Laplacian Eigenmaps

National Tsing Hua University

September 27, 2024

Contents

Laplacian Eigenmaps

You

September 27, 2024

1 Introduction

Sometimes, we often face to data lying in a very high-dimensional space made us hard to analyze. The solution to understand the data is dimensionality reduction like principal components analysis (PCA) or multidimensional scaling, which perform a linear mapping while Laplacian eigenmaps that generate non-linear maps.

The idea of Laplacian eigenmaps is first to construct a graph contained the neighborhood information (weights and connection) of the data set. Then we can find a low-dimensional representation of the data set which preserves **"local"** neighborhood information in a certain sense. And such algorithm is as simple as described.

This locality-preserving character makes it to be stable when the data has some outliers or noise. Also, we found this algorithm can be also viewed as some kind of clustering.

2 Algorithm

(a) Construct a Graph:
The first step is to construct a graph G where each node represents a data point.
Edges between nodes are typically created by k-NN or ϵ-neighborhood.
k-NN: select the K closest data.
ϵ-neighborhood: select the data in the ball with radius ϵ.
Edge weights w_{ij} can be assigned using the heat kernel $w_{ij} = \exp -\frac{||x_i - x_j||^2}{t}$ where x_i and x_j are two data points, and t is a scale parameter.

(b) Compute the Laplacian Matrix:
Compute the degree matrix D, which is a diagonal matrix where each diagonal element D_{ii} is the sum of the weights of all edges connected to

vertex i.
Compute the Laplacian matrix $L = D - W$.

(c) Eigenvalue Decomposition:
Solve the generalized eigenvalue problem $Lx = \lambda Dx$ to obtain the eigenvectors corresponding to the smallest non-zero eigenvalues.
The eigenvectors provide a new set of coordinates for the data points in the reduced dimensional space. By selecting the eigenvectors associated with the smallest non-zero eigenvalues, one ensures that the low-dimensional representation preserves local neighborhood structures.
$x_i \xrightarrow{Embedding} (f_2(i), f_3(i), \ldots, f_{d+1}(i))$, where $f_2, f_3, \ldots, f_{d+1}$ is the eigenvectors corresponding to the first d smallest non-zero eigenvalues.

Algorithm 1 Laplacian Eigenmap

```
function LAPLACIAN EIGENMAP(X,k(or $\epsilon$),t,d)
            ▷ X:input data, k: number of neighbors, t: constant of Heat Kernel
    $G \leftarrow GraphConstruction(X, k(or\ \epsilon))$
    $W \leftarrow Heatkernel(G, t)$                                  ▷ W:Weight matrix
    $D \leftarrow diag(\sum_i W_{1i}, \sum_i W_{2i} \ldots, \sum_i W_{ni})$       ▷ D:degree matrix
    $L \leftarrow D - W$
    eigenvalues, eigenvectors $\leftarrow Solveeigenvalue(D^{-1}L)$          ▷ $Lf = \lambda Df$
    return $f_2, \ldots f_{d+1}$
end function
```

$Heatkernel(x, y, t) = e^{-\frac{||x-y||^2}{t}}$
$D^{-1}L$ can be replaced by $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$
$\lambda_2, \ldots, \lambda_{d+1}$ is first d smallest eigenvector larger than 0 corresponding to eigenvectors $f_2, \ldots, f_{d+1}$.

3 Justification

The goal is to find a mapping maps the original data $(x_1, x_2, ..., x_n)$ to $(y_1, y_2, ..., y_n)$. We hope such map satisfying $min \sum_{i,j}(y_i^{(k)} - y_j^{(k)})^2 W_{ij}$ for all k, which makes if x_i and x_j are close, then y_i and y_j are close as well, while x_i and x_j are far, it would be a heavy penalty on W_{ij} (close to 0). Before we explained how we do this optimization, we first introduce the Laplacian matrix applied in building our graph. [1]

