San José State University

Math 250: Mathematical Data Visualization

Laplacian Eigenmaps (and spectral clustering)

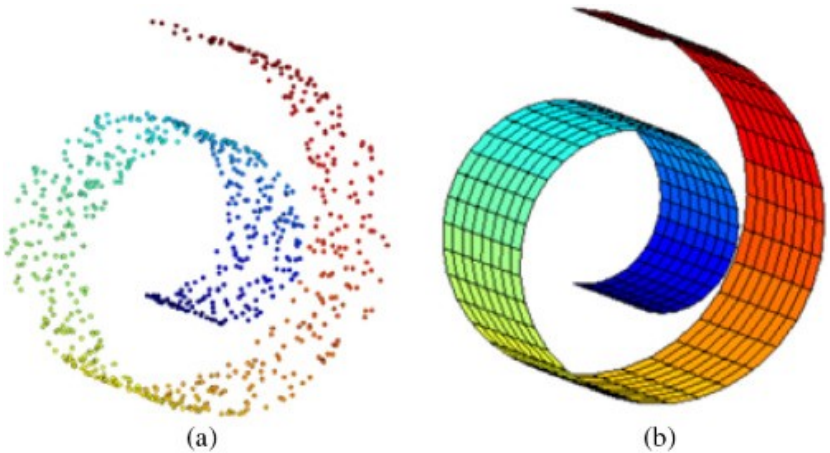
Dr. Guangliang Chen

Outline of the lecture:

- Background
  - Similarity graphs
  - Spectral graph theory
- Laplacian Eigemaps (LE)
  - For dimension reduction (covered in this lecture)
  - For clustering (LE + $k$means = spectral clustering)
- Spectral clustering and scalability

Introduction

Consider the **manifold learning** problem again: Given a set of points along a manifold embedded in a high dimensional Euclidean space, $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{M} \subset \mathbb{R}^d$, find another set of vectors in a low-dimensional Euclidean space, $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^k$ (for some $k \ll d$), such that $\mathbf{y}_i$ "represents" $\mathbf{x}_i$ by preserving certain kind of information.



Laplacian Eigenmaps (and spectral clustering)

We have already seen ISOmap as a nonlinear dimensionality reduction approach to finding a low-dimensional representation for manifold data in high dimensional Euclidean spaces.

It consists of the following steps:

1. Build a neighborhood (dissimilarity) graph from the given data
2. Compute the shortest-path distances along the graph
3. Apply MDS to find a low-dimensional representation

The goal of ISOmap is to directly preserve the global (nonlinear) geometry.

In contrast, Laplacian Eigenmaps will focus on preserving the local geometry - *nearby points in the original space remain nearby in the reduced space*.

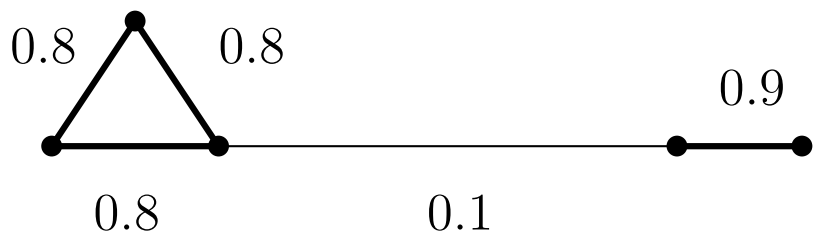
It consists of the following steps:

1. Build a similarity graph from the given data
2. Compute the graph Laplacian matrix
3. Use the eigenvectors of the Laplcian matrix to form a low-dimensional embedding of the data

Similarity graphs

A similarity graph is a weighted graph whose edge weights are levels of similarities of the connected vertices.

For example, the following is a similarity graph on 5 vertices:



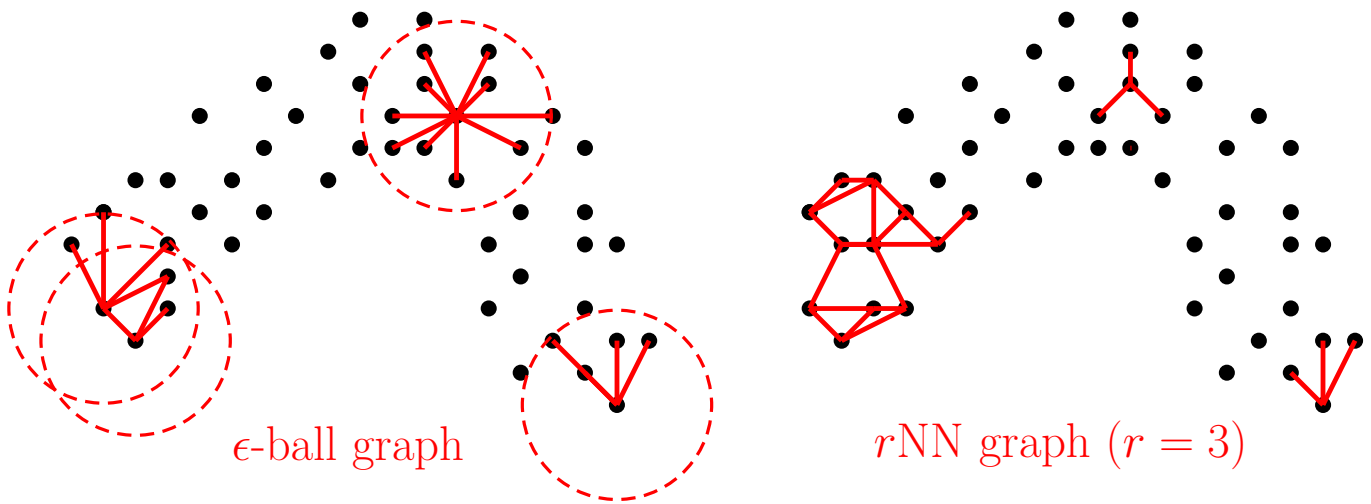
Its weight matrix is displayed below:

$$\mathbf{W} = \begin{pmatrix} 0 & .8 & .8 & 0 & 0 \\ .8 & 0 & .8 & 0 & 0 \\ .8 & .8 & 0 & .1 & 0 \\ 0 & 0 & .1 & 0 & .9 \\ 0 & 0 & 0 & .9 & 0 \end{pmatrix}$$

In general, $\mathbf{W}$ is square, symmetric, and nonnegative.

How to construct similarity graphs from data

- **The ϵ-neighborhood graph**: connect with weight 1 any two points $\mathbf{x}_i, \mathbf{x}_j$ whose distance is less than ϵ
- **The rNN graph**: connect with weight 1 any two points $\mathbf{x}_i, \mathbf{x}_j$ if one is among the r nearest neighbors of the other;



- **The fully connected graph**: connect any two points $\mathbf{x}_i, \mathbf{x}_j$ with weight according to some similarity function s:

$$w_{ij} = s(\mathbf{x}_i, \mathbf{x}_j), \quad \text{for all } i, j = 1, \ldots, n$$

  For example,

  - **Gaussian weights**: $s(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$, where $\sigma > 0$ is a scale parameter whose value is fixed.
  - **Cosine weights**: $s(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|} \cdot \frac{\mathbf{x}_j}{\|\mathbf{x}_j\|}$, which is often used in documents clustering

It is also possible to mix up the different kinds of graphs.

