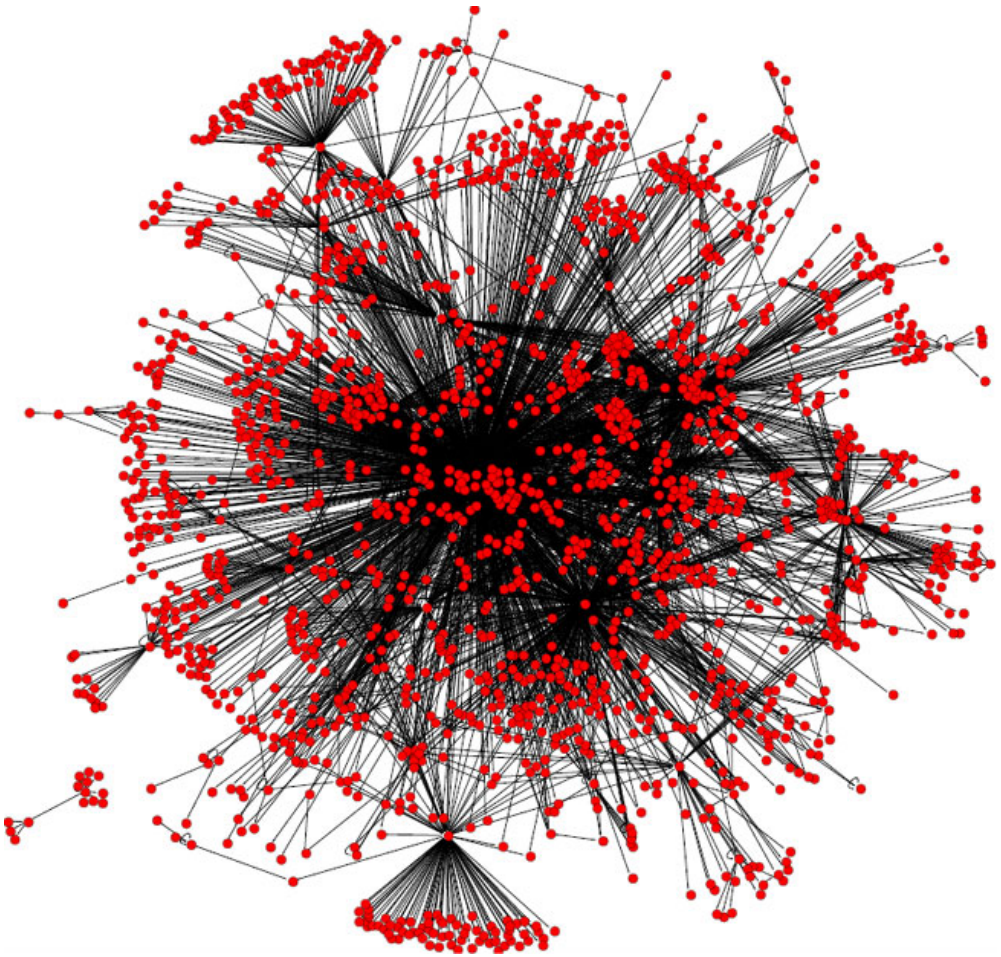
Community Detection[1]

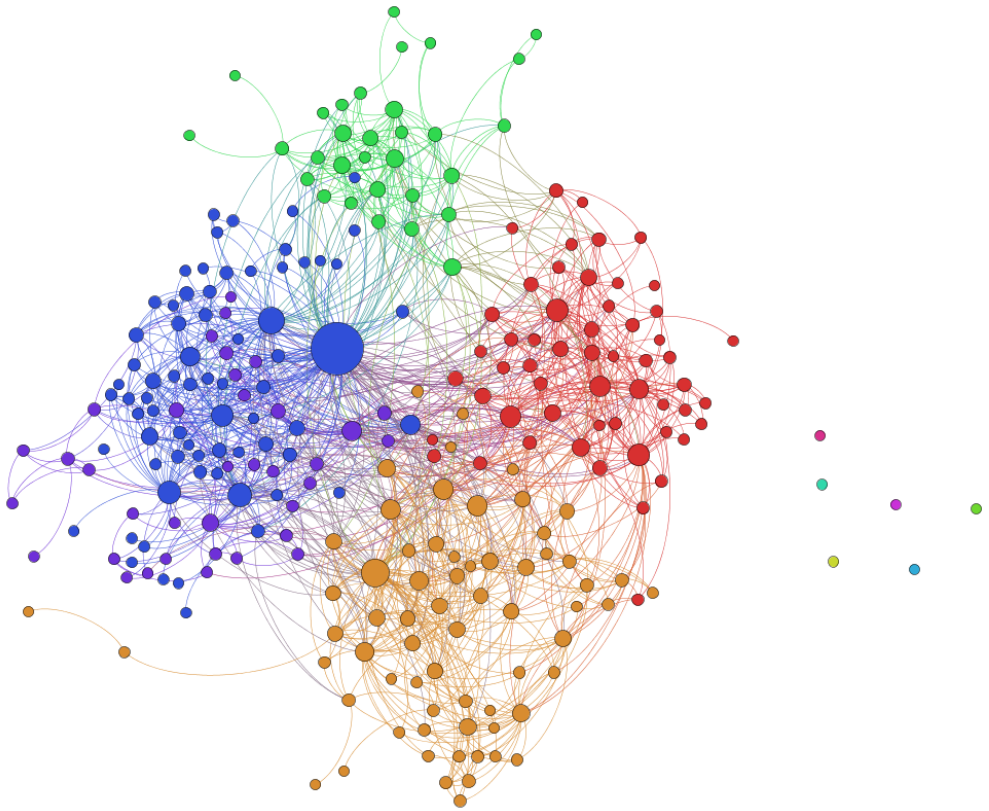
[1] Based on materials in http://www.mmds.org/

Motivation: gene regulatory networks



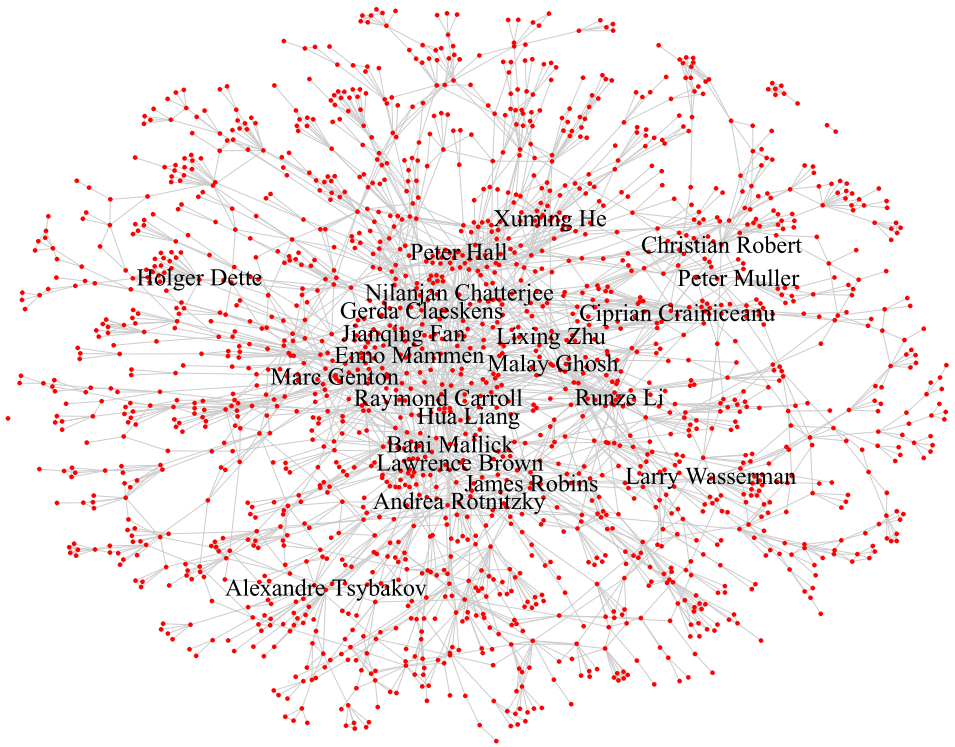
Nodes are genes, and edges represent regulatory interactions.

Motivation: friendship networks



Nodes are Facebook users, and edges represent friendship ties.

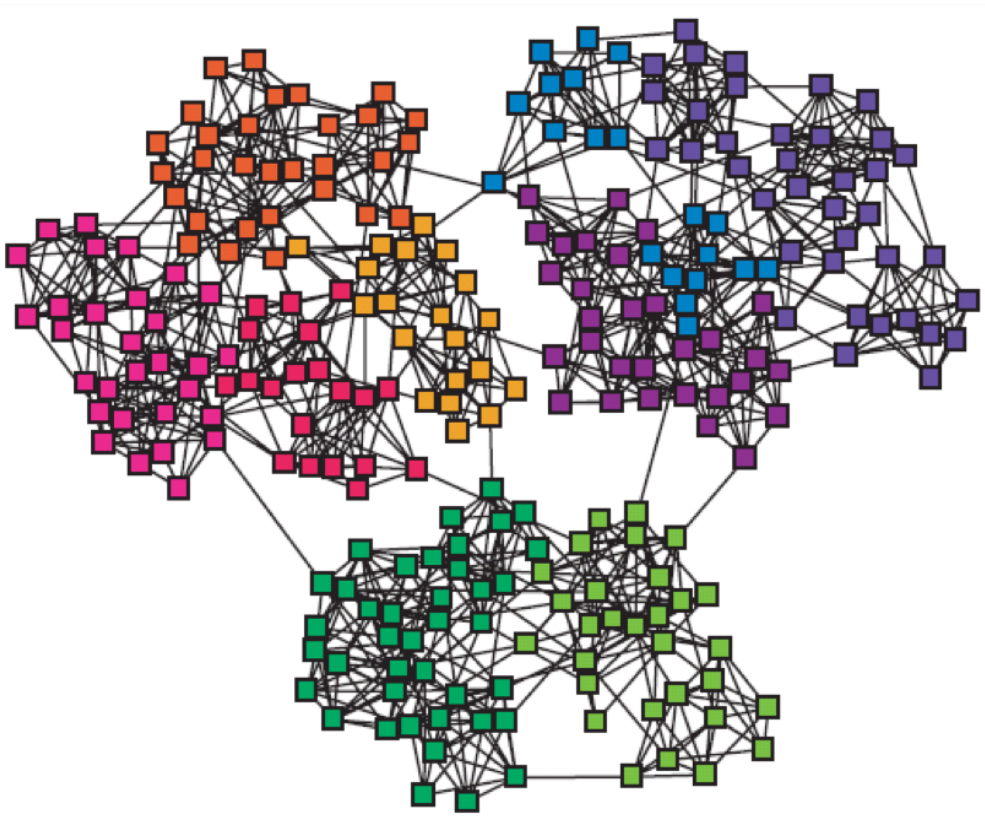
Motivation: co-authorship networks for statisticians



Nodes are statisticians and edges represent coauthorships. Names are shown for nodes with degree of 18 or larger.