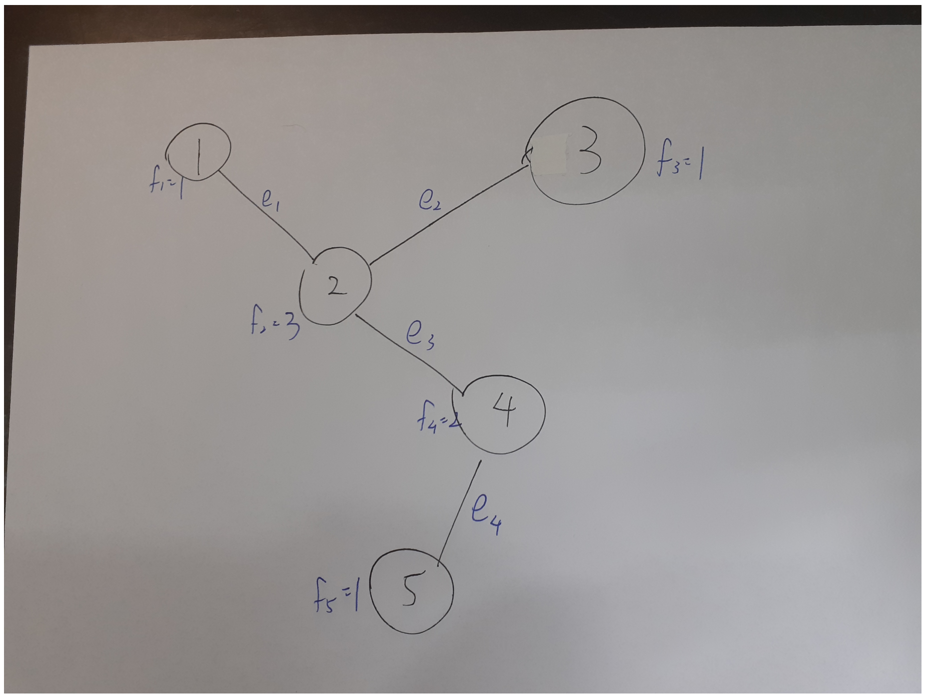

Figure 1: graph example

3.1 graph Laplacian (Laplacian matrix)

For the Euclidean space, the Laplace operator is the divergence of the gradient of a function. That's $\nabla^2 \mathbf{f} = \nabla \cdot \nabla \mathbf{f} = div(grad(f))$. For our data which is always discrete, we want to imitate the Laplace operator on this. what's the use of this? Let's take an example.

Consider the graph:

Define f to perform the value of each nodes.

$$f = \begin{bmatrix} 1 \\ 3 \\ 1 \\ 2 \\ 1 \end{bmatrix}$$

and k with each column representing a edge connecting two nodes. -1 and 1 means the direction of flow-out and flow-in individual.

$$k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

For gradient operator, giving the derivative of the function along each direction. Naturally, we can define the gradient to be $f(i) - f(j)$ if i, j is connected. Write

$$\nabla \mathbf{f} = k^T f = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} f(1) - f(2) \\ f(2) - f(3) \\ f(2) - f(4) \\ f(4) - f(5) \end{bmatrix}$$

The i-row means the flow edge i.
For example: $-2 = 1 - 3 = f(1) - f(2)$. The flow on edge 1.

For divergence, divergence at a point gives the net outward flux of a vector field.Ex. if we want to calculate the divergence at node 2. It could be $(-2) * (-1) + 2 * 1 + 1 * 1 = 5$.

$$k \nabla \mathbf{f} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} -2 \\ 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 5 \\ -2 \\ 0 \\ 1 \end{bmatrix}$$

Conclude that the Laplacian $Lf = kk^T f$

$$L = kk^T = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} = D - W$$

Observe that we can define W, D directly by

$$W_{i,j} = \begin{cases} 1 & i, j \textit{ are connected} \\ 0 & \textit{otherwise} \end{cases}$$

$$D_{i,j} = \begin{cases} \sum_{k=1}^{n} W_{i,k} & \textit{for } i = j \\ 0 & \textit{otherwise} \end{cases}$$

