San José State University

Math 253: Mathematical Methods for Data Visualization

Laplacian Eigenmaps

Dr. Guangliang Chen

Outline of the lecture:

- Background
  - Similarity graphs
  - Spectral graph theory
- Laplacian Eigemaps
  - Dimension reduction
  - Clustering

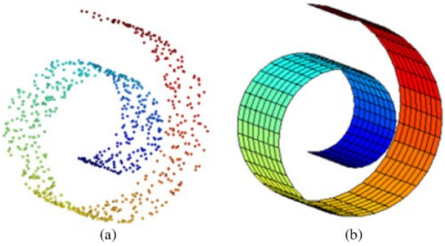
Main reference paper

"Laplacian Eigenmaps for dimensionality reduction and data representation", Mikhail Belkin and Partha Niyogi, *Neural Computation* 15, 1373–1396 (2003).

URL:
`https://www2.imm.dtu.dk/projects/manifold/Papers/Laplacian.pdf`

Introduction

Consider the **manifold unfolding** problem again: Given a set of points in a high dimensional Euclidean space but along a manifold, $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{M} \subset \mathbb{R}^d$, 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" $\mathbf{x}_i$.



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

It consists of the following steps:

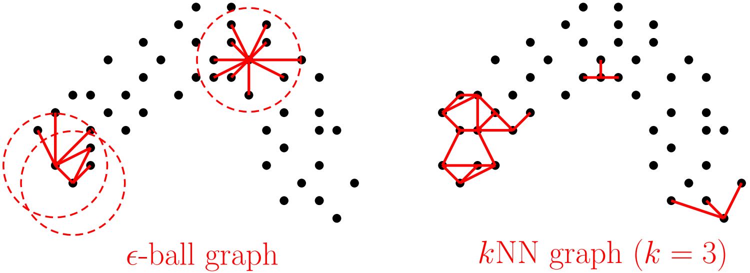
- Build a neighborhood graph from the given data
- Compute the shortest-path distances along the graph
- Apply MDS to find a low-dimensional representation

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*.

Similarity graphs

Like ISOmap, the first step of Laplacian Eigenmaps is to build a neighborhood graph G from the given data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ by connecting only "nearby" points, where nearby is defined in one of the following ways:

- ϵ**-ball approach**: Two points $\mathbf{x}_i, \mathbf{x}_j$ are nearby if $\|\mathbf{x}_i - \mathbf{x}_j\| \leq \epsilon$,
- k**NN approach**: Two points $\mathbf{x}_i, \mathbf{x}_j$ are nearby if one is among the k nearest neighbors of the other.



However, they use different kinds of weights for the connected edges:

- ISOmap: The edges are weighted by the Euclidean distances:

$$d_X(i,j) = \|\mathbf{x}_i - \mathbf{x}_j\| \quad \text{if } \mathbf{x}_i, \mathbf{x}_j \text{ are connected}$$

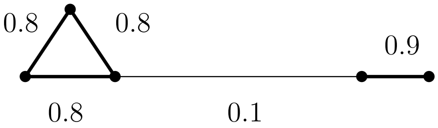
We call the correspondingly weighted graph a dissimilarity graph.

- Laplacian Eigenmaps: The Euclidean distances between nearby points are transformed to similarity scores (to be used as weights) in one of the following ways:
  - **0/1 weights**: $w_{ij} = 1$ if there is an edge between $\mathbf{x}_i, \mathbf{x}_j$
  - **Gaussian weights**: $w_{ij} = \exp(-d_X(i,j)^2/t)$ if there is an edge between $\mathbf{x}_i, \mathbf{x}_j$ ($t > 0$ is a parameter to be selected by the user)

  If there is no edge between two points $\mathbf{x}_i, \mathbf{x}_j$, we set $w_{ij} = 0$.

  Each of such weighting methods leads to a so-called similarity graph, with weights stored in a weight matrix: $\mathbf{W} = (w_{ij}) \in \mathbb{R}^{n \times n}$.

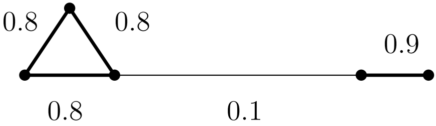
Example 0.1. The following displays a similarity graph on a set of 5 data points (called vertices or nodes), with associated weight matrix $\mathbf{W}$:



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

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 nodes will still be close on the line*. $\longleftarrow$ Locality-preserving



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

$$\min_{\mathbf{f}\in\mathbb{R}^n} \frac{1}{2}\sum_{i,j} w_{ij}(f_i - f_j)^2$$

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

$$\min_{\mathbf{f}\in\mathbb{R}^n} \frac{1}{2}\sum_{i,j} w_{ij}(f_i - f_j)^2$$

is not well defined yet. Why?

