **minimi quadrati**. Facendo ciò, il sistema lineare $A\underline{x} = \underline{b} + \underline{r}$, ammette soluzione.

Un modo efficiente per risolvere questo problema è fattorizzando la matrice A. Una fattorizzazione conveniente è la fattorizzazione QR.

Figura 3: $\sin x + \varepsilon$

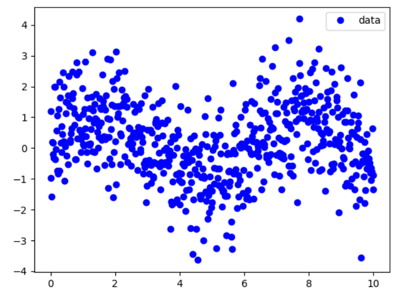
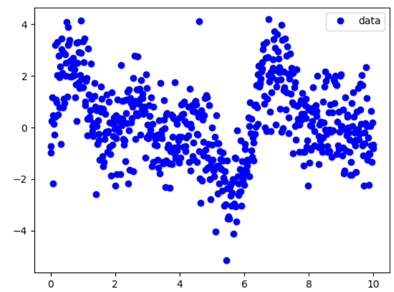


Figura 4: $\sin 2x + \sin 3x + \varepsilon$



Teorema 1 (Fattorizzazione QR)**.** *Data la matrice A, esistono le matrici:*

1. $Q \in \mathbb{R}^{m \times n}$, *ortogonale,*
2. $\hat{R} \in \mathbb{R}^{n \times n}$, *triangolare superiore*

Tali che

$$A = QR = Q \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix} \tag{4}$$

Osservazione.

$$Q^T A = R = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix} \begin{matrix} (n) \\ (m-n) \end{matrix} \tag{5}$$

Lemma 2.

$$Q^T \underline{b} = \begin{pmatrix} \underline{c} \\ \underline{d} \end{pmatrix} \begin{matrix} (n) \\ (m-n) \end{matrix} \tag{6}$$

Figura 5: $\frac{1}{1+25x^2} + \varepsilon$ ($\varepsilon \sim N(0, 0.1)$)

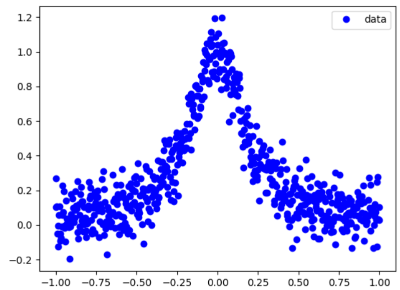
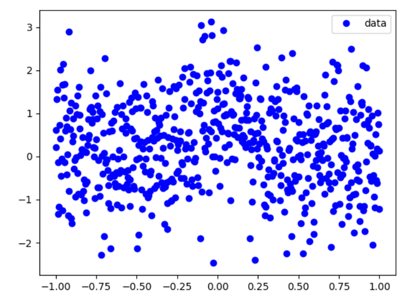


Figura 6: $\frac{1}{1+25x^2} + \varepsilon$ ($\varepsilon \sim N(0, 1)$)



Utilizzando questa fattorizzazione possiamo ridurre il problema:

$$\begin{aligned} ||A\underline{x} - \underline{b}|| &= ||Q^T A\underline{x} - Q^T \underline{b}|| \quad \text{(la norma 2 non viene modificata da una matrice ortogonale)} \\ &= ||\hat{R}\underline{x} - \underline{c}|| + ||\underline{d}|| \quad \text{(per l'osservazione 2 e per il lemma 2)} \end{aligned}$$

Ci siamo dunque ricondotti a dover risolvere il il seguente sistema lineare:

$$\hat{R}\underline{x} = \underline{c} \tag{7}$$

Tale sistema ha soluzione in tempo lineare, essendo $\hat{R}$ una matrice triangolare superiore. La fattorizzazione QR, se si utilizza il metodo di *Householder*, richiede $\approx \frac{2}{3}n^2(3m - n)$ flops. La funzione qr della libreria **numpy**, implementa la fattorizzazione con il metodo di *Householder*.

