Data Mining and Matrices

07 – Graphs

Rainer Gemulla, Pauli Miettinen

Jun 6, 2013

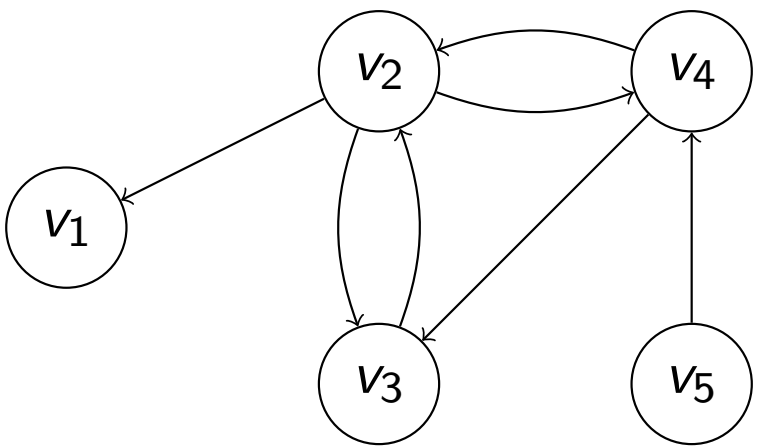
Graph mining

- Graphs everywhere
  - Internet
  - World wide web
  - Social networks
  - Protein-protein interactions
  - Similarity graphs
  - ...
- Goals of graph mining
  - As data mining: classification, clustering, outliers, patterns
  - Output often also one or more graphs
  - Interesting subgraphs (e.g., communities, near-cliques, clusters)
  - Important vertices (e.g., influential bloggers, PageRank, outliers)
  - Web mining (e.g., topic predicition, classification)
  - Web usage mining (e.g., frequent subgraphs, patterns)
  - Recommender systems (e.g., movie recommendation, edge prediction)
  - ...

Spectral analysis of matrices associated with graphs is an important tool in graph mining. Our focus: spectral clustering and link analysis.

A graph is a matrix is a graph

- Let $G = (V, E)$ be a (weighted) graph
- Vertices $V = \{ v_1, \ldots, v_n \}$
- Edge $(i, j) \in E$ has positive weight w_{ij} (or 1 if graph is unweighted)
- Convention: absent edges $(i, j) \notin E$ have weight $w_{ij} = 0$
- **Adjacency matrix** $\mathbf{W}$ is $n \times n$ matrix with $\mathbf{W}_{ij} = w_{ij}$
- Undirected graph $\iff$ $\mathbf{W}$ symmetric ($\mathbf{W} = \mathbf{W}^T$)
- Degree of vertex i given by $d_i = \sum_j w_{ij} = \mathbf{W}_{i*}\mathbf{1}$
- **Degree matrix** $\mathbf{D}$ is $n \times n$ diagonal matrix with $\mathbf{D}_{ii} = d_i$



G

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

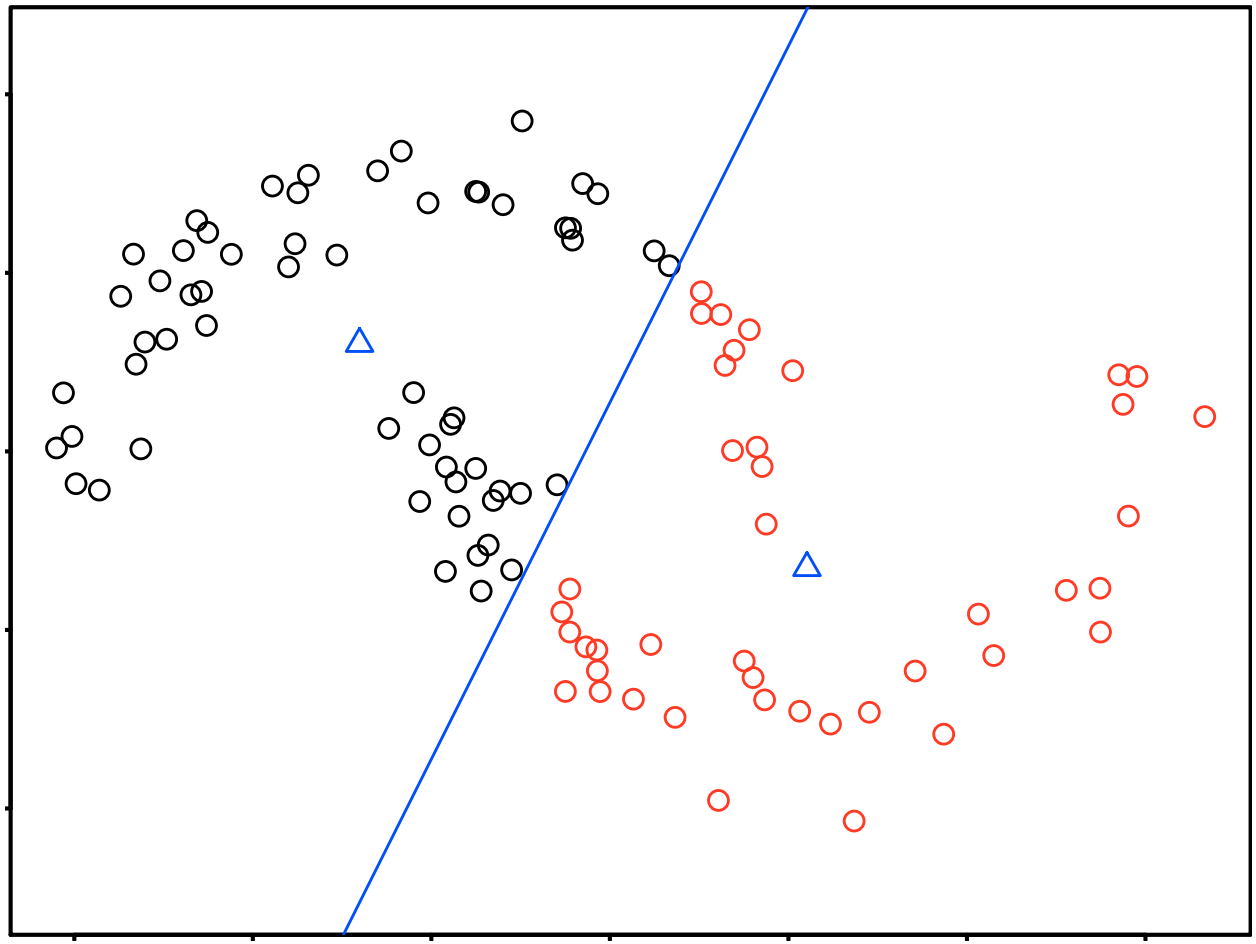
$\mathbf{W}$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$\mathbf{D}$

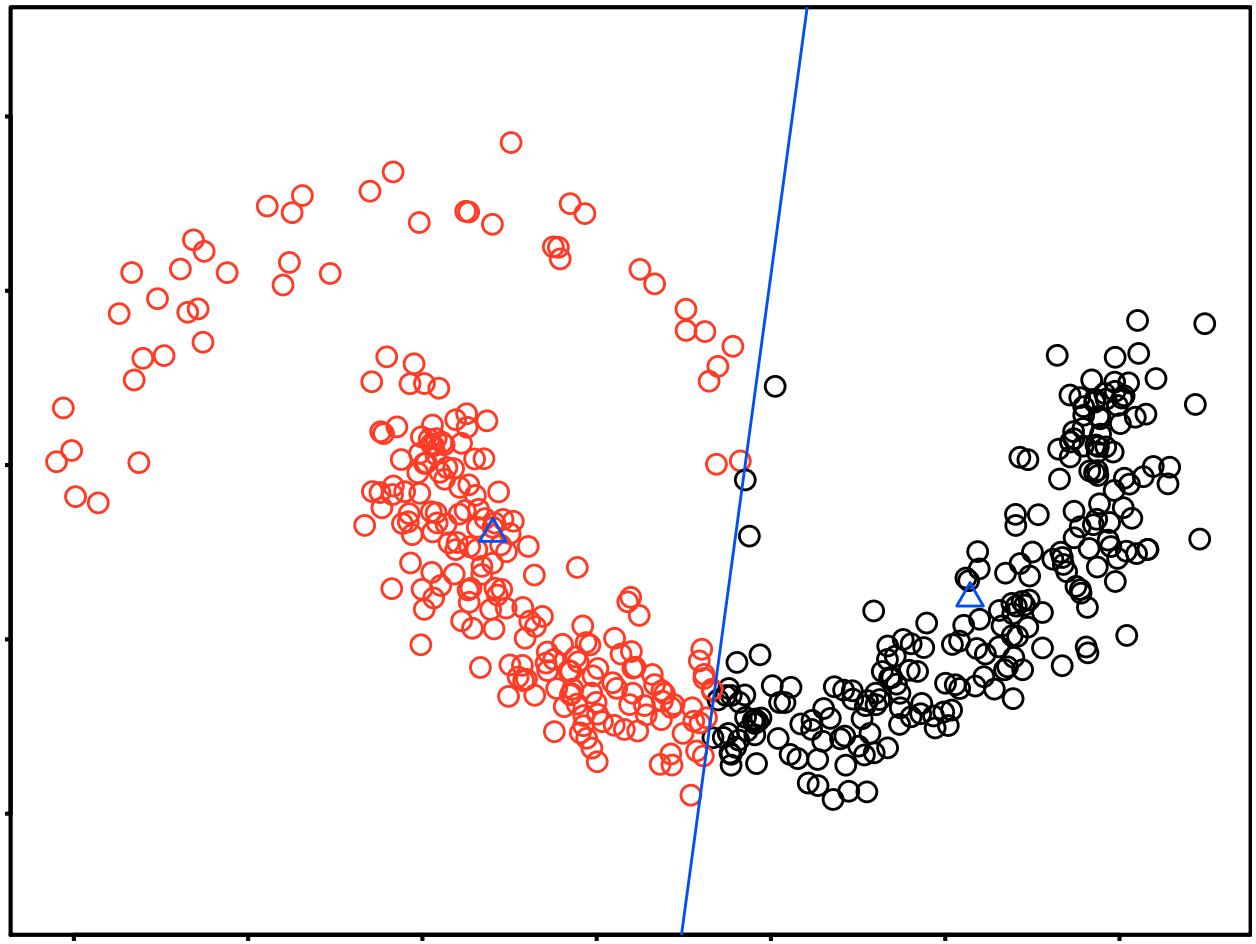
Outline

k-Means example (1)



