Community Detection: Graph Cuts & Spectral Clustering

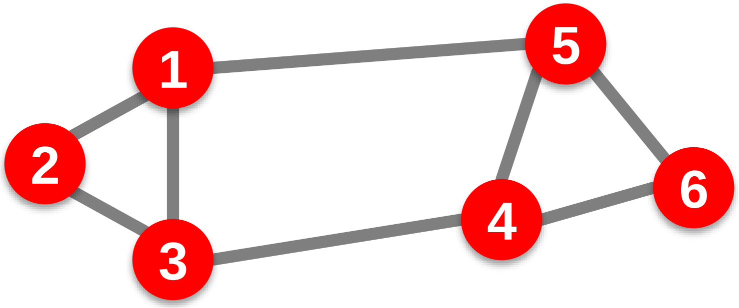
Thanks to Jure Leskovec, Stanford and Panayiotis Tsaparas, Univ. of Ioannina for slides

Agenda

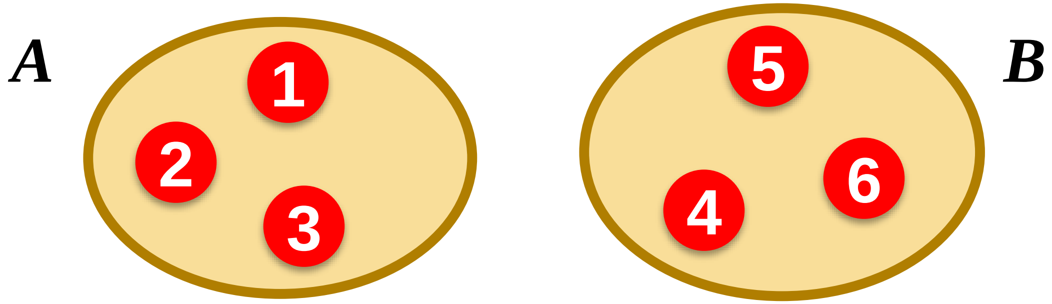
- Graph Partitioning
  - Graph Cuts
  - Spectral Clustering

Graph Partitioning

- **Undirected graph $G(V, E)$:**



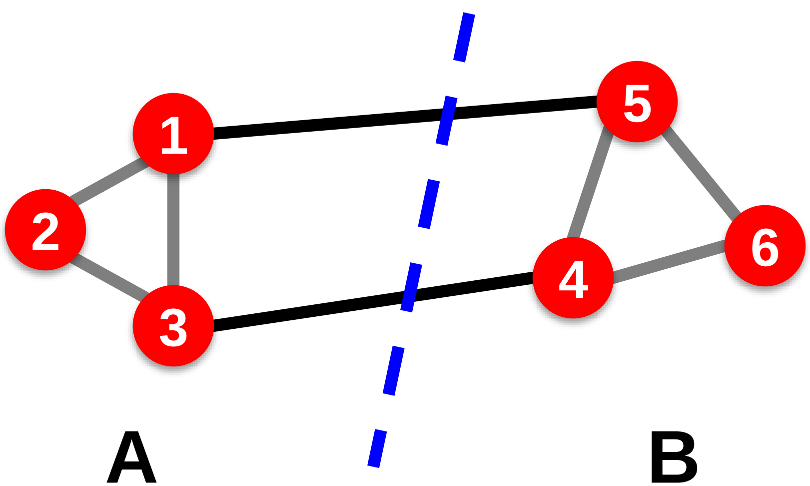
- **Bi-partitioning task:**
  - Divide vertices into two disjoint groups A, B



- **Questions:**
  - How can we define a "good" partition of G?
  - How can we efficiently identify such a partition?

Graph Partitioning

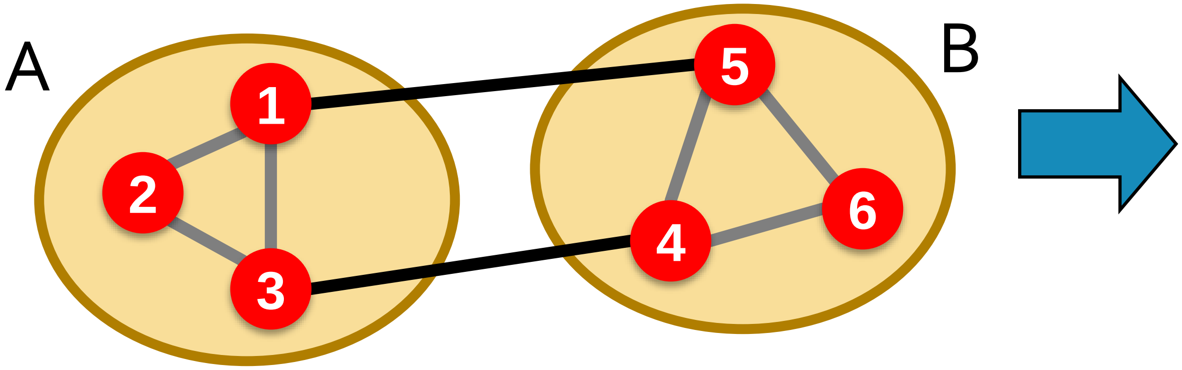
- **What makes a good partition?**
  - Maximize the number of within-group connections
  - Minimize the number of between-group connections



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:

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



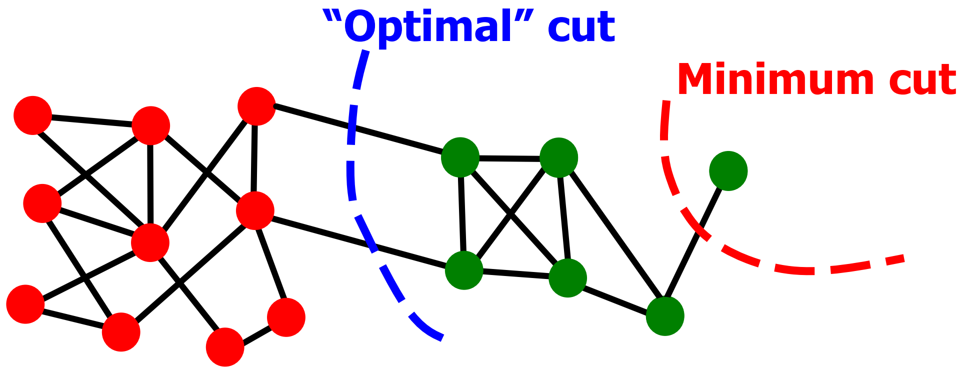
$cut(A,B) = 2$

Graph Cut Criterion

- **Criterion: Minimum-cut**
  - Minimize weight of connections between groups

$$\arg\min_{A,B} cut(A,B)$$

- **Degenerate case:**



- **Problem:**
  - Only considers external cluster connections
  - Does not consider internal cluster connectivity

Graph Bisection

- Since the minimum cut does not always yield good results we need extra constraints to make the problem meaningful
- **Graph Bisection**
  - Partition the graph into two *equal sets* of nodes
- **Kernighan-Lin algorithm**
  - Start with random equal partitions
  - Swap nodes to improve some quality metric (e.g., cut, modularity, etc)

Ratio Cut

Criterion: Ratio-cut

Normalize cut by the *size* of the groups

$$\textbf{Ratio-cut} = \frac{\mathrm{Cut(U,V-U)}}{|U|} + \frac{\mathrm{Cut(U,V-U)}}{|V-U|}$$

Normalized Cut

Criterion: **Normalized-cut**

Connectivity between groups relative to the *density* of each group

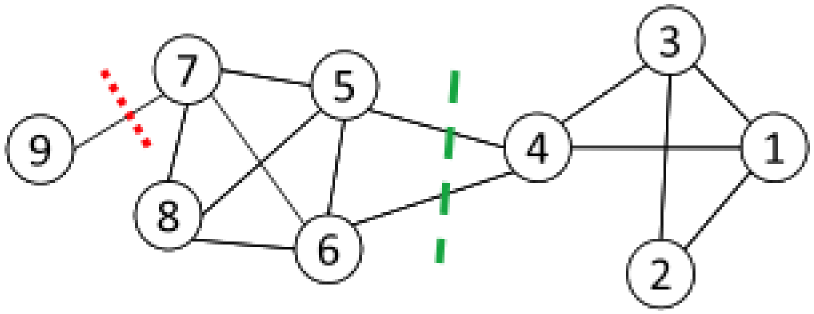
$$\textbf{Normalized-cut} = \frac{\mathrm{Cut(U,V-U)}}{Vol(U)} + \frac{\mathrm{Cut(U,V-U)}}{Vol(V-U)}$$

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

Why use these criteria?

- Produce more balanced partitions

An Example of Min Cut Options



Cut(Red) = 1

Cut(Green) = 2

Ratio-Cut(Red) = $\frac{1}{1} + \frac{1}{8} = \frac{9}{8}$

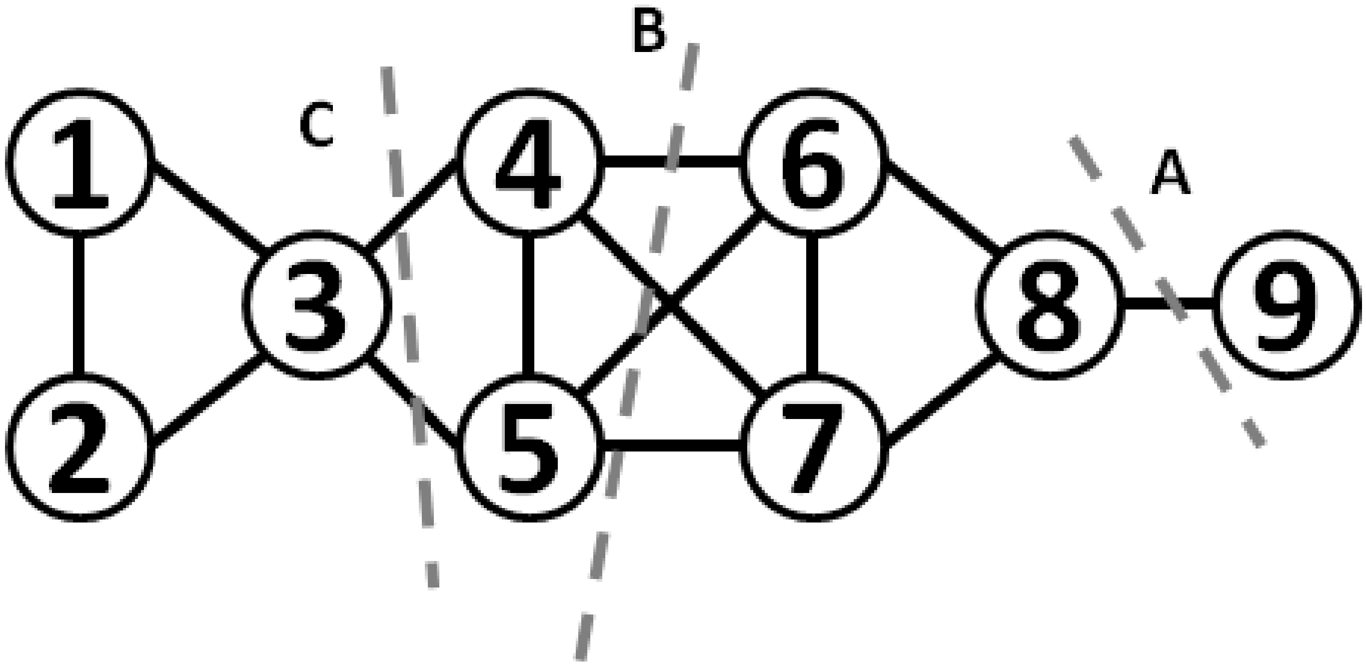
Ratio-Cut(Green) = $\frac{2}{5} + \frac{2}{4} = \frac{18}{20}$

