Alcuni algoritmi per la ricerca delle comunità su grafi

Paolo Dolce

13 febbraio 2015

Cosa sono le comunità?

Cosa sono le comunità?

Definizione (informale)

Sia $\mathscr{C} = \{C_1 \ldots, C_r\}$ una partizione di un grafo G. Diremo che $\mathscr{C}$ è *una suddivisione un comunità* per G se per ogni C_i il numero di spigoli con entrambe le estremità in C_i è "molto maggiore" del numero di spigoli con una sola estremità in C_i. Ogni elemento di $\mathscr{C}$ è una *comunità* di G.

Cosa sono le comunità?

Definizione (informale)

Sia $\mathscr{C} = \{C_1 \ldots, C_r\}$ una partizione di un grafo G. Diremo che $\mathscr{C}$ è *una suddivisione un comunità* per G se per ogni C_i il numero di spigoli con entrambe le estremità in C_i è "molto maggiore" del numero di spigoli con una sola estremità in C_i. Ogni elemento di $\mathscr{C}$ è una *comunità* di G.

- La divisione in comunità è una proprietà topologica di G che fornisce informazioni importanti sulla natura "dell'oggetto reale" modellizzato da G.

Cosa sono le comunità?

Definizione (informale)

Sia $\mathscr{C} = \{C_1 \ldots, C_r\}$ una partizione di un grafo G. Diremo che $\mathscr{C}$ *è una suddivisione un comunità* per G se per ogni C_i il numero di spigoli con entrambe le estremità in C_i è "molto maggiore" del numero di spigoli con una sola estremità in C_i. Ogni elemento di $\mathscr{C}$ è una *comunità* di G.

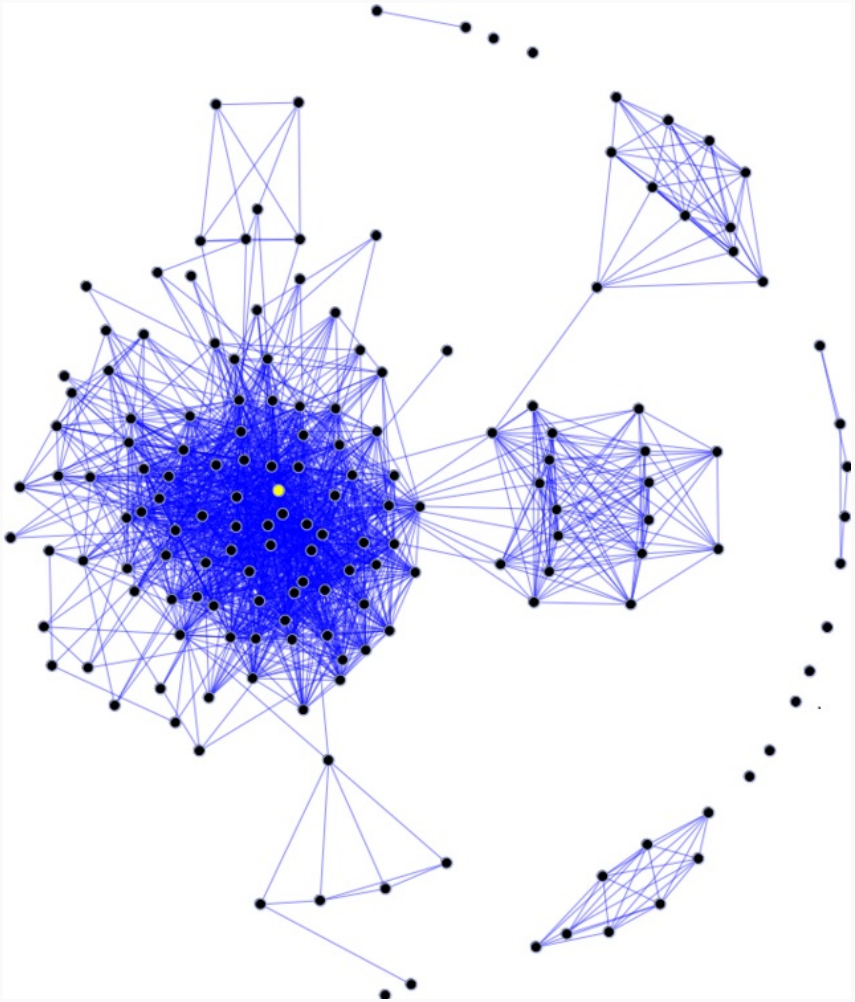
- La divisione in comunità è una proprietà topologica di G che fornisce informazioni importanti sulla natura "dell'oggetto reale" modellizzato da G.
- Reti sociali e reti biologiche in genere presentano comunità.

Cosa sono le comunità?

- In alcuni casi la presenza di comunità può essere direttamente verificata osservando una rappresetazione del grafo.

Cosa sono le comunità?

- In alcuni casi la presenza di comunità può essere direttamente verificata osservando una rappresetazione del grafo.



Modularità

Modularità

- Cerchiamo una misura su grafo partizionato che indichi la presenza di comunità (non banali).

Modularità

- Cerchiamo una misura su grafo partizionato che indichi la presenza di comunità (non banali).
- $\mathscr{C} = \{V(G), \emptyset\}$ è una divisione in comunità banale, in tal caso la nostra misura deve assumere valori "non significativi".

Modularità

- Cerchiamo una misura su grafo partizionato che indichi la presenza di comunità (non banali).
- $\mathscr{C} = \{V(G), \emptyset\}$ è una divisione in comunità banale, in tal caso la nostra misura deve assumere valori "non significativi".

Notazioni:

Modularità

- Cerchiamo una misura su grafo partizionato che indichi la presenza di comunità (non banali).
- $\mathscr{C} = \{V(G), \emptyset\}$ è una divisione in comunità banale, in tal caso la nostra misura deve assumere valori "non significativi".

Notazioni:

- G è un grafo non-diretto con n vertici ed m spigoli.

Modularità

- Cerchiamo una misura su grafo partizionato che indichi la presenza di comunità (non banali).
- $\mathscr{C} = \{V(G), \emptyset\}$ è una divisione in comunità banale, in tal caso la nostra misura deve assumere valori "non significativi".

Notazioni:

- G è un grafo non-diretto con n vertici ed m spigoli.
- $\mathscr{C} = \{C_1, \ldots, C_r\}$ è una qualsiasi partizione di G (fissata).

Modularità

- Cerchiamo una misura su grafo partizionato che indichi la presenza di comunità (non banali).
- $\mathscr{C} = \{V(G), \emptyset\}$ è una divisione in comunità banale, in tal caso la nostra misura deve assumere valori "non significativi".

Notazioni:

- G è un grafo non-diretto con n vertici ed m spigoli.
- $\mathscr{C} = \{C_1, \ldots, C_r\}$ è una qualsiasi partizione di G (fissata).
- $A = [A_{ij}]_{i,j}$ è la matrice di adiacenza di G.

Modularità

- Cerchiamo una misura su grafo partizionato che indichi la presenza di comunità (non banali).
- $\mathscr{C} = \{V(G), \emptyset\}$ è una divisione in comunità banale, in tal caso la nostra misura deve assumere valori "non significativi".

Notazioni:

- G è un grafo non-diretto con n vertici ed m spigoli.
- $\mathscr{C} = \{C_1, \ldots, C_r\}$ è una qualsiasi partizione di G (fissata).
- $A = [A_{ij}]_{i,j}$ è la matrice di adiacenza di G.
- k_i è il grado del vertice i.

Modularità

- Cerchiamo una misura su grafo partizionato che indichi la presenza di comunità (non banali).
- $\mathscr{C} = \{V(G), \emptyset\}$ è una divisione in comunità banale, in tal caso la nostra misura deve assumere valori "non significativi".

Notazioni:

- G è un grafo non-diretto con n vertici ed m spigoli.
- $\mathscr{C} = \{C_1, \ldots, C_r\}$ è una qualsiasi partizione di G (fissata).
- $A = [A_{ij}]_{i,j}$ è la matrice di adiacenza di G.
- k_i è il grado del vertice i.
- Per ogni coppia $i, j \in V(G)$:

$$\Delta_{ij} = \begin{cases} 1 & \text{se } i \text{ e } j \text{ stanno nello stesso elemento di } \mathscr{C} \\ 0 & \text{altrimenti} \end{cases}$$

Modularità

Costruzione della misura di modularità:

Modularità

Costruzione della misura di modularità:

- $$Q_1(\mathscr{C}) := \frac{1}{2} \sum_{i,j} A_{ij} \Delta_{ij}$$

  è il numero complessivo di spigoli che hanno entrambe le estremità nello stesso gruppo

Modularità

Costruzione della misura di modularità:

- $$Q_1(\mathscr{C}) := \frac{1}{2}\sum_{i,j} A_{ij}\Delta_{ij}$$

  è il numero complessivo di spigoli che hanno entrambe le estremità nello stesso gruppo

- $$Q_2(\mathscr{C}) := \frac{1}{2}\sum_{i,j} \frac{k_i k_j}{2m}\Delta_{ij}$$

  è il numero atteso di spigoli con entrambe le estremità nello stesso gruppo (preservando i gradi).

