# Progetto di Calcolo Scientifico: Spectral Clustering

Valeria Tateo

Ottobre 2024

L'obiettivo del progetto è suddividere un insieme di dati in $k$ gruppi, detti cluster. Definita una metrica, i membri di uno stesso cluster sono "vicini", mentre vettori in cluster differenti sono ben separati.

## 1 Introduzione

Il problema del clustering può essere studiato introducendo il concetto di matrice Laplaciana associata ad un grafo. Dato un insieme di dati, si definisce una funzione

$$s(x_i, x_j) \geq 0$$

simmetrica e non negativa, che misura quanto due dati sono simili. Consideriamo il grafo pesato non orientato $G(V, E)$ e i pesi $w_{ij}$ per gli archi $(x_i,x_j)$ che siano funzione di $s(x_i, x_j)$. La matrice Laplaciana si definisce come

$$L = D - W$$

dove $W =(w_{ij})_{ij}$ è la matrice di adiacenza pesata e $D$ è la matrice diagonale che ha come entrate le somme sulle righe di $W$.

## 2 Grafi di Somiglianza

In base alla funzione $s$ scelta, si possono determinare Grafi di Somiglianza differenti. Di seguito si riportano gli algoritmi utilizzati per calcolare la matrice Laplaciana associata a tre tipi diversi di Grafi di Somiglianza.

**Algorithm 1** Si connettono tutti i punti la cui distanza è minore di $\epsilon$

**procedure** $\epsilon$_NEIGHBORHOOD($v$, $\epsilon$)
    $n \leftarrow$ length($v$)
    $W \leftarrow$ zeros $(n, n)$
    $D \leftarrow$ zeros $(n, n)$

    **for** $i = 1$ **to** $n$ **do**
        **for** $j = 1$ **to** $n$ **do**
            **if** $(i \neq j)$ **and** $\|v(i) - v(j)\|_2 < \epsilon$ **then**
                $W(i, j) \leftarrow \|v(i) - v(j)\|_2$ ▷ In alternativa: $W(i, j) \leftarrow \exp\left(-\frac{\|v(i)-v(j)\|_2^2}{2}\right)$
                $D(i, i) \leftarrow D(i, i) + W(i, j)$
            **end if**
        **end for**
    **end for**

    $L \leftarrow D - W$

**return** (L,D,W)
**end procedure**

**Algorithm 2** Si connette ogni vertice con i k-vertici più vicini

**procedure** K_NEAREST_NEIGHBOR_GRAPH($v, k$)
  $n \leftarrow \text{length}(v)$
  $W \leftarrow \text{zeros}\ (n, n)$
  $D \leftarrow \text{zeros}\ (n, n)$

  **for** $i = 1$ **to** $n$ **do**

    $dists \leftarrow \exp\left(-\frac{(v-v(i))^2}{2}\right)$ ▷ In alternativa: $dists \leftarrow \|v(i) - v(j)\|_2$
    $dists(i) \leftarrow -\infty$ ▷ Esclude l'elemento stesso
    $[\sim, sorted\_indices] \leftarrow \text{sort}(dists, \text{'descend'})$
    $nearest\_indices \leftarrow sorted\_indices(1:k)$ ▷ Seleziona i k vicini più vicini
    $W(i, nearest\_indices) \leftarrow 1$ ▷ Aggiorna la matrice dei pesi

  **end for**

  **for** $i \leftarrow 1$ **to** $n$ **do**

    **for** $j \leftarrow 1$ **to** $n$ **do**

      **if** $i \neq j$ **then**

        $D(i,i) \leftarrow D(i,i) + W(i,j)$

      **end if**

    **end for**

  **end for**

  $L \leftarrow D - W$

**return** (L,D,W)
**end procedure**

**Algorithm 3** Si connettono tutti i punti e si pone il peso degli archi uguale alla somiglianza, dove la funzione di somiglianza è la densità di una gaussiana standard

**procedure** FULLY_CONNECTED_GRAPH($v, \sigma$)
  $n \leftarrow \text{length}(v)$
  $W \leftarrow \text{zeros}\ (n, n)$
  $D \leftarrow \text{zeros}\ (n, n)$

