

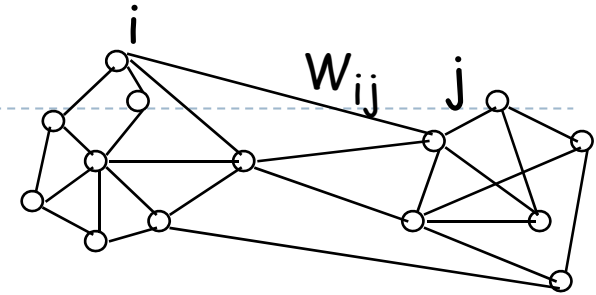


Spectral Clustering



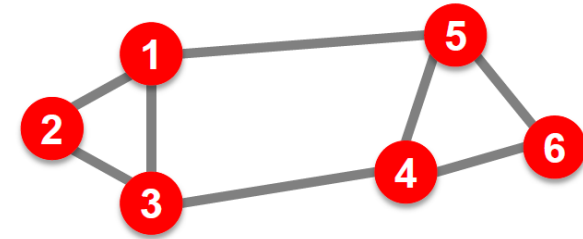
Basic Graph Notation

- ▶ A graph $G = (V, E)$ consists of vertices V and edges E
- ▶ Edges can be **weighted** or **unweighted**.
- ▶ The **adjacency matrix** 邻接矩阵 A (weight matrix W) describes the graph:
 - ▶ $A_{ij}=0$ ($W_{ij}=0$) if vertices i and j are not connected
 - ▶ $A_{ij}=1$ ($W_{ij}=\text{weight of the edge}$), if they are connected
- ▶ The **degree of a vertex** is the sum of all adjacent edge weights:
 $d_i = \sum_j A_{ij} \quad (\sum_j W_{ij})$
- ▶ All vertices which can be reached from each other by a path form a **connected component** (连通分支)



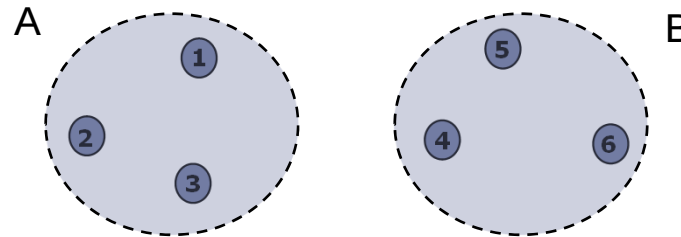
Graph Partitioning

- ▶ Undirected graph $G=(V, E)$:



- ▶ Partitioning task:

- ▶ Divide vertices into two disjoint groups (A, B)



- ▶ $V=A \cup B$

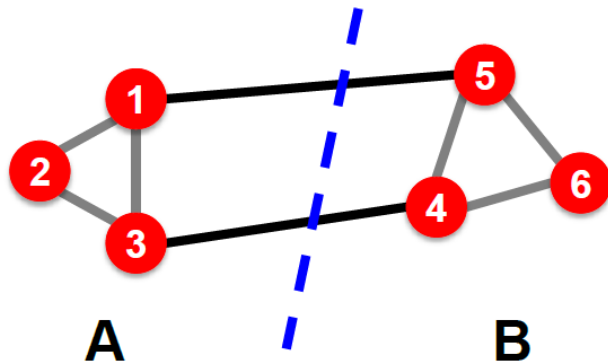
- ▶ Questions:

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

Graph Partitioning

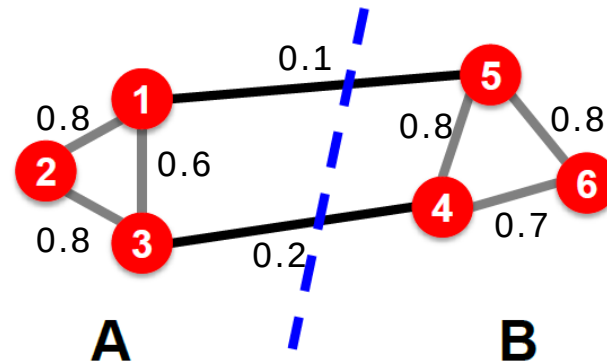
What makes a good partition?

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



What makes a good clustering?

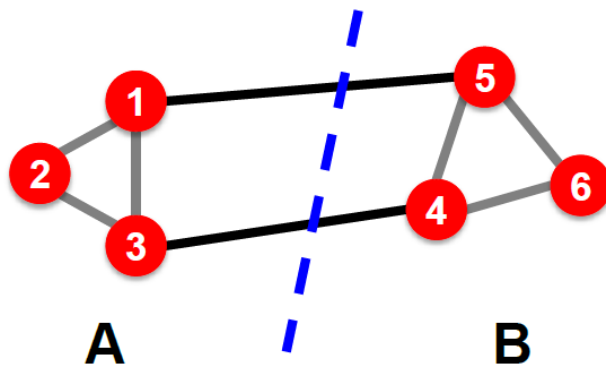
- ▶ Points assigned to the same cluster should be highly similar
- ▶ Points assigned to different clusters should be highly dissimilar



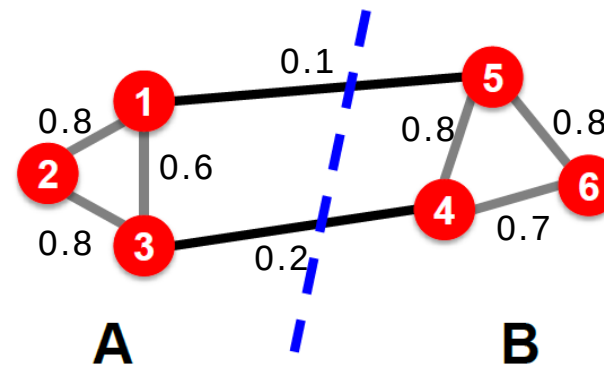
Graph Cuts

- ▶ Express partitioning objectives as a function of the “edge cut” of the partition.
- ▶ **Cut:** Set of edges with only one vertex in a group.