1D dimension reduction by Laplacian Eigenmaps

Assuming a weighted similarity graph (constructed on the given data set), we first consider the problem of **mapping the graph to a line** in a way such that *close neighbors on the graph are still close on the line*. ⟵ Locality-preserving

Let $\mathbf{f} = (f_1, \ldots, f_n)^T$ represent the 1D embedding of the nodes. We then formulate the following problem:

$$\min_{\mathbf{f} \in \mathbb{R}^n} \frac{1}{2} \sum_i \sum_j w_{ij} (f_i - f_j)^2$$

Interpretation:

- If w_{ij} is large (close to 1, meaning $\mathbf{x}_i, \mathbf{x}_j$ are originally very close), then f_i, f_j must still be close (otherwise there is a heavy penalty).
- If w_{ij} is small (close to 0, meaning $\mathbf{x}_i, \mathbf{x}_j$ are originally very far), then there is much flexibility in putting f_i, f_j on the line.

However, the problem

$$\min_{\mathbf{f}\in\mathbb{R}^n} \frac{1}{2}\sum_i\sum_j w_{ij}(f_i - f_j)^2$$

is not well defined yet. Why?

To remove the scaling and translational invariances in $\mathbf{f}$ (and get rid of the trivial solutions $\mathbf{0}, \mathbf{1}$), we add the following constraints on $\mathbf{f}$ (for now):

$$\min_{\mathbf{f}\in\mathbb{R}^n} \frac{1}{2}\sum_i\sum_j w_{ij}(f_i - f_j)^2$$

subject to

$$\mathbf{f}^T\mathbf{1} = \sum f_i = 0, \quad \|\mathbf{f}\|^2 = \sum f_i^2 = 1.$$

Equivalently, it can be reformulated as

$$\min_{\mathbf{f}^T\mathbf{1}=0,\,\|\mathbf{f}\|=1} \frac{1}{2}\sum_i\sum_j w_{ij}(f_i-f_j)^2, \quad \text{or} \quad \min_{\mathbf{f}\neq\mathbf{0},\,\mathbf{f}^T\mathbf{1}=0} \frac{\frac{1}{2}\sum_i\sum_j w_{ij}(f_i - f_j)^2}{\sum_i f_i^2}$$

Spectral graph theory (a little bit)

Let $G = (V, E, \mathbf{W})$ be a weighted graph with vertices $V = \{1, \ldots, n\}$ and weights $w_{ij} \geq 0$ (there is an edge $e_{ij} \in E$ connecting nodes i and j if and only if $w_{ij} > 0$).

The *degree* of a vertex $i \in V$ is defined as $d_i = \sum_{j=1}^{n} w_{ij}$. It measures the connectivity of the vertex in the graph.

The degrees of all vertices can be used to form a *degree matrix*

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

An equivalent way of defining the degree matrix is $\mathbf{D} = \mathrm{diag}(\mathbf{W1})$.

Example 0.1. For the following graph, $\mathbf{D} = \mathrm{diag}(1.6, 1.6, 1.7, 1, 0.9)$.

$$\mathbf{W} = \begin{pmatrix} 0 & .8 & .8 & 0 & 0 \\ .8 & 0 & .8 & 0 & 0 \\ .8 & .8 & 0 & .1 & 0 \\ 0 & 0 & .1 & 0 & .9 \\ 0 & 0 & 0 & .9 & 0 \end{pmatrix}$$

A **subgraph** of a given graph $G = (V, E, \mathbf{W})$ is another graph, formed from a subset of the vertices of the graph, $A \subset V$ by keeping only all of the edges connecting pairs of vertices in A.

A **path** in the graph is a sequence of vertices and edges in between such that no vertex or edge can repeat.

A subgraph $A \subset V$ of a graph is **connected** if any two vertices in A can be joined by a path such that all intermediate points also lie in A.

A subgraph $A \subset V$ is called a **connected component** if it is connected and if there are no edges between A and its complement $\bar{A} = V - A$.

A graph is said to be connected if it has only one connected component.

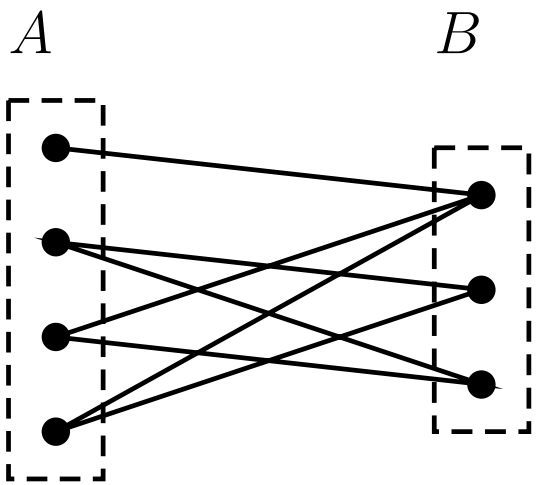
Example 0.2. The following graph has only 1 connected component, and thus is a *connected graph*.

The left three nodes (and the three edges connecting them to each other) form a subgraph, and is connected (but is not a connected component).

A graph $G = (V, E, \mathbf{W})$ is called a **bipartite graph** if there is a partition of the nodes $V = A \cup B$ such that there is no edge inside each of the two parts A and B:

$$w_{ij} = 0,\ i, j \in A, \qquad \text{and } w_{ij} = 0,\ i, j \in B.$$

In other words, all the edges in E are between A and B.



The graph Laplacian is a very important (yet challenging) concept in spectral graph theory.

Def 0.1. Given a graph $G = (V, E, \mathbf{W})$ with size $|V| = n$, the **graph Laplacian** is defined as the following matrix

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \in \mathbb{R}^{n \times n}, \qquad \text{where} \quad \mathbf{D} = \text{diag}(\mathbf{W1}).$$

Example 0.3. For the previous graph, the graph Laplacian matrix is

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

The graph Laplacian has many interesting properties.

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

- $\mathbf{L}$ is symmetric.
- All the rows (and columns) sum to 0, i.e., $\mathbf{L}\mathbf{1} = \mathbf{0}$. This implies that $\mathbf{L}$ has a eigenvalue 0 with eigenvector $\mathbf{1} \in \mathbb{R}^n$.
- For every vector $\mathbf{f} \in \mathbb{R}^n$ we have

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

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

- The algebraic (and also geometric) multiplicity of the eigenvalue 0 equals the number of connected components of the graph.

Proof. The first two are obvious. We prove the third result below:

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

and skip the proof for the last one. □

Example 0.4. For the graph below (which is connected), the eigenvalues of the graph Laplacian are $0 < 0.0788 < 1.8465 < 2.4000 < 2.4747$.

