Lecture IV
Introduction to complex networks

Santo Fortunato

Plan of the course

I. **Networks: definitions, characteristics, basic concepts in graph theory**

II. Real world networks: basic properties

III. Models

IV. Community structure I

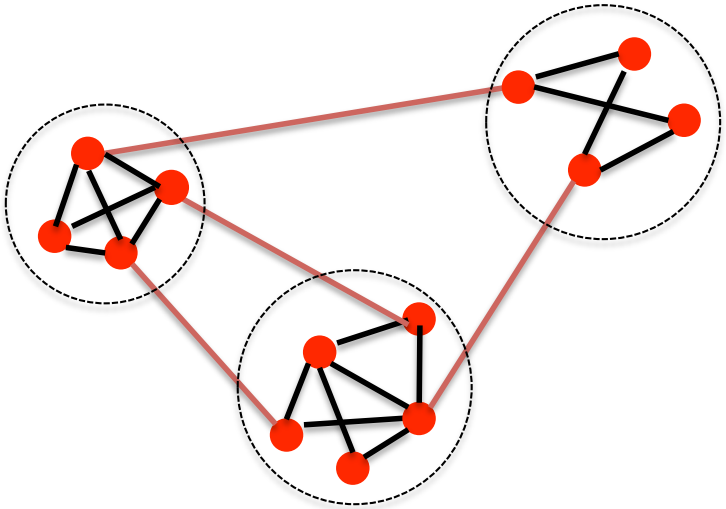
V. Community structure II

VI. Dynamic processes in networks

Outline

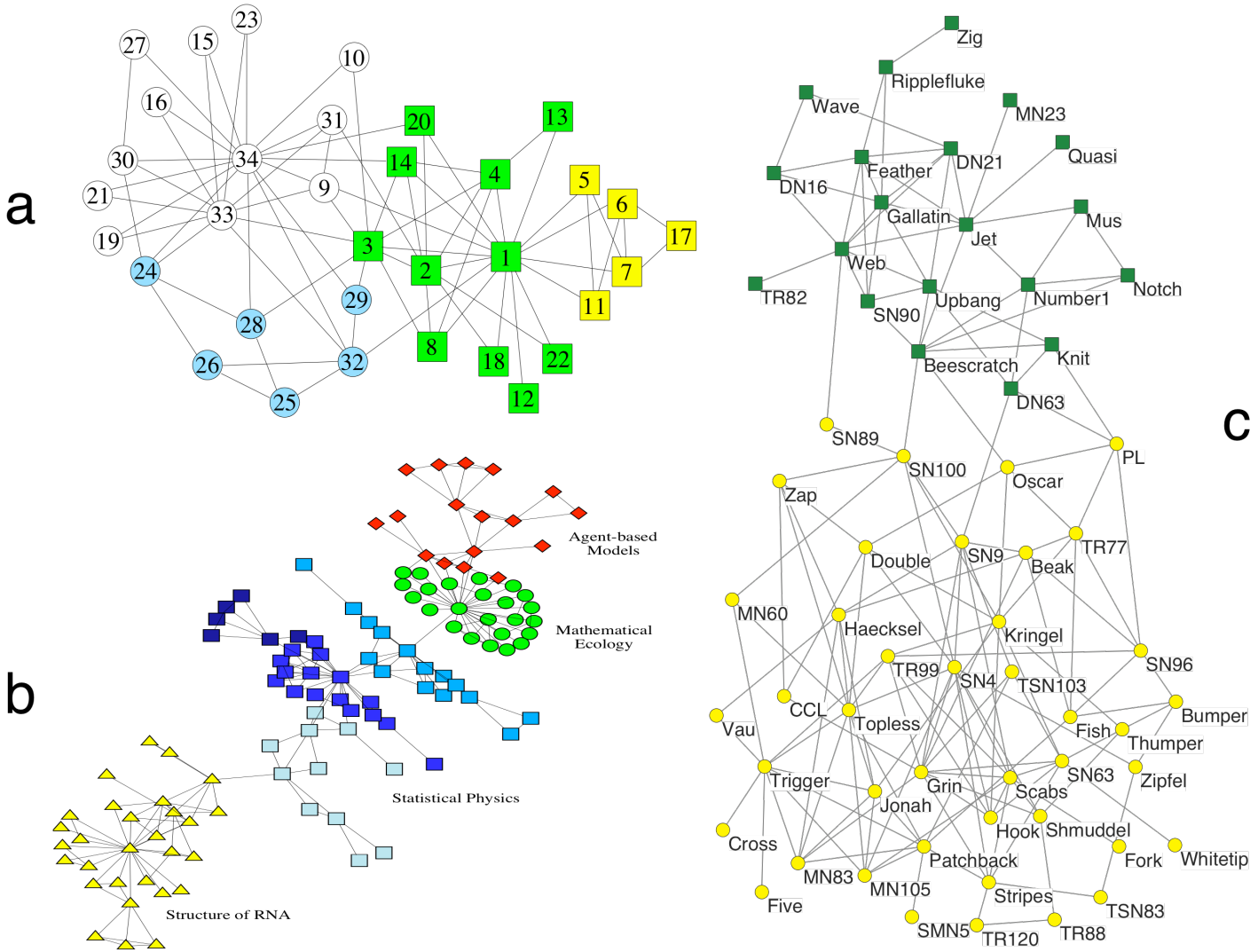
- Communities in real networks
- Elements of community detection:
  a) community definitions
  b) partitions
- Traditional clustering methods
  a) graph partitioning
  b) hierarchical clustering
  c) partitional clustering
  d) spectral clustering

Communities

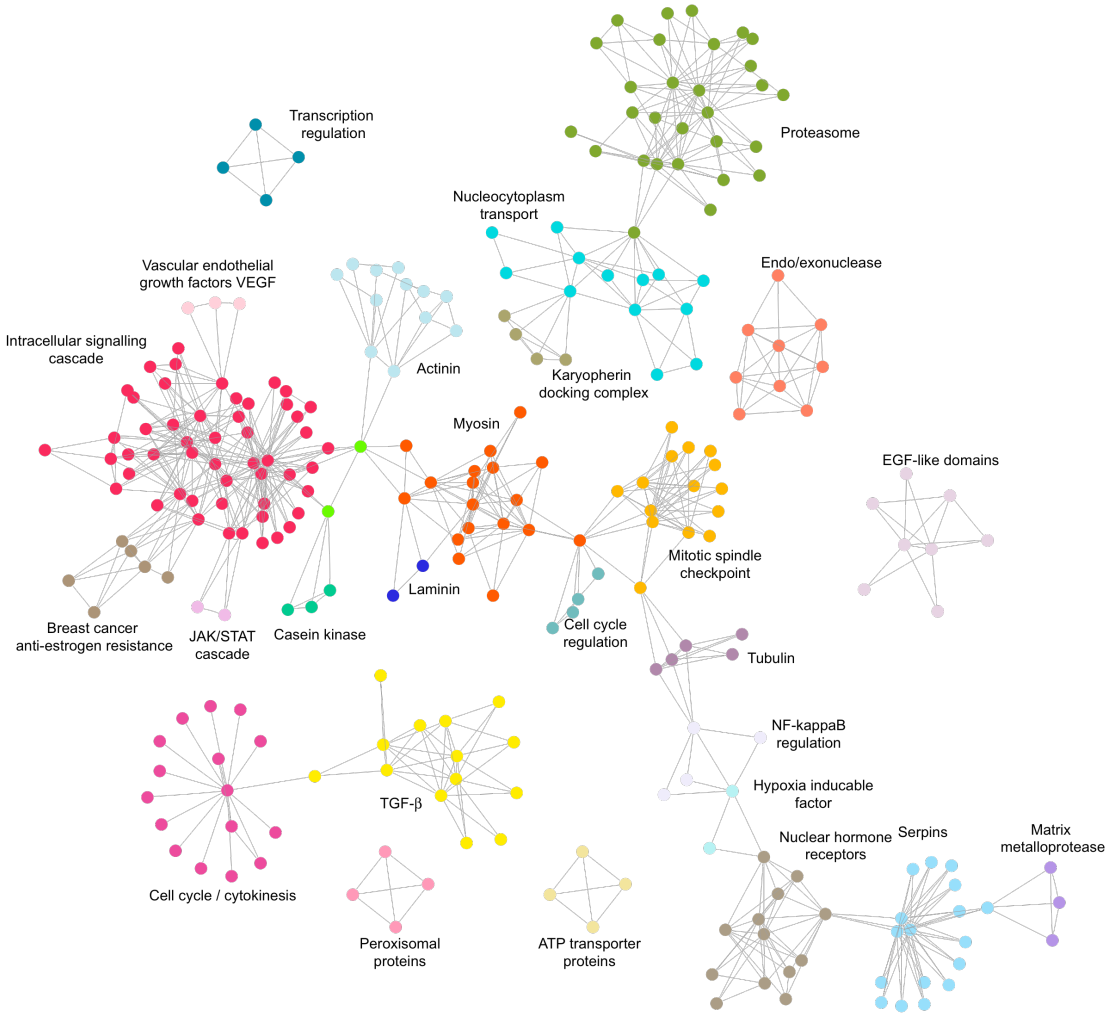


More links “inside” than “outside”

