San José State University

Math 263: Stochastic Processes

Spectral Clustering

Dr. Guangliang Chen

Outline of the presentation

- Introduction
- Spectral graph theory
- Spectral clustering algorithms
- Diffusion distance and commute time

References

Tutorial: von Luxburg, U. A tutorial on spectral clustering. Stat Comput 17, 395–416 (2007). https://arxiv.org/pdf/0711.0189.pdf

Original papers:

- Shi and Malik (2000), "Normalized cuts and image segmentation", in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 888–905.
- Ng, Jordan, and Weiss (2001). "On spectral clustering: analysis and an algorithm". Advances in Neural Information Processing Systems, Pages 849–856.
- Coifman and Lafon (2006), "Diffusion maps", Applied and Computational Harmonic Analysis, Volume 21, Issue 1, Pages 5–30.

Data clustering

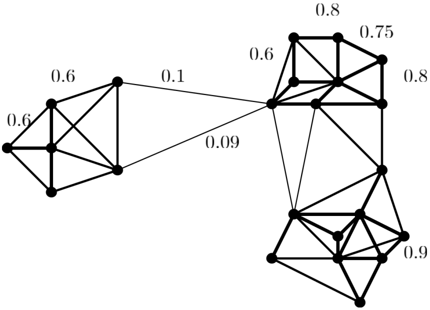
Clustering is an unsupervised learning task in machine learning.

Problem 0.1. Given a set of objects and a similarity measure, partition the data set into k disjoint subsets (i.e., clusters) such that

- objects in the same cluster are similar to each other;
- objects in different clusters are generally not similar.

We often represent such information via an undirected, weighted graph, called **similarity graph**:

- Nodes represent the objects to be clustered;
- Edges connect similar objects (and the weights on them indicate the level of similarity).



Accordingly, clustering is converted to a graph partitioning problem.

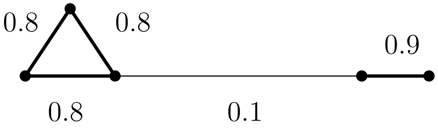
Def 0.1. Mathematically, an undirected, weighted graph $\mathscr{G} = (V, E, \mathbf{W})$ is a structure that has the following components:

- vertex set $V = \{v_1, \ldots, v_n\}$
- edge set $E = \{e_{ij}\}$
- weight matrix $\mathbf{W} = (w_{ij})$

An edge exists between two vertices i, j if and only if $w_{ij} > 0$.

Remark. A similarity graph is uniquely defined by a given weight matrix.

$$\mathbf{W} = \begin{pmatrix} & 0.8 & 0.8 & & \\ 0.8 & & 0.8 & & \\ 0.8 & 0.8 & & 0.1 & \\ & & 0.1 & & 0.9 \\ & & & 0.9 & \end{pmatrix}$$

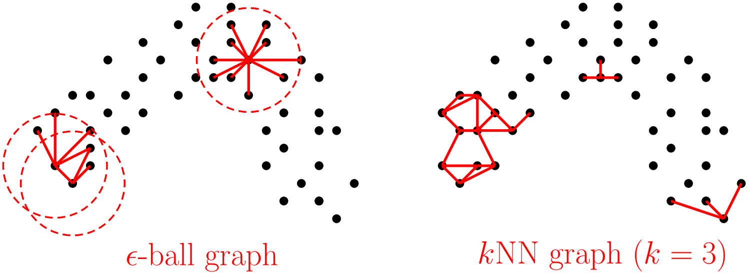


How to construct similarity graphs on vector data

Given a data set $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, we can construct a similarity graph on it in one of the following ways:

- ϵ-neighborhood graph:

$$w_{ij} = \begin{cases} 1, & \text{if } \|\mathbf{x}_i - \mathbf{x}_j\| < \epsilon \\ 0, & \text{otherwise} \end{cases}$$



- kNN graph:

$$w_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_i \in k\text{NN}(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in k\text{NN}(\mathbf{x}_i) \\ 0, & \text{otherwise} \end{cases}$$

where $k\text{NN}(\mathbf{x})$ represents the k nearest neighbors set of $\mathbf{x}$ in V.