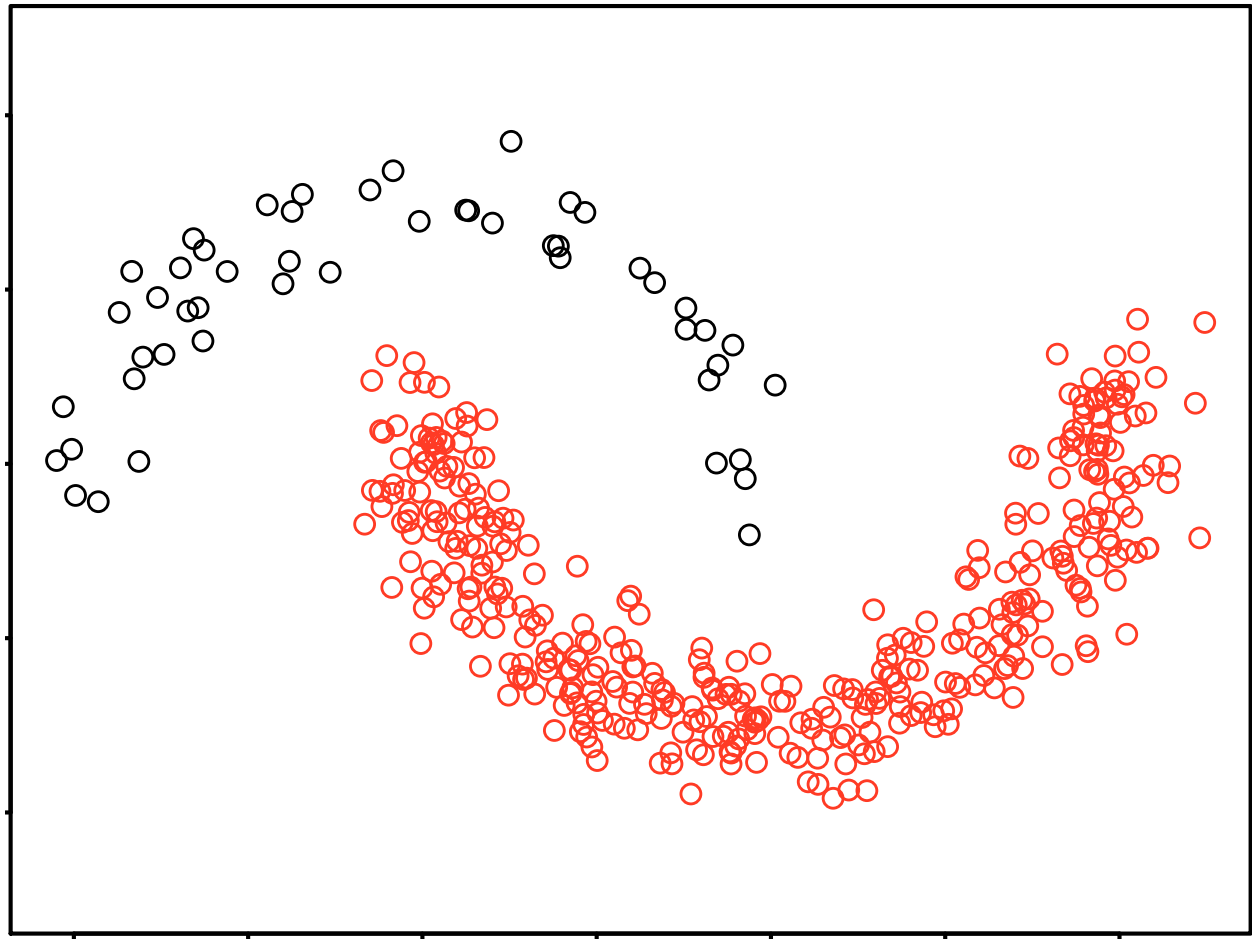
k-Means cannot detect non-convex clusters well.

k-Means example (2)



k-Means is sensitive to skew in cluster sizes.

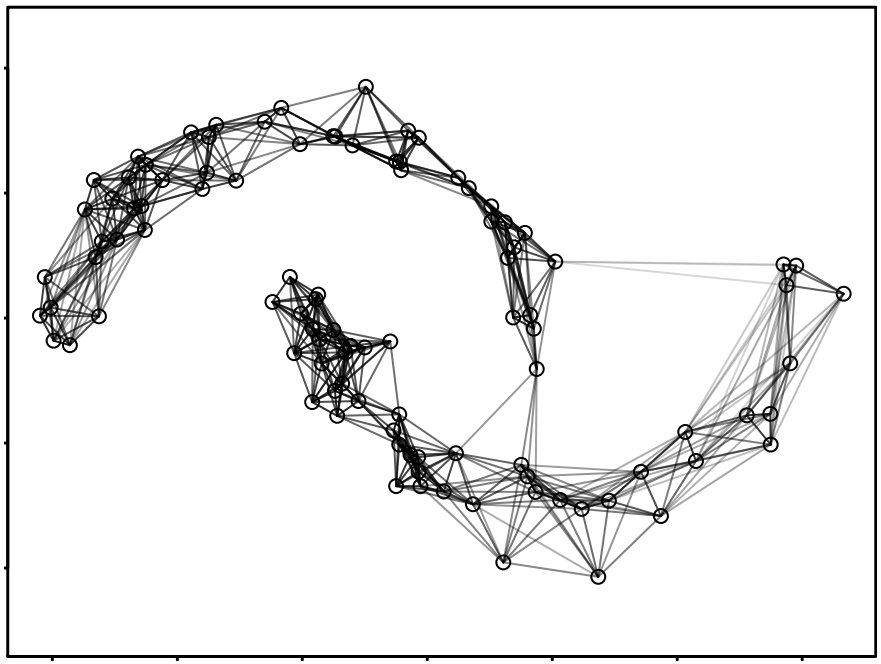
A better clustering



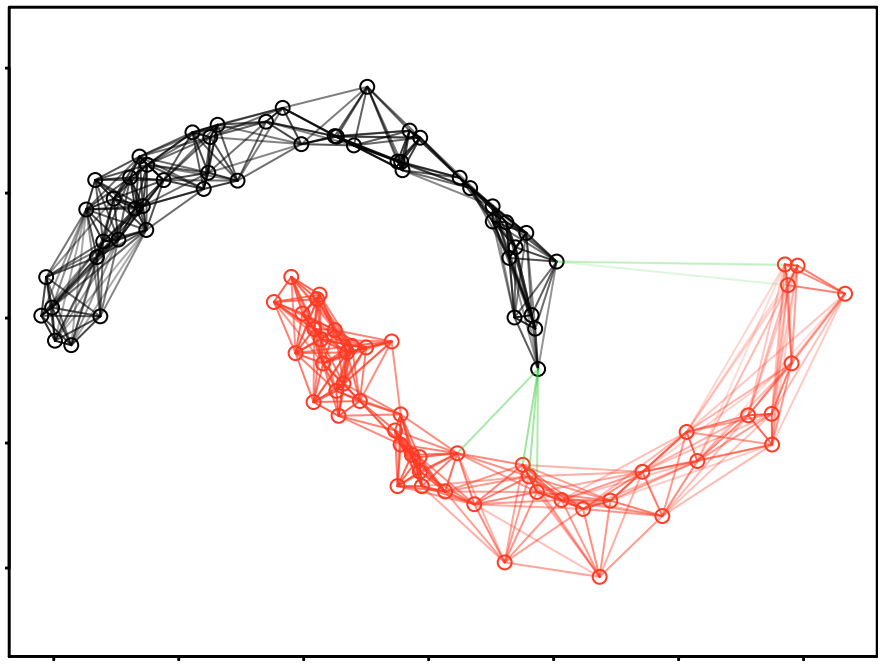
In this clustering, points within a cluster are close to their neighbors, but not necessarily to all the points in the cluster.