Hence we got the equation:

$$\|k^T f\|^2 = (f^T k)(k^T f) = f^T L f$$

3.2 Embedding using Graph Laplacian

First, we need to prove that

$$\min_{\mathbf{f}} \sum_{i,j} (f_i - f_j)^2 W_{ij} \equiv \min_{\mathbf{f}} \mathbf{f}^T L \mathbf{f}$$

Proof:

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

This proof also demonstrates that L is positive semi-definite and the algebraic multiplicity of the eigenvalue 0 equals the number of connected components of the graph.

However, the problem as stated is not well-defined. Therefore, we impose the constraint $\|\mathbf{f}\| = 1$, making the problem equivalent to

$$\min_{\mathbf{f} \neq 0} \frac{\mathbf{f}^T L \mathbf{f}}{\mathbf{f}^T \mathbf{f}}$$

The trivial solution $\mathbf{f} = \mathbf{1}$, which places all nodes of the graph at the same point on a line, is not desirable. To eliminate this solution, we introduce an additional constraint: $\mathbf{f}^T \mathbf{1} = 0$ (i.e., $\mathbf{f} \perp \mathbf{1}$).

To further refine the problem by removing the scaling factor in $\mathbf{f}$, we consider:

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

This is equivalent to solving the generalized eigenvalue problem $L\mathbf{f} = \lambda D \mathbf{f}$ to find the eigenvectors corresponding to the smallest non-zero eigenvalues.

3.3 Laplace Beltrami operator

As we have known that how we connect the graph Laplacian to our mapping. Now we want to give the outline proof that the mapping can ensure $f(x), f(y)$ being close if x, y is close enough on the graph we made.

Since the method of making our graph is to describe the relation between one and the others neat it. The graph can also be linked as the manifold which locally resembles Euclidean space near each point. Thus the graph Laplacian is also analogous to the Laplace operator on manifold called Laplace Beltrami operator.

Then we can use the property of manifold to prove. The whole proof is nothing but using Taylor's approximation to estimate the distance. The key made we can do such estimation is geodesic curve. Geodesic curve means the path connects two points with the shortest distance on manifold. Thus we can parameterize the geodesic to get the following result:

$$|f(x_i) - f(x_j)| \leq \|\nabla f(x)\| \|x_i - x_j\| + o(\|x_i - x_j\|)$$

It's nothing but the mean value theorem for vector-valued functions. If x, y is close enough then $|f(x_i) - f(x_j)|$ can be controlled. Notice that this inequality only guarantee the value for two point being close. This also give an explanation why our optimization

$$\min_{\mathbf{f}} \sum_{i,j} (f_i - f_j)^2 W_{ij}$$

would like to multiply the weight W_{ij}. That's to control the value for x, y being far away

4 Connections to Clustering

Laplacian Eigenmap and Spectral Clustering both use the graph Laplacian matrix to analyze high-dimensional data; the former focuses on reducing dimensions by preserving local neighborhood structures for visualization and further analysis, while the latter clusters data by transforming it into a space where traditional clustering algorithms can more effectively discern and separate groups.

Remark: Clustering the original data is equivalent to finding a partition of the associated graph:$V = A_1 \cup A_2 \cup \cdots \cup A_c$where $A_i \cap A_j = \emptyset$ for $i \neq j$
Definition:
let A,B be subsets of V
(1) $Vol(A) = \sum_{i \in A} d_i$
(2) $W(A, B) = \sum_{i \in A, j \in B} w_{ij}$

(3) if $B = \bar{A}$, $W(A, B)$ is call a cut $Cut(A, B) = W(A, B) = \sum_{i\in A, j\in B} w_{ij}$

To find the "optimal" bipartition of a graph $V = A \cup B$ with $B = \bar{A}$, it is proposed to minimize the following normalized cut:

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

To solve the Ncut problem, consider any partition $V = A \in B$ with $Vol(A) = a, Vol(B) = b$

Define $\mathbf{f} = \frac{1}{a}\mathbf{1}_A - \frac{1}{b}\mathbf{1}_B$ with:

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

For this f, we have:

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

Additionally, f satisfies

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

Therefore, we can obtain the following equivalent problem

$$\min_{\substack{A\cup B=V \\ A\cap B=\emptyset}} NCut(A, B) \Longleftrightarrow \min_{\substack{\mathbf{f}\neq 0 \\ \mathbf{f}^T D\mathbf{1}=0}} \frac{\mathbf{f}^T L\mathbf{f}}{\mathbf{f}^T D\mathbf{f}}$$

The minimizer f^* represents an approximate solution to the Ncut problem, providing information about the labels of the data.

we impose the constraint $\|\mathbf{f}\| = 1$

$$\Longrightarrow \min_{\mathbf{f}\neq 0} \frac{\mathbf{f}^T L\mathbf{f}}{\mathbf{f}^T\mathbf{f}}$$

eliminate the trivial solution $\mathbf{f} = \mathbf{1}$, and refine the problem by removing the scaling factor D in $\mathbf{f}$ (Matrix D provides a natural measure on the vertices of the graph. The bigger the value D_{ii} is, the more important is the ith vertex.)

$$\Longrightarrow \min_{\substack{\mathbf{f}\neq 0\\ \mathbf{f}^T D\mathbf{1}=0}} \frac{\mathbf{f}^T L\mathbf{f}}{\mathbf{f}^T D\mathbf{f}}$$

This is equivalent to solving the generalized eigenvalue problem $L\mathbf{f} = \lambda D\mathbf{f}$ to find the eigenvectors corresponding to the smallest non-zero eigenvalues.

The goal is to find a mapping maps the original data $(x_1, x_2, ..., x_n)$ to $(y_1, y_2, ..., y_n)$. We hope such map satisfying $min \sum_{i,j}(y_i - y_j)^2 W_{ij}$, which makes if x_i and x_j are close then W_{ij} is large, there would be a heavy penalty on y_i and y_j, so y_i and y_j need to be close as well, while x_i and x_j are far, the distance of y_i and y_j are not important because W_{ij} is close to 0.

Theorem 4.1. The algebraic multiplicity of the eigenvalue 0 of matrix L is exactly 1 if the graph is connected.

Proof. Recall that $L = D - W$. That's $L_{ii} + \sum_{\substack{j=1\\ j\neq i}}^{n} L_{ij} = 0$ and $L\mathbf{1} = \mathbf{0}$

Take the eigenvector $v = (v_1, v_2, ..., v_n)^T$, corresponding to the eigenvalue 0.

Scaling v s.t. $v_k = 1$ and $|v_i| \leq v_k = 1$ for some k and $i \in \{1, ..., n\}$.

Since v is an eigenvector. In particular, consider the k-th row of L.

$$\begin{bmatrix} L_{k1} & \cdots & L_{kn} \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = 0$$

$$\Rightarrow 0 = L_{kk} * v_k + \sum_{\substack{j=1\\ j\neq k}}^{n} L_{kj} * v_j \leq L_{kk} + \sum_{\substack{j=1\\ j\neq k}}^{n} L_{kj} = 0$$

So the equality would hold. Implying all $v_i = 1$ since $sgn(L_{kk}) = -sgn(L_{kj})$ for $j \neq k$

Thus all eigenvectors v are parallel to $\mathbf{1}$. The multiplicity of the eigenvalue 0 of matrix L is exactly 1.

$\square$

References

[1] Muni Sreenivas Pydi. What's the intuition behind a laplacian matrix? https://www.quora.com/Whats-the-intuition-behind-a-Laplacian-matrix-Im-not-so-much-interested-in-mathematical-
answer/Muni-Sreenivas-Pydi.