Metodi di clustering spettrale

Course of Machine Learning
Master Degree in Computer Science
University of Rome ``Tor Vergata''

Giorgio Gambosi

a.a. 2017-2018

Problema del Clustering

Dati del problema

- Un insieme di punti x_1,.....,x_n
- Qualche relazione tra questi punti (similarità, vicinanza, distanze, connettività)

Obiettivo

Dividere l'insieme di punti x_1,.....,x_n in gruppi diversi tale che:

- i punti che si trovano nello stesso gruppo sono simili
- i punti che si trovano in gruppi diversi sono dissimili

Metodo di clustering partizionale: K-means (1/3)

Idea dell'algoritmo

Input: Un insieme di punti $x_1, ..., x_n \in \mathbb{R}^d$

1. Si inizia scegliendo casualmente i K punti da usare come centri dei cluster
2. Si assegna ogni punto, in maniera esclusiva, al cluster il cui centro gli è più vicino, secondo il criterio di distanza scelto(euclidea, Mahalanobis,..)
3. Si ricalcolano i centri dei cluster attraverso la media delle coordinate dei punti che appartengono ad essi
4. Si ripetono questi passi finché l'algoritmo non converge.
   Si ha convergenza quando il valore dei nuovi centri calcolati non cambiano dai valori dei precedenti e quindi le partizioni non cambiano tra due passi successivi

Metodo di clustering partizionale: K-means (2/3)

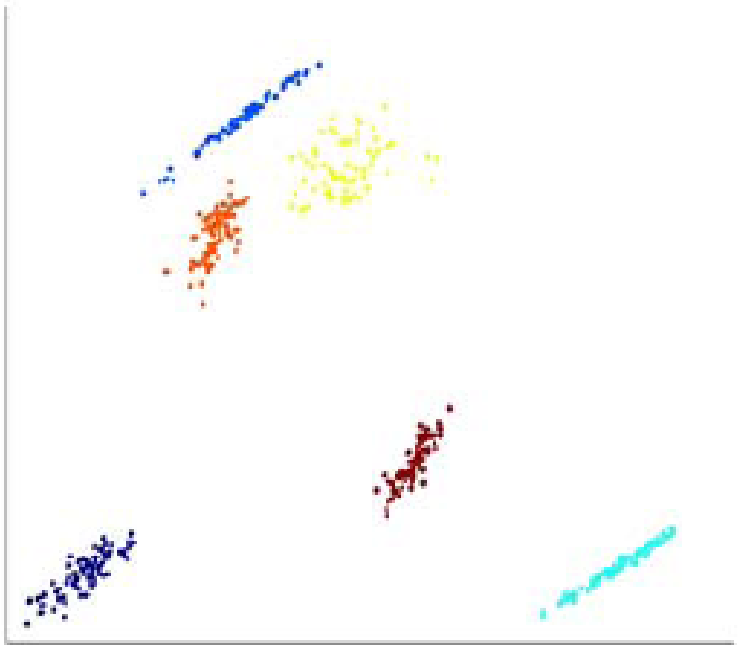
Vantaggi

- Semplice da implementare
- Converge piuttosto velocemente

Svantaggi

- E' sensibile alla scelta dei valori centrali di partenza.
  Scelte differenti possono portare a clustering diversi
- Si comporta bene con cluster che assumono forme globulari, sferiche
- Lavora poco con cluster che hanno forme arbitrarie (La soluzione è proprio lo Spectral Clustering)

Metodo di clustering partizionale: K-means (3/3)



- Nel caso di sinistra il K-means lavora
- Nel caso di destra il K-means non lavora

Spectral Clustering: Idea

1. Dati:
   - Un insieme di punti x_1,.....,x_n
   - Qualche nozione di similarità $s_{ij} \geq 0$ tra tutte le coppie di punti x_i e x_j
2. Costruire un **grafo di similarità** *G = (V, E)* dove:
   - Ogni vertice v_i del grafo rappresenta un punto dei dati x_i
   - Due vertici vengono connessi se la similarità s_{ij} tra i punti che rappresentano, x_i e x_j, è positiva o più grande rispetto ad una certa soglia
   - Gli archi vengono pesati con i valori di similarità s_{ij}
3. Definire una funzione obiettivo che permetta di **tagliare (cut)** il grafo pesato in gruppi (cluster) disgiunti in modo tale che:
   - Gli archi tra gruppi diversi abbiano un peso basso (punti in cluster diversi sono dissimili tra loro)
   - Gli archi interni ad un gruppo abbiano un peso alto (punti nello stesso cluster sono simili tra loro)

Definizioni e notazioni (1/5)

Sia *G = (V, E)* un **grafo pesato non orientato** dove:

- *V* = $\{v_1, ..., v_n\}$ è l'insieme dei vertici
- *E* = $\{e_{ij}\}$ è l'insieme degli archi
- Ogni arco tra due vertici v_i e v_j ha un peso non negativo $w_{ij} \geq 0$

Matrice di adiacenza pesata

La **matrice di adiacenza pesata** del grafo è la matrice
W = $(w_{ij})_{ij=1,...n}$ dove:

$$w_{ij} = \begin{cases} 0 & \text{se i vertici } v_i \text{ e } v_j \text{ non sono connessi da nessun arco,} \\ s_{ij} & \text{altrimenti.} \end{cases}$$

Nota: siccome il grafo *G* è non orientato allora $w_{ij} = w_{ji}$

Definizioni e notazioni (2/5)

Grado di un vertice

Il **grado di un vertice** $v_i \in V$ è definito come:

$$d_i = \sum_{j=1}^{n} w_{ij}$$

Grado di una matrice

Il **grado di una matrice** D è definito come la matrice diagonale con i gradi d_1,.....,d_n sulla diagonale:

$$D = diag(d_1,....,d_n)$$

Definizioni e notazioni (3/5)

Misurare la dimensione di un sottoinsieme di vertici

Ci sono due modi per misurare la dimensione di un sottoinsieme $A \subset V$:

$$| A | := \text{numero di vertici in } A$$

$$vol(A) := \sum_{i \in A} d_i$$

Nota: $| A |$ misura la dimensione di A in base al numero dei suoi vertici
$vol(A)$ misura la dimensione di A sommando il peso di tutti gli archi collegati ai vertici in A