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



$$\text{cut}(A, B) = 2$$



$$\text{cut}(A, B) = 0.3$$

Graph Cuts

- ▶ The partitioning can be described by a labeling function y that indicates the two joint sets

$$y(u) = \begin{cases} 1 & \text{if } u \in A \\ -1 & \text{if } u \in B \end{cases}$$

$$\text{cut}(A, B) = \frac{1}{4} \sum_{i \in A, j \in B} w(i, j) (y(i) - y(j))^2$$

$$= \frac{1}{8} \sum_{i, j} w(i, j) (y(i) - y(j))^2$$

$$= \frac{1}{4} \sum_{i, j} w(i, j) (1 - y(i)y(j))$$

$$= \frac{1}{4} (\mathbf{y}^\top D \mathbf{y} - \mathbf{y}^\top W \mathbf{y}) = \frac{1}{4} \mathbf{y}^\top (D - W) \mathbf{y}$$

$$D = \begin{bmatrix} d_1 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & d_n \end{bmatrix}$$

D is a diagonal matrix with the degrees of all the vertices on the diagonal

Laplacian matrix L

Spectral properties of the Laplacian matrix L

- ▶ L is always positive semidefinite (半正定)
- ▶ Smallest eigenvalue of L is 0, corresponding eigenvector is $\mathbf{e}$

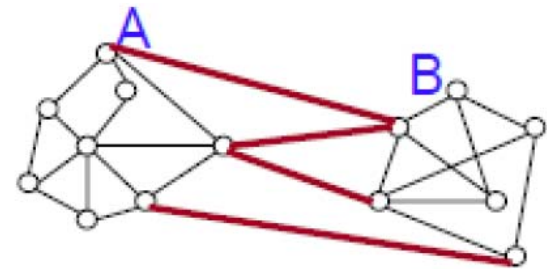
$$L\mathbf{e} = D\mathbf{e} - W\mathbf{e} = \begin{bmatrix} d_1 \\ d_2 \\ \dots \\ d_n \end{bmatrix} - \begin{bmatrix} \sum_j w_{1j} \\ \sum_j w_{2j} \\ \dots \\ \sum_j w_{nj} \end{bmatrix} = 0$$

- ▶ Thus eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$

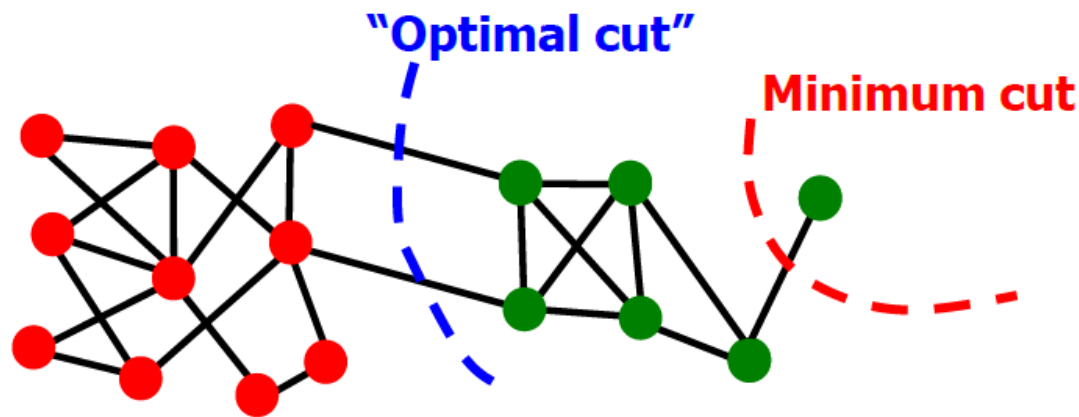
Graph cut criteria

- ▶ Min-cut: minimize weight of connections between groups

$$\min cut(A,B)$$



- ▶ Easy to solve: $O(|V| |E|)$ algorithm
- ▶ Not satisfactory partition – often isolates (孤立) vertices
 - Only considers external cluster connections
 - Does not consider internal cluster density



