Community detection and spectral clustering

Rui Vilela Mendes
CMAF, Lisbon
http://label2.ist.utl.pt/vilela/

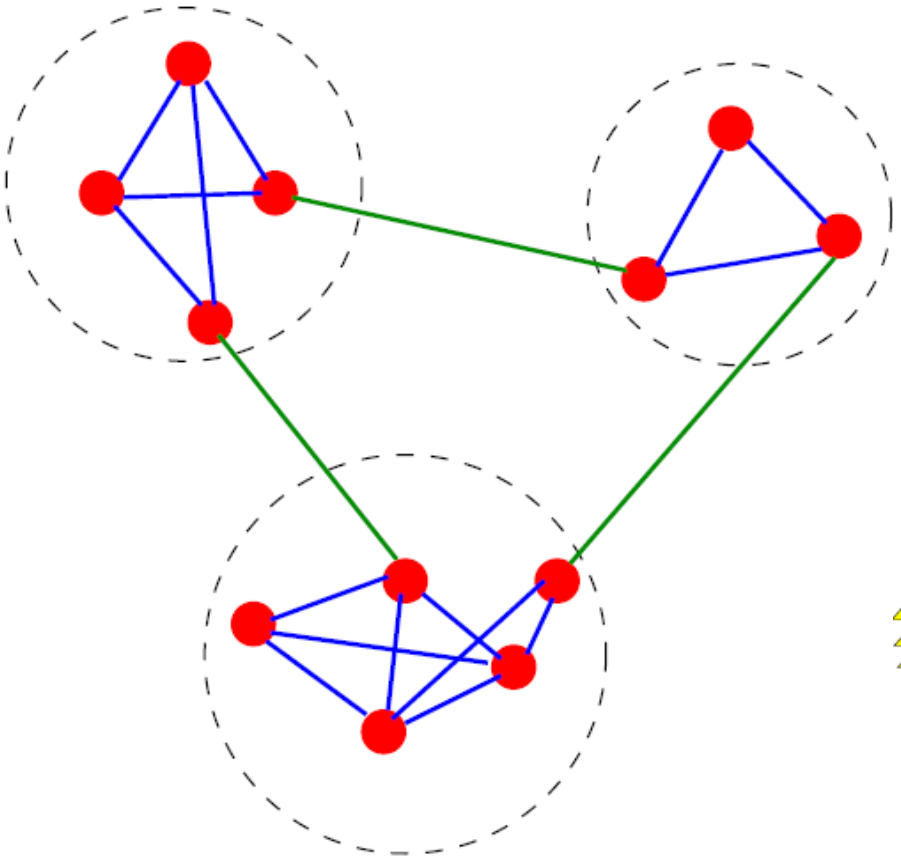
Data on networks

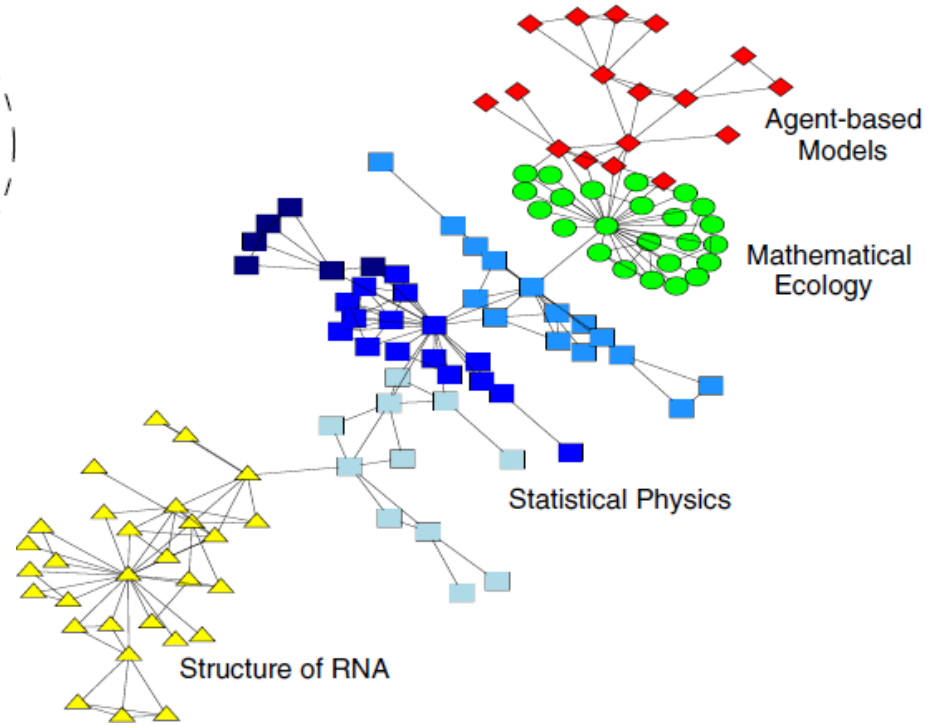
- The analysis of networks and network dynamics generates large datasets collected in very different settings;
  Social and economic networks,
  Information networks,
  The internet and the world wide web,
  Immunization and epidemiology networks,
  Molecular and gene regulatory networks,
  Sensor networks,
  Power grids amd transportation networks, etc.
- *Relational structure:* connectedness, size of the giant component, component sizes, degree and clique distributions, node or edge parameters, clustering coefficients, path length, diameter, betweenness and closeness centralities, **communities.**
- *Network function:* diffusion of diseases, spread of news and information, voting trends, imitation and social influence, crowd behavior, failure propagation, **synchronization and (or) dynamical correlations.**

