Laplacian Eigenmap

Final Report

Group 1

鄭旭晴、林暐晉、林元鴻、馬翌翔、韓曉汀

2024.6.3

Contents

Chapter 1

Introduction

In the fields of artificial intelligence, information retrieval and data mining, it is usually encountered that some intrinsically low dimensional data **lie** in a very high dimensional space. In other words, some high dimensional data are actually **embedded** in a low dimension manifold.

Dimensionality reduction is to find a mapping that can map the high dimension data to a low dimension space while satisfying certain goals. When the goal is to make the first k coordinates' variation as large as possible, we get the well known principal component analysis(PCA).

In this paper, we will talk about another dimensionality reduction method, whose goal is to **preserve locality**. The Laplacian Eigenmap, proposed in Belkin and Niyogi(2001), takes into account the structure of the manifold on which the data reside. It finds nonlinear mappings that best preserve "local neighborhood information" to restore the geometric structure of the manifold in low dimensional space.

Chapter 2

Algorithm

2.1 Main idea

The main idea of Laplacian Eigenmap contains 2 steps.

The first step is to construct an **adjacency graph** $G(V, E)$ from the original data.This graph contains the **local neighborhood information** in the original data, and does not contain information about the absolute values of coordinates.

The second step is to map the graph vertices to the low dimensional space. The mapping should restore the **local neighborhood information** as much as possible.

In the first step, to construct the adjacency graph, we have 2 methods to determine whether 2 vertices are connected, namely k-nn and ϵ-neighborhood.

- k-nn: for each data point, select k nearest neighbors to connect with it.
- ϵ-neighborhood: for each data point, consider its radius ϵ neighboring ball, select data points within the ball to connect with it.

We also have 2 methods to assign edge weights, namely simple connection and kernel method.

- simple connection: for data x_i and x_j, w_{ij} is 1 if they are connected, else 0.

- kernel method: for data x_i and x_j, edge weights w_{ij} can be assigned using the heat kernel $w_{ij} = \exp\{-\frac{||x_i - x_j|^2}{t}\}$ where t is a scale parameter.

2.2 Objective function

In the second step, we attempt to restore the **local neighborhood information** by solving the following optimization problem.

$$\arg\min_f \sum_{i,j} (f(x_i) - f(x_j))^2 W_{ij}$$

Here, $x_i, x_j, i, j = 1, ...n$ denote original data. f denotes the mapping in the second step. W_{ij} denotes the weight of the edge connecting v_i and v_j, the corresponding vertices in G of the 2 data points.

When x_i is close to x_j, W_{ij}, the weight of the edge connecting them in G, is large. We do not want the 2 embedded points to be far away from each other. Therefore, the objective function **penalize the square difference with large weight** W_{ij}. When x_i is far away from x_j, W_{ij} is small. In this case, we do not care much about the 2 embedded points' distance. Since the weight W_{ij} is small, the square distance can be either small or large.

2.3 From summation to matrix computation

In this section, we will show that the objective function can be rewritten in a matrix computation form.

$$\sum_{i,j} (f(x_i) - f(x_j))^2 W_{ij} = 2\mathbf{f}^T L\mathbf{f}, \quad L = D - W$$

Here, W is the *weight matrix*. W_{ij} is the weight of the edge connecting v_i and v_j. D is the *degree matrix*. It is a diagonal matrix, and its diagonal elements are the row summation of W. You can think it as the total degree of a vertex in the adjacency graph

G.

Proof.

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

⋈

L is called Laplacian matrix. We will demonstrate its physical meaning in the third part. This proof also implies that L is positive semi-definite.

2.4 Scaling constraint

We introduce 2 kinds of scaling constraints to make the problem well defined.

$$\min_{\mathbf{f}^T\mathbf{f}=1} \mathbf{f}^T L\mathbf{f} \qquad \text{or} \qquad \min_{\mathbf{f}\neq 0} \frac{\mathbf{f}^T L\mathbf{f}}{\mathbf{f}^T\mathbf{f}}$$

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

The $f^T Df$ version is to assign more weights to those vertices with larger total degree. We take this version in the following discussion. Using Lagrange multiplier method, the restricted optimization problem could be rewritten as a generalized eigenvalue problem.

$$F = f^T Lf - \lambda(f^T Df - 1)$$

$$\begin{cases} \frac{\partial}{\partial f}F = 2Lf - 2\lambda Df = 0 \\ \frac{\partial}{\partial \lambda}F = f^T Df - 1 = 0 \\ \lambda > 0 \end{cases}$$

$$\Longrightarrow L\mathbf{f} = \lambda D\mathbf{f}$$

Each eigenvector in $\{f_i\}_{i=1}^n$ is a mapping, from original high dimensional space to the embedded low dimensional space. By selecting the eigenvectors associated with the smallest eigenvalues, it can be ensured that the low-dimensional representation preserves most local neighborhood structures.

2.5 The problem of 0 eigenvalue

Assume the graph G is fully connected. It is easy to confirm that there is a trivial solution to the generalized eigenvalue problem, $\mathbf{f}_0 = \frac{1}{\sqrt{n}}\mathbf{1}$. In fact, $\mathbf{f}_0$ is the eigenvector of matrix L corresponding to 0 eigenvalue. It could be understood as "mapping all vertices to the same point". Of course, this mapping can best preserve locality (All pairs of embedded vertices are literally "close" to each other, since they are in the same place!). However, this mapping hardly contains any information. Therefore, we choose to discard this mapping by adding a new constraint.

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

Then, we can get the following low dimensional representation, or embedded coordinates in low dimensional space.

$$x_i \overset{Embedding}{\rightarrow} \begin{pmatrix} f_2(i) \\ f_3(i) \\ \dots \\ f_{d+1}(i) \end{pmatrix}$$

where $f_2, f_3, \dots f_{d+1}$ is the eigenvectors corresponding to the first d smallest non-zero

eigenvalues.

As a further discussion, when the adjacency graph G has k connected components, the algebraic multiplicity of 0 eigenvalue is k. These k corresponding eigenvectors (mappings) only have 2 image values. The proof of these 2 results can be found in the appendix.

Therefore, if the graph has k connected components, there will be k eigen vectors corresponding to 0 eigenvalues that can be used for classifying. The left $n-k$ eigenvectors corresponding to non-zero eigenvalues could be used for embedding.

2.6 Pseudo code

Algorithm 1 Laplacian Eigenmap

Input: X:input data, k: number of neighbors (or ϵ: the neighboring ball radius), t: parameter of heat kernel

Output: $f_2, \dots f_{d+1}$

$G \leftarrow GraphConstruction(X, k(or\ \epsilon))$

$W \leftarrow Heatkernel(G, t)$ ▷ W:weight matrix

$D \leftarrow diag(\sum_i W_{1i}, \sum_i W_{2i} \dots, \sum_i W_{ni})$ ▷ D:degree matrix

$L \leftarrow D - W$

eigenvalues, eigenvectors $\leftarrow Solveeigenvalue(D^{-1}L)$ ▷ $Lf = \lambda Df$

return $f_2, \dots f_{d+1}$

▷ eigenvectors corresponding to the first d smallest nonzero eigenvalues

when using simple weight matrix,

$$W_{ij} = \begin{cases} 1 & \text{node i and j are connected} \\ 0 & \text{node i and j are not connected} \end{cases}$$

when using heat kernel,

$$W_{ij} = \begin{cases} e^{-\frac{||x_i - x_j||^2}{t}} & \text{node i and j are connected} \\ 0 & \text{node i and j are not connected} \end{cases}$$

2.7 Step by step demonstration

In this section, we are going to demonstrate the LE algorithm on a small problem step by step to make readers understand the procedures. First, consider 4 points in a 2-dim space in Figure2.1. The problem is to map these points to **a line**(1-dim space) and make close points stay as together as possible.