- mutual kNN graph:

$$w_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_i \in k\text{NN}(\mathbf{x}_j) \text{ and } \mathbf{x}_j \in k\text{NN}(\mathbf{x}_i) \\ 0, & \text{otherwise} \end{cases}$$

- Gaussian similarity graph (fully connected):

$$w_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$$

where $\sigma > 0$ is a parameter to be set by the user.

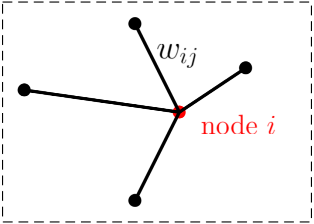
Given an undirected, weighted graph $\mathcal{G} = (V, E, \mathbf{W})$, define

- the **degree** of a single vertex $v_i \in V$:

$$d_i = \sum_{j \in V} w_{ij}$$

- and also the **degree matrix**:

$$\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n \times n}$$
$$= \mathrm{diag}(\mathbf{W1}).$$



Note that d_i measures the connectivity of node i in the graph: The larger the degree, the more strongly connected the node.

For example, the degree matrix associated with the previous graph is

$$\mathbf{W} = \begin{pmatrix} & 0.8 & 0.8 & & \\ 0.8 & & 0.8 & & \\ 0.8 & 0.8 & & 0.1 & \\ & & 0.1 & & 0.9 \\ & & & 0.9 & \end{pmatrix} \longrightarrow \quad \mathbf{D} = \begin{pmatrix} 1.6 & & & & \\ & 1.6 & & & \\ & & 1.7 & & \\ & & & 1.0 & \\ & & & & 0.9 \end{pmatrix}$$

For any subset $A \subset V$, define

$$1_A = (f_1, \ldots, f_n), \quad f_i = \begin{cases} 1, & i \in A; \\ 0, & i \notin A \end{cases}$$

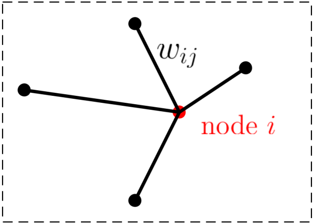
$$|A| = \#\text{ vertices in } A$$

$$\mathrm{Vol}(A) = \sum_{i \in A} d_i$$

The first quantity is an indicator variable for the subgraph A, and the last two are two different measures of the sizes of A.

We have already shown that a Markov chain can be induced by any undirected, weighted graph $\mathscr{G} = (V, E, \mathbf{W})$ by letting $S = V$ (state space) and $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ (transition matrix), i.e.,

$$p_{ij} = \frac{w_{ij}}{d_i}, \quad \text{for all (connected) nodes } j \in V.$$

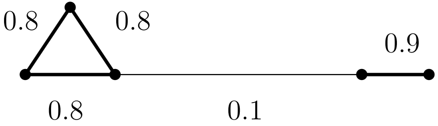


Let $\mathscr{G}$ be an undirected, weighted graph with weight matrix $\mathbf{W}$ and degree matrix $\mathbf{D} = \mathrm{diag}(\mathbf{W} \cdot \mathbf{1})$.

Def 0.2. The unnormalized graph Laplacian is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}, \qquad \ell_{ij} = \begin{cases} -\sum_{k \neq i} w_{ik}, & i = j; \\ -w_{ij}, & i \neq j \end{cases}$$

Example 0.2. Determine the graph Laplacian of the following graph:



Answer:

$$\mathbf{L} = \begin{pmatrix} 1.6 & -0.8 & -0.8 & & \\ -0.8 & 1.6 & -0.8 & & \\ -0.8 & -0.8 & 1.7 & -0.1 & \\ & & -0.1 & 1 & -0.9 \\ & & & -0.9 & 0.9 \end{pmatrix}$$

The graph Laplacian has many interesting properties.

Theorem 0.1. Let $\mathbf{L} \in \mathbb{R}^{n\times n}$ represent a graph Laplacian. Then

(1) $\mathbf{L}$ is symmetric (thus all the eigenvalues are real).

