

Chapter 8:
Graph Data

Part 2:
Community Detection

Based on
Leskovec, Rajaraman, Ullman 2014:
Mining of Massive Datasets

Outline

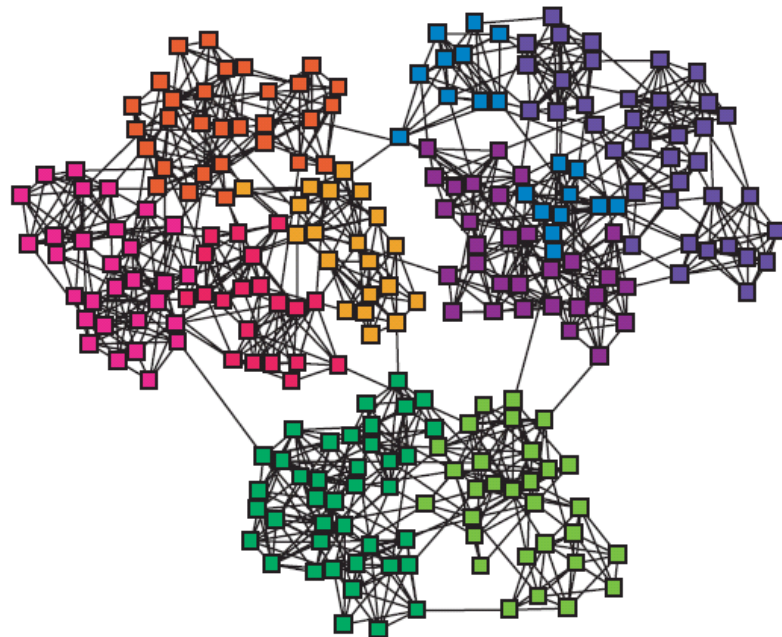
Community Detection

- Social networks
- Betweenness
 - Girvan-Newman Algorithm
- Modularity
- Graph Partitioning
 - Spectral Graph Partitioning
- Trawling

Networks & Communities:

Think of networks being organized into:

- **Modules**
- **Cluster**
- **Communities**



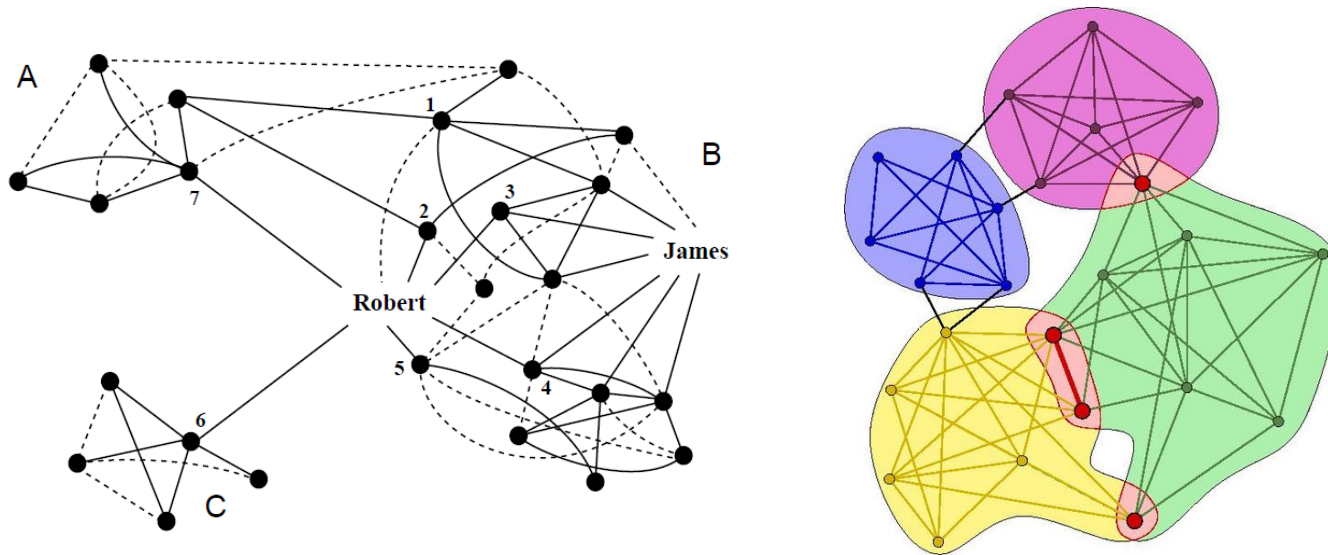
→ **Goal:** Find densely linked clusters

What is a Social Network?

Characteristics of a social network:

- **Collection of entities** participating in the network (entities might be individuals, phone numbers, email addresses , ...)
- At least one **relationship between entities** of the network. (Facebook: 'friend'). Relationship can be all-or-nothing or specified by a degree (e.g. fraction of the average day that two people communication to each other)
- Assumption of **nonrandomness** or **locality**, i.e. relationships tend to cluster. (e.g. A is related to B and C \rightarrow higher probability that B is related to C)

How to find communities?



- Here we will work with undirected (unweighted networks)
- We need to resolve 2 questions:
 - How to compute betweenness?
 - How to select the number of clusters?

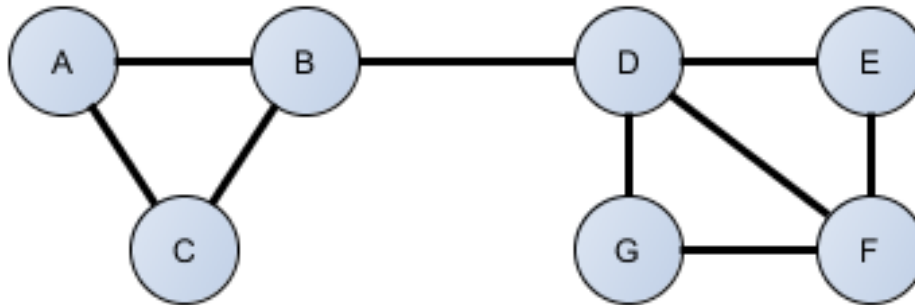
Outline

Community Detection

- Social networks
- Betweenness
 - Girvan-Newman Algorithm
- Modularity
- Graph Partitioning
 - Spectral Graph Partitioning
- Trawling

Betweenness

Definition: The betweenness of an edge (a,b) is the number of pairs of nodes x and y such that (a,b) lies on the shortest-path between x and y .



Example:

Edge (B,D) has highest betweenness (shortest path of A,B,C to any of D,E,F,G)

→ Betweenness of (B,D) aggregates to: $3 \times 4 = 12$

Betweenness of edge (D,F) ?