Graph cut criteria

Balanced min-cut:

$$\min cut(A, B) \quad \text{subject to} \quad |A| = |B|$$

Ratio cut:

$$RatioCut(A, B) = cut(A, B) \left(\frac{1}{|A|} + \frac{1}{|B|} \right)$$

Normalized cut:

$$Ncut(A, B) = cut(A, B) \left(\frac{1}{vol(A)} + \frac{1}{vol(B)} \right)$$

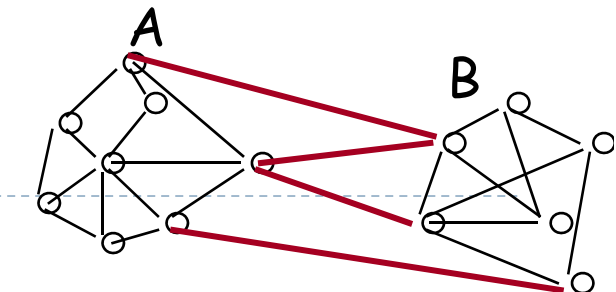
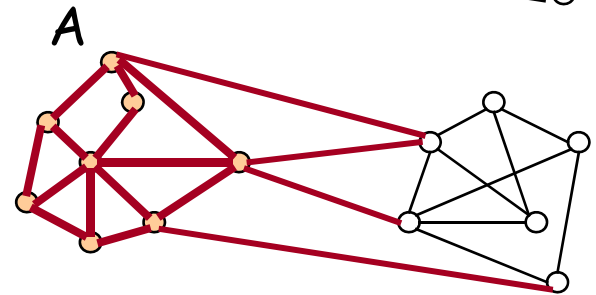
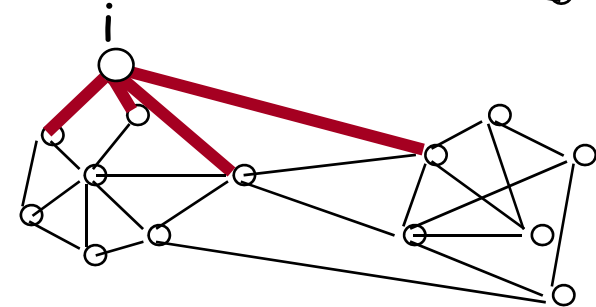
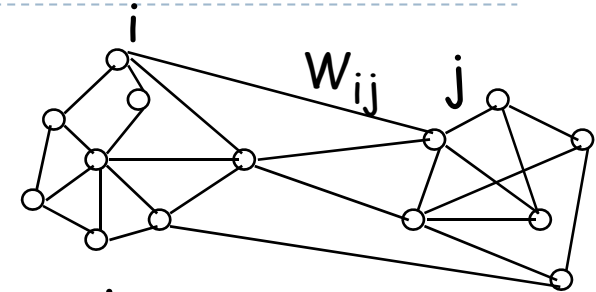
What's the difference between ratio cut and normalized cut in behavior?

NP-hard to solve!

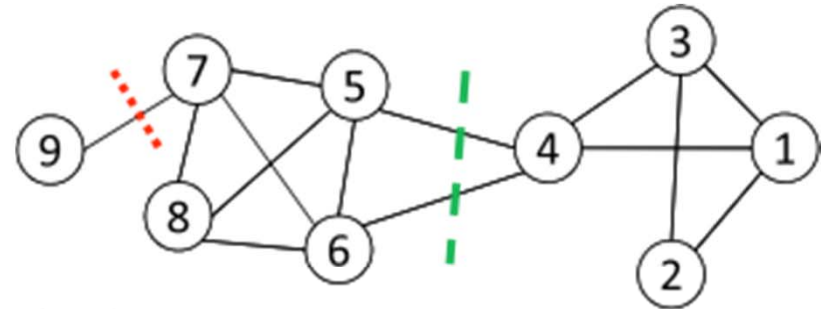
Spectral clustering is a relaxation of these

Basic Graph Notation

- ▶ $W = [W_{ij}]$
- ▶ $d_i = \sum_{j \in G} W_{ij}$ degree of i
- ▶ $\text{Vol}(A) = \sum_{i \in A} d_i$ degree of $A \subseteq G$
- ▶ $\text{cut}(A,B) = \sum_{i \in A} \sum_{j \in B} W_{ij}$



Ratio Cut & Normalized Cut Example



For partition in red: π_1

$$\text{Ratio Cut}(\pi_1) = \frac{1}{2} \left(\frac{1}{1} + \frac{1}{8} \right) = 9/16 = 0.56$$

$$\text{Normalized Cut}(\pi_1) = \frac{1}{2} \left(\frac{1}{1} + \frac{1}{27} \right) = 14/27 = 0.52$$

For partition in green: π_2

$$\text{Ratio Cut}(\pi_2) = \frac{1}{2} \left(\frac{2}{4} + \frac{2}{5} \right) = 9/20 = 0.45 < \text{Ratio Cut}(\pi_1)$$

$$\text{Normalized Cut}(\pi_2) = \frac{1}{2} \left(\frac{2}{12} + \frac{2}{16} \right) = 7/48 = 0.15 < \text{Normalized Cut}(\pi_1)$$

Both ratio cut and normalized cut prefer a balanced partition

Derivation of Normalized cut

Let $\beta = \frac{\sum_{i \in A} d(i)}{\sum_i d(i)}$, then

$$\begin{aligned} Ncut(A, B) &= \frac{1}{\sum_i d(i)} \left(\frac{1}{\beta} + \frac{1}{1 - \beta} \right) cut(A, B) \\ &= \frac{\sum_{i,j} w(i, j)(y(i) - y(j))^2}{8\beta(1 - \beta) \sum_i y^2(i)d(i)} \end{aligned}$$

and define a new function g :

$$g(u) = y(u) + (1 - 2\beta) = \begin{cases} 2(1 - \beta) & \text{if } u \in A \\ -2\beta & \text{if } u \in B \end{cases}$$