(2) All the rows (and columns) sum to 0, i.e., $\mathbf{L1} = \mathbf{0}$. This implies that $\mathbf{L}$ has a eigenvalue 0 with eigenvector $\mathbf{1}$.

(3) For every vector $\mathbf{f} \in \mathbb{R}^d$ we have

$$\mathbf{f}^T\mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

This implies that $\mathbf{L}$ is positive semidefinite and accordingly, its eigenvalues are all nonnegative: $0 = \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$.

(4) The algebraic multiplicity of the eigenvalue 0 equals the number of connected components in the graph.

Proof. Properties (1) and (2) are obvious, so we only prove the last two.

(3) By direct calculation,

$$\begin{aligned}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 &= \sum_{i,j} w_{ij} f_i^2 + \sum_{i,j} w_{ij} f_j^2 - 2\sum_{i,j} w_{ij} f_i f_j \\ &= \sum_{i} d_i f_i^2 + \sum_{j} d_j f_j^2 - 2\sum_{i,j} w_{ij} f_i f_j \\ &= 2\mathbf{f}^T \mathbf{D}\mathbf{f} - 2\mathbf{f}^T \mathbf{W}\mathbf{f} = 2\mathbf{f}^T \mathbf{L}\mathbf{f}.\end{aligned}$$

(4) Let $\mathbf{v}$ be any eigenvector of $\mathbf{L}$ corresponding to eigenvalue 0, i.e., $\mathbf{Lv} = 0 \cdot \mathbf{v} = \mathbf{0}$. Then

$$0 = \mathbf{v}^T \mathbf{L} \mathbf{v} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (v_i - v_j)^2$$

It follows that

$$w_{ij}(v_i - v_j)^2 = 0, \quad \forall\, i, j$$

From this we obtain that $v_i = v_j$ whenever $w_{ij} > 0$ (if there is an edge between i, j).

Therefore, $\mathbf{v}$ is piecewise constant on the connected components $A_1, \ldots, A_k$, i.e.,

$$\mathbf{v} = \sum_{i=1}^{k} c_i \mathbf{1}_{A_i}.$$

In particular, $\mathbf{1}_{A_1}, \ldots, \mathbf{1}_{A_k}$ are (linearly independent) eigenvectors.

The geometric (and also algebraic) multiplicity of eigenvalue 0 is thus equal to the number of connected components.

Example 0.3. The previous graph is connected. The graph Laplacian has eigenvalues

$$\lambda_1 = 0,\ \lambda_2 = 0.0788,\ \lambda_3 = 1.8465,\ \lambda_4 = 2.4000,\ \lambda_5 = 2.4747.$$

Example 0.4. Consider the following modified graph with two connected components:

$$\mathbf{W} = \begin{pmatrix} 0 & .8 & .8 & 0 & 0 \\ .8 & 0 & .8 & 0 & 0 \\ .8 & .8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .9 \\ 0 & 0 & 0 & .9 & 0 \end{pmatrix}$$

It can be shown that

$$\det(\lambda \mathbf{I} - \mathbf{L}) = \lambda(\lambda - 2.4)^2 \cdot \lambda(\lambda - 1.8).$$

Thus, the unnormalized graph Laplacian has a repeated eigenvalue 0, with multiplicity 2 (which is the number of connected components).

We next define two normalized graph Laplacians.

Def 0.3.

$$\widetilde{\mathbf{L}}_{\mathrm{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W} = \mathbf{I} - \mathbf{P};$$
$$\widetilde{\mathbf{L}}_{\mathrm{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}.$$

Remark.

- $\widetilde{\mathbf{L}}_{\mathrm{rw}}\mathbf{1} = (\mathbf{D}^{-1}\mathbf{L})\mathbf{1} = \mathbf{D}^{-1}(\mathbf{L}\mathbf{1}) = \mathbf{D}^{-1}\mathbf{0} = \mathbf{0}$. This shows that $\widetilde{\mathbf{L}}_{\mathrm{rw}}$ has an identical row sum of zero. Moreover, $\widetilde{\mathbf{L}}_{\mathrm{rw}}$ has an eigenvalue of 0 with corresponding eigenvector $\mathbf{1}$.