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

Example 0.5. Consider the modified graph by removing the middle edge with weight 0.1 (which now has two connected components)

$$\mathbf{W} = \begin{pmatrix} 0 & .8 & .8 & 0 & 0 \\ .8 & .0 & .8 & 0 & 0 \\ .8 & .8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .9 \\ 0 & 0 & 0 & .9 & 0 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1.6 & -0.8 & -0.8 & & \\ -0.8 & 1.6 & -0.8 & & \\ -0.8 & -0.8 & 1.6 & & \\ & & & 0.9 & -0.9 \\ & & & -0.9 & 0.9 \end{pmatrix}$$

It can be shown that

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

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

Returning to the 1D Laplacian Eigenmaps problem

which embeds the nodes of a similarity graph $G = (V, E, \mathbf{W})$ into a line:

$$\min_{\substack{\mathbf{f} \neq \mathbf{0} \in \mathbb{R}^n \\ \mathbf{f}^T \mathbf{1} = 0}} \frac{\frac{1}{2} \sum_i \sum_j w_{ij} (f_i - f_j)^2}{\sum_i f_i^2}.$$

Applying the theorem on graph Laplacians, we can rewrite the above problem as follows:

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

Again, we have encountered a Rayleigh quotient problem (but with an extra constraint this time)!

Without the extra constraint $\mathbf{f}^T\mathbf{1} = 0$, a minimizer of the Rayleigh quotient is an eigenvector of the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$ corresponding to the smallest eigenvalue $\lambda_1 = 0$, i.e.,

$$\mathbf{v}_1 = \mathbf{1}.$$

However, as previously pointed out, this is a trivial solution which puts all nodes of the graph at the same point of a line.

With the extra constraint, we force $\mathbf{f}$ to be perpendicular to the eigenvector $\mathbf{1}$. The minimizer of this new problem is given by the second smallest eigenvector of $\mathbf{L}$:

$$\mathbf{f}^* = \mathbf{v}_2,$$

and the minimum value of the Rayleigh quotient is λ_2.

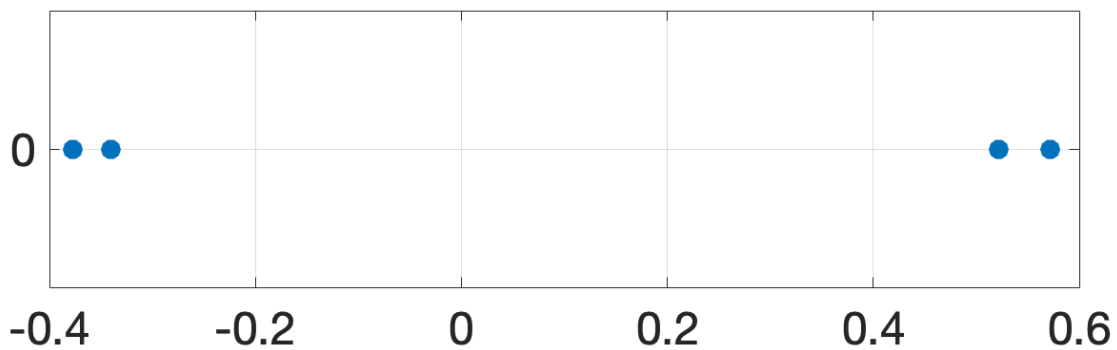
If the similarity graph is connected (which is the interesting, nontrivial case), the algebraic multiplicity of the eigenvalue 0 is one.

Consequently, we must have $\lambda_2 > 0$, and

$$0 < (\mathbf{f}^*)^T \mathbf{L}\mathbf{f}^* = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i^* - f_j^*)^2$$

This shows that $\mathbf{f}^* = \mathbf{v}_2$ will lead to a nontrivial embedding of the graph.

Example 0.6. For the graph below (which is connected), the 2nd smallest eigenvector is $\mathbf{v}_2 = (-.3771, -.3771, -.3400, .5221, .5722)$.



So far so good (for the sake of presenting ideas), but the original Laplacian Eigenmaps algorithm proposed by Belkin and Niyogi (2003) corresponds to solving the following problem:

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

where

- The denominator $\mathbf{f}^T\mathbf{D}\mathbf{f}$ is for removing the scaling factor in $\mathbf{f}$, and
- The condition $\mathbf{f}^T\mathbf{D}\mathbf{1} = 0$ is for removing the translational invariance:

$$0 = \mathbf{f}^T\mathbf{D}\mathbf{1} = \sum d_i f_i$$

  and also for removing a trivial solution, which we show later.

To better understand the situation, we need to study the matrix $\mathbf{D}^{-1}\mathbf{L}$, which is a normalized graph Laplacian.

Def 0.2. For any graph $G = (V, E, \mathbf{W})$ with graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$, let

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

It is called the **random-walk normalized graph Laplacian**, because the matrix $\mathbf{P}$ is the ℓ_1 row normalized version of $\mathbf{W}$ and thus is row-stochastic.

Remark. It is easy to show that

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

That is, $(\lambda, \mathbf{v})$ is an eigenpair for $\widetilde{\mathbf{L}}_{\text{rw}}$ if and only if $(1-\lambda, \mathbf{v})$ is an eigenpair for $\mathbf{P}$.

This relationship is useful in computing, as later we will need to compute the bottom eigenvectors of $\widetilde{\mathbf{L}}_{\text{rw}}$, which can be equivalently computed as the top eigenvectors of $\mathbf{P}$.

Example 0.7. For the graph below (which is connected),

the normalized graph Laplacian is

$$\widetilde{\mathbf{L}}_{\mathrm{rw}} = \begin{pmatrix} 1 & -0.5 & -0.5 & & \\ -0.5 & 1 & -0.5 & & \\ -0.4706 & -0.4706 & 1 & -0.0588 & 0 \\ & & -0.1 & 1 & -0.9 \\ & & & -1 & 1 \end{pmatrix}$$

Theorem 0.2. Properties of the normalized graph Laplacian:

- $\widetilde{\mathbf{L}}_{\text{rw}}\mathbf{1} = \mathbf{0}$ (rows sums are 1; it has eigenvalue 0 with eigenvector $\mathbf{1}$).
- $\widetilde{\mathbf{L}}_{\text{rw}}$ is asymmetric, but has n nonnegative eigenvalues

$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$$

  Additionally, the multiplicity of the 0 eigenvalue is also equal to the number of connected components in the graph.

- For all weighted graphs, $\lambda_n \leq 2$, with bipartite graphs attaining the upper bound.

Now consider the original Laplacian Eigenmaps problem again:

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

This is a restricted generalized Rayleigh quotient problem, with the smallest generalized eigenvector $\mathbf{1}$ being excluded.