Definizioni e notazioni (4/5)

Complemento di un sottoinsieme di vertici

Il **complemento**, $\bar{A}$, di un sottoinsieme di vertici $A \subset V$ si denota con $V \setminus A$

Vettore indicatore

Si definisce **vettore indicatore** il vettore $1_A = (f_1, ..., f_n)^T \in \mathbb{R}^n$ dove:

$$f_i = \begin{cases} 1 & \text{se } v_i \in A\ , \\ 0 & \text{altrimenti.} \end{cases}$$

Definizioni e notazioni (5/5)

Sottoinsieme connesso

Un sottoinsieme $A \subset V$ di un grafo si dice **connesso** se per ogni coppia di nodi esiste un percorso che li unisce

Componente connesso

Un sottoinsieme A si dice **componente connesso** se è connesso e se non contiene connessioni tra i vertici di A e $\bar{A}$

Partizione del grafo

Un sottoinsieme non vuoto $A_1, ...A_n$ forma una **partizione** del grafo se:

- $A_i \cap A_j = \emptyset$
- $A_1 \cup .... \cup A_k = V$

Definizione di diversi grafi di similarità

Ci sono diversi modi per trasformare un insieme di punti $x_1, ..., x_n$ con delle similarità tra i vertici, s_{ij}, in un grafo.
Quando si costruisce un grafo di similarità l'obiettivo è modellare le relazioni di vicinanza locale tra i punti.
Si propongono i seguenti grafi di similarità:

- **ϵ-neighborhood graph**
- **k-nearest neighbor graph**
- **fully connected graph**

Tutti i grafi proposti sono regolarmente usati nel clustering spettrale.
Inoltre, non esiste ancora qualche risultato teorico su come la scelta del grafo di similarità influenzi il risultato del clustering spettrale.

ϵ-neighborhood graph

Questo grafo si ottiene:

- connettendo le coppie di punti la cui distanza è più piccola di ϵ

Siccome le distanze, tra tutti i punti connessi, sono più o meno simili (al massimo ϵ), pesare gli archi non darebbe alcuna informazione aggiuntiva, riguardo i dati, al grafo.
Quindi, il **ϵ-neighborhood graph** è solitamente considerato un grafo non pesato.

k-nearest neighbor graph

In questo caso è possibile costruire il grafo diretto, quando le relazioni di vicinanza non sono simmetriche:

- connettendo il vertice v_i al vertice v_j se v_j è tra i primi k vicini di v_i

Oppure si può ottenere il grafo non diretto in due diversi modi:

1. Ignorare le direzioni degli archi, connettendo, quindi, v_i e v_j tramite un arco non orientato se v_i è tra i primi k vicini di v_j **o** se v_j è tra i primi k vicini di v_i.
   Il grafo risultante è il **symmetric k-nearest neighbor graph**.
2. Connettere i vertici v_i e v_j se v_i è tra i primi k vicini di v_j **e** se v_j è tra i primi k vicini di v_i.
   Il grafo risultante è il **mutual k-nearest neighbor graph**.

In entrambi i casi, dopo aver connesso tutti i vertici appropriati, si pesano gli archi in base alla similarità dei loro endpoint.

Fully connected graph

Questo grafo si ottiene:

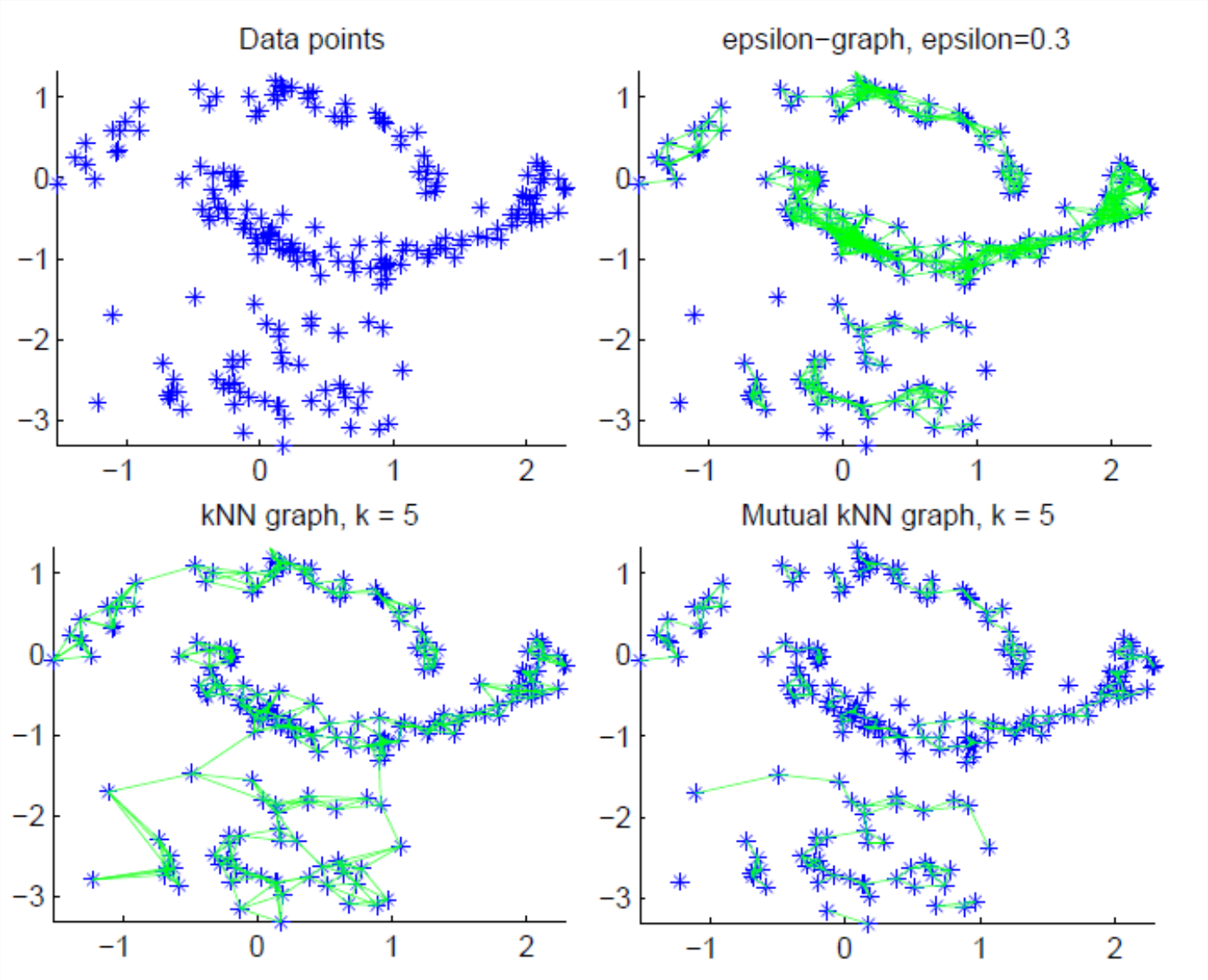
- connettendo tutti i punti con similarità positiva a tutti gli altri
- pesando tutti gli archi con il valore s_{ij}