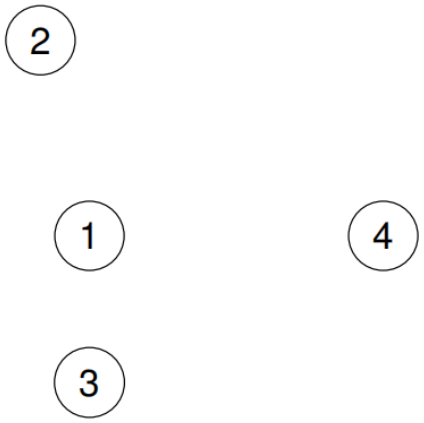


Figure 2.1: 4 points in a 2-dim space.

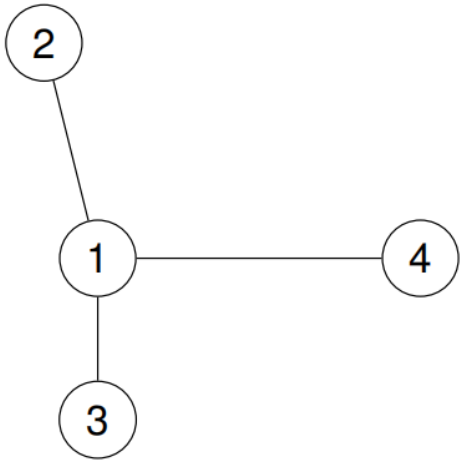


Figure 2.2: 4 points in a 2-dim space.

The first step is to construct an adjacency graph $G(V, E)$. Here, we choose knn as the graph construction method, and use $k = 1$ to select the first nearest neighbour for each vertex. Then, we can get a matrix like Figure 2.2.

The second step is to find a mapping that can best preserve local neighborhood information. We can write the weight matrix W, degree matrix D and Laplacian L as follow.

$$W = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad D = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad L := D - W = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}$$

Having these matrix, we can focus on the generalized eigenvalue problem

$$Lf = \lambda Df$$

or its equivalent problem $D^{-1}Lf = \lambda f$.

The eigenvalues are

$$\lambda_1 = 0, \lambda_2 = 1, \lambda_3 = 1, \lambda_3 = 2$$

pick an eigenvector for $\lambda_2 = 1$ as y,

$$y := f_2 = \begin{pmatrix} 0 \\ -3 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

Then, $y_1 = 0, y_2 = -3, y_3 = 1, y_4 = 2$ are the embedded coordinates of the original data points. And the embedded line is shown in Figure 2.3.

Figure 2.3: The embedded line

Chapter 3

Understanding the algorithm

3.1 The physical meaning of graph Laplacian

In the second step of Laplacian Eigenmap, we try to find a mapping that map the graph vertices to a low dimensional space. Consider the case of mapping vertices to the 1-dimensional space. Then, the mapping can be seen as a scalar field that associates a scalar to every vertices in the graph. Just compare it to **the temperature of nodes in a net**. For each node, it has a corresponding scalar: the temperature at this node.

We can find the gradient in a scalar field. The gradient, being a vector, could be understood as **the flow of scalar in a certain direction**. Just compare it to **the heat flow along the edges in a net**. If the net is made of materials with good ability to conduct heat, and the nodes are not in the same temperature, heat will transfer along the edges, from a vertex to another vertex.

Gradient itself forms a vector field, and we can find the divergence of gradient in this vector field. The divergence, being a scalar, could be understood as **the volume of the outward flux from a source point**. Just compare it to **the total heat flux running out from a certain node**.

3.2 Step by step demonstration of a Laplacian matrix

In this part, we will try to demostrate the concept of Laplacian matrix step by step. Consider a fully connected graph.

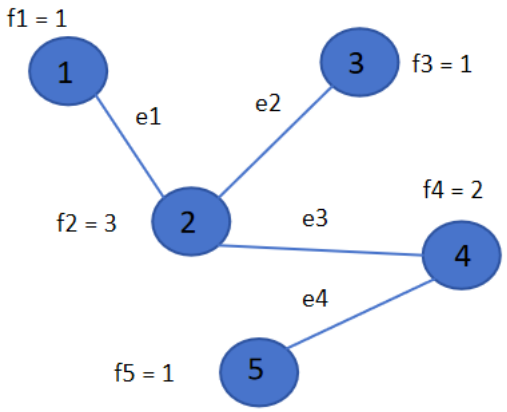


Figure 3.1: A connected adjacency graph and the mapping values of its vertices.

In Figure3.1, each vertex i has its own mapping value f_i. We can use f to denote this graph mapping.

$$f = \begin{pmatrix} 1 & 3 & 1 & 2 & 1 \end{pmatrix}^T$$

Then, we propose an incidence matrix, $k \in \mathbf{R}^{|V| \times |E|}$. Each row of k denotes a vertex in V, and each column of k denotes an edge in E.

$$k_{ij} = \begin{cases} -1 & j\text{-th edge comes from vertex } i \\ 1 & j\text{-th edge run into vertex } i \\ 0 & o.w. \end{cases}$$

and the incidence matrix for graph in Figure3.1 is shown below.

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

Using the incidence matrix, we can calculate the gradient and Laplacian of this mapping. Gradient could be understood as **the flow quantity along a certain edge**. Write

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

The elements of ∇f denote the quantity of flow along each edge in graph G. For example, the first element is $f(1) - f(2) = -2$, that is, the flow comes from $f(2)$ to $f(1)$, and the quantity is -2.

Laplacian can be understood as the **divergence of gradient**. Write

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

The elements of $k\nabla f$ denote the volume of the outward flux from a source node in graph G. For example, the second element is $-\nabla f(1) + \nabla f(2) + \nabla f(3) = 5$, that is, edge 1 comes out from node 2, while edge 2 and edge 3 come in node 2. The total volume of the outward flux from node 2 is 5.

Now, we can define the graph Laplacian with

$$Lf = kk^T f, \quad L = kk^T = D - W$$

This Laplacian matrix could provide the total volume of the outward flux from each node(vertex) in graph G, and reflects **how bumpy the graph is.** A local maximum vertex should have positive divergence, while a local minimum vertex should have negative divergence.

3.3 Review of the objective function

Now, we can better understand the objective function in matrix computation form.

$$\min_{\mathbf{f}^T D\mathbf{f}=1} \mathbf{f}^T L\mathbf{f}$$

Laplacian captures the amount of divergence from a point. If a point is a local minimum/maximum, its Laplacian is positive/negative because its diverging neighbors are upper/lower than it. If the point is on a linear manifold, its Laplacian is zero, because the scalar field of its neighboring points is symmetric: the number of points larger than it is equal to the number of points smaller than it. Therefore, Laplacian shows how *bumpy* the scalar field is. When we try to minimize $\frac{\mathbf{f}^T L\mathbf{f}}{\mathbf{f}^T D\mathbf{f}=1}$, we are actually trying to find the most "unbumpy" scalar field(mapping).

3.4 Comparison with the Laplace-Beltrami operator

We have justified that the graph Laplacian captures the divergence from a source point in the vector field defined by mapping's gradient. In fact, the graph Laplacian is analogous to the Laplace Beltrami operator on manifolds. In this section, we are going to deduce the objective function on a smooth manifold to give readers a deeper understanding about LE method's objective function.

Using Taylor approximation and Cauchy Schwarz inequality, we have the following

inequality.

$$|f(x+\delta x)-f(x)| \approx | < \nabla f(x), \delta > | \leq ||\nabla f|| ||\delta x||$$

That means, **if we can control the norm of the map gradient $||\nabla f||$ small enough, close points will be mapped to close images.** That's what we want.

Therefore, to make sure all the gradient norms are small enough, we try to find

$$\arg \min_{||f||_{L^2(\mathcal{M})}} \int_{\mathcal{M}} ||\nabla f(x)||^2$$

In the objective above, we want to make sure that **on the whole manifold the square norm of mapping gradient should not be too large.**

Consider the Stokes theorem: given a function f and a vector field $\mathbf{X}$, $\int_{\mathcal{M}} < \mathbf{X}, \nabla f >= \int_{\mathcal{M}} div(\mathbf{X}) f$. Using this theorem, we have

$$\int_{\mathcal{M}} ||\nabla f||^2 = \int_{\mathcal{M}} < \nabla f, \nabla f >= \int_{\mathcal{M}} div(\nabla f), f = \int_{\mathcal{M}} \mathcal{L}(f) f$$

Therefore, the objective function on a smooth manifold has the same form with our objective function in the LE algorithm.

$$\int_{\mathcal{M}} ||\nabla f||^2 = \int_{\mathcal{M}} \mathcal{L}(f) f \quad \text{v.s.} \quad ||k^T f||^2 = f^T L f$$

3.5 Comparison with Least N-Cut Clustering

In this section, we are going to briefly explain a clustering method. **Its objective function's form is quite similar with LE's.** After learning about this clustering method, we can better understand the objective function in the LE method.