Thus, the minimizer is given by the second smallest eigenvector of $\widetilde{\mathbf{L}}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{L}$:

$$\widetilde{\mathbf{L}}_{\text{rw}}\mathbf{v}_2 = \lambda_2\mathbf{v}_2 \quad \Longleftrightarrow \quad \mathbf{L}\mathbf{v}_2 = \lambda_2\mathbf{D}\mathbf{v}_2.$$

Example 0.8. For the graph below (which is connected),

the normalized graph Laplacian $\widetilde{\mathbf{L}}_{\mathrm{rw}}$ has the following eigenvalues

$$0 < 0.0693 < 1.4773 < 1.5000 < 1.9534$$

Its second smallest eigenvector (corresponding to $\lambda_2 = 0.0693$) is

$$\mathbf{v}_2 = (-0.2594, -0.2594, -0.2235, 0.6152, 0.6610).$$

Remark. For which graph Laplacian, $\mathbf{L}, \widetilde{\mathbf{L}}_{\text{rw}}$, should we use its eigenvectors for embedding graph data?

They correspond to two different formulations of the embedding problem:

$$\min_{\substack{\mathbf{f}\neq\mathbf{0}\in\mathbb{R}^n \\ \mathbf{f}^T\mathbf{1}=0}} \frac{\mathbf{f}^T\mathbf{L}\mathbf{f}}{\mathbf{f}^T\mathbf{f}}, \qquad \text{versus} \qquad \min_{\substack{\mathbf{f}\neq\mathbf{0}\in\mathbb{R}^n \\ \mathbf{f}^T\mathbf{D}\mathbf{1}=0}} \frac{\mathbf{f}^T\mathbf{L}\mathbf{f}}{\mathbf{f}^T\mathbf{D}\mathbf{f}}.$$

The two criteria work (nearly) the same when all nodes of the graph have (nearly) the same degrees (i.e., $\mathbf{D} \approx \gamma\mathbf{I}$ for some $\gamma > 0$).

In general, the normalized graph Laplacian should be used.

Embedding graph data to 2D or higher

To produce a k-dimensional embedding of the nodes of a connected graph $G = (V, E, \mathbf{W})$, one can just take more eigenvectors of the normalized Laplacian $\widetilde{\mathbf{L}}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{L}$:

$$\widetilde{\mathbf{L}}_{\text{rw}}\mathbf{v}_i = \lambda_i \mathbf{v}_i \iff \mathbf{L}\mathbf{v}_i = \lambda_i \mathbf{D}\mathbf{v}_i, \quad i = 2, \ldots, k+1$$

to form the embedding matrix

$$\mathbf{Y} = [\mathbf{v}_2, \ldots, \mathbf{v}_{k+1}] \in \mathbb{R}^{n \times k}$$

(Rows of $\mathbf{Y}$ are new coordinates for the original data points $\mathbf{x}_i \in \mathbb{R}^d$)

The Laplacian Eigenmaps algorithm

Input: Similarity graph $G = (V, E, \mathbf{W})$, embedding dimension k

Output: A k-dimensional representation of the input data ($\mathbf{Y} \in \mathbb{R}^{n \times k}$).

1. Compute the row-stochastic matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$.
2. Find the the 2nd to $(k+1)$st largest eigenvectors of $\mathbf{P}$ (note that $\lambda_1 = 1, \mathbf{v}_1 = \mathbf{1}$):

$$\mathbf{P}\mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad i = 2, \ldots, k+1$$

3. Return: $\mathbf{Y} = [\mathbf{v}_2 \ldots \mathbf{v}_{k+1}] \in \mathbb{R}^{n \times k}$.

Implementation details

Assume vector data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ combined with Gaussian weights:

$$w_{ii} = 0; \qquad w_{ij} = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2)\right), \ i \neq j$$

The parameter σ can be set directly as the average distance of the data points to their respective rNNs in the data: $\sigma = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{x}_i^{(r\text{nn})}\|$.

For fast speed, use a subset of 30 to 50 randomly selected points to calculate σ. Additionally, $r = \mathcal{O}(\log(n))$ and typically, r is 6 to 10.

When the data set has several groups, the embedding dimension k should be set to the number of groups.

Computer demonstration

Comments on Laplacian Eigenmaps

Handles nonlinear geometry well.

Can reveal/separate clusters (by mapping points in each cluster together).

Choice of the parameter σ in the Gaussian similarity function is important.

High computational complexity though.

Connections to spectral clustering

Laplacian Eigenmaps is originally proposed as a nonlinear dimension reduction method by preserving local geometry of the given data.

In fact, the new coordinates found by the algorithm, $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^T \in \mathbb{R}^{n \times k}$, can be directly used for clustering purposes:

$$\mathbf{x}_i \in \mathbb{R}^d \quad \longmapsto \quad \mathbf{y}_i \in \mathbb{R}^k, \quad i = 1, \ldots, n$$

The combination of Laplacian Eigemaps with k-means (for the clustering step) is exactly the Normalized Cut algorithm proposed by Shi and Malik (2000), which is one of the standard spectral clustering methods.

What is spectral clustering?

A family of clustering algorithms that utilize the **spectral decomposition of a similarity matrix** constructed on the given data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$:

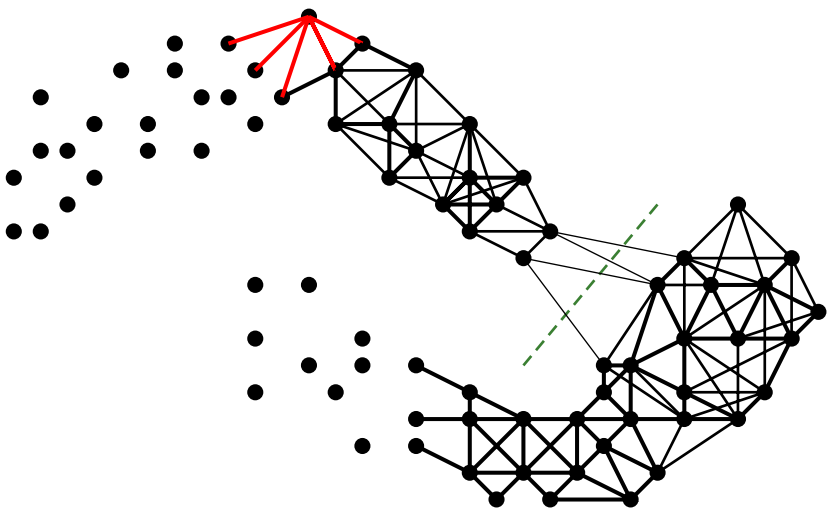
$$\mathbf{W} = (w_{ij}) \in \mathbb{R}^{n \times n}, \quad w_{ij} = \begin{cases} s(\mathbf{x}_i, \mathbf{x}_j), & \text{if } i \neq j \\ 0, & \text{if } i = j. \end{cases}$$

Here, $s(\cdot, \cdot)$ is a similarity function, such as

- a 0/1-valued **indicator function**,
- the **Gaussian radial basis function (RBF)**, and
- the **cosine similarity**.

SC via a graph cut point of view

$\mathbf{W}$ (as a weight matrix) defines a weighted graph on the given data.