Normalized-Cut(Red) = $\frac{1}{1} + \frac{1}{27} = \frac{28}{27}$

Normalized-Cut(Green) = $\frac{2}{12} + \frac{2}{16} = \frac{14}{48}$

Minimizing **Normalized-cut** is even better for Green due to density constraint (volume)

Another Example



Which of the three cuts has the best (min, normalized, ratio) cut?

Graph Cut Criteria

- **Criterion: Conductance** [Shi-Malik, '97]
  - Connectivity between groups relative to the density of each group

$$\phi(A,B) = \frac{cut(A,B)}{\min(vol(A), vol(B))}$$

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

- **Why use this criterion?**
  - Produces more balanced partitions
- **How do we efficiently find a good partition?**
  - **Problem:** Computing optimal cut is NP-hard

Graph Cuts

- **Ratio-cut** and **normalized-cut** can be reformulated in matrix format and solved using **spectral clustering**

Spectral Clustering for Graph Partitioning

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
- But first, let’s define the problem

Spectral Graph Partitioning

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

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

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

- **Entry y_i is a sum of labels x_j of neighbors of i**

Spectral Graph Theory

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

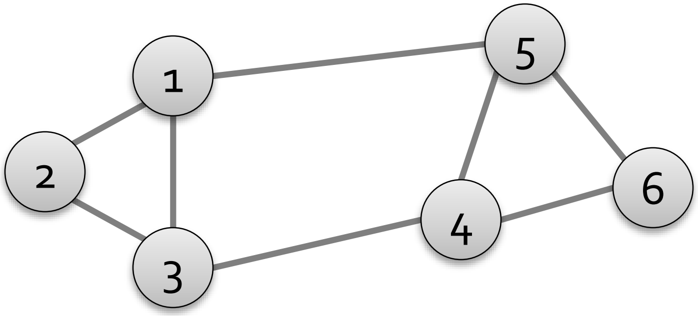
$$\boldsymbol{A \cdot x = \lambda \cdot x}$$

- **Spectral Graph Theory:**
  - Analyze the "spectrum" of matrix representing $\boldsymbol{G}$
  - **Spectrum:** Eigenvectors $\boldsymbol{x_i}$ of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues $\boldsymbol{\lambda_i}$: $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$

    Note: We sort $\boldsymbol{\lambda_i}$ in ascending (not descending) order!
  - **Spectral clustering**: use the eigenvectors of $\boldsymbol{A}$ or graphs derived by it (mostly **graph Laplacian**)

Matrix Representations

- **Adjacency matrix (A):**
  - $n \times n$ matrix
  - $A=[a_{ij}]$, $a_{ij}=1$ if edge between node i and j

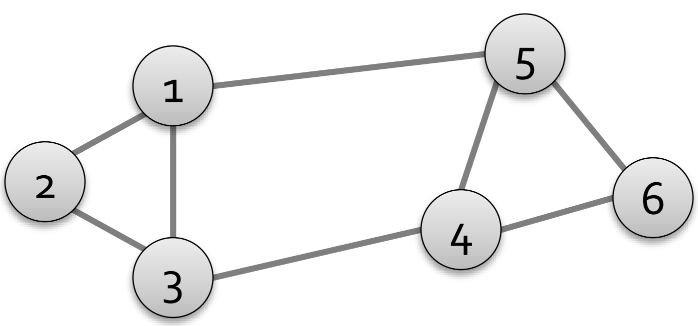


	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

- **Important properties:**
  - Symmetric matrix
  - Eigenvectors are real and orthogonal

Matrix Representations

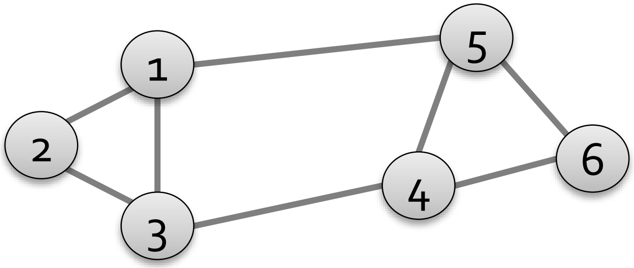
- **Degree matrix (D):**
  - $n \times n$ diagonal matrix
  - $D=[d_{ii}]$, d_{ii} = degree of node i



	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

Matrix Representations

- **Laplacian matrix (L):**
  - $n \times n$ symmetric matrix

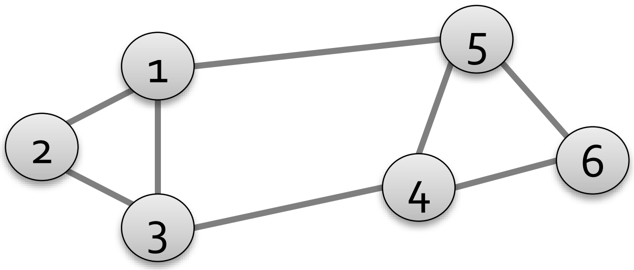


	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

$$\boxed{L = D - A}$$

- **Laplacian matrix L important properties:**
  - **Eigenvalues** are non-negative real numbers
  - **Eigenvectors** are real and orthogonal