- $\widetilde{\mathbf{L}}_{\text{sym}}$ is symmetric while $\widetilde{\mathbf{L}}_{\text{rw}}$ is not, but they are similar matrices:

$$\widetilde{\mathbf{L}}_{\text{rw}} = \mathbf{D}^{-1/2}\widetilde{\mathbf{L}}_{\text{sym}}\mathbf{D}^{1/2}.$$

Thus, they have the same eigenvalues (but different eigenvectors).

- $\widetilde{\mathbf{L}}_{\text{sym}}$ is also positive semidefinite (but $\widetilde{\mathbf{L}}_{\text{rw}}$ is not):

$$\mathbf{f}^T\widetilde{\mathbf{L}}_{\text{sym}}\mathbf{f} = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2,$$

with the multiplicity of the zero eigenvalue equal to the number of connected components in the graph.

- λ is an eigenvalue of $\widetilde{\mathbf{L}}_{\text{rw}}$ with associated eigenvector $\mathbf{v}$ if and only if $1-\lambda$ is an eigenvalue of $\mathbf{P}$ with the same eigenvector $\mathbf{v}$:

$$\widetilde{\mathbf{L}}_{\text{rw}}\mathbf{v} = \lambda\mathbf{v} \qquad \text{if and only if} \quad \mathbf{P}\mathbf{v} = (1-\lambda)\mathbf{v}.$$

  This shows that the largest eigenvalue of $\mathbf{P}$ is 1 (with its multiplicity equal to the number of connected components of the undirected graph).

Example 0.5. For the connected graph in the preceding examples, the two normalized graph Laplacians, $\widetilde{\mathbf{L}}_{\mathrm{rw}}$, $\widetilde{\mathbf{L}}_{\mathrm{sym}}$, have eigenvalues

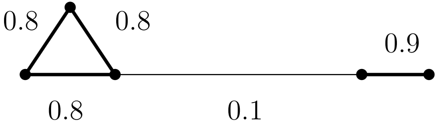
$$\lambda_1 = 0,\ \lambda_2 = 0.0693,\ \lambda_3 = 1.4773,\ \lambda_4 = 1.5000,\ \lambda_5 = 1.9534.$$

For any two subsets $A, B \subset V$, define

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

If $B = \bar{A}$, then it is called a cut

$$\mathrm{Cut}(A,\bar{A}) = W(A,\bar{A}) = \sum_{i\in A,\, j\notin A} w_{ij}$$



Another special case of $W(A,B)$ is when $B = V$:

$$W(A,V) = \sum_{i\in A,\, j\in V} w_{ij} = \sum_{i\in A} d_i = \mathrm{Vol}(A)$$

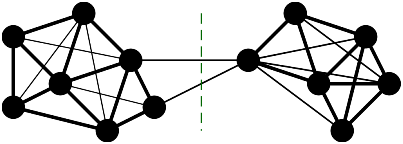
A collection of subsets $A_1,\ldots,A_k \subset V$ is called a partition of V if

$$A_1 \cup \cdots \cup A_k = V, \quad \text{and } A_i \cap A_j = \emptyset,\ \forall\, i \neq j$$

For a partition of size $k \geq 3$, the cut is defined as

$$\mathrm{Cut}(A_1,\ldots,A_k) = \frac{1}{2}\sum_{i=1}^{k} W(A_i, \bar{A}_i).$$

The Normalized Cut (NCut) algorithm



Given a similarity graph $\mathscr{G} = \{V, E, \mathbf{W}\}$ to be partitioned into two parts, Shi and Malik (2000) proposed to perform 2-way spectral clustering by solving

$$\min_{\substack{A \cup B = V \\ A \cap B = \emptyset}} \mathrm{NCut}(A, B) \stackrel{\text{def}}{=} \mathrm{Cut}(A, B) \left(\frac{1}{\mathrm{Vol}(A)} + \frac{1}{\mathrm{Vol}(B)} \right).$$

Remark. To minimize the NCut function, we need to

- minimize the cut,
- maximize the volume of each subgraph

Thus, we are seeking a balanced cut with minimal loss of edge weights.