Graph-based clustering

1. Given a dataset, construct a *similarity graph* modeling local neighborhood relationships
2. Partition the similarity graph using suitable *graph cuts*



Similarity graph



Clustering

Discussion

- Clustering
  1. Points within a cluster should be similar
  2. Points in different clusters should be dissimlar

- k-Means is global
  1. *All* points within a cluster should be similar (close)
  2. Points in different clusters should be dissimilar (far apart)

- Graph-based clustering is local
  1. *Neighboring* points within a cluster should be similar (close)
  2. Points in different clusters should be dissimilar (far apart)

Which cut? (1)

- $G = (V, E)$: Undirected, weighted similarity graph
- $A \subset V$, $\bar{A} = V \setminus A$
- A and $\bar{A}$ form a partitioning of V into two clusters
- **Minimum cut**

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

- Can be solved efficiently (in P)
- Often not useful in practice, e.g., may separate a single vertex
  $\rightarrow$ Need to balance cut weight and cluster sizes

Which cut? (2)

- **Minimum ratio cut** (penalize different sizes w.r.t. vertices)

$$\text{RatioCut}(A, \bar{A}) = \sum_{i \in A, j \in \bar{A}} w_{ij} \left(\frac{1}{|A|} + \frac{1}{|\bar{A}|} \right)$$

- **Minimum normalized cut** (penalize different sizes w.r.t. edges)

$$\text{Ncut}(A, \bar{A}) = \sum_{i \in A, j \in \bar{A}} w_{ij} \left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(\bar{A})} \right),$$

where $\text{vol}(A) = \sum_{i \in A} d_i = \sum_{i,j \in A} w_{ij}$

- Unfortunately, both problems are NP-hard

Spectral clustering is a relaxation of RatioCut or Ncut, is simple to implement, and can be solved efficiently.

Which cut? (3)

- Recall clustering objectives
  1. Points in different clusters should be dissimilar (minimize between-cluster similarity)
  2. Points in same cluster should be similar (maximize within-cluster similarity)
- (1) = minimize $\text{cut}(A, \bar{A})$
- (2) = $\text{vol}(A)$ and $\text{vol}(\bar{A})$ are both large
- cut, RatioCut, and Ncut all implement (1)
- Only Ncut additionally implements (2)
- Ncut achieves both goals $\rightarrow$ usually good choice

Outline

From distances to similarities

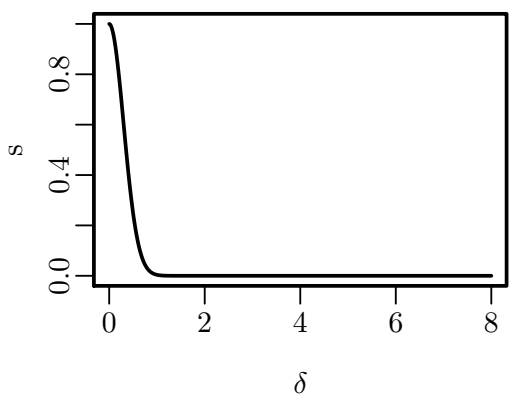
- Need to "convert" distances to similarities
- Large distance $\delta_{ij} \iff$ small similarity w_{ij} (and vice versa)
- Simplest choice: reciprocal (problematic, unbounded)

$$w_{ij} = \frac{1}{\delta_{ij}}$$

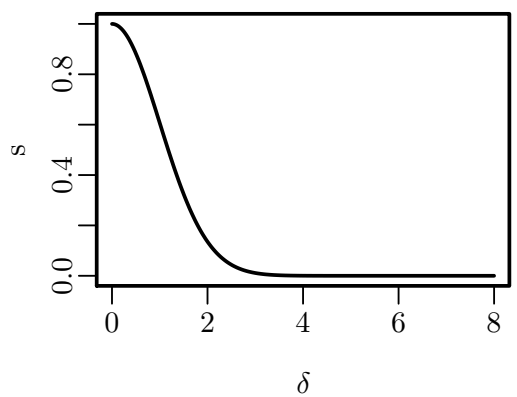
- Common choice: **Gaussian similarity function** (in $[0, 1]$)

$$w_{ij} = \exp(-\delta_{ij}/(2\sigma^2))$$

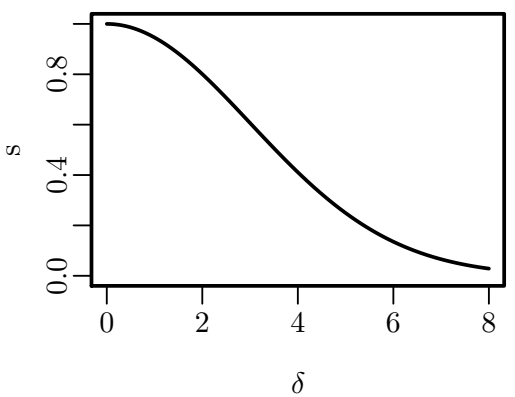
- Parameter σ controls what is considered local (large σ = large neighborhood)



$\sigma = 0.3$

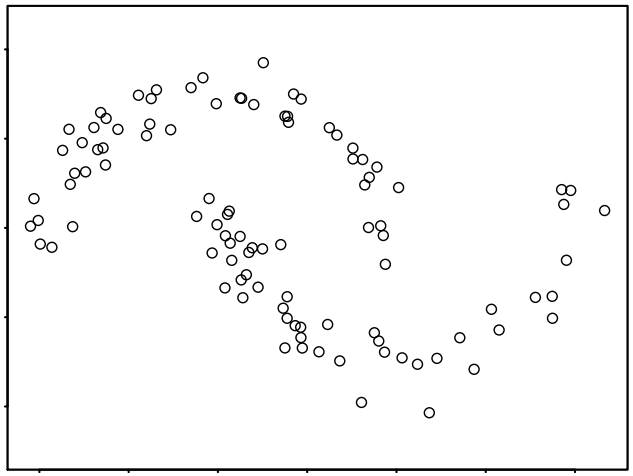


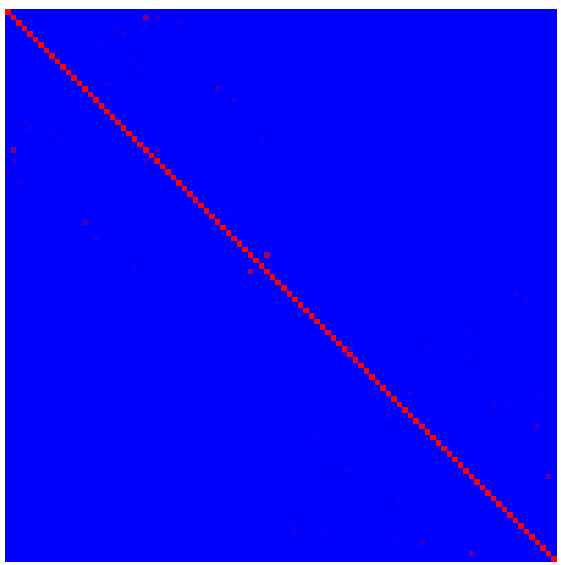
$\sigma = 1$



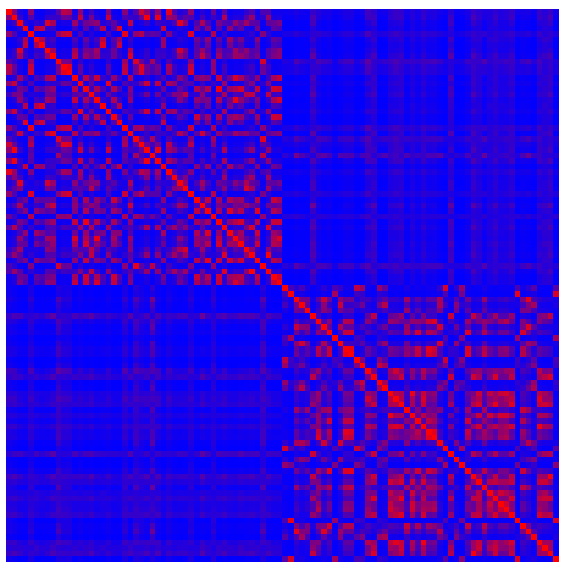
$\sigma = 3$

From distances to similarities (examples)





$\sigma = 0.1$
(too small)