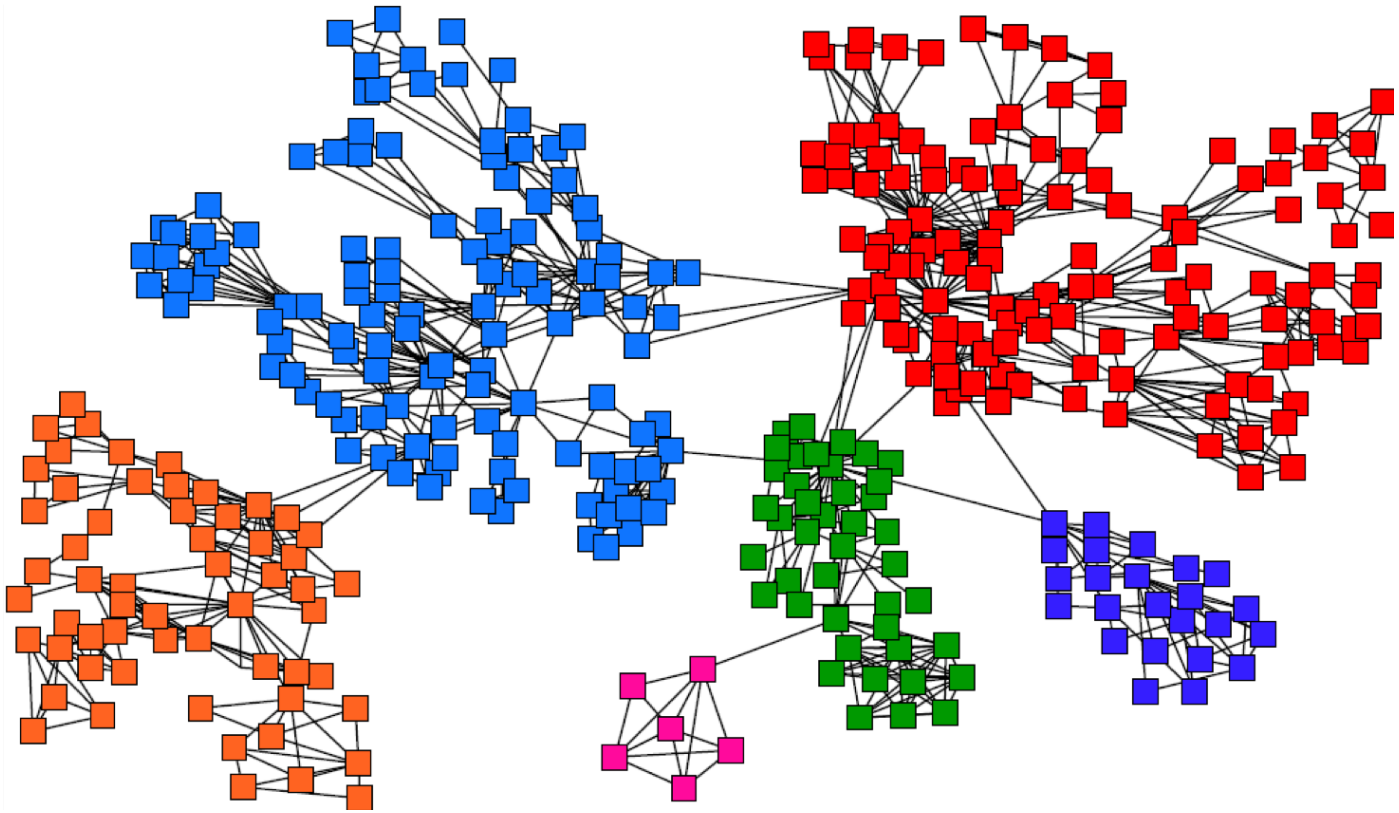
Communities

We often think of networks being organized into **cluster or communities**:



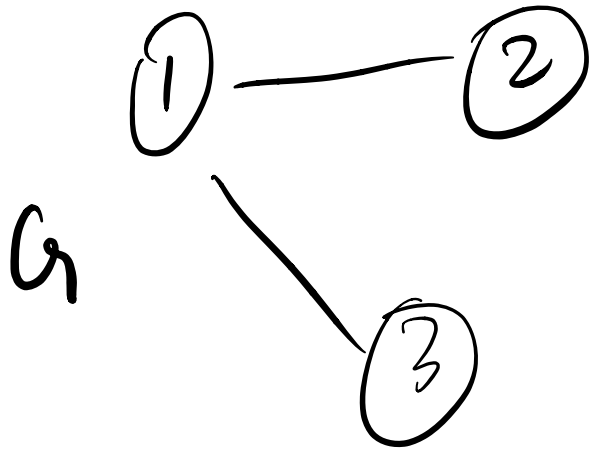
Communities

Goal: find clusters/communities such that nodes are densely connected within each cluster but sparsely connected across clusters.



Graphs

- $G = (V, E)$ is a graph or network which consists of
  - a set of vertices or nodes $V = \{1, \ldots, n\}$
  - a set of edges or links $E \subseteq V \times V$
- We only consider undirected graphs i.e. edges have no directions.



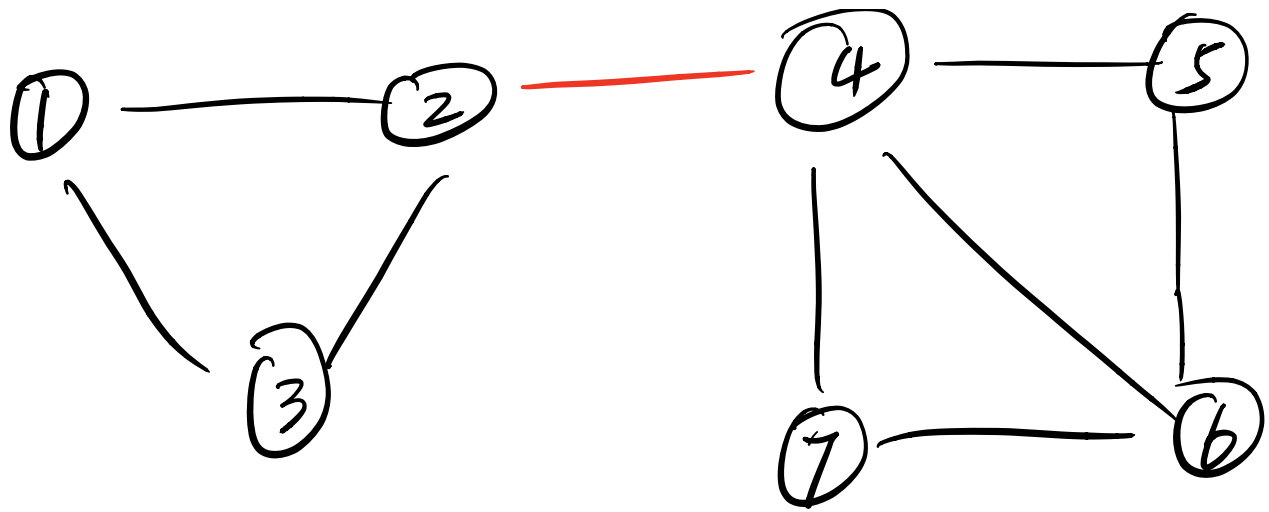
- $V = \{1, 2, 3\}$
- $E = \{\{1, 2\}, \{1, 3\}\}$

Community Detection Methods

- **Girvan-Newman algorithm** clusters nodes based on the notion of edge betweenness. Similarly to hierarchical clustering, the result can be viewed as a dendrogram.
- **Spectral clustering** cuts a graph into several components. Using spectral theorem, we essentially embed nodes onto low dimensional real space and cluster the embedded features.
- **Stochastic block model** is a generative model which generates random graphs given model parameters. The learning process is the reverse: given an observed graph, estimate the model parameters.

Girvan-Newman algorithm

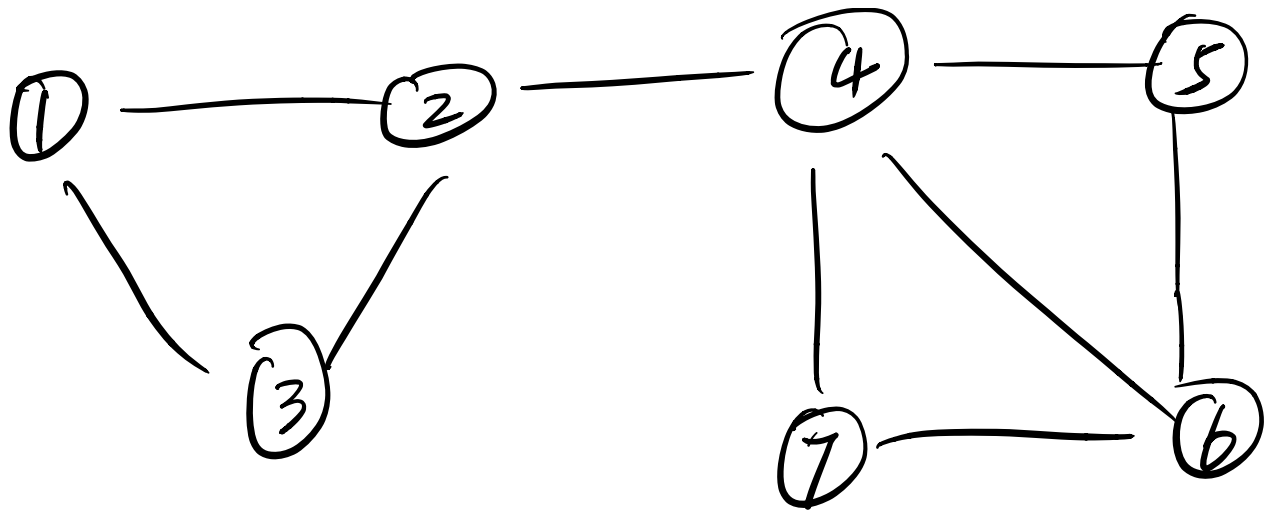
Intuition: if an edge is "heavily traveled", it's likely to be an edge between two communities. Girvan-Newman algorithm builds on this intuition.



Edge betweenness

- A **path of length** K is a sequence of $K+1$ distinctive nodes $i_0, \ldots, i_K$ (except possibly $i_0 = i_K$) such that $\{i_{k-1}, i_k\} \in E$ for $k = 1, \ldots, K$.
- The path between two nodes may not be unique. The **shortest path** is crucial here.
- **Edge betweenness:** the number of shortest paths passing through the edge

Example

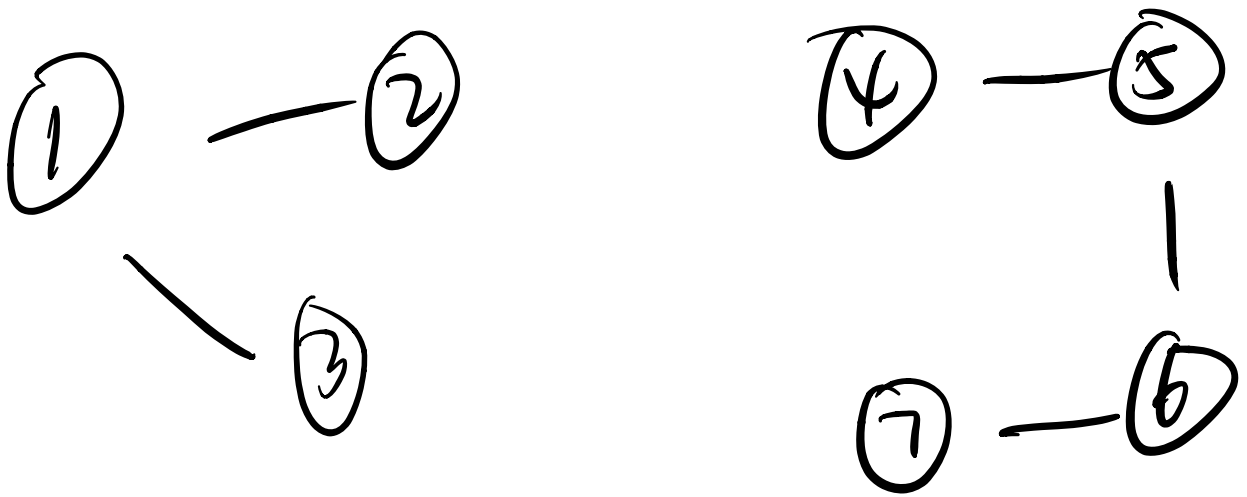


Edge $\{2, 4\}$ has the highest betweenness. In fact, this edge is on every shortest path between any of 1, 2, and 3 to any of 4, 5, 6, and 7. Its betweenness is therefore $3 \times 4 = 12$.

