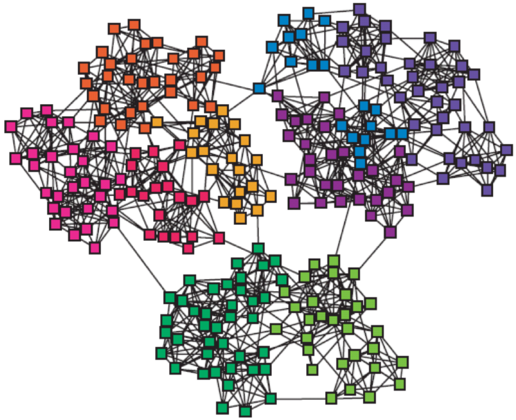
Mining Large Scale Datasets

Mining Network Graphs

(Adapted from CS246@Starford.edu; http://www.mmds.org)

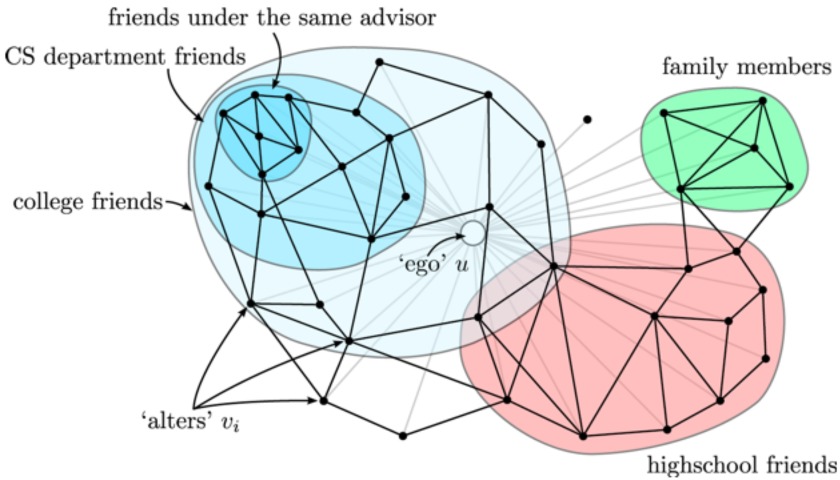
Sérgio Matos - aleixomatos@ua.pt

Networks



Analyze networks in terms of modules, cluster, communities

Example: Social circles

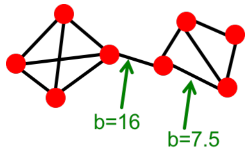


[McAuley, Leskovec: Discovering social circles in ego networks, 2012]

Class roadmap

- Betweenness and Girvan-Newman Algorithm
- Graph cuts
- Spectral graph partitioning

Betweenness

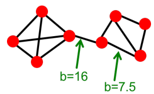


Betweenness of edge (a, b) is the number of pairs of nodes x, y such that the edge (a, b) lies on the shortest path between x and y

Since there can be several shortest paths between x and y, edge (a, b) is credited with the fraction of those shortest paths that include the edge (a, b)

Girvan-Newman Algorithm

Divisive hierarchical clustering based on the notion of edge **betweenness**: Number of shortest paths passing through an edge