To make the objective function scaling invariant in $\mathbf{f}$ (and also to get rid of the trivial solution $\mathbf{0}$), we consider adding the following constraint on $\mathbf{f}$:

$$\min_{\mathbf{f}\in\mathbb{R}^n} \frac{1}{2}\sum_{i,j} w_{ij}(f_i - f_j)^2 \qquad \text{subject to} \quad \sum_i f_i^2 = 1.$$

Equivalently, it can be reformulated as an unconstrained problem:

$$\min_{\mathbf{f}\in\mathbb{R}^n:\|\mathbf{f}\|=1} \frac{1}{2}\sum_{i,j} w_{ij}(f_i - f_j)^2, \quad \text{or} \quad \min_{\mathbf{f}\neq\mathbf{0}\in\mathbb{R}^n} \frac{\frac{1}{2}\sum_{i,j} w_{ij}(f_i - f_j)^2}{\sum_i f_i^2}$$

Remark. Later we will see a different constraint on $\mathbf{f}$ for dealing with the zero solution. Also, there is another trivial solution to be identified and removed.

Spectral graph theory (a little bit)

To solve the problem formulated on the preceding slide, we need to present some graph terminology and theory.

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 $w_{ij} > 0$).

Two vertices are *adjacent* if they are connected by an edge (i.e., $w_{ij} > 0$).

An edge is *incident* on a vertex if the vertex is an endpoint of the edge.

When the binary weighting method is used (i.e., all positive weights are equal to 1), the weight matrix is also referred to as the *adjacency matrix*.

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 0.2. 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.

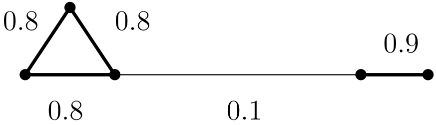
A path in the graph is a sequence of vertices and edges in between such that no vertex or edge can repeat.

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 0.3. 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).

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

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \in \mathbb{R}^{n\times n}, \qquad \text{where} \quad \mathbf{D} = \mathrm{diag}(\mathbf{W}\mathbf{1}).$$

Example 0.4. 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., $\mathbf{L1} = \mathbf{0}$. 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

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

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

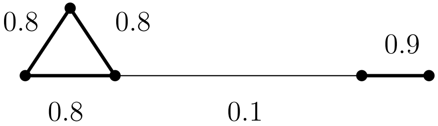
- The algebraic (and also geometric) multiplicity of the eigenvalue 0 equals the number of connected components of the graph.

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). $\square$

Example 0.5. 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 0.6. Consider the modified graph (which has two connected components)

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

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

... for embedding the nodes of a graph $G = (V, E, \mathbf{W})$ into a line:

$$\min_{\mathbf{f}\neq\mathbf{0}\in\mathbb{R}^n} \frac{\frac{1}{2}\sum_{i,j} w_{ij}(f_i - f_j)^2}{\sum_i f_i^2}.$$

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

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

Again, we have encountered a Rayleigh quotient problem!

Clearly, a minimizer of the Rayleigh quotient is an eigenvector of the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$ corresponding to the smallest eigenvalue $\lambda_1 = 0$:

$$\mathbf{f}^* = \mathbf{v}_1 = \mathbf{1}.$$

However, this is another trivial solution which puts all nodes of the graph at the same point of a line.

To eliminate this trivial solution, we add an additional constraint on $\mathbf{f}$:

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

which forces $\mathbf{f}$ to be perpendicular to the vector $\mathbf{1}$. This can also be interpreted as removing the translational invariance in $\mathbf{f}$.