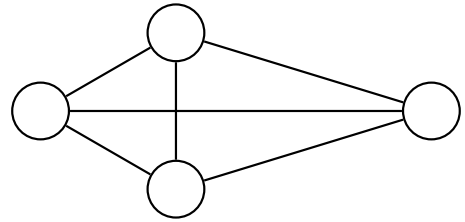


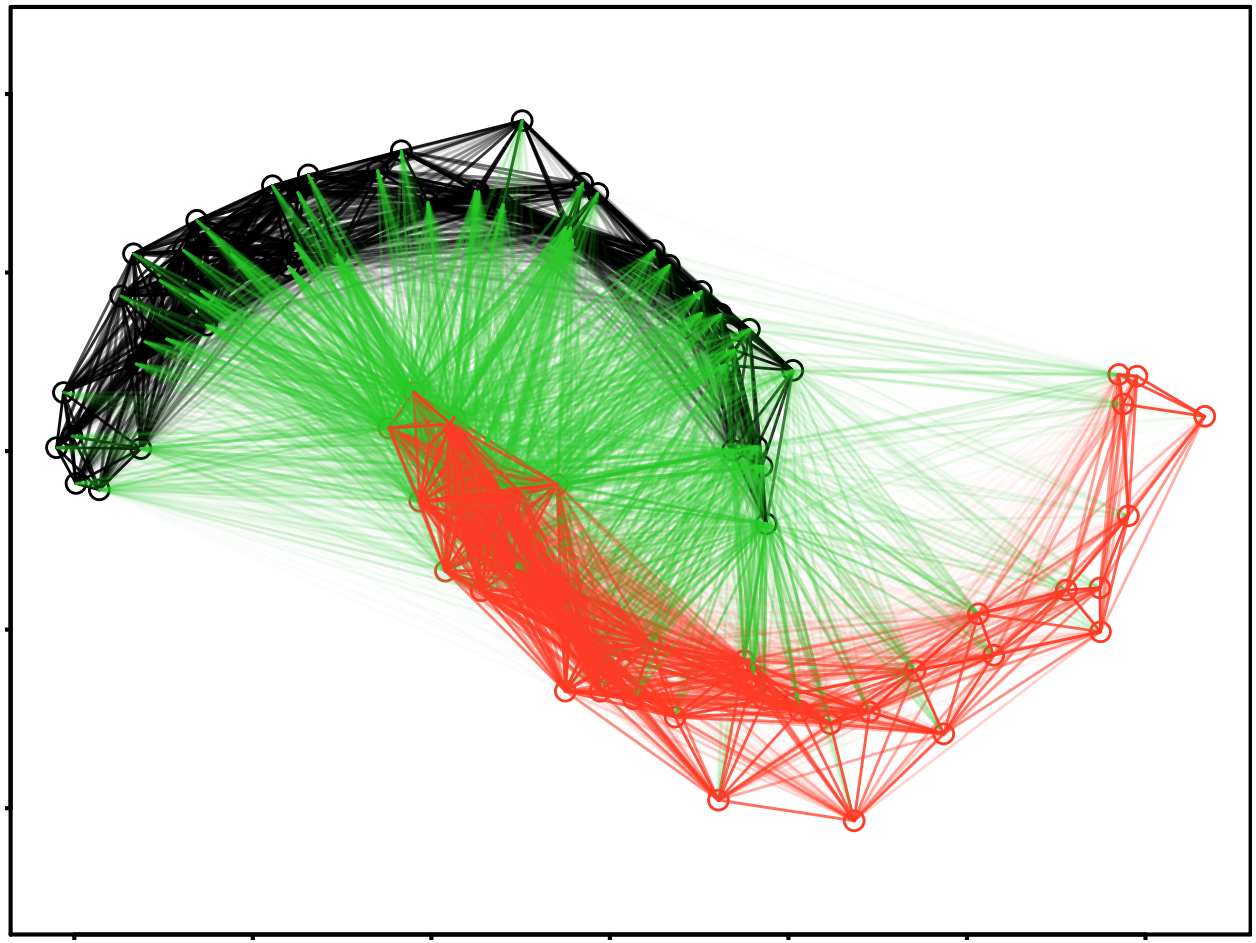
$\sigma = 0.5$
(good)

$\sigma = 3$
(too large)

Full graph

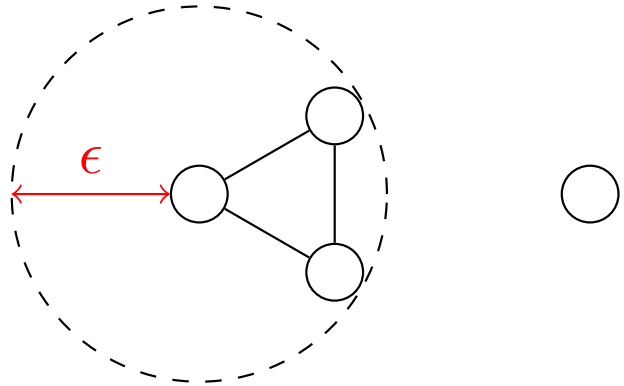
- Connect all pairs of vertices
- Weigh edges by similarity
- Generally expensive, not feasible for large datasets

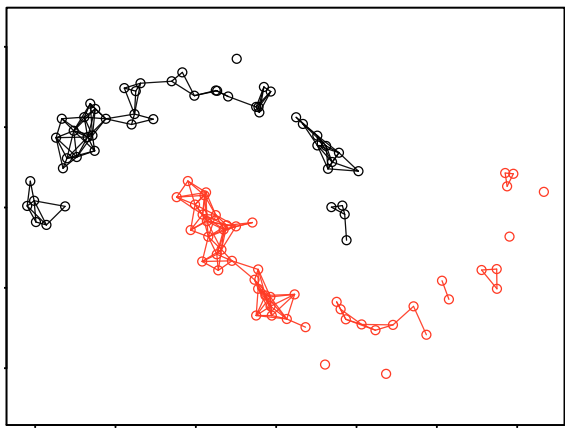




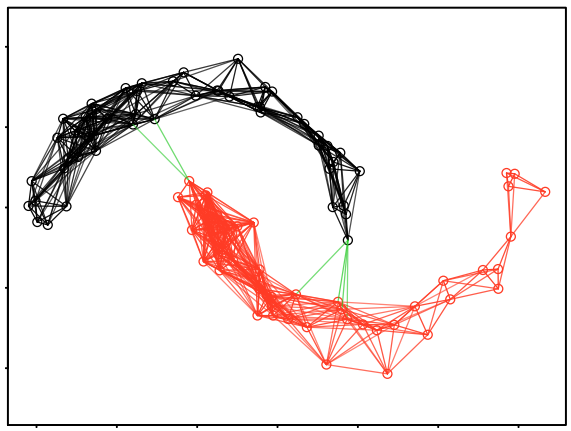
ϵ-Neighborhood graph

- Pick neighborhood size ϵ
- Connect vertices of distance $\leq \epsilon$
- Unweighted or weighted by similarity

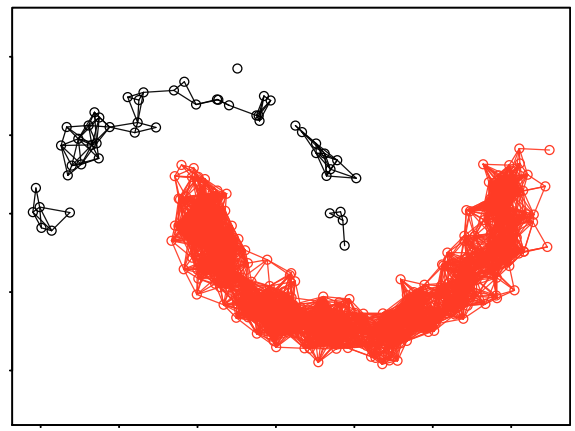




ϵ too small



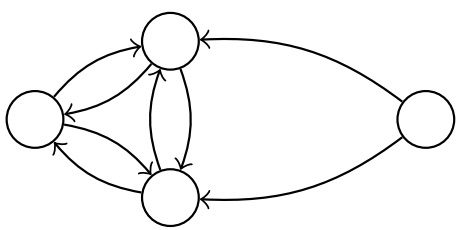
ϵ good



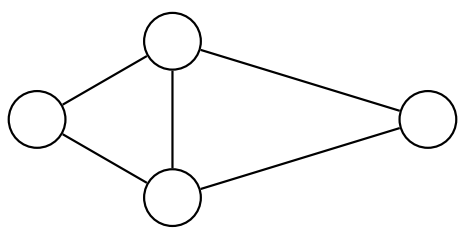
Skewed clusters: ϵ too large for red, too small for black

Nearest neighbor graphs

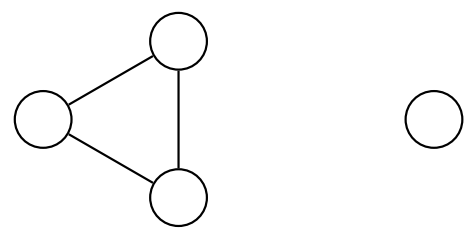
- Pick number k of neighbors
- Directed k-nearest neighbor graph
  - Add directed edge (i,j) if j is among k closest neighbors of i
  - But: need undirected graph for well-defined similarities
- (Symmetric) k-nearest neighbor graph
  - Connect (i, j) if (i, j) or (j, i) in directed kNN-graph (OR)
  - Each node has at least k, but potentially more than k "neighbors"
- Mutual k-nearest neighbor graph
  - Connect (i, j) if (i, j) and (j, i) in directed kNN-graph (AND)
  - Each node has at most k, but potentially less than k "neighbors"
- Weigh edges by similarity



directed



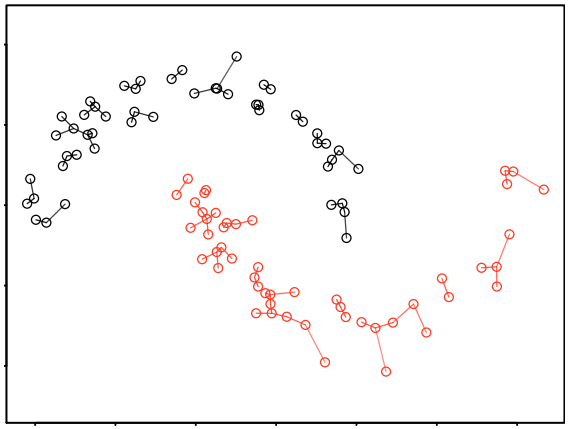
symmetric



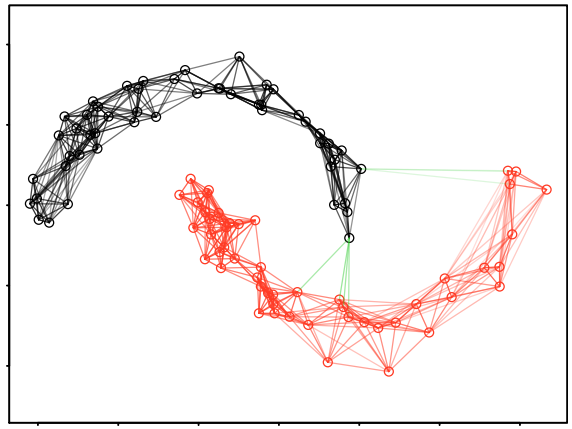
mutual

k-Nearest neighbor graph (examples)