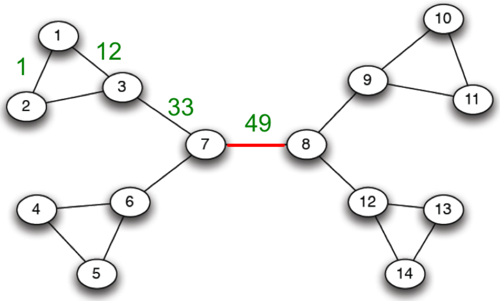


Process: repeat until no edges are left

- Calculate betweenness of edges
- Remove edges with highest betweenness

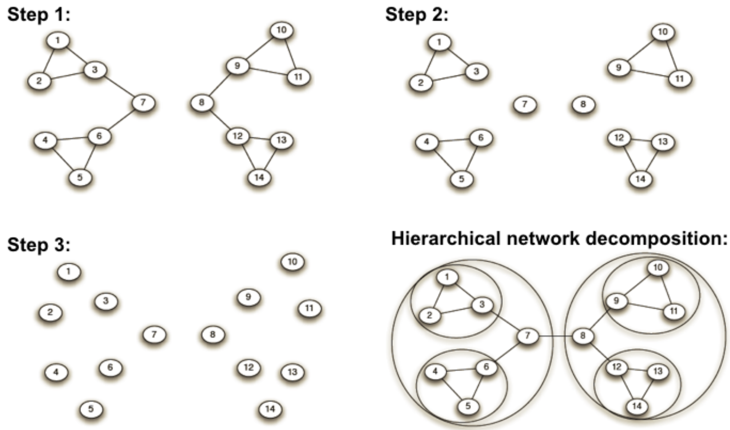
Connected components are communities
Gives a hierarchical decomposition of the network

Girvan-Newman Algorithm: Example

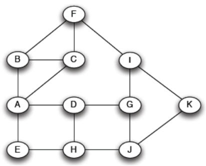


Note: betweenness is recomputed at every step.

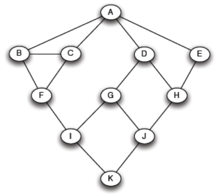
Girvan-Newman Algorithm: Example



Computing betweeness

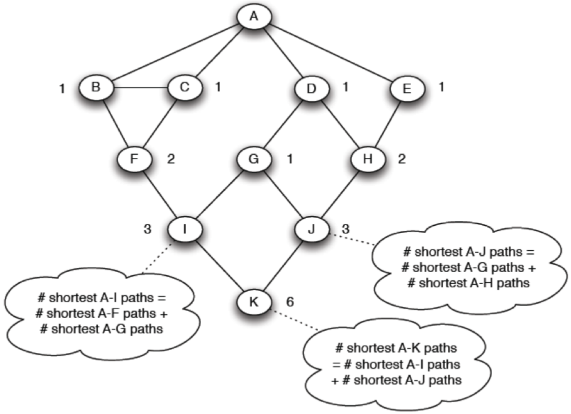


Calculate betweenness of paths starting at node A



Step 1: Breath first search starting from A

Computing betweeness

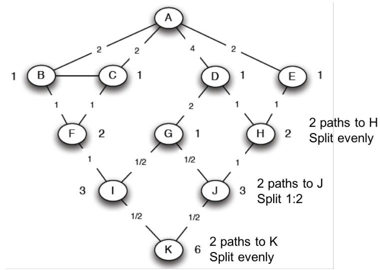


Step 2: Count the number of shortest paths from A to all other nodes of the network

Computing betweeness

Step 3: Compute betweenness by working up the tree

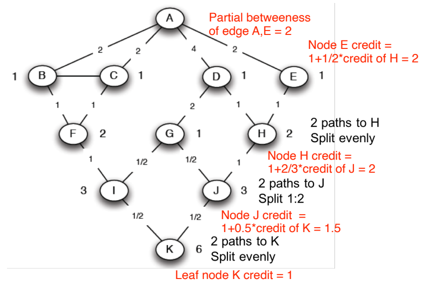
- Each leaf gets a credit of 1
- Non-leaf nodes get credit 1 plus the (weighted) sum of the credits of the edges to the level below
- Weights are defined by the relative number of shortest paths going through the node



Computing betweeness

Step 3: Compute betweenness by working up the tree

- Each leaf gets a credit of 1
- Non-leaf nodes get credit 1 plus the (weighted) sum of the credits of the edges to the level below
- Weights are defined by the relative number of shortest paths going through the node