  **for** $i = 1$ **to** $n$ **do**
    **for** $j = 1$ **to** $n$ **do**
      **if** $(i \neq j)$ **then**
        $W(i,j) \leftarrow \exp\left(-\frac{\|v(i)-v(j)\|_2^2}{2}\right)$ ▷ In alternativa: $W(i,j) \leftarrow \|v(i) - v(j)\|_2$
        $D(i,i) \leftarrow D(i,i) + W(i,j)$
      **end if**
    **end for**
  **end for**

  $L \leftarrow D - W$

**return** (L,D,W)
**end procedure**

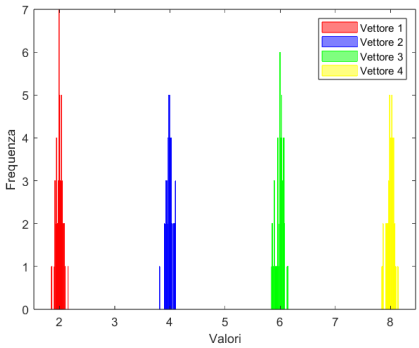
# 3 Inverse subspace iteration con shift negativo

Al fine di implementare una strategia di clustering, è necessario studiare un algoritmo che ci permetta di calcolare gli autovettori associati agli autovalori più piccoli di una matrice semidefinita positiva. Il seguente algoritmo ne è un esempio: sia $A$ una matrice, $num\_eigenvectors$ il numero di autovettori che si vuole calcolare, $num\_iterations$ il numero di iterazioni (ad esempio pari a 100) che l'algoritmo esegue e $shift$ lo shift negativo utilizzato (ad esempio pari a -1).