Data on networks

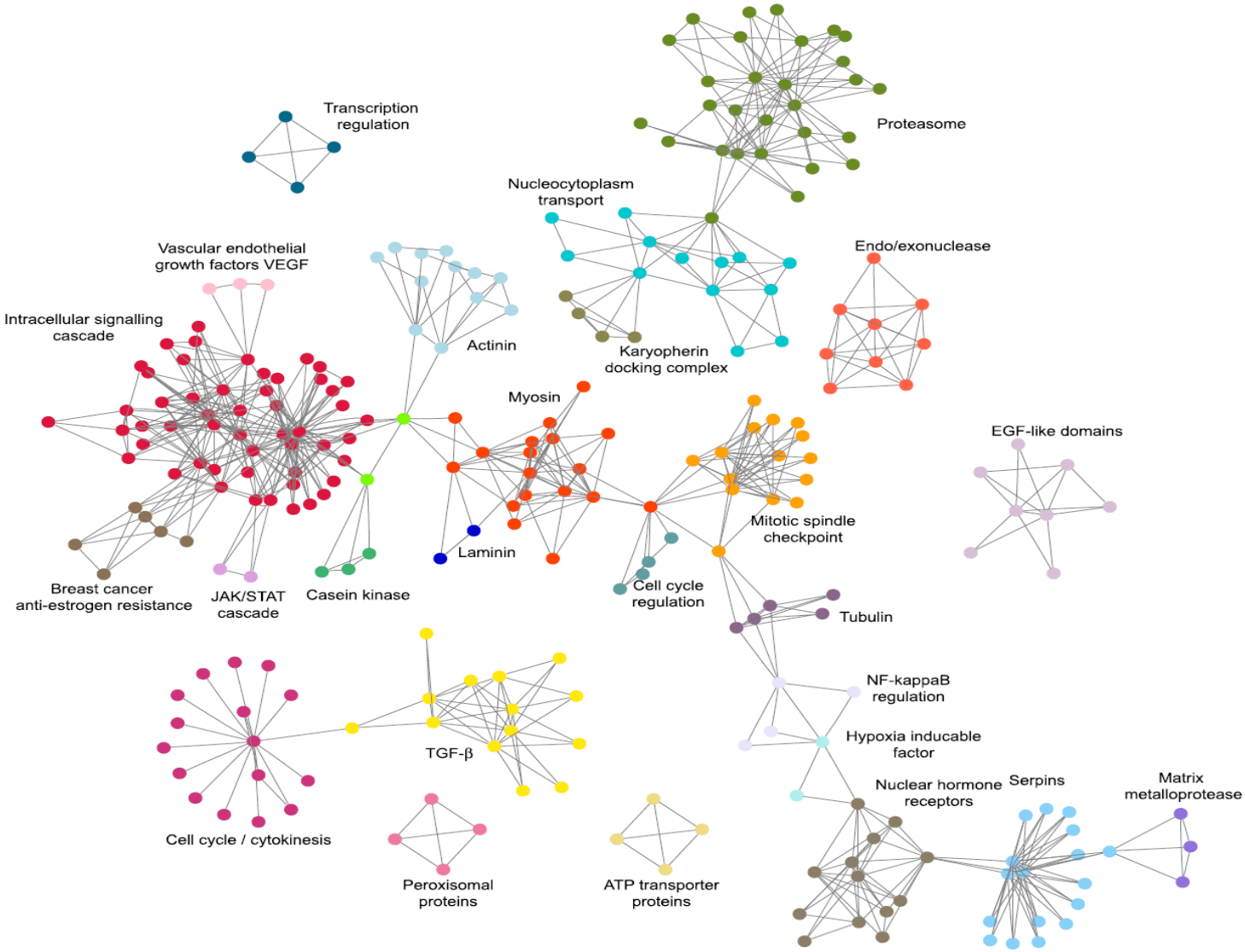
- *Inference and learning:* Bayesian networks, Markov random fields, Markov-Gibbs equivalence, belief propagation
- *Models:* deterministic complete or regular graphs, random Erdős-Rényi, Poisson graphs, small world, scale free networks
- *Efficient data representations:* **spectral graph theory** and the graph Laplacian to derive low-dimensional representations by projecting the data on a low-dimensional subspace generated by a small subset of the Laplacian eigenbasis. Discrete approximations to other continuous operators
- *Transforms for data indexed on graphs*: regression algorithms, **signal processing,** wavelet decompositions, **filters on graphs**, de-noising and compression, **Fourier transform of a signal on a graph.**

Networks and communities

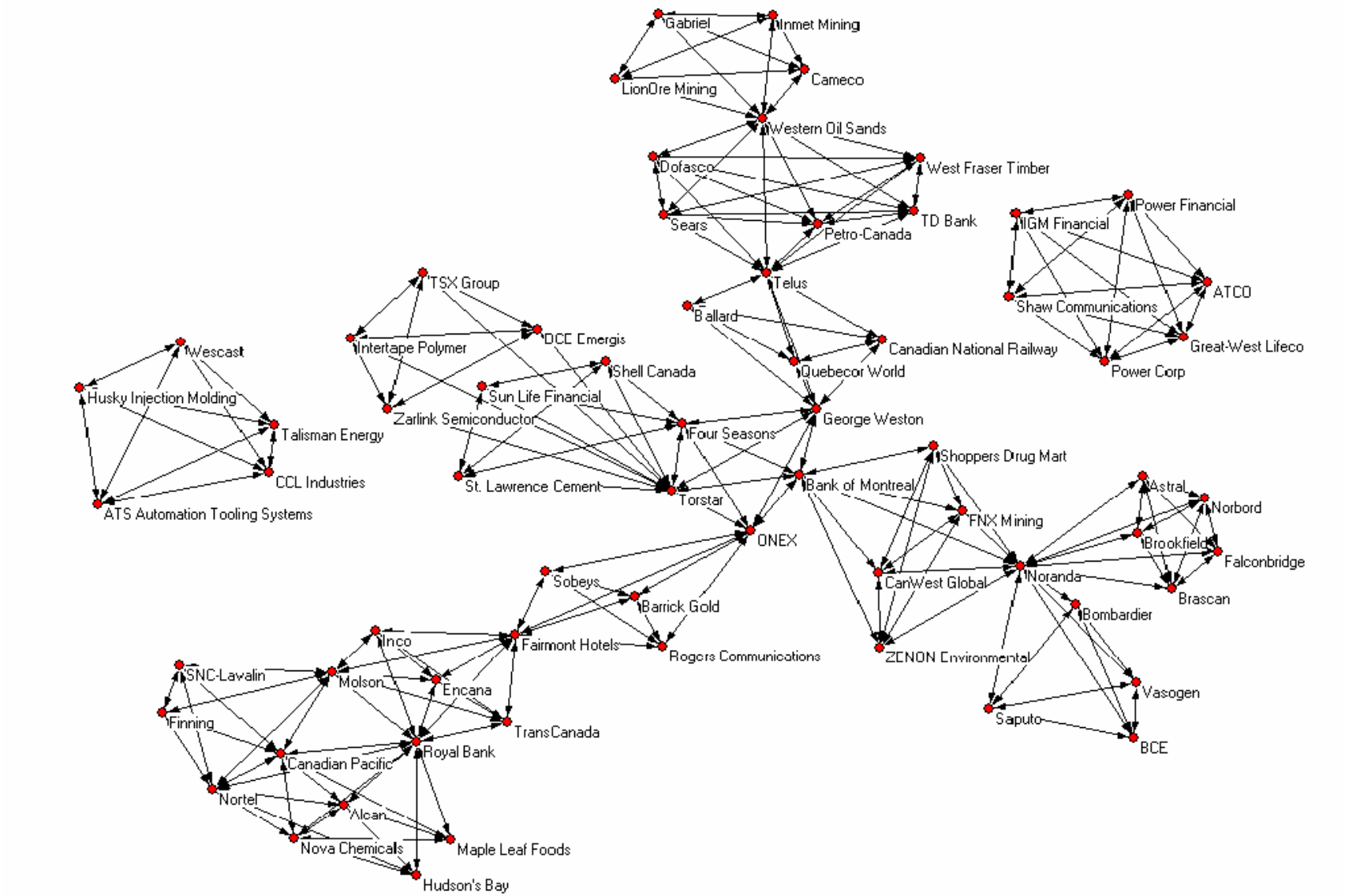




Networks and communities



Networks and communities



From coordinates to distances

- The Euclidean distance

$$d_{ij}^{E} = \sqrt{\sum_{k=1}^{N} \left| x_k\left(i\right) - x_k\left(j\right) \right|^2}$$