Siccome tale grafo dovrebbe rappresentare le relazioni di vicinanza locale, questa costruzione è utile solo se la funzione di similarità stessa modella la vicinanza locale.
Questo grafo viene spesso utilizzato con funzione la funzione di similarità Gaussiana:

$$s(x_i, x_j) = exp(-\| x_i - x_j \|^2/(2\sigma^2))$$

dove il parametro σ controlla l'ampiezza del vicinato.

Esempi di diversi grafi di similarità



Taglio del grafo (Graph cut)

- A partire dal grafo di similarità, che rappresenta i dati iniziali e dalla matrice di adiacenza pesata W, l'idea dello Spectral Clustering è proprio quella di tagliare nel miglior modo possibile questo grafo
- Quindi lo Spectral Clustering può essere visto come un'approssimazione al problema del partizionamento di un grafo
- Il modo più semplice e diretto di costruire una partizione del grafo è quello di risolvere il **problema del taglio minimo (mincut problem)**

Problema del taglio minimo (mincut problem)

Definizione

Dato un grafo pesato, si definisce problema del taglio minimo quello di partizionare tale grafo in due insiemi disgiunti, A, B, tali che il peso degli archi che connettono i vertici di A a quelli di B sia minimo.
Più precisamente, risolvere il problema del taglio minimo significa minimizzare la quantità:

$$cut(A, B) := \sum_{i \in A, j \in B}^{n} w_{ij} \tag{1}$$

Attenzione: L'equazione (1) porta spesso a scarsi risultati, in quanto in molti casi viene tagliato via dal grafo un solo vertice. Per questo motivo si sono proposti dei tagli bilanciati.

Bilanciamento dei tagli

$min_{A,B} cut(A, B)$ bilanciato

$$cut(A, B) := \sum_{i \in A, j \in B}^{n} w_{ij} \qquad tale\ che\ |A| = |B| \tag{2}$$

Ratio cut

$$RatioCut(A, B) := cut(A, B) \left(\frac{1}{|A|} + \frac{1}{|B|} \right) \tag{3}$$

Normalized cut

$$Ncut(A, B) := cut(A, B) \left(\frac{1}{vol(A)} + \frac{1}{vol(B)} \right) \tag{4}$$

Spectral Clustering ed il mincut problem

- Purtroppo, aggiungendo le condizioni di bilanciamento il problema del minimo taglio diventa da semplice ed efficientemente risolvibile a NP hard.
- Lo Spectral Clustering è proprio un modo per risolvere versioni rilassate di questi problemi NP hard:
  - Rilassando RatioCut si ha uno Spectral Clustering non normalizzato
  - Rilassando NCut si ha uno Spectral Clustering normalizzato

Laplaciano non normalizzato di un grafo (1/2)

Il **Laplaciano non normalizzato di un grafo G**, non orientato, pesato e di n nodi, è rappresentato dalla matrice **L** nxn definita come segue:

$$L = D - W \tag{5}$$

- *D* è la matrice diagonale di ordine n che ha sulla diagonale principale il grado di ciascun vertice definita come $D = diag(d_1,....,d_n)$
- *W* è la matrice di adiacenza pesata del grafo di ordine n definita come $W = (w_{ij})_{ij=1,...n}$ dove:

$$w_{ij} = \begin{cases} 0 & \text{se i vertici } v_i \text{ e } v_j \text{ non sono connessi da nessun arco,} \\ s_{ij} & \text{altrimenti.} \end{cases}$$

  Si ricorda che siccome il grafo *G* è non orientato allora $w_{ij} = w_{ji}$ e che $w_{ij} \geq 0$

Laplaciano non normalizzato di un grafo (2/2)

Proprietà della matrice *L*:

1. Per ogni vettore $f \in \mathbb{R}^n$:

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (f_i - f_j)^2 \tag{6}$$

2. L è simmetrica e semi-definita positiva
3. Il più piccolo autovalore di L è 0, il corrispondente autovettore è il vettore $\mathbb{1}$ che ha tutti i suoi elementi pari ad 1, $\mathbb{1} = [1, ..., 1]^T$
4. L ha n autovalori non negativi, a valori reali $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$

Dimostrazione delle proprietà di L (1/2)

Parte (1): Per definizione di d_i,

$$f^T L f = f^T D f - f^T W f = \sum_{i=1}^{n} d_i (f_i)^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij}$$

$$= \frac{1}{2} \left(\sum_{i=1}^{n} d_i (f_i)^2 - 2 \sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j (f_j)^2 \right)$$

$$= \frac{1}{2} \left(\sum_{i=1}^{n} \left(\sum_{j=1}^{n} w_{ij} \right) (f_i)^2 - 2 \sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} \left(\sum_{i=1}^{n} w_{ij} \right) (f_j)^2 \right)$$

$$= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(w_{ij} (f_i)^2 - 2 f_i f_j w_{ij} + w_{ij} (f_j)^2 \right) = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (f_i - f_j)^2$$

□

Dimostrazione delle proprietà di *L* (2/2)

Parte (2): La simmetria di *L* segue direttamente dalla simmetria di *D* e *W*. La semi-positività è una conseguenza diretta della Parte (1) in cui si è dimostrato che $f^T L f \geq 0$ per ogni $f \in \mathbb{R}^n$.