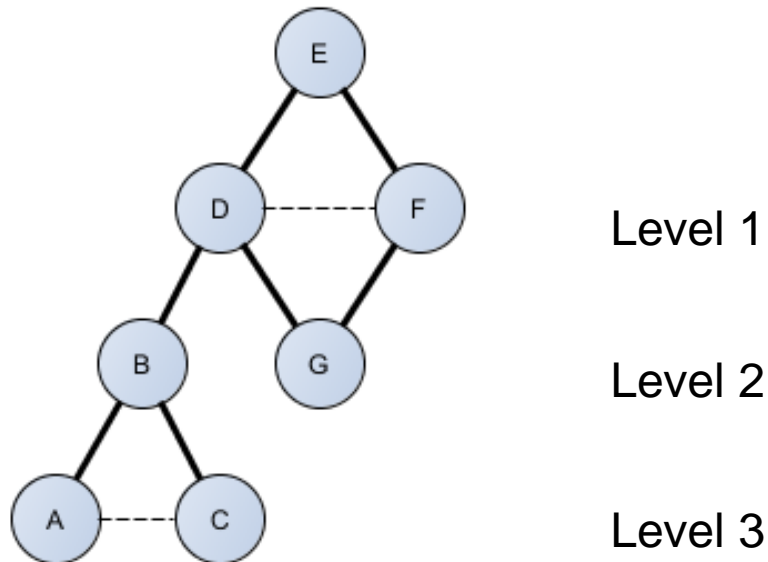
Girvan-Newman Algorithm

Goal: Computation of betweenness of edges

Step 1: Perform a breadth-first search, starting at node X and construct a DAG (directed, acyclic graph)

Example:

- Start at node E

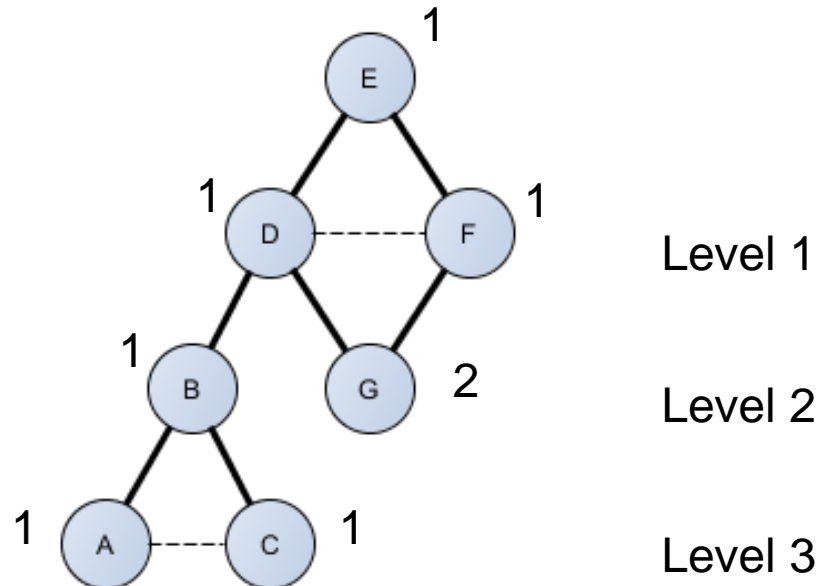


Girvan-Newman Algorithm

Goal: Computation of betweenness of edges

Step 2: label each node by the number of shortest paths that reach it from the root. Label of root = 1, each node is labeled by the sum of its parents.

Example:



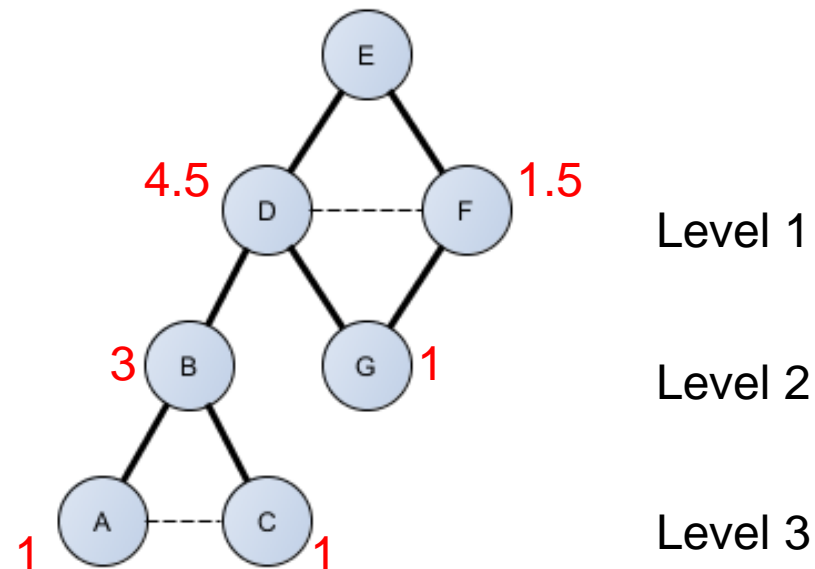
Girvan-Newman Algorithm

Goal: Computation of betweenness of edges

Step 3: calculate for each edge e the sum over all nodes Y the fraction of shortest paths from the root X to Y :

In Detail:

1. Each leaf gets a credit of 1.
2. Non-leaf nodes gets a credit of 1 plus the sum of its children
3. A DAG edge e entering node Z from the level above is given a share of the credit of Z proportional to the fraction of shortest paths from the root to Z . **Formally:** let Y_1, \dots, Y_k be the parent nodes of Z with $p_i, 1 \leq i \leq k$, be the number of shortest path to Y_i . The credit for edge (Y_i, Z) is given by: $Z * p_i / \sum_{j=1}^k p_j$



Find Communities using Betweenness

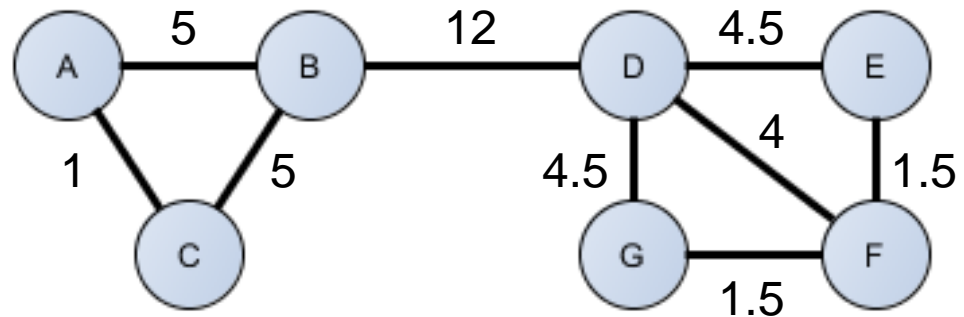
Idea: Clustering is performed by taking the edges in order of increasing betweenness and add them to the graph or as a process of edge removal

Example:

GN-Algorithm has been performed for every node and the credit of each edge has been calculated (by summing the credits up and dividing them by 2. **Why?**)

Remove edges, starting by highest betweenness:

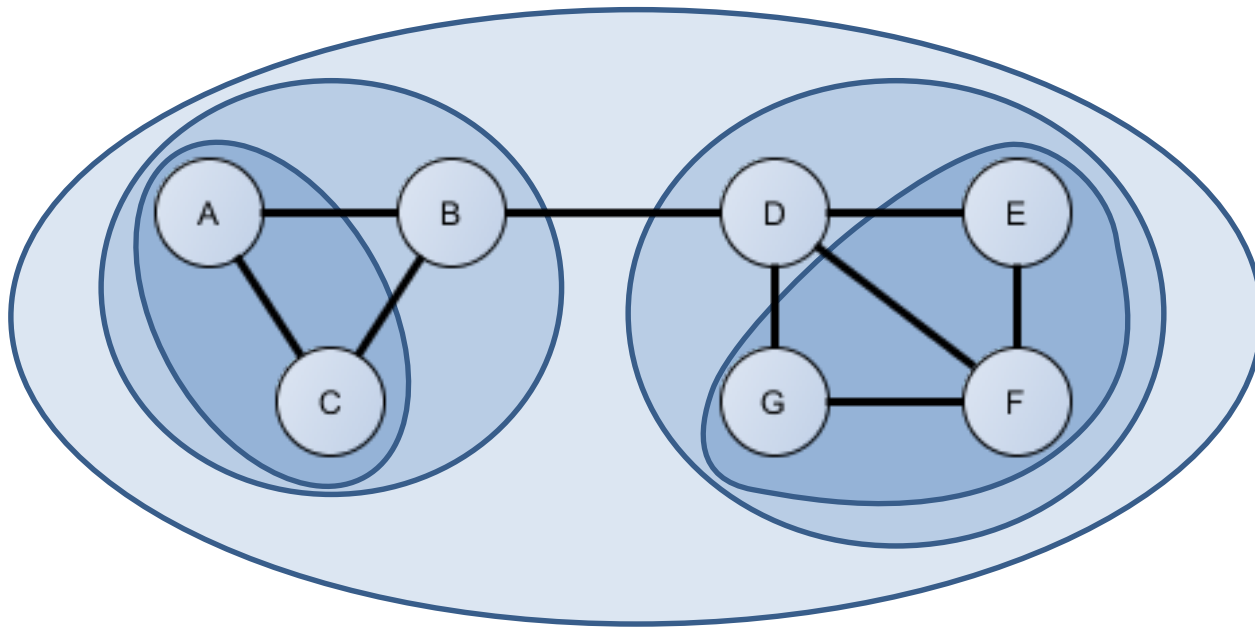
- 1. Remove (B,D)
 - Communities {A,B,C} and {D,E,F,G}
- 2. Remove (A,B), (B,C), (D,G), (D,E), (D,F)
 - Communities {A,C} and {E,F,G}
 - Node B and D are encapsulated as ,traitors' of communities