Remark. If $|A|, |B|$ are used to measure the sizes of the clusters instead, then it is called **ratio cut**:

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

We show that the normalized cut criterion can be expressed as a Rayleigh quotient in terms of the graph Laplacian.

Theorem 0.2. For any similarity graph $\mathscr{G} = \{V, E, \mathbf{W}\}$ and partition $A \cup B = V$, we have

$$\mathrm{NCut}(A,B) = \frac{\mathbf{x}^T \mathbf{L} \mathbf{x}}{\mathbf{x}^T \mathbf{D} \mathbf{x}},$$

where

$$\mathbf{x} = \frac{1}{\mathrm{Vol}(A)} \mathbf{1}_A - \frac{1}{\mathrm{Vol}(B)} \mathbf{1}_B, \quad x_i = \begin{cases} \frac{1}{\mathrm{Vol}(A)}, & i \in A \\ \frac{-1}{\mathrm{Vol}(B)}, & i \in B \end{cases}$$

Proof. By direct calculation:

$$\begin{aligned}
\mathbf{x}^T\mathbf{L}\mathbf{x} &= \frac{1}{2}\sum_{i,j} w_{ij}(x_i - x_j)^2 \\
&= \sum_{i\in A, j\in B} w_{ij}\left(\frac{1}{\mathrm{Vol}(A)} + \frac{1}{\mathrm{Vol}(B)}\right)^2 \\
&= \mathrm{Cut}(A,B)\left(\frac{1}{\mathrm{Vol}(A)} + \frac{1}{\mathrm{Vol}(B)}\right)^2 \\
\mathbf{x}^T\mathbf{D}\mathbf{x} &= \sum_i d_i x_i^2 = \sum_{i\in A} d_i \cdot \frac{1}{\mathrm{Vol}(A)^2} + \sum_{i\in B} d_i \cdot \frac{1}{\mathrm{Vol}(B)^2} \\
&= \frac{1}{\mathrm{Vol}(A)} + \frac{1}{\mathrm{Vol}(B)}.
\end{aligned}$$

Remark. The vector $\mathbf{x}$ is completely defined by the partition, containing only two distinct values and satisfying a hidden constraint:

$$\mathbf{x}^T \mathbf{D}\mathbf{1} = 0.$$

To see the last one, write

$$\mathbf{x}^T \mathbf{D}\mathbf{1} = \sum_i x_i d_i = \frac{1}{\mathrm{Vol}(A)} \sum_{i \in A} d_i - \frac{1}{\mathrm{Vol}(B)} \sum_{i \in B} d_i = 1 - 1 = 0.$$

The vector $\mathbf{x}$ also uniquely defines the partition. Thus, finding the optimal partition is equivalent to finding the minimizer $\mathbf{x}$.

We have arrived at the following equivalent problem:

$$\min_{\substack{\mathbf{x}\in\{a,-b\}^n:\\ \mathbf{x}^T\mathbf{D}\mathbf{1}=0}} \frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\mathbf{x}^T\mathbf{D}\mathbf{x}}.$$

This problem is NP-hard, so we solve a relaxed problem instead:

$$\min_{\substack{\mathbf{x}\neq\mathbf{0}\in\mathbb{R}^n\\ \mathbf{x}^T\mathbf{D}\mathbf{1}=0}} \frac{\mathbf{x}^T\mathbf{L}\mathbf{x}}{\mathbf{x}^T\mathbf{D}\mathbf{x}}.$$

Theorem 0.3. A minimizer of the above relaxed problem is given by the second smallest eigenvector of $\widetilde{\mathbf{L}}_{\text{rw}}$: $\widetilde{\mathbf{L}}_{\text{rw}}\mathbf{x} = \lambda_2\mathbf{x}$.