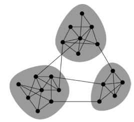


Repeat procedure for each starting node and sum the credits for each edge

Use subset of nodes as starting nodes to speed-up computation

Selecting 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

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

Modularity

For a partitioning S of graph G with n nodes and m edges

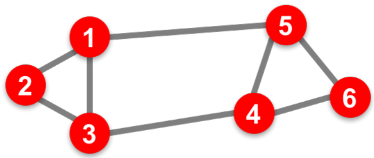
$$Q(G,S) = \frac{1}{2m} \sum_{s \in S} \sum_{i \in s} \sum_{j \in s} \left(A_{ij} - \frac{d_i d_j}{2m} \right)$$

$A_{ij} = 1$ if there is an edge from i to j
d_i degree of node i

Modularity takes values in range [-1,1]

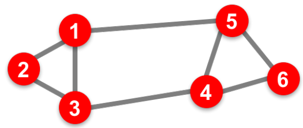
- Positive if the number of edges within groups exceeds the expected number
- $Q > 0.3..0.7$ means significant community structure

Spectral Clustering



Graph partitioning

Undirected graph $G(V, E)$

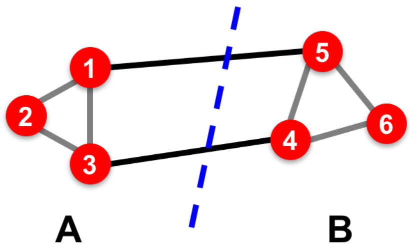


Bi-partitioning task:
Divide vertices into two disjoint groups

- How can we define a “good” partitioning of G?
- How can we efficiently identify such a partition?

Graph partitioning

What makes a good partition?



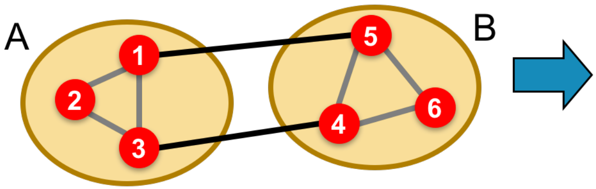
Maximize the number of within-cluster connections
Minimize the number of between-cluster connections

Graph cuts

Express partitioning quality in terms of the "edge cut" of the partition

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

$Cut(A, B) = \sum_{i \in A; j \in B} w_{ij}$ ($w_{ij} = 1$ for unweighted graphs)

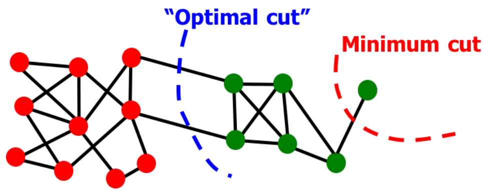


$cut(A,B) = 2$

Graph cut criteria: Minimum cut

Minimize weight of connections between groups

$$argmin_{A,B} cut(A, B)$$



Problem:

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

Graph cut criteria: Normalized cut

Connectivity between groups relative to the density of each group

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

$vol(A) = \sum_{i \in A} d_i$ $\quad$ ($d(i)$ degree of node i)

total weight of the edges with at least one endpoint in A

↪ Produces more balanced partitions

Graph cut criteria: Conductance

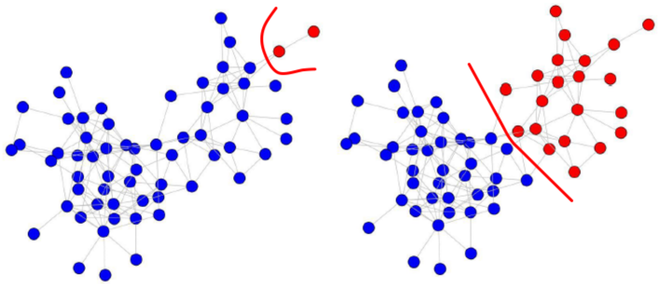
Connectivity of the group to the rest of the network relative to the density of the group