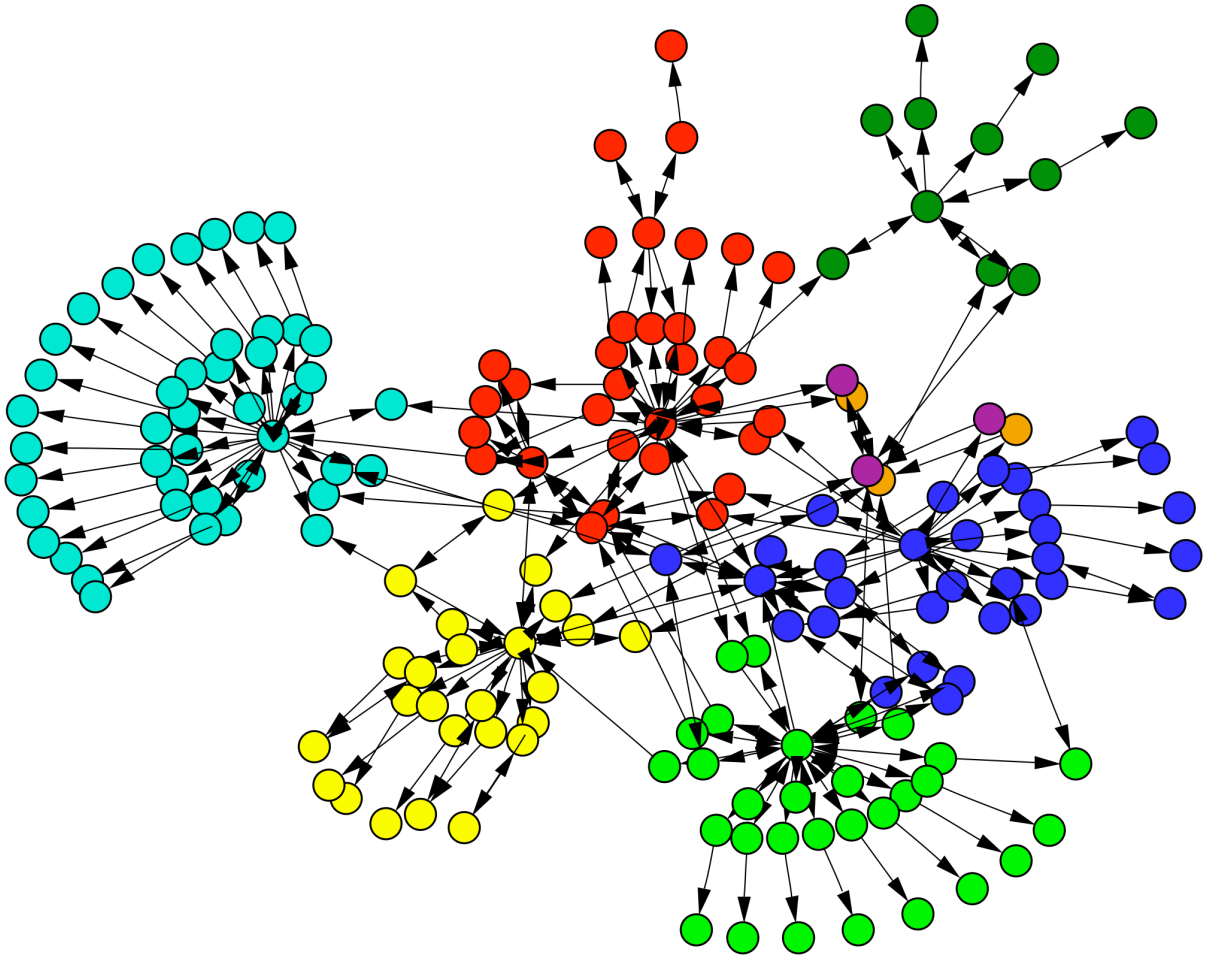
Communities in social networks



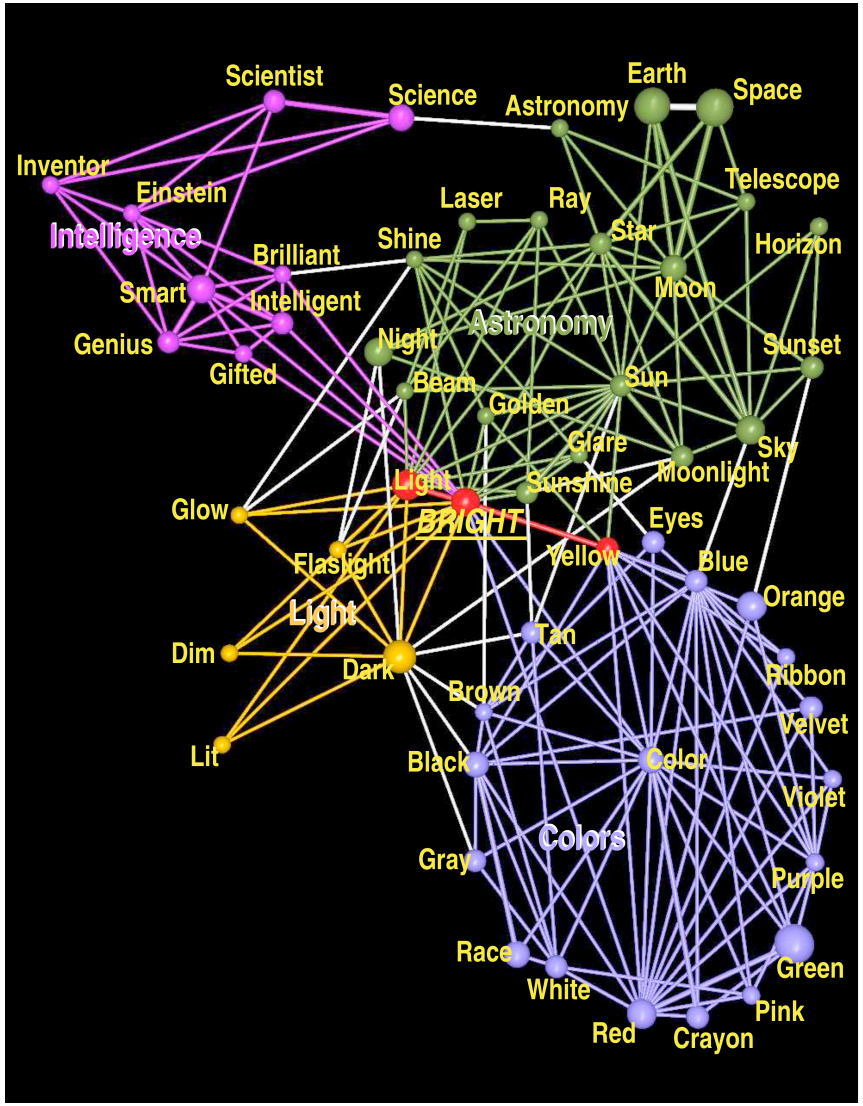
Communities in biological networks



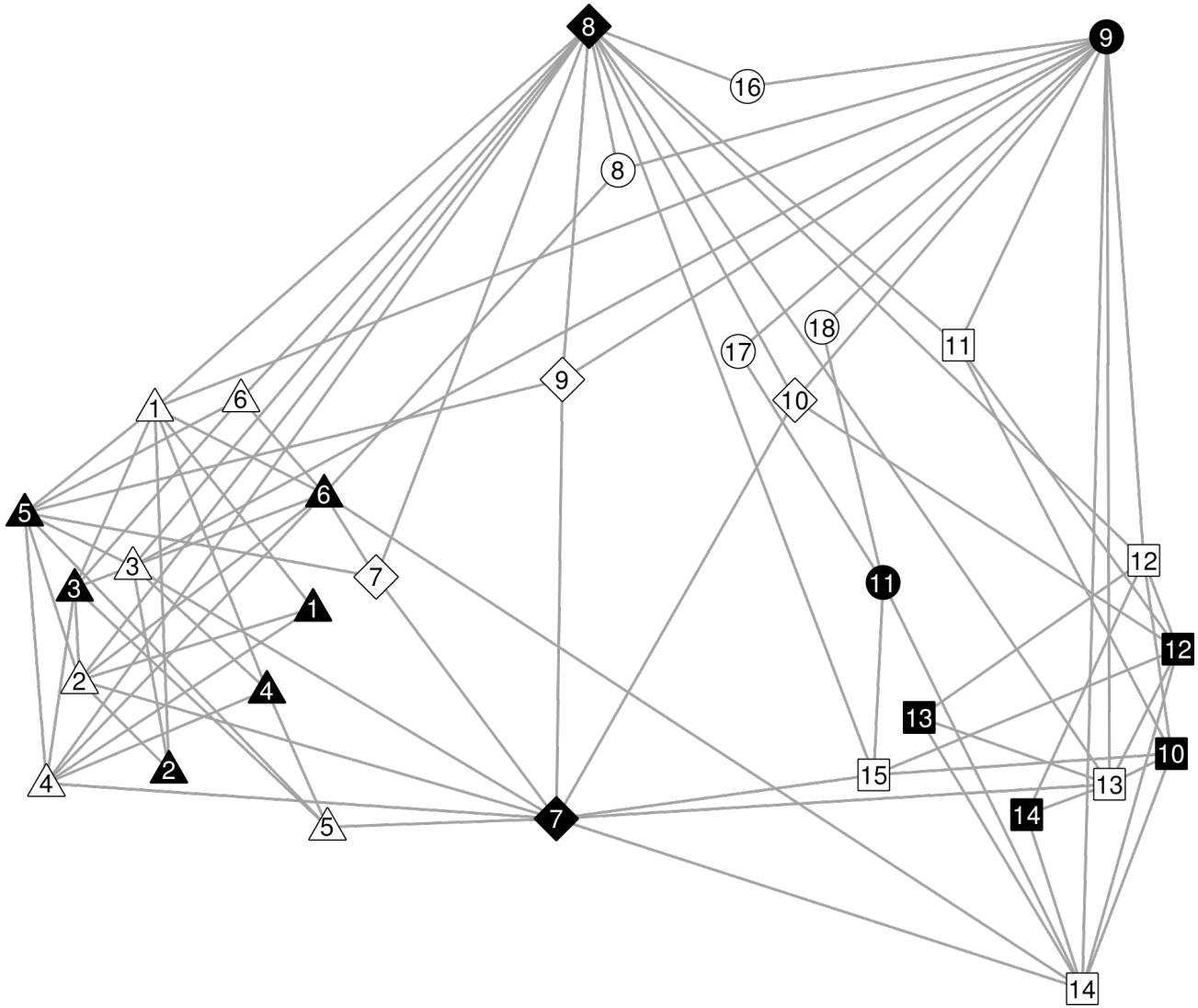
Communities in information networks



Networks with overlapping communities



Communities in bipartite networks



Elements of community detection

Warnings:

1) Null hypothesis: communities are inferred just from structural information, relationship with actual groups is unclear!
2) The number m of edges of the graph is of the order of the number of vertices n, $m \sim n$, otherwise problem becomes similar to data clustering!