In contrast, the edge $\{4, 6\}$ is on only four shortest paths: those from any of 1, 2, 3, and 4 to 6.

Connect components

A **(connected) component** is a maximal set of nodes such that each pair of nodes is connected by a path.

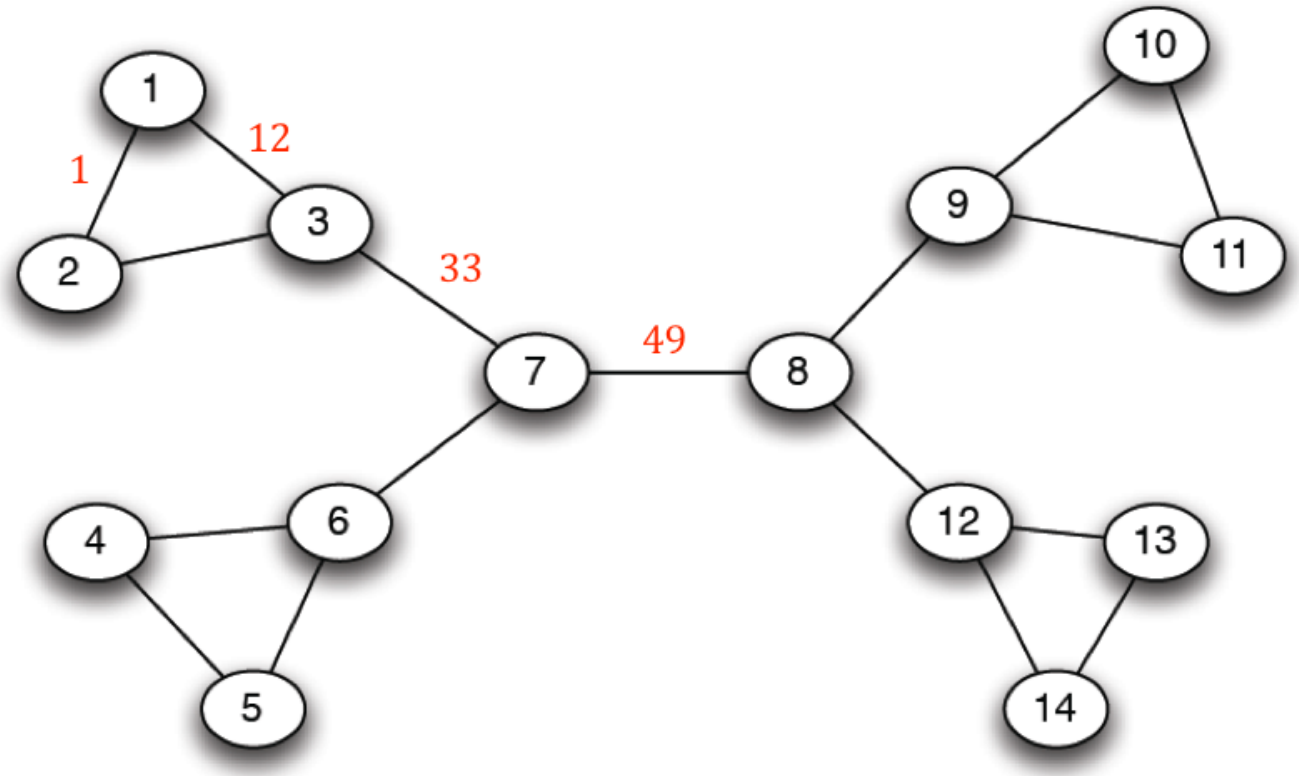


Two connected components: $\{1, 2, 3\}$ and $\{4, 5, 6, 7\}$

Girvan-Newman algorithm

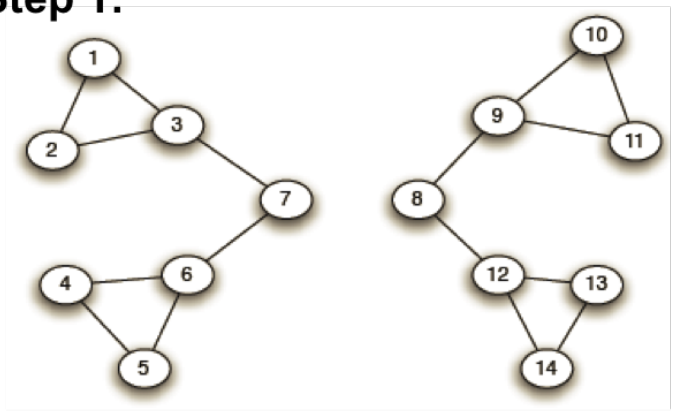
- We build a hierarchy in a "top-down" fashion. (Recall: we have discussed building a hierarchical clustering dendrogram in a "bottom-up" fashion.)
- Start with all points in one cluster. Repeat until no edges are left:
  1. Calculate betweenness of edges
  2. Remove edges with highest betweenness
- Connected components are communities
- Gives a hierarchical decomposition of the network

Example

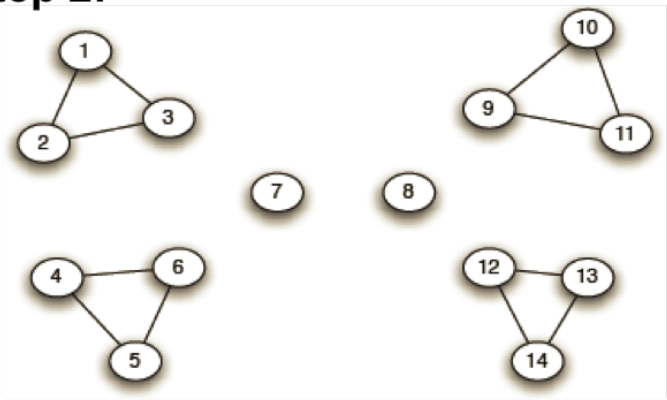


Example

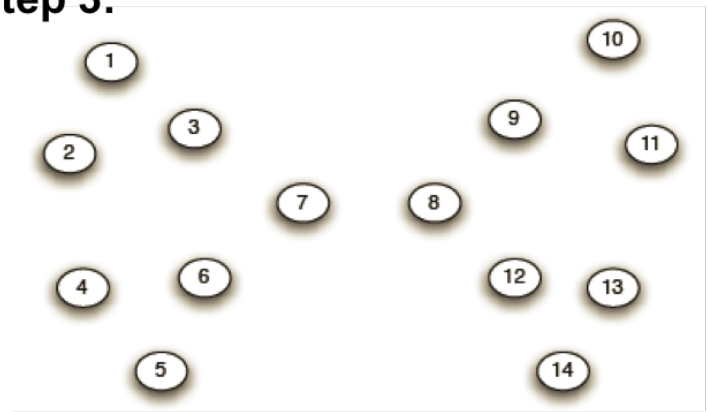
Step 1:



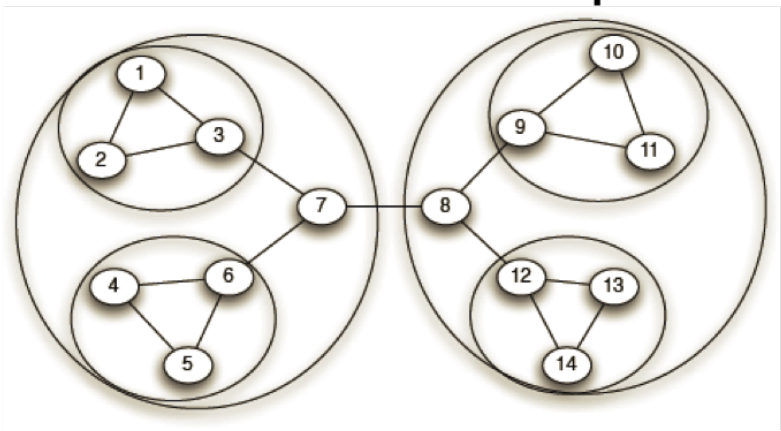
Step 2:



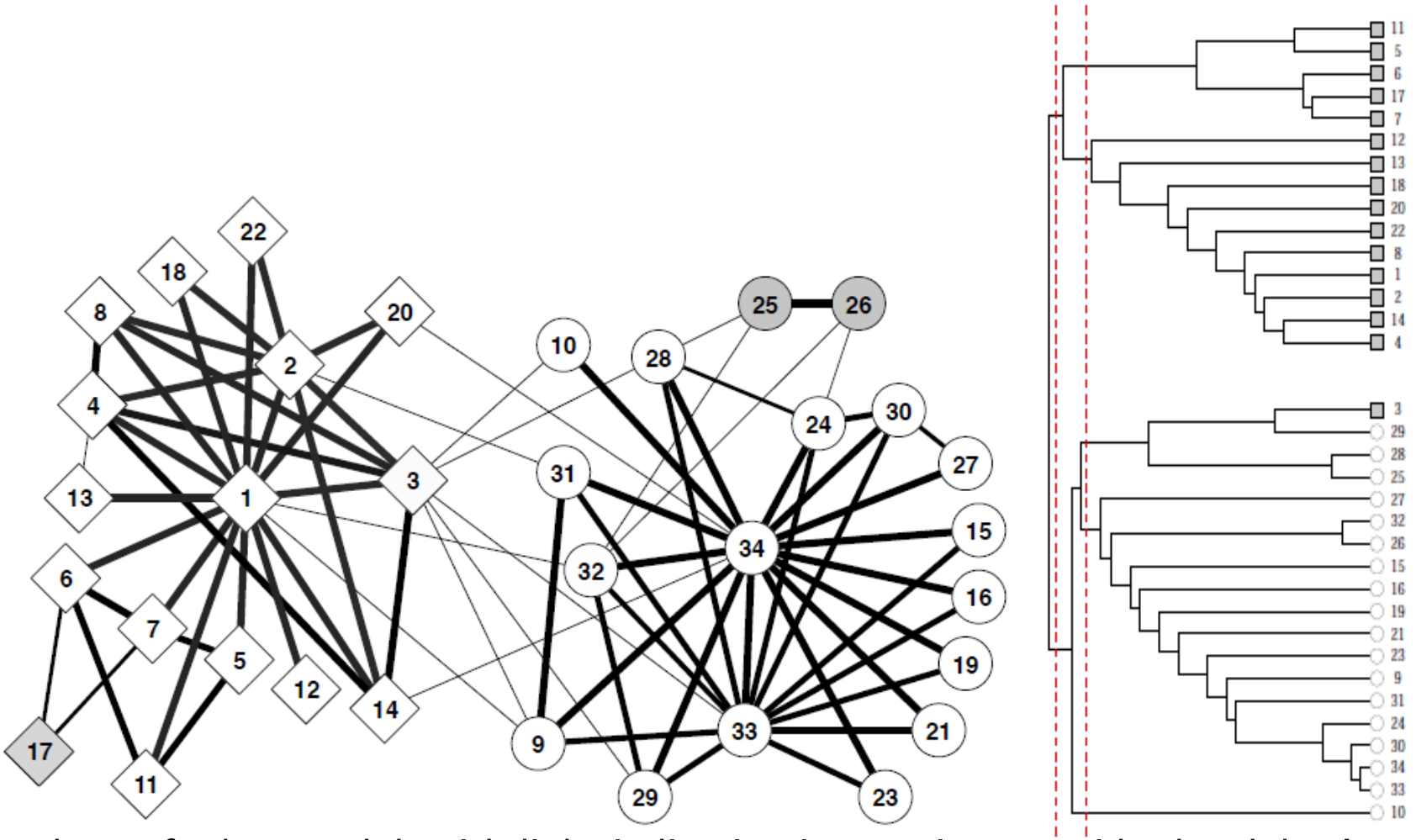
Step 3:



Hierarchical network decomposition:



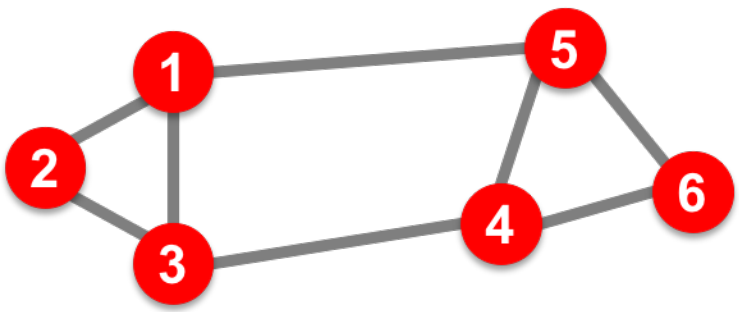
Zachary's Karate club

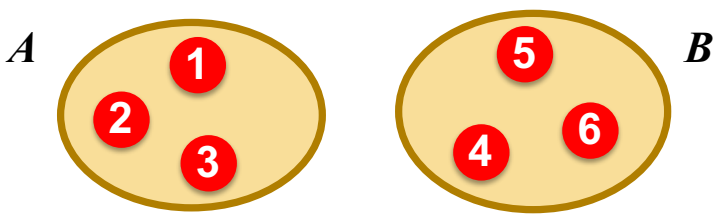


34 members of a karate club with links indicating interactions outside the club. A conflict arose between the administrator (node 34) and instructor (node 1), which led to the split of the club. Half of the members formed a new club around the instructor; members from the other part found a new instructor or gave up karate. G-N algorithm correctly assigned all but one member of the club to the groups they actually joined after the split.

Spectral clustering

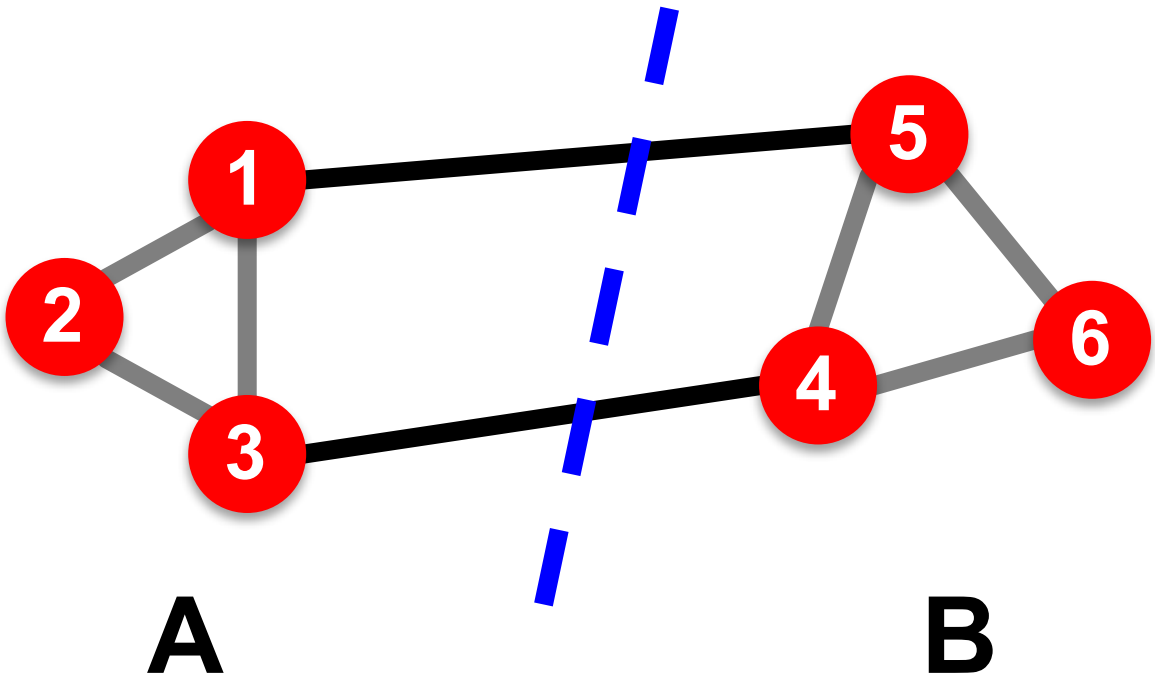
- Partition the network into two components by cutting edges.
- What makes a good partition?
- How can we efficiently identify such a partition?
- What if we want to partition the network into more than two components?



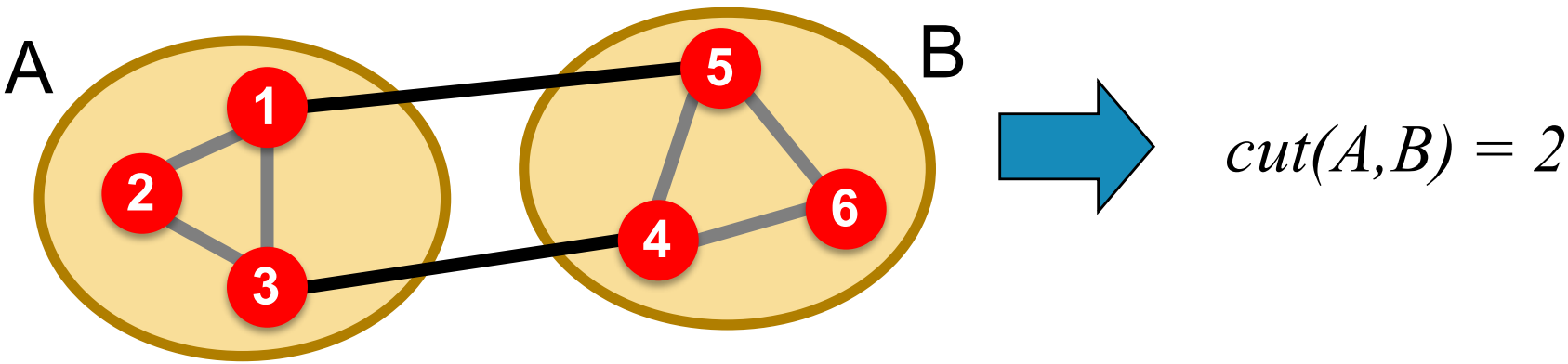


What makes a good partition?

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



- Express partitioning objectives as a function of the edge cut of the partition
- Cut: the number of edges between two groups, denoted by $cut(A, B)$

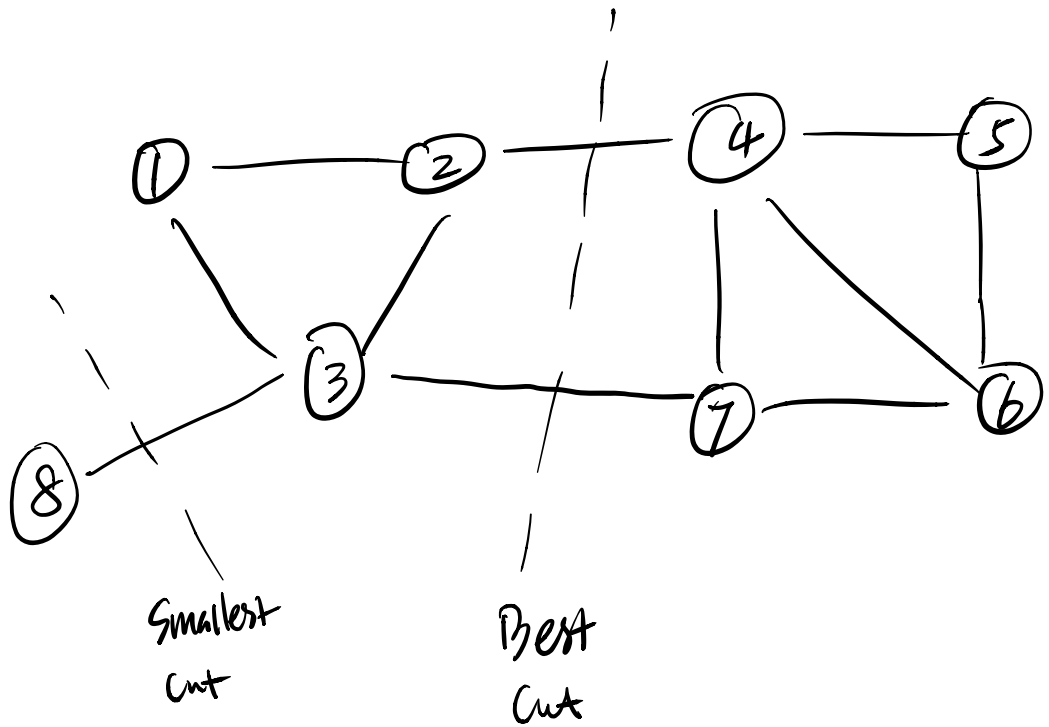


Minimize cut

- Find a partition $\{A, B\}$ of nodes that minimizes the graph cut

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

- Degenerate case:



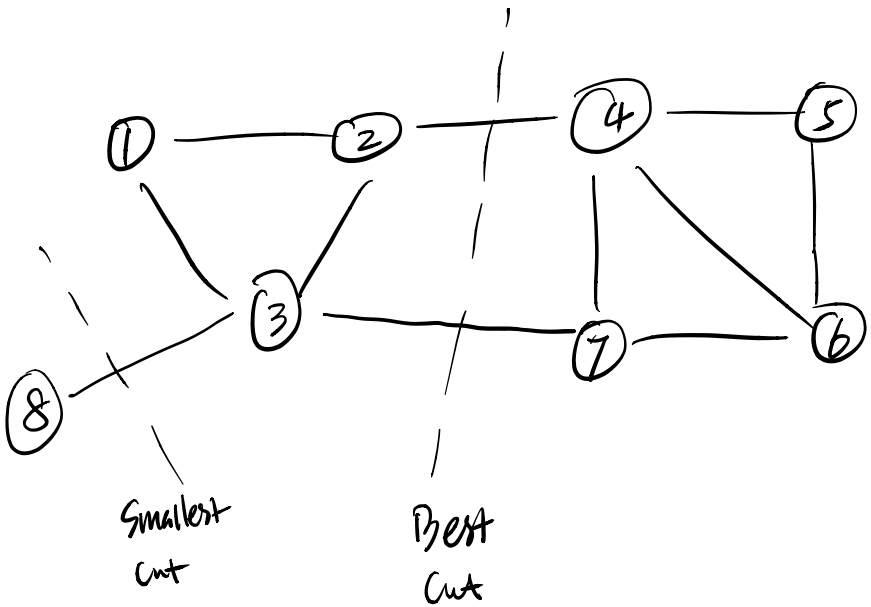
- Problem: we only consider edges between groups. Edges within groups are ignored.