The task is to divide a group of graph vertices into 2 subgroups. We want the strong connections across sub groups as few as possible. Therefore, we can check the criterion

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

However, the objective function sometimes leads to unbalanced division: it can pick a single point out and achieve minimum cut. Therefore, we adjust the objective function, make it divided by the harmonic mean of the 2 sub groups volume. The adjusted criterion is named as NCut.

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

where

$$Vol(A) = \sum_{i\in A} d_i, Vol(B) = \sum_{i\in B} d_i, d_i = \sum_{j=1}^{n} w_{ij}$$

To find the "optimal" bipartition of a graph $V = A \bigcup B$ with $B = \bar{A}$ is to find the bipartition that minimizes the normalized cut

$$A = \arg\min_{A} NCut(A,B)$$

Now, we are going to show that, **the objective function of this clustering method is a special case of LE's objective function.**

Proof. Define f as

$$\mathbf{f} = \frac{1}{a}1_A - \frac{1}{b}1_B = \begin{cases} \frac{1}{a}, i \in A \\ -\frac{1}{b}, i \in B \end{cases}$$

, Plug this f into the objective function in the Laplacian Eigenmap algorithm, we have

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

$$
\begin{aligned}
f^T D f &= \sum_i f_i^2 d_{ii} \\
&= \sum_{i \in A} \frac{1}{a^2} d_{ii} + \sum_{i \in B} \frac{1}{b^2} d_{ii} \\
&= \mathrm{Vol}(A) \frac{1}{a^2} + \mathrm{Vol}(B) \frac{1}{b^2} \\
&= \frac{1}{a} + \frac{1}{b}
\end{aligned}
$$

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

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

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

The minimizer f^* of the restricted LE problem represents a solution to the Ncut problem, providing information about the labels of the data. $\bowtie$

Chapter 4

Implementation

4.1 Swiss roll data

Brief Introduction

Swiss roll data set is a simple "hello world" data set for testing dimensionality reduction techniques and algorithms. To generate a swiss roll data set, we need to generate a 2-dim data set (t, z) and then map the points to the 3-dim space $(t \cdot cos(t), t \cdot sin(t), t)$. $\{t_i\}_{i=1}^n$ is generated from standard normal distribution, and $\{z_i\}_{i=1}^n$ is generated from uniform distribution $Unif(0, 10)$. The dataset is standardized before testing.

Referring to Figure4.1, it can be seen that most of the points pairs have distances between 1 and 5. When using the $\epsilon - neighborhood$ method to transform the data, the length is advised to be chosen from this range.

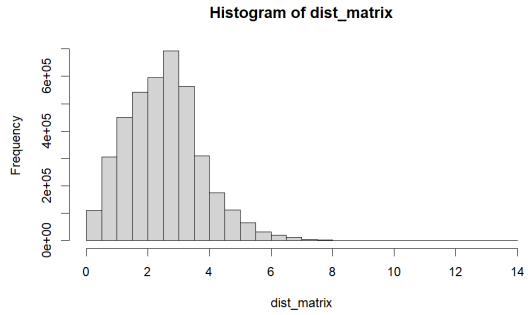


Figure 4.1: Histogram of pairwise distances

Since the 3-dim dataset $\{x_i, y_i, z_i\}_{i=1}^n$ comes from the 2-dim dataset $\{t_i, z_i\}_{i=1}^n$, we will change the color of points according to the value of t to help visualize the analysis results. From Figure4.1 , we can see that there is an outlier on the x-axis with negative value of y and some relatively far away points on the red trajectory.

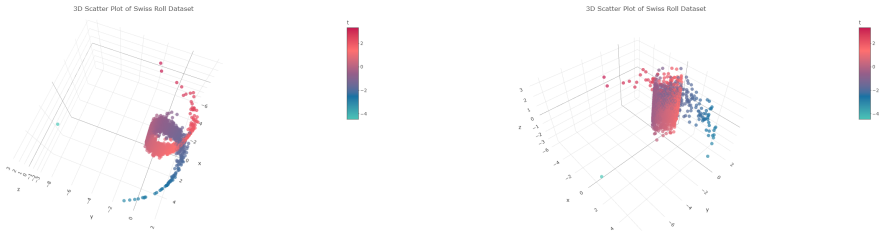


Figure 4.2: 3D Plot

The left plot below is fixed along the z-axis, the middle is fixed along the x-axis, and the right plot is fixed along the y-axis. The noticeable color changes primarily occur along the x and y axes. Therefore, the color changes observed after dimension reduction should be similar to the left plot. We want to capture the smooth color-changing trend.

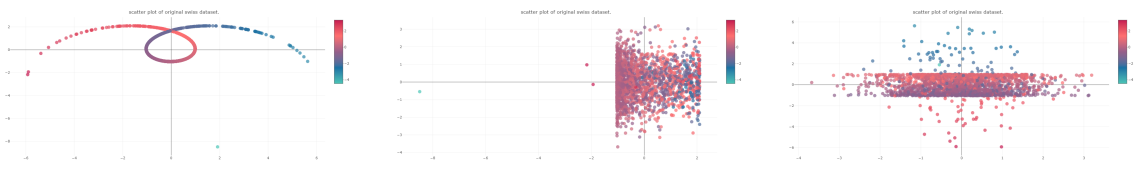


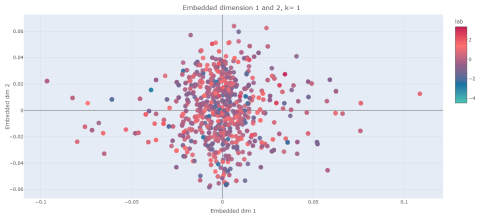
Figure 4.3: 2D Plot

Experimental plan

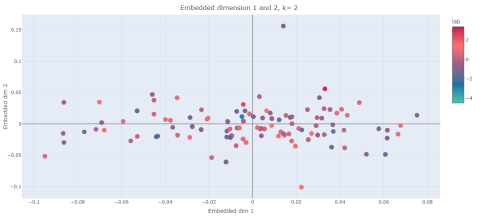
Number	kernel Weight	Graph Construction	Observation
1	False	KNN	Dim1, Dim2
2	True	KNN	Dim1, Dim2
3	False	ENN	Dim1, Dim2
4	True	ENN	Dim1, Dim2

Experiment 1: KNN without kernel weight

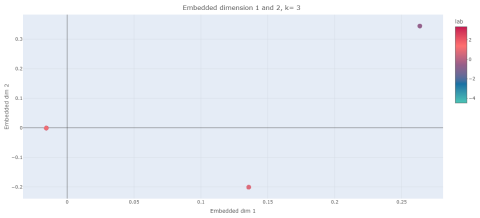
- When k grows from 1 to 3, the embedded points tend to overlap. This is because if two or three points are k-neighbors of each other, mapping them to the same point can perfectly maintain locality. However, when k is larger than or equal to 4, this phenomenon disappears. In this data set, when more than or equal to 4 nearest neighbor is concerned, the mapping better preserves the geometric structure of the manifold while maintaining locality.