Modularità

Definizione

La *modularità* della partizione $\mathscr{C}$ è

$$Q(\mathscr{C}) := \frac{Q_1(\mathscr{C}) - Q_2(\mathscr{C})}{m} = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \Delta_{ij} \in [-0.5, 1[$$

Inoltre $\mathscr{C}$ è una divisione in comunità se $Q(\mathscr{C}) \geq 0.3$.

Modularità

Definizione

La *modularità* della partizione $\mathscr{C}$ è

$$Q(\mathscr{C}) := \frac{Q_1(\mathscr{C}) - Q_2(\mathscr{C})}{m} = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \Delta_{ij} \in [-0.5, 1[$$

Inoltre $\mathscr{C}$ è una divisione in comunità se $Q(\mathscr{C}) \geq 0.3$.

Ricerca delle comunità mediante la modularità.

Individuare, fra tutte le possibili scelte, una partizione $\mathscr{C}$ di G che abbia modularità massima. Se la modularità di $\mathscr{C}$ è almeno 0.3, allora $\mathscr{C}$ è una divisione in comunità.

Modularità

Definizione

La *modularità* della partizione $\mathscr{C}$ è

$$Q(\mathscr{C}) := \frac{Q_1(\mathscr{C}) - Q_2(\mathscr{C})}{m} = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \Delta_{ij} \in [-0.5, 1[$$

Inoltre $\mathscr{C}$ è una divisione in comunità se $Q(\mathscr{C}) \geq 0.3$.

Ricerca delle comunità mediante la modularità.

Individuare, fra tutte le possibili scelte, una partizione $\mathscr{C}$ di G che abbia modularità massima. Se la modularità di $\mathscr{C}$ è almeno 0.3, allora $\mathscr{C}$ è una divisione in comunità.

- Si tratta di un problema NP-hard.

Modularità

Definizione

La *modularità* della partizione $\mathscr{C}$ è

$$Q(\mathscr{C}) := \frac{Q_1(\mathscr{C}) - Q_2(\mathscr{C})}{m} = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \Delta_{ij} \in [-0.5, 1[$$

Inoltre $\mathscr{C}$ è una divisione in comunità se $Q(\mathscr{C}) \geq 0.3$.

Ricerca delle comunità mediante la modularità.

Individuare, fra tutte le possibili scelte, una partizione $\mathscr{C}$ di G che abbia modularità massima. Se la modularità di $\mathscr{C}$ è almeno 0.3, allora $\mathscr{C}$ è una divisione in comunità.

- Si tratta di un problema NP-hard.
- Cerchiamo soluzioni approssimate mediante algoritmi euristici.

Modularità

Problema principale della modularità: resolution limit

In presenza di comunità "piccole", la partizione con la modularità massima non è quella "più naturale".

Modularità

Problema principale della modularità: resolution limit

In presenza di comunità "piccole", la partizione con la modularità massima non è quella "più naturale".

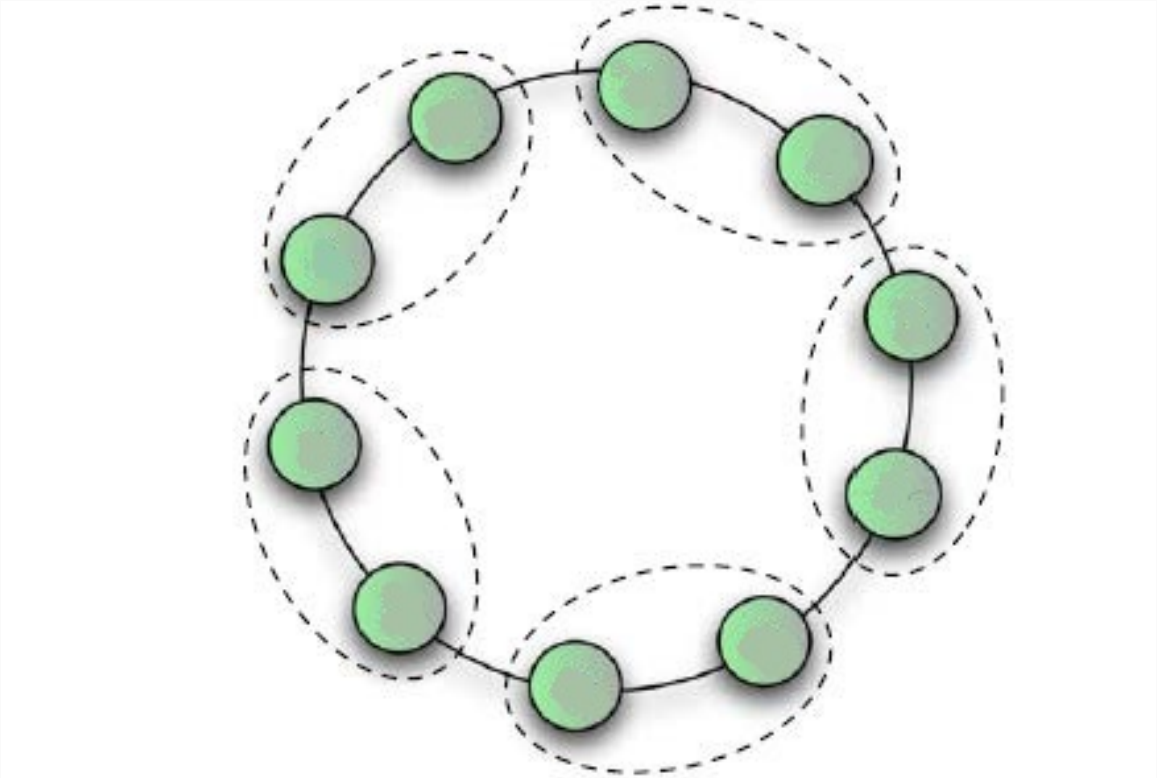
Esempio:

Modularità

Problema principale della modularità: resolution limit

In presenza di comunità "piccole", la partizione con la modularità massima non è quella "più naturale".

Esempio:



Algoritmo “Fastgreedy”

Vantaggi:

Algoritmo “Fastgreedy”

Vantaggi:

- Di facile implementazione.

Algoritmo “Fastgreedy”

Vantaggi:

- Di facile implementazione.
- Per la maggior parte dei grafi “reali”, il running time è $O(nd \log n)$ (d numero naturale positivo)

Algoritmo “Fastgreedy”

Vantaggi:

- Di facile implementazione.
- Per la maggior parte dei grafi “reali”, il running time è $O(nd \log n)$ (d numero naturale positivo)

Svantaggi:

Algoritmo “Fastgreedy”

Vantaggi:

- Di facile implementazione.
- Per la maggior parte dei grafi “reali”, il running time è $O(nd \log n)$ (d numero naturale positivo)

Svantaggi:

- Se il grafo è molto sparso l’algoritmo non riesce a trovare comunità anche se presenti

Algoritmo “Fastgreedy”

Idea dell’algoritmo:

Algoritmo “Fastgreedy”

Idea dell’algoritmo:

- Si parte con una partizione $\mathscr{C}_0 = \{\{x\} : x \in V(G)\}$.

Algoritmo “Fastgreedy”

Idea dell’algoritmo:

- Si parte con una partizione $\mathscr{C}_0 = \{\{x\} : x \in V(G)\}$.
- Per ogni coppia C_h e C_k della partizione $\mathscr{C}_n$ ($n \in \mathbb{N}$) si considera l’ipotetica partizione $\mathscr{C}_n^{(h,k)}$ identica a $\mathscr{C}_n$, eccetto che al posto di C_h e C_k sostituiamo $C_h \cup C_k$.

Algoritmo "Fastgreedy"

Idea dell'algoritmo:

- Si parte con una partizione $\mathscr{C}_0 = \{\{x\} : x \in V(G)\}$.
- Per ogni coppia C_h e C_k della partizione $\mathscr{C}_n$ ($n \in \mathbb{N}$) si considera l'ipotetica partizione $\mathscr{C}_n^{(h,k)}$ identica a $\mathscr{C}_n$, eccetto che al posto di C_h e C_k sostituiamo $C_h \cup C_k$.
- Fra tutte le partizioni $\mathscr{C}_n^{(h,k)}$ scegliamone una tale per cui la quantità $Q\left(\mathscr{C}_n^{(h,k)}\right) - Q(\mathscr{C}_n)$ sia massima. Tale partizione sarà proprio $\mathscr{C}_{n+1}$.