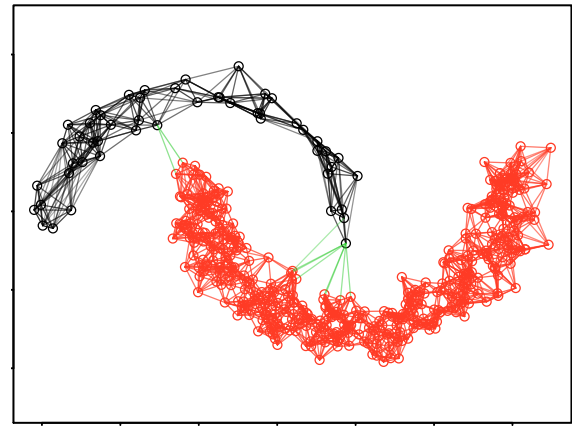
Symmetric kNN graph



$k = 1$ (too small)

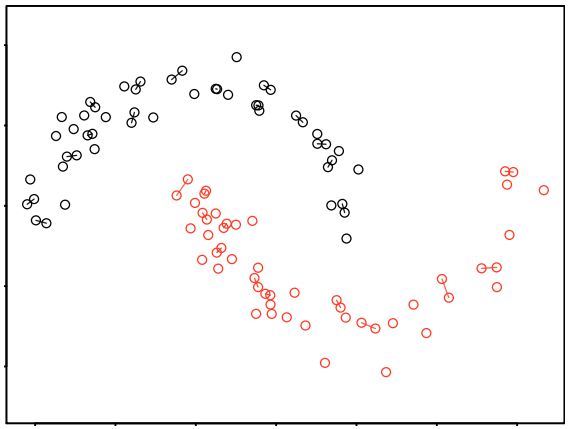


$k = 10$ (good)

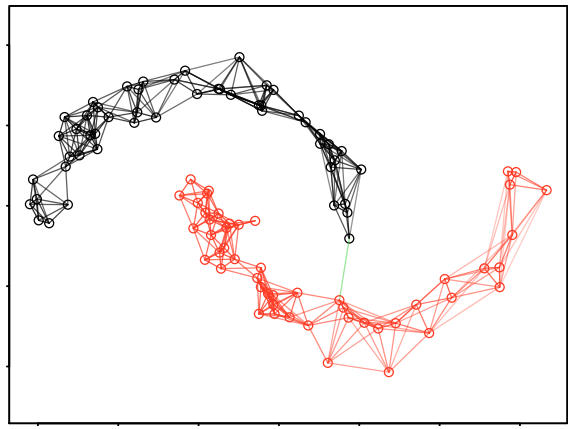


Skewed, $k = 10$ (good)

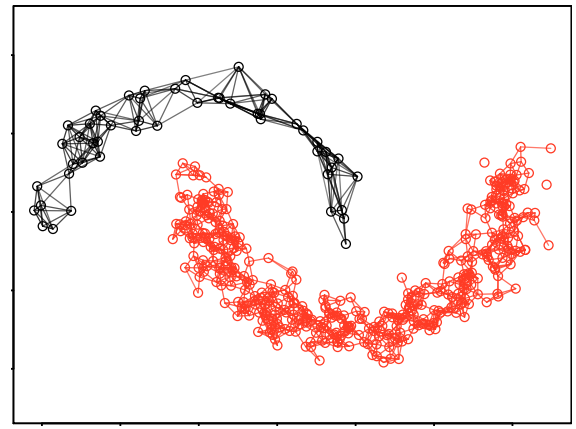
Mutual kNN graph



$k = 1$ (too small)



$k = 10$ (good)



Skewed, $k = 10$ (too small)

Discussion (1)

- Construction of similarity graph non-trivial and not well-understood
- Clustering results sensitive to choice of graph
- Which similarity function?
  - Should capture similarity of most-similar objects well (other edges pruned by neighborhood graphs)
  - Gaussian similarity function common choice for data in Euclidean space
  - Generally application-dependent
- Which graph?
  - Fully connected graph requires suitable similarity function, dense similarity matrix
  - ϵ-neighborhood graph cannot deal well with clusters of different densities
  - kNN graph can connect points in regions with different densities $\rightarrow$ Generally recommended choice, sparse similarity matrix
  - Mutual kNN graph is somewhere in between

Discussion (2)

- Which parameters? (ϵ, k, σ)
  - ϵ and k should be small so that similarity matrix is sparse
  - But large enough to ensure that similarity graph is connected (or at least has fewer components than desired clusters)
  - Otherwise: clustering sizes arbitrarily unbalanced, sensitive to outliers
  - kNN: try various values (start with, e.g., $k = O(\log(n))$
  - Mutual kNN: no good heuristics known
  - ϵN: around length of longest edge in minimal spanning tree (problematic with outliers or clusters that are far apart)
  - σ: neighbors with similarity significantly larger than 0 "neither too small nor too large" (e.g., mean distance to k-th nearest neighbor, or ϵ as above)

Skilled data miners do not run out of jobs.

Outline

Graph Laplacian

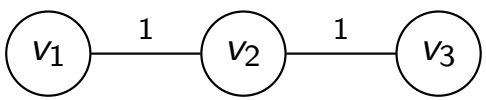
Definition

Let G be an undirected graph with positive edge weights. Denote by $\mathbf{W}$ the (weighted) adjacency matrix of G, and by $\mathbf{D}$ the degree matrix of G. Then

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

is called the *(unnormalized) graph Laplacian* of G.

Note that self edges ($w_{ii} > 0$) do not affect the graph Laplacian.



G

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$\mathbf{D}$

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$\mathbf{W}$

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$

$\mathbf{L}$

Graph Laplacians are the main tool for spectral clustering, but they have many other uses too (e.g., label propagation, graph drawing).

Properties of the graph Laplacian (1)

Theorem

For every vector $\mathbf{x} \in \mathbb{R}^n$, *we have* $f(\mathbf{x}) = \mathbf{x}^T \mathbf{L}\mathbf{x} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(x_i - x_j)^2$.

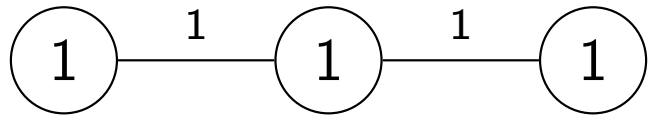
- $\mathbf{x}$ assigns a real value to each vertex
- $f(\mathbf{x})$ is a quadratic form and small when "similar" vertices—i.e., vertices connected with high-weight edges—take similar values

Proof.

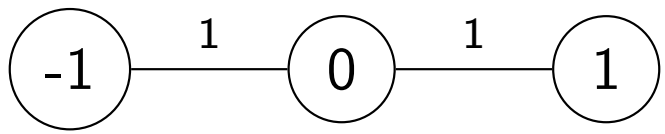
$$\begin{aligned}
\mathbf{x}^T \mathbf{L}\mathbf{x} &= \mathbf{x}^T \mathbf{D}\mathbf{x} - \mathbf{x}^T \mathbf{W}\mathbf{x} = \sum_{i=1}^{n} d_i x_i^2 - \sum_{i,j=1}^{n} w_{ij} x_i x_j \\
&= \frac{1}{2} \left(\sum_{i=1}^{n} d_i x_i^2 - 2 \sum_{i,j=1}^{n} w_{ij} x_i x_j + \sum_{j=1}^{n} d_j x_j^2 \right) \\
&= \frac{1}{2} \left(\sum_{i,j=1}^{n} w_{ij}(x_i^2 - 2x_i x_j + x_j^2) \right) = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(x_i - x_j)^2
\end{aligned}$$

Properties of the graph Laplacian (2)

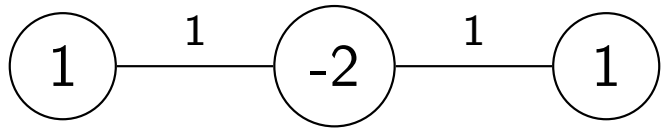
$$\mathbf{x}^T \mathbf{L} \mathbf{x} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (x_i - x_j)^2$$



$$\mathbf{x}^T \mathbf{L} \mathbf{x} = 0$$



$$\mathbf{x}^T \mathbf{L} \mathbf{x} = 1$$



$$\mathbf{x}^T \mathbf{L} \mathbf{x} = 9$$

Properties of the graph Laplacian (3)

Theorem

$\mathbf{L}$ *is symmetric and positive semi-definite.*

- Implies that $f(\mathbf{x}) = \mathbf{x}^T\mathbf{L}\mathbf{x}$ is a convex function
- Implies that $\mathbf{L} = \mathbf{A}^T\mathbf{A}$ for some $\mathbf{A}$ ($=$ incidence matrix)

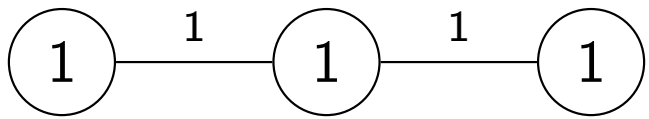
Proof.