- The taxi metric (or Manhattan distance)

$$d_{ij}^{M} = \sum_{k=1}^{N} \left| x_k\left(i\right) - x_k\left(j\right) \right|$$

- L_∞ norm

$$d_{ij}^{\infty} = \max_{k=1,\ldots,N} \left| x_k\left(i\right) - x_k\left(j\right) \right|$$

- Cosine similarity

$$d_{ij}^{\cos^{-1}} = \arccos \frac{\sum_k x_k\left(i\right) x_k\left(j\right)}{\sqrt{\sum_k x_k\left(i\right)^2} \sqrt{\sum_k x_k\left(j\right)^2}}$$

Graphs

- Consider a dataset with N elements, with a relational structure $\Longrightarrow$ adjacency matrix.
- Several ways to to construct a graph from a similarity (or distance) table:
  - *The $\varepsilon-$neighborhood graph*
  (Uniform connections if distance $< \varepsilon$)
  - *$k-$nearest neighbor graphs*
  (Connect vertex v_i to vertex v_j if v_j is among the k nearest neighbors of v_i. Mutual or non-mutual)
  - *The fully connected graph*

From distances to the adjacency matrix

- Computing the adjacency matrix from a distance d_{ij} or a similarity s_{ij} table. $(d_{ij} = 1 - s_{ij})$

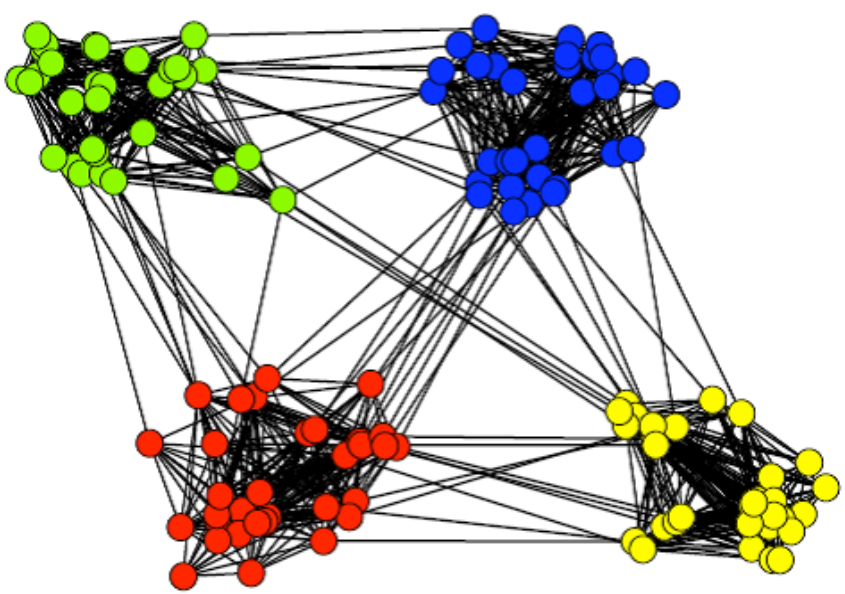
$$\mathbf{A}^{\#}_{ij} = \frac{d_{\min}}{d_{ij}}$$

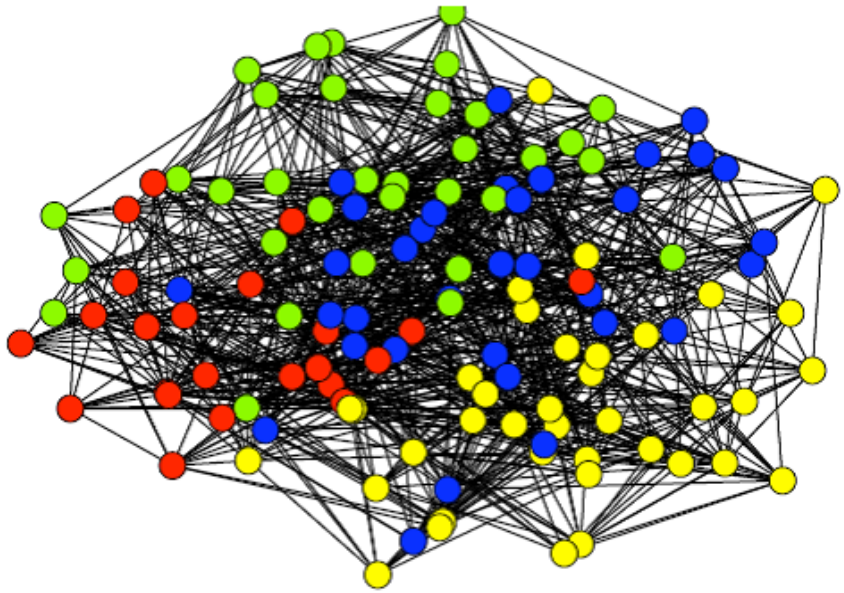
$$\mathbf{A}^{(\beta)}_{ij} = \exp\left(-\beta\left(d_{ij} - d_{\min}\right)\right)$$

- Represent this information by a graph $G = (\mathcal{V}, \mathbf{A})$, where
  $\mathcal{V} = \{v_0, \ldots, v_{N-1}\}$ is the set of vertices and
  $\mathbf{A}$ is the weighted adjacency matrix of the graph.
  Each weight $\mathbf{A}_{n,m}$ is the weight of a *directed* edge from v_m to v_n which can take arbitrary real or complex values.
  $\mathcal{N}_n = \{m \mid \mathbf{A}_{n,m} \neq 0\}$ is the *neighborhood* of v_n
- A *graph signal* is a map from the set $\mathcal{V}$ of vertices into the set of complex numbers $\mathbb{C}$, writen as $\mathbf{s} = \begin{pmatrix} s_0 & s_1 & \ldots & s_{N-1} \end{pmatrix}^T$ each element s_n being *indexed* by vertex v_n

Communities

Communities: Groups of vertices that are densely connected among themselves while being sparsely connected to the rest of the network
Applications: Classification, drug interactions, disease spreading, CPU optimization, modular structure, cycles, functional groupings in metabolic networks, customer profiles, etc.
The problem: *Partition network to maximize (minimize) internal (external) connections* and *Determine if community structure is actually present*





A dictionary of community notions

- *Intra-cluster density*

$$\delta_{int}(C) = \frac{\text{no. of internal edges of } C}{n_c(n_c - 1)}$$

- *Inter-cluster density*

$$\delta_{int}(C) = \frac{\text{no. of inter-cluster edges of } C}{n_c(n - n_c)}$$

- *Clique* = subgraph where all vertices are adjacent to each other.
- *n-clique* = maximal subgraph where the distance of each pair of vertices is not larger than n ($n = 1$ is a clique)
- *n-clan* = n-clique with diameter not larger than n
- *n-club* = maximal subgraph of diameter n