(In terms of $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$, the minimizer $\mathbf{x}$ is the second largest eigenvector)

Proof. Define $\mathbf{y} = \mathbf{D}^{1/2}\mathbf{x}$. Then the above problem can be rewritten as

$$\min_{\mathbf{y}\neq\mathbf{0},\, \mathbf{y}^T\mathbf{D}^{1/2}\mathbf{1}=0} \frac{\mathbf{y}^T\widetilde{\mathbf{L}}_{\text{sym}}\mathbf{y}}{\mathbf{y}^T\mathbf{y}}. \qquad \longleftarrow \text{Rayleigh quotient}$$

Note that $\mathbf{D}^{1/2}\mathbf{1}$ is an eigenvector of $\widetilde{\mathbf{L}}_{\text{sym}}$ corresponding to eigenvalue 0:

$$\widetilde{\mathbf{L}}_{\text{sym}} \cdot \mathbf{D}^{1/2}\mathbf{1} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{1} = \mathbf{0} = 0 \cdot \mathbf{D}^{1/2}\mathbf{1}$$

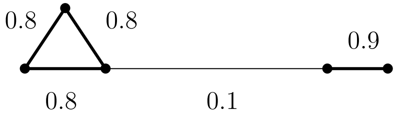
Thus, the minimizer $\mathbf{y}$ is given by the second smallest eigenvector of $\widetilde{\mathbf{L}}_{\text{sym}}$:

$$\widetilde{\mathbf{L}}_{\text{sym}}\mathbf{y} = \lambda_2\mathbf{y}.$$

In terms of $\mathbf{x}$, this equation becomes

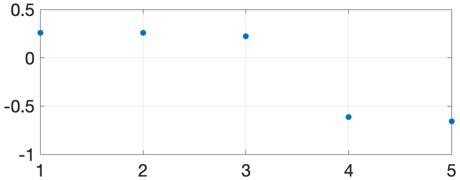
$$\widetilde{\mathbf{L}}_{\text{sym}}\mathbf{D}^{1/2}\mathbf{x} = \lambda_2\mathbf{D}^{1/2}\mathbf{x}, \quad \text{or equivalently, } \widetilde{\mathbf{L}}_{\text{rw}}\mathbf{x} = \lambda_2\mathbf{x}.$$

Example 0.6. Consider the graph again:



The second largest eigenvector of $\mathbf{P}$ (also the second smallest eigenvector of $\widetilde{\mathbf{L}}_{\mathrm{rw}}$) is

$$\mathbf{v}_2 = [.2594, .2594, .2235, -.6152, -.6610]^T.$$



Algorithm 1 2-way NCut (Shi and Malik, 2000)

Input: Data $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$, scale parameter σ

Output: A bipartition of $X = C_1 \cup C_2$

Steps:

1: Construct a weighted graph by assigning weights

$$w_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$$

2: Find the second largest eigenvector $\mathbf{v}_2$ of $\mathbf{P} = \mathbf{D}^{-1/2}\mathbf{W}$.

3: Assign labels based on the sign of the coordinates of $\mathbf{v}_2$

Remark. When there are $k > 2$ clusters in the data, one can apply 2-way NCut repeatedly until a total of k clusters have been found.

Alternatively, one can extend the 2-way NCut algorithm to deal with $k > 2$ clusters as follows:

- Step 2 → find the largest eigenvectors $\mathbf{v}_2, \ldots, \mathbf{v}_k$ of $\mathbf{P}$ to form an embedding matrix $\mathbf{Y} = [\mathbf{v}_2, \ldots, \mathbf{v}_k] \in \mathbb{R}^{n \times (k-1)}$, and
- Step 3 → apply the $k$means algorithm to group the rows of $\mathbf{Y}$ (treated as new coordinates of the original data) into k clusters.

Demonstrations

Comments on spectral clustering

Spectral clustering is simple, powerful and highly accurate, achieving state-of-the-art results in many applications:

- Image segmentation
- Image clustering
- Document clustering
- Community detection in social networks

However, a significant drawback is its $O(n^2d)$ complexity when having large data sets in high dimensions.