Parte (3): Per costruzione la somma per righe della matrice laplaciana è nulla, quindi si ha che il primo autovalore per ogni grafo è 0, cioè $\lambda_1 = 0$.
$\lambda_1 = 0$ è ovviamente l'autovalore più piccolo, perché L è una matrice definita semi-positiva quindi i suoi autovalori sono tutti ≥ 0.
A questo punto per la relazione che lega autovalori ed autovettori: $L\mathbb{1} = 0$, allora $\mathbb{1} = [1, ..., 1]^T$, quindi l'autovettore destro di L relativo all'autovalore zero, avrà tutti i suoi elementi pari ad 1 .

Parte (4): Diretta conseguenza delle Parti da (1) a (3). □

Prima relazione tra spettro e cluster

Numero di componenti connesse e lo spettro di L

Sia G un grafo non diretto con pesi non negativi:

- La molteplicità k dell'autovalore 0 di L è uguale al numero di componenti connesse $A_1, ..., A_k$ nel grafo.
- L'autospazio dell'autovalore 0 è generato dai vettori indicatori $\mathbb{1}_{A_1}, ..., \mathbb{1}_{A_k}$ di quelle componenti.

Laplaciane normalizzate di un grafo (1/2)

Esistono due matrici, in letteratura, chiamate **Laplaciane normalizzate di un grafo** definite come:

$$\mathbf{L_{sym}} := \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}} \tag{7}$$

$$\mathbf{L_{rw}} := \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W} \tag{8}$$

- L_{sym} è una matrice simmetrica
- L_{rw} è una matrice normalizzata sommando per riga

Laplaciane normalizzate di un grafo (2/2)

Proprietà delle matrici L_{sym} e L_{rw} :

1. Per ogni vettore $f \in \mathbb{R}^n$:

$$f^T L_{sym} f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2 \tag{9}$$

2. λ è un autovalore di L_{rw} con autovettore u se e solo se λ è un autovalore di L_{sym} con autovettore $w = D^{\frac{1}{2}} u$
3. λ è un autovalore di L_{rw} con autovettore u se e solo se λ e u risolvono la relazione $Lu = \lambda Du$
4. 0 è un autovalore di L_{rw} con il vettore $\mathbb{1} = [1, ..., 1]^T$ come autovettore. 0 è un autovalore di L_{sym} con autovettore $D^{\frac{1}{2}} \mathbb{1}$
5. L_{sym} e L_{rw} sono semi-definite positive e hanno n autovalori non negativi, a valori reali $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$

Dimostrazione delle proprietà di L_{sym} e L_{rw}

Parte (1): Può essere provata come la Parte (1) delle proprietà della matrice L.

Parte (2): Segue immediatamente moltiplicando $D^{-\frac{1}{2}}$ all'equazione $L_{sym}w = \lambda w$ e sostituendo $u = D^{-\frac{1}{2}}w$.

Parte (3): Segue immediatamente moltiplicando D all'equazione $L_{rw}u = \lambda u$.

Parte (4): La prima affermazione è ovvia siccome $L_{rw}\mathbb{1} = 0$, la seconda segue da (2).

Parte (5): L'affermazione relativa a L_{sym} segue da (1), quindi quella relativa a L_{rw} segue da (2).

□

Seconda relazione tra spettro e cluster

Numero di componenti connesse e gli spettri di L_{sym} e L_{rw}

Sia G un grafo non diretto con pesi non negativi:

- La molteplicità k dell'autovalore 0 di L_{sym} e L_{rw} è uguale al numero di componenti connesse $A_1, ..., A_k$ nel grafo.
- Per L_{rw}, l'autospazio dell'autovalore 0 è generato dai vettori indicatori $\mathbb{1}_{A_i}$ di quelle componenti.
- Per L_{sym}, l'autospazio dell'autovalore 0 è generato dai vettori $D^{\frac{1}{2}}\mathbb{1}_{A_i}$ di quelle componenti.

Algoritmo di Spectral Clustering (1/4)

Assunzioni

- I dati iniziali sono rappresentati da punti $x_1, ..., x_n$ dove ogni $x_i \in \mathbb{R}^l$
- La similarità $s_{ij} = s(x_i, x_j)$, tra coppie di punti, è misurata tramite una funzione di similarità, simmetrica e non negativa
- Si denota con $S = (s_{ij})_{i,j=1,...,n}$ la matrice di similarità

Input

- La matrice di similarità $S \in \mathbb{R}^{nxn}$
- Numero k di cluster da ottenere

Passi dell'algoritmo (1/2)

- Costruire il grafo di similarità, scegliendone uno tra ϵ-neighborhood graph, k-nearest neighbor graph, Fully connected graph.
- Calcolare:

$$\begin{cases} L & \text{in caso di Spectral Clustering non normalizzato,} \\ L_{rw} & \text{in caso di Spectral Clustering normalizzato.} \end{cases}$$

- Calcolare i primi k autovettori $v_1, ..., v_k$ di:

$$\begin{cases} L & \text{in caso di Spectral Clustering non normalizzato,} \\ L_{rw} & \text{in caso di Spectral Clustering normalizzato.} \end{cases}$$

Algoritmo di Spectral Clustering (3/4)

Passi dell'algoritmo (2/2)

- Costruire la matrice $V \in \mathbb{R}^{nxk}$ che ha questi autovettori $v_1, ..., v_k$ come colonne
- Per $i = 1, ..., n$ sia $y_i \in \mathbb{R}^k$ il vettore corrispondente alla i-esima riga di V. E' possibile interpretare i vettori y_i come nuovi punti di dati
- Partizionare i punti $(y_i)_{i=1,...,n}$ in $\mathbb{R}^k$, tramite l'algoritmo K-means, nei cluster $C_1, ..., C_k$

Output: I cluster $C_1, ..., C_k$

Algoritmo di Spectral Clustering (4/4)