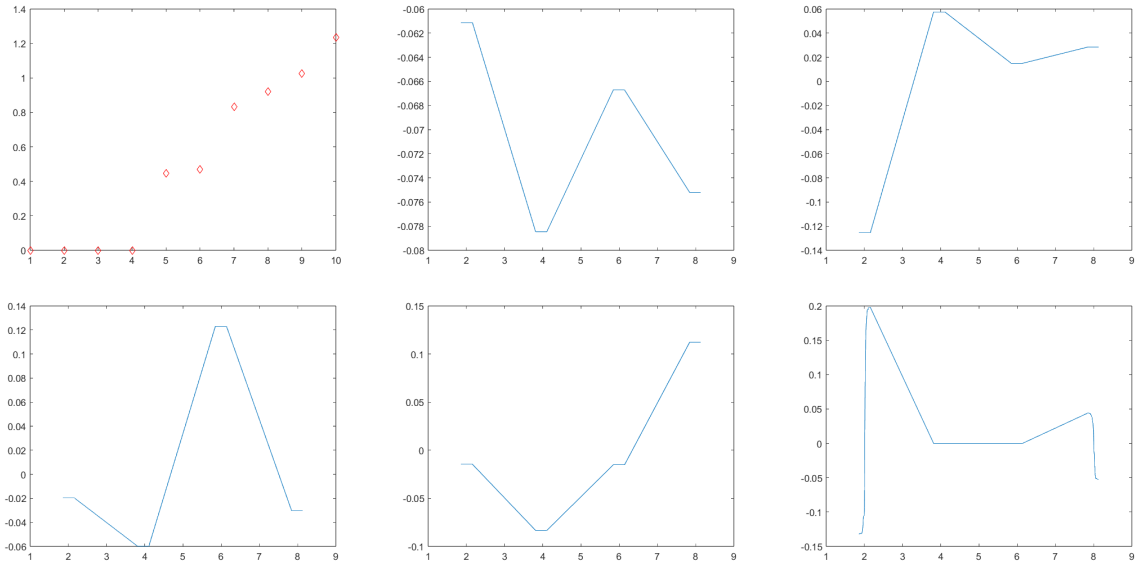
**Algorithm 4**

**procedure** INVERSE_SUBSPACE_ITERATION($A$, $num\_eigenvectors$, $num\_iterations$, $shift$)
  $n \leftarrow \text{size}(A, 1)$
  $X \leftarrow \text{rand}(n, num\_eigenvectors)$
  **for** $iter = 1$ **to** $num\_iterations$ **do**
    $X \leftarrow (A - shift \times \text{eye}(n)) \backslash X$
    $[X, \sim] \leftarrow \text{qr}(X, 0)$
  **end for**
  $eigenvalues \leftarrow \text{diag}(X^\intercal A X)$ ▷ Calcola gli autovalori approssimati
  $eigenvectors \leftarrow X$
  **return** ($eigenvectors$, $eigenvalues$)
**end procedure**

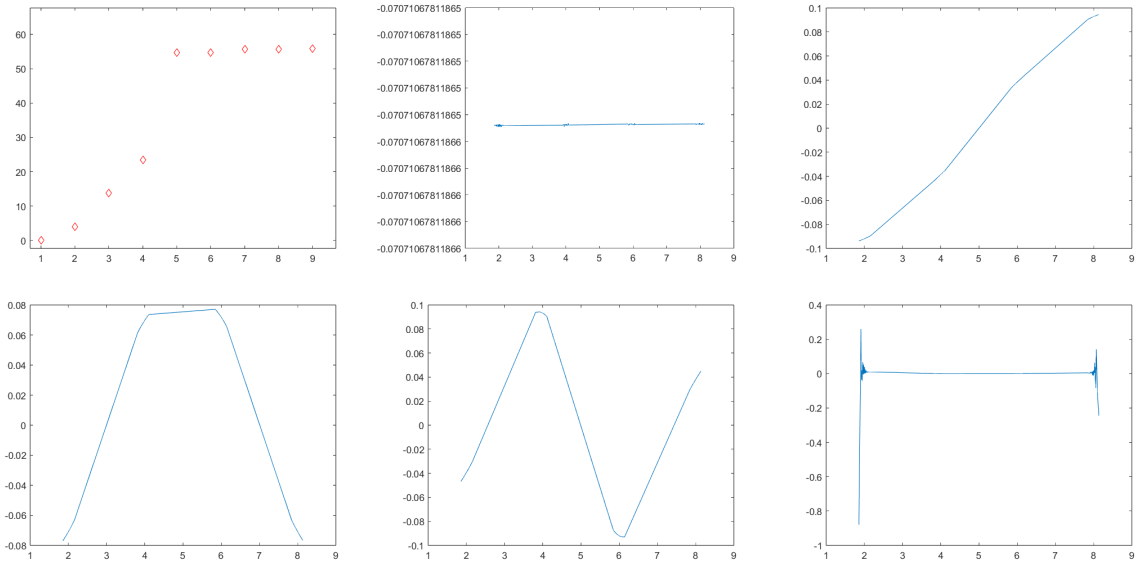
Per verificare la correttezza dell'algoritmo, lo testiamo sulla matrice Laplaciana calcolata utilizzando l'Algoritmo del 10-Nearest Neighbor Graph e su quella calcolata con l'Algoritmo del Fully Connected Graph, partendo da un insieme di dati costituito da 200 punti reali presi da 4 distribuzioni Gaussiane (50 per distribuzione) centrate in 2, 4, 6, 8 con varianza 16.
Di seguito è riportato l'istogramma relativo ai dati utilizzati e i primi 10 autovalori e 5 autovettori di L, ottenuti utilizzando gli algoritmi sopra citati.




Algoritmo del 10-Nearest Neighbor Graph:



Algoritmo del Fully Connected Graph:



Osserviamo che nel primo caso ci sono 4 autovalori nulli, dunque gli autovettori corrispondenti individuano 4 cluster, in accordo con il fatto che i dati sono stati presi da 4 gaussiane diverse. Nel secondo caso invece c'è un unico autovalore nullo: ciò deriva dal fatto che il grafo è completamente connesso.