Derivation of Normalized cut cont'd

We have $g(i) - g(j) = y(i) - y(j)$

$$\begin{aligned}\text{And } \sum_i g^2(i)d(i) &= \sum_{i \in A} 4(1 - \beta)^2 y^2(i)d(i) + \sum_{i \in B} 4\beta^2 y^2(i)d(i) \\ &= 4(1 - \beta)^2 \beta \sum_i y^2(i)d(i) + 4\beta^2(1 - \beta) \sum_i y^2(i)d(i) \\ &= 4\beta(1 - \beta) \sum_i y^2(i)d(i)\end{aligned}$$

Therefore

$$\begin{aligned}Ncut(A, B) &= \frac{\sum_{i,j} w(i,j)(g(i) - g(j))^2}{2 \sum_i g^2(i)d(i)} \\ &= \frac{\mathbf{g}^\top (D - W) \mathbf{g}}{\mathbf{g}^\top D \mathbf{g}}\end{aligned}$$

Derivation of Normalized cut cont'd

- ▶ However, we still have the discrete constraint on g , which causes NP-hardness.
- ▶ Relaxation (松弛) : $g^\top D e = 0$
- ▶ The normalized cut problem can now be written as

$$\min_g \frac{g^\top (D - W) g}{g^\top D g} \quad \text{s.t.} \quad g^\top D e = 0$$

- ▶ Or let $g' = D^{\frac{1}{2}} g$

$$\min_{g'} \frac{g'^\top D^{-\frac{1}{2}} (D - W) D^{-\frac{1}{2}} g'}{g'^\top g'} \quad \text{s.t.} \quad g'^\top D^{\frac{1}{2}} e = 0$$

Normalized
Laplacian matrix L'

Derivation of Normalized cut cont'd

► Rayleigh-Ritz theorem

$$\min_{\mathbf{X}} \frac{\mathbf{X}^\top S \mathbf{X}}{\mathbf{X}^\top \mathbf{X}} = \lambda_{\min}(S) \quad - \text{smallest eigenvalue of } S$$

Smallest eigenvalue of $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$ is 0 with
corresponding eigenvector $D^{\frac{1}{2}}\mathbf{e}$ because

$$D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}} \cdot D^{\frac{1}{2}}\mathbf{e} = D^{-\frac{1}{2}}(D - W)\mathbf{e} = 0$$

But $\mathbf{g}'$ cannot be $D^{\frac{1}{2}}\mathbf{e}$ according to the constraint $\mathbf{g}'^\top D^{\frac{1}{2}}\mathbf{e} = 0$

Therefore, solution $\mathbf{g}'$ is the eigenvector of $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$
corresponding to **second smallest** eigenvalue, aka second
eigenvector

Spectral Graph Theory

- ▶ Possible approach

- ▶ Represent a similarity graph as a matrix
- ▶ Apply knowledge from Linear Algebra...

- The eigenvalues and eigenvectors of a matrix provide global information about its structure.

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

- *Spectral Graph Theory*

- Analyse the “spectrum (光谱)” of matrix representing a graph.
- *Spectrum*: The eigenvectors of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues.

$$\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$$

$$\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$$

So far...

- ▶ How to define a “good” partition of the 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

► Ratio Cut

$$\min_{\mathbf{h}} \frac{\mathbf{h}^\top (D - W) \mathbf{h}}{\mathbf{h}^\top \mathbf{h}} \quad \text{s.t.} \quad \mathbf{h}^\top \mathbf{e} = 0$$

► Normalized Cut

$$\min_{\mathbf{g}'} \frac{\mathbf{g}'^\top D^{-\frac{1}{2}} (D - W) D^{-\frac{1}{2}} \mathbf{g}'}{\mathbf{g}'^\top \mathbf{g}'} \quad \text{s.t.} \quad \mathbf{g}'^\top D^{\frac{1}{2}} \mathbf{e} = 0$$

Spectral Clustering Algorithms

- ▶ Construct the affinity matrix W from data (A in community detection setting)
- ▶ Compute the second vector of the matrix
 - ▶ $D - W$ Ratio Cut
 - ▶ $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$ Normalized Cut
- ▶ Taking the signs of the second eigenvector and cluster the data

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 Clustering Algorithms

1) Pre-processing:

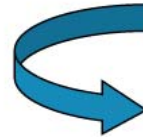
- Build Laplacian matrix L of the graph



	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 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?