Topics:

1) Computational complexity
2) Communities
3) Partitions

Computational complexity

Definition: the computational complexity of an algorithm is the amount of resources required by the algorithm to perform a task

Resources: number of computation steps and memory units

Notation: $O(n^{\alpha}m^{\beta})$, polynomial complexity (class P)

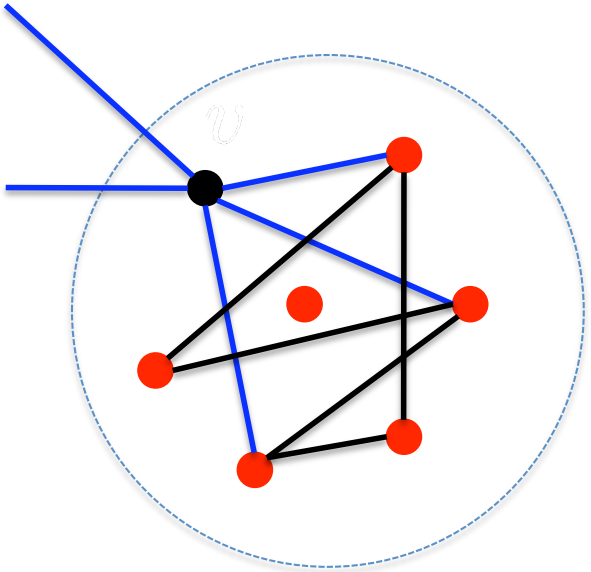
Class NP: problems whose solution can be verified in polynomial time

Class NP-hard: problems whose solution can be translated into a solution of any NP problem

Class NP-complete: problems which are NP-hard and in NP

Exact complexity often unknown: worst-case complexity!

Communities: basics



k_v^{int} internal degree, number of internal neighbors

$$k_{int}^{\mathcal{C}} = \sum_{v \in \mathcal{C}} k_v^{int}$$ internal degree of community $\mathcal{C}$

k_v^{ext} external degree, number of external neighbors

$$k_{ext}^{\mathcal{C}} = \sum_{v \in \mathcal{C}} k_v^{ext}$$ external degree of community $\mathcal{C}$

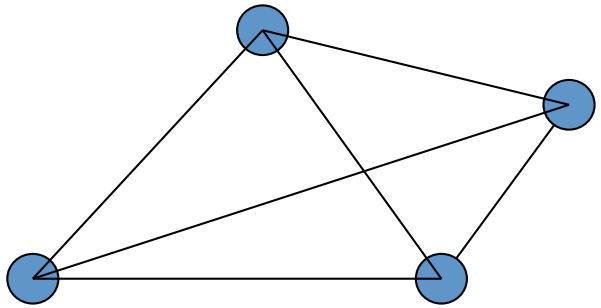
$$\delta_{int}(\mathcal{C}) = \frac{\#\text{ internal edges of } \mathcal{C}}{n_c(n_c-1)/2}$$ Intra-cluster edge density

$$\delta_{ext}(\mathcal{C}) = \frac{\#\text{ external edges of } \mathcal{C}}{n_c(n-n_c)}$$ Inter-cluster edge density

Communities: local definitions

Principle: look at the subgraph, forget the rest of the graph

Clique or complete graph

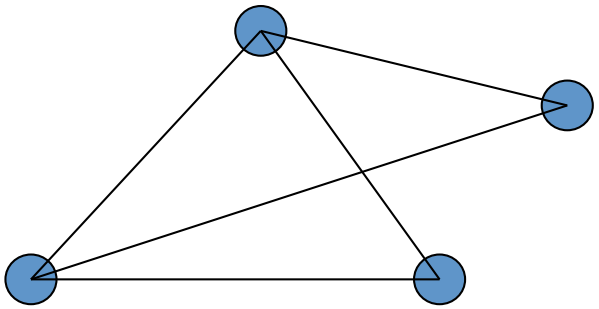


Problems

1) Condition is too strict!
2) All vertices are symmetric, whereas in real communities they usually have different roles
3) Cliques are hard to find: NP-complete problem. Bron-Kerbosch method has a complexity growing exponentially with the graph size (Bron & Kerbosch, 1973)

Communities: local definitions

n-clique: subgraph such that the distance between each pair of vertices does not exceed n (Luce, 1950)



Problems

1) Paths can go outside the community, so diameter of n-clique may be even bigger than n!
2) n-clique can be disconnected

Alternatives (Mokken, 1979)

1) n-clan = n-clique with diameter does not exceed n
2) n-club = maximal subgraph with diameter n

Communities: local definitions

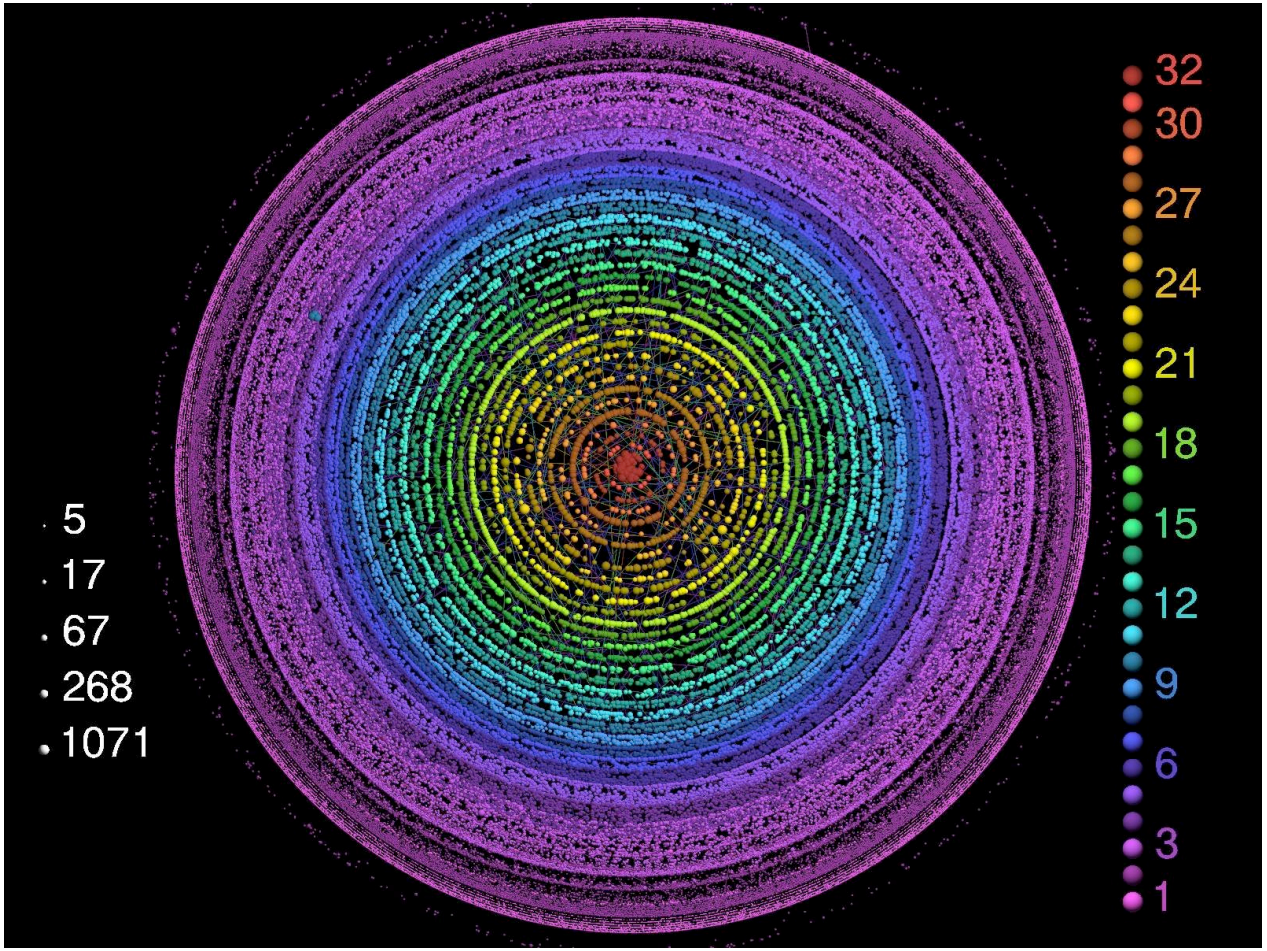
Principle: cohesion through vertex adjacency

k-plex: maximal subgraph such that each vertex is adjacent to all other vertices of the subgraph except at most k of them (Seidman & Foster, 1978)

k-core: maximal subgraph such that each vertex is adjacent to at least k other vertices of the subgraph (Seidman, 1983)

p-quasi complete subgraph: maximal subgraph such that the degree of each vertex is larger than $p(k-1)$ [$p \in [0,1]$, k is the order of the subgraph (Matsuda et al., 1999)]