Therefore, **clustering = finding an optimal cut** (under some criterion).

Some graph terminology:

–**Degree matrix**: $\mathbf{D} = \mathrm{diag}(\mathbf{W1})$ with $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ij}$.

–**Graph Laplacian**: $\mathbf{L} = \mathbf{D} - \mathbf{W}$ and its normalized version:

$$\mathbf{L}_{\mathrm{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \underbrace{\mathbf{D}^{-1}\mathbf{W}}_{\mathbf{P}\ \text{(row stochastic)}}$$

Remark. $\mathbf{P}$ defines a random walk on the graph.

We (need to) introduce more graph terminology below.

Given a subset of vertices $A \subset V$, we define the *indicator vector* $\mathbf{1}_A$ of A as

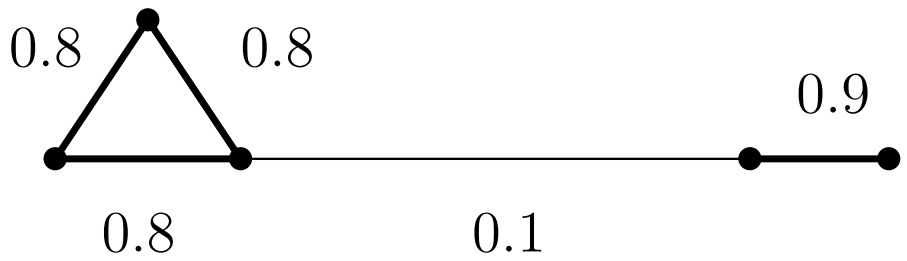
$$\mathbf{1}_A = (a_1, \ldots, a_n)^T, \quad a_i = 1 \text{ (if } i \in A) \text{ and } a_i = 0 \text{ (if } i \in \bar{A}).$$

There are two ways to measure the "size" of a subset $A \subset V$:

$$|A| = \#\text{vertices in } A;$$
$$\text{Vol}(A) = \sum_{i \in A} d_i$$

The former simply counts the number of vertices in A while the latter measures how strongly the vertices in A are connected to all vertices of G.

Example 0.9. In the graph below, the left three vertices induce a subgraph A with $\mathbf{1}_A = (1,1,1,0,0)^T$, $|A| = 3$ and $\mathrm{Vol}(A) = 1.6 + 1.6 + 1.7 = 4.9$.



$$\mathbf{D} = \begin{pmatrix} 1.6 & & & & \\ & 1.6 & & & \\ & & 1.7 & & \\ & & & 1 & \\ & & & & 0.9 \end{pmatrix}$$

For any two subsets $A, B \subset V$ (not necessarily disjoint), define

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

Two special cases:

- If $B = \bar{A}$, $W(A, \bar{A})$ is called a cut:

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

- If $B = V$,

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

To find the "optimal" bipartition of a graph $V = A \cup B$ with $B = \bar{A}$, Shi and Malik (2003) proposed to minimize the following normalized cut

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

such that

- $\mathrm{Cut}(A, B)$ is as small as possible (minimal loss of edge weights);
- both $\mathrm{Vol}(A)$ and $\mathrm{Vol}(B)$ are large (for achieving a balanced cut).

This is a combinatorial optimization problem which is NP-hard.

To solve the NCut problem, consider any partition $V = A \cup B$. Denote $\mathrm{Vol}(A) = a, \mathrm{Vol}(B) = b$.

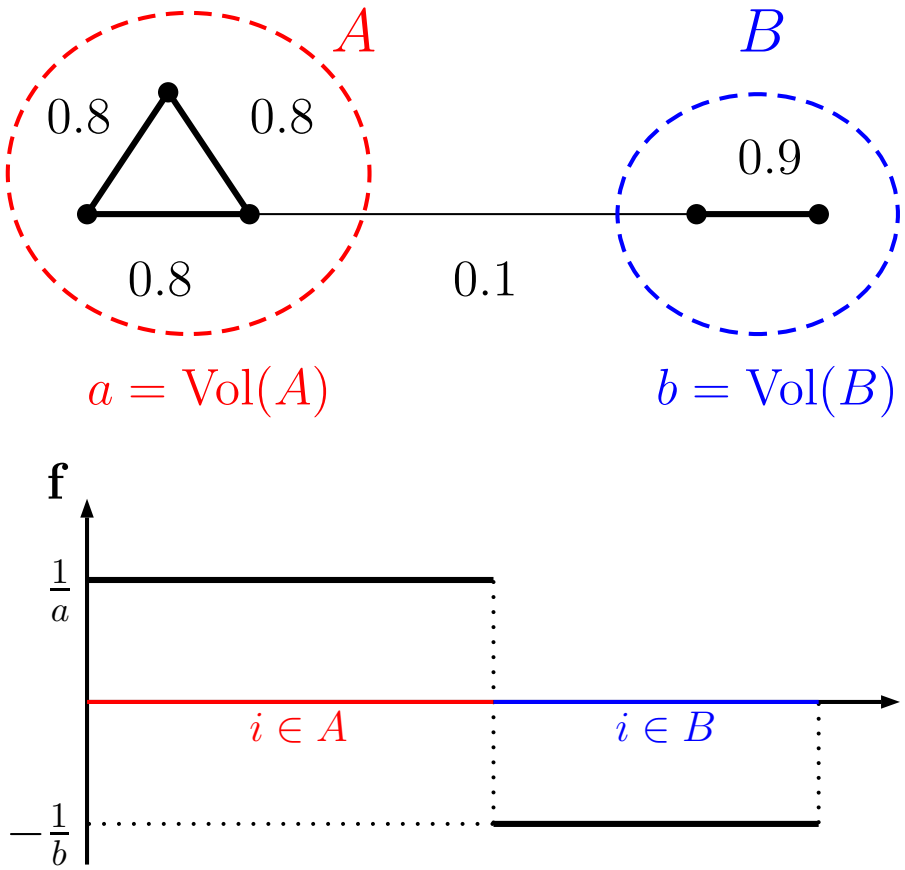
Define

$$\mathbf{f} = \frac{1}{a}\mathbf{1}_A - \frac{1}{b}\mathbf{1}_B \in \mathbb{R}^n$$

with

$$f_i = \begin{cases} \frac{1}{a}, & i \in A \\ -\frac{1}{b}, & i \in B \end{cases}$$

Note that f is an indicator variable for the bipartition.