Example: Eigenvalues & Eigenvectors



Eigenvalue	0	1	3	3	4	5
Eigenvector	1	1	-5	-1	-1	-1
	1	2	4	-2	1	0
	1	1	1	3	-1	1
	1	-1	-5	-1	1	1
	1	-2	4	-2	-1	0
	1	-1	1	3	1	-1

The Smallest Eigenvalue

- **What is a trivial eigenpair?**
  - $\boldsymbol{x} = (\mathbf{1}, \dots, \mathbf{1})$ then $\boldsymbol{L} \cdot \boldsymbol{x} = \mathbf{0}$ and so $\boldsymbol{\lambda} = \boldsymbol{\lambda}_{\mathbf{1}} = \mathbf{0}$
  - $\boldsymbol{\lambda_1 = 0}$ is the smallest eigenvalue

The Second Smallest Eigenvalue

- The second smallest eigenvalue (also known as Fielder value) λ_2 satisfies

$$\lambda_2 = \min_{x \perp w_1, \|x\|=1} x^T L x$$

- For the Laplacian, it is:

$$x \perp w_1 \Rightarrow \sum_i x_i = 0$$

$$x^T L x \Rightarrow \sum_{(i,j)\in E} \left(x_i - x_j\right)^2$$

The Second Smallest Eigenvalue

Thus, the eigenvector for eigenvalue λ_2 (called the Fielder vector) minimizes

$$\min_{x \neq 0} \sum_{(i,j) \in E} (x_i - x_j)^2 \quad \text{where} \quad \sum_i x_i = 0$$

Intuitively:

- minimum when *x_i and x_j close whenever there is an edge between nodes **i** and **j*** in the graph
- ***x*** must have *some positive* and *some negative* components

Cuts + Eigenvalues: Intuition

- A ***partition*** of the graph by taking:
  - one set to be the nodes **i** whose corresponding vector component $\mathbf{x_i}$ is ***positive*** and
  - the other set to be the **j** nodes whose corresponding vector component $\mathbf{x_j}$ is ***negative***.

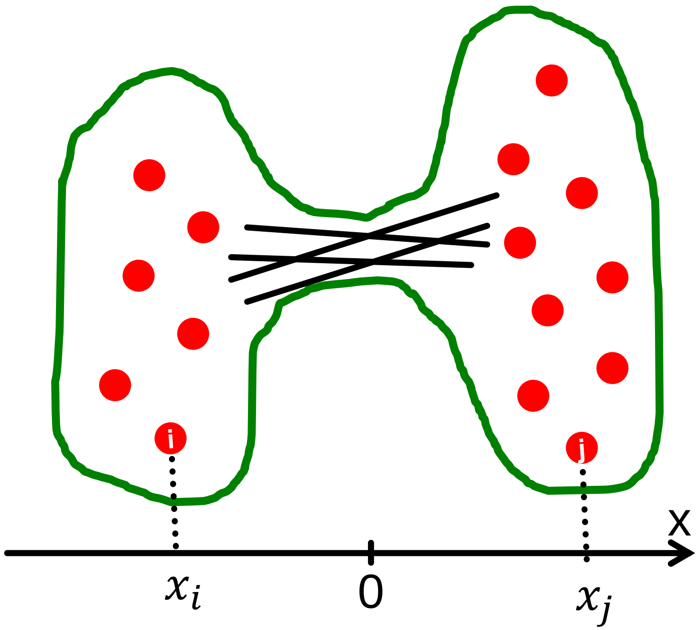
- The ***cut*** between the two sets will have a small number of edges because $(x_i - x_j)^2$ is likely to be **smaller** if both $\mathbf{x_i}$ and $\mathbf{x_j}$ have the **same sign** than if they have different signs.

- Thus, ***minimizing*** $x^T L x$ under the required constraints will end giving $\mathbf{x_i}$ and $\mathbf{x_j}$ the same sign if there is an edge **(i, j)**.

Cuts + Eigenvalues: Summary

- What we know about x?
  - x is unit vector: $\sum_i x_i^2 = 1$
  - x is orthogonal to 1st eigenvector $(1, \dots, 1)$ thus: $\sum_i x_i \cdot 1 = \sum_i x_i = 0$

$$\lambda_2 = \min_{\substack{\text{All labelings} \\ \text{of nodes } i \text{ so} \\ \text{that } \sum x_i = 0}} \frac{\sum_{(i,j)\in E} (x_i - x_j)^2}{\sum_i x_i^2}$$



Balance to minimize

We want to assign values x_i to nodes i such that few edges cross 0.
(we want x_i and x_j to subtract each other)

So far...

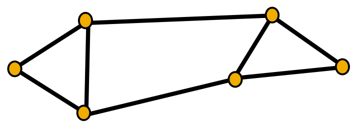
- **How to define a "good" partition of a graph?**
  - Minimize a given graph cut criterion
- **How to efficiently identify such a partition?**
  - Approximate using information provided by the eigenvalues and eigenvectors of a graph
- **Spectral Clustering**

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



	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 corresponding components of λ_2

$\lambda=$

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 Partitioning

- 3) **Grouping:**
  - Sort components of reduced 1-dimensional vector
  - Identify clusters by splitting the sorted vector in two
- **How to choose a splitting point?**
  - Naïve approaches:
    - Split at **0** or median value
  - More expensive approaches:
    - Attempt to minimize normalized cut in 1-dimension (sweep over ordering of nodes induced by the eigenvector)

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

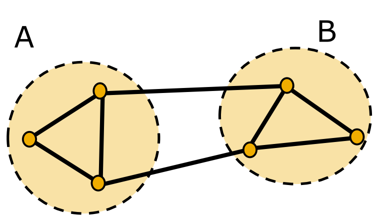
Split at 0:

Cluster A: Positive points

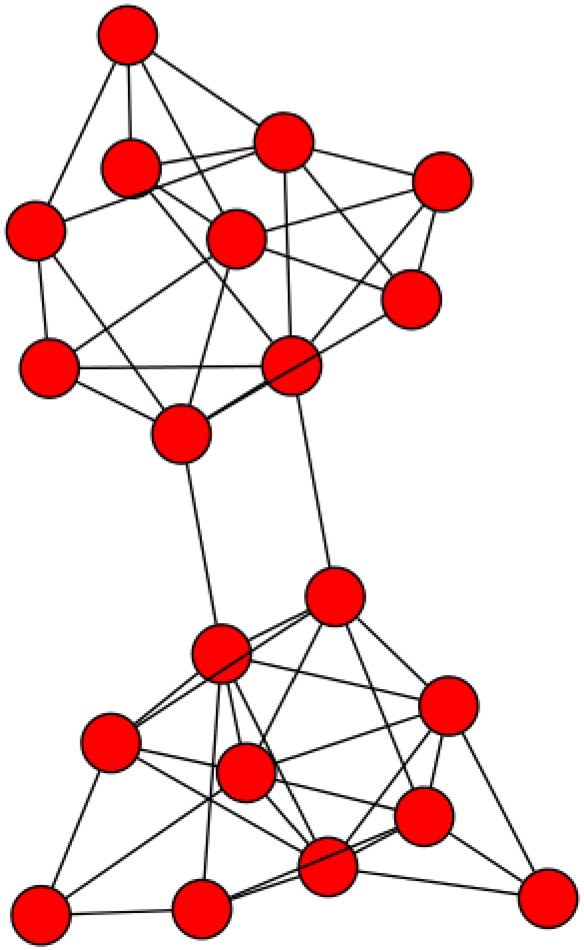
Cluster B: Negative points

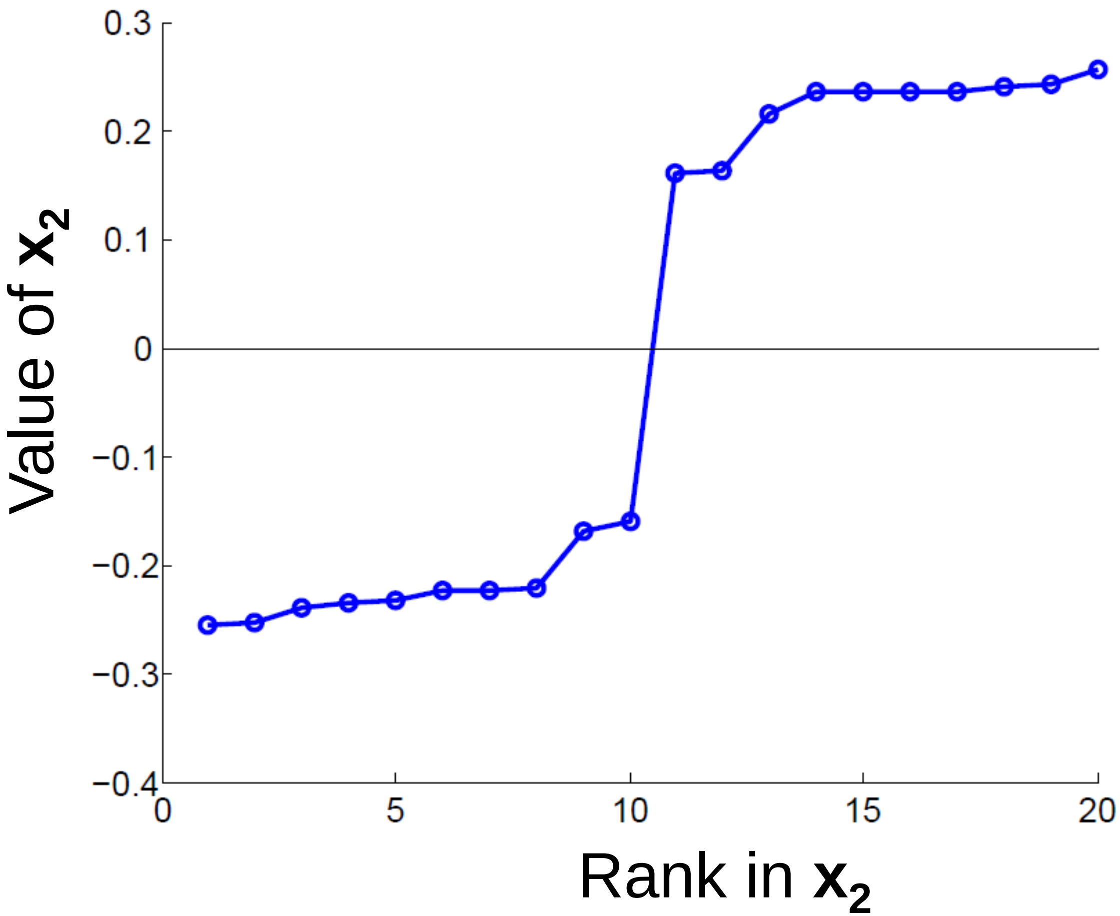
1	0.3
2	0.6
3	0.3

4	-0.3
5	-0.3
6	-0.6

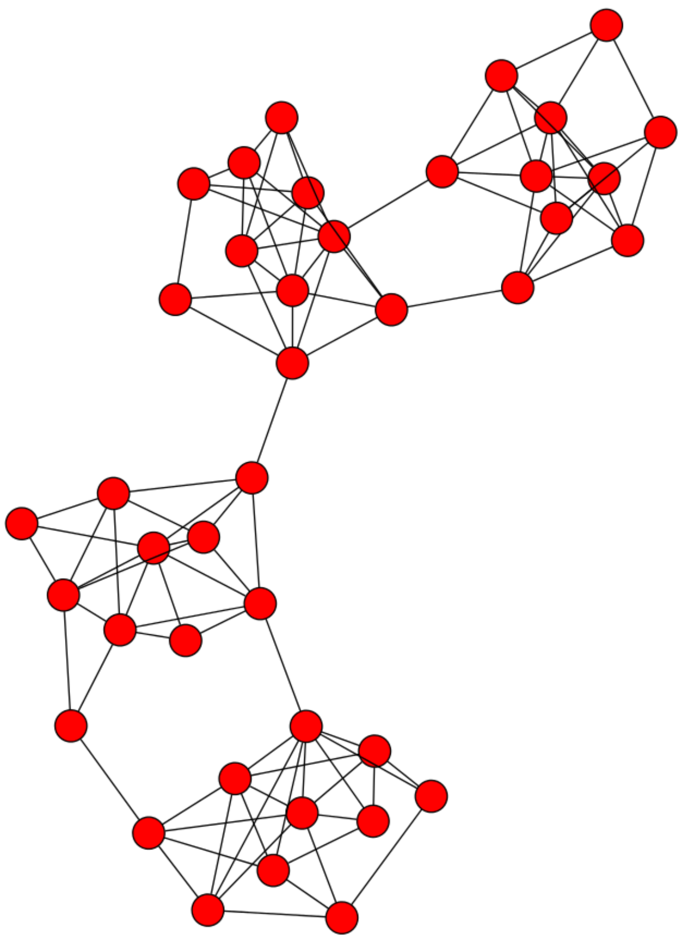


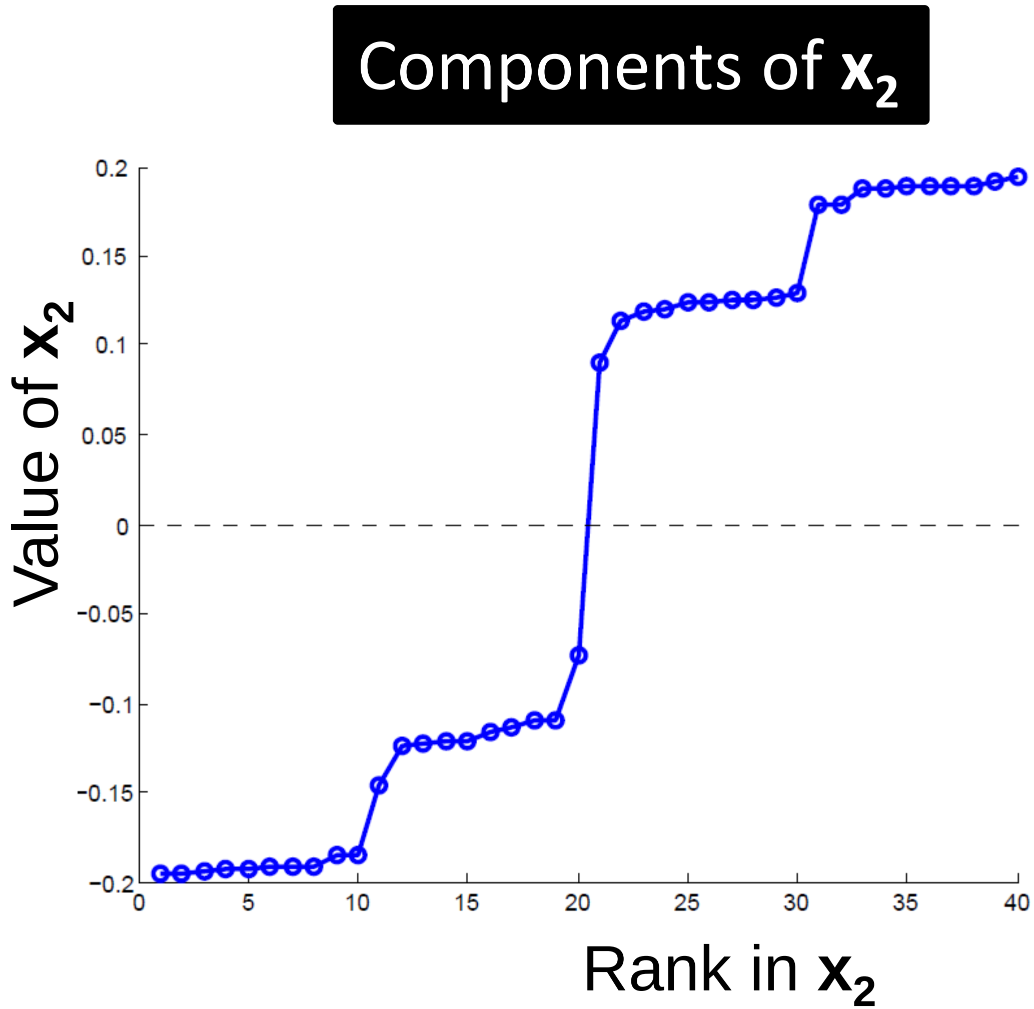
Example: Spectral Partitioning



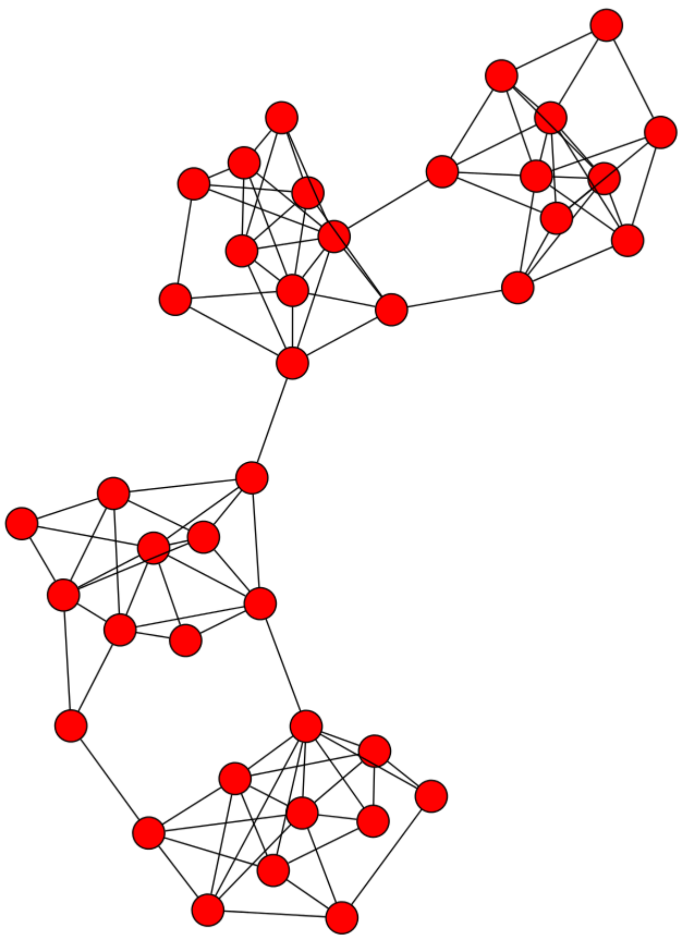


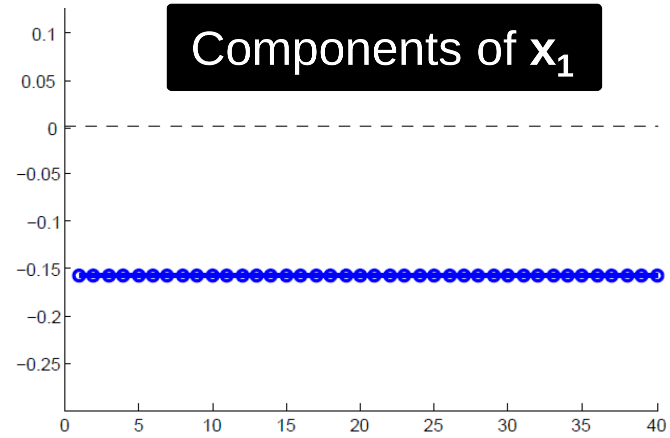
Example: Spectral Partitioning

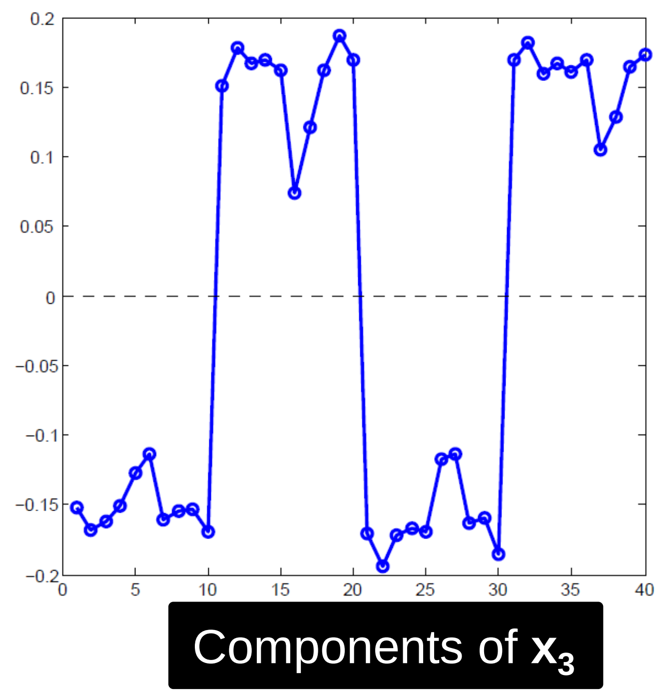




Example: Spectral Partitioning



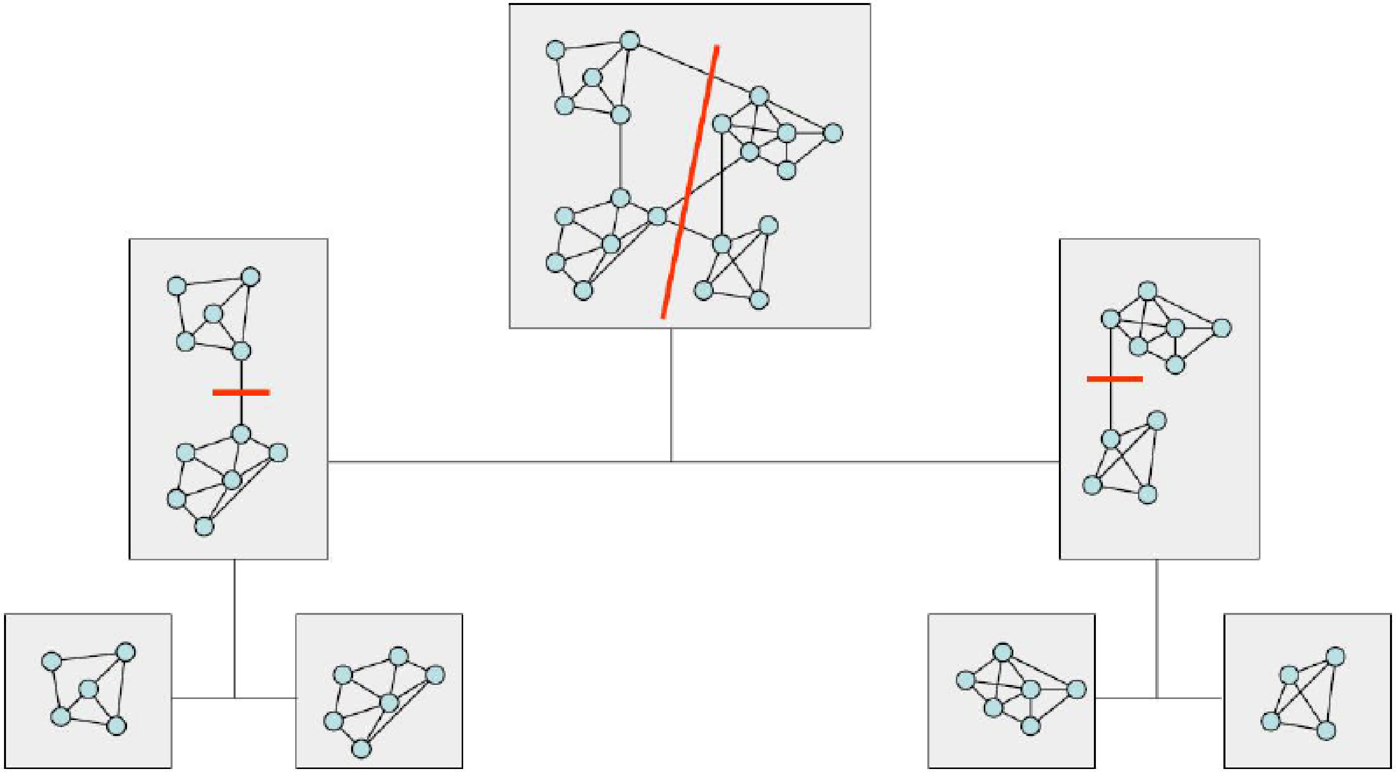




k-Way Spectral Clustering

- **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 multiple eigenvectors** [Shi-Malik, '00]
    - Build a reduced space from multiple eigenvectors
    - Commonly used in recent papers
    - A preferable approach…

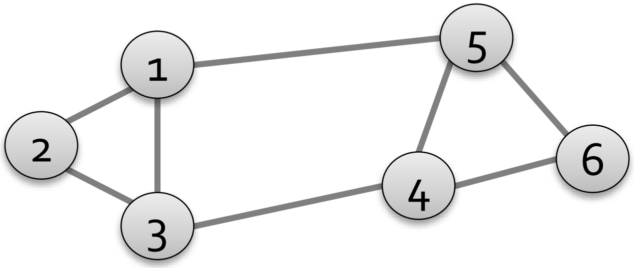
Recursive Bi-partitioning



Cluster Multiple Eigenvectors

- Use ***several of the eigenvectors*** to partition the graph