Algoritmo "Fastgreedy"

Idea dell'algoritmo:

- Si parte con una partizione $\mathscr{C}_0 = \{\{x\} : x \in V(G)\}$.
- Per ogni coppia C_h e C_k della partizione $\mathscr{C}_n$ ($n \in \mathbb{N}$) si considera l'ipotetica partizione $\mathscr{C}_n^{(h,k)}$ identica a $\mathscr{C}_n$, eccetto che al posto di C_h e C_k sostituiamo $C_h \cup C_k$.
- Fra tutte le partizioni $\mathscr{C}_n^{(h,k)}$ scegliamone una tale per cui la quantità $Q\left(\mathscr{C}_n^{(h,k)}\right) - Q(\mathscr{C}_n)$ sia massima. Tale partizione sarà proprio $\mathscr{C}_{n+1}$.
- Continuiamo in tal modo fino a raggiungere un certo $N \in \mathbb{N}$ tale per cui $\mathscr{C}_N = \{V(G)\}$ (non ci sono più gruppi da unire).

Algoritmo "Fastgreedy"

Idea dell'algoritmo:

- Si parte con una partizione $\mathscr{C}_0 = \{\{x\} : x \in V(G)\}$.
- Per ogni coppia C_h e C_k della partizione $\mathscr{C}_n$ ($n \in \mathbb{N}$) si considera l'ipotetica partizione $\mathscr{C}_n^{(h,k)}$ identica a $\mathscr{C}_n$, eccetto che al posto di C_h e C_k sostituiamo $C_h \cup C_k$.
- Fra tutte le partizioni $\mathscr{C}_n^{(h,k)}$ scegliamone una tale per cui la quantità $Q\left(\mathscr{C}_n^{(h,k)}\right) - Q(\mathscr{C}_n)$ sia massima. Tale partizione sarà proprio $\mathscr{C}_{n+1}$.
- Continuiamo in tal modo fino a raggiungere un certo $N \in \mathbb{N}$ tale per cui $\mathscr{C}_N = \{V(G)\}$ (non ci sono più gruppi da unire).
- L'ouput dell'algoritmo è un elemento di $\mathscr{C}_0, \mathscr{C}_1, \ldots, \mathscr{C}_N$ con modularità massima.

Modello stocastico a blocchi (stochastic block model)

Modello stocastico a blocchi (stochastic block model)

- Vogliamo generare un grafo G con n vertici.

Modello stocastico a blocchi (stochastic block model)

- Vogliamo generare un grafo G con n vertici.
- I vertici sono partizionati in r gruppi $C_1, \ldots, C_r$

Modello stocastico a blocchi (stochastic block model)

- Vogliamo generare un grafo G con n vertici.
- I vertici sono partizionati in r gruppi $C_1, \ldots, C_r$
- È data una matrice quadrata reale simmetrica $M = [M_{hk}]$ di dimensione $r \times r$ tale che $M_{hk} \in [0, 1]$.

Modello stocastico a blocchi (stochastic block model)

- Vogliamo generare un grafo G con n vertici.
- I vertici sono partizionati in r gruppi $C_1, \ldots, C_r$
- È data una matrice quadrata reale simmetrica $M = [M_{hk}]$ di dimensione $r \times r$ tale che $M_{hk} \in [0, 1]$.
- Per $i \in C_h$ e $j \in C_k$ la probabilità che ci sia uno spigolo fra i e j è $P(A_{ij} = 1) = M_{hk}$.

Modello stocastico a blocchi (stochastic block model)

- Vogliamo generare un grafo G con n vertici.
- I vertici sono partizionati in r gruppi $C_1, \ldots, C_r$
- È data una matrice quadrata reale simmetrica $M = [M_{hk}]$ di dimensione $r \times r$ tale che $M_{hk} \in [0, 1]$.
- Per $i \in C_h$ e $j \in C_k$ la probabilità che ci sia uno spigolo fra i e j è $P(A_{ij} = 1) = M_{hk}$.
- G viene generato in modo probabilistico tramite M.

Modello stocastico a blocchi (stochastic block model)

- Vogliamo generare un grafo G con n vertici.
- I vertici sono partizionati in r gruppi $C_1, \ldots, C_r$
- È data una matrice quadrata reale simmetrica $M = [M_{hk}]$ di dimensione $r \times r$ tale che $M_{hk} \in [0, 1]$.
- Per $i \in C_h$ e $j \in C_k$ la probabilità che ci sia uno spigolo fra i e j è $P(A_{ij} = 1) = M_{hk}$.
- G viene generato in modo probabilistico tramite M.
- In Python: G=igraph.Graph.SBM(n,M,c)

Modello stocastico a blocchi (stochastic block model)

- Vogliamo generare un grafo G con n vertici.
- I vertici sono partizionati in r gruppi $C_1, \ldots, C_r$
- È data una matrice quadrata reale simmetrica $M = [M_{hk}]$ di dimensione $r \times r$ tale che $M_{hk} \in [0, 1]$.
- Per $i \in C_h$ e $j \in C_k$ la probabilità che ci sia uno spigolo fra i e j è $P(A_{ij} = 1) = M_{hk}$.
- G viene generato in modo probabilistico tramite M.
- In Python: G=igraph.Graph.SBM(n,M,c)

Per ottenere un grafo con r comunità basta dare una matrice M i cui elementi della diagonale siano più grandi rispetto agli altri.

Applicazioni dell'algoritmo Fastgreedy

- n=1000; M=[[0.2,0.05],[0.05,0.2]]; c=[500,500]

Applicazioni dell'algoritmo Fastgreedy

- n=1000; M=[[0.2,0.05],[0.05,0.2]]; c=[500,500]

  G=igraph.Graph.SBM(n,M,c)

Applicazioni dell'algoritmo Fastgreedy

- n=1000; M=[[0.2,0.05],[0.05,0.2]]; c=[500,500]

  G=igraph.Graph.SBM(n,M,c)

  G.community_fastgreedy() trova come previsto le due comunità $[0, \ldots, 499]$, $[500, \ldots, 999]$

Applicazioni dell'algoritmo Fastgreedy

- n=1000; M=[[0.2,0.05],[0.05,0.2]]; c=[500,500]

  G=igraph.Graph.SBM(n,M,c)

  G.community_fastgreedy() trova come previsto le due comunità $[0, \ldots, 499]$, $[500, \ldots, 999]$
- n=1000; M=[[0.005,0.001],[0.001,0.005]]; c=[500,500]

Applicazioni dell'algoritmo Fastgreedy

- n=1000; M=[[0.2,0.05],[0.05,0.2]]; c=[500,500]

  G=igraph.Graph.SBM(n,M,c)

  G.community_fastgreedy() trova come previsto le due comunità $[0, \ldots, 499]$, $[500, \ldots, 999]$
- n=1000; M=[[0.005,0.001],[0.001,0.005]]; c=[500,500]

  G=igraph.Graph.SBM(n,M,c)

Applicazioni dell'algoritmo Fastgreedy

- n=1000; M=[[0.2,0.05],[0.05,0.2]]; c=[500,500]

  G=igraph.Graph.SBM(n,M,c)

  G.community_fastgreedy() trova come previsto le due comunità $[0, \ldots, 499]$, $[500, \ldots, 999]$
- n=1000; M=[[0.005,0.001],[0.001,0.005]]; c=[500,500]

  G=igraph.Graph.SBM(n,M,c)

  G.community_fastgreedy() restituisce come output una partizione in 91 comunità che in realtà non esistono.

Algoritmo spettrale di Newman

Algoritmo spettrale di Newman

- Supponiamo che G sia un grafo **non eccessivamente sparso** con n vertici, costruito con SBM dove $r = 2$ e tale che $p_{\text{in}} := M_{11} = M_{22}$ e $p_{\text{out}} := M_{12} = M_{21}$.

Algoritmo spettrale di Newman

- Supponiamo che G sia un grafo **non eccessivamente sparso** con n vertici, costruito con SBM dove $r = 2$ e tale che $p_{\text{in}} := M_{11} = M_{22}$ e $p_{\text{out}} := M_{12} = M_{21}$.
- Siano: $c_{\text{in}} = np_{\text{in}}$; $c_{\text{out}} = np_{\text{out}}$.