$$\phi(A) = \frac{cut(A)}{min(vol(A), 2m - vol(A))}$$

$vol(A) = \sum_{i \in A} d_i$ $\quad$ ($d(i)$ degree of node i)

total weight of the edges with at least one endpoint in A

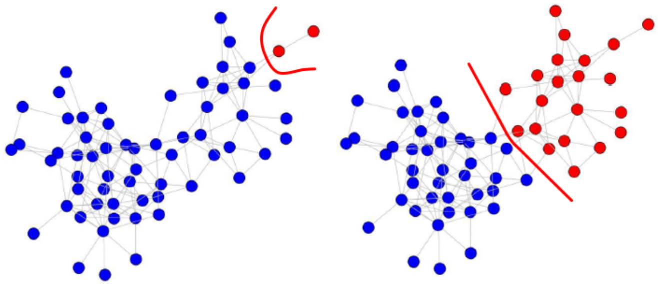
Graph cut criteria: Conductance



$\phi = 2/4 = 0.5$ $\qquad$ $\phi = 6/92 = 0.065$

$\hookrightarrow$ Produces more balanced partitions

Graph cut criteria: Conductance



$\phi = 2/4 = 0.5$ $\qquad$ $\phi = 6/92 = 0.065$

↪ Produces more balanced partitions

↪ **How do we efficiently find a good partition?**

Spectral Graph Partitioning

Consider

- A: adjacency matrix of undirected G
  $A_{ij} = 1$ if (i, j) if an edge, else 0
- x: vector in $\mathcal{R}^n$ with components $(x_1, \ldots, x_n)$
  Think of it as a label/value of each node of G

What is the meaning of $A \cdot x$?

Spectral Graph Partitioning

Consider

- A: adjacency matrix of undirected G
  $A_{ij} = 1$ if (i,j) if an edge, else 0
- x: vector in $\mathcal{R}^n$ with components $(x_1, \dots, x_n)$
  Think of it as a label/value of each node of G

What is the meaning of $A \cdot x$?

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & n_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad y_i = \sum_{j=1}^{n} A_{ij} x_j = \sum_{(i,j) \in E} x_j$$

Spectral Graph Partitioning

Consider

- A: adjacency matrix of undirected G
  $A_{ij} = 1$ if (i, j) if an edge, else 0
- x: vector in $\mathcal{R}^n$ with components $(x_1, \ldots, x_n)$
  Think of it as a label/value of each node of G

What is the meaning of $A \cdot x$?

$$\begin{bmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \ldots & n_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad y_i = \sum_{j=1}^{n} A_{ij} x_j = \sum_{(i,j) \epsilon E} x_j$$

Entry y_i is the sum of labels x_j of the neighbors of x_i

Spectral Graph Partitioning

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & n_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \lambda \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad A \cdot x = \lambda \cdot x$$

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

  $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$

Spectral Graph Theory

Consider a d-regular graph G

- All nodes have degree d and G is connected

$A \cdot x = \lambda \cdot x$

What are the eigenvectors of G

Spectral Graph Theory

Consider a d-regular graph G

- All nodes have degree d and G is connected

$A \cdot x = \lambda \cdot x$

What are the eigenvectors of G

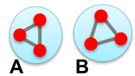
- Consider $x = (1, 1, ...1)$
- $A \cdot x = (d, d, ..., d) = \lambda \cdot x$
- So, $\lambda = d$

- We found the first *eigenpair* of G
  $x = (1, 1, ...1)$, $\lambda = d$

Spectral Graph Theory

What if G is not connected?

- G has 2 components, each d-regular



What are the eigenvectors of G

- x: vector of **1**s on **A** and **0**s on **B** (and vice versa)

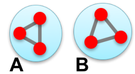
  $x' = (1, \ldots, 1, 0, \ldots, 0)$ then $A \cdot x' = (d, \ldots, d, 0, \ldots, 0)$

  $x'' = (0, \ldots, 0, 1, \ldots, 1)$ then $A \cdot x'' = (0, \ldots, 0, d, \ldots, d)$

- In both cases, $\lambda = d$

Spectral Graph Theory

What if G is not connected?