Find Communities using Betweenness

Girvan-Newman Algorithm:

- connected components are communities
- gives a hierarchical decomposition of the network



Outline

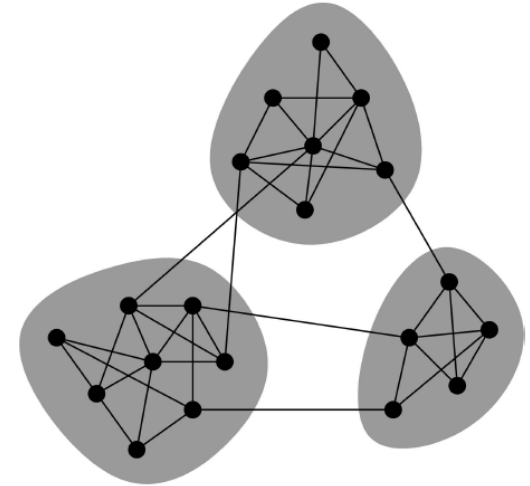
Community Detection

- Social networks
- Betweenness
 - Girvan-Newman Algorithm
- Modularity
- Graph Partitioning
 - Spectral Graph Partitioning
- Trawling

Network Communities

- ✓ How to compute betweenness?
- How to select the number of clusters?

Communities: sets of tightly connected nodes



Modularity Q:

- A measure of how well a network is partitioned into communities
- Given a partitioning of the network into groups $s \in S$:

$$Q \propto \sum_{s \in S} [(\#edges\ within\ group\ s) - \underbrace{(\text{expected}\ \#edges\ within\ group\ s)}_{\text{Defined by null model}}]$$

Defined by null model

Null Model: Configuration Model

- Given a graph G with **n nodes** and **m edges**, construct rewired network G' :
 - Same degree distribution but random connections
 - Consider G' as a **multigraph**

→ The **expected number of edges between nodes i and j** of degrees k_i and k_j is given by: $\frac{1}{2m} * k_i k_j$

Proof that G' contains the expected number of m edges:

$$\frac{1}{2} \sum_{i \in N} \sum_{j \in N} \frac{k_i k_j}{2m} = \frac{1}{2} \frac{1}{2m} \sum_{i \in N} k_i \left(\sum_{j \in N} k_j \right) = \frac{1}{4m} * 2m * 2m = m$$

Modularity

Modularity of partitioning S of graph G :

$$Q \propto \sum_{s \in S} [(\#edges\ within\ group\ s) - (expected\ \#edges\ within\ group\ s)]$$

$$Q(G, S) = \frac{1}{2m} \sum_{s \in S} \sum_{i \in s} \sum_{j \in s} (a_{ij} - \frac{k_i k_j}{2m})$$

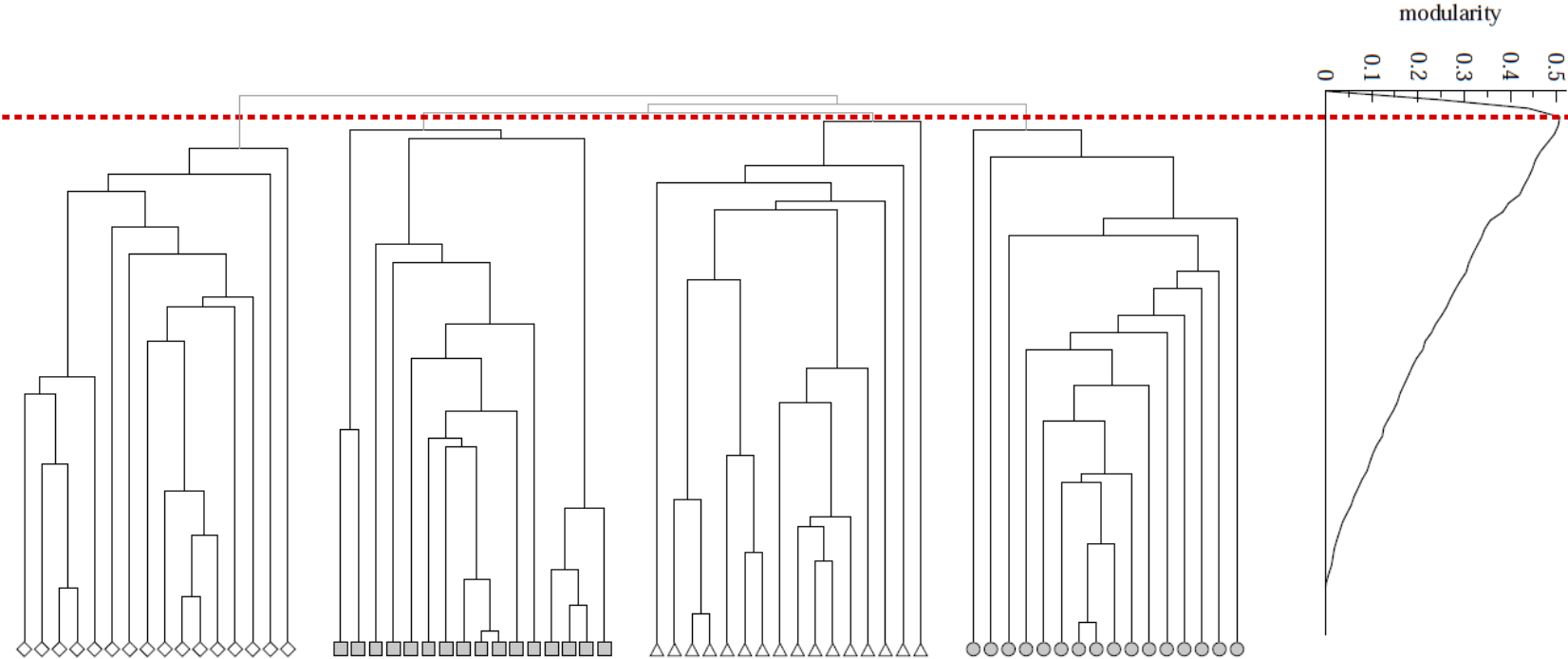
Normalizing: $-1 < Q < 1$

Modularity values take range $[-1, 1]$:

- Positive if the number of edges within groups exceeds the expected number
- **$0.3 - 0.7 < Q$** means significant community structure