We have

$$\begin{aligned}
\mathbf{f}^T\mathbf{L}\mathbf{f} &= \sum_{i,j} w_{ij}(f_i - f_j)^2 \\
&= \sum_{i\in A,\, j\in B} w_{ij}\left(\frac{1}{a}+\frac{1}{b}\right)^2 \\
&= \mathrm{Cut}(A,B)\left(\frac{1}{a}+\frac{1}{b}\right)^2 \\
\mathbf{f}^T\mathbf{D}\mathbf{f} &= \sum_i d_{ii} f_i^2 \\
&= \sum_{i\in A} \frac{1}{a^2} d_{ii} + \sum_{j\in B} \frac{1}{b^2} d_{ii} \\
&= \frac{1}{a^2}\,\mathrm{Vol}(A) + \frac{1}{b^2}\,\mathrm{Vol}(B) = \frac{1}{a}+\frac{1}{b}
\end{aligned}$$

It follows that

$$\frac{\mathbf{f}^T\mathbf{L}\mathbf{f}}{\mathbf{f}^T\mathbf{D}\mathbf{f}} = \mathrm{Cut}(A,B)\left(\frac{1}{a}+\frac{1}{b}\right) = \mathrm{NCut}(A,B)$$

Additionally, $\mathbf{f}$ satisfies

$$\mathbf{f}^T\mathbf{D}\mathbf{1} = \sum_i f_i d_{ii} = \sum_{v_i\in A}\frac{1}{a}d_{ii} - \sum_{v_i\in B}\frac{1}{b}d_{ii} = \frac{1}{a}\mathrm{Vol}(A) - \frac{1}{b}\mathrm{Vol}(B) = 0$$

Therefore, we can obtain the following equivalent problem

$$\min_{\substack{A\cup B=V\\ A\cap B=\emptyset}} \mathrm{NCut}(A,B) \quad\Longleftrightarrow\quad \min_{\substack{\mathbf{f}\in\{\alpha,-\beta\}^n\\ \mathbf{f}^T\mathbf{D}\mathbf{1}=0}} \frac{\mathbf{f}^T\mathbf{L}\mathbf{f}}{\mathbf{f}^T\mathbf{D}\mathbf{f}}$$

This problem is still discrete in nature. To find an approximate solution, we eliminate the condition $\mathbf{f} \in \{\alpha, -\beta\}^n$ to solve the relaxed problem

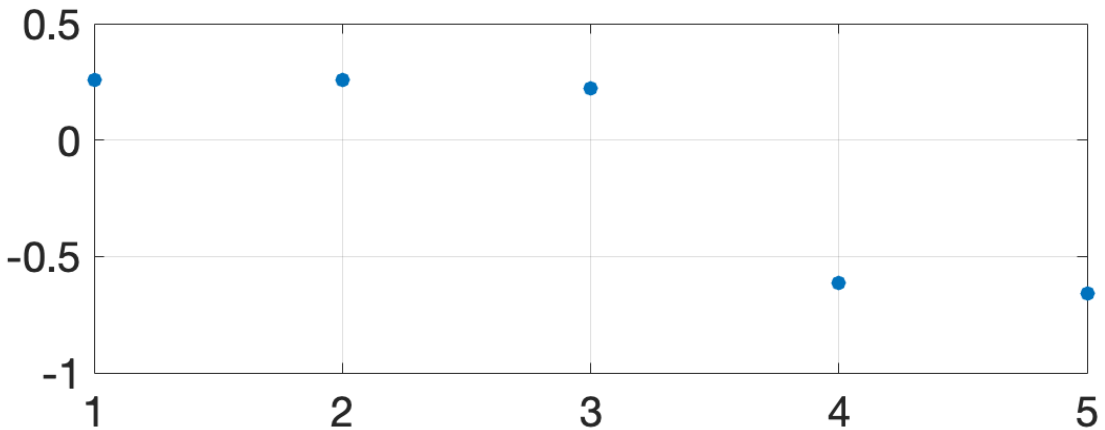
$$\min_{\substack{\mathbf{f} \neq \mathbf{0} \in \mathbb{R}^n \\ \mathbf{f}^T \mathbf{D} \mathbf{1} = 0}} \frac{\mathbf{f}^T \mathbf{L} \mathbf{f}}{\mathbf{f}^T \mathbf{D} \mathbf{f}}$$

This is exactly the same generalized Rayleigh quotient problem we obtained for Laplacian Eigenmaps, with the same minimizer $\mathbf{f}^* = \mathbf{v}_2$ (the second smallest eigenvector of $\widetilde{\mathbf{L}}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{L}$).

New interpretation: The eigenvector $\mathbf{v}_2$ represents an approximate solution to the NCut problem, providing information about the labels of the data.

Example 0.10. For the graph below (which is connected),

the second smallest eigenvector of the normalized graph Laplacian $\widetilde{\mathbf{L}}_{\mathrm{rw}}$ is $\mathbf{v}_2 = (0.2594, 0.2594, 0.2235, -0.6152, -0.6610)$.



Computer demonstration

Remark. The RatioCut algorithm uses $|\cdot|$ instead of $\mathrm{Vol}(\cdot)$ to measure the size of each cluster so as to seek a balanced cut:

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

It can be shown to lead to the following relaxed problem

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

whose solution is given by the second smallest eigenvector of $\mathbf{L}$.

In general, the NCut algorithm works better, especially when the cluster sizes vary a lot.

What if $k > 2$?

Use the subsequent eigenvectors (besides $\mathbf{v}_2$) of $\mathbf{P}$:

$$\mathbf{V} = [\mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_k] \in \mathbb{R}^{n \times (k-1)}$$

since they represent suboptimal 2-way partitions.

Now regard the rows of $\mathbf{V}$ as an embedding of the original data in $\mathbf{X}$,

$$\mathbf{X}(i,:) \in \mathbb{R}^d \quad \longrightarrow \quad \mathbf{V}(i,:) \in \mathbb{R}^{k-1}, \quad i = 1, \ldots, n$$

and apply k-means to group the row vectors of $\mathbf{V}$ into k clusters.

Algorithm 1 Normalized Cut (by Shi and Malik)

Input: Data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, #clusters k, scale parameter σ

Output: A partition $C_1, \ldots, C_k$

1: Construct a weighted graph by assigning weights

$$\mathbf{W} = (w_{ij}), \quad w_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$$

2: Find the degree matrix $\mathbf{D} = \mathrm{diag}(\mathbf{W1})$ and use it to normalize $\mathbf{W}$ to get $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$.

3: Find the 2nd to kth largest eigenvectors $\mathbf{V} = [\mathbf{v}_2 \ldots \mathbf{v}_k]$ of $\mathbf{P}$.

4: Apply k-means to group the rows of $\mathbf{V}$ into k clusters.