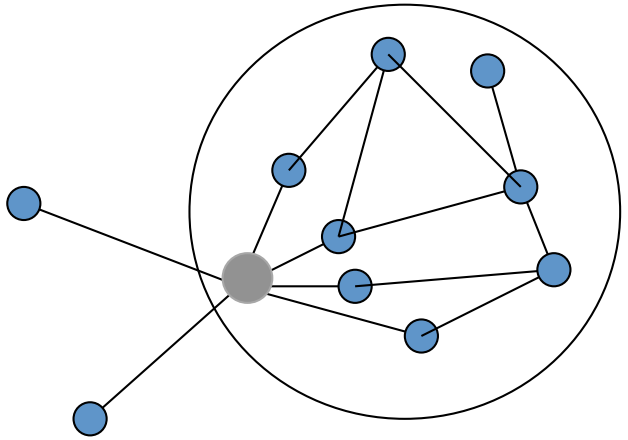
K-core decomposition (Batagely & Zaversnick, 2003)



Communities: local definitions

Principle: comparison between internal and external cohesion of a subgraph

LS-set or strong community: subgraph such that the internal degree of each vertex is greater than its external degree (Luccio & Sami, 1969)



Problem: condition too strong, unrealistic in practical cases!

Weak community: subgraph such that the internal degree of the subgraph is greater than its external degree (Radicchi et al., 2004)

Communities: local definitions

Variant of concepts of strong and weak community (Hu et al., 2008)

Strong community: subgraph such that the internal degree of each vertex is greater than its internal degree in any of the other communities of the partition

Weak community: subgraph such that the internal degree of the subgraph is greater than its external degree in each of the other communities

Link with planted ℓ-partition model (Condon & Karp, 1999)

Communities: local definitions

Communities can be defined through a fitness measure

Example 1: Community= subgraph C such that $\delta_{int}(C) > \xi$

NP-complete problem, as it coincides with the Clique Problem when $\xi = 1$

Example 2. Relative density $\rho(C) = \frac{\sum_{v \in C} k_v^{int}}{\sum_{v \in C} k_v}$

Community: subgraph C such that $\rho(C) > \xi$

NP-complete problem (Šima & Schaeffer, 2006)

Communities: global definitions

Principle: comparison between subgraph and the whole system

Definition often depending on choice of null models, i.e. randomized versions of the original graph

Most popular null model: random graph with the same expected degree sequence of the original graph

Same null model as in modularity of Newman and Girvan (Newman & Girvan, 2002)

Communities: definitions based on vertex similarity

Principle: communities are subgraphs of vertices “similar” to each other

Basic ingredient: measure of similarity between vertices

Similarity measures essential for methods like hierarchical, partitional and spectral clustering

Two classes of measures:

1) Graphs embedded in space
2) Graphs not embedded in space

Vertex similarity for graphs embedded in space

There is a distance between each pair of vertices, it could be used as dissimilarity measure

Pair of vertices: $A(a_1, a_2, \ldots, a_n), B(b_1, b_2, \ldots, b_n)$

Euclidean distance (L^2 norm)

$$d_{AB}^{E} = \sum_{k=1}^{n} \sqrt{(a_k - b_k)^2}$$

Manhattan distance (L^1 norm)

$$d_{AB}^{M} = \sum_{k=1}^{n} |a_k - b_k|$$

Cosine similarity

$$\rho_{AB} = \arccos \frac{\mathbf{a} \cdot \mathbf{b}}{\sqrt{\sum_{k=1}^{n} a_k^2} \sqrt{\sum_{k=1}^{n} b_k^2}}$$

Vertex similarity for graphs not embedded in space

Only information: adjacency matrix

Structural equivalence dissimilarity

$$d_{ij} = \sqrt{\sum_{k \neq i,j} (A_{ik} - A_{jk})^2}$$

Neighborhoods' overlap

$$\omega_{ij} = \frac{|\Gamma(i) \cap \Gamma(j)|}{|\Gamma(i) \cup \Gamma(j)|}$$

Pearson correlation coefficient

$$C_{ij} = \frac{\sum_k (A_{ik} - \mu_i)(A_{jk} - \mu_j)}{n \sigma_i \sigma_j}$$

$$\mu_i = (\sum_j A_{ij})/n, \sigma_i = \sqrt{\sum_j (A_{ij} - \mu_i)^2/n}$$

Vertex similarity for graphs not embedded in space

Path-dependent measures

1) Number of edge- (or vertex-) independent paths between a pair of vertices (related to max flow)
2) Total number of paths between pair of vertices, each weighted by factors like α^l or $1/l!$ (l = length of the path, $\alpha < 1$) [Estrada & Hatano, 2008]

Measures based on random walks

1) Commute-time: average number of steps needed for a random walker starting at any of two vertices to hit the other for the first time and come back to the first (Saerens et al.)
2) Average first passage time: average number of steps needed to hit for the first time the target vertex from the source vertex (White & Smyth, 2003; Zhou, 2003)
3) Escape probability: probability that the walker hits the target vertex before coming back to the source vertex (Palmer & Faloutsos, 2003; Tong et al., 2008)

Warning

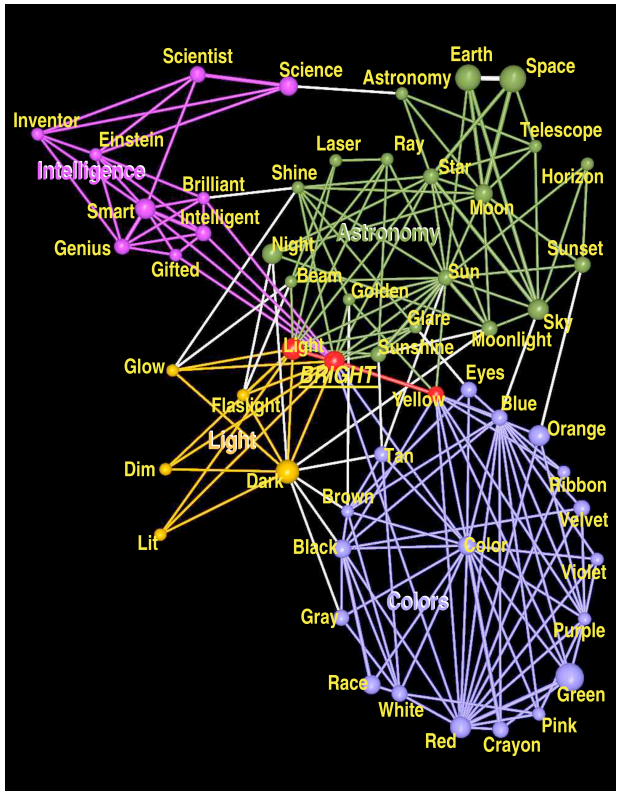
Communities are usually implicitly defined by the specific algorithm adopted, without an explicit definition!

The practical definition may depend on the specific system/application

Partitions: basics

A partition is a division of a graph into clusters, such that each vertex is assigned to one and only one cluster!

If vertices can belong to two or more clusters simultaneously, one speaks of covers



G. Palla, I. Derényi, I. Farkas, T. Vicsek, Nature 435, 814, 2005

Partitions: basics

The number of possible partitions in k clusters of a graph with n vertices is the Stirling number of the second kind $S(n,k)$