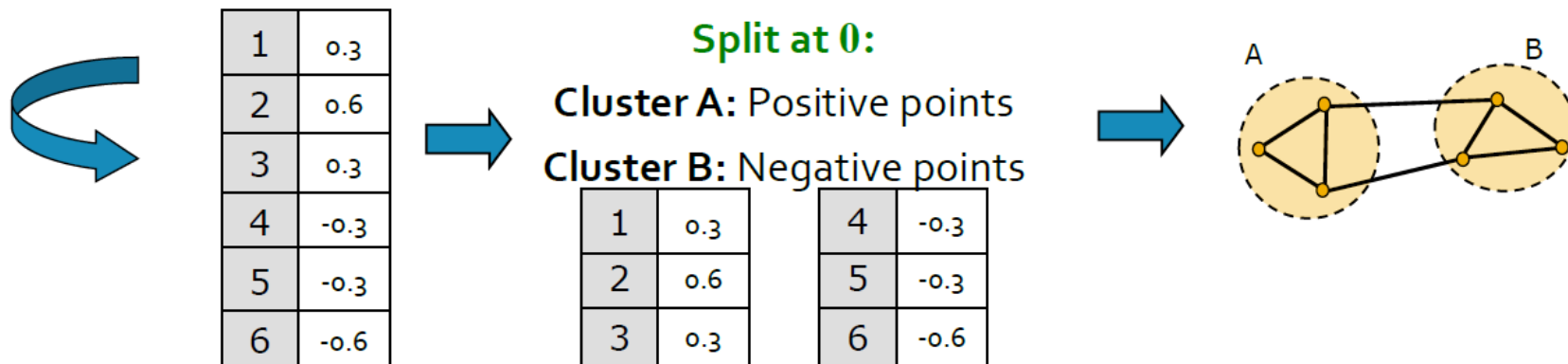
Spectral Clustering Algorithms

■ 3) Grouping:

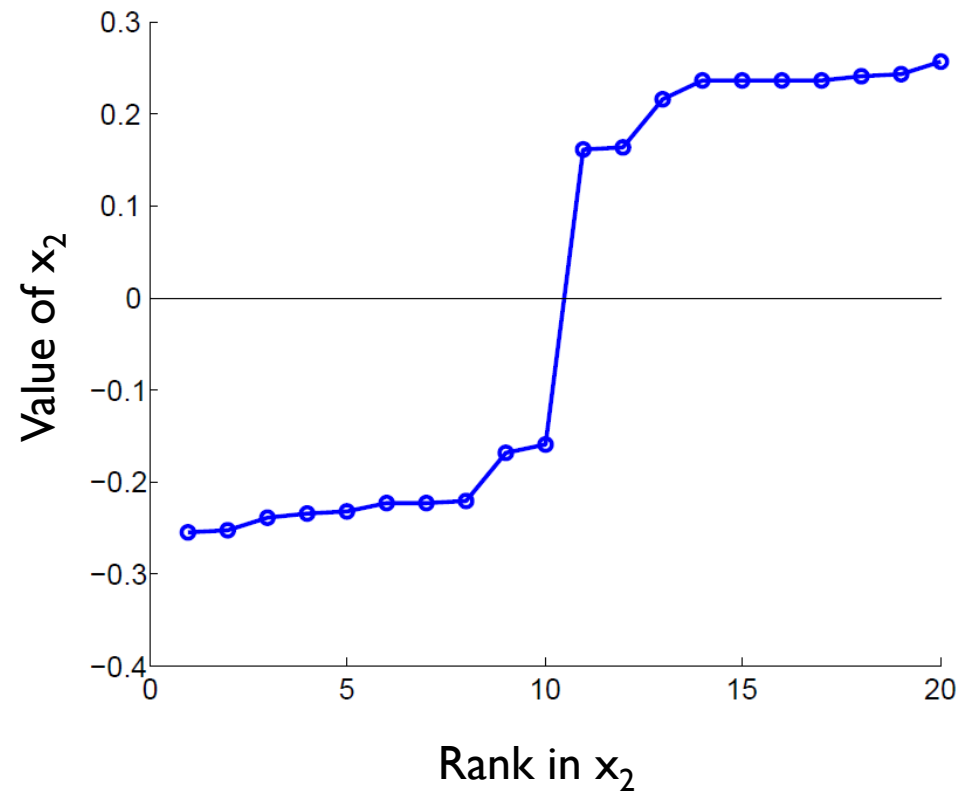
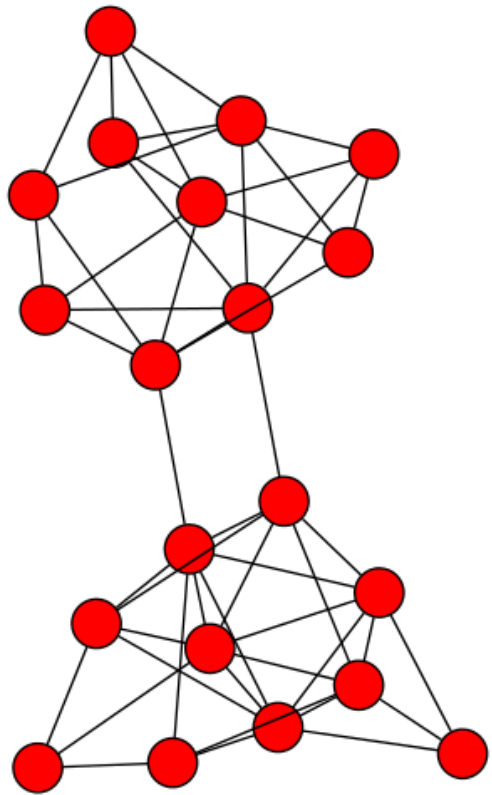
- Sort components of reduced 1-dimensional vector
- Identify clusters by splitting the sorted vector in two

■ How to choose a splitting point?