Incorporating the new constraint $\mathbf{f}^T\mathbf{1} = 0$ leads to the following problem:

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

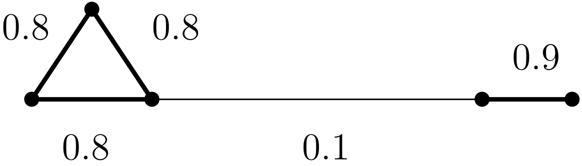
The minimizer of this new problem is given by the second smallest eigenvector of $\mathbf{L}$:

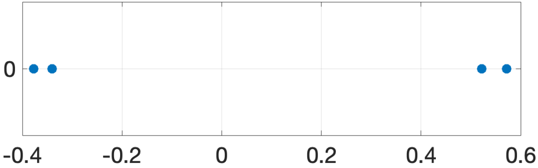
$$\mathbf{f}^{**} = \mathbf{v}_2,$$

and the minimum value of the Rayleigh quotient is λ_2.

If the graph is connected, the algebraic (and geometric) multiplicity of the eigenvalue 0 is one. Consequently, we must have $\lambda_2 > 0$. This shows that $\mathbf{v}_2$ will lead to a nontrivial embedding of the graph.

Example 0.7. For the graph below (which is connected), the second smallest eigenvector ($\lambda_2 = 0.0788$) 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 next.

Remark. Without the constraint $\mathbf{f}^T\mathbf{D}\mathbf{1} = 0$, the solution of the generalized Rayleigh quotient problem is given by the smallest eigenvector $\mathbf{v}_1$ of $\mathbf{D}^{-1}\mathbf{L}$:

$$\mathbf{D}^{-1}\mathbf{L}\mathbf{v}_1 = \lambda_1\mathbf{v}_1 \quad \Longleftrightarrow \quad \mathbf{L}\mathbf{v}_1 = \lambda_1\mathbf{D}\mathbf{v}_1.$$

And we must have $\mathbf{v}_1 = \mathbf{1}$ (and $\lambda_1 = 0$) because

$$\frac{\mathbf{f}^T\mathbf{L}\mathbf{f}}{\mathbf{f}^T\mathbf{D}\mathbf{f}} \ge 0, \quad \frac{\mathbf{1}^T\mathbf{L}\mathbf{1}}{\mathbf{1}^T\mathbf{D}\mathbf{1}} = 0 \quad (\mathbf{L}\mathbf{1} = \mathbf{0})$$

and

$$(\mathbf{D}^{-1}\mathbf{L})\mathbf{1} = \mathbf{D}^{-1}(\mathbf{L}\mathbf{1}) = \mathbf{D}^{-1}\mathbf{0} = \mathbf{0} = 0 \cdot \mathbf{1}$$

Thus, the corresponding problem only has a trivial solution.

To better understand the situation, we need to study the normalized graph Laplacians.

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

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

Remark. $\mathbf{L}, \mathbf{L}_{\mathrm{rw}}, \mathbf{L}_{\mathrm{sym}}$ are all square matrices of the same size ($n \times n$ with $n = |V|$). Which of them are symmetric (and PSD)?

Theorem 0.2. Properties of the normalized graph Laplacians:

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

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

  This implies that both matrices have the same eigenvalues

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

  Additionally, it can be shown that the multiplicity of the zero eigenvalue is also equal to the number of connected components in the graph.

- A vector $\mathbf{v}$ is an eigenvector of $\mathbf{L}_{\mathrm{rw}}$ if and only if the vector $\mathbf{D}^{1/2}\mathbf{v}$ is an eigenvector of $\mathbf{L}_{\mathrm{sym}}$:

$$\underbrace{\mathbf{D}^{-1}\mathbf{L}}_{\mathbf{L}_{\mathrm{rw}}}\mathbf{v} = \lambda\mathbf{v} \quad \Longleftrightarrow \quad \underbrace{\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}}_{\mathbf{L}_{\mathrm{sym}}}\mathbf{D}^{1/2}\mathbf{v} = \lambda\mathbf{D}^{1/2}\mathbf{v}.$$

  In particular, for the eigenvalue 0, the associated eigenvectors for $\mathbf{L}_{\mathrm{rw}}$ and $\mathbf{L}_{\mathrm{sym}}$ are $\mathbf{1}$ and $\mathbf{D}^{1/2}\mathbf{1}$, respectively.