Normalized cut

- Connectivity between groups relative to the density of each group

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

where $vol(A)$ is the total number of edges in A



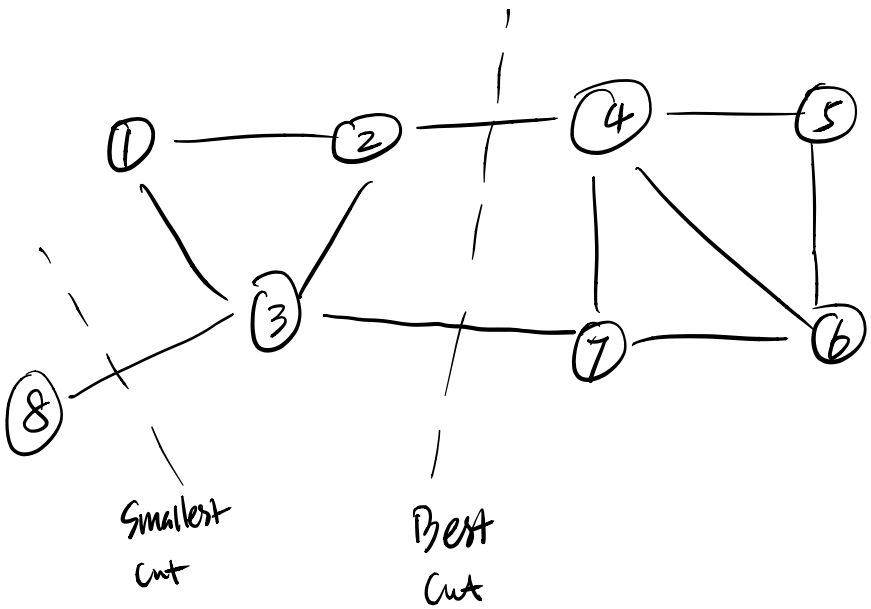
If we choose $S = \{8\}$ and $T = \{1, 2, 3, 4, 5, 6, 7\}$ (the smallest cut), then $cut(S, T) = 1$. $vol(S) = 1$, because there is only one edge connected to 8. On the other hand, $vol(T) = 11$, because all the edges have at least one end at a node of T. Thus, the normalized cut for this partition is $1/1 + 1/11 = 1.09$.

Normalized cut

- Connectivity between groups relative to the density of each group

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

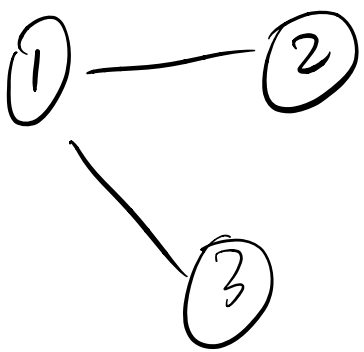
where $vol(A)$ is the total number of edges in A



Now if $S = \{1, 2, 3, 8\}$ and $T = \{4, 5, 6, 7\}$ (the best cut), then $cut(S, T) = 2$, $vol(S) = 6$, and $vol(T) = 7$. The normalized cut for this partition is thus only $2/6 + 2/7 = 0.62$.

How do we efficiently find a good partition?

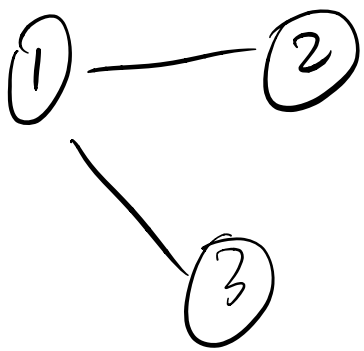
- In principle, we can compute $ncut(A, B)$ for all pairs of (A, B) and pick the pair with the smallest $ncut(A, B)$.
- But how many pairs in a network of n nodes? 2^{n-1}
- So can't solve the graph cut problem exactly.
- To find a good partition,we need a bit of matrix theory.
- How to represent a graph by a matrix?
- A graph can be represented by an **adjacency matrix** A with $A_{ij} = 1$ if there is an edge between i and j, and $A_{ij} = 0$ otherwise. A is binary and symmetric.



$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Graph Laplacian

- The degree d_i of a node i is the number of edges connected to it.
- The **degree matrix** D is a diagonal matrix with the diagonal entries being d_i.
- The **Laplacian matrix** is $L = D - A$.



$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$L = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

- Let $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ be a vector in $\mathbb{R}^n$.
- Think of it as a label/value of each node.

Spectral Graph Theory

- Let $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ be a vector in $\mathbb{R}^n$.
- Think of it as a label/value of each node.
- We are going to use spectral theory which studies eigenpairs $(\boldsymbol{x}_i, \lambda_i)$ of the Laplacian matrix L, ordered by the magnitude (strength) of the eigenvalues $\lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n$.

Example: d-regular graph

- Suppose all nodes have degree d and the graph G is connected. This is also known as d-regular graph.
- What are some eigenvalues/vectors of L?
  $\boldsymbol{L}\boldsymbol{x} = \lambda \boldsymbol{x}$ What's λ? What's $\boldsymbol{x}$?

Example: d-regular graph

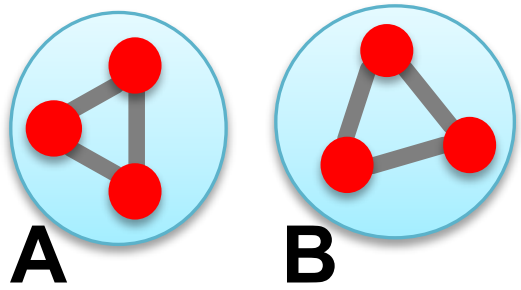
- Suppose all nodes have degree d and the graph G is connected. This is also known as d-regular graph.
- What are some eigenvalues/vectors of L?
  $L\boldsymbol{x} = \lambda \boldsymbol{x}$ What's λ? What's $\boldsymbol{x}$?
- Let's try $\boldsymbol{x} = \mathbf{1} = (1, 1, \ldots, 1)^T$.
- Then $L\boldsymbol{x} = \mathbf{0} = 0 \cdot \boldsymbol{x}$. So $\lambda = 0$.

Example: d-regular graph

- Suppose all nodes have degree d and the graph G is connected. This is also known as d-regular graph.
- What are some eigenvalues/vectors of L?
  $L\boldsymbol{x} = \lambda \boldsymbol{x}$ What's λ? What's $\boldsymbol{x}$?
- Let's try $\boldsymbol{x} = \mathbf{1} = (1, 1, \ldots, 1)^T$.
- Then $L\boldsymbol{x} = \mathbf{0} = 0 \cdot \boldsymbol{x}$. So $\lambda = 0$.
- We found an eigenpair of L: $\boldsymbol{x} = \mathbf{1}$ and $\lambda = 0$.

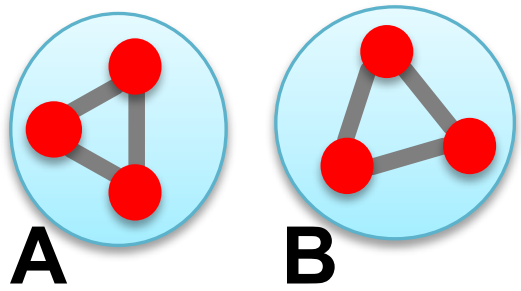
Graph with 2 components

- What if G is not connected?
- G has 2 components, each d-regular
- What are some eigenvectors?



Graph with 2 components

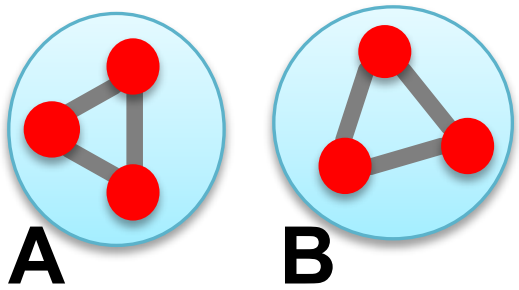
- What if G is not connected?
- G has 2 components, each d-regular
- What are some eigenvectors?



- $\boldsymbol{x} = (\underbrace{1, 1, \ldots, 1}_{A}, \underbrace{0, \ldots, 0}_{B})^T \implies L\boldsymbol{x} = \mathbf{0} = 0 \cdot \boldsymbol{x}.$

Graph with 2 components

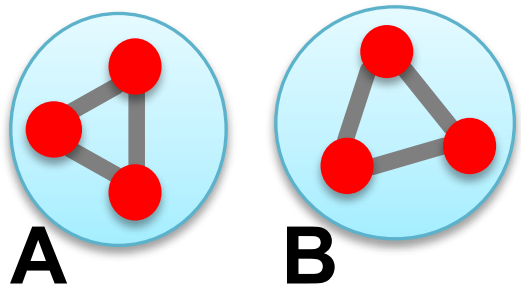
- What if G is not connected?
- G has 2 components, each d-regular
- What are some eigenvectors?



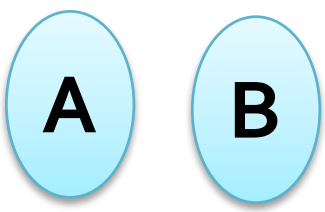
- $\boldsymbol{x} = (\underbrace{1, 1, \ldots, 1}_{A}, \underbrace{0, \ldots, 0}_{B})^T \implies L\boldsymbol{x} = \mathbf{0} = 0 \cdot \boldsymbol{x}.$
- $\boldsymbol{x} = (0, \ldots, 0, 1, 1, \ldots, 1)^T \implies L\boldsymbol{x} = \mathbf{0} = 0 \cdot \boldsymbol{x}.$
- And so in both cases the corresponding $\lambda = 0$ and $\boldsymbol{x}$ indicates the right cluster membership.

Graph with 2 components

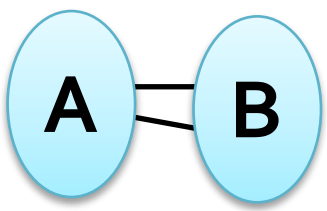
- What if G is not connected?
- G has 2 components, each d-regular
- What are some eigenvectors?



- $\boldsymbol{x} = (\underbrace{1, 1, \ldots, 1}_{A}, \underbrace{0, \ldots, 0}_{B})^T \implies L\boldsymbol{x} = \mathbf{0} = 0 \cdot \boldsymbol{x}.$
- $\boldsymbol{x} = (0, \ldots, 0, 1, 1, \ldots, 1)^T \implies L\boldsymbol{x} = \mathbf{0} = 0 \cdot \boldsymbol{x}.$
- And so in both cases the corresponding $\lambda = 0$ and $\boldsymbol{x}$ indicates the right cluster membership.
- A bit of intuition



$\boldsymbol{\lambda_1 = \lambda_2}$



$\boldsymbol{\lambda_1 \approx \lambda_2}$

2nd small e-val. λ_2 now has value very close to λ_1

Properties of Laplacian Matrix

- $L = D - A$ is symmetric.
- Eigenvalues are non-negative numbers.
- Eigenvectors are orthogonal, i.e. $\boldsymbol{x}_i^T \boldsymbol{x}_j = 0$ for any $i \neq j$.
- The trivial eigenpair $(\boldsymbol{x} = \mathbf{1}, \lambda = 0)$
- Obviously, $\lambda_1 = \lambda = 0$ is the smallest eigenvalue.