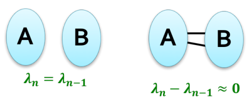


- G has 2 components, each d-regular

What are the eigenvectors of G

- x: vector of **1**s on **A** and **0**s on **B** (and vice versa)
  $x' = (1, \ldots, 1, 0, \ldots, 0)$ then $A \cdot x' = (d, \ldots, d, 0, \ldots, 0)$
  $x'' = (0, \ldots, 0, 1, \ldots, 1)$ then $A \cdot x'' = (0, \ldots, 0, d, \ldots, d)$
- In both cases, $\lambda = d$

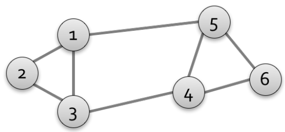
Intuition



second largest eigenvalue λ_2
now has value very close to λ_1

Spectral Graph Theory: Adjacency Matrix

- Adjacency matrix A
  - *nxn* matrix
  - $A = [a_{ij}]$, $a_{ij} = 1$ if edge between nodes i and j

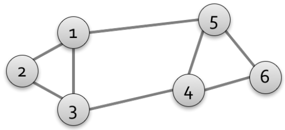


	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

- Symmetric matrix
- Eigenvectors are real and orthogonal

Spectral Graph Theory: Degree Matrix

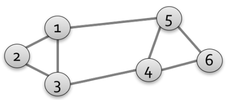
- Degree matrix D
  - nxn diagonal matrix
  - $D = [d_{ii}]$, d_{ii} = degree of node i



	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 Theory: Laplacian Matrix

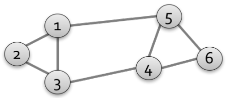
- Laplacian matrix L
  - *nxn* symmetric matrix
  - $L = D - A$



	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

Spectral Graph Theory: Laplacian Matrix

- Laplacian matrix L
  - *nxn* symmetric matrix
  - $L = D - A$

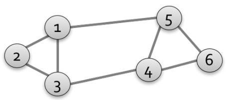


	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

- Trivial eigenpair ?

Spectral Graph Theory: Laplacian Matrix

- Laplacian matrix L
  - *nxn* 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
  - $x = (1, ..., 1)$ then $L \cdot x = 0$, so $\lambda = \lambda_1 = 0$

- Important properties
  - Eigenvalues are non-negative real numbers
  - Eigenvectors are real and orthogonal

λ_2 as an optimization problem

Fact: For symmetric matrix M

$$\lambda_2 = \min_x \frac{x^T Mx}{x^T x}$$

λ_2 as an optimization problem

Fact: For symmetric matrix M

$$\lambda_2 = \min_x \frac{x^T M x}{x^T x}$$

What is the meaning of $\min_x x^T L x$ for graph G?

$$x^T L x = \sum_{i,j=1}^{n} L_{ij} x_i x_j = \sum_{i,j=1}^{n} \left(D_{ij} - A_{ij}\right) x_i x_j$$

$$= \sum_i D_{ii} x_i^2 - \sum_{(i,j)\in E} 2 x_i x_j$$

$$= \sum_{(i,j)\in E} (x_i^2 + x_j^2 - 2 x_i x_j) = \sum_{(i,j)\in E} (x_i - x_j)^2$$

λ_2 as an optimization problem

What we know about x

- x is a unit vector: $\sum_i x_i^2 = 1$
- x is orthogonal to first eigenvector (1, ..., 1):

  $\sum_i x_i \cdot \mathbf{1} = \sum_i x_i = 0$

Note: x is the solution to the λ_2 eigenvector problem

λ_2 as an optimization problem

What we know about x

- x is a unit vector: $\sum_i x_i^2 = 1$
- x is orthogonal to first eigenvector $(1, \ldots, 1)$:

  $\sum_i x_i \cdot \mathbf{1} = \sum_i x_i = 0$

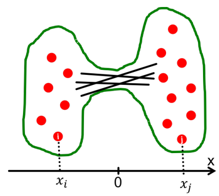
$$\lambda_2 = \min_x \frac{x^T L x}{x^T x} = \min_{x_i : \sum x_i = 0} \frac{\sum_{(i,j) \in E} (x_i - x_j)^2}{\sum_i x_i^2}$$

λ_2 as an optimization problem

What we know about x

- x is a unit vector: $\sum_i x_i^2 = 1$
- x is orthogonal to first eigenvector $(1, \ldots, 1)$:
  $\sum_i x_i \cdot \mathbf{1} = \sum_i x_i = 0$

$$\lambda_2 = \min_x \frac{x^T L x}{x^T x} = \min_{x_i : \sum x_i = 0} \frac{\sum_{(i,j)\in E} (x_i - x_j)^2}{\sum_i x_i^2}$$



Forces values x_i for nodes i such that few edges cross 0
Forces x_i and x_j to subtract each other

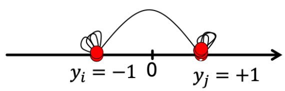
Relation to finding the optimal cut

- Express partition (A,B) as a vector

$$y_i = \begin{cases} +1, & \text{if } i \in A \\ -1, & \text{if } i \in B \end{cases}$$

- Minimize the cut of the partition by finding a vector that minimizes

$$\underset{y \in [-1,+1]^n}{\operatorname{argmin}} f(y) = \sum_{(i,j) \in E} (y_i - y_j)^2$$



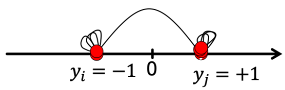
Relation to finding the optimal cut

- Express partition (A,B) as a vector

$$y_i = \begin{cases} +1, & \text{if } i \in A \\ -1, & \text{if } i \in B \end{cases}$$

- Minimize the cut of the partition by finding a vector that minimizes

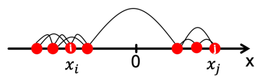
$$\underset{y \in [-1,+1]^n}{\operatorname{argmin}} f(y) = \sum_{(i,j) \in E} (y_i - y_j)^2$$



Can be solved if we relax y to allow any real value

Finding the optimal cut: Rayleigh theorem

$$\min_{y \in R^n} f(y) = \sum_{(i,j) \in E} (y_i - y_j)^2 = y^T L y$$



- $\lambda_2 = \min_y f(y)$
  The minimum value of $f(y)$ is given by the 2$^{\text{nd}}$ smallest eigenvalue λ_2 of the Laplacian matrix $\mathbf{L}$
- $x = \operatorname{argmin}_y f(y)$
  The optimal solution for y is given by the corresponding eigenvector x, referred as the Fiedler vector

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 to a lower-dimensional representation based on one or more eigenvectors

3) Grouping

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

Spectral Partitioning Algorithm

1) Pre-processing

   Build Laplacian matrix L of the graph G

2) Decomposition

   Find eigenvalues λ and eigenvectors x of L

   Map vertices of G to corresponding components of λ_2

	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

λ=

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

Spectral Partitioning Algorithm

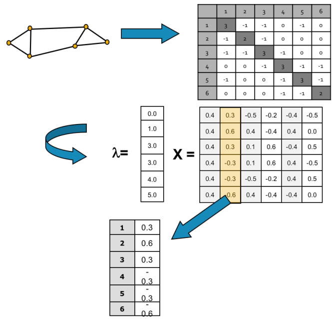
1) Pre-processing

   Build Laplacian matrix L of the graph G

2) Decomposition

   Find eigenvalues λ and eigenvectors x of L

   Map vertices of G to corresponding components of λ_2



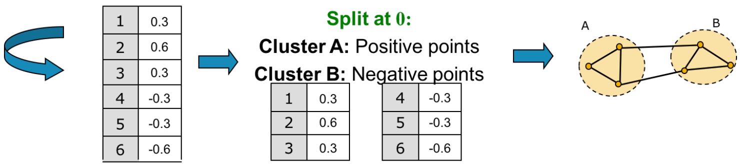
$\hookrightarrow$ How to find the clusters?