Algortirmo spettrale di Newman

La matrice di adiacenza A ha le seguenti proprietà spettrali per $n \to \infty$ e quando $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$:

Algortirmo spettrale di Newman

La matrice di adiacenza A ha le seguenti proprietà spettrali per $n \to \infty$ e quando $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$:

- Gli autovalori $\lambda_3, \ldots, \lambda_n$ sono confinati nell'intervallo reale

$$\left[-\sqrt{2(c_{\text{in}} + c_{\text{out}})}, \sqrt{2(c_{\text{in}} + c_{\text{out}})}\right]$$

.

Algortirmo spettrale di Newman

La matrice di adiacenza A ha le seguenti proprietà spettrali per $n \to \infty$ e quando $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$:

- Gli autovalori $\lambda_3, \ldots, \lambda_n$ sono confinati nell'intervallo reale

$$\left[-\sqrt{2(c_{\text{in}} + c_{\text{out}})}, \sqrt{2(c_{\text{in}} + c_{\text{out}})}\right]$$

.

- $\lambda_1, \lambda_2 >> \sqrt{2(c_{\text{in}} + c_{\text{out}})}$.

Algortirmo spettrale di Newman

La matrice di adiacenza A ha le seguenti proprietà spettrali per $n \to \infty$ e quando $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$:

- Gli autovalori $\lambda_3, \ldots, \lambda_n$ sono confinati nell'intervallo reale

$$\left[-\sqrt{2(c_{\text{in}} + c_{\text{out}})}, \sqrt{2(c_{\text{in}} + c_{\text{out}})}\right]$$

.

- $\lambda_1, \lambda_2 >> \sqrt{2(c_{\text{in}} + c_{\text{out}})}$.
- $\lambda_1 = \frac{1}{2}(c_{\text{in}} - c_{\text{out}}) + \frac{c_{\text{in}}+c_{\text{out}}}{c_{\text{in}}-c_{\text{out}}}$; $\quad \lambda_2 = \frac{1}{2}(c_{\text{in}} + c_{\text{out}}) + 1$

Algortirmo spettrale di Newman

La matrice di adiacenza A ha le seguenti proprietà spettrali per $n \to \infty$ e quando $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$:

- Gli autovalori $\lambda_3, \ldots, \lambda_n$ sono confinati nell'intervallo reale

$$\left[-\sqrt{2(c_{\text{in}} + c_{\text{out}})}, \sqrt{2(c_{\text{in}} + c_{\text{out}})}\right]$$

.

- $\lambda_1, \lambda_2 >> \sqrt{2(c_{\text{in}} + c_{\text{out}})}$.
- $\lambda_1 = \frac{1}{2}(c_{\text{in}} - c_{\text{out}}) + \frac{c_{\text{in}}+c_{\text{out}}}{c_{\text{in}}-c_{\text{out}}}$; $\quad \lambda_2 = \frac{1}{2}(c_{\text{in}} + c_{\text{out}}) + 1$
- Il segno dell'autovettore relativo a λ_2 assegna i vertici alle comunità.

Algoritmo spettrale di Newman

Riassumendo, sotto le seguenti 3 ipotesi:

Algoritmo spettrale di Newman

Riassumendo, sotto le seguenti 3 ipotesi:

- $n \to \infty$ (Grafo G molto grande).

Algoritmo spettrale di Newman

Riassumendo, sotto le seguenti 3 ipotesi:

- $n \to \infty$ (Grafo G molto grande).
- $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$.

Algoritmo spettrale di Newman

Riassumendo, sotto le seguenti 3 ipotesi:

- $n \to \infty$ (Grafo G molto grande).
- $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$.
- G sufficientemente denso.

Algoritmo spettrale di Newman

Riassumendo, sotto le seguenti 3 ipotesi:

- $n \to \infty$ (Grafo G molto grande).
- $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$.
- G sufficientemente denso.

La ricerca delle comunità avviene come segue:

Algoritmo spettrale di Newman

Riassumendo, sotto le seguenti 3 ipotesi:

- $n \to \infty$ (Grafo G molto grande).
- $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$.
- G sufficientemente denso.

La ricerca delle comunità avviene come segue:

- Si calcola "il secondo" autovalore λ_2 di A.

Algoritmo spettrale di Newman

Riassumendo, sotto le seguenti 3 ipotesi:

- $n \to \infty$ (Grafo G molto grande).
- $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$.
- G sufficientemente denso.

La ricerca delle comunità avviene come segue:

- Si calcola "il secondo" autovalore λ_2 di A.
- Se $\lambda_2 >> \sqrt{2(c_{\text{in}} + c_{\text{out}})}$ allora sono presenti 2 comunità e possiamo trovarle esplicitamente.

Algoritmo spettrale di Newman

Riassumendo, sotto le seguenti 3 ipotesi:

- $n \to \infty$ (Grafo G molto grande).
- $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$.
- G sufficientemente denso.

La ricerca delle comunità avviene come segue:

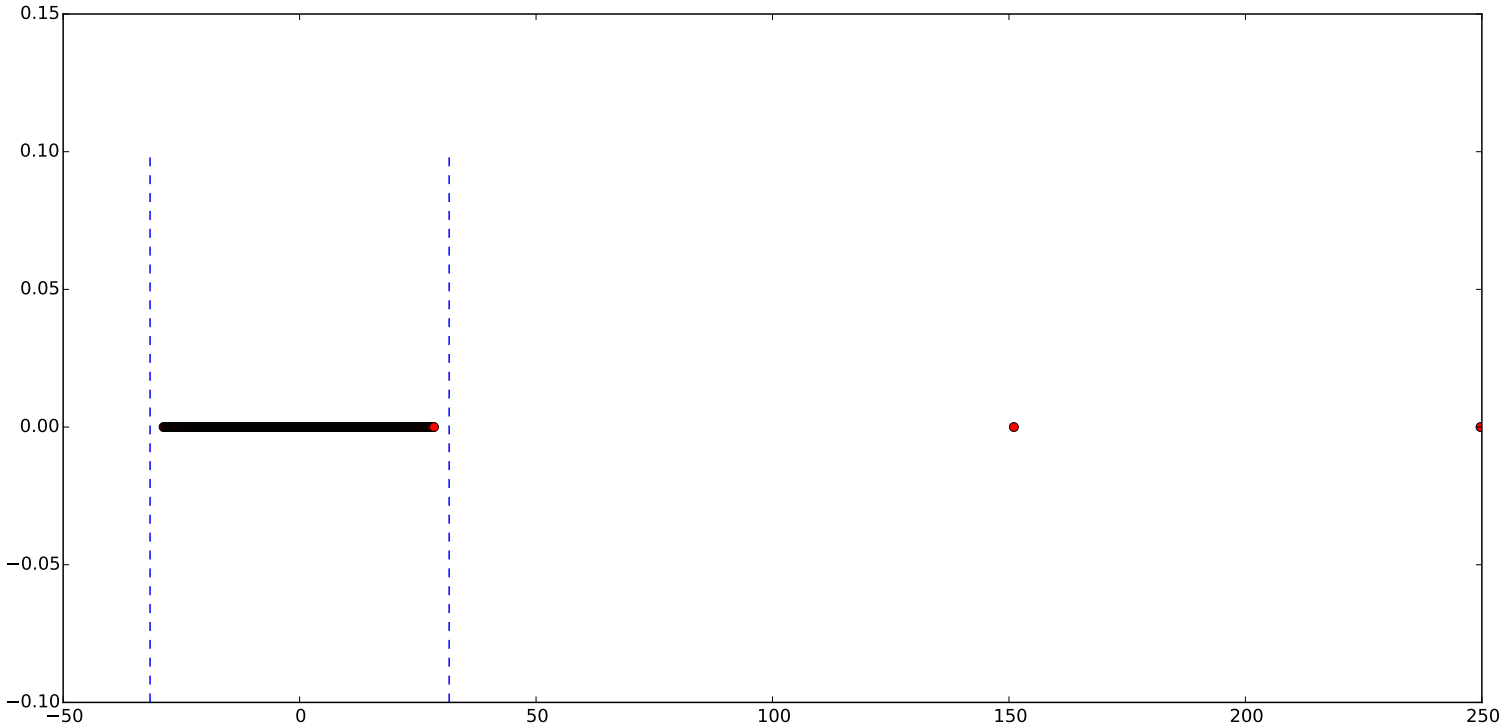
- Si calcola "il secondo" autovalore λ_2 di A.
- Se $\lambda_2 >> \sqrt{2(c_{\text{in}} + c_{\text{out}})}$ allora sono presenti 2 comunità e possiamo trovarle esplicitamente.
- Se il punto precedente è verificato, si calcola "il secondo" autovettore di A per assegnare i vertici alle rispettive comunità.

examples.spectral(n,M,c)