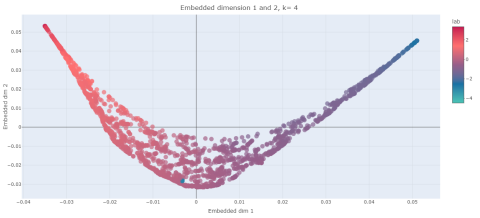
(a) k=1



(b) k=2



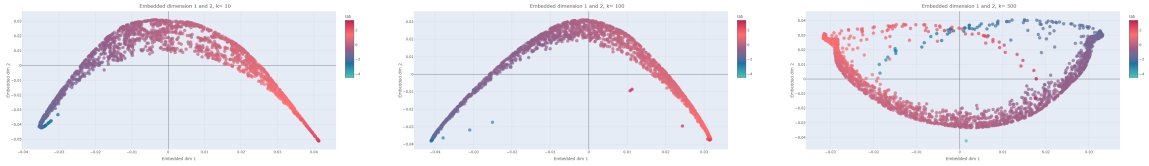
(c) k=3



(d) k=4

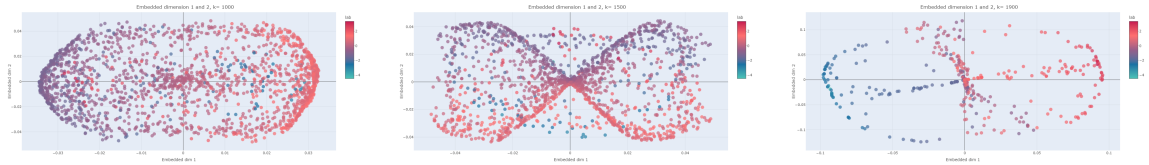
- Since the Swiss data is a fully connected graph with 2000 data points, transformations can still maintain the original color-changing pattern (i.e., nearby points retain their proximity characteristics) with an appropriate k.

- The embedded manifold in 3-dim space seems to "tie a knot". When k approaches one-fourth of the data size, it becomes difficult to capture straightforward regional neighboring relationships. Instead, the mapping begins to capture the tied geometric structure and global neighborhood relations. (Sub figure (g))



(e) k=10 (f) k=100 (g) k=500

- When the value of k approaches half of n, the local trend and the overall trend conflicts and cannot see any explicit patterns. (sub figure h)
- Increasing the value of k, we find that the transformation will only be able to capture the relative position features within the two larger subgraphs at the periphery of data, and excessively large k will eventually result in the transformation losing all information about the central points in the data. (sub figure graph i, j)

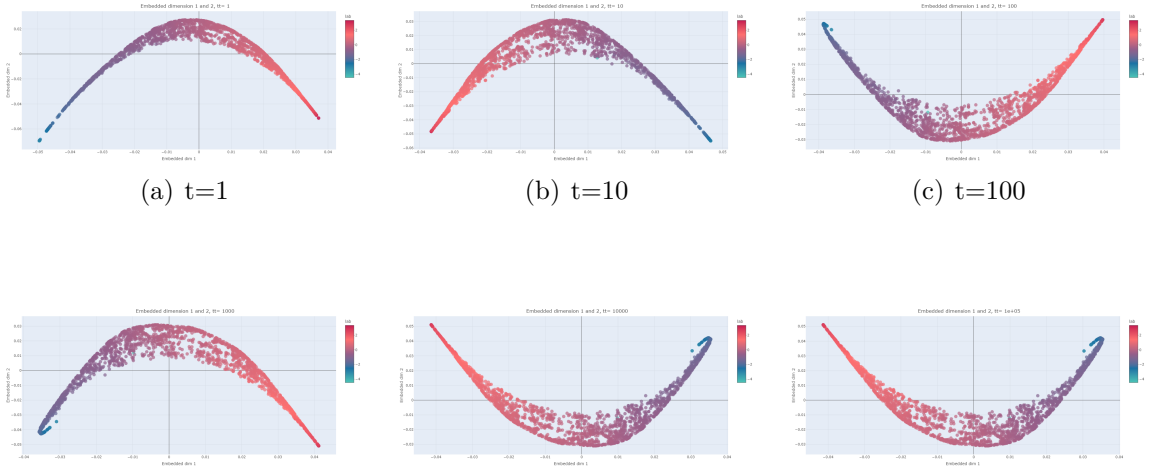


(h) k=1000 (i) k=1500 (j) k=1900

In summary, we may choose $k = 100$ for this problem since it preserve some position information of the peripheral points.

Experiment 2: KNN with different kernel weight

Fixed at k=100, we tried $t = 1, 10, 100, 1000, 10^5, 10^7$, and no significant difference was observed except some change of peripheral points. This suggests that for a fully connected graph, once an effective k is chosen, the weight parameter itself has little impact on the fully connected data but only affects some peripheral points.

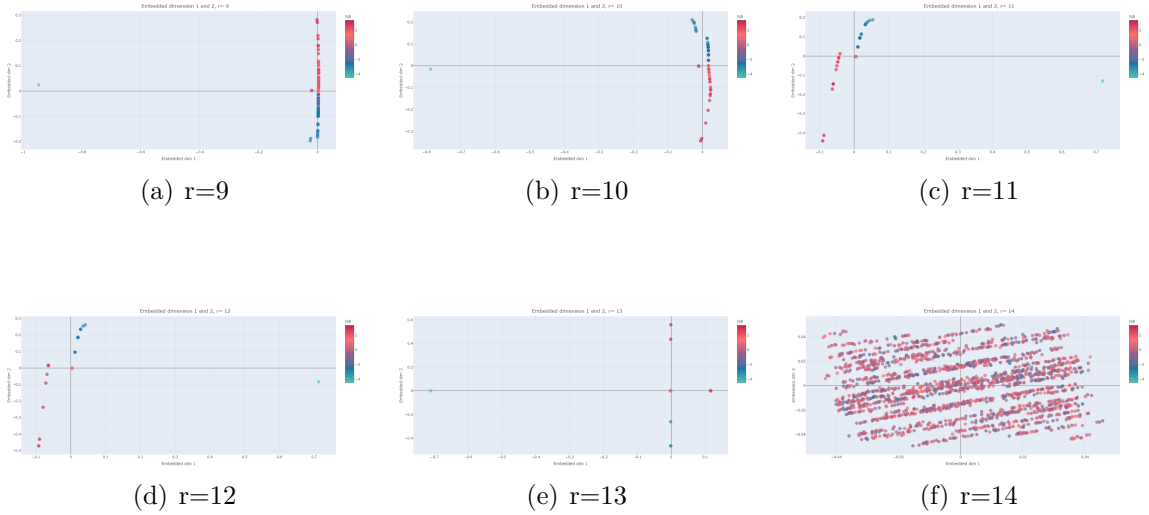


(a) t=1 (b) t=10 (c) t=100

(d) t=1000 (e) $t=10^5$ (f) $t=10^7$

Experiment 3: ENN without kernel weight

Due to the presence of peripheral points in the data, the ϵ-neighborhood method fails when $r < 9$. However, as most point distances are below 5 and none exceed 14, although transformations work for $r > 9$, they still fail to provide a neat embedding that preserving neighboring relationships.

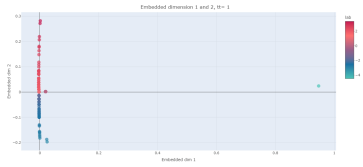


(a) r=9 (b) r=10 (c) r=11

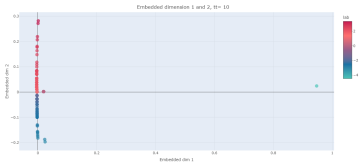
(d) r=12 (e) r=13 (f) r=14

Experiment 4: ENN with different kernel weight

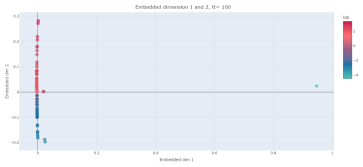
Fixed at r=9, we tried $tt = 1, 10, 100, 1000, 10^5, 10^7$, and no significant difference was observed.