Since $\mathbf{D}$ and $\mathbf{W}$ are symmetric, so is $\mathbf{L}$. Since $\mathbf{x}^T\mathbf{L}\mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{L}$ is positive semi-definite. □

Properties of the graph Laplacian (4)

Theorem

The smallest eigenvalue of $\mathbf{L}$ *is zero, the corresponding eigenvector is constant one vector* $\mathbf{1}$.



$$\lambda_3 = 0$$

Proof.

The row sums $\mathbf{L1} = \mathbf{0}$ by construction. □

Properties of the graph Laplacian (5)

Theorem

All eigenvalues are non-negative and real-valued, i.e.,
$\lambda_1 \geq \ldots \leq \lambda_{n-1} \geq \lambda_n = 0$.

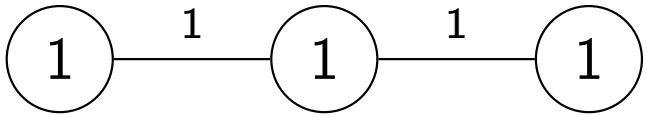
Proof.

All eigenvalues of a symmetric matrix are real. If $\mathbf{Lv} = \lambda \mathbf{v}$, then
$0 \leq \mathbf{v}^T \mathbf{Lv} = \lambda \|\mathbf{v}\|^2$ and thus $\lambda \geq 0$. □

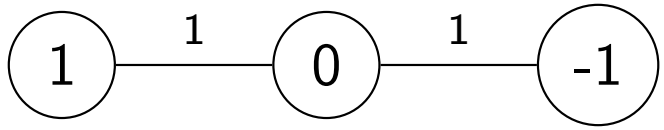
Connected graphs

Theorem

If G is connected, then eigenvalue 0 has multiplicity 1, i.e., $\lambda_{n-1} > 0$.



$\lambda_3 = 0$



$\lambda_2 = 1 > 0$

Proof.

Recall that $\mathbf{1}$ is an eigenvector of $\mathbf{L}$ with eigenvalue 0. Suppose that $\mathbf{0} \neq \mathbf{v} \neq c\mathbf{1}$ is an eigenvector of $\mathbf{L}$ with eigenvalue λ. Since G is connected, this implies that there are two neighboring vertices i' and j' such that $v_{i'} \neq v_{j'}$. Now

$$\lambda \|\mathbf{v}\|^2 = \mathbf{v}^T \mathbf{L} \mathbf{v} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(v_i - v_j)^2 \geq w_{i'j'}(v_{i'} - v_{j'})^2 > 0$$

so that $\lambda > 0$. □

Connected components

Theorem

The multiplicity k of eigenvalue 0 is equal to the number of connected components $G_1, \ldots, G_k$ of G. The corresponding eigenspace is spanned by the indicator vectors $\mathbf{1}_{G_i}$ (value 1 for vertices in V_i, value 0 otherwise).

Proof.

Let $\mathbf{L}_1, \ldots, \mathbf{L}_k$ be the graph Laplacian of the connected components. Order w.l.o.g. the vertices by their connected components. Then

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_1 & & & \\ & \mathbf{L}_2 & & \\ & & \ddots & \\ & & & \mathbf{L}_k \end{pmatrix}.$$

Since $\mathbf{L}$ is block-diagonal, the spectrum of $\mathbf{L}$ is given by the union of the spectra of $\mathbf{L}_i$. The corresponding eigenvectors are the eigenvectors of $\mathbf{L}_i$, filled with 0 at the position of the other blocks. □

Connected components (example)

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

$\lambda_6 = 0$

$\lambda_5 = 0$

$\lambda_4 = 1$

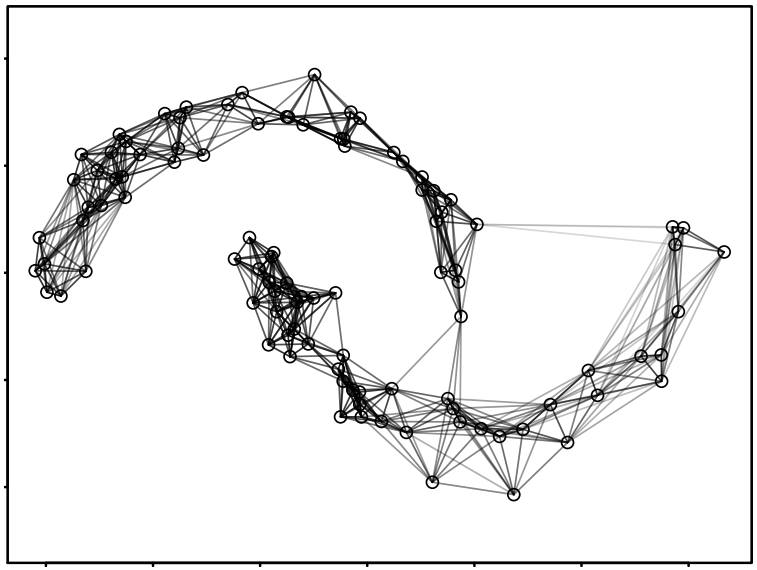
Outline

Algorithm

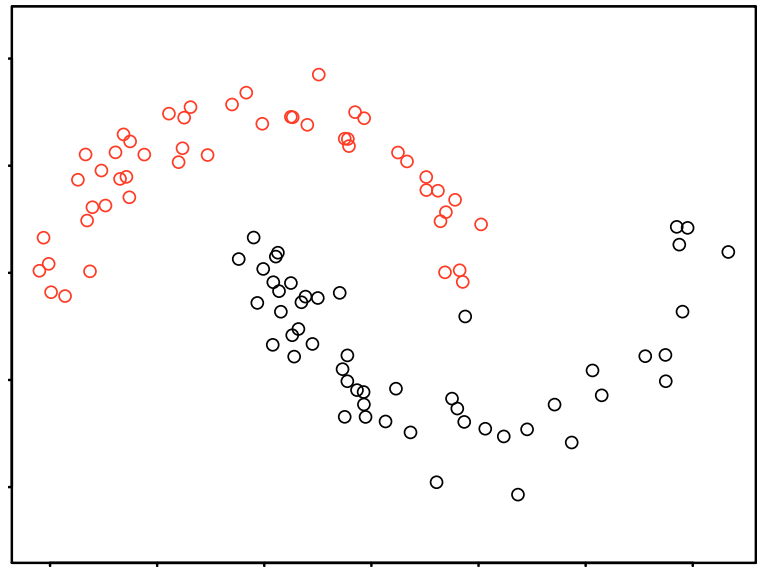
- Algorithm to construct k clusters
  1. Construct similarity graph $\mathbf{W}$
  2. Compute its (unnormalized) graph Laplacian $\mathbf{L}$
  3. Compute the last k eigenvectors $\mathbf{u}_n, \ldots, \mathbf{u}_{n-k+1}$ of $\mathbf{L}$ (i.e., having k smallest eigenvalues)
  4. Construct $n \times k$ matrix $\mathbf{U} = \begin{pmatrix} \mathbf{u}_n & \mathbf{u}_{n-1} & \cdots & \mathbf{u}_{n-k+1} \end{pmatrix}$
  5. Cluster the rows of $\mathbf{U}$ using k-means
- Simple, easy to implement
- Main trick: represent (or "embed") each data point into $\mathbb{R}^k$ ($=$ rows of $\mathbf{U}$)
- Change of representation enhances cluster-properties in the data

Why does this work? Why are we interested in the smallest eigenvalues?