1) n=2000; M=[[0.2,0.05],[0.05,0.2]]; c=[1000,1000]

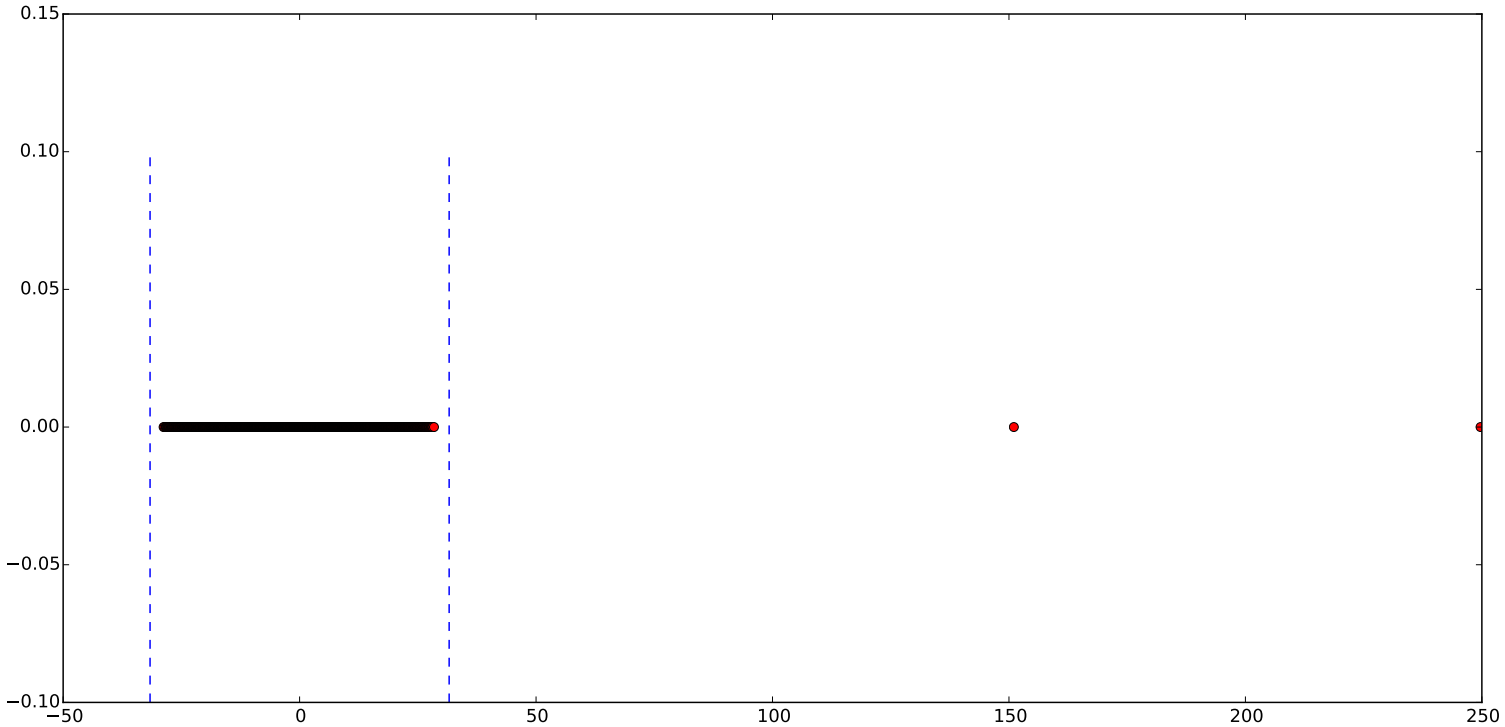
examples.spectral(n,M,c)

1) n=2000; M=[[0.2,0.05],[0.05,0.2]]; c=[1000,1000]



examples.spectral(n,M,c)

1) n=2000; M=[[0.2,0.05],[0.05,0.2]]; c=[1000,1000]



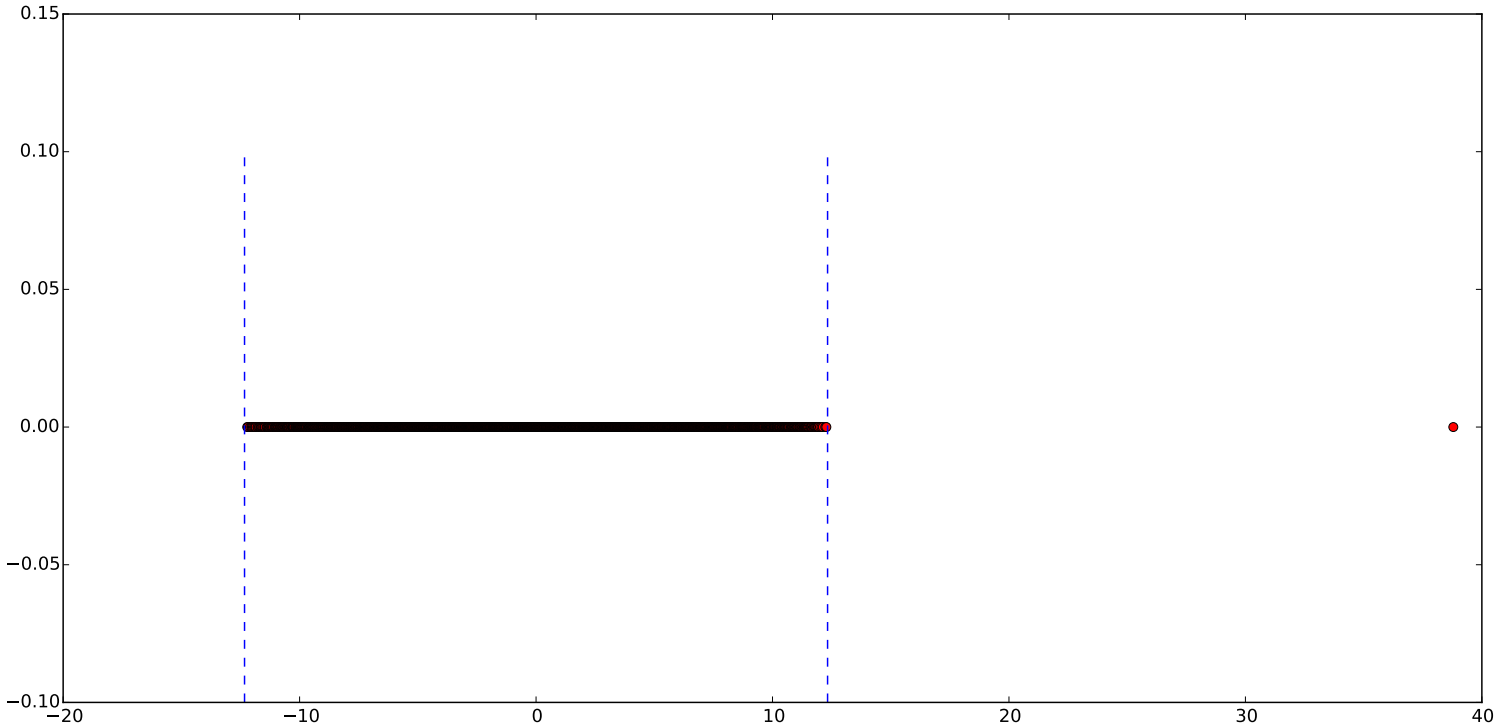
Troviamo le due comunità $C_1 = [0, \ldots, 999]$ e $C_2 = [1000, \ldots, 1999]$

examples.spectral(n,M,c)

2) n=2000; M=[[0.02,0.018],[0.018,0.02]]; c=[1000,1000]