A dictionary of community notions

- *k-plex* = maximal subgraph with each vertex adjacent to all others except at most k of them
- *k-core* = maximal subgraph with each vertex adjacent to at least k other vertices
- *Strong community* = subgraph where the internal degree of each vertex is greater than the external degree
- *Edge connectivity* of a pair of vertices = minimal no.of edges that need to be removed to disconnect the pair.
- *Lambda set* = subgraph where any pair of vertices has a larger edge connectivity than any pair formed by one vertex inside and other outside

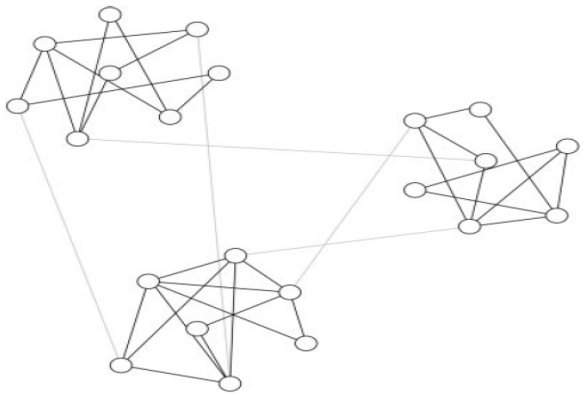
- Fitness measures for communities. *Modularity, Cut size, RatioCut, Ncut, etc.*

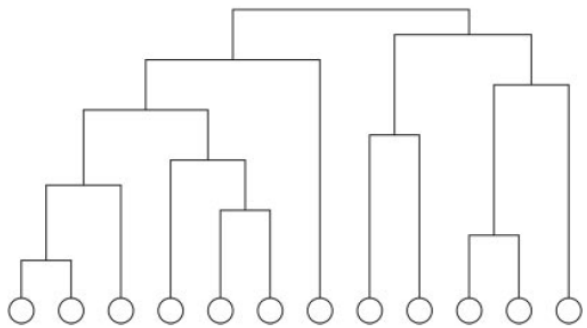
Community detection

Community detection: Partition of the data into disjoint groups such that data points belonging to the same cluster are similar, while those belonging to different clusters are dissimilar. (*Conditioned or non-conditioned partition*)

Methods:

- *Agglomerative (hierarchical clustering)*
  Start from an empty graph (N vertices, 0 edges), add the edges in decreasing order of their strength $\Longrightarrow$ dendogram





Community detection

- **Divisive (centrality based)**
- *Cycle based (edge clustering)*
- **K-means**
- **Spectral**
- **Modularity based methods**
- *Conductance-based methods*
  $\phi(S) = \frac{c_S}{\min(vol(S), vol(V \setminus S))}$ $\quad c_S = |\{(u, v) : u \in S, v \notin S\}|$
- *Message passing. Belief propagation*
- **Graph signal regularization**

A divisive algorithm (Girvan-Newman). Centrality based

- *Edge betweenness (geodesic)* = number of shortest paths between all vertex pairs that run along the edge
- *Random walk betweenness* = frequency of crossings by a random walker
- *Current flow betweenness* = average value of the current carried by the edge if a voltage difference is applied between the vertices

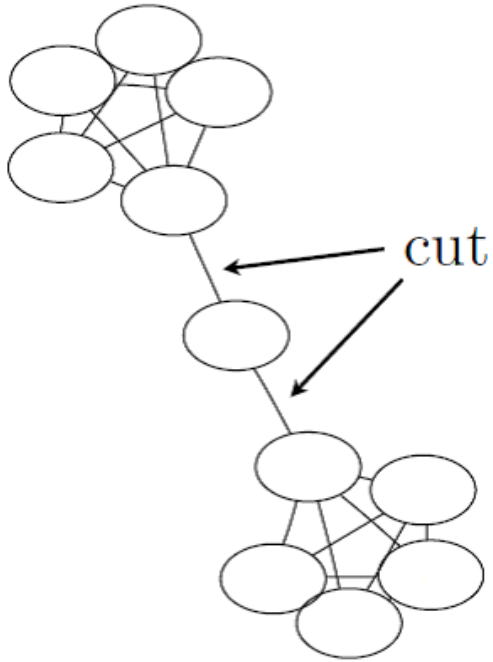
The algorithm:

1 - Compute the (betweenness) centrality of all edges

2 - Remove the edge with largest centrality

3 - Recalculate the centralities

4 - Repeat from 2



The K-means method

- **The K-means method**
  Looks for K clusters in n data points $(x_1, x_2, ..., x_n)$ by minimizing J_K

$$J_K = \sum_{k=1}^{K} \sum_{i \in C_k} |x_i - c_k|^2$$

$$= \sum_{k=1}^{K} \sum_{i,j \in C_k} \frac{|x_i - x_j|^2}{2n_k} \qquad c_k = \sum_{i \in C_k} \frac{x_i}{n_k}$$

  where c_k is the centroid of the cluster C_k.

The K-means method

The simplest algorithm:

1 - Start from a set of centroids

2 - Each vertex is assigned to the closest centroid

3 - Find the centers of mass of each cluster, which become the new centroids

4 - Make a new classification of the vertices as in 2

5 - Repeat from 3

6 - Stop when the centroid positions are stable

- In general, the result depends on the initial position of the centroids. Repeat with different initial sets. Choose the one that yields the least intra-cluster distance
- The number of clusters must be specified. The method does not derive it.

The K-means method

Let $K = 2$, $y_i = x_i - \langle x \rangle$ be the centered coordinates and $(D)_{ij} = d_{ij} = |x_i - x_j|^2$ the distance matrix.
J_2 may be rewriten

$$J_2 = \sum_{k=1}^{2} \sum_{i,j \in C_k} \frac{|x_i - x_j|^2}{2n_k} = n \langle y^2 \rangle - \frac{1}{2} J_D$$

$$J_D = \frac{n_1 n_2}{n} \left\{ 2 \frac{d(C_1, C_2)}{n_1 n_2} - \frac{d(C_1, C_1)}{n_1^2} - \frac{d(C_2, C_2)}{n_2^2} \right\}$$

$$d(C_k, C_l) = \sum_{i \in C_k} \sum_{j \in C_l} |x_i - x_j|^2$$

Hence $\min J_2$ is $\max J_D$. Defining a cluster indicator vector by

$$q(i) = \begin{cases} \sqrt{\frac{n_2}{n_1 n}} \text{ if } i \in C_1 \\ -\sqrt{\frac{n_1}{n_2 n}} \text{ if } i \in C_2 \end{cases}$$

one sees that

$$J_D = -q^T D q$$

The K-means method

By relaxing the condition that the q vector can only have the two discrete values, the solution of $\max J_D$ is obtained by finding the lowest (largest negative) eigenvalue of

$$Dq = \lambda q$$

An equivalent problem is obtained by centering the matrix of distances D

$$\left(\overset{\curlywedge}{D}\right)_{ij} = d_{ij} - \frac{1}{n}\sum_j d_{ij} - \frac{1}{n}\sum_i d_{ij} + \frac{1}{n^2}\sum_{i,j} d_{ij}$$

$$J_D = -q^T Dq = -q^T \overset{\curlywedge}{D} q$$

Then the solution is obtained by the lowest eigenvalue of $\overset{\curlywedge}{D}q = \lambda q$ (all the eigenvectors here have zero sum). Furthermore

$$\overset{\curlywedge}{d}_{ij} = -2\left(x_i - \langle x \rangle\right)^T \left(x_j - \langle x \rangle\right)$$

$$\overset{\curlywedge}{D} = -2Y^T Y$$

Principal component analysis (PCA)

$$X = (x_1, x_2, ..., x_n) \qquad x_i \in \mathbb{R}^m$$

- Covariance

$$S = XX^T \qquad m \times m \text{ matrix}$$

$$Su_k = \lambda u_k$$

$u_k \in \mathbb{R}^m$ are the principal directions.