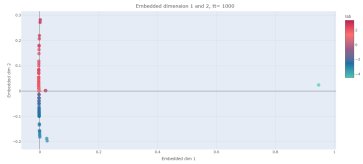
(a) tt=1



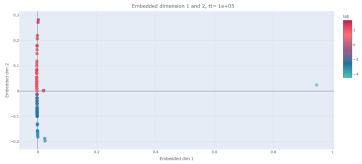
(b) tt=10



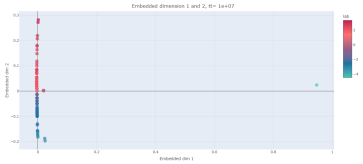
(c) tt=100



(d) tt=1000



(e) tt=10^5



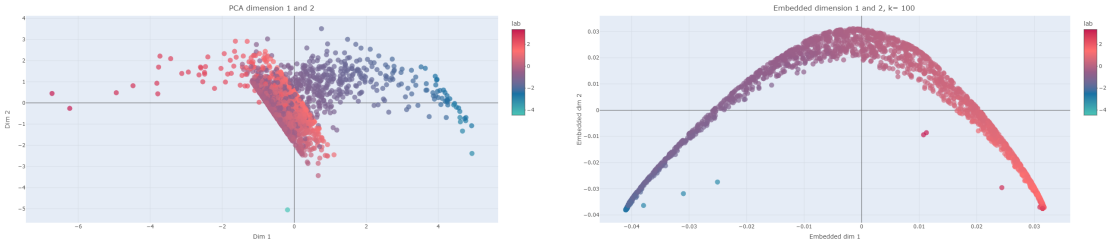
(f) tt=10^7

Conclusion

We may conclude several properties of the Laplacian Eiganmap method:

- The value k in knn method should not be too large if we want to observe the overall neighboring position pattern. But when we want to observe the position pattern of the subsets in the data, a larger k would be better.
- The kernel weight would not affect the transformation of the connected part, but would affect the unconnected peripheral points.
- The ϵ-neighborhood method would be largely affected by the unconnected peripheral points and easily collapsed in the first dimension. If the set is not totally connected, the knn method may be better.

4.1.1 Comparison with PCA



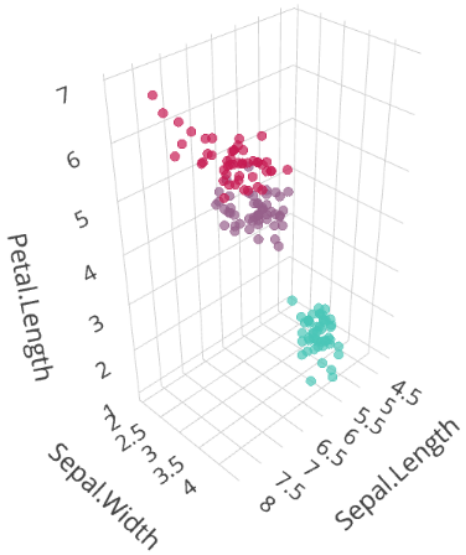
(a) PCA (b) LE

We compare the results from PCA and LE, and can see that when the data spread has a complex geometric structure, PCA methods cannot fully reflect its internal structure and can only capture the components with significant variations. This leads to a plot that fails to clearly indicate specific changes. In contrast, since the LE method focuses on capturing local neighborhood information, it can highlight the distinct neighboring characteristics of the data.

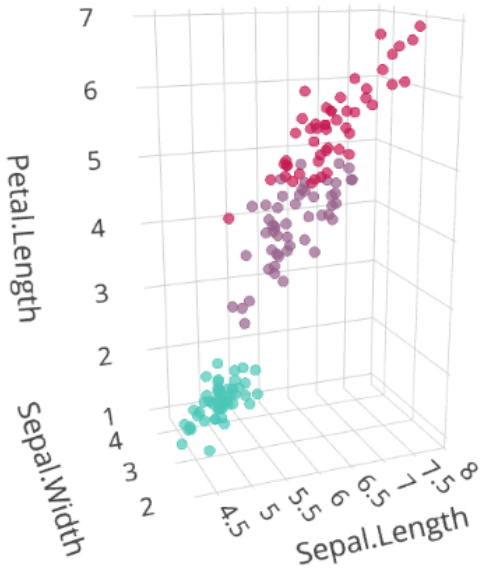
4.2 Iris data

4.2.1 Brief Introduction

Iris data set is a well known data set widely used in machine learning. It contains 150 observations of iris (a kind of flower) from 3 species, namely setosa, virginica and versicolor. There are four features in the dataset, namely sepal length, sepal width, petal length, petal width.



(c) 3d scatter plot of iris data



(d) 3d scatter plot of iris data

4.2.2 Experiment: KNN with different k

See the following figures. When $k = 10$, the setosa species (colored in cyan) could be perfectly separated from the other 2 species. Meanwhile, the versicolor species (colored in pink) and the virginica species (colored in purple) is slightly mixed with each other. Since the setosa species' data points are separated with other points in the original space, when k is as small as 10, the adjacency graph forms into 2 connected components (cyan points are only connected with cyan points). Consequently, as we have proved in Section 2.2.5, the algebraic multiplicity of eigenvalue 0 should be 2. The eigenvectors of 0 eigenvalue could only map data points to 2 points in the low dimensional space. Since we only discard 1 eigenvector, the other is set as the first mapping (coordinates in the first dimension). The separation is successful, but we are not satisfied because no geometric structure can be shown in the first dimension.

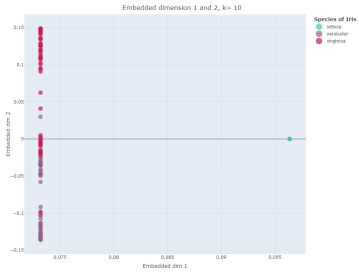
As k grows larger, points are connected with more neighbors. This results in more information to be stored in the adjacency graph to distinguish different points. When

$k = 30$, we can see that the first dimension coordinates are no longer "binary" distributed, which implies some cyan points are connected with purple or pink points in the adjacency graph.

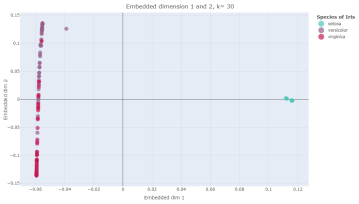
When $k = 75$ (half of the sample size), we observe the best separation effect for 3 species in the 2-dim space. It seems that the embedded points of 3 species could be separated by straight lines.

When k becomes as large as 100, the performance becomes worse. This is because the adjacency graph becomes too dense. With the total sample size equals to 100, a point may be connected with all other points, since it is one of their 100-nearest neighbors. This kind of "super connected" points can not be distinguish from each other since they all share the same neighboring relations in the adjacency graph.

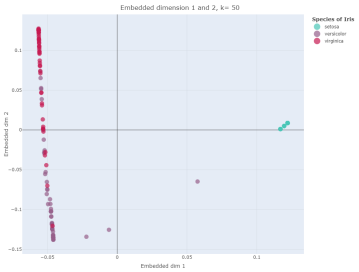
When $k = 150$, all data pairs of points are connected, and the adjacency graph can not provide any information about the data location. So we get the messy plot (f).