# 4 Strategia di clustering

Utilizzando l'algoritmo kmeans di Matlab e le funzioni sopra riportate, possiamo eseguire l'algoritmo **Unnormalized Spectral Clustering**. L'obiettivo è quello di ottenere una rappresentazione dei dati tale che i punti di colore diversi appartengano a cluster diversi. Sia $x$ il vettore dei dati, $\epsilon$ la soglia per l'algoritmo $\epsilon$_neighborhood, $k$ il numero di vicini dell'algoritmo k_nearest_neighbor_graph, $c$ il numero di cluster cercati e $num\_eigenvectors$ il numero di autovettori della matrice Laplaciana.

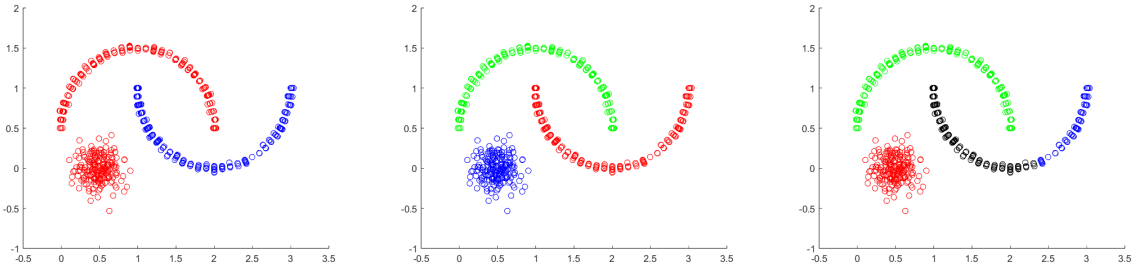
**Algorithm 5**

**function** SPECTRAL_CLUSTERING($x$, $\epsilon$, $k$, $c$, $num\_eigenvectors$)
  $[L, \sim, \sim] \leftarrow \epsilon$_neighborhood$(x, epsi)$
    ▷ In alternativa: k_nearest_neighbor_graph(x,k) o fully_connected_graph(x)
  $[U, \sim] \leftarrow$ inverse_subspace_iteration$(L, num_eigenvectors, 100, -1)$
  $y \leftarrow$ kmeans$(U(:, 1:c), c)$
  $clust \leftarrow$ cell$(1, c)$
  **for** $i = 1$ to length(x) **do**
    **for** $j = 1$ to $c$ **do**
      **if** $y(i) == j$ **then**
        $z \leftarrow i$
        $clust\,\{j\} \leftarrow [clust\,\{j\}, z]$
      **end if**
    **end for**
  **end for**
  $col \leftarrow$ {'ro','bo','go','ko'}
  **for** c **do**
    $a \leftarrow find(y == i)$
    hold on
    **for** $j = 1$ to size(a,1) **do**
      $plot(x(a(j)), col\,\{i\})$
    **end for**
  **end for**
  **return** $clust$
**end function**

Siamo pronti per testare la stretegia di clustering descritta su un file di dati in $\mathbb{R}^2$ al variare di alcuni parametri. Varieranno: l'algoritmo per la costruzione del grafo utilizzato; la funzione di somiglianza (distanza euclidea e densità gaussiana standard); numero di cluster cercati (2, 3, 4); numero di autovettori della matrice Laplaciana.

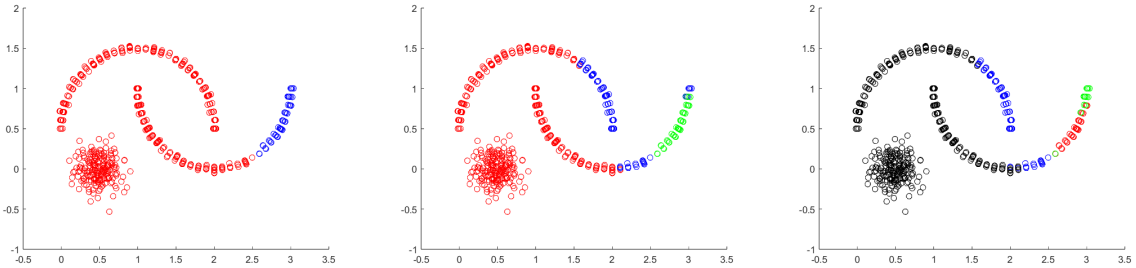
**Caso 1:** Algoritmo: $\epsilon$ -neighborhood ($\epsilon = 0.25$); Funzione di somiglianza: distanza euclidea; numero di autovettori della matrice Laplaciana pari al numero di cluster cercati.