Osservazione importante

- L'algoritmo, attraverso le proprietà dei Laplaciani dei grafi, riesce a ridurre la dimensione dei punti iniziali.
- Più precisamente, gli n punti iniziali, $x_1, ..., x_n$ in $\mathbb{R}^l$, verranno rappresentati da un nuovo insieme di n punti $y_1, ..., y_n$ in $\mathbb{R}^k$ dove $k \ll l$
- Ovviamente è molto più facile cercare cluster in uno spazio ridotto
- In particolare, l'algoritmo di clustering, K-means non avrà difficoltà a riconoscere i cluster in questa nuova rappresentazione

Dettaglio pratico: Normalizzato o non normalizzato? (1/2)

Una delle domande fondamentali nello Spectral Clustering è quale definizione di Laplaciano utilizzare per calcolare gli autovettori.
Per rispondere a questa domanda bisogna capire la differenza pratica tra i due approcci:

Spectral Clustering normalizzato

- **Minimizza** il taglio
- **Massimizza** il volume di ogni partizione
- Statisticamente consistente

Spectral Clustering non normalizzato

- **Minimizza** il taglio
- **Massimizza** la cardinalità di ogni partizione
- **Non** statisticamente consistente

Dettaglio pratico: Normalizzato o non normalizzato? (2/2)

Cosa scegliere?

Sicuramente lo **Spectral Clustering normalizzato!**
Per due motivi:

1. Massimizzare il volume è più importante rispetto alla cardinalità, perché il volume indica la connettività interna di una partizione, mentre la cardinalità il numero di vertici nella stessa
2. Essere statisticamente consistente significa, che se la cardinalità dei punti iniziali tende a infinito, i risultati del Laplaciano convergono ad una partizione. Questo non vale per gli approcci non statisticamente consistenti

Dettaglio pratico: Come scegliere il numero di cluster?

- Scegliere il numero k di cluster è un problema per ogni algoritmo di clustering.
- Di solito nello Spectral Clustering si utilizza un'euristica speciale: **eigengap heuristic**
- Tale euristica sceglie il numero k se gli autovalori $\lambda_1, ..., \lambda_k$ sono molto piccoli rispetto all'autovalore λ_{k+1}
- Questa euristica viene scelta perché limita velocemente il valore di Ncut

Simulazioni

- Si presenteranno ora le immagini dei risultati di alcune simulazioni fatte su tre tipi di dati iniziali:
  1. Due partizioni sferiche
  2. Dati a mezzaluna
  3. Una scritta in corsivo
- Le simulazioni sono state eseguite tramite:
  1. L'algoritmo di Spectral Clustering, utilizzando come funzione di similarità quella Gaussiana
  2. L'algoritmo K-means