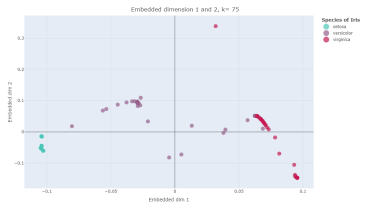
(a) k=10



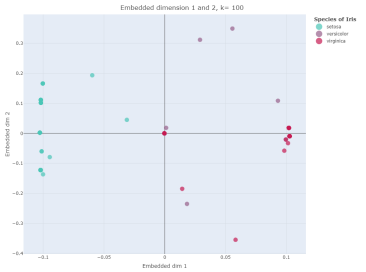
(b) k=30



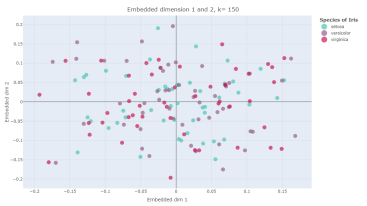
(c) k=50



(d) k=75



(e) k=100



(f) k=150

4.3 Brown Corpus data

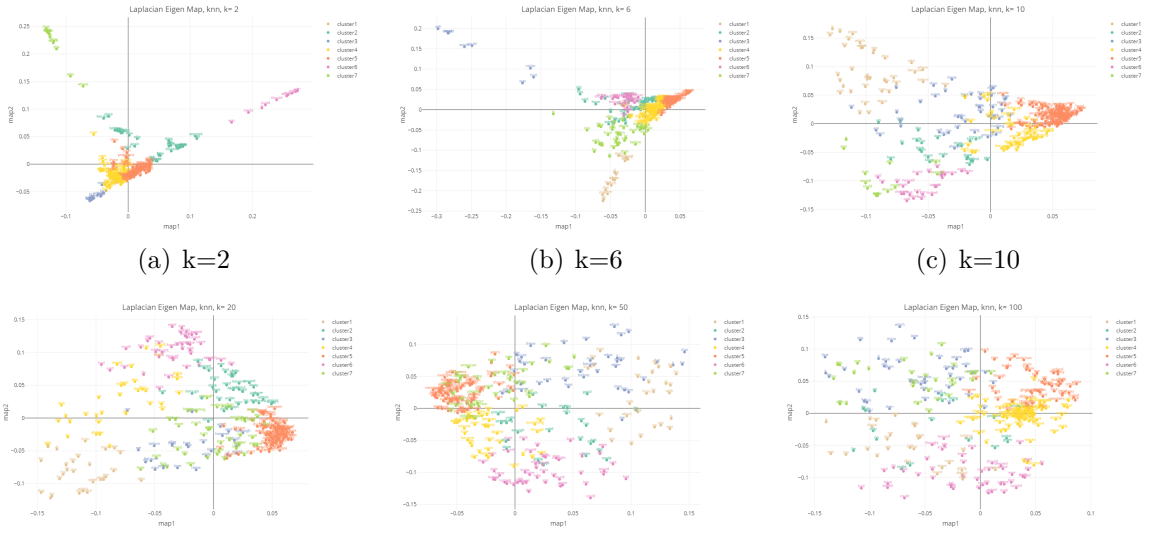
4.3.1 A linguistic sample

Brown Corpus is a famous English corpus, with a variety of text sources and a total volume of more than one million words. Following Belkin and Niyogi (2002), we use Brown Corpus to construct a dataset of contextual vector representations of the top 300 words in terms of frequency of co-occurrence, and apply the LE method to the dataset to reduce dimensionality. Here comes the data construction procedures.

1. After excluding the stop words - the, of, and, to, in, a, an, I - 300 words with the highest frequency of occurrence in the corpus are selected and denoted as T300.
2. For each word in T300, find the position in the corpus where it occurs.
3. For each position, check two words to its left and right hand side, respectively, and see if there are any words in T300.
4. Let i,j,k be 3 words in T300. Each time word j is observed as the first or second nearest neighbor in the left side of word i, add 1 count to (i, j) of the data matrix. Each time word k is observed as the first and second nearest neighbor in the right side of word i, add 1 count to $(i, 300 + k)$ of the dataset. There are 600 columns in total. The first 300 columns represent the left context words of a certain word, and the last 300 columns represent the right context words of a certain word.

4.3.2 Experiment 1: Using knn, comparing different k

When k is set too small, the algorithm may only focus on a narrow neighborhood of the data points, making it hard to capture the geometric structure of the manifold. When k is set too large, the algorithm may try to preserve global distances, and would not be able to extract the structure of low dimensional manifold. Therefore, it is important to try, compare and find the best k setting for a certain dataset.



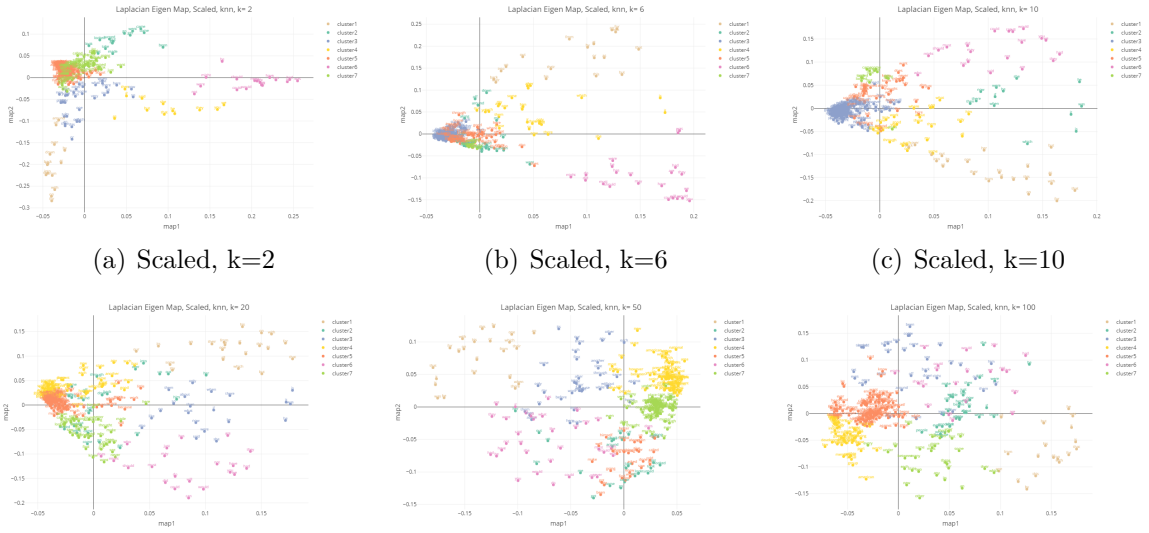
(a) k=2 (b) k=6 (c) k=10

(d) k=20 (e) k=50 (f) k=100

Since the objective function penalises the distance between two connected vertices, it can be imagined that there are "gravity" between connected vertices, which prevents them from getting far away. When k is small, each data point is only "pulled" by a few neighbours, so it can maintain the geometrical shape of "sequentially connected" on a one-dimensional curve. When k is large, each data point has a bunch of edges and will be pulled by the global data points. Therefore, the data points are distributed evenly in the two-dimensional plane. Therefore, we make experiment 3 to try the weighted edge.

4.3.3 Experiment 2: Comparing the result of scaling and not scaling.

Generally speaking, data are often standardised in pre-process procedure to eliminate the effect caused by the different scales of the attributes. Then, researchers could compare the effect of different variables fairly. However, in our experiments, all 600 attributes are of the same category: they are counts of co-occurrences of the words represented by rows and columns. Although the magnitudes are different, the values of each scale are not completely incomparable. Therefore, standardisation by columns is not a natural must. In Experiment 2, we choose the data after column-wise standardisation and keep the other parameter settings the same as in Experiment 1.



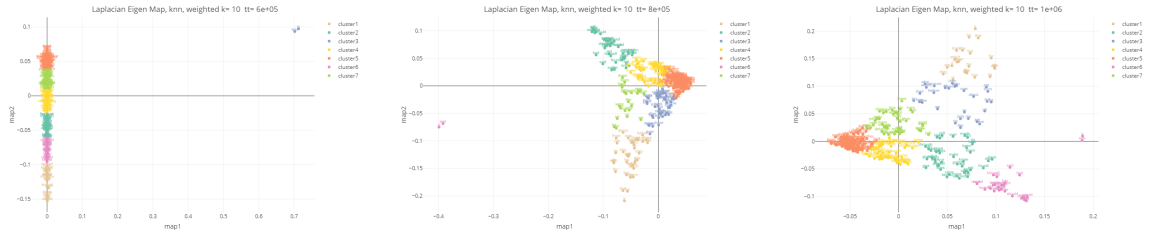
(a) Scaled, k=2 (b) Scaled, k=6 (c) Scaled, k=10

(d) Scaled, k=20 (e) Scaled, k=50 (f) Scaled, k=100

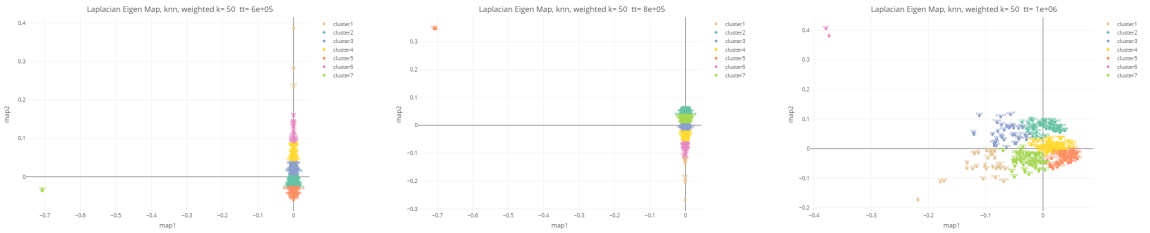
After standardisation, smaller k reveal a more differentiated point layout. If the 2D presentation in experiment 1 appears in a broom shape, the layout in experiment 2 appears like a snail shape. The similarity is that most of the data points are still centred around the origin, i.e. the area of the broomstick or the snail's head. The difference is that in Experiment 1, the dispersed points are roughly distributed on a nearly parallel curve, with a sense of sequentially connection. In Experiment 2, the dispersed points are distributed around the circle, and the points in the middle of the square are quite sparse.