- If we use **m** eigenvectors, and set a threshold for each, we can get a partition into **2^m groups**, each group consisting of the nodes that are above or below threshold for each of the eigenvectors, in a particular pattern.

Example



Eigenvalue	0	1	3	3	4	5
Eigenvector	1	1	-5	-1	-1	-1
	1	2	4	-2	1	0
	1	1	1	3	-1	1
	1	-1	-5	-1	1	1
	1	-2	4	-2	-1	0
	1	-1	1	3	1	-1

If we use both the **2nd** and **3rd** eigenvectors:

- nodes **2** and **3** (positive in both)
- nodes **5** and **6** (negative in 2nd, positive in 3rd)
- nodes **1** and **4** alone

Note that while each eigenvector tries to produce a minimum-sized cut, successive eigenvectors have to satisfy more and more constraints => the cuts progressively worse.

Why use multiple eigenvectors?

- **Approximates the optimal cut** [Shi-Malik, '00]
  - Can be used to approximate optimal k-way normalized cut
- **Emphasizes cohesive clusters**
  - Increases the unevenness in the distribution of the data
  - Associations between similar points are amplified, associations between dissimilar points are attenuated
  - The data begins to "approximate a clustering"
- **Well-separated space**
  - Transforms data to a new "embedded space", consisting of $\boldsymbol{k}$ orthogonal basis vectors
- Multiple eigenvectors prevent instability due to information loss

Many Other Partitioning Methods

- **METIS:**
  - Heuristic but works really well in practice
  - http://glaros.dtc.umn.edu/gkhome/views/metis
- **Graclus:**
  - Based on kernel k-means
  - http://www.cs.utexas.edu/users/dml/Software/graclus.html
- **Louvain:**
  - Based on Modularity optimization
  - http://perso.uclouvain.be/vincent.blondel/research/louvain.html
- **Clique percolation method:**
  - For finding overlapping clusters
  - http://angel.elte.hu/cfinder/

Spectral Clustering

- Use the lowest ***k*** eigenvalues of ***L*** to construct the ***nxk*** graph ***G'*** that has these eigenvectors as columns

- *The **n-rows** represent the graph vertices in a **k**-dimensional Euclidean space*

- Group these vertices in ***k*** clusters using ***k***-means clustering or similar techniques

Summary

- The values of x minimize

$$\min_{x \neq 0} \sum_{(i,j)\in E} (x_i - x_j)^2 \qquad \sum_i x_i = 0$$

- For weighted matrices

$$\min_{x \neq 0} \sum_{(i,j)} A[i,j](x_i - x_j)^2 \qquad \sum_i x_i = 0$$

- The ordering according to the x_i values will group similar (connected) nodes together

- Physical interpretation: The stable state of springs placed on the edges of the graph

Spectral Clust. (besides graphs)

- Can be used to cluster any points (not just vertices), as long as an appropriate similarity matrix
- Needs to be symmetric and non-negative
- How to construct a graph:
  - ε-neighborhood graph: connect all points whose pairwise distances are smaller than ε
  - k-nearest neighbor graph: connect each point with each k nearest neighbor
  - full graph: connect all points with weight in the edge (i, j) equal to the similarity of i and j