Total number of possible partitions: Bell number $B_n = \sum_{k=0}^{n} S(n,k)$

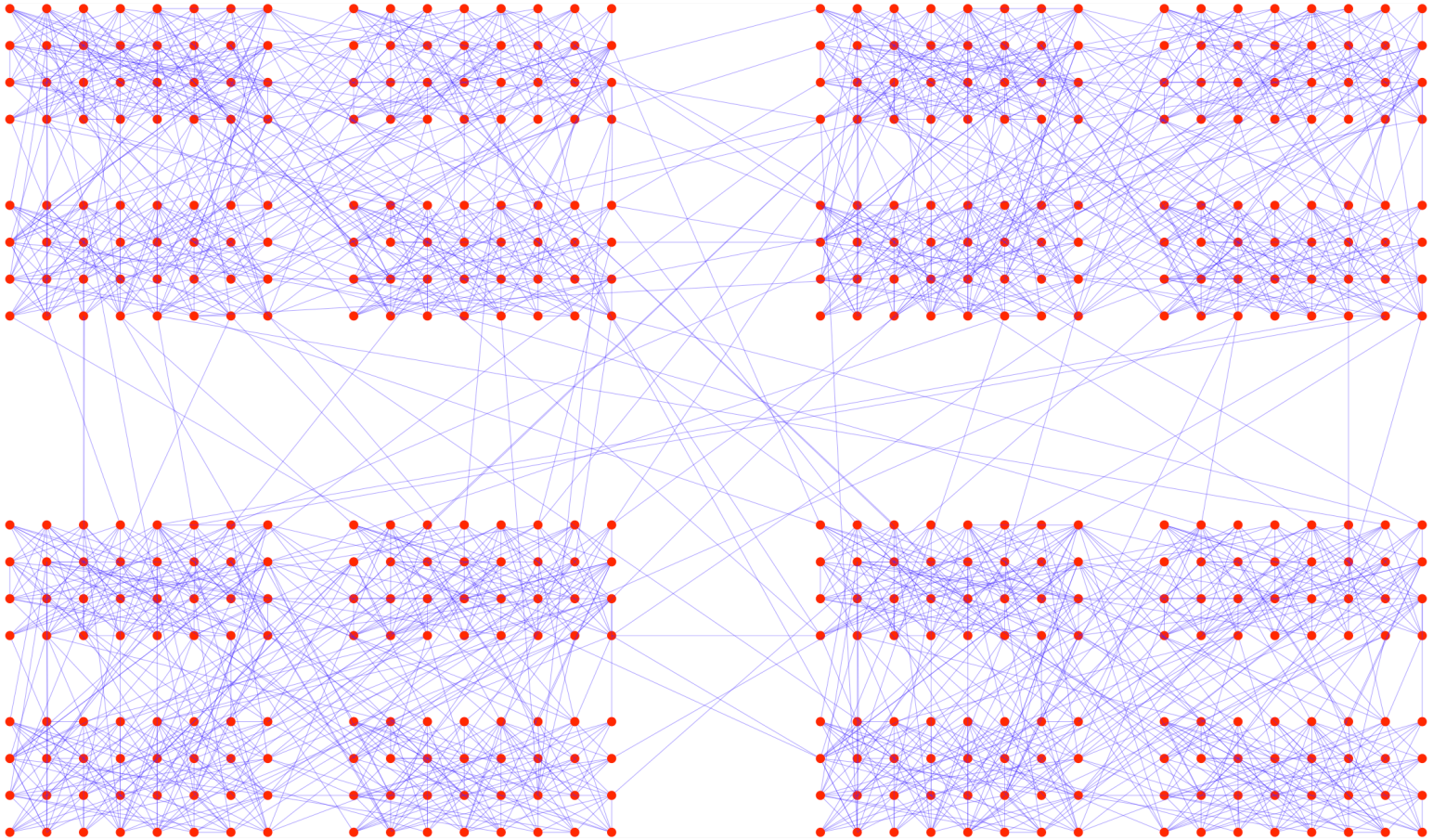
Large-n limit

$$B_n \sim \frac{1}{\sqrt{n}}[\lambda(n)]^{n+1/2} e^{\lambda(n)-n-1}$$

$\lambda(n) = e^{W(n)} = n/W(n), W(n)$ Lambert function

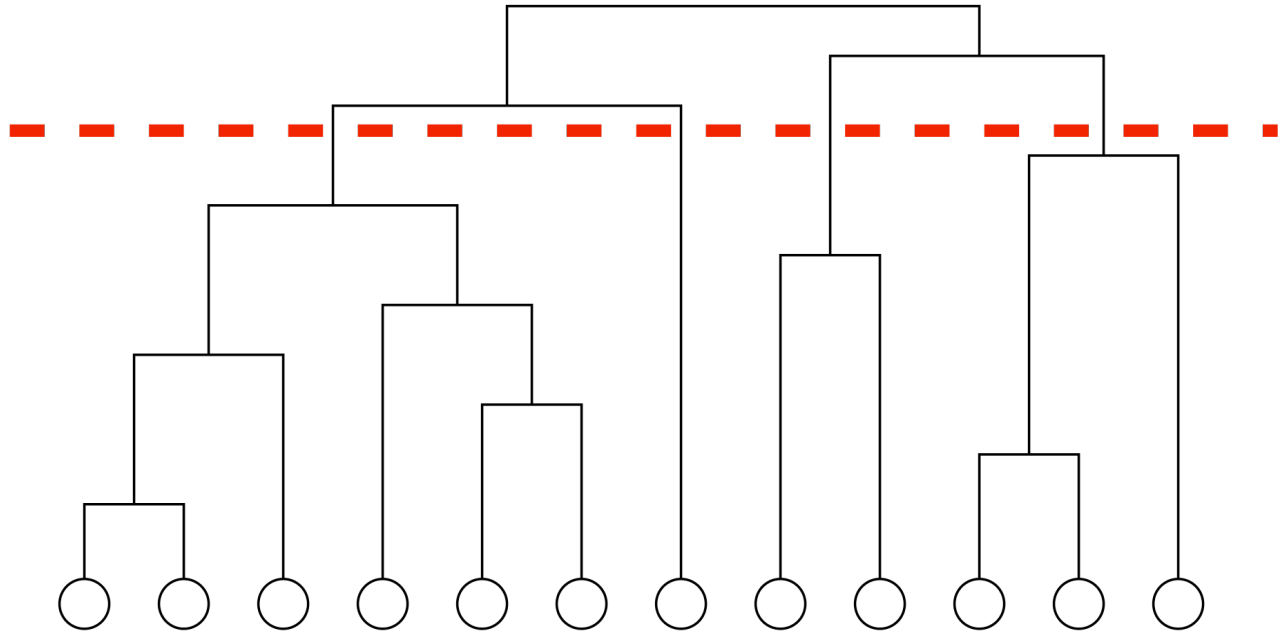
Hierarchy

Clusters may be included in other clusters, etc. (hierarchical order!)



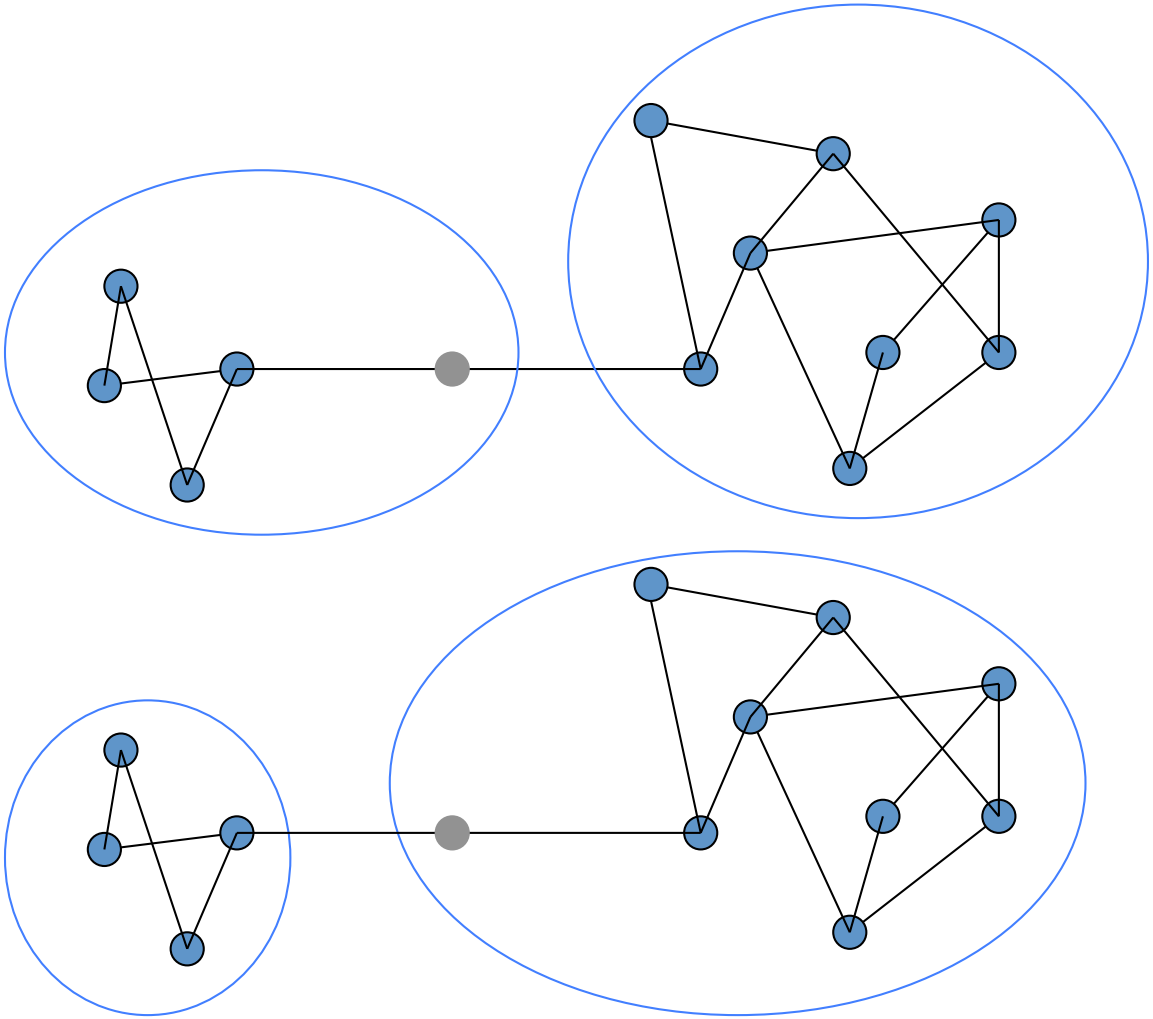
Clauset, Moore & Newman, LNCS 4503, 1, 2007

Dendrograms



Partitions: quality functions

What is a “good” clustering ?



?