Second smallest eigenvalue λ_2

- Second smallest eigenvalue is critical for spectral clustering.
- It turns out

$$\lambda_2 = \min_{\boldsymbol{x}} \boldsymbol{x}^T L \boldsymbol{x} \quad \text{subject to} \quad ||\boldsymbol{x}||_2 = 1$$

- What is the meaning of $\min_{\boldsymbol{x}} \boldsymbol{x}^T L \boldsymbol{x}$ for a graph G?

$$\begin{aligned}
\boldsymbol{x}^T L \boldsymbol{x} &= \sum_{i=1}^{n} \sum_{j=1}^{n} L_{ij} x_i x_j = \sum_{i=1}^{n} \sum_{j=1}^{n} (D_{ij} - A_{ij}) x_i x_j \\
&= \sum_{i=1}^{n} D_{ii} x_i^2 - \sum_{\{i,j\} \in E} 2 x_i x_j \\
&= \sum_{\{i,j\} \in E} (x_i^2 + x_j^2 - 2 x_i x_j) = \sum_{\{i,j\} \in E} (x_i - x_j)^2
\end{aligned}$$

λ_2 as optimization problem

- What else do we know about x?
  1. $\boldsymbol{x}$ is a unit vector: $\sum_{i=1}^{n} x_i^2 = 1$
  2. $\boldsymbol{x}$ is orthogonal to the eigenvector $\mathbf{1} = (1, \ldots, 1)^T$:

$$\boldsymbol{x}^T \mathbf{1} = \sum_{i=1}^{n} x_i = 0$$

λ_2 as optimization problem

- What else do we know about x?
  1. $\boldsymbol{x}$ is a unit vector: $\sum_{i=1}^{n} x_i^2 = 1$
  2. $\boldsymbol{x}$ is orthogonal to the eigenvector $\mathbf{1} = (1, \ldots, 1)^T$:

$$\boldsymbol{x}^T \mathbf{1} = \sum_{i=1}^{n} x_i = 0$$

- The above implies some x_i are positive and some are negative.

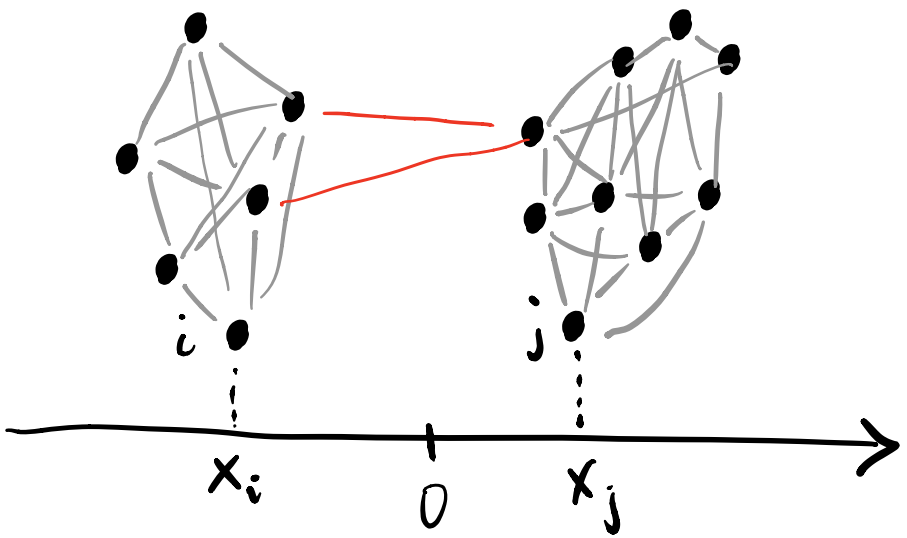
λ_2 as optimization problem

- What else do we know about x?
  1. $\boldsymbol{x}$ is a unit vector: $\sum_{i=1}^{n} x_i^2 = 1$
  2. $\boldsymbol{x}$ is orthogonal to the eigenvector $\mathbf{1} = (1, \ldots, 1)^T$:

$$\boldsymbol{x}^T \mathbf{1} = \sum_{i=1}^{n} x_i = 0$$

- The above implies some x_i are positive and some are negative.

- Recall: $\lambda_2 = \min_{\boldsymbol{x}} \sum_{\{i,j\} \in E} (x_i - x_j)^2$.
- We want x_i and x_j to have the same sign if node i and node j are connected.



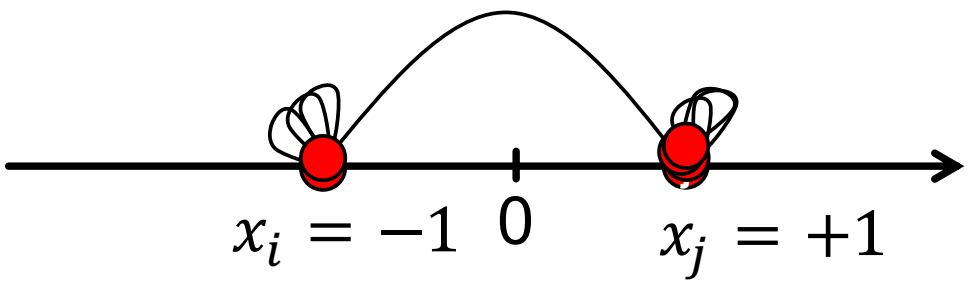
Find Optimal Cut

- Back to finding the optimal cut
- Express partition (A, B) as a vector

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

- We can minimize the cut of the partition by finding a non-trivial vector $\boldsymbol{x}$ that minimizes:

$$\underset{\boldsymbol{x} \in \{-1,+1\}^n}{\arg\min} \sum_{\{i,j\} \in E} (x_i - x_j)^2$$



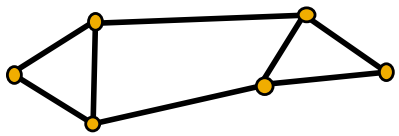
- Can't solve exactly. Let's relax $\boldsymbol{x}$ and allow it to take any real values.

Continuous relaxation

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \sum_{\{i,j\}\in E} (x_i - x_j)^2 = \boldsymbol{x}^T L \boldsymbol{x}$$

- As stated earlier, the minimum is the second smallest eigenvalue λ_2 of the Laplacian matrix L and its corresponding eigenvector $\boldsymbol{x}$ is the solution.
- Assign nodes with positive x_i to one cluster and nodes with negative x_i to another.

Illustration



	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

$\lambda =$

0.0
1.0
3.0
3.0
4.0
5.0

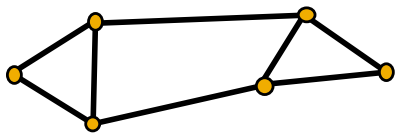
$X =$

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?

Illustration



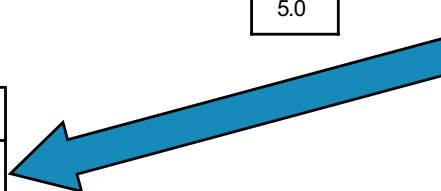
	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

$\lambda =$

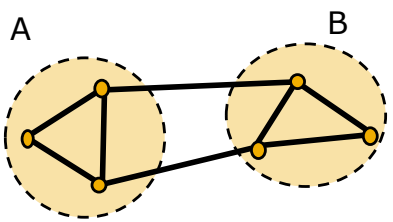
0.0
1.0
3.0
3.0
4.0
5.0

$X =$

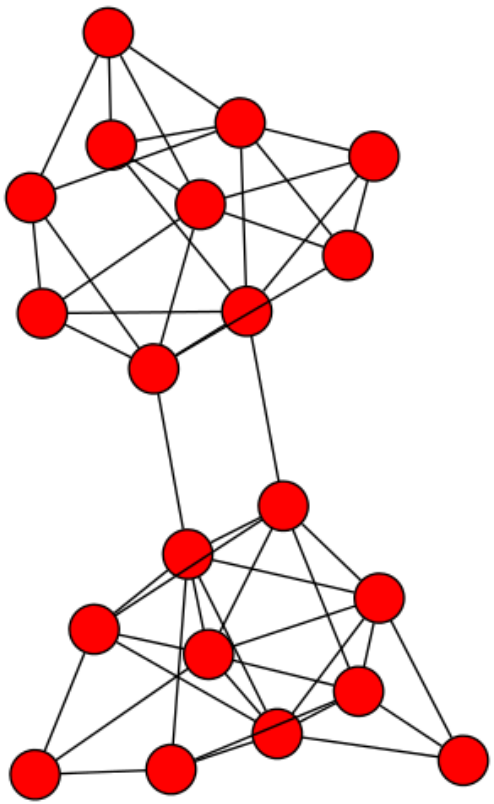
0.4	0.3	-0.5	-0.2	-0.4	-0.5
0.4	0.6	0.4	-0.4	0.4	0.0
0.4	0.3	0.1	0.6	-0.4	0.5
0.4	-0.3	0.1	0.6	0.4	-0.5
0.4	-0.3	-0.5	-0.2	0.4	0.5
0.4	-0.6	0.4	-0.4	-0.4	0.0

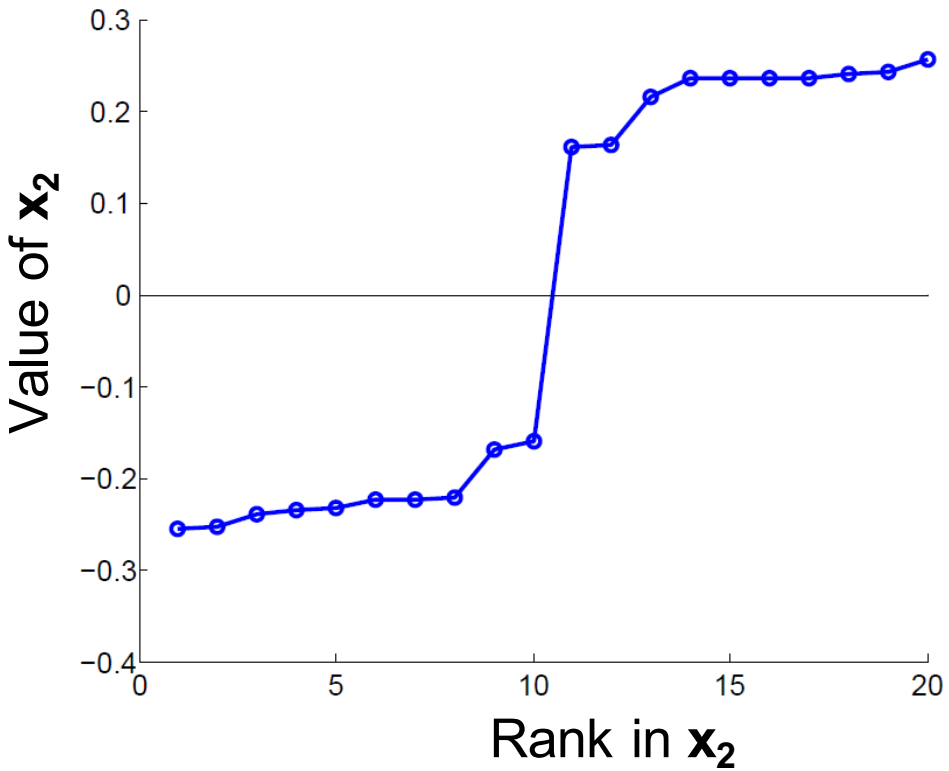


1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6



Example




0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
0
5
10
15
20
Value of $\mathbf{x_2}$
Rank in $\mathbf{x_2}$

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

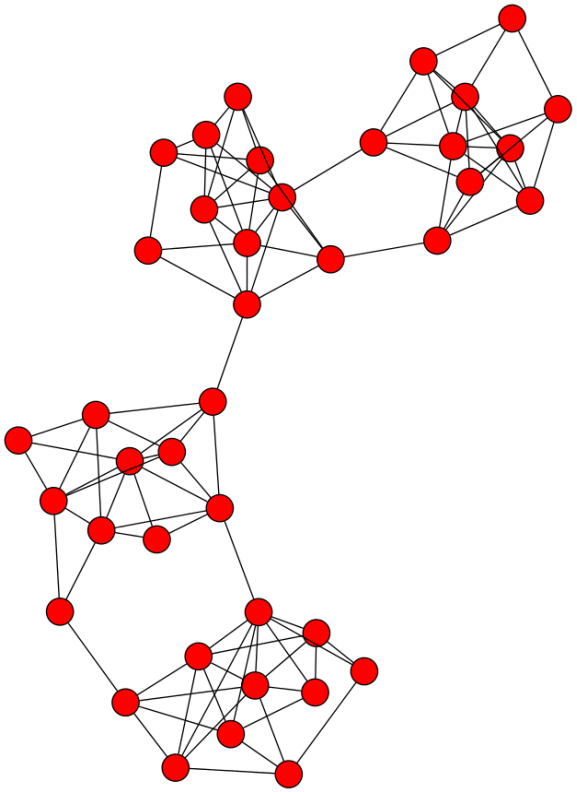
So far...