There has been a considerable amount of research to develop fast spectral clustering algorithms with $O(nd)$ complexity. A few examples are

- K. Pham and G. Chen. Large-scale Spectral Clustering using Diffusion Coordinates on Landmark-based Bipartite Graphs. The 12th Workshop on Graph-based Natural Language Processing (TextGraphs-12), New Orleans, Louisiana, June 2018
- G. Chen. "Scalable Spectral Clustering with Cosine Similarity". The 24th International Conference on Pattern Recognition (ICPR), Beijing, China, August 2018
- G. Chen. "A General Framework for Scalable Spectral Clustering Based on Document Models". Pattern Recognition Letters, 125: 488-493, July 2019

A matrix perturbation perspective

Ng, Jordan and Weiss (2001) proposed a different version of spectral clustering by using the top k eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ of $\widetilde{\mathbf{W}}$ (instead of $\mathbf{P}$)

$$\widetilde{\mathbf{L}}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W} = \mathbf{I} - \mathbf{P};$$
$$\widetilde{\mathbf{L}}_{\text{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} = \mathbf{I} - \widetilde{\mathbf{W}}.$$

and then applying the $k$means algorithm to the rows of $\mathbf{Y} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k] \in \mathbb{R}^{n \times k}$ to find k clusters.

They then justified the algorithm by viewing $\widetilde{\mathbf{W}}$ as a noisy version of a clean, block-diagonal $\mathbf{W}$ (with each block corresponding to a distinct cluster).

A random walk perspective

Consider the Markov chain defined on the similarity graph $\mathscr{G} = \{V, E, \mathbf{W}\}$, with transition matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$.

The chain is finite, and if the graph is connected, then the Markov chain is irreducible and thus also positive recurrent. Accordingly, it possesses a unique stationary distribution.

$$\boldsymbol{\pi} = (\pi_i), \qquad \text{where} \quad \pi_i = d_i / \mathrm{Vol}(V).$$

If the graph is also non-bipartite, then the chain always converges to the above stationary distribution.

Theorem 0.4. Let $\mathscr{G} = \{V, E, \mathbf{W}\}$ be connected but non-bipartite. Assume that we run the random walk $\{X_t, t = 0, 1, 2, \ldots\}$ starting with X_0 in the stationary distribution π. Then

$$\mathrm{NCut}(A, \bar{A}) = P(X_1 \in \bar{A} \mid X_0 \in A) + P(X_1 \in A \mid X_0 \in \bar{A}).$$

Proof. First, for any subset $A \subset V$,

$$\begin{aligned}
P(X_0 \in A, X_1 \in \bar{A}) &= \sum_{i\in A, j\in \bar{A}} P(X_0 = i, X_1 = j) \\
&= \sum_{i\in A, j\in \bar{A}} P(X_1 = j \mid X_0 = i) P(X_0 = i) \\
&= \sum_{i\in A, j\in \bar{A}} p_{ij}\pi_i = \sum_{i\in A, j\in \bar{A}} \frac{w_{ij}}{d_i} \frac{d_i}{\mathrm{Vol}(V)} \\
&= \frac{1}{\mathrm{Vol}(V)} \mathrm{Cut}(A, \bar{A}).
\end{aligned}$$

It follows that

$$P(X_1 \in \bar{A} \mid X_0 \in A) = \frac{P(X_1 \in \bar{A}, X_0 \in A)}{P(X_0 \in A)} = \frac{\mathrm{Cut}(A, \bar{A}) / \mathrm{Vol}(V)}{\mathrm{Vol}(A) / \mathrm{Vol}(V)} = \frac{\mathrm{Cut}(A, \bar{A})}{\mathrm{Vol}(A)}.$$

Similarly, we can show that

$$P(X_1 \in A \mid X_0 \in \bar{A}) = \frac{\mathrm{Cut}(A, \bar{A})}{\mathrm{Vol}(\bar{A})}.$$

Combining the two equations together would complete the proof. $\square$

Let $G = (V, E, \mathbf{W})$ be a connected, undirected graph. The induced Markov chain has state space $S = V$ and transition matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$.