Spectral Partitioning Algorithm

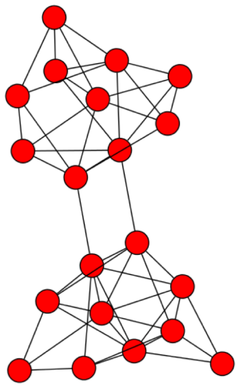
3) Grouping
   Sort components of reduced 1-dimensional vector
   Identify clusters by splitting the sorted vector in two

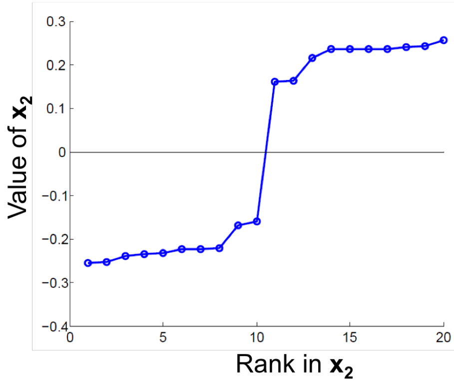
Select the splitting point

- Split at **0** or median value
- More expensive approach
  - Sweep over ordering of nodes induced by the eigenvector and attempt to minimize the normalized cut in 1 dimension

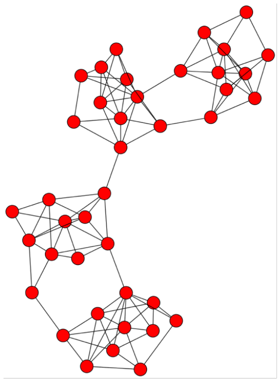


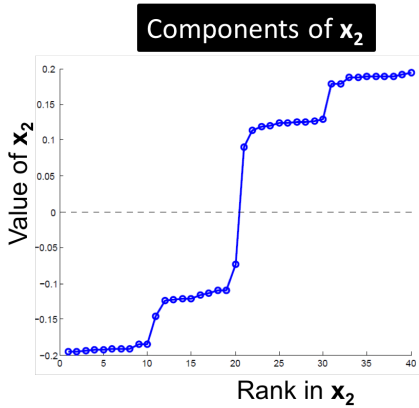
Spectral partitioning: example



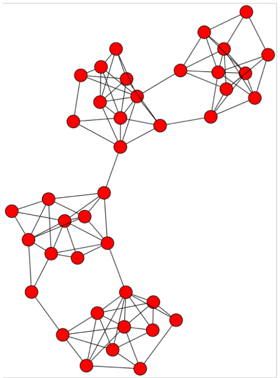


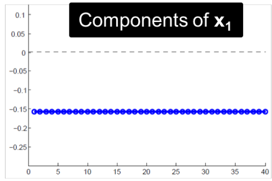
Spectral partitioning: example

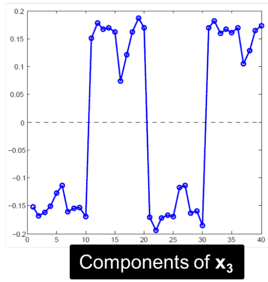




Spectral partitioning: example







k-way spectral partitioning

- How do we partition a graph into k clusters?
- Two basic approaches:
  - Recursive bi-partitioning [Hagen et al., '92]
    Recursively apply bi-partitioning algorithm in a hierarchical divisive manner
    Disadvantages: Inefficient, unstable
  - Cluster using multiple eigenvectors [Shi-Malik, '00]
    Build a reduced space from multiple eigenvectors
    Commonly used in recent papers; better results