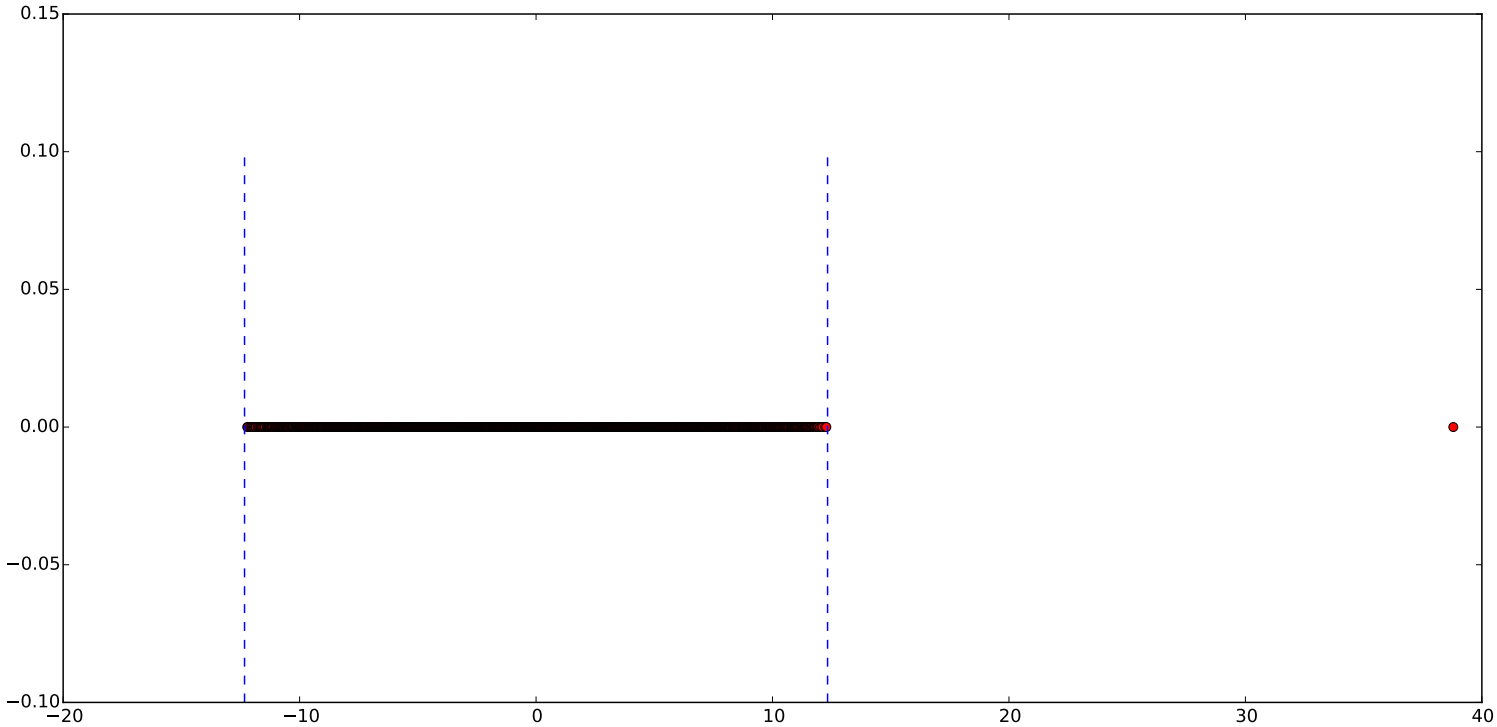
examples.spectral(n,M,c)

2) n=2000; M=[[0.02,0.018],[0.018,0.02]]; c=[1000,1000]



examples.spectral(n,M,c)

2) n=2000; M=[[0.02,0.018],[0.018,0.02]]; c=[1000,1000]



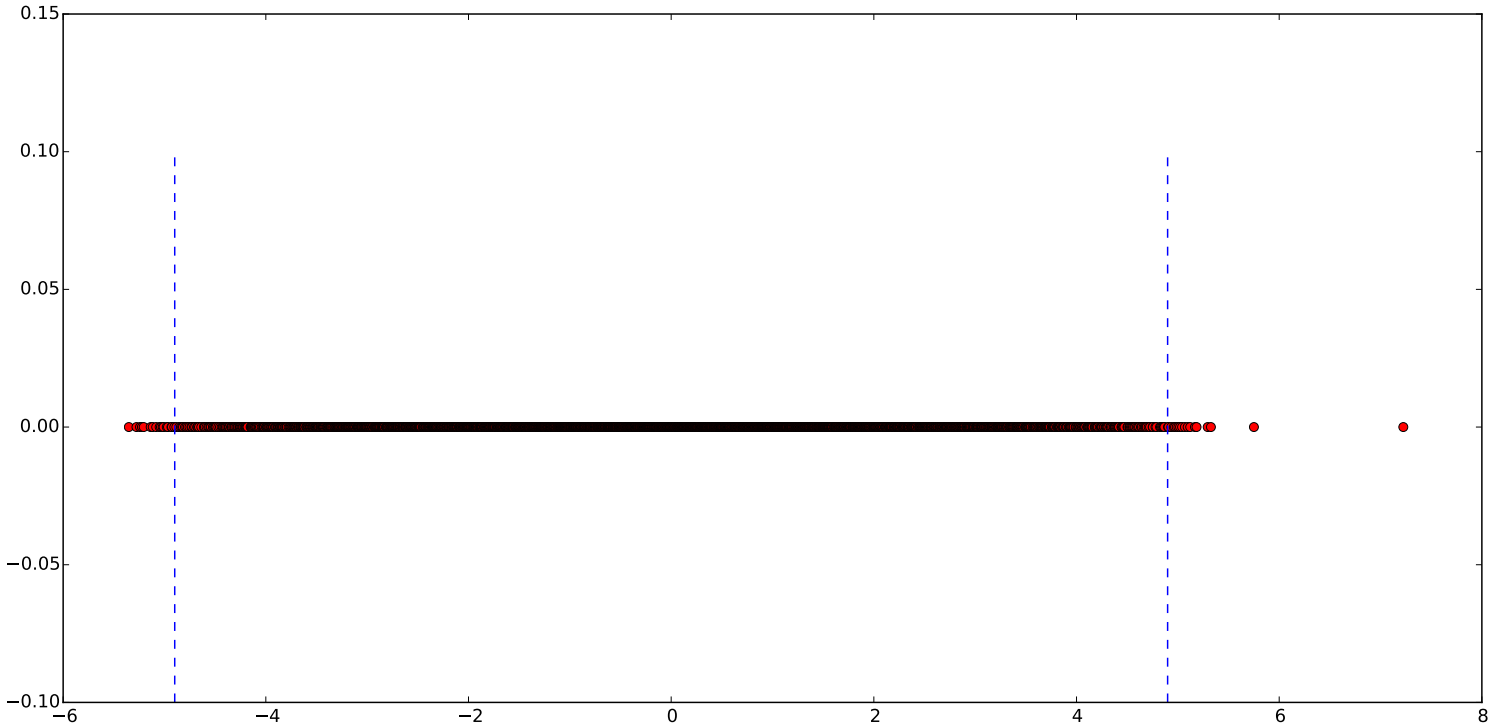
Troviamo le due comunità $C_1 = V(G)$ e $C_2 = \emptyset$

examples.spectral(n,M,c)

3) n=2000; M=[[0.005,0.001],[0.001,0.005]]; c=[1000,1000]

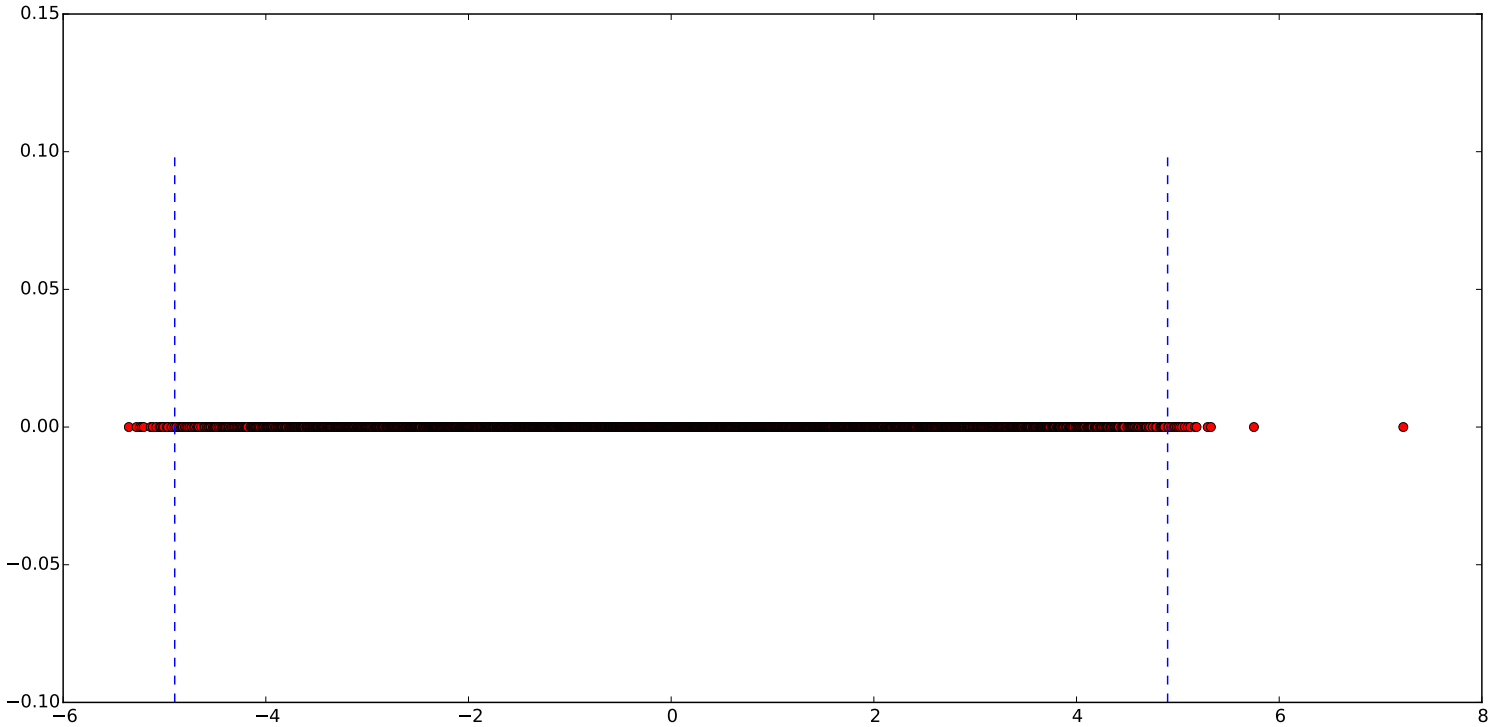
examples.spectral(n,M,c)