Using the random walk perspective, one can define two kinds of distances between the vertices of the graph:

- **Diffusion distance**[1]: Define based on powers of the transition matrix, i.e., $\mathbf{P}^t$
- **Commute distance**[2]: Defined based on the pseudoinverse of the graph Laplacian, i.e., $\mathbf{L}^\dagger$

[1] `https://www.sciencedirect.com/science/article/pii/S1063520306000546`
[2] `https://arxiv.org/pdf/0711.0189.pdf`; see page 15

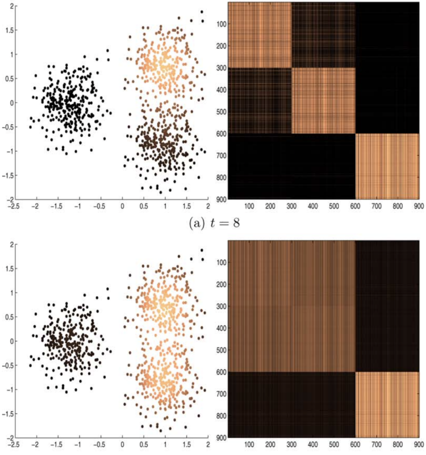
Let $1 = \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n$ be the eigenvalues of $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$, with associated eigenvectors $\mathbf{1} = \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$. The t-step **diffusion distance** between vertices i and j is

$$D_t(i,j) = \sqrt{\sum_{\ell=2}^{n} \lambda_\ell^{2t} (\mathbf{v}_\ell(i) - \mathbf{v}_\ell(j))^2}$$

This is equal to the Euclidean distance on the embedding space

$$i \mapsto [\lambda_2^t \mathbf{v}_2(i), \ldots, \lambda_n^t \mathbf{v}_n(i)]$$

Note that the columns can be truncated for reduced dimensionality.



(a) $t = 8$

(b) $t = 64$

The **commute distance** c_{ij} (also called resistance distance) between two vertices $i, j \in V$ of the graph is the expected time it takes the random walk to travel from one vertex to the other vertex and back:

$$c_{ij} = m_{ij} + m_{ji}, \quad m_{ij} = \mathrm{E}\left(\min_{n \ge 1} \{X_n = j\} \mid X_0 = i\right)$$

Unlike the shortest-path distance, the commute distance c_{ij} is small only when there are many different short ways to get from one vertex to another.

On the other hand, it can avoid short-circuiting and is thus robust to a small subset of edges.

Theorem 0.5. For any connected, undirected graph $G = (V, E, \mathbf{W})$, the commute time between any two vertices $i, j \in V$ is

$$c_{ij} = \mathrm{Vol}(V) \cdot \left(\ell^\dagger_{ii} - 2\ell^\dagger_{ij} + \ell^\dagger_{jj}\right)$$
$$= \mathrm{Vol}(V) \cdot (\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{L}^\dagger (\mathbf{e}_i - \mathbf{e}_j)$$

where

- $\mathbf{L}^\dagger = (\ell^\dagger_{ij})$: Moore-Penrose pseudoinverse[3] of the graph Laplacian $\mathbf{L}$;
- $\mathbf{e}_i$: the ith canonical basis vector for $\mathbb{R}^n$.

---

[3] https://www.sjsu.edu/faculty/guangliang.chen/Math250/lec6ginverse.pdf

Demonstration on the toy graph:

```
>> L_dag = pinv(L)
C = diag(L_dag) + diag(L_dag)' - 2 * L_dag;
C = C * sum(d)

L_dag =

    2.0778    1.6611    1.4944   -2.5056   -2.7278
    1.6611    2.0778    1.4944   -2.5056   -2.7278
    1.4944    1.4944    1.7444   -2.2556   -2.4778
   -2.5056   -2.5056   -2.2556    3.7444    3.5222
   -2.7278   -2.7278   -2.4778    3.5222    4.4111


C =

         0    5.6667    5.6667   73.6667   81.2222
    5.6667         0    5.6667   73.6667   81.2222
    5.6667    5.6667         0   68.0000   75.5556
   73.6667   73.6667   68.0000         0    7.5556
   81.2222   81.2222   75.5556    7.5556         0
```