Now consider the original and full Laplacian Eigenmaps problem again:

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

We show that the minimizer is given by the second smallest eigenvector of $\mathbf{L}_{\mathrm{rw}} = \mathbf{D}^{-1}\mathbf{L}$ (when the graph is connected):

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

Through a change of variables $\tilde{\mathbf{f}} = \mathbf{D}^{1/2}\mathbf{y}$ such that

$$\mathbf{f}^T \mathbf{D} \mathbf{1} = \mathbf{y}^T \mathbf{D}^{1/2} \mathbf{D}^{1/2} \mathbf{1} = \tilde{\mathbf{f}}^T \mathbf{D}^{1/2} \mathbf{1}$$
$$\mathbf{f}^T \mathbf{D} \mathbf{f} = \mathbf{f}^T \mathbf{D}^{1/2} \mathbf{D}^{1/2} \mathbf{f} = \tilde{\mathbf{f}}^T \tilde{\mathbf{f}}$$
$$\mathbf{f}^T \mathbf{L} \mathbf{f} = \mathbf{f}^T \mathbf{D}^{1/2} \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-1/2} \mathbf{D}^{1/2} \mathbf{f} = \tilde{\mathbf{f}}^T \mathbf{L}_{\mathrm{sym}} \tilde{\mathbf{f}}$$

we obtain the following equivalent problem:

$$\min_{\substack{\tilde{\mathbf{f}}\neq\mathbf{0}\in\mathbb{R}^n\\ \tilde{\mathbf{f}}^T(\mathbf{D}^{1/2}\mathbf{1})=0}} \frac{\tilde{\mathbf{f}}^T\mathbf{L}_{\text{sym}}\tilde{\mathbf{f}}}{\tilde{\mathbf{f}}^T\tilde{\mathbf{f}}}.$$

The optimal $\tilde{\mathbf{f}}$ is given by the second smallest eigenvector of $\mathbf{L}_{\text{sym}}$ (since $\mathbf{D}^{1/2}\mathbf{1}$ is the eigenvector corresponding to the smallest eigenvalue $\lambda_1 = 0$):

$$\mathbf{L}_{\text{sym}}\tilde{\mathbf{f}} = \lambda_2\tilde{\mathbf{f}}$$

In the original variable $\mathbf{f}$, this becomes

$$\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}\,\mathbf{D}^{1/2}\mathbf{f} = \lambda_2\,\mathbf{D}^{1/2}\mathbf{f} \;\longrightarrow \mathbf{D}^{-1}\mathbf{L}\mathbf{f} = \lambda_2\,\mathbf{f}$$

Thus, the optimal $\mathbf{f}$ is given by the second smallest eigenvector of $\mathbf{L}_{\text{rw}}$.

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

the normalized graph Laplacian $\mathbf{L}_{\text{rw}}$ is

$$\mathbf{D}^{-1}\mathbf{L} = \begin{pmatrix} 1 & -0.5 & -0.5 & & \\ -0.5 & 1 & -0.5 & & \\ -0.4706 & -0.4706 & 1 & -0.0588 & 0 \\ & & -0.1 & 1 & -0.9 \\ & & & -1 & 1 \end{pmatrix}$$

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. Compare with the unnormalized graph Laplacian $\mathbf{L}$:

$$\lambda_2 = 0.0788, \quad \mathbf{v}_2 = (.3771, .3771, .3400, -.5221, -.5722).$$

Which graph Laplacian we should use in general embedding graph data? The two 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 Laplacian should be preferred; we will see their difference more clearly in the context of clustering.

Embedding graph data to 2D or higher

Naturally, 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 $\mathbf{L}_{\mathrm{rw}} = \mathbf{D}^{-1}\mathbf{L}$:

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

to form the embedding matrix

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

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

Alternatively, we could directly formulate the following minimization problem over a k-dimensional embedding matrix $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^T \in \mathbb{R}^{n\times k}$:

$$\min_{\substack{\mathbf{Y}^T\mathbf{D}\mathbf{Y}=\mathbf{I}\\ \mathbf{Y}^T\mathbf{D}\mathbf{1}=\mathbf{0}}} \frac{1}{2}\sum_{i,j} w_{ij}\|\mathbf{y}_i - \mathbf{y}_j\|^2$$

which can be rewritten as

$$\min_{\substack{\mathbf{Y}^T\mathbf{D}\mathbf{Y}=\mathbf{I}\\ \mathbf{Y}^T\mathbf{D}\mathbf{1}=\mathbf{0}}} \mathrm{trace}(\mathbf{Y}^T\mathbf{L}\mathbf{Y})$$

It turns out that the solution is given by the same eigenvectors of $\mathbf{L}_{\mathrm{rw}}$:

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

The Laplacian Eigenmaps algorithm

Input: $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, embedding dimension $k \geq 1$, neighborhood graph method (ϵ-ball or kNN), weighting method (binary or Gaussian)

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

Steps:

1. Construct a neighborhood graph G from the given data
2. Set the edge weights using the specified method to form the weight matrix $\mathbf{W}$.