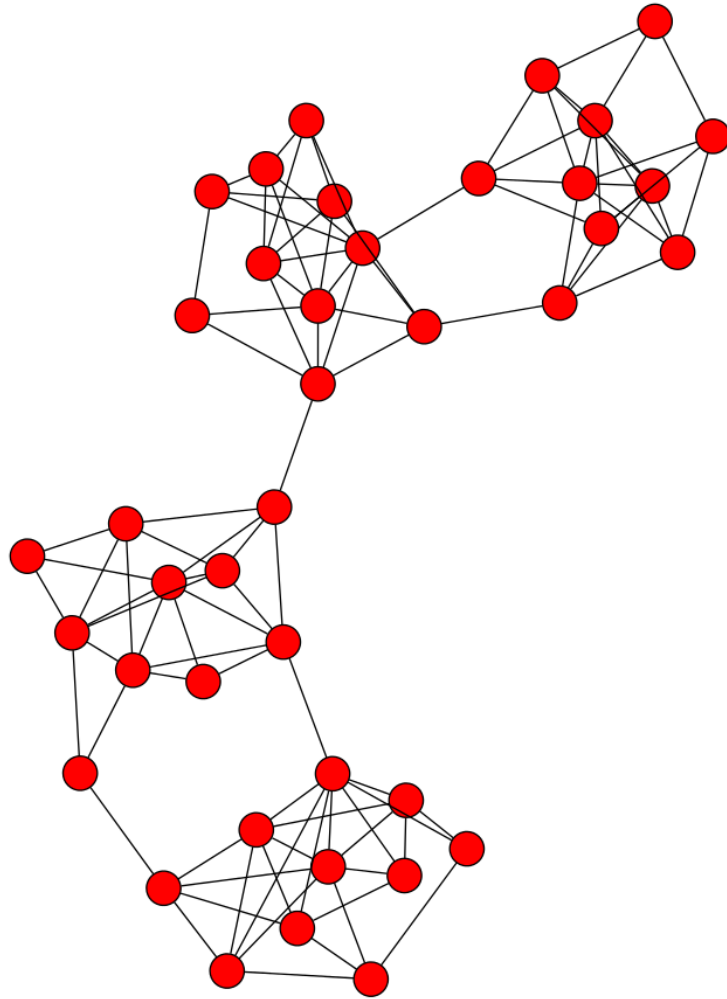
- Naïve approaches:
 - Split at **0** or median value
- More expensive approaches:
 - Attempt to minimize normalized cut in 1-dimension (sweep over ordering of nodes induced by the eigenvector)