- Gram matrix ($n \times n$ matrix)

$$X^T X v_k = \lambda_k v_k \qquad v_k \in \mathbb{R}^n$$

- Singular value decomposition

$$X = \sum_{k=1}^{m} \lambda_k u_k v_k^T \qquad m \times n \text{ matrix}$$

Therefore in the $K = 2$ case, K-means is obtained by PCA of the centered coordinates

K-means, larger K

Let us assume the clusters are ordered in blocks (to be found of course). Then

$$\begin{aligned} J_K &= \sum_i x_i^2 - \sum_k \frac{1}{n_k} \sum_{i,j \in C_k} x_i^T x_j \\ &= Tr\left(X^T X\right) - Tr\left(H_K^T X^T X H_K\right) \end{aligned}$$

where

$$H_K = (h_1, h_2, ..., h_K)$$

the $h's$ being the indicator vectors

$$h_k = \left(0...0, \overbrace{1, ...1}^{n_k}, 0, ..., 0\right)^T / n_k^{1/2}$$

H_K is just the projector on the blocks. The $h_k's$ are not linearly independent $\sum_{k=1}^K n_k^{1/2} h_k = \mathbf{e}$.

K-means, larger K

Make a linear transformation on the $h'_k s$ in such a way that the last vector is the identity

$$\begin{aligned} J_K &= Tr\left(X^T X\right) - \mathbf{e}^T X^T X \mathbf{e}/\mathbf{n} - Tr\left(Q_{K-1}^T X^T X Q_{K-1}\right) \\ &= Tr\left(Y^T Y\right) - Tr\left(Q_{K-1}^T Y^T Y Q_{K-1}\right) \end{aligned}$$

and the optimization of J_K is

$$\max Tr\left(Q_{K-1}^T Y^T Y Q_{K-1}\right)$$

with the constraints

$$\begin{aligned} Q_{K-1}^T Q_{K-1} &= I_{K-1} \\ q_k^T \mathbf{e} &= 0 \qquad\qquad k = 1, ..., K-1 \end{aligned}$$

Ignoring the last constraint, that is, allowing the $h'_k s$ to take continuous values, the transformed cluster indicators are the $K-1$ principal components, (details and further results: C. Ding and X. He; ICML '04 Proceedings of the 21st. Int. Conf. on Machine learning)

Partition vs. cover. Fuzzy K–means

- *Partition* of a graph is the decomposition into independent subgraphs.
- If some vertices are shared by different clusters one speaks of a *cover*.
- A cover algorithm: Fuzzy K-means

$$J_K = \sum_{i=1}^{N} \sum_{k=1}^{K} u_{ik} \left| x_i - c_k \right|^2$$

u_{ik} is the membership of element i in the cluster k and $\sum_k u_{ik} = 1$.
The algorithm may be similar to the one discussed before, with an initial choice for the membership functions.

Graph matrices

We have met the distance matrix D and the adjacency matrix $\mathbf{A}$. Other useful graph matrices are

- *The degree matrix* $\mathbf{G}$*:* a diagonal matrix listing the degree of the vertices
- *The Laplacian matrix:* $\mathbf{L} = \mathbf{G} - \mathbf{A}$
- *The symmetrically normalized Laplacian matrix:* $\mathbf{L}_{sym} = \mathbf{G}^{-\frac{1}{2}}\mathbf{L}\mathbf{G}^{-\frac{1}{2}}$
- *The random walk matrix:* $\mathbf{P} = \mathbf{G}^{-1}\mathbf{A}$
- *The random walk Laplacian:* $\mathbf{L}_{RW} = \mathbf{G}^{-1}\mathbf{L}$
- *The lazy random walk matrix:* $\mathbf{P}' = \left(\mathbf{I} + \mathbf{G}^{-1}\mathbf{A}\right)/2$
- *The incidence matrix* ∇*:* is the $m \times N$ matrix (m=no. edges, N=no. of vertices) given by

$$\nabla_{e,v} = \begin{cases} 1 & \text{if} \quad e = (v, w) \quad \text{and } v < w \\ -1 & \text{if} \quad e = (v, w) \quad \text{and } v > w \\ 0 & \quad \text{otherwise} \end{cases}$$

Graph matrices

- *The edge adjacency matrix:* is a $m \times m$ matrix determined by the adjacencies of edges

$$ {}^{e}\mathbf{A}_{i,j} = \begin{cases} 1 & \text{if edges i and j are adjacent} \\ 0 & \text{otherwise} \end{cases} $$

Some properties

$$ h^T \mathbf{L} h = \frac{1}{2} \sum_{i,j}^{n} A_{ij} \left(h_i - h_j\right)^2 $$

- **L** is symmetric and positive semi-definite
- The smallest eigenvalue of **L** is 0, the corresponding eigenvector the constant vector
- **L** has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$
- For an undirected graph with non-negative weigths, the multiplicity of the zero eigenvalue of **L** equals the number of connected components in the graph.

Graph matrices

$$h^T \mathbf{L}_{sym} h = \frac{1}{2} \sum_{i,j}^{n} A_{ij} \left(\frac{h_i}{\sqrt{d_i}} - \frac{h_j}{\sqrt{d_j}} \right)^2$$

d_i is the degree of vertex i

- λ is an eigenvalue of $\mathbf{L}_{RW}$ with eigenvector h if and only if λ is an eigenvalue of $\mathbf{L}_{sym}$ with eigenvector $h' = \mathbf{G}^{1/2} h$
- λ is an eigenvalue of $\mathbf{L}_{RW}$ with eigenvector h if and only if λ and h solve the generalized eigenproblem $\mathbf{L}h = \lambda \mathbf{G}h$
- 0 is an eigenvalue of $\mathbf{L}_{RW}$ with constant eigenvector . 0 is an eigenvalue of $\mathbf{L}_{sym}$ with eigenvector $\mathbf{G}^{1/2}1$
- $\mathbf{L}_{sym}$ and $\mathbf{L}_{RW}$ are positive semi-definite and have n non-negative real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$

Community detection: The graph cut

Another way to define communities (clusters) is to minimize *the cut*.
Given a partition $(C_1, ..., C_K)$ define

$$Cut\left(C_1, ..., C_K\right) = \frac{1}{2}\sum_{k=1}^{K} W\left(C_k, \overline{C_k}\right)$$

with $W\left(C_k, \overline{C_k}\right) = \sum_{i \in C_k, j \in \overline{C_k}} A_{ij}$
Other alternatives, to avoid partition into too small clusters, is to minimize

$$RatioCut\left(C_1, ..., C_K\right) = \frac{1}{2}\sum_{k=1}^{K} \frac{W\left(C_k, \overline{C_k}\right)}{|C_k|}$$

$$NCut\left(C_1, ..., C_K\right) = \frac{1}{2}\sum_{k=1}^{K} \frac{W\left(C_k, \overline{C_k}\right)}{vol\left(C_k\right)}$$