4.3.4 Experiment 3: Using weighted matrix and trying different parameters t.

When the number of connected edges is large, heat kernel can distinguish the importance of edges and balance the emphasis on locality and global distances. Heat kernel is useful when the distances between points in the original space are not too large. When there are outlying clusters in the original space (the distance between clusters is much larger than the distance within clusters), the selection of heat kernel parameter t will be a dilemma: if t is set too large, all edges within clusters will be close to one; if the t is set too small, the inter-cluster edge weights will be close to 0, and the whole map is approximately divided into two or more connected components.



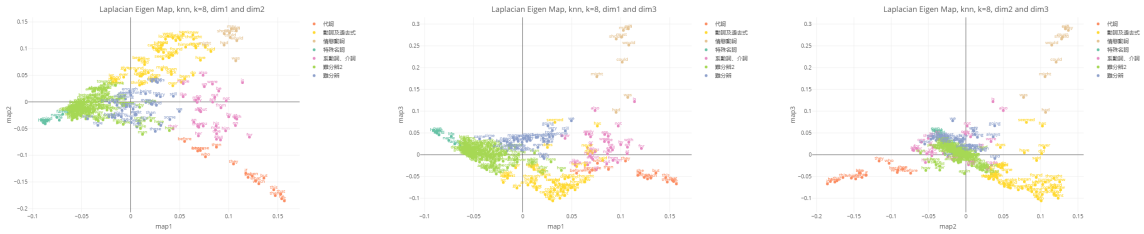
(a) k=10,t=6e5 (b) k=10,t=8e5 (c) k=10,t=1e6



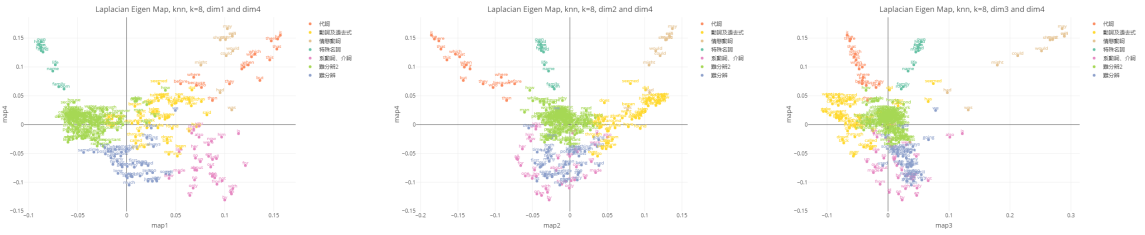
(d) k=50,t=6e5 (e) k=50,t=8e5 (f) k=50,t=1e6

4.3.5 Experiment 4: Observing from different perspectives.

As mentioned above, the first and second eigen maps may be affected by the connected components and contain little information. Sometimes, the information contained in the third and fourth eigen maps may have better interpretability than the results of the first and second eigen maps. It is necessary to extend our attention to the combination of different dimensions and examine them to obtain a low-dimensional representation that is easy to understand and consistent with true clustering labels.

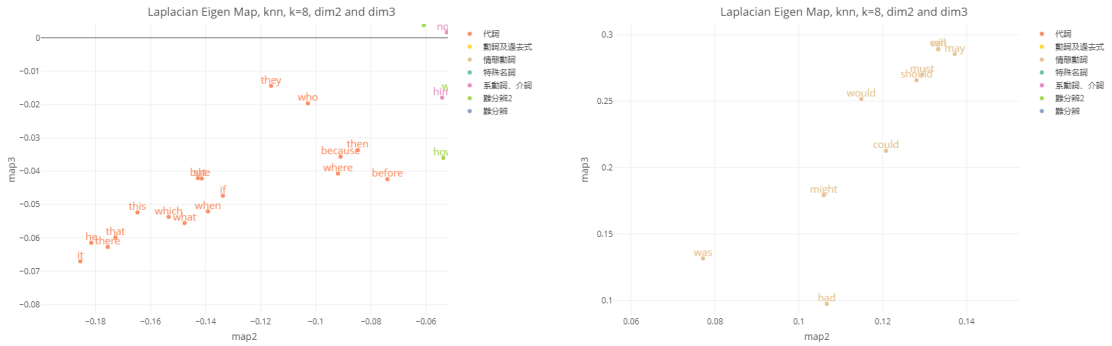


(a) Dimension 1 and 2. (b) Dimension 1 and 3. (c) Dimension 2 and 3.

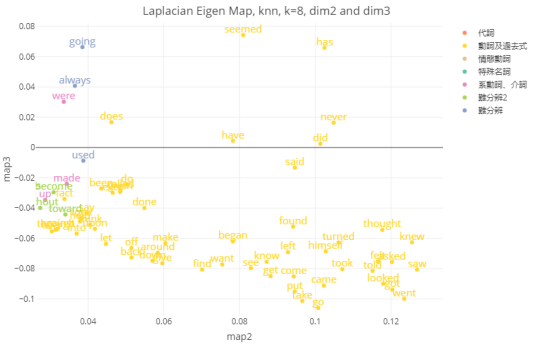


(d) Dimension 1 and 4. (e) Dimension 2 and 4. (f) Dimension 3 and 4.

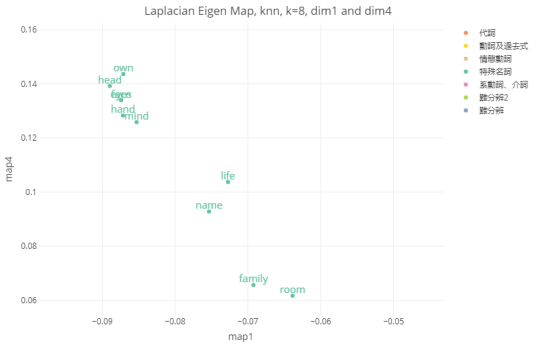
Among the 6 scatter plots of dimension pairs, pair $(2, 3)$, $(2, 4)$ turn out to be most elegant. In sub figure (c), orange, brown and yellow clusters are separated clearly, while in sub figure (d), cyan, green, violet and pink clusters are separated clearly. Here comes the zoomed in pictures of these clusters, and in 5 of 7 clusters, a cluster name could be generalized according to the dominant words' category.



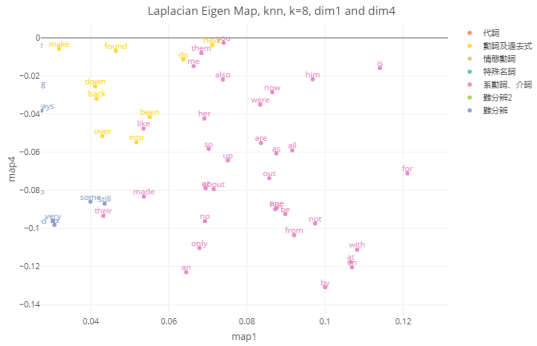
(a) Orange. (b) Brown.