Partitions: quality functions

Kleinberg's impossibility theorem (2002)

Set S of points, distance function d, positive definite and symmetric

Theorem: No clustering f based on the distances between the points satisfies the three conditions:

1) Scale-invariance: by multiplying any distance function by any constant α the clustering is the same
2) Richness: any possible partition of S can be recovered with a suitable choice of the distance function d
3) Consistency: given a partition, any modification of the distance function that does not decrease the distance between points of different clusters and does not increase the distance between points of the same cluster, yields the same clustering

Partitions: quality functions

A quality function assigns a score to each partition/cover of a graph

High-score partitions are “good”, low-score partitions are “bad”

Additivity

$$Q(\mathcal{P}) = \sum_{\mathcal{C} \in \mathcal{P}} q(\mathcal{C})$$

Examples of quality functions

1) Performance:

$$P(\mathcal{P}) = \frac{|\{(i,j) \in E, C_i = C_j\}| + |\{(i,j) \notin E, C_i \neq C_j\}|}{n(n-1)/2}$$

2) Coverage:

$$C(\mathcal{P}) = \frac{|\{(i,j) \in E, C_i = C_j\}|}{m}$$

Partitions: modularity

Newman & Girvan, 2004

Principle: random graphs have no community structure!

Method: comparing the edge density in each cluster with the edge density of the cluster in a randomized version of the graph

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

Null model in principle arbitrary

Ex. Bernoulli random graph

$$P_{ij} = p = 2m/[n(n-1)]$$

Partitions: modularity

Problem with Bernoulli random graph: degree distribution is binomial/ Poissonian

Modularity's null model: random graph with identical expected degree sequence of original graph

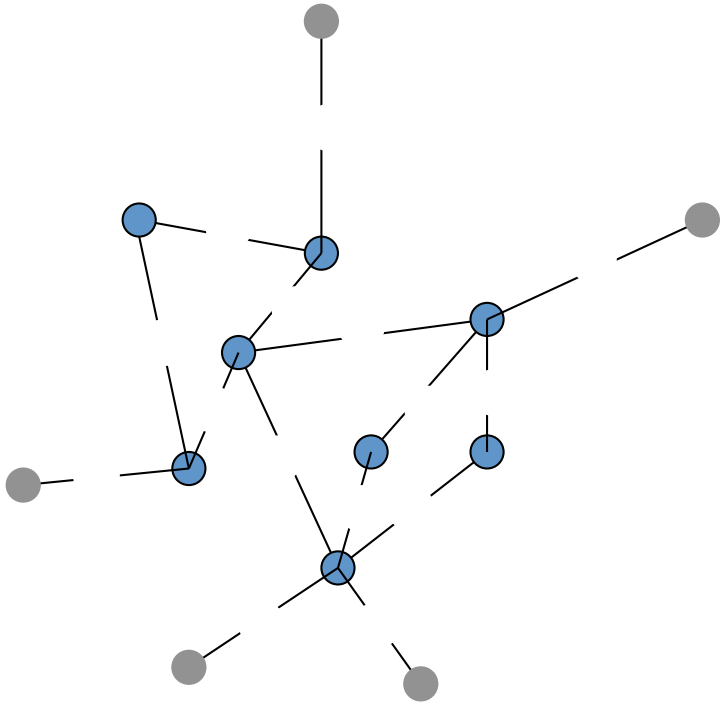
$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(C_i, C_j)$$

$$Q = \sum_{c=1}^{n_c} \left[\frac{l_c}{m} - \left(\frac{d_c}{2m} \right)^2 \right]$$

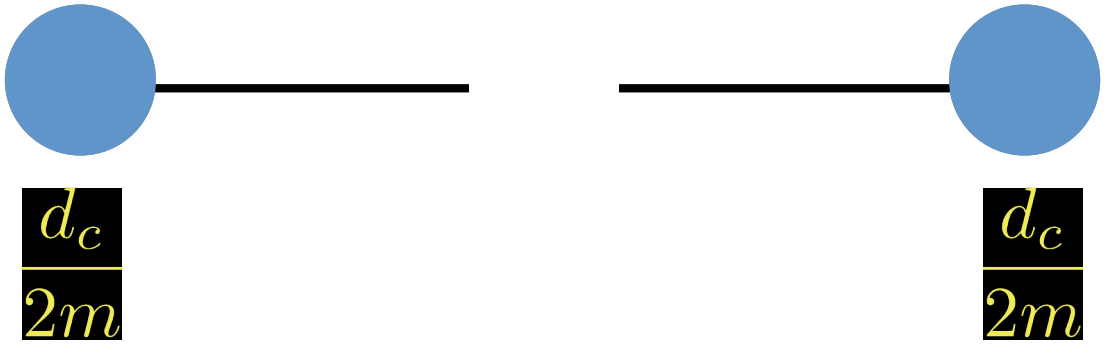
$l_c =$ Number of edges inside cluster c

$d_c =$ Total degree of cluster c

$n_c =$ Number of clusters



$\frac{d_c}{2m}$ = probability that a stub, randomly selected, ends in module c



$$\frac{d_c}{2m} \cdot \frac{d_c}{2m} = \frac{d_c^2}{4m^2}$$

probability that the link is internal to module c

$$\frac{d_c^2}{4m^2} \cdot m = \frac{d_c^2}{4m}$$

expected number of links in module c

Partitions: modularity

$$Q = \sum_{c=1}^{n_c} \left[\frac{l_c}{m} - \left(\frac{d_c}{2m} \right)^2 \right]$$

Some features:

1) $Q < 1$
2) $Q = 0$ for the partition in which the whole graph is one cluster
3) Modularity can be negative (multipartite structure)
4) Modularity in general depends on graph size: partitions of different graphs cannot be compared to each other based on their modularity values
5) Large values of modularity not necessarily indicate good partitions: random graphs may have high-modularity partitions

Traditional methods: graph partitioning

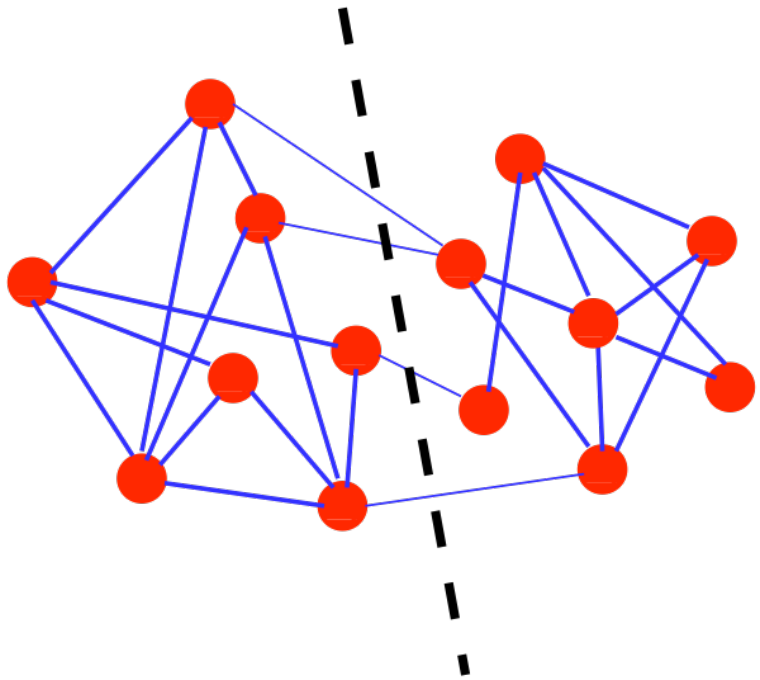
"Divide a graph in n parts, such that the number of links between them (*cut size*) is minimal"

Problems

1. Number of clusters must be specified
2. Size of the clusters must be specified (other measures may help)

Graph bisectioning

"Divide a graph in two parts of equal size, such that the number of links between them (*cut size*) is minimal"



Kernighan-Lin algorithm

Kernighan & Lin, 1970

1) Split in two groups of predefined size
2) At each step, a pair of nodes of different groups are swapped so to decrease the cut size
3) (Some) moves decreasing the cut size are accepted to avoid being trapped in local minima
4) After a series of moves one picks the one which yielded the lowest cut size, which is used as starting point for a new iteration

Complexity: $O(n^2 \log n)$ if each iteration consists of a constant number of moves [on sparse graphs it can be lowered to $O(n^2)$]

Performance: results strongly dependent on initial partition, the method is often used to improve the results of other methods

Spectral partitioning

Laplacian matrix

Unnormalized Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A} \quad (D_{ij} = \delta_{ij} k_i)$

Normalized Laplacian $\mathbf{L_{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$

Normalized Laplacian $\mathbf{L_{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} = \mathbf{I} - \mathbf{T}$

Spectral partitioning

Index vector: $\mathbf{s} \rightarrow s_i = \pm 1$

Relation between cut size R and Laplacian

$$R = \frac{1}{4}\mathbf{s}^T \mathbf{L}\mathbf{s}$$

$\mathbf{s} = \sum_i a_i \mathbf{v}_i, \quad \mathbf{v}_i$ Laplacian eigenvectors