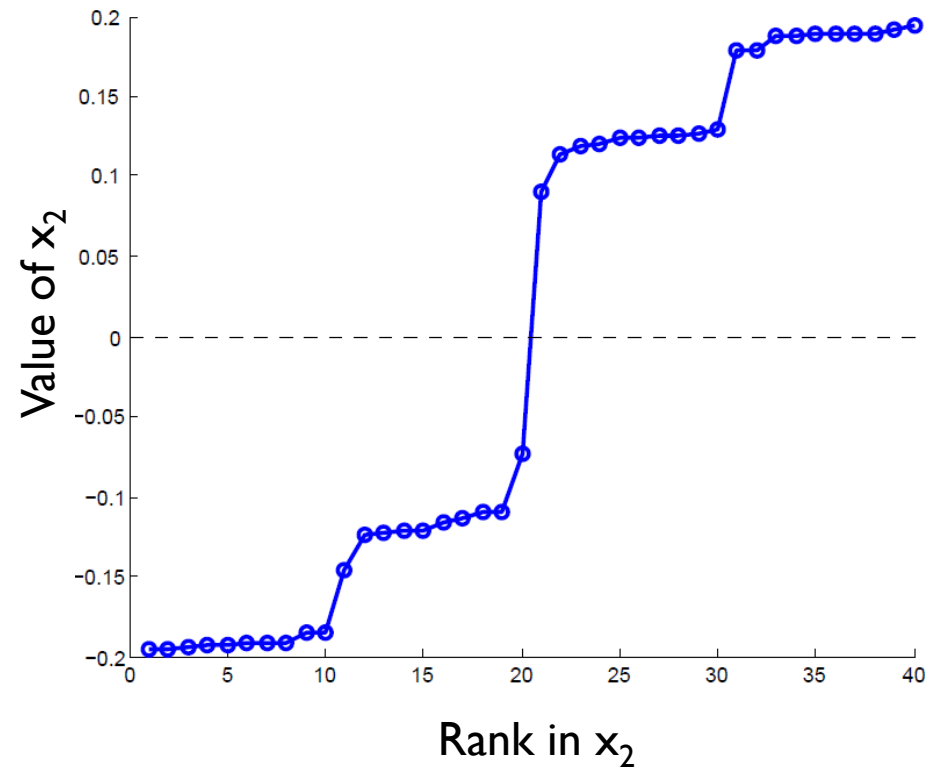
Example



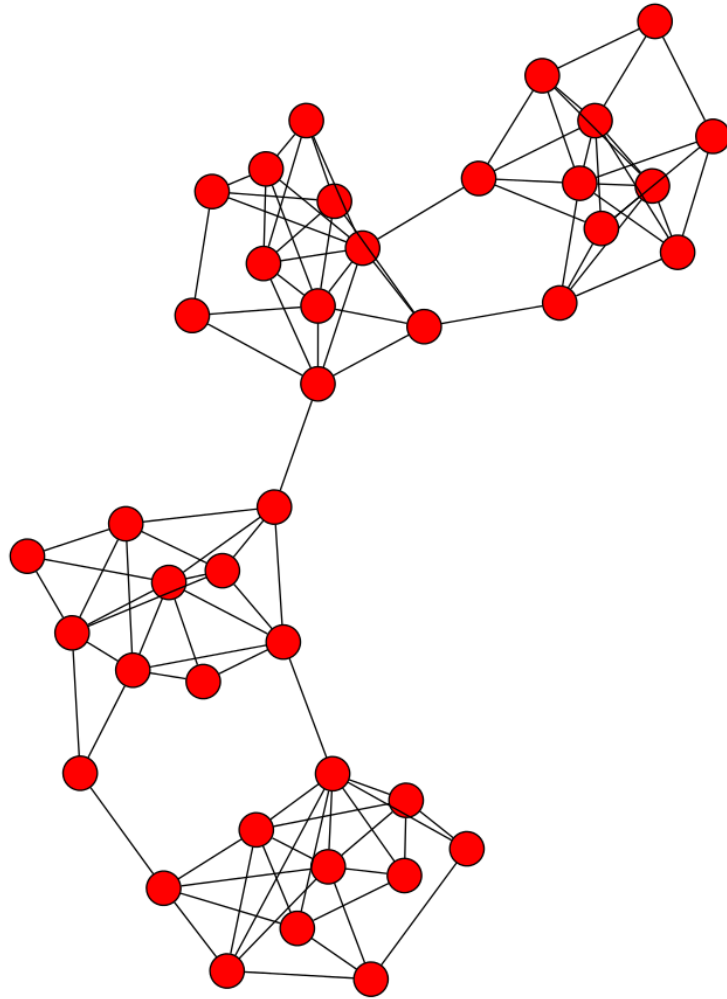
Example



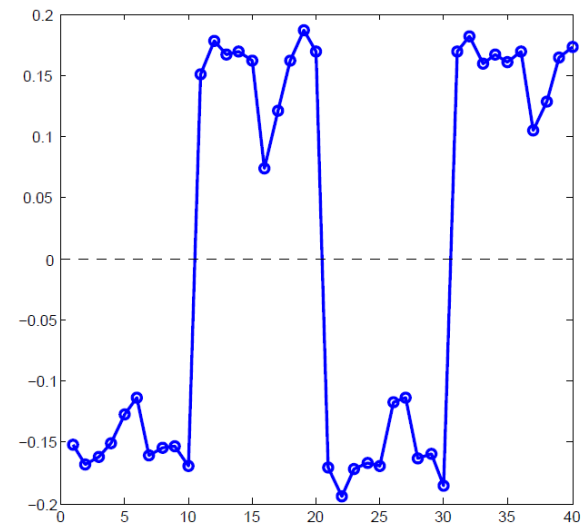
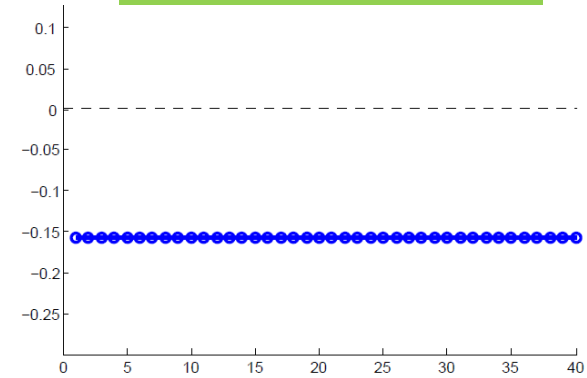
Components of x_2