Simulazione: Due partizioni sferiche

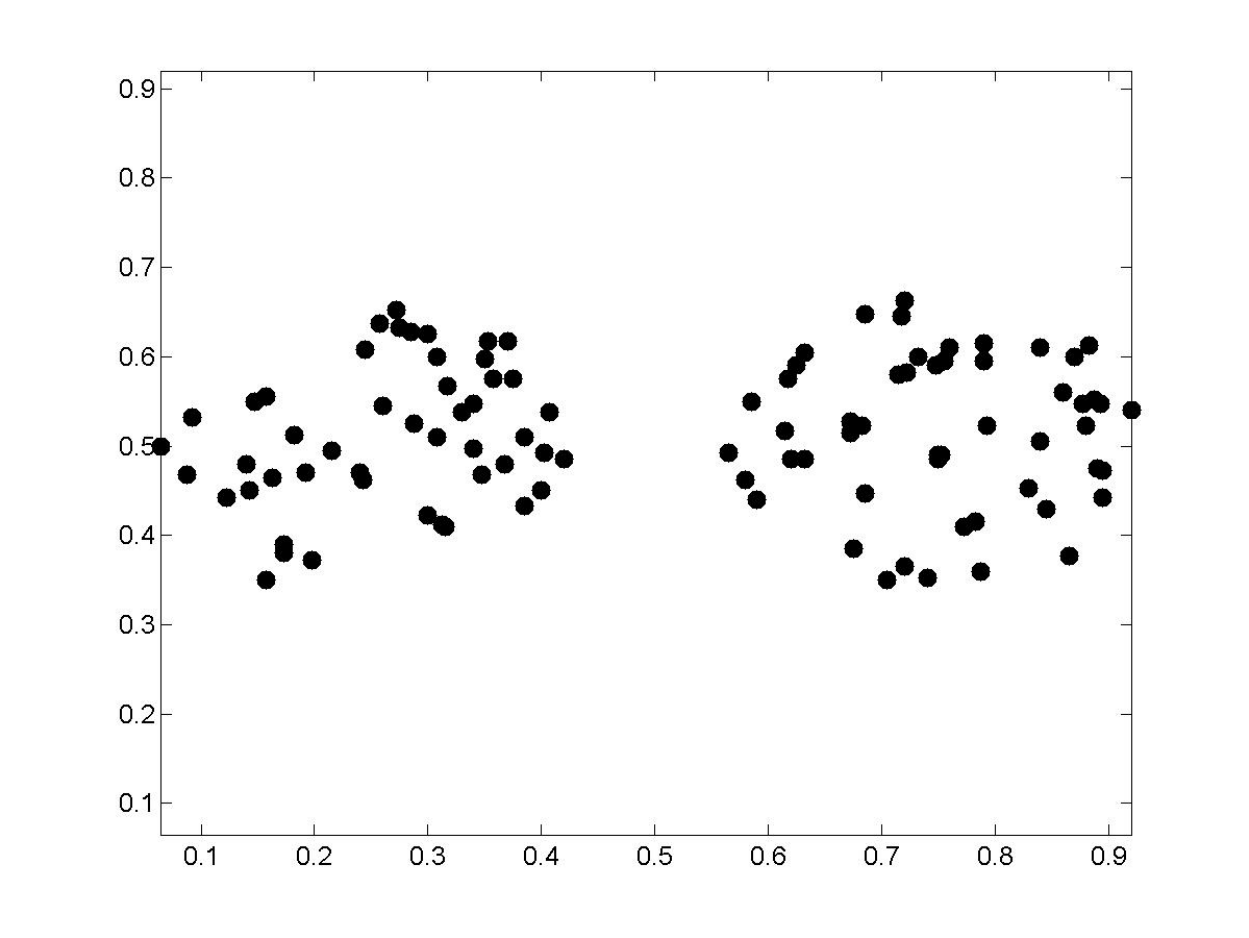
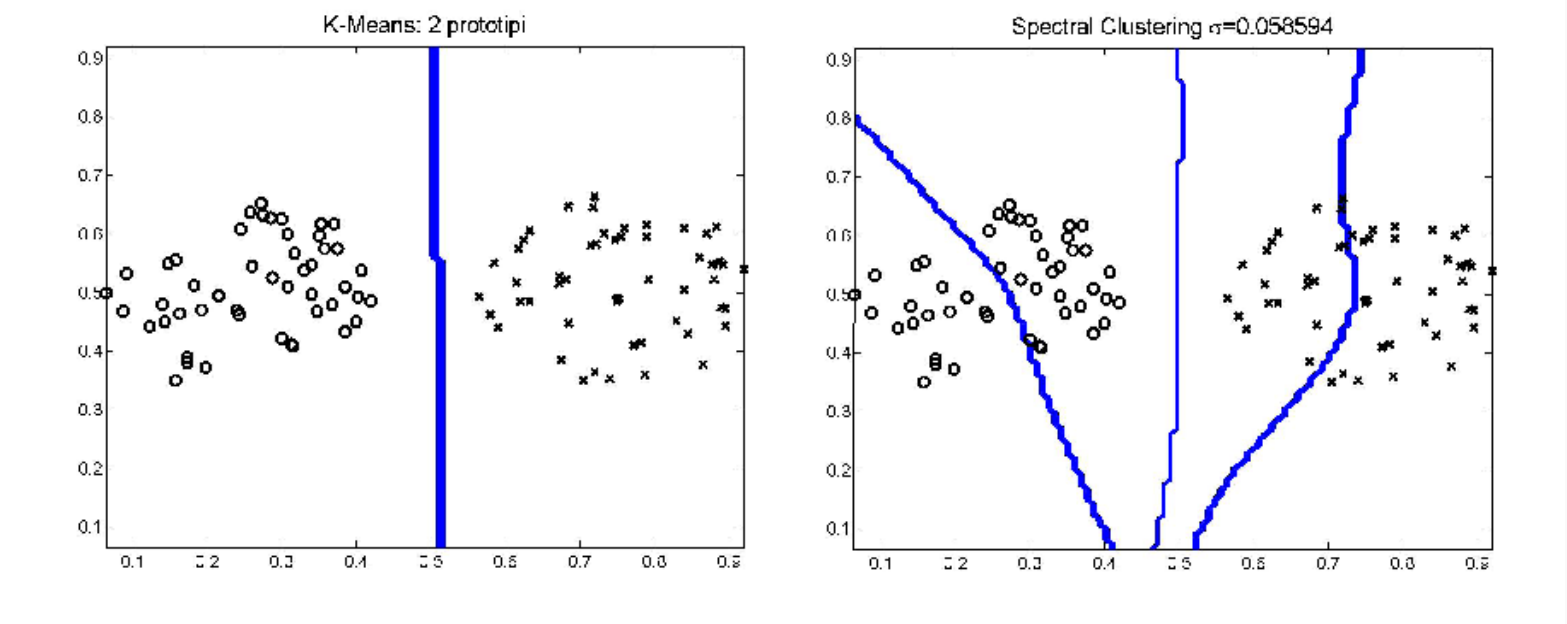