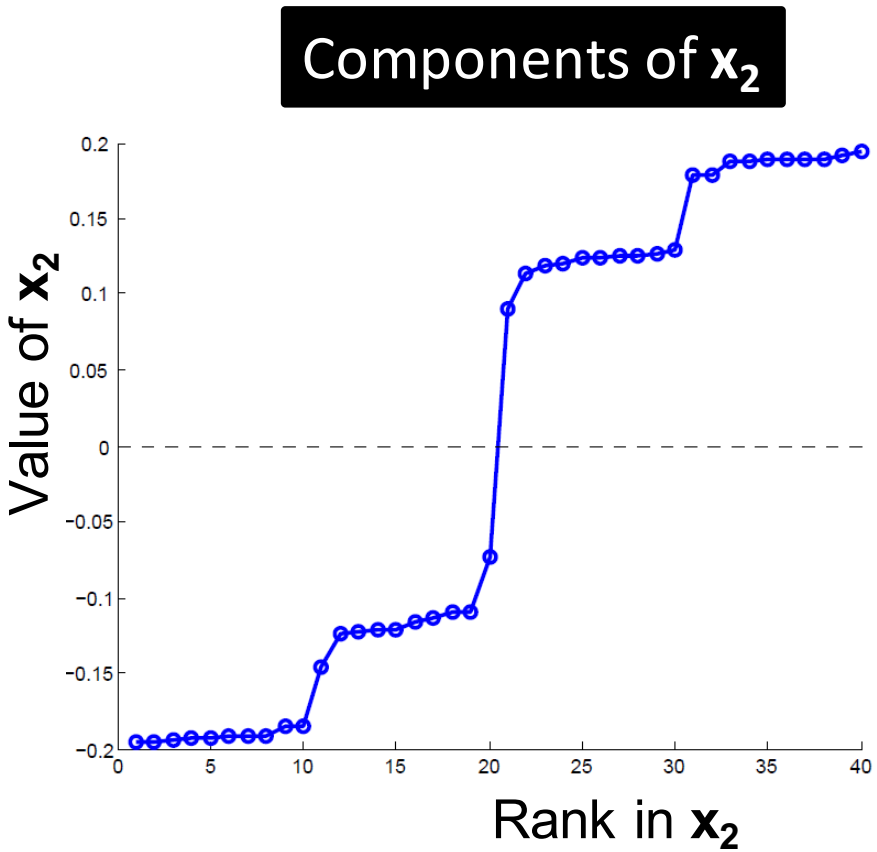
- How to define a "good" partition of a graph?
  - Minimize a given graph cut criterion
- How to efficiently identify such a partition?
  - Approximate using information provided by the eigenvalues and eigenvectors of a graph
- We are not quite done yet...
  1. Graph can have weights. We replace adjacency matrix, degree matrix and Laplacian matrix by their weighted versions.
  2. There are other types of Laplacian matrix which are more commonly used than the one we introduced. E.g. symmetric normalized Laplacian and random walk normalized Laplacian.
  3. It's applicable for $K > 2$ clusters as well.
  4. It's useful beyond clustering network data.

K-way Spectral Clustering

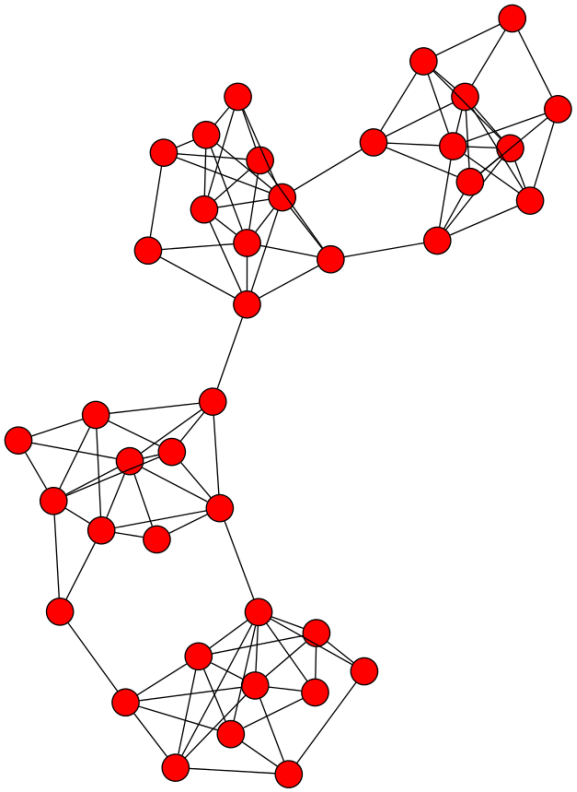
- How do we partition a graph into K clusters?
- Two basic approaches:
  1. Recursive bi-partitioning: recursively apply bi-partitioning algorithm in a hierarchical manner. Disadvantages: Inefficient, unstable
  2. Cluster multiple eigenvectors. Commonly used in recent papers. See next two slides for intuitions.

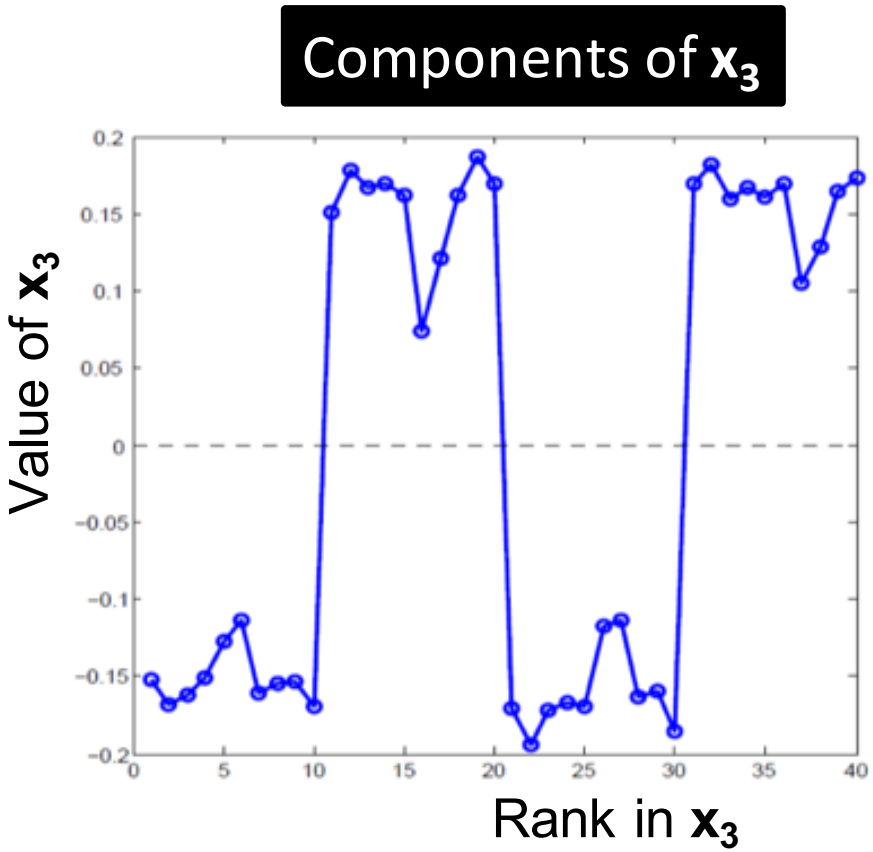
Some intuitions





Some intuitions





Graph Spectral Clustering Algorithm

Input: a graph $\mathcal{G}$ and the number K of clusters

1. Compute the Laplacian matrix L from $\mathcal{G}$ and the first K eigenvectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K$ of L. Let $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K]$ be an $n \times K$ matrix of eigenvectors. Why K?
2. Use any clustering algorithm such as K-means to cluster the rows of X.

Output: clusters $C_1, \ldots, C_K$

General Spectral Clustering Algorithm

Input: data $Y \in \mathbb{R}^{n \times p}$ and the number K of clusters

1. Construct a similarity graph $\mathcal{G}$ from Y. (see next slide)
2. Compute the Laplacian matrix L from $\mathcal{G}$ and the first K eigenvectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K$ of L. Let $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K]$ be an $n \times K$ matrix of eigenvectors.
3. Use any clustering algorithm such as K-means to cluster the rows of X.

Output: clusters $C_1, \ldots, C_K$

Ways to construct a similarity graph

Let s_{ij} denote a similarity measure between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the ith and jth rows of X. E.g. Gaussian similarity

$$s_{ij} = \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{2\sigma^2}\right)$$

- k-nearest neighbor graphs: connect nodes i and j if i is among the k-nearest neighbors of j or i is among the k-nearest neighbors of i. Weight all edges by s_{ij}.
- ϵ-neighborhood graph: connect nodes i and j if $||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 < \epsilon$. For small ϵ, no need to weight the edges. Why?
- Fully connected graph: connect all nodes and weight all edges by s_{ij}.

Advantage of spectral clustering

Advantages:

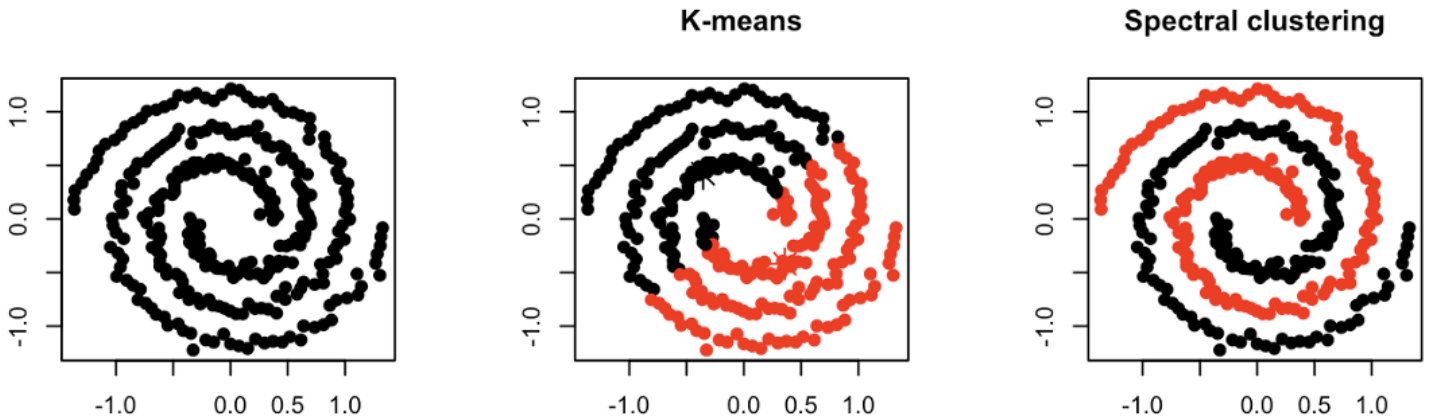
- Spectral clustering reduces the dimension from p to K.
- Capable of finding non-convex clusters.