$R = \sum_i a_i^2 \lambda_i, \quad \lambda_i$ Laplacian eigenvalues

Reminder: $\lambda_1 = 0, \lambda_2 > 0$

Spectral partitioning

Special case: $\lambda_2 \sim 0, \lambda_2 \ll \lambda_i \quad (i > 2)$

Cut size R is minimal (with good approximation) for

$\mathbf{s} = \mathbf{v_2}, \quad \mathbf{v_2}$ is the Fiedler vector

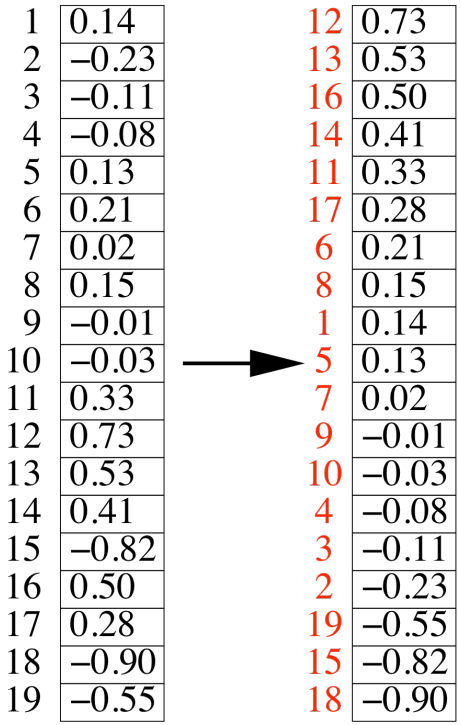
$R = \lambda_2$

Problem: components of Fiedler vector are real-valued, those of index vector are integer (±1) by definition

Best approximation: separating vertices with positive and negative components of the Fiedler vector

Spectral partitioning

If one wants a split into n_1 and $n_2=n-n_1$ vertices, one orders the components of the Fiedler vector from the largest positive to the smallest negative and picks the n_1 largest (smallest) components of the Fiedler vector



Spectral partitioning

Complexity

Eigenvectors of the Laplacian matrix can be computed with the Lanczos method, complexity depends on size of eigengap

$$\lambda_{k+1} - \lambda_k$$

Performance

Results are good, they can be further improved with the Kernighan-Lin algorithm

Partitions in more than two clusters can be obtained via iterative bisectioning

Graph partitioning: alternative measures

Conductance $\Phi(\mathcal{C})$ of subgraph $\mathcal{C}$ of graph $\mathcal{G}$

$$\Phi(\mathcal{C}) = \frac{c(\mathcal{C}, \mathcal{G} \setminus \mathcal{C})}{min(k_{\mathcal{C}}, k_{\mathcal{G}\setminus\mathcal{C}})}$$

Ratio cut

$$\Phi_C(\mathcal{C}) = \frac{c(\mathcal{C}, \mathcal{G} \setminus \mathcal{C})}{n_{\mathcal{C}} n_{\mathcal{G}\setminus\mathcal{C}}}$$

Normalized cut

$$\Phi_N(\mathcal{C}) = \frac{c(\mathcal{C}, \mathcal{G} \setminus \mathcal{C})}{k_{\mathcal{C}}}$$

Advantage over cut size: no need to specify size of clusters

Graph partitioning: alternative measures

Optimization of conductance is NP-hard (Šima & Schaeffer, 2006)

Optimization of ratio cut is NP-hard (Matula & Shahrokhi, 1990)

Optimization of normalized cut is NP-complete (Shi & Malik, 2000)

Ratio cut and normalized cut can be well normalized via spectral clustering

Problems of graph partitioning methods:

1) Number of clusters needs to be given by input
2) Iterative bisectioning not ideal to find partitions in more than two clusters

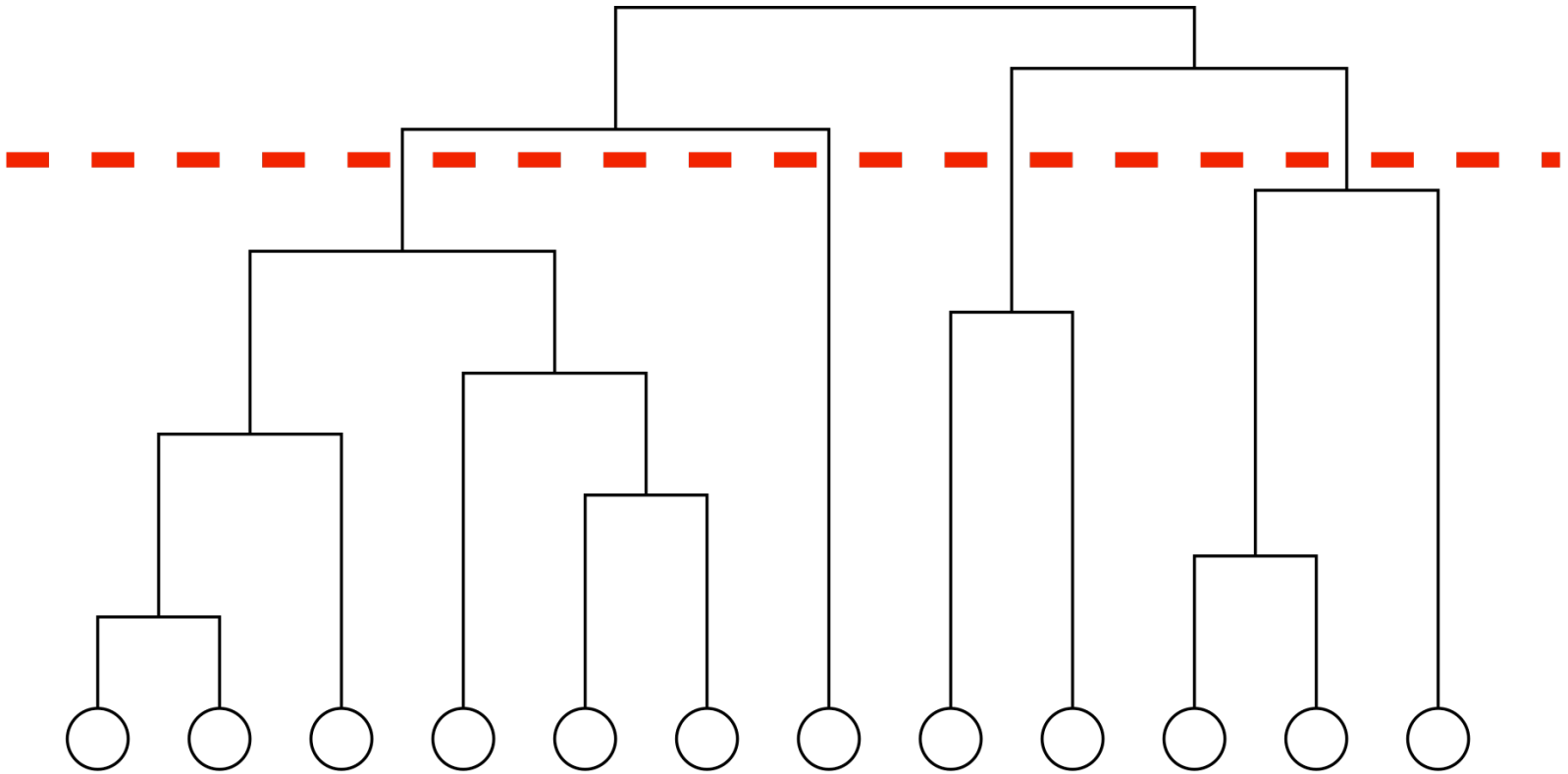
Hierarchical clustering

Very common in social network analysis

Methods can be agglomerative or divisive

1. A criterion is introduced to compare nodes based on their similarity
2. A similarity matrix X is constructed: the similarity of nodes i and j is X_{ij}
3. (Agglomerative) Starting from the individual nodes, larger groups are built by joining groups of nodes based on their similarity
4. (Divisive) Starting from the graph as a single cluster, separates the most dissimilar parts, etc.

Hierarchical clustering



Hierarchical clustering: agglomerative techniques

Problem: how to define similarity of clusters from similarity matrix

1. Single-linkage clustering: similarity between two clusters is minimum element x_{ij} with i in one cluster and j in the other
2. Complete-linkage clustering: similarity between two clusters is maximum element x_{ij} with i in one cluster and j in the other
3. Average-linkage clustering: average of the similarity elements x_{ij} with i in one cluster and j in the other

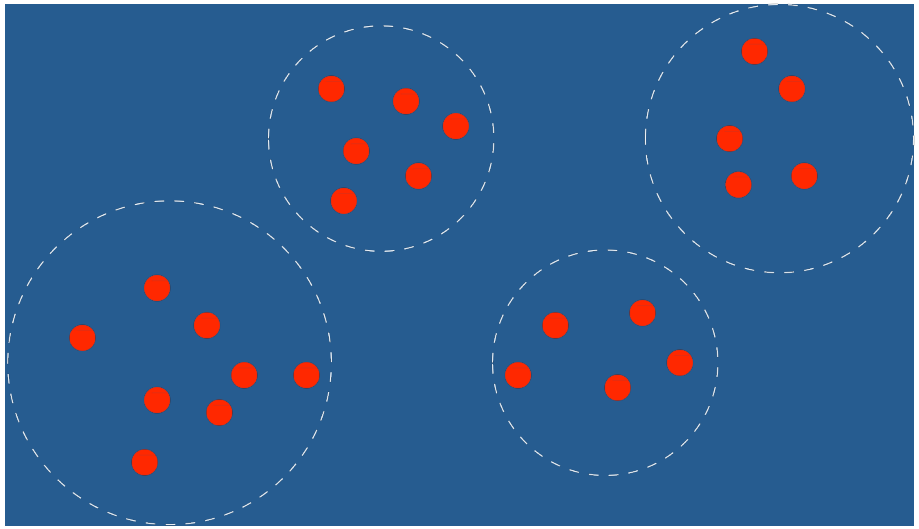
Problems of hierarchical clustering

1. n-1 partitions recovered: which ones are meaningful?
2. Hierarchical structure given by the method may be artificial
3. Vertices with just one neighbor are often classified as separate clusters
4. Methods do not scale well: if graph is embedded in space the complexity is $O(n^2)$ for single-linkage and $O(n^2 \log n)$ for complete and average linkage clustering