Figura 1: Insiemi di dati da partizionare

Simulazione: Due partizioni sferiche



Simulazione: Dati a mezzaluna

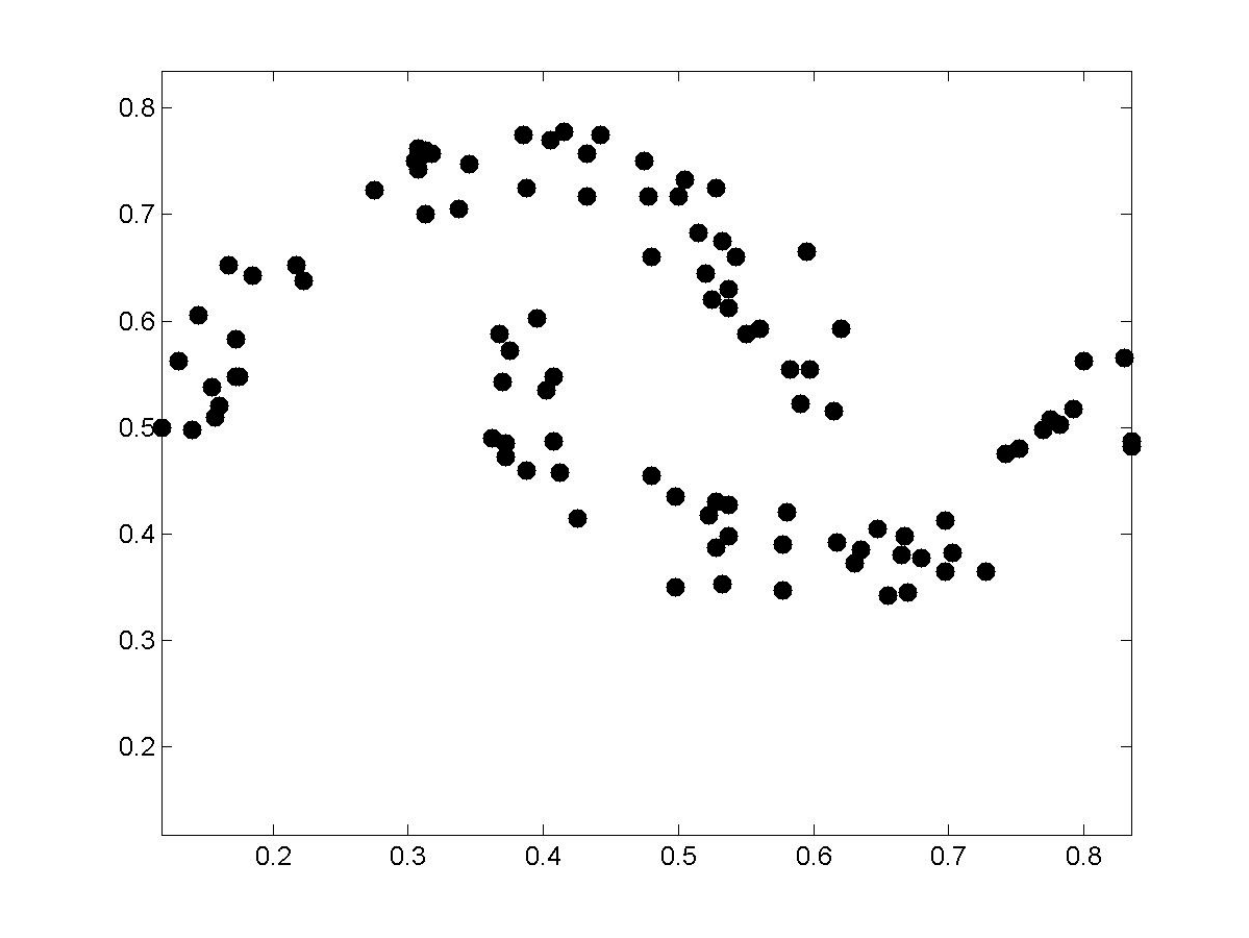
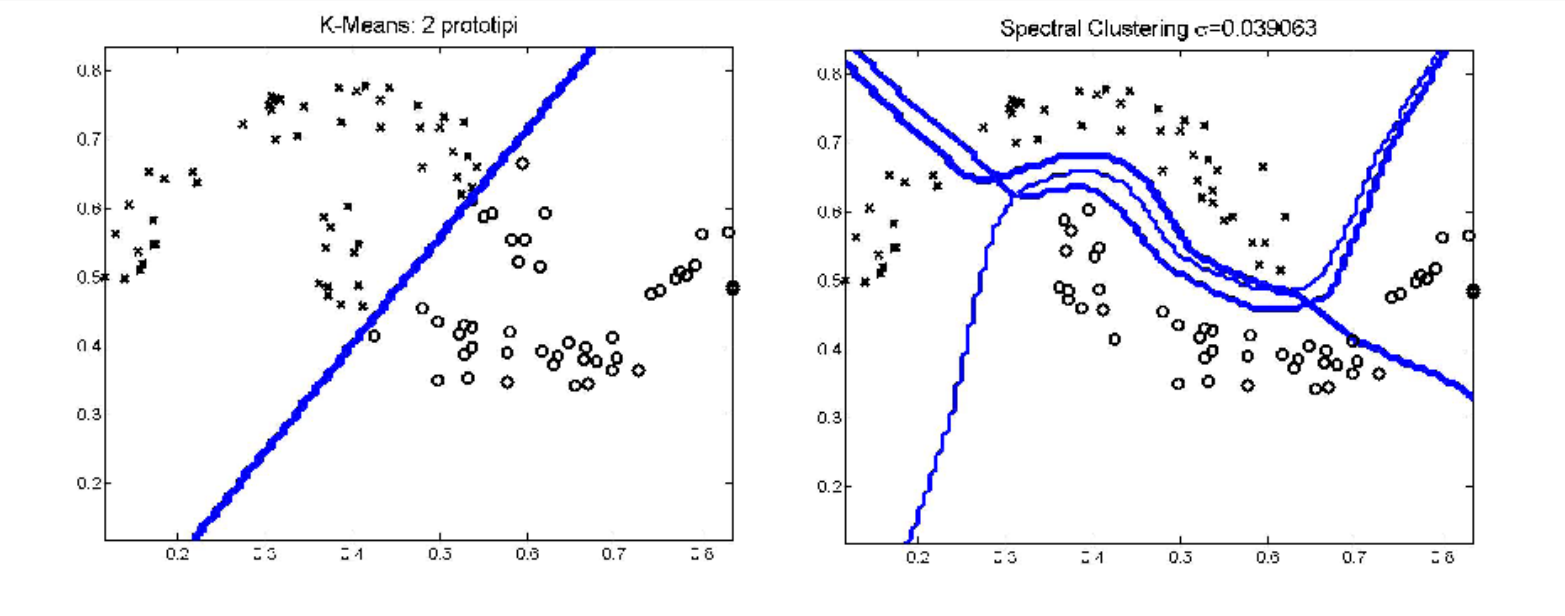