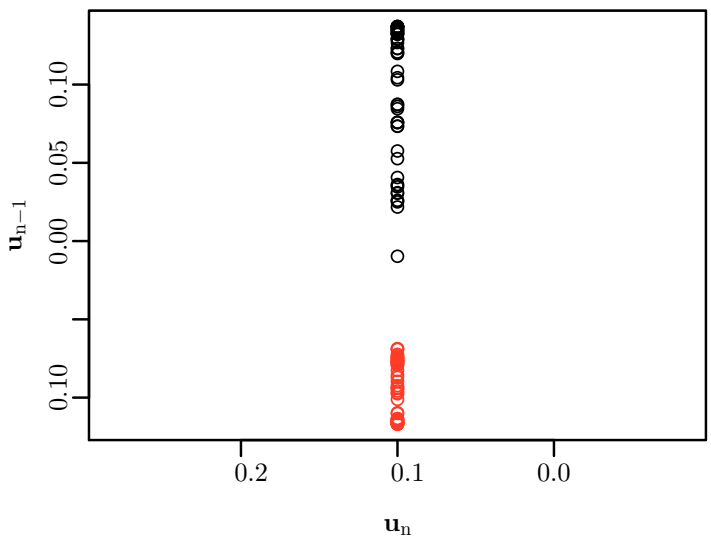
Unnormalized spectral clustering (example)

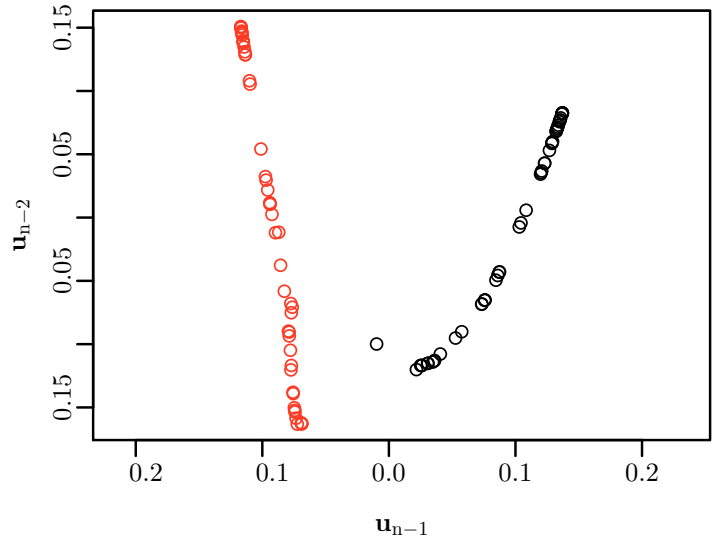


Similarity graph

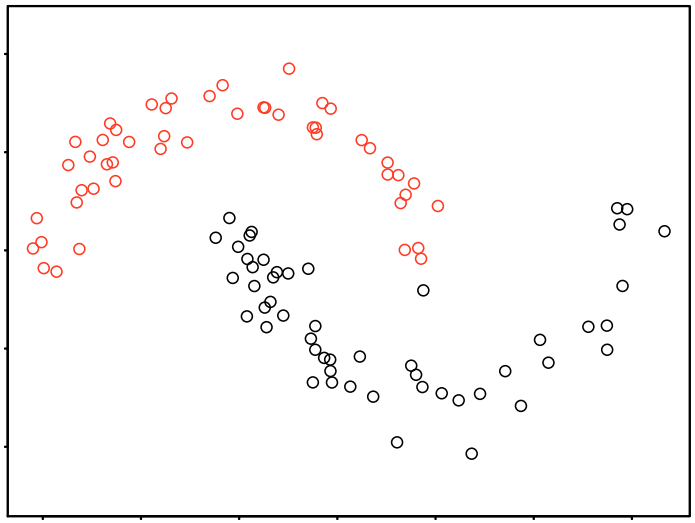


Spectral clustering ($k = 2$)

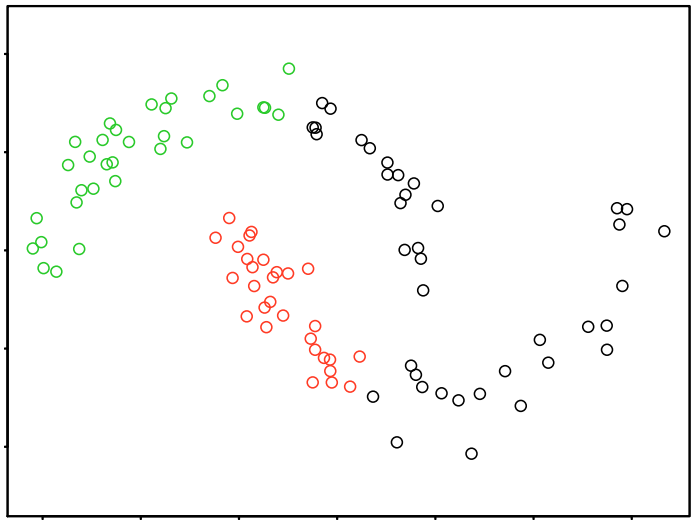




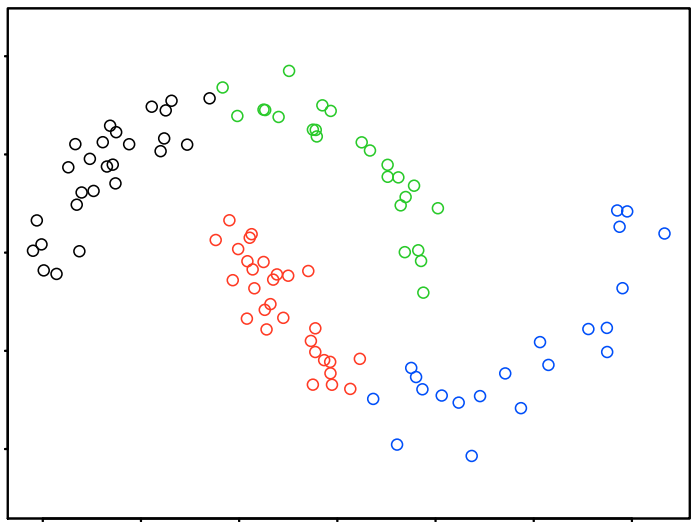
Unnormalized spectral clustering (example)



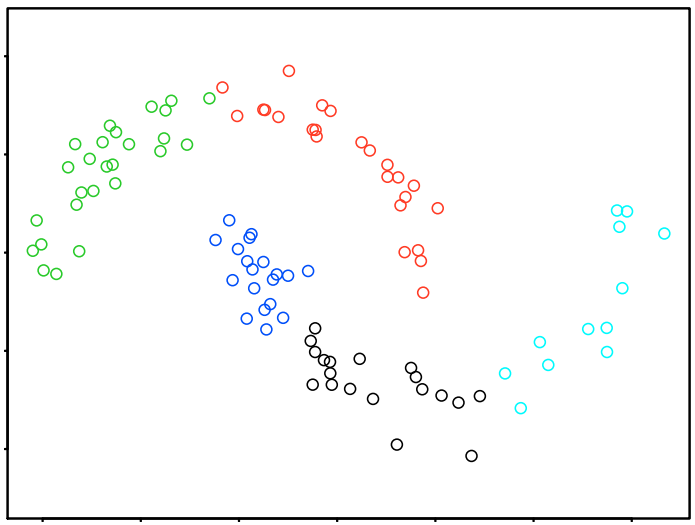
Spectral clustering ($k = 2$)



Spectral clustering ($k = 3$)



Spectral clustering ($k = 4$)



Spectral clustering ($k = 5$)

Why does spectral clustering work? (1)

- Consider the minimum ratio cut problem ($k = 2$)

$$\min_{A \subset V} \text{RatioCut}(A, \bar{A}) = \min_{A \subset V} \sum_{i \in A, j \in \bar{A}} w_{ij} \left(\frac{1}{|A|} + \frac{1}{|\bar{A}|} \right)$$

- Given A, set $\mathbf{x} \in \mathbb{R}^n$ such that

$$x_i = \begin{cases} \sqrt{|\bar{A}|/|A|} & \text{if } v_i \in A \\ -\sqrt{|A|/|\bar{A}|} & \text{if } v_i \in \bar{A} \end{cases}$$

- Easy to show
  1. $\mathbf{x}^T \mathbf{L} \mathbf{x} = n \cdot \text{RatioCut}(A, \bar{A})$
  2. $\sum_{i=1}^{n} x_i = 0$ so that $\mathbf{x} \perp \mathbf{1}$
  3. $\|\mathbf{x}\|^2 = n$

Why does spectral clustering work? (2)

- Minimum ratio cut can be rewritten as

$$\begin{aligned} \text{minimize} \quad & \mathbf{x}^T \mathbf{L} \mathbf{x} \\ \text{subject to} \quad & \mathbf{x} \perp \mathbf{1} \\ & \|\mathbf{x}\| = \sqrt{n} \\ & \mathbf{x} \text{ takes form defined in previous slide} \end{aligned}$$

- Still NP-hard; relax by dropping discreteness constraint

$$\begin{aligned} \text{minimize} \quad & \mathbf{x}^T \mathbf{L} \mathbf{x} \\ \text{subject to} \quad & \mathbf{x} \perp \mathbf{1} \\ & \|\mathbf{x}\| = \sqrt{n} \end{aligned}$$

- By Rayleigh-Ritz theorem: solution is eigenvector corresponding to second-smallest eigenvalue (appropriately normalized)
- $\mathbf{u}_{n-1}^T \mathbf{L} \mathbf{u}_{n-1} = \mathbf{u}_{n-1}^T \lambda_{n-1} \mathbf{u}_{n-1} = n\lambda_{n-1}$
- Note: $\lambda_{n-1} \leq \min_{A \subset V} \text{RatioCut}(A, \bar{A})$
- Similar argments for $k > 2$ (solutions of relaxation = last k eigenvectors)