$vol\left(C_k\right)$ is the sum of the degrees of the vertices in C_k

Ratiocut and the Laplacian matrix

Laplacian matrix $\mathbf{L} = \mathbf{G} - \mathbf{A}$

For every vector h in $\mathbb{R}^n$

$$\begin{aligned} h^T \mathbf{L} h &= \sum_{i=1}^{n} d_i h_i^2 - \sum_{i,j} A_{ij} h_i h_j \\ &= \frac{1}{2} \sum_{i,j}^{n} A_{ij} \left(h_i - h_j\right)^2 \end{aligned}$$

Define K indicator vectors for the clusters $\left(C_1, ..., C_K\right)$

$$h_k\left(i\right) = \begin{cases} \frac{1}{\sqrt{|C_k|}} \text{ if } v_i \in C_k \\ 0 \text{ otherwise} \end{cases}$$

$i = 1, ..., n$ and $k = 1, ..., K$. Let H be the matrix containing these indicator vectors as columns $\left(H^T H = I\right)$

$$h_k^T \mathbf{L} h_k = \left(H^T \mathbf{L} H\right)_{kk} = \frac{Cut\left(C_k, \overline{C_k}\right)}{|C_k|}$$

Ratiocut and the Laplacian matrix

Therefore

$$RatioCut\left(C_1,...,C_K\right) = Tr\left(H^T \mathbf{L} H\right)$$

and the problem of minimizing *RatioCut* is the same as minimizing $Tr\left(H^T \mathbf{L} H\right)$ subject to $H^T H = I$.

Relaxing the problem, that is, allowing the $h's$ to have arbitrary real values the solution is obtained by having the columns of H to be the first eigenvectors of $\mathbf{L}$ associated to the K lowest eigenvalues. In the end one has to re-convert the real valued matrix solution to a discrete partition.

Likewise if instead of the above indicator vectors one chooses

$$h_k\left(i\right) = \begin{cases} \frac{1}{\sqrt{vol(C_k)}} \text{ if } v_i \in C_k \\ 0 \text{ otherwise} \end{cases}$$

one obtains the minimization problem for the *NCut*

$$\min Tr\left(N^T \mathbf{G}^{-\frac{1}{2}} \mathbf{L} \mathbf{G}^{-\frac{1}{2}} N\right)$$

subject to $N^T N = I$.

Ratiocut and the Laplacian matrix

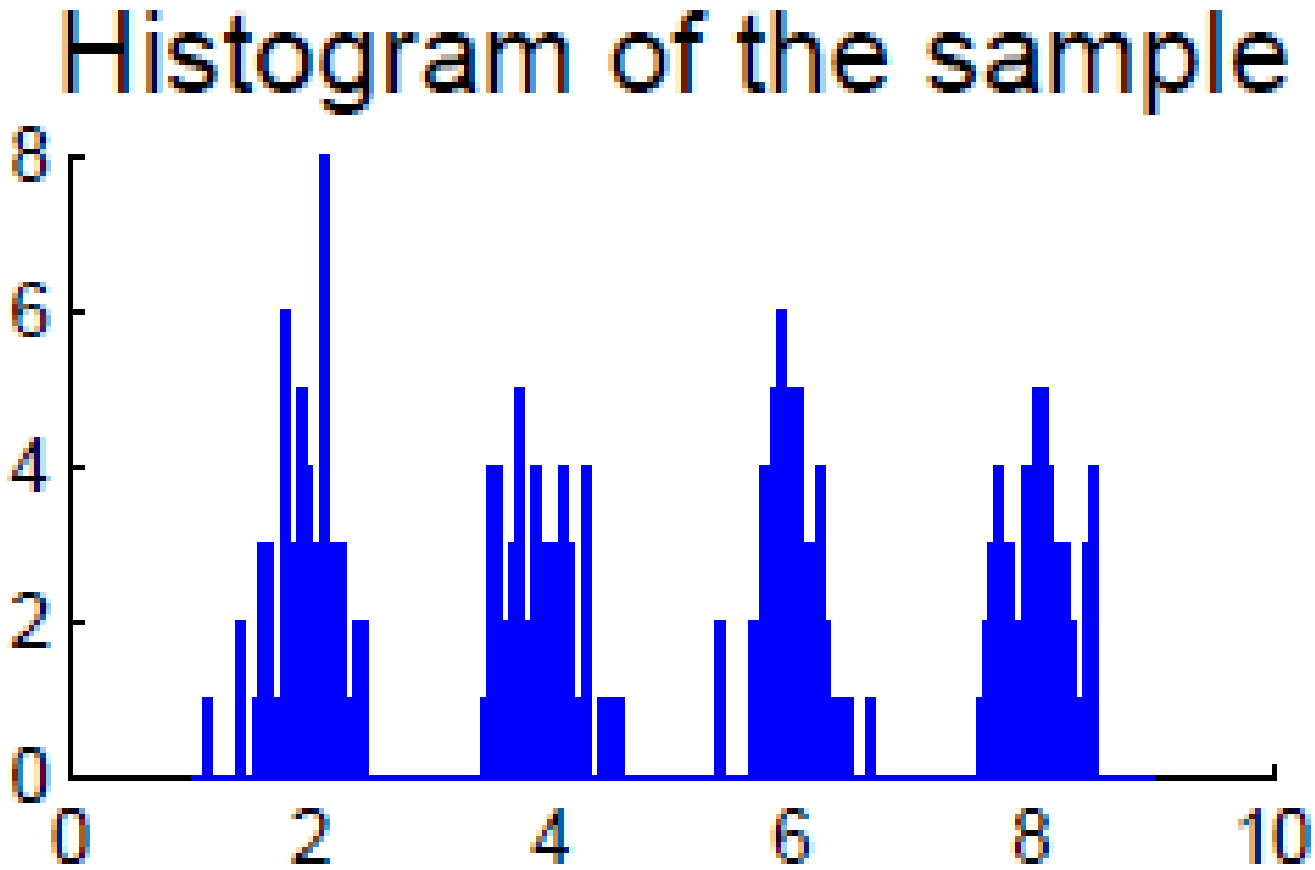
Conclusion: $\min(RatioCut)$ is associated to the lowest eigenvalues of the Laplacian matrix and $\min(NCut)$ to the symmetrically normalized Laplacian.
If

$$\left(\mathbf{G}^{-\frac{1}{2}}\mathbf{L}\mathbf{G}^{-\frac{1}{2}}\right)N = \lambda N$$