. 2 cluster 3 cluster 4 cluster



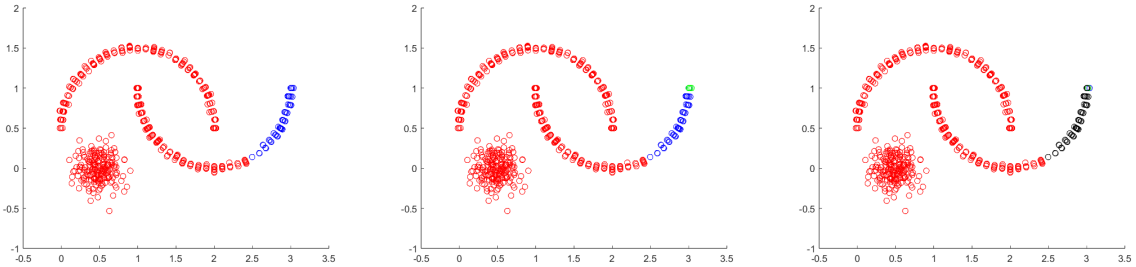
**Caso 2:** Algoritmo: k_nearest_neighbor_graph (k=12); Funzione di somiglianza: densità gaussiana; numero di autovettori della matrice Laplaciana pari al numero di cluster cercati.

. 2 cluster 3 cluster 4 cluster



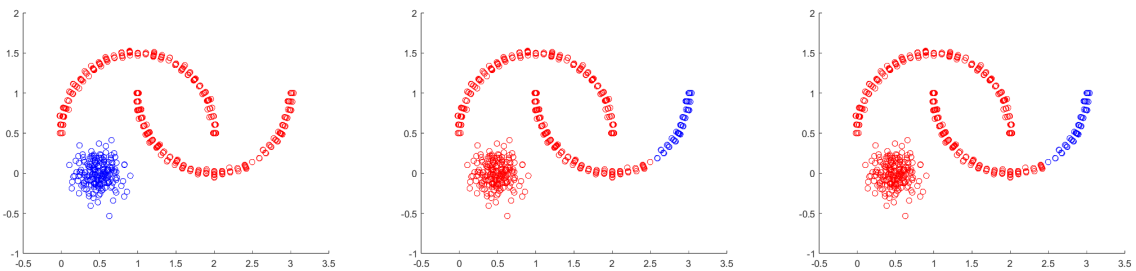
**Caso 3:** Algoritmo: fully_connected_graph; Funzione di somiglianza: densità gaussiana; Numero di autovettori della matrice Laplaciana pari al numero di cluster cercati.

. 2 cluster 3 cluster 4 cluster



**Caso 4:** Algoritmi: $\epsilon$ -neighborhood ($\epsilon = 0.25$), k_nearest_neighbor_graph (k=12) e fully_connected_graph; Funzione di somiglianza: densità gaussiana; Caso del vettore di Fiedler.

. $\epsilon$ -neighborhood k_nearest_neighbor_graph fully_connected_graph



# 5 Conclusioni

Dai risultati ottenuti è evidente che la strategia di clustering migliore è quella del caso 1.3. In generale, l'algoritmo $\epsilon$ -neighborhood funziona meglio rispetto agli altri. Notiamo infatti che:

- utilizzando l'algoritmo k_nearest_neighbor_graph, i dati vengono suddivisi in maniera casuale all'aumentare del numero di cluster;
- utilizzando l'algoritmo fully_connected_graph e aumentando il numero di autovettori, vengono individuati 2 cluster principali e i restanti cluster includono pochi dati.

Infine, nel caso 4 abbiamo analizzato il metodo del vettore di Fiedler che funziona bene per ogni grafo considerato.

# 6 Bibliografia

[1] Luxburg. *A Tutorial on Spectral Clustering*