Per quanto riguarda il nostro problema, possiamo utilizzare una forma della matrice A più conveninte. Scelta una base qualsiasi è possibile costruirsi la matrice A che assume nomi diversi in base ai campi. Ad esempio può essere chiamata matrice *dei coefficienti*, *di costruzione*, *di design*. Dato che utilizzeremo le basi delle B-spline, essa prende il nome di **matrice di collocazione**.

Definizione 2.1 (matrice di collocazione)**.** *La matrice di collocazione A è definita nel seguente modo:*

$$A \equiv \begin{pmatrix} N_{0,k}(x_0) & \cdots & N_{n,k}(x_0) \\ \vdots & \ddots & \vdots \\ N_{0,k}(x_m) & \cdots & N_{n,k}(x_m) \end{pmatrix} \tag{8}$$

Dove

- $N_{i,k}$ *è la i-esima B-spline di ordine k*
- $x_0 \cdots x_m$ *sono le ascisse di valutazione*
- $x_0 = t_{k-1}$, $x_m = t_{n+1}$ *e* $\underline{t}$ *è il vettore esteso dei nodi*
- $n + 1 = dim(\mathbb{S}_{m,\tau})$

I dati sono contenuti nel vettore $\underline{b}$ di dimensione $m \times 2$ (nel caso bidimensionale). Il vettore delle incognite $\underline{x}$ equivale ai punti di controllo di de Boor, una volta trovati dobbiamo costruire una curva B-spline seguendo la definizione:

Definizione 2.2 (Curva B-Spline)**.**

$$\underline{X}(t) = \sum_{i=0}^{n} \underline{x}_i N_{i,k}(t) \tag{9}$$

con $t \in [t_{k-1}, t_{n+1}]$

3 Esempi

Iniziamo a vedere qualche esempio di approssimazione di funzione su dati generati casualmente. Gli esempi sono presi dal Notebook Jupyter della repository. Mostriamo inizialmente l'esempio più semplice, ovvero l'approssimazione della retta in figura 1. Cominciamo importando le librerie necessarie e creando una base B-Spline e una base gerarchica.

Listing 1: Dichiarazione della base

```
from Curve_Fitting import Model
from HB_Spline import HB_Spline
from B_Spline import B_Spline
import numpy as np
from matplotlib import pyplot as plt

np.random.seed(1304)

base  = B_Spline(
```

```
        knots=np.linspace(0,10,5+1),
        order=3
    )

hb = HB_Spline(base)
```

Nell'esempio è stata creata una base di ordine 3 con nodi uniformi. Il dominio dei nodi può essere arbitrario. In tutti gli esempi verrà fatto coincidere con il dominio dei dati solo per comodità nella rifinitura. Passiamo ora alla generazione dei dati. Per scopi di riproducibilità è stato impostato un seme. Per evitare problemi di dimensionalità il numero di dati equivale al numero di elementi nel vettore delle **ascisse di valutazione**. Un altro metodo potrebbe essere quello di scegliere in maniera equiprobabile dei dati da quelli generati.

Listing 2: Creazione e fit del modello

```
samples = np.shape(
    base.compute_base().get_collocation_matrix()
)[1]

x = np.linspace(0, 10, samples)
y= x + np.random.normal(0, 1, samples)

data = np.matrix([x, y]).T
```

A questo punto possiamo creare il modello passando la base gerarchica(che non essendo rifinita equivale alla B-Spline madre) e i dati che abbiamo generato.

Listing 3: Plot dei risultati

```
model = Model(
    base=hb,
    data=data
).fit()

model.plot()
plt.plot(x, x, "y--", label="real")
plt.legend(loc="best")
```

Otteniamo l'output come mostrato in figura 7

Elenco delle figure

Figura 7: Approssimazione di una retta