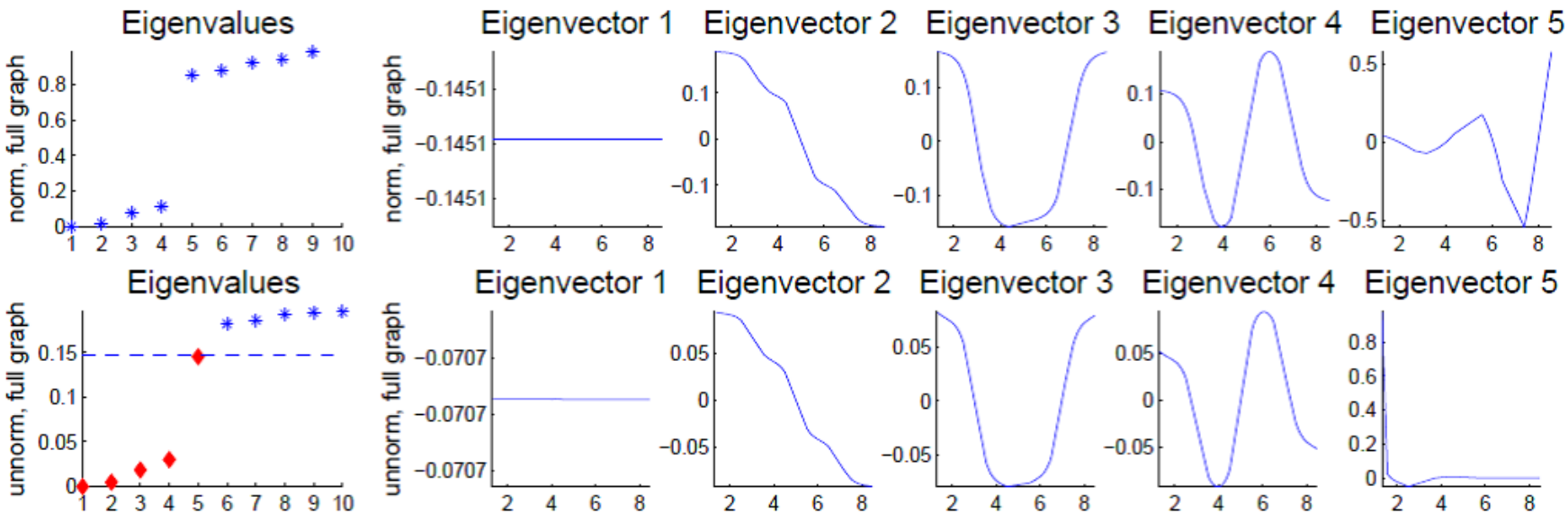
$$\mathbf{G}^{-1}\mathbf{L}\left(\mathbf{G}^{-\frac{1}{2}}N\right) = \lambda\left(\mathbf{G}^{-\frac{1}{2}}N\right)$$

thherefore an equivalent problem is to find the lowest eigenvalues of $\left(\mathbf{G}^{-1}\mathbf{L}\right) = \mathbf{L}_{RW}$

Ratiocut and the Laplacian matrix. An example



Ratiocut and the Laplacian matrix. An example



Random walks and spectral clustering

Image a random walker on the graph with transition probability

$$p_{ij} = \frac{A_{ij}}{d_i}$$

The transition matrix is

$$P = \mathbf{G}^{-1}\mathbf{A}$$

But

$$\mathbf{L}_{RW} = \mathbf{G}^{-1}\mathbf{L} = I - P$$

hence to find the smallest eigenvalues of $\mathbf{L}_{RW}$ is equivalent to find the largest eigenvalues of P. Therefore spectral clustering is equivalent to find a partition of the set of points in such a way that the random walk stays a long time inside each cluster and seldom jumps between clusters.
A related important notion is the notion of commute distance c_{ij} = the expected time for a random walk to travel from vertex v_i to v_j and back to v_i. Is different from the shortest path, because it decreases if there are many different short paths from v_i to v_j.

Random walks and spectral clustering

Define the generalized inverse of L

$$\mathbf{L} = U \Lambda U^T$$

$$\mathbf{L}' = U \Lambda' U^T$$

Λ' has diagonal entries $1/\lambda_i$ if $\lambda_i \neq 0$ and 0 if $\lambda_i = 0$. $\mathbf{L}'$ is called the generalized inverse of $\mathbf{L}$. Then

$$c_{ij} = vol\,(V)\,(e_i - e_j)\,\mathbf{L}'\,(e_i - e_j)^T$$

When are communities detectable?

In general cannot be answered. Studied in specific models.

The stochastic block model

N vertices. Each vertex has an hidden label $t_i \in \{1, 2, ..., q\}$ denoting the block to which it belongs. Labels chosen at random with probability n_a to belong to block a ($\sum_{a=1}^{q} n_a = 1$). Graph generated by putting an edge between vertices i and j with probability $p_{t_i t_j}$. For the sparse graph case $p_{ab} = \frac{C_{ab}}{N}$ with $C_{ab} = O(1)$. The case $n_a = \frac{1}{q}$, $c_{ab} = c_{out}$ if $a \neq b$ and $c_{ab} = c_{in}$ if $a = b$ with $c_{in} > c_{out}$ is called *planted partitioning*.

It has been proven that in the $N \longrightarrow \infty$ (a large sparse graph) detectability of communities (blocks) requires

$$c_{in} - c_{out} > \sqrt{2\left(c_{in} + c_{out}\right)}$$

otherwise the randomness of the graph washes out the block structure in such a way that no algorithm is able to label the blocks better than by chance. Most algorithms not even reach this limit. An algorithm based on the edge adjacency matrix has recently been proposed to reach this limit.

Graph spectrum and robustness

Given the Laplacian matrix spectrum $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$

- **Algebraic connectivity** $= \lambda_2$
  - $= 0$ if and only if the graph is unconnected
  - $0 \leq \lambda_2 \leq k_v \leq k_e \leq \delta_{\min}$

  (k_v = vertex connectivity; k_e = edge connectivity; $\delta_{\min}$ = minimum degree of the vertices)
- ξ = Number of spanning trees (subgraphs with $n-1$ edges and no cycles)

$$\xi = \frac{1}{n} \prod_{i=2}^{n} \lambda_i$$

- **Effective resistance,** Let each edge have resistance $r_{ij} = 1.$ and the resistance between vertices a and b be R_{ab}. Then

$$R = \sum_{1 \leq a \leq b \leq n} R_{ab} = n \sum_{i=2}^{n} \frac{1}{\lambda_i}$$

small resistance $\Longrightarrow$ robustness

Modularity based methods

- Modularity (m =no. of edges)

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta \left(C_i, C_j \right)$$

  $\frac{k_i k_j}{2m}$ is the average value of a random matrix with the same degree distribution as **A**

- For directed graphs use $\frac{k_i^{out} k_j^{in}}{m}$
- Modularity maximaization may be obtained by:
  1 - Greedy techniques
  2 - Simulated annealing
  3 - Spectral methods