Example



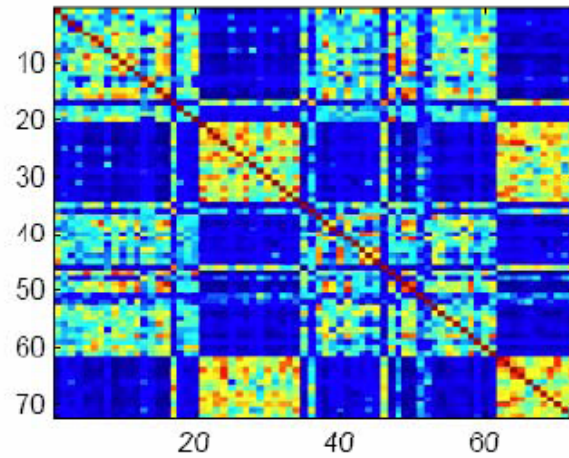
Components of x_1



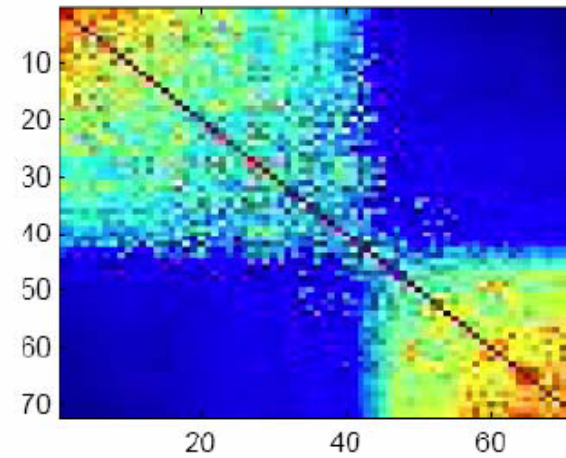
Components of x_3

Example

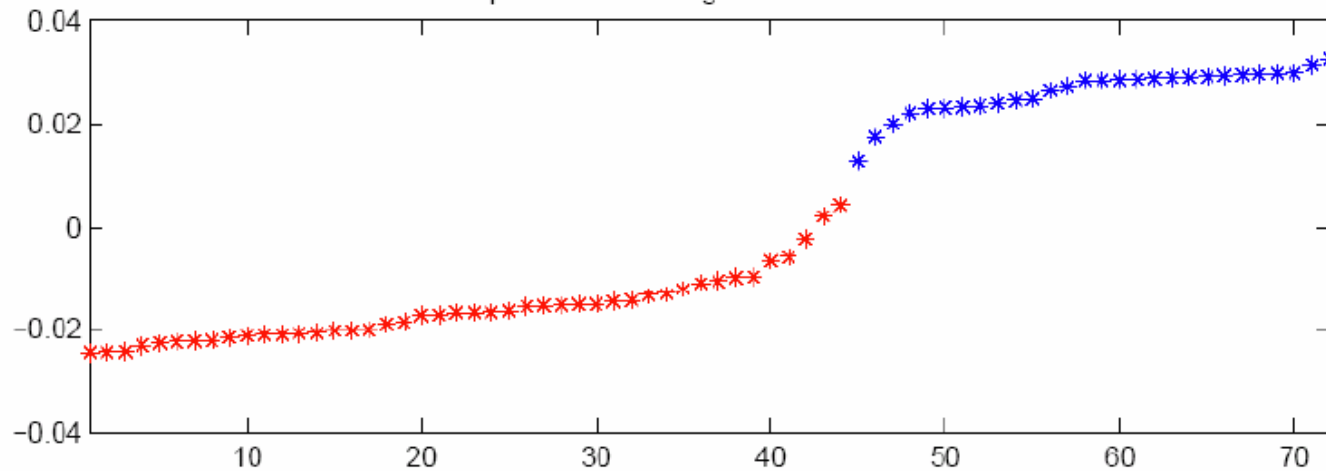
input affinity matrix



affinity matrix reordered according to solution vector



the partition according to the solution vector



K-way Spectral Clustering

- ▶ How do we partition a graph into k clusters?
- ▶ Two basic approaches:
 - ▶ Recursive bi-partitioning
 - ▶ Recursively apply bi-partitioning algorithm in a hierarchical divisive manner
 - ▶ Disadvantages: Inefficient, unstable
 - ▶ Cluster multiple eigenvectors
 - ▶ Build a reduced space from multiple eigenvectors
 - ▶ Commonly used in recent papers
 - ▶ A preferable approach...

K-way Spectral Clustering Algorithm

Three basic stages:

1) Pre-processing

- ▶ Construct a matrix representation of the graph ($n \times n$ matrix)

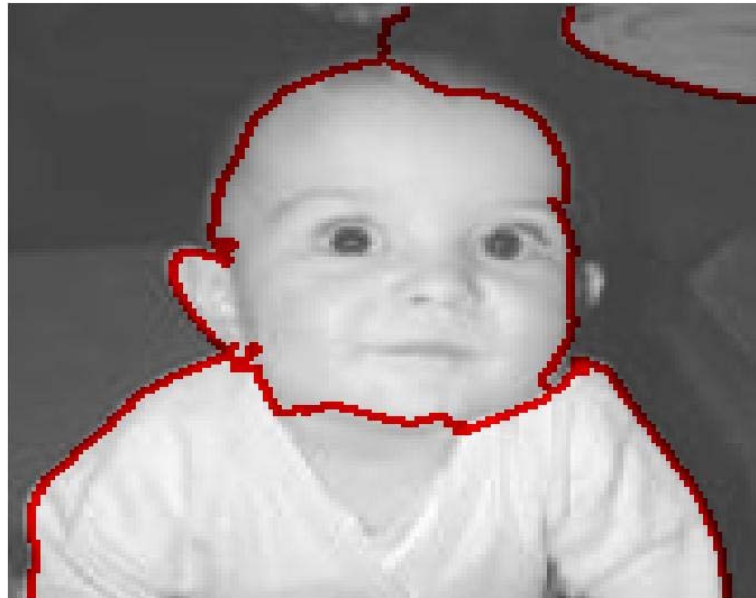
2) Decomposition

- ▶ Compute eigenvalues and eigenvectors of the matrix
- ▶ Map each point to a lower-dimensional representation based on smallest k eigenvectors ($V: n \times k$ matrix)

3) Grouping

- ▶ Assign points to k clusters, by running k -means on the new representation ($kmeans(V, k)$)

Normalized Cut in Image Segmentation



Comparison with K-means



Normalized Cuts



K-means Segmentation

Spectral Clustering Summary

- ▶ Algorithms that cluster points using eigenvectors of matrices derived from the data
- ▶ Useful in hard non-convex clustering problems
- ▶ Obtain data representation in the low-dimensional space that can be easily clustered
- ▶ Empirically very successful