3) n=2000; M=[[0.005,0.001],[0.001,0.005]]; c=[1000,1000]



examples.spectral(n,M,c)

3) n=2000; M=[[0.005,0.001],[0.001,0.005]]; c=[1000,1000]



Troviamo le due comunità $C_1 = V(G) \setminus \{1105, 1226, 1367\}$ e $C_2 = [1105, 1226, 1367]$

“Spectral redemption”

Vogliamo un algortimo spettrale che possa funzionare anche nel caso di grafi molto sparsi (le altre due condizioni non possono essere eliminate).

“Spectral redemption”

Vogliamo un algortimo spettrale che possa funzionare anche nel caso di grafi molto sparsi (le altre due condizioni non possono essere eliminate).

- Trasformiamo G in un grafo diretto.

“Spectral redemption”

Vogliamo un algortimo spettrale che possa funzionare anche nel caso di grafi molto sparsi (le altre due condizioni non possono essere eliminate).

- Trasformiamo G in un grafo diretto.
- Definiamo la matrice B di dimensione $2m \times 2m$ sugli spigoli diretti di G:

$$B_{(x\to y)(w\to z)} = \begin{cases} 1 & y = w \text{ e } x \neq z \\ 0 & \text{altrimenti} \end{cases}$$

  B è detta matrice non-backtracking

“Spectral redemption”

Sotto le ipotesi:

- $n \to \infty$ (Grafo G molto grande).

“Spectral redemption”

Sotto le ipotesi:

- $n \to \infty$ (Grafo G molto grande).
- $c_{\mathsf{in}} - c_{\mathsf{out}} > \sqrt{2(c_{\mathsf{in}} + c_{\mathsf{out}})}$.

“Spectral redemption”

Sotto le ipotesi:

- $n \to \infty$ (Grafo G molto grande).
- $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$.

La matrice B ha le seguenti proprietà spettrali che indicano la presenza di comunità:

“Spectral redemption”

Sotto le ipotesi:

- $n \to \infty$ (Grafo G molto grande).
- $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$.

La matrice B ha le seguenti proprietà spettrali che indicano la presenza di comunità:

- Tutti gli autovalori, tranne i due di norma maggiore, sono contenuti nel cerchio (sul piano complesso) di raggio $\sqrt{2(c_{\text{in}} + c_{\text{out}})}$

“Spectral redemption”

Sotto le ipotesi:

- $n \to \infty$ (Grafo G molto grande).
- $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$.

La matrice B ha le seguenti proprietà spettrali che indicano la presenza di comunità:

- Tutti gli autovalori, tranne i due di norma maggiore, sono contenuti nel cerchio (sul piano complesso) di raggio $\sqrt{2(c_{\text{in}} + c_{\text{out}})}$
- I due autovalori di norma maggiore sono entrambi reali positivi e stanno fuori dal suddetto cerchio.

“Spectral redemption”

Sotto le ipotesi:

- $n \to \infty$ (Grafo G molto grande).
- $c_{\text{in}} - c_{\text{out}} > \sqrt{2(c_{\text{in}} + c_{\text{out}})}$.

La matrice B ha le seguenti proprietà spettrali che indicano la presenza di comunità:

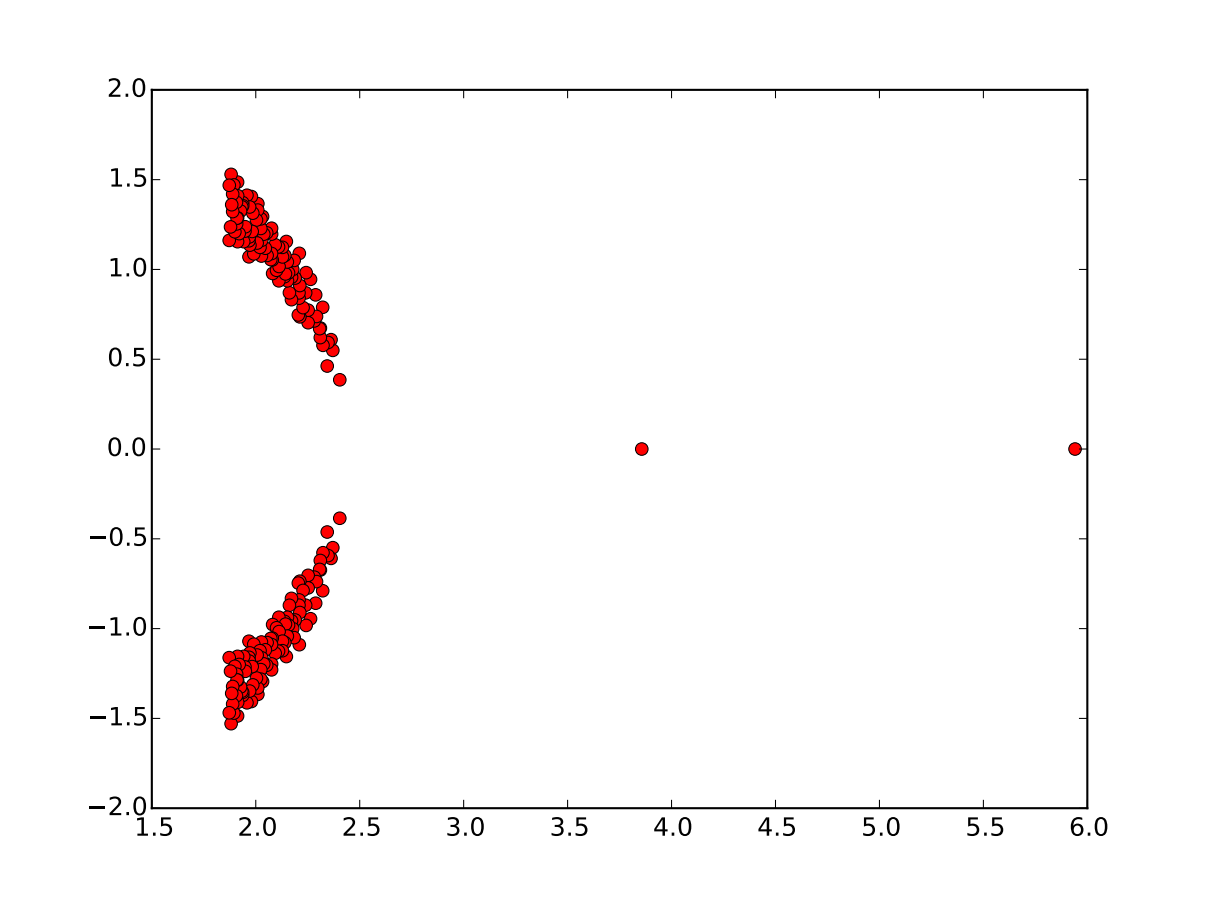
- Tutti gli autovalori, tranne i due di norma maggiore, sono contenuti nel cerchio (sul piano complesso) di raggio $\sqrt{2(c_{\text{in}} + c_{\text{out}})}$
- I due autovalori di norma maggiore sono entrambi reali positivi e stanno fuori dal suddetto cerchio.
- Il segno del secondo autovettore z_2 determina l'appartenenza alle comunità nel modo seguente: per ogni vertice $i \in V(G)$ si sommano le componenti di z_2 che corrispondono ad archi che puntano verso i. Se tale somma è non negativa, allora $i \in C_1$

examples.spectral_redemption(n,M,c)

n=2000; M=[[0.005,0.001],[0.001,0.005]]; c=[1000,1000]