Why does spectral clustering work? (3)

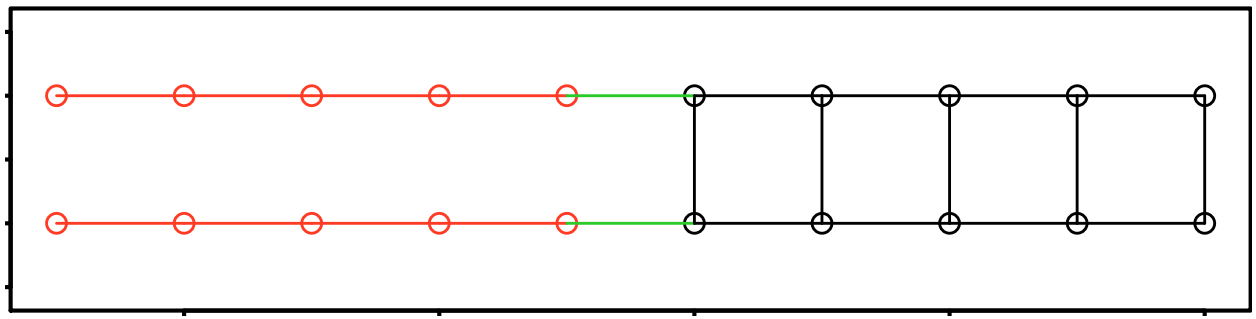
- Need to obtain clustering from $\mathbf{u}_{n-1}$
- Recall

$$x_i = \begin{cases} \sqrt{|\bar{A}|/|A|} & \text{if } v_i \in A \\ -\sqrt{|A|/|\bar{A}|} & \text{if } v_i \in \bar{A} \end{cases}$$

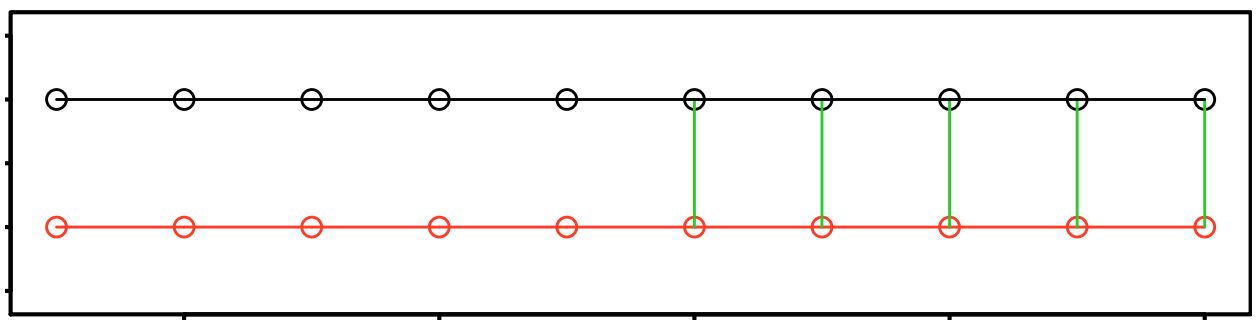
- Simple heuristic: use sign as cluster indicator
- k-means often produces better results
- Spectral clustering has no theoretical guarantees whatsoever
  But: popular because simple, standard linear algebra problem
- Approximation of balanced graph cuts (up to constant factor) still hard

Cockroach graph

- Example where spectral clustering performs particularly bad
- Minimum ratio cut cut: $8/n$
- Spectral clustering ratio cut: 1
- Spectral clustering is $O(n)$ times worse



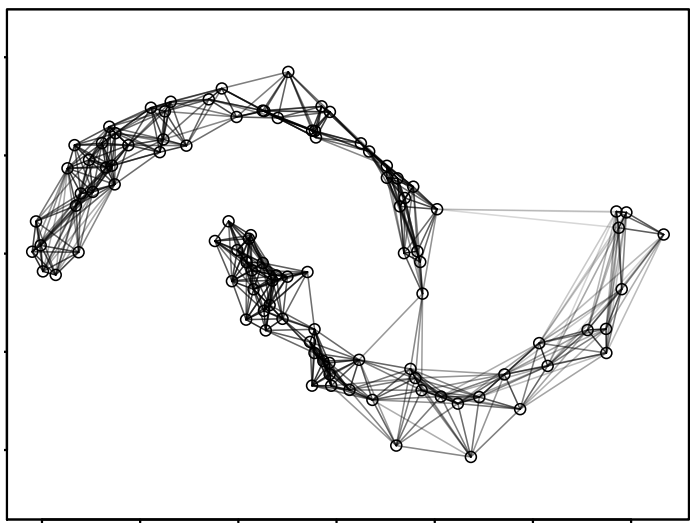
Minimum ratio cut

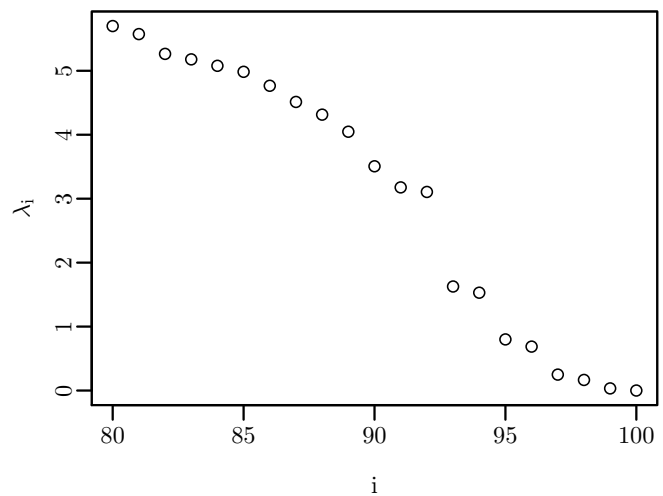


Ratio cut with spectral clustering and sign heuristic

Discussion

- Computation of eigenvectors
  - Graph can be very large
  - But Laplacian is sparse
  - Many efficient algorithms exist finding the eigendecomposition of such matrices
- Number k of clusters
  - Difficult problem
  - Standard approaches can be used
  - Eigengap heuristic: choose k such that eigenvalue $\lambda_1, \ldots, \lambda_{n-k}$ large, eigenvalues $\lambda_{n-k+1}, \ldots, \lambda_n$ small





Outline

Normalized graph Laplacians

Definition

There are two common normalizations of the graph Laplacian:

$$\mathbf{L}_{\text{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$$

$$\mathbf{L}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$$

- Normalization is performed w.r.t. degree
- $\mathbf{L}_{\text{sym}}$ is symmetric, $\mathbf{L}_{\text{rw}}$ is not

$$\underset{\mathbf{L}}{\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}} \quad \underset{\mathbf{L}_{\text{sym}}}{\begin{pmatrix} 1 & -1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1 & -1/\sqrt{2} \\ 0 & -1/\sqrt{2} & 1 \end{pmatrix}} \quad \underset{\mathbf{L}_{\text{rw}}}{\begin{pmatrix} 1 & -1 & 0 \\ -0.5 & 1 & -0.5 \\ 0 & -1 & 1 \end{pmatrix}}$$

Normalized spectral clustering

- Normalized graph Laplacians have similar spectral properties
- Normalized spectral clustering (using $\mathbf{L}_{\text{rw}}$)
  1. Construct similarity graph $\mathbf{W}$
  2. Compute its normalized graph Laplacian $\mathbf{L}_{\text{rw}}$
  3. Compute the last k eigenvectors $\mathbf{u}_n, \ldots, \mathbf{u}_{n-k+1}$ of $\mathbf{L}_{\text{rw}}$ (i.e., having k smallest eigenvalues)
  4. Construct $n \times k$ matrix $\mathbf{U} = \begin{pmatrix} \mathbf{u}_n & \mathbf{u}_{n-1} & \cdots & \mathbf{u}_{n-k+1} \end{pmatrix}$
  5. Cluster the rows of $\mathbf{U}$ using k-means
- Normalized spectral clustering is a relaxation of Ncut
- Better behaved from statistical point of view

The normalized spectral clustering algorithm above is often method of choice.

Outline

Lessons learned

- Graphs can be represented by matrices (and vice versa)
  - Adjacency matrix
  - Degree matrix
  - Walk matrix
  - Graph Laplacian
- Spectral properties of these matrices relate to properties of the graph
- Spectral clustering
  - Find non-convex clusters using neighborhood graphs
  - Good clustering $\approx$ good graph cut (RatioCut or Ncut)
  - Related to smallest eigenvectors of graph Laplacian

Literature

- David Skillicorn
  Understanding Complex Datasets: Data Mining with Matrix Decompositions (Chapter 4)
  Chapman and Hall, 2007

- Ulrike von Luxburg
  A Tutorial on Spectral Clustering
  Statistics and Computing, 17(4), 2007
  `http://www.kyb.mpg.de/publications/attachments/Luxburg07_tutorial_4488%5B0%5D.pdf`