Partitional clustering

Set of data points, distance for each pair of points i,j

Goal: dividing the points in k groups such to maximize/ minimize a given measure



Partitional clustering

Minimum k-clustering : minimizing the “diameter” of a cluster, i.e. the largest distance between points of the cluster

k-clustering sum : minimizing the average intra-cluster distance

k-center : minimizing the maximum distance of cluster points from a “centroid”

k-median : minimizing the average distance of cluster points from a “centroid”

K-means clustering

MacQueen, 1967

Principle:

1) K points in space, or “centroids”
2) Minimization of the total squared distance of each point from its centroid

$$\sum_{i=1}^{k} \sum_{\mathbf{x_j} \in S_i} ||\mathbf{x_j} - \mathbf{c_i}||^2$$

K-means clustering

The Lloyd algorithm (Lloyd, 1982)

1. Initial distribution of centroids, as far as possible from each other
2. Each vertex is assigned to the nearest centroid
3. Centers of mass of clusters are computed
4. The centers of mass become the new centroids
5. Repeat from 2

Plus: convergence is fast

Minus: strong dependence on initial conditions

Partitional clustering

Problem: number of clusters needs to be given as input, like in hierarchical clustering

Spectral clustering

Spectral clustering includes all clustering methods that use the eigenvectors of graph matrices

How it works

1) Graph vertices are transformed into a set of n points in a k-dimensional Euclidean space, where k is the number of clusters: coordinates are eigenvector components
2) Points are grouped in clusters via standard methods like, e.g., k-means clustering

Advantage over direct clustering of vertices: change of representation makes clustering properties more evident!

Spectral partitioning

Laplacian matrix

Unnormalized Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A} \quad (D_{ij} = \delta_{ij} k_i)$

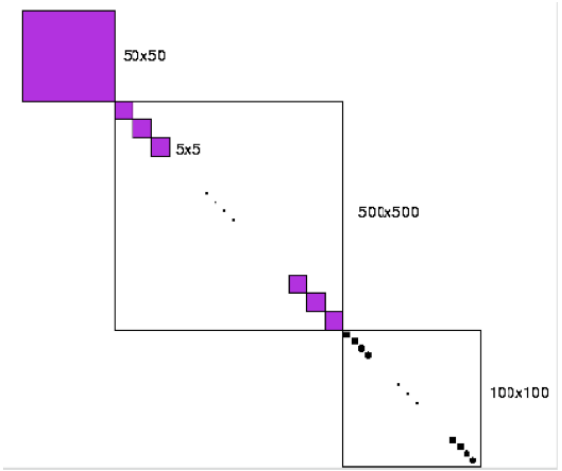
Normalized Laplacian $\mathbf{L_{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$

Normalized Laplacian $\mathbf{L_{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} = \mathbf{I} - \mathbf{T}$

Spectral clustering

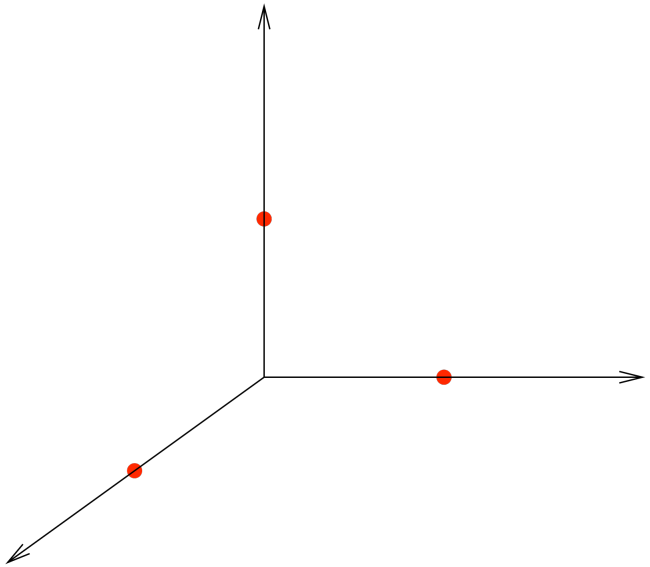
Properties of the Laplacian matrix

- All eigenvalues are non-negative
- If the graph is divided in g components, there are g zero eigenvalues
- In this case the Laplacian can be rewritten in a block-diagonal form

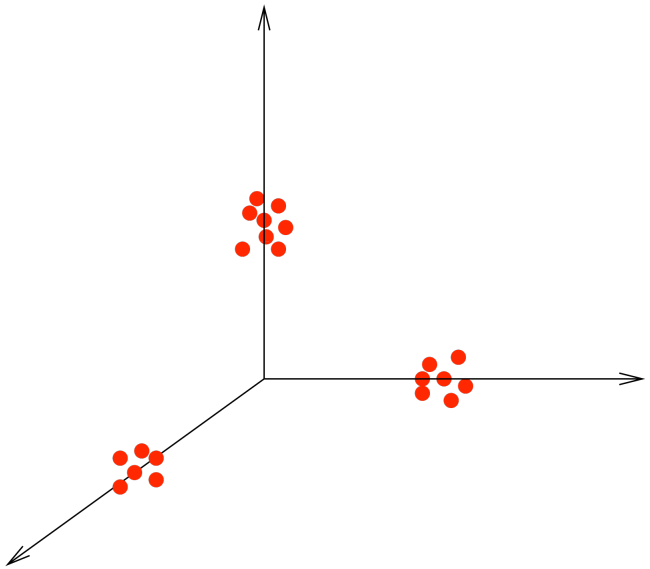


Spectral clustering

Disconnected graph with 3 connected components



Connected graph with 3 weakly connected clusters



Unnormalized spectral clustering

Inputs: number of clusters k, adjacency (weight) matrix A (W)

Steps

1) Compute the k eigenvectors corresponding to the k lowest eigenvalues of the unnormalized Laplacian $\mathbf{L}$
2) Matrix $\mathbf{V}$ is built, $n \times k$ matrix whose columns are the k eigenvectors of $\mathbf{L}$
3) The n rows of $\mathbf{V}$ are vectors with k components representing the vertices as points in an Euclidean space
4) Points are grouped in k clusters with k-means clustering or similar methods

Normalized spectral clustering I

Shi & Malik, 1997

Inputs: number of clusters k, adjacency (weight) matrix A (W)

Steps

1) Compute the k eigenvectors corresponding to the k lowest eigenvalues of the normalized Laplacian $\mathbf{L_{rw}}$
2) Matrix $\mathbf{V}$ is built, $n \times k$ matrix whose columns are the k eigenvectors of $\mathbf{L_{rw}}$
3) The n rows of $\mathbf{V}$ are vectors with k components representing the vertices as points in an Euclidean space
4) Points are grouped in k clusters with k-means clustering or similar methods

Normalized spectral clustering II

Ng et al., 2001

Inputs: number of clusters k, adjacency (weight) matrix A (W)

Steps

1) Compute the k eigenvectors corresponding to the k lowest eigenvalues of the normalized Laplacian $\mathbf{L_{sym}}$
2) Matrix $\mathbf{V}$ is built, $n \times k$ matrix whose columns are the k eigenvectors of $\mathbf{L_{sym}}$
3) The elements of each row of $\mathbf{V}$ are normalized by dividing them by their sum
4) The n rows of $\mathbf{V}$ are vectors with k components representing the vertices as points in an Euclidean space
5) Points are grouped in k clusters with k-means clustering or similar methods

Properties of spectral clustering

Graph partitioning: relaxed optimization

Ex. Finding bipartition with minimum cut size R

$$R = \frac{1}{4}\mathbf{s}^T\mathbf{L}\mathbf{s}$$

in the space of all vectors **s**, with real-valued components, not integer!

Good solution: **s** = Fiedler vector

Problem: what is the relationship between this solution and the actual solution one seeks, where **s** has integer components?

Properties of spectral clustering

Relaxed optimization versus spectral clustering

1) Minimum ratio cut partition in k clusters ➔ unnormalized spectral clustering
2) Minimum normalized cut partition ➔ normalized spectral clustering à la Shi-Malik

Random walks versus spectral clustering

$$\mathbf{L_{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} = \mathbf{I} - \mathbf{T}$$

Transfer matrix

Ex. Normalized cut for a bipartition equals the total probability that a walker moves from one cluster to the other in either sense (Meilă & Shi, 2001)

Properties of spectral clustering

Complexity

Eigenvectors of the Laplacian matrix can be computed with the Lanczos method, complexity depends on size of eigengap

$$\lambda_{k+1} - \lambda_k$$

Problem of spectral clustering methods: number of clusters needs to be given by input

Possible solution: searching for gaps in the spectrum, but not easy for graphs with mixed communities

Warning: using $\mathbf{L}$ or $\mathbf{L_{rw}}$; $\mathbf{L_{sym}}$ eigenvector components do not have approximately the same value for vertices in the same cluster!

Problems of traditional methods

- Graph partitioning, partitional clustering and spectral clustering: one needs to specify the number and the size of the clusters
- Hierarchical clustering: many partitions recovered, which one is the best?

One would like methods that can predict the number and the size of the partition and indicate a subset of "good" partitions

Plan of the course

I. Networks: definitions, characteristics, basic concepts in graph theory
II. Real world networks: basic properties
III. Models
IV. Community structure I
V. Community structure II
VI. Dynamic processes in networks