Figura 2: Insiemi di dati da partizionare

Simulazione: Dati a mezzaluna



Simulazione: Una scritta in corsivo

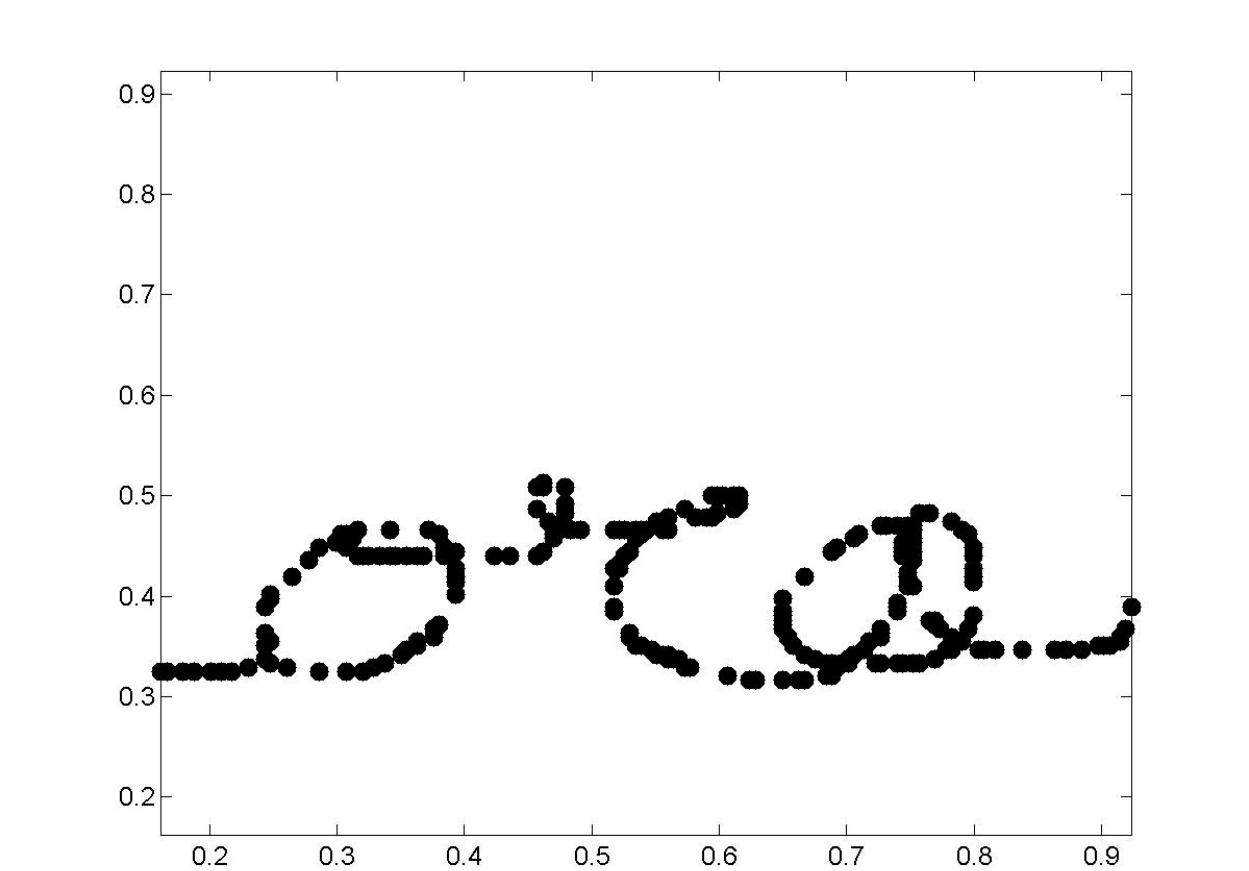
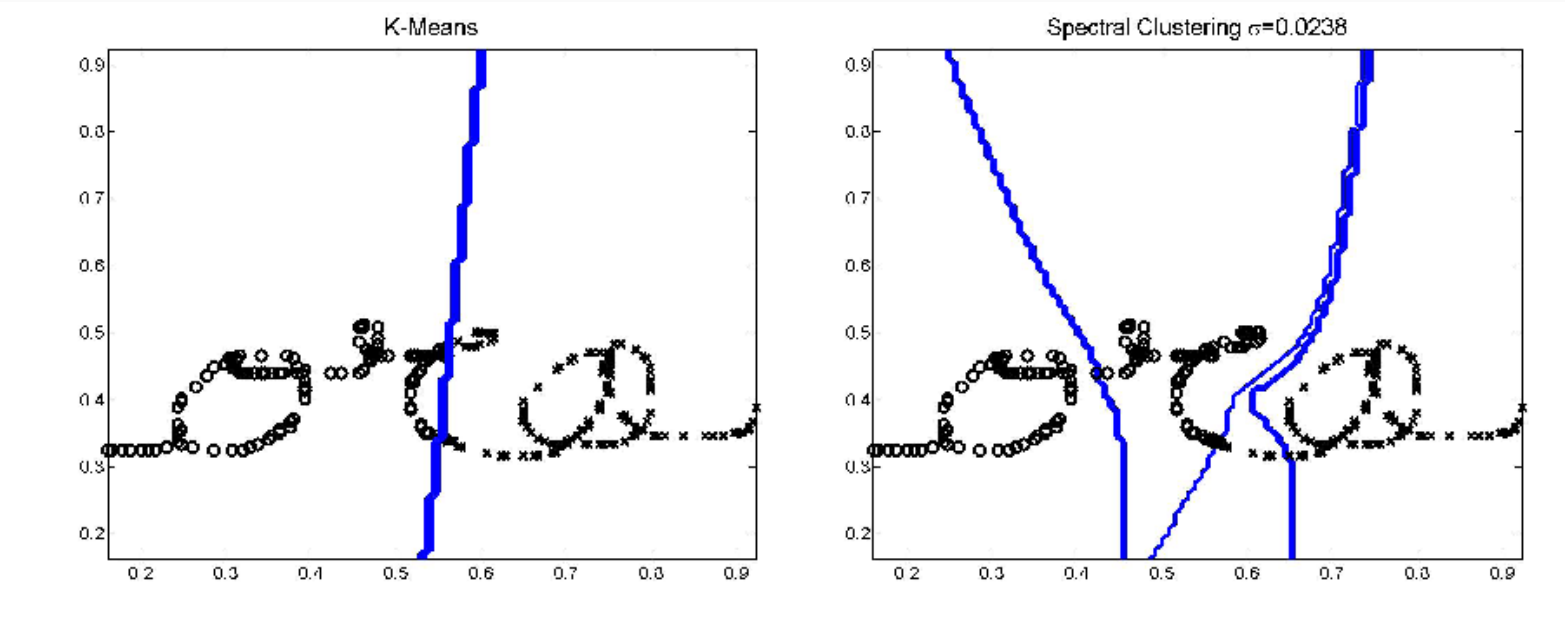


Figura 3: Insiemi di dati da partizionare

Simulazione: Una scritta in corsivo



Perché lo Spectral Clustering è utile?

- Perché rappresentando i dati iniziali in uno spazio dimensionale più piccolo li rende facilmente partizionabili
- Perché è facilmente implementabile
- Perché può essere risolto efficientemente tramite i metodi standard dell'algebra lineare
- Perché non fa forti assunzioni sulla forma dei cluster
- Perché il suo obiettivo non è trovare ottimi locali (come K-means)
- Ha successo in molte applicazioni come Community Detection, Image Segmentation, Speech Separation

Quali sono i potenziali problemi dello Spectral Clustering?

- Può essere sensibile alla scelta dei parametri (k in k-nearest neighbor graph)
- Non è molto chiaro come lavora su grafi non regolari
- Computazionalmente costoso su grandi grafi non sparsi