Computer demonstration

Computational challenges

Spectral clustering has achieved superior results in many applications (such as image segmentation, documents clustering, social network partitioning), but requires significant computational power:

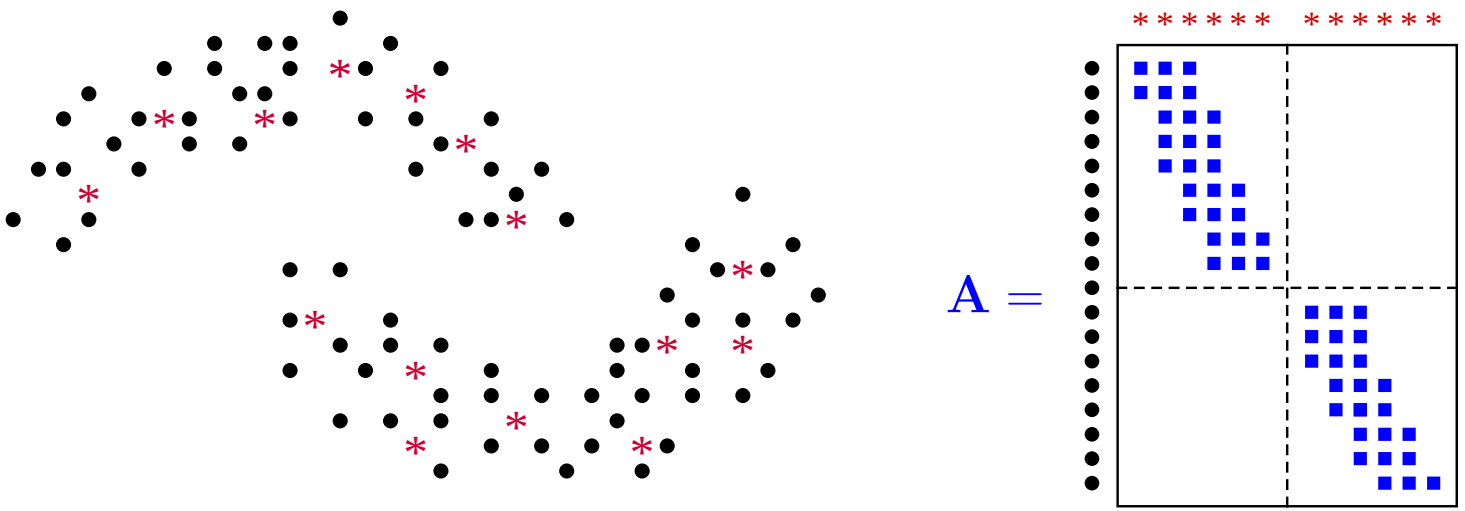
- Extensive memory requirement by $\mathbf{W} \in \mathbb{R}^{n \times n}$: $\mathcal{O}(n^2)$
- High computational cost:
  - Construction of $\mathbf{W}$: $\mathcal{O}(n^2 d)$
  - Spectral decomposition of $\mathbf{W}$: $\mathcal{O}(n^3)$

Consequently, there has been an urgent need to develop **fast**, **approximate** spectral clustering algorithms that are **scalable to large data**.

Landmark-based scalable methods

Most existing scalable methods use a small landmark set $\mathbf{y}_1, \ldots, \mathbf{y}_m \in \mathbb{R}^d$, selected from the **given data** $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ (e.g., uniformly at random or via k-means), to construct a (sparse) similarity matrix between them:

$$\mathbf{A} = (a_{ij}) \in \mathbb{R}^{n \times m} \ (m \ll n), \quad a_{ij} = s(\mathbf{x}_i, \mathbf{y}_j) \text{ for } r \text{ nearest } \mathbf{y}_j$$

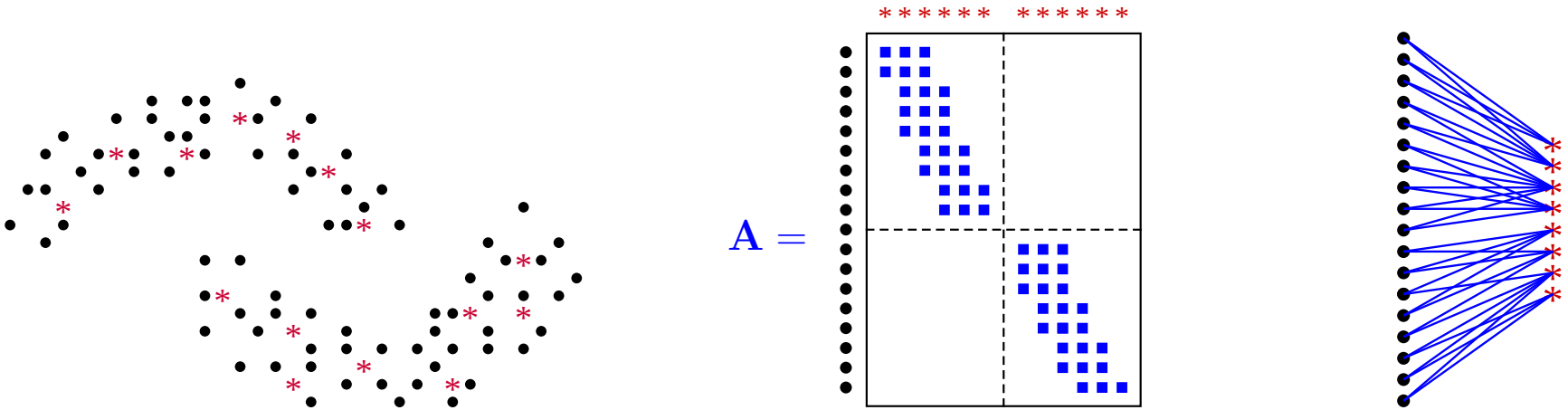


Afterwards, different methods use the similarity matrix $\mathbf{A}$ in different ways:

- **cSPEC** (Wang et al., 2009): Regards $\mathbf{A}$ as a **column-sampled** version of $\mathbf{W}$ and uses linear algebra to estimate eigenvectors of $\mathbf{W}$
- **KASP** (Yan, Huang and Jordan, 2009): Uses **vector quantization** technique (i.e., $k$means) to aggressively reduce the given data to a collection of centroids (landmarks) and applies spectral clustering to group them
- **LSC** (Cai and Chen, 2015): Obtains the matrix $\mathbf{A}$ from a **sparse coding** perspective with the landmarks as a dictionary and then applies spectral clustering to the rows of $\mathbf{A}$ (after performing certain row and column normalizations).

Overview of our approaches

We propose two new landmark-based scalable spectral clustering methods:
(1) **The documents model**: We regard $\mathbf{A}$ as a "documents" data set and cluster them based on the **cosine similarity**.
(2) **The bipartite graph model**: We use $\mathbf{A}$ to form a bipartite graph with the given data and selected landmarks as the two parts.



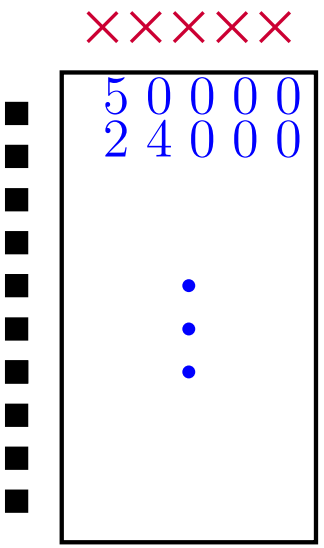
Main papers:

1. "Large-scale spectral clustering using diffusion coordinates on landmark based bipartite graphs", K. Pham and G. Chen (TextGraphs 2018, New Orleans, LA)
2. "Scalable spectral clustering with cosine similarity", G. Chen (ICPR 2018, Beijing)
3. "A general framework for scalable spectral clustering based on document models", G. Chen (Pattern Recognition Letters, June 2019)

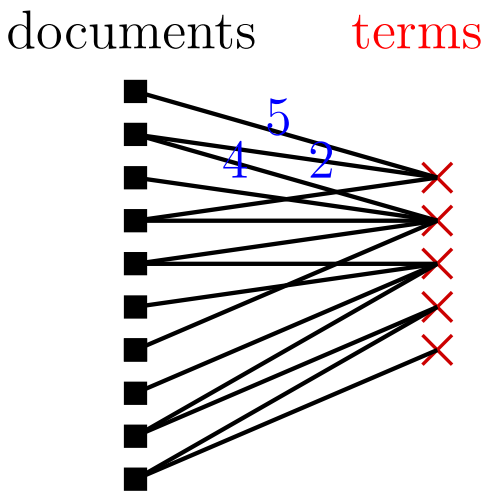
We present the **bipartite graph model** next.