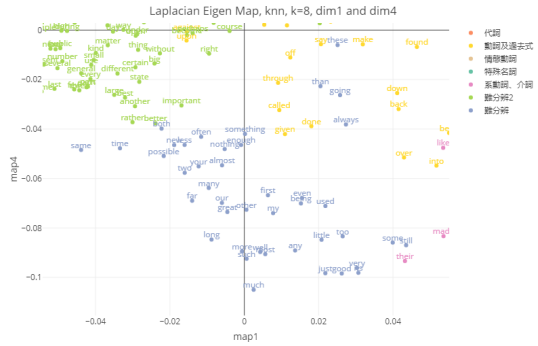
(c) Yellow.



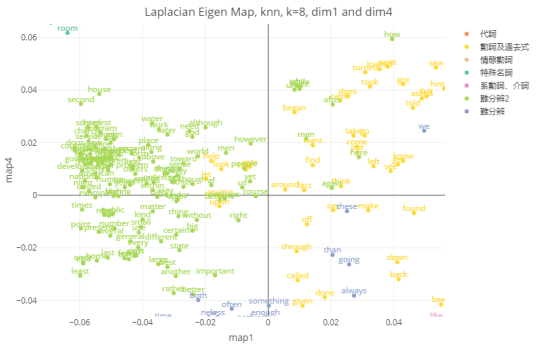
(d) Cyan.



(a) Pink.

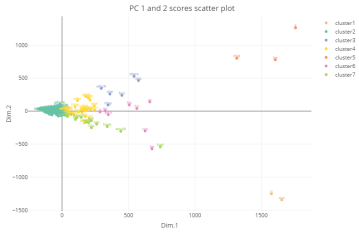


(b) Violet.

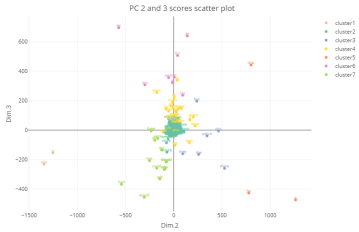


(c) Green.

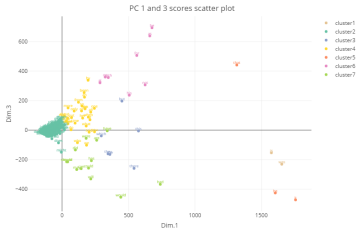
4.3.6 Experiment 5: Comparison with PCA



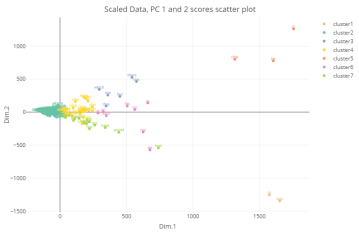
(a) Not scaled, dim (1,2).



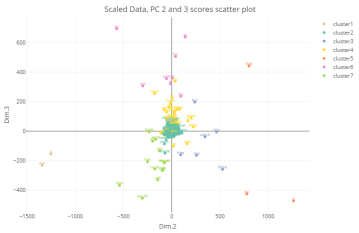
(b) Not scaled, dim (2,3).



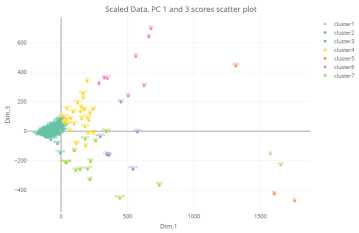
(c) Not scaled, dim (1,3).



(d) Scaled, dim (1,2).



(e) Scaled, dim (1,3).



(f) Scaled, dim (2, 3).

PCA method does not work well. Maybe there does not exist a linear combination of variables that could categorized words data points properly.

Chapter 5

Discussion

In this paper, we introduce Laplacian Eigenmap method as a simple and effective dimensionality reduction method. It has several properties:

1. The computation procedures after constructing the map is relatively simple. The time costing procedure is to calculate distances between data pairs to find nearest neighbors.

2. LE method preserves locality, and is relatively insensitive to outliers and noises.

3. A by-product of the LE method is its clustering property. When the data is mapping to a low-dimensional plane, the original clustering relationships are easily revealed. This principle is similar to the **spectral clustering methods** developed in machine learning and computer vision. However, not all datasets can be divided into meaningful clusters.

4. The choice of t becomes more important when N is large. This is because if a point is connected to most of the points on the graph, it is necessary to associate these connections with a degree of closeness, so as to distinguish the really close and distant points and preserve locality.

Chapter 6

Reference

Belkin, Mikhail, and Partha Niyogi. "Laplacian eigenmaps and spectral techniques for embedding and clustering." Advances in neural information processing systems 14 (2001).

Belkin, Mikhail, and Partha Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation." Neural computation 15, no. 6 (2003): 1373-1396.

Ghojogh, Benyamin, Ali Ghodsi, Fakhri Karray, and Mark Crowley. "Laplacian-based dimensionality reduction including spectral clustering, Laplacian eigenmap, locality preserving projection, graph embedding, and diffusion map: Tutorial and survey." arXiv preprint arXiv:2106.02154 (2021).

Tai, Mariko, Mineichi Kudo, Akira Tanaka, Hideyuki Imai, and Keigo Kimura. "Kernelized supervised laplacian eigenmap for visualization and classification of multi-label data." Pattern Recognition 123 (2022): 108399.

Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, Surya Ganguli. "A theory of multineuronal dimensionality, dynamics and measurement" . bioRxiv 214262; doi: https://doi.org/10.1101/214262

Joram Keijser,Low-dimensional manifolds in neuroscience and evolution,https://joramkeijser.github.io/2022/02/04/manifolds.html.

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-int

erested-in-mathematical-details-or-technical-applications-Im-trying-to-grasp-what-a-laplacian-matrix-actually-represents-and-what-aspects-of-a-graph-it-makes-accessible/answer/Muni-Sreenivas-Pydi.

Guangliang Chen. "Laplacian Eigenmaps." https://www.sjsu.edu/faculty/guangliang.chen/Math253S20/lec12laplacian.pdf.

Juan Orduz. "On Laplacian Eigenmaps for Dimensionality Reduction."https://www.youtube.com/watch?v=U31TIICsHiA"LaplacianEigenmaps."https://www.cs.cmu.edu/~aarti/Class/10701/slides/Lecture21_1.pdf

Chapter 7

Appendix

The algebraic multiplicity of the eigenvalue 0 of matrix L is exactly 1 if the graph is fully 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.

$\bowtie$

Lemma. *For a graph with k connected components, the dimensionality of the eigen space of the graph Laplacian corresponding to the eigen value* 0 *is at least k.*

Proof. Now, consider the graph with k connected components, by adjusting the rows and columns, we can get a block diagnonal Laplacian.

$$L = \begin{bmatrix} L_1 & 0 & ... & 0 \\ 0 & L_2 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & L_k \end{bmatrix}$$

According to **Lemma 1**, the eigen space of a fully connected graph's Laplacian is of 1 dimension, and the only unit eigenvector is an scaled all one vector 1_n. Then, for $L_i \in \mathbf{R}^{m_i \times m_i}$, we have eigen vectors l_{m_i}'s such that

$$1_{m_i}^T L_i 1_{m_i} = 0$$

Then, they can be constructed into an orthogonal basis for the eigen space of the origianl Laplacian corresponding to the eigen value 0.

$$e^{(1)} = \frac{1}{\sqrt{m_1}} \begin{bmatrix} 1 \\ ... \\ 1 \\ 0 \\ ... \\ 0 \\ 0 \\ ... \\ 0 \end{bmatrix}, e^{(i)} = \frac{1}{\sqrt{m_i}} \begin{bmatrix} 0 \\ ... \\ 0 \\ 1 \\ ... \\ 1 \\ 0 \\ ... \\ 0 \end{bmatrix}, e^{(k)} = \frac{1}{\sqrt{m_k}} \begin{bmatrix} 0 \\ ... \\ 0 \\ 0 \\ ... \\ 0 \\ 1 \\ ... \\ 1 \end{bmatrix}$$

$$e^{(i)} \perp e^{(j)}, i \neq j, i, j = 1, 2, ...k$$

$$e^{(i)T} L e^{(i)} = 0$$

$\bowtie$