Modularity

→ Q is useful for selecting the number of clusters



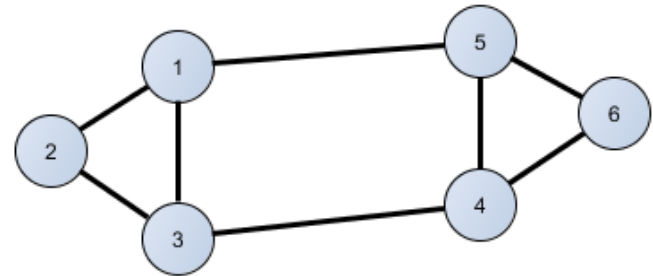
Outline

Community Detection

- Social networks
- Betweenness
 - Girvan-Newman Algorithm
- Modularity
- Graph Partitioning
 - Spectral Graph Partitioning
- Trawling

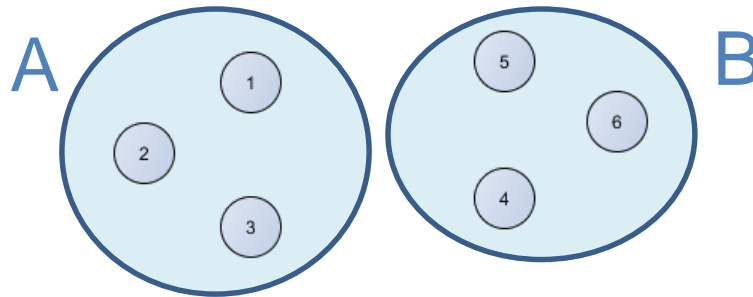
Partitioning of Graphs

Given an undirected Graph $G(V, E)$:



Bi-partitioning task:

- Divide vertices into two **disjoint** groups A, B



Questions:

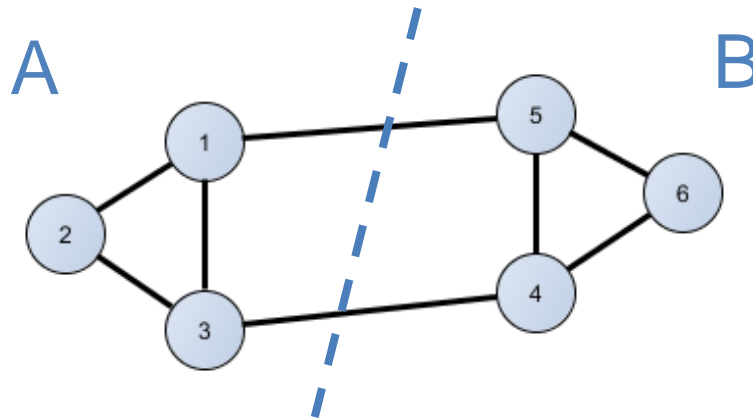
- How can we define 'good' partition of G ?
- How can we efficiently identify such a partition?

Partitioning of Graphs

What makes a good partition?

- Maximize the number of within-group connections
- Minimize the number of between-group connections

Example:



Partitioning of Graphs

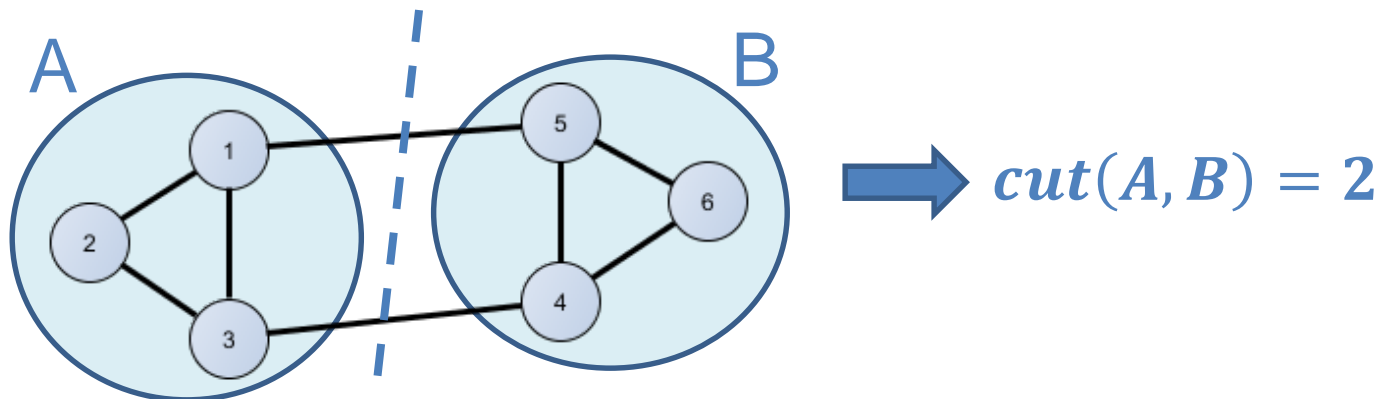
Graph Cuts

Express partitioning objectives as a function of the 'edge cut' of the partition

Cut: Set of edges with only one vertex in a group:

$$cut(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

Example:



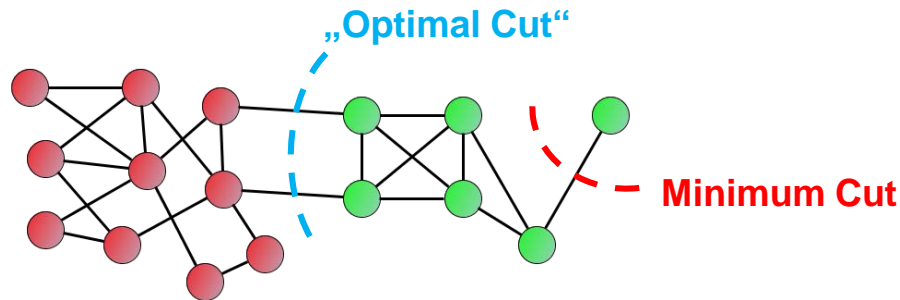
Partitioning of Graphs

Minimum-cut

Minimize weight of connections between groups:

$$\mathit{arg\ min}_{A,B} \mathit{cut}(A, B)$$

Example:



Problem:

- Only considers external cluster connections
- Does not consider internal cluster connectivity

Partitioning of Graphs - Graph Cuts

Normalized-cut: Connectivity between groups relative to the density of each group

$$ncut(A, B) = \frac{cut(A, B)}{vol(A)} + \frac{cut(A, B)}{vol(B)}$$

$vol(X)$: total weight of edges with at least one endpoint in X : $vol(X) = \sum_{i \in X} k_i$

→ Produces more balanced partitions

How to find a good partition efficiently?

Problem: Computing optimal cut is **NP-hard!**

Spectral Graph Partitioning

- **Adjacency matrix** of an undirected Graph G
 $a_{ij} = 1$ if (i, j) exist in G , else 0
- Vector $x \in R^n$ with components (x_1, \dots, x_n)
 Think of it as a label/value of each node of G

What is the meaning of $A * x$?

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} * \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \quad y_i = \sum_{j=1}^n a_{ij} * x_j = \sum_{(i,j) \in E} x_j$$

→ Entry y_i is a sum of labels / values x_j of neighbors of i

Spectral Graph Partitioning

What is the meaning of $A * x$?

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} * \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} * \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

$A * x = \lambda * x$ *Eigenvalue Problem*

Spectral Graph Theory:

- Analyze the ‚spectrum‘ of matrix representing G
- **Spectrum:** Eigenvectors x_i of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues λ_i
- $\Lambda = \{\lambda_1, \dots, \lambda_n\}$ with $\lambda_1 \leq \dots \leq \lambda_n$

Spectral Graph Partitioning

Intuition

Suppose all nodes in G have degree d and G is connected

What are some eigenvalues/vectors of G ?