Motivation

Dhillon (2001) proposed a **bipartite graph** model for the setting of documents data, with the goal to **co-cluster documents and terms**.



Frequency matrix (under bag of words model)



The vertices of the bipartite graph are documents and terms combined.

The weight matrix is

$$\mathbf{W} = \begin{pmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}.$$

Note that there is no edge inside each part of the graph.

Next, in principle, they just apply the NCut algorithm to the bipartite graph with $\mathbf{W}$ as the weight matrix, in order to co-cluster the documents and terms.

Computing the eigenvectors of $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ directly is costly because of the size of $\mathbf{W}$. However, there is a shortcut method by using the SVD of the following normalized version of $\mathbf{A}$:

$$\widetilde{\mathbf{A}} = \mathbf{D}_1^{-1/2}\mathbf{A}\mathbf{D}_2^{-1/2},$$

where

$$\mathbf{D}_1 = \mathrm{diag}(\mathbf{A}\mathbf{1}), \quad \mathbf{D}_2 = \mathrm{diag}(\mathbf{A}^T\mathbf{1})$$

contain the row and column sums of $\mathbf{A}$, respectively.

Let the singular values and singular vectors of $\widetilde{\mathbf{A}}$ be

$$\widetilde{\mathbf{A}}\widetilde{\mathbf{v}}_i = \sigma_i \widetilde{\mathbf{u}}_i, \quad 1 \le i \le m.$$

Define

$$\mathbf{w}_i = \begin{pmatrix} \mathbf{D}_1^{-1/2} & \\ & \mathbf{D}_2^{-1/2} \end{pmatrix} \begin{pmatrix} \widetilde{\mathbf{u}}_i \\ \widetilde{\mathbf{v}}_i \end{pmatrix} = \begin{pmatrix} \mathbf{D}_1^{-1/2}\widetilde{\mathbf{u}}_i \\ \mathbf{D}_2^{-1/2}\widetilde{\mathbf{v}}_i \end{pmatrix}.$$

Then

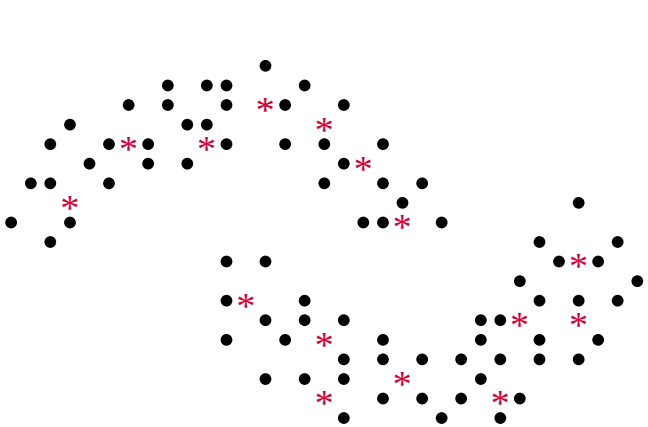
$$\mathbf{P}\mathbf{w}_i = \sigma_i \mathbf{w}_i, \quad 1 \le i \le m$$

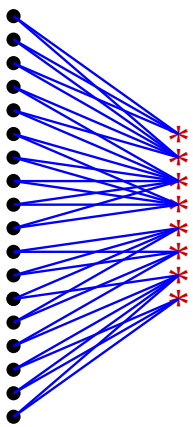
This shows that $\mathbf{P}$ has eigenvalues σ_i with corresponding eigenvectors $\mathbf{w}_i$.

Note that in each $\mathbf{w}_i$ space, the documents and terms appear together, thus enabling co-clustering to be done (via $k$means).

Our bipartite graph model

We adapt the bipartite graph model by Dhillon (2001) for landmark-based clustering by using the given data and a landmark set as the two parts.





We then apply $k$means in the eigenvector space to first co-cluster the data and landmarks (and then remove the landmark points later).

Algorithm 2: Scalable spectral clustering

Input:

- Data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$
- similarity function s (e.g., Gaussian)
- #clusters k
- #landmark points m (at most a few hundred)
- #nearest landmark points r (between 3 and 10)

Output: Clusters $C_1, \ldots, C_k$

Steps:

1. Select m landmark points $\{\mathbf{y}_j\}$ by uniform sampling.
2. Compute the similarity matrix $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{n\times m}, a_{ij} = s(\mathbf{x}_i, \mathbf{y}_j)$ between each given data point $\mathbf{x}_i$ and the r nearest landmarks $\mathbf{y}_j$.
3. Find the row and columns of $\mathbf{A}$: $\mathbf{D}_1 = \mathrm{diag}(\mathbf{A1}), \mathbf{D}_2 = \mathrm{diag}(\mathbf{A}^T\mathbf{1})$, and use them to normalize $\mathbf{A}$: $\widetilde{\mathbf{A}} = \mathbf{D}_1^{-1/2}\mathbf{A}\mathbf{D}_2^{-1/2}$.
4. Perform the rank-k SVD of $\widetilde{\mathbf{A}}$ to obtain its left and right singular vector matrices, $\widetilde{\mathbf{U}}_k \in \mathbb{R}^{n\times k}$ and $\widetilde{\mathbf{V}}_k \in \mathbb{R}^{m\times k}$.
5. Apply $k$means to cluster the rows of $\begin{pmatrix} \mathbf{D}_1^{-1/2}\widetilde{\mathbf{U}}_k \\ \mathbf{D}_2^{-1/2}\widetilde{\mathbf{V}}_k \end{pmatrix}$.

Algorithmic complexity

Total running time is linear in the size of the data:

$$\mathcal{O}(nmd),$$

where

- n: number of given data points
- m: number of landmark points
- d: dimension of the data

As a result, the algorithm is scalable to large data.

Computer demonstration

Numerical considerations

The total number of selected landmark points (m) should grow with the size of the data (n), but should be at most a few hundred.

The quality of the landmark points is more important. Ideally, they should be local centers and cover the given data well.

A better landmark selection method is to first use $k$means to divide the data into m small subsets and then take the centroids as landmark points (extra computational burden).