3. Compute the normalized graph Laplacian

$$\mathbf{L}_{\mathrm{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{W}) = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W},$$

   where $\mathbf{D} = \mathrm{diag}(\mathbf{W1})$.

4. Find the eigenvectors of $\mathbf{L}_{\mathrm{rw}}$ corresponding to the second to $(k+1)$st smallest eigenvalues

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

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

Implementation

Refer to the Matlab Toolbox for Dimensionality Reduction developed by Laurens van der Maaten, which can be downloaded from the url: `http://lvdmaaten.github.io/drtoolbox/`

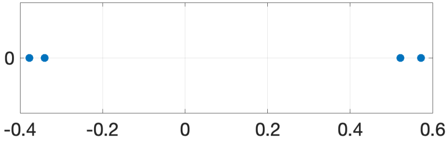
It contains Matlab implementations of 34 techniques for dimensionality reduction and metric learning, including Laplacian Eigenmaps (LE).

Connections to spectral clustering

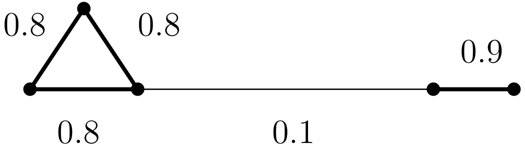
Laplacian Eigenmaps is originally proposed as a nonlinear dimension reduction method by preserving local geometry of the given data.

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:

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



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 was derived from a clustering perspective.



Remark. Clustering the original data is equivalent to finding a partition of the associated graph: $V = A_1 \cup \cdots \cup A_c$ where $A_i \cap A_j = \emptyset,\ i \neq j$.

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

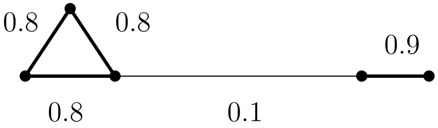
$$\mathbf{1}_A = (a_1, \ldots, a_n)^T, \quad a_i = 1 \text{ (if } i \in A) \text{ and } a_i = 0 \text{ (if } i \in \bar{A}).$$

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 $V = A \cup B$ with $\mathrm{Vol}(A) = a, \mathrm{Vol}(B) = b$.

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 $\mathbf{f}$ is uniquely determined by the bipartition. On the other hand, if $\mathbf{f}$ is given first, then $\mathbf{A}, \mathbf{B}$ can be easily and uniquely identified.

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

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

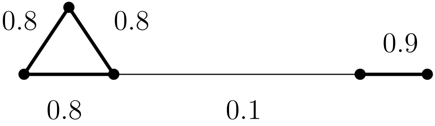
This problem is still discrete in nature. To find an approximate solution, we eliminate the condition $\mathbf{f} \in \{\alpha, -\beta\}^n$ to solve the relaxed problem

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

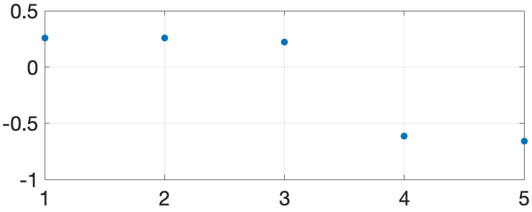
This is exactly the same generalized Rayleigh quotient problem we obtained for Laplacian Eigenmaps, with the same minimizer $\mathbf{f}^* = \mathbf{v}_2$ (the second smallest eigenvector of $\mathbf{L}_{\mathrm{rw}} = \mathbf{D}^{-1}\mathbf{L}$).

New interpretation: $\mathbf{v}_2$ 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 $\mathbf{L}_{\text{rw}}$ is $\mathbf{v}_2 = (-0.2594, -0.2594, -0.2235, 0.6152, 0.6610)$.



Matlab demonstration

- Two Gaussians
- Two circles

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.

Further learning on spectral clustering

- "Normalized Cuts and Image Segmentation", Jianbo Shi and Jitendra Malik, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pages 888–905, August 2000. URL: `https://people.eecs.berkeley.edu/~malik/papers/SM-ncut.pdf`
- "A Tutorial on Spectral Clustering", Ulrike von Luxburg, *Statistics and Computing*, Volume 17, pages 395–416 (2007). URL: `https://arxiv.org/pdf/0711.0189.pdf`