Modularity based methods

- *Greedy techniques:* Start from N clusters, each containing a single vertex. Act as if edges where not present, but count them for the calculation of Q.
  Then add links, one at a time, recompute the modularity, accept the choice if the modularity increases, etc.
  There are many variations for the operation of adding vertices in this aglomerative process.
- *Simulated annealing:* Start from an initial configuration of clusters. Change randomly a vertex from one cluster to another. Accept the move if Q increases. Otherwise only with probability $\exp(\beta \Delta Q)$ if ΔQ is negative.
- *A spectral method:*
  Modularity matrix

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

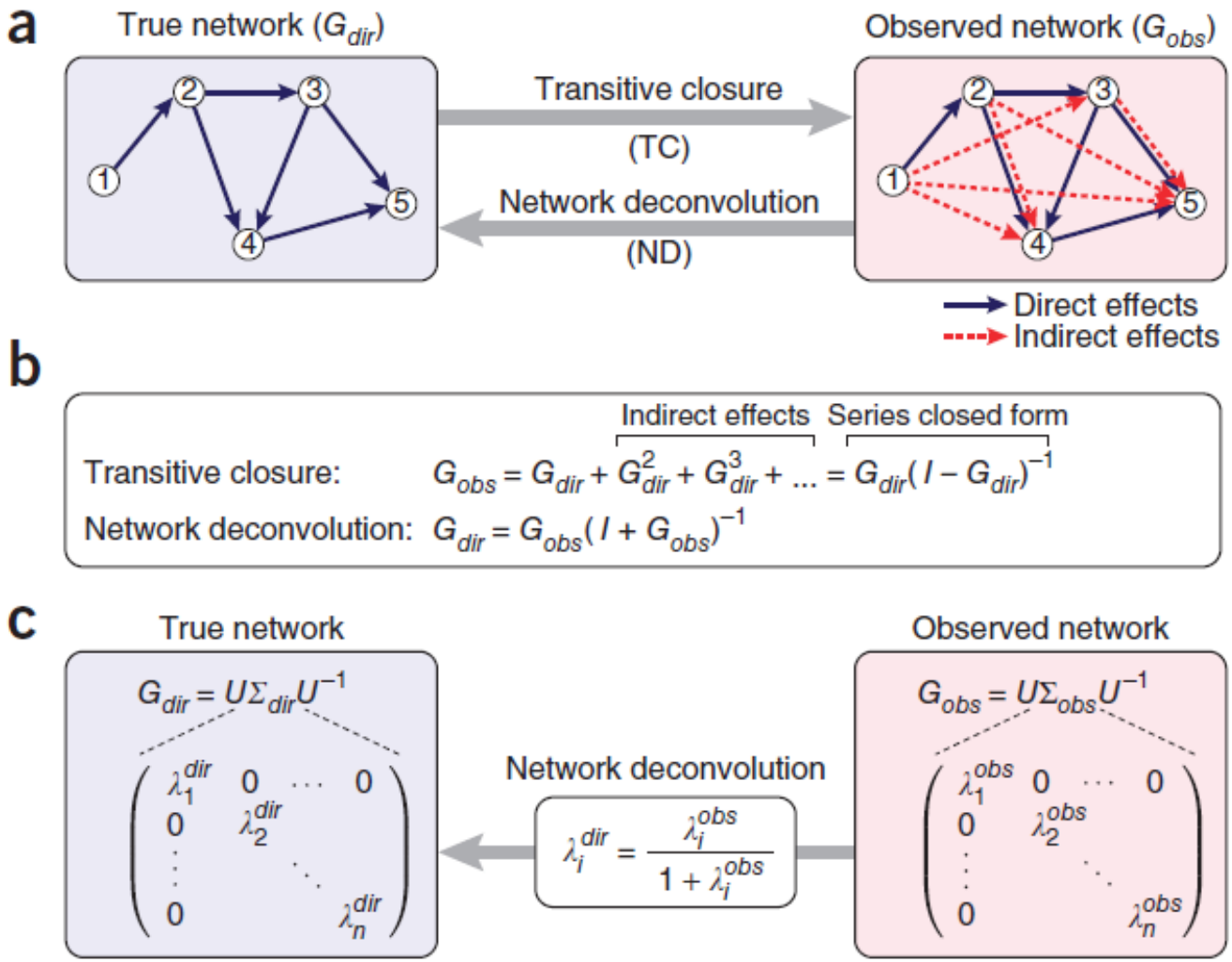
Modularity based: A spectral method

- 2 clusters: $s_i = +1$ for cluster a and $s_i = -1$ for cluster b

$$\begin{aligned} Q &= \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(C_i, C_j) = \frac{1}{4m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)(s_i s_j + 1) \\ &= \frac{1}{4m}\sum_{ij} B_{ij} s_i s_j = \frac{1}{4m} s^T B s \end{aligned}$$

- B has an eigenvalue zero with eigenvector $(1, 1, 1, ..., 1)$. If B has no positive eigenvalue, the maximum coincides with the trivial solution, the whole graph as a single cluster. Otherwise the largest positive eigenvalue provides the two clusters by the positive and negative signs of the components of the corresponding eigenvector.
- The process can now be applied to each cluster separately to refine the clustering, until no more positive eigenvalues appear.
- Alternatively if more than one positive eigenvalue appear, one may use the eigenvectors associated to the positive eigenvalues as in Laplacian-based method.

A warning concerning the construction of the adjacency matrix from correlation data. Network deconvolution



An exercise: Communities in the financial market

- From the daily returns

$$r(t) = \log S(t) - \log S(t-1)$$

$S(t)$ being the closing price at day t, compute a dynamical distance between the company stocks i and j by

$$d_{ij} = \sqrt{\sum_{t=1}^{T} \left(r_i(t) - r_j(t)\right)^2}$$

the sum being over the T trading days.

- Now compute the smallest non-zero d_{ij} ($d_{\min}$) and an adjacency matrix A by

$$A_{ij}^{(\beta)} = (1 - \delta_{ij}) \exp\left(-\beta \left(d_{ij} - d_{\min}\right)\right)$$

- Find the communities for each year in the SP500 from 2008 to 2012.
- For the data use, for example, DataStream or Reuters or http://label2.ist.utl.pt/Cotacoes

References

- F. Chung; *Spectral Graph Theory*, AMS, Providence, R. I., 1997.
- S. Fortunato; *Community detection in graphs*, Physics Reports 486 (2010) 75–174
- U. Von Luxburg; *A tutorial on spectral clustering*, Stat. Comput. 17 (2007) 395-416.
- C. Ding and X. He; *K-means Clustering via Principal Component Analysis*, ICML '04 Proceedings of the twenty-first international conference on Machine learning
- R. Nadakuditi and M. Newman; *Graph spectra and the detectability of community structure in networks*, Phys. Rev. Lett. 108 (2012) 188701.
- M. Newman; *Finding community structure in networks using the eigenvectors of matrices*, Phys. Rev. E 74 (2006) 036104.
- A. Decelle et al.; *Inference and phase transitions in the detection of modules in networks*, Phys. Rev. Lett. 107(2011) 065701.
- F. Krzakala et al; *Spectral redemption in clustering sparse networks*, PNAS 110 (2013) 20935–20940
- W. Ellens, R.E. Kooij; *Graph measures and network robustness*,