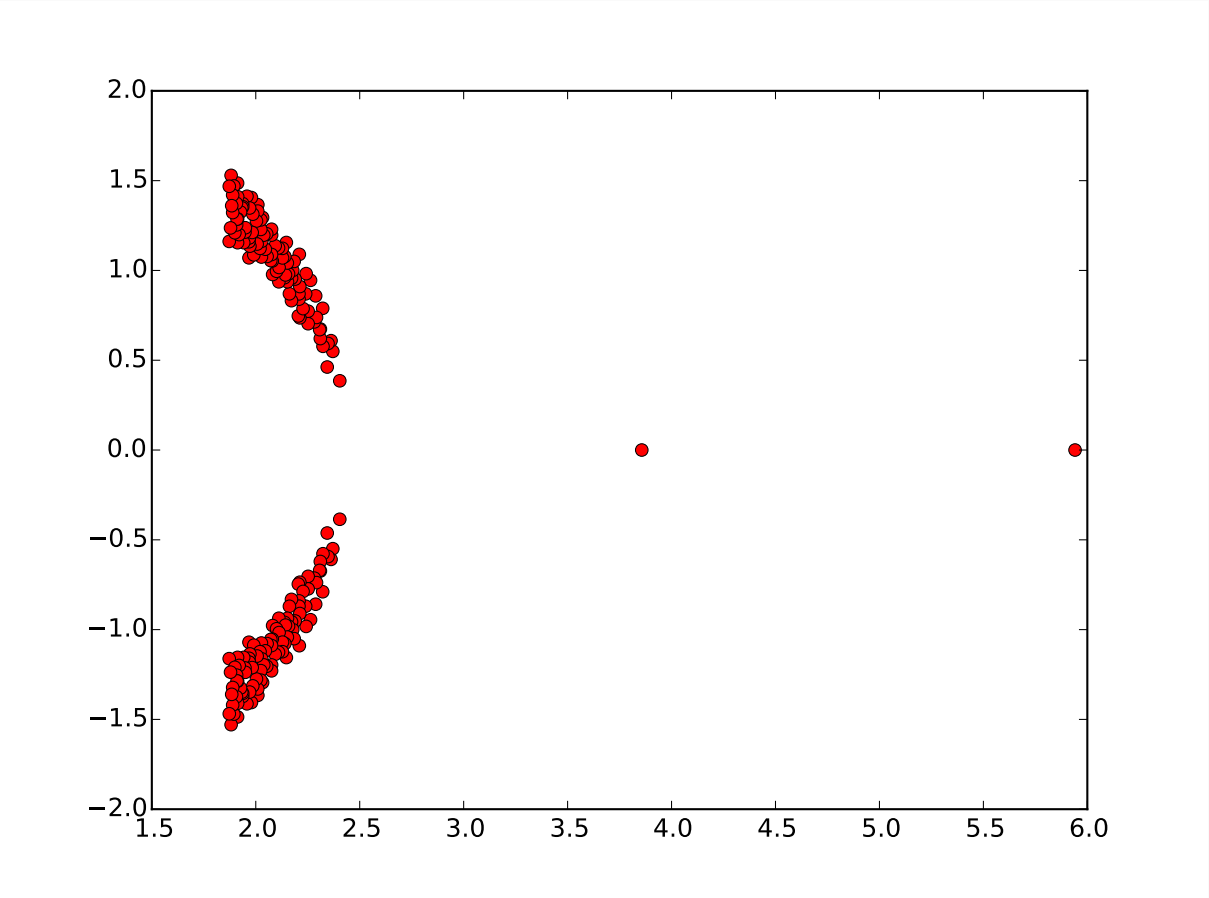
examples.spectral_redemption(n,M,c)

n=2000; M=[[0.005,0.001],[0.001,0.005]]; c=[1000,1000]



examples.spectral_redemption(n,M,c)

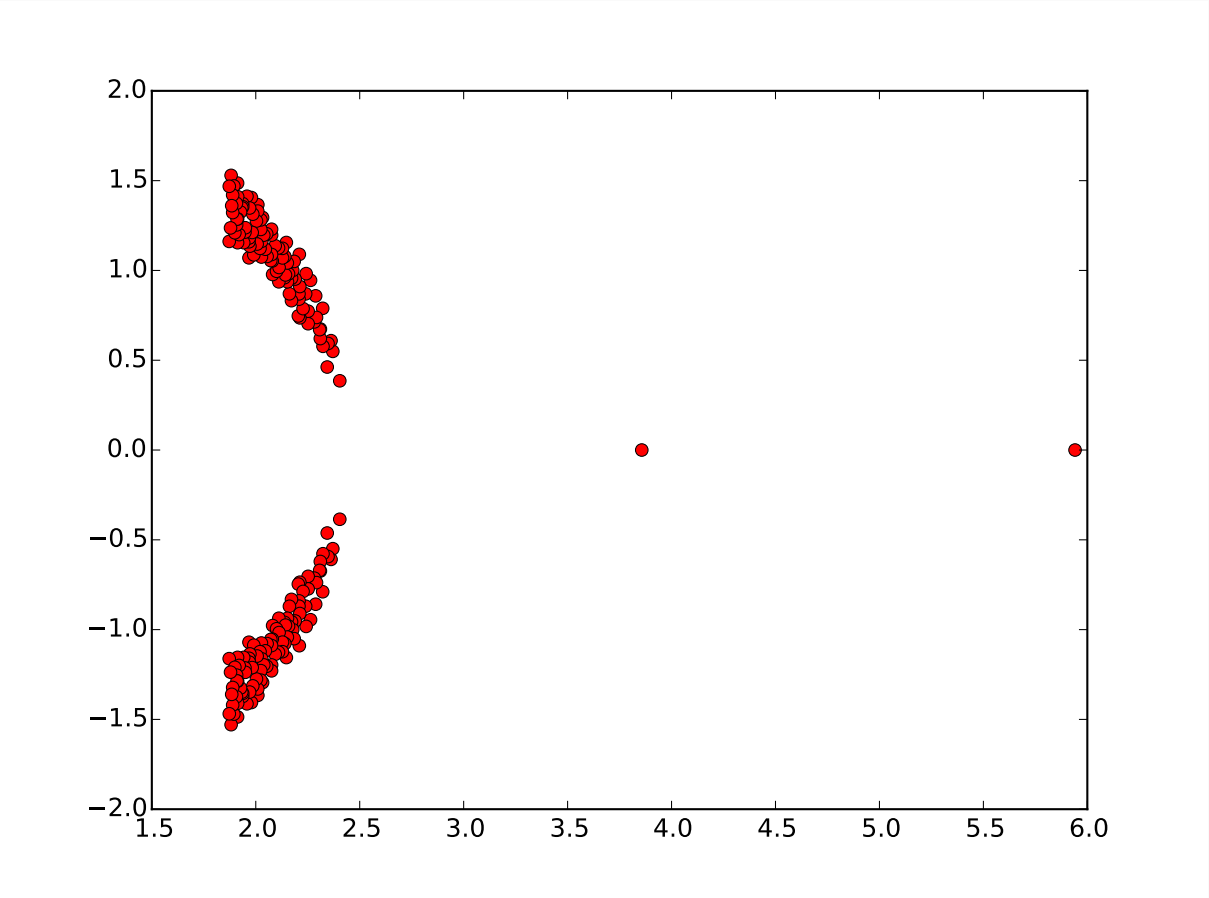
n=2000; M=[[0.005,0.001],[0.001,0.005]]; c=[1000,1000]



Troviamo le due comunità C_1 e C_2 tali che $|C_1| = 965$, $|C_2| = 1035$

examples.spectral_redemption(n,M,c)

n=2000; M=[[0.005,0.001],[0.001,0.005]]; c=[1000,1000]



Troviamo le due comunità C_1 e C_2 tali che $|C_1| = 965$, $|C_2| = 1035$ Inoltre 1799 vertici su 2000 sono stati classificati

Bibliografia

- S. Fortunato; Community detection in graphs; 2009
- R.R. Nadakuditi, M.E.J. Newman; Graph spectra and the detectability of community structure in networks; 2012
- F. Krzakala et al.; Spectral redemption in clustering sparse networks; 2013
- M.E.J. Newman; Spectral community detection in sparse networks; 2014
- M.E.J. Newman; Fast algorithm for detecting community structure in networks; 2004
- M.E.J. Newman; Finding community structure in very large networks; 2004
- E.Abbe, A.S. Bandeira, G. Hall; Exact Recovery in the Stochastic Block Model; 2014