Eigenvalue Problem: $A * x = \lambda * x \rightarrow$ find λ and x

- Let's try $x = (1, \dots, 1)$
- Then $A * x = (d, \dots, d) = \lambda * x \rightarrow \lambda = d$

Remember:

$$y_i = \sum_{j=1}^n a_{ij} * x_j = \sum_{(i,j) \in E} x_j$$

Spectral Graph Partitioning

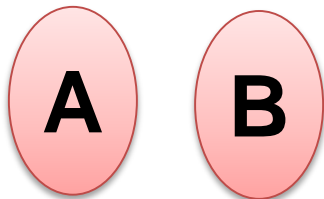
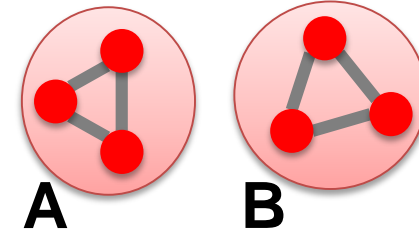
Intuition

What if G is not connected?

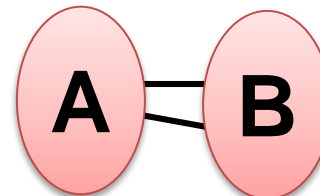
- G has 2 components, each d -regular

What are some eigenvectors?

- $x =$ put all 1s on A and 0s on B or vice versa
 - $x' = (1, \dots, 1, 0, \dots, 0)$, then $A * x' = (d, \dots, d, 0, \dots, 0)$
 - $x'' = (0, \dots, 0, 1, \dots, 1)$, then $A * x'' = (0, \dots, 0, d, \dots, d)$
 - \rightarrow in both cases the corresponding $\lambda = d$



$$\lambda_n = \lambda_{n-1}$$

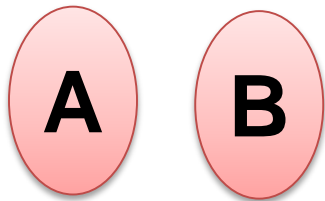


$$\lambda_n - \lambda_{n-1} \approx 0$$

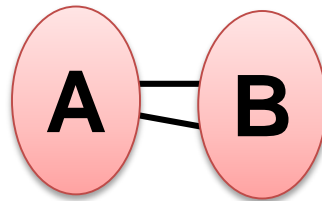
2nd largest eigval.
 λ_{n-1} now has
 value very close
 to λ_n

Spectral Graph Partitioning

Intuition



$$\lambda_n = \lambda_{n-1}$$



$$\lambda_n - \lambda_{n-1} \approx 0$$

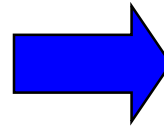
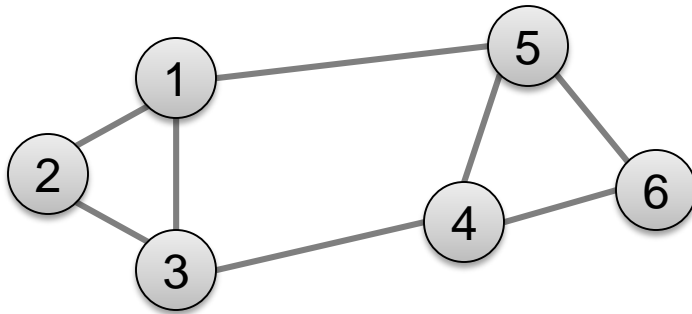
2nd largest eigval.
 λ_{n-1} now has
 value very close
 to λ_n

- If the graph is connected (right example) then we already know that $x_n = (1, \dots, 1)$ is an eigenvector
- Since eigenvectors are orthogonal then the components of x_{n-1} sum to 0
 - Why? \rightarrow Because $\sum_i x_n[i] * \sum_i x_{n-1}[i] = 0$
- **General Idea:** we can look at the eigenvector of the 2nd largest eigenvalue and declare nodes with positive label in A and negative label in B

Spectral Graph Partitioning

- Adjacency matrix A :

- $n \times n$ matrix
- $A = [a_{ij}]$, $a_{ij} = 1$ if there is an edge between node i and j

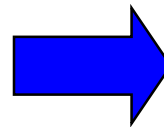
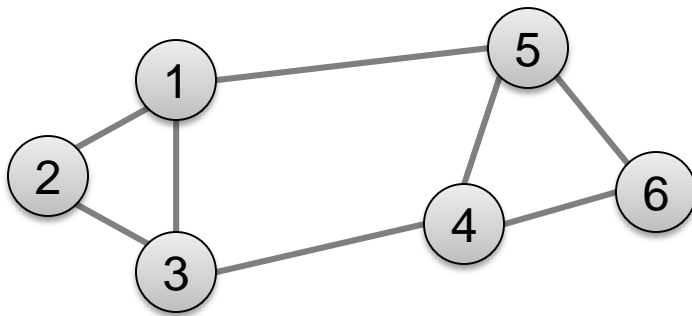


	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

Spectral Graph Partitioning

- Degree matrix D :

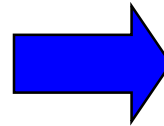
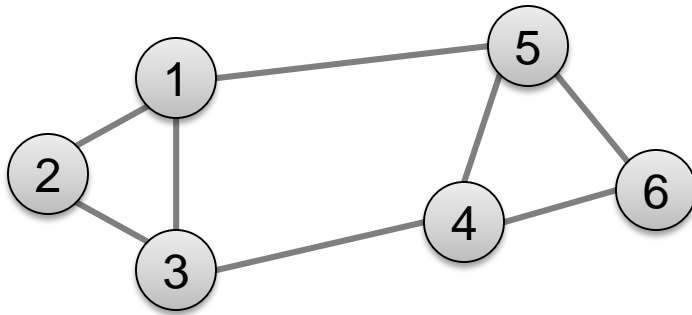
- $n \times n$ diagonal matrix
- $D = [d_{ii}]$, $d_{ii} = \text{degree of node } i$



	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	3	0
6	0	0	0	0	0	2

Spectral Graph Partitioning

- Laplacian Matrix L :
 - $n \times n$ symmetric matrix
 - $L = D - A$



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

- Trivial eigenpair?
 - $X = (1, \dots, 1)$, then $L * x = 0$ and so $\lambda_1 = 0$

Spectral Clustering Algorithms

Three basic stages:

1. Pre-processing

- Construct a matrix representation of the graph

2. Decomposition

- Compute eigenvalues and eigenvectors of the matrix
- Map each point point to a lower-dimensional representation on one or more eigenvectors

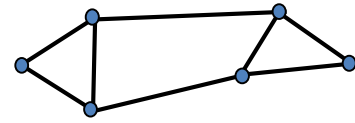
3. Grouping

- Assign points to two or more clusters, based on the new representation

Spectral Clustering Algorithms

1. Pre-processing:

- Build Laplacian matrix L of the graph



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

2. Decomposition:

- Find eigenvalues λ and eigenvectors x of the matrix L
- Map vertices to lower-dimensional representation


 $\lambda =$

0.0
1.0
3.0
3.0
4.0
5.0

 $X =$

0.4	0.3	-0.5	-0.2	-0.4	-0.5
0.4	0.6	0.4	-0.4	0.4	0.0
0.4	0.3	0.1	0.6	-0.4	0.5
0.4	-0.3	0.1	0.6	0.4	-0.5
0.4	-0.3	-0.5	-0.2	0.4	0.5
0.4	-0.6	0.4	-0.4	-0.4	0.0

1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6

How do we now find the clusters?

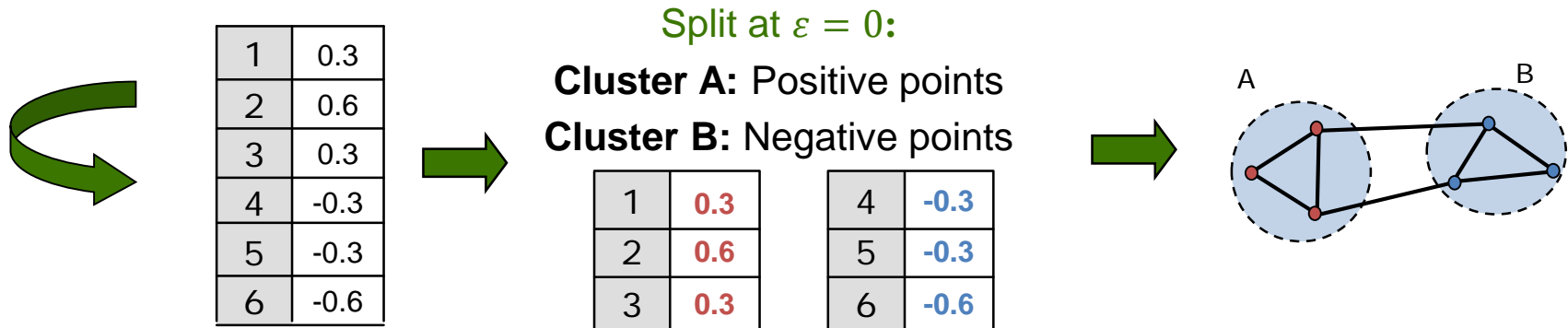
Spectral Clustering Algorithms

3. Grouping:

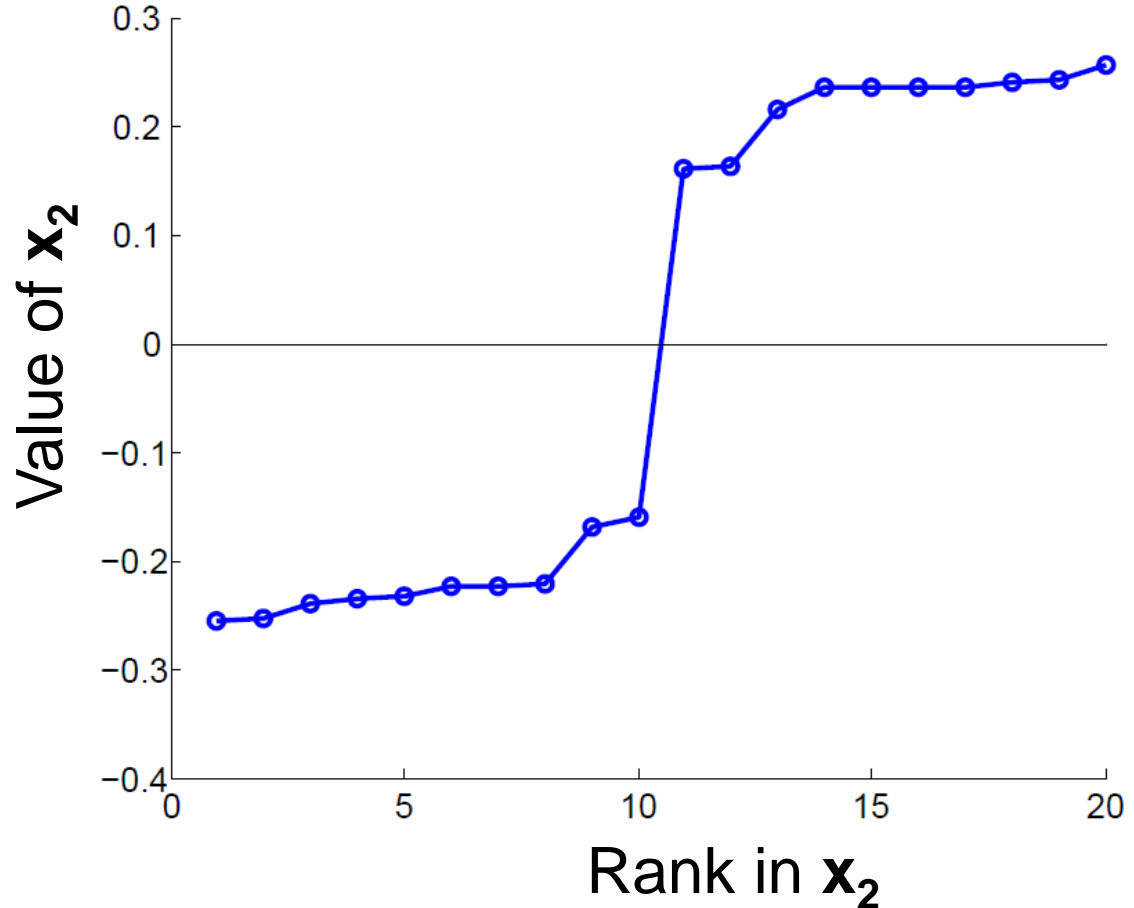
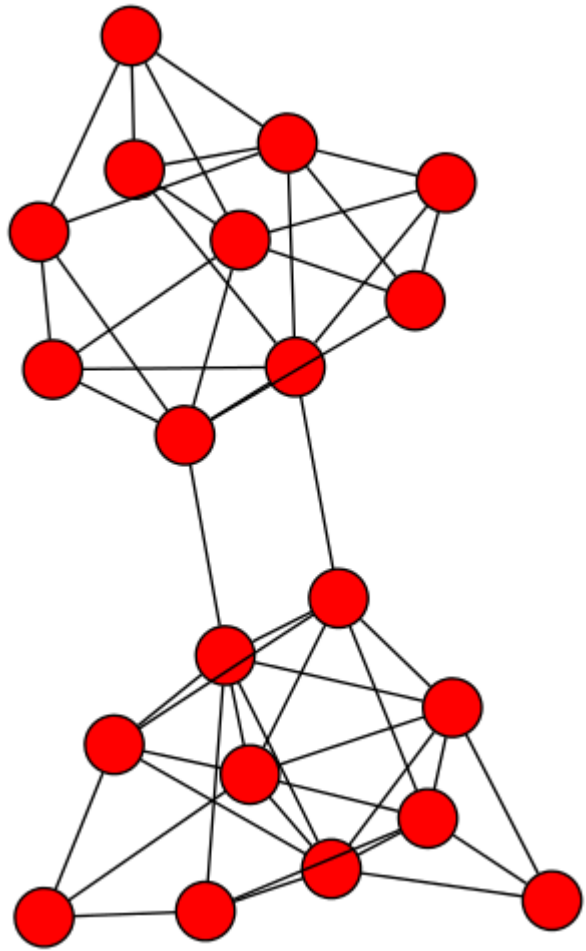
- Sort components of reduced 1-dimensional vector
- Identify clusters by splitting the sorted vector in two (threshold ε)
- By choosing m vectors, there are max. 2^m clusters

→ How to choose a splitting point, i.e threshold ε ?

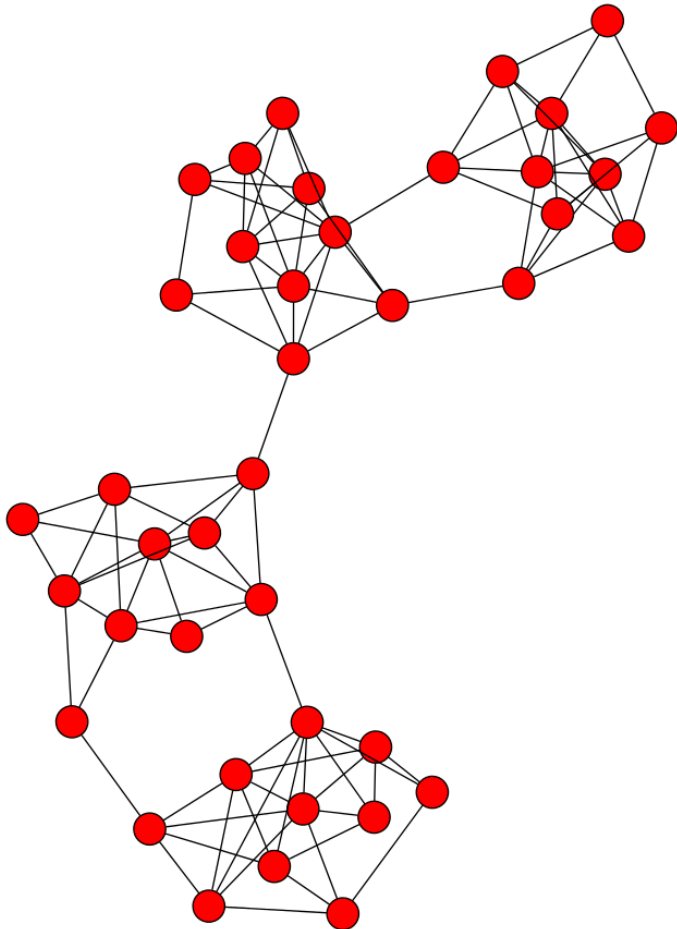
- Naive approaches:
 - Split at $\varepsilon = 0$ or median value
- More expensive approaches:
 - Attempt to minimize normalized cut in 1-dimension
 - (sweep over ordering of nodes induces by the eigenvector)



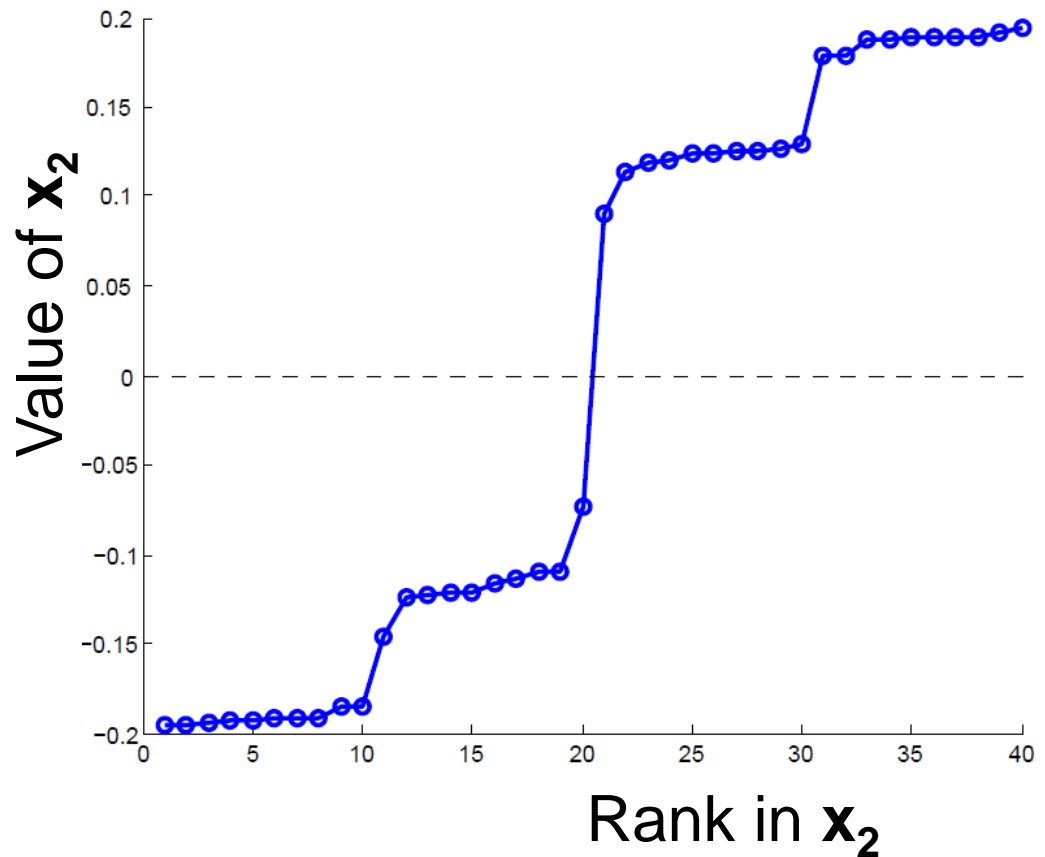
Spectral Clustering Algorithms



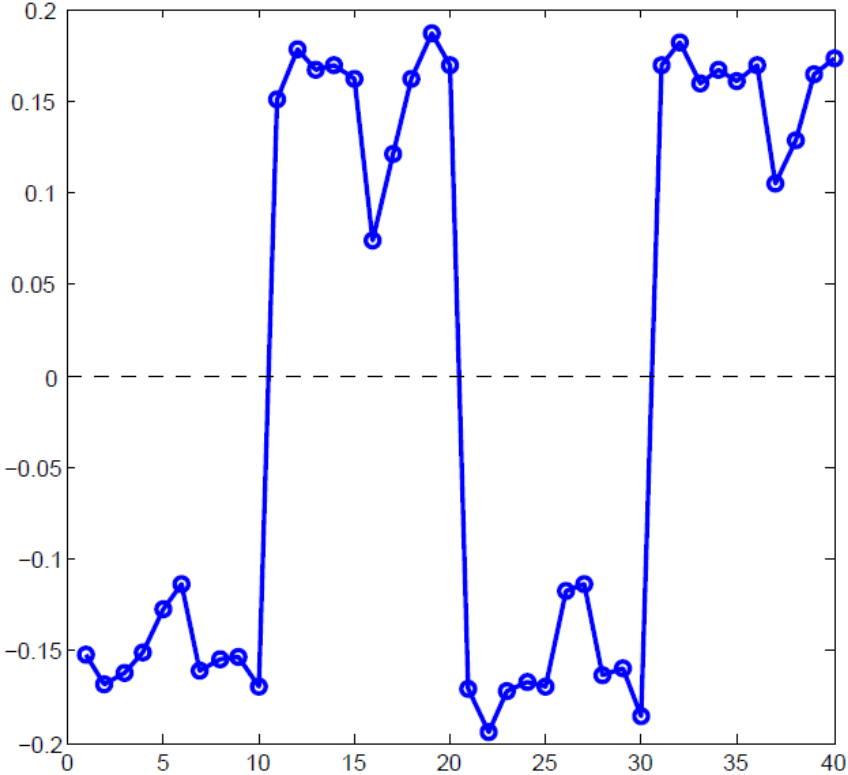
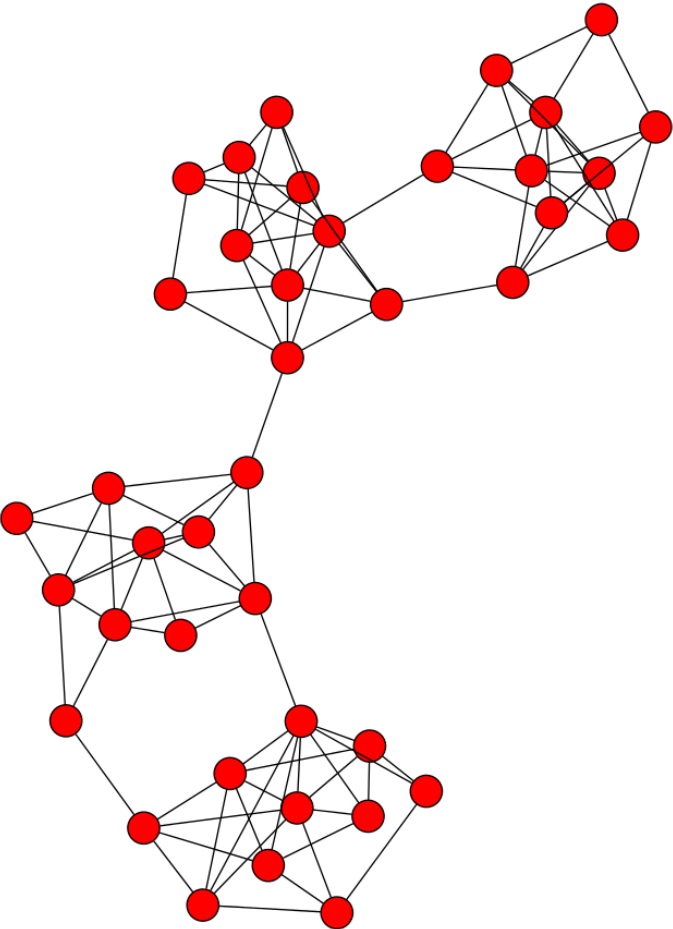
Spectral Clustering Algorithms



Components of x_2



Spectral Clustering Algorithms



Components of x_3

Outline

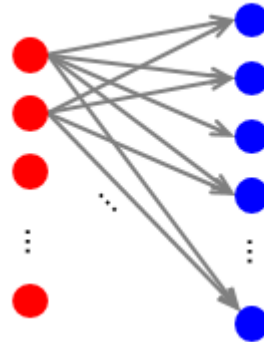
Community Detection

- Social networks
- Betweenness
 - Girvan-Newman Algorithm
- Modularity
- Graph Partitioning
 - Spectral Graph Partitioning
- Trawling

Trawling

- Goal: find small communities in huge graphs
- E.g. how to describe community/discussion in a Web

Example:

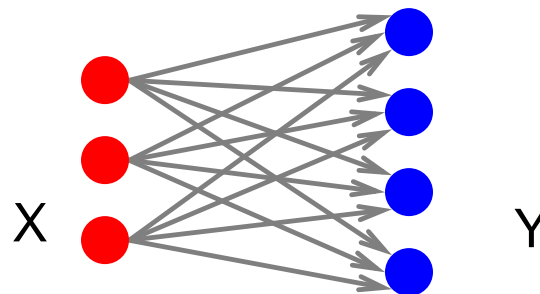


E.g. people talking about the same things or visited web pages

Problem definition:

Enumerate complete bipartite subgraphs $K_{s,t}$:

- All vertices in $K_{s,t}$ can be partitioned in two sets. Each vertex in the first set of size s is linked to each vertex in second set of size t
- Where $K_{s,t}$: s nodes on the “left” where each links to the same t other nodes on the “right”



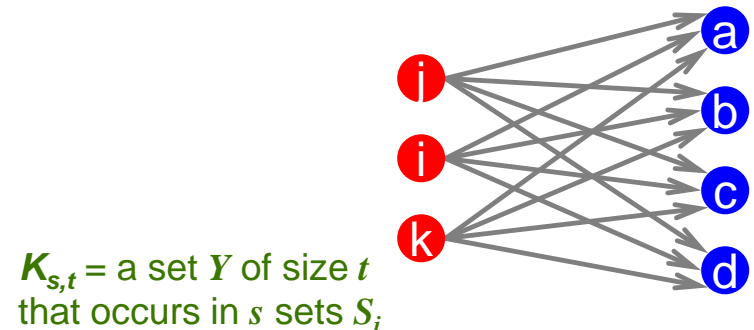
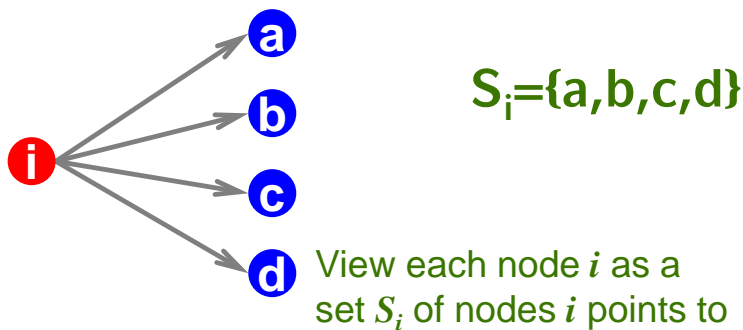
$$|X| = s = 3$$

$$|Y| = t = 4$$

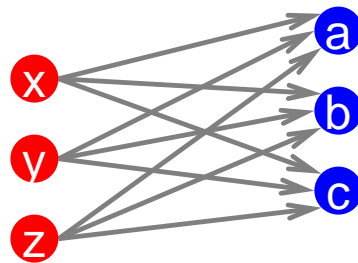
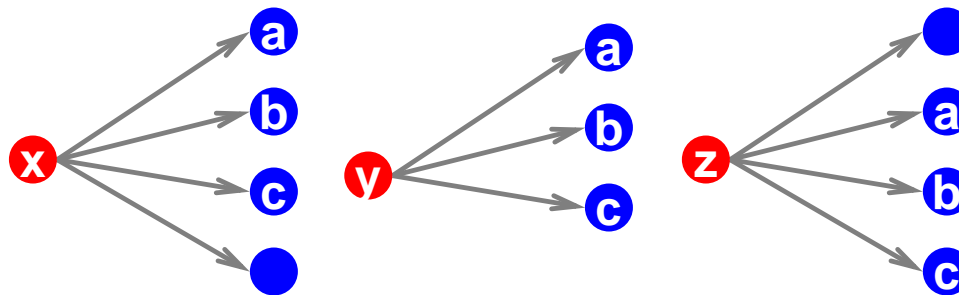
Frequent Itemset Analysis – Market Basket Analysis

- **Market:** Universe U of n items
- **Baskets:** subsets of U : $S_1, S_2, \dots, S_m \subseteq U$
 - (S_i is a set of items one person bought)
- **Support:** frequency threshold
- **Goal: Find all subsets T s.t. $T \subseteq S_i$ of at least f sets S_i**
 - (items in T were bought together at least f times)

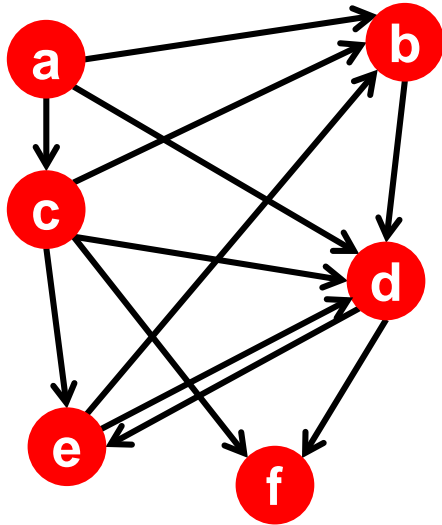
Frequent itemsets = complete bipartite graphs



E.g. Bipartite subgraph $K_{3,4}$ a **frequent itemset** $Y=\{a,b,c\}$ of **supp s** . So, there are s nodes that link to all of $\{a,b,c\}$:



We found $K_{s,t}$!
 $K_{s,t}$ = a set Y of size t
 that occurs in s sets S_i



Itemsets:

$a = \{b, c, d\}$

$b = \{d\}$

$c = \{b, d, e, f\}$

$d = \{e, f\}$

$e = \{b, d\}$

$f = \{\}$

Frequent itemsets support > 1

$\{b, d\}$: support 3

$\{e, f\}$: support 2