Disadvantages:

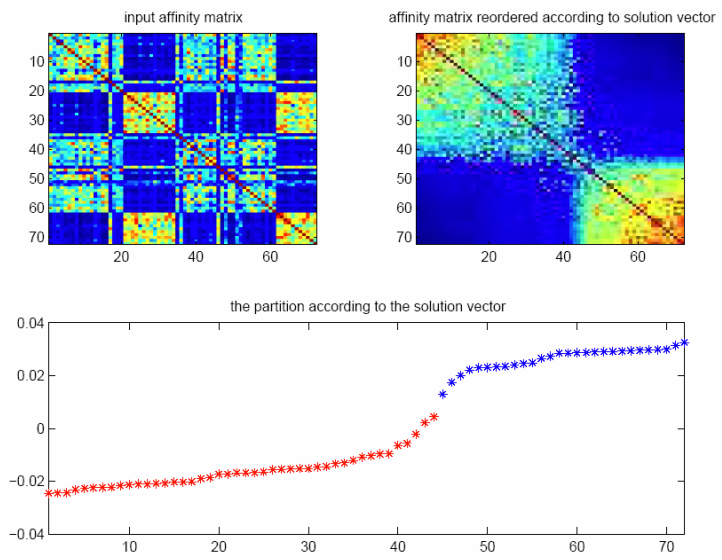
- Similarity function and its associated parameters matter.

Examples

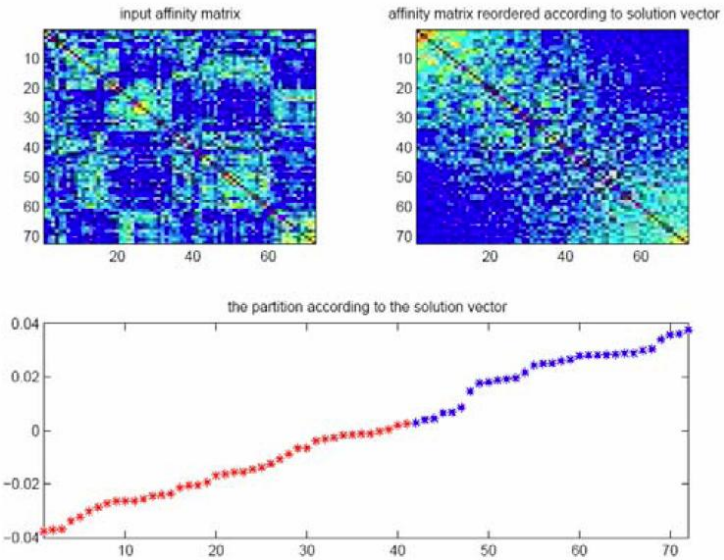
Non-covexitiy



Good v.s. poor similarity measure



Good similarity measure



Poor similarity measure

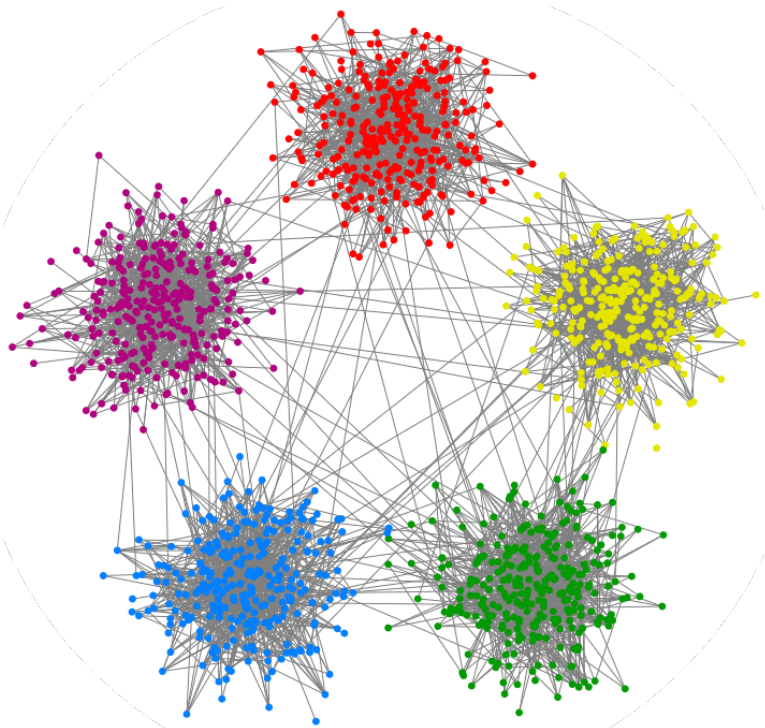
Stochastic Block Model

- Stochastic Block Model (SBM) is a generative model which generates random graphs with clusters/communities. An SBM depends on three sets of parameters:
  1. Number K of clusters.
  2. Cluster label $s_i \in \{1, \ldots, K\}$. Node i belongs to cluster k if $s_i = k$.
  3. Connectivity probabilities matrix $Q \in [0,1]^{K \times K}$. Q_{rs} is the probability of an edge between any node in cluster r and any node in cluster s.
- Given these parameters, SBM generates a random graph by

$$A_{ij} | Q, K, s_i, s_j \overset{ind}{\sim} Bernoulli(Q_{s_i s_j})$$

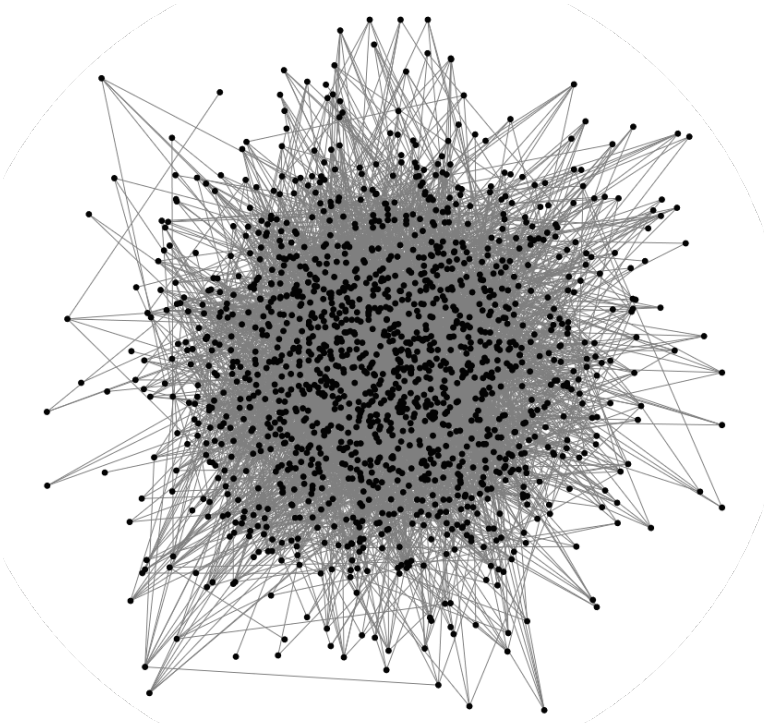
Simulated Example – generating a network with communities

- $n = 1000$ nodes
- $K = 5$ balanced clusters
- Within-cluster probability $Q_{rr} = 1/50$
- Across-cluster probability $Q_{rs} = 1/1000$ for $r \neq s$



Learning SBM

- We are given a messy network data without cluster labels and connectivity probabilities matrix.



- We need to learn $s = \{s_1, \ldots, s_n\}$ and Q from the data.

Likelihood

- It's easy to write down the joint likelihood

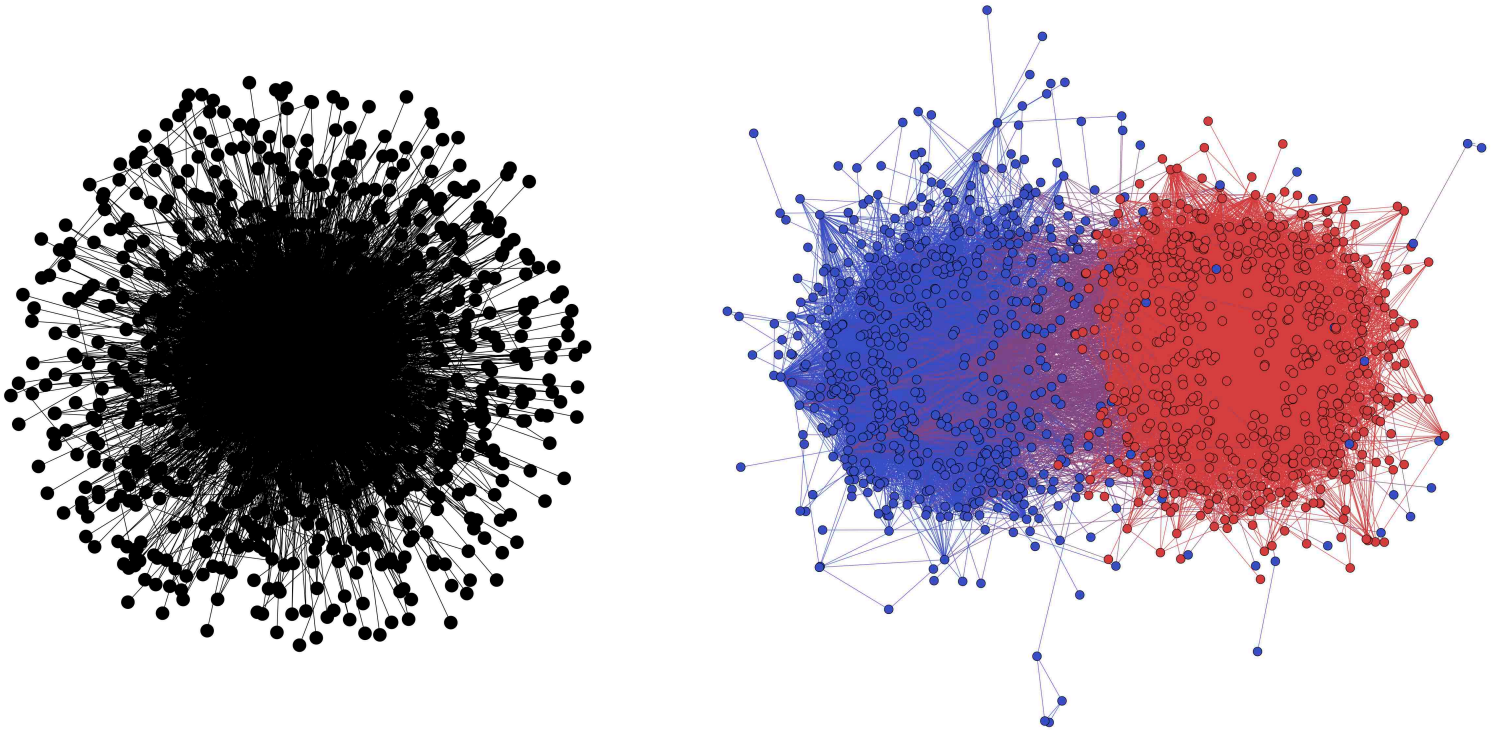
$$p(A|s, Q, K) = \prod_{i<j} Q_{s_i s_j}^{A_{ij}} (1 - Q_{s_i s_j})^{1-A_{ij}}$$

- In principle, for a given K, we can then

$$\max_{s,Q} p(A|s, Q, K)$$

- However, it is very hard to solve it exactly.
- Instead, a lot of approximated solutions were proposed such as semidefinite programming, variational EM algorithm, Markov chain Monte Carlo, belief propagation, spectral methods, etc.

Example



The above graphs represent the real data set of the political blogs from Adamic and Glance (2005). Each vertex represents a blog and each edge represents the fact that one of the blogs refers to the other. The left graph is plotted with a random arrangement of the vertices, and the right graph is the output of SBM (using belief propagation), which gives 95% accuracy on the reconstruction of the political inclination of the blogs (blue and red colors